CS 43: Computer Networks

21:Link Layer, Media Access Protocols

Dec 5, 2024

Adapted from Slides by: J.Kurose, J.Rexford, D. Choffness, K. Webb

SWARTHMORE COLLEGE

The Link Layer

Application: the application (e.g., the Web, Email)
Transport: end-to-end connections, reliability
Network: routing
Link (data-link): framing, error detection
Physical: 1's and 0's/bits across a medium (copper, the air, fiber)

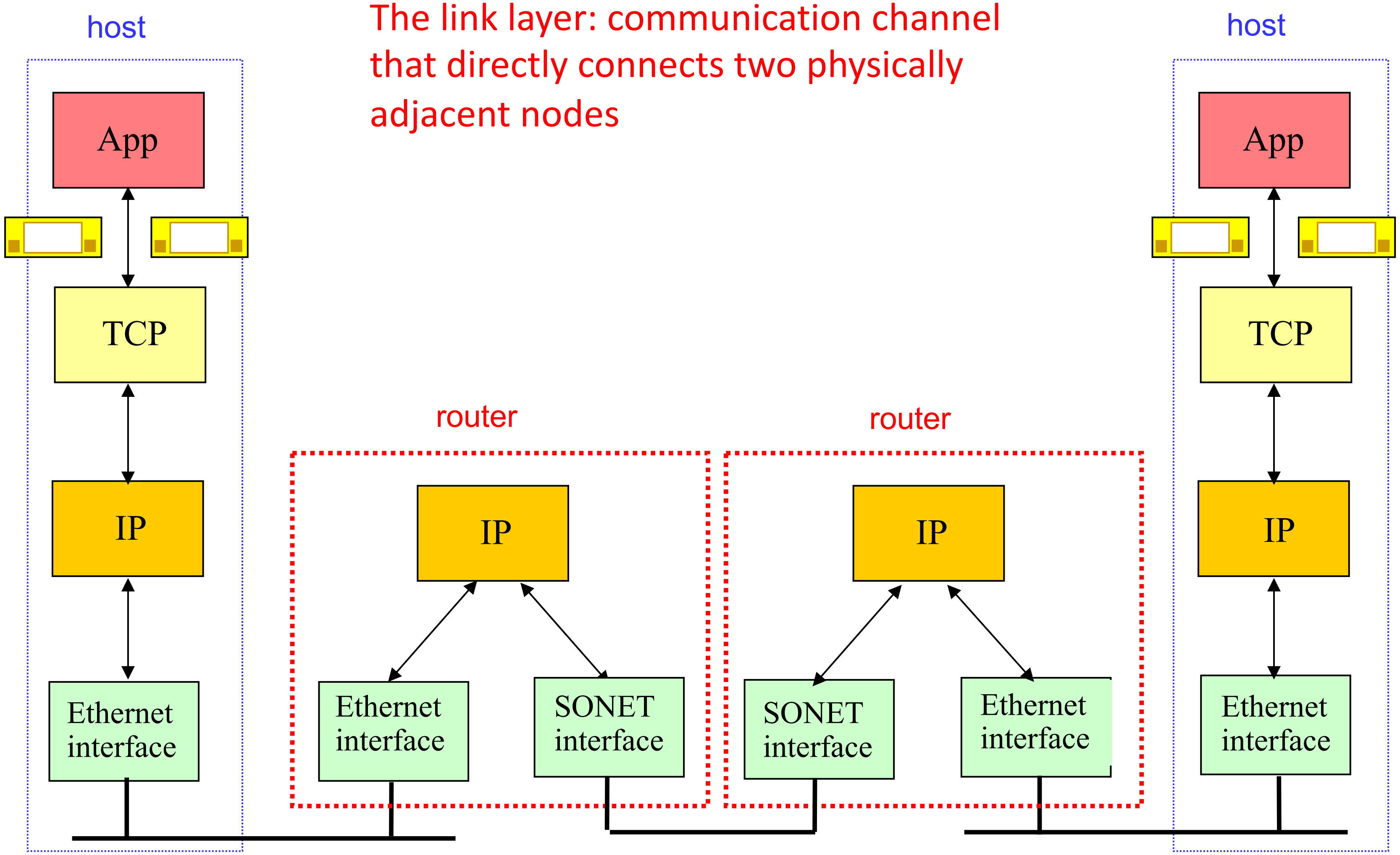
What is a Link?

Communication Medium

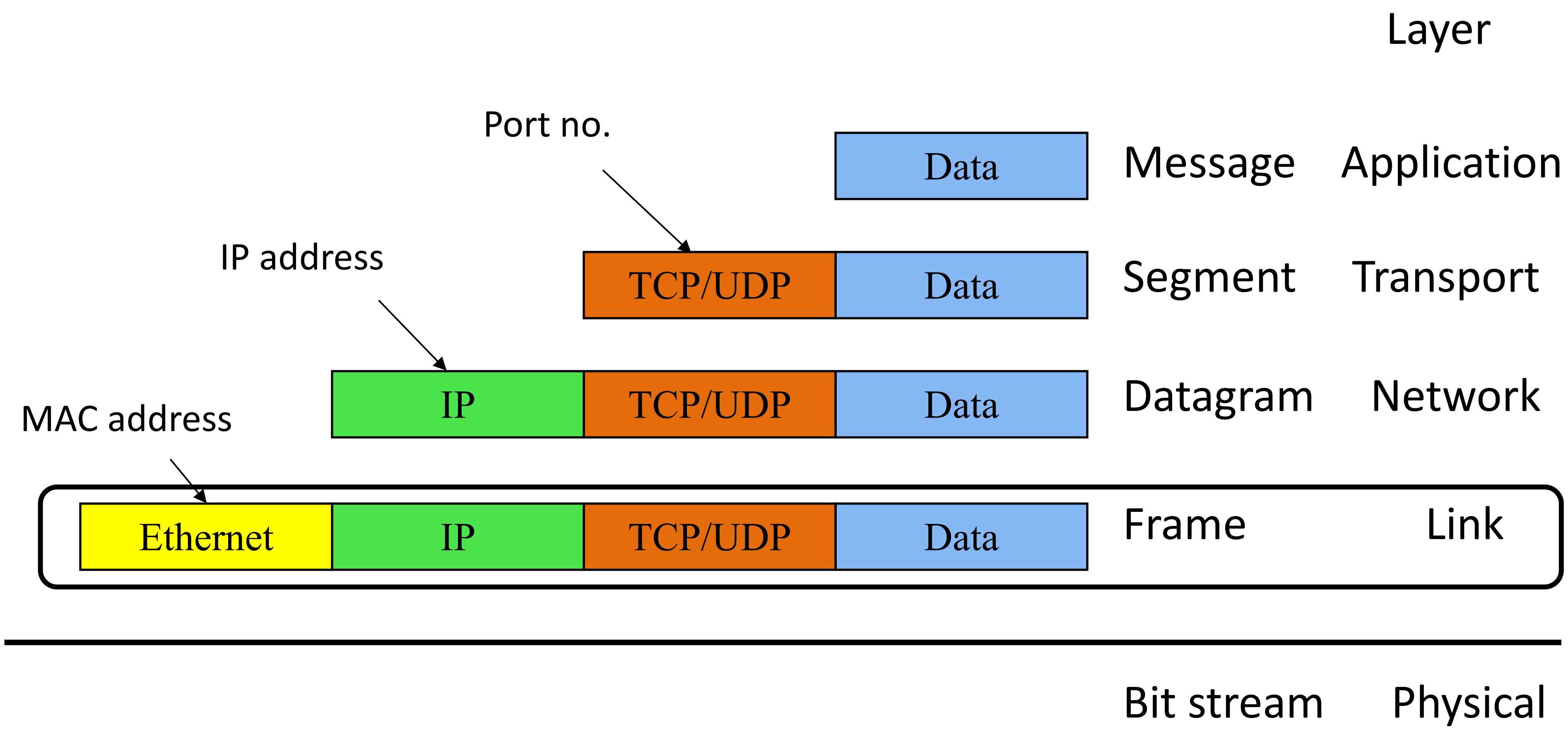
Network Adapter

The Link Layer

The link layer: communication channel that directly connects two physically adjacent nodes

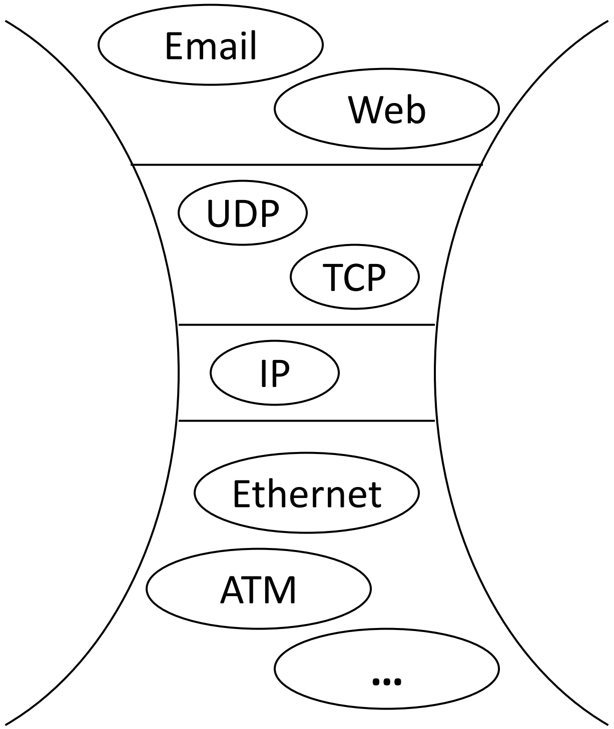


Layering and encapsulation



Internet Protocol Stack

- Application: Email, Web, …
- Transport: TCP, UDP, …
- Network: IP
- Link: Ethernet, WiFi, SONET, …
- Physical: copper, fiber, air, …



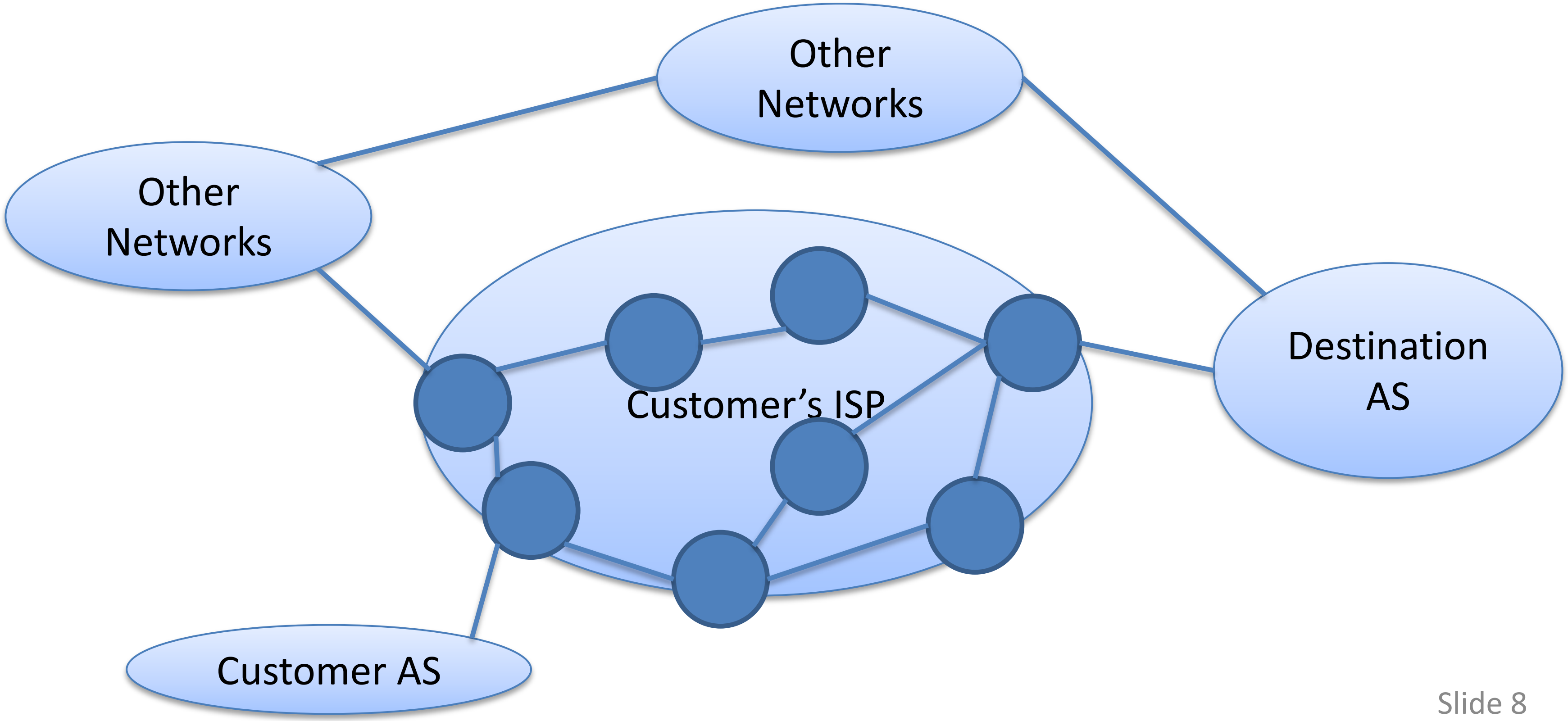
- “Hourglass” model, “thin waist”, “narrow waist”

Recall IP Motivation

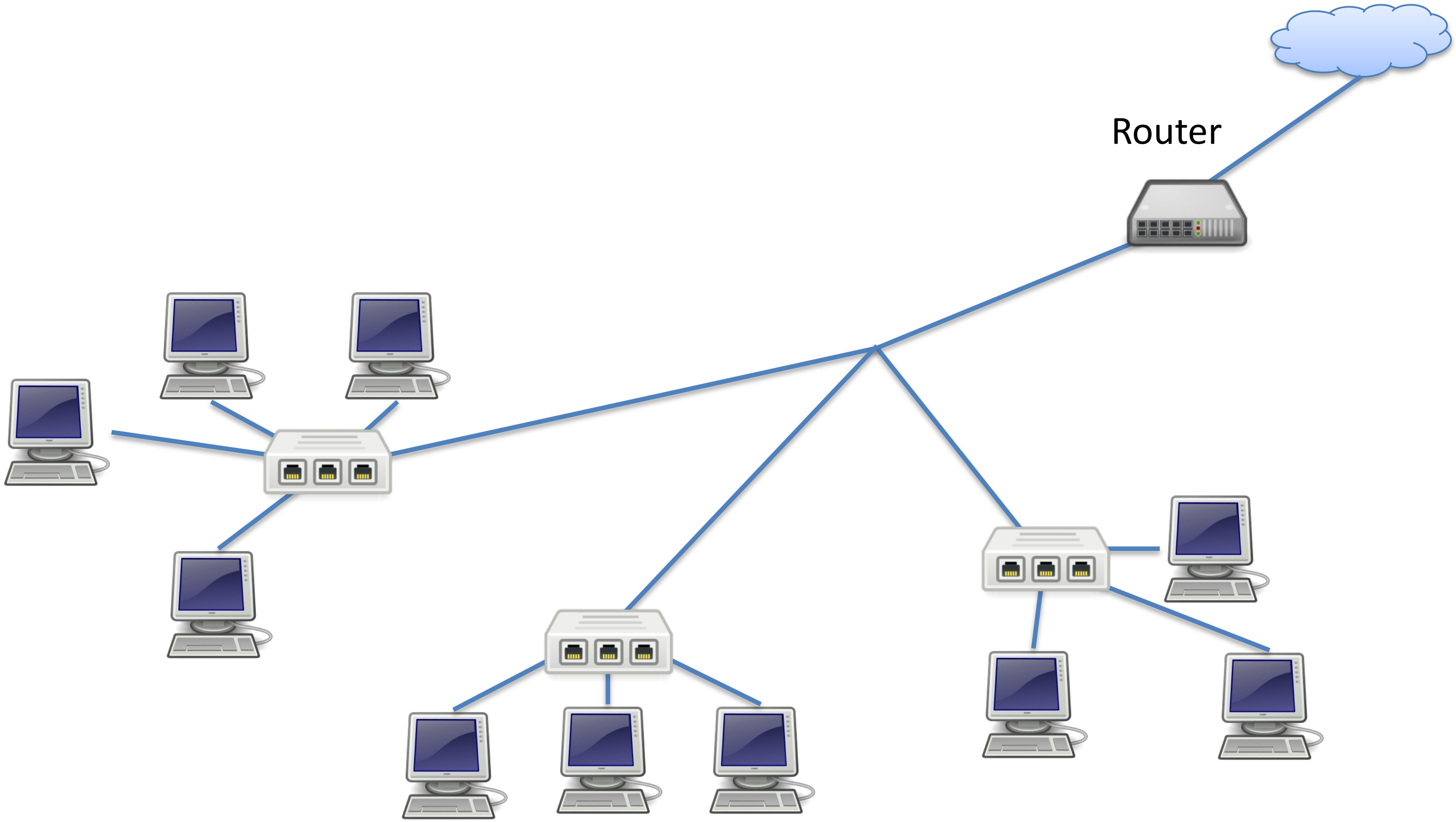
- 1970's: new network technologies emerge
  - SATNet, Packet Radio, Ethernet
  - All "islands" to themselves – didn't work together
- IP question: how to connect these networks?
- This implies: These networks do all the stuff networks need to do, without IP or routers.
  - Solves some of the same problems as IP
  - Often in a different way (smaller scale)

From Macro- to Micro-

- Previously, we looked at Internet scale…



Within a Subnet



Link Layer Goal

- Get from one node to it’s adjacent neighbor.

- Abstract the details of the underlying network technology from the protocols above it (IP).

- Lots of media with different characteristics:
  - Copper cable
  - Fiber optic cable
  - Radio/electromagnetic broadcast
  - Satellite

Challenges

- Even with one medium:
  - Potentially many ways to format & signal data.
  - Multiple users may contend to transmit.
  - How do we address endpoints?
  - How do we locate destinations?

Today Link Layer Functions

1. Addressing: identifying endpoints

2. Framing: Dividing data into pieces that are sized for the network to handle.

3. Link access: Determining how to share the medium, who gets to send, and for how long.

Link Layer Functions

1. Addressing: identifying endpoints

- Must be able to uniquely identify each host on the network. Can't assume IP.

- Implication: each host on the Internet will have **two** addresses: IP & link-layer

Typically referred to as "MAC address"
Media Access Control

MAC Addresses

- MAC (or LAN or physical or Ethernet) address:
  - 48 bit MAC address
  - e.g.: 1A-2F-BB-76-09-AD

hexadecimal (base 16) notation
(each digit represents 4 bits)

MAC vs. IP Addresses

- 32-bit IP address
- IP hierarchical address not portable!
  - address depends on IP subnet to which node is attached
- used by network layer for end-to-end routing

- 48 bit MAC address burned in NIC ROM.
- MAC flat address: portability
  - can move LAN card from one LAN to another
- used locally to get from one interface to another physically-connected interface

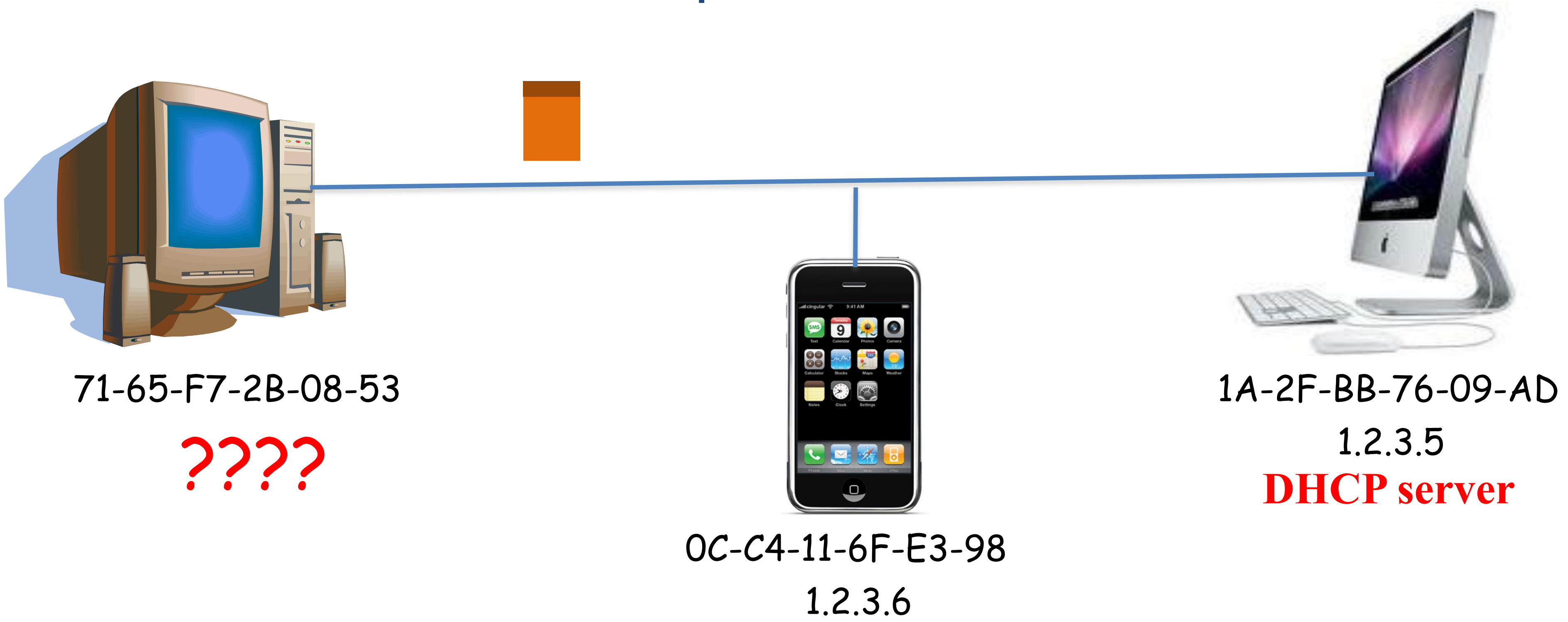
Analogy:
MAC address: like Social Security Number
IP address: like postal address

Addressing

- Typically, humans deal in IP addresses
  (or DNS names that resolve to them)

- Network needs a mechanism to determine corresponding MAC address for local sending

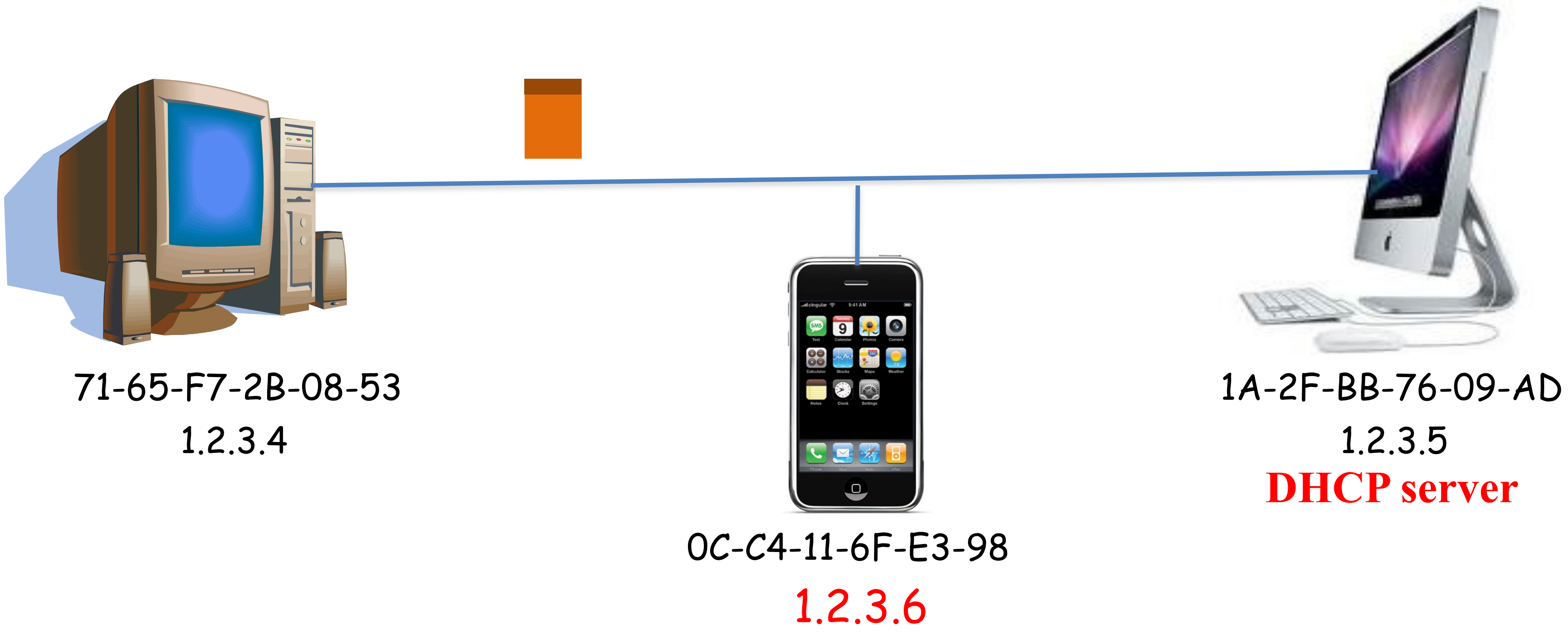
Who Am I: Acquiring an IP Address



- Dynamic Host Configuration Protocol (DHCP)
  - Broadcast “I need an IP address, please!”
  - Response “You can have IP address 1.2.3.4.”

Adapted from J.Rexford

Who Are You: Discovering the Receiver



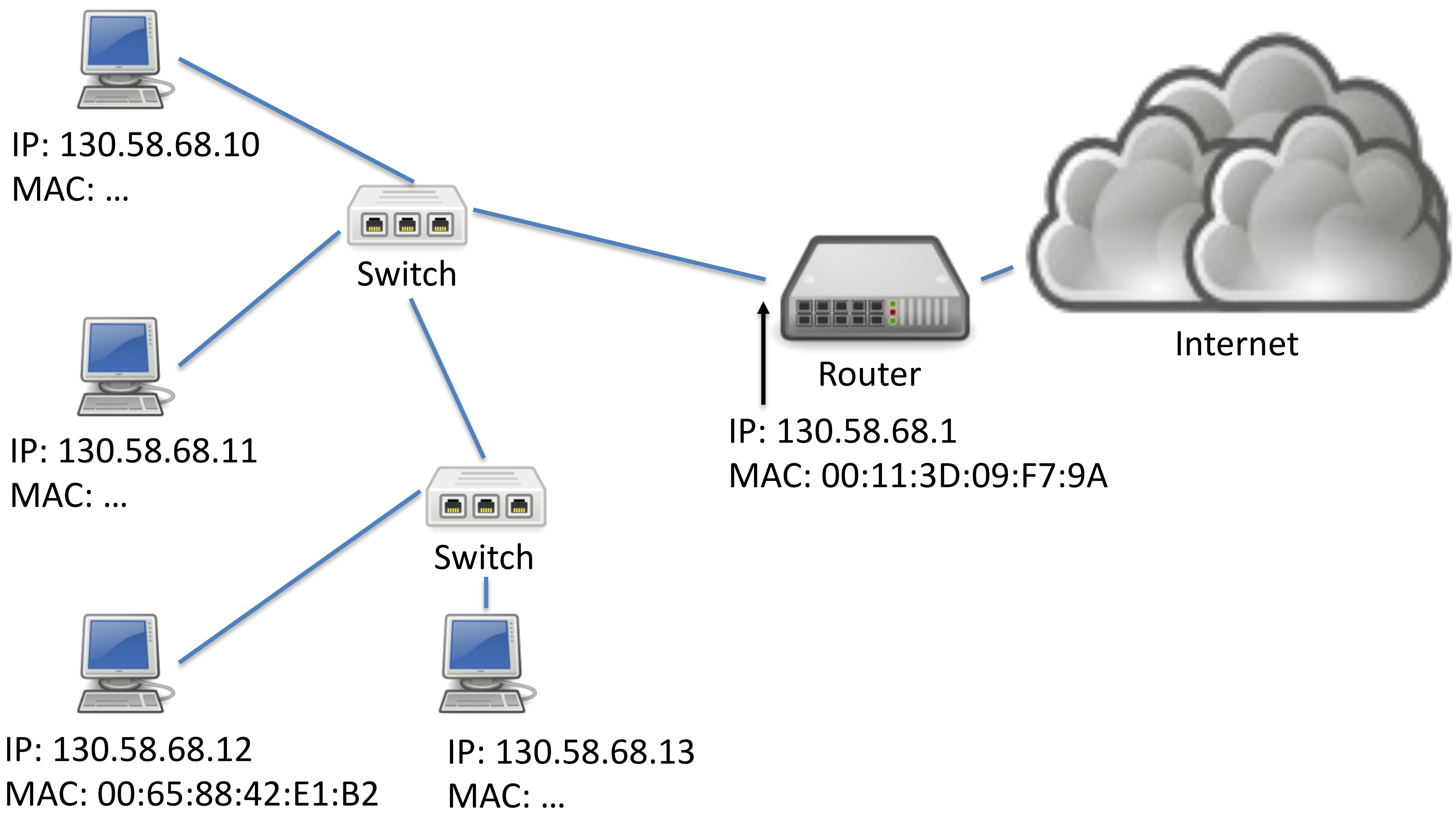
- Address Resolution Protocol (ARP)
  - Broadcast “who has IP address 1.2.3.6?”
  - Response “0C-C4-11-6F-E3-98 has 1.2.3.6!”

Adapted from J.Rexford

ARP: Address Resolution Protocol

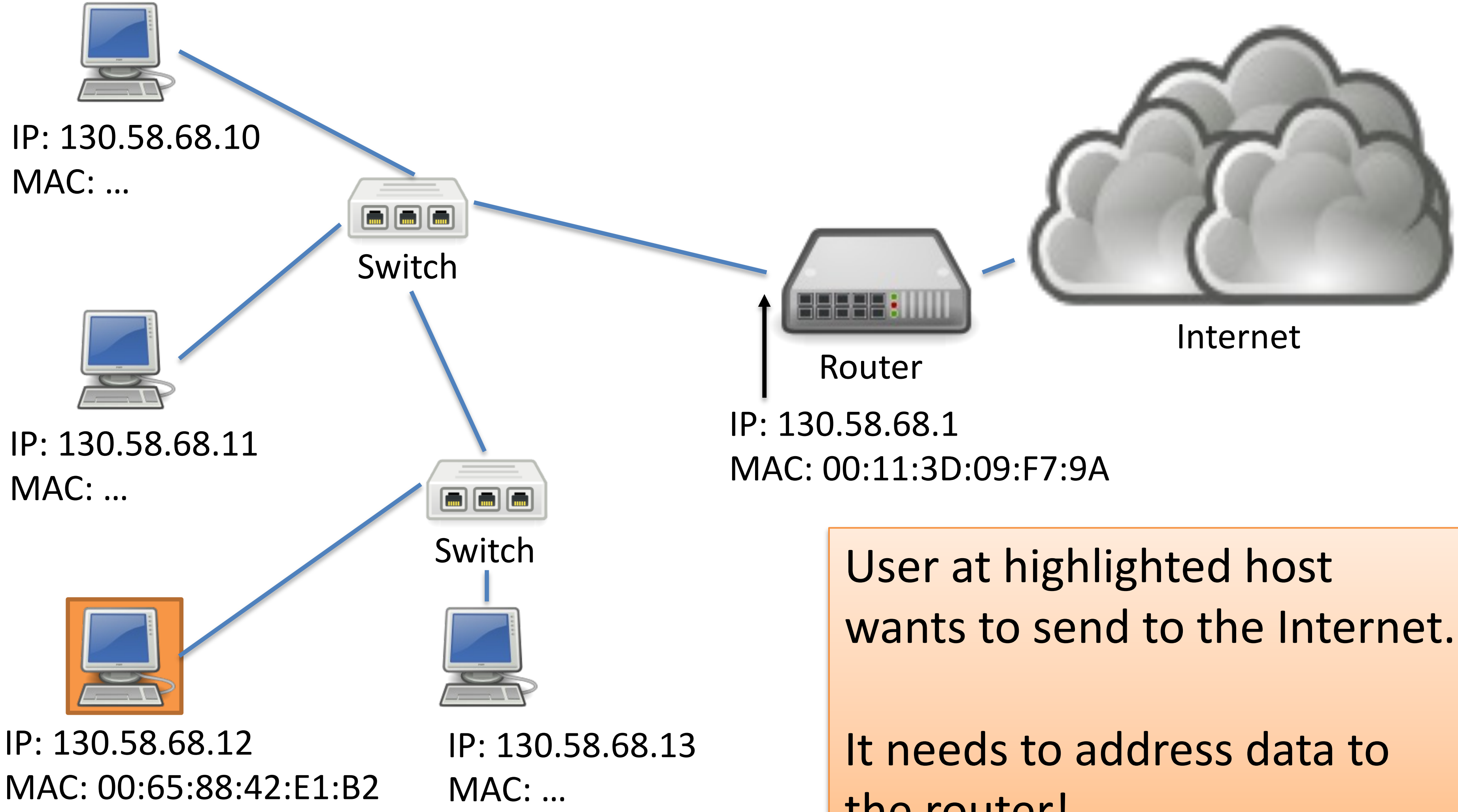
- Common in networks you use: Ethernet, WiFi
- Broadcast to entire local network:
  - "I'm looking for the MAC address of the host with IP address A.B.C.D. If you're out there, please respond to me!"

ARP Example



IP: 130.58.68.10
MAC: …
Switch
IP: 130.58.68.11
MAC: …
Switch
IP: 130.58.68.12
MAC: 00:65:88:42:E1:B2
IP: 130.58.68.13
MAC: …
Router
IP: 130.58.68.1
MAC: 00:11:3D:09:F7:9A
Internet

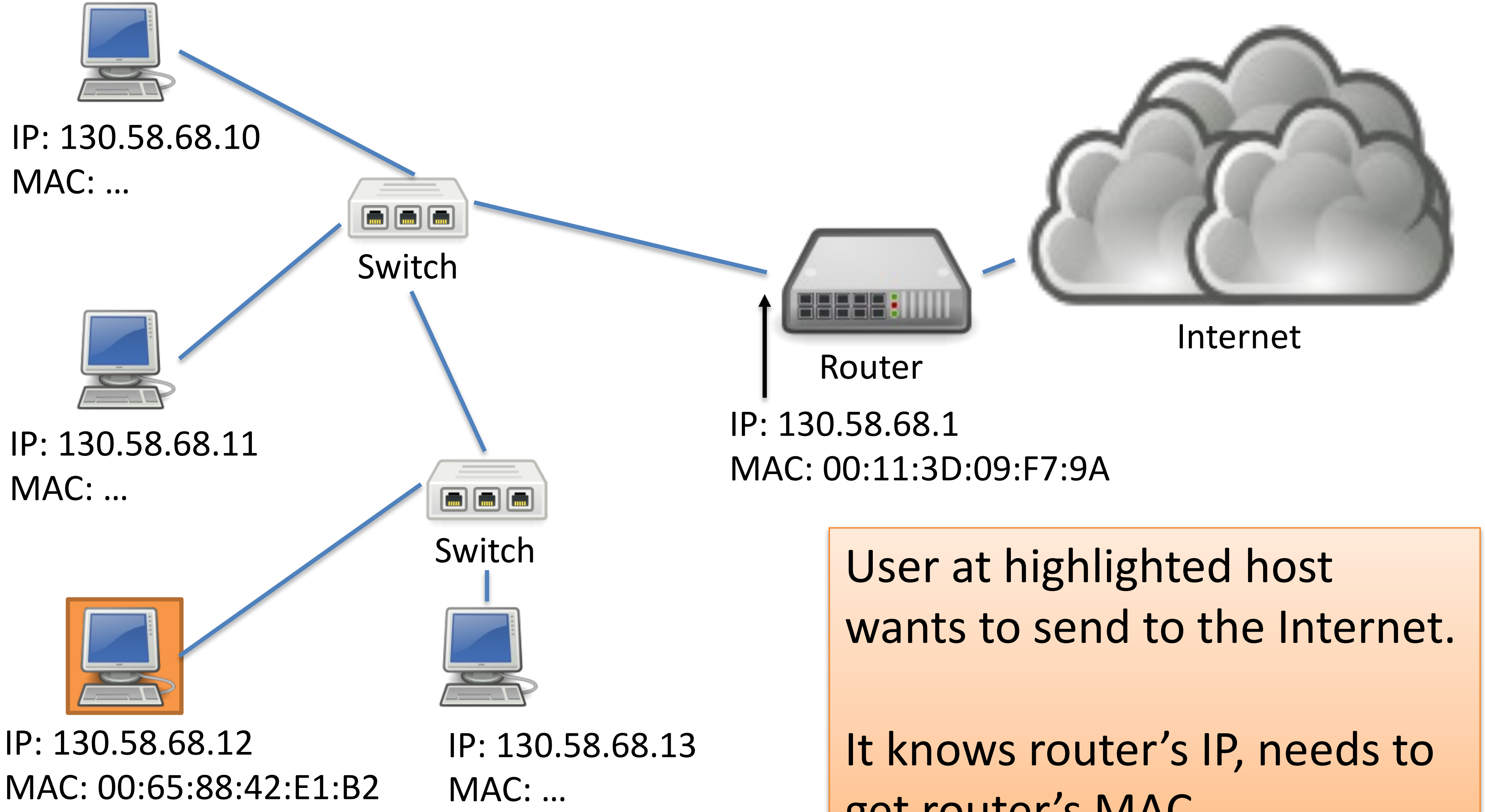
ARP Example



User at highlighted host wants to send to the Internet.

It needs to address data to the router!

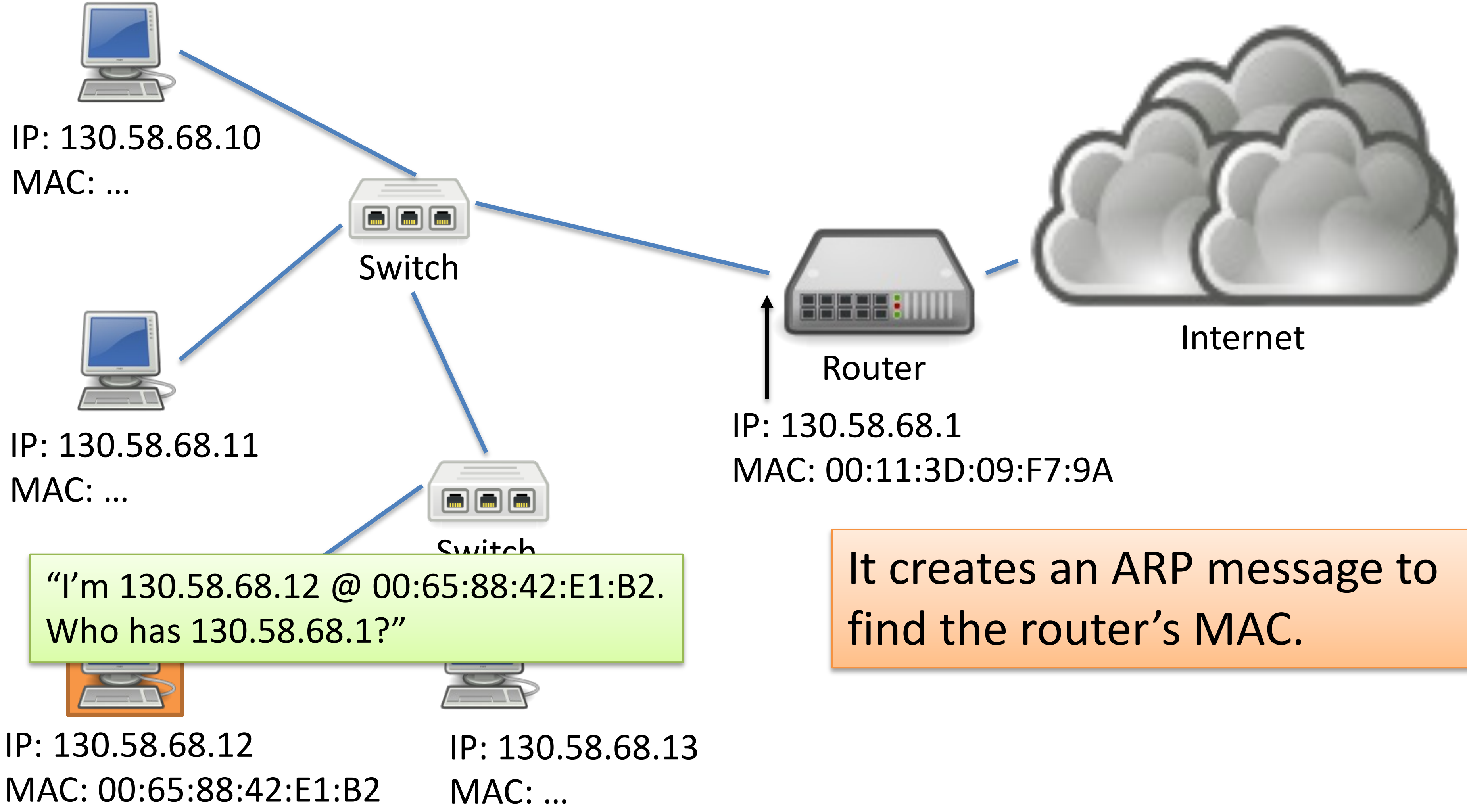
ARP Example



User at highlighted host wants to send to the Internet.

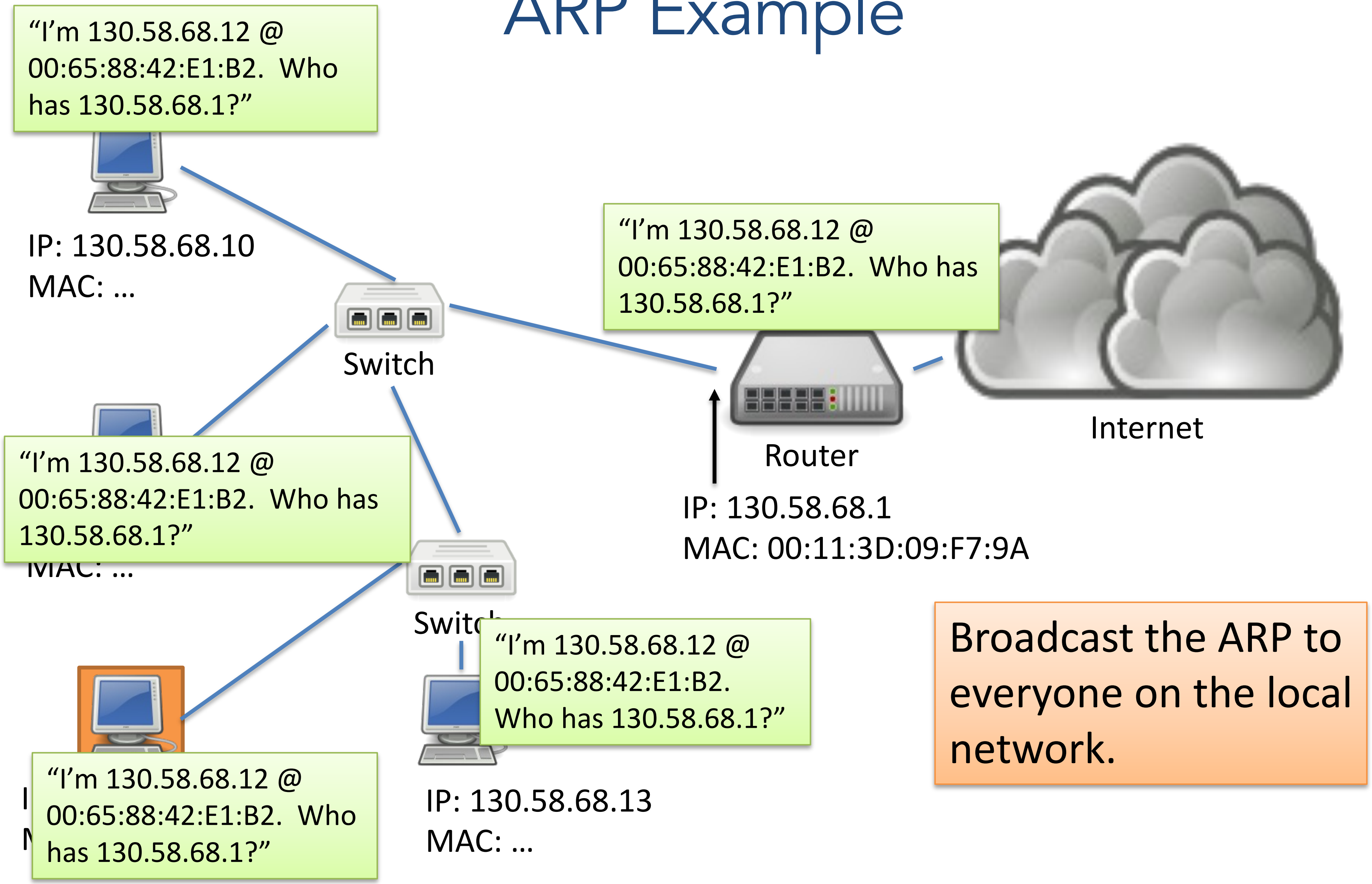
It knows router's IP, needs to get router's MAC.

ARP Example

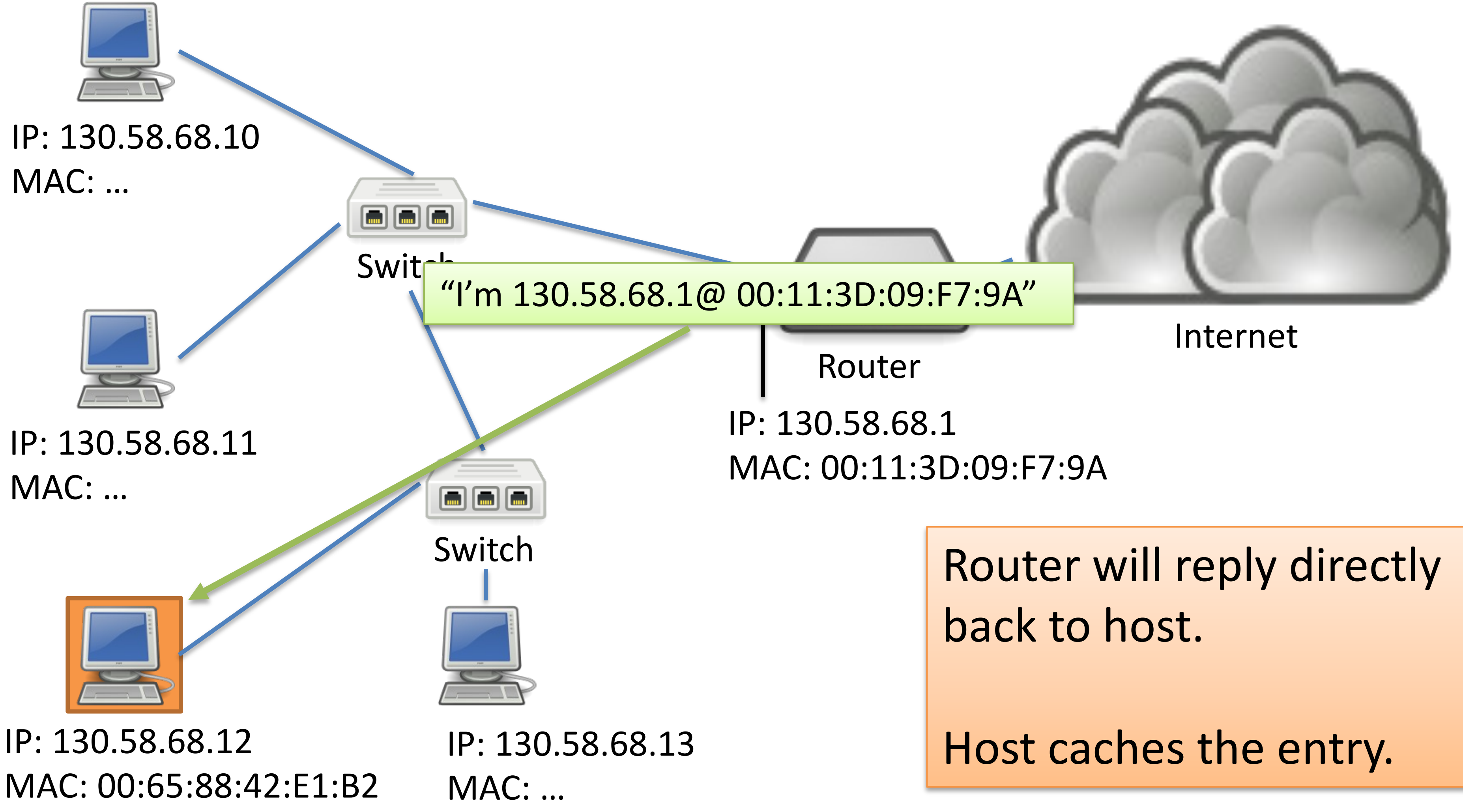


It creates an ARP message to find the router’s MAC.

ARP Example


"I'm 130.58.68.12 @ 00:65:88:42:E1:B2. Who has 130.58.68.1?"
IP: 130.58.68.10
MAC: …
Switch
"I'm 130.58.68.12 @ 00:65:88:42:E1:B2. Who has 130.58.68.1?"
Router
IP: 130.58.68.1
MAC: 00:11:3D:09:F7:9A
Internet
"I'm 130.58.68.12 @ 00:65:88:42:E1:B2. Who has 130.58.68.1?"
"I'm 130.58.68.12 @ 00:65:88:42:E1:B2. Who has 130.58.68.1?"
IP: 130.58.68.13
MAC: …
"I'm 130.58.68.12 @ 00:65:88:42:E1:B2. Who has 130.58.68.1?"
Broadcast the ARP to everyone on the local network.

ARP Example

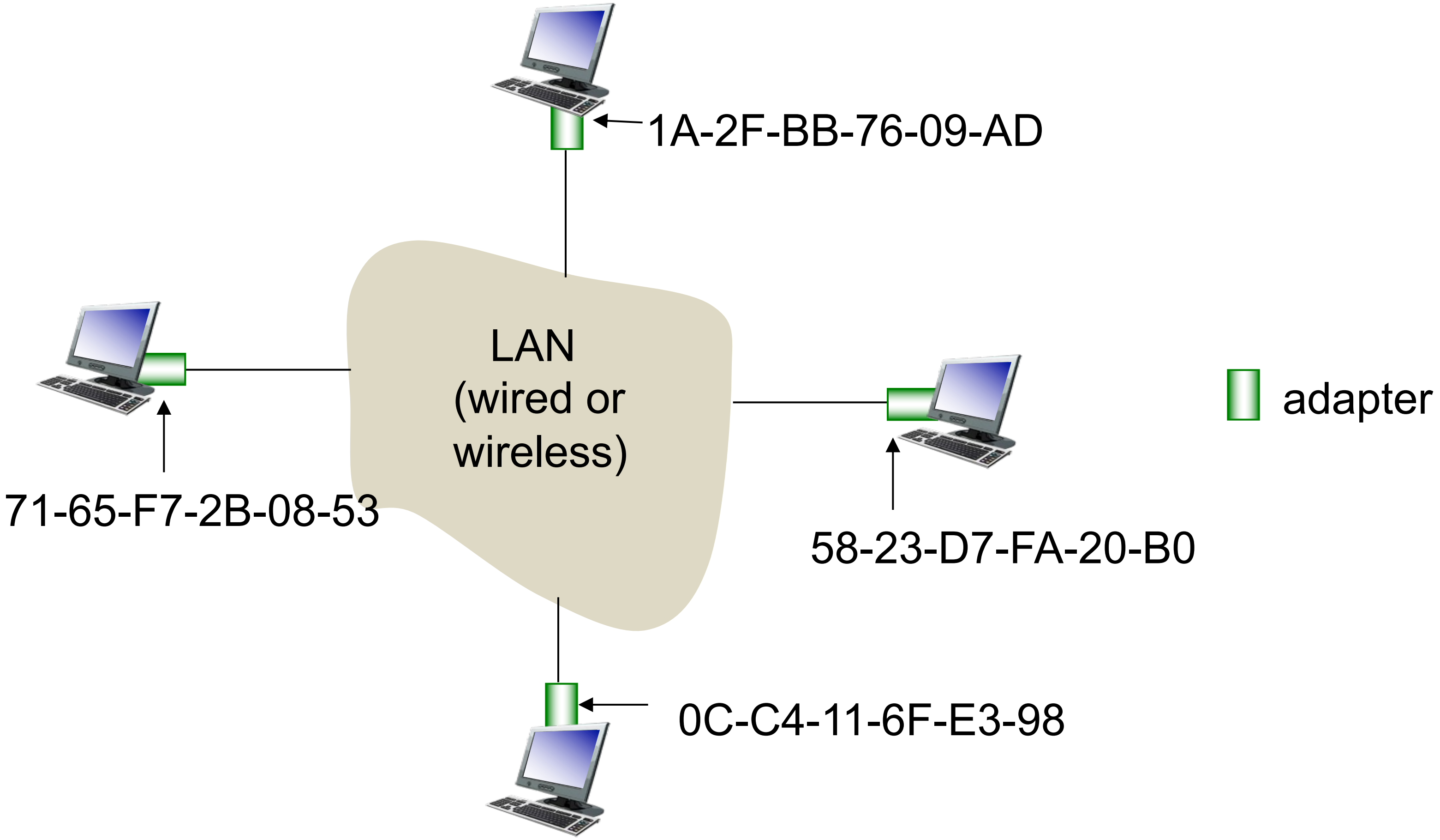


Router will reply directly back to host.

Host caches the entry.

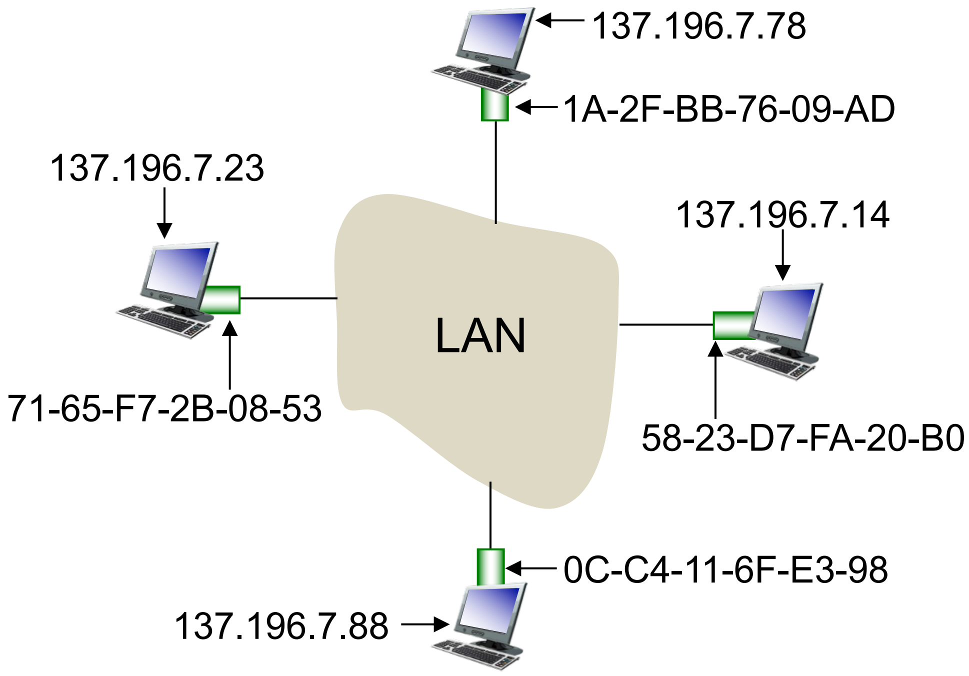
MAC Addresses

Each interface/adapter on LAN has unique MAC address



ARP: Address Resolution Protocol

Question: how to determine interface's MAC address, knowing its IP address?



ARP table: each IP node (host, router) on LAN has table

- IP/MAC address mappings for some LAN nodes:

  < IP address; MAC address; TTL>

- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

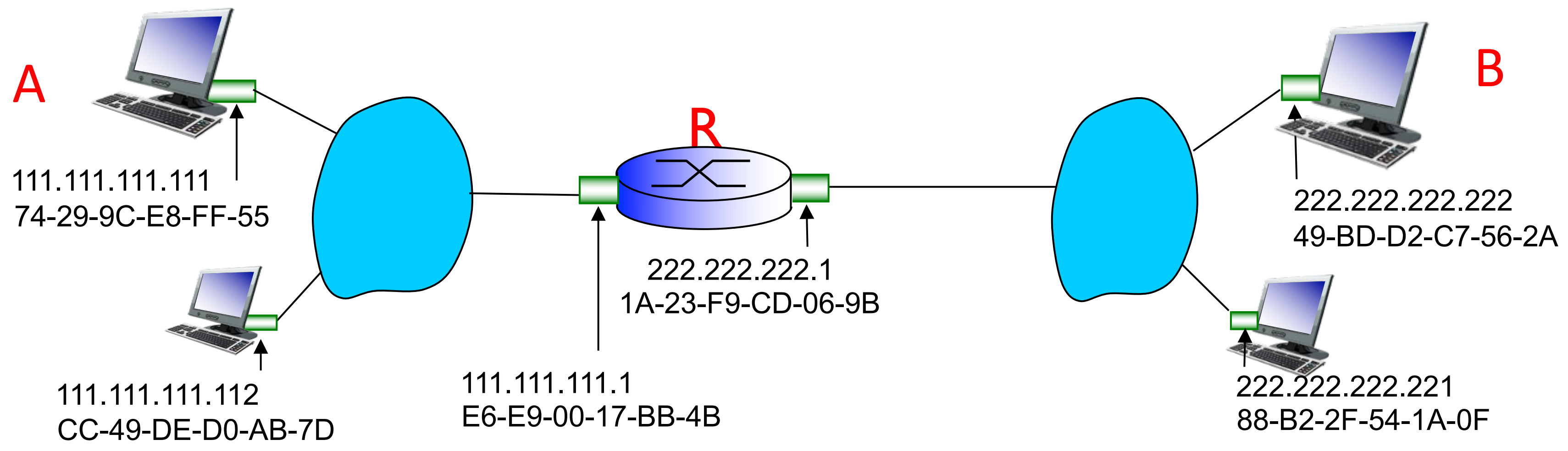
ARP protocol & LAN communication

- A wants to send datagram to B. A knows B’s IP address.
  - B’s MAC address not in A’s ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest Ethernet address = FF-FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query, most ignore it
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A’s MAC address (unicast)
- A caches IP-to-MAC address pair in its ARP table until timeout
  - soft state: times out unless refreshed, can be reacquired

Addressing: routing to another LAN

Walkthrough: send datagram from A to B via R

- focus on addressing – at IP and MAC layer
- assume A knows B's IP address (e.g., DNS lookup)
- how many subnets are present in this figure?

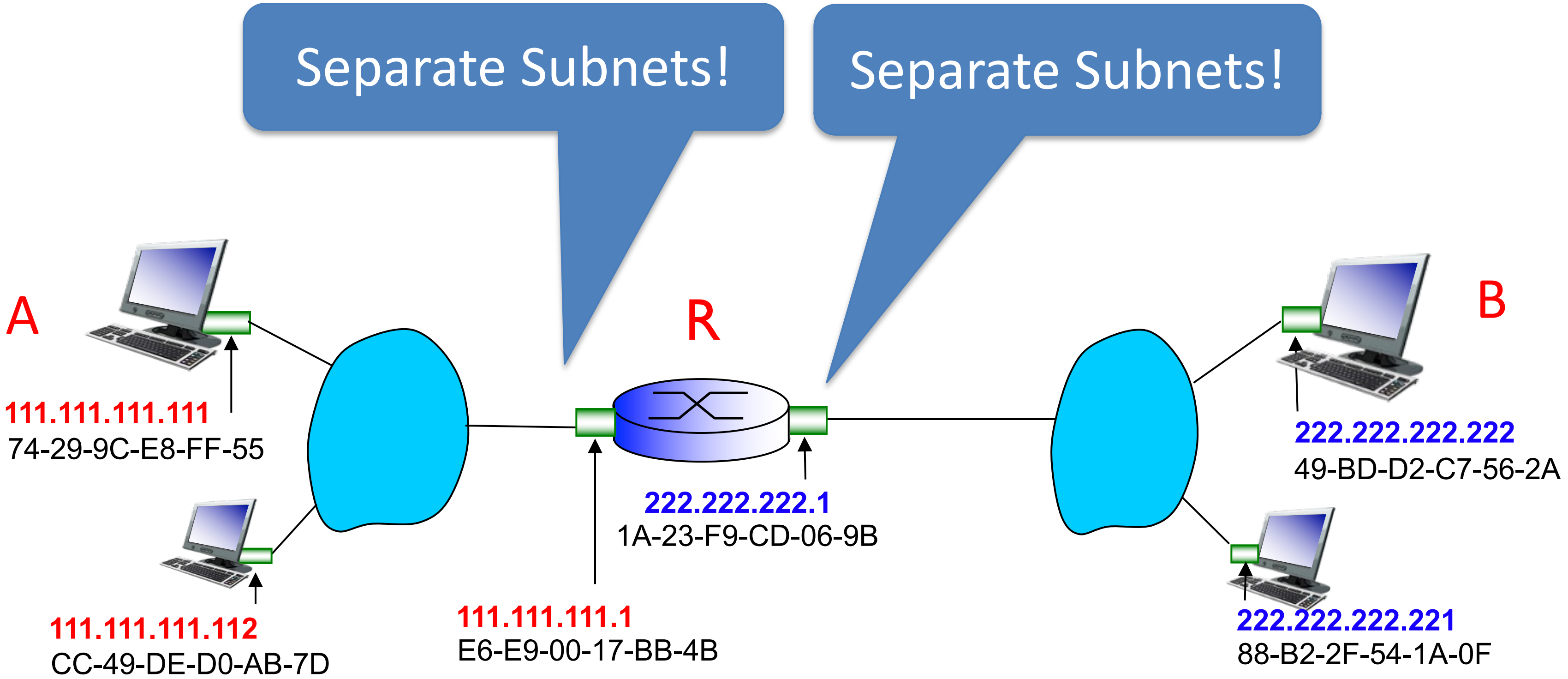


Addressing: routing to another LAN

Walkthrough: send datagram from A to B via R

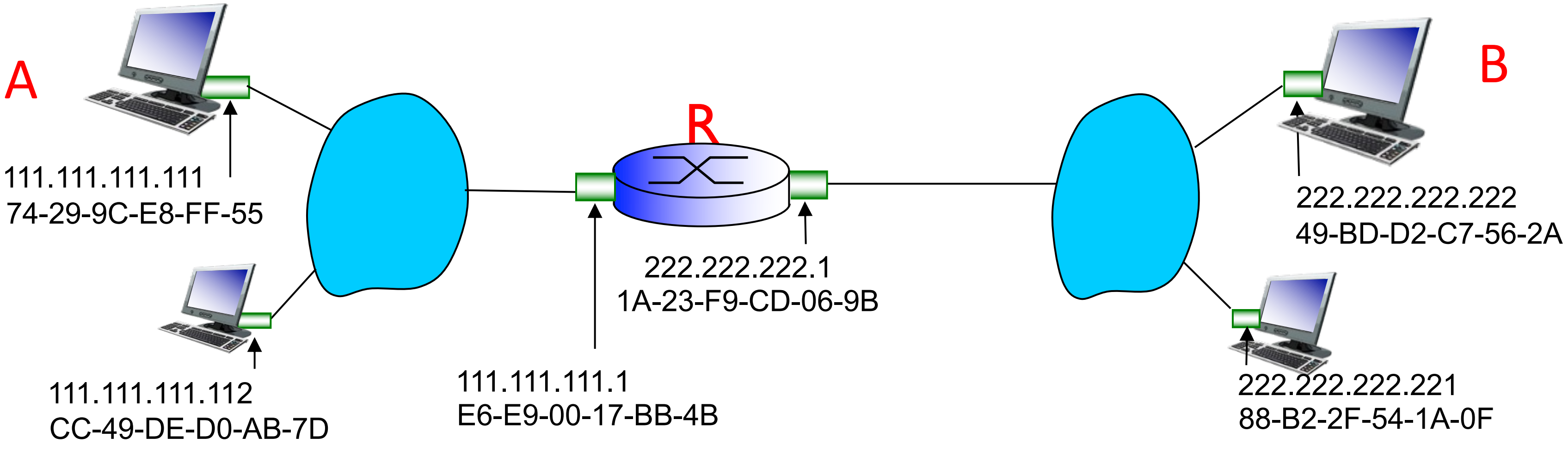
– focus on addressing – at IP and MAC layer

– assume A knows B's IP address (e.g., DNS lookup)



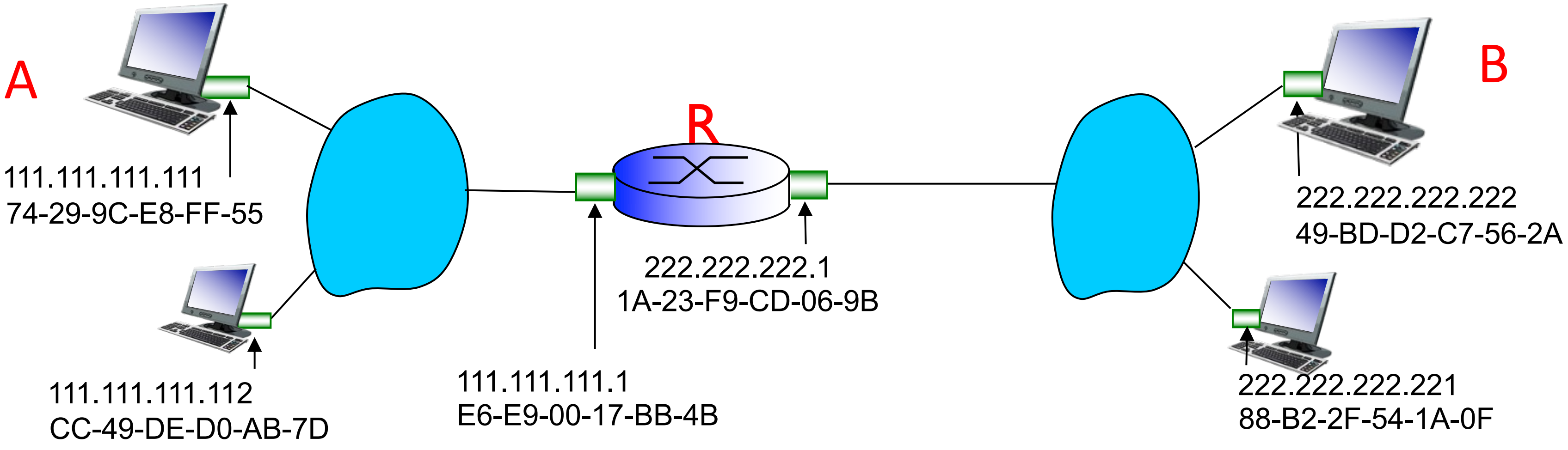
Walkthrough: send datagram from A to B via R

1. Who do we address as the IP packet destination ?
2. Who do we forward it to on the first hop?



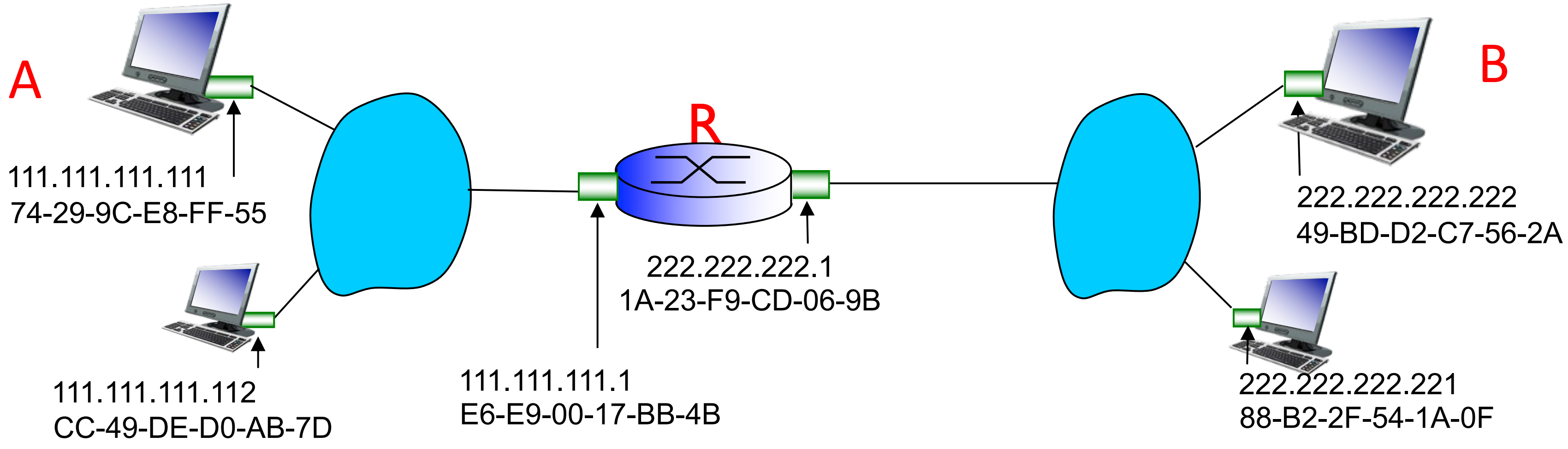
Walkthrough: send datagram from A to B via R

1. Who do we address as the IP packet destination ?
    - IP Address to B (End-to-end address to express where we want to get to)



Walkthrough: send datagram from A to B via R

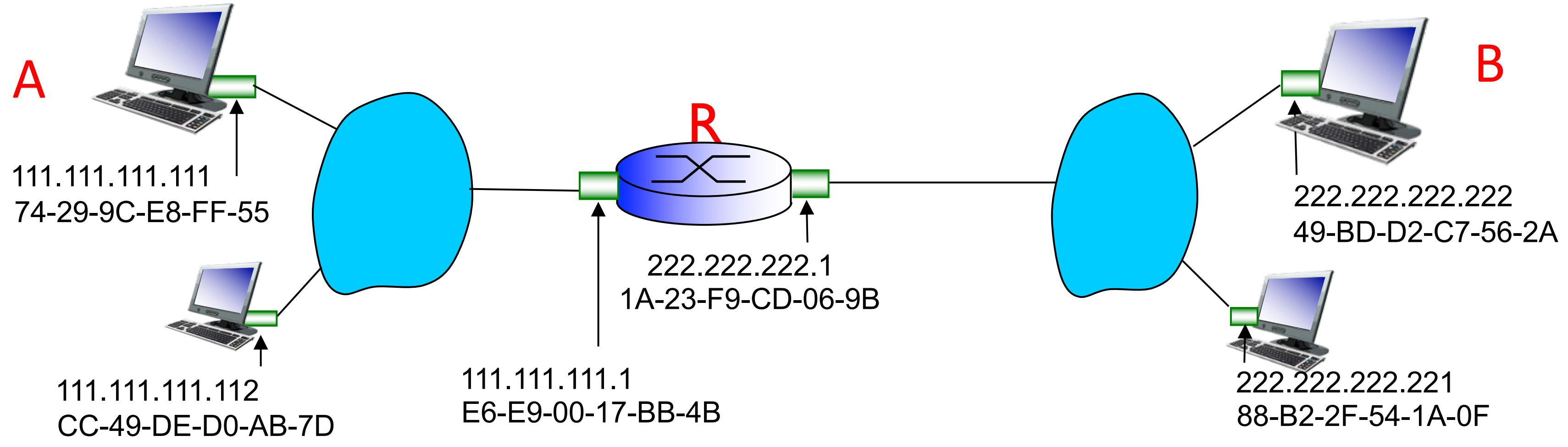
2. Who do we forward it to on the first hop?
   - MAC Address to R (Intermediate address to send to router)



How does A learn the IP address of the router?

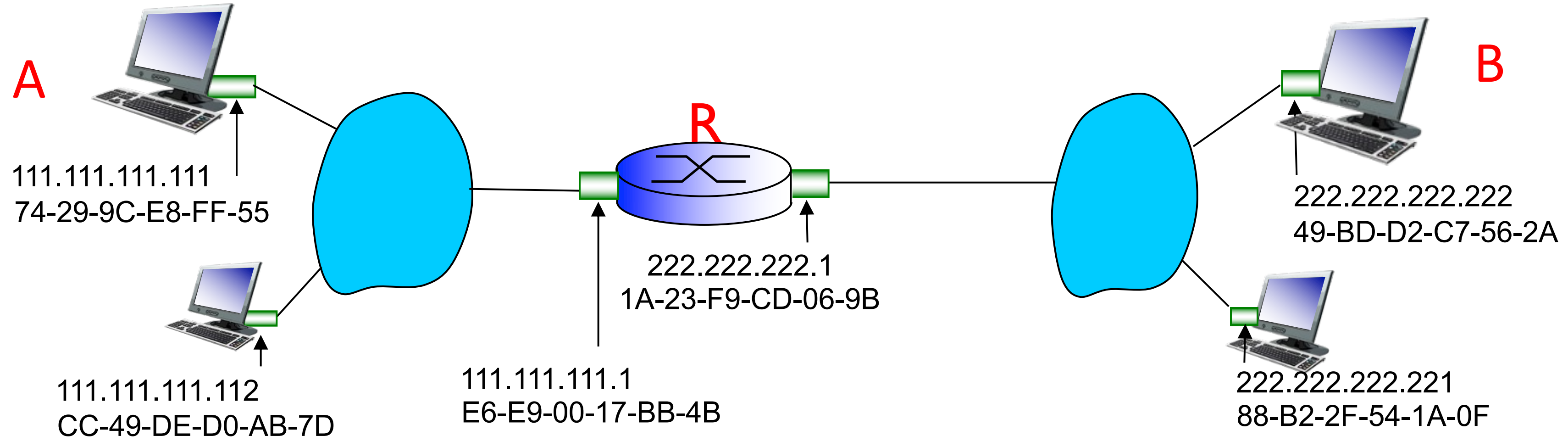
A. ARP: Address Resolution Protocol
B. DHCP: Dynamic Host Configuration Protocol
C. IP: Internet Protocol
D. Routing Protocol

why do we even need the IP address of Router ?



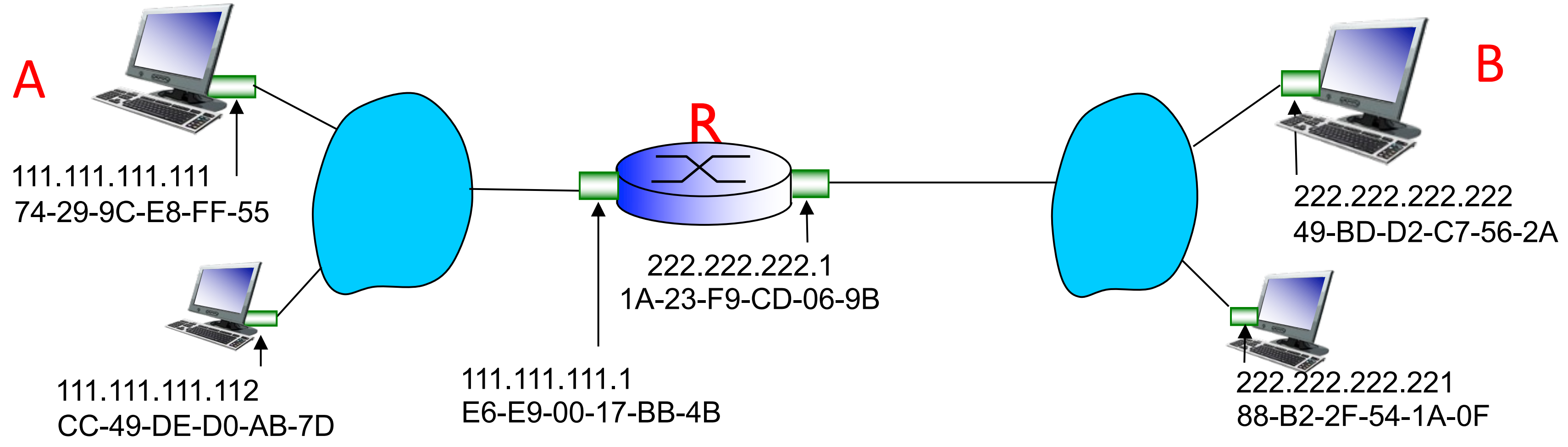
How does A learn the IP address of the Router ?

A. ARP: Address Resolution Protocol
B. DHCP: Dynamic Host Configuration Protocol (it gives you your IP address, and the IP address of the router to get to the Internet and it is up to you figure out the MAC address)
C. IP: Internet Protocol
D. Routing Protocol



How does A learn the MAC address of the router?

A. ARP: Address Resolution Protocol
B. DHCP: Dynamic Host Configuration Protocol
C. IP: Internet Protocol
D. Routing Protocol



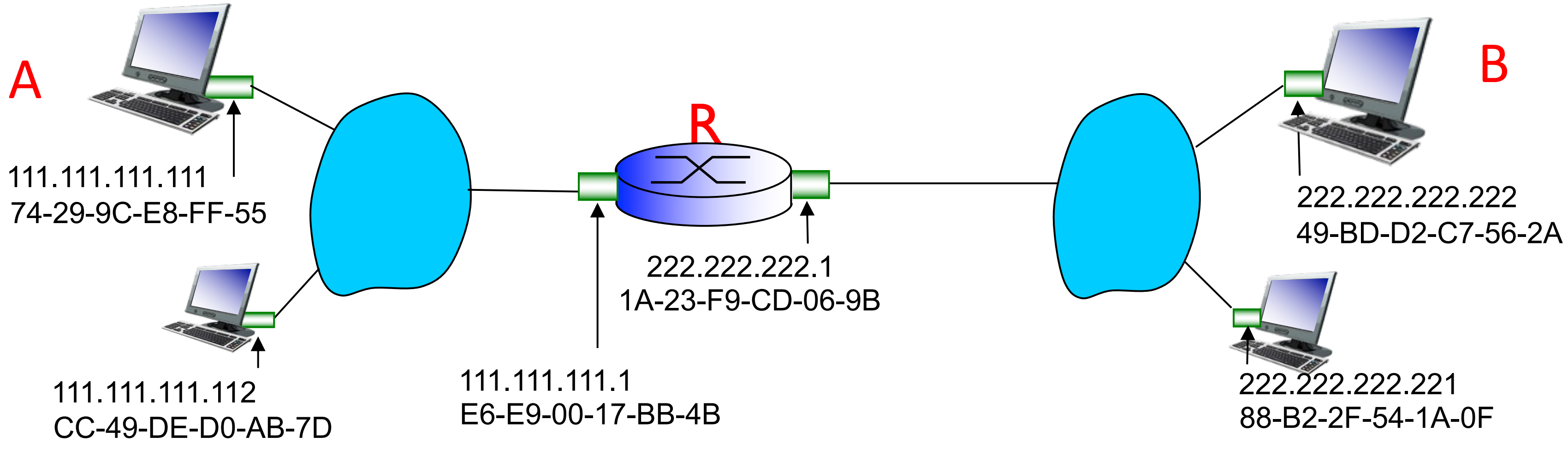
How does A learn the MAC address of the router?

A. ARP: Address Resolution Protocol

B. DHCP: Dynamic Host Configuration Protocol

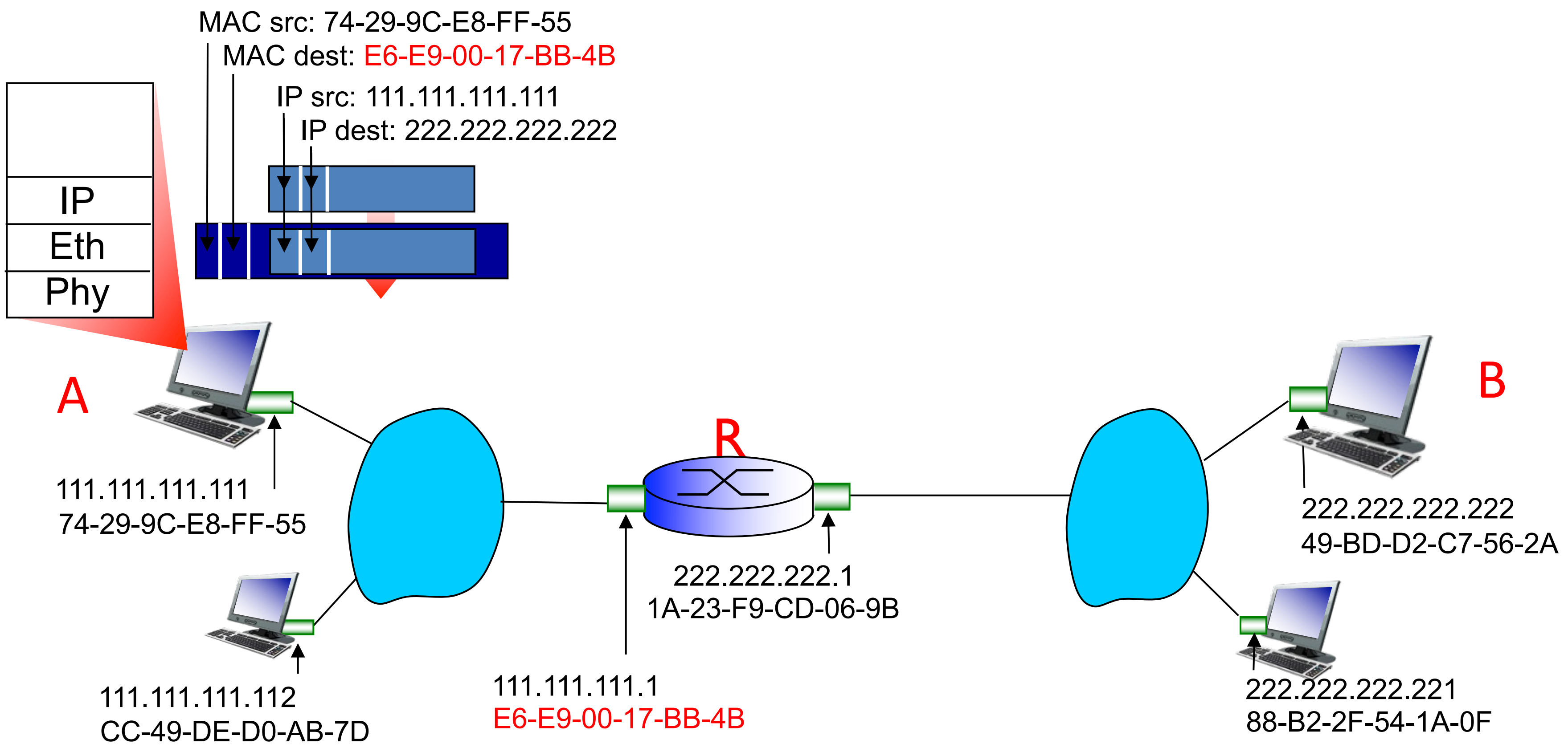
C. IP: Internet Protocol

D. Routing Protocol



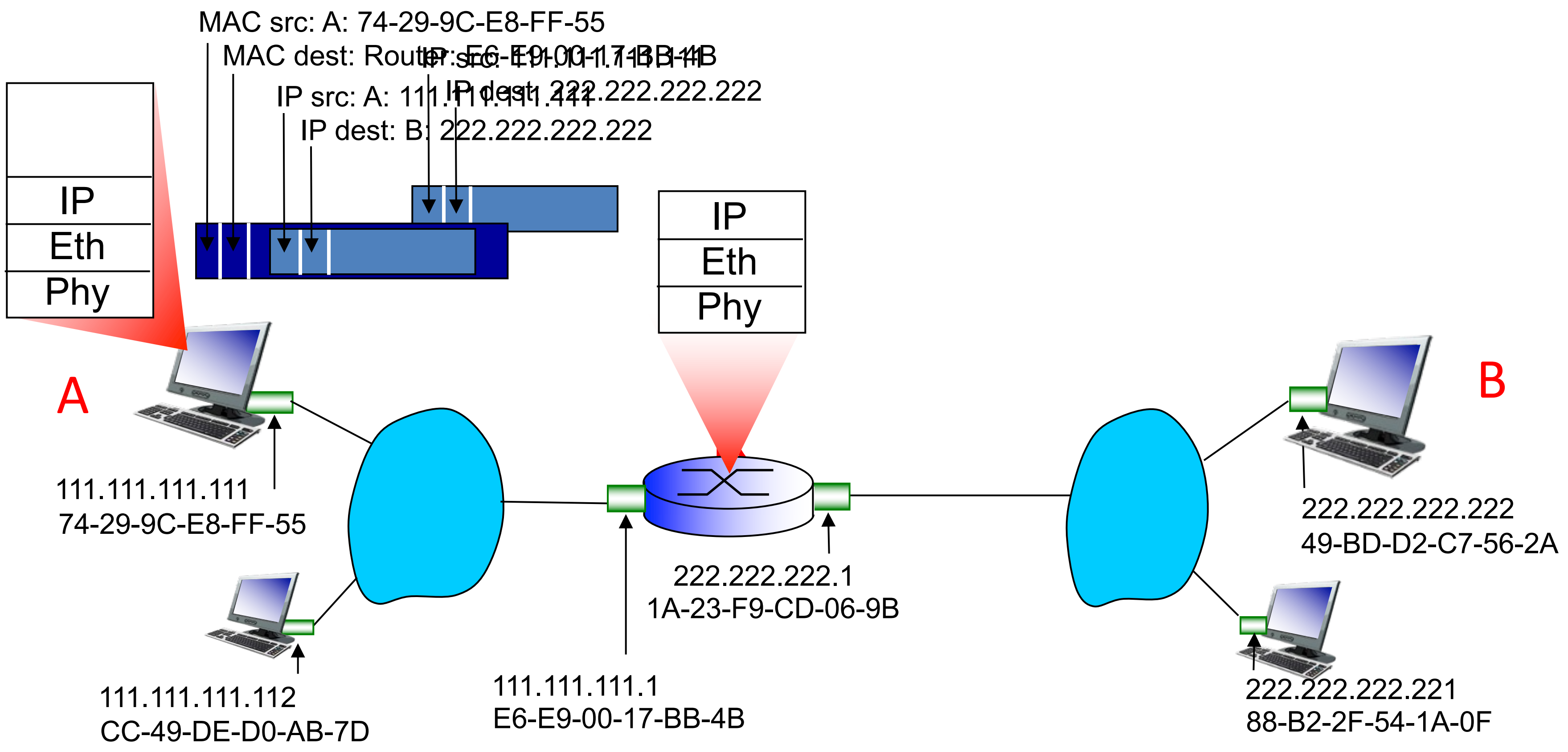
Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram



Addressing: routing to another LAN

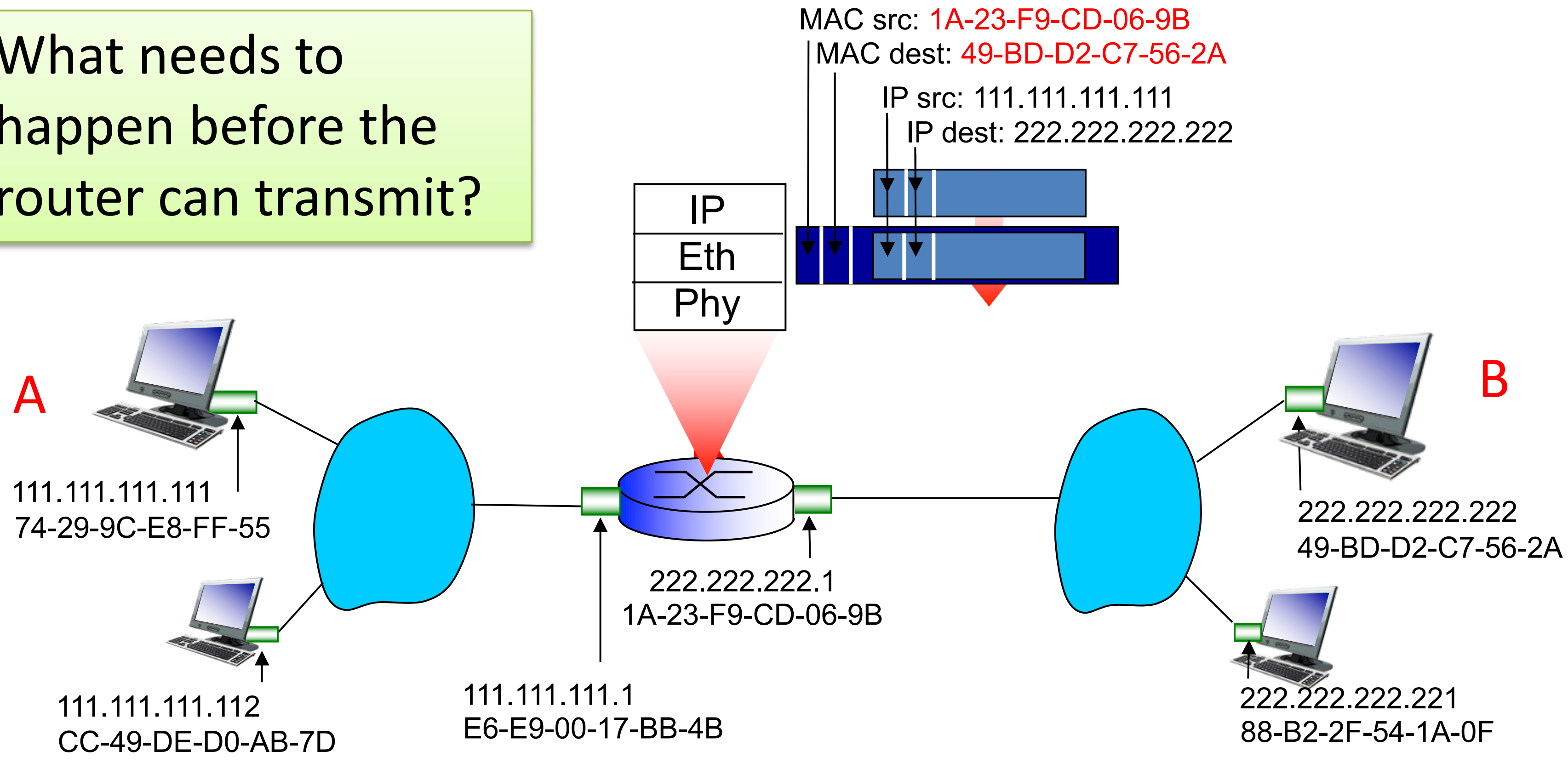
- frame sent from end host A to Router
- frame received at Router, datagram removed, passed up to IP



Addressing: routing to another LAN

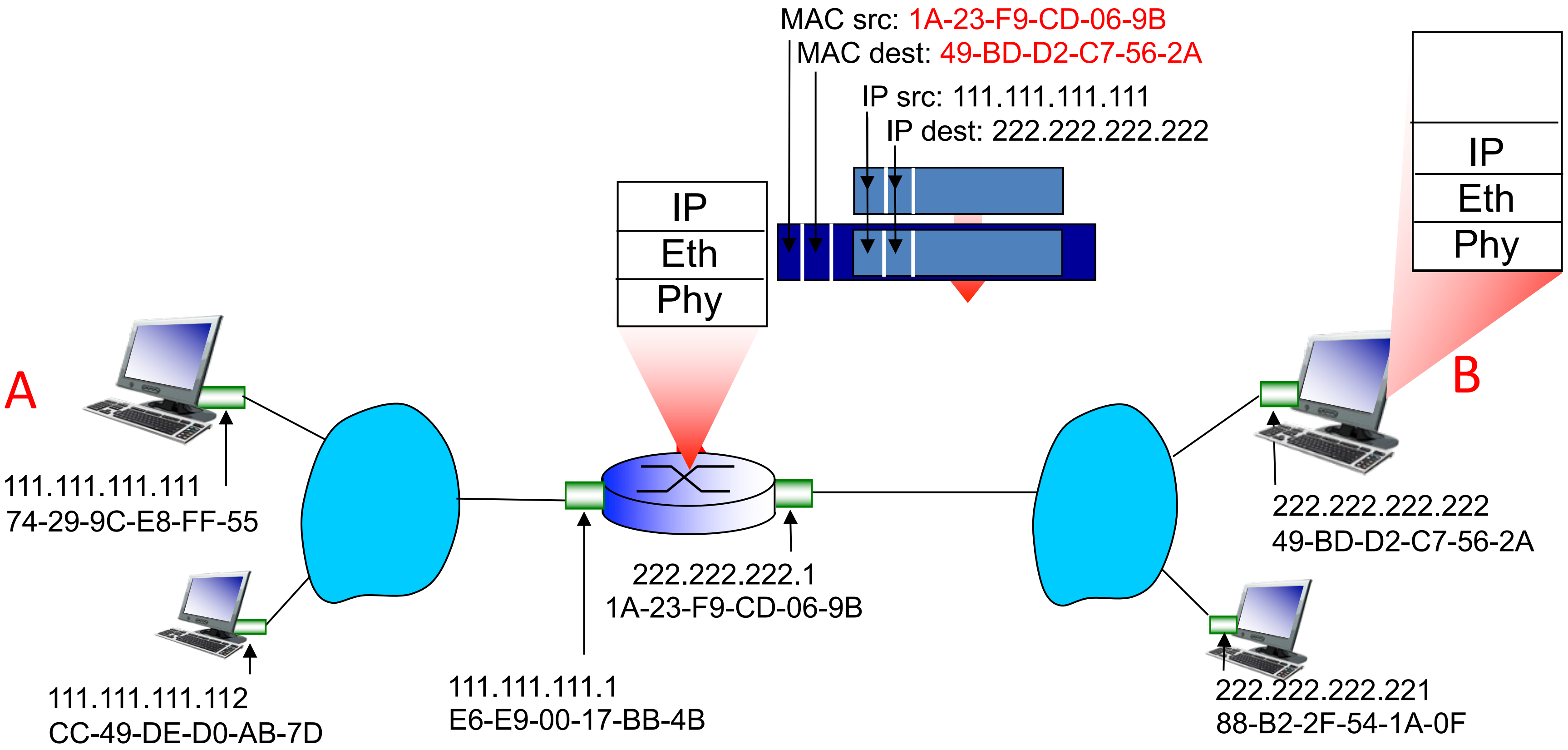
- Router forwards datagram with IP source A, destination B
- Router creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram

What needs to happen before the router can transmit?



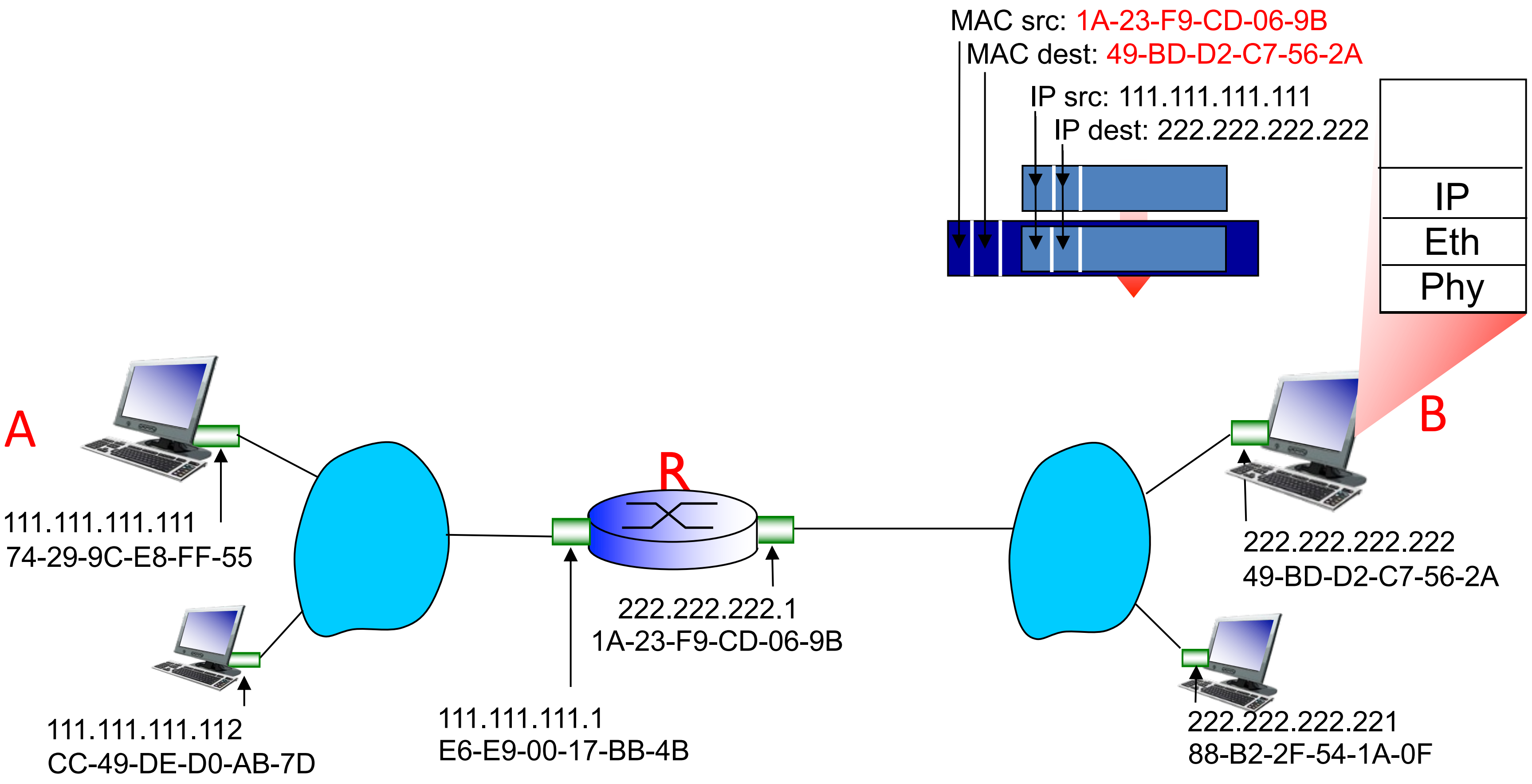
Addressing: routing to another LAN

- Router forwards datagram with IP source A, destination B
- Router creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Addressing: routing to another LAN

- Router forwards datagram with IP source A, destination B
- Router creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



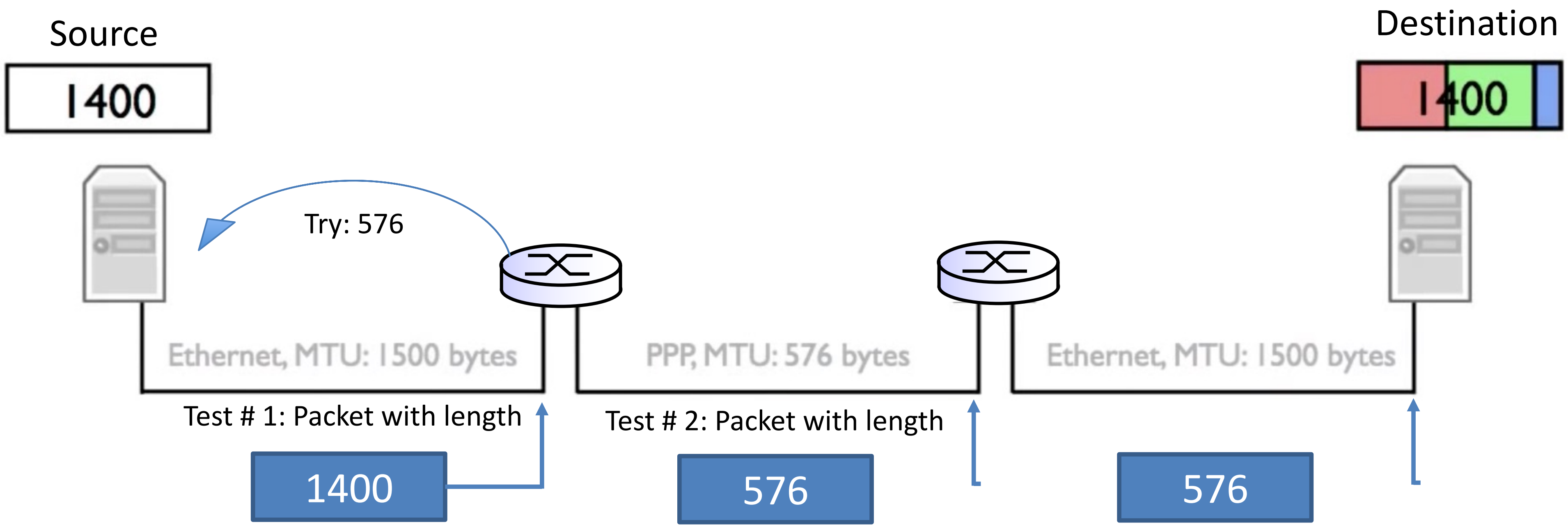
Today Link Layer Functions

1. Addressing: identifying endpoints

2. Framing: Dividing data into pieces that are sized for the network to handle.

3. Link access: Determining how to share the medium, who gets to send, and for how long.

Varying link capacities

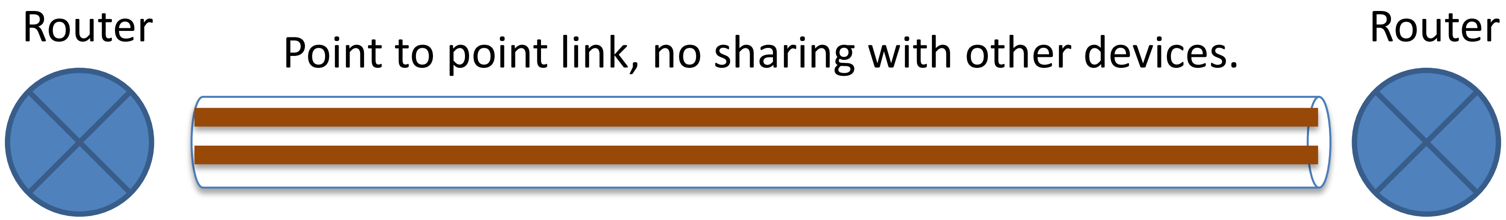


Link Layer Functions

1. Addressing: identifying endpoints
2. Framing: Dividing data into pieces that are sized for the network to handle.
3. Link access: Determining how to share the medium, who gets to send, and for how long.

Link Access

- Some networks may not require much.

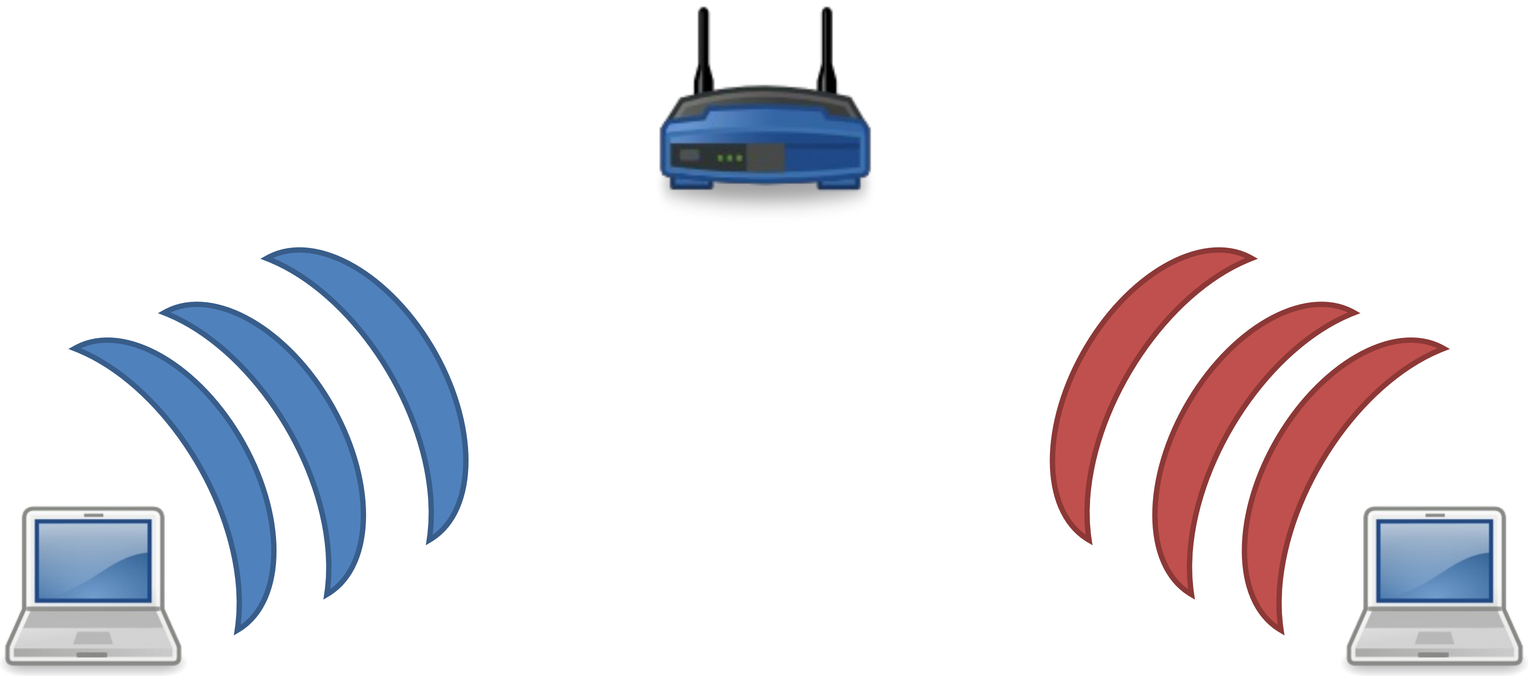


Example 1: Single copper wire, only one of them can send at a time.

Example 2: Two copper wires in cable, each can send on one simultaneously.

Link Access

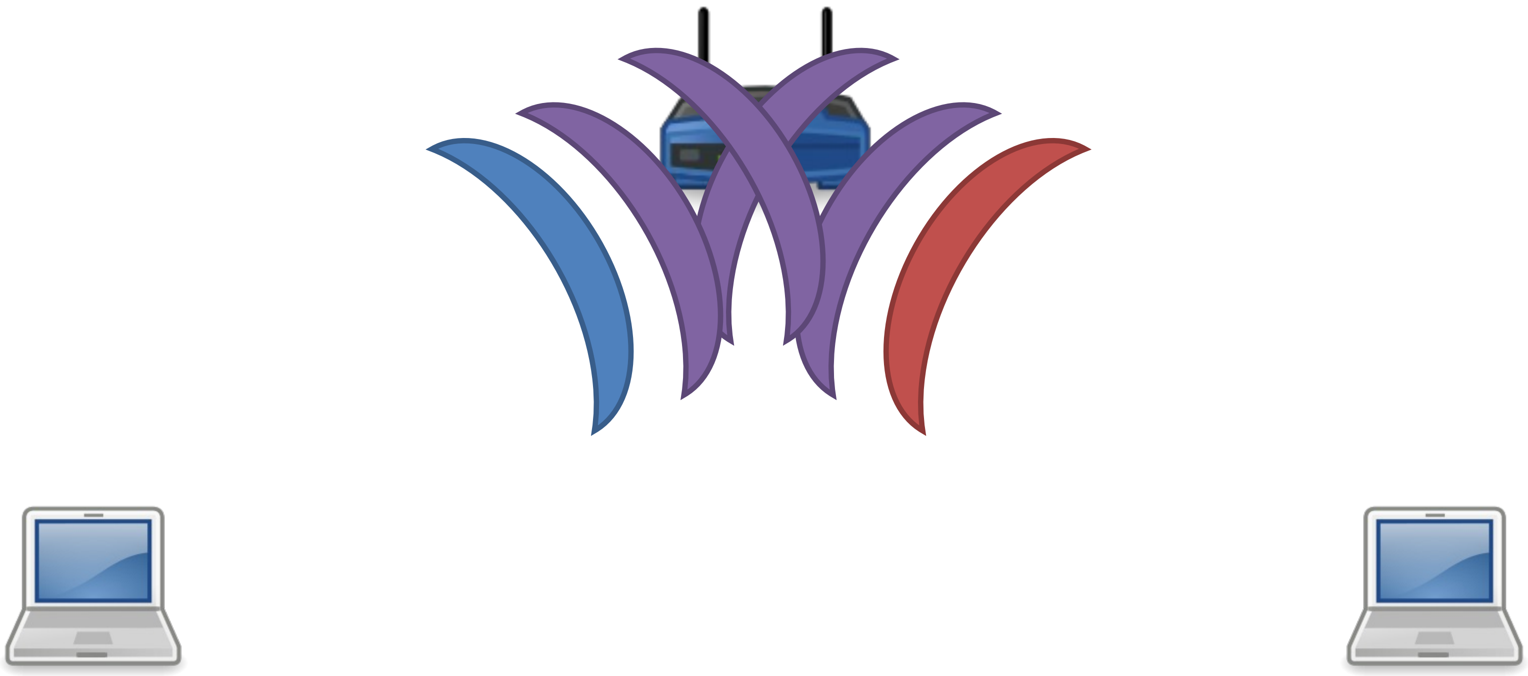
- For other networks, this is a huge challenge.



Link Access

- For other networks, this is a huge challenge.

Collision!



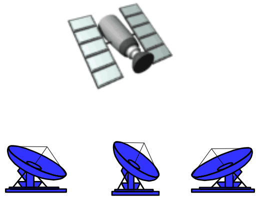
Multiple Access Links & Protocols

Two classes of "links":

- point-to-point
  - dial-up access
  - link between Ethernet switch, host
- *broadcast (shared wire or medium)*
  - old-fashioned Ethernet
  - 802.11 wireless LAN

shared wire (e.g., wired Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

Multiple Access Protocols

- Broadcast channel - every node hears every transmission
- If two or more nodes simultaneously transmit:
  - collision if node receives two or more signals at the same time

multiple access protocol

- algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel!
  - no out-of-band channel for coordination

An ideal multiple access protocol…

Given: broadcast channel of rate R bps

1. if only one node wants to transmit, it can send at rate R.
2. when M nodes want to transmit, each can send at average rate R/M (fairness)
3. fully decentralized:
   - no synchronization of clocks, time slots
   - no special node to coordinate transmissions
4. simple

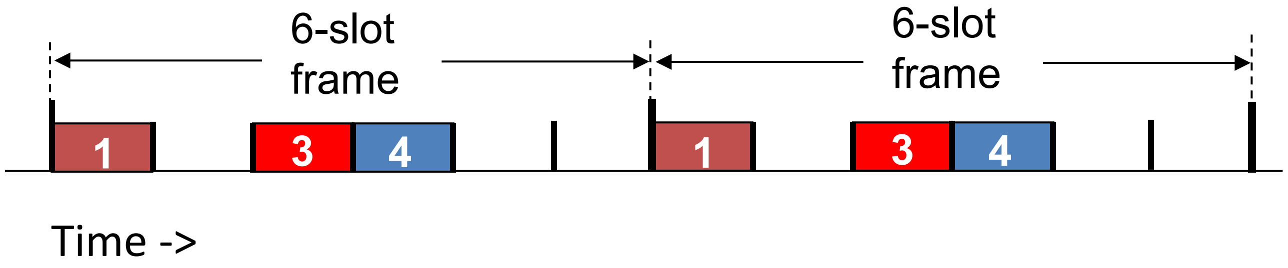
Media Access Control (MAC) Strategies

- channel partitioning
  - divide channel into smaller "pieces" (time slots, frequency, code)
  - allocate piece to node for exclusive use
- random access
  - channel not divided, allow collisions
  - "recover" from collisions
- taking turns
  - nodes coordinate with one another to take turns, share channel

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

- Access to channel in "rounds", like round robin
- Each node gets fixed length time slot (length = pkt trans time) in each round
- Example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle

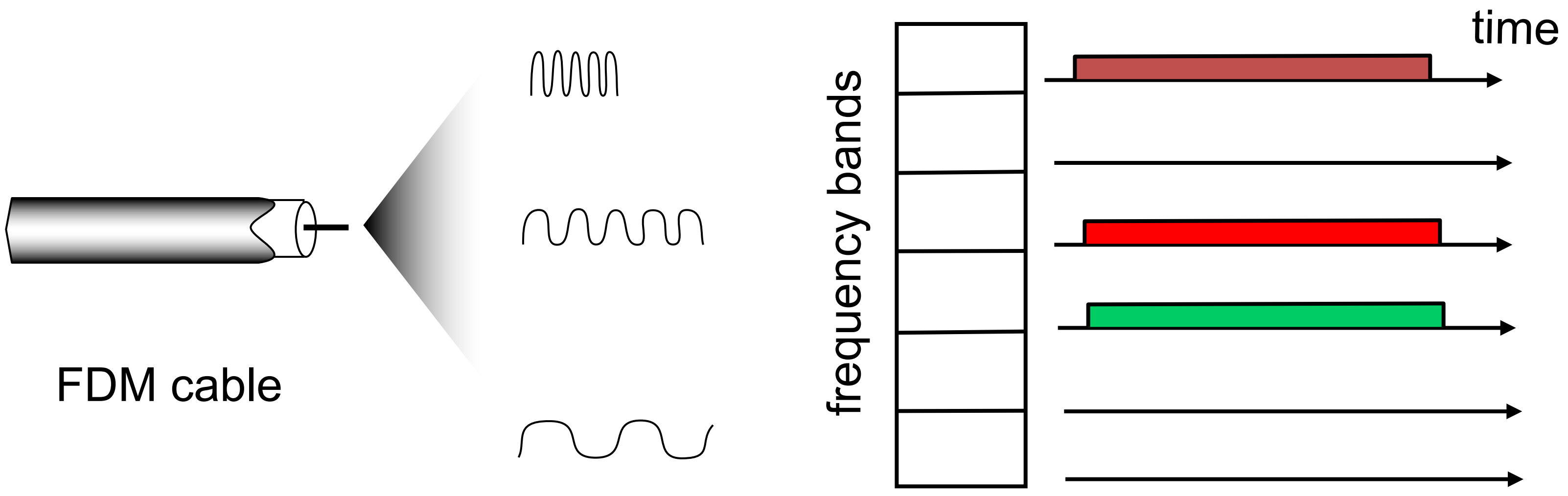


Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- Channel spectrum divided into frequency bands
- Each node assigned a fixed frequency band

- Example: 6-station LAN, 1,3,4 have pkt, bands 2,5,6 idle



Do we use channel partitioning?

- In what applications might this be a good idea?

- Terrestrial radio/TV (frequency division)
- Satellite (frequency division)
- Fiber optic links (wavelength division)
- Cell phones
  - Old generations (time division)
  - Current generation (code division)

Random Access Protocols

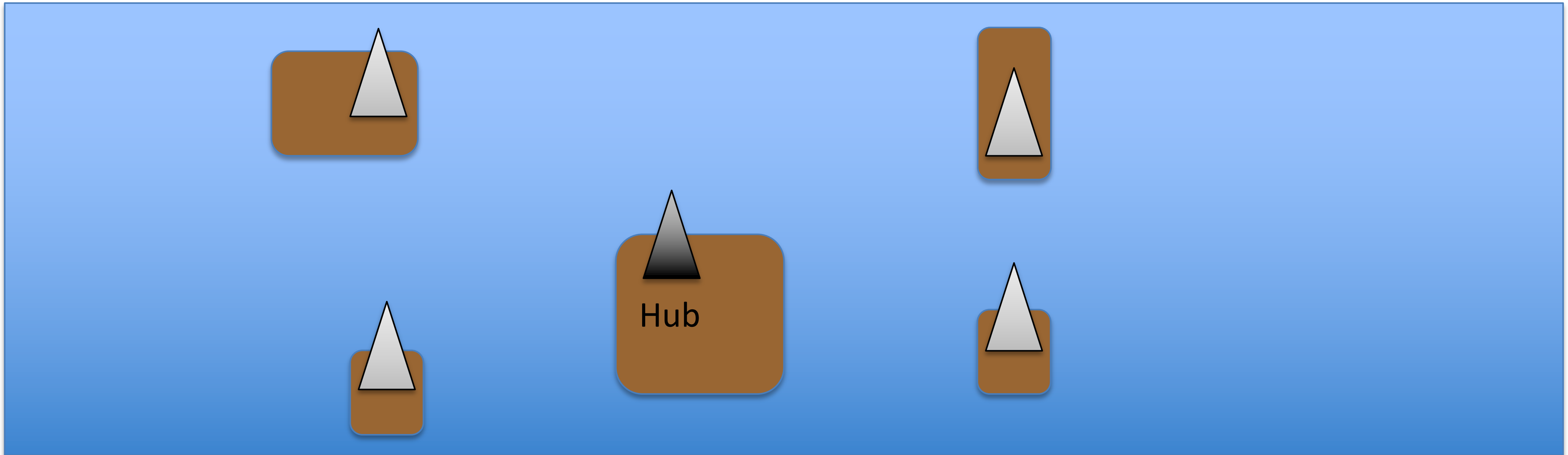
- When node has a packet to send, try to send it
  - no *a priori* coordination among nodes

- Two or more transmitting nodes ➜ "collision"

- random access MAC protocol specifies:
  - how to minimize collisions
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)

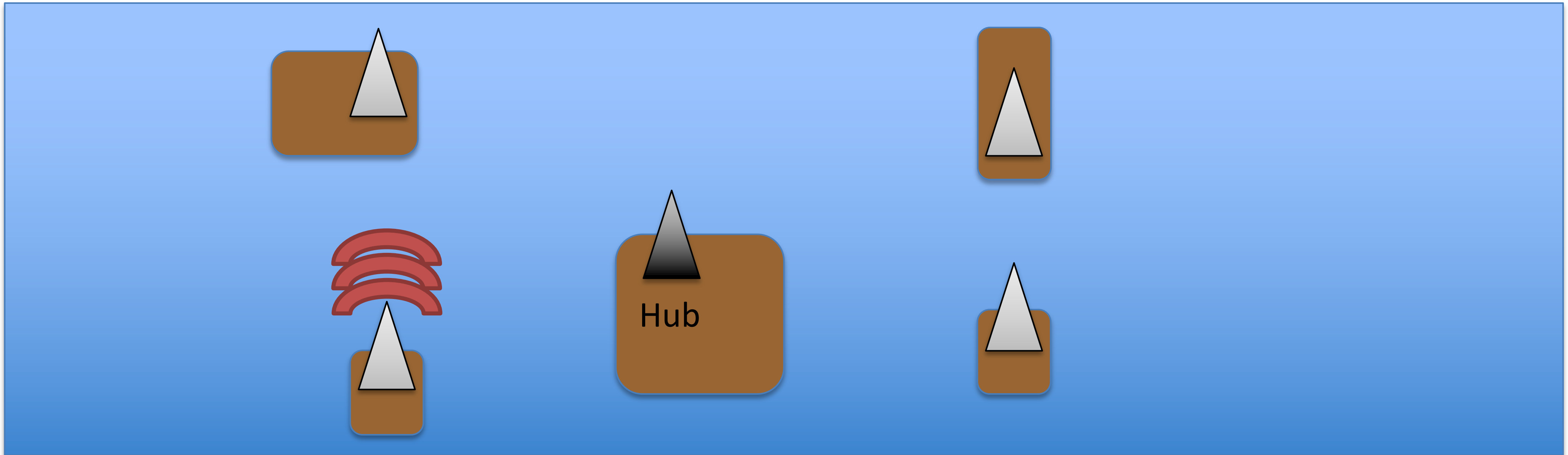
ALOHAnet (Unslotted / Pure)

- Norm Abramson at U of Hawaii in late 1960’s
- Goal: network between islands
- Shared medium: radio



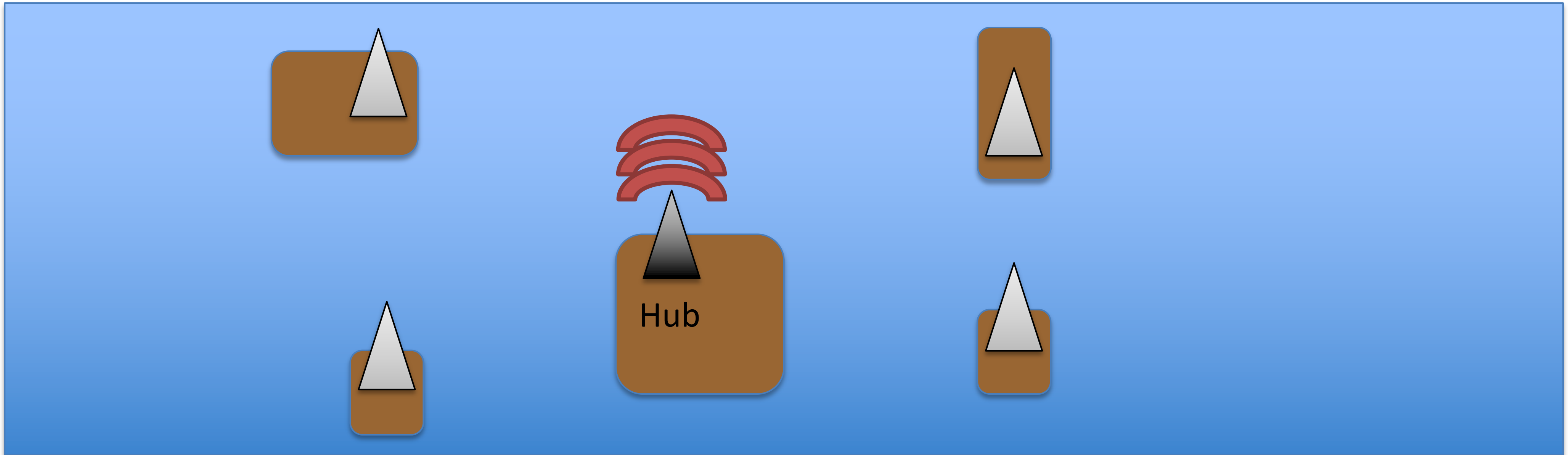
ALOHAnet

- If user gives you data, send it all, immediately.



ALOHAnet

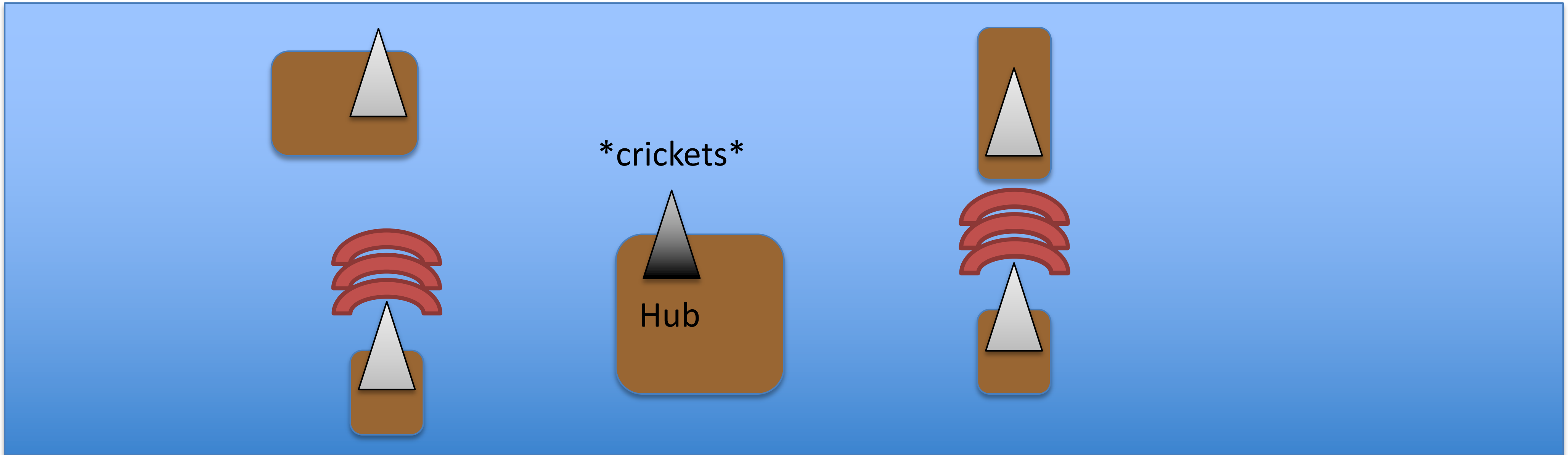
- If the hub received everything, it sends ACK.



ALOHAnet

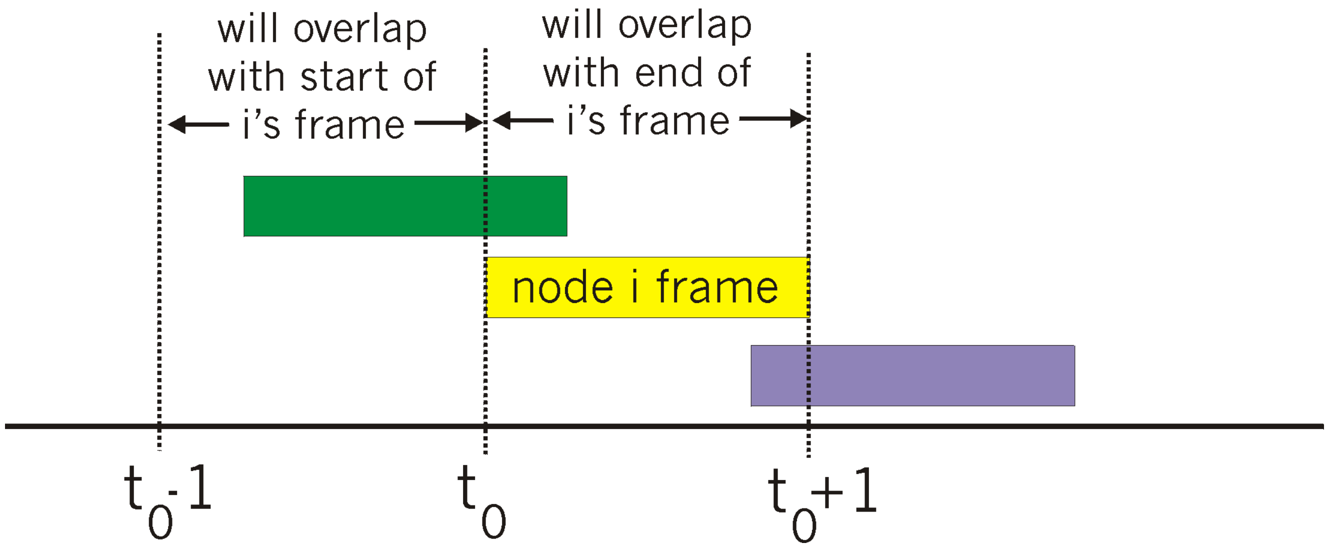
- If two senders collide…
- …hub sends back no ACKs.

- Senders wait a random time, send again.



(Unslotted / Pure) ALOHA

- Problems:
  - Sends immediately upon receiving data
  - Sends entire packets all at once



Carrier Sensing Multiple Access (CSMA)

CSMA: listen before transmit:

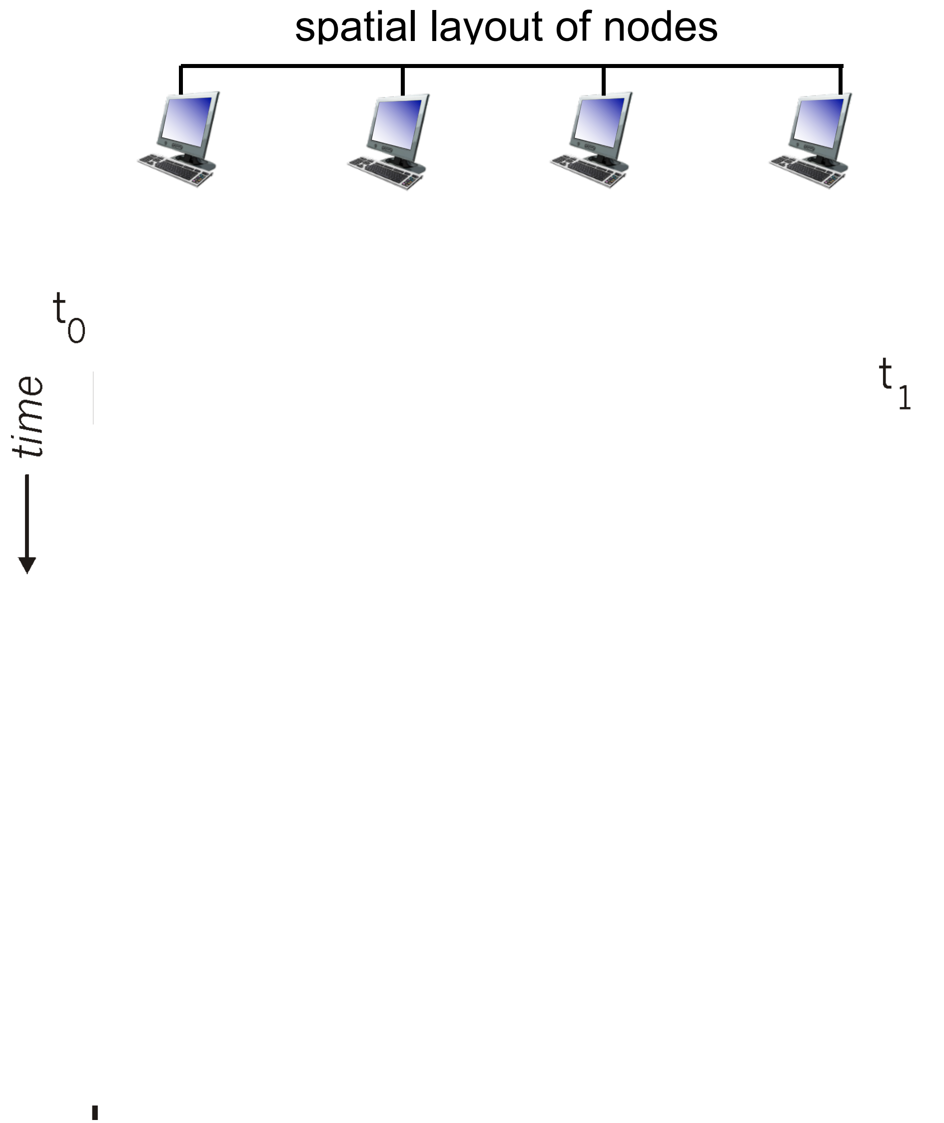
if channel sensed idle: transmit

- if channel sensed busy, defer transmission

human analogy: don’t interrupt others!

CSMA collisions

- Collisions can still occur: propagation delay means two nodes may not hear each other's transmission
- Collision: entire packet transmission time wasted
  - distance & propagation delay play role in in determining collision probability



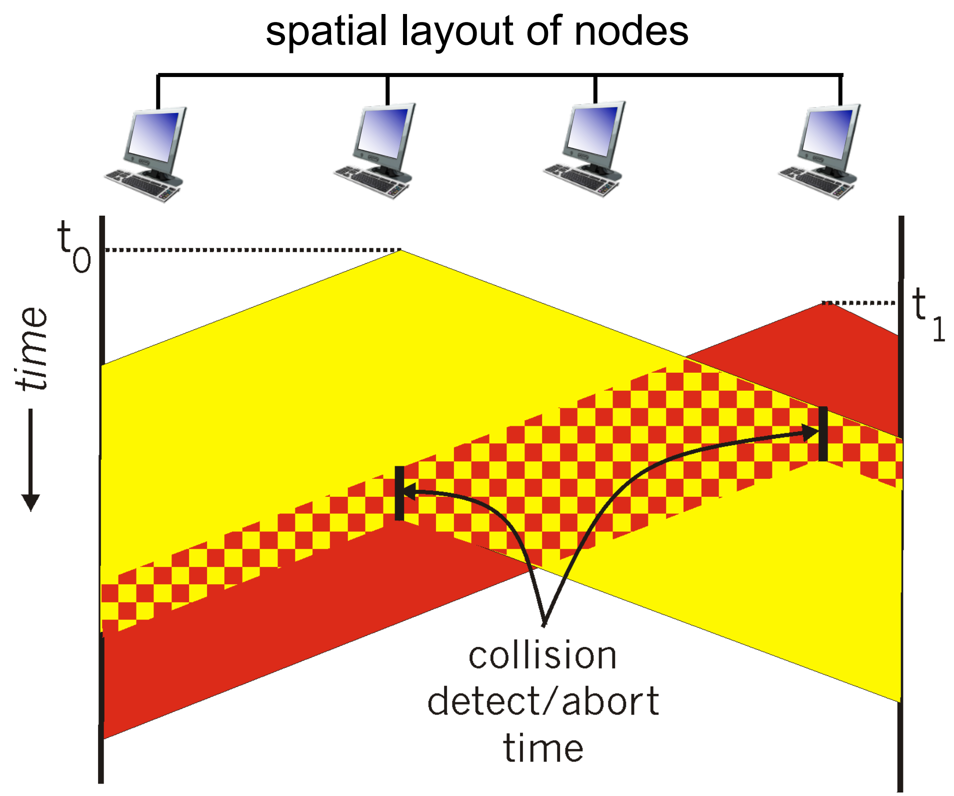
CSMA/CD (Collision Detection)

CSMA/CD: carrier sensing, deferral as in CSMA

- collisions detected within short time
- colliding transmissions aborted, freeing channel

- Collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

CSMA/CD (collision detection)



Ethernet and CSMA/CD

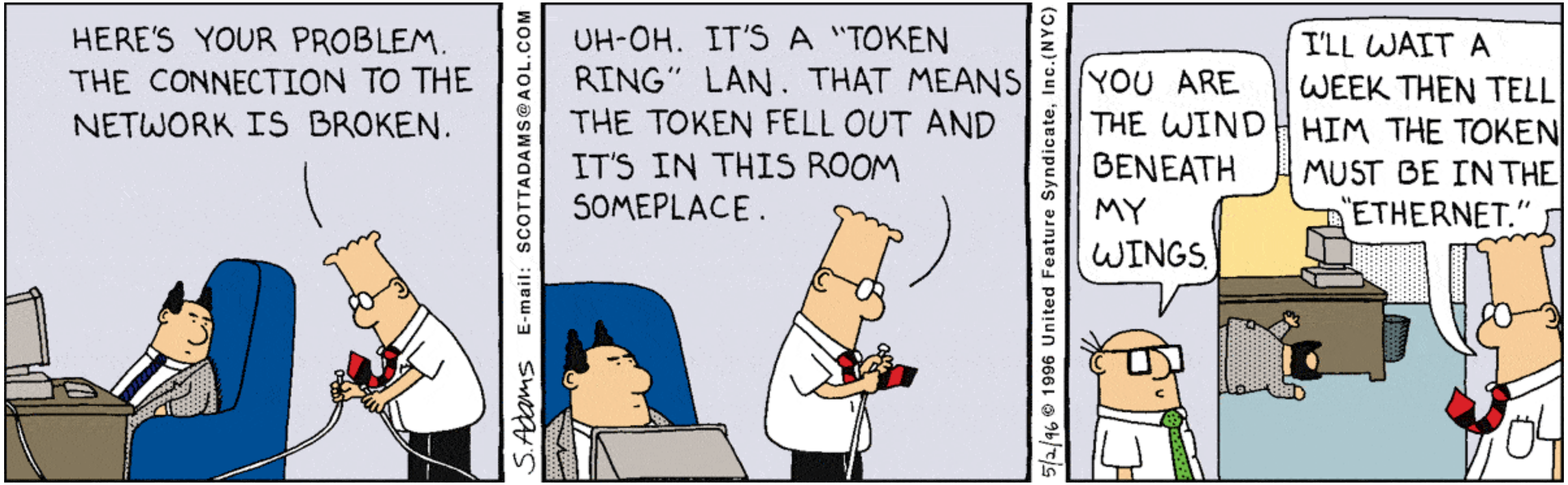
1. NIC (Network Interface Card) receives datagram from network layer, creates frame.
2. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, then transmits.
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame!
4. If NIC detects another transmission while transmitting, aborts and sends jam signal (maximize interference).
5. After aborting, NIC enters binary (exponential) backoff to send data again.

Exponential Back off

- After *mth collision*, NIC chooses *K* at random from *{0,1,2, …, 2^m-1}*.
- NIC waits K·512 bit times, then returns to checking if the channel is idle
- Longer back-off interval with more collisions

Like Human Conversation…

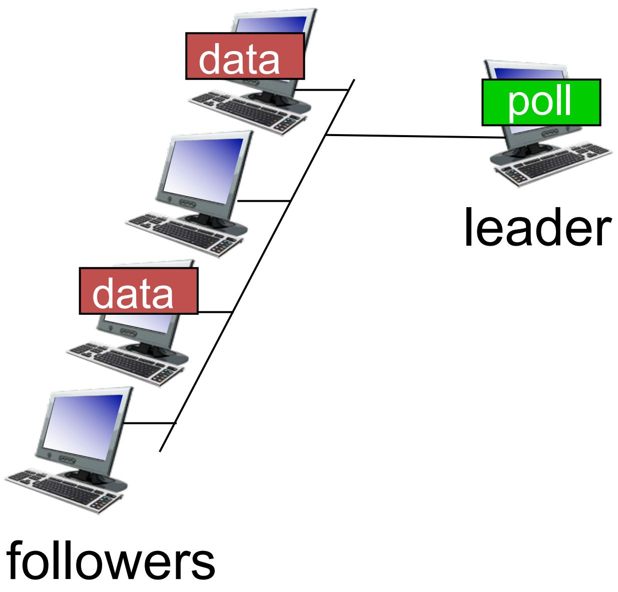
- Carrier sense
  - – Listen before speaking
  - – …and don't interrupt!
- Collision detection
  - – Detect simultaneous talking
  - – … and shut up!
- Random access
  - – Wait for a random period of time
  - – … before trying to talk again!


HERE'S YOUR PROBLEM. THE CONNECTION TO THE NETWORK IS BROKEN.
S. Adams E-mail: SCOTTADAMS@AOL.COM
UH-OH. IT'S A "TOKEN RING" LAN. THAT MEANS THE TOKEN FELL OUT AND IT'S IN THIS ROOM SOMEPLACE.
5/2/96 © 1996 United Feature Syndicate, Inc.(NYC)
YOU ARE THE WIND BENEATH MY WINGS.
I'LL WAIT A WEEK THEN TELL HIM THE TOKEN MUST BE IN THE "ETHERNET."

“Taking turns” MAC protocols

Polling:

- leader node “invites” follower nodes to transmit in turn
- typically used with “dumb” follower devices
- Concerns:
  - polling overhead
  - latency
  - centralized leader



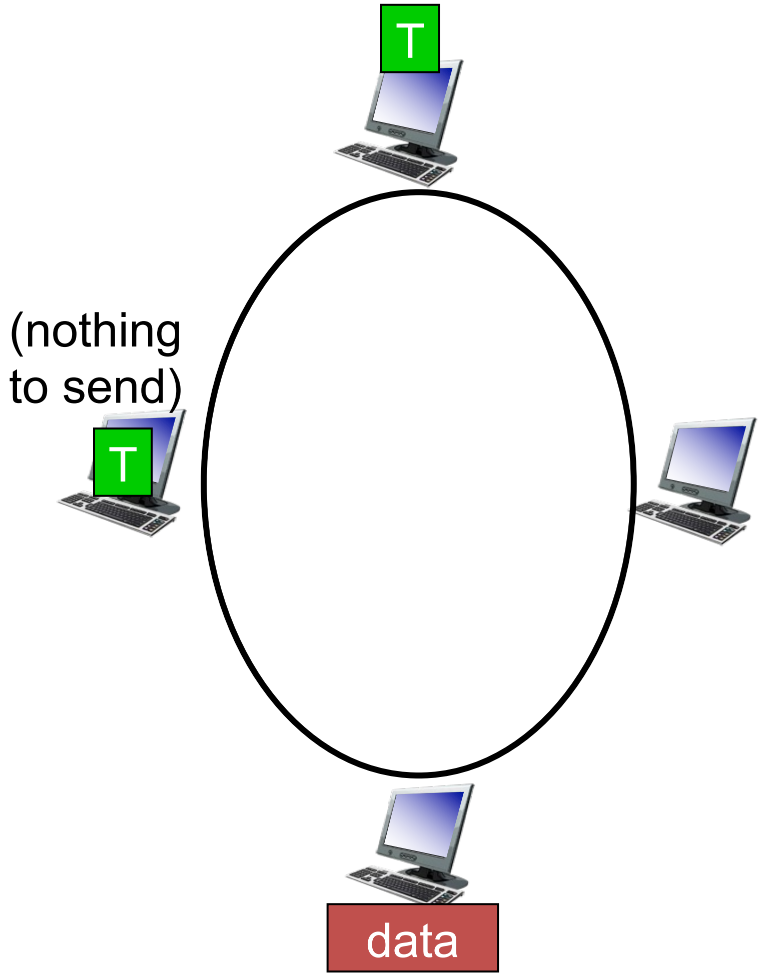
"Taking turns" MAC protocols

Token passing:

- Control token passed from one node to next sequentially.

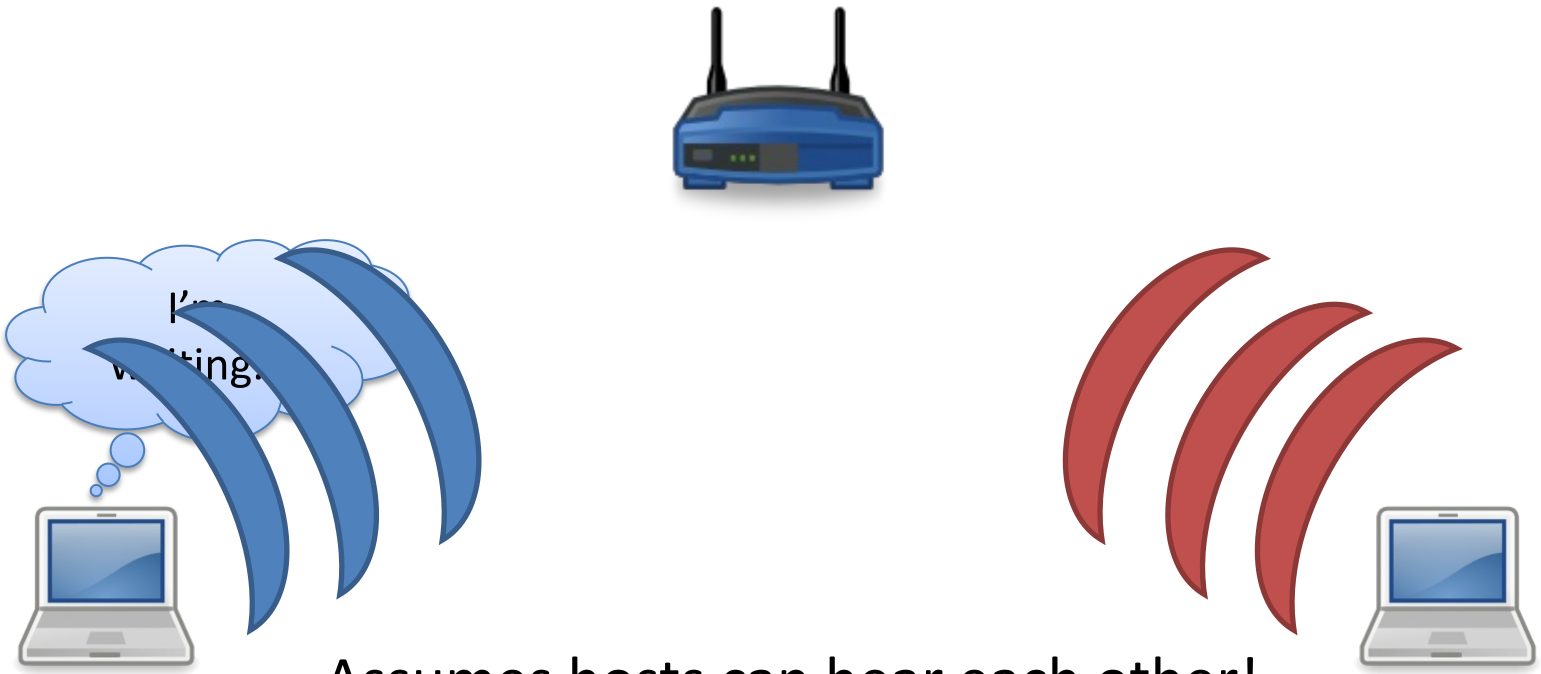
- Can only transmit if holding the token.

- Limit on number of bytes sent per token.



WiFi (802.11)

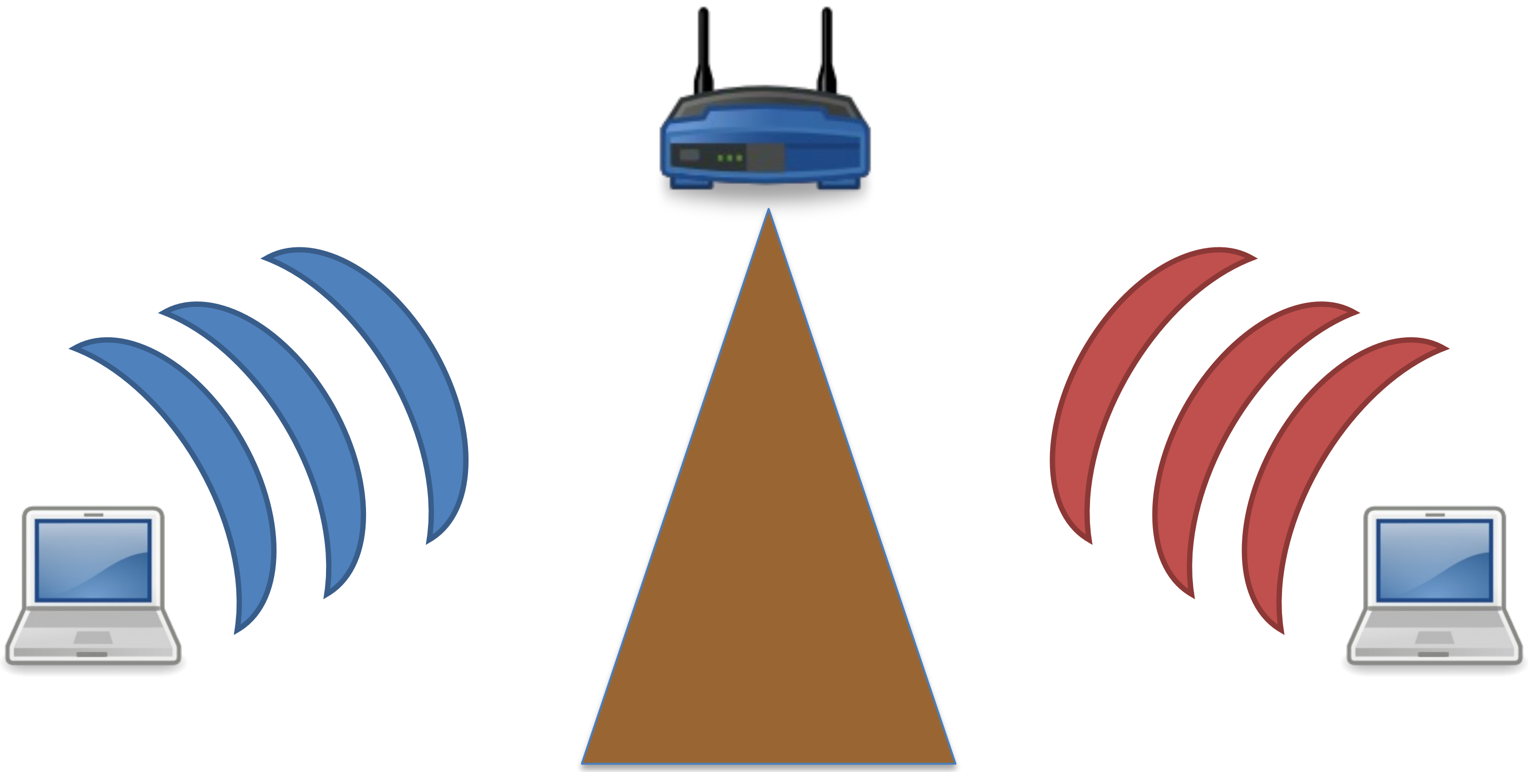
- Senders do carrier sensing like Ethernet.



Assumes hosts can hear each other!

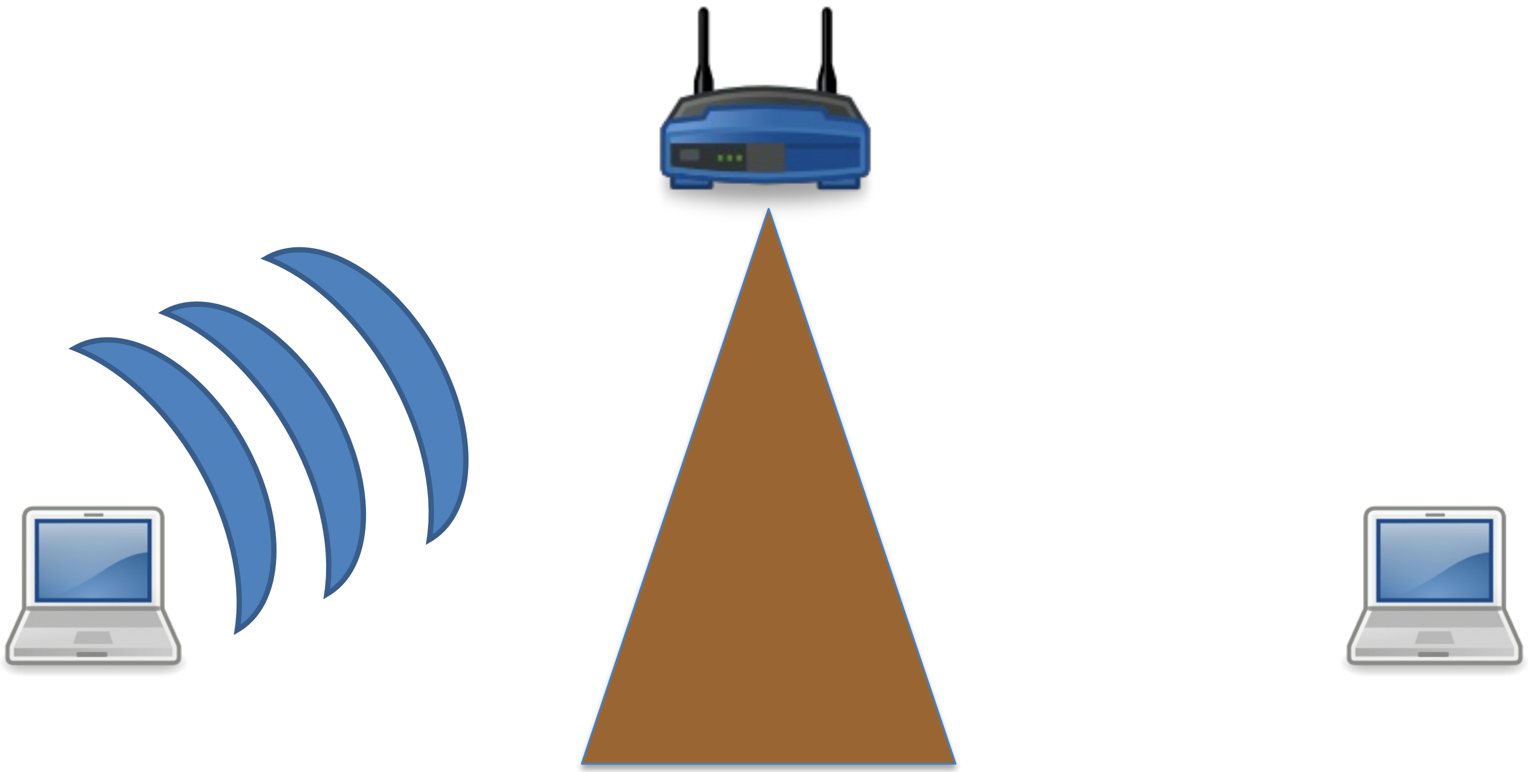
"Hidden Terminal" Problem

- Senders collide at receiver, but they can't hear each other!



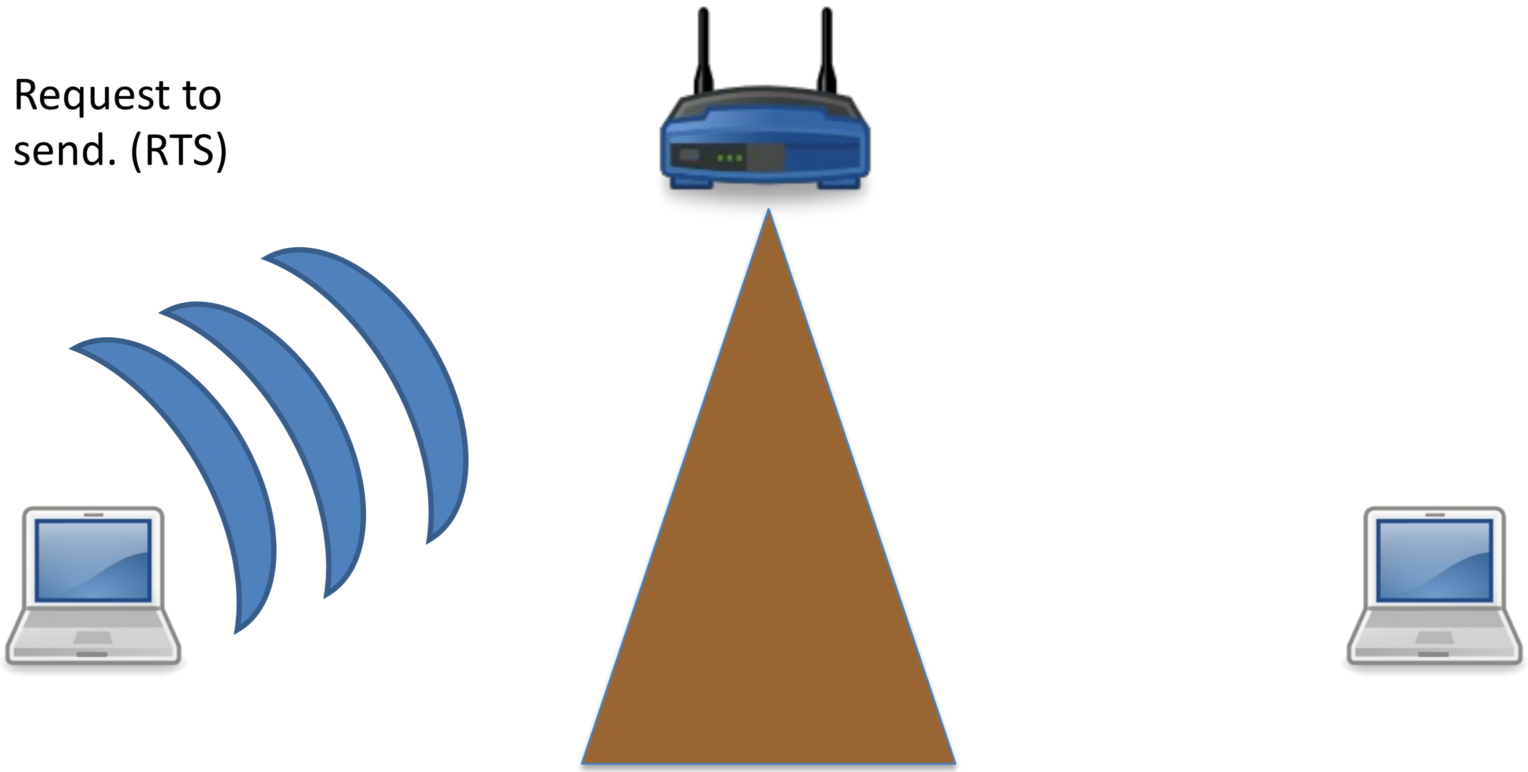
CSMA/CA (Collision Avoidance)

- If sending small (threshold configurable) frame, just send it.



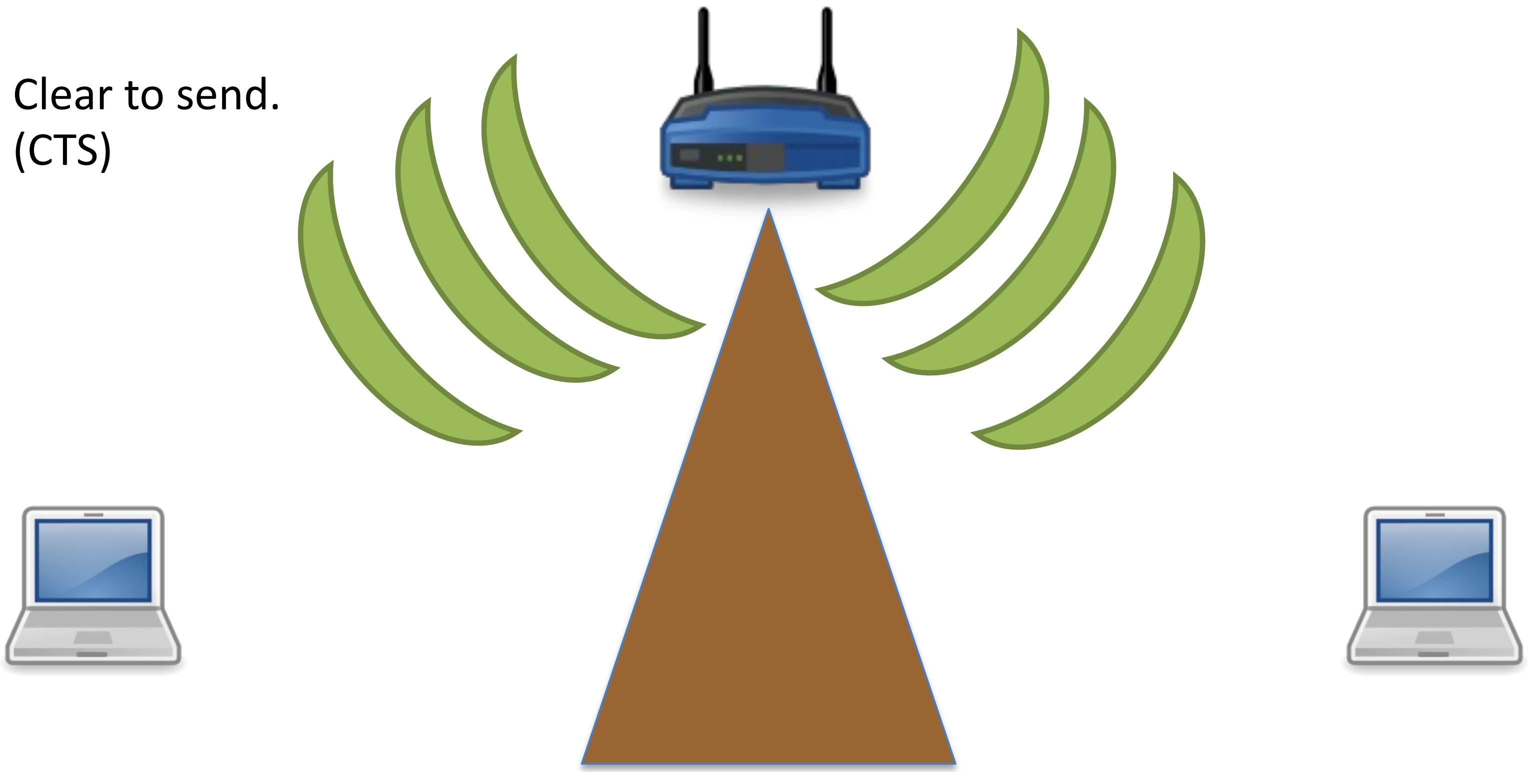
CSMA/CA (Collision Avoidance)

- If sending large frame, ask for permission first.



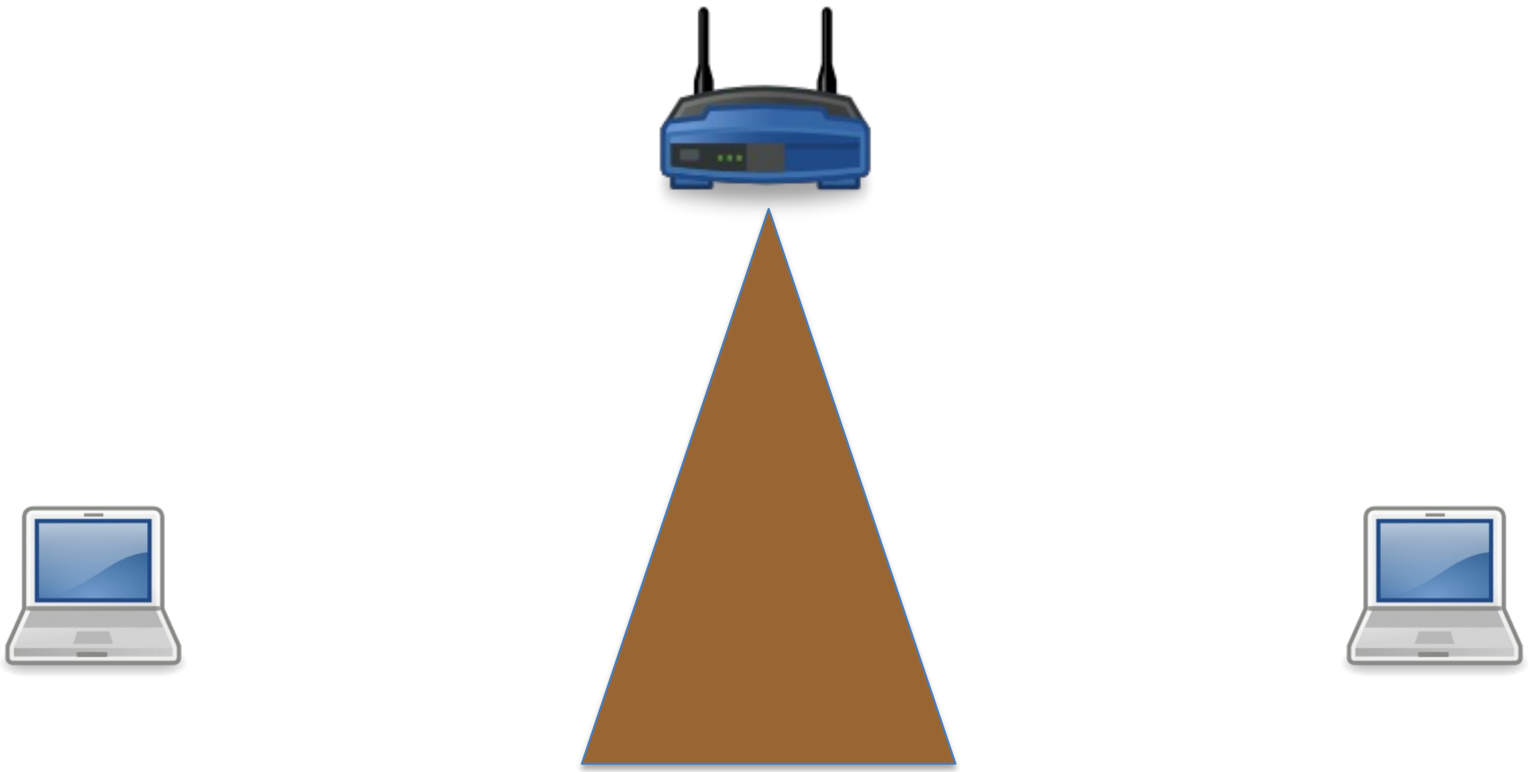
CSMA/CA (Collision Avoidance)

- If granted, it will be heard by everyone.



CSMA/CA (Collision Avoidance)

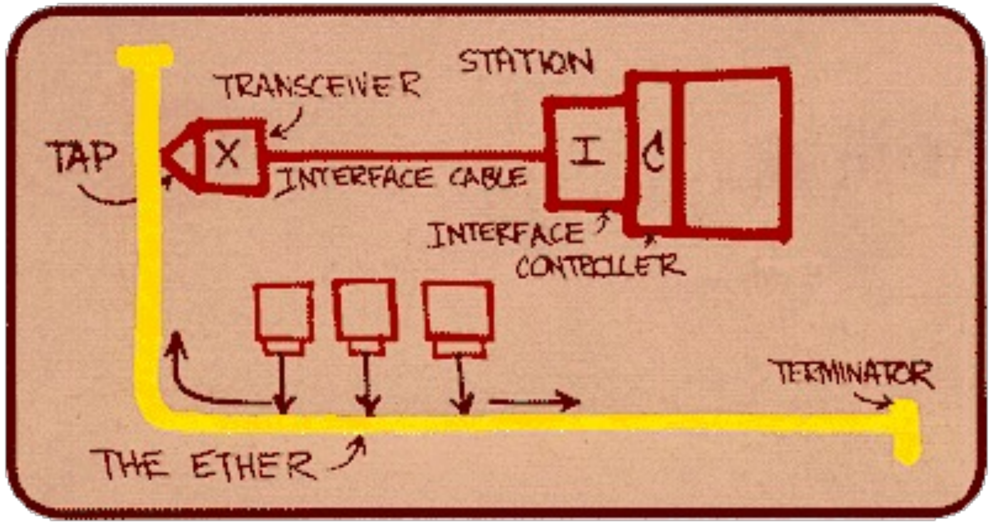
- RTS/CTS is like taking turns.



Summary of MAC protocols

- channel partitioning, by time, frequency or code
  - Time Division, Frequency Division
- random access (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing:
    - easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11
- taking turns
  - Polling from central site, token passing
  - Bluetooth, FDDI, token ring

Ethernet



Metcalfe's Ethernet sketch

"Dominant" wired LAN technology:

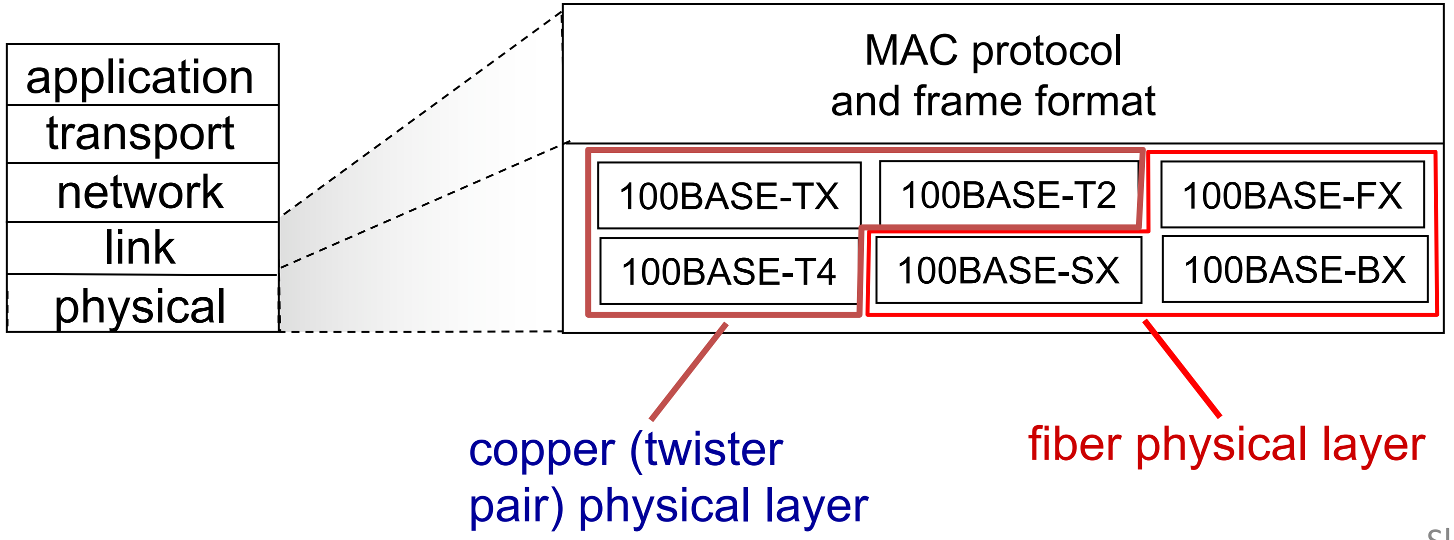
- cheap $20 for NIC
- first widely used LAN technology
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 10 Gbps

Ethernet: unreliable, connectionless

- Connectionless: no handshaking between sending and receiving NICs
- Unreliable: receiving NIC doesn't send acks or nacks to sending NIC
  - data in dropped frames recovered only if initial sender uses higher layer reliable delivery (e.g., TCP), otherwise dropped data lost
- Ethernet's MAC protocol: CSMA/CD with binary exponential backoff

802.3 Ethernet standards: link & physical layers

- Many different Ethernet standards
  - Common MAC protocol and frame format
  - Speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10Gbps
  - Physical layer media: fiber, copper cable



Ethernet frame structure

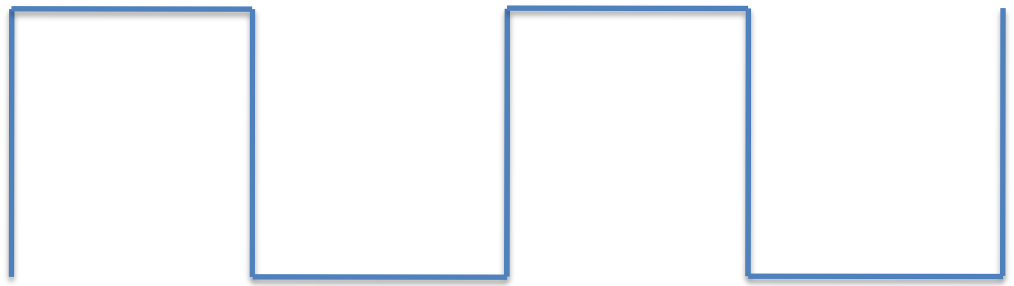
Sender encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

preamble:

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

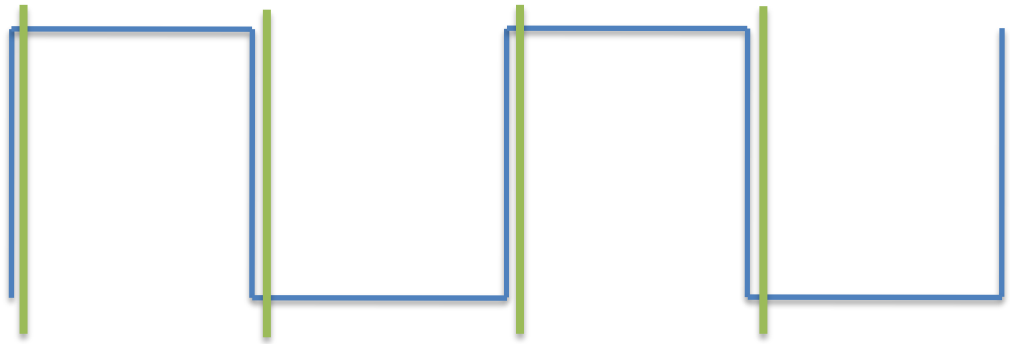
Clock Synching

- Bits represented as voltages, either low or high
- We will read one bit per clock cycle



Clock Synching

- Bits represented as voltages, either low or high
- We will read one bit per clock cycle

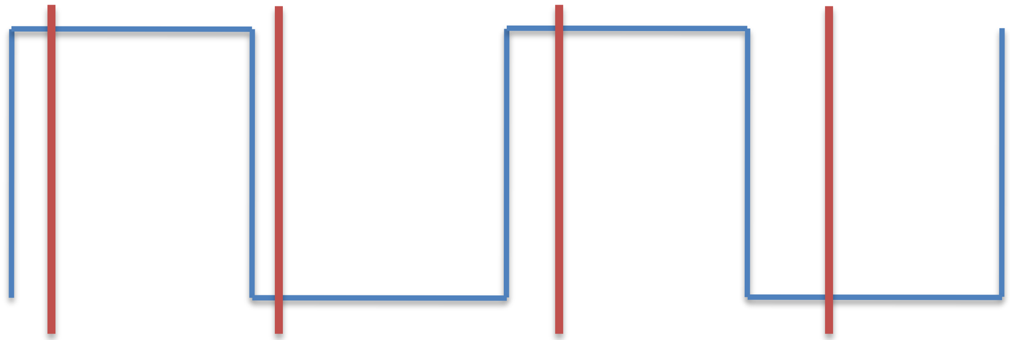


Ideal receiver: Sample signal at regular interval.

For 1 Gbps Ethernet, ~1 nanosecond interval.

Clock Synching

- Bits represented as voltages, either low or high
- We will read one bit per clock cycle



Problem: receiver clock may not agree with sender!

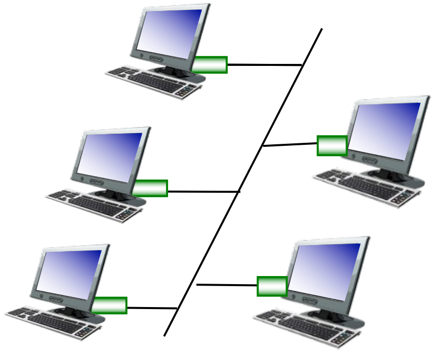
Preamble let's receiver see several 0 -> 1 -> 0 -> … transitions.

Ethernet frame structure (more)

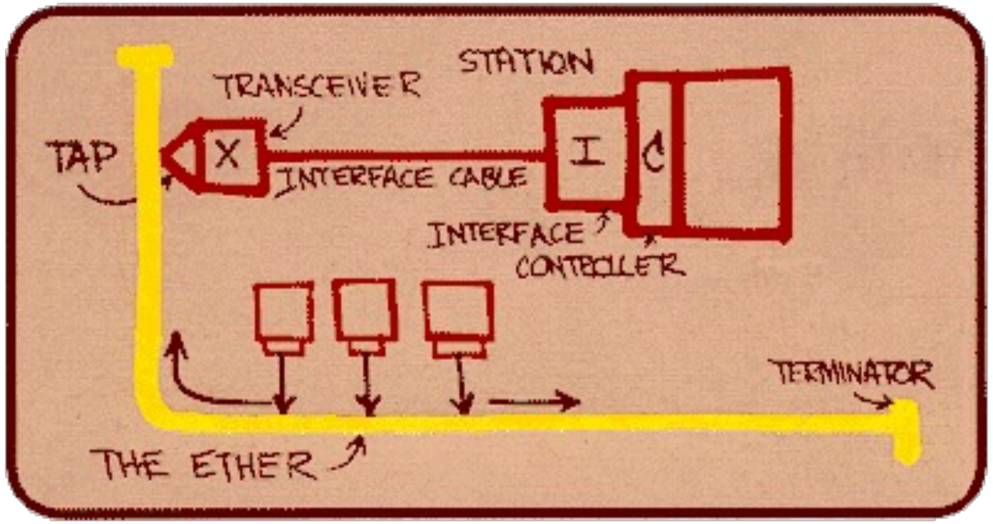
- addresses: 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- type: indicates higher layer protocol (mostly IP)
- CRC: cyclic redundancy check at receiver
  - error detected: frame is dropped

Physical Topology: Bus

- *Bus:* popular through mid 90s
  - all nodes in same collision domain (transmissions collide with each other)

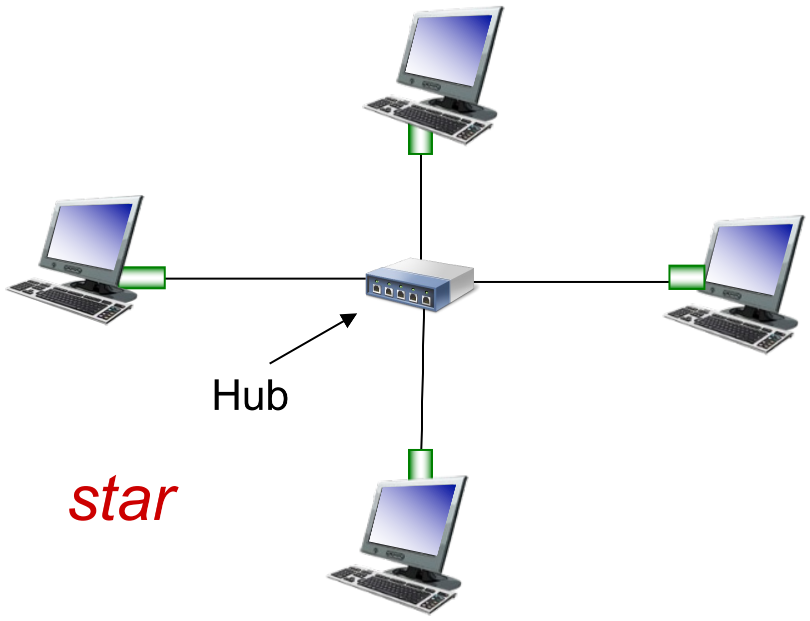


bus: coaxial cable



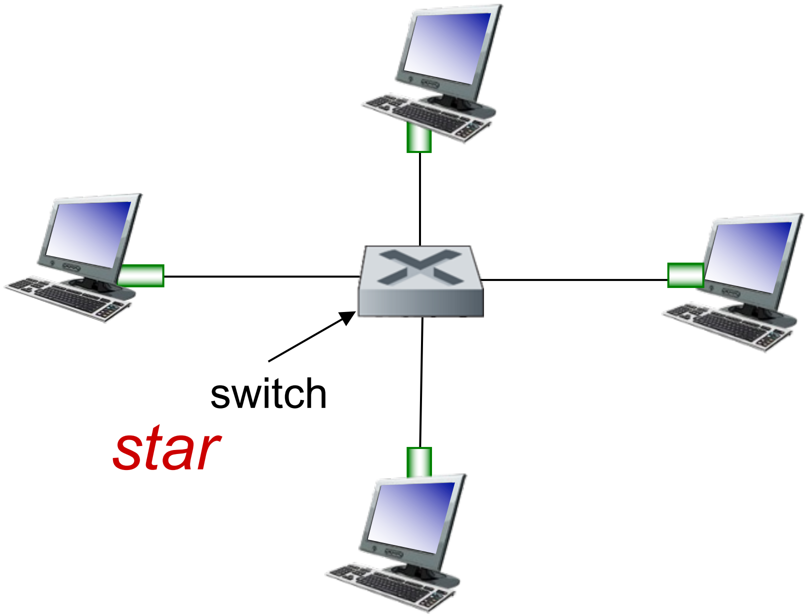
Physical Topology: Star

- *Hub* in the center:
  - broadcasts all messages to all hosts
  - retransmits on collisions
  - often considered a physical layer device (like a bus wire)

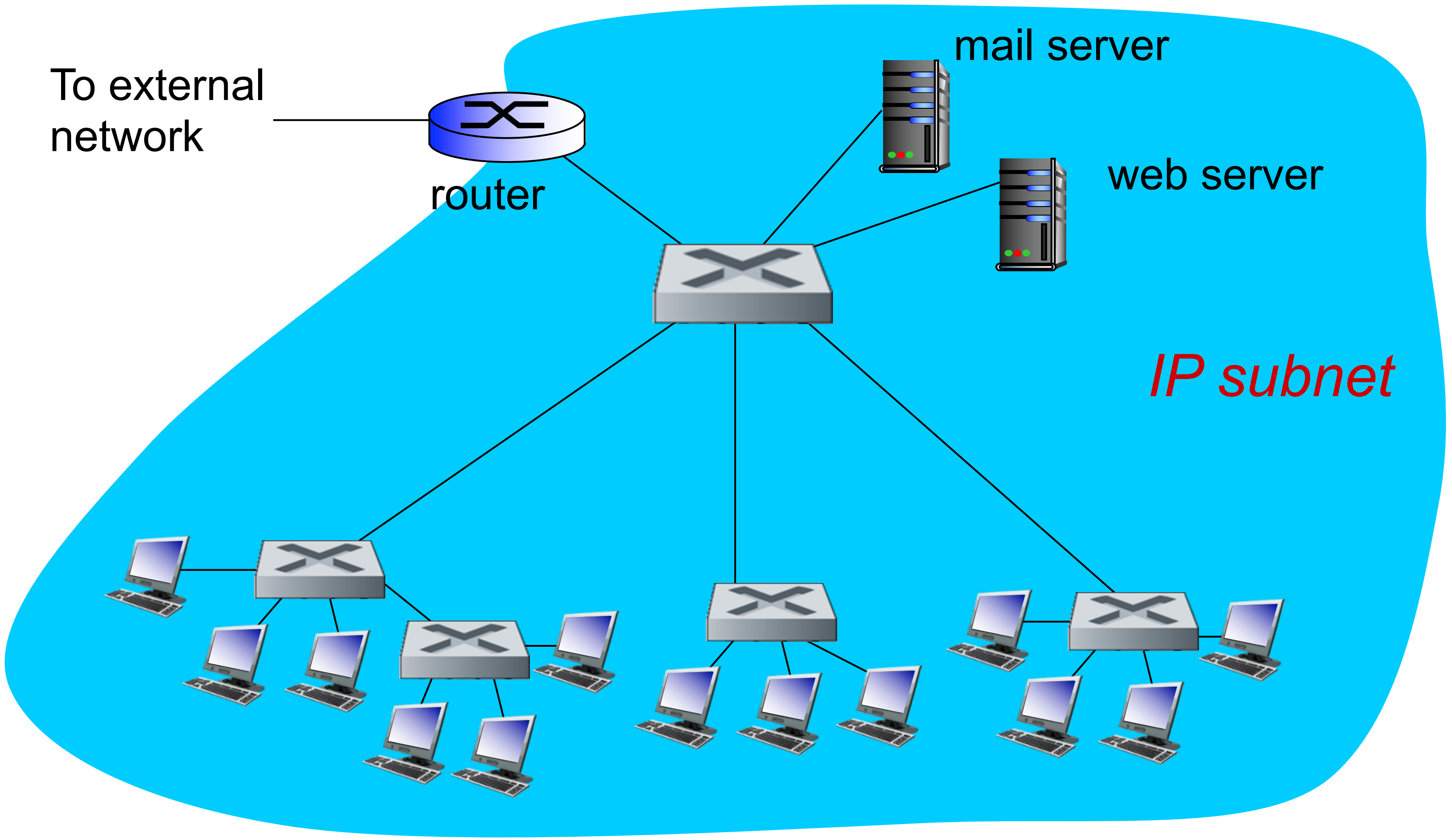


Physical Topology: Star (Switched)

- *Switch:* prevails today
  - each "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)
  - Full duplex: No collisions on spoke



Institutional Network (Tree)

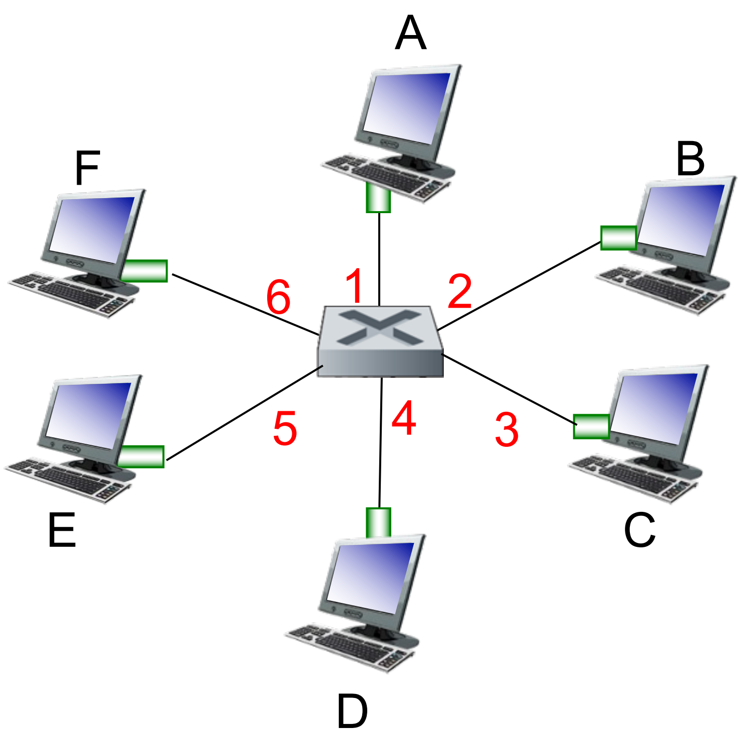


Ethernet switch

- link-layer device: takes an *active* role
  - store, forward Ethernet frames
  - examines incoming frame's MAC address, selectively forwards frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- transparent
  - hosts are unaware of presence of switches
- plug-and-play, self-learning
  - switches do not need to be configured

Switch: multiple simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-D and B-to-E can transmit simultaneously, without collisions

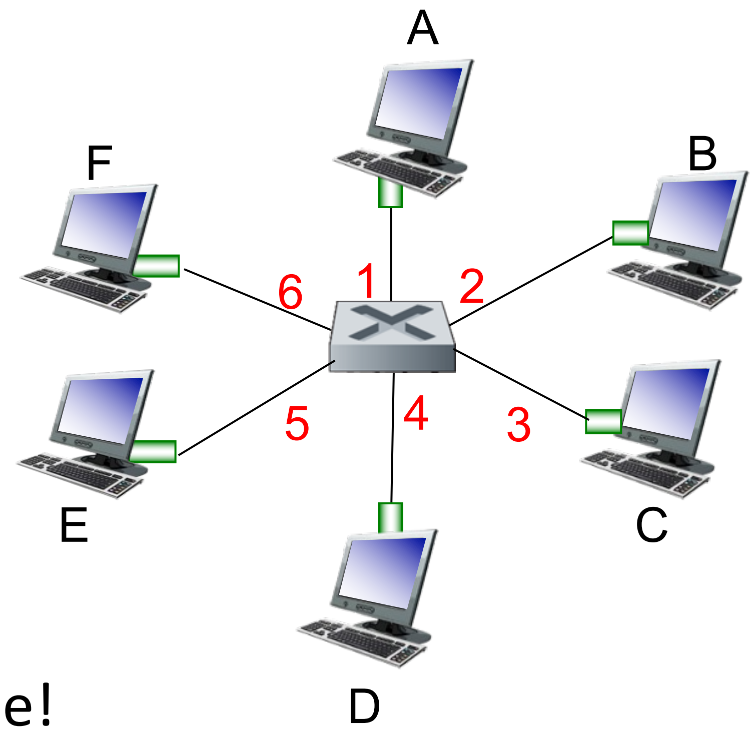


switch with six interfaces (1,2,3,4,5,6)

Switch forwarding table

Q: how does switch know D reachable via interface 4, E reachable via interface 5?

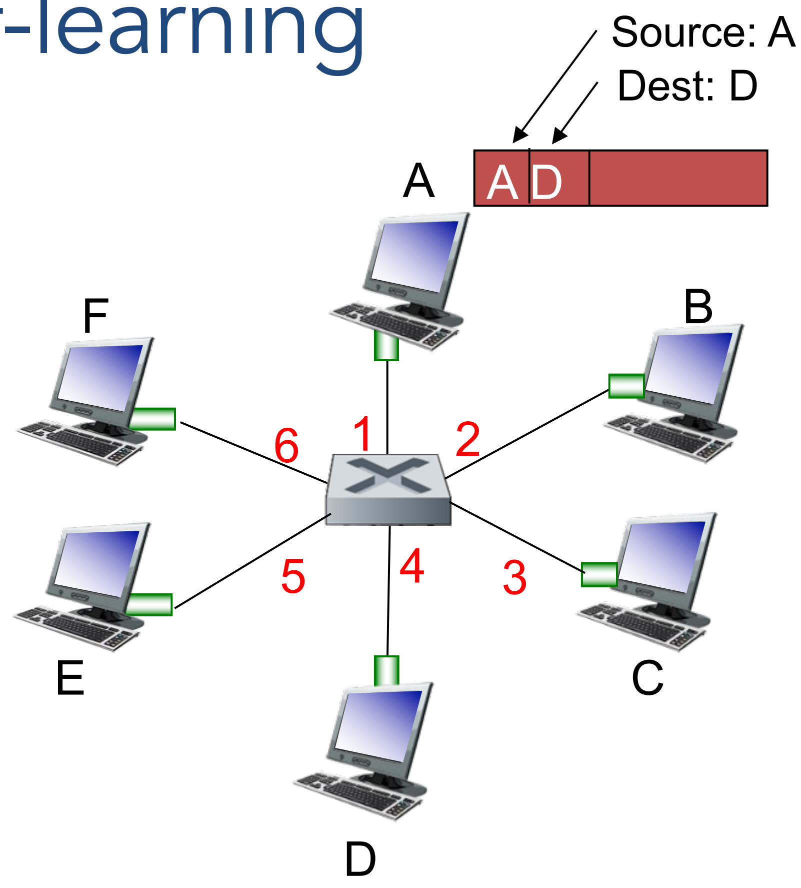
- A: each switch has a forwarding table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
  - looks like a router's forwarding table!



switch with six interfaces (1,2,3,4,5,6)

Switch: self-learning

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
  - records sender/location pair in switch table

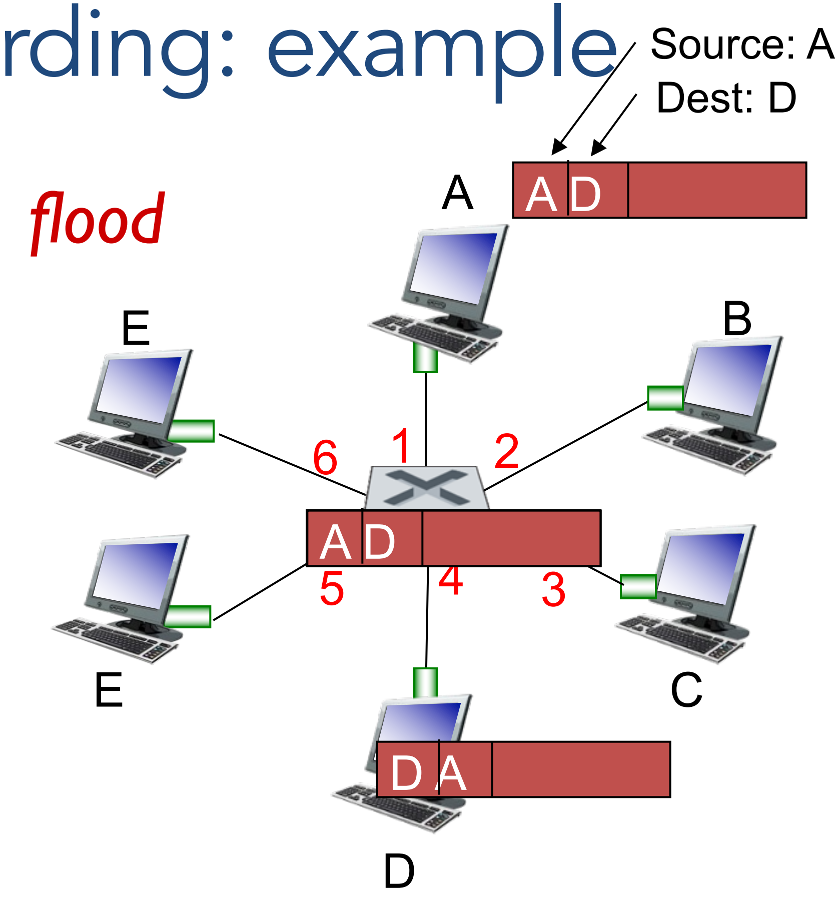


MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

Self-learning, forwarding: example

- frame destination, D, location unknown:
- destination A location known: selectively send on just one link



MAC addr	interface	TTL
A	*1*	*60*
D	*4*	*60*

switch table (initially empty)

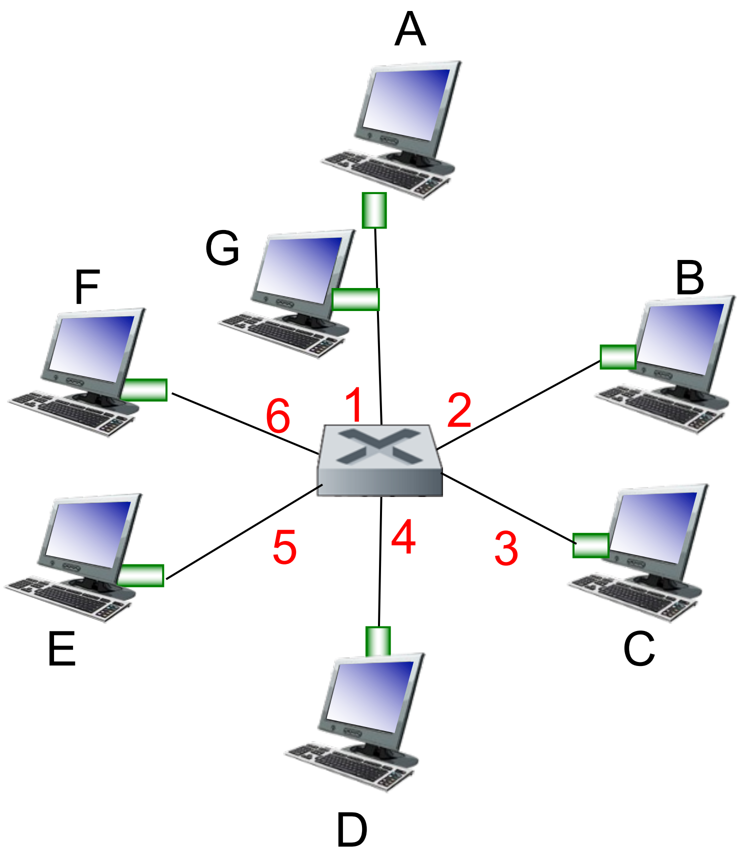
Suppose the switch receives a packet from A to G. (Assume it knows what interface both A and G are on.) It should…

A. Flood the packet

B. Throw the packet away

C. Send the packet out on interface 1

D. Do something else



Switch: frame filtering/forwarding

when frame received at switch:

1. record incoming link, MAC address of sending host
2. index switch table using MAC destination address
3. if entry found for destination:

    if destination on segment from which frame arrived:
        drop frame

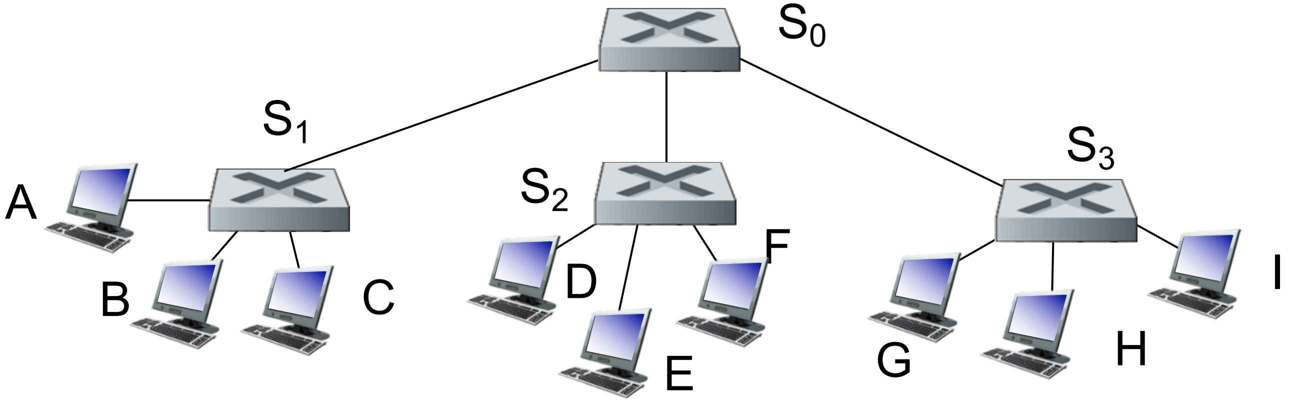
    else:

        forward frame on interface indicated by entry

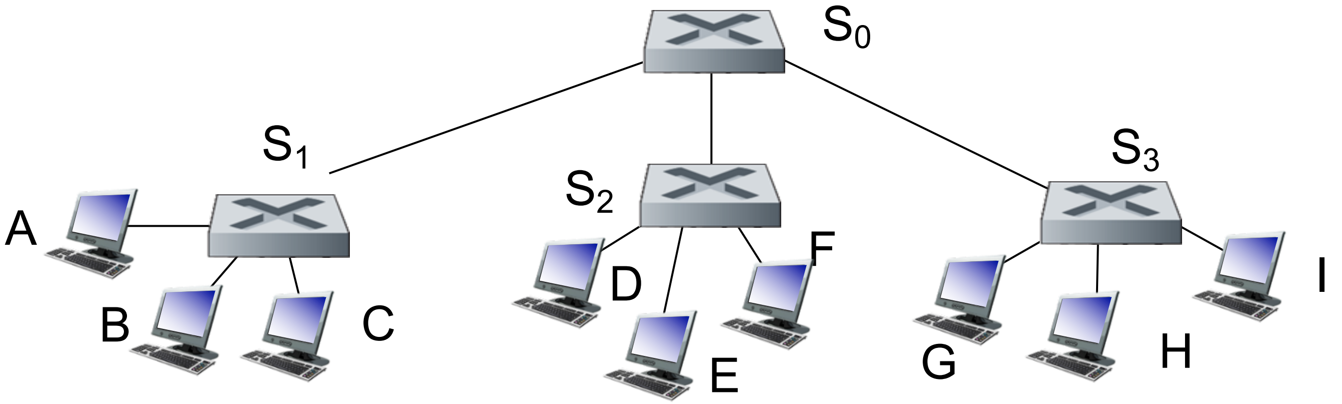
else: flood: forward on all interfaces except arriving
interface

Interconnecting switches

- Switches often connected to form trees.



Sending from A to G - how does S1 know to forward frame destined to G via S0 and S3?



A. A network administrator will need to configure this.

B. S_1 will automatically learn the entire path.

C. S_1 will learn to send packets to G on the interface that leads to S_0.

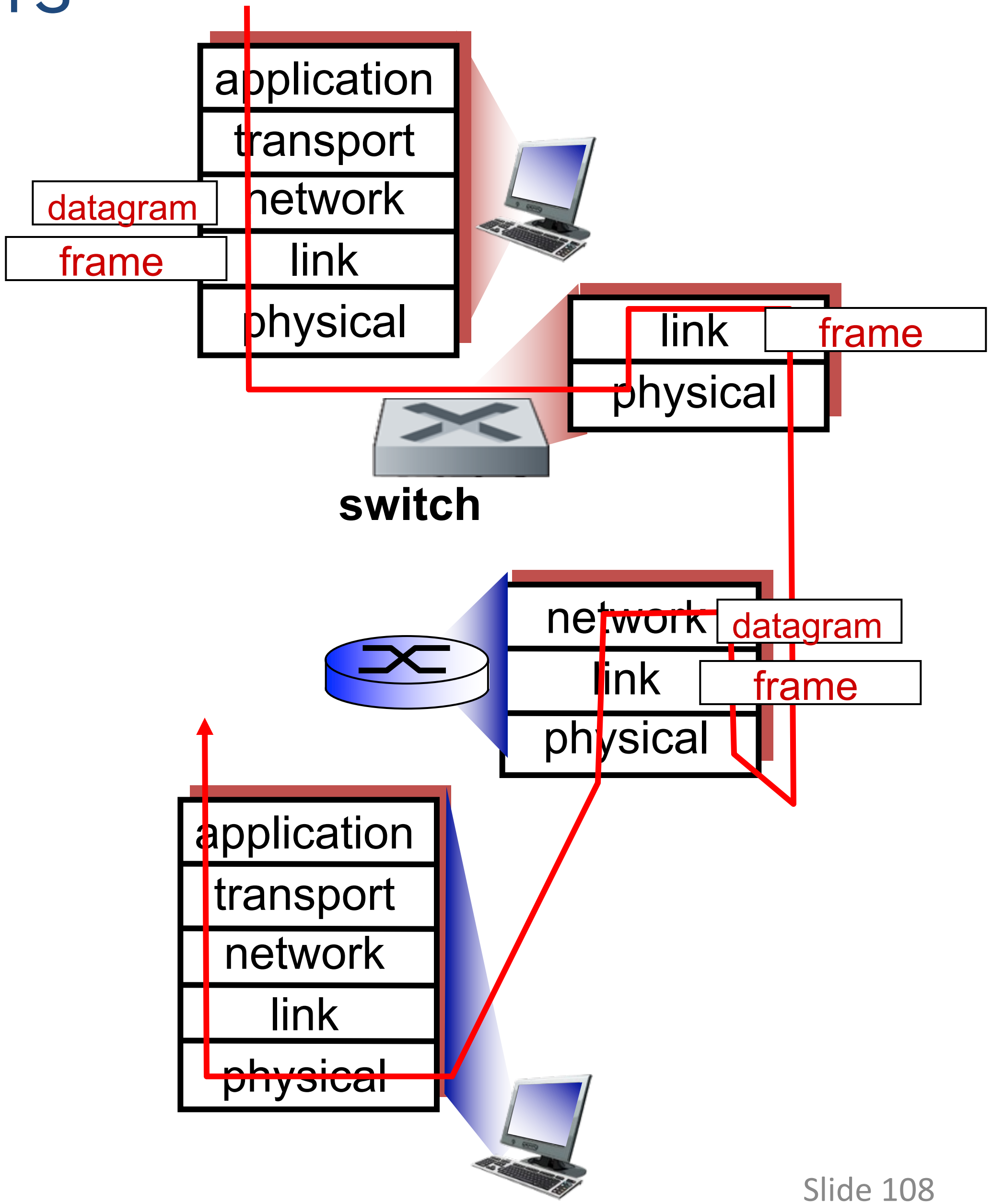
Switches vs. routers

both are store-and-forward:

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

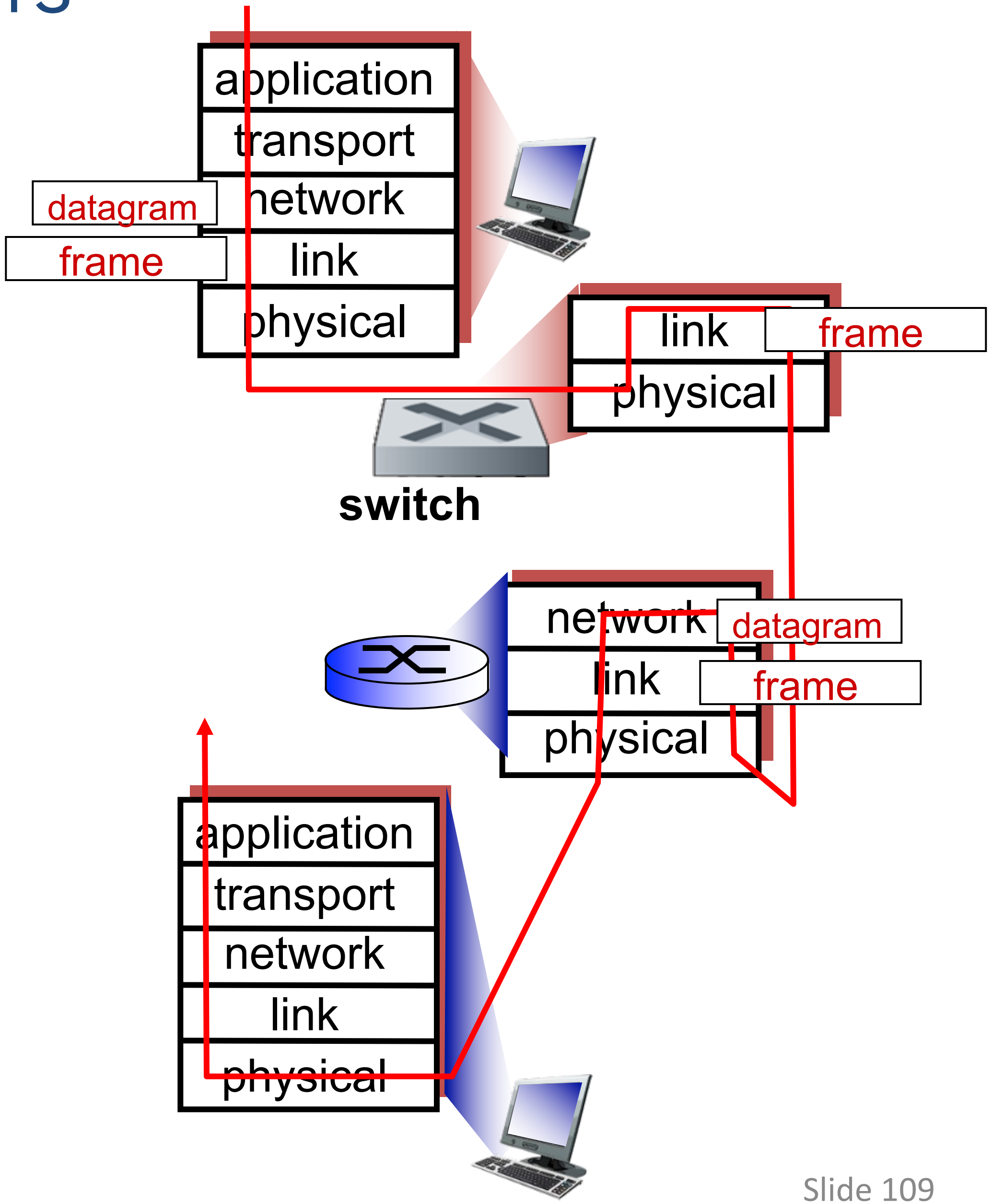
both have forwarding tables:

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



Switches vs. routers

Switches do NOT run a complex coordination protocol like routing.



both have forwarding tables:

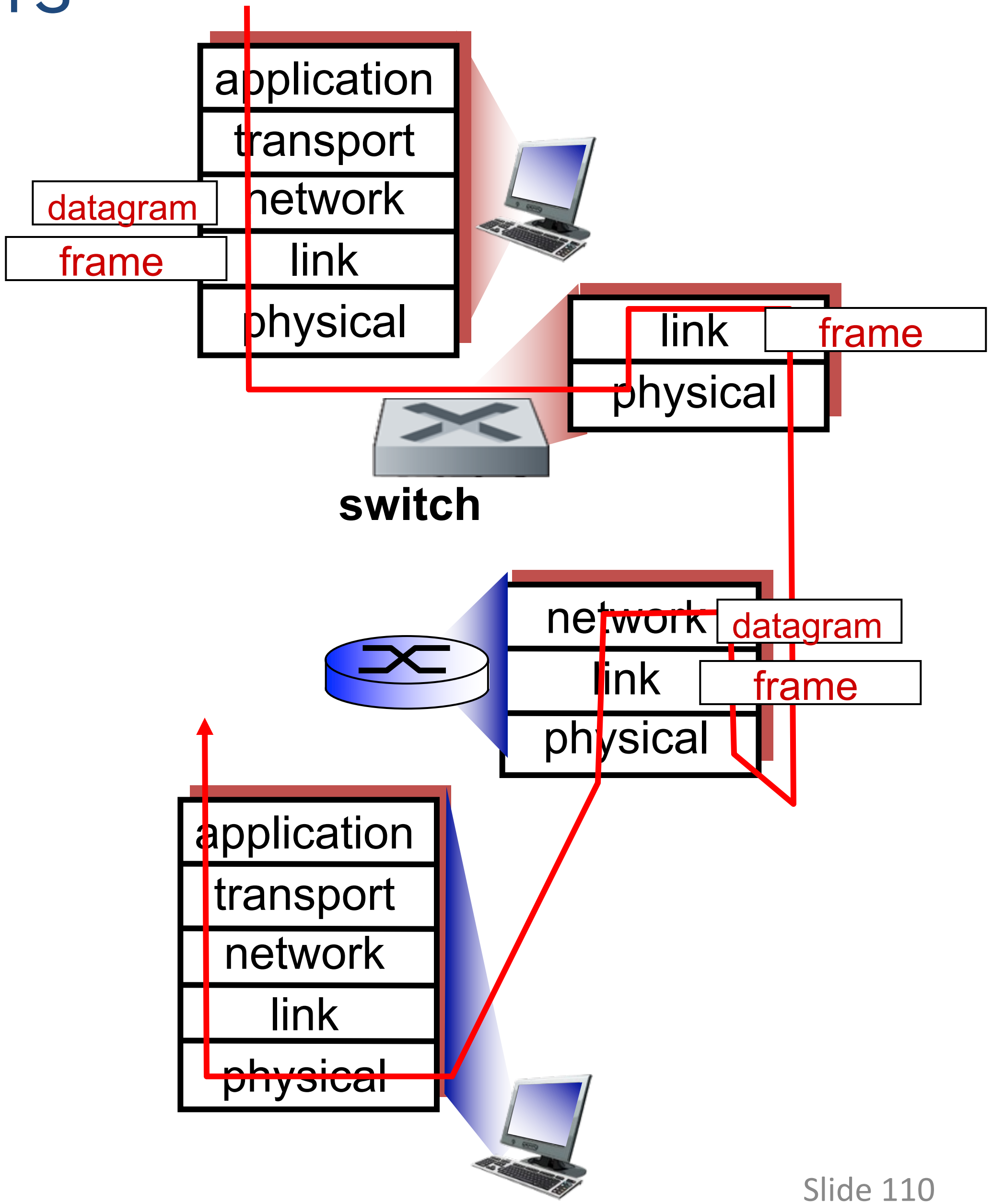
- *routers:* compute tables using routing algorithms, IP addresses
- *switches:* learn forwarding table using flooding, learning, MAC addresses

Switches vs. routers

You do NOT address frames directly to a switch (unless you're configuring it).

both have forwarding tables:

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



Summary

- LAN address: flat (vs. hierarchical IP)

- Many potential topologies:
  - Bus: shared wire, star (hub)
  - Switched: star, tree

- Switches learn who is connected, selectively forward toward destination

The Internet protocol stack

Application: the application (e.g., the Web, Email)

Transport: end-to-end connections, reliability

Network: routing

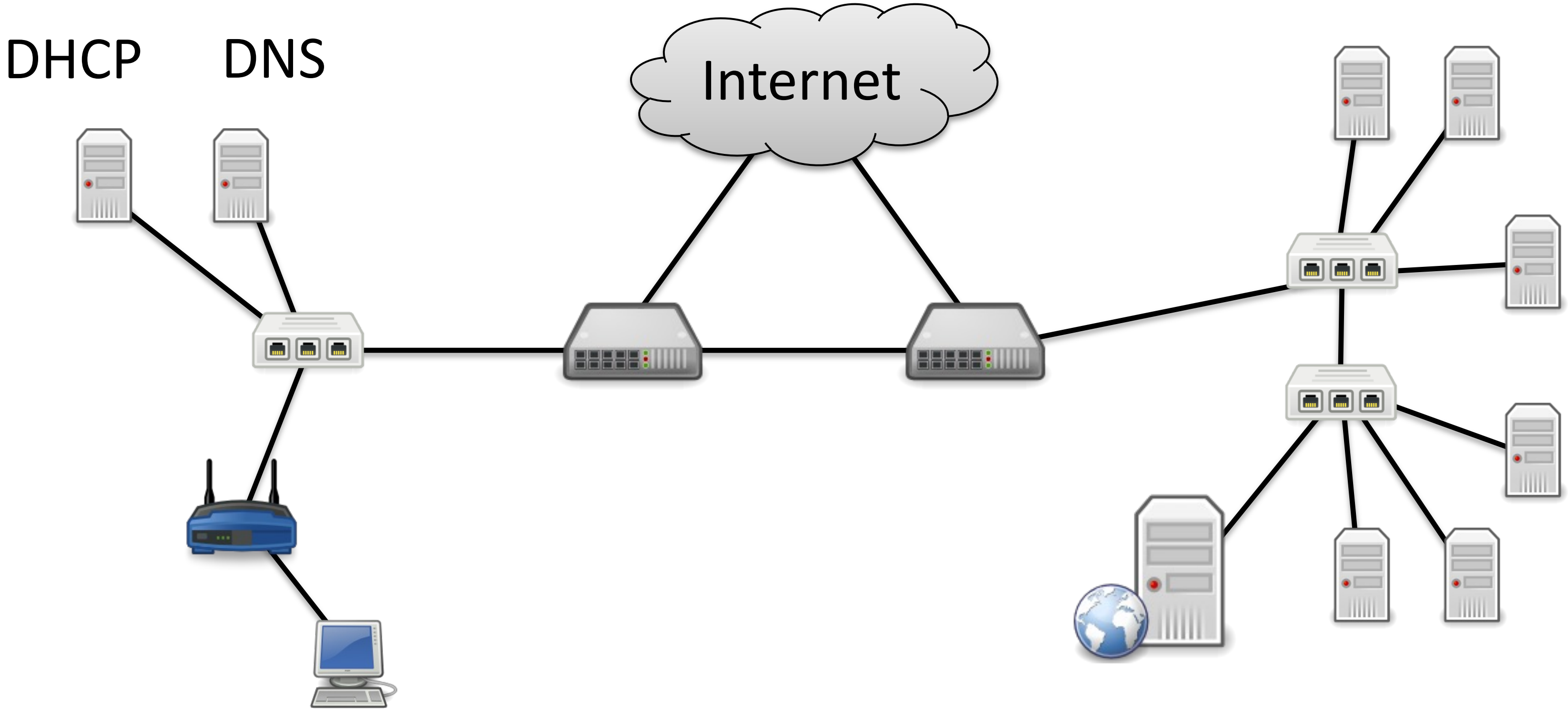
Link (data-link): framing, error detection

Physical: 1's and 0's/bits across a medium (copper, the air, fiber)

Putting it all together…

- What happens when a user shows up to a new network and wants to access a web site?

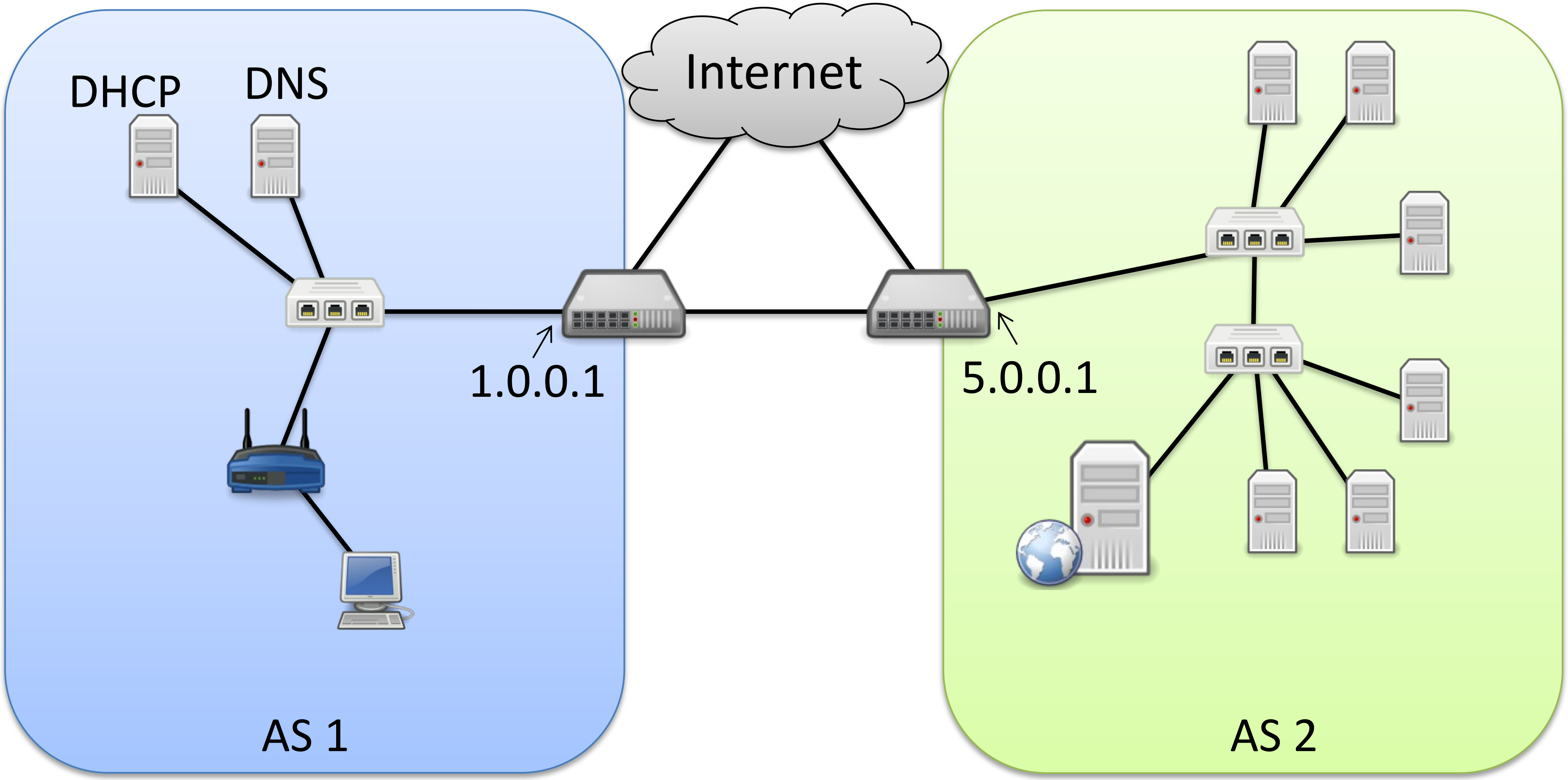
Scenario



Scenario

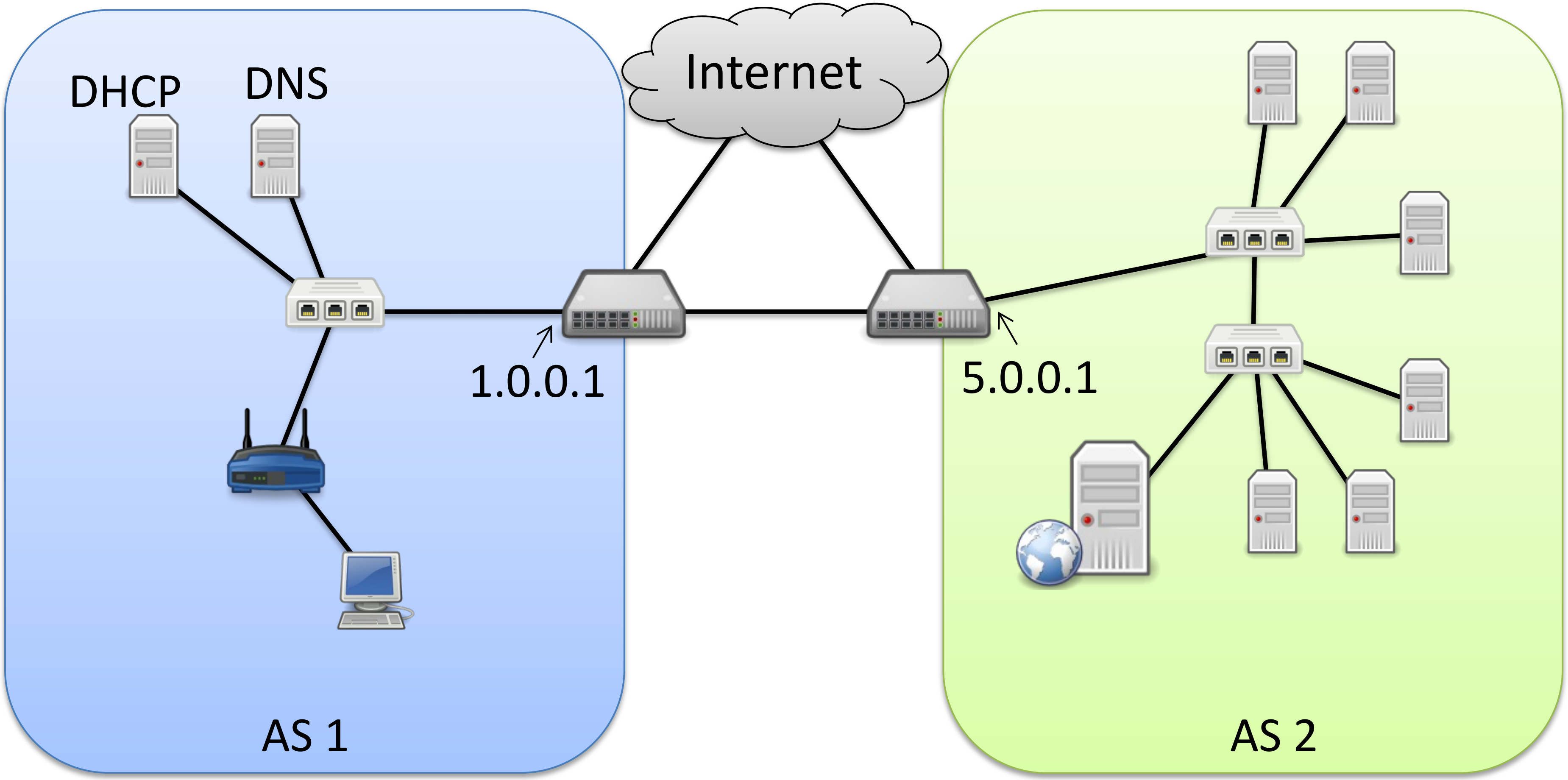
Network: 1.0.0.0/24
24 bits: network
8 bits: host

Network: 5.0.0.0/16
16 bits: network
16 bits: host



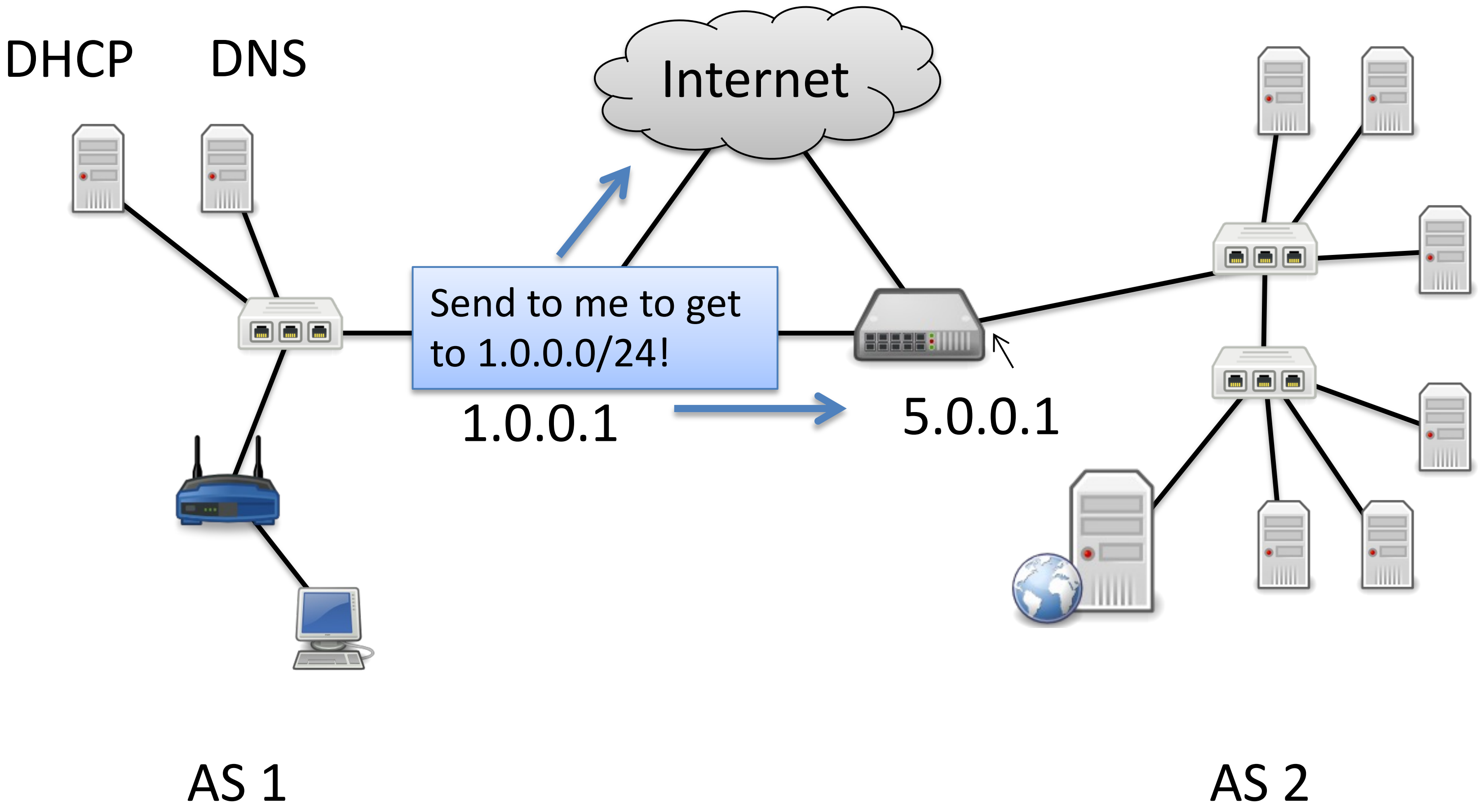
Network: 1.0.0.0/24
24 bits: network
8 bits: host:
- all host addresses are 1.0.0.*
- 256 possible addresses

Network: 5.0.0.0/16
16 bits: network
16 bits: host:
- all host addresses are 5.0.*
- 65,536 possible addresses!



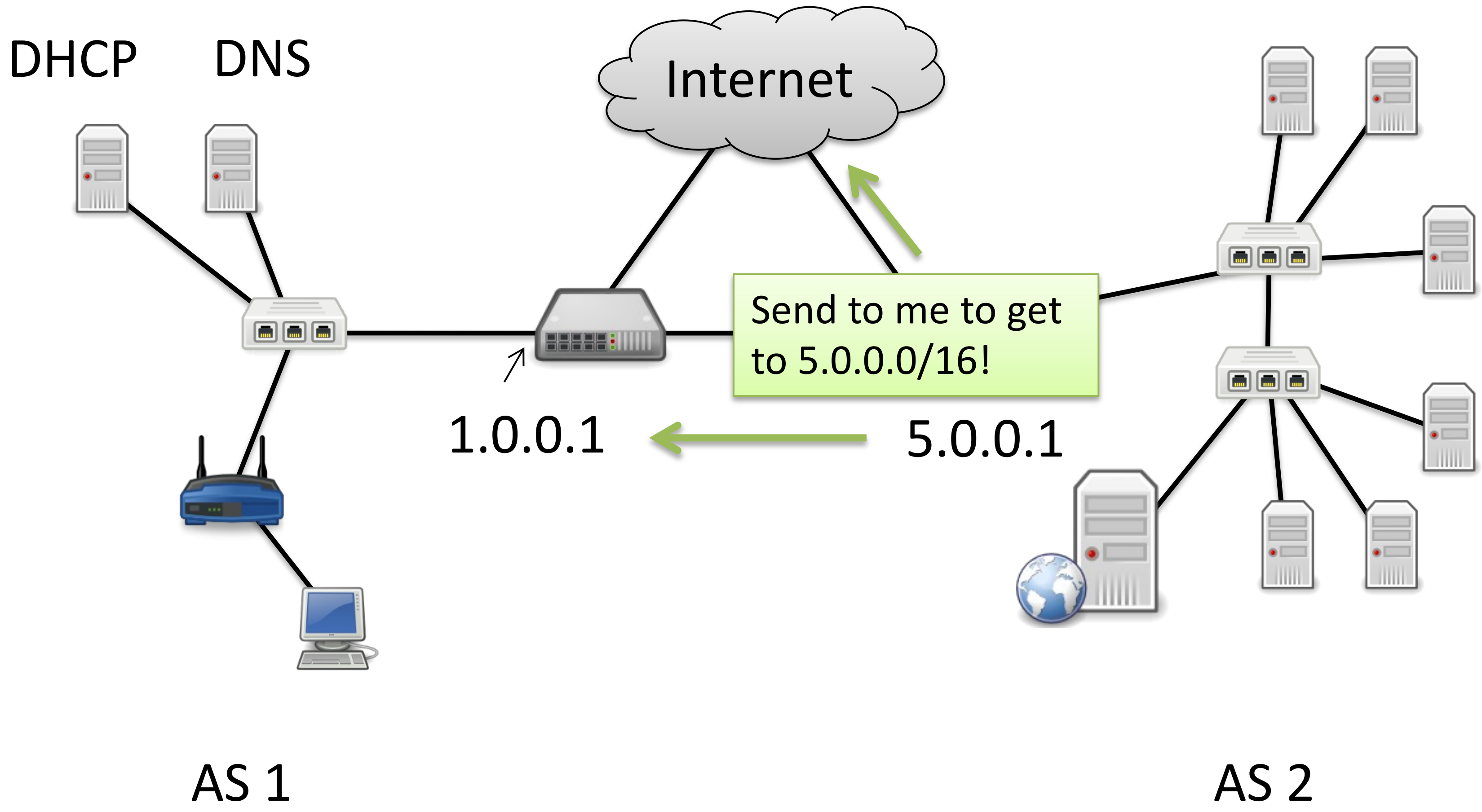
Step 0: Routing Protocol

Before anyone starts sending data, we'll assume the routers have run a routing protocol (BGP) to learn about each other.



Step 0: Routing Protocol

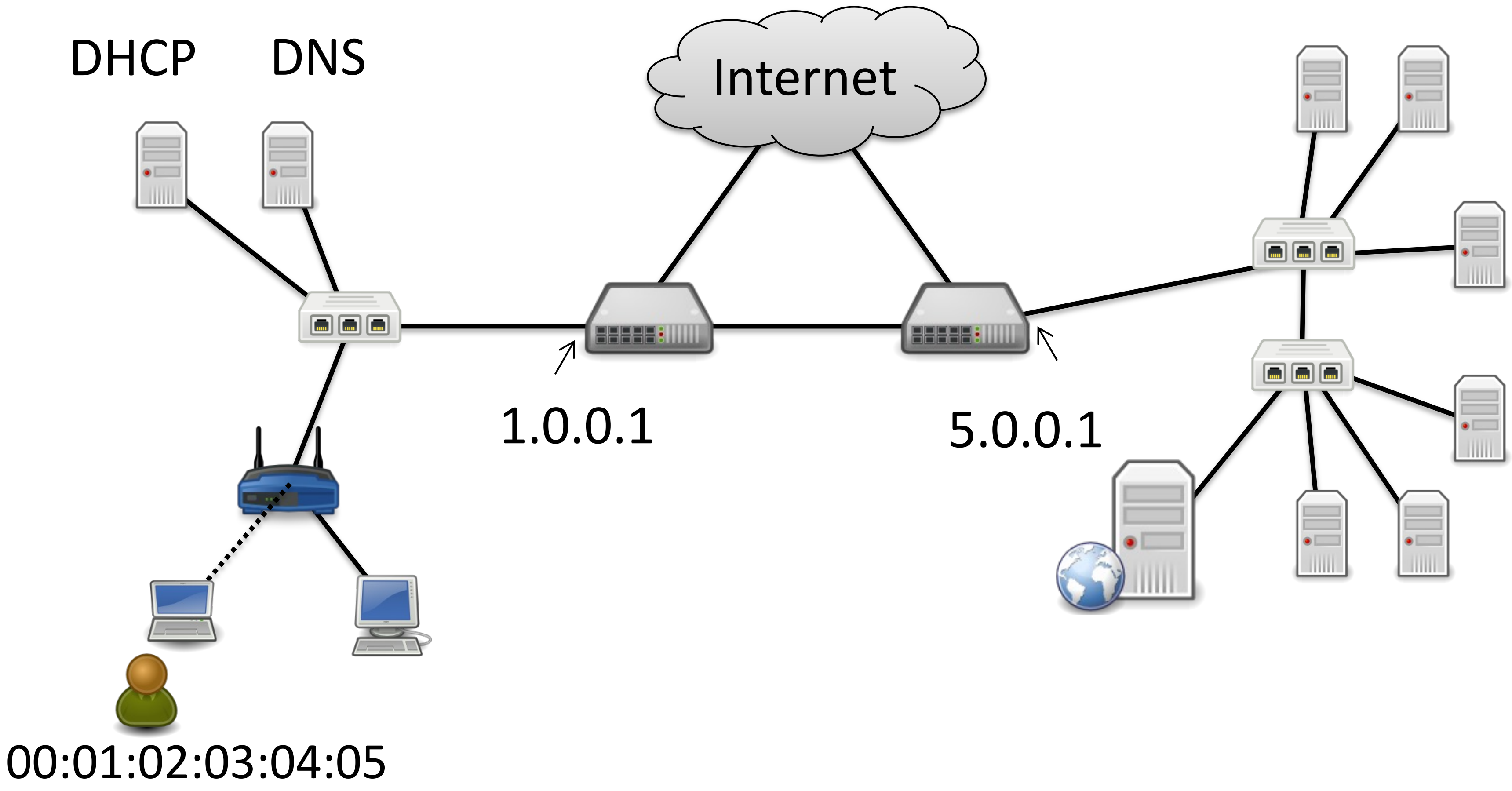
Before anyone starts sending data, we'll assume the routers have run a routing protocol (BGP) to learn about each other.



Step 1: User Joins the Network

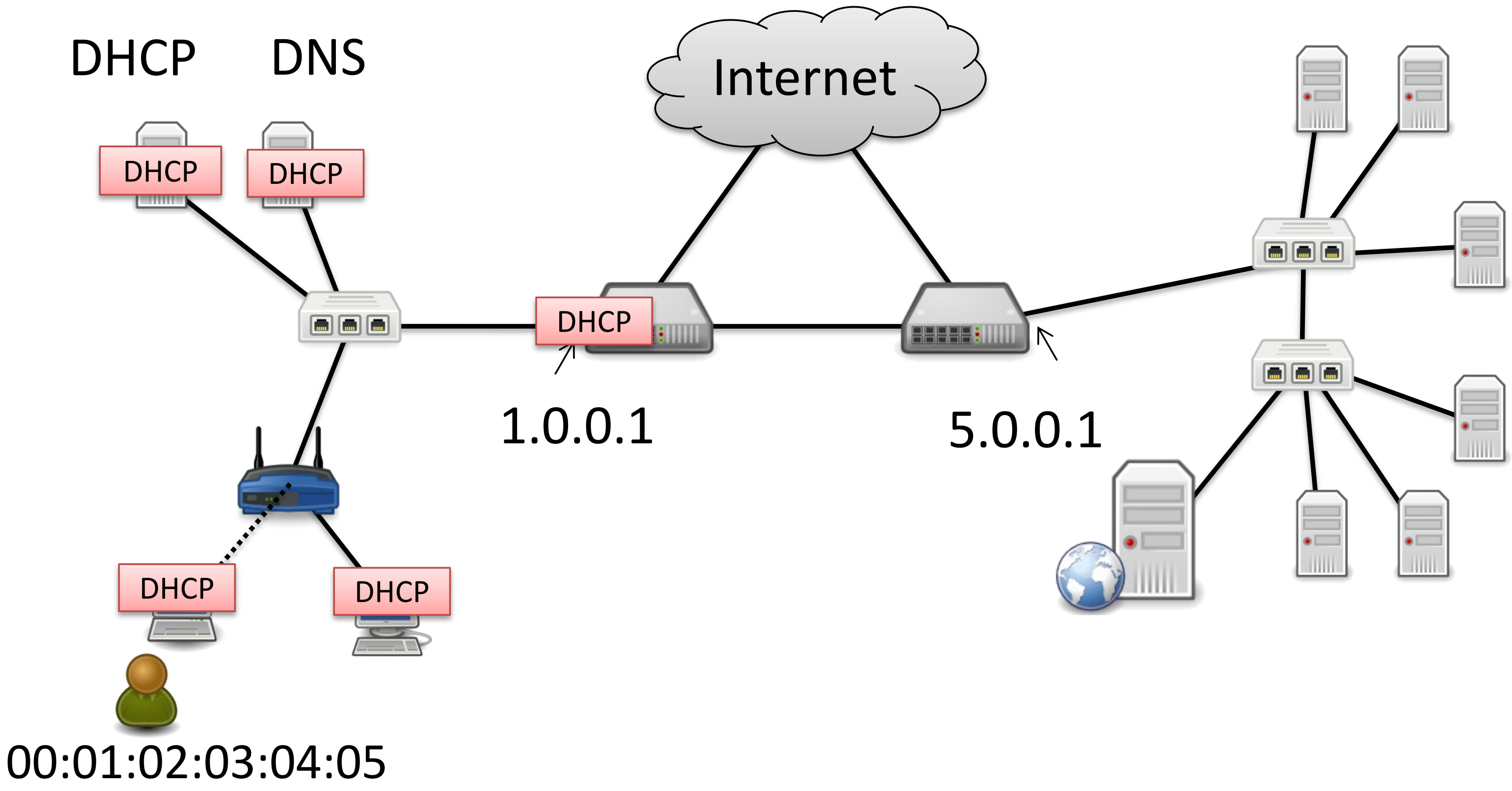
User arrives and needs an IP address.

They bring MAC address with them (built in to hardware).



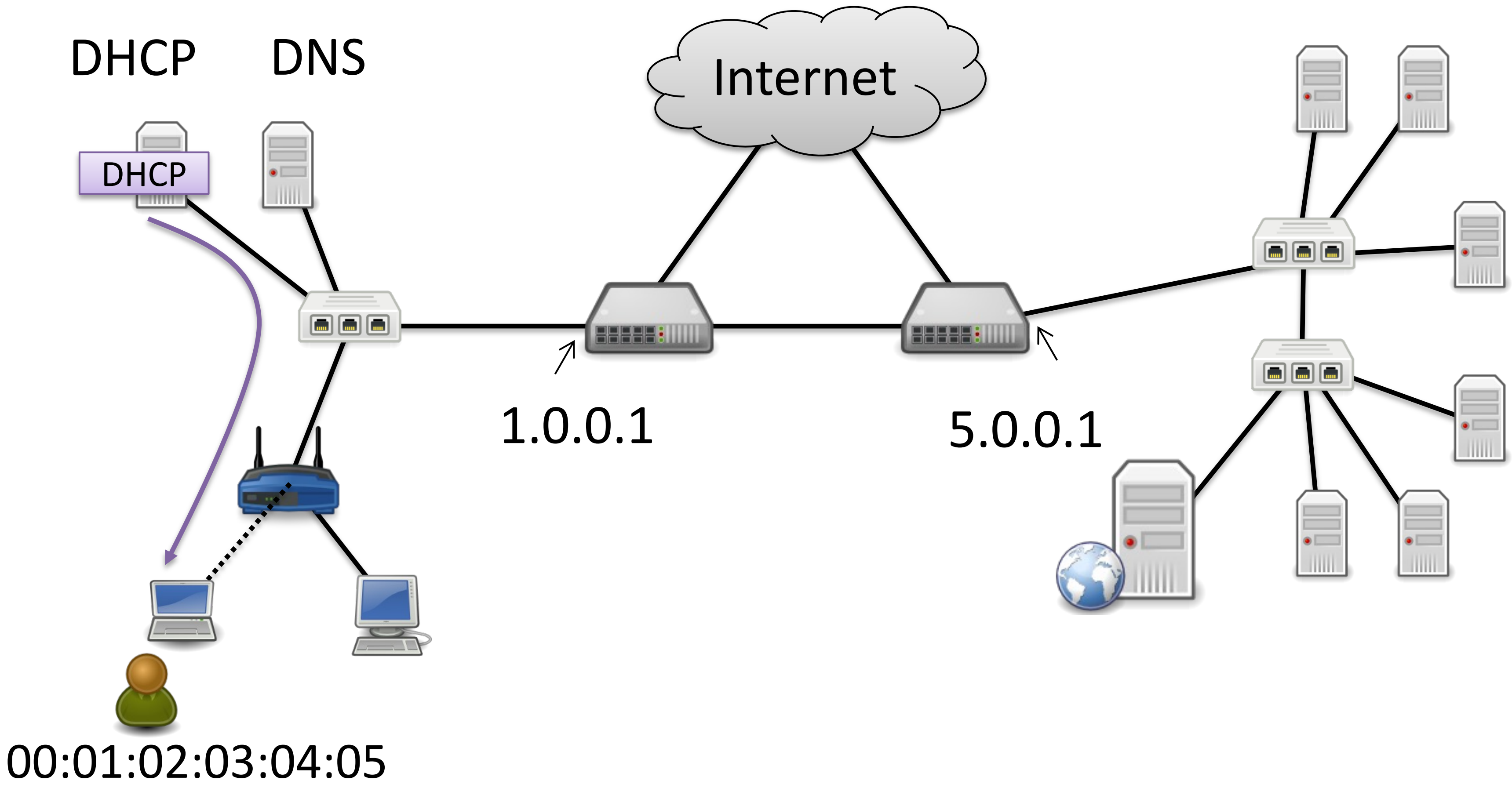
Step 1: User Joins the Network

User broadcasts DHCP DISCOVER message to acquire IP address. (Alternative, they manually enter IP config details.)



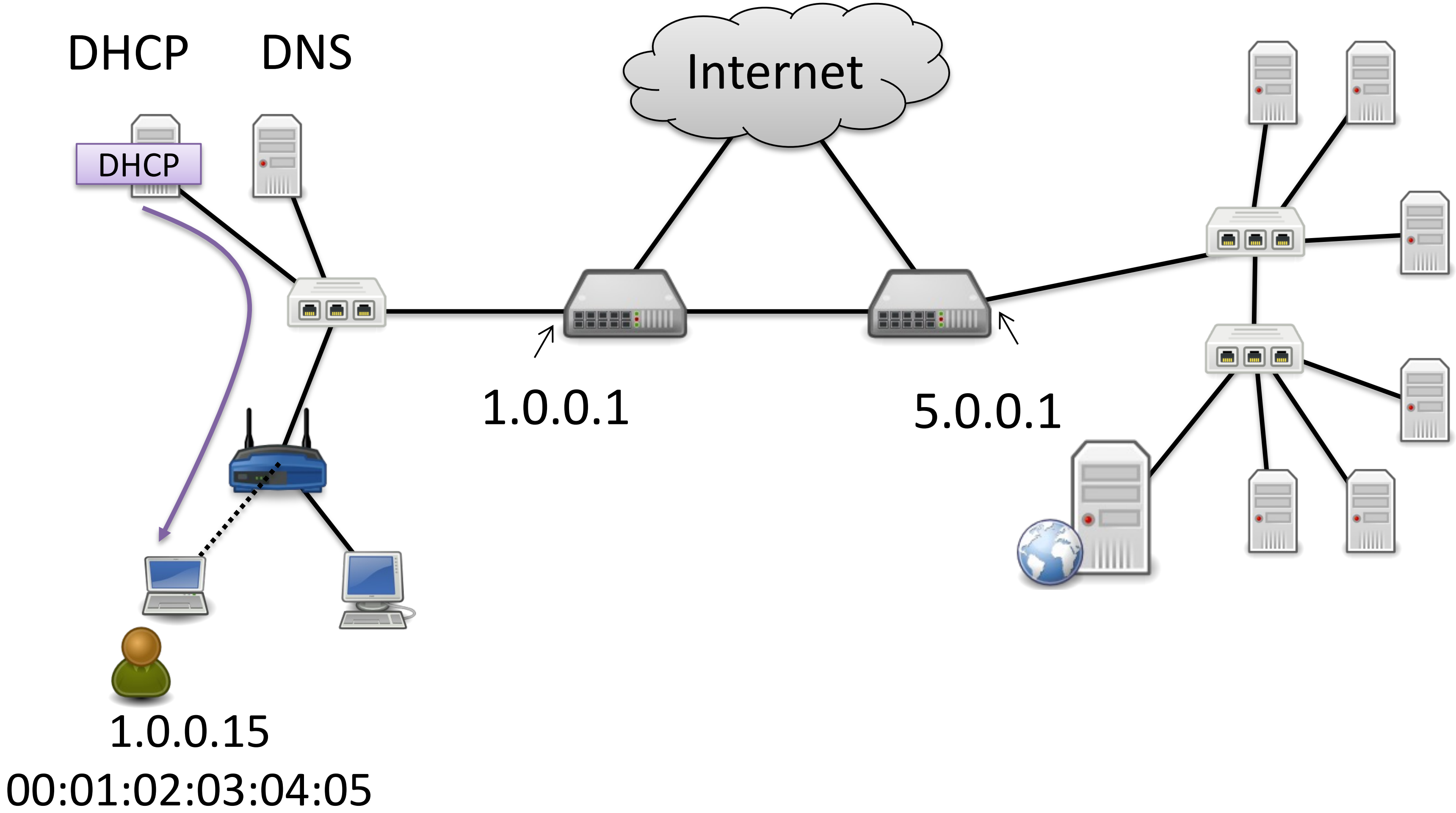
Step 1: User Joins the Network

DHCP responds with: IP address (1.0.0.15), subnet mask (255.255.255.0), gateway (1.0.0.1), and DNS server (1.0.0.2).



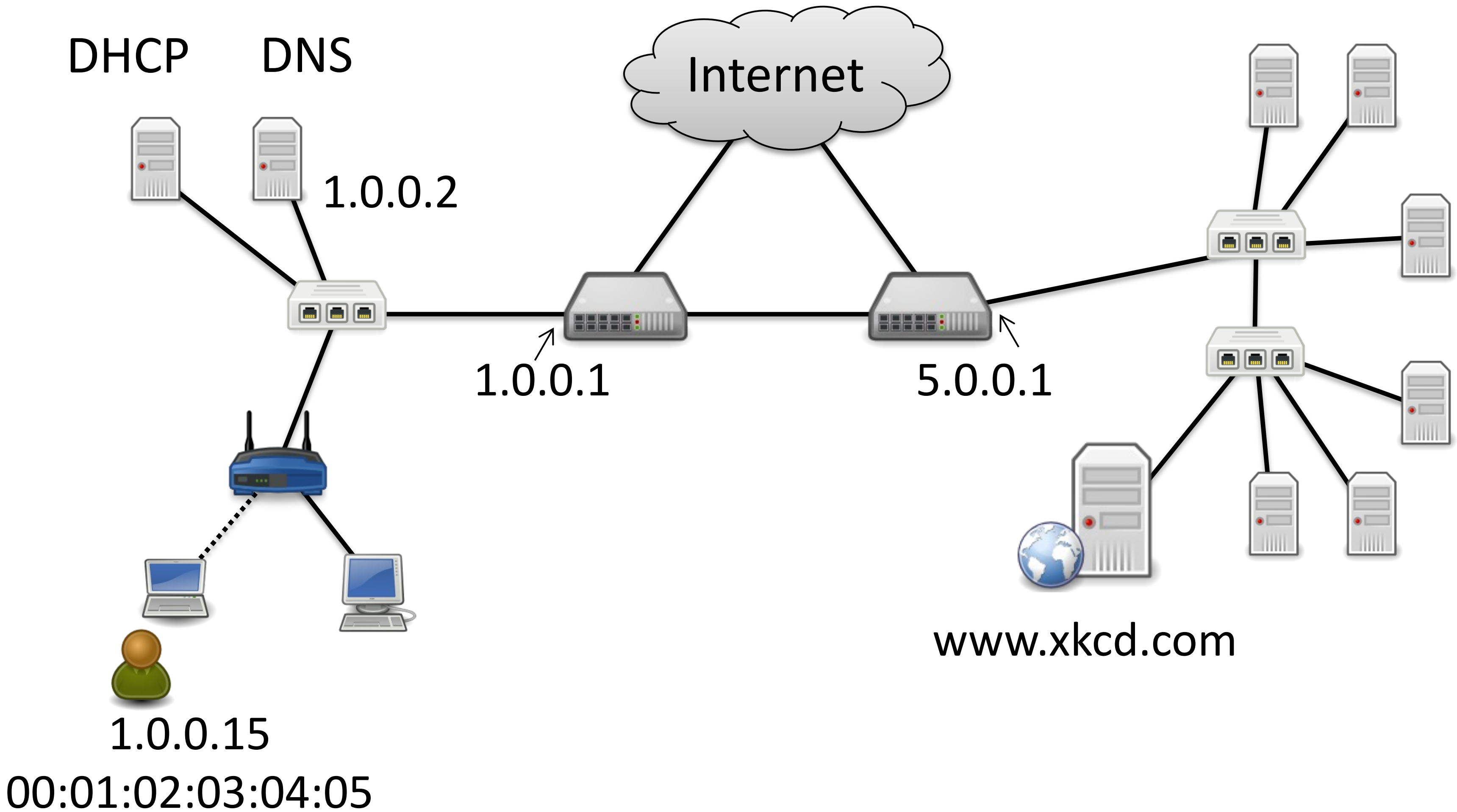
Step 1: User Joins the Network

DHCP responds with: IP address (1.0.0.15), subnet mask (255.255.255.0), gateway (1.0.0.1), and DNS server (1.0.0.2).



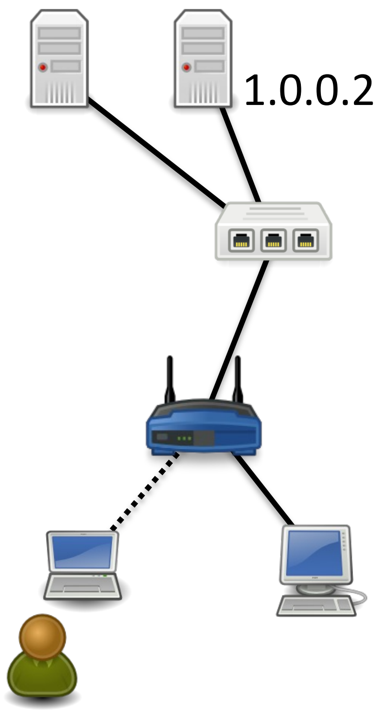
Step 2: User Resolves Name

Suppose user tries to access website: www.xkcd.com
Must resolve name using DNS. Query local resolver.



Step 2: User Resolves Name

User's PC must answer: is the DNS resolver (1.0.0.2) I was given by DHCP on my subnet? (Local vs. Internet)



my address → 1.0.0.15:

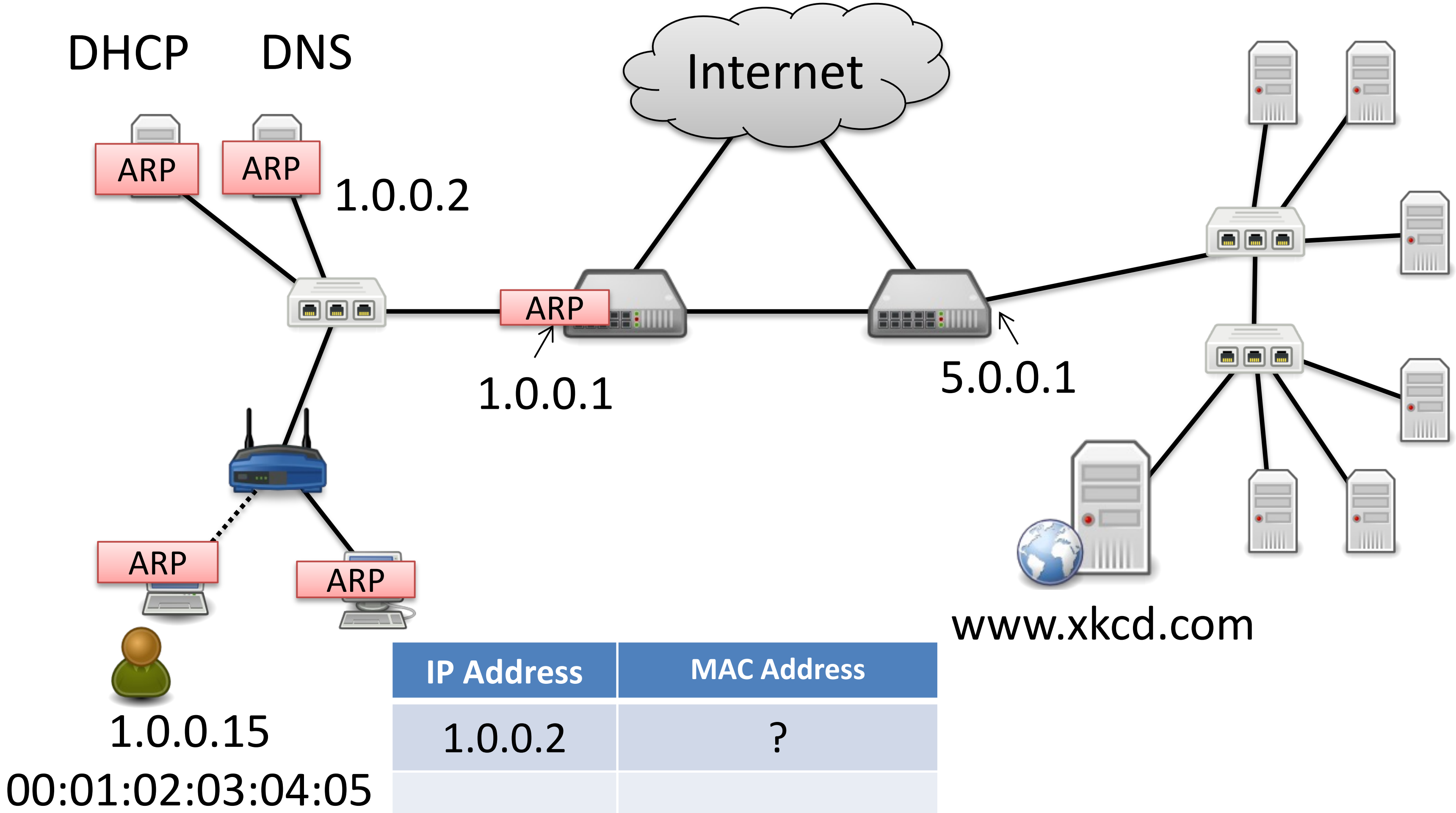
subnet mask → 255.255.255.0:

1.0.0.15:	00000001	00000000	00000000	00001111
255.255.255.0:	11111111	11111111	11111111	00000000
ANDed together: my network prefix	00000001	00000000	00000000	
target address 1.0.0.2	00000001	00000000	00000000	00000010

Match! It's local. Send directly, no need to go through Internet gateway (router).

Step 2: User Resolves Name

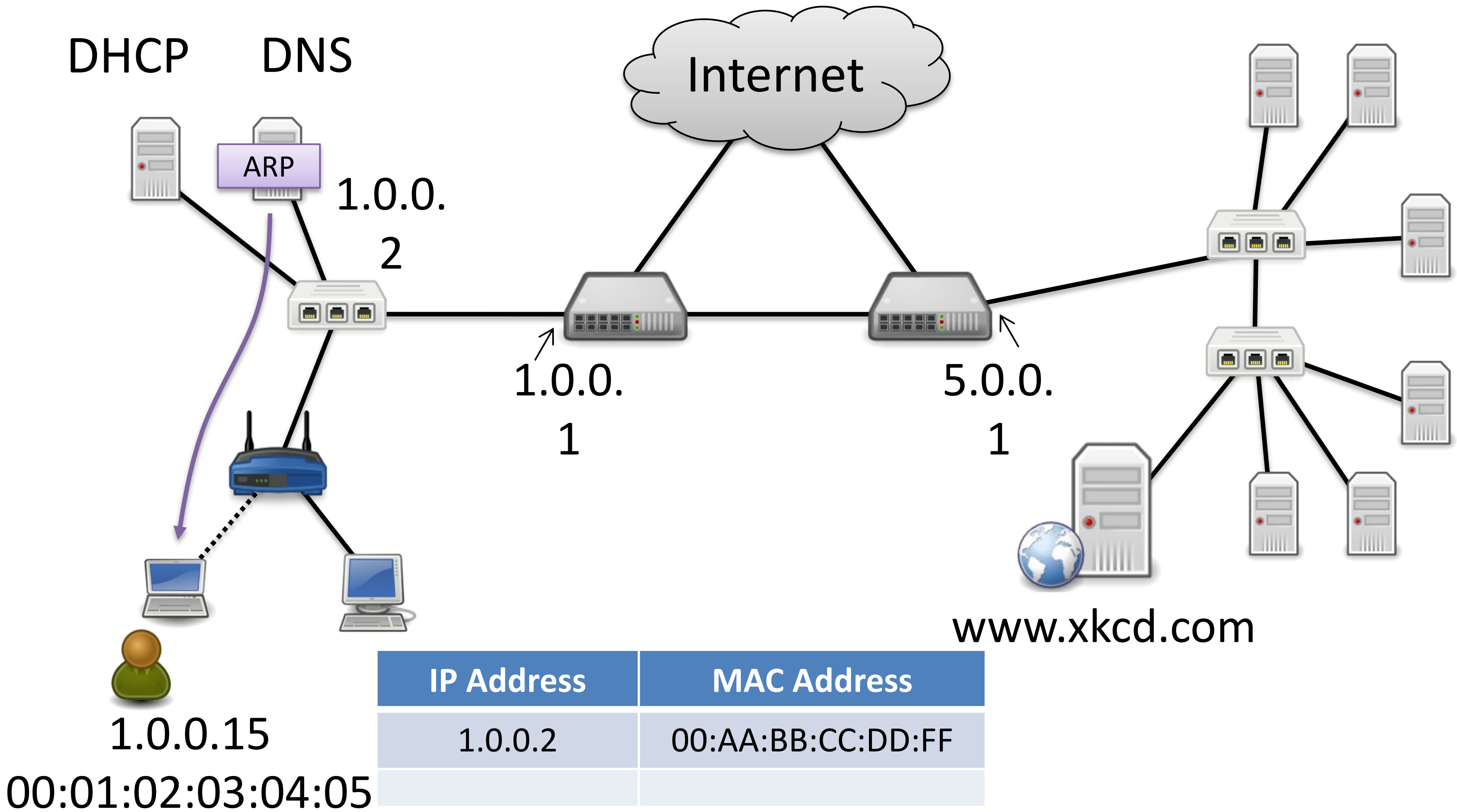
User's PC does NOT know DNS server's MAC address!
Broadcast ARP request looking for 1.0.0.2!



IP Address	MAC Address
1.0.0.2	?

Step 2: User Resolves Name

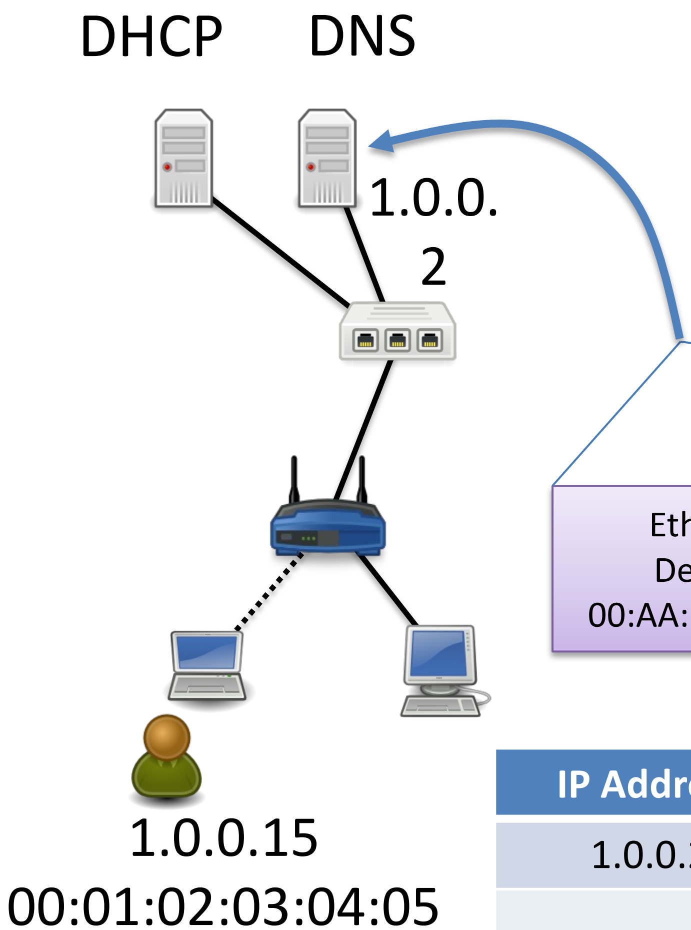
DNS server responds with MAC address.

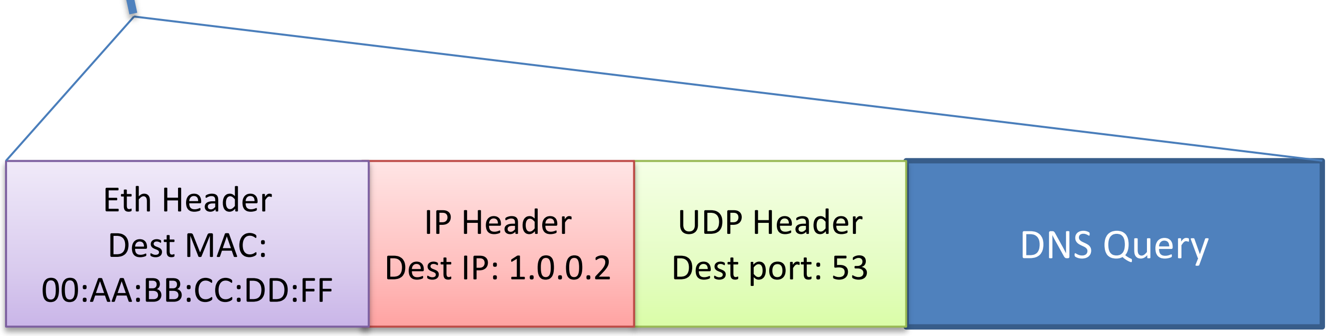


IP Address	MAC Address
1.0.0.2	00:AA:BB:CC:DD:FF

Step 2: User Resolves Name

User queries local DNS resolver for www.xkcd.com.
Resolver runs necessary queries (root, TLD, etc.)

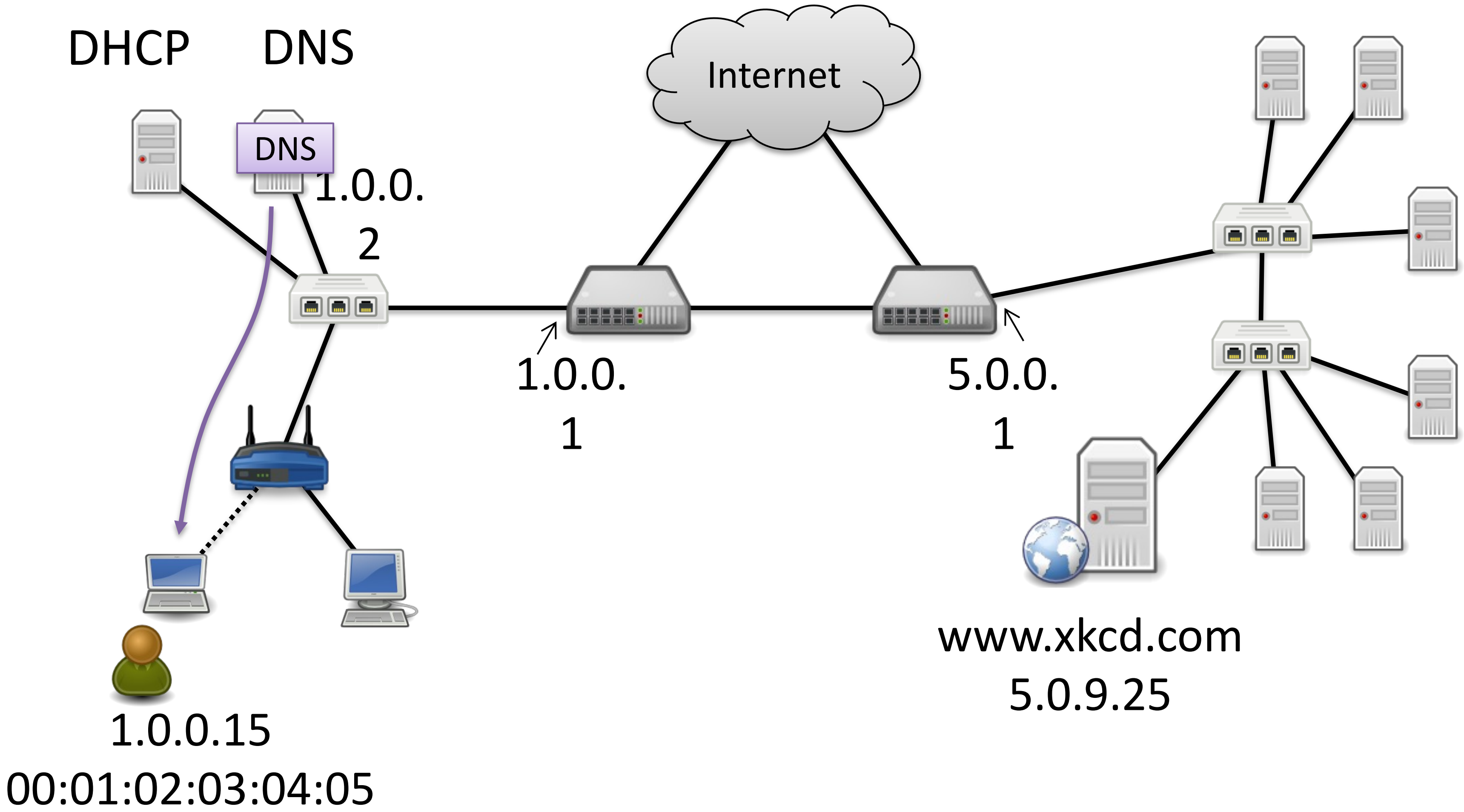




IP Address	MAC Address
1.0.0.2	00:AA:BB:CC:DD:FF

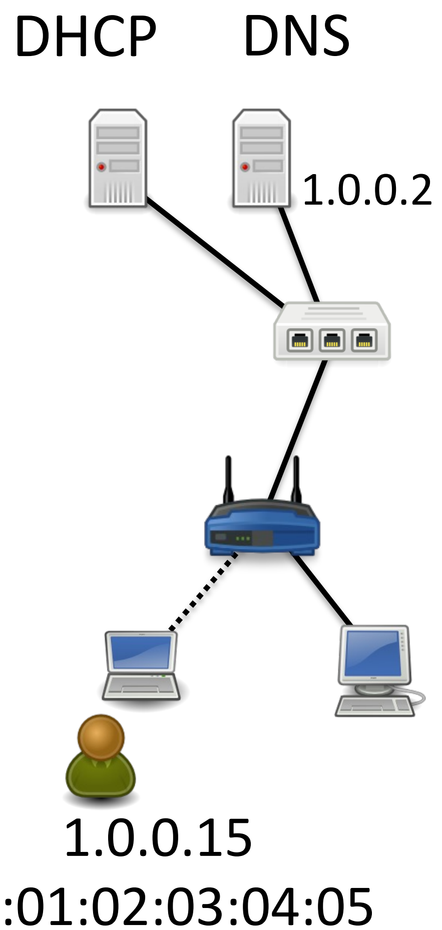
Step 2: User Resolves Name

DNS reply says that www.xkcd.com is 5.0.9.25.



Step 3: Establish a TCP Connection

User's PC must answer: is the destination (5.0.9.25) on my subnet? (Local vs. Internet)



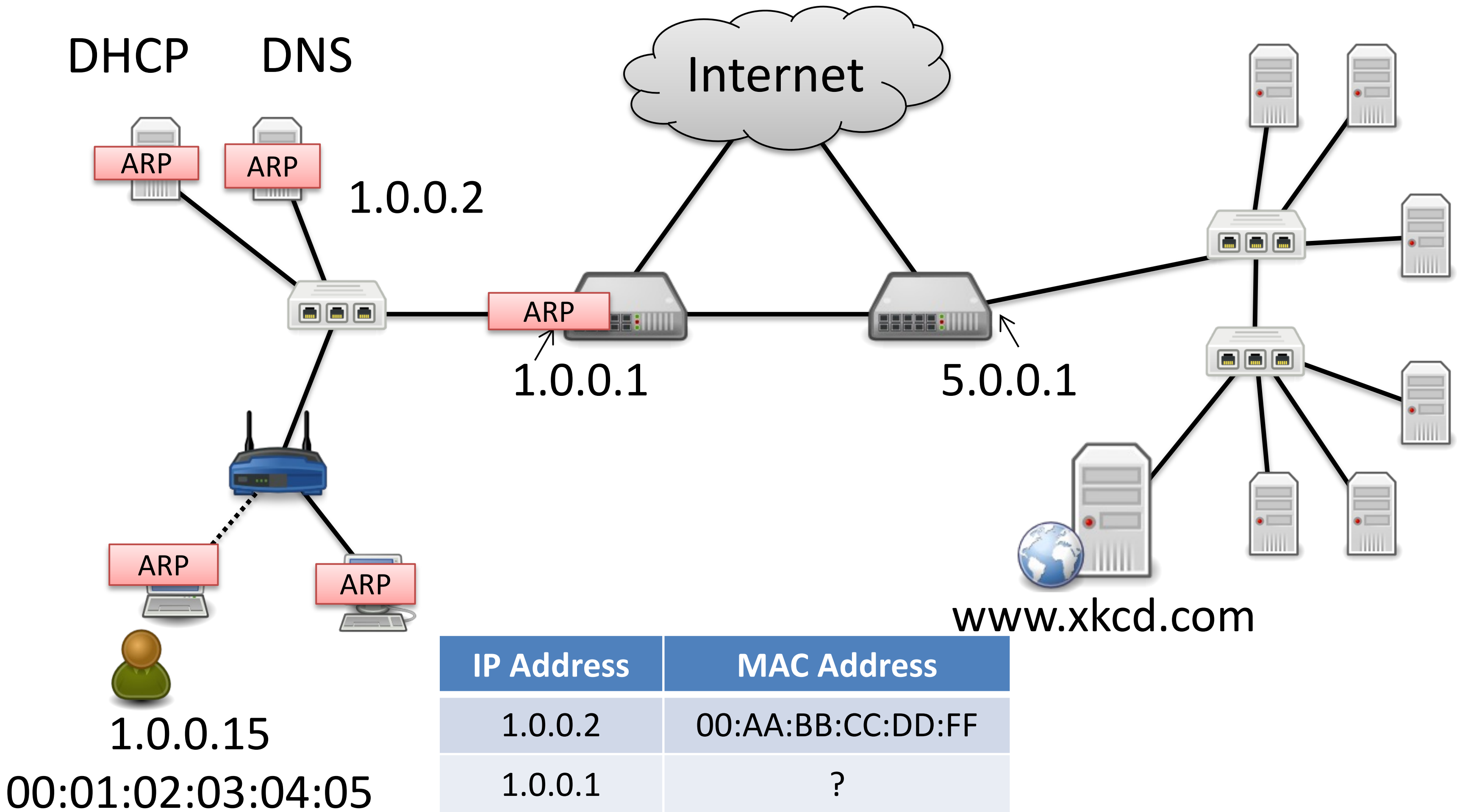
my address → 1.0.0.15; subnet mask → 255.255.255.0

1.0.0.15:	00000001	00000000	00000000	00001111
255.255.255.0:	11111111	11111111	11111111	00000000
ANDed tog. ether: my network prefix	00000001	00000000	00000000	
target address 5.0.9.25	00000101	00000000	00001001	00011101

No Match! Send it to the default gateway (router that connects to the Internet) that DHCP gave us (1.0.0.1).

Step 3: Establish a TCP Connection

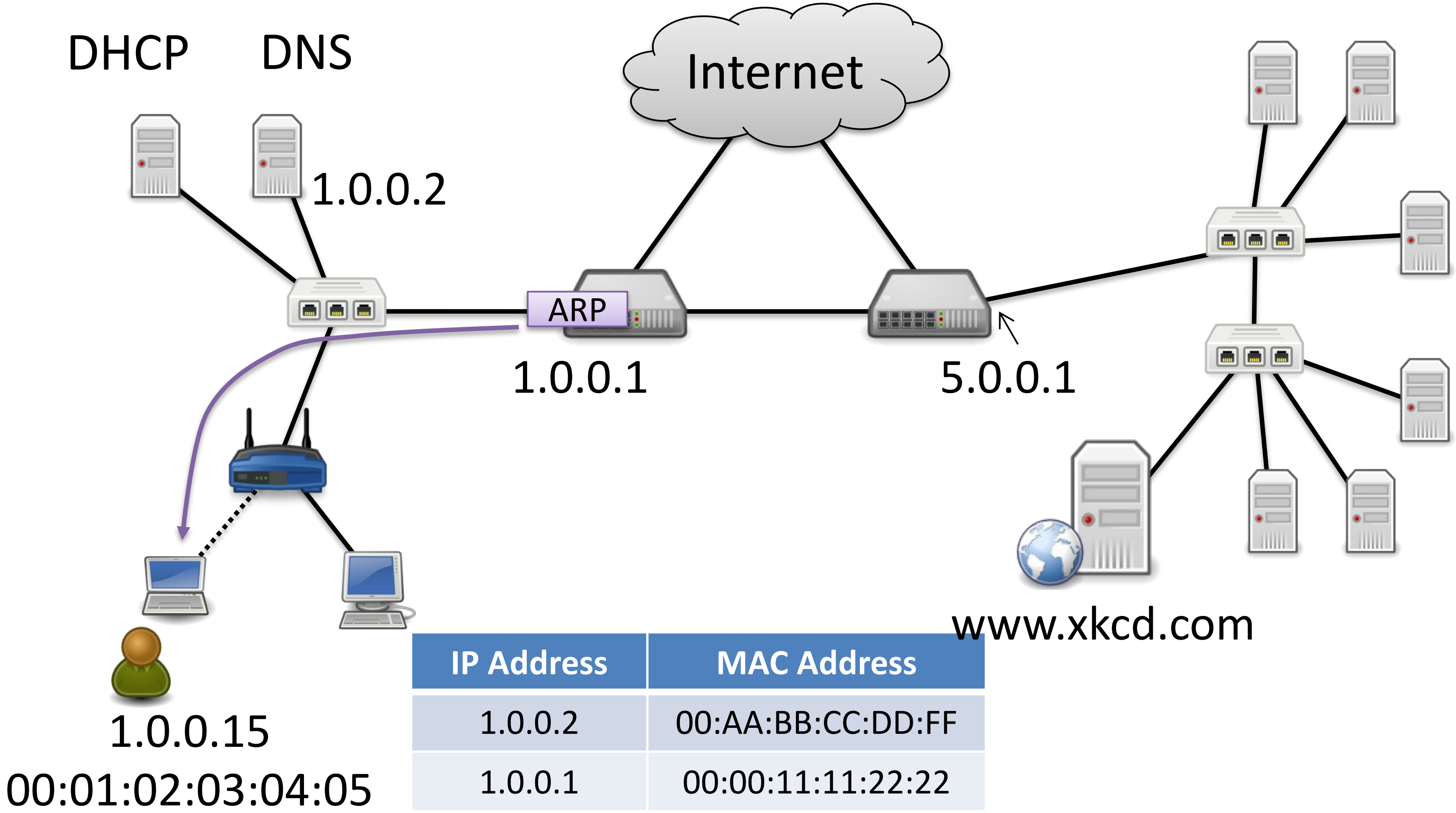
User's PC does NOT know router's MAC address!
Broadcast ARP request looking for 1.0.0.1!



IP Address	MAC Address
1.0.0.2	00:AA:BB:CC:DD:FF
1.0.0.1	?

Step 3: Establish a TCP Connection

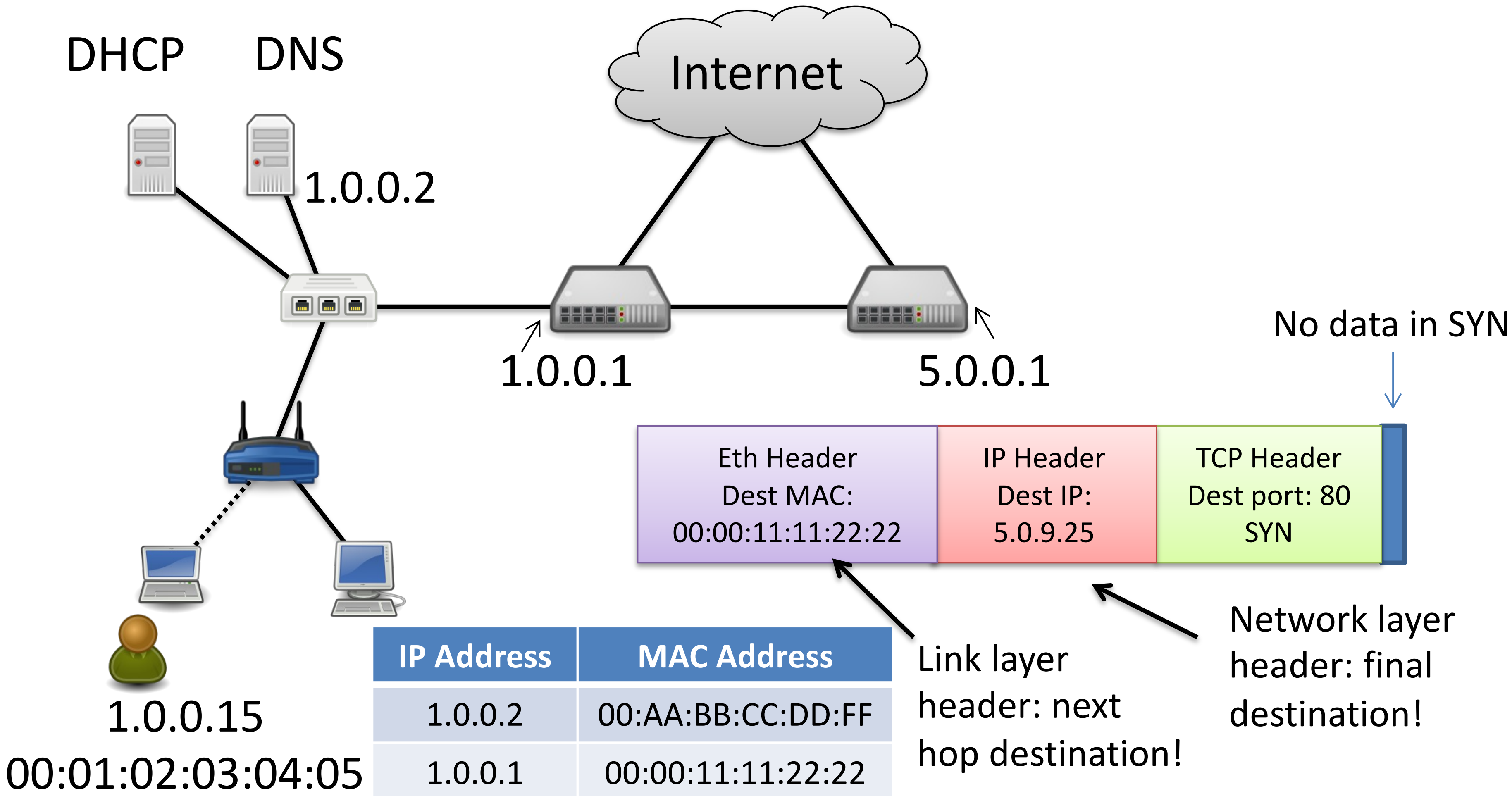
Router responds with MAC address.



IP Address	MAC Address
1.0.0.2	00:AA:BB:CC:DD:FF
1.0.0.1	00:00:11:11:22:22

Step 3: Establish a TCP Connection

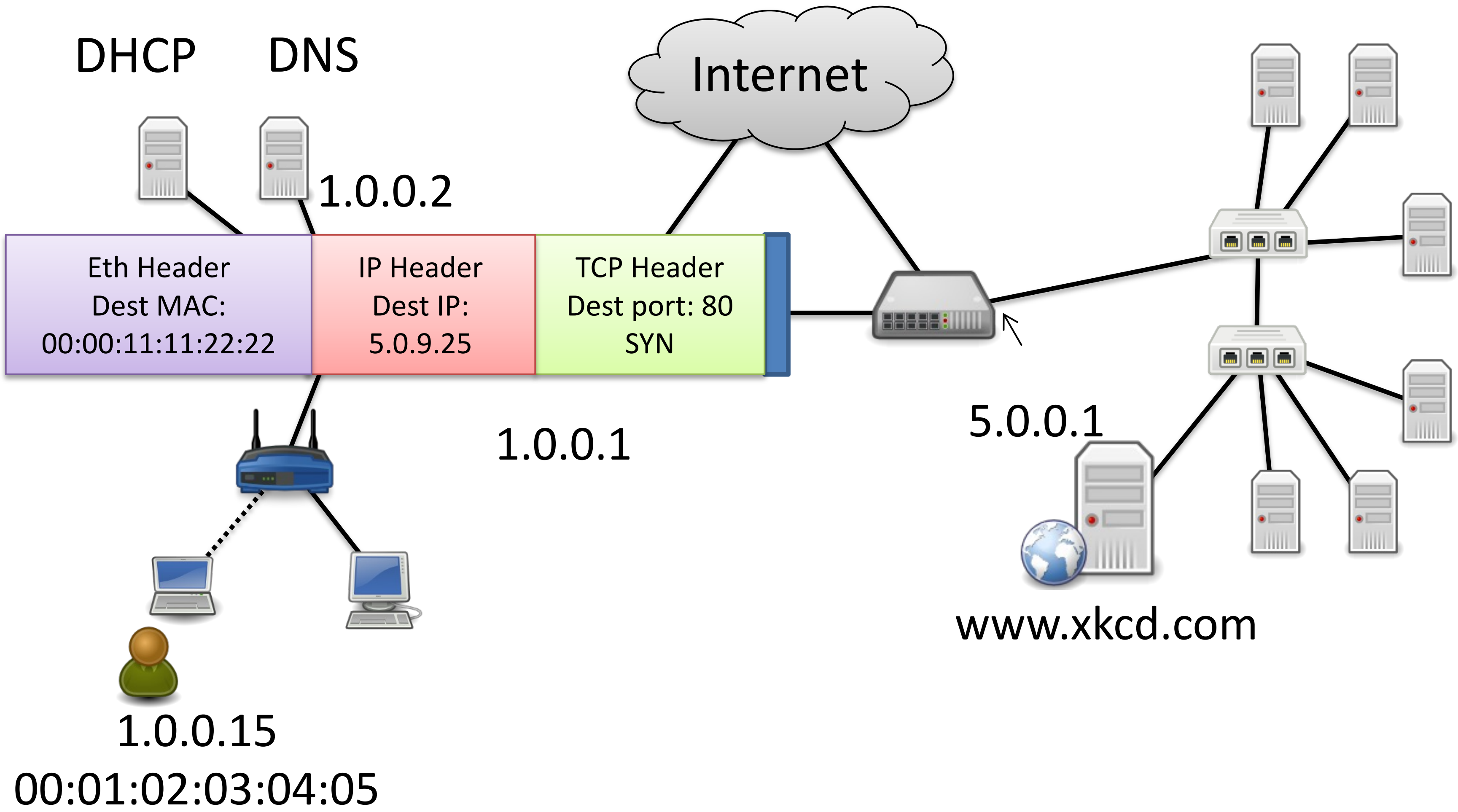
Send TCP SYN to the destination, start 3-way handshake.



IP Address	MAC Address
1.0.0.2	00:AA:BB:CC:DD:FF
1.0.0.1	00:00:11:11:22:22

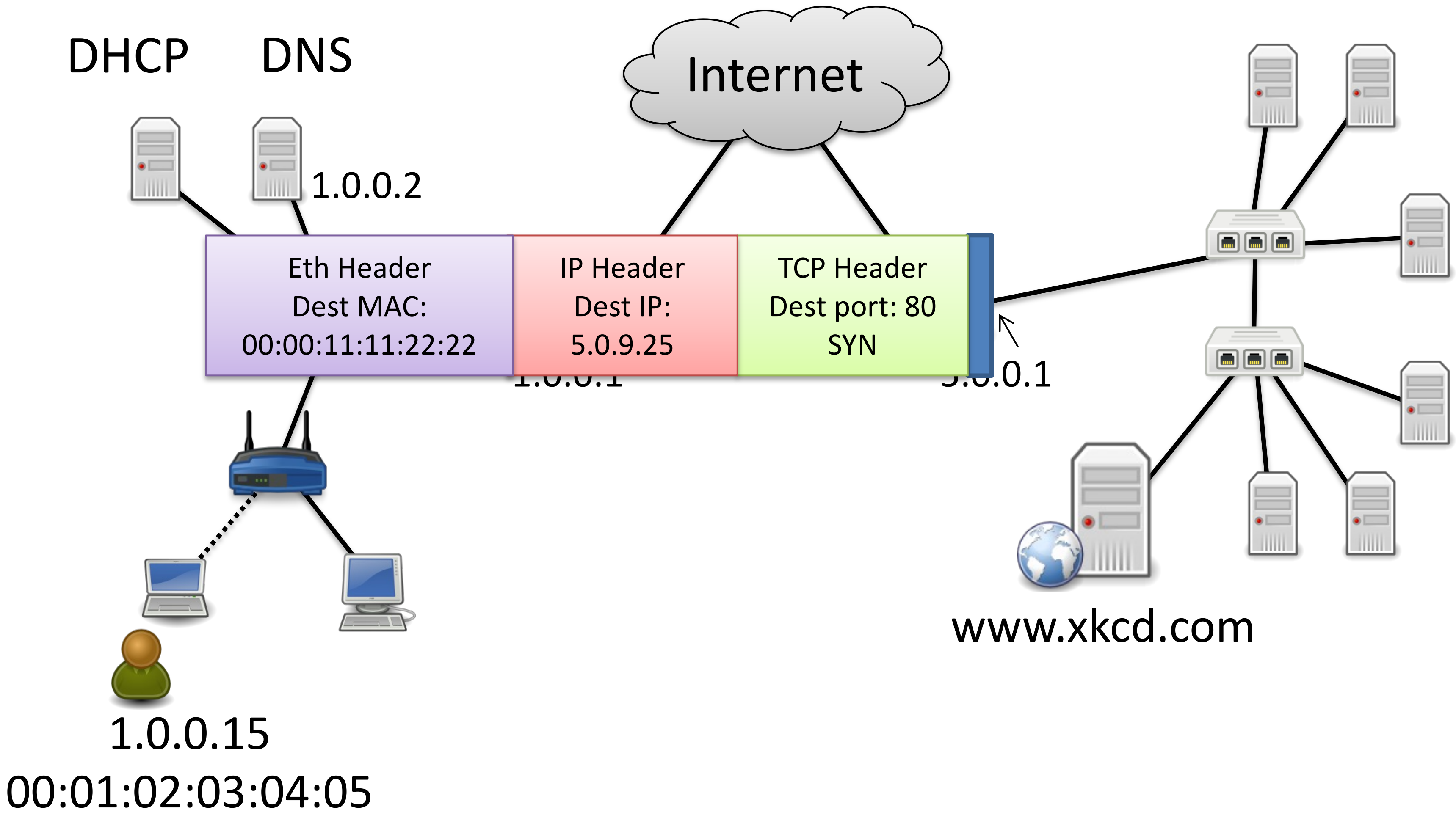
Step 3: Establish a TCP Connection

Send SYN to router. NOTE: while the switch moves the frame to router, it is not ever addressed directly.



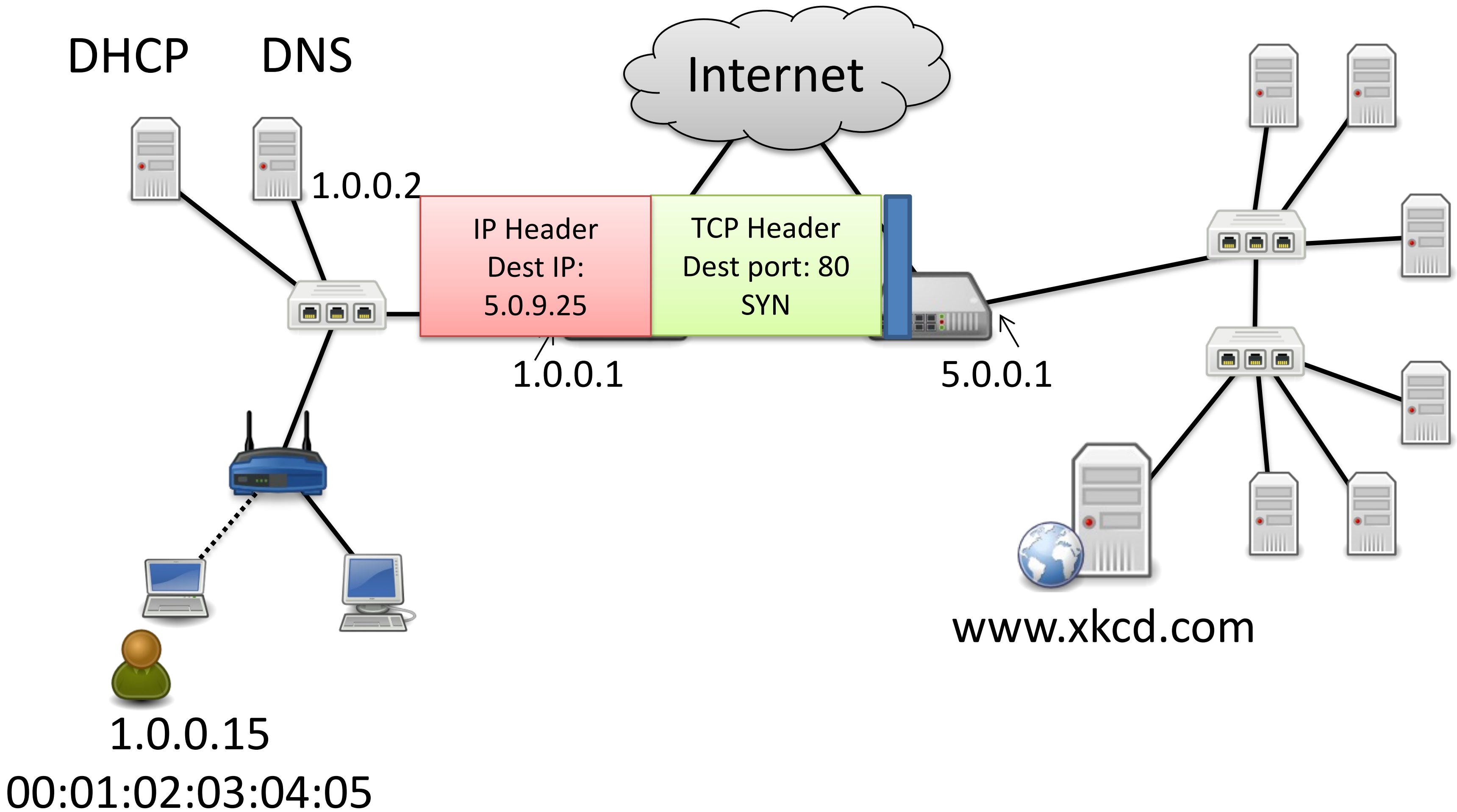
Step 3: Establish a TCP Connection

Router removes Ethernet header.



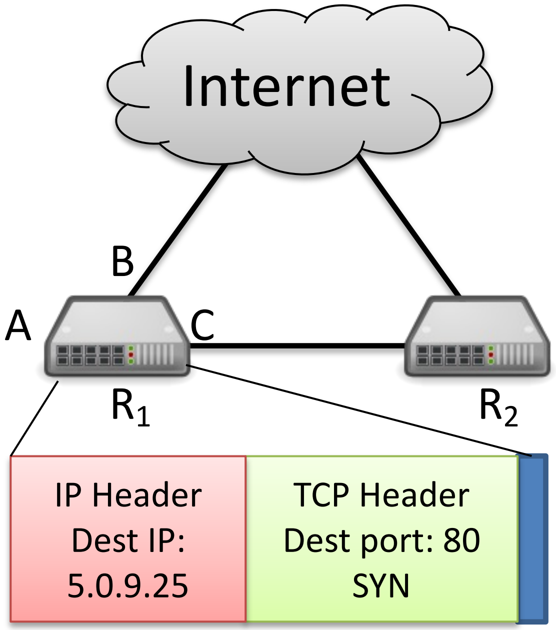
Step 3: Establish a TCP Connection

Router removes Ethernet header.



Step 3: Establish a TCP Connection

Router R1 compares destination IP with its forwarding table, looks for longest prefix match.

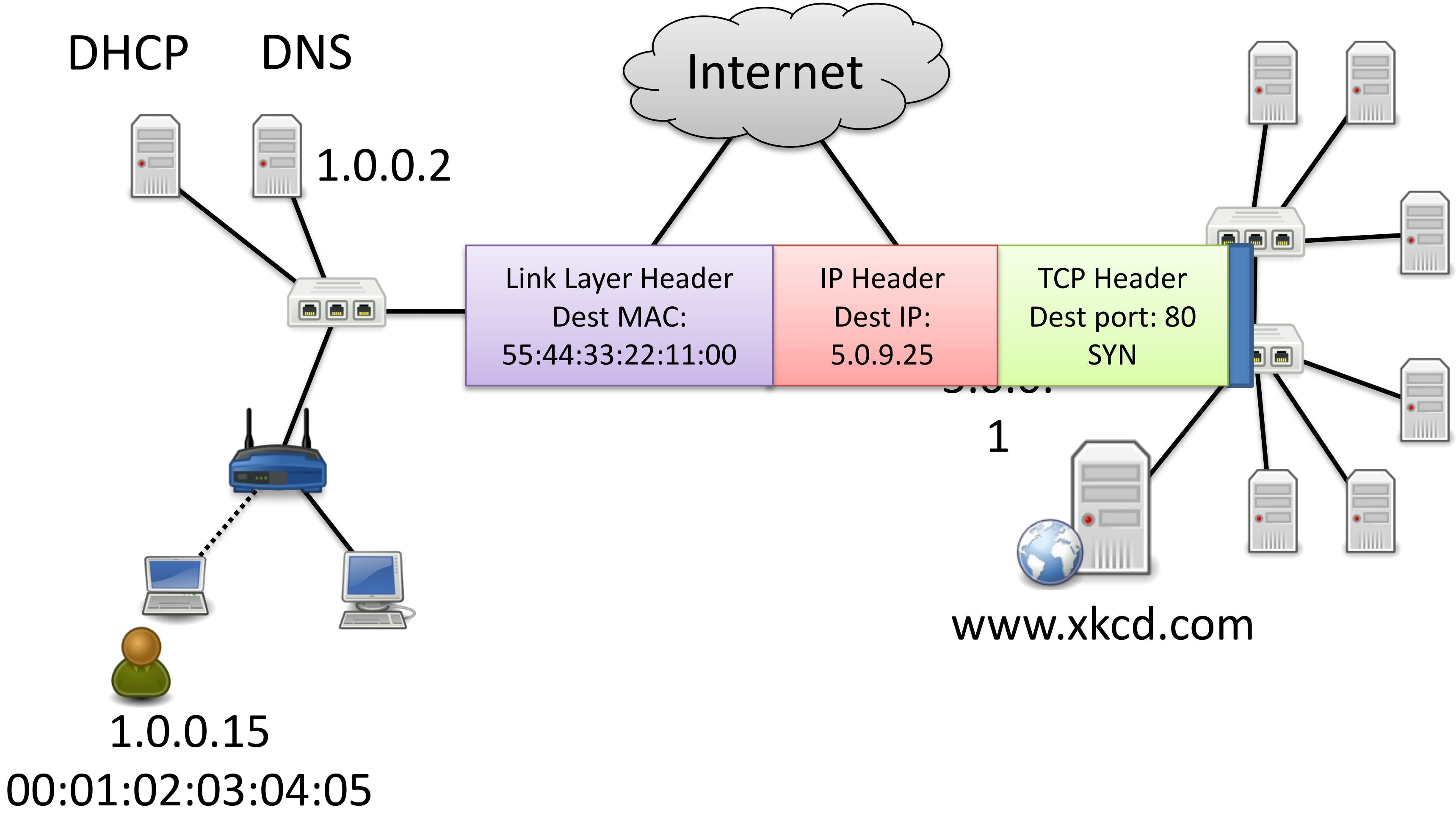


Prefix	Output Port	Next Router's Link Layer Addr
1.0.0.0/24	A	(N/A - no router there)
…	…	…
5.0.0.0/8	B	Some Internet router's address
5.0.0.0/16	C	R_2's Address: 55:44:33:22:11:00
…		

Best match: 5.0.0.0/16 -> Output port C
Destination MAC: 55:44:33:22:11:00

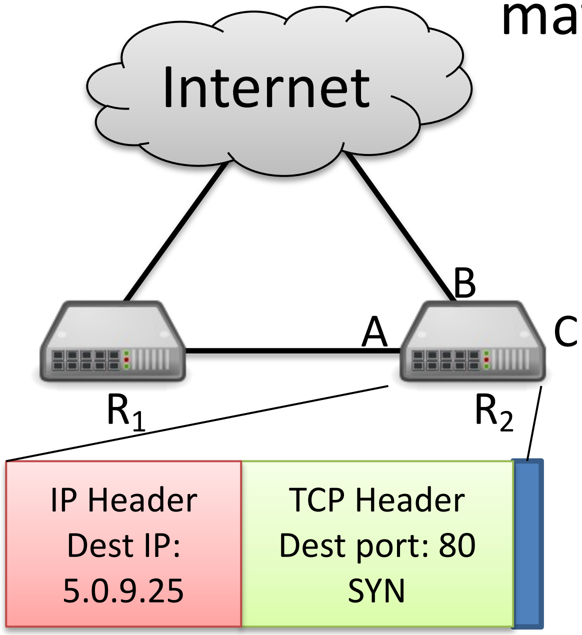
Step 3: Establish a TCP Connection

Router R1 constructs frame and forwards it to R2.



Step 3: Establish a TCP Connection

Router R2 compares destination IP with its forwarding table, looks for longest prefix match.

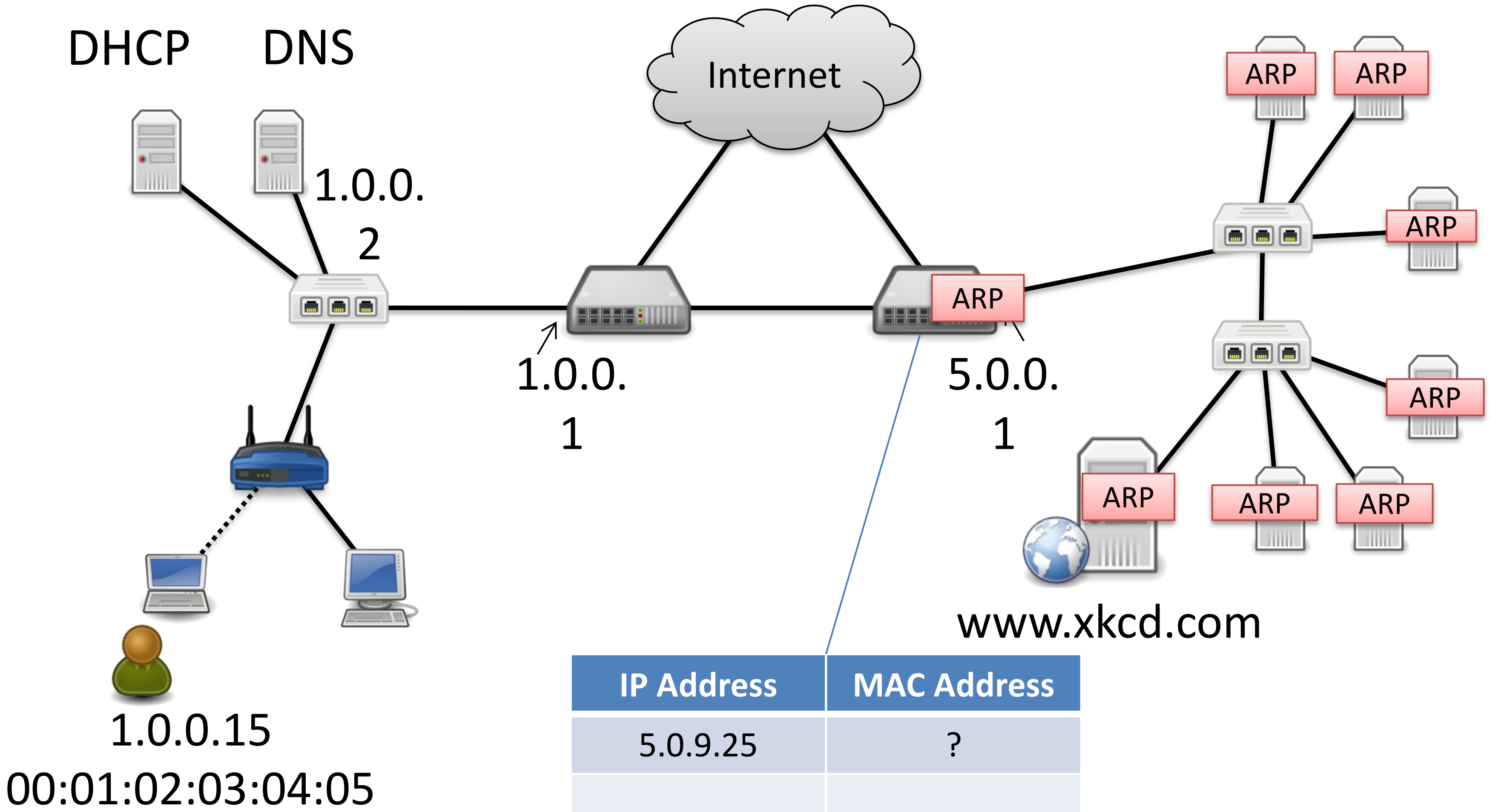


Prefix	Output Port	Next Router's Link Layer Addr
1.0.0.0/24	A	R_1's Address
…	…	…
5.0.0.0/8	B	Some Internet router's address
5.0.0.0/16	C	(N/A - no router there)
…		

Best match: 5.0.0.0/16 -> Output port C
Destination MAC: ?

Step 3: Establish a TCP Connection

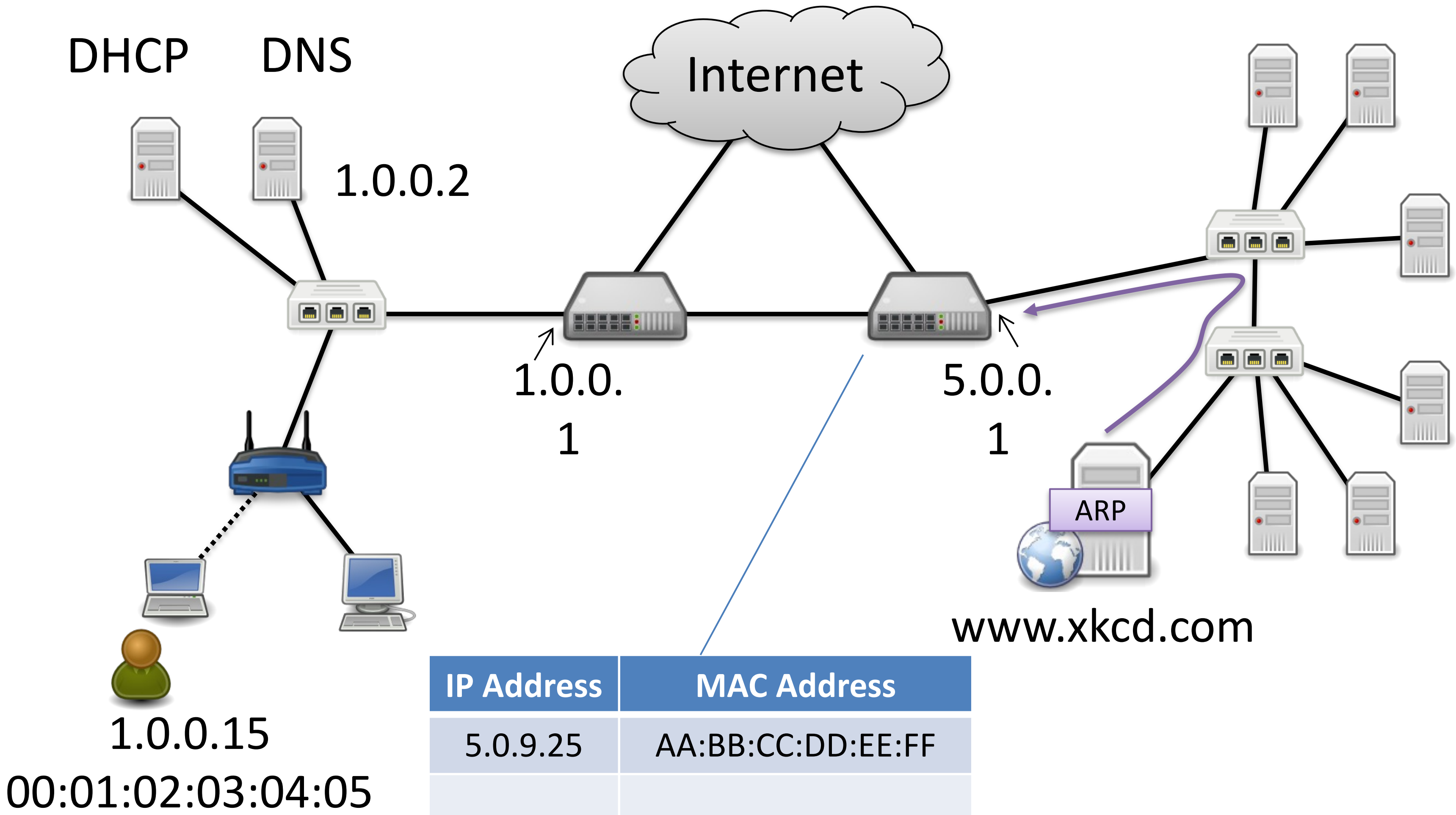
R2 does NOT know destination's MAC address!
Broadcast ARP request looking for 5.0.9.25!
Data packet is queued while waiting for ARP to resolve.



IP Address	MAC Address
5.0.9.25	?

Step 3: Establish a TCP Connection

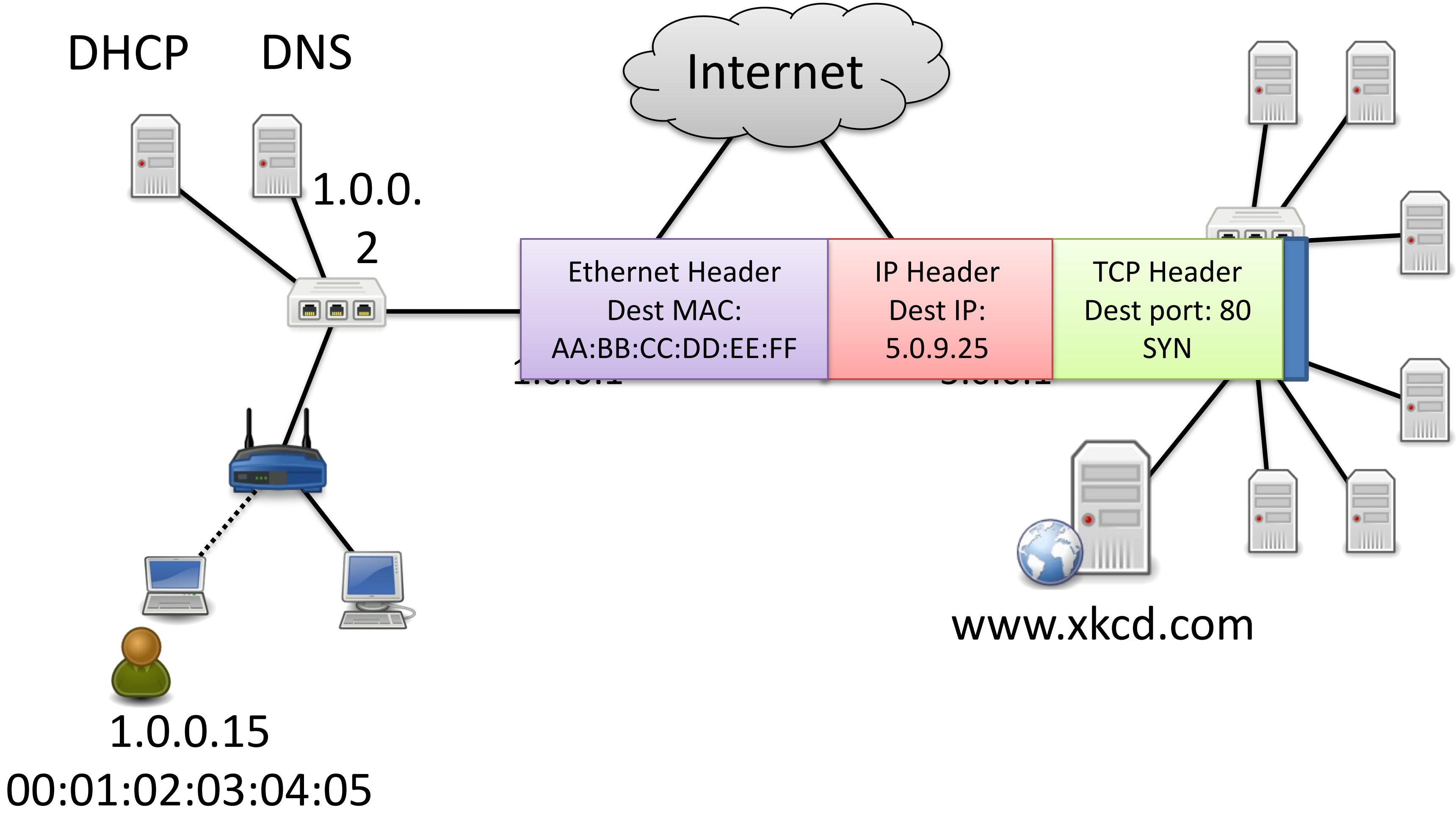
Host replies with MAC address.



IP Address	MAC Address
5.0.9.25	AA:BB:CC:DD:EE:FF

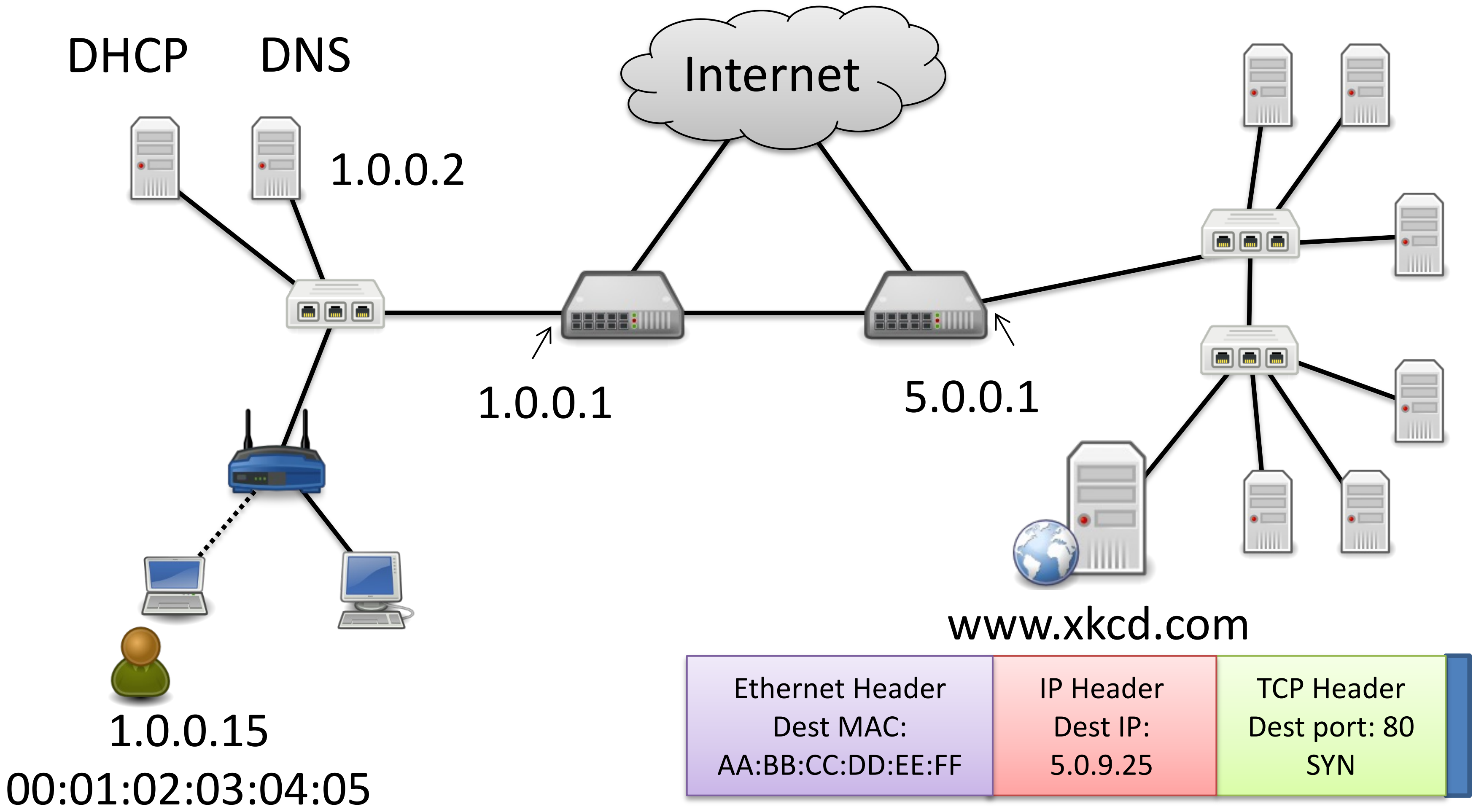
Step 3: Establish a TCP Connection

R2 constructs frame, forwards it to destination.



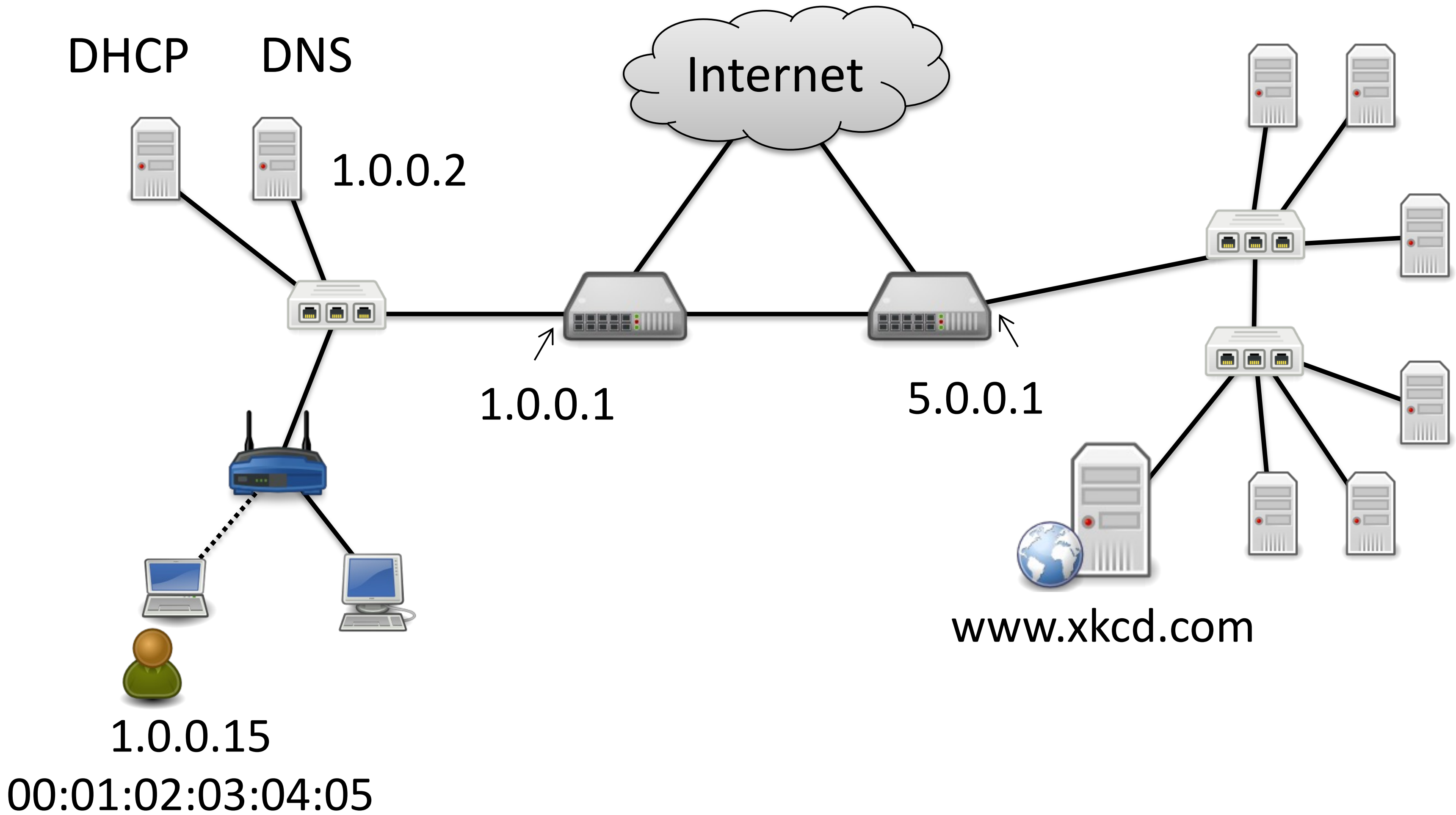
Mission Accomplished!

Destination peels off headers, generates reply (SYN+ACK).



Mission Accomplished!

Process repeats in the opposite direction, without the ARPs this time. (MAC addresses were recently used, thus cached.)



Steady State

- With DNS cached and ARP entries cached, host encapsulates data in TCP, IP, Eth headers and sends to router. Router forwards.

- Even with all the DNS/ARP, all that stuff happens in < 1 second.
  (besides step 0: routing protocol)


That's all Folks!