Data Link Layer

Primarily taken from the slides accompanying the textbook by Kurose and Ross

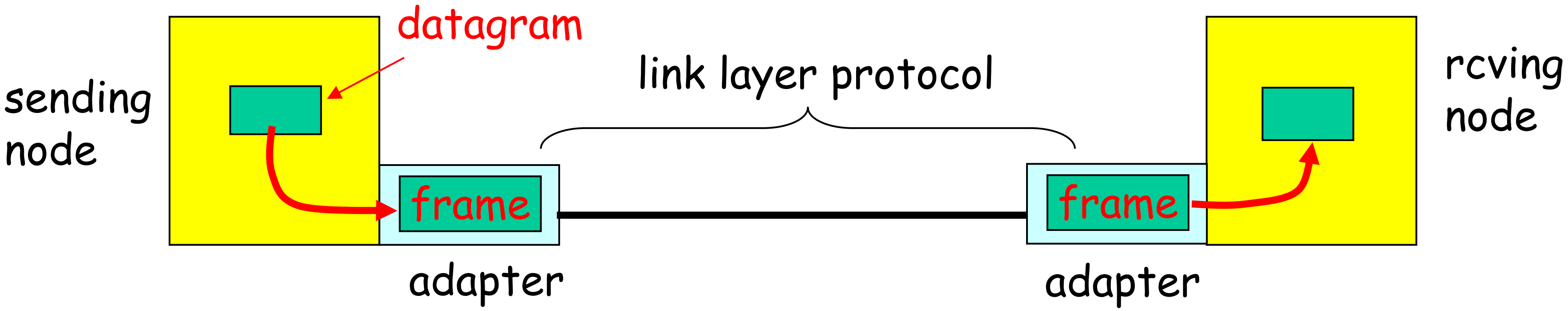
Link Layer Services

- 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

Adaptors Communicating

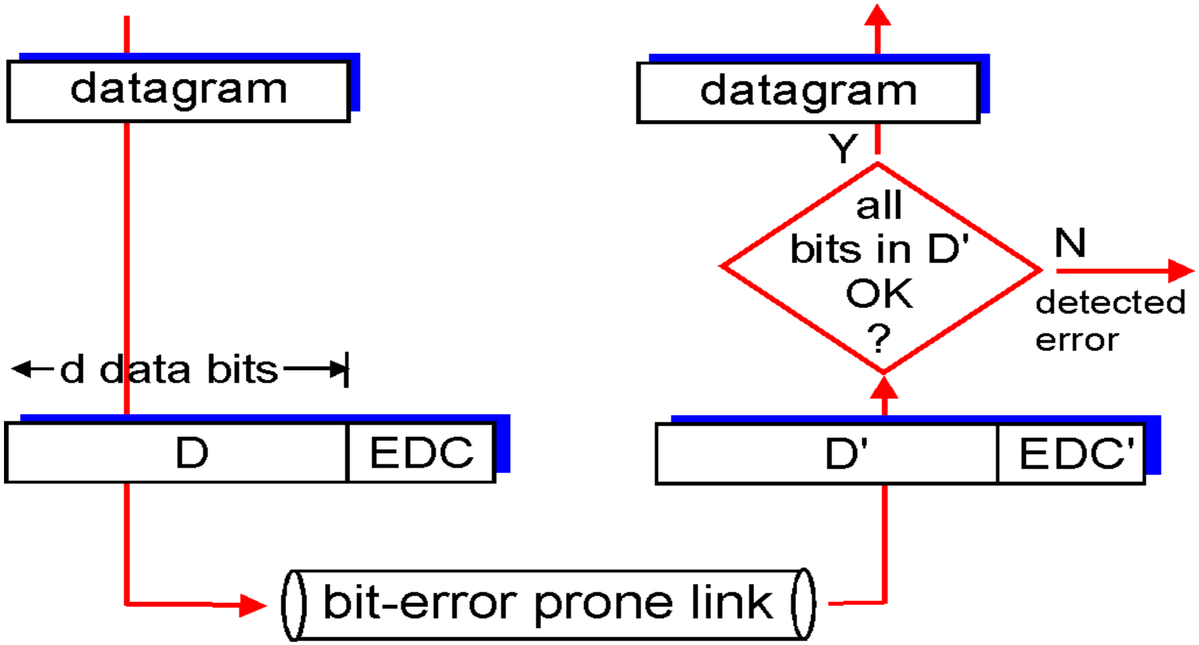


- link layer implemented in "adaptor" (aka NIC)
  - Ethernet card, PCMCIA card, 802.11 card
- sending side:
  - encapsulates datagram in a frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to rcving node
- adapter is semi-autonomous
- link & physical layers

Error Detection

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

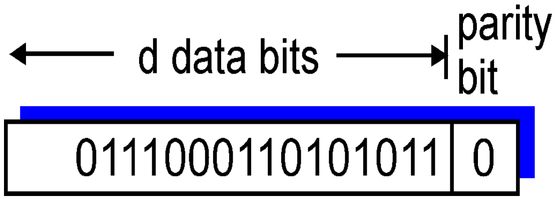
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



Parity Checking

Single Bit Parity:

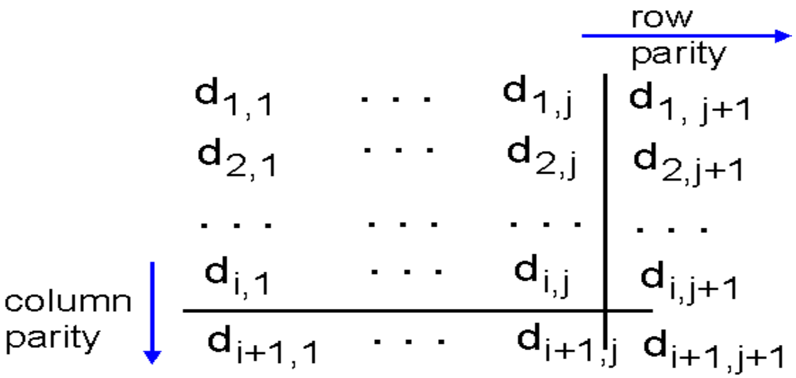
Detect single bit errors

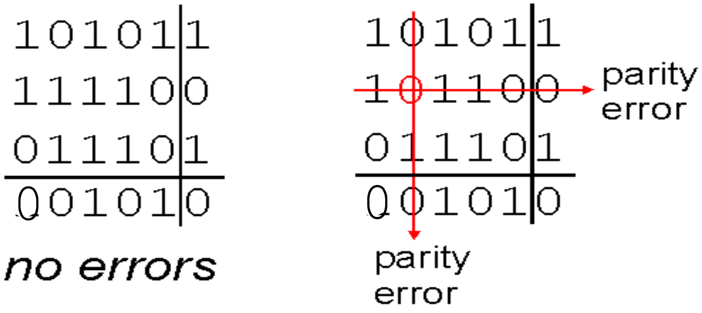


Odd parity scheme

Two Dimensional Bit Parity:

Detect *and correct* single bit errors


row parity
column parity
$d_{1,1}$ . . . $d_{1,j}$ $d_{1,j+1}$
$d_{2,1}$ . . . $d_{2,j}$ $d_{2,j+1}$
. . . .
$d_{i,1}$ . . . $d_{i,j}$ $d_{i,j+1}$
$d_{i+1,1}$ . . . $d_{i+1,j}$ $d_{i+1,j+1}$

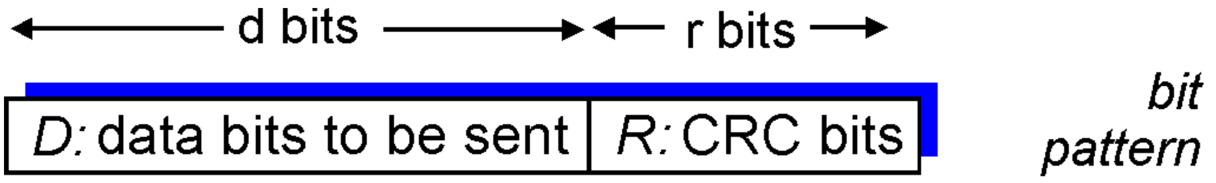


correctable single bit error

Even parity scheme

Checksumming: Cyclic Redundancy Check

- 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, HDLC)



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

mathematical formula

CRC Example: How to find R?

Want:

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

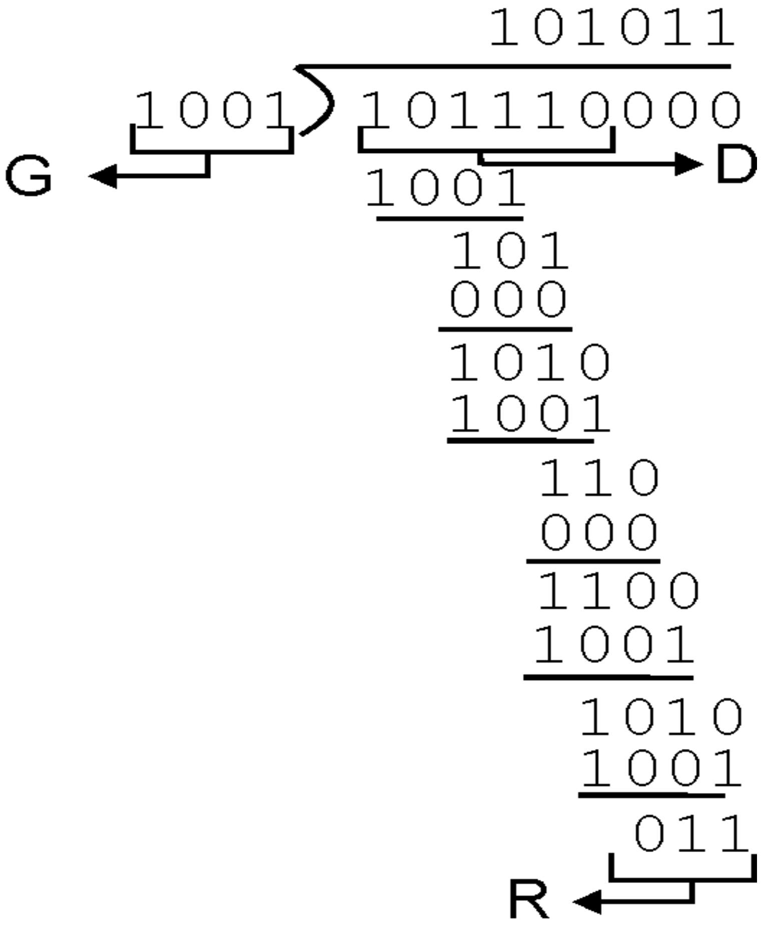
equivalently:

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

equivalently:

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

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



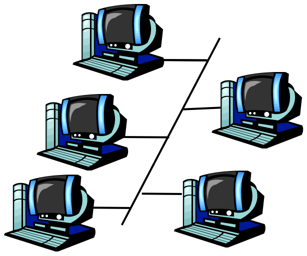
What errors can be detected?

- CRC can detect:
  - All single-bit errors
  - All double-bit errors as long as G (generator) has at least three 1s
  - Any odd number of bit errors as long as "11" divides into G. i.e., generator polynomial G(x) has x+1 as a factor
  - Any burst of length smaller than length of G(x)
  - "Most" larger burst errors
- See: Analysis of CRC [not in Kurose/Ross]

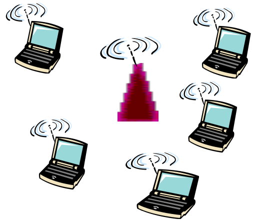
Multiple Access Links and Protocols

Two types of “links”:

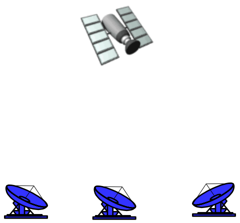
- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch and host
- broadcast (shared wire or medium)
  - old-fashioned Ethernet
  - upstream HFC
  - 802.11 wireless LAN



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

Ideal Multiple Access Protocol

Broadcast channel 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: a taxonomy

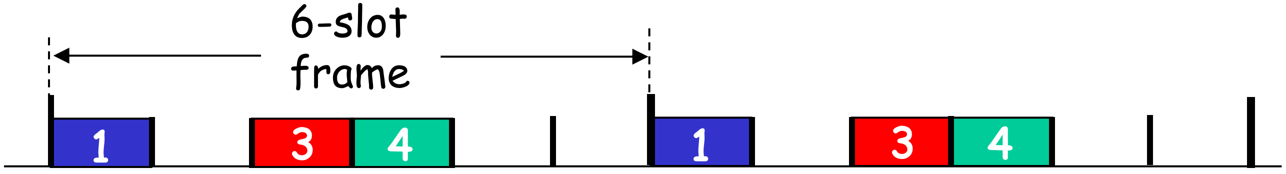
Three broad classes:

- Channel Partitioning
  - divide channel into smaller “pieces” (time slots, frequency, code) e.g., TDMA, FDMA, CDMA
  - allocate piece to node for exclusive use
- Random Access
  - channel not divided, allow collisions
  - “recover” from collisions
  - Examples: slotted ALOHA, ALOHA, CSMA/CD
- “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

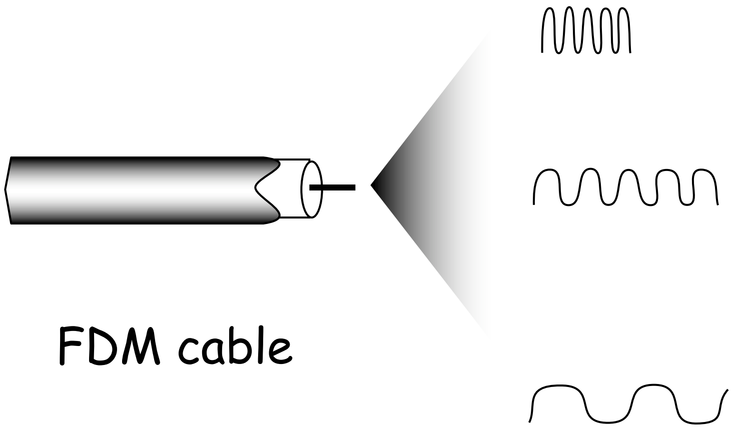
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle

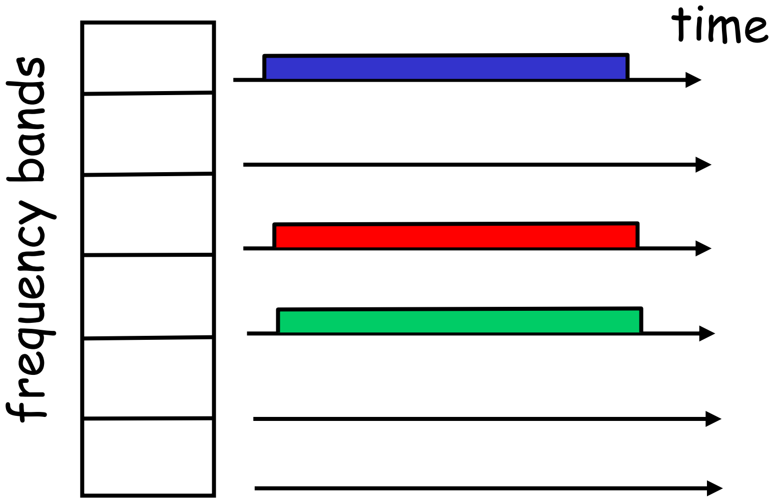


Channel Partitioning MAC protocols: FDMA

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





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

Slotted ALOHA

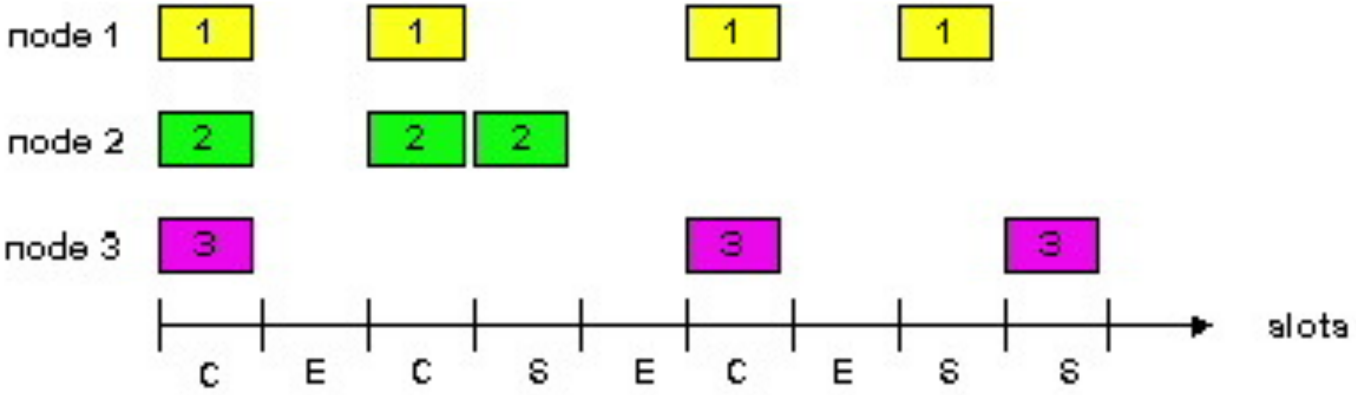
Assumptions

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

Operation

- when node obtains fresh frame, it transmits in next slot
- 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 is the long-run fraction of successful slots when there are many nodes, each with many frames to send

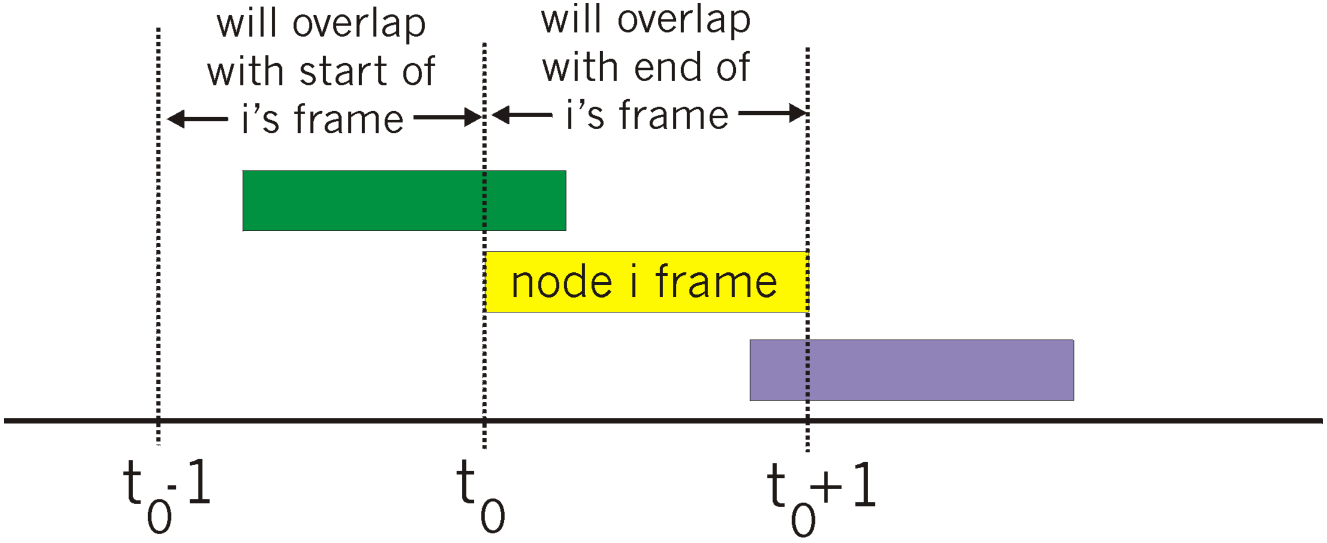
- Suppose N nodes with many frames to send, each transmits in slot with probability *p*
- prob that node 1 has success in a slot $= p(1-p)^{N-1}$
- prob that any node has a success $= Np(1-p)^{N-1}$

- For max efficiency with N nodes, 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 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 $[t_0-1,t_0+1]$


will overlap with start of i's frame
will overlap with end of i's frame
node i frame
t_0-1
t_0
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 $[t_0,t_0+1]$

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

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

… choosing optimum p and then letting n -> 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!

CSMA collisions

collisions *can* still occur:

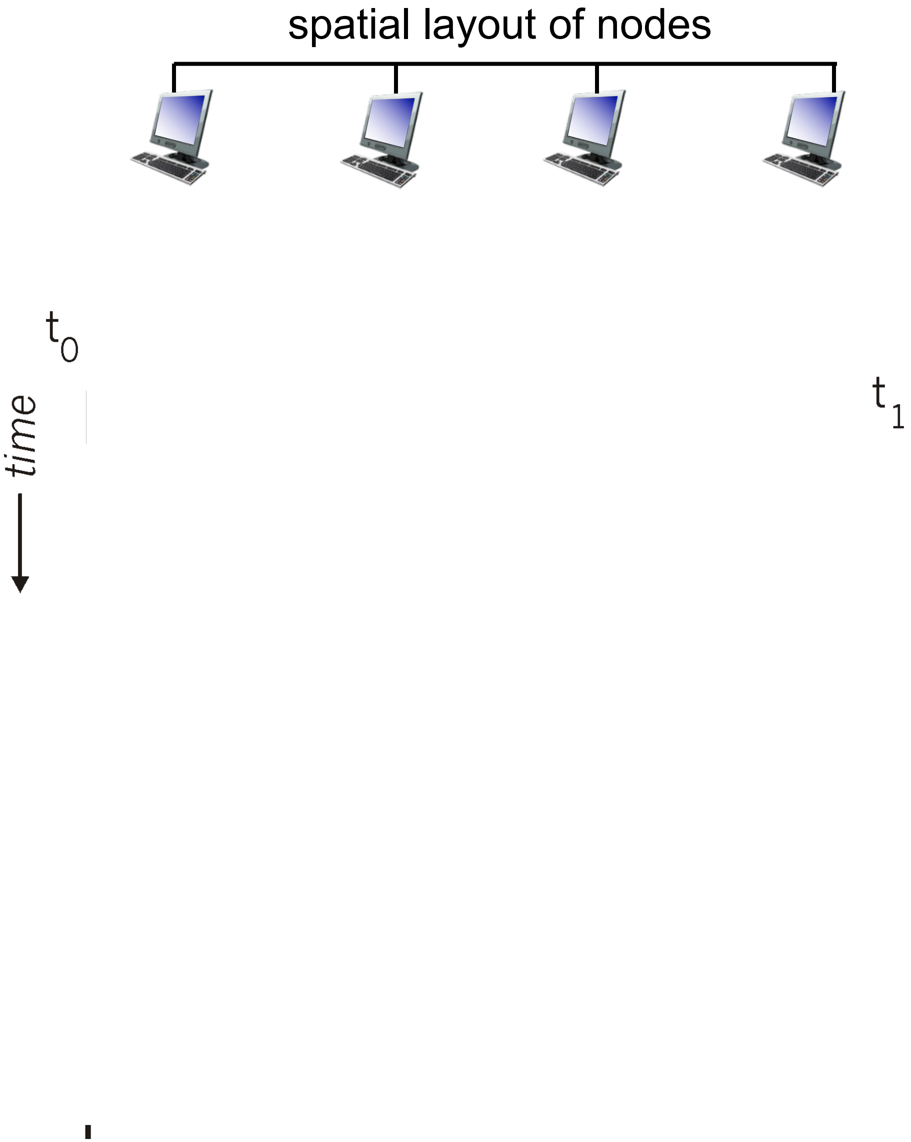
propagation delay means two nodes may not hear each other's transmission

collision:

entire packet transmission time wasted

note:

role of distance & propagation delay in determining collision probability



CSMA collisions

collisions *can* still occur:

propagation delay means two nodes may not hear each other's transmission

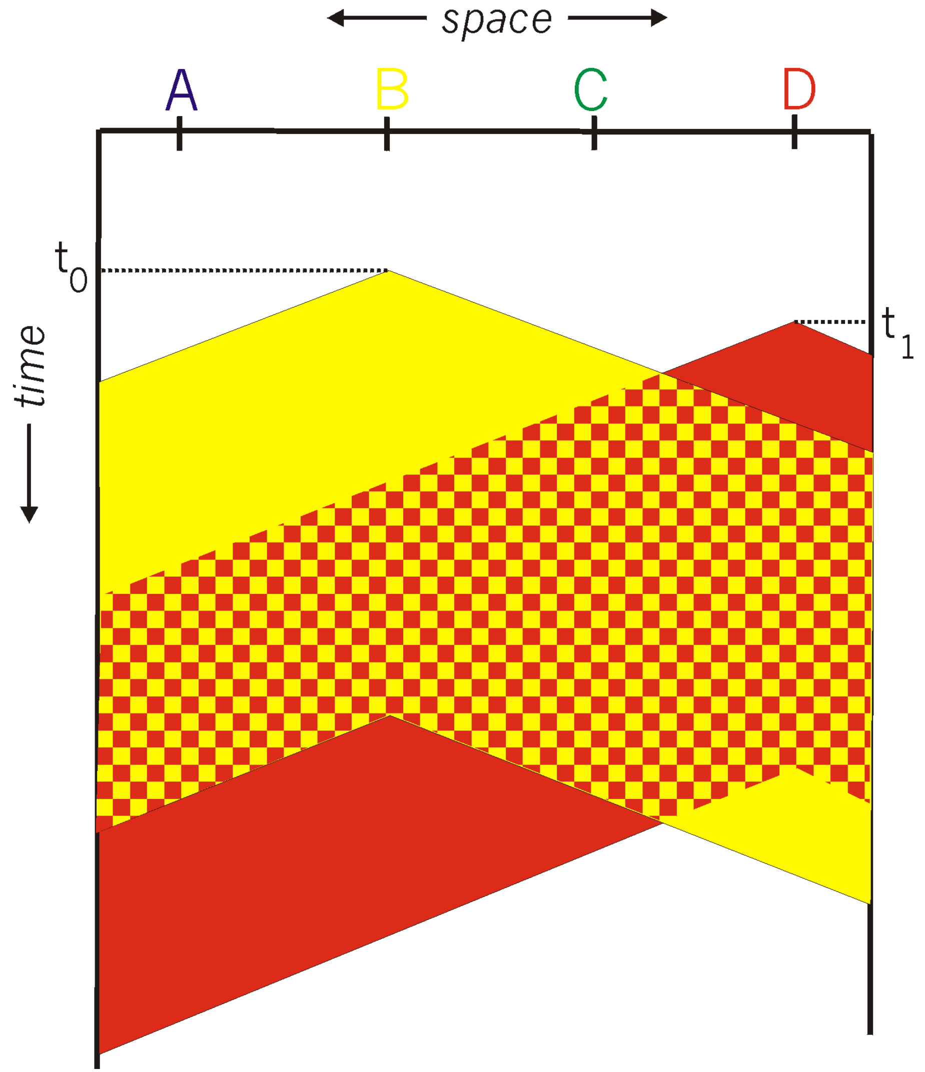
collision:

entire packet transmission time wasted

note:

role of distance & propagation delay in determining collision probability

spatial layout of nodes



CSMA/CD (Collision Detection)

CSMA/CD: carrier sensing, deferral as in CSMA

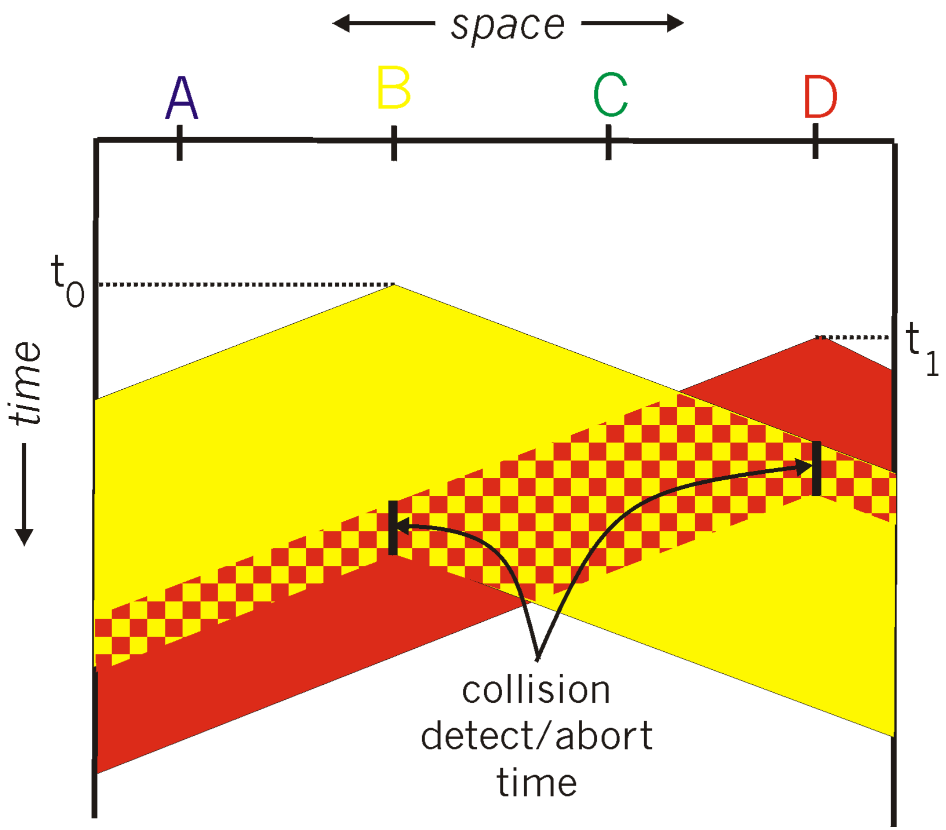
- m collisions *detected* within short time
- m colliding transmissions aborted, reducing channel wastage

r collision detection:

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

r human analogy: the polite conversationalist

CSMA/CD collision detection



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, …, 2^m-1*}. NIC waits K·512 bit times, returns to Step 2
   - longer backoff interval with more collisions

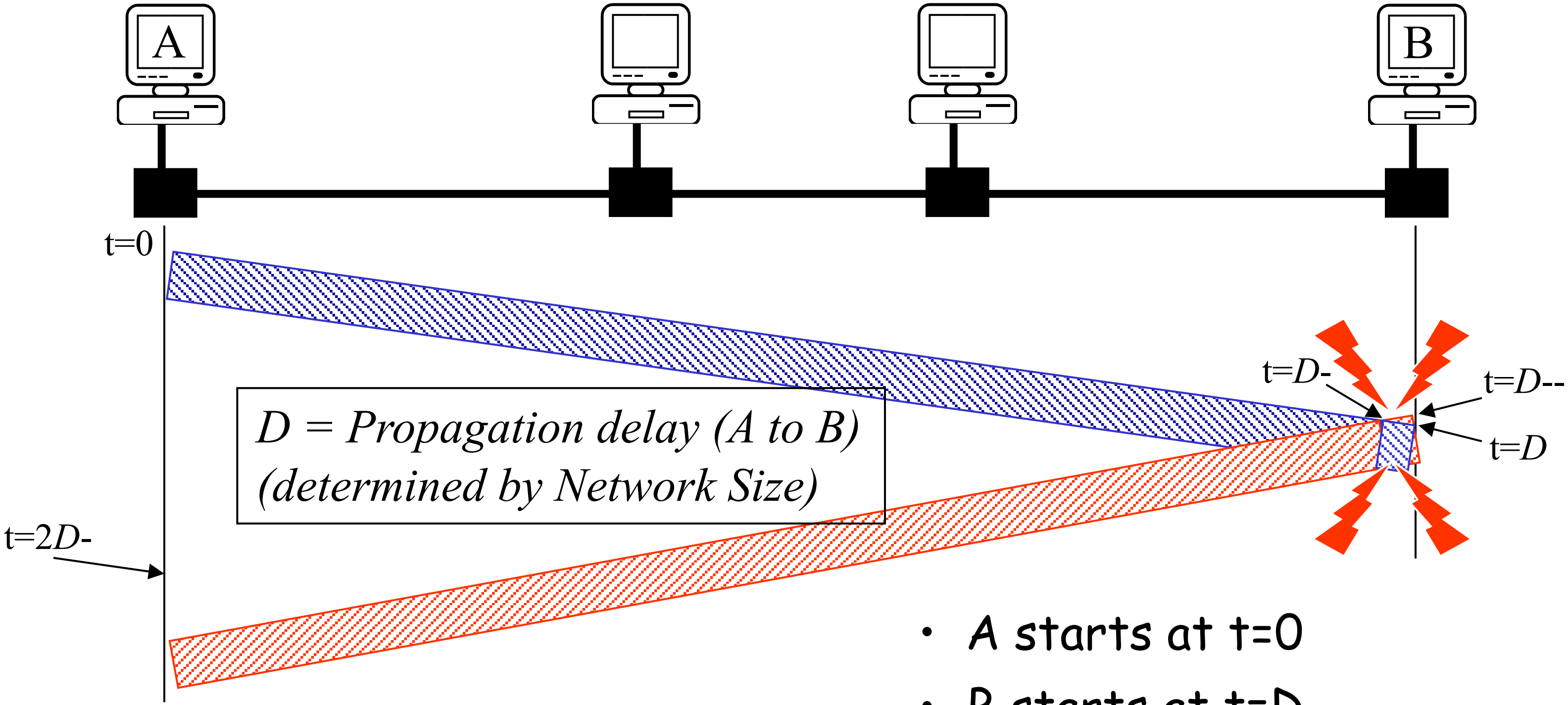
Minimum Frame Size (CSMA/CD)

- A node can only detect collision if:
  - it receives a packet when it is still transmitting

- The minimum frame should be large enough
  - Such that it is still being sent when you hear a DISTANT station
  - Otherwise, a collision may go undetected!

- LAN size and minimum frame size
  - Related to each other

Relation between network size and minimum packet size in CSMA/CD



- A starts at t=0
- B starts at t=D--
- Collision occurs at t=D-
- B detects collision at t=D
- A detects collision: t= (D) + (D--)

- Transmission time must be as large as twice the propagation delay, else you may finish transmission before detecting a potential collision
- Why? Consider the scenario above!

Maximum Frame Size (MTU)

- Every data link incurs bit error rate
  - It is the probability that a bit is in error

- Packet error rate (P_p) >> Bit error rate (P_b)
  - Packet is in error when ANY bit in the packet is in error
  - We design systems for a maximum packet error rate
  - This limits the maximum size of the packet

$$\boxed{P_p = 1 - (1-P_b)^n \qquad \text{with n bits per packet}}$$

- Applicable to data link layer in general

CSMA/CD Efficiency

- For Slotted Aloha, Probability of success
  - Directly gives the efficiency
  - Chances that “exactly” one node transmits
- For Pure Aloha, Probability of success is
  - No other frame overlaps with this frame
- CSMA/CD efficiency
  - Frame time (or useful time) divided by total time before next frame
  - Let each frame has fixed transmission time T
  - Frame time T is at least twice the prop delay D
  - Each collision wastes 2D time (at least!)

CSMA/CD Efficiency

- After a successful transmission, let W is the expected number of "time slots" wasted
  - A time slot refers to a transmission opportunity
  - There are no real "time slots" in CSMA

$$\eta = \frac{T}{T + time\_wasted\_before\_next\_sucess} = \frac{T}{T + 2DW}$$

  - After the channel is sensed busy, a node transmits with probability p at the next transmission opportunity (this applies to new as well as retransmission attempts)
  - As in slotted Aloha
    - If there are N nodes, p*=1/N
    - Max probability of success: α^* = 0.368 = 0.4 (approx)
  - $W=0\times\alpha^* + (1-\alpha^*)(1+W)$ which gives W=1.5

CSMA/CD Efficiency

$$\eta = \frac{T}{T+2DW} = \frac{T}{T+3D} = \frac{1}{1+3a}$$ where $$a \equiv \frac{D}{T}$$

Above result is by using an approximate CSMA/CD model
For example, all packets considered same size!

From more precise models and from simulation:

$$\eta = \frac{1}{1+5a}$$

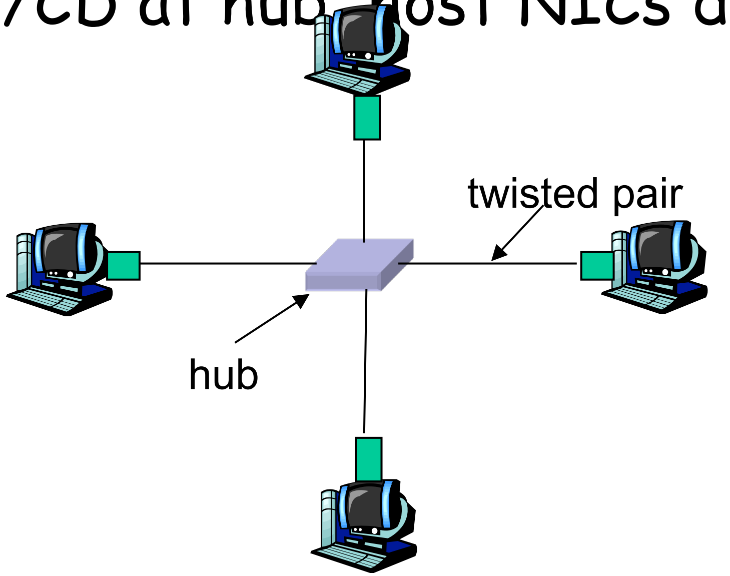
Link Layer

- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3 Multiple access protocols
- 5.4 Link-layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches, LANs, VLANs

Hubs

… physical-layer ("dumb") repeaters:

- bits coming in one link go out *all* other links at same rate
- all nodes connected to hub **can collide** with one another – single collision domain
- no frame buffering
- no CSMA/CD at hub: host NICs detect collisions

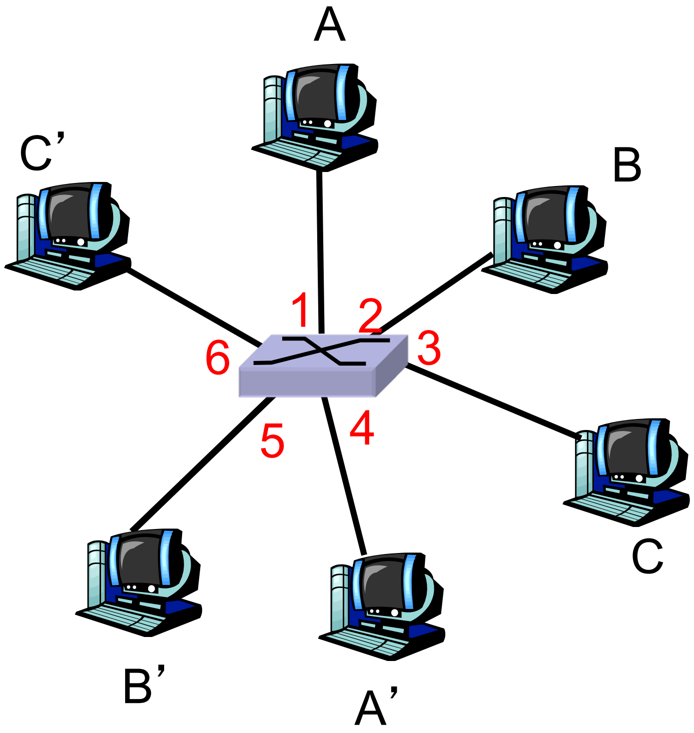


Switch (or Bridge)

- link-layer device: smarter than hubs, take *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 are unaware of presence of switches
- *plug-and-play, self-learning*
  - switches do not need to be configured

Switch: allows *multiple* simultaneous transmissions

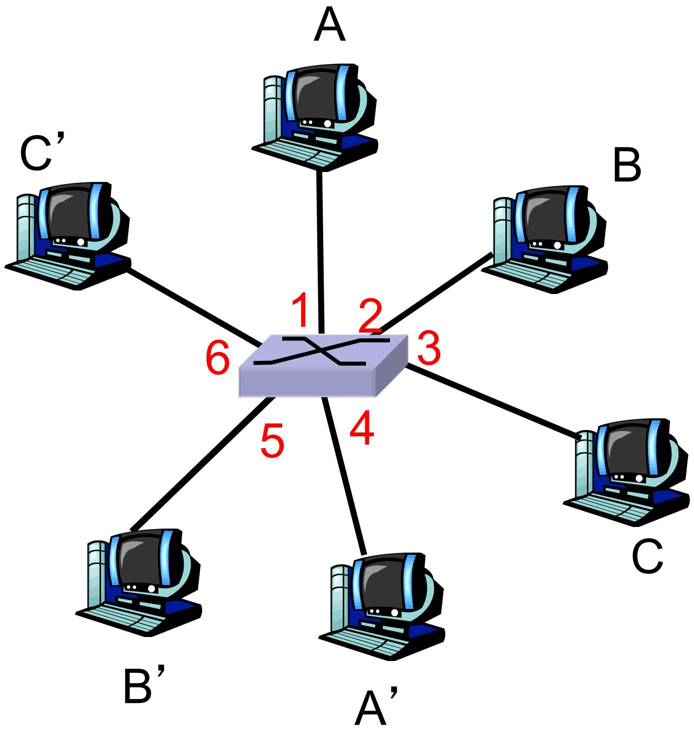
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' simultaneously, without collisions
  - not possible with dumb hub



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

Switch Table

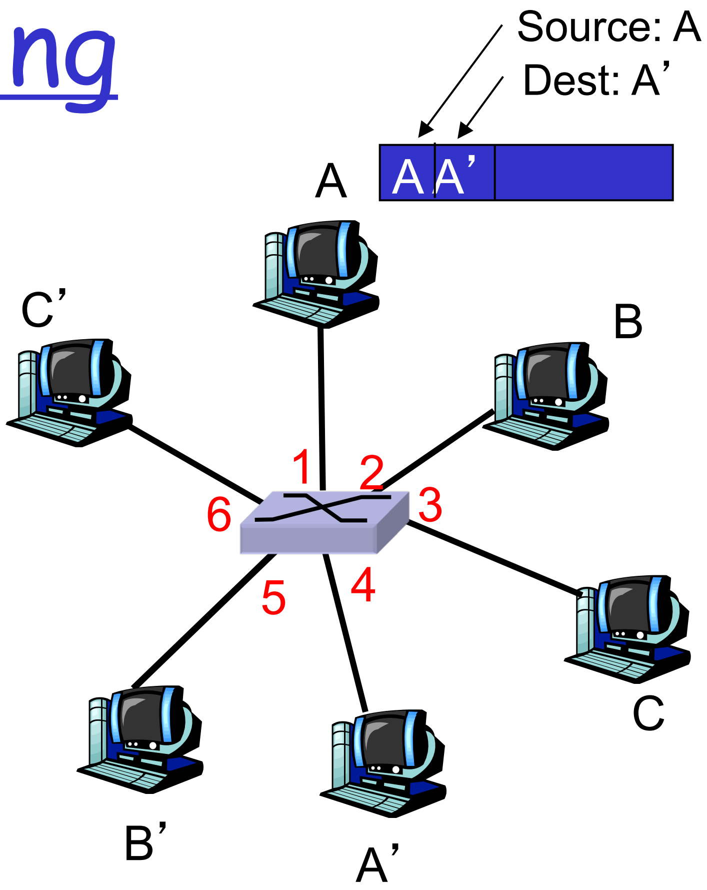
- *Q:* how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- *A:* each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!
- *Q:* how are entries created, maintained in switch table?
  - something like a routing protocol?



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

Switch: self-learning

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



MAC addr	interface	TTL
A	1	60

Switch table (initially empty)

Switch: frame filtering/forwarding

When frame received:

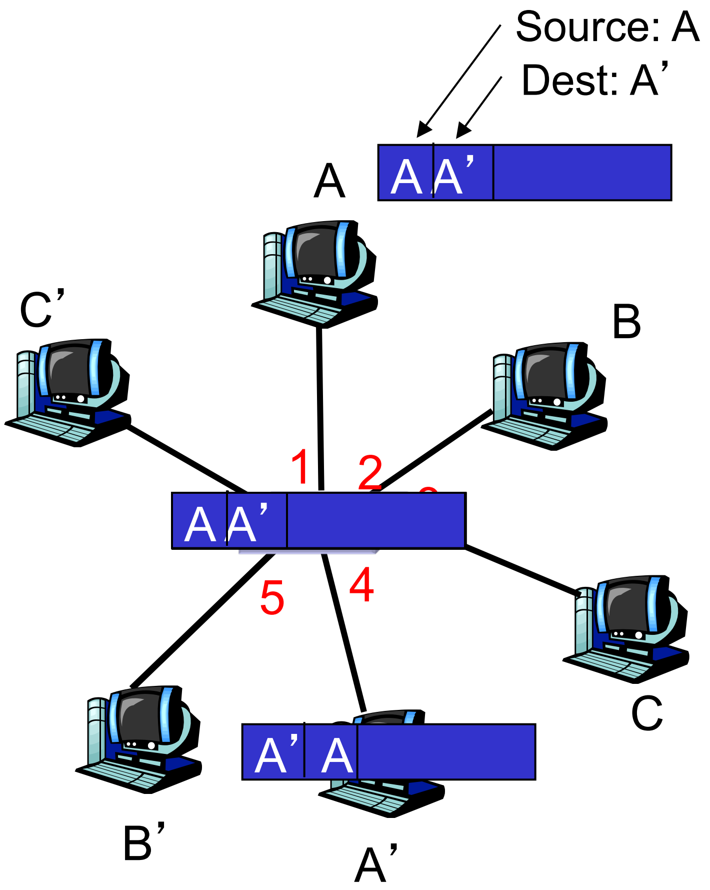
```
1. record link associated with sending host
2. index switch table using MAC dest address
3. if entry found for destination
     then {
    if dest on segment from which frame arrived
         then drop the frame
         else forward the frame on interface indicated
     }
     else flood
```

forward on all but the interface on which the frame arrived

Self-learning, forwarding: example

r frame destination unknown: flood

- destination A location known:
  selective send

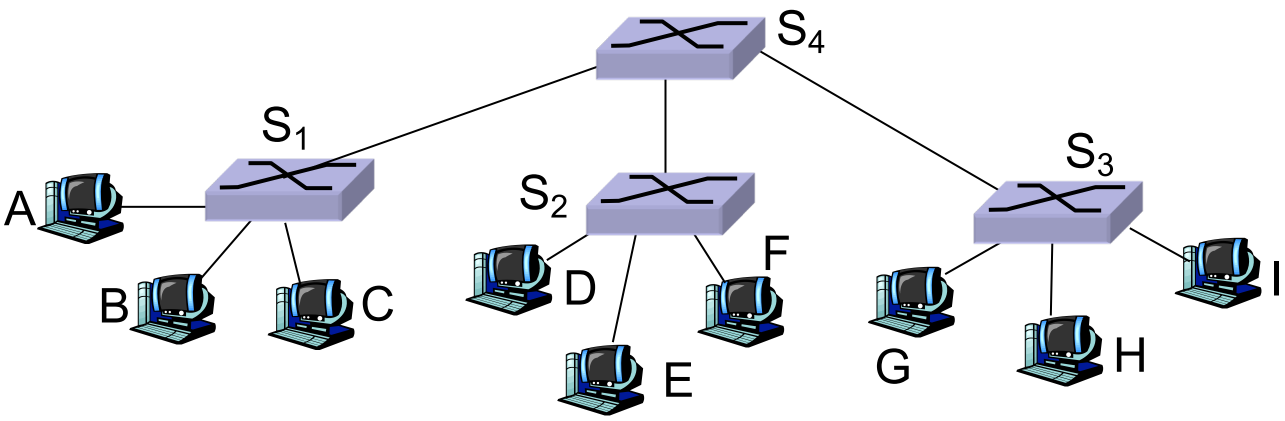


MAC addr	interface	TTL
A	1	60
A'	4	60

Switch table
(initially empty)

Interconnecting switches

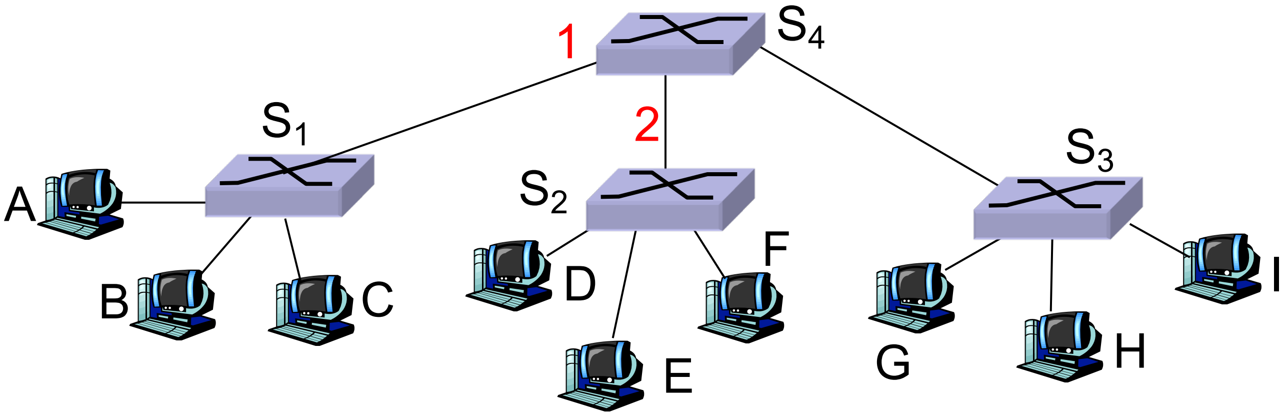
- switches can be connected together



- 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!)

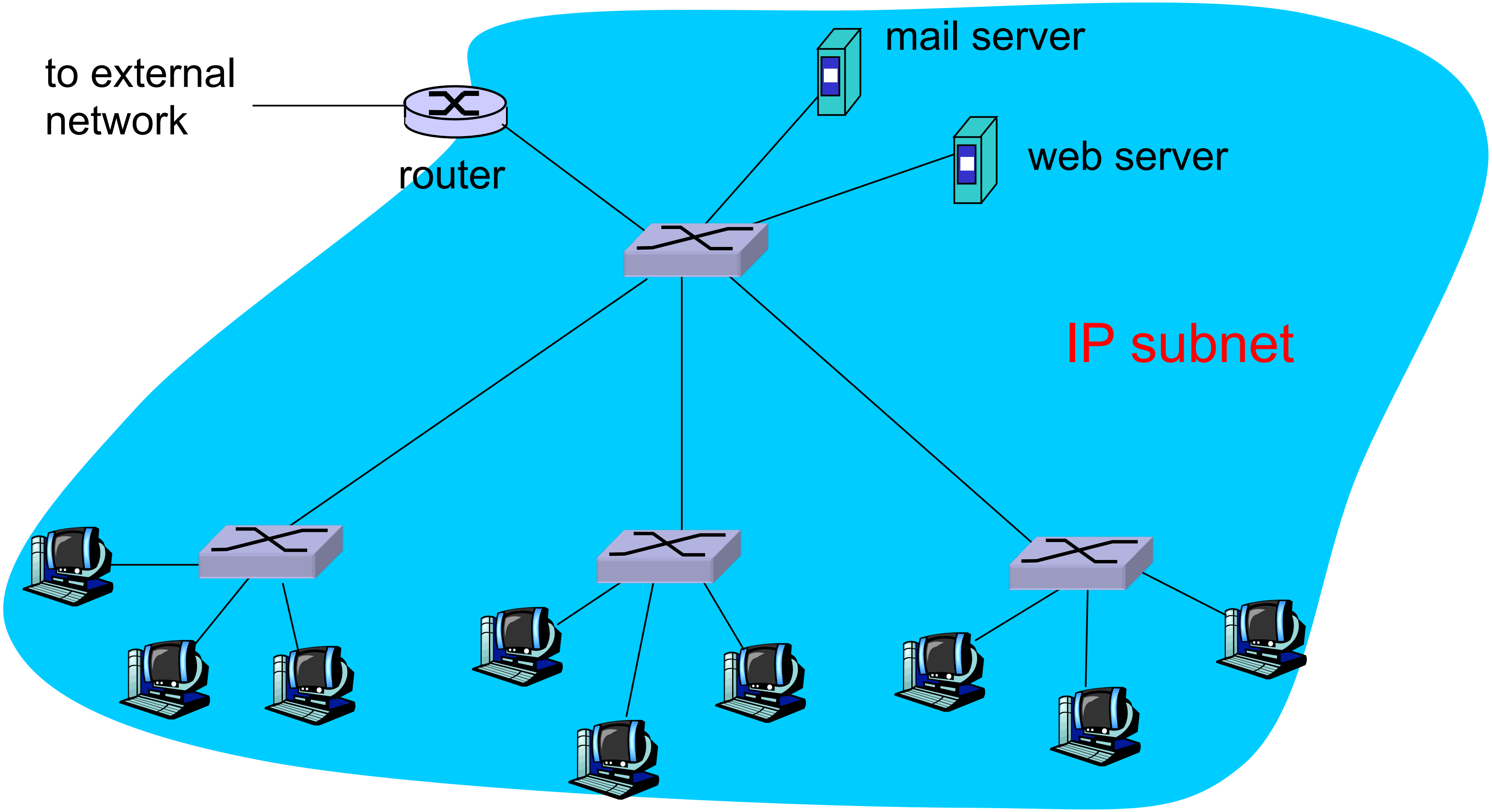
Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



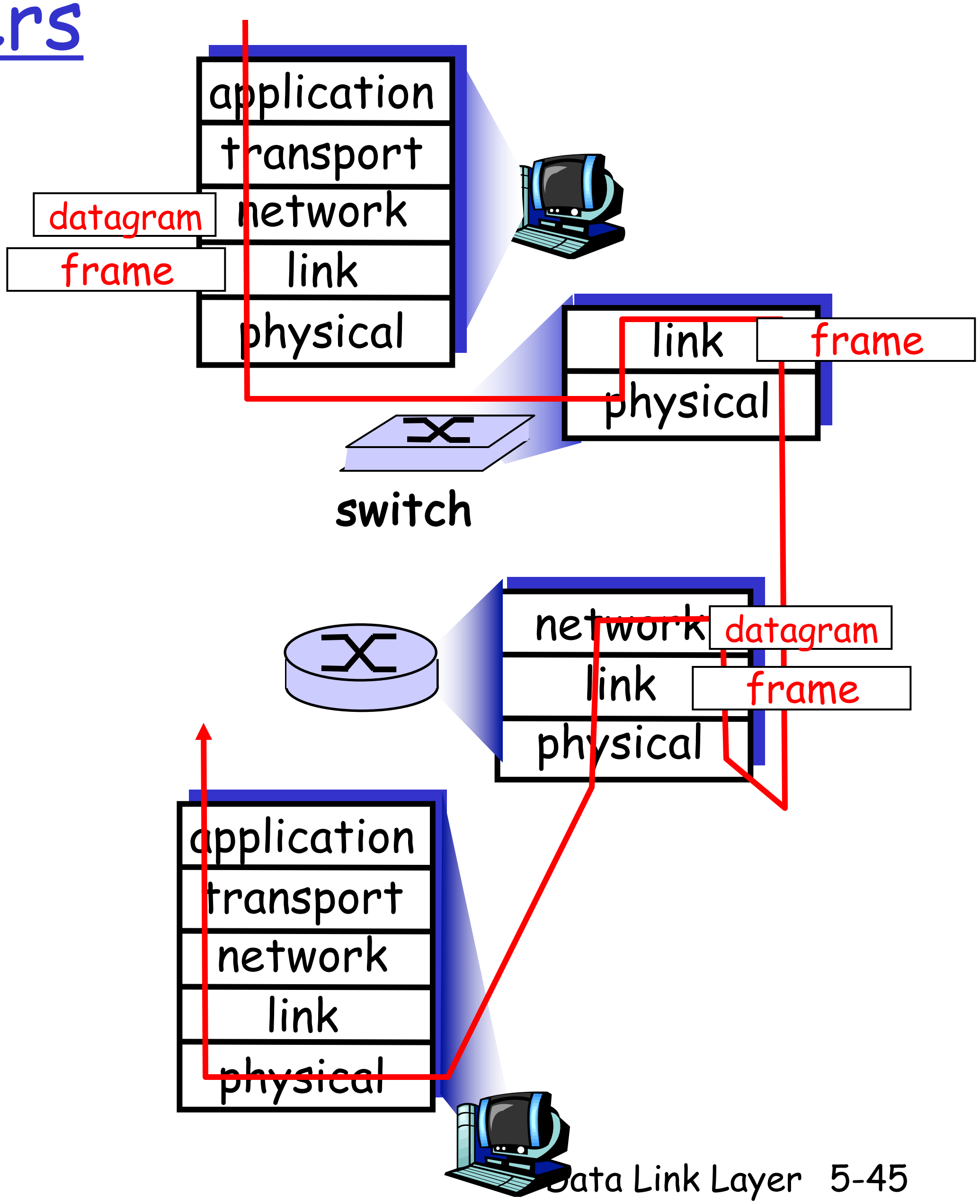
- Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4

Institutional network: small



Switches vs. Routers

- both store-and-forward devices
  - routers: network-layer devices (examine network-layer headers)
  - switches are link-layer devices (examine link-layer headers)
- routers maintain routing tables, implement routing algorithms
- switches maintain switch tables, implement filtering, learning algorithms


application
transport
network
link
physical
datagram
frame
link
physical
frame
switch
network
link
physical
datagram
frame
application
transport
network
link
physical