Computer Networks

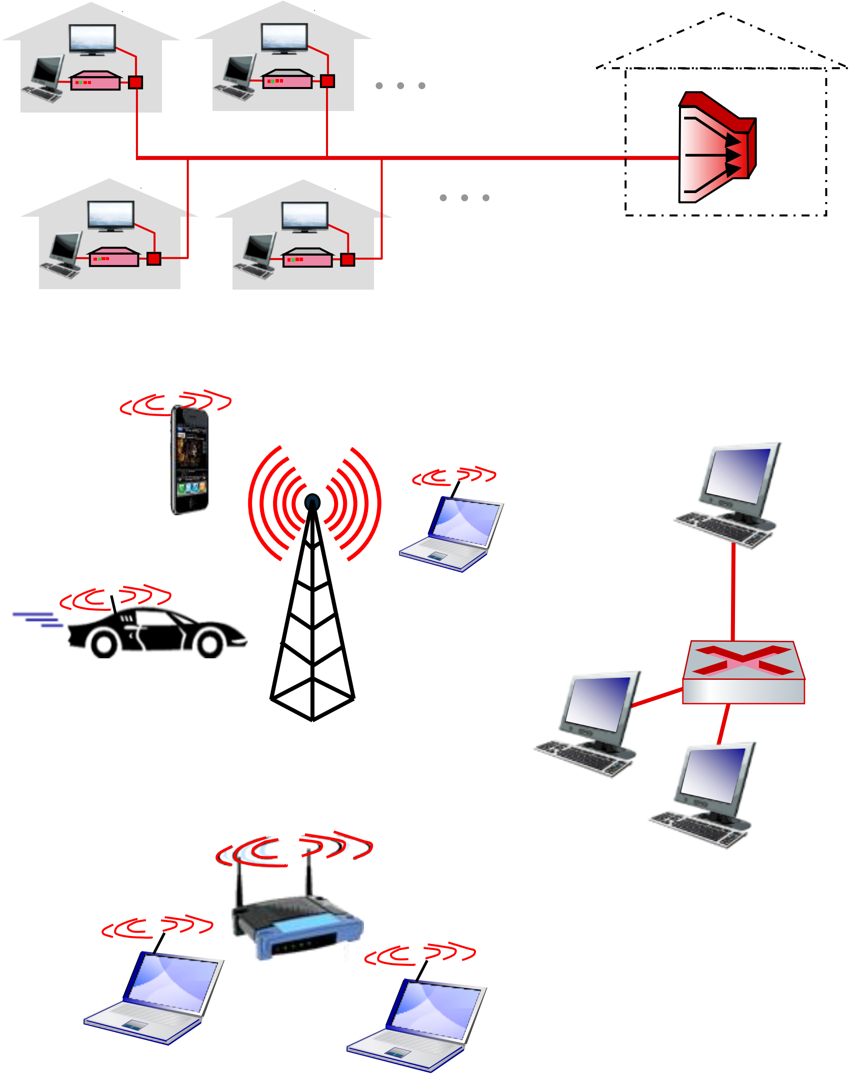
(SCC.203)

Datalink Layer

Muhammad Bilal

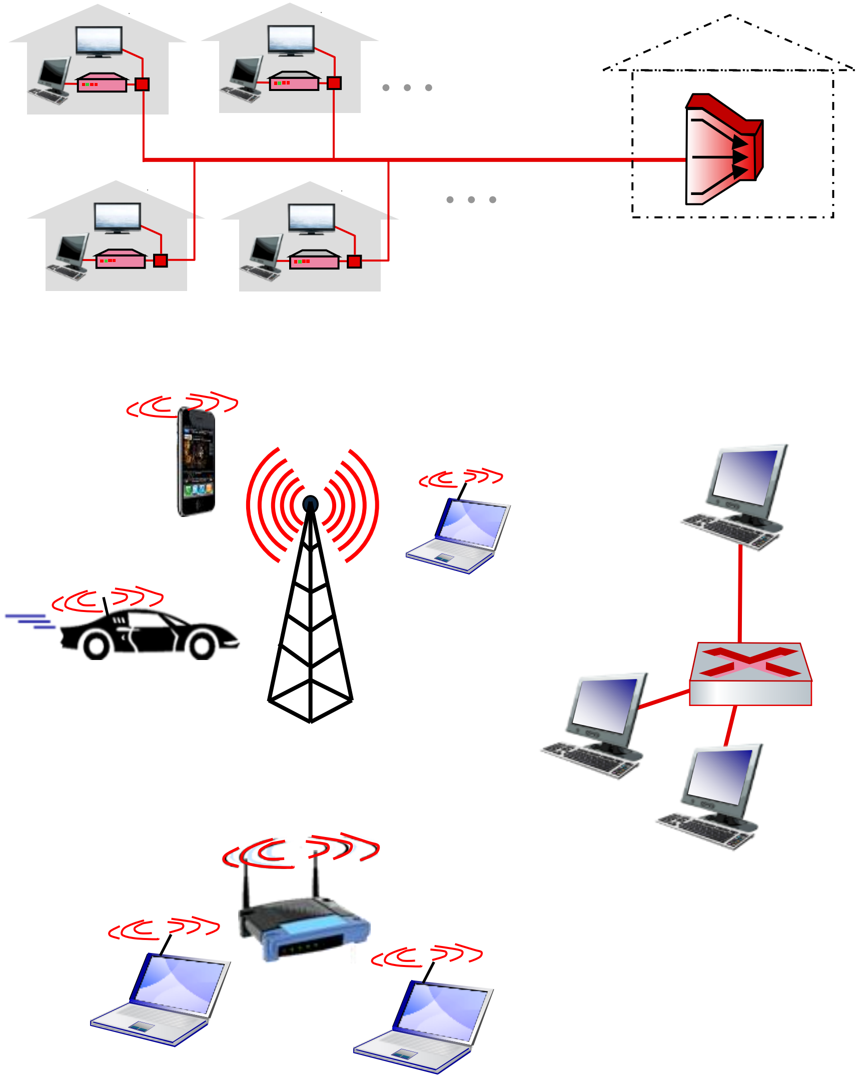
Link layer: services

- framing, link access:
  - encapsulate datagram into frame (**based on underlying tech**), adding header, trailer
  - channel access if shared medium
  - “MAC” addresses in frame headers identify source, destination (different from IP address!)
- reliable delivery between adjacent nodes
  - we already know how to do this! rtd,..
  - wireless links: high error rates
  - ARQ (Automatic Repeat reQuest) used in Wi-Fi and Ethernet networks.



Link layer: services (more)

- 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



Error Detection-Correction

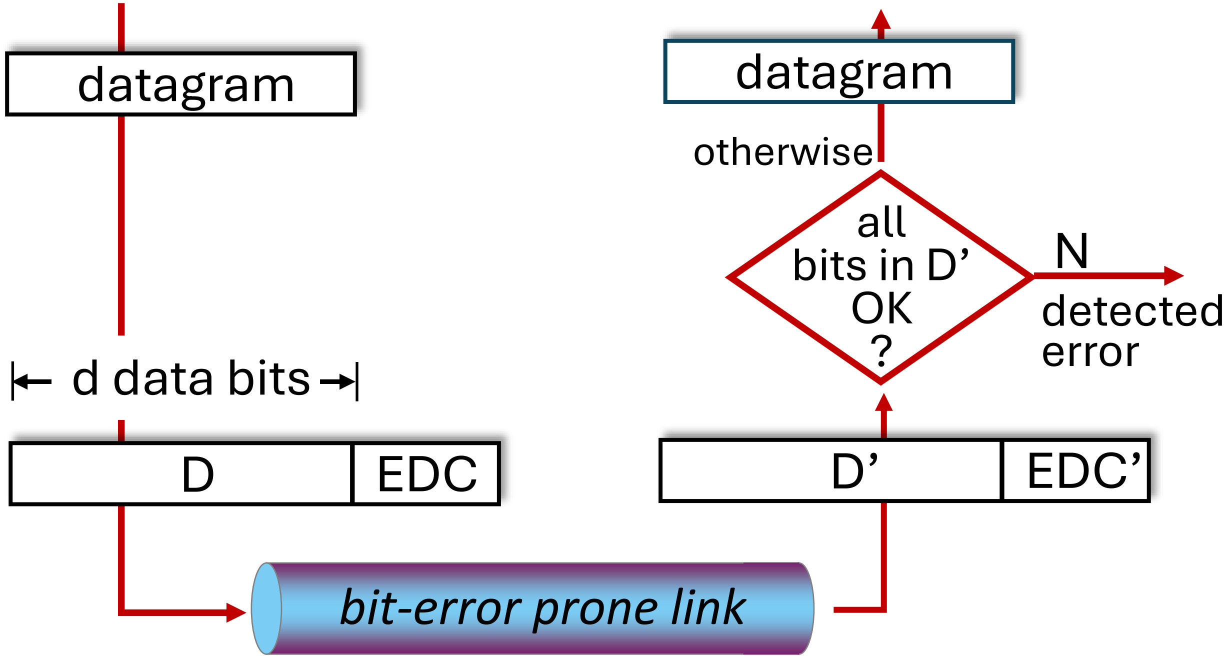
Parity, Checksum, CRC

Dealing with Noise

- The physical world is inherently noisy
  - Interference from electrical cables
  - Cross-talk from radio transmissions, microwave ovens
  - Solar storms
- How to detect bit-errors in transmissions?
- How to recover from errors?

Error detection

EDC: error detection and correction bits (e.g., redundancy)
D: data protected by error checking, may include header fields



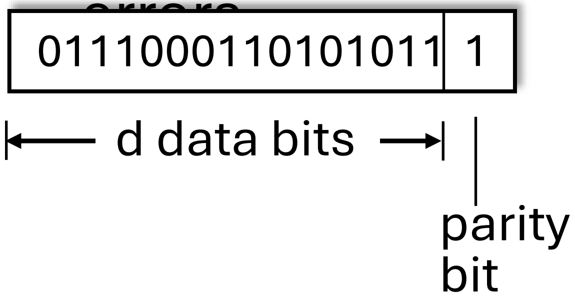
Error detection not 100% reliable!

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

Parity checking

single bit parity:

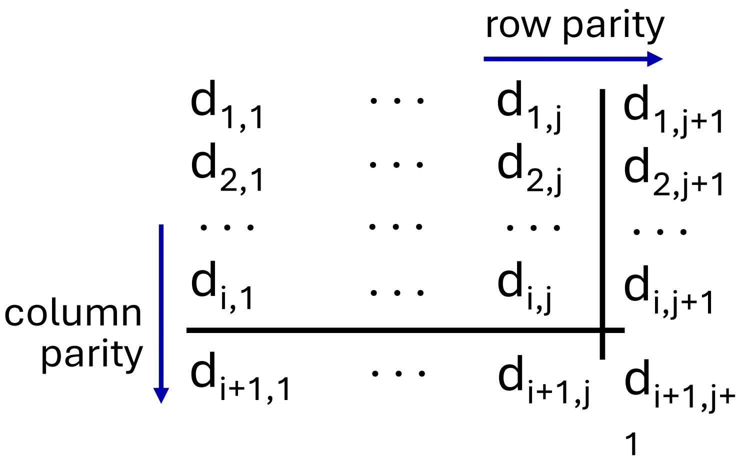
- detect single bit errors



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

two-dimensional bit parity:

- detect *and correct* single bit errors

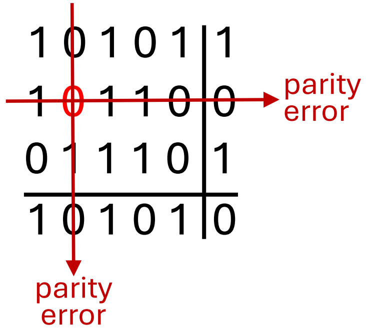


$$\begin{array}{cccc|c}
d_{1,1} & \cdots & d_{1,j} & & d_{1,j+1} \\
d_{2,1} & \cdots & d_{2,j} & & d_{2,j+1} \\
\cdots & \cdots & \cdots & & \cdots \\
d_{i,1} & \cdots & d_{i,j} & & d_{i,j+1} \\
\hline
d_{i+1,1} & \cdots & d_{i+1,j} & & d_{i+1,j+1}
\end{array}$$

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:



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Internet checksum (review)

Checksum in IPv6 does not exit, why?

Goal: detect errors (*i.e.,* flipped bits) in transmitted segment

sender:

- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

receiver:

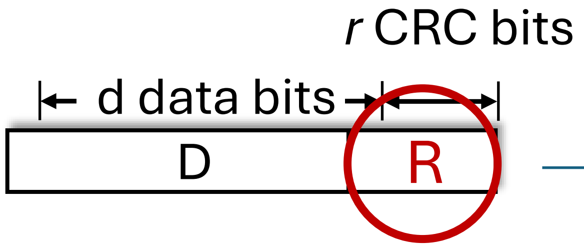
- compute checksum of received segment
- check if computed checksum equals checksum field value:
    - not equal - error detected
    - equal - no error detected. *But maybe errors nonetheless?* More later ….

Cyclic Redundancy Check (CRC)

- Uses field theory to compute a semi-unique value for a given message
- Much better performance than previous approaches
  - Fixed size overhead per frame (usually 32-bits)
  - Quick to implement in hardware
  - Only 1 in 2^{32} chance of missing an error with 32-bit CRC

Cyclic Redundancy Check (CRC)

- D: data bits (given, think of these as a binary number)
- G: Sender and Receiver agree on a *r+1* bit pattern G (generator)



r CRC bits

d data bits

D

R

bit pattern

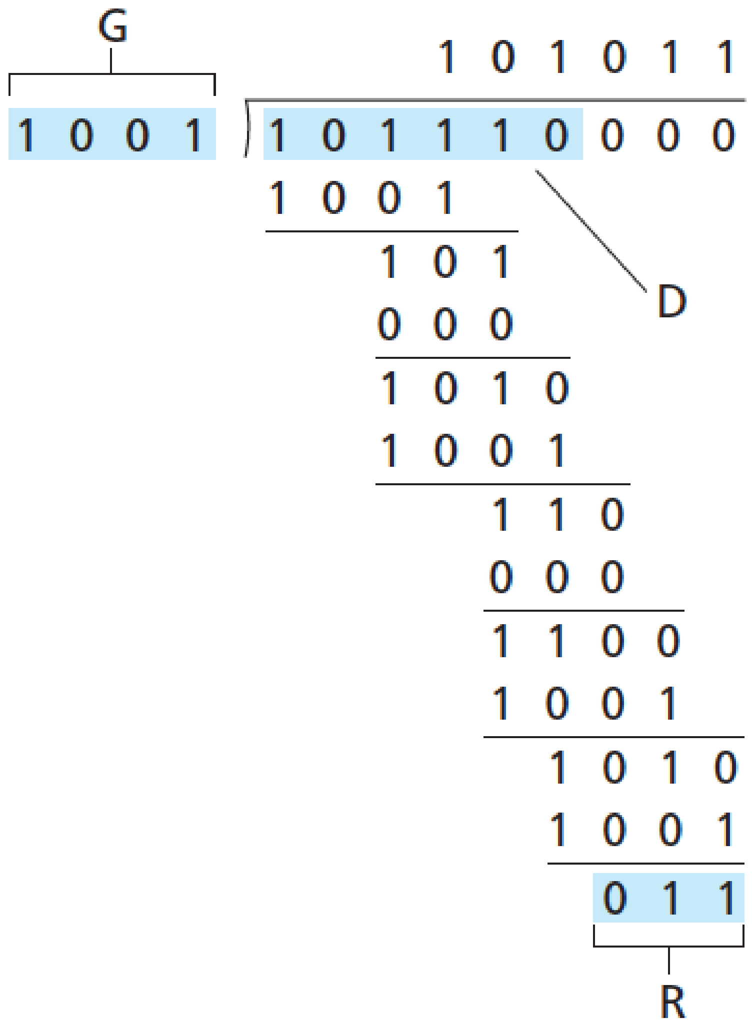
$\langle D,R\rangle = D*2^r \text{ XOR } R$

formula for bit pattern

goal: choose *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)

CRC Example



- D = 101110
- Generator polynomial:
    - $x^3 + 1 = 1x^3 + 0x^2 + 0x^1 + 1x^0$
      G = 1001
- Since G is 4 bits we append 3 0 bits at the end of our data D
- Divide the padded data by G using modulo-2 arithmetic

CRC Example 2

G 1 1 0 0 1 0

D 1 0 1 1 0 1 0 0 1 0

```
1 1 0 0 1 0 | 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0
              1 1 0 0 1 0
              -----------
                1 1 1 1 1 0 0 1 0 0 0 0 0 0
                1 1 0 0 1 0
                -----------
                  0 1 1 0 0 0 1 0 0 0 0 0 0
                  0 0 0 0 0 0
                  -----------
                    1 1 0 0 0 1 0 0 0 0 0 0
                    1 1 0 0 1 0
                    -----------
                      0 0 0 1 1 0 0 0 0 0 0
                      0 0 0 0 0 0
                      -----------
                        0 0 1 1 0 0 0 0 0 0
                        0 0 0 0 0 0
                        -----------
                          0 1 1 0 0 0 0 0 0
                          0 0 0 0 0 0
                          -----------
                            1 1 0 0 0 0 0 0
                            1 1 0 0 1 0
```

CRC Example 2 (cont.)

cont.

```
1 1 0 0 0 0 0 0
1 1 0 0 1 0
-------------
  0 0 0 1 0 0 0
  0 0 0 0 0 0 0
  -------------
    0 0 1 0 0 0
    0 0 0 0 0 0
    -----------
      0 1 0 0 0
```

Data to send:

1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 0 1 0 0 0

Receiver repeats process on D with same G.

If remainder == 0, no error

If remainder != 0, error!

Standardized polynomials

$$\mathrm{CRC} - 12 = x^{12} + x^{11} + x^{3} + x^{2} + x^{1} + 1$$

$$\mathrm{CRC} - 16 = x^{16} + x^{15} + x^{2} + 1$$

$$\mathrm{CRC} - \mathrm{CCITT} = x^{16} + x^{12} + x^{5} + 1$$

- CRC - CCITT recognizes
  - All single and duplicate errors
  - All errors with odd bit numbers
  - All burst errors up to a length of 16
  - 99.99 % of all burst errors of a length of 17 and more

Multiple access

TDM, FDM, ALOHA, CSMA

Multiple access links, protocols

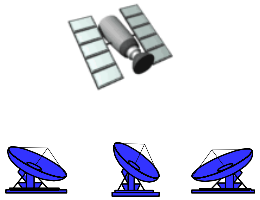
two types of “links”:

- point-to-point
  - point-to-point link between Ethernet switch, host
  - PPP for dial-up access
- broadcast (shared wire or medium)
  - old-fashioned Ethernet
  - upstream HFC in cable-based access network
  - 802.11 wireless LAN, 4G/4G. satellite

shared wire (e.g., cabled Ethernet)

shared radio: 4G/5G

shared radio: WiFi



shared radio: satellite

humans at a cocktail party (shared air, acoustical)

Multiple access protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes: interference
  - *collision* if node receives two or more signals at the same time

multiple access protocol

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

An ideal multiple access protocol

given: multiple access channel (MAC) of rate *R* bps

1. when 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*
3. fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots
4. simple

MAC protocols: classification

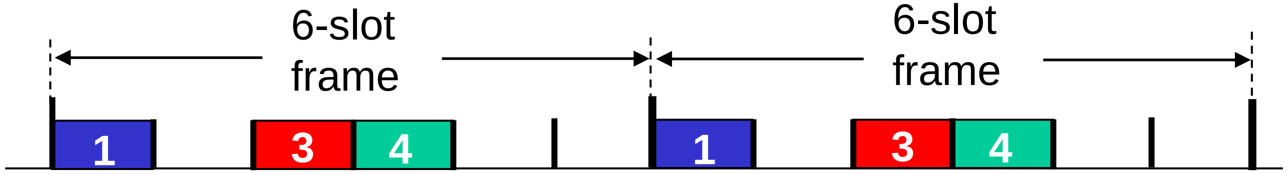
three broad classes:

- 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 take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

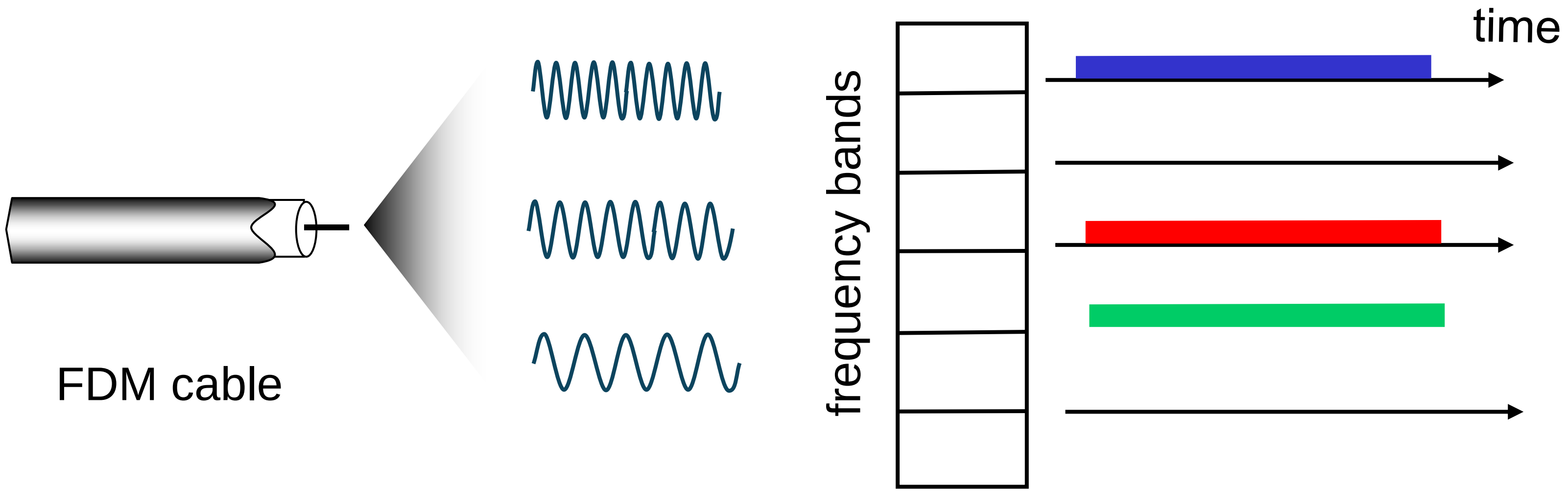
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have packet to send, frequency bands 2,5,6 idle



Random access protocols

- when node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- two or more transmitting nodes: “collision”
- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- examples of random access MAC protocols:
  - ALOHA, slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA

Slotted ALOHA

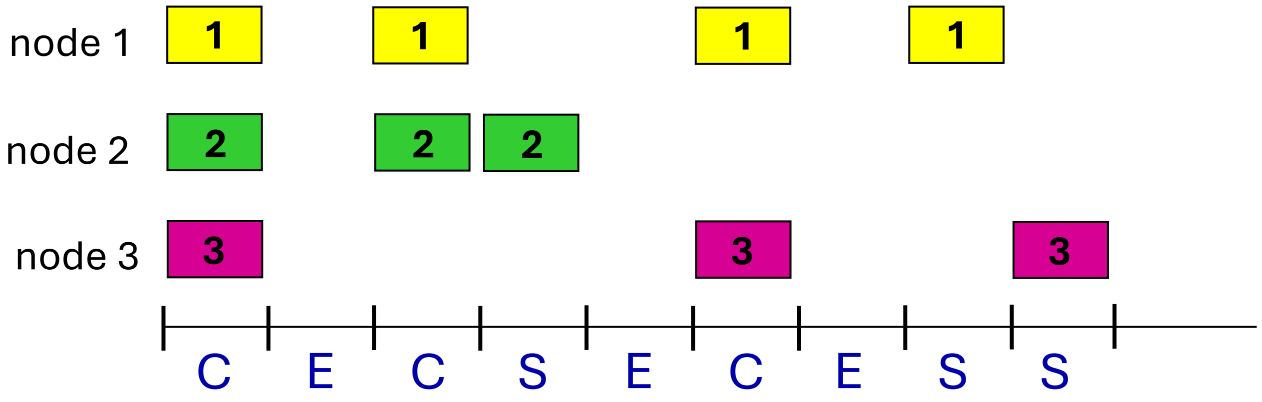
assumptions:

- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

operation:

- when node obtains fresh frame, transmits in next slot
    - *if no collision:* node can send new frame in next slot
    - *if collision:* node retransmits frame in each subsequent slot with probability *p* until success

Slotted ALOHA



C: collision
S: success
E: empty

Pros:

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons:

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

Slotted ALOHA: efficiency

efficiency: long-run fraction of successful slots (many nodes, all with many frames to send)

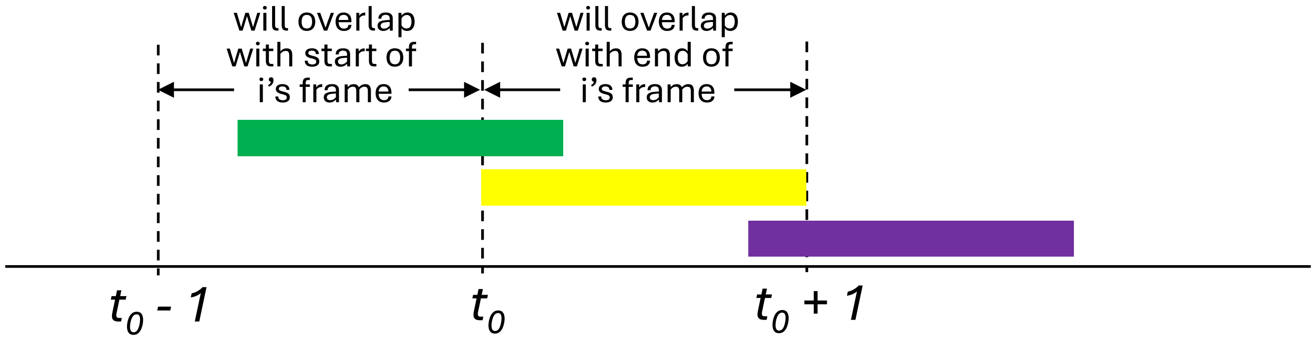
- *suppose:* N nodes with many frames to send, each transmits in slot with probability p
  - prob that given node has success in a slot $= p(1-p)^{N-1}$
  - prob that *any* node has a success $= Np(1-p)^{N-1}$
  - max efficiency: find p^* that maximizes $Np(1-p)^{N-1}$
  - for many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives:

max efficiency $= 1/e = .37$

- *at best:* channel used for useful transmissions 37% of time!

Pure ALOHA

- unslotted Aloha: simpler, no synchronization
  - when frame first arrives: transmit immediately
- collision probability increases with no synchronization:
  - frame sent at t_0 collides with other frames sent in $[t_0-1, t_0+1]$



- pure Aloha efficiency: 18% !

CSMA (carrier sense multiple access)

simple CSMA: listen before transmit:

- if channel sensed idle: transmit entire frame
- if channel sensed busy: defer transmission

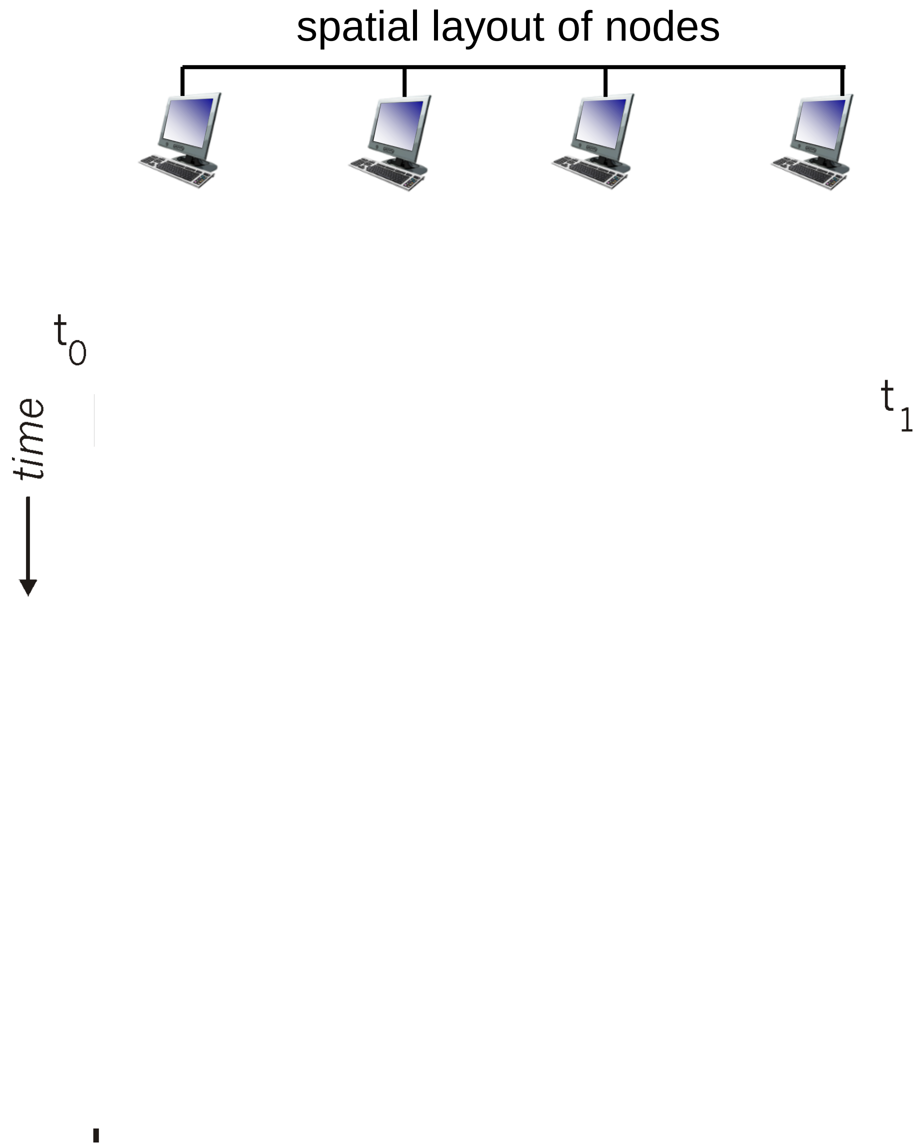
- human analogy: don't interrupt others!

CSMA/CD: CSMA with *collision detection*

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage
- collision detection easy in wired, difficult with wireless

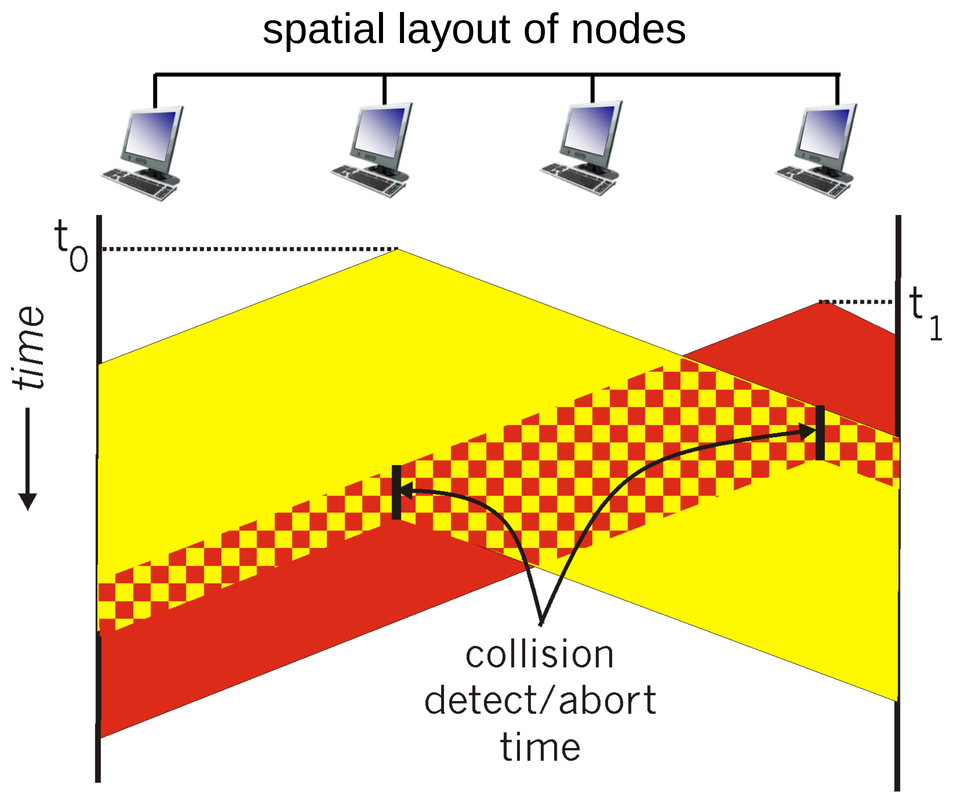
CSMA: collisions

- collisions *can* still occur with carrier sensing:
  - propagation delay means two nodes may not hear each other's just-started transmission
- collision: entire packet transmission time wasted
  - distance & propagation delay play role in in determining collision probability



CSMA/CD:

- CSMA/CS reduces the amount of time wasted in collisions
  - transmission aborted on collision detection
- NICs detect collisions by comparing the signal they are transmitting with the signal they are receiving on the network.
- In a properly functioning network, the NIC should detect its own carrier signal that it's transmitting.



Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel:
   if idle: start frame transmission.
   if busy: wait until channel idle, then transmit
3. If NIC transmits entire frame without collision, NIC is done with frame !
4. If NIC detects another transmission while sending: abort, send jam signal
5. After aborting, NIC enters *binary (exponential) backoff:*
   - after *m*th collision, NIC chooses *K* at random from $\{0,1,2, \ldots, 2^m-1\}$. NIC waits $K \cdot 512$ bit times, returns to Step 2
   - more collisions: longer backoff interval

CSMA/CD efficiency

- T_{prop} = max prop delay between 2 nodes in LAN
- t_{trans} = time to transmit max-size frame

$$efficiency = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

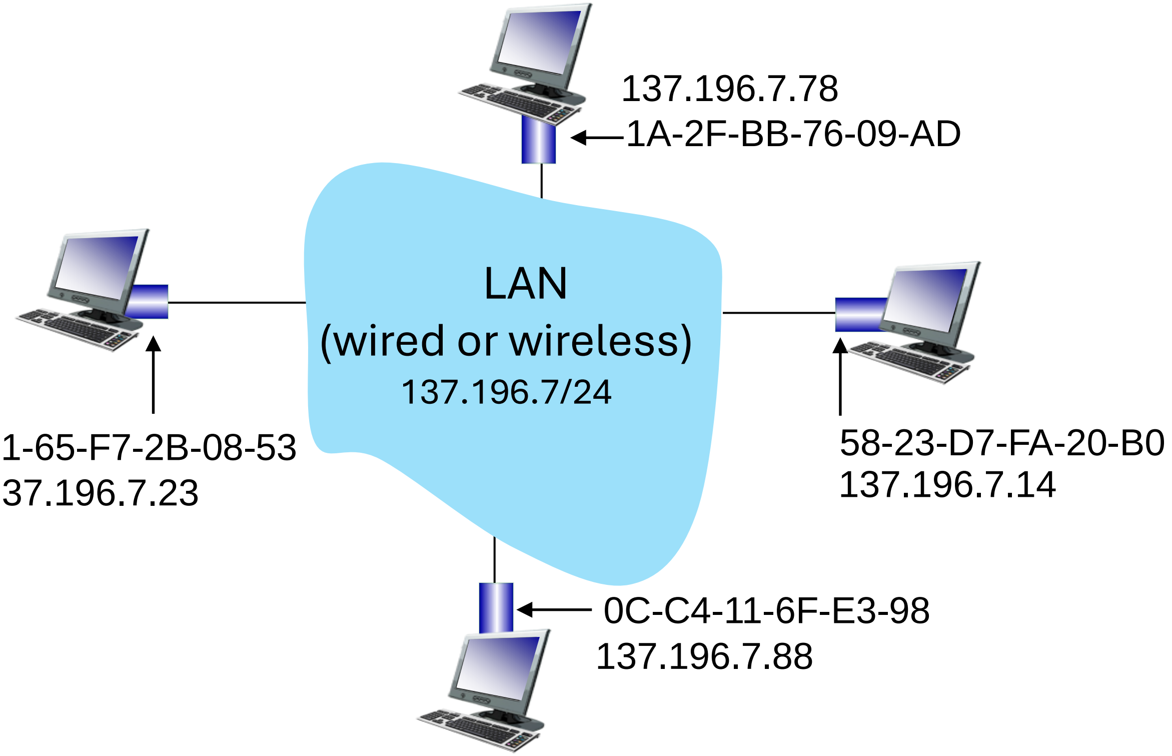
- efficiency goes to 1
  - as t_{prop} goes to 0
  - as t_{trans} goes to infinity
- better performance than ALOHA: and simple, cheap, decentralized!

Address Resolution Protocol

MAC addresses

each interface on LAN

- has unique 48-bit MAC address
- has a locally unique 32-bit IP address (as we've seen)

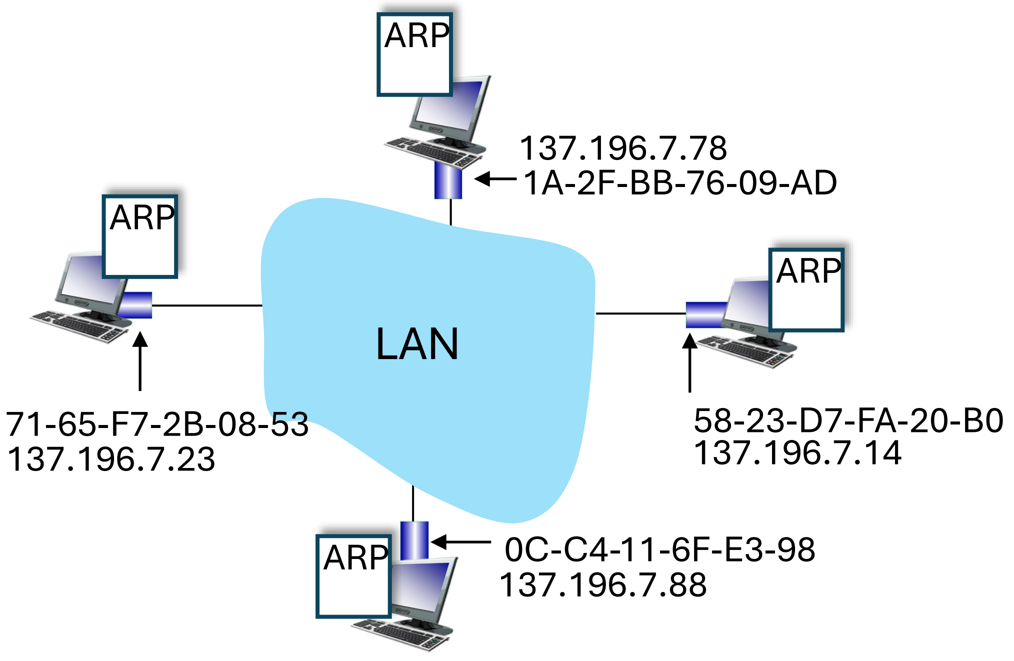


MAC addresses

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
- MAC flat address: portability
  - can move interface from one LAN to another
  - recall IP address *not* portable: depends on IP subnet to which node is attached

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 in action

example: A wants to send datagram to B

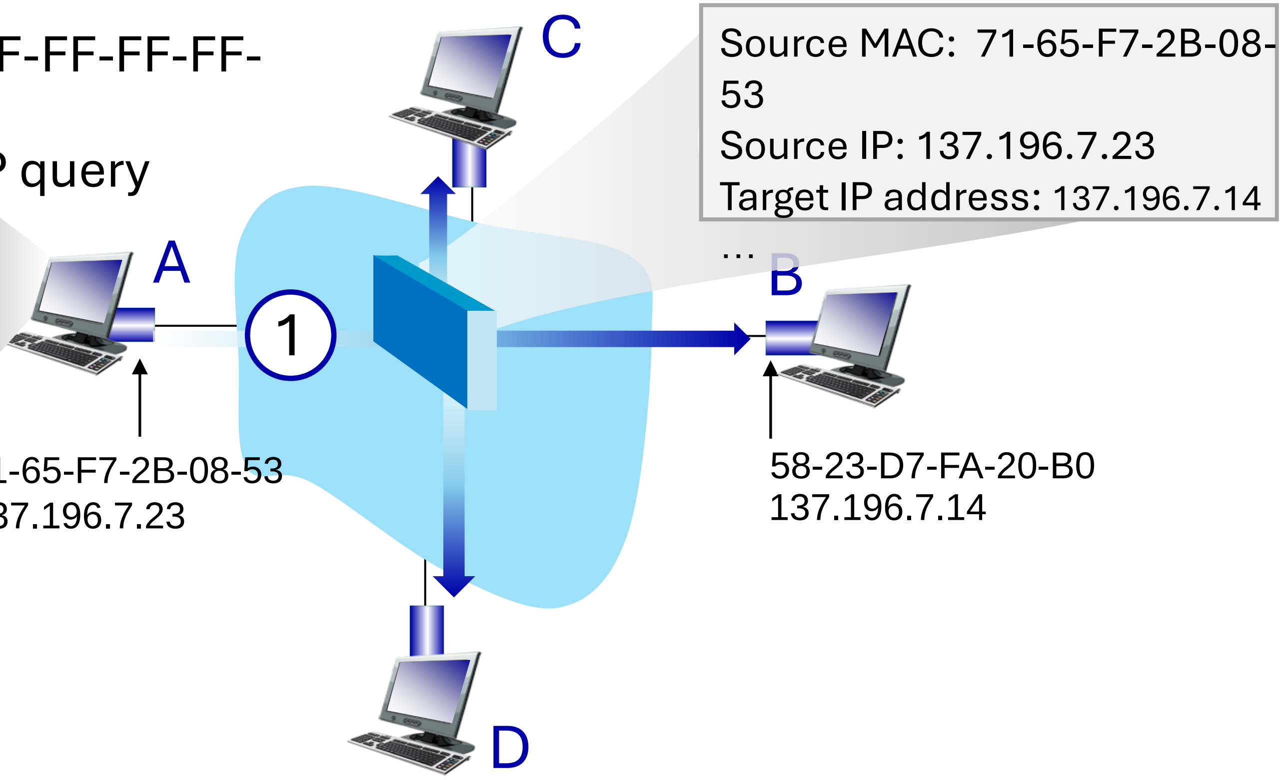
- B’s MAC address not in A’s ARP table, so A uses ARP to find B’s MAC address

① A broadcasts ARP query, containing B's IP addr

- destination MAC address = FF-FF-FF-FF-FF-FF
- all nodes on LAN receive ARP query

ARP table in A

IP addr	MAC addr	TTL



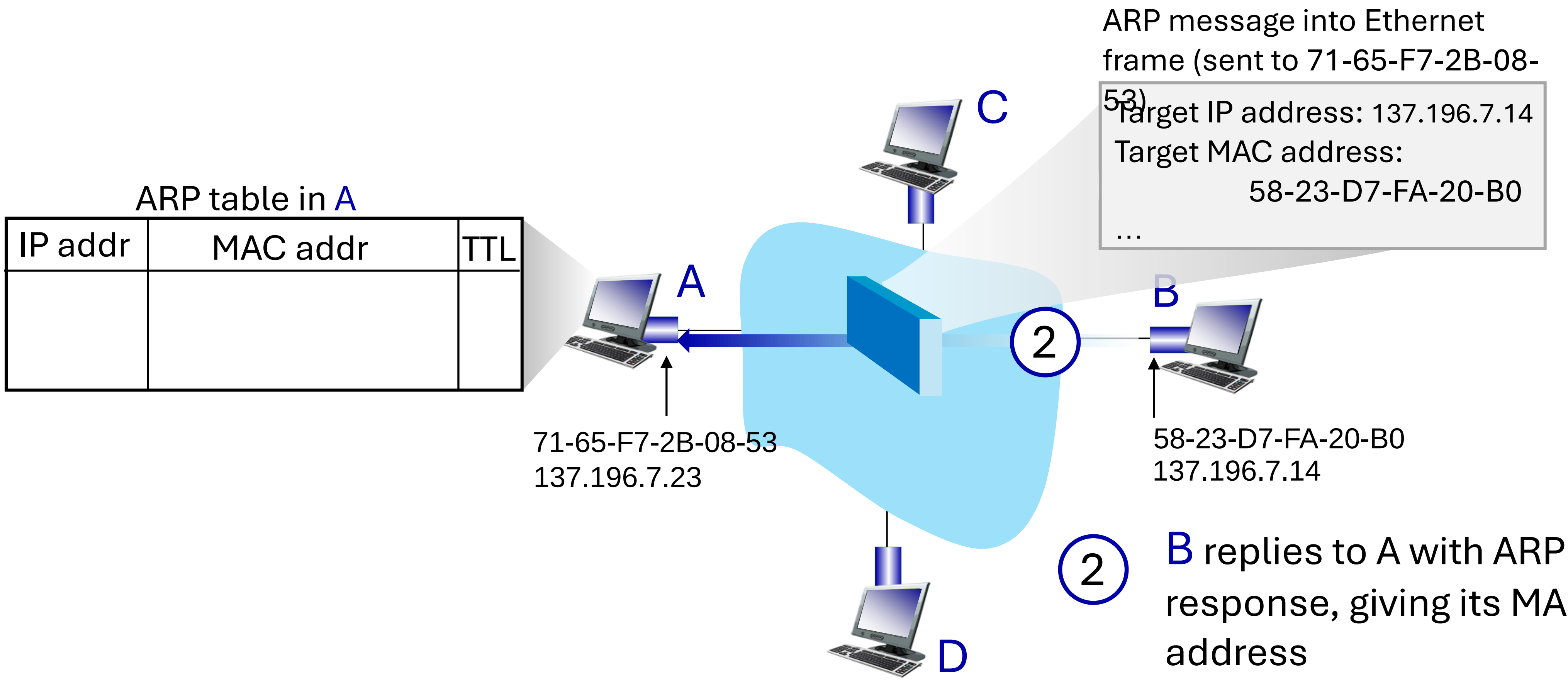
Ethernet frame (sent to FF-FF-FF-FF-FF-FF)

Source MAC: 71-65-F7-2B-08-53
Source IP: 137.196.7.23
Target IP address: 137.196.7.14
…

ARP protocol in action

example: A wants to send datagram to B

- B’s MAC address not in A’s ARP table, so A uses ARP to find B’s MAC address



ARP table in A

IP addr	MAC addr	TTL

ARP message into Ethernet frame (sent to 71-65-F7-2B-08-53)

Target IP address: 137.196.7.14
Target MAC address:
58-23-D7-FA-20-B0
…

② B replies to A with ARP response, giving its MAC address

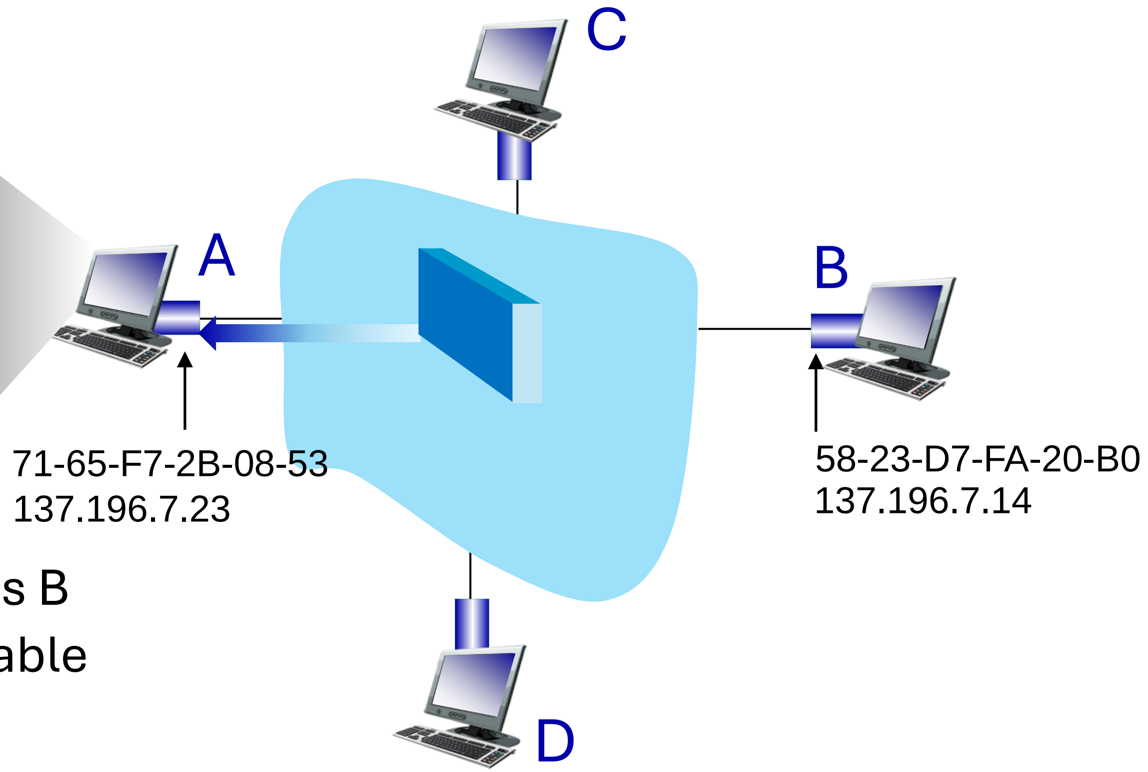
ARP protocol in action

example: A wants to send datagram to B

- B’s MAC address not in A’s ARP table, so A uses ARP to find B’s MAC address

ARP table in A

IP addr	MAC addr	TTL
137.196. 7.14	58-23-D7-FA-20-B0	500

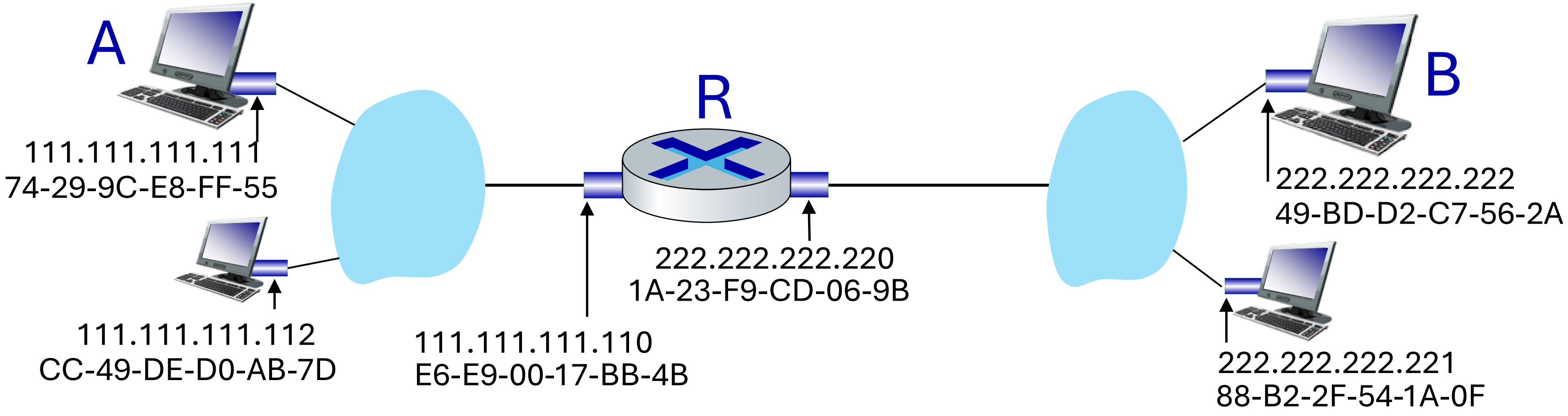


③ A receives B’s reply, adds B entry into its local ARP table

Routing to another subnet: addressing

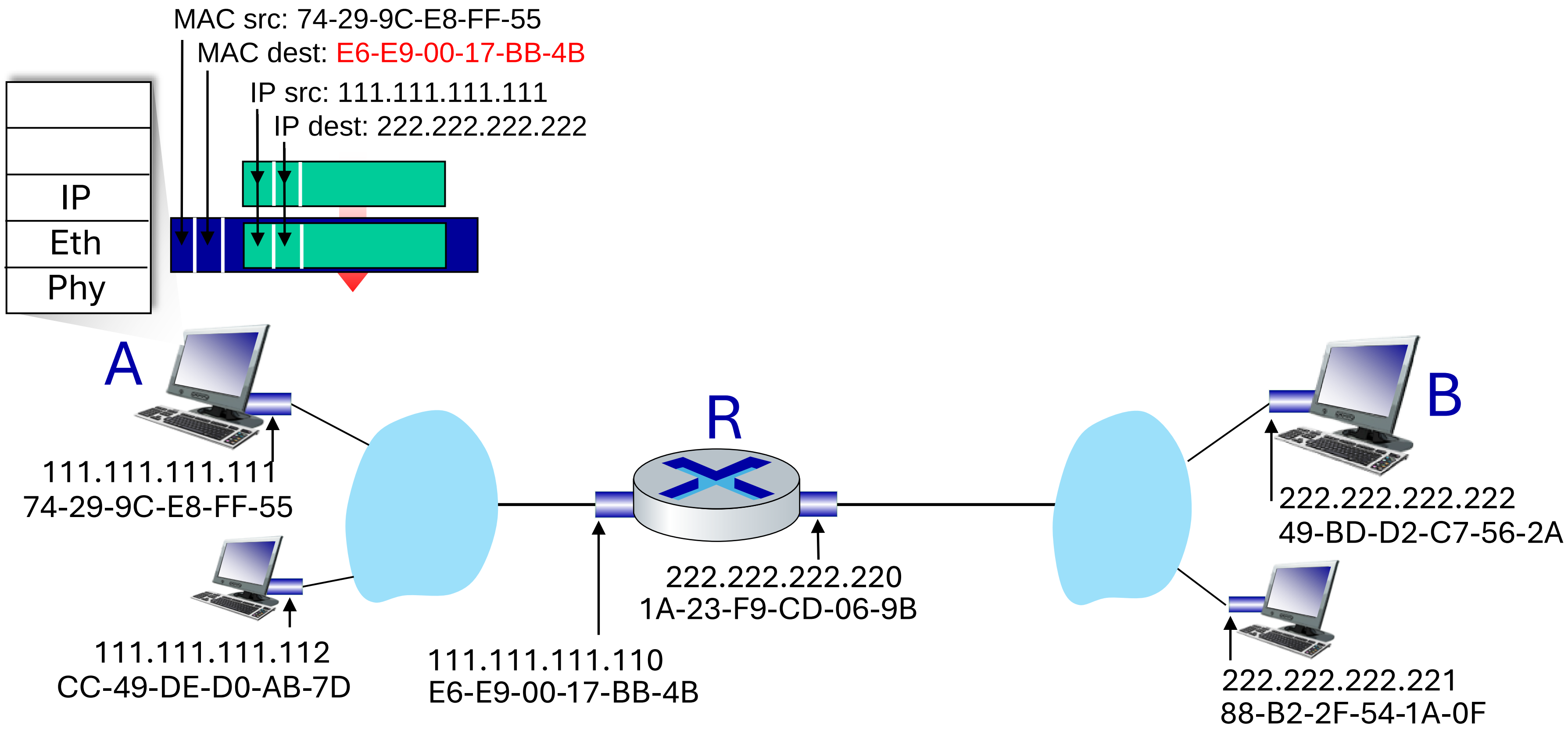
walkthrough: sending a datagram from *A* to *B* via *R*

- focus on addressing – at IP (datagram) and MAC layer (frame) levels
- assume that:
  - A knows B's IP address
  - A knows IP address of first hop router, R (how?)
  - A knows R's MAC address (how?)



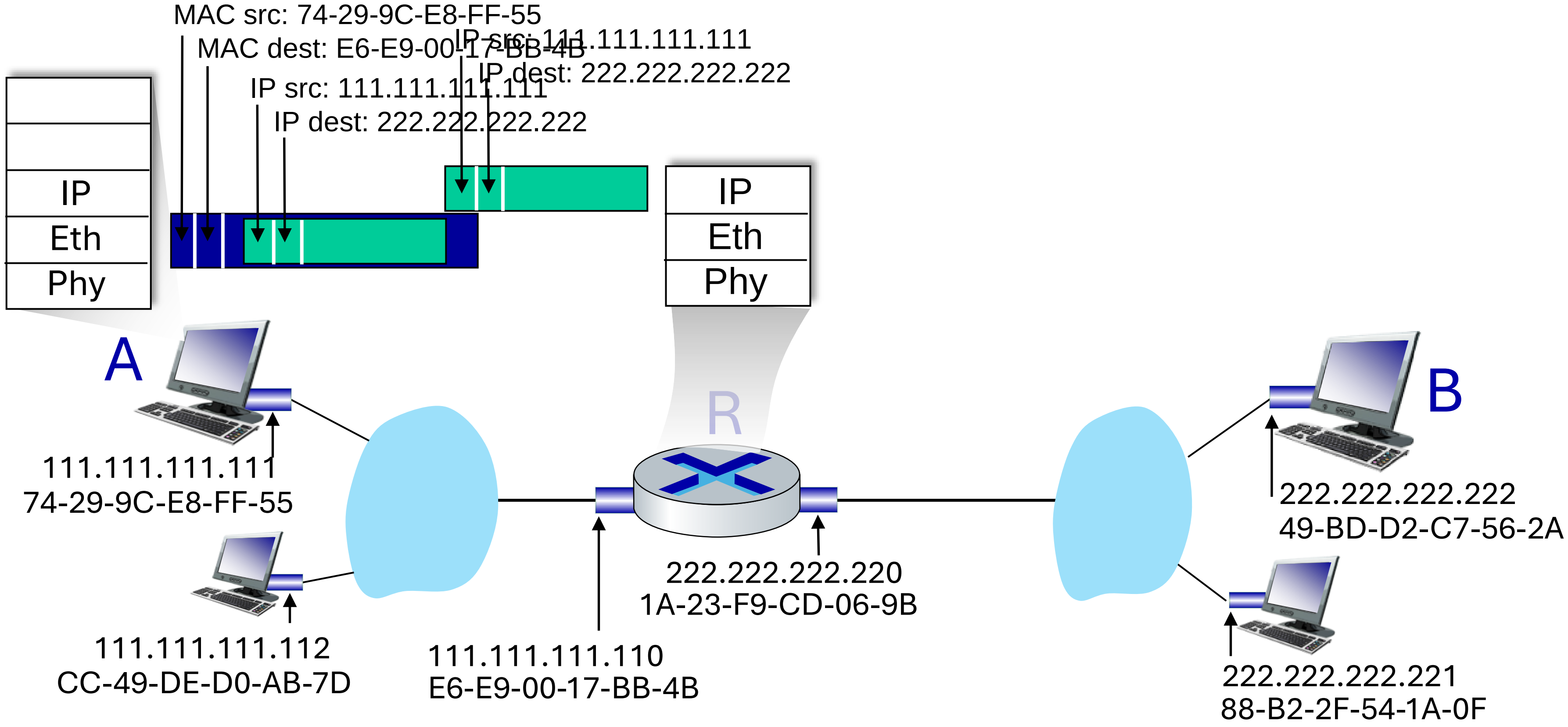
Routing to another subnet: addressing

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame containing A-to-B IP datagram
  - R's MAC address is frame’s destination



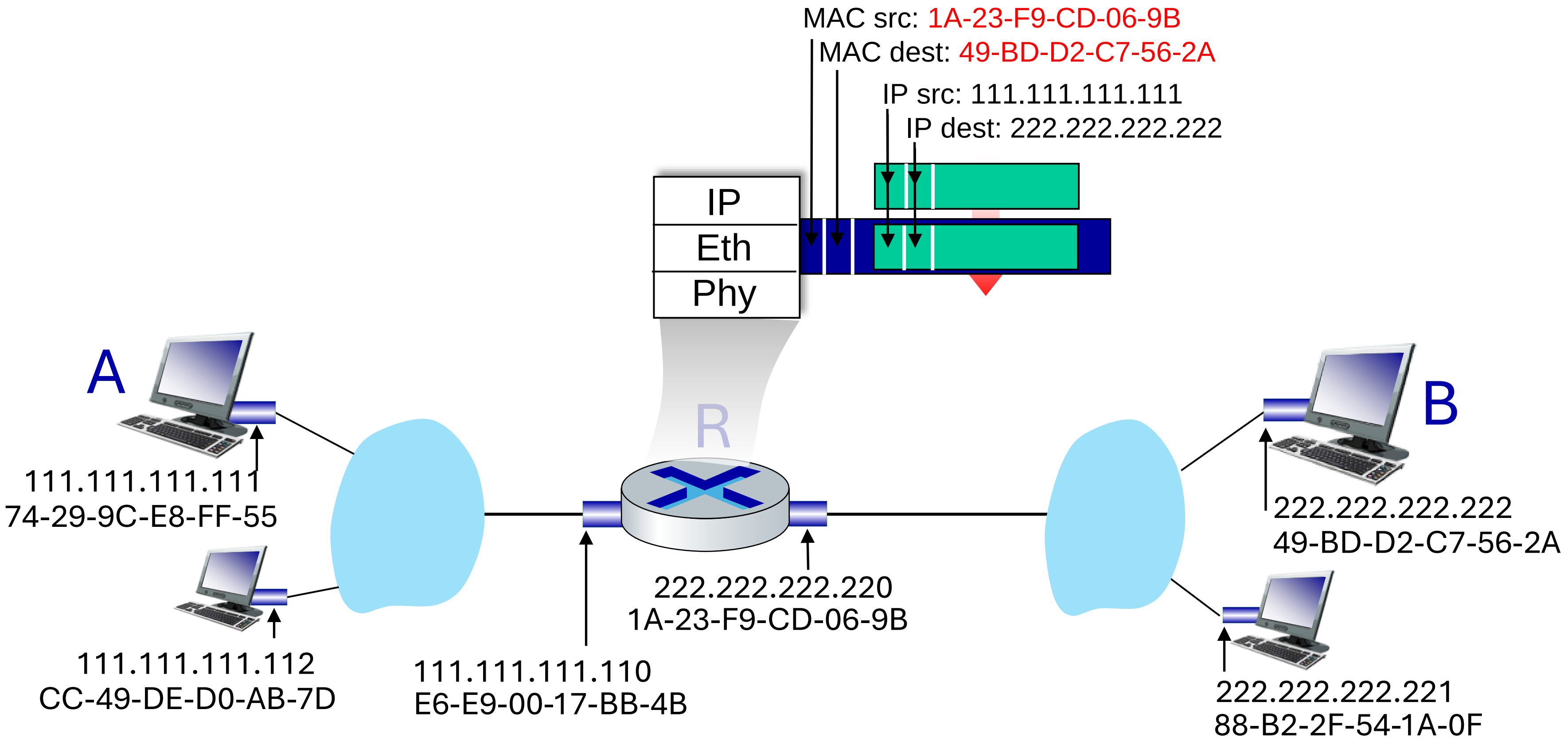
Routing to another subnet: addressing

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



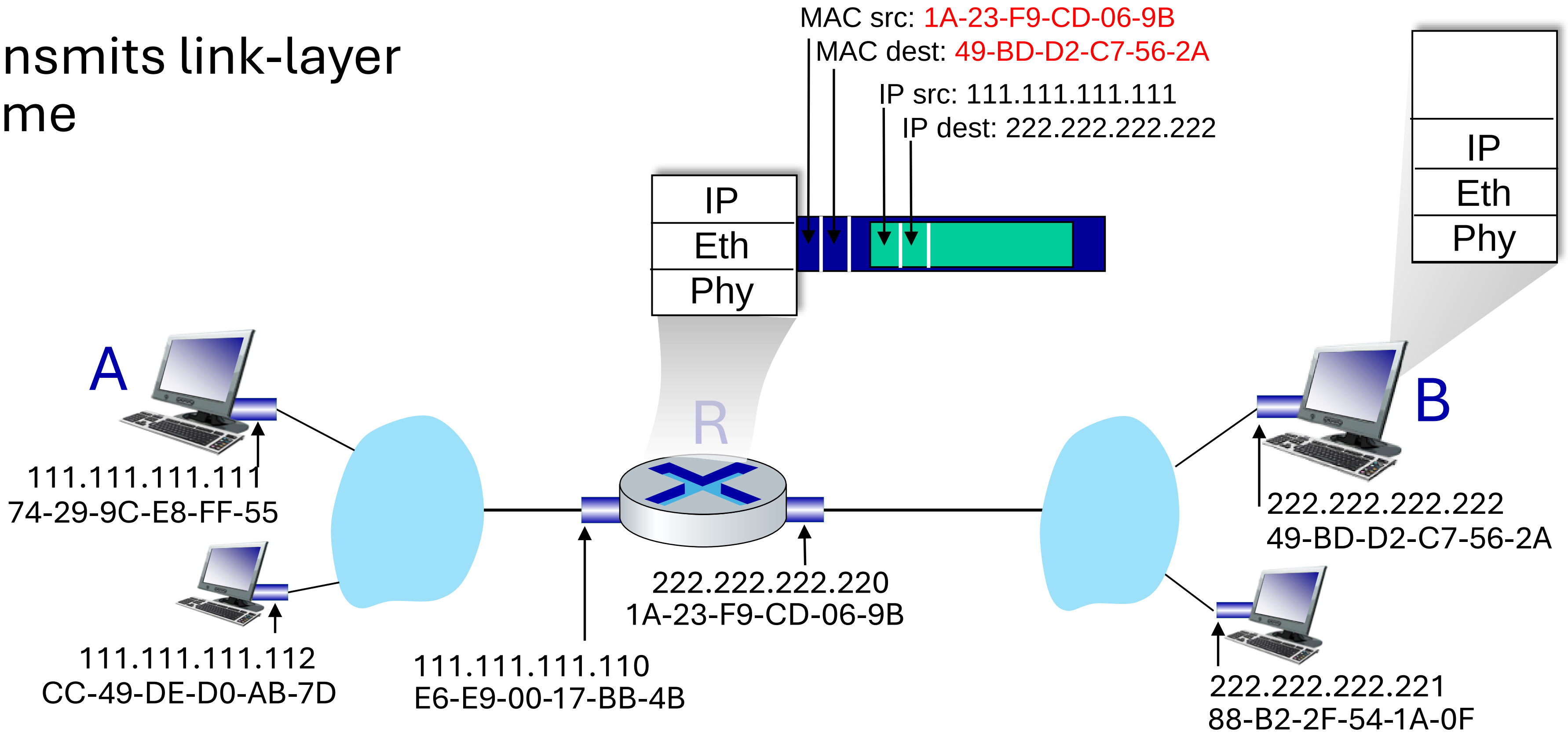
Routing to another subnet: addressing

- R determines outgoing interface, passes datagram with IP source A, destination B to link layer
- R creates link-layer frame containing A-to-B IP datagram. Frame destination address: B's MAC address



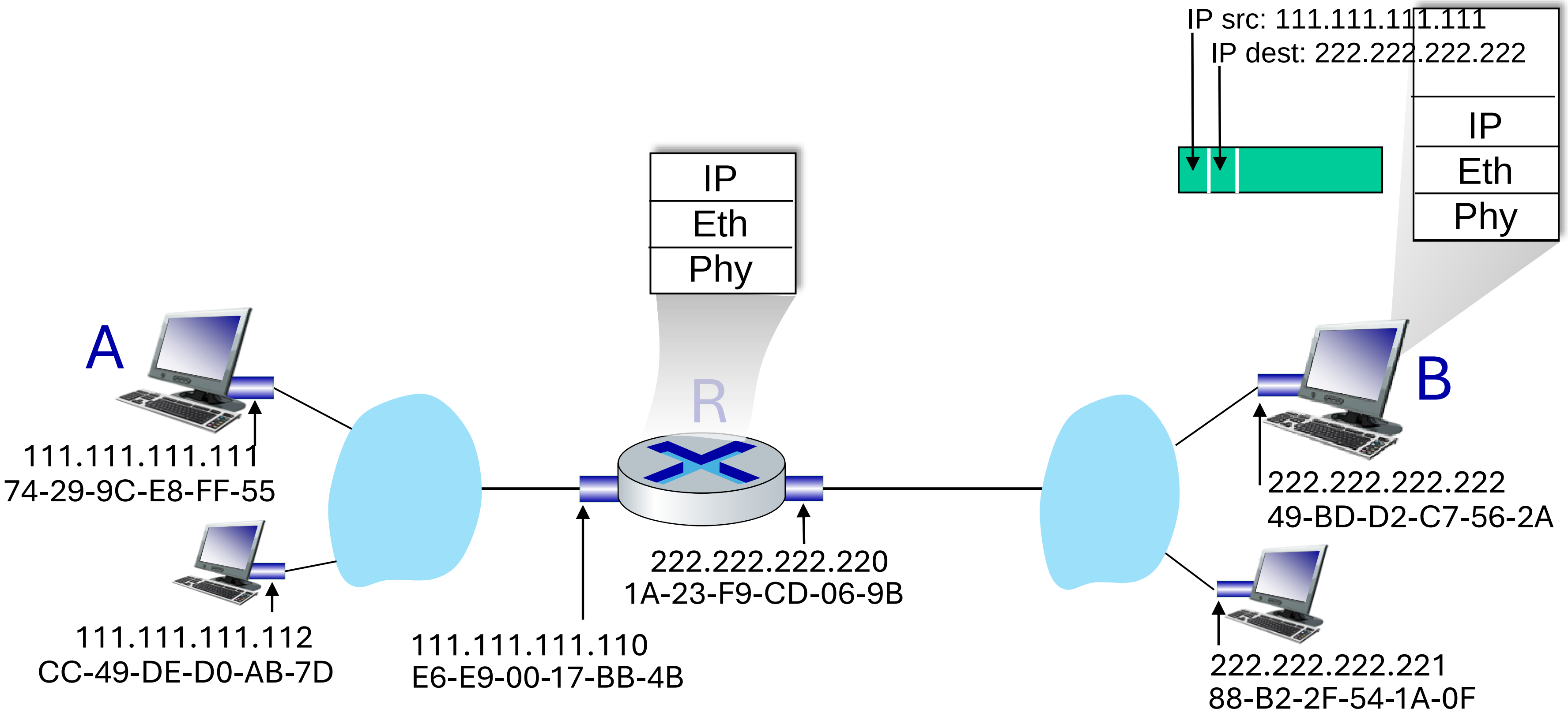
Routing to another subnet: addressing

- R determines outgoing interface, passes datagram with IP source A, destination B to link layer
- R creates link-layer frame containing A-to-B IP datagram. Frame destination address: B's MAC address
- transmits link-layer frame



Routing to another subnet: addressing

- B receives frame, extracts IP datagram destination B
- B passes datagram up protocol stack to IP

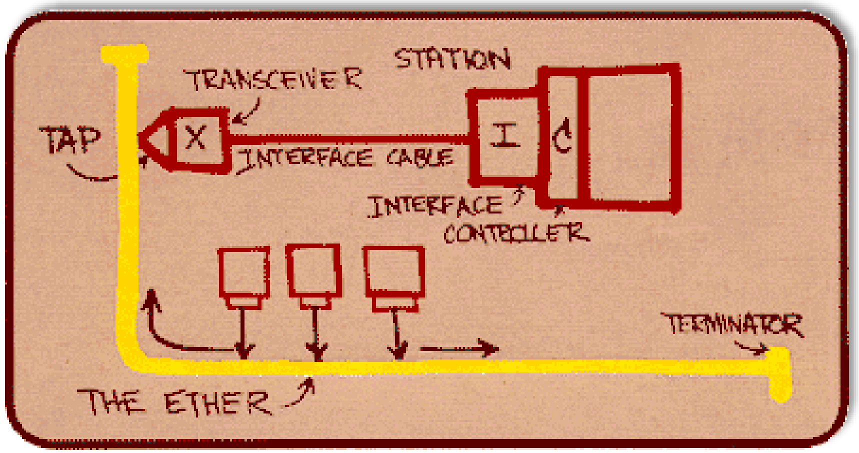


Optional

Ethernet

"dominant" wired LAN technology:

- first widely used LAN technology
- simpler, cheap
- kept up with speed race: 10 Mbps – 400 Gbps
- single chip, multiple speeds (e.g., Broadcom BCM5761)



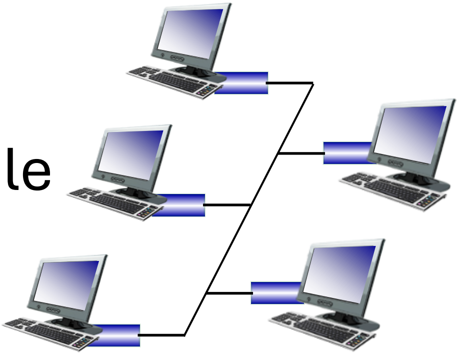
Metcalfe's Ethernet sketch

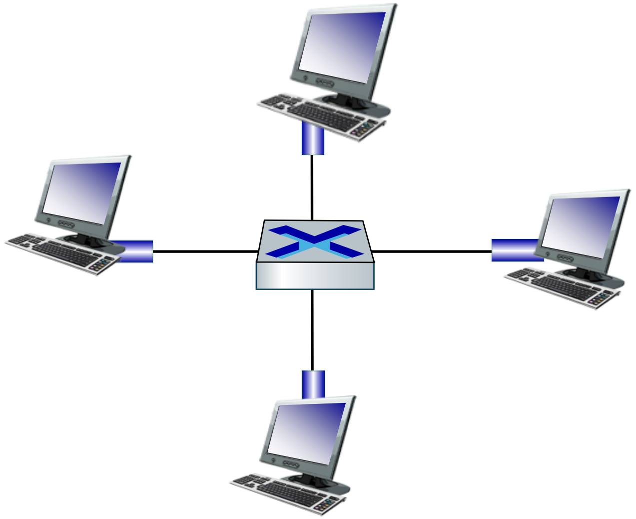
https://www.uspto.gov/learning-and-resources/journeys-innovation/audio-stories/defying-doubters

Ethernet: physical topology

- bus: popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- switched: prevails today
  - active link-layer 2 *switch* in center
  - each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other)

bus: coaxial cable





switched

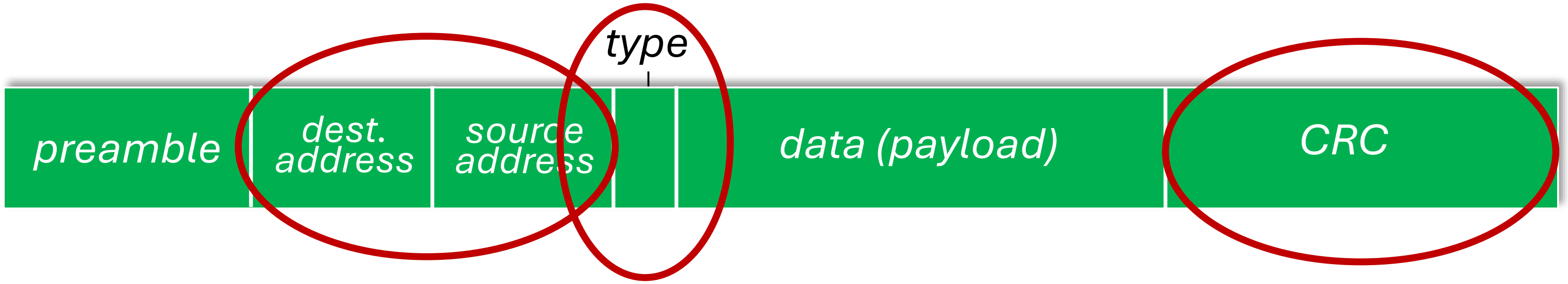
Ethernet frame structure

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

preamble:

- used to synchronize receiver, sender clock rates
- 7 bytes of 10101010 followed by one byte of 10101011

Ethernet frame structure (more)



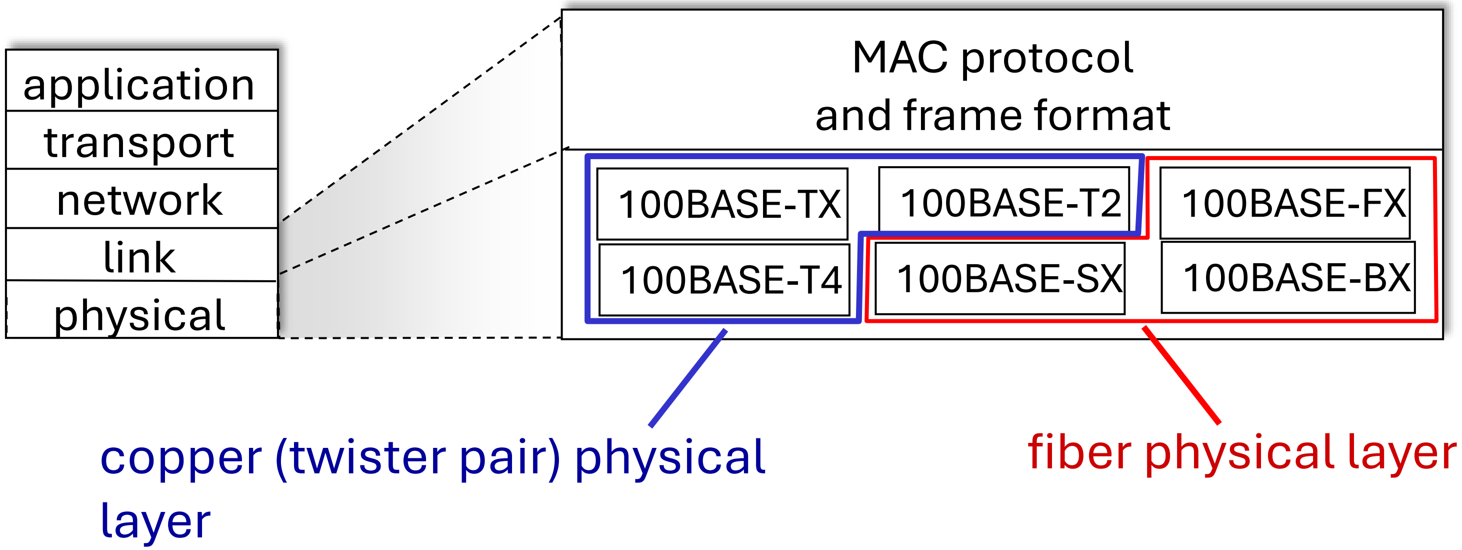
- addresses: 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (e.g., ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- type: indicates higher layer protocol
  - mostly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

Ethernet: unreliable, connectionless

- connectionless: no handshaking between sending and receiving NICs
- unreliable: receiving NIC doesn’t send ACKs or NAKs to sending NIC
  - data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost
- Ethernet’s MAC protocol: unslotted CSMA/CD with binary backoff

802.3 Ethernet standards: link & physical layers

- *many* different Ethernet standards
  - common MAC protocol and frame format
  - different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10 Gbps, 40 Gbps
  - different physical layer media: fiber, cable

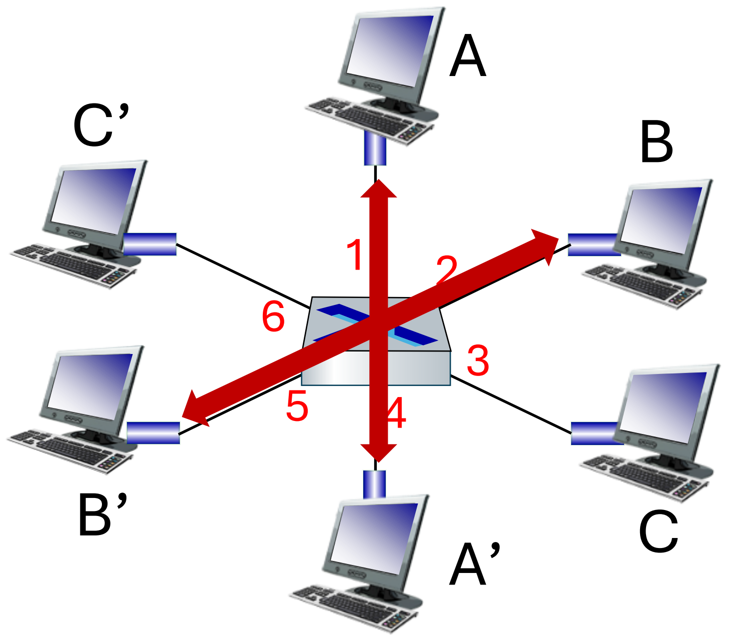


Ethernet switch

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

Switch: multiple simultaneous transmissions

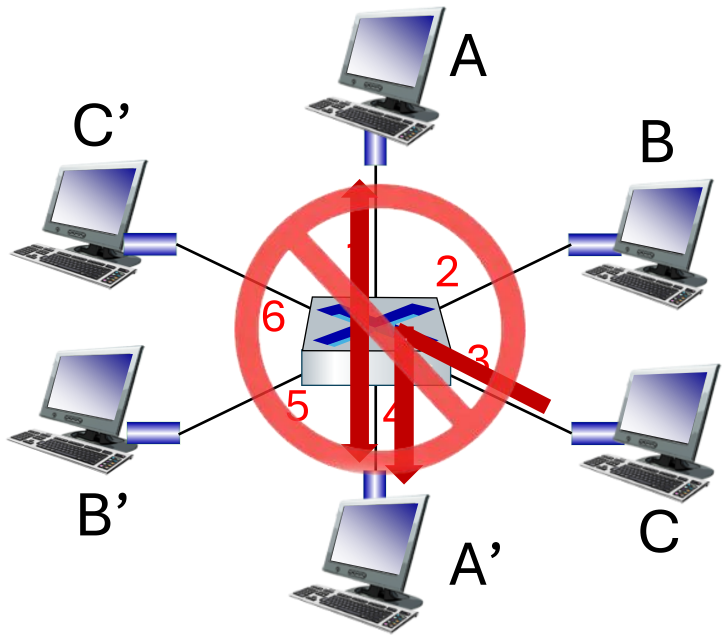
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
  - no collisions; full duplex
  - each link is its own collision domain
- switching: A-to-A' and B-to-B' can transmit simultaneously, without collisions



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

Switch: multiple simultaneous transmissions

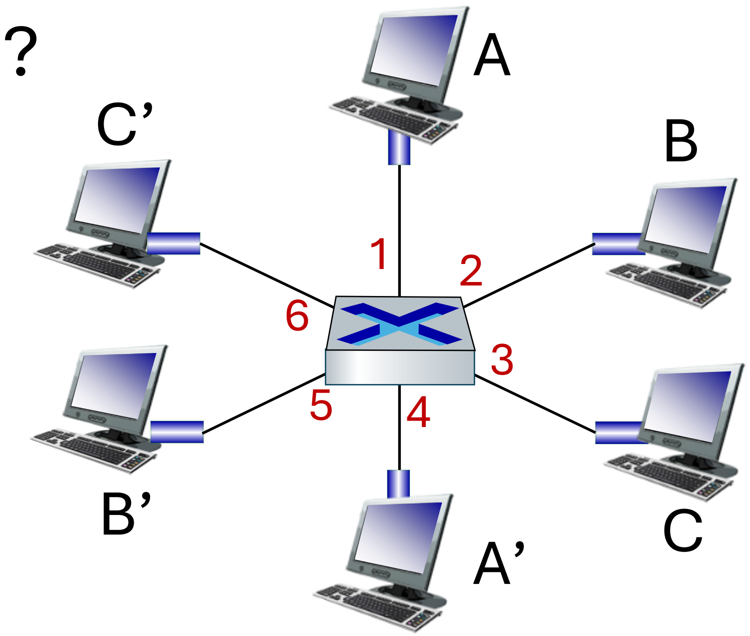
- switching: A-to-A’ and B-to-B’ can transmit simultaneously, without collisions
  - but A-to-A’ and C to A’ can *not* happen simultaneously



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

Switch forwarding table

Q: how does switch know A’ reachable via interface 4, B’ reachable via interface 5?



A: each switch has a switch table, each entry:

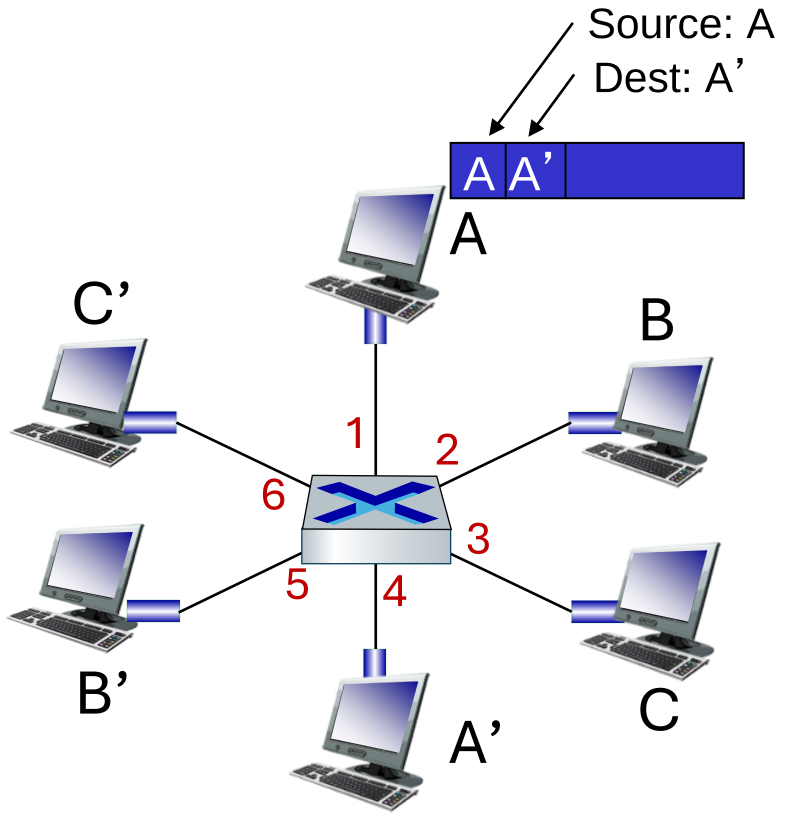
- (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!

Q: how are entries created, maintained in switch table?

- something like a routing protocol?

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

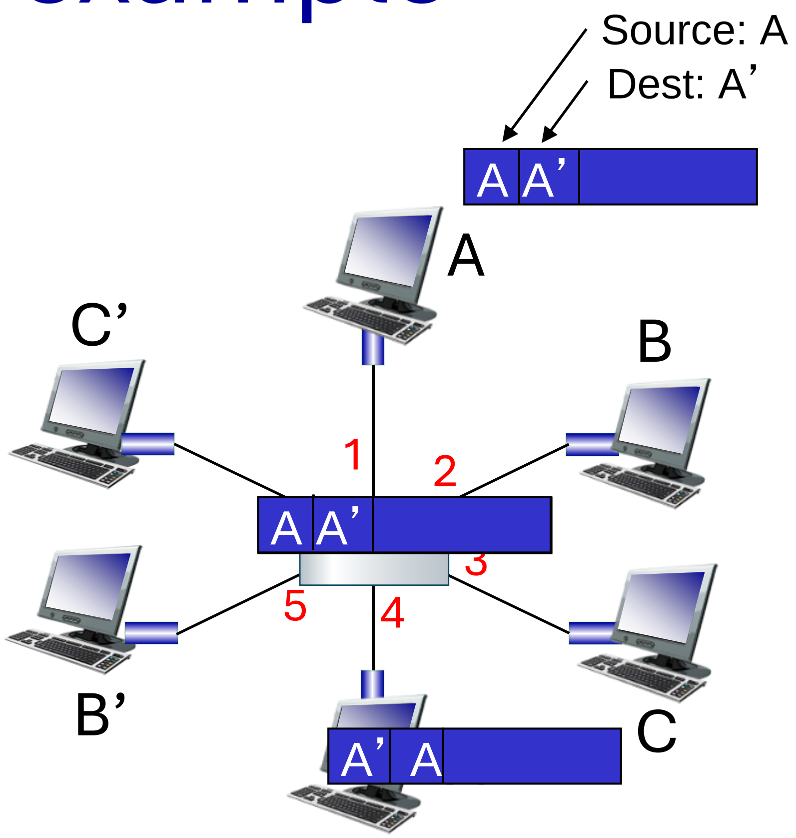


MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

Self-learning, forwarding: example

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

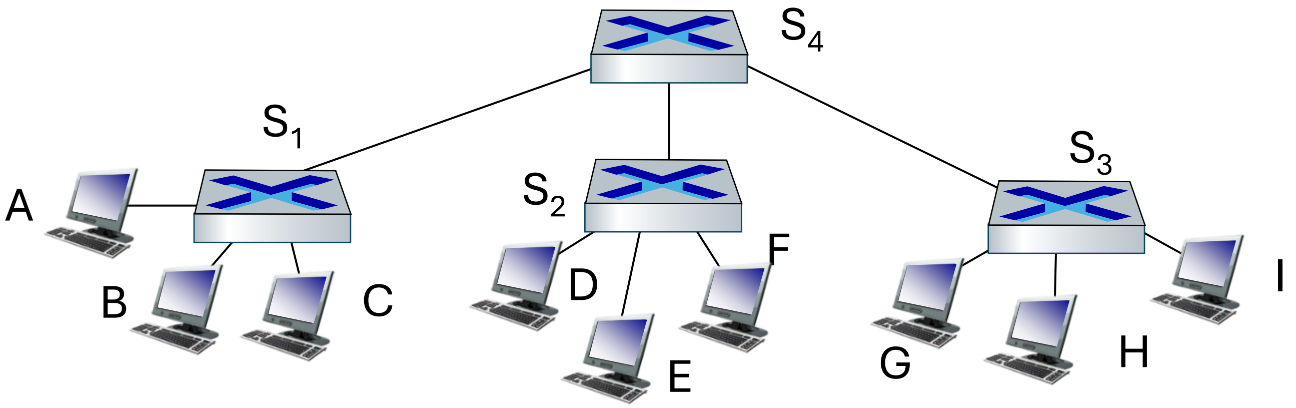


MAC addr	interface	TTL
A	*1*	*60*
A’	*4*	*60*

switch table (initially empty)

Interconnecting switches

self-learning switches can be connected together:

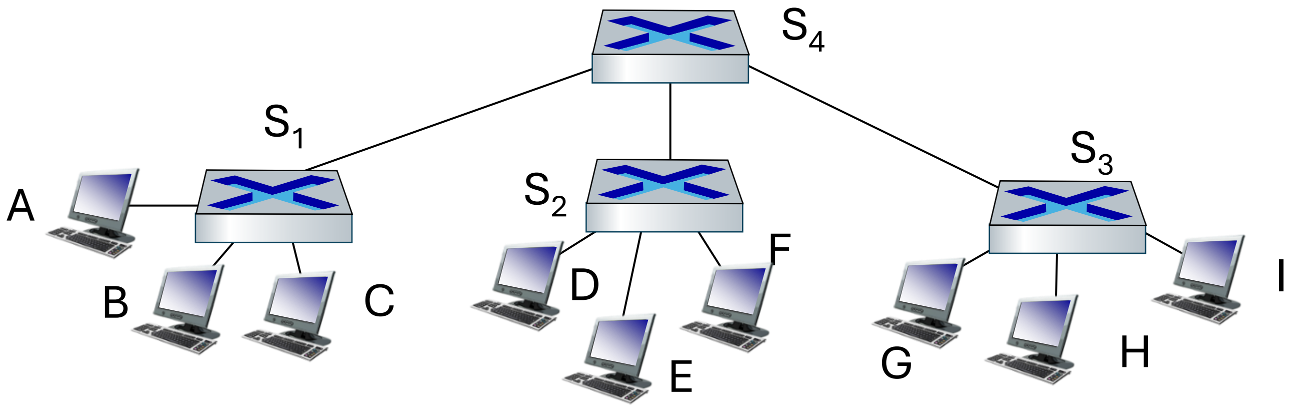


Q: sending from A to G - how does S_1 know to forward frame destined to G via S_4 and S_3?

- *A:* self learning! (works exactly the same as in single-switch case!)

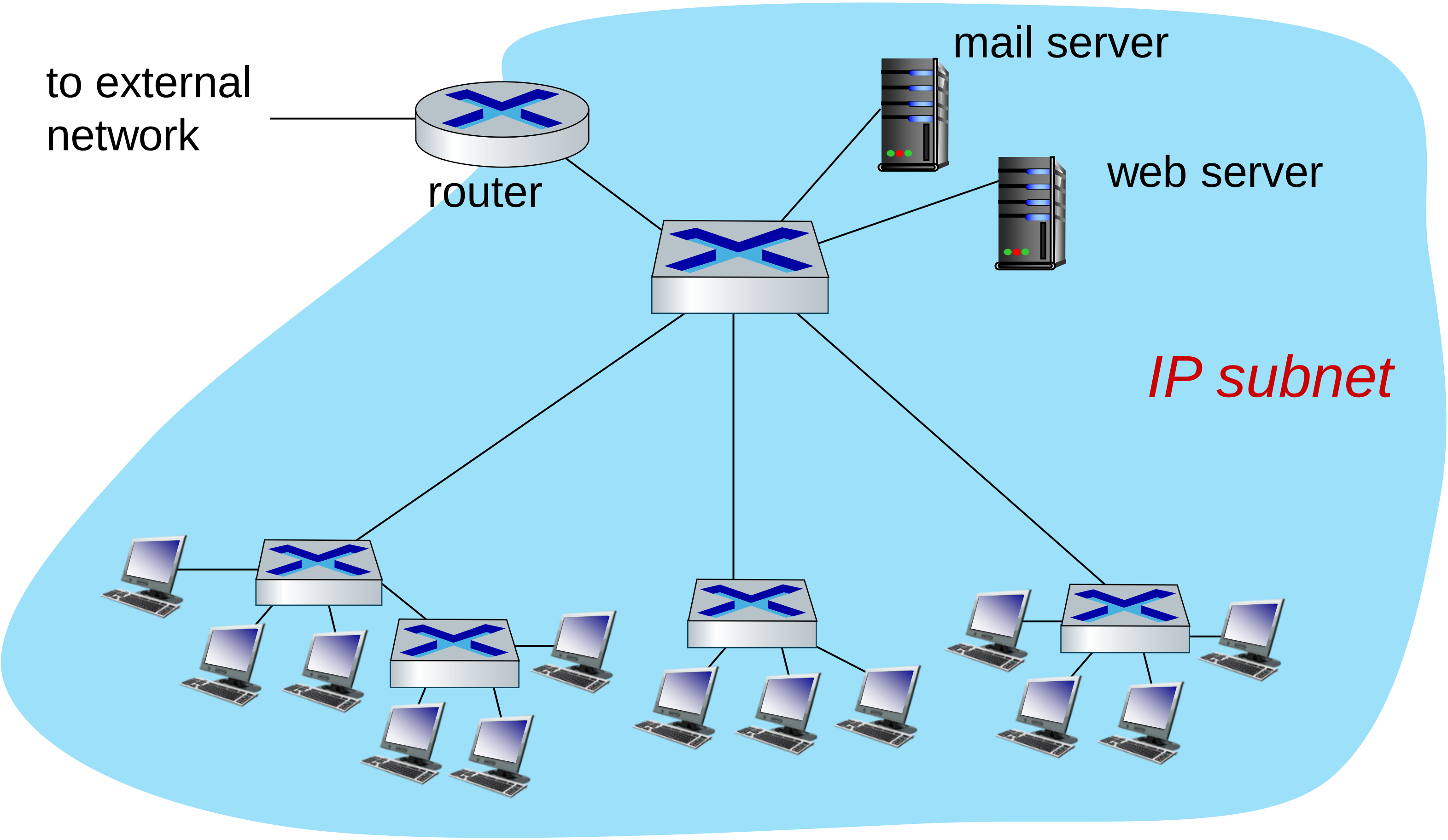
Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4

Small institutional network



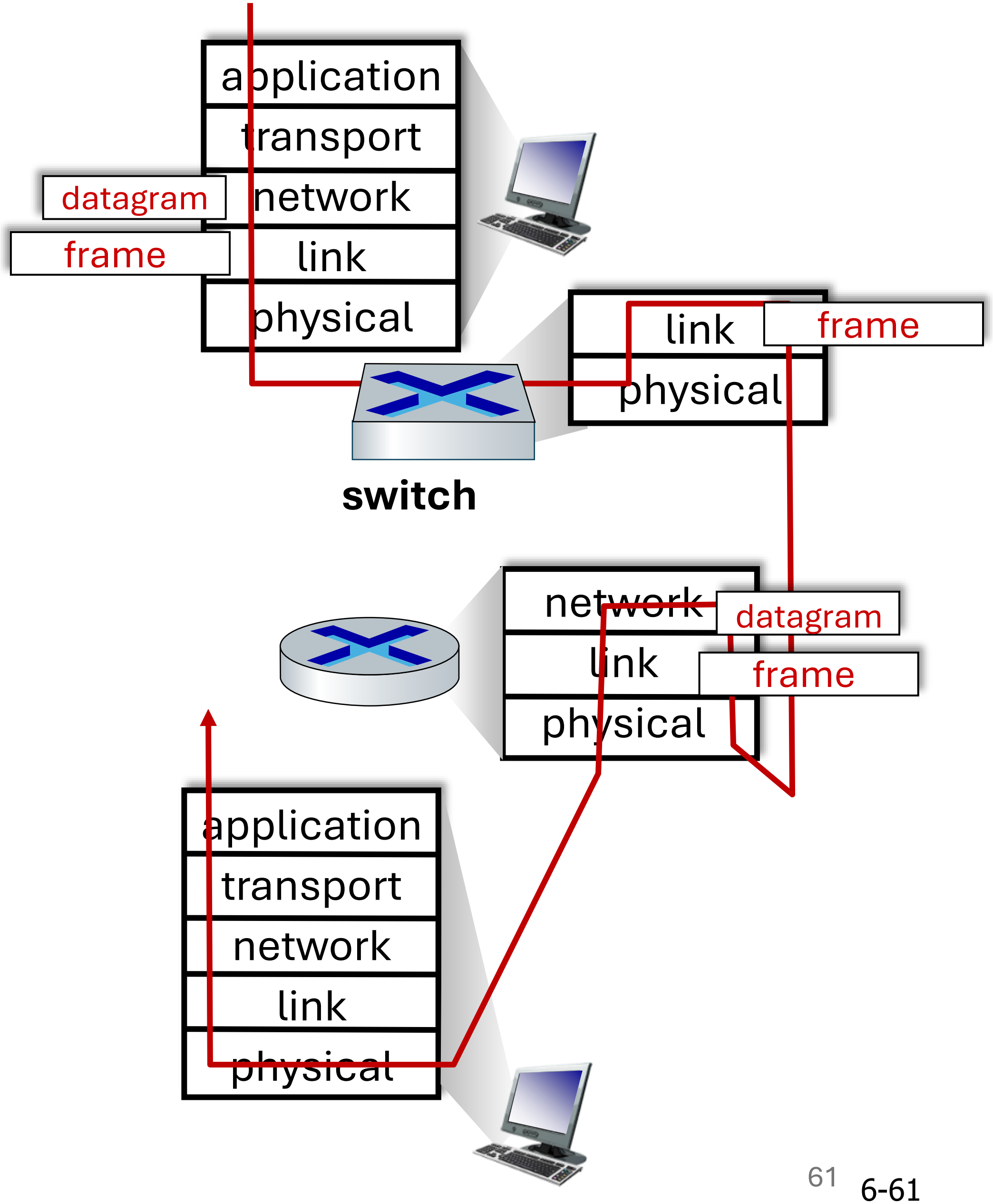
Switches vs. routers

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



Thanks for listening!
Any questions?

Acknowledgment

- James F. Kurose University of Massachusetts, Amherst
- Keith W. Ross NYU and NYU Shanghai