Proceedings of the Seminar in Computer Science (CS-E4000), Spring 2022

Antti Ylä-Jääski and Sara Ranjbaran

Tutors for seminar topics

Alexander Jung, Anton Debner, Antti Ainamo, Antti Ylä-Jääski, Ashutosh Vaishnav, Chris Brzuska, Esa Vikberg, Jaakko Harjuhahto, Jaakko Harjuhahto, Kamyar Khodamoradi, Lachlan Gunn, Lorenzo Corneo, Miika Komu, Nam Hee Kim, Sanna Suoranta, Stephan Sigg, Ti John, Tian Yu, Tuomas Aura, Vesa Hirvisalo and Wencan Mao

Preface

The *Seminar on Network Security*, *Seminar on Internetworking* and *Seminar on Software Technology and Systems Research* were previously separate Master's level courses in computer science at Aalto University. These seminar courses have now merged into one seminar course. These seminar series have been running continuously since 1995. From the beginning, the principle has been that the students take one semester to perform individual research on an advanced technical or scientific topic, write an article on it, and present it on the seminar day at the end of the semester. The articles are printed as a technical report. The topics are provided by researchers, doctoral students, and experienced IT professionals, usually alumni of the university. The tutors take the main responsibility of guiding each student individually through the research and writing process.

The seminar course gives the students an opportunity to learn deeply about one specific topic. Most of the articles are overviews of the latest research or technology. The students can make their own contributions in the form of a synthesis, analysis, experiments, implementation, or even novel research results. The course gives the participants personal contacts in the research groups at the university. Another goal is that the students will form a habit of looking up the latest literature in any area of technology that they may be working on. Every year, some of the seminar articles lead to Master's thesis projects or joint research publications with the tutors.

Starting from the Fall 2015 semester, we have merged the three courses into one seminar that runs on both semesters. Therefore, the theme of the seminar is broader than before. All the articles address timely issues in security and privacy, networking technologies and software technology.

These seminar courses have been a key part of the Master's studies in several computer-science major subjects at Aalto, and a formative experience for many students. We will try to do our best for this to continue. Above all, we hope that you enjoy this semester's seminar and find the proceedings interesting.

Seminar papers

Securing small networks through automated firewall testing

Matteo Calabrese
matteo.calabrese@aalto.fi

Tutor: Tuomas Aura

Abstract

Due to a shift in the work environment and the steady rise in the number of connected devices, small network environments are becoming ubiquitous. While scenarios vary, a common denominator to most is the lack of professional administration and security auditing. The resulting vast threat surface is only mitigated at the device level, with little consideration of the network itself. Firewalls represent a simple yet effective way of protecting many devices in a centralised way, but ensuring the correctness of their policies can be cumbersome to many. The proposed solution is an application capable of testing a firewall configuration from the point of view of the attacker, as a tool to make securing small office/home office (SOHO) networks more accessible. By simulating both desired and malicious behaviour, a generated report guides the user towards a more reliable firewall configuration.

KEYWORDS: networks, IoT, security, firewall, testing

1 Introduction

In recent years, access to the Internet has become a commodity nearly as vital as electricity or water. With a penetration rate of 97% in Northern

Europe [8], virtually every household and office has an Internet connection available.
Additionally, either because of personal, professional, or mixed use, the number of connected devices has reached yet a new spike, attributable in part to the recent gain in popularity of smart devices and the Internet of Things (IoT) in general. As a result, securing this heterogeneous ensemble of devices has become a challenging, as well as vital, task.

Firewalls have always been the cornerstone of network security, and still play a vital role in securing home, office, or enterprise environments. Due to the continuously changing scenario, and introduction of new technologies and devices, a persistent effort is required in order to keep firewall policies up-to-date and secure. In fact, changes in the policies or rules can easily pose a risk to the effectiveness of the firewall, as human error is by nature unpreventable. Thus, the only way to ensure that a firewall configuration is correct and complete is to put it to the test, and even do so periodically to verify that no tampering occurred.

In order to facilitate the task of securing networks, this paper focuses on the study, the development and the evaluation of an application for automated firewall testing. Starting from a modeled scenario, the configuration of a firewall will be tested by carrying out typical type of attacks. The results will then be used to assess the quality of said configuration and evaluate the performance of the application.

The structure of this paper is as follows:
Section 2 briefly introduces firewalls, their common configuration, and the challenges that arise in situations when policy updates and changes need to be implemented; Section 3 covers the solution and its implementation; finally, Section 4 draws the conclusions.

2 Background

This section contains a brief introduction to firewalls and their usage and defines the modeled scenario used in the development of the application.

2.1 Firewalls

A firewall is a network security device that monitors incoming and outgoing network activity and decides whether to allow or block specific traffic based on a defined set of rules. This device can be either software or

hardware, and it is located on the perimeter of a network, wherever the network connects to the outside world.

There are numerous kinds of firewalls, and their categorisation is based on the functionalities offered. A broad distinction divides them into two large groups: *stateless* and *stateful*. Nonetheless, all of them operate from the internet layer of the TCP/IP model [3] upwards (i.e., the internet layer, the transport layer and the application layer). Stateless firewalls, also referred to as *packet filters*, are the most simple type of firewall, and analyse network traffic by reading each packet independently, i.e., no attempt is made to connect them logically. On the other hand, stateful firewalls handle network activity with the help of context, e.g., if a connection happens to spawn another connection, these are considered related. This approach gives the administrator better control over the traffic, as numerous services require multiple parallel connections to work.
One of the most common amongst firewall software frameworks is *netfilter* [9], and it will be the firewall of choice for the modeled scenario in this study.

2.2 Firewall policies

The behaviour of firewalls is programmed using rules, or policies, that describe what traffic is to be permitted or rejected in the network. Normally, default configurations on home or office firewalls permit all traffic with no restriction whatsoever. Instead, good practice is to block all traffic by default, and then proceed to permit only the desired traffic selectively.
As a result, defining policies can easily become a challenging task when many services are available in the network. Most importantly, implementing an update on a stable and secure configuration might result in unexpected behaviour due to human error. Moreover, the firewall could be the target of an unauthorised modification. For these reasons, periodic probing of the firewall can offer the network administrator a valuable insight into the conditions of the filter at any given time.

2.3 Modeled scenario

Figure 1 shows the chosen scenario that will be used to simulate the network environment. This consists of three networks: the two client sites on the left, containing multiple IoT devices, and one cloud site on the right, containing two servers. The gateways act as firewalls for all three net-

works. In this scenario, the hosts in the client sites need to be able to communicate with the servers in the cloud network, following a client-server paradigm. In order to protect the traffic and encrypt all communications, a Virtual Private Network (VPN) is assumed to be configured.

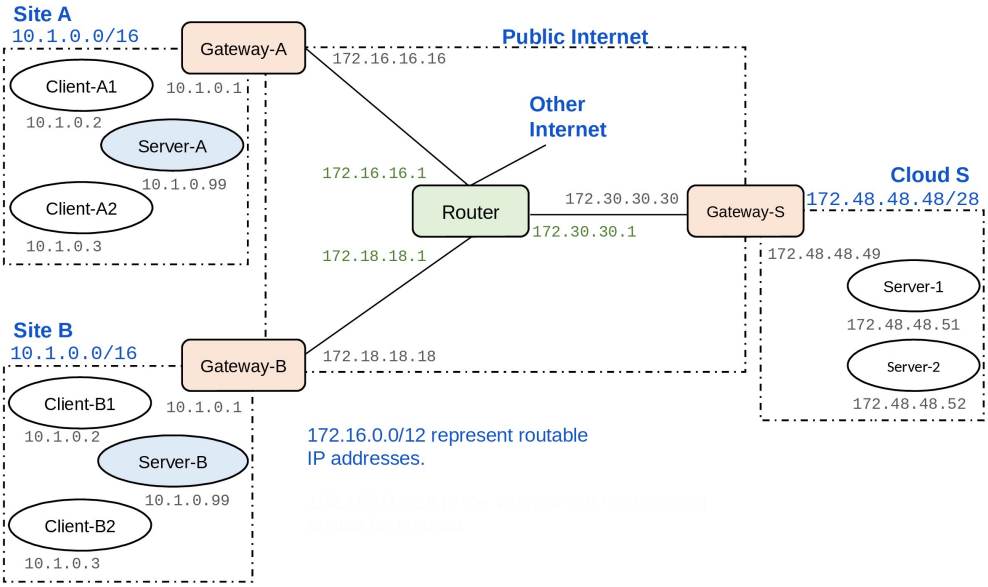


Figure 1. Network topology of the modeled scenario. Source: [1]

The goal of the firewall on the cloud site is to protect the server machines from unwanted traffic coming from the internet while only allowing communications with the client sites. Additionally, the filtering has to ensure that only the clients can initiate a connection towards the servers, and not vice versa. On the other hand, the firewalls on the client sites perform both Network Address Translation (NAT) and filtering. No inbound traffic from any machine other than the servers should be accepted, as well as no outbound traffic which is not precisely addressed to the servers.

The desired final result for the presented environment consists of two communication channels: Site A to Cloud S and Site B to Cloud S. More specifically, isolation between the sites is desired for security reasons, i.e., a client site can only exchange information with one cloud server (and not, for instance, with the server assigned to the other client site, or with the other client site itself). Therefore, Site A will be served by Server-1, and Site B will be server by Server-2.

3 Solution

3.1 Goal

Maintaining firewall policies and ensuring that they are always up to date with the correct security standards can become a challenging task. As services rely increasingly on network communication to function, firewall policies easily grow complex, and incorrect updates to the rule set can undermine its effectiveness. While formal verification of policies is a way to address this problem [6], the goal of this solution is to provide an application that automatically ensures a correct configuration of a firewall by probing it periodically.

3.2 Design and implementation

The application has been developed using the Python [10] programming language, with the support of libraries like Scapy [2], and network tools like Nmap [7]. The software consists of three main modules: the port scanner, the traffic generator and the traffic sniffer.

Firstly, the application is programmed to match the allowed traffic of the network. Secondly, it is deployed in the environment by being installed on all the machines. Finally, both wanted and unwanted traffic is generated to test the configuration of the firewall.
As the applications runs, it gathers information about the traffic it sees on the network. By comparing this data against its configuration, it is able to produce a report describing the quality of the firewall policies.

3.3 Workflow

Programming the application can be done using a configuration file, which describes the services that should be allowed to flow within the network. Automated support is provided for the most common services, to allow for a quicker and simpler set-up process. A more detailed configuration is also possible by providing additional details such as specific protocols, ports and addresses.

At first, the port scanner module probes the configured ports to ensure that the required services are in fact available and reachable. If this step yields a positive result, the traffic generator then starts flooding the network with wanted and unwanted traffic, both from a legitimate node on

the network and from one that simulates an attacker. In order to account for possible compromised machines, similarly to the scenario represented by the Byzantine Generals Problem [5], attacks are also launched from the clients themselves.

At this point, the traffic generator and the traffic sniffer work in tandem to gather insightful data from the network. Both responses received by the traffic generator and data intercepted by the traffic sniffer contribute to the results.
Finally, the application consults the configuration to draw conclusions and shows, through an output file, if the desired behaviour is achieved.

3.4 Test cases and attacks

While the tests that will be executed by the application depend on the provided configuration, this study focuses on the scenario pictured in Figure 1. Firstly, the tests will ensure that the required services are available and reachable by the intended machines. For this purpose, Nmap is used to execute port scanning on the server machines from the client sites. The scanner probes specific ports by sending different types of packets addressed to them, and analyzes the responses to infer what services are running on the target. Nmap supports different kinds of techniques to achieve this, including the TCP SYN scan, the TCP connect scan, the UDP scan [7]. The expected result in this case is for the VPN service to be available, so that clients can establish a tunnel.

Next, the security of the configuration mentioned in Section 2.3 is tested. In order to do this, attacks are carried out following the Dolev-Yao attacker model [4]. More specifically, in addition to making the attacker carry the message (i.e., launch the attack from the central Router), some of the client machines will also simulate malicious behaviour. As a result, three different type of attacks will be executed:

Attacking from the internet
In this scenario, the attacker is an unauthorised machine on the internet. Since there are no other machines outside of the client sites, the Router will impersonate the attacker. To test the configuration of the firewall, the application will attempt to open a TCP connection with both servers. This behaviour is programmed using the Scapy Python library, which offers a high level interface to craft raw packets.

Two results are possible: either the server receives the incoming con-

nection and completes the handshake, or the packets are dropped by the firewall on Gateway-S before even reaching it. If the connection succeeds the firewall policy is not strict enough, and the application will report this in the output.

Attacking from the internet, as a client

The aforementioned attack can be reproduced with a variation: instead of showing its identity, the attacker could craft the attack in such a way that the initiator looks like one of the client site gateways. Since NAT is executed on outbound traffic from these sites, when a client contacts the server, the traffic appears to be generated by the gateways itself. Thus, a server cannot determine the specific client that sent the request, at least judging by the network layer only. However, if the source address of the traffic is changed, the attacker will not be able to receive a possible response from the server. Therefore, the router will be used again to carry out this test. Being on the network, it can intercept all outgoing traffic coming from the cloud site, thus verifying if a response to the attack is returned by the servers.

As previously, two are the possible results: either the packets get erroneously let through and reach the servers, or they get dropped on the path. However, in this case, more than one configuration is tested for correctness. Firstly, the firewall should not accept packets from the clients which are not sent through a VPN (and this can be discerned by the protocol of the traffic). As only the client site gateways are able to open a tunnel to the cloud site, this factor plays an important role in the filtering decision. Moreover, if the VPN is configured correctly on Gateway-S, then by default all traffic originating from the client sites is only forwarded if sent through the tunnel.

Cross site contamination

This last attack aims to verify that the configuration achieves isolation between the client sites. A client simulates compromised behaviour by trying to connect to the server belonging to the other site. This time, the VPN configuration should not let traffic go through the tunnel from the client site gateway. In case the configuration is not strict enough, and the encrypted packets do reach the target, then the router will intercept the response and signal the outcome in the output.

4 Conclusions and future work

Developing a solution that is capable of adapting to heterogeneous and distributed networks is definitely a challenging task, but the recurrence in typical configurations makes it possible to provide a general solution. The solution presented in this study represents a valuable tool when the environment contains a relatively small number of services, but covering a truly exhaustive set of tests may be not possible. For this reason, customising the application to better suit the scenario is vital.

Room of improvement can also be found in the architecture. At the time of writing, the application requires deployment on all the machines in the network to function properly. While the installation process is by design simple, and the overhead of the running software is minimal, there are still cases in which this set-up is not possible. A more targeted and selective configuration could be possible, but would require greater caution.

References

[1] Tuomas Aura and Aleksi Peltonen. CS-E4300 - Network Security. https://github.com/tuomaura/cs-e4300_testbed, 2021. Aalto University School of Science.

[2] Philippe Biondi and the Scapy community. Scapy, packet crafting for python. https://scapy.net/, 2021.

[3] Douglas E. Comer. *Internetworking with TCP/IP, Volume I*. Prentice-Hall Inc, 4th edition, 2000.

[4] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[5] Leslie Lamport, Robert Shostak, and Marshall Pease. *The Byzantine Generals Problem*, page 203–226. Association for Computing Machinery, New York, NY, USA, 2019.

[6] A. X. Liu. Formal verification of firewall policies. In *2008 IEEE International Conference on Communications*, pages 1494–1498, 2008.

[7] Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, 2009.

[8] We Are Social, Hootsuite, and DataReportal. Global internet penetration rate as of april 2021, by region. https://www.statista.com/statistics/269329/penetration-rate-of-the-internet-by-region/.

[9] Netfilter Core Team. The netfilter.org project. https://www.netfilter.org, 2021. Netfilter.org.

[10] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

University ecosystem for student startups: a comparison among European nations

Stefano Facchini
stefano.facchini@aalto.fi

Tutor: Antti Ainamo

Abstract

With the evolution of the technological sector, the number of startups had a noticeable growth, and, with it, universities and institutions adapted themselves to provide notions, courses, and support to these kinds of projects. In this paper, we give an overview of startup ecosystems, including their actors and their flow of interactions, in different universities of the European landscape. This paper outlines similarities and differences among these universities, and it analyzes which factors might contribute the most to inspiring students and in startup formations.

KEYWORDS: startup, university, entrepreneurship, student, accelerator

1 Introduction

In the last years, the situation regarding tech companies has changed significantly, moving from a few big companies with several employees to the current state where more and more startups are emerging and trying to enter the market. This is mostly due to the digitalization of every social aspect, from casual interactions to office meetings, as well as banking, entertainment, and government services. Following this trend, in the last 90 years, several tools such as university courses, events, and accelera-

tors were created to help, support, and sustain potential ideas to grow into well-defined projects and drive them into the market [1].
In this paper, we focus on the university environment, which is known for the high number of ideas that are generated in this context, but also for the low experience of the involved actors [2]. For this reason, these tools have emerged into the university context, to help students with legal, financial and personal support if they wanted to approach the startup world.
While this trend was followed by most of the European countries, each nation has a different social, economical and political situation, and hence the location of the university is relevant in the creation of startups by students [3] [4]. This paper focuses on analyzing some European universities and defining how different structures, locations, and other characteristics might have different impacts on how startups are assisted and led to market.

2 How university relates with startups

Based on the description given by Ainamo et al. [1], the students' startups can be described with the concept of "Platform of trust". The idea is that different individuals or entities get together and interact, more or less frequently, to reach a status of symbiosis where every involved part works for its objective and gains some kind of benefits from others. An ecosystem of students, mentors, professors, and investors is made: students come up with ideas and take the central role in their project, aiming to bring them to the market; mentors and professors offer their experience and time to help students reach success, and get rewarded when a startup makes some progress; investors offer the financial legal support when a startup wants to grow and proceed to the next step while gaining part of the shares when the project becomes relevant. Additionally, other initiatives and entities are formed and enter the ecosystem, such as dedicated degree programmes and external extracurricular organizations.
A key role is held by university incubators and accelerators: although some differences can be narrowed down, we will use for simplicity these two terms as synonyms. These structures are typically a sub-unity of the university that supports potential entrepreneurs with common spaces, consultancy services by tutors, and strategic relationships with founders, investors, and other entrepreneurs, and their close-to-university locations

make it easy for students and professors to have a position here [5]. In the next chapters, we explain how different countries relate to startups, how this general idea is applied to different universities, and how these differences are reflected in the interactions and events that occur around startups.

3 Startups overview in different countries

Before focusing on some specific universities and their support to startups, it's worth contextualizing the startup situation within the European Landscape, as it might differ from other regions such as America and Asia-Pacific (APAC) areas. In Europe, data has been collected by Kollmann et al. in the European Startup Monitor (ESM) project [6].

First of all, startups "are defined by three characteristics: startups are younger than 10 years. Startups feature (highly) innovative technologies and/or business models. Startups have (strive for) significant employee and/or sales growth" [6]. As shown by [7], the average number of startups per millions of inhabitants in Europe is 190. Here, Estonia claims the record for this metric with 865 startups per 1M citizens, followed by Ireland (666) and Denmark (573). The three countries we will focus on are ranked fourth place (Finland, 525), fifth (The Netherlands, 507), and fourteenth (Italy, 234).

Additionally, it's remarkable to see how startups evaluate their countries' governments in terms of support to them. On a scale from 1 to 6, Finland takes first place with an average score of 5.0 out of 6, while the dutch country scores 3.1 out of 6. Italy is graded 2.4/6, below the Europe Average (2.7/6) [6]. Another metric to t is the average evaluation of universities in these countries in "promoting and communicating entrepreneurial thinking/acting": the European Startup Monitor project reports that the "highest ratings were found in Finland (4.0) [...] and the Netherlands (3.0)", while Italy was ranked with a lower score (2.0) [6].

These metrics can partially explain why, even while we stay within the same continent and even inside the European Union, different countries have such different startups landscape: the cultural and financial conditions of each country, also dictated by their history, influence how much comprehension and support the government gives to startup founders and entrepreneurs.

As shown in Figure 1, The New York Times documented what percent-

age of the GDP consisted of Venture Capital for each country, fetching data from the Organization for Economic Cooperation and Development (OECD). As it can be seen, Finland stood out taking second place when considering only Early Stages (light blue portion of the bars), while the Netherlands and Italy were ranked lower positions.

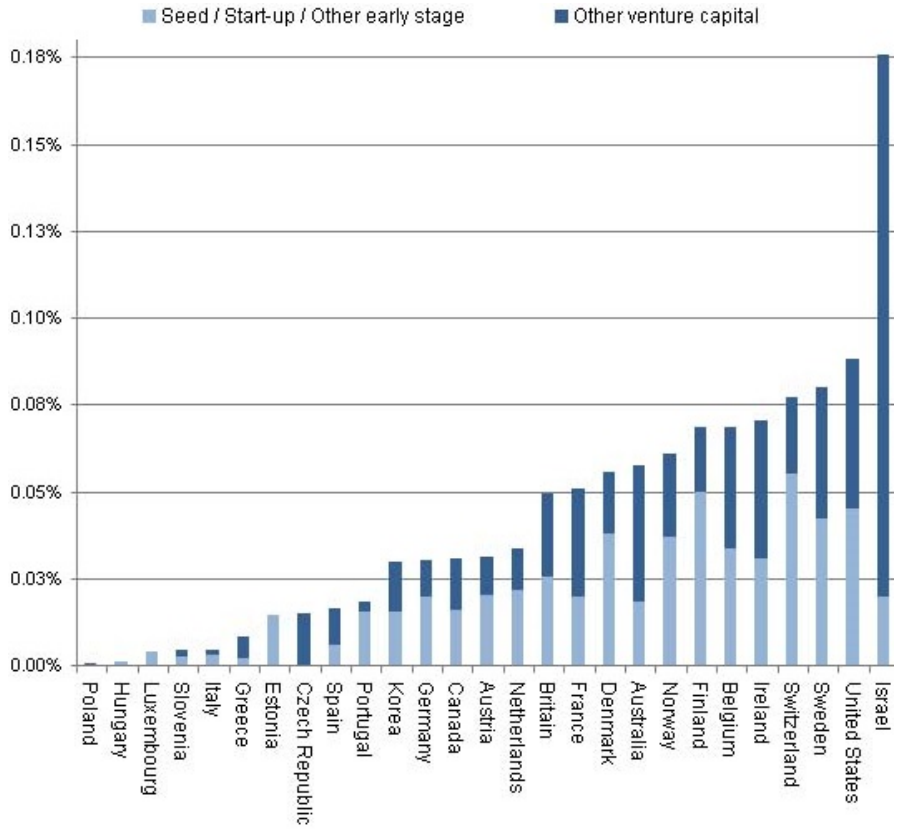


Figure 1 Venture capital as a percentage of GDP, 2009 [8]

4 Startup ecosystem at Aalto University and comparison with others

Being Finland one of the countries with the most startups, as well as being ranked top positions in the previously analyzed metrics, we want to compare the startup environment Aalto University offers against some other European universities. More specifically, we will take into consideration the Netherlands (University of Twente) and Italy (Università degli Studi di Trento). However, to begin with, we will also introduce the EIT Digital Master School, a programme that can be a key element in the startup formations within universities and which is present in all the institutions we are going to consider.

4.1 EIT Digital Master School

The EIT Digital Master School is a two-years study programme, financed by the European Union through the EIT Digital organization, which consists of a double-degree Master for students who want to found their own technological start-up. Partnering with 16 top-rated technical universities in 7 different countries [9], it offers a mixed study plan involving a Master Degree in Computer Science integrated with a Minor in Innovation and Entrepreneurship. With these skills, the students and alumni of EIT Digital have the necessary mindset to ideate, design and grow their business idea into a well-defined startup company. The universities we will inspect later are all partners of EIT Digital and therefore they all offer this Master Programme. As we will describe later, EIT Digital partner universities all show interest toward new technologies and provide support to students in their attempt to found a new company, offering dedicated networking with investors, hackathon competitions and organizing the Innovation Days, where participants can show and pitch their ideas to mentors and investors.

4.2 Aalto University

Aalto, as explained by Ainamo et al., can be taken as an example in the startup context, as "Aalto University has developed an international reputation for innovation and mobilization of students and their ideation and co-creation [...] The Massachusetts Institute of Technology recognized in 2014 Aalto University as a rising star among universities internationally" [1]. According to their website, "up to 100 companies are founded every year in our ecosystem and 50% of Finnish startups that originate from universities come from the Aalto community" [10]. Supporting these statements is the fact Aalto provides common spaces for their students, and they gain an entrepreneurial mindset, ready in case they would like to try to do anything in this context. The main body at Aalto University is the Aalto Entrepreneurial Society, also known as AaltoES: this organization is located at the Startup Sauna Hub and its aim is to provide students with events and contacts to know entrepreneurs and other mentors. In the same building are also hosted some successful companies which can be used as inspirations, such as Junction and Startuplifers [11].
Moreover, Aalto can also benefit from the presence of some accelerators, and some programs associated with them. Kiuas, for instance, is "an ac-

celerator program targeted primarily at Aalto students. Kiuas runs two biannual accelerator programs and has notable alumni companies" [12]. Some popular stories can certify the effectiveness of the entrepreneurial system of Aalto: Rovio is probably one of the most known, but also Supercell and Swappie started in this institute.

4.3 University of Twente

The University of Twente (often called UTwente or simply UT), has been nominated in 2017 as the most entrepreneurial institute of the Netherlands for the fourth time in a row [13]. This award is given following four guidelines which are directly related to the entrepreneurial environment, i.e. the number of spin-off companies, the number of patents applied for, the availability of financing, and the size of science parks. The high scores obtained by UTwente in these categories imply that this institute plays a central role in the startup landscape of the Netherlands.
UTwente boasts the collaboration with Novel-T, an accelerator program whose aim is to assist their students during their startup experience. Novel-T defines itself as an ecosystem composed of four elements: Fundings, Talents, Knowledge & facilities and Network. The University of Twente and Novel-T together furtherly created Incubase, a student incubator dedicated to supporting startup founders with common spaces and resources to continue their growth.
Finally, the University of Twente can take advantage of a science park called Kennispark Twente, an entire neighborhood focused on research companies, startups, and university buildings that work closely with the university itself. Here are established some spin-off companies of UT: the physical closeness to the campus is a key element of the science park so that companies and institutions can share resources and knowledge with the university when needed. Given that there are approximately 50 new students enrolled every year for each university, EIT Digital contributes in the formation of around 350 potential new entrepreneurs every 12 months.

4.4 Università degli Studi di Trento

The University of Trento (UniTN) is one of the main institutions of the Trentino region. While being a relatively small city, Trento features the highest number of innovative startups when this data is normalized on the number of companies [14]. This university is supported by several entities: one of the central ones is CLab, a laboratory of UniTN which creates contacts among different students and professionals to support innovation. CLab offers training on how to open a startup, while also hosting the Innovation Olympics, a real-world challenge that students are required to solve using their creativity. To date, CLab involved more than 2000 students and contributed to the launch of more than 15 startups [15]. The University of Trento also collaborates with different incubators such as Hub Innovazione Trentino (HIT) and Trentino Sviluppo. Although these are not restricted to students, they support young startups offering mentoring sessions, spaces, and foundings. Another structure of the University of Trento that supports innovation and entrepreneurship is the School Of Innovation (SOI). As stated by its rector Paolo Collini, the University of Trento can also rely on the School of Innovation project, which helps in gaining the management and entrepreneur skills for students attending scientific, technological, and social studies [16].

5 Structural analysis

Each one of these institutions, as seen above, offers collaborations with external entities and/or expanded their structures to include entities dedicated to this matter. Given that, it is possible to outline some commonalities among these.

To begin with, all of the aforementioned universities work closely with incubators and allow students to get in touch with them, through events and contacts. These incubators, in turn, offer courses and spaces where the interested students can learn the basics of entrepreneurship and are guided by experts in their adventure.

Another element is the foundation and financing of external bodies that specifically focus on assisting students in unleashing their creativity and transforming their ideas into concrete projects. As the School of Innovation works with the University of Trento, similar projects can be found in several European universities.

Finally, some universities take advantage of dedicated neighbourhoods where they advertise and promote innovation and research. Similar to the science park of the University of Twente, this is a trend being followed by more and more universities.

	Incubators	Programmes	Science Parks	EIT Partner
Aalto University	Kiuas	AaltoES	Campus Area	Y
University of Twente	Incubase	Novel-T	Kennispark	Y
University of Trento	HIT Trentino Sviluppo	School of Innovation	N/A	Y

6 Conclusion

After inspecting some universities and their relationships with startup development, we outlined some characteristics that they feature to enhance the interest in startup founding. More specifically, the universities we analyzed are central figures in their respective countries: for this reason, their traits can be taken into consideration when defining a model to take inspiration from.

Although there is no unique structural organization that is defined to work the best, we can see different elements that attempt to involve the students in generating ideas and pursuing the creation of a company after that. There are several ways universities can exploit to promote entrepreneurial thinking, each one different from the others, but the institutes that are lacking this aspect should evaluate the implementation of one of these to help their members to unlock their potential.

References

[1] Antti Ainamo, Ergo Pikas, and Kari Mikkelä. University ecosystem for student startups: A 'platform of trust'perspective. In *International Conference on Interactive Collaborative Learning*, pages 269–276. Springer, 2020.

[2] Alexander Nolte, Irene-Angelica Chounta, and James D Herbsleb. What happens to all these hackathon projects? identifying factors to promote hackathon project continuation. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW2):1–26, 2020.

[3] Heiko Bergmann, Christian Hundt, and Rolf Sternberg. What makes student entrepreneurs? on the relevance (and irrelevance) of the university and the regional context for student start-ups. *Small business economics*, 47(1):53–76, 2016.

[4] Christopher S Hayter, Roman Lubynsky, and Spiro Maroulis. Who is the academic entrepreneur? the role of graduate students in the development of university spinoffs. *The Journal of Technology Transfer*, 42(6):1237–1254, 2017.

[5] Christos Kolympiris and Peter G Klein. The effects of academic incubators on university innovation. *Strategic Entrepreneurship Journal*, 11(2):145–170, 2017.

[6] Tobias Kollmann, Christoph Stöckmann, Simon Hensellek, and Julia Kensbock. *European startup monitor 2016*. Universität Duisburg-Essen Lehrstuhl für E-Business Graz, 2016.

[7] Number of start-ups per capita by country. https://2020.stateofeuropeantech.com/chart/746-3309. Accessed: 2022-02-21.

[8] Despite economic slump, europe gets more tech start-ups. https://bits.blogs.nytimes.com/201 europes-economy-slumps-a-rise-in-successful-tech-start-ups/. Accessed: 2022-02-25.

[9] Home // eit digital master school. https://masterschool.eitdigital.eu/. Accessed: 2022-04-08.

[10] Aalto university and the startup event slush. https://www.aalto.fi/en/advancing-entrepreneurship-and-innovations-in-aalto-university/aalto-university-and-the-startup. Accessed: 2022-02-25.

[11] Space - startup sauna. startupsauna.com/space/. Accessed: 2022-02-26.

[12] Aalto university students guide the way for finnish startup culture. https://www.enterespoo. story/aalto-university-students-guide-way-finnish-startup-culture. Accessed: 2022-02-25.

[13] The netherlands' most entrepreneurial university. https://www.utwente.nl/en/business/mos entrepreneurial-university. Accessed: 2022-02-26.

[14] Le startup innovative sono 6.745. https://www.truenumbers.it/startup-innovative/. Accessed: 2022-02-26.

[15] Discover clab | clab trento. https://clabtrento.it/en/discover. Accessed: 2022-02-26.

[16] Paolo Collini. L'innovazione didattica nell'università di trento. *INNOVAZIONE DIDATTICA UNIVERSITARIA E STRATEGIE DEGLI ATENEI ITALIANI*, page 69.

CNN over unordered sets

Karol Lasocki
karol.lasocki@aalto.fi

Tutor: Stephan Sigg

Abstract

Convolutional Neural Networks (CNNs) currently dominate the visual image processing field. However, they require a grid-like structure to operate on, such as pixels or voxels. This makes applying CNNs challenging for data without clear ordering, for instance, point clouds coming from sensory measures in robotics, where only an error-prone notion of distances between points exists. This paper reviews methods for utilizing deep learning on such data, either converting it to a form suitable for CNNs or processing the point cloud directly using alternative methods.

KEYWORDS: CNN, Convolution, Neural Network, Point Cloud, Unordered sets, Distance, Metric, Voxelization, Voxel, 3D CNN, Attention, Graph, Geometric, Deep Learning, Robotics, Object Mapping, Gesture Mapping, Density Grid, Occupancy

1 Introduction

In recent years, Convolutional Neural Networks (CNNs) have become a state-of-the-art method for multiple tasks, including the arrival of AlexNET [4] in 2012, dominating the image recognition field, or models such as Lightspeech [7] achieving great quality-to-performance ratio in text gen-

eration. CNNs are a substantial improvement in terms of complexity over classical multi-layer perceptron (MLP) networks, due to a significant reduction of required computations by repeatedly considering only a fixed amount of neighbouring points.

However, one serious limitation of CNNs is that they require the notion of neighbouring points to exist, that is, they assume some kind of ordering of the dataset - whether a 2D Euclidean space as is the case with images, or the 1D time-based order of text or speech. It is not possible to directly apply CNNs to sets of unrelated or loosely related features, often called a point cloud, which do not satisfy this property.

Example applications using this type of data include shape modeling and representation [3], navigation and mapping in robotics [1, 19], and autonomous vehicles [18, 6, 9, 5]. In these increasingly important research areas, data frequently comes from radars and sensors prone to noise and the Doppler effect. Therefore, only a vague idea of which of the points are actually neighbours is present, resulting in point clouds. Moreover, sensory measures are typically non-uniform, with some parts of the space being covered sparsely, while others having dense clusters of points.

This paper presents an overview of current methods of overcoming these difficulties and applying CNNs to cloud point data, their weaknesses and strengths, as well as areas requiring more research.

2 Existing methods

While a comprehensive extraction of information from point cloud data is still an open problem, some ways of approaching it have been introduced in recent years. Sec. 2.1 shows the traditional methods of learning from volumetric grids using voxelization, and their shortcomings. Sec. 2.2 presents examples of learning features directly from the point cloud, aiming to overcome these shortcomings. Finally, Sec. 2.3 demonstrates a geometric deep learning, graph based approach to point neural networks.

2.1 Volumetric grids and voxelization

VoxNet [8] and OctNet [12] represent the original way of coping with point cloud data. They use a volumetric occupancy grid representation together with a 3D CNN. OctNet also aims to exploit the sparsity of the data by hierarchically partitioning the space, focusing on dense regions to enable

deeper neural networks.

Voxelization and VoxNet

A voxel could be thought of as a 3D equivalent of a pixel, a number representing a value in a given point of a 3D space. A **volumetric occupancy grid** represents the data as voxels containing probability values. In a point cloud, the values are frequently sensory binary measures, indicating whether a given point in space has reflected a sensor beam or not. Occupancy grid allows using this information to obtain continuous probabilities: 0 representing a certain absence of an object, 1 meaning the object is surely there, and 0.5 being an unknown state. These probabilities are typically calculated as a function of sensor readings and previous states. This allows for a much more precise mapping than a simple 0/1 representation would.

The VoxNet paper [8] provides three ways of calculating such an occupancy grid. The best performance is achieved by the *density grid*, where each voxel has a density, corresponding to a probability that this voxel would block a sensor beam. The formula for the density at given coordinates *(i, j, k)* and time *t* is given as the following posterior mean

$$\mu_{ijk}^t = \frac{\alpha_{ijk}^t}{\alpha_{ijk}^t + \beta_{ijk}^t}$$

where α and β are defined as follows for measurement z^t at time *t*:

$$\alpha_{ijk}^t = \alpha_{ijk}^{t-1} + z^t$$

$$\beta_{ijk}^t = \beta_{ijk}^{t-1} + (1 - z^t).$$

However, a simplified *hit grid* model also provided surprisingly good results, despite ignoring differences between unknown and free space. In this model, the initial value at time 0 is given as 0, and is updated as:

$$h_{ijk}^t = min(h_{ijk}^{t-1} + z^t, 1)$$

Having obtained this occupancy grid representation, VoxNet utilizes it as the input to a standard CNN, with two 3D convolutional layers, followed by a maximum pooling layer and two fully-connected layers. This allowed VoxNet to beat state-of-the-art solutions in three major 3D object classification datasets at the time of publishing [18]. However, the main drawback is the need of voxelization and operating the CNN on each voxel of the entire 3D space, which raises the complexity of the algorithm to $O(N^3 * T)$. Due to this, VoxNet operates at a fixed voxel grid resolution of 32^3

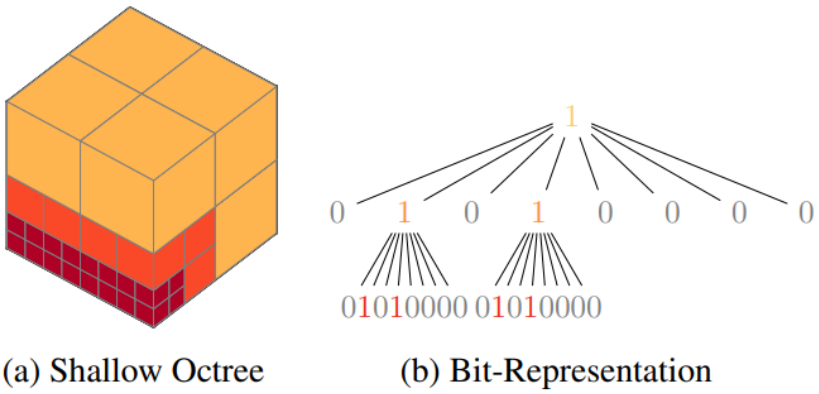


Figure 1. Hybrid grid-octree data structure and its bit-representation. Source: [12]

OctNet

The main assumption of OctNet is that 3D data is often sparse [12]: most of the space is unoccupied, and objects take only a small portion of it. This allows for a development of an adaptive partitioning scheme, focusing computation on important areas.

The partitions are done using octrees, recursively dividing space into octans. Subdivisions occur only for cells containing relevant information, that is, crossing a surface boundary or containing at least one 3D point. To provide fast access to neighbouring points for CNNs without the need of tree traversals, a hybrid grid-octree data structure is used, as shown in Figure 1. By limiting the depth of the tree to a small number, e.g. 3, and placing the resulting shallow trees inside a 3D grid, a compromise between efficiency and ease of access to neighbouring points could be achieved. Such a grid could easily be represented as a bit string, and an octree of depth 3 still reduces the required storage from $8^3 = 512$ values for each voxel at the highest resolution, to a single bit vector.

By rewriting the most common CNN operations, such as convolutions, pooling and unpooling, to work in this structure, significant computational savings can be made. While the exact mathematics behind these modified operations is beyond the scope of this paper, the principle behind the mapping from regular tensors to their octree versions is given in the following paragraph.

Let *O[i, j, k]* be the value of the smallest cell in the grid-octree structure containing the voxel at coordinates *(i, j, k)* (note that O is not injective). Then, the mapping from grid-octree O to tensor T is as follows:

$$oc2ten : T_{ijk} = O[i, j, k]$$

The reverse mapping is given by:

$$ten2oc : O[i, j, k] = pool\_voxels(T)$$

where *pool_voxels* is a function pooling all voxels belonging to the smallest grid-octree cell containing the coordinates (i, j, k), necessary since that cell could contain up to 512 different values.

While using these mappings directly would imply costly conversions, with clever adjustment of the math behind CNN operations, OctNet was able to beat state-of-the-art results in terms of both accuracy and runtime. Both these achievements are owed to reduced computational complexity, which allows for deeper networks, improving results on, e.g., 3D object classification tasks. It outperformed VoxNet's accuracy by approximately 5 percentage points on the ModelNet10 [17] 3D object classification dataset.

Moreover, OctNet's efficiency enables the use of high resolution inputs, ranging up to 256^3 voxels, substantially more than 32^3 supported by VoxNet. Figure 2 visualizes the importance of resolution for object classification. When classifying a bathtub vs a bed, it makes a significant difference; however distinguishing a dresser from a night stand remains a difficult problem regardless of the resolution. Nevertheless, since 3D point cloud labeling usually entails high resolution data, and OctNet proves that depth of the networks matters, it remains an improvement over low-resolution CNNs.

2.2 PointNet and PointNet++

While methods from Sec. 2.1 convert unordered data into ordered and apply usual CNNs, PointNet [10] and PointNet++ [11] are networks directly consuming point clouds. Therefore, they do not require the increased volume of the data that voxelization entails. They can be applied *for applications ranging from object classification, part segmentation, to scene semantic parsing* [10].

PointNet

Consuming unordered data directly with neural networks usually entails transforming the features to values invariant to input permutation. This can be done in various ways, most notable of which are: applying a recursive neural network (RNN) to the training sequences augmented by varying permutations; or using a symmetric function aggregating the in-

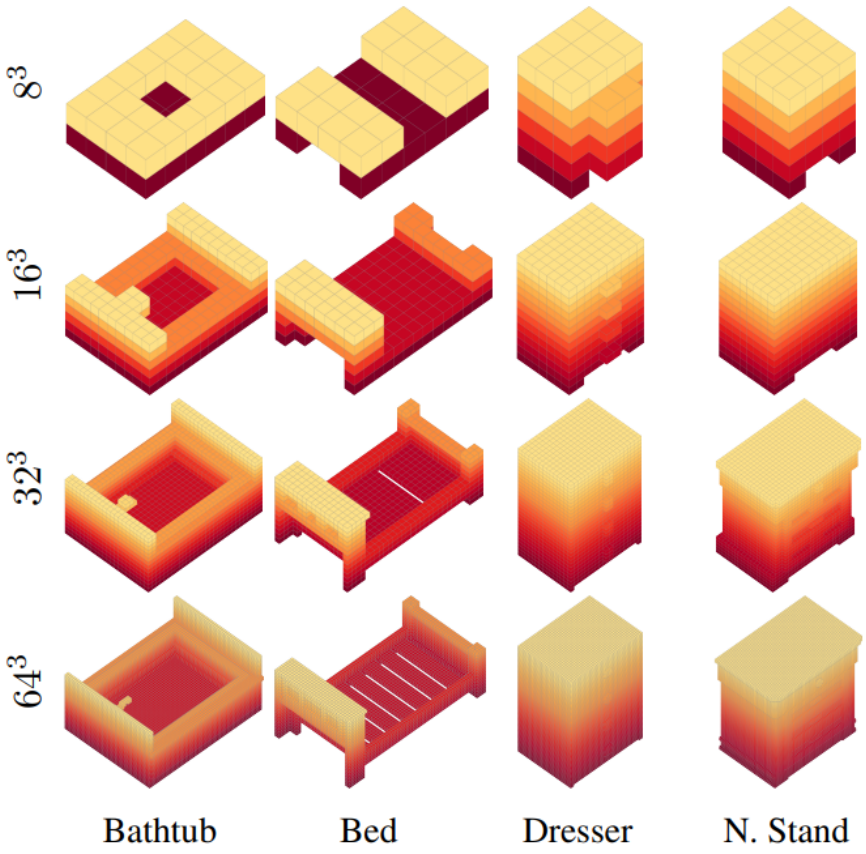


Figure 2. Effect of input resolution on 3D objects. Source: [12]

formation from individual points into a single representation, e.g. addition, multiplication or pooling.

RNNs, while robust to ordering of short sequences, tend to perform poorly on longer sequences of thousands of elements [15], which is typically the case for point clouds. Thus, the authors of PointNet have decided to choose the second approach - they transformed the points using an MLP network, and combined them using a composition of max pooling and a single variable function. A general function defined on a point cloud is then represented as:

$$f(\{x_1, ..., x_n\}) \approx g(h(x_1), ..., h(x_n))$$

where h is approximated by a multi-layer perceptron, and g is pooling composed with a single-variable function, unfortunately not described in detail in the paper.

Using this input transformation, PointNet was able to achieve results on par, and sometimes even exceeding state-of-the-art, including an accuracy improvement from **85.9%** to **89.2%** on the ModelNet40 [17] dataset over VoxNet. As shown in Table 1, max pooling proved to be the best practical choice for the point cloud information aggregation function.

Method	Accuracy
MLP (unsorted input)	24.2
MLP (sorted input)	45.0
LSTM	78.5
Attention sum	83.0
Average pooling	83.8
Max pooling	**87.1**

Table 1. Results of different aggregation functions for point cloud data using PointNet. Source: [10]

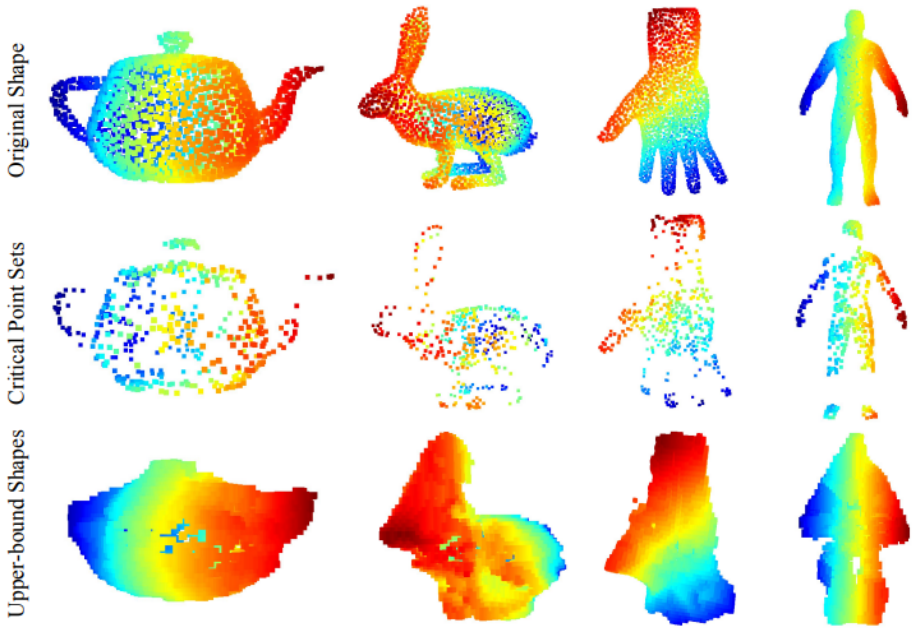


Figure 3. Critical points and upper-bound shapes for unseen objects. Source: [10]

Another advantage of PointNet over VoxNet (potentially also over OctNet, however it was not tested) is robustness to missing data points. *When half of the input points are missing, the performance of VoxNet drops from 86.3% to 46.0% with a 40.3% difference, while PointNet only has a 3.7% performance drop* [10]. This is largely due to the fully neural network oriented architecture of PointNet, enabling it to determine a set of critical points as shown in Figure 3, making it robust to noisy data.

PointNet++

PointNet++ is an improvement over PointNet aiming to utilize the local structure given by the metric of the data space, important for CNNs. It is done by applying PointNet recursively over overlapping partitions of neighbouring points from the input set. The chosen partitioning scheme

is using the farthest point sampling (FPS) algorithm, resulting in neighbourhood balls in an Euclidean space, defined by their centroid and scale. This is in contrary to PointNet, which uses max pooling to aggregate the whole point set. The raw point cloud data is preprocessed using three layers:

Sampling layer. Using the FPS algorithm, this layer finds a subset of points $\{x_1, x_2, ..., x_m\}$ such that any point x_j from this set is the furthest in terms of cumulative metric distance from the preceding points $\{x_1, x_2, ..., x_{j-1}\}$. This results in good coverage of the whole input space. Contrary to standard CNNs which use sliding filters of fixed size independently of the underlying data distribution, this strategy generates data-dependent receptive fields.

Grouping layer. This layer groups all the points from the set into neighbourhoods centered around centroids from the previous layer. It is either done by selecting a fixed number of K nearest neighbours (KNN), or all the points in a given radius up to a maximum of K. Contrary to standard CNN kernel sliding, this way of partitioning using euclidean distance could result in overlapping clusters of varying sizes. However, the neural networks of the following layer are able to cope with differences in number of input points when calculating feature vector representations.

PointNet layer. The final step is using PointNet to obtain dense neighbourhood encodings. Point coordinates are translated, in order to make the vector space of each neighbourhood relative to its centroid.

Including this notion of neighbouring points resulted in an accuracy improvement from **89.2%** to **91.9%** over PointNet on the ModelNet40 dataset. Moreover, the performance drops only by less than 1% when the number of test points is randomly reduced from 1024 to 256, which is essential for real word sensory data, commonly suffering from sampling irregularities. Unfortunately, the paper does not analyse the runtime performance of the algorithm, solely indicating that more optimizations in the form of sharing computation between overlapping neighbourhoods might be needed.

2.3 GAPNet

GAPNet [2] is a novel approach to point cloud data, combining graph attention mechanism with stacked Multi-Layer-Perceptron (MLP) layers. Attention has gained tremendous popularity in areas such as Natural Language Processing, with methods such as Transformer networks [13] dominating the state-of-the-art. Inspired by graph attention networks

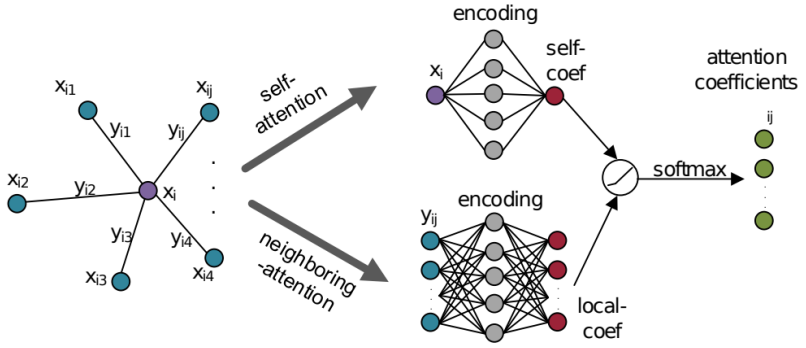


Figure 4. An illustration of how an attention network is applied to the neighbourhood graph. Source: [2]

[14], Chen et al. show that this mechanism has the potential for coping with unordered data as well.

Architecture

The most important part of GAPNet is the **GAPLayer**. Since attention is computed per point, doing it for every other point in the cloud is computationally infeasible. In order to focus on the most important relations, a k-nearest neighbour graph is constructed from the point cloud. In the simplest case of a 3D space without any additional information, such as color, each point $x_i \in R^3$ is a vector of coordinates *(x, y, z)*. The edge features of the k-nearest neighbour graph are defined as

$$y_{ij} = x_i - x_{ij}$$

where x_{ij} is the *jth* neighbour of the point x_i. These features are then used to compute neighbouring attention coefficients, which together with self-attention allow GAPNet to learn the importance of individual points and their relationships with the neighbourhood, as shown in Figure 4.

These self and neighbouring attention networks shown above form a single-head GAPLayer. Its attention coefficients, normalized by a softmax function, are then concatenated to obtain multi-headed attention features as shown in figure 5. Softmax is needed to turn the attention coefficients into a probability distribution, making them comparable across neighbourhoods. Note that the computations can be efficiently parallelized, as there is no correlation between the self and neighbouring attention networks nor between separate attention heads - each of them is learning to attend to the points in its own way, which has a stabilizing effect on the training process and allows for learning finer structural information. In order to increase robustness and improve the quality of predictions, the multi-graph features are then passed through a max pooling function,

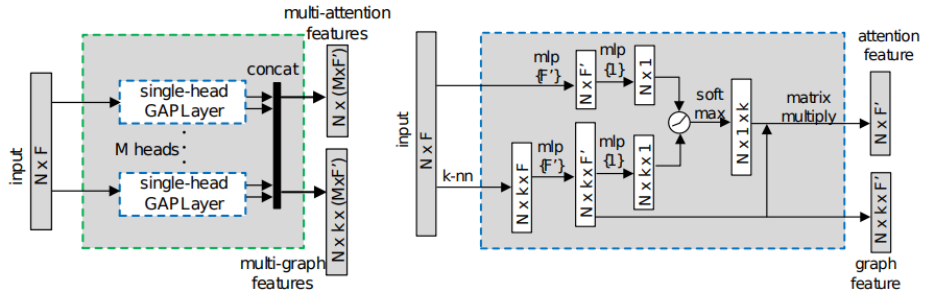


Figure 5. The architecture of a single GAPLayer, taking as input N points of dimension F. Source: [2]

Method	Mean class accuracy	Overall accuracy	Size in MB	Forward time in ms
VoxNet [18]	83.0	85.9	-	-
PointNet [10]	86.0	89.2	41.8	**14.7**
PointNet++ [11]	-	90.7	**19.9**	32.0
DGCNN [16]	**90.2**	92.2	22.1	52.0
GAPNet	89.7	**92.4**	22.9	27.9

Table 2. Results of different models on the ModelNet40 dataset. DGCNN is explained in Section 3. Source: [2]

which selects the most important out of a neighbouring subset of attention heads.

Results

GAPNet trained on an augmented (by rotating, scaling, and randomly jittering the location of the points) version of the ModelNet40 dataset was able to achieve state-of-the-art results, beating PointNet++ by nearly 2 percentage points as shown in Table 2. Although it is not the best in neither size nor computational complexity, it seems like the best trade-off between the two, while also providing the best overall accuracy. As we can see from 6, the architecture of GAPNet is also capable of performing semantic part segmentation, with the majority of the points being classified correctly.

More study is needed on the robustness of GAPNet, as the paper does not mention how well the network deals with missing points, which was claimed to be a key feature of PointNet.

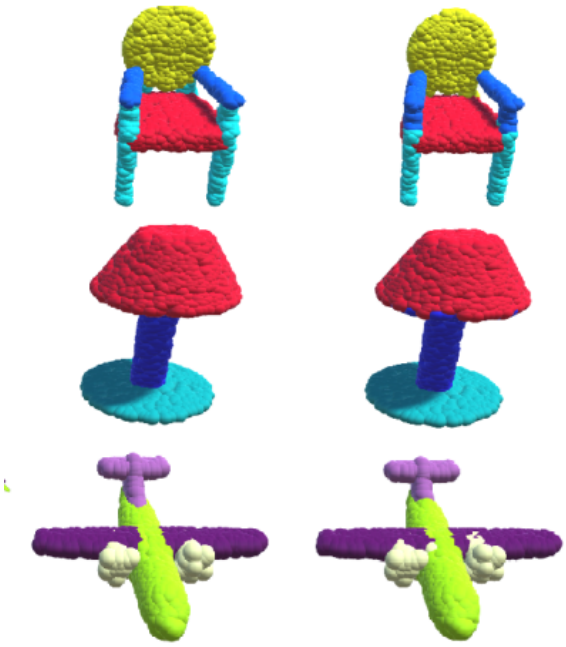


Figure 6. Comparison of the ground truth (left) and GAPNet predictions (right) on a semantic part segmentation task. Source: [2]

3 Conclusion

It is clear that much progress has been made in recent years in regard to point cloud data, as the models improved greatly since the inception of VoxNet in 2015. Despite failing to perform a comprehensive robustness study, it seems that GAPNet is the best out of the presented methods, thanks to its superior performance-to-complexity ratio. Graph based networks and the attention mechanism are increasingly popular in the Deep Learning community, and they have proven very potent in the 3D space as well.

This conclusion is perhaps slightly surprising given the title of this paper, since GAPNet is a network exploiting the local ordering (although its 3D space is not completely ordered - the distance information does not need to be perfect, and the data does not have the grid-like structure of a standard 3D voxel space), and not utilizing CNNs at all, using standard MLP networks instead. DGCNN [16] is a novel method proposing an edge convolution operation over a neighbourhood graph, related to both GAPNet and PointNet (in fact, PointNet is simply the most basic case of the DGCNN method). However, as can be seen in Table 2, DGCNN fails to clearly outperform GAPNet, while being nearly 2 times slower to perform a forward pass, and thus has been left out of this review. Nevertheless,

applying convolutions to graphs or directly to unstructured data is definitely a promising research area that should be carefully explored in the following years.

Another interesting research direction could be utilizing the well performing main idea of OctNet to improve GAPNet as well. Instead of fixing an arbitrary number of k for constructing the neighbourhood graph, it seems reasonable to increase it for important areas, such as shape boundaries or areas with multiple sensory readings. A modified convolution operation could then use this structure to perform efficient computations, possibly learning similar features as the GAPLayer does, eliminating the overhead of calculating attention also for some points the network ultimately does not attend to.

References

[1] Joydeep Biswas and Manuela Veloso. Depth camera based indoor mobile robot localization and navigation. In *2012 IEEE International Conference on Robotics and Automation*, pages 1697–1702, 2012.

[2] Can Chen, Luca Zanotti Fragonara, and Antonios Tsourdos. Gapnet: Graph attention based point neural network for exploiting local feature of point cloud, 2019.

[3] Aleksey Golovinskiy, Vladimir G. Kim, and Thomas A. Funkhouser. Shape-based recognition of 3d point clouds in urban environments. *2009 IEEE 12th International Conference on Computer Vision*, pages 2154–2161, 2009.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[5] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation, 2018.

[6] Zhongze Liu, Huiyan Chen, Huijun Di, Yi Tao, Jian wei Gong, Guang ming Xiong, and Jianyong Qi. Real-time 6d lidar slam in large scale natural terrains for ugv. *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 662–667, 2018.

[7] Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Jinzhu Li, Sheng Zhao, Enhong Chen, and Tie-Yan Liu. Lightspeech: Lightweight and fast text to speech with neural architecture search, 2021.

[8] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[9] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data, 2018.

[10] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.

[11] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017.

[12] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions, 2017.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[15] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets, 2016.

[16] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds, 2018.

[17] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes, 2015.

[18] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection, 2017.

[19] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning, 2016.

Security analysis on challenge-response based biometric authentication methods

Stefano Rumi
stefano.rumichiapella@aalto.fi

Tutor: Sanna Suoranta

Abstract

Biometric authentication protocols allow users to authenticate without passwords, using a unique physical property of the person. This paper will focus on a type of protocols called challenge-response biometric authentication protocols. Their main characteristic is that they emit challenges, which the human body responds to, and these challenges are never reused.

This paper will analyse in depth three different implementations of challenge-response biometric authentication protocols: electrical muscle stimulation authentication, reflexive eye movements authentication and hand-surface vibration authentication.

The main purpose of the paper is to introduce the properties the protocol and compare the security, deployability and usability of the aforementioned systems.

All three systems have advantages and disadvantages in different areas. Therefore, the choice between them must be done considering which properties are more valuable to the system being implemented.

KEYWORDS: Challenge-response, Biometric, authentication, Electrical muscle stimulation (EMS), ElectricAuth, Reflexive Eye Movements, Hand-surface vibration, Velody.

1 Introduction

Due to the current growth of web and mobile applications being developed by different vendors, password management has become a quotidian problem [6]. Passwords rely on human memory, which encourages people to reuse their passwords. According to Capela [2], Nordpass made a survey where they found out that on average, a person has between 70 and 80 passwords, such numbers reveal the importance of developing authentication methods which are not password based. There are three classifications for authentication methods: those based on something that the person knows, on something that the person has, or on something that the person is [7].

Biometric authentication methods verify the identity of a person automatically based on a unique human physical characteristic [10]; thus, on what the person is. Verifying an identity automatically means that the authentication process is done by a computer without asking for something that the person knows; therefore, eradicating the vulnerabilities produced by human memory in password-based methods.

However, most of the existing systems for biometric authentication have design flaws, such as allowing replay attacks [6]. These attacks consist on the biometric data being captured by an attacker and replayed later to gain access to a system identified as another person [9]. This can be achieved by intercepting the communication in an authentication process or by stealing the database where the biometric information is stored. Active biometrics via challenge-response authentication methods prevent these attacks by using different human properties each time that the user authenticates [3]. Therefore, if the attacker obtains the user biometric information from a previous authentication attempt, it cannot be reused.

This article focuses in the use of active biometric authentication methods and their security properties. It analyzes primarily three methods: hand-surface vibration response [5], reflexive eye movement [8] and electrical muscle stimulation [3]. The hand-surface vibration protocol analyses hand responses to vibrations to authenticate the user. The reflexive eye movement protocol consists of analyzing the reaction of eye pupils to light variations. The electrical muscle stimulation protocol verifies the user by applying electrical stimulation on the forearm of the users and analyzes the movements on their fingertips.

This paper is organized as follows. Section 2 provides background infor-

mation on the aforementioned methods. Section 3 introduces the threat model used to analyse the security of these methods. Sections 4, 5 and 6, present each method in detail and analyze their security properties. Sections 7 and 8 compare these methods and section 9 presents the conclusions of the paper.

2 Challenge-response authentication

The research on biometric authentication methods is soaring due to the development of wearables [1]. These are electronic devices that people use in their body, such as smart watches or glasses. They trigger new challenges for authentication, as they do not use the classic computer systems input methods. Authentication relies on sensors that the device has; the majority of them rely on biometric properties of the person.

As the attention these devises rose, companies launched them to production without a deep analysis of the security vulnerabilities that their authentication methods have, allowing attackers to gain access [4] to those devices. However, researchers have made a great effort to develop secure methods to improve them.

Moreover, the effort on improving the biometric sensors, there has been major developments improving how the protocols work. Consequently, dynamic authentication protocols were created. They rely on a unique way the human body responds to a certain challenge.

These protocols have two phases. The enrollment phase is when responses to challenges are recorded. In the authentication phase, one of the prerecorded challenges is yielded and the response is compared to the previous one, if it is accepted after such comparison, the user gains access to the system.

Challenge-response protocols use once a specific challenge and then discard them for future authentication attempts, thus preventing replay attacks. As a consequence, these protocols should have a wide range challenge-response pairs to be used.

One major security characteristic of these protocols, is that responses to a challenge cannot be calculated, even if the attacker has other challenge-response pairs recorded. This property is given by the nonlinearity of the specific protocol implementation, and is studied in detail in the original papers.

The following parts of this section introduce background information for

the specific authentication methods analysed in this paper, which will be described later with more details.

2.1 Electrical muscle stimulation (EMS)

ElectricAuth [3] is an system that authenticates the users by analysing their fingertip movements after applying electrical impulses to their forearm. The fingertip movements are unique due to the human body physiology, this enables the model to be a valid authentication method.

Although the protocol is not restricted to a single measurement method, we will consider the version of the case study presented in the original paper, which uses a VR-headset to measure the fingertip movements.

2.2 Reflexive eye movements

The system by Sluganovic et al. [8] records the users eye-reflexes to certain image inputs when the user registers. As these reflexes are unique due to the physiological eye characteristic, the system later repeats the visual stimuli and authenticates the user. This reflexes are the used with fresh images to identify the user disabling possible replay attacks. Only involuntarily eye movements are analyzed, as the voluntarily ones can enable impersonation attacks.

2.3 Hand-surface vibration response

Velody [5] is a system based on people hands vibrating differently to the same vibration input. This uniqueness is due to the hand's physiological characteristics. Moreover, the human hand is a nonlinear medium for acoustic propagation due to its geometry and composition, which makes the vibration responses difficult to predict.

The case study presented in the original paper uses speakers to produce the vibration and a microphone to record the resulting ones.

3 Threat model

This section introduces a threat model, which is used to analyse the security of all three methods presented in this paper. This threat model is produced by combining the threat models of each specific method and serves to compare the security properties between them. The threats are:

- Impersonation: the attacker imitates what the authentic user would do to authenticate.

- Replay attack: the attacker acquires a previously-used response of the authentic user and replays it to the system.

- Synthesis attack: the attacker calculates a response of a challenge based on previously observed challenge-response pairs.

In addition, some generic security concepts are used to evaluate all three methods. False-negative rate is the rate of mistakenly rejecting a legitimate user. False-positive is a rate which represents how many illegitimate samples are accepted.

Out of those two measures, the equal error rate (EER) can be calculated. Although there is no universal consensus on what a good EER value is, it will be used to compare the security level of the systems. Considering that the lower the value is, the more secure that the protocol is.

4 ElectricAuth

The ElectricAuth [3] system is composed by an array of electrodes and a fingertip movement sensor. The electrodes are attached to the users forearm and give electric impulses to the muscles in the region. Due to such stimulation, the fingers of the user move involuntarily, these movements are captured by the sensor. This involuntary fingertip movements are unique due to the individual properties of human physiology. The wide variety of available sensors to measure the fingertip movements include virtual reality glasses, depth cameras and accelerometers.

This method by Chen et al. [3] sends six impulses to four muscles in seven time gaps, and as the response of the muscle varies depending on the previous state of contraction, it enables 68 million possible challenges to be executed in 1.2 seconds.

Based on a prototype using virtual reality glasses as a movement sensor, studies concluded that it prevents replay attacks completely, synthesis attacks in a 99.75% of the (advanced) tests, and impersonation attacks with an impressive 1.31% EER.

5 Reflexive eye movements

Sluganovic et al. [8] state that eye tracking devices have been massively produced, but little effort has been done to achieve a high level of security. Given that reflexive eye movements can be triggered by visual stimulus, they have developed a challenge-response authentication system based on such property. This system relies on eye-reflexes being unique in each person due its physiology.

The proposed implementation uses a gaze-tracking device to capture the eye movements and a screen to show the visual stimuli. Both of this components can be found in mobile phones, as the frontal camera of such can be used for gaze-tracking.

Studies on the prototype developed in the paper [8] showed a perfect protection against replay and synthesis attacks, but disappointing results against impersonation attacks. Depending on the time spent on authenticating, the EER changes drastically. With a 0.5 stimuli, the EER is above 12%, which is a lower value than in other authentication methods. The EER value drops to 7% when using four seconds of stimuli, which a competitive value against other methods but the usability is strongly deteriorated.

Nevertheless, the paper by Sluganovic et al. [8] is fairly old (2016) for a rapid-evolving industry, and the negative results might be improved with current technology advances on the input devices.

6 Velody

Velody [5] is a system that enables user to authenticate using vibrations on their hands and analysing their responses. Its implementation consists of three main components: the vibration speaker, the vibrating surface and the vibration receiver. The vibration speaker is a simple vibration output like the ones found in mobile devices. The vibrating surface is where the vibration from the speaker flows. It can be of almost any material, as the only requirement is that it must vibrate linearly according to the emitted vibration, which is the case for all regular surfaces without intrinsic movement. The receiver is any device which can read the vibrations on the surface, which could be a simple accelerometer that can be found in the majority of modern mobile phones (built in 2015 or later).

These components work together to read vibration responses in the fol-

lowing way: The vibration speaker emits a vibration challenge, which is transmitted to the user hand by the linearly-vibrating surface. After reaching the hand which is a nonlinear-vibrating medium, the vibration flows modified through the surface until it reaches the receiver.

As explained in previous sections, the vibration after passing through the hand is unique due to the human physiology, and the specific hand geometry and composition.

The received signal is then analysed by a complex pipeline-process which in summary: Normalizes the signal, extracts features, scales those features and uses a classification algorithm to conclude if it is the supposed user or not.

The challenges are designed to be distinguishable among all users and unpredicted from previously used challenges. These security considerations for the challenge design contribute preventing impersonation and synthesis attacks.

Studies based on a prototype of the system produced, conclude that it prevents replay and synthesis attacks completely, and has an EER for impersonation attacks of 5.8%, which is considered low.

7 Security comparison

This paper will use the previously introduced threat model to compare the security properties of the authentication methods. Such model considers impersonation attacks, replay attacks and synthesis attacks.

All three methods protect 100% of replay attacks because they are challenge-response protocols and discard their challenge-response pairs after one usage. This strongly protects users in the case of a data leak, which is one of the major problems of traditional biometric authentication methods, because the leaked challenge-responses can be disabled and new ones can be registered with ease.

Although synthesis attacks are not analysed in the reflexive eye movements paper, we can assume that they are ignored because the own nature of the analysed reflexes are impossible to synthesise. Given such assumption, we end up having two methods, reflexive eye movements and hand-surface vibration, which completely block synthesis attacks. However, electrical muscle stimulation enables them, but its protection against them is really high, as the studies concluded that it protects 99.75% of the attacks using cutting edge synthesis technology.

All three methods show great protection against replay and synthesis attacks; therefore, we will distinguish them using impersonation attacks. The reflexive eye movements method is the one that performs worst, with an unacceptable EER of 12% in 0.5s of stimuli duration or a barely-acceptable EER of 7% in 4s of stimuli, which makes the method unusable for many applications. Better does the hand-surface vibration method perform against impersonation attacks, as it gets an EER of 5.8% in stimuli of only 200ms. The electrical muscle stimulation system perform best here, with an EER as low as 1.31% in 1.2s of stimuli duration.

8 Usability and deployability comparison

Usability and deployability are critical to biometric authentication protocols, it is the main reason for multiple of them to have been deployed ignoring their issues with replay attacks. To enumerate a few, fingerprint and face-recognition authentication systems are deployed in all modern devices because of usability, although they are not as secure as other methods.

One important aspect to consider is which hardware components are used and how accessible they are. The EMS solution has complex hardware (electrodes, and VR-headset or fingertip-accelerometers) which are not to be found in any day-to-day devices. Moreover, the device should have a sleeve to place correctly the electrodes in the forearm, which is difficult to conceive in any widely spread device. The vibration speaker and accelerometers used in the hand-surface vibration solution can be found in every mobile device or tablet. However, as the vibration surface should be at least the size of the hand, it can only be used in large devices, leaving mobile devices out of the scope. The best of the methods regarding these aspects is the reflexive eye movement authentication, as any medium-quality mobile device frontal camera can be used to track the eye movements.

The second aspect that this paper will consider is the time that the authentication process takes. We will consider for this analysis only the time of the challenge stimulation, as the previous and next steps of the authentication process depend highly on the processing power of the device used for the implementation. The reflexive eye movement solution stimuli lasts four seconds for acceptable security properties, this time is considered to be long for devices we are authenticating to constantly, such

as mobile phones, but it can be an acceptable time for devices people authenticate to just a couple of times a day. ElictricAuth's stimuli lasts 1.2s and Velody's one just 200ms, both of them can be considered to be used in any authentication scenario.

9 Conclusions

All three analysed challenge-response systems have similar security properties, the differences between them could be ignored in the majority of the access control scenarios, as they perform sufficiently well in this aspect. However, they have very different usability and deployability properties, which would be decisive upon implementation.

Although ElectricAuth has the best security properties, it is complex to deploy, as the authenticating device should have a sleeve to position the electrodes in the forearm correctly.

Velody has the second best security properties, but it can only be used in devices at least as large as a person's hand.

On the contrary, the reflexive eye movement authentication system could be applied in any device with a medium-quality camera, which makes it easy to deploy, but it is the least secure system of all three.

It is worth to highlight once more that all these methods protect against replay and synthesis attacks, which are attacks that are not even considered in the majority of presently deployed biometric authentication systems. Thus, making them very appealing for further investigation and development for security experts.

References

[1] A. Bianchi and I. Oakley, "Wearable authentication: Trends and opportunities," *it - Information Technology*, vol. 58, 01 2016.

[2] F. Capela, "Self-sovereign identity for the internet of things: A case study on verifiable electric vehicle charging," Ph.D. dissertation, University of Groningen, 2021.

[3] Y. Chen, Z. Yang, R. Abbou, P. Lopes, B. Y. Zhao, and H. Zheng, "User authentication via electrical muscle stimulation," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021.

[4] K. W. Ching and M. M. Singh, "Wearable technology devices security and privacy vulnerability analysis," *International Journal of Network Security & Its Applications*, vol. 8, no. 3, pp. 19–30, 2016.

[5] J. Li, K. Fawaz, and Y. Kim, "Velody: Nonlinear vibration challenge-response for resilient user authentication," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1201–1213.

[6] Z. Rui and Z. Yan, "A survey on biometric authentication: Toward secure and privacy-preserving identification," *IEEE Access*, vol. 7, pp. 5994–6009, 2019.

[7] S. Shunmugam and R. K. Selvakumar, "Electronic transaction authentication — a survey on multimodal biometrics," in *2014 IEEE International Conference on Computational Intelligence and Computing Research*, 2014, pp. 1–4.

[8] I. Sluganovic, M. Roeschlin, K. B. Rasmussen, and I. Martinovic, "Using reflexive eye movements for fast challenge-response authentication," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1056–1067.

[9] D. F. Smith, A. Wiliem, and B. C. Lovell, "Face recognition on consumer devices: Reflections on replay attacks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 4, pp. 736–745, 2015.

[10] J. Wayman, A. Jain, D. Maltoni, and D. Maio, *An Introduction to Biometric Authentication Systems*. London: Springer London, 2005, pp. 1–20.

Multiagent Deep Reinforcement Learning for Video Game Playing

Zixuan Liu
zixuan.liu@aalto.fi

Tutor: Debner Anton

Abstract

Deep reinforcement learning has shown promising performance in many research fields. One of its recent successes is in the area of video-game playing. Recent works have focused on learning beyond single agent settings and explored multiagent scenarios in complex video-game environments, such as StarCraft. The primary motivation of this report is to review the state-of-art multiagent deep reinforcement learning for video-game playing. We review the recent works related to this emerging area, compare the fundamental difference between learning multiagent and single-agent, explore the open challenges as well as point out some possible future directions. We hope this work can give a broad overview of existing literature and resources available for future research.

KEYWORDS: Reinforcement Learning, Deep Learning, Multiagnent Deep Reinforcement Learning, Game AI

1 Introduction

Deep Reinforcement Learning (DRL) has been widely used for controlling agents in complex environments and shown promising performance in various research fields. Among all of them, a considerable amount

of existing DRL methods has been applied in video-game playing. DRL has been successfully adopted to play single agent games (one agent versus another) and a human-level agent can even beat human professionals [27]. However, multiple agents are needed to control and cooperate with each other in a more complex game environment, especially in real-time strategy games, such as StarCraft and GTA. Multiagent Deep Reinforcement Learning (MARL) has recently been used to address the problem but still faces various challenges, include the non-stationary problem [26] and over-generalization [4].

The primary motivation of this report is to review the state-of-art multiagent deep reinforcement learning for video-game playing. We review the recent works related to this emerging area, compare the fundamental difference between learning multiagent and single-agent, explore the open challenges as well as point out some possible future directions. It is important to note that there are many surveys and critiques on MARL [6]. However, this report works in an effort to apply MARL techniques in video-game playing.

The report is structured as follows. Section 2 introduces some basic DRL algorithms used in video-game playing, such as DQN and REINFORCE. Section 3 reviews the DRL methods for multiagent setting and overviews recent research related to this field. Section 4 explores the application of MARL in StarCraft. Section 5 concludes the report.

2 Deep Reinforcement Learning Approaches

Reinforcement learning formalizes the problem of an agent interacting with the environment, which is usually described by a Markov Decision Process (MDP) [22]. In general, reinforcement learning mainly consists of five elements: S denotes the states of the environment, A denotes the actions of the agent, T represents the transition probability from one state S to another state $S^{'}$, $R(s,a)$ defines the immediate reward given to the agent when it performs action a in state s, and γ determines the discount factor that balances between the immediate rewards and future rewards. Thus, an MDP can be denoted by $\langle S, A, T, R, \gamma \rangle$.

Q-learning [34] is one of the most famous algorithms for reinforcement learning. A Q-learning agent updates Q-value as follows:

$$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_{a} Q(S^{'},a) - Q(S,A)] \tag{1}$$

where α is the learning rate. During the iteration, a Q-learning agent which starts in state S, takes an action A as $Q(S, A)$, receives the reward R, and transfers to next state $S^{'}$. The best future state-action pair $\max_a Q(S^{'}, a)$ is selected as the optimal Q function. Q-learning has been proven to be flexible and efficient in solving small reinforce learning cases. However, Q-learning algorithm needs to maintain a Q-table internally. In the case of a large state spaces, the Q-table can be extremely large and the lookup efficiency is prohibitively slow, which limits the application scenarios of Q-learning. In recent years, with the rise of neural networks, reinforcement learning based on deep learning has solved the above problems and become the mainstream, namely, Deep Reinforcement Learning (DRL). Below we review two types of DRL methods: value-based methods and policy gradient methods.

2.1 Value-based Methods

Q-learning can use a deep neural network as a function approximation for the large Q-table [23]. This combination generates one major breakthrough algorithm called Deep Q-Network (DQN) [17]. The Q-Network used in DQN consists of several layers. The input layer receives the last four stacked frames an agent encountered in the game. Then, several convolution layers capture the features from the input frames. At last, the learned features go through some fully-connected layers and the network output the actions that the agent can take. To improve the stability of the learning process, DQN keeps an experience replay (ER) buffer [11], which stores batches of $\langle S, A, R, S^{'} \rangle$ tuples. During the training process, random mini-batches of $\langle S, A, R, S^{'} \rangle$ tuples are sampled from the ER buffer and fed into the Q-Network. In addition to the Q-Network, DQN maintains another copy of the parameters of the Q-Network as the target network. The parameters of the target network are updated according to Q-Network periodically to stabilize the training.

DQN has been improved in many ways. For example, the input of the Q-Network is four consecutive frames because DQN assumes full state observations. Therefore, the performance of DQN declines when given incomplete state observations, such as using only one frame as input. Deep Recurrent Q-Network (DRQN) [5] was proposed to address the partially observations problem. In practice, the penultimate layer of the Deep Recurrent Q-Network is replaced with a Long Short-Term Memory (LSTM) layer [7]. In this setting, DRQN maintains a memory capacity to work

with one input frame. Other improvements include using double estimators with Double DQN [33] and finite state controllers [16].

2.2 Policy Gradient Methods

Value-based methods have been successfully applied in many fields. However, it comes with some disadvantages, such as not being able to deal with continuous action space and stochastic policy problems. While Policy Gradient Methods estimate the probability that one action A is taken at a certain state S given the parameter of the network θ. REINFORCE [35] is one of the basic policy gradient algorithms for such domains. It first samples several possible actions based on the policy and updates the parameter θ using the following equation:

$$\nabla_\theta \sum_A \pi_\theta(S, A) R(S) \tag{2}$$

where $R(S)$ is the discounted cumulative rewards. This equation maximizes the likelihood that the most successful actions should be taken in the future. However, REINFORCE algorithm assumes sampling a complete episode to update the parameters, which is not efficient. Actor-Critic is an algorithm [30] using temporal difference (TD) learning to update the parameter periodically. The Actor-Critic Policy Gradient is composed of two parts: an actor learns policy $\pi_\theta(S, A)$ and a critic estimates the rewards R using $Q(S, A)$ value. The loss function is given by:

$$\nabla_\theta log \pi_\theta(S, A) Q(S, A) \tag{3}$$

In conclusion, we have reviewed some of the famous algorithms about reinforcement learning. However, the list is not exhaustive and more state-of-art techniques can be found in [1].

3 Multiagent Deep Reinforcement Learning (MARL)

In this section, we review the framework of Multiagent Deep Reinforcement Learning (MARL) and the recent research.

Multiagent learning to the same level of performance as a single agent is inherently complex. Because agents not only interact with the environment, they interact with each other at the same time [2]. In general, consider a simple Markov game $\langle S, N, A, T, R \rangle$ as the extension of MDP to multiple agents [12]. Compared to the MDP discussed in Section 2, the

transition T and the reward function R in multiagent settings depend on the actions of all N agents $A = A_1 \times ... \times A_N$, which means, $R = R_1 \times ... \times R_N$ and $T = S \times A_1 \times ... \times A_N$. Let i denote the i-th agent, and $-i = N \setminus \{i\}$ represents the rest of the agents, then the value function $V_i^{\pi}(S)$ is defined as follows:

$$V_i^{\pi}(S) = \sum_{A} \pi(S, A) \sum_{S'} T(S, A_i, A_{-i}, S^{'})[R_i(S, A_i, A_{-i}, S^{'}) + \gamma V_i(S^{'})] \quad (4)$$

where $\pi(S, A) = \prod_j \pi_j(S, A_j)$. The optimal policy is the one that maximize the V function, which depends on the policy of other agents.

$$\begin{aligned} \pi_i^*(S, A_i, \pi_{-i}) &= \arg\max_{\pi_i} V_i^{\pi_i, \pi_{-i}}(S) \\ &= \arg\max_{\pi_i} \sum_{A} \pi(S, A) \sum_{S} T(S, A_i, A_{-i}, S^{'})[R_i(S, A_i, A_{-i}, S^{'}) \\ &+ \gamma V_i^{(\pi_i, \pi_{-i})}(S^{'})] \end{aligned} \quad (5)$$

where $\pi(S, A) = \pi_i(S, A_i)\pi_{-i}(S, A_{-i})$. The equation above shows that the environment is no longer stationary in the multiagent setting since the policy of other agents $\pi_{-i}(S, A_{-i})$ changes over time [10], which leads to some convergence problems of MARL. As described by Littman [13], the convergence can only be guaranteed in adversarial environments, where an optimal policy can be found against a random opponent by minimizing Q-learning [12]. In cooperative environments where all agents coordinate with each other and share the same reward function, special assumptions should be made to obtain optimal convergence. While in other environments, no value-based MARL methods guarantee convergence.

Recent work on MARL has mainly focused on learning communication and cooperation. In practice, agents need to cooperate in a partial observation environment by maximizing their collective utility function. For better cooperation, agents usually share information by sending messages directly through communication protocols [3] or via a shared memory [21]. Reinforced Inter-Agent Learning (RIAL) [3] is an algorithm that uses deep neural networks as a discrete communication channel. The neural network outputs the same Q-value as in other DRL methods. Moreover, it outputs a message used to communicate with other agents. RIAL maintains another network to share parameters with all the agents. In a similar algorithm called Differentiable Inter-Agent Learning (DIAL) [3], messages are discretized and transformed into several communication actions that need to be executed. Instead of using a discrete communication channel, CommNet [29] used a communication channel with a continuous vec-

tor, where each agent receives the summed transmissions of other agents. Since the communication vectors are continuous, it is differentiable and can be trained via back-propagation, which makes the model simple and versatile.

Another technique in learning communication is Memory-driven (MD). Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [15] is an algorithm that uses shared memory for communication. During execution, each agent has access to a specific memory and writes information. In this case, the agent can learn policies that only depend on local information and own observations.

In conclusion, learning communication and cooperation still faces major challenges in MARL field. More discussions can be found in [14].

4 MARL Methods for StarCraft

StarCraft is one of the most popular games in the Real-Time Strategy (RTS) genre. StarCraft consists of several competitions that require complex strategies and multi-agent control techniques, thus providing an ideal platform for multi-agent study and control with different difficulty levels [9]. The goal of StarCraft AI is to solve numerous challenges, including multi-agent collaboration, opponent modeling and adversarial planning [19]. Currently, researches about improving StarCraft AI have made impressive progress and many API and libraries have been developed to support the study, such as Brood War API (BWAPI), TorchCraft [31], μRTS [18] and ELF [32]. However, building an intelligent game AI for the complete StarCraft is still out of reach and many researchers regard multi-agent management as the first step to develop StarCraft AI [24].

Particularly, in this paper, we focus on StarCraft micromanagement tasks [31], which is considered to be one of the most complex games because of the large possible states space. In this task, each agent controls their role to destroy the army of enemies with different terrain conditions. One major challenge of the task is that the parameter spaces grow exponentially when controlling more agents and we cannot tackle the dynamic change in the number of agents.

Multiagent Bidirectionally-Coordinated Network (BiCNet, 2017) [20] formalizes the StarCraft micromanagement tasks as a zero-sum (adversarial environment) Stochastic Game, where the agents collaborate within their teams and compete between teams. The paper considers that both con-

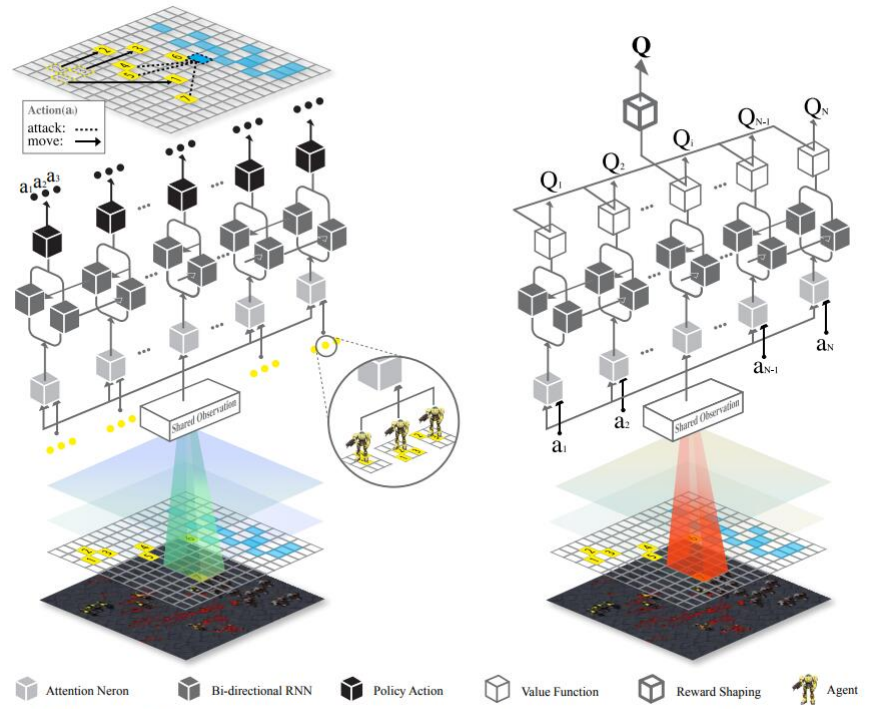
trolled agents and enemy agents have the same continuous action spaces A and B, which reduces the redundancy in large discrete action spaces. In terms of the reward function, the agents within the same team share the same reward and a time-variant reward based on the difference between two time steps is introduced as the global reward:

$$r(S,A,B) = \frac{1}{M}\sum_{j=N+1}^{M}\Delta R_j(S,A,B) - \frac{1}{N}\sum_{i=1}^{N}\Delta R_i(S,A,B) \tag{6}$$

However, the above reward function comes with potential drawbacks, such as it ignores that each agent has its own goal to drive cooperation. To extend this behaviour, each agent maintains a top-K list that records the attribution of other agents currently being interacted:

$$r_i(S,A,B) = \frac{1}{|j|}\sum_{j=N+1\cap top-K(i)}^{M}\Delta R_j(S,A,B) - \frac{1}{|i'|}\sum_{i'=1\cap top-K(i)}^{N}\Delta R_{i'}(S,A,B) \tag{7}$$

A bidirectionally-coordinated net (BiCNet) is proposed to provide communication between different agents. The architecture of the network is illustrated in Figure 1. It is based on bi-directional RNN and consists of two parts: a policy network and a Q-Network.



(a) Multiagent policy networks with grouping (b) Multiagent Q networks with reward shaping

Figure 1. The architecture of Bidirectionally-Coordinated Net (BiCNet).

The bi-directional recurrent mechanism is used as a communication middleware between agents. Experiments were performed with different settings of the StarCraft combats. The results highlighted the fact of

Inputs	CoolDown	HitPoint	OwnSumInfo	OwnMaxInfo
Type	$\in R$	$\in R$	$\in R$	$\in R$
Dimension	1	1	8	8
Inputs	EnemySumInfo	EnemyMaxInfo	TerrainInfo	Action
Type	$\in R$	$\in R$	$\in R$	$\in cat.$
Dimension	8	8	8	9

Table 1. State representation of StarCraft. R denotes real value and $cat.$ denotes the type of input is categorical and one-hot encoded.

a strong correlation between specific rewards and policies, which can be further investigated to learn how policies are shared over the networks.

To further solve the problem of large parameter spaces, Parameter Sharing Multi-agent Gradient Descent Sarsa (PS-MAGDS, 2018) [25] algorithm constructs a special state representation of the input of StarCraft, which is composed of eight parts, as depicted in Table 1. This special representation is independent of the changes in the number of agents, thus is efficient and can be generalized to other combat games. Moreover, the input is further divided into three components: the current step state information, the last step state information and the last step cation. The architecture of the learning model of one agent in StarCraft is shown in Figure 2. The three parts of state representation are the input of the neural network. Then the network outputs the probabilities of turning to 8 directions with a fixed distance, including Up, Down, Left, Right, Upper-left, Upper-right, Lower-left and Lower-right, and whether attacking. If choosing to attack, the agent will fire on the enemy agents.

To train the model, PS-MAGDS algorithm extends the traditional SARSA methods to multiple agent settings. The proposed method shares the parameters of the policy network among all the agents and uses eligibility traces [28] to tackle the delayed rewards. The method updates the policy network using gradient-based method as follows:

$$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}; \theta_t) - Q(S_t, A_t; \theta_t) \tag{8}$$

$$\theta_{t+1} = \theta_t + \alpha \delta_t e_t \tag{9}$$

$$e_t = \gamma \lambda e_{t-1} + \Delta_{\theta_t} Q(S_t, A_t; \theta_t), e_0 = 0 \tag{10}$$

where e_t refers to the eligibility traces at time t and λ is a factor that determines the weight of each backup in e_t.

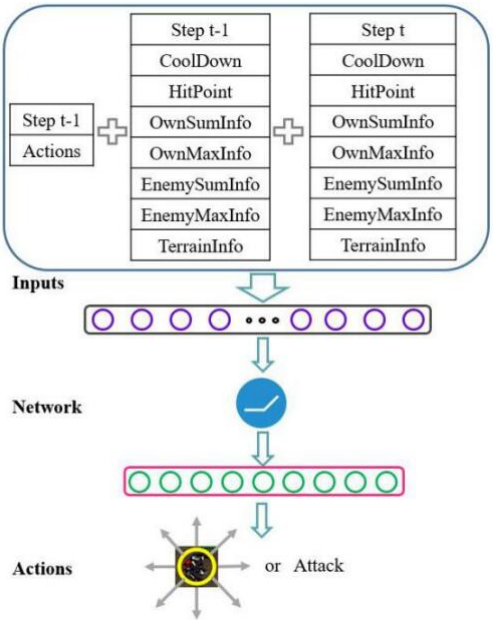


Figure 2. The architecture of the learning model of one agent in StarCraft.

Experiments have been conducted in both small and large scale micromanagement scenarios in StarCraft and have shown better performance over other baseline algorithms. The cooperative behaviors between different agents can be successfully learned by sharing the parameters of the policy network and the proposed method defeats the built-in AI with 100% win rates in small scale scenarios. However, exist some coordination strategies can not be effectively learned by the agents and more intelligent algorithms should be considered. Moreover, it is not sufficient to solve the delayed reward only using eligibility traces. Advanced methods, such as hierarchical reinforcement learning [8], should be further explored.

5 Conclusion

Deep reinforcement learning has shown promising performance in many research fields. One of its recent successes is in the area of video-game playing. A natural next step is to investigate multiagent deep reinforcement learning scenarios in complex strategy game environments, such as StarCraft. However, learning in multiagent settings can be intrinsically difficult since the environment is non-stationary, which leads to various convergent issues.

This paper provided an overview of multiagent deep reinforcement learning algorithms applied in video-game playing. First, we reviewed the basic concepts and classical methods of deep reinforcement learning. Then,

we focused on the recent works of multiagent deep reinforcement learning. We provided the general terms of MARL, exemplified the reason that MARL is intrinsically complex and pointed out some techniques for agents learning communication and cooperation in multiagent settings. In addition, we explored the performance of two specific algorithms applied in StarCraft: Multiagent Bidirectionally-Coordinated Network (BiCNet) and Parameter Sharing Multi-agent Gradient Descent Sarsa (PS-MAGDS). We also reflected on the potential challenges in each of the algorithms.

Recent work demonstrates that some practical problems hinder us to build MARL methods in complex video-game environments. However, some reviewed methods can achieve notable performance and we hope this work can give a broad overview of existing literature and resources available for future research.

References

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[2] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[3] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

[4] Nancy Fulda and D. Ventura. Predicting and preventing coordination problems in cooperative q-learning systems. In *IJCAI*, 2007.

[5] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015.

[6] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33:750 – 797, 2019.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[9] Raúl Lara-Cabrera, Carlos Cotta, and Antonio J Fernández-Leiva. A review of computational intelligence in rts games. In *2013 IEEE Symposium on*

Foundations of Computational Intelligence (FOCI), pages 114–121. IEEE, 2013.

[10] Guillaume J Laurent, Laëtitia Matignon, Le Fort-Piat, et al. The world of independent learners is not markovian. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 15(1):55–64, 2011.

[11] Long Ji Lin. Programming robots using reinforcement learning and teaching. In *AAAI*, pages 781–786, 1991.

[12] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[13] Michael L Littman. Value-function reinforcement learning in markov games. *Cognitive systems research*, 2(1):55–66, 2001.

[14] Ryan Lowe, Jakob Foerster, Y-Lan Boureau, Joelle Pineau, and Yann Dauphin. On the pitfalls of measuring emergent communication. *arXiv preprint arXiv:1903.05168*, 2019.

[15] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[16] Nicolas Meuleau, Leonid Peshkin, Kee-Eung Kim, and Leslie Pack Kaelbling. Learning finite-state controllers for partially observable environments. *arXiv preprint arXiv:1301.6721*, 2013.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[18] Santiago Ontanón. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2013.

[19] Santiago Ontanón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. A survey of real-time strategy game ai research and competition in starcraft. *IEEE Transactions on Computational Intelligence and AI in games*, 5(4):293–311, 2013.

[20] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

[21] Emanuele Pesce and Giovanni Montana. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. *Machine Learning*, 109(9):1727–1747, 2020.

[22] Martin L. Puterman. Markov decision processes: Discrete stochastic dynamic programming. In *Wiley Series in Probability and Statistics*, 1994.

[23] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pages 317–328. Springer, 2005.

[24] Kun Shao, Yuanheng Zhu, and Dongbin Zhao. Cooperative reinforcement learning for multiple units combat in starcraft. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6. IEEE, 2017.

[25] Kun Shao, Yuanheng Zhu, and Dongbin Zhao. Starcraft micromanagement with reinforcement learning and curriculum transfer learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 3(1):73–84, 2018.

[26] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artif. Intell.*, 171:365–377, 2007.

[27] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[28] Satinder P Singh and Richard S Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1):123–158, 1996.

[29] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29, 2016.

[30] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[31] Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, and Nicolas Usunier. Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint arXiv:1611.00625*, 2016.

[32] Yuandong Tian, Qucheng Gong, Wenling Shang, Yuxin Wu, and C Lawrence Zitnick. Elf: An extensive, lightweight and flexible research platform for real-time strategy games. *Advances in Neural Information Processing Systems*, 30, 2017.

[33] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[34] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 2004.

[35] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

The Design, Architecture and Scalability of Microservices

Jayshree Rathi
jayshree.rathi@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

Microservices architecture, which emerged from service-oriented architecture, has become a leading design approach to developing applications for large-scale systems. Microservices are getting increasingly widespread in these systems owing to the many benefits it provides, such as shorter deployment cycles, increased scalability, and effective separation of services. This article provides a generalized introduction of microservices architecture, compares it with traditional architectures, and evaluates microservices from a technological and architectural standpoint.

KEYWORDS: microservices, monolithic, scalability, service-oriented-architecture (SOA)

1 Introduction

In recent years, the demand for domain-driven design, continuous delivery, scalability, and automation has given rise to the development and advancement of microservices [1]. Microservices are small, autonomous services that interact with each other through standard interfaces. Each of these services is developed, deployed, and maintained independently.

The microservices approach enables flexibility in the use of programming languages and databases. The microservices do not exchange data but communicate with each other using the Representational State Transfer (REST) protocol [2]. The main goal of microservices is to isolate a business functionality into its service, allowing the services to run independently of each other and perform their associated tasks.

Microservices offer several advantages, including increased scalability, fault isolation, and agility. Microservices also fits well with DevOps, which streamlines the process of moving software from development to production by removing barriers between software development and IT operations [3]. These benefits have prompted software giants to leverage microservices patterns and deploy large applications in the cloud as a collection of small services that can be developed, tested, deployed, scaled, operated, and upgraded independently. As a result, microservices architecture has been instrumental in increasing their agility, reducing complexity, and scaling their cloud applications more efficiently [2]. This paper focuses on some of the critical benefits of microservices over monolithic architecture. It reviews its design and architecture approaches, emphasizing microservices for scalability.

This paper is organized in the following manner. Section 2 compares microservices and monolithic architecture; Section 3 discusses the fundamental concepts of building and modeling microservices; Section 4 discusses one of the key features of microservices: scalability; Section 5 presents a generic idea of integration, communication and deployment in microservices, and Section 6 explores a case study that compares microservices over monoliths.

2 Key Benefits of Microservices Architecture

The microservices design provides numerous advantages over monolithic and service-oriented architectures, including technological heterogeneity, replaceability, scalability, simplicity of deployment, and resilience [4].

Monolithic applications consist of a single codebase on which all the application logic and services are developed. This codebase is shared across developers, and any change in a monolithic codebase impacts a sizable portion of the system. The updated code should ensure that all other services continue to work. Thus, the changes stack up between releases with

a monolithic program, increasing the deployment risk. More importantly, this presents a single point of failure. Multiple machines can be used in the monolithic system to reduce the possibility of this failure, but this increases cost and adds redundancy.

Some enterprises have used Service-Oriented Architecture (SOA) to resolve some of the issues associated with monolithic systems. In the SOA, an application is divided into a set of business functionalities, and it offers services through different protocols. The protocols include SOAP and routing mechanisms, such as ESB (Enterprise Service Bus) [2]. However, SOA implementations can be time-expensive and sophisticated, adding a layer of undesirable overhead. Furthermore, ESBs require a complex configuration on a large scale and can generate high latency, providing a single point of failure.

Microservices enable building resilient systems that handle the total failure of services [1]. With microservices, a single service can be changed and deployed independently. It allows a feature to deploy the code faster, hence providing continuous delivery. The continuous delivery method enables companies, such as Amazon and Netflix [5], [6] to use microservices architecture and get their latest functionality out to customers as fast as possible. In addition, being a system composed of multiple services, microservices allows the use of heterogeneous technology stack based on the use case and suitability for a particular service. Microservices adopt SOA concepts with a focus on agility and simplicity.

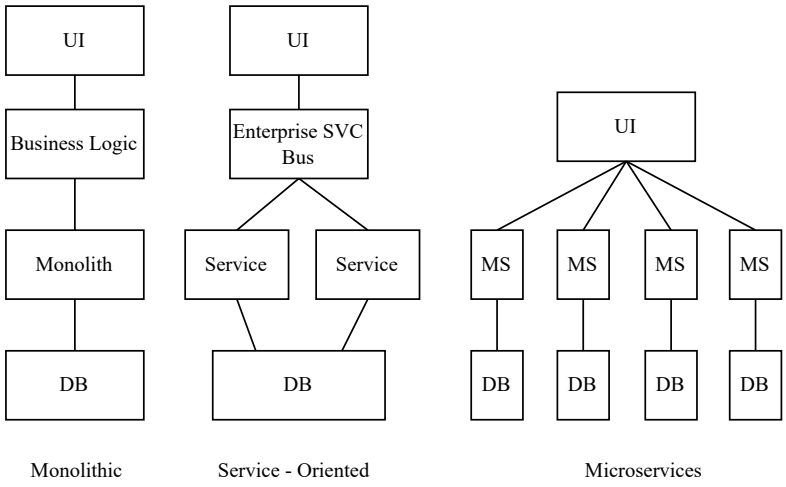


Figure 1. Monolithic, Service-Oriented and Microservices Architecture
Adapted from [7]

3 Building and Modeling Microservices

As discussed in Section 2, microservices architecture offers numerous advantages over conventional monolithic architecture. However, implementing a system based on Microservice Architecture (MSA) approach may increase the development cost and complexity of the system [8]. Therefore, while designing microservices, the boundaries of microservices must be defined. Instead of quantifying the "micro" of the microservice to a numerical value, companies define the "quality," the use case of the service to be used, and the functionality it fulfills [9].

Evans [10] presents a model-centric approach to software design and discusses bounded contexts and domain-driven philosophy that associates with the challenges of designing and building microservices. Model-driven development provides an efficient way to design a microservices-based system and helps define the scope of the system. The key concepts to consider while modeling microservices are discussed below [11].

- **Single Responsibility Principle**: Each microservice should be responsible for a single task. When it comes to developing microservices, this is one of the most crucial design principles. For optimal agility, it aids in defining the scope and boundary of the service. Moving away from this idea may result in creating a new monolithic system.

- **High Cohesion**: A microservice that is highly cohesive has a single responsibility and completes it. It should not share responsibilities with other components, delegate responsibilities to other services, or attempt to complete tasks that are not related to it.

- **Loose Coupling**: The entire concept of a microservice is that it enables updating one service and deploying it without affecting the rest of the system. Loose coupling of the services ensures that modification to one service should not necessitate a change to other services. This is contingent on a variety of factors. Certain software development practices should be avoided or minimized to ensure loose couplings, such as shared libraries, database sharing, over-exposure of a service, synchronous communication, and implementation sharing.

The discussions above do not reach any significant conclusions on the modeling language better suited for the Model-driven Development (MDD) of MSAs. Mike [9] compares the effectiveness of two different approaches to microservices modeling, namely the domain-specific modeling language - Language Ecosystem for Modeling Microservice Architecture (LEMMA) and the Unified Modeling Language (UML). The UML is a common (standard) modeling approach for service-based software systems that is also applicable to MSA-based systems [12]. However, being a general-purpose language, the UML lacks specialized ideas for microservices, which may lead to misleading models. This is because the generic modeling parts of UML must be modified to represent the specific functionalities that the MSA is trying to encapsulate. On the contrary, LEMMA is a textual-modeling language where each modeling language is aligned with a specific MSA to fulfill a particular business purpose. LEMMA-based modeling serves domain experts, service developers, and operators, also allowing to customize the deployment [13], [14].

4 Scalability

Scalability is the ability of a system to manage an increasing quantity of work by increasing resources simultaneously [15]. Building scalable software allows planning for future growth while also producing a leaner solution that meets present demands without adding unnecessary complexity. Scalability is one of the most important factors when designing software systems, as they help deal with failure and increase performance.

Generally, when discussing scaling in monolithic applications, the entire application is scaled in one piece. Although a single component experiences high demand, the entire monoliths need to be scaled, increasing the cost and demand of a powerful hardware [16]. Microservices architecture overcomes this limitation of monoliths. Microservices are designed according to business functionalities and require a full-stack software deployment for each business sector. Furthermore, since microservices are independent and autonomous, only those services experiencing a growing load can be scaled. Some of the common scaling techniques are discussed below:

- **Splitting Services**: Microservices are self-contained processes that

communicate over the network, allowing to migrate them to their hosts to boost performance and scale. This can also improve the robustness of the system because a single host outage will affect a smaller number of microservices.

- **Risk Distribution**: One strategy to scale for resilience is to avoid operating several services on the same host. Several virtualization solutions ensure that the hosts are distributed across multiple separate physical boxes to mitigate this risk. Another approach could be to ensure that services are not all running on a single rack in the data center or scattered over multiple data centers.

- **Load Balancing**: Load balancing is the technique of spreading incoming network traffic amongst servers server pools at a concurrent or discrete period. The simplest method to accomplish resilience and scalability with a microservice that exposes a synchronous HTTP endpoint is to run a microservice instance on numerous hosts behind a load balancer. Load balancers enable transparent scaling of microservices to any service customers. This increases the capacity to manage load and also mitigates the impact of a single host failure.

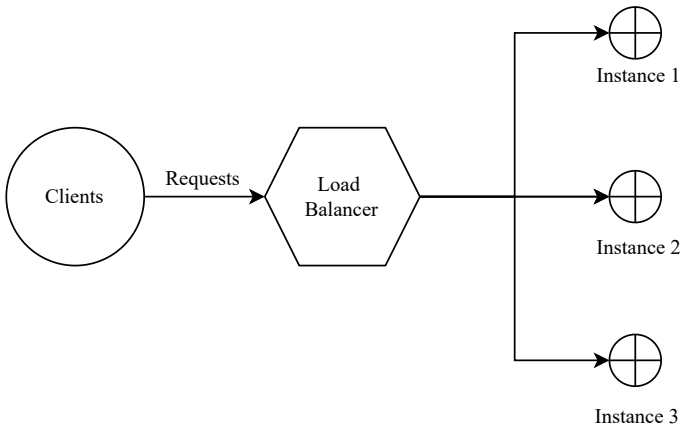


Figure 2. Server-side load balancing approach for scaling
Adapted from [17]

A microservice that is architected to be scalable still faces some challenges. With microservices, scaling may include managing many distinct components and services. This means that either all components must be upscaled simultaneously, or there must be a technique for determining which components should upscale and guaranteeing that they continue to integrate with the rest of the system [18]. Since every microservice is a part of intricate dependency chains, if any dependency of microservices does not scale with it, this creates a bottleneck for its clients [19]. In addition, scaling microservices should not compromise the end-user experience. The speed and reliability of the system should be intact with the implementation of a scalability solution.

While scalability and performance are distinct measures, they are inextricably linked. Concurrency and partitioning are two critical features of designing a microsystem that will affect its scaling performance [18]. Concurrency is a term that refers to the process of breaking down each particular task into smaller components. Meanwhile, partitioning will decide how efficiently these smaller bits may be processed in parallel. Scalability is determined by the efficiency with which jobs are divided and decomposed, whereas performance measures the capacity of the system to process these tasks efficiently. Over time, a popular and successful microservice system can anticipate a consistent increase in traffic and resource demands. Each microservice must scale independently and as part of a more extensive system to scale successfully. This necessitates that each dependency of the microsystem scales in lockstep with it.

5 Communication, Integration and Deployment

Microservices communicate with one another via messaging, a lightweight and straightforward mechanism. Microservices can communicate using synchronous or asynchronous messaging techniques and a variety of message formats, depending on their purpose and requirements. REST is one of the most frequently used synchronous messaging techniques in microservices, where an HTTP request-response defines constraints based on the resource- Application Programming Interface (API) [20]. Asynchronous messaging, such as AMQP, STOMP, or MQTT, can be used in situations where an immediate response is not required [20]. Advanced Queue Messaging Protocol (AMQP) is an open standard application layer

protocol for reliable and secure asynchronous messaging. AMQP is a binary, flow-controlled, encryption-enabled communication protocol [20].

As microservices perform distinct tasks within their scope, implementing a business use case requires coordination and delivery of the desired result by multiple Microservices. Thus, inter-service communication must occur via a lightweight message bus or gateway with minimal routing and no business logic to avoid architectural complexity. There are various ways to perform inter-service communication, such as point-to-point inter-service communication and API Gateway Inter-services Communication, depending on the requirements and frequency of communication [20].

Deployment is one of the most complex tasks in microservices. Microservices should be deployed independently to ensure agility and minimize the impact on the application. Microservices can be deployed using Docker [20]. This further allows the liberty to break down a microservice into different processes, and each process can be run in a separate docker container. Provisioning tools, such as Kubernetes, can also manage and orchestrate the docker containers.

6 Microservice Case Study

Microservices design patterns have enabled businesses to move applications to the cloud and gain agility and scalability using cloud technologies, such as Infrastructure as a Service (IaaS), and Platform as a Service(PaaS). Companies can increase efficiency in their operations by transitioning to IaaS/PaaS solutions, which offer scalability on-demand and handle peak periods [2]. Microservice architecture is a good fit for iterative development processes, such as agile, and DevOps [21]. Mario and Oscar [2] present a case study in which they developed and deployed a cloud-based enterprise application utilising both a monolithic approach and a microservice architecture that is based on the Play-web framework.

The case study uses the Play web framework with Java to implement both architectures. The embedded server Netty, optimized for rapid startup and low resource consumption, is utilized to execute the play application. Two distinct applications were used to implement the monolithic architecture: a web and a front-end application. Four independent applications were used to implement the microservice architecture: service 1, service 2,

gateway application, and front-end application. Both architectures were deployed using Amazon Web Services (AWS). In the monolithic architecture deployment, a single instance of the web server was used, whereas, in the microservice architecture deployment, three instances were used: one for each microservice and one for the gateway [2].

In the case study, aspects of the monolithic vs. microservice architecture have been analyzed at different levels, including performance, development methodology, deployment and operation, and the business adoption process. The implementation of microservice architecture on AWS required the deployment of numerous independent applications (microservices and gateways). Each microservice or gateway required unique application configurations and AWS services; when new versions of the gateway or microservice were published, it was very easy to break externally dependent services; thus, it is critical to maintaining service versioning in microservices architectures. Additionally, they validate that microservice architectures enable more granular instance type selection (per microservice and gateway), which helps to reduce costs [2].

7 Discussion

Microservices design also has a few drawbacks, alongside providing more benefits than monolithic systems. Firstly, in the model-driven design of microservices, the enterprise/software architect needs to be able to specify the appropriate constrained contexts for service because of its focus on domain awareness. Any misunderstanding at this stage will result in incoherent services. Secondly, the principle of resilience for each microservice places additional resource demands on the concept of scalability. While a microservices architecture, compared to an equivalent SOA and monolithic architecture, is conceptually more scalable, the overhead of monitoring each service necessitates more processing cycles and data storage. As businesses shift towards cloud computing, the additional overhead can be observed using elastic compute resources [22].

Furthermore, the future of microservices also involves integrating a wide range of microservices with server-less functions, reducing the overhead associated with traditional microservice implementations. Serverless architecture provides a foundation to support the agility of microservices in a way that traditional servers are incapable of [23]. In addition,

more research should also be focused on the adoption of solutions already implemented in service-oriented and monolithic architecture to the microservices environment in a cost and time-efficient manner.

8 Conclusion

Microservices architecture is a distributed design approach that attempts to solve the problems of conventional monolithic systems, specifically by enabling businesses and applications to scale while reducing cycle times. It provides numerous benefits over monolithic and service-oriented architecture, such as better scalability, technical heterogeneity, easier manageability, and ease of deployment. The case study in Section 6 highlights that microservices architecture enables more granular instance selection, which results in increased efficiency and reduced costs. Thus, the microservices architecture is the preferable solution for a large, dynamic application with well-defined domains. It, however, comes with a few drawbacks, such as increased architectural complexity, and operational strain.

Nevertheless, microservices are currently the most effective architecture available for large-scale applications despite the few drawbacks. It is the only practical option available for large enterprises that build complex software solutions to deal with complexities and remain competitive. They are thus rapidly becoming the preferred way of building applications in a wide range of industries.

References

[1] Sam Newman. *Building microservices: Designing fine-grained systems*. O'Reilly Media, 2021.

[2] Mario Villamizar, Oscar Garcés, Harold Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas, and Santiago Gil. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *2015 10th Computing Colombian Conference (10CCC)*, pages 583–590, 2015.

[3] Microservice architectures: What they are and why you should use them. `https://newrelic.com/blog/best-practices/microservices-what-they-are-why-to-use-them`.

[4] Sara Hassan and Rami Bahsoon. In *Microservices and Their Design Trade-Offs: A Self-Adaptive Roadmap*, pages 813–818, 2016.

[5] The biggest thing amazon got right: The platform. https://gigaom.com/2011/10/12/419-the-biggest-thing-amazon-got-right-the-platform/.

[6] Microservices at netflix: Lessons for architectural design. www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/, Aug 2021.

[7] Best architecture for an mvp: Monolith, soa, microservices, or serverless. https://rubygarage.org/blog/monolith-soa-microservices-serverless.

[8] Jonas Sorgalla, Florian Rademacher, Sabine Sachweh, and Albert Zündorf. Modeling microservice architecture: a comparative experiment towards the effectiveness of two approaches. pages 1506–1509, 03 2020.

[9] Mike Amundsen. *Microservice Architecture*. O'Reilly, 2016.

[10] Eric Evans. *Domain Driven Design: Tackling Complexity in the Heart of Business Software*. 09 2002.

[11] Modeling microservices. https://medium.com/geekculture/modeling-microservices-df0aaa89ddf9, 2021.

[12] Nenad Medvidovic, David Rosenblum, David Redmiles, and Jason Robbins. Modeling software architectures in the unified modeling language. In *ACM Transactions on Software Engineering and Methodology*, volume 11, 01 2002.

[13] Florian Rademacher, Sabine Sachweh, and Albert Zundorf. Aspect-oriented modeling of technology heterogeneity in microservice architecture. In *2019 IEEE International Conference on Software Architecture (ICSA)*, 2019.

[14] Florian Rademacher, Jonas Sorgalla, Sabine Sachweh, and Albert Zundorf. Viewpoint-specific model-driven microservice development with interlinked modeling languages. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, 2019.

[15] André Bondi. Characteristics of scalability and their impact on performance. In *Proceedings Second International Workshop on Software and Performance WOSP 2000*, pages 195–203, 01 2000.

[16] Nicola Dragoni, Ivan Lanese, Stephan Thordal Larsen, Manuel Mazzara, Ruslan Mustafin, and Larisa Safina. Microservices: How to make your application scale. *CoRR*, abs/1702.07149, 2017.

[17] Load balancing in microservices. https://mesutyakut.medium.com/load-balancing-in-microservices-474ad84b847d/.

[18] Scaling microservices: The challenges and solutions. www.dzone.com/articles/scaling-microservices-the-challenges-and-solutions.

[19] Susan J. Fowler. *Production-ready microservices: Building Standardized Systems across an engineering organization*. O'Reilly, 2017.

[20] Microservices in practice: From architecture to deployment. www.cuelogic.com/blog/microservices-in-practice-from-architecture-to-deployment.

[21] Muhammad Waseem and Peng Liang. Microservices architecture in devops. In *2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW)*, pages 13–14, 2017.

[22] Dharmendra Shadija, Mo Rezai, and Richard Hill. Towards an understanding of microservices. In *2017 23rd International Conference on Automation and Computing (ICAC)*, pages 1–6, 2017.

[23] The future of microservices. https://www.goodworklabs.com/future-of-microservices/.

Formal Methods for Security Analysis of Smart Contracts: A Survey

Ádám Balassa
adam.balassa@aalto.fi

Tutor: Lachlan Gunn

Abstract

Smart contracts are public, immutable, and executable programs, deployed on a blockchain, typically handling cryptocurrency transactions. As such, their security properties and overall correctness are highly critical. In recent years, numerous tools have emerged that aim to verify smart contracts with high certainty by utilizing formal techniques. While their results are reliable, the usability of these tools is commonly limited by their level of automation. Vulnerability detection is often not the main focus of their design, thus assessing their capabilities for security analysis is challenging. This survey gives an overview of existing formal tools for the Ethereum platform while focusing on their usability and limitations for vulnerability detection. It analyzes typical vulnerabilities in smart contracts and defines the challenges that formal techniques have to overcome for their verification.

KEYWORDS: Formal verification, smart contacts, blockchain, Ethereum, Solidity

1 Introduction

Blockchain technology provides an append-only distributed ledger of transactions, which are recorded based on a consensus of miners. The technology has far exceeded its initial purpose, cryptocurrency transactions, pioneered by Bitcoin [14]; a new trend has emerged leveraging blockchains as a distributed computing platform for smart contracts.

Smart contracts are self-executed agreements between multiple parties, typically handling financial transactions. Conditions and terms are defined by lines of code, which are then deployed to a blockchain, by making them immutable. Ethereum is a blockchain platform for a cryptocurrency, called Ether, that is the most well known for supporting runs smart contracts. These decentralized applications are executed by nodes of the blockchain in the Ethereum Virtual Machine (EVM) and are stored in a form of EVM bytecode [5]. Ethereum smart contracts are usually developed in a high-level, object-oriented language, Solidity, which is syntactically similar to JavaScript and C but is very different in its underlying semantics. Writing Solidity code, therefore, requires a very deep understanding of the technology from the programmer.

At the point of writing this paper, the Ethereum platform handles around $300B worth of Ether, which makes it a tempting target for attackers. In 2016, an attack has managed to steal $60M from a smart contract for Decentralized Autonomous Organization (DAO), which had to be reverted by a hard-fork in the Ethereum blockchain [3]. In the also infamous Parity Wallet hack, an attacker drained $30M from an Ethereum contract in 2017 [17]. These attacks were all possible due to simple application bugs, and have drawn a lot of attention to the verification of smart contracts [6].

In the past years, many tools have emerged to support the development of secure smart contracts. As Ethereum smart contracts are generally simple, deterministically terminating Turing-complete programs, that's soundness is critical, they are suitable targets of formal verification [2].

This paper will present, summarize, and compare the existing methods and tools for smart contract verification, which are based on formal techniques, with a major focus on their capabilities of vulnerability detection. The study focuses on the Ethereum platform, which is by far the most popular and well-researched smart contract platform.

The paper is organized as follows. It introduces some of the most common vulnerabilities of smart contracts in Section 2. Section 3 systemati-

cally categorizes formal methods and describes the existing related tools, alongside their capabilities for vulnerability detection.

2 Smart contract vulnerabilities

There are various of techniques to carry out attacks against smart contracts, out of which this paper focuses on application bugs. Such bugs most usually occur due to the lack of understanding of the unique properties of the blockchain platform or the programming language. Table 1 summarizes the vulnerabilities that this paper discusses, while organizing them into major classes.

Non-trivial smart contract properties	Reentrancy
	Delegatecall
	Uninitialized pointers
	Private function exposure
Insufficient exception handling	Limited stack size
	Out of gas
	Unchecked send
Non-trivial blockchain platform properties	Transaction order dependence (TOD)
	Timestamp dependence (TSD)
	Entropy illusion
	Unexpected ether
Arithmetic errors	Overflow
	Underflow
	Truncation
	Signedness

Table 1. Classification of some of the most common smart contract vulnerabilities

Reentrancy: Reentrancy has become a very well-known, non-trivial property since it was the cause of the vulnerability in the DAO contract. When a smart contract sends money to an unknown address (for example in a withdrawal method) and does not count on that such an action will make a callback to the fallback function of that contract, it can lead to a vulnerability. An attacker can override the default fallback function to call the vulnerable contract again, which results in multiple recursive invocations of that contract, while possibly draining money from it. If, for example, such a withdraw method would perform operations in the follow-

ing order: `check balance, send, decrease balance`, the whole contract can be drained, since the recursive withdraw calls would always pass the check stage, as the balance has not yet been decreased [3].

Delegatecall: The Solidity language, compiler, and the EVM have multiple properties that are not typical for other languages and execution environments. The examples include reentrancy, and the mechanisms of the `delegatecall` function, which can be used to invoke external contracts, for example as a way of modularizing the contract logic into "libraries". `delegatecall` enables the user to invoke a contract with the caller's context, meaning that when it accesses the `storage`, it will actually access the caller's `storage`.

Storage variables in a Solidity contract are stored in `slots`, in the order of their declaration. A programmer may think that variables are bound by their names during delegating a call, while they are frankly bound by their position in the `slots`. If they made the mistake of declaring variables in a different order in the caller contract than in the called contract, it results in undesirable behavior that can be exploited [1].

Uninitialized pointers: The same phenomenon influences uninitialized pointers in the Solidity code, which by default, are initialized to point to `slot[0]`. Writes to such a pointer will override the first storage variable with whatever value it is assigned, possibly with a parameter that can be controlled by an attacker.

Private function exposure: The most commonly exploited vulnerability is failing to declare a private function as private, thus allowing attackers to control the state of the contract in a way that was unintended by the programmer [1].

Limited stack size: Programmers of a smart contract must be prepared to handle errors during calls to other contracts. The EVM limits the stack depth to 1024 frames, after which a stack overflow exception is thrown.

Out of gas: [5] Ethereum invented the term gas, which is a small price paid for miners, proportional to the computational resources used by the execution of a smart contract method. When a method is called, an amount of gas is sent with it, that can run out between any 2 EVM statements. This is especially likely if the contract's code includes a costly loop [17].

Unchecked send: In the Ethereum platform, failed calls (due to an exception in the fallback function, out of gas failure, reaching the upper

limit of stack frames, etc.) do not throw an exception, but return `false`. Not checking the return value of a call can cause a vulnerability, if the contract state is modified while naively assuming the success of the call.

Transaction order dependence (TOD): Properties of the execution environment of a blockchain platform can also be easily neglected. The order in which transactions are appended to the blockchain is controlled by the miners. Contracts that do not take this into account can be exploited by malicious miners.

Timestamp dependency (TSD): Smart contract programmers must not build logic on the timestamp value of the execution, as it is also in the miners' control. If Ether transfer depends on the timestamp of the execution, miners can mine the transaction just when it is ideal for them.

Entropy illusion: Similarly, random number generation cannot be trusted, since a malicious miner can perform rejection sampling on the executions' results, by repeatedly discarding and creating new transactions until the generated number favors her interest [17]. It is also common that random number generation is seeded by the current time, which relates this vulnerability to TSD.

Unexpected ether: Smart contract programmers should not trust the value of `balance` either. Even though it seems it is only controlled by public method invocations, it can actually be modified by, for example, 'self-destructing' another contract that will send its balance to a chosen address [1].

Arithmetic errors: Although Solidity looks like a high-level language, similar to JavaScript, the underlying representation of numbers is strict (as it also appears in the Solidity source code). Programmers can make errors by not accounting for overflows, or numerical-precision-related issues when designing a contract [18].

3 Formal methods and tools

While informal techniques, like linting and testing, are widely used in the industry, they do not provide a high level of certainty in the correctness and security of a smart contract, therefore the academic community has turned towards formal methods [6].

Formal techniques require the smart contract to be transformed into a formal model, thus enabling tools to reason about their correctness with regards to some criteria or a full formal specification. These methods

are capable of proving soundness with high certainty. Such techniques include *theorem proving*, *model-checking* and *symbolic execution* [6].

The following section overviews 11 of the most successful formal tools that fall into these classes of verification techniques. The paper lists their special measures and limitations in detecting some of the vulnerabilities listed in Section 2, namely, Reentrancy, Integer overflow, TSD, TOD, and Delegatecall. This set of vulnerabilities was chosen to be representative in a way that they cover all typical challenges that such tools have to overcome. Reentrancy can be verified by all the discussed methods, as their motivation partly originated from the attacks, of which the DAO is the most infamous. To be able to check integer overflows, the used abstraction must be able to correctly model EVM's type-system. TSD and TOD can be easily checked by over-approximations, but are difficult to precisely check without modeling the blockchain execution environment. This paper introduces two properties for verifying `delegatecall`: properly modeling the communication of multiple contracts, and either a low-level representation on EVM's memory model or possessing the information of state-variables' declaration order in Solidity.

3.1 Symbolic execution

Symbolic execution is described as a practical approach between informal and formal techniques. Instead of testing correct behavior for single inputs, it verifies correctness for an entire class of inputs. This method, however, is not suitable to provide full formal proof of correctness. The technique requires the program to be executed while having input variables represented as symbols. The execution, therefore, does not yield numeric results, but symbolic formulae. The programmer interacts with the execution engine by defining predicates for the input variables (which yields a class of expected inputs) and for the correct output. Symbolic execution can be used to prove whether the output predicates hold with all inputs that satisfy the input predicate.

Oyente: The first successful step towards formal verification of smart contracts was realized by Luu et al., in 2016, by introducing a smart contract verification tool based on symbolic execution, called Oyente [12]. It is a fully automated, bug-finding tool for the Ethereum platform, that takes the global Ethereum network state and a single smart contract's EVM bytecode as input, and reasons about its correctness with regards to pre-defined properties. It only works on simplified EVM semantics, which

they named EtherLite. The tool is capable of identifying the bug in the DAO contract, along with many other security vulnerabilities like transaction order dependence. Oyente leverages a formal tool, Z3, to reason about reachability logic, so that it can cut down infeasible symbolic paths, yielding fewer false-positive bug reports.

Although the tool does not aim to provide formal proof of the soundness of a contract, additional properties can be added in the form of plugins. It is able to detect reentrancy and TOD-related vulnerabilities by an over-approximation, which results in a larger number of false positives. It has a more sophisticated way of checking TSD, by only marking execution flows as vulnerable, if Ether depends on the timestamp value. Another crucial limitation of Oyente is that it can only work with integer inputs of arbitrary magnitude, which limits its capabilities for detecting overflows or truncation issues [11].

Osiris: Osiris is a fully automated tool that focuses on a subset of arithmetic errors, namely, integer bugs [18]. It utilizes symbolic model-checking and taint-analysis for verification. Taint-analysis is applied to improve the tool's accuracy, it is suitable for distinguishing between harmless and exploitable overflows. The authors claimed that Osiris is more accurate than any other tool in detecting integer bugs, but it is limited to this single category of security vulnerabilities.

3.2 Model checking

These approaches have the system modeled as a state machine and prove properties by exploring the state-space of the model. The complexity of the state machine can span from finite automata to a complete Turing machine [16]. The prover can often find concrete counter-examples to given properties, making correction of both the code and the model easier. Model-checking techniques are challenged by the combinatorial explosion of states to explore, which gives a strong limitation on their applicability [15]. To overcome this issue, multiple alternative model-checking approaches have emerged, like symbolic model checking, which relies on a more efficient representation of the state-space [10]. Symbolic model checking is often complemented by abstraction techniques or SMT/SAT solving techniques.

The applicability of such an approach is also constrained by the level of automation it provides: models often have to be created manually. It is also usual that these models do not represent low-level EVM mecha-

nisms properly, thus they are not able to reason about arithmetic errors or `delegatecall`. They are, however, suitable for detecting non-trivial platform properties, as it is generally easy to model the execution environment by this formalism.

Nehai's NuSMV model: Nehai et al. propose an approach using a symbolic model checking tool based on Binary Decision Diagrams, called NuSMV [16]. Their method requires the designer of the smart contract to model program logic in NuSMV's modeling language, which is suitable for describing finite automata on which an engine can perform formal reasoning. Verification is based on temporal logic properties, that can be defined by employing classical logic operators and temporal operators. Reasoning about platform-level properties requires the modeling of the execution environment. Nehai et al.'s solution models clients and transactions, but leaves several concepts and mechanisms, like blocks, mining, or gas consumption, for later work.

The Solidity NuSMV translation remains manual, the writers claim that it could be automated, but they only define rules and heuristics for modeling and do not formally prove either completeness or soundness of the abstraction. Even though NuSMV relies on symbolic model-checking, it suffers from the combinatorial explosion of states [15].

The solution in the paper is only capable of verifying reentrancy (and over-approximating timestamp dependency), but the approach would generally be able to properly verify the non-trivial platform properties.

Zeus: Zeus is a symbolic model checking tool, that can automatically reason about safety properties (fairness and correctness) [10]. The tool translates Solidity contracts into an intermediate representation (IR) with a formal specification (LLVM IR). It inserts assertions into the bitcode based on policies that are automatically extracted from the Solidity syntax tree and based on fairness properties, which users can define in a static template. With the inserted assertions, verification reduces to a state-reachability problem, for which Zeus leverages SMT solvers. Automatically verified correctness properties include checking for reentrancy, TOD, TSD, or arithmetic errors.

Even though the user of the tool can add as many policies to ensure safety properties, liveness properties require Zeus to be extended with support for linear temporal logic. It only supports a majority of Solidity features, it is limited in verifying cross-function reentrancy, and its arithmetic error detection is not sound [18].

FSolidM: Mavridou et al. proposed a graphical tool to create smart contracts in the form of Finite State Machines (FSM) [13]. Their tool, FSolidM, comes with complete automation to translate the formal model into Solidity. FSolidM is extensible with plugins to ensure security properties, the authors created plugins for reentrancy and TOD. Even though the authors emphasize that FSMs are formal models, FSolidM's Solidity translator is not verified formally. The tool does not verify security properties via formal analysis but ensures them by generating dynamic checks into the Solidity source code.

3.3 Theorem proving

Such methods require the program logic to be modeled mathematically and can reason about properties defined by the same mathematical formalism. While these methods verify properties with high certainty, their level of automation is rather limited, as the programmer has to interact with the prover to define properties [6]. Today, multiple formal semantics of the EVM have been defined, which all fall into this category of formal methods.

The vulnerability detection capabilities of these formal definitions rely on their completeness, on how well does the new semantics resemble the respective EVM instructions. The novel approach is creating small-step semantics for EVM bytecode, where the formal definition models all single atomic operations. This way the proof assistants can reason about the exact same execution as what EVM would perform [7]. Theorem proving methods can also be limited in reasoning about, for example, TOD, as it would either require a formal model of the execution environment (like the process of transaction mining) or an imprecise over-approximation.

Hirai's Lem definition: Yoichi Hirai was the first to create a complete formal specification of the EVM [9]. The definition was written in Lem, which can be translated into both an executable format in OCaml, making it possible to cross-check the definition against existing test-suites, and also into multiple interactive theorem provers, such as Isabelle/HOL, Coq, or HOL4. The work of Hirai was not meant to be used as a standalone tool, but more as a base for further formal specifications. The executable OCaml translation was successfully tested on the EVM test suite. Invariants and safety properties were manually defined by Hirai for an example contract and proved by Isabelle/HOL.

Proving safety properties is excessively inefficient, the proof of a single

property on a simple contract took 3 hours. Even though the definition of EVM contains all instructions, some mechanisms, like gas consumption and interactions with other contracts, are modeled completely nondeterministically. This makes the method capable of verifying all the security properties, but `delegatecall`.

Amani's Isabelle/HOL framework: Hirai's EVM definition is cumbersome to include in the development process of smart contracts, since without any frameworks or automation, formally defining soundness properties is immensely challenging [2]. Amani et al. aim to overcome this issue by building a sound program logic at the bytecode level atop Hirai's definition, which serves as a framework for formal verification, making it possible to create a formal specification of an example contract in 30-40 lines. The framework introduces a hierarchical concept for program logic, which makes reasoning possible on the level of instructions, blocks (sequential instructions with no `JUMP`s), and the entire program. Amani et al. demonstrate how Isabelle/HOL tactics can introduce some automation to generating verification logic. This provides assistance for manually writing formal specifications in a precondition-postcondition style. The program logic could not restore key control structures of Solidity, such as loops and function calls, which is a strong limitation on its applicability.

Solidity* and EVM*: Bhargavan et al. proposed transpilers from both Solidity and EVM bytecode to F*, in the paper referred to as Solidity* and EVM*, respectively [4]. F* is a functional programming language that is suitable for formal verification based on SMT solving. Solidity* and EVM* were not implemented in F*, but in OCaml, thus other theorem provers could also be utilized, however, they do not take advantage of it. Their work was the first in formal verification of smart contracts, it only serves as a proof of concept that demonstrates the suitability of relying on F* to reason about the correctness of smart contracts. Their transpiler did not include all language features of the source languages, and they do not mention verifying the correctness of the transpilation against the EVM test suite.

Grischenko's F* formalization: Grischenko et al. were the first to create small-step semantics for EVM bytecode, their model maps every EVM instruction individually [7]. A large proportion of their semantics was formalized in F*, the executable OCaml translation was tested against the EVM test suite. They have applied special measures to model inter-contract communication properly, which was a key limitation of mul-

tiple earlier formal definitions. They were also the first to create a formal definition for key security properties, like call integrity and independence from miner-controlled parameters. This introduces significant automation for verifying reentrancy, TSD, TOD, entropy illusion, and many others in a highly sophisticated manner. Arithmetic errors can be properly verified by utilizing such small-step semantics, and the formalization's model for communication of multiple contracts makes it suitable for verifying `delegatecall` too, according to the requirements in 3.

Nehai's Why3-based approach: Following the work with NuSMV, Nehai et al. proposed a theorem proving method, utilizing Why3, a deductive program verification tool, that uses Hoare-logic [15]. Why3 provides a rich language for specification, WhyML, and relies on well-tested theorem provers for reachability logic, like Alt-ergo or Z3. The method is also only a proof-of-concept, the writers demonstrate how an example contract, with fair complexity, can be modeled in WhyML and propose a proof-of-concept compiler from Why3 to EVM bytecode. Their modeling makes use of oracles, defined as links between the real world and the system, which are essentially ports for public methods of a smart contract. Such an approach enables them to specify private methods' correctness in a simple, pre-condition/post-condition style, and leave only public methods for the subject of proving correctness with any given input values. They also propose a gas model, based on Why3 ghost functions, which gives an over-approximation of the execution's gas consumption. The authors demonstrate Why3's flexibility in modeling algebraic data types, thus making the framework suitable for arithmetic error detection. Since the framework could allow creating correct-by-construction contracts, sound verification of reentrancy, TSD and TOD should be possible.

KEVM: The second complete formal semantic definition for EVM was released by Hildenbrandt et al. in 2018, in a language-independent framework: $\mathbb{K}$ [8]. The $\mathbb{K}$ definition allows the user to automatically derive correct-by-definition tools, such as a parser, interpreter, symbolic execution tools, debugger, and formal tools, which can be used for model checking and deductive verification. The deductive verifier uses reachability logic reasoning, employing Z3, to formally prove properties defined in a pre-condition post-condition Hoare-triple style. In contrast to Hirai's work, KEVM's executable specification is extended with a formal gas analysis tool and can be run with the whole VM test suite. It is more than 8 times more performant than the Lem specification's executable code. It

makes it possible to reason about the execution of any ordering of multiple transactions, making it capable of verifying all examined vulnerabilities.

FEther: FEther introduces a high level of automation into formally verifying smart contracts [19]. It extends the authors' previous work of creating a Formal Process Virtual Machine for the Ethereum platform, by including a formal proof engine. They developed a formal memory model in Coq, called GERM, which supports specifications at the code level, and simulates a low-level formal memory space. They utilize a hybrid approach by symbolically executing contracts on the GERM framework and they use the Coq assistant to formally verify them. Contracts must be developed in Lolisa, a large subset of Solidity, for which they provide automatic translation. FEther is supported by Coq tactics that introduce a high level of automation into the verification process. FEther's correctness is certified by Coq, thus reasoning about any of the security properties is sound. GERM's number representation is the same as Coq's, which is suitable for modeling EVM data types properly. As modeling inter-contract communication was not clearly claimed, verifying `delegatecall` might be a limitation.

4 Conclusion

Formal techniques for smart contract verification have gained a lot of attention since the DAO attack from both the industry and the academic community. This paper gave an overview of the most typical vulnerabilities caused by application bugs, originating from special properties of smart contracts and blockchain platforms.

Table 2 summarizes the capabilities of the discussed methods. In addition to the vulnerabilities they are able to detect, the table shows whether they're suitable for gas analysis, and their level of automation. This summary assumes that capabilities that were not specifically claimed by the related papers (like modeling multiple contracts, or gas analysis) are not possessed by the tools. Methods that are not capable of verifying properties without user interaction but introduce assistance automation are marked with a half-circle; while methods that require minimal or no interaction are marked with a full circle.

This paper showed that while numerous techniques exist for formal verification, there are still steps yet to be taken for them to become a routine in smart contract development. A tool that utilizes a formalism of

Method	**RE**	**Int**	**TSD**	**TOD**	**DC**	**Gas**	***Aut***
Symbolic execution							
Oyente [12]	◗	❍	●	◗	❍	●	●
Osiris [18]	❍	●	❍	❍	❍	❍	●
Model-checking							
Nehai's NuSMV model [16]	●	❍	●	◗	❍	❍	❍
Zeus [10]	●	●	●	●	❍	❍	●
FSolidM [13]	●	❍	❍	●	❍	❍	◗
Theorem proving							
Hirai's Lem definition [9]	●	●	●	●	❍	❍	❍
Amani's Isabelle/HOL framework [2]	●	●	●	●	❍	❍	◗
Solidity*, EVM* [4]	◗	●	◗	◗	❍	●	◗
Grischenko's F* formalization [7]	●	●	●	●	●	●	◗
Nehai's Why3-based approach [15]	◗	●	◗	◗	❍	●	❍
KEVM [8]	●	●	●	●	●	●	❍
FEther [19]	●	●	●	●	❍	●	◗

Table 2. Capabilities of the discussed formal methods. Abbreviations: reentrancy (RE), integer overflow (Int), timestamp dependence (TSD), transaction-order dependence (TOD), delegatecall (DC), analysis of gas conumption (Gas), level of automation (Aut)

small-step semantics for EVM, (like Grischenko's F* formalization [7]) could overcome all challenges by performing automatic checks for all well-known security properties. Such a tool should also be capable of formally verifying custom properties provided by the user in the form of a static template, as seen in Zeus [10].

References

[1] Solidity Security: Comprehensive list of known attack vectors and common anti-patterns. `https://blog.sigmaprime.io/solidity-security.html`. Author: Dr. Adrian Manning, Accessed: 2022-02-28.

[2] Sidney Amani, Myriam Bégel, Maksym Bortin, and Mark Staples. Towards verifying Wthereum smart contract bytecode in Isabelle/HOL. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 66–77, 2018.

[3] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on Ethereum smart contracts (SOK). In *International conference on principles of security and trust*, pages 164–186. Springer, 2017.

[4] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Anitha Gollamudi, Georges Gonthier, Nadim Kobeissi, Natalia Kulatova, Aseem Rastogi, Thomas Sibut-Pinote, Nikhil Swamy, et al. Formal verification of smart contracts: Short paper. In *Proceedings of the 2016 ACM workshop on programming languages and analysis for security*, pages 91–96, 2016.

[5] Vitalik Buterin et al. Ethereum white paper. *GitHub repository*, 1:22–23, 2013.

[6] Ikram Garfatta, Kais Klai, Walid Gaaloul, and Mohamed Graiet. A Survey on Formal Verification for Solidity Smart Contracts. In *2021 Australasian Computer Science Week Multiconference*, pages 1–10, 2021.

[7] Ilya Grishchenko, Matteo Maffei, and Clara Schneidewind. A semantic framework for the security analysis of Ethereum smart contracts. In *International Conference on Principles of Security and Trust*, pages 243–269. Springer, 2018.

[8] Everett Hildenbrandt, Manasvi Saxena, Nishant Rodrigues, Xiaoran Zhu, Philip Daian, Dwight Guth, Brandon Moore, Daejun Park, Yi Zhang, Andrei Stefanescu, et al. Kevm: A complete formal semantics of the Ethereum virtual machine. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 204–217. IEEE, 2018.

[9] Yoichi Hirai. Defining the Ethereum Virtual Machine for interactive theorem provers. In *International Conference on Financial Cryptography and Data Security*, pages 520–535. Springer, 2017.

[10] Sukrit Kalra, Seep Goel, Mohan Dhawan, and Subodh Sharma. Zeus: analyzing safety of smart contracts. In *Ndss*, pages 1–12, 2018.

[11] James C King. Symbolic execution and program testing. *Communications of the ACM*, 19(7):385–394, 1976.

[12] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 254–269, 2016.

[13] Anastasia Mavridou and Aron Laszka. Designing secure Ethereum smart contracts: A finite state machine based approach. In *International Conference on Financial Cryptography and Data Security*, pages 523–540. Springer, 2018.

[14] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[15] Zeinab Nehai and François Bobot. Deductive proof of Ethereum smart contracts using Why3. *arXiv preprint arXiv:1904.11281*, 2019.

[16] Zeinab Nehai, Pierre-Yves Piriou, and Frederic Daumas. Model-checking of smart contracts. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 980–987. IEEE, 2018.

[17] Sarwar Sayeed, Hector Marco-Gisbert, and Tom Caira. Smart contract: Attacks and protections. *IEEE Access*, 8:24416–24427, 2020.

[18] Christof Ferreira Torres, Julian Schütte, and Radu State. Osiris: Hunting for integer bugs in Ethereum smart contracts. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 664–676, 2018.

[19] Zheng Yang and Hang Lei. Fether: An extensible definitional interpreter for smart-contract verifications in Coq. *IEEE Access*, 7:37770–37791, 2019.

Sim-to-Real transfer learning using Deep Reinforcement Learning (DRL)

Jeyhun Yagublu
jeyhun.yagublu@aalto.fi

Tutor: Anton Debner

Abstract

In recent years, Reinforcement Learning and Deep Learning techniques have been brought together by many researchers to solve complex tasks, including navigation, motion and exploration. Due to time inefficiency and complexity of training models in real life increasing number of research projects has been conducted on Sim-to-Real Transfer learning using Deep Reinforcement Learning. This paper provides an overview on some of Sim-to-Real techniques and problems surrounding it, and discusses some of the recent projects by main research entities in the field.

KEYWORDS: sim-to-real, deep reinforcement learning (DRL)

1 Introduction

Over the last several years, the robotics community has increasingly utilized reinforcement learning (RL) algorithms to manage complicated robotic systems, or to give end-to-end policies from observation to operation [16]. These algorithms are based on the principle of trial and error that individuals use to acquire new skills, and they draw their knowledge from the rewards that agents obtain for responding in certain ways to a range of circumstances. Considering how limited time and experience diversity are

in real-world circumstances, this necessitates a high number of training sessions. Furthermore, while working with actual robots, it is necessary to consider the possibility of possibly harmful or unexpected behaviors in security sensitive applications [3]. In simulation contexts, deep reinforcement learning (DRL) algorithms have proved successful, but their performance outside of simulated worlds has been restricted [16].

Recent advances in the sim-to-real sector, however, have proved that DRL is a very effective technique. Previously, intractable actions, including managing limb motion and exploring an unknown area, have become accessible due to these new technologies [8]. Although simulation-based training is a low-cost method of gathering data, it has inherent inconsistencies when compared to real-world conditions [16]. To close the gap between simulation and reality, first and foremost, methodologies must be developed that can account for mismatches in both sensing and actuation, among other things [16]. The sensing mismatches has received great attention in the deep learning industry in recent years, for example, with adversarial assaults on computer vision systems [1]. The actuation errors may be mitigated by conducting more accurate simulations. In each of these circumstances, some of the current techniques involve work that introduces environmental perturbations [17] or focuses on domain randomization [7]. Another important consideration is that an agent deployed in the real world may be exposed to unique experiences that have not been considered in the simulation models, but also the possibility of having to modify their rules in order to cover a broader range of tasks that have not been previously considered in the models [17].

This paper analyses current state and results of research and experiments in sim-to-real domain and organized as follows. In Section 2, we briefly introduce RL/DRL. In Section 3, we introduce sim to real and problems surrounding it. Section 4 discusses recent significant Sim To Real Experiments . Finally, Section 5 presents some concluding remarks.

2 Brief Introduction to RL/DRL

2.1 Reinforcement Learning and Deep Reinforcement Learning

Reinforcement Learning (RL) [12] is a process in which an agent interacts with its surroundings to obtain the most optimal policy via trial and error.

In the context of Markov Decision Process (MDP) $M = (S, A, P, r, P_0)$, the RL task may be stated as the state space S, the action space A, the state transition probability $P(s_{t+1}|s_t, a_t)$, the reward function r and lastly the probability distribution P_0 of the initial state $s_0 \in S$ [12]. The objective of the reinforcement learning agent is to discover a strategy that maximizes the expected return; at time t it is generally defined as

$$R_t = \sum_t^T r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t} r_{t_T}$$

where T is the terminal time step and γ is the discount factor [2].

Deep neural networks may be adopted to approximate any of the key components of reinforcement learning, such as value functions, policies, and models (state transition function and reward function), resulting in deep reinforcement learning [6]. One example of this combination is Deep Q-learning which brings together one of widely used RL algorithms - Q-learning and Convolutional Neural Networks (CNN). Q-learning tries to maximize the Q-value of the curret states's action where Q(S,A) is defined as

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma max Q(S', A') - Q(S, A)]$$

and the algorithm is repeated several times until the value converging to Q-table value is obtained by the algorithm [4]. Hester et al. 2017 has developed Deep Q-learning and demonstrated that it over-performs conventional reinforcement learning algorithms.

3 Introduction to Sim-To-Real and Problems Surrounding it

It is very costly and time-consuming to train a model in real environments even using most modern technologies. First of all, creating environment suitable for training the model is harder in real-life situations and in some cases related to robot locomotion, model should be protected to not break the expensive equipment. Additionally, making the suitable environment and training process of the model takes considerably amount of time since most parts of training is manual. The simplest example of it would be resetting the real training environment to restart the learning process or advance to a new state which happens almost instantly with simulated environments. Thus, many methodologies have been developed for training the model in simulated environments and then implementing the policy to real-world tasks. The main challenge of sim-to-real is that the

simulation is almost never a perfect model of the real-world environment. Therefore, an agent trained in a simulation will probably not behave optimally in the real-world environment, which is often more complex than the simulated environment.

There are multiple ways of transferring the policy from simulation to real-world implementation.

Zero-shot Transfer

A zero-shot transfer is the most basic method of transferring policy from simulation to reality [16]. It involves developing a realistic simulator or testing the policy sufficient amount of time in simulation in order to implement the policy immediately in real-world scenarios which is considered as Zero-Shot Transfer. However, realistic simulator is not easy to implement, but some techniques have been developed to create it, referred as System Identification and Domain Randomization.

System Identification

Notably, simulators do not provide a realistic depiction of the actual world in all circumstances. System identification [5] is the process of developing an accurate mathematical model of a physical system, and thorough calibration is required to make the simulator more realistic. Nevertheless, most of techniques for system identification are for models that are linear in the parameters, while real environment has many physical parameters that vary in non-linear way, such as smoothness of a road, air thickness. All this variety adds up to difficulty and complexity of environment, thus making it extremely difficult to simulate.

Domain Randomization

Domain Randomization [14] is method that, rather than meticulously modeling all of the parameters of the real world, simulation could be highly randomized in order to cover the true distribution of the real-world data despite the difference between what is predicted by the model and what actually happens in the real world. Figure 1 depicts domain randomization technique:

Domain randomization can be categorized into types: visual randomization and dynamics randomization, which are distinguished by the elements of the simulator that are randomly selected. The training data for some robotic vision activities comes from a simulator, which always has different textures, lighting, and camera angles than the real-world

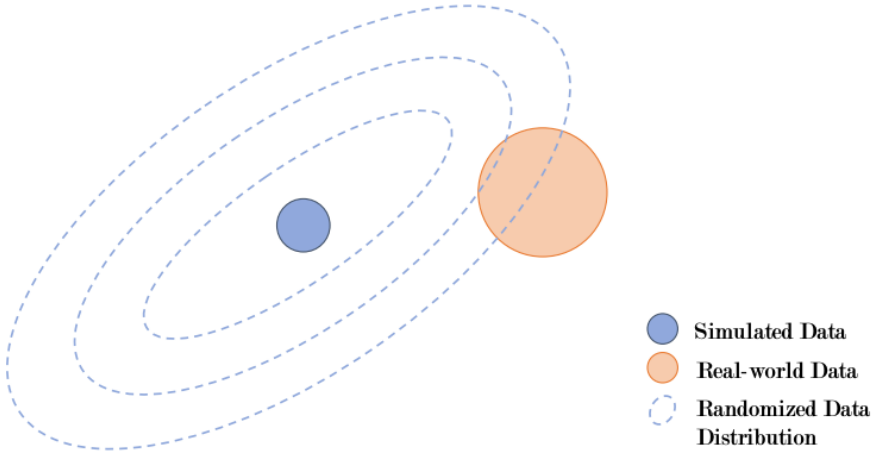


Figure 1. Through proper randomization of parameters, Simulated distribution can be made to cover true real world distribution.

situations in which they are performed. As a result, visual domain randomization seeks to offer sufficient simulated diversity of the visual characteristics during training in order to be able to generalize the model to real-world data during testing [13] (Figure 2).In addition to introducing randomness to the visual input, dynamics randomization may also aid in the acquisition of a strong policy through randomizing several physical features in simulation environment including respective dimension and sizes, physical parameter coefficients and overall physics, which is particularly important when the resulting policy will act as a controller for various robotic tasks.

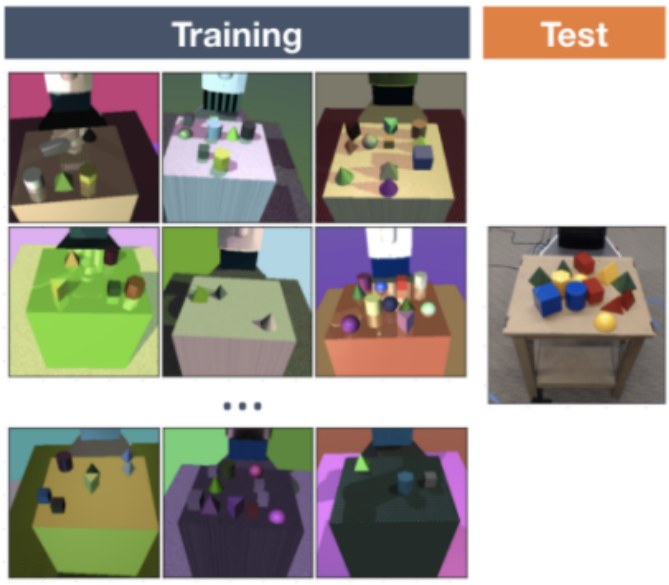


Figure 2. Randomized images are created through visual randomization on which model is trained to be able to transfer to real non-randomized images[15].

Domain Adaptation Methods

Domain adaptation (DA) [16] is a collection of transfer learning approaches for updating the data distribution in simulation to match the real world distribution via a mapping or regularization imposed by the task model (Figure 3). Because there are typically a variety of feature spaces between the source domain and the target domain, differently from the domain randomization which uses pure randomization, the DA technique attempts to unify these two feature spaces through mathematical regularization and mapping, in order to improve the transfer of knowledge from source data to the target domain [16].

Simulation Environments

The selection of a simulation environment is an important consideration in the simulation-to-real transfer process. Regardless of the methodologies employed for effectively transferring information to actual robots, the more realistic a simulation is, the better the outcomes that may be expected in the long run [16].

Mujoco and Bullet, two popular robotic simulators, are known for their fast multi-body dynamics implementations and their greater interaction with DL and RL libraries. However, they were designed to run on CPUs with a limited level of parallelism [11]. Gazebo, on the other hand, has the benefit of being deeply connected with the Robot Operating System (ROS)

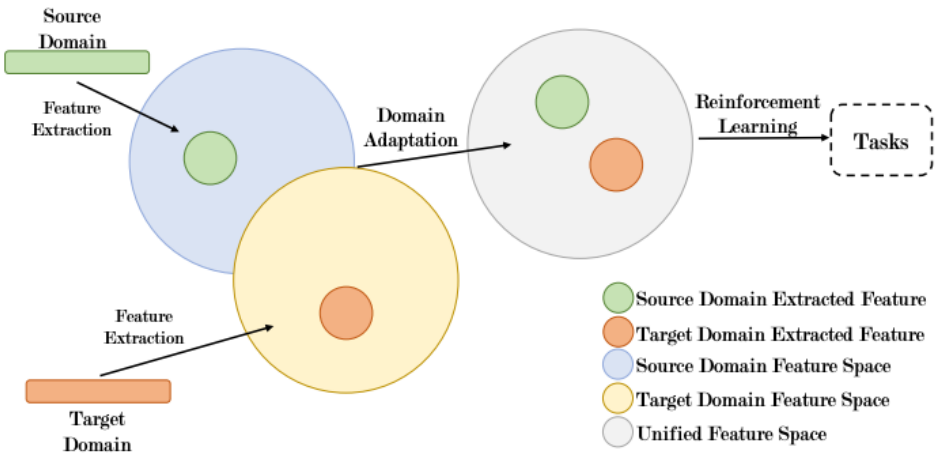


Figure 3. Domain adaptation [16]

middleware; as a result, it can be used in conjunction with a portion of the robotics stack that is present in real robots, which is a significant advantage [16]. However, NVIDIA's Isaac Gym is a simulation environment which uses the GPU to handle both simulation and training and is capable of simulating hundreds of robots in parallel, which makes it possible to train successful policies in as much as 20 minutes opposed to 120 hours [11].

4 Sim To Real Experiments

In recent years, many developments have been made in this domain by companies such as OpenAI, Google, and NVIDIA. While they have used similar techniques in their approaches, they have also focused in their own aspects. For example, NVIDIA's project has focused more on training policy using massively parallel deep reinforcement learning on a single GPU. This section discusses some of the main projects .

4.1 Google

RL-CycleGAN

The ability to learn visual representations from beginning to end while working with a task controller comes at a high cost in terms of sample complexity [10]. In addition, since the data required for RL is often task

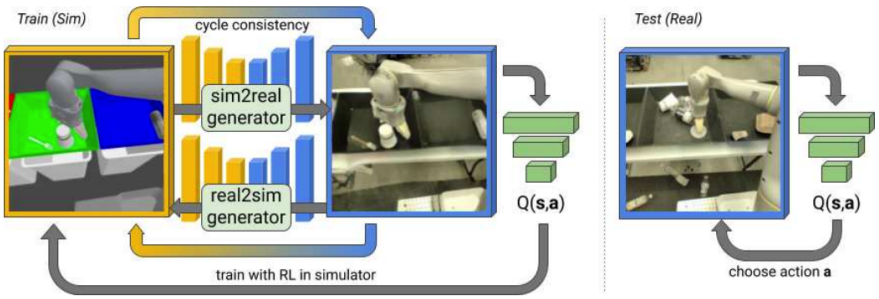


Figure 4. RL-CycleGAN trains a CycleGAN that maps a picture from the simulator (on the left) to a realistic image (in the middle), and a concurrently trained RL task guarantees that these images are usable for the specific task being taught on the right hand side. The RL model may be transferred to a real robot during testing (right)[10]

.

Sim-to-real model	Robot 1 Grasp Success
Sim-Only	21%
Randomized Sim	37%
RL-CycleGAN	70%

Table 1. Success rates of Grasp robot

and policy relevant, gathering this data while still in the loop with policy training may be extremely challenging [10]. The use of RL to train policies in simulation and then transferring these policies to real-world systems is an attractive suggestion. RL-CycleGan project uses CycleGAN to adapt the pixels of simulated images to real environment pictures and then training the RL model on these images (Figure 4). The results showed that this method has improved success rates of grasp robot from 21% when trained Sim-only to 70% while training with RL-CycleGan (Table 1).

4.2 OpenAI

Solving Rubik's Cube with a Robot Hand

Rubik's Cube is common challenge among robotics thus it is not surprising that OpenAI in 2019 has trained a five-fingered humanoid hand to handle the cube and solve the task successfully [9]. This robotic hand has been trained on simulations only, first to handle the cube using their own algorithm referred as automatic domain randomization (ADR), in order to be able to handle the cube even when there are disturbances in the en-

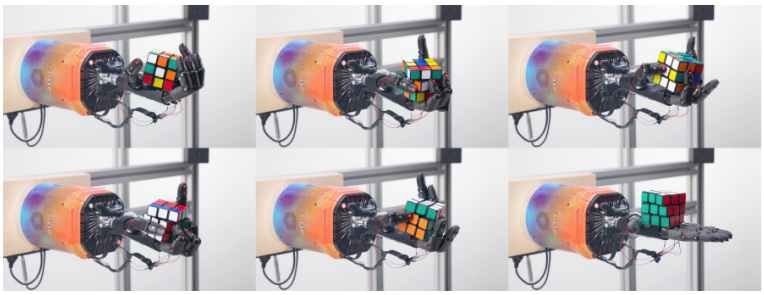


Figure 5. A five-fingered humanoid hand solved a Rubik's cube after being taught using reinforcement learning and automated domain randomization [9]

.

vironment, such as pushing the cube or adding alien objects, or tying its fingers (Figure 5). Three Camera feeds from different angles have been used to calculate the position and pose of the cube and relative fingertip positions. The policy have been trained for several months which includes also developing the algorithm and tuning its parameters and features. Distributed training method have been used which overall consists of 96 NVIDIA V100 GPUs and 1320 worker machine with 32 CPU cores each . Those several months of distributed training would convert to 13 000 years of cumulative non-distributed training time.

4.3 NVIDIA

Training Robots to Walk in Minutes

DRL has been used to solve tasks involving walking and self-navigation in not only humanoid robots but also multiple legged robots. However, this task usually takes much time which includes creating the training model and tuning hyperparameters for several months and actually training the model for several days. In 2021 Rudin et al. at NVIDIA have focused on decreasing the time it takes to train such robots using Massively Parallel Deep Reinforcement Learning in NVIDIA's Isaac Gym simulation environment [11]. Their results are astonishing comparing to the immensity of the task, as they have achieved to train ANYbotics ANYmal C robot to walk in under 20 minutes on a single GPU workstation. 4096 robots have been trained for 1500 policy updates using step size of 25 on NVIDIA RTX A6000 GPU for 20 minutes which obtained close to 100% of success rate for walking up and down the stairs and on uneven terrain when transferred to the robot in real environment.

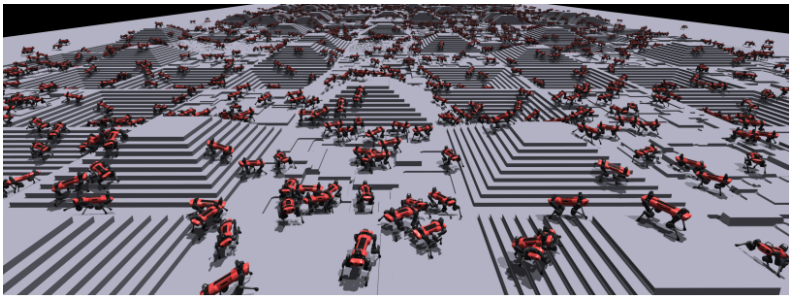


Figure 6. ANYbotics ANYmal C robots training simultaneously in NVIDIA's Isaac Gym [11]

.

5 Conclusion

Sim-to-Real transfer learning gets used increasingly more in variety of research projects as many new techniques are tested and results obtained prove them to be successful. However, there are still many drawbacks as although simulation environments can be made close to reality, they still can't resemble real environment with all of its features and dynamics. Nevertheless, combining it with DRL techniques have proved to be suitable for many complex tasks since in essence RL is similar to how humans learn anything, including walking, dancing and driving.

Many of the research projects in the field have been carried out only in recent years, due to access to more computational power and discovery of new techniques. As a consequnce, there are still many challenges to address and much of field remains unexplored.

References

[1] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[2] Alessandro Capasso, Giulio Bacchiani, and Alberto Broggi. From simulation to real world maneuver execution using deep reinforcement learning. 05 2020.

[3] Javier García, Fern, and o Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015.

[4] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Wook Kim. Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, 7:133653–133667, 2019.

[5] Kristinn Kristinsson and Guy Albert Dumont. System identification and

control using genetic algorithms. *IEEE Trans. Syst. Man Cybern.*, 22:1033–1046, 1992.

[6] Yuxi Li. Deep reinforcement learning: An overview. *CoRR*, abs/1701.07274, 2017.

[7] Fabio Muratore, Christian Eilers, Michael Gienger, and Jan Peters. Bayesian domain randomization for sim-to-real transfer. *CoRR*, abs/2003.02471, 2020.

[8] Philipp Reist Marco Hutter Nikita Rudin, David Hoeller. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning. Technical report, NVIDIA, October 2021. https://arxiv.org/pdf/2109.11978.pdf.

[9] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019.

[10] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. *CoRR*, abs/2006.09001, 2020.

[11] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. *CoRR*, abs/2109.11978, 2021.

[12] R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.

[13] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[14] Joshua Tobin. Real-world robotic perception and control using synthetic data. 2019.

[15] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017.

[16] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. *CoRR*, abs/2009.13303, 2020.

[17] Wenshuai Zhao, Jorge Peña Queralta, Li Qingqing, and Tomi Westerlund. Towards closing the sim-to-real gap in collaborative multi-robot deep reinforcement learning. In *2020 5th International Conference on Robotics and Automation Engineering (ICRAE)*, pages 7–12, 2020.

Theoretical Framework for Cloud Computing Network Measurements

Bipin Khatiwada
bipin.khatiwada@aalto.fi

Tutor: Lorenzo Corneo

Abstract

Numerous experiments and researches are conducted to measure the internet and network parameters, such as latency, patterns in bandwidth consumption, and network traffic analysis. These experiments are done with different methodologies, tools and platforms, data captured and analyzed, time, intervals and length. This paper compares some of these papers and articles to identify the similarities and dissimilarities between them. It then presents insight in terms of the methodologies, results, target and several other parameters employed for the network measurement.
KEYWORDS: Network measurement, Speed test, network throughput, RIPE Atlas

1 Introduction

As most of the applications and day-to-day tasks are shifting towards virtual over the internet, the use of cloud applications has skyrocketed in recent years. This growth is further pushed with most people working from home after the COVID19 pandemic started [26]. Video conferencing, social media, streaming services, and distance learning are a few of the major areas for which the internet is used for. These services are served from different data centers and cloud providers. Therefore, it is crucial to measure the internet and cloud computing networks, as the measured re-

sult can provide insights on the current status, problems, and bottlenecks of the infrastructure. This allows cloud providers to identify problems and propose new solutions to fix them. The result also improves the experience of the end-user, saving cost and energy by effectively setting up the network components to best serve the contents.

Researchers use different approaches to measure the network metrics. Frameworks, such as RIPE Atlas[9], BISmark[2], and SamKnows[6] use different devices scattered in different regions to measure in different ways and for a different purpose [17]. Furthermore, researchers employ several techniques that include using various source-destination pairs, targets, duration of experiments, and interval between adjacent measurements. Such varieties are difficult to track and decide which results to follow for different use cases. Services, such as Amazon Web Service (AWS)[3], Google Cloud Platform (GCP)[4] and Microsoft Azure[5], which contribute a major role in delivering online services add options to the end-users making it even more difficult with the decision [22]. This paper aims to provide a holistic view of network measurement techniques for cloud computing, as well as data, facts and a comparison of them.

We focus on three major aspects: (i) understanding the current cloud system infrastructure, (ii) comparative analysis of performance metrics, and (iii) analysis of different cloud service providers. All of these provide an overview of the cloud computing infrastructure, understand the factors that affect the performance, and how the service delivery is optimized by major content providers.

The paper is organized as follows. Section 2 provides background and theory of related topics. Section 3 provides the literature review of different papers focusing on the comparison and presentation on different metrics. Section 4 discusses the findings and speculation of the data. Finally, Section 4 provides concluding remarks.

2 Background

Network Basics

A network is a collection of two or more computers, servers, peripherals or other devices to allow data and resource sharing. The internet is the network of different networks spread across the different geographic locations. An Autonomous System (AS) is a collection of such networks (or routers) which are controlled by one or more network operators managed by a single administrative entity [7]. The administrative entity deter-

mines the routing policy of those addresses over the internet. Each of such ASes is identified by a unique number called an Autonomous System Number (ASN). ASN is a unique number that is mainly used for internetwork communication and used by routing protocols, including Border Gateway Protocol (BGP). Each router in the network is identified by a unique identifier called Internet Protocol (IP) Address. The most popular IP addressing schemes are IPv4 and IPv6. IPv4 uses a 32-bit address format using dotted-decimal notation allowing the total of 2^{32} = 4.19 billion addresses. IPv6 is developed to overcome the shortage of IPv4 addresses with a 128-bit address format supporting up to 2^{128} internet addresses in total [14]. Multiple programs, services, processes, protocols and devices can access the network using a single router. In order to do so, each entity is labeled by a unique number called Port Number. Using port, any requests sent and received to an IP address is correctly passed to the origin of the request.

Border Gateway Protocol

A Border Gateway Protocol (BGP) is a protocol that governs the flow of traffic between different ASes. BGP records the routing policy of the ASes, IP addresses they control and other ASes that they can connect to. Based on this information, it forms a routing table to determine the fastest route between any two ASes. Public Wide Area Network (WAN) uses this record and determines the next hop for the data packets to reach the destination faster. Once the data packet enters AS, it is routed according to its routing policy. Any changes within ASes, such as change in the addresses they control, is constantly updated. Many cloud service providers implement their private WAN to replace or run alongside BGP to make routing faster.

Cloud Infrastructure and its elements

The major elements of cloud infrastructures are networking equipment, servers, and data storage. Servers provide the computing resources needed to process the data and produce the needed output while data storage serves as a means to store any media, files and databases associated with the service. Networking equipments include the Autonomous System (AS), Internet Service Provider (ISP), Internet Exchange Point (IXP), Private and Public Wide Area Network (WAN), Border Gateway Protocol (BGP), Domain Name System (DNS), Router, Point of Presence (PoPs). They employ virtualization of resources providing hardware abstraction layer, which will reduce the cost of operation and help to scale of the ser-

vices [12].

A Content Provider (CP) is the service or entity which serves the content to the internet users. For example, Youtube serves videos, Spotify serves music and Wikipedia serves text information. A Content Delivery Network (CDN) has distributed infrastructures that enable to serve the content from regions close to the user. The major advantages of CDN include latency reduction, throughput maximization, shortening of packet routes and therefore reduce network congestion [19]. CP uses CDN to distribute the information. They either have their own CDN networks or rents it from big CPs.

3 Literature Review

3.1 Round Trip Time

Experiments measure RTT from the user's local machines, virtual machines deployed in the cloud and also from analyzing the traffic data in different points, such as, PoP point, AS, ISP and router. Performing several experiments at different times and from different locations can provide a more accurate idea of the actual latency.

An experiment by Arnold et al. used Facebook's load balancers data from all across the world and found that the latency improvement of using private WAN is not significantly better than using public WAN that uses BGP [15]. Another analysis of Facebook PoPs traffic data for over 10 days shows that the majority of Facebook users have less than 40ms median RTT and achieve goodput for streaming High-Definition (HD) video [28].

There are several measures employed by cloud service providers to minimize latency. Content providers have PoPs in several geographic locations connected with their private WAN across the internet to provide a faster response. Study shows that faster RTT provides better client's Quality of Experience (QoE) [25].

3.2 Core-internet traffic

The term "core" refers to a subset of ASes which are deeply connected. Data analysis by Labovitz et al. shows that the number of ASes responsible for generating most of the traffic has shrunk from thousands of ASes to only tens of ASes in recent years [23]. The top seven players responsible for generating internet content are Akamai, Amazon, Facebook, Apple, Google, Microsoft and Netflix. Akamai has output traffic of over 10 Tbps, while Facebook and Netflix each have 1 Tbps. During the peak pe-

riod, the highest amount of traffic in North America is used by Netflix (35%), Youtube (17%) and Amazon Video (4%). Similarly, the largest data transfers are carried out by Amazon (42%), Microsoft (15%) and Google (7%) [19]. Figure 1 shows the evolution of the traffic by CPs based on the publicly available database. All of these CPs have their CDNs, most of which are offloaded from the Akamai CDNs. The strategic placement of CDNs across different continents and large userbases worldwide have boosted this traffic generation from these CPs. Currently, a large number of smaller CPs also use or rent the CDNs of these top CPs, which is also the major reason for them to rule the internet traffic.

3.3 Network performance of AWS, Azure and GCP

Amazon's AWS, Google's GCP and Microsoft's Azure are the major cloud service providers. The network performance combined with the pricing and services is crucial in deciding which service to take. The 160 million data points taken over 30 days at the 10-minute interval show some interesting insights on their performance [22]. The inter-Availability Zone (AZ) performance is best in GCP (0.79 ms) and AWS (0.82 ms), followed by Azure (1.05 ms). However, based on the user location, these providers have varying performance. When comparing bi-directional latencies from Australia and Asia, it shows that Tokyo has the lowest latencies in any platform, while locations, including Singapore and Mumbai have the worst performance using AWS and GCP.

Based on the hosting region, the end-user experience varies. For instance, hosting in the UK provide less than 25ms bi-directional latencies in all platforms for European users, while it is 250ms average for Asia, 120 ms average for North America and 200 ms average for South America. Changing the host region to Virginia has average bi-directional latencies of 48 ms in North America, 95 ms for Europe, 210 ms for Asia and 130 ms for South America.

The performance of AWS in Asia is less predictable with the latency variation higher than GCP and Azure. This is mainly due to its architectural design of it: the traffic traverses longer on AWS deployments.

All of these 3 providers peer with each other well. The packet loss in multi-cloud is 0.01% in all of them, while jitter is less in Azure-GCP (0.29 ms) compared to AWS-Azure (0.43ms) and GCP-AWS (0.5 ms).

3.4 BGP and Private WAN

BGP is the internet's intern-domain routing protocol that forwards the requests choosing the effective path to reach the destination. It is designed to make efficient routing. However, some large companies, such as, Amazon, Google and Facebook also implement their own WAN network for routing and use it alongside BGP to reduce the latency. Different researches [15, 17] show that the private WAN provides little benefits for replacing BGP. BGP, which uses anycast, performs as well as the best unicast, implemented by private WAN, for 70% of the traffic. In comparison, the best unicast is faster than anycast by a minimum of 100ms for 10% of requests. A little less than 50% of the observations showed improvement on using private WAN of cloud provider's private WAN while observing the measurements from ASes that hosts 91% of world users [16]. These analyses show that private WAN could be a benefit for the companies with billions or trillions of requests every day. However, there is no more advantage of replacing BGP for the normal cloud services.

3.5 Tools and platforms

Different tools and software can be used to measure different attributes of a network. The trend shows that "traceroute", "netflow[8]" and "yarp[18]" seem to be a popular choice to measure the latency [17, 21, 24]. The tools provide easy to implement and evaluate data while analyzing the response time and different hops of the requests. For speed measurement, tools such as "Speedchecker[11]" and "ping" are the popular ones [16, 17]. These are employed to measure the speed from the end-users location as well as the virtual machines in the cloud. "zmap[13]" and "zmap6" are used to find responsive addresses in IPv4 and IPv6 respectively [24]. Another useful platforms used is the RIPE Atlas. With different devices and anchors deployed worldwide, RIPE Atlas allows measuring the custom measurements using its API [10]. The most common measurement types are the ping and traceroute. Other popular techniques researchers employ are the sniffing of packets using tools, such as, Wireshark, log files from ISPs and IXPs, DNS redirection, logs of the cloud provider's PoPs, private WAN and other logs that are made available by the companies [24, 27, 28, 21, 16].

3.6 Length and interval

The result and inference largely differ by the length of the experiment and the interval on which they are repeated. Most of the studies related

to traceroute, ping and DNS are repeated for several days to weeks [17]. A study related to Port 0 traffic [24] measured the IXP data of 1 week. Similarly, facebook's performance measurement study [28] gathered the results based on 10 days of data.

Based on the data, time and interval, it seems that in case the data being measured are not affected by time and day, then it is usually good enough to work on data collected over 1 or 2 weeks. However, more intervals and longer length of the experiment is always more reliable and provide accurate insights.

3.7 IPv4 and IPv6 addresses

The study by Prehn et al. [27] spotlights on the market of IPv4 and IPv6 addresses. Since the IPv4 addresses are limited, most of the Regional Internet Registry (RIR) already have their IPv4 exhausted. The study performed using 2 databases: information observed by BGP collectors and RDAP databases operated by RIR shows that the market for leasing and buying IPv4 addresses is gaining more popularity. The study sheds light on some interesting business models: ISP buy larger blocks of IPv4 and lease the potential customer, young businesses start their domain by leasing, and VPN services rotate the leased IP addresses to prevent blocking.

3.8 Other Traffic analysis

Some of the popular insights we gathered are:

- Maghsoudlou et al. [24] analyzed 73 GB of port 0 traffic and 1 week of IXP traffic to find out that few hosts generate port 0 traffic, most of which do not have payload and the ones with payload are of BitTorrent. The response rate is high in TCP port 0 probes in IPv4.
- The RIPE Atlas probe distribution is skewed with 45% of the probes falling within AS rank less than 102. The rest of the ASes contain less than 10 probes. COMCAST has the highest number of probes [17].
- Not all RIPE Atlas probes reveal their public IP or AS origin. The origin of such probes can be obtained by using a traceroute.
- The study of TCP and HTTP layers from Facebook's PoPs [28] shows that almost half of the requests have less than 2 KB of response payload.
- Number of networks hosting HyperGiant(HG) off-nets has increased thrice from 2013 to 2021, and that 5 major HGs have 50% of the internet traffic [20].

4 Discussion

4.1 Trends in network measurement experiments

We evaluated 14 papers published in different journals to see the different type of data they measured to evaluate the network performance. The figure 3, shown below presents the insight of popularity of data that the researches consider for. Each paper has one or more data considered.

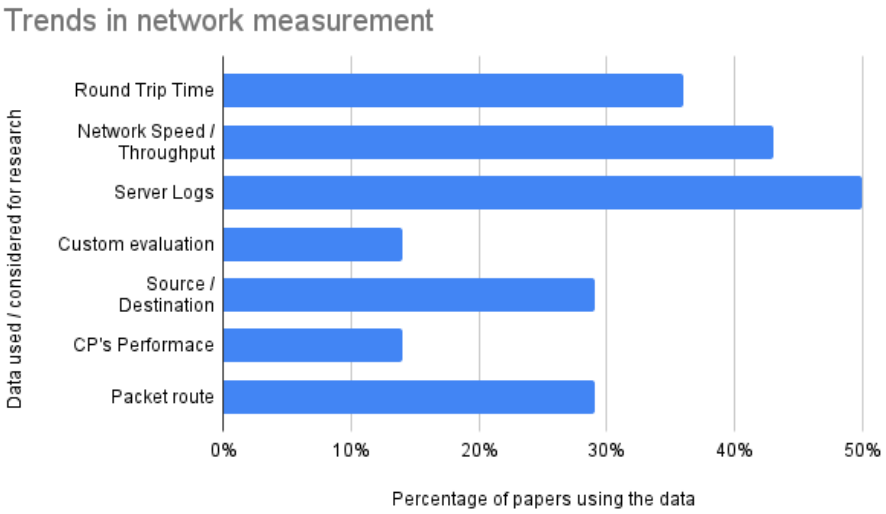


Figure 1. Percentage of papers and data used for network measurement experiments

Server logs seem to be the most popular option followed by RTT and Network Speed / Throughput. The reason for logs being most popular is because it provides more information about different variables that researcher considers. Similarly, RTT and network speed are the major important metric that companies and CDN aim to get the best result in. Table 1 shows the different popular methods, tools and platforms used, along with the data collection period they use to measure the data.

4.2 Logs from PoPs and CDNs are more popular

Logs of Facebook's PoPs and CDNs are the most popular data source to analyze traffic [28, 15, 16, 22]. Facebook has billions of users world wide spread across all regions of the globe, so, the collected sample would be more representative of the real population. Also, they process very high number of requests within short amount of time, which is enough to deduce patterns of use in research [28]. However, normal people have limited access to the Facebook's logs and other popular CDN's traffic. Thus the researchers have challenge of having access to the platform and data. Most of the researches that use this Facebook's logs are either the company's employees or are supported by the company for the research.

Trends in network measurement experiments			
Measured data	**Methods Used**	**Tools / Platforms**	**DCP**
RTT	End user device to server, VM deployed in cloud to server	Traceroute[a], yarp [18], netflow[b], HTTP session logs from CP PoPs, RIPE Atlas[c]	1d - 10d
Throughput	CDN placement simulation, tracing packet route from different sources	Speedchecker[d], ping	NA
Custom measurements	RIPE Atlas's API, Log analysis and categorization	RIPE Atlas, Network logs	3d - 1m
Source / Destination	packet sniffing, router monitoring, DNS redirection	zmap[e], zmap6, Logs from ISPs and IXPs, Wireshark[f], Logs of CP PoPs, BGP database	1d - 7m
CP Performace	price vs performance, end user to server, inter-AZ performence, end user to AS	Logs of CP PoPs, BGP database	7d - 30d
Packet route	hop count, comparing optimal vs real path	BGP database, traceroute	10d to 4m

Table 1: Trends in network measurement experiments *Index: DCP = Data Collection Period d = days, m = months*

[a] https://linux.die.net/man/8/traceroute
[b] https://www.ietf.org/rfc/rfc3954.txt
[c] https://atlas.ripe.net
[d] https://www.speedchecker.com/
[e] https://zmap.io
[f] https://www.wireshark.org/

4.3 Big CPs upgrade to their own infrastructure

Small businesses and CPs rely on few big CPs for providing their contents faster and from closer to the request points. Most big CPs today that has sheer large number of users worldwide, for instance over 1 billion, they relied on Akamai during the initial days and offloaded their service later [22, 14]. It makes more economical sense to own private WAN and CDNs for such huge user base. This way the businesses can have more control over the infrastructure, reduce costs in long term and dedicated lines of connections based on the requirements of different geographic locations.

4.4 Choosing CPs should depend on own necessity

The 3 major CPs (Azure, Google and AWS) have competitive service in terms of latency and throughput. They have a very good architecture and physical inter-region connections [19, 22]. Also, they peer with each other well. The decision to choose CP should primarily rely on the proximity of the server and location of majority of the users. Then the "best" and "worst" region pairs should be decided based on the experiment data. Marketed nternet data should be trusted yet verified as the performance may affect by a lot of variables not considered during the research period.

Also, "data gravity" pricing, a term used by Stoica and Shenker [29], should be considered while choosing CP. Essentially, data gravity refers to the cost that CPs charge for transferring data out of their service to some other CP. Most CPs, including AWS, have free model for ingesting data but impose 0.05-0.09$/GB for transfer, which is the cost of storing a GB of data for a couple of months [1]. Therefore, it is wise for businesses to choose CP that provide lower data gravity with other backup CP options, or at least not migrate to the CP without enough trial and validations.

5 Conclusion

This paper reviewed different network measurement experiments, analyzed them, and derived several insights. First, RTT is the most popular metric, and log analysis is the preferred methodology of many experiments. Second, Most CPs grow their businesses using the infrastructure of big CPs and establish their own infrastructure only when they have at least a billion users. Third, Recent researches prefer log analysis from PoPs and CDNs of big CPs, such as Facebook, but these data are not available to everyone all the time.

References

[1] Amazon s3 pricing. https://aws.amazon.com/s3/pricing/, last accessed on: 04.04.2022.

[2] Bismark-m-lab. https://www.measurementlab.net/tests/bismark/, last accessed on: 04.04.2022.

[3] Cloud computing services - amazon web services (aws). https://aws.amazon.com/, last accessed on: 04.04.2022.

[4] Cloud computing services | google cloud. https://cloud.google.com/, last accessed on: 04.04.2022.

[5] Cloud computing services | microsoft azure. https://azure.microsoft.com/en-gb/, last accessed on: 04.04.2022.

[6] Homepage | samknows. https://www.samknows.com/, last accessed on: 04.04.2022.

[7] Rfc 1930 - guidelines for creation, selection, and registration of an autonomous system (as). https://datatracker.ietf.org/doc/html/rfc1930, last accessed on: 27.03.2022.

[8] Rfc 3914. https://www.ietf.org/rfc/rfc3954.txt, last accessed on: 04.04.2022.

[9] Ripe atlas. https://atlas.ripe.net/, last accessed on: 04.04.2022.

[10] Ripe atlas api reference. https://atlas.ripe.net/docs/api/v2/reference/, last accessed on: 04.04.2022.

[11] Speedchecker | crowdsourcing for telecoms regulators. https://www.speedchecker.com/, last accessed on: 04.04.2022.

[12] What is cloud infrastructure? https://www.sumologic.com/glossary/cloud-infrastructure, last accessed on: 26.02.2022.

[13] The zmap project. https://zmap.io/, last accessed on: 04.04.2022.

[14] Ipv4 vs ipv6, what is ipv4, what is ipv6, ipv6 to ipv4 basis - fs.com, Mar 2022. https://community.fs.com/blog/ipv4-vs-ipv6-whats-the-difference.html, last accessed on: 03.03.2022.

[15] Todd Arnold, Matt Calder, Italo Cunha, Arpit Gupta, Harsha V Madhyastha, Michael Schapira, and Ethan Katz-Bassett. Beating bgp is harder than we thought. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pages 9–16, 2019.

[16] Todd Arnold, Ege Gürmeriçliler, Georgia Essig, Arpit Gupta, Matt Calder, Vasileios Giotsas, and Ethan Katz-Bassett. (how much) does a private wan improve cloud performance? In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 79–88. IEEE, 2020.

[17] Vaibhav Bajpai, Steffie Jacob Eravuchira, and Jürgen Schönwälder. Lessons learned from using the ripe atlas platform for measurement research. *ACM SIGCOMM Computer Communication Review*, 45(3):35–42, 2015.

[18] Robert Beverly. Yarrp'ing the internet: Randomized high-speed active topology discovery. In *Proceedings of the 2016 Internet Measurement Conference*, pages 413–420, 2016.

[19] Esteban Carisimo, Carlos Selmo, J Ignacio Alvarez-Hamelin, and Amogh Dhamdhere. Studying the evolution of content providers in the internet core. In *2018 Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–8. IEEE, 2018.

[20] Petros Gigis, Matt Calder, Lefteris Manassakis, George Nomikos, Vasileios Kotronis, Xenofontas Dimitropoulos, Ethan Katz-Bassett, and Georgios Smaragdakis. Seven years in the life of hypergiants' off-nets. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 516–533, 2021.

[21] Said Jawad Saidi, Anna Maria Mandalari, Roman Kolcun, Hamed Haddadi, Daniel J Dubois, David Choffnes, Georgios Smaragdakis, and Anja Feldmann. A haystack full of needles: Scalable detection of iot devices in the wild. *arXiv e-prints*, pages arXiv–2009, 2020.

[22] Archana Kesavan. Nanog homepage. https://pc.nanog.org/static/published/-meetings/NANOG75/1909/20190218_Kesavan_Comparing_The_Network_v-1.pdf.

[23] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. Internet inter-domain traffic. *ACM SIGCOMM Computer Communication Review*, 40(4):75–86, 2010.

[24] Aniss Maghsoudlou, Oliver Gasser, and Anja Feldmann. Zeroing in on port 0 traffic in the wild. In *International Conference on Passive and Active Network Measurement*, pages 547–563. Springer, 2021.

[25] Ciamac C Moallemi and Mehmet Saglam. The cost of latency. *SSRN eLibrary*, 2010.

[26] Ricky KP Mok, Hongyu Zou, Rui Yang, Tom Koch, Ethan Katz-Bassett, and KC Claffy. Measuring the network performance of google cloud platform. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 54–61, 2021.

[27] Lars Prehn, Franziska Lichtblau, and Anja Feldmann. When wells run dry: the 2020 ipv4 address market. In *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, pages 46–54, 2020.

[28] Brandon Schlinker, Italo Cunha, Yi-Ching Chiu, Srikanth Sundaresan, and Ethan Katz-Bassett. Internet performance from facebook's edge. In *Proceedings of the Internet Measurement Conference*, pages 179–194, 2019.

[29] Ion Stoica and Scott Shenker. From cloud computing to sky computing. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, pages 26–32, 2021.

Layer-2 Integrity Protection

Kun Ren
kun.ren@aalto.fi

Tutor: Tuomas Aura

Abstract

The majority of security protocols are based on IP protocol. However, the link layer based on MAC has not been protected carefully. In this paper, we will introduce recent research on link layer security. This paper discussed the known attacks on link layer. Then Mainly analyze and compare MACsec and SecOC, to show their solutions to different tricky problems, such as the trade-off of security and overhead. Finally, The purpose of this paper is to review recent research on link layer security and suggest suitable approaches to implement in real-world networks.

KEYWORDS: Secure Communication, MacSec, SecOC, Link Layer

1 Introduction

Many known protocols have been developed to ensure the integrity and confidentiality of data, such as IPsec, TLS, and SSH. These methods provide security at network, transport, and application layers. However, when the information is transmitted at the link layer, it is not be protected carefully. In addition, WPA, WPA2, and WPA3 were developed to secure the entire frame in the wireless network [11]. However, some weaknesses remain in the wired Ethernet networks.

It is necessary to increase security at the link layer. There are some solutions that try to address the security problems at the link layer such as MACsec, SecOC, Cryptographic Link Layer (CLL) [9] and Packet Security Protocol (PSP) [11]. The main idea is to add encryption and cryptographic integrity check to each Ethernet frame to protect its contents. Although different solutions have their own features, for example, MACsec requires new hardware support, SecOC is a in-vehicle network protocol, CLL did not provide a way to protect ARP and DHCP pakects. the encryption of each frame causes considerable computational overhead.

This paper reviews the latest approaches used for protecting the link layer. Based on standard protocols, we analyze the solutions used to prevent different threats. We also do some comparisons of different link-layer security solutions to find their advantages and disadvantages.

This paper is structured as follows. The next section describes the threats at the link layer. Section 3 analyzes the packet format of MACsec and SecOC and how those protocol works. Section 4 shows the security properties are provided to protect against those threats, Section 5 draws conclusions and suggestions for future work.

2 Threat model

2.1 Attacks

There are some known threats to the link layer, such as MAC flooding attack, DHCP attack, and ARP attacks [8]. Here are their potential threats:

MAC flooding attack

The Content Addressable Memory (CAM) table maps MAC addresses to physical interfaces. However, CAM table is limited in size. Attackers can flood the switch using a number of fake MAC addresses [13]. Then the CAM table is full, and it is unable to save new MAC addresses. As a result, the switch will broadcast the traffic that CAM table does not contain. The attacker can easily receive all the frames on the local network.

DHCP starvation attack

In the Dynamic Host Configuration Protocol (DHCP) starvation attack, an attacker sends a large number of requests to consume all the available DHCP addresses in the DHCP server. Then the attacker established a fake DHCP server to provide fake addresses to the clients. When clients send their data frames to the fake destination address, the attacker can intercept all frames.

ARP attacks

The Address Resolution Protocol (ARP) maps IP addresses to MAC addresses on the local network. Each host machine on network maintains a table. When someone is changing the ARP table without authorization. Hackers can spoof and intercept data frames, modify or stop them. then launch other attacks, such as Man-In-The-Middle and Denial of Service.

In addition, there are other attacks at link layer, such as MAC cloning and Layer 2-based broadcasting.

2.2 Threats analysis

In this paper, we assume that an active attacker uses several methods to attack our security. An active attacker means can get access to the network locally, and may also inject malicious frames into the network. Therefore, the attacker can easily exploit known weaknesses of the link layer.

In reality, the attacker may inject and modify frames that are used to run a specific functionality. For example, injecting a number of packets with different MAC addresses and building fake ARP messages. The attacker in general aims to mount the following attacks [5]:

Replay: reuse old valid frames with malicious purposes.

Tampering: spoil content of frames, so receiver cannot understand the original meaning.

Forging: generate a valid frame to activate a specific functionality with malicious purpose.

A Denial-of-Service (DoS): flooding the target with traffic, or sending it information that triggers a network crash.

2.3 Security properties

In order to mitigate those threats, a secure protocol should satisfy at least the following security properties:

Freshness: Time information of a related message. The order of a frame can be detected.

Integrity: The data is not modified when transferring. Integrity ensures data cannot be altered by unauthorized people.

Authentication: The sender of a frame can be verified and the data source is reliable.

Confidentiality: Only authorized receiver can get the content of a frame.

3 Protocol

3.1 MACsec protocol

The IEEE has proposed their solution to protect the link layer, IEEE 802.1AE. This standard is an extension of the Ethernet specification that appends a header and tail to all Ethernet frames [2]. MAC security (MACsec) provides data integrity, data confidentiality, and data origin authentication at the link layer. However, its implementation can represent a large investment in new hardware. Key management is a considerable obstacle limiting the usage of this protocol.

Packet format
Based on the standard Ethernet frame format, MACsec appends SecTAG and ICV (Integrity Check Value). When receiving MACsec frames, MACsec service will check the information in the SecTAG and the ICV to determine whether to drop or accept the frame [7]. The whole MACsec packet format is shown in Figure 1.

The length of ICV used to ensure the integrity of data is between 8 to 16 bytes depending on Cipher Suite. The SecTAG field is 16 bytes long, defined as follows:

EtherType: The length of Ethertype is 2 bytes, the value of Ethertype is 0x88e5 to indicate this frame is a MACsec frame.

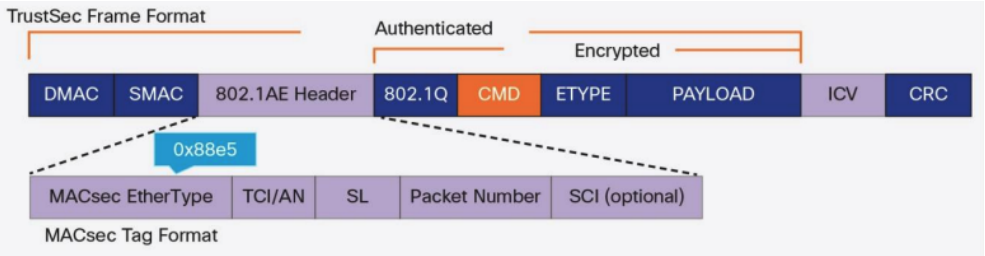


Figure 1. MACsec packect format [6]

TCI/AN: The third byte is the TAG Control Information contains several pieces of information. Version number (V), End Station (ES), SCI present (SC), Single Copy Broadcast (SCB), Encrypted payload (E), Changed Text (C) and Association Number (AN).

SL: SL means short length, which is used to indicate the length of the encrypted data.

PN: Packet Number (4 bytes) is used for replay protection and used as Initial Value (IV) for the Cipher Suite. In each MACsec frame the PN is unique, usually is an increasing number.

SCI: The secure channel identifier is 8 bytes. SCI is used to indicate to which security association (SA) the frame belongs.

How MACsec Works

MACsec implements the MACsec Key Agreement (MKA), which is a part of the IEEE 802.1XREV-2010 Port-Based Network Access Control Standard [1]. MKA provides a method to discover other MACsec stations and negotiate the security keys needed to create the link between stations.

In more detail, MACsec defines the MAC Security Entity (SecY) to operate with the MAC Security Key Agreement Entity (KaY). KaY can discover and authenticate other stations on the local network, then creates and maintains the secure relationships between stations that are used by the SecYs to transmit and receive frames [4].

The Connectivity Association (CA) created by MKA is used to connect stations. Each SecY can only participate in one CA at any one time. There are several Secure Channels (SCs) in CA that it uses to send traffic to other MACsec participants. Each channel is one-directional, as shown in Figure 2.

In the figure, station A, B and C are in the CA, and CA excludes station D. The three SCs provide secure communication among the stations.

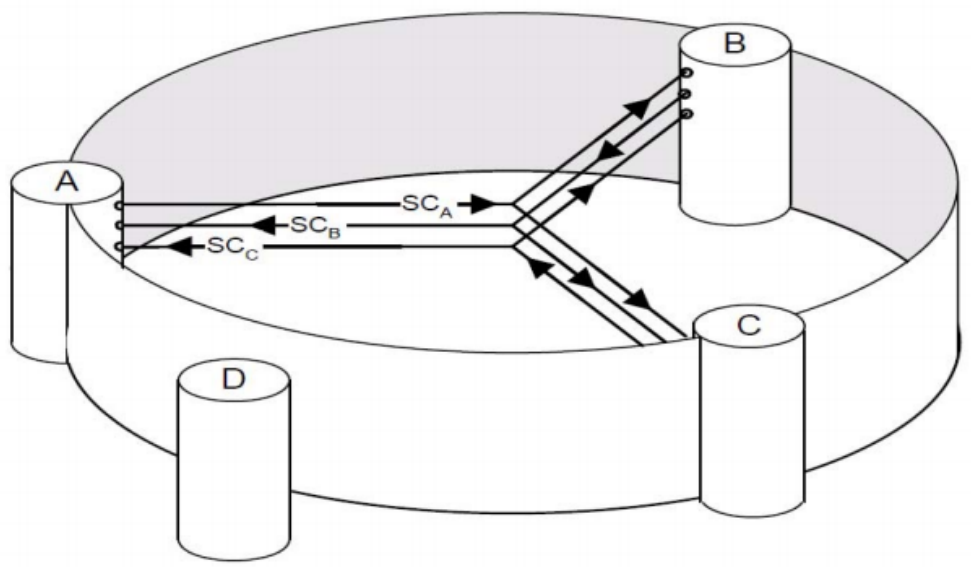


Figure 2. Secure Communication between three stations [8]

Each SC has an 8-byte secure channel identifier (SCI). The first 6 bytes of SCI match the MAC address of the device transmitting through that channel. The remaining 2 bytes are port number used to distinguish between multiple channels from the same device. Each SC may maintain Secure Associations (SA) at any point in time. Each SA contains Secure Association Key (SAK) and Counters related to packet numbers. An SA is identified through a combination of the channel's SCI and an Association Number (AN). Then the Secure Association Identifier (SAI) is created, which allows the receiving SecY to identify the SA, and the session key to be used to decrypt and authenticate the received frame [4].

3.2 SecOC protocol

In the AUTomotive Open System ARchitecture (AUTOSAR) specification for automotive networks, SecOC module is designed to minimize the overhead of the integrity protection in terms of the number of bits per frame [10]. The SecOC module provides freshness, integrity and authentication of the protocol data units (PDUs) between electronic control units (ECUs). This method requires both sender and receiver to implement a SecOC module.

The SecOC module gets freshness from an external Freshness Manager for each uniquely identifiable Secured I-PDU [3]. The packet format is shown in Figure 3:

The payload of a Secured I-PDU consists of the header, the authentic

Figure 3. Secured I-PDU packet format [3]

I-PDU, Freshness value and an Authenticator. Authenticator refers to a unique string generated by a key, identifier of the I-PDU, Authentic payload, and Freshness value. It is necessary to minimize Authenticator length when the message payload is limited in length. If truncation is possible, the Freshness and the Authenticator should only be truncated down to the most significant bits of the resulting generated by the authentication algorithm. Figure 4 shows an example:

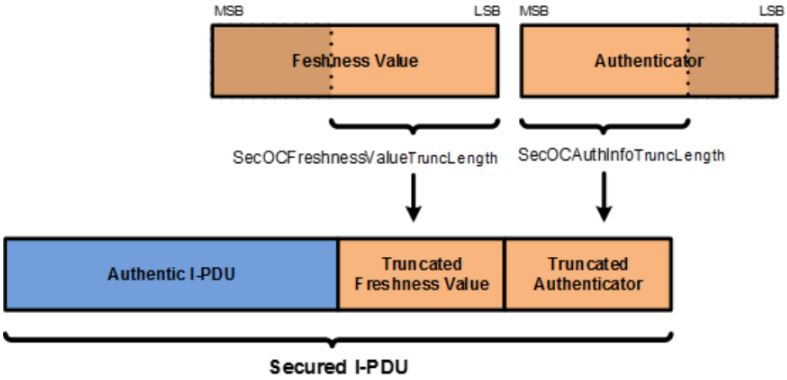


Figure 4. Secured I-PDU with truncated Freshness and truncated Authenticator [3]

In case a Message Authentication Code (MAC) is used, it is possible to transmit and compare only parts of the MAC. This is known as MAC truncation. However, this approach will decrease the secure level of Secured I-PDU. In general, MAC sizes of 64 bit and above are considered to provide sufficient protection [12].

4 Protocol analysis

4.1 MACsec protocol Analysis

Freshness: Every MACsec frame has a unique PN, which is usually increased by one for each frame. The peer receiving a frame compares its number with its acceptable PN and drops any frames with lower numbers.

Integrity: Integrity Check Value (ICV), which is generated by ci-

pher suite, guarantees packets have not been modified on the way.

Authentication and Confidentiality: MKA provides a method for discovering MACsec peers and establishing secure tunnels. MACsec uses SCs to transmit keys and identifiers. Finally, MACsec will derive a session key used to decrypt and authenticate received frames.

Latency: MACsec adds about 40 bytes of extra content in each frame and may cause a little latency. Switch needs time to encrypt and decrypt the payload for integrity. So it is necessary to implement MACsec in new hardware that can accelerate the chipper suite operation.

4.2 SecOC protocol Analysis

Similar to Macsec, SecOC Freshness Value Manager provides Freshness value to protect against replay attack. Authenticator use MAC to provide Authentication and Integrity. For latency, SecOC provides a way to minimize the size of each frame. The packets are truncated down to small part of the MAC. However, in SecOC protocol, the Key Exchange part has not been standardized.

4.3 Comparison

For similarity, Macsec and SecOC support several security perporties, such as Freshness, Authentication and Integrity. For difference, MACsec follows hop-by-hop encryption but SecOC supports end-to-end protection. Macsec can provide Multicast Broadcast and Confidentiality, because of the implementation of MKA. However, the Key Exchange of SecOC has not been standardized. So SecOC have massive numbers of keys. All in all, SecOC is Very flexible for new applications, but there are no open-source implementations. Macsec is standardized very well and can protect all traffic but it requires new hardware support.

5 Conclusion

This paper reviewed recent research about the link layer security, mainly discussing MacSec for normal wired network and SecOC for in-vehicle networks.

Firstly, this paper analyzed the main attacks of the link layer and established the threat model. Then illustrated packet format of MACsec and explained how the MACsec protocol provides serveral security perporties, such as Freshness, Integrity and Authentication. However, bigger packet size and frequent encryption would cause a visible latency. Then this paper explained how SecOC minimizes the packet size using truncation. Finally did some comparisons between Macsec and SecOC to find their advantage and disadvantages.

References

[1] Ieee standard for local and metropolitan area networks–port-based network access control. *IEEE Std 802.1X-2010 (Revision of IEEE Std 802.1X-2004)*, pages 1–205, 2010.

[2] IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security. *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, pages 1–239, 2018.

[3] Specification of secure onboard communication protocol. 2020.

[4] Hayriye Celebi Altunbasak. *Layer 2 security inter-layering in networks*. PhD thesis, Georgia Institute of Technology, 2006.

[5] Giampaolo Bella, Pietro Biondi, Gianpiero Costantino, and Ilaria Matteucci. CINNAMON: A Module for AUTOSAR Secure Onboard Communication. In *2020 16th European Dependable Computing Conference (EDCC)*, pages 103–110. IEEE, 2020.

[6] Stephen Orr Craig Hill. *Innovations in Ethernet Encryption (802.1AE - MACsec) for Securing High Speed (1-100GE) WAN Deployments*. Cisco.

[7] Sabrina Dubroca. Macsec: Encryption for the wired lan. In *netdev 1.1*. Red Hat, 2016.

[8] Narayana Raju Indukuri. Layer 2 security for smart grid networks. In *2012 IEEE International Conference on Advanced Networks and Telecommunciations Systems (ANTS)*, pages 99–104, 2012.

[9] Yves Igor Jerschow, Christian Lochert, Björn Scheuermann, and Martin Mauve. Cll: A cryptographic link layer for local area networks. In *International conference on Security and Cryptography for Networks*, pages 21–38. Springer, 2008.

[10] Timm Lauser, Daniel Zelle, and Christoph Krauß. Security analysis of automotive protocols. In *Computer Science in Cars Symposium*. Association for Computing Machinery, 2020.

[11] Reiner Augusto Campillo Terrero. A layer 2 protocol to protect the ip communication in a wired ethernet network. Rochester Institute of Technology, 2014.

[12] Jo Van Bulck, Jan Tobias Mühlberg, and Frank Piessens. Vulcan: Efficient component authentication and software isolation for automotive control networks. ACSAC 2017, page 225–237. Association for Computing Machinery, 2017.

[13] Angus Wong and Alan Yeung. Protecting network infrastructure–a new approach. In *Network Infrastructure Security*, pages 219–262. Springer, 2009.

Cryptographic privacy: an overview of identity hiding and one-sided authenticated key-exchange protocols

Alessandro Chiarelli
alessandro.chiarelli@aalto.fi

Tutor: Prof. Chris Brzuska

Abstract

Key-exchange protocols are the foundations on which secure communications are built upon. Thus, it is imperative to define and analyse secure protocols in order to protect the communications that happen every day. In order to do so, this paper defines two security properties (identity-hiding and one-way authentication) that allow us to analyse a number of different key-exchange protocols. These properties focus on different, yet similar goals on the topics of anonymity, privacy and authentication for protocols used on the network layers.

KEYWORDS: cryptography, key-exchange protocol, one-sided authentication, identity hiding

1 Introduction

The expression *online privacy* can be defined as the set of protocols, concepts and technologies that focus on protecting the identity and the information of the users of the Internet, be it individuals or organizations. A number of issues concerning privacy have been identified, such as data collection by companies to track the behavior of their users, one example being the *Cambridge Analytica* scandal [3], a political consulting company

that managed to get access to personal data of Facebook users and used it in its political campaigns.

Cryptographic primitives, such as *message authentication codes (MAC)* and *encryption*, are the foundations of digital security. They serve different purposes: *encryption* is used to conceal data to eavesdroppers (data confidentiality), and *MACs* are used to preserve data integrity. The two properties rely on keys to operate; for this reason all parties involved in a communication need to share secret keys. The goal of key-exchange protocols is to perform this operation in an efficient and secure manner.

Two security properties addressing online privacy at the network layer for key-exchange protocols [8, 4] are commonly referred to as *identity hiding* and *one-sided authentication*. In this paper, we is define these two properties and analyse exemplary key-exchange protocols with respect to these definitions. We then compare them to examine the way they satisfy either property. This paper focuses on the cryptographic properties on a conceptual level, and considers practical implementations only when necessary for contextualising their original use case.

Organization This paper is organised as follows. Section 2 clarifies the meaning of *key-exchange protocol* and defines the two security properties which are the focus of our work, i.e., *identity hiding* and *one-sided authentication*. Section 3 describes the key-exchange protocols: *SKEME*, *Noise*, *OPACITY*, *AAKE-R*, and *Ntor*. Finally, Section 4 presents concluding remarks.

2 Definitions

2.1 Key-exchange protocol

Cryptography can rely on *symmetric keys* or *asymmetric keys*. *Symmetric keys* need to be agreed upon by both parties involved prior to the communication. *Asymmetric keys* instead are unique for each party and consists of a pair of two keys, one *private* (secret) and one *public*. The two keys are related through mathematical operations that allow one to encrypt a message using the *public key* and decrypt it using the *private key*. *Symmetric key* cryptographic systems are usually faster and more computationally efficient, but *asymmetric key* cryptographic systems solve the problem of generating and distributing a shared key between all possible communi-

cation parties on the internet.

The **Diffie-Hellman key-exchange protocol (DH)** [5] is a default component in many current key-exchange protocols. It is based on exponentiation and modular arithmetics. The two parties involved in the communication, Alice and Bob, agree on a common generator g and a group, generate a key pair of *private keys* (a for Alice and b for Bob) and *public keys* (g^a for Alice and g^b for Bob). They now exchange the public shares and compute the *session key* g^{ab}. Both parties can compute the key thanks to the arithmetic properties of exponentiation. *The session key* can now be used as a *symmetric key* to encrypt and decrypt messages. The **DH** protocol is believed to be secure against passive adversaries, as it is considered extremely difficult and time consuming to compute the exponent given the result of an exponentiation and due to the *Decisional Diffie-Hellman problem* [1].

The original **DH** protocol does not support authentication as was already noted by Diffie and Hellman [5] in their original work. Thus, it is useful to additionally use a *signature scheme* to digitally sign the transmitted data using the private key. The obtained signature can then be verified using the respective public key. In addition to signing protocol messages, digital signatures can be used to generate *certificates* that bind the identity of the owner to their specific public key. *Certificates* are currently widely used, and we refer to the work by Canetti [2] for an in-depth discussion about digital signatures and certificates.

Finally, we remark that it is desirable that entities involved in a **DH** protocol use ephemeral keys whenever possible to guarantee *Perfect forward secrecy (PFS)*. *PFS* is security property which assures that session keys will not be compromised in case long-term secrets are compromised. *PFS* is not the focus of this work, but we recommend the *SKEME* [9] or *Noise* [6, 10] specifications and analysis for a discussion about it, or also the work by which the concept was originally introduced [7].

2.2 Identity hiding

If a system relies on certificates, then eavesdroppers can use them to authenticate the parties involved in the communication if the certificates are sent in plaintext. For this reason, it is desirable use an *identity hiding* key-exchange protocol. Dagdelen et al. [4] define *identity hiding* as the property that allows the two parties A and B to mutually authenticate each other and protect their identities from eavesdroppers during

the execution of the key-exchange protocol. The two parties can satisfy this property by sharing an encryption of their certificates rather than sending them in plain text format.

This technique works against eavesdroppers since the two parties first perform a key-exchange protocol to generate a shared *session key* and finally using it to encrypt the certificates. Once a certificate is received, it is first decrypted and later used to authenticate the other party. This technique does not work against active attackers, however, as one attacker would easily be able to impersonate a legitimate party in the protocol and obtain the identity of the victim.

It is possible to extend the definition of *identity hiding* to the concept of *one-sided identity-hiding*. In this situation one of the two parties sends its certificate during the key-exchange phase of the *DH* protocol and the other one sends its own certificate only after the the execution of the *DH* protocol in order to encrypt it. Thus, an eavesdropper is able to learn the identity of the first party but not of the second one.

2.3 One-sided authentication

One-sided authentication is a useful security property to be achieved whenever the two parties involved in the communication have different authentication needs or abilities. For example, one party requires to be anonymous and authenticate the other party. Goldberg et al. [8] define one-sided authentication formally. A protocol provides *one-sided authentication* if given the communication initiated by party A with B, party A authenticates the identity of B and keeps its own anonymity.

2.4 Differences between the two properties

Identity-hiding and one-sided authentication are different, as it is possible to infer from their definitions. As Goldberg et al. [8] showed in their analysis, *one-sided authentication* is desirable in a number of situations, for example whenever a patient wishes to talk to their doctor about their health without revealing their identity or whenever the responder of a communication does not need to authenticate the initiator (news sites, search engines) but the initiator does. *Identity-hiding*, instead, is desirable whenever both parties need to authenticate each other, such as in private communications, as in instant messaging or email, or online banking.

One-sided identity-hiding is useful in situations where the anonymity needs between the two parties are still asymmetric, but both wish to authenticate each other. An example could be a public message board, where the message board being public means that it does not need to hide its identity, but still needs to authenticate its users. Instead, a user accessing the website may wish to hide their identity in the key-exchange protocol.

3 Exemplary protocols

In this section, we present and analyse the key-exchange *SKEME*, *Noise*, *OPACITY*, *AAKE-R* and *Ntor* and highlight which of the previously introduced security properties they achieve.

3.1 SKEME

SKEME [9] has been designed as a standard protocol for Internet applications at a time when the IETF was working on IPSEC. Among its many components, *SKEME* offers a key-exchange protocol. The protocol consists of three phases named SHARE, Exchange (EXCH) and Authentication (AUTH). The structure and operation of each phase can be adjusted according to security and performance requirements.

The SHARE phase establishes a common secret key K_0. The basic version of the protocol relies on asymmetric encryption. Assuming Alice initiates the communication, she uses the public key of Bob to encrypt a fresh random number K_A and sends it along with her identity. Bob replies sending another encrypted fresh random number K_B along with his identity and both parties compute K_0 as the hash $K_0 = H(K_\mathrm{A}, K_\mathrm{B})$ (where H is a hash function). If the parties require anonymity, they can also encrypt their identities under the public key of the recipient of the transmitted message. The SHARE phase can be omitted if a long-term shared secret K_0 already exists between the parties, for example, in case the end devices where shipped with a shared secret key.

The EXCH phase uses the *DH* key-exchange to share public *DH* shares g^a and g^b to be used for the symmetric encryption of messages between the two parties. The EXCH phase can share encrypted nonces $nonce_\mathrm{a}$ and $nonce_\mathrm{b}$ in place of *DH* shares for faster computation, but this compromises *Perfect Forward Secrecy*, because the nonces can be recovered once the adversary learns the long-term secrets.

The AUTH phase authenticates the EXCH phase and reassures both parties that the protocol has been completed successfully with the intended peer. In order to prove their identities, the two parties A and B send the following messages that essentially work as *MACs*:

$$A \rightarrow B : F_{\mathrm{K}_0}(g^b, g^a, id_{\mathrm{A}}, id_{\mathrm{B}})$$

$$B \rightarrow A : F_{\mathrm{K}_0}(g^a, g^b, id_{\mathrm{B}}, id_{\mathrm{A}})$$

where F_{K_0} is a pseudorandom function using key K_0 and id_{i} is the identity of party i. In case the EXCH phase used nonces, they replace the *DH* shares. Each party A and B verifies the MAC they receive by performing the same computation and comparing the result to the message they received.

SKEME requires that both parties involved in the communication share their identity. If the identities are identified by certificates, then both parties can authenticate the respective peer, thus it does not offer *one-sided authentication*. In its basic operation mode, it does not offer *identity-hiding* as the identities are sent without encryption. If the identities are sent encrypted, then the protocol offers *identity-hiding* as long as the private keys of the involved parties are uncompromised and as long as the pseudorandom function used for authentication in the AUTH phase is secure.

3.2 Noise

Noise [10] is a modern protocol framework which has many different implementations, such as in Slack, Amazon AWS, WhatsApp and Wireguard [6]. *Noise* uses static key pairs for authentication and ephemeral key pairs for forward secrecy. A *Noise* session is based on messages exchanged between the two parties and they can either be **transport messages** or **handshake messages** [10]. *Handshake messages* are used to perform the key-exchange and after the completion of the key-exchange process *transport messages* are encrypted and sent between the two parties. *Handshake messages* are thus the focus of our discussion.

Noise is a very flexible protocol and it offers a number of patterns, each identified by a string of two characters [10]. **N** stands for **n**o static key, **K** stands for static key **k**nown, **I** stands for static key **i**mmediately transmitted (as soon as the first message is sent) and **X** stands for static key transmitted (**x**-mitted) during the key-exchange, but not in the first message. By combining the characters, it is possible to obtain many different

patters, such as *NN*, *KN*, *NK*, *KK*, *XX* each with its own security properties. There are two letters, one for each party. The first letter refers to the *initiator*. of the exchange and the second letter refers to the *responder*.

The simplest pattern is the *NN* pattern, where the two parties simply exchange their ephemeral keys with no authentication, thus no information about the identities of either of the parties affects the protocol messages. In the *KN* or *NK* case, one of the two parties knows prior to the communication the static public key of the other party. These variants are *identity-hiding* under the assumption that the known static key has been shared prior to the key-exchange in a secure manner. When static public keys are transmitted, they are transmitted immediately (*I* pattern) before the handshake, thus lacking identity-hiding, or after a successful completion of an ephemeral key-exchange (*X* pattern) and thus providing identity-hiding because they are encrypted. One-sided authentication is provided in all *N-* or *-N* patterns, where - stands for *K I* or *X*. We refer to the official *Noise* specification [10] for a detailed description of all possible message patterns.

3.3 OPACITY

OPACITY [4] is a protocol used in contactless environments to create secure channels. *OPACITY* is a variation of the DH key-exchange and exists in two versions called *Zero-Key Management (O-ZKM)* and *Full-Secrecy (O-FS)*. In both protocols, a terminal *T* and a card *C* generate and share session keys for authentication and encryption, but in *O-ZKM* there is no need to store static keys in the devices, which is instead required for *O-FS*. We will now focus our discussion on *O-FS*.

We refer to the two machines involved in an *OPACITY* session as terminal *T* and card *C*. *T* and *C* each possess a key pair, a certificate that includes their static public key, and a public key to verify certificates. *T* starts the session by generating an ephemeral key pair and sending its certificate and its ephemeral public key. *C* now verifies the certificate and generates its own ephemeral key pair. *C* then uses the static and ephemeral contributions of both parties to compute the session keys, encrypts its own certificate and computes a MAC over the message. *C* finally sends the message to *T*, which will verify the MAC and derive the session keys. Analogously, both parties obtained the same session key and can use it to encrypt messages.

As it is possible to deduce from the above description, this protocol does

not provide *one-sided authentication* as both parties send certificates to each other and it does not hide the identity of the terminal since it is sent in cleartext. In turn, the identity of the card is hidden, because it encrypts its certificate.

3.4 AAKE-R

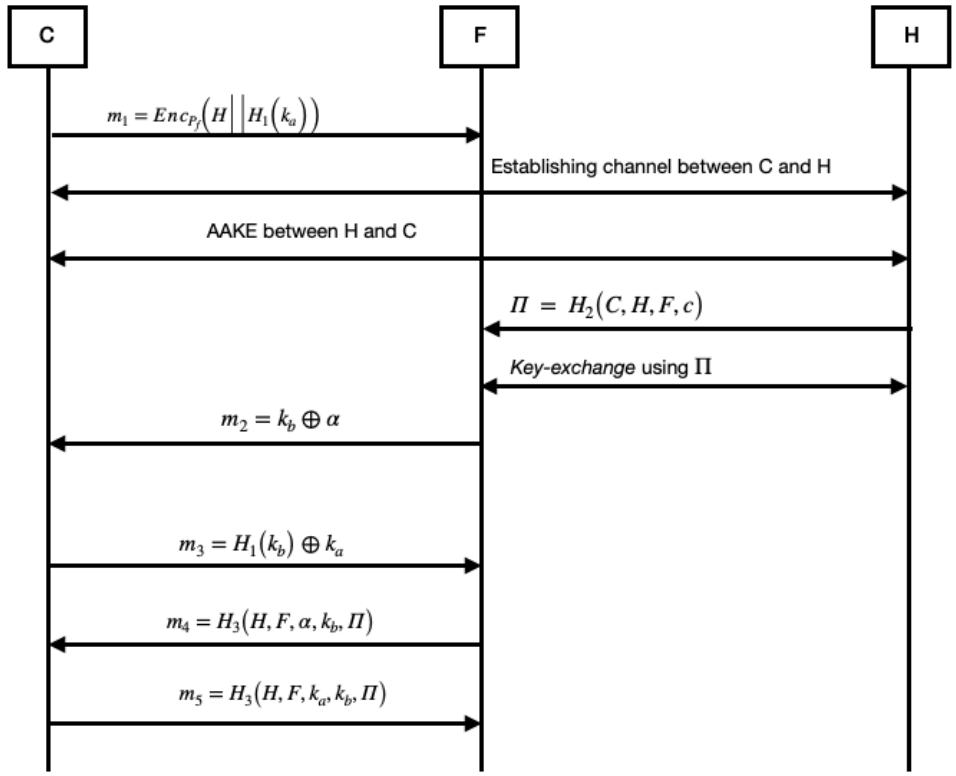


Figure 1. Sequence diagram for AAKE-R protocol

AAKE-R (*Anonymous and Authenticated Key Exchange for Roaming Networks*) [11] is a protocol originally conceived for roaming networks, such as the cellular network but it can also be adapted to other situations, such as ATM networks. *AAKE-R* is a three-party protocol where we identify three entities: the Customer, the home Server of the Customer and the foreign server *F*. In this set up, the Customer *C* pays a subscription for the service to their home server *H* and wishes to access the network through the foreign server *F*. The goal of *AAKE-R* is to allow *C* to access the network without revealing their identity to *F* in order to protect their privacy; at the same time *F* needs to be sure that it is interacting with an actual paying customer.

AAKE-R [11] is designed as follows. All entities have public keys P_{i} used by a public encryption scheme Enc and make use of three different hash functions named H_1, H_2, H_3. *C* initiates the communication and

generates a random number k_{a} and computes $m_1 = Enc_{\mathrm{P_f}}(ID_{\mathrm{H}}||H_1(k_{\mathrm{a}}))$, where "$ID_{\mathrm{H}}$" denotes the identity of the home server and "||" stands for string concatenation. We define $\alpha = H_1(k_{\mathrm{a}})$. ***F*** now decrypts m_1 and contacts ***H*** to verify that ***C*** has a valid subscription. Now ***C*** and ***H*** perform an *Anonymous Authenticated Key Exchange* (AAKE) using the channel established by ***F***. The article [11] does not specify how to perform the *AAKE*, but it clarifies that both parties authenticate each other but only the identity of ***C*** will be hidden. Thus, *AAKE* is a protocol that offers one-sided identity hiding, which is reasonable considering that the identity of ***H*** has been already sent to ***F***. ***H*** verifies that ***C*** holds a valid subscription and computes $\Pi = H_2(C, ID_{\mathrm{H}}, F, c)$, where ***C***, ***H*** and ***F*** are the identities of the three parties involved and *c* is the session key generated during the *AAKE*. Now ***H*** and ***F*** perform a key exchange using Π as a key to confirm that the subscription exists. Then, ***F*** obtains a fresh random number k_{b} and sends $m_2 = k_{\mathrm{b}} \oplus \alpha$ to ***C***. ***C*** now obtains $k_{\mathrm{b}} = m_2 \oplus \alpha$ and sends $m_3 = H_1(k_{\mathrm{b}}) \oplus k_{\mathrm{a}}$ to ***F***. Now ***F*** can compute k_{a} from m_3 and checks α using H_1. ***C*** and ***F*** can finally compute the session key as $H_3(ID_{\mathrm{H}}, F, k_{\mathrm{a}}, k_{\mathrm{b}}, \Pi)$ and they perform three confirmation steps. ***F*** sends $m_4 = H_3(ID_{\mathrm{H}}, F, \alpha, k_{\mathrm{b}}, \Pi)$, ***C*** verifies it and then sends $m_5 = H_3(ID_{\mathrm{H}}, F, k_{\mathrm{a}}, k_{\mathrm{b}}, \Pi)$. In the end, ***F*** verifies m_5 and now the protocol is completed. In the following page a sequence diagram of *AAKE-R* is presented.

AAKE-R is a protocol that shows us how the two security definitions that were defined in the prior sections do not always apply transparently to all key-exchange protocols. In this case, since *AAKE-R* is a three party protocol the definitions defined in Section 2 cannot be applied directly. Therefore, we can only affirm that the identity of ***C*** is known only by ***H***. The protocol does not offer *identity-hiding* as the identity ***H*** is revealed in plaintext and it does not provide *one-sided authentication* as all parties involved are authenticated by at least an other peer. Nonetheless, the identity of ***C*** can be considered safe since they are only revealed to the trusted home server and not to the broader network. This protocol shows that the two security properties we identified do not cover all possible scenarios.

3.5 Ntor

Ntor [8] is a protocol based on asymmetric cryptography and relies on certificates. Let us assume that we have a client *Alice* and a server *Bob*. *Bob* generates a static key pair *(b,B)* and obtains a certificate; *Alice* then

retrieves it from a trusted source. Whenever *Alice* starts a session with *Bob*, she creates an ephemeral key pair *(e,E)* to be used only once. *Alice* then sends *Bob* a message containing the ephemeral public share E, the identity of *Bob* and the protocol identifier. *Bob* follows up with generating an ephemeral key pair *(eb, EB)* and with computing both a secret key and a MAC as follows. The MAC and the secret key are computed as hashes of the ephemeral keys, the protocol identifier and the identity of *Bob*. *Bob* finally sends the MAC and his ephemeral public key to *Alice* who can finally compute the secret key and verify the MAC. If all validations succeed, then the protocol has been completed, otherwise the session is aborted.

As it is possible to deduce from the protocol description, *ntor* is a one-sided authentication protocol as *Alice* is aware of the identity of *Bob* but not vice-versa, and the protocol relies on ephemeral keys both for forward secrecy and to protect the identity of *Alice*. Therefore, *Ntor* also provides one-sided identity-hiding.

4 Conclusion

In this paper, we defined two security properties for key-exchange protocols aimed at protecting user privacy in key-exchange protocols with two parties. *Identity-hiding* and *one-way authentication* are not sufficient to cover all possible scenarios as we have seen when discussing the *AAKE-R* protocol, as this protocol is a three-party protocol and the application of these definitions is not straightforward.

We noticed how some protocols can be adaptable to different situations, and according to the variant that is currently being used its security properties differ. Examples are *SKEME* and *Noise*, as each comes with its own set of variants. *SKEME* in its basic operation mode. does not offer either *one-sided authentication* nor *identity-hiding*, whereas *Noise* has more variety when it comes to its variants and to their security properties. Other protocols such as *Ntor* can satisfy both properties but do not offer much variety when it comes to protocol variants.

We advocate for some research into new security properties protecting user privacy that cover scenarios different from the ones dealt by *one-sided authentication* and *identity-hiding*. Furthermore, this paper does not focus on implementations, and further work is needed to study said implementations and analyse if the security properties are still satisfied

even in a real world setting, since they may differ from the original standard.

References

[1] Dan Boneh. The Decision Diffie-Hellman Problem. In *IN THIRD ALGORITHMIC NUMBER THEORY SYMPOSIUM, LNCS 1423*, pages 48–63. Springer-Verlag, 1998.

[2] Ran Canetti. Universally composable signatures, certification and authentication. Cryptology ePrint Archive, Report 2003/239, 2003. https://ia.cr/2003/239.

[3] Nicholas Confessore. Cambridge Analytica and Facebook: The Scandal and the Fallout So Far. *The New York Times*, April 2018. ISSN 0362-4331.

[4] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzurra Marson, Arno Mittelbach, and Cristina Onete. A cryptographic analysis of opacity. In *Computer Security – ESORICS 2013*, pages 345–362, 2013.

[5] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, volume 22, number 6, pages 644-654, 1976.

[6] Benjamin Dowling, Paul Rösler, and Jörg Schwenk. Flexible Authenticated and Confidential Channel Establishment (fACCE): Analyzing the Noise Protocol Framework. In *Public-Key Cryptography – PKC 2020*, pages 341–373, 2020.

[7] Christoph G. Günther. An identity-based key-exchange protocol. In *Advances in Cryptology — EUROCRYPT '89*, pages 29–37, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[8] Douglas Stebila Ian Goldberg and Berkant Ustaoglu. *Anonymity and one-way authentication in key exchange protocols*, 2012. https://www.cypherpunks.ca/ iang/pubs/ntor.pdf.

[9] H. Krawczyk. Skeme: a versatile secure key exchange mechanism for internet. In *Proceedings of Internet Society Symposium on Network and Distributed Systems Security*, pages 114–127, 1996.

[10] Trevor Perrin. *The Noise Protocol Framework*. Noise Protocol, 2018. Revision 34. https://noiseprotocol.org/noise.pdf.

[11] Guomin Yang, Duncan S. Wong, and Xiaotie Deng. Anonymous and authenticated key exchange for roaming networks, 2007. In *IEEE Transactions on Wireless Communications*, volume 6, number 9, pages 3461-3472.

A Survey of Deterministic Networking

Aapo Linjama
aapo.linjama@aalto.fi

Tutor: Miika Komu

Abstract

The need for deterministic network services is growing beyond the classical cases such as industrial automation and control systems. Deterministic network services provide boundaries on latency and guarantee ultra-low packet loss. The Deterministic Networking Working Group was set up to respond to these needs and create standards for implementing such networks from Layer 3 perspective. The goal of this paper is to provide a survey of the Deterministic Networking standard family and to give the reader a basic understanding of the topic. The survey of this standard is organised to cover the main features of the DetNet network, the security aspect and the data plane features. Functionalities such as resource allocation, service protection and explicit routes are covered as well as few security threats imposed on the DetNet network. It is worth mentioning that the DetNet standard is still a work in progress and some features covered here might change in future.

KEYWORDS: deterministic network, time-sensitive networking, deterministic networking, real-time application

Terminology

App-flow	The native data flow between the source and destination end systems within a DetNet enabled network.
DetNet flow	A sequence of packets that conforms uniquely to a flow identifier and to which the DetNet service is to be provided. It includes any DetNet headers added to support the DetNet service and forwarding sub-layers.

1 Introduction

Over the last 30-40 years, the Internet has not been the only digital network that has experienced soaring growth. Applications that have critical requirements on timing and reliability such as Industrial Automation and Control Systems (IACSs) have traditionally used special-purpose fieldbus technologies to meet the requirements [6]. However, these technologies are not part of a packet-switched network. Deterministic Networking (DetNet) [4] aims to enable the migration of these applications to packet-switched networks and at the same time it aims to support the existing packet network applications over the same physical network.

This document provides an overview of parts of the DetNet family of standards. It should help the reader to become familiar with the key Level 3 solutions described in the standards that enable deterministic networks.

The structure of the paper is as follows. Section 2 introduces Open System Interconnection (OSI) model, deterministic networks, Time-Sensitive Networking (TSN) and presents essential terminology. Section 3 presents Deterministic Networking (DetNet), a set of standards by the IETF DetNet Working Group. Section 3 focuses on describing main features of DetNet, some of its security aspects and data plane functionalities. Finally section 4 presents the conclusion about the features covered and the state of DetNet standardization.

2 Background

2.1 Open System Interconnection (OSI) model

The Open Systems Interconnection model (OSI model) [9] is a conceptual model that characterizes and standardizes a system's communication functionalities regardless of its underlying hardware and technology. Its purpose is to ensure that different communication systems can communicate with each other using standard communication protocols. The OSI model divides a communication system's data flow into seven abstraction layers, each with its own functionalities related to data transportation and transformation.

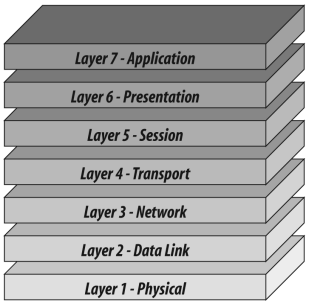


Figure 1. Open Systems Interconnection (OSI) model [9]

Figure 1 represents the layers of the OSI model. This paper focuses on Data Link (Layer 2) and Network (Layer 3) layers as Time-Sensitive Networking (TSN) standards define deterministic data paths for networks operating in Layer 2 and Deterministic Networking (DetNet) focus on deterministic data paths in Layer 3. Layer 3 has core functionalities such as adding routing information to messages, choosing the best transmission facilities, breaking the messages into packets and reassembling them back into messages [9]. Layer 2 has core functionalities such as formation of the message for transmission and handling error control and point-to-point synchronization over the physical link.

2.2 Deterministic Network

A deterministic network [10] is a network that can reliably and predictably deliver services between network users. It differs from more traditional network solutions in that it provides a guaranteed upper and lower limit for delay. In addition, a deterministic network aims to reduce loss, occurring due to congestion, to zero. Congestion loss refers to a situation where

the buffer of a network resource such as a router is overflown and packets are dropped for this reason [3]. This paper focuses on the IETF Deterministic Networking standard body which defines the design principles of deterministic networks from a Layer 3 perspective.

2.3 Time-Sensitive Networking (TSN)

Time-Sensitive Networking (TSN) refers to a set of standards which are defined in IEEE 802 standard family. TSN Task Group [2] aims to define deterministic data paths for IEEE 802 networks which operate in Layer 2. TSN network has guaranteed packet transport, low packet loss, bounded low latency and low packet delay variation [10]. The TSN Task Group is part of IEEE 802.1 Working Group.

TSN network has three essential features: time synchronization, contracts between transmitter and the network and coexistence with other network services [3]. Requirement for time synchronization means that internal clocks of all network devices and hosts can be synchronized to an accuracy of 1 µs to 10 ns. Precision Time Protocol is used to achieve synchronization. Contracts between transmitter and the network are the second essential feature. It means that every TSN stream, which is defined as a flow of data from a transmitter to a receiver [10], is subject to a contract between the transmitter and the network. With this contract TSN network can provide bounded low latency and low packet loss. Part of the contract is also the possibility for the TNS network to send copies of the same data frames over multiple paths. This way single equipment failure does not cause a packet loss. Contracts are also flexible which means that new ones can be created and old ones terminated for different TSN streams.

3 Deterministic Networking (DetNet)

As the TSN standard progressed there was also a need for expanding TSN concepts to cover Layer 3 routed technologies. In 2015, Internet Engineering Task Force (IETF) created a Working Group called Deterministic Networking (DetNet) [3]. DetNet Working Group is focusing on Layer 3 routed segments and their goal is to create deterministic data paths that have bounds on latency, packet delay variation and exceedingly low packet loss. DetNet Working Group and the IEEE 802.1 TSN Task Group have

a tight working relationship in order to define a common architecture for layers 2 and 3. Initially DetNet Working Group is working on solutions that cover campus-wide networks and private WANs but providing solutions for larger networks such as the Internet is not in their scope [1].

3.1 Features of DetNet Network

Similar to TSN, DetNet has common characteristics such as time synchronization, bounds on latency, packet loss and high reliability [14]. DetNet uses three techniques to achieve the QoS defined by those characteristics. These techniques are: Resource allocation, Service protection and Explicit routes [5].

Resource allocation
Resource allocation means that a certain amount of resources are allocated to each DetNet flow in the network. These resources are, for example, buffer space and link bandwidth. By allocating enough resources, the congestion loss of packets in the DetNet network is minimized while enabling the network to define upper boundary for the end-to-end latency.

Service Protection
Equipment failures are an important factor when considering packet loss. DetNet addresses this factor with a mechanism called Service protection. The main mechanisms that implement the service protection are packet replication, elimination and encoding. Packet replication mechanism sends multiple copies of one DetNet flow through different paths in the network. This way it is ensured that if the DetNet flow loses one path, caused for example by equipment failure, it does not lead to packet loss. Elimination of duplicate packets are also handled by DetNet network in the event that multiple packets arrive via different paths to the destination. Packet encoding mechanism uses network coding algorithms to merge multiple DetNet packets into one. The accumulated packet is forwarded to the destination where the packet is decoded back to its original packet components. This way DetNet packet can be encoded to be part of multiple transmission units.

Explicit routes
In networks which are controlled by routing protocols such as Intermediate System to Intermediate System (IS-IS) or Open Shortest Path First (OSPF), a network event such as failure or recovery of a node can reflect

and affect other nodes of the network that are far from the original source of the event. However, a DetNet network employs explicit routes, where the route will not change even if there are errors or other events on the network that could affect the routing. When explicit routes are combined with service protection mechanisms, the DetNet network can provide a high probability of interrupt free data delivery.

3.2 DetNet Data Plane

The role of the data plane is to carry App flows over the DetNet network [12]. The data plane functions connected to DetNet are split into two sub-layers: a service sub-layer and a forwarding sub-layer [5]. These layers implement the DetNet functionality in the network and provide DetNet services to other layers in the protocol stack.

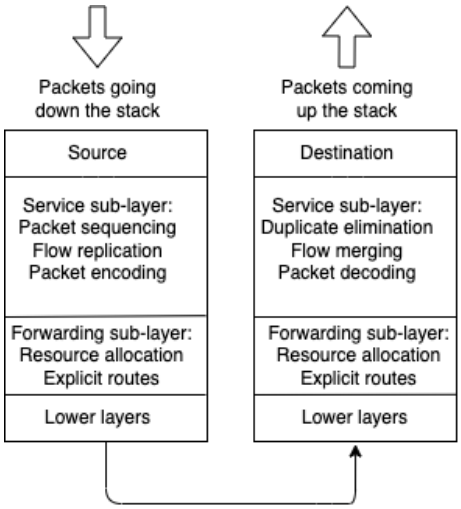


Figure 2. DetNet data plane layering model [5]

Figure 2 represents the DetNet data-plane layering model, however not all layers and functionalities are required for every DetNet node in the network. The concepts represented in the figure are:

Application
Source and destination represent application end systems capable of originating and terminating a DetNet flow.

Packet sequencing
Packet sequencing function provides the sequence number for packets. This functionality enables packet replication and elimination. Duplicate elimination functionality is the peer of packet sequencing.

Duplicate elimination
Duplicate elimination ensures that replicated packets are eliminated. This elimination is based on sequence number provided by the packet sequencing feature and is part of DetNet service protection (Section 3.1).

Flow replication
Flow replication functionality splits DetNet flow into multiple parts and duplicates these parts. This functionality is part of DetNet service protection.

Flow merging
Flow merging functionality merges the separated parts of DetNet flow back to one flow. This functionality is separate from duplicate elimination but also part of DetNet service protection feature.

Packet encoding
Packet encoding functionality combines the data of multiple packets belonging to DetNet flow by using an encoding algorithm. Combined packets can originate even from different DetNet flows. Packet encoding is part of DetNet service protection feature.

Packet decoding
Packet decoding functionality takes as an input the packets encoded by packet encoding functionality and computes the original packets from them using a decoding algorithm. Packet decoding is the peer of packet encoding and also part of DetNet service protection feature.

Resource allocation
As described in section 3.1 resource allocation functionality ensures that enough resources such as bandwidth and buffer space are reserved at each DetNet node in the network.

Explicit routes
As described in section 3.1 DetNet employs explicit routes, which ensures with other service protection mechanisms that the probability of continuous connectivity is high in the DetNet network.

3.3 Security considerations of DetNet

As DetNet technologies enable deterministic flows over wider networks, the attack surface grows wider too. Applications for DetNet include for

example power grid devices, industrial and building control systems [7]. Vulnerabilities in these applications can be very costly and critical, which is why they are also potential targets for cyber attackers. Therefore security is also a very important factor when designing a DetNet system or component. Another important consideration is that the DetNet network can be built utilizing MPLS [13] and/or IP [11] technologies, and hence inherits the security features of them.

Examples of security threats

In this section three examples of security threats and their impacts are discussed. The threats discussed are flow modification or spoofing and path manipulation.

In DetNet flow modification [7], an attacker manages to modify headers or body fields of packets in order to cause, for example, incorrect routing or dropping of packets. Spoofing on the other hand refers to a situation where an attacker injects traffic to the network which is tailored to look like part of a legitimate DetNet flow. Successful spoofing of packets can increase buffer use and processing utilization in routers across the network, resulting in higher resource consumption [7]. This in turn can lead into increased delay caused by resource exhaustion. In the case that the attacker successfully creates valid headers for the packets, legitimate DetNet flow packets can be dropped as the router resources are exhausted.

Because DetNet relies on an underlying time-synchronization system, a faulty synchronization mechanism caused by an attacker might lead DetNet nodes to fail. DetNet flows, in particular, may fail to achieve their latency and predictable behavior criteria, resulting in denial-of-service (DoS) to DetNet applications [8].

4 Conclusion

As the need for deterministic data flows, which provide bounded latency and high reliability, grows in different industries it is certain that DetNet and other time-sensitive networking standards are important enablers of networks capable of delivering such services. DetNet aims to provide deterministic data paths over Layer 3 and enable time-critical systems such as industrial control systems to migrate from special purpose technologies to packet networks.

The three main techniques that DetNet has to achieve its quality of service were covered in the paper. These techniques are resource allocation, service protection and explicit routes. Resource allocation allows the DetNet network to minimize packet congestion loss while allowing the network to define an upper limit for end-to-end delay. This is possible with sufficient resource allocation. Service protection aims to minimize packet losses caused by equipment failures. The main mechanisms that implement the service protection are packet replication, elimination and encoding. Explicit routes are static routes for the packets that do not change in the event of equipment failure.

Paper also described the data plane features of the DetNet. The data plane covers the aspects that are needed to carry App flows over the DetNet network. The DetNet data plane functionalities are split into two sub-layers: service sub-layer and forwarding sub-layer. These layers implement the features such as resource allocation, service protection and explicit routes. However, not all nodes in the network need to implement both layers.

Security aspect of the DetNet was covered in the paper. It is an important factor in the DetNet as its use cases involve many critical systems. Examples of security threats in DetNet network are flow modification and spoofing in which an attacker modifies packets or injects packets to the network in order to cause incorrect routing or dropping of packets.

This paper has covered parts of the DetNet standard, such as its main features, data plane and security aspects. These sections were chosen because they provide the reader with a basic understanding of the topic, enabling the reader to explore the subject in more detail. However, it should be noted that the DetNet standard is still work in progress and many aspects are still in fairly early stages of development.

References

[1] Deterministic networking (detnet). https://datatracker.ietf.org/wg/detnet/about/. Accessed: 2022-01-30.

[2] Time-sensitive networking (tsn) task group. https://1.ieee802.org/tsn/. Accessed: 2022-01-30.

[3] Norman Finn. Introduction to time-sensitive networking. *IEEE Communications Standards Magazine*, 2(2):22–28, 2018.

[4] Norman Finn and Pascal Thubert. Deterministic Networking Problem Statement. RFC 8557, May 2019.

[5] Norman Finn, Pascal Thubert, Balazs Varga, and János Farkas. Deterministic Networking Architecture. RFC 8655, October 2019.

[6] Ethan Grossman. Deterministic Networking Use Cases. RFC 8578, May 2019.

[7] Ethan Grossman, Tal Mizrahi, and Andrew J. Hacker. Deterministic Networking (DetNet) Security Considerations. RFC 9055, June 2021.

[8] Tal Mizrahi. Security Requirements of Time Protocols in Packet Switched Networks. RFC 7384, October 2014.

[9] Deborah Russell and G. T. Gangemi. *Computer Security Basics*. O'Reilly Associates, Inc., USA, 1991.

[10] B. Varga, J. Farkas, L. Berger D. Fedyk, and D. Brungard. The quick and the dead: The rise of deterministic networks. https://www.comsoc.org/publications/ctn/quick-and-dead-rise-deterministic-networks, 2021. Accessed: 2022-01-30.

[11] Balazs Varga, János Farkas, Lou Berger, Don Fedyk, and Stewart Bryant. Deterministic Networking (DetNet) Data Plane: IP. RFC 8939, November 2020.

[12] Balazs Varga, János Farkas, Lou Berger, Andrew G. Malis, and Stewart Bryant. Deterministic Networking (DetNet) Data Plane Framework. RFC 8938, November 2020.

[13] Balazs Varga, János Farkas, Lou Berger, Andrew G. Malis, Stewart Bryant, and Jouni Korhonen. Deterministic Networking (DetNet) Data Plane: MPLS. RFC 8964, January 2021.

[14] Xiaotian Yang, Dominik Scholz, and Max Helm. Deterministic networking (detnet) vs time sensitive networking (tsn). 2019.

Detecting Anomalies in Firewall Configurations

Zetong Zhao
zetong.zhao@aalto.fi

Tutor: Tuomas Aura

Abstract

A firewall is a network security device that monitors incoming and outgoing network traffic and permits or blocks data packets based on a set of security rules. It is vital to guard the network boundary and defend against malicious invasion. A firewall plays its role mainly based on the configuration of the firewall rules, which guide the behaviour of the firewall. Nowdays, the firewall rules grow more and more complicated due to the development of network environment and attack technologies. In such cases, unnoticeable flaws usually occur within complex firewall configuration tables, which may degrade the performance of the firewall or may be exploited by attackers. Facing this situation, this paper discussed common firewall anomalies and corresponding detection methods. Among these methods, the paper focuses on a static detection method and presents a thorough discussion of that.

KEYWORDS: Firewall, access control, policy anomaly management

1 Introduction

With the development of information technology, Internet has already become an integral part of our lives. While enjoying the convenience it

brings, people also need to bear the corresponding risks and threats. As a guardian in the information world, the firewall significantly eliminates the potential dangers by preventing suspicious packets from entering the private network, namely the intranet. The intranet controls the incoming and outgoing data based on the administrator's preset security rules. These rules fundamentally decide the quality of a firewall. Carefully configured rules can enable the firewall to effectively filter malicious packets to the maximum extent. On the other hand, inappropriate rules may lead to anomalies and conflicts. These vulnerabilities can be cleverly utilized to bypass the firewall.

The Internet is known as an extremely complex system. Due to this, managing security rules is a challenging task. The fast-evolving internet services and communications further aggravate the difficulties in the policy configuration. As Al-Shaer [3] stated that all the tested firewall configurations contain a variable amount of anomalies and conflict. The participated administrators even included nine experts. Furthermore, Wool's study in 2010 [6] confirmed his conclusion since 2004 that the firewalls of companies are usually poorly configured, and the complexity of the rules positively influenced the number of errors.

To address this issue, this paper describes and implements a specific anomaly detection algorithm based on the study of different types of anomalies and common detection methods.

The paper is organized as follows. Section 2 elaborates on the different kinds of anomalies. Section 3 overviews the common anomaly detection methods. Sections 4 describe a detection algorithm in detail. Finally, the paper is concluded in Section 5.

2 The types of anomalies

A firewall policy is in fact a set of firewall rules that specify the target network packet and corresponding behaviour. The rules are defined in the form of <condition, action>, in which the condition is a sequence of fields that describe the target field. It usually contains attributes that related to the characteristics of network packet, such as protocol, source IP and destination IP. A typical firewall policy is shown in Table 1, which demonstrate the vital elements of a firewall policy.

According to studies[2] [7], firewall policy anomalies could be classified into the following categories.

Rule	Source IP	Source Port	Destination IP	Destination Port	Action
r1	156.10.2.*	*	192.26.1.*	52	deny
r2	156.10.*.*	*	192.26.1.*	52	deny
r3	156.10.*.*	*	192.167.*.*	26	allow
r4	156.10.1.*	*	192.167.1.*	26	deny
r5	156.10.1.*	*	*	*	allow

Table 1. An Example of Firewall Rules

2.1 Shadowing

Just like the word "shadow" means, the shadowing anomaly means certain firewalls rules are shielded by their previous rules. If the all the packets matched by the rule are also matched by its preceding rules but with different actions, the rule will be regarded as being shadowed. In this case, all the packets that should be processed by the rule will be intercepted by its preceding rules, and the rule will never take effect. In Table 1, rule r4 is shadowed by r3, since r3 allowed packets with source IP 156.10.1.* and destinations IP 192.167.1.* while these packets are supposed to be denied according to r4.

The shadowing anomaly is usually considered as a severe flaw in firewall construction because it will lead to the block of permitted packets and vice versa.

2.2 Generalization

For a certain rule, if a subset of the packets matched by this rule is also matched by the preceding rules but with actions, the rule will be regarded as a generalization of the preceding rules. In Table 1, the rule r5 is a generalization of r4, since packets defined in r4 is a subset of the packets defined in r5 and the action executed by r4 and r5 is different.

2.3 Correlation

A rule is correlated with other rules if the packets defined by these rules are overlapped. And these rules lead to different actions. In this case, the intersected packets allowed by one rule may be denied by another rule. In Table 1, the rule r2 is correlated with r5. Packets from the source IP 156.10.1.* to the port 52 in the destination IP 192.26.1.* that matched

by r2 is also within the range of packets defined in r5. And r2, r5 has opposite strategies for these packets.

2.4 Redundancy

A rule is regarded as redundant if its job is completely done by another rule, and the deletion of the rule will cause no difference. In Table 1, the rule r1 can be considered as redundant since both r1 and r2 will deny the user datagram protocol (UDP) packets come from IP 156.10.2.* to port 52 on IP 192.26.1.*.

3 Common anomalies detection methods

Over the past few years, the detection methods for the firewall anomalies have been widely studied. Many kinds of strategies have been proposed from different aspects. Al-Shaer et al. [1] used a single rooted tree to model the firewall rules. The tree makes it easier and more straightforward to show the relationship between rules and helps to discover the relations and anomalies among the rules. Within the tree, the node is used to represent each field of the filtering rule, and the branch is used to represent a possible value of the associated field. Specifically, the protocol field of a rule will be represented by the root node, and the action filed will be represented by the leaf node. Other fields, such as source IP, source port, destination IP and destination port will be represented by the nodes between the root node and leaf nodes. In this way, a completed policy rule can be represented by a tree path beginning from the root node and terminating at a leaf node. If some rules possess the same field value at a specific node, they will share the same branch representing that value. Using this approach, the detection of anomalies can be fulfilled on the basis of pairs of filtering rules. However, if too many rules is recorded in the configuration file, the process procedure could be very long.

Yuan et al. [7] demonstrated how to utilize static analysis to detect firewall anomalies. Based on this approach, they constructed a tool named FIREMAN, which can effectively discover the violations of user-specified security policies and the inconsistencies as well as inefficiency among firewall rules. FIREMAN use a compact representation and translated the firewall rules into this format according to the operational semantics of a firewall. In a single firewall, a certain access control list (ACL) will

be translated into a rule graph, while in distributed firewalls, it will be translated into an ACL-tree with additional information about network topology. The anomalous configurations can be checked according to the rule graph and ACL-tree. However, there are also limitations of FIREMAN when detecting anomalies. For each firewall rule, only its preceding rules will be examined by FIREMAN and all the following rules will not be checked. Apart from that, when detecting anomaly, FIREMAN can only demonstrate whether a misconfiguration occurs between one rule and its preceding rules, and cannot indicate all rules related to the anomaly accurately.

Facing the complexity of policy anomalies and the difficulties in changing the conflicting rules, Hu et al. argues in [4] that to cope with conflicting issues, we should be able to to identify which rule involved in a conflict situation should take effect preferentially when multiple conflicting rules can match a particular network packet simultaneously. In practice, to cope with this conflict, a firewall primarily adopts the first-match resolution mechanism based on the order of rules. However, there are limitations when applying the first-match strategy to cope with policy conflicts. When a conflict occurs, the first matching rule chosen may not be the ideal rule that should take precedence to resolve the conflict. Therefore, it is necessary to find a first-matching mechanism in the firewall to bridge the gap between conflict detection and conflict resolution. In this case, Hu et al. proposed a new firewall anomaly management framework based on rule segmentation technology, which can not only achieve more accurate anomaly detection, but also achieve effective anomaly parsing.

Based on logic programming, Cengiz et al. proposed a novel firewall policy anomaly detection (FPAD) module and also developed a corresponding software tool to define firewall rulesets in [5]. The tool will automatically convert rule sets into logical programming constructs and discover anomalies inside firewalls by using FPAD implemented in Prolog. The FPAD module is composed by multiple parts. The firewall administrator can define, update, or import firewall ruleset using a client-side GUI. The ruleset will be saved in the MySQL database management system for persistent storage. Then, utilizing the provided ruleset, the Prolog engine will detect the potential anomalies, and the detected anomalies will be returned to the base tool and then forwarded to the client-side.

4 The description of the static analysis detection methods

This section discusses a static analysis method to detect the firewall configuration anomalies. The primary idea of this method is to represent the filtering rules with a tree shaped data structure called Firewall Exception Tree (FAT) and then supervise the leaves to detect anomalies. The tree can show if there are overlaps between different rules, and these overlaps are treated as exceptions.

4.1 Preprocessing firewall rules

It is difficult to construct the FAT directly from the firewall rules. Since firewall rules and a FAT has very different structure, the former is composed of conditions and action while the latter is composed of nodes and edges. Therefore, we need to first define some auxiliary structures to help constructing the FAT from firewall rules.

The normalization of firewall filtering rules
Observing the construction of firewall rules, we can find their condition part, namely the source IP, source port, destination IP and destination port attributes can in fact be represented by four categories of value.

1. A bit string, such as IP range like 127.20.0.0/16.
2. A single value, such as a port 80.
3. A set of value, such as a port set {80, 110}
4. A range, such as a destination port domain [80...1024].

Since all the values above are numeric. Therefore we can denote them in the binary format, and then simplify that to the form $(byte, mask)_b$, in which the byte represents a value, and the mask represents the number of fixed bits. For example, $(4, 8)$ represents the value 4 since the value of byte filed is 4 and all the bits of that byte are fixed. And $(8, 5)$ represents the range [8...15], since the value of a byte is 8 and only five bits of that byte are fixed, the rest bits denote the range from zero to seven. Concerning the above four types of data, they can be represented as follows using the $(byte, mask)_b$ format.

1. For the bit string such as 127.20.0.0/16, it can be represented by a $(byte, mask)_b$ tuple such as $(127, 8)_b$, $(20, 8)_b$, $(0, 8)_b$, $(0, 8)_b$.

2. For a port like 80, it can be represented as $(80, 8)_b$.

3. For a set of value such as $[127.20.0.2, 127.20.0.3]$, it can be transformed to $(127, 8)_b, (20, 8)_b, (0, 8)_b, (2, 7)_b$.

4. A range such as $[32...63]$ can be represent as $(32, 3)_b$.

In this paper, the data in form of $(byte, mask)_b$ is called **masked byte**. For a masked byte with a mask equal to 8, it is named as **complete byte** since it represents a certain value. For the masked byte with a mask less than 8, it is named as **partial byte** since it in fact denotes a range.

The rule paths of firewall filtering

After the normalization process, the firewall filtering rules can be transformed into a series of masked bytes. However, this is not enough, since different fields of a filtering rule may have the same masked bytes. For example, if the source IP and the destination IP both begin with 192, then they will have the same $(192, 8)_b$ byte mask. It will disturb the construction of a FAT. To cope with this issue, we introduced a new pair of integers called **position** to denote the position of each field within a firewall rule. Therefore, to represent a field of a firewall rule precisely, we need both the byte mask and position. And the combination of them is called an **element**.

The position of a masked byte is denoted by two flags. The first one is called dim, which defines the sequence position of a certain field in a firewall rule. And the other one is called ord, which represents the byte order in the field. Therefore, a certain position is represented as $(dim, ord)_p$, in which the p is a symbol to denote it defines a position, to distinguish it from the masked byte. For a certain firewall rule, it has four fields, namely source IP, source port, destination IP and destination port. They correspond to the dim numbers 1, 2, 3 and 4 respectively. In this way, for the second element of the rule R1 in Table 1, it can be depicted as $((10, 8)_b, (1, 2)_p))$.

We have defined the structure of an element, however, it is not enough to just obtain a series of elements. We also need a method to arrange them in a certain order, which requires defining a rule for comparing these elements. Such a rule is described as follows:

For two elements $e_1 = ((v_1, m_1)_b, (d_1, o_1)_o)$ and $e_2 = ((v_2, m_2)_b, (d_2, o_2)_o)$, we say $e_1 \leq e_2$ if the two elements satisfy one of the following relationships:

1. $m_2 < m_1$ && $m_1 = 8$.
2. $((m_1 = m_2 = 8)$ || $(m_1 < 8$ && $m_2 < 8))$ && $(d_1 < d_2)$
3. $((m_1 = m_2 = 8)$ || $(m_1 < 8$ && $m_2 < 8))$ && $(d_1 = d_2)$ && $(o_1 < o_2)$

Using these comparing rules, we can sort the elements of the firewall

rules in a certain order. For example, for the first rule in the Table 1, it can be represent by ordered elements like $[(156, 8)_b, (1, 1)_p]$, $[(10, 8)_b, (1, 2)_p]$, $[(2, 8)_b, (1, 3)_p]$, $[(192, 8)_b, (3, 1)_p]$, $[(26, 8)_b, (3, 2)_p]$, $[(1, 8)_b, (3, 3)_p]$, $[(0, 8)_b, (4, 1)_p]$, $[(52, 8)_b, (4, 2)_p]$.

4.2 The Construction of Firewall Anomaly Tree

Using the aforementioned methods, we can represent a firewall rule with a sequence of ordered elements. These elements can be used to construct the FAT, this section discussed the data structure of a node and some essential methods used for building the tree. For the following discussion, we defined a function named **Domain** to specify the domain of a filtering rule.

- **Domain(Set RS)**: RS means a set of rules. The domain of firewall rules represents the set of all the possible network packets that could be processed by these rule, representing a range.

The node object

The FAT is composed by nodes and edges. A node represents one position that exits in the path of one or more filtering rules, it can be defined as an object which contains the following attributes:

1. id. An identifier to a node.
2. Position. The current position of the node.
3. Type. There are four possible types for a node, which are F, C, P and PC. F denotes this is a leaf node. C denotes all the labels of the output edges are complete bytes, P denotes all the labels are partial bytes and PC denotes the mix of complete bytes and partial bytes.
4. Edge. An array indicates the output edges from N. Its range is from 0 to 256.
5. P refers to primary rules, which in fact is a set of candidate filtering rules
6. S refers to secondary rules, which in fact is a set of filtering rules such that Domain(S) $\subsetneq$ Domain(P).
7. T refers to tertiary rules, which in fact is a set of filtering rules that Domain(P) $\cap$ Domain(T) is not null, Domain(T) $\not\supset$ Domain(P) and Domain(P) $\not\supset$ Domain(T).
8. bptr. This field is a pointer, referring to the parent node of N.

9. bval. This is the edge label, between N and its parent node.

According to the above specification, a root node should have the identifier 1, a set P containing all filtering rules, two empty sets S and T, along with the empty bptr and bval fields.

Construct a child node

Generally, a tree structure should be constructed recursively based on an initial condition, which refers to the root node in our case. This section discusses how to construct a child node. To construct a child node, we have to first create a new node and then assign a new identifier to its id field. And the fields bptr and bval are also easy to compute if we know the edge from the parent node to the child node. For the remaining fields, we defined six functions to calculate the value needed for building a child node and they are classified into three categories as follows:

Computing the position of nodes

- **minpos(Set RS)**. RS denotes a set of firewall rules, which should have been converted to a group of elements. This function is used to find out the smallest element using the predefined comparing rule and return its position. For example, if the RS corresponds to the rule R1 and R2 in the Table 1, the function should return $(1, 1)$.

- **nextpos(Set RS, Position p)**. The function takes in a set of firewall rules and a position. The position is extracted from a certain element e1. What this function does is to find out the next element of e1 using the predefined comparing rule and return its position. For example, if the function takes rule R1 and R2 as the RS parameter, and $(1, 1)$ as the position. The returned value should be $(1, 2)$.

Calculating the labels of edges and the type of node

- **proj(Set TS, Position p)** Taken a set of firewall rules and a position p, this function will find out all the elements that match the position p and return their byte masks as a set. The return set will be used to define the type of node. If the set is empty, the type of the node should be F (Final), indicating that this is a leaf node. If all the returned byte masks are partial, the node should have the type P (Partial). If all the return byte masks are completed, then the node has type C (Complete). And if there are both completed and partial byte masks, then the node should

have the type PC (Partial Complete).

Computing the candidate rules
This section defines three functions used to generate the value for the P, S and T field in a certain node, which are the candidate, subCandidate and superCandidate function.

- **candidate(Set RS, Element e1)**: This function requires two-parameter, the RS denotes a set of rules. And e1 denotes an element within a rule, in the form ((byte, mask), (dim, ord)). This function will return all the rules that possess this certain element.

- **subCandidate(Set RS, Element e1)**: The parameter here has the same meaning as the ones in the candidate function. The function will find all the rules which possess an element e2, and the IP range denoted by e2 includes the range denoted in the passed element e1.

- **subCandidate(Set RS, Element e1)**: The parameter here has the same meaning as the ones in the candidate function. The function will find all the rules which possess an element e2, and the IP range denoted by this e2 is included by the range denoted in the passed element e1.

4.3 Construct FAT based on the root node

This section discusses a function named **Develop**. Based on a given node, the Develop function can generate all its sub-nodes recursively until finally reach a leaf node that should not have any offspring. In such a case, if the passed in node is the root node, the whole FAT can be generated. The Develop function is described in Figure 1.

The main process of the function can be divided into three phases. First, the function generates a new node structure and calculates its position. Then based on the calculated position, the function will use the **proj** function to calculate all the byte masks in the rule set that share this position. The returned byte masks will be used to decide the type of the node in the third phase. In the last phase, the constructed node will be completed based on the previously calculated value and call the **Develop** function recursively to construct the following nodes.

```
Develop Function:
Paramater:  Node=(-, -, -, -, P, S, T, bptr, bval)
Output: FAT(Firewall Anomaly Tree)
begin:
    id=Node.id; pos=Node.pos; type=Node.type; P=Node.P;
    edge[]=Node.edge[]; S=Node.S; T=Node.T; bptr=Node.bptr
    bval=Node.bval
    id = getNewId()
    if (bptr == null) then pos=minpos(P) else pos=next(P, bptr.pos)

    if (|P∪S∪T| == 1) then pos=FINAL
    if(pos≠FINAL) then
        V=Proj(P, pos)
        if ( V is complete) then
            type = C
        else
            if (V is partially complete ) then
                type=PC; V=V-{masked bytes}
            else type=P
        Let V = {val-1,...,val-n}
        for 1<=  i  <= n do
            NewNode-i = getNewNode(); edge[hash(val-i)]=NewNode-i
            NewNode-i.bptr = Node; NewNode-i.bval = val-i
            Candidate (NewNode-i); Develop (NewNode-i)
        if (type == PC) then
            NewNode-(n+1) = newnode(); Node.edge[256] = NewNode-(n+1)
            NewNode-(n+1).bptr = Node; NoewNode-(n+1).bval = ϵ
            NewNode-(n+1).P = P\ (∪ NewNode-i.P, 1<=i<=n)
            NewNode-(n+1).S = S; NewNode-(n+1).T = T
            Develop(NewNode-(n+1))
    else  type = F
```

Figure 1. The develop function

4.4 Firewall anomalies detection

After constructing the FAT, we can detect the firewall anomalies by inspecting the leaf nodes within the FAT. For redundant rules, if there are multiple primary candidate rules in a leaf node, then they can be detected. The set of secondary candidate rules contains filtering rules whose domains include the domain of the primary candidate rules. The order of rules is vital for deciding the type of anomaly. If there is a secondary candidate rule whose identifier is lower than the identifier of the primary rule, it indicates a shadowing or down redundancy depending on the actions of the rule. However, if a secondary candidate rule comes after the primary rule, we encounter a generalization or an up redundancy anomaly. Finally, if the domain of the tertiary candidate rule overlaps with that of the primary rule, we detect a correlation or a partial redundancy depending on the rule's actions in that case. A detailed description of the anomalies detection process is presented in Figure 2.

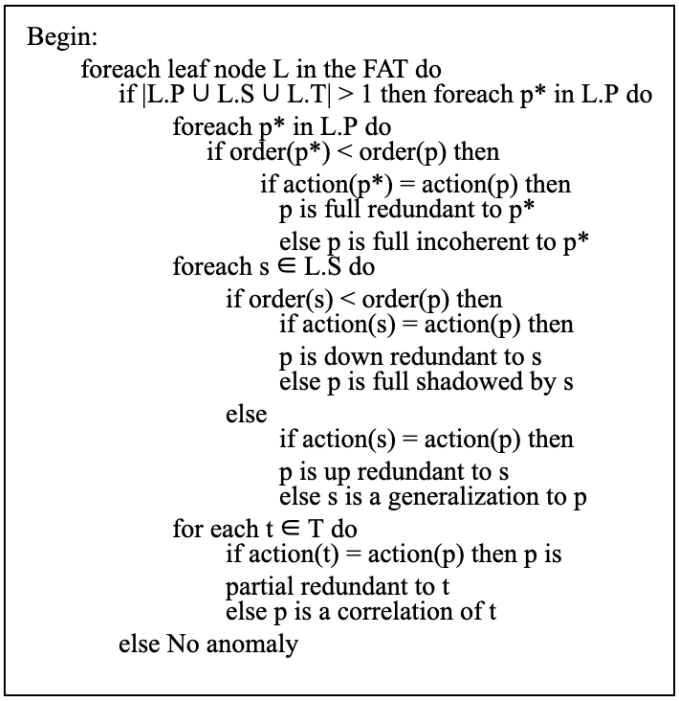


Figure 2. The detection function

5 Conclusion

After trying out the above algorithms using the Java language with some wrongly configured firewall rules, the FAT structure can detect the firewall anomalies defined in Section two. However, the algorithm still has some demerits in practice. The algorithm can only detect the anomalies in a fixed set of firewall rules. After the construction of FAT, there is no way to add a new rule or delete an old rule from it. Therefore, once the firewall rule table changes, the FAT tree also needs to be rebuilt, which will be very time consuming and inconvenient in practice since real-life firewall configuration tables always contain lots of filtering rules. Thus, further research could try to revise the FAT algorithm, making it can be updated dynamically.

References

[1] Ehab S Al-Shaer and Hazem H Hamed. Firewall policy advisor for anomaly discovery and rule editing. In *International Symposium on Integrated Network Management*, pages 17–30. Springer, 2003.

[2] Ehab S Al-Shaer and Hazem H Hamed. Discovery of policy anomalies in distributed firewalls. In *Ieee Infocom 2004*, volume 4, pages 2605–2616. IEEE, 2004.

[3] Ehab S Al-Shaer and Hazem H Hamed. Modeling and management of

firewall policies. *IEEE Transactions on network and service management*, 1(1):2–10, 2004.

[4] Hongxin Hu, Gail-Joon Ahn, and Ketan Kulkarni. Detecting and resolving firewall policy anomalies. *IEEE Transactions on dependable and secure computing*, 9(3):318–331, 2012.

[5] Cengiz Togay, Ahmet Kasif, Cagatay Catal, and Bedir Tekinerdogan. A firewall policy anomaly detection framework for reliable network security. *IEEE Transactions on Reliability*, 2021.

[6] Avishai Wool. Trends in firewall configuration errors: Measuring the holes in Swiss cheese. *IEEE Internet Computing*, 14(4):58–65, 2010.

[7] Lihua Yuan, Hao Chen, Jianning Mai, Chen-Nee Chuah, Zhendong Su, and Prasant Mohapatra. Fireman: A toolkit for firewall modeling and analysis. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 15–pp. IEEE, 2006.

Vehicular Fog Computing: Vision, Capacity Planning, and Resource Allocation

Sinan Sakaoglu
sinan.sakaoglu@aalto.fi

Tutor: Wencan Mao

Abstract

Vehicular Computing Solutions (VCS) has drawn great interest in the recent years. Applications such as object detection and autonomous driving required vast amount of computing resources. Traditional cloud computing model cannot satisfy the requirements of time-critical and data-intensive tasks. Vehicular Fog Computing (VFC) is a technology that provide multiple viable solutions to this problem. This paper examines capabilities, drawbacks and performance of VFC and analyzes state-of-the-art algorithms behind latest VCS frameworks.

KEYWORDS: vehicular fog computing, edge computing, task allocation, v2x, vcs

1 Introduction

Vehicle-to-Everything (V2X) is an emerging concept where vehicles equipped with networking capabilities are able to exchange information with any device capable of telecommunication, around the vehicle [3]. It can enable vehicles to offload data generated by their internal systems to the cloud for analysis and processing, or receive data relevant to their surroundings. This technology is the foundation of Vehicular Computing Solutions

(VCS), which are the key to solving near future challenges in the automotive industry [7].

Advancements in smart vehicles and artificial intelligence have made a push for promoting automotive applications, such as collaborative autonomous driving, crowd-sensing and cooperative lane change [5]. These applications require real-time transmission and processing of data, which makes them computing-expensive. Most vehicles might not be equipped with the required resources. One solution is to move the computing to cloud servers. However, due to the round trip times between the vehicle and the cloud, traditional cloud computing model cannot meet ultra-low latency demands of applications, including auto/assisted driving and emergency failure management [2, 12].

VCS brings an alternative approach: Vehicular Fog Computing (VFC). VFC moves computational resources required by the vehicle to nearby fog servers [12]. These servers could be other vehicles in traffic that have excess resources or commercial fleets equipped with dedicated computing power [11]. However, it is challenging to find an optimal way to allocate units of computing work, i.e. tasks, to vehicular fog servers in a highly dynamic environment. In traffic, there are multiple moving parts, the vehicle, fog servers around the vehicle, and pedestrians. Therefore, the vehicle must maintain stability and continuity of the network connection while the vehicles with the fog server are also in motion. In addition, concrete buildings, trees and other obstacles around the traffic may act as a signal barrier [4]. As a result, communication can malfunction, be disrupted or interrupted entirely.

In this paper, we review the state-of-the-art methods to address task allocation problem in VFC. This paper is organized as follows. Section II introduces vehicular edge computing, fog computing and task allocation. Section III presents different methods of task allocation. Section IV analyzes and compares these methods. Section V concludes the paper.

2 Vehicular Computing Solutions

With development of 5G and beyond networks, real-time, high bandwidth mobile transmission is at reach. Portable devices can transmit large volumes of data to the cloud and communicate with each other rapidly. These advancements have paved the way for VCS. Vehicles equipped with wireless networking devices can communicate with their surroundings. This

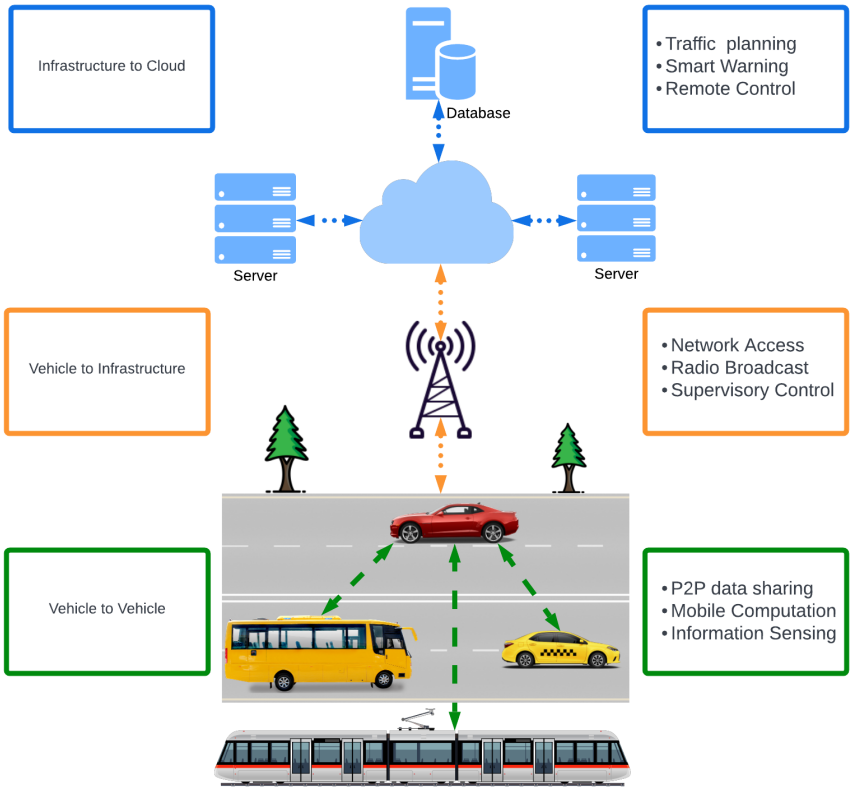


Figure 1. Networking demonstration and different applications of VCS.

communication can be between vehicles (V2V), between vehicles and infrastructures (V2I) or vehicles and networks (V2N) [3]. There are two main types of VCS, Vehicular Edge Computing (VEC) and Vehicular Fog Computing (VFC). While both of them can be deployed standalone, VFC and VEC are used together for best results [3]. An example of this combination can be seen on Figure 1.

Networking is only one piece of the VCS puzzle. Next piece is resolving how to allocate units of computation to nearby servers. We can call one of these units of computation a *task*. The content of a task can vary based on the application, such as encoding of a dash cam video clip, object recognition and pattern recognition. [12, 10]. After a task has been created by a vehicle, depending on its computational capacity, it can either compute it on-board. or select a server to allocate the task.

2.1 Vehicular Edge Computing

The computing paradigm where tasks are offloaded to and executed on statically deployed, near device servers are called Edge Servers [3] . In the context of VEC, these servers can be placed near cell towers, traffic lights, street lamps, road decoration and other road side devices. Hav-

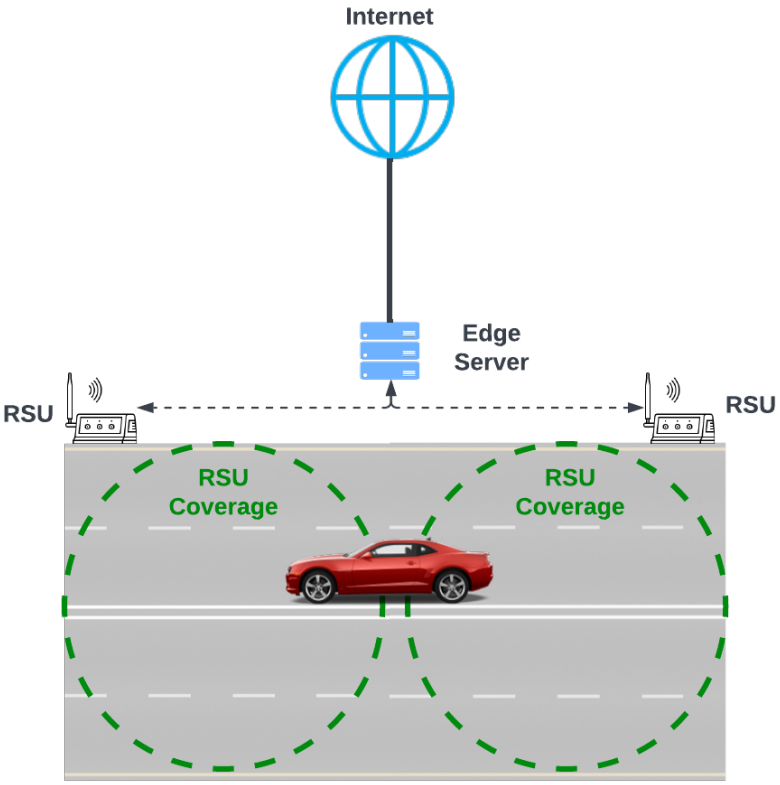


Figure 2. Example of a Vehicular Edge Computing schema.

ing stable connection to the internet and constant supply of power, these edge servers are ideal for jobs, such as power-heavy processing, cloud backup and communication coordination between nearby fog servers [2]. Due to their stationary nature, they are further away to the clients than fog nodes. Especially in rural roads, it is increasingly difficult to achieve ultra-low latency as the nearest server or network could be hours away. This is one of the important differences between vehicular fog and edge computing. While edge servers are planted to specific coordinates, vehicular fog servers are mobile and dynamic.

Road side units (RSU) are devices that act as a link between vehicles, networks or servers. As demonstrated in Figure 2, RSUs are fixed infrastructure along the road with some amount of distance between them depending on their signal coverage. Networking technology they utilize may differ based on the location. Inner city RSUs can use short range, high-speed wireless protocols while RSUs on highways may prefer long range signal technologies. During travel, vehicles will need to change the RSU they are connected to, as they leave coverage of one and enter another. Depending on the application, this can cause loss of state and progress of the running process. However, next generation networks, such as 5G,

and networking equipment are able cover larger areas of road with higher computational power.

One of the great disadvantages of VEC is that compute capacity must be planned ahead of time. This can require a high amount of capital and time investment before the system can function, depending on the scale. Where and how many edge servers to deploy are crucial questions for system designers to answer. Over provisioning of servers will result in a waste of resources during non-peak times. On the other hand, stationing an insufficient amount of servers will cause instability in the system. This creates an optimization problem where the task is to minimize the amount of servers while ensuring enough supply of computation. A solution can be to rent excess resources to third parties, similar to some approaches based on VFC.

2.2 Vehicular Fog Computing

When it comes to computing resources, modern vehicles are equipped with computational, storage and networking units on board [8]. These units are made for the vehicle's own functions, but depending on the situation they are under-utilized. In case of under utilization, the vehicle is wasting a potential income source [7]. By allowing a percentage of their unused computing power, cars present on the road can be converted to mobile fog nodes at minimal cost. This raises the question if VCS can be reliable by depending on partially renting computing power of these vehicles. One solution from state-of-the-art works is to use parked vehicles [10]. An alternative source for computation would be commercial vehicles, such as buses, taxis, trolleybus and trams [11]. Public transport vehicles frequently have predetermined routes. This helps the system designers create educated plans on where and how much computing resources to deploy.

VFC is a great candidate for applications that require ultra-low latency with high mobility [12]. In contrast to VEC, it does not mandate installation of stationary servers or RSUs, but requires extra compute capacity built into the vehicles. As a result, there is less upfront capital cost to building VFC systems. Demonstrated by Figure 3, fog nodes may have different amount of computation available. Hence, the total amount of computation at any moment is highly dynamic. The advantage is depending on the usage data, amount of fog nodes can be increased. This can be done by either installing servers on commercial vehicles or incentivizing

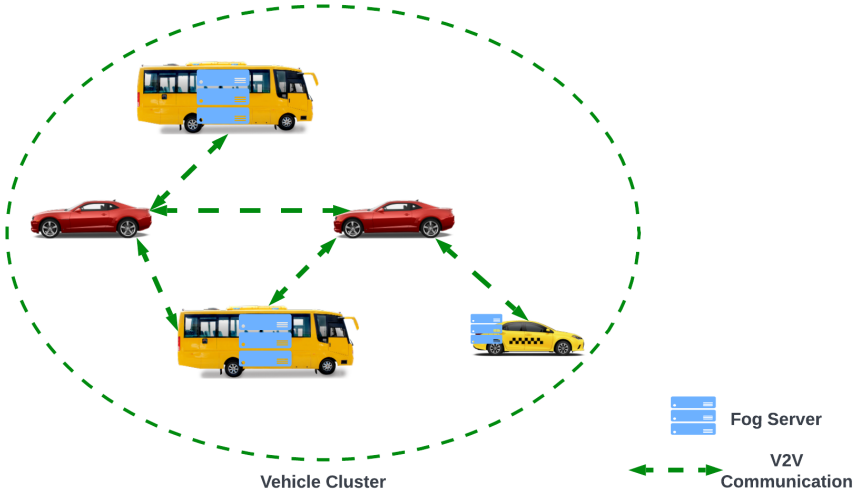


Figure 3. Overview of Fog Computing. Size of the fog server icon represents computational power.

other vehicles on the road to join the system [5].

As seen in Figure 3, VFC can be completely flexible. A *vehicle cluster* (VC) represents group of active actors in the VFC system. Inside the VC, there are users, vehicles demanding computation, and fog nodes, vehicles supplying computation. Depending on the architecture, users can communicate with other users to relay information and discover nearby vehicles, with fog nodes to send tasks and receive results, or with a *base station* for task distribution coordination [2]. Due to the dynamic nature of the traffic, vehicles can frequently enter and leave proximity of the user. For this reason, VFC systems must be designed with fault tolerance and state recovery capabilities.

3 Computational Task Allocation

When there are large numbers of vehicles and users, an important issue is to determine how to distribute user computing tasks across vehicles in order to minimize the overall network latency. In VCS, computational task allocation can be formulated in different ways, including linear programming and Markov decision process (MDP) [2, 11], due to the variety of challenges. First, potential solutions have an important time constraint as real-time applications cannot afford long delays between creation of tasks and their assignment.

In frameworks with information asymmetry assumption, key metrics, such as available resources and sharing target are information that is

private to the fog servers and are not known initially [9]. As a result, the system cannot assign tasks and instead must develop a method, such as task contracts, which allows tasks to be selected by the servers. In addition, given the option, vehicles will request the most powerful server, creating conflict. Thus, the solutions should add fairness to the equation to prevent monopoly, causing users leave the network. Another key challenge is, in an open framework where fog servers are also actors, it is in their interest to receive maximum reward for minimal effort. As a result, solutions should ensure unbiased competition based metrics, such as amount of available tasks and servers.

3.1 Partially Observable Markov Decision Process

Not all solutions depend on a central controller to allocate tasks. Zhu et al. [10] formulates the task allocation problem as a Partially Observable Markov Decision Process (POMDP) where client vehicles create their own task offloading strategy. POMDP uses probability matrices to describe workload state transitions of the Markov chain. Authors assumed that in a given time, workload variation evolved as a Markov chain. They divide time into *time buckets*, as during a period of time the traffic density changes regularly. Time buckets can be further divided into equal sequence of time slots. Assuming that all clients are running the same computational application and produce service demand at equal speeds, *fog node workload* is directly related to the number of adjacent clients.

During the task offloading strategy process, clients decides whether to compute locally or offload based on observation of fog node workload and receives a reward. Depending on the client action and the next workload state of the fog node, a new observation of fog node workload is sent to the client. Goal of the overall strategy is to maximize the cumulative reward in each time bucket.

3.2 Matching-Learning

Contract theory offers a way to solve information asymmetry in a free labor market [8]. It can be applied to VFC in a similar way, fog nodes are modeled as independent contractors and vehicles offloading tasks are the clients. A Base Station (BS) acts as a central agency between the clients and the contractors. Initially, BS will create contracts based on different levels of computational power. By signing a contract, vehicles can become

fog nodes. After initial recruitment of fog nodes, matching theory can be utilized to pair clients with nodes.

First step for the client is to create a server preference list. This list will be based on task offloading delay of servers in descending order. In the matching phase, clients will propose to be paired to the top server in their preference list. Matching will be completed if there is only one proposal to the server. Otherwise, the contracting server will increase its price, which in turn requires competing clients to update their preference list and create new proposal. Stages 1 to 3 will be repeated until only one client asks to be paired with the server.

Under information uncertainty, plain matching based solution suffers in performance. Matching-learning addresses this problem using Multi-armed bandit (MAP), a low-complexity learning methodology. In the first phase of this approach, the client vehicle tries and measures performance of every server available. Next, based on the learned performance metrics, the client can choose to utilize the current optimal server which is a greedy action (G). Conversely, it may continue to experiment with other servers, a non-greedy action (NG). The learning phase will continue until the client is performance satisfaction threshold is reached, finding the most optimal server to minimize the task offloading delay.

3.3 Deep Reinforcement Learning

In a dynamic system where available computation and network state is varied over time, making decisions according to the current state would lead to non-optimal results [5]. Shi et al. formulates this problem as a Markov decision process, and proposes a model-free reinforcement learning algorithm (DRL) based on Soft actor-critic (SAC). The algorithm aims to maximize the mean utility of task offloading at a given time. It accomplishes this while the server selection and cost of task offloading is done at the same time. By including entropy of the policy in the calculation, SAC can take advantage of better strategies. When there are multiple feasible options, it will choose each option with equal probability, allowing for easy adjustments in stochastic vehicular environment.

3.4 Deep Q-Network

Another algorithm used for task allocation optimization that is based deep reinforcement learning is Deep Q-Network (DQN) [11]. DQN uses neural

network to approximate the Q-value function using the previous state. In contrast to Q-tables, it does not calculate each possible state which saves memory and time. Zhu et al. proposes to install a DQN agent at BS, which is the coordinator for a given service zone. Initially, DQN agents randomly assign processing tasks to each fog node and set a collection rate for all data collecting vehicles. DQN agent will update its task allocation strategy as it gradually learns the variations in the workload of fog nodes and the results of its past strategies. Agents do not require any predefined rule-set as they solely develop using experience. Additionally, they can optimize multiple parameters at the same time, such as Quality of Information (QOI) and processing latency.

3.5 Fault-tolerant Particle Swarm Optimization

Particle swarm optimization (PSO) algorithm is known for its efficient global search capability [2]. It is a great choice for solving non-convex and NP-hard problems since it has no requirement to the convexity of the problem. To maximize reliability of task allocation, Hou et al. proposes a fault-tolerant algorithm based on PSO named FPSO-MR, as it is an important property of VCS due to the dynamic nature of the traffic. A particle is a solution of the optimization problem. The swarm seeks the best position which is the optimal solution to the problem. Each iteration of FPSO-MR, particles adjust their direction and speed based on their historical location and position of the optimal particle. Eventually, all of the particles in the swarm will converge to the optimal solution. The time complexity of the algorithm is $O(2^{p+q})$ where p is the mobile-edge nodes and q is fixed edge nodes. It is quite high since the algorithm traverses all possible task allocation to find the optimal strategy. This is not applicable to real world cases. Including the appropriate nodes only to the algorithm increases the performance of FPSO-MR while maintaining level of utility. Additionally, lowering the failure threshold of the framework improves the algorithms efficiency.

3.6 Binary Particle Swarm Optimization

In order to achieve real-time task allocation, Zhu et al. utilized Binary Particle Swarm Optimization (BPSO), a heuristic algorithm [12]. Compared to linear programming based optimization (LBO), it is more efficient and has lower computational complexity. BPSO initializes the pop-

ulation of particles in the swarm with local (LV) and global velocity (GV) vectors. It runs two ongoing comparisons for each particle, local and global best positions. If fitness value of the particle is better than local best position, then it will be set as the new local best position. If the local best is greater than global best, it will become the new global best. The algorithm then uses these values combined with particle's inertia weight to calculate new velocity and position. Each particle represents a task assignment decision and quality loss of results (QLR) selection. BPSO calculates fitness and quality value for each particle. It will try to minimize the QLR and latency, until the maximum iteration number is reached.

4 Analysis

No VCS framework has a silver bullet solution to every problem surrounding the field. Researchers aim to optimize one part of the system while sacrificing from another. Overall, it is a substantial challenge to build an efficient and reliable VCS framework on top of multiple moving pieces. Developing new algorithms to lower the complexity will become easier as the amount of available real world data increases. This section will discuss results of the latest research on VCS based on VFC.

4.1 Latency

Although some papers focus on optimizing only one property of the system, most of them view latency and quality as a single target. Latency itself has many meanings in the context of VCS. Delay between client and fog node, fog node and base station, initialization of the system and task allocation all share the umbrella term latency. To get real-world performance improvements, authors must consider the overall system latency. When comparing different algorithms, it should be kept in mind that results of simulations will vary significantly based on the location and timeframe of the data the authors have used. Furthermore, not all papers are comparable with each other since they have different frameworks and optimization targets.

A POMDP based task offloading strategy were tested by Zhu et al. [10]. When compared with Random and Adaptive task offloading strategies, it reduced average service latency by 65% and 58% respectively. Another approach based on Matching-Learning proved reduce average delay by

17.63% compared to upper confidence bound algorithms [8]. Authors note that selecting the right exploration degree c is critical for the learning performance as a inappropriate value can cause the algorithm to lose its advantage.

Task offloading using SAC, a DRL algorithm, managed to achieve close to 100% completion ratio performance as the vehicle density of the traffic past 15 vehicles/km [5]. Additionally, it performed faster than greedy and random based algorithms in high-priority task allocation. FlexSensing proposed by Zhu et al., a DQN based strategy, reduced average processing latency by up to 51% compared to MUEECA and adaptive task allocation strategies [11]. Both of the deep learning based algorithms have shown great promise to improve the overall system latency. However, they require great amount of data to be functional.

4.2 Quality

Although latency is the fundamental optimization target of a VCS running real-time applications, quality of the computation cannot be disregarded. No matter how performant the task allocation or node communication is, an incorrect result of object detection algorithm may result in serious accidents. Quality is a very broad term, in the context of VCS it could be one of many application outputs, including object detection and image compression rate. Researchers that have realized this include quality parameter in their optimization algorithms.

Folo, a BPSO based dynamic task allocation solution, can be adjusted to the latency and quality requirements of the application [12]. Compared to naive and random node selection, it reduces average latency by up to 27% while lowering quality loss by up to 56%. Another solution based on FPSO-MR, a sub algorithm of PSO, showed that there is a reverse correlation between reliability and latency [2]. As the latency constraint became stronger, more errors occurred, resulting in lower quality and reliability. However, FPSO-MR achieved above 95% reliability with 0.6 latency constraint, surpassing genetic and simulated annealing algorithms.

Deep learning based algorithms have also proven to be successful at optimizing for quality. Priority aware DRL solution proposed by Shi et al. outperforms greedy and random based algorithms in terms of high-priority task completion rate [5]. However, it falls behind in low density traffic with 5-10 vehicles/km since other algorithms focus on maximizing completion ratio all tasks regardless of their priority. FlexSensing on

the other hand, can be tuned to maximize quality of information (QoI), latency or a compromise between both [11]. A case study of FlexSensing computing a real-time object detection application showed up to 34% improvement of QoI, trained using real-world data from city of Helsinki. Compared to MUEECA it can reduce latency by 18% while only losing 2% quality. Configurable algorithms provide great flexibility for VCS applications and are improve applicability for variety of applications.

4.3 Security

The lack of central trust in some systems can allow dishonest actors to disrupt communication, invalidate computation and cause failures in applications [6]. Without proper security mechanisms, both the users and the fog nodes are vulnerable to such attacks. Base stations that are used for system orchestration can also act as a gate, requiring authentication of all actors that join. However, Peer-to-Peer (P2P) only VFC systems must utilize decentralized solutions, which may require effort [1]. When designing such complex communication systems and algorithms, engineers and architects must consider security as a crucial part of the whole solution.

5 Conclusion

This paper outlined the paradigms surrounding Vehicular Computing Solutions, mainly Vehicular Fog Computing. As the number of vehicles with high capacity of computation grows, VFC presents great opportunities and challenges. We have reviewed various solutions of using mobile and parked vehicles as a communications and computing resource. VCS research shows the potential for significant improvements in communication and computing capabilities that can be achieved through VFC. In particular, with the help of task allocation optimization algorithms, lower service latency and higher output quality can be achieved, resulting in greatly reliable VCS frameworks. VFC also significantly improves computing power compared to traditional systems by optimally using the currently underutilized computing resources of individual vehicles. As communication and computing capabilities improve, more advanced developments in vehicle applications and decentralized compute networks will emerge. VFC paradigm is a promising model that can radically transform vehicle networks and various vehicle applications in the future.

References

[1] Zeinab Bakhshi and Ali Balador. An overview on security and privacy challenges and their solutions in fog-based vehicular application. pages 1–7, 2019.

[2] Xiangwang Hou, Zhiyuan Ren, Jingjing Wang, Wenchi Cheng, Yong Ren, Kwang-Cheng Chen, and Hailin Zhang. Reliable computation offloading for edge-computing-enabled software-defined iov. *IEEE internet of things journal*, 7(8):7097–7111, 2020.

[3] Gaurang Naik, Biplav Choudhury, and Jung-Min Park. Ieee 802.11bd amp; 5g nr v2x: Evolution of radio access technologies for v2x communications. *IEEE Access*, 7:70169–70184, 2019.

[4] Gopika Premsankar, Bissan Ghaddar, Mario Di Francesco, and Rudi Verago. Efficient placement of edge computing devices for vehicular applications in smart cities. 2018.

[5] Jinming Shi, Jun Du, Jingjing Wang, Jian Wang, and Jian Yuan. Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning. *IEEE transactions on vehicular technology*, 69(12):16067–16081, 2020.

[6] Vipul Tiwari and Brijesh Kumar Chaurasia. Security issues in fog computing using vehicular cloud. pages 1–4, 2017.

[7] Jingjing Wang, Chunxiao Jiang, Kai Zhang, Tony Q. S Quek, Yong Ren, and Lajos Hanzo. Vehicular sensing networks in a smart city: Principles, technologies and applications. *IEEE wireless communications*, 25(1):122–132, 2018.

[8] Zhenyu Zhou, Haijun Liao, Xiaoyan Wang, Shahid Mumtaz, and Jonathan Rodriguez. When vehicular fog computing meets autonomous driving: Computational resource management and task offloading. *IEEE network*, 34(6):70–76, 2020.

[9] Zhenyu Zhou, Pengju Liu, Junhao Feng, Yan Zhang, Shahid Mumtaz, and Jonathan Rodriguez. Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach. *IEEE transactions on vehicular technology*, 68(4):3113–3125, 2019.

[10] Chao Zhu, Yi-Han Chiang, Abbas Mehrabi, Yu Xiao, Antti Yla-Jaaski, and Yusheng Ji. Chameleon: Latency and resolution aware task offloading for visual-based assisted driving. *IEEE transactions on vehicular technology*, 68(9):9038–9048, 2019.

[11] Chao Zhu, Yi-Han Chiang, Yu Xiao, and Yusheng Ji. Flexsensing: A qoi and latency-aware task allocation scheme for vehicle-based visual crowdsourcing via deep q-network. *IEEE internet of things journal*, 8(9):7625–7637, 2021.

[12] Chao Zhu, Jin Tao, Giancarlo Pastor Figueroa, Yu Xiao, Yusheng Ji, Quan Zhou, Yong Li, and Antti Ylä-Jääski. Folo: Latency and quality optimized task allocation in vehicular fog computing. 2019-06.

Federated learning in industrial applications: opportunities and challenges

Bastien Gouila
bastien.gouila@aalto.fi

Tutor: Alexander Jung

KEYWORDS: federated learning, machine learning, distributed data, industrial applications

1 Introduction

During the past few years, Machine Learning has been an area of focus for many companies to design and commercialize innovative products. With more and more applications emerging, technologies improving, and data being generated, more complex problems can be solved [11]. Federated learning proposes a decentralized computing strategy to train a neural network model from heterogeneous datasets. Datasets can be generated by a single or multiple devices. Unlike centralized training where all the datasets are sent to a server, the devices are analysing their local data and uploading their own model updates to the server, improving the global model [8, 17].

Federated learning has been suggested for different purposes and industrial companies might beneficiate from what it has to offer. It is important to consider that at the time of writing this paper, actual implementations of this technology are still rare. Indeed, many academic articles and researches have been exploring the potential implementations and theoretical benefits. However, referenced federated learning applications which have been released into production are limited.

This paper reviews some properties of federated learning such as, privacy-

preservation, connectivity and customizable applications. It attempts to discuss on how these properties could be integrated to industrial products as well as the difficulties they may encounter. Furthermore, it examines on how suitable federated learning is for an industrial product based on its impact on exiting products.

This paper is divided into the following sections. Section 2 introduces the transition from machine learning to federated learning and its properties. Section 3 presents the opportunities and challenges of federated learning in industrial applications. Section 4, discusses the usefulness of federated learning in a specific industrial product. Finally, Section 5 provides conclusion to this paper.

2 From machine learning to federated learning

Machine learning (ML) has recently seen an increasing amount of use cases. Certain problems were no more solvable with traditional algorithms. Learning algorithms, meant to be trained on data to identify and learn from non-explicit patterns and carrying out tasks hardly done by humans, but computationally feasible, started to emerge [2].

In machine learning, a model is created to learn general features from sample data. Once the model is trained, it attempts to provide predictions or decisions based on what was previously learned. Typically, a machine learning model assumes an evenly and/or randomly distributed input data then attempts to highlight and learn generic pattern on the entire dataset [2].

Due to a growing number of mobile and Internet of Things (IoT) devices, cloud services and wider networks, new potential applications and challenges followed. For instance, heavy data consumption, datacenters scalability problems, communications bottlenecks and data privacy concerns were problems that common ML algorithms were facing and prevented them to provide entirely satisfying results [11]. Hence, to face such issues, and to optimize ML methods for decentralized networks, new angles had to be explored [10].

The next section describes the transition from machine learning to federated learning as well as its relevant properties to this paper.

2.1 Transition to federated learning

Federated learning (FL), is one of these different angles. The goal is to adapt a wider approach. The distribution and the links between some features of the input data help towards building a more global and accurate model, yet also more specific and privacy-preserving at the same time. The idea is to draw a graph of similarities between the datasets using hyper parameters. The similarities regroup the sample data creating clusters of datasets, also called nodes, on which models will be trained, called local models [11, 9].

Combining decentralized networks and machine learning training techniques, a global model learns overall and generic patterns by supervising and updating the local models working on the clusters. In machine learning, the model works on the regularization of the weights to minimize the loss. For the global model in federated learning, the edges of the graph linking the sample data together and the local model updates can be considered as the weights. However, the global model does not ignore what is happening inside a cluster. Indeed, even though the local models are managing their own cluster, the global model learns from the entire dataset and combines the feedback from the local models at the same time [6].

Figure 1 shows an overview of different federated learning network topologies computational plans.

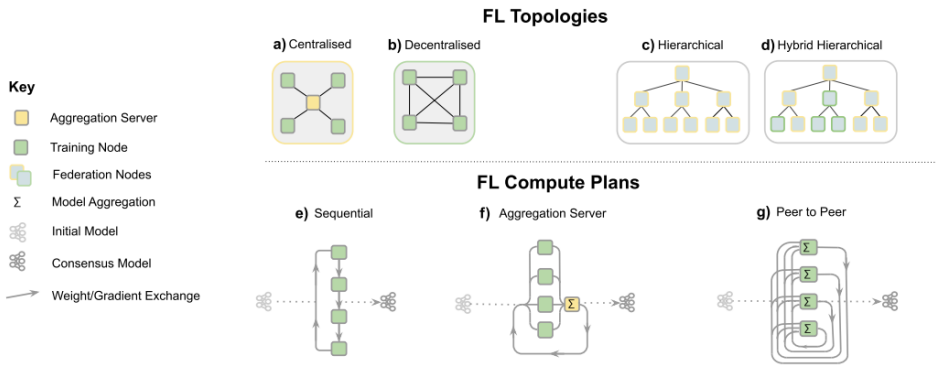


Figure 1. Overview of different FL design choices [15]

As illustrated in Figure 1, federated learning is still similar to traditional machine learning in terms of internal components and computations. However, the organization of the network of nodes and their connections are utilizing well-known network topologies and distributed algorithms methods.

Overall, the training procedure can be separated into the following steps:

initialization, nodes selection, configuration, reporting and finalization [2]. The first step chooses a ML model to be trained and prepares the local nodes for instructions. Then, the nodes selection involves using parts of the local nodes to run a single iteration round, while the others, wait for the next one. Afterwards, in the configuration step, the global server prepares the nodes and starts the training of the local models for the selected nodes, then transits to the reporting phase. The reported results of the local models are handled by the global server which transmits back the required local updates. If more training is required, the cycle loops back to the nodes selection and repeats the previous steps with different local nodes. Otherwise, it proceeds to the final step, where the global server performs the last updates to the global and local models and finishes the training procedure [10].

2.2 Federated learning properties

Federated learning possesses key properties which can be fitted to many applications. The first one is enhanced data privacy compared to traditional machine learning applications. Creating more accurate and custom applications in machine learning sometimes require more information to be extracted from the original data. However, ensuring data anonymization and complying to data privacy regulations can limit such practices. In federated learning, even though the global model is training on all the datasets scattered across each cluster, no raw local data is exchanged externally. During the training and update process, local model results and/or partial amount of their weights are shared. Hence, the learned features are the primary focus making it harder to reverse-engineer or hack into the original input data point [6].

The second property is a more customizable and tailored final output. Although the global model is learning to generalize like a traditional machine learning model, local models can adapt their results for their own purposes. In deep learning neural networks applications, the first layers are focusing on general pattern recognition while the final ones are oriented towards the targeted results. Hence, sharing the first layers across the federated learning network and letting the local models manage their last layers is a way to improve local customization [1]. Several algorithms exist to enhance the personalization of FL models. For instance, the "Masking Trick" proposes a way to improve this customization by allowing more flexibility on the local models while still permitting them

to assist in the training of the global model. This flexibility is achieved by coordinating the training of local models designed as reduced versions of the global model, resulting in a single global inference model [6].

Finally, once a model training has been transitioned to federated learning, the deployment and implementation of the FL-trained model is straightforward. Since federated learning shares the characteristics and challenges from distributed algorithms and large scale communication networks, the training process and requirements of a FL model are different. However, the lifecycles of federated learning models are similar to the ones of classic ML models. Figure 2 gives an example of a lifecycle as well as the interactions of a trained model in a FL system.

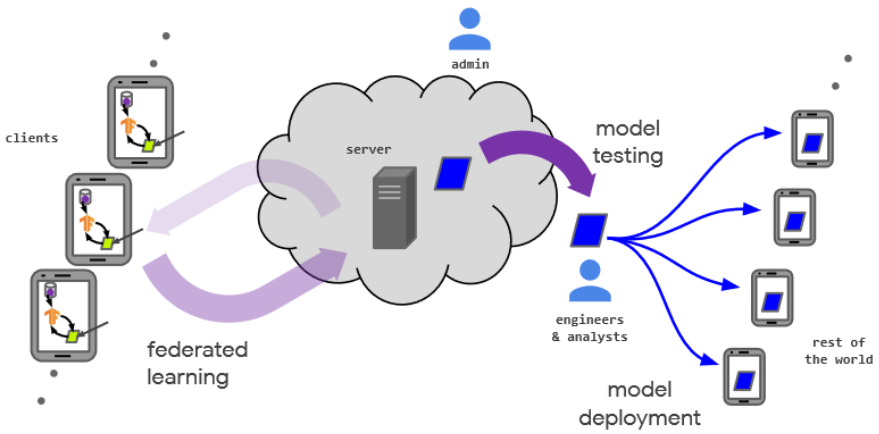


Figure 2. Lifecycle of an FL-trained model and the various actors in a federated learning system [10]

As pictured in Figure 2, a federated learning system regroups the main actors of a machine learning one. A server supervising results of different trainings, engineers and analysts reviewing and testing the models, followed by the deployment of the selected one. This closeness, can offer a smooth transition from traditional ML model deployment that might already exist in a system. Indeed, the local models improvements and changes will be updated from a FL system similarly to a standard machine learning model deployment. Furthermore, it also offers an improvement of scalability due to the wider scope inherent to federated learning [10].

3 Opportunities and challenges in the industry

The next section explains in more details some properties of federated learning , such as, personalized application, data privacy and connectivity as well as the opportunities and challenges they represent in the scope of potential industrial applications.

3.1 Tailored applications

Federated learning has been investigated across several areas. Self-driving vehicles being one of many, where machine learning and computer vision applications are present to analyse and learn from the surroundings of the vehicles [7]. Digital health is also an area of focus with the learning of clinical symptoms and diagnoses [16]. The COVID-19 pandemic, and the challenges it brought, encouraged the research conducted on machine learning for clinical pattern recognition in the health sector [5].

For instance, Figure 3 highlights one of the potential federated systems in the goal of future epidemics warning and response applications.

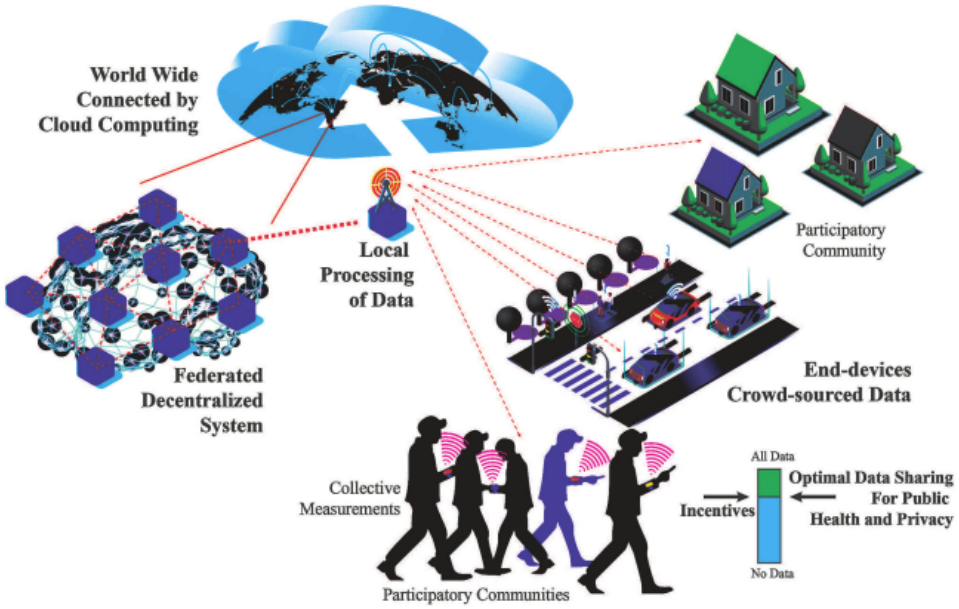


Figure 3. A federated ubiquitous systems for epidemiological warning and response [3]

As illustrated in Figure 3, this federated learning system suggests the usage of crowd-source data. Individual mobile devices being gathered on local processing data nodes and being integrated to a larger decentralized federated system. Such system could be used to provide better estimations and individual diagnoses, recommended procedures to follow, epidemic clusters predictions and more.

Finally, federated learning has been considered for industrial applications, such as smart manufacturing or equipment health. Inspired by the previously mentioned epidemiological warning and response system, a similar prediction and recommendation system could be integrated for the following integrations: process monitoring, supply chain management, quality and maintenance control or energy consumption feedback [4].

3.2 Privacy and security concerns

The ability of federated learning approach to prevent the exchange of local data externally improves data privacy and might convince more companies to utilize their data. Protection of users or equipment private data has become a sensitive legal topic for the past few years and federated learning could align with such concerns [17].

Other problems appear in different matters. For instance, from a security point of view, if a local model is flawed or manipulated, its impact on the global model must be limited [14]. Furthermore, with less access to the client-side annotation, it is more difficult to detect unwanted biases in the training data [6].

Some users or customers might be competitors and the concerns of sharing their data, which will be mixed with others, might lead them to refuse machine learning applications. Combined with the previously mentioned tailored applications possibility, each competitor could gain access to a global federated learning model between them while their local models and formulas would remain their own.

3.3 Connectivity

Federated learning relies on connectivity and interactions between nodes that may be data points from sources scattered around the globe. If customers sometimes refuse to share data, it might be because they simply cannot due to poor network connection. In such case, beneficiating from cloud and distributed systems is not possible [12].

However, if the generated data and the computing resources present on a single industrial site are substantial enough, perhaps a federated learning approach could be considered to tackle some issues on the site level instead.

Even if the remotely located sites or plants have a bad connectivity to the outside world, their premises and equipment are connected to each

other locally to ensure the proper control and supervision of the processes. Hence, a stable internal connectivity on the site level could be sufficient to run a federated learning application.

4 Federated learning applied to existing products

This section develops the impact of federated learning when it is applied to an existing product. This section also attempts to observe how it could be transposed to industrial products.

An implementation of a federated learning model has been designed for Google mobile virtual keyboard, also known as GBoard. The implementation meant to improve the existing next-word prediction model based on n-gram finite state transducer [8]. Two other models, trained from scratch, were designed based on a modified Long Short-Term Memory (LSTM) recurrent neural network, called the Coupled Input and Forget Gate (CIFG). One model was trained on a central server, and the other one, was trained using federated learning. The different models evaluated the top-1 and and top-3 recalls of the next-word prediction on different data samples, such as server-hosted logs, client owned or live user traffic data [8]. The results of the analysis on live user data can be observed in Figure 4.

Model	Top-1 recall [%]	Top-3 recall [%]
N-gram	5.24 ± 0.02	11.05 ± 0.03
Server CIFG	5.76 ± 0.03	13.63 ± 0.04
Federated CIFG	5.82 ± 0.03	13.75 ± 0.03

Figure 4. Prediction impression recall for the server and federated CIFG models compared with the n-gram baseline, evaluated in experiments on live user traffic [8]

The decentralized CIFG model, achieved with federated learning, has shown better results than the identical server-trained and n-gram models for next-word predictions as illustrated in Figure 4. Furthermore, it manages to achieve such results while preserving user data privacy. This successful implementation of federated learning is promising. Indeed, this decentralized learning applied to mobile phones could be considered for other devices, for instance, sensors or smart equipments in industrial plants.

Another integration of federated learning was experimented for a recommender system using federated collaborative filtering. The purpose of this research was to offer a general privacy-preserving federated recommendations system. The system was evaluated using the Hit Ratio (HR@K) metric. HR@K measures the probability of the top-K recommended items to contain left-out test data [13].

On the public MovieLens dataset, the recommender achieved a HR@10 probability of 0.68 on 50,000 users with 5,000 items as well as a HR@10 score higher than 0.50 on the full dataset without impacting the users privacy. In terms of utility, this recommender achieved better HR@K scores than the baseline used in the tests. On the other hand, it was still behind other pre-existing recommenders which could achieve HR@10 scores between 0.70 and 0.80 in the previously mentioned tests. However, even though it was not the most efficient solution, this federated collaborative filtering demonstrated the possibility of achieving a decent performance without sacrificing data privacy [13].

Based on the previously mentioned examples, it can be observed that federated learning could offer a positive impact, with improvements in terms of both performance and data privacy. These properties could provide to industrial companies new ways to process their data and to collaborate into making more accurate applications without risking the privacy of their products and users. Regarding a potential use of such technology in the industry, prediction or recommendation systems similar to the previously mentioned examples could be investigated for process prediction, energy consumption recommendations or supply chain management [4].

5 Conclusion

This article reviewed Federated Learning (FL) and some of its properties, such as, stronger data privacy, customizable applications as well as the potential benefits it could bring to industrial products. Additionally, this paper described the challenges that FL might face. Furthermore, it observed the impact of federated learning on existing products and how it could be integrated to some products in the industry.

Based on the implementations on Google mobile virtual keyboard predictions and the federated collaborative filtering for recommendations system presented in this article, it can be concluded that federated learning offers promising results. Indeed, both the recommender system and the

keyboard next-word prediction models using FL present stronger user-data privacy than other systems without sacrificing correctness of the end results. Furthermore, the FL model designed for GBoard performs better than centralized machine learning applications. Federated learning could reduce the concerns of industrial companies regarding the sharing of industrial data as well as offering new approaches to process users or equipment data towards improving their products.

Finally, as mentioned earlier in this paper, references on successful implementations of federated learning for industrial products are rare. Similar prediction or recommender systems adapted for supply chain management, smart manufacturing or equipment health could be relevant potential implementations of federated learning in the industry.

References

[1] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[2] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019.

[3] Dick Carrillo, Lam Duc Nguyen, Pedro HJ Nardelli, Evangelos Pournaras, Plinio Morita, Demóstenes Z Rodríguez, Merim Dzaferagic, Harun Siljak, Alexander Jung, Laurent Hébert-Dufresne, et al. Containing future epidemics with trustworthy federated systems for ubiquitous warning and response. *arXiv preprint arXiv:2010.13392*, 2020.

[4] Raffaele Cioffi, Marta Travaglioni, Giuseppina Piscitelli, Antonella Petrillo, and Fabio De Felice. Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions. *Sustainability*, 12(2):492, 2020.

[5] Ittai Dayan, Holger R Roth, Aoxiao Zhong, Ahmed Harouni, Amilcare Gentili, Anas Z Abidin, Andrew Liu, Anthony Beardsworth Costa, Bradford J Wood, Chien-Sung Tsai, et al. Federated learning for predicting clinical outcomes in patients with covid-19. *Nature medicine*, 27(10):1735–1743, 2021.

[6] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

[7] Ahmet M Elbir, Burak Soner, and Sinem Coleri. Federated learning in vehicular networks. *arXiv preprint arXiv:2006.01412*, 2020.

[8] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[9] Alexander Jung. Networked exponential families for big data over networks. *IEEE Access*, 8:202897–202909, 2020.

[10] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[11] Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[12] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[13] Lorenzo Minto, Moritz Haller, Benjamin Livshits, and Hamed Haddadi. Stronger privacy for federated collaborative filtering with implicit feedback. In *Fifteenth ACM Conference on Recommender Systems*, pages 342–350, 2021.

[14] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[15] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):1–7, 2020.

[16] Chi-Ren Shyu, Karisma Trinanda Putra, Hsing-Chung Chen, Yuan-Yu Tsai, KSM Hossain, Wei Jiang, Zon-Yin Shae, et al. A systematic review of federated learning in the healthcare area: From the perspective of data properties and applications. *Applied Sciences*, 11(23):11191, 2021.

[17] Abraham Woubie and Tom Bäckström. Federated learning for privacy-preserving speaker recognition. *IEEE Access*, 9:149477–149485, 2021.

Deep Learning for Kinematic Character Animation

Alena Shchevyeva
alena.shchevyeva@aalto.fi

Tutor: Nam Hee Kim

Abstract

The animation industry can benefit significantly from rapidly evolving machine learning solutions, such as neural networks. These methods could allow for faster and cheaper production, thus democratizing the market and allowing more creative freedom. This paper reviews the state-of-the-art neural network techniques in 3D character animation. The review is limited to kinematic animation and covers only motion retrieval and motion synthesis techniques. Motion retrieval covers learned motion matching, whereas motion synthesis is focused on phase and mode adaptive neural network models, such as PFNN, MANN, and local motion phases framework. Finally, the paper summarizes the implications of the methods discussed and possible areas for future research.

KEYWORDS: 3D character animation, neural networks, kinematics

1 Introduction

The rise of computer technologies led to the appearance of 3D graphics in live-action films by the late 1980s and entire 3D feature films by the mid-1990s. By the 2010s, 3D computer graphics became the prevalent animation technique that is still being developed continuously. Although 3D

animation tends to lead to faster production than traditional hand-drawn animation, it is still extremely expensive and time-consuming. Production of a fully animated full-length film can take anywhere between two to four years and require a crew of hundreds of employees [1]. For example, *Tangled*, an animated film by Walt Disney Animation Studios, spent six years in production with a budget estimate comparable to that of the whole *The Lord of the Rings* trilogy [2].

Artificial neural networks offer promising solutions to the issues of 3D animation outlined above. The possible applications of neural networks include character motion synthesis [3], eliminating corrupt motion errors [4], character control over various terrains [5], and animating quadruped animals [6]. Utilizing these techniques has the potential to make animation more affordable and accessible to individuals and smaller teams while decreasing costs and production time for larger studios.

The paper aims to review current state-of-the-art neural network solutions to 3D computer character animation problems. Here, the animation is limited only to kinematic animation where the object's state is described by its pose, and the motion of the object is simulated by subsequently changing its poses. The poses and the rules for their update are described via some set of equations. The dynamic animation, which focuses on how different forces affect the movement, is left out of the scope of this paper.

The rest of the paper is organized as follows. Section 2 briefly describes the history of 3D animation and neural networks. Section 3 introduces basic concepts of character animation and minimal formalism. Section 4 overviews state-of-the-art neural network techniques in character animation. Section 5 discusses the implications of these methods and avenues for future research. Finally, Section 6 summarizes the paper.

2 History of 3D Animation and neural networks

2.1 3D Animation

Although the first digital computer animation was produced already in the 1960s [7], 3D animation in its current form started to evolve only in the mid-1970s with the development of microprocessors and reached industry production by the late 1980s [1]. These advancements allowed for the emergence of the character 3D animation in the same decade. Three

innovative 3D-animated short films were produced at the University of Montreal: *Dream Flight*, the first 3D generated film telling a story with some limited character animation [8]; *Tony de Peltrie* portraying the first 3D animated human character expressing emotions [9]; and *Rendez-vous in Montreal*, a short film simulating Marilyn Monroe meeting Humphrey Bogart in a café in the old town of Montreal [10]. Another key animated short film of the time, Luxo Jr. produced by Pixar, became the first CGI film nominated for an Academy Award. This short film was a breakthrough not only by the technologies used but also by its strong appeal to a wider audience.

In the same decade, Ginsberg and Maxwell at MIT presented the Graphical Marionette project, an early version of motion capture [11]. However, the realistic and complex animation created entirely with motion capture was presented only in 1993 by Acclaim Entertainment when faster sensors and more efficient hardware were developed [11]. The next milestone in the industry was the release of *Toy Story* in 1995 – the first full-length 3D animated feature film produced by Pixar Animation Studios [12].

2.2 Neural networks

Neural networks were first introduced in the mid-1940s. However, convolutional neural networks (CNN) that are currently widely used in image recognition and generation tasks were first introduced only in 1987 by Atlas et al. [13]. In 2004, Oh and Jung [14] significantly improved the runtime of standard neural networks by utilizing GPU instead of CPU. This allowed the creation of deeper networks and the improvement of the results. By 2011, CNNs achieved rapid learning and low error rates on standard image recognition datasets, such as MNIST, CIFAR-10, and NORB [15].

The first impactful application of the CNNs in the field of 3D character animation emerged in 2015-2016 when Holden et al. [3, 4] introduced new solutions based on the convolutional autoencoders to the problems of corrupt motion data or character motion synthesis. Finally, novel neural network architectures, such as phase-functioned neural network (PFNN) and mode-adaptive neural network (MANN), were developed in the past 5 years to address issues of character control over difficult terrain and quadruped motion control [5, 6]. These advancements, including PFNN, MANN, and recently emerging solutions based on them, will be further discussed in Section 4.

3 Concepts of character animation

Character animation focuses on a movement of a particular character in a two- or three-dimensional space. The character structure consists of bones and joints that form chains of rigid body structures modelling the skeletal structure of real-life animals. Furthermore, it includes a root joint that is meant to control the character's global body position and orientation. An example of such animation skeletons is shown in Figure 1. Every joint can be described by its position, rotation, and in some applications by its angular velocity at any given time. Traditionally, character animation is performed by manually adjusting joints for each frame to create smooth movement. However, this is a tedious, time-consuming process that does not scale well. A controller – an instance that displays frame sequences based on the context – simplifies this process by defining what the character's next position should be (i.e., position and rotation for each joint), based on the current state, target state, trajectory, or other user input.

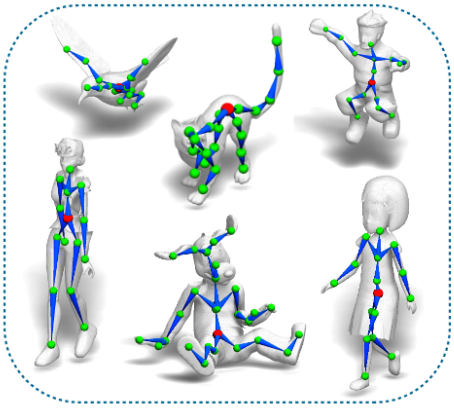


Figure 1. Animation skeletons with bones marked in blue, joints in green, and root joint in red [16].

At a rather basic level, animation is a sequence of frames, where each frame contains information about the current state of the world, based on which one can visualize it. More formally, animation can be represented as an input vector $\mathbf{x}$ for frame ϕ, which includes, for example, joint positions, rotations, and velocities. To produce believable motion sequences based on the available data (for example, from motion capture), we introduce some function f, which produces an output vector $\mathbf{y}$ of joint positions,

rotations, and velocities for frame $\phi + 1$ when applied on a vector x:

$$\mathbf{x} = \begin{bmatrix} j_{0_\phi} \\ j_{1_\phi} \\ \vdots \\ j_{n_\phi} \end{bmatrix}, \mathbf{y} = f(\mathbf{x}) = \begin{bmatrix} j_{0_{\phi+1}} \\ j_{1_{\phi+1}} \\ \vdots \\ j_{n_{\phi+1}} \end{bmatrix} \tag{1}$$

, where $j_{0_\phi} ... j_{n_\phi}$ represent joint positions, rotations and velocities at the frame (time) ϕ. The output y is a prediction of the next state needed for the visualization based on the current character state. Predicting complex motions, such as playing basketball, requires complex function f or a combination of several functions constituting a model, as well as more information in the input x. Such additional information may include, for example, movements of other objects that affect the character (e.g., ball movement for animating a basketball player) or phases for cyclical motion, such as walking [17].

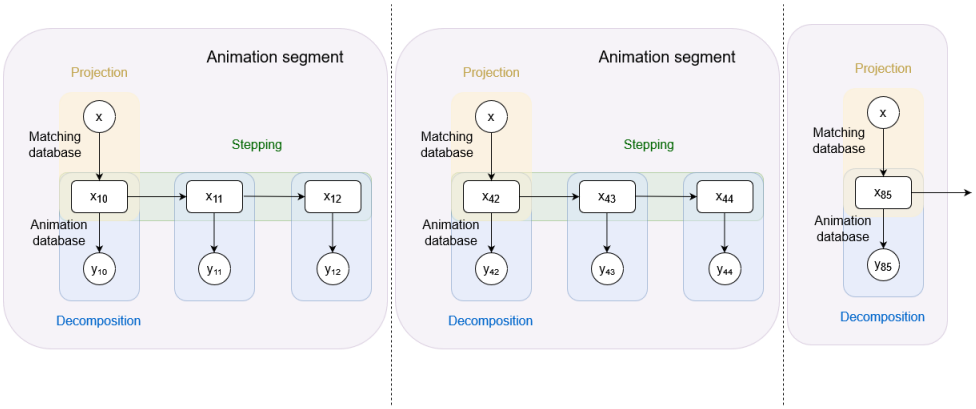
4 Evolution of neural networks based techniques for character animation

This paper focuses on two clusters of neural network techniques currently used for character animation: motion retrieval techniques and motion synthesis techniques. Both approaches require sufficiently large motion capture datasets, but they utilize them differently. Motion retrieval techniques, such as motion matching, are focused on selecting the next animation fragment from a large database of animations based on how well it fits the current context [18]. In contrast, motion synthesis techniques advance animation by generating new poses based on the current ones. The motion synthesis techniques utilize learned functions, i.e., functions trained on some dataset; however, they do not consult any database after the training is completed. These different approaches are analyzed further in Sections 4.1 and 4.2.

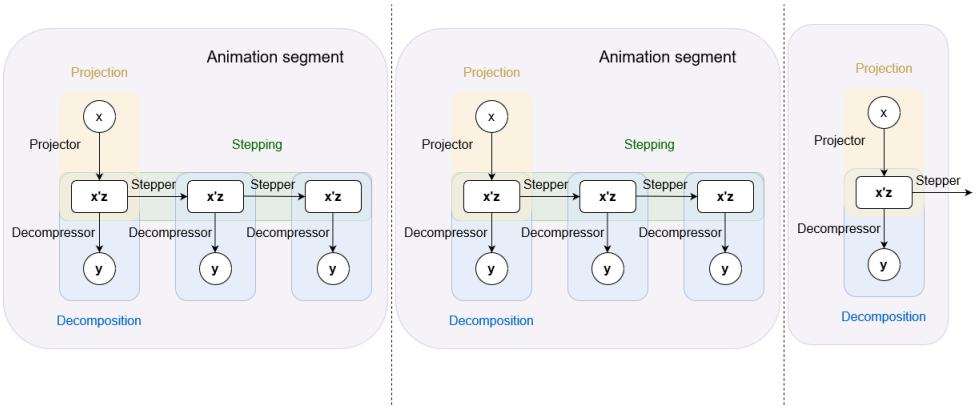
4.1 Motion matching techniques

Once the task is defined, the general process of motion matching can be described as first identifying a few crucial features for the current task and then based on them formulating a query vector to retrieve the next animation from the database. Finally, the transition from the source to the target frame is blended to achieve smoother results [18]. Here, the

fully-parameterized vector x, as in Section 3, often contains excessive information, such as joints irrelevant to the current movement. These features might hinder the formulation of the query vector and unreasonably limit the available results. Therefore, a smaller subset x' of only relevant crucial features are selected from x and each complete pose vector is matched to this resulting feature vector. All available poses (vector y from Section 3) and corresponding feature vectors are concatenated into two large matrices that are called Animation and Matching databases respectively [18].



(a) Basic motion matching: Matching and Animation databases are used for querying.



(b) Learned motion matching: Decompressor, Projector and Stepper networks are utilized to replace Matching and Animation databases for a memory-efficient solution.

Figure 2. Comparison of basic and learned motion matching techniques [18].

The advantages of motion matching include predictability, low preprocessing time and visual quality [18]. However, continuously storing Animation and Matching databases in memory results in this method being memory-inefficient – memory usage scales linearly with the size of the databases. To address this issue, Holden et al. [18] suggest a learned motion matching algorithm, which applies several neural networks to re-

move the dependency of the basic motion matching on Animation and Matching databases (see the comparison of the methods in Figure 2). The first two networks, the Projector and the Stepper, work together to eliminate the Matching database. The Projector replaces the traditional nearest neighbour search and finds the next best-matching feature vector via a neural network that takes as input a query vector $\mathbf{x}$ and outputs the feature vector $\mathbf{x}'$ and additional latent vector $\mathbf{z}$ which adds state context missing from $\mathbf{x}'$ otherwise. The Stepper then recurrently advances the animation by utilizing the found representations at frame i, i.e., $\mathbf{x}'_i$ and $\mathbf{z}_i$, to find the representations for frame $i+1$, i.e., $\mathbf{x}'_{i+1}$ and $\mathbf{z}_{i+1}$. The final network, the Decompressor, eliminates the Animation database by predicting output pose vector $\mathbf{y}$ from the learned representations, i.e., input vectors $\mathbf{x}'$ and $\mathbf{z}$ representing the current character pose. Furthermore, the Decompressor can be used separately from the Projector and the Stepper to achieve different trade-offs between memory usage and evaluation time. With all three networks deployed, the resulting algorithm achieves a significant improvement in memory usage for a variety of tasks compared to the basic motion matching.

4.2 Motion synthesis techniques

There is a vast variety of methods utilizing neural networks for motion synthesis. This paper covers only a small subset of them, which mostly focus on game-controlled animation with forward kinematics models trained on motion capture datasets.

One of the notable solutions is phase-functioned neural networks (PFNN) [5], which significantly contributed to the further development of character control techniques. It is aimed at solving locomotion tasks, such as stepping, jumping, or climbing, over difficult terrain. These tasks are cyclical, hence utilizing the global motion phase enhances the output quality. The global motion phase serves for generating the weights of a regression network at each frame, which prevents mixing data from different phases and allows smoother outcome [5]. Once the weights are generated, the network performs regression from the input feature vector $\mathbf{x}$ to the character pose $\mathbf{y}$ of the next frame. Aside from the basic motion capture preprocessing, data preprocessing for PFNN includes additionally phase extraction and terrain fitting. Here, terrain fitting refers to fitting a database of heightmaps to separately captured motion data, since motion capture does not allow for easy simultaneous capturing of both

terrain and motion in a motion capture studio. Although the PFNN as introduced by Holden et al. is limited to only generating cyclic behaviour, it proposes a fast and lightweight system, which is capable of producing realistic walking or running sequences for humanoid characters.

While PFNN mostly addresses human locomotion, mode-adaptive neural networks, MANN, aim to solve the challenges of the quadruped motion control [6]. Additionally, MANN address the challenges of preprocessing by eliminating the need for manual phase labelling, reducing preprocessing time, and avoiding human error in gait mislabelling. To achieve this, the system learns consistent features across the wide range of unstructured quadruped motion with different periodic gait types (e.g., gallop or trot) and develops gait-specific controllers that can be then blended with a mixture of experts. MANN implements the mixture of experts by taking inputs in a similar format to PFNN (but excluding phase labels) and utilizing two neural networks: a gating network and a motion prediction network. The gating network receives a motion feature $x' \in x$ and dynamically computes blending coefficients, which are then utilized as weights by the motion prediction network to produce output vector y based on the input x [6]. The overall architecture can be found in Figure 3. In comparison to PFNN and vanilla feed-forward networks, MANN achieves smoother and more natural results for quadruped locomotion; however, its implementation by Zhang et al. [6] is still limited to flat terrains and does not support more complex motions, such as jumping.

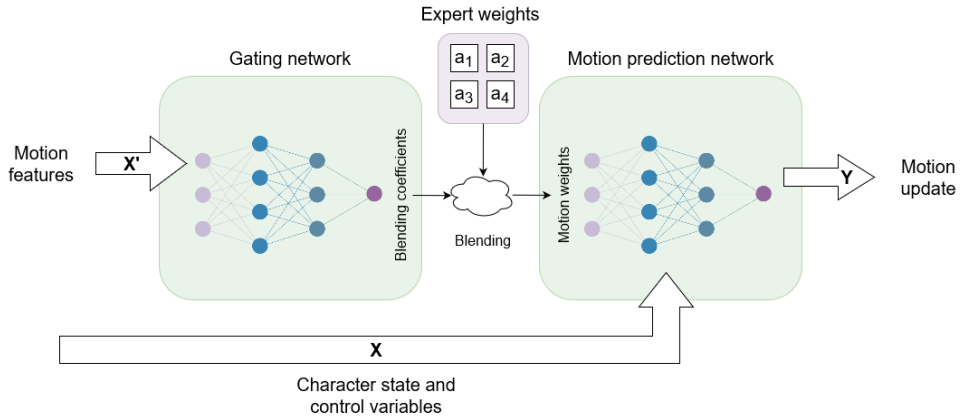


Figure 3. MANN architecture [6].

In 2020, Stark et al. [17] introduced a local motion phases framework that combines the ideas introduced in PFNN and MANN for learning multi-contact character movements. It uses an architecture similar to MANN with gating and motion prediction networks. Its high-level archi-

tecture overview can be found in Figure 4. The input to the gating network consists of local motion phases. The idea of local motion phases extends the phase information introduced earlier by Holden et al. in PFNN. Instead of using one global phase for the character, Starke et al. [17] suggested multiple independent local phases for each bone to improve performance on asynchronous movements, where different body parts are moving at different and consistently changing phase shifts. The resulting framework produces realistic outputs for fast-paced, asynchronous character interaction, such as a basketball game with two players. Furthermore, it is faster and more responsive than MANN for the quadruped motion, especially when switching between different locomotion modes, such as from sitting to walking [17].

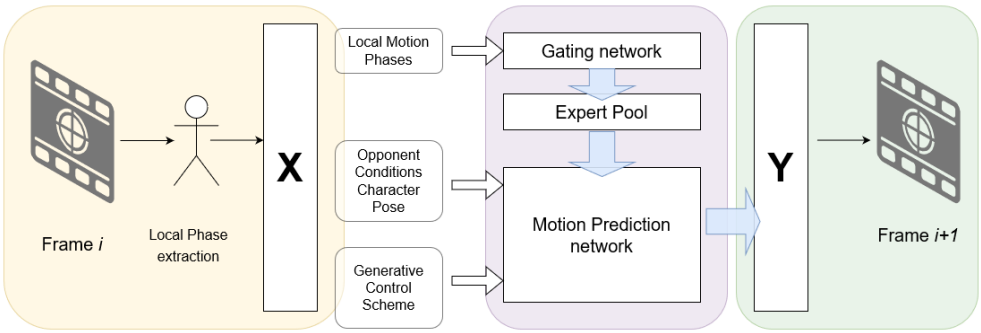


Figure 4. Architecture of local motion phases framework [17].

5 Discussion

Currently, the majority of the research is focused on bipedal and quadruped motion, such as the motion of human or four-legged animal characters. Although the methods reviewed in this paper achieve great results, they are still limited to a few character structures they can operate on. Future research could investigate the possibilities of transferring these methods to a wider range of characters – for example, birds or other non-humanoid characters, such as aliens. This might bring additional challenges, among which are complex rigging (i.e., defining the bones and joints of a character) and lack of motion capture data. Moreover, there might be new kinds of motion that are completely different from the bipedal or quadruped motion.

These arising challenges can be at least partially solved by neural network solutions. For example, RigNet introduced by Xu et al. [16] can already generate a skeleton matching animators' expectations based on a

provided 3D model, including cases of characters with an arbitrary number of limbs or non-humanoid characters. Motion retargetting [19] is another approach to adapting existing methods to novel character models.

Overall, the animation is only a small part of a 3D production pipeline that consists of a multitude of steps, including, but not limited to, initial idea, story, design, modelling, rendering, and compositing. Future research could target the challenges of integrating neural network models into the production pipeline and the implications it would have for industry professionals, especially animators. Automating manual labour-intensive tasks might lead to the role of animators shifting towards post-processing (for example, validating and correcting generated results) or data engineering and MLOps positions since neural networks in the production environment would require continuous maintenance and very likely continual learning as new data arrives. This process might have certain ethical implications, which also presents an opportunity for further research.

6 Conclusion

This paper has reviewed two different approaches in current animation techniques utilizing neural networks: motion retrieval and motion synthesis. It covered several state-of-the-art techniques associated with each approach and discussed their aims, advantages, and disadvantages. These methods have the potential to make 3D character animation faster and more affordable and therefore more accessible to small independent studios or individual creators. This could allow for more creative freedom with less technical limitations and lead to more diversity among 3D projects targeting a wide audience. Therefore, many parties, including studios, artists, and audience, could benefit from the further development of neural network animation techniques.

References

[1] A. Beane, *3D animation essentials*. John Wiley & Sons, 2012.

[2] Box office mojo. Accessed on: 03.02.2022. [Online]. Available: https://www.boxofficemojo.com

[3] D. Holden, J. Saito, and T. Komura, "A deep learning framework for character motion synthesis and editing," *ACM Trans. Graph.*, vol. 35, no. 4,

jul 2016. [Online]. Available: https://doi.org/10.1145/2897824.2925975

[4] D. Holden, J. Saito, T. Komura, and T. Joyce, "Learning motion manifolds with convolutional autoencoders," in *SIGGRAPH Asia 2015 Technical Briefs*, ser. SA '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2820903.2820918

[5] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," *ACM Trans. Graph.*, vol. 36, no. 4, jul 2017. [Online]. Available: https://doi.org/10.1145/3072959.3073663

[6] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Trans. Graph.*, vol. 37, no. 4, jul 2018. [Online]. Available: https://doi.org/10.1145/3197517.3201366

[7] P. Lundin, "Databehandling vid väg- och vattenbyggnadsstyrelsen/vägverket 1957–1980: Transkript av ett vittnesseminarium vid tekniska museet i stockholm den 22 maj 2006," 2007.

[8] N. Magnenat-Thalmann and D. Thalmann, "The use of high-level 3-d graphical types in the mira animation system," *IEEE Computer Graphics and Applications*, vol. 3, no. 9, pp. 9–16, 1983.

[9] P. Bergeron and P. Lachapelle, "Controlling facial expressions and body movements in the computer-generated animated short: Tony de peltrie," *Computer Graphics (SIGGRAPH'85), Course Notes: Techniques for Animating Characters*, 1985.

[10] N. Magnenat-Thalmann and D. Thalmann, "The direction of synthetic actors in the film rendez-vous a montreal," *IEEE Computer Graphics and Applications*, vol. 7, pp. 9–19, 1987.

[11] D. J. Sturman, "A brief history of motion capture for computer character animation," *SIGGRAPH 94, Character Motion Systems, Course notes*, vol. 1, 1994.

[12] D. Price, *The Pixar Touch: The Making of a Company*. Alfred A. Knopf, 2008. [Online]. Available: https://books.google.fi/books?id=ExfiwAEACAAJ

[13] L. Atlas, T. Homma, and R. Marks, "An artificial neural network for spatio-temporal bipolar patterns: Application to phoneme classification," in *Neural Information Processing Systems*, D. Anderson, Ed. American Institute of Physics, 1988.

[14] K.-S. Oh and K. Jung, "Gpu implementation of neural networks," *Pattern Recognition*, vol. 37, no. 6, pp. 1311–1314, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320304000524

[15] D. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification." 07 2011, pp. 1237–1242.

[16] Z. Xu, Y. Zhou, E. Kalogerakis, C. Landreth, and K. Singh, "Rignet: Neural rigging for articulated characters," *ACM Trans. Graph.*, vol. 39, no. 4, jul 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392379

[17] S. Starke, Y. Zhao, T. Komura, and K. Zaman, "Local motion phases for learning multi-contact character movements," *ACM Trans. Graph.*, vol. 39, no. 4, jul 2020. [Online]. Available: https://doi-org.libproxy.aalto.fi/10.1145/3386569.3392450

[18] D. Holden, O. Kanoun, M. Perepichka, and T. Popa, "Learned motion matching," *ACM Trans. Graph.*, vol. 39, no. 4, jul 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392440

[19] M. Gleicher, "Motion editing with spacetime constraints," in *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, ser. I3D '97. New York, NY, USA: Association for Computing Machinery, 1997, p. 139–ff. [Online]. Available: https://doi.org/10.1145/253284.253321

The Multiple Layers of Anonymity

Sara Kanerva
sara.kanerva@aalto.fi

Tutor: Chris Brzuska

Abstract

Anonymity is the cornerstone of truly honest communication. Anonymity can shield communicating parties from any major consequences that expressing their opinions or experiences might accompany. Hence, removing inhibitors of discussion and speech. Anonymity can be condensed by the notion of not allowing an attacker or observer on a network to link participants to certain actions. This paper discusses the notion of anonymity, while examining diverse models for reinforcing and enhancing the notion itself, predominantly within anonymous communication systems.

KEYWORDS: anonymity, authentication, cryptography, anonymous communication, anonymity set, anonymous communication systems

1 Introduction

One can no longer question whether electronic communication technologies are here to stay. Due to the progression of technology, the amount of communication conducted through technological devices has increased significantly. Thus, the demand for identifying mechanisms accompanied with the technologies and cryptographic mechanisms enabling them has grown [1]. Through the emergence of web services such as social media,

online shopping and banking, the quantity of personal information circulating online has skyrocketed. With these online tools it has become child's play to gain knowledge about a specific person. Furthermore, numerous bits and pieces of personal information harvested by online services are being put up for sale, which is only one of the plethora of reasons for which a party could wish to remain anonymous during communication [2].

True anonymity can be seen to include the notion of unlinkability. Indeed, within the context of cryptographic key exchange, anonymity and unlinkability are nearly equivalent [3]. Still, anonymity as a concept can be viewed both from the points of hiding the identity of the entity who is preforming an action as well as hiding the action itself [4]. Users can waver between what identity information they are willing to succumb while using various online services. Thus, even how the users themselves experience privacy is prone to fluctuate. In some cases concealing who is responsible for the message might be more crucial than the sent message itself [5].

The technical and the social infrastructure behind anonymity is tangled and complex. The large scale of implementation environments does not make the task of fulfilling the architectural needs of anonymity any easier. Various ways of improving and measuring the strength of the implemented anonymity processes have been suggested, but few seem to meet the requirements of all implementation objectives.

2 Anonymity

Rooted in the Greek language, the word anonymity has been given the meaning of "namelessness" or "without a name" [6]. Pfitzmann and Köhntopp [7] define anonymity as the state of being unidentifiable within a set of subjects, also referred to as the anonymity set. Anonymity can be considered as one of the cornerstones for democracy, by empowering the principle of freedom of speech [1]. Actions which in the absence of anonymity can result in the torment of the individual, construct significant use cases for anonymity [2]. When it comes to communication in the cyberspace, the identities of message sender's can be anonymised trough various processes. This allows for activists and so-called whistle blowers to share critical knowledge on companies, organizations and individuals taking part in criminal or unjust activities, in the safety of their anonymity. Alter-

natively anonymity opens a possibility for users to spread messages with malicious intentions [1].

The concept of anonymity can be further divided into subcategories. Two of which are true anonymity and pseudo-anonymity [7]. True anonymity is completely untraceable. When true anonymity is applied the identity of the sender is unable to be detected. True anonymity can be seen to possess grave potential for exploitation in the realm of illegal activities and cyber crime [1]. In pseudo-anonymity the communication between the senders is indeed traceable. The sender's identity is anonymised, but possible to be uncovered, although only through extensive efforts [7]. Thus, the sender commits to taking accountability of their actions, although initially protected by anonymity. Pseudo-anonymity can facilitate governments in misusing their access to the identities of pseudo-anonymous users [8].

3 Anonymous Communication Systems

Anonymity and privacy are central concepts of the field of communication. Users seeking higher degrees of anonymity are moving towards systems which can provide them with the promise of anonymous communication [9]. Anonymous systems for communication first surfaced in the mid-1980s, providing users with the first ways of anonymous electronic communication through relay servers [5]. The central ambition of anonymous systems is to protect the identities of the communicating entities. Further motivators include personal privacy and prevention of data mining and tracking [10].

Anonymous communication systems (ACSs) can be categorised based on system architecture and the latency their users are succumbed to during communication [10]. Architecturally the systems are divided into client-server communication systems and peer-to-peer anonymous networks. Latency wise, into low and high latency systems. Using a peer-to-peer architecture allows for no distinction to be made between a user and a server, hindering the possibility of an attacker tracking the network traffic. The architecture in client-server communication systems provides anonymity for the system users only through a few nodes, thus making it easier for the attacker to extract data.

Anonymous communication systems brought into being the technique of routing messages through multiple relays, to offer their users the anonymity they are seeking for [11]. Tor, and the onion routing the system im-

plements is one of the most widespread anonymous communication systems [12]. Tor has a large user-base accompanied with a limited number of servers, which can result in performance issues when it comes to anonymity [10]. Hence, some systems such as Riffle and Vuvuzela provide additional security against adversaries by using communication rounds [11]. Communication rounds regulate the time frames clients can communicate in. Messages are sent every round either by choice of the user or through system fabrication. A real message sent by the user is indistinguishable from system fabricated messages.

The privacy goals of anonymous communication systems may vary to a great extent. The systems are constructed with a range of use cases in mind. Advantages of the implemented anonymity technologies often come with drawbacks such as latency and lack of malicious attack detection [10]. However, new improved techniques and approaches are being researched. As the design of anonymous systems advances, so does the overall perception of the concept of anonymity itself.

4 Attacks on anonymity

As with most protective measures, weaknesses found in anonymous systems are targeted by various attacks. Anonymous systems have been targeted with attacks such as encrypted data cracking, man-in-the-middle attacks, data replays and traffic analysis to list a few [5]. However passive long term attacks such as intersection attacks form one of the strongest threats posed against anonymous communication systems, such as Tor [11].

Intersection attacks are deterministic attacks where an adversary of a global and passive nature is able to observe outgoing and incoming message traffic completely within a ACS [11]. The adversary can then link the sender and targeted receiver as illustrated in Figure 1. The attack is performed by the adversary recording which recipients are online at the time a message was sent, hence by repetition being able to weed out a single client out of the possible recipients.

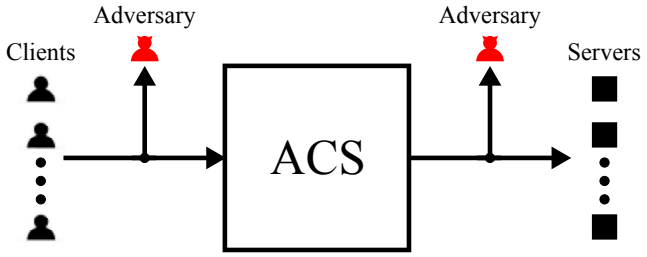


Figure 1. Threat model of a global passive adversary monitoring all messages incoming and outgoing from the ACS, modified from [11]

In addition to the attacks of deterministic nature, anonymous communication systems possess vulnerabilities to statistical disclosure attacks [11]. Statistical disclosure attacks can be conducted when an adversary is able to estimate the likelihood of a single recipient from a group of possible recipients, being the targeted receiver of the message in question [11]. The attack is conducted by churning of the ACS's user base. Resistance against these attacks could be improved by setting random delays to sending messages and by utilizing strong anonymity sets.

5 Anonymity loves company

The more users a service such as an ACS has, the more there are possible entities to link with the messages sent across the system. To increase the anonymity resistance of these services, their user base can be further grouped into anonymity sets. An anonymity set includes a group of users, so called "honest" users. All of these "honest" users might be responsible for sending a message, since they are in possession of the same attributes from the perspective of the adversary [13].

Within a system consisting of an arbitrary number of users, the system can reach the highest level of anonymity the moment an adversary recognizes all entities within a anonymity set to have equivalent probabilities as for being the senders of a message [13]. In practise latency and user behaviour are constraints on the anonymity set. This enables an adversary to succeed in the creation of persistent behavioral user profiles [14]. Basing security mechanisms on the assumption of constantly participating clients might prove detrimental for an ACS, since human user behaviour patterns can be unpredictable.

An anonymity set consisting of a lone user is impossible to maintain, thus the minimum size of an anonymity set is two entities [13]. Insert-

ing fake identities to grow the size of an anonymity set can be done, but issues arise when the users within one anonymity set stop using the service or change their posting behaviours. This can lead to the shrinkage of the anonymity set. Hence, the set increasingly exposed to a intersection attack. This can be referred to as anonymity degradation [11]. Dummy traffic and identities are indeed a way to increase the workload of an adversary during attack, but they are incapable of eliminating the attack itself.

6 Measuring anonymity

Anonymity can be considered as a young area within the field of security, when compared to the areas of confidentiality and authentication [15]. Measuring the level of anonymity a certain Anonymous communication service provides is a troublesome task. Partly due to the narrow safety marginals set for the services [16]. Since it has been established anonymity can take shape in various forms within the technical realm, privacy notions on the topic can be riddled with naming issues and lack of consistency. However, much like with other security measurements, the strength of anonymity can be assessed by the trouble an adversary is demanded to go through to overcome it [15]. For metrics to be applicable, success in reflecting precisely this is key.

By measuring the source-hiding properties of anonymous services, more specifically the ways in which the service is able to upkeep concealing the identity of the sender and the destination of the message, the workload of an adversary can be quantified [17]. Indeed, when defining the security measurements of a network according to metrics, the attacker's capabilities require to be defined [16]. Applying the right metrics and adversary models can turn out challenging.

Real anonymous communication systems are unable to reach the ideal of producing an uniform distribution of senders and receivers for a message enclosed by an anonymity set [16]. Distinguishing the direction of data flow will allow the adversary to make distinctions between senders and receivers. Thus the unevenness of the probability distribution can function as a useful metric for the degree of anonymity within ACSs.

By using attributes such as the entropy of an attackers probability distribution, the volume an adversary needs to match messages and users is quantified [17]. Nevertheless, anonymity measures based on entropy

have been criticized due to the shortcomings they exhibit. Entropy based measures are unable to take into the account the amount of knowledge the adversary is in possession of. Thus supplying an incomplete view of the level of compromised information, on which ACS design decisions be built upon.

7 Amplifying anonymity

Methods by which anonymous communication systems can amplify their anonymisation processes are under constant research. Design choices focus on advancing security through user-base enlargement and minimizing information disclosed to observers [16]. Unfortunately the implementation of these objectives are followed by the reduction of network capacity. Prioritizing anonymity in system architectures includes trade-offs set between security and performance, still the future of user needs demands for a balance to be struck.

Utilizing the Buddies architecture can provide level k anonymity to publishing clients within an ACS [11]. The Buddies architecture functions by denying clients publishing requests, when an ACS deems them vulnerable to a intersection attack. The biggest downside accompanying the buddies architecture is the latency it generates, through blocking clients from interacting [11]. Thus, the usability and availability of the system are deteriorated at the stake of anonymity. Hence, services which implement the buddies architecture should possess a high buffer for latency.

Danezis, Hayes and Troncoso [11] have lead the work Towards Anonymity sets that Persist. Hence, introducing TASP, a protocol for ACS's to facilitate grouping their clients into anonymity sets by gathering clients with similar patterns of communication together. In addition, anonymity could be amplified by using local randomisation [4]. By combining locally randomised inputs through secret-sharing with global mixing of the shares which are supplied by the initial anonymity processes taking place, inputs could be kept private.

The discussed methods of amplifying anonymity, all have a downfall when it comes to the usability of anonymous services. Some causing latency problems which can prove to be detrimental to user satisfaction. Possibilities might lie in improving the strength of anonymity sets and optimizing them to the fullest. To succeed in this additional research is required. Striving towards assuring the status of data integrity whilst

being able to avoid traffic analysis while utilizing anonymity amplifying measures is essential [5]

8 Conclusion

Every anonymous system can be seen as a double-edged sword [5]. Authors writing under pen names, newspaper articles published under pseudonyms and anonymous peer reviews are few of the examples where one can run into anonymity on the daily [1]. In light of these positive applications of anonymity one might be steered away from contemplating the possible harm causing effects. Including the possibility of cyber crime taking advantage of anonymity, through numerous unlawful action such as fraud as well as selling and distributing illegal commodities.

Anonymity alone remains insufficient to construct the notion of privacy thoroughly [4]. Still, anonymity can open doors for privacy by establishing a barrier between the action and the user [18]. When the context of distributed computation is utilized, anonymity hides the entities accountable for specified outputs. Enabling privacy would demand hiding every bit of information except the results of the outputs. Privacy is often framed as a trade-off taking place between social values and norms and individual needs [18]. Indeed, as Chawki [1] states: "Each problem relies on striking a fair balance between the interests of the individual on the other hand, and the interests of the state on the other." Anonymous systems have not been left unnoticed by government organizations. Reactions have been varied, some governments opting towards banning, while others sponsoring the systems. For example Tor is sponsored by The United States Defence Advanced Research Projects Agency (DARPA) [5].

The notion of whether anonymity is required or not, and the level of implementation needs to be on, depends on numerous aspects. A controlled setting in need of anonymity measures such as an election voting system, constructs a case for anonymity which is more predictable, when compared to the broad concept of communication on the internet. Constructing special anonymity regulations to establish standards considering measurements and models, could benefit the future research. Simultaneously steps should be taken to improve public awareness and enabling users to demand better privacy reinforcing technologies for themselves. Urging the research efforts towards anonymity systems, which hold up even when put against global adversaries, such as governments.

References

[1] M. Chawki, "Anonymity in cyberspace: Finding the balance between privacy and security," vol. 9, no. 3. Inderscience Publishers, 2010, pp. 183–199.

[2] R. Kang, S. Brown, and S. Kiesler, "Why do people seek anonymity on the internet? informing policy and design," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2013, pp. 2657–2666.

[3] I. Goldberg, D. Stebila, and B. Ustaoglu, "Anonymity and one-way authentication in key exchange protocols," vol. 67, no. 2. Springer, 2013, pp. 245–269.

[4] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Cryptography from anonymity," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 2006, pp. 239–248.

[5] R. A. Haraty and B. Zantout, "A collaborative-based approach for avoiding traffic analysis and assuring data integrity in anonymous systems," vol. 51. Elsevier, 2015, pp. 780–791.

[6] M. Simioni, P. Gladyshev, B. Habibnia, and P. R. N. de Souza, "Monitoring an anonymity network: Toward the deanonymization of hidden services," vol. 38. Elsevier, 2021, p. 301135.

[7] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity—a proposal for terminology," in *Designing privacy enhancing technologies*. Springer, 2001, pp. 1–9.

[8] G. T. Marx, "What's in a name? some reflections on the sociology of anonymity," vol. 15, no. 2. Taylor & Francis, 1999, pp. 99–112.

[9] A. Kwon, D. Lazar, S. Devadas, and B. Ford, "Riffle," vol. 2016, no. 2, 2016, pp. 115–134.

[10] R. A. Haraty, M. Assi, and I. Rahal, "A systematic review of anonymous communication systems." 2017, pp. 211–220.

[11] J. Hayes, C. Troncoso, and G. Danezis, "Tasp: Towards anonymity sets that persist," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2016)*, November 2016.

[12] Y. Wang, "Privacy-enhancing technologies," in *Handbook of research on social and organizational liabilities in information security*. IGI Global, 2009, pp. 203–227.

[13] C. Diaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *International Workshop on Privacy Enhancing Technologies*. Springer, 2002, pp. 54–68.

[14] S. Oya, C. Troncoso, and F. Pérez-González, "Do dummies pay off? limits of dummy traffic protection in anonymous communications," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2014, pp. 204–223.

[15] P. Syverson, "Why I'm not an entropist," in *Proceedings of Security Protocols XVII: 17th International Workshop, April 2009, Revised Selected Papers*, B. Christianson, J. A. Malcolm, V. Matyáš, and M. Roe, Eds. Springer-Verlag, LNCS 7028, 2013, pp. 231–239.

[16] S. J. Murdoch, "Quantifying and measuring anonymity," in *Data Privacy Management and Autonomous Spontaneous Security*. Springer, 2013, pp. 3–13.

[17] G. Tóth, Z. Hornák, and F. Vajda, "Measuring anonymity revisited," in *Proceedings of the Ninth Nordic Workshop on Secure IT Systems*. Citeseer, 2004, pp. 85–90.

[18] C. Kuhn, M. Beck, S. Schiffner, E. Jorswieck, and T. Strufe, "On privacy notions in anonymous communication," vol. 2019, no. 2, April 2019.

Task Allocation for Vehicular Fog Computing: A Survey

Mariam Moustafa
mariam.moustafa@aalto.fi

Tutor: Wencan Mao

Abstract

Collaborative driving has been the subject of numerous studies and experimentation throughout the years. Autonomous vehicles require heavy and time-sensitive processing of collected data in order to make safe decisions. Vehicular fog computing (VFC) is an emerging paradigm designed to fulfil time critical visual-based tasks using edge devices. VFC utilizes the resources co-located with the neighboring base stations and cars in order execute the required processing tasks. A VFC network must reliably allocate tasks in order to guarantee the efficient use of available resources. This paper reviews different task allocation techniques, compares how those techniques handle the dynamic VFC environment, and provides an analysis of the current state of task offloading implementation.

KEYWORDS: fog computing, vehicular fog computing, task allocation

1 Introduction

The emerging concept of autonomous cars has led current and future vehicles to be increasingly aware of their surroundings. This awareness can be translated into computational-intensive and latency-sensitive tasks, such as video processing and object recognition [12]. One possible solu-

tion to perform those tasks is to offload them to a cloud server. However, cloud servers are in remote locations and such time-sensitive computations cannot afford the latency of data processing in such locations. In order to avoid this delay, the notion of fog computing was proposed [14]. The idea behind the fog computing paradigm is to allocate computing resources closer to the end devices that require data processing. Therefore, fog computing solved the problem of round-trip delays cause by communicating with a cloud server [8]. Researchers took this concept a step further and proposed Vehicular Fog Computing (VFC). VFC is an extension of the edge computing paradigm where vehicles can act as fog nodes. It is envisioned because the technological advancements of cars will lead to computational resources that might be under-utilized [11]. The VFC paradigm can utilize these potential idle resources by including vehicles as part of the fog resources in the network [8], [6]. Therefore, this paradigm can reduce the idle state of resources. Furthermore, it is better suited for the demands required by smart and context-aware vehicles because data-intensive tasks can be offloaded to either stationary or mobile fog nodes [14]. The flexibility of mobile fog nodes would also increase the geographical area of serving cars in need of data computation as the nodes are now moving closer to the consumers [8].

While VFC leverages the under-utilized resources in vehicles by adding the support of mobile fog devices, critical challenges emerged [11]. The flexibility and mobility provided by a VFC network can lead to synchronization problems, such as communication loss. One of the other major open issues of VFC is task allocation. Task allocation is the process of assigning tasks to fog nodes, whether mobile or stationary. During this process, factors such as available resources, energy consumption, location of vehicles, the degree of quality loss and time-sensitivity need to be taken into consideration [8], [9], [14]. These factors affect not only the individual performance of task offloading but also the overall performance of the VFC resource provisioning. There are several optimization schemes and architecture implementations that try to perform task allocation while considering some of the aforementioned factors. Modern research has investigated task allocation using different methods, including particle swarm optimization [14], reinforcement learning [6], and Markov chains [12].

This paper aims to review and compare some of the latest approaches proposed for task allocation in VFC. The rest of the paper is organized as

follows. Section 3 explains the various methods proposed for task allocation in VFC. Section 4 compares and analyzes the different approaches of related papers. A discussion of existing issues and future direction is explored in Section 5. Finally, Section 6 provides some concluding remarks and insights on the topic.

2 Background

The principle of VFC was established because vehicles are evolving to posses powerful computing capabilities that enables them to perform complex data processing [7]. Therefore, vehicles can now be viewed as possible resource providers as opposed to just resource consumers [5]. Figure 1 demonstrates the common architecture of a VFC network. The architecture typically consists of two kinds of devices that perform data processing in the edge. Those two kinds are the immobile devices, such as Road Side Units (RSUs) and base stations, and mobile devices, such as cars and busses. Moreover, cloud servers are used in computation-intensive tasks that are not time-critical [2]. This architecture is flexible enough to allow

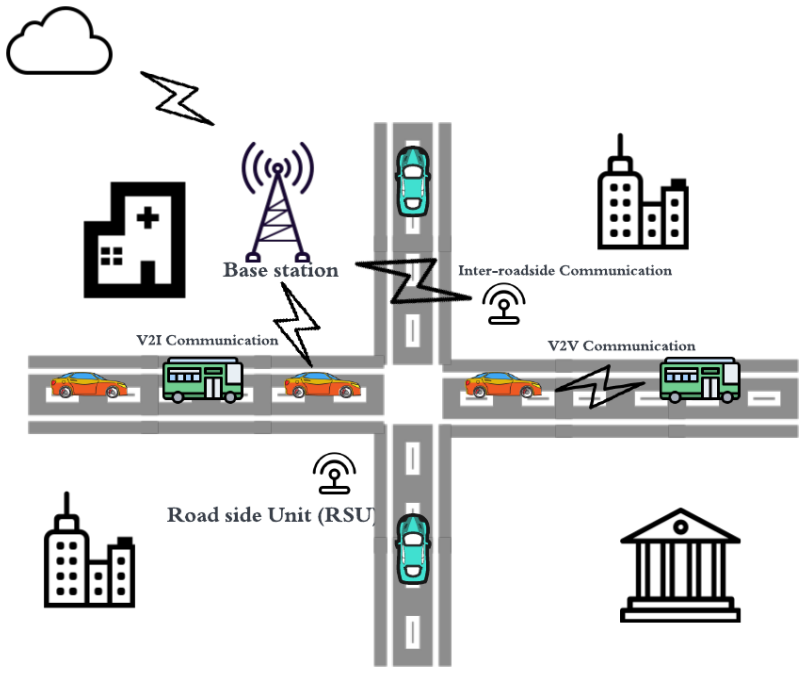


Figure 1. VFC network architecture (Image by Mariam Moustafa 2022)

different kinds of devices to communicate with each other in order to efficiently utilize all resources in this network. Figure 1 illustrates three

different types of communication in a VFC network:

1. Vehicle-to-Vehicle (V2V) communication between two vehicles where one is a resource consumer and the other is a resource provider.
2. Vehicle-to-Infrastructure (V2I) communication between a vehicle and a stationary device. This kind of communication is done either for resource provisioning when a vehicle subscribes itself to the network or for resource consumption when a vehicle needs to have a time critical task executed.
3. Inter-roadside communication between two immobile devices with the goal to share information about current devices and tasks in the network.

However, the flexibility of this network resulted in a task allocation challenge. In other words, finding an efficient use and distribution of the network resources became an essential problem in need of a solution. The following section will present the different approaches researchers investigated to perform efficient and reliable task allocation.

3 Related Work

3.1 Binary Particle Swarm based Optimization

Zhu et al. [14] investigated task allocation in VFC as an optimization problem between the quality loss and latency. The architecture includes a coordinator called the zone head, which knows all the capabilities of devices in the network, their location, and their speed and assigns tasks accordingly. The zone head is able to assign tasks to nodes by utilizing a heuristic algorithm known as Binary Particle Swarm based Optimization (BPSO). The algorithm creates a set of particles where each particle is a vector. The particle maps each task to a fog node and determines the quality that this fog node is able to process the task with. Then, the algorithm runs for a determined number of times trying to minimize the quality loss for all tasks while decreasing latency. The particle that is able to balance between the quality loss and latency the most is used as the mapping.

Optimization Components	Approach in [14]	Approach in [1]
Objective	minimize the maximum service latency and total quality loss	maximize reliability of tasks
Inputs	the maximum service latency and the set of quality loss of results (QLR) levels	transmission rate between edge node and the RSU, data size, CPU cycles per task, and computation time
Constraints	quality loss, latency and vehicle capacity	latency
Decision variables	task assignment decisions and their corresponding QLR selections	position and velocity vectors representing how optimal is the current task allocation

Table 1. Summary of optimization techniques for task offloading in VFC

3.2 Fault-tolerant Particle Swarm Optimization

Hou et al. [1] utilized the concept of software defined networks (SDN). SDNs contain global information of the domain and are able to orchestrate the edge computing resources. Hou et al. [1] also introduced a heuristic algorithm called Fault tolerant Particle Swarm Optimization for Maximizing Reliability (FPSO-MR). The algorithm provides several solutions, represented as vectors, for optimizing task allocation. The algorithm adjusts these vectors iteratively in order to represent the optimal solution.

3.3 Summary of Optimization Techniques

Both approaches in sections 3.1 and 3.2 formulate task allocation as an optimization problem to be solved. An optimization problem is composed of an objective, inputs, constraints and decision variables. A decision variable refers to the values an optimization problem finds that satisfy the given constraints. In VFC, it represents possible task assignments [3]. Table 1 summarizes those key aspects of the optimization problems discussed in the previous sections.

3.4 Markov Chains

Zhu et al. [12] approached task allocation from a different perspective. Constant communication with a central node is not feasible due to the

high mobility of vehicles. Therefore, distributed task offloading was used such that client vehicles perform task offloading by themselves. The authors define the workload of a fog node as the current number of neighboring clients. This number changes according to a particular time bucket. The fog nodes translate those workloads and time buckets into a probability transition matrix that describes the transition of the Markov chain. In other words, it describes how the workload will change from one time bucket to another based on probabilities. The client then requests from nodes that are one-hop away this matrix and decides which node can best handle the tasks this client needs.

3.5 Deep Q-Networks

Similar to the approach in [12], Zhu et al. [13] also modelled task allocation as a Markov decision process. The aim in [13] is to maximize the quality of information (QoI) while also minimizing the data processing latency. The authors proposed converting commercial vehicles, such as busses, to Vehicular Fog Nodes (VFNs) which are equipped with the necessary devices to perform required computations. Based on this assumption, Zhu et al. [13] utilize Deep Q-Networks (DQNs) to solve the optimization problem of increasing QoI and decreasing latency. An agent running a DQN is able to adapt and learn from previously seen workload patterns in VFNs.

3.6 Deep Reinforcement Learning

In [6], Shi et al. proposed using deep reinforcement learning for task offloading. The approach focused on incentivisation techniques to motivate cars for sharing their idle resources while taking into consideration the priority of tasks and the availability of computing devices. Shi et al. used a deep reinforcement learning (DRL) algorithm known as soft actor-critic (SAC), which is more efficient in solving problems with high-dimensional action space than DQN. In other words, SAC allocates task more efficiently under different traffic densities. The algorithm works by maximizing the reward and the entropy of policy to offload tasks in a robust and efficient manner. The entropy of policy represents how informative is the choice of task allocation.

RL Components	Approach in [12]	Approach in [6]
Agent	a DQN agent in zone head that assigns tasks to VFNs and selects the data collection rate	- an agent in base station that determines the fog node and corresponding unit service price - a critic is implemented for policy evaluation
State	geographical information, camera configuration (client) and number of neighboring client vehicles (fog node)	wireless link between client and fog node, remaining computing resouources, data size and task utility
Action	a pair containing the fog node the task is assigned to and the frame rate the node will use in processing	choosing a service vehicle for each task and determining the corresponding price
Reward	weighted difference between the number of pixels a node processed and task duration	directly proportional to the utility of the assigned task

Table 2. Summary of RL for task offloading in VFC

3.7 Summary of Reinforcement Learning Techniques

Sections 3.5 and 3.6 utilize variants of reinforcement learning (RL) in order to perform task offloading. Reinforcement learning rewards a task assignment based on the suitability of the allocation. The task allocation problem can be formulated as a markov decision process (MDP) [4]. The components of an MDP include agent, state, action, and reward. Table 2 summarizes how the papers utilizing RL defined these components.

3.8 Matching-learning

Zhou et al. [10] represented the actions needed in VFC networks as a two-stage framework. The first stage is recruiting vehicles with idle resources to act as a fog server, while the second stage is allocating tasks to those recruited devices using a matching algorithm. Vehicles running the matching algorithm first orders the fog servers by their preference, which is inversely proportional to task offloading delay. The vehicle then requests a match from the top candidate on the list. If this candidate has only one request, then the vehicle's tasks are allocated to the candidate. Otherwise, if the fog server receives multiple requests, it raises its costs and the vehicles repeat the ordering step again.

4 A comparison of task allocation techniques

Several factors have to be considered when implementing task allocation techniques in VFC networks. One factor is the mobility of VFNs. A task allocation mechanism should anticipate and react to a fog node leaving the network. Hence, the algorithm should consider the remaining time of the fog node in the network and assign tasks that can be completed within this time. The priority of a task is also an important factor, it is essential to serve tasks that require immediate responses first as failure to accomplish these tasks in a timely matter might impact the road. Another factor is the quality of the result of an offloaded task. A task allocation algorithm should also be able to determine the number of tasks it can offload in a given time window as well as handle tasks with variable time lengths. This section will compare how the different approaches discussed in Section 3 consider these factors.

4.1 Mobility

Several factors affect how long a vehicular fog node remains in the network, including speed, traffic congestion, and traffic signals. These factors are directly correlated to how long a fog node can provide a service. The controller responsible for task allocation should estimate the task duration and how long the vehicle remains within the communication range of other users.

Zhu et al. [12] created a distributed task allocation scheme to accommodate the high mobility of vehicles. Instead of having a central node assigning the tasks, each client vehicle offloads its tasks by itself. A client vehicle broadcasts a probe message to surrounding vehicular fog nodes that are one-hop away and gets the capabilities of those vehicles as a response. Afterwards, the client vehicle chooses the most suitable fog node. If the quality of the connection between two vehicles decreases to a certain threshold, the vehicular fog node notifies the client vehicle to stop offloading tasks and the connection between those two vehicles will stop after finishing any tasks already offloaded. This implementation is able to gracefully end the connection between moving cars and guarantees the completion of allocated tasks.

Zhu et al. [14] implement a more reactive approach. The zone head in the network is triggered whenever a service interruption occurs because of moving vehicles. The zone head assigns another fog node the interrupted

Paper	Approach
[12]	a distributed task allocation approach where clients offload tasks to fog nodes independently
[14] and [13]	zone head is triggered by service interruption caused by moving vehicles
[6]	evaluation of vehicle speed, traffic density and distance before task assignment
[1]	transfer control from one RSU to another according to vehicle proximity

Table 3. Comparison of mobility-handling techniques

task for execution. Similar to [14], Zhu et al. [13] also respond to the service interruption in the same way. Furthermore, the timing, location and overall state of the network at the time of interruption is saved and used by the Q-learning algorithm to learn task allocation strategies.

Shi et al. [6] consider the mobility of VFNs before assigning tasks. The speed of vehicles, traffic density and distance between client and fog node are evaluated before allocating any tasks. This approach would able to provide an estimate of how long the two parties will be within each other's range. This estimate would help the model assign a more suitable task. Hou et al. [1] provide a VFC architecture that relies on V2I communication. RSUs control and communicate with clients in order to mitigate and avoid dynamic changes in the network caused by high mobility. The designed algorithm measures the distance between the available RSUs and the mobile client at consecutive time intervals and determine the best RSU that a client can be assigned to. Table 3 summarizes how each paper handles mobility.

4.2 Quality of information and quality of service

Most investigated research papers associate the quality of information of task results with the quality of service (QoS). One aspect of QoI is the image or video quality used in processing while QoS is concerned with the latency or time it takes to perform a task. These two concepts affect each other because in order to process high quality frames for better results, more communication time, resources and processing is required. These requirements increases the overall latency of the network. Most approaches involve formulating the trade-offs between QoI and latency as an optimization problem.

Paper	Approach
[12]	formulate task offloading as a dynamic programming problem that rewards fast and accurate task results
[14]	optimization problem that minimizes both variables latency and total quality loss.
[13]	reward system that decreases as processing latency increases and increases as the quality of information increases
[6]	RL to maximize latency-awareness in fog devices
[1]	stationary RSUs to avoid latency from mobile devices

Table 4. Comparison of mobility-handling techniques

Zhu et al. [12] assume that quality of information is directly proportional to image resolution. Their task offloading scheme is able to reduce service latency while increasing the QoI levels their scheme is able to support. Task offloading is formulated as a dynamic programming problem that increases the rewards of the network in each time bucket depending on how well a device was able to balance between the quality of the video and the time it takes to process the video. On the other hand, Zhu and Tao et al. [14] formulate the problem as a mathematical equation involving the two variables latency and total quality loss. Their goal is to minimize both the latency and quality loss. Binary particle swarm optimization algorithm is used to solve the equation.

In order to balance between QoI and latency, Zhu et al. [13] created a reward system that decreases as processing latency increases and increases as the quality of information increases. The goal of the architecture is to maximize the overall system reward.

Other implementations only consider latency. For example, Shi et al. [6] use reinforcement learning to in order to maximize latency-awareness in fog devices, while Hou et al. [1] propose using stationary RSUs in order to avoid any additional latency. Table 4 provides a comparison of the various task offloading techniques used to increase QoI and decrease latency.

4.3 Varying task count and duration

A VFC task allocation method should take into account the uncertainty of the number of tasks in the network at any given time. An important point to note is how many tasks can a VFC task allocation algorithm process at any given time without affecting the network. Another factor to note in a

Paper	Assumptions
[12]	all the tasks are the same and have the same duration
[14]	tasks are either video processing or object recognition and have the same duration
[6]	networks have varying number of tasks but task processing occurs only in fixed timeslots
[10]	fixed number of tasks with the same duration

Table 5. Assumptions about task variability and duration

task allocation implementation is whether the implementation is adjusted for tasks with varying processing times.

The implementation of Zhu et al. [12] assumes that the the computing tasks are homogeneous. In other words, the implementation does not support different kinds of tasks. The authors also limited the number of tasks processed simultaneously in the network to a fixed number. Zhu and Tao et al. [14] assume two kinds of different tasks (video streaming and real-time object recognition) and the tasks are run with different qualities. The method also assumes that there are numerous vehicles that can generate a various number of tasks with a given threshold. Such an implementation is robust against differing number of tasks. However, the authors assume that each task lasts for a fixed length of time and hence the implementation is not adjusted for tasks with different durations. Shi et al. [6] assume a network with a variable range of vehicles per kilometer. The number of vehicles are proportional to the number of tasks. However, their implementation assumes processing a task in a fixed time slot. Hence, if a task is completed before its assigned time slot ends, the fog node carrying out the task will remain idle for the remainder of the time slot. Zhou et al. [10] performed a matching between a fixed number of vehicles and servers. All tasks are assumed to have the same duration, thus the time of each task is not incorporated in the algorithm. Table 5 explains how these approaches handle the uncertain number of tasks with different duration.

5 Analysis and Discussion

While most of the discussed papers use real-world data for simulating various scenarios, there still lacks a real-world implementation of VFC.

Despite of this, simulations provide the environment with flexibility to create different architectures and varying network traffic in order to evaluate the effectiveness of the proposed task allocation solutions.

In addition, most of the investigated task allocation approaches operated with the underlying assumption that tasks are homogeneous. While this assumption is enough to provide a proof of concept, different kinds of tasks are required when implementing a visual-based autonomous driving car. These tasks include video processing, object recognition, and mathematical calculations. Moreover, the heterogeneity of computing resources must be studied in a task allocation implementation. In other words, different tasks require different hardware resources. An RL model would require graphics processing units (GPUs), while dynamic programming or matching approaches would rely more on CPUs. Therefore, an implementation should consider heterogeneous tasks that require different hardware capabilities.

Another important area of study in task offloading is error handling. Task allocation techniques should proactively handle various types of errors in the network, including network errors, task failure or resource overload. A robust system should proactively handle such failures by updating the resource allocation plan in order to prevent catastrophic consequences of not processing critical or safety-related tasks on time.

6 Conclusion

This paper provides a brief background on vehicular fog computing. In addition, it summarizes several task allocation techniques and highlights how those techniques try to optimize task offloading assignments. Those approaches include formulating task allocation as an optimization problem, creating a reinforcement learning algorithm that rewards a task assignment and running matching algorithms that assign tasks based on preferences. Furthermore, the paper provides a comparison between those various approaches. Several aspects of a VFC network are considered in the comparison, including the dynamic nature of the VFC environment, quality of information, latency and uncertainty of task count and duration. Finally, the paper touches some of the open issues in task allocation in VFC.

References

[1] Xiangwang Hou, Zhiyuan Ren, Jingjing Wang, Wenchi Cheng, Yong Ren, Kwang-Cheng Chen, and Hailin Zhang. Reliable computation offloading for edge-computing-enabled software-defined IoV. *IEEE Internet of Things Journal*, 7(8):7097–7111, 2020.

[2] Cheng Huang, Rongxing Lu, and Kim-Kwang Raymond Choo. Vehicular fog computing: Architecture, use case, and security and forensic challenges. *IEEE Communications Magazine*, 55(11):105–111, 2017.

[3] Seung-Seob Lee and Sukyoung Lee. Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information. *IEEE Internet of Things Journal*, 7(10):10450–10464, 2020.

[4] Zhaolong Ning, Peiran Dong, Xiaojie Wang, Lei Guo, Joel J. P. C. Rodrigues, Xiangjie Kong, Jun Huang, and Ricky Y. K. Kwok. Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme. *IEEE Transactions on Cognitive Communications and Networking*, 5(4):1060–1072, 2019.

[5] Zhaolong Ning, Jun Huang, and Xiaojie Wang. Vehicular fog computing: Enabling real-time traffic management for smart cities. *IEEE Wireless Communications*, 26(1):87–93, 2019.

[6] Jinming Shi, Jun Du, Jingjing Wang, Jian Wang, and Jian Yuan. Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12):16067–16081, 2020.

[7] Xuefeng Xiao, Xueshi Hou, Xinlei Chen, Chenhao Liu, and Yong Li. Quantitative analysis for capabilities of vehicular fog computing. *Information Sciences*, 501:742–760, 2019.

[8] Jindou Xie, Yunjian Jia, Zhengchuan Chen, Zhaojun Nan, and Liang Liang. Efficient task completion for parallel offloading in vehicular fog computing. *China Communications*, 16(11):42–55, 2019.

[9] Rahul Yadav, Weizhe Zhang, Omprakash Kaiwartya, Houbing Song, and Shui Yu. Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing. *IEEE Transactions on Vehicular Technology*, 69(12):14198–14211, 2020.

[10] Zhenyu Zhou, Haijun Liao, Xiaoyan Wang, Shahid Mumtaz, and Jonathan Rodriguez. When vehicular fog computing meets autonomous driving: Computational resource management and task offloading. *IEEE Network*, 34(6):70–76, 2020.

[11] Zhenyu Zhou, Pengju Liu, Junhao Feng, Yan Zhang, Shahid Mumtaz, and Jonathan Rodriguez. Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach. *IEEE Transactions on Vehicular Technology*, 68(4):3113–3125, 2019.

[12] Chao Zhu, Yi-Han Chiang, Abbas Mehrabi, Yu Xiao, Antti Ylä-Jääski, and Yusheng Ji. Chameleon: Latency and resolution aware task offloading for visual-based assisted driving. *IEEE Transactions on Vehicular Technology*, 68(9):9038–9048, 2019.

[13] Chao Zhu, Yi-Han Chiang, Yu Xiao, and Yusheng Ji. FlexSensing: A QoI and latency-aware task allocation scheme for vehicle-based visual crowdsourcing via deep Q-Network. *IEEE Internet of Things Journal*, 8(9):7625–7637, 2021.

[14] Chao Zhu, Jin Tao, Giancarlo Pastor, Yu Xiao, Yusheng Ji, Quan Zhou, Yong Li, and Antti Ylä-Jääski. Folo: Latency and quality optimized task allocation in vehicular fog computing. *IEEE Internet of Things Journal*, 6(3):4150–4161, 2019.

Benefits and Drawbacks of Using Microservices in Big Data Platform Applications

Buket Karakas
buket.karakas@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

An architecture of microservices provides independent components that contribute to a business model. It is scalable, highly maintainable and loosely coupled. An architecture utilizing microservices is intended to solve universal or general problems. We will explore a different type of architecture to better understand microservice architecture, which is monolithic architecture. Often, monolithic architectures consist of a single system providing needed functionality and deployed as a whole. This paper discusses and analyzes the benefits and disadvantages of using a microservices architecture for developing a big data application. Microservices are used in various sectors. One important usage area of microservice architecture discussed in this article is smart city projects. The application for city transportation system is developed using a microservice-based architecture, which gave satisfactory results. Another area that is analyzed in this paper is machine learning algorithms used in big data applications. Another microservice-based architecture is proposed with multiple layers to create most efficient structure. Considering two usecases and overall characteristics of both architectures, the choice to use microservices should be based on the detailed project structure. Microservices can be an excellent choice depending on the complexity and requirements of the project.

KEYWORDS: Microservices, Monolithic, Big Data

1 Introduction

The data produced by millions of people every day is one of the most significant assets in a business. Currently, such large amounts of data that can not be handled with simple relational databases and technologies is called Big Data [10]. Big Data has a major impact on organizations because it provides meaningful information about user behaviors [9]. However, it is not easy to process this data and use it effectively in the decision-making process [10]. This challenge is mainly caused by the five characteristics of Big Data [3]. The first is the volume of the data. The increasing amount of data is a major challenge for Big Data applications. The second is the velocity of the data, which means how frequently the data is arriving to the system. In Big Data applications, such as IoT systems, this can be a major problem. The third characteristic is the diversity of the data. Incoming data can be different from each other, and they can also be unstructured. The fourth characteristic is the authenticity of the data. It needs to be protected at each step of structuring and processing. The last property is the value property. It focuses on the effectiveness of the data.

These characteristics bring different challenges with them. Conventionally, these challenges are solved by monolithic software solutions where the application is tightly coupled [10]. The problem with using this kind of architecture is that it becomes laborious to maintain with the large size of the application. Therefore, there is a need for a scalable solution to overcome the obstacles of the traditional way of designing.

One of the possible solutions is to use microservices architecture. Microservices are autonomous services that construct a more complex structure to solve different kinds of problems [4]. The emphasis is on modularity for this architecture type. Therefore, it provides a more agile and scalable solution. However, using microservices comes with a number of disadvantages, including refactoring the services and granularity.

This paper discusses and analyzes the benefits and drawbacks to using microservices architecture in the context of developing a big data application.

This paper is organized as follows. Section 2 defines monolith architecture and microservices architecture. Section 3 presents two use cases of microservices architectures. Section 4 discusses the usage of microservices in big data applications. Section 5 gives concluding remarks.

2 Comparison of Monolithic and Microservices Architectures

This section discusses the advantages and disadvantages of a microservices architecture relative to the monolithic architecture in general. An overview of this comparison can be also seen in Figure 4.

2.1 Monolithic Architectures

In monolithic architectures, the entire system is deployed as a single unit that provides all of the required functionality [7]. The single-process architecture is one type of this architecture. In this kind of architecture, the code is deployed as a single process. This is one of the basic monolithic systems that can be seen in Figure. 1.

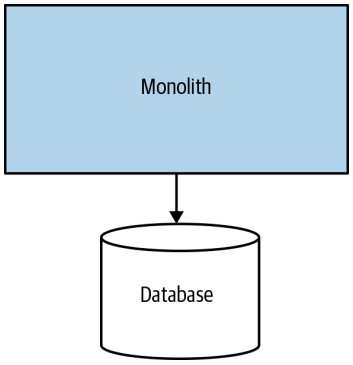


Figure 1. Single-Process Monolith [7]

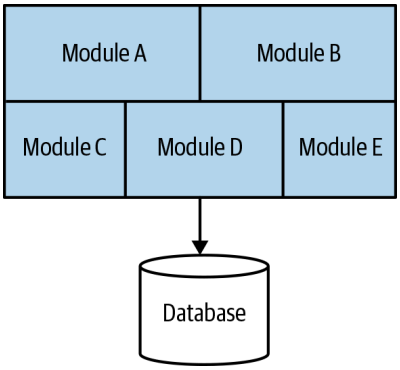


Figure 2. Modular Monolith[7]

Another type of monolithic system is the modular monolith [7]. In this architectural type, the application contains multiple modules that solve separate problems. Although the system has different modules, all modules use the same database, which can be seen in Figure 2.

The final monolithic architecture type is the distributed monolith where also databases differ according to different modules [7], illustrated in Figure 3. While it is similar to microservices, all modules are still deployed together.

Monolithic architecture has benefits for a small group of developers and a small size project, since it allows for simplicity, fewer configuration issues, and simplified design [7]. However, In terms of Big Data applications, this architecture is insufficient to develop an efficient solution.

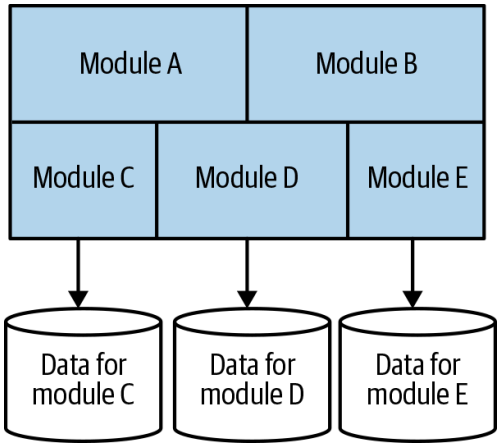


Figure 3. Distributed Monolith [7]

2.2 Microservices Architecture

Microservices, which support business models, are composed of independent components [2]. The foundation of microservices consist of different functional services. These services compose a more complex system. The independent services are considered as black-box services since their internal implementation is hidden and irrelevant for the people who want to use them. Therefore, most of them have their own database [6].

The microservices architecture can be clarified through a few concepts. The first concept is independent releasability [7]. A change in mindset is required when developing an application for independent releasability. There should be explicit design and loose coupling of all services. This ensures that when a change is required, only services that are directly related to that change will be deployed. By doing this saves time and energy associated with deploying an entire application. The second concept is the size of each microservice. This is highly debatable among the researchers. Newman [7] believes that the number of microservices that the system can handle and the boundaries around certain microservices are more important than their size. The key is to ensure a loosely coupled architecture and to hide information between microservices. The final concept is the flexibility. The microservice architecture allows for different technologies and programming languages to be used for each service without affecting their relationship and the overall application.

There are advantages and disadvantages to these concepts. One of the most obvious advantage of using microservices is technology heterogeneity [7]. Applications can be developed using a variety of technologies depending on their performance needs. An example of using different tech-

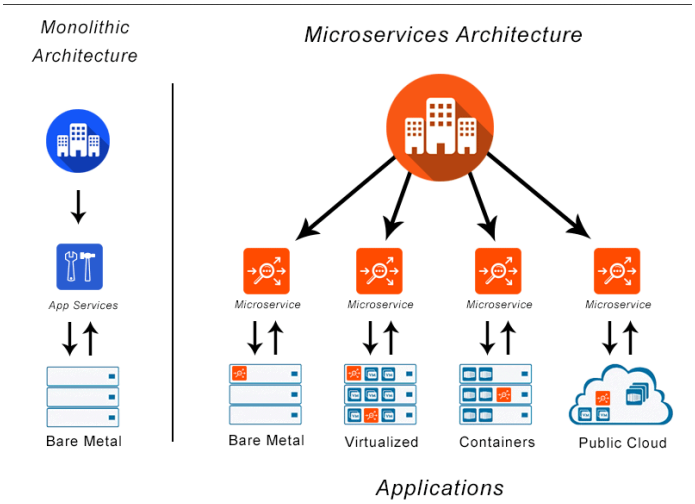


Figure 4. Microservices vs Monolithic Architecture [5]

nologies can be seen in Figure 5. This heterogeneity comes with overhead, which should not be ignored.

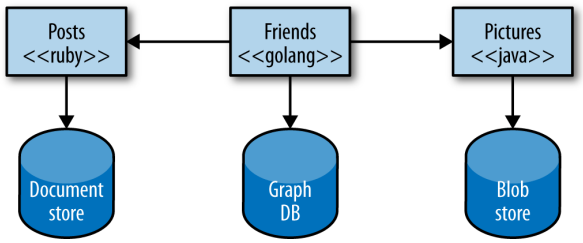


Figure 5. Using different technologies [7]

Another important advantage of using microservices is its resilience towards system fails [7]. In monolithic systems, whole system fails if a service fails. However, in microservices, system can continue to work with reducing overall functionality if a service fails. This resilience brings also a downside which is configuring and maintaining distributed systems. It is a challenging task to maintain all services since network problems are likely to occur.

In addition to the resilience and heterogeneity, microservices provide a major scalability advantage. In monolithic systems, whole system needs to be scaled when it is required. In contrast, microservices allow only to scale the services that needs to be scaled instead of the entire system.

3 Usecases of Microservices in Big Data Applications

Microservices are used increasingly in big data applications for various reasons. The following subsections present a microservices architecture for smart city systems and a microservices architecture for machine learning and data analysis processes.

3.1 Microservice Based Architecture for Smart City Systems

Currently, intelligent transportation systems (ITSs) are a major part of city planning and development [1]. These systems need to increase the service performance in order to satisfy all the parties involved in the project. The data collected from the transportation system presents a number of challenges since it is huge and inconsistent. The 5 major challenges are:

1. Fluctuating data structure

2. Changing structure of the output corresponding to the different needs

3. Volume of the data and handling it with an efficient algorithm

4. Post-processing the data for making it more readable

5. Producing insights from the data.

Asaithambi [1] proposes a Microservices-Oriented Big Data Architecture (MOBDA) that makes it easier to create an effective platform for smart transportation and analytics. This platform is designed to address different requirements of ITS. These requirements split into two major categories: functional microservice performance requirements and non-functional performance requirements. Functional microservice performance requirements include storing big data in a secure platform, real-time data reporting, data visualization, predictive modelling for transport services, and creating API for consuming data to produce useful insights. Non-functional requirements include reliability of public transportation system, availability of seats on public transportation, congestion on public roads, queuing information for public transportation, monitoring public transport incentives, predicting the demand, and information on traffic

enforcement. MOBDA fulfills these requirements using a hybrid architecture and having multiple layers. It has a distributed broker backend that works as a middleware between different layers, which can be seen in Figure 6. In this architecture, the microservices are not structured as the smallest possible unit. Instead, they are developed in the right size in order to enhance processes. As a result of this, microservices applications both in real time stream processing and batch processing layer, work together closely with the other components in the layer. The use of microservices enabled non-real time services to be easily deployable and loosely coupled. In addition, overall architecture is more scalable and adaptable.

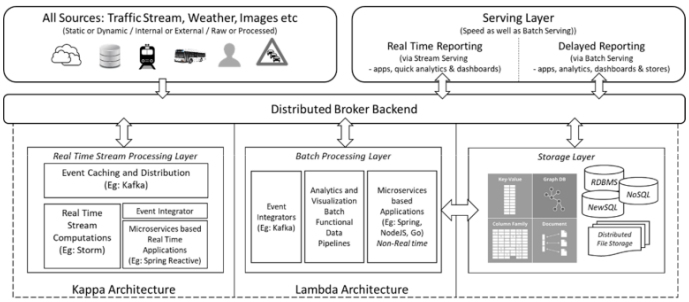


Figure 6. Overview of MOBDA [1]

This microservice-based architecture was tested on the data that is obtained from Singapore transportation system [1]. The use of MOBDA on this dataset resulted a successful computation of bus service headway metrics in terms of a reporting service in ITS.

3.2 Machine Learning and Data Analysis in Big Data Applications using Microservices Architecture

Use of machine learning algorithms carries a great importance in the context of big data applications. This usage presents great challenges. Therefore, Shahoud [8] proposes an architecture using microservices and web technologies to address these challenges. This framework consists of three layers, which are UI layer, service layer and persistence and processing layer.

UI layer is responsible for jobs execution of UI, model management, data management and cluster configuration [8]. It has different applications for each UI. The service layer is utilizing microservices to provide different functionalities which are scalable and independently deployable. It

has two services, which are data management service and job management service. There is one responsibility per service and each service is independent. The persistence and processing layer is providing data storage and modelling facilities. The overall architecture can be seen in Figure 7.

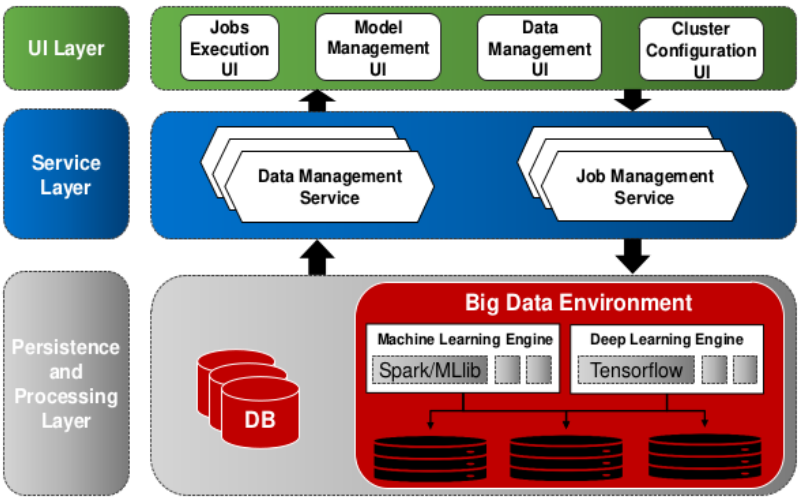


Figure 7. Overview of ML architecture in Big Data Applications [8]

The use of two microservices in the service layer seems to make overall architecture work more efficiently [8]. The evaluation of the framework demonstrates high performance and low overhead in the model. Furthermore, the architecture has been tested with increasingly large amounts of data and the results are satisfactory with low overhead, which is critical for big data applications.

4 Discussion

The use of microservices for big data applications has become increasingly popular. Analyzing the project structure thoroughly is essential to making the right decision. The highly scalable and customizable nature of microservices makes them an effective way to implement large-scale applications. However, monolithic architecture is easier to develop in small-scale applications because of its straightforward structure.

In big data applications, using microservices in computation-heavy layers is usually the most efficient solution. The layers of big data applications include data processing, data wrangling, stream analytics, and batch analytics. The use of microservices means that these layers are indepen-

dent of each other, allowing the application to continue to function when one service fails. Furthermore, layers can be scaled to suit their needs without affecting everything at once. As a result, it saves money and computation power while increasing performance. Utilizing microservices features allows the application to be more efficient by choosing different technologies based on the task.

Despite all these advantages, there are also some drawbacks. Due to the different services provided by the layers, communication is typically done through a network configuration. Therefore, more work has to be done. There can be problems within the network as well, and the team has to be knowledgeable enough to handle these problems effectively. In addition to that, their security needs to be ensured, since they're not part of the same component anymore. Communication between them needs to be protected.

5 Conclusion

Microservices provide an example of what software systems will look like in the future. Its scalability, autonomy, and modularity creates an effective solution to some of the challenges that come with complex software systems. This paper examines the use of microservices in big data applications. The paper illustrates usefulness of microservices with two examples. The first is in terms of transportation data, while the second is in machine learning. Both of these data-intensive applications utilize microservices to increase their performance.

Even though microservices can dramatically improve the performance of a project, the disadvantages should also be acknowledged, and each project must be thoroughly understood in order to build an effective architecture. Big data applications benefit from microservices more than monolithic architectures. In relation to big data applications and microservices, more research can be conducted since there are not many papers on the topic at present.

References

[1] Suriya Priya R. Asaithambi, Ramanathan Venkatraman, and Sitalakshmi Venkatraman. Mobda: Microservice-oriented big data architecture for smart city transport systems. *Big Data and Cognitive Computing*, 2020.

[2] Antonio Bucchiarone, Nicola Dragoni, Schahram Dustdar, Patricia Lago, Manuel Mazzara, Victor Rivera, and Andrey Sadovykh. *Microservices Science and Engineering*. Springer Nature Switzerland AG, 2020.

[3] Andreas Freymann, Florian Maier, Kristian Schaefer, and Tom Böhnel. Tackling the Six Fundamental Challenges of Big Data in Research Projects by Utilizing a Scalable and Modular Architecture. In *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security*, page 249–256, Prague, Czech Republic, 2020.

[4] Pooyan Jamshidi, Claus Pahl, Nabor C. Mendonça, James Lewis, and KStefan Tilkov. Microservices: The journey so far and challenges ahead. *IEEE SOFTWARE*, 35(3):24 – 35, 2018.

[5] AVI Networks. Microservices definition.

[6] Sam Newman. *Building Microservices*. O'Reilly Media, Inc., 2015.

[7] Sam Newman. *What Are Microservices?* O'Reilly Media, Inc., 2 edition, 2020.

[8] Shadi Shahoud, Sonja Gunnarsdottir, Hatem Khalloof, Clemens Duepmeier, and Veit Hagenmeyer. Facilitating and managing machine learning and data analysis tasks in big data environments using web and microservice technologies. In *MEDES '19*, page 80–87, Limassol, Cyprus, 2019.

[9] Neelam Singh, Devesh Pratap Singh, Bhasker Pant, and Umesh Kumar Tiwari. µbigmsa-microservice-based model for big data knowledge discovery: Thinking beyond the monoliths. *Wireless Personal Communications*, 2020.

[10] Daniel Staegemann, Matthias Volk, Aamir Shakir, Erik Lautenschlager, and Klaus Turowski. Examining the interplay between big data and microservices – a bibliometric review. *Complex Systems Informatics and Modeling Quarterly (CSIMQ)*, 27(157):87 – 118, 2021.

Assessment of Security Challenges Encountered in Microservice Architecture Compared to Traditional Monolithic Architecture

Bojana Bakić
bojana.bakic@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

A paradigm that has emerged in the last decade and generated considerable commotion in the domain of software design is microservice architecture. This design principle is based on service-oriented architecture (SOA) that treats services as components of the system rather than the final output. It is recognized as a step away from the traditional monolithic architecture due to the modularity of source files that make it more scalable. Although it is a state-of-the-art innovation, the implementation of microservice architecture bears a handful of security risks that will be evaluated in this seminar paper and compared to the monolithic architecture.

KEYWORDS: software architecture, monolith, microservices, security

1 Introduction

Software architecture dictates the way applications are written and structured. At present, most applications follow the design principles of monolithic software architecture, with the standard approach providing an all-in-one solution available on a single platform of choice, proving trustworthy in terms of initial development, tests and deployment [6]. However, in cases when an application needs to be modified and scaled, developers

may run into obstacles due to its complexity. This is where microservices can make a difference as they provide a more lightweight, adaptable solution consisting of multiple independent services [15]. Each service has its own business objective, database, and sometimes it can also employ a different programming language. What allows this distributed system to act as a whole is services communicating among themselves over the network and working towards the same goal. Consequently, these smaller units facilitate faster development.

Recently, the information security field has simultaneously advanced and gained significant public attention due to numerous security breaches. Unfortunately, the consequences of these attacks have become greater by the day prompting a substantial demand for secure output. New technologies and products are closely inspected for flaws and microservices have not escaped this scrutiny.

This paper examines both the pervasive monolithic architecture and its successor, microservice architecture. Security aspects, such as authentication, authorization, data protection, defense against cyber attacks and many more are analyzed and correlated between these software design concepts.

The rest of the paper is organized as follows. Section 2 considers monolithic architecture, its benefits, and drawbacks. In Section 3, similarly, the paper closely evaluates microservices. Section 4 assesses these frameworks from a security perspective. Finally, Section 5 contains a conclusion of the paper.

2 Defining Monolithic Architecture

Traditional settings encountered in the software development life cycle (SDLC) limit engineers regarding the technologies, platforms, software, and hardware they select while creating a single application. Such applications are sophisticated and unified, consisting of four components: user interface, business logic, data access layer, and a database, as presented in Figure 1. Hence, it is often referred to as *monolith*, which Dragoni et al. [4] define as a program whose components are not modular. Yet, due to its prevalence, it is still the default and safe choice in software development. Debugging and deployment of monoliths is straightforward because they are contained in one executable directory. When an issue arises, it is considered to be a single point of failure. Although some might find this

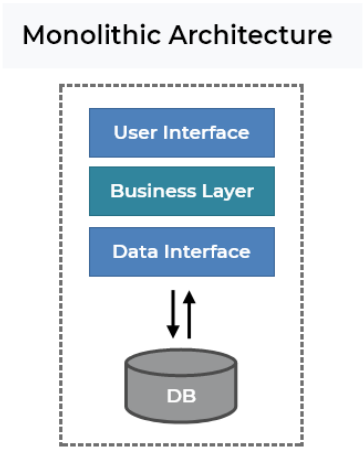


Figure 1. Monolithic Architecture

Source: Adapted from [3].

troublesome, it is in fact easier to discover and manage an error in code if it comes from a single source.

However, contemporary advancements have challenged the way we conceptualize computer programs. Thereupon, numerous faults in monolithic architecture have surfaced and prompted a discussion surrounding its efficiency and adaptability. As applications grow over time and more features are added, they become more complex and oftentimes even convoluted. This happens because adjustments need to be employed throughout the entire code base [12]. At this point, it is almost impossible to make a considerable alteration, for example, change the programming language or platform for application development. Likewise, teams who maintain the system expand and face complications as a result of working with the same files. Under those circumstances, it was inevitable that a new concept should be constituted, and microservices were developed with the intention to overcome the aforementioned pitfalls.

3 Defining Microservices

In 2011, at a small architectural workshop near Venice, a group of participants proposed a term, *microservices* [10], to describe a software architecture pattern. Although the phrase was new, similar architectural concepts have been present for some time in the field. Most notably, Netflix followed an approach labeled as *fine-grained service-oriented architecture (SOA)*. Indeed, microservices are founded on *SOA principles*. SOA

[13] refers to the segregation of systems into individual services, available on the network for other modules to use. Due to their resemblance, microservices are often recognized as a subclass of SOA. However, in a recent article by IBM [17], a clear distinction is made between these two approaches. It revolves around the scope: while SOA is concerned with enterprise scope, microservices are focused on more lightweight architecture concepts encountered in mundane applications.

Microservice architecture [15] decomposes older, monolithic systems into a suite of small and independent units connected to User Interface (UI), so-called *services*, as seen in Figure 2. Each service has a single purpose and communicates with other services through network calls, most often employing the HTTP protocol. They are *loosely coupled*, implying the least possible dependency on each other, which further promotes autonomy and ease in coordination. When designing a particular service, developers can opt for the best suited technologies and tools (for example, programming languages, database solutions and frameworks). Another benefit is the resilience of infrastructure since problematic modules can be confined and mitigated, while the rest of the application continues running smoothly. Similarly, this approach contributes to scaling efforts [5] in terms of size and scope, while not hindering efficiency and load balancing.

In spite of all the advantages, microservices are not suitable for every-

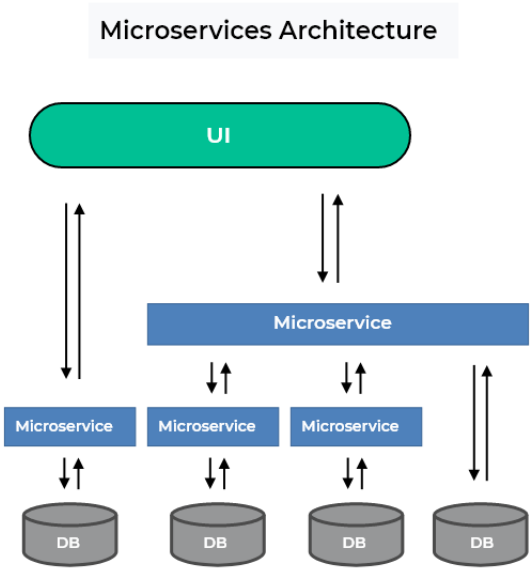


Figure 2. Microservices Architecture

Source: Adapted from [3].

one [8]. Obstacles mostly arise from inadequate implementation that can happen when the size of service is not predetermined, front-end integration is faulty, infrastructure is not monitored in its entirety, among others. However, this paper addresses an even greater concern that has captured the attention of cyber professionals as of recently. Due to the nature of distributed systems, intrinsic security vulnerabilities in microservices are omnipresent and are growing by the day.

4 Security Evaluation

With contemporary concepts and further improvements in the field of software architecture, unfamiliar threats emerge all the time. Peers are focused on new features, while applications are becoming vulnerable to attacks coming from all directions. As mentioned previously, some security issues are recognized because of SOA and other approaches related to distributed computing, but more problems were unraveled with recent implementations of microservices. The following subsections cover different aspects of security encountered in the above-mentioned architecture approaches.

4.1 Authentication and Authorization of Users

When creating a secure environment for any distributed system, it is of paramount importance to establish an all-encompassing authentication and authorization process. *Authentication* validates someone's identity, while *authorization* refers to a mechanism for verifying whether that person has sufficient privileges to access resources. This is an easy task for applications following the principles of monolithic architecture [1] that has a centralized signing-in point. With many pre-existing frameworks and libraries, creating a login module is rather straightforward.

On the other hand, microservices are more troublesome because users need to sign into multiple services. An almost universal approach to this problem is a *single sign-on (SSO)* solution that would enable the utilization of a single user ID for log-in purposes. In particular, frequently used standards include Security Assertion Markup Language (SAML), OpenID Connect [1, 15] and Open Authorization (OAuth) [14]. SAML is considered to be quite perplexing and favored in more centralized systems, while other protocols have faced difficulties with the lack of identity providers

(i.e., official entities that manage digital identities). Yet, they are still preferred in the case of public-facing applications. Keeping this in mind, the decision on protocol selection should be made in line with the application and its properties.

4.2 Authentication and Authorization of Services

A monolithic approach creates a unified system whose components function in sync. Contrarily, services encountered in microservice architecture need to act as a whole. In this ecosystem, they constantly communicate with each other and there needs to be a mechanism in place for authenticating and authorizing between services themselves. If services are not repeatedly verified, they are susceptible to a *man-in-the-middle (MitM) attack* [18] where the perpetrator intercepts the conversation and impersonates services in both directions. A common way of combating this is by switching from HyperText Transfer Protocol (HTTP) to Hypertext Transfer Protocol Secure (HTTPS), however, this is does not offer enough protection [9]. It only contributes to higher security of communication channels between two hosts that are authenticated through certificates.

Since servers manage their own Secure Sockets Layer (SSL) certificates [11], there needs to be a procedure in place to regulate the certification of multiple machines. This can be achieved by employing the existing protocols used for user authentication and authorization, such as SAML and OAuth, but this raises the question of credential storage. Some opt for distributing X.509 identity certificates [15] to services which also appears to be arduous due to the complexities of certificate management. Another option includes signing HTTP requests with a unique hash-based messaging code (HMAC). Here, both sides involved in the conversation are familiar with HMAC and this serves as means for authentication, while providing an additional layer of protection against MitM attacks. Overall, it is a concept that is up to the interpretation of developers who implement it. This includes tackling the topic of allocation of shared secrets before the communication commences and the type of hash function used to generate HMAC. Furthermore, API keys have recently gained significant traction after being adopted by Google and Facebook. They identify and authorize the service that is initiating the conversation and are considered to be unambiguous in terms of implementation and maintenance.

Over time, another threat to authentication was uncovered - *the confused deputy problem* [18]. This attack can be executed in circumstances

where the perpetrator seizes control of at least one service. Additionally, the system must be configured not to check the identity of the service issuing a call and the adversary is able to dispatch malicious requests to other services. As a result, communication is being eavesdropped on, and, sometimes, new messages are implanted and existing ones are altered. Unfortunately, as of right now, the solution is not clear-cut and developers must rely on implicit trust between services.

4.3 Data Protection

Although authentication, authorization, and credential management are the main focus of security studies of microservice systems [16], another significant vulnerability is observed in so-called *data at rest*. Data at rest refers to information that is not actively processed by the system. Oftentimes, this data is stored in an unencrypted state and can be easily exploited by the perpetrators [15].

According to best practices, all data should be encrypted with well-known and acclaimed encryption schemes such as Advanced Encryption Standard (AES), Rivest-Shamir-Adleman (RSA) and Triple DES (Data Encryption Standard). These encryption methods are frequently tested and patched and are suitable for nearly all platforms. Since these algorithms require private keys, developers should also consider how and where these keys should be stored. Although this may point towards encrypting all data, that should not be the case because some information can be valuable for logging purposes [11] in case of cyber attacks. Hence, this is where the microservice architecture proves useful since it provides fine-grained segregation of services that can later be evaluated for further protection with encryption.

4.4 Defence Mechanisms

What differentiates microservices from the traditional monolithic application is the addition of another vulnerable component – network. Multiple entry points (i.e. IP addresses) can cause traffic overload leading to all kinds of security issues. Security threats can either emerge internally (caused by people who have developed or utilized them) or externally (caused by third parties). If not detected and alleviated in time through preventative measures, they are mitigated with reactive, defense mechanisms.

A way to implement security from the start is through customized network architecture. Depending on the number of services, their ownership, and risk level, the network can be segregated into smaller subunits. These segments are then easier to control and maintain. Due to increased exposure to network threats, it is recommended to implement actively maintain a firewall solution [15]. A firewall is a software that provides filtering of incoming and outgoing traffic. Its policies define the types of network packets that are allowed to enter the network, considering the details such as IP address, port and protocol. Then, monitoring solutions collect activity logs from systems and can be used as both a proactive and reactive measurement. They are specifically useful for controlling distributed systems since they are able to centralize logs from multiple services in a single application [2]. Monitoring should be implemented both at the gateway and service level in order to detect a wider range of possible cyber attacks including injection attack and reuse of the bearer token. Furthermore, intrusion detection systems (IDS) and intrusion prevention systems (IPS) audit the network and its hosts for malicious activity. In the case of IPS, it can also mitigate the uncovered threats, thus, providing another layer of security. These systems are universally acclaimed and recommended to have. Finally, regardless of other solutions, all applications and software should be regularly patched and updated. Outdated systems are highly susceptible to malware and unauthorized intrusions. Therefore, companies should invest in a comprehensive patch management system, implemented over all software and middleware solutions, ensuring they are up to date at all times.

5 Conclusion and Outlook

Microservices have given a great insight into the future capabilities of software systems. The notion of scalability allows them to grow by adding more services and tailoring them to personal needs. However, due to their novelty, there are many obscure aspects, including the security of this architectural approach. A study conducted by Hannousse et al. [7] revealed that until 2019, there were only 46 available research papers that examined the security of microservices. Furthermore, it affirmed that these papers were mostly concerned with threats that emerge internally, in contrast to a growing trend of cyber breaches.

This paper tackled aspects of microservices that are the most vulnera-

ble: the process of authentication and authorization of users and services, protection of data at rest, and defense mechanisms against internal and external threats. Although most of the hazards are familiar to applications based on a monolithic architecture, microservice architecture has generated a variety of new threats owing to features inherited from distributed systems. With further implementations and expansions of this concept, security practitioners should expect more and more problems to arise. To combat this, the proposed proactive and reactive measures should be widely adopted. Likewise, this field would greatly benefit from further research efforts focused on providing security to microservice systems during the initial planning, development, utilization, and maintenance of the system.

References

[1] A. Bánáti, E. Kail, K. Karoczkai, and M. Kozlovszky. *Authentication and authorization orchestrator for microservice-based software architectures*. 2018.

[2] R. Chandramouli. *Security Strategies for Microservices-based Application Systems*. 2019.

[3] H. Siebert Domareski. *Monolithic & Microservices Architecture*. [Online]. Available: https://henriquesd.medium.com/monolithic-microservices-architecture-239e8799d3e1, 2021.

[4] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L Safina. *Microservices: yesterday, today, and tomorrow*. 2016.

[5] N. Dragoni, I. Lanese, S. Larsen, M. Mazzara, R. Mustafin, and L. Safina. *Microservices: How To Make Your Application Scale*. 2017.

[6] M. Fowler. *MonolithFirst*. [Online]. Available: https://martinfowler.com/bliki/MonolithFirst.html, 2015.

[7] A. Hannousse and S. Yahiouche. *Securing microservices and microservice architectures: A systematic mapping study*. 2021.

[8] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov. *Microservices: The Journey So Far and Challenges Ahead*. 2018.

[9] K. Jander, L. Braubach, and A. Pokahr. *Defense-in-depth and Role Authentication for Microservice Systems*. 2018.

[10] J. Lewis and M. Fowler. *Microservices, a definition of this new architectural term*. [Online]. Available: https://martinfowler.com/articles/microservices.html, 2014.

[11] N. Mateus-Coelho, M. Cruz-Cunha, and L. Ferreira. *Security in Microservices Architectures*. 2021.

[12] F. Ponce Mella, Gastón G. Márquez, and H. Astudillo. *Migrating from monolithic architecture to microservices: A Rapid Review*. 2019.

[13] N. Naghmeh, I. Waidah, G. Imran, N. Behzad, B. Mahadi, and H. Ab Razak Bin Che. *Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation*. 2020.

[14] A. Nehme, V. Jesus, K. Mahbub, and A. Abdallah. *Securing Microservices*. 2019.

[15] S. Newman. *Building Microservices*. O'Reilly Media, Inc., 1st edition, 2015.

[16] A. Pereira-Vale, G. Márquez, H. Astudillo, and E. B. Fernandez. *Security Mechanisms Used in Microservices-Based Systems: A Systematic Mapping*. 2019.

[17] IBM Cloud Team and IBM Cloud. *SOA vs. Microservices: What's the Difference?* [Online]. Available: https://www.ibm.com/cloud/blog/soa-vs-microservices, 2021.

[18] T. Yarygina and A.H. Bagge. *Overcoming Security Challenges in Microservice Architectures*. 2018.

Asymmetric Multi-core Scheduling

John Wickström
john.wickstrom@aalto.fi

Tutor: Jaakko Harjuhahto

Abstract

The growing market for long-lasting, battery-powered devices, such as smartphones or the Internet of Things, created a demand for a new type of processor with the capability of dynamically transitioning between modes of operation. This new architecture increased the complexity of task scheduling by transforming previously independent properties, such as throughput and fairness, to now be mutually exclusive. Such limitations necessarily introduce an unprecedented blind spot for scheduling policies that may be used to circumvent intended behaviour, and thus warrants further exploration. In this review paper, we discuss the implications, limitations and subsequent problem formulation of asymmetric multi-core scheduling to establish a baseline. Afterward, we analyze a handful of practical examples of scheduling policies and transformation compilers with unique aspirations. By understanding these contrasting approaches, we gain a greater understanding of how this problem plays out in practice, and what the current state of research is.

KEYWORDS: asymmetric systems, multi-core processors, scheduling

1 Introduction

Scheduling problems are part and parcel of nearly all facets of modern life, but rarely is the degree of complexity as prolific as it is for modern operating systems. With millions of interwoven or independent tasks to consider at any given point in time, the state space quickly becomes infeasible to fully explore, thus making heuristic accuracy imperative [12]. While hardware manufacturers include some built-in safety measures to prevent cascading system-wide failures, the resolution to the scheduling problem is delegated to the operating system.

Over time, Central Processing Units (CPU) have evolved and become exponentially more powerful and complex with the capability of concurrent multitasking, parallel execution and arbitrary combinations of both [10]. However, the advent of battery-powered smart devices created a market for a CPU with the adaptive capability of switching between modes, such as energy conservation or high throughput. This architecture was realized by equipping a CPU with different types of cores that each specialized in their own niche, in contrast to standard processors that have multiple instances of the same core type [11]. However, with core asymmetry, the scheduling problem grows in complexity due to the inherent need to match specific tasks with specific cores to emulate a desired behaviour. To reduce the search space, scheduling algorithms commonly maximize one primary property and set strict limitations for others [12, 3]. By definition, limitations represent an algorithm's blind spot which makes them a candid target for entities who may want to circumvent normal behavior. This distinction implies that both positive and negative effects should be considered when evaluating the efficacy of a scheduling policy. The proliferation of Internet of Things-based fog and edge applications as well as the societal reliance on smartphones warrants further investigation into these aspects of scheduling.

This review paper is structured as follows. Section 2 starts with a brief overview of processors as well as establishing the implications of core symmetry (or lack thereof) with respect to scheduling. This section also details how our collective understanding of Asymmetric Multi-core Processors (AMP) has evolved, and how conventional computational assumptions may not apply to them. Section 3 starts by defining operating system-level scheduling for AMPs through its problem formulation and limitations. Afterward, we review some state-of-the-art scheduling poli-

cies and transformation compilers that have been designed for a particular purpose. Section 4 continues with a contrasting discussion regarding the implications of policy unfairness. Finally, section 5 concludes this review-paper with a summary.

2 Background

The processor is generally considered to be the brain of a machine, as it is the component that coordinates the order of operations on an exceptionally low level. To formulate *how* machine code is handled, we use an abstract model called an Instruction Set Architecture (ISA). An operating-system capable processor, such as a Central Processing Unit (CPU), is a typical example of an active enforcer of this ISA, in contrast to a Graphics Processing Unit (GPU) which is not.

Historically, processors only contained one "*core*" which limited them to working on a single process at any point in time. However, in 2001, IBM released the first processor with two separate cores, meaning that it was capable of working on two processes in parallel [10]. The number and complexity of processor cores have kept increasing ever since, to which operating systems and programs have adapted and consequently become more complex.

2.1 Core Symmetry

Multi-core processors were initially homogeneous, meaning that each core was identical, and tasks could therefore be scheduled to the first available core without any greater consideration. However, this locked processors into a single *modus operandi* that was designed for devices that had a constant source of power, which was less than ideal for the growing market of battery-powered handheld devices. Thus, the notion of processors with a heterogeneous core setup was suggested to give these devices an on-demand capability to transition between modes. Today the former type is known as Symmetric Multi-core Processors (SMP) and the latter as Asymmetric Multi-core Processors (AMP).

One prominent AMP breakthrough was the *ARM big.LITTLE* [11] processor that was designed for portable handheld devices in the early 2010s. It came equipped with slower, energy-efficient cores (*LITTLE*) as well as fast, energy-hungry cores (*big*) that nevertheless abide by the same ISA.

The implications of these distinctions are imperative to understand as they fundamentally change the dynamics of scheduling by making core selection non-trivial [21, 20, 17]. In practice, a well-designed AMP scheduling policy allows a device to adapt to its current circumstances by balancing or maximizing properties such as throughput, energy conservation or fairness.

2.2 Heuristics and Inference

While historically cited computational theories such as *Amdahl's Law* [1] (AL) and *Gustafson's Law* [7] (GL) still roughly represent a correct trend for SMPs, numerous works have proven that they overlook many nuanced properties of AMPs. For example, Eyerman et al. [5] created a scheduling policy that assigns memory-bound tasks to *LITTLE* cores, and CPU-bound tasks to *big* cores, as this methodology minimizes the penalty for cache misses. Although this premise is central to both of the aforementioned laws, Van Craeynest et al. [18] found that the approach was too naive and extended it with a heuristic estimator. Eyerman and Eeckhout [4] present a version of AL that incorporates the notion of critical sections and their synchronization to the formula. Juurlink and Meenderinck [9] propose a hybrid equation that assumes that the truth lies somewhere in-between AL and GL.

The use of these laws is primarily to evaluate the general complexity of the architecture rather than to obtain a truthful and precise reading of a scheduling policy. To accomplish that, highly sophisticated parsing tools, such as QuickIA [2], are used to retroactively explore large state spaces with the intent of gaining insights through static analysis.

3 Operating System-level Scheduling

Hardware components, such as the processor and its ISA, have a physical limitation that dictates their upper boundary of performance. No scheduling policy can raise this threshold, and the goal is merely to come as close to the upper boundary as possible by intelligently sorting and executing processes. Changdae et al. [12] emphasize that a clever and efficient scheduling policy is what elevates the diverse potential of AMPs above SMPs, whilst a bad policy does the inverse. Due to how nuanced the problem of arranging millions of individual tasks is in real-time, these

scheduling strategies are delegated to the operating system rather than being inherited from the hardware and enforced by manufacturers.

3.1 Problem Formulation

The decision state space of an AMP scheduler can be both enormous and non-convex [17]. Thus, the continuous traversal of said space is unfeasible for the purpose of computing an optimal solution quickly. On the other hand, an exhaustive search *can* be performed retroactively to find a lower bound of performance to strive for [2]. The complexity of this problem makes the ideal lower bound effectively impossible to realistically reach, so solutions are generally compared to each other rather than a hypothetical.

While a primitive queue, such as *FIFO*, can be used to obtain a trivial solution to this problem, heuristic estimation allows us to produce a specialized solution even on large-scale systems. Naturally, primitive queues seldom schedule tasks in an optimal order, however, they can still be competitive *exactly* because time is not wasted on computing an order. Specialized scheduling algorithms need a brief moment to consider the options before assembling an order that results in a shorter cumulative turnaround time on average, and therefore an overall superior policy. Complexity is a point of contention because it can prevent an algorithm from functioning at scale while still being perfectly suitable for small systems. To circumvent this problem, contributions such as [17, 15] present both greedy and non-greedy versions of their algorithms intended for different systems.

3.2 Policy Limitations

Scheduling policies for AMPs generally prioritize one of three properties: *Throughput*, *Fairness* or *Energy conservation*. Increasing throughput necessarily increases a device's energy consumption and vice versa. Perhaps due to how central these two properties are for the *big.LITTLE* core paradigm, there exists an abundance of research that addresses the balance between them. The dynamic transition from prioritizing one to the other is crucial for any modern battery-powered smart device [21].

For SMPs with homogeneous cores, fairness and throughput are independent problems and can therefore be maximized simultaneously. However, for AMPs, there is a drastic performance difference between cores,

therefore absolute fairness and maximum performance become two mutually exclusive properties [12]. Fairness is a particularly deceptive problem because the lack of it, meaning the unequal treatment of processes, opens up the possibility of *Starvation* [3]. For example, if a process is not guaranteed its timeshare on the processor within a finite amount of time, there exists a hypothetical scenario where this process awaits its turn infinitely. The probabilistic nature of this problem makes it relatively abstract and difficult to evaluate in comparison to throughput or energy efficiency.

3.3 Scheduling Policies

Tsai et al. [17] present *AMS*, an adaptive scheduler for parallel processing with an awareness of both cache hierarchies as well as core types. The authors prove that an application's initial preference toward a deep or shallow cache hierarchy is often temporary. By re-mapping threads periodically (50 ms) according to a cache-specific miss curve, their algorithm can quickly adapt to a program's new phase and schedule it appropriately. *AMS* has two optimization alternatives, greedy and non-greedy. The non-greedy optimizer leverages dynamic programming to efficiently explore a large state space by recursively solving and memoizing smaller and smaller sub-problems. While the greedy optimizer is not as thorough, it performs within 1% of the non-greedy optimizer by utilizing established cache partitioning techniques and consequently has a smaller overhead. With respect to average throughput, *AMS* performs 18% better than asymmetry unaware schedulers, such as Linux CFS [19], and 6% better than other similar previous works, such as PIE [18].

Yu et al. [20] present *COLAB*, a collaborative scheduler for multi-threaded processes that utilizes three separate heuristics: thread criticality, core sensitivity and fairness. The performance of multi-threaded processes is often tied to critical sections that manipulate a shared resource. Thus, the authors argue that an optimal scheduling policy needs to accelerate threads that handle these segments of a process first, as they are the primary bottleneck. Raw metrics, such as thread speedup or blocking factors, are obtained from the Linux kernel with low overhead, and are used on a pre-trained linear regression model to predict an outcome. Threads with high prediction scores are prioritized for *big* cores, whilst low predictions are prioritized for *LITTLE* cores. The remainder of threads can be scheduled for either core type, so the decision is left for a load balancer. Evaluations show that *COLAB* produces a 10% reduction in process turnaround

time on average, compared to the Linux CFS [19] and ARM GTS [8].

3.4 Transformation Compilers

Sreelatha et al. [16] present *CHOAMP*, a compiler that probabilistically optimizes and transforms SMP software to run efficiently on AMPs, with respect to performance profiles. Contrary to most other previous works, the authors don't intend to reinvent the wheel and instead leverage the vast knowledge of SMP scheduling to achieve their goal. The compiler analyzes programs written in *OpenMP* pragma notation, which is a popular C/C++ extension that greatly reduces the complexity of creating software intended for parallel processing. The output of this analytical process is first optimized for an SMP before finally being transformed to run efficiently on an AMP. While the experimental evaluations of *CHOAMP* look promising, its value is further amplified by its relative simplicity as well as the ease in which it can be realistically adopted.

Krishna et al. [14] conducted experiments using graph algorithms that frequently entail workloads with irregular parallelism, in contrast to Sreelatha et al. [16] who only explored patterns of regular parallelism. The primary concern for irregular workloads is to identify and schedule large workloads to *big* cores in order to achieve a shorter cumulative turnaround time. However, the authors take this one step further by transforming the input graph to maximize cache utilization, as well as the subsequent *OpenMP* program to gain a longer than average timeshare on the *big* cores. The effects of the transformation are decided by a two-step prediction by a multi-variable regression model which takes graph features and hardware configurations as input and locates an optimal solution in accordance with a knowledge base. Results show a near 43% improvement in the energy-delay-product $(duration \times energy)$ compared to unoptimized base-cases.

4 Discussion

Since the schedulers' job is integral for enforcing a certain behaviour, its decisions are intentionally abstracted away from programs running on a machine. Tsai et al. [17] argue that it would be unreasonable, perhaps even dangerous, to let programmers affect this decision-making process directly. However, this does not mean that programmers cannot af-

fect it indirectly by exploiting the limitations of the underlying algorithm through software design.

Many prior works [13, 6, 12] detail how *Linaro*, the Linux kernel scheduler, generally does a good job regarding fairness through its *Completely Fair Scheduler* [19] (CFS) policy, but it lacks the granularity to enforce this on an AMP thread level. In fact, one clock tick on the *big* core is considered equivalent to one clock tick on the *LITTLE* core [6], which obviously cannot equate to the same amount of progression for a process. This inequality opens the possibility for a program to get preferential treatment by exploiting some arbitrary pattern that the scheduler frequently assigns to the big core. Such a program would be unfair by definition, as it would spend a higher percentage of time on the big core compared to other programs [14].

5 Conclusion

In summary, task scheduling for processors with asymmetric cores presents a very complex but necessary problem to solve. In the case studies, we observed monumental improvements in metrics such as energy efficiency, which indicates that we are far from reaching a convergence and there are plenty of strides to be made. The concept of fairness remains problematic due to its constraining effect on throughput and its ambiguous nature which makes it difficult to measure.

While this scheduling research is gaining more traction as of late, it still does not seem even remotely proportional to its potential impact on prominent industries, such as *Big Data* or *Distributed Computation*. A slight increase in throughput or energy efficiency via smart scheduling may seem like a ripple in the ocean when applied to a singular device. However, when applied on a scale that is equivalent to the *Internet of Things*, a ripple turns into a tidal wave. Since scheduling policies are intrinsically tied to the operating system of these devices, the existing infrastructure already supports massive scale distribution of such improvements via firmware updates. Regardless, there is ample room for research of various levels of granularity as well as for general-purpose tools that allow us to explore and analyze the vast search space of schedulers more efficiently.

References

[1] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), page 483–485, New York, NY, USA, 1967. Association for Computing Machinery.

[2] Nagabhushan Chitlur, Ganapati Srinivasa, Scott Hahn, P K Gupta, Dheeraj Reddy, David Koufaty, Paul Brett, Abirami Prabhakaran, Li Zhao, Nelson Ijih, Suchit Subhaschandra, Sabina Grover, Xiaowei Jiang, and Ravi Iyer. Quickia: Exploring heterogeneous architectures on real prototypes. In *IEEE International Symposium on High-Performance Comp Architecture*, pages 1–8, 2012.

[3] Eiman Ebrahimi, Chang Joo Lee, Onur Mutlu, and Yale N. Patt. Fairness via source throttling: A configurable and high-performance fairness substrate for multicore memory systems. *ACM Trans. Comput. Syst.*, 30(2), apr 2012.

[4] Stijn Eyerman and Lieven Eeckhout. Modeling critical sections in amdahl's law and its implications for multicore design. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, page 362–370, New York, NY, USA, 2010. Association for Computing Machinery.

[5] Stijn Eyerman, Lieven Eeckhout, Tejas Karkhanis, and James E. Smith. A mechanistic performance model for superscalar out-of-order processors. *ACM Trans. Comput. Syst.*, 27(2), may 2009.

[6] Adrian Garcia-Garcia, Juan Carlos Saez, and Manuel Prieto-Matias. Contention-aware fair scheduling for asymmetric single-isa multicore systems. *IEEE Transactions on Computers*, 67(12):1703–1719, 2018.

[7] John L. Gustafson. Reevaluating amdahl's law. *Commun. ACM*, 31(5):532–533, may 1988.

[8] Brian Jeff. big. little technology moves towards fully heterogeneous global task scheduling. *ARM white paper*, 2013.

[9] B.H.H. Juurlink and C. H. Meenderinck. Amdahl's law for predicting the future of multicores considered harmful. 40(2):1–9, may 2012.

[10] Ronald N. Kalla, Balaram Sinharoy, and Joel M. Tendler. Ibm power5 chip: a dual-core multithreaded processor. *IEEE Micro*, 24:40–47, 2004.

[11] Shubham Kamdar and Neha Kamdar. big. little architecture: Heterogeneous multicore processing. *International Journal of Computer Applications*, 119:35–38, 2015.

[12] Changdae Kim and Jaehyuk Huh. Exploring the design space of fair scheduling supports for asymmetric multicore systems. *IEEE Transactions on Computers*, 67(8):1136–1152, 2018.

[13] Myungsun Kim, Soonhyun Noh, Sungju Huh, and Seongsoo Hong. Fair-share scheduling for performance-asymmetric multicore architecture via scaled virtual runtime. In *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 60–69, 2015.

[14] Jyothi V.S. Krishna and Rupesh Nasre. Optimizing graph algorithms in asymmetric multicore processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2673–2684, 2018.

[15] Lei Mo, Angeliki Kritikakou, and Olivier Sentieys. Approximation-aware task deployment on asymmetric multicore processors. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1513–1518, 2019.

[16] Jyothi Krishna Viswakaran Sreelatha, Shankar Balachandran, and Rupesh Nasre. Choamp: Cost based hardware optimization for asymmetric multicore processors. *IEEE Transactions on Multi-Scale Computing Systems*, 4(2):163–176, 2018.

[17] Po-An Tsai, Changping Chen, and Daniel Sanchez. Adaptive scheduling for systems with asymmetric memory hierarchies. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 641–654, 2018.

[18] Kenzo Van Craeynest, Aamer Jaleel, Lieven Eeckhout, Paolo Narvaez, and Joel Emer. Scheduling heterogeneous multi-cores through performance impact estimation (pie). In *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, pages 213–224, 2012.

[19] C.S. Wong, I.K.T. Tan, R.D. Kumari, J.W. Lam, and W. Fun. Fairness and interactive performance of o(1) and cfs linux kernel schedulers. In *2008 International Symposium on Information Technology*, volume 4, pages 1–8, 2008.

[20] Teng Yu, Runxin Zhong, Vladimir Janjic, Pavlos Petoumenos, Jidong Zhai, Hugh Leather, and John Thomson. Collaborative heterogeneity-aware os scheduler for asymmetric multicore processors. *IEEE Transactions on Parallel and Distributed Systems*, 32(5):1224–1237, 2021.

[21] Yuhao Zhu and Vijay Janapa Reddi. High-performance and energy-efficient mobile web browsing on big/little systems. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 13–24, 2013.

Intelligent Character Animation with Deep Reinforcement Learning

Daniel Zsemberi
daniel.zsemberi@aalto.fi

Tutor: Nam Hee Kim

Abstract

Virtual character animation is crucial in the entertainment industry, such as virtual reality, computer games, and animated movies. Physics-based techniques provide an excellent framework to produce realistic and physically accurate animations. To this end, deep neural networks through reinforcement learning (RL) is one of the go-to methods to drive recent advances. This survey puts forward a study of recent approaches to physics-based character animation. First, physics-based animation and RL are introduced. Second, state-of-the-art animation methods with and without motion capture data are discussed. Finally, the limitations and possible future work are reviewed.

KEYWORDS: Deep reinforcement learning, Physics-based character animation

1 Introduction

Various challenges in computer graphics involve characters and objects engaging with each other or their environment. Producing these animations to be true to life and responsive is difficult for human experts since the number of possible interactions is immense.

Programmatically, character animation can be solved by two main approaches: kinematics-based and physics-based animation frameworks. The objects are moved according to a set of given velocity and acceleration equations in the former. In the latter, the characters automatically interact in a physically accurate manner by utilizing a physics simulator. Physics-based methods may automate work in the current animation pipelines. However, this method has several challenges. The first problem is the lack of user control due to physical forces being the primary control method for characters. Second, although physically accurate, the animation style might appear robotic or stiff. Finally, it is expensive computationally.

This paper aims to review the deep reinforcement learning approach to solving the above problems in physics-based character animation.

This survey is constructed in the following manner. Section 2 introduces computer and physics-based animation. Section 3 dives deeper into deep reinforcement learning (DRL) intricacies, Section 4 describes data-driven DRL animation techniques, and Section 5 discusses the data-free versions. Section 6 discusses the current state of physics-based animation and provides concluding remarks.

2 Character animation

Character animation is the process of generating digitally animated images where digital characters are moved and rendered to create a motion picture. Presently, digital characters require control by human experts to make their movements believable, which is a time-consuming process that various computational techniques, such as kinematics-based and physics-based animation methods, have sped up. This section explores where physics-based animation originated and how it can be formalized.

2.1 History of computer animation

Hand-drawn animation has evolved immensely from the late 1890s to the early 1990s, including feats like the 12 principles of animation [26], Technicolor animation, Disney's well-known characters, and anime from Japan. This period also included some partly animated movies, e.g., Tron, Jurassic Park, and Terminator. Eventually, with the success of Toy Story (1995) and Shrek (2001), computer animation started dominating the field of commercial animation.

2.2 Animation research history

Computer animation research gained traction in the 1960s: Nikolai simulated and rendered a walking cat [30], and Special Interest Committee on Computer Graphics (SICGRAPH), now known as ACM SIGGRAPH, was established (1967). In the 1980s, research towards producing animation via simulated characters was on the rise with kinematic constraints and reverse kinematics [11] [7], dynamic analysis using torques and forces [29], and attempts to make these more lifelike with optimization [31]. In the late 1990s, neural networks were introduced to replace cost-heavy simulations of physics [8]. By the late 2010s, the spread of deep neural networks (DNN) also influenced animation by, e.g., learning low-level representations of motion data using autoencoders [10] or unsupervised learning from unstructured quadruped data [35]. Furthermore, during this period, deep reinforcement learning (DRL) was also employed to generate character animation by, e.g., using references to mimic motions and learning to recover [21] and utilizing hierarchical DNN policy that learns physics-based locomotion skills [22].

2.3 Physics-based animation

Physics-based animation is a highly multidisciplinary field based on physics, mathematics, and engineering. Three components are necessary for physics-based character animation: a physics-based character (robotic creature), a physics simulator, and a control policy that controls the character. This section introduces these three in detail.

Physics simulator

Standard physics simulators include soft-body simulators, cloth/hair simulators, and the one detailed in this survey: constrained rigid body simulators. "Rigid" refers to objects being non-elastic. A rigid object has geometry, mass, a center of mass, position, orientation, linear velocity, and angular velocity. "Constrained" denotes that the object's parts, called links, are connected through joints. A joint might have several degrees of freedom (DoF) - one, two, or three - both in rotation and translation.

Forward dynamics is the process of calculating the acceleration of a rigid-body system given the applied forces and torques. This topic is beyond the context of this paper, and the reader may refer to [4] for more information.

Animated character

Building an animated character means using links and joints to model a real-life creature or build a fictional one. In the former case, the joints need to be carefully chosen to reflect real-life limitations. For instance, hip joints often use ball-and-socket joints.

The physics simulator requires torques and forces to move the character, while the actuator is used to generate those torques and forces. Most applications use internal forces to control the character, but external forces may be used for additional support such as balance skills [32] as well. However, real-life robots cannot leverage external forces. Furthermore, some papers have also shown interest in muscle-based actuators where muscles and tendons are simulated to apply forces and torques to physically-based characters [6] [27] [17].

Control policy

A control policy generates actuation patterns that help the character achieve its goals in the same vein as how animal brains govern motor skills. For example, goals might include reaching a coordinate or matching a reference motion.

According to robotics literature, there are two main types of controllers: closed-loop controllers, which employ the current state of the environment as input, and open-loop controllers, which optimize over an entire control trajectory. The focus of this paper is closed-loop controllers, which model-free DRL algorithms mainly employ.

A control policy might also expose high-level control parameters, such as a label of the current task or desired character speed. Exposure of these parameters is a popular technique to enable user control during runtime.

3 Deep reinforcement learning

Deep reinforcement learning (DRL) blends reinforcement learning (RL) and deep learning. RL was inspired by behavioral psychology [25] and solves the task of sequential decision-making: in an uncertain environment consisting of states $s_1, s_2, ...s_n \in \mathcal{S}$, an agent performs predefined actions $a_1, a_2... \in \mathcal{A}$ according to a policy $\pi : \mathcal{S} \Rightarrow \mathcal{A}$. These actions move the agent from state to state until it terminates or reaches its goal. Meanwhile, the environment provides a reward or punishment $r_1, r_2, ... \in \mathbb{R}$ based on the quality of the action at each timestep according to some cri-

teria designed by an expert. The agent's goal is to maximize the received reward over its trajectory of states and actions.

Using RL to automate laborious animation tasks is a promising method over others. The structure of RL provides an abstraction that can fit task animation naturally. RL agents use trial-and-error to explore the action space and have dynamic interactions with the environment, making its policy more sound to previously unseen states.

Deep neural networks were introduced to RL tasks to establish the policy function π as a learnable function. Usually, the policy function is parameterized as $\pi_\theta(a|s)$ and it outputs the probability density function from which actions will be sampled. These techniques can address high dimensional-space problems like playing Atari games [15], mastering Go [24] or even autonomous driving [18]. For a more thorough overview, refer to [5].

3.1 RL and physics-based animation

This section discusses how RL and physics-based animation are connected and the crucial considerations one must take while designing an RL algorithm: state representation, action space concerns, and reward engineering.

Actions

In physics-based character animation, the actions describe how a character may move. Due to the characters being physics-based, the actions must be incorporated into the simulated physical system, which means that the actions might be actual physical intrinsic forces or, more commonly, some target orientation for the joints. In the latter case, the physics engine takes care of moving the joints: the engine actuates (moves) the joints based on the desired joint torques, typically utilizing a position-derivative controller method. A more complicated but also more life-like character can be designed with virtual muscles that have been modeled as part of the physics-based character [28]. Muscle simulations, however, instill a more complicated action space.

States

In the case of animation, the state commonly describes the configuration of the character's body. Essential features that describe a character include root link orientation, joint angles, and joint velocities.

The state might also include other values that help interact with the environment, such as whether any body part is touching the ground. A phase variable $\in [-1, 1]$ using a sinusoid as a function of timesteps might also be added to describe the character's state in the walking cycle. This phase variable helps the agent perceive the relative place in the walking cycle more explicitly.

The state might also include information about the environment surrounding the agent, such as height maps, the position of the footsteps, or the pose of other characters. These added variables help the agent in perceiving the world.

Reward function

Reward engineering, meaning designing the reward function for a specific task, is also a substantial part of DRL for physics-based character animation. Successful and efficient performance of DRL algorithms depends on the clever choice of reward function to negate the effects of task difficulty and high dimensionality. However, reward engineering is a long, precise, and tedious process since developing a mathematical form of some task's desired goal is challenging and laborious. Reward engineering also limits one agent to one specific task, making the eventual goal of a general agent much harder to reach.

Reward functions are frequently broken into several terms that help specify the task or the goal. Furthermore, regularization terms might also be added to help the character, e.g., produce weaker movements or walk symmetrically.

4 Data-driven DRL for character animation

In contrast to data-free animation methods, data-driven animation utilizes reference motion either during the pre-training or the training phase of learning, making use of imitation learning to mimic the motions. This has the advantage of producing more life-like results, requiring less training time and simplifying the training. The disadvantages include needing to collect motion data and being restrained to life-like characters. This section introduces the latest advances in data-driven DRL animation, starting with unstructured motion data and moving on to methods based on Generative Adversarial Networks.

Unstructured motion data

Converting kinematic motion capture clips containing no information regarding forces and collision to a physically-based character has been a longstanding goal in character animation. The aim is to make the character execute the same motion while also withstanding perturbation. Peng et al. [21] aims to solve this exact problem with DeepMimic. It incorporates a reward function where a step-wise reward is given when the agent's action results in a character state close to the corresponding frame's reference state. This reward is then used to train a separate policy for each specific motion, forming a composite policy together. The resulting composite policy can perform a broad range of challenging skills like spin-kick and backflip. Nevertheless, the training requires a significant amount of time and extensive manual tuning, even for short clips, and only demonstrates a few clips in the composite policy. Chentanez et al. [3] builds upon DeepMimic [21] to address the limitation in terms of number of clips that may be used. The authors train a character on a massive dataset of animations by separating the character into two brains: a tracking brain and a recovery brain. The two brains are trained separately with the aim of activating only one of them at a time, therefore separating tracking and recovery skills into specific expert controllers. The resulting policy learned more than 1300 motions, but the resulting motions were not fully physically accurate.

DeepMimic [21] lacks user interaction during the runtime of the algorithm. DReCon by Bergaming et al. [2] builds upon DeepMimic and tries to solve this issue by training an additional feedback policy that reacts to perturbations due to user input. The resulting policy is robust and responsive with humanoid characters on flat terrains but is only capable of running and walking. Park et al. [19], concurrently to DReCon, has shown an interactive character that is capable of not only locomotion but jumping and rolling as well. Two networks are trained separately for the kinematic and dynamic aspects of the motion data. The resulting controllers are capable of a broad range of tasks, but in the presence of perturbation, the characters become stiff, and the networks still require multiple days to train.

GAN-based

To be effective, data-based methods using large datasets require carefully designed objective functions and a separate component that selects the

appropriate motion for the character. Luo et al. [13] utilizes the idea of Generative Adversarial Imitation Learning (GAIL) to combat these. A separate neural network (discriminator) is trained to distinguish generated performances from expert demonstrations with supervised learning. The DRL policy of the character is then encouraged to generate performances that deceive the discriminator. This rids the algorithm of the need for an objective function for learning how to respond to user input during runtime. However, the agent is trained two more times: with DRL for perturbation recovery and with imitation learning for an initial policy. This results in long learning times and adds reward engineering, one of the issues GAIL tries to eliminate. To this end, Peng et al. [20] introduces AMP, an algorithm that removes the need for multiple training passes. AMP uses two components in the reward function: one based on GAIL for low-level control (joint movement) and one engineered by an expert for high-level objectives. The results show the capabilities of combining various motion data, even if the combination was not present in the training dataset. The combinations may be quite extreme as well, such as zombie walking combined with soccer dribbling.

5 Data-free deep reinforcement learning animation methods

While traditional and data-driven DRL methods involve using some kind of reference motion to aid the animator, data-free DRL algorithms utilize no reference motion for learning purposes. Unfortunately, this approach is known to cause remarkably unnatural movements and therefore poses a considerable challenge. Consequently, other types of assistance have been employed to ease this pain point, such as reward engineering or handcrafted goals for the agent (e.g., pre-defined touchpoints). This section examines a few of the latest methods in data-free DRL for physics-based animation: TO-based and curriculum-based.

TO-based

Trajectory optimization (TO) [31] can be used for physically-based character animations. TO finds a numerical solution to a large user-constrained optimization problem. Solving a TO problem results in a motion trajectory and a trajectory that controls the character (e.g., forces), therefore essentially simulating the movement of a character. However, a detailed explanation of the underlying mathematics is out of the scope of this pa-

per.

TO may also be employed to constrain characters such that their movements are physically accurate. For example, joints should not bend too much, and the movements should not be too convulsive. These constraints are often called spacetime constraints. Spacetime constraints bound the components of the character in state space and time, e.g., touching the ball with the hand at the peak of the ball's movement.

The output trajectories of TO can be employed to augment data-free learning methods with a synthetic database. Mordatch et al. [16] successfully parameterized TO as a learned generative process with a neural network. However, well-known problems with TO (gradient discontinuity, contact constraints, slow compute) prevented its wide adoption, and the expressivity was limited compared to deep RL methods that followed suit. Liu et al. [12] takes this idea further to simulate ball movement for basketball dribbling and use it as an input for a DRL algorithm. This methodology rids the practitioners of the cumbersome process of recording motion capture data for the ball. Furthermore, the developed methodology addresses the problem of learning only from TO motions by using TO as only a sub-component of their system. The resulting agent is capable of complicated and realistic-looking dribbling skills. The above papers show that TO can be a valuable tool for producing a synthetic animation that an advanced algorithm may use for motion imitation, but additional input data might be required for more realistic visuals.

Reward engineering is a long, precise, and tedious process in state-of-the-art physics-based animation work since specific tasks require special reward functions. Ma et al. [14] addresses this by proposing an objective function that employs spacetime constraints which leads to a more intuitive view of reward functions. For example, to make a character learn to jump, the algorithm is the instructed to have both feet of the character in the air (space constraint) for some time (time constraint) and nothing else touching the ground (space constraint). This work substantially streamlined the reward design process and showed a robust way to learn motion skills without motion capture data.

Curriculum-based

Learning complicated tasks is a challenging problem for RL agents due to the extreme size of the state-action space. Exploring this space naively is naturally time-consuming and ineffective. The core idea of curriculum-

based reinforcement learning (CB-RL) is to increase the task difficulty gradually throughout the learning process. Put differently, the agent starts with a small optimization landscape, which is extended to a more demanding one as the agent acquires experience. As a result, CB-RL facilitates faster and more effective learning. In this section, recent developments in CB-RL for physics-based character animation are discussed.

Long learning times are frequent in DRL due to the complexity of the task. Babadi et al. [1] proposes a unique CB method called Termination Curriculum (TC) to attempt to resolve the issues of learning efficiency. TC forces the agent to terminate when the instantaneous reward is below the hyper-parameter threshold, which is lowered throughout the learning process. The results show that TC decreases sample complexity and training time. However, the agent is only capable of flat terrain movement. The ALLSTEPS algorithm presented by Xie et al. [33] makes the terrain uneven, meaning that it solves the stepping stone and continuous terrain problems in addition to flat-terrain locomotion. The algorithm uses no reference motion but relies on CB-RL and heavy reward engineering. The resulting policies are robust and capable of walking the varying terrains but would require more stylistic rewards for natural-looking hand movements.

Heess et al. [9] poses the challenge of using only simple rewards, less reward engineering, and no reference motion to learn locomotion. Therefore, the authors employ a simple reward function, consisting of only a few generic components (e.g., speed, deviation from the center of path). The resulting policy demonstrates that an elementary reward function might be adequate for producing a physically-based character that is proficient at walking. However, the agent had overly exaggerated movements with high oscillations in actuations, exposing challenges in the regularization of character strengths and action magnitudes. Yu et al. [34] used the same notion of basic reward but used a virtual assistant that is capable of applying assistive forces to the character, which helped the agent learn balance and forward movement, therefore making the motion more realistic. The amount of assistive forces was steadily reduced as the agent gained experience (CB-RL), which encouraged the agent to discover optimal motions in a data-efficient manner. The resulting motions are relatively natural. However, the quality was still not on par with data-based methods.

Mimicking characters with a form not present in real-life creatures (e.g., 12 legs and five heads) is an issue for data-based algorithms. However, it

is instrumental in robotics, where robots may take any shape or form. Rodrigez et al. [23] present an omnidirectional walking controller for robots that uses CB-RL to control a real-life robot. The agent is encouraged to reach a given velocity, and throughout the training process, the desired velocity is increased. This makes learning easier since slower walking requires less balance. The resulting policy is demonstrated on a real-life humanoid robot but is easily extendable to other archetypes.

6 Discussion

In this paper, we explored some of the latest techniques in physics-based character animation with RL. These included TO-based, curriculum-based, DeepMimic-like, and GAN-based approaches. In this section, some of the the future paths and limitations for the field are discussed.

First, heuristics in reward engineering such as energy minimization, symmetry, and stability may improve realism, but these are not widely applicable to all animations and require clever expert tuning and trial-and-error, which are time-consuming. Data-driven approaches attempt to resolve this issue by using reference motion. Nevertheless, extending these methods to leverage a large variety of motions is still challenging, as these methods cannot express the motion distribution in the desired range. This is partly due to the deficit of non-laboratory motion data. Current motion capture methods exhibit hardware restrictions, as they necessitate hardware to be installed on real-life characters with a multi-camera arrangement. In the future, markerless data might aid in collecting varied motion data. By combining in-the-wild and controlled motion data, one may design a curriculum-based algorithm where controlled motion data is employed first, and then a diverse range of in-the-wild data is utilized to make the model more data-efficient and robust.

Second, the regularization of naturalness as perceived by humans is another challenge. Reward engineering is presently unfit to capture this because of the lack of its quantifiable measure. Currently, regularizations such as energy minimization or symmetry are used. However, these do not capture the fundamental nature of human motion. This challenge is yet to be tackled and prevents data-free methods from reaching their full potential. One approach to solving this would be to employ synthetic data generation and motion matching - as seen with some methods - but the quality of this data for complex motions is still lacking. Human in-

teraction during training could be harnessed to provide keyframes that could be used as sparse data for motion matching. Furthermore, dense motion data from keyframes provided by humans could be generated with in-betweening algorithms and used in imitation learning.

Third, motion editing is yet to be scrutinized comprehensively despite the topic being essential to enable animators to utilize these novel techniques. One interesting approach could be for neural networks to learn low-level latent motion parameters which human experts can later control. However, this requires both interpretability and controllability. The former refers to establishing an association between the physical control and latent representation. The latter refers to direct handle of animations with some latent model.

Fourth, the interactivity of controllers, meaning models dynamically responding to user inputs while maintaining realistic and physics-based movement, is yet to be explored thoroughly. As of today, controllers can manage complicated motions such as jumping, cartwheeling, or pushing objects. However, in my opinion, these controllers often fail to capture human naturalness. The techniques close to achieving organic human motion are data-based ones. However, these are still too unpredictable and low-quality compared to traditional real-world animation pipelines and thus not well-utilized in practice. I would expect advancements in character animation to growingly depend on data-driven approaches in the future.

Fifth, training times for DRL algorithms are long, and researchers gain feedback on their training days after starting it. This could be resolved by harnessing the power of GPUs. Recent methods have started utilizing the GPU for RL (GA3C), and future methods running fully on GPU without any interaction of the CPU ought to speed up the process.

To conclude this survey, character animation is a fruitful field of DRL and will certainly grant DRL the ability to solve movement problems in the real world, especially in the entertainment and robot industry. It is an actively researched field with very fast progress, therefore future work will resolve limitations listed in this survey rapidly.

References

Babadi. Self-imitation learning of locomotion movements through termination curriculum. 2019. doi: 10.1145/3359566.3360072. URL https://doi.org/10.1145/3359566.3360072.

et al. Bergamin. Drecon: data-driven responsive control of physics-based characters. *ACM Transactions On Graphics (TOG)*, 38(6):1–11, 2019.

Nuttapong Chentanez and et al. Müller. Physics-based motion capture imitation with deep reinforcement learning. MIG '18. Association for Computing Machinery, 2018. doi: 10.1145/3274247.3274506.

Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

et al. François-Lavet. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*, 2018.

et al. Geijtenbeek. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)*, 32(6):1–11, 2013.

Michael Girard and Anthony A Maciejewski. Computational modeling for the computer animation of legged figures. *ACM SIGGRAPH Computer Graphics*, 19(3):263–270, 1985.

et al. Grzeszczuk. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on CG and IT*, pages 9–20, 1998.

et al. Heess. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

et al. Holden. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016.

Paul M Isaacs and Michael F Cohen. Controlling dynamic simulation with kinematic constraints. *ACM SIGGRAPH Computer Graphics*, 21(4):215–224, 1987.

Libin Liu and Jessica Hodgins. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

Ying-Sheng Luo and et al. Soeseno. Carl: Controllable agent with reinforcement learning for quadruped locomotion. 2020.

Li-Ke Ma and et al. Yang. Learning and exploring motor skills with spacetime bounds. In *Computer Graphics Forum*, volume 40, pages 251–263. Wiley Online Library, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, and et al. Silver. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Igor Mordatch and et al. Lowrey. Interactive control of diverse complex characters with neural networks. *Advances in neural information processing systems*, 28, 2015.

Akihiko Murai and Katsu Yamane. A neuromuscular locomotion controller that realizes human-like responses to unexpected disturbances. IEEE, 2011.

Xinlei Pan and et al. You. Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*, 2017.

Soohwan Park and et al. Ryu. Learning predict-and-simulate policies from unorganized human motion data. 2019.

et al. Peng. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4):1–20, 2021.

Xue Bin Peng and et al. Abbeel. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

Xue Bin Peng and et al. Berseth. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

Diego Rodriguez and Sven Behnke. Deepwalk: Omnidirectional bipedal gait by deep reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3033–3039. IEEE, 2021.

David Silver, Aja Huang, Chris J Maddison, and et al. Guez. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Richard Stuart Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

Frank Thomas and et al. Johnston. *The illusion of life: Disney animation*. Hyperion New York, 1995.

Jack M Wang and et al. Hamner. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics (TOG)*, 31(4):1–11, 2012.

J. Wilhelms. Animals with anatomy. *IEEE Computer Graphics and Applications*, 17(3):22–30, 1997. doi: 10.1109/38.586015.

Jane P Wilhelms and Brian A Barsky. Using dynamic analysis to animate articulated bodies such as humans and robots. In *Computer-Generated Images*, pages 209–229. Springer, 1985.

Booth Wilson. Computer animation across the iron curtain: Early digital character design in kitty. *Animation Journal*, 21:4–25, 2013.

Andrew Witkin and Michael Kass. Spacetime constraints. *ACM Siggraph Computer Graphics*, 22(4):159–168, 1988.

Pawel Wrotek and et al. Jenkins. Dynamo: dynamic, data-driven character control with adjustable balance. In *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, pages 61–70, 2006.

Zhaoming Xie, Hung Yu Ling, and et al. Kim. Allsteps: Curriculum-driven learning of stepping stone skills. 2020.

Wenhao Yu, Greg Turk, and C Karen Liu. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

He Zhang and et al. Starke. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)*, 37(4):1–11, 2018.

Industrial Applications of Federated Learning: Google Keyboard query suggestions and next-word predictions

Dan Suman
dan.suman@aalto.fi

Tutor: Alexander Jung

Abstract

Machine Learning is at the heart of innovations that drive our day to day life, ranking search results, recommending videos from streaming service as YouTube or Netflix, as well as helping to drive cars and more. Equally important is the quality and the quantity of data that these models rely on in order to produce accurate predictions. In many industries, such as healthcare, GDPR and personal data concerns play a big role in the quality of the datasets, respecting user's choice for opt-in. To address these concerns, Federated Learning has surfaced as a viable approach of distributed ML, by training central models on decentralized data. In these paper we will explore methods on how to select the data, as well as implementation of Federated algorithms into everyday products.

KEYWORDS: federated learning, google keyboard, LSTM

1 Introduction

Federated learning is a method of distributed computation where the sensitive is stored on client device rather than being collected directly. Instead, model updates are sent using secure computation and differential

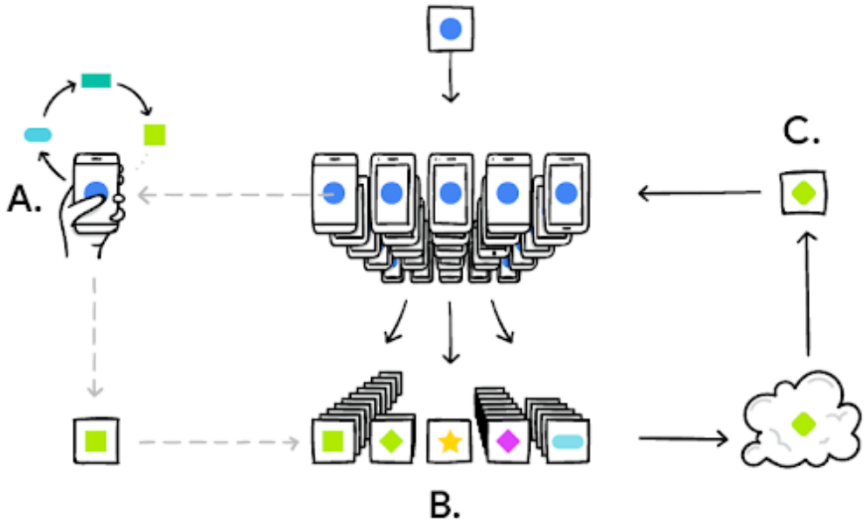


Figure 1. Device trains the FL model locally (A). User updates are collected (B), and if the consensus threshold is reached (C), the shared model is updated and the lifecycle continues.

privacy approaches [5]. The optimal application of federated learning is verified where one of the following propositions is true:

1. User interaction determines the task labels.
2. Datasets are confidential and/or expose sensitive information.
3. The size of the training dataset is too vast to be examined centrally.

In essence, Federated Learning works through a series of rounds in which a a subset of clients, depending on the population heterogeneity and clients available, will participate in the overall model training. Each client gets a copy of the current model parameters and updates it using local training data, such as by running a mini batch stochastic gradient descent epoch. The client then uploads their model update, which the server then averages before combining into a global model. Model lifecycle is better illustrated in Figure 1.

Unlike the traditional way of sending training data to a server, the Federated Learning approach specifies that only the bare minimum of data required for model training (deltas) is sent [3]. There will never be more information in the updates than is required. This decreases the danger of deanonymization through correlation with other datasets, in particular.

Furthermore, approaches such as secure aggregation and differential privacy can be used to improve Federated Learning [1].

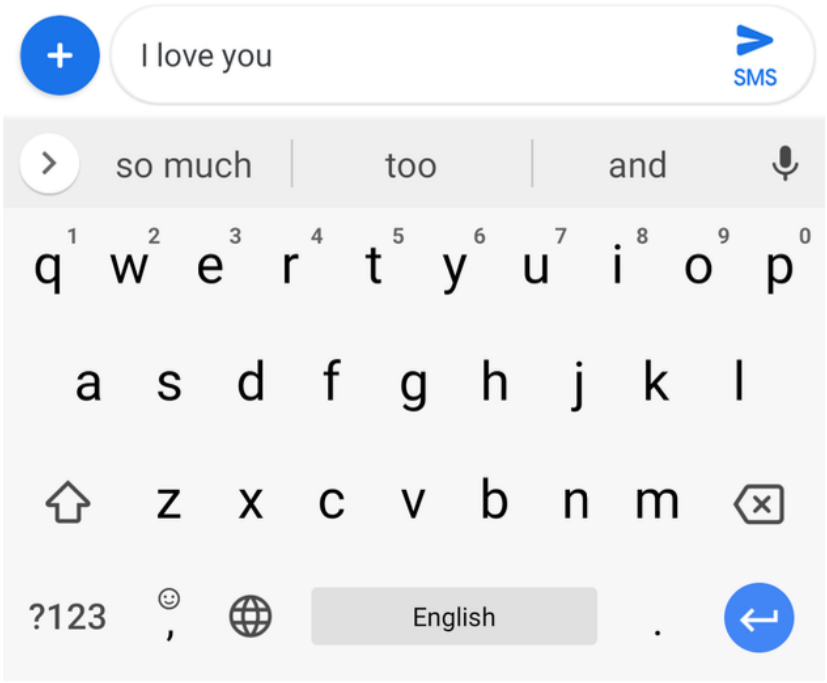


Figure 2. Example of Google Keyboard next word prediction feature. Given the context the keyboard suggests the next words.

2 Google Keyboard Next-Word Prediction

This section will introduce and explore a Google Keyboard Federated Learning application. A server-based stochastic gradient descent is compared against Federated Averaging algorithm, the latter achieving better prediction recall [**?**]. The federated context gives the user better control over their own data, while providing privacy by default and reducing the risk of exposing sensitive data that could be correlated directly to the user.

2.1 Model Architecture

The next word prediction model uses a variant of LSTM (Long Short Term Memory) called the CIFG (Coupled Input and Forget Gate). This architecture has the advantage of reducing the amount of computations and parameter sizes while maintaining performance. On the server, the FederatedAveraging algorithm is used to integrate client updates and create a new global model. A global model w_t is delivered to a subset K of client devices during training round t. In the special case of $t = 0$, client devices start from the same global model that has either been randomly initialized or pre-trained on proxy data [4]. The client computes one or more SGD (Stochastic Gradient Descent) steps with a pre-defined learn-

ing rate. SGD updates are computed locally by the clients and then reported to the server and aggregated. Hyperparameters such client batch size, number of client epochs, and number of clients per round (global batch size) are modified to improve the model. Client updates are never kept on the server; instead, they are processed in memory and deleted as soon as a weight vector has accumulated. In keeping with the principle of data reduction, the submitted content is confined to model weights. Finally, only the aggregated findings are used: the global model is enhanced by merging updates from several client devices. Users must trust that the aggregate server will not investigate individual weight uploads in order to apply the federated learning technique described here.

2.2 Experiments and Results

Starting with random weight initializations, on-device federated learning and server-based SGD are used to train the CIFG model described in the paragraph above.The models are evaluated using server-based logged data, on-device datasets as well as in production scenario (model is deployed to live clients). The findings shows that the FL model outperforms server-trained model on recall metrics. Data shows that federated CIFG improves top-1 recall by a relative 5 percent when evaluated on client cache [4]. In the production context, FL model also improves top-1 and top-3 recall by 1 percent. The authors mention that, while this comparison isn't exactly fair as multiple SGD flavors have been used, the results show that federated learning models represents a viable and preferred solution for neural language models.

3 Google Keyboard Query Suggestions

In this section an application of Federated Learning will be explored in context of Google Keyboard, where the FL trained model is used as a triggering model in additional to a baseline Google Keyboard model. The baseline model provides a series of suggestions as next word suggestion, and the FL model determines if this word is to be shown to the user or not.

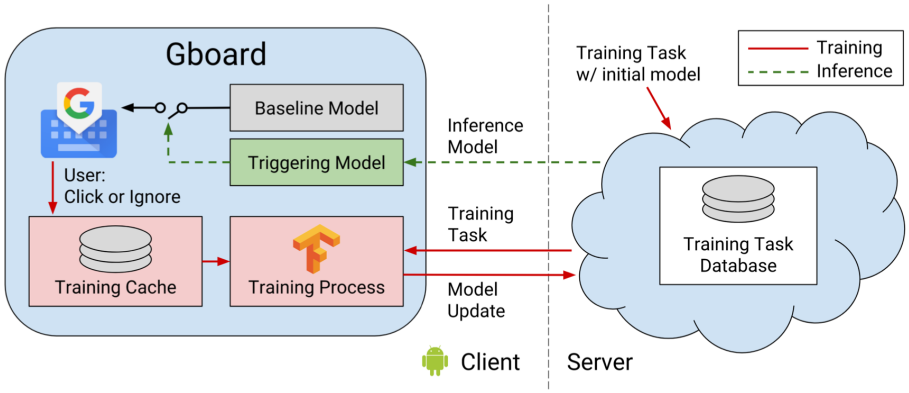


Figure 3. Architecture overview of the Gboard query suggestions.

3.1 Baseline Model

For query suggestions, a baseline model is utilized first, which has been pre-trained using typical machine learning techniques. The model began by comparing user input to a pre-loaded on-device subset of GKG (Google Knowledge Graph) to create query recommendation candidates. To detect probable question candidates, the recommendations are subsequently graded using a LSTM network trained on conversation data. When the Knowledge Graph category of a word in a sentence matches the expected category, the LSTM provides higher scores. The user is shown the candidate with the highest score from the baseline model. The user has the option of implementing or ignoring the proposal. These impressions and clicks are subsequently employed in the Federated Learning model's on-device training.

3.2 Federated Model

The task of the federated model is to take the query candidate from the baseline model and determine whether or not it should be delivered. The model is a logistic regression model that was trained to estimate the likelihood of a click, and the output is a score for each query. A higher score indicates greater confidence in the outcome. A threshold tau is chosen while deploying the model in order to achieve the appropriate triggering rate. By adjusting the threshold, one may strike a balance between delivering value to the user and degrading the user experience. The score is in logit space in a logistic regression model, and the anticipated probability of a click is the logistic sigmoid function applied to the score. The features used as an input to the Federated Model are [6]:

1. Previous Impressions and Clicks are log converted real numbers that represent the current user's number of impressions on past suggestions. The model takes into account clicks and impressions broken down by Knowledge Graph category. As a result, the model can tailor triggering based on previous user behavior.

2. Initial Score, whether or not to show the candidate based on the score generated by the baseline model. Binned, real-valued features are represented. This score is calculated using an LSTM model that incorporates context into the input text.

3. The model can capture patterns in query suggestion click behavior using temporal information stored as one-hot vectors.

4 Conclusion

Federated Learning represents a significant move from traditional Machine Learning in order to address the ongoing concerns around data ownership, GDPR and privacy of sensitive data points. However, the implementation of Federated Learning algorithms at scale in terms of applications is still rather in its infancy. Implementation of such algorithms requires significant re-thinking in terms of existing tools, data models as well as model deployment lifecycle with communication cost as a limiting factor. The set of problems that it can aim to solve is still rather small considering that the majority of state-of-the-art models, as the Gmail spam filters, are still trained fully on cloud because of dataset type, size and computational availability. Federated Learning works best when on-device data is more relevant than data on servers (e.g., the devices generated the data in the first place), is private, or is otherwise unwanted or difficult to transfer to servers. The present applications of FL are for supervised learning problems using labels derived from user behavior (e.g., clicks or typed words) [2]. The user benefits and the overall Federated Learning model performance compared to traditional models make it worth to continue exploring this emerging field while contributing to the Machine Learning community in tackling the challenges that this new approach poses.

References

[1] Muhammad Ammad-ud din, Elena Ivannikova, Suleiman A. Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. Federated collaborative filtering for privacy-preserving personalized recommendation system.

[2] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design.

[3] Adrian Flanagan, Were Oyomno, Alexander Grigorievskiy, Kuan Eeik Tan, Suleiman A. Khan, and Muhammad Ammad-Ud-Din. Federated multi-view matrix factorization for personalized recommendations. 12458:324–347.

[4] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction.

[5] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farokhi Farhad, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis.

[6] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions.

Object tracking for mobile augmented reality

Sami Mairue
sami.mairue@aalto.fi

Tutor: Ashutosh Vaishnav

Abstract

This paper will introduce and evaluate mobile augmented reality object tracking methods that utilize machine learning and traditional tracking methods. The methods presented are divided into three groups: solutions utilizing only the mobile device, solutions using edge and device, and ones using cloud and device. The paper analyzes these methods based on tracking accuracy and latency, bandwidth consumption and battery consumption.

KEYWORDS: mobile augmented reality, object tracking, combined methods, deep learning, edge, cloud

1 Introduction

Over the recent years, augmented reality (AR), and especially its variant that utilizes mobile devices, known as mobile AR, has gained a wide range of use cases from entertainment, such as Pokemon GO, to medical training [8]. Augmented reality is the process of adding virtual elements to a real-life scene taken, for example with a mobile phone camera. This is accomplished by tracking the position and orientation of a plane or an object, to which the virtual object is then placed. In order to maintain

a believable representation, the tracking must occur in real time, which means that the position should be updated tens of times per second [5].

This poses an issue for mobile AR, as mobile devices are typically constrained by processing power and battery consumption, thus limiting the amount of computational resources that can be allocated to tracking. Several solutions have been proposed for this issue. Some utilize external resources, such as cloud computing or edge nodes, but this generates additional latency, which is unwanted [11]. Others have proposed algorithms which run solely on mobile devices [3], but they have to work within the constraints of the mobile hardware. In addition, the object tracking may be implemented in several different ways. Some solutions measure movement by utilizing features in the video codec, such as motion vectors [13], while others take advantage of machine learning [12], or a combination of the two [4].

In order to evaluate the strengths and weaknesses of the different approaches of the combined methods, this paper reviews object tracking approaches that utilize a combination of machine learning and traditional methods. This paper discusses methods that utilize solely the mobile device, and ones that take advantage of external cloud or edge resources. In addition, this paper summarizes the different trade-offs in the methods.

The paper is structured as follows. Section 2 introduces concepts related to object tracking, while Section 3 discusses current solutions. Section 4 analyzes the strengths and weaknesses between the different solutions, and discusses possibilities for future research. Finally, Section 5 provides concluding remarks.

2 Background

Placing virtual objects into the scene in AR typically consists of two phases: object detection and object tracking. Object detection occurs when a new object enters the scene. Object tracking, on the other hand, is performed for every frame that the device captures in order to maintain a correct orientation for the virtual object.

2.1 Object detection

Object detection is the process of estimating bounds for an object or area from the captured video footage, which is then used for overlaying virtual

data or objects on it. The bounds can be either two-dimensional, in which case the tracked area can be visualized as a square (e.g., [6]) or three-dimensional, in which case the visualization can be represented by a cube (e.g., [3]). Recent approaches regarding object detection have typically utilized machine learning algorithms for this problem (e.g., [3], [11]).

2.2 Object tracking

Object tracking is performed once the bounds for the virtual object have been determined. Traditional methods for object tracking include motion vectors, which indicate the magnitude of movement of one or more points between two consecutive frames of video. However, this method does not provide reliable tracking over a long period of time, as was noted by Liu et al. [11]. In order to improve accuracy, solutions utilizing machine learning, such as neural networks have emerged over the recent years (e.g., [9]). However, utilizing only them is not feasible in a mobile device, as running a neural network model is computationally intensive, and as a result, running the model for every frame leads to very poor battery life (see [4]). As a natural follow-up, solutions utilizing both motion vectors and machine learning have emerged (e.g., [4], [3]). These utilize traditional tracking for a large part of the tracking process, but occasionally run the neural network model in order to make sure that the object bounds are correct.

3 Implementations

This section introduces in more detail the different types of implementations of object tracking with hybrid methods and evaluate them. Section 3.1 focuses on implementations that run only on the mobile device, Section 3.2 inspects solutions that move the deep learning models slightly further away to edge resources while keeping traditional tracking in the device, and Section 3.3 inspects solutions that utilize cloud resources, thus moving the deep learning model even further away from the mobile device.

The evaluation will be performed based on the following criteria: tracking accuracy, tracking latency, and battery consumption. For the solutions utilizing external resources, network bandwidth consumption will also be examined. For tracking accuracy, the intersection over union (IoU) metric will be used. IoU is calculated based on the volume, or area that the bounding box of the detected object, and the ground truth box share. If

the two boxes are similar enough, the detection is considered successful. However, different papers use different metrics for determining, whether tracking or detection is successful. For example, Ahmadyan et al. [3] used 0.5, while Liu et al. [11] used a more strict 0.75 as the limit. As such, these should be taken as guiding values that indicate trends between the different methods. In addition, some papers have not measured the battery consumption of their solution.

3.1 Mobile device-only solutions

Solutions utilizing only the mobile device are self-explanatory, they only run solely on the mobile hardware with no need for external resources. The hybrid solutions rely on both traditional methods and deep learning ([3], [4], [10]). The basic idea for these tracking methods is to run the lightweight traditional tracking methods for most of the time, and invoke the deep learning model for only some of the frames. As the processing power of mobile devices has only recently reached a level where running neural network models on them is viable, these are a fairly new field, and as such, research on them is still quite limited.

The working principle of traditional tracking
The traditional tracking part typically analyzes the location of a few key points of the image between consecutive video frames [14]. This can be implemented in several different ways. For example, Ahmadyan et al. [3] utilize the 3D coordinates of the previous frame, and based on them, captured 2D coordinates of the current frame are transformed into 3D coordinates, representing the object in the new frame. As an alternative, Li et al. [10] utilize ARCore for object tracking, which takes advantage of the device's motion sensors for tracking [1]. In addition, Apicharttrisorn et al. [4] included an additional module in the tracking pipeline that keeps track of the changes between frames, and invokes the deep learning model only, if the changes between two subsequent images are too significant.

Results
In terms of accuracy, Ahmadyan et al. [3] were able to reach an accuracy of 0.59 at 0.5 IoU. However, when Li et al. [10] tested the same object detection with their own dataset, they were only able to reach an accuracy of 0.14 at 0.5 IoU, implying that the results are highly situational. With their own solution, Li et al. reached an average accuracy of 0.20, but they

did not state which limit was used for IoU. Meanwhile, Apicharttrisorn et al. [4] reached an accuracy of approximately 0.37 IoU. This is based on the provided graphs, as the actual figure was not explicitly stated.

For latency, the solutions are capable of real-time tracking. Ahmadian et al. stated that the object tracking runs at above 30 frames per second, which translates to approximately 30 ms. Similarly, Apicharttrisorn et al. also stated that their solution runs at 30 frames per second. Both of these can be considered good enough for real-time object tracking. Li et al. stated that their neural network object recognition runs at approximately 7 frames per second, but the ARCore tracking runs in real time.

In terms of effects on battery life, only Apicharttrisorn et al. reported it. This was measured by testing battery drop with the optimized solution and comparing it to a baseline solution that executes the neural network constantly. Each test lasted for 30 minutes on a Google Pixel 2 phone, and it only included battery consumption caused by the object tracking. For all test cases, the baseline consumed 11% of battery. For the optimized solution, battery consumption was 3% in a situation when the network was invoked rarely, and 5% when the network was invoked more frequently due to more movement.

3.2 Edge solutions

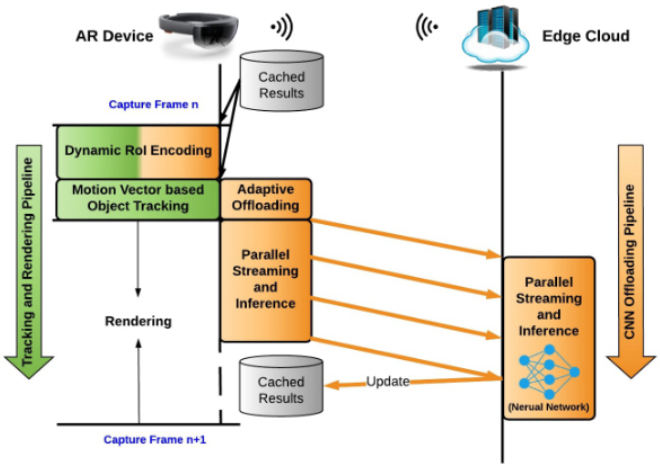


Figure 1. A diagram of an edge solution architecture. Source: [11]

Over the recent years, the popularity of edge computing has increased dramatically, and mobile AR solutions utilizing edge resources have naturally emerged as well. Unlike in traditional cloud computing, edge com-

puting resources are typically close to the user, which in turn leads to a relatively low latency, making it ideal for use cases that have strict latency requirements, such as AR. Object tracking in edge solutions is characterized by offloading the computationally intensive machine learning parts to the edge resources, and maintaining the less demanding traditional tracking methods on the device (e.g., [11]). As the computing power of a typical edge resource can be considered to be approximately equal to a modern desktop PC, it is, at least in theory, capable of running more computationally demanding models, which in turn should lead to a higher accuracy.

Determining when to offload

In general, the tracking process works similarly to the solutions using only the mobile device. The traditional tracking methods utilize motion vectors [11], or other methods of keeping track of key points in the image [15]. Moreover, sending the current frame, and thus invoking the machine learning model can occur for every frame of the video feed [15], or by keeping track of the differences between the last offloaded frame, and only sending the frame, if the differences are large enough [11]. This works similarly to the change detector in the solution by Apicharttrisorn et al. [4], where the motion vectors of the image frames are compared. If the motion vectors are large, meaning that either the camera or objects move rapidly, or a high number of pixels have changed, the image is offloaded to the server.

Results

The main point of concern in both edge and cloud solutions is latency between sending the image to the server and receiving the result with identified objects. This is known as offloading latency [11]. If the image is not processed and sent back to the device quickly enough, the detected points may have moved, potentially leading to inaccurate tracking. In addition, as the edge resources are shared and not scalable, the number of simultaneous users may also affect latency, as the time required to process an image increases.

When testing with no background traffic, Liu et al. [11] were able to reach a latency of approximately 15 ms, depending on the used WiFi frequency, with the higher 5 GHz frequency having a slightly lower latency compared to the 2.4 GHz frequency. This is equivalent to approximately 60 frames per second. While they did not utilize machine learning in their

solution, Zhang et al. [15] highlighted the latency advantage of employing edge computing instead of cloud computing. With edge resources, they were able to reach a latency of approximately 33 ms, or 30 frames per second. This is significantly lower than the 100 ms that was achieved with cloud resources.

When testing the impact of background traffic, Liu et al. did not simulate traffic at the server. Instead, they simulated background traffic that does not necessarily reach the edge server by increasing traffic load at the router. The effects were reported in terms of false detection rate. With no background traffic, the false detection rate was 4.68% when using 5 GHz WiFi frequency, and 10.68% with 2.4 GHz WiFi. When increasing the traffic load of the router to 90%, false detection rate increased to 19.92% for 5 GHz frequency and 32.65% for 2.4 GHz frequency.

In terms of accuracy, Liu et al. were able to reach an accuracy of 0.9 with 0.75 IoU. For battery consumption, Liu et al. did not measure it separately, only stating that the program utilized 15 percent of CPU resources and 13 percent of GPU resources during a 20 minute test session. However, on the device side, Zhang et al. also rely on lightweight tracking and sending data to the edge server. As such, it can be used to roughly estimate battery consumption in edge solutions. In a 10 minute session, when using a fully charged Samsung Galaxy S8, battery consumption was 4.71%.

For network bandwidth usage, Liu et al. tested two methods: their optimized solution that does not send every frame and varies the encoding bitrate for different frames, and a baseline that sends every frame encoded at the same bitrate. For the optimized solution, to reach an accuracy of 0.9 at 0.75 IoU required approximately 55 Mbps of bandwidth. With the baseline solution, reaching the same result required a bandwidth of 150 Mbps. Both cases utilized an image that has a resolution of 1280x720.

3.3 Cloud solutions

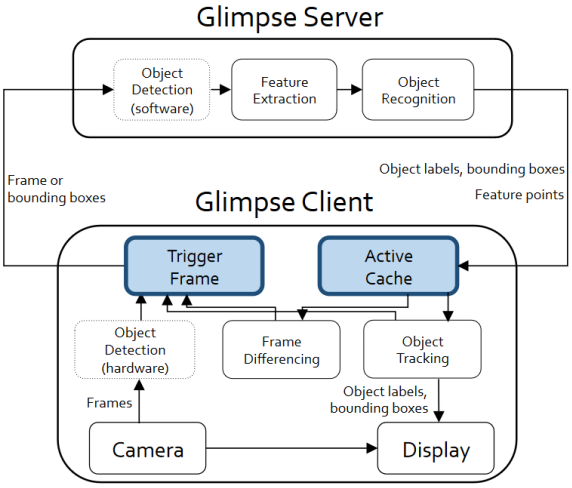


Figure 2. A diagram of a cloud solution architecture. Source: [6]

Utilizing cloud computing in mobile AR has been a popular topic in the research community, starting from the early days of smartphones (e.g., [7]), when the processing power of smartphones was not sufficient to run demanding tracking algorithms locally. The main benefit of utilizing cloud resources instead of edge resources is that the processing time should not depend on traffic load, as unlike in edge servers, the cloud servers do not need to be close to the user, and as such, large datacenters with effectively unlimited processing power can be utilized. However, this comes with the drawback of significantly increased latency (see [15]). As a result, offloading every frame to the server is not possible, and because of this, recent research has mainly focused on determining which frames to offload, as demonstrated in the solution by Chen et al., known as Glimpse [6]. Another key issue in offloading parts of tracking to cloud is how to match the more accurate result received from the cloud, to the current local result that is most likely several frames ahead of the cloud result.

Managing the offloading process

Glimpse [6] aims to solve the issue of matching the results by maintaining a cache of intermediate frames between the current frame and the processed frame received from the server. Once the frame is received, the object tracking is performed from the received frame to the current frame via the cached frames. The main issue is to determine, how many and which frames should be cached. The percentage of cached frames is determined by the end-to-end processing time of a frame, and the time it takes

to compute tracking for one frame on the device. If the processing time is small, more intermediate frames can be cached. Conversely, if computing the tracking requires more time, the number of intermediate frames should be lower. Naturally, increasing the percentage increases accuracy, but also increases the time that it takes to track the changes from the received frame to the current frame. Determining, which frames should be cached, is measured by the differences between two frames. The current frame is cached, if there are significant changes between the previous and current frame. In addition, Glimpse also supports the ability to send only certain trigger frames to the server, in order to reduce the number of frames sent to the server, and thus required bandwidth.

Results

The performance of Glimpse was tested with three different network schemes, two different datasets, and two different configurations of Glimpse. The network schemes consisted of WiFi, Verizon LTE, and AT&T LTE network connections. Of the datasets, one consisted of human faces and the other consisted of roadsigns, while one configuration used trigger frames, while the other did not. Finally, two different configurations that only used server-side tracking were also tested.

When measuring latency, WiFi was the most consistent and fastest, reaching an end-to-end latency of 425-455 ms on face tracking and 510-548 ms on road sign tracking. Verizon LTE had a latency of 656-721 ms on the face tracking, and 901-963 ms on face tracking, and AT&T had a latency of 927-1041 ms on face tracking, while road sign tracking was not tested due to high latency.

For accuracy, when testing without trigger frames and tracking faces, Glimpse was able to reach an accuracy of 92.1% at 0.5 IoU when using WiFi. This decreased to 90.7% when using Verizon LTE network, and further down to 82.6% when using the slower AT&T LTE network. With the roadsign dataset, accuracy was approximately 60% with WiFi, and approximately 38% with Verizon LTE network. Utilizing trigger frames had a slight negative impact on accuracy, with the face tracking accuracy decreasing to approximately 90% on WiFi, approximately 88% on Verizon LTE, and approximately 81% on AT&T LTE. With road signs, the decrease was similar, being approximately 55% on WiFi, and approximately 32% on Verizon LTE. In the server-only configurations, accuracy for face tracking was at most approximately 59%, 40%, and 30% for WiFi, Verizon LTE,

and AT&T LTE respectively. For road sign tracking, the accuracy is under 1%.

Battery consumption was tested on a Samsung Galaxy Nexus, which is a significantly older phone than the ones in other solutions. In addition, battery consumption was only estimated based on power consumption of different operations, and the results were reported in terms of estimated battery life and not percentage. Based on the estimations, battery life for running Glimpse is 1.9 hours with WiFi and 1.5 hours with Verizon LTE.

Network bandwidth was consistent between the two different datasets. When trigger frames were not used, bandwidth was approximately 900 kbps for WiFi and Verizon LTE, and approximately 400 kbps for AT&T. With trigger frames, bandwidth decreased to approximately 500 kbps for WiFi and Verizon LTE, and approximately 300 kbps for AT&T.

4 Discussion

Solution	Accuracy	Latency	Bandwidth consumption	Expected battery life
Mobile device only	20% [10] - 60% [3] (0.5 IoU)	30 ms [3]	none	10 hours [4]
Edge	approx. 90% (0.75 IoU) [11]	approx. 30 ms [11]	approx. 55 Mbps [11]	3.5 hours [15]
Cloud [6]	at most 92% (0.5 IoU) (WiFi)	425-455 ms (WiFi)	approx. 900 Kbps (WiFi)	2 hours

Table 1. A table summarizing the key metrics of the different methods

Based on the results, there are clear trends between the different methods: device-only methods have the lowest and the most predictable latency, but this comes at a fairly significant cost to accuracy. On the other hand, edge and cloud solutions provide higher accuracy, but latency is higher and more unpredictable. This is to be expected, as even the most high-end mobile devices still have less processing power than a modern

PC. As a result, mobile devices are unable to take advantage of more accurate deep learning models, thus hampering accuracy. However, as all of the computation occurs within the device, it does not suffer from delays that may occur during data transfer. In addition, edge solutions may also affected by other load at the server, as the resources cannot be scaled, as is the case with cloud resources.

The battery consumption provided more interesting results, as device-only solutions had significantly lower battery consumption than edge or cloud versions. However, it should be noted that Glimpse [6] used significantly older hardware than the other tests. Regardless, assuming that the battery consumption stays constant in longer sessions, battery life for device-only solutions can be estimated to be approximately 10 hours, while the edge solution should have a battery life of approximately 3.5 hours. The cloud solution has the shortest expected battery life, at approximately 2 hours. However, at 1850 mAh, the battery size of Samsung Galaxy Nexus [6] is approximately half of the battery size of Samsung Galaxy S8 [2] which was used when testing the edge cloud solution [15]. Thus, if the experiments were performed on equivalent devices, the battery life of the cloud solution can be estimated to be roughly similar to the edge solution, which is somewhat unsurprising, as both solutions have the basic operating principle.

Possibilities for future research

Based on these trends, future research could be divided into two different focus areas: improving the accuracy of device-only solutions, and reducing the effects of latency in edge and cloud solutions. For device-only tracking, the deep learning models that modern devices are capable of running in real time are clearly less accurate than the ones that edge solutions are capable of running. Thus, a possible avenue might be to investigate, if it is viable to run a more demanding model, possibly for only a part of the frames, along a lightweight tracking method, and then merge the results.

Continuing research into solutions utilizing external resources is also necessary, as not all devices, especially lower end models, can be expected to be able to run complex deep learning models on the device. Thus, inspecting how to reduce the deteriorated accuracy in situations, where the result is not received before the next frame should be investigated. Implementing a cache, as was done in Glimpse [6], might be a viable way forward. Alternatively, as suggested by the difference in accuracy between

the measurements of Ahmadyan et al. [3] and Li et al. [10], the complexity of the tracked object may affect the tracking accuracy, especially when using only the mobile device. As such, this would be an interesting topic to explore further, and if the effects are significant, solutions that utilize device-only tracking for simple targets, while offloading more complex tasks might be a viable way forward.

5 Conclusion

This paper reviewed and evaluated various implementations of object tracking for mobile augmented reality utilizing deep learning and lightweight traditional tracking methods. Each solution has its clear benefits and drawbacks: device-only implementations do not need a network connection, and they have the lowest latency, but they are the least accurate. Edge solutions provide a relatively low latency and good accuracy, but they require a network connection, and their accuracy may be degraded, if the server is congested. Cloud implementations are not affected by traffic as significantly, but they have the highest latency, which means that methods for hiding the latency from the user must be implemented for a smooth user experience.

References

[1] Fundamental concepts | ARCore | Google Developers. https://developers.google.com/ar/develop/fundamentals. Accessed 5.03.2022.

[2] Samsung Galaxy S8 - Full phone specifications. https://www.gsmarena.com/samsung_galaxy_s8-8161.php. Accessed 17.03.2022.

[3] Adel Ahmadyan, Tingbo Hou, Jianing Wei, Liangkai Zhang, Artsiom Ablavatski, and Matthias Grundmann. Instant 3d object tracking with applications in augmented reality. *CoRR*, abs/2006.13194, 2020.

[4] Kittipat Apicharttrisorn, Xukan Ran, Jiasi Chen, Srikanth V. Krishnamurthy, and Amit K. Roy-Chowdhury. Frugal following: Power thrifty object detection and tracking for mobile augmented reality. *SenSys 2019 - Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pages 96–109, 11 2019.

[5] Ronald T. Azuma. *A survey of augmented reality*, volume 6. MIT Press Journals, 8 1997.

[6] Citation Chen, Tiffany Yu-Han, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, Hari Balakrishnan, and Tiffany Yu-Han Chen. Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices. 2015.

[7] Bai Ruei Huang, Chang Hong Lin, and Chia Han Lee. Mobile augmented reality based on cloud computing. *Proceedings of the International Conference on Anti-Counterfeiting, Security and Identification, ASID*, 2012.

[8] Pier Luigi Ingrassia, Giulia Mormando, Eleonora Giudici, Francesco Strada, Fabio Carfagna, Fabrizio Lamberti, and Andrea Bottino. Augmented reality learning environment for basic life support and defibrillation training: Usability study. *Journal of Medical Internet Research*, 22(5):e14910, 5 2020.

[9] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. Overlay: Practical mobile augmented reality. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, page 331–344, New York, NY, USA, 2015. Association for Computing Machinery.

[10] Xiang Li, Yuan Tian, Fuyao Zhang, Shuxue Quan, and Yi Xu. Object Detection in the Context of Mobile Augmented Reality. 2020.

[11] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, 8 2019.

[12] Nguyen Loc HUYNH, Youngki Lee, Rajesh Krishna BALAN, Nguyen Loc, Rajesh Krishna, Loc N Huynh, and Rajesh Krishna Balan. DeepMon: Mobile GPU-based deep learning framework for DeepMon: Mobile GPU-based deep learning framework for continuous vision applications continuous vision applications Citation Citation DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications. 2017.

[13] Gabriel Takacs, Vijay Chandrasekhar, Bernd Girod, and Radek Grzeszczuk. Feature tracking for mobile augmented reality using video coder motion vectors. pages 141–144, 12 2007.

[14] Jianing Wei, Genzhi Ye, Tyler Mullen, Matthias Grundmann, Adel Ahmadyan, and Tingbo Hou. Instant motion tracking and its applications to augmented reality. *CoRR*, abs/1907.06796, 2019.

[15] Wenxiao Zhang, Bo Han, and Pan Hui. Jaguar: Low latency mobile augmented reality with flexible tracking. *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*, pages 355–363, 10 2018.

Bitrate Adaptation Algorithms in Multimedia Streaming

Timo Laalo
timo.laalo@aalto.fi

Tutor: Esa Vikberg

Abstract

Bitrate adaptation algorithms can be utilized to switch up or switch down the bitrate. Switching the bitrate comes into question when bandwidth fluctuation is detected. With bitrate adaptation the next video segment can be send in different quality than the previous segment. Bitrate is chosen based on bandwidth estimations. Estimations can be done by measuring throughput or by detecting changes in buffer occupancy. Some algorithms measure both throughput and buffer occupancy.

KEYWORDS: bitrate, algorithms, multimedia

1 Introduction

Cisco has estimated that video traffic formed 81% of consumer Internet traffic and 70% of business Internet traffic in 2021 [1]. This number is expected to grow in 2022. Bitrate adaptation algorithms can be utilized during video streaming events. Additionally, same algorithms can be applied for other kinds of multimedia streams. Bitrate can be scaled up or scaled down when the amount of available bandwidth fluctuates [2]. Bandwidth fluctuation can happen for a variety of reasons. Some of

the reasons include changes in signal strength or in network congestion [2]. Bitrate adaptation algorithms operate by adapting to these changes. Bitrate adaptation algorithms provide the best possible bitrate for the client, given the circumstances, such as the network congestion in any given time. Streaming service providers use these algorithms to ensure high quality of experience for their customers [2]. The purpose of this work is to introduce bitrate adaptation algorithms and compare them. In this work the form of multimedia discussed is video. Video stream is more bandwidth intensive compared to other forms of media such as audio making the bitrate adaptation for video particularly beneficial.

In the second section we will overview streaming multimedia content from the server to the client. Third section introduces different types of bitrate adaptation algorithms. In forth section bitrate algorithm testing is done with Sabre. Additionally, other research findings related to algorithms introduced in this work will be further discussed.

2 An overview of streaming multimedia from server to the client

When video is sent from the server to the client, the video is first encoded [6]. Encoding compresses the video to bitstream, which is is transmitted over the network [6]. Multimedia content, such as video is sent from server to the client in segments [17]. The client uses a decoder to decode the encoded bitstream back to the original form. This process is done continuously as the video stream is requested from server to client.

Client utilizes a playback buffer to store the downloaded video segments. Video segment can be played by the client when certain number of bytes of the current segment is downloaded [17]. During this download and play process, bitrate adaptation algorithms are needed. In the case that the available bandwidth is reduced, the client's playback buffer starts to fill and if the playback buffer is full client will experience events of rebuffering and stops in video playback [4]. The reduction of available bandwidth occurs when network congestion is increasing [2]. Transrating can be used to reduce the bitrate of the video [6]. After transrating the video will still have the same media format. That is, the encoding remains the same [6]. Transrating makes the live video stream more scalable. Additionally, the amount of required bandwidth can be reduced by using transrating [6].

Adaptive bitrate algorithms for live streaming have not been researched

as much as bitrate algorithms for video on demand streaming [10]. Developing bitrate adaptation algorithms for live video streaming is harder than developing these algorithms for video on demand streaming. Live streaming bitrate adaptation has additional requirements such as maintaining low latency [18]. In live streaming latency is caused by events such as uploading, encoding, downloading and decoding the source stream [10]. When we compare this process on video on demand streaming, there are additional events. Video on demand streams are already uploaded to the server and each video segment is already encoded with different bitrates. Figure 1 encapsulates the idea of bitrate adaptation.

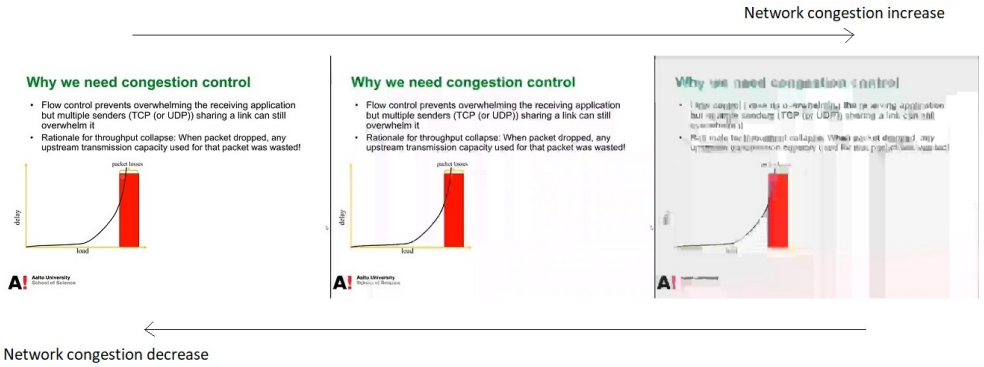


Figure 1. Screenshots taken from a couple second video record of Aalto course CS-E4260. Original bitrate in the left most picture was approximately 420 kbits. To demonstrate the bitrate adaption the screenshot in the center was taken from a video which was transrated to 90 kbits. The right most screenshot is taken from video transrated to 40 kbits. During video stream, bitrate is reduced from left to right as the network congestion increases. When network congestion is decreased from right to left, available amount of bandwidth increases, and the bitrate of the video is increased.

3 Types of bitrate adaptation algorithms

3.1 Throughput-based adaptation

Throughput-based bitrate adaptation considers only the amount of throughput when deciding in which bitrate the next video segment will be send [11]. Throughput is calculated as a ratio of size of the video segment and delivery time of this segment [11]. Throughput-based adaptation works by estimating the available throughput between the server and client [16]. At the most basic level, the initial throughput for some interval i can be used to estimate the throughput for the next $(i + 1)$ segment of the video:

$$T(i) = T^{\mathrm{e}}(i + 1) \quad [11].$$

This measurement produces instant throughput [11]. Since this definition is too sensitive to any bandwidth fluctuation, which might be very short-termed, other types of throughput estimation is usually used, such as smoothed throughput [11]. Throughput is used to estimate amount of network bandwidth to obtain sustainable bitrate and as a metric to detect network congestion [12]. Based on the average throughput, the bitrate can be either switched up if the available bandwidth exceeds the bitrate of the media, or the bitrate can be switched down if the fetch time for current media segment is less than the switch down threshold [12]. Having switch down threshold enables to detect network congestion before the client's playback buffer is fully filled [12]. This prevents buffer overflows and helps the client to save network bandwidth [12]. Examples of throughput-based algorithms include Squad, Festive and PANDA [16][19].

Festive algorithm promotes fairness in addition to bitrate selection [21]. When multiple clients are competing for bottlenecked link, network resources should be allocated equally [21]. Festive computes harmonic mean of the latest throughput estimates to find the amount of available bandwidth after which reference bitrate is computed [21]. The bitrate can be decreased after a chunk of the next video segment is downloaded for clients which have higher bitrate and increased for clients which have lower bitrate [21]. The idea is that bitrate for each client will converge to amount which is allocated fairly [21].

3.2 Buffer-based adaptation

Clients have playback buffer where downloaded segments are stored. Buffer occupancy is measured in seconds of video [14]. Each second one second is played to the user [13]. This video second is then removed from the buffer [13]. The rate in which the buffer occupancy develops depends on video rate $R(t)$ of the segment and on the download rate $C(t)$ [13][14]. When the ratio of download rate $C(t)$ and video rate $R(t)$ is less than one the playback buffer shrinks and when the ratio is greater than one buffer occupancy grows [13][14]

$$C(t)/R(t) < 1 \quad decrease$$

$$C(t)/R(t) > 1 \quad increase$$

Buffer-based adaptation selects the bitrate for the next video segment based on the buffer occupancy level. In buffer-based adaptation there ex-

ists multiple thresholds. Clients buffer occupancy amount is somewhere between these thresholds. An example of thresholds for playback buffer:

$$0 < B_1 < B_2 < B_3 < B_{\max} \quad [11]$$

Bitrate can be scaled up or down or it can remain the same as the buffer occupancy level develops between these thresholds. Suitable bitrate and state of the network can be determined from the speed in which the playback buffer is filling up [19]. Thus, the estimation of the current throughput is obtained from buffer level [11]. When the buffer has become full the estimation of throughput is set to equal the previous estimation of throughput and it will be multiplied by down-scaling factor [11]. Some buffer-based algorithms are BBA, BOLA and ELASTIC [16][19]. BBA algorithm decides which bitrate will be selected based on Quality of Experience metrics, which are used as coefficients for the mapping function which does the bitrate selection [19]. The amount of current throughput is determined indirectly [19]. This is generally true for all buffer-based adaptation algorithms. BOLA algorithm inspects two performance metrics: time-average expected playback utility and expected time that is not spend rebuffering [20]. Time-average expected playback utility is a function of the segments, which has already been viewed by the client [20]. Expected time that is not spend rebuffering can be seen as a metric of video playback smoothness [20]. Algorithm works by maximizing these metrics using Lyapunov optimization [20]. Video playback is divided in time slots [20]. For each timeslot the decision is made whether to download the next video segment or not. When the playback buffer is full the next segment won't be downloaded. When the playback buffer has available space the bitrate of next segment is decided using stochastic optimization between previously mentioned time-average expected playback utility and expected time that is not spend rebuffering [20].

Buffer-based adaptation is also used in live streaming in addition to video on demand streaming. However, there is one key difference: instead of having fixed thresholds for buffer level, the algorithm uses dynamic thresholds [10]. For live streamed multimedia, dynamic threshold usage offers some advantages [10]. Because the buffer is not bound to fixed thresholds, some drawbacks associated with having too long threshold length can be avoided [10]. Mainly, having a too large length would cause initial buffering delays and when the bandwidth fluctuates, more frequent stops in video playback would occur [10]. Algorithm prevents buffer starvation by setting bitrate to equal estimated bandwidth when

buffer occupancy is less than dynamic underflow-threshold θ [10]. When the buffer occupancy is higher than dynamic overflow-threshold, higher bitrate than estimated bandwidth is selected [10]. Unlike in video on demand streaming, in live streaming the buffer will never overflow [10]. This is simply because the server can produce only t seconds of video at startup and the client can only download the amount, which has already been produced, namely the amount of t seconds [10]. In live streaming overflow-threshold is used to utilize fully the available amount of bandwidth and to maximize Quality of Experience by selecting the best bitrate rather than detecting actual buffer overflows [10]. In essence the dynamic threshold algorithm can be summarized in the following idea: When there are fluctuations in bandwidth the goal is to ensure continuous playback and when the state of the network can be considered stable, the focus is to provide best quality in terms of bitrate [10].

3.3 Hybrid adaptation

3.3.1 Conventional algorithms

Throughput algorithms and buffer-based adaptation algorithms both have their drawbacks [27]. Throughput algorithms make the bitrate selection based on the most recent bandwidth estimation [27]. This can lead to overestimating the amount of bandwidth and lead to selection of too high bitrate [27]. This tends to happen when the amount of available bandwidth varies often [27].

Buffer-based adaptation is relying on estimating the bandwidth and bitrate based on the occupancy of the buffer [27]. Buffer-based adaptation estimates the bandwidth based on the source of the stream [27]. As an example, the source can be Content delivery network (CDN) or a peer. Content delivery network will probably provide higher bandwidth than a peer in network would [27]. This will affect the rate in which the buffer is filling making it possible to miscalculate the amount of available bandwidth [27].

Hybrid adaptation combines throughput-based adaptation and buffer-based adaptation. Bitrate selection is done by Throughput Module and Buffer Module. When buffer level drops below minimum threshold, Buffer Module starts requesting next video segments with lower bitrate regardless of throughput algorithms estimations [27]. When buffer level is above

the minimum threshold, Throughput Module will select the bitrate for next video segment instead of the Buffer Module [27].

Model Predictive Control (MPC) is one example of Hybrid adaptation algorithm utilizing throughput estimates and buffer occupancy level when selecting the bitrate for next video segment [28]. When playback bitrate equals the average download bitrate, bitrate for the next segment is selected to equal the bitrate of previous segment [28]. When playback bitrate is more than download bitrate, the algorithm idles for t seconds [28]. Otherwise, the algorithm executes rate adjustment [28].

A variation of MPC algorithm named iMPC is also used in live streaming in addition to video on demand streaming [22]. Algorithm inspects network conditions in sliding horizons [22]. Length of one horizon is set to $m = 3$ for 3 next video segments [22]. At the beginning of the live stream system is in initial state x_0 [22]. Each state is represented as $x_{\mathrm{i}} = [b_{\mathrm{i\text{-}1}}, u_{\mathrm{i\text{-}1}}]^{\mathrm{T}}$ where b_{i} is the buffer length and u_{i} is the bitrate selection action [22]. For each state the throughput is estimated in horizon m. Based on this prediction, optimal bitrate $\mathrm{u_i}$ for each next video segment m in the horizon is estimated [22]. Quality of Experience metrics are used throughout the execution of the algorithm [22]. These metrics penalize rate fluctuations, high latency and video freezes [22]. Reward is given for high bitrate [22]. By utilizing these Quality of Experience metrics, bitrate selection action u_{i} for each state x_{i} can be considered as an optimization problem to maximize Quality of Experience by minimizing the associated cost function

$$\underset{\{u_i\}}{min} \sum_{i=1}^{m} c(x_i, u_i)$$

which is subject to

$$x_{i+1} = f(x_i, u_i)$$

This process is repeated for all video segments during the playback of the live stream. To summarize, algorithm predicts the throughput for each next m segments in horizon [22]. Then algorithm finds the optimal bitrate for m given the network conditions [22]. Optimal bitrate is selected and system is driven to the next state $x_{\mathrm{i+1}}$ [22].

Congestion control adaptation is another type of hybrid adaptation. Algorithms such as Google Congestion Control can adjust sending rate when congestion is detected at link [3]. Multimedia applications such as Google Hangouts utilize Google congestion control algorithm [3]. Additionally, same algorithm is used in other real-time communication applications

and in Chrome web browser in general [3][9]. One-way delay of the transmission is used to infer congestion [3]. Queuing delay is used to estimate delay variations [3]. Queuing delay $q(t)$ represents time it takes for packet to reach the receiver once the sender has transmitted said packet. One-way queuing delay gradient is obtained from the derivative of queuing delay $T_{\mathrm{q}}(t)$ [3]. Formally queuing delay gradient can be represented as

$$T_q(t) = \frac{q(t)}{C} \quad [3]$$

where C is the link capacity [3]. Buffer $q(t)$ with queue filling rate $r(t)$ is

$$\hat{q}(t) = \begin{cases} R(t) - C & 0 \leq q(t) \leq q_M \\ 0 & otherwise \end{cases} \quad [3]$$

To ensure continuous uninterrupted multimedia stream the queue should be kept small while avoiding the situations where the link C would be underutilized at the same time. Network congestion is detected when $\hat{T}_{\mathrm{q}}(t) > 0$ [3]. When $\hat{T}_{\mathrm{q}}(t) = 0$ the length of the queue is constant [3]. This might refer to link underutilization or that queue filling rate is higher than the capacity of the link [3]. Third option is that the filling rate equals the link capacity.

For multimedia streams such as video streams the one-way queuing measurement between sender and receiver can be found by inspecting the transmission of some video frame i [3]. At the beginning the first packet which forms this video frame i is send [3]. Each of these packets that form video frame i has a timestamp T_{i} [3]. Time t_{i} is the time when the last packet for the ith frame is obtained by the receiver [3]. From this the one-way delay variation $d_{\mathrm{m}}(t_{\mathrm{i}})$ is derived as

$$d_{\mathrm{m}}(t_{\mathrm{i}}) = (t_{\mathrm{i}} - t_{\mathrm{i\text{-}1}}) - (T_{\mathrm{i}} - T_{\mathrm{i\text{-}1}}) \quad [3]$$

3.3.2 Reinforcement learning based algorithms

Reinforcement learning is used to deploy bitrate adaptation policies [23]. Similarly, to throughput and buffer-based adaptation, throughput and buffer occupancy are points of interest. Reinforcement learning of bitrate adaptation utilizes states to learn bitrate adaptation policies [23]. Each state is represented as $s_{\mathrm{t}} = (x_{\mathrm{t}}, o_{\mathrm{t}}, \vec{n}_{\mathrm{t}})$ for all steps t where x is the estimated bandwidth available for downloading next video chunk, o represents buffer occupancy and $\vec{n}$ contains sizes for next chunk [23]. Additionally, the action a is defined for bitrate selection action [23]. At each

step t video chunk is downloaded [23]. Reward mechanism used is defined as weighted combination of the selected bitrate and stall time of previous chunk [23]. States s are used as input for neural networks referred as policy neural networks [23]. The output value of policy neural network is a priority value q_t^i for selecting associated bitrate i [23]. Priority values are mapped to probability distribution p_t^i [23]. From p for bitrate selection action a_t the parametrized policy is obtained

$$\pi_\delta(a_\mathrm{t}|s_\mathrm{t}) = [0,1] \quad [23]$$

Deploying this kind of bitrate adaptation policy for clients is suggested to be done in front end service on streaming platforms instead of server applying these rules for each client [23]. Clients will fetch the most recent bitrate adaptation policy [23]. Quality of Experience metrics are closely related for training the model for bitrate adaptation [24]. When video chunk is played for user either reward is given, or penalty is applied [24]. Reward is given based on segment quality [24]. Penalty is applied based on quality fluctuations and rebuffering events [24].

4 Testing bitrate adaptation algorithms and comparing them

4.1 Network simulation and bitrate adaptation with Sabre

Testing bitrate adaptation would require setting up hardware to simulate network or at least setting up virtual machines. To emulate these conditions software called Sabre has been developed [25]. In this section some algorithms provided by Sabre will be tested, including buffer-based adaptation algorithm BOLA, throughput-based network probing bitrate adaptation algorithm and hybrid algorithm dynamic. First test had higher amount of bandwidth $\sim$ 1,5 Mbps - 5 Mbps. The results of first test are in first table. Second test was done in more limited network where capacity ranged from $\sim$ 0.125 Mbps - 1 Mbps. Test results for second test are in second table. None of the algorithms required rebufferings during the first test. Highest average bitrate was achieved with hybrid algorithm in first test. Second highest was obtained with BOLA algorithm and throughput algorithm had the lowest average bitrate. During the first test dynamic algorithm overestimated the bandwidth exactly as often as throughput algorithm having 103 overestimations. BOLA algorithm had 99 overes-

timations. All algorithms required rebufferings during the second test. The amount of required rebuffering events for both BOLA and Throughput algorithms were almost identical with BOLA algorithm requiring 30 rebuffering events and throughput algorithm requiring 32. Dynamic algorithm required most rebufferings having total amount of 39 rebuffering events. In second test highest average bitrate was obtained with BOLA algorithm. However, dynamic algorithm obtained almost same average bitrate. Throughput algorithm had lowest average bitrate. BOLA and dynamic had less bandwidth overestimations during the second test. BOLA did 29 overestimations and dynamic did 49 overestimations. Throughput algorithm overestimated the bandwidth in 180 occasions. Problem with Sabre testing was algorithm availability. Sabre had only one implementation for throughput-based adaptation algorithm and limited amount of hybrid adaptation algorithms. This made the comparison inconclusive.

Algorithm	Time average bitrate	Rebuffering events	Bandwidth overestimation
Throughput	1964	0	103
BOLA	2877	0	99
Dynamic	2905	0	103

Algorithm	Time Average bitrate	Rebuffering events	Bandwidth overestimation
Throughput	296	32	180
BOLA	455	30	29
Dynamic	449	38	49

4.2 Bitrate adaptation comparisons

In research it has been found that buffer-based BOLA algorithm, ELASTIC and PANDA had similar accuracy when estimating throughput [20]. In the same research it was found that hybrid algorithm MPC had more oscillation in bandwidth prediction than BOLA, ELASTIC and PANDA [20]. ELASTIC ensures fair bandwidth allocation with network assistance. However, ELASTIC does not consider Quality of Experience metrics. When bandwidth fluctuations happen multiple bitrate switches might

happen making the Quality of Experience poor [2]. MPC and BOLA achieved similar average bitrate, but at times MPC had higher bitrate [20]. However, more events of rebuffering were observed with MPC than with BOLA [20]. Like in the test which were conducted in section 4.1, in this research also different network profiles were used. The most noticeable observation is that MPC had highest amount of rebuffering events in most of the network test profiles [20]. During Sabre testing hybrid adaptation also had highest amount of rebufferings. However, verifying that rebufferings are more prevalent in hybrid bitrate adaptation would require implementing the actual MPC algorithm in Sabre and compare it with dynamic hybrid algorithm and eventually against throughput and buffer-based adaptation. Buffer-based BOLA algorithm required rebufferings only in limited 3g based network [20]. In 3g network measurement throughput-based PANDA algorithm required even less rebufferings than BOLA [20]. Even though MPC had most rebufferings in 3g network, MPC achieved the highest average bitrate [20]. However, the difference between the average bitrate was very small [20]. It is debatable whether such a small bitrate gain justifies many more rebufferings compared to other algorithms.

5 Conclusion

Bitrate adaptation is needed to provide continuous video playback when the network conditions are fluctuating. When network conditions are stable bitrate adaptation can be used to maximize the Quality of Experience for users. Quality of Experience is important metric when bitrate adaptation algorithm is designed. Sometimes the trade-off between high bitrate and rebufferings must be made. Especially when the amount of available bandwidth is not enough to ensure both. In reinforcement learning for bitrate adaptation models are trained based on Quality of Exprerience metrics. Rewards are given for high bitrate and penalties are applied for events of rebufferings and for other aspects that affect Quality of Exprerience negatively.

Bitrate adaptation can be done as video and other forms of multimedia can be streamed from the server to the client in segments. When the amount of bandwidth changes, bitrate adaptation algorithms react to this by sending the next segment in different bitrate. Bitrate adaptation for live streaming is more challenging than bitrate adaptation for video on

demand streaming. In live streaming the video segments are not yet generated. As the stream is played these segments will be uploaded and encoded after which they can be send to the clients. These additional events will increase latency. Major challenge for live streaming bitrate adaptation is to achieve low latency.

Bitrate adaptation algorithms can be divided in subsections of throughput-based adaptation, buffer-based adaptation and Hybrid adaptation algorithms. Throughput-based adaptation considers only the estimated amount of throughput when deciding the bitrate for the next video segment. Buffer-based adaptation selects the bitrate based on the occupancy level of client's playback buffer. Hybrid bitrate adaptation combines throughput-based and buffer-based bitrate adaptation algorithms. In Sabre testing it was found that Hybrid adaptation algorithm dynamic had the best average bitrate in first test. BOLA and dynamic had almost same average bitrate in second test when bandwidth was more limited. However, dynamic required most rebufferings of all algorithms. Throughput-based adaptation algorithm had worst average bitrate in both tests.

6 References

1. United States 2021 Forecast Highlights - Cisco 2021, accessed 11 January 2022, https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2020_Forecast_Highlights.pdf

2. A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer and R. Zimmermann, "A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP," in IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 562-585, Firstquarter 2019, doi: 10.1109/COMST.2018.2862938.

3. Carlucci, Gaetano & De Cicco, Luca & Holmer, Stefan & Mascolo, Saverio. (2016). Analysis and design of the google congestion control for web real-time communication (WebRTC). 1-12. 10.1145/2910017.2910605.

4. Jabbar, Saba & Kadhim, Dheyaa & Li, Yu. (2018). An Adaptive Bitrate Algorithm Based on Estimation and Video Adaptation for Improving QoE in DASH. 10.2991/csece-18.2018.41.

5. Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. 2011. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In Proceedings of the second annual ACM conference on Multimedia systems (MMSys '11). Association for Computing Machinery, New York, NY, USA, 157–168. DOI:https://doi.org/10.1145/1943552.1943574

6. B. Jedari, G. Premsankar, G. Illahi, M. D. Francesco, A. Mehrabi and A. Ylä-Jääski, "Video Caching, Analytics, and Delivery at the Wireless Edge: A Survey and Future Directions," in IEEE Communications Surveys & Tutorials, vol. 23, no. 1, pp. 431-471, Firstquarter 2021, doi: 10.1109/COMST.2020.3035427.

7. D. Madhuri and P. C. Reddy, "Performance comparison of TCP, UDP and SCTP in a wired network," 2016 International Conference on Communication and Electronics Systems (ICCES), 2016, pp. 1-6, doi: 10.1109/CESYS.2016.7889

8. D. Sisalem and A. Wolisz, "LDA+: a TCP-friendly adaptation scheme for multimedia communication," 2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532), 2000, pp. 1619-1622 vol.3, doi: 10.1109/ICME.2000.871080.

9. G. Carlucci, L. De Cicco, S. Holmer and S. Mascolo, "Congestion Control for Web Real-Time Communication," in IEEE/ACM Transactions on Networking, vol. 25, no. 5, pp. 2629-2642, Oct. 2017, doi: 10.1109/TNET.2017.2703615.

10. L. Xie, C. Zhou, X. Zhang and Z. Guo, "Dynamic threshold based rate adaptation for HTTP live streaming," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1-4, doi: 10.1109/ISCAS.2017.8050574.

11. T. C. Thang, H. T. Le, A. T. Pham and Y. M. Ro, "An Evaluation of Bitrate Adaptation Methods for HTTP Live Streaming," in IEEE Journal on Selected Areas in Communications, vol. 32, no. 4, pp. 693-705, April 2014, doi: 10.1109/JSAC.2014.140403.

12. Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. 2011. Rate adaptation for adaptive HTTP streaming. In Proceedings of the second annual ACM conference on Multimedia systems (MMSys '11). Association for Computing Machinery, New York, NY, USA, 169–174. DOI:https://doi-org.libproxy.aalto.fi/10.1145/1943552.1943575

13. Y. Sani, A. Mauthe and C. Edwards, "Adaptive Bitrate Selection: A Survey," in IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2985-3014, Fourthquarter 2017, doi: 10.1109/COMST.2017.2725241.

14. Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: evidence from a large video streaming service. SIGCOMM Comput. Commun. Rev. 44, 4 (October 2014), 187–198. DOI:https://doi.org/10.1145/2740070.2626296

15. Z. Li et al., "Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale," in IEEE Journal on Selected Areas in Communica-

tions, vol. 32, no. 4, pp. 719-733, April 2014, doi: 10.1109/JSAC.2014.140405.

16. Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2019. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. ACM Trans. Multimedia Comput. Commun. Appl. 15, 2s, Article 67 (April 2019), 29 pages. DOI:https://doi.org/10.1145/3336497

17. K. Miller, E. Quacchio, G. Gennari and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," 2012 19th International Packet Video Workshop (PV), 2012, pp. 173-178, doi: 10.1109/PV.2012.6229732.

18. Huan Peng, Yuan Zhang, Yongbei Yang, and Jinyao Yan. 2019. A Hybrid Control Scheme for Adaptive Live Streaming. In Proceedings of the 27th ACM International Conference on Multimedia (MM '19). Association for Computing Machinery, New York, NY, USA, 2627–2631. DOI:https://doi-org.libproxy.aalto.fi/10.1145/3343031.3356049

19. Jessica Chen, Henry Milner, Ion Stoica, and Jibin Zhan. 2021. Benchmark of Bitrate Adaptation in Video Streaming. J. Data and Information Quality 13, 4, Article 22 (December 2021), 24 pages. DOI:https://doi-org.libproxy.aalto.fi/10.1145/3468063

20. K. Spiteri, R. Urgaonkar and R. K. Sitaraman, "BOLA: Near-Optimal Bitrate Adaptation for Online Videos," in IEEE/ACM Transactions on Networking, vol. 28, no. 4, pp. 1698-1711, Aug. 2020, doi: 10.1109/TNET.2020.2996964.

21. J. Jiang, V. Sekar and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive," in IEEE/ACM Transactions on Networking, vol. 22, no. 1, pp. 326-340, Feb. 2014, doi: 10.1109/TNET.2013.2291681.

22. L. Sun, T. Zong, S. Wang, Y. Liu and Y. Wang, "Towards Optimal Low-Latency Live Video Streaming," in IEEE/ACM Transactions on Networking, vol. 29, no. 5, pp. 2327-2338, Oct. 2021, doi: 10.1109/TNET.2021.3087625.

23. Mao, Hongzi & Chen, Shannon & Dimmery, Drew & Singh, Shaun & Blaisdell, Drew & Tian, Yuandong & Alizadeh, Mohammad & Bakshy, Eytan. (2020). Real-world Video Adaptation with Reinforcement Learning.

24. Y. Zhang and Y. Liu, "Buffer-Based Reinforcement Learning for Adaptive Streaming," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017, pp. 2569-2570, doi: 10.1109/ICDCS.2017.146.

25. Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2018. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. In MMSys '18: 9th ACM Multimedia Systems Conference, June 12-15, 2018, Amsterdam, Netherlands. https://doi.org/10.1145/3204949.3204953

26. Hiba Yousef, Jean Le Feuvre, Paul-Louis Ageneau, and Alexandre Storelli. 2020. Enabling adaptive bitrate algorithms in hybrid CDN/P2P networks. In Proceedings of the 11th ACM Multimedia Systems Conference (MMSys '20). Association for Computing Machinery, New York, NY, USA, 54–65. DOI:https://doi-org.libproxy.aalto.fi/10.1145/3339825.3391859

27. M. Aguayo, L. Bellido, C. M. Lentisco and E. Pastor, "DASH Adaptation Algorithm Based on Adaptive Forgetting Factor Estimation," in IEEE Transactions on Multimedia, vol. 20, no. 5, pp. 1224-1232, May 2018, doi: 10.1109/TMM.2017.2764325.

28. C. Zhou, X. Zhang, L. Huo and Z. Guo, "A control-theoretic approach to rate adaptation for dynamic HTTP streaming," 2012 Visual Communications and Image Processing, 2012, pp. 1-6, doi: 10.1109/VCIP.2012.6410740.

29. Te-Yuan Huang, Chaitanya Ekanadham, Andrew J. Berglund, and Zhi Li. 2019. Hindsight: evaluate video bitrate adaptation at scale. In Proceedings of the 10th ACM Multimedia Systems Conference (MMSys '19). Association for Computing Machinery, New York, NY, USA, 86–97. DOI:https://doi-org.libproxy.aalto.fi/10.1145/3304109.3306219

Firewalls and filtering policies for small networks

Martin Spiering – `martin.spiering@aalto.fi`

Tutor: Aura Tuomas

Abstract

With the rise of cyberattacks, it is necessary to correctly protect one netwrok against external threats and attacks. But setting up efficient filtering rules and firewalls is not that easy. There is, however, a common basis from which one can start and this is what this article aims to provide: what common basis can be used for firewall and filtering rules.

KEYWORDS: network, firewall, security

1 Introduction

Cyber-attacks have risen sharply in 2021. 2021 saw a lot of DDoS attacks [10, 12] and saw a lot of new vulnerabilities: 18,000 at the beginning of December 2021 and 55% of them requires no privileges to be exploited [8]. Allianz, a major European insurance company, stated in its last Risk Barometer that cyber incidents were the most important risks for businesses in 2022.[1]

IoT is also another threat for companies as they are often poorly secured. They often used default passwords that are more often than not let as they are. [9] This means they can be easily compromised and this has lead to some big DDoS attacks: for instance OVH suffered a 1.5Tbps DDoS attacks due to more than 150,000 IoT devices in 2016. [5] They can also

be leverage to have an easy access to the internal network of a company.

Companies need to protect their network against these external threats and firewalls and filtering rules are the first step to that. We will more study those in the context of small networks.

2 Context

In this article, we will study a practical case: see what the current rules are, what can be improved and if we can use that specific case to draw some general rules that can be applied for all small networks.

We will study the firewall and filtering policies set up by MiNET. MiNET is a student non-for profit association of the computer science engineering school Telecom SudParis located in Evry, France. They specialize in network as their goal is to provide a working Internet connection to all the students in their student room on the campus. They have set up over the years a fully working infrastructure in order to achieve that goal including security measures such as firewall and filtering policies. It is fully managed by around 20 students and provide approximately 700 students with a fiber connection. We will study what kind of firewall and filtering policies they have set up, see if they can improve and finally try to draw some general conclusions for firewall and filtering policies in the context of small networks.

We will see in section 3 the existing protections measures, what can be improved in section 4, what are the challenges posed by IPv6 in section 5 and finally draw some general rules about firewalls and filtering rules in section 6.

3 Existing protection measures

3.1 Network overview

Before we start to analyze the already existing firewall rules and protections existing, I must first give an overview of how our network is structured.

Telecom SudParis has allocated a /16 subnet of IPv4 and /48 subnet of IPv6. That allowed the IT department of the school who gives us the Internet connection to allocate us quite a substantial amount of IPv4 and

IPv6 addresses to use and to give to our subscribers. We have a fiber connected to the IT department with a bandwidth of 1 Gbps currently but it will be upgraded very soon to 10 Gbps in order to adapt to the growing bandwidth demand of our subscribers mainly due to streaming.

We have a /21, /23 and a few /24 of IPv4 subnets and a /57 and a /63 IPv6 subnets. That means around 3,000 IPv4 addresses and a virtually unlimited amount of IPv6 addresses for us to use. They are allocated as follow :

- There are 7 student buildings. Each of them has a /24 IPv4 subnet and a /64 IPv6 subnet for wired connections.

- The Wi-Fi is separated from that and has 3 /24 IPv4 subnets and no IPv6 subnet.

- Our production servers have a /25 IPv4 subnet and a /64 IPv6 subnet.

- Our development servers have a /26 IPv4 subnet and a /64 IPv6 subn.et

- Our network equipment such as switches, routers, etc. are on a private IPv4 network that has no Internet connection and is not reachable from outside.

- We also propose a hosting service that is on its own /24 IPv4 subnet and /64 IPv6 subnet.

We require each student to register each of their device using their MAC address to be able to connect to our network. When they register their devices, each device is allocated an IPv4 and IPv6 address that they will use when they connect to our network. We also require them to keep their devices up to date.

3.1.1 Internet access

The IT department of Telecom SudParis provides us with an Internet connection. They are connected to the Internet through Internet Service Providers – Zayo and Renater – and an Internet Exchange Point – France IX. We are connected to the IT department through a fiber between their router and ours.

3.1.2 Wired connections

For the wired connections, as they are less and less used, we are able to provide an IPv4 and an IPv6 address by device. The device can connect to our network using the 802.1X protocol and using PEAP-MSCHAPV2 as the authentication protocol.

Having public addresses, the devices are reachable from the Internet. This allow our subscribers to host their web server for instance with prior agreement from us.

3.1.3 Wi-Fi connections

For the Wi-Fi connections, as they are widely used today, we are not able to do as for the wired connections and provide each device an IPv4 and an IPv6 address. Instead, we provide each subscriber with a unique IPv4 address for all of their device. Their devices are then in a private isolated network and are behind a NAT to reach the Internet. We do not use IPv6 addresses for the Wi-Fi connections.

To connect to our Wi-Fi protocol, the devices have to support WPA2-Enterprise with 802.1X using PEAP-MSCHAPV2 as the authentication protocol.

3.1.4 Production servers

For our production servers, the default rule is that they have no public IPv4 or IPv6 address unless they need it. They are in a private IPv4 network not reachable from the Internet. If they need access to the Internet, there is a proxy for them to use.

If they need to be reached from the Internet, we have a public reverse proxy for all kind of web services and it covers more than 75% of our needs. For the servers that need a direct Internet connection and be reachable from the Internet such as our email servers, they are allocated a public IPv4 and IPv6 address. That explains why we only need a /25 IPv4 subnet for our productions servers as most of them does not require a public IPv4 address.

3.1.5 Development servers

The same default rule applies for our development servers: they have no public IPv4 or IPv6 address. That rule is even stricter has they can have a public IP address only if the service they are running requires it to behave correctly. For instance, when we installed a Matrix instance, we needed a public IP address for the federation to be able to work properly. Apart

from that, only very specific servers have a public IP address such as the proxy server to allow our server to reach the Internet. That explains why we are only using a /26 IPv4 subnet as servers that have a public IP address are the exception.

3.1.6 Hosting service

We provide our subscribers with up to 2 Linux VMs for their personal or student. Each have an IPv4 and an IPv6 public address to have a full Internet access and be fully reachable from the Internet to meet the various needs.

3.2 Firewall

Our network is behind two firewalls. The first one is the firewall of the IT department which is located at the limit of their network with the Internet. We do not have access to their firewall rules, but they filter any incoming connections on any ports <= 1024 unless we ask them to not filter those ports with good justifications – for instance, to host a web service. The justification of that firewall rule is that any port <= 1024 is a privileged port that needs admin privileges to be used. It reduces the risk of someone gaining full access of a machine that is running a compromised service publicly reachable from the Internet. The attacker would need to do a privilege escalation to gain an admin access.

Then, there is our own firewall between our router and theirs with our own firewall rules. We rely on a Linux distribution and iptables [6] as a firewall. We could have also used pfSense [2], but we preferred to use iptable as it works with any Linux distributions whereas pfSense is based on FreeBSD and we are more comfortable with Linux.

We filter packets that have an invalid state using the conntrack module of iptables. We also drop any TCP packet that have an anormal combination of flagged raised:

- NEW without SYN
- various combination of christmas tree scans used by software such as Nmap: FIN,SYN,RST,PSH,ACK,URG for instance [11]
- FIN,SYN or SYN,RST or no flag at all that are used for port scanning

We drop SYN TCP packets after too many of them have been sent by the same source in a small amount of time to avoid SYN flooding.

We also drop packets that we know have a spoofed IP or that are using a private IP as a source for incoming traffic.

We have more precise rules based on the destination or the source of the traffic.

3.2.1 Wired and Wi-Fi connections

For our subscribers, we drop any entering connection on port <= 1024 for the same reason as the IT department. But for the remaining entering connections, whether there are new or already established connections, they are not filtered.

We filter a few ports based on the fact that they are more often that not used for malicious intents and should not be exposed to the public Internet. We filter the following ports:

- ports 135 and 1025: used for RPC
- ports 137 to 139: used for Netbios
- ports 445: used for microsoft-ds
- ports 3450 and 5000: used by various trojans
- ports 5554 and 9996 used by the Sasser Worm

We are also filtering common P2P ports as the French law requires us to do so. As such, ports 1214, 4672, 6346, 6347 and 6881 to 6999 are filtered.

We do not filter any outgoing traffic.

3.2.2 Production and development servers

By default, any incoming traffic is dropped. We allow incoming traffic based on a whitelist that correspond with the services that we are provided. For instance, traffic related to web for our reverse proxy is accepted. Same for the traffic related to email for our mail server.

We have a VPN server in order to reach our network from the outside and it relies on a key authentication.

Outgoing traffic is not filtered.

3.2.3 Hosting service

We do not filter any kind of traffic for the VMs that we host for our subscribers. They are required to properly secure their own VMs and to implement their own firewall if needed.

3.2.4 Mitigation of DoS and DDoS attakcs

We already have been subject to DoS attacks that have completely saturated our bandwidth and firewall. The solution for that is quite simple: blacklist the IP of attacker and ask the IT department to do the same on their firewall to reduce the charge on the network of the school. The IT department also ask the Internet provider to blacklist the IP to reduce the bandwidth used at the highest point possible.

As for DDoS attacks, we have not been a subject of one, but if one would happen, the only thing we could do is to hope our network and firewall will sustain the attack until it stops as there is not much to do expect ban some recurring IPs.

3.3 VLANs

We make a heavy usage of VLANs to isolate different kind of traffic.

3.3.1 Wired and Wi-Fi connections

Each building has its own VLAN. The Wi-Fi has also its own VLAN. The goal is not to isolate the traffic as it is only public Internet traffic but rather to be able to see easily the destination or the source of the traffic.

3.3.2 Production and Development servers

Each have their own VLAN with a private network in order to isolate them from the Internet as much as possible and to isolate them from other kind of internal traffic.

3.3.3 Hosting service

It has its own VLAN to isolate it from the rest of our network as virtually any kind of traffic can go and come from there. It is up to the users to correctly secure their VMs, including deploying a firewall to filter the traffic of their VMs.

3.4 NAT

We only use NATs for the Wi-Fi connection. The main goal is to reduce the number of IPv4 addresses used but it allows us also to isolate devices

by subscriber and reducing the risk for a poorly protected IoT device to be reached and compromised from the Internet. And even if it is compromised, it can only reach the device from the same subscriber, not the devices of other subscribers. But that isolation is only available for Wi-Fi connection and not wired connection even if IoT devices relies rather on Wi-Fi than on wired connections.

4 Improvements

4.1 Firewall

We could filter any new incoming connections as it is only useful and legitimate if you are hosting a service. Even if we want to allow our subscribers to host their own service, we could always allow new incoming connections on a case by case basis.

There is no need to filter the outgoing traffic as we do not want to restrict the usage of our subscribers. Furthermore, such restriction are quite easy to circumvent using HTTP for instance which cannot be blocked for obvious reasons.

4.2 NAT

We could also implement a NAT for the wired connection to have a complete isolation of devices by subscribers. So if a device is compromised, it cannot reach the other subscribers and it is restrained to its own NAT and private network even if it can still reach the Internet. That would also make any IoT device connected using a wired connection not directly reachable from the Internet. Knowing that quite some of them have default admin passwords, it would be better for them not to be reachable from the Internet.

5 The challenges posed by IPv6

The issue with IPv6 vs IPv4 is that we do not own the IP address block. Our Internet provider – the IT department for MiNET or Renater for the university, Zayo does not provide the university with IPv6 capability – could change our IP address block or we could change our Internet Provider. For MiNET, the last option is not possible but the university

could theoritically change its Internet provider.

That is really unlikely as Renater is the national Internet provider for all public universities in France. And as their is no shortage of IPv6 addresses and there will not be one most likely – with 2^{64} /64 IPv6 blocks, we should be fine –, it is also very unlikely that Renater would change the IPv6 block of the university.

The next issue is that, by default, devices are only communicated the IPv6 prefix and then the IPv6 address is selected by the device passively. There is no active attribution. We could not let that default behaviour as we are required by the law, that is why we give a fixed IPv6 address for each device registered when it is registered as describe is section 3.1. The devices can get their attributed IPv6 address by DHCP. And for our servers, we fixed their IPv6 addresses once and for all at the installation of the OS. So we know exactly to which part of our network correspond which IPv6 address.

Furthermore, the IT department gave us enough IPv6 blocks to clearly separate the different parts of our network as describe in section 3.1 too. Finally, our firewall rules are based on where the packet is inbound in our network.

So we just had to copy the IPv4 rules, change the IPv4 addresses to their corresponding IPv6 addresses in our network map and we were able to setup IPv6 filtering rules quite easily when we deployed IPv6 almost a decade ago.

6 Conclusion

A set of general rules that could be applied as a base for firewalls and filtering rules would be the following:

- filter every incoming new connection by default
- authorize case by case any new connection and the strict minimum, only for services that really need to be reachable from the Internet
- no need to block outgoing traffic as it is very easy to circumvent these rules using HTTP for instance
- use VLANs and NATs to isolate different parts of the network according

to the needs

- isolate completely IoT devices if you cannot change the default password. In general, IoT devices should be rather isolated from the rest of the network as we have less control over it as over computers

It could be summarized like this: only expose the bare minimum and have a structured network.

The following references were used to analyze and make improvements to the study case. [3, 7, 4]

References

[1] Allianz. Allianz risk barometer 2022. 2022. https://www.agcs.allianz.com/news-and-insights/reports/allianz-risk-barometer.html.

[2] Electric Sheep Fencing LLC and Rubicon Communications LLC. pfSense Documentation, 16th of February 2022. https://docs.netgate.com/pfsense/en/latest/.

[3] Craig Hunt. TCP/IP Network Administration. O'Reilly, 3rd edition, 2002.

[4] LLC NetCitadel. Firewall Builder Cookbook. 2011.

[5] Pierluigi Paganini. 150,000 IoT Devices behind the 1Tbps DDoS attack on OVH . 2016. https://securityaffairs.co/wordpress/51726/cyber-crime/ovh-hit-botnet-iot.html.

[6] Gregor N. Purdy. Linux iptables Pocket Reference. O'Reilly, 1st edition, 2004.

[7] Michael Rash. Linux Firewalls, Attack, Detection and Response. O'Reilly, 2007.

[8] RedScan. Redscan analysis of NIST NVD reveals record number of vulnerabilities in 2021. 2021. https://www.redscan.com/news/nist-nvd-analysis-2021-record-vulnerabilities/.

[9] Silviu Stahie. Common Credentials Criminals Use in IoT Dictionary Attacks Revealed. 2016. https://www.bitdefender.com/blog/hotforsecurity/common-credentials-criminals-use-in-iot-dictionary-attacks-revealed/.

[10] Alethea Toh. Azure DDoS Protection—2021 Q3 and Q4 DDoS attack trends. 2022. https://azure.microsoft.com/en-us/blog/azure-ddos-protection-2021-q3-and-q4-ddos-attack-trends/.

[11] Wikipedia. Christmas tree packet, 9th of March 2022.

[12] Omer Yoachimik and Vivek Ganti. DDoS Attack Trends for Q4 2021. 2022. https://blog.cloudflare.com/ddos-attack-trends-for-2021-q4/.

Approximation Algorithms for Clustering Problems

Kasper Henriksson
kasper.henriksson@aalto.fi

Tutor: Kamyar Khodamoradi

Abstract

This paper presents the k-median problem with focus on approximation algorithms solving a set of problems called facility location problems. The main intention is to compare and show the different approximation ratios and approaches of the local search algorithms used in capacitated, uncapacitated and mobile facility location problems.

This shows that few advancements have been made in the area of local search algorithms during the past twenty years, although recent papers do show a slight improvement in the approximation ratio, achieving a $(2.836 + \epsilon)$ *approximation scheme using a fixed number of p swaps. Many of the improvements do combine several approximation techniques, such as linear programming and greedy heuristics, to achieve better results.*

Future research should could focus on the mobile facility location problem and achieving improved approximation schemes, as it could help solve and further refine the response to environmental disasters and crises, to better accommodate the needs of humans.

KEYWORDS: k-median problem, approximation algorithms, local search, location problem

1 Introduction

Approximation algorithms are a set of algorithms that aim to solve NP-hard problems by approximating them instead of producing an optimal solution, and they are used when the optimal solution is not known or cannot be computed in polynomial time.

Approximation algorithms are popular in solving the clustering *k-median problem*, and Arora et al. [2] define the problem as where a set S of n points in a metric or Euclidean space is given, in addition to a positive integer k. The k-median problem is then to locate k *medians*, while minimizing the sum of the distances from each of the points of S. The problem is considered a clustering problem, where objects in a set with the same characteristics are grouped together, such that similar objects and characteristics are clustered together.

The k-median problem is closely related to the k-means problem. However, the k-median problem does not take into account the cost of each point in the set S. One popular set of algorithms for solving the k-median problem are the *local search algorithms*.

The algorithms that solve the k-mediam problem can be applied to several different areas, such as pattern classification and data mining, and more practically the facility location problem [2]. The facility location problem relates to finding the optimal location of a facility, such as a warehouse, while satisfying the clients needs and minimizing the cost [8]. By finding an efficient algorithm, businesses can optimize for profit.

This paper examines local search algorithms, a type of algorithm used to solve the k-median problem, in facility location problems. The outline is as follows. Section 2 presents the k-median problem and algorithms more in depth and variations of it. Section 3 discusses the usage of k-median algorithms in local search problems for facility location problems. Section 4 discusses the findings in the paper. Section 5 concludes the paper with remarks and conclusions.

2 K-median problem

This section discusses and introduces the K-median problem in general, and gives several examples for different algorithms with different runtimes.

2.1 Introducing the k-median problem

The k-median problem is a commonly researched clustering problem, that aims to partition a set of points into clusters that have more in common with respect to a defined characteristic [5]. This problem can be approximated by using a set of algorithms called *local search algorithms*. Charikar et al. [4] further state that out of the points, a fixed number of k-points are chosen to be the centers of their respective clusters. The algorithm then goes through the rest of the points in the set S assigning each point to its respective cluster, measuring the distance between each point and the center. The distance between the point and the center is defined as the cost, where the goal is to find an optimal k that minimizes the sum of the costs. The k-median problem and algorithms that solve the problems using local search algorithms, also solve the facility location problem, where the optimization problem is more practical and refers to locations in a metric space.

When discussing the *k-median problem* and the related approximation algorithms, one refers to the cost of the algorithm as how close the solution is to the optimal solution as a factor of OPT. For example, in a minimization problem, a $\rho - approximation$ algorithm, where $\rho > 1$, computes a reasonable solution with cost at most ρ times OPT in the input size and in polynomial time [10]. Furthermore, an approximation schemes computes a reasonable solution in polynomial time as a $(1+\epsilon)$-approximate scheme, where ϵ is any arbitrary $\epsilon > 0$.

Whelan et al. [14] defined the problem mathematically as follows in Equation 1.

$$\underset{y \in R^n}{argmin} \sum_{j=1} ||x_i - y||_2 \tag{1}$$

Megiddo and Supowith [12] define their k-median algorithm as follows in Equation 2.

$$\sum_{i=1}^{n} \underset{1 \leq j \leq p}{min} \{|x_i - z_j| + |y_i - t_j|\} \tag{2}$$

They further prove that the k-median problem is NP-hard to within 50% and that the time complexity of the solution is in $O(n^2p)$ time.

There also exists a version of the k-median problem where the the facilities must be located at input points, called the discrete k-median problem. However, this is out of the scope for this paper. Wang et al. [13] also present two local search algorithms for the k-median problem with linear penalties (k-MPLP) and k-facility locations problem with linear penalties (k-FLPLP), where if a client is denied service it incurs a linear penalty, though these two problems are also out of scope for this paper.

2.2 Initializing cluster centers

There exists several approaches to initialize centers in k-median algorithm [14]. One possible initialization is by randomly choosing k-points from the cluster. Another way to initialize the centers is by analyzing the density and finding high density areas, and selecting a point from this area. This is called *density analysis*, which differs from choosing points according to *single dimension subsets*, where column vectors are examined independently, thereafter choosing the vector with the largest variance and dividing the points inside into k-subsets. *Diagonal initialization* instead creates 2d-grid where the points in the data is split into k-rows and k-columns. We then weigh the points on the diagonal that is created in the 2d-grid by their density, then applying random initialization for the k-centers while taking the density into account. A lightweight initialization is *sampling*, where we take small samples from the set and apply the k-median algorithm on the samples.

3 Application of Local Search algorithms on FLPs

This section introduces the application of local search algorithms on facility location problems, and in short compares the approximation ratios of the found algorithms. The approximation scheme for local search algorithms have improved substantially, with different variations and dependencies.

Cohen-Addad and Mathieu [7] defined a local search algorithm that has a cost within $(1 + \epsilon)$ of the optimal, with a few caveats where the local search algorithm swaps up to $1/\epsilon^c$ points, further stating that the algoritm reaches the solution for the travelling salesman problem (TSP) with fever iterations. The algorithm achieves a similar result by using $(1 + \epsilon)k$, as opposed to using only k.

A polynomial-time local search algorithm was defined by Cohen-Addad and Mathieu [7] as follows in Algorithm 1, this definition is confirmed by Wang et al. [13].

Algorithm 1 Local Search(C)

$S \leftarrow$ Random feasible solution

while $\exists$ S' s.t. Condition(S', ϵ) and $cost(S') \leq (1-1/n)cost(S)$ **do** $S \leftarrow S'$

end while

return S

3.1 Capacitated facility location problem

In the capacitated facility location problem, the facility is associated with a capacity that is the maximum number of clients it can serve [3], which can further be extended to the k-capacitated facility location problem, where we can open a maximum of k-facilities in any location. Cornuejols et al. [8] refer to this as the maximum demand that each facility can supply.

For solving the capacitated facility location (CFL) problem, Arya et al. [3] use a method called the single swap, where each swap closes a facility $s \in S$ and opens a new facility $s' \in S$. Their results show that a single swap local search has an approximation ratio (locality gap) of 5, which is further improved to a $3+2/p+\epsilon = 3+\epsilon$ approximation algorithm if closing and opening a number of p facilities.

As recently as in 2022, Cohen-Addad et al. [6] present a $(2.836 + \epsilon)$-approximation, further improving on the result by Arya et al., where they introduce a new approach that uses a potential function built on the facilities closest and second-closest to the client. This potential function is the sum over all clients, and the addition of distance to its closest facility and the truncated second closest times a small constant. They further state that the facilities are only swapped if and only if we can obtain the latter solution by swapping a constant number of facilities, and the solution has a smaller potential than the earlier solution.

3.2 Uncapacitated facility location problem

In the *uncapacitated facility location* (UFL) problem [10], for each point i in set S there is also a cost c_i defined, and the goal is to minimize the sum of the cost c_I of opening the facilities in addition to the cost of assigning

each point to the closest open facility. In the UFL problem, the capacity that is used in the CFL problem is not used [8].

Cornuejols et al. [8] formalize the problem by considering a set I of customers, that all have a demand. Furthermore, we have a set J of sitese that is the facilities and the locations of them, and let f_j be the cost of opening facility j, c_{ij} the profit that we can gain from client i. They further define c_ij to be a function with parameters such as the price per unit and transportation cost.

Korupolu et al. [11] analyse local search heuristics for facility location problems, and for the UFL problem, they present a proof that the local search algorithm has the approximation ratio $5 + \epsilon$. They prove the following theorem:

Theorem 1 *Let S be any subset of F such that $C(S) > F(5 + \epsilon)C(S^*)$ for some constant $\epsilon > 0$. Then there exists $T \subseteq C$ such that $|S - T \geq 1|$, $|T - S \leq -1|$, and $C(S) - C(T) \geq \frac{C(S)}{p(n)}$.*

By proving this theorem, they form a corollary stating that the approximation ratio for the local search algorithm is $5 + \epsilon$.

Charikar et al. [4] present a greedy local search algorithm that uses several techniques to achieve the approximation ratio of $(2.414 + \epsilon)$, such as *cost scaling* and *greedy local improvement*. The approximation ratio can be achieved with a runtime of $O(n^2(\log n + \frac{1}{\epsilon})$.

3.3 Mobile facility location problem

One variation of the facility location problem is the mobile facility location problem. Halper et al. [9] summarize the problem as seeking to relocate existing facilities, simultaneously assigning customers to the new relocated facilities while minimizing the facility relocation cost and customer travel cost. In essence, the problem differs from the regular facility location problem as the algorithm decision takes into account the cost of moving a location from one place to another [1]. Therefore, the total incurred cost of the solution S becomes as below in Equation 3:

$$\text{MFL(S)} = \sum_{i \in F} c(i, s_i) + \sum_{j \in D} d_j c(j, \sigma(j)) \tag{3}$$

The main results from Ahmadian et al. [1] is a local-search algorithm using combinatorial techniques with a $(3 + \epsilon)$ approximation ratio, improving on earlier approximations using LP-rounding. To achieve this,

Ahmadian et al. allow a fixed number of swaps p, and for any fixed p the optimal local move can be achieved in polynomial time.

Halper et al. [9] introduce the n-OptSwap and n-SmartSwap heuristics for the MFL problem. These two new heuristics improve on the earlier local search heuristics, since they terminate when the facilities and clients have been assigned optimally. As a matter of fact, n-OptSwap is what Ahmadian et al.'s [1] analyze, and showed the approximation ratio $(3 + \epsilon)$ when n is large enough. Unfortunately the paper does not examine the approximation ratio for the n-SmartSwap heuristic, however the runtime of their algorithm is $O(n^3)$, which is less than for the $O()n|F|^2$ n-OptSwap.

Less research can be found on the mobile facility location problem and the usage of local search algorithms on it. There are several practical needs for efficient solutions, such as determining the locations of distribution centers during disasters. [9]

3.4 Comparison

Table 1 shows the main approximation ratios achieved with CFL, UFL and MFL papers discussed in this paper. Cohen-Addad et al. [6] achieve the currently best possible approximation ratio for the capacitated facility location problem, with an approximation ratio of $(2.836 + \epsilon)$, slightly improving on Arya et al.'s [3] single swap solution. For the uncapacitated facility location problem, Charikar et al [4] achieve a $(2.414+\epsilon)$ approximation ratio with a greedy local search algorithm, and for the mobile facility location problem, Ahmadian et al [1] achieve a $(3+\epsilon)$ approximation ratio.

Problem	Author	Approx. ratio	Swaps
CFL	Arya et al.	$(5 + \epsilon)$	Single
	Arya et al.	$(3 + \epsilon)$	Single
	Cohen-Addad et al.	$(2.836 + \epsilon)$	Fixed p swaps
UFL	Korupolu et al.	$(5 + \epsilon)$	[Unknown]
	Charikar et al.	$(2.414 + \epsilon)$	[Unknown]
MFL	Ahmadian et al.	$(3 + \epsilon)$	Fixed p swaps

Table 1. Comparisons of problems and approximation ratios

4 Discussion

The approximation ratios for the local search algorithms have improved during the past decades. Arya et al. [3] show that the approximation ratio for the capacitated facility location problem is $(5 + \epsilon)$ with single swaps, however, this can further be improved by opening and closing set number of p facilities. Furthermore, Cohen-Addad et al. [6] further improve this approximation ratio to $(2.836 + \epsilon)$, where they introduce a new function involving the closest and second closest facilities.

For the uncapacitated facility location problem, Korupolu et al. [11] present an approximation ratio of $(5+\epsilon)$. However, Charikar et al. [4] have already presented an extended local search algorithm that introduces a greedy property, improving the approximation to $(2.414+\epsilon)$, although their approach combines a local search and greedy approach. Furthermore, they introduce a 4-approximation to the k-median problem

For the mobile facility location problem, little can be found related to local search algorithms. As we notice from many of the references, much of the research around local search algorithms stems from more than 20 years back with one of the few major later contributiosn being the improved CFL algorithm introduced by Cohen-Addad et al. [6]. One interesting space that is fairly open and less researched is as mentioned the MFL problem, where there could be room for improvement and new findings.

Much of the research also combines numerous different algorithmic techniques, instead of focusing solely on local search heuristics, some combines linear programming and greedy heuristics. These could offer possibilities for improving approximation algorithms for the k-median problem.

5 Conclusion

This paper has presented and discussed the k-median problem with focus on local search algorithms related to the capacitated, uncapacitated and mobile facility location problems. The k-median problem is a well-researched clustering problem that can be approximated using local search algorithms. These approximation algorithms are commonly used for solving the facility location problem, where the optimal solution seldom can be found, although it exists.

Few new findings have emerged during the past twenty years, although Cohen-Addad et al. [6] do present new research that improves on the more than twenty year old approximation ratio of $3+\epsilon$, to an improved $2.836+\epsilon$ for the capacitated facility location problem. A sub-area of local search research focuses on combinatorial techniques, combining approximation techniques from numerous frameworks instead of only focusing on local search.

Future research could include focusing on the mobile facility location problem or the differences in runtimes between algorithms and approaches.

References

[1] Sara Ahmadian, Zachary Friggstad, and Chaitanya Swamy. *Local-Search based Approximation Algorithms for Mobile Facility Location Problems: (Extended Abstract)*, pages 1607–1621.

[2] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 106–113, 1998.

[3] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on computing*, 33(3):544–562, 2004.

[4] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 378–388. IEEE, 1999.

[5] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[6] Vincent Cohen-Addad, Anupam Gupta, Lunjia Hu, Hoon Oh, and David Saulpic. An improved local search algorithm for k-median. *CoRR*, abs/2111.04589, 2021.

[7] Vincent Cohen-Addad and Claire Mathieu. Effectiveness of Local Search for Geometric Optimization. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 329–343, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[8] Gérard Cornuéjols, George Nemhauser, and Laurence Wolsey. The uncapicitated facility location problem. Technical report, Cornell University Operations Research and Industrial Engineering, 1983.

[9] Russell Halper, S. Raghavan, and Mustafa Sahin. Local search heuristics for the mobile facility location problem. *Computers Operations Research*, 62:210–223, 2015.

[10] Stavros G Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the euclidean k-median problem. *SIAM Journal on Computing*, 37(3):757–782, 2007.

[11] Madhukar R Korupolu, C Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of algorithms*, 37(1):146–188, 2000.

[12] Nimrod Megiddo and Kenneth J Supowit. On the complexity of some common geometric location problems. *SIAM journal on computing*, 13(1):182–196, 1984.

[13] Yishui Wang, Dachuan Xu, Donglei Du, and Chenchen Wu. Local search algorithms for k-median and k-facility location problems with linear penalties. In Zaixin Lu, Donghyun Kim, Weili Wu, Wei Li, and Ding-Zhu Du, editors, *Combinatorial Optimization and Applications*, pages 60–71, Cham, 2015. Springer International Publishing.

[14] Christopher Whelan, Greg Harrell, and Jin Wang. Understanding the k-medians problem. In *Proceedings of the International Conference on Scientific Computing (CSC)*, page 219. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2015.

3D Object Tracking for Mobile Augmented Reality using Deep Learning Methods

Valtteri Valtonen
valtteri.valtonen@aalto.fi

Tutor: Ashutosh Vaishnav

Abstract

Object tracking systems produce object state measurements (position, rotation) from incoming data, and update this state over time, thus achieving tracking. Real-time 3D object tracking systems are necessary for augmented reality (AR) applications on mobile devices, as the information they provide of the physical world is needed to construct a convincing AR overlay on a camera image. Recent object tracking systems for mobile AR use deep learning methods to achieve tracking. This paper is a literature survey of such recent systems. It aims to discover how the deep learning methods are used in the systems, and how they perform. As an end result, the paper presents two functional convolutional neural network based real-time tracking systems for mobile AR applications. Two additional systems are also presented that use deep learning methods in interesting alternative ways compared to the first systems. These methods might be usable with mobile AR applications in the future. The overall conclusion about the state of the art is positive.

KEYWORDS: Computer vision, Augmented reality, Artificial Neural Network, Object tracking, Object detection

1 Introduction

Object tracking is an important problem in the field of computer vision. In the last few decades, it has been the topic of much research. In object tracking, object state (position, orientation) is inferred from an image frame, and this information is maintained over subsequent frames. An object is something that is of interest in the data [17]. Traditionally, object tracking has been useful in tasks, such as air space monitoring, weather monitoring, traffic monitoring and video indexing [17], [2]. However, more recent advances in smartphone technology have enabled a new interesting application area for object tracking: mobile augmented reality applications.

Mobile augmented reality (AR) applications overlay computer-generated graphics on top of the view of the smartphone camera. This AR view is built based on the 3D positions of real life objects [15], meaning that tracking them is important for a realistic experience. These applications present interesting new possibilities for object tracking, but there are also new challenges. One such challenge is the need to function in real-time on the relatively lightweight smartphone hardware.

Recent studies on mobile AR applications have been able to achieve real-time 3D tracking using deep machine learning methods [1]. These methods are able to learn complex nonlinear relations from data, using a trial and error approach. In the case of object tracking, this involves inferring object state from image frame data.

This paper surveys recent deep learning methods for object tracking that are either directly or potentially usable for mobile AR applications. It is organized as follows. Section 2 discusses the basic principles and challenges related to object tracking. Section 3 discusses the principles of machine learning and related challenges. Section 4 presents recent tracking systems and their results. Section 5 discusses these systems and the state of object tracking for mobile AR applications. Lastly, section 6 concludes the paper.

2 Object tracking systems

This section discusses object tracking systems in more detail. It presents the properties and stages of a typical object tracker. Finally, it discusses challenges related to object tracking systems.

2.1 Properties and stages

A general object tracking system accepts data as input and produces object state data (position, orientation) as output. Many recent mobile object tracking systems are computer vision based, so they use image data as input. This is natural, as modern smartphones have powerful cameras. Object tracking systems are not limited to just one type of input at a time. An old hybrid system by Ribo et. al. [11] combines video data with motion sensor data, for instance. The object state data that is output by the system can take different forms. 3D bounding box data is most suitable for AR, as it allows the digital elements of the AR-scene to be placed realistically in relation to real-world objects. However, other types of data, such as 2D bounding box data, are usable as well. It is also possible to convert 2D object state data to 3D form. The EPnP algorithm can be used for this purpose [1]. A tracking system commonly functions in two stages: a detection stage and an update stage.

The detection stage
In the detection stage, one or many objects are detected in the input data, and initial object state is inferred [17]. This is done using features in the data. Features such as color, edges, optical flow and texture are used in computer vision based trackers [17].

Machine learning methods as well as more traditional methods can be used for detection. Recent work [1], [7], [18], [9] uses deep machine learning methods in this phase, but other machine learning methods, such as support vector machines, have also been viable in the past [17]. Yilmaz et. al. [17] discuss three categories of more traditional object detection methods that do not involve machine learning: point detectors, segmentation methods and background modelers. Point detectors, such as the Harris detector [4], Kanade-Lucas-Tomasi detector (KLT) [14] and scale invariant feature transform detector (SIFT) [8], find image interest points at locations that contain some expressive texture. Segmentation methods split the image and produce data on seemingly similar regions. Background modelers construct a model of the image scene, and connect deviations from this model to moving objects.

The update stage
In the update stage, an object tracking system updates the state of known objects. This allows the system to have up-to-date information on the ob-

jects over time, so that tracking is achieved. Different paradigms exist for implementing the update stage, but these are not always exactly followed.

One popular paradigm is tracking-by-detection. In this paradigm, an object detection method is used every frame to produce new object state measurements [15]. If multiple objects are tracked, an object correspondence method is used either jointly or separately with detection to decide, which new measurement corresponds to which old measurement [17].

Another paradigm is simultaneous tracking and detection. In this paradigm, the object detection and update steps are combined inside a single end-to-end method or framework [18]. This helps reduce system complexity compared to tracking-by-detection, which uses potentially slow and complex association strategies [18].

Tracking systems can also utilize motion models to update old state measurements [16]. Motion models contain different rules that are used to produce predictions of the movement of an object soon in the future. They can be used to produce state updates in the time interval between firings of the object detection method, for instance.

Machine learning methods as well as more traditional methods have been used for the update stage. In their survey, Yilmaz et. al. [17] discuss deterministic and statistical point tracking methods, kernel (object shape and appearance) tracking methods and silhouette tracking methods. Deterministic point tracking calculates costs for associating new measurements to known objects. Statistical point tracking methods, such as the Kalman filter, model object state and take data noise and uncertainty into account. Kernel tracking methods do object motion computation. Silhouette methods track complex shapes from one frame to another using an object model generated based on previous frames. Recent work [1], [7], [18], [9] uses deep artificial neural networks in this stage.

2.2 Challenges

Computer vision based object tracking is a challenging problem. Classical challenges include noisy image data, object occlusions, complex object movements and illumination changes [17]. 3D tracking methods that use monocular RGB images have suffered from complex initialization for depth estimation, lacking robustness and the need for heavy computation [1]. The mobile AR environment brings additional challenges, such as the need to function in real-time on limited smartphone hardware. This increases the difficulty of using machine learning methods, as they can be

computationally expensive [7].

3 Basic principles of machine learning

This section discusses the basic properties of machine learning methods as well as challenges related to them. After the basic properties, deep learning methods are discussed. This provides background for further discussion on how deep learning methods are used with object tracking.

Machine learning methods are powerful tools for predicting useful properties from data [6], such as object locations from image data. They have become popular in different fields because of their useful properties. One such field is computer vision, where deep learning has been used to achieve great performance increases in tasks such as object classification [13].

Machine learning methods operate using the principle of trial and error [6]. They accept data as input. This data consists of so called "data points" that are comprised of easily measurable and available properties "features" and the properties of interest that the method tries to predict, "labels". The final aim of machine learning methods is to find a relation between data features and labels, that produces correct predictions as often as possible. This optimal "hypothesis function" is found by enumerating through different function candidates and measuring the error they produce. The function with the minimum error is chosen in the end. This is the training process, and there are different algorithms for it. The training process uses a subset of all possible candidate functions. This subset is the model the machine learning method uses. The error measurement is done using an error function. Different methods can use different error functions.

An example of this process could be found in linear regression, which is a method for fitting a straight line into a collection of data points [6]. In this case, the model would be the set of all possible lines. The training process would enumerate through these lines and measure the goodness of each fit using the square error loss. The line with the smallest loss value is chosen.

A great strength of machine learning methods is their ability to capture complex nonlinear relations between the properties of interest and the generated data using previously generated data [13]. However, challenges are related to the use of these methods as well. It can be difficult to gather

enough data to use with the machine learning method. The data can also include biases that affect the predictions of the method.

Deep learning methods are machine learning methods that use so called artificial neural networks as their hypothesis function. Artificial neural networks consist of nodes that are also called "artificial neurons" [3]. These neurons are connected to each other, and they form layers. The input given to the network propagates through the neurons and is modified, eventually producing output. Deep learning methods include convolutional neural networks (CNN) and recurrent neural networks (RNN) among others. CNNs include convolution and pooling layers to extract high-level features from low-level ones, thus reducing the amount of features [12]. RNNs function in iterations [9]. At least a part of the output of one iteration is fed as input to the next iteration.

4 Deep learning based 3D object tracking

This section discusses object tracking systems that use deep learning methods to perform their task. System operating principles, as well as results are presented.

Deep learning methods have been widely used in recent object tracking work [1], [18], [7]. They can capture relations between image features and objects, as well as previous and new measurements. As long as there is enough training data, deep learning methods are able to handle noisy data and different illuminations. This makes deep learning methods useful in comparison to more traditional methods.

Object tracking systems can use deep learning methods to perform only some part or parts of their task, or the full task [13]. In the object detection stage, they can recognize image features and produce initial object state measurements. In the update stage, the methods can produce object state predictions, as well as do object correspondence. In many cases, tracking systems that utilize deep learning follow the tracking by detection or simultaneous detection and tracking paradigms [18]. In the following, recent deep learning based tracking systems are described. The first two systems are mobile AR tracking systems. The following two systems could potentially be used for mobile AR applications with some modification. The systems discussed are summarized in table 1.

Name	Year	Type	Deep learning method	Performance
System of Ahmadyan et. al. [1]	2020	3D-tracker (Detection plus tracking)	CNN	Real time in mobile AR
System of Liu et. al. [7]	2019	2D-tracker (Motion vectors + tracking by detection)	Cloud edge CNN	Real time in mobile AR
CenterTrack [18]	2020	3D-tracker (Simultaneous detection and tracking)	CNN	Real time on desktop computer
System of Milan et. al. [9]	2017	2D-tracker (Motion model + tracking by detection)	RNN and LSTM	Real time on desktop computer

Table 1. Deep learning object tracking systems discussed in this study

Tracking system of Ahmadyan et. al.

The system by Ahmadyan et. al. [1] is a multiple-object 3D bounding-box tracker meant for mobile AR applications. It tracks unseen objects without prior shape size or model knowledge, if the object belongs to a category the tracker has been trained to recognize. The tracker follows a model called detection plus tracking, where objects are not detected every frame [1]. This helps it achieve real-time performance on mobile.

The system detects objects using MobilePose [5], a CNN based object detector. This detector estimates the 3D bounding boxes of the objects. The 3D bounding boxes are projected to 2D and given as input to a planar tracking method. The method [16] fits a parametric model to motion vectors, and produces updated 2D measurements. These measurements are again brought to 3D using the EPnP algorithm, and tracking is achieved.

Ahmadyan et. al. [1] tested the system on a smartphone. It managed to track objects at above 26 frames per second [1]. Objects were detected quickly, without need for initializing movement. However, tracking the movement of symmetric objects was difficult, as rotating symmetric objects look the same regardless of rotation. A different tracking system called BB8 uses an additional machine learning classifier to solve this problem [10].

Tracking system of Liu et. al.

The system by Liu et. al. [7] is a multiple-object 2D bounding box tracker meant for mobile AR applications. The system achieves real-time performance by combining a heavy object detector CNN at the cloud edge with an on-device motion vector tracking method.

The system encodes incoming video frames with a novel encoding process that provides a better encoding quality to potential regions of interest [7]. This is done to reduce offloading latency to the cloud. An adaptive offloading mechanism then decides whether to offload frames to the cloud edge. This decision is made on the basis of how much the frames have changed since the last offloaded frame. It helps reduce bandwidth and power consumption. If the frame is not offloaded, the on-device motion

vector method updates object state based on the latest detection result that has been cached. If the frame is offloaded, it is sent in slices to the detection CNN at the cloud edge. A detection result is sent back, cached and used during later updates.

Liu et. al. [7] report that the system achieved a high level of accuracy in the object detection task. It achieved latency results sufficient for 6o fps AR on a Nvidia Jetson TX2 mobile development board based AR device. An unstable or slow network might cause trouble for the system, as the latency between the AR device and the cloud server increases.

CenterTrack

CenterTrack by Zhou et. al. [18] is a multiple-object 3D bounding box center tracking system, that follows the simultaneous detection and tracking paradigm. It utilizes heatmaps produced by a CNN.

The system uses a modified version of the CenterNet [19] object detection system for object tracking. Centernet by itself accepts an input image and produces a heatmap based on it, using a CNN. Peaks included in the heatmap are interpreted as object centers. The system can also use image features at these locations to predict object properties, such as bounding box width and height [19]. In CenterTrack, the detector is modified to accept two consecutive frames and a previous heatmap as input. An additional offset vector from current object center to object center in previous frame is also computed as additional output. Object association is done greedily using the distance between the predicted offset and the previous frame center point. Thus, tracking is achieved with an end-to-end trainable network.

The system was tested on the MOT, KITTI and nuScenes datasets [18]. The system was run on a powerful desktop computer, as well as testing servers [18]. MOT relates to pedestrian tracking, and KITTI as well as nuScenes are autonomous driving datasets. CenterTrack outperformed prior state of the art on both the MOT and KITTI datasets. In the case of nuScenes, a monocular baseline was outperformed. Reconnecting long range tracks was challenging for the system [18].

Tracking system of Milan et. al.

The tracking system by Milan et. al. [9] is a multiple object 2D bounding box tracking system. The system uses a recurrent neural network (RNN) as well as a long-short term recurrence network (LSTM).

In their work, Milan et. al. [9] do not describe how objects are detected,

instead they only discuss updating existing measurements. These measurements could presumably be produced by any object detection method. Once object state measurements are available, they are fed to the RNN along with network hidden state, possible new sensor measurements object measurement associations and probability value and probability difference vectors. The RNN predicts new object state values using a learned motion model based on hidden network state values. New hidden state values are also produced and fed into the network at later iterations. Oncenew sensor measurement data, as well as object-measurement association data becomes available, the RNN corrects the motion model predictions and produces true updated object state.

The measurement-object association data is produced by the LSTM-network based on training data [9]. The network accepts hidden state, memory component state and pairwise-distance matrix state. The matrix contains euclidean distance values between predicted object state and measured state. The network utilizes non-linear transformations and a memory component to predict object-measurement assignments step-by-step one object at a time. In addition to the association data, the network produces new hidden state and memory component state, that are fed to the network again at the next iteration.

Milan et. al. [9] report that their system managed to track objects at 300 hZ on a standard CPU. In a pedestrian tracking test, their system did not achieve top accuracy, but it was faster than the top accuracy system.

5 Discussion

This section presents findings made during the literature review. The current state of object tracking for mobile augmented reality is discussed. Potential solutions for current challenges are presented, and new research directions are identified.

Based on existing literature, the state of object tracking for mobile augmented reality is rather good. There is quality research on the topic, and some functional systems have already been built. Deep learning methods, especially CNNs, are popular in the field. Examples of functional CNN systems include the ones by Ahmadyan et. al. [1] and Liu et. al [7].

Of course, the existing systems still face challenges. For instance, the system by Ahmadyan et. al. [1] faced difficulties in tracking rotating symmetrical objects properly. The size of the CNNs also needed to be con-

strained for performance reasons, and the system still reached only 26 fps performance. In turn, the system by Liu et. al. [7] achieved 60 fps performance, but potentially suffers from adverse network conditions. Some of these challenges could potentially be alleviated by combining ideas from different systems. For instance, offloading CNN computation to the cloud could allow a tracking system to use more complex deep networks without worrying too much about device performance. Including an on-device tracking system as a reserve could help deal with adverse network conditions. This problem could also be alleviated by time, as future networks probably have better coverage and stability. An additional deep learning classifier can help with rotational similarity [10].

Therefore, it seems possible that interesting new research could spring from combining these different techniques with each other. For instance, an Ahmadyan et. al. [1] -like system with an additional deep learning classifier that uses heavier CNNs with cloud offloading might provide interesting results. Cloud offloading also makes it possible to use methods that were not originally designed with mobile AR in mind, such as CenterTrack and the RNN system by Milan et. al. [9]

6 Conclusion

This paper has reviewed recent real-time object tracking systems that are either directly usable for mobile AR applications, or could potentially be used with slight modifications. These systems, like many others, are based on deep learning methods that are used to good effect at different parts of the object tracking pipeline.

Deep learning object trackers have already achieved good accuracy and performance results in mobile AR applications [1], [7]. However, at least in some cases the performance could still be better, and other challenges, such as dealing with rotational similarity remain as well. These challenges could potentially be dealt with by applying a combination of existing techniques used in different research. For instance, offloading computation to the cloud edge seems to be a powerful technique that can help waive some of the performance limitations the systems would otherwise have to adapt to.

Object tracking for mobile AR is a challenging problem in the field of computer vision, as it includes the challenges of classical object tracking and introduces new ones, such as the real-time performance requirement

on limited hardware. The problem will be challenging in the future as well, but good progress has already been made towards solving the problem. Thus, the overall state of object tracking for mobile AR seems positive. Research is active, and new methods are being proposed for solving old challenges. Better functioning applications for the technology are not too far from being implemented. These could help build a better more positive future.

References

[1] Adel Ahmadyan, Tingbo Hou, Jianing Wei, Liangkai Zhang, Artsiom Ablavatski, and Matthias Grundmann. Instant 3d object tracking with applications in augmented reality. arXiv, 2020.

[2] Sudha Challa. *Fundamentals of object tracking*. Cambridge University Press, 2011.

[3] IBM Cloud Education. Neural networks. https://www.ibm.com/cloud/learn/neural-networks, 2020. Accessed: 3.3.2022.

[4] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15. Citeseer, 1988.

[5] Tingbo Hou, Adel Ahmadyan, Liangkai Zhang, Jianing Wei, and Matthias Grundmann. Mobilepose: Real-time pose estimation for unseen objects with weak shape supervision. arXiv, 2020.

[6] Alexander Jung. *Machine Learning: The Basics*. Springer, Singapore, 2022.

[7] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. Association for Computing Machinery, 2019.

[8] David Lowe. Sift-the scale invariant feature transform. *Int. J*, 2004.

[9] Anton Milan, S Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. In *Thirty-First AAAI conference on artificial intelligence*, 2017.

[10] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[11] Miguel Ribo, Peter Lang, Harald Ganster, Markus Brandner, Christoph Stock, and Axel Pinz. Hybrid tracking for outdoor augmented reality applications. *IEEE Computer Graphics and Applications*, 22:54–63, 11 2002.

[12] Towards Data Science. A comprehensive guide to convolutional neural networks – the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 2018. Accessed: 4.4.2022.

[13] Krebs Sebastian, Duraisamy Bharanidhar, and Flohr Fabian. A survey on leveraging deep neural networks for object tracking. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[14] Jianbo Shi et al. Tomasi. good features to track. In *Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[15] Hideaki Uchiyama and Eric Marchand. Object detection and pose tracking for augmented reality: Recent approaches. In *18th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, 2012.

[16] Jianing Wei, Genzhi Ye, Tyler Mullen, Matthias Grundmann, Adel Ahmadyan, and Tingbo Hou. Instant motion tracking and its applications to augmented reality. arXiv, 2019.

[17] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), 2006.

[18] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*. Springer, 2020.

[19] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. arXiv, 2019.

Firewall and filtering policy for hybrid cloud

Otso Friman
otso.friman@aalto.fi

Tutor: Tuomas Aura

Abstract

Adoption of public cloud in the IT industry is overwhelming. This adoption comes with multiple challenges, a specific one of which is firewall configuration in the public cloud. Firewall rules were traditionally configured by a small set of network engineers, possibly from a private cloud provider. However, in modern DevOps oriented organizations firewall rules for public cloud are not only 'in house', but also configured 'as code' and accessible by a much larger group of developers and DevOps engineers. Further, using hybrid cloud during migration, or more often indefinitely, brings more complexity, which has to be supported by firewall configuration. This paper explores existing research on the topics of public and hybrid cloud and firewalls, their policy, configuration and visualization. This is done to draft a firewall policy for a real world hybrid cloud setup.

KEYWORDS: firewall, public cloud, hybrid cloud, Azure.

1 Introduction

Cloud computing has risen to prominence in the technology industry. A blocker for adoption has been concerns over security on public cloud. One security concern arises from the user being responsible for configuring

network security themselves on public cloud, yet being dependent on the cloud service providers (CSP) resources to do so [9]. For smaller companies that cannot afford, or don't see a benefit in, building and managing on-premise infrastructure, this differs from the other, often previous option, private cloud [8]. There, the CSP typically manages networks and security for the customer.

Another rising trend in the industry and integrating well with public cloud is DevOps. The specific DevOps practice of automation is in part implemented by using infrastructure as code (IaC), which allows storing the infrastructure code in version control and using it to automate infrastructure deployment and configuration [7, 13]. This ability in public cloud to rapidly change infrastructure, enabled by DevOps and IaC, imposes a challenge in keeping the changing infrastructure secure. An addition to this challenge is a quite typical hybrid cloud setup, where a private cloud is connected to public cloud [8]. Connecting the two clouds securely further complicates network configuration and security.

The goal of this paper is to construct a firewall policy for the public cloud side of a hybrid cloud setup, taking into account the above discussed challenges and that in accordance with DevOps, the firewall and its rules should be defined in code and automatically deployed. Central research questions are: **1.How to configure a public cloud firewall policy for hybrid cloud? How to do this firewall configuration in a way keeps the firewall functioning securely in changing infrastructure? 2. How to ensure firewall correctness and usability?**

This paper starts with a literature review into network security for hybrid cloud, focusing specifically on firewalls and public cloud. After this, constructing a firewall policy and firewall configurations suitable for the setup are examined. This is followed by researching ways to visualize firewall configuration to support the usability and verifying correctness of the firewall. These examinations are used to finally create a firewall policy.

2 Firewalls for public and hybrid cloud

In this section, background research and an introduction into the topics is conducted. Firewalls are first discussed to understand the different types, and challenges in configuring them. This is followed by reviewing what firewall options exist in public cloud, specifically Azure. After this, hybrid cloud and connecting the private cloud side to public cloud is discussed.

2.1 From firewalls to public cloud firewalls

Firewalls are the main defence between a trusted machine or network and an untrusted network, specifically in this paper's context a private network and the internet. The purpose of a firewall is to prevent unwanted connections from the internet to the private network or from the network to the internet. Ingham et al. [5] do a good job of mapping out the history and development of different firewalls, starting from basic packet filters to more advanced solutions like stateful inspection firewalls (SPI) and application level gateways, also called application proxies. Packet filters simply inspect network packets attributes against a set of rules. These attributes include the IP addresses and ports of the source and destination as well as the protocol being used. SPIs take this further by storing the session information to determine which side initiated the connection. This allows separate inbound and outbound rules [5].

CSPs offer native firewall solutions for their platforms, the one on Azure being Azure firewall, which is an SPI firewall, though it has some 'next generation firewall' features as well, for a price premium [14, 16]. These 'next generation' features are proprietary software and do not have great visibility into them, so they are mostly ignored in this paper, though offerings like an intrusion detection system (IDS) can be quite useful. Azure firewall is in a sense a PaaS component as the actual implementation cannot directly be interacted with, and the firewall rules are configured on a higher abstraction layer, through APIs or a web interface [16]. These APIs allow the programmatic control of the firewall, enabling writing the firewall rules in code, conforming to the DevOps practice of IaC [13]. The Azure firewall allows for creating rules for IPs, fully qualified domain names (FQDNs) or sources using network address translations (NAT). Rules can also be grouped into sets, which are then given a priority number to set the rule order (the group with smallest priority number considered first) [1].

In literature, firewall configuration is found to be complicated and prone to errors with these errors leading to severe vulnerabilities [17]. A challenge comes from a firewall policy consisting of many rules which can be conflicting, and the large number of rules leads to complexity, which is found to drastically increase the likelihood of configuration errors [19]. Various firewall analysis and configuration tools and methods have been developed to help with this problem [17]. From a public cloud perspective

these challenges are even trickier as the tools and methods often cannot be applied directly because public cloud firewalls are not directly configurable as mentioned above.

2.2 Hybrid cloud and connecting networks

Hybrid cloud is a cloud computing setup where a private and a public cloud are used together. This can be during a migration phase, when a company is moving to public cloud or can be a permanent solution, for example if something cannot be migrated to public cloud because of legal reasons. The arising challenge is how to securely connect the two knowing they can be physically relatively far from each other. The simplest and unsafest way is to have public endpoints on the public cloud and simply access them from the private cloud over the internet. This insecurity can be addressed by using a virtual private network (VPN). Azure has a VPN solution in VPN Gateway, which allows connecting a site-to-site VPN from the private cloud side using standard protocols IKEv2 and IPSec [16].

Using a VPN does have one major downside, while traffic is secured through encryption, it still goes over the public internet. This may not be ideal, if some critical interaction is happening between the public cloud and private cloud side. Congestion during times of high traffic, or network outages are out of one's control and can completely halt a service. Because of this, basically all CSPs offer network peering, directly physically connecting the private cloud to the public cloud. On Azure this is called Express route. Express route offers guarantees for uptime, speed and latency in the form of a service level agreement (SLA), since the connection is private [3]. Express route is connected to a virtual network on Azure with a virtual network gateway, and standard BGP is used for exchanging routing information between the two networks, effectively creating one large private network [3].

3 Applying firewall practises to public cloud

This section first discusses the hybrid cloud setup to determine requirements for firewall policy. After this, research into firewall policy is covered, which is used to make suggestions about the final firewall policy. Following policy, the solution of distributed firewalls is reviewed to make decisions on how to configure, and where to place firewalls. Finally, fire-

wall usability and visualization are covered to support the secure updating of rules.

3.1 The hybrid cloud setup

The hybrid cloud setup in question is a development environment migrating from private cloud to Azure. Express route is established between the two clouds. On Azure, a main firewall is in place facing the internet, and resource or subnet specific firewalls are configurable as well. The resources in Azure, along with firewalls and rules, are defined in code using Terraform. The Terraforms are hierarchical, and since the main Azure firewall is a critical network component, updating firewall rules requires applying the lowest level Terraforms. Although Terraform is quite good at showing what changes will be made to infrastructure, frequently applying the network level Terraforms imposes at least some risk. Further, on complex changes, seeing exactly how the firewall configuration changes is not easy from the Terraform changelog. Desirable goals for a firewall policy are minimizing changes to the main firewall, configuring changes in a way that minimizes risks for misconfiguration and some sort of visualization of the firewall policy.

3.2 Firewall policy

A NIST technical report [18] composes firewall policy guidelines. The process starts by assessing the network to be secured, what are the main threats and weak points. With this knowledge, a decision on what firewalls to use and where should be made. Next what tools or applications are allowed to pass through the firewall and under what conditions needs to be reviewed. From this a firewall policy and rule set can be drafted. The firewall policy should also include when and how to test the firewall and how to manage it. These steps should also be repeated to consolidate the policy, whenever the security policy changes notably [18].

In this case, as this is a Development environment, the firewall ruleset is simplified by the fact that no inbound connections are allowed. Users of the environment connect to the private cloud using VPN and can access the public cloud side through the Express route peering. Outbound access is required for various build and test automation tools to access external resources such as container registries and code repositories. To minimize the attack vector, outbound connections should also be dropped by default.

Firewall rules can be set up for all required external resources, and the rules can be grouped into logical groupings for similar resources. This is to minimize the affected rules in case of misconfiguration, as external resources and the rules for accessing them are expected to change often. It is reasonable to assume that frequency of change is correlated with the type of resource, which justifies the grouping. As all created rules are to allow connections, the ordering of rules is not critical, but some performance improvement can be found from ordering the rule groups by how often they are used, assigning the most often used group with the smallest priority number. Using only 'allow' rules also simplifies complexity and reduces the likelihood of rules conflicting.

3.3 Firewall and rule distribution

Public cloud allows for the easy creation of multiple environments from sandboxes and development environments to staging and production. As the number and use case of these environments grows, having a single firewall handling incoming and outgoing traffic (north-south traffic) is no longer sufficient, but traffic between environments (west-east traffic) needs to also be controlled. To solve this, the practice of distributed firewall design can be utilized, as it addresses how to set up and manage firewalls both on the perimeter and inside the network, on the assumption that internal traffic may not be trusted [4]. Distributed firewalls offer various benefits including no longer having a single point of failure in the main firewall of a network, fine grained access control to resources inside the network, performance benefits as the main firewall is no longer a choke point and not having to tunnel external machines into an isolated network [4]. Some benefits cannot be realized in the cloud, such as having the firewall and the application on the same machine, as cloud typically works on a higher level of abstraction utilizing orchestration tools such as Kubernetes. In this specific scenario, distributed firewalls do offer an additional benefit, as having distributed firewalls means changes to specific firewalls (and possible configuration errors) become less frequent and the scope affected by a misconfiguration decreases.

The main challenges for distributed firewalls are how to define the distributed policy and how to distribute the centrally managed rules to the firewalls securely [6]. Azure uses role based access control (RBAC) for managing resources [2], which allows securely distributing the policy and configuring the firewalls. On the policy definition side, implementing the

original idea of distributed firewalls with no main firewall and no topological isolation is not desirable, as this scenario does not need its main benefit of external access without tunneling, but does become less secure due to lack of isolation. Instead, a hybrid approach (also called 'a cascade of distributed firewalls' [15]) with one main firewall on the perimeter, and firewalls configured for subnets and possible resources as needed seems practical. This approach does keep the other desirable benefits of distributed firewalls that were listed in the paragraph above. Typically, the biggest downside of this approach is seen to be the uncertainty of whether the added security of this approach is worth the added cost of additional firewalls [15]. On Azure, this is not an issue, as higher level components such as keyvaults and file or blob storages come with built in firewalls for no additional cost, and subnets used by compute solutions like Kubernetes or simple virtual machines can similarly be secured with Network Security Groups (NSGs) for no additional cost [16].

The starting point for rule configuration, as was the case with the main firewall configuration discussed above, is to drop all traffic by default, and configure firewall rules only for necessary connections. The main firewall can have more generic rules that apply to the whole network, and resource or subnet specific firewalls can have use case specific rules. This allows network level access control and isolation of more sensitive applications as needed. This approach is depicted in figure 1. The cascaded firewalls approach also allows for 'repeated' rules, defining similar rules on multiple firewalls, which can further reduce the impact of misconfiguration of singular firewalls. Infrastructure changes are also much less likely to cause security risks, as traffic is blocked by default, and rules to connect to new or changed infrastructure need to be explicitly set.

3.4 Firewall usability and visualization

The firewall distribution and repeated rules discussed above increase the amount of firewalls and rules. Knowing that usability of a firewall configuration in general is a challenge [17], this increased complexity needs to be met with ways to support it. Using automation for firewall management eases supporting complexity, and clear guidelines on only creating rules for necessary connections helps understanding complex rulesets by giving clear reasoning behind rules. In literature, one big way of helping with firewall usability has been using visualization [17]. With firewalls the actual rules can be visualized, some notable examples being [12, 10].

Mansmann et al. [12] point out a challenge in detecting and removing rules that have become unused [12]. This seems indeed applicable to our case of rapidly changing cloud infrastructure and quite restrictive policy with rules for every required external resource, which can also change quite frequently. Kim et al. [10] note a problem in conflicting rules, resulting in some rules being ignored and thus unnecessary complexity, which can be noticed by their visualization tool [10]. Another thing to visualize is monitoring of the firewall. As the amount of rules and firewalls increase, relying on firewall logs alone for monitoring is no longer enough and using visualization is beneficial [11].

Firewall visualization tools existing in literature are not directly usable on the proprietary Azure firewall, but their findings can be utilized while creating visualizations on an Azure visualization tool called Monitor Workbooks [16]. An example workbook exists for firewall use, with some useful visuals like firewall usage over time and most frequent allowed and blocked IPs and FQDNs [1]. This workbook can be expanded upon, visualizing any firewall logs or diagnostics [1]. One very useful metric to visualize is hits per firewall rule, since notable concerns are unused rules or ignored rules, caused by conflicting rules. These will be easily detectable, since they will not have any hits. Storing firewall rules in version control also helps with determining if rules are obsolete, by looking at change history and commit messages of rulesets.

4 Results

The chapters above suggest practices for firewall policy, firewall distribution and visualization using justifications from literature. This chapter combines the suggestions and findings to a coherent policy. A simplified example setup is used to explain the policy, in order to protect intellectual property of the true setup. Figure 1 visualizes this example. It depicts the setup in Azure, two Kubernetes clusters with different functions and sensitivity as well as the Express route to private cloud and the Azure firewall. Arrows depict connections allowed by firewall rules. Finally, after covering the policy, improvements and ideas for future work are discussed.

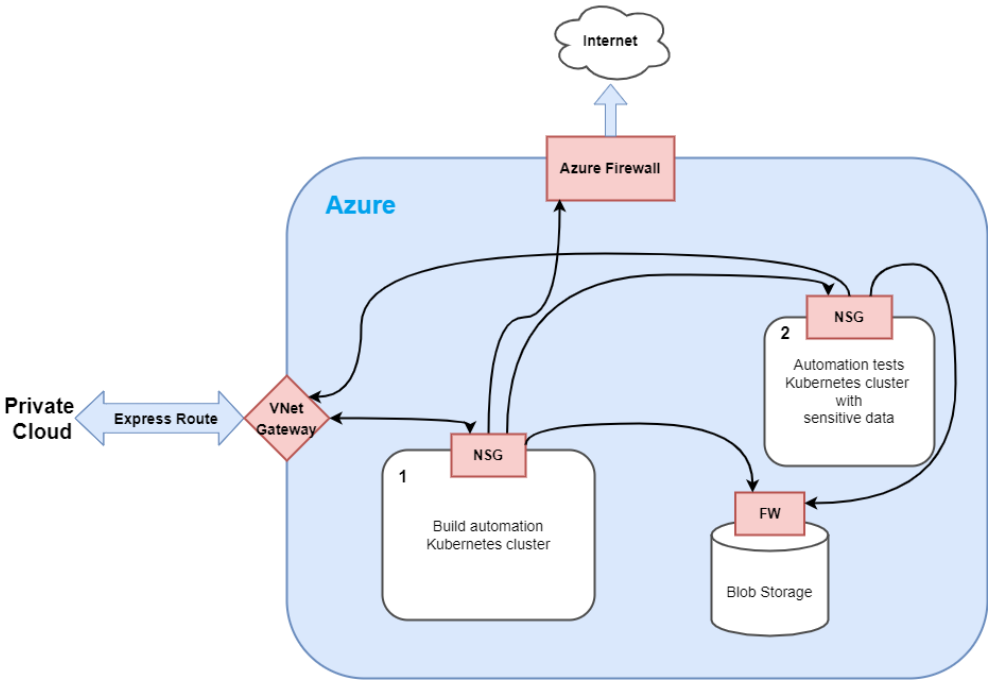


Figure 1. Example firewall configuration in Azure

4.1 Suggested firewall policy

A main Azure firewall is configured on the perimeter, and all traffic going to the public internet passes through it. All subnetworks inside Azure should have Network security groups defined, working as firewalls for the subnetwork. Further, all higher level resources such as blob and file storages and keyvaults should have firewalls enabled as well. Firewalls should drop all traffic by default, and rules must be defined for all necessary connections. This allows isolation of sensitive applications like cluster 2 in figure 1, which is managed by cluster 1 but cannot access the internet, and cannot be accessed directly from the private cloud side, but can post results to the private cloud. Repeating rules on different firewall levels allows robustness against misconfigurations in single firewalls, for example if the main firewall is misconfigured to allow access to erroneous external resources the NSGs for clusters 1 and 2 will still block this access. To connect to the private cloud side, express route should be used, using a virtual network gateway to connect it to the Azure virtual network, also depicted in figure 1. Since connecting to Azure can be done from the private cloud side via Express route, there is no reason to allow any inbound traffic through the Azure firewall.

All firewalls and rules should be configured as code using Terraform, and stored in version control. Firewall rules should be grouped by use

case, for example external build repositories being one group. The rule groups should be ordered so most often used resources are in rule groups considered first. Automation should be used to deploy both the firewalls and rules. Code reviews can be used to review rule changes before they are merged and automation takes over to deploy them. The Terraform changelog can be checked to verify that the intended firewall changes are applied. For larger rule changes, visualization enabled by Azure workbooks can be checked to verify correctness. Workbooks should also be used to periodically review rules with no firewall hits to determine if they are obsolete or ignored due to conflicting rules. Since rules are configured on multiple firewall levels, obsolete rules can also be detected by finding rules which do not have counterparts in other firewalls. For example in figure 1, the main Azure firewall having a rule for a build repository, which cannot be found in the cluster 1 NSG may be obsolete.

4.2 Discussion on improvements and future work

The suggested policy in the end relies on humans to check visualizations to determine correctness of policy and detecting obsolete rules. As automation is already used for deploying firewalls and their configuration, it is logical to extend this automation to verify correctness and detect obsolete rules. Correctness could be verified for example by automated probing, while firewall logs could be consumed by automation to detect obsolete rules with no hits. This automatic policy control and compliance monitoring could be the basis of future research.

5 Conclusions

Cloud infrastructure can be rapidly changing, which challenges firewalls to keep the changing infrastructure secure. On the other hand, public cloud allows easy creation of firewalls for all requiring resources and central control of firewalls using IaC and automation. By dropping traffic on firewalls by default, and explicitly setting rules for allowed connections and repeating them on different levels of firewalls increases the amount of rules, but ensures security during misconfigurations or infrastructure changes. Visualizations of firewalls and their metrics help with managing these large rulesets.

References

[1] Azure firewall documentation. https://docs.microsoft.com/en-us/azure/firewall/, 2022. Accessed: 5.4.2022.

[2] Azure RBAC documentation. https://docs.microsoft.com/en-us/azure/role-based-access-control/, 2022. Accessed: 5.4.2022.

[3] Expressroute documentation. https://docs.microsoft.com/en-us/azure/expressroute/, 2022. Accessed: 5.3.2022.

[4] Steven M Bellovin. Distributed firewalls. *;login: Magazine, Special Issue on Security*, 1999.

[5] Kenneth Ingham, Stephanie Forrest, et al. A history and survey of network firewalls. *University of New Mexico, Tech. Rep*, 2002.

[6] Sotiris Ioannidis, Angelos D Keromytis, Steve M Bellovin, and Jonathan M Smith. Implementing a distributed firewall. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 190–199, 2000.

[7] Ramtin Jabbari, Nauman bin Ali, Kai Petersen, and Binish Tanveer. What is DevOps? a systematic mapping study on definitions and practices. In *Proceedings of the Scientific Workshop Proceedings of XP2016*, pages 1–11, 2016.

[8] Yashpalsinh Jadeja and Kirit Modi. Cloud computing - concepts, architecture and challenges. In *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, pages 877–880, 2012.

[9] Wayne A Jansen, Tim Grance, et al. Guidelines on security and privacy in public cloud computing. Special publication, National Institute of Standards and Technology (NIST), 2011.

[10] Hyungseok Kim, Sukjun Ko, Dong Seong Kim, and Huy Kang Kim. Firewall ruleset visualization analysis tool based on segmentation. In *2017 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8. IEEE, 2017.

[11] Christopher P Lee, Jason Trost, Nicholas Gibbs, Raheem Beyah, and John A Copeland. Visual firewall: real-time network security monitor. In *IEEE Workshop on Visualization for Computer Security, 2005.(VizSEC 05).*, pages 129–136. IEEE, 2005.

[12] Florian Mansmann, Timo Göbel, and William Cheswick. Visual analysis of complex firewall configurations. In *Proceedings of the ninth international symposium on visualization for cyber security*, pages 1–8, 2012.

[13] Kief Morris. *Infrastructure as Code, 2nd Edition*. O'Reilly Media, 2020.

[14] Kishan Neupane, Rami Haddad, and Lei Chen. Next generation firewall for network security: A survey. In *SoutheastCon 2018*, pages 1–6, 2018.

[15] Robert N Smith, Yu Chen, and Sourav Bhattacharya. Cascade of distributed and cooperating firewalls in a secure data network. *IEEE Transactions on Knowledge and Data Engineering*, 15(5):1307–1315, 2003.

[16] Julian Soh, Marshall Copeland, Anthony Puca, and Micheleen Harris. *Microsoft Azure: Planning, Deploying, and Managing the Cloud*. Springer, 2020.

[17] Artem Voronkov, Leonardo Horn Iwaya, Leonardo A Martucci, and Stefan Lindskog. Systematic literature review on usability of firewall configuration. *ACM Computing Surveys (CSUR)*, 50(6):1–35, 2017.

[18] John Wack, Ken Cutler, and Jamie Pole. Guidelines on firewalls and firewall policy. Technical report, National Institute of Standards and Technology (NIST), 2002.

[19] Avishai Wool. A quantitative study of firewall configuration errors. *IEEE Computer*, 37(6):62–67, 2004.

Firewalls and filtering policies for small networks

Sepehr Javid
sepehr.javid@aalto.fi

Tutor: Prof. Tuomas Aura

Abstract

The increase in the number of home offices and the trend towards working from home has risen recently, requiring access to the main office infrastructures through the unsafe internet and from home. Additionally, equipped homes with smart home appliances rely on cloud services over the internet. Firewalls have been employed to provide the security of small home networks; however, with the advent of the internet of things (IoT) *devices and the home office trend, small home and office networks are increasingly exposed to the outside world, posing security threat against internal devices and company assets. On the other hand, lack of regular administration and policy update due to management restrictions in such firewalls intensifies the threat.*

We have implemented an automated testing software whose aim is to verify the firewall policies. The software is designed to analyze different types of attacks, evaluate the results, and report the successful attacks. The software also reports the possible reason of the success of the attacks, and proposes methods to prevent them.

KEYWORDS: Firewall, IoT, Policy, VPN, Server, Client, Attack

1 Introduction

In recent years, workplaces have been transferred to homes, introducing the term "home office". This new trend has required access to work resources from home and through the internet, transmitting critical information over an untrusted path [15]. To secure such information exchange, companies have adopted *virtual private network (VPN)* solutions; however, home and corporate networks are still exposed to other external attacks, such as *denial of service (DoS)* attacks. DoS is an attack where the attacker flood the target server with undesired traffic, causing disruptions to the service [14]. DoS is mainly the initial step for more severe and threatening attacks.

On the other hand, the dream of smart houses has led to utilization of IoT devices at homes. These devices, such as Google Nest Family provide security surveillance, automated heating, and sound systems [1] are constantly accessing the internet. Such constant access can further expose the home network to the internet and attacks originated from it. For example, an IoT device could be compromised, join the Mirai bot network, and act as a threat to the internet and to the home local network [7].

In order to protect the home networks from the aforementioned threats, firewalls are employed to filter the network traffic and block untrusted packets. Each firewall consists of rules, predicates, and clauses, which are called *structural entities* and determine the policy of the firewall [10]. The ability of a firewall to provide sufficient protection against the specified risks is achieved through constant professional administration and policy measurements. The existing firewalls in our home networks are commonly incorporated within the network router and may use an outdated software version. In addition, home firewalls are generally not maintained and administered by a professional. Consequently, they are vulnerable to cybersecurity hazards.

This paper aims to develop an automated software for firewall administration and policy management. The method utilizes the concept of *firewall policy coverage* to verify the correctness of each entity. In this concept, packets are the inputs and outputs of the test. The output packets are assessed against the correspondent goal of the entity to verify if the effect of the entity on the inputs were as intended [10]. As the intention for this method is not to require prior knowledge of firewalls, the input packets are generated automatically. Hwang et al. [11] propose four au-

tomatic packet generation techniques that inspires the packet generation of our software.

We will develop the automated software based on a scenario that simulates a real-world home network containing IoT devices. In this network, IoT devices are assumed to constantly access a cloud service through a VPN tunnel. The firewall will be probed by automatically generated packets and will be evaluated based on the output packets.

The rest of the paper is organized as follows. Section 2 provides deeper information about firewall policies. Section 3 describes the scenario in detail, and presents the requirements of the scenario. Section 4 provides information about the packet generation method utilized by the software. In section 5, the paper explains the possible attacks on the provided scenario, and the reason behind the success of these attacks. In Section 6, the implementation of the software is thoroughly explained, including the incorporated scripts. Section 7 concludes the paper and presents future work.

2 Firewall Policies

A firewall policy is a set of rules used to filter undesired traffic to and from the protected network [12]. Filtering can be based on the source and destination IP address, utilized protocol, application, content and used ports. Firewall policies require constant validation, update and optimization. On the other hand, managing the firewall policies are difficult because of the ever-changing nature of networks [12]. Policies can filter packets in two different ways. *stateless* and *state-full*. Stateless filtering makes decision for the packet based on the packet information and regardless of the connection state. However, state-full filtering uses the connection state corresponding to the packet to make filtering decisions [13]. One of the major use cases of state-full filtering is to filter any traffic that aims to start a connection form outside of the local network. This ensures that only a client from the local network can initiate a connection, and the incoming packets corresponding to the initiated connection. Another way to achieve the mentioned goal is full cone network address translation (NAT) protocol. Full cone NAT maps the internal private IP address network and port to an external public IP address and port [9]. This behavior of the NAT requires every traffic from the internet to be related to a connection previously initiated by a client on the local network. Such

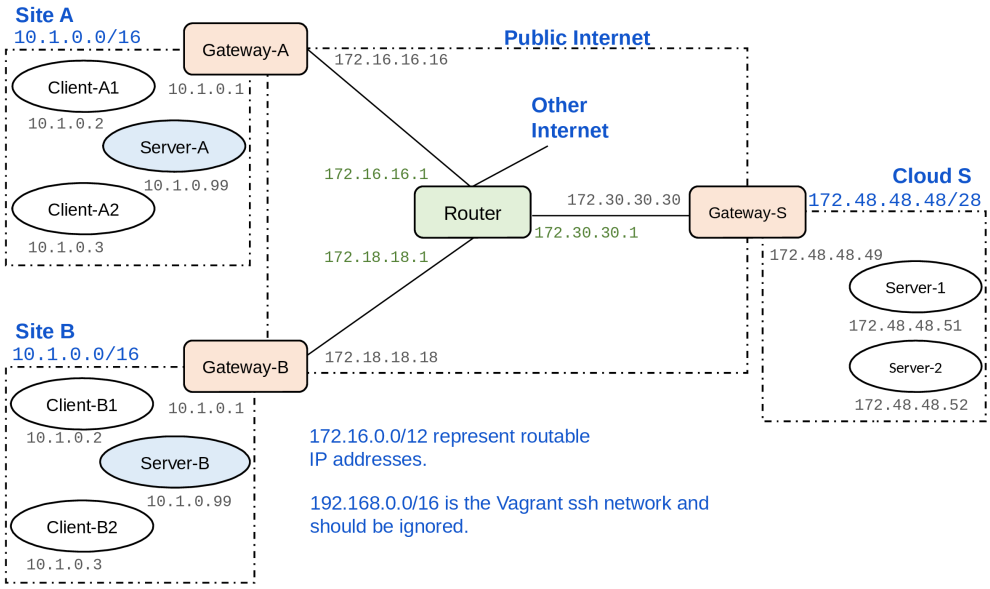


Figure 1. Utilized scenario [6].

functionality is close to the state-full filtering behavior of a firewall.

3 Scenario and Goals

This paper uses a scenario simulating a real-world environment (Figure 1). In this environment, there are two different sites, each containing several IoT devices which rely on a service provided by a cloud server. The firewalls involved in such scenario must fulfill the following requirements:

- The services dedicated to each site must be exclusive to that site. This means that a client from one site cannot be served by the server dedicated to the other site.

- A cloud server may not be reached from the internet, unless there exists a host on the internet which is explicitly reported as trusted.

- The clients cannot access the local networks of the other client sites.

- The clients should receive a response from the cloud server if and only if the client has initiated the connection. This implies that a server cannot send any packets to the client unless there has been a request from the client initially. This requirement expects the firewall to support state-full filtering.

- A client cannot be reached from the internet [6].

To achieve these requirements, the firewalls must block the undesired traffic. The firewall on each site can use full cone NAT to ensure that the connection to the cloud server is initiated by the IoT devices on the local network. The alternative way to achieve the same objective is to use statefull firewall policies on the client site firewalls. The firewall protecting the server site must block any traffic from the internet except traffic from the white-listed devices. Ultimately, the client site and cloud firewalls should establish a VPN connection, and allow all the corresponding traffic [6]. The VPN connection ensures that no attacker on the internet can spoof the IP address, and deceive the cloud server into providing the service. IP address spoofing is the act of generating an IP packet with a false origin IP address to deceive the destination about the source address, and impersonate a desired source [2].

4 Automated Packet Generation

The developed software uses a configuration file which contains a summary of the network addresses and the number of clients and server sites. This configuration is utilized by different components of the software to detect the target firewalls to probe. The file also declares the white-listed devices on the internet that each firewall can allow. Additionally, in order to fulfill the mentioned requirements, this paper proposes deploying the software on all of the devices existing in the topology in order to initiate an attack from any possible source. This possibility provides the flexibility for the software to perform a wide range of attacks. The main responsibility of the software is to send packets from different sources to the destination firewall to determine the correctness of the policy. Each probing packet must use different protocols to cover the entire test cases.

5 Possible Attacks

The developed software probes the firewall using two main attack sources. Attacks from the internet and cross site attacks. These attacks are performed by different scripts from different nodes of the topology. Each script collects the cloud server addresses containing the IP address and

the port on which the service is running, by accessing the configuration file to analyze the access to the servers. Additionally, it is possible that the cloud servers run in the same physical server, however different containers. A container is a unit of software that holds the code and the dependencies of an application, and runs separately from other applications on the same server [3]. In this case, the IP addresses of both cloud services in the specified topology are the same. However, the port numbers are different to distinguish the services. Therefore, the script has to analyze the access to both servers to probe the firewall in between.

5.1 Attacks from the Internet

These attacks are performed from the router node in the topology to imitate an attacker listening to the path on the internet.

Accessibility from the internet
In this attack, the router node tries to access the cloud servers using TCP and ICMP protocols. ICMP protocol is used to probe the Accessibility of the physical server from the internet. Accessibility of the cloud servers using ICMP protocol can lead to other serious attacks, such as, ICMP tunneling, smurf attack, etc [4]. TCP protocol is utilized to determine if a TCP connection can be established with the server from the internet. Availability of TCP connections from the internet can provide the environment for attacks, such as, TCP SYN flooding attacks [8].

Spoofing attack from the internet
Another attack from the internet is to initiate a TCP connection using IP address spoofing, and impersonating the client sites. Ideally, because of the required VPN connection between the client site and the server site, such attack should not succeed. The reason is that the VPN on the server site requires the packets sourced from the client sites to only arrive from the VPN tunnel. Therefore, it blocks the packets with the source addresses of the clients that enter from the outside of the tunnel. As a result, the server site drops the spoofed TCP packets by the attacker.

5.2 Cross-site Attacks

These attack imitate a case where one client site is compromised. The source of these attacks in this topology is the client gateways. Based on the requirements, the script should ensure that each client site cannot

reach the service of the other client sites.

Accessibility from the opposite client site
To execute this attack, a client from one site targets the service of the opposite client site. The client tries to open a TCP connection to receive service from the server of the opposite client site. According to the requirements, the service should drop such packet, or the packet should be redirected to the corresponding service instead of the service of the opposite site.

6 Implementation

The proposed software by this paper aims to perform the aforementioned attacks using different scripts. Each script contains the attacks from a specific source and initiates them upon execution. The software, including the scripts, uses python3 as the programming language. In order to generate spoofed and original packets, the scripts , utilize *Scapy* tool. Scapy is a packet manipulation program which is able to forge or decode packets with different protocols [5]. Scapy also provides a tool to capture packets traveling through a node. Each script provides the desired destination and source IP address to the Scapy interfaces, and selects the necessary protocol to generate the packet. In the case where the script sends a spoofed packet, the response reaches the victim instead of the node from which the attack initiated. Therefore, the attacker cannot receive and evaluate the response packet. As a solution, the router node uses Scapy to capture the response packets, and inform the attacking script about the captured packets.

The main goal of the topology is that the firewalls allow the clients to receive service from the cloud servers. Therefore, the script must verify that despite the firewall policies, the main functionality is not corrupted. As a result, a script must be executed from the clients to confirm that the clients can successfully receive service from the servers.

Each script is executed from a different node in the topology. Therefore, there needs to be a main script to orchestrate the execution of these components. Since the entire topology is a virtual scenario [6], the main script can run on the host. This script first calls a *secure shell (SSH)* command. This command connects the host to a desired node in the topology. Later, the main script executes the script related to the attack to be ini-

tiated from the connected node. Additionally, the main script connects to the cloud servers to initiate the the services, in order to host the attacks. Ultimately, each executed script creates a report file in a shared directory where the main script can access. The main script collects these reports and presents the final output to the user. The final report consists of the attacks that succeeded and a brief explanation of why such attack was successful. In addition, the report proposes possible solutions to prevent the successful attacks.

7 Conclusion and Future Work

In this paper, we propose a software to probe firewalls in a frequent topology. The main idea of the topology is to provide a protected path for the IoT clients to request for service from the cloud servers. The developed software can verify safety of the path by probing the firewalls on the path and performing attacks from different sources. It can be utilized in an environment where there is SSH access to a device on the path, and the client site gateways. The output of the software is a report consisting of a list of succeeded attacks, the reason behind the possibility of the attack, and available solution to prevent the attack. In the future, we plan to consider adding a wider range of attacks to initiate, for more precise verification. In addition, we consider using other tools to determine all the open ports and accessible IP addresses in the topology to use as the target of attacks. This helps the software provide a more detailed report as an output.

References

[1] Google nest. Accessed 2022-02-02. https://store.google.com/us/category/connected_home.

[2] Ip address spoofing. Accessed 2022-04-09. https://en.wikipedia.org/wiki/IP$_{a}ddress_{s}poofing.$

[3] What is a container? Accessed 2022-04-09. https://www.docker.com/resources/what-container/.

[4] Icmp attacks. Accessed 2022-04-10. https://resources.infosecinstitute.com/topic/icmp-attacks/.

[5] Scapy. Accessed 2022-04-10. https://scapy.net/.

[6] Tuomas Aura. Accessed 2022-03-05. https://github.com/tuomaura/cs-e4300$_{t}estbed.$

[7] Christoph Haar and Erik Buchmann. FANE: A Firewall Appliance for the Smart

Home. In *Federated Conference on Computer Science and Information Systems (FedCSIS)*, volume 18, page 449–458, Sep 2019.

[8] W. Eddy. TCP SYN Flooding Attacks and Common Mitigations. RFC 4987, August 2007. https://www.hjp.at/doc/rfc/rfc4987.html.

[9] dynamicsoft C. Huitema R. Mahy J. Rosenberg, J. Weinberger. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC 3489, The Internet Engineering Task Force, March 2003. https://www.ietf.org/rfc/rfc3489.txt.

[10] Fei Chen Alex X. Liu JeeHyun Hwang, Tao Xie. *Systematic Structural Testing of Firewall Policies*. PhD thesis, North Carolina State University, Michigan State University, Dec 2008.

[11] Fei Chen Alex X. Liu JeeHyun Hwang, Tao Xie. *Systematic Structural Testing of Firewall Policies*. PhD thesis, North Carolina State University, Michigan State University, Jan 2012.

[12] K. Golnabi, R.K. Min, L. Khan, E. Al-Shaer. Analysis of Firewall Policy Rules Using Data Mining Techniques. In *2006 IEEE/IFIP Network Operations and Management Symposium NOMS 2006*, Oct 2006.

[13] Thawatchai Chomsiri, Xiangjian He, Priyadarsi Nanda, Zhiyuan Tan. A Stateful Mechanism for the Tree-Rule Firewall. In *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 122–129, Sep 2014.

[14] Zouheir Trabelsi, Safaa Zeidan. Resilence of Network Stateful Firewalls against Emerging DoS Attacks: A Case Study of the BlackNurse Attack. In *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8, Nov 2019.

[15] Alina Škiljić. Cybersecurity and remote working: Croatia's (non-)response to increased cyber threats. 1:51–61, Oct 2020.

Optimizing packet classification in firewalls and routers

Anastasia Safargalieva
anastasia.safargalieva@aalto.fi

Tutor: Tuomas Aura

Abstract

Packet classification helps to manage the network traffic by using specified rules. The algorithms for packet classification consist of complex data structures which require considerable time and computing resources. It is an issue because packet filtering needs to be executed in the real-time. In this paper we consider packet classification problem as an algorithmic problem. We review HiCuts and HyperCuts algorithms to solve packet classification problem.

KEYWORDS: packet classification, firewalls, HiCuts, HyperCuts

1 Introduction

Packet classification is a building block for many services in the computer network. In this paper, we examine packet classification as an algorithmic problem and observe that there are generic algorithmic solutions across that can be used for a broad range of applications. The methods of packet classification concern the tradeoff between the speed needed for real-time packet classification and availability of the computing resources.

A literature survey of the packet classification problem was conducted, explaining the principle behind some key algorithms for implementing

the packet classifiers.

In more detail, the paper has the following goals. Firstly,to understand the applications and requirements for packet classification. Secondly, to find efficient algorithms from the literature with focus on firewalls and packet filtering. Thirdly, to demonstrate the algorithms with examples.

The work is organized as follows. Section 2 presents the background of the problem. It introduces routers and firewalls and explains the need for packet classification. In the Section 3, we examine packet classification as an algorithmic problem. Section 4 concerns packet classification algorithms – HiCuts and HyperCuts. Section 5 concludes the paper.

2 Background

2.1 Routers and firewalls

Routers are the intermediaries in terms of communication over the Internet. They are connected to the end-points through the network-level protocol – Internet Protocol (IP). The data which users exchange on the network is divided into segments known as packets. The IP routers are responsible for many Internet services, for example, forwarding a packet from the source to the destination. Another service provided by the routers is limiting a particular category of the traffic to prevent unwanted access. Load balancing to distribute the incoming traffic evenly across multiple servers is also managed by IP routers.

In addition to packet filtering in routers, there are specialized firewall appliances whose only purpose is packet filtering. They forward packets between two networks and apply a filtering policy to drop unwanted packets.

Firewalls are at the gates of the network infrastructures, they allow the traffic to leave and enter the network only according to the predefined policies [4, 7]. This is possible with a set of rules listed in the tables. A collection of rules is called a classifier. With the growth of the network system, these tables increase in complexity. The main goal is to make as few rules as possible without lowering or even increasing the security state of the infrastructure [12]. There are different types of algorithms to tackle such challenging problem as the tradeoff between time and memory consumption of the computing algorithms [5, 12, 7, 10].

There exist different types of firewalls. Stateful inspection firewalls allow packet transmission between two earlier established endpoints. They control the way packets are transmitted and their connection status. However, these stateful firewalls are designed to be generic so that they can be used in nearly any environment [6]. To provide control for more specific applications, there is application layer firewall which is the most secure type of firewall. In this case, the client and the destination server never interacts. When there is a request to process the packet, the application proxy initiates its own request to the destination server. The destination server sends the result of the request back to the proxy. The proxy communicates back to the client on the behalf of the server. The packet is inspected in this case entirely [11].

2.2 Need for packet classification

In order to provide different services to a variety of applications, IP router has packet classification and filtering functions merged into it [2]. Packet classification algorithms differentiate packets into different flows according to the content of the packet. Packets from the same flow will be processed similarly. There are defined rules for each type of incoming traffic. Arriving packets are matched against the rules until the first matching rule is found [4]. It helps service providers to identify and isolate packets from some users and provide service to others [2, 1]. Packet classification is a fundamental issue in computer networking. It is considered to be a building block for firewalls. Efficiency of the work of the firewalls plays an important role in the overall network performance [5].

Packet classification is needed for filtering policies in routers. The algorithm defines which packets are allowed to reach the destination address, and which packets have to be dropped. Denial of Service (DoS) attacks can be prevented by limiting the number of packets or bytes in each traffic class. The packets are dropped when a specified maximum rate is exceeded.

Another example of the service where packet classification is needed is Quality of Service (QoS) and traffic prioritization. Firewalls allow to classify the traffic in the groups based on its final purpose and apply specific QoS requirements of different applications to these groups. The requirements could be faster forwarding of the packets or lower probability of the packet being dropped because of the lack of the buffer resources. Deep packet inspection (DPI) is one of the technologies that help to inspect and

classify the arriving data. But this method produces delays in the services because each packet needs to be inspected. In addition, DPI takes a lot of computing resources of the controller. Another method for traffic classification uses machine learning. It is more efficient in terms of time and computational resources.

Packet classification helps to minimize the cost of the packet delivery when there is a tradeoff between faster and lower cost routes. Another reason for packet classification is that is helps to define which packets to forward through the encrypted tunnels. By configuring IPsec policies, route-based VPNs regulate which traffic can be sent through the encrypted channels.

In all these applications of packet classification, the packet header fields are first matched against the rules and then action specified by the matching rule is taken.

3 Packet classification as an algorithmic problem

Packet classification defines the flow to which one or the other packet belongs to. It allows the firewall to process a stream of packets of one flow in the same manner. For example, if the packets have the same source and destination IP addresses, they can be assigned to one flow based on the packet header [2].

There are d components in each rule in the classifier. Each component represents destination and source addresses, ports and a protocol field. For example, the packet comes from source address 192.142.55.45, destination address 192.171.110.23 with destination port 80 and source port 1800, TCP protocol field. It can be represented as the following combination of headers: $(192.142.55.45, 192.171.110.23, 80, 1800, TCP)$. Packet classification problem is to find the rule for the packet as fast as possible with least memory demanded.

The set of rules is denoted as $R = R_1, R_2, ..., R_N$, $k = 1, 2, ..., N$. $R_k[i]$ is the i-th component, or a header field of rule R_k. It is commonly agreed that rules are listed in the order of priority. For example, R_1 has the highest priority and R_N has the lowest priority. P stands for packets. A packet P matches the rule R_k, if each field of the header of P matches the corresponding field of $R_k[i]$ [1, 3].

In firewalls, the cost is the priority order of the rules. The priority order of rules in the firewall policy is an essential part of the security policy

Table 1. Examples of rules.

Rule	D. Address	S. Address	D. Port	S. Port	Protocol
R_1	201.15.17.21	201.15.75.4	=80	=1800	TCP
R_2	110*	1*	=80	=1024	TCP
R_3	201.18.20.25	201.15.100.10	>80	<1024	TCP
R_4	201.18.20.25	[100-192]*	>80	<1800	TCP

definition, and the highest-priority matching rule must be applied to the packet. On the other hand, when packets are classified for quality-of-service and economic reasons, an algorithm that approximates the lowest cost may be acceptable. Cost function can represent almost anything, for example, how many resources it would take to process the packet [1].

There can be four different types of matches between rules and header fields of the packets. The first type is an exact match when the values of rule fields and packet fields must be the same. Exact match rules can be represented as Rule 1 in the Table 1. Prefix match means that the rule field value should be a prefix of the header field of the packet. Rule 2 is an example of prefix match for the destination and source addresses. Range match specifies the range of the packet header values. Rule 3 is an example of the range match for the ports and there is an exact match for the addresses. Regular expression match requires the header field of the packet match the string expression specified in the rule. Rule 4 is an example of the regular expression for the source address. There is specification on the rule what should be done with the packet after classification, e.g., how to process the packet, does it have to be accepted or denied, encrypted or decrypted [1, 3].

In order to analyze the efficiency of the algorithms for packet classification, there are such metrics as the speed of the classification and the amount of memory the algorithm consumes. The search speed is important as the bandwidth of IP networks is growing and hence also the performance requirements of router and firewalls. The ideal situation for the memory consumption metric is when the data structures used by the algorithm take as little memory as possible. It allows to implement fast memory technologies like static random access memory (SRAM). Another critical feature of the algorithm efficiency is how fast the data structure is updated when new rules appear, or when the rules get obsolescent [2, 1].

4 Packet classification algorithms

The solutions can be divided into linear algorithms, geometric approaches, heuristic algorithms, and hardware-specific search algorithms. Heuristic solutions are based on decision-tree approaches.

Work of the packet classification algorithms is divided into two stages: preprocessing stage and classification stage. The preprocessing stage starts with building the most suitable data structure according to the rules. The goal is to build such a decision tree that each packet matches the right action at the lowest computational cost. The classification stage is responsible for distinguishing the values of the headers of the packets and finding the best matching rule in the tree.

The easiest data structure that can be applied to packet classification problem is a linked list of rules. Each packet goes through the sequence of rules until headers of the packet find the appropriate rule. This algorithm is effective in terms of memory requirements. The disadvantage of this approach is that time grows linearly during the classification process [1].

The packet classification problem is similar to the problem of point location problem in a multi-dimensional geometric space. Fields in the packet header stand for the dimensions, and the packet is viewed as a point in hyperspace. Rules are hypercubes.

Throughput, power, and memory size are the characteristics by which we can assess the efficiency of the scheme used in packet classification.

The following section overviews the existing algorithms of packet classification on multiple fields. Section 4.1 focuses on the HiCuts algorithm. Section 4.2 describes HyperCuts. Section 4.3 covers potential advances in the technology of packet classification.

4.1 HiCuts

Hierarchical intelligent cuttings (HiCuts) is an algorithm based on the decision tree approach. There is one root node. Internal nodes contain enough information to guide the packet to match with the best leaf node. Each leaf contains a single rule or a small set of rules. In the latter case, linear search method is applied to find the best matching rule.

The cutting starts from the root node. The root node represents the entire d-dimensional space. Child nodes appear by cutting the root node. HiCuts divides the multi-dimensional rule space evenly at each step across one dimension at a time. The process of cutting the child nodes continues

Table 2. Classifier.

	F_1	F_2
R_1	00*	00*
R_2	0*	01*
R_3	0*	0*
R_4	10*	10*
R_5	11*	10*
R_6	11*	11*
R_7	0*	10*
R_8	*	11*

until the number of rules in each leaf node reaches the threshold. Then these cuts form subtrees and distinct leaves in the end. The characteristics of the tree are defined while preprocessing the classifier. The main disadvantage of the HiCuts algorithm is significant rule replication where some of the child nodes can be identical. This requires more memory allocation [8, 9].

Let us examine following example of the classifier with two fields (Table 2). The threshold is set at up to 2 rules per child node and at most 4 child nodes per root node. We make 3 cuts along dimension F_1: A, B and C. These 3 cuts give us 4 child nodes. Another cut D along dimension F_2 (Fig.1) gives us 4 new nodes containing 2 rules each. The number of rules under each leaf node satisfies the chosen threshold two. Therefore, the cuttings can be stopped. The following tree (Fig.2) was build from the geometric representation of HiCuts.

4.2 HyperCuts

HyperCuts is the successor of HiCuts. It is also based on the decision-tree structure. This algorithm makes cuttings across many dimensions at a time. This gives advantage in both memory and decision time. This algorithm cuts the rules into one dimension, which reduces the depth of the tree and redundancy of the rules in it. If HyperCuts sees the same rules common to many branches, it puts these rules into one parent node. This helps to manage the redundancy of rules in the firewalls and, therefore, addresses memory problems [9, 8]. One of the main drawbacks of cutting in several dimensions at the same time is growth in the storage

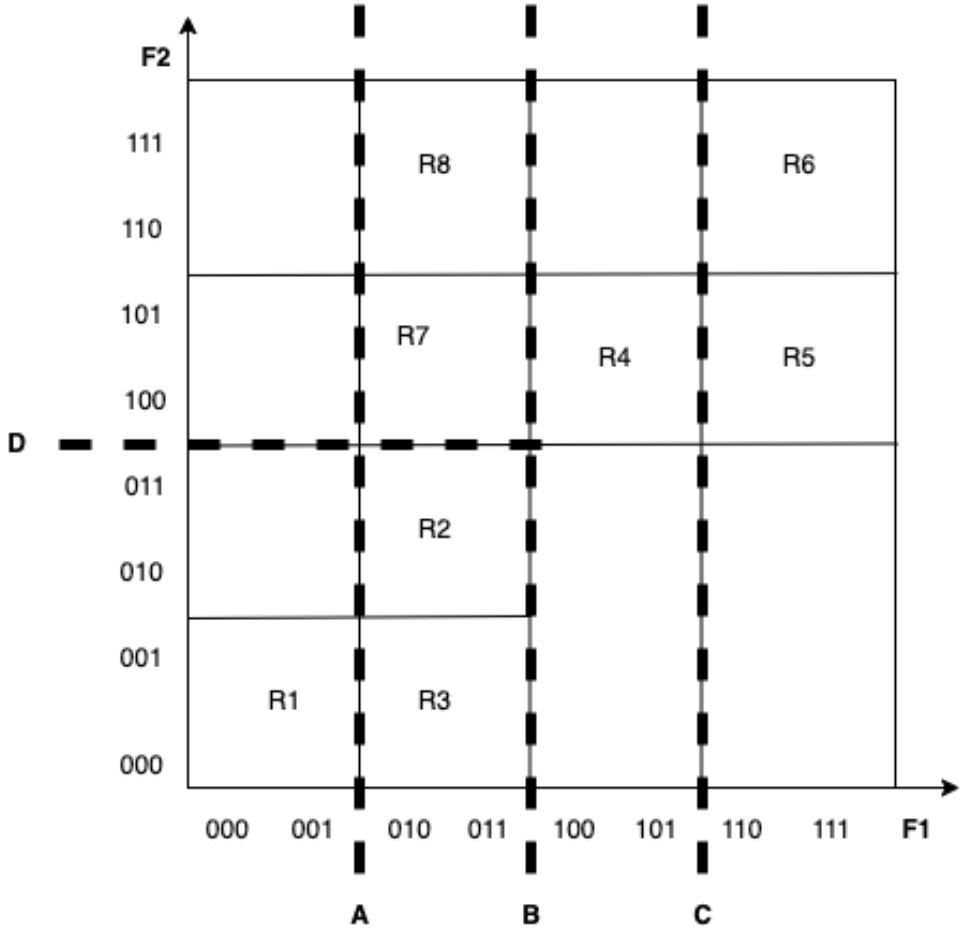


Figure 1. Geometric representation of cuttings for HiCuts and HyperCuts

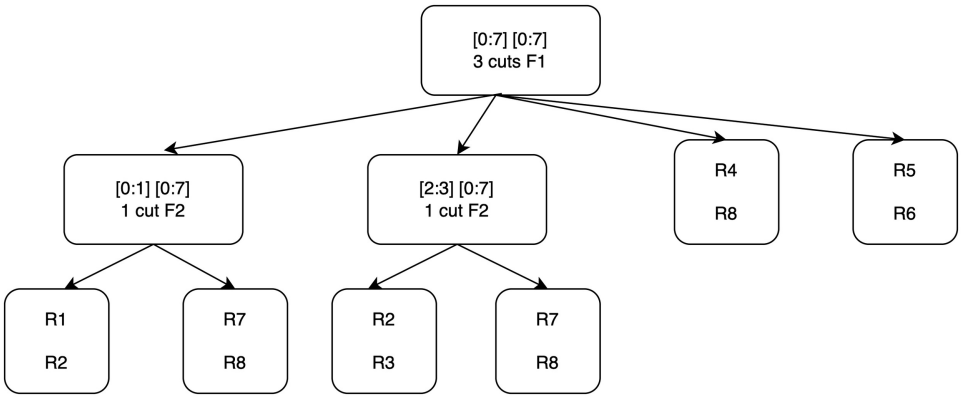


Figure 2. HiCuts decision tree

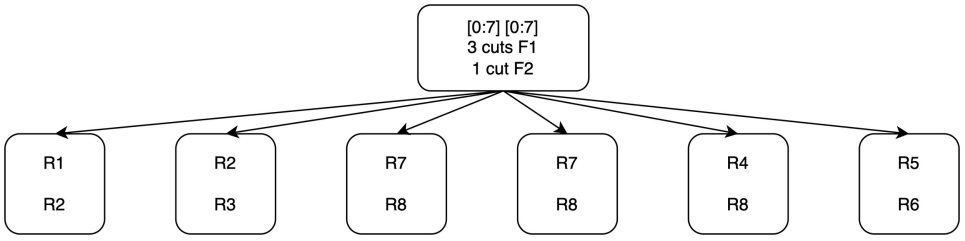


Figure 3. HyperCuts decision tree

requirements.

As an example, let us choose the same geometric representation of the cuttings from Fig.1. The only difference is that the cuts A, B, C and D are made at the same time. Decision tree for the HyperCuts contains one level of leaf nodes (Fig.3). The work of the HyperCuts algorithm can be described as follows:

1. Identify the set of dimensions for splitting. We need the most uniform distribution of rules.

2. Decide on the number of splits in each of the dimensions. For this, the mean number of the rules in the child nodes helps. After this, we can run the following algorithm to get the matching rule for the packet.

Data: Arriving packet headers in the binary form
Result: Rule
current leaf node = 0; **for** *all headers* **do**
  if $H_1 \not\subset currentleafnode$ **then**
    current leaf node++;
  else
    return rule from the current leaf node;
  end
end

Algorithm 1: Algorithm for matching rules

4.3 Future improvements

However, HiCuts and HyperCuts are known for memory overhead because of the exhaustive number of rules. Another more sophisticated algorithm EffiCuts avoids these problems. EffiCuts creates separated trees with different subsets of rules. It merges the resulting trees to optimize the running time of the search. Applying equi-dense cuts, we get evenly distributed rules in the child nodes. [9, 12]. Another improvement can be achieved by applying deep reinforcement learning (RL) methods to the problem of packet classification [5].

5 Conclusion

This paper reviewed the problem of packet classification and its applications in IP networks. We explained two packet classification algorithms: HiCuts and HyperCuts. HiCuts method makes one cut at a time. HyperCuts makes the cuttings in all dimensions simultaneously, decreasing the time needed to process the packet filtering. These approaches are good for the classifiers with many fields, as they show better trade-off between memory and speed.

While the algorithms were originally designed for firewalls, they can be used in other applications of packet classifiers, such as QoS routing and IPsec policies. Packet classification problem remains actual because the existing methods are complex and hard to optimize. The presented algorithms can be further optimized based on the knowledge of the application and the type of traffic in the network.

References

[1] Karthik Ramasamy Deep Medhi. Network routing: Algorithms, protocols, and architectures, chapter 15. Morgan Kaufmann Publishers, 2nd edition, 2018.

[2] P. Gupta and N. McKeown. Classifying packets with hierarchical intelligent cuttings. *IEEE Micro*, 20(1):34–41, 2000.

[3] Pankaj Gupta. Algorithms for routing lookups and packet classification. Master's thesis, Stanford University, December 2000.

[4] Tihomir Katic and Predrag Pale. Optimization of firewall rules. In *2007 29th International Conference on Information Technology Interfaces*, pages 685–690. IEEE, 2007.

[5] Eric Liang, Hang Zhu, Xin Jin, and Ion Stoica. Neural packet classification. arXiv, 2019.

[6] Derrick Rountree. Security for Microsoft Windows System Administrators. Syngress, Boston, 2011.

[7] Arjun Singh, Divyanshi Singh, Arun Kumar Singh, Harikesh Pandey, and P. C. Vashist. Security through optimization techniques of firewall rule sets. In *2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, pages 452–455. IEEE, 2020.

[8] Sumeet Singh, Florin Baboescu, George Varghese, and Jia Wang. Packet classification using multidimensional cutting. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, page 213–224, New York, NY, USA, 2003. Association for Computing Machinery.

[9] Balajee Vamanan, Gwendolyn Voskuilen, and T. N. Vijaykumar. EffiCuts: Optimizing packet classification for memory and throughput. volume 40, page 207–218, New York, NY, USA, Aug 2010. Association for Computing Machinery.

[10] Chenghong Wang, Donghong Zhang, Hualin Lu, Jing Zhao, Zhenyu Zhang, and Zheng Zheng. An experimental study on firewall performance: Dive into the bottleneck for firewall effectiveness. In *2014 10th International Conference on Information Assurance and Security*, pages 71–76. IEEE, 2014.

[11] Brad Woodberg. Configuring juniper networks netscreen and SSG firewalls. Syngress, Burlington, 2007.

[12] Sorrachai Yingchareonthawornchai, James Daly, Alex X. Liu, and Eric Torng. A sorted-partitioning approach to fast and scalable dynamic packet classification. *IEEE/ACM Transactions on Networking*, 26(4):1907–1920, 2018.

Formal verification of distributed systems

Juan Pablo Valencia Gómez
juanpablo.valenciagomez@aalto.fi

Tutor: Lachlan Gunn

Abstract

As software systems grow in complexity, they become harder to verify, increasing the probability of failures. Formal verification methods seek to ease the verification of complex systems with mathematical rigor. Since distributed systems are notoriously complex, this paper analyzes how formal verification methods are applied to different kinds of distributed systems, such as distributed algorithms, cryptographic protocols and their implementations. As a result, this paper finds that formal verification methods are being used with success in cryptographic protocols and implementations, while in distributed algorithms their application has been more limited.

KEYWORDS: formal verification, formal methods, distributed systems, cryptographic protocols

1 Introduction

Software systems are present in every aspect of current-day society, from transport to healthcare, banking, entertainment, commerce, and even human interaction, to name a few. Their reliable operation is of large societal importance [5], since their failure or malfunctioning can have serious

consequences.

Testing and peer reviewing are the most used techniques for software verification in the industry [5]. However, as systems start to grow in scale, they become increasingly complex. Complexity increases the probability of human error in the design and implementation of systems, and while the mentioned verification techniques remain necessary, they may not be sufficient [10].

Formal methods aim to reduce and ease the verification efforts while increasing their coverage, by establishing the correctness of software (and hardware) systems with mathematical rigor [5]. Baier and Katonen [5] point out that formal modeling of a system often leads to the discovery of ambiguities and inconsistencies in its informal specification, even before the implementation process of the system has started. Improvements in the underlying algorithms, optimization techniques, and the increasing availability of computing capacity, have allowed to apply formal verification methods in real life systems [5, 10].

Distributed systems imply multiple challenges, including maintaining consistency, dealing with network issues, such as delayed or dropped messages, and being tolerant to node faults. The interaction between components, and overall behavior of the system, is often too complex to analyze, leading to software being deployed into the real world containing critical bugs [14]. Formal verification techniques can help minimize the occurrence of such errors.

This paper aims to review and compare the formal verification methods applied to different kinds of distributed systems, such as distributed algorithms [8, 10], cryptographic protocols [7, 12] and their implementations [15]. It is organized as follows. Section 2 provides background on distributed systems, and section 3 presents different formal verification techniques. Section 4 shows how formal verification methods are used in distributed systems, and section 5 provides analysis and discussion. Section 6 concludes the paper.

2 Background

Van Steen and Tanenbaum [13] present the definition and properties of distributed systems. In principle, a distributed system is composed of multiple independent *computing elements*. A computing element, or node, can be either a hardware device or a software process. Nodes need to

collaborate in order to achieve their common goals. The way this is accomplished in practice is by exchanging messages with each other [13].

Cryptographic protocols (and their implementations) fall into this definition of distributed systems: they involve multiple participants with common goals (e.g., establishing a secret key) that communicate through message passing. Formal verification methods can be applied to verify whether the protocol meets the desired security properties under the presence of an adversary.

In distributed algorithms goals can include fault tolerance, consistency or availability, for example. Formal verification methods can be used to verify whether these properties are achieved even on the event of network or node failures.

3 Formal verification techniques

3.1 Model checking

Model checking is a formal method based on creating a model to describe the system behavior in a mathematically precise manner, together with algorithms that explore all possible states of the model [5]. The objective of model checking is to verify that the system satisfies certain properties. For this, it is required to have a precise statement of the properties as well. As in the modeling process, this step often leads to the discovery of inconsistencies in the informal specification of the system [5].

The model checker explores all possible system states to verify whether they satisfy the desired properties. If a state that violates a property is found, the model checker returns a counter-example that shows how the system goes from the initial state to this particular one [5].

Although model checking is a powerful tool, it has some drawbacks [5]. One of them is that, in practice, it may be difficult to determine if the formalization of the system (model and properties) is an accurate description of the real system. This is called the *validation problem*. Another drawback is that, for many practical systems, the state space may be extremely large. Despite the development of techniques to combat this problem, models of realistic systems may still exceed the available computer memory. This is known as the *state-space explosion problem*.

3.2 Theorem provers

Theorem provers (also called proof assistants) are computer programs that aid in the proof of theorems [6]. One of the most prominent examples of automated theorem proving was the verification of the proof of the Four Color Theorem, performed using the Coq theorem prover.

Coq allows the user to formalize mathematical concepts and then assists them in generating machine-checked proofs of theorems [1]. The architecture of Coq consists of two levels: a small kernel, consisting of a few primitive constructions and a few logical rules of inference, and on top of the kernel a rich environment to help design proofs [6, 11]. In this way, it is only necessary to trust the kernel, instead of the entire environment, since ultimately any definition and proof is checked by the kernel [1, 11].

Although not its main application, Coq is well suited for software verification due to its powerful language that includes functional programming and high-level specifications [11]. A great example of this is CompCert [9], a compiler for the C programming language that is formally verified using Coq. This means that it is formally proven that the executable code produced by CompCert will behave exactly as specified by the semantics of the C program.

Coq also supports extraction of verified programs to different programming languages, such as OCaml and Haskell, allowing to create verified software libraries [1]. Others have also expanded upon the program extraction capabilities of Coq. For instance, Verdi [14] is a framework based on Coq for specifying, implementing and formally verifying distributed systems.

Verdi provides two key components that make it specifically suited for distributed systems: *network semantics* and *verified system transformers* (VST). Network semantics encode different network behaviors, for example, reordering, duplicating or dropping packets. The idea is that a programmer initially implements and proves the system under an ideal network model. Then, a VST transforms the application to work under a new network semantic, and additionally generates a proof that the new system preserves the properties of the previous one. This effectively transforms the initial application into a new version that tolerates faults [14].

Benchmarking found that a verified version of a program achieves comparable performance to an unverified one [14]. Wilcox et al. [14] conclude that Verdi is a promising first step towards the goal of easing the imple-

mentation of verified distributed systems.

There are other proof assistants, such as Isabelle [3], which can also be used for the formal verification of software systems. Isabelle provides code generation capabilities as well, allowing to extract specifications into executable code in languages such as OCaml, Haskell and Scala [3].

3.3 F* and KaRaMeL

F* is a programming language designed to support formal verification [2, 15], with syntax similar to functional programming languages, such as OCaml and F#. After verification, programs can be extracted to C via the KaRaMeL compiler (formerly known as KReMLin) [15]. This enables to formally verify realistic applications. The resulting C program can be compiled to verified machine code using CompCert. However, code compiled with CompCert is not yet as fast as that compiled with standard compilers, such as Clang or GCC [15].

3.4 Tamarin

Tamarin is a tool for symbolic modeling and analysis of security protocols [4]. Its specification language allows to construct detailed models of the protocols, their security requirements, and the capabilities of the adversary. Then, it can be used to produce a proof that the protocol fulfills the desired properties, even when taking into account the actions of the adversary [4]. Alternatively, it may return a counterexample, meaning that it found an attack that violates a property.

Since the correctness of security protocols is an undecidable problem [4], Tamarin may not terminate on a given input. In this case, users can resort to an interactive mode, which allows them to explore the proof states and inspect attack graphs, effectively combining automated proof search and manual proof assistance [4].

4 Applications

This section presents several examples of formal verification methods applied to real life distributed systems.

4.1 Distributed algorithms

Amazon Web Services

Amazon Web Services (AWS) offers a considerable amount of cloud services for different purposes. They aim to make these services simple for customers to use, but this external simplicity is supported by internal complexity: the services rely on distributed algorithms to achieve different goals, such as fault tolerance, consistency or auto-scaling [10].

Newcombe et al. [10] describe how AWS has adopted TLA+, a formal specification language, with the objective of minimizing the occurrence of critical bugs in their systems. They state that formal methods are a big success at AWS, and their adoption has been increasing across different teams within the company [10].

TLA+ is used both for describing the desired correctness properties of the system and for designing the system itself. For this purpose, they also model the *operating environment* of the system. This means that different events are specified, such as network and disk errors, process crashes or data-center failures [10]. Then, the model checker verifies that the specification maintains the desired properties despite any combination of events in the operating environment.

The two main benefits of the adoption of TLA+ in AWS are the ability to quickly verify whether proposed changes (even deep ones) are safe, and the possibility to test innovative performance optimizations [10]. An additional benefit is that the formal specification of the system serves as an excellent form of documentation.

Newcombe et al. also address the *validation problem*, that is, the lack of certainty that the executable code developed by engineers correctly implements the formal specification. They report that they have not found any tool capable of verifying the executable code of distributed systems as large and complex as those being built at Amazon [10]. Despite this, formal methods are still helpful in multiple ways at the company:

- They help get the design right. This is very important because, given a flawed design, the engineers are unlikely to spot the flaws while focused on implementing it.

- They help engineers gain a better understanding of the design, which increases the chances of getting the code right.

Byzantine Paxos

Lamport [8] presents a formal proof of correctness for a Byzantine variant of the Paxos algorithm. Paxos a distributed consensus algorithm, that is, an algorithm that enables multiple nodes to agree on a value. Moreover, it is fault tolerant: $2f + 1$ nodes can reach a consensus tolerating the failure of any f of them [8]. In the Byzantine variant, $3f + 1$ nodes are used and it can tolerate f of them being malicious [8]. The proof was performed using TLA+.

4.2 Cryptographic protocols

This section presents examples of two different approaches for using formal verification methods in cryptographic protocols. In one of them, they are used to prove that the protocol satisfies the claimed security properties. On the other one, they are used to find vulnerabilities in the protocol.

TLS 1.3

Transport Layer Security (TLS) is the main protocol for secure communications on the internet [7]. Multiple attacks have been found over the years in TLS versions 1.2 and below, resulting in protocol modifications to provide security enhancements or increased functionality. For the next version of the protocol (TLS 1.3), the IETF an adopted an "analysis-prior-to-deployment" philosophy [7]. In this context, Cremers et al. [7] perform a symbolic analysis on a release candidate for TLS version 1.3, to verify its claimed security properties.

The analysis was performed using Tamarin, and modeled six out of the eight fundamental properties that the TLS handshake protocol must satisfy according to the specification. It concluded that, in general, TLS 1.3 meets the specified security properties [7]. However, it also found an unexpected behavior, which may have security implications: in the post-handshake client authentication, the client does not receive explicit confirmation that the server has successfully received the client response.

Bluetooth pairing, EAP-NOOB and DPP

Peltonen et al. [12] present a study on different misbinding attacks against protocols for secure device pairing and bootstrapping. In a misbinding attack, malicious behavior of one protocol participant causes another participant to be confused about the identity of their communication counterpart [12].

Secure device pairing is seeks to establish a secure wireless communication channel between two devices [12]. Device bootstrapping refers to protocols for registering Internet-of-Things (IoT) devices to an online server [12]. Although these two kinds of protocols are considerably different, they are similar in the sense that the identity of the devices is defined by physical access to them. The lack of verifiable device identifiers results in the protocols being vulnerable to misbinding attacks [12].

The study analyzes Bluetooth pairing, and two protocols for device bootstrapping (EAP-NOOB and DPP). The protocols and their security requirements were modeled with a tool called ProVerif. In all the cases, ProVerif returned counter-examples, that is, execution traces that violated the security property, meaning that a misbinding attack was found. In the case of Bluetooth pairing, five different variants of misbinding attacks were identified [12].

Peltonen et al. [12] conclude that the impact of the misbinding attacks is relatively marginal compared to the advantage of using encryption and authentication. However, warn that protocol designers should understand the misbinding vulnerability and make an informed judgment about whether additional countermeasures are needed [12].

4.3 Cryptographic implementations

The HACL library*

Cryptographic libraries are the core of secure communications on the internet, and therefore are held to high correctness, robustness and security standards [15]. Bugs in these libraries have historically been found by manual inspection and testing. A more robust approach is to use formal verification to prove the absence of multiple kinds of errors.

Zinzindohoué et al. [15] present HACL*, a verified library that implements modern cryptographic primitives: the ChaCha20 and Salsa20 encryption algorithms, Poly1305 and HMAC message authentication, SHA-256 and SHA-512 hash functions, the Curve25519 elliptic curve, and Ed25519 signatures. The following properties are verified for each primitive [15]: memory safety, functional correctness, and secret independence (an important property to mitigate against side-channel attacks).

HACL* is written on the F* language and compiled to C via the KaRaMeL compiler [15]. Zinzindohoué et al. [15] mention that, at the moment of writing, they used GCC to compile the C code for performance reasons,

but expect that CompCert will become faster over time. Benchmarking yielded that, for most primitives, the HACL* implementations are at least as fast as the most efficient implementations in other libraries [15].

HACL* is used as the main cryptographic library of the miTLS project, a verified implementation of TLS. It is also being integrated in Mozilla's NSS cryptographic library, which is used by the Firefox browser [15].

5 Analysis

Formal methods are a very powerful tool for designing systems and verifying that they meet their desired properties. Complex systems can benefit greatly from this, resulting in less error-prone implementations. Formal verification also provides a way to test the safety of proposed changes, when it might be difficult to estimate beforehand all the implications that they might bring.

Different kinds of distributed systems use formal methods in different ways. While in some cases, e.g. cryptographic protocol analysis, they are used for the design or verification of the protocol, in other cases, such as cryptographic implementations, they are also used to verify the executable code libraries.

In distributed algorithms, however, it is not yet possible to apply formal verification methods to working code, as noted by the article by Amazon engineers [10]. Verdi shows promise in this regard, but it is still on its early stages and requires further development.

Another thing we can observe from the presented examples is that there exists a wide variety of libraries, frameworks and tools for formal verification. To recall only the ones mentioned in this paper: Coq, Isabelle, TLA+, ProVerif, F*, and Tamarin. While having different options is positive, it might also make it harder to choose which tool to use for a given application.

In the case of software systems, we know that Amazon uses formal methods. It is possible that other "tech giants" use them too. However, their usage is not widespread in the software industry, especially not in medium and small sized companies. At this point it is unclear if the usage of formal methods will ever become standard practice in the software development industry.

Two challenges for adopting formal verification methods in software companies are: integrating the verification into the development process

in a way that does not cause disruption to the developer workflow, and the need for training the engineers in the chosen formal verification tool.

Nevertheless, the clear benefits of formal methods make them an essential component for the verification of critical safety applications, complex systems of all kinds, and, in particular, distributed systems. It is expected that their usage in these areas will continue to increase in the coming years.

6 Conclusions

This paper has reviewed and compared how formal verification methods are applied to different kinds of distributed systems, such as distributed algorithms, cryptographic protocols and their implementations. First, different methods were presented, including model checking, theorem provers, among others. Then, real life applications were analyzed.

This paper finds that formal verification methods are being used with success in cryptographic protocols and implementations, being an important tool for modern cryptography. In the field of distributed algorithms, however, their application is more limited. Formal methods have been used to model and design systems, but verification of working implementations is still on its early stages.

References

[1] Coq reference manual. `https://coq.github.io/doc/v8.15/refman/`. [Online; accessed 24-February-2022].

[2] F* introduction. `http://fstar-lang.org/#introduction`. [Online; accessed 31-March-2022].

[3] Isabelle overview. `https://isabelle.in.tum.de/overview.html`. [Online; accessed 24-February-2022].

[4] Tamarin user manual. `https://tamarin-prover.github.io/manual/book/001_introduction.html`. [Online; accessed 07-April-2022].

[5] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[6] Alan Bundy. Automated theorem provers: a practical tool for the working mathematician? *Annals of Mathematics and Artificial Intelligence*, 61(1):3, 2011.

[7] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In *Pro-*

ceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, pages 1773–1788. Association for Computing Machinery, 2017.

[8] Leslie Lamport. Byzantizing Paxos by refinement. In *Distributed Computing*, pages 211–224. Springer Berlin Heidelberg, 2011.

[9] Xavier Leroy, Sandrine Blazy, Daniel Kästner, Bernhard Schommer, Markus Pister, and Christian Ferdinand. CompCert - A Formally Verified Optimizing Compiler. In *ERTS 2016: Embedded Real Time Software and Systems, 8th European Congress*, 2016.

[10] Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, and Michael Deardeuff. How Amazon Web Services uses formal methods. *Communications of the ACM*, 58(4):66–73, 2015.

[11] Christine Paulin-Mohring. *Introduction to the Coq proof-assistant for practical software verification*, pages 45–95. Springer Berlin Heidelberg, 2012.

[12] Aleksi Peltonen, Mohit Sethi, and Tuomas Aura. Formal verification of misbinding attacks on secure device pairing and bootstrapping. *Journal of Information Security and Applications*, Volume 51, 2020.

[13] Maarten Van Steen and Andrew S. Tanenbaum. *Distributed systems*. Maarten van Steen Leiden, The Netherlands, 2017.

[14] James R. Wilcox, Doug Woos, Pavel Panchekha, Zachary Tatlock, Xi Wang, Michael D. Ernst, and Thomas Anderson. Verdi: A framework for implementing and formally verifying distributed systems. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 357–368. Association for Computing Machinery, 2015.

[15] Jean-Karim Zinzindohoué, Karthikeyan Bhargavan, Jonathan Protzenko, and Benjamin Beurdouche. HACL*: A verified modern cryptographic library. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1789–1806. Association for Computing Machinery, 2017.

Theoretical Framework for Cloud Computing Network Measurements

Santeri Sipilä
santeri.sipila@aalto.fi

Tutor: Corneo Lorenzo

Abstract

The global pandemic has affected the world in multiple ways including the traffic changes of the internet. Measuring and analyzing this traffic may help to improve the creation of future cloud computing solutions. This paper researches recent studies on the measurements of cloud computing and summarizes them. These studies include research on the effects of the pandemic on the internet, reachability of cloud data centers and measuring cloud providers off net deployments. In addition, this paper discusses the different methods of improving reachability of cloud datacenters and their drawbacks. Edge computing, using sophisticated routing protocols and using private WAN are all discussed on.

Keywords: Cloud Computing, Data Center, RIPE Atlas, Networks, Network Measurements,

1 Introduction

The global pandemic has affected the world in multiple ways. One of the affected areas includes the variations in the traffic of the internet [5] [14]. This traffic is often directed toward cloud data centers which have been adopted globally by many different corporations and institutions in the re-

cent decade. These data centers manage high amounts of traffic through the internet while serving many different types of applications. Measuring the traffic of these centers that create and receive it is required to analyze the effectiveness and to plan new applications of the data centers.

Many tools are available on the market, which allows for analyzing the latencies, the traffic, and the surrounding networks of these data centers. These tools are either software, hardware, or a combination of both. For example, a common tool used for analyzing networks is the Ripe Atlas platform [2], which combines both hardware and software to create a platform that can measure network performance from many different locations.

Multiple recent studies have been published on measuring different network statistics of the cloud. However, recent studies on summarizing and analyzing these measurements have not been published. This paper studies the current tools available for cloud computing network measurements and looks into recent studies that examine trends in networks surrounding cloud computing. In addition, this paper studies improving the reachability of cloud data centers. These are accomplished by reviewing research on the matter.

The structure of this paper is as follows. Section 2 presents the common metrics that are used in measuring networks, as well as some of the standard measurement tools that are commonly used. The third section overviews recent research that has been conducted on the matter of network measurements. The fourth section analyzes the use cases of these studies and examines how the reachability of cloud data centers could be improved. Finally, the fifth section concludes the findings of this paper.

2 Network traffic measurements

This chapter reviews the basic concepts of network measurements, common network measurement tools, the validity of these measurements and cloud data centers.

2.1 Network measurement metrics

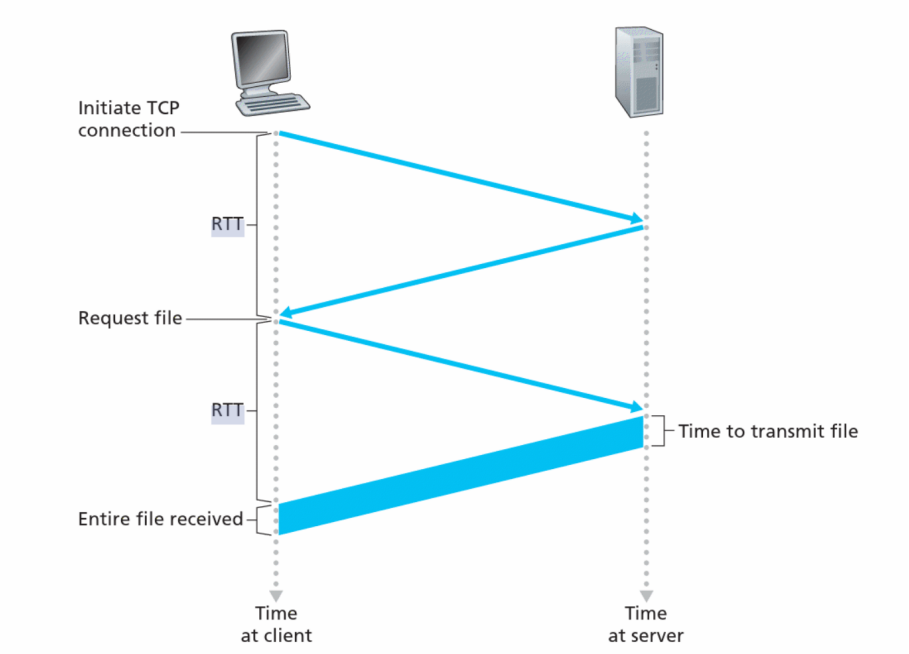


Figure 1. A visualisation of the Round Trip Time (RTT) of a packet.(J. Kurose & K. Ross 2013 [16])

There are many quantities to measure in a network, but the most commonly used metrics are connectivity, latency, bandwidth, throughput, and packet loss [16]. Connectivity is a binary metric that indicates if two endpoints are connected in a network or not. The time it takes for a packet to travel from one point to another and back is known as latency or the Round Trip Time (RTT) [16]. A visualization of the RTT is presented above in Figure 1.

Bandwidth is the maximum capacity of data flow in an endpoint, link or network [16]. Throughput is the measure of how much traffic is going through a link or a network at one time [16]. Finally, Packet loss is the number of packets lost while trying to send packets from one endpoint to another. Although there are many quantities to measure, this paper mainly studies the connectivity, latency, and bandwidth of cloud data centers and networks.

2.2 Common network measurement tools

There are many common measurement tools made for personal computers. One of the most common tools is the `ping` command built into many operating systems, such as the Linux operating system [3]. This tool can be used to measure the connectivity, the RTT, and the packet loss ratio

from the host computer to the endpoint. Another tool embedded in the Linux operating system is the `traceroute` command [4]. This command allows the host machine to see the route that the package takes while traveling to a certain endpoint in the network. The `traceroute` command shows then the RTTs of each hop within this network.

Network measurements can also be done via a probe in a network. There are many types of probes available on the market, but this paper focuses mainly on the uses of the RIPE Atlas probe and the platform that it uses [2]. The probe collects various types of network measurements, such as ping, traceroute, SSL/TLS, DNS, NTP, and HTTP measurements, and sends these to the RIPE Atlas NCC platform [2]. Data from multiple (12 000 connected [1]) probes are collected on to this platform. This data can then be filtered, analyzed, and exported on the platform.

2.3 Ways of measuring network traffic

Network measurements methods are currently divided into two main classes [10]. The first class is passive measuring. In this way of measuring, the inserted probe does not generate any additional traffic to the network but only "listens" for ongoing traffic. Data is collected by placing measurement devices in the desired network locations. For example, measuring the throughput of a single link via a probe would be passive measuring.

The second class of network measurement is active measuring. This type of measuring is performed by a probe that sends probe packets to the network and then measures different metrics of these packets, such as the RTT. The commonly used tools ping [3] and traceroute [4] are both examples of active measuring.

2.4 Cloud data centers and edge computing

Cloud data centers, also known as data centers are buildings that have multiple interconnected and co-located servers within them [9]. Their computing power consists of nodes, which can be bare-metal servers or virtual machines. Data centers can be used for high-performance computing, such as scientific computing or acting as a distributed server. Often, this is achieved by parallel processing the data over multiple nodes in the data center.

Cloud data centers can utilize edge computing. Edge computing is a method of computing, which distributes the components of data centers to

a wider area. For example, a data center can distribute standalone servers that host some of their commonly used applications widely throughout a region. The benefits of using edge computing are discussed in chapter 4.

Cloud data centers are often provided by content hypergiants, such as Google, Amazon, and Facebook. Data centers can offer multiple different kinds of services to clients, such as Infrastructure-as-a-Service (IaaS), Platforms-as-a-Service (PaaS), Software-as-a-Service (SaaS), and Function-as-a-Service (FaaS) [9].

3 Measurements of cloud datacenters

This section explores previous research done on the measurements of cloud data centers. The first part presents the current situation with the reachability of data centers. The effects of the global pandemic in cloud data centers are then evaluated after. Finally, the validity of the RIPE Atlas platform is evaluated.

3.1 Reachability of cloud datacenters

A key factor in cloud data centers is the reachability of these centers from a client perspective. Some applications may require a low latency while using, such as cloud gaming. One option to measure the reachability would be to place probes globally and measure the latencies to data centers from these probes. Corneo et al. [12] conducted a study on the matter, in which they measured the average latencies to ten major cloud data networks globally, using over 8500 distributed ripe atlas probes [2] over 12 months. The study concluded that the majority of the world's population has access to a cloud data center within a latency of 100 ms.

Although a latency of 100 ms is viable in many applications, such as hosting websites, there are still many cases where this is not fast enough. Autonomous cars, cloud gaming, and augmented reality all need faster latencies. Potential reachability improvements are presented in chapter 4.

3.2 Measuring the off-net deployments of cloud providers

Content hypergiants, such as Facebook, Google, and Netflix, have been increasingly moving towards deploying services and servers inside end-user networks. In addition, these hypergiants often encrypt the traffic that

they use. This makes it possible to publicly analyze the amount and the locations of these off-net services by firstly scanning the default HTTPS port 443 and then retrieving the Transport Layer Security (TLS) certificates. These certificates can then be used to determine the entity that hosts that service [13]. A study, conducted by Gigis et al. proposed this method and used it to measure the growth of off-net deployments by four content hypergiants, Facebook, Google Netflix, and Akamai from 2013 to 2021 [13]. The study concluded that these hypergiants have more than doubled the amount of their off-net services during the period between 2013 and 2014. The growth of off-net deployments differs depending on the region. In recent years, the growth of these deployments has been rapid in Europe, Asia, and Latin America.

3.3 Effects of the pandemic in cloud computing

The global covid-19 pandemic not only affected humans but also the networking traffic as a result. A study conducted by Feldmann et al. [5] researched the effects of lockdowns on the bandwidth usage and destinations of network traffic. It was found that the pandemic increased traffic usage by over 200% in certain applications, such as VPN and video conferencing applications, while the traffic in local networks, such as university networks was decreased [5]. This kind of traffic could increase the latency to cloud data centers since increased traffic may lead to increased delays in the service. For example, latencies may increase in situations where the educational platform is hosted in the same data center as other services. Despite these increased loads, the study also concluded that the Internet managed to cope well despite having to manage rapid load variations in the network traffic.

Similar results were found in a study conducted by Favale et al., which analyzed the traffic variations from the perspective of the Politecnico di Torino campus [14]. It was observed that incoming traffic drastically decreased, while outgoing traffic increased and the usage of remote desktop environments and VPN services increased.

3.4 Validity of measurements

This paper refers to several studies that have been conducted using the RIPE Atlas platform [2]. Thus, it is useful to know if the platform has valid measurements. A study has been conducted on the subject by Bajpai

et al [15]. The paper studied the challenges with the Ripe Atlas platform. The paper concludes that the atlas platform has some weaknesses.

The first problem is that the first and the second version (v1, v2) of probes are not always accurate in latency measurements [15]. This however is only a small percentage of probes since Ripe atlas does not distribute v1 and v2 probes anymore. Measurements can also be done without the first and second-generation probes, leaving only valid measurements.

The RIPE Atlas platform may also suffer from an inherent sampling bias. Probes are set up often by enthusiasts that may have more complex home networks than a regular user of the internet [15]. This should be accounted for in the measurements, especially in the areas where the amount of nodes is not large [15].

The RIPE Atlas platform has created many new traceroute vantage points. This has resulted in extended coverage of the internet. However, these vantage points can be restrictive in terms of resources. This can be a problem when studying a network with many traceroutes over a long period, since new network coverage creates stale traceroute results. This could partially be prevented by using BGP updates as signals to monitor overlaps in the traceroute corpus [8].

4 Meaurement usages

This section explores the ways of improving the reachability of cloud data centers and combines information from previous studies to indicate key metrics that should be accounted for to measure cloud-based systems.

4.1 Improving the reachability of cloud datacenters

The reduction of latency in cloud data centers is a major focus of improvement. This reduction allows for faster services in addition to an entirely new genre of cloud-based real-time applications, such as augmented reality-based applications. Thus, it is in the interest of major service providers to reduce the latency.

The latencies that are measured consist of multiple different delays. One major part of this delay is the route that the packet takes. One of the most common routing protocols for packets is the Border Gateway Protocol (BGP). Although this protocol is widely used, it has been proven sev-

eral times to be inefficient, meaning that the protocol does not always find the route with the lowest possible latency between endpoints [7]. Large content providers, such as Facebook and Google have already been trying to solve this problem with their routing protocols. However, a study conducted by A. Todd et al [7], concluded that using more sophisticated routing protocols only proves a minor reduction in these latencies despite BGP offering possibly a slower route. A drawback of not using BGP is that it is expensive to create a new routing protocol when compared to BGP.

Another way of improving latency could be to use a private Wide Area Network (WAN). Using a private WAN can improve latencies since large content providers can build a private network and determine what kind of protocols and equipment they want to use inside them instead of relying on the possibly inefficient public internet. These private WANs are then connected to the public internet via Points of Presence (PoP). This method allows the cloud providers to bypass large parts of the public internet, thus allowing for better control and possibly better performance. A study conducted by Arnold et al. [6] studied the possible improvements of private WAN. The study concluded that WAN can often improve latency. However, these improvements are highly dependent on the geographical locations of the endpoint and the data center, the distance between them, and the traffic that occurs in between. A private WAN does not automatically improve the reachability, since it requires better paths and routing policies to outperform the public internet. The study states that Google's and Amazons's private WAN solutions offer better performance than the public internet for most users, but they could still be improved.

Latency could also be reduced by bringing data centers closer to the end-user. This would mean creating new data centers. This could improve the latency, especially in remote regions where the distance creates a large portion of the latency. This would reduce the distance between end-users and data centers. However, data centers are costly and require large amounts of power to operate. Economically it would not make sense to create large data centers in places with few users. The solution to this problem might be edge computing. Edge computing would bring cloud computing resources closer to the end-users by creating smaller less expensive cloud computing facilities more widely, which in turn reduces latency to the end-user. However, this would result in notable improvements only in regions where the latency is already high. For example, according to Corneo, et al. study [11], edge computing would only result

in reductions of around 2 ms in areas with already low latencies to cloud datacenters. This way of reducing latency is more viable in regions, where the latency to the cloud is already high, such as Asia, Latin America, and Africa [11].

All of these methods of improving reachability have their areas of improvement and drawbacks. The table, presented below, summarizes the benefits and drawbacks of each of these methods of improving reachability.

Improvement method	**Possible improvements**	**Drawbacks**
Using more performance aware routing protocols instead of BGP [7]	Performance aware routing can improve latency by 5+ms in 2-4%of cases where BGP routing is used.	Creating a new performance aware routing protocol is expensive when compared to using BGP.
Using a private wan [6]	Can provide improved latency in cases where the private WAN outperforms the public internet.	Building a private WAN can be expensive and does not always guarantee better performance than the public Internet.
Using edge computing [11]	Edge computing could improve latencies to cloud data centers by up to 30% in certain cases.	Latency gains fall when the distance from the client to the data center gets smaller, thus making it less effective in some cases.

Table 1. Summarized methods of improving the reachability of cloud data centers

4.2 Summary of cloud computing measurements

Measuring a traditional server/service has been straightforward since often they have been deployed to a single location. However, with cloud computing these measurements become increasingly complex since they can be distributed globally with different kinds of equipment resulting in different results depending on the measurement location. Recent studies have proven that there are many good tools to analyze the cloud, but there is no silver bullet for measuring the cloud. Instead, it requires careful planning to measure different aspects of the cloud successfully.

One of the key metrics to analyze in cloud computing systems is to measure the reachability of the system. Corneo et al. presented a framework for this using the RIPE Atlas platform, measuring the reachability of certain cloud data centers [12].

The expansion of cloud computing resources can be useful to measure planning networks. Measuring can be achieved by using the method of retrieving and analyzing HTTPS certificates from these cloud providers,

proposed by P. Gigis et al [13].

Analyzing the traffic and its changes in cloud computing can be used to analyze different events on the planet, such as wars and pandemics. In addition, they can be used to plan networks. There are many ways of measuring the traffic of the cloud. The cloud is such a large concept that it requires lots of data points to analyze. One solution is to analyze data from vantage points, Internet Service providers, and edge locations, as proposed by Feldmann et al [5].

5 Conclusion

Measuring different properties of the cloud is a key point in creating new technologies that utilize the cloud. This paper reviewed the recent studies that have been conducted on cloud computing measurements and summarized the key factors of measuring the cloud. This paper also reviewed some of the methods of improving the reachability of the cloud.

There is no silver bullet for measuring cloud computing. The nature of distribution in the cloud will always make cloud computing measurements complex. The result is that measurements of the cloud should be planned in a controlled way while taking into account all of the variables that the measurements may have. Measuring the cloud also has different properties when compared to a regular service

Improving cloud reachability also suffers from a similar problem. Many of the improvements are highly dependable on the network and devices, which in turn means that improvements should be planned and tested before fully committing to a way of improvement.

Due to the nature of cloud computing, there is a lot of possible research to conduct. One possible future research topic could be to research if it is possible to lower latencies by implementing new laws/regulations. For example, requiring new nodes to have a certain minimum processing capacity. Another possible future research topic could be to research the increase of traffic to a single cloud service provider over time and how it correlates to the distribution of the cloud.

References

[1] Global ripe atlas network coverage. https://atlas.ripe.net/results/maps/network-coverage/. Accessed: 15.2.2022.

[2] What is ripe atlas? https://atlas.ripe.net/about/. Accessed: 15.2.2022.

[3] GNU coreutils. *ping(8) Linux User's Manual*, 8.32 edition, March 2020. Accessed: 1.2.2022.

[4] GNU coreutils. *traceroute(8) Linux User's Manual*, 8.32 edition, March 2020. Accessed: 1.2.2022.

[5] A. Feldmann et al. The lockdown effect: Implications of the covid-19 pandemic on internet traffic. IMC '20: Proceedings of the ACM Internet Measurement Conference, pages 1–18, 2020.

[6] A. Todd et al. (how much) does a private wan improve cloud performance? In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 79–88, 2020.

[7] A.Todd et al. Beating bgp is harder than we thought. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, HotNets '19, page 9–16, New York, NY, USA, 2019. Association for Computing Machinery.

[8] G. Vasileios et al. Reduce, reuse, recycle: Repurposing existing measurements to identify stale traceroutes. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 247–265, New York, NY, USA, 2020. Association for Computing Machinery.

[9] L. Barroso et al. *The Datacenter as a Computer: Designing Warehouse-Scale Machines, Third Edition*. Morgan Claypool, 2018.

[10] L. Chengmin et al. Analysis and research of network measurement technologies. In *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things*, pages 117–121, 2015.

[11] L. Corneo et al. (how much) can edge computing change network latency? 2021 IFIP Networking Conference (IFIP Networking), 2021.

[12] L. Corneo et al. Surrounded by the clouds: A comprehensive cloud reachability study. WWW '21: Proceedings of the Web Conference 2021, pages 295–304, 2021.

[13] P. Gigis et al. Seven years in the life of hypergiants' off-nets. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 516–533, New York, NY, USA, 2021. Association for Computing Machinery.

[14] T. Favale et al. Campus traffic and e-learning during covid-19 pandemic. *Computer Networks*, 176:107290, 2020.

[15] V. Bajpai et al. Lessons learned from using the ripe atlas platform for measurement research. *ACM SIGCOMM Computer Communication Review*, 45(3):35–42, 2015.

[16] J. Kurose and K. Ross. *Computer Networking: A Top-Down Approach: International Edition*. Pearson, 2013.

julianjessen.howardbaker@aalto.fi

Biometric authentication: a survey of different modalities and their potential use cases.

Julian Jessen Howard Baker

Tutor: Sanna Suoranta

Abstract

This seminar paper examines the biometric authentication landscape. It focuses on creating a broad overview of numerous methods and their applications. After an evaluation of survey papers and articles, it summarises some notable unimodal, multimodal and continuous multimodal biometric authentication methods and concludes that, given the literature continuous multimodal biometric authentication is the most relevant method. Especially given the pervasive and mobile state of technology in the present world. Although this is mostly the case, there are still various significant use cases for unimodal and multimodal methods, so it is not as simple as which is better, the context of use is important.

key words: *Biometric authentication, authentication, Enrollment, Authentication, Templates, Continuous biometric authentication*

1 Introduction

This section aims to introduce the concept of biometric authentication, its importance, relevant advantages and disadvantages, introduce relevant literature on the topic and state the direction of the paper

Alphanumeric passwords have been and still are the most common method of user authentication despite their high cognitive load and susceptibility to compromisation such as password database attacks and spoofing [16]. Passwords have served as an efficient alternative to manual checks in more archaic access control systems and enabled the scaling of many different services that could not have been possible manually.

Due to user familiarity, alphanumeric passwords are user-friendly and natural to use although often lead to reduced security for the user. In many cases, to reduce the mental burden of remembering many passwords for different places, users will choose short, memorable (common) passwords or store them in insecure locations. This presents a significant attack avenue for capable, malicious actors.

With the advent of password managers and encrypted physical password devices, the password has retained its usefulness to an extent. However, new, reliable authentication methods, e.g, biometric authentication, have been developed over the years that have either extended the functionality of the password or replaced it entirely. Biometric authentication is increasing in popularity not solely due to increased security and performance. It has the advantage of being efficient to use, non-intrusive and has been shown to be the preferred method of identity verification by general smartphone users [16].

Biometric authentication is the process of using a physical, behavioural, or vital biological characteristic to verify an identity [7].It typically consists of two stages, enrollment and verification. Enrollment involves capturing the particular characteristic(s) through a sensor on the device, extracting features of that characteristic, quantifying it, producing and storing a template of the characteristic somewhere safe. The latter stage consists of comparing a current sensor recording to a template previously stored on the system, and then making a acceptance or rejection decision based on the similarity of the current recording and the stored template [1].

2 Biometric systems: The generic architecture

To accomodate the processes of enrollment and validation, the system must possess certain modules that allow for the collection of raw biometric data [13]. The important modules include:

1. Sensor Module that captures the raw data of the trait being captured

2. Feature extractor that converts the raw data from the sensor module into a compact representation

3. Classifier module / Matching module that uses a classifier to compare the extracted feature to those residing in the local database and determine similarity

4. Decision Module Uses a threshold to determine if the similarity from the classifier module is sufficient to accept the individual. If not rejects the individual.

Many biological characteristics have properties that are especially useful for the purpose of identity verification:

1. Universality The idea that everyone has this characteristic innately, by being human. This means that everyone has a password or means to be identified,

2. Uniqueness The concept that the chosen bio characterisic is different between all potential users of the system, and

3. Permanence The lack of change of the characteristic over time. [3]

Using a singular biological characteristic to identify a user is called a unimodal biometric authentication method, whereas .using a combination of them is referred to as a multi-modal method [7]. The works [7, 2] discuss several significant, detrimental aspects of unimodal methods such as: Susceptibility to circumvention, Non-universality, and highlight that multi-modal address these aspects very well. However, some multimodal methods can incur significant cost and performances penalties [11]. Therefore depending on the requirements of the system, uni-modal methods could be more appropriate.

However, an additional point that is not widely considered in the literature is the non-retractability of a bio-metric trait, which is especially significant when using unimodal methods. If a system uses a biological characteristic to decide entry to a system and if a user's template is com-

promised, that incurs a massive usability or security tax, depending on the response taken by the system governing body. This is an especially important consideration in uni-modal systems, where if a user cannot validate with a specific characteristic, they can either not use the system or need to find another way of Authenticating themselves, such as reverting to password use, which can incur a security penalty.

This paper aims to evaluate different unimodal and multimodal methods of authentication and analyse their performance and potential use in different use case scenarios. The evaluation will address various contemporary unimodal and multimodal authentication methods and refer to the components below to model security.

1. Performance vs Usability

2. Cost of implementation

3. Overall security - False Acceptance and False rejections

4. Interoperability

3 Notable unimodal methods

This section discusses and evaluates various notable unimodal methods. Unimodal biometric authentication is still a widely used system of authentication mainly due to its simplicity of implementation, speed of computation and minimal system hardware requirements. The most common methods of unimodal biometric authentication include face, iris, hand geometry, voice and fingerprint [17]. Unimodal methods are gradually being replaced by multi-modal methods due to their susceptbility to circumvention, Non-universality, high rates of error and noisy data.

3.1 Iris based authentication

Iris based biometric authentication well regarded, and is considered to be a highly reliable authentication method. This is mainly due to low interclass similarity, making the false acceptance rate of a system using this modality very low [9]. Using the Iris also has various other benefits:

1. Unchangability with time

2. Ease of collection

3. Uniqueness

4. Large feature span

5. Non-contact and unintrusive based collection
   [4]

The iris is a protected, internal structure located within the eye. It is mainly responsible for controlling the ammount of light that reaches the retina by contracting and relaxing various muscles in the eye. The iris consists largely of complex, unique tissue where, much like the fingerprint can contain grooves, ridges, loops, bulges and spots [9]. These components contribute to the complexity and uniqueness of the iris between individuals. To optimize performance, high quality iris images are needed, which can incur significant computational hardware costs.

Evaluation of iris based unimodal biometric authentication
In the review [6], a number of different iris based unimodal biometric authentication works are evaluated. Overall, biometric authentication using the iris yields high performance. The iris as a feature is preserved and changes at a slow rate over time. Hence, the template is valid for a long period of time and does not need frequent updating compared to a feature set that is subject to change like the face. Using the Iris yields highly results even on mobile platforms [6], meaning it will remain extremely relevant with devices becoming increasingly portable.

3.2 Face based authentication

Similarly to iris based biometric authentication, use of the face for authentication purposes is extremely common. This is mainly due to the availability of high definition cameras that are able to capture regions of the face with very high accuracy. A potential problem with using the face as a main authentication feature is the lack of permanence. The face and its various features change dramatically in the process of aging. Over long periods of time, the facial template will need to change. This incurs a us-

ability cost to the user, as they will have to periodically reconstruct their facial template. This section presents some significant methods using the face as a main authentication feature.

3.3 Basic methods and structure of the face

Face based biometric authentication using soft biometrics
In the paper exploring face based biometric authentication, [10] state the use of using soft biometrics to improve the overall face-recognition accuracy. Soft biometrics are facial traits that are not distinctive on their own although can significantly aid in facial recognition. These features can include scars, moles and freckles [10]. These features are especially helpful in non-ideal situations, where the individual to be verified is off-center or in an off-frontal pose. Additionally, due to increasing quality and higher resolution images, these features can be captured with ease and is therefore a relatively effortless way of improving face based biometric authentication.

4 Multimodal methods

The introduction of multimodal biometric authentication methods helped solve issues such as spoofing attacks, non-universality and intraclass variation in unimodal methods. The characteristic that makes this possible is fusion, which is where evidence of multiple , independent features are taken from many different biological or behavioural sources via sensors, and combined into valid template. Multimodal methods can also address the problem of 'liveness detection', whereby the authentication device must try to detect if the individual to be authenticicated is actually present in that moment [13]. By requesting a random subset of biometric traits, the device can ensure that the individual is actually present. This section introduces some early methods of biometric authentication and discusses their applications in mobile contexts. The different methodologies used to combine different modalities are described and discussed. Then, some more contemporary methods are discussed and compared with their earlier multimodal and unimodal counterparts.

4.1 Multi-modal methods: Early feature fusion techniques

The work [13] discusses some early vairiations of multi-modal methods of biometric authentication. It highlights different methods of feature fusions and evaluates their performance.

The findings suggest that using multiple, independent biometric traits increase performance significantly. Arun Ross et al. state that there are various issues that unimodal methods have to contend with that reduce their effectiveness. Examples of this include: Noisy data(Improperly maintained sensor, burnt fingerprint or scarred finger), Intra-class variation (wrong pose, poor lighting) and Inter-class similarity (similarity between members from different classes. This results in false acceptance). The generic aim of the multi-modal method is to address these problems by fusing multiple independent biometric traits of an individual whilst meeting the performance requirements of the various applications using them. Arun Ross et al. introduce several main categories of feature fusion available when using multimodal methods, they include:

Single biometric trait, multiple sensors

The use of multiple sensors to retrieve the raw data of a singular biometric trait. This data can then be fused at different stages during data processing. Generally, fusion happens at one of two stages, matching or at the recording of data and feature extraction stage. In feature level fusion, the data from multiple sensors are combined into a single template. One way that this is done is by calculating the weighted average of the various data sets to produce one single template from a series of feature sets [12]. In match level fusion, the individual features sets are first passed through the match module - which determines how similar the recorded feature set is to the correct, stored feature set. After passing the multiple sensor recordings of the biometric trait through the matching module, the matching scores are combined for subsequent use in verification [12]. Although the discussion of the match level fusion techniques are outside the scope of this paper, it is easy to see the performance benefit of combining several feature sets into single numbers.

Single biometric trait, multiple classifiers

The use of a single sensor to retrieve multiple instances of the same biometric trait. This helps in creating a more complete, concrete representation of the trait. Various examples of this include Multi-instance systems

- the usage of multiple instances of the same body trait (could be various different angles of the face, recording and combining the feature set of different fingers etc.) [12] - and multi-sample systems - usage of a single sensor to acquire multiple samples of data from the same biometric trait - for example, taking multiple recordings from the same finger to obtain a more complete and accurate feature set. An early example of this the use of a multi-instance systems by J.Kittler et al. who conduct an investigation into the usage of multiple sources of information to increase accuracy of biometric authentication. In this case, multiple images of different angles of the face are utilized to increase performance over unimodal biometric authentication. They achieved a reduction in error rates of up to 40% [8].

Multiple biometric traits

Utilizing multiple sensors to record mutiple, independent biometric traits. The use of physically uncorellated traits is shown to result in high performance [12]. However, the deployment of multiple varied sensors is more expensive and there are further contextual considerations to make regarding deployment costs, computational performance and error rates [12]. D.Jagadiswary et al. introduce a multimodal biometric authentication system that fuses data at a feature level from fingerprint, finger vein and retina to produce high GAR (general acceptance rate) and low FAR (false acceptance rate) [5]. The method uses feature level fusion to turn three independent sensor readings and feature extractions into a single feature set via matrix fusion. The use of multiple biometric traits increases the the complexity and detail of the feature space, thereby reducing the overlap of features between individuals. This works to reduce false acceptance [5]. Jagadiswary et al. do not specify the computation requirements in this paper, it is also important to consider that due to their use security mechanisms such as encryption of templates, the space and processing power needed for this method are significantly higher than a unimodal method without encryption. Although, it is not clear whether these differences will be noticable or significant. What can be concluded from this paper is that high levels of performance can be achieved even with the use of encryption [13].

4.2 Continuous multimodal biometric authentication

After summarising and evaluating numerous multi-modal and uni-modal biometric authentication methods the disadvantages of both approaches are clear. Namely, that unimodal approaches, although easy to implement and have good potential for accuracy, suffer significantly when the data quality is poor (e.g dirty or wet hand) and is highly susceptible to interclass similarities which cause false acceptance [14]. And multimodal approaches have great potential for accuracy and performance [5] incur high computational and hardware costs due to multiple, highly specific sensors and simultaneous reading and combination of information from multiple sources. Furthermore, both of these static modalities have a fatal vulnerability in that post-authentication the system could be misused. The nature of static authentication being that the user is authenticated once per session (e.g the user is authenticated until they lock the screen of their phone). A solution to this is continuous biometric authentication, a method in which the identity of the user is continuously re-verified. Within continuous authentication, it has been shown that a combination of physiological biometrics is preferrable due to the ease of collection and low cost of implementation.

4.3 Continuous multimodal biometric authentication: Face recognition and keystroke dynamics

After noting that continuous multimodal biometric authentication is an advanced and relevant alternative to the previously discussed categories, it is important to introduce and evaluate a specific case to discuss its use and potential in different contexts. Stuti Srivastava et al. introduce a continuous method using facial recognition and keystroke dynamics. Keystroke dynamics is the analysis of timing information of a persons typing rhythm and habits,the data used consists of the hold time for each key (time between press and release) and latency between two successive key presses [15]. In addition, to make use of the multimodal approach, face recognition is used. Alone, this would be susceptible to noisy data and other disdvantages as dicussed previously. However, in combination with keystroke dynamics, they will serve as reliable identifiers. Compared to physiological characteristics, keystroke dynamics can be recorded non-invasively and without specialist hardware (e.g fingerprint sensors)[15]. Using weighted sum feature-level fusion, the experiment yielded a 0.0176

EER (equal error rate - the point at which the false acceptance rate and false rejection rate intersect), which is extremely accurate.

This is extremely high performance given its hardware and software requirements. The methods of data acquisition are also non-intrusive and, superficially, do not interfere with the use of the system. Something that is not really considered is when this authentication takes place. Further research could definitely consider the optimal periods to record data and perform the authentication such that it interfers as little as possible with the user experience.

5 Conclusion

This paper provides an overview of the biometric authentication space. It draws attention to the different modalities and briefly discusses their potential uses. It covers the basics of uni,multi and continuous biometric authentication methods and their potential use compared to the familiar, archaic albeit efficient passwords. It serves as a timeline, considering more dated approaches and their initial developments, starting with unimodal approaches and leading to more contemporary, continuous multimodal biometric authentication. These more contemporary approaches are especially relevant with the increasingly pervasive nature of technology, specifically smartphones. A conclusion that can be drawn from the literature is that all biometric authentication methods can potentially be useful if implemented correctly. Their potential usefulness is, however, restricted to the use context. As mentioned above, continuous multimodal biometric authenticationhas wide potential use especially in pervasive systems, where the devices will be used with a high degree of frequency. It is clear to see that in another context, authentication to this degree would be unnecessary and costly.

References

[1] Shaymaa Adnan AbdulRahman and Bilal Alhyami. A comprehensive survey on the biometric systems based on physiological and behavioural characteristics. 2021.

[2] A.Jain, A.Ross, and S.Prabhakar et al. An introduction to biometric recognition. January 2004.

[3] Juan Esteban Ordonez Bonilla. Biometric authentication. May 2021.

[4] Arezou Banitalebi Dehkordi and Syed A.R.Abu-Bakar. A review of iris recognition system. 2015.

[5] D.Jagadiswary and D.Saraswady. Biometric authentication using fused multimodal biometric. 2016.

[6] Fathima, Aleemath Sana, and Dr.M Sharmila Kumari Mohammed Hafeez M.K. A review on iris recognition systems. 2019.

[7] Ujwalla Gawande and Yogesh Golhar. Biometric security system: a rigorous review of unimodal and multimodal biometrics techniques. May 2018.

[8] J.Kitler, J.Matas, K.Jonsson, and M.U.Ramos Sanchez. Combining evidence in personal identity verification systems. 1997.

[9] Vahid Nazmdeh, Saghayegh Mortazavi, Daniel Tajeddin, Hossein Nazmdeh, and Morteza Mdarresi Asem. Iris recognition; from classic to modern approaches.

[10] Unsang Park and Anil K.Jain. Face matching and retrieval using soft biometrics. September 2010.

[11] Norman Poh, Thirimachos Bourlai, and Josef Kittler et al. Benchmarking quality-dependent and cost-sensitive score-level multimodal biometric fusion algorithms. November 2021.

[12] Arun Ross. *Encyclopedia of Biometrics*. Springer Link, 2009.

[13] Arun Ross and Anil K.Jain. Multimodal biometrics: An overview. September 2004.

[14] Riseul Ryu, Soonja Yeom, and David Herbert. Continuous multimodal biometric authentication schemes: A systematic review. 2021.

[15] Stuti Srivastava and Prem Sewak Sushish. Continuous multi-biometric user authentication fusion of face recognition and keystroke dynamics. 2016.

[16] Nina Gerber Verena Zimmermann. The password is dead, long live the pasword - a laboratory study on user perceptions of authentication schemes. August 2019.

[17] Krishnakumari y. A review on unimodal and multimodal biometric systems. May 2017.

Explainable Empirical Risk Minimization

Yifan Zhu
yifan.zhu@aalto.fi

Tutor: Alexander Jung

Abstract

The demand for explainability in machine learning (ML) predictions has been increasing with the wider application of ML techniques and the significance of the decisions from predictions. An ideal ML model should have both high predictive accuracy and explainability, whereas there is a trade-off between these two targets. As a potential solution, Explainable Empirical Risk Minimization (EERM) has been proposed to learn a predictive model by balancing its empirical risk and user-specific explainability. This paper presents the details of EERM, evaluates its performance by implementing EERM in linear hypothesis space, and compares experiment results with linear regression, ridge regression and LASSO regression. The result preliminarily shows the utility of EERM in linear hypothesis space.

KEYWORDS: empirical risk minimization, explainable machine learning, regularized loss minimization

1 Introduction

With the development of Machine Learning (ML) in recent years, ML techniques have been applied to much broader areas, including drug design [1], healthcare [2], and finance [3]. Many decisions of these appli-

cations, which might be affected by ML predictions, are crucial in that wrong decisions can cause severe outcomes [3]. Furthermore, not knowing why and how the prediction is generated makes it difficult to adjust the ML model. Thus, the demand for explainability of ML predictions is rapidly increasing, which even limits the further broader application of ML techniques.

Both predictive performance and explainability are significant indicators of ML model performance. However, for most models, there is a trade-off between these two factors [4]. With the improvement of predictive performance, the model can become more complex, then it is more difficult to interpret the model. As a result, Explainable Empirical Risk Minimization (EERM) [5] has been proposed to learn an ML model such that both predictive performance and explainability are considered. In EERM, a user's knowledge for the data is quantified by an information-theoretic measure as a "user summary" [6], and then the explainability of the model can be measured by the distance between the prediction and user summary. As an instance of Regularized Loss Minimization (RLM), EERM uses the explainability quantification as the regularization function to learn the optimal ML model, which is to have low empirical risk and to be clear for the user to understand.

To the best of our knowledge, EERM has only been formulated, and there is little further evaluation and analysis of it. This paper reviews the main features of EERM, evaluates its performance with experiments, and analyses its utility together with other RLM algorithms.

This paper is organized as follows. Section 2 introduces the basic knowledge and prerequisites of EERM. Section 3 presents the fundamentals and details of EERM. Section 4 contains the experiment details, analyses the experiment results and discusses the utility of EERM. Finally, Section 5 provides conclusions of EERM.

2 Prerequisites

In this section, we present some prerequisites for understanding EERM in detail. Section 2.1 introduces the basic concepts of ML and learning principles related to EERM. Section 2.2 elaborates on the definition and methods of explainable machine learning. Finally, Section 2.3 presents the basic knowledge of information-theoretic measures used in EERM.

2.1 Learning Principles in ML

Machine Learning is a technique that detects important patterns from existing knowledge to obtain new information [7]. For example, in the application of weather prediction, taking the weather data of the past year as input, ML algorithms can extract features, such as the correlation between weather and seasons, to predict the future weather.

Key Principle

The key principle of most ML algorithms is to train a model which builds a prediction map from the input data to the output prediction. Let $\mathcal{X} = \{\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_n\}$ denote the input space, $\mathrm{x}_i = (\mathrm{x}_i^1, \mathrm{x}_i^2, ..., \mathrm{x}_i^m)$ denote each instance in $\mathcal{X}$, which is also a feature vector , $\mathcal{Y}$ denote the output space, which is the label set for $\mathcal{X}$. Then, to learn a model is to learn a prediction map h

$$\begin{aligned} y &= h(\mathrm{x}) \\ h &\in \mathcal{H} : \mathcal{X} \to \mathcal{Y} \end{aligned} \tag{1}$$

where x is a feature vector in $\mathcal{X}$, and y is the predicted label for x. The prediction map h is referred to as a hypothesis in the hypothesis space $\mathcal{H}$ as well as a predictor.

Generalization

One of the most significant goals of ML algorithms is to learn a predictor with a strong performance in generalization. Generalization is an ability that the predictor can make predictions on unseen instances with satisfying accuracy based on the training process. This ability can be measured by a loss function $\mathcal{L}(y, h(\mathrm{x}))$, which calculates a score of error based on the distance between the true label y and the prediction $h(\mathrm{x})$. High value given by $\mathcal{L}(y, h(\mathrm{x}))$ indicates that there are many differences between the prediction result and the true label, and then the predictor has poor prediction performance. Hence, the goal of ML can be quantified as finding the optimal hypothesis h^* such that

$$h^* = \arg\min_{h \in \mathcal{H}} R(h) \tag{2}$$

where $R(h)$ denotes the generalization error of hypothesis h.

Empirical Risk Minimization

Since the true label for the unseen data is unknown, we cannot directly measure the generalization error. Then, Empirical Risk Minimization

(ERM) [8] was proposed as a learning principle to approximate generalization performance by computing the average error over the training set Γ

$$\widetilde{R}(h) = \frac{1}{|\Gamma|} \sum_{(\mathrm{x},y)\in\Gamma} \mathcal{L}(h(\mathrm{x}), y) \tag{3}$$

$\widetilde{R}(h)$ is referred to as the empirical risk of hypothesis h. With ERM, we may select the hypothesis h^* with the minimum empirical risk as the optimal predictor

$$h^* = \arg\min_{h\in\mathcal{H}} \widetilde{R}(h) \tag{4}$$

However, ERM may arise the problem of overfitting. Overfitting is a phenomenon such that the predictor with small empirical risk has high risk in the unseen data, which indicates that the predictor cannot generalize on the unseen data. This phenomenon is mainly caused by the gap between the size of training set and the complexity of hypotheses. Since the training set size is often limited, constraining complexity of hypotheses may avoid overfitting.

Regularized Loss Minimization

The learning principle Regularized Loss Minimization (RLM) was proposed to balance the empirical risk and the predictor complexity by minimizing the sum of $\widetilde{R}(h)$ and a regularization function $\mathcal{R}(h)$

$$h^* = \arg\min_{h\in\mathcal{H}} \widetilde{R}(h) + \lambda\mathcal{R}(h) \tag{5}$$

where λ is a hyperparameter. $\mathcal{R}(h)$ measures the complexity of hypothesis h. For example, Lasso regression adopts L1 norm $||\mathrm{w}||_1 = \sum_{i=1}^{n} |w_i|$ as the regularization function to constrain the number of model parameters as well as the complexity in RLM; thus we can obtain a hypothesis with both satisfying empirical risk and complexity.

2.2 Explainable Machine Learning

The growing demands for the explainability have brought increasing popularity and importance for Explainable Machine Learning. Explainable ML is defined by Murdoch [9] as extracting the relations from the learned ML model, which are either between data features and labels or learned by the predictor.

The main categories for Explainable ML methods are intrinsic and post hoc [10]. Intrinsic methods refer to ML models that are intrinsically explainable due to their simple structure or relatively low complexity, while

methods belonging to post hoc analyse the information extracted from the learned model to obtain explanations. The key issue of Intrinsic methods is the trade-off between explainability and prediction accuracy, since model with high complexity may lead to low explainability and high accuracy. EERM is then proposed by Jung [5] as an intrinsic method to learn a predictor with balanced accuracy and explainability.

2.3 Differential entropy

Entropy is a basic concept in Information theory, which quantifies the uncertainty of a random variable [11]. The uncertainty can be interpreted as the amount of information contained in the random variable. Specifically, the occurrence of a rare event can convey a large amount of information as well as high uncertainty.

Differential entropy is the entropy for continuous random variables. Conditional entropy $H(\mathrm{Y}|\mathrm{X})$ is an extension of entropy in that it quantifies the uncertainty of random variable Y give the information of X. Let Y be a continuous random variable, $\mathcal{S}_Y$ be the corresponding domain set, then $H(\mathrm{Y}|\mathrm{X})$ can be defined by

$$\begin{aligned} H(\mathrm{Y}|\mathrm{X}) &= -\int_{\mathcal{S}_X,\mathcal{S}_Y} f(x,y)\log f(y|x)dxdy \\ &= -\mathrm{E}[\log f(y|x)] \end{aligned} \tag{6}$$

where $f(x,y)$ is the joint probability density function of X and Y, $f(y|x)$ is the conditional probability distribution of X and Y.

3 Regularized Loss Minimization algorithms

In the following, we present the details of EERM algorithm, which is an instance of RLM, and classical RLM algorithms, namely ridge regression and LASSO regression. Then, in Section 4, we conduct experiments on these algorithms to compare their performances and then discuss the utility of EERM.

3.1 Explainable Empirical Risk Minimization

Explainable Empirical Risk Minimization is proposed by Jung [5] to search for the optimal hypothesis with low empirical risk and high explainability in the hypothesis space, which is intrinsically explainable. The essence of EERM is the combination of RLM and user background. EERM models

users' background, the understanding of data to be specific, as a user summary, which quantifies users' expected value for the prediction results. Then, the explainability of the predictor is measured by the conditional differential entropy between prediction results and the user summary. To learn the optimal model in the form of RLM, the empirical risk is used to quantify the model generalization performance, and the explainability quantification is used as the regularization function. The optimal model obtained from training is an optimized result of the trade-off between explainability and predictive performance, which has satisfying generalization performance and is explainable to the user.

User Summary

In Explainable ML, it is significant to notice that explainability is measured based on the context [12], which is the user's background for, for example, the application and ML techniques. Users with different backgrounds can have different level of understanding and different concerns for the predicted results. For example, a user who is familiar with linear regression can interpret the predicted results directly by the weight of each feature, which represents the significance of the feature, while a user with little knowledge of the algorithm may need a more detailed explanation for understanding the result.

To process the varying user background, EERM proposed user summary to model user's understanding for the data and features. User summary is defined as user's expected value for the predicted result of given data, which is largely based on user's intuition for the correlation between data features and labels [5]. For example, in weather prediction, the user can provide own prediction for future weather based on the observations and intuitions, such as the correlation between weather and seasons. Thus, by providing the data and user summary to the training process, users with different backgrounds can obtain a personalized explainable ML model.

Explainability Quantification

With user summary $\hat{u}$ and predicted result $\hat{y}$ for data x, the explainability of the predictor can be modeled as conditional differential entropy $H(\hat{y}|\hat{u})$, which quantifies the uncertainty of observing the predicted result while given user summary. Since user summary contains user's expectation for the prediction, high uncertainty indicates that the predicted result is largely different from user's expectation, which suggest that the predictor is out of user's understanding; thus the predictor has poor explainability.

Furthermore, if we narrow down the range of hypothesis space to linear predictors and assume that the data and its user summary follow multivariate normal distribution with zero mean, the explainability quantification can be further derived as $E[(\hat{y} - \alpha\hat{u})^2]$ [5] with the entropy of normal distribution, where α is a constant. This quantification directly measures the distance between predicted results and user summary, which is easier to understand as well as compute.

Regularization

The objective function in EERM can be obtained by combining ERM and the explainability quantification [5]

$$\begin{aligned} h^* &= \arg\min_{h\in\mathcal{H}} \widetilde{R}(h) \\ s.t.\ &H(\hat{y}|\hat{u}) \leq \eta \end{aligned} \tag{7}$$

where the entropy is constrained to be smaller than η.

In practice, if we keep the assumptions above and quantify explainability as $E[(\hat{y} - \alpha\hat{u})^2]$, then we can write (7) as [5]

$$h^* = \arg\min_{h\in\mathcal{H}} \widetilde{R}(h) + \lambda E[(\mathrm{w}^T\mathrm{x} - \alpha\hat{u})^2] \tag{8}$$

where w is the weight parameters in hypothesis h.

It is noticeable that (8) is in the form of RLM with explainability quantification as the regularization term. Clearly, during the training process, EERM algorithm will seek to find the optimal hypothesis with balanced generalization performance and explainability based on the user's specific background.

3.2 Classical Regularization Algorithms

Both ridge regression and LASSO regression are classical regularization algorithms. They share the similarity of being performed on linear hypotheses, and the major difference between them is the choice of regularization functions, which leads to the difference in their preferences for training the weight parameters.

Ridge Regression

Ridge regression is a regularization algorithm performed on linear regression with l_2 norm as the regularization function

$$h^* = \arg\min_{h\in\mathcal{H}} \lambda||\mathbf{w}||_2^2 + \sum_{i=1}^{n}(h(\mathbf{x}_i) - y_i)^2 \tag{9}$$

where λ is the regularization parameter that controls the penalty on the parameters; $||\mathbf{w}||_2$ is the l_2 norm, which equals to $\sqrt{\sum_{i=1}^{m} w_i^2}$.

Utilizing l_2 norm as the regularization function, ridge regression tends to regularize the weight parameters to approach zero with a large value of λ; thus, it is capable of decreasing model complexity.

LASSO Regression

LASSO stands for least absolute shrinkage and selection operator. Unlike ridge regression, LASSO regression uses l_1 norm as the regularization function

$$h^* = \arg\min_{h \in \mathcal{H}} \lambda ||\mathbf{w}||_1 + \sum_{i=1}^{n} (h(\mathbf{x}_i) - y_i)^2 \tag{10}$$

where $||\mathbf{w}||_1$ equals to $\sum_{i=1}^{m} |w_i|$.

From (10) we can observe that if the regularization parameter λ is chosen be a large value, then LASSO regression will penalize weight parameters of insignificant features to be equal to zero. In other words, LASSO regression is capable of performing feature selection, which leads to the similar effects of ridge regression.

4 Experiments and Results

To evaluate the performance of EERM and compare it with ridge regression as well as LASSO regression, we implemented the EERM algorithm specifically for a linear regression task; conducted experiments of these three algorithms as well as the baseline linear regression algorithm on the same dataset; presented their scores as well as learned weight parameters respectively as the experiment result. Codes for the experiments can be found in GitHub.

4.1 Dataset

The dataset we used is the California Housing dataset. It contains the housing information of 20640 census location blocks from the 1990 California census. Ten attributes of census blocks are included in the dataset, which are *longitude*, *latitude*, *the median age of housing*, *total amount of rooms*, *total amount of bedrooms*, *population*, *total amount of households*, *median income*, *distance to the ocean* and *the median house value*. The first nine attributes are used as data features, and the median house value is used as the target.

4.2 Experiment details

To ensure that the dataset is usable, we cleaned the data in advance by processing missing value and categorical value. Furthermore, to avoid data overflow in the implementation of EERM, the data is scaled by standardization. The dataset is then randomly split into training set and test set for training and evaluating the performance of the algorithms.

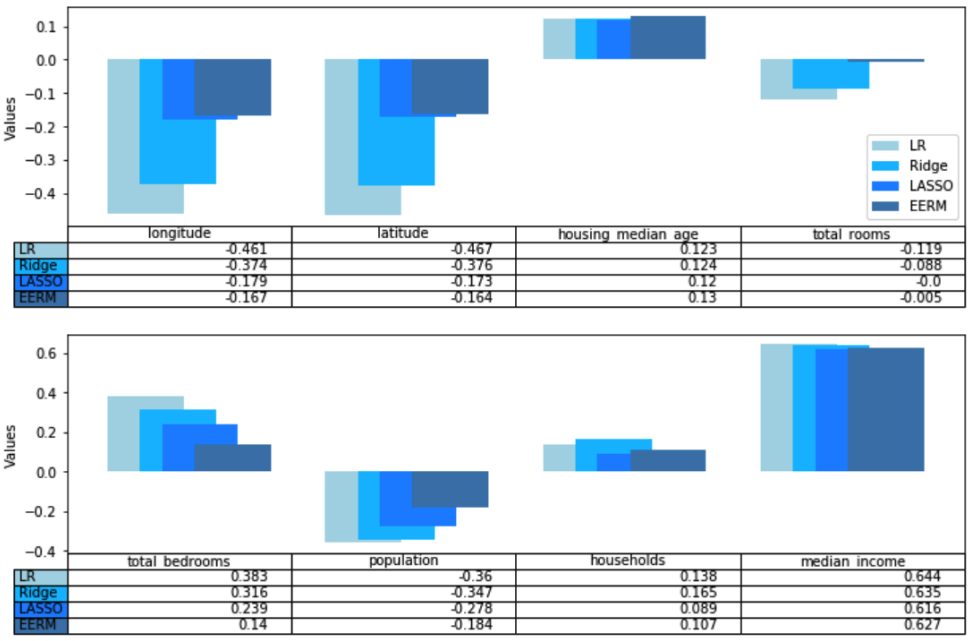
The models of Linear regression, ridge regression and LASSO regression are directly fetched from scikit-learn library, and the model of EERM is implemented according to (8). In particular, to obtain the user summary for EERM, we first plot the correlation between all ten attributes to simulate the intuition for the data. Then, we remove features that are not strongly correlated to the target, which are *longitude*, *total amount of bedrooms*, *population* and *total amount of households*, to form a new dataset and fit it with another linear regression. The prediction of the new dataset is used as the user summary. Coefficient of determination regression (R^2) score is used to measure the model generalization performance. Score approaching to 1 indicates that the model can fit unseen data well, and negative score may suggest that the model is worse than a random model.

4.3 Experiment results

R^2 score for linear regression, ridge regression, LASSO regression and EERM are 0.461, 0.450, 0.408 and 0.413 respectively. Figure 1 presents the learned weight parameters of these models.

From R^2 scores we can observe that models of linear regression and ridge regression have a similar performance, which is better than the other two; the EERM model performs slightly better than the LASSO regression model. These observations may result from the fact that the dataset includes few features compared to its large amount of data samples, which indicates that regularization is hardly needed here to constrain model complexity, especially the feature selection from LASSO regression.

From Figure 1 we can clearly observe the effects of regularization algorithm compared to the baseline model. The weights from regularization algorithms are smaller than the one from the baseline model, which suggests that models from regularization have lower complexity. Furthermore, the features which are deemed to be insignificant in the user summary have the smallest learned weights from EERM, which indicates that



	longitude	latitude	housing median age	total rooms
LR	-0.461	-0.467	0.123	-0.119
Ridge	-0.374	-0.376	0.124	-0.088
LASSO	-0.179	-0.173	0.12	-0.0
EERM	-0.167	-0.164	0.13	-0.005

	total bedrooms	population	households	median income
LR	0.383	-0.36	0.138	0.644
Ridge	0.316	-0.347	0.165	0.635
LASSO	0.239	-0.278	0.089	0.616
EERM	0.14	-0.184	0.107	0.627

Figure 1. 8 learned weight coefficients of linear regression, ridge regression, LASSO regression and EERM.

the model from EERM is more consistent with the user's intuition for the data, and thus it is more explainable to the user.

The experiment shows the essence of EERM, which is the trade-off between empirical risk and explainability. The model of EERM has acceptable empirical risk and better user-specific explainability compared to other models. Nevertheless, more circumstances need to be considered for a comprehensive evaluation of EERM, such as giving a user summary that contains wrong intuition for the data and applying EERM to non-linear hypothesis space.

5 Conclusion

This paper introduces the essence and details of Explainable Empirical Risk Minimization. For evaluating the performance of EERM preliminarily, we implement EERM in linear hypothesis space and then conduct experiments together with algorithms of linear regression, ridge regression and LASSO regression. The result shows that in linear hypothesis space, with appropriate user summary, EERM is capable of learning a model with satisfying empirical risk and better user-specific explainability. Further work needs to be done to evaluate EERM comprehensively, including implementations in non-linear hypothesis space and experiments with user summary containing wrong data intuition.

References

[1] Katsunori Sasahara, Masakazu Shibata, Hiroyuki Sasabe, Tomoki Suzuki, Kenji Takeuchi, Ken Umehara, and Eiji Kashiyama. Feature importance of machine learning prediction models shows structurally active part and important physicochemical features in drug design. *Drug metabolism and pharmacokinetics*, 39:100401, 2021.

[2] K. Shailaja, B. Seetharamulu, and M. A. Jabbar. Machine learning in healthcare: A review. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 910–914, 2018.

[3] Wei-Yang Lin, Ya-Han Hu, and Chih-Fong Tsai. Machine learning in financial crisis prediction: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):421–436, 2012.

[4] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Springer, 2013.

[5] Alexander Jung. Explainable empirical risk minimization. *CoRR*, abs/2009.01492, 2020.

[6] Alexander Jung. A gentle introduction to supervised machine learning. *CoRR*, abs/1805.05052, 2018.

[7] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014.

[8] V. Vapnik. Principles of risk minimization for learning theory. In J. Moody, S. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.

[9] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, Oct 2019.

[10] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[11] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006.

[12] Katharina Beckh, Sebastian Müller, Matthias Jakobs, Vanessa Toborek, Hanxiao Tan, Raphael Fischer, Pascal Welke, Sebastian Houben, and Laura von Rüden. Explainable machine learning with prior knowledge: An overview. *CoRR*, abs/2105.10172, 2021.

A Comprehensive Analysis of Generative Models

Maryum Hamid
maryum.hamid@aalto.fi

Tutor: Tian Yu

Abstract

Generative models have been existent for many years in the machine learning field. Generative Adversarial Networks and other generative models have been used to perform image processing, image generation, video generation and prediction. With advancements in the field of machine learning and artificial intelligence, it is now possible to generate art pieces and music as well. We study the algorithm and architecture of different generative models in order to provide the reader with insights on which generative model to choose while solving a problem.

KEYWORDS: Convolutional Neural Network (CNN); Variational autoencoders (VAEs); Generative Adversarial Networks (GANs); deep learning; machine learning.

1 Introduction

Most classification machine learning models are discriminative models, where decision boundaries are built between classes of training data. Whereas generative models are built based on probability distribution. Model estimates the probability distribution from the training data and generates a new distribution approximate to the distribution of the training data [24].

Considering a set of data with features X and labels Y, generative models capture the joint probability P(X, Y), such as Naive Bayes, Gaussian Discriminant Analysis (GDA) if there are labels, or just P(X) if there are no labels, such as GMM, Variational autoencoders (VAE). Whereas, discriminative models capture the conditional probability P(Y | X), such as Logistic Regression.

Training the generative models takes more time and hardware than discriminative models as the creation of probability distribution as the original dataset requires a higher number of correlation than just determining the labels to the most probable classes [10]. Taking an example, a convolutional neural network (CNN) model captures the differences between the images of cats and dogs. Whereas, a deep convolutional generative adversarial network (DCGAN) learns the features and generates new images of cats and dogs. Generative models capture almost all the features whereas discriminative models might miss a few features. Generative models trained on unlabeled data can be used as efficient feature extraction tools [17], and downstream classifiers could be trained or fine tuned on labeled data [4], these classifiers are used in self driven cars, computer vision based automation [18], disease detection [25], speech recognition [20] [27] and weather prediction [23].

This paper presents a comprehensive analysis of the generative models and their application. The involvement of deep learning in the building of generative models helps the reader get necessary information to decide which model should be used. This paper is organized as follows: Section 2 provides information on different Generative Adversarial Networks models. Section 3 provides information on Variational Autoencoders. Finally, Section 4 provides concluding remarks.

2 Generative Adversarial Networks

Most machine learning models are used for classification and for this, two approaches have been taken in supervised and unsupervised learning. Generative Adversarial Networks (GANs) belong to the family of Generative models, whose models generate new samples by learning the probability density function from the given training samples [11]. The pattern thus determined is used by the model to generate new data, which falls under the category of realistic images of the type of original training dataset [9]. The resolution and quality of images produced improve with

every new architecture. GANs consist of two simultaneously trained neural networks: Generator and Discriminator. Generator is responsible for capturing the data distribution and generating new samples [9]. Whereas a Discriminator is usually a binary classifier [2] that is trained to label the actual and generated samples by the Generator as precisely as possible. Generators and Discriminators are trained together in a zero-sum game, Generators are trained to fool the Discriminators, while Discriminators are trained to accurately detect generated fake examples.

Earlier versions of GANs had shallow Generator and Discriminator structures, this caused instability in training. Afterward, several GANs adopted deep residual network Generator backbones for generating high resolution images. Hardware advancement has played a major role in training more sophisticated Generators and Discriminators. This enables GANs to serve for data augmentation [13], image to image translation [28] and data generation [19].

2.1 Basic Structure

The GAN working structure consists of three structures: model learning, model training and deep learning integration with generative models. GANs basically serve the unsupervised solutions [8]. However, by the advancement in GANs, they have also proved to serve semi-supervised and supervised solutions [9]. GANs are used to provide solution software, such as in the field of healthcare, banking, and sports. GANs are an innovation in the field of generative models built on the structure of deep learning models, Discriminators and Generators. These models are of the convolutional neural networks (CNN) framework. Generator models read the features from the input data and imitate these features to generate fake images. The Discriminator is built on top of a Generator model and is used to check the performance of the Generator. When the Discriminator cannot distinguish between actual images and fake images, training moves towards completion. We now discuss a few major contributions done in the field of GANs.

Neural Architecture

GANs are based on minimax game theory, consisting of two neural networks: Generator and Discriminator as shown in Fig. 1. The Generator (G) takes a noise vector Z and generates new samples with G(z). The sample is then passed through a Discriminator (D). The Generator inputs are

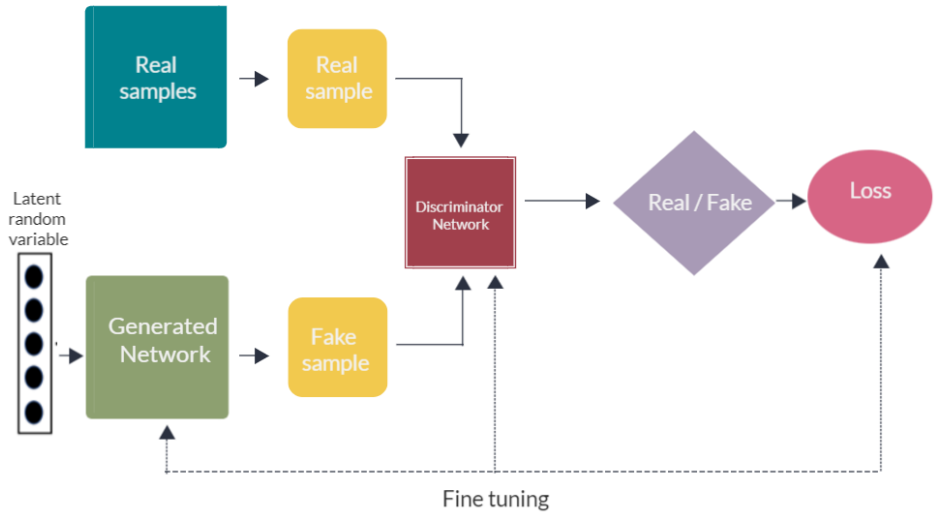


Figure 1. GAN Basic Structure

updated based on the probability results of the Discriminator. Generator and Discriminator basically compete. During training parameters of both are updated using backpropagation with the aim to make the Generator able to generate realistic samples and Discriminator to get better at labeling real and fake images. When a sample image X is added, the Discriminator model gives out the probabilistic results of the sample being real or fake. These results are then fed into the Generator model.

The Loss function of Minimax is given as:

$$Min_G Max_G f(D,G) = E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]. \quad (1)$$

Here, E_x is the expected value of the real data samples, $D(x)$s the probability estimate of the Discriminator. If x is real, $G(z)$s the output of the Generator with a noise vector z as input, $D(G(z))$ is the Discriminator probability estimate. If the fake generated sample tends towards real, E_z the expected value over random inputs fed to the Generator. Generator tries to reduce the minimax and Discriminator tries to increase it.

Conditional Generative Adversarial Nets (CGAN)

CGANS, an extension of GANs, are used for conditional sample generation. CGANs are fed with extra information,y, defining the conditions of data generation [21]. CGANs take into concatenation of the extra informtion and input, to perform conditioning as shown in Fig. 2. The extra information can be class labels or any other functionality provided with the dataset [22].

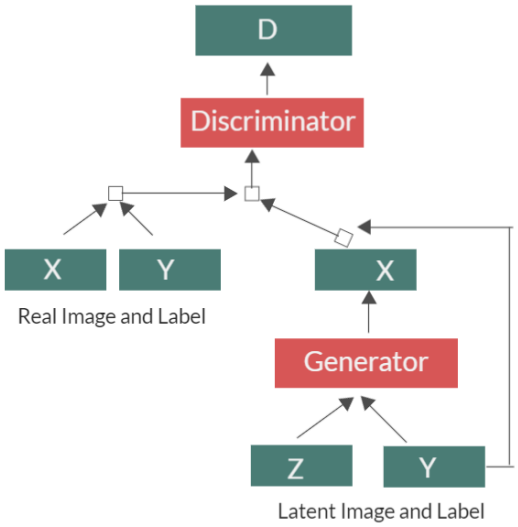


Figure 2. CGAN Structure

$$Min_G Max_G f(D,G) = E_x[\log(D(x|y))] + E_z[\log(1 - D(G(z|y)))]. \quad (2)$$

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (DCGANs)

DCGANs use deep convolutional neural networks as the base structure for both the Generator and Discriminator models [24]. Deep Learning is a secondary field of Machine Learning, built on the structure and functionality of the brain. It can also be used as a generative model. Deep learning techniques involve many layers of neural networks in one architecture. Discriminative models are the most important technique of deep learning, which the split into different class labels based on some discriminative features. Generative models with deep learning applications have the advantage that they can work with large datasets.

The basic GAN architecture uses multi-layer perceptron (MLPs). The DCGANs use CNN instead of MLPs as they produce better results with images than MLPs. The key features of DCGANs are listed below:

- Generator, shown in Fig. 3, convolution layers are replaced with transposed convolution layers, as a result, the representation of the Generator is larger at each layer, as it maps a lower dimensional vector into a higher dimensional image. Furthermore, the pooling layers are replaced with stridden convolutions (Discriminator) and fractionally stridden convolutions (Generator) [3].

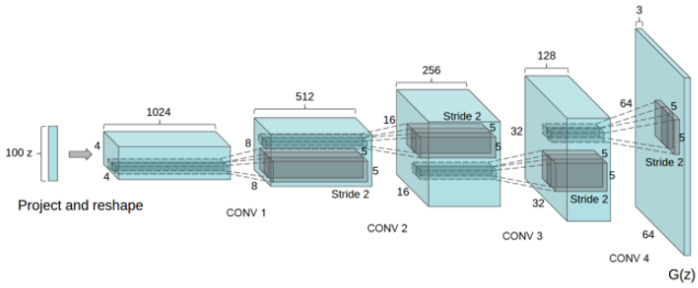


Figure 3. DCGAN architecture [24]

- Batch normalization is used in both the Generator and the Discriminator.

- All layers of the Generator, excluding the output layer, uses the ReLU activation [1].

- Lastly, Stochastic Gradient Descent (SGD) [26] is replaced with Adam [14].

Progressive Growing of GANs for Improved Quality, Stability, and Variation (ProGAN)

ProGANs were developed for training GANs to generate high resolution images via incremental growing of Discriminator and Generator [13]. ProGANs train the generator in an incremental way, first, it is trained for low resolution images and then moves on to higher resolution images. The generator is trained for a very low resolution image first, then other layers on top add the details for high resolution results [7].

The Fig. 4 shows that the training set starts with Generator and Discriminator both training at a very low resolution of 4 x 4 pixels. Going forward, layers are added to Generator and Discriminator sequentially. All the layers remain trainable throughout the training process. N x N is to represent the convolutional layers processing on N x N spatial resolution. On right six images are shown as a result of training and obtaining the images of 1024 x 1024 resolution.

2.2 Variational Autoencoders

This section is divided into two parts: 3.2 Autoencoders and 3.3 Variational Autoencoders.

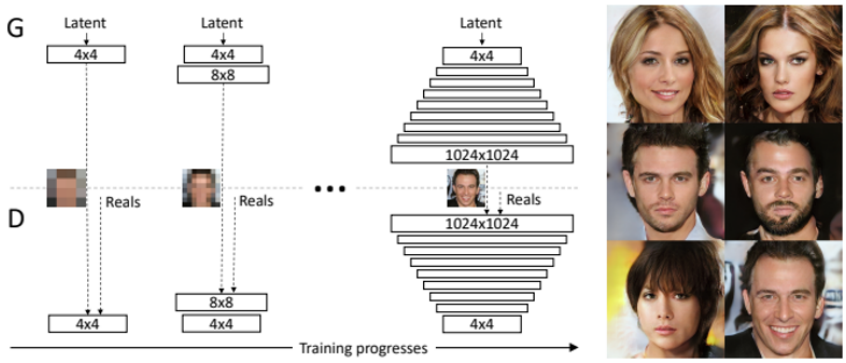


Figure 4. ProGAN architecture [13]

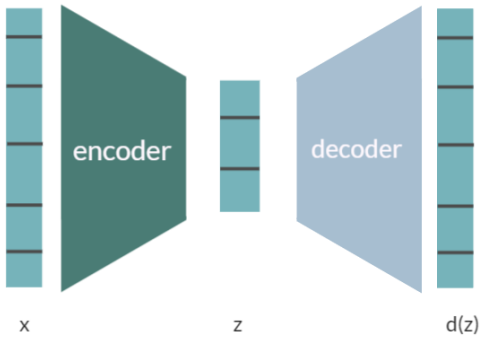


Figure 5. Autoencoders with Loss function

Autoencoders

Autoencoders have encoder and decoder sets, neural networks, use an iterative optimization process. After every iteration, the output of the encoder-decoder model is compared with the input data and the error is backpropagated to the architecture, in order to update the weights of the network. The illustration of an autoencoder with loss function is given as:

$$Loss = ||x - \hat{x}||^2 = ||x - d(z)||^2 = ||x - d(e(x))||^2. \tag{3}$$

The encoder and decoder architectures have just one layer with a linear autoencoder like Principal Component Analysis (PCA) works up the best linear subspace to generate data with the least information loss while encoding [12]. Encoding and decoding matrices that PCA calculates, are only one of the required solutions. Whereas there are many encoder-decoder combinations that give the optimal solution.

The encoder and decoder are non-linear. The architecture of the autoencoder determines the dimensionality it can perform reduction on, the more complex architecture higher the resolution reduction can be. The ultimate purpose of reduction is to reduce the number of dimensions while

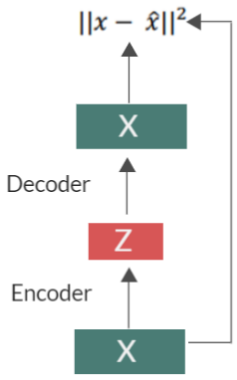


Figure 6. Loss Structure

keeping the important information in the reduced form. For this purpose, the length of the latent space and depth of the autoencoder is controlled and adjusted depending on the condition of dimension reduction.

Autoencoders comprise three layers. Input layer X, codding middle layer Z, and the output layer $\hat{X}$. The encoder output weights are kept in Z and decoder outputs in $\hat{X}$. The mapping function for encoding can be:

$$Z = f(WX + b). \tag{4}$$

Here (Equation. 4) b is the bias and W is the weight vector. The loss function can be stated as:

$$L = ||x - \hat{x}||^2. \tag{5}$$

The error so calculated is then backpropagated to the network and the new weights are generated using these errors as shown in Fig. 6.

Variational Autoencoders (VAEs)

Variational autoencoders are encoders trained to avoid overfitting. The latent space has suitable properties that enable the generative process [5]. Variational encoders, like any other autoencoder, consist of an encoder and decoder which are trained to reduce the backpropagation error between initial data and generated data. In VAE models, the encoders encode the input as a distribution over the latent space. A point in the latent space is taken and then sampled. The sampled point is then decoded, and the backpropagation error is calculated [6]. The calculated error is then fed to the network to update encoder-decoder weights.

The input is encoded as a distribution with a variance that expresses the

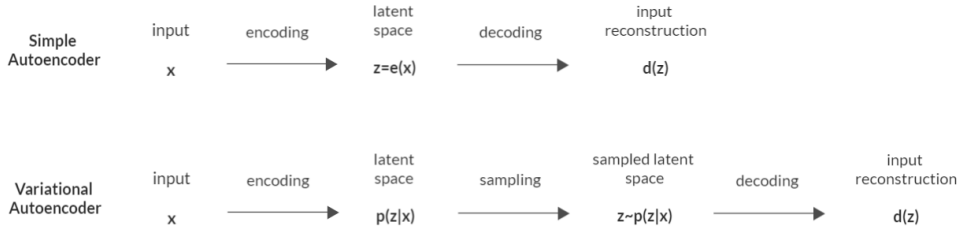


Figure 7. Difference between autoencoder and variational auto encoders

latent space regularization, forcing the encoders to output latent descriptive attributes that are from a standard normal distribution [16]. The loss function in VAE training comprises a reconstruction term on the final layer. The reconstruction term regularizes the latent space by making the distribution close to the standard normal distribution [15]. The regularity from the latent space to complete the generative process involves two properties, continuity and completeness. Continuity is when two closely located points in the latent space would not give entirely different outputs when decoded. Completeness is that a point in latent space should give the required results once decoded. This regularization term enables the model to encode data in the latent space to overlap. VAEs, the autoencoders, that encode inputs as distribution, unlike PCA which uses point encoding, and the latent space is regularized by constraining distributions.

3 Conclusion

In this paper, we present an overview of most of the generative models and their applications. We presented the architectures and applications of these models. All the models described in this paper are being actively used in different machine learning domains. The paper also highlights the flaws and shortcomings of the models. We hope that this study will assist academic and industry researchers in gaining a good understanding of generative models and their applications.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Miguel A. Carreira-Perpinan and Ramin Raziperchikolaei. Hashing with binary autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[3] Ashutosh Chapagain. Dcgan–image generation. 02 2019.

[4] Yushi Chen, Xing Zhao, and Xiuping Jia. Spectral–spatial classification of hyperspectral data based on deep belief network. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6):2381–2392, 2015.

[5] Carl Doersch. Tutorial on variational autoencoders, 2016.

[6] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[7] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with WaveNet autoencoders. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1068–1077. PMLR, 06–11 Aug 2017.

[8] Harshvardhan GM, Mahendra Kumar Gourisaria, Manjusha Pandey, and Siddharth Swarup Rautaray. A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38:100285, 2020.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[11] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.

[12] Ian Jolliffe. *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[13] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[15] Siddique Latif, Rajib Rana, Junaid Qadir, and Julien Epps. Variational autoencoders for learning latent representations of speech emotion: A preliminary study, 2017.

[16] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 689–698, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[17] Ping Liu, Shizhong Han, Zibo Meng, and Yan Tong. Facial expression recognition via a boosted deep belief network. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1805–1812, 2014.

[18] Ping Liu, Shizhong Han, Zibo Meng, and Yan Tong. Facial expression recognition via a boosted deep belief network. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1805–1812, 2014.

[19] Francesco Marra, Diego Gragnaniello, Davide Cozzolino, and Luisa Verdoliva. Detection of gan-generated fake images over social networks. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 384–389, 2018.

[20] Daniel Michelsanti and Zheng-Hua Tan. Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification. In *Interspeech 2017*. ISCA, aug 2017.

[21] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[22] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. 11 2014.

[23] Munir Nayak and Subimal Ghosh. Prediction of extreme rainfall event using weather pattern recognition and support vector machine classifier. *Theoretical and Applied Climatology*, 113, 03 2013.

[24] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[25] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning, 2017.

[26] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[27] Peng Shen, Xugang Lu, Sheng Li, and Hisashi Kawai. Conditional Generative Adversarial Nets Classifier for Spoken Language Identification. In *Proc. Interspeech 2017*, pages 2814–2818, 2017.

[28] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.

Optimizing firewall policies

Massimo Bertocchi
massimo.bertocchi@aalto.fi

Tutor: Tuomas Aura

Abstract

Firewalls are a crucial point to secure a Local Area Network, which allows protecting our network from the internet. In the last decades, the firewall rules have been increasing growth due to the variety of applications and more complex communication systems. In order to enable and prevent an overloading of the host, rule based classification needs to be taken into account, improving the speed of the firewall itself. This paper introduces the multidimensional classification for packet classification, analyzing the advantages and the drawbacks these technologies.

KEYWORDS: HiCuts, Packet Classification, Decision tree

1 Introduction

Digitalization has increased internet traffic, making firewalls a critical part of home, enterprise and cloud provider networks [4]. The firewall is intended to be the only contact point between the Internet and the local network; thus, all packets must pass through it. This gate needs to handle traffic in and out of the network [7] [6], resulting in a bottleneck for several companies [15]. This paper aims to review and analyse a Packet Classification algorithm [12] for optimizing firewall policies, verifying the

possible improvement and proposing an implementation. The paper uses some literature surveys and preview researches, and the implementation is written in python code.

The goal of this paper is to describe the logic behind the multidimensional packet classification methods and evaluate the advantages of this viewpoint. Firstly, HiCuts [1] laid the foundation of this problem class, forging the new concept. Further improvements were made using HyperCuts [3], using a different data structure, and BiCuts [8], cutting in equidensity points; EffiCuts instead [14] made a memory optimization advancement to release unused memory waste.
Additionally, during the last decades, non-multidimensional splitting methods were invented and tested [9] to face the problem from a different angle and to obtain a better performance [5].
This paper explains the main characteristics and scenario in which perform better, highlighting the advantages of a solution over their drawbacks [16].

The rest of this paper is structured as follows. Section 2 describes the algorithms and multidimensional splitting. Section 3 describes the experiment using Hicuts. Section 4 outlines the advantages and the drawbacks, and how to address the issues. Section 5 concludes the paper.

2 Background

2.1 Packet Classification

A firewall policy consists of several rules, where each rule represents a requirement and an action section. The scope of packet classification [12] is to match the requirement section of the incoming packet, without analysing every rule of the policy table. A packet matches a rule when each field of the packet matches the corresponding field of the rule [11]; considering that a packet can match multiple rules, a value needs to be checked for the priority. The priority is defined as the arrangement of the rules in policy table; thus, the rules on top represents the most significant and the last one the least important.
To avoid the checking of the entire table, the table needs to be divided into small buckets, and a multidimensional cutting is used as partition points

of the initial domain. Therefore, the domain is separated into small rule sets, and the algorithm could only check the subspace and not the entire policy table, as shown in Figure 1.

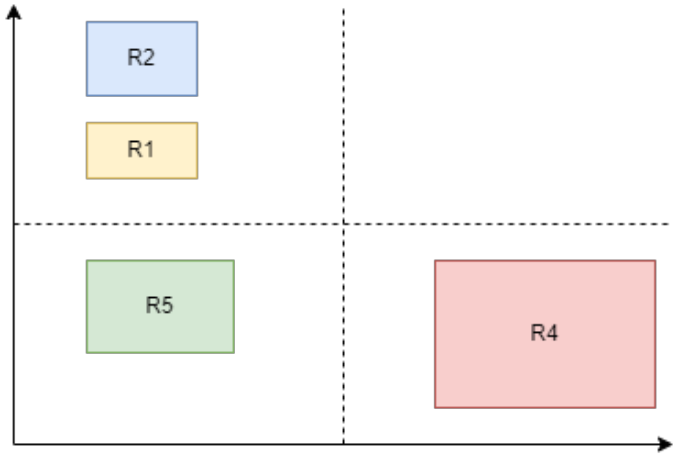


Figure 1. Multidimensional division of ruleset

2.2 Decision Tree

A decision tree is a tool to express a classification [13]. It consists of a "root node" that has no incoming edges, "leaf nodes" that have no outgoing edges and "internal nodes" that have outgoing and incoming edges. Each node splits the domain into several sub-domains [10].
When each node separates the domain into precisely two different sub-domains, as with Hicuts, such a tree is called “binary tree”.
The complexity of a binary tree is determined by the depth, or height, of a tree, i.e. the number of nodes on the longest path between the root and the furthest leaf, is used to determinate its complexity. The implementation of trees in Hicuts could be seen as a geometrical collection of hyperplanes, with internal nodes representing cutting locations in various dimensions.

2.3 Hicuts

The packet classification using Hierarchical Intelligent Cuttings was developed by Gupta and McKeown [2]. This algorithm consists of cutting the multidimensional space into equally divided segments based on the headers of the incoming packet to the firewall. This method begins with a cutting point in one dimension and builds the decision tree on it, based on the cutting value. The process is repeated recursively for each dimensional cutting, using a user-specified input parameter, which can be de-

fined as the depth of the final tree. Furthermore, the rules are divided into various sub-categories, resulting in a linear search of $\frac{n}{2^c}$, where n is the number of rules and c the number of cuts.

3 Implementation

3.1 Overview

This paper presents an implementation of Hicuts with Python and evaluates its efficiency comparing it to a standard firewall policy lookup. The program divides the initial rule space into different lists using a multidimensional cutting.
The initial assumption is that finding a match in a policy table of n elements takes $O(\frac{n}{2})$, and going through the tree of height m takes $O(m)$.
Firstly, the number of cuts is defined based on a user-input value for the IP dimension and another value for the port dimension. As the result of the process must be a balanced tree, the number of cutting points is even. Each cut is then performed such that the dimensional space is divided into two equivalent subspaces. Subsequently, one tree is created using the IP cuttings in the previous step as the internal nodes of the tree, with leaves pointing to an additional tree structure. The second tree is created using the cutting points of the port, but a list object is assigned for each leaf instead of a tree pointer. The last step is a tree merging, combing the port root to the IP leaves.
At this point the classifier should be ready to analyse the policy rules for one specific system and to classify incoming packets in the network.

3.2 Input

The input file is formatted in the following way:
The first value represents the source IP address with the relative network, and the second value the destination IP. Regarding the third and fourth values, the port number is described as a port range divided by a colon in the middle of it; thus, the port range $x : y$ represents all the ports starting from x until y; in order to indicate only one specific port, the values x and y have to be same.

Src IP	Dst IP	Src Port	Dst Port
88.108.88.203/16	251.105.175.122/32	0 : 65535	1724 : 1724
176.19.181.6/32	52.174.116.2/31	0 : 65535	1490 : 1490
0.0.0.0/0	69.0.0.0/12	0 : 65535	0 : 65535

Table 1. Example of input data for the Hicuts classficator

3.3 Experiment

The input file contains 947 randomly generated firewall rules. There are some default routes, such as 0.0.0.0, and different ranges of ports (i.e., broad range and single port).
During the experiment, 4 cuts were performed: two port-based and two IP-based. As such, the resulting tree will have a depth of 4 with 8 sub-spaces. Figure 2 is the graphical representation of the resulting tree.

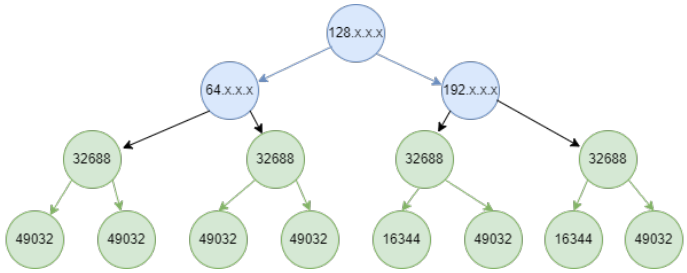


Figure 2. Decision tree based on four cuts

In the next steps, the rules will be separated into different sub-lists using the tree classifier created in the previous step. Therefore, the 947 rules will be separated into various lists: the final rule lists will not be evenly balanced, since the policies are not uniformly distributed in the domain.
Finally, the output of the program consists of multiple buckets, coming from each leaf of the tree, and some general statics of the experiment itself. Considering the previous input file, the major unbalanced bucket counts 242 elements. The height of the tree is 4 and, in the average case, the program checks the match against 125 values. The average case is calculated by counting the checks in the tree traversal, and the rule matches in the sub-lists.

4 Evaluation

In the best and in the average case, the algorithm performed as predicted. However, there were overheads in the worst-case scenario.
To assess the method, a technical process must be considered, beginning with the problem's baseline. An unordered list can be traversed in $O(n)$ time. Running through a sub-section of the tree, on the other hand, takes $O(n/2c)+O(c)$ in the best and average cases, and $O(n)+O(c)$ in the worst case, where n is the number of rules and c the number of cuts. The tree is created with an additional cost of $O(n)$, but this value is negligible for the classification's correct runtime.
Moreover, in the worst scenario, the memory consumption is drastically increased, due to the unnecessary creation of empty lists.
A further problem could be found when using several wildcards in the rules as shown in Figure 3. Having rules that are widespread along all the domains can provoke redundant rule policy in multiple leaf-list, and therefore it might produce a memory overhead with no optimization, considering that all the rules are written in many lists.

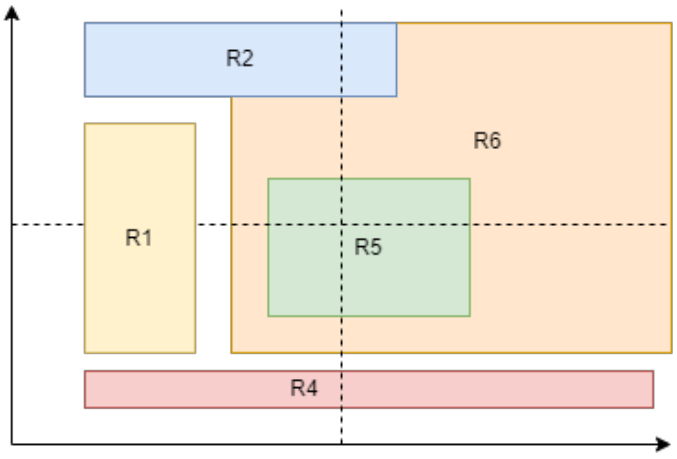


Figure 3. Overlapping rules

4.1 Efficuts

Efficuts is an improvement of Hicuts and Hypercuts, with three major characteristics [14]. To save memory waste, the method divides larger rules with several wildcards from strict rules, avoiding repetition and redundancy of non-homogeneous rules. It builds separable trees and constructs a decision tree using the split input, such as in Hicuts. The second enhancement is made by merging the output of the trees, using a selective

tree merging algorithm. The unification of the tree can result in optimizing the running time, since the depth of the resulting tree is less than the sum of the outputs trees, resulting in fewer memory accesses. The third enhancement is the equal-density cuttings. Fine cuts are used to split densely clustered rules, while broad cuts are employed to separate widespread one, addressing the previous problem of Hicuts and incurring in less memory overhead.

4.2 Bicuts

This new approach based on Hicuts was developed by Zhi Liu, Shijie Sun, Hang Zhu, Jiaqi Gao, and Jun Li [8]. It addresses the issues in Hi-cuts by applying the cutting points based on the input rule set, avoiding the worst-case scenario. Moreover, the internal nodes have a bitmask to separate the two branches, making it a faster operation for any device. The no-binary tree does not compare the value of its node, but it performs an AND operation against the headers of the incoming packet. The outcome of that paper proposes an improvement of memory usage, consuming only 19% of memory usage and 59% of memory accesses compared to Efficuts.

5 Conclusion

Hierarchical Intelligent Cuttings consists of many advantages in the simplicity of the code and the data structure, having a relevant network performance in homogeneous rule sets. Furthermore, in case of its best case scenario, HiCuts outperforms its successor HyperCuts and Efficuts. HyperCuts implies complex data structure and more memory space, while Efficuts implies multiple data structures and significant runtime.
This paper proposes an implementation of the algorithm Hicuts. In a trade off between memory and runtime, the program performs slightly better over the standard ACL being a possible option for enterprise networks. The Python code can be further optimised to save memory and improve the average and the worst case scenario.

References

[1] Hatam Abdoli. Balanced hicuts: an optimized packet classification algorithm. 2009.

[2] P. Gupta and N. McKeown. Classifying packets with hierarchical intelligent cuttings. *IEEE Micro*, 20(1):34–41, 2000.

[3] Dr Jabbar and Noor Almalah. Design and implementation of a network on chip using fpga. *al rafdeen*, 21, 01 2012.

[4] Weirong Jiang and Viktor K. Prasanna. Large-scale wire-speed packet classification on fpgas. FPGA '09, page 219–228, New York, NY, USA, 2009. Association for Computing Machinery.

[5] Tihomir Katic and Predrag Pale. Optimization of firewall rules. pages 685 – 690, 07 2007.

[6] Alan Kennedy, Xiaojun Wang, Zhen Liu, and Bin Liu. Low power architecture for high speed packet classification. pages 131–140, 01 2008.

[7] T. V. Lakshman and D. Stiliadis. High-speed policy-based packet forwarding using efficient multi-dimensional range matching. 28(4):203–214, oct 1998.

[8] Zhi Liu, Shijie Sun, Hang Zhu, Jiaqi Gao, and Jun Li. Bitcuts: A fast packet classification algorithm using bit-level cutting. *Computer Communications*, 109:38–52, 2017.

[9] Yan Luo, Ke Xiang, and Sanping Li. Acceleration of decision tree searching for ip traffic classification. pages 40–49, 01 2008.

[10] Yan Luo, Ke Xiang, and Sanping Li. Acceleration of decision tree searching for ip traffic classification. In *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '08, page 40–49, New York, NY, USA, 2008. Association for Computing Machinery.

[11] Yaxuan Qi, Lianghong Xu, Baohua Yang, Yibo Xue, and Jun Li. Packet classification algorithms: From theory to practice. pages 648 – 656, 05 2009.

[12] Sumeet Singh, Florin Baboescu, George Varghese, and Jia Wang. Packet classification using multidimensional cutting. SIGCOMM '03, page 213–224, New York, NY, USA, 2003. Association for Computing Machinery.

[13] T S Urmila and Raman Balasubramanian. Decision tree based network packet classification algorithms. 2015.

[14] Balajee Vamanan, Gwendolyn Voskuilen, and T. N. Vijaykumar. Efficuts: optimizing packet classification for memory and throughput. In *SIGCOMM '10*, 2010.

[15] Chenghong Wang, Donghong Zhang, Hualin Lu, Jing Zhao, Zhenyu Zhang, and Zheng Zheng. An experimental study on firewall performance: Dive into the bottleneck for firewall effectiveness. In *2014 10th International Conference on Information Assurance and Security*, pages 71–76, 2014.

[16] Sorrachai Yingchareonthawornchai, James Daly, Alex X. Liu, and Eric Torng. A sorted-partitioning approach to fast and scalable dynamic packet classification. *IEEE/ACM Transactions on Networking*, 26(4):1907–1920, 2018.

Task Allocation for Vehicular Fog Computing

Zixin Zhou
zixin.zhou@aalto.fi

Tutor: Wencan Mao

Abstract

Vehicular fog computing (VFC) provides latency-sensitive and data-intensive computation for vehicular networks. In VFC, idle computing resources of vehicles are shared between nearby vehicles by allocating computation tasks to service vehicles. However, the mobility of vehicles has posed a challenge for task allocation in VFC. This paper reviews recent task allocation schemes for VFC, comparing them in terms of communication standards, architectures, application scenarios, objectives, constraints, problem formulation and algorithms. Based on the analysis, challenges and future research directions for task allocations in VFC are discussed.

KEYWORDS: Task allocation, Vehicular fog computing, Cloud computing

1 Introduction

The emerging vehicular applications, such as autonomous driving [1], video streaming [2] and parking navigation [3], facilitate more efficient transportation. These applications have led to the increasing demand for ubiquitous real-time computing resources. Due to the limited computing capability of a single vehicle, it is necessary to offload tasks to entities that are rich in computing resources, e.g., cloud data centers and base

stations.

However, offloading tasks to the remote cloud can cause high transmission delays [4]. In addition, the vehicular traffic demand changes according to time and place, while stationary fog nodes deployment needs to satisfy the peak demand, thus causing over-provisioning of resources. Furthermore, the advance in Vehicle-to-Vehicle (V2V) communication technologies enable efficient and reliable communication between vehicles. Therefore, vehicular fog computing (VFC), a promising paradigm that combines fog computing with the idle computing capacity of vehicles, has been proposed to provide computation for latency-sensitive and data-intensive applications.

As an important topic of VFC, task allocation, which decides how to offload tasks from client vehicles to fog nodes, has received notable attention. Existing research has studied task allocation mechanisms in various scenarios, including autonomous driving [1], visual crowdsourcing [5] and traffic management [6]. The varying needs lead to different objectives and constraints, such as latency, quality of service, energy consumption and reliability. Additionally, researchers have devised different architectures to model the responsibility of VFC components [7]. Then, the task allocation problem is formulated as an optimization problem or a Markov decision process (MDP) and solved with various algorithms [8].

This paper reviews recent task allocation schemes for VFC, comparing them in terms of communication standards, architectures, objectives, constraints, application scenarios, problem formulation and algorithms. In addition, this paper identifies the challenges of research in task allocation for VFC.

The remainder of the paper is organized as follows. Section 2 introduces the basic concepts of VFC and motivation for using VFC. Section 3 presents a review of recent task allocation schemes. Section 4 discusses the challenges and possible research directions. Finally, the paper is concluded in Section 5.

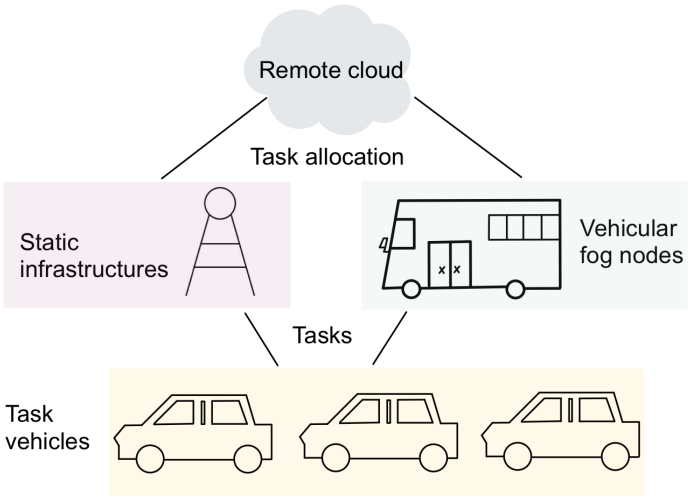


Figure 1. Structure and components of VFC

2 Concepts and motivation

2.1 Vehicular fog computing

Fog computing is a distributed computing paradigm with computing resources close to end users [9], which reduces communication latency between users and computing resources. For vehicular fog computing, the fog nodes can be deployed at both static infrastructures (e.g., cellular base stations and roadside units) and vehicles [4]. In many studies, service vehicles refer to vehicles that serve as fog nodes, while task vehicles represent the vehicles whose tasks will be offloaded to fog nodes. Figure 1 presents the structure and components of VFC.

According to Xiao and Zhu [10], the motivation for utilizing vehicles as fog nodes is two-fold. First, the density of vehicles varies with regard to time and location. Therefore, it is not feasible to rely on static infrastructures to satisfy the changing needs of computing resources of vehicles. Due to the proximity of vehicles during traffic, vehicles with idle computing capacity can be deployed as fog nodes. Second, the advent of reliable and high-speed V2V communication protocols facilitates the implementation of vehicles as fog nodes.

2.2 Task allocation

Tasks are operations that could be assigned to and executed on fog nodes. Task allocation schemes match fog nodes with tasks in the queue, deciding when and where the tasks will be executed. The decisions are made ac-

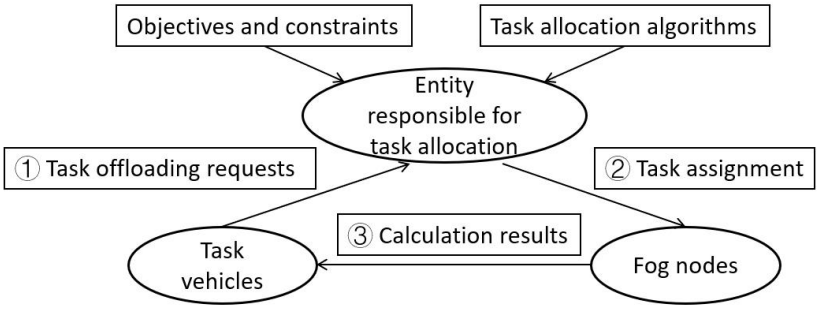


Figure 2. Workflow of task allocation in VFC

cording to their objectives and constraints. Figure 2 illustrates the workflow of task allocation in VFC.

3 Task allocation schemes review

3.1 Communication standards

Communication between components of VFC serves as the basis of task allocation. Existing studies commonly use five types of communication systems for VFC [8]: V2V, Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrian (V2P), Vehicle-to-Network (V2N) and Vehicle-to-Everything (V2X). The V2X system includes the other four types of systems. Several studies [2], [11], [12] utilize V2V and V2I systems. FlexSensing [5] uses V2X system. Typical inter-vehicle communication technologies for VFC include Dedicated Short Range Communications (DSRC) and Long Term Evolution (LTE) [13].

In V2V communication, the number of transmission hops between task vehicles and service vehicles affects the range of communication [8]. One-hop communication refers to the direct communication between task vehicles and service vehicles. In multi-hop communication, some vehicles forward the task allocation requests to service vehicles and relay the calculation results to task vehicles. Compared with one-hop communication, multi-hop communication exploits more potential computing resources. However, the increased number of transmission hops can cause higher latency, and the dynamic topology of vehicular networks could result in more vulnerable connections. Most existing research adopts one-hop communication. The Mobility Aware Multi-hop Task Offloading (MMTO) scheme [1] utilized multi-hop communication.

3.2 Architectures

According to the entities that make the task offloading decisions, the architecture of VFC task allocation approaches can be classified into three categories [7]: server-based, distributed and other models. In server-based models, the task vehicles generate tasks and send task offloading requests to the server. Then, the server offloads tasks to service vehicles and sends the task execution results back to task vehicles. The static infrastructure generally serves as the server. Most studies [2], [3], [5], [12], [14], [15] assume that there is a single server in one region. In contrast, distributed models [1], [16] allow service vehicles to decide on details of task allocation. Other models of task allocation include the mixed model [17] and the three-layer model [6], [11]. The mobility aware blockchain-enabled offloading (MABOS) method [17] adopts a combination of server-based model and distributed model. In [11], task vehicles send task offloading requests to an access roadside unit (RSU), and the RSU sends the request to a software-defined network controller, which makes offloading decisions. Ning et al. [6] added a cloud layer to the server-based model to perform city-level traffic control, which complements the local management of the cloudlet layer.

In terms of the basic unit of task offloading, partial offloading schemes [11], [12] assign a proportion of the task to fog nodes, while other schemes operate on individual tasks. Partial offloading aims at reducing the possibility of recomputing time-consuming tasks caused by frequent changes in V2V connection [11]. This task allocation policy can be adopted when reliability and transmission delay are jointly considered. Table 1 presents the architecture of recent VFC task allocation schemes.

3.3 Application scenarios, objectives and constraints

The objectives and constraints of task allocation approaches vary with regard to different application scenarios. For multimedia-related applications, the quality of images is an important factor. Zhu et al. [16] designed the Chameleon approach for visual-based assisted driving, which aims at reducing service latency and increasing image resolution. FlexSensing [5] utilized VFC for vehicle-based visual crowdsourcing, maximizing quality of information with latency constraints.

Energy consumption and economic incentives are considered for applications related to commercial activities. Zhang et al. [3] investigated park-

Table 1. Architectures of task allocation schemes for VFC

Method	Model	Partial Offloading
FlexSensing [5]	Server-based	✗
Parked vehicle assisted VFC [3]	Server-based	✗
PVFChain [14]	Server-based	✗
Folo [2]	Server-based	✗
Priority-Aware Offloading [15]	Server-based	✗
DRL Based V2V Partial Offloading [12]	Server-based	✓
QUOTA-UCB[18]	Server based	✗
MMTO [1]	Distributed	✗
Chameleon [16]	Distributed	✗
MABOS [17]	Mixed	✗
VFC-enabled offloading [6]	Three-layer	✗
EC-SDIoV [11]	Three-layer	✓

ing guidance based on VFC and devised a parked vehicle assisted VFC system to achieve a win-win for system actors, including moving vehicles, parked vehicles, commercial fog nodes, and fog node controller (FNC), by maximizing utilities for vehicles and FNC with economic constraints. To exploit the idle computing resources of parked vehicles and gain profits, Huang et al. [14] included service fee and energy consumption minimization in their objectives.

Some studies focus on general-purpose applications, most of which aim at reducing latency. Zhu et al. [2] proposed a dynamic task allocation mechanism named Folo to minimize service latency and quality loss with fog node capacity constraints. Some researchers include priority as a task allocation objective. Huang et al. [14] encoded priority in the utility function of fog servers. Shi et al. [15] utilized different utility functions for tasks with high and low priority, which consider service availability and user incentives. For mobility-aware task allocation schemes [1], [12], [15], service availability or link connectivity is considered.

Security and reliability of VFC task offloading have also been investigated. Several studies have leveraged blockchain technology to provide secure task allocation for VFC. Huang et al. [14] designed a parked vehicle-assisted fog computing method based on blockchain (PVFChain) using smart contract operations, supporting authentication, request validation and reward integrity checking. Similarly, Liao et al. [18] employed smart contract and blockchain to ensure privacy, fairness and security of the task offloading process. To increase reliability, Edge computing-aided

Internet of Vehicles based on software-defined networking (EC-SDIoV) [11] considered both transmission and computation failure.

Table 2 shows a summary of application scenarios, objectives and constraints of the mentioned studies.

Table 2. Objectives and constraints of task allocation schemes for VFC

Method	Application Scenario	Objectives	Constraints
FlexSensing [5]	Vehicle-based visual crowdsourcing	Quality of information	Latency
Chameleon [16]	Visual-based assisted driving	Image resolution	Latency
Parked vehicle assisted VFC [3]	Parking navigation	Driving cost, walking cost, energy consumption, parking payment, FNC profit	Latency
VFC-enabled offloading [6]	Traffic management	Latency	None
MMTO [1]	Autonomous driving	Latency, incentive pay	Link connectivity
DRL Based V2V Partial Offloading [12]	General	Latency, priority, incentive	Service availability
Folo [2]	General	Latency, quality loss	Latency, quality loss, fog node capacity
Priority-Aware Offloading [15]	General	Latency, task priority, service availability, incentive	Price, single task assignment
EC-SDIoV [11]	General, reliability	Reliability	Latency
PVFChain [14]	General, security	Service fee, energy consumption, priority	Latency
MABOS [17]	General, security	Calculation costs	Bandwidth, resources, security, deadline
QUOTA-UCB[18]	General, security	Task offloading delay	Handover cost, queuing delay

3.4 Problem formulation and algorithms

Many existing studies consider the task allocation problems for VFC as optimization problems. Zhu et al. [2] formulated the task allocation

problem for VFC as a bi-objective minimization problem, which is optimized using a linear programming-based optimization algorithm and binary particle swarm optimization algorithm. Hou et al. [11] investigated partial offloading and reliable task offloading, considering the problem as an optimization problem, and proposing a heuristic algorithm named fault-tolerant particle swarm optimization algorithm for maximizing the reliability as a solution. Shi et al. [12] formulated the partial task offloading of VFC as a sequential decision-making problem, which is solved by a deep reinforcement learning algorithm (DRL) based on soft actor-critic. Huang et al. [14] modeled the process of task offloading as an optimal smart contract design problem, which is transformed into a user payment minimization problem and solved by the Stackelberg game framework. Lakhan et al. [17] formulated the multi-side task offloading for VFC as a convex optimization problem with movement, task priority, and computation capability constraints. A Mobility Aware Blockchain-Enabled offloading algorithm is conceived to solve it. Liao et al. [18] formulated the problem as a series of short-term deterministic optimization subproblems and proposed a QUeuing-delay aware, handOver-cost aware, and Trustfulness Aware Upper Confidence Bound (QUOTA-UCB) algorithm as a solution.

In addition, some studies use MDP to model task allocation problems. In Chameleon [16], the problem is modeled as a partially observable MDP and solved with a stochastic dynamic programming algorithm. Shi et al. [15] formulated a tri-objective problem as an MDP and solved it using a deep reinforcement learning algorithm. Similarly, FlexSensing [5] formulated the problem as an MDP and utilized a deep Q-network to solve it.

The mentioned algorithms for solving task allocation problems can be categorized into five types [8]: mathematical programming, machine learning, metaheuristics, game theory and auction methods. The problem formulation and algorithms of VFC task allocation schemes are presented in Table 3.

4 Discussion

Despite the progress in VFC task allocation research, many challenges remain to be solved. First, security and reliability issues of task allocation for VFC are a major concern. For applications that require secure and re-

Table 3. Problem formulation and algorithms of task allocation schemes for VFC

Method	Problem Formulation	Algorithm	Algorithm Category
Parked vehicle assisted VFC [3]	Optimization	Single-round/ multi-round parking reservation auction	Auction
PVFChain [14]	Optimization	Stackelberg game framework	Game theory
Folo [2]	Optimization	Linear programming-based optimization algorithm and binary particle swarm optimization algorithm	Metaheuristics
EC-SDIoV [11]	Optimization	Fault-tolerant particle swarm optimization algorithm	Metaheuristics
MMTO [1]	Optimization	A semidefinite relaxation method adjusted by an adaptive policy	Mathematical programming
VFC-enabled offloading [6]	Optimization	Brand-and-bound algorithm, Edmonds-Karp algorithm	Mathematical programming
MABOS [17]	Optimization	Linear search-based task scheduling algorithm based on blockchain technology	Mathematical programming
DRL Based V2V Partial Offloading [12]	Optimization	DRL algorithm based on SAC	Machine learning
QUOTA-UCB[18]	Optimization	Online learning-based algorithm	Machine learning
FlexSensing [5]	MDP	Deep Q-network	Machine learning
Priority-Aware Offloading [15]	MDP	DRL algorithm based on SAC	Machine learning
Chameleon [16]	MDP	Stochastic dynamic programming	Mathematical programming

liable transactions, such as autonomous driving, any possible successful attack could be fatal. Most existing studies adopt the server-based model, in which static infrastructures allocate the tasks. Hence, if attackers control the server, the vehicles will suffer from terrorism, free-riding and false-payment issues [14]. Recently, researchers have utilized blockchain technology and the distributed model to secure VFC task offloading. However, distributed task allocation methods receive incomplete information as input, thus resulting in local optimum solutions. Apart from security, to achieve reliable task allocation, task interruption and reprocessing need to be considered in future studies.

Furthermore, due to the assumptions for simplification, the task allocation schemes might not be feasible for real-world applications. Some

studies [2], [5], [12] assume that one server would be responsible for one region. However, in real applications, when a road becomes busy, one server could be overloaded with task offloading requests. Therefore, the policy of choosing servers can be further studied. In addition, many existing studies overlooked the dependency of tasks. For example, visual assisted driving requires object detection to be completed before planning the movement of vehicles. To avoid these problems, more factors need to be considered as metrics of task allocation methods.

Finally, there is a lack of real-world implementation. Currently, most task allocation methods are tested by simulation. Although most vehicles are equipped with the necessary hardware for communication, the communication standards have not been unified. Considering the advent of 5G technologies, future studies can also focus on adapting current inter-vehicle communication standards to 5G technologies to provide more efficient transmission.

5 Conclusion

This paper reviews recent task allocation approaches for VFC. First, related concepts and motivation are presented. Then, the VFC allocation schemes are compared in terms of communication standards, application scenarios, architectures, objectives, constraints, problem formulation and algorithms. In the end, challenges posed for future research are presented.

References

[1] Lei Liu, Ming Zhao, Miao Yu, Mian Ahmad Jan, Dapeng Lan, and Amirhosein Taherkordi. Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[2] Chao Zhu, Jin Tao, Giancarlo Pastor, Yu Xiao, Yusheng Ji, Quan Zhou, Yong Li, and Antti Yla-Jaaski. Folo: Latency and quality optimized task allocation in vehicular fog computing. *IEEE Internet of Things Journal*, 6(3):4150–4161, 2019.

[3] Yi Zhang, Chih-Yu Wang, and Hung-Yu Wei. Parking reservation auction for parked vehicle assistance in vehicular fog computing. *IEEE Transactions on Vehicular Technology*, 68(4):3126–3139, 2019.

[4] Tesnim Mekki, Issam Jabri, Lamia Chaari, and Abderrezak Rachedi. A survey on vehicular fog computing: motivation, architectures, taxonomy, and

issues. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 159–168. Springer, 2020.

[5] Chao Zhu, Yi-Han Chiang, Yu Xiao, and Yusheng Ji. Flexsensing: A qoi and latency-aware task allocation scheme for vehicle-based visual crowdsourcing via deep q-network. *IEEE Internet of Things Journal*, 8(9):7625–7637, 2021.

[6] Zhaolong Ning, Jun Huang, and Xiaojie Wang. Vehicular fog computing: Enabling real-time traffic management for smart cities. *IEEE Wireless Communications*, 26(1):87–93, 2019.

[7] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. Vehicular edge computing and networking: A survey. *Mobile networks and applications*, 26(3):1145–1168, 2021.

[8] Alisson Barbosa De Souza, Paulo AL Rego, Tiago Carneiro, Jardel Das C Rodrigues, Pedro Pedrosa Rebouças Filho, Jose Neuman De Souza, Vinay Chamola, Victor Hugo C De Albuquerque, and Biplab Sikdar. Computation offloading for vehicular environments: A survey. *IEEE Access*, 8:198214–198243, 2020.

[9] Paolo Bellavista, Javier Berrocal, Antonio Corradi, Sajal K. Das, Luca Foschini, and Alessandro Zanni. A survey on fog computing for the internet of things. *Pervasive and Mobile Computing*, 52:71–99, 2019.

[10] Yu Xiao and Chao Zhu. Vehicular fog computing: Vision and challenges. In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pages 6–9. IEEE, 2017.

[11] Xiangwang Hou, Zhiyuan Ren, Jingjing Wang, Wenchi Cheng, Yong Ren, Kwang-Cheng Chen, and Hailin Zhang. Reliable computation offloading for edge-computing-enabled software-defined iov. *IEEE Internet of Things Journal*, 7(8):7097–7111, 2020.

[12] Jinming Shi, Jun Du, Jian Wang, and Jian Yuan. Deep reinforcement learning-based v2v partial computation offloading in vehicular fog computing. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2021.

[13] Jiadai Wang, Jiajia Liu, and Nei Kato. Networking and communications in autonomous driving: A survey. *IEEE Communications Surveys & Tutorials*, 21(2):1243–1274, 2018.

[14] Xumin Huang, Dongdong Ye, Rong Yu, and Lei Shu. Securing parked vehicle assisted fog computing with blockchain and optimal smart contract design. *IEEE/CAA Journal of Automatica Sinica*, 7(2):426–441, 2020.

[15] Jinming Shi, Jun Du, Jingjing Wang, Jian Wang, and Jian Yuan. Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12):16067–16081, 2020.

[16] Chao Zhu, Yi-Han Chiang, Abbas Mehrabi, Yu Xiao, Antti Yla-Jaaski, and Yusheng Ji. Chameleon: Latency and resolution aware task offloading for visual-based assisted driving. *IEEE Transactions on Vehicular Technology*, 68(9):9038–9048, 2019.

[17] Abdullah Lakhan, Muneer Ahmad, Muhammad Bilal, Alireza Jolfaei, and Raja Majid Mehmood. Mobility aware blockchain enabled offloading and scheduling in vehicular fog cloud computing. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4212–4223, 2021.

[18] Haijun Liao, Yansong Mu, Zhenyu Zhou, Meng Sun, Zhao Wang, and Chao Pan. Blockchain and learning-based secure and intelligent task offloading for vehicular fog computing. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4051–4063, 2020.

Privacy preserving techniques for continuous authentication

Hussam Aldeen Alkhafaji
husamu-aldeen.alkhafaji@aalto.fi

Tutor: Sanna Suoranta

Abstract

Authentication is a procedure to guarantee protection and to control access to devices and services. Continuous authentication is sometimes needed to guarantee only allowed usage of the system. Continuous authentication brings many security and privacy concerns. Ensuring confidentiality of sensitive and private data stored in our devices is fundamental to preserve the rights of individuals and the interests of companies. In this work, we first review what authentication and continuous authentication are. Secondly, we showcase the use cases where continuous authentication is necessary. We demonstrate the modes of function of continuous authentication. Thirdly, we address the security and privacy concerns of continuous authentication. Finally, we demonstrate methods and techniques that are used to overcome the aforementioned limitations. We conclude with a reflection on what the future hold for continuous authentication.

KEYWORDS: Authentication, Continuous Authentication, Access Control, Privacy, Cryptography.

1 Introduction

Computational systems are ubiquitous in every aspect of people's lives. Modern societies are surrounded by many small systems, usually known as the Internet of Things or IoT devices that serve various purposes. A prime example of such a device is the smartphone in most people's pockets. While the smartphone is not exactly an IoT device, it encompasses many sensors that can be considered as IoTs separately. These systems usually require some form of authentication to be accessed by users as they contain sensitive data, private data, or both [1].
Authentication is the process of identifying the user by verifying who they claim to be. This process is usually done using a technique known as static authentication (SA), which is done at the beginning of a user session.

Currently, there is a new trend of continuous authentication (CA) [2]. This method authenticates the user continuously throughout the usage period of a system to ensure that the user never changed after the first static authentication [3]. Continuous authentication uses many sources of user data to achieve its goals such as (1) sensor gathered biological data, including blood sugar levels or body prints, (2) behavioral data, including keystrokes dynamics and locations. These new methods to authenticate

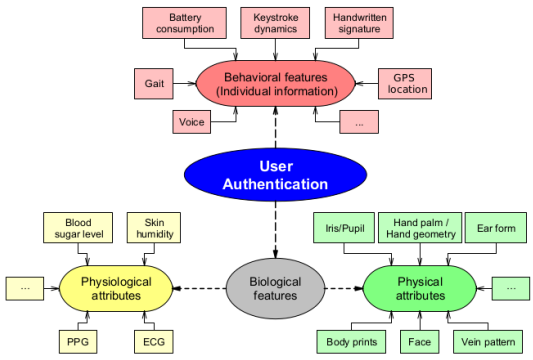


Figure 1. Features scheme for sensor-based user authentication.

Figure 1. [2]

users introduce new security and privacy concerns ,such as how the data is being handled, where the authentication is happening, and what could go wrong. To answer these questions, researchers have studied and invented privacy-preserving techniques applied to continuous authentication. Examples of such studies is the work by Acar et al [4] which offers a non-invasive CA scheme to allow users to access proximity based ser-

vices. The implementation in Chuang et al [5] which offers a lightweight CA protocol for IoT devices. The research in Lopez [6] which offers a privacy aware CA scheme for proximity based access control. The study in Govindarajan et al [7] which offers a privacy preserving protocol for outsourcing CA of smartphone users with touch data. The purpose of this paper is to highlight the current issues that faces continuous authentication and then to showcase the current solutions available.

This paper, defines both authentication and continuous authentication. Then we will explore the use cases of continuous authentication and where it is needed. After that we will explain all the concerns attached to each use case. Finally, the last section will introduce the possible solutions for these concerns and explain how they solve the problems.

2 Authentication and Continuous Authentication

This section briefly defines what authentication is in the context of computer systems before we then explain what is CA. This section elaborates how they cooperate to achieve common goals.

2.1 Authentication

Authentication can be defined as the act of verifying an identity [8]. Authentication is generally categorized into three sub-categorizes depending on the method the user uses to authenticate, These are:

- Something you have, such as a ticket.
- Something you know, such as a password.
- Something you are, such as a fingerprint.

Authentication exists in different forms and types. The most common form of authentication is Static Authentication (SA) where the users authenticates themselves once before they start using the system. Recently other types of authentication has emerged, such as the Strong authentication used by various governments as well as continuous authentication, which will be discussed next.

2.2 Continuous Authentication

Continuous authentication [3] is a passive form of authentication that is not meant to replace the traditional SA discussed above but rather, it is meant to help achieve identity confirmation of the user on an ongoing basis. The main advantage of CA is that it does not interrupt the normal workflow of the user while adding an additional layer of much needed security. This layer of security is usually implemented using machine learning and depends on a variety of factors, including data comprising biometric, behavioral, and context-oriented characteristics. A typical implementation of CA would be composed of

- the continuous stream of data collected from the user.
- a user template or a user profile that is being used as the basis of judgment.
- a reward / penalty system where a certain threshold would lead the user to be "suspicious" and cause the system to lock.

The location (local or cloud) and specifics of those three parts differ from one implementation to another. This paper discuses CA in general, and specifies certain implementations when deemed necessary.

3 Example use cases of CA

Continuous authentication is used in many use cases in the real world. This section examines them briefly to highlight the importance, relevance and prominence of CA.

3.1 Military

One of the security issues that concerns military personnel is leaving their devices unattended. This issue can lead to unauthorized persons gaining access to information and assets. CA would solve this issue by monitoring the user's behavior at all time. In a scenario where CA is implemented correctly, the device would lock itself before the unauthorized person can do any substantial damage. In a military context, security is prioritized

over usability. This is to protect classified data as well as protect vital structures of a government. This context is a perfect use case to implement CA.

3.2 Civil

Many used cases have been proposed for CA in a civil context. This section will demonstrate two examples.

Hospital
Hospitals store a large quantity of sensitive data that belong to patients [9]. Doctors in these hospitals need frequent access to this data throughout their shifts. Usually, the doctor will have to login using static authentication to access a patient's data. One issue concerning this context is doctors are usually quite busy and risk leaving their sessions open. Another issue is that the frequent access of this data means frequent reauthentication, which can be cumbersome for a doctor, especially if they need different credentials for each patient. CA can solve this problem by profiling each doctor's behavior and continuously authenticating the usage while the doctor accesses patient's data throughout the day. CA will work alongside SA to ensure that the sensitive data is protected at all times.

Remote Work
Due to the recent pandemic, many companies have switched to remote work. This switch to "work from home" introduced with it many security concerns. One such security concern is that employees get to work on companies propriety business secrets from their homes. As soon as such secrets leave the premise of the company's building, it is suspect to many attacks. As mentioned before, one such attack is leaving a device unattended. This behavior would allow for attackers to steal important data and information from companies laptops. Another attack is stealing an employee's credentials. It is much easier to do so by scamming them using various social engineering methods. This will also result in an outsider gaining access to a company's data. CA helps in preventing all of these attacks and more. The company will have to decide on the exact implementation of the authentication itself in a way that suits their needs.

4 Flaws and concerns in CA

As it is usually the case with every new technology, it has its advantages and disadvantages. This section demonstrates the flaws and concerns relating to the usage of continuous authentication.

4.1 Security of CA

We define the security of continuous authentication in two points:

- The performance of a specific modality of CA. The performance includes accuracy, false acceptance rate, and false rejection rate.

- The feasibility of forging a biometric modality using different attacks, such as mimicry attacks, template leaking attacks.

4.2 Privacy concerns

The privacy issues [10] stems from two different perspectives. The first is the possibility of stealing the user templates or models from the device that has all the collected data used by CA. The second is using cloud continuous authentication. This term refers to either processing the data in the cloud to access a cloud service or authenticating a user in the cloud to allow usage of a locked device. These privacy challenges vary depending on the various modes and modalities of continuous authentication. The modes differ in the source of data that is used to achieve authentication .

- Continuous authentication that monitors personal information, such as location data, mouse movements and keyboard clicks, and personal device usage. This mode does not provide any privacy to the user as it uses personal information gathered to function.

- Continuous authentication that monitors online presence, such as sites visited and frequency of visits, The time spent in a given site, presence of certain cookies. While this mode does not reveal information directly, much information can be inferred from the data, such as the user's gender and age. The user's usage may disclose vital information, such as fragments of medical history, employer related secrets, or personal in-

formation.

- Continuous authentication that monitors physiological characteristics and other biological and behavioral data, such as facial features, gait and movement patterns, keystroke dynamics and other similar data. This mode of continuous authentication is the most dangerous to user privacy as it uses lot of identifying information. The data collected by this mode is sensitive on its own. However it can also be used to infer further information on the user, such as health related data as well as age and gender.

5 Privacy preserving techniques

This section discusses privacy preserving techniques and algorithms that can be applied to the modes discussed previously. These techniques are meant to surpass the privacy risks that accompany using continuous authentication.

5.1 Homomorphic Encryption

Cryptographic approaches require decryption before authentication to be able to achieve access control using continuous authentication [11]. This creates a privacy issue discussed above in all modes of continuous authentication, an untrusted cloud provider can see the biometric data after decryption. The same problem occurs if a device is stolen and the decryption keys are in the device itself. One privacy preserving technique that can be used is Homomorphic encryption. Homomorphic encryption is a type of encryption that allows certain computations to be done on encrypted data. This means that there is no need to decrypt the data first to manipulate it. This allows a user using continuous authentication to send its encrypted data to an untrusted cloud server. The server can do comparisons on the encrypted data without being able to decrypt it. The server then returns a response to the user. The cloud server has no means to discover the template used nor the data being sent hence preserving the privacy of the user.

5.2 Zero knowledge proofs

Zero knowledge proofs is a set of encryption schemes that allows a prover (the user) to authenticate itself without having to share the data that demonstrates their identity [12]. This is done through a process of mathematical algorithms that proves a statement to be true without revealing any additional data. In Continuous authentication, the user can prove their identity to an untrusted cloud server or to a local server using this process without having to share personal information and thus preserving privacy.

5.3 Cancel-able biometrics

Cancel-able biometrics is a privacy preserving technique used specifically to hide physiological biometric data [13]. The concept of cancel-able biometrics is similar to hash functions as well as Homomorphic encryption. biometric data goes through a non-reversible process to reach a state where it can be used for authentication without being able to discover the original data. In practice, biometric data can be images of body parts, such as the face or the eye pupils. This technique help in distorting the image in a way that it is still usable for authentication but the original image cannot be constructed from it anymore.

6 Analysis

Privacy preserving techniques give continuous authentication a chance to be implemented on a larger scale for commercial use. Introducing continuous authentication to civilian users and corporations will certainly augment the overall security. However, there is still much to be researched and many attack vectors to explore. Technology similar to continuous authentication take time to mature. A haste deployment of continuous authentication might lead to disastrous outcomes for users, thus users should be informed of how continuous authentication work and what data is being collected. Consent will play a major role in the deployment of continuous authentication. The future will tell how these intricate details will be handled and if continuous authentication will be viable for usage.

7 Conclusion

This paper has introduced Static Authentication as well as Continuous Authentication. It explained the differences between the two and why both of them are needed to achieve better access control. The paper demonstrated real life examples where this superior access control is needed. The paper has also reviewed viable techniques used to preserve the privacy of users using Continuous Authentication. The techniques presented offer good privacy to usability ratio as they add little overhead. There is still much speculation if continuous authentication will be part of our secure cyber future. Research is already underway to fully comprehend all the factors that need to be settled. Further more, much work is still to be conducted to further secure Continuous Authentication. As yet, all the results obtained point to a future where continuous authentication is present.

References

[1] F. H. Al-Naji and R. Zagrouba, "A survey on continuous authentication methods in internet of things environment," *Computer Communications*, vol. 163, pp. 109–133, 2020.

[2] A. F. Baig and S. Eskeland, "Security, privacy, and usability in continuous authentication: A survey," *Sensors*, vol. 21, no. 17, 2021.

[3] L. Hernández-Álvarez, J. M. de Fuentes, L. González-Manzano, and L. Hernández Encinas, "Privacy-preserving sensor-based continuous authentication and user profiling: A review," *Sensors*, vol. 21, no. 1, 2021.

[4] A. Acar, S. Ali, K. Karabina, C. Kaygusuz, H. Aksu, K. Akkaya, and S. Uluagac, "A lightweight privacy-aware continuous authentication protocol-paca," *ACM Trans. Priv. Secur.*, vol. 24, no. 4, sep 2021.

[5] Y.-H. Chuang, N.-W. Lo, C.-Y. Yang, and S.-W. Tang, "A lightweight continuous authentication protocol for the internet of things," *Sensors*, vol. 18, no. 4, 2018.

[6] I. Agudo, R. Rios, and J. Lopez, "A privacy-aware continuous authentication scheme for proximity-based access control," *Computers & Security*, vol. 39, pp. 117–126, 2013.

[7] S. Govindarajan, P. Gasti, and K. S. Balagani, "Secure privacy-preserving protocols for outsourcing continuous authentication of smartphone users with touch data," pp. 1–8, 2013.

[8] S. Z. Syed Idrus, E. Cherrier, C. Rosenberger, and J.-J. Schwartzmann, "A Review on Authentication Methods," *Australian Journal of Basic and Applied Sciences*, vol. 7, no. 5, pp. 95–107, Mar. 2013.

[9] F. Kargl, E. Lawrence, M. Fischer, and Y. Y. Lim, "Security, privacy and legal issues in pervasive ehealth monitoring systems," in *2008 7th International Conference on Mobile Business*, 2008, pp. 296–304.

[10] I. C. Stylios, O. Thanou, I. Androulidakis, and E. Zaitseva, "A review of continuous authentication using behavioral biometrics," in *Proceedings of the SouthEast European Design Automation, Computer Engineering, Computer Networks and Social Media Conference*, ser. SEEDA-CECNSM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 72–79.

[11] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, ser. CCSW '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 113–124.

[12] D. Gabay, K. Akkaya, and M. Cebe, "Privacy-preserving authentication scheme for connected electric vehicles using blockchain and zero knowledge proofs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5760–5772, 2020.

[13] V. M. Patel, N. K. Ratha, and R. Chellappa, "Cancelable biometrics: A review," *IEEE Signal Processing Magazine*, vol. 32, no. 5, pp. 54–65, 2015.

Object tracking for mobile augmented reality

Elias Arte
elias.arte@aalto.fi

Tutor: Ashutosh Vaishnav

Abstract

This paper introduces general object tracking process, reviews existing object trackers, and evaluates how feasible these object trackers are for mobile augmented reality. The reviewed object trackers are a client-side object tracker based on SURF and system that divides process so that the object detection is on server-side and the object tracking is on client-side. The evaluation for mobile augmented reality focuses on challenges, such as performance and mobility.

KEYWORDS: augmented reality, AR, object tracking

1 Introduction

In recent years, augmented reality has got increasingly popular. Augmented reality refers to a view of the real world that has been enchanted with computer generated graphics. The use cases vary from mobile games to educational solutions to industrial repair and maintenance solutions of equipment.

However, object tracking complicates some possible augmented reality applications. While multiple different methods for tracking objects exist, not all of these can be used for all augmented reality applications. Some

of these applications track real world objects and replace those objects with computer generated objects. This requires near real-time tracking of objects or the application is unpleasant to use. Another problem related to near real-time tracking is the computing power of mobile devices. While near real-time tracking might be possible on a desktop PC or a server, mobile devices might not be efficient enough.

As previously mentioned, multiple different methods for tracking objects exist, thus this paper will only introduce two different solutions. First one is a object tracking system running locally that can in theory track any object the user wants to. Second one is a system that is divided between server-side and client-side and can track objects entered into a database.

This paper is organized as follows. Section 2 presents general information about object detection and tracking. At first, it will present two sub-issues, object representation and feature selection, which are critical for object detection and tracking, and then continue about object detection and tracking. Section 3 will present two real object tracking solutions and their technicalities. Then, in Section 4, the paper will discuss how feasible the presented solutions in Section 3 are for augmented reality. Finally, Section 5 will end the paper with some concluding remarks.

2 Object detection and tracking in general

Object tracking is a process that can be divided into multiple smaller components. These components are introduced as object representation, feature selection, object detection, and object tracking in this paper. This section introduces the components and how to address them when creating an object tracker based on the survey created by Yilmaz et al. [1].

2.1 Object representation

According to Yilmaz et al. [1], when tracking an object, it needs to be defined as the point of interest. This is done by creating the object representation. The representations are divided into shape and appearances categories. Some of the commonly used shape representations are points, primitive geometric shapes, object silhouette and contour, articulated shape models, and skeletal models. Meanwhile, common appearance representations include probability densities of object appearance, templates, active appearance models, and multiview appearance models.

It is important to choose the correct representation for the object, as some representations are more suitable than others. For example, points are suitable for tracking smaller objects and primitive geometric shapes are more suitable for simple rigid objects, while complex shapes, such as humans, object contour or silhouette might be most suitable representation.

2.2 Feature selection

For object tracking to work, the object has to differ from the objects that are not interesting. This means that the object needs to have some feature to it, that can be used to identify it. In general, this means the visual uniqueness of the object. Yilmaz et al. [1] present some feature examples such as the object's color, edges, optical flow, or texture. It is important to choose features that fit the application domain, but also features that work together with the chosen object representation [1].

2.3 Object detection

Object detection is a task where the object is detected from video footage. Object detection is supposed to find the object from the frame in which the object first appears in or from every frame. Previously defined object representation and feature selection are used in object detection, and different object detection methods use different object representations and feature selections.

Yilmaz et al. [1] present four different categories for object detection. The categories are: point detectors, background subtraction, segmentation, and supervised learning. Point detectors find interest points in images at locations with an expressive texture. Background subtraction uses a representation of the scene called the background model. The background model finds deviations of each incoming frame by comparing it to the background model and each change points to a moving object. Segmentation methods partition the image into similar regions and these regions should represent the objects. Supervised learning uses a large collection of samples of objects and machine learning to detect objects [1].

2.4 Object tracking

The purpose of object tracking is to track the detected object and generate its trajectory from each frame of the video input. There are two different

possible approaches to tracking according to Yilmax et al. [1]. The tracking can be performed separately or jointly. Separately performed tracking means that the tracker detects object and its location from each frame and then connects objects across frames. Jointly performed tracking however, uses information of the object location and region from previous frames and estimates correspondence using that information. In either approach, the object is then represented using the chosen object representation.

3 Object tracking methods

Object tracking has a long history, and many solutions have been developed over the years to solve it. This section will review two different approaches to object tracking. Both approaches use SURF for object detection. However, the first approach runs on client-side and can track any object, while the second approach runs both on server-side and client-side and can only track objects in a database.

3.1 A robust object tracking algorithm based on SURF

Speeded up robust features (SURF) algorithm contains three sections: interest points detector, interest points descriptor, and interest points matching [2]. Interest points are distinctive locations in the image, such as corners, blobs, and T-junctions. The purpose of interest point detector is to find the same interest points from different frames. SURF gets interest points by approximating a Hessian matrix using a Fast-Hessian Detector. SURF also uses box filters and integral images to reduce the computational complexity of the Hessian matrix. Next, interest point descriptor (feature) vectors are created. Feature vectors represent the neighbourhood of every interest point and are created using Haar wavelets. Lastly, the descriptor vectors are matched between different frames. Usually, the matching is based on the distance between the vectors, using the Mahanalobis or Euclidean distance [2] [3].

Zhou and Hu [3] use the above process for interest points but have implemented a different, two-stage, matching algorithm. In the first stage, they use a traditional SURF matching method, which uses Euclidean distance. The matching formula is as follows:

$$\frac{nearest\ \ distance\ \ PP'}{second\ \ nearest\ \ distance\ \ PP"} \leq T1 \tag{1}$$

In the formula, P is interest point in the current frame, and P' and P" are

interest points in some previous frame. If the ratio of PP' and PP" is less than the threshold T1, then the points are matched. T1 is here set as 0.8, as it will allow the system to remove some wrong matching points, while ensuring that the system can extract enough matching points.

In the second stage, Zhou and Hu [3] use dominant orientation to remove more wrong matching points. Dominant orientation can be got from the SURF interest point descriptor, which also contained the location information for the first stage. The angle difference of the dominant orientation between two initial matching points that were obtained from the first stage is then calculated. Next, an angle difference histogram is established and maximum Bin area is found. In theory, as the angle difference of any two matching points should be close to others, Zhou and Hu select a range of ten percent around the maximum Bin area as the angle difference range of right matching points. Thus, the formula for the angle difference range R of right matching points is as follow:

$$\frac{(N_0 + N_1) \times 0.9}{2} < R < \frac{(N_1 + N_2) \times 1.1}{2} \tag{2}$$

N_1 is the value of the maximum Bin area, N_0 is the value of the left Bin area next to N_1, and N_2 is the value of the right area next to N_1.

Next, as the matching process in the system created by Zhou and Hu [3] relies on templates, which are generated from previous frames, the templates need to be saved and updated on certain conditions. First, they establish a template cache which can hold 10 templates in total. Each template contains information about the object region, interest points, and number of matching points. If the cache does not hold ten templates a new template will be added to the cache. In a case that the cache already holds ten templates, the oldest one will be replaced by the new one. However, as the tracking performance depends on the number of matching points between the frame and the template, the templates are not updated on each frame, but rather when the new frame and template matching points amount is greater than set threshold T2.

For tracking, Zhou and Hu [3] use a two-stage method again for better accuracy, as over time with template updates and wrong matching points the system will start to drift. In the first stage they run the two-stage matching between the new frame and new template. New template is the template with the largest number of matching points. Then they calculate the center of matching points gravity offset and determine the initial position of the object. The second stage is similar to the first stage, with

the exception that they run the two-stage matching between the initial position obtained in the first stage and a fixed template. The fixed template contains the information of the very first frame, as it contains the most original information of the object. The final object position is thus obtained from the first stage initial position and fixed template. In cases where the frames do not contain sufficient matching points to track the object Zhou and Hu [3] adopt Meanshift [4] algorithm to track the object.

3.2 Server-side object recognition and client-side object tracking

Gammeter et al. [5] propose a solution that divides some of the task to a server, while keeping rest of the task on client-side. Their solution runs object detection on server-side to take advantage of a large database of objects to gain near real-time results back from the database, while keeping the object tracking on client-side to keep the server interactions at minimum. This approach allows them to gain near real-time tracking on the client-side.

For object detection, Gammeter et al. [5] follow closely their previous work [6], which was used for landmark recognition, and combine it with a media cover (books, CDs, DVDs) recognition service offered by the company kooaba. Gammeter et al. [5] assume that kooaba builds on the same approach, using local features and visual vocabularies, as their landmark recognition system.

Their landmark recognition system has crawled over 12 million images that were geotagged on Flickr, and clustered them into 300 000 objects. Gammeter et al. [6] use the approach proposed by Quack et al [7], which provides a system for building the cluster database that is fully automatic. The system sends geo-tagged queries to Flickr, runs clustering process to images that are geographically close to each other by checking for their visual similarity using SURF features [2] and bundles similar objects together, thus creating database of clusters where each object represents some landmark. Then metadata from Flickr is added to it from the cluster. For each object (landmark), a bounding box is then calculated. The bounding box is calculated by using feature matches, which were used for creating the object clusters at first, and their confidence values. An estimated bounding box is drawn around features which have high enough confidence value [6].

Now with the database established, it can be used to detect objects from

new images. Gammeter et al. [6] quantize the cluster SURF features using a visual vocabulary of 1 million visual words which was learned using approximate k-means [8]. Every image is thus represented as a set of visual words and each incoming query image also get labeled with a set of visual words. The set of visual words of the query is then matched against the whole database using set intersection as distance measure. For 500 closest candidate images a geometric consistency check is performed by using RANSAC estimation of the homography mapping of feature matches. Instead of matching visual words as correspondence pairs for the homography estimation, which is the common approach, Gammeter et al. [6] use a different approach. They reestimate the correspondences based on the second nearest neighbor distance ratio of the original features, yielding far better retrieval accuracy. However, this comes with a drawback that each SURF feature file has to be loaded into memory from disk, which is slow. This is overcome by using a product quantizer [9], which decomposes the feature space into a Cartesian product of low dimension subspaces. This enables the system to keep the features in memory instead of having to load them from the disk each time. In final step, the database images vote for their clusters, and most voted cluster is the result [6].

A working object detection system can now be used for object tracking. To track the object, it means the object location needs to be updated and labeled for every frame of video input. Multiple different approaches can be used for tracking the object, but Gammeter et al. [6] use a combined technique. Their approach combines sensor based tracking and visual tracking to gain access to advantages of both of these methods.

For visual tracking Gammeter et al. [6] use FAST [10] corners combined with 8x8 pixel image patches as feature descriptors. Their system has two different visual tracking modes, direct tracking and incremental tracking, which are used complement each other. With each object detection request to the server, it saves the features of current frame as reference features. Direct tracking attempts to match the features between the current frame and reference features. If the matching was successful then the object can be very accurately located in most cases. However, the tracking can fail for multiple reasons with the most obvious being the object leaving the field of the camera. In these cases the system uses incremental tracking, which matches the features between current frame and the previous frame. This estimates the interframe motion and the location is updated based on the

estimated motion. However, incremental tracking is subject to drifting, therefore if the tracker remains in the mode for several frames, the system will initiate a new object detection request [6].

Instead of using the sensors from a smartphone for directly tracking the location, the sensor data is used as a backup for recovering from errors. The smartphone's pose can be estimated using accelorometer and magnetic sensors and this pose is not prone to drift. Gammeter et al. [6] let direct tracking to track the object accurately and update the sensor tracker's world view to match the visual tracker. However, when direct tracking is unsuccessful and incremental tracking that is prone to drifting, is used, the system compares the locations between visual and sensor tracker. After over 10 successive frames do not coincide, it is assumed that the object has been lost, and new object detection request will be made and visual tracker's world view will be reset to match the sensor tracker's world view [6].

4 Evaluation for augmented reality

This section will discuss how feasible the solutions presented in Section 3 are for augmented reality usage. Rabbi and Ullah [11] categorize challenges in augmented reality tracking to performance challenges, alignment challenges, interaction challenges, mobility challenges, and visualization challenges. The main challenge from the perspective of this section is performance and mobility. Augmented reality applications need near real-time tracking for the application to be comfortable to use. In addition, the object tracking should be light enough to run on mobile devices. Heavier algorithms can be used for object tracking by taking advantage of cloud or server-side computing, but this also causes a delay to the tracking, which should be near real-time.

4.1 A robust object tracking algorithm based on SURF

Zhou and Hu [3] created a object tracking algorithm based on SURF and its technicalities were explained in Section 3. In their solution, the user has to select a trackable object with a bounding box from the first frame of video footage. In theory this means that their algorithm is capable of tracking any object the user wants to. No accuracy statistics such as pixel errors were not provided for the algorithm but in example footage

of their algorithm, it shows that their tracker is accurate in three different demos and can overcome object occlusion with their template update system. This makes it a very capable object tracker for augmented reality in general. If their solution would be combined with an automatic object detector, it could be used in multiple different augmented reality applications. However, with the growing use cases of augmented reality, this object tracker might not be the best solution for most of them. In the end it still seems like a very capable object tracker in general and might see usage especially in demos that do not need to be perfectly accurate.

Augmented reality also needs near real-time tracking and Zhou and Hu [3] do not provide much information about the performance of the object tracker other than that it was run on an Intel Core i5 CPU@2.67GHz. However, they compare the algorithm with a similar algorithm by Du et al. [12], which was running on a weaker Intel Core i3-380 CPU@2.53GHz, and it was able to process 12 frames per second of a 640x480 pixels resolution video. Now, assuming the algorithms perform similarly, the algorithm created by Zhou and Hu [3] should be able to process frames fast enough on a modern mobile phone considering the performance difference between a year 2010 x86 architecture CPU and a year 2022 ARM CPU.

4.2 Server-side object recognition and client-side object tracking

While previously introduced object tracker by Zhou and Hu [3] tracked moving objects from video footage and ran the algorithm locally, Gammeter et al. [5] developed an algorithm that tracks both stationary and non-stationary objects and uses server-side computing to boost the performance. Their algorithm takes advantage of a huge database of objects and their SURF features on a server for object detection, while performing tracking locally on a mobile device using FAST and the sensors of the device. This solution is not a general one such as the one developed by Zhou and Hu [3], but a more specific solution. As Gammeter et al. [5] use a database of objects for detection, all the objects that need to be trackable have to be added to the database by photographing the object from multiple different viewpoints. However, their solution seems suitable for example tourism applications as their solution can detect popular landmarks, or museums if pieces of art were to be added to the database.

Gammeter et al. [5] reached a performance of about 12 frames per second on 480x320 pixels resolution on a Google Nexus One phone, which

was released in 2010, meaning modern 2022 phones could reach a higher performance. Android platform was also not yet fully optimized for vision based AR applications and there was a substantial overhead due to unnecessary memory allocation and garbage collection when grabbing frames from the camera [5]. While their sensor tracking was quite inaccurate with typical errors on the order of 50 pixels, their direct tracking error was at most 4 pixels, and incremental tracking after 50 frames was below 8 pixels. Sensor tracking is also not directly used for tracking the object but rather used to reset the tracking process in case of failures which it can efficiently detect. As the solution relies on server-side object detection, it makes it dependable on internet connection and the performance might suffer under varying internet connections.

5 Conclusion

The goal of this paper was to introduce the basics of object tracking and review some existing solutions. Important themes were mobile augmented reality and the performance and mobility of the reviewed solutions.

Both solutions reviewed in this paper were working solutions. One solution was to run object tracking on client-side and to use SURF for detecting the object. This object tracker can be used for implementing general applications when combined with an automatic object detector. Other solution was to run object tracking on client-side while using a server-side object detection with a database of objects. This object tracker can be use for more specific applications and could see usage in tourism or museums for example.

Based on the reviewed content, object tracking is on a level where it can be used for mobile augmented reality. However, the accuracy and performance of existing solutions can still be improved with new algorithms and new research. In addition, with the increasing popularity of augmented reality applications, new application areas might appear where existing solutions do not work.

References

[1] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13–es, dec 2006.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up ro-

bust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[3] Dan Zhou and Dong Hu. A robust object tracking algorithm based on surf. In *2013 International Conference on Wireless Communications and Signal Processing*, pages 1–5, 2013.

[4] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. volume 2, pages 142–149 vol.2, 02 2000.

[5] Stephan Gammeter, Alexander Gassmann, Lukas Bossard, Till Quack, and Luc Van Gool. Server-side object recognition and client-side object tracking for mobile augmented reality. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 1–8, 2010.

[6] Stephan Gammeter, Lukas Bossard, Till Quack, and Luc Van Gool. I know what you did last summer: object-level auto-annotation of holiday snaps. In *2009 IEEE 12th International Conference on Computer Vision*, pages 614–621, 2009.

[7] Till Quack, Bastian Leibe, and Luc Van Gool. World-scale mining of objects and events from community photo collections. In *Proceedings of the 2008 International Conference on Content-Based Image and Video Retrieval*, CIVR '08, page 47–56, New York, NY, USA, 2008. Association for Computing Machinery.

[8] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[9] Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[10] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[11] Ihsan Rabbi and Sehat Ullah. A survey of augmented reality challenges and tracking. *ACTA GRAPHICA*, 24:29–46, 02 2013.

[12] Kai Du, Yongfeng Ju, Yinli Jin, Gang Li, Yanyan Li, and Shenglong Qian. Object tracking based on improved meanshift and sift. In *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pages 2716–2719, 2012.

Privacy in Authenticated Key Exchange Protocols

Otso Pohjola
otso.pohjola@aalto.fi

Tutor: Chris Brzuska

Abstract

This literature review is an examination of privacy preserving authenticated key exchange (AKE) protocols. The goal is to understand how privacy has been taken into consideration in four AKE protocols, TLS, Signal, OTR and Noise. These protocols were chosen, since there was enough privacy centered research conducted of them to form a clear picture. They use methods, such as encrypting the authentication step, sharing a signature key and periodically publicizing old message authentication code keys to preserve the privacy of the AKE participants.

KEYWORDS: Privacy, AKE, TLS, Signal, OTR, Noise

1 Introduction

Authenticated key exchange (AKE) means authenticating key exchange parties to confirm their identity, in addition to establishing a secure communication channel [1]. Probably, the most common example of an AKE implementation is TLS [2], a protocol, which is often used in web communications. Private messaging applications utilize AKE as well, while attempting to protect the identity of the users at the same time [3][4][5]. Suitable examples of these are, e.g., Signal, OTR and Noise protocols

[6][7][8]. This paper reviews these four AKE protocols from the viewpoint of privacy properties called anonymity, deniability and identity hiding.

In the context of AKE, anonymity can be defined as hiding the identity of each AKE participant from everyone else, meaning its peers and external adversaries [1]. In real-life applications, it is difficult to achieve this for everyone, as usually at least one party authenticates during the session initiation. In addition, anonymity cannot be produced, and the AKE schemes can only attempt to preserve the current state defined by the underlying infrastructure [9].

Deniability focuses on maintaining repudiation, and it makes a party's involvement in a key exchange improvable. Note that this still allows the protocol to leak other identifying information [1], but only as long as a party's participation cannot be proven based on it. In practice, deniability promises that even when an AKE participant might trust the identities of other peers, they cannot convince anyone else to believe the same.

The last property that this paper takes into account is identity hiding. Identity hiding aims to hide the identity only from the external adversaries [1]. For example, in a client-server architecture, while the server loses its anonymity by authenticating to the client, it can still preserve identity of the client hiding property by encrypting any of its identifying data that is transferred.

The existing research analyzes these properties in relation to AKE protocols [10][3][4] and also suggest new models and methods that could help to improve them [11][12]. The paper is structured as follows: Section 2 considers the four different AKE protocols, starting with TLS and continuing with instant messaging related protocols, Signal and OTR. Section 2 finishes with discussion regarding the complicated Noise protocol, after which Section 3 considers the differences and similarities between these AKE protocols. Finally, section 4 concludes the report.

2 Authenticated Key Exchange Protocols

2.1 TLS 1.3

Transport Layer Security (TLS) is used to protect web application communications varying from HTTPS to SMTP and VoIP. Its primary use cases are encrypting transferred data, authenticating the parties and verifying

data integrity [13] [10].

In the previous version, TLS 1.2, identifying information, such as digital signatures and certificates, were sent in clear [11]. Although the transport layer messages were encrypted with a secret key, identities were transmitted before the establishment of the key and thus publicly observable. The current version, TLS 1.3, was released in 2018 by Internet Engineering Task Force (IETF) [13]. It shifted the encryption to earlier phase, making authentication operations also encrypted. Another change was the addition of Encrypted Server Name Indicators (ESNI). It is an encrypted version of SNI, a field that used to leak TLS 1.2 traffic destination to passive adversaries.

Ghada Arfaoui, Xavier Bultel, Pierre-Alain Fouque, Adina Nedelcu and Cristina Onete report that TLS 1.3 itself does not have innate privacy weaknesses, but some of its properties could be abused [10]. TLS 1.3 supports two different modes: full handshake and session resumption. According to Arfaoui *et al.*, the full handshake mode is more frequently used and has less weaknesses. In this mode, a client does an authenticated Diffie-Helman (DH) key exchange with a server, while also offering forward secrecy. An active "man in the middle" attack is the only privacy related attack that Arfaoui *et al.* could apply to full handshake mode. In this case, protecting both identities is not possible, since the initiator has to disclose its identity before authentication occurs. This weakness is not due to a technical detail in TLS 1.3, but rather because of its design choices.

If TLS 1.3 session resumption mode is used, a server sends a session ticket to a client. This ticket is used to avoid initiating the authentication sequence next time the client connects to resume the previous handshake, and it is transmitted in public. Session resumption mode weakens privacy, since the ability to resume sessions means automatically an ability to differentiate based on a client ticket state. Moreover, distinguishing client identities becomes simpler, if any custom property, such as an identification number or an incrementing number is added to the ticket. Continuing the same session in a pre-shared key only mode also removes any forward secrecy.

2.2 Signal

Signal is an AKE protocol used by the messaging application "Signal", an app that is widely regarded to have high cryptographic standards [3]. It

especially focuses on protecting identity through deniability. This is a desired attribute for messaging applications, since it assures that the messages cannot be traced back to the AKE participants afterwards. Having a deniable messaging system brings an application closer to a face-to-face conversation, where the only identifying link to data remains in the memory of participants [5]. While Signal does not offer online deniability, its offline deniability has strong cryptographical proof [3]. However, some conditions are related to this, which will be discussed later.

Signals deniability mainly originates from the 3DH protocol [3]. The protocol bases on creating a shared secret by hashing concatenation of ephemeral keys (X and Y in the figure below) and static public keys (A and B). Participants also use their private DH values (a and b) when calculating the key K, authenticating themselves. Any participant can calculate the same session key, therefore the encrypted messages cannot be linked to any single party, and the recipient could even be sending messages to itself from outsider's perspective.

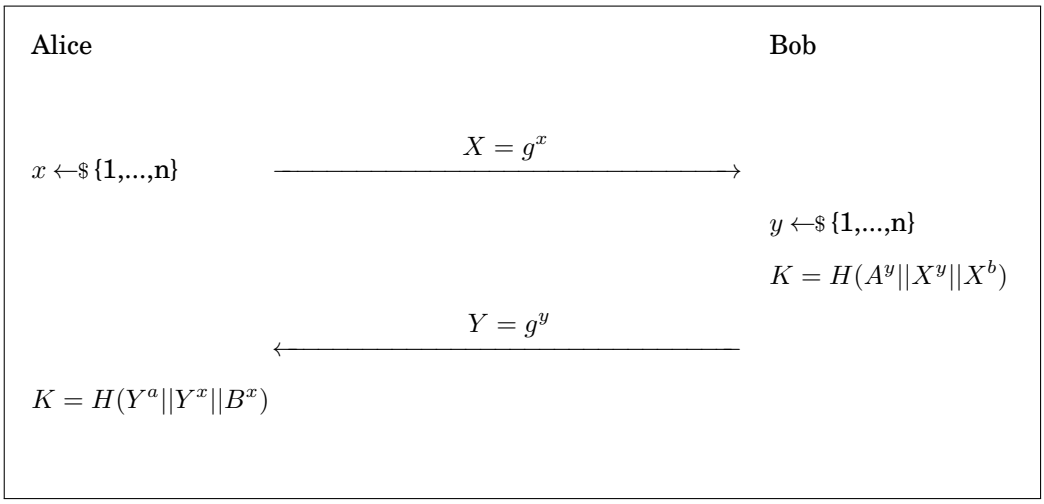


Figure 1. The 3DH protocol, where the public keys of Alice and Bob are A = g^a and B = g^b respectively [3]. H is a suitable hash function and || denotes concatenation.

3DH is extended by the Extended Triple Diffie-Hellman (X3DH) protocol [3][6]. The X3DH variant adds a third party to 3DH: A key distributor server is used for storing the ephemeral public keys, allowing them to be used for establishing a 3DH handshake even when the participants are offline [14]. The initial public keys are signed to protect against impersonation attacks by the server. The server also takes care of caching sent messages, when the recipient is offline.

In addition to X3DH, Signal implements a double ratchet step that occurs during the message exchange. It is utilized for validating the existing session and for generating a new session key using a key deriva-

tion function (KDF) [3][6]. The ratcheting step can be further divided into symmetric or asymmetric types, the difference being the input fed to the KDF. Symmetric ratcheting step only inputs the current session key, while asymmetric also calculates a new DH output and feeds it to the KDF. Generating new DH values periodically ensures forward and backward secrecy.

The use of a key distribution server raises a problem related to identity hiding. The signed public keys that it stores cannot be encrypted, and the server can easily identify the parties initiating the handshakes [14]. However, the server is publicly accessible, therefore the signed key could have been uploaded by anyone [3], which offers a certain level of deniability.

Additionally, there is a forward secrecy weakness related to the double ratchet step. To save resources, asymmetric ratcheting and a new round of DH is initiated only when a previous message is sent by the other party [14]. For this reason, an adversary that possesses a stolen session can decrypt incoming messages continuously as long as a reply is not sent back. Having the session key get stolen is not a far fetched scenario, since Signal stores the session key on a secondary location, such as a hard drive. This is required for Signal protocol to work offline.

Perhaps surprisingly, current cryptographic proofs of Signals offline deniability rely on the Knowledge of Diffie-Hellman (KDH) assumption. The KDH assumption is used to simulate situations, where a malicious participant pre-calculates a partial result of the handshake, or where they frame another participant by making them interact with a randomly generated key, would be impossible to simulate [3].

2.3 OTR

Currently, OTRv3 is the most popular Off-the-Record (OTR) messaging version, but OTRv4 is also under development. Messaging programs utilizing it can be found on various platforms, e.g., OS X, Unix, iOS and Android [5]. Just as Signal, OTR is a protocol designed for messaging applications that shall provide a strong deniability guarantees. Despite that, unlike signal, OTR is almost entirely an online based solution, and it is designed only for two participants [14]. OTR also encrypts public keys during the initial handshakes, which is something Signal cannot achieve, as its X3DH server consumes unencrypted keys.

OTR consists of four steps [5].

1. An unauthenticated DH key exchange is used to setup a secure channel. This will allow a mutual authentication over SIGMA AKE protocol to be performed without revealing the identities to the public [7]. After the authentication, the participants will know their partners public DH key, another shared value s and that s is also possessed by their partner.

2. Communication can now be encrypted by combining each participants DH keys. In addition, each message is accompanied with a MAC, calculated using s [5][7].

3. The third step is re-keying, where both the encryption key and the MAC keys are re-negotiated incrementally, after a set amount of messages are transmitted. This renders the old messages unreadable, even for the participants.

4. Finally, the old MAC key is publicized, ensuring that, in theory, anyone could have computed the old message authentication codes.

Having the messages authenticated with MACs, instead of digital signatures, is a crucial detail for deniability in the case of OTR [5]. Since initially, only two people know the MAC key, the participants can reason that the messages were sent by the other. At the same time, making anyone else believe this claim is difficult, as either one of them could have generated the same MAC. Furthermore, after the last step anyone can sign forged transcripts that are indistinguishable from the originals.

There are some privacy weaknesses with OTRv3. Being designed with only real-time communication in mind, saving the messages for long-term offline storage would violate its design principles and puts the privacy of the other party at risk [5]. Another threat to privacy is the lack of anonymity during the handshaking step. One of the details shared via secure channel is a signature, and leaking it would act as a evidence of taking part in the event [14].

OTRv4 addresses these points by supporting offline conversations and by introducing a new signature algorithm, Ring Signature Algorithm [5][14]. The Ring Signature Algorithm consists of a set of public keys and a corresponding private key for each participant. It allows having a deniable signature at the handshaking step, since once again the other party cannot prove not computing the signature using their own private key. Still,

using OTRv4 for offline messaging is not recommended for deniability use cases, since its downside is the initiator losing their deniability [14]. It should be noted that [14] does not clarify this statement any further.

2.4 NOISE

The Noise framework is a collection of protocols that are used to construct DH based key exchange protocols [15][8]. It is fundamentally used for establishing secure communication channels and is utilized by a diverse array of applications, such as WhatsApp, Lightning and WireGuard [15]. The Noise framework is flexible, and it offers protocol patterns that focus on handshake attributes, such as low latency, identity hiding and various expectations of cryptographic key types (e.g. pre-knowledge and lifetime). However, the same flexibility complicates estimating the properties of the patterns and which of them should be applied to a use case. Not every developer pays attention to the impact on identity hiding when planning to transfer long-term identifiers in clear.

The paper "The Noise Protocol Framework" by Trevor Perrin describes the framework and acts as its specification [8]. Furthermore, it goes through the different identity hiding patterns of Noise, and the potential attack vectors applied to each of them. For example, a pattern, where a responder issues a static key, is prone to active probing by an anonymous initiator. These patterns are rated using ten identity hiding levels, ranging from weak properties, such as "Transmitted in clear.", to stronger ones, such as "An active attacker who pretends to be the initiator and records a single protocol run can then check candidates for the responder's public key.". Some of the level definitions are refined further by Guillaume Girol, Lucca Hirschi, Ralf Sasse, Dennis Jackson, Cas Cremers, and David A. Basin [15]. They also discovered that the identity hiding levels do not necessarily get monotonically stronger, even if it might appear the other way at first. This means that, in some situations, you might lose privacy when "upgrading" the pattern from identity hiding level 3 to 5.

Girol *et al.* computed most useful Noise Framework patterns for different use cases, anonymity being one of those [15]. Nonetheless, the analysis did not cover all the identity hiding levels in Noise and thus there is room for future research. The analysis also used an adversary with restricted capabilities, and therefore, a more capable adversary should be simulated. The restrictions were in place, as analyzing anonymity claims of the Noise Framework is computationally challenging. Girol has also

written a hundred page master thesis, which is an in depth review of the Noise framework [16].

A major weakness of the Noise Framework is that its session identifier is not treated as a strictly private value [15]. The session identifier is computable from public values including, in particular, the public keys of the handshake participants. Therefore, given the link between session identifier and public keys, an adversary can identify the handshake participants.

3 Discussion

One topic for future research could be adding deniability to web traffic. It would allow a client to send sensitive information to a server and deny that any communication happened. This might be difficult to achieve when the infrastructure is based on public and private key pairs. Majority of web traffic uses TLS 1.3, but deniability is not applicable in its current state. The traffic is signed with short time private keys, and anyone can confirm that the client is the only one capable of singing the messages.

Noise is another protocol related to web traffic, and it is utilized in at least one VPN solution (WireGuard). Once more, because of using asymmetric keys, it would take extra effort to provide the deniability through shared symmetric secred. The WireGuard users would benefit from deniability, since VPNs are often used for sensitive business. Another issue with Noise is the freedom of choice it offers. Privacy issues seem to infiltrate applications when complicated frameworks are not understood correctly (e.g. inserting plain text session IDs into sessions).

4 Conclusion

This paper has reviewed four AKE protocols and presented their privacy-preserving capabilities. The focus has been on properties interpreted as anonymity, deniability and identity hiding.

TLS 1.3 improves over its previous version by moving authentication inside an encrypted channel and by encrypting the SNI field, thus hiding the identities more successfully. However, utilizing TLS 1.3 in the session resumption mode will diminish its forward secrecy capabilities.

Signal and OTR attempt to create deniability using shared secrets, which

are mostly derived from public keys. Signal preserves offline deniability well, as long as the session key does not leak, or the server is not compromised. Meanwhile, OTRv3 retains the online deniability of the participants, providing their private keys are not lost, or the messages are not saved locally. OTRv4 introduces the Ring Signature Algorithm and support for offline messaging.

The Noise framework documents ten identity hiding levels, but these are not as straightforward as they seem initially. Each level is meant to protect an identity in a specific context, therefore careful consideration is required before choosing the one to use. Noise has been extensively researched by Girol *et al.*, and a more complete review of their work is recommended.

References

[1] Ian Goldberg, Douglas Stebila, and Berkant Ustaoglu. Anonymity and one-way authentication in key exchange protocols, 2013. https://eprints.qut.edu.au/218542/.

[2] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3, 2018. https://www.rfc-editor.org/info/rfc8446.

[3] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. On the cryptographic deniability of the signal protocol, 2021. https://eprint.iacr.org/2021/642.

[4] Keita Emura, Kaisei Kajita, Ryo Nojima, Kazuto Ogawa, and Go Ohtake. Membership privacy for asynchronous group messaging, 2022. https://eprint.iacr.org/2022/046.

[5] Carlisle Adams. Introduction to privacy enhancing technologies: A classification-based approach to understanding pets, 2021. http://www.noiseprotocol.org/noise.pdf.

[6] Signal specification, 2022. https://signal.org/docs/specifications/.

[7] OTR specification, 2022. https://otr.cypherpunks.ca/Protocol-v3-4.0.0.html.

[8] Trevor Perrin. The noise protocol framework, 2018.

[9] Markulf Kohlweiss, Ueli Maurer, Cristina Onete, Björn Tackmann, and Daniele Venturi. Anonymity-preserving public-key encryption: A constructive approach, 2013. https://eprint.iacr.org/2013/238.

[10] Ghada Arfaoui, Xavier Bultel, Pierre-Alain Fouque, Adina Nedelcu, and Cristina Onete. The privacy of the TLS 1.3 protocol, 2019. https://hal.archives-ouvertes.fr/hal-02482253.

[11] Sven Schäge, Jörg Schwenk, and Sebastian Lauer. Privacy-preserving authenticated key exchange and the case of IKEv2, 2020. https://eprint.iacr.org/2020/1519.

[12] Loïc Ferreira. Privacy-preserving authenticated key exchange for constrained devices, 2021. https://eprint.iacr.org/2021/1647.

[13] Cloudflare documentation, 2022. https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/.

[14] Richard B. Riddick. vault1317/signal-dakez: An authenticated key exchange protocol with a public key concealing and a participation deniability designed for secure messaging, 2020. https://eprint.iacr.org/2020/1231.

[15] Guillaume Girol, Lucca Hirschi, Ralf Sasse, Dennis Jackson, Cas Cremers, and David A. Basin. A spectral analysis of noise: A comprehensive, automated, formal analysis of Diffie-Hellman protocols. https://www.usenix.org/conference/usenixsecurity20/presentation/girol.

[16] Guillaume Girol. Formalizing and verifying the security protocols from the noise framework, 2019.

A survey on touch-based continuous authentication systems in mobile phones

Josephus Jasper Limbago
jasper.limbago@aalto.fi

Tutor: Sanna Suoranta

Abstract

Continuous authentication using behavioral biometrics has been explored in recent studies. It has been proposed as an additional security layer to existing user authentication systems, such as personal identification number, passwords and face recognition. This paper presents recent studies related to touch-based continuous authentication systems in mobile phones. It first gives an overview of continuous authentication and behavioral biometrics in mobile phones. Next, it summarizes selected research papers that address the current limitations and problems related to this technology as well as papers with recommendations for better performance. Then, it provides an idea of how users perceive the adoption of this technology and also possible attacks. Finally, it discusses the feasibility of this method of continuous authentication and some suggestions for further implementations. This paper will help researchers, who are getting into continuous authentication in mobile phones, identify what aspects of the technology needs further improvement.

KEYWORDS: Continuous authentication, behavioral biometrics, mobile phones

1 Introduction

In recent years, mobile phones have been an integral part in our daily activities. These devices provide users access to multiple services, such as messaging, banking and more through the different mobile applications. For convenience, people usually allow these applications to store their credentials and other personal information. This poses a serious security risk when a malicious outsider gets access to the device. Different user authentication methods have already been implemented to address this concern. The use of PINs, face recognition and fingerprint recognition are some of the methods available now. These entry-point based methods, however, are useless against intruders that attack after a successful authentication [8]. Continuous authentication (CA), which is a method involving constant monitoring if the legitimate user authenticated at the beginning is still the same person using the device, is proposed as a complementary solution to entry-based methods.

Multiple continuous authentication methods focusing on different behavioral modalities, such as walking gait, touch gestures, multi-modal, input patterns, location familiarity and power consumption, have been investigated [16]. Among these, touch gesture is one of the most explored modalities due to it being a good candidate for a viable biometric [1].

This paper reviews the state-of-the-art of touch-based continuous authentication systems for mobile phones. This paper is organized as follows. Section 2 gives an overview of touch-based CA systems. Section 3 presents different recent studies. Section 4 discusses user acceptance. Section 5 explains possible attacks. Section 6 discusses the feasibility of the technology. Lastly, Section 7 concludes the paper.

2 Continuous Authentication

Different continuous authentication systems have been proposed to work as additional security feature on top of the existing entry-point based authentication mechanisms. These systems either use physiological or behavioral biometrics to authenticate a user [3]. Physiological biometrics use a person´s unique physical features found in the fingers, eyes, and face. Meanwhile, behavioral biometrics use a person's unique behavior that is maintained when doing regular activities. This includes patterns in actions, such as walking, writing and other daily tasks. Behavioral

biometrics have characteristics that make it more attractive to use for authentication compared to physiological biometrics.

Liang et al. [10] characterize behavioral authentication as secure, continuous, transparent, and cost effective. Behavioral authentication is secure since it works in the background when users perform certain activities. This makes it impossible to steal, copy or forge the data used for authentication. Thus, behavioral authentication is secure and robust against smudge attacks, replay attacks, thermal attacks and adversary attacks. Behavioral authentication is continuous and transparent since it continuously does user profiling naturally based on the user´s interactions with the system without constant interruption. It is also cost effective since it uses embedded sensors, such as microphones, touchscreens, and accelerators, which are widely available and thus does not require extra hardware.

Some of the common behavioral biometric traits used for continuous authentication systems on mobile phones are gesture-based authentication, keystroke dynamics, behavioral profiling and gait recognition [11]. Gesture-based authentication involves capturing hand drawn shapes or strokes, represented as ordered pair of numerical coordinates, as the user interacts with the mobile phone. Keystroke dynamics refers to the monitoring of a user´s typing pattern, including the duration and latency of keypresses, sizes, and pressure. This paper will only discuss touch-based continuous authentication (gesture-based and keystroke dynamics) since touch input is the most frequent method we use with mobile phones and therefore can be used to continuously authenticate the user.

Stylios et al. [14] summarize data collection and feature extraction for both behavioral biometrics. For gesture-based biometrics, user actions inputted on the touch screen are converted into gesture output template. These actions contain parameters such as speed, velocity, size, length, direction and pressure, which are unique between users. The collected data from the interaction between the user's finger and the mobile screen generates series of data that include time stamps, finger pressure, finger blocked area, finger orientation, device orientation and the total number of swipes. Meanwhile, for keystroke dynamics, typing input on the mobile device is recorded. The extracted features used for this method are duration, latency, pressure and location.

Authentication systems are commonly evaluated by their accuracy and other related scores. These values are calculated from number of true

positives, true negatives, false positives and false negatives. False Acceptance Rate (FAR) measures how likely an unauthorized user will get accepted while False Rejection Rate (FRR) measures how likely an authorized user will get rejected. Equal Error Rate (ERR) is the point in which FAR is equal to FRR and lower values mean more accurate system [5]. Below are the formulas to calculate these values:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$FAR = \frac{FP}{FP + TN} \tag{2}$$

$$FRR = \frac{FN}{FN + TP} \tag{3}$$

$$TAR = \frac{TP}{TP + FN} \tag{4}$$

3 Recent studies

This section summarizes recent studies on touch-based continuous authentication systems.

Siirtola et al. [13] investigated the effect of context on continuous authentication using touchscreen and accelerometer readings extracted from swipe gestures. The researchers used the Hand Movement Orientation and Grasp (HMOG) dataset since it is the only public multi-modal dataset available. The authors focused on analysing two contexts, phone usage and user's physical activity. The aim was to prove that improving the accuracy of authentication systems can be do by training separate models for specific context. The experiment was conducted using four scenarios: reading a document while sitting, reading while walking, navigating while sitting and navigating while walking. The study compared the performance of context-specific models and general models. Context-specific models were trained and tested from the same scenario while general models were trained using data from all scenarios. Their results showed that context-specific models have lower equal error rate (EER) than general models. For example, the average EER of the reading scenario dropped from 21.3% to 11.7% and 15.8% to 7.0% for the sitting and walking scenario, respectively. The study also concluded that accelerom-

eter signals should only be used with touch features when the user is moving.

Stylios et al. [15] developed BioPrivacy, a behavioral biometrics continuous authentication system that utilizes keystroke dynamics. The system uses Multi-Layer Perceptron (MLP) as the system's classifier. An android keyboard application was built to collect the cellphone keystrokes. Data is sent to an API endpoint, where it is stored in an online database. The features that were extracted from the data are duration, latency, pressure on keys and location points. The experiment had 39 participants, who were all smartphone users and were familiar with the experimental setup. The experiment consisted of 16 sessions that were each 2 minutes long, in which the participants had to input predefined sentence or series of numbers either after immediately seeing them or after memorizing. One participant is designated as the genuine user and the rest are treated as the impostors. MLP was then used to evaluate the system. Their approach achieved an accuracy of 97.18%, EER 0.02%, TAR 97.2% and FAR 0.02%.

Ananya et al. [4] devised the first keystroke dynamics based continuous authentication system that does not need preregistration. This is done by creating the verification template after the initial login instead of during the user enrollment in the training phase. They also developed metrics to evaluate the performance of their system, namely Strokes to False Rejection (SFR) and Strokes to False Acceptance (SFA). SFR and SFA is the mean number of keystrokes to reject a legitimate user and imposter respectively.

Another study that utilized keystroke for continuous authentication is [9]. A software keyboard application was also developed for collecting the data. The classifier used for their approach was Support Vector Machine (SVM). The experimental setup involved 315 participants, who required to write predefined texts. The participants were split into 2 groups, 303 participants entered the text only once while the remaining 12 entered the text 10 times to simulate authorized users. Receiver Operating Characteristics (ROC) curve was calculated for each participant. Some of the participants had an area under the curve less than 0.8 and were hardly distinguishable. The evaluation on the users who were distinguishable had a TPR of %92 at FPR %1.

Researchers also explored the use of multi-modal behavioral biometrics. Mallet et al. [7] combined phone movement and touchscreen data in im-

plementing a multi-modal continuous authentication system. The public datasets, HMOG and BioIdent, were used in training the model. Three classifiers were used namely Random Forest (RF), K-Nearest Neighbors (KNN) and SVM. RF performed the best among the three with an EER of 13.56% while SVM and KNN had an EER of 19.21% and 20.68%, respectively. The lower performance of the SVM can be attributed to the small dataset that was used. High recall scores were reported which meant that the models performed well in identifying genuine users but the moderate precision scores shows that impostors are sometimes not detected. This means that the model has good usability but security should still be improved. It is difficult to manage the tradeoff between these two when developing a real-world CA system but security should be prioritized over usability.

Touch-based CA systems have shown potential as an added layer of security in mobile phones with regards to being unobtrusive, but accuracy problems still hinder the technology from being adapted in the consumer market. This shortcoming in accuracy is mainly attributed to the lack of training samples available. Buriro et al. [6] propose the use of Generative Adversarial Networks (GAN) to augment the swiping gestures dataset with synthetic samples. GAN has already been used for generating synthetic data for image and video applications but their solution, named SWIPEGAN, is the first to explore its use in generating more swiping samples for behavioral biometrics. The study used the DRIVER-AUTH, a publicly available data set containing 10,320 swiping samples from 86 users. The accuracy of the classifier improved, with TAR increasing from 84.66% to 91.65% and the FAR decreasing from 14.78% to 11.04%, after adding the generated synthetic samples to the original training samples. This approach is particularly useful for one-class classifiers, which need larger number of samples to perform well compared to two-class classifiers.

4 User acceptance

Continuous authentication is a new process that has not been incorporated as a standard security mechanism in mainstream mobile phones. It is important to take into account how they feel about introducing such new technology into a device that is valuable to them. There are usability factors that hinders the adaptation of continuous authentication [5].

It has privacy challenges that may cause sensitive information to be extracted from the user. Power consumption is also a serious issue since the implementation of such feature regardless whether the computation is done on the cloud or in the device itself will consume more power than the regular use. The reliability of the authentication can also be affected by the user's mood. Therefore, performance is impacted by the user's emotional state.

Rasynayaka et al. [12] surveyed roughly 500 respondents to assess the usability of continuous authentication. Their results show that users are now more accepting towards biometrics based security than 20 years ago. The study also shows that users are welcoming towards multi-level security schemes and find continuous authentication systems to be useful.

5 Attacks

Touch-based continuous authentication is resilient against some known attacks such as replay attacks but is still vulnerable to some like masquerade attacks. The poor matching rate causes malicious users to not be locked out of the system in some instances. Another attack that this CA is vulnerable from is reconstruction attack.

Al-Rubaie et al. [2] investigated the possibility of reconstructing raw data from user's authentication profile. They selected one system to be the compromised system and four others that would be the targets. First, feature vectors must be obtained to reconstruct the raw data. This step was performed in two ways, full profile attack and decision value attack. Full-profile attack involves obtaining the actual feature vectors stored in the system. Decision value attack, on the other hand, begins with a random feature vector. It performs small random changes onto the feature vector and then monitors the decision value if it increases. This is repeated until no more randomization can be performed to improve the decision value. Next, the attack involves reconstructing the raw data using either numerical estimation or randomization algorithm. Numerical estimation is only tailored for one CA system while randomization can work with multiple system. The reconstructed raw data is then injected as sensor input to the other CA systems to simulate the attack. Their experiments showed that the reconstruction attacks had a success range of 73% to 100%. To address this, the authors recommends to avoid exposing user profiles for example with the use of privacy preserving algorithms and to return only

binary classification result.

6 Discussion

Researchers have done multiple studies to further improve touch-based continuous authentication systems. The incremental improvements should be incorporated by future designers to build a CA system that could possibly be of practical use in the real world. It is still hard to say which machine learning algorithm is the way to go or if swipe gesture outperforms keystroke dynamics since there is no standardized dataset or evaluation method that exists. However, it is clear that some practices improve the performance or security of the CA and therefore should be an excellent reference for future implementations. For example, higher accuracy can be obtained by incorporating context into the model and by augmenting the training sample with the use of GANS. The use of privacy preserving algorithms should also be observed to increase the security. Researchers should also start measuring the energy consumption of these systems since it will be an important factor in the adoption of this technology. Overall, touch-based CA is a promising candidate for the implementation of continuous authentication for mobile phones. Consumers are open to the idea of adopting this new technology as long as it adds to the security and does not take away privacy.

7 Conclusion

Multiple studies have pushed touch-based continuous authentication systems in mobile phones closer to real world adoption. The accuracy is not high enough for deployment in actual devices but improvements are constantly being achieved by researchers. There is no common benchmark that is used to compare different implementations. Also, the publicly available datasets are lacking. Privacy is still a problem with most existing implementations and power consumption has not been thoroughly studied yet. Touch-based continuous authentication systems is promising and could be adopted into mobile phones in the future.

References

[1] Mohit Agrawal, Pragyan Mehrotra, Rajesh Kumar, and Rajiv Ratn Shah. Defending touch-based continuous authentication systems from active adversaries using generative adversarial networks. *CoRR*, abs/2106.07867, 2021.

[2] Mohammad Al-Rubaie and J. Morris Chang. Reconstruction attacks against mobile-based continuous authentication systems in the cloud. *IEEE Transactions on Information Forensics and Security*, 11:2648–2663, 12 2016.

[3] Abdulaziz Alzubaidi and Jugal Kalita. Authentication of smartphone users using behavioral biometrics. *CoRR*, abs/1911.04104, 2019.

[4] Ananya and Saurabh Singh. Keystroke dynamics for continuous authentication. In *2018 8th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 205–208, 2018.

[5] Ahmed Fraz Baig and Sigurd Eskeland. Security, privacy, and usability in continuous authentication: A survey. *Sensors*, 21(17), 2021.

[6] Attaullah Buriro, Francesco Ricci, and Bruno Crispo. Swipegan: Swiping data augmentation using generative adversarial networks for smartphone user authentication. pages 85–90. Association for Computing Machinery, Inc, 6 2021.

[7] Rushit Dave, Naeem Seliya, Laura Pryor, Mounika Vanamala, Evelyn R. Sowells Boone, and Jacob mallet. Hold on and swipe: A touch-movement based continuous authentication schema based on machine learning. *CoRR*, abs/2201.08564, 2022.

[8] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *CoRR*, abs/1207.6231, 2012.

[9] Hugo Gascon, Sebastian Uellenbeck, Christopher Wolf, and Konrad Rieck. Continuous authentication on mobile devices by analysis of typing motion behavior. 03 2014.

[10] Yunji Liang, Sagar Samtani, Bin Guo, and Zhiwen Yu. Behavioral biometrics for continuous authentication in the internet-of-things era: An artificial intelligence perspective. *IEEE Internet of Things Journal*, 7:9128–9143, 9 2020.

[11] Ahmed Mahfouz, Tarek M. Mahmoud, and Ahmed Sharaf Eldin. A survey on behavioral biometric authentication on smartphones. *Journal of Information Security and Applications*, 37:28–37, 12 2017.

[12] Sanka Rasnayaka and Terence Sim. Who wants continuous authentication on mobile devices? In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–9, 2018.

[13] Pekka Siirtola, Jukka Komulainen, and Vili Kellokumpu. Effect of context in swipe gesture-based continuous authentication on smartphones. 5 2019.

[14] Ioannis Stylios, Spyros Kokolakis, Olga Thanou, and Sotirios Chatzis. Behavioral biometrics and continuous user authentication on mobile devices: A survey. *Information Fusion*, 66:76–99, 2 2021. Useful introduction for mobile phones security.

[15] Ioannis Stylios, Andreas Skalkos, Spyros Kokolakis, and Maria Karyda. Bioprivacy: Development of a keystroke dynamics continuous authentication system. 10 2021.

[16] Ioannis C. Stylios, Olga Thanou, Iosif Androulidakis, and Elena Zaitseva. A review of continuous authentication using behavioral biometrics. In *Proceedings of the SouthEast European Design Automation, Computer Engineering, Computer Networks and Social Media Conference*, SEEDA-CECNSM '16, page 72–79, New York, NY, USA, 2016. Association for Computing Machinery.

Active Learning for image processing

Elena Serkova
elena.serkova@aalto.fi

Tutor: Ti John

Abstract

Though computer vision is important field of information technologies, it is limited by the costs of labelling data for training machine learning algorithms. However, Active Learning can reduce these costs with high accuracy of predictions. This paper reviews existing Active Learning models used in image processing and compares their efficiency in the experiments and real world. Moreover, the existing problems of Active Learning and ways to solve them were analyzed.

KEYWORDS: Active Learning, image classification, object recognition, computer vision, Convolutional Neural Network, acquisition function, Deep Learning, Deep Active Learning, Bayesian Deep Learning

1 Introduction

Computer vision denotes a subfield of artificial intelligence that teaches computers to extract meaningful information from images or videos by performing object detection, tracking and recognition, image classification and segmentation. Though human sight performs these tasks effectively, it suffers from fatigue and cannot analyze thousands of images in a short time or notice imperceptible differences between objects [13].

Hence, computer vision has an advantage in the actions which require high accuracy, quick reaction and long monotonous work.

Currently, the list of applications that use computer vision includes different areas, such as automotive safety, 3D model building, machine inspection, biometrics and medical imaging. However, machine learning applications for computer vision depend on the quality of data and usually require a large pool of labelled instances [13]. As labelling many images is time-consuming and requires many hours of human labour, the cost of computer vision applications increases significantly.

Active Learning, a subtype of machine learning, can solve this problem. The main difference of this method is that algorithm can request a relatively small number of labels for the data which it chooses itself [14]. The instances to label are chosen in such a way as to maximize the information gained.

This paper considers Active Learning applications in image processing, identifies the problems in this field and the ways to solve them and compares the most efficient approaches.

This paper is organized as follows. Section 2 reviews existing research and revealed problems of machine learning. Section 3 describes variety of Active Learning models and modifications. Finally, section 4 discusses and compares described methods.

2 Active Learning

In other types of machine learning, the algorithm is trained on thousands of labelled objects. However, in some cases, such as speech recognition or drug discovery, labelling a training dataset is computationally expensive [14]. The same problem exists in computer vision because images are usually complicated, noisy and contain many objects. This reduces the economic value of the development of the machine learning model. Active Learning provides a solution for this problem because it can learn from the unlabelled dataset by requesting a relatively small number of labelled instances.

Several approaches are mainly used to train an Active Learning model. The pool-based sampling, the most common sampling method, assumes that a large pool of data can be collected at once, after that the informativeness of all the instances is evaluated. The model trained via stream-based selective sampling analyses data instances sequentially. In

the membership query synthesis, the model can generate a new instance to label instead of selecting from the existing dataset [14].

The Active Learning algorithm should not choose an unlabelled instance to query randomly. Instead of it, the data points must be chosen accordingly to their informativeness. This increases the efficiency of the algorithm. The labels are provided by an oracle (usually a human expert).

This section states problems of Active Learning in the field of image processing and provides an overview of recent research.

2.1 Related Research

In recent decades, the rapid development of information technologies increased significantly the amount of produced data. As a result, Deep Learning models attracted more attention because they can extract informative features from high-dimensional data and predict the target variable by using a massive number of parameters [2]. However, interest in combining Active Learning and Deep Learning models appeared relatively recently. This subsection provides an overview of recent research in Deep Learning models for Active Learning on image data.

Gal et. al. used recent advances in Bayesian Deep Learning to apply Active Learning to high-dimensional image data [4]. The study explored the difficulties of Active Learning applications for high-dimensional data and made a conclusion that Deep Learning models can solve the dimensionality problem. Gal et. al. suggested using Bayesian Convolutional Neural Networks (CNN) in combination with the regularisation technique of dropout to obtain uncertainty data. The uncertainty information was used as an input for 5 different acquisition functions, which determine images for label requesting. One of the functions, random acquisition, was used as a baseline for comparison. BALD, Variation Ratios, and Max Entropy acquisition functions showed better results. Moreover, Bayesian CNN showed better performance than deterministic CNN. This proves the positive effect of uncertainty. As an example of the practical application of Deep Bayesian Active Learning, the Gal et. al. presented a skin cancer detection algorithm. They faced also the problem of an unbalanced dataset and solved it by reducing the class sizes to a fixed number of samples.

Kirsch et. al. developed a Deep Learning algorithm BatchBALD with a batch-based query strategy. Kirsch et. al. also used the advantages of Bayesian Active Learning. They implemented an acquisition function

based on an approximation of the mutual information between a set of data points and model parameters. While this approach causes additional computational costs during acquisition, it reduces the total number of labelled data points and the learning time of the model [8]. The efficiency of the model was tested with a balanced image dataset. Though this method proved its practical potential, it has limitations and can be improved in further works.

Another Deep Learning approach was suggested by Yin et. al. They have also chosen batch mode as it is more practical. The authors defined the problems in existing batch query strategies. The batch of queried data points must be diverse. However, diversity can be hard to achieve because similarity is calculated from the feature vectors rather than being extracted from a classification model. Moreover, a set of points close to the decision boundary can be smaller than needed. The study proposed a deep neural network-based algorithm that measures similarity more precisely and selects a subset of data points that are maximally uncertain, minimally redundant and most diverse from the labelled subset [18]. The accuracy was evaluated on an image dataset.

Though these studies are just several examples of research in the considered field, they represent the main strategies of Deep Active Learning. The basic models, acquisition functions and their performance are considered in the following sections.

2.2 Challenges and problems

An analysis of recent work on Active Learning in image processing revealed several related problems. From the perspective of an machine learning model, an image is a high-dimensional object. Every image has such attributes as height and width in pixels. Moreover, every pixel has characteristics of the colour. For example, if a greyscale picture has m pixels in width and n pixels in height, it can be represented as an n*m matrix, each cell of which equals the greyscale intensity of a pixel [n, m] [15]. Thus, images are the objects of a high-dimensional space and machine learning for computer vision is a high-dimensional problem.

Machine learning models are more efficient with low-dimensional data or high-dimensional data processed with dimensionality reduction techniques [9]. The problems that appear with the dimensionality increase are called the Curse of Dimensionality [10]. One aspect of the Curse of Dimensionality is the risk of overfitting if the number of fea-

tures is higher than the number of observations. Consequently, high-dimensional data processing requires large training sets. High-dimensional is also difficult to visualize and analyze. Another aspect is that the models for high-dimensional data consist of thousands of parameters and finding optimal values for them is more difficult because of slow convergence.

The majority of Active Learning models are trained under the assumption that the dataset is balanced. However, the real-world data is generally imbalanced, and one of the classes can be represented only by several percent of the entire dataset [1]. In this case, the overall accuracy remains high even if the minor class is completely ignored.

Though for such critical fields, such as medicine, labelling must be done by experts, some developers of Active Learning models use crowdsourcing to collect labelled images. Originally, the considered Active Learning methods assume that the experts are trusted and label the instances properly. Because of poor guidance, unreliable annotators or original ambiguity of the data, the collected data may be noisy or irresponsibly labelled [5]. This is another issue developers should be aware of.

Described problems require additional experiments and research in the development of Active Learning models used in image processing.

3 Overview of Active Learning models for image processing

A combination of Deep Learning and Active Learning is beneficial because it would process high-dimensional data with lower labelling costs. Nevertheless, this approach is complicated by several factors. Firstly, Active Learning models require uncertainty values to query new samples, but it is difficult to measure uncertainty in Deep Learning [4]. Secondly, Deep Learning relies on large data sets and Active Learning is trained on a small labelled pool.

Deep Learning models use layers of algorithms to process information. The first layer in a model is called the input layer. Then data is passed through many hidden layers, where the previous layer contributes to the following one. The last layer of the model is called an output layer. Several types of Deep Learning models can be used in image classification [12]. This section describes the most popular model with variations and modifications for Active Learning purposes.

3.1 Convolutional neural networks

A convolutional neural network (CNN) is the most popular Deep Learning algorithm suitable for computer vision problems. It takes an image as an input, processes pixel data using convolution function in hidden layers and performs computer vision tasks, such as classification, recognition and segmentation [16]. The CNNs need large amounts of data for training, but many algorithms do not perform the uncertainty quantification. Uncertainty assessment is crucial in the safety-critical fields, where even rare incorrect predictions of an over-confident neural network can result in severe damage. Moreover, uncertainty quantification is pivotal for optimization and decision-making in Active Learning. The problem of CNNs is that in a new for them situation they can become overconfident in their output probability distribution [4].

Bayesian CNN represents each output prediction as a probability distribution, but not as a single point. The weights of the CNN are also a distribution, for instance, a Gaussian distribution with such parameters as mean and variance [4]. Because of this feature, Bayesian CNNs can be used for Active Learning.

However, it can be difficult, for example, to derive gradients for the mean and variance of each weight in the backpropagation step [4]. The dropout regularization can be used to make this task simpler. During dropout regularization, the outputs of some layers are randomly ignored. It makes the training process noisy, and nodes must probabilistically take on more or less responsibility for the inputs. This method is usually used only for the training stage, but, in the case of Bayesian CNNs, keeping dropout in the inference stage provides the data for uncertainty estimation. If dropout is used, CNN returns a slightly different result for each inference. The variance of the predicted results is a way to quantify the uncertainty of the model.

Another property of Bayesian methods in machine learning is marginalization instead of using a single setting of parameters [4]. In machine learning and probability theory, marginalization is a method of summing a probability of a variable that has a joint probability distribution with other variables. It allows determining the marginal contribution of the variable by calculating the sum over the possible values of other variables.

The possibility to extract uncertainty values from the model made

Bayesian Convolutional Neural Networks popular in recent years. The results of different experiments with image data show that this method in combination with Active Learning reduces computational costs with better accuracy. Future modifications of the Bayesian approach can be done via acquisition functions, which are analyzed in the next subsection.

3.2 Query strategies

Choice of samples for labelling is one of the most important tasks of Active Learning models. Intuitively, it is more beneficial to query the most informative data points which would increase the model accuracy. This section describes different acquisition functions that can be used in deep Active Learning models.

The basic approach, which can also be used as a baseline for evaluating other methods, is known as random acquisition. This acquisition function chooses a data point uniformly at random from the unlabelled set [14].

Another popular acquisition function is the maximum predictive entropy search. The predecessor of this function, entropy search, used an approximation of the posterior entropy of the model to select data examples for labelling [7]. The selected set of data points was supposed to minimize the uncertainty about the parameters of the model. However, the computational cost of entropy search was high. To address this problem, the entropy search was modified using the symmetry of mutual information in the predictive entropy search.

Similar to predictive entropy search, acquisition function based on variation ratios measures lack of confidence. This function tries to maximize variation ratios, a measure of statistical dispersion in nominal distributions defined as the proportion of cases that are not in the mode category [3].

The acquisition function based on the mean standard deviation is not so widely used and became more popular in recent years. The main idea of this method is to maximize the dispersion of a dataset relative to the mean value averaged over all classes that a data point can take.

Bayesian Active Learning by Disagreement (BALD) became a basis for another acquisition function, which evaluates the mutual information between the predictions and parameters of the model. This approach calculates the entropy of the model prediction and the expected entropy of the prediction over the posterior of the parameters. The first element

must be high, and the second one is expected to be low. This indicates that the prediction of the model is uncertain, but in general, the model is certain for each set of settings from the posterior [4]. In other words, the model is uncertain on selected data points on average, but some variants of the model produce output for these points with high certainty, and these predictions differ from each other. Detecting and labelling these data points helps to train the model faster.

The next considered approach is based on contrastive examples acquisition and uses both uncertainty and diversity to acquire data [11]. Contrastive examples are represented by data points that are close to each other in the feature space but classified differently by the model. The points for labelling are selected based on two rules. Firstly, the distance between data points in their feature space must be small. Secondly, predictive probability distributions for these data points must maximally diverge. Chosen samples are good candidates for labelling.

A separate CNN can be used for the extraction of the most informative samples [17]. In the process of training, CNNs learns representation space, in which similar examples have smaller distance. The goal is to choose data points that can reduce the uncertainty of the model after addition to the train set. The efficiency of this approach depends on chosen distance functions. Euclidean and Chebyshev distance are examples of these functions.

Though considered acquisition methods are based on different uncertainty or diversity metrics, they all have a common goal, to select the most informative data points for labelling. The efficient acquisition function allows increasing the accuracy of the faster with less labelled points. Described methods are compared in the next section.

4 Discussion

Though Active Learning proved its efficiency and potential for real-life tasks, there are still many problems to solve. Choice of the model settings is the first and most important one.

One of the previously described studies evaluated five acquisition functions on image datasets using Keras MNIST CNN implementation [4]. This study assessed a number of labelled data points required to achieve 5 and 10 percent of erroneous predictions. BALD, Variation Ratios, and Max Entropy achieved target test error with much fewer labelled

data points than Mean STD and Random acquisition functions. BALD, Variation Ratios, and Max Entropy were also compared as an acquisition function of Bayesian and deterministic CNN. Bayesian models learned from fewer labelled data points and showed higher accuracy. It proves that uncertainty, propagated throughout the Bayesian models, is beneficial for deep Active Learning models.

Another research implemented the BatchBALD model and compared the results with the basic BALD acquisition [8]. Acquisitions of a set of points instead of a single data point increased diversity and showed increased performance over BALD and other functions.

Previously considered methods, including the Deep Bayesian Active Learning approach, require a balanced training dataset and without modification can suffer from false accuracy. For example, in the case of a melanoma diagnosis, the accuracy of a Deep Bayesian Active Learning model can be improved by marking all the points as benign, because malignant is a minor class. However, different types of modifications were proposed to tackle this problem.

The majority of data imbalance solutions are applied in the preprocessing or training stage. During the preprocessing, two techniques are commonly used: generating instances for the minority class or deleting data points from the majority class [1]. The problem of imbalanced data can be solved by the modification of the acquisition function and implementation of a deep model, pretrained on the source data without an initial labelled subset. The pretrained model stores classes and their probability for each instance. The instance is queried only if the class with the highest probability has not already been encountered too often. Moreover, the balancing step was added to reduce the propagation of unlebelled pool imbalance to the labelled set. This step prioritizes the classes which are underrepresented in the labelled pool.

Although the first approach is easier to implement, the generated dataset does not represent clearly the original one and can contain misleading artificially generated datapoints. At the same time, modification of acquisition functions can be used in many types of models without changing the original dataset. The accuracy of this approach demonstrates that it can be developed further and potentially used in many application fields.

If the labelling or data collection is built on a crowd-sourcing basis, these models require additional modifications to reduce the noise and

eliminate labels from not trusted oracles. One way to select only reliable annotators is the majority voting-based method, which determines a confidence interval for the oracle reliability and excludes annotators with reliability below the threshold. Another solution is to use an Active Learning algorithm not only to select data points to label but also to select noisy labelled instances to label again. Gang Hua et. al. proposed the Kernel Machine Ensemble model which efficiently solves some of the problems of crowd-sourced data via collaborative learning [6]. Each annotator has his own Active Learning process with provided guidance. The learning processes in total are not independent, as they have a pool of shared data points. The framework uses a unified discriminative approach to model the consistency of labels. As a result, the Active Learning model becomes more robust to the noise, labellers' performance can be evaluated.

5 Conclusion

This paper considered the existing problems of Active Learning applied to image processing, and described the most efficient approaches.

Active Learning demonstrated efficiency in image processing because it reduces costs by using smaller labelled dataset. Because of the high dimensionality of visual data, Deep Learning models are more beneficial than other machine learning models. However, adapting Deep Learning models to Active Learning tasks is complicated by factors, such as lack of uncertainty measurements, imbalanced data, and choice of acquisition approach.

The Deep Bayesian Active Learning model provides required uncertainty and achieves high accuracy with different acquisition functions, such as BALD, Variation Ratios, and Max Entropy. BatchBALD acquisition function, which adapts BALD to batch queries, improves Active Learning further. Nevertheless, these models require modifications to be used in practice because real-world data is imbalanced and noisy.

Recent advances in Active Learning show that these technologies can be used in many critical areas where the costs of labeling image datasets outweigh the benefits of digitalization.

References

[1] Javad Zolfaghari Bengar, Joost van de Weijer, Laura Lopez-Fuentes, and Bogdan C. Raducanu. Class-balanced active learning for image classification. *CoRR*, abs/2110.04543, 2021.

[2] Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021.

[3] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1050–1059. JMLR.org, 2016.

[4] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1183–1192, August 2017.

[5] Gang Hua, Chengjiang Long, Ming Yang, and Yan Gao. Collaborative active learning of a kernel machine ensemble for recognition. In *2013 IEEE International Conference on Computer Vision*, pages 1209–1216, 2013.

[6] Gang Hua, Chengjiang Long, Ming Yang, and Yan Gao. Collaborative active learning of a kernel machine ensemble for recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[7] Michael Kampffmeyer, Arnt-Børre Salberg, and Robert Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. 07 2016.

[8] Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *CoRR*, abs/1906.08158, 2019.

[9] Daphne Koller and Simon Tong. Active learning: theory and applications. 2001.

[10] Nikolaos Kouiroukidis and Georgios Evangelidis. The effects of dimensionality curse in high dimensional knn search. In *2011 15th Panhellenic Conference on Informatics*, pages 41–45, 2011.

[11] Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. Active learning by acquiring contrastive examples. *CoRR*, abs/2109.03764, 2021.

[12] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *CoRR*, abs/2009.00236, 2020.

[13] Sourav Dey Roy and Mrinal Kanti Bhowmik. A comprehensive survey on computer vision based approaches for moving object detection. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 1531–1534, 2020.

[14] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[15] Nakul Shahdadpuri. Real image of computer vision application and its impact: Future and challenges. 12 2020.

[16] Neha Sharma, Vibhor Jain, and Anju Mishra. An analysis of convolutional neural networks for image classification. *Procedia Computer Science*, 132:377–384, 2018. International Conference on Computational Intelligence and Data Science.

[17] Asim Smailagic, Pedro Costa, Hae Young Noh, Devesh Walawalkar, Kartik Khandelwal, Adrian Galdran, Mostafa Mirshekari, Jonathon Fagert, Susu Xu, Pei Zhang, and Aurélio Campilho. Medal: Accurate and robust deep active learning for medical image analysis. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 481–488, 2018.

[18] Changchang Yin, Buyue Qian, Shilei Cao, Xiaoyu Li, Jishang Wei, Qinghua Zheng, and Ian Davidson. Deep similarity-based batch mode active learning with exploration-exploitation. pages 575–584, 11 2017.

Image generation with generative models

Guangkai Jiang
guangkai.jiang@aalto.fi

Tutor: Yu Tian

Abstract

In recent years, the need for image generation with AI is growing and many generative models have been invented and received much attention, such as PixelRNN/CNN, VAEs and GANs. In this report, I aim to review the state-of-the-art solutions, understand their core concepts and differences, analyse their advantages and flaws, and compare them, in order to gain some perspective on this promising field.

KEYWORDS: Image generation, Generative model, Pixel recurrent neural networks, Pixel convolutional neural network, Variational autoencoder, Generative adversarial networks

1 Introduction

In recent years, image generation with AI has become an essential part of entertainment industry, especially for the creation of films and games. In earlier days, the computer generated images may seems unrealistic however the gap between artificial and authentic images has dramatically reduced over the years with the advancement of image generation approach. The machine learning approach to generate images uses generative models. Generative models is one type of models in statistical

classification as opposed to discriminative models [10]. Unlike discriminaive models that labels different classes of data, generative models generates new data instances similar to given data instead. Mathematically speaking, with a set of data X and a set of labels Y, generative models focus on the joint probability $p(X, Y)$ while discriminative models focus on conditional probability $p(Y|X)$. The current state-of-art approach for image generation with generative models is generative adversarial networks (GANs)[3]. Since the idea of GANs was invented in 2014 [5],it has become one of the most effective models among generative models and attracted many attentions. GANs is prized as "the most interesting idea in the last 10 years in machine learning" by Yann LeCun as cited in [2]. GANs and their variances have out-performed most of other generative methods, however, there are still some competing alternatives, such as variational autoencoders (VAEs) and autoregressive models.

This paper reviews existing literature on generative models, such as autoregressive models, variational autoencoders and generative adversarial networks, to understand their similarities and differences with the focus on applications related to image generation.

The remaining part of the report is stuctured as follows:

Section 2 explores the application of generatives models with focusing on image.

Section 3 reviews the core idea and difference of generative models.

Section 4 reviews autoregressive models.

Section 5 reviews variational autoencoders.

Section 6 reviews generative adversarial networks.

Section 7 compares the models mentioned above and makes conclusion.

2 Generative model and its applications

"A generative model describes how a dataset is generated, in terms of a probabilistic model. By sampling from this model, we are able to generate new data"[4]. With its unique feature, generative models can generate instances of data that are highly similar in certain ways to original data. This enables generating unlimited data with a relatively small amount of original data, which is similar to the pseudo random generator generates unlimited amount of numbers given only seed data. Generative models have many applications related to images, such as super-resolution, compression, morphing, and text-to-image [9].

3 Core idea and difference of generative models

The core task of any generative models is to estimate the probability distribution $P(X)$ and then sample X from the learned function $F(x)$. Different generative models differ in the way they approximate the underlying true probability distribution.

Due to the curse of dimensionality, it is unfeasible to compute the underlying true probability distribution. In order to limit the hypothesis space, human knowledge is introduced as prior (existing knowledge or belief), which lead to different Bayesian prior probability. Different Bayesian prior probabilities and approaches towards the probability distribution lead to various models.

Existing solutions use either explicit density or implicit density to estimate probability distribution. With the explicit density approach, there are models, like PixelRNN/CNN, which use tractable density and models, like VAEs, which use approximated density. As for the implicit density approach, GANs are the most popular models.

4 Autoregressive Models

Autoregressive models include Pixel Recurrent Neural Networks (PixelRNN) and Pixel Convolutional Neural Networks (PixelCNN). PixelRNN/CNN aim to explicitly estimate a distribution that can be used to tractably compute the likelihood of an image and to generate new ones [12].

4.1 PixelRNN

PixelRNN models try to find a tractable probability distribution that can be used to generate new images. The probability of a image x with a $n \times n$ pixels can be written as the product of conditional distribution over all pixels:

$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1})$$

, where x_i is the pixels and $p(x_i)$ is the probability of the i-th pixel x_i given all pixels $x_1, ..., x_{i-1}$.

PixelRNN models are trained with recurrent neural networks (RNN), iterating pixels one by one. The training process is sequential and cannot be parallelized due to the nature of the recurrent neural networks, thus the training requires a relatively long time. However, as the pixels are calculated one by one, it offers the unique usage of image completion. In order to improve the training time, RNNs can be replaced with a convolutional neural network (CNN), making the training process parallel, with only little sacrifice on the quality of generated image.

4.2 PixelCNN

PixelCNN is based on PixelRNN, but, replaces the RNN with CNN to speed up the training. For each pixel, a convolutional kernel is used to calculate the probability distribution. As each pixel should not have knowledge about the pixels after it, a mask kernel is applied to hide the pixels after it.

Note that only the training process is parallelized, the image generation is still pixel by pixel, similar to PixelRNN. Compared to PixelRNN whose pixels see all pixels before them, the pixels in PixelCNN only see the pixels close to them. The quality of the generated image is not as high as the ones generated by PixelRNN, as all information beyond the convolutional kernel is missing.

5 Variational autoencoders

VAEs can work with intractable posterior distribution by reparameterizing the variational lower bound to yield a differentiable unbiased estimator of the lower bound, evidence lower bound (ELBO) [8].

A VAE consists of two multi-layer perceptrons, the encoder and the de-

coder. The original image data x is encoded by the encoder with Gaussian Distribution $p_\phi(z|x)$ into representation z, and then z is reconstructed by decoder $p_\phi(x|z)$ back to $\tilde{x}$. The loss function is consists of the reconstruction loss $-E_{z\sim q_\phi(z|x_i)}[logp_\phi(x_i|z)]$ and Kullback–Leibler divergence (KL divergence) between encoder's distribution $q_\phi(z|x)$ and $p(z)$. Because there is no analytical solution to optimize the likelihood, VAEs resort to optimize the ELBO of log-likelihood of the observed data.

One major challenge face by VAEs is the heavy computation power required for high quality results and the training difficulty due to posterior collapse, that is, when the input signal to posterior is either too weak or too strong, rendering the learned latent space meaningless.

6 Generative adversarial networks

The prior belief of GANs is that given any vector sampled from Gussian Distribution, it can be transformed to satisfy the underlying distribution of the input data. GANs abandon the idea of finding explicit probability distribution. Instead they take the approach of two player minimax games[5]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z_z(z)[log(1-D(G(z)))]}$$

GANs consist of two multilayer perceptrons, the discriminators and the generators. Given original image data, the generators generate fake images via learning the distribution of the original image data, then the original data and the fake data are both fed to the discriminators. The discriminators aim to distinguish the original data from the fake data while the generators aim to make the fake data indistinguishable from the original.

The advantage of the minimax game is that it can be expressed with Jensen-Shannon divergence (JS divergence):

$$C(G) = -log(4) + 2 \cdot JSD(p_{data}||p_g)$$

In contrast to KL divergence, JS divergence is symmetric [6], which makes the minimax game more efficient to compute. Unlike other unsupervised learning models, GANs use supervised loss function with self-labeled (original image or fake image) data, which speeds up the training process.

However, even as the state of art solution, GANs are still facing issues, especially the difficulty in training the models. During train iterations,

the generators and discriminators need balanced mutual progress, if either side is too strong, it would annihilate the other side and break the training process[5]. There is also the issue of model collapse, that is, when either generators or discriminators learned the weakness of the adversary and got stuck in the local minimum, causing the model unable to improve anymore over the iterations. Many variants of modified GANs are invented aiming to conquer these problems.

6.1 Wasserstein GAN

Wasserstein GAN analysed the flaw in using JS divergence and proposed the idea of replacing it with Wasserstein metric.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||]$$

This new model improves the stability of learning, gets rid of problems like mode collapse, and provides meaningful learning curves useful for debugging and hyperparameter searches [1].

6.2 Progressive GANs

Progressive GANs introduce a new training methodology for GANs, by growing both the generator and discriminator progressively. In a progressive GAN, the generator's first layers start from low resolution, and subsequent layers model increasingly fine details as training progresses. This technique speeds up the training and greatly stabilizes it, making it faster then most comparable non-progressive GANs, and produces images of higher quality [7].

6.3 DCGANs

The original GANs [5] are new and popular unsupervised learning algorithms whose parameters of the networks are massive and difficulty to train because they implement the generator and discriminator mainly based on the fully connected layers to ,and the dimensions of images is so high. Whereas CNNs is easier to train and have seen huge adoptions and success in supervised learning computer vision applications. Deep Convolutional Generative Adversarial Networks (DCGANs) combine GANs with CNNs to help bridge the gap between the success of CNNs for supervised learning and unsupervised learning. DCGANs make three major modifications to the original GANs: replace pooling layers with fractionally-

strided convolutions, enable the network to learn the its own downsampling; the trend towards eliminating fully connected lays on top of convolutional features, as global average pooling increases model stability but reduce convergence speed, a middle ground was chosen; use Batch Normalization which stabilizes learning by normalizing the input to each unit to have zero mean and unit variance, helps gradient flow in deeper models and speed up the convergence [11].

7 Conclusion

Autoregressive models calculate tractable log likelihood and have a simple and stable training process , but they take a long time to generate images and are too slow to be used in quick image generation tasks.

VAEs approximate the likelihood by optimizing ELBO, it is more efficient than autoregressive models, but the generated samples tend to be blurry because of the loss of information during encoding and reconstruction.

GANs can generate sharp images quickly, but as GANs train directly with implicit probability density, they work in the way of black box. although the training speed is fast, it is difficult to tune the training dynamics to train them successfully.

All three approaches solve the problem of image generation starting from Bayesian probability distribution, they share the same core concept and goal but differ in mathematical techniques and technical implementations. Their difference has granted them unique strength in different scenarios, and they all inspired and paved the way for latecomers.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[2] Jamie Beckett. What's a generative adversarial network? inventor explains. *The NVIDIA Blog*, May 2017.

[3] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models, 2021.

[4] D. Foster. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*. O'Reilly Media, 2019.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[6] Yongchao Huang. Kullback-Leibler (KL) divergence and Jensen-Shannon divergence - a revisit to KLD and JSD, July 2020.

[7] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017.

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[9] Shakir Mohamed and Danilo Rezende. Tutorial on deep generative models. *UAI 2017 Australia*, 2017.

[10] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

[11] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[12] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks, 2016.

Energy-efficient asymmetric multi-core scheduling for virtual machines

Tommi Räsänen
tommi.rasanen@aalto.fi

Tutor: Jaakko Harjuhahto

Abstract

Virtual machines are increasingly used to perform computing tasks, mainly by provisioning them as computing resources from large cloud data centers. These data centers consume a large amount of energy, thus it would be desirable to optimize the energy-efficiency of the underlying hardware. Energy-efficiency has been previously optimized on mobile hardware by using asymmetric multi-core processors (AMPs), which have cores with different performance characteristics. Thus, using AMPs could potentially also improve the energy-efficiency of virtual machines.

The key challenge is scheduling; deciding which virtual machine tasks should be executed on which physical processor core. This paper reviews four different scheduling methods specifically designed for scheduling of virtual machines with AMPs.

It was concluded that current scheduling methods focused on mapping physical cores to virtual cores mainly based on virtualized core frequency. Thus, it would be possible to improve energy-efficiency further with a scheduling method that considers the performance requirements of the actual tasks being executed by virtual machines.

KEYWORDS: asymmetric multi-core, heterogenous computing, scheduling, virtual machines, virtualization, cloud, data center, energy-efficiency

1 Introduction

Computing tasks are increasingly performed by provisioning computing resources from a cloud provider. Cloud providers operate large data centers, where physical machines are pooled and resources can be provisioned by third parties according to the exact computing power needed. This pooling means that the resources are virtualized and not physical machines.

These data centers consume a large amount of energy. In 2005, the energy consumption of data centers in the US accounted for 1% of total energy consumption and is estimated to increase by around 15% per year [1]. Therefore, it is important to implement methods to improve the energy efficiency of data centers.

One possibility for improving energy efficiency is by utilizing asymmetric multi-core processors (AMPs). These processors have heterogenous cores; some cores are less powerful and use less energy while other cores are more powerful but consume more energy [2]. This application of heterogenous cores is already commercially used in mobile phones [2].

However, a key challenge with the use of heterogenous cores is scheduling. Scheduling is the process whereby a scheduler, often the operating system, decides which tasks are executed by each processor core. In order to improve energy efficiency without reducing throughput, the scheduler needs to identify which tasks require more computing power and must be executed with the more powerful core and which tasks are less demanding and may be executed with the less powerful core, improving energy efficiency [3].

The scheduling of virtual machines is more complicated than that of physical machines. This is because virtualization adds two more layers: the hypervisor and the guest operating system scheduler [4].

The paper aims to review different methods presented in literature for scheduling tasks within virtual machines that are physically running on hardware powered by AMPs.

This paper is structured as follows. Section 2 covers some key concepts that must be understood when reading the article. Next, Section 3 covers AMP scheduling methods that have been proposed in literature. Then, Section 4 discusses these proposed methods. Finally, Section 5 offers a

conclusion.

2 Concepts

This section explains some of the core concepts surrounding the scheduling of asymmetric multi-core processors on virtual machines.

2.1 Scheduling

Scheduling is the process whereby a computer decides which tasks are executed on processors [5]. Proper scheduling ensures that the system remains responsive and achieves high efficiency.

In a traditional single processor setting with one core, only one task is executed at a time, and the remaining tasks are held in a task queue. However, with multiprocessor systems, it becomes possible to execute multiple tasks in parallel. Usually, the processors share a common task pool, or there may be different task pools according to different task priorities [5].

Scheduling may be pre-emptive or cooperative [5]. Cooperative scheduling means that processes have to explicitly yield the processor for use by other processes, while pre-emptive scheduling allows the scheduler to pause execution at any point. Most modern operating systems employ pre-emptive scheduling.

A context switch is the event where the scheduler pauses the execution of the current process and allows another process to execute [5]. These context switches come with a cost; it is necessary to store the processor state of the current process and then load the processor state of the next process before execution may resume. Thus, an efficient scheduler must balance between the cost of making context switches and keeping all running tasks responsive.

2.2 Virtual machines

Virtual machines allow multiple separate operating systems to be executed on a single machine. The virtualized operating systems are not necessarily aware that they are running on a virtualized machine [6].

The core of virtual machines is the hypervisor, also called a virtual machine monitor. This hypervisor is responsible for assigning physical hardware resources to the virtualized machines [6]. For example, when the

guest OS tries to perform an I/O operation, the hypervisor intercepts this call and then translates it into a call that the host OS can process. Once the host system has processed the request, the hypervisor transforms the response to conform with the expectations of the original guest OS. Thus the whole process appears to the guest operating system as it had been communicating with actual hardware [7]. Figure 1 shows the relationship between the guest OS, the hypervisor and the host OS.

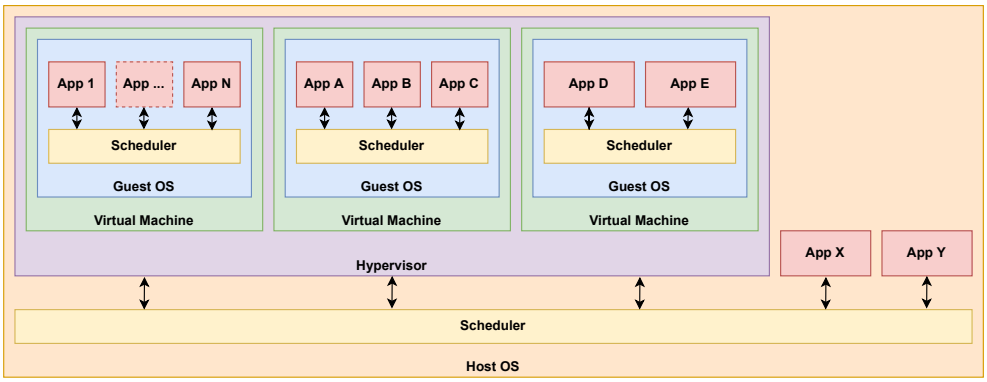


Figure 1. Overview of the relationship between the host OS scheduler, the hypervisor and the guest OS schedulers. Adapted from [8] and [7].

Virtual machines have at least three properties that make them attractive for cloud providers: security, separation of hardware and software management [6].

In the context of this review, virtual cores refer to virtualized processor cores that guest OSes posses. Physical cores refer to processor cores present on the physical hardware where the host OS resides.

2.3 Asymmetric multi-core processor

An asymmetric multi-core processor (AMP) is a processor that has cores with differing types. AMPs may have cores that differ in terms of microarchitecture or instruction set architecture [2]. Microarchitecture denotes the way in which the functionality of the processor is implemented [9]. Instruction set architecture denotes the model for programming on the processor. This includes supported opcodes, processor registers and data types [9]. Examples include Intel x86, ARM A64 and PowerPC.

There are different kinds of asymmetric multi-core processors. In the simplest case, the cores can be strictly ordered in terms of performance. However, in some cases cores may be specialized and perform better at specific tasks, such as video encoding. These types of AMPs are called monotonic and nonmonotonic, respectively [2].

Method	Requires training	Fairness guarantee	Energy efficiency	Performance guarantee	Implementable
Kazempour et al.	No	Yes	No	No	Yes
Takouna et al.	No	No	Yes	Yes	No
Wang et al.	Yes	No	Yes	No	Yes
Lin et al.	No	No	Yes	Yes	Yes

Table 1. Summary of the different scheduling methods presented in Section 3.

In addition, some processors have configurable performance characteristics. In these configurable processors, distributed processor resources can be dynamically assigned to specific cores. These AMPs are referred to as reconfigurable AMPs, while non-reconfigurable AMPs are called static AMPs [2]. Reconfigurable processors reduce the overhead associated with context switches because the processor may be reconfigured instead of performing a context switch. However, reconfiguring the processor incurs overhead [2].

AMPs have been used in mobile phones phones for around a decade, with the first mobile phone being released in 2013 [10]. Recently, AMPs have also appeared in computers, with Apple releasing its M1 processor and Intel releasing its Alder Lake processor [11].

The scheduling methods reviewed in this article apply to single microarchitecture, monotonic and non-reconfigurable cores. The more powerful cores are referred to as fast cores, while the less powerful cores are referred to as slow cores.

3 AMP scheduling of virtual machines

The following section presents four methods that have been proposed in literature for scheduling virtual machines on asymmetric multi-core processors. The covered scheduling methods differ in their goals regarding efficiency, fairness and performance. A summary of the methods and their goals are listed in Table 1.

First, this paper will review a basic scheduling method that ensures fair sharing of the physical fast cores, then it will review methods specifically aiming to improve energy efficiency.

3.1 Fair sharing of fast cores

Kazempour et al. proposed a scheduler for ensuring equal sharing of physical CPU cores across virtual machines [3]. Their main goals were to ensure simplicity and to minimize thread migration overhead. The scheduler was implemented by modifying the open source hypervisor Xen. Xen utilizes the concept of credits for scheduling virtual CPUs. In this case, the credit concept was amended with separate fast credits and slow credits for utilizing the fast and slow cores, respectively. Fast credits are distributed using a queue. All virtual CPUs are placed into this queue and during events called accounting periods, both slow and fast credits are distributed among virtual cores. Fast credits are assigned to the top n virtual CPUs, where n is the number of physical fast cores. After distributing the credits, the n last entries in the queue will be moved to the top, thus each virtual CPU eventually gains fast credits. If there are fewer virtual fast cores than physical fast cores, virtual slow cores will be tasked to physical fast cores. Virtual cores that do not receive fast credits during an accounting period receive slow credits. Migration overheads were reduced by using 30 ms as the accounting period, which was deemed a long enough period to prevent excess context switching. They concluded that the migration overhead was only a maximum of 4%.

Experiments conducted by the team showed that the scheduler offered a performance improvement of up to 36% on some tasks, while suffering minor performance losses on other tasks [3].

3.2 Task appropriate core scheduling

Takouna et al. present a scheduler that aims to execute CPU intensive tasks on fast physical cores and I/O bound tasks on slow physical cores [12]. The scheduler has separate fast and slow CPU queues. The fast queue is used for CPU intensive virtual CPUs while I/O intensive virtual CPUs are placed in the slow queue. The article does not specify how tasks are classified as CPU intensive or I/O intensive, but simply states that virtual machine characteristics should be analyzed. Their evaluation was performed by pinning virtual cores to specific physical cores.

The scheduler was evaluated by pinning the slow physical cores to virtual cores performing I/O intensive tasks while the fast physical cores were pinned to virtual cores performing CPU intensive tasks [12]. The researchers observed performance improvements of up to 70% and mea-

sured power savings of up to 25%.

3.3 Energy-efficient scheduling by mapping virtual cores to power appropriate cores

Wang et al. propose a method that focuses solely on improving energy-efficiency [13]. Their method is based on optimizing the performance per watt value, i.e., whether the performance per watt value is higher when executing a particular virtual CPU on a fast physical core versus executing it on a physical slow core. There is a difference in energy efficiency depending on the characteristics of the virtualized versus the physical core.

The efficiency is estimated, since obtaining actual energy efficiency data for each process would be infeasible [13]. This estimation is performed by first executing a series of performance benchmarks at each virtualized CPU frequency. During the execution of the benchmarks, 22 different performance metrics are collected and power consumption is measured using a power meter. Furthermore, performance data is also collected when executing the actual workload. However, the researchers note that dynamic updating is difficult because CPUs may not have power meters. The collected data is then used to estimate the energy efficiency in the form of an energy-efficiency coefficient by using linear regression. These estimated energy-efficiency coefficients are then used when assigning virtual CPU cores to physical cores. The energy-efficiency coefficient uses a model whereby a higher value indicates that it is more energy-efficient to execute the process on a fast core. Thus, at each scheduling interval, the estimated energy-efficiency coefficients are divided into two sets by first sorting them according to the coefficient and then splitting the set at the n-th element where n is the number of physical fast cores. Then when scheduling is performed, slow physical cores prefer virtual cores from the set of smaller values while fast physical cores prefer virtual cores from the set of larger values. Scheduling is performed at the same interval as the default Xen scheduler, since performing context switches more often would result in additional overhead.

In their evaluation, they executed various workloads common for cloud-hosted virtual machines, including running Java and PHP web servers [13]. The resulting efficiency was on average about 10% better for the proposed scheduling policy. However, there were also some experiments (10 out of 64) where the observed energy efficiency was worse than with

the default Xen credit based scheduler.

3.4 Linear programming scheduler

Lin et al. propose a scheduling method that assigns virtual cores to physical cores by solving a linear programming optimization problem [14]. The optimization problem minimizes power consumption, but also includes a performance constraint. Power consumption is modeled based on physical core type (fast or slow), core frequency and core load (the ratio of a time slot that a physical core is being used). Performance is modeled as the sum of the actual assigned physical core frequencies times their duration fraction, divided by desired virtual core frequency. The main optimization target is the minimization of the power consumption. The optimization has additional constraints for performance, to ensure that virtual cores are executed on at most one physical core at a time, and finally that physical cores do not exceed their available computing power. The researchers note that currently, this linear programming problem can be solved quickly, since the number of virtual and physical cores are small constants.

The researchers evaluated their scheduler against Xen's credit based scheduler by executing two tests: a small workload and a heavy workload [14]. Executing the smaller workload showed an energy-efficiency improvement of 57.2%, while the larger workload showed an improvement of 4.4%. The improvement in the larger workload was due to the large workload periodically being executed on the slow physical core when using the original scheduler.

4 Discussion

When considering the scheduling policy by Kazempour et al. that ensures fair sharing of fast physical cores, the method does not achieve optimal power efficiency since all fast physical cores are always assigned to virtual cores regardless of their virtualized frequencies. The researchers also acknowledged this and suggested leaving fast physical cores unused if fewer fast virtual cores had been provisioned [3]. However, efficiency is still not optimal since the scheduler only considers the frequency of the virtualized core and does not consider the actual tasks being executed. Thus, it is possible that a CPU intensive task is given to the slow physical

core while the fast physical core is executing an I/O intensive task. The researchers deemed scheduling of the individual tasks to be the responsibility of an asymmetric-aware operating system, arguing that adding this functionality to the hypervisor would result in duplicating existing work [3]. Their proposed scheduler generally outperforms AMP unaware schedulers by ensuring that the fast core is utilized as much as possible and that it is fairly shared. Therefore it is fair to consider it better than an AMP unaware scheduler, however this method does not provide an optimal scheduling in terms of performance or energy efficiency.

Task appropriate core scheduling is proposed by Takouna et al. and their evaluation shows huge improvements in both energy-efficiency and performance. However, their proposed method is more theoretical than practical, since they do not offer a method for distinguishing tasks or virtual machines that should be executed on specific physical cores. Instead it is perhaps most useful by showing the potential improvements that can be achieved with an optimal AMP scheduling policy.

Mapping virtual cores to power appropriate physical cores as suggested by Wang et al. provides a noticeable improvement in the energy-efficiency of virtual machines. However, the method requires a two-step training process, offline and online, which would make it difficult to use for very heterogenous workloads that occur in practice. It is also worth pointing out that its effect on performance is not considered in detail.

Finally, the linear programming scheduler shows great improvement in terms of energy-efficiency, particularly when executing small tasks. In addition, the method is simpler to implement than the previous method, since it does not require extensive collection of runtime performance parameters.

Since the evaluation scenarios of each method were very different, it is difficult to compare the results directly. However, the linear programming method seems to be the most promising of the methods, because it considers both energy-efficiency and performance, it aims to schedule slow virtual cores on slow physical cores and it is relatively simple to implement because it does not need extensive runtime data.

It is worth noting that none of the covered methods try to consider the actual task being executed on the virtual machine, instead they all base the target performance on the frequency of the virtualized core. If it were possible to recognize I/O bound tasks being executed on fast virtual cores and execute them on a slow physical core, energy-efficiency could be in-

creased further. However, this has previously been considered impractical because cloud providers have no visibility into the applications that are being executed on provisioned instances [13]. But recently, CPUs, such as Intel's Alder Lake, collect performance metrics as they are executing tasks and use this for future scheduling. Perhaps it would be possible to extend this technology to also cover virtual machines.

5 Conclusion

There have been a few different methods proposed specifically for AMP scheduling in virtual machines. They offer an improvement when compared to running an AMP on an AMP unaware scheduler. However, it would theoretically be possible to achieve even greater power savings by creating a scheduling method that considers the tasks being executed in each virtual machine. Thus, there is still room for improvement in this area. Future research could be conducted to implement a scheduler that exploits the performance metrics gathered by modern AMP processors to improve AMP scheduling in virtual machines in terms of energy-efficiency.

References

[1] Vimal Mathew, Ramesh K. Sitaraman, and Prashant Shenoy. Energy-aware load balancing in content delivery networks. In *2012 Proceedings IEEE INFOCOM*, pages 954–962, 2012.

[2] Sparsh Mittal. A survey of techniques for architecting and managing asymmetric multicore processors. *ACM Comput. Surv.*, 48(3), February 2016.

[3] Vahid Kazempour, Ali Kamali, and Alexandra Fedorova. AASH: An asymmetry-aware scheduler for hypervisors. *SIGPLAN Not.*, 45(7):85–96, March 2010.

[4] Tianxiang Miao and Haibo Chen. Flexcore: Dynamic virtual machine scheduling using VCPU ballooning. *Tsinghua Science and Technology*, 20(1):7–16, 2015.

[5] William Stallings. *Operating Systems - Internals and Design Principles (7th ed.)*. Pitman, 2011.

[6] John L. Hennessy and David A. Patterson. *Computer Architecture, Sixth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 6th edition, 2017.

[7] Matthew Portnoy. *Virtualization Essentials*. SYBEX Inc., USA, 2nd edition, 2016.

[8] Ameer Alaasam, Gleb Radchenko, and Andrei Tchernykh. Comparative analysis of virtualization methods in big data processing. *Supercomputing Frontiers and Innovations: an International Journal*, 6:48–79, March 2019.

[9] William Stallings. *Computer Organization and Architecture*. Pearson Education, 10th edition, 2016.

[10] Samsung. Spotlight on the Exynos 5 Octa. `https://web.archive.org/web/20130818205335/http://www.samsung.com/global/business/semiconductor/minisite/Exynos/blog_Spotlight_on_the_Exynos5Octa.html`, April 2013.

[11] Karthik Iyer. Apple M1 series vs Intel Core i9-12900HK: Which laptop CPU is better? `https://www.xda-developers.com/apple-m1-vs-intel-core-i9-12900hk/`, April 2022.

[12] Ibrahim Takouna, Wesam Dawoud, and Christoph Meinel. Efficient virtual machine scheduling-policy for virtualized heterogeneous multicore systems. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 1–7, January 2011.

[13] Yefu Wang, Xiaorui Wang, and Yuan Chen. Energy-efficient virtual machine scheduling in performance-asymmetric multi-core architectures. In *Proceedings of the 2012 8th International Conference on Network and Service Management (CNSM) and 2012 Workshop on Systems Virtualiztion Management (SVM)*, pages 288–294, 2012.

[14] Ching-Chi Lin, Chao-Jui Chang, You-Cheng Syu, Jan-Jan Wu, Pangfeng Liu, Po-Wen Cheng, and Wei-Te Hsu. An energy-efficient hypervisor scheduler for asymmetric multi-core. In *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*, pages 507–509, 2014.

Explainable Linear Regression

Anton Pirhonen
anton.pirhonen@aalto.fi

Tutor: Alexander Jung

Abstract

This paper reviews an explainable linear regression algorithm. The algorithm utilizes input from a user to improve the explainability of a standard linear model. The user input is used in the explainable empirical risk minimization process as a regularization term. The algorithm is implemented and tested with the task of maximum daytime temperature prediction. The experiments give information about the behaviour and the characteristics of the algorithm. The experiments show that using the explainable linear regression with an accurate user signal can improve the validation error of the model compared to standard linear regression. The asymptotic behaviour of the algorithm with respect to explainability is inspected. Variations of the algorithm are proposed to address the potential issues regarding the asymptotic behaviour and the explainability.

KEYWORDS: explainable linear regression, user signal, explainable empirical risk minimization, EERM, explainable machine learning

1 Introduction

Machine learning is utilized in many domains, such as scientific research, business and healthcare. A machine learning method is based on a mathe-

matical model which uses data to provide predictions. These models have varying complexities. Some of them are explainable with simple mathematics, such as a few multiplication and addition operations, whereas other models are essentially "black-boxes", meaning that it is impossible to understand why the model makes a certain prediction. To utilize machine learning models efficiently, it is important to ensure that users can trust the models' decision making [1].

Zhang et al. [2] present an algorithm which formalizes explainable linear regression. The algorithm utilizes input from the user to make its predictions explainable. In this paper, the algorithm is tested with the task of maximum daytime temperature prediction. The experiment results are used to research the behaviour and the characteristics of the algorithm.

The paper is organized as follows. Section 2 defines empirical risk minimization and the related mathematical concepts. Those concepts are then used in Section 3 to describe how the explainability of a machine learning model can be measured with explainable empirical risk minimization. Section 4 presents the experiments conducted with the algorithm and reviews the experiment results. The observations from the results are used in Section 5 to propose variations of the algorithm to address the potential shortcomings regarding explainability. Finally, Section 6 concludes the paper.

2 Risk minimization

This section presents the mathematical definition of risk minimization and related concepts. Section 2.1 defines the concepts of a hypothesis and a loss function, which are then used in Section 2.2 to define empirical risk minimization. These definitions follow the notation and terms used by Zhang et al. [2] and Jung [3].

2.1 Hypothesis and loss function

A supervised machine learning model learns a hypothesis h to compute a predicted label $\hat{y}$ from a data point $(\mathbf{x}, y)$. Each data point has a feature vector and a true label. The feature vector $\mathbf{x}$ consists of one or more features $\mathbf{x} = (x_1, ..., x_n)^T \in \mathbb{R}^n$, and y is the value of the true label, $y \in \mathbb{R}$. The

hypothesis can thus be expressed as

$$h(\cdot) : \mathbb{R}^n \to \mathbb{R} : \mathbf{x} \mapsto \hat{y} = h(\mathbf{x}).$$

The quality of predictions by a hypothesis h is measured with a loss function. $L((\mathbf{x}, y), h)$. A loss function measures the deviation of the predicted label $h(\mathbf{x}) = \hat{y}$ from the true label y. The choice of loss function depends on the type of the machine learning problem. The expected value of the loss function for a data point determines the expected loss, which is also called the risk.

$$\bar{L}(h) := \mathbb{E}\{L((\mathbf{x}, y), h)\}. \tag{1}$$

Machine learning attempts to find a hypothesis $\hat{h}$ with minimum risk $\bar{L}(\hat{h}) = \min_{h \in \mathcal{H}} \bar{L}(h)$. $\mathcal{H}$ denotes the hypothesis space of the machine learning method.

2.2 Empirical risk minimization

It is not possible to use minimum risk (1) in machine learning problems where we do not know the underlying probability distribution for the measured data points [3]. Instead, it is possible to approximate the risk (1) with the average loss using a training set of collected data points.

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), ..., (\mathbf{x}^{(m)}, y^{(m)})\}.$$

With this approximation we can replace the risk (1) with the empirical risk

$$\hat{L}(h|\mathcal{D}) := (1/m) \sum_{i=1}^{m} L((\mathbf{x}^{(i)}, y^{(i)}), h).$$

As with equation 1, the goal is to find a hypothesis $\hat{h} \in \mathcal{H}$ that minimizes the empirical risk

$$\operatorname*{argmin}_{h \in \mathcal{H}} \hat{L}(h|\mathcal{D})$$

3 Explainable empirical risk minimization

This section describes how the explainability of an algorithm is measured. Section 3.1 introduces the concept of a user signal. Section 3.2 describes how explainability is quantified and how the user signal is used to measure explainability. Finally, Section 3.3 presents the definition of the explainable linear regression algorithm formalized by Zhang et al. [2].

3.1 User signal

To formalize explainable empirical risk minimization, Zhang et al. [2] introduce a user signal u to each data point. The user signal u "characterizes a data point from a perspective of a specific human user" [2].

Many different kinds of user signals can be measured. The user signal u can, for instance, be the user's prediction about the label y based on the features x. Alternatively, the user signal can be a physiological measurement, such as the users heart rate. The heart rate could be measured from the user when presenting him or her the features, such as an image. The experiment in Section 4 simulates the user's prediction about the following day's weather as the user signal. A data set $\mathcal{D}$ where a user signal is introduced to each data point has the form

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}, u^{(1)}), \dots, (\mathbf{x}^{(m)}, y^{(m)}, u^{(m)})\}.$$

3.2 Quantifying explainability

Zhang et al. [2] measure the subjective explainability $E(h|u)$ of a prediction $\hat{y}$ by hypothesis h regarding a data point $(\mathbf{x}, y, u)$ with

$$E(h|u) := C - H(h|u). \tag{2}$$

In Equation 2 the term $H(h|u)$ denotes conditional entropy (3) which quantifies the users uncertainty about the prediction $\hat{y} = h(\mathbf{x})$. Zhang et al. [2] imply that a model with low conditional entropy (3) is more explainable to the user.

$$H(h|u) := -\mathbb{E}\Big\{\log(p(h(\mathbf{x})|u))\Big\} \tag{3}$$

The calibration constant C in Equation 2 is used to due to the convention that explainability is a non-negative quantity. In the explainable linear regression algorithm, the conditional entropy (3) is estimated by $\hat{H}(h^{(\mathbf{w})}|u)$.

$$\hat{H}(h^{(\mathbf{w})}|u) = \underset{\alpha \in \mathbb{R}}{\operatorname{argmin}} (1/m) \sum_{i=1}^{m} (\mathbf{w}^T \mathbf{x}^{(i)} - \alpha u^{(i)})^2$$

3.3 Explainable linear regression algorithm

The explainable linear regression algorithm formalized by Zhang et al. [2] uses a loss function, squared error loss, with a regularization term, measuring the subjective explainability, to compute the explainable empirical

risk L_{EER} (4).

$$L_{\text{EER}} = \sum_{i=1}^{m} \underbrace{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}_{\text{Squared error loss}} + \underbrace{\lambda(\mathbf{w}^T\mathbf{x}^{(i)} - \alpha u^{(i)})^2}_{\text{Subjective explainability}} \tag{4}$$

Algorithm 1 is obtained when the explainable empirical risk is used with a linear model.

Algorithm 1 Explainable linear regression

Require: explainability parameter λ, training set $\mathcal{D}$

1: solve

$$\hat{\mathbf{w}} \in \underset{\alpha\in\mathbb{R},\mathbf{w}\in\mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{m} (y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2 + \lambda(\mathbf{w}^T\mathbf{x}^{(i)} - \alpha u^{(i)})^2$$

return $h^{(\lambda)}(\mathbf{x}) := \mathbf{x}^T\hat{\mathbf{w}}$

The input parameter λ is called the explainability parameter. It controls the emphasis on the subjective explainability. High values of λ direct the algorithm to minimize the risk from the subjective explainability. When $\lambda = 0$ the explainable linear regression performs the same way as the standard linear regression.

In addition to finding the weights $\mathbf{w}$, the algorithm also finds a user signal scalar α that minimizes the L_{EER} (4). The α allows the user signal u to be scaled to the magnitude of the predicted label.

The algorithm assumes that the features $\mathbf{x}$ and the user signal u are realizations of jointly Gaussian random variables with mean zero [2].

4 Explainable linear regression experiments

This section presents the experiments conducted with the explainable linear regression algorithm (Algorithm 1). The goal of the experiments is to learn about the performance and the characteristics of the algorithm. The behaviour of the algorithm is studied with different user signals. Each experiment contains multiple observations which are obtained with different levels of emphasis on the subjective explainability.

4.1 Experiment setup

In Section 4, the explainable linear regression algorithm (Algorithm 1) is used to predict the maximum daytime temperature y in Kaisaniemi, Helsinki. The features x are the maximum daytime temperature measurements from nearby locations, Porvoo, Vuosaari, Hyvinkää, Nuuksio

and Tapiola, from the same day. User predictions about the maximum daytime temperature were not available. Thus, the user signal u is simulated with the maximum daytime temperature measurement from the previous day. The user signal value corresponds to the user predicting the following day's maximum temperature to equal the ongoing day's maximum temperature. The location of the user signal measurement varies depending on the experiment. These locations include Kaisaniemi, Nurmijärvi, Jyväskylä and Muonio. The features, the label and the user signal from a single day constitute a data point. The data point with index i has the following structure

$$\mathcal{D}^{(i)} = (x^{(i)}_{Hyvinkaa}, x^{(i)}_{Nuuksio}, x^{(i)}_{Porvoo}, x^{(i)}_{Tapiola}, x^{(i)}_{Vuosaari}, y^{(i)}_{Kaisaniemi}, u^{(i)}_{Kaisaniemi}).$$

In the experiments, the training set consists of 365 data points from the year 2021 and the validation set from 366 data points from the year 2020. The data points were measured and shared by the Finnish Meteorological Institute [4]. The results are plotted with Matplotlib [5].

In each experiment, the explainable linear regression model is trained with the data from the training set. The training produces a model, which is determined by the weight vector $\mathbf{w}$. A training error is obtained from the training process and it is equal to the minimal mean explainable empirical risk as defined in Section 3.3. The trained model is then used to predict the labels of the validation set. The validation error is obtained by calculating the mean squared error of the predicted labels and the true labels. The validation error is used to measure the performance of the model.

A scientific computing Python3 library, SciPy [6], was used to perform the explainable empirical risk minimization. More specifically, the explainable empirical risk minimization used the scipy.optimize.minimize function [7] utilizing the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm. The number of iterations was limited to 500.

Analysis of convexity of the explainable empirical risk (Equation 4) was performed to acquire information about the minimization result. The Hessian matrix $\mathbf{H}$ for subjective explainability $(\hat{y} - \alpha u)^2$ with respect to the predicted label $\hat{y}$ and the user signal scalar α is a positive semidefinite matrix with $u \in \mathbb{R}$. The eigenvalues are $\lambda_1 = 0$ and $\lambda_2 = 2(u^2 + 1)$.

$$\mathbf{H} = \begin{bmatrix} 2 & -2u \\ -2u & 2u^2 \end{bmatrix}$$

Thus, the explainable empirical risk (Equation 4) is a sum of non-negatively

weighted convex functions and therefore a convex function [8]. Convex optimization tools can be used for finding a global minimum of the explainable empirical risk.

4.2 Data and the statistical assumptions

As mentioned in Section 3.3, Zhang et al. [2] assume that the features x and the user signal u are realizations of jointly Gaussian random variables with mean zero. Statistical tests are used to examine whether the data conforms to these assumptions.

Table 1 displays the means and normality tests of the features and the user signals. The tests are conducted with the programming language R [9]. Normality is tested with Shapiro-Wilk and Bowman-Shenton tests which both share the null hypothesis that the data is obtained from a Gaussian distribution. The values of Shapiro-Wilk and Bowman-Shenton rows in Table 1 correspond to the p-values of these hypotheses. According to Table 1, the null hypothesis of normality can be rejected with 95% confidence for all features and user signals using either one of the tests. Additionally, the means of the features and the user signals are above zero. In the following experiments the means are not shifted to zero. Thus, the weather data used for the experiments does not fulfill the assumptions by Zhang et al. [2]. The experiments are expected to yield insights about the performance and the characteristics of the explainable linear regression algorithm despite the lack of normality and zero means.

	$x_{Hyvinkaa}$	$x_{Nuuksio}$	x_{Porvoo}	$x_{Tapiola}$	$x_{Vuosaari}$
Mean temperature (°C)	9.7	9.3	9.2	9.7	8.8
Shapiro-Wilk	1.1e-03	2.9e-04	1.1e-03	1.1e-03	4.9e-03
Bowman-Shenton	1.1e-02	5.9e-03	2.0e-02	1.5e-02	3.1e-02

	$u_{Kaisaniemi}$	$u_{Nurmijarvi}$	$u_{Jyvaskyla}$	u_{Muonio}
Mean temperature (°C)	9.7	9.2	8.0	3.9
Shapiro-Wilk	1.3e-03	7.8e-04	6.3e-05	3.2e-03
Bowman-Shenton	1.8e-02	9.8e-03	3.1e-03	1.3e-02

Table 1. Means and normality tests for the user signal and the features

In addition to non-normal and non-zero mean data, the selection of closely related, collinear features might affect the regression. According to Gareth [10], "collinearity reduces the accuracy of the estimates of the

regression coefficients". Collinearity can be measured with the variation inflation factor (VIF). Table 2 displays the variation inflation factor scores related to each feature. Gareth [10] recommends using features with VIF scores less than 5 or 10. The VIF scores of the chosen features exceed the recommended limits. Additionally, the VIF scores of a linear model using the combination of any two of the features exceed the recommended VIF scores. This observation suggests, that choosing only one of the features would be beneficial for the experiment. This problem was noticed after conducting the experiments. Thus, the experiment setting can be improved by choosing features which are not significantly collinear.

	$x_{Hyvinkaa}$	$x_{Nuuksio}$	x_{Porvoo}	$x_{Tapiola}$	$x_{Vuosaari}$
VIF	166	252	258	310	153

Table 2. The variance inflation factors (VIF) of the features

4.3 Experiment with an accurate user signal

In the first experiment the explainable linear regression algorithm (Algorithm 1) is tested with a user signal measured at the label location, Kaisaniemi. The user signal u is the maximum daytime temperature from the previous day. Figure 1 displays the label and the user signal plotted on a line chart. The figure shows that the difference between the user signal and the label is small and their values are highly correlated. This user signal can be thought of as a very good user signal which contains valuable information about the label.

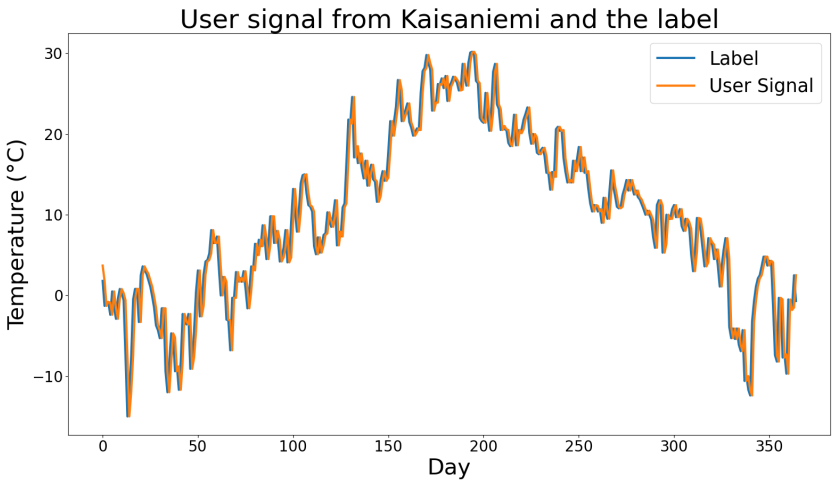


Figure 1. Label to user signal

Figures 2 and 3 show the training and validation error of the explain-

able linear regression, respectively. The errors are measured as a function of the base two logarithm of the explainability parameter λ. The transformation to $\log_2(1 + \lambda)$ is applied to enable visualizing errors with different magnitudes of λ.

Figure 2 shows that the training error increases as a function on λ. The training error of the explainable linear regression with $\lambda = 0$ corresponds to the training error of the standard linear regression with squared error loss as the loss function. The squared error loss is already minimized when $\lambda = 0$. Thus, an increase in λ will incur a greater training error due to the algorithm having to select the weights to compromise between the risk from the squared error loss and the subjective explainability.

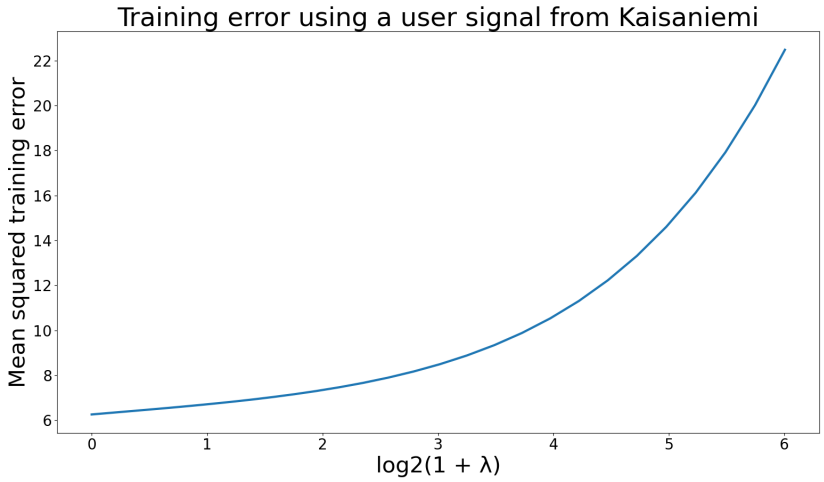


Figure 2. Training error

In Figure 3, the validation error first decreases until $\log_2(1+\lambda) \approx 0.9$ and then starts to increase. In this experiment, the explainable linear model using the optimal explainability parameter has a smaller validation error than the standard linear model ($\lambda = 0$). This observation implies that using an accurate user signal in model training can slightly improve the model's predictions.

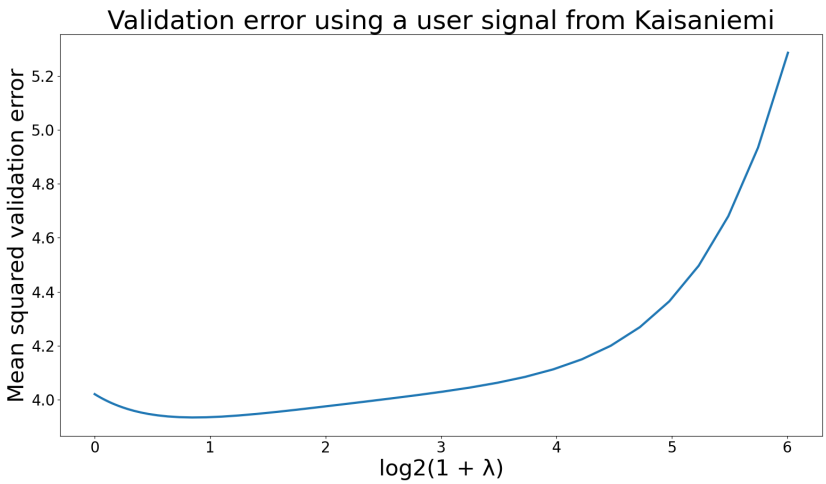


Figure 3. Validation error

4.4 Using noisier user signals

The user signal of the first experiment, as seen in Figure 1, is a very accurate user signal. This section experiments with signals from greater distances to introduce noise to the user signal. User signals with noise are expected to give less reliable information about the label value. The new user signal locations, Nurmijärvi, Jyväskylä and Muonio, are located 33, 235 and 869 kilometers from Kaisaniemi, respectively. Models trained with these user signals are compared with the model of the first experiment from Section 4.3. Additionally, the range of the explainability parameter λ is increased.

Figure 4 visualizes the user signal from Muonio. The deviation between the user signal and the label is greater than in the last experiment. The root mean squared errors (RMSE) between the user signals and the label are shown in Table 3. The RMS errors increase as we move further away from the label location.

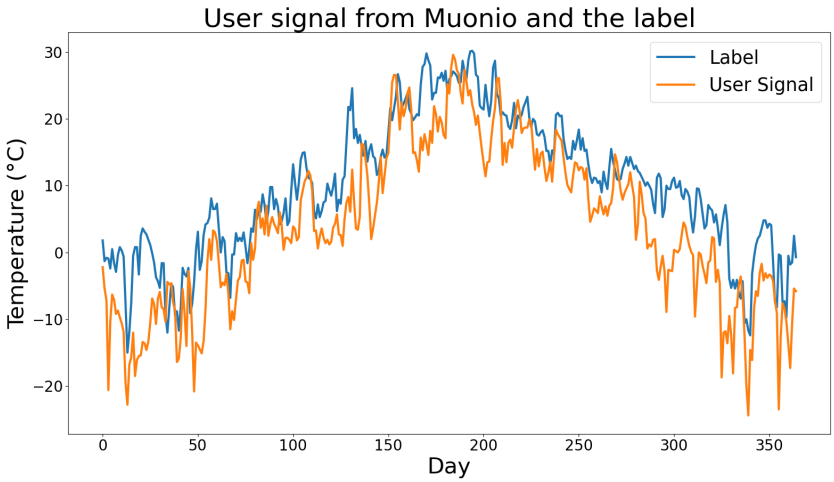


Figure 4. Label to Muonio user signal

	$u_{Kaisaniemi}$	$u_{Nurmijarvi}$	$u_{Jyvaskyla}$	u_{Muonio}
RMSE (°C)	2.6	3.0	3.8	7.7

Table 3. The root mean squared errors (RMSE) between the user signals and the label

Figure 5 displays the training error of the explainable linear regression using a noisier user signal from Muonio. The training error increases faster than in the first experiment. A similar increase in the growth rate of the training error is observed with the user signals from Nurmijärvi and Jyväskylä, although the growth rate of the training error was not as large as with user signal from Muonio.

Additionally, the training error with the user signal from Muonio increases steadily until $\lambda \approx 2^5$ before starting to plateau. The plateauing behaviour is inspected in a further experiment in Section 4.6.

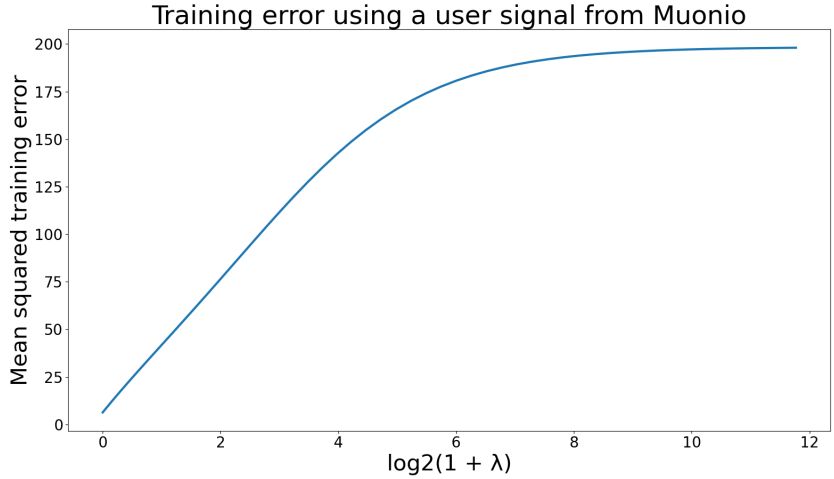


Figure 5. Training error with user signal from Muonio

Figure 6 shows the validation error with user signals from Nurmijärvi. The validation error with Nurmijärvi user signal increases steadily and with the explainability parameter λ. A closer inspection of the validation error in Figure 7 shows that the Nurmijärvi user signal is not able to improve the validation error compared to the standard linear regression. A similar behaviour is observed with the validation errors using user signals from Jyväskylä and Muonio. Only the Kaisaniemi user signal was able to improve the validation error of the explainable linear regression compared to the standard linear regression.

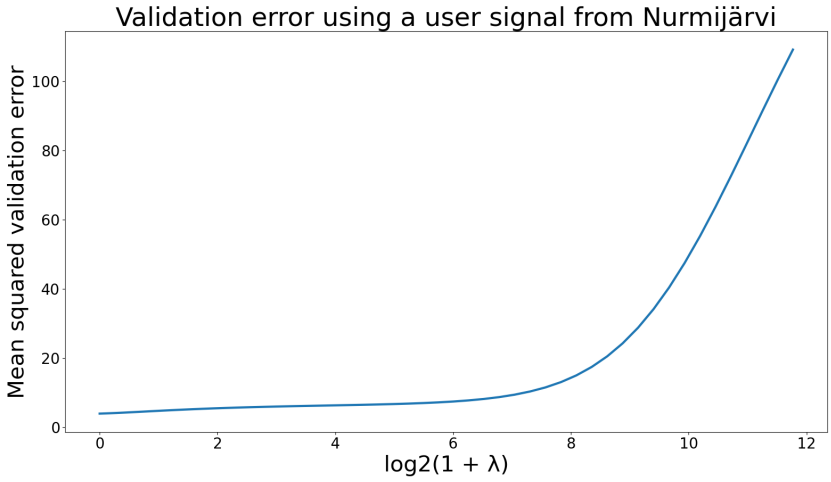


Figure 6. Validation error with user signal from Nurmijärvi

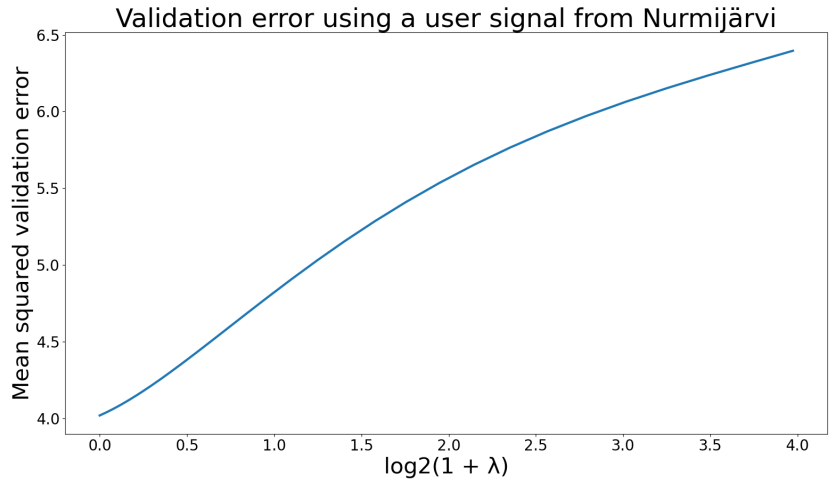


Figure 7. Zoomed validation error with user signal from Nurmijärvi

The Muonio validation error in Figure 8 shows an error plateauing behaviour similar to that observed with the Muonio training error in Figure 5.

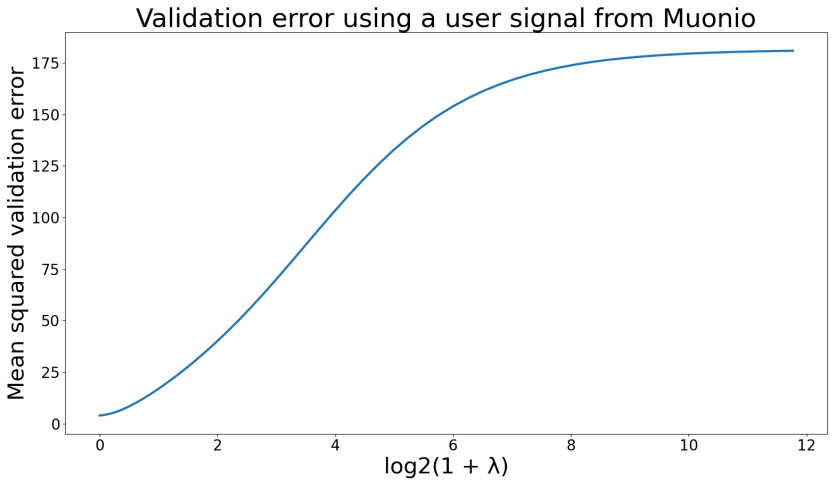


Figure 8. Validation error with user signal from Muonio

The experiments with user signals containing more noise demonstrate that the capability of the user signal to improve the model's validation error depends on the quality of the user signal. Only the Kaisaniemi user signal was able to improve the validation error of the explainable linear regression compared to the standard linear regression. In this experiment, a greater root mean squared error between the user signal and the label in Table 3 corresponded with greater validation errors.

4.5 Weight behaviour

In this section, the behaviour of the linear model weights are observed. Figure 9 shows the behaviour of the linear model weights as λ increases. The Nurmijärvi user signal is used. The distances between the features and the user signal measurement locations are displayed in Table 4. Figure 9 shows that as λ increases the features measured near the user signal, in Hyvinkää and in Nuuksio, retain higher weights than the three other features, which quickly near zero. Root mean squared errors in Table 4 confirm that the measuring stations which are located close to the user signal measurement location measure maximum daytime temperatures closer to the user signal. When the deviation from the user signal is heavily penalized with large values of λ, the linear regression weights favor the features which predict the user signal. The behaviour of the weights might also be affected by collinearity discussed in 4.2. Thus, the effect of the explainable linear regression on the feature weighing should be researched more. Additionally, Figure 9 shows that the explainable linear regression weights become closer to zero with large values of λ.

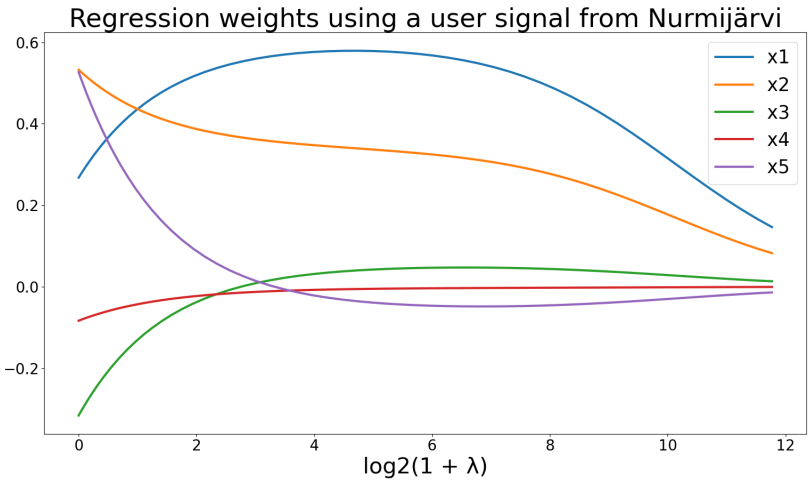


Figure 9. Linear regression weights with user signal from Nurmijärvi

Feature	x_1	x_2	x_3	x_4	x_5
Location	Hyvinkää	Nuuksio	Porvoo	Tapiola	Vuosaari
Distance (km)	19	24	49	32	34
RMSE (°C)	0.82	0.76	1.90	1.46	2.71

Table 4. Distance and RMSE between the user signal and a feature

4.6 Asymptotic behaviour of the explainable linear regression algorithm

The observation of plateauing training and validation error made in Section 4.4 and the observation of weights nearing zero as with large values of λ in Section 4.5 arouse interest in the asymptotic behaviour of the explainable linear regression algorithm with respect to the explainability parameter λ. This section studies the regression weights and the user signal scalar α with large values of λ.

Figure 10 shows the regression weight behaviour using the user signal from Muonio. Figure 11 plots the values of the α with the same user signal. These figures show that as λ increases, both the weights and α near zero.

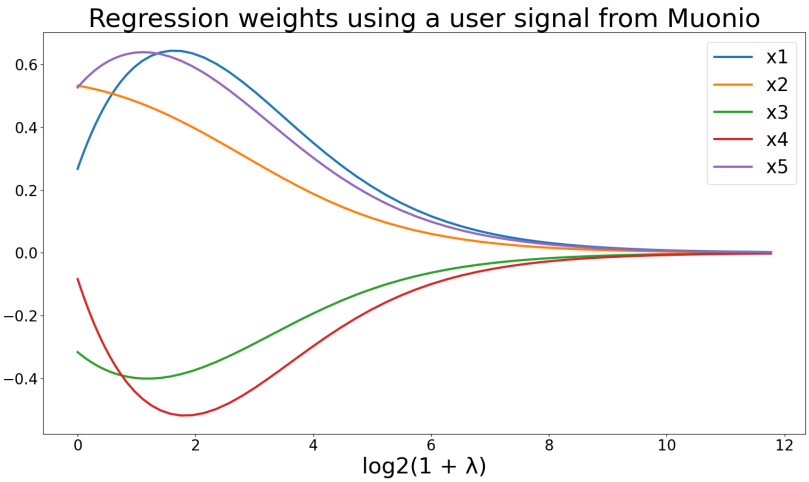


Figure 10. The weights as a function of λ with user signal from Muonio

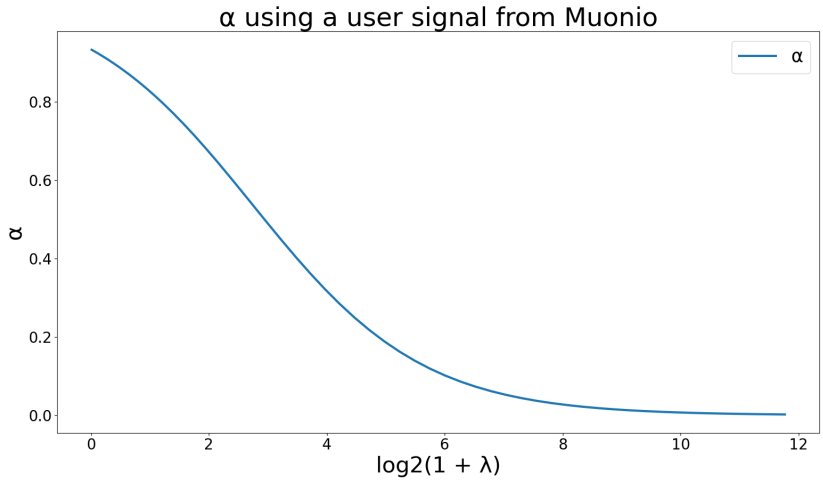


Figure 11. The value of user signal scalar α as a function of λ

When λ is increased enough, the risk from the subjective explainability is so heavily penalized that the algorithm starts to focus on minimizing it almost exclusively. It is possible to minimize the subjective explainability by setting the weight vector $\mathbf{w}$ and the α to zero. Thus, the resulting risk from subjective explainability will also equal zero. In this case, the model predicts zero despite the values of the features. This explains the plateauing behavior of errors observed in Section 4.4. As λ increases, the validation error of the model approaches the squared error loss of a linear model with all weights set to zero.

The observed asymptotic behaviour might be a downside of defining explainability using conditional entropy. Consider a user, who makes predictions about the following day's maximum temperature. He or she might be surprised that a model using his or her input to create a more explain-

able model predicts the maximum temperature to always be zero when the focus on subjective explainability is extremely large. In this case, the model's prediction might deviate from what the user would expect when supplying the user signal. In this context, defining explainability as the deviation between the user prediction and the model prediction might work better.

5 Improving the explainable linear regression

This section explores possibilities of improving the explainable linear regression. This section proposes two different variations of the algorithm: the other for situations where the user signal does not need to be scaled, and the other for performing the α scaling and explainable linear regression separately.

5.1 Explainable linear regression without α

As noted in Section 4.6, both the weights $\mathbf{w}$ and α near zero when subjective explainability is increased enough. The α in Figure 11 starts to decrease immediately when λ is increased. Even though the weight zeroing behaviour is not as significant with small values of λ, it might still have an unnecessary effect on the regression, favouring smaller weights.

In the special case, when the user attempts to predict the value of the label, the user signal could be used without scaling. In this case, the subjective explainability could be defined as the squared error between the user prediction and the model prediction $(\mathbf{w}^T\mathbf{x}^{(i)} - u^{(i)})^2$. This could be useful in situations similar to experiment conducted in Section 4.

Explainable linear regression without the scalar α would not have the drawback of the algorithm zeroing the weights and the α when minimizing the risk from the subjective explainability. Was α removed, the explainable linear model would perform the regression between the user signal and the label. With large values of the explainability parameter λ, the linear model would fit the user signal instead of the label.

5.2 Performing the scaling and the regression separately

The weight zeroing behaviour could potentially be avoided by performing the user signal scaling and the explainable linear regression separately. In the first step, both the user signal and the label are normalized to mean

zero. After that the algorithm finds α that minimizes the squared error between the user signal and the label.

$$\alpha \in \underset{\alpha \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{m} (y^{(i)} - \alpha u^{(i)})^2$$

The goal of process is to scale the user signal u as close to the label y as possible. After finding the optimal α, the explainable linear regression could be used to fit the weights $\mathbf{w}$ only.

$$\hat{\mathbf{w}} \in \underset{\mathbf{w} \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{m} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2 + \lambda (\alpha u^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2$$

This would avoid the weight and user signal diminishing behaviour resulting from attempting to find the α and the weights at the same time.

6 Conclusion

This paper reviewed an explainable linear regression algorithm proposed by Zhang et al. [2]. The algorithm finds a linear model that minimizes the explainable empirical risk consisting of the squared error loss and a regularization term measuring subjective explainability. The algorithm was implemented with Python3 and scientific computing library SciPy [6]. The experiment data was provided by the Finnish Meteorological Institute [4].

The implemented algorithm was tested with the task of maximum daytime temperature prediction. The experiments utilized various user signals and suggested that the user signal effect on the validation error depended on the quality of the user signal. Using an accurate user signal with the explainable linear regression had the potential to decrease the validation error of the linear model compared to standard linear regression. This decrease in validation error was identified with one user signal that had the lowest root mean squared error with the label. Using the other three, more inaccurate user signals only increased the validation error. Another observation was that the explainable linear regression weights potentially favoured the features that predicted the user signal as the emphasis on the subjective explainability increased.

The explainable linear regression algorithm asymptotically approaches a linear model with zero weights as the explainability parameter λ increases. This results from the minimization of the risk from the subjective explainability term. The algorithm minimizes the subjective explainability by setting the regression weights $\mathbf{w}$ and user signal scalar α close to

zero. A linear model with zero weights is not useful for maximum daytime predictions and might also be confusing for the user.

This paper proposed two methods for improving the asymptotic behaviour of the algorithm. The first method can be used in problems where the user tries to predict the label. The second method separates the user signal scaling with α into a separate process, after which the linear regression is performed. Whether the proposed methods improve the explainable linear regression algorithm in terms of validation error or explainability, requires further research.

References

[1] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[2] L. Zhang, G. Karakasidis, A. Odnoblyudova, L. Dogruel, and A. Jung. Explainable empirical risk minimization, 2022.

[3] Alexander Jung. *Machine Learning*. Springer, Singapore, 2022.

[4] The finnish meteorological institute, 2022. https://en.ilmatieteenlaitos.fi/download-observations.

[5] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[6] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[7] Scipy manual: scipy.optimize.minimize, 2022. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[9] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017.

[10] Gareth James. *An introduction to statistical learning : with applications in R*. Springer texts in statistics. Springer, New York, 2013 - 2013.

Co-inference techniques for Edge and Fog computing

Soumya Lekkala
soumya.lekkala@aalto.fi

Tutor: Vesa Hirvisalo

Abstract

With the increase in the number of Internet of Things (IoT) devices, there is a large volume of data being generated and meeting the demanding requirements of the IoT applications has been challenging with cloud computing alone. This led to the new paradigms, edge and fog computing. The main purpose of these paradigms is to satisfy the real-time critical application needs by utilizing less energy thus improving the performance in a distributed computing environment. When Deep Learning (DL) is used within IoT, these paradigms need efficient inference methods to provide low latency, energy-efficient, high-performance results. However, it is a challenge to lower the inference costs. To minimize these inference costs, and to improve the accuracy many inference techniques are proposed. This paper overviews the co-inference techniques of edge and fog computing, such as model compression, offloading, model partitioning and model early exit.

KEYWORDS: Internet of Things, Deep Learning, Machine Learning, Artificial Intelligence, Edge computing, Fog Computing, Co-inference techniques

1 Introduction

Recently, the IoT is a trending word and plays a significant role in day-to-day activities. With the ever-growing data and the extensive use of IoT applications, approximately 41.6 billion IoT devices and 79.4 Zettabytes (zt) of data is projected to be generated in 2025 [1]. These globally connected digital devices constitute the IoT and possibly generate a large amount of data, which is stored in the cloud, and the current network architectures will not be able to process it. Data traffic might occur due to the simultaneous access of data from various regions, and this can cause some delays in processing the data.

Moving massive data from IoT devices to the cloud may not be efficient due to bandwidth limits, and it will be challenging to meet the extremely low latency needs of applications like health monitoring and self-driving automobiles [2]. Cloud computing cannot afford the bandwidth needed, and a little delay in the detection of obstacles might cause a major accident in self-driving cars. To mitigate these bandwidth constraints and to achieve the low-latency requirements of the applications and real-time decision making, edge and fog computing have been proposed.

Edge and fog computing are the cloud computing extensions that are still being researched. The edge and fog computing domains, on the other hand, have their own set of criteria and constraints, necessitating the employment of efficient inference methods when applied together with AI. One such method is Deep Neural Networks (DNNs) for reliable inference [3]. On-Device inference techniques, such as designing compact networks, model compression methods and co-inference techniques at the edge, such as offloading, DNN model partitioning and model early exit. These inference techniques are designed to lower inference costs.

This paper overviews edge and fog computing, and highlights their challenges along with the co-inference techniques in edge and fog computing. Section 2 defines edge and fog computing, while Section 3 overviews Machine Learning (ML) and Artificial Intelligence (AI). Section 4 discusses the co-inference techniques in edge and fog computing. Finally, Section 5 gives the concluding remarks.

2 Edge and Fog Computing

Cloud computing is the offering of computing services on demand. Cloud computing reduces the maintenance cost, and the information technology overhead to the end-users [4]. However, cloud data centres are built far away from the end-users, which might result in data transmission delays. With the ever-growing use of IoT devices, satisfying the demands for critical and low-latency applications is not possible [5]. Edge and fog computing are proposed to address these issues of cloud computing.

2.1 Edge Computing

Edge computing is a type of cloud computing that extends beyond the cloud. Here, computation happens at the network's edge in close proximity to the IoT device, reducing the latency and enabling real-time services. These computation devices are referred to as Edge nodes. Edge computing provides storage and computing resources closer to the user and located near end devices [6].

2.2 Fog Computing

Fog computing is a type of computing that sits between the edge and the cloud. Fog nodes receive a significant volume of data, analyze it, and only transfer the most critical information to the cloud. Fog nodes are more scalable than edge computing since they are closer to the cloud and can be placed in multiple geographical locations.

Figure 1 shows how cloud, edge and fog computing are connected. Edge nodes are near the IoT devices and fog nodes are near the cloud.

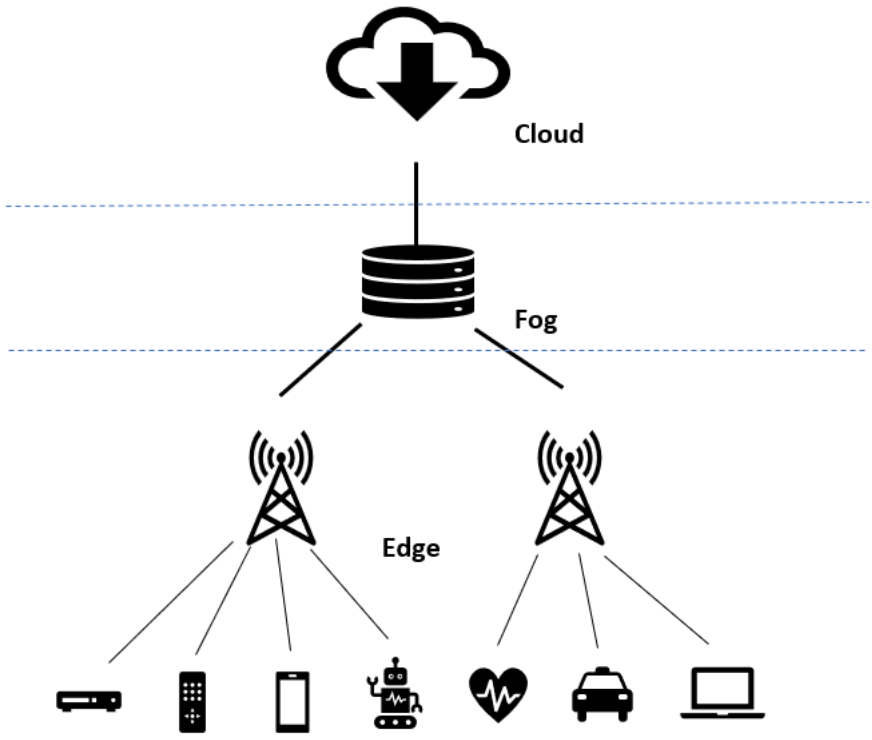


Figure 1 Cloud, Edge and Fog computing

2.3 Challenges of Edge and Fog Computing

By shifting some computational resources, fog and edge computing could provide a number of benefits. The main purpose of these paradigms is to create an IoT network environment with a large number of interconnected distributed heterogeneous devices with the goal of deploying and managing critical applications closer to the user. However, designing such platforms meeting all of their needed properties is very difficult [7].

Resource management, security and privacy, and network management are the main challenges of edge and fog computing. Edge and fog computing emerged to bring the computational resources near to the end nodes, and closer to the IoT devices. Resource management techniques are required to optimize the utilization of available resources to provide scalability and desired performance and to utilize all the benefits provided by edge and fog computing. The main challenges in resource management are resource estimation - to estimate the number of resources required to perform a particular task, resource discovery - to identify the available resources, resource allocation - to get the identified resources and allocate them for the task and resource optimization - a technique to optimize the

utilization of available resources [7].

New security and privacy issues also have been raised along with the advantages of edge and fog computing. Due to the distributed architecture of these paradigms, authentication has become a major issue. Maintaining confidentiality, integrity and authentication called as CIA triad model has been a major challenge [8]. Maintaining and protecting the personal details of the user is also challenging in a distributed network. Since edge and fog computing involves heterogeneous devices distributed across, one more challenge here is managing the network to accommodate all the tasks which are needed to be performed [7].

3 Machine Learning and Artificial Intelligence

AI is defined as the decision making and human intelligence capabilities of computers. AI has been developed rapidly over the years and is used in day-to-day life applications. ML methods are instrumental for the design and analysis of AI [9]. ML refers to training the data sets, identifying patterns, and making decisions without human intervention. There are several basic to complex ML algorithms, such as decision trees, k-means clustering and logistic regression. Deep learning is a different branch of ML, which allows automatic learning by taking in a large amount of unstructured data, and has more complex algorithms [1].

The development process for software systems becomes much more difficult when it includes ML-based components. The training iterations that are used to develop the best possible prediction model are at the heart of the machine learning process. Modern software development approaches, such as DevOps, have become popular, and they often emphasize frequent development iterations and continuous software delivery. DevOps is a software development methodology that emphasizes collaboration between software development and operations to deliver software changes more quickly. Integrating ML model workflow with DevOps is required for faster results in the development process [10].

A group of highly intensive computational models is referred to as DL. Fully connected multi-layer neural networks are an excellent example, as they need a precise estimation of a huge number of network parameters. The cornerstone for reaching this goal is the availability of a significant volume of data. DL also has many challenges, one such is training the models. Huge volumes of training data is required to get the highly ef-

ficient and desired outcomes [11]. These challenges call for distributed and federated learning. Distributed learning has centralized data but multiple nodes for model training, whereas federated learning has decentralized data and training but effectively has a central model. The recent trend is towards learning environments, such as Ray and Rlib for Reinforcement Learning (RL). Ray RLlib is a distributed execution toolkit for RL based on the Ray framework.

Edge and fog computing when applied together with AI requires efficient inference techniques. There are various types of DNNs, such as Convolutional Neural Networks (CNNs), Artificial Neural Networks (ANNs), Recurrent Neural Networks (RNNs), and with these DL models can be integrated with the edge computing environment.

4 Co-inference Techniques

Inference is the process of deploying the DNN models onto a device, which then looks for the patterns which it has been trained to identify. Inference in the cloud may incur additional network delays, and this cannot be tolerated for time-critical applications. Training and inference can be done in many ways. DNN models are designed to reduce these inference costs and to improve accuracy. To optimize these inference costs and co-inference in the edge, few inference techniques are available.

4.1 On-Device Inference

Generally, the computational costs of the DNN models can be reduced using two approaches. One approach is to design the model compact and efficient. The other is to optimize the trained networks. Common optimization technique is model compression.

Designing Compact Networks:
Usually, scientists strive to reduce the number of parameters while preserving accuracy. Their goal is to develop a compact and efficient model. A variety of models are used for this purpose. Instead of the normal convolution, these models do a factorized convolution, thereby reducing the computational parameters while maintaining great accuracy [1].

Model Compression:

Model compression reduces the model complexity and resources with a slight loss in accuracy. Parameter pruning and sharing, Knowledge distillation and quantization are a few of the model compression techniques.

Parameter pruning removes the redundant parameters to improve the performance. Model pruning strategies are classified as structural or non-structural. Non-structural pruning depends on the specific algorithm or hardware [12] whereas structural pruning does not depend on a specific platform and can be run directly on the model. Structural pruning is more effective than non-structural.

The on-device inference is made possible by knowledge distillation, a model compression method, which teaches a smaller network to understand the behaviour of the large network by trying to replicate its output. Visual recognition, speech recognition, natural language processing (NLP), and recommendation systems are all examples of how knowledge distillation is utilized in artificial intelligence. Additionally, knowledge distillation can be utilized for a variety of other objectives, including data privacy and defense against hostile attacks. Three distillation schemes for both student and teacher models are present based on whether the teacher model is being updated simultaneously with the student model or not [13]. 1) Offline distillation, here teacher network is pre-trained and then during distillation it is used to train the student model. 2) Online distillation - To further improve the performance of the student model and both student and teacher models updated simultaneously [14].3) Self distillation, this is can be viewed as a type of online distillation where the same network is used for both student and teacher models.

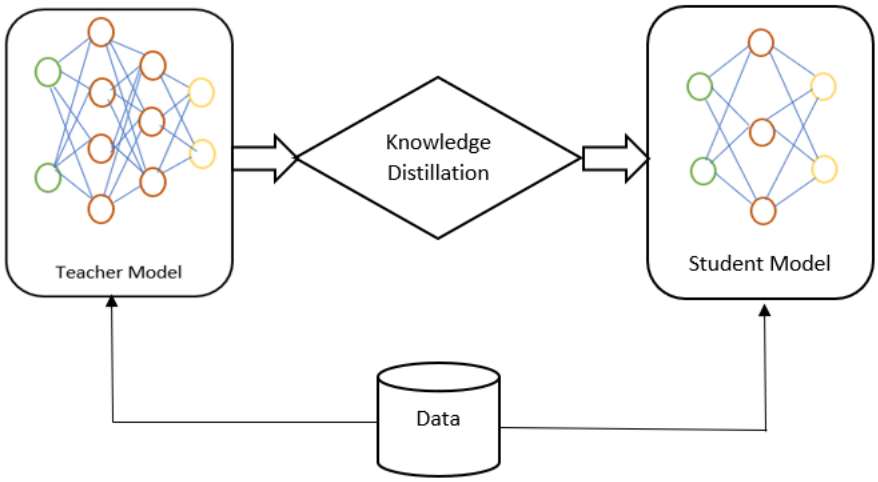


Figure 2 Knowledge Distillation - Basic teacher student network model.

Quantization achieves higher computation speed without sacrificing accuracy by using low bit data computation. Deep Compression: a three-stage pipeline, Compressing DNNs with pruning, trained quantization and Huffman coding all together reduces the storage requirements of neural networks by 35x to 49x without compromising accuracy. [12] A Huffman code is an encoding algorithm used for lossless data compression [15].

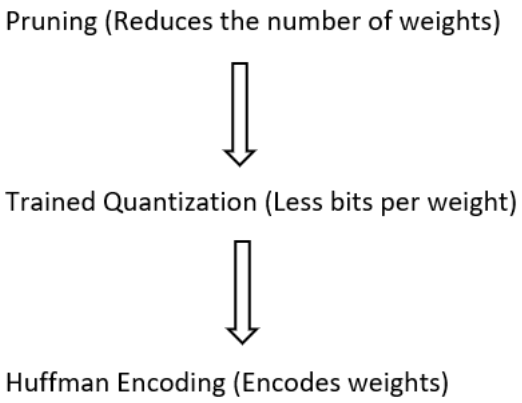


Figure 3 Three stage pipeline of Deep Compression

4.2 Coinference at the Edge

The aforementioned inference techniques make the inference directly on the end device. However, these are not feasible for deploying a complex, high computation model. Here, one good option is to offload the computations to local end devices or to segment the models into partitions [1]

Offloading

Offloading plays a significant role in edge computing through dynamic partitioning between cloud data centres and edge servers and edge devices [16]. Through offloading, it is possible to scale the computational servers and improve their performance. Computational offloading is used to achieve fast inference by offloading computations to more stable remote servers thus reducing the computation load on the devices and improving the performance. Leveraging the remote servers to supplement the computing capabilities of less powerful devices, such as mobiles, is the main purpose of offloading [17]

DNN Model Partitioning

Offloading techniques are highly dependent on the network and unpredictable server availability. Model partitioning takes into account the structure of the DNN and divides its layers into several parts. For computation, some layers are offloaded to the server, some are executed on the end devices directly. This approach reduces latency and provides better performance. The most common method of partitioning DNN models is done horizontally whereas vertical partition is done on CNN models. By utilizing the computational capacity of the end devices, the power consumption of the IoT device is also reduced. However, there will be a delay at the partitioning points. When it comes to partitioning the DNN model, special attention should be given on where to partition it. Neurosurgeon is a system that intelligently splits DNN models at the granularity of neural network layers to achieve the optimal latency and energy usage at the end device [18].

Figure 4 illustrates how DNN model partitioning can be done and allocated to multiple devices, few are offloaded to the edge nodes, few directly to end devices such as mobiles, and few to the cloud.

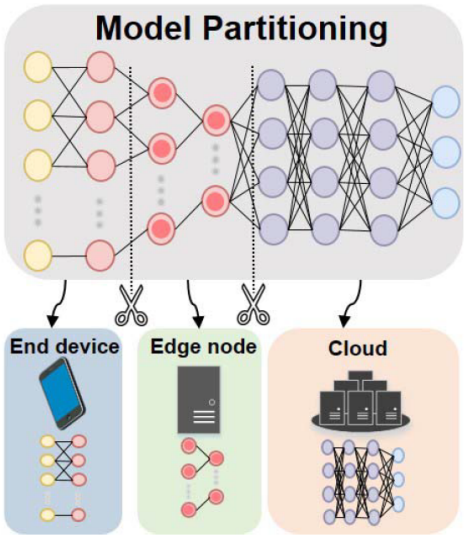


Figure 4 DNN Model Partitioning at edge adapted from [1]

Model Early Exit

To lessen the increasing costs, side branches are added to the main branch of neural architecture, allowing specific samples to exit early stages avoiding layer-by-layer processing for all layers.

BranchyNet is one such fast inference with an early exit model. BranchyNet, a neural network architecture in which side branches are added to the main neural network branch, to allow some test samples to escape early,

is presented to reduce these rising inference costs. This new architecture takes advantage of the fact that characteristics learnt earlier in the deep network's life cycle can typically properly predict a substantial chunk of the data population. BranchyNet significantly decreases the runtime and energy usage for the majority of samples by departing these samples with prediction at earlier levels and so avoiding layer-by-layer processing for all layers [19].

Figure 5 shows how the partial DNN model employs the model early exit approach for DL inference at the edge to extract features by quitting at an early stage, decreasing communication delays and resulting in rapid inference.

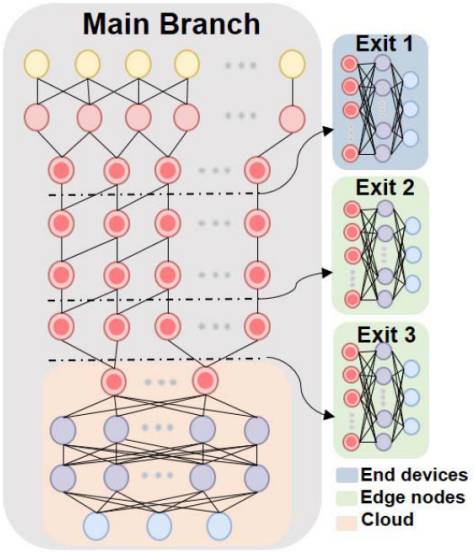


Figure 5 Model Early Exit adapted from [1]

5 Conclusion

Edge and fog computing are the future technologies, and there is a lot of research being done in these fields. When these domains are combined with AI, effective inference approaches are required to get the most out of them. The ML models are trained and appropriate inference techniques are selected based on the application usage, needs (such as low latency, high availability, energy-efficient), and design. Many experiments are still ongoing on these co-inference techniques and new techniques are emerging to satisfy the critical requirements of the industry.

References

[1] Zhuoqing Chang, Shubo Liu, Xingxing Xiong, Zhaohui Cai, and Guoqing Tu. A survey of recent advances in edge-computing-powered artificial intelligence of things. *IEEE Internet of Things Journal*, 8(18):13849–13875, 2021.

[2] Ben Zhang, Nitesh Mor, John Kolb, Douglas S. Chan, Ken Lutz, Eric Allman, John Wawrzynek, Edward Lee, and John Kubiatowicz. The cloud is not enough: Saving IoT from the cloud. In *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*, Santa Clara, CA, July 2015. USENIX Association.

[3] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, 2015.

[4] M. A. Vouk. Cloud computing – issues, research and implementations. *IEEE Journal of computing and information technology*, 16(4):235–246, 2008.

[5] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

[6] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P Jue. All one needs to know about fog computing and related edge computing paradigms. *Journal of Systems Architecture*.

[7] Schahram Dustdar, Cosmin Avasalcai, and Ilir Murturi. Invited paper: Edge and fog computing: Vision and research challenges. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pages 96–9609, 2019.

[8] Anjum Khairi, M Farooq, M Waseem, and S Mazhar. A critical analysis on the security concerns of internet of things (iot). *Perception*, 111, 2015.

[9] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

[10] Lucy Ellen Lwakatare, Ivica Crnkovic, and Jan Bosch. Devops for ai – challenges in development of ai-enabled applications. In *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6, 2020.

[11] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, 19(6):1236–1246, 05 2017.

[12] S Han, H Mao, and WJ Dally. Compressing deep neural networks with pruning, trained quantization and huffman coding. arxiv 2015. *arXiv preprint arXiv:1510.00149*.

[13] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, Jun 2021.

[14] Defang Chen, Jian-Ping Mei, Can Wang, Yan Feng, and Chun Chen. Online knowledge distillation with diverse peers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3430–3437, Apr. 2020.

[15] Jan Van Leeuwen. On the construction of huffman trees. In *ICALP*, pages 382–410, 1976.

[16] Congfeng Jiang, Xiaolan Cheng, Honghao Gao, Xin Zhou, and Jian Wan. Toward computation offloading in edge computing: A survey. *IEEE Access*, PP:1–1, 08 2019.

[17] Li Lin, Xiaofei Liao, Hai Jin, and Peng Li. Computation offloading toward edge computing. *Proceedings of the IEEE*, 107(8):1584–1607, 2019.

[18] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *SIGARCH Comput. Archit. News*, 45(1):615–629, apr 2017.

[19] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016.

Automated probing of firewalls for small business networks

Anand Vasudevan
anand.vasudevan@aalto.fi

Tutor: Tuomas Aura

Abstract

This paper explores a way to automatically probe firewalls for security flaws in a small network with an IPSec VPN configured in tunnelling configuration in a controlled environment. It describes the workings of a firewall, VPN, IP address spoofing and packet sniffing. It uses a set of python scripts with scapy and nmap to simulate network packets from various attackers that have gained complete control over the internet (Dolev Yao attacker). The final test suite is meant to run in a vagrant virtual machine environment for the CS-E4300 course at Aalto University.

KEYWORDS: firewall, small network firewalls, automated firewall testing, IPSec VPN

1 Introduction

A firewall is a network security tool that is responsible for keeping out unnecessary packets of communication from an internal network. Firewalls are present at the very edge of a network, between any internal devices on the network (e.g., computers and smartphones) and the internet. This way, all network packets that are going to, or coming from, the internet have to pass through the firewall before they can arrive at their destina-

tion.

This paper aims to explore an automated testing tool that is used to check the efficacy of a firewall used in a small home or office network. These kinds of networks are usually not professionally monitored for security flaws, and so, they are more susceptible to attacks from bad actors on the internet. When network complexities, such as IoT devices and VPNs, are added to the mix to the mix, the attack surface only increases [7].

This paper explores a method used for checking firewall correctness in a small network scenario with three locations all connected to each other through the internet. This testbed is quite similar the one used for the CS-E4300 Network Security course at Aalto University [8]. Each of the networks has multiple devices internally. The cloud server hosts services that were previously contained on the same premises and on the same network as the users. The devices on the client side are now connected to the cloud using a site-to-cloud VPN (Virtual Private Network). This increases the complexity of the firewall rules that need to be set up on either side.

A further aim is to create an automated suite of tests that probes the networks in the testbed and checks if they are correctly configured. The operation and working of this suite should be described in this paper. This test suite should run automatically on a regular basis to check if there have been any changes to the firewall configuration. The test should also be triggered manually when a new device is added to any of the sites. The test suite should be able to pinpoint what firewall vulnerabilities exist in each network and hint at what rule would be required to patch that vulnerability.

The outcome will be a set of scripts that are tailored to work in a setup similar to the testbed, not a generic all-encompassing tool that will work for all different network topologies.

The rest of the paper is structured as follows. Section 2 will explore other work done in the field of firewalls for small networks and firewall checking. Section 3 elaborates the exact problems that can arise in configuring a firewall for this testbed and how some of those can be mitigated. Section 4 will elaborate on the solution implementation and operation, and explore its functionalities. Section 5 will show the results of different test case scenarios the solution works effectively under. Section 6 will discuss potential improvements to the solution and its limitations. Section 7

will conclude the paper.

2 Background

2.1 Firewall

A firewall is a perimeter defence mechanism used to ensure that only necessary traffic is allowed to enter (or exit) a network. In general, firewalls work on the Internet layer of the Internet protocol suite, TCP/IP (Transmission Control Protocol/ Internet Protocol) suite, to filter IP packets. There are two broad types of firewalls: stateful and stateless. Stateless firewalls are simple packet filters that analyze each packet without any memory or context. Stateful firewalls, on the other hand, are more complex and remember the nature of different connections established from the network.

In this paper, we will proceed with the assumption that the firewalls are implemented using simple stateless packet filters. This filters packets based on a fixed set of rules, set by a network administrator, that contains a pre-condition and a resultant action. The conditions match different fields in the IP packet, mainly the source and destination IP address, the source and destination port, and the protocol used to transport the data. The actions include "pass", "log", or "drop". More specifically, the paper targets a use case where the firewall is set up using the "iptables" [13] command on a Linux router.

2.2 Virtual Private Network

A Virtual Private Network (VPN) is a tool used to extend a virtual network across different geographic locations. It is most commonly used along with security measures to encrypt and protect the data between the different locations it covers. While there are many types of VPNs, this paper will focus on the use of an Internet Protocol Security (IPsec) VPN [12] that is typically used to connect two different sites together, as opposed to VPNs that connect a single device to a site or another single device. The testbed configuration uses IPSec in tunnel mode, ensuring security by encrypting and encapsulating the original IP packet that it needs to transport within an IPSec IP packet.

In the testbed, ideally, IPSec is configured between the gateways of the

different sites in tunnel mode [6]. This is a type of VPN which is set up using the routers at both ends of the networks that want to be bridged to each other. Every packet that passes through one router, destined towards the other end of the IPSec tunnel, is encrypted and encapsulated within an IPSec packet. So to anybody sniffing packets outside the tunnel, they will only see the headers of the encapsulating packet and all of the details of the original packet will be encrypted, including the original headers on the packet. This way the devices within each network can continue to operate without any reconfiguration or knowledge about the change in the network configuration.

Figure 1. Simple IPSec packet encapsulation

3 Project aim

3.1 Testbed

The testbed used is quite similar the one used for the CS-E4300 Network Security course at Aalto University [8]. However, our program should be flexible enough to work with multiple client sites connecting to multiple different server sites.

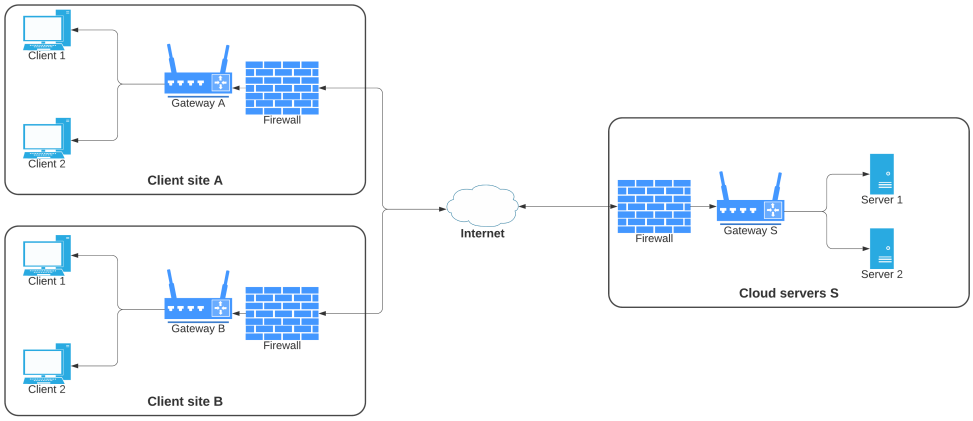


Figure 2. Testbed for the paper

3.2 Output format

The final project should produce a binary executable file that future students of the CS-E4300 course can viably use to check the correctness of

their project submissions. The program will provide an output with different categories of data: the basic setup of the network, the firewall configuration for the functioning of the VPN, and extra firewall configuration for added security. On each of these fronts, the program will run a suite of tests and provide an output for each.

The execution of the program can be augmented by providing a configuration file with details about the network topology, VPN setup, and exemptions to the assumed flow of data. This will allow users to alter the program to work with different network configurations and topologies. Additionally, the default functionality of the program can be easily edited to work with different requirements for future iterations of the course.

Although this kind of rudimentary program is not feasible for any large scale use in the real-world, it can be a good starting point to verify the basic functionality of a similar topology. There are plausibly many real world setups that closely resemble the project testbed that we have selected, as shown in the project PDF [9].

4 Solution

4.1 Tools and technologies

The project is implemented using python scripts because of pythons ubiquity in Linux systems and its ease of use to program and maintain. The python scapy [5] packet management library is used to carry out different network functions including sending packets, spoofing IP addresses, and sniffing packets. Additionally, the nmap program [1] is used to probe the gateway to see if there are any open ports that an attacker could target.

Spoofing IP packets
IP address spoofing [10] is a technique used to change the source IP addresses of an IP packet to be different from the real IP address of the sender. This can be used to deceive the receiver of the packet to believe that the sender was the machine with the spoofed IP address.

Sniffing packets
Sniffing a network packet is a way for a computer to monitor and see the details of internet packets that are targeted to, or pass through, one of its network interfaces. A packet sniffer, also knows as a packet analyzer [4],

is a program that captures the packet, extracts information about it, and logs the information to some location.

Running tests
In the testbed, all the computers are created using virtual machines using vagrant [2], all using Ubuntu 18.04. The tests have to be executed on different virtual machines, sometimes simultaneously. This is because the router needs to be used to sniff the different packets that passes through it and check if there are responses to the spoofed packets sent by the clients.

There is a YAML configuration file into which the user can enter the gateway IP addresses of each of the clients and the cloud location into.

The test suite is initiated by running a single script on the host machine which triggers the appropriate test to be run on the virtual machine and collates all the results into a single location.

4.2 Test cases covered

The attacks on the system can be categorized in two ways: where the attack originates and whom the attack targets. The attack can originate from a client site, a server site, or the rest of the internet (Dolev Yao attacker [11]). The attack can target a client site or a server site.

The use case for the testbed in question is that the devices from the client site and the server site do not need to be in contact with any devices other than their corresponding client or server, and a designated administration server that is in charge of maintenance and updates. This means that the firewall and VPN should be configured to ensure that the two sites are connected to the cloud server and no other non-VPN traffic should be allowed to flow through the gateways. Additionally, the two client sites are to be isolated from each other, i.e., client A should not be able to tamper with the server B.

Positive cases
The positive cases covered include ensuring that the tunnel is set up properly by checking the appropriate IPsec ports are open and checking that the client can successfully communicate with the server. Without any additional tests, these positive cases does not guarantee that the VPN and firewall are configured appropriately, but combined with the results of the negative cases a complete picture of the system is visible.

Outside attackers

In the testbed, since the router handles all the traffic between the sites, a Dolev Yao attacker can be simulated by controlling the router to intercept and spoof packets.

If the router can successfully access either of the servers using a spoofed packet, pretending to be a random outside IP address that doesn't belong to any client, any real-world attacker can also access the server. This is a severe security flaw that exposes the servers and means that the firewall was not set up to filter non-client IP addresses from accessing the server.

If the router can successfully access a server using a spoofed packet, pretending to be one of the client gateways, then a real-world attacker who is on-path [3] will be able to spoof an IP address to target the server. This means that the firewall configuration does not prevent non-VPN tunnel packets from reaching the server.

Cross site attackers

A real world analogue to the testbed in this paper would be a startup company that offers cloud services to replace bare-metal servers for multiple clients. In this scenario, even another client hosted in the same cloud location could be a potential attacker to a clients server. For example, client B could send a packet to server A to poision the data.

Another such attack, which is more difficult to execute by client B, involves client B spoofing the IP address of client A and making a request to the server through the VPN tunnel. TO defend against this, the VPN needs to be configured to separate the two clients into their own tunnels without allowing traffic from one tunnel to be passed to the other server. The firewall and VPN configuration should ensure that these types of attacks are not possible.

4.3 Test results

The results of the all the tests are written in a text file and shown to the user per test. There is a message shown for each use case to indicate what kind of problems their configuration has without revealing what the immediate fix could be.

5 Conclusion

The aim of the project was to explore the field of automated firewall and VPN configuration testing and to make a tool that can be used by students taking the CS-E4300 course at Aalto University. After exploring the possible security vulnerabilities in a firewall and ways to combat these using python, this was accomplished using python scripts, the scapy library, and nmap.

The same testing suite can be used for the WireGuard VPN project of the CS-E4300 as well. Since both the projects use the same testbed with minimal changes in the expected outcome, this same program can be used with minimal modifications.

There is considerable scope for improvement for this type of app being used as a more generic teaching tool. With some additional graphical reporting that visual explains the weaknesses in the system combined with some graphics explaining the correct solution, this tester could be a more useful method of teaching.

Another possible improvement would be to enable this project to check for any bonus edge scenarios that are not part of the project description. For example, Docker containers can be used on the server side instead of running a dedicated server for each client. In a real-world scenario, this would come with considerable cost savings, and hence is another use case that adds complexity in the firewall rules.

References

[1] Nmap: the Network Mapper - Free Security Scanner.

[2] Vagrant by HashiCorp.

[3] What is an on-path attacker?

[4] Packet analyzer, April 2022. Wikipedia, the free encyclopedia.

[5] Scapy, April 2022.

[6] Tunneling protocol, April 2022. Wikipedia, the free encyclopedia.

[7] Gunes Acar, Danny Yuxing Huang, Frank Li, Arvind Narayanan, and Nick Feamster. Web-based Attacks to Discover and Control Local IoT Devices. In *Proceedings of the 2018 Workshop on IoT Security and Privacy*, IoT S;P '18, pages 29–35, New York, NY, USA, August 2018. Association for Computing Machinery.

[8] Tuomas Aura. tuomaura/cs-e4300_testbed, January 2022. original-date: 2020-11-17.

[9] Tuomas Aura and Aleksi Petonen. Project 2 Network Security CS-E4300, 2021-2022, September 2021.

[10] Bastian Ballmann. *Understanding Network Hacks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[11] Danny Dolev. On the Security of Public Key Protocols. *IEEE TRANSACTIONS ON INFORMATION THEORY*, (2):11, 1983.

[12] Naganand Doraswamy. *IPSec*. Prentice-Hall PTR Web infrastructure series. Prentice Hall PTR, Place of publication not identified, 2nd ed. edition, 2003.

[13] Herve Eychenne. iptables(8) - Linux man page.

Task Allocation for Vehicular Fog Computing: A Review

Lizzy Tengana
lizzy.tenganahurtado@aalto.fi

Tutor: Wencan Mao

Abstract

Vehicular Fog Computing (VFC) has emerged as an answer to the demand of efficient computational power in vehicular scenarios. Reaching computational resources in the cloud implies a network delay that latency-sensitive applications cannot afford. Hence, binding computational units to vehicles and stations along the road (fog nodes) can enable vehicle clients to borrow these resources and offload tasks that require more computational capacity than what is locally available. However, scheduling tasks efficiently in an extremely dynamic vehicular environment is known to be an NP-hard problem [5] due to the combinatorial nature of matching a set of tasks to a set of fog nodes while considering the constraints of each participant. Thus, this article reviews the state-of-the-art approaches to task allocation for vehicular fog computing and provides a summary of the challenges faced in the literature thus far as well as the ones forthcoming.

KEYWORDS: Vehicular fog computing, edge computing, computation offloading, dynamic task allocation.

1 Introduction

The emerging growth of computational power in a hyper-connected world has brought the idea of real-time applications from fiction to reality. Applications such as autonomous driving, augmented reality, emergency failure reaction and smart city management require resource intensive tasks to be performed with almost no delay [5, 15, 19]. From real-time object recognition to crowdsourced decision-making, these tasks cannot afford the round trip time to reach the cloud and be resolved [20, 21], thus, vehicular fog computing (VFC) has risen as a feasible solution to bring a pool of computational resources near service consumers and data sources [3].

A fog computing node usually refers to any server located in the path from the cloud to the clients of the network [10]. Vehicular for computing focuses on the challenges that emerge from allowing computing resources to change their physical location dynamically, i.e., fog node mobility. The VFC field studies the interaction between clients and fog nodes, particularly, how to optimize the way in which vehicular clients can harness idle computing resources from mobile and stationary fog nodes [6]. The complexity of introducing mobile fog nodes lies on the heterogeneity of resources that each node brings; in this scenario, efficient task allocation strategies are needed to make the best use of the computational power available at any given moment.

The complexity of task allocation is far from trivial. It has been found to be a non-convex and NP-hard optimization problem because of its restrictions regarding processing and transmission latency, quality of information and energy utilization [5]. Several approaches to optimize task allocation have been proposed covering a variety of contexts and objectives. For instance, the energy efficiency aspect of VFC has been widely researched [1, 4, 7, 8, 16, 17], thus, this work reviews the latest directions in the VFC field beyond that of energy efficiency, which include the minimization of processing and transmission latency [5, 14, 19, 20, 21], the maximization of quality of information [19, 20] and quality of results [21], ensuring reliability and fault tolerance [5], and decision making under information asymmetry and uncertainty [18].

This paper is organized as follows. Section 2 presents the background and key aspects relevant to VFC. Section 3 presents the the state-of-the-art approaches to task allocation in VFC. Section 4 discusses the perfor-

mance of said strategies and gives an outlook to future trends in task allocation in VFC. Finally, Section 5 provides concluding remarks.

2 Fundamentals of Task Allocation in Vehicular Fog Computing

This section comprises the core concepts in the study of task allocation in VFC, ranging from fundamental definitions to the main challenges to consider when transitioning from a stationary resource ecosystem to one that can take advantage of the temporary proximity of heterogeneous resources.

2.1 The need for Vehicular Fog Computing

As stated in the introduction, the VFC field is concerned with finding the best possible use of idle computing resources in the path from the cloud to the edge of the network [10]. These idle resources are located in fog nodes (servers) that can be either stationary or mobile. In particular, having mobile fog nodes near the edge of the network, as in taxis, buses and other vehicles, introduces the complex problem of optimizing the demand and supply of resources in many vehicular use cases. These VFC use cases include autonomous driving [18], augmented reality [18], visual-based diving assistance [19], emergency failure management [5] and collection of information of common interest [15, 20].

2.2 Task Allocation Complexity

In VFC, task allocation refers to transferring resource-intensive computation from resource-constrained clients to external fog nodes that posses enough idle resources to lend them to nearby consumers [11]. This client-server environment is highly dynamic due to the mobility of its actors, and finding the best possible match between tasks and fog servers requires an exhaustive search among all possible pears, i.e., a combinatorial effort that would take non-deterministic polynomial time to complete [18]. Consequently, due to the time complexity of this problem, modern strategies to task allocation in near real-time imply a trade-off between service latency and accuracy of results.

2.3 Task Migration

The challenge of task migration is often encountered in VFC because there are no guarantees of task completion in complex mobile environments once the allocation has been decided. This migration process can be managed in a centralized [5, 21] or decentralized [19] manner. Centralized approaches usually place the responsibility of reassigning incomplete tasks on a zone head, which monitors the whole life-cycle of tasks within its communication range. Decentralized approaches, on the other hand, may leave this decision to the task owner.

2.4 Network Latency

One of the main motivations for pushing computational resources closer to the edge of the network is precisely the need to optimize the performance of latency-sensitive tasks [2]. Critical tasks, such as emergency failure reaction in autonomous driving require a handling time of less than 50ms [13]. However, optimizing only the latency would leave behind critical objectives in different use cases. In this sense, Zhu et al. [21] consider the joint optimization of service latency and quality loss to improve real-time situational awareness and cooperative lane change. In subsequent works, Zhu et al. optimizes data quality and latency in the particular case of image processing for visual-based assisted driving [19] and visual crowdsourcing of events of common interest [20]. Hou et al. [5] highlights the importance of reliable task allocation and fault tolerance under latency constraints to enable auto/assisted driving and emergency failure management for the future of intelligent transportation systems. Shi et al. [14] introduces task prioritization into the latency optimization problem to promote idle resource sharing among vehicles using a dynamic pricing model.

2.5 Bandwidth Consumption

The exponential growth of internet connected electronic devices and the emergence of intelligent transportation systems constantly increases the need of higher data transfer throughput, which is reflected in network bandwidth consumption. As indicated by Hou et al. [5], the heterogeneity of computing resources, failure rates and communication equipment among fog nodes hinders the forecasting of bandwidth consumption. Fur-

thermore, Zhu et al. [21] observe that under limited bandwidth, the size of the data is the main factor in transmission delay.

2.6 Security and Privacy

There are few works related to the security and privacy of task allocation in VFC. Hou et al. [5] explores the reliability and reprocessing needs when allocating critical tasks. The authors propose allowing partial computation offloading of tasks than can be executed in parallel and reprocessing of tasks as necessary. Here, reliable communication occurs within a software defined network (SDN) and its complexity of orchestration is overcome by using a fault-tolerant particle swarm optimization algorithm (PSO).

2.7 Situational Awareness

Situational awareness refers to collecting raw data from any given context, translating it into structured information and infer an accurate big picture of a situation with respect to a specific goal [12]. As the goal often involves decision making, it is paramount that this whole processing occurs in a timely manner. Given that there are not two identical tasks that would happen in the exact same context, there is a need for dynamic task prioritization [14], as well as data-sharing schemes, such as crowdsourcing, to obtain the most relevant up-to-date information [9].

3 Modern Approaches to Task Allocation in VFC

3.1 Optimization

Linear programming optimization (LPO) and particle swarm optimization (PSO) have been studied to offload tasks from clients to vehicular fog nodes [21, 5]. Linear programming optimization (LPO) has been used to solve an optimization matrix that has tasks from client vehicles and available fog nodes as the rows and columns, and task transmission rate between them as the values [21]. Then, an efficient linear programming solver finds an optimal matching for the tasks and the available fog resources.

On the other hand, algorithms based on particle swarm optimization

(PSO) have also been used to find the best allocation of tasks to fog nodes with respect to given metrics (e.g., service latency) [21, 5]. Using PSO, all possible task allocation decisions for a set of tasks and fog nodes comprise a search space, and the particles represent a subset of candidate decisions that can change over time to find an optimal solution. Within the search space, the next move of a particle is influenced by the best candidate it has found so far and the global optimum candidate found so far by the swarm. By this iterative process, the swarm is expected to eventually converge to the optimal task allocation decision.

3.2 Task Allocation as a Partially Observable Markov Decision Process

Task allocation has also been modeled as a Partially Observable Markov Decision Process (POMDP). Zhu et al. [19], for instance, designed a POMDP that considered the workload fluctuation in vehicular fog nodes as a Markov chain. This chain has a transition probability matrix that represents different workload states within time intervals and is updated regularly with the goal of maximizing cumulative task utility at each time interval. In this framework, the task allocation is initiated by a client vehicle, which broadcasts a message within its range of communication, then, the nearby fog nodes reply with their location and transition probability matrix, and finally, the client selects fog node that offers the least communication delay.

3.3 Reinforcement Learning

Due to the high complexity of the task allocation problem, deep learning algorithms have been proposed to learn the best allocation strategies from the data itself [20, 14]. Zhu et al. [20] modelled the task allocation problem as a Markov decision process where vehicles are agents that interact with an environment by transitioning through different states and collecting rewards. As possible states in VFC can be infinite, the authors proposed an advanced Deep Q-network (DQN), which approaches the state scalability issue by learning the optimal parameters of a neural network. DQN in based on Q-learning (or quality learning), a reinforcement learning algorithm that can find the optimal policy in a Markov decision process based solely on experience.

Another example of task allocation using reinforcement learning was

presented by Shi et al. [14]. In this work, task allocation is again formulated as a Markov decision process, then, a fog node controller collects information about the agents in its environment and feed it into a soft actor-critic (SAC) reinforcement learning algorithm. This SAC makes decisions based on the outputs of three neural networks (NN), on for the actor and two for the critic, where the actor NN focuses on improving its task allocation policy while the critic NNs jointly evaluate the policy and influence the actor's decisions. This SAC strategy can manage infinite action spaces, as in the case of VFC, and enable a maximum entropy configuration to allow the exploration of more alternative strategies within an action space.

3.4 Task Allocation as a Stable Matching Problem

Task allocation between client vehicles and fog nodes is a combinatorial problem that can be approached as a stable matching problem where both clients and fog nodes maintain preference lists based on their own priorities [18]. For instance, Zhou et al. [18] presented a task offloading as a two dimensional stable matching problem with preferences based on service latency and dynamic prices of computation. A client's preference list is sorted according to its goal of minimizing the latency of task computation and network transmission. On the other hand, a fog node's preference list is organized to maximize its earnings, thus, the fog node only accepts a client proposal when there no other proposals in its list. This task allocation process based on preference lists occurs in rounds; first, the client vehicles propose their tasks to idle fog nodes, then the fog nodes count the number of proposals and if some receive more than one, those would raise the price of their computation and notify their prospect clients. This step is repeated until there is only one client vehicle proposing to a fog node.

4 Discussion and Future Directions

Solving the task allocation problem entails the optimization of a dynamic matching problem known to be NP-hard [18]. Therefore, finding the absolute best matching between tasks and fog nodes is infeasible in near real time. Nonetheless, there can be infinite ways of approaching the search for the a good-enough matching under given constraints, including latency and quality of service. Thus, the relevant constraints for a "good"

dynamic matching depend on the use cases, and the state-of-the-art approaches and results are not straightforwardly comparable. Tables 1 and 2 present a side-by-side overview of the strategies found in recent literature as a tool to identify the similarities, differences and nuances of task allocation for vehicular scenarios.

4.1 Open Issues

Issues that require further research regarding task allocation in vehicular fog computing are detailed as follows:

- Cache systems: benefiting from similar tasks that have taken place in the past and have an efficient access to the most relevant ones [19].

- Vehicular coordination: collaborative situational awareness by having real time access to road state information [19]

- Mobile node diversity: considering not only traditional transportation systems like cars and buses, but also other vehicles that can contribute to a resource pool, such as drones [19, 20].

- Distributed learning: recent advances in federated and parallel learning can increase the efficiency of training AI algorithms [20] and the availability of information of common interest.

- Standardization: in order to have an extensive appropriation of vehicular fog computing, standardizing protocols are a need yet to be met [15].

5 Conclusion

This work presented an overview of the state-of-the-art strategies for task allocation in vehicular fog computing. These strategies strive to overcome the time complexity challenge of exhaustive search by proposing different optimizations that go beyond service latency to include quality of information, reliability guarantees, fault-tolerance, task prioritization and utility schemes. Furthermore, VFC has the potential to benefit from recent advances in Artificial Intelligence, and bring latency-sensitive use cases, such as reliable autonomous driving, closer to reality.

Table 1. Task allocation methods, context/use cases, objectives and constraints.

State-of-the-art	Method	Context / Use cases	Objective	Constraints
Folo [21]	Linear programming optimization (LPO). Binary particle swarm optimization (BPSO).	Time critical and data intensive tasks.	Optimization of latency and quality of service.	Service latency. Quality loss of results. Fog capacity.
EC-SDIoV [5]	Particle swarm optimization.	Autonomous/ assisted driving, emergency failure management.	Fault-tolerant task allocation and latency optimization.	Latency.
Chameleon [19]	Partially Observable Markov Decision Process.	Visual-based assisted driving.	Optimize latency and image resolution awareness as a proxy for data quality.	Latency. Data quality. Processing delay. Spatiotemporal variation in vehicular traffic density.
Flexsensing [20]	Reinforcement learning: Deep Q-network (DQN)	Vehicle-based crowdsourcing of visual data.	Maximize quality of information (QoI) and minimize processing latency.	Latency.
Priority-aware [14]	Reinforcement learning: Soft Actor-Critic (SAC) neural networks	Sharing computational resources among vehicles.	Maximize the average utility when sharing comutational resources.	Latency. Priority. Utility.
Matching-learning [18]	Matching-learning based on latency and dynamic prices of computation.	Autonomous driving, visual based assisted driving.	Optimization of task allocation under information asymmetry and uncertainty.	Information asymmetry. Information uncertainty. Non-negative utilities. Reward fairness.

Table 2. Task allocation inputs, outputs, time complexity and performance.

State-of-the-art	Inputs	Outputs	Time Complexity	Performance
Folo [21]	Traces of clients and mobile fog nodes. Location of zone head. Unassigned task set.	Task assignment decision.	LPO: $O((\|J\|\|K\| + 2\|K\|+1)^3.5xB^2)$, fog nodes J, tasks K, bit count B. BPSO: heuristic (not provided).	Average service latency reduced by 27% and quality loss reduced by 56% (against random fog node selection).
EC-SDIoV [5]	Location of clients, fog nodes, zone head. CPU cycles, data size and latency per task.	Best position of the particle swarm. Reconstructed target function.	$O(u*MaxG*2^{p+q+1})$	Service latency is improved between 14% and 72% compared to different task allocation schemes.
Chameleon [19]	Object recognition profiles. Trajectories of fog nodes and clients.	Task assignment decision.	Not provided.	65% reduction in service latency and a 83% increase in average image resolution.
Flexsensing [20]	Traffic traces of clients and fog nodes.	Task assignment decision.	Depends on forward and backward propagation of the DQN.	51% reduction in in average processing latency and 34% increase in quality of information collected.
Priority-aware [14]	Task set. Input data size per task. CPU cycles required per task. Delay constraint per task. Task priority.	Task assignment decision.	$O(KT)$, T is a given time period, K is the set of vehicles within the communication range of a task owner vehicle.	Aprox. 100% complete tasks for beyond 15 vehicles/km
Matching-learning [18]	Vehicles with residual computational resources. Pending tasks.	Task assignment decision.	$O(Nz)(N \geq M)$ or $O(Mz)(M \geq N)$, z is the number of price raising operations.	Round-trip delay of nearly 100ms and 17% delay reduction compared to exhaustive search.

References

[1] Zheng Chang, Zhenyu Zhou, Tapani Ristaniemi, and Zhisheng Niu. Energy efficient optimization for computation offloading in fog computing system. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[2] Mung Chiang and Tao Zhang. Fog and iot: An overview of research opportunities. *IEEE Internet of things journal*, 3(6):854–864, 2016.

[3] Alisson Barbosa De Souza, Paulo AL Rego, Tiago Carneiro, Jardel Das C Rodrigues, Pedro Pedrosa Rebouças Filho, Jose Neuman De Souza, Vinay Chamola, Victor Hugo C De Albuquerque, and Biplab Sikdar. Computation offloading for vehicular environments: A survey. *IEEE Access*, 8:198214–198243, 2020.

[4] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H Luan, and Hao Liang. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE internet of things journal*, 3(6):1171–1181, 2016.

[5] Xiangwang Hou, Zhiyuan Ren, Jingjing Wang, Wenchi Cheng, Yong Ren, Kwang-Cheng Chen, and Hailin Zhang. Reliable computation offloading for edge-computing-enabled software-defined iov. *IEEE Internet of Things Journal*, 7(8):7097–7111, 2020.

[6] Xueshi Hou, Yong Li, Min Chen, Di Wu, Depeng Jin, and Sheng Chen. Vehicular fog computing: A viewpoint of vehicles as the infrastructures. *IEEE Transactions on Vehicular Technology*, 65(6):3860–3873, 2016.

[7] Yuyi Mao, Jun Zhang, and Khaled B Letaief. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12):3590–3605, 2016.

[8] Olga Munoz, Antonio Pascual-Iserte, and Josep Vidal. Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading. *IEEE Transactions on Vehicular Technology*, 64(10):4738–4755, 2014.

[9] Jianbing Ni, Aiqing Zhang, Xiaodong Lin, and Xuemin Sherman Shen. Security, privacy, and fairness in fog-based vehicular crowdsensing. *IEEE Communications Magazine*, 55(6):146–152, 2017.

[10] Carlo Puliafito, Enzo Mingozzi, Francesco Longo, Antonio Puliafito, and Omer Rana. Fog computing for the internet of things: A survey. *ACM Transactions on Internet Technology (TOIT)*, 19(2):1–41, 2019.

[11] Firdose Saeik, Marios Avgeris, Dimitrios Spatharakis, Nina Santi, Dimitrios Dechouniotis, John Violos, Aris Leivadeas, Nikolaos Athanasopoulos, Nathalie Mitton, and Symeon Papavassiliou. Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions. *Computer Networks*, 195:108177, 2021.

[12] Mahadev Satyanarayanan. Edge computing for situational awareness. In *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pages 1–6. IEEE, 2017.

[13] Philipp Schulz, Maximilian Matthe, Henrik Klessig, Meryem Simsek, Gerhard Fettweis, Junaid Ansari, Shehzad Ali Ashraf, Bjoern Almeroth, Jens Voigt, Ines Riedel, et al. Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture. *IEEE Communications Magazine*, 55(2):70–78, 2017.

[14] Jinming Shi, Jun Du, Jingjing Wang, Jian Wang, and Jian Yuan. Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12):16067–16081, 2020.

[15] Jingjing Wang, Chunxiao Jiang, Kai Zhang, Tony QS Quek, Yong Ren, and Lajos Hanzo. Vehicular sensing networks in a smart city: Principles, technologies and applications. *IEEE Wireless Communications*, 25(1):122–132, 2017.

[16] Changsheng You, Kaibin Huang, Hyukjin Chae, and Byoung-Hoon Kim. Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, 16(3):1397–1411, 2016.

[17] Pengtao Zhao, Hui Tian, Cheng Qin, and Gaofeng Nie. Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. *IEEE Access*, 5:11255–11268, 2017.

[18] Zhenyu Zhou, Haijun Liao, Xiaoyan Wang, Shahid Mumtaz, and Jonathan Rodriguez. When vehicular fog computing meets autonomous driving: Computational resource management and task offloading. *IEEE Network*, 34(6):70–76, 2020.

[19] Chao Zhu, Yi-Han Chiang, Abbas Mehrabi, Yu Xiao, Antti Ylä-Jääski, and Yusheng Ji. Chameleon: Latency and resolution aware task offloading for visual-based assisted driving. *IEEE Transactions on Vehicular Technology*, 68(9):9038–9048, 2019.

[20] Chao Zhu, Yi-Han Chiang, Yu Xiao, and Yusheng Ji. Flexsensing: A qoi and latency-aware task allocation scheme for vehicle-based visual crowdsourcing via deep q-network. *IEEE Internet of Things Journal*, 8(9):7625–7637, 2020.

[21] Chao Zhu, Jin Tao, Giancarlo Pastor, Yu Xiao, Yusheng Ji, Quan Zhou, Yong Li, and Antti Ylä-Jääski. Folo: Latency and quality optimized task allocation in vehicular fog computing. *IEEE Internet of Things Journal*, 6(3):4150–4161, 2018.

Cloud and Local Game Streaming

Sergei Kaukiainen
sergei.kaukiainen@aalto.fi

Tutor: Esa Vikberg

Abstract

KEYWORDS: Gaming, cloud gaming, video games, quality of service

1 Introduction

Major progress in information and communications technology (ICT) in recent decades have enabled many new services for consumers' everyday life. Probably one of the biggest has been video streaming over the internet. Another service that has grown its popularity in the same way, especially during the last few years, is cloud gaming or sometimes called Games/Gaming as a Service (GaaS). This can be explained partly for two reasons. Since the global pandemic hit the world and everybody had to spend more time at home, people started to find new ways to spend time or get back to old hobbies. This can be seen in statistics as time spent on video gaming has increased globally 39%[12]. At the same time as Covid-19 introduced itself, the industrial world got hit by a global microchip shortage that made Graphics Processing Units (GPUs) and next-generation gaming consoles very hard to get. In these cases, cloud gaming offered a good alternative since it wasn't necessary to require a new con-

sole or powerful computer to play demanding games.

The idea behind cloud gaming service [22] is that games run on a remote server in data centers and content are streamed over the internet to the user's device. This essentially means that even light-power devices like phones, tablets, and older PCs can run new, high-demanded AAA games anywhere on the go. It does not only make gaming more accessible but also saves a lot of costs since the user does not need to update the hardware to meet the requirements needed for those AAA games.

This paper is organized as follows. Section 2 presents the related work and states the motivation behind this paper. Section 3 gives an overall history of cloud gaming. After this, Section 4 introduces the technical aspects of cloud gaming services. Section 5 reviews the biggest currently available services on the market, their providers, and gives a short comparison between them. And lastly, Section 6 provides concluding remarks

2 Related work

First cloud gaming services found their way to the market in the late 2000s. Since then, markets have changed tremendously, especially during the last couple of years and cloud gaming has gained a lot of popularity. This has resulted in many research papers that have studied cloud gaming from different perspectives. Cai et al. studies in their paper cloud gaming platforms and different optimization techniques [9]. Subjective quality assessment of cloud gaming was studied by Martini et al. [22]. Some work was provided on the future of cloud gaming [20] [10] and the possible use of cloud gaming in education [28]. Excellent network analysis and overall technical aspects of cloud gaming systems were surveyed by [26] [14] [27]. This paper tries to combine many aspects of current researches into one. The motivation behind this paper is to elaborate on the history of cloud gaming, give an overall picture of the GaaS mechanics, and lastly combine that with the review of available products currently on the market.

3 History

When talking about the history of cloud gaming, we can illustrate it as a long series of retries, rebranding and hits and misses. First-time cloud

gaming was introduced in 2000 by Finnish startup G-Cluster [23]. Their goal was to bring GaaS to the market for a wide audience. Unfortunately for G-Cluster, at the time the market was still not developed enough and the firm could not reach enough potential users. In 2010 G-Cluster started to target IPTV users.

In 2005 Crytek, a video game development studio behind popular Crysis games began its research on cloud gaming [21]. However, in 2007 Crytek concluded that global communication infrastructure was not ready for cloud gaming yet and put all research on hold.

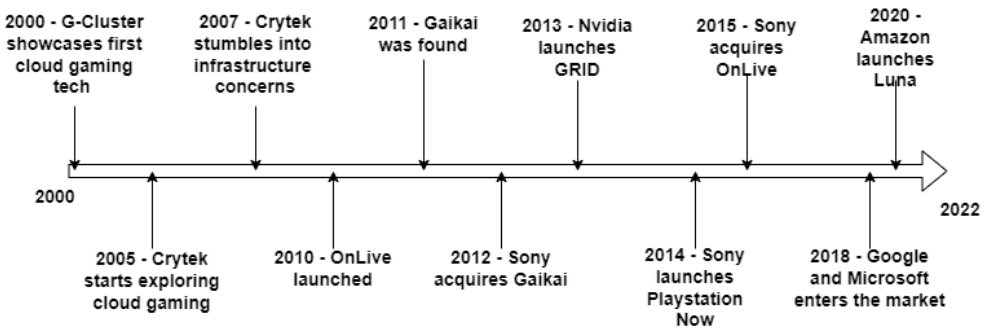


Figure 1. Simplified timeline from 2020 to 2022

In the late 2000s and early 2010s, the cloud gaming industry stepped into a new chapter. First OnLive and Gaikai were found. In 2010 OnLive [20] [19] launched their commercial cloud gaming service and a year later Gaikai followed. Both acquired support from big publishers, although only Gaikai was successful enough to be profitable. Both were later acquired by Sony Computer Entertainment, Gaikai in 2012 [9] and OnLive [4] in 2015. Sony introduced its cloud gaming platform Playstation Now in 2014. It was successful deal because in 2021 Playstation Now had 3.2 million active subscribers [24].

In 2013, Nvidia brought its cloud gaming service Nvidia Grid to the market which was later rebranded as formally known GeForce Now. Originally Grid aimed to partner with Gaikai to provide games as a service but Sony got Gaikai first [16]. Later Nvidia partnered with Agawi, Cloudunion, Cyber Cloud, G-cluster, Playcast, and Ubitus [15]. Now GeForce Now is one of the major service providers. In 2018 also Google and Microsoft entered the market and are branded now as Google Stadia and Microsoft xCloud. To follow the example of the other cloud providers, Amazon introduced its cloud gaming service in 2020 named Luna.

4 Technical aspects of cloud gaming

In this chapter, we jump into the technical part of the paper. First, we will look at the simplified architecture of the cloud gaming system and review its different parts and their requirements. After this, we will compare different network requirements and streaming qualities. In both sections, we will compare four of the most popular service providers Nvidia GeForce Now, Sony Playstation Now, Google Stadia, and Microsoft xCloud.

4.1 Components and cloud system

Let's review the technical aspect of the cloud gaming system. For many years cloud gaming was held back because of the lack of proper infrastructure. Gaming services require good bandwidth so that games themselves would stay playable in terms of performance. And of course, this is not the only challenge that GaaS encounter, as the whole cloud system is very complex from the technical point of view. Below in figure 1 is illustrated simplified version of a typical framework of a GaaS.

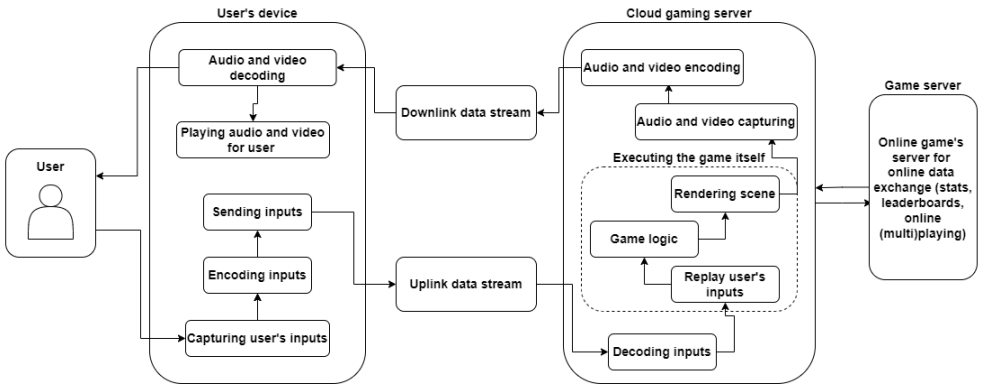


Figure 2. Simplified GaaS architecture [17] [11]

The main idea behind cloud gaming is that all game-related actions and logic, like scene rendering or storage, are executed on the cloud server-side. Therefore client-side only needs to handle capturing the user's inputs and sending them to the server and audio and video decoding when the server responds.

GaaS require two data streams [26]. The first data stream is transmitted to the cloud and contains the user's input events. The second data stream is from the cloud to the user's device and contains a rendered scene from the game. This data stream is the most problematic because it transmits many large data packets thus requiring good bandwidth and reliable

network protocol.

As the network connection is prone to delays and packet losses, cloud architecture needs to use reliable network protocols with certain properties. One of these is the absence of retransmission if packet loss happens during transmission. Typically both, uplink and downlink streams, uses User Datagram Protocol (UDP) based protocols [14]. UDP does not use acknowledgment, retransmission, or timeout and is therefore preferable for time-sensitive applications. It is better that the packet is dropped than it is delayed because of retransmission [18]. For uplink streams, which are used to deliver user inputs to the cloud, typical choices for network protocol are UDP, UDP based custom protocol, or Datagram Transport Layer Security (DTLS). Although DTLS can also be UDP based. Downlink streams which deliver audio and rendered video of the game, we can see protocols like Real-Time Streaming Protocol (RTSP), Real-time Transport Protocol (RTP), Web Real-Time Communication (WebRTC), and other UDP based custom protocols.

For video encoding, there are many codecs to choose from with different encoding speeds and compression efficiency. Despite this, as we can see in Table 1, H.264 codec, also referred to as MPEG-4, is the typical choice at the moment. Salah et al. [25] reported that applying H.264 video codec at the server-side boosted the whole process almost by 8.86%, which might explain the popularity of the codec. In 2021, H.264, was according to a Bitmovin annual global survey [8] used by 83% of the developers. H.254 has at the moment successor H.265 that provides better compression ratio than H.264 but codec increases latency which prevents major switch to H.265 [27].

Table 1. Used protocols for different actions [14] [27]

	Streaming	User's inputs	Video codec
GeForce Now	RTP	Custom (UDP)	H.264
Stadia	RTP, RTCP	DTLS	H.264
PS Now	Custom (UDP)	Custom (UDP)	H.264, VP9
Microsoft xCloud	Custom (UDP)	-	H.264

4.2 Streaming capabilities of different services

The beauty of cloud gaming is that users do not need to have powerful gaming systems to run and play new AAA games. But they do need to have a good and stable internet connection. Below in Table 2 are gathered minimum network requirements for different streaming qualities. We can see that you don't actually need that high bandwidth to play, as Playstation Now requires a minimum of 5Mb/s for 720p video quality. The highest requirements are for Stadia and GeForce Now. Stadia offers 4k locked at 60FPS for a minimum of 35 Mb/s. In comparison to Stadia's 4k, Netflix's 4k video stream consumes around 15 Mb/s [14]. For the same speed as Stadia, GeForce Now offers 1440p or 1600p video quality up to 120FPS. Nvidia is the only one that offers a frame rate up to 120FPS, the video quality of other services' is locked at 30FPS or 60FPS. The reason why quality differs from 1440p to 1600p is that some MacBooks have a resolution of 2560 x 1600px and the GeForce Now service adapts to it.

Table 2. Network requirements [1] [2] [5] [3]

	720p	1080p	1440p/1600p	4k
GeForce Now	15 Mb/s	25 Mb/s	35 Mb/s	-
Stadia	10 Mb/s	20 Mb/s	-	35 Mb/s
PlayStation Now	5 Mb/s	15 Mb/s	-	-
Microsoft xCloud	10 Mb/s	20 Mb/s	-	-

5 Overview of current market

Firstly in this chapter, we take a look at different service providers in the current industry. The chapter provides also a small glance at the popularity of these companies by presenting numbers of active subscribers. We also try to categories current services to give the reader a better overall picture of current markets and their potential.

5.1 Current services

Currently, gaming console and PC gaming markets are mostly divided by a few big companies and markets rarely see new entries. Gaming consoles are dominated by Sony, Microsoft and Nintendo, and vital PC parts for gaming like CPU and GPU are dominated by Intel, Nvidia and

AMD. Cloud gaming on the other hand has just started to develop. We see many same companies that dominate other gaming platforms providing also GaaS but the industry has also many smaller potential companies providing cloud gaming services and also many, in some way, surprising companies trying to enter the industry.

At the moment of writing this paper, there was found much over twenty companies that offer cloud gaming services. The biggest companies on the market in early 2022 are Nvidia GeForce Now, Sony Playstation Now, Google Stadia, Microsoft xCloud and Amazon Luna. Other popular services to name a few are Shadow by Blade, Boosteroid, Playkey, Vortex and Blacknut. All these companies offer the same idea, users play games through the company's servers via subscription.

For local streaming we have services like Steam Link, Nvidia Gamestream, and Moonlight. They offer a platform that allows users to stream games from their local machine to their other devices. This way if the user already has a powerful system and a game library, they don't need to pay extra money for a subscription and possibly lose their gaming data as well. To name some other not-so-traditional platforms there are companies for example like Paperspace, Parsec and Playkey.io. Paperspace and Parsec are similar to each other. They rent users cloud desktops that can be used, besides gaming, for heavy computation, rendering or just working remotely alone or collaborating with multiple people. Playkey.io is owned by the previously mentioned Playkey. Where just Playkey has data centers that render users gaming in tradiotional way, Playkey.io has taken the idea behind mining cryptocurrency and applied it to cloud gaming. So the idea behind Playkey.io is that computing is decentralized and miners and others having powerful systems can lend their computing power to players. Lenders then get paid for this in Playkey Tokens or PKT's shorter which is a cryptocurrency made by Playkey. These PKT's can then be used like normal cryptocurrencies and exchange for other cryptocurrencies or money.

5.2 Userbase

To put the popularity of cloud gaming into numbers let's take a look at the size of the active userbases. To take a look at previously presented companies let's compare the biggest players in the market. From big to small, Microsoft xCloud has over 18 million active users [29], EA Play has over 13 million [13], GeForce Now has over 12 million [6] and Playstation

Now has over 3.2 million active users [7]. Alone these four companies have nearly 50 million active users which creates a huge market. To put this into perspective, according to Statista, cloud gaming market value was in 2021 worldwide approximately 1.5 billion US dollar [13]. Statista also forecasts market value to grow over 6.5 billion by 2024.

6 Conclusion

In this paper, we have reviewed the history of cloud gaming, showed the technical side of GaaS systems, and examined current cloud gaming services on the market. On the technical side, it was surprising how similar protocols were used by currently the most popular service providers. For delivering user's inputs to the cloud and for delivering video and audio streams to a user were used basically just different variations of UDP protocol. Also, for video encoding same H.264 codec was used by all four of the reviewed services. It was positive that it is possible to game as low as with 5 Mb/s bandwidth. Although the highest video streams at 1440p 120FPS or 4k 60FPS required a minimum of 35 Mb/s bandwidth. This could be, especially in bigger families where there are many network users, difficult to achieve.

In the current market overview, it was found over twenty cloud gaming providers globally. This tells about the popularity of cloud gaming and it will be interesting to see how this sector develops in the future.

References

[1] Geforce now -pilvipelaamisen järjestelmävaatimukset.

[2] Kaistanleveys, datankäyttö ja striimauslaatu - stadianbsp;ohjeet.

[3] Network requirements microsoft xbox cloud gaming.

[4] Onlive official web page.

[5] Ps now'n käytön aloittaminen.

[6] John Ballard. Here's why nvidia's cloud gaming service is the real deal, Oct 2021.

[7] Sammy Barker. Ps now has a respectable 3.2 million subscribers, May 2021.

[8] Bitmovin. Bitmovin's 5th annual video developer report 2021.

[9] Wei Cai, Ryan Shea, Chun-Ying Huang, Kuan-Ta Chen, Jiangchuan Liu, and Cheng-Hsin Hsu. A survey on cloud gaming: Future of computer games. *IEEE Access*, 4:1–1, 01 2016.

[10] Wei Cai, Ryan Shea, Chun-Ying Huang, Kuan-Ta Chen, Jiangchuan Liu, Victor C. M. Leung, and Cheng-Hsin Hsu. The future of cloud gaming [point of view]. *Proceedings of the IEEE*, 104(4):687–691, 2016.

[11] Wei Cai, Ryan Shea, Chun-Ying Huang, Kuan-Ta Chen, Jiangchuan Liu, Victor C. M. Leung, and Cheng-Hsin Hsu. The future of cloud gaming [point of view]. *Proceedings of the IEEE*, 104(4):687–691, 2016.

[12] Clement. Covid-19 impact on the gaming industry worldwide - statistics facts. 2021.

[13] J. Clement. Global cloud gaming market size 2024, Oct 2021.

[14] Andrea Di Domenico, Gianluca Perna, Martino Trevisan, Luca Vassio, and Danilo Giordano. A network analysis on cloud gaming: Stadia, geforce now and psnow. *Network*, 1(3):247–260, 2021.

[15] Ben Gilbert. Nvidia details the grid, a card built for powering cloud computing, Jan 2013.

[16] Sean Hollister. Nvidia announces geforce grid: Cloud gaming direct from a gpu, with games bynbsp;gaikai, May 2012.

[17] Chun-Ying Huang, Cheng-Hsin Hsu, Yu-Chun Chang, and Kuan-Ta Chen. Gaminganywhere: An open cloud gaming system. In *Proceedings of the 4th ACM multimedia systems conference*, pages 36–47, 2013.

[18] James F. Kurose and Keith W. Ross. *Computer networking: A top-down approach*. Pearson, 2013.

[19] JP Mangalindan. Cloud gaming's history of false starts and promising reboots, Oct 2020.

[20] Bryce Mariano and Simon G. M. Koo. Is cloud gaming the future of the gaming industry? In *2015 Seventh International Conference on Ubiquitous and Future Networks*, pages 969–972, 2015.

[21] Matt Martin. Crysis core. *GamesIndustry.biz*, Apr 2009.

[22] Maria Martini, Abdul Wahab, Nafi Ahmad, and John Schormans. Subjective quality assessment for cloud gaming. *J*, 4:404–419, 08 2021.

[23] Arto Ojala and Pasi Tyrväinen. Developing cloud business models: A case study on cloud gaming. *IEEE Software*, 28:42–47, 07 2011.

[24] Jim Ryanv. Sony game network services segment, 2021.

[25] Mosa Salah, Ahmad A. Mazhar, and Manar Mizher. Optimization of video cloud gaming using fast hevc video compression technique. *International Journal of Advances in Soft Computing and its Applications*, 13(3):249–265, 2021.

[26] Steven Schmidt. Assessing the quality of experience of cloud gaming services. 2021.

[27] Ivan Slivar. *Quality of experience driven video encoding adaptation strategies for cloud gaming under network constraints*. PhD thesis, University of Zagreb. Faculty of Electrical Engineering and Computing . . . , 2021.

[28] Mirko Suznjevic and Maja Homen. *Use of Cloud Gaming in Education*. 02 2020.

[29] Tom Warren. Xbox game pass subscribers hit 18 million, Jan 2021.

Visualizing firewall configuration anomalies

Joose Lehtinen
joose.lehtinen@aalto.fi

Tutor: Tuomas Aura

Abstract

Firewall configuration anomalies are often signs of configuration errors. Configuration errors may allow a third party access to devices they are not authorized to access and may cause digital or in some cases even physical security risks. This paper presents an algorithm for visualizing firewall configuration anomalies. The algorithm is based on the idea of dividing the space of possible network packets into segments based on the rules the packets on that hypercube match and visualizing those segments. The algorithm is particularly suited for finding shadowed, correlated and redundant rules. The visualization can also be used for further algorithmic analysis.

KEYWORDS: firewall, configuration anomalies

1 Introduction

Firewall configurations are used to protect private networks from external attacks. They allow or block network traffic between networks by comparing them to a set of rules. These rules may relate to the transport-layer protocol, IP addresses and ports of the communicating parties [2]. Some

services in the network, for example NetBIOS services, may be insecure and therefore should not be accessible from outside the network [5]. Access to one vulnerable service can often be used to attack other services in the same network. Because of this, even a single vulnerability in the firewall configuration can compromise the entire network.

Anomalies and vulnerabilities are rarely intentional and are both easy to cause by mistake. Because of this, internal inconsistencies and other anomalies may be signs of mistakes in the configuration and may indicate the existence of vulnerabilities in the firewall configuration. For example, a rule overriding another rule might have been intended to affect a different IP address range.

This paper implements an algorithm to help find and visualize configuration anomalies in firewall policies.

The paper first explains different types of anomalies presented by prior research. Then the paper presents an algorithm for visualizing firewall policies and explains how it helps at finding anomalies. The implementation is inspired by the algorithm presented by Abbes et al. [1]. The result of the algorithm is a matrix, from which anomalies can be found either manually or with simple algorithms. The implementation is suitable for visualizing shadowed, correlated and redundant rules.

Earlier attempts at trying to find and resolve configuration anomalies include use of decision trees [1, 3, 4]. In contrast, we break the packet space into a structure resembling a list. Each list entry may represent arbitrarily complex subspace of packet space. This should give us advantage in terms of anomalies we can detect, but will cost considerably more computational resources.

Section 2 explains the types of anomalies in firewall configurations. Section 3 introduces an algorithm for visualizing anomalies in firewall configurations. Section 4 evaluates the performance of the algorithm in terms of space complexity and the types of anomalies it can detect. Section 5 concludes the paper.

2 Firewall anomalies

Yuan et al. [6] detail different types of firewall anomalies. They explain six types of firewall anomalies that apply to systems with a single firewall: policy violations, shadowing, generalization, correlation, redundancy and verbosity. We will look at each type in bit more detail and also look at

some examples.

Policy Violations Policy violations occur when the firewall does not follow the higher-level policies it is supposed to follow. Policy violations cannot be detected without some knowledge from outside the firewall itself.

Shadowing A rule is shadowed if preceding rules already match all the packets that the shadowed rule would match, but assign a different action. A shadowed rule is often a sign of misconfiguration because shadowed rules are never used, and usually firewall rules are added with the intention that all rules are significant. Figure 1 is a simple example of a policy where a rule is shadowed. The second rule is never used because any packet matching the rule would have already matched the first rule.

Figure 1. Example of a policy with a shadowed rule

rule description	accepted?			
destination port in range 0-1024 denied	0	0	1	1
destination port 443 (HTTPS) accepted	1	0	0	1
any packet denied	0	1	1	1
is packet accepted by policy?		0	0	0

Generalization A rule is generalization of another rule if a preceding rule affects a subset of packets matched by the generalizing rule but the rules define different actions. A generalization is not necessarily a misconfiguration. In fact, generalizations are a common practice in firewall configurations [6]. In figure 2, the second rule is generalization of the first rule. This is an example of a policy where a generalization is not a misconfiguration.

Figure 2. Example of a policy with generalization

rule description	accepted?		
destination port 443 (HTTPS) accepted	1	0	1
any packet denied	0	1	1
is packet accepted by policy?		0	1

Correlation Two rules correlate if their packet spaces intersect but they specify a different action. Similar to generalizations, correlations are a common practice in firewall configurations [6]. In figure 3, the first and second rules correlate. It is not obvious what the intended behaviour of the policy is, so we can not tell whether the policy is misconfigured or not.

Figure 3. Example of a policy with correlation

rule description	accepted?				
destination port 443 (HTTPS) allowed	1	0	0	1	1
source port in range 0-1024 denied	0	0	1	0	1
any packet denied	0	1	1	1	1
is packet accepted by policy?		0	0	1	1

Redundancy A rule is redundant if removing it would not alter the behavior of the firewall. Removing redundant rules may help reduce packet processing time and memory consumption [6]. In figure 4, the first rule is redundant, because the second rule would match and accept any packet matched by the first rule. This might be intended to improve the readability of the policy, but the first rule could also be removed.

Figure 4. Example of a policy with redundancy

rule description	accepted?			
destination port 443 (HTTPS) allowed	1	0	0	1
destination port in range 0-1024 allowed	1	0	1	1
any packet denied	0	1	1	1
is packet accepted by policy?		0	1	1

Verbosity A firewall policy is verbose if it could be expressed with smaller number of rules. Verbosity is not necessarily a misconfiguration, but less verbose policies generally require less memory and processing power. Verbose policies may be created in practice when system administrators iteratively add new rules [6].

Figure 5. Example of a policy with verbosity

rule description	accepted?			
destination port 443 (HTTPS) allowed	1	0	0	1
destination port 444 (SNPP) allowed	1	0	1	0
any packet denied	0	1	1	1
is packet accepted by policy?		0	1	1

If anomalies are found, the network administrator should try to understand what caused that anomaly. For example, if a rule is redundant because of a later, more general rule, it may have been meant as an exception and have been labeled wrong. If the mistake that led to someone writing an anomalous rule is not found, the underlying mistake might persist in the configuration even if an anomaly detection algorithm finds no further anomalies.

3 Anomaly visualization

This section discusses the implementation of the algorithm. The algorithm is largely based on ideas presented by Hu et al. [2], but the details of the implementation are different. Hu et al. use binary decision diagrams to represent the packet space segments, while we use ranges of IP addresses and ports. The output of the algorithms should be equivalent, despite the different implementations.

Subsection 3.1 discusses how the packet space is represented and the details of operations on that packet space that we need for the algorithm. Subsection 3.2 discusses the high level algorithm implemented using components discussed in subsection 3.1.

3.1 Representation of packet space

Each packet in the network can be represented as a vector. The elements of the vector may represent IP addresses, ports and a protocol. We represent each packet as a vector with length of 6. The vector's elements are source and destination ipv4 addresses and ports, the used protocol and the network adapter. Network adapter is not commonly included in the firewall policy in the literature.

Figure 6 shows three examples of packets represented in the vector space.

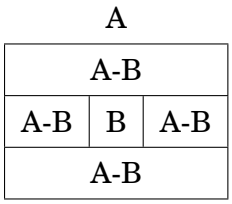
Figure 6. Examples of packets represented as vectors

source IP	destination IP	source port	destination port	network protocol	network interface
1.1.1.1	2.2.2.2	22	22	0 (TCP)	0
1.1.1.1	3.3.3.3	40006	443	0 (TCP)	0
3.3.3.3	1.1.1.1	443	40006	0 (TCP)	1

Firewall rules can be thought of as boxes or hyperrectangles in this vector space, along with a binary value indicating whether the rule allows or blocks traffic. For purposes of this paper, we define packet space segment as lists of these hyperrectangles. The algorithm we implement requires addition and subtraction for packet space segment. Addition simply append one of the lists into the other. Subtraction loops over each hypercube in the subtrahend and removes them from the minuend in sequence. Subtracting individual hypercube is done by looping over the hypercube list of the minuend and subtracting it from each hypercube separately. These differences are then collected to a new list, which is returned as the difference. Hypercubes with non-positive volume are not added to this list.

The difference between individual hypercubes is calculated by going over each axis, removing the parts of the minuend that do not overlap with the subtrahend along that axis from the minuend, and adding them to the difference. If the hypercubes don not intersect, the difference is the minuend and there is no need to split it into smaller parts. The difference of two hypercubes can, in the worst case, consist of 2*d hypercubes, where d is the number of dimensions of the packet space. The figure 7 shows how hypercube difference is calculated in two dimensions. B (the small rectangle in the middle) is subtracted from A (the large rectangle). The resulting difference consists of 4 rectangles.

Figure 7. Visualization of hypercube difference in two dimensions



3.2 High level algorithm

The goal of the algorithm is to divide the entire packet space into non-overlapping segments, each described by unique mapping from the rules in the firewall policy to boolean values indicating whether the rule would be applied to this segment or not. This information can be used to generate a matrix, which can visualize which segments are affected by which rules.

When the algorithm is initialized, it creates a list of packet space segments. A segment containing the entire packet space is added to the segment list. This segment is associated with an empty list denoting, which rules match to this segment.

After the algorithm is initialized, all firewall rules are applied sequentially. My implementation goes from the last rule to the first, but having done that, I would recommend going from first rule to the last, because that is probably easier to implement. When a new rule is applied, it is applied to each segment separately. Each time a new rule is applied, a new list of segment is constructed from the old list. First the packet space of the rule is subtracted from the segment, forming a segment that does not fit the rule. Then this new segment is subtracted from the original segment, forming a new segment that is the intersection of the original segment and the rule. Then these segments are added to the the new list of segments, while the old segment is removed form the old list. These segments are also associated with a list denoting which rules match to this segment. The list is constructed from the old list associated with the old packet space segment, with information about the applicability of the currently handled rule added to it.

The resulting list contains mutually exclusive segments of the packet space associated with the information about which rules match each segment. The individual hypercubes are also mutually exclusive, but that is not required for the algorithm to work. The results of the algorithm can be viewed in a form of a matrix. The matrix can either be used to manually find configuration anomalies as demonstrated in section 4.1 or as basis for further algorithmic analysis.

4 Performance evaluation

This section evaluates the algorithm presented in the previous section. First we describe what types of anomalies we can detect using the algorithm. Then, subsection 4.2 evaluates computational complexity of the algorithm. Subsection 4.1 shows an example of how the algorithm could be used to find anomalies in a firewall policy.

In this paper, we do not try to find policy violations, because that would require some representation for the policy, which we do not have. Our approach could be used to find some cases of verbosity. A pair of rules with the same action that would combine to a single hypercube could be found fairly easily. However, most types of verbosity would be more difficult to find. For all we know, this approach could be useless for finding all but the most simple cases of verbosity. Finding and visualizing verbosity is therefore outside of the scope of this paper. Shadowed rules could be detected fairly easily algorithmically. We would only have to find rules before the shadowed rule that match its packet space and assign a different action. Redundant rules can also be found fairly easily algorithmically by removing individual rows from the firewall policy. If the action assigned to a packet space segment does not change as a result of removing a rule, that rule is redundant. This does not require running the algorithm again, because the assigned actions can be calculated from the visualized matrix. Correlations can also be found algorithmically using the visualization by comparing two of the rules. If the rules' packet spaces intersect and the rules assign different actions, they are by definition correlated.

4.1 Analyzing sample firewall policy

Figure 8 shows the packet space visualization of a hypothetical firewall policy. We will analyze this policy manually to demonstrate how the table could be used to find anomalies. From the table we can see, that the fourth rule is redundant, as all packets it matches have already been matched by the third rule. Less obviously the second rule is also redundant, because any packet it matches will also be matched by the last rule, which will assign the same action. The first and third rules are correlated. This anomaly could potentially be a misconfiguration, but we should consult the firewall policy to know for sure.

Figure 8. Packet space segments of example policy

rule description	accepted?							
packet to office printer	0	0	0	1	1	0	1	0
packet from outside to non-standard port	0	0	0	0	0	1	1	0
packet to port 443 (HTTPS)	1	0	1	0	1	0	0	1
packet to the web server's port 443	1	0	0	0	0	0	0	1
any packet	0	1	1	1	1	1	1	1
is packet accepted by the policy?		0	1	0	1	0	0	1

4.2 Evaluating space complexity

Each new rule may divide the packet space along each axis to up to 3 segments, creating 2 new segments. There are 5 dimensions along which we have a reasonably large number of possible values: IP addresses, ports and protocol. Therefore, it stands to reason that $(2n + 1)^5$ is an upper bound for the space complexity of the algorithm, where n is the number of rules in the firewall policy.

We create malicious inputs trying to show that the upper bound is tight. In the malicious policy, each rule defines IP address ranges and port ranges that include all previous ranges and the endpoints are not endpoints of any of the previous ranges. Each rule defines a protocol range, which is included in all previous ranges and the enpoints are not endpoints of any previous of any of the previous ranges. The algorithm seems to divide such malicious policy to $2/15(2n^5+5n^4+20n^3+25n^2+23n)$ hypercubes, where n is the number of rules. As a practical example, a malicious policy with just 12 rules generates 85304 hypercubes.

Because we have both an upper bound and an example case as 5th degree polynomials, the worst case space complexity of the algorithm is a 5th degree polynomial. Generally, the space complexity should be a polynomial of degree d, where d is the number of rule dimensions. If we assume there are only small number of possible protocols the space complexity would be 4th degree polynomial.

5 Conclusion

This paper implements an algorithm to help visualize firewall policies. The visualization is intended to help find configuration anomalies. The

paper then discusses the capabilities and limitations of the algorithm. We demonstrate that the algorithm's results can be used for manual analysis, although its results could be better used as a starting point for other algorithms. The implementation is particularly suitable for visualizing shadowed, correlated and redundant rules. The algorithm runs in polynomial space.

References

[1] Tarek Abbes, Adel Bouhoula, and Michaël Rusinowitch. Detection of firewall configuration errors with updatable tree. *International Journal of Information Security*, 15(3):301–317, 2016.

[2] Hongxin Hu, Gail-Joon Ahn, and Ketan Kulkarni. Detecting and resolving firewall policy anomalies. *IEEE Transactions on dependable and secure computing*, 9(3):318–331, 2012.

[3] Amina Saâdaoui, Nihel Ben Youssef Ben Souayeh, and Adel Bouhoula. Fare: Fdd-based firewall anomalies resolution tool. *Journal of Computational Science*, 23:181–191, 2017.

[4] Sumeet Singh, Florin Baboescu, George Varghese, and Jia Wang. Packet classification using multidimensional cutting. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 213–224, 2003.

[5] Avishai Wool. Firewall configuration errors revisited. *arXiv preprint arXiv:0911.1240*, 2009.

[6] Lihua Yuan, Hao Chen, Jianning Mai, Chen-Nee Chuah, Zhendong Su, and Prasant Mohapatra. Fireman: A toolkit for firewall modeling and analysis. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 15–pp. IEEE, 2006.