COMP 3331/9331: Computer Networks and Applications

Week 9
Data link Layer I

Reading Guide: Chapter 6, Sections 6.1 – 6.3

Link layer and LANs

our goals:

- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 Switched LANs
- addressing, ARP
- Ethernet
- Switches
- VLANS (**EXCLUDED**)

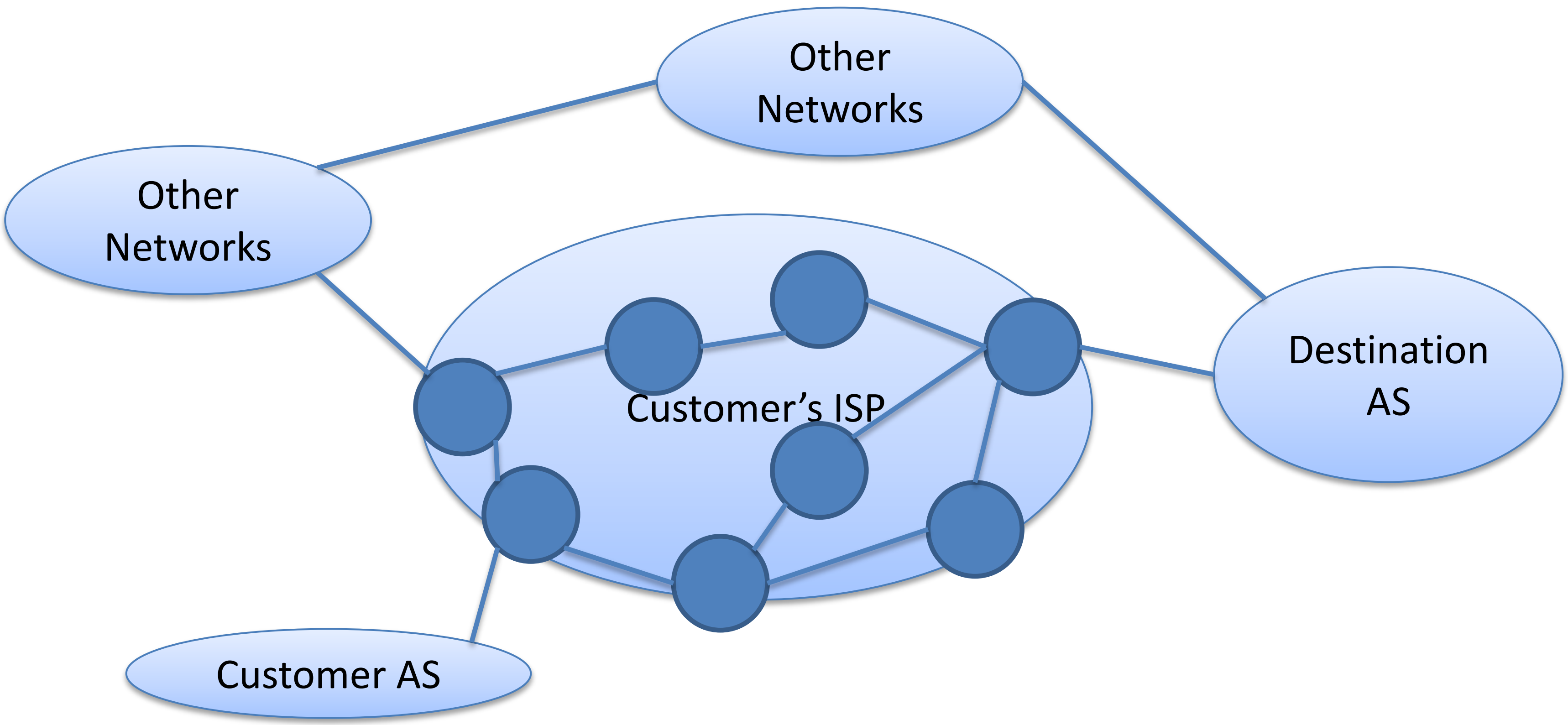
6.5 link virtualization: MPLS (**EXCLUDED**)

6.6 data center networking (**EXCLUDED**)

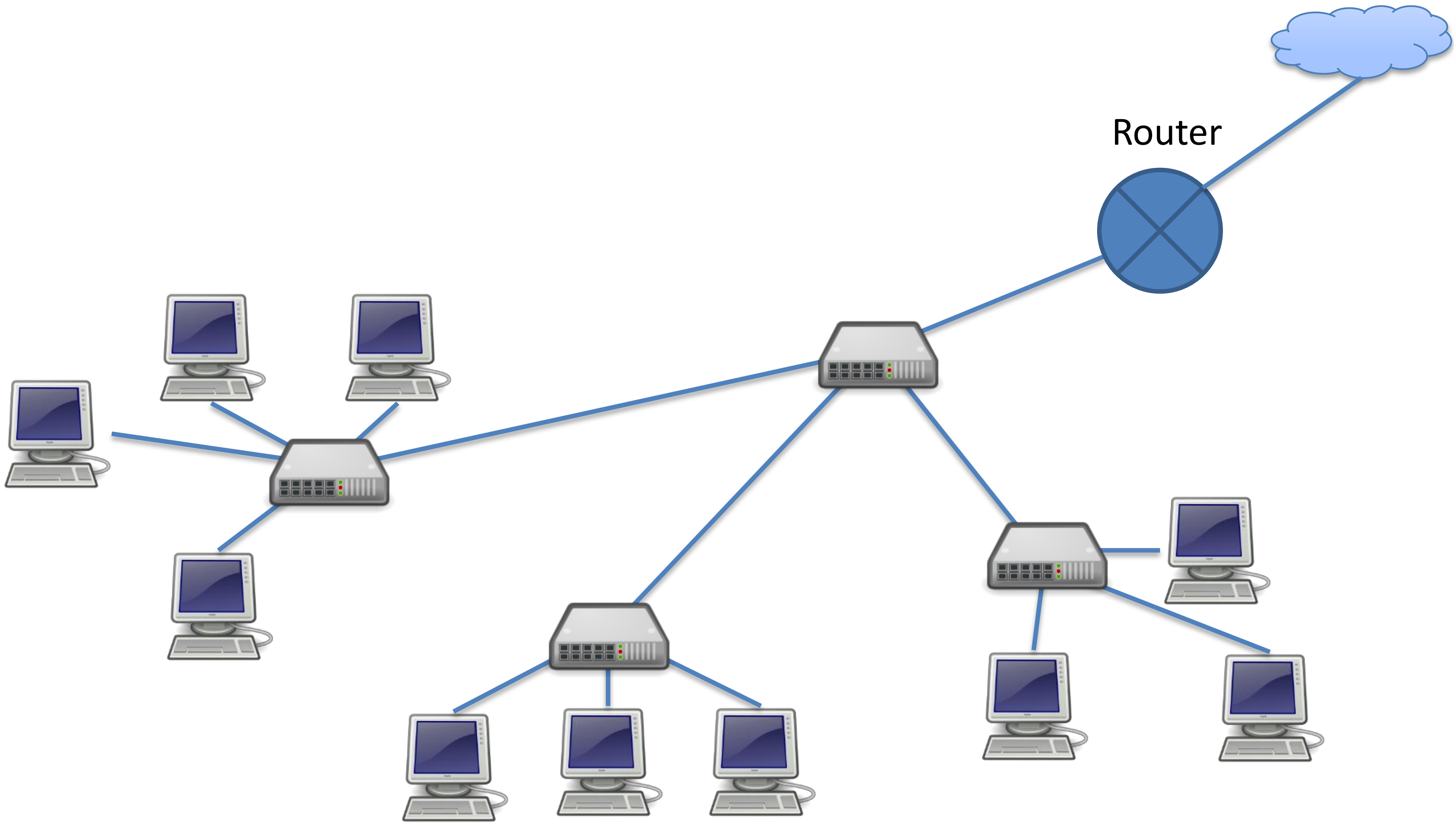
6.7 a day in the life of a web request

From Macro- to Micro-

- Previously, we looked at Internet scale…



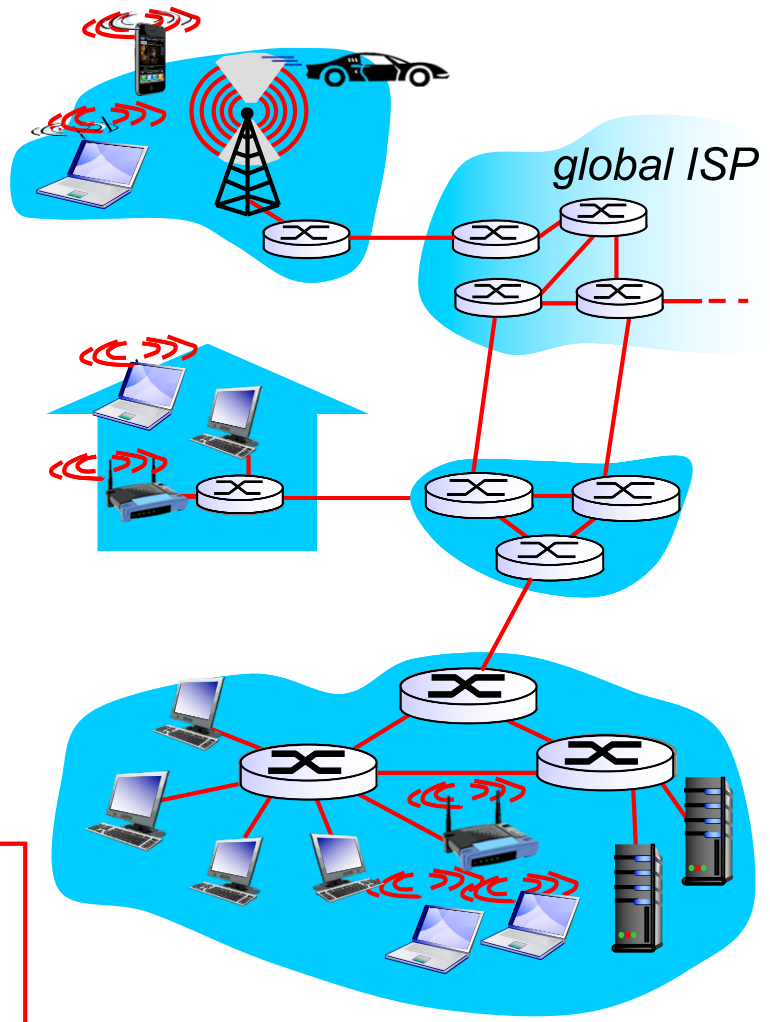
Link layer focus: Within a Subnet



Link layer: introduction

terminology:

- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired links
  - wireless links
  - LANs
- layer-2 packet: frame, encapsulates datagram



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

Link layer: context

- datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, frame relay 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 Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = datagram
- transport segment = communication link
- transportation mode = link layer protocol
- travel agent = routing algorithm

Link layer services

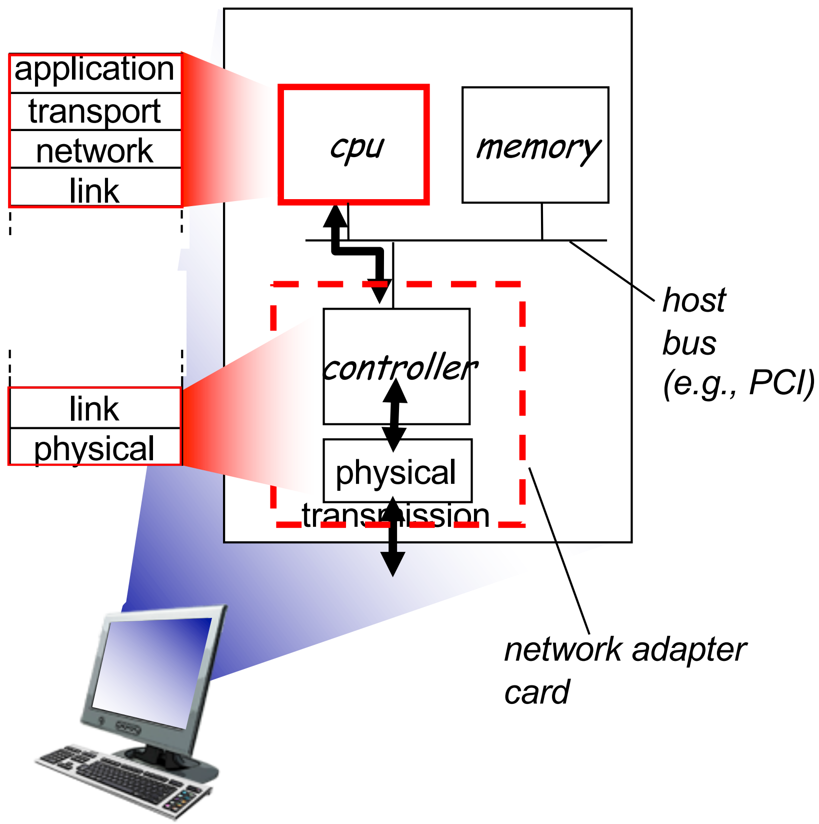
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - we learned how to do this already (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)

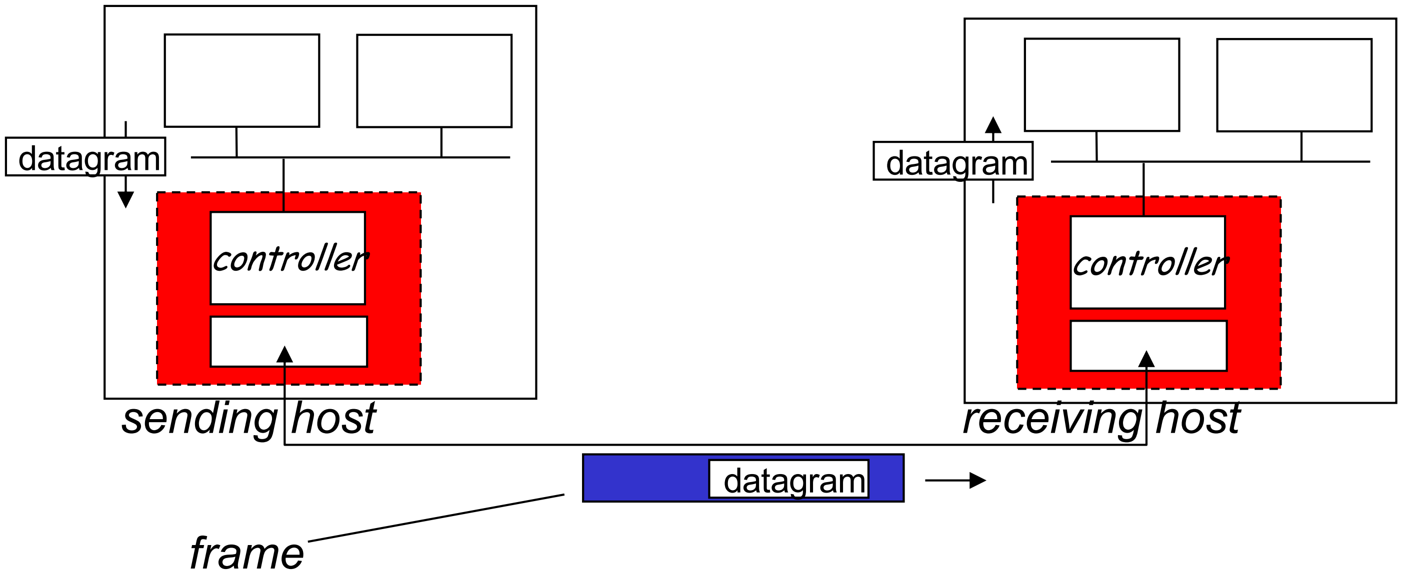
- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection:*
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- *error correction:*
  - receiver identifies *and corrects* bit 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



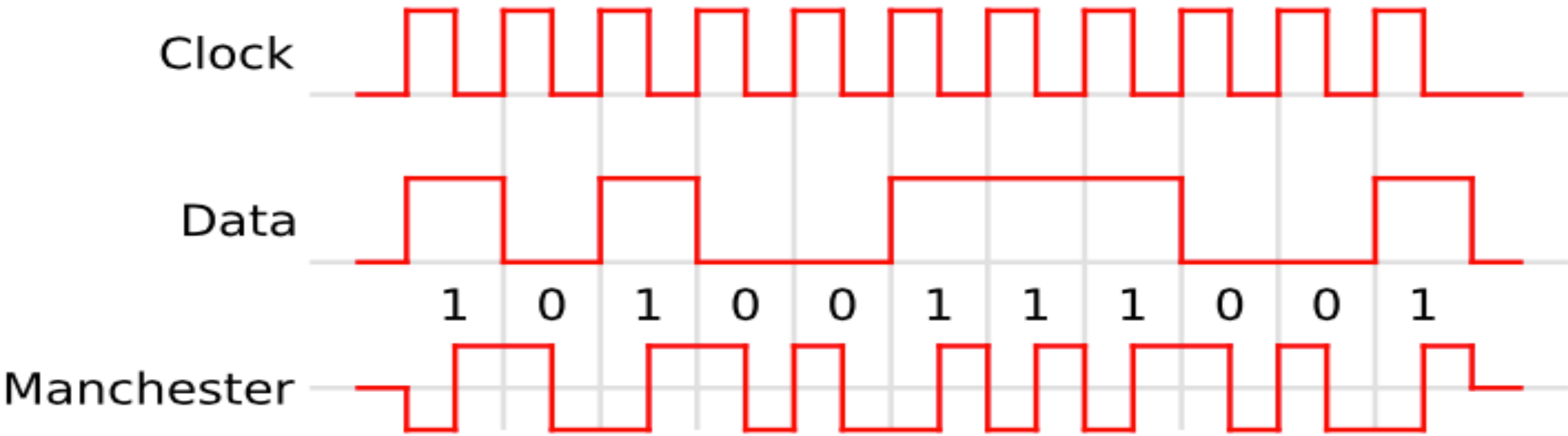
- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

What is framing?

- Physical layer talks in terms of bits.
- How do we identify frames within the sequence of bits?
- Need to do Framing
  - Delimit the start and end of the frame
- Ethernet Framing
  - Timing/Physical layer

Framing in Ethernet

- Start of frame is recognized by
  - Preamble : Seven bytes with pattern 10101010
  - Start of Frame Delimiter (SFD) : 10101011
- End of Frame: Absence of transition in Manchester encoded signal



Framing in Ethernet

- Start of frame is recognized by
  - Preamble : Seven bytes with pattern 10101010
  - Start of Frame Delimiter (SFD) : 10101011
- End of Frame: Absence of transition in Manchester encoded signal

Preamble 7 Bytes	SFD 1 Byte	Dest MAC 6 Bytes	Source MAC 6 Bytes	Type/Length 2 Bytes	Payload 46-1500 Bytes	FCS/CRC 4 Bytes	Inter Frame Gap

- Inter Frame Gap is 12 Bytes (96 bits) of idle state
  - 0.96 microsec for 100 Mbit/s Ethernet
  - 0.096 microsec for Gigabit/s Ethernet

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

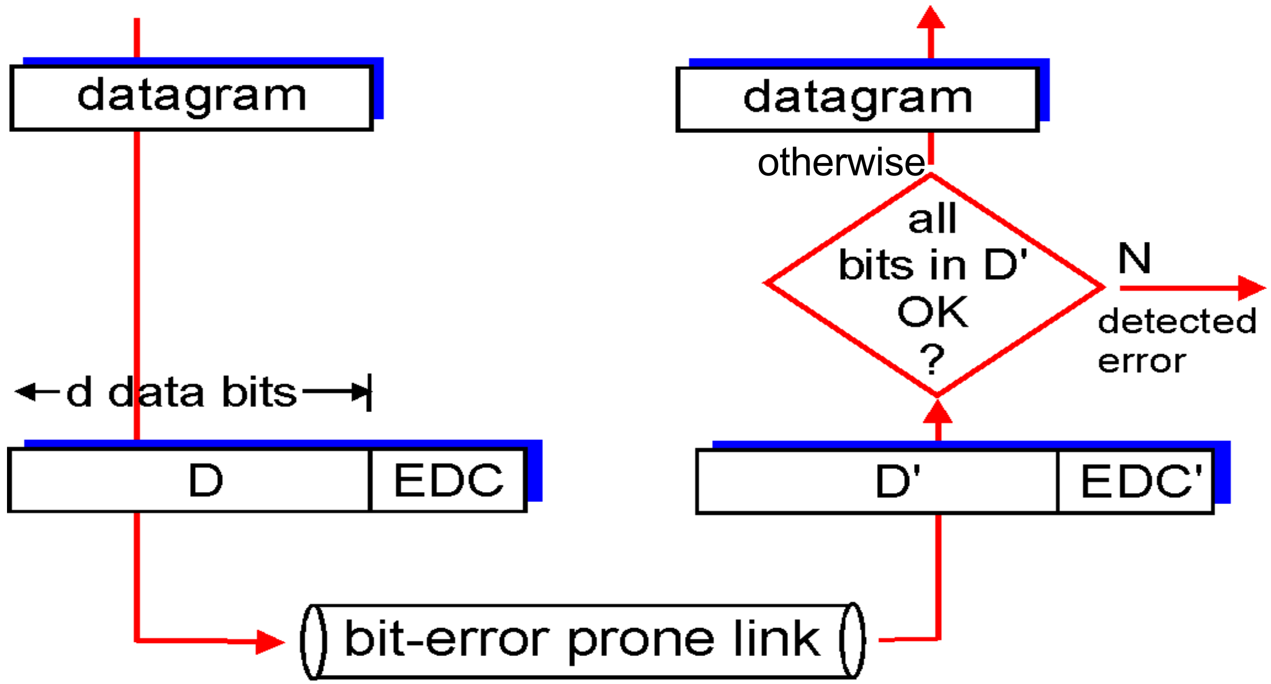
6.4 Switched LANs
- addressing, ARP
- Ethernet
- switches

6.7 a day in the life of a web request

Error detection

EDC= Error Detection and Correction bits (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



Error Detection

- Error coding
- Add check bits to the message bits to let some errors be detected and some be corrected
- How to structure the code to detect many errors with few check bits and modest computation?
- A simple code
  - Send two copies of the same message : 101101
  - Error if the copies are different : 101100
  - How many errors can it correct? 0
  - How many errors can it detect? At most 3
  - How many errors will make it fail? Specific 2 bits errors
  - What is the overhead? 50% (*half of the data contains error check bits*)

Simple Parity - Sender

Example of EVEN Parity

- Suppose you want to send the message:
  - 001011011011000110010
- For every *d* bits (e.g., *d* = 7), add a parity bit:
  - 1 if the number of one's is odd
  - 0 if the number of one's is even

Message chunk	Parity bit
0010110	1
1101100	0
0110010	1

  - 001011011101100001100101

Simple Parity - Receiver

- For each block of size *d*:
  - Count the number of 1's and compare with following parity bit.
- If an odd number of bits get flipped, we'll detect it (can't do much to correct it).
- Cost: One extra bit for every *d*
  - In this example, 21 -> 24 bits.

Two-Dimensional Parity

- Suppose you want to send the same message:
  - 001011011011000110010
- Add an extra parity byte, compute parity on "columns" too.
- Can detect 1, 2, 3-bit (and some 4-bit) errors

	Message chunk	Parity bit
	0010110	1
	1101100	0
	0110010	1
Parity byte:	1001000	0

Forward Error Correction

- With two-dimensional parity, we can even *correct* single-bit errors.

Parity bits

	0	0	1	0	1	1	0	1
	1	0	1	0	0	0	1	0
	1	0	0	1	0	1	1	0
	1	1	1	0	1	1	0	1
Parity byte	1	1	1	1	1	1	0	0

Exactly one bit has been flipped. Which is it?

Row 2 Column 5

In practice

- Bit errors occur in bursts.

- We're willing to trade computational complexity for space efficiency.
  - Make the detection routine more complex, to detect error bursts, without tons of extra data

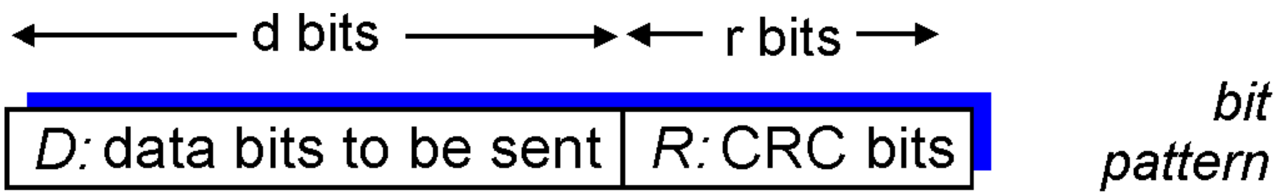
- Insight: We need hardware to interface with the network, do the computation there!

Error Detection and Correction

- Checksum
  - Sum up data in N-bit words
  - Internet Checksum uses 16 bit words
- What have we gained as compared to parity bit?
  - Can now detect all burst errors up to 16

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 burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)



$$D * 2^r \quad \text{XOR} \quad R$$

mathematical formula

Cyclic redundancy check

- Sender operation
  - Extend D data bits with R zeros
  - Divide by generator G
  - Keep remainder, ignore quotient
  - Adjust R check bits by the remainder
- Receiver Procedure
  - Divide and check for zero remainder

CRC example

want:

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

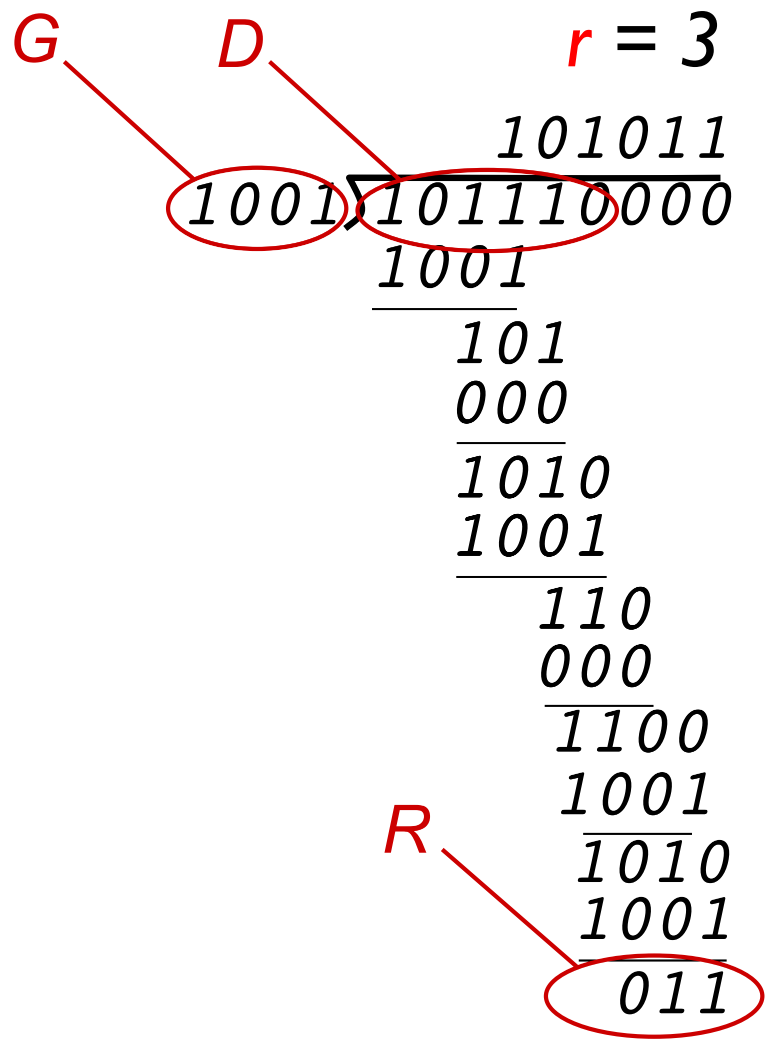
equivalently:

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

equivalently:

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

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



Modulo-2 Arithmetic

- All calculations are modulo-2 arithmetic
- No carries or borrows in subtraction
- Addition and subtraction are identical and both are equivalent to XOR
  - 1011 XOR 0101 = 1110
  - 1011 - 0101 = 1110
  - 1011 + 0101 = 1110
- Multiplication by 2^k is essentially a left shift by *k* bits
  - 1011 x 2^2 = 101100

Quiz: Error Detection/Correction

- Can these schemes, respectively, correct bit errors: Internet checksums, two-dimensional parity, cyclic redundancy check (CRC)?

a) Yes, No, No
b) No, Yes, Yes
c) No, Yes, No
d) No, No, Yes
e) No, No, No

C

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 Switched LANs

- addressing, ARP
- Ethernet
- switches

6.7 a day in the life of a web request

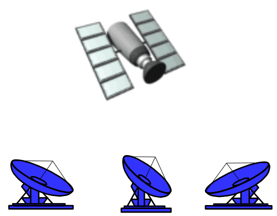
Multiple access links, protocols

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
  - upstream HFC
  - 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 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

Requirements of an idea protocol:

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

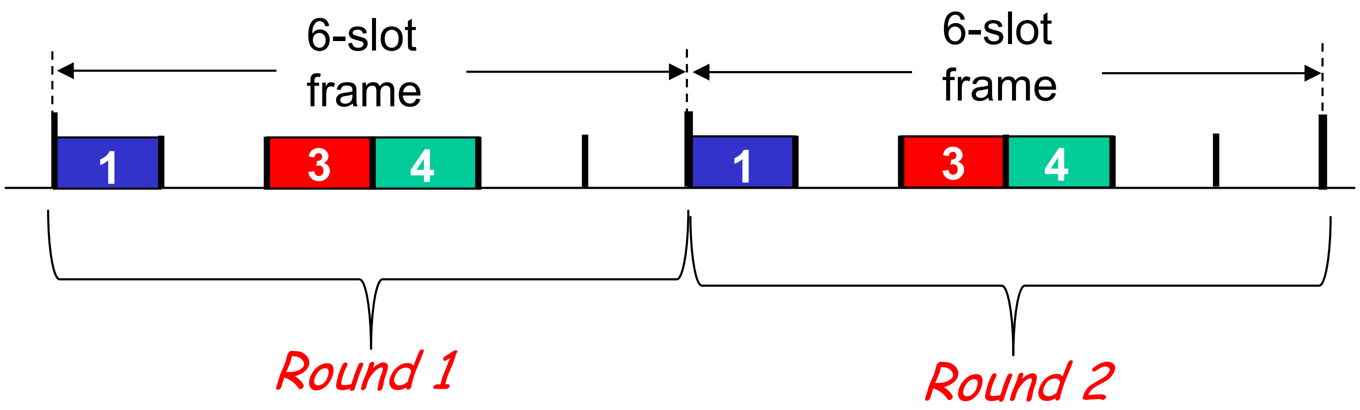
three broad classes:

- *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

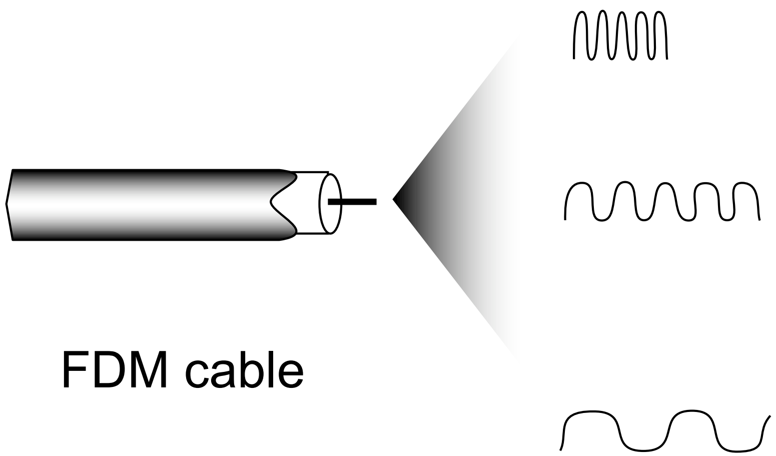
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle

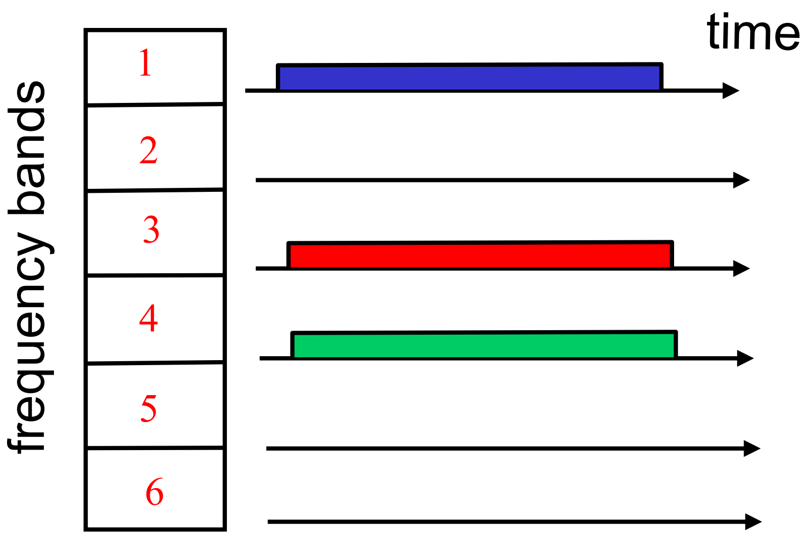


Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

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





Quiz: Does *channel partitioning* satisfy ideal properties ?

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, slots
   - no special node to coordinate transmissions
4. simple

A. 0
B. 1
C. 2
D. 3
E. 4

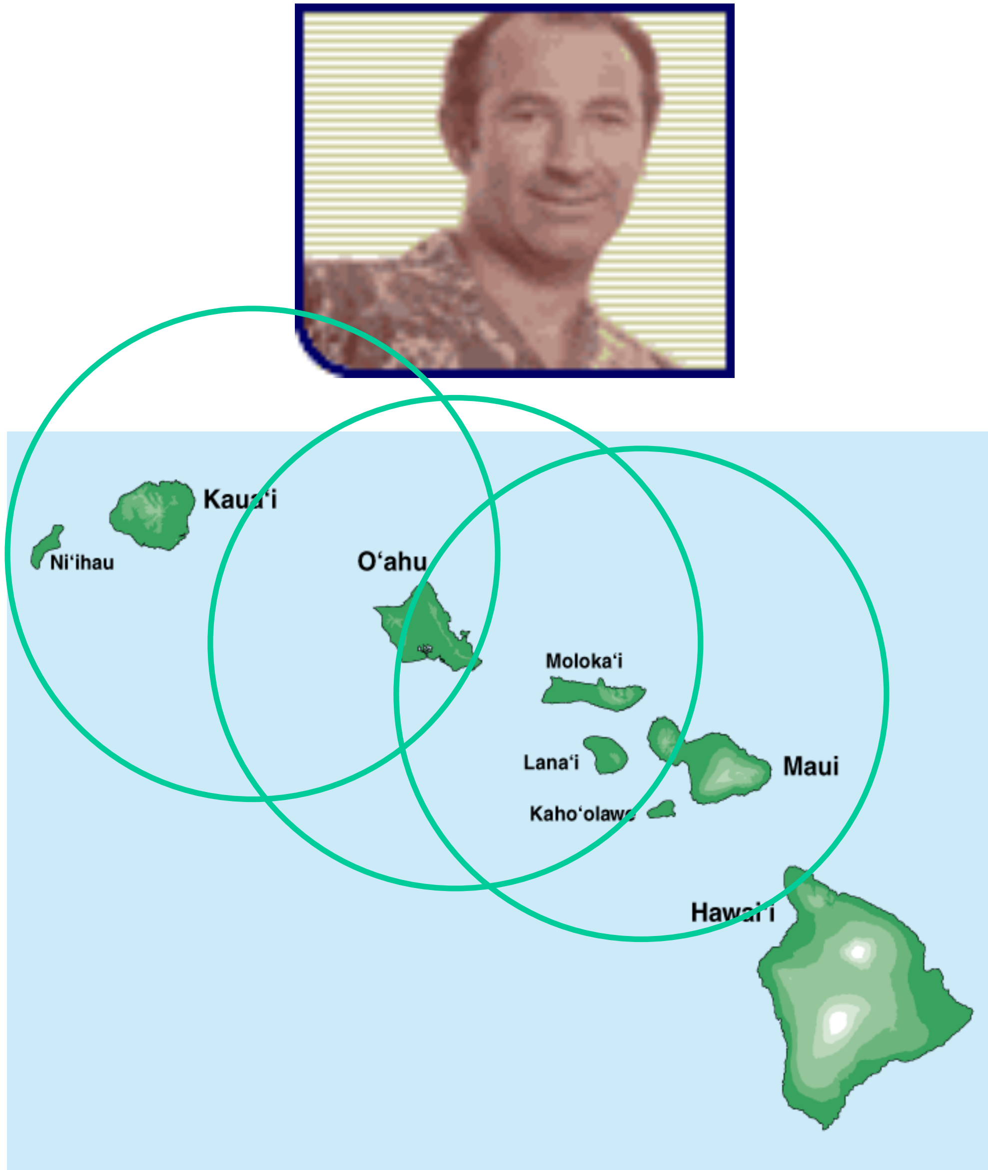
(Which ones?)

E 4(it is simple, but other requirements are not really fulfilled)

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:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

Where it all Started: AlohaNet



- Norm Abramson left Stanford in 1970 (***so he could surf!***)
- Set up first data communication system for Hawaiian islands
- Central hub at U. Hawaii, Oahu

Slotted ALOHA

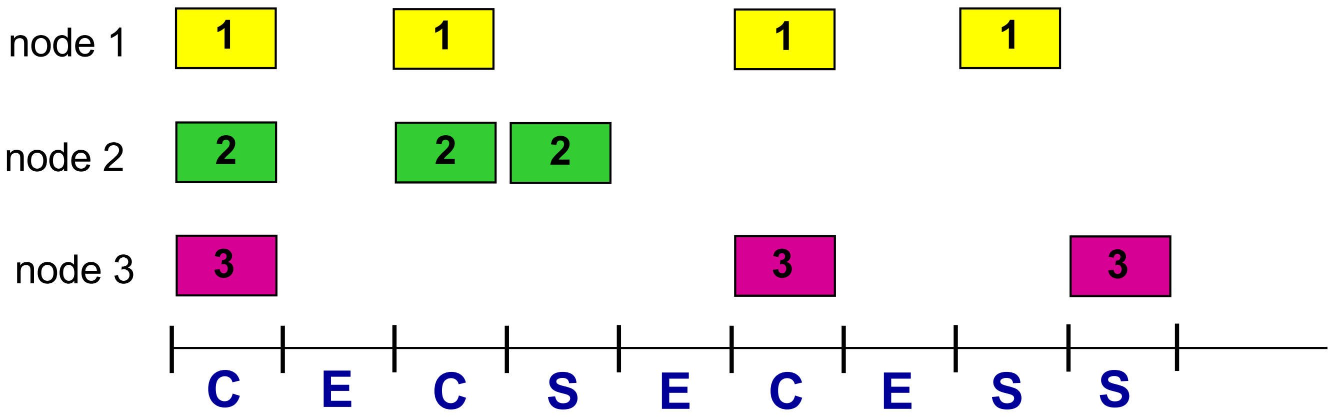
assumptions:

- 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 prob. 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
- 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:

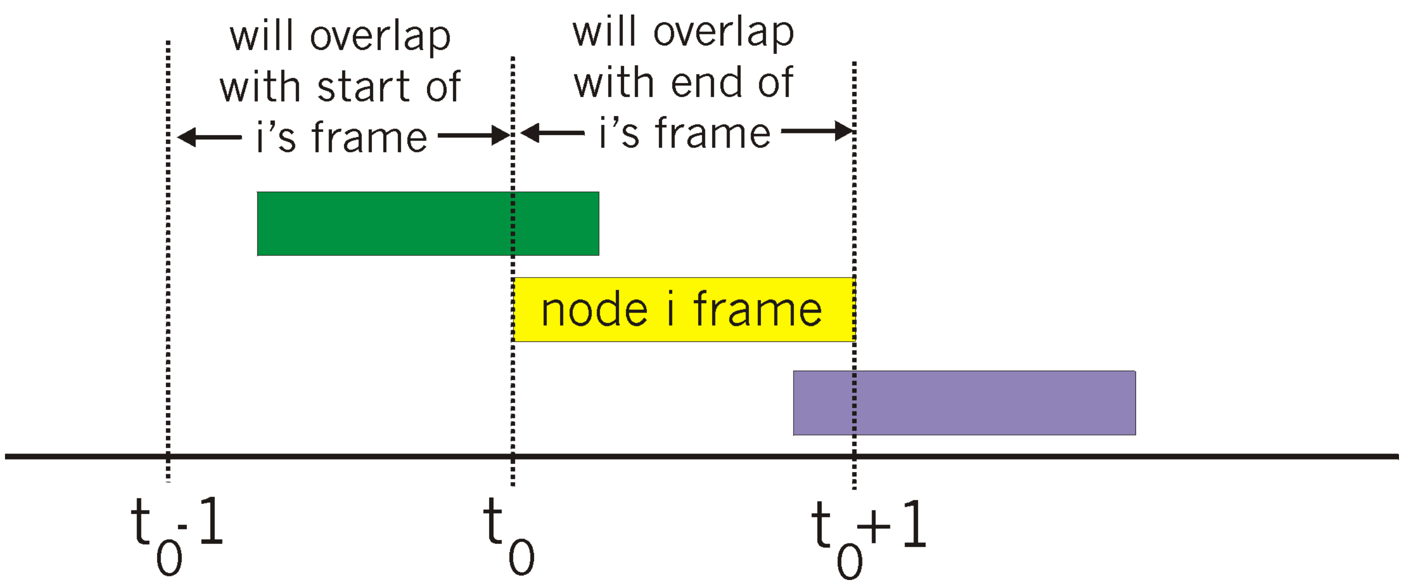
max efficiency = 1/e = .37

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

!

Pure (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) $\cdot$

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

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

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

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

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

$$= 1/(2e) = .18$$

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!

- Does this eliminate all collisions?
  - No, because of nonzero propagation delay

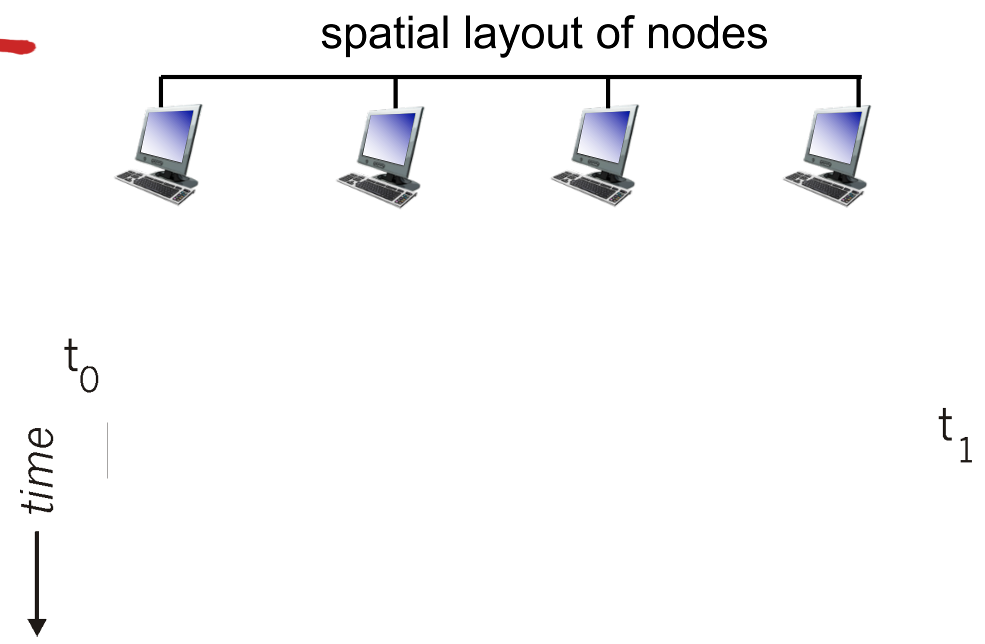
CSMA collisions

- 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 reduces but does not eliminate collisions

Biggest remaining problem?

Collisions still take full slot!



CSMA/CD (collision detection)

CSMA/CD: carrier sensing, deferral as in CSMA

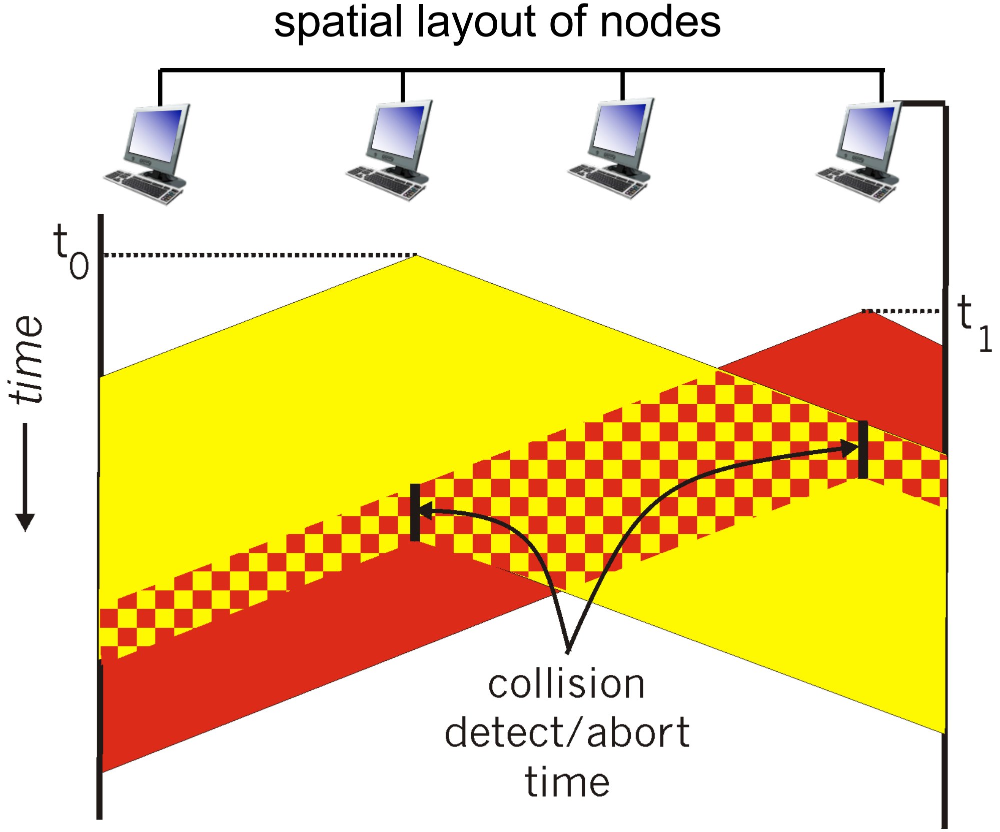
- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- 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
- human analogy: the polite conversationalist

CSMA/CD (collision detection)

Note: for this to work, need restrictions on minimum frame size and maximum distance.

Why?



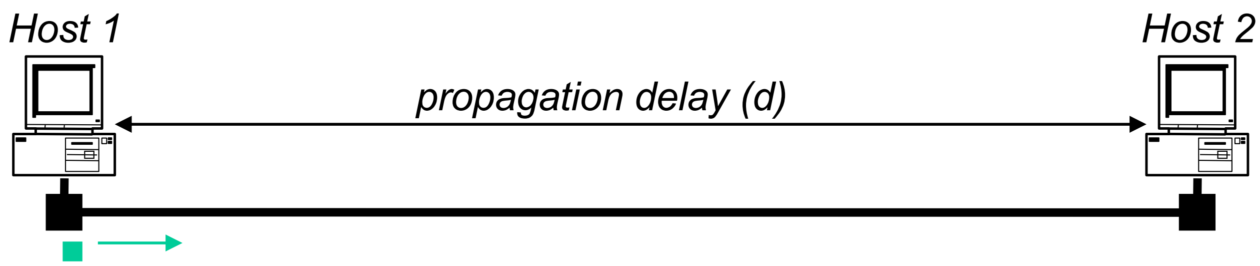
http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/csmacd/csmacd.html

Minimum Packet Size

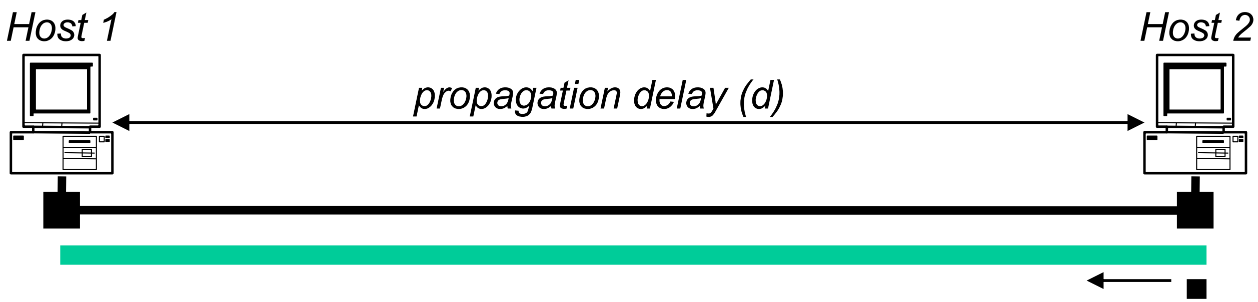
- Why enforce a minimum packet size?
- Give a host enough time to detect collisions
- In Ethernet, minimum packet size = 64 bytes (two 6-byte addresses, 2-byte type, 4-byte CRC, and 46 bytes of data)
- If host has less than 46 bytes to send, the adaptor pads (adds) bytes to make it 46 bytes
- What is the relationship between minimum packet size and the length of the LAN?

Limits on CSMA/CD Network Length

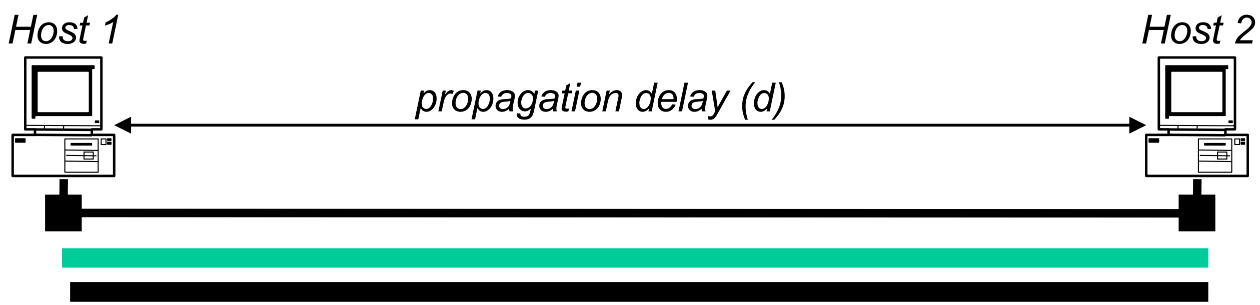
a) Time = t; Host 1 starts to send frame



b) Time = t + d; Host 2 starts to send a frame, just before it hears from host 1's frame



*c) Time = t + 2*d; Host 1 hears Host 2's frame → detects collision*



LAN length = (min_frame_size)(propagation_speed)/(2*bandwidth) =*
*= (8*64B)*(2*10^8mps)/(2*10^7 bps) = 5120m approx*

What about 100 mbps? 1 gbps? 10 gbps?

Ethernet CSMA/CD algorithm

1. 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

Quiz: Does CSMA/CD satisfy ideal properties ?

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, slots
   - no special node to coordinate transmissions
4. simple

A. 0
B. 1
C. 2
D. 3
E. 4

(Which ones?)

"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

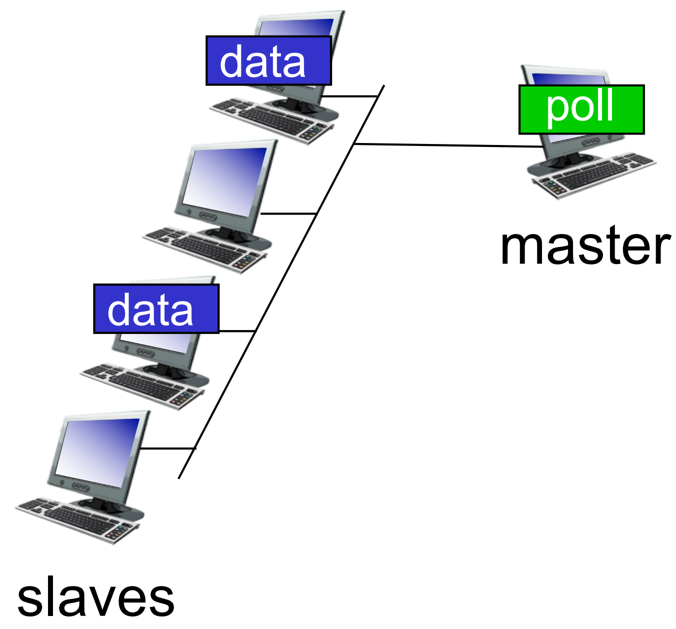
"taking turns" protocols

look for best of both worlds!

"Taking turns" MAC protocols

polling:

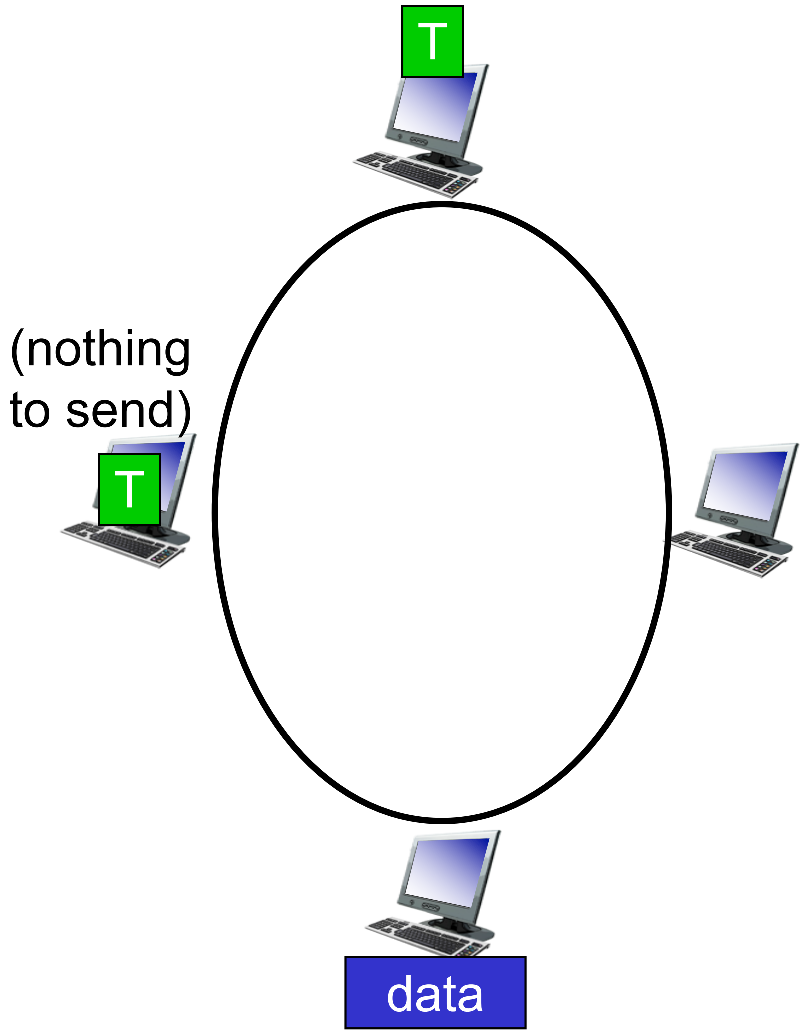
- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



“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)



Quiz: Does taking turns satisfy ideal properties ?

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, slots
   - no special node to coordinate transmissions
4. simple

A. 0
B. 1
C. 2
D. 3
E. 4

(Which ones?)

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