Link Layer
MAC & ARP

OSI 5 Layer Model

Application	Defines how individual applications communicate. For example, **HTTP** defines how browsers send requests to web servers.
Transport	Allows a client to establish a connection to specific services (e.g., web server on port 80). Provides reliable communication.
Network	Packet forwarding. How to get a packet to the final destination when there are many hops along the way.
Data Link	How to get packet to the next hop. Transmission of data frames between two nodes connected by a physical link.
Physical	How do bits get translated into electrical, optical, or radio signals

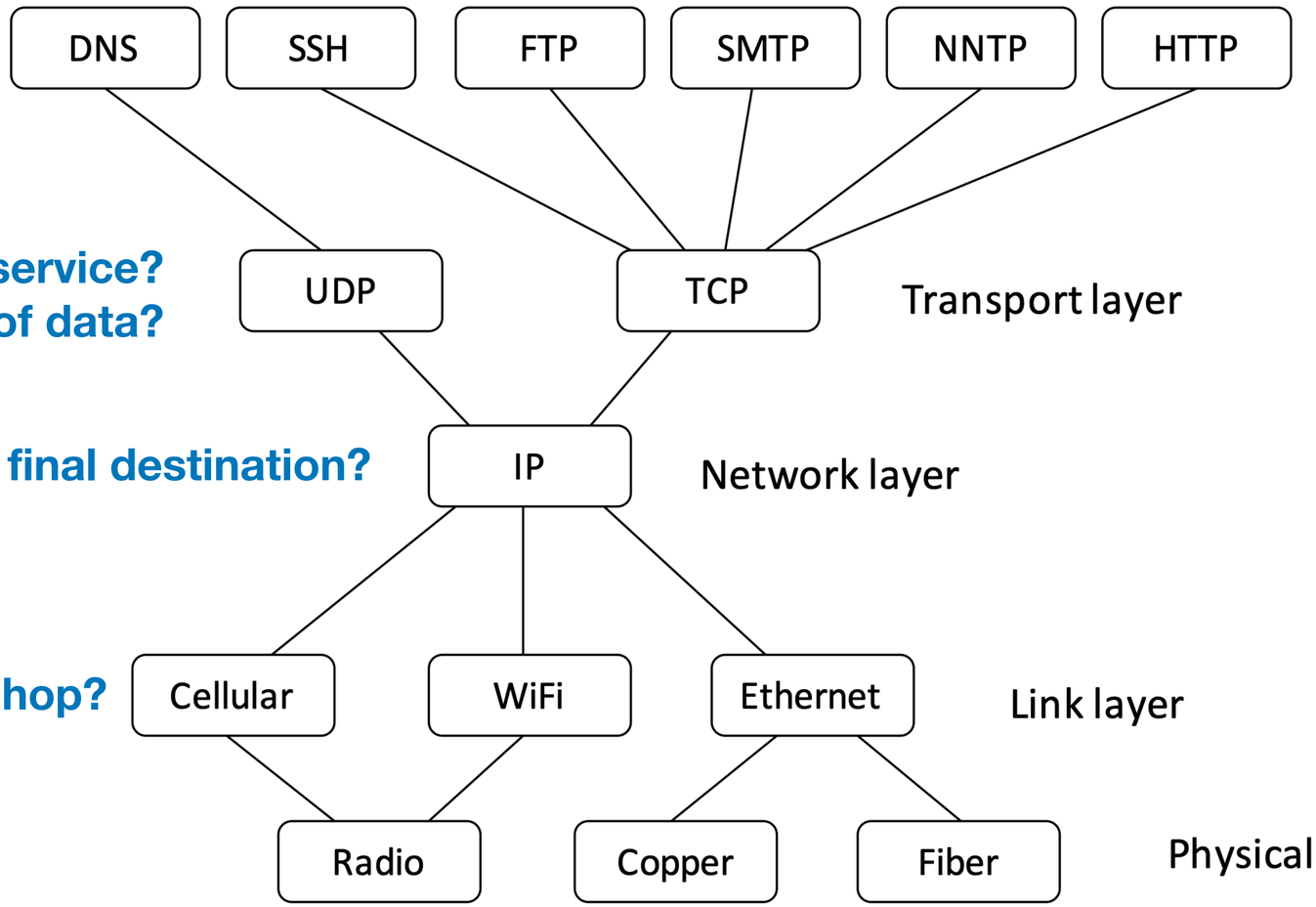
Protocol Layering

How does Application structure data?

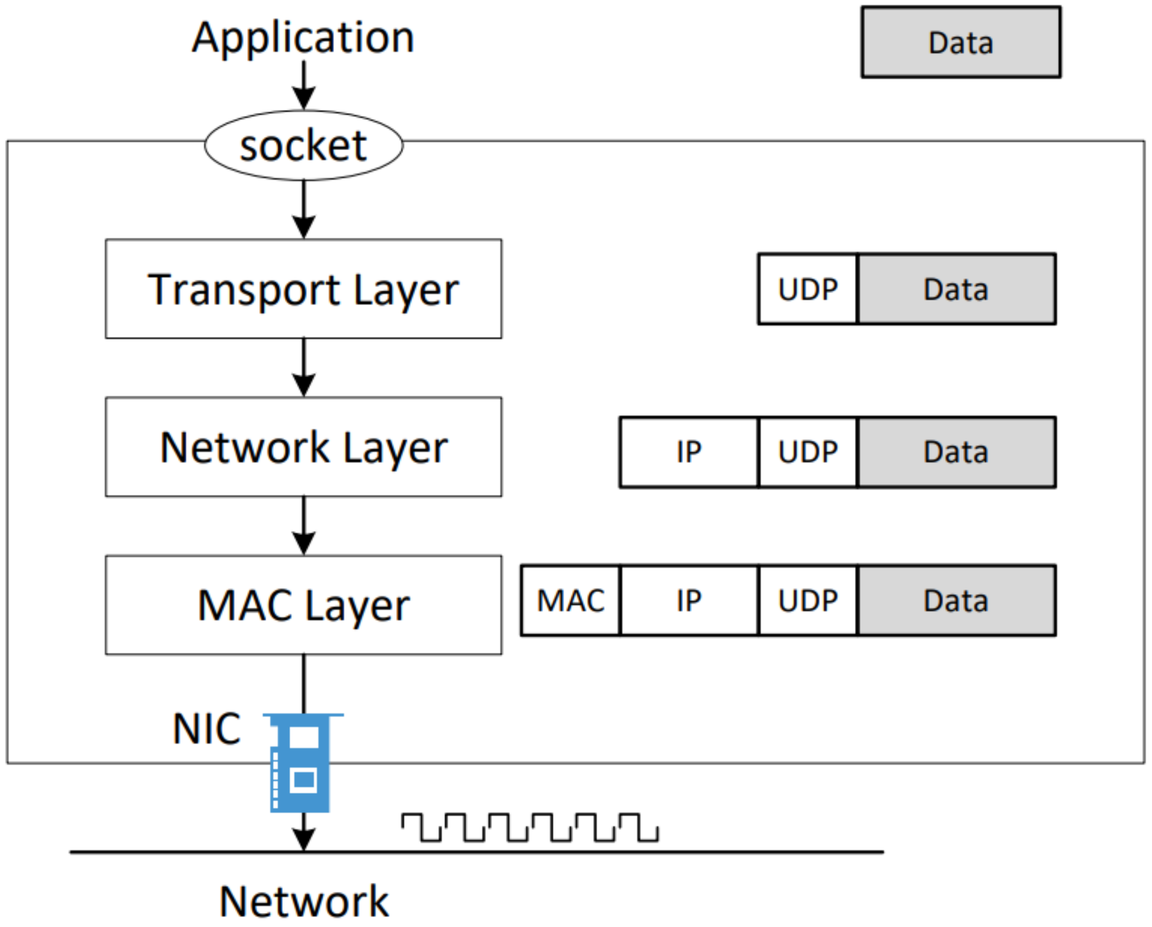
How do I get to the right service?
How do I have a reliable “stream” of data?

How a does packet final destination?

How do I get to next hop?



How Packets Are Constructed



Internet Protocol (IP)

Internet Protocol (IP) defines what packets that cross the Internet need to look like to be processed by routers

Every host is assigned a unique identifier (“IP Address”)

Every packet has an IP header that indicates its sender and receiver

Routers forward packet along to *try* to get it to the destination host

Rest of the packet should be ignored by the router

IP Addresses

192.168.1.65

10.8.2.9

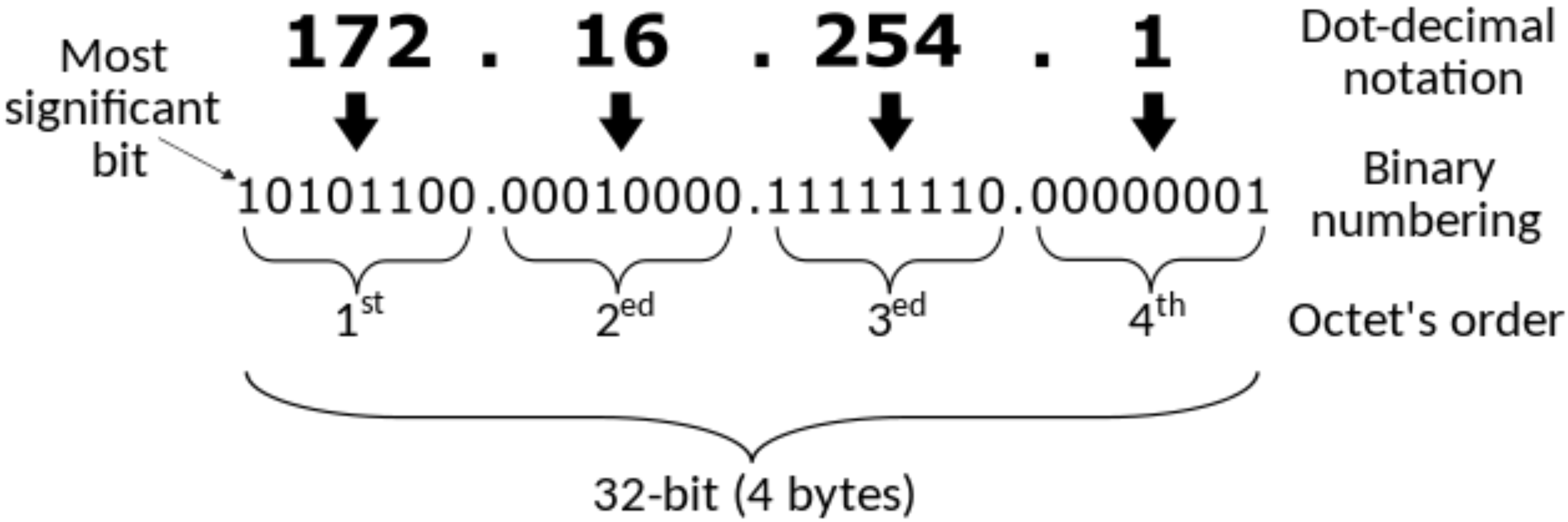
172.16.8.4

10.16.67.14

192.168.1.42

192.168.1.23

IPv4 Address

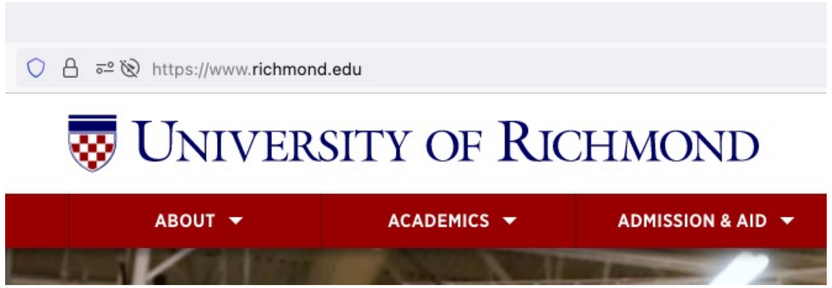


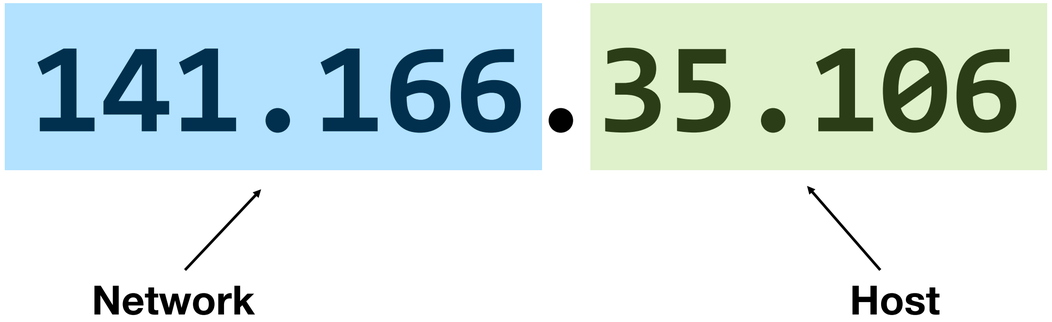
IPv4 Address

1. The first part of an Internet address identifies the network on which the host resides

2. The second part identifies the particular host on the given network

richmond.edu



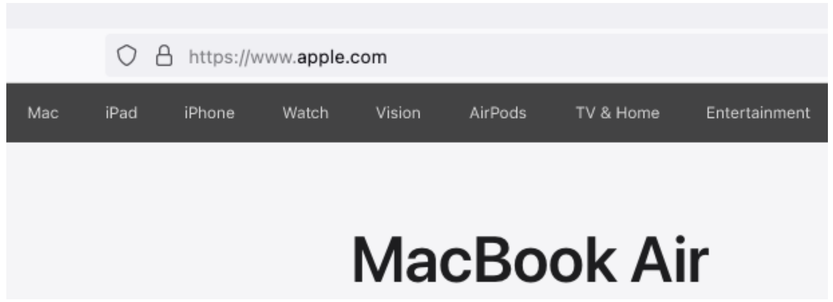


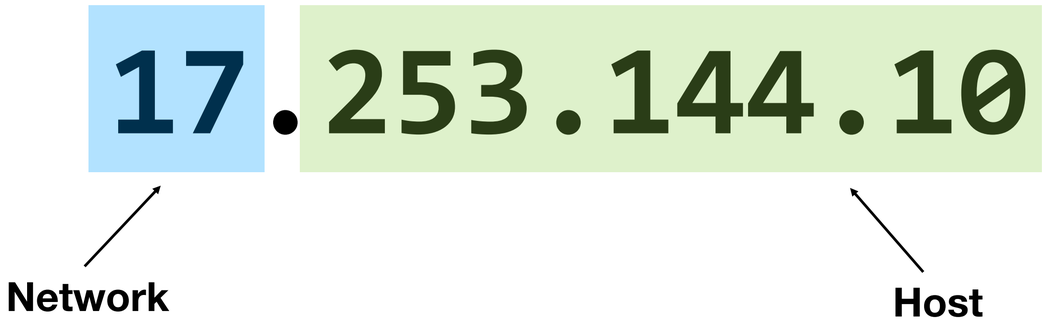
IPv4 Address

1. The first part of an Internet address identifies the network on which the host resides

2. The second part identifies the particular host on the given network

apple.com

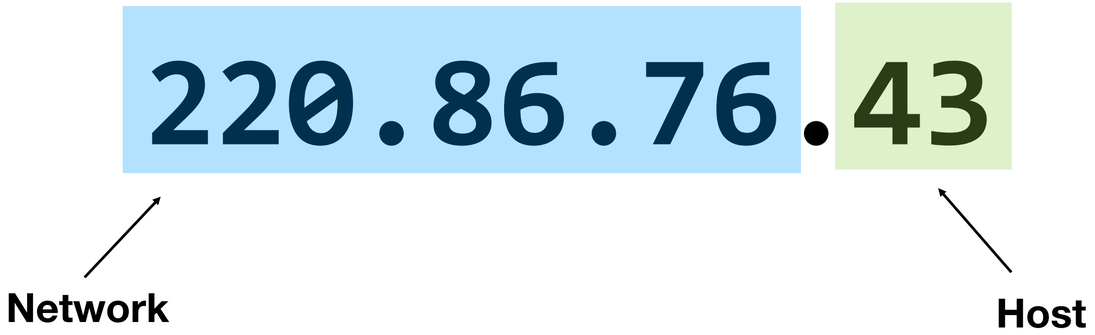




IPv4 Address

1. The first part of an Internet address identifies the network on which the host resides

2. The second part identifies the particular host on the given network



IPv4 Address

1. The first part of an Internet address identifies the network on which the host resides

2. The second part identifies the particular host on the given network

Subnet Mask

255.255.255.0

Network number will be this part

220.86.76.43

Network

Host

CIDR Scheme
(Classless Inter-Domain Routing)

192.168.60.5/24

Indicates that the first 24 bits are network ID

Subnet

192.168.1.4

192.168.1.5

Router

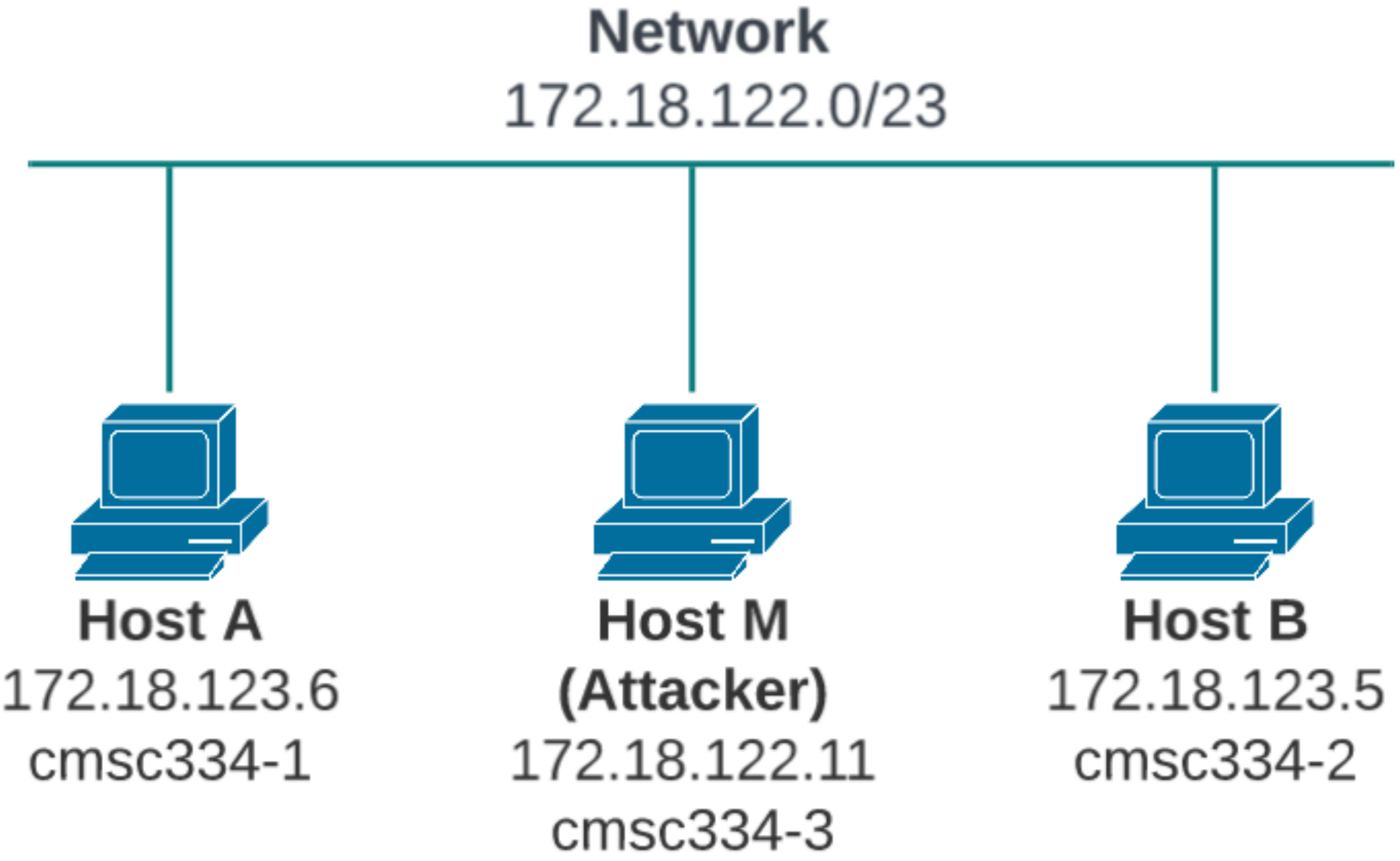
45.233.102.19

192.168.1.3

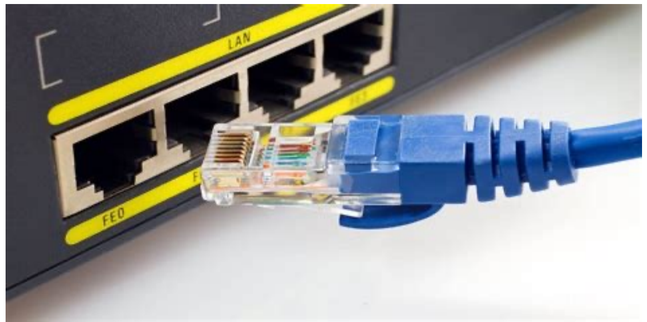
Dynamic Host Configuration Protocol (DHCP)

192.168.1.6

Subnet


Network
172.18.122.0/23
Host A
172.18.123.6
cmsc334-1
Host M
(Attacker)
172.18.122.11
cmsc334-3
Host B
172.18.123.5
cmsc334-2

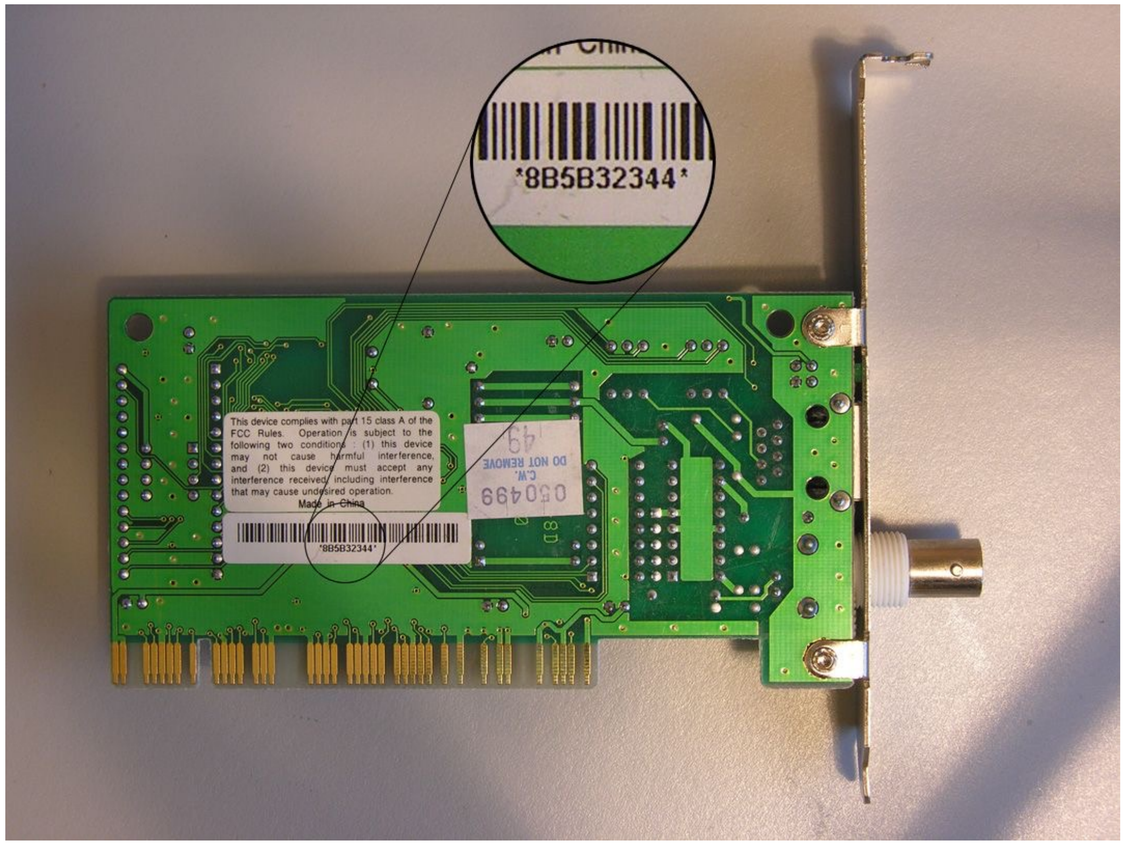
Subnet



MAC Address

- **M**edium **A**ccess **C**ontrol address is a unique identifier assigned to a network interface controller (NIC)

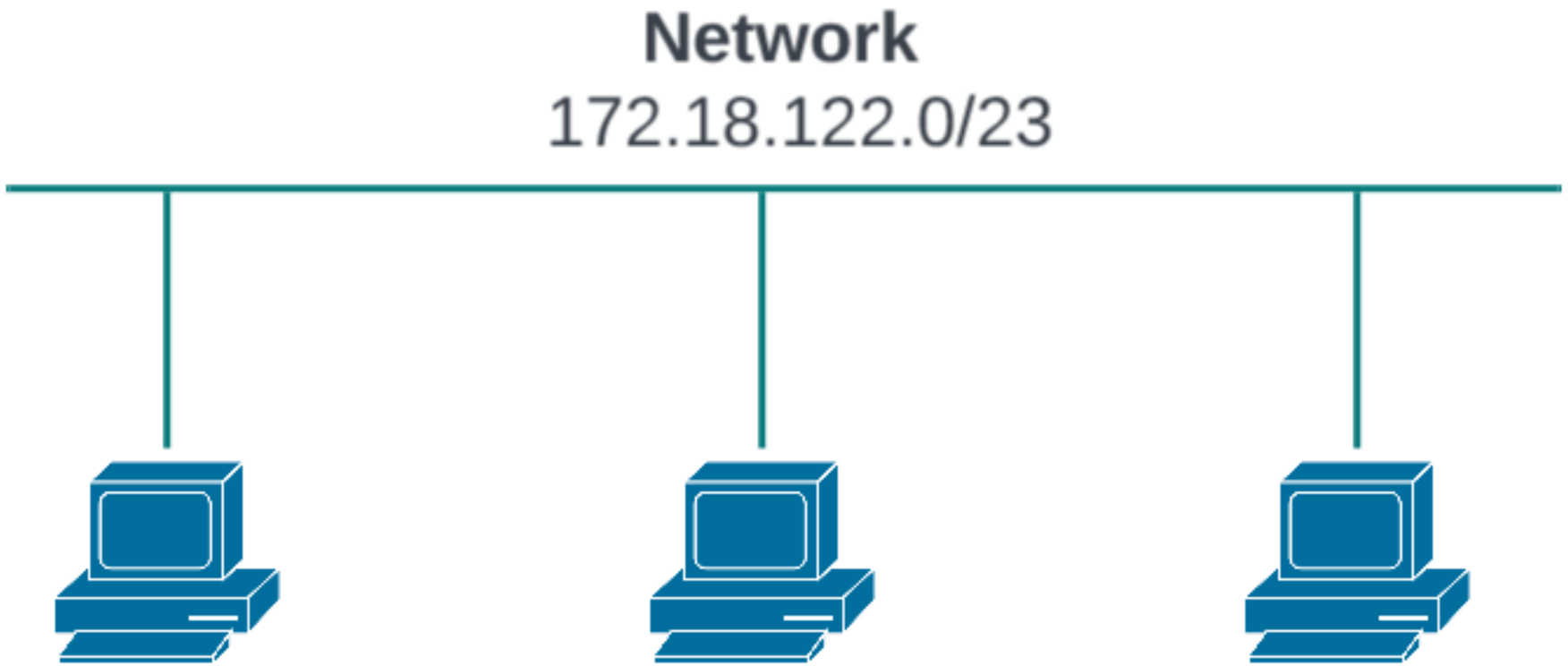
- Assigned by device manufacturers, and are therefore often referred to as the **burned-in address**, or as an **Ethernet hardware address**



Ethernet

```
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.123.6  netmask 255.255.254.0  broadcast 172.18.123.255
        inet6 fe80::ea6a:64ff:fece:4bdf  prefixlen 64  scopeid 0x20<link>
        ether e8:6a:64:ce:4b:df  txqueuelen 1000  (Ethernet)
        RX packets 861161  bytes 323755456 (308.7 MiB)
        RX errors 0  dropped 1  overruns 0  frame 0
        TX packets 1182988  bytes 1312417683 (1.2 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 17  memory 0xb1100000-b1120000
```

Subnet


Network
172.18.122.0/23
Host A
172.18.123.6
cmsc334-1
MAC: e8:6a:64:ce:4b:df
Host M
(Attacker)
172.18.122.11
cmsc334-3
Host B
172.18.123.5
cmsc334-2

Link Layer

Assumes: Local nodes are physically connected

Task: Transfer bytes between two hosts on the physically connected network

Link Layer: Ethernet

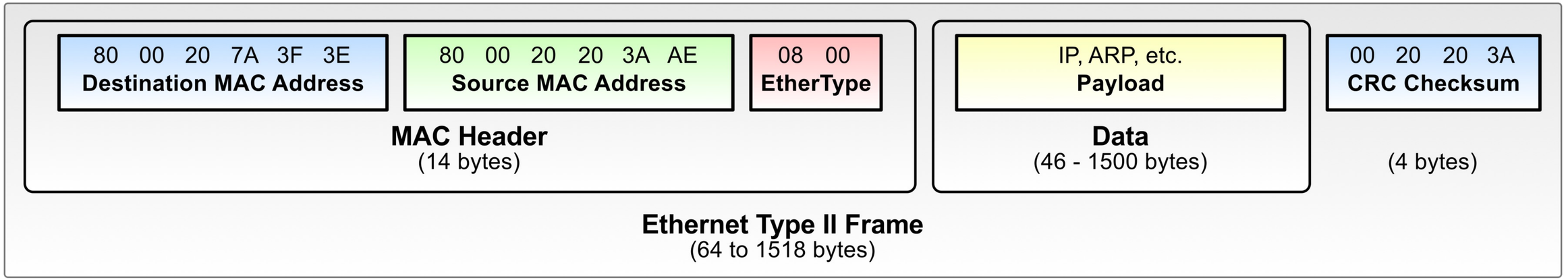
Provides connectivity between hosts on a single *Local Area Network*

Data is split into ~1500 byte **Frames,** which are addressed to a device's 6-byte physical (MAC) address — assigned by manufacturer

No security (confidentiality, authentication, or integrity)

Ethernet

Most common Link Layer Protocol. Let's you send packets to other local hosts.



At layer 2 (link layer) packets are called *frames*

MAC addresses: 6 bytes, universally unique

EtherType gives layer 3 protocol in payload
- 0x0800: IPv4
- 0x0806: ARP
- 0x86DD: IPv6

Problem

- How does a host know what **MAC address** their destination has given an **IP address**?

ARP: Address Resolution Protocol

ARP is a Network protocol that lets hosts map IP addresses to MAC addresses

Host who needs MAC address M corresponding to IP address N broadcasts an ARP packet to LAN asking, “who has IP address N?”

Host that has IP address N will reply, “IP *N* is at MAC address *M*.”

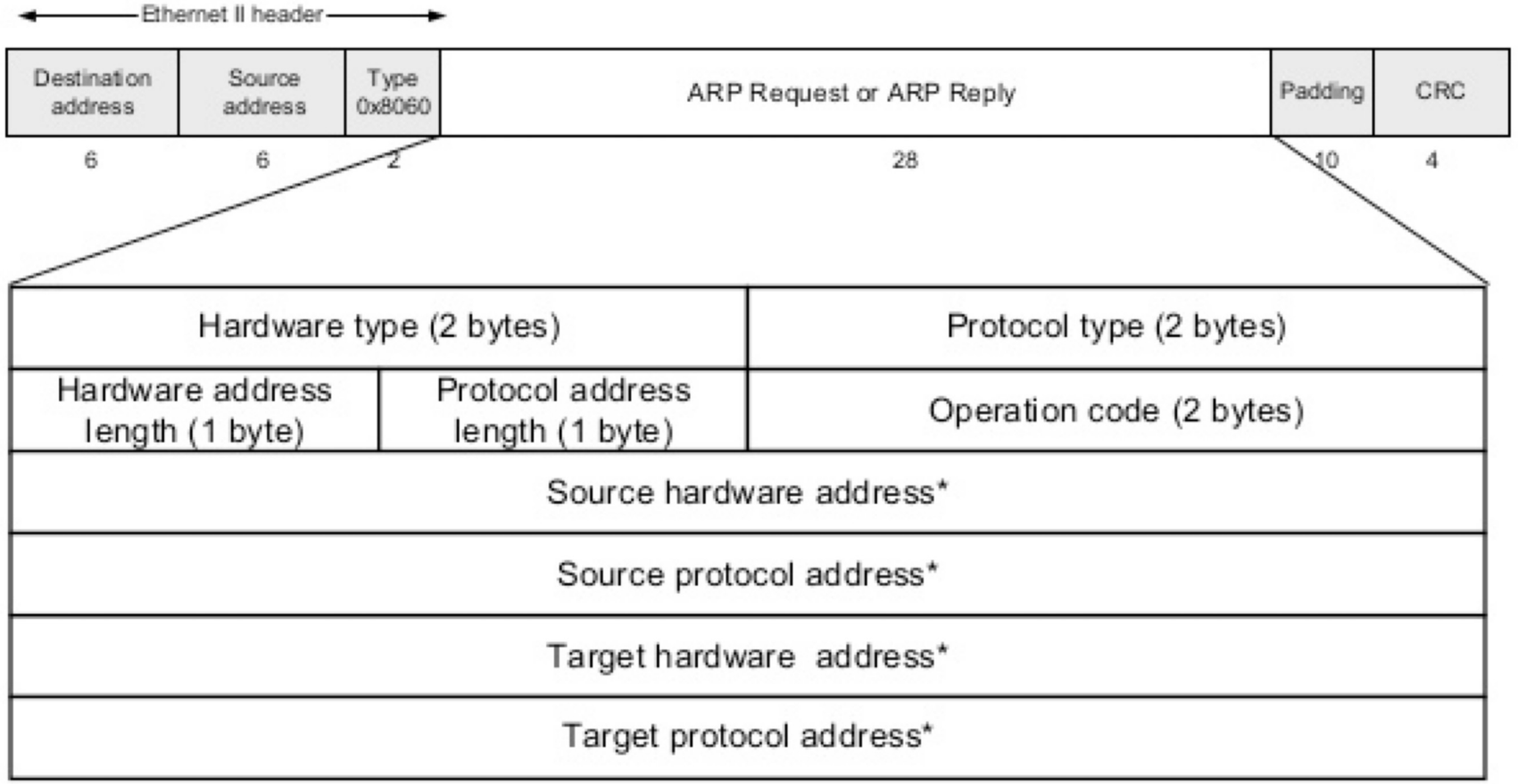
ARP: Address Resolution Protocol

ARP lets hosts to find each others' MAC addresses on a local network. For example, when you need to send packets to the upstream router to reach Internet hosts

Client: Broadcast (all MACs): Which MAC address has IP 192.168.1.1?
Response: I have this IP address (sent from correct MAC)

No built-in security. Attacker can impersonate a host by faking its identity and responding to ARP requests or sending gratuitous ARP announcements

ARP Packet



* Note: The length of the address fields is determined by the corresponding address length fields

ARP Security

Any host on the LAN can send ARP requests and replies: *any host can claim to be another host on the local network!*

This is called *ARP spoofing*

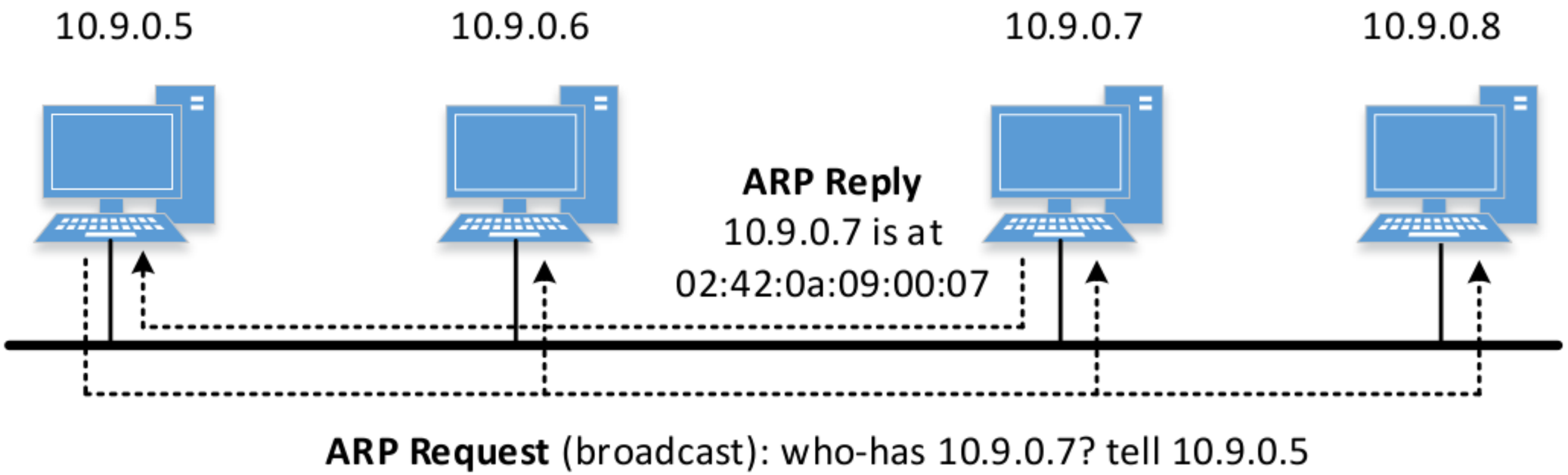
This allows any host X to force IP traffic between any two other hosts A and B to flow through X *(MitM!)*

Claim N_A is at attacker's MAC address M_X

Claim N_B is at attacker's MAC address M_X

Re-send traffic addressed to N_A to M_A, and vice versa

ARP Request/Reply



Send ARP Request: Example 1

ping 10.9.0.6 from 10.9.0.5

```
// On 10.9.0.5
# tcpdump -i eth0 -n
03:10:44.656336 ARP, Request who-has 10.9.0.5 tell 10.9.0.6, ...
03:10:44.656362 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, ...
03:10:44.656382 IP 10.9.0.6 > 10.9.0.5: ICMP echo request, ...
03:10:44.656392 IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, ...
```

ARP Cache

```
# arp -n                                    empty cache
# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.138 ms
...
# arp -n
Address      HWtype  HWaddress           Flags Mask  Iface
10.9.0.6     ether   02:42:0a:09:00:06   C           eth0
                                  ↖ Cached MAC address
```

ARP Cache Poisoning

- Spoof ARP Messages
  - Request
  - Reply
  - Gratuitous message
- Spoofed message might be cached by the victim
  - Which type of message will be cached depends on OS implementation

Constructing ARP Message

Construct ARP packet

```
#!/usr/bin/python3

from scapy.all import *

E = Ether()
A = ARP()

pkt = E/A
sendp(pkt)
```

Fields of ARP and Ether Class

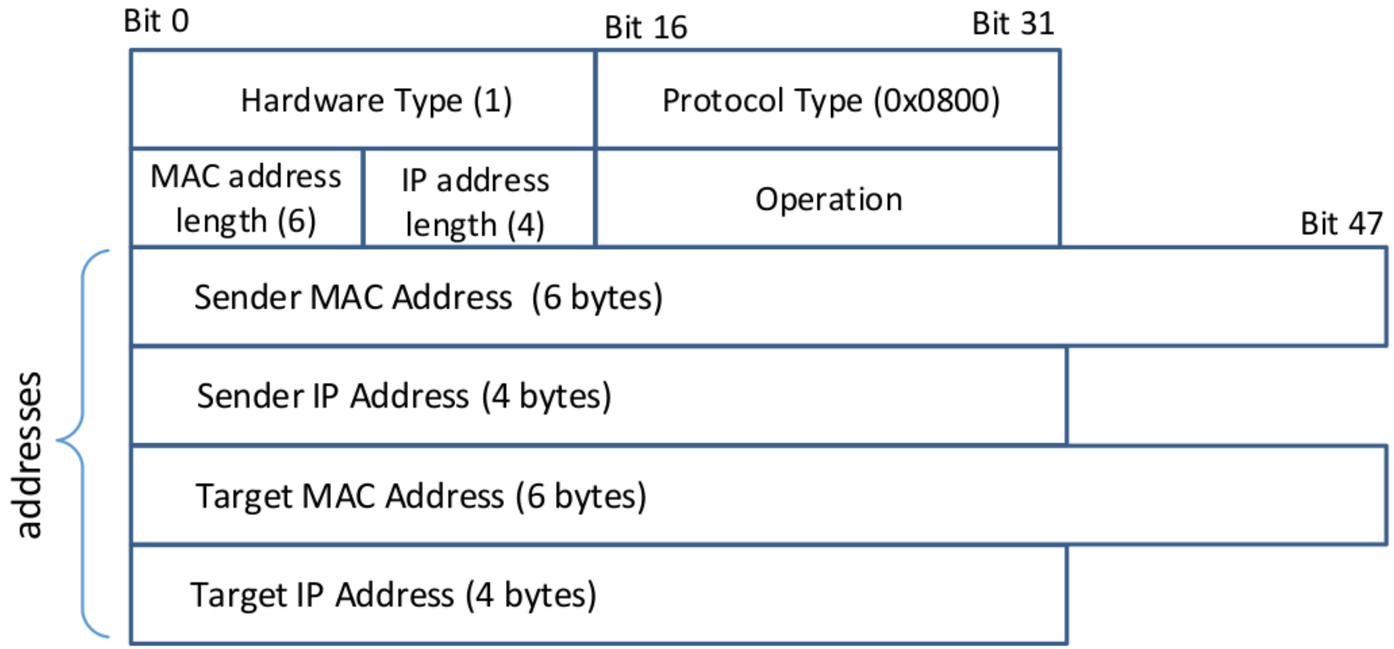
```
>>> ls(ARP)
hwtype     : XShortField                          = (1)
ptype      : XShortEnumField                      = (2048)
hwlen      : FieldLenField                        = (None)
plen       : FieldLenField                        = (None)
op         : ShortEnumField                       = (1)
hwsrc      : MultipleTypeField                    = (None)
psrc       : MultipleTypeField                    = (None)
hwdst      : MultipleTypeField                    = (None)
pdst       : MultipleTypeField                    = (None)
>>> ls(Ether)
dst        : DestMACField                         = (None)
src        : SourceMACField                       = (None)
type       : XShortEnumField                      = (36864)
```

Spoof ARP Request/Reply: Code Skeleton

```
target_IP    = "10.9.0.5"
target_MAC   = "02:42:0a:09:00:05"

fake_IP      = "10.9.0.99"
fake_MAC     = "aa:bb:cc:dd:ee:ff"

# Construct the Ether header
ether = Ether()
ether.dst =
ether.src =

# Construct the ARP packet
arp = ARP()

arp.hwsrc =
arp.psrc  =

arp.hwdst =
arp.pdst  =

arp.op = 1

frame = ether/arp
sendp(frame)
```

victim: 10.9.0.5

goal: map 10.9.0.99 to aa:bb:cc:dd:ee:ff



Spoofing Gratuitous Message

- Special type of ARP message

```
IP_fake = "10.9.0.99"
ether = Ether(src="aa:bb:cc:dd:ee:ff", dst="ff:ff:ff:ff:ff:ff")
arp   = ARP(psrc=IP_fake, hwsrc="aa:bb:cc:dd:ee:ff",
            pdst=IP_fake, hwdst="ff:ff:ff:ff:ff:ff")
arp.op = 2
```