Aalto University

# Network Security:
# Firewall: stateful filtering

Tuomas Aura

CS-E4300 Network security
Aalto University

# Stateful packet filter example: TCP

Stateful filter that passes only outbound connections:

| Input interface | Protocol | Src IP | Src port | Dst IP | Dst port | Flags | Other | Action | Comment |
|---|---|---|---|---|---|---|---|---|---|
| lan | TCP | 1.2.3.0/24 | * | * | 80 | | | Pass, create state | Outbound HTTP requests |
| wan | TCP | * | 80 | 1.2.3.0/24 | * | | state | Pass | Inbound HTTP responses |
| * | * | * | * | * | * | | | Drop | Default rule |

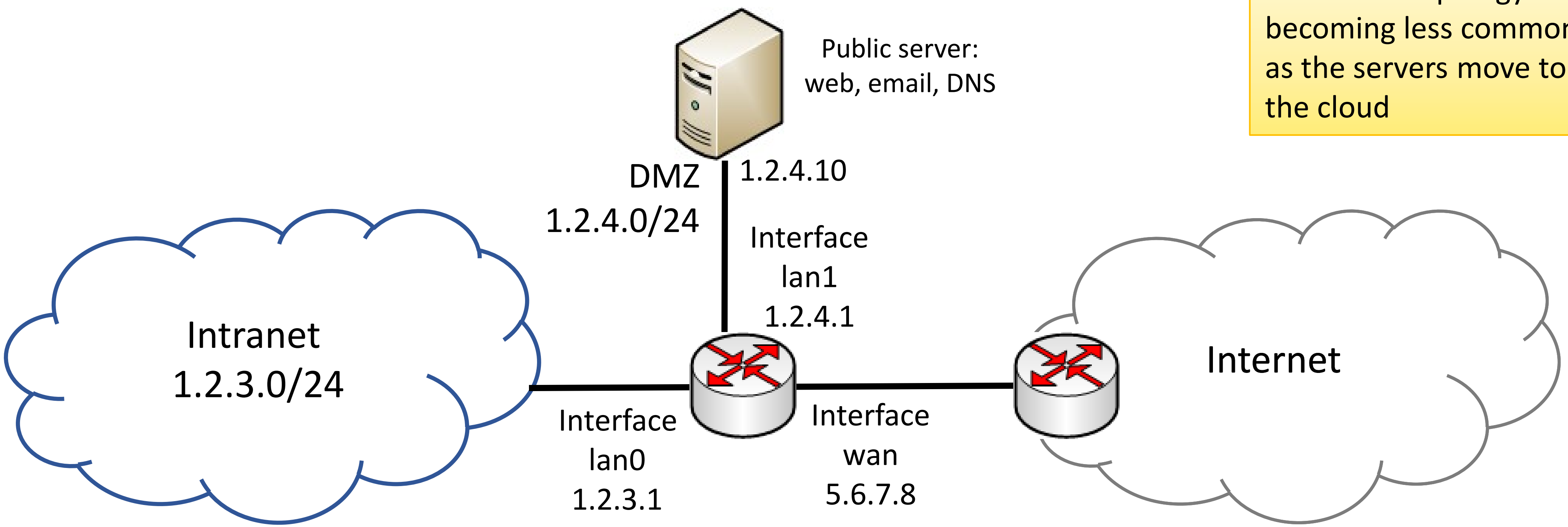
- Good firewalls only create a small pinhole: the state is stored in a connection table that remembers the protocol, local IP address, local port, remote IP address, remote port
- Some firewalls might only remember the rule (pair) that was matched, but that is less secure (why?)

# Dynamic firewall

- Stateful packet inspection (SPI) detects packets that start new connections or belong to an old one
- Outbound TCP or UDP packet creates a pinhole for inbound packets of the same connection
  - TCP pinhole is closed when connection closes or after timeout; UDP after timeout, e.g., 30 min
- Firewalls may support stateless filtering of additional protocols:
  - ICMP errors that match previous outbound packets
  - FTP and X Windows open TCP connections in reverse direction
- Stateful filtering may significantly lower firewall/router throughput and increase latency if the processing is done in software

# Network topology with DMZ

- Services accessible from the Internet are isolated to a demilitarized zone (DMZ), i.e., in a separate subnetwork
- The DMZ is typically a virtual LAN (VLAN) instead of a physical network



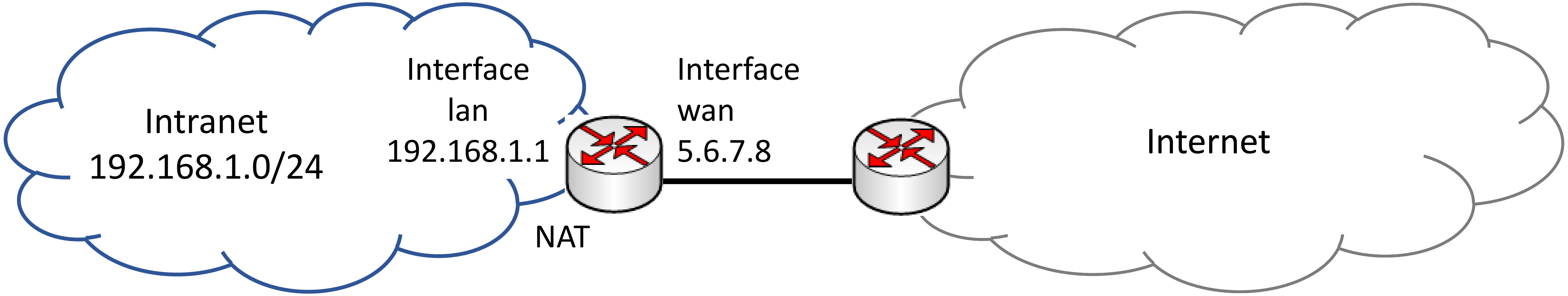
Note: This topology is becoming less common as the servers move to the cloud

| Input if | Protocol | Src IP | Port | Dst IP | Port | Other | Action | Comment |
|---|---|---|---|---|---|---|---|---|
| wan, lan1 | * | 1.2.3.0/24 | * | * | * | | Drop | Anti-spoofing (LAN) |
| wan, lan0 | * | 1.2.4.0/24 | * | * | * | | Drop | Anti-spoofing  (DMZ) |
| * | * | {1.2.3.1, 1.2.4.1, 5.6.7.8} | * | * | * | | Drop | Anti-spoofing (router addresses) |
| wan, lan0 | UDP | * | * | 1.2.4.10 | 53 | | Pass, create state | DNS query to local server |
| lan1 | UDP | 1.2.4.10 | 53 | * | * | State | Pass | DNS response from local server |
| lan1 | UDP | 1.2.4.10 | * | * | 53 | | Pass, create state | DNS query to ISP |
| wan | UDP | * | 53 | 1.2.4.10 | | State | Pass | DNS response from ISP |
| wan | TCP | * | * | 1.2.4.10 | 22,25,80,443 | | Pass | Server access from Internet |
| lan0 | TCP | 1.2.3.0/24 | * | 1.2.4.10 | 22,25,80,443 | | Pass, create state | Server access from intranet |
| lan1 | TCP | 1.2.4.10 | 22,25,80,443 | 1.2.3.0/24 | * | State | Pass | Responses |
| lan0 | * | 1.2.3.0/24 | * | 1.2.4.0/24 | * | | Drop | Unnecessary LAN-DMZ traffic |
| lan1 | * | 1.2.4.0/24 | * | 1.2.3.0/24 | * | | Drop | Unnecessary LAN-DMZ traffic |
| lan0 | * | 1.2.3.0/24 | * | * | * | | Pass, create state | Outbound to Internet |
| wan | * | * | * | * | * | State | Pass | Responses from Internet |
| lan0 | TCP | 1.2.3.0/24 | * | 1.2.3.1 | 22 | | Pass, create state | Router management |
| local | TCP | 1.2.3.1 | 22 | 1.2.3.0/24 | * | State | Pass | Router management |
| * | * | * | * | * | * | | Drop | Default rule |

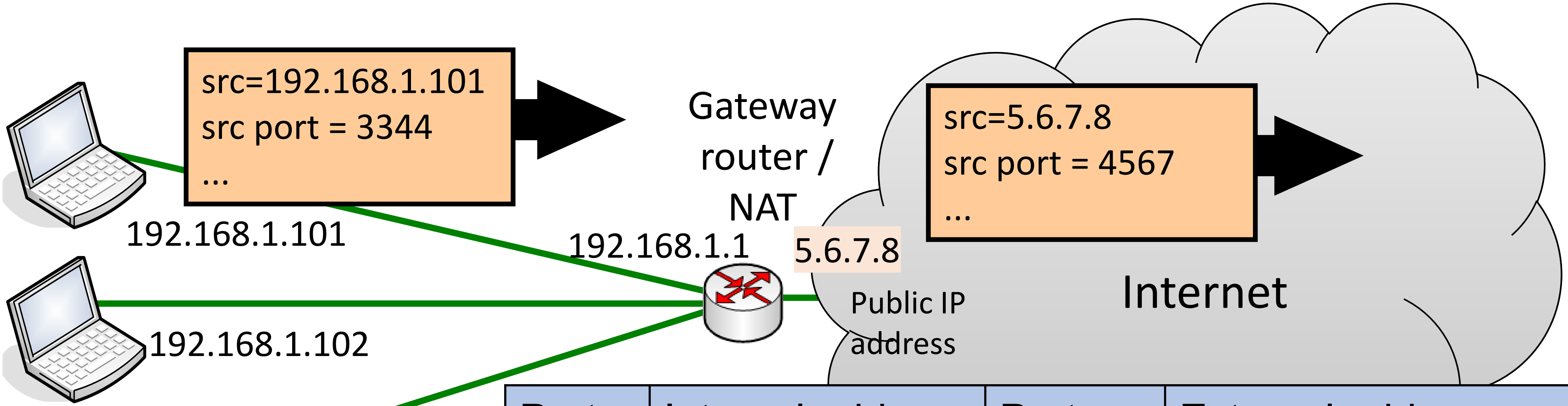
Is this correct?  What happens if the rules are in wrong order What kind of state is stored in each case?

# NAT

- Network address translation (NAT) is a mechanisms for sharing one IPv4 address between multiple hosts
- Hosts in intranet use a private address space
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
  - 100.64.0.0/10 for carrier-grade NAT (e.g., ISPs and mobile operators)
- Hosts behind NAT can only act as TCP or UDP clients, not as servers
- Intranet
  192.168.33.0/24



# NAT



src=192.168.1.101
src port = 3344
…

Gateway router / NAT

192.168.1.101

192.168.1.102

192.168.1.103

192.168.1.1

5.6.7.8

Public IP address

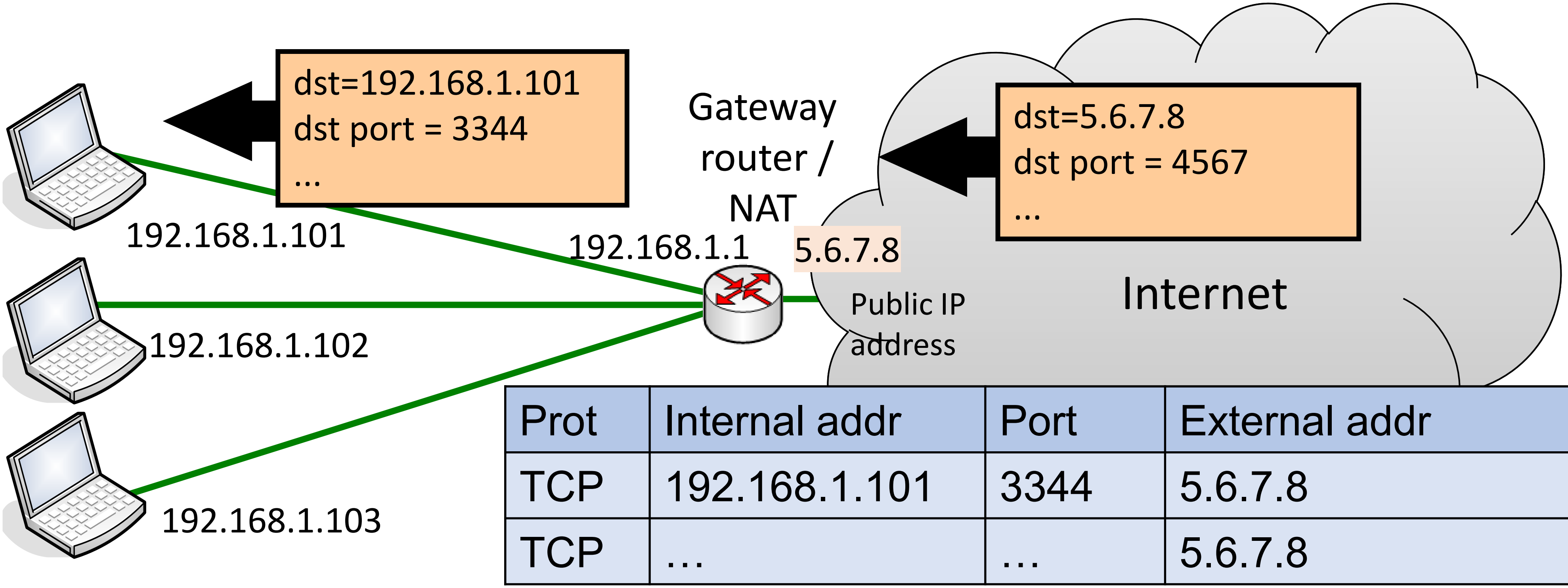
src=5.6.7.8
src port = 4567
…

Internet

| Prot | Internal addr | Port | External addr | Port |
|---|---|---|---|---|
| TCP | 192.168.1.101 | 3344 | 5.6.7.8 | 4567 |
| TCP | … | … | 5.6.7.8 | … |

← Private IP addresses → ← Internet addresses →

# NAT



dst=192.168.1.101
dst port = 3344
…

Gateway
router /
NAT

dst=5.6.7.8
dst port = 4567
…

192.168.1.101

192.168.1.1

5.6.7.8

Public IP
address

Internet

192.168.1.102

192.168.1.103

| Prot | Internal addr | Port | External addr | Port |
|---|---|---|---|---|
| TCP | 192.168.1.101 | 3344 | 5.6.7.8 | 4567 |
| TCP | … | … | 5.6.7.8 | … |

← Private IP addresses → ← Internet addresses →

# NAT as a firewall

- NAT maps internal <private IP addr, port> pairs to external <public IP addr, port> pairs and back
  - NAT creates the mapping based on an outbound packet
    → a node on the intranet must initiate each connection
    → NAT acts as a dynamic firewall
- NAT may remember additional connection parameters:
  - Remote IP address and port, TCP protocol state
  - The more connection parameters it remembers and filters, the more like a stateful firewall it becomes
- The previous example is a full cone NAT: does not even remember remote address or port → worst for security, best for NAT traversal

# Implementation terminology

- Endpoints:
  - – Source – destination,
  - – Local – remote
  - – Inbound – outbound
- Direction of connection:
  - – Existing connections = state exists
  - – Mirror rule = pass packets also in the other direction

# Port forwarding and UPnP

- Port forwarding: firewall administrator creates a permanent NAT table entry or firewall hole
  - Inbound connections to the firewall are routed to a specific host and port in the intranet
  - Internet client can connect to a servers behind the NAT or firewall, e.g., Minecraft server or personal media server
- Universal plug and play (UPnP):
  - Discovery protocol with which intranet hosts can request the firewall to set up port forwarding
  - Security-aware people generally disable UPnP in their router

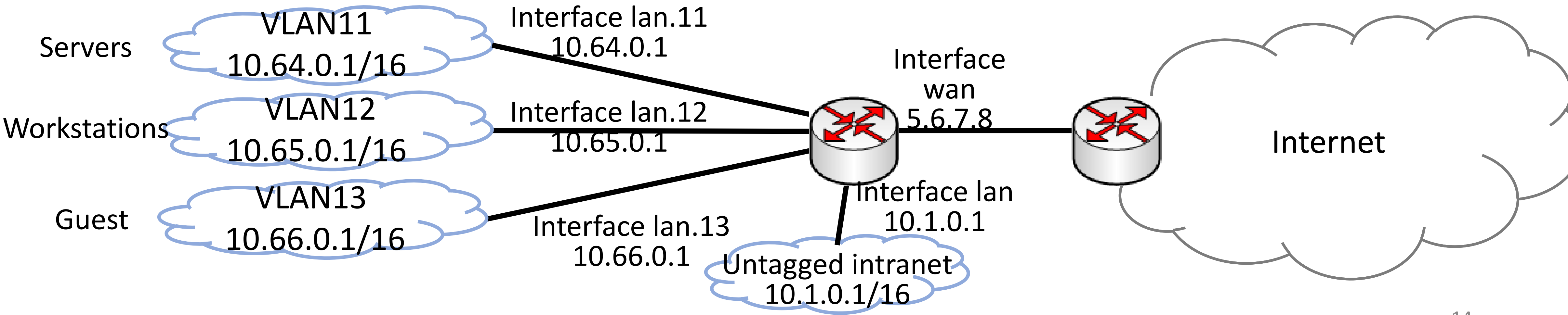
# iptables

- Firewall implementation for Unix/Linux
- Complex policies can be defined as multiple chains of rules:
  - Action can be a reference to another chain
  - Provides modularity (“subroutines”) for firewall policies
- Lots of good examples:
  - http://fwbuilder.sourceforge.net/4.0/docs/users_guide5/cookbook.shtml
    (browse the subsections for iptables, PF and PIX examples)

# FILTERING BETWEEN NETWORK SEGMENTS

# Filtering between VLANs

| Input if | Prot | Src IP | Src port | Dst IP | Dst port | Flags | Action | Comment |
|---|---|---|---|---|---|---|---|---|
| lan.11 | TCP | 10.64.0.0/16 | * | not 10.0.0.0/8 | * |  | Pass | Outbound from server VLAN11 |
| lan.12 | TCP | 10.65.0.0/16 | * | not 10.0.0.0/8 | * |  | Pass | Outbound from workstation VLAN12 |
| lan.13 | TCP | 10.66.0.0/16 | * | not 10.0.0.0/8 | * |  | Pass | Outbound from guest VLAN13 |
| wan | TCP | * | * | 10.64.0.0/10 | * | ACK | Pass | Inbound from WAN to VLANs |
| lan.11 | TCP | 10.65.0.0/16 | * | 10.64.0.0/16 | * |  | Pass | Workstations to servers |
| lan.12 | TCP | 10.64.0.0/16 | * | 10.65.0.0/16 | * | ACK |  | Servers to workstations |
| * | * | * | * | * | * |  | Drop | Default rule |



# Zone-based filtering

- What we often want is to limit access between network zones, i.e., LANs or VLANs. Something like this:

| Client if | Server if | Action | Comment |
|---|---|---|---|
| lan.11 | wan | Pass | Outbound connections from servers VLAN11 |
| lan.12 | wan | Pass | Outbound connections from workstations VLAN12 |
| lan.13 | wan | Pass | Outbound connections from guest VLAN13 |
| lan.12 | lan.11 | Pass | Connections from workstations to servers |
| * | * | Drop | Default rule |

- How to implement zone-based firewall rules? Two solutions:
  - Each zone has a different IP subnet. Filter inbound packets at each source interface based on their destination IP address. Define separate rules for IPv4 and IPv6.
  - Mark packet at input interface, filter based on the mark at the output interface. (Works even after network renumbering, e.g., by IPv6 prefix delegation.)
- Still need stateless or stateful implementation of connection direction