

Chapter 5

Data Link Layer

Reti di Elaboratori

Corso di Laurea in Informatica

Università degli Studi di Roma "La Sapienza"

Canale A-L

Prof.ssa Chiara Petrioli

Parte di queste slide sono state prese dal materiale associato al libro
Computer Networking: A Top Down Approach, 5th edition.
All material copyright 1996-2009

J.F Kurose and K.W. Ross, All Rights Reserved

Thanks also to Antonio Capone, Politecnico di Milano, Giuseppe Bianchi and
Francesco LoPresti, Un. di Roma Tor Vergata

Link Layer

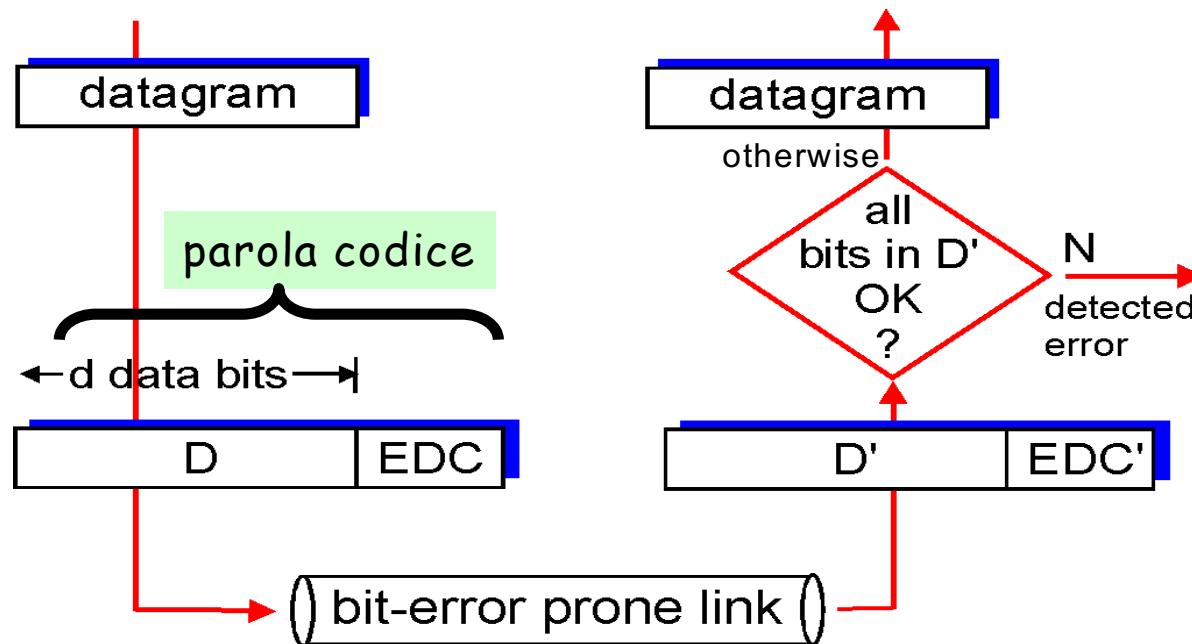
- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3 Multiple access protocols
- 5.4 Link-layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches
- 5.7 PPP
- 5.8 Link virtualization: MPLS
- 5.9 A day in the life of a web request

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



Distanza di Hamming

- Date due parole codice e.g., 10001001 e 10110001 è possibile determinare in quanti bit 'differiscono' (XOR delle due parole e contate il numero di 1 del risultato)
 - Il numero di posizioni nelle quali le due parole di codice differiscono determina la loro distanza di Hamming
 - Se due parole codice hanno una distanza di Hamming d ci vorranno d errori sui singoli bit per tramutare una parola di codice nell'altra
 - Per come sono usati i bit di ridondanza se la lunghezza delle parole di codice è $n=m+r$ sono possibili 2^m messaggi dati ma non tutte le 2^n parole codice
 - la distanza di Hamming di un codice è la minima distanza di Hamming tra due parole codice

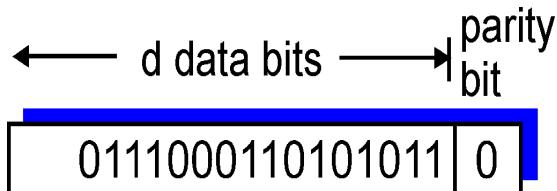
Distanza di Hamming

- Date due parole codice e.g., 10001001 e 10110001 è possibile determinare in quanti bit 'differiscano' (XOR delle due parole e contate il numero di 1 del risultato)
 - Per come sono usati i bit di ridondanza se la lunghezza delle parole di codice è $n=m+r$ sono possibili 2^m messaggi dati ma non tutti 2^n parole codice
 - la distanza di Hamming di un codice è la minima distanza di Hamming tra due parole codice
 - Per fare il detection di d errori serve un codice con distanza di Hamming $d+1$
 - Per correggere d errori serve un codice con distanza di Hamming $2d+1$

Parity Checking

Single Bit Parity:

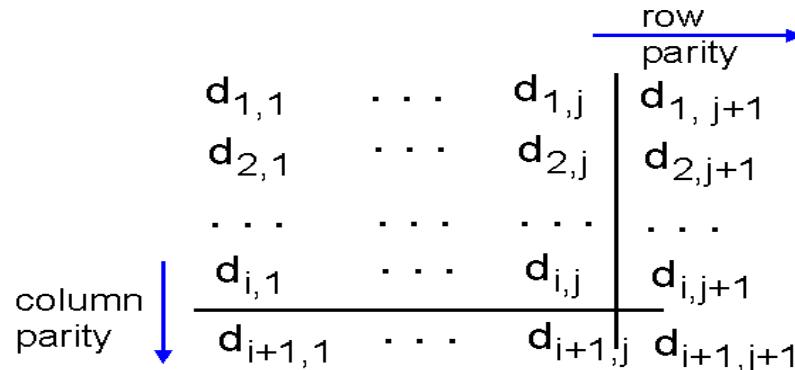
Detect single bit errors



Schema di parità dispari:
 Il mittente include un bit addizionale e sceglie il suo valore in modo che il numero di uno nei **d+1** bit sia dispari

Two Dimensional Bit Parity:

Detect and correct single bit errors



1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
<hr/>					
0	0	1	0	1	0

no errors

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
<hr/>					
0	0	1	0	1	0

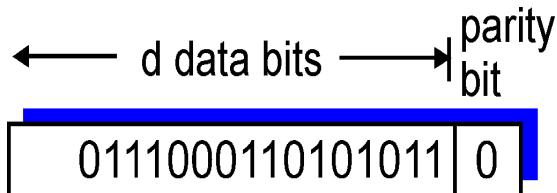
parity error

*correctable
single bit error*

Parity Checking

Single Bit Parity:

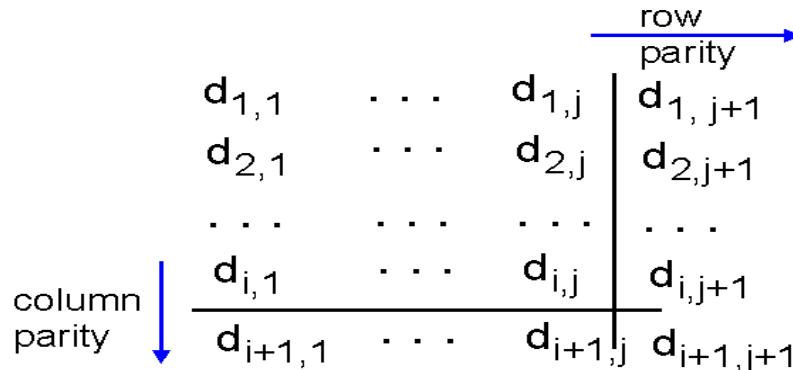
Detect single bit errors



Schema di parità dispari:
 Il mittente include un bit addizionale e sceglie il suo valore in modo che il numero di uno nei $d+1$ bit sia dispari

Two Dimensional Bit Parity:

Detect and correct single bit errors



1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
<hr/>					
0	0	1	0	1	0

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
<hr/>					
0	0	1	0	1	0

Schemi semplici possono essere sufficienti nel caso di errori casuali
 Cosa si può fare nel caso di errori a burst?

- Maggiore ridondanza
- Interleaving

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

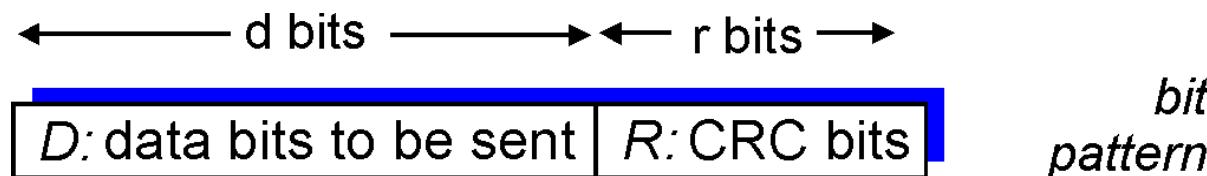
Receiver:

- 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

- o view data bits, D , as a binary number
- o choose $r+1$ bit pattern (generator), G
- o goal: choose r CRC bits, R , such that
 - $\langle D, R \rangle$ exactly divisible by G (modulo 2)
 - receiver knows G , divides received $\langle D', R' \rangle$ by G . If non-zero remainder: error detected!
 - can detect all burst errors less than $r+1$ bits
- o widely used in practice (Ethernet, 802.11 WiFi, ATM)



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

mathematical formula

CRC

- r è l'ordine del polinomio generatore $G(x)$
- Appendi r bit zero al messaggio $M(x)$ che ora corrisponde a $x^r M(x)$
- dividi $x^r M(x)$ per $G(x)$ modulo 2
- Sottrai (modulo 2) il resto della divisione da $x^r M(x) \rightarrow$ si ottiene $T(x)$, il risultato da trasmettere

- In ricezione controlla che il resto della divisione per $G(x)$ sia 0
- Estrai la parte di messaggio $M(x)$
→ Individua un burst di fino a r errori

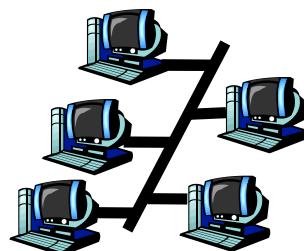
Link Layer

- 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
- 5.7 PPP
- 5.8 Link virtualization: MPLS
- 5.9 A day in the life of a web request

Multiple Access Links and Protocols

Two types of “links”:

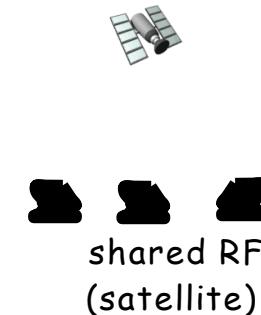
- point-to-point
 - PPP for dial-up access
 - point-to-point link between Ethernet switch and host
- broadcast (shared wire or medium)
 - old-fashioned Ethernet
 - upstream HFC
 - 802.11 wireless LAN



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

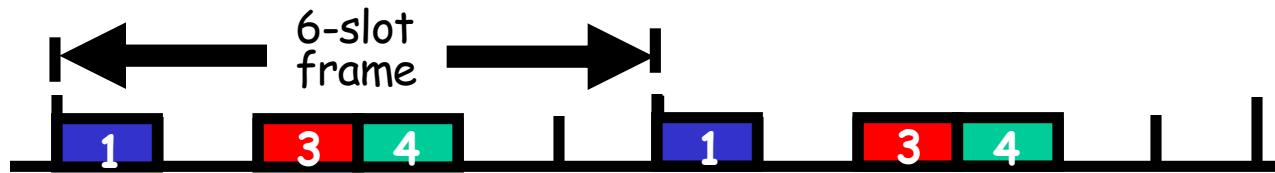
Three broad classes:

- **Channel Partitioning**
 - divide channel into smaller “pieces” (time slots, frequency, code)
 - 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

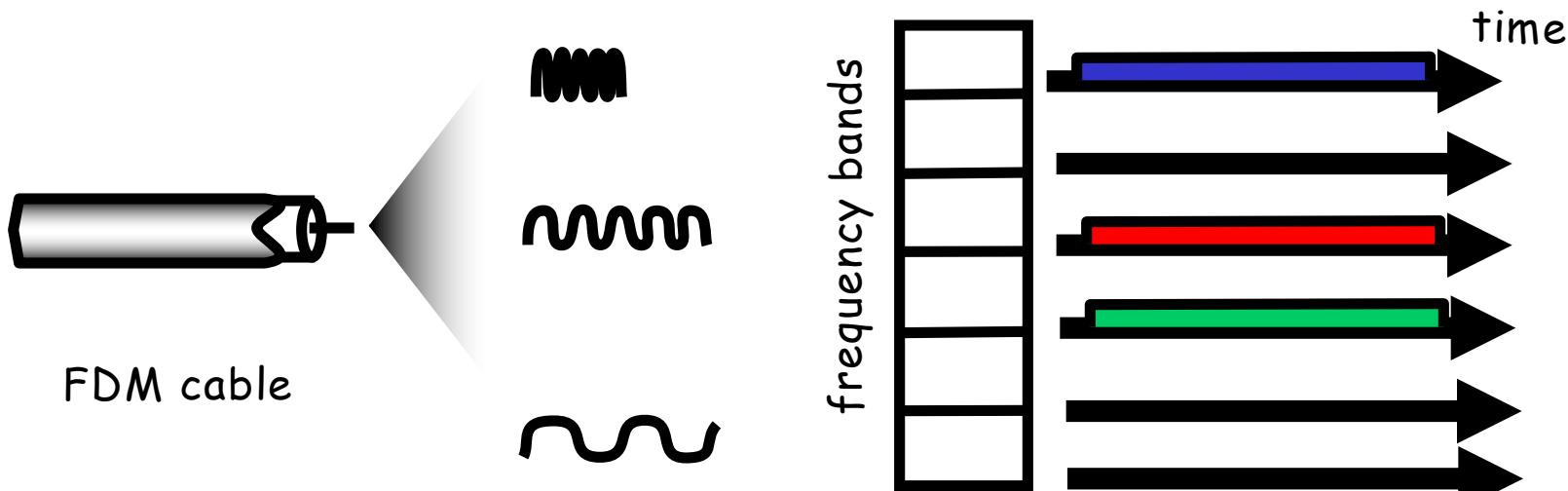
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



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



TDMA/FDMA Vs. Ideal Multiple Access Protocol

Broadcast channel of rate R bps

1. when one node wants to transmit, it can send at rate R. → NOT MET BY TDMA/FDMA
2. when M nodes want to transmit, each can send at average rate R/M → MET BY TDMA/FDMA
3. fully decentralized:
 - no special node to coordinate transmissions
 - no synchronization of clocks, slots
4. simple

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

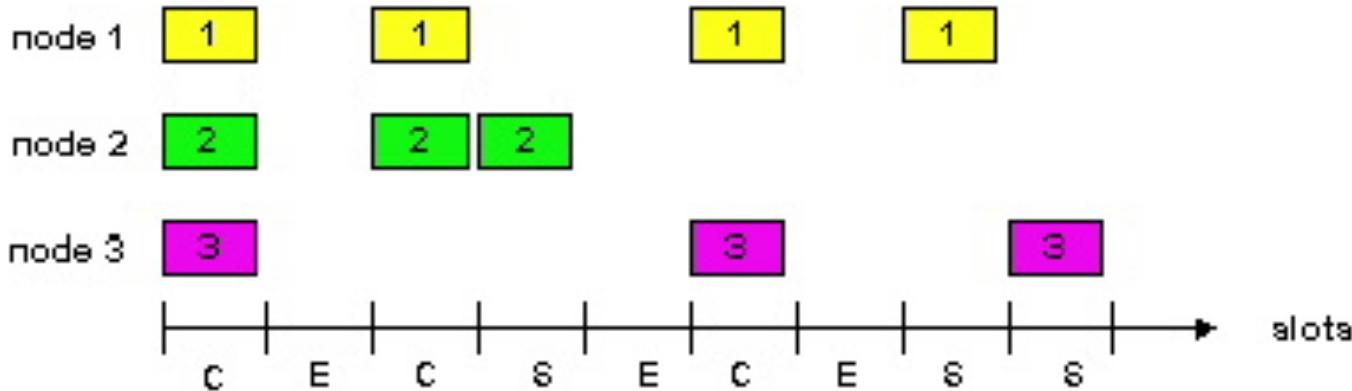
Assumptions:

- 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



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:

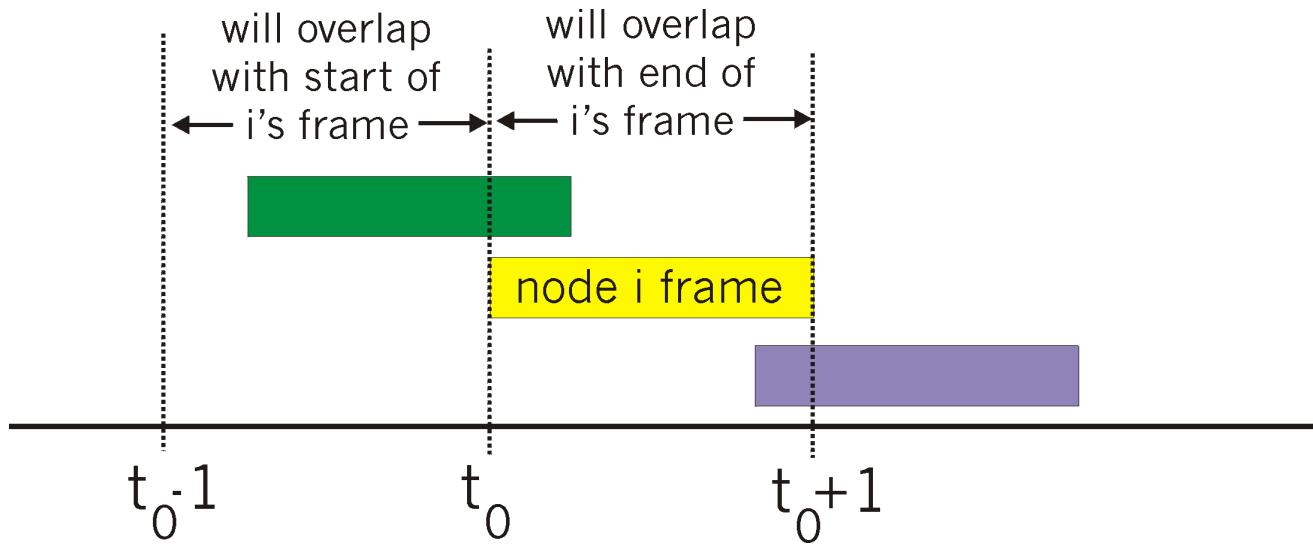
$$\text{Max efficiency} = 1/e = .37$$

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

$P(\text{success by given node}) = P(\text{node transmits}) \cdot$

$P(\text{no other node transmits in } [p_0-1, p_0]) \cdot$

$P(\text{no other node transmits in } [p_0, p_0+1,])$

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

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

... choosing optimum p and then letting $n \rightarrow \infty$...

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

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*:

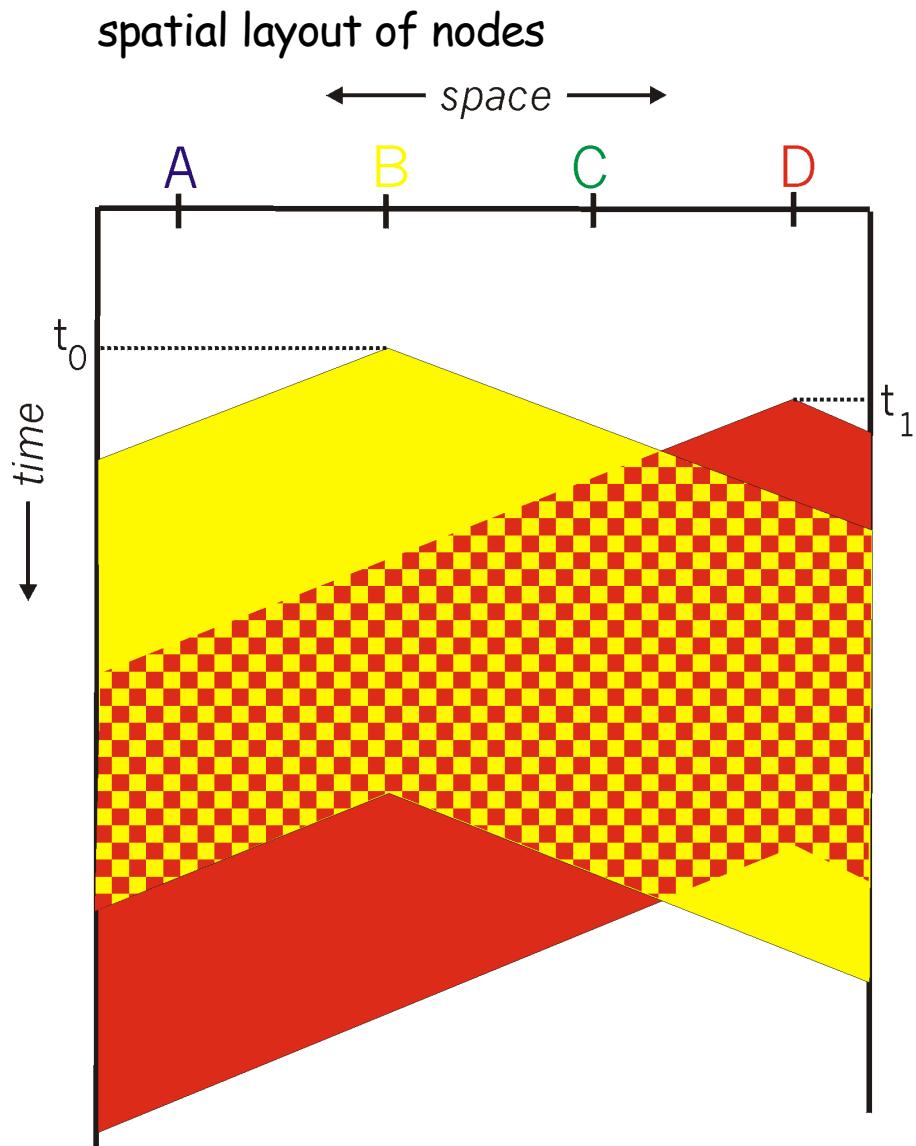
propagation delay means
two nodes may not hear
each other's transmission

collision:

entire packet transmission
time wasted

note:

role of distance & propagation
delay in determining collision
probability

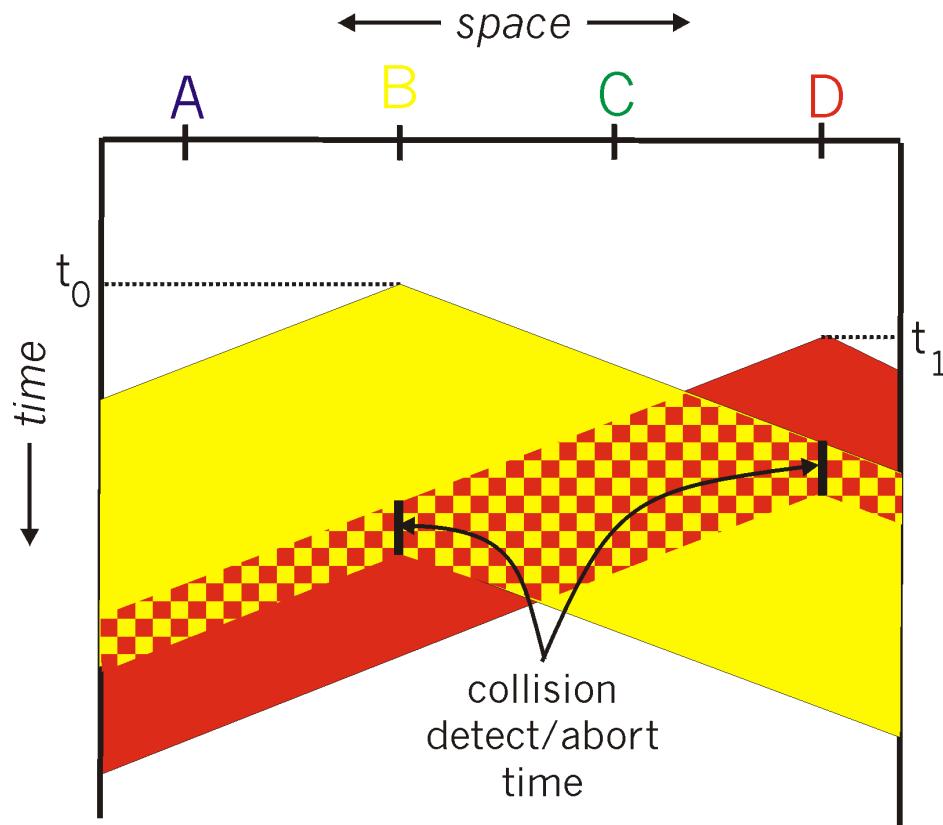


CSMA/CD (Collision Detection)

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



“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

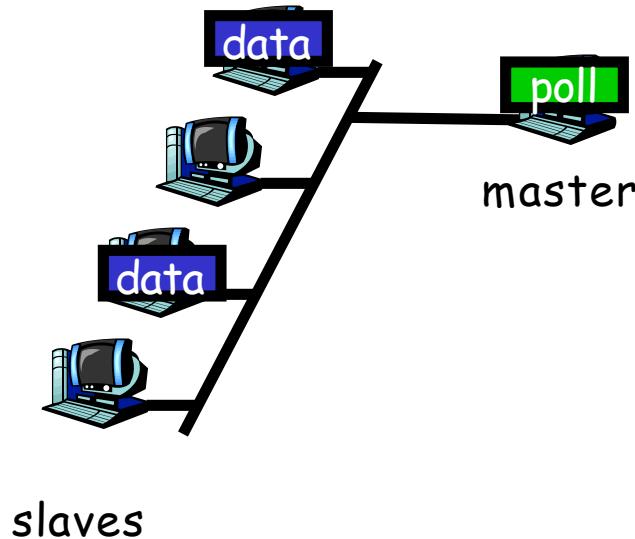
“taking turns” protocols

look for best of both worlds!

“Taking Turns” MAC protocols

Polling:

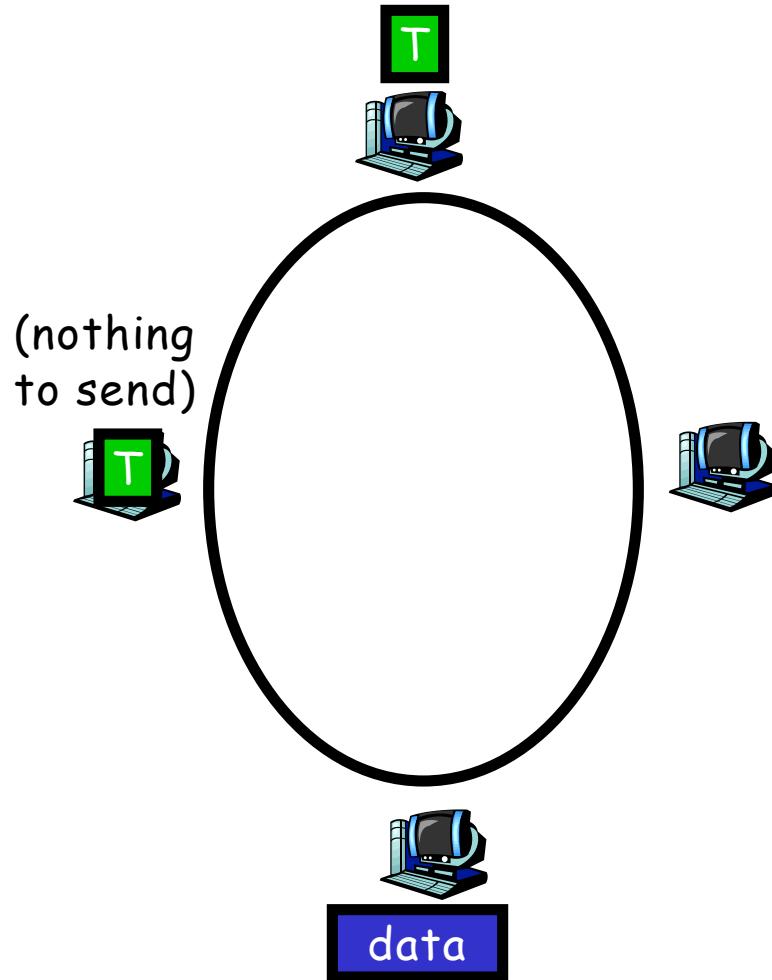
- ❑ master node “invites” slave nodes to transmit in turn
- ❑ typically used with “dumb” slave devices
- ❑ concerns:
 - polling overhead
 - latency
 - single point of failure (master)



“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
 - 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, IBM Token Ring