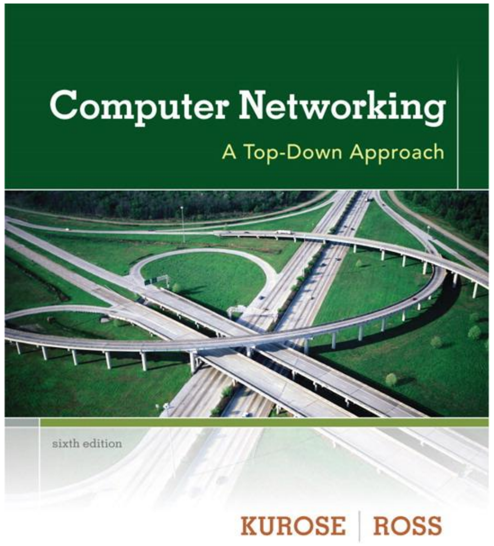
Chapter 5 Link Layer

Part 1 (of 2)



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

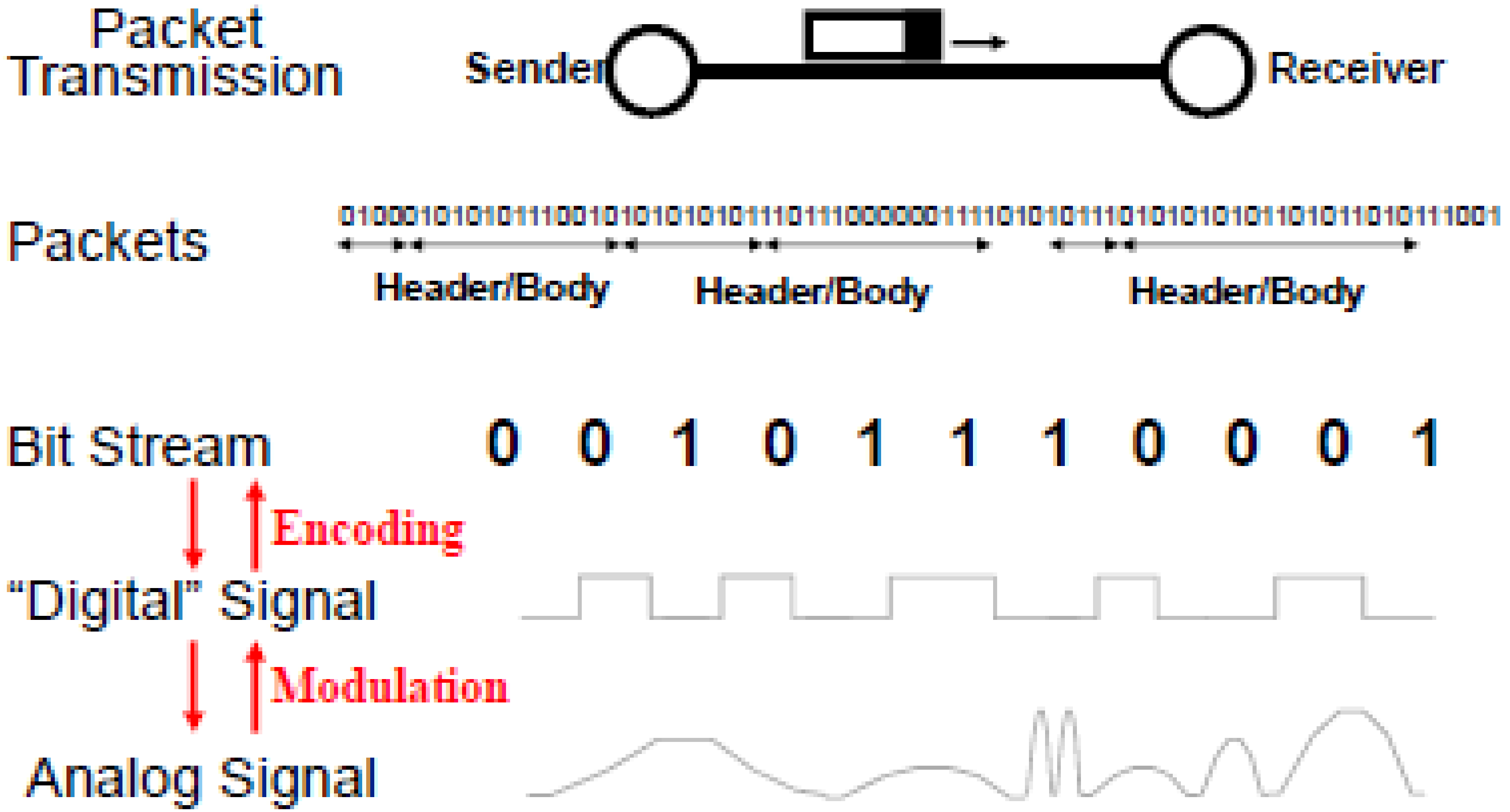
Chapter 5: Link layer

our goals:

- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet, VLANs
- instantiation, implementation of various link layer technologies

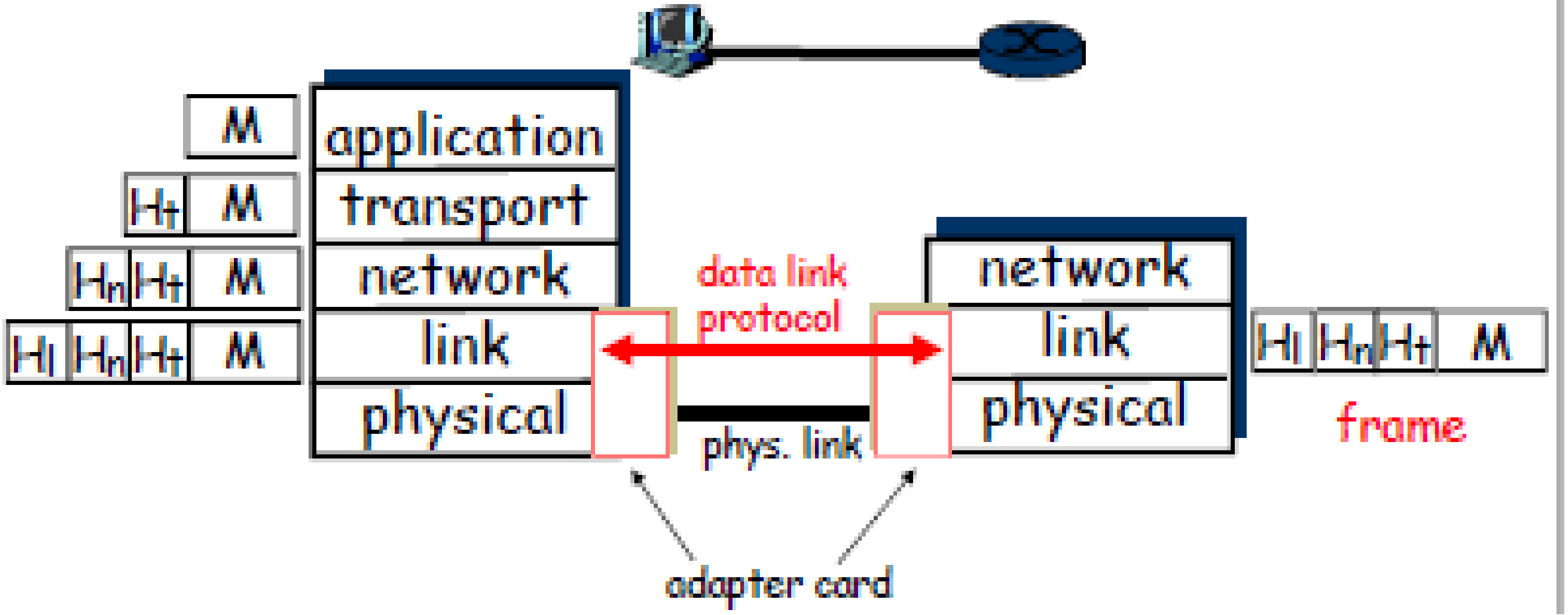
Link layer, LANs: outline

From Signals to Packets



Link Layer: Implementation

- Implemented in "adapter"
  - E.g., Ethernet card or chip
  - Typically includes: RAM, DSP chips, host bus interface, and link interface

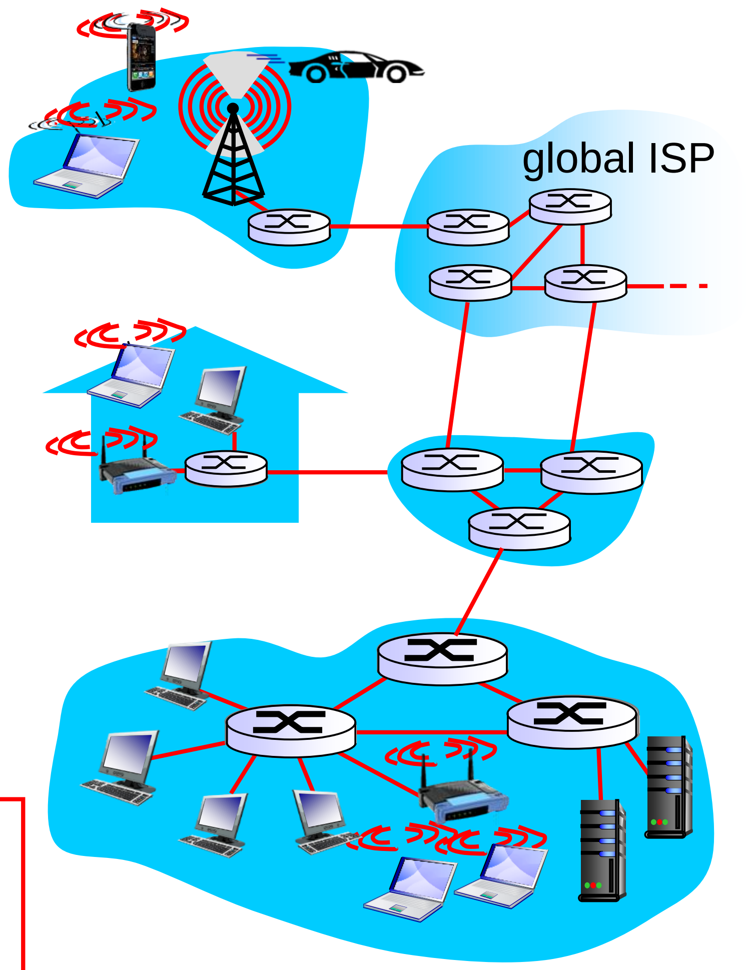


Link layer: introduction

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

    (reliable data transfer)

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

Datalink Functions

- Encoding: change bit stream before transmission
- Framing: encapsulating a network layer datagram into a bit stream.
  - Add header, mark and detect frame boundaries
- Error control: error detection and correction to deal with bit errors.
  - May also include other reliability support, e.g. retransmission
- Flow control: avoid that sender outruns the receiver
- Media access: controlling which frame should be sent next over datalink.
- Hubbing, bridging: extend the size of the network

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?

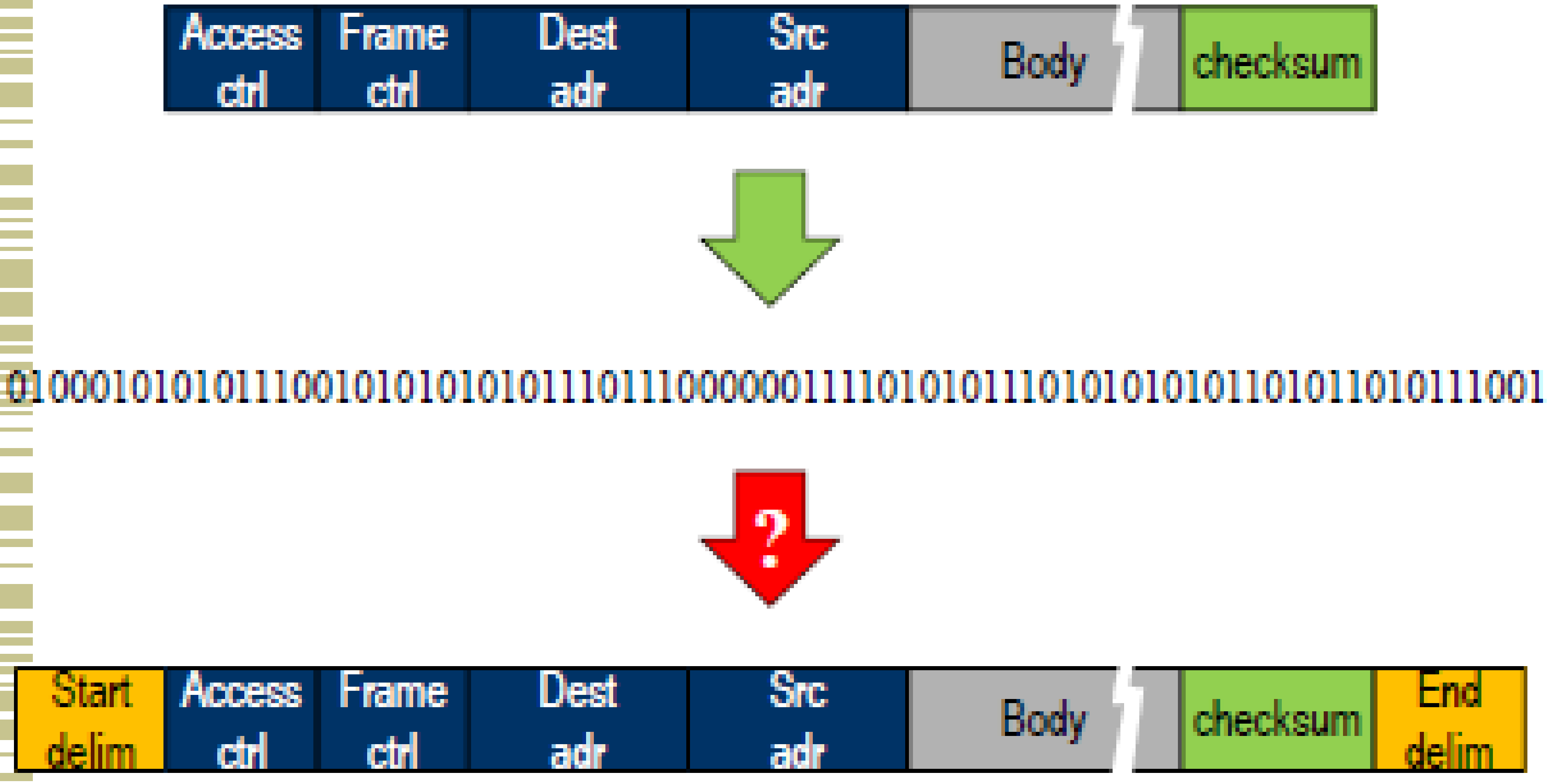
Link layer services (more)

- *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

Framing

- A link layer function, defining which bits have which function.
- Minimal functionality: mark the beginning and end of packets (or frames).

Framing

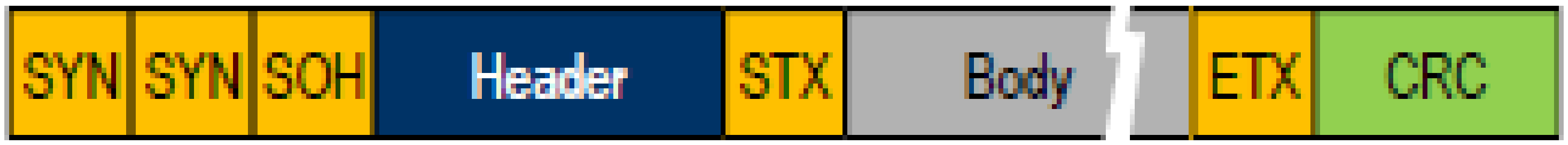

Access ctrl
Frame ctrl
Dest adr
Src adr
Body
checksum
0100010101011100101010101011101110000001111010101110101010101101011010111001
?
Start delim
Access ctrl
Frame ctrl
Dest adr
Src adr
Body
checksum
End delim

Delimiter Based

- SYN: sync character
- SOH: start of header
- STX: start of text
- ETX: end of text

- What happens when ETX is in Body?

  **Use escape character*

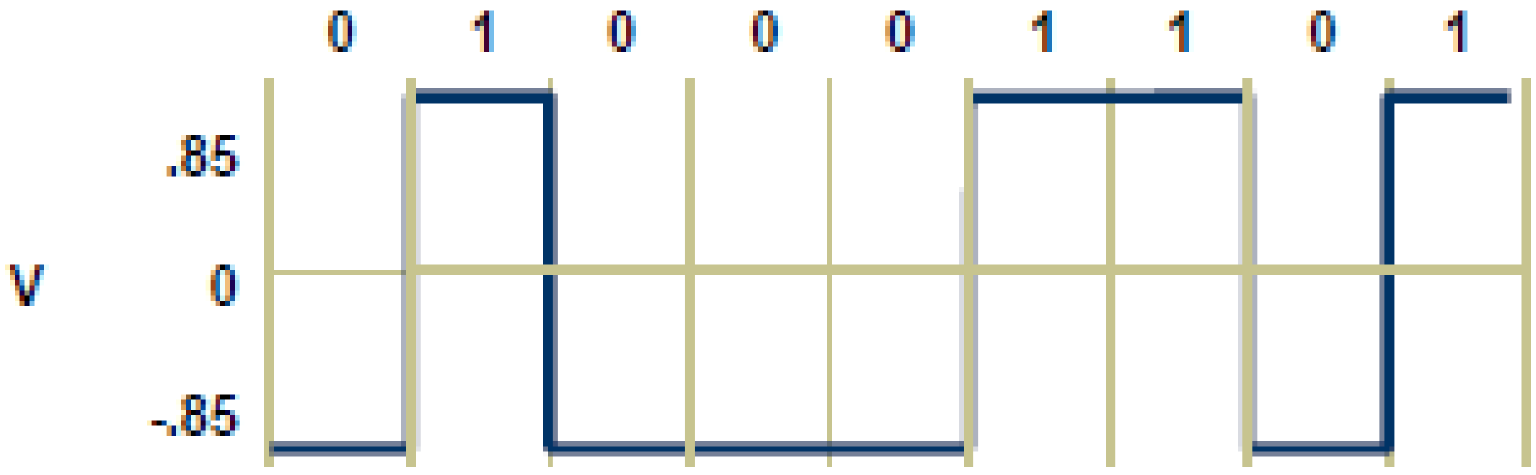


Why Do We Need Encoding?

- Keep receiver synchronized with sender.
- Create control symbols, in addition to regular data symbols.
  - E.g. start or end of frame, escape, ...
- Error detection or error corrections.
  - Some codes are illegal so receiver can detect certain classes of errors
  - Minor errors can be corrected by having multiple adjacent signals mapped to the same data symbol
- Encoding can be done one bit at a time or in multi-bit blocks, e.g., 4 or 8 bits.
- Encoding can be very complex, e.g. wireless

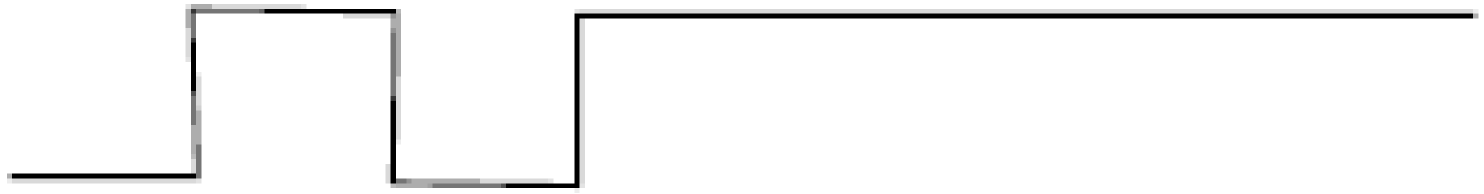
How to Encode?

- Seems obvious, why waste time on this? Just modulate the signal!



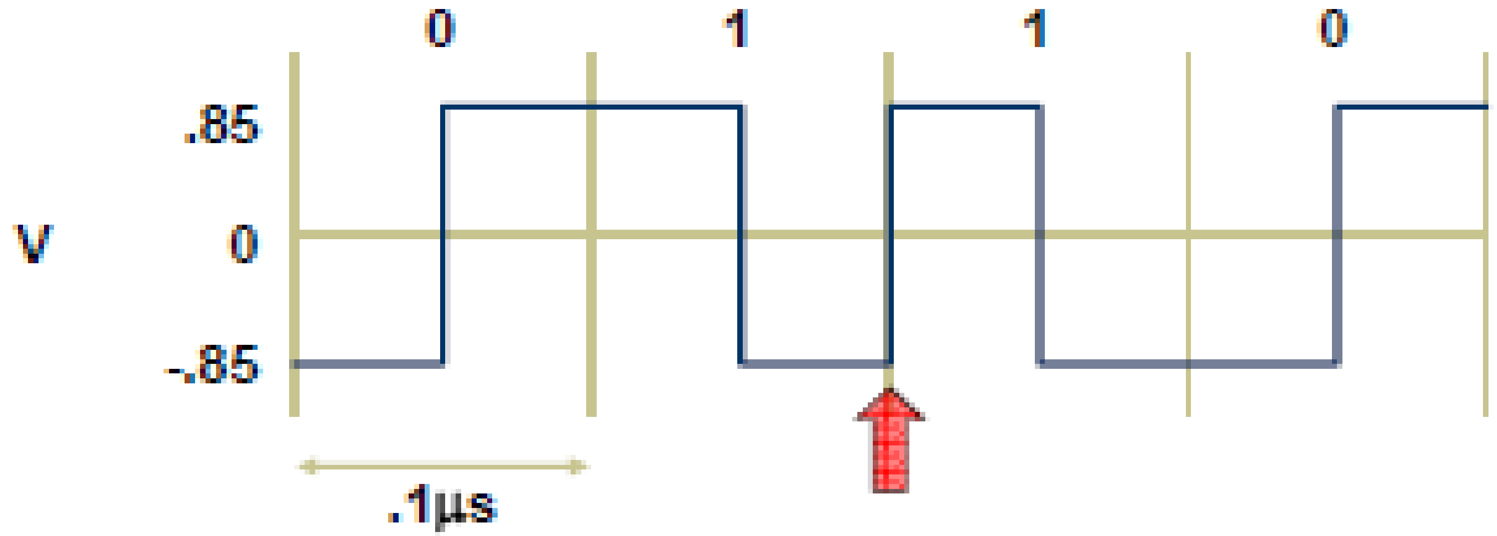
- But:
  - How easily can the receiver retrieve the bit stream?
  - What happens when there are errors: a bit gets flipped?

How about the Poor Receiver?



- Sender needs to help the receiver by "shaping" the digital bit stream so it easy to correctly interpret
  - Applies to combination of modulation and coding
- Problem in this case: not enough transitions

Manchester Encoding



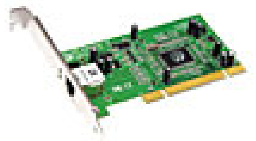
- Used by Ethernet
- 0=low to high transition, 1=high to low transition.
- Transitions simplify clock recovery and good electrical properties for any bit stream
- But you pay a price!
  - Doubles the number of transitions – more spectrum!
  - Circuitry must run twice as fast

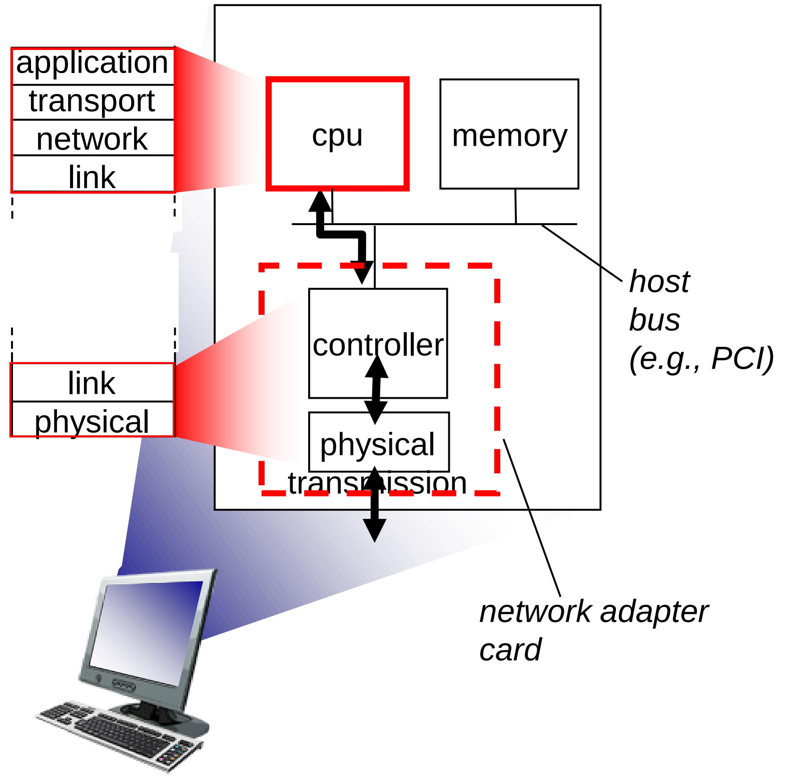
Take-away: Encoding and Modulation

- Encoding and modulation work together
  - Must generate a signal that works well for the receiver – has good electrical properties
  - Must be efficient with respect to spectrum use
  - Can shift some of the burden between the two layers
  - Tradeoff is figured out by our electrical engineers
- Maintaining good electrical properties
  - Spectrum efficient modulation requires more encoding
  - For example: 4B/5B encoding (later)
- Error recovery
  - Aggressive modulation needs stronger coding

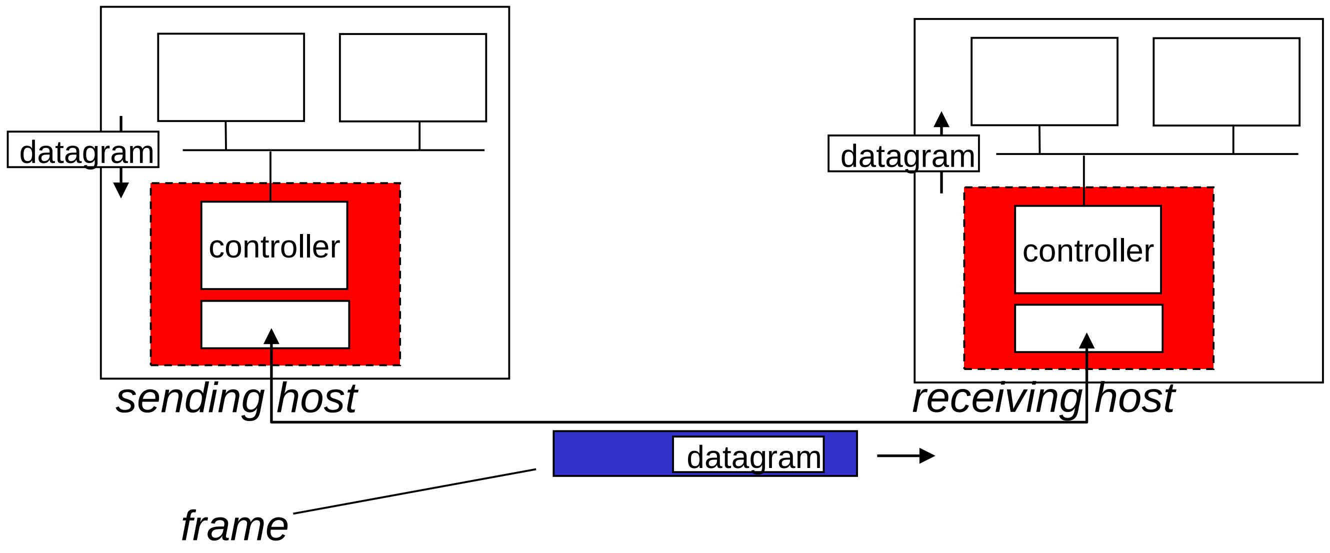
Where is the link layer implemented?

- 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



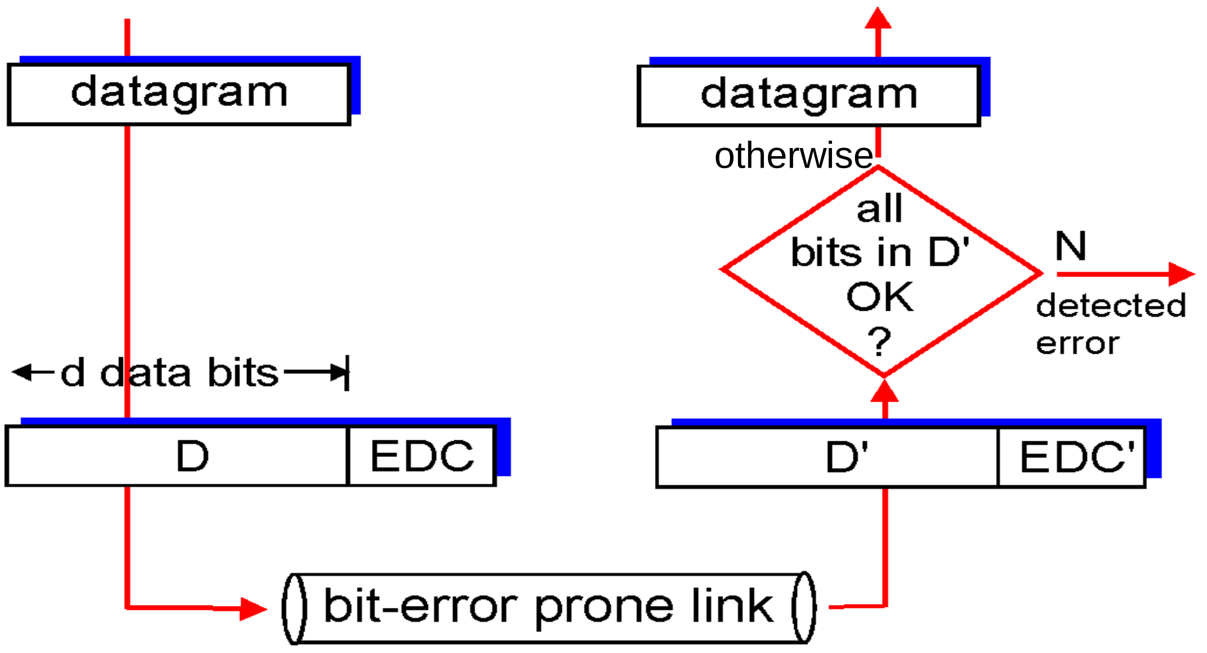
- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt *(reliable data transfer)*, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

Link layer, LANs: outline

Error detection

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

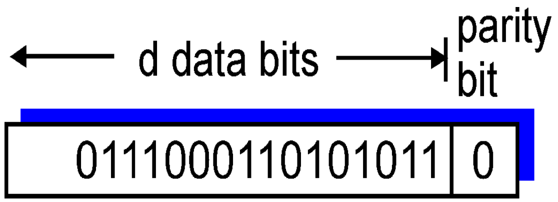
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



Parity checking

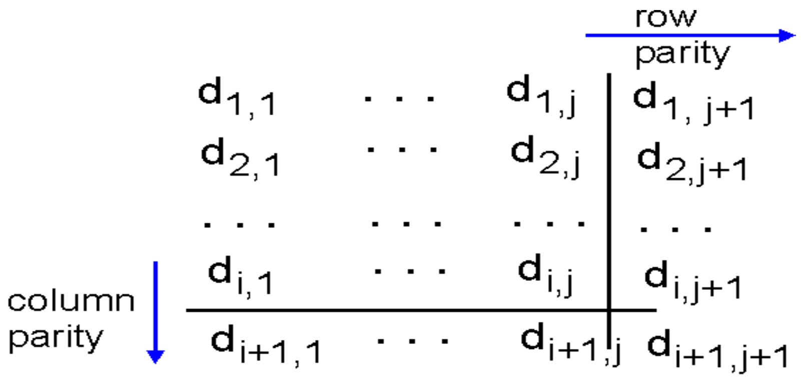
single bit parity:

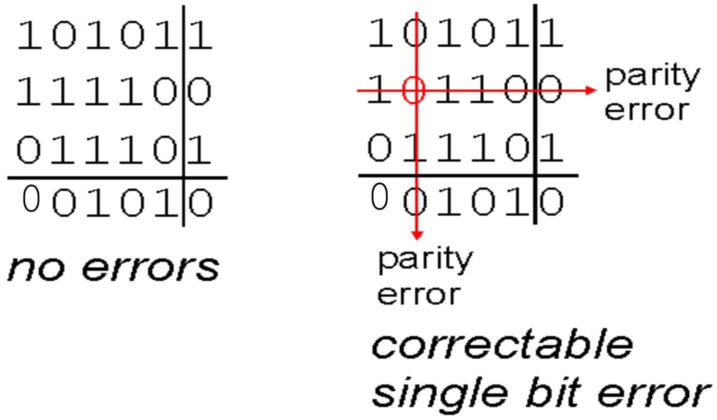
- detect single bit errors



two-dimensional bit parity:

- detect and correct single bit errors


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



Internet checksum (review)

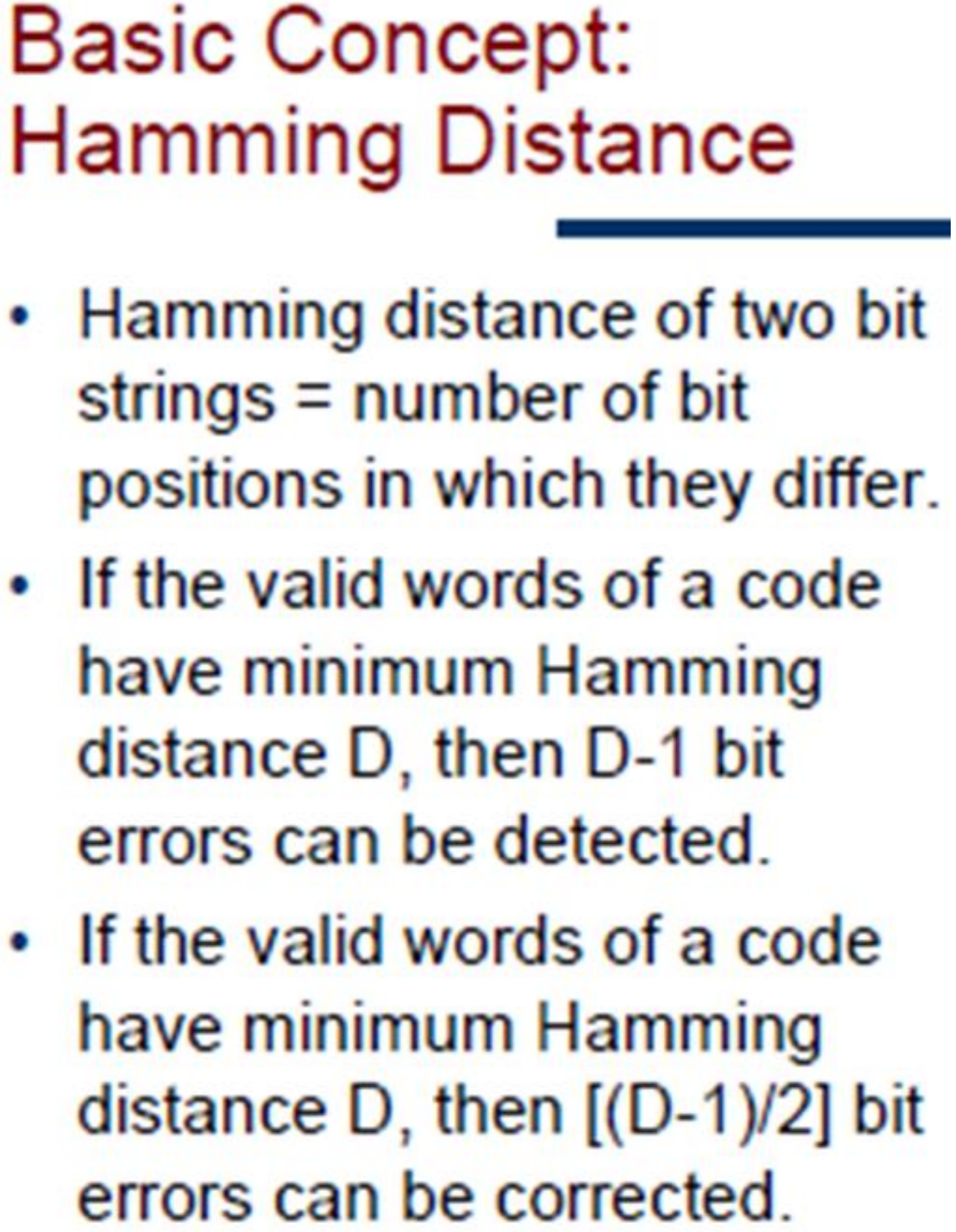
goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer *only*)

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?*



Basic Concept: Hamming Distance

- Hamming distance of two bit strings = number of bit positions in which they differ.
- If the valid words of a code have minimum Hamming distance D, then D-1 bit errors can be detected.
- If the valid words of a code have minimum Hamming distance D, then [(D-1)/2] bit errors can be corrected.

Example 1:

- A -> 000
- B -> 001
- C -> 010
- D -> 011
- E -> 100
- F -> 101
- G -> 110
- H -> 111

- The transmission is binary
- Every letter is encoded in a string of the same length
- The sender and receiver agrees on this way to code the eight letters
- If no errors occur, this is a perfectly good way to code the eight letters

Example 1 (cont.):

Sender wants to send the message ‘bad’,

so sending the string 001 000 011.

If no errors occur, then receiver gets the string

001 000 011, then he breaks it up into blocks of three: 001, 000 and 011. He knows 001 represents B, 000 represents A, and 011 represents D and so he decodes the message correctly.

Example 1(cont.):

But what happens if an error occurs?

For instance sending 001 000 011

but receiving 101 000 011,

and after decoding... 101 -> F, 000 -> A, 011 -> D.

Not only does the sender get the wrong message, but he is not even aware that an error has occurred.

Example 2:

- A -> 000 000
- B -> 001 001
- C -> 010 010
- D -> 011 011
- E -> 100 100
- F -> 101 101
- G -> 110 110
- H -> 111 111

- Doubling the size of the letter’s representation
- Can detect the error but inefficient
- Detects one error
- Can’t to correct an error

Example 2 (cont.):

Sender wants to send the message ‘bad’,

so sending the string 001 001 000 000 011 011.

One error occurs, receiver gets the string

101 001 000 000 011 011.

Receiver breaks up the message into blocks of six: 101 001, 000 000 and 011 011. Now, the receiver knows that 000 000 is A and 011 011 is D. But the string 101 001 was not assigned to any letter, so an error must occurred

Example 3:

A	0000
B	0011
C	0101
D	0110
E	1001
F	1010
G	1100
H	1111

- The binary parity check: there are an even number of ones in each string of four digits
- Can detect a single error
- Uses strings of length 4 instead of 6

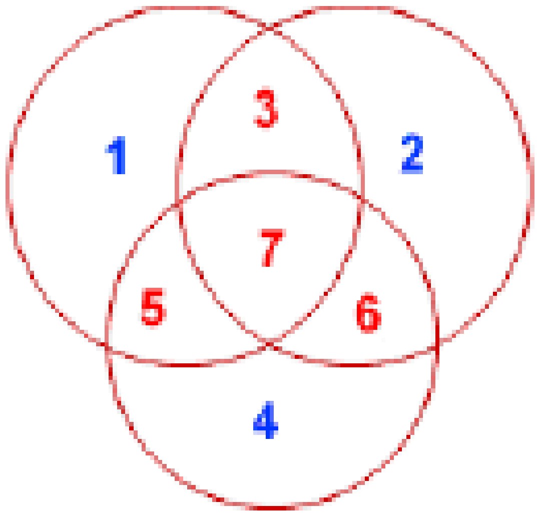
Example 4:

- A -> 000 000 000
- B -> 001 001 001
- C -> 010 010 010
- D -> 011 011 011
- E -> 100 100 100
- F -> 101 101 101
- G -> 110 110 110
- H -> 111 111 111

- The way to not only detect an error but also correct it
- Uses blocks of length 9 to represent each letter
- Can correct any single error

Example 5:

0000000	1
0000111	2
0011001	3
0011110	4
0101010	5
0101101	6
0110011	7
0110100	8
1001011	9
1001100	A
1010010	B
1010101	C
1100001	D
1100110	E
1111000	F
1111111	G



- Binary [7, 4] Hamming code

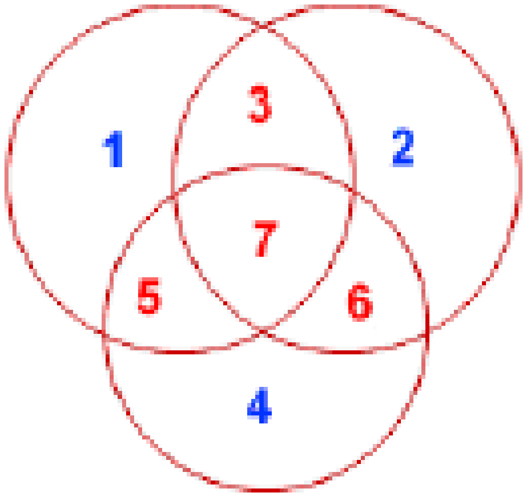
Example 3(revised):

- A -> 0000
- B -> 0011
- C -> 0101
- D -> 0110
- E -> 1001
- F -> 1010
- G -> 1100
- H -> 1111

- Minimum distance 2
- Detects a single error
- Will not correct any error

Example 5(revised):

0000000	1
0000111	2
0011001	3
0011110	4
0101010	5
0101101	6
0110011	7
0110100	8
1001011	9
1001100	A
1010010	B
1010101	C
1100001	D
1100110	E
1111000	F
1111111	G



- Binary [7, 4] Hamming code
- Minimum distance 3
- Detects 2 errors
- Corrects a single error

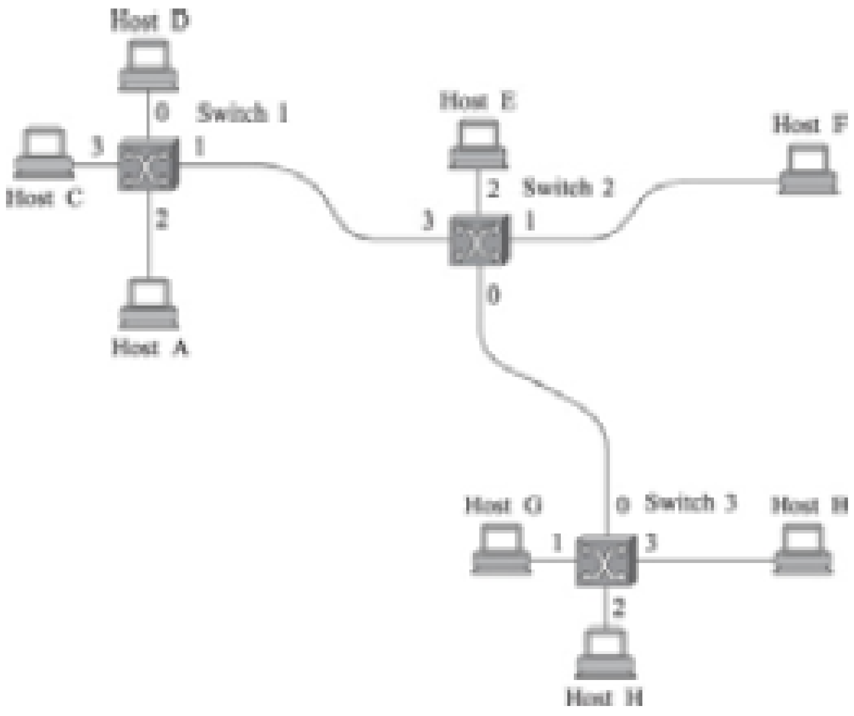
Cyclic Redundancy Codes (CRC)

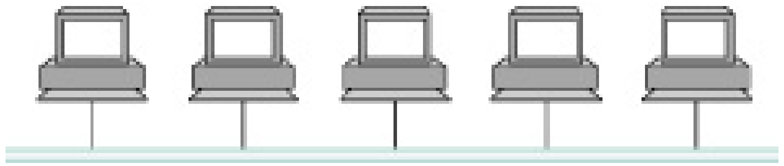
- Commonly used codes that have good error detection properties.
  - Can catch many error combinations with a small number of redundant bits
- Based on division of polynomials.
  - Errors can be viewed as adding terms to the polynomial
  - Should be unlikely that the division will still work
- Can be implemented very efficiently in hardware.
- Examples:
  - CRC-32: Ethernet
  - CRC-8, CRC-10, CRC-32: ATM

Link layer, LANs: outline

Datalink Architectures

- Switches connected by point-to-point links -- store-and-forward.
  - Used in WAN, LAN, and for home connections
  - Conceptually similar to "routing"
    - But at the datalink layer instead of the network layer
  - MAC = (local) scheduling
- Multiple access networks -- contention based.
  - Multiple hosts are sharing the same transmission medium
  - Used in LANs and wireless
  - Access control is distributed and much more complex

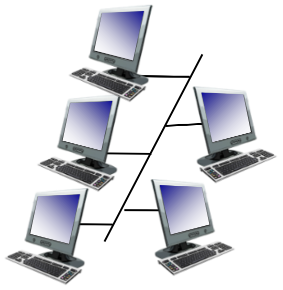


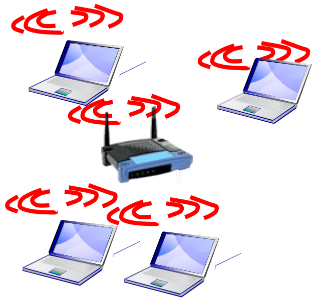


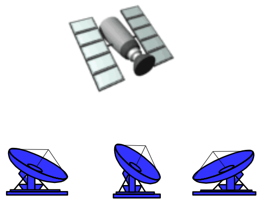
Multiple access links, protocols

two types of "links":

- point-to-point
  - PPP (Point to Point Protocol) for dial-up access
  - point-to-point link between Ethernet switch, host
- *broadcast (shared wire or medium)*
  - old-fashioned Ethernet
  - upstream HFC (Hybrid fiber-coaxial – used by cable tv etc…)
  - 802.11 wireless LAN

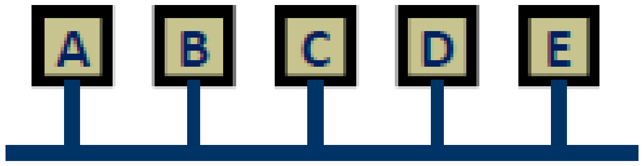

shared wire (e.g., cabled Ethernet)


shared RF (e.g., 802.11 WiFi)


shared RF (satellite)

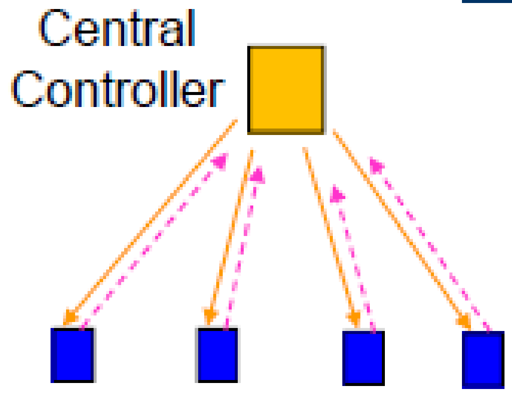
humans at a cocktail party (shared air, acoustical)

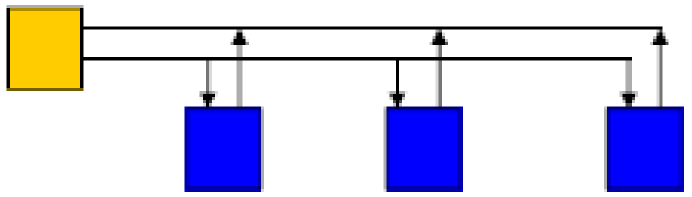
Multiple Access: How to Share a Wire (or the wireless ether)

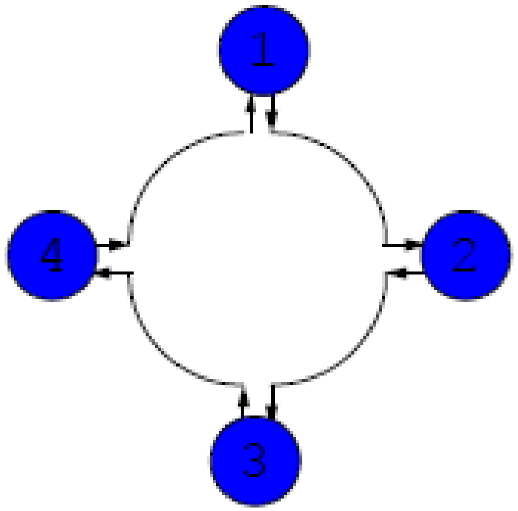


- Problem: how do you prevent nodes from “talking” at the same time – causes “collision”
- Two classes of solutions:
  - Explicit coordination: schedule transmissions sequentially
  - Randomly access medium: send and hope you get lucky

Scheduled Access MACs







- Reservation systems
  - Central controller
  - Distributed algorithm, e.g. using reservation bits in frame
- Polling: controller polls each nodes
- Token ring: token travels around ring and allows nodes to send one packet
  - Distributer version of polling
  - FDDI, ...

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

desired:

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

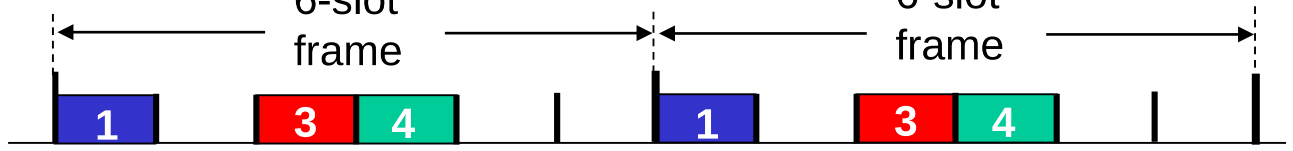
three broad classes:

- *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

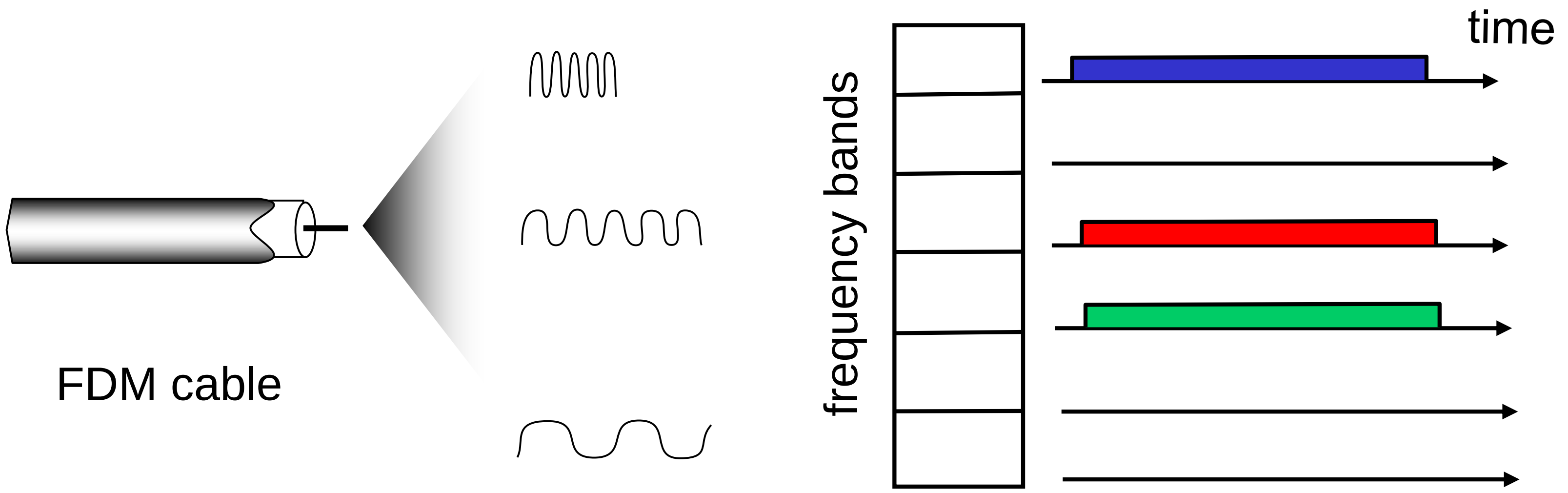
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



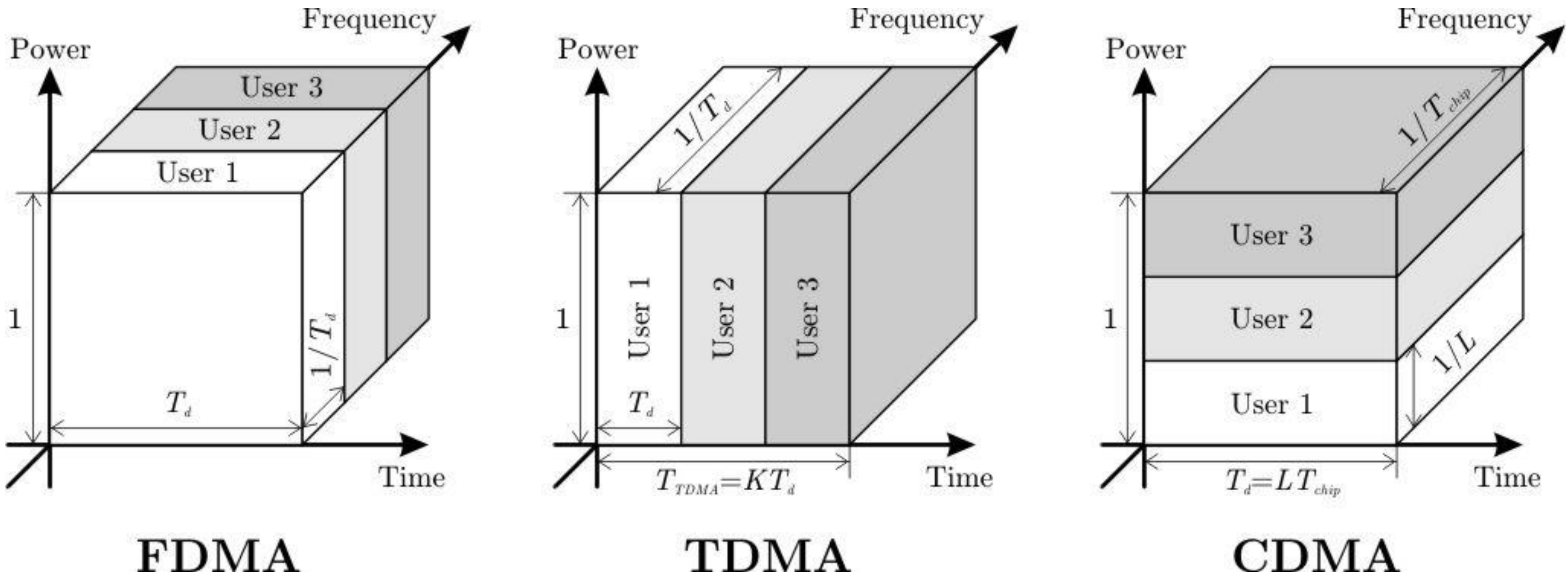
Channel partitioning MAC protocols: FDMA

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



Code division (CDMA)



Users utilize different codings (languages), same freq. at the same time, no confusions occur

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

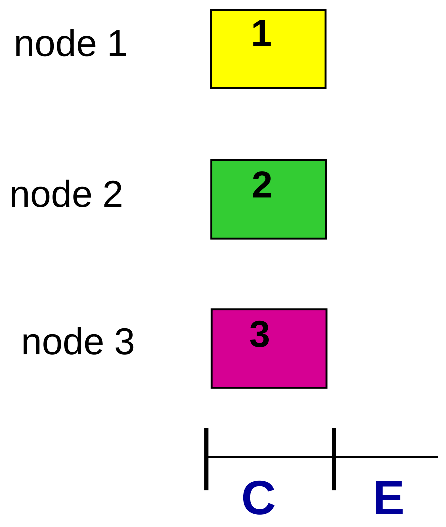
assumptions:

- all frames same size
- time 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

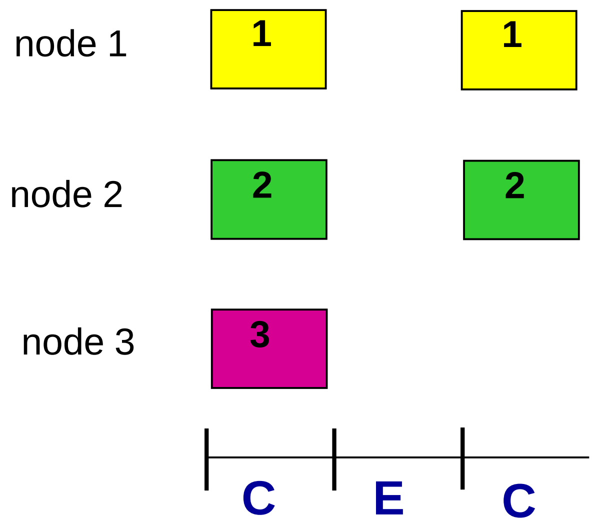
operation:

- 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

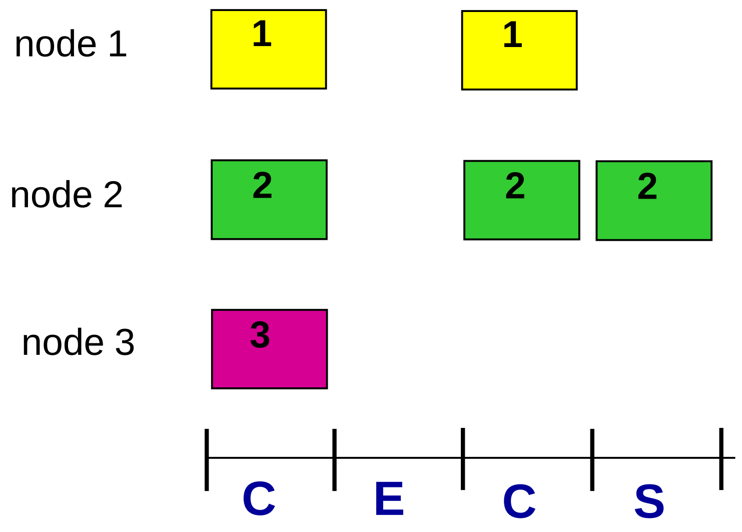
Slotted ALOHA



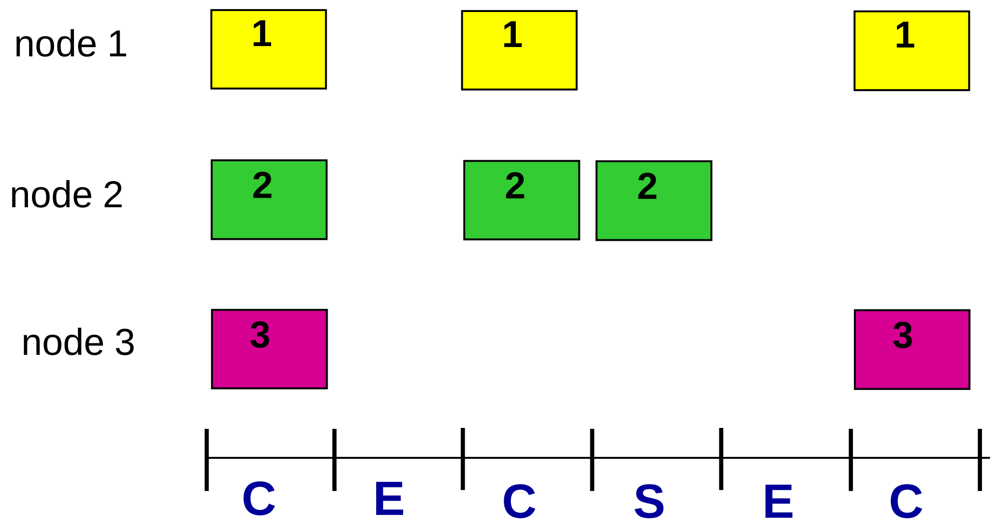
Slotted ALOHA



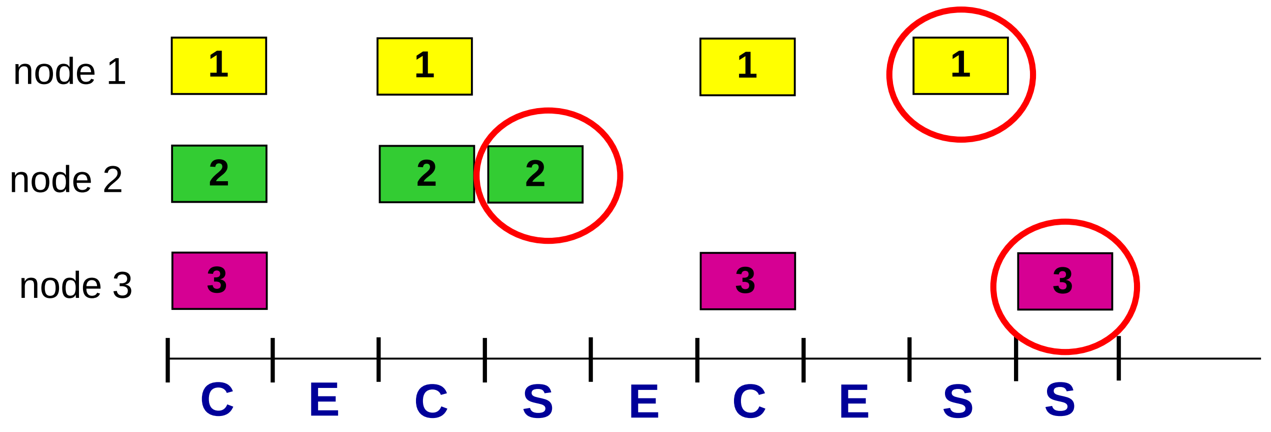
Slotted ALOHA



Slotted ALOHA



Slotted ALOHA



(Example) Efficiency = 3 / 9 = 0.33

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

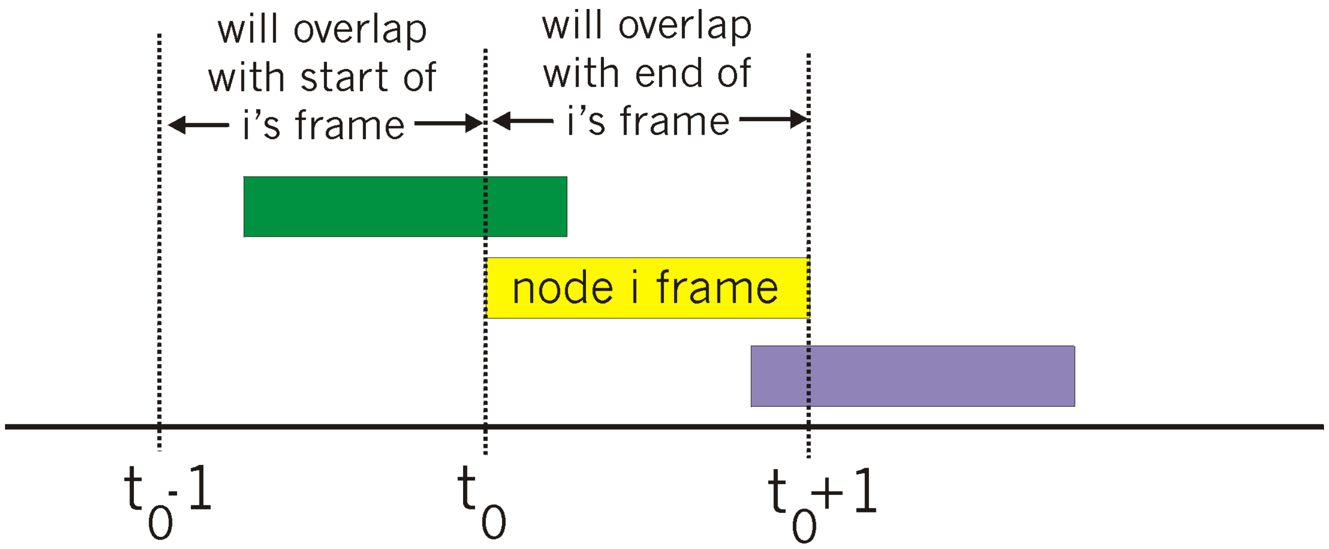
efficiency: long-run fraction of successful slots (many nodes, all with many frames to send)

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

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

even *worse* than slotted Aloha!

CSMA (carrier sense multiple access)

CSMA: listen before transmit:

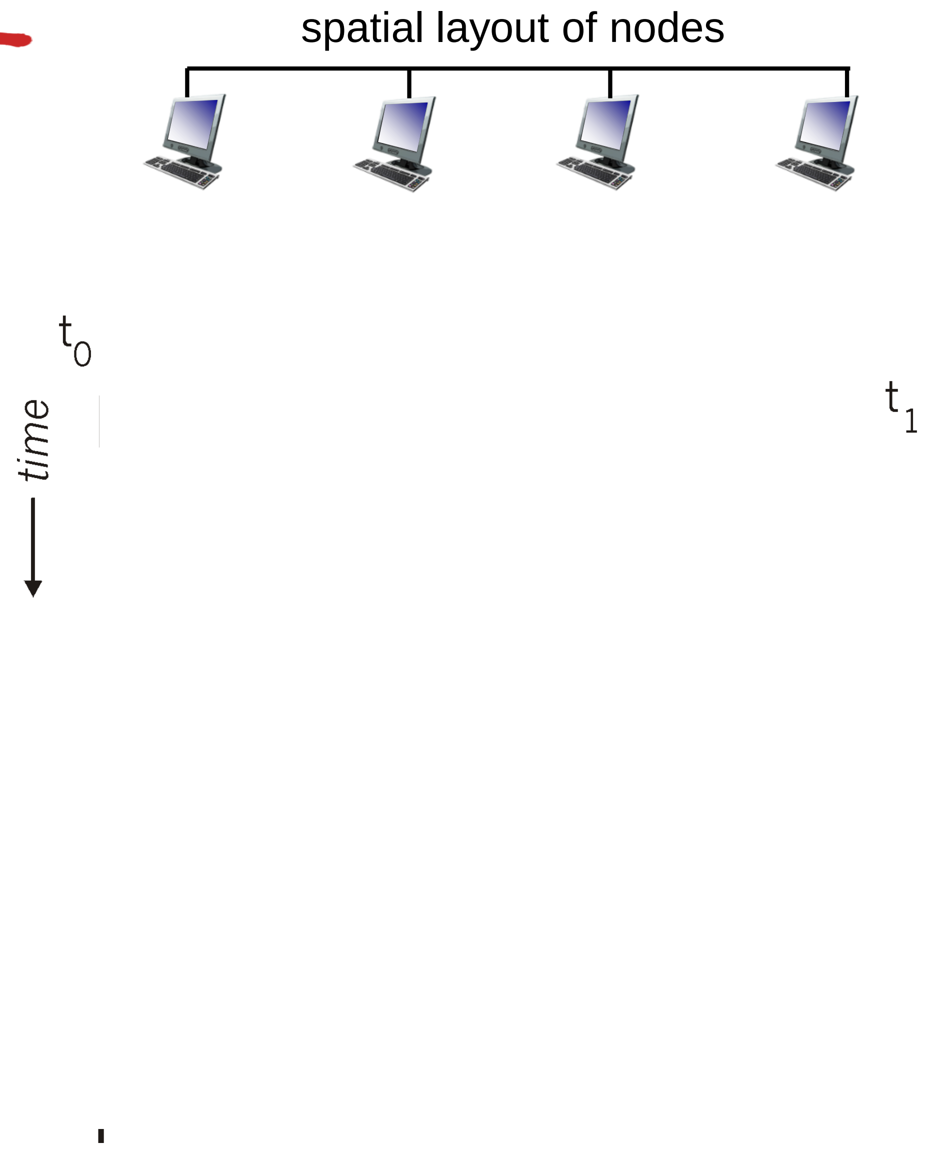
if channel sensed idle: transmit entire frame

- 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
  - distance & propagation delay play role in in determining collision probability



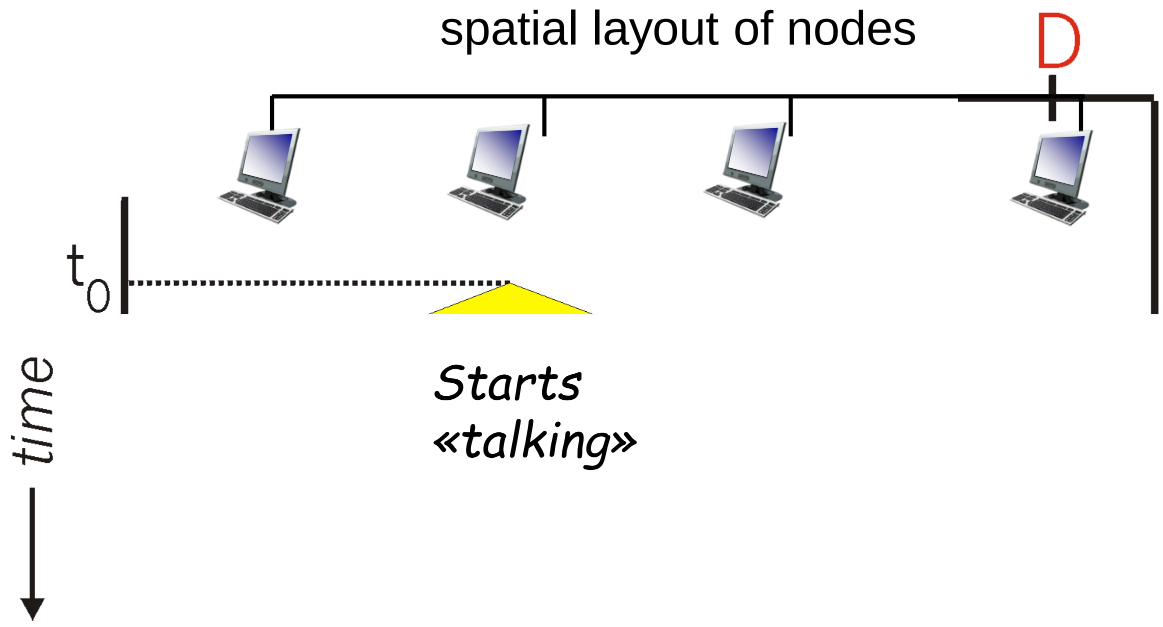
CSMA/CD (collision detection)

CSMA/CD: carrier sensing, deferral as in CSMA

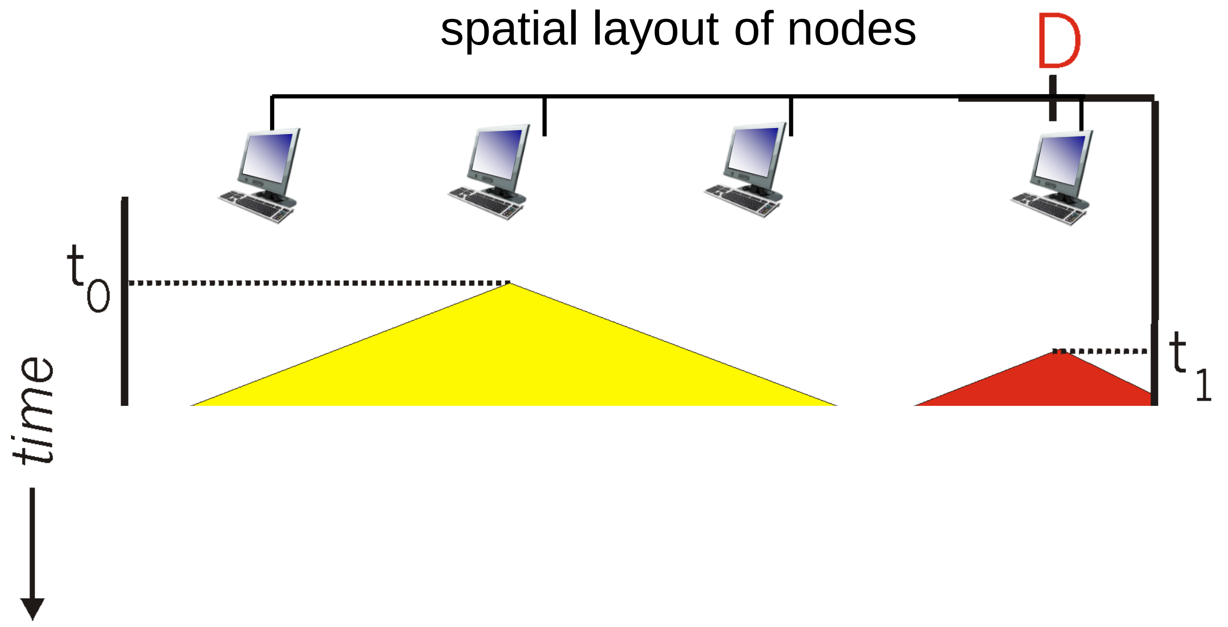
- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- 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

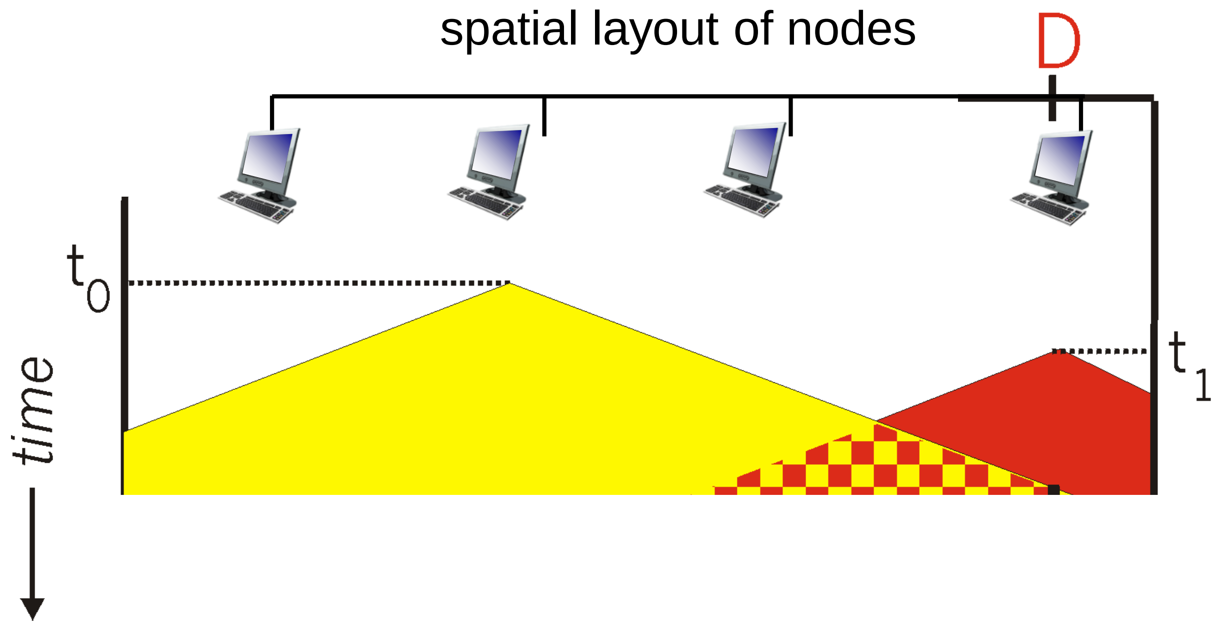
CSMA/CD (collision detection)



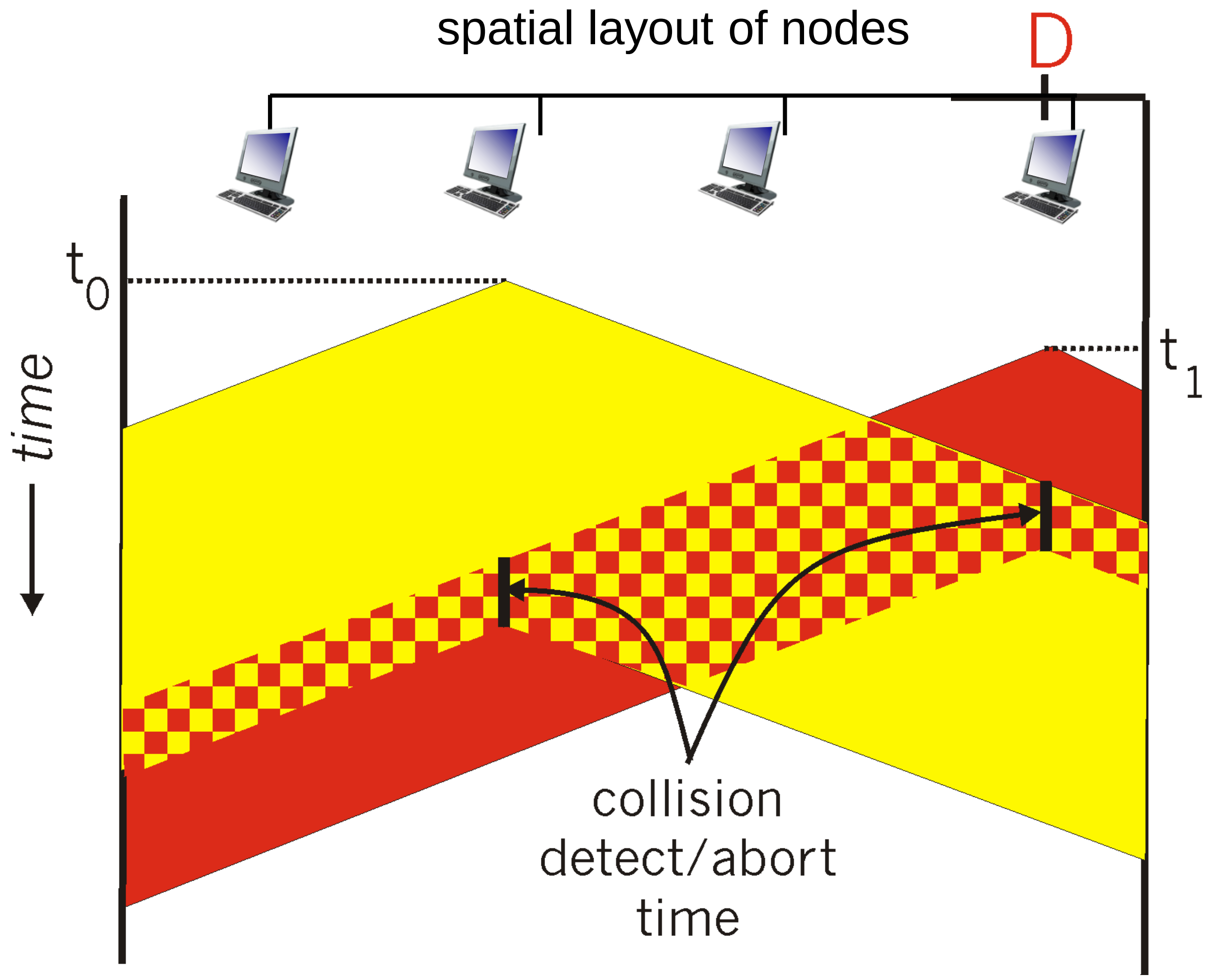
CSMA/CD (collision detection)



CSMA/CD (collision detection)



CSMA/CD (collision detection)



Ethernet MAC Features

- Carrier Sense: listen before you talk
  - Avoid collision with an ongoing transmission
- Advantage is that it is very efficient
  - No coordination overhead or transmission delay
- But it does not always work: simultaneous transmissions can happen
  - Speed of light is "only" 1 foot/nsec ☺
- Collision Detection during transmission
  - Listen while transmitting
  - If you notice interference → assume collision
  - Abort transmission immediately and schedule a retransmission

Ethernet CSMA/CD algorithm

network interface card

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

CSMA/CD efficiency

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

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

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

"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

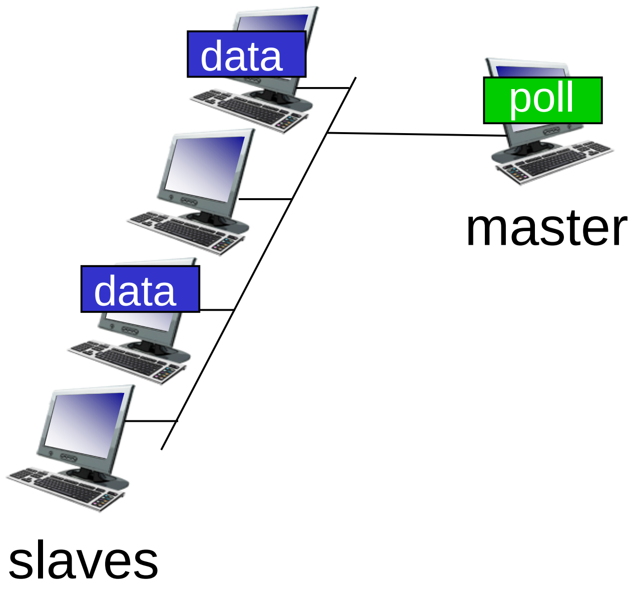
"taking turns" protocols

look for best of both worlds!

"Taking turns" MAC protocols

polling:

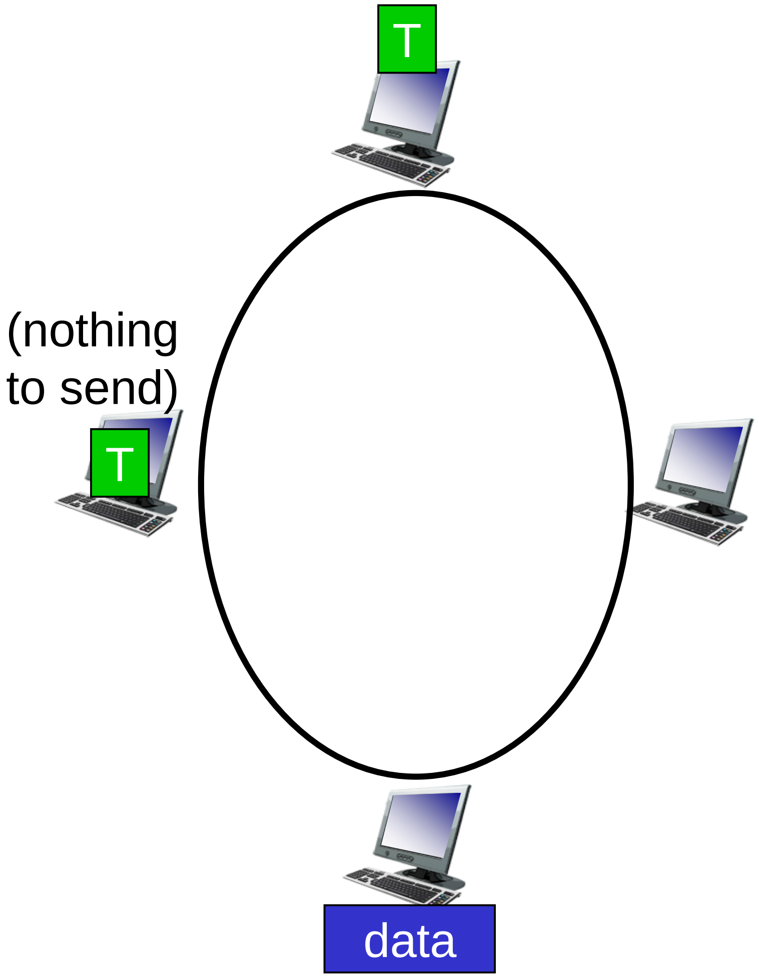
- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



"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)



Summary of MAC protocols

- *channel partitioning*, by time, frequency or code (CDMA – 3G…)
  - 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

    Fiber Distributed Data Interface