# CS 305: Computer Networks

Fall 2024

**Link Layer**

**Tianyue Zheng**

Department of Computer Science and Engineering
Southern University of Science and Technology (SUSTech)

# Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

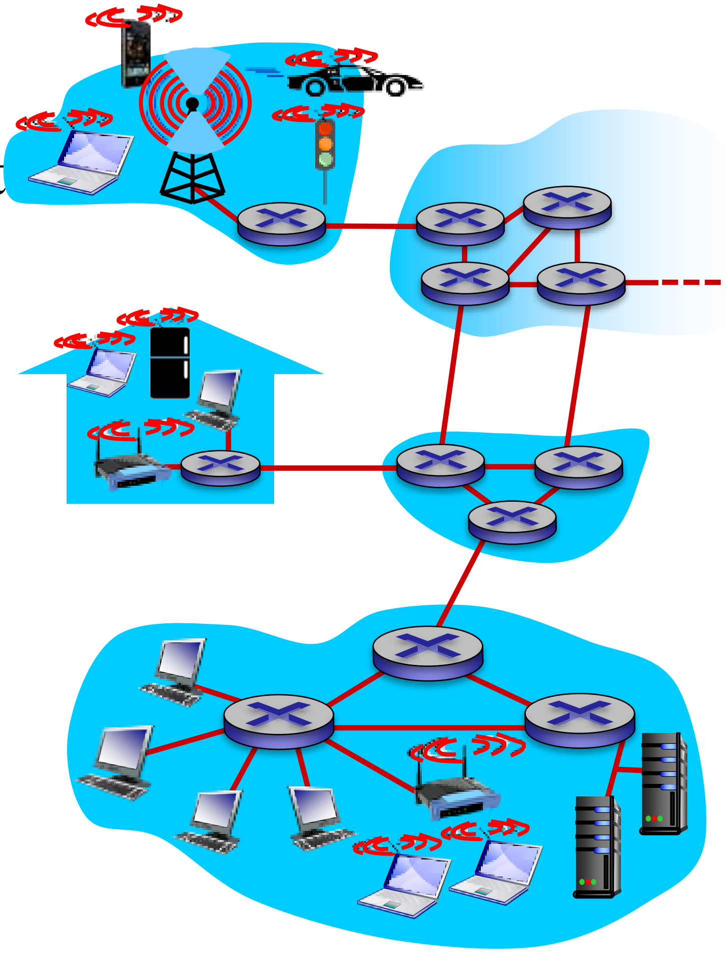
6.6 data center networking

6.7 a day in the life of a web request

# Link layer: introduction

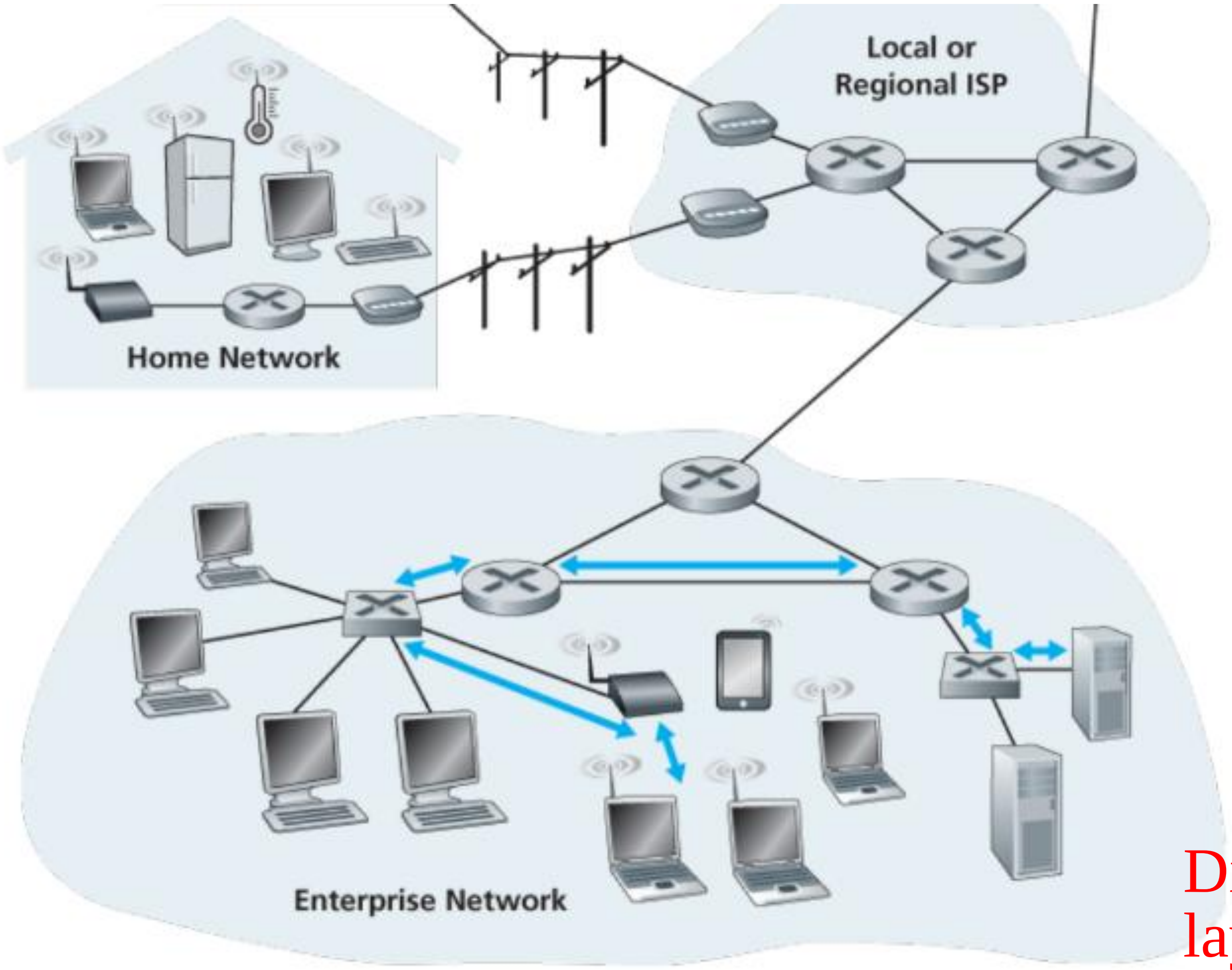
*terminology:*

- Hosts, switches, access points: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired links
  - wireless links
- layer-2 packet: frame, encapsulates datagram



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

# Link layer: introduction




Different link layer protocols at different links.

# Link layer: context

- datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, PPP on intermediate links, 802.11 on last link
- each link protocol provides different services
  - e.g., may or may not provide rdt over link

*transportation analogy:*

- trip from SUSTech to Universal Studio
  - metro: SUSTech to SZ North
  - High speed train: SZ North to Beijing West
  - taxi: Beijing West to Universal Studio
- tourist = datagram
- transport segment = communication link
- transportation mode = link layer protocol
- travel agent = routing algorithm

# Link layer services

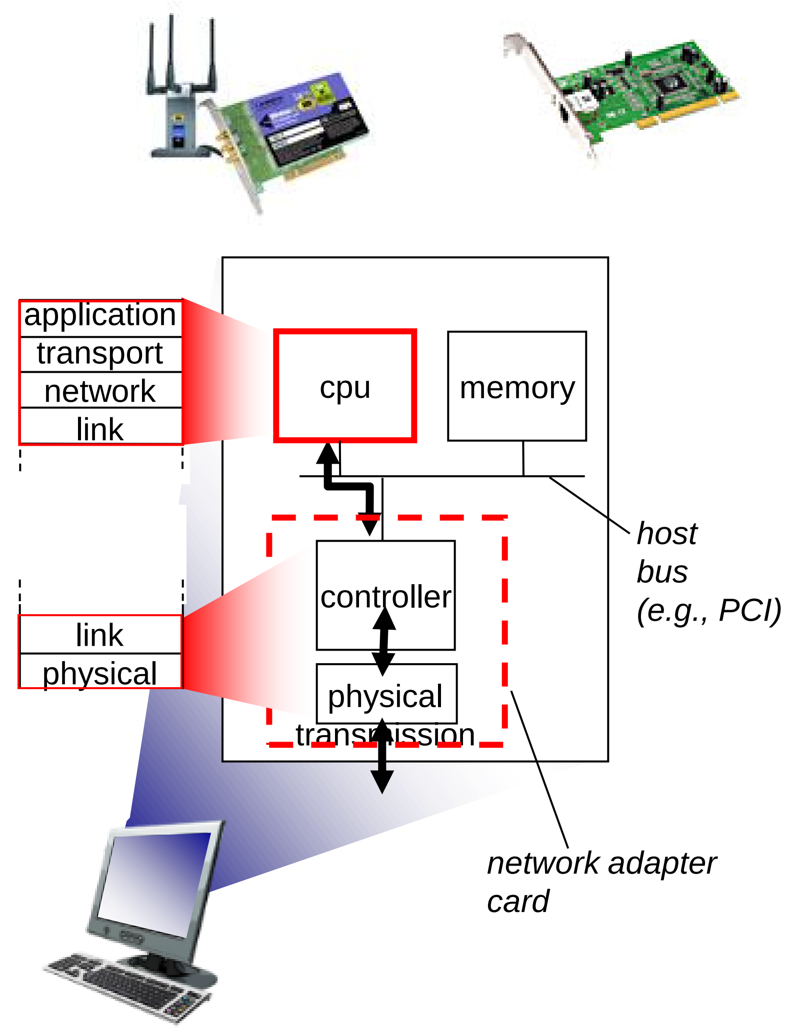
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, destination
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - *Q:* why both link-level and end-end reliability?

# Link layer services (more)

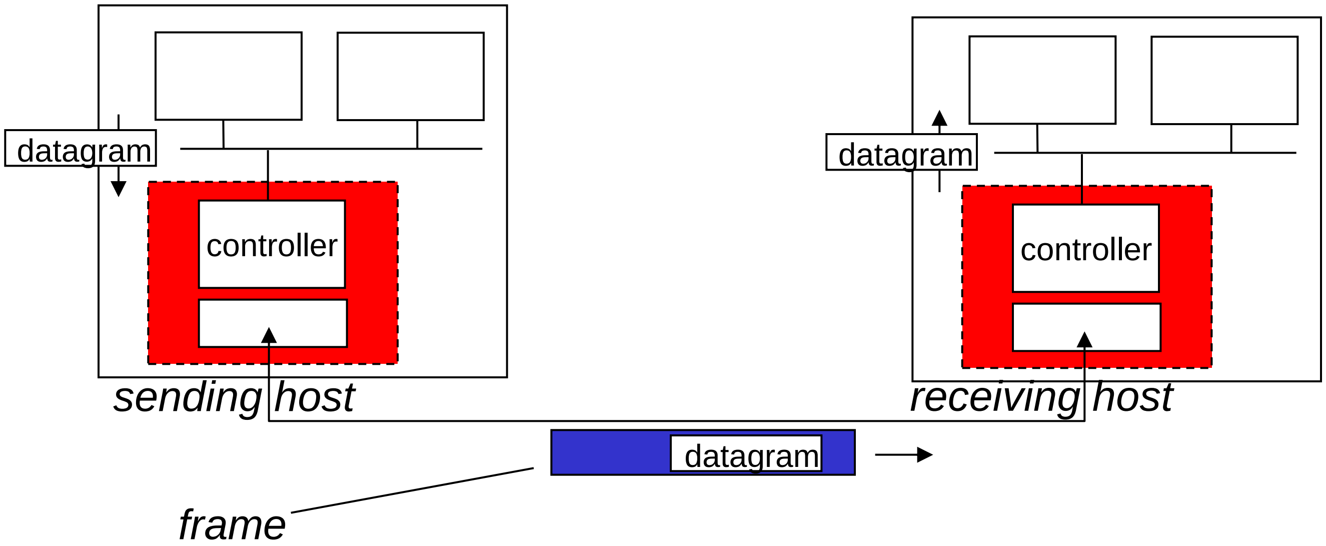
- *error detection*:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- *error correction:*
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- *half-duplex and full-duplex*
  - with half duplex, nodes at both ends of link can transmit, but not at same time

# Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC) or on a chip
  - Ethernet card, 802.11 card; Ethernet chipset
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



# Adaptors communicating




- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, etc.
- receiving side
  - looks for errors, rdt, flow control, etc.
  - extracts datagram, passes to upper layer at receiving side

# Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

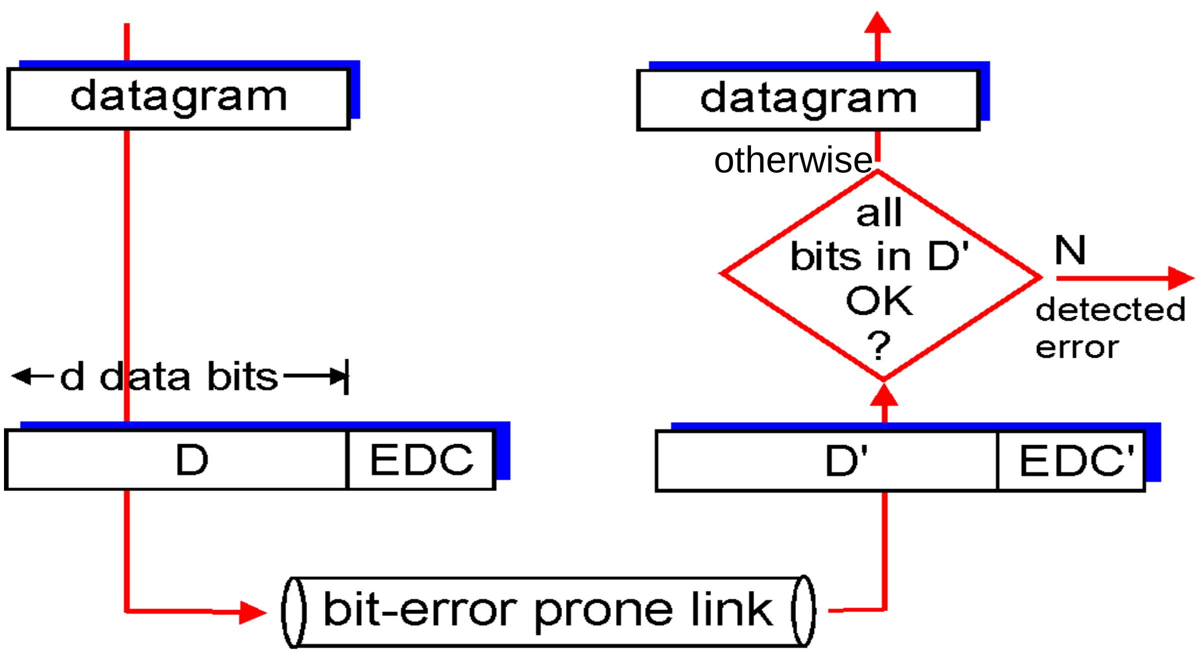
6.6 data center networking

6.7 a day in the life of a web request

# Error detection and correction

EDC= Error Detection and Correction bits
D = Data protected by error checking, may include header fields

- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction, but larger overhead

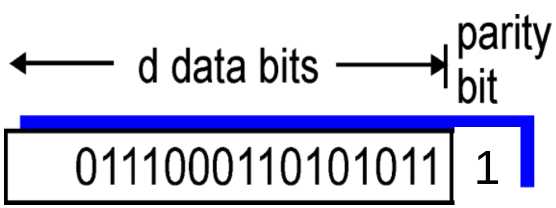



- Parity checks
- Check-summing methods
- Cyclic-redundancy check

# Parity checking

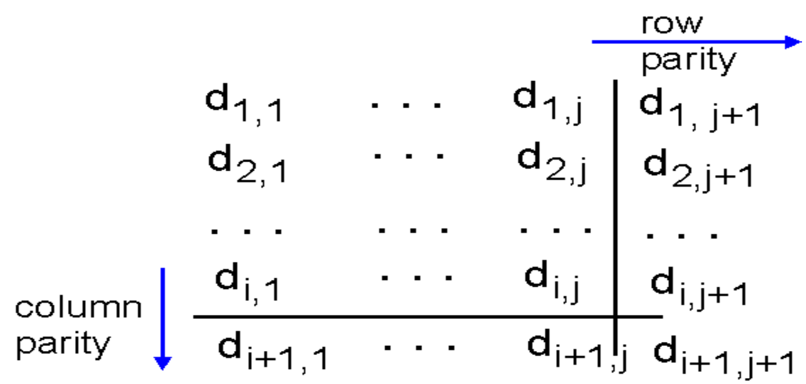
*single bit parity:*

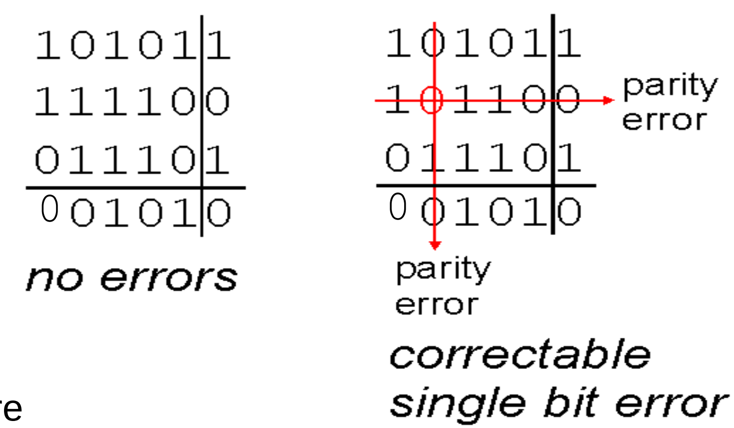
- *d*etect single bit errors
- Even parity scheme
- Odd parity scheme




*two-dimensional bit parity:*

- detect and correct single bit errors

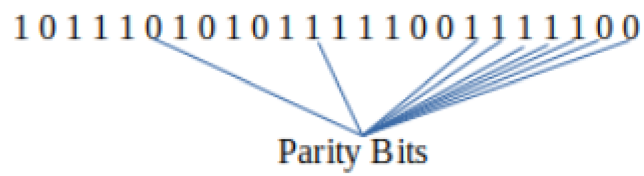


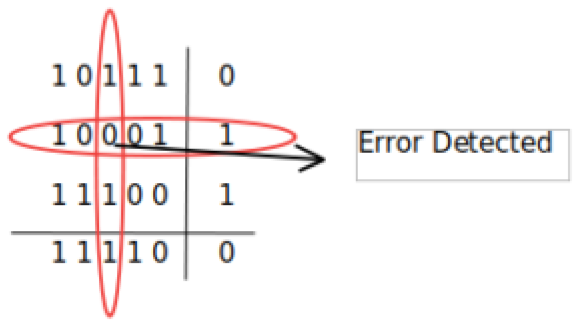

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

# Parity checking

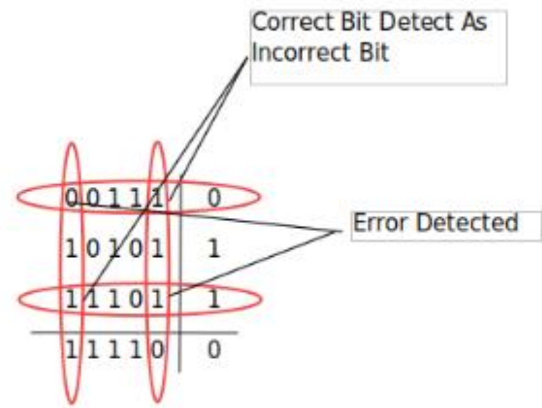
| | |
|---|---|
| 1 0 1 1 1 | 0 |
| 1 0 1 0 1 | 1 |
| 1 1 1 0 0 | 1 |
| 1 1 1 1 0 | 0 |



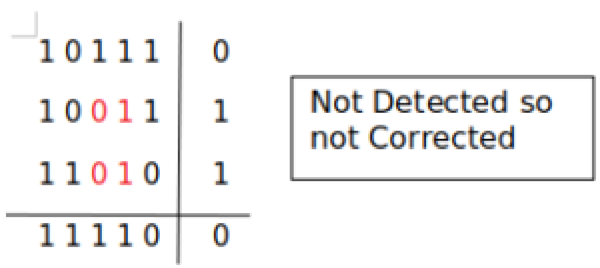

**Case 1: a bit is in error.**




**Case 2: two bits are in error.**




**Case 3: error not detected**




**Many other cases …**

# Internet checksum (review)

goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer only)

*sender:*

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

*receiver:*

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?*

# Cyclic redundancy check

- more powerful error-detection coding
- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - can detect all consecutive bit errors of r bits or less
- widely used in practice (Ethernet, 802.11 WiFi, ATM)

```
1011 XOR 0101 = 1110
1001 XOR 1101 = 0100

1011 - 0101 = 1110
1001 - 1101 = 0100
```

← d bits → ← r bits →

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

*bit pattern*

$$D * 2^r \quad XOR \quad R$$

*mathematical formula*

# Cyclic redundancy check

All CRC calculations are done in modulo-2 arithmetic without carries in addition or borrows in subtraction.

- This means that addition and subtraction are identical, and
- both are equivalent to the bitwise exclusive-or (XOR) of the operands.

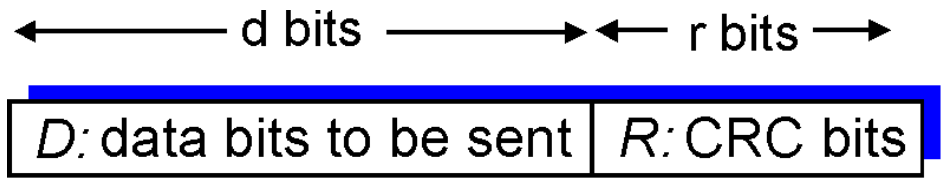
```
1011 XOR 0101 = 1110
1001 XOR 1101 = 0100

1011 - 0101 = 1110
1001 - 1101 = 0100
```

```
             10001  remainder 101
       ___________
 10011|100100110
       10011
           10110
           10011
             101
```

Multiplication and division are the same as in base-2 arithmetic, except that any required addition or subtraction is done without carries or borrows.

# CRC example




want:

$$D \cdot 2^r \text{ XOR } R = nG$$

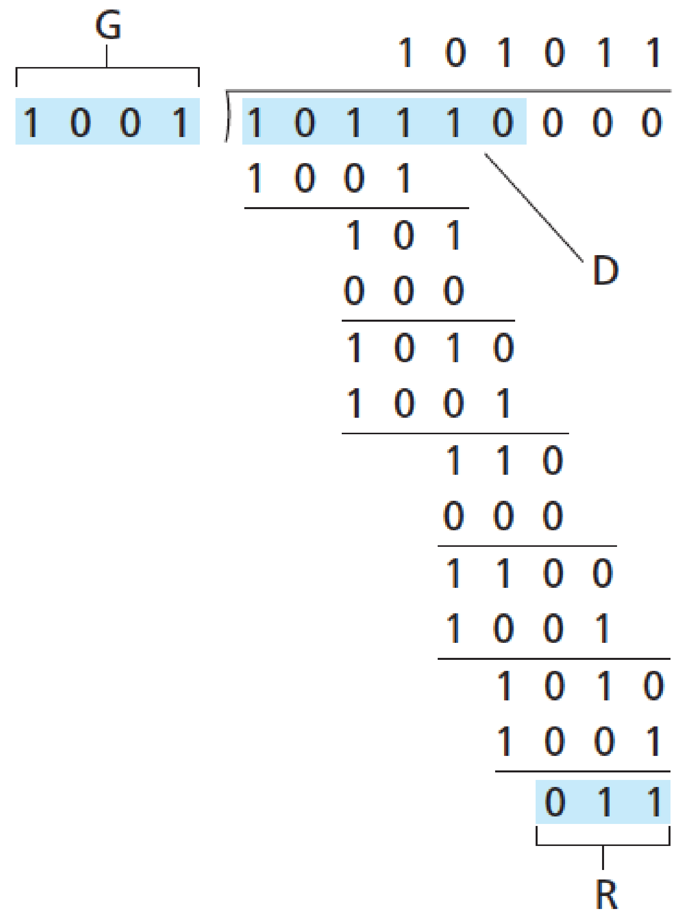
*equivalently:*

$$D \cdot 2^r = nG \text{ XOR } R$$

*equivalently:*

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

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




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

# Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

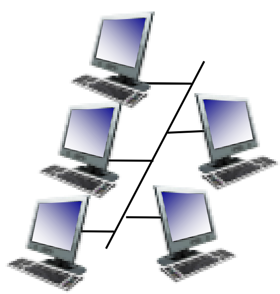
6.6 data center networking

6.7 a day in the life of a web request

# Multiple access links, protocols

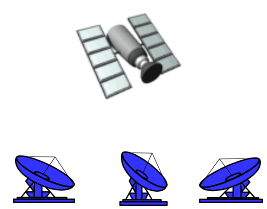
two types of "links":

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



shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

# Multiple access 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 which and 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:* broadcast channel of rate R bps

*Desired properties:*

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: taxonomy

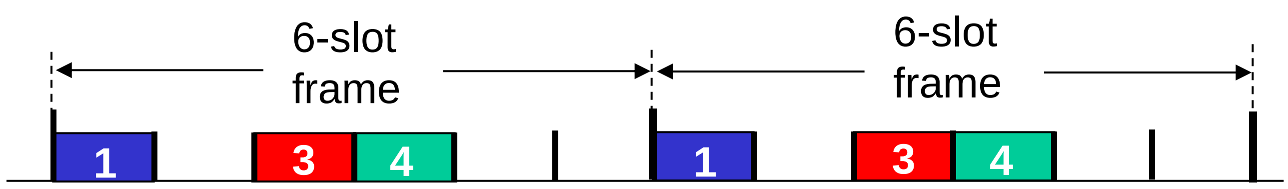
three broad classes:

- *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

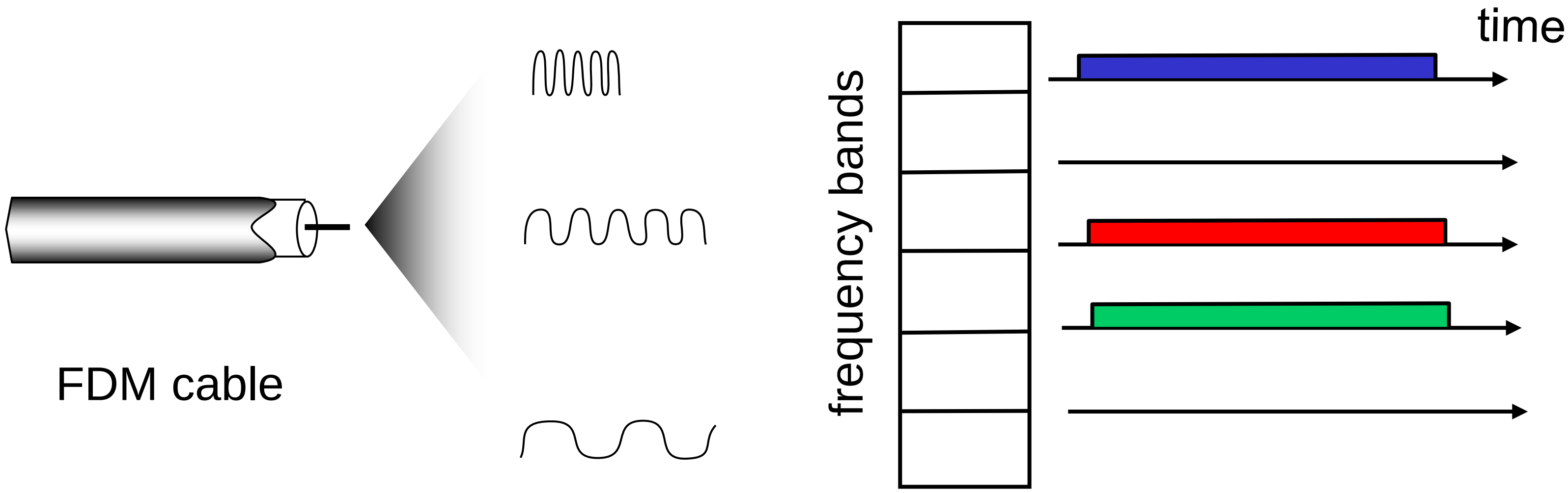
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



# 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



# Channel partitioning：limiations

*Desired properties:*

1. 【x】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

# Random access protocols

- Random access; if fails, wait for a random time
- 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)
- **Advantage:** when node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD (Ethernet), CSMA/CA (802.11)

# Slotted ALOHA

*assumptions:*

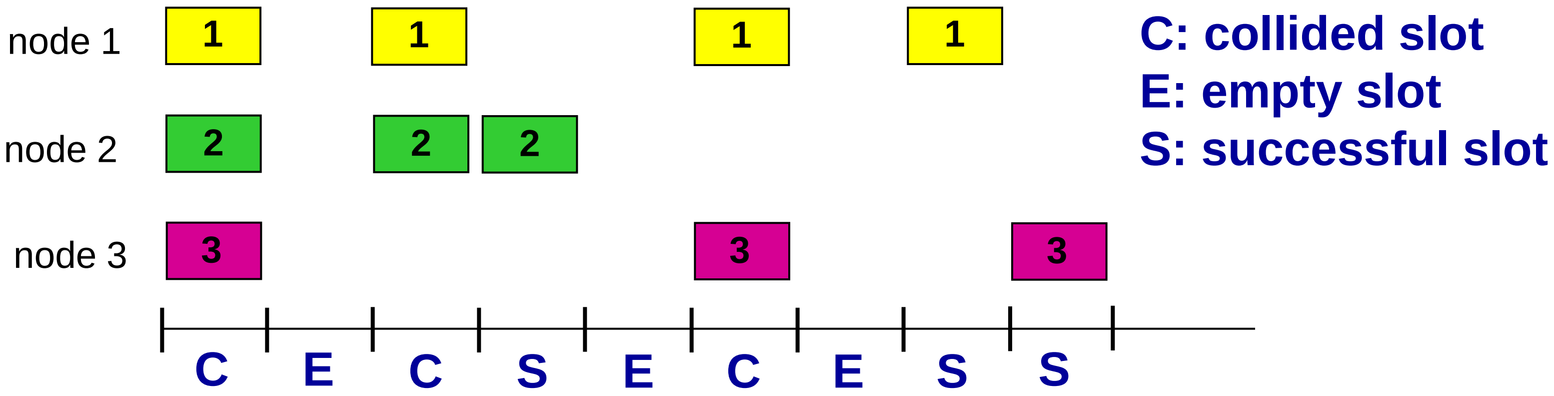
- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

*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




*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
- 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:

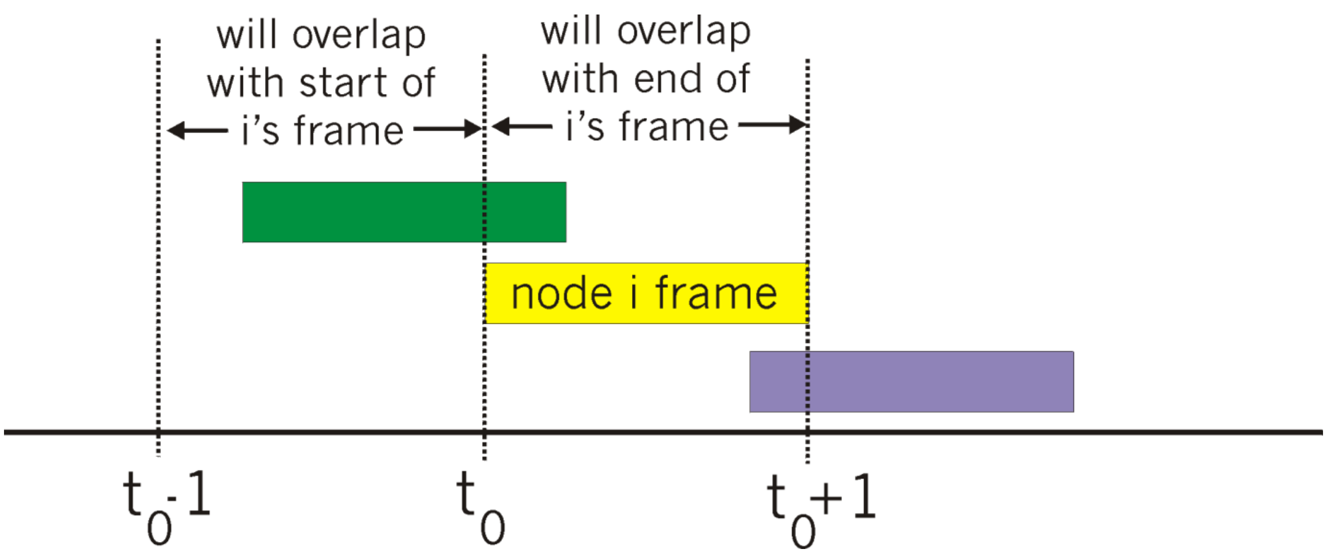
*max efficiency = 1/e = 0.37*

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

!

# Pure (unslotted) ALOHA

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



# Pure ALOHA efficiency

P(success by given node) = P(node transmits) ·

P(no other node transmits in $[t_0-1,t_0]$ ·

P(no other node transmits in $[t_0-1,t_0]$

$$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$$

$$= p \cdot (1-p)^{2(N-1)}$$

… choosing optimum p and then letting $n \longrightarrow \infty$

$$= 1/(2e) = 0.18$$

even *worse* than slotted Aloha!

# CSMA (carrier sense multiple access)

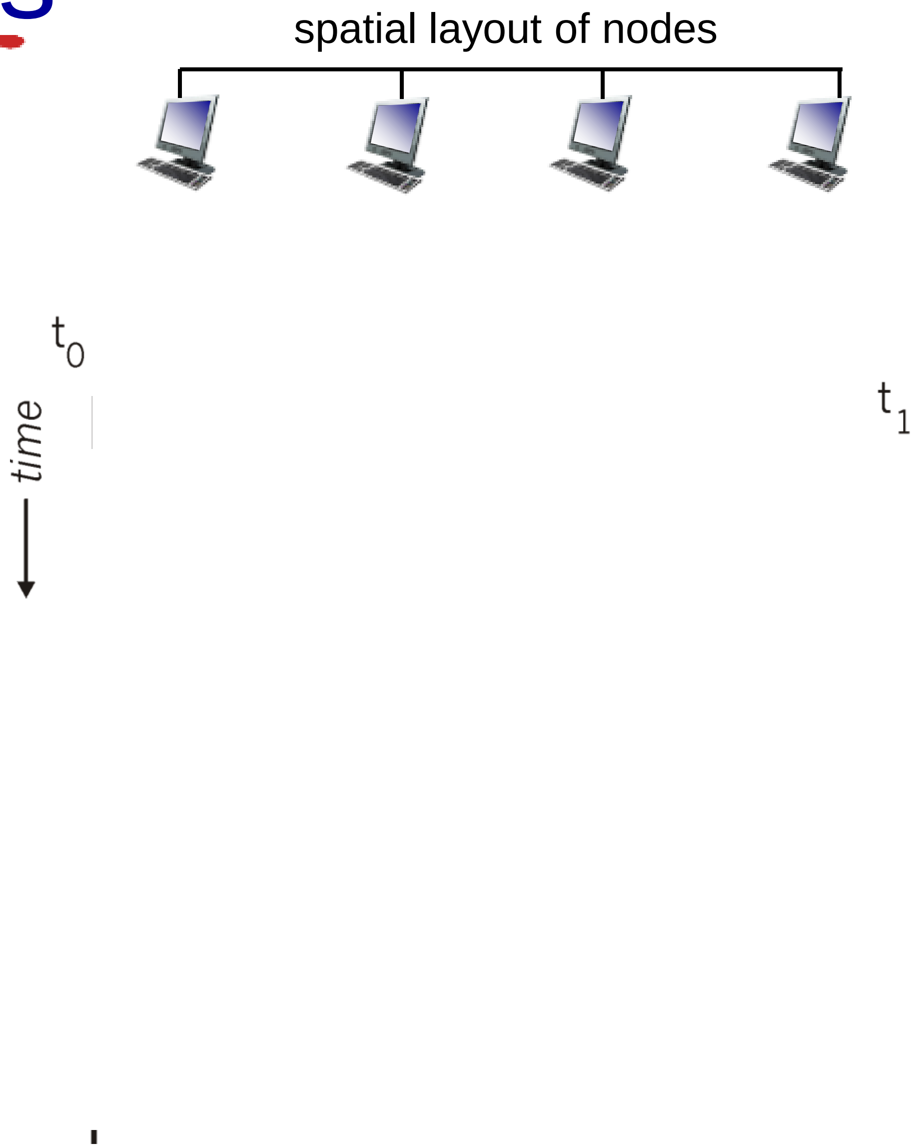
*CSMA*: listen before transmit:

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

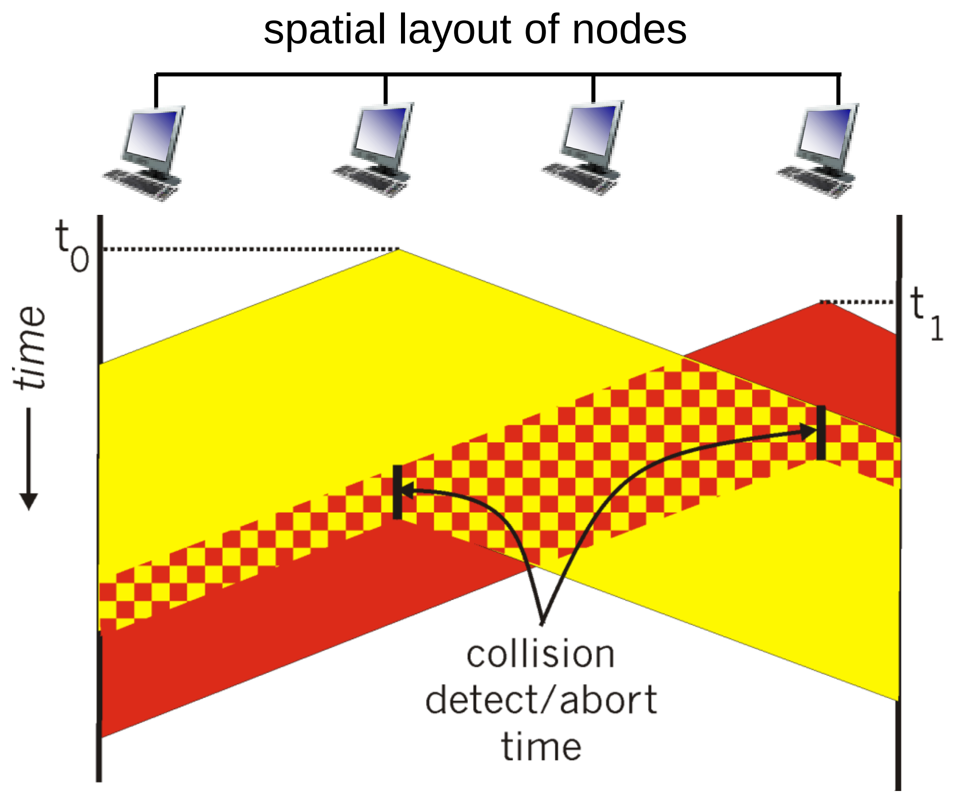


# CSMA/CD (collision detection)

*CSMA/CD:* CSMA+CD (collision detection)

- collision detection → stop talking:
  - collisions *detected* within short time
  - colliding transmissions are aborted, reducing channel wastage
- Suitable scenarios:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength
- human analogy: the polite conversationalist

# CSMA/CD (collision detection)



# Ethernet CSMA/CD algorithm

1. network adapter (network interface card, NIC) 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
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·512 bit times, returns to Step 2
   - longer backoff interval with more collisions

# "Taking turns" MAC protocols

channel partitioning MAC protocols:

- share channel *efficiently* and *fairly* at high load
- inefficient at low load: delay in channel access, 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

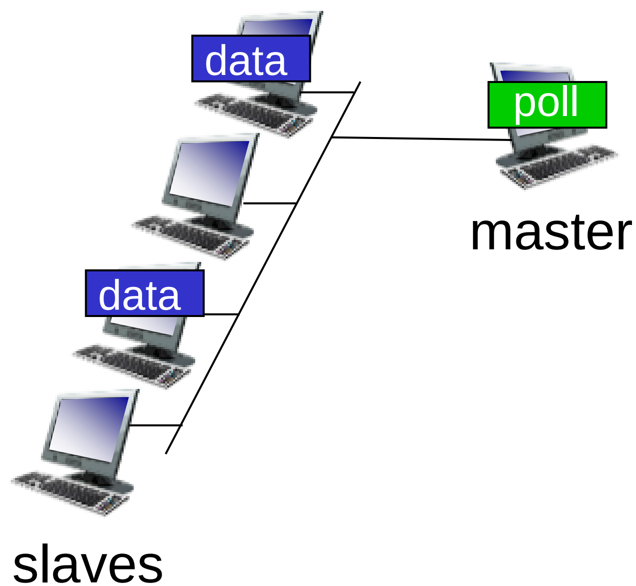
look for best of both worlds!

- when one node wants to transmit, it can send at rate R.
- when M nodes want to transmit, each can send at average rate R/M

# "Taking turns" MAC protocols

polling:

- <u>master node</u> "invites" slave nodes to transmit in turn
- maximum number of frames
- concerns:
  - polling overhead
  - single point of failure (master)

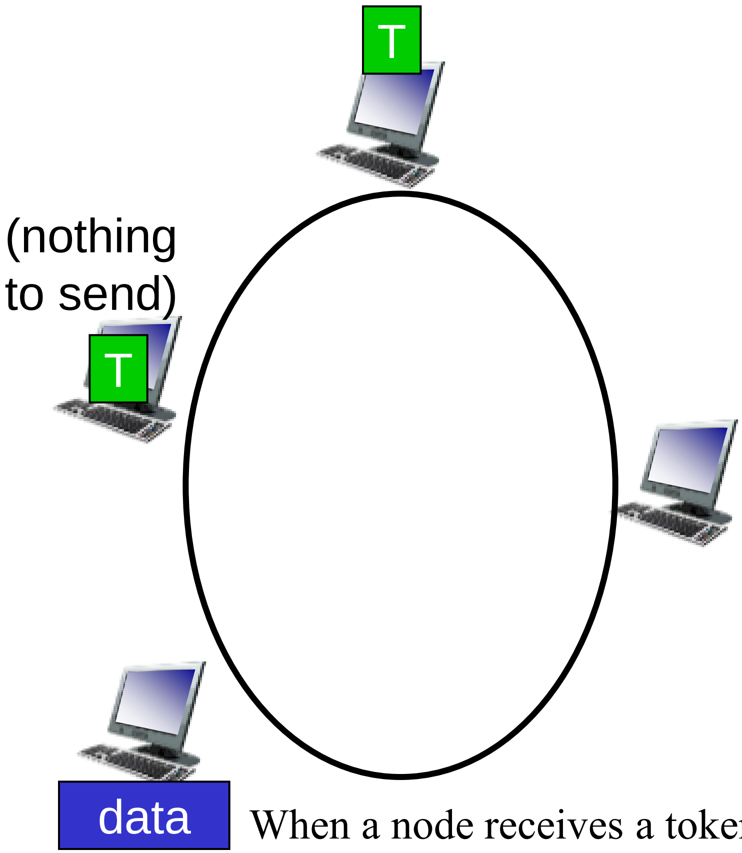



The master node must poll each of the nodes in turn no matter they have frames or not

# "Taking turns" MAC protocols

## token passing:

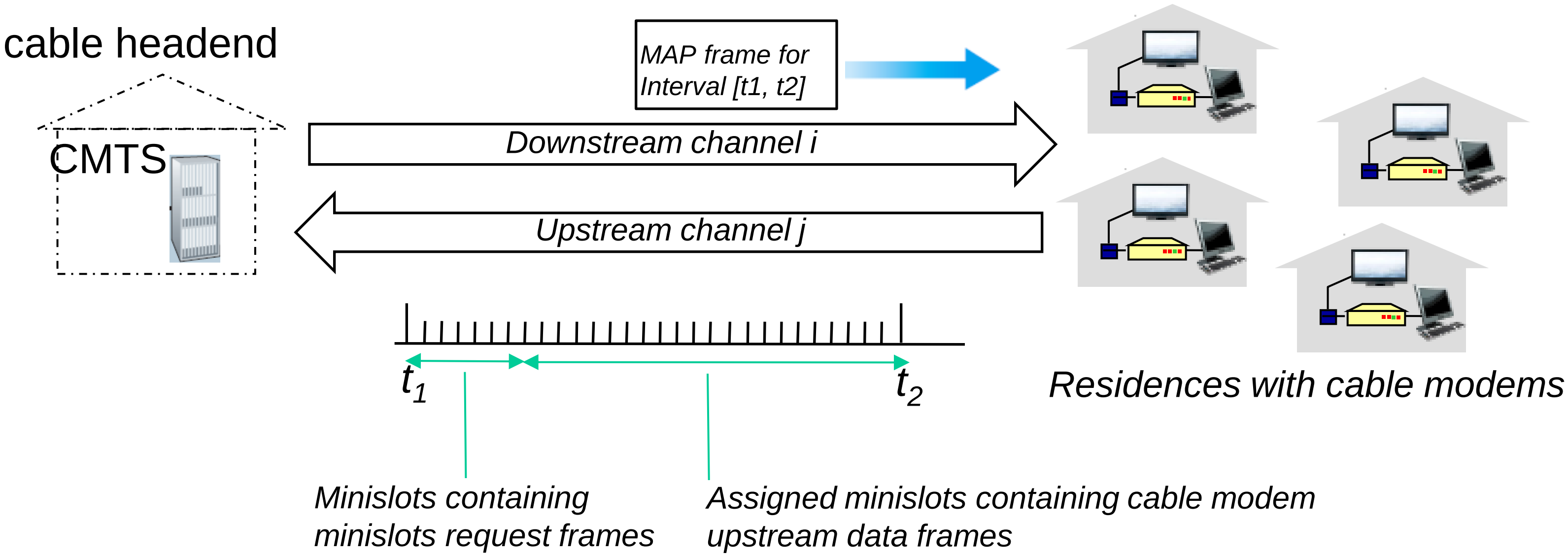
- control *token* passed from one node to next sequentially (fixed order).
- distributed
- concerns:
  - token overhead
  - single point of failure (token)




When a node receives a token,

- if it has some frames to transmit. it holds onto the token;
- otherwise, it immediately forwards the token to the next node.

# Case study: Cable access network




DOCSIS: data over cable service interface specification

- FDM over upstream, downstream frequency channels
  - Downstream (no multiple access); upstream (collision)
- TDM upstream: some slots assigned, some have contention
  - downstream MAP frame: assigns upstream slots
  - request for upstream slots (and data) transmitted random access (binary backoff) in selected slots

# 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

# Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs

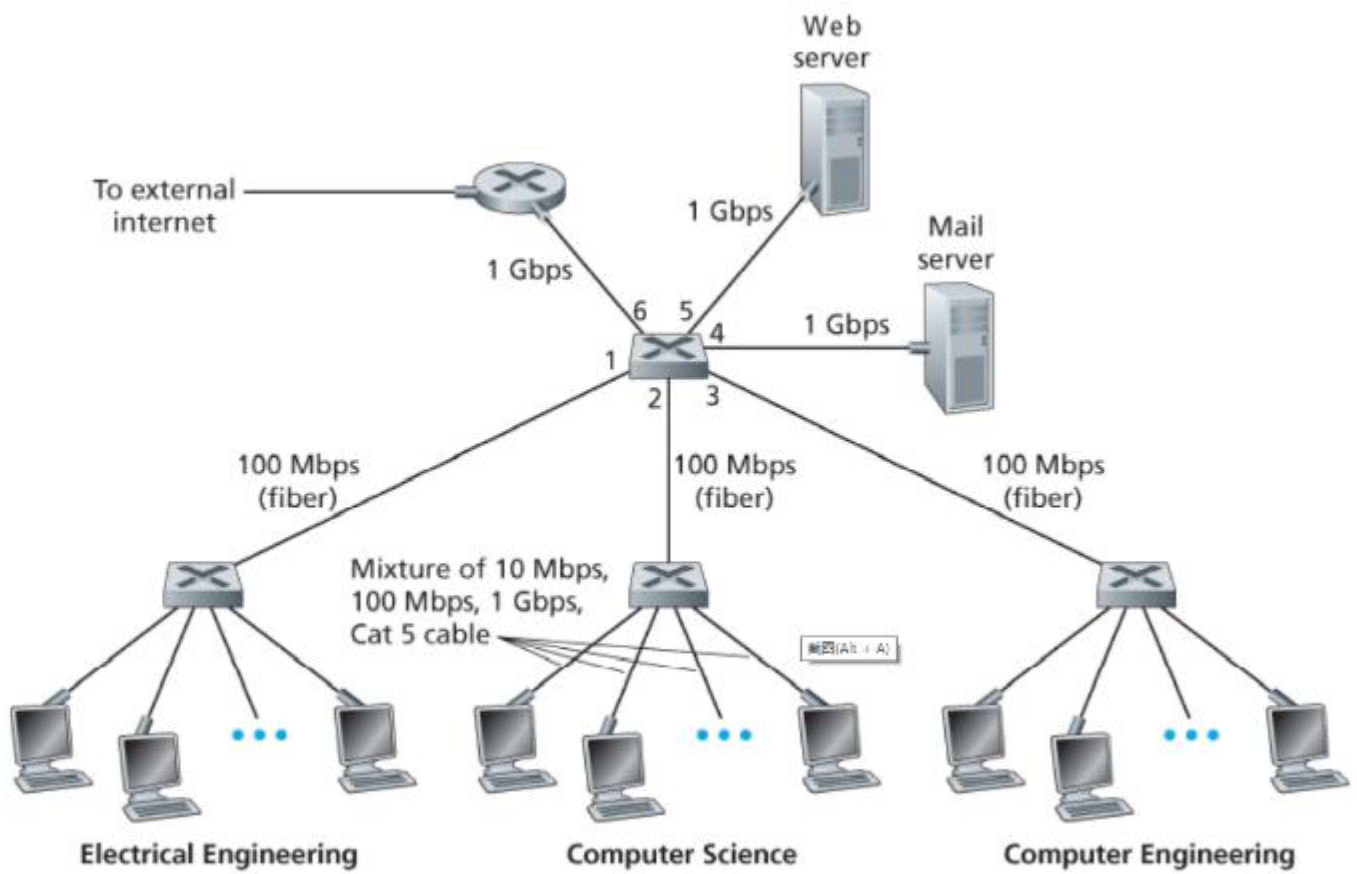
- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

6.6 data center networking

6.7 a day in the life of a web request

# LANs




Because these switches operate at the link layer,

- don’t recognize network-layer addresses
- don’t use routing algorithms like RIP or OSPF to determine paths through switches

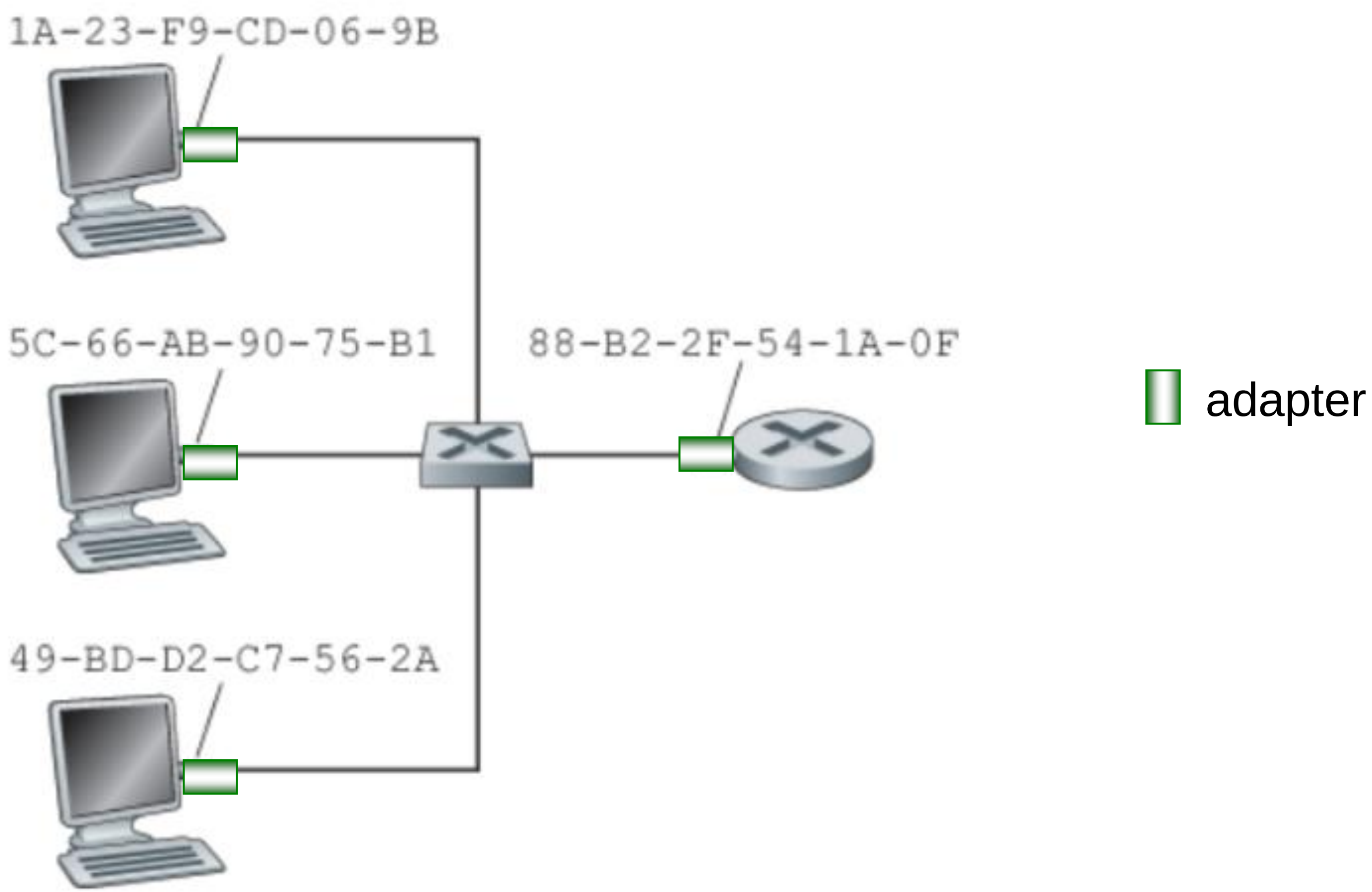
# MAC addresses and ARP

- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
- MAC (or LAN or physical or Ethernet) address:
  - Adapter (network interface) rather than host or routers
  - Link-layer switches do NOT have MAC addresses
  - function: *used "locally" to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)*
  - 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable; no two adapters have the same address
  - e.g.: 1A-2F-BB-76-09-AD

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

# LAN addresses and ARP

each adapter on LAN has unique *LAN* address



# LAN addresses (more)

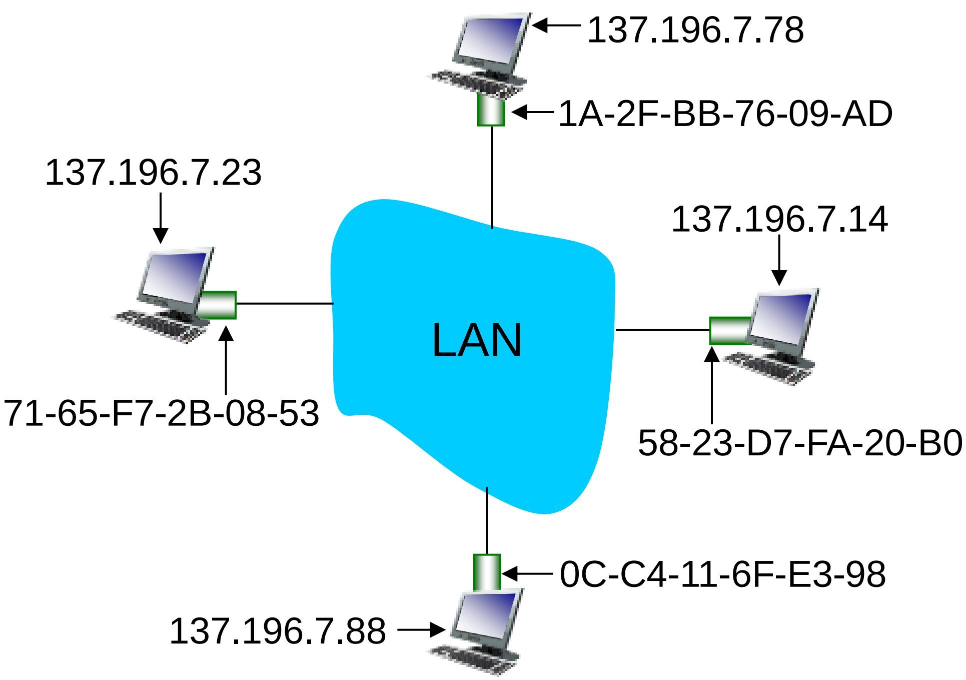
- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like ID Number
  - IP address: like postal address
- MAC flat address
  - can move LAN card from one LAN to another
- IP hierarchical address
  - address depends on IP subnet to which node is attached

# LAN addresses (more)

- an adapter sends a frame to some destination adapter,
  - inserts the destination adapter's MAC address into the frame and then sends the frame into the LAN
- an adapter receive a frame
  - If there is a match, extracts the enclosed datagram and passes the datagram up the protocol stack ;
  - If there isn't a match, discards
- MAC broadcast address FF-FF-FF-FF-FF-FF

# ARP: address resolution protocol

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




*ARP: IP* → MAC

- Resolve addresses only for interfaces on the same subnet

*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)

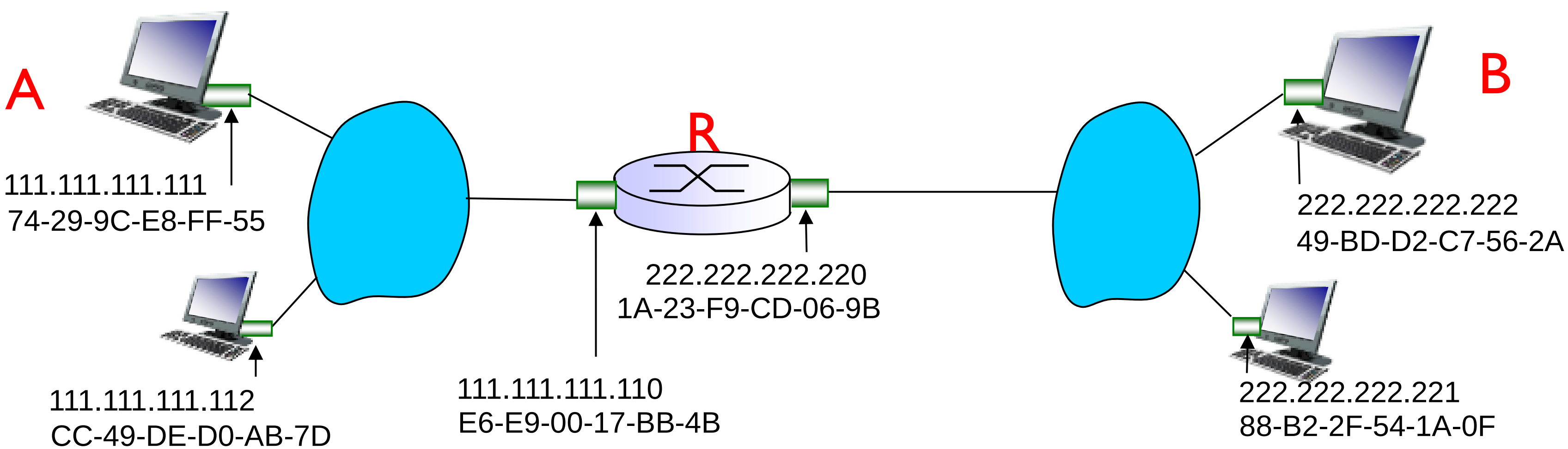
# ARP protocol: same LAN

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

# Addressing: routing to another LAN

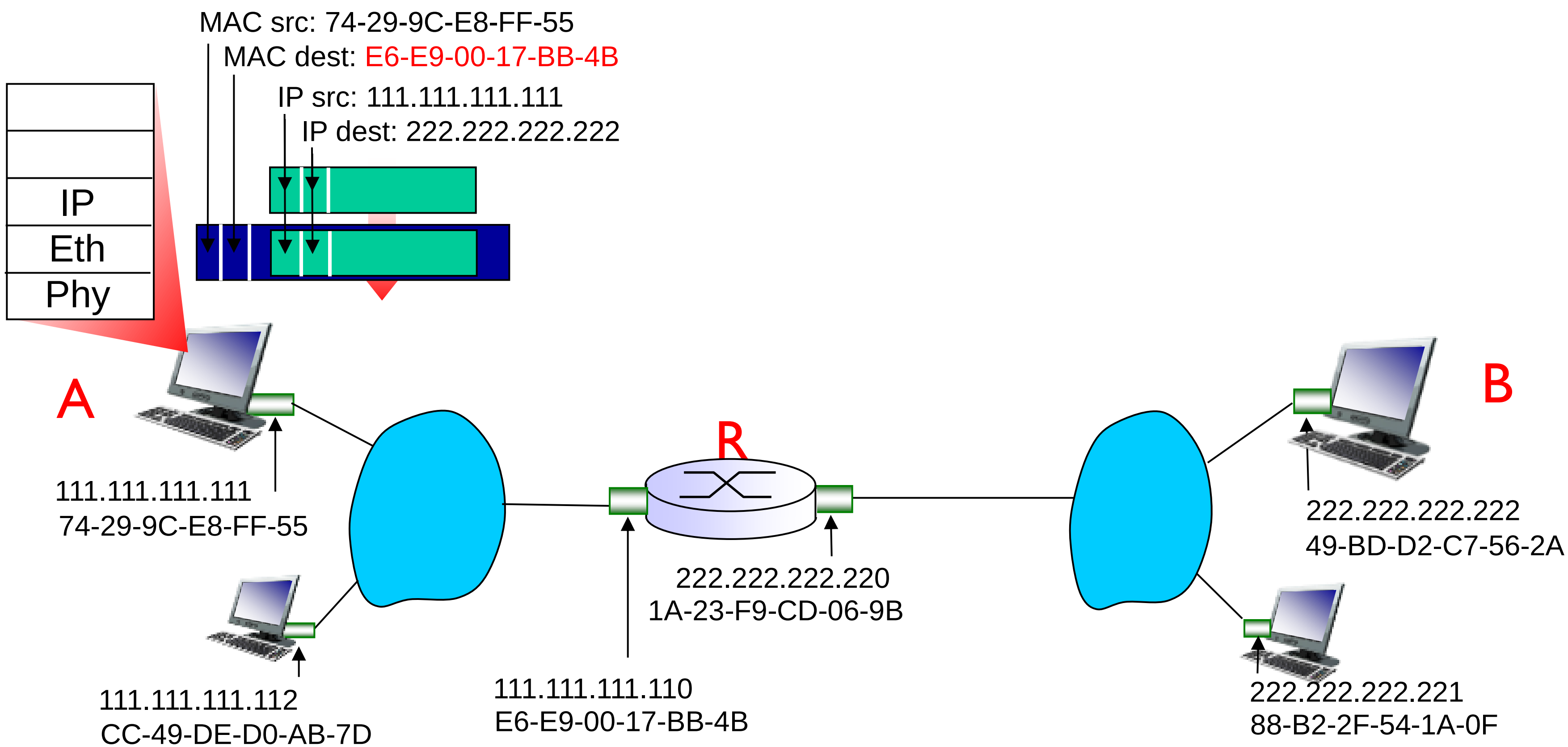
walkthrough: send datagram from A to B via R

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



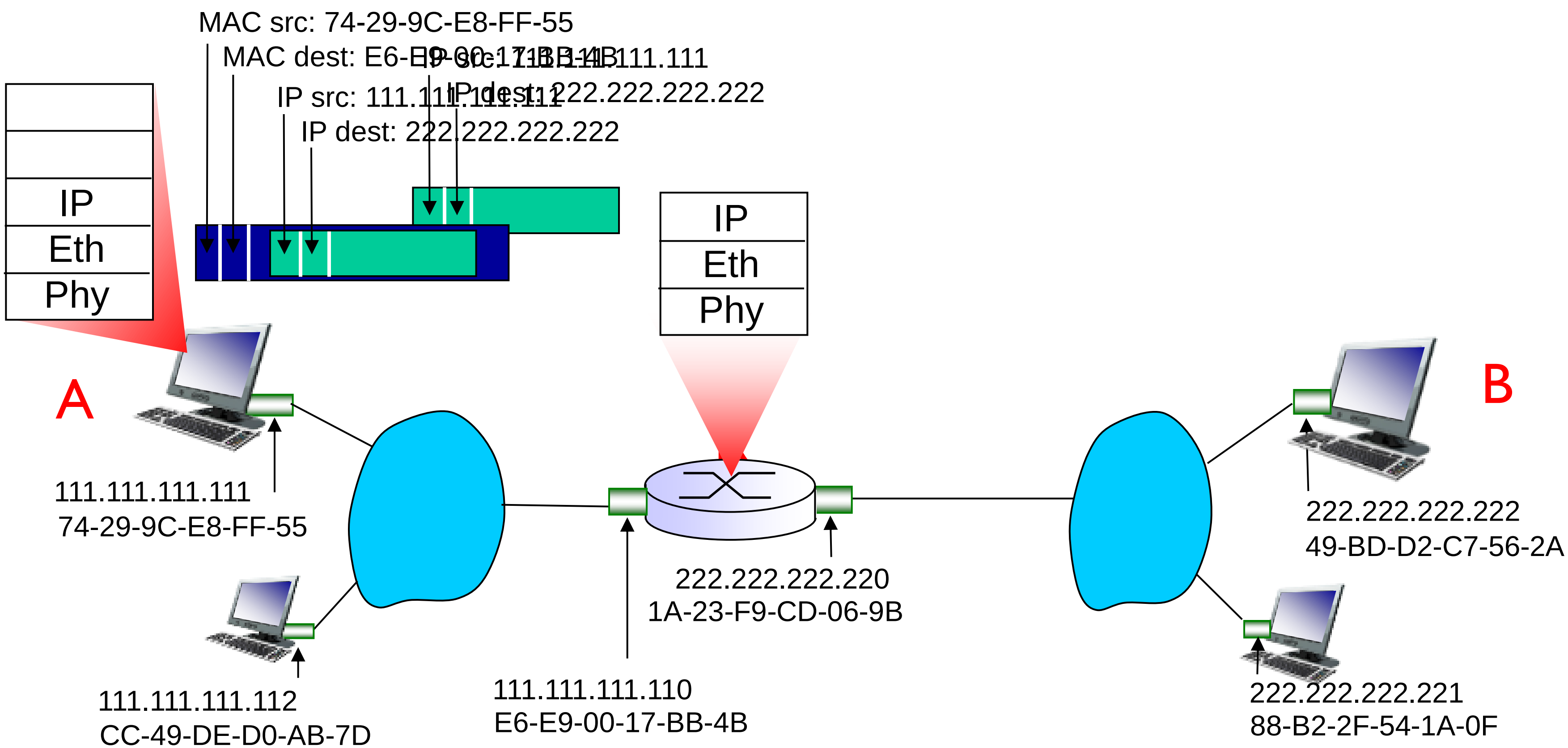
# Addressing: routing to another LAN

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



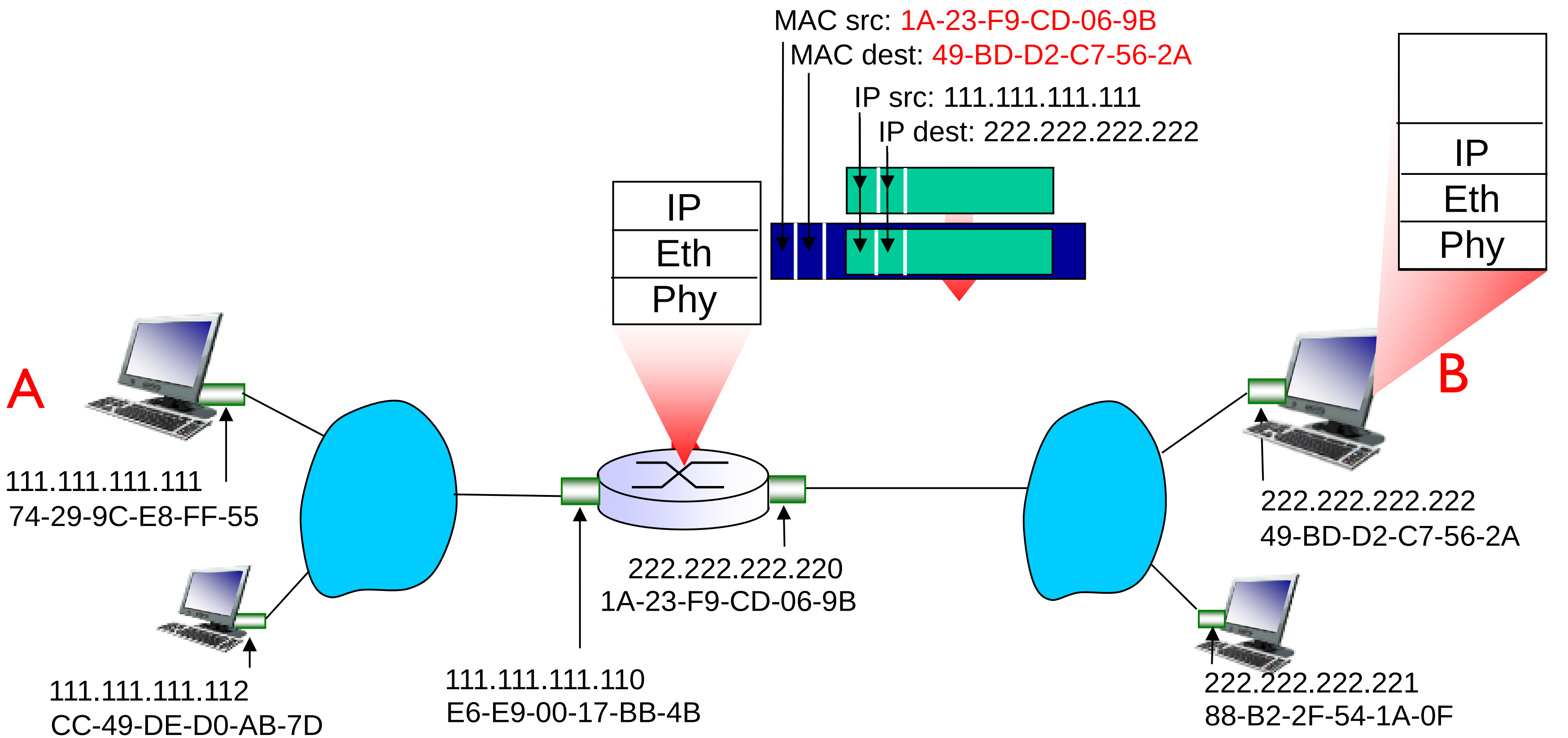
# Addressing: routing to another LAN

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



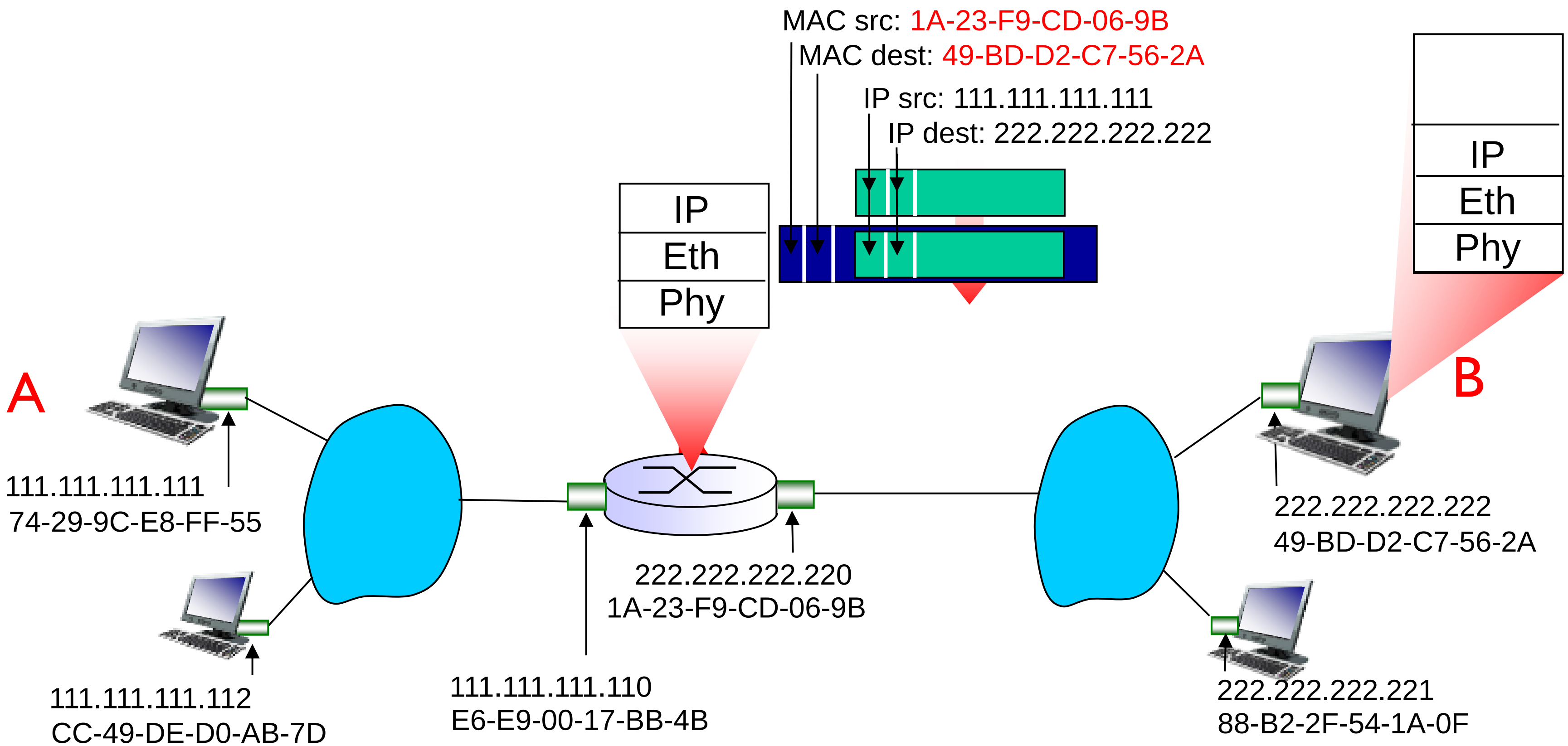
# Addressing: routing to another LAN

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



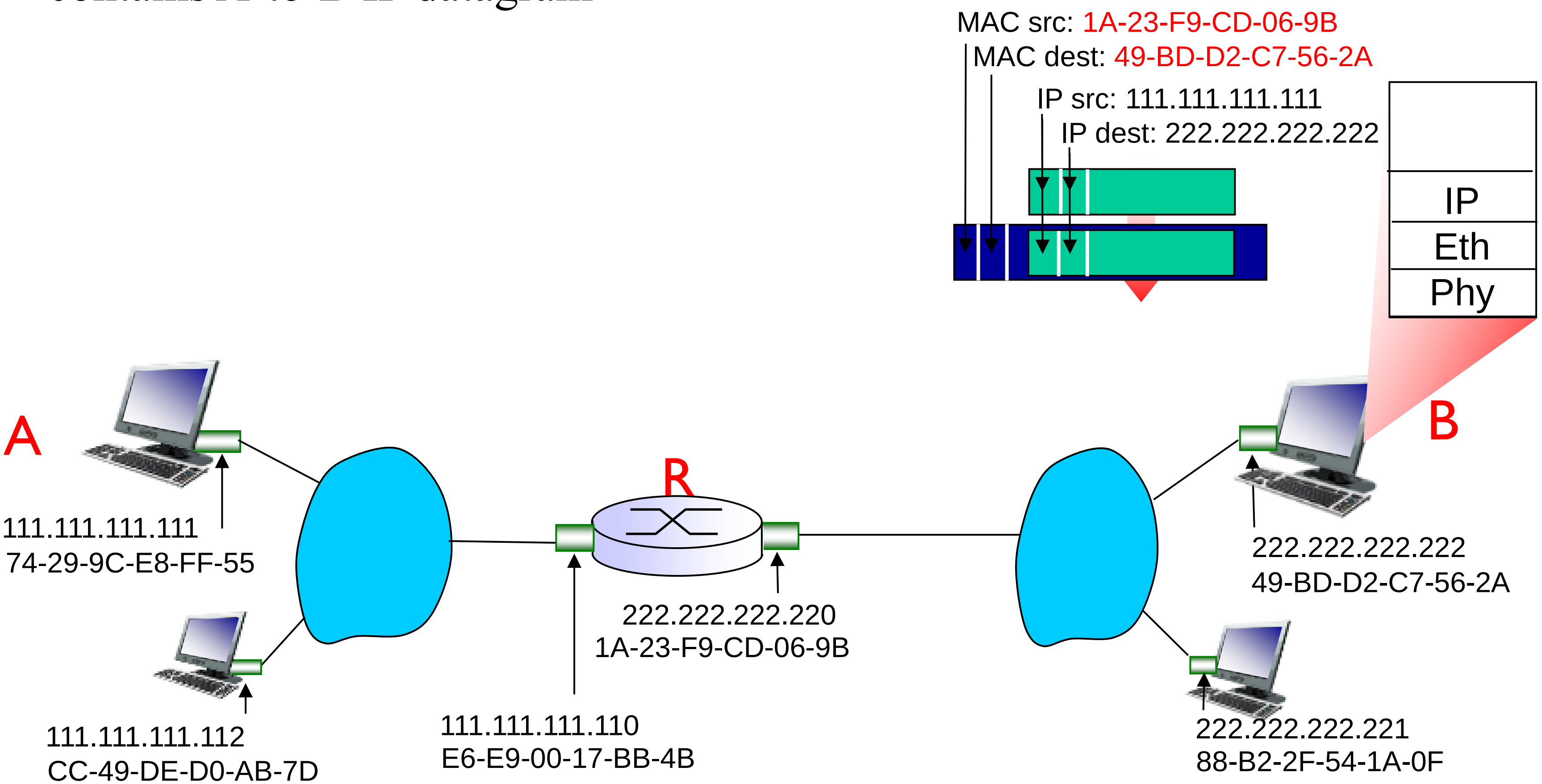
# Addressing: routing to another LAN

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



# Addressing: routing to another LAN

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




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

# Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

6.6 data center networking

6.7 a day in the life of a web request