# Network Security

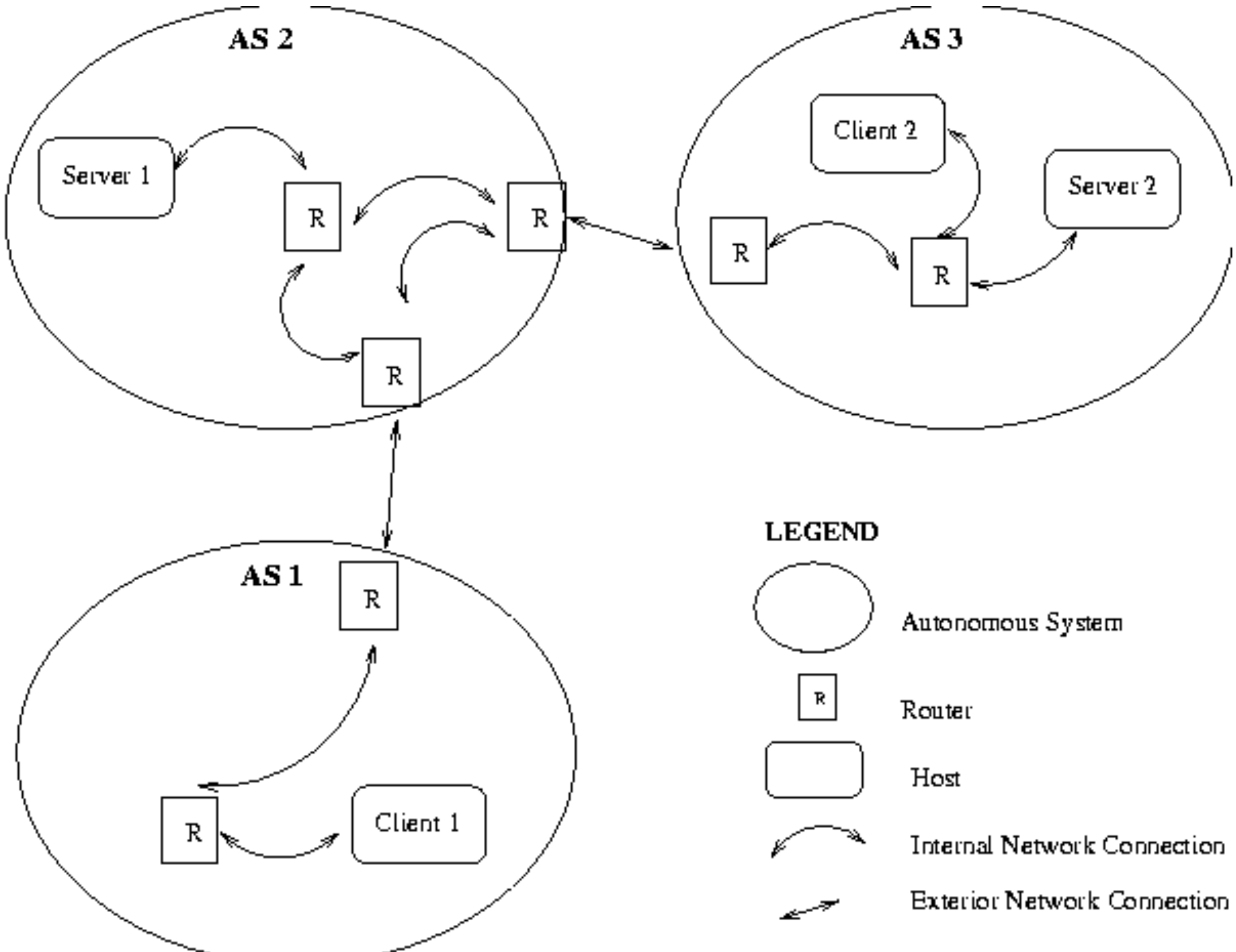
AA 2020/2021

Network aspects

# Internet communication

- Internet is made of several logically separated networks → **Autonomous Systems** (AS)
  - Internet= network of networks
- Each AS autonomously manages communications within itself
  - Interior Gateway Protocols (IGP) → route within each AS
  - e.g. two commonly used IGPs are them Routing Information Protocol (RIP) and the Open Shortest Path First (OSPF) protocol.
- Each AS can communicate to other AS
  - Exterior Gateways Protocols → route between ASs
    - Border Gateway Protocol

# Internet autonomous systems




https://www.usenix.org/legacy/event/lisa98/full_papers/pultar/pultar_html/pultar.html

# OSI model

DATA
Examples:
HTTP / Mail / Chat protocols / econding info / RPC / Telnet

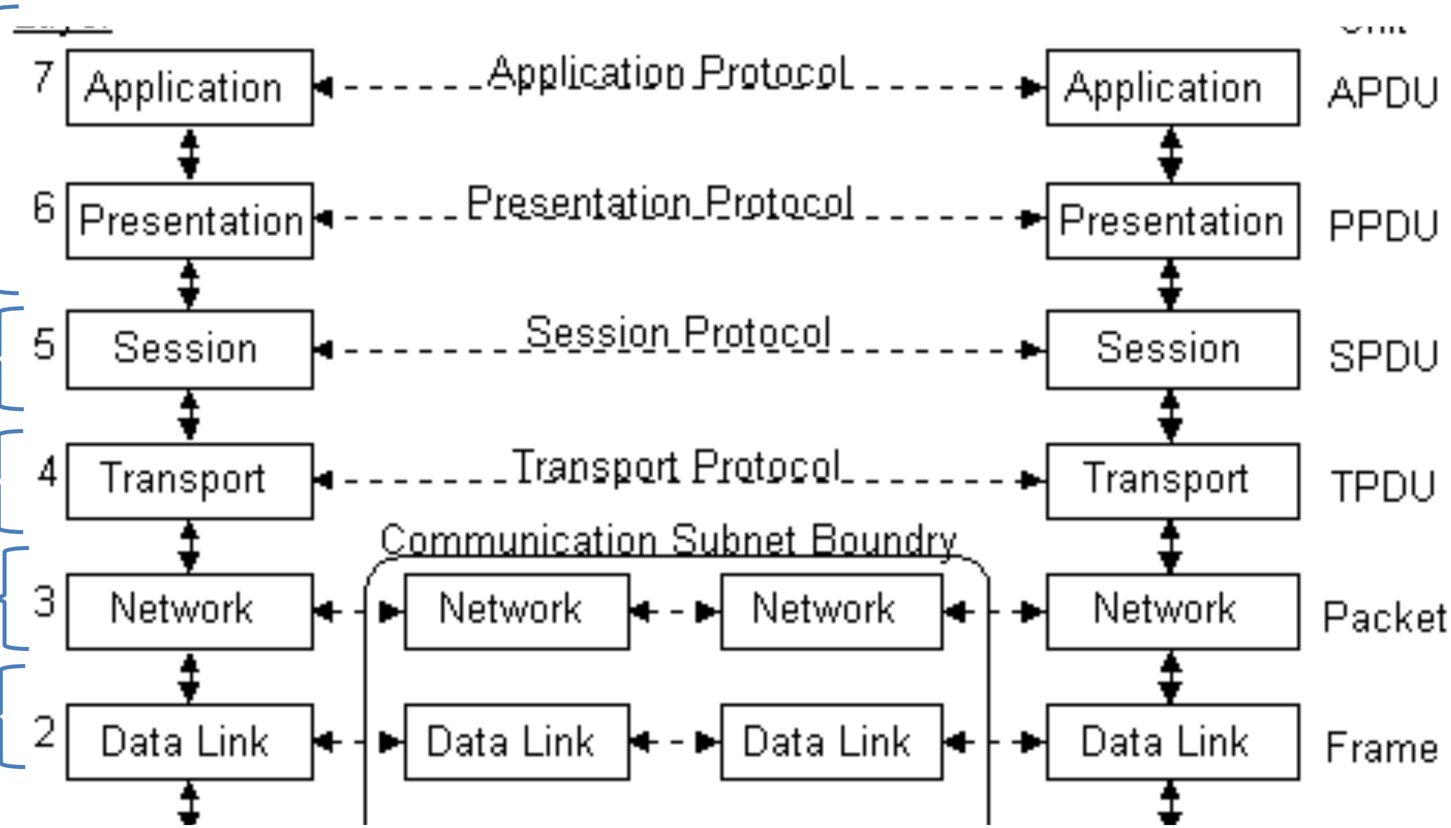
Session
Examples: TLS, SSL

SEGMENTS or Datagrams
Examples: TCP, UDP

Packet
Examples: IPv4, IPv6

Frame
Examples: Ethernet, PPP

Bit
Examples: ethernet cable / optical fiber



# OSI DATA LINK LAYER

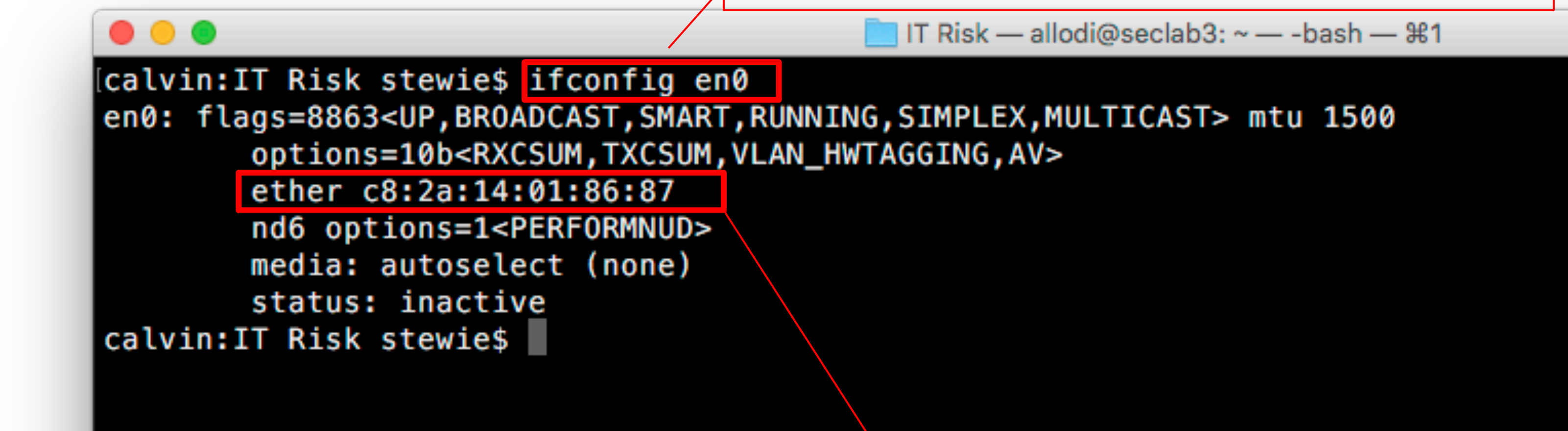
# Data link layer

- Lowest “logical” level
- Data link interconnects physical interfaces
- Each physical interface is identified by a MAC address
  - “Ethernet address”
  - 48-bit Network interface **identifiers**
  - Closest representation of final destination of a frame
  - HEX notation
    - HH-HH-HH-HH-HH-HH
  - Used to route packets in local networks

# Mac addresses

- Uniquely identify a network interface
- Assigned by the producer according to the standard IEEE 802

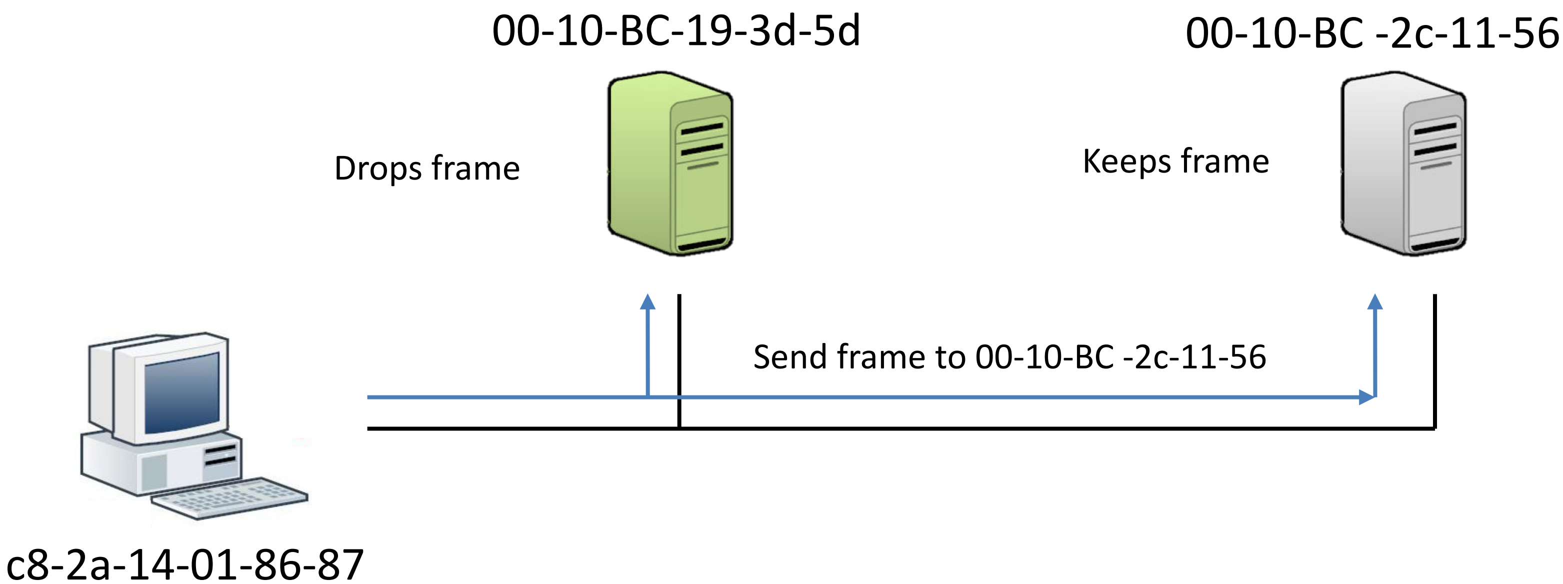
ifconfig: *unix system command to list net interfaces
“ipconfig” on windows machines
en0: name of interface




Mac address of interface “en0”

# Mac addresses example

- First 24 bit are set by IEEE standard
- Identify network interface producer
  - 00-10-BC → Aastra Telecom
  - https://regauth.standards.ieee.org/standards-ra-web/pub/view.html#registries



# OSI NETWORK LAYER

# The Network Layer

- Provides information on how to reach other systems
  - Addressing functionalities
- IP operates at this layer
  - High-level representation of a host's addresses
  - Conveys information to route the datagram
  - IPv4 defined in RFC 791
- Most IP addresses are dynamically assigned by an authority (e.g. ISP's DHCP server)
  - As opposed to MAC addresses that are fixed by the vendor
  - "**Connectionless**" protocol (stateless)
    - No notion of "established connection" at this stage
    - Only provides the means necessary for a **packet** to reach its destination

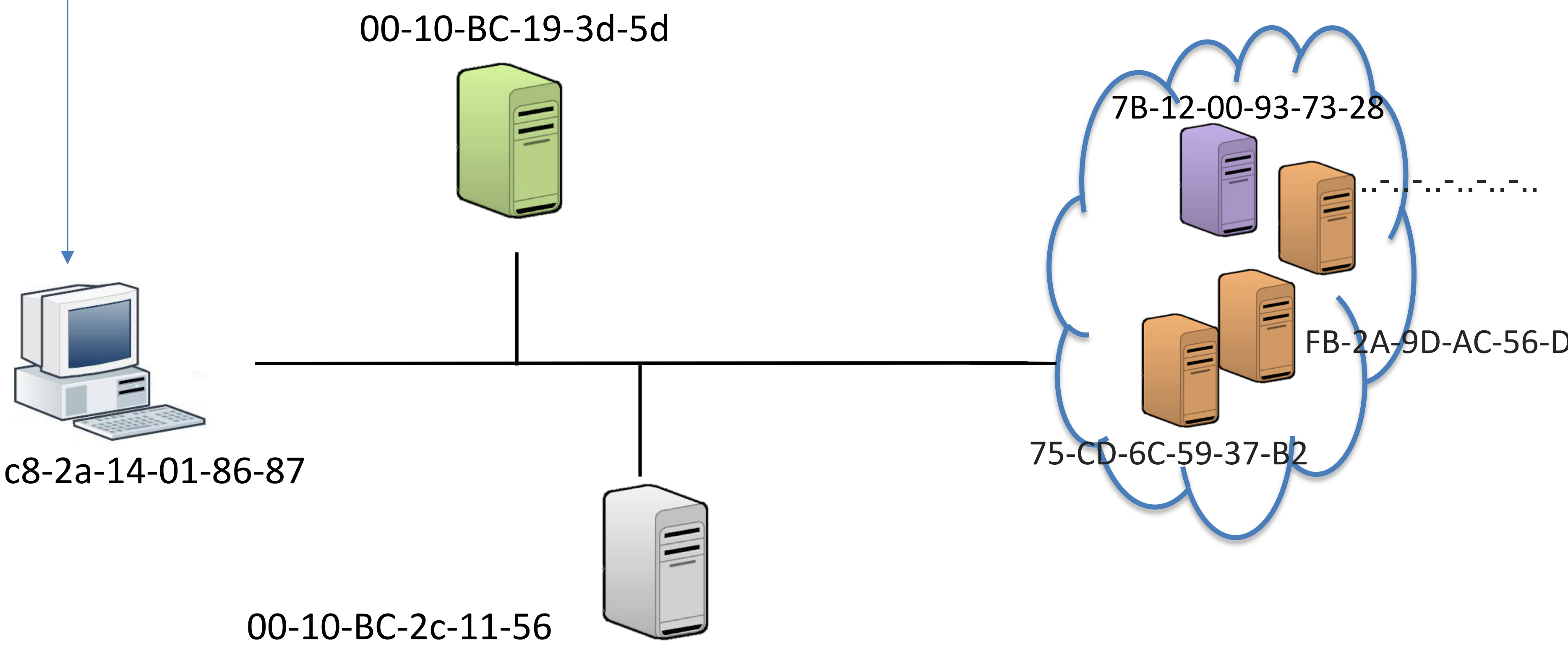
# stateless vs stateful

- A communication is made of a number of messagges
- Communications start, develop, and ends
  - Stateful protocols provide means to establish and close a connection
    - e.g. TCP
  - Stateless protocols do not have this notion
    - IP messages are stand-alone packets

# IP vs MAC addresses

48 bit → $2^{48}$ addresses = 281474976710656 → 1536 terabyte

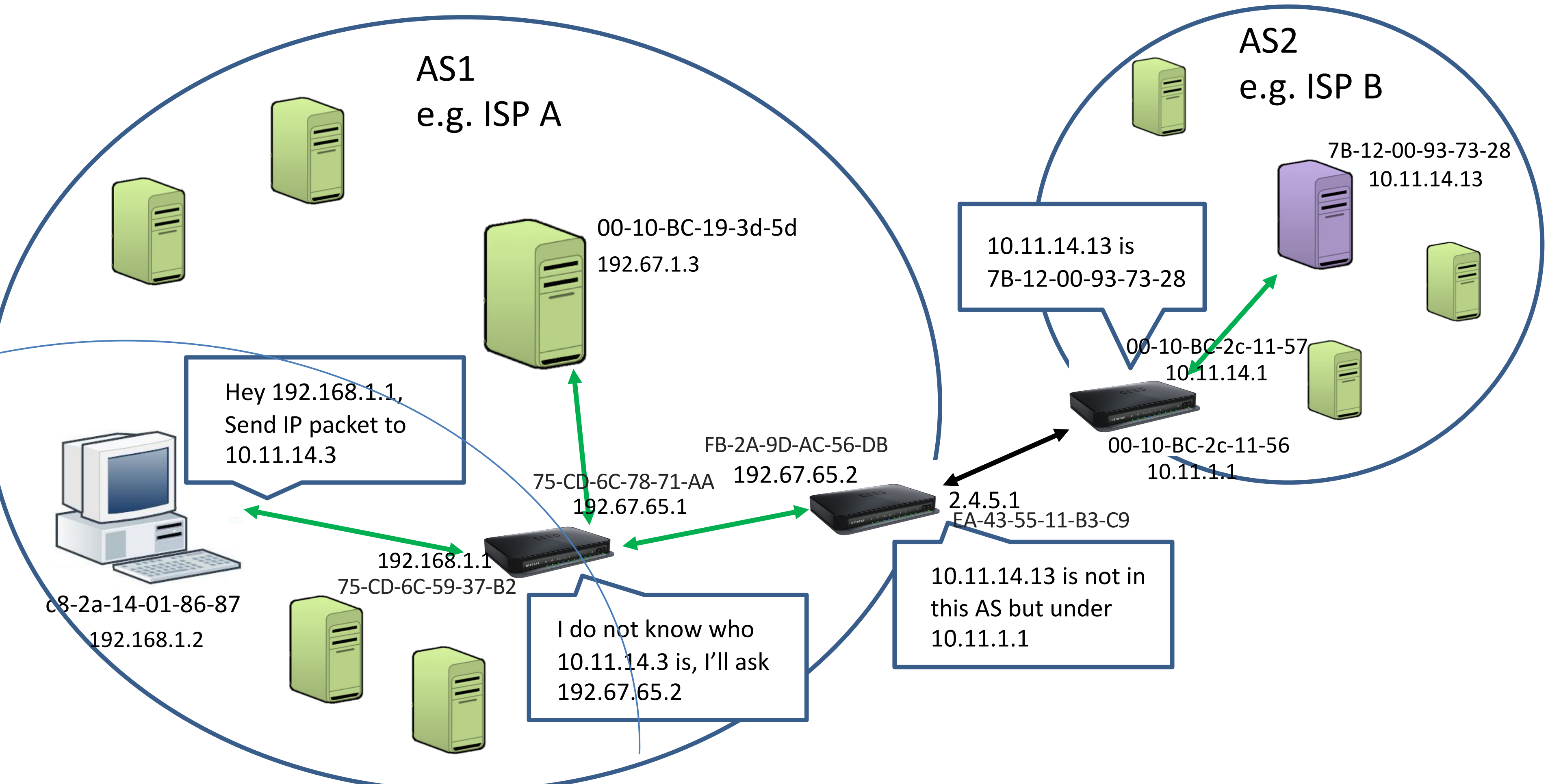
- How to manage revoking? (e.g. One ethernet card gets substituted)
- How to manage routing?



# IP addresses

- IP provides a structured way to abstract host addresses away from their physical properties
- Two versions
  - IPv4 → most common, currently used
    - 32 bits
  - IPv6 → early adoption, will be seen commonly in the future
    - 128 bits
- Make it possible to efficiently talk between systems in different AS

# IP addresses – routing (simplified)




Details: http://disi.unitn.it/locigno/index.php/teaching-duties/computer-networks/102-reti-aa13-14

“And they of a man’s own household will be his worst enemy."

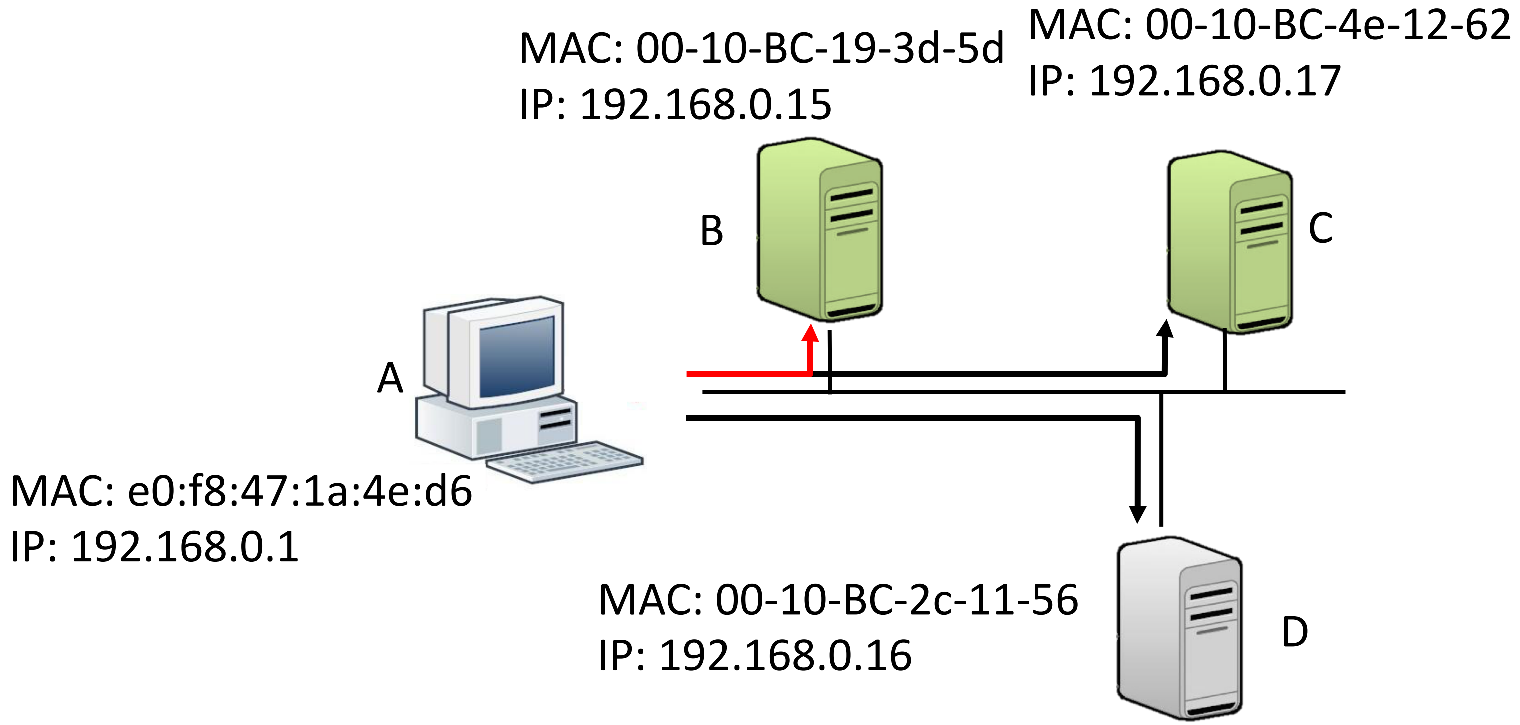
# ARP protocol

- ARP = address resolution protocol
  - Allows systems to associate an IP address to a MAC address
  - Allows discovery through broadcast
- ARP tables contain information to translate IP addresses into MAC addresses

```
calvin:IT Risk stewie$ arp -i en1 -a
? (10.196.192.1) at 3c:94:d5:48:25:c1 on en1 ifscope [ethernet]
? (10.196.192.14) at d0:25:98:90:ee:95 on en1 ifscope [ethernet]
? (10.196.192.246) at 2c:1f:23:4f:84:a4 on en1 ifscope [ethernet]
? (10.196.193.5) at 74:e5:b:20:b7:e8 on en1 ifscope [ethernet]
? (10.196.193.162) at 34:36:3b:d4:90:44 on en1 ifscope [ethernet]
? (10.196.193.178) at 4c:25:78:78:f1:e8 on en1 ifscope [ethernet]
? (10.196.193.230) at 94:65:9c:31:55:dd on en1 ifscope [ethernet]
? (10.196.194.52) at 48:50:73:60:3c:1c on en1 ifscope [ethernet]
? (10.196.194.53) at 78:31:c1:c9:81:24 on en1 ifscope [ethernet]
? (10.196.194.223) at 28:5a:eb:17:19:3f on en1 ifscope [ethernet]
? (10.196.195.63) at 64:76:ba:b3:eb:12 on en1 ifscope [ethernet]
```

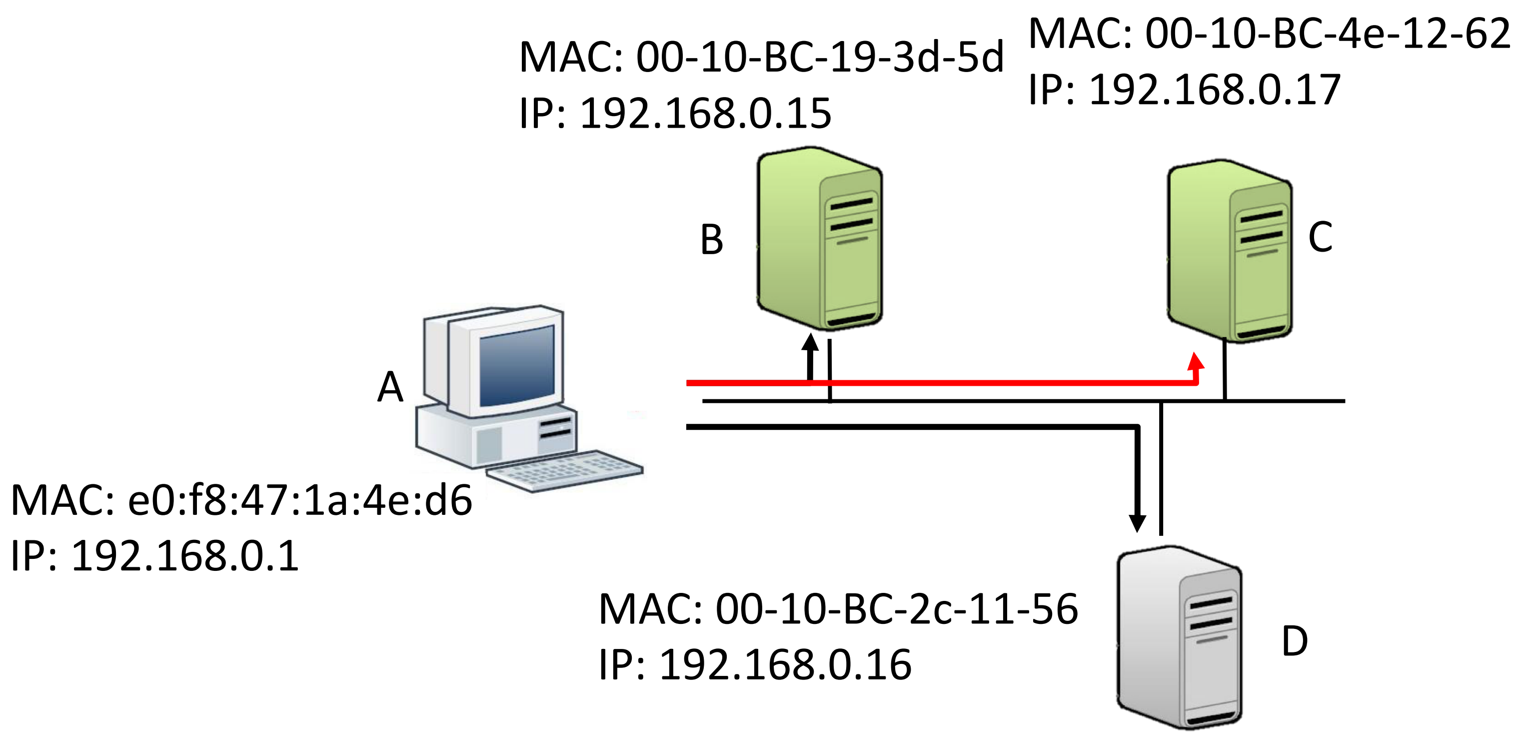
# ARP tables A→B

| IP address | MAC address | … (e.g. TTL, interfaces..) |
|---|---|---|
| 192.168.0.15 | 00-10-BC-19-3d-5d | … |
| 192.168.0.17 | 00-10-BC-4e-12-62 | … |
| | | |



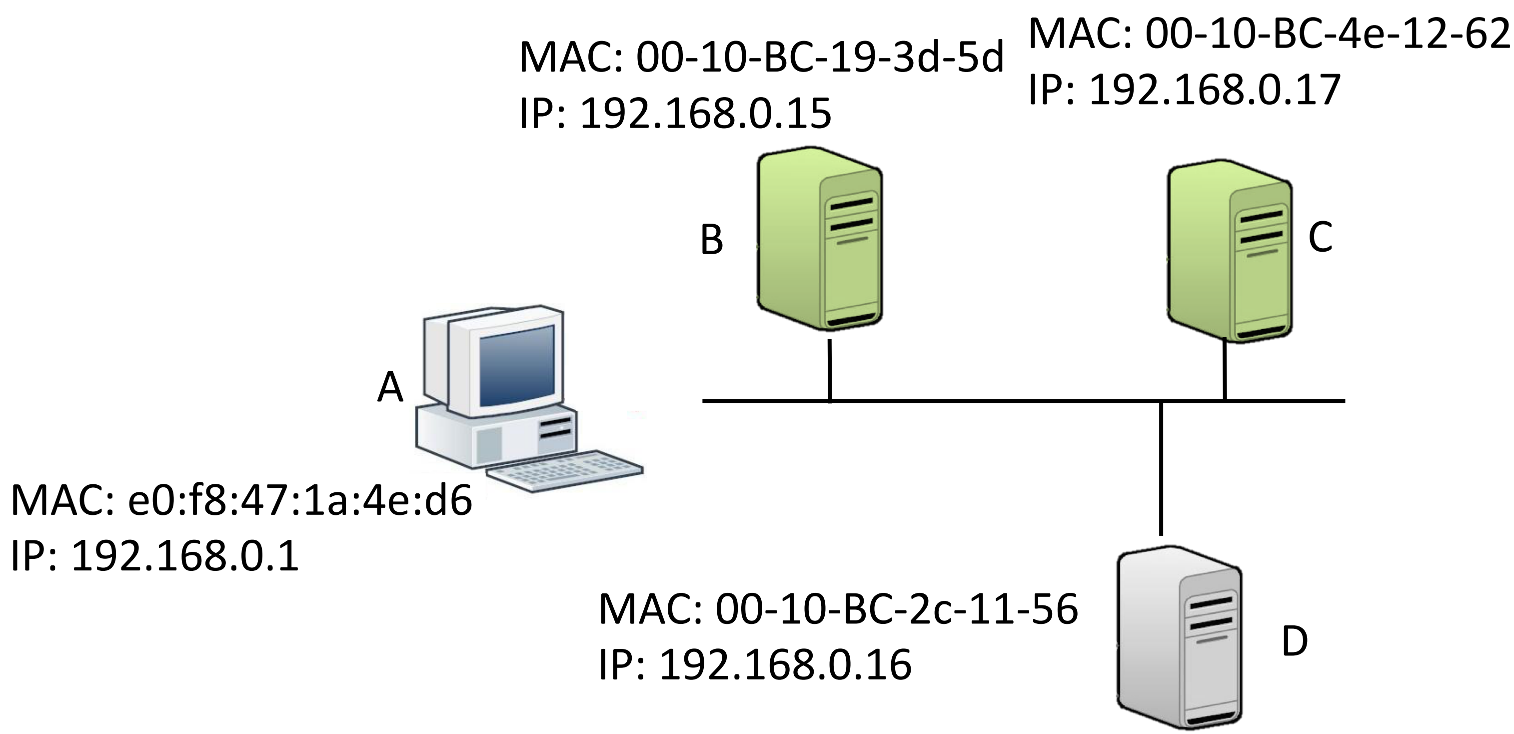
# ARP tables A→C

| IP address | MAC address | … (e.g. TTL, interfaces..) |
| --- | --- | --- |
| 192.168.0.15 | 00-10-BC-19-3d-5d | … |
| 192.168.0.17 | 00-10-BC-4e-12-62 | … |
| | | |



# ARP tables A→D

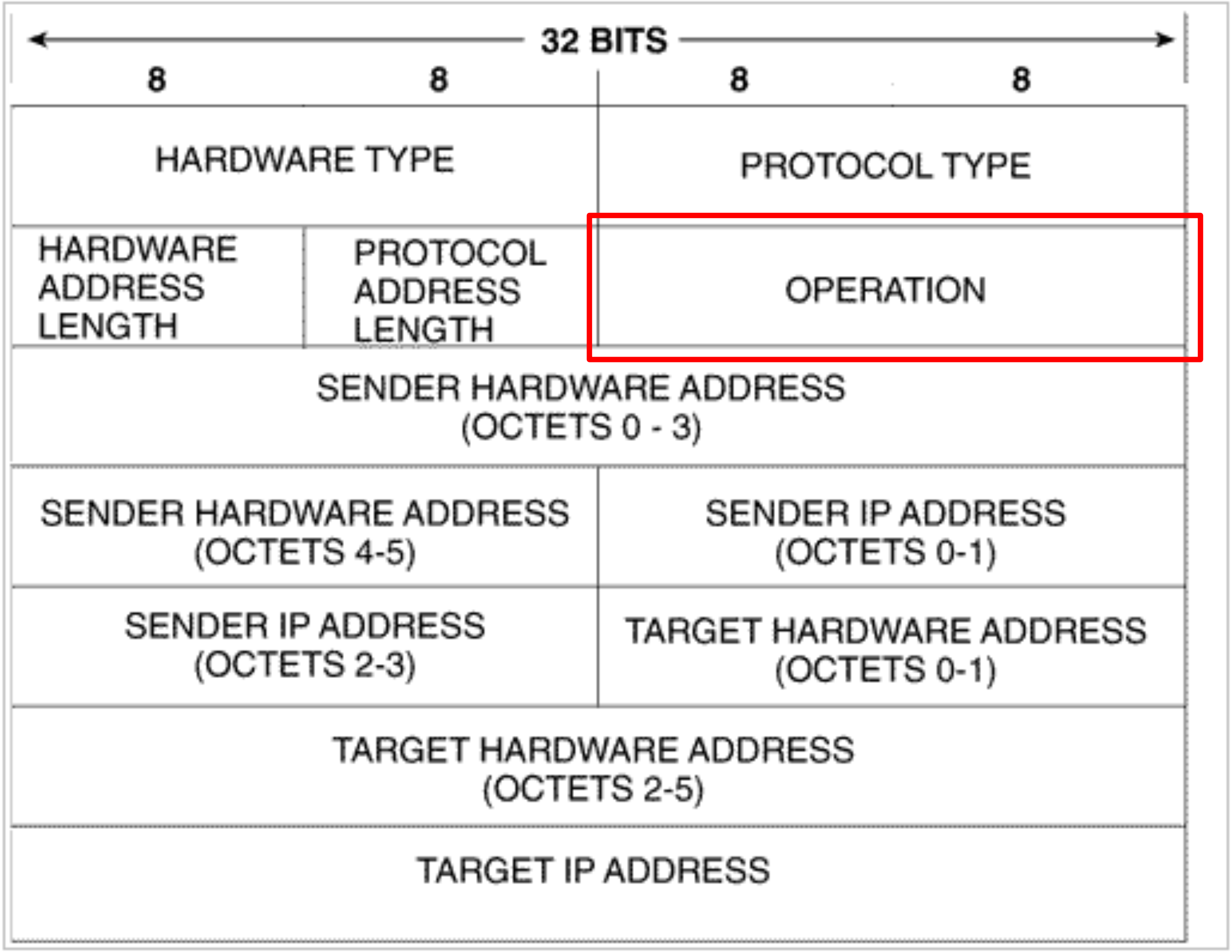
| IP address | MAC address | … (e.g. TTL, interfaces..) |
|---|---|---|
| 192.168.0.15 | 00-10-BC-19-3d-5d | … |
| 192.168.0.17 | 00-10-BC-4e-12-62 | … |
| ??? | | |



# ARP query

- All addresses in an ARP table are added by one of two mechanisms
  - ARP request-reply
    - → **who is** 192.168.0.16 **tells** 192.168.0.1
    - → 192.168.0.16 **is at** 00-10-BC-2c-11-56
  - Gratuitous ARP
    - → 192.168.0.16 **is at** 00-10-BC-2c-11-56
- The discovery process happens through queries to neighbor devices
  - Broadcast message to the desired IP
    - L2 ethernet address FF-FF-FF-FF-FF-FF
- The system with the requested IP replies back with its correct mac address

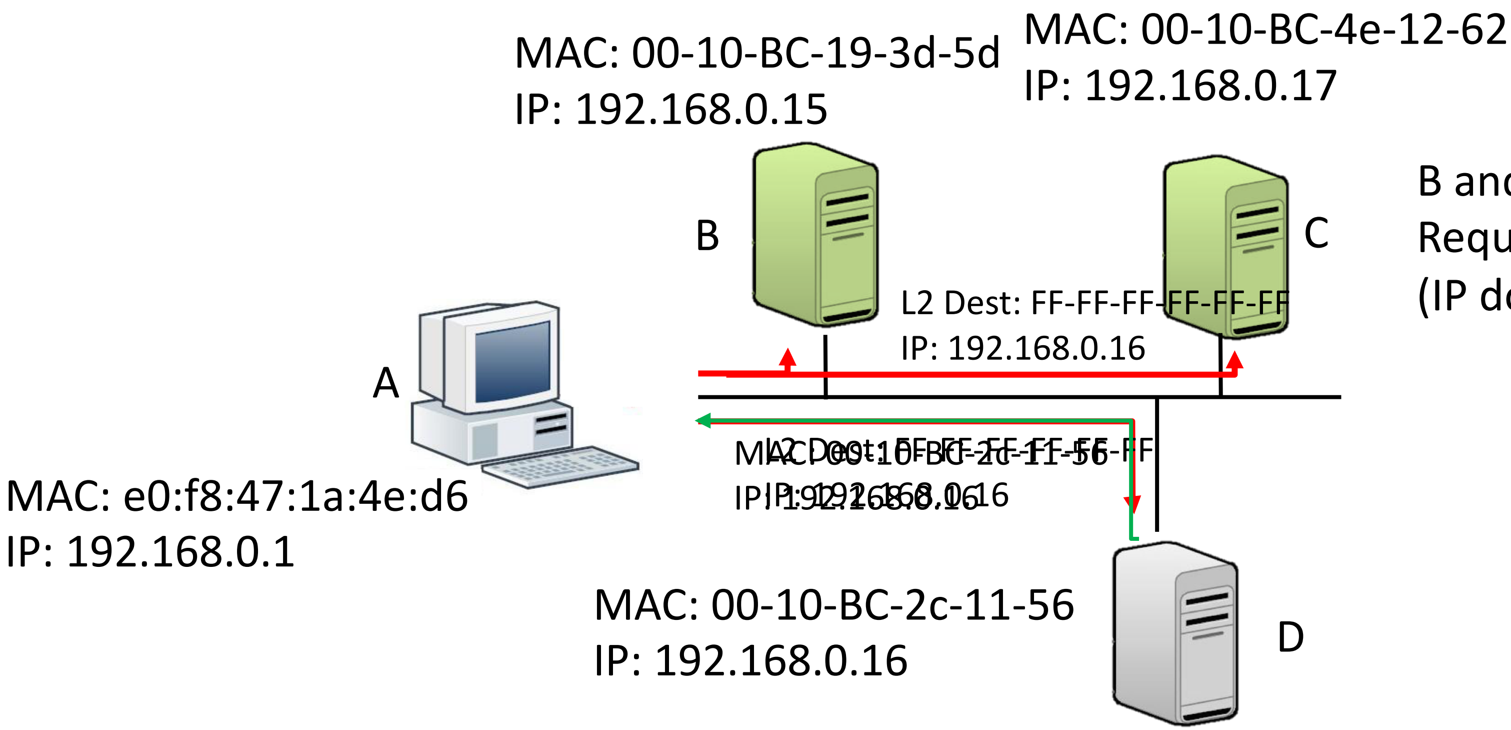
# ARP frame header




1=request
2=reply

# ARP tables A→D

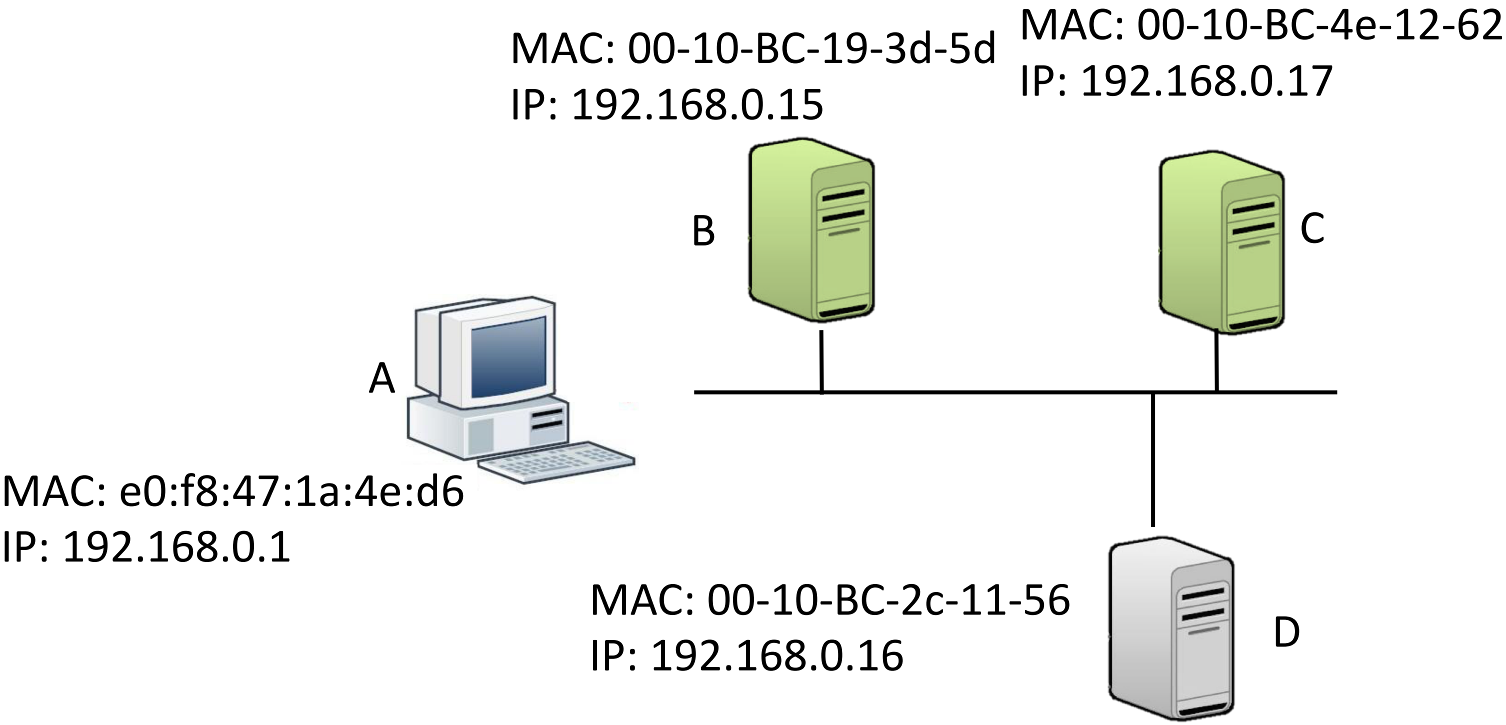
| IP address | MAC address | … (e.g. TTL, interfaces..) |
|---|---|---|
| 192.168.0.15 | 00-10-BC-19-3d-5d | … |
| 192.168.0.17 | 00-10-BC-4e-12-62 | … |
| | | |




B and C drop Request (IP does not match)

# ARP tables A→D

| IP address | MAC address | … (e.g. TTL, interfaces..) |
|---|---|---|
| 192.168.0.15 | 00-10-BC-19-3d-5d | … |
| 192.168.0.17 | 00-10-BC-4e-12-62 | … |
| 192.168.0.16 | 00-10-BC-2c-11-56 | |



# Example of ARP request-reply

request

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 9 | 5.008920000 | CadmusCo_0a:a1:1 | RealtekU_12:35:02 | ARP | 42 | Who has 10.0.2.2? Tell 10.0.2.15 |
| 10 | 5.009100000 | RealtekU_12:35:0 | CadmusCo_0a:a1:14 | ARP | 60 | 10.0.2.2 is at 52:54:00:12:35:02 |

```
▸ Frame 9: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: CadmusCo_0a:a1:14 (08:00:27:0a:a1:14), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: CadmusCo_0a:a1:14 (08:00:27:0a:a1:14)
    Sender IP address: 10.0.2.15 (10.0.2.15)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.0.2.2 (10.0.2.2)
```

reply

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 9 | 5.008920000 | CadmusCo_0a:a1:1 | RealtekU_12:35:02 | ARP | 42 | Who has 10.0.2.2? Tell 10.0.2.15 |
| 10 | 5.009100000 | RealtekU_12:35:0 | CadmusCo_0a:a1:14 | ARP | 60 | 10.0.2.2 is at 52:54:00:12:35:02 |

```
▸ Frame 10: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▸ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: CadmusCo_0a:a1:14 (08:00:27:0a:a1:14)
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: RealtekU_12:35:02 (52:54:00:12:35:02)
    Sender IP address: 10.0.2.2 (10.0.2.2)
    Target MAC address: CadmusCo_0a:a1:14 (08:00:27:0a:a1:14)
    Target IP address: 10.0.2.15 (10.0.2.15)
```

# ARP broadcast example

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 2 | 0.99[illegible]964000 | CadmusCo_0a:a1:1 | Broadcast | ARP | 42 | Who has 10.0.2.5? Tell 10.0.2.15 |
| 3 | 1.997994000 | CadmusCo_0a:a1:1 | Broadcast | ARP | 42 | Who has 10.0.2.5? Tell 10.0.2.15 |
| 4 | 3.017323000 | CadmusCo_0a:a1:1 | Broadcast | ARP | 42 | Who has 10.0.2.5? Tell 10.0.2.15 |
| 5 | 4.014031000 | CadmusCo_0a:a1:1 | Broadcast | ARP | 42 | Who has 10.0.2.5? Tell 10.0.2.15 |

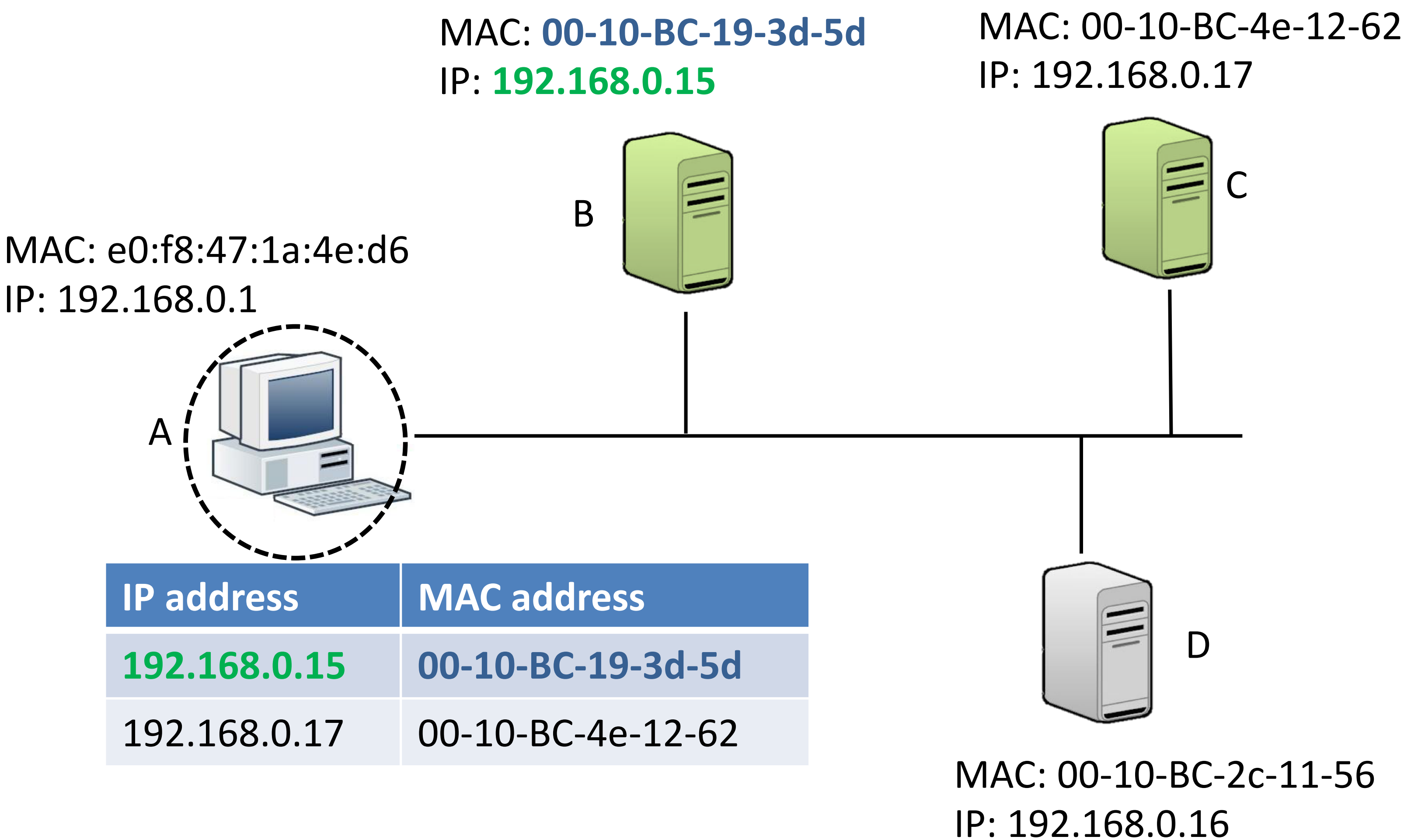
```
▸ Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▾ Ethernet II, Src: CadmusCo_0a:a1:14 (08:00:27:0a:a1:14), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▸ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▸ Source: CadmusCo_0a:a1:14 (08:00:27:0a:a1:14)
    Type: ARP (0x0806)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: CadmusCo_0a:a1:14 (08:00:27:0a:a1:14)
    Sender IP address: 10.0.2.15 (10.0.2.15)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.0.2.5 (10.0.2.5)
```

L2 dest

# ARP poisoning

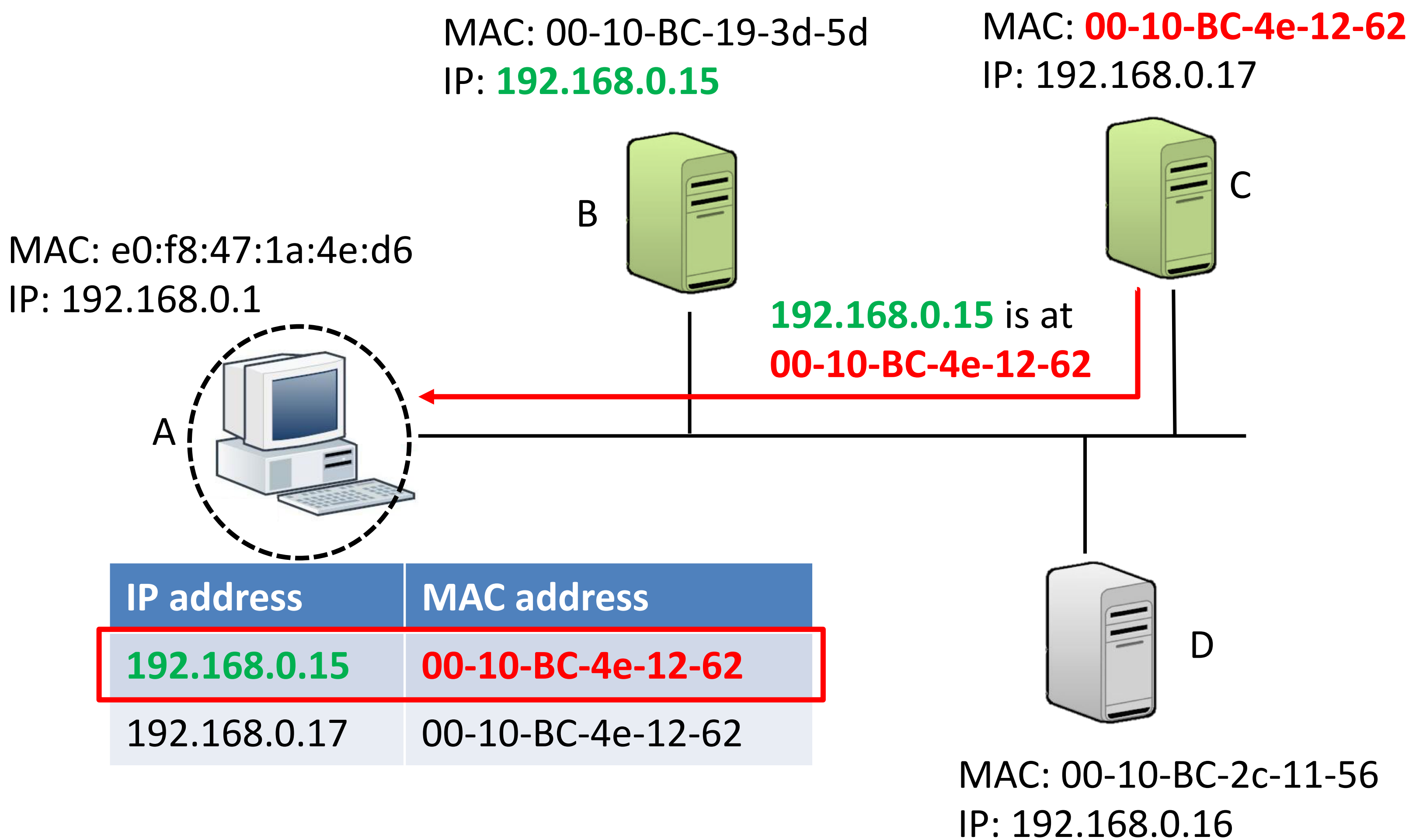
- ARP answers or Gratuitous ARP frames do not require an (additional) answer/confirmation
  - It's a *declarative* protocol
- Nodes are not authenticated
  - Whomever can say I am x.x1.x2.x3, my mac address is hh.hh1.hh2.hh3.hh4.hh5
- C can tell B "D is at [C mac address]"
- C can tell D "B is at [C mac address]"
- As a result every communication between B and D will pass by C

# ARP poisoning



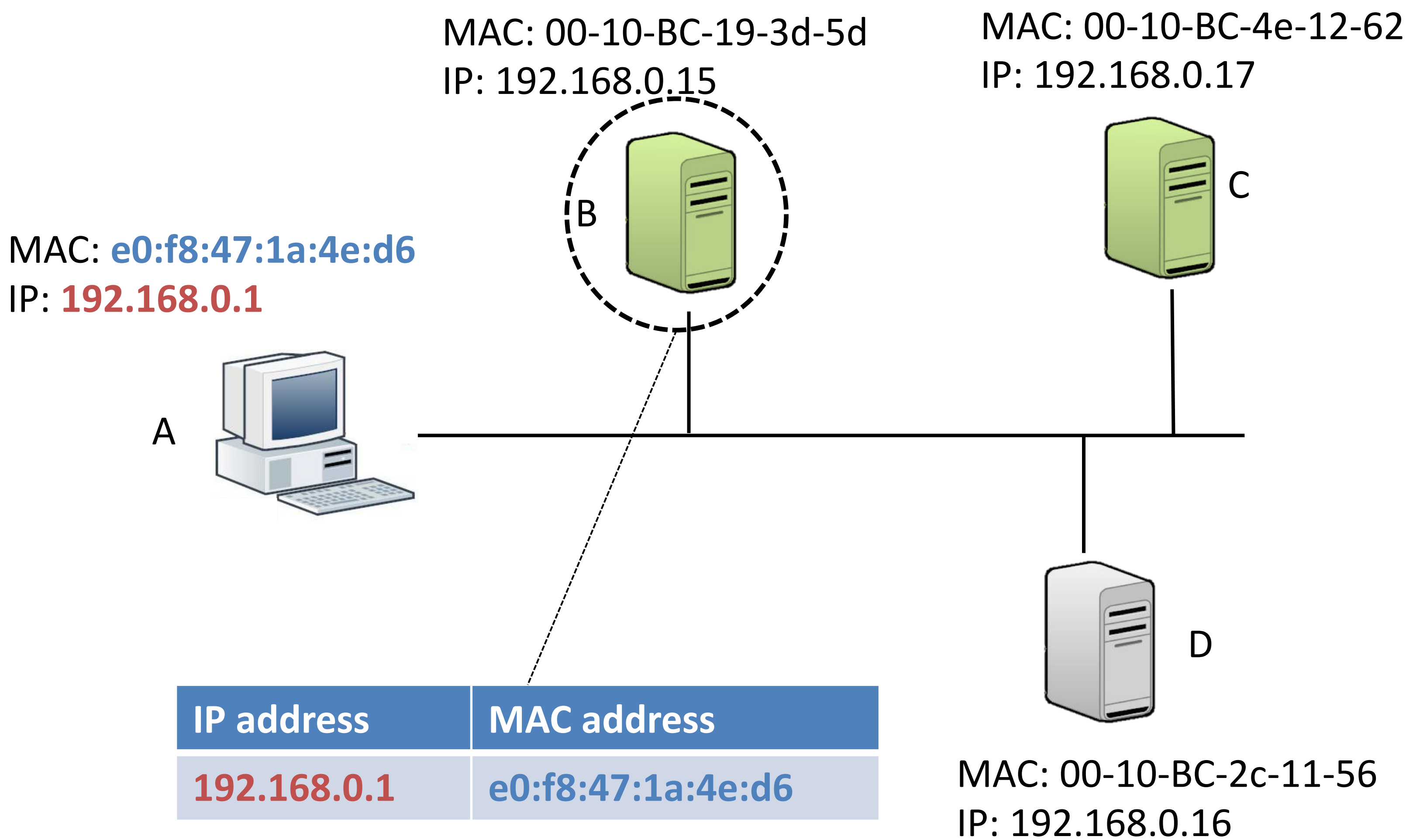

| IP address | MAC address |
|---|---|
| **192.168.0.15** | **00-10-BC-19-3d-5d** |
| 192.168.0.17 | 00-10-BC-4e-12-62 |

# ARP poisoning



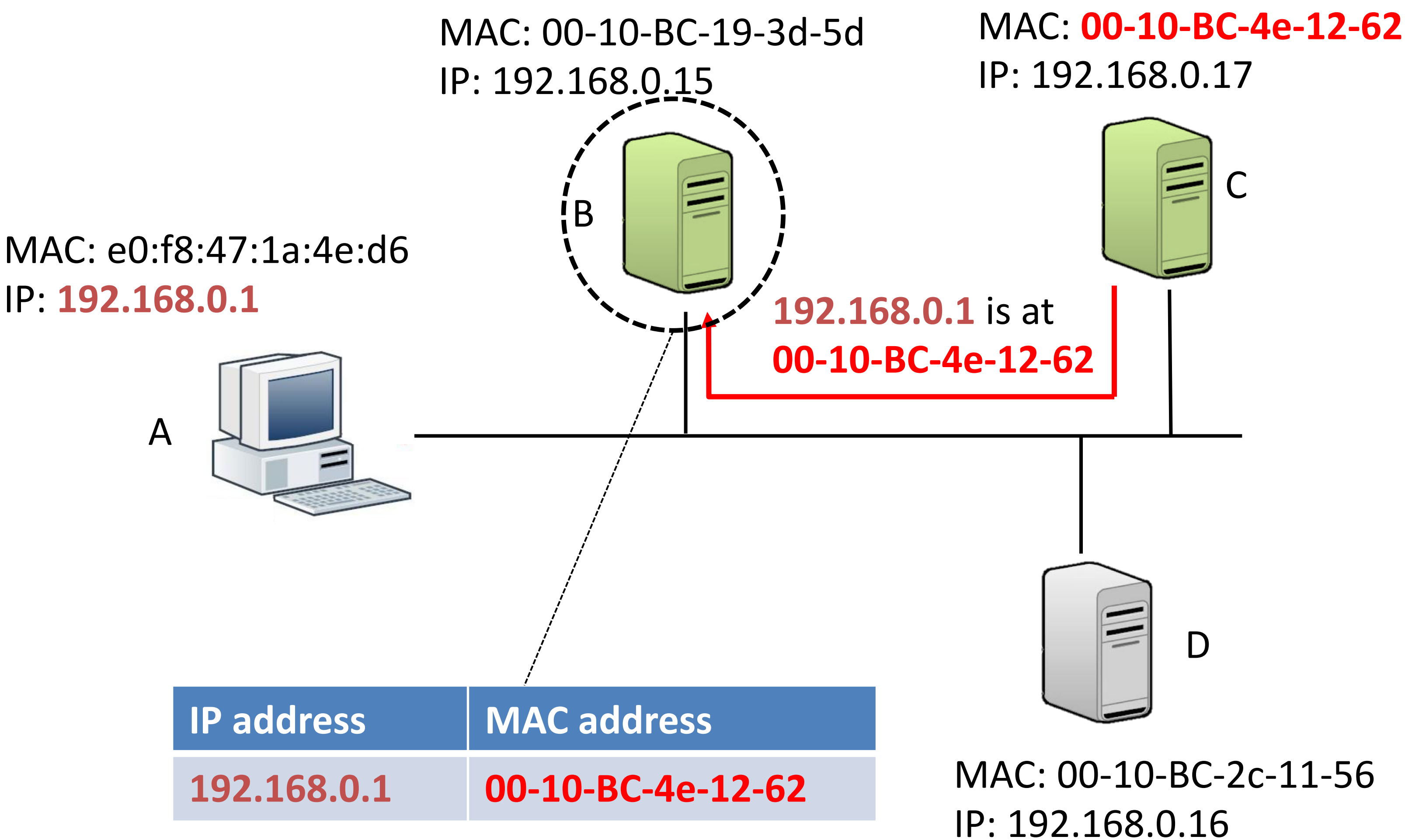

| IP address | MAC address |
|---|---|
| 192.168.0.15 | 00-10-BC-4e-12-62 |
| 192.168.0.17 | 00-10-BC-4e-12-62 |

# ARP poisoning



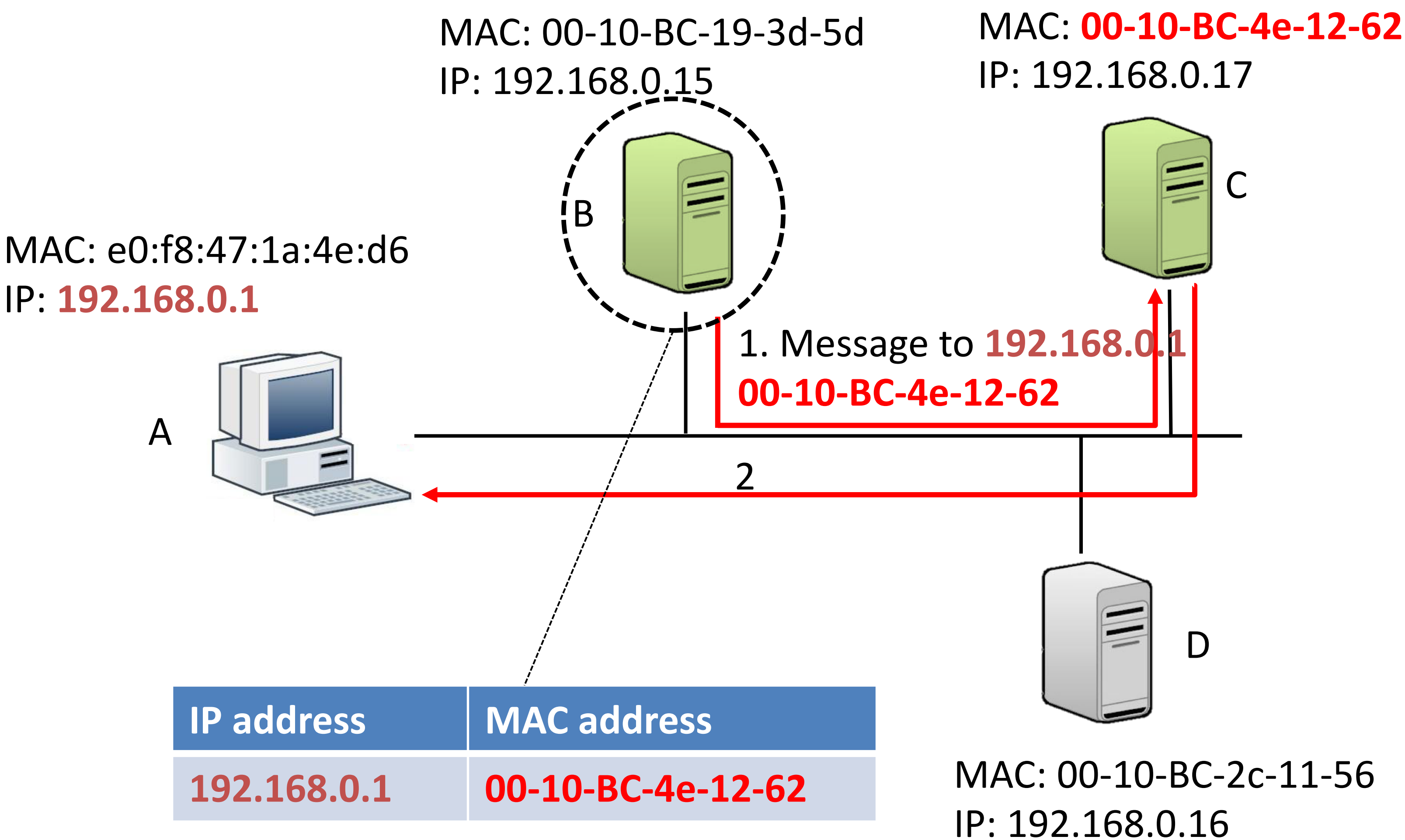

| IP address | MAC address |
|---|---|
| 192.168.0.1 | e0:f8:47:1a:4e:d6 |

# ARP poisoning



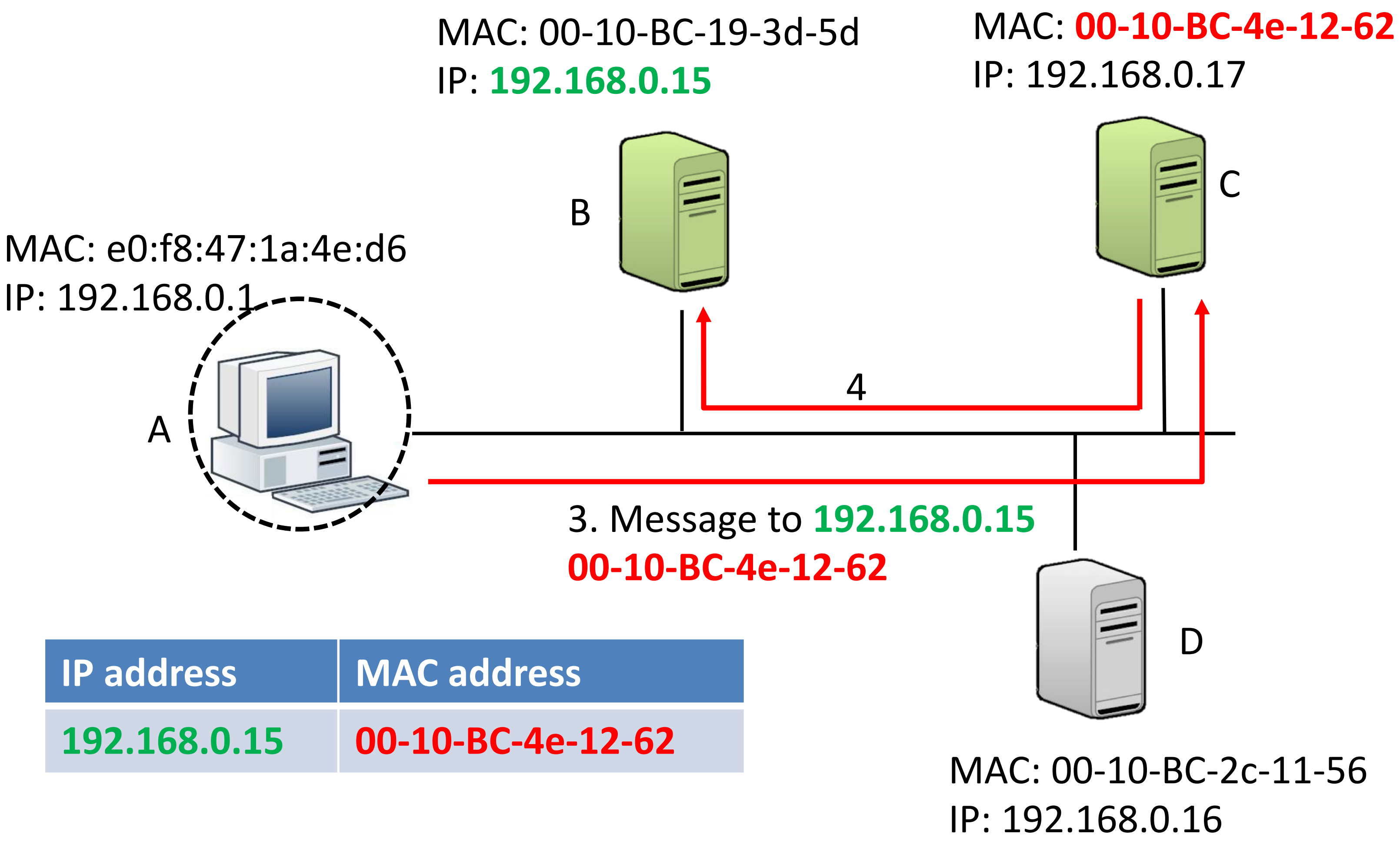

| IP address | MAC address |
|---|---|
| 192.168.0.1 | 00-10-BC-4e-12-62 |

# ARP poisoning




| IP address | MAC address |
|---|---|
| 192.168.0.1 | 00-10-BC-4e-12-62 |

# ARP poisoning




| IP address | MAC address |
|---|---|
| 192.168.0.15 | 00-10-BC-4e-12-62 |

# ARP poisoning - limitations

- Works only on local networks, where MAC addresses are actually meaningful
  - When communication is targeted to different network, IP addresses are used
- Routers and DNSs have MAC addresses too..
- The poisoning works because systems are not authenticated
  - Some implementations/third party tools can mitigate the problem
    - Check for anomalies
  - Can you think of a possible mitigation?

# IP Header

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

# Subnets and CIDR

- Subnets are logical divisions of IP addresses
  - Possible to split a network in multiple sub-networks
- IP bits are divided in
  - x network bits
  - y subnet bits
  - z host bits
- Subnet mask indicates sections of IP addresses meant for network+subnet
  - 255.255.255.0 → 24 bits to network+subnet, 8 bits to hosts
- CIDR → synthetic way to represent subnet masks
  - **Classless Inter-Domain Routing**
  - Indicates number of bits covered by the mask
  - 192.168.10.1/24 = 192.168.10.1/255.255.255.0

# Subnet example

| | NETWORK | SUBNET | HOST |
|---|---|---|---|
| binary | 10000100 | 10000110 | 00001111 01100000 |
| decimal | 132 | 134 | 15 96 |

- Ip address → 132.134.15.96
- Network mask?
  - 255.255.0.0
- CIDR representation?
  - 132.134.15.96/16
- How many hosts?
  - $2^{16}$ -1 = 65,536 - 1

# Subnet example

- Ip address → 132.134.15.96

| | NETWORK | SUBNET | HOST |
|---|---|---|---|
| Binary IP | 10000100 | 10000110 | 00001111 01100000 |
| Binary Subnet mask | 11111111 | 11111111 | 00000000 00000000 |
| Network= IP AND Subnet | 10000100 | 10000110 | 00000000 00000000 |
| Host=IP AND complement(subnet) | 00000000 | 00000000 | 00001111 01100000 |

# IP classes

- IPv4 has several classes
- Defined over
  - Range of IP
  - Number of referenceable hosts
- Classes:
  - A : 0.0.0.0/8 → 127.255.255.255/8
  - B: 128.0.0.0/16 → 191.255.255.255/16
  - C: 192.0.0.0/24 → 223.255.255.255/24
  - D: 224.0.0.0 → 239.255.255.255
  - E: 240.0.0.0 → 254.255.255.254

Standard communications

Multicast

Experimental

# IP addresses – private addresses

- Some IPs are reserved for private networks
  - 10.0.0.0 → 10.255.255.255
  - 192.168.0.0 → 192.168.255.255
  - 172.16.0.0 → 172.31.255.255

- These should not be routed on the internet
  – Gateway should **drop** the datagram

# IP fragmentation (datagram size > MTU)

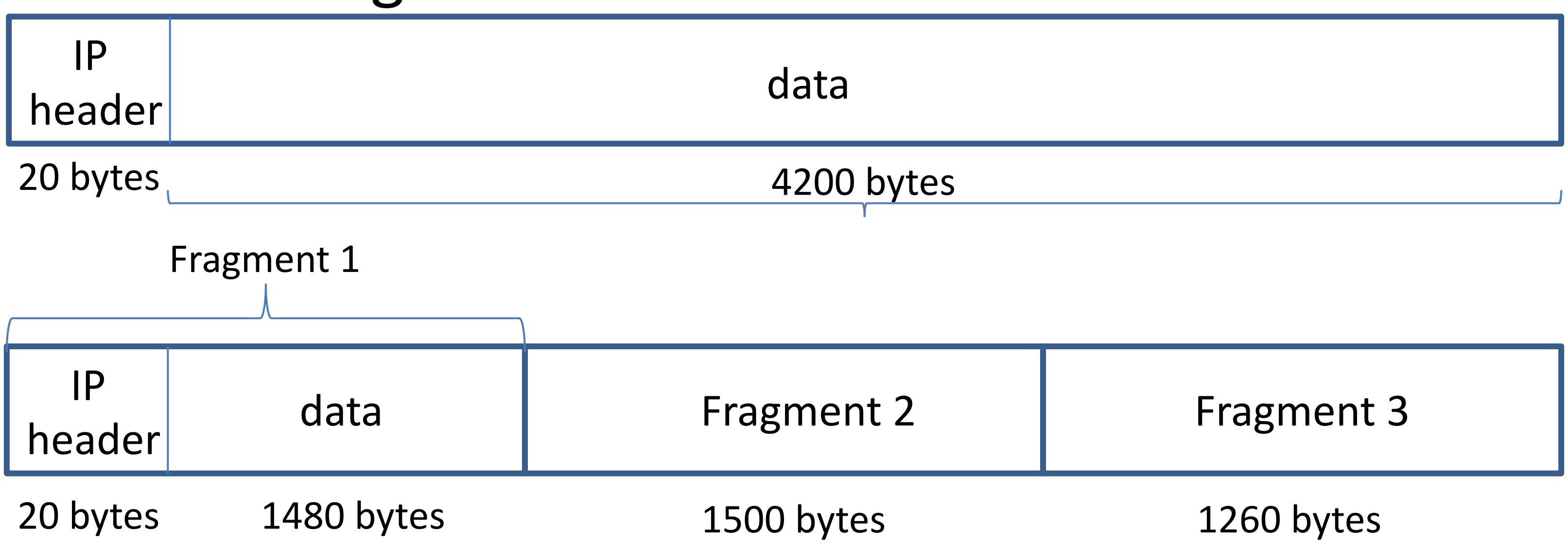
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

# IP Fragments

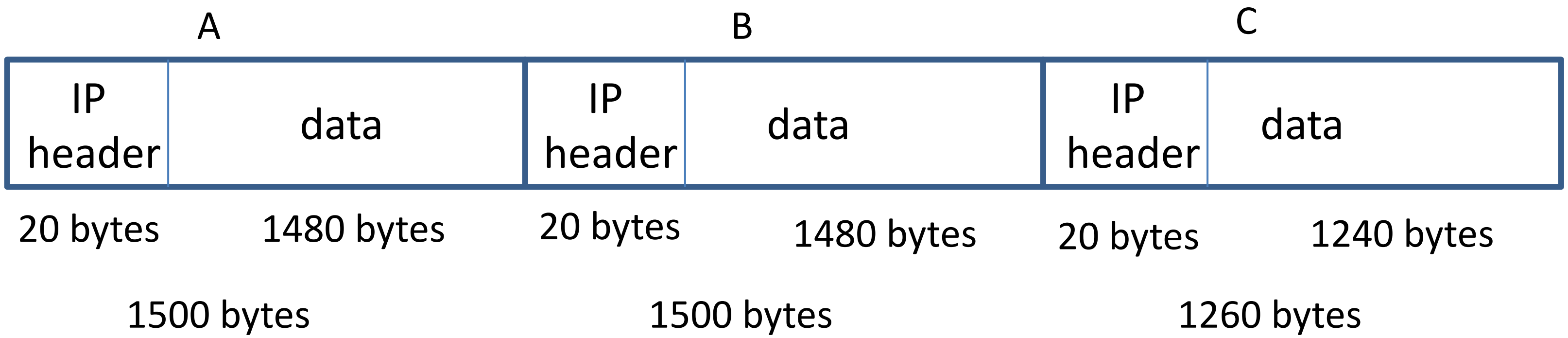
- **Identification**, 16 bit: unique identifier of the fragmented datagrams
  – All fragments have the same identification number
- **Flags**, 3 bit
  – 0 → Reserved, must be 0
  – DF → Don't fragment
    - 0 = there may be fragments
    - 1 = don't fragment. If must be fragmented, drop datagram
  – MF → More fragments
    - 0 = last fragment
    - 1 = there are more fragments
- **Offset**, 13 bits: offset of this datagram w.r.t first fragment with that ID.

# Fragmentation example

- Need to send a 4200 bytes of data over IP
  - Maximum Transmission Unit on ethernet channel is 1500 bytes
  - The datagram does not fit in the MTU

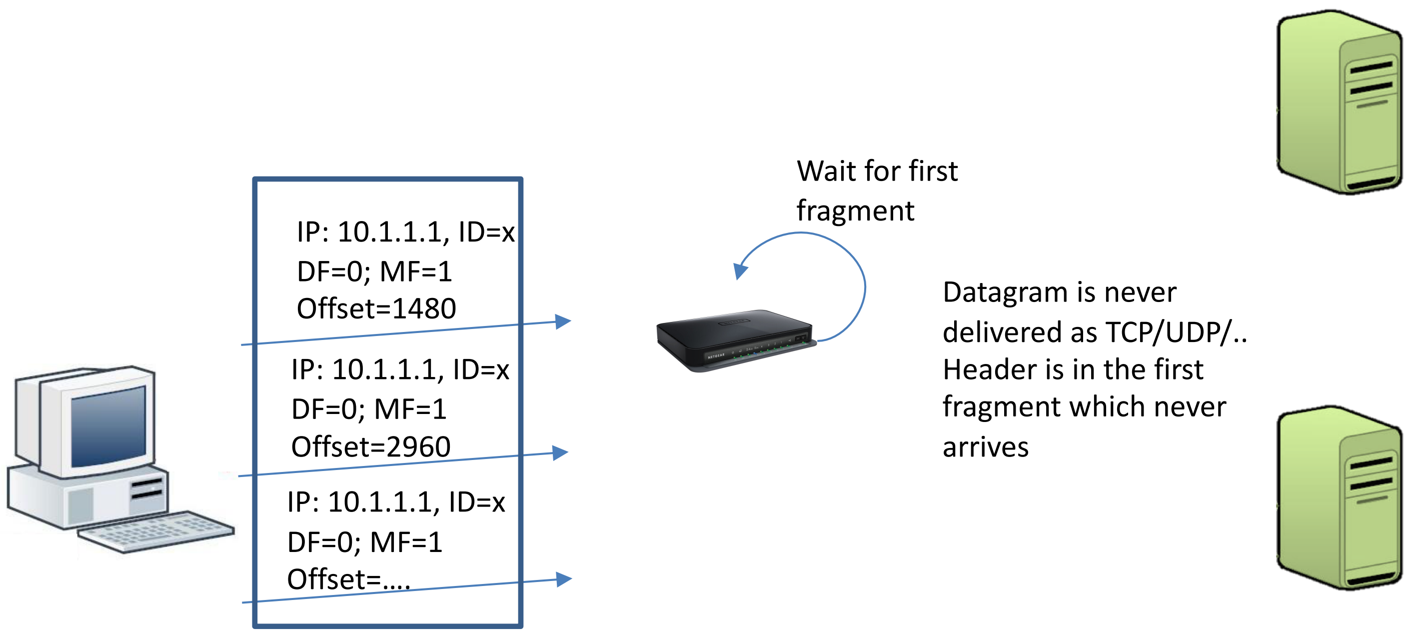


# Fragmentation example (cntd)




| | A | B | C |
|---|---|---|---|
| **Identification** | 4452 | 4452 | 4452 |
| **Flags** | • DF=0<br>• MF=1 | • DF=0<br>• MF=1 | • DF=0<br>• MF=0 |
| **Offset** | 0 | 1480 | 2960 |

# Denial of service with IP fragments



# Internet Control Message Protocol

- Defined in RFC 792
- Relies on IP
  - However, it is an **integral** part of the Internet Protocol
  - All IP modules must have ICMP support

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             unused                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

# Some ICMP Message types

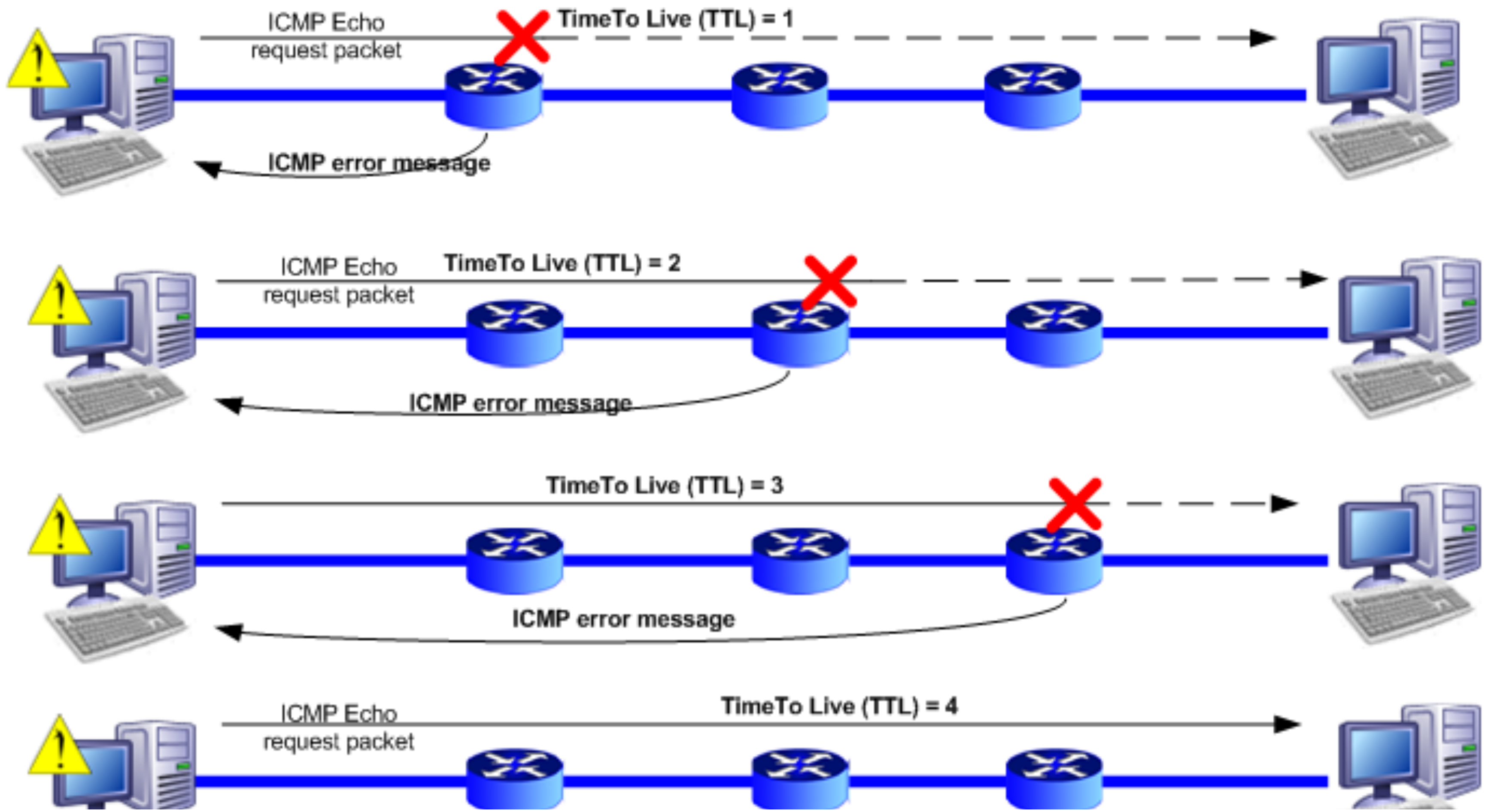
- **Destination Unreachable Message (Type 3)**
- Code
  - 0 = net unreachable;
  - 1 = host unreachable;
  - 2 = protocol unreachable;
  - 3 = port unreachable;
  - 4 = fragmentation needed and DF set;
  - 5 = source route failed.

- **Time Exceeded Message (Type 11)**
- Code
  - 0 = net unreachable;
  - 1 = host unreachable;

- **Echo or Echo Reply Message**
- Type
  - 8 = echo message;
  - 0 = echo reply;
- Code
  - 0

# Traceroute

See for example:
http://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/ios-software-releases-121-mainline/12778-ping-traceroute.html




Picture from: http://www.loriotpro.com/Products/On-line_Documentation_V5/LoriotProDoc_EN/J10-Loriotpro_tools/J10-U21_Trace_Route_EN.htm (no affiliation)

# List of all message types

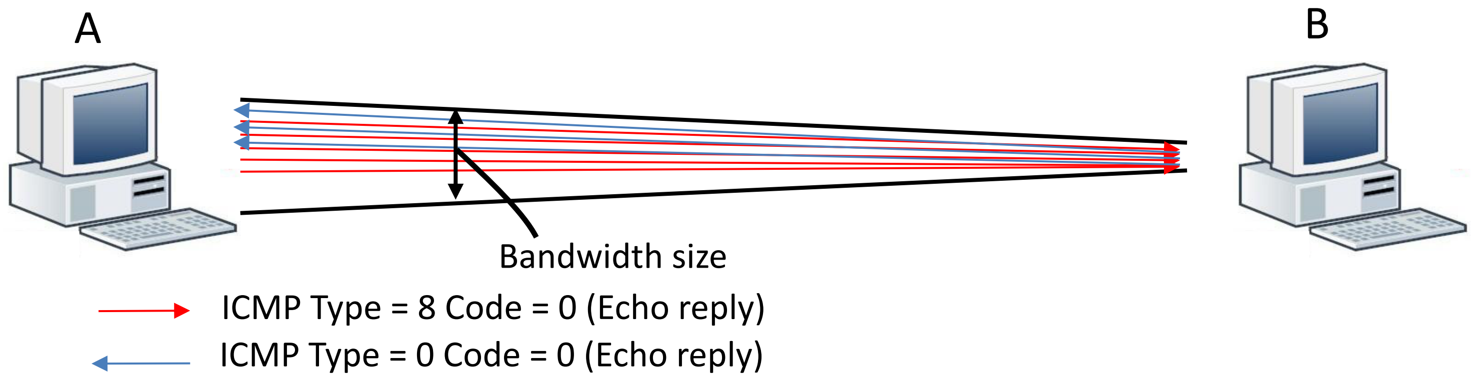
- 0  Echo Reply
- 3  Destination Unreachable
- 4  Source Quench
- 5  Redirect
- 8  Echo
- 11  Time Exceeded
- 12  Parameter Problem
- 13  Timestamp
- 14  Timestamp Reply
- 15  Information Request
- 16  Information Reply

# Denial of Service

- Denial of service (DoS) is a type of attack that aims at congesting or overpowering a system’s capacity by generating requests the system will have to answer
  - Can affect the performance of the attacked system or its channels
  - Can lead to a system crash due to resource consumption
- DoS can be operated
  - Locally
  - Over the network

# A simple DoS (Ping Flood)

- Network DoS attacks usually exploit protocol features

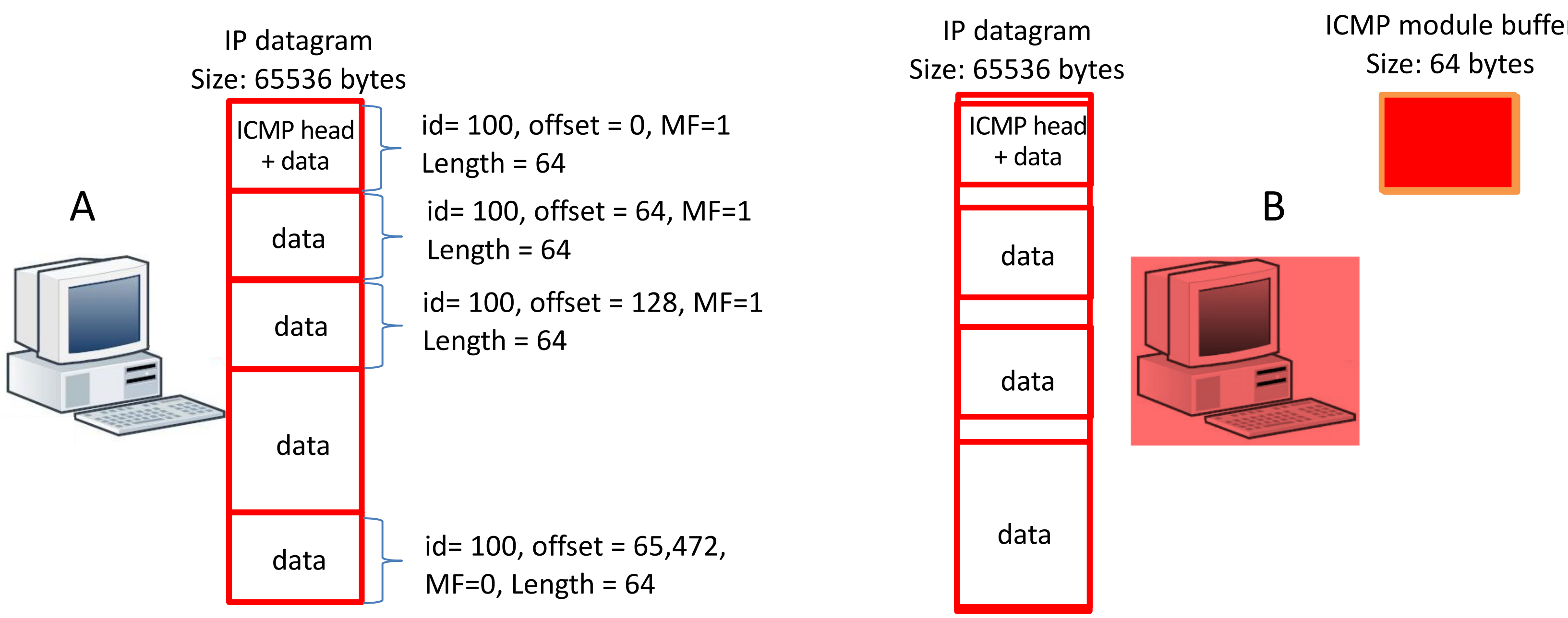



- A can exploit its wider bandwidth to flood B with ICMP echo requests
- B's bandwidth gets (quickly, relatively to A's) exhausted with
  - A's requests
  - B's replies
- B can no longer operate on its network channel

# A more advanced DoS – Ping of Death

- ICMP packets are typically 64 bytes in size including IP headers and data
- IP datagram can extend up to 65,535 bytes
  - Data Length field is 16 bit
- Early implementations of Internet modules were strictly implementing RFC directives
  - Not handling exceptions properly
- Ping of Death
  - Generate large ICMP packet
  - Fragment in 1024 IP packets of 64 Bytes
  - Destination receives regular packet
    - IP module compose fragments
    - ICMP module tries to read datagram bigger than assigned buffer size
  - Destination crashes
    - "buffer overflow" → possible execution of code in memory (more on this in this course)

# Ping of death → visualisation



# Useful network tools

- Wireshark / tcpdump → traffic monitoring
  - ARP requests
  - DNS requests
  - TCP 3-way handshake → SYN ACK
  - Network stack overview
- Nmap → scans (TCP; UDP; ..)
- Scapy → Python interface to generate network packets at the stack level
  - Manually craft network packets
- Other tools:
  - Ettercap → MitM attacks (ARP poisoning etc.)
  - Netcat → legacy tool to generate UDP/TCP traffic