

Tutorial on HTTP2 and Wireshark

Chao Zhu, chao.1.zhu@aalto.fi

This tutorial covers

- HTTP/2
 - A brief introduction on HTTP/2
 - How to implement your own HTTP/2 client & server
- Wireshark
 - How to use Wireshark to monitor network packets
 - How to deal with the encrypted packets

HTTP/2

- The next version of HTTP/1.x
- Documented in [RFC 7540](#)
- Features
 - Binary protocol
 - [Multiplexing](#)
 - Header compression
 - [Server push](#)
 - Request prioritization
- HTTP/2 outperforms HTTP/1.x in many aspects[1][2], read the papers for details
- [Demo page](#)



[1] HTTP/1.1 pipelining vs HTTP2 in-the-clear: Performance comparison, NOTERE ' 16

[2] How Speedy is SPDY? NSDI ' 14

HTTP/2 Clients & Servers

- Java
 - Netty, Tomcat
- C/C++
 - Nginx, H2O, Nhttp2
- Python
 - ***Hyper-h2***
- More
 - <https://github.com/http2/http2-spec/wiki/Implementations>

Hyper-h2 Installation (For Linux)

- Install Python 3
 - `sudo apt-get install python3.6`
- Install Hyper-h2
 - `pip install h2`
- Install Hyper
 - `pip install hyper`

A Simple Hyper-h2 Server

- A simple server implementation
- Demo
 - Check HTTP/2 request 'special headers'

HTTP/2 in Chrome

- By default, Chrome enables HTTP/2
- Openssl and Libapr required
 - For macOS, *brew install apr-util openssl*
 - For Ubuntu, *apt-get install libssl-dev libapr1-dev*
- Demo
 - Clone code from Github
 - Visit server from chrome
- Developer panel

Implement Client & Server

- Notes:
- Free to use any language (C++, Java and Python)
- Hyper-h2 does not provide I/O
- You need to implement your own client and server (not just browser)
- Just focus on file sending, receiving and packet decrypt (we do not care about GUI and rendering)

Wireshark

- A packet analyzer that captures data packets flowing over the network
- Installation
 - [Downloads](#)

Wireshark

Filter



Search your computer

tcp

No.	Time	Source	Destination	Protocol	Length	Info
190	36.287086017	10.100.20.212	52.205.234.159	TLSv1.2	142	Application Data
191	36.287282640	10.100.20.212	52.205.234.159	TLSv1.2	1253	Application Data
192	36.289135691	10.100.20.212	52.205.234.159	TLSv1.2	373	Application Data
193	36.440302051	52.205.234.159	10.100.20.212	TCP	66	443 → 52624 [ACK] Seq=68...
194	36.440953389	52.205.234.159	10.100.20.212	TLSv1.2	260	Application Data
195	36.441063236	10.100.20.212	52.205.234.159	TCP	66	52624 → 443 [ACK] Seq=74...
196	36.441147759	52.205.234.159	10.100.20.212	TLSv1.2	260	Application Data
197	36.441174370	10.100.20.212	52.205.234.159	TCP	66	52624 → 443 [ACK] Seq=74...
198	37.589343216	162.125.18.133	10.100.20.212	TLSv1.2	323	Application Data
199	37.598998979	10.100.20.212	162.125.18.133	TLSv1.2	549	Application Data
200	37.798143160	162.125.18.133	10.100.20.212	TCP	66	443 → 37412 [ACK] Seq=25...
201	37.798192153	10.100.20.212	162.125.18.133	TLSv1.2	651	Application Data
202	38.007001832	162.125.18.133	10.100.20.212	TCP	66	443 → 37412 [ACK] Seq=25...
203	39.935831780	10.100.20.212	136.243.37.214	TCP	66	51960 → 443 [ACK] Seq=1 ...
204	39.963993895	136.243.37.214	10.100.20.212	TCP	66	[TCP ACKed unseen segmen...

Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0

- Ethernet II, Src: IntelCor_d1:48:63 (00:e1:8c:d1:48:63), Dst: Cisco_b6:bf:7a (bc:16:65:b6:bf:7a)
- Internet Protocol Version 4, Src: 10.100.20.212, Dst: 198.252.206.25
- Transmission Control Protocol, Src Port: 51660, Dst Port: 443, Seq: 1, Ack: 1, Len: 0

```
0000  bc 16 65 b6 bf 7a 00 e1 8c d1 48 63 08 00 45 00  ..e.z...Hc..E.
0010  00 34 01 0b 40 00 40 06 85 6b 0a 64 14 d4 c6 fc  .4.@.@.k.d...
0020  ce 19 c9 cc 01 bb 3b ce 92 30 43 74 f6 68 80 10  .....;..Oct:h..
0030  00 f5 aa 6f 00 00 01 01 08 0a 51 59 2b 06 10 28  ...o.....QY+..(
0040  b7 1f
```

Bytes 58-61: Timestamp value (tcp.options.timestamp.tsval) Packets: 204 · Displayed: 188 (92.2%) Profile: Default

Packets



Packet Detail



Wireshark Demo

- Wireshark Tutorial for Beginners,
<https://www.youtube.com/watch?v=TkCSr30UojM>

Hints on Assignments

- Assignment 1, implement server and client
 - Read the protocol specification
 - Design test cases for evaluating the protocol design
 - Access a webpage
 - Post an image
 - Post tags (both 'POST' and 'PUT' are okay)
 - Receive notification from server

Hints on Assignments

- Assignment 2, test HTTP/2 new features
 - Select 2 out of the 4 new features
 - Flow control
 - `local_flow_control_window(stream_id)`
 - `increment_flow_control_window(increment, stream_id=None)`
 - Stream priority
 - `prioritize(self, stream_id, weight=None, depends_on=None, exclusive=None)`
 - Multiplexing
 - Sending multiple requests over a communications link without response
 - Server push
 - Server pushes a resource directly to the client without the client asking for the resource.

Other Hints

- HTTPS packets decryption with Wireshark
- Netty-based Implementations

Reference

M. Jiang, X. Luo, T. Miu, S. Hu and W. Rao, "[Are HTTP/2 Servers Ready Yet?](#)," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, 2017, pp. 1661-1671. doi: 10.1109/ICDCS.2017.279