CSCI 466: Networks

Link Layer: Error Detection, Multiple Access Protocols

Reese Pearsall

Fall 2024

Announcements

Quiz on Friday

- No class on Quiz days

Next Wireshark lab will be posted later today

OSI Model

- Application Layer
- Presentation Layer *
- Session Layer *
- Transport Layer
- Network Layer
- Data Link Layer
- Physical Layer

Application Layer

Messages from Network Applications

Physical Layer

Bits being transmitted over a copper wire

In the textbook, they condense it to a 5-layer model, but 7 layers is what is most used

Data Link Layer

The link layer is responsible for the **actual node-to-node delivery** of data and ensure error-free transmission of information

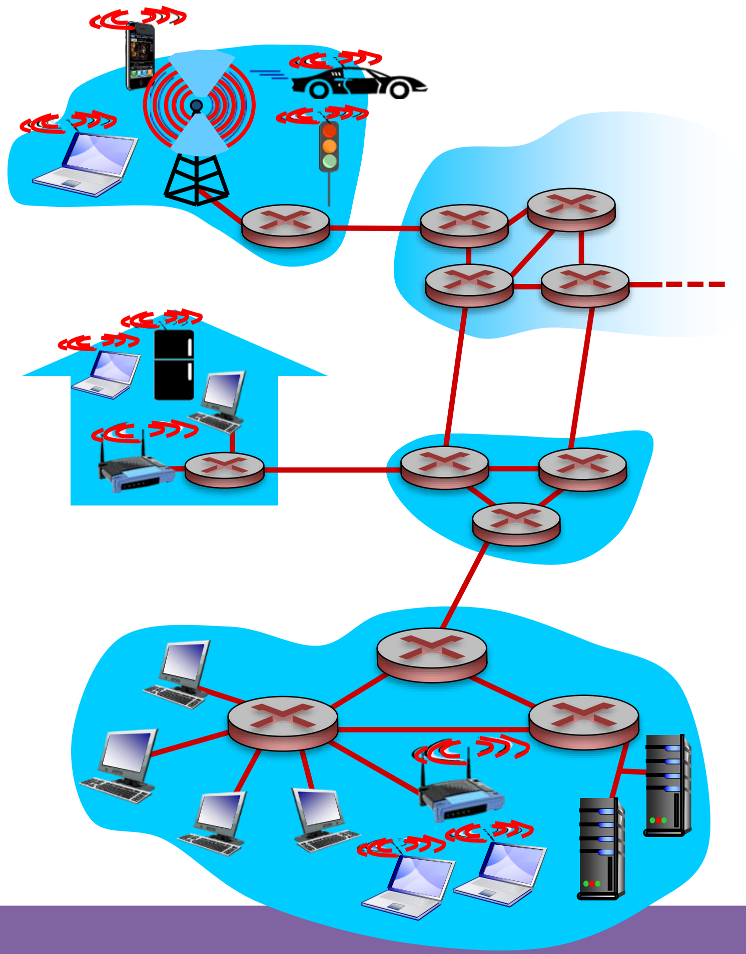
terminology:

hosts and routers: **nodes**

communication channels that connect adjacent nodes along communication path: **links**

- wired links
- wireless links
- LANs

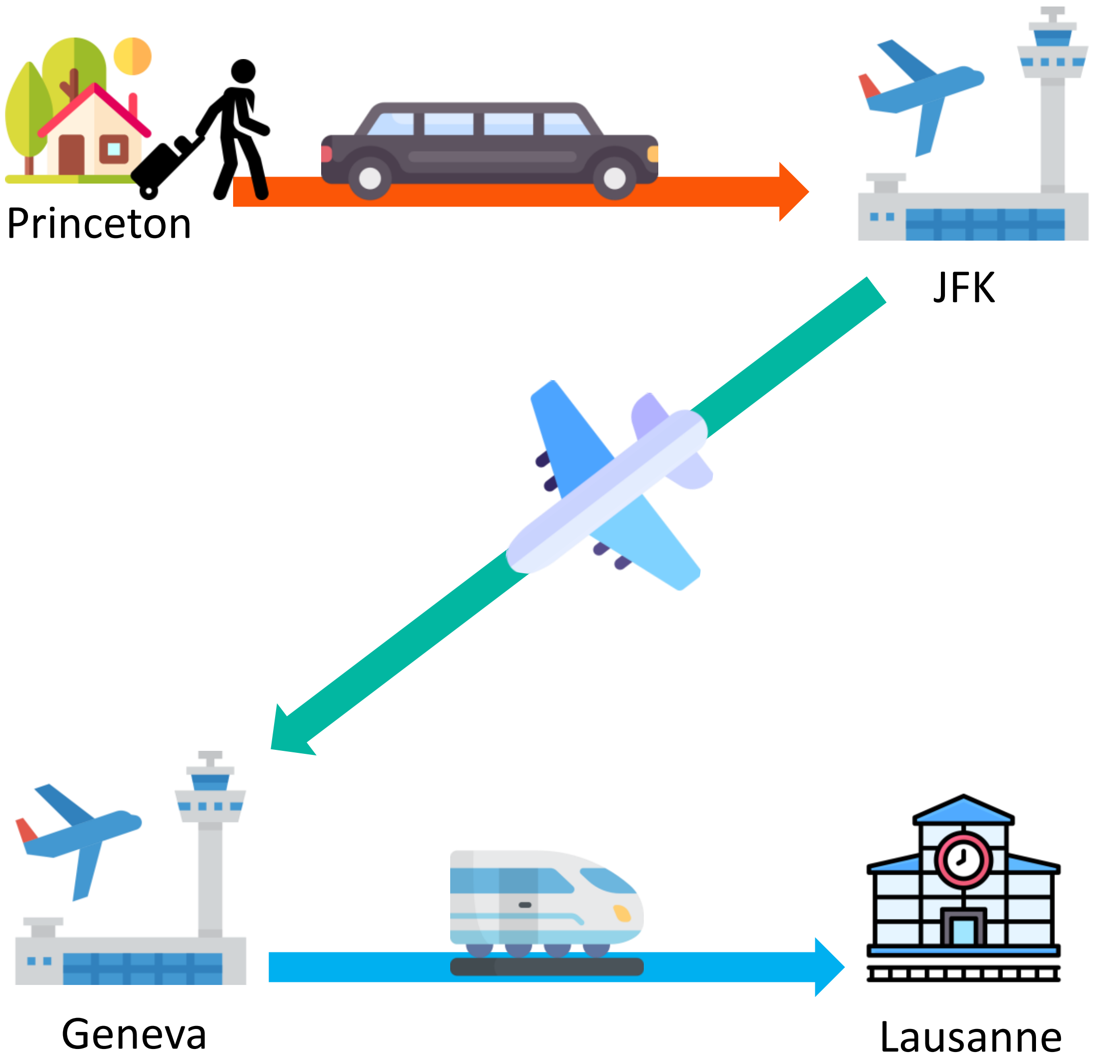
layer-2 packet: frame, encapsulates **datagram**



We have not addressed how we will overcome various transmission mediums!

data-link layer has responsibility of transferring datagram from one node to *physically adjacent* node over a link

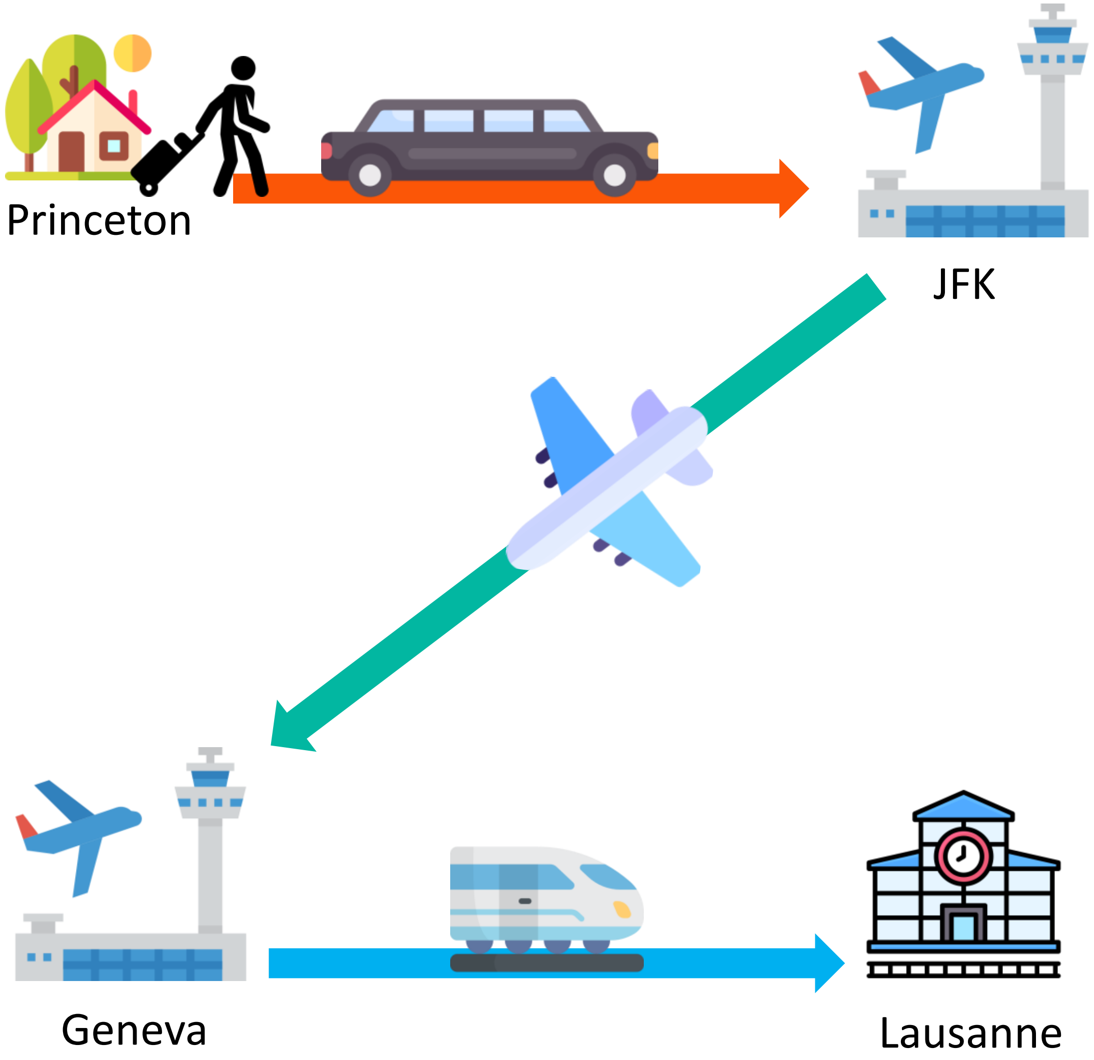
Data Link Layer



transportation analogy:

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne

Data Link Layer



transportation analogy:

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = datagram
- transport segment = communication link
- transportation mode = link-layer protocol
- travel agent = routing algorithm

Data Link Layer

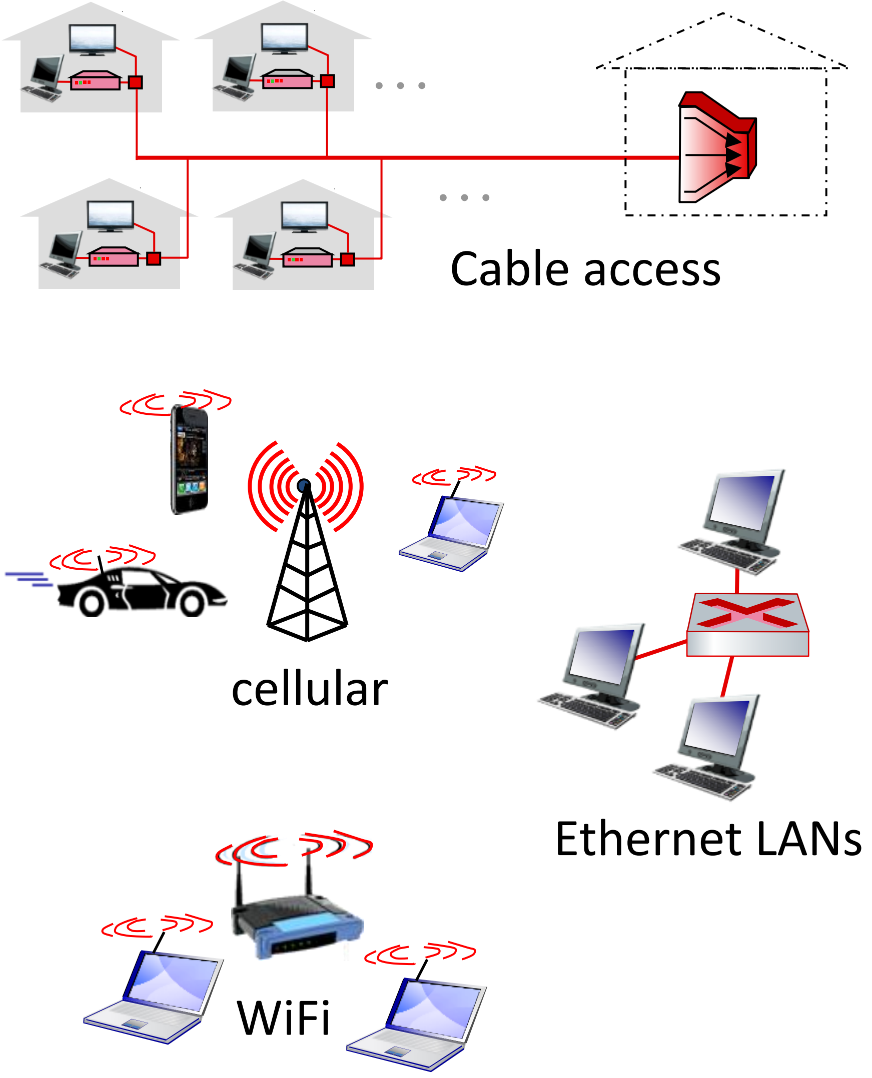
Services offered by the Link Layer

- Framing
  - → Encapsulate a network layer Datagram in *another* header
- Link access
  - → LL dictate the rules and process of transmitting a frame over a link
- Reliable Delivery
  - → For unreliable link, some reliable delivery mechanisms may need to be used
- Error Detection and Correction
  - → Bits can get messed up as the are transmitted through a medium

Why do we need RDT and error detection in the link layer when it is also offered in the transport layer?

Data Link Layer

- **flow control:**
  - pacing between adjacent sending and receiving nodes
- **error detection:**
  - errors caused by signal attenuation, noise.
  - receiver detects errors, signals retransmission, or drops frame
- **error correction:**
  - receiver identifies *and corrects* bit error(s) without retransmission
- **half-duplex and full-duplex:**
  - with half duplex, nodes at both ends of link can transmit, but not at same time



Data Link Layer

Services offered by the Link Layer

- Framing
  → Encapsulate a network layer Datagram in *another* header
- Link access
  → LL dictate the rules and process of transmitting a frame over a link
- Reliable Delivery
  → For unreliable link, some reliable delivery mechanisms may need to be used
- Error Detection and Correction
  → Bits can get messed up as the are transmitted through a medium

Why do we need RDT and error detection in the link layer when it is also offered in the transport layer?

Some packets of data don't even travel through the transport layer…

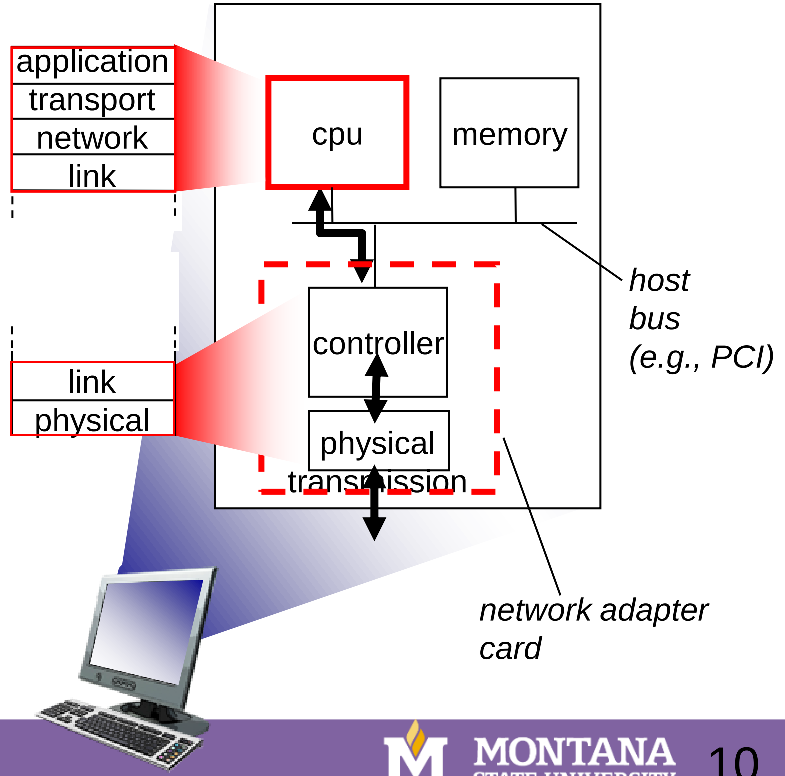
Data Link Layer

Implementation of Link Layer

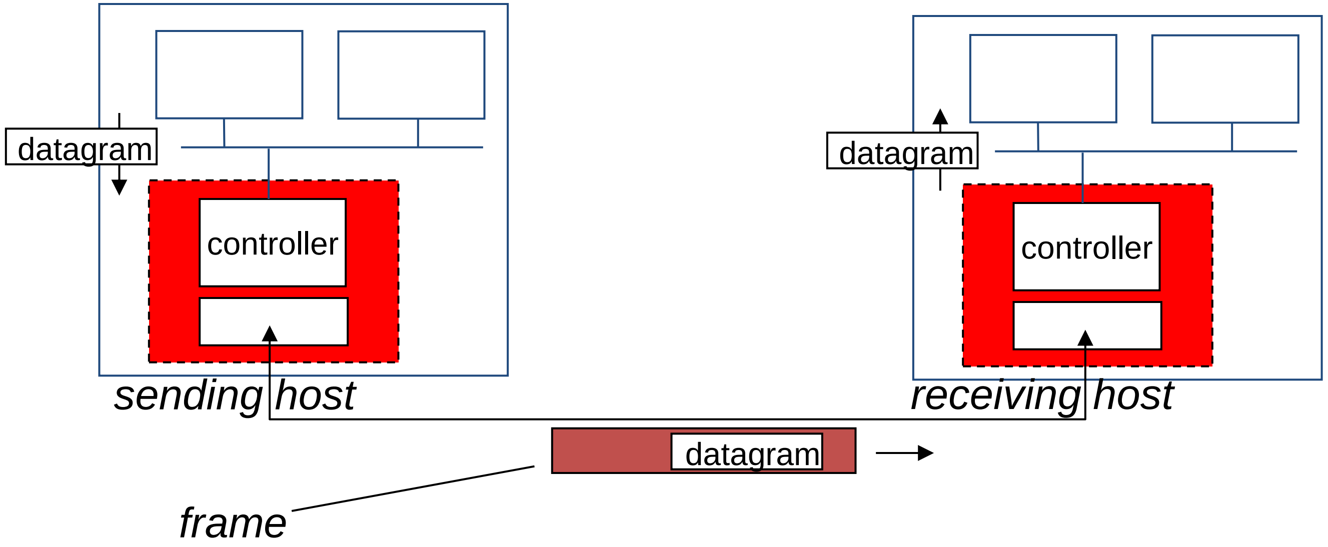
- Implemented within the hardware of your computer

NIC (Network Interface Controller)- Integrated into the motherboard and allows the machine to use LL services such as ethernet (combination of hardware, software, and some firmware)

Wireshark uses your NIC to determine which packets should be sniffed!



Data Link Layer



sending side:

encapsulates datagram in frame
adds error checking bits, rdt, flow control, etc.

receiving side

looks for errors, rdt, flow control, etc.
extracts datagram, passes to upper layer at receiving side

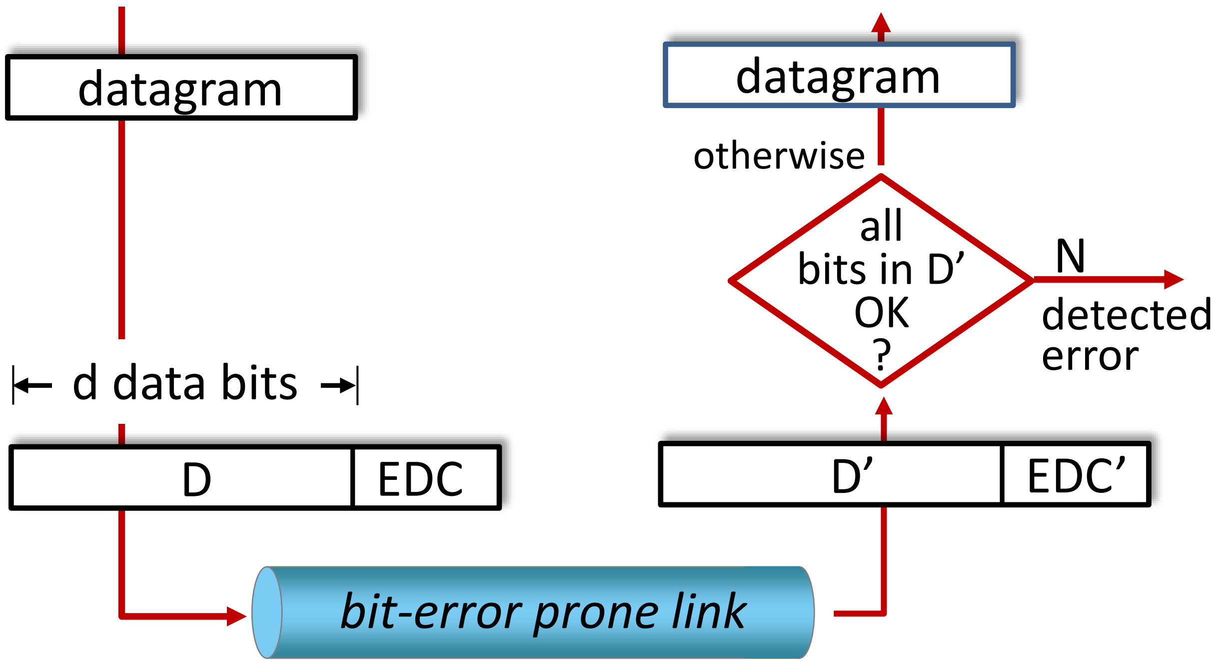
Data Link Layer

Bits can get messed during the physical layer and link layer

- Faulty wires
- NIC issues
- Unreliable mediums

EDC: error detection and correction bits

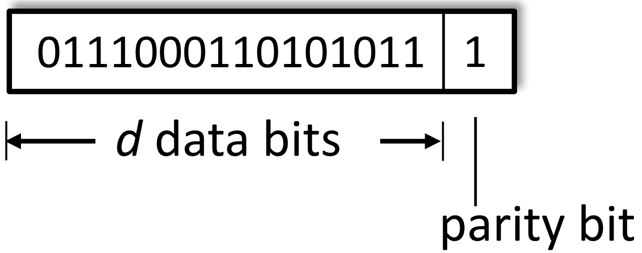
D: data protected by error checking, may include header fields



Error detection not 100% reliable!

- protocol may miss some errors, but rarely
- larger EDC field yields better detection and correction

Data Link Layer



Even/odd parity: set parity bit so there is an even/odd number of 1's

At receiver:

- compute parity of *d* received bits
- compare with received parity bit – if different than error detected

Data Link Layer

0111000110101011	1

← *d* data bits → parity bit

Even/odd parity: set parity bit so there is an even/odd number of 1's

At receiver:

- compute parity of *d* received bits
- compare with received parity bit – if different than error detected

Can detect *and* correct errors (without retransmission!)

- two-dimensional parity: detect *and correct* single bit errors

row parity →

column parity ↓

$$
\begin{array}{cccc|c}
d_{1,1} & \cdots & d_{1,j} & & d_{1,j+1} \\
d_{2,1} & \cdots & d_{2,j} & & d_{2,j+1} \\
\cdots & \cdots & \cdots & & \cdots \\
d_{i,1} & \cdots & d_{i,j} & & d_{i,j+1} \\
\hline
d_{i+1,1} & \cdots & d_{i+1,j} & & d_{i+1,j+1}
\end{array}
$$

no errors:

$$
\begin{array}{ccccc|c}
1 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 0 & 1 & 0
\end{array}
$$

detected and correctable single-bit error:

$$
\begin{array}{ccccc|c}
1 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 0 & 1 & 0
\end{array}
$$

parity error

Data Link Layer

Checksum

(Sender)

```
  0110011001100000
+ 0101010101010101
+ 1000111100001100
```

0100101011000010 Binary sum of words

(one's complement)

1011010100111101 Checksum!

(Receiver)

```
  0110011001100000
+ 0101010101010101
+ 1000111100001100
+ 0100101011000010
```

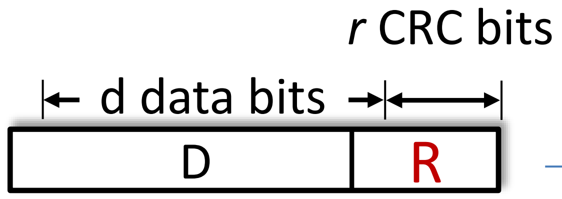
(Binary Sum → One's Complement)

= 1111111111111111

All 1s = No error!

Data Link Layer

- more powerful error-detection coding
- D: data bits (given, think of these as a binary number)
- G: bit pattern (generator), of *r+1* bits (given, specified in CRC standard)



bits to send

$\langle D,R \rangle = D * 2^r \text{ XOR } R$

formula for these bits

sender: compute *r* CRC bits, R, such that <D,R> *exactly* divisible by G (mod 2)

- receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
- can detect all burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi)

Sender/Receiver has D and G.
Need to compute R

Data Link Layer

Sender wants to compute R such that:

$$D \cdot 2^r \;\; XOR \;\; R = nG$$

... or equivalently (XOR R both sides):

$$D \cdot 2^r = nG \;\; XOR \;\; R$$

... which says:

if we divide $D \cdot 2^r$ by G, we want remainder R to satisfy:

$$R = remainder\left[\frac{D \cdot 2^r}{G}\right]$$

algorithm for computing R

```
                 G
                       1 0 1 0 1 1
         1 0 0 1 ) 1 0 1 1 1 0 0 0 0    D*2^r  (here, r=3)
                   1 0 0 1
                       1 0 1
                       0 0 0
                       1 0 1 0
                       1 0 0 1
                           1 1 0
                           0 0 0
                           1 1 0 0
                           1 0 0 1
                             1 0 1 0
                             1 0 0 1
                               0 1 1
                                 R
```

Data Link Layer

← d bits → ← r bits →

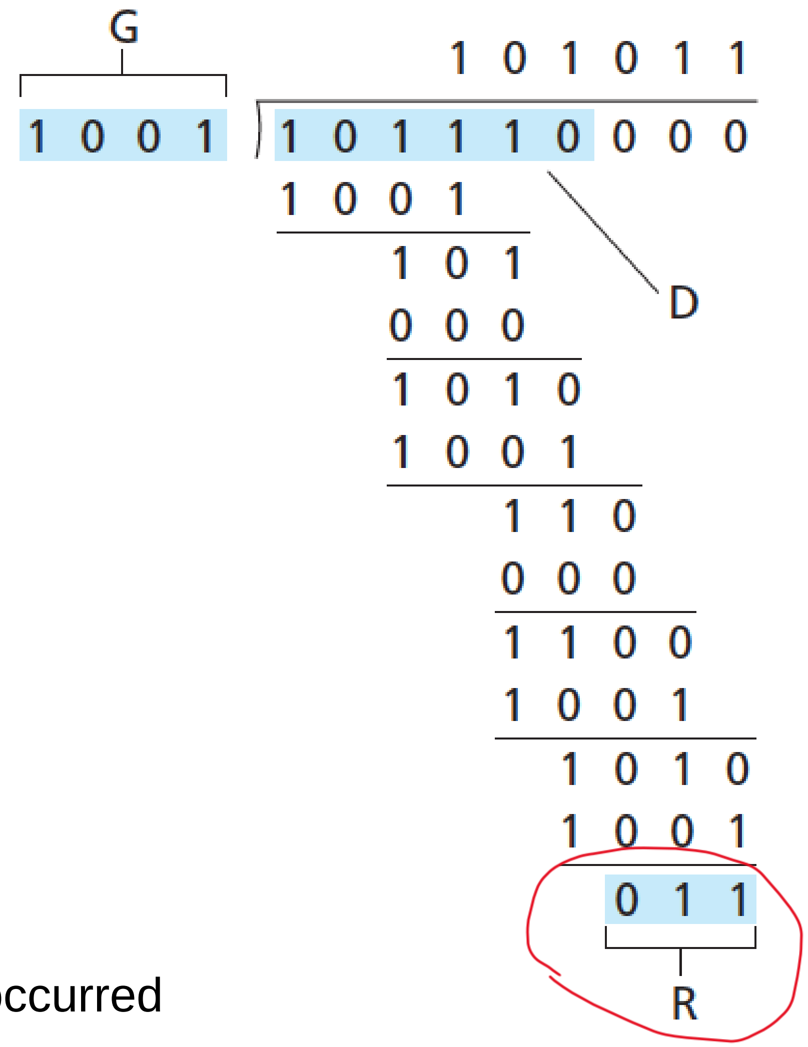
D: data bits to be sent	*R:* CRC bits

bit pattern

$D * 2^r$ XOR R *mathematical formula*

(Do some algebra to find R)

$$R = remainder\left[\frac{D \cdot 2^r}{G}\right]$$



Sender sends D + R bits.

Receiver divides D + R bits by G. Result should always be Zero if no errors occurred

Data Link Layer

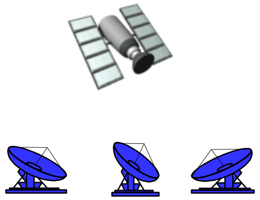
Access links

- Point to Point – Single sender, Single Receiver at each end of link

- Broadcast – shared medium

shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)

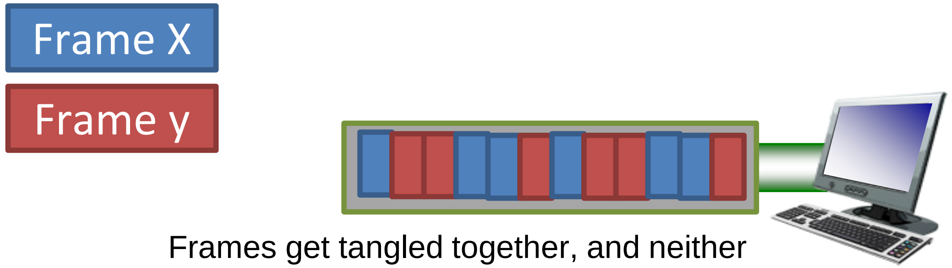


shared RF (satellite)

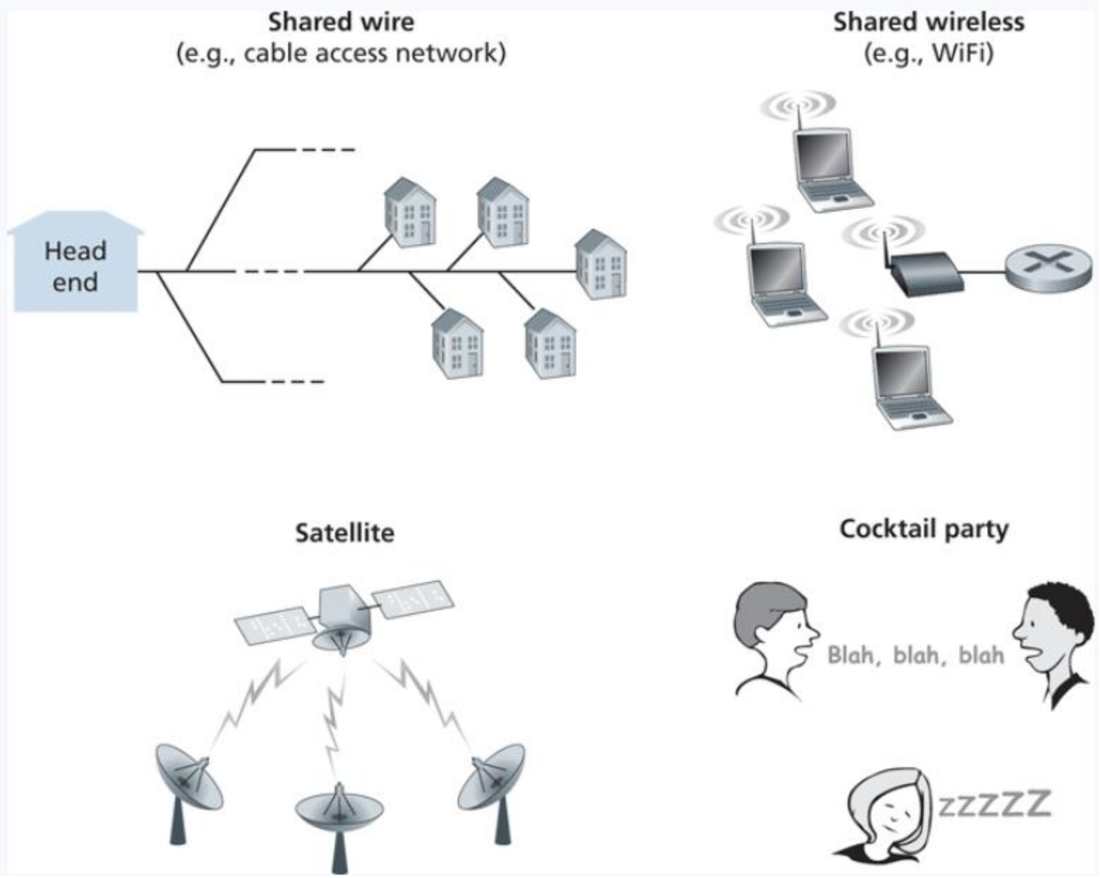
humans at a cocktail party (shared air, acoustical)

Multiple Access Links

Shared medium = possibility for receivers to get two frame at the same time, AKA a **collision**



Frames get tangled together, and neither frame can be read clearly by receiver

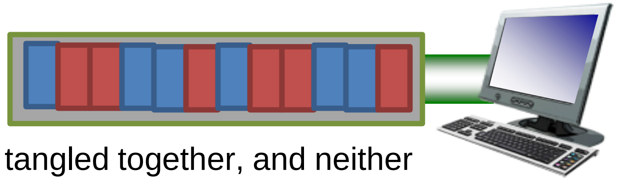


Multiple Access Links

Shared medium = possibility for receivers to get two frame at the same time, AKA a **collision**

Frame X

Frame y



Frames get tangled together, and neither frame can be read clearly by receiver

"Give everyone a chance to speak."

"Don't speak until you are spoken to."

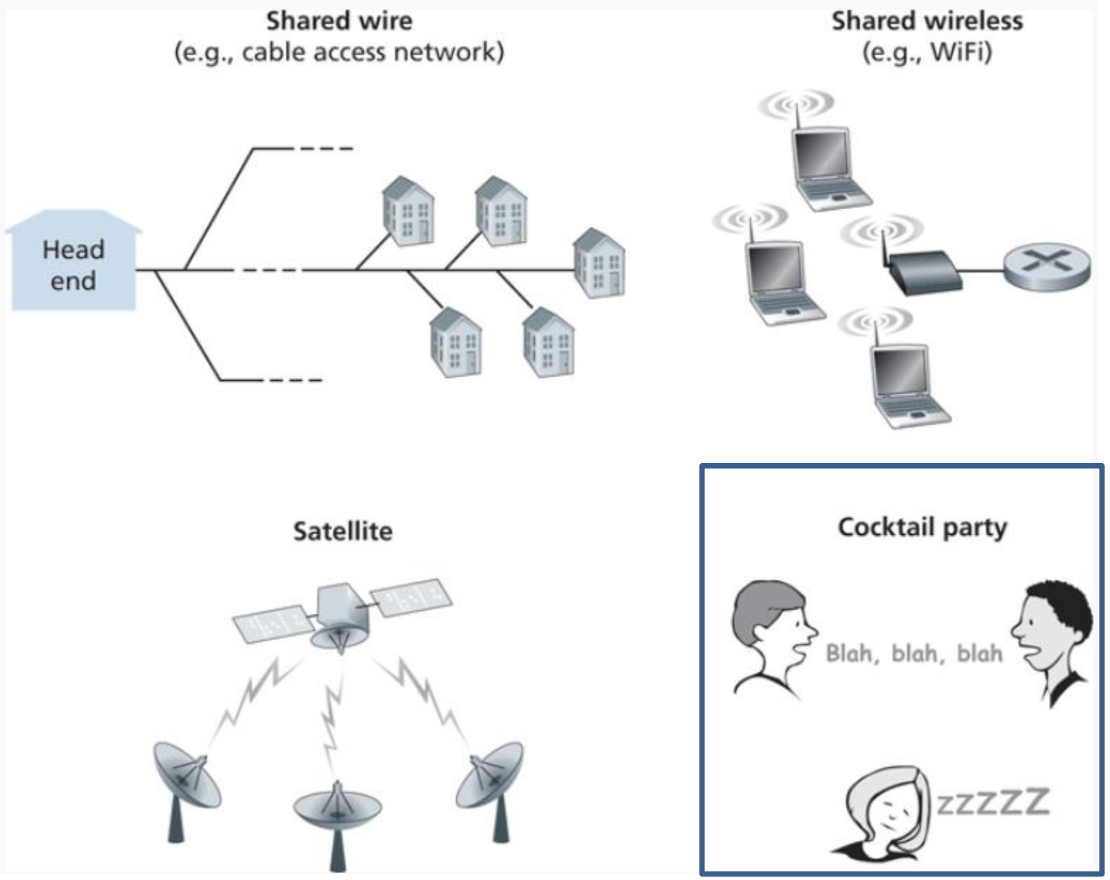
"Don't monopolize the conversation."

"Raise your hand if you have a question."

"Don't interrupt when someone is speaking."

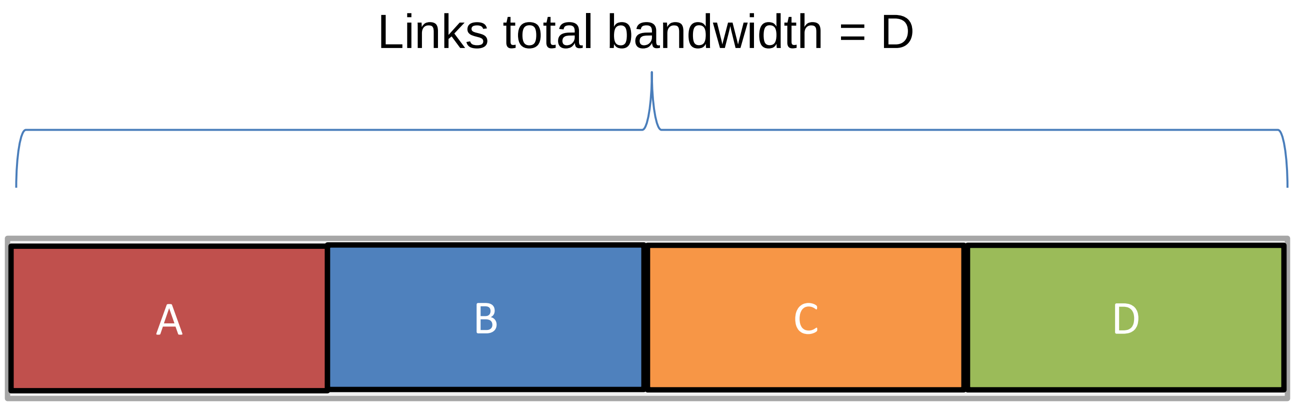
"Don't fall asleep when someone is talking."

In English, we have some rules to prevent collisions from happening

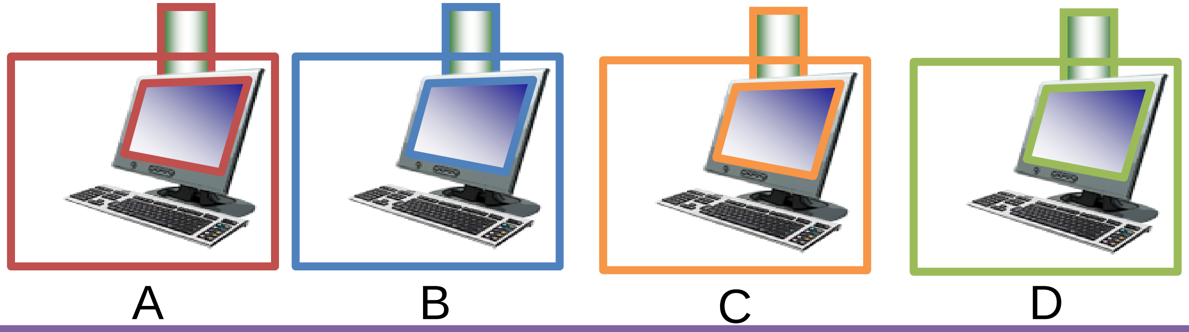


In the link layer, we will discuss 3 multiple access protocols: **Channel Partitioning**, **Random Access**, and **Taking Turns**

Channel Partitioning

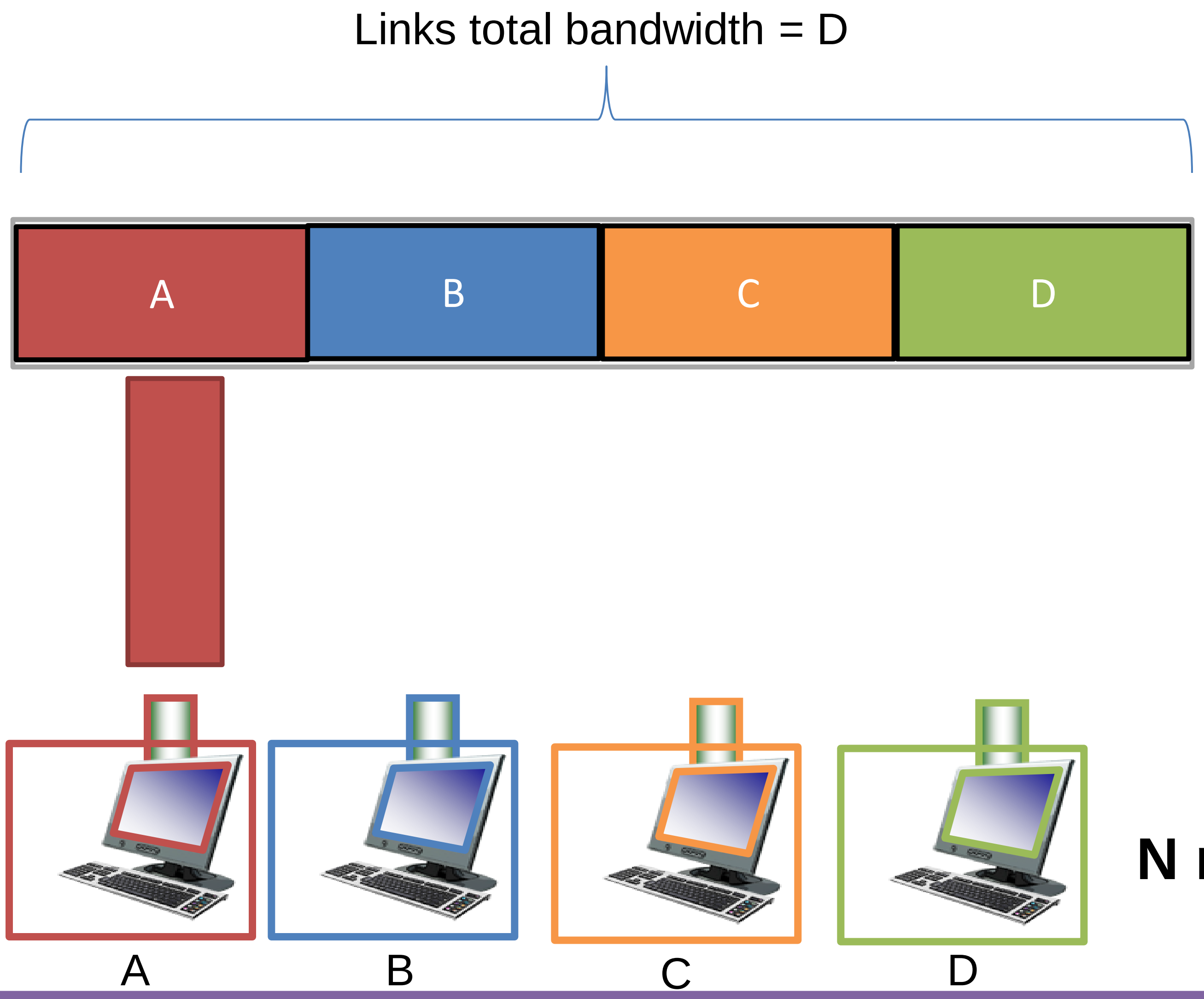


- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit



N nodes

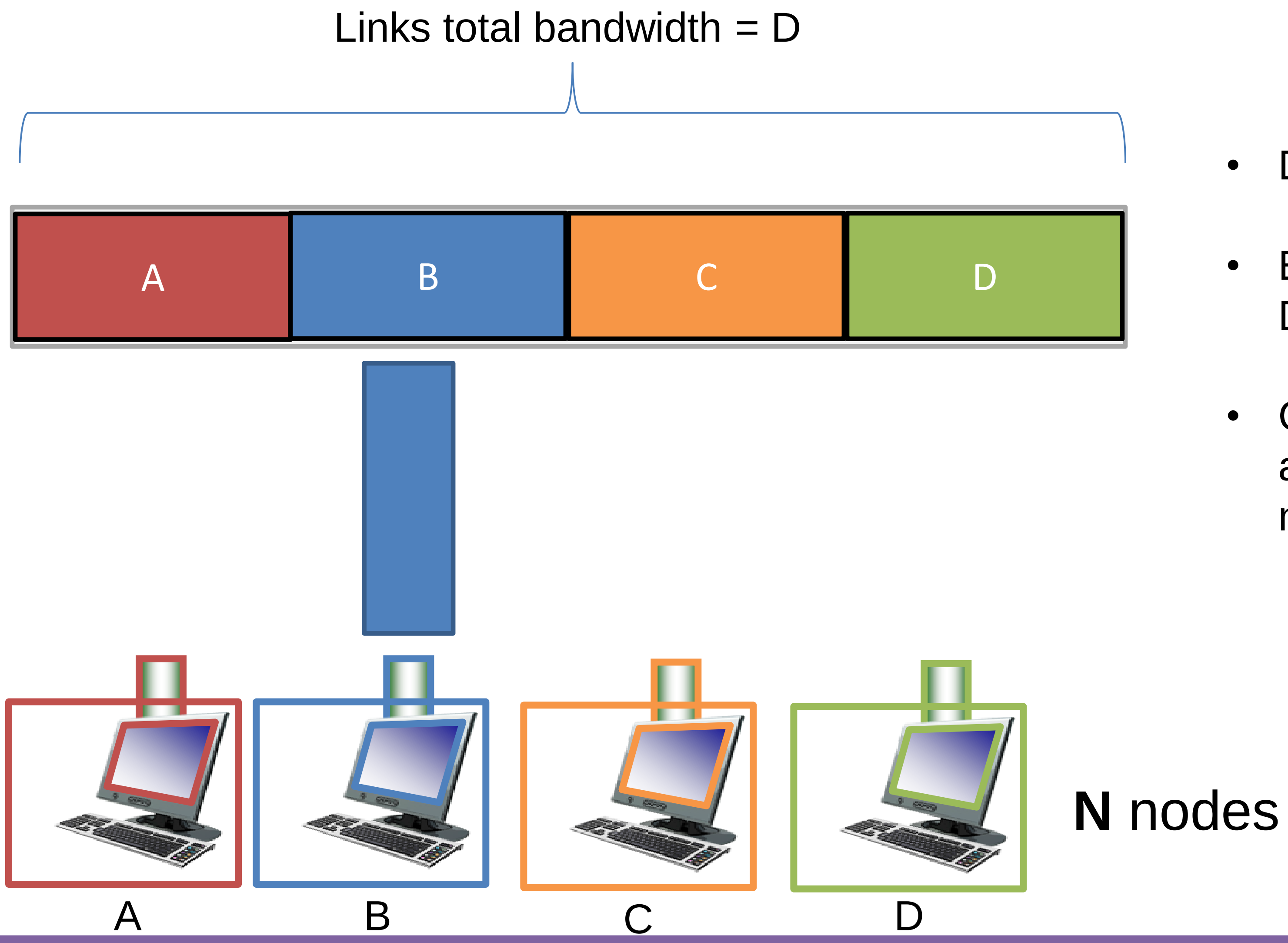
Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

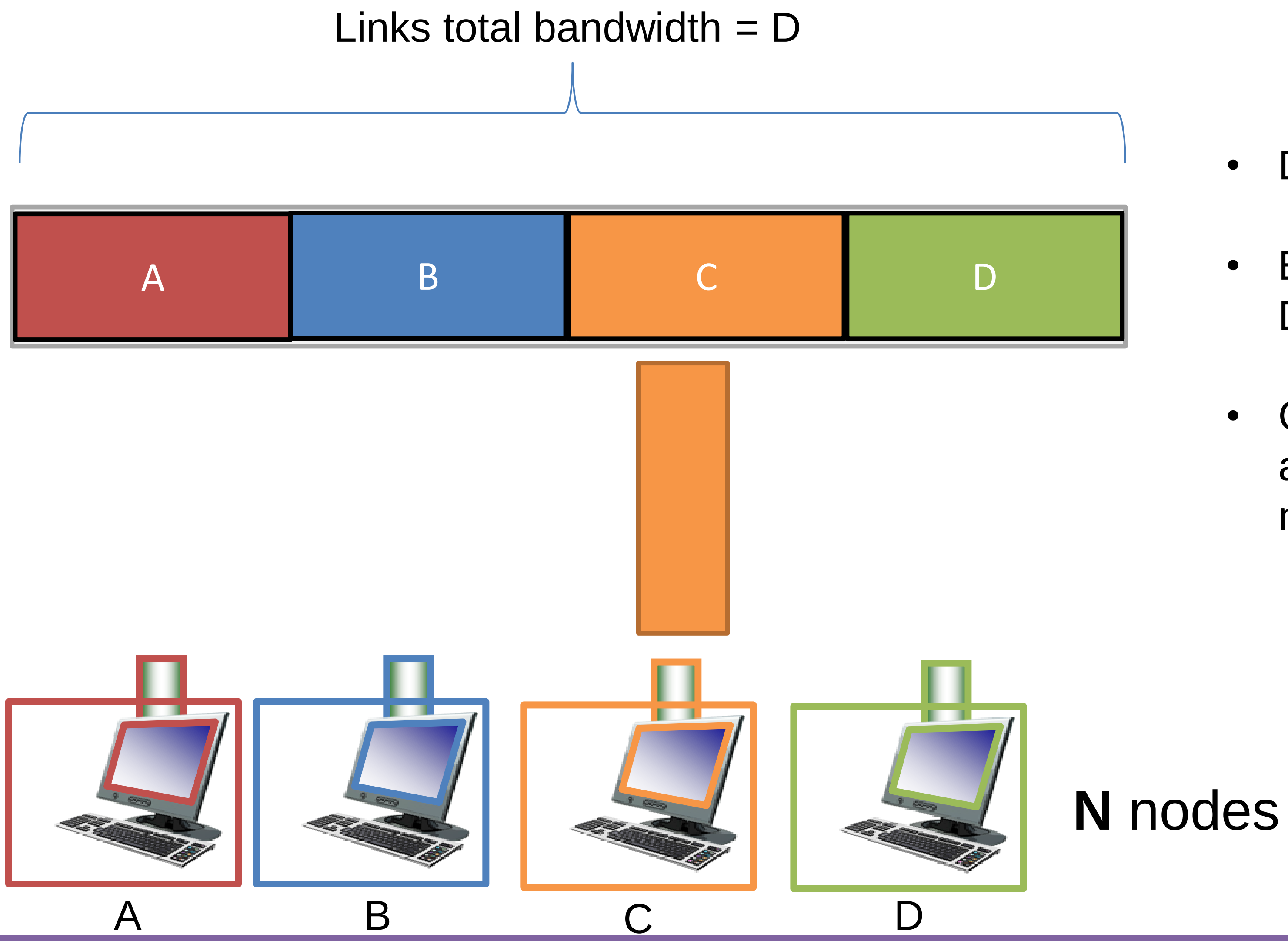
N nodes

Channel Partitioning



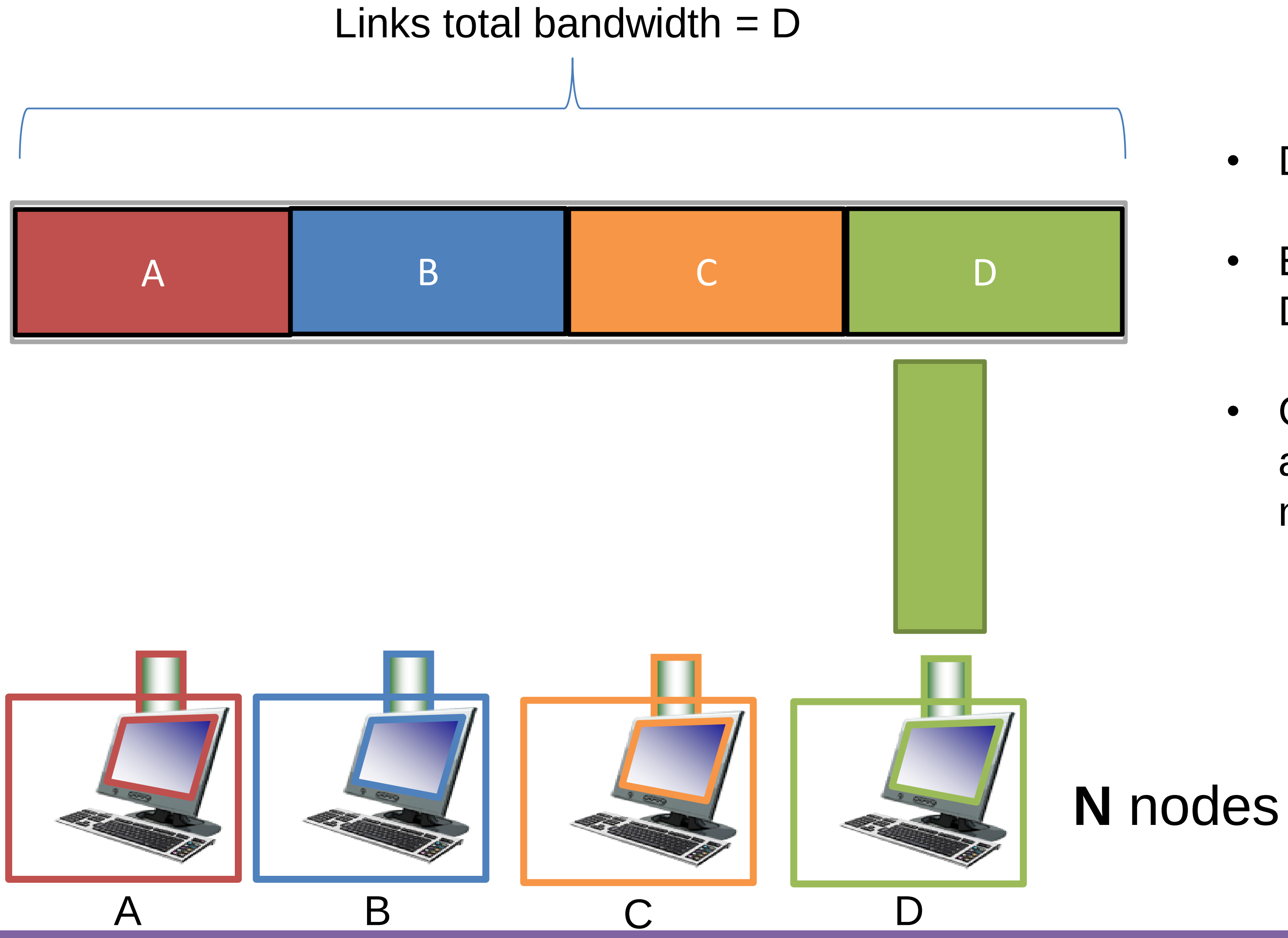
- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

Channel Partitioning



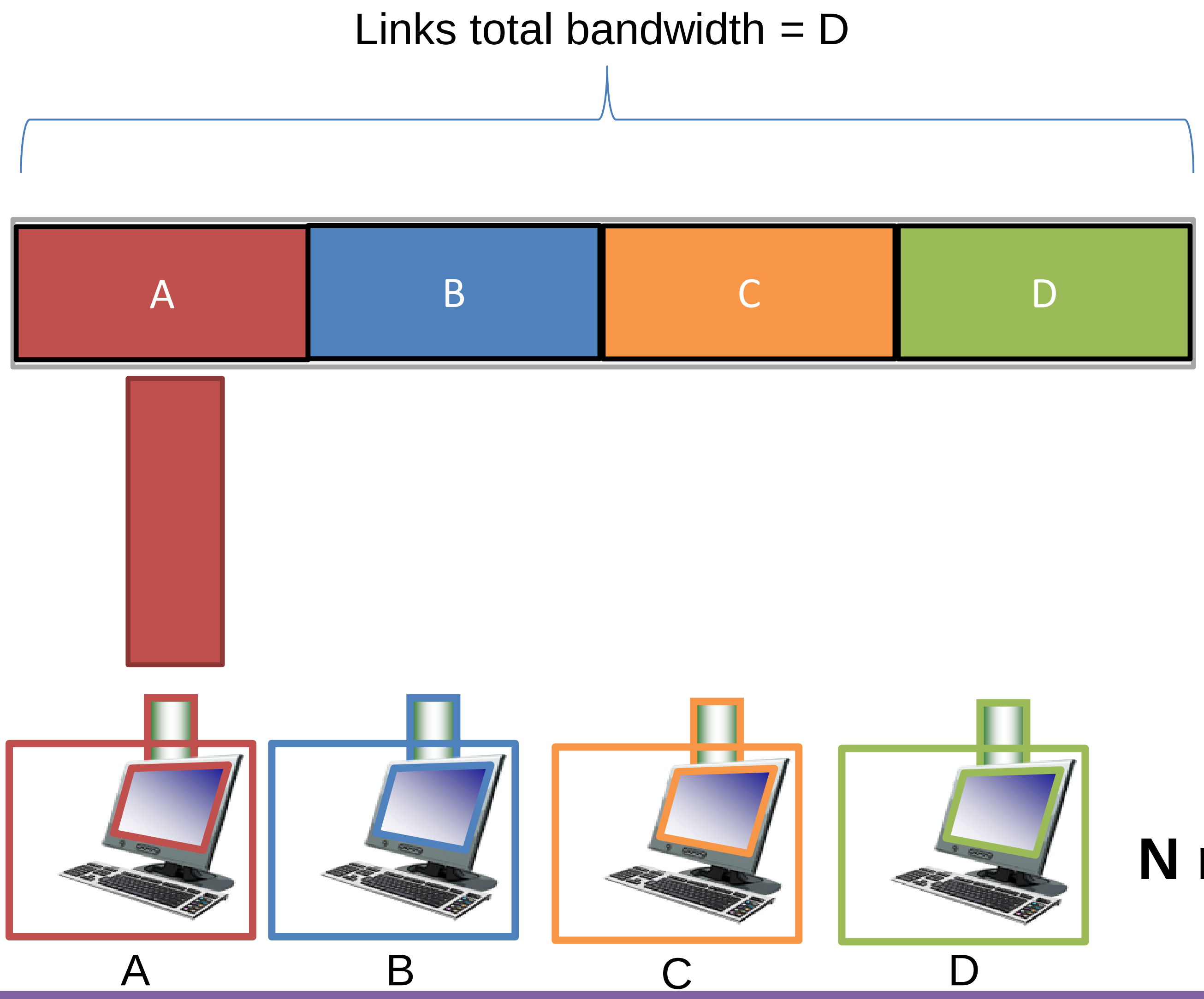
- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

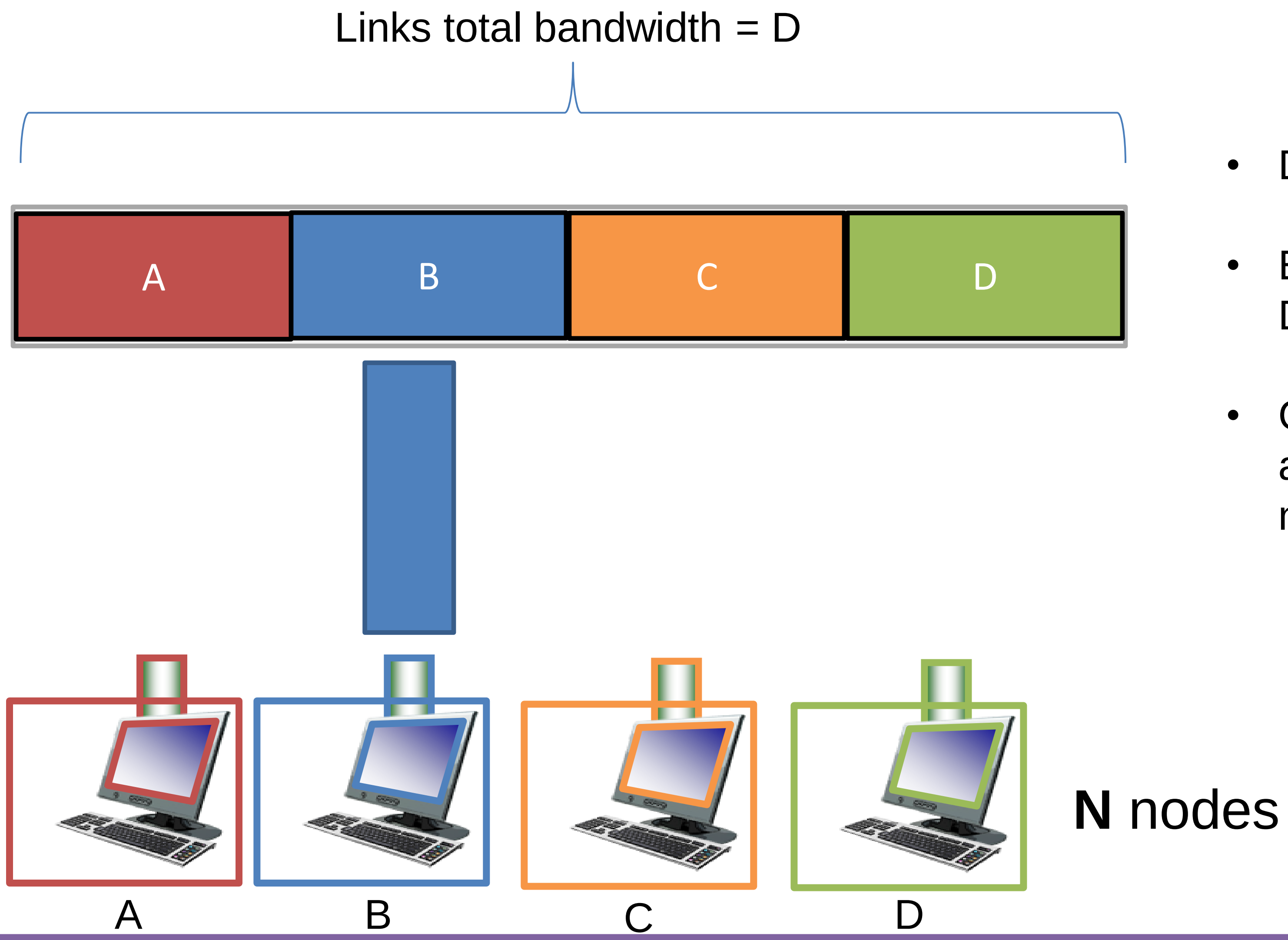
Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

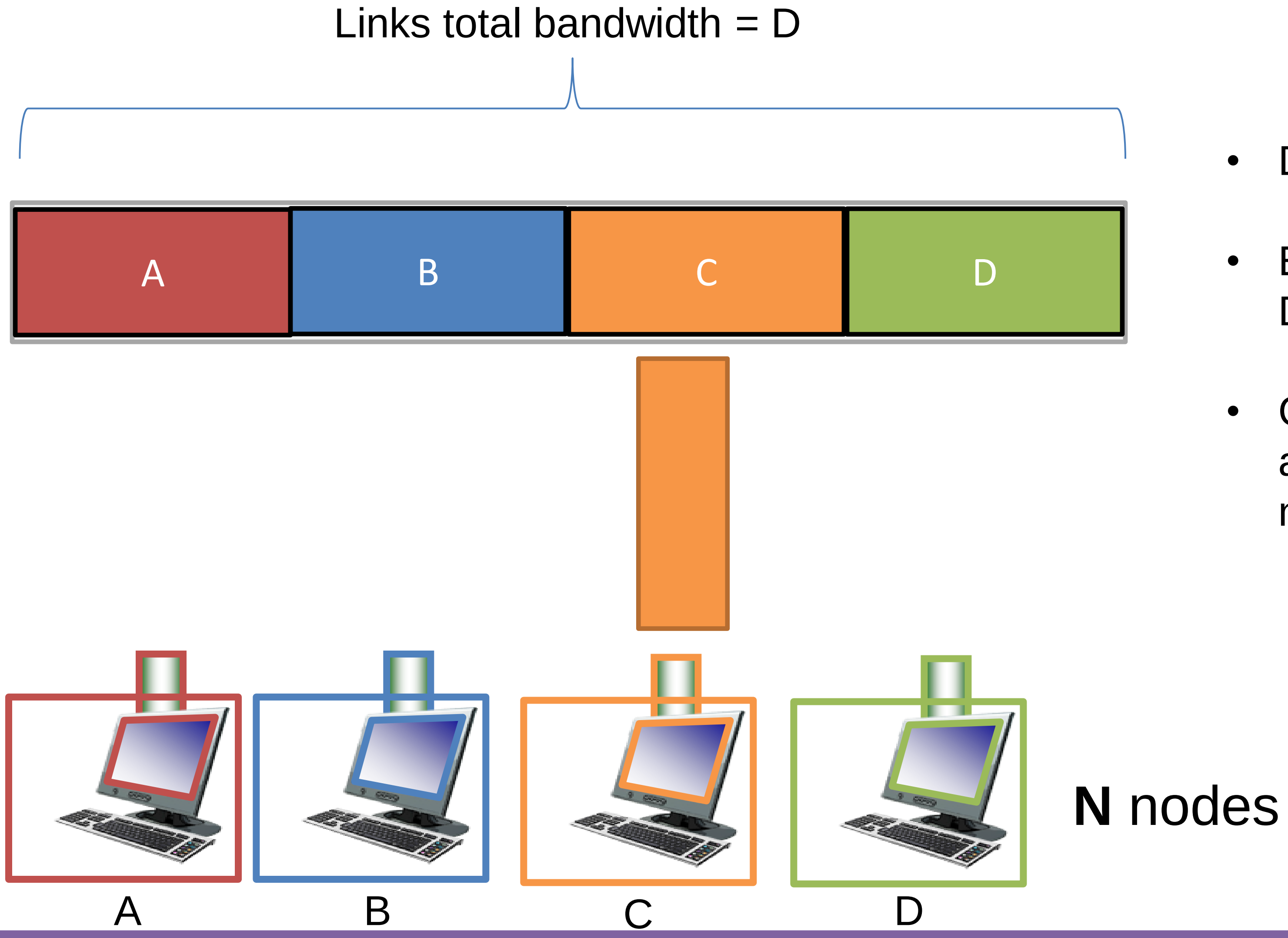
N nodes

Channel Partitioning



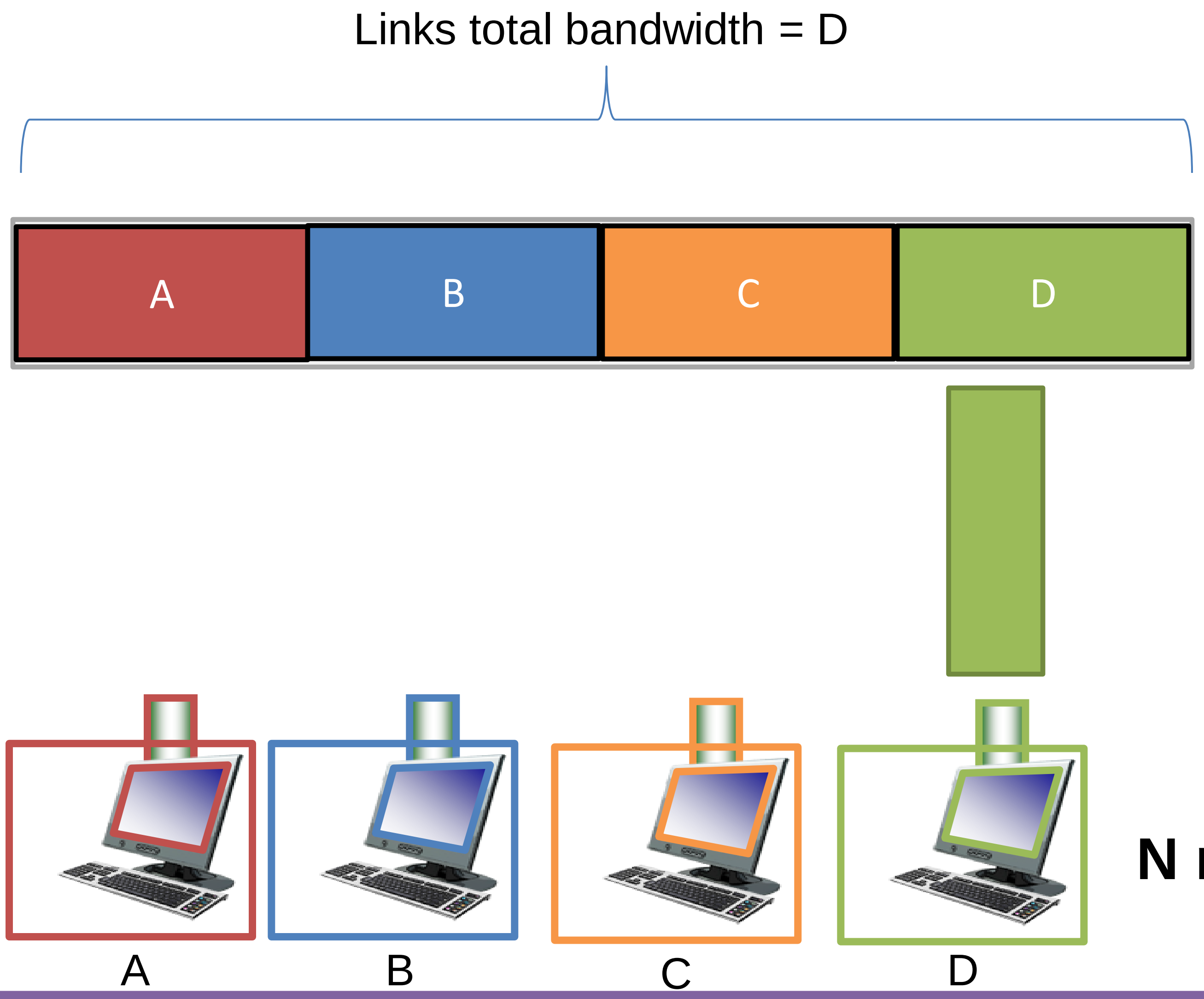
- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

Channel Partitioning



N nodes

- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

Random Access

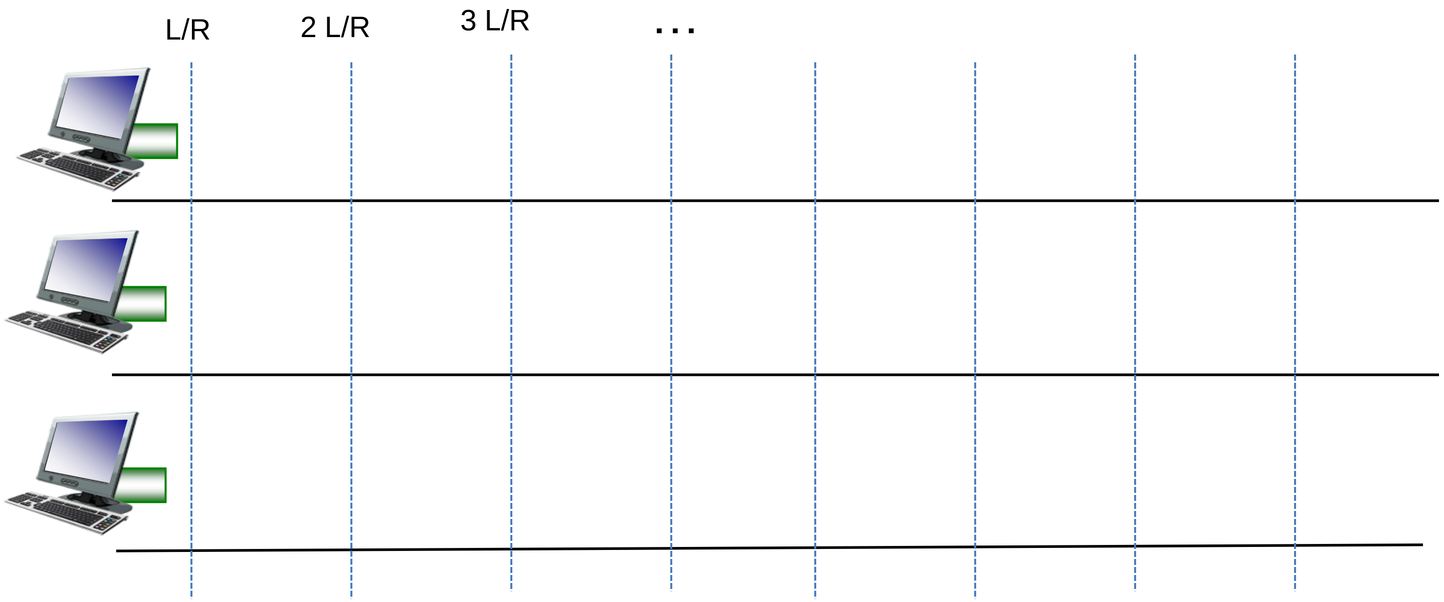
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Can only transmit frames at beginning of slots. If collision occurs, the nodes can detect collision before the slot ends

Random Access

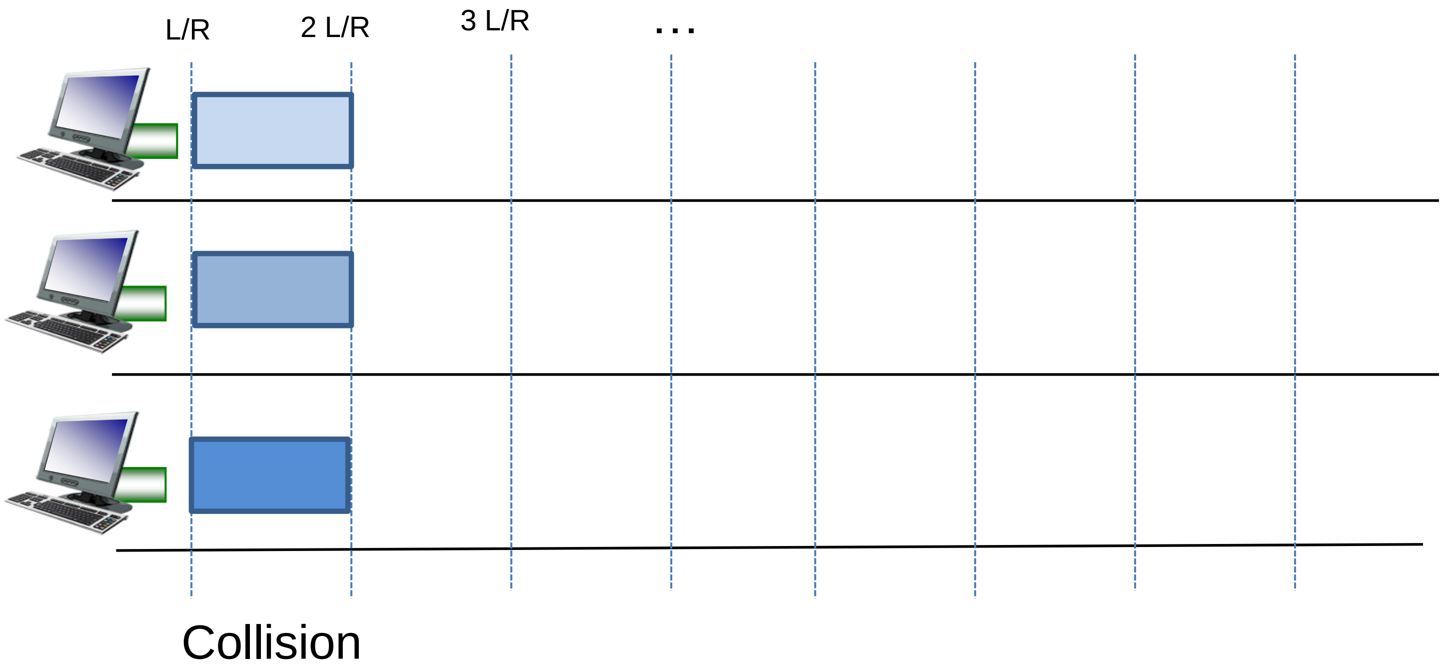
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R "slots"

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Random Access

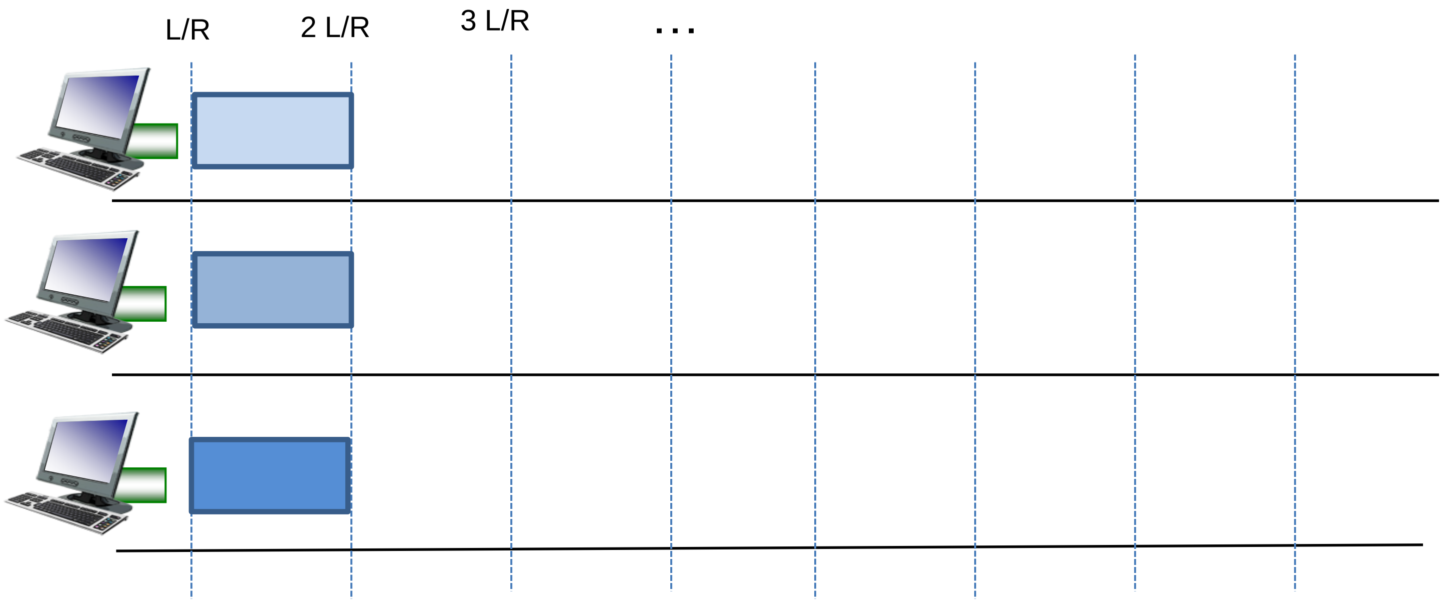
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Collision

Do some probability ***p*** and retransmit if needed

Random Access

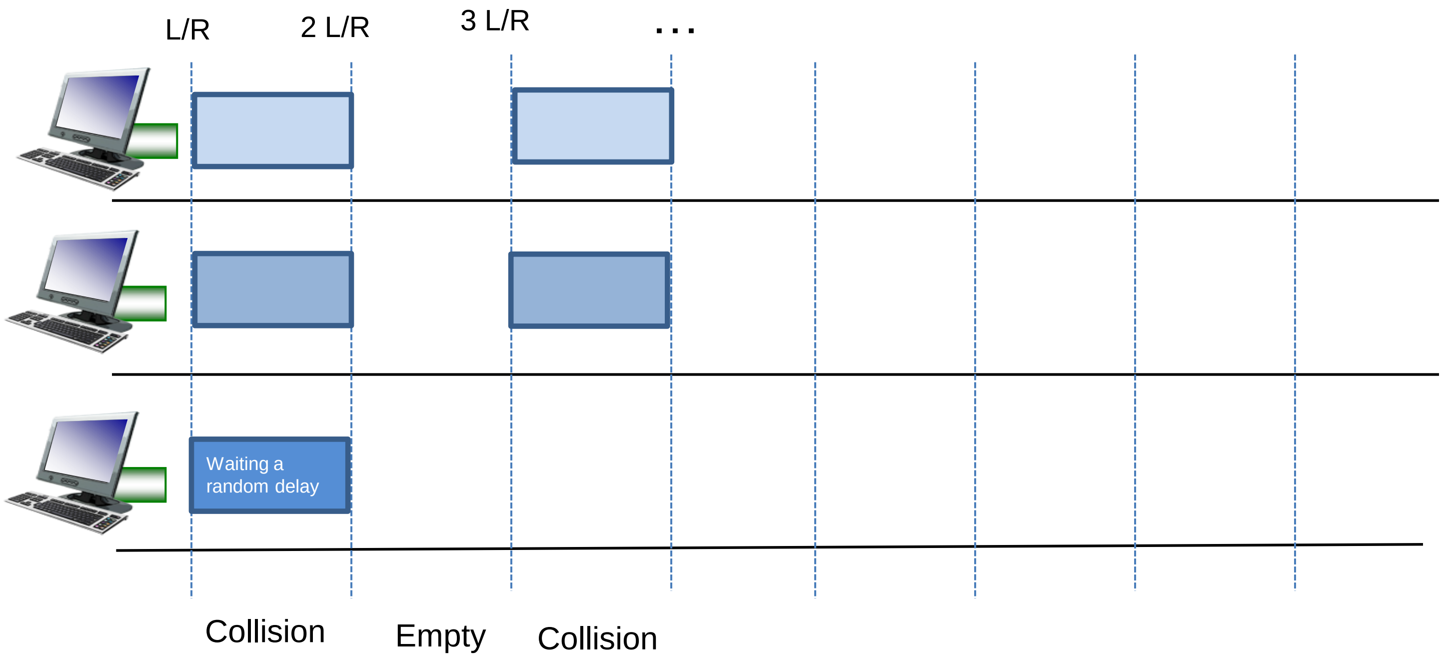
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Random Access

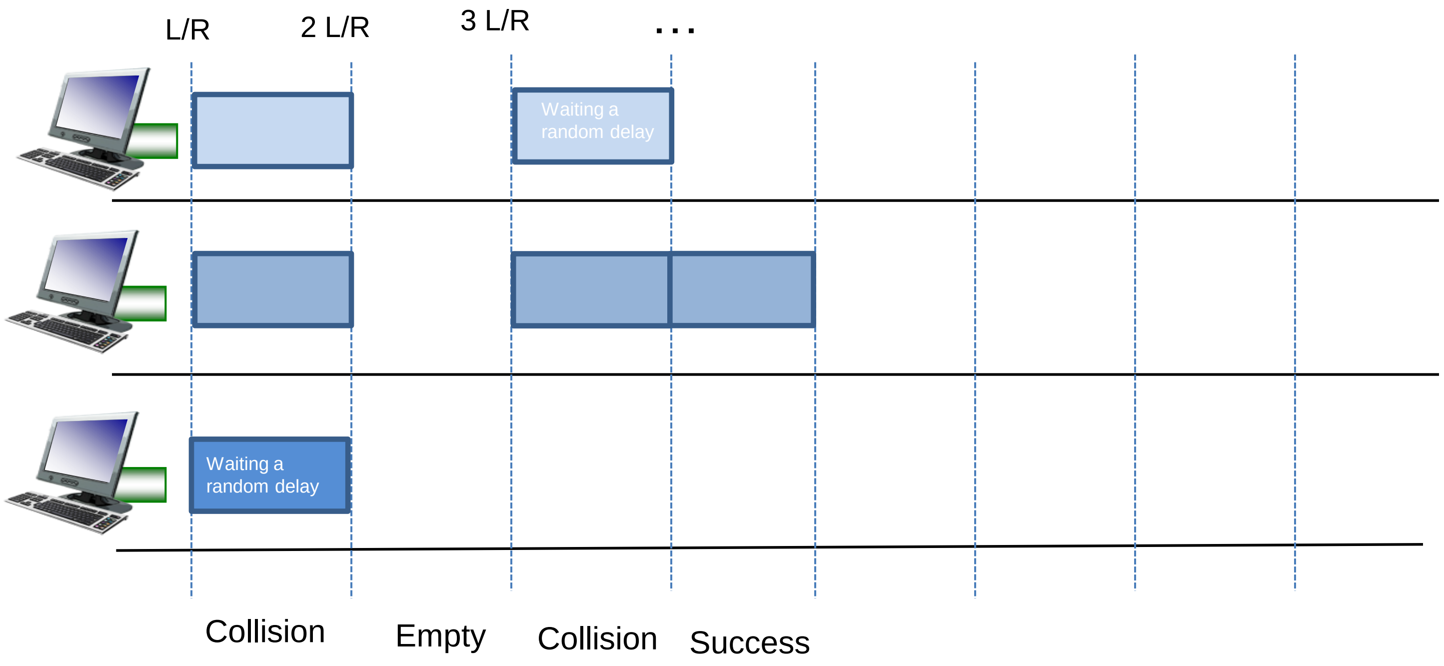
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R "slots"

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Random Access

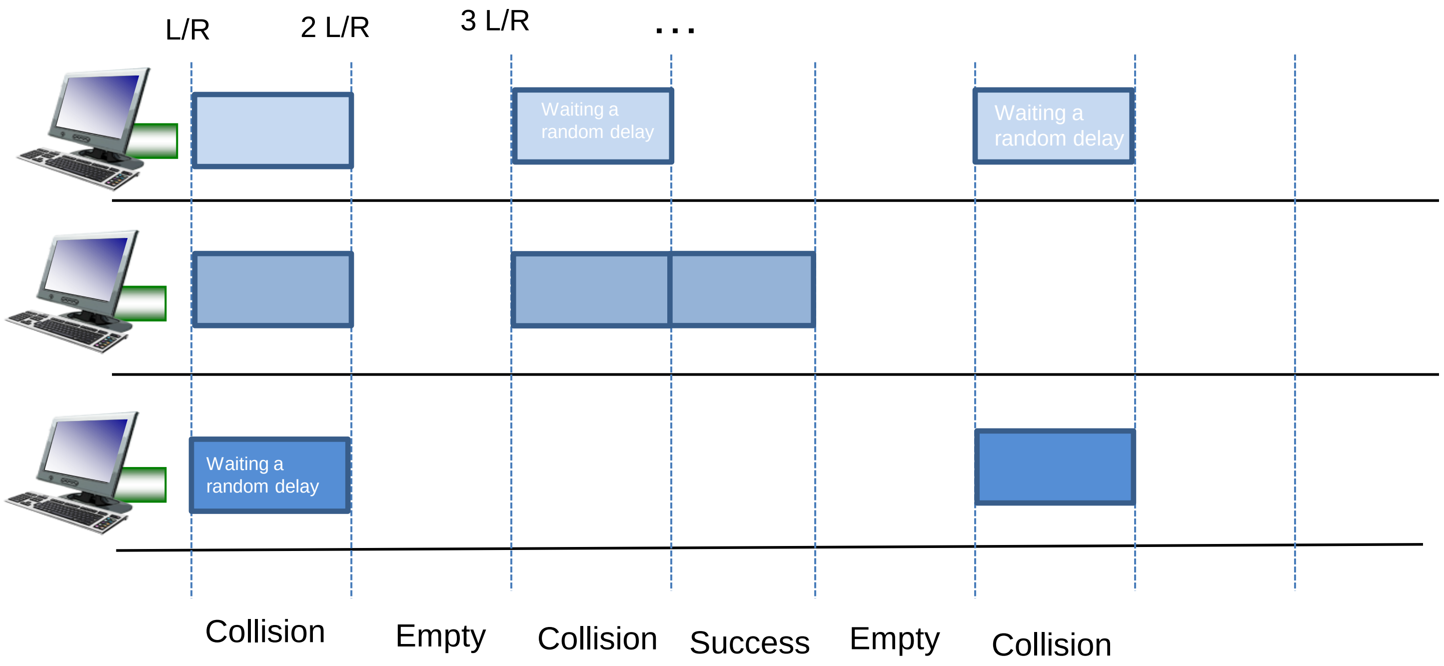
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R "slots"

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Random Access

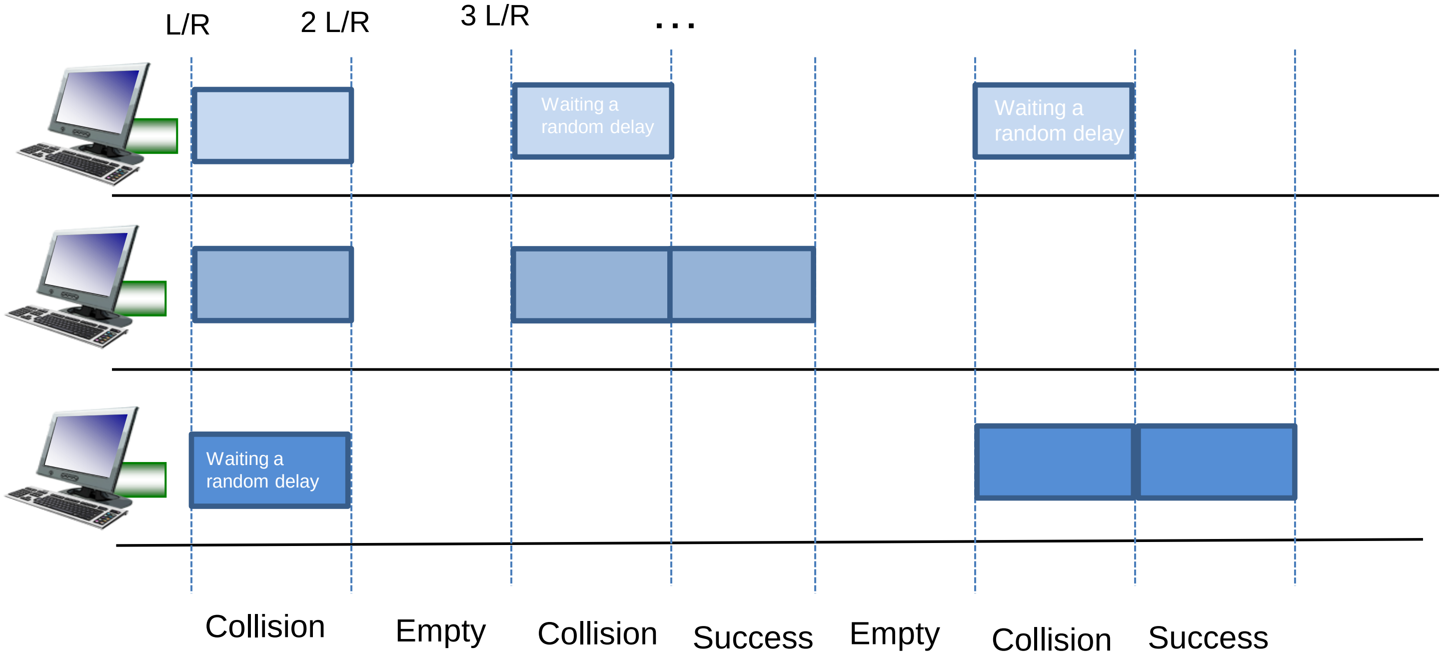
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R "slots"

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Random Access

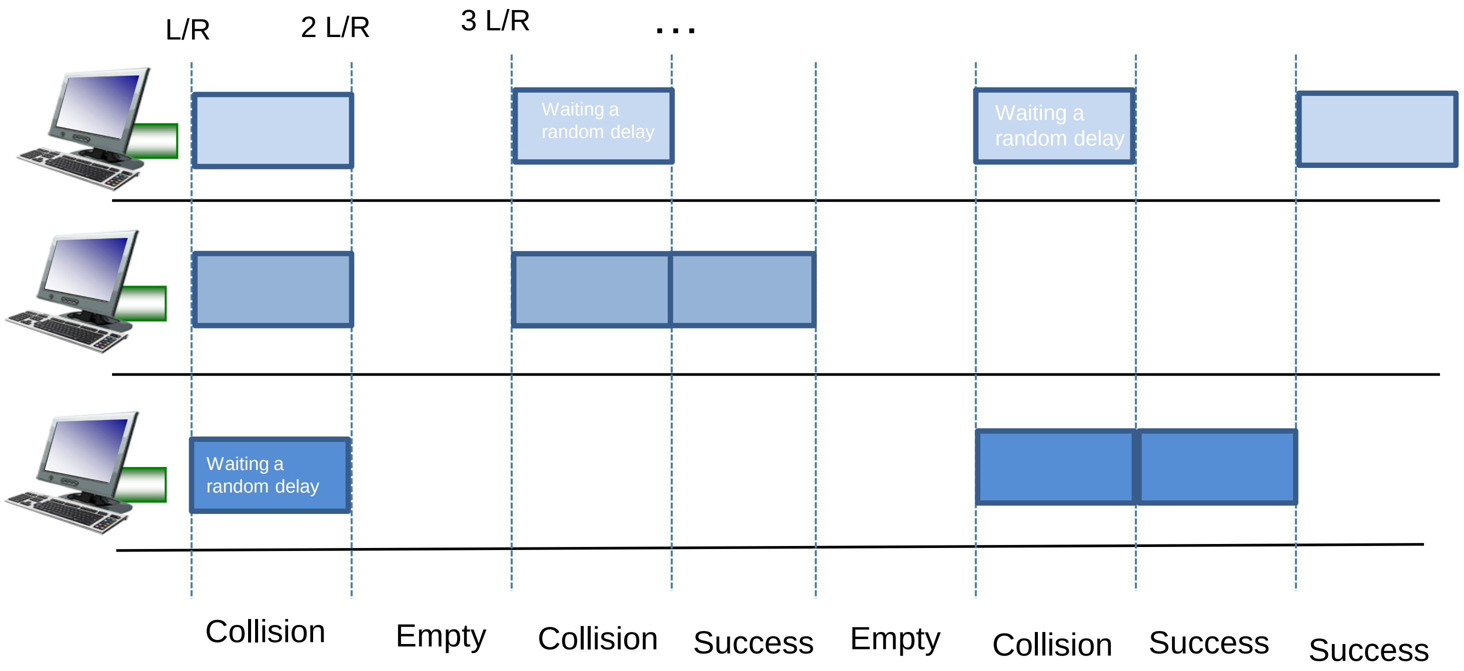
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

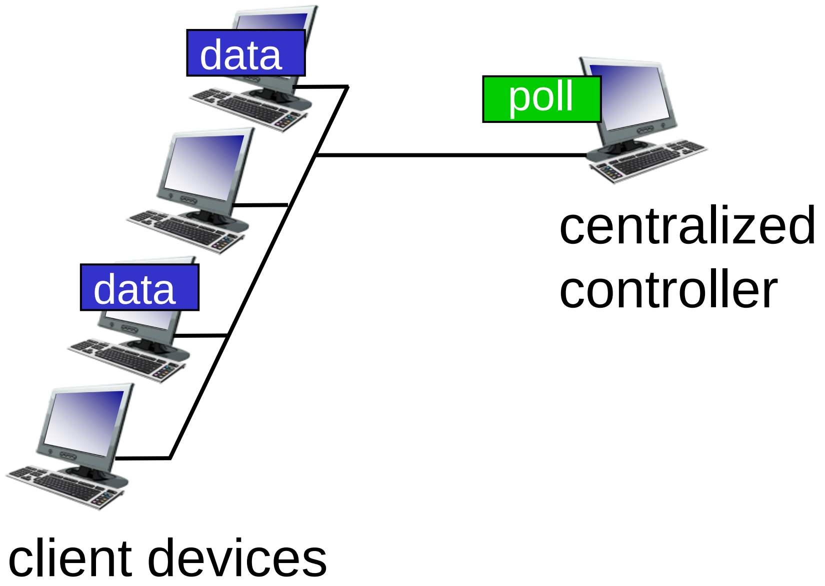
L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



polling:

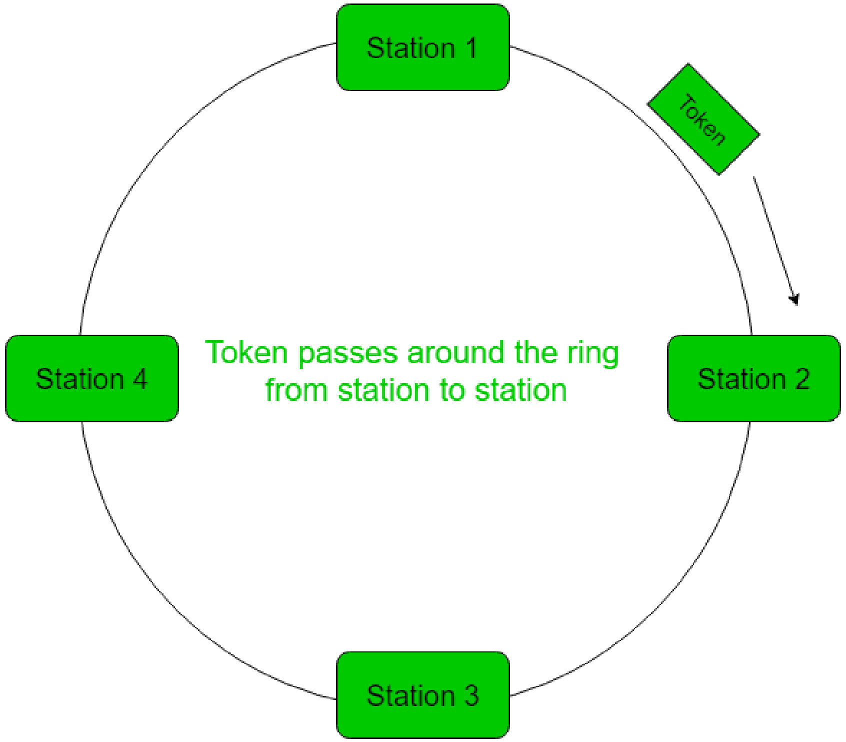
- centralized controller "invites" other nodes to transmit in turn
- typically used with "dumb" devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)
- Bluetooth uses polling



Taking Turns

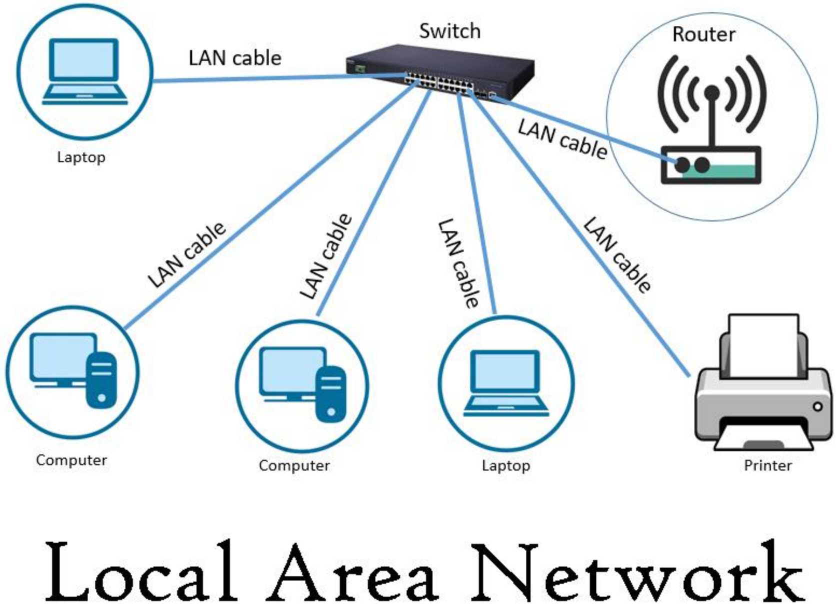
Token Passing
Nodes are connected in a circular manner, and pass a special frame (token) between each other

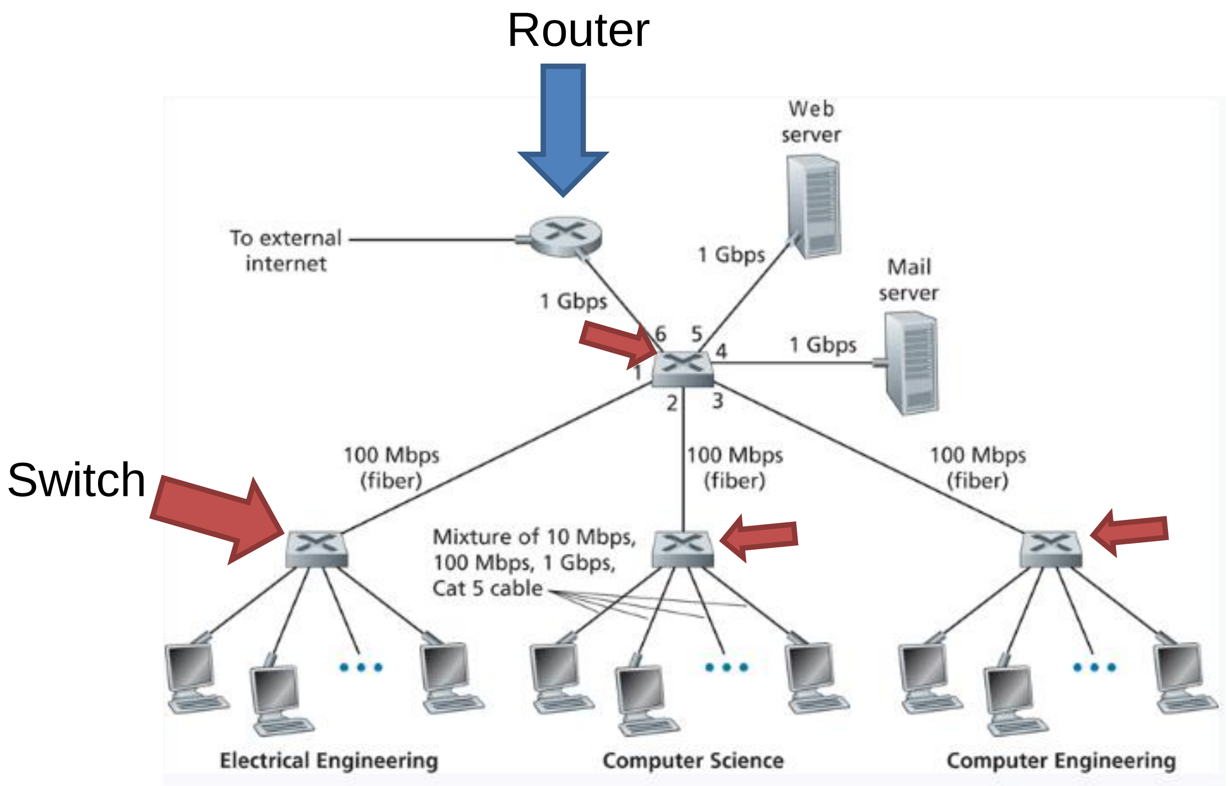
Can only transmit messages if you have the token



LAN

Local Area Network (LAN)- A collection of devices in one physical location, typically that share a centralized internet connection





(Within a LAN, we could have several Subnets)

MAC addresses

- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
  - e.g.: 128.119.40.136
- MAC (or LAN or physical or Ethernet) address:
  - function: used "locally" to get frame from one interface to another physically-connected interface (same subnet, in IP-addressing sense)
  - 48-bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
  - e.g.: 1A-2F-BB-76-09-AD

hexadecimal (base 16) notation
(each "numeral" represents 4 bits)

Why do we need MAC addresses?

We need a way to *physically identify* a device on a network

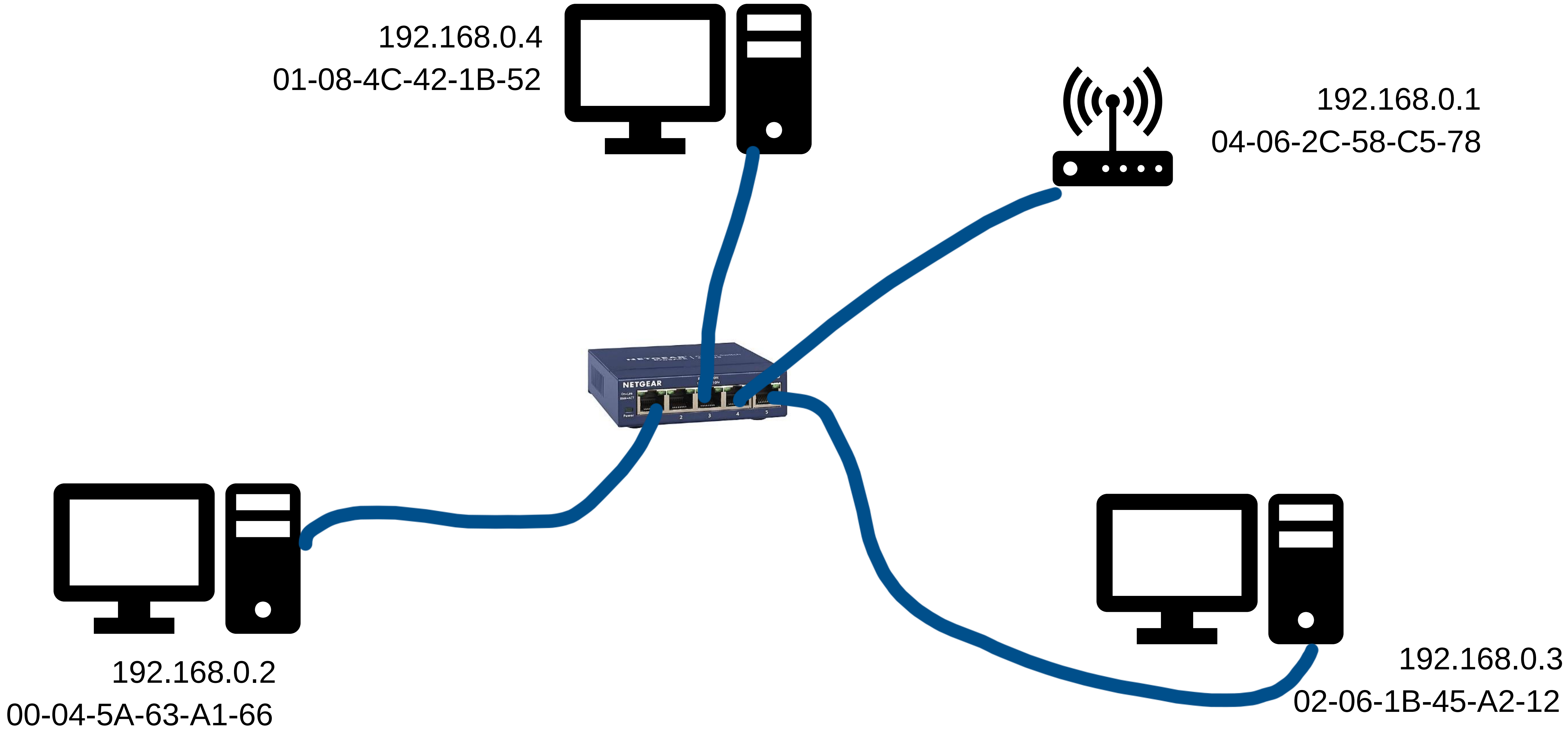
IP addresses change frequently, but a MAC address will always be the same

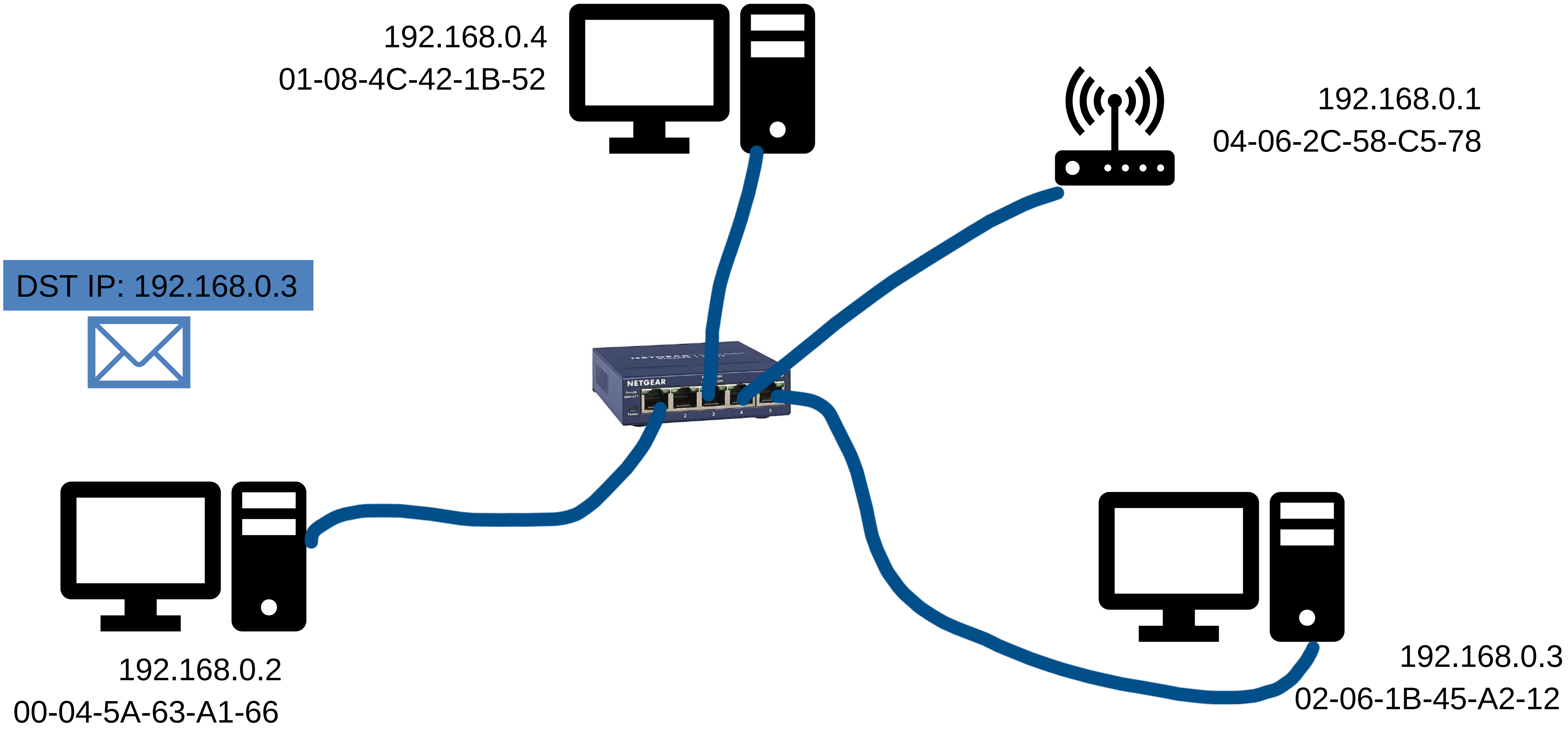
An IP address is used to locate a device, a MAC address is used to identify a device

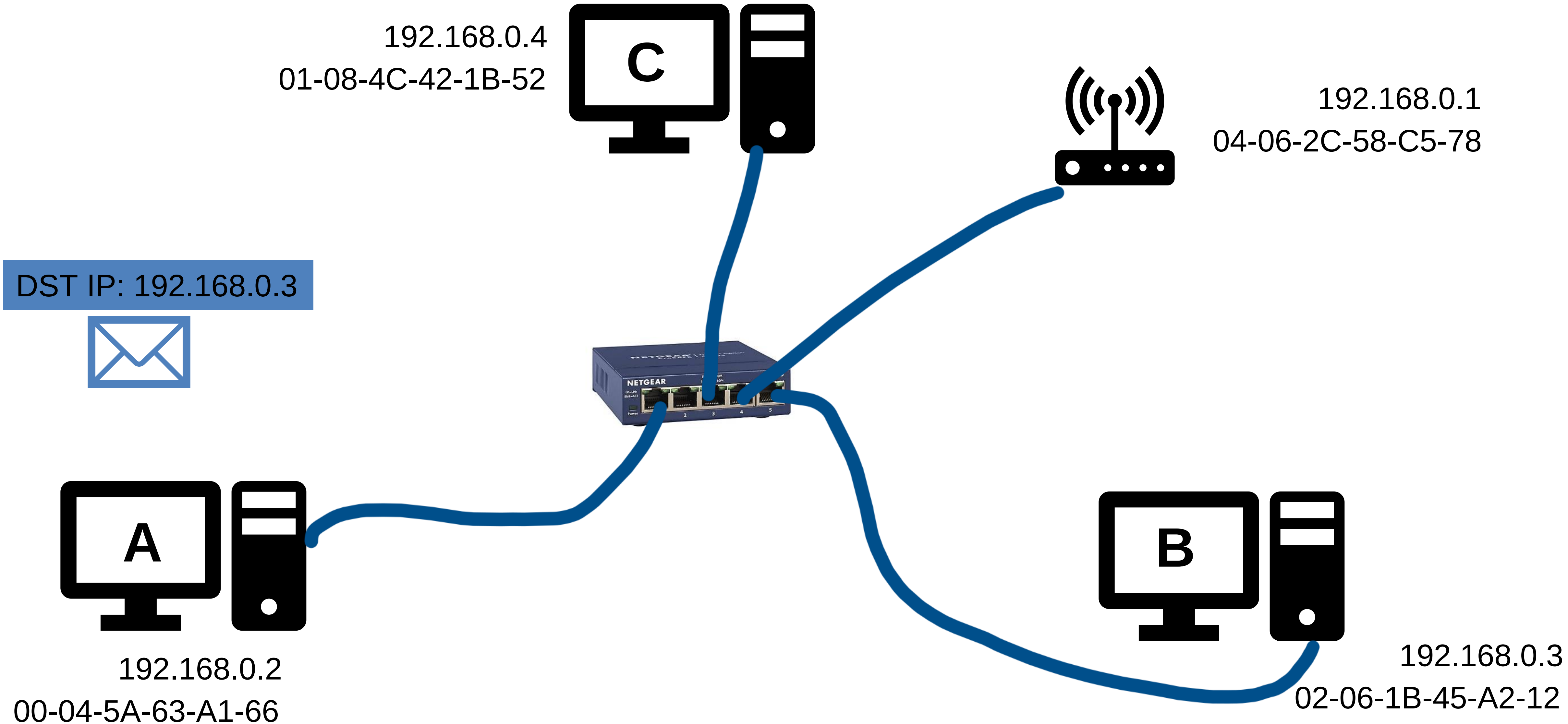
IP Address = Street Address,   MAC Address = Name of person living in House

We need both an IP address and a MAC address to transmit a message

Additionally, Ethernet and WiFi are all designed to use MAC address, not IP address


192.168.0.4
01-08-4C-42-1B-52
192.168.0.1
04-06-2C-58-C5-78
NETGEAR
192.168.0.2
00-04-5A-63-A1-66
192.168.0.3
02-06-1B-45-A2-12


192.168.0.4
01-08-4C-42-1B-52
192.168.0.1
04-06-2C-58-C5-78
DST IP: 192.168.0.3
NETGEAR
192.168.0.2
00-04-5A-63-A1-66
192.168.0.3
02-06-1B-45-A2-12



We need Computer B’s MAC address!

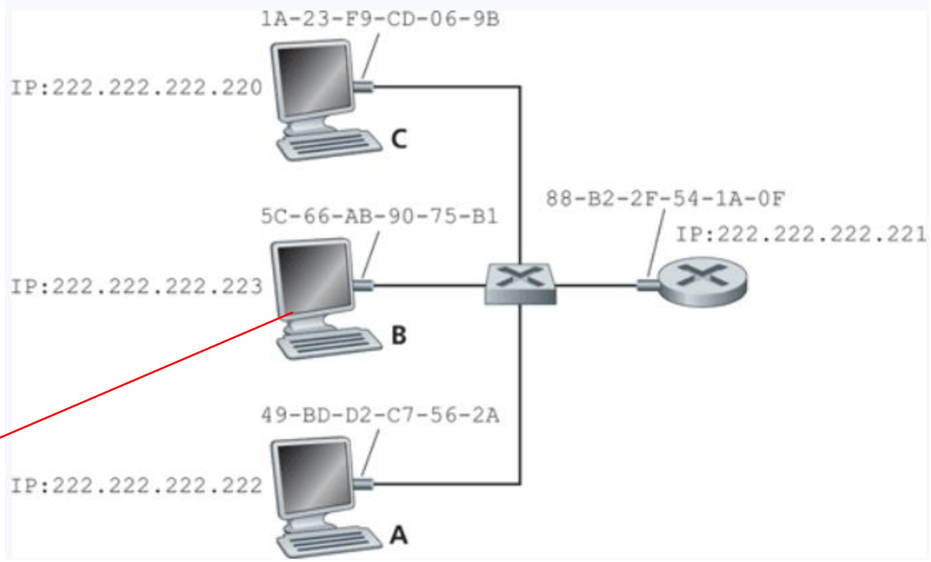
ARP

Protocol for mapping **IP Addresses** to **MAC addresses**

Used *only* for hosts and router interfaces **on the same subnet**

First the machine checks its **ARP table**

IP Address	MAC Address	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00



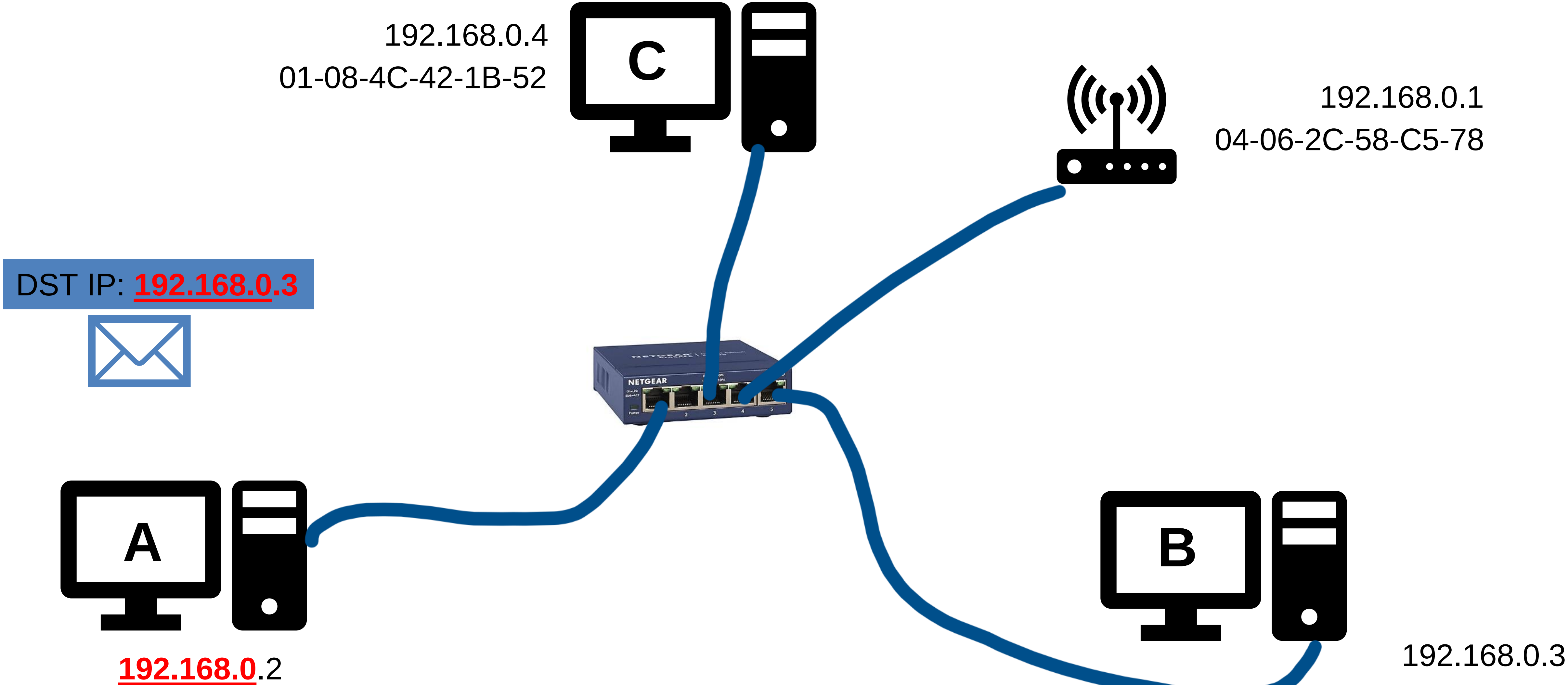
If the entry does not exist in the table, construct and send an **ARP** packet

Broadcasts the ARP packet to all interfaces on the LAN (255.255.255.255)

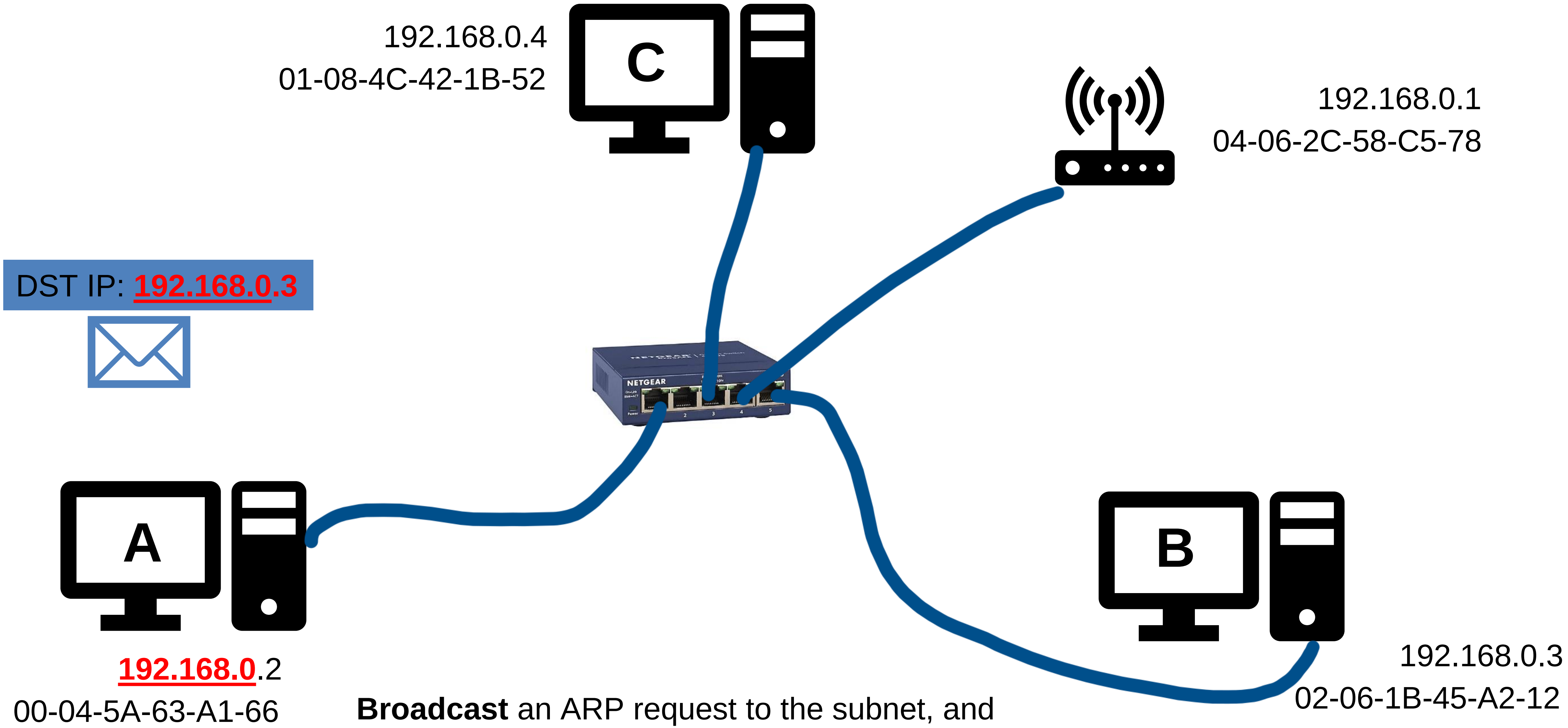
These tables are self-updated, and do not require manual entry*

ARP

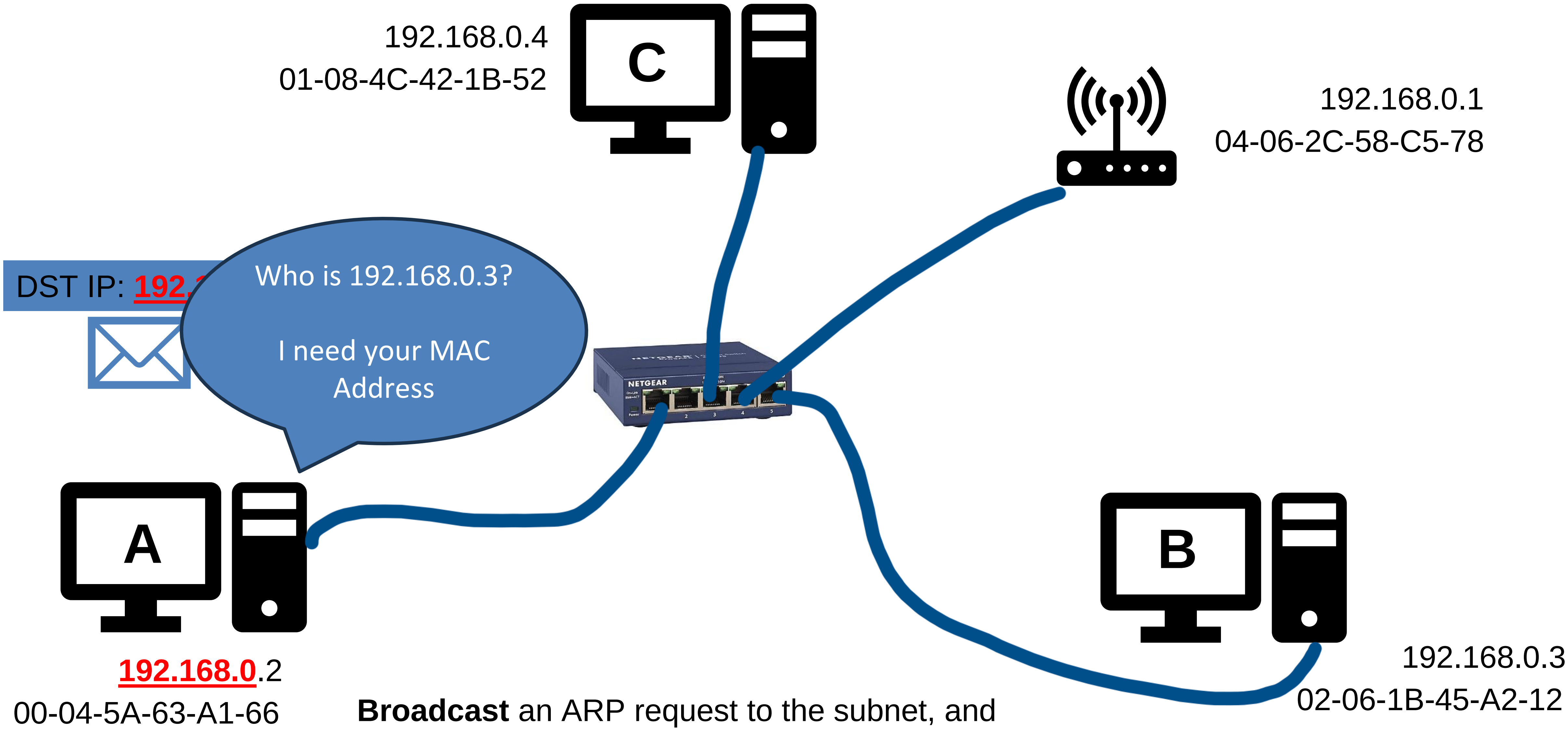
- A wants to send datagram to B
  - B's MAC address not in A's ARP table.
- A **broadcasts** ARP query packet, containing B's IP address
  - destination MAC address = FF-FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*



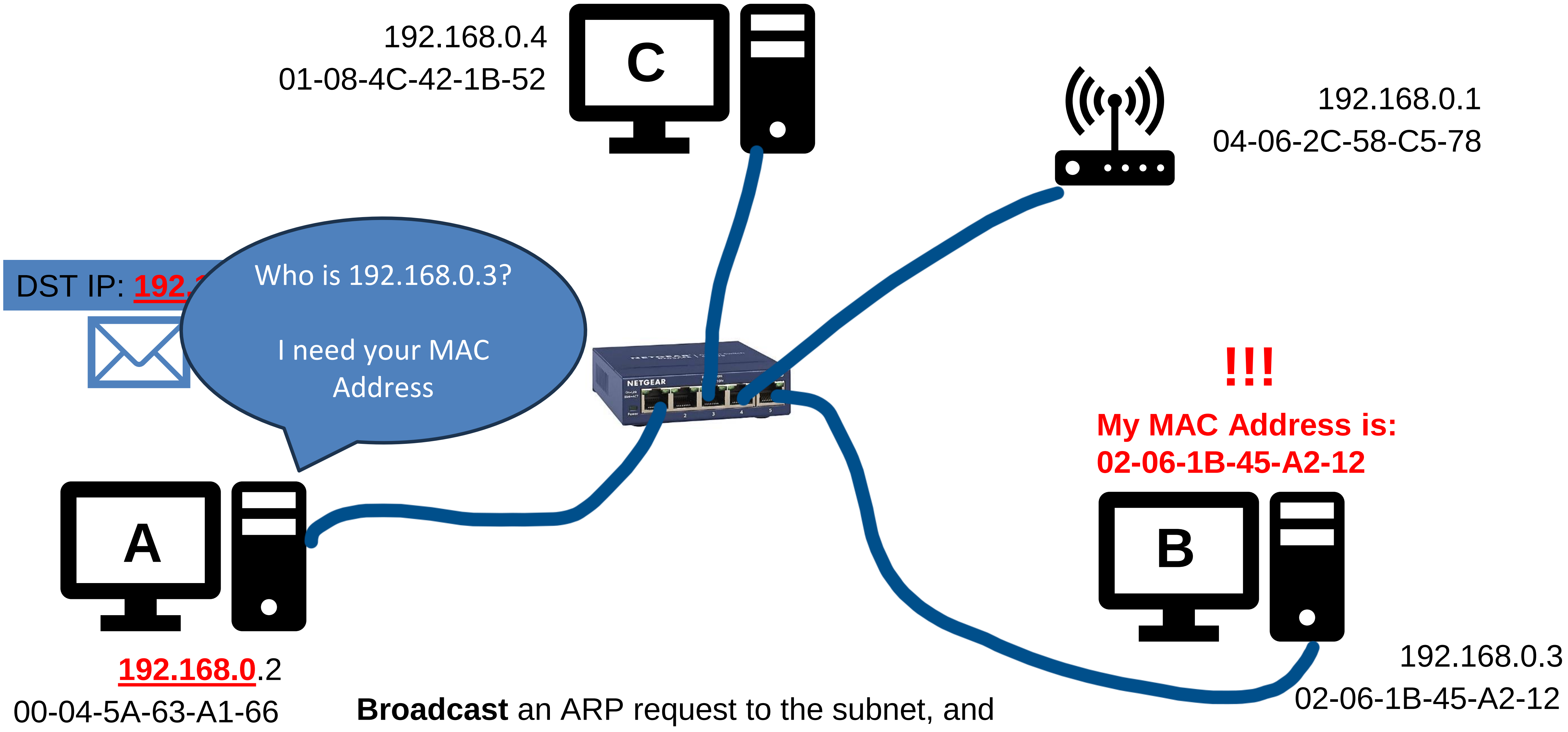
Computer A see’s that this IP is in its “group”, so it knows that Computer B must be in the same subnet!



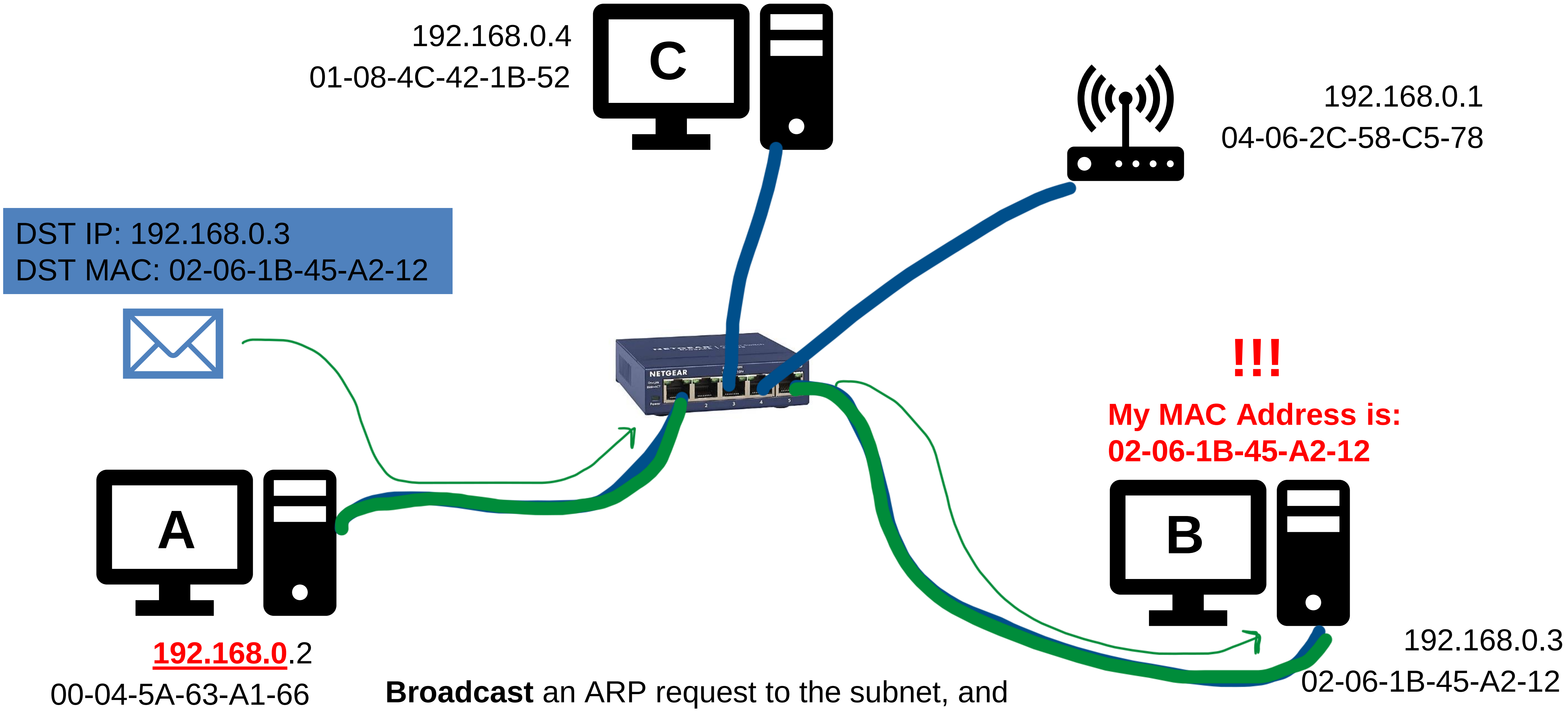
Broadcast an ARP request to the subnet, and hope that we get a response back



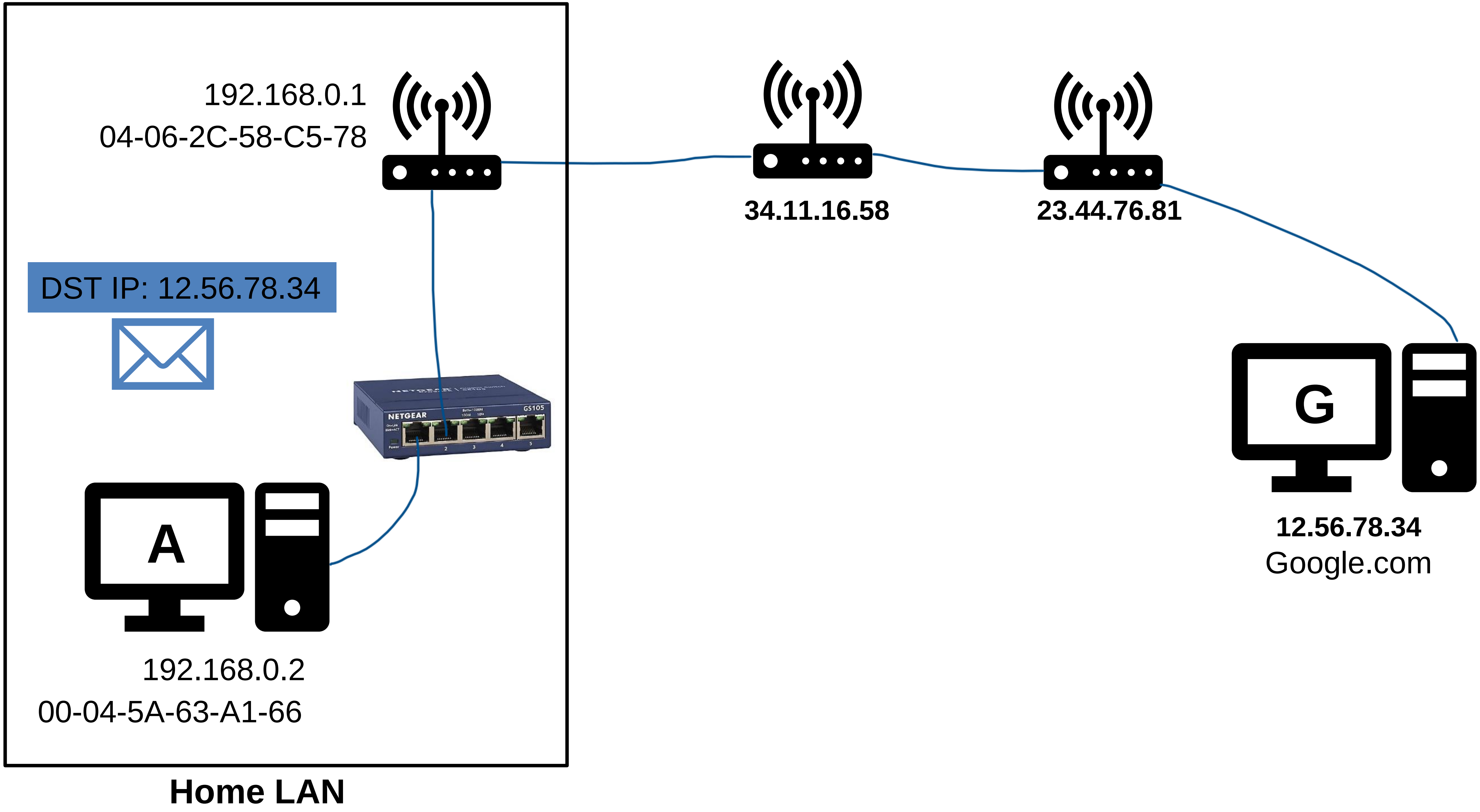
Broadcast an ARP request to the subnet, and hope that we get a response back

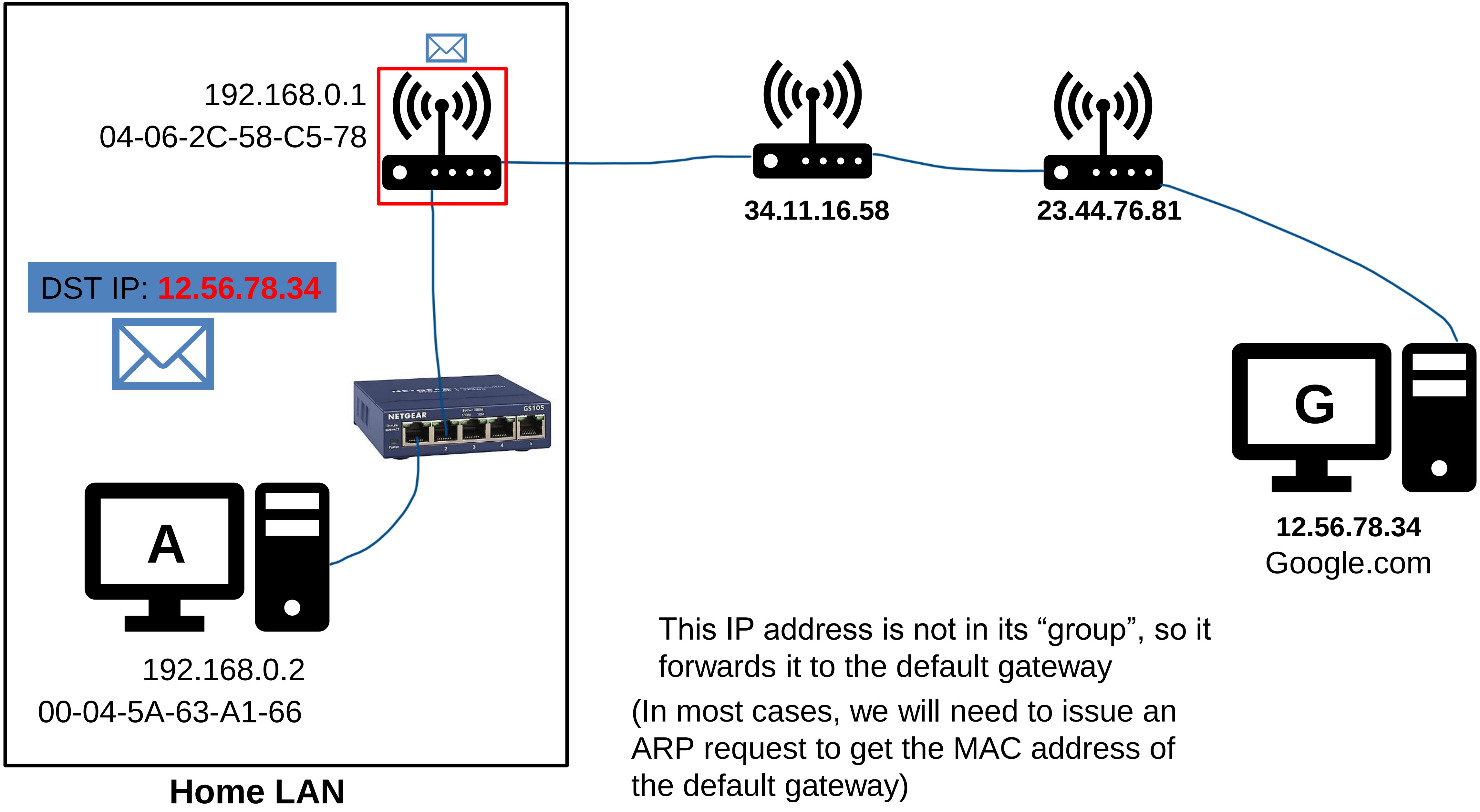


Broadcast an ARP request to the subnet, and hope that we get a response back



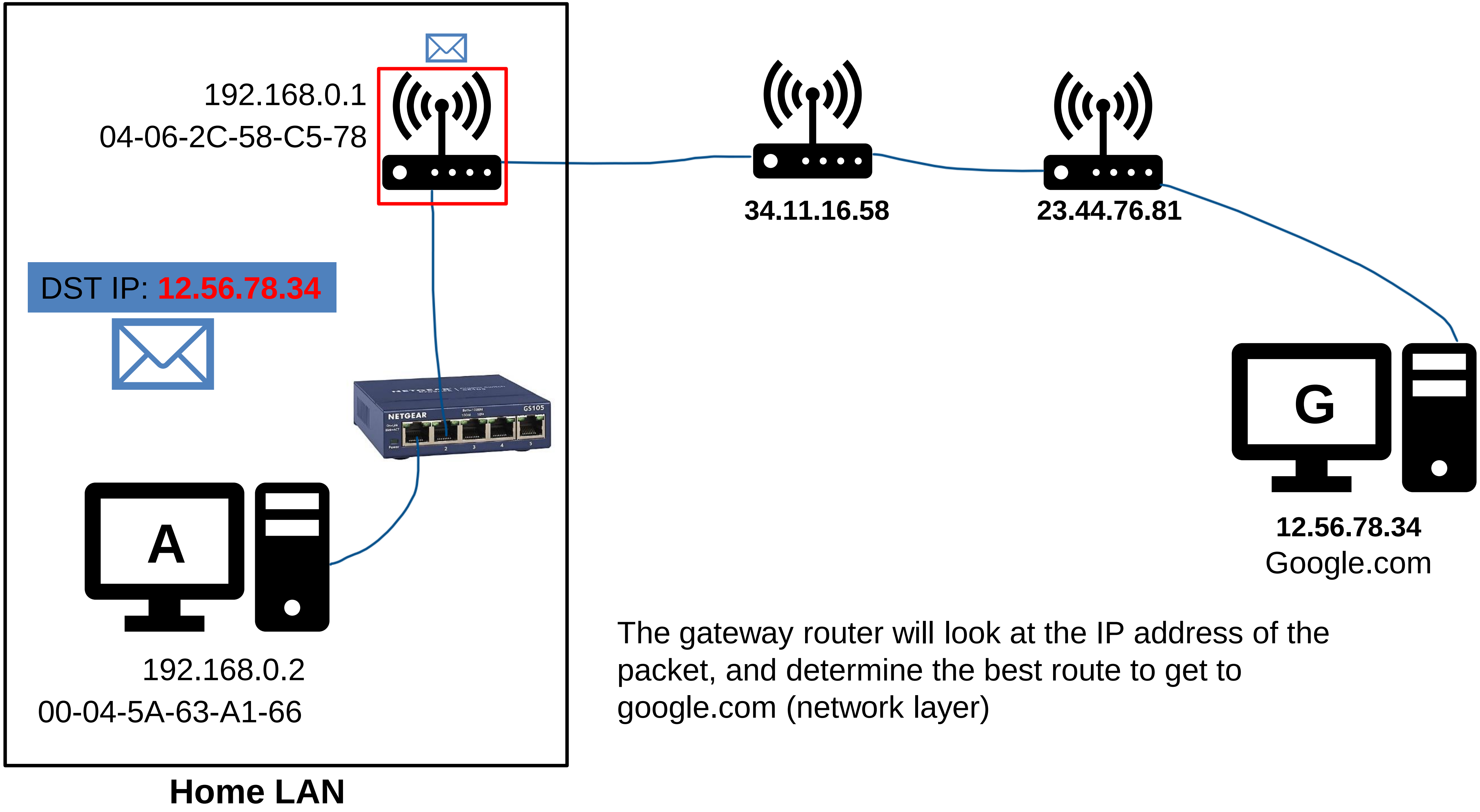
Broadcast an ARP request to the subnet, and hope that we get a response back


192.168.0.1
04-06-2C-58-C5-78
34.11.16.58
23.44.76.81
DST IP: 12.56.78.34
NETGEAR
GS105
A
192.168.0.2
00-04-5A-63-A1-66
Home LAN
G
12.56.78.34
Google.com

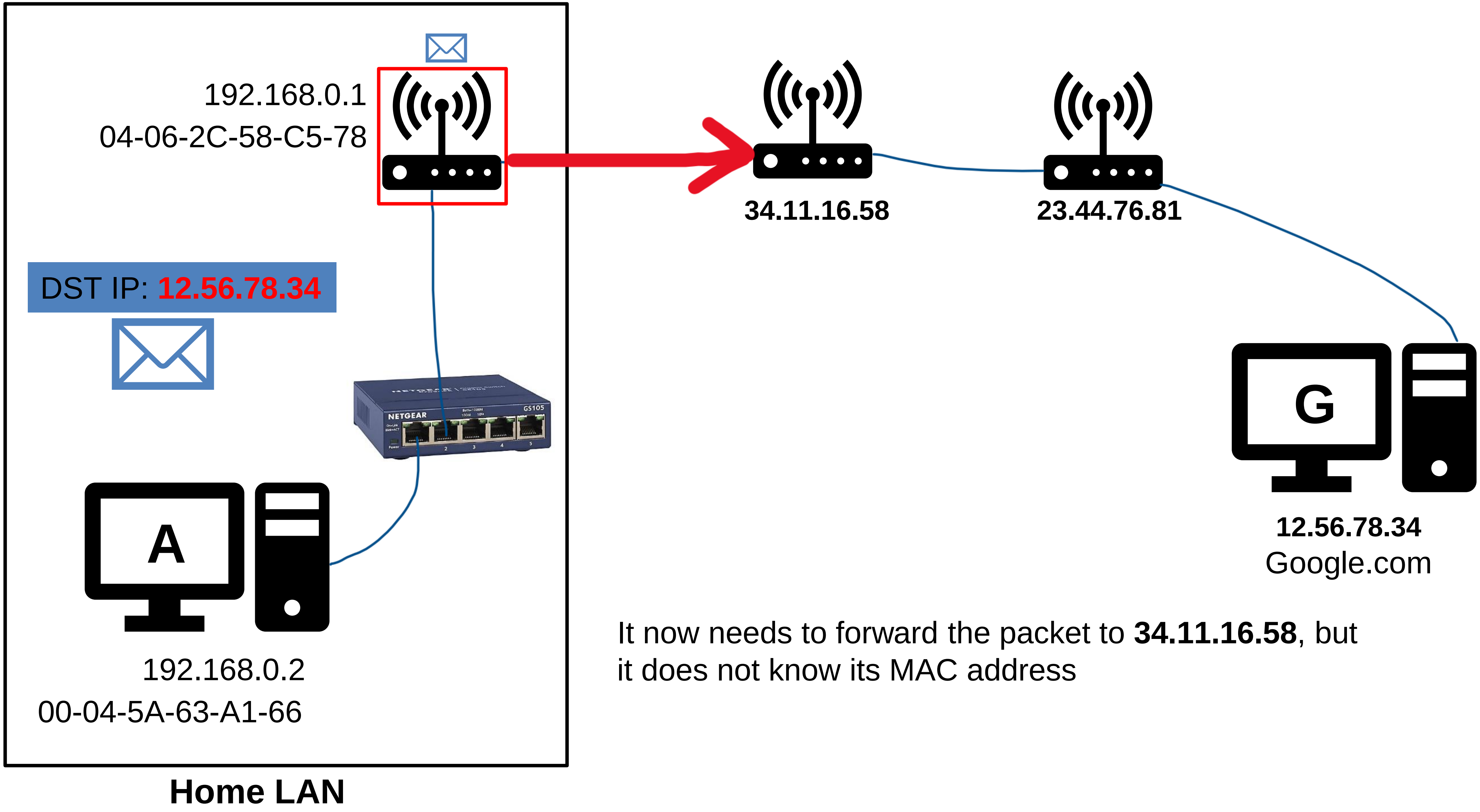


This IP address is not in its “group”, so it forwards it to the default gateway

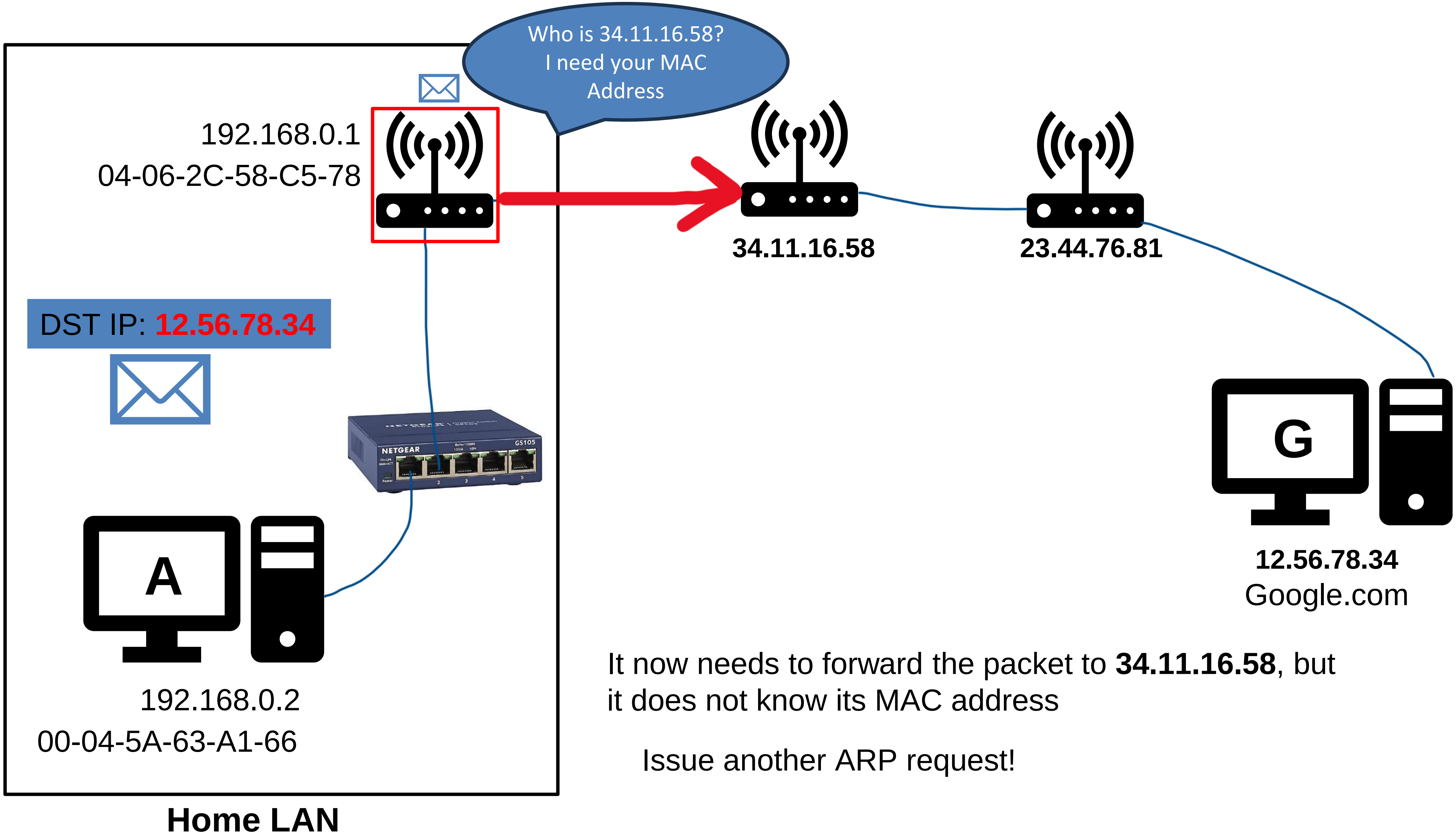
(In most cases, we will need to issue an ARP request to get the MAC address of the default gateway)



The gateway router will look at the IP address of the packet, and determine the best route to get to google.com (network layer)

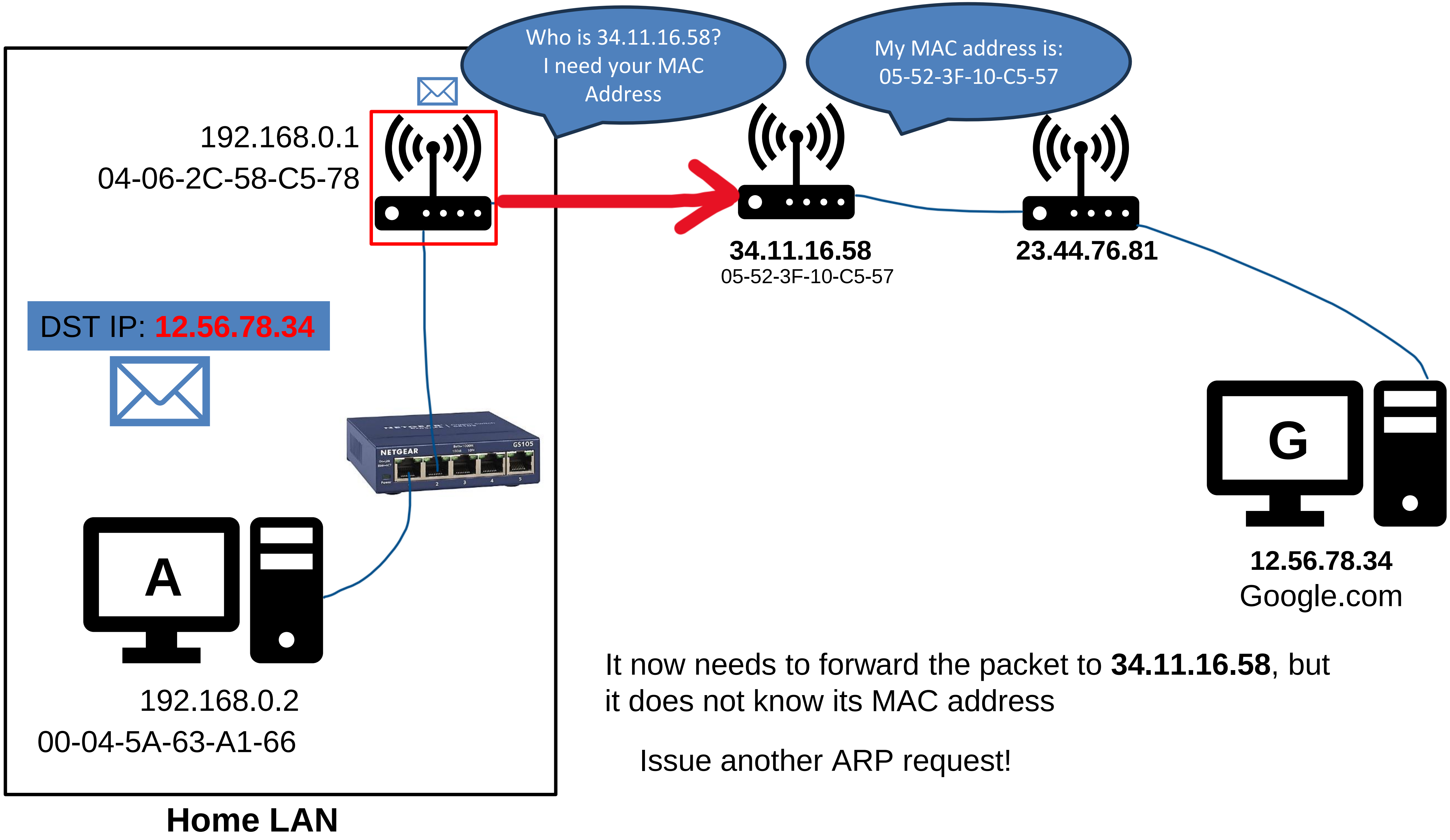


It now needs to forward the packet to **34.11.16.58**, but it does not know its MAC address



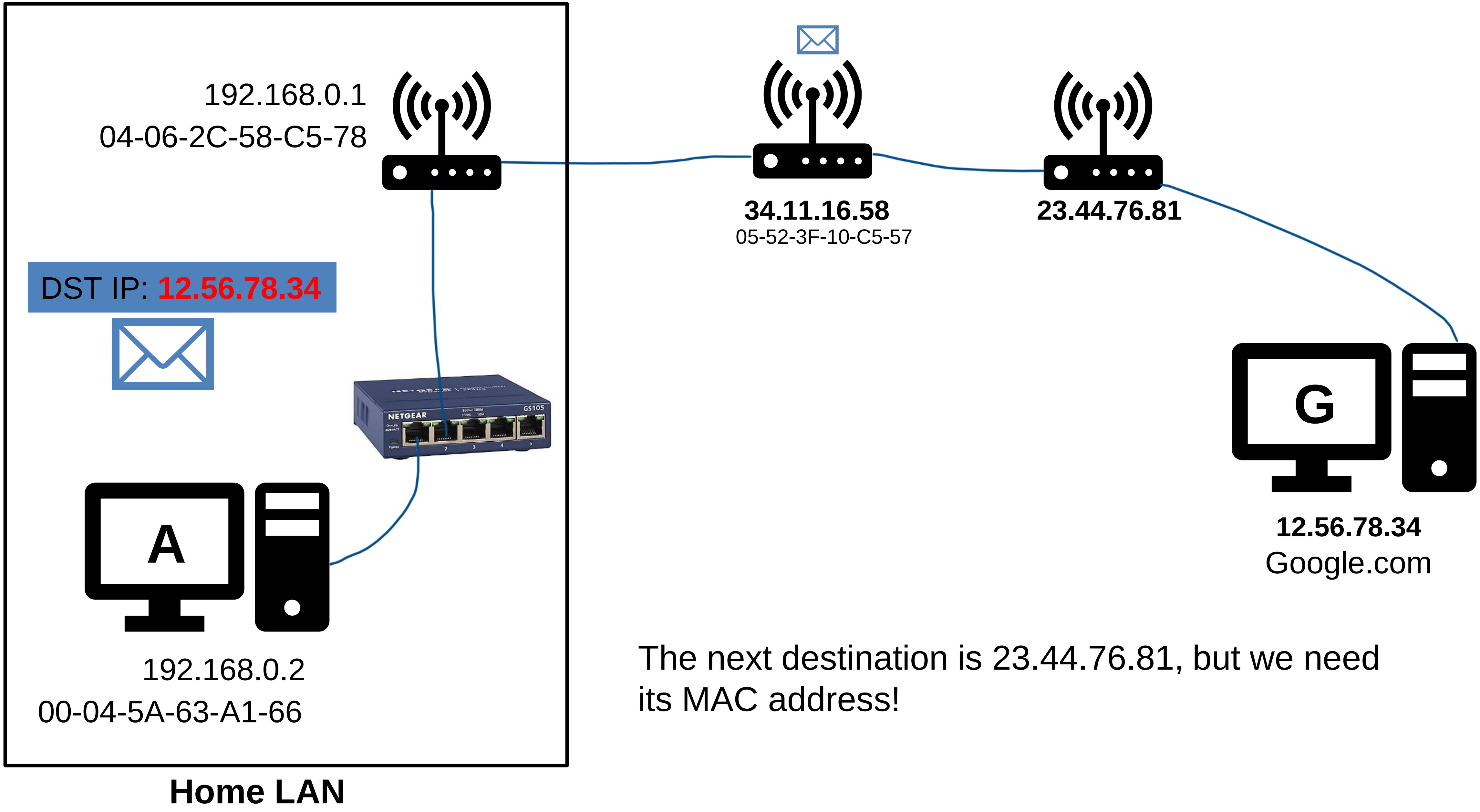
It now needs to forward the packet to **34.11.16.58**, but it does not know its MAC address

Issue another ARP request!

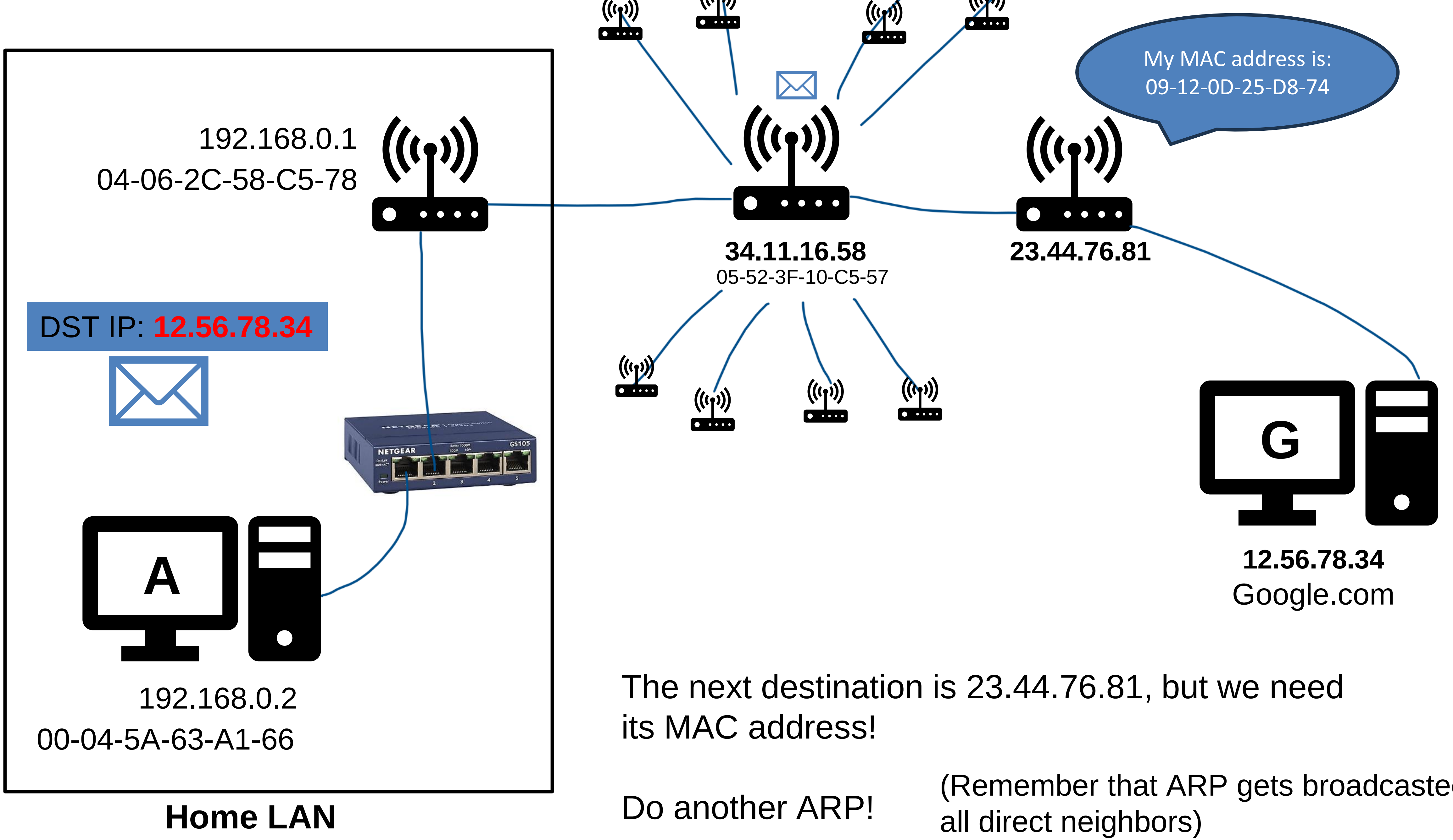


It now needs to forward the packet to **34.11.16.58**, but it does not know its MAC address

Issue another ARP request!



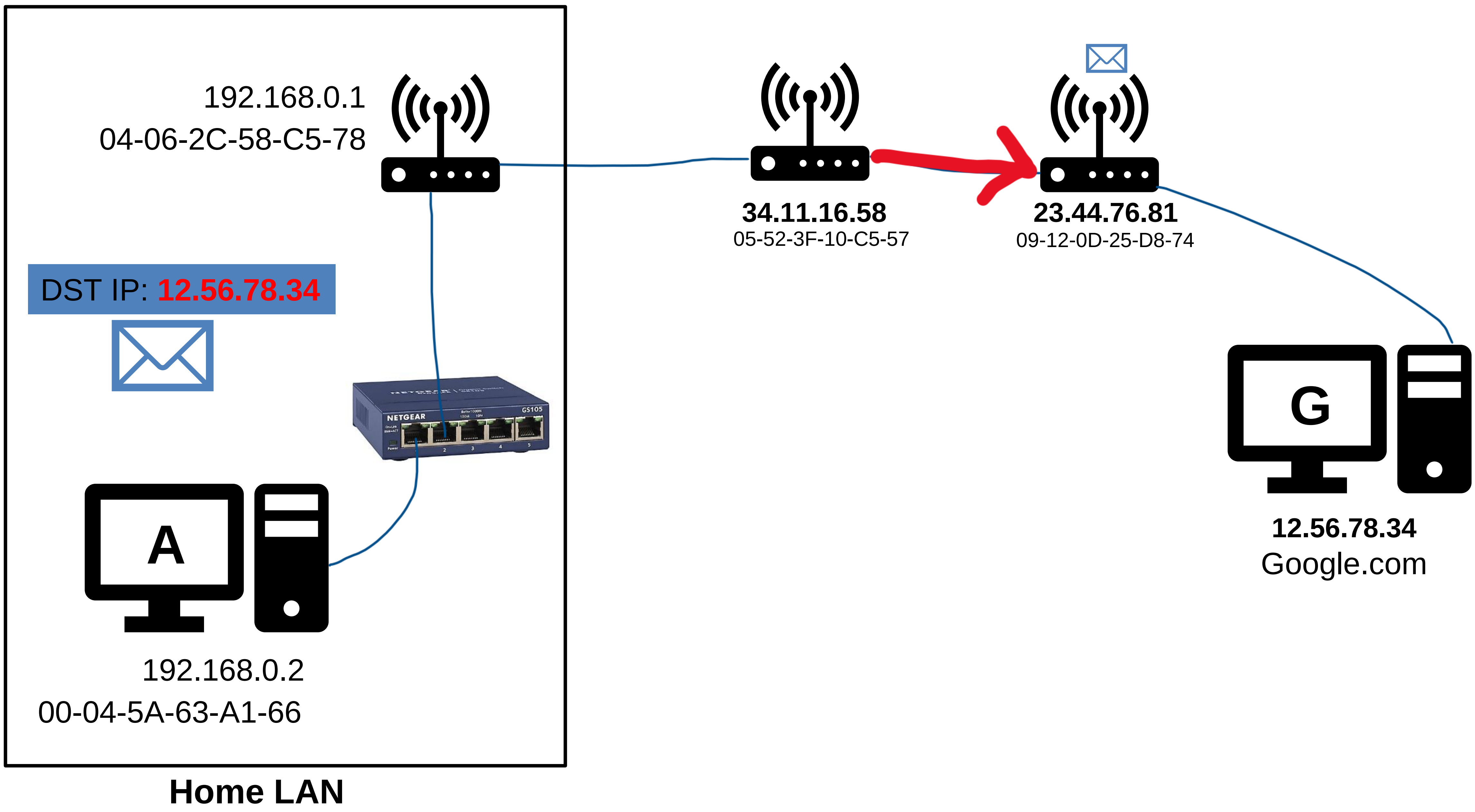
The next destination is 23.44.76.81, but we need its MAC address!



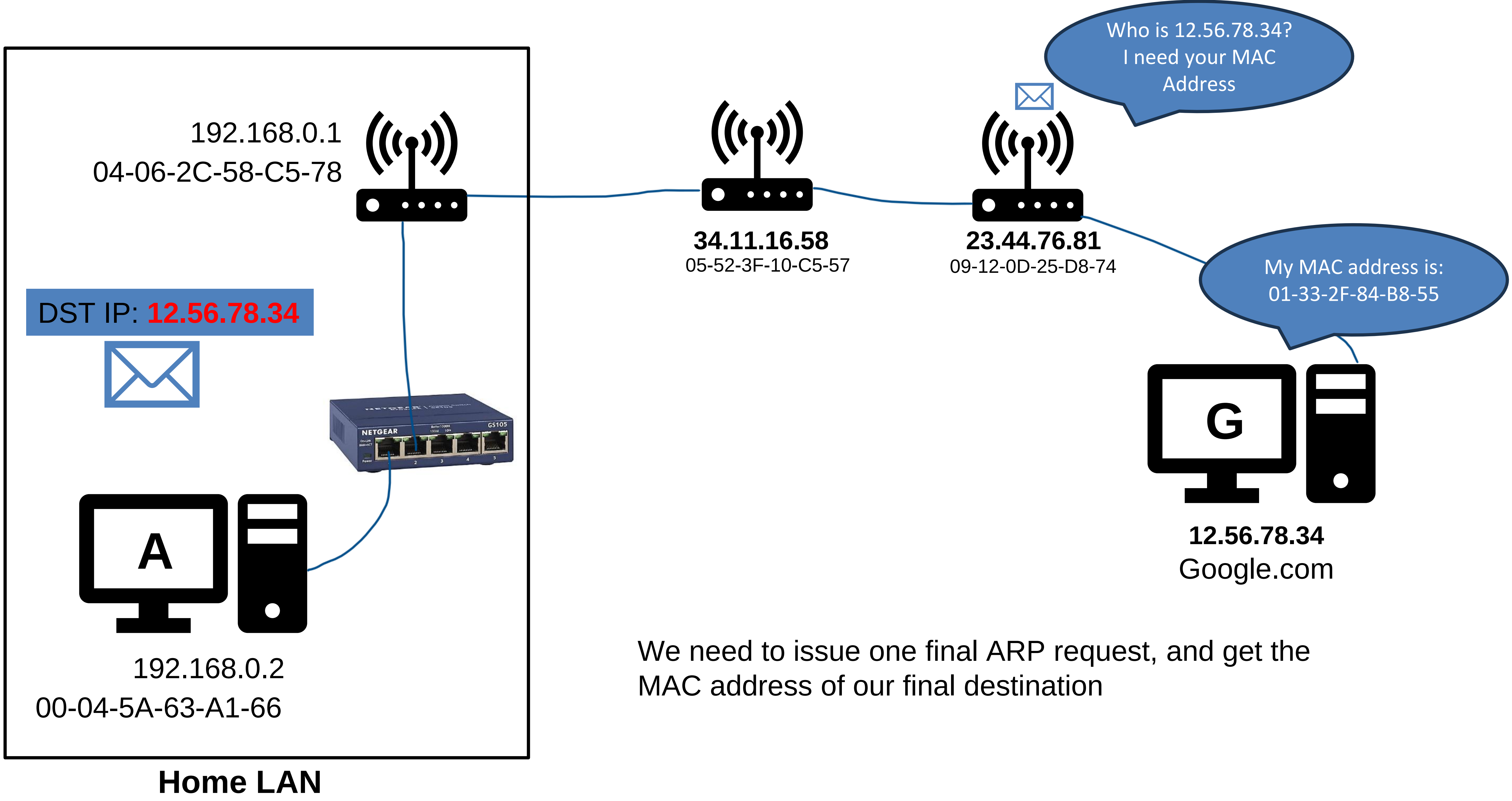
The next destination is 23.44.76.81, but we need its MAC address!

Do another ARP!

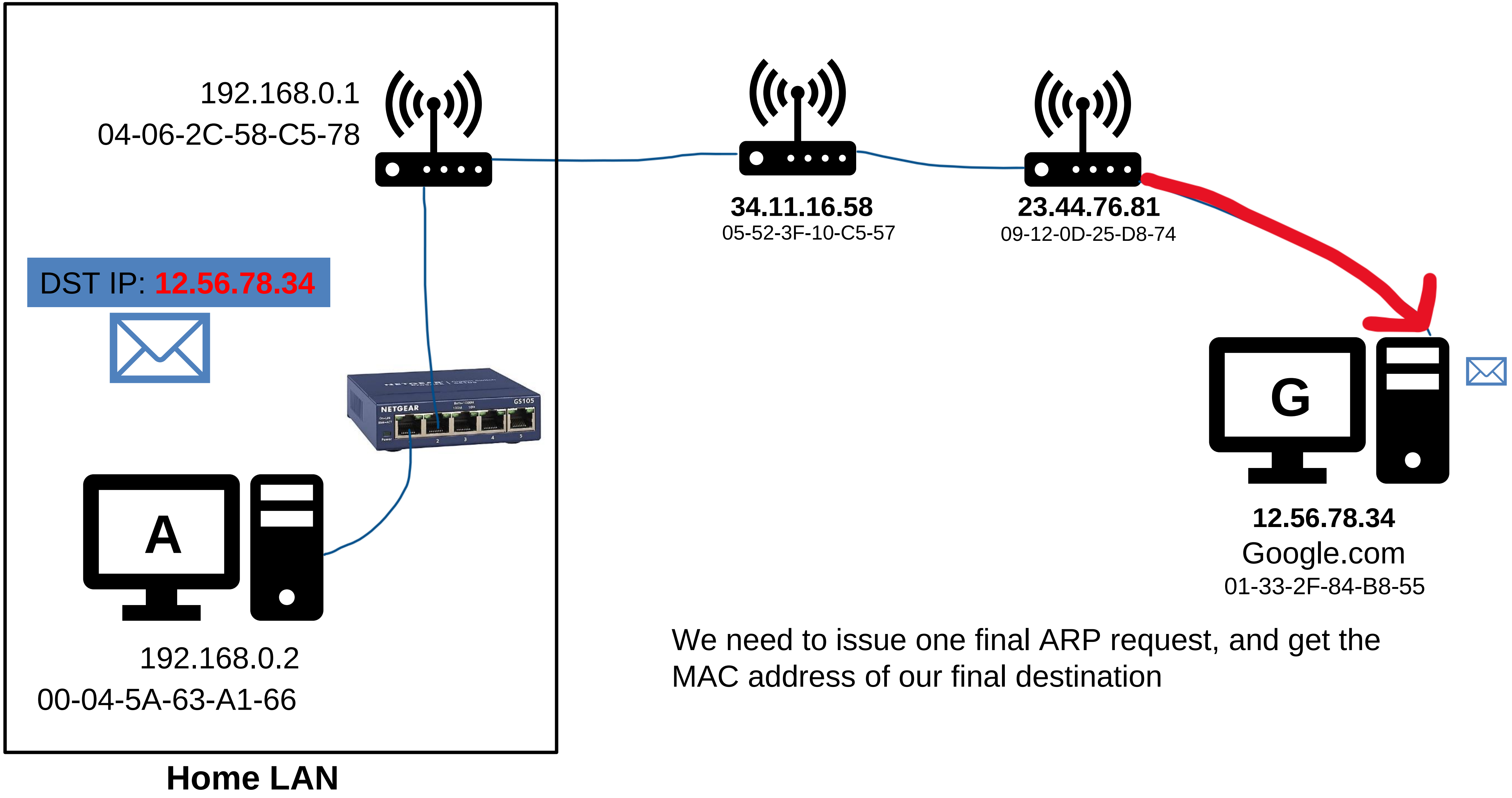
(Remember that ARP gets broadcasted to all direct neighbors)



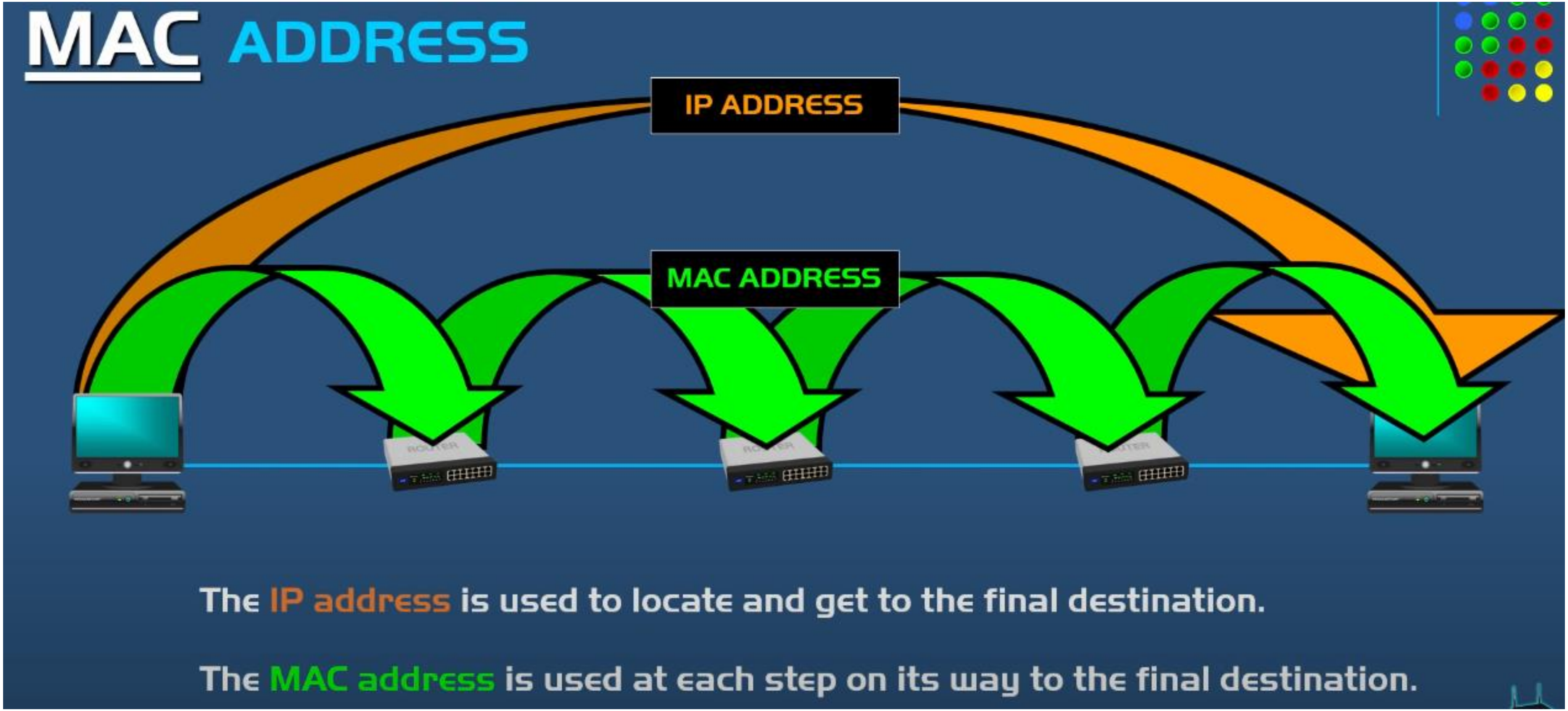
192.168.0.1
04-06-2C-58-C5-78
34.11.16.58
05-52-3F-10-C5-57
23.44.76.81
09-12-0D-25-D8-74
DST IP: 12.56.78.34
NETGEAR
GS105
G
12.56.78.34
Google.com
A
192.168.0.2
00-04-5A-63-A1-66
Home LAN



We need to issue one final ARP request, and get the MAC address of our final destination



We need to issue one final ARP request, and get the MAC address of our final destination

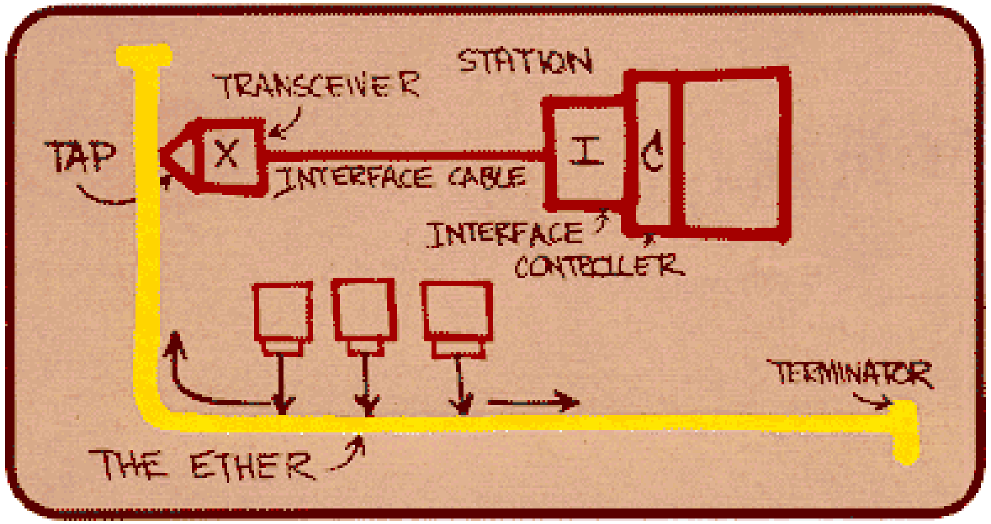

MAC ADDRESS
IP ADDRESS
MAC ADDRESS
The IP address is used to locate and get to the final destination.
The MAC address is used at each step on its way to the final destination.

Finding your MAC Address

Ipconfig/all

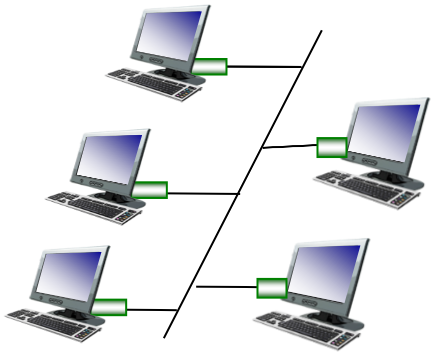
Ethernet

- “dominant” wired LAN technology:
- single chip, multiple speeds (e.g., Broadcom BCM5761)
- first widely used LAN technology
- simpler, cheap
- kept up with speed race: 10 Mbps – 10 Gbps



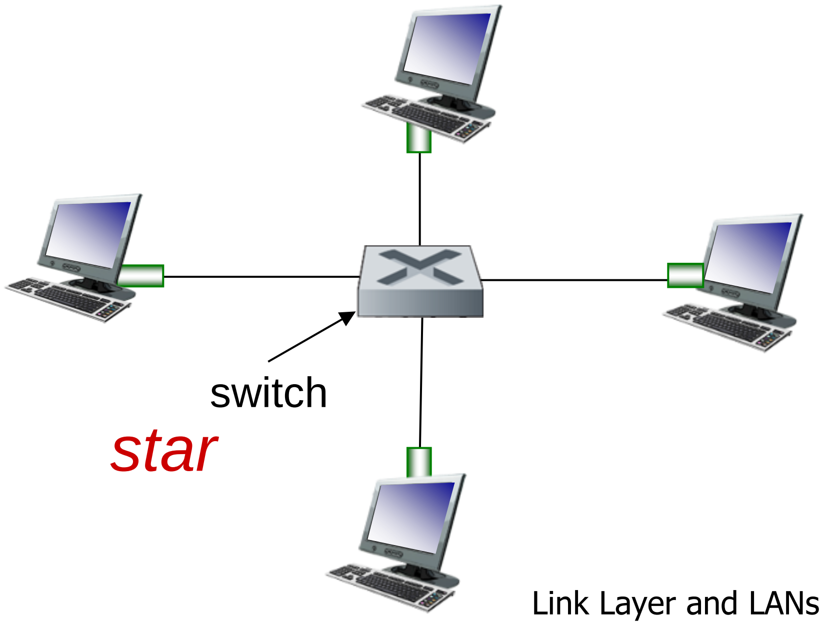
Metcalfe’s Ethernet sketch

Ethernet Topology



bus: coaxial cable

(outdated)

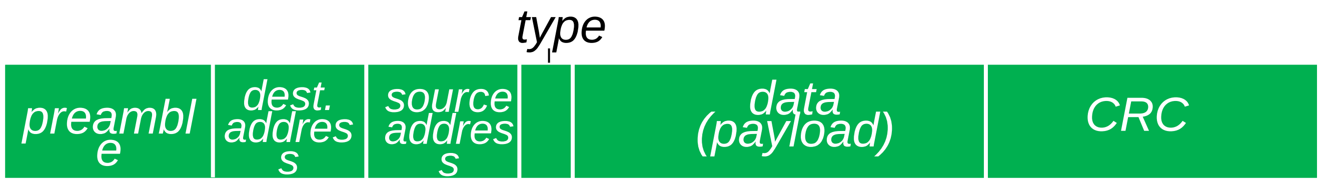


star

Link Layer and LANs

Ethernet frame structure

sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame



preamble:

7 bytes with pattern 10101010 followed by one byte with pattern 10101011

used to synchronize receiver, sender clock rates

Ethernet frame structure (more)

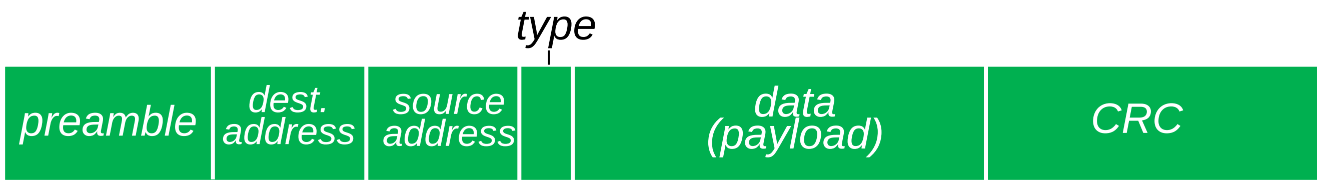
addresses: 6 byte source, destination MAC addresses

- if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame to network layer protocol
- otherwise, adapter discards frame

type: indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)

CRC: cyclic redundancy check at receiver

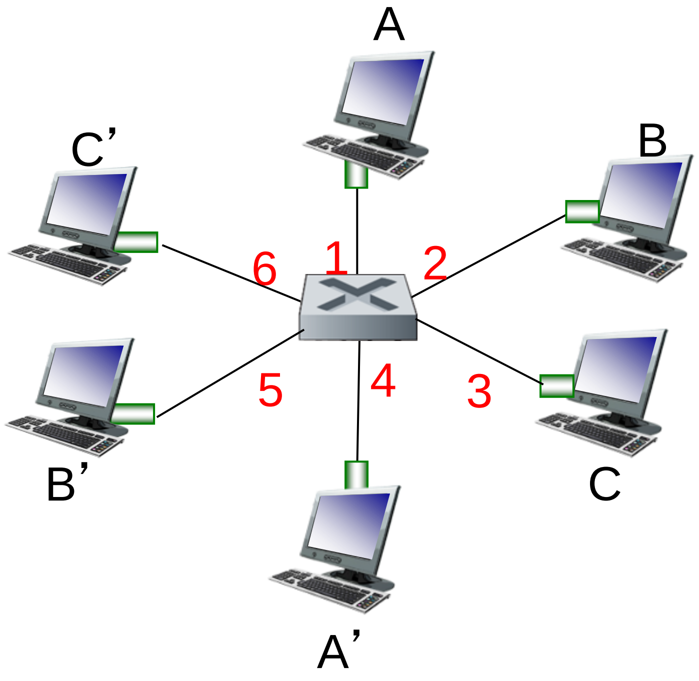
- error detected: frame is dropped

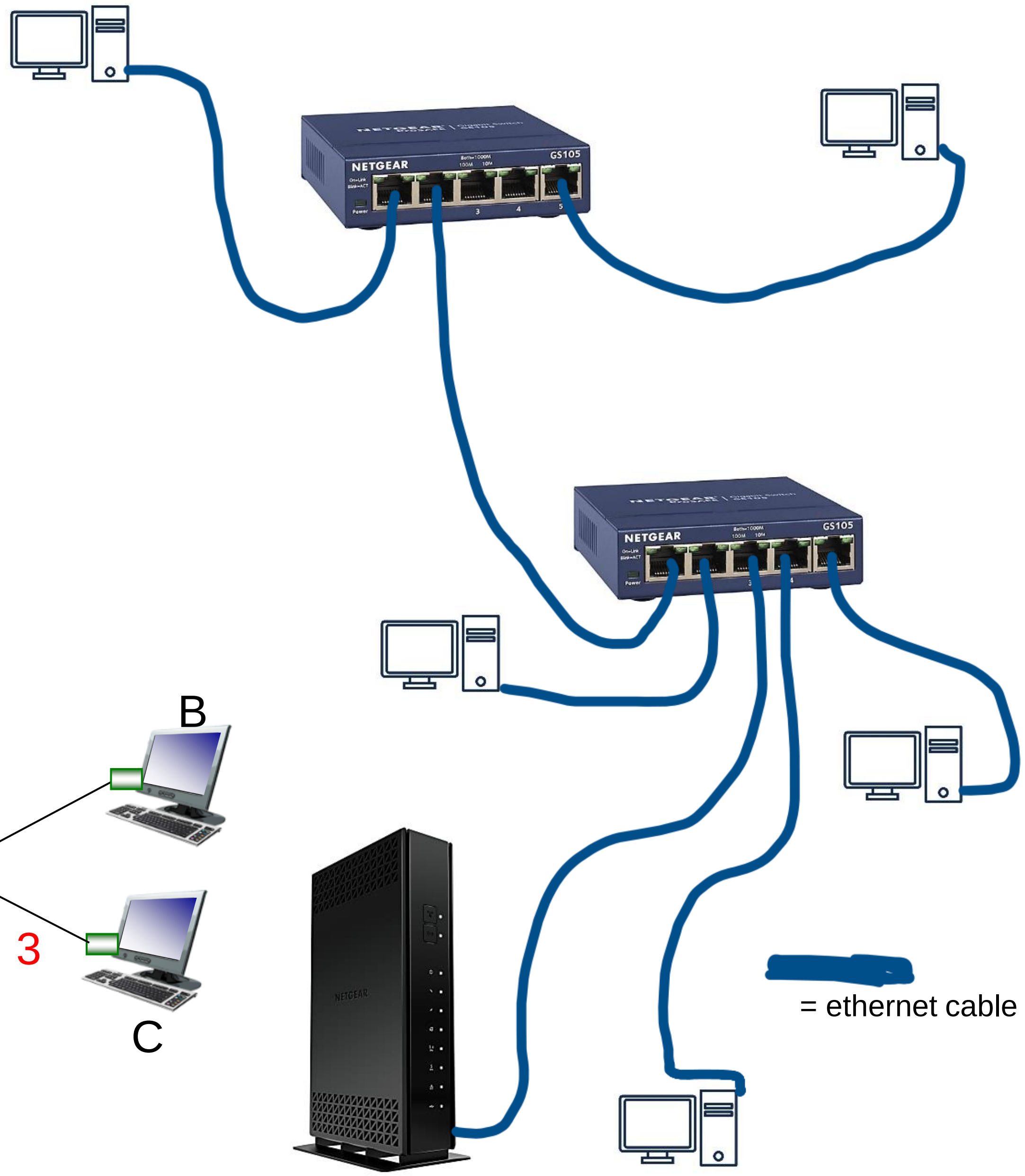


Ethernet switch

- Switches will store and forward ethernet frames
- Hosts have *dedicated*, direct connection to switch
- Ethernet protocol used on each incoming link, **but no collisions between links**
- Switching: A-A’ and B-B’ can transmit simultaneously, without collisions

- Transparent: Hosts are not aware they are connected to a switch
- Plug and play; self-learning





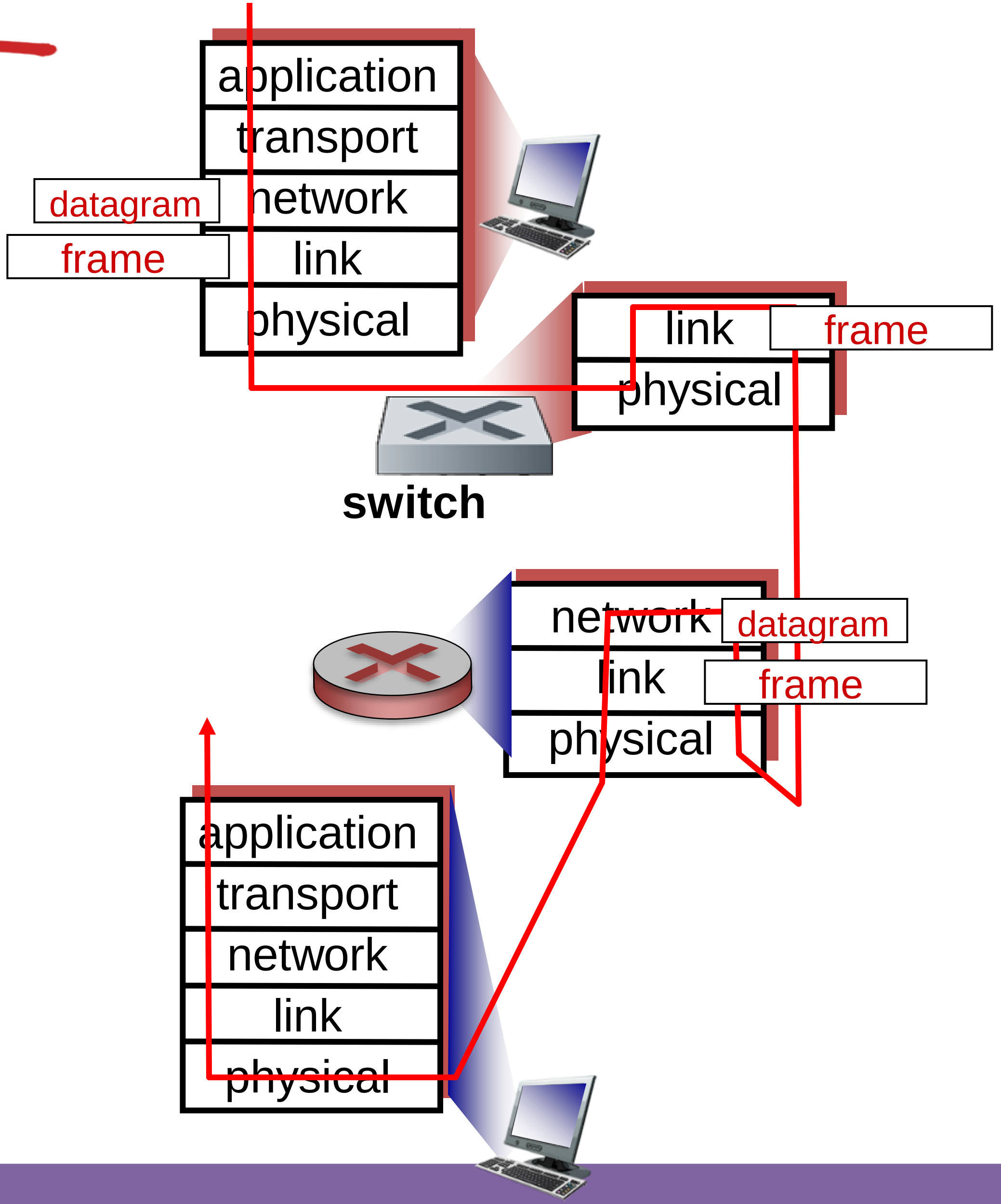
Switches vs. routers

both are store-and-forward:

- *routers:* network-layer devices (examine network-layer headers)
- *switches*: link-layer devices (examine link-layer headers)

both have forwarding tables:

- *routers:* compute tables using routing algorithms, IP addresses
- *switches:* learn forwarding table using flooding, learning, MAC addresses



https://www.youtube.com/watch?v=1z0ULvg_pW8