THE UNIVERSITY OF ADELAIDE
AUSTRALIA

COMPUTER NETWORKS AND APPLICATIONS

COMP SCI 3001
Faculty of Engineering, Computer and Mathematical Sciences

Error Detection and Error Correction

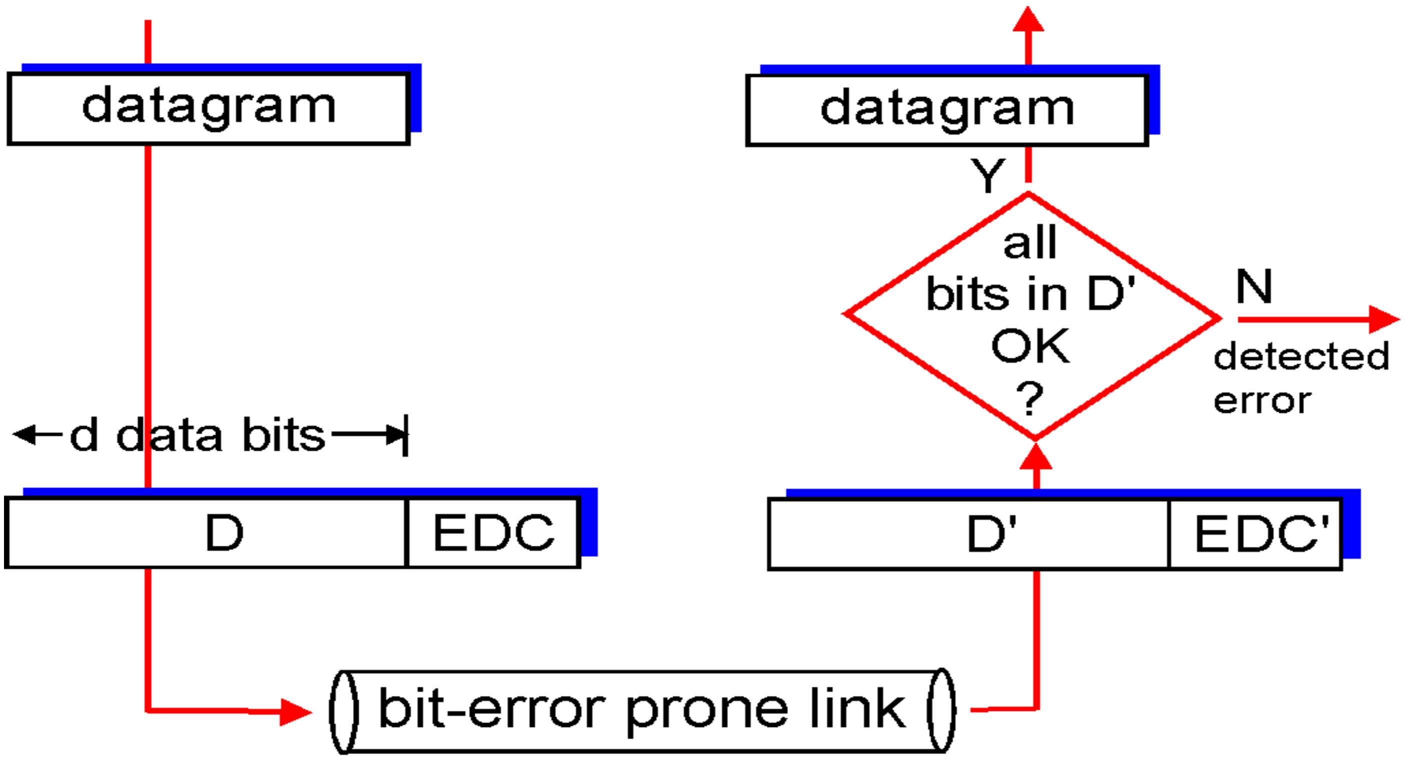
Error detection

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**

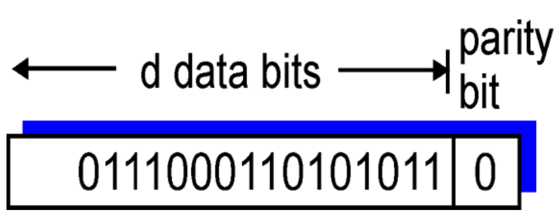
Error detection



Parity

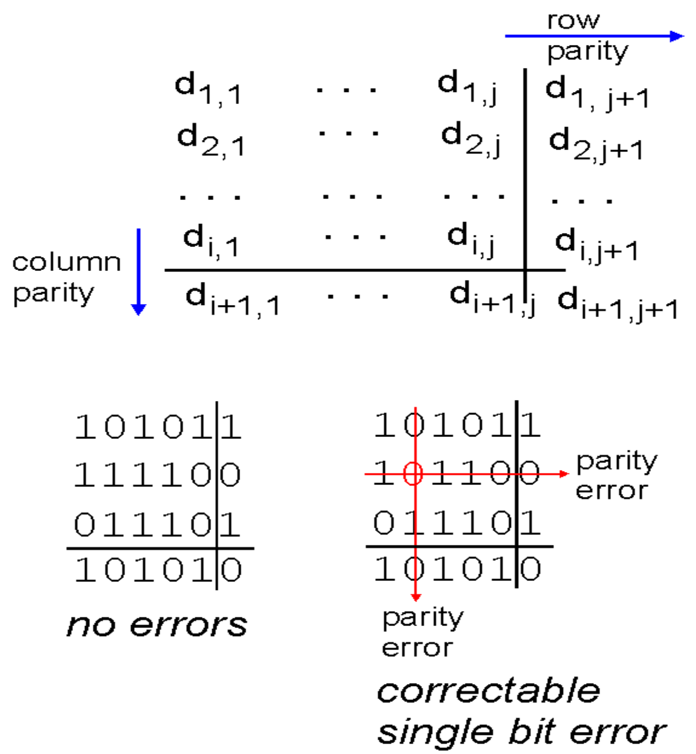
Single bit parity

Detect single bit errors



Two dimensional bit parity

Detect **and correct** single bit errors

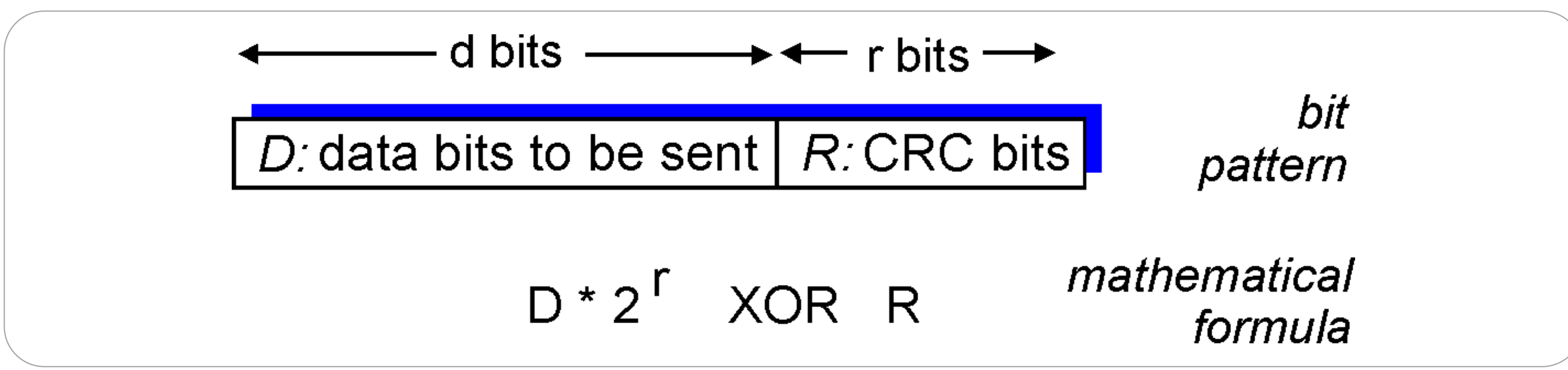


Checksums

- Errors do **not usually occur as a one-off single bit error**
  - we normally have an error burst
  - burst length = k implies bit 0 and k are in error, but some of the others **might** be OK
- Checksums deal with multiple bit errors
  - already seen checksums (remember TCP?)
- Internet checksum is 1's complement sum of the segment contents (viewed as 16 bit numbers)
- General principle of checksums
  - sender computes checksum and sends it
  - receiver computes and compares

Checksums - CRCs

- Cyclic Redundancy Check polynomials
- 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 and 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 (ATM, HDLC)



COMPUTER NETWORKS AND APPLICATIONS

COMP SCI 3001

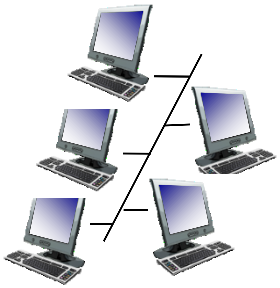
Faculty of Engineering, Computer and Mathematical Sciences

Multiple Access Protocols

Multiple access links

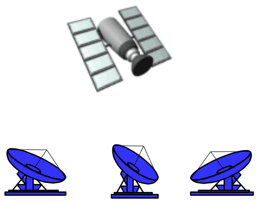
Two types of "links":

- point-to-point
  - Point-to-Point (PPP) (single wire, e.g. 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)

We need multiplexing strategies

Multiplexing is the process of combining multiple signals and transmitting them over a common channel and it is implemented on telecommunication using **Multiple Access Protocols implemented as an application**.

Multiple access protocols - taxonomy

Three broad classes:

1) Channel partitioning

- Divide channel into smaller ‘pieces’ (time slots, frequency)
- Allocate piece to node for exclusive use

2) Random access

- channel not divided, **allow** collisions (deal with them)
- ‘Recover’ from collisions

3) ‘Taking turns’

- Tightly coordinate shared access to **avoid** collisions

Goal: efficient, fair, simple, decentralised

Channel Partitioning methods – What to look for…

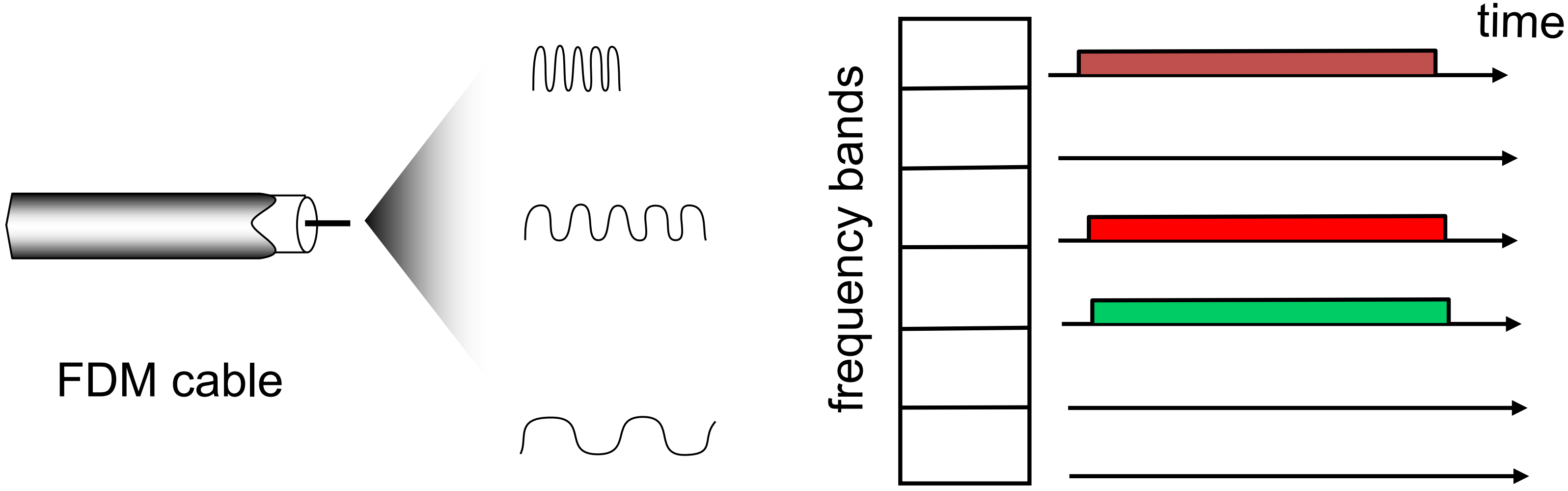
- Fairness
- Efficiency
- Complexity (Cost of implementing)
- Centralised or Decentralised

Channel partitioning MAC protocols

Frequency Division Multiple Access (FDMA)

- Channel spectrum divided into frequency bands
- Each station assigned fixed frequency band
- Unused transmission time in frequency bands go idle

eg 6-station LAN, 1,3,4 have packets, frequency bands 2,5,6 idle

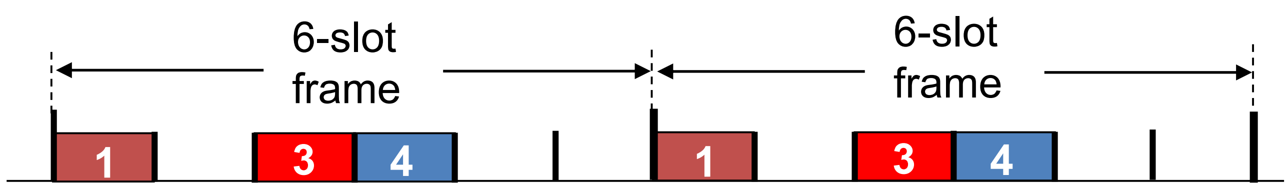


Channel partitioning MAC protocols

Time Division Multiple Access (TDMA)

- Access to channel in 'rounds'
- Each station gets fixed length slot (length = packet transmission time) in each round
- Unused slots go idle

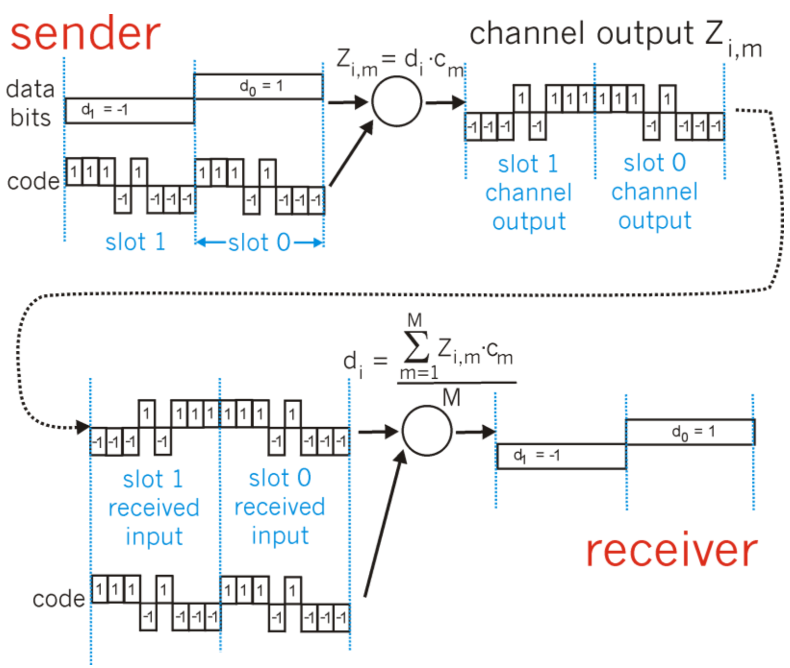
eg 6-station LAN, 1, 3, 4 have packets, slots 2, 5, 6 idle



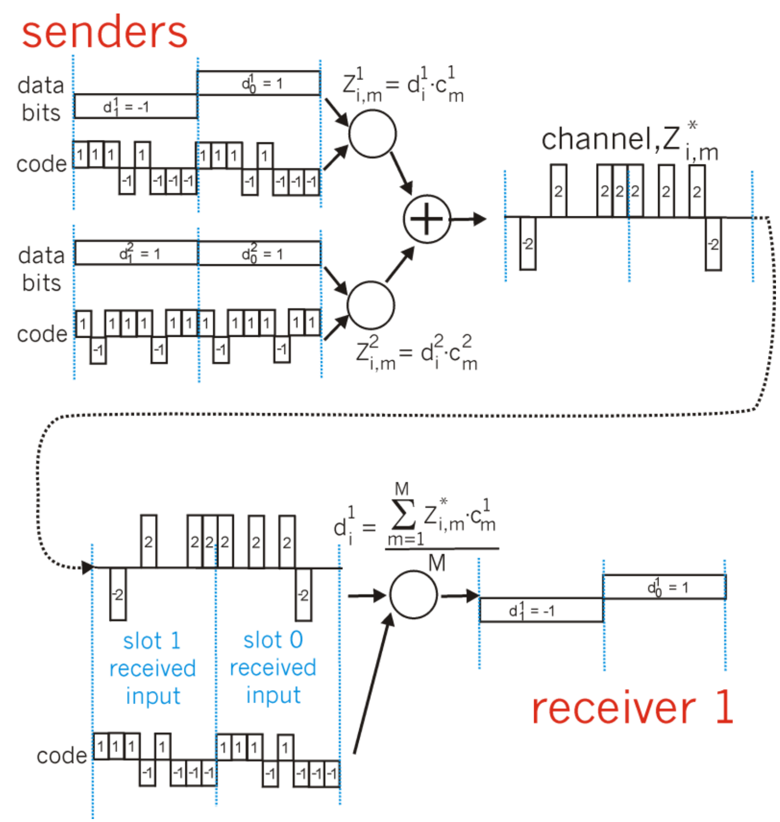
Code Division Multiple Access (CDMA)

- Unique ‘code’ assigned to each user, ie code set partitioning
- Used mostly in wireless broadcast channels (cellular, satellite etc)
- All users share same frequency, but each user has own ‘chipping’ sequence (ie code) to encode data
- **Encoded signal** = (original data) X (chipping sequence)
- **Decoding:** inner product of encoded signal and chipping sequence
- Allows multiple users to ‘coexist’ and transmit simultaneously with minimal interference (if codes are ‘orthogonal’)

CDMA encode/decode sequence



CDMA interference

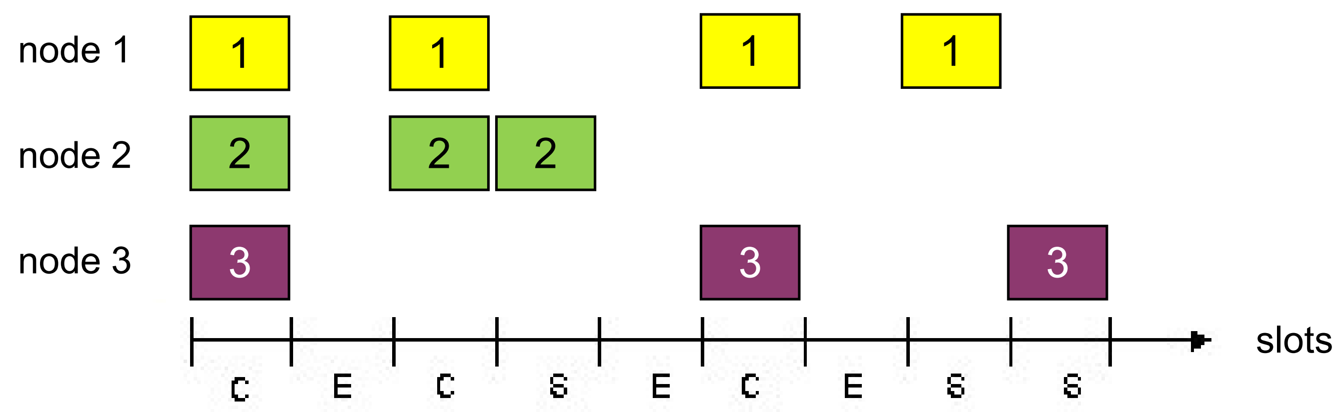


Random access protocols introduction

- 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 need to deal with:
  1. how to detect collisions
  2. how to recover from collisions (e.g., via delayed retransmissions)
- examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

Random access protocols: Slotted ALOHA

- Time is divided into equal size slots (= packet transmit time)
- Node with new arriving packet: transmit at beginning of next slot
- If there’s a collision: re-transmit the packet in future slots with probability p, until successful



Success (S), Collision (C), Empty (E) slots

What are the advantages and problems with slotted ALOHA?

Slotted ALOHA : efficiency

efficiency: long-run fraction of successful slots

Suppose N stations **have packets 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 $Np(1-p)^{N-1}$

Then as number of nodes become large

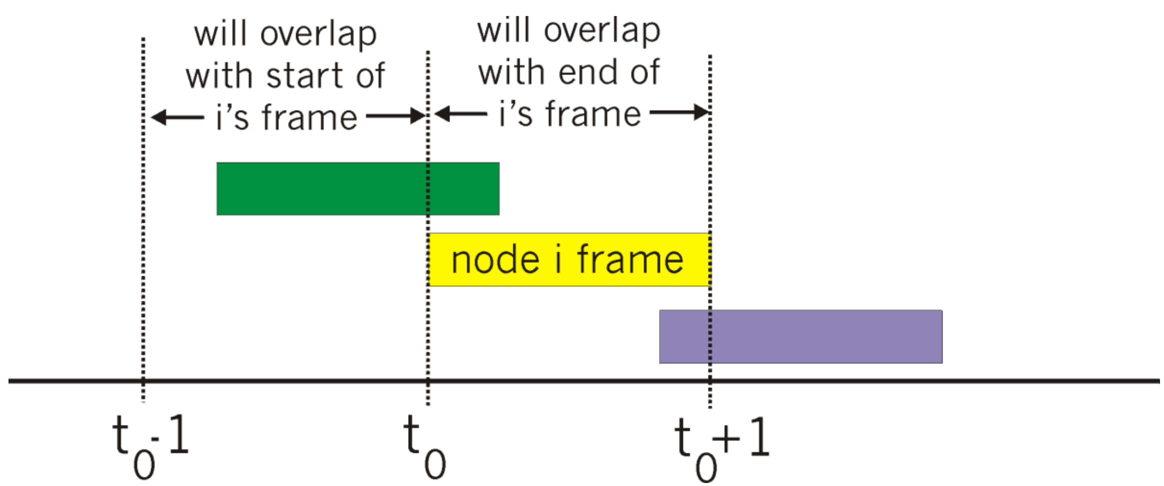
- take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives:

max efficiency = 1/e = .37

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

Pure (unslotted) ALOHA

- Unslotted ALOHA: simpler, no synchronisation
- Packet needs transmission
  - transmit immediately without awaiting for beginning of slot
- Collision probability **increases**
  - frame sent at t_0 collides with other packets 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 $[t_0,t_0+1]$]

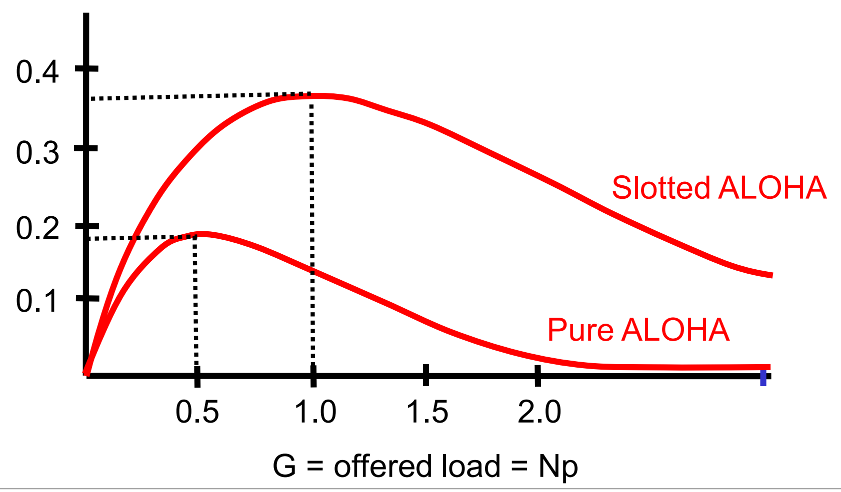
= $p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$

P (success by any of N nodes) = $Np \cdot (1-p)^{(N-1)} \cdot (1-p)^{(N-1)}$

…choosing optimum ***p*** and letting n > infinity...

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

protocol *constrains* effective channel throughput!



Random access protocols: Carrier Sense Multiple Access (CSMA)

CSMA - **listen before transmit**

- If channel sensed idle: transmit entire packet
- If channel sensed busy, defer transmission

Human analogy: don't interrupt others!

1. **persistent CSMA**: retry *immediately* with probability *p* when channel becomes idle (may cause instability)
2. **non-persistent CSMA**: retry after *random interval*

Why is this better than ALOHA?

Carrier Sense Multiple Access (CSMA)

Are collisions still possible?

Carrier Sense Multiple Access (CSMA)

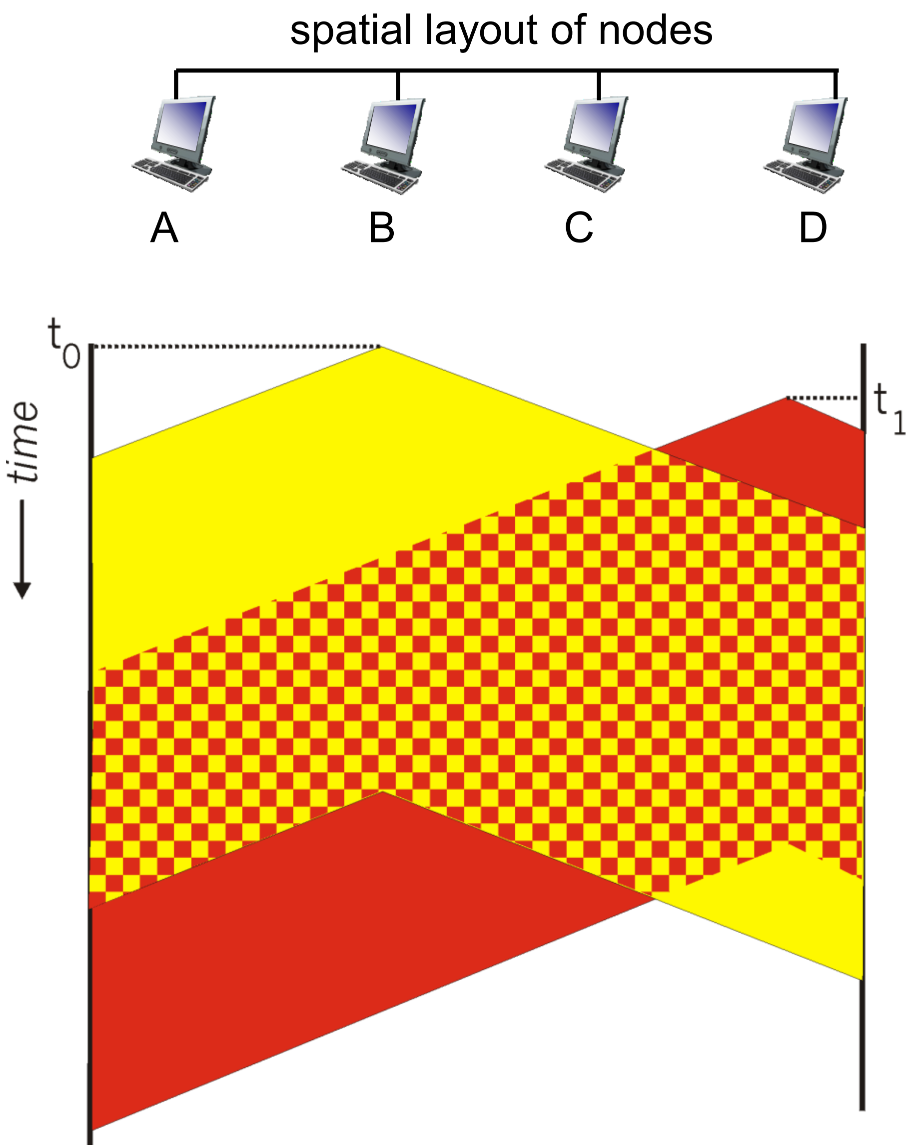
We can listen then how can collisions still occur?

CSMA collisions

- ***Collisions can occur due to propagation delay***
- Collision means: entire packet transmission time is wasted - up to **two packet times**...

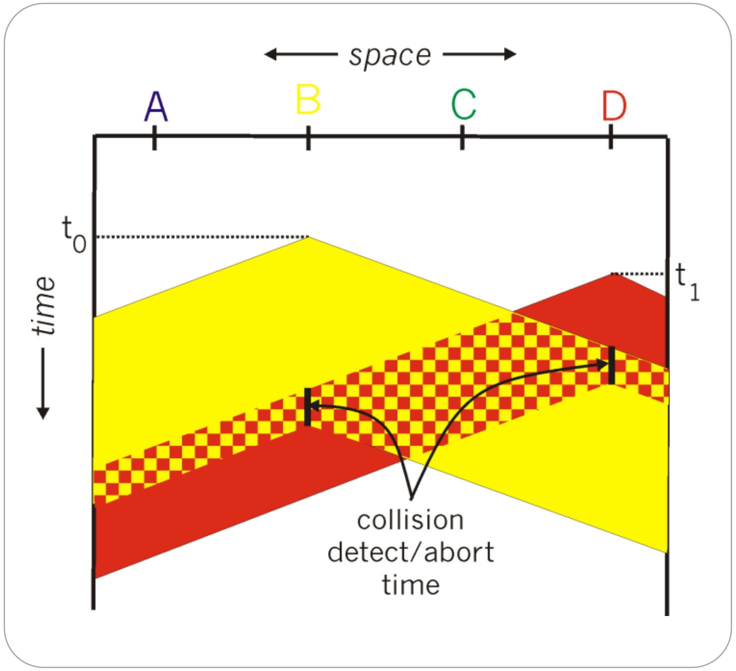
NOTE

- The propagation delay determines the collision detection time!
- Don't forget the speed of light!



CSMA/CD (Collision Detection)

- Carrier sensing, deferral as in CSMA
- Collisions **detected** within short time
- Colliding transmissions aborted, reducing channel wastage
- Persistent or non-persistent retransmission
- *E.g Ethernet => not needed now!*



Collision detection:

- Easy in wired LANs: ***measure signal strengths***, ***compare transmitted*** and ***received signals***
- Difficult in wireless LANs: receiver shut off while transmitting

Human analogy: the polite conversationalist

Random access protocols: Taking turns

Channel partitioning MAC protocols?

- Share channel efficiently 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*** utilise channel
- High load: ***collision*** overhead

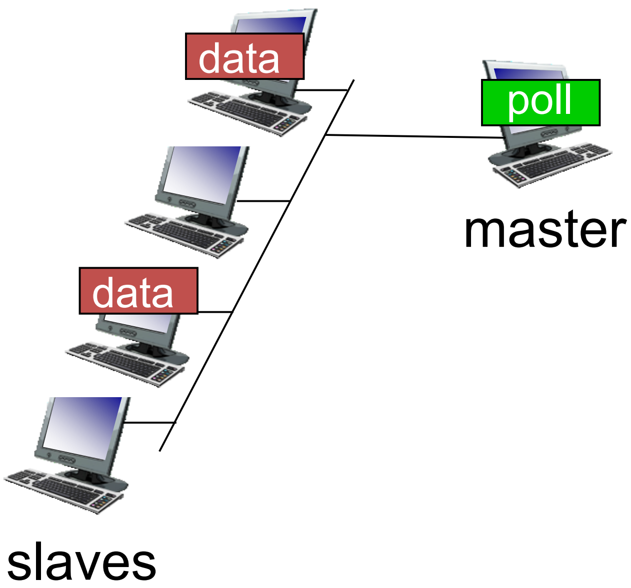
Taking turns protocols

- Look for best of both worlds!

Taking turns protocols

polling:

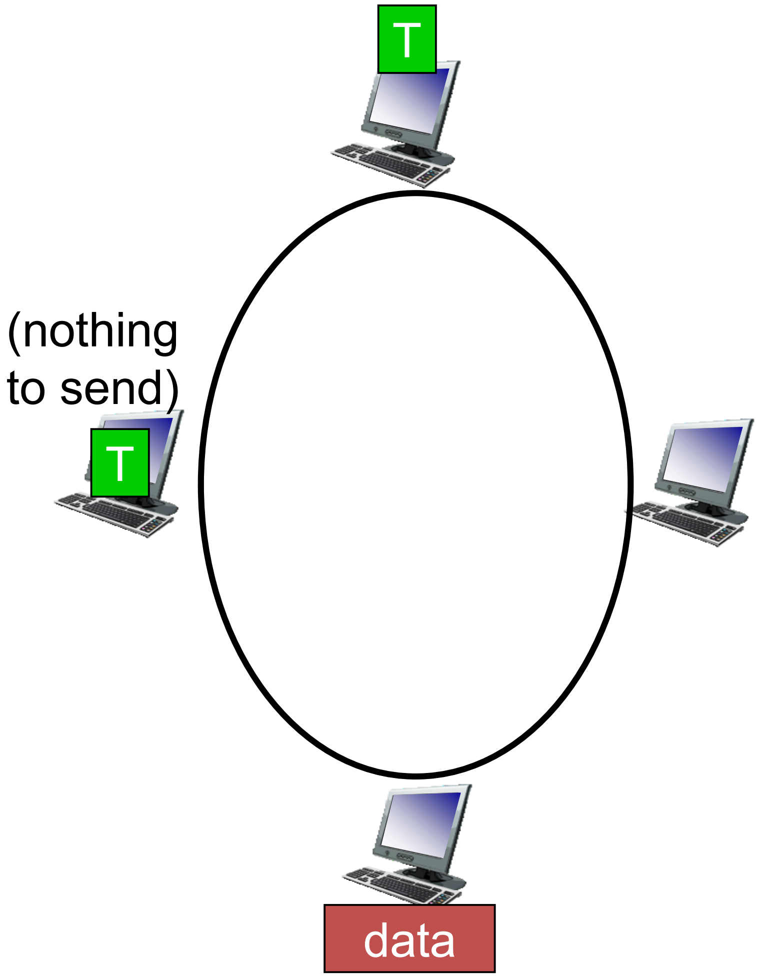
- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



Taking turns protocols (cont.)

token passing

- Control **token** passed from one node to next sequentially
- Token message
- Concerns
  - token overhead
  - latency
  - single point of failure (token)



Reservation based protocols

Distributed polling

- Time divided into slots
- Begins with N short **reservation slots**
  - reservation slot time equal to channel end-end propagation delay
  - station with message to send posts reservation
  - reservation seen by all stations
- After reservation slots, message transmissions ***ordered by known priority***