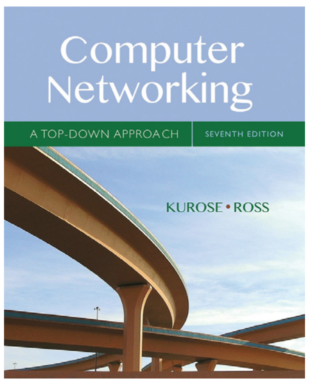
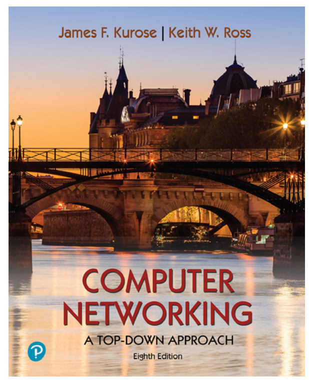
Chapter 6
The Link Layer and LANs

Courtesy to the textbooks' authors and Pearson Addison-Wesley because many slides are adapted from the following textbooks and their associated slides.



Jim Kurose, Keith Ross, "Computer Networking: A Top Down Approach", 7th Edition, Pearson, 2016.



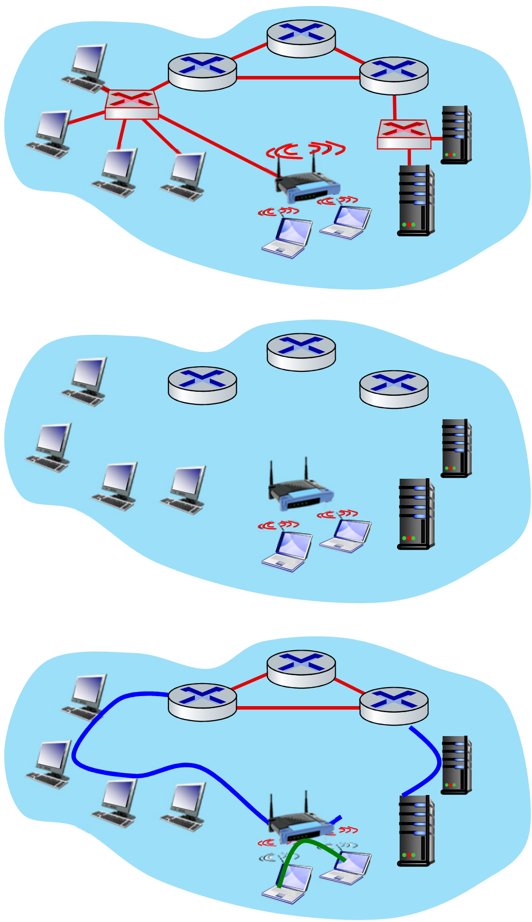
Jim Kurose, Keith Ross, "Computer Networking: A Top Down Approach", 8th Edition, Pearson, 2020.

Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- link virtualization: MPLS
- data center networking
- a day in the life of a web request

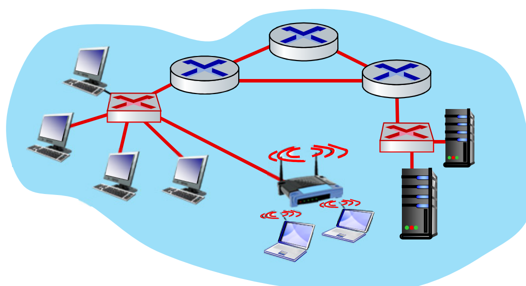
Link layer: introduction

- Terminology
  - host and router are ≥L3 nodes
    - switch (and bridge) is layer-2
  - frame is layer-2 packet
- link layer is responsible to transfer packets from one node to another via a link (within a subnet)
  - either wired or wireless link
  - LAN



Link layer: context

- datagram transferred by different link-layer protocols over different links:
  - e.g., WiFi on first link, Ethernet on next link

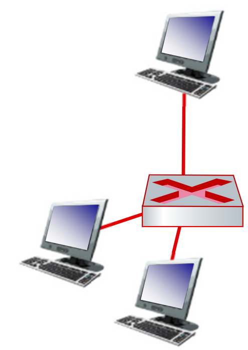


- each link protocol provides different services
  - e.g., may or may not provide reliable data transfer over link

Link layer: services

- framing
  - encapsulate datagram into frame
    - adding header, trailer
  - "MAC" addresses in frame headers identify source, destination
    - MAC address is different from IP address!
- channel access (multiple access, MAC)
  - if shared medium
- reliable delivery in link layer
  - we already know how to do this!
  - seldom used on low bit-error links
  - wireless links: high error rates
  - *Q:* why both link-level and end-to-end reliability?

datagram → H_{link} | datagram | T_{link}

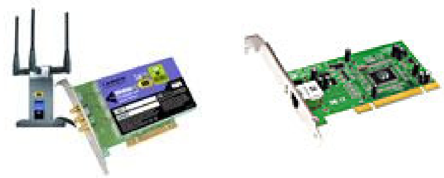


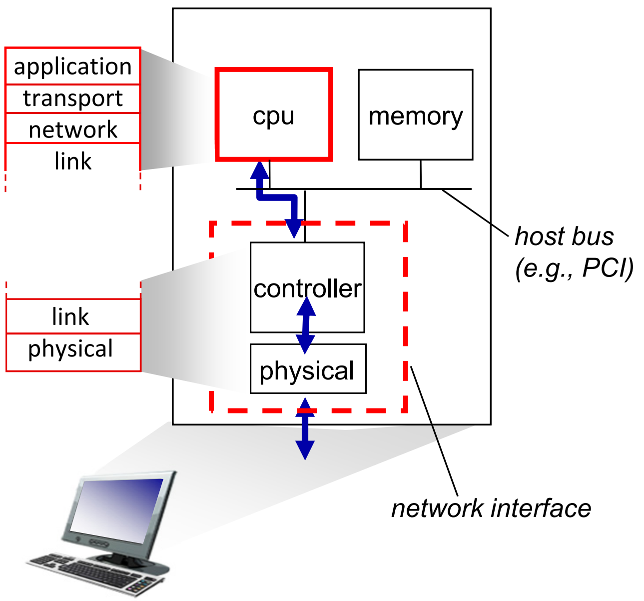
Link layer: services (more)

- flow control:
  - pacing between adjacent sending and receiving nodes
- error detection:
  - errors caused by signal attenuation, noise, collision.
  - retransmits or drops frame, if receiver detects errors
- 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

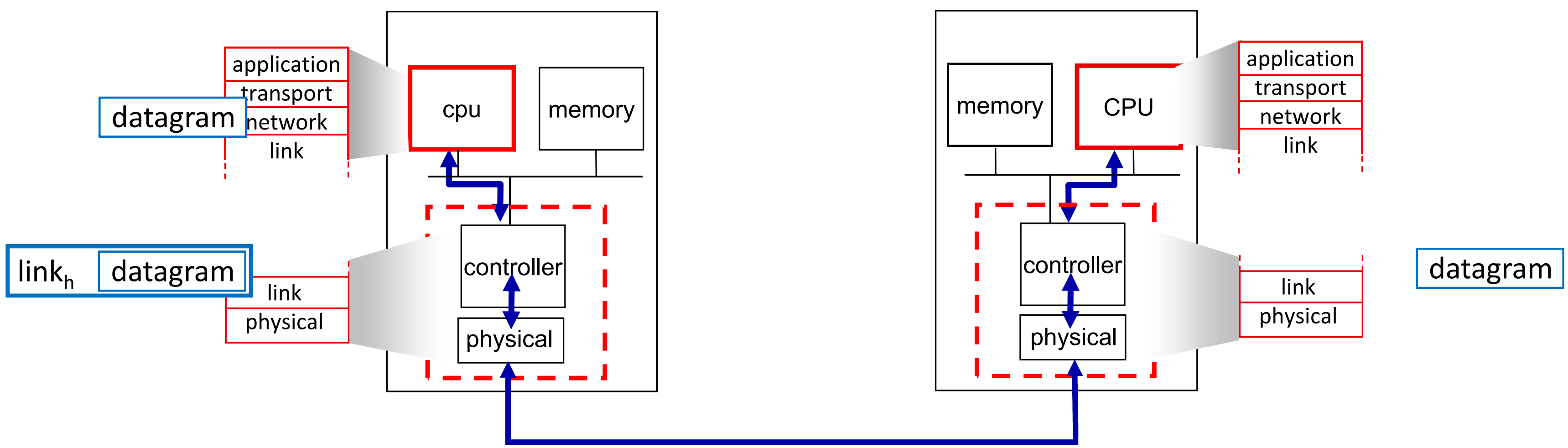
Where is the link layer implemented?

- in each host
- link layer implemented in *network interface card* (NIC)
  - (Ethernet, WiFi) card or chip
  - implements link+physical layers
- attaches into host's system bus
- combination of hardware, software, firmware





Interfaces communicating



sending side:

- encapsulates datagram in frame
- adds error checking bits, reliable data transfer, flow control, etc.

receiving side:

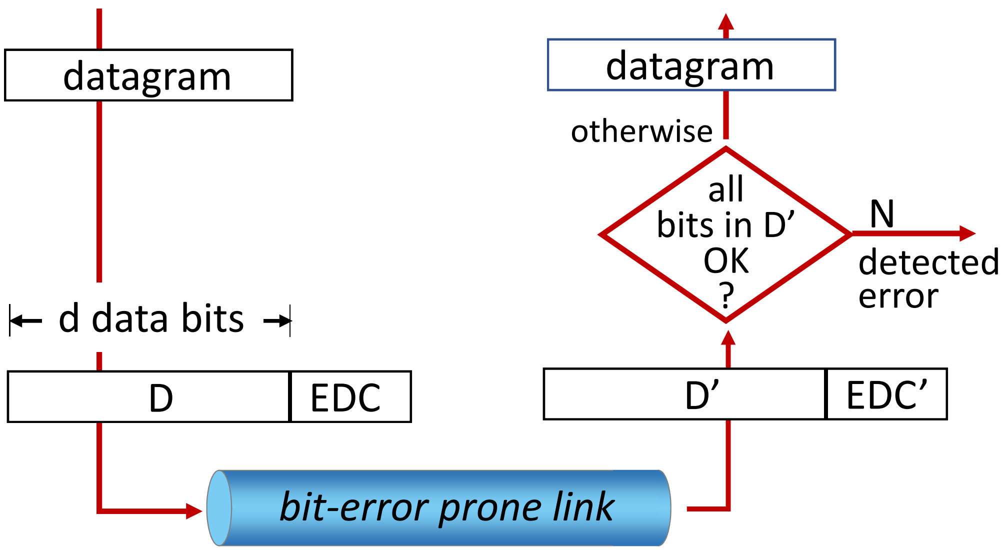
- looks for errors, reliable data transfer, flow control, etc.
- extracts datagram, passes to upper layer at receiving side

Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- link virtualization: MPLS
- data center networking
- a day in the life of a web request

Error detection

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



Error detection has its capability

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

Parity checking

single bit parity:

- detect single bit errors

0111000110101011	1

$\longleftarrow$ d data bits $\longrightarrow$ | parity bit

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

two-dimensional bit parity:

- detect *and correct* single bit errors

row parity $\longrightarrow$

column parity $\downarrow$

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

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

parity error

UDP/TCP/IP checksum (review)

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

sender:

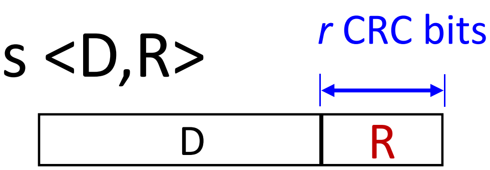
- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

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 ….

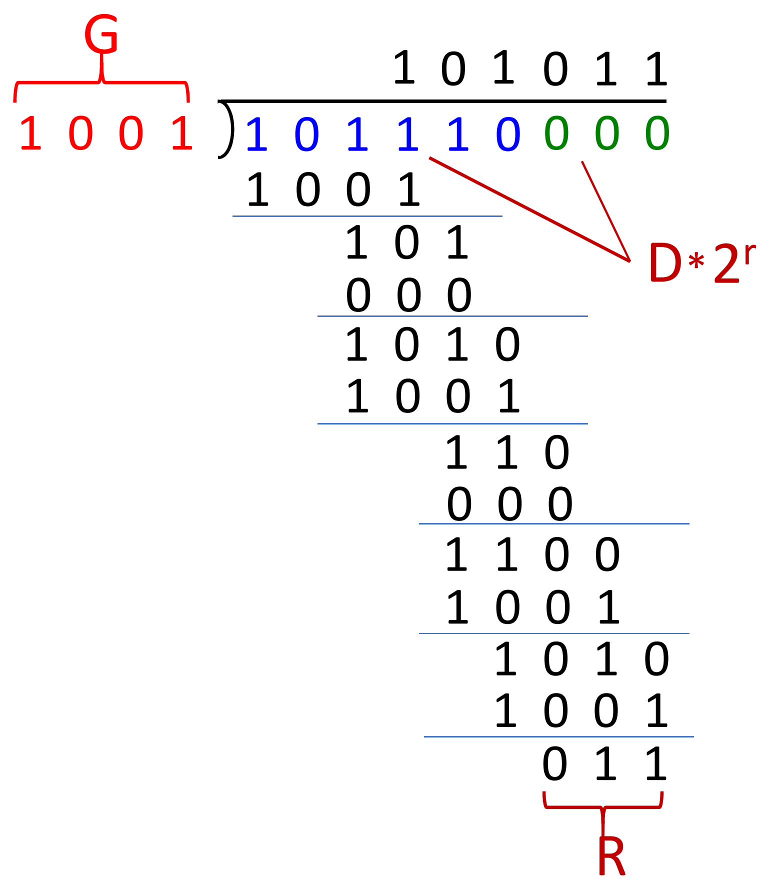
Cyclic Redundancy Check (CRC)

- error-detection code more powerful than TCP/UDP/IP's checksum
  - used by Ethernet and WiFi
  - can detect all burst errors less than r+1 bits
- both sender and receiver know G in advance:
  - G: generator of length r+1 bits
- sender computes the CRC bits R and transmits <D,R>
  - D: data bits
  - $R = D\cdot 2^r \ \% \ G = \langle D,\overbrace{00\ldots0}^{r \text{ CRC bits}}\rangle \ \% \ G$ is the remainder
    - the CRC bits R is of length r bits
    - use bitwise-XOR for addition/subtraction
    - <D,R> is divisible by G
- receiver checks out whether <D,R> % G = 0
  - non-zero remainder $\rightarrow$ error detected
  - zero remainder $\rightarrow$ either no error or error not detected



Cyclic Redundancy Check (CRC): example

- G = 1001 and D = 101110
  - in binary representations
  - *r* = 3 in this example
- sender computes the remainder R
  - R = D·2^r % G
    = <D,00…0> % G



Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- link virtualization: MPLS
- data center networking

- a day in the life of a web request

Multiple access links, protocols

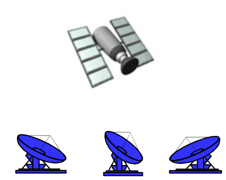
two types of "links":

- point-to-point
  - point-to-point link between Ethernet switch and host
- broadcast (shared wire or medium)
  - old-fashioned Ethernet
  - 802.11 wireless LAN, mobile networks (4G/5G), 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 (or centralized) 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 capacity *R* bps

desiderate:

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 single point of failure
   - no synchronization of clocks, slots
4. simple

MAC protocols: taxonomy

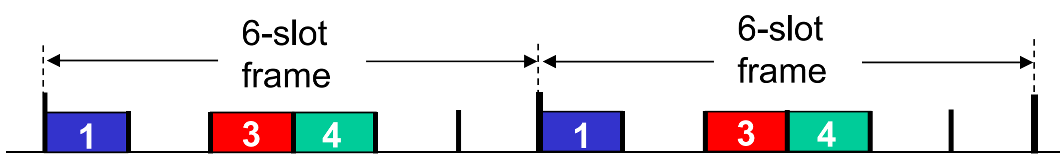
three broad classes:

- channel partitioning
  - divide channel into smaller “pieces” (time slots, frequency, …)
  - allocate piece to node/user 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
    - similar to round robin

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

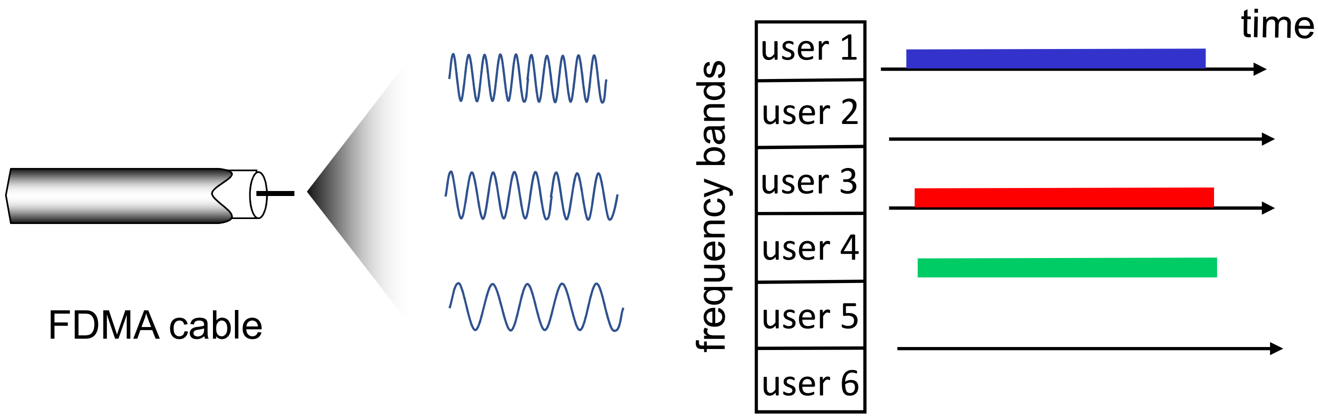
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



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

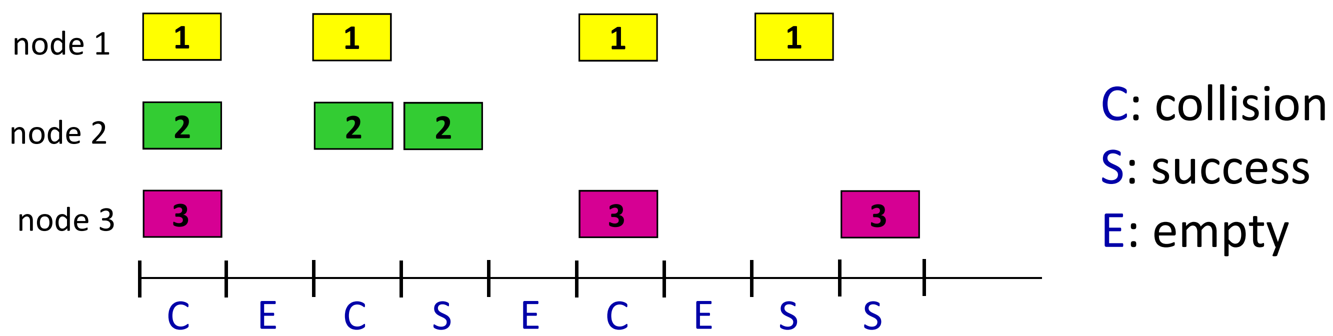
assumptions:

- time is divided into equal-sized slots
  - slot: time to transmit 1 frame
    - all frames have the same size
  - nodes are time-synchronized
  - nodes start to transmit only slot beginning
- if 2 or more nodes transmit in a slot, all nodes detect collision

operation:

- stop-and-wait
- 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



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
- 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

- *suppose:* Given N nodes, each node has many frames to send and transmits in every slot with probability p
  - probability that a given node has success in a slot
  - probability that *any* node has a success = $Np(1-p)^{N-1}$

Slotted ALOHA: efficiency (cont'd)

- slotted ALOHA's efficiency = probability that *any* node has a success
  - Given N and p, $f(p) = Np(1-p)^{N-1}$
- max efficiency: find p^* that maximizes $f(p)$

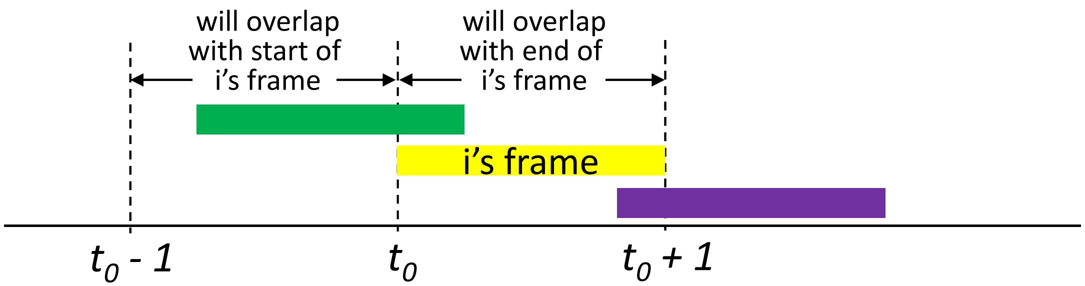
- for many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity:

$$\lim_{N\to\infty} f(p^*) = \lim_{N\to\infty} N \cdot \frac{1}{N} \cdot \left(1 - \frac{1}{N}\right)^{N-1} = \lim_{N\to\infty} \left(\frac{1}{1+\frac{1}{N-1}}\right)^{N-1} = \frac{1}{e}$$

- slotted ALOHA's efficiency is at best 1/*e* = 37%
  - 37% of time is useful transmissions.

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)

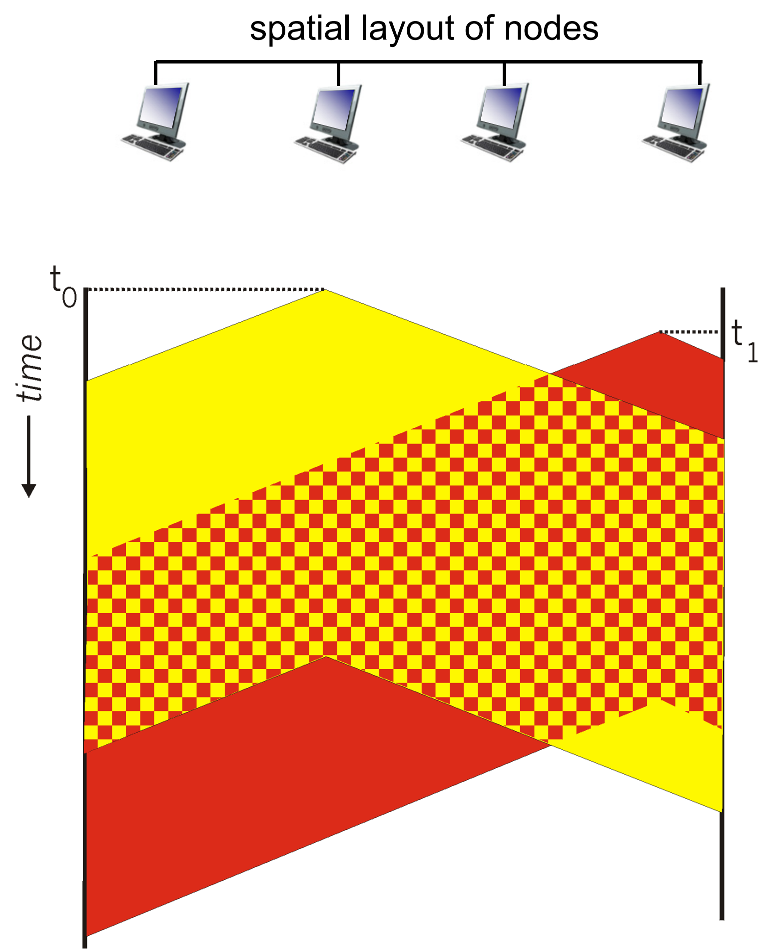
simple CSMA: listen before transmit:

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

- human analogy: don't interrupt others!

CSMA: collisions

- collisions *can* still occur if two nodes send frames around the same time
  - 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

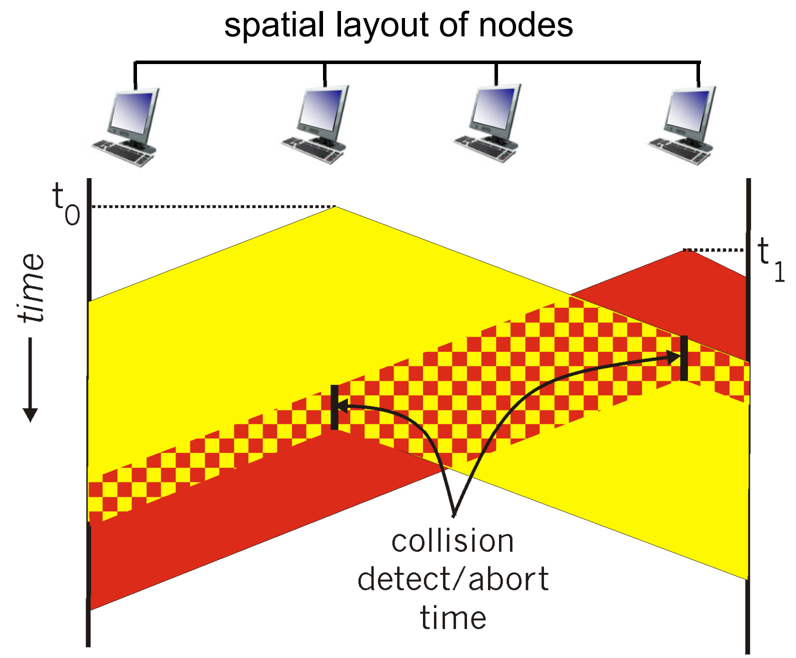
CSMA/CD: CSMA with *collision detection*

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

- human analogy: the polite conversationalist

CSMA/CD

- CSMA/CS reduces the amount of time wasted in collisions
  - transmission aborted on collision detection



Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. 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* slot times, returns to Step 2. (1 slot = minimum frame size = 64 bytes for 10/100 Mbps or 512 bytes for 1Gbps)
   - more collisions: longer backoff interval

CSMA/CD efficiency

- $t_{prop} \triangleq$ maximum propagation delay between 2 nodes in LAN
- $t_{trans} \triangleq$ time to transmit max-size frame
- efficiency is (approximately)

$$\rho \doteqdot \frac{1}{1 + 5 \cdot \frac{t_{prop}}{t_{trans}}}$$

  - efficiency goes to 1 as t_{prop}/t_{trans} goes to 0
- CSMA/CD
  - better efficiency than ALOHA
  - simple, cheap, decentralized

“Taking turns” MAC protocols

channel partitioning MAC protocols:

- share channel *efficiently* and *fairly* at high load
- inefficient at low load: 1/N bandwidth allocated even if only 1 active node!

random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

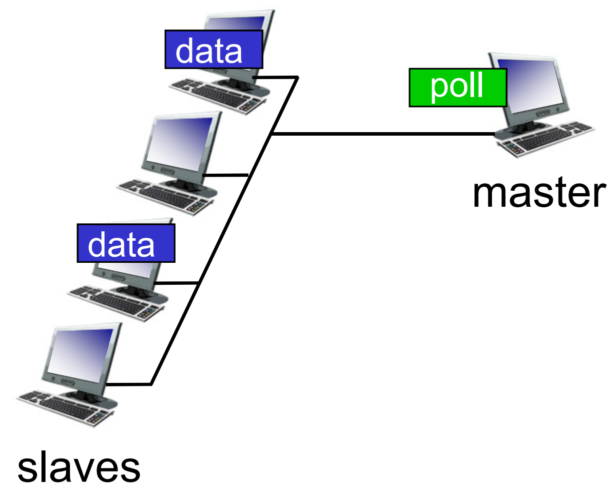
“taking turns” protocols

- look for best of both worlds!

“Taking turns” MAC protocols

polling:

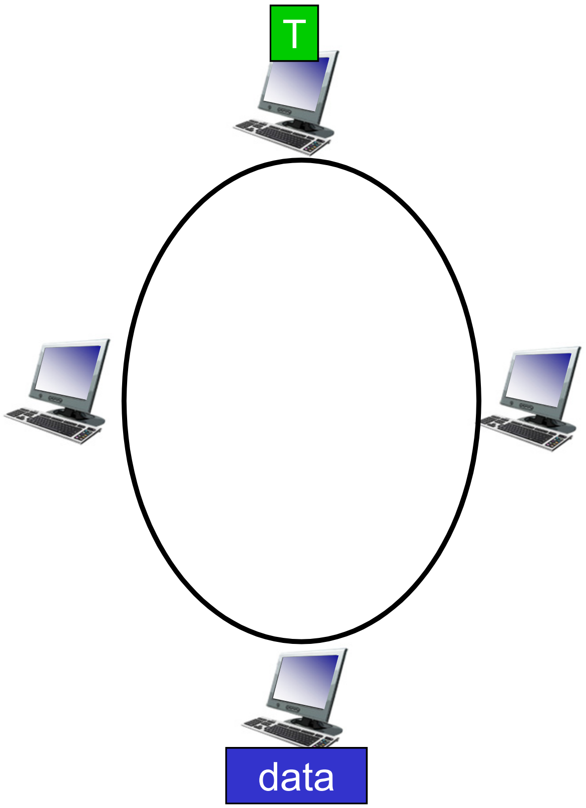
- master node “invites” other nodes to transmit in turn
- typically used with “dumb” devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



“Taking turns” MAC protocols

token passing:

- control *token* passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)



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 (wired), 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

Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- link virtualization: MPLS
- data center networking

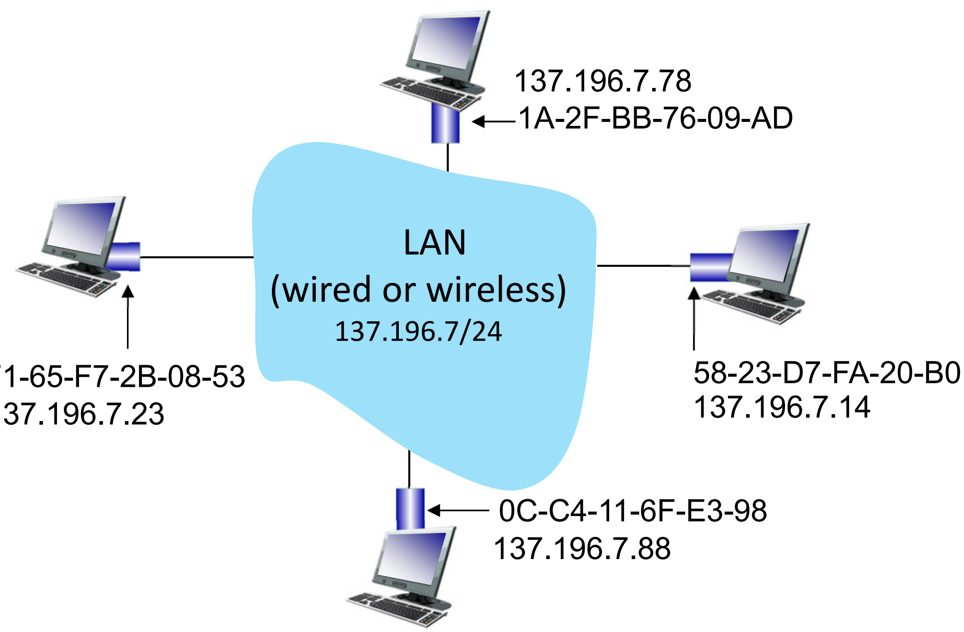
- a day in the life of a web request

MAC addresses

- IP address:
  - *network-layer* address for interface
    - used for layer-3 (network layer) forwarding
      - switch doesn't have any IP address
  - 32-bit (in IPv4)
    - 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 in link layer (within same subnet)
  - 48-bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
    - e.g.: 1A-2F-BB-76-09-AD (or 1A:2F:BB:76:09:AD)

Each interface at host or at router

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

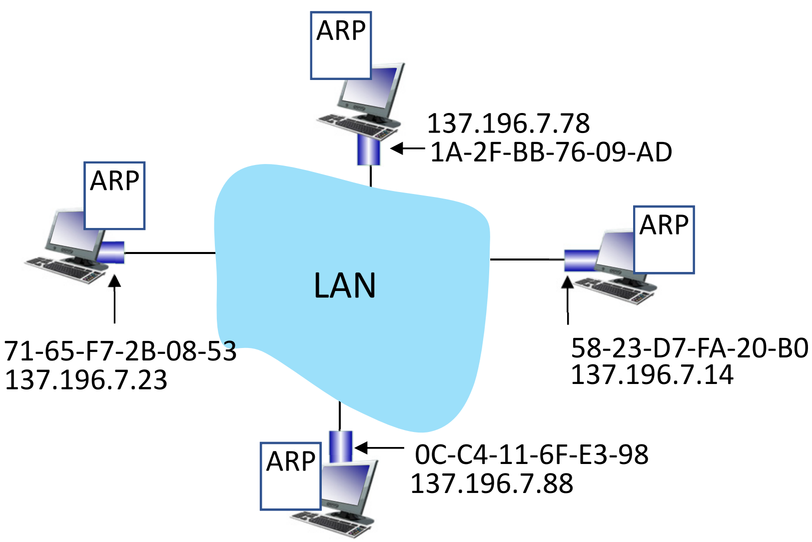


MAC addresses

- 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 LAN
  - 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 (layer-3) 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

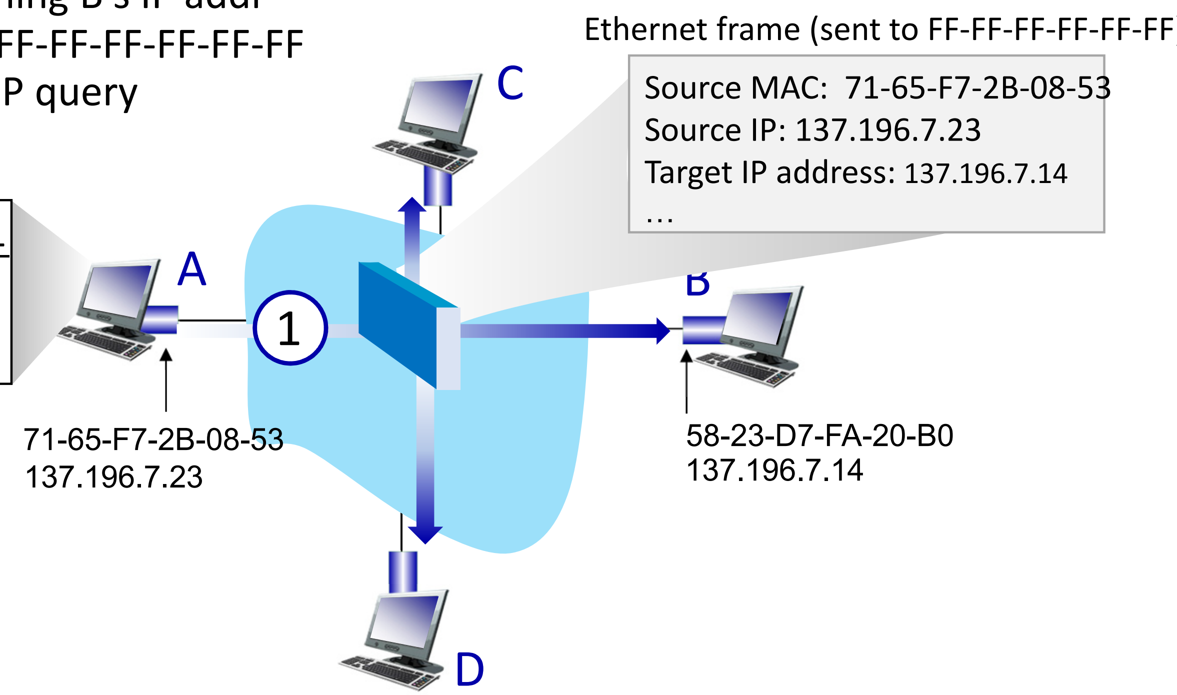
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

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

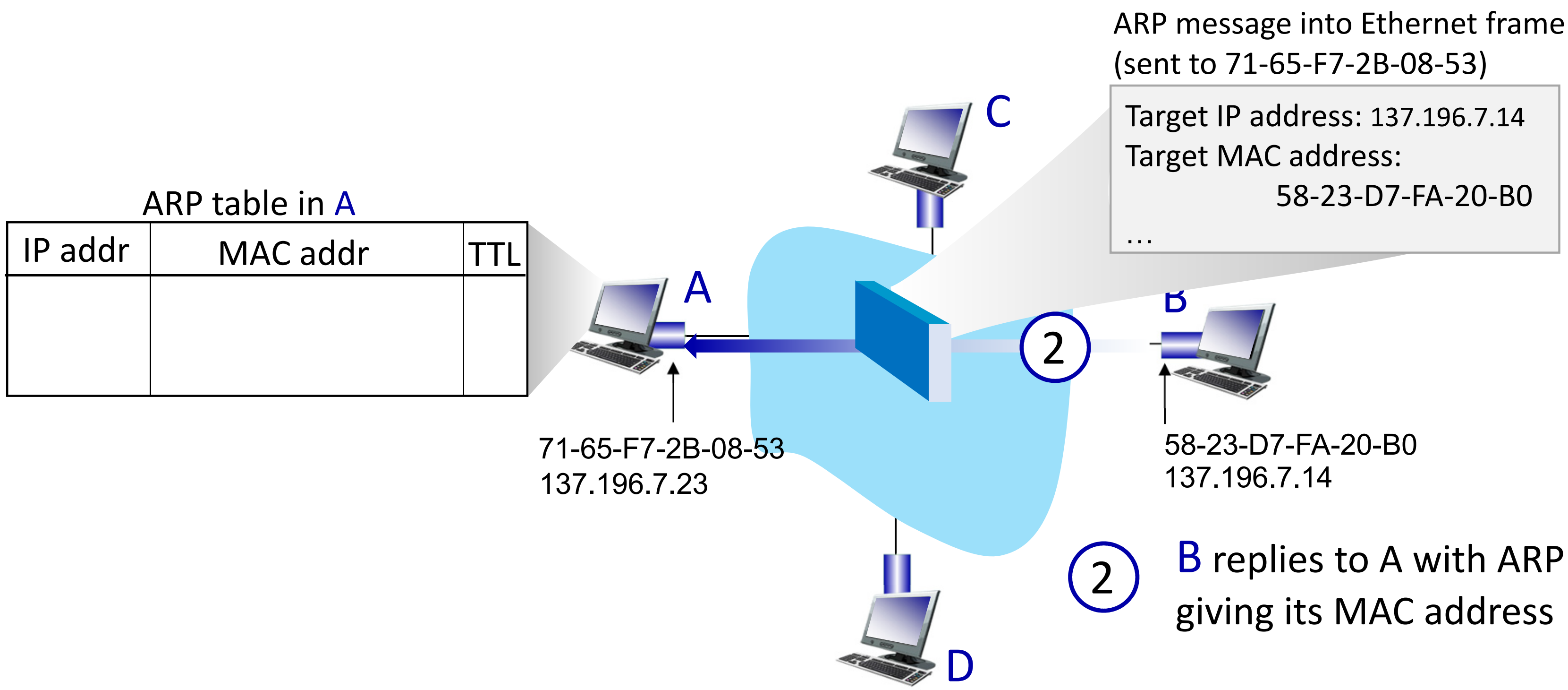
IP addr	MAC addr	TTL



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



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

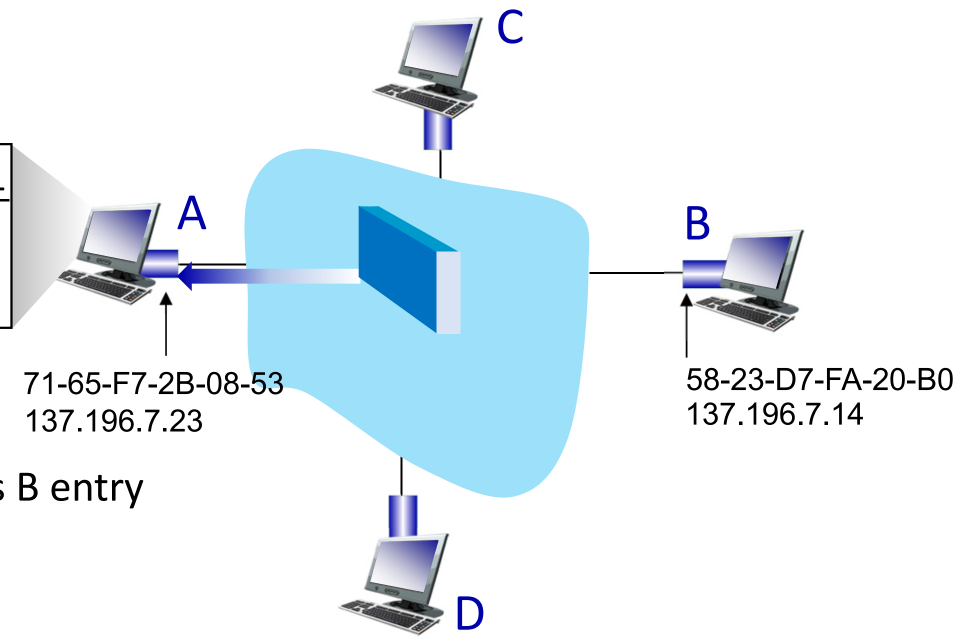
ARP protocol in action

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

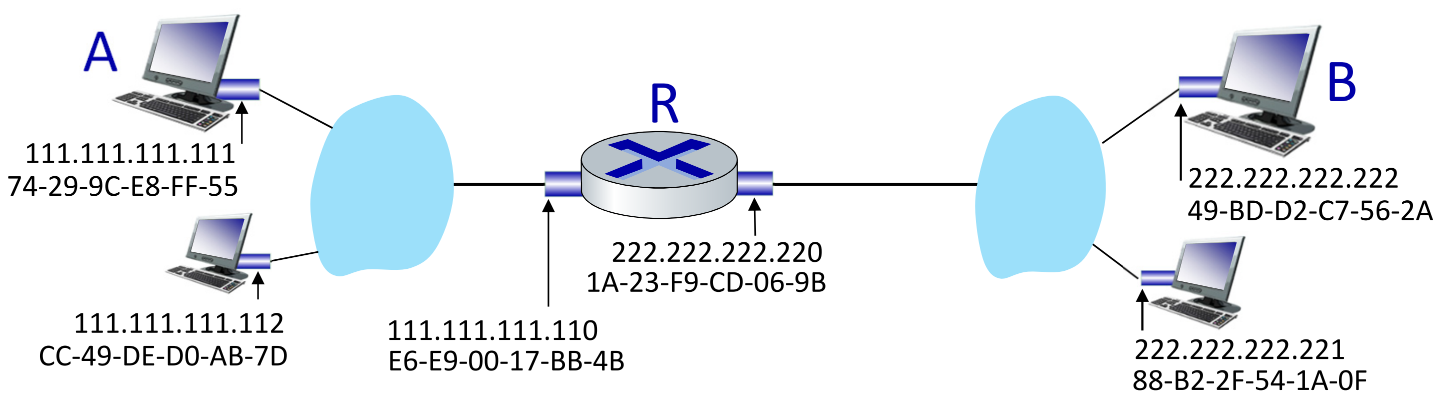


3 A receives B's reply, adds B entry into its local ARP table

Routing to another subnet: addressing

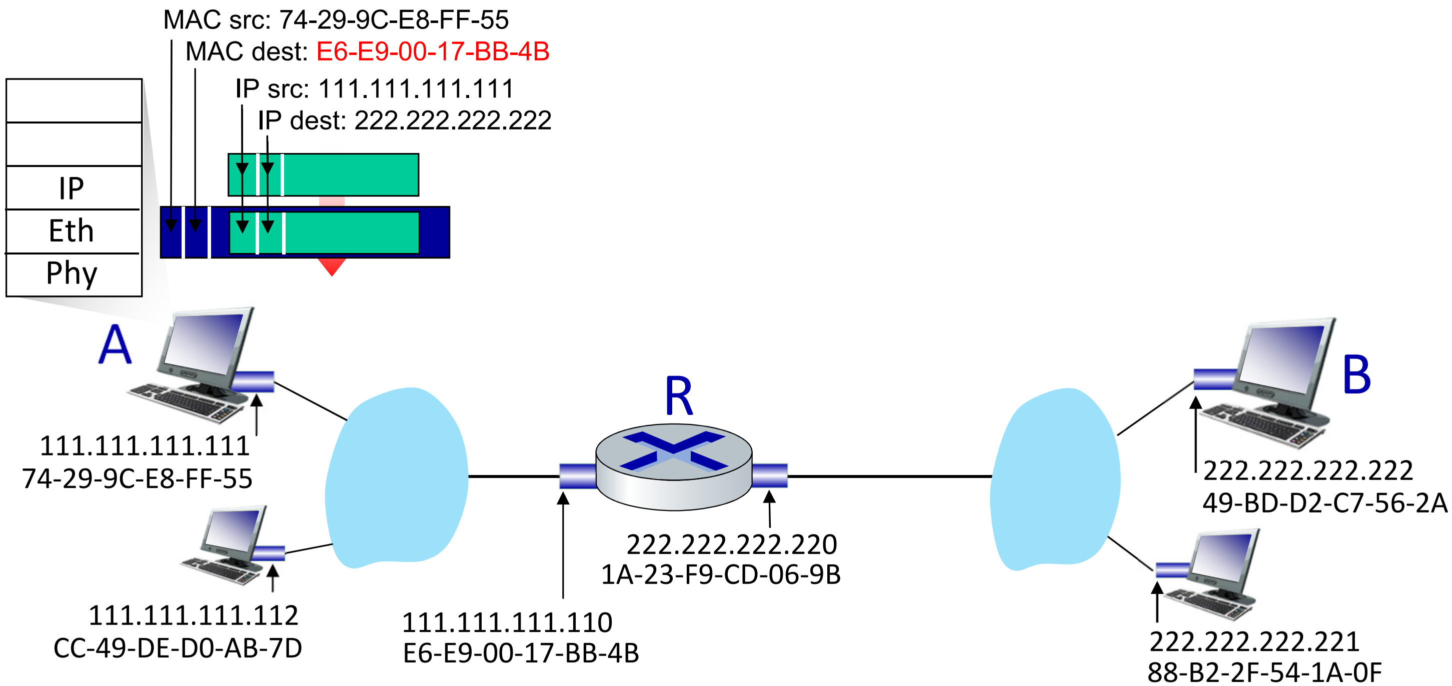
walkthrough: sending a datagram from *A* to *B* via *R*

- 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?)



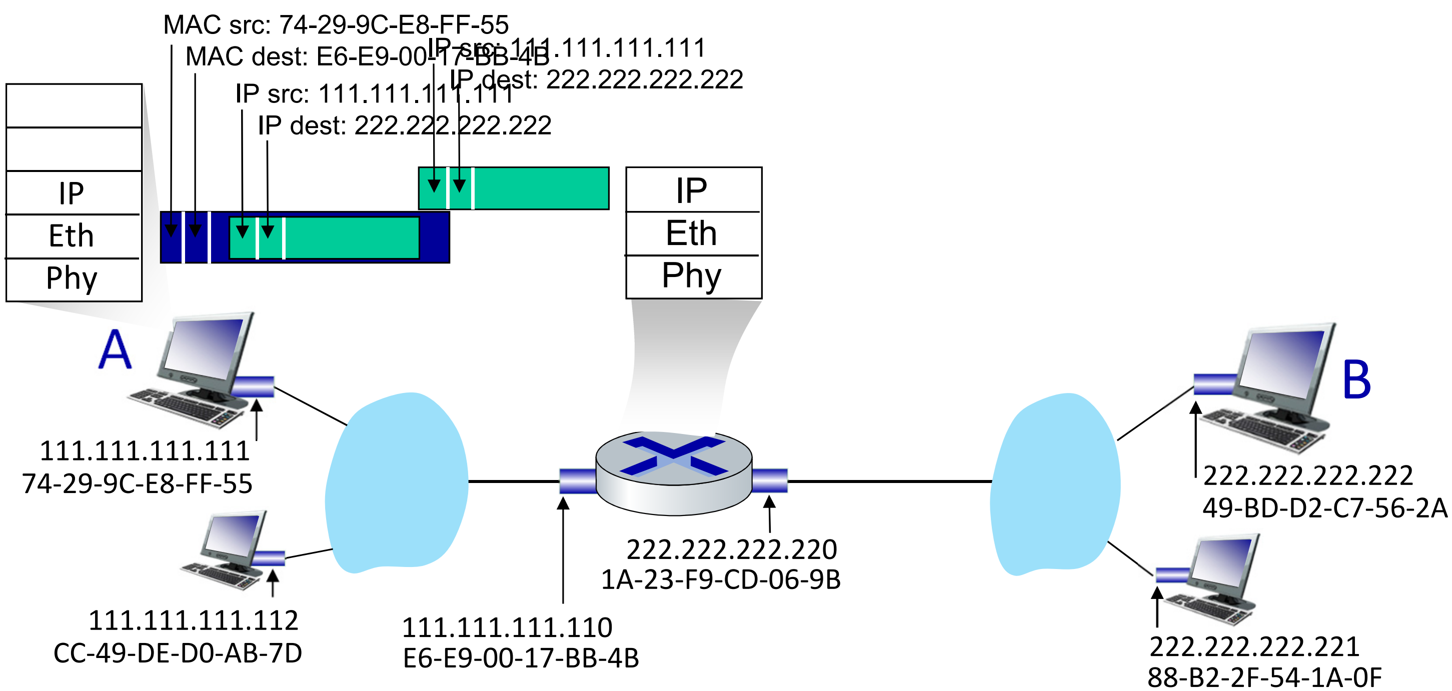
Routing to another subnet: addressing

- 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



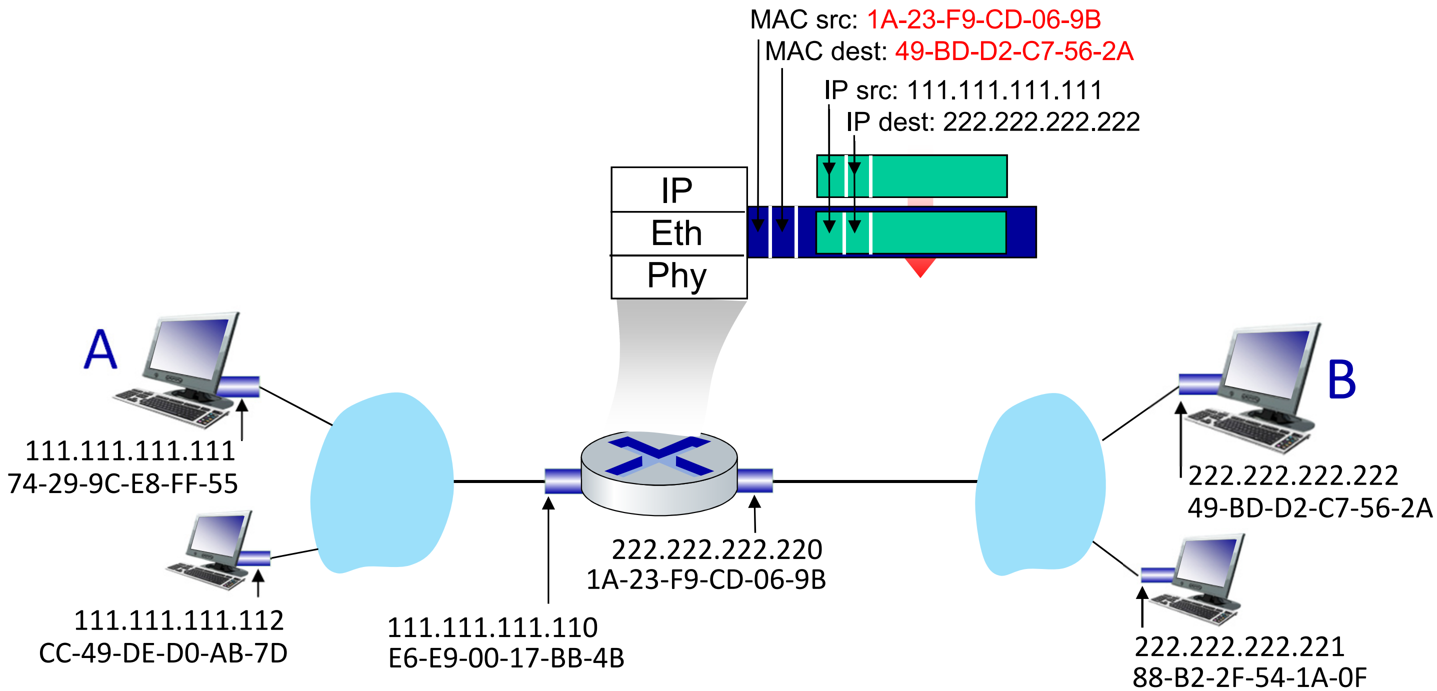
Routing to another subnet: addressing

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



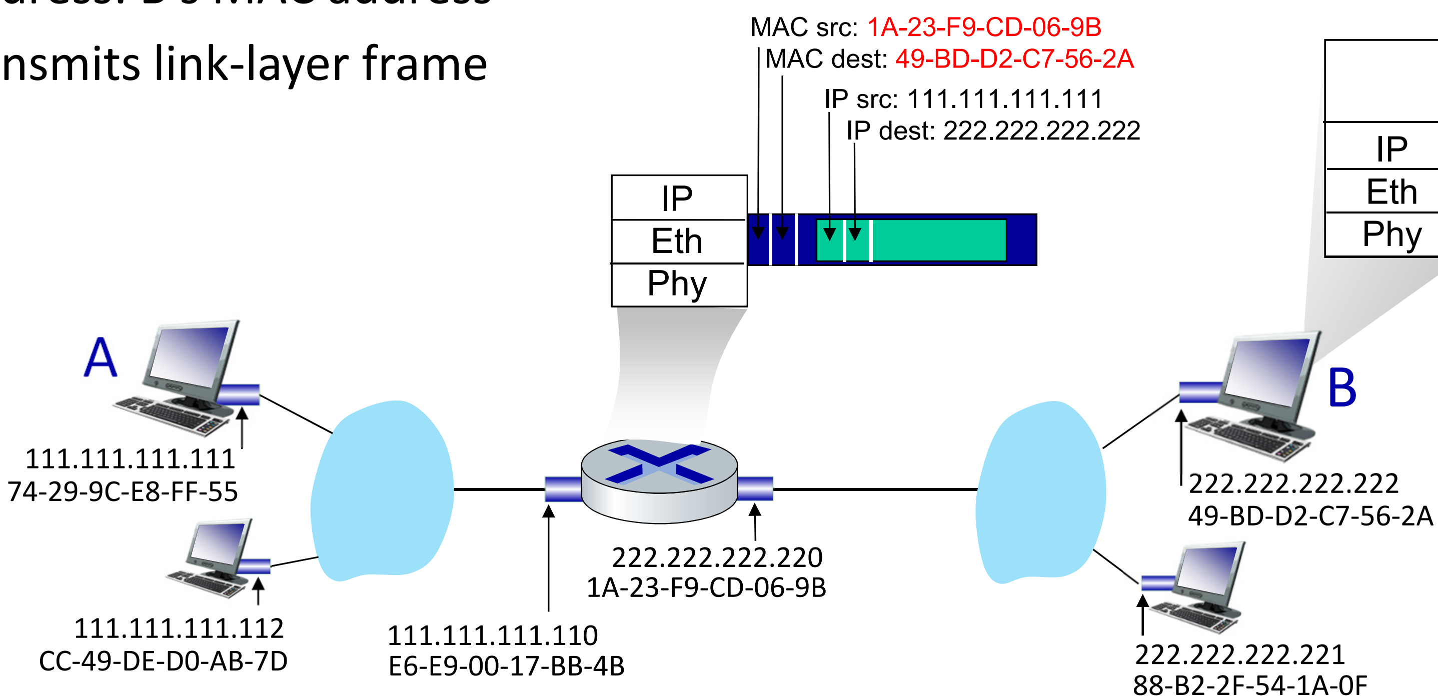
Routing to another subnet: addressing

- 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



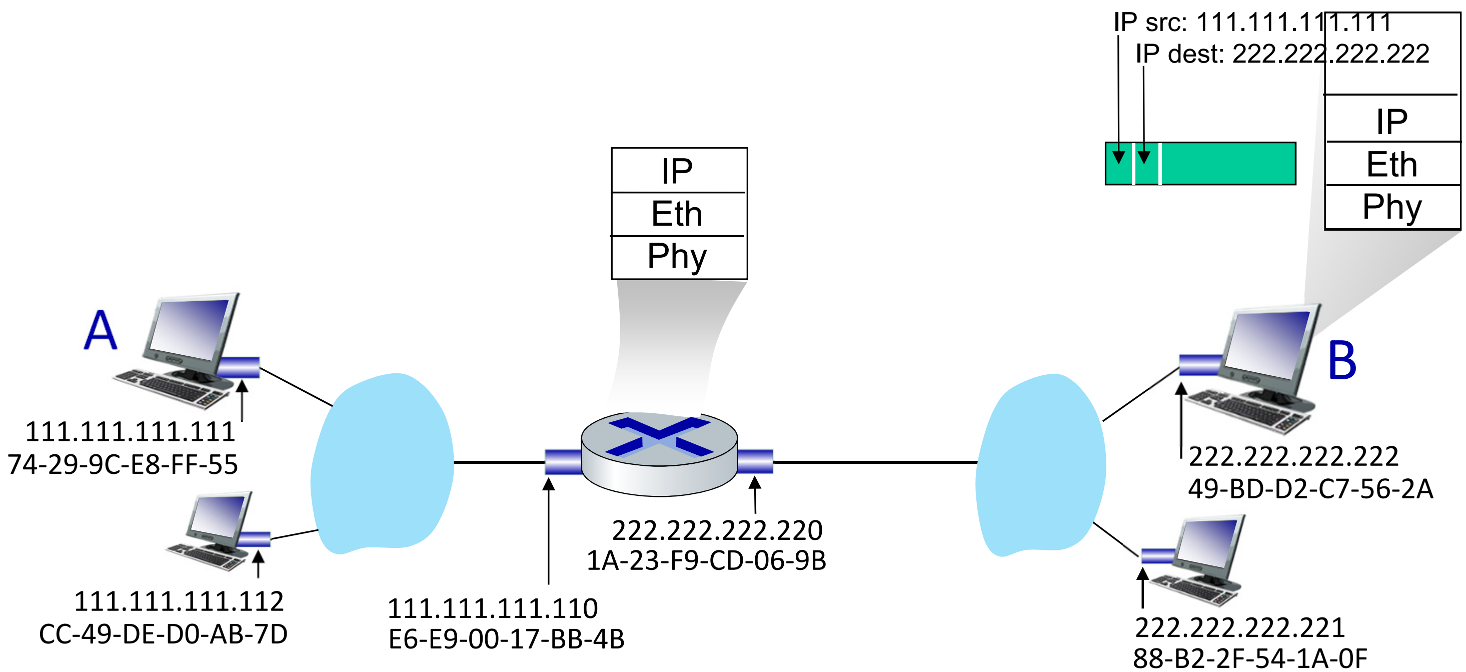
Routing to another subnet: addressing

- 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



Link layer, LANs: roadmap

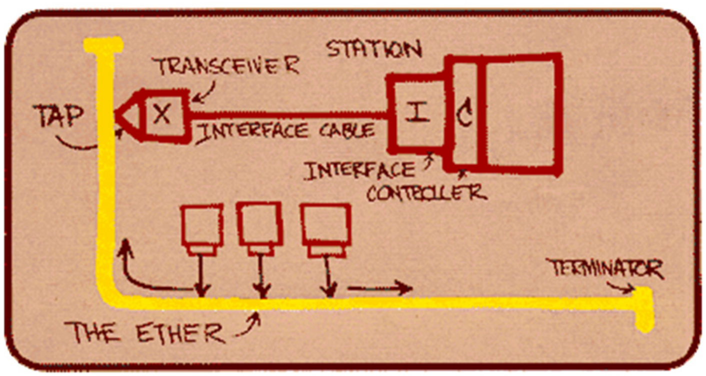
- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- link virtualization: MPLS
- data center networking

- a day in the life of a web request

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)



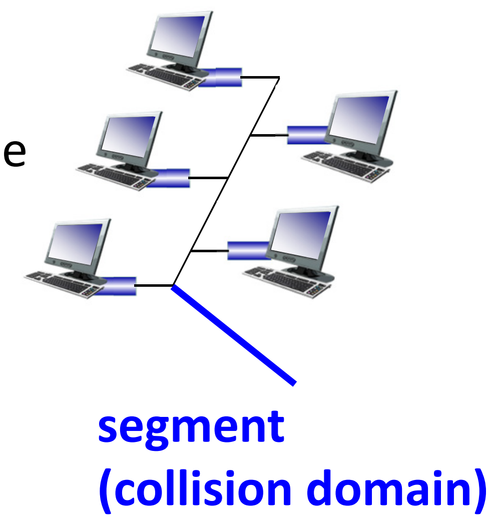
Metcalfe's Ethernet sketch

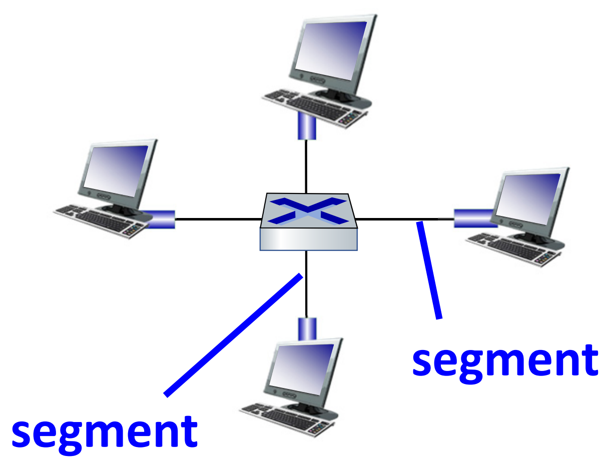
https://www.uspto.gov/learning-and-resources/journeys-innovation/audio-stories/defying-doubters

Ethernet: physical topology

- bus: popular through mid 90s
  - all nodes in same collision domain (or called “segment” later)
- switched: prevails today
  - *switch* in center (hosts are connected to switches)
  - each segment runs a (separate) Ethernet protocol
    - store-and-forward (frames are stored in a switch and then forwarded)
    - (different) segments 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

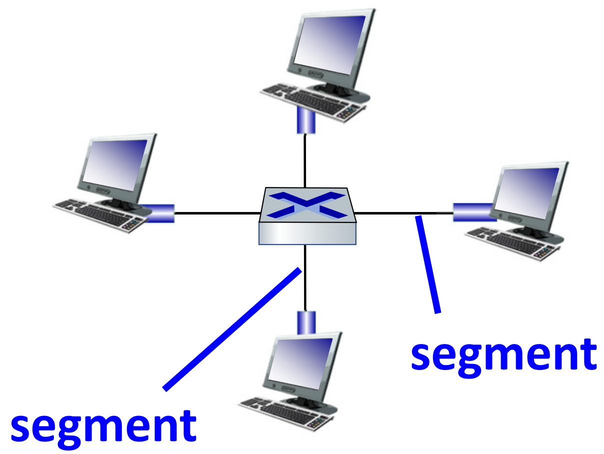
- connectionless: no handshaking between sending and receiving NICs
- Ethernet's MAC protocol: unslotted CSMA/CD with binary backoff
  - backoff and retransmit
- unreliable: receiving NIC doesn't send ACKs or NAKs to sending NIC
  - data in dropped frames recovered only if initial sender uses higher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

Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- link virtualization: MPLS
- data center networking

- a day in the life of a web request

Ethernet switch



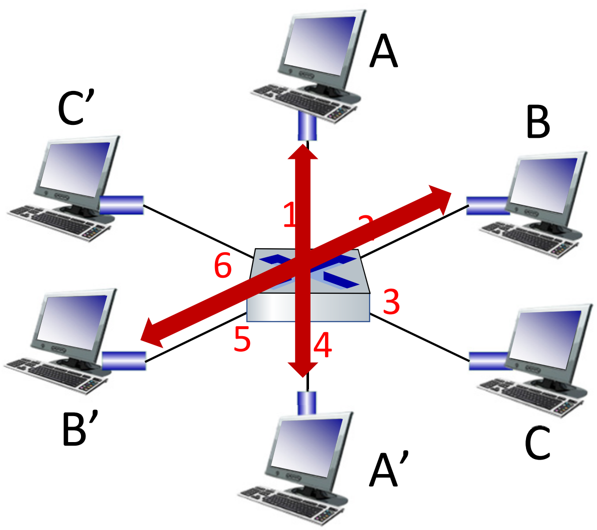
- switch is a link-layer device
  - store and forward Ethernet frames
    - based on incoming frame's destination MAC address
  - when frame is to be forwarded on a segment, uses CSMA/CD to access the segment
- switch is transparent:
  - in layer 3, hosts unaware of the presence of switches
- switch is plug-and-play, self-learning
  - switches do not need to be configured

Switch: multiple simultaneous transmissions

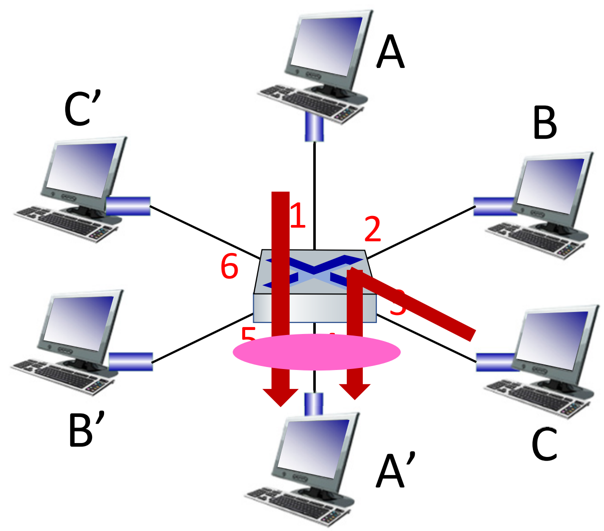
- hosts often have dedicated, direct connection to switch
- switches buffer packets
- Ethernet’s MAC protocol used on *each* incoming link, so:
  - each link is its own collision domain
  - de facto collision-free and full-duplex
    - since 10Base-T (10Mbps rate)

Switch: multiple simultaneous transmissions

- A-to-A’ and B-to-B’ can transmit simultaneously, without collisions



- but A-to-A’ and C to A’ *can’t* happen simultaneously



Switch forwarding table

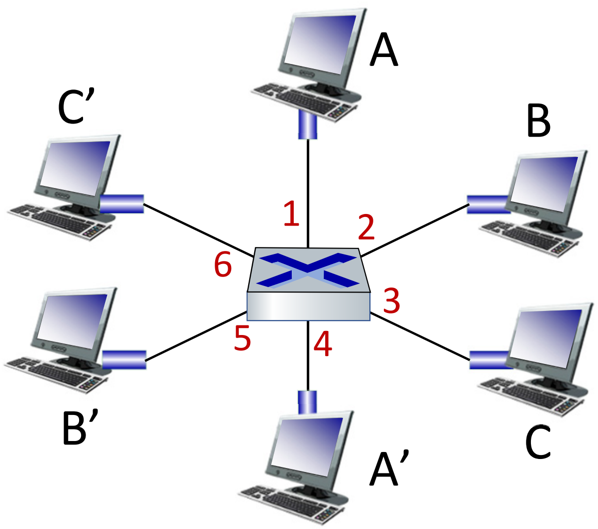
Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

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

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

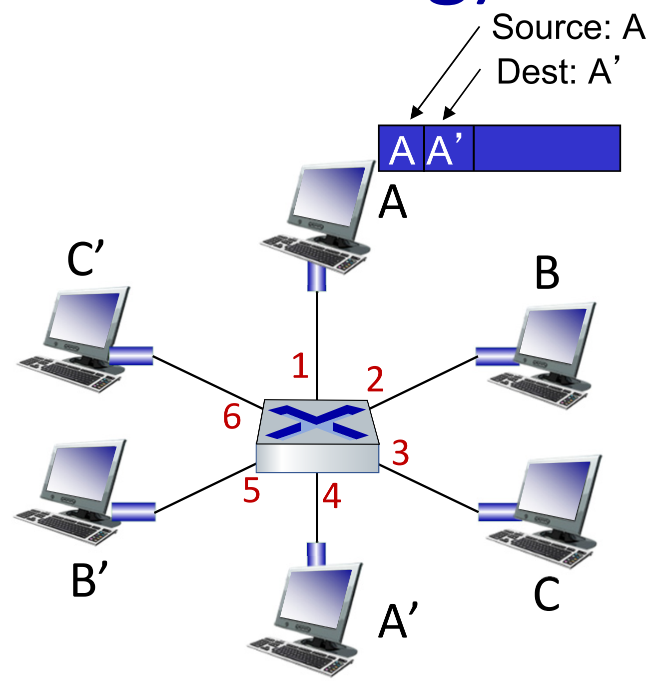
Q: how are entries created, maintained in switch table?

- something like a routing protocol? No!



Switch: self-learning (backward 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

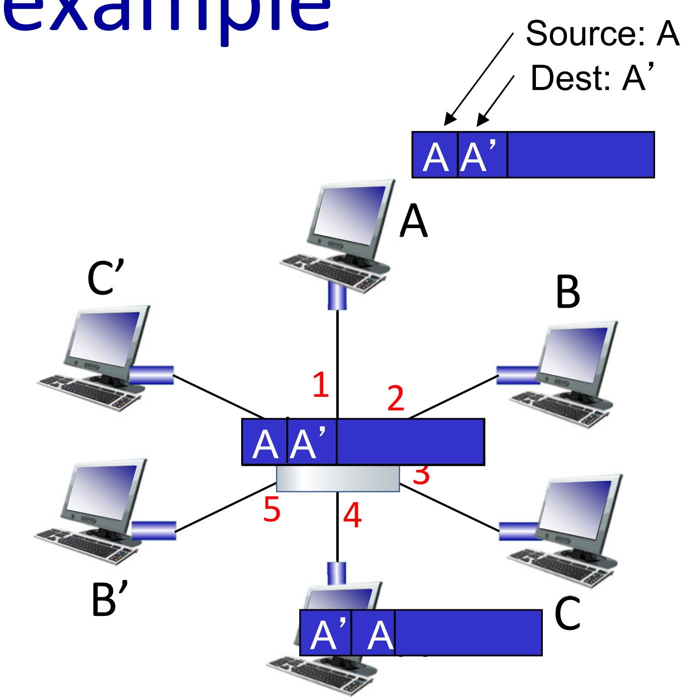


MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

Self-learning, forwarding: example

- if switch table has no entry for the frame destination (e.g. A') → flood (except the incoming link)
- if switch table has an entry for the frame destination → send on just one link

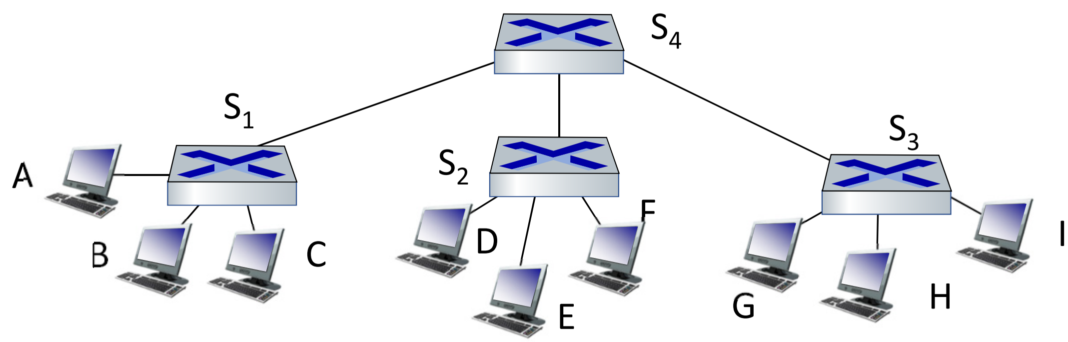


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

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!)
  - no loop (because in a LAN, all links that cause loops are disabled)