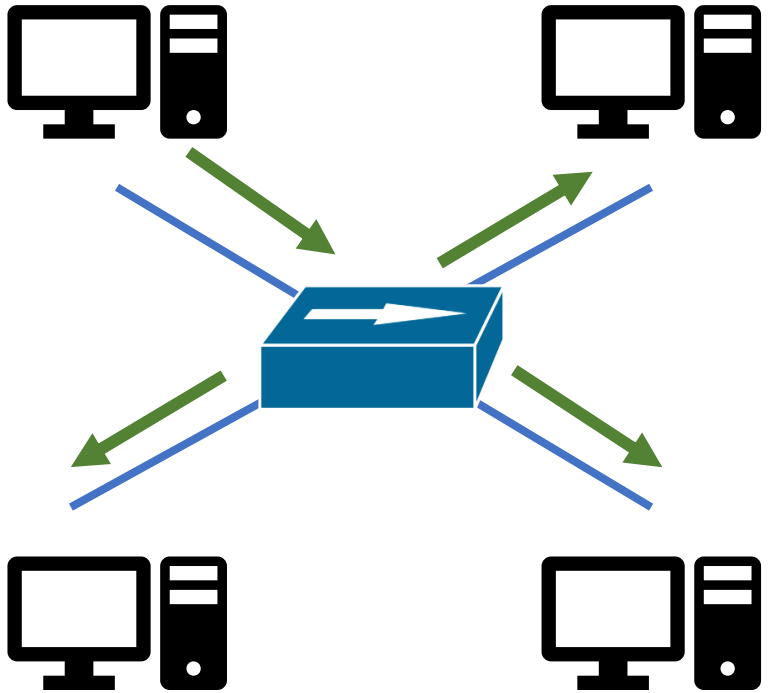
Network Monitoring

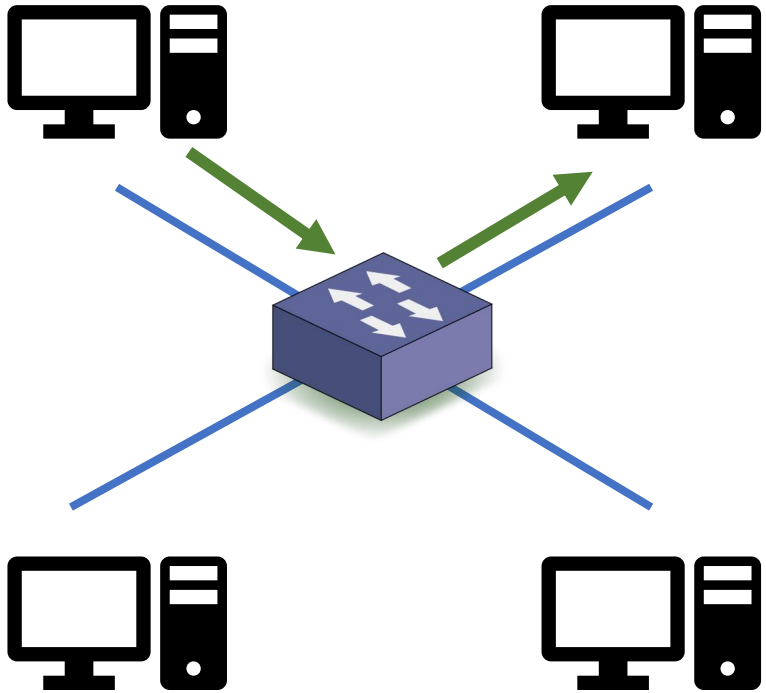
Hub

- L1 device
- Repeats the traffic on one port to other ports (i.e., broadcast)
  - Often runs at half-duplex
- Usages:
  - Mirror traffic for analysis
  - Making multiple network devices act as one segment
- Obsolete and rarely deployed in modern networks



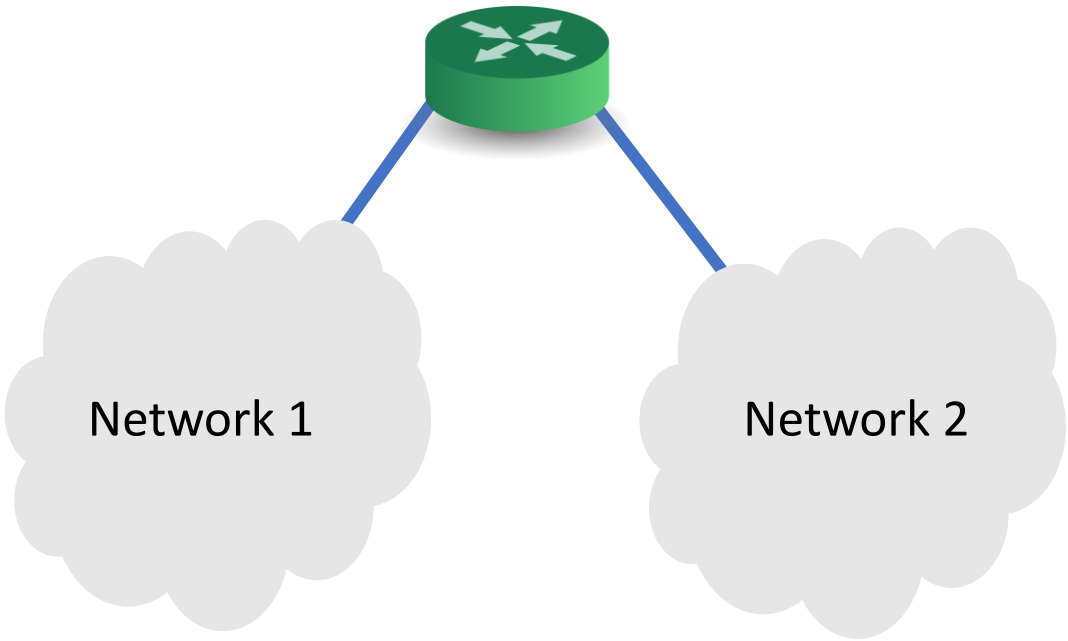
Ethernet Switch

- L2 device
- Decides outgoing port based on dst MAC
- Maintains a mapping between MAC address and outgoing ports
  - Using a Forwarding Information Base
- Modern switches become smarter
  - Programmable



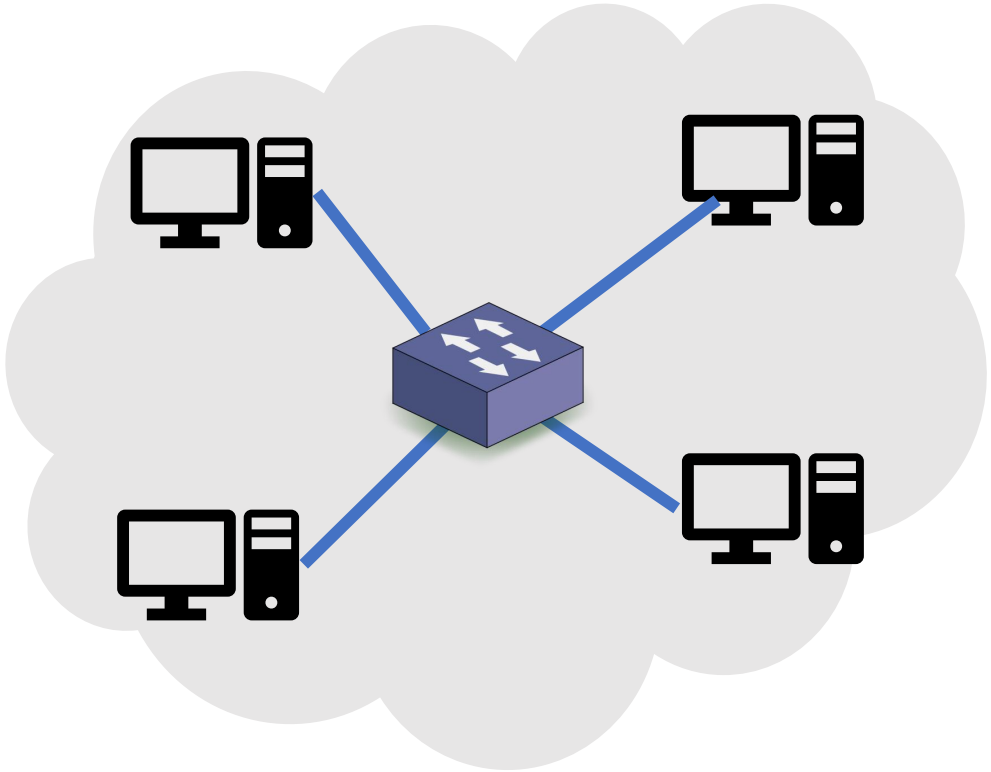
Router

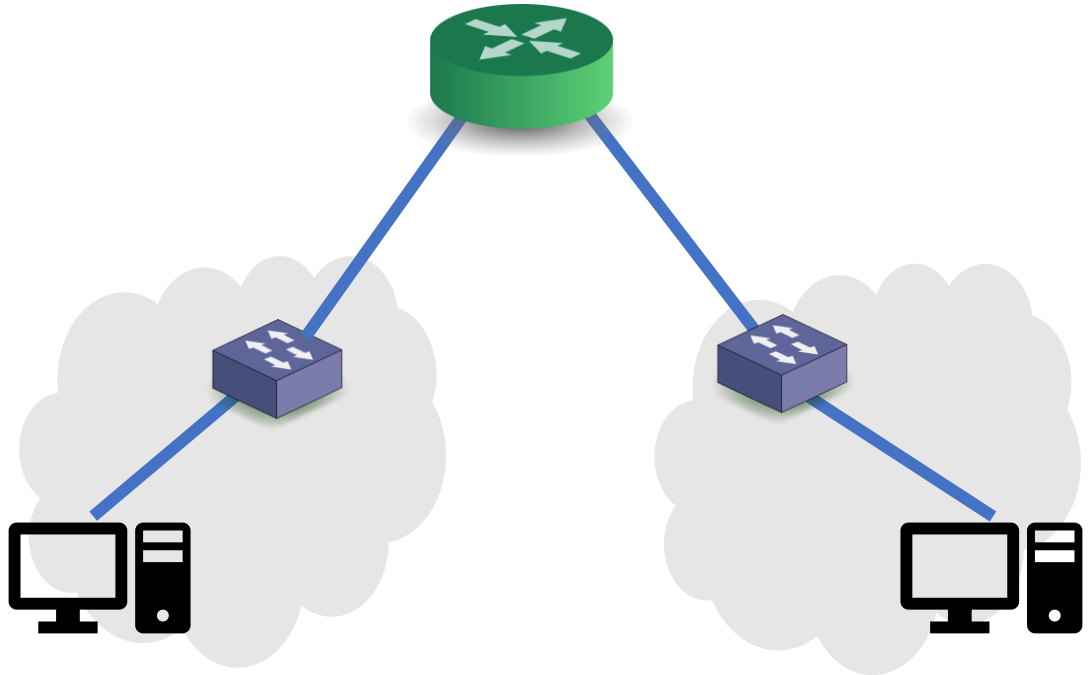
- L3 device
- Forwards packets based on IP address
- Knows L3 routes and neighbors, understands network topology
  - Using a Routing Information Base, created and conveyed with OSPF/BGP
- Does a router need to know L2 information?



Address Resolution Protocol (ARP)

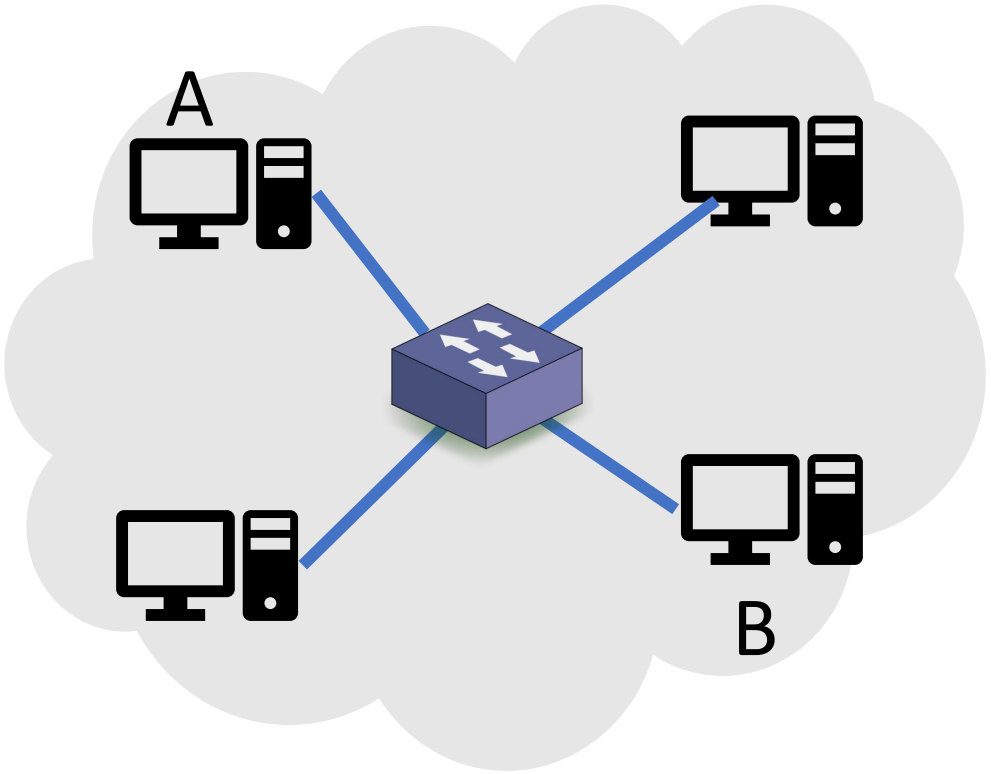
- Two types of addresses are used for communication:
  - Physical (e.g., MAC): within a single network
  - Logical (e.g., IP): among multiple networks, and indirectly connected devices





Address Resolution Protocol (ARP)

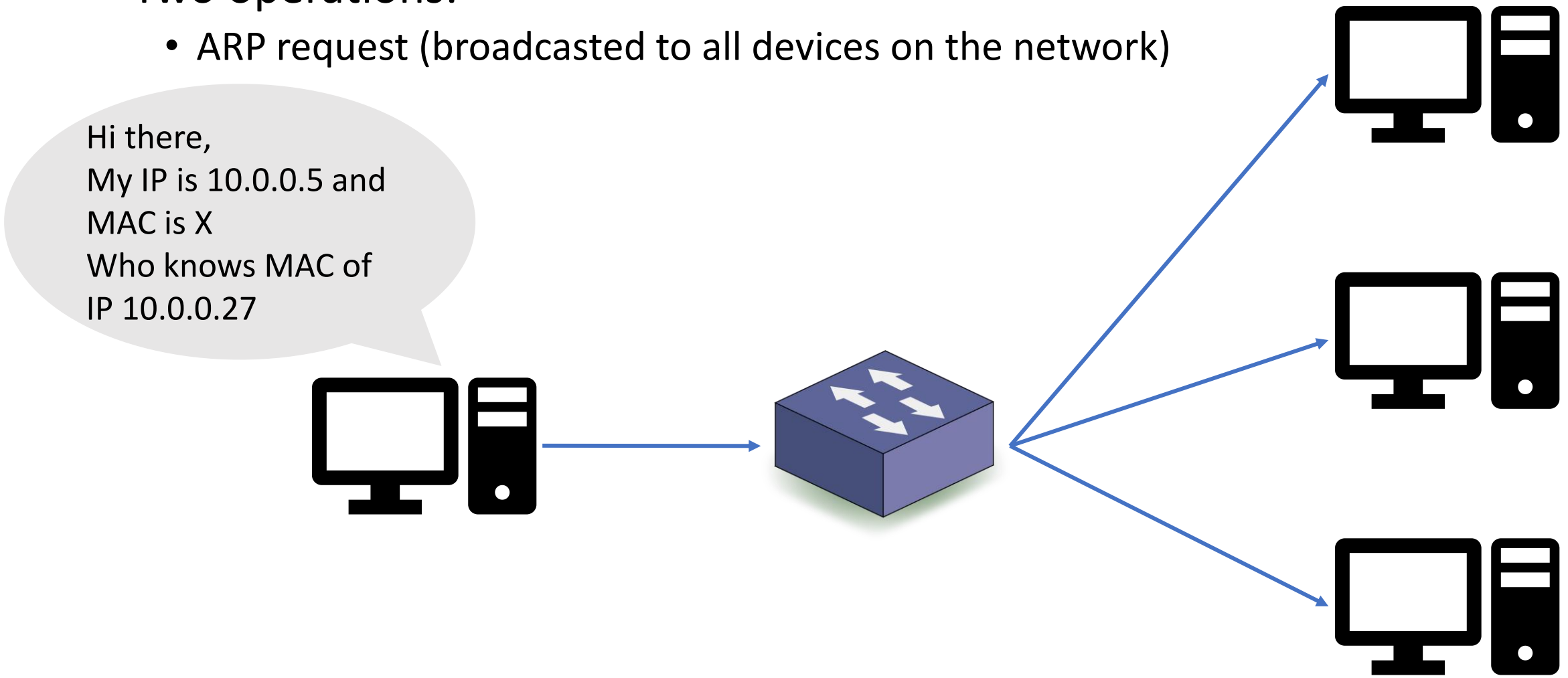
- Consider the case when:
  - an application at A communicates with an app at B
- Device A needs to fill fields L2—L5
  - It has all the information of L3 (why?)
- However, device A does not know the MAC address of device B
  - A field in L2 (dst MAC)



ARP (RFC 826): a protocol to map an IP address to MAC address

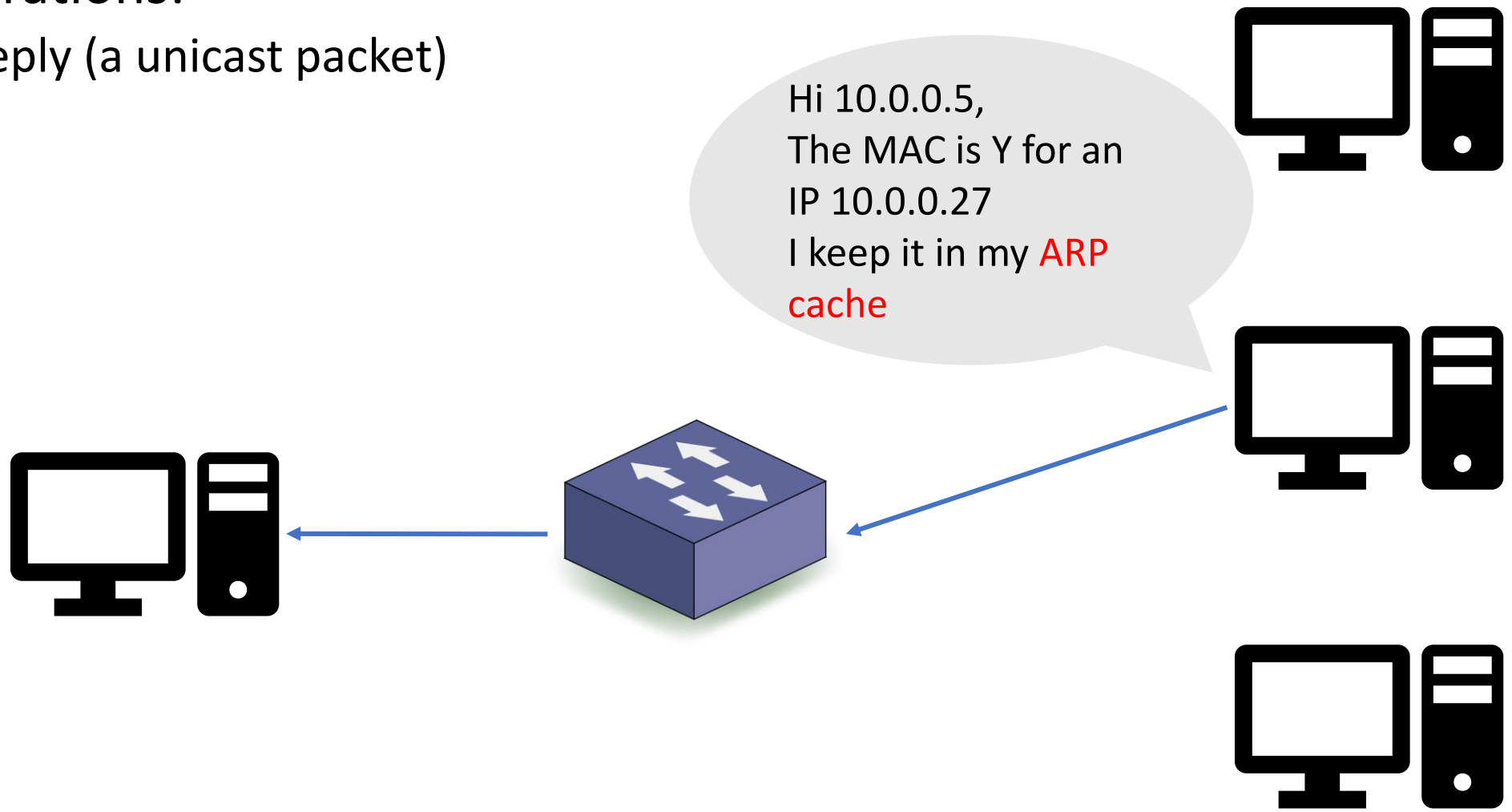
Address Resolution Protocol (ARP)

- Two operations:
  - ARP request (broadcasted to all devices on the network)



Address Resolution Protocol (ARP)

- Two operations:
  - ARP reply (a unicast packet)

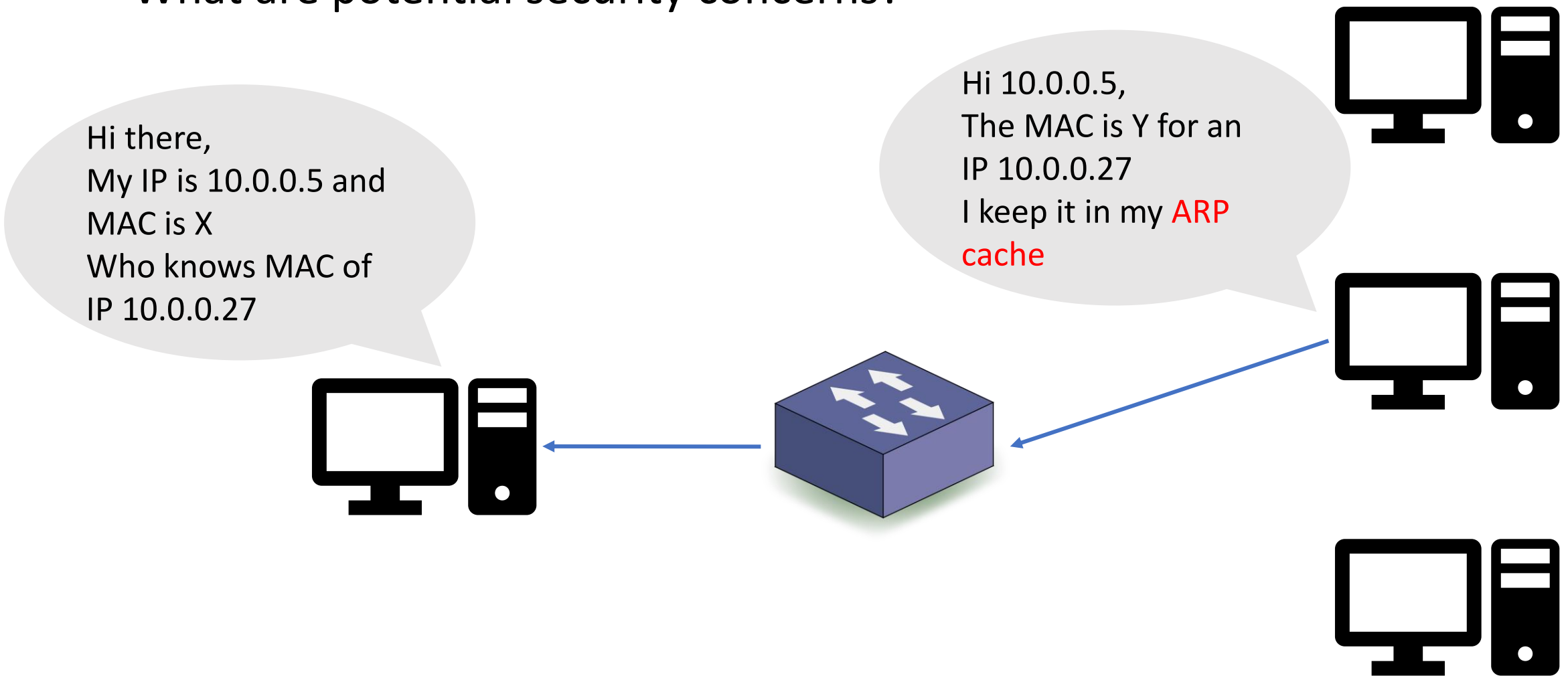


ARP Packet Structure

Address Resolution Protocol (ARP)					
Offsets	Octet	0	1	3	4
Octet	Bit	0–7	8–15	0–7	8–15
0	0	Hardware Type		Protocol Type	
4	32	Hardware Address Length	Protocol Address Length	Operation	
8	64	Sender Hardware Address			
12	96	Sender Hardware Address		Sender Protocol Address	
16	128	Sender Protocol Address		Target Hardware Address	
20	160	Target Hardware Address			
24+	192+	Target Protocol Address			

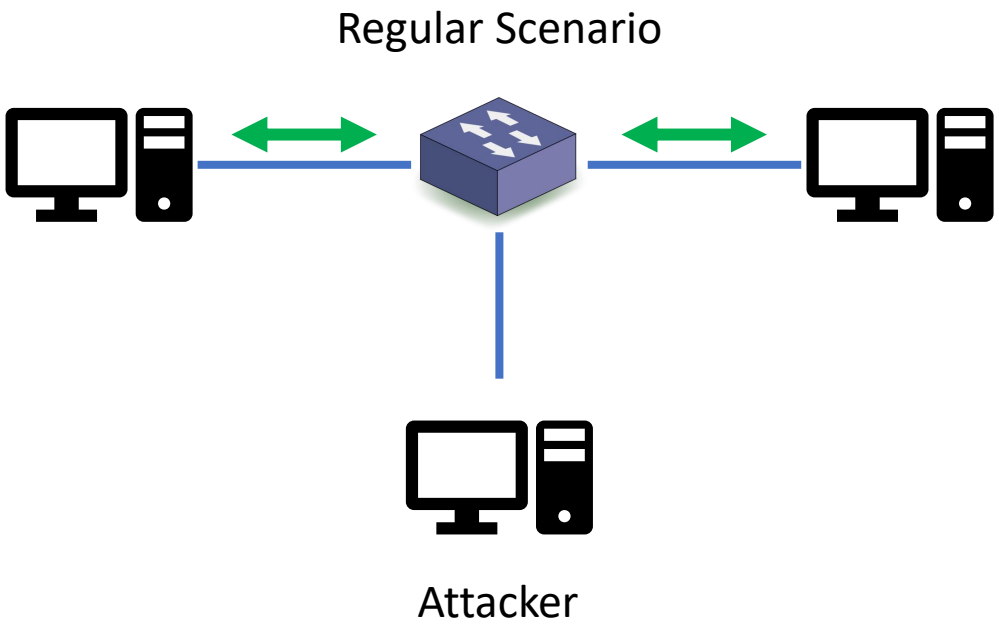
Address Resolution Protocol (ARP)

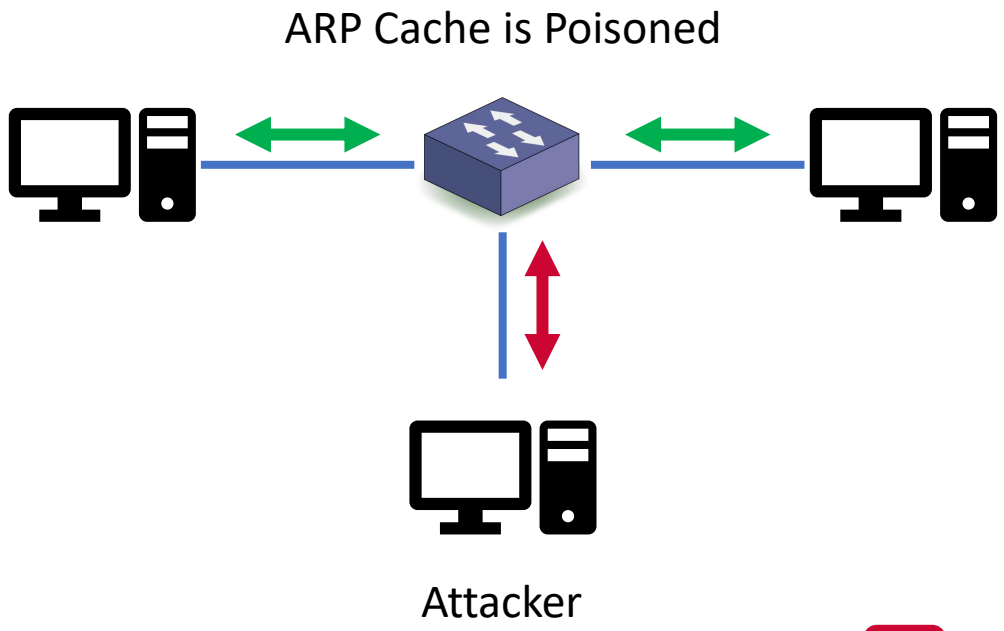
- What are potential security concerns?



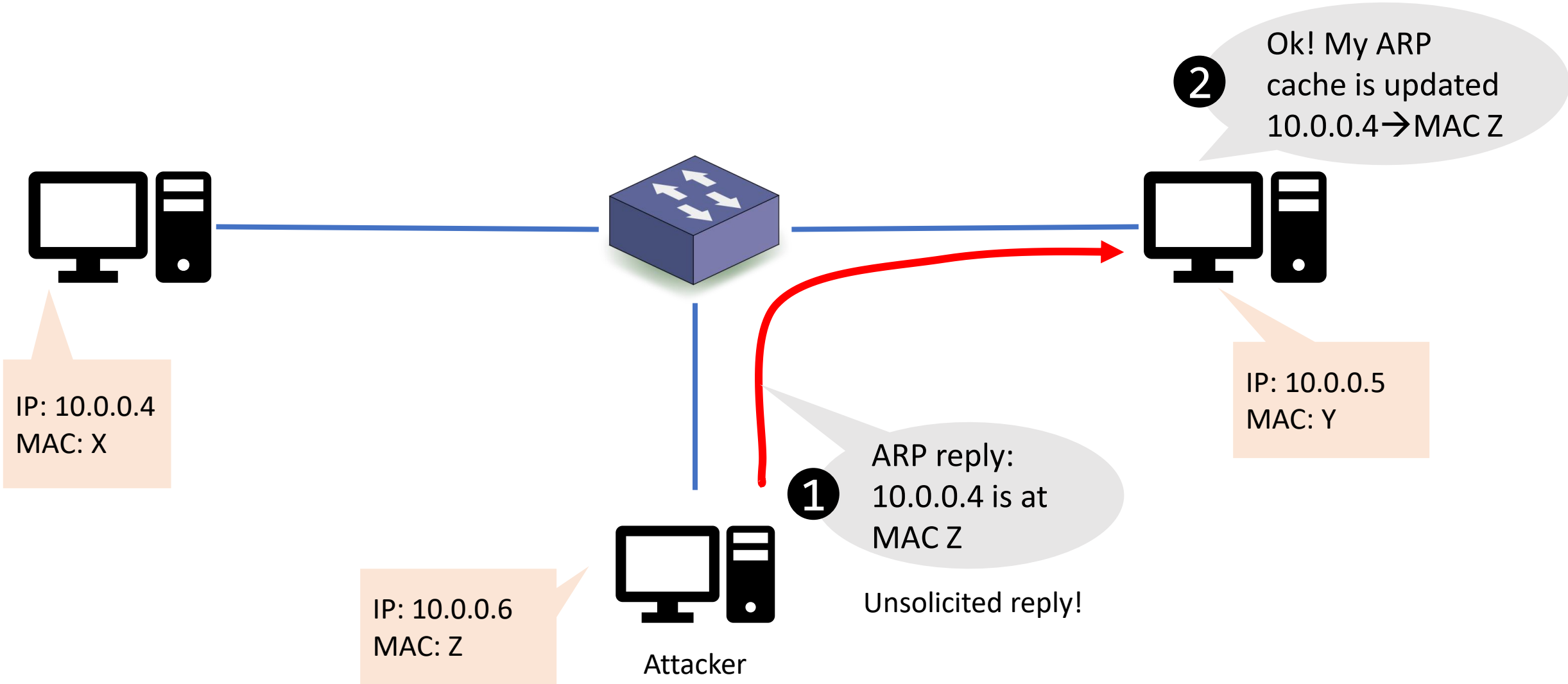
ARP Cache Poisoning

- A crafted ARP packet:
  - tricks two endpoints into thinking they're communicating with each other
  - but, they are communicating with the attacker!
- Consequences: DoS, MITM (e.g., HTTP session hijacking).





ARP Cache Poisoning



ARP Cache Poisoning: Root Cause

- Weakness: ARP is a stateless protocol
  - Doesn’t store requests in memory
- ARP hosts don’t authenticate ARP replies:
  - Even if a host doesn’t send an ARP request
  - Overwrites an ARP entry (even if it hasn’t expired)!

ARP Cache Poisoning: Defenses

- Static ARP entries:
  - Cannot be changed by the attacker
  - Good for small networks (or networks that don’t change)

- IDS or Ethernet switches
  - Detect unsolicited replies

Internet Control Message Protocol (ICMP)

- RFC 792
- A utility protocol of TCP/IP
- Provides information about availability of:
  - Devices, services, or routes on a TCP/IP network
- Popular utilities that use ICMP:
  - Ping
  - Traceroute

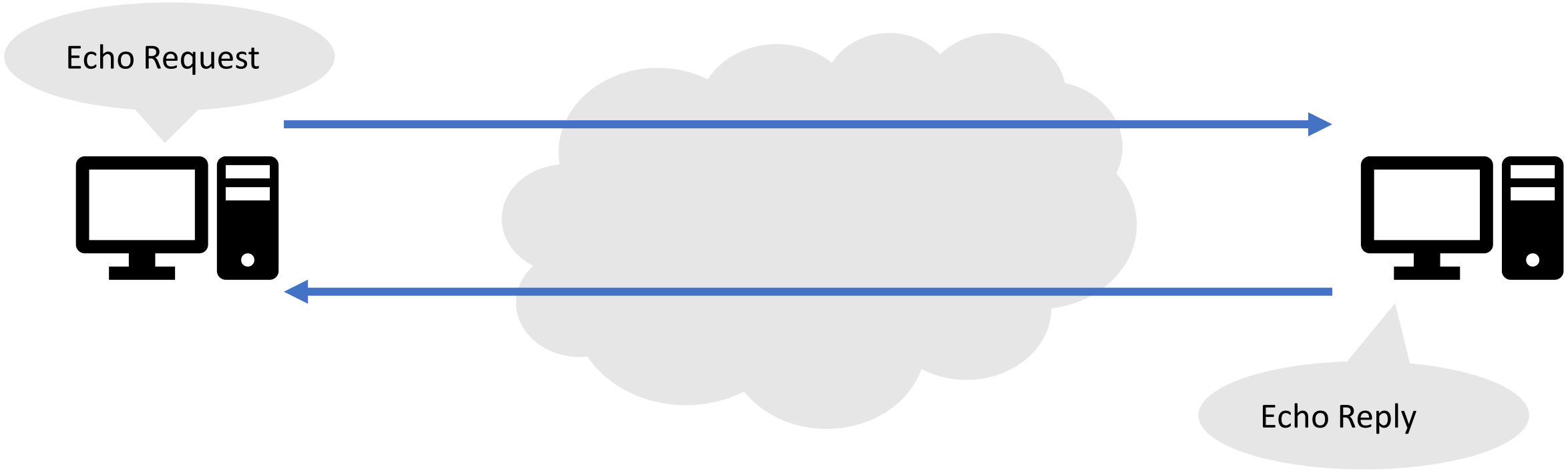
ICMP Packet Structure

Internet Control Message Protocol (ICMP)					
Offsets	Octet	0	1	2	3
Octet	Bit	0–7	8–15	16–23	24–31
0	0	Type	Code	Checksum	
4+	32+	Variable			

0 : Echo Reply
8 : Echo Request
11: Time Exceeded

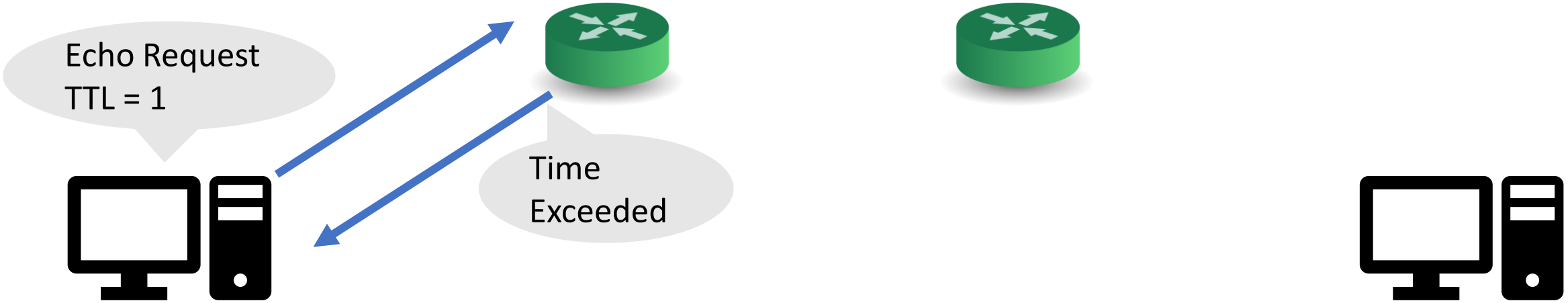
ICMP: `ping`

Often used to check availability



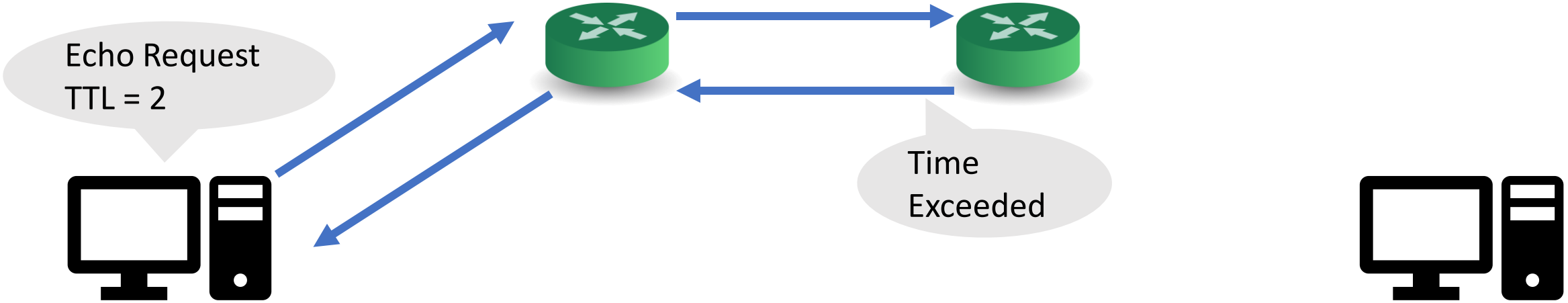
ICMP: `traceroute`

Build a path of routers from source to destination. How?



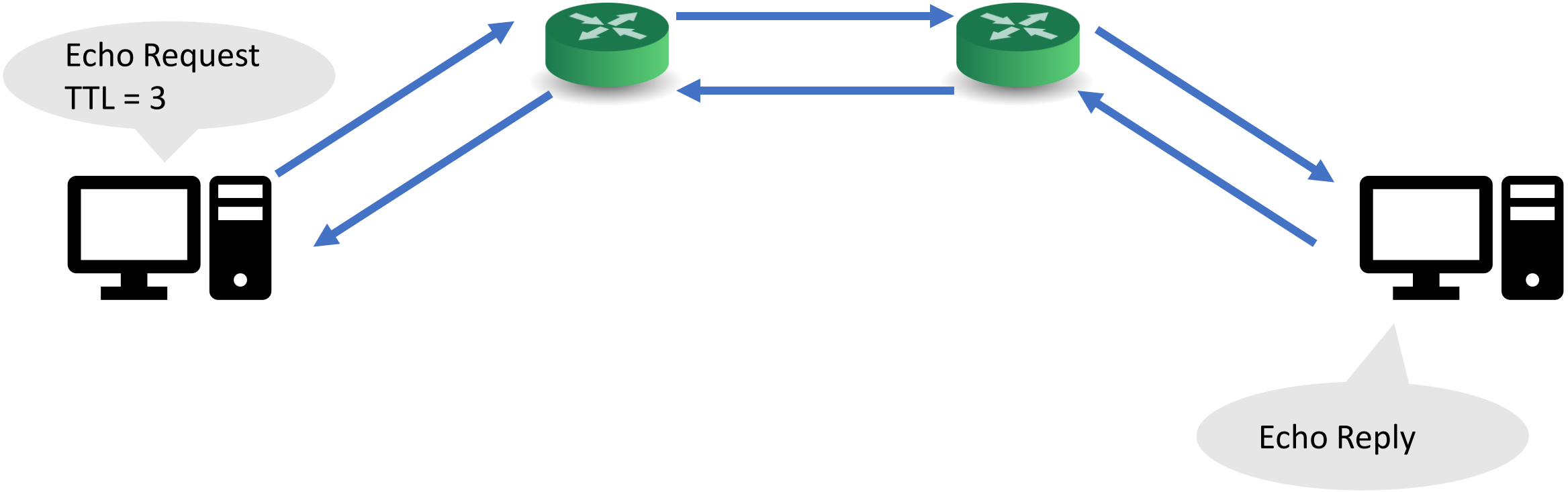
ICMP: `traceroute`

Build a path of routers from source to destination. How?



ICMP: `traceroute`

Build a path of routers from source to destination. How?

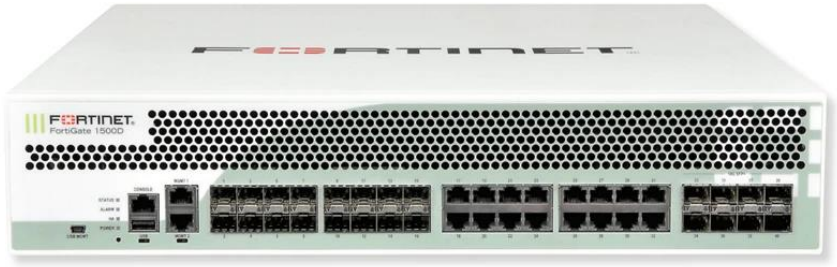


Network Analysis

- Analyze network traffic for different goals.

- Useful for:
  - Intrusion Analyst: dissect network traffic to study intrusions
  - Forensic Investigator: check the extent of a malware infection
  - Attackers: understand their victim networks!

Phases of Network Security Monitoring



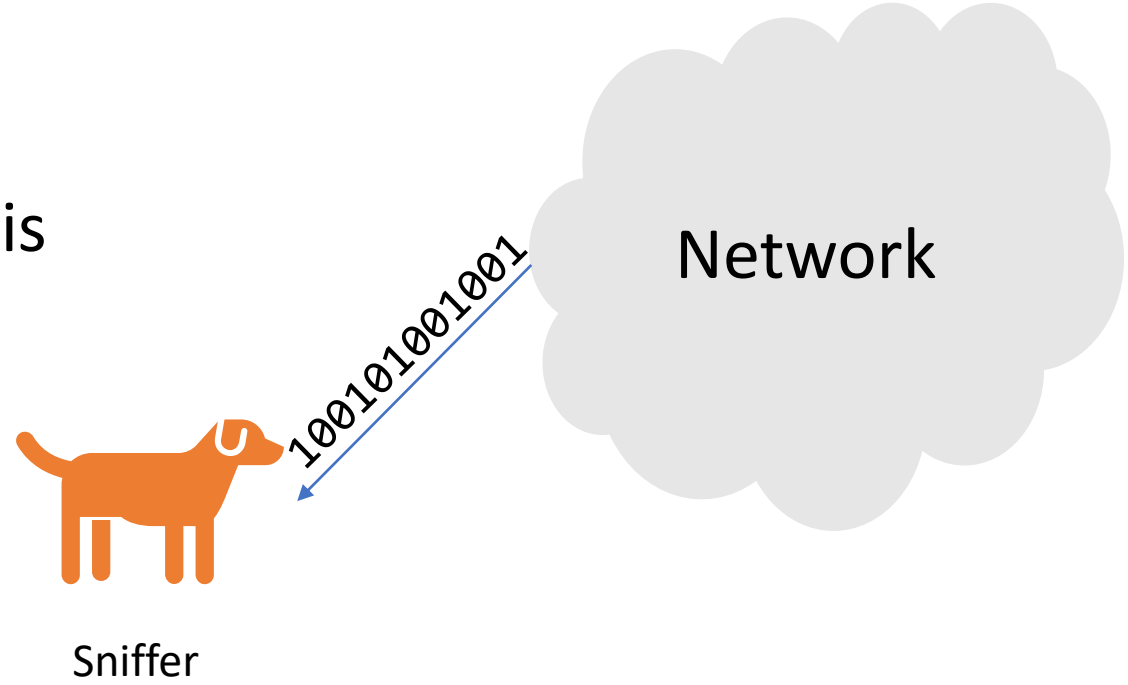
Some devices can perform the three operations

Data Types

- Full Packet Capture Data
  - All transmitted packets, their headers and contents
  - Popular format is pcap
  - Large size but useful for analysis
- Session (Flow) Data
  - Derived from pcap by analyzing headers
  - e.g. Wireshark is able to analyze specific sessions
  - Can choose to store only useful information

Sniffing Packets

- The process of capturing network traffic (i.e., packets)
  - By a sniffer (or a sensor)
- Packets are stored for further analysis
- This requires modifications to:
  - The network
  - The sniffer

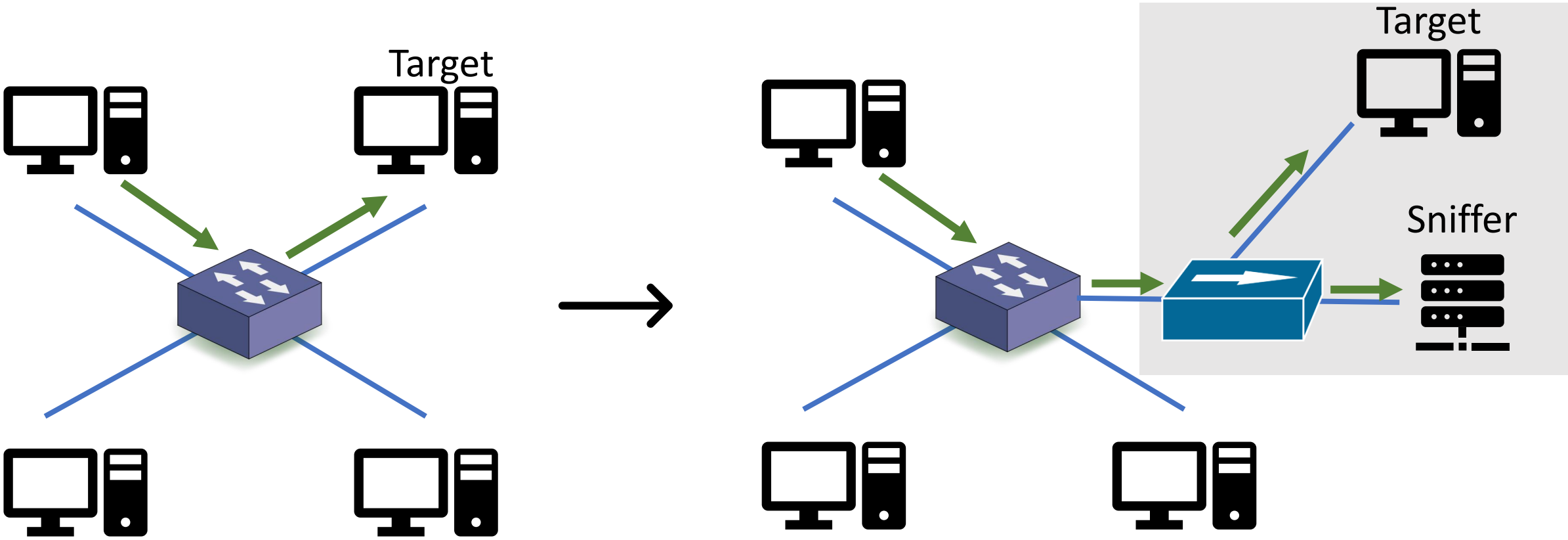


Tapping into the Wire

- How can a sniffer capture traffic?

- Three techniques in switched networks:
  - Installing a Hub
  - Port mirroring
  - Network TAP

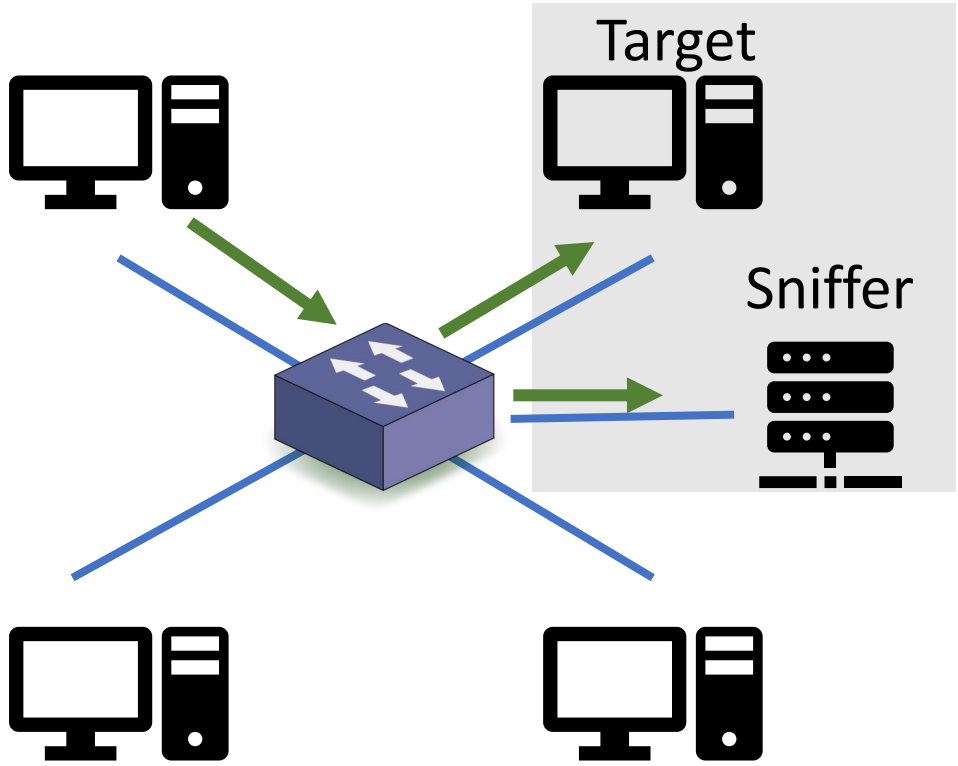
Installing a Hub



Port Mirroring

- Copies traffic from one port to another
- Easy way to capture traffic
- Low-cost option

- Requirements:
  - Access to switch command line
  - Support of port mirroring
  - Available port

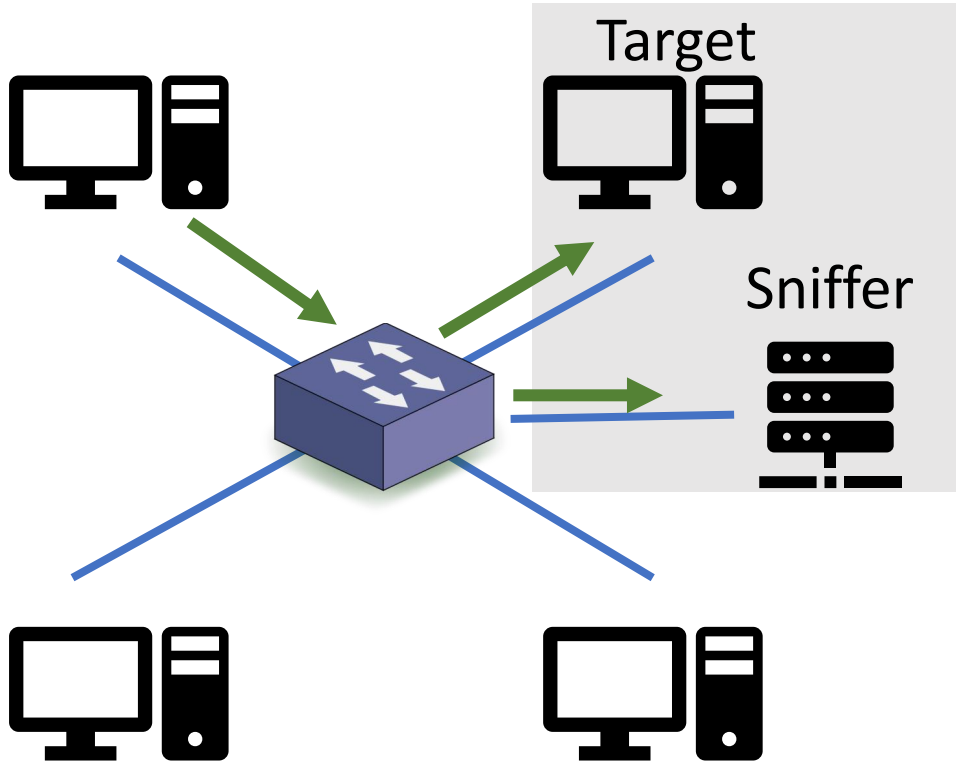


Port Mirroring: Configuration

- Configuring port mirroring on a switch is vendor-specific.
  - Usually happens through command line
  - Sometimes through GUI or web interface

- For example, for Cisco switches:

```
set span <src_port> <dst_port>
```

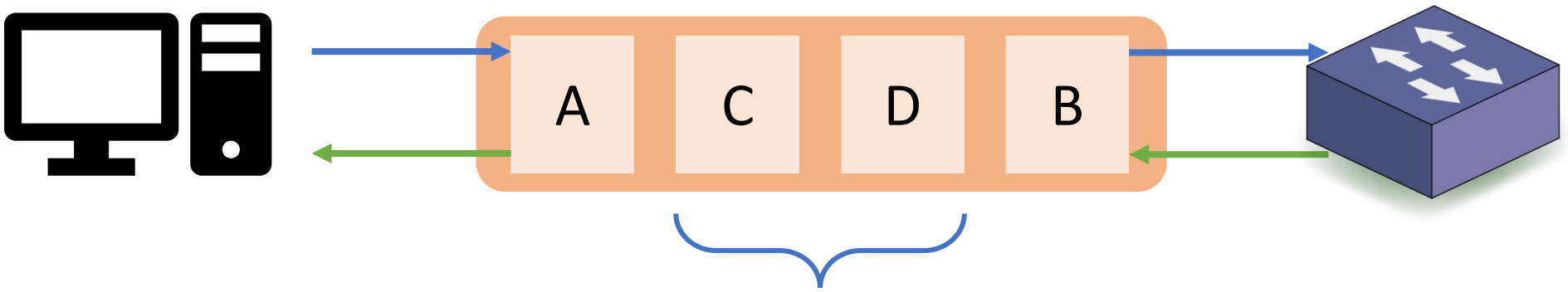


Port Mirroring

- In general, port mirroring is not reliable for high-throughput applications such as network security monitoring
- If multiple ports are mirrored to a single output port (oversubscription)
  - Packet losses
  - Slowing down the switch
- Timing is not accurate

Installing a TAP

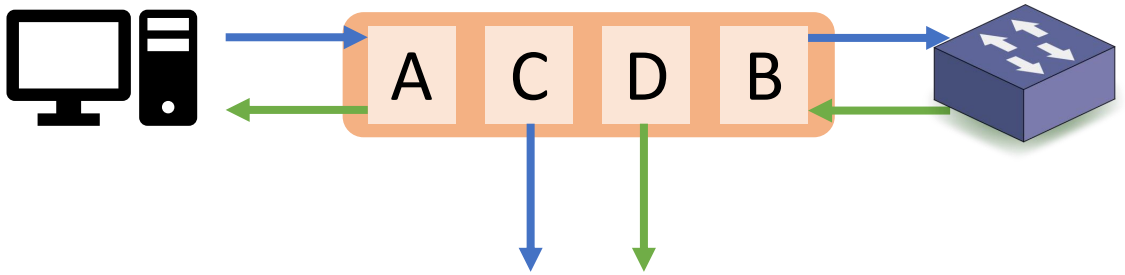
- TAP: Test Access Point
- Specialized hardware that allows traffic to flow:
  - from port A → port B, and
  - from port B → port A
- Creates an exact copy of both sides of the flow
  - Without loss



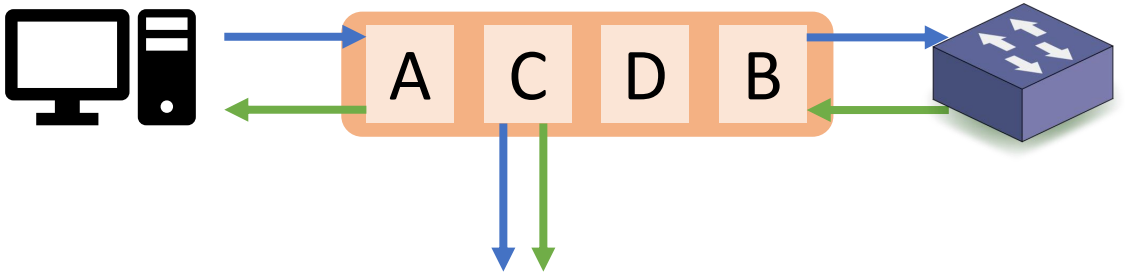
To a sniffer/monitoring device

Installing a TAP: Modes

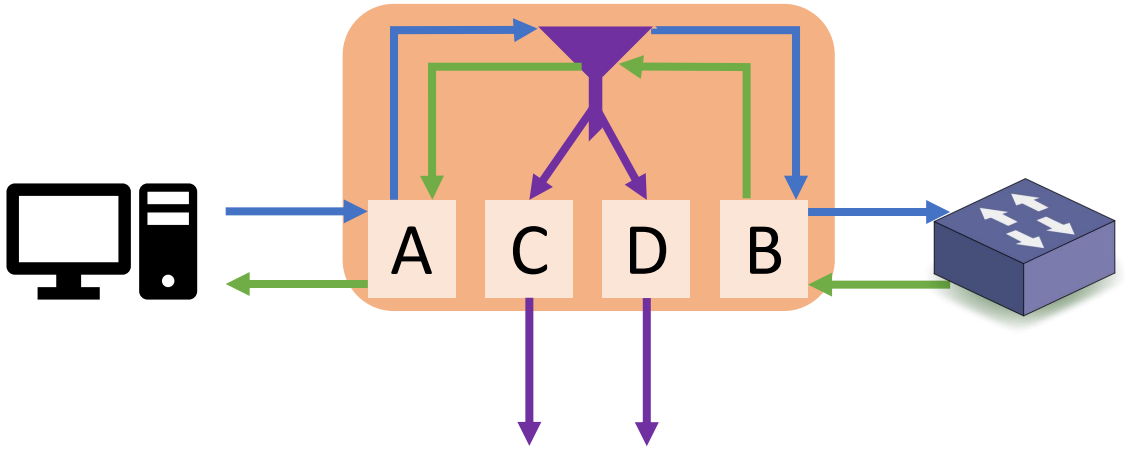
Breakout Mode



Aggregation Mode

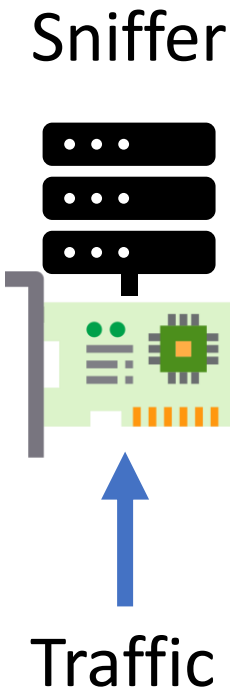


Filter Mode



Sniffer Machine

- The sniffer receives network traffic
  - that wasn't destined for the sniffer
- This happens in some network protocols as well.
  - Examples?
- The default behavior of NIC is to discard these packets
  - Reduce CPU processing
  - Not useful for the sniffer!



NIC: Promiscuous Mode

- NICs support “promiscuous” mode
  - Allows the NIC to receive traffic not destined for the sniffer
  - The NIC then passes sniffed packets to the CPU for further processing

- Check an interface:

```
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default qlen 1
...
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP group default qlen 1000
```

loopback intf

NIC intf

NIC: Promiscuous Mode

- Enable promiscuous mode:

```
$ sudo ip link set enp0s3 promisc on
```

- Check again:

```
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default qlen 1
...
2: enp0s3: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500
qdisc pfifo_fast state UP group default qlen 1000
```

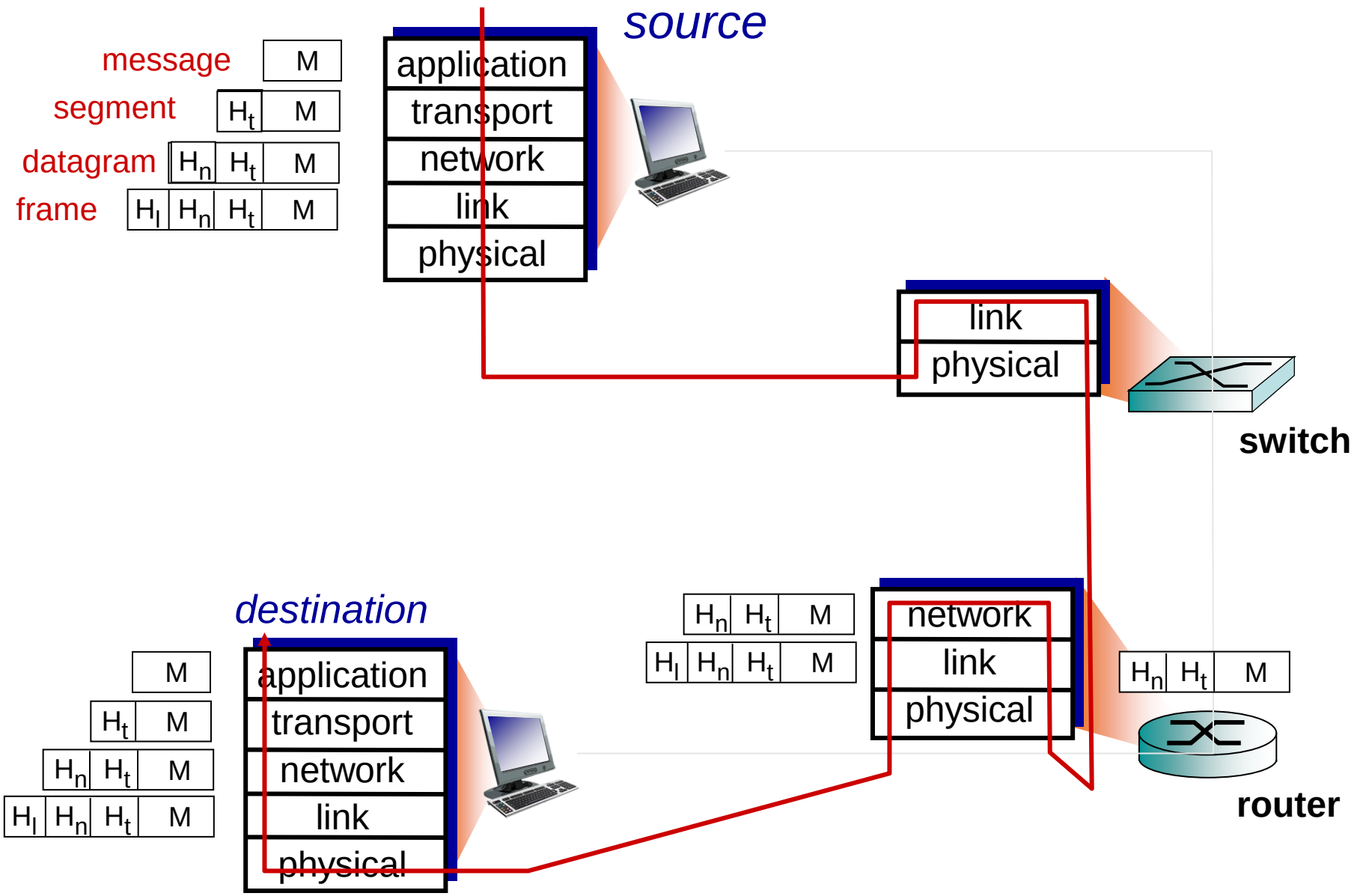
PROMISC enabled

Dissecting Packets

Recall: Packet Switching

- Packet Switching: Hosts break application-layer messages into packets
  - Forward packets from one router to the next, across links on path from source to destination
  - Each packet is transmitted at full link capacity (no reservation)
- The header of each packet carries necessary information
  - Routers examine the header and make forwarding decisions

Recall: Encapsulation



Packet Representation

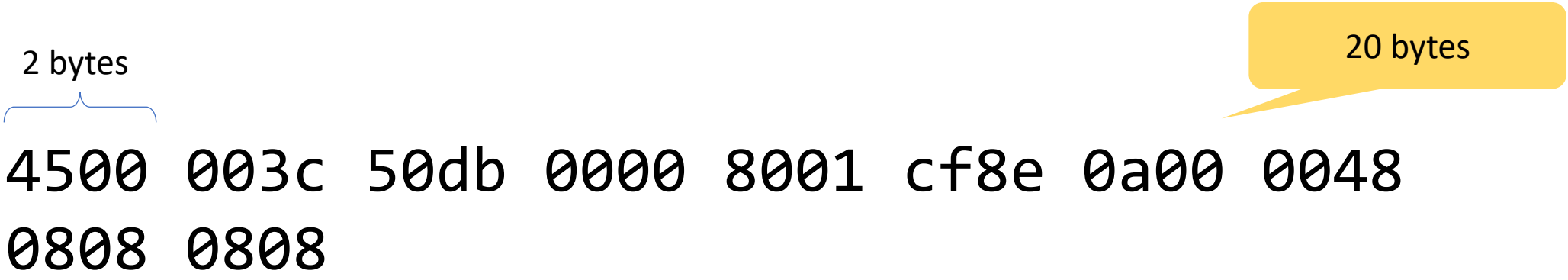
- Packet is a sequence of bytes
  - Formatted based on the rules of protocols
  - Multiple fields, each has a specific value

- Binary representation:
  - Sequence of 0's and 1's
  - E.g.,
    `01000101000000000000000001111000101000011011011000000000`
    `00000001000000000000011100111110001110`

  - Hard to read

Packet Representation

- Hex representation
- Uses numbers 0–9 and letters a–f
- A byte is represented using two characters
  - E.g., 2a is one byte



What is this protocol? What information is here?

Packet Diagram

- A graphical representation of a packet
  - Allows analysts to map bytes to fields
  - Often based on protocol's RFC

Internet Protocol Version 4 (IPv4)							
Offsets	Octet	0		1	2		3
Octet	Bit	0–3	4–7	8–15	16–18	19–23	24–31
0	0						
4	32						
8	64						
12	96						
16	128						
20	160						
24+	192+						

Packet Diagram

- A graphical representation of a packet
  - Allows analysts to map bytes to fields
  - Often based on protocol's RFC

Internet Protocol Version 4 (IPv4)							
Offsets	Octet	0		1	2		3
Octet	Bit	0–3	4–7	8–15	16–18	19–23	24–31
0	0	Version	Header Length	Type of Service	Total Length		
4	32	Identification			Flags	Fragment Offset	
8	64	Time to Live		Protocol	Header Checksum		
12	96	Source IP Address					
16	128	Destination IP Address					
20	160	Options					
24+	192+	Data					

Packet Diagram

```
4500 003c 50db 0000 8001 cf8e 0a00 0048
0808 0808
```

Internet Protocol Version 4 (IPv4)							
Offsets	Octet	0		1	2		3
Octet	Bit	0–3	4–7	8–15	16–18	19–23	24–31
0	0	4	5	00	003c		
4	32	50db			Flags	Fragment Offset	
8	64	80		01	cf8e		
12	96	0a00 0048					
16	128	0808 0808					
20	160	Options					
24+	192+	Data					

Packet Diagram

- Protocol is 0x01. What is this protocol?
- Check IP protocol numbers.

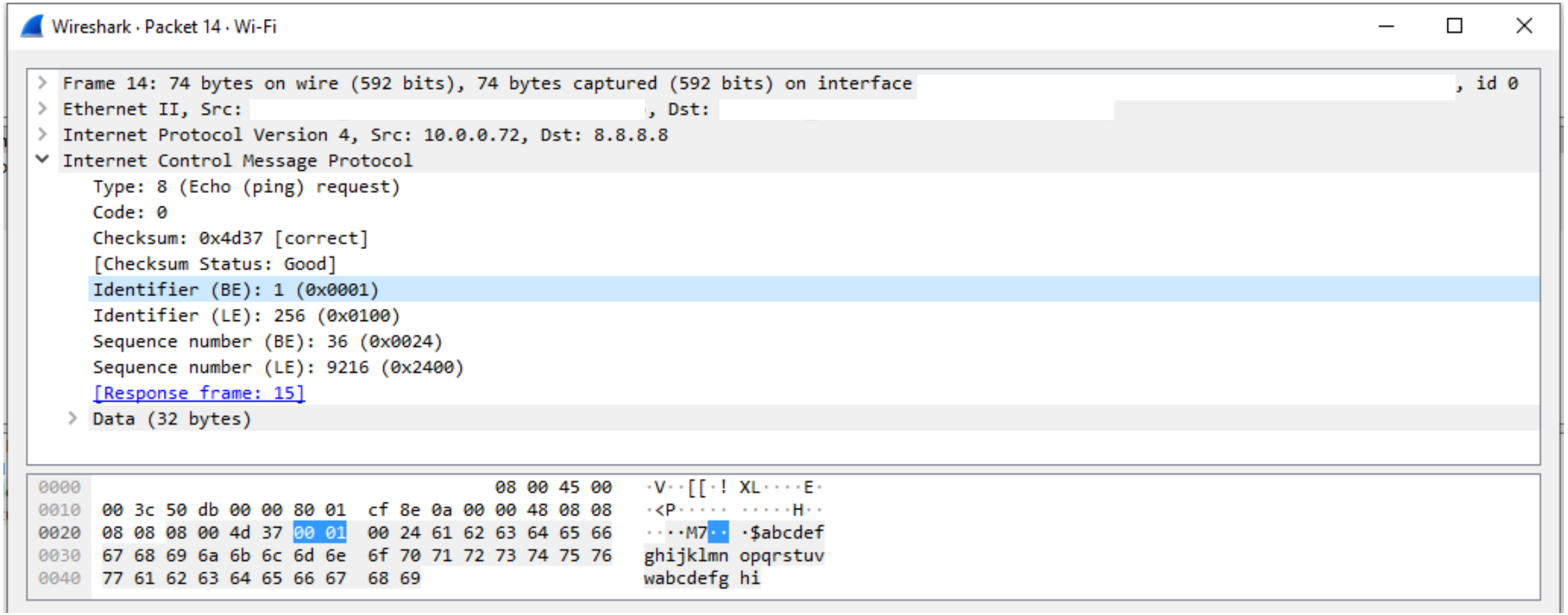
Internet Protocol Version 4 (IPv4)							
Offsets	Octet	0		1	2		3
Octet	Bit	0–3	4–7	8–15	16–18	19–23	24–31
0	0	4	5	00	003c		
4	32	50db			Flags	Fragment Offset	
8	64	80		01	cf8e		
12	96	0a00 0048					
16	128	0808 0808					
20	160	Options					
24+	192+	Data					

IP Protocol Numbers: Examples

Protocol Number (Hex)	Protocol
0x01	ICMP
0x06	TCP
0x11	UDP
0x29	IPv6 (why?)
0x2f	GRE
0x59	OSPF

Tools for Dissecting Packets

- Various tools can be used to dissect and decode a packet



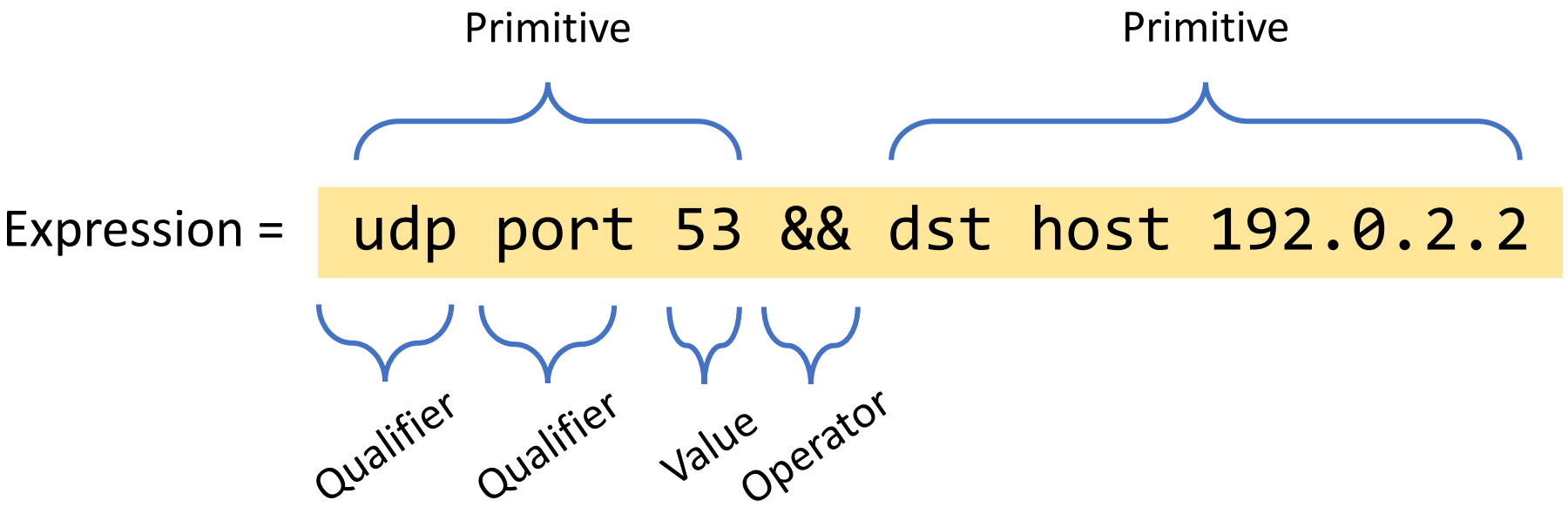
Packet Filtering

- Capture or show packets matching specific fields or criteria
- Packet filtering is used during:
  - The capturing phase. Sniffer may eliminate:
    - unwanted traffic, or
    - traffic that isn't useful for detection/analysis
  - The analysis phase:
    - Analysts often need to focus on specific packets
      - E.g., HTTP packets, ARP requests, Ping (echo reply), etc.
- Berkeley Packet Filter (BPF) is the most commonly used syntax

Berkeley Packet Filters (BPFs)

- McCanne and Jacobson'93 https://www.tcpdump.org/papers/bpf-usenix93.pdf:
  - Filters are translated into a simple instruction/register set used to specify if packets are to be rejected, accepted
  - A simple VM ran the instructions in-kernel and filtered appropriately
  - Safety was the key criterion when injecting filter code.
    - All programs must complete in a bounded time (no loops)

BPF Syntax


Primitive
Primitive
Expression =
udp port 53 && dst host 192.0.2.2
Qualifier
Qualifier
Value
Operator

BPF Syntax

- Three types of qualifiers:
  - **type**: `host, net, port, portrange`
  - **dir**: `src, dst`
  - **proto**: `ether, arp, ip, ip6, icmp, tcp, udp`

```
tcp port 80
```

```
ip host 10.0.0.1
```

=

```
ether proto \ip and host 10.0.0.1
```

BPF Syntax

- Match specific fields in the packet:
  - `icmp[0] == 8`

Internet Control Message Protocol (ICMP)					
Offsets	Octet	0	1	2	3
Octet	Bit	0–7	8–15	16–23	24–31
0	0	Type	Code	Checksum	
4+	32+	Variable			

`0` : Echo Reply
`8` : Echo Request
`11`: Time Exceeded

BPF Syntax

- Match specific fields in the packet:
  - `ip[8] > 64`

Internet Protocol Version 4 (IPv4)							
Offsets	Octet	0		1	2		3
Octet	Bit	0–3	4–7	8–15	16–18	19–23	24–31
0	0	Version	Header Length	Type of Service	Total Length		
4	32	Identification			Flags	Fragment Offset	
8	64	Time to Live		Protocol	Header Checksum		
12	96	Source IP Address					
16	128	Destination IP Address					
20	160	Options					
24+	192+	Data					

BPF Syntax

- Match specific fields in the packet:
  - `tcp[14:2] == 0`

Transmission Control Protocol (TCP)						
Offsets	Octet	0		1	2	3
Octet	Bit	0–3	4–7	8–15	16–23	24–31
0	0	Source Port			Destination Port	
4	32	Sequence Number				
8	64	Acknowledgment Number				
12	96	Data Offset	Reserved	Flags	Window Size	
16	128	Checksum			Urgent Pointer	
20+	160+	Options				

APIs and Tools

- Scapy
- `libpcap`
- `tcpdump`
- `nmap`
- Wireshark
- `tshark`
- …

Questions?