



CSC/CPE 138 - Computer Network Fundamentals

The Link Layer and LANs

The presentation was adapted from the textbook: *Computer Networking: A Top-Down Approach* 8th edition Jim Kurose, Keith Ross, Pearson, 2020

Redefine the Possible™

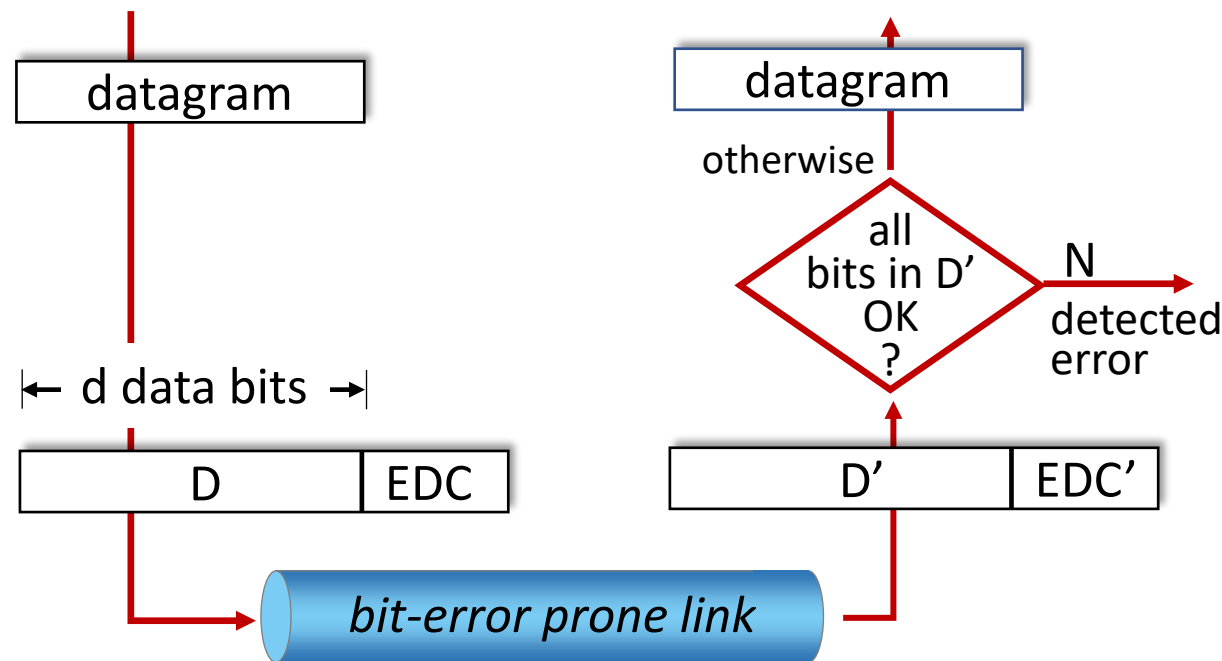
- introduction
- **error detection, correction**
- multiple access protocols
- LANs
 - addressing, ARP
 - Ethernet
 - switches
 - VLANs
- link virtualization: MPLS
- data center networking



- a day in the life of a web request

EDC: error detection and correction bits (e.g., redundancy)

D: data protected by error checking, may include header fields



Error detection not 100% reliable!

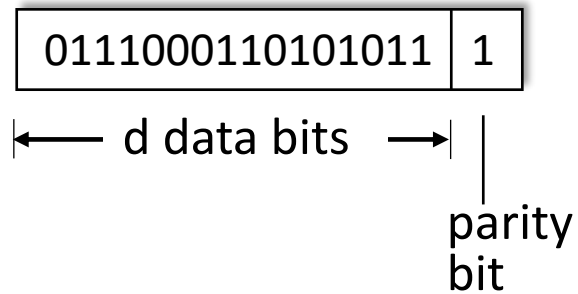
- protocol may miss some errors, but rarely
- larger EDC field yields better detection and correction

Parity checking



single bit parity:

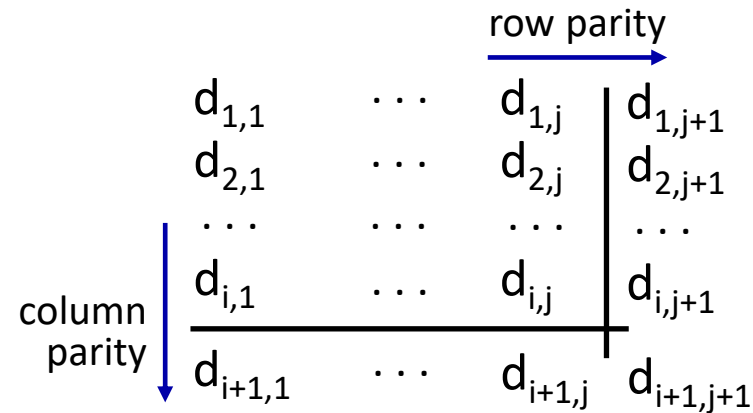
- detect single bit errors



Even parity: set parity bit so there is an even number of 1's

two-dimensional bit parity:

- detect *and correct* single bit errors



no errors:

1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
1	0	1	0	1	0

detected
and
correctable
single-bit
error:

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
1	0	1	0	1	0

parity error (pointing to the 0 in row 2, column 2)

parity error (pointing to the 0 in row 4, column 1)

Goal: detect errors (*i.e.*, flipped bits) in transmitted segment

sender:

- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- **checksum:** addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

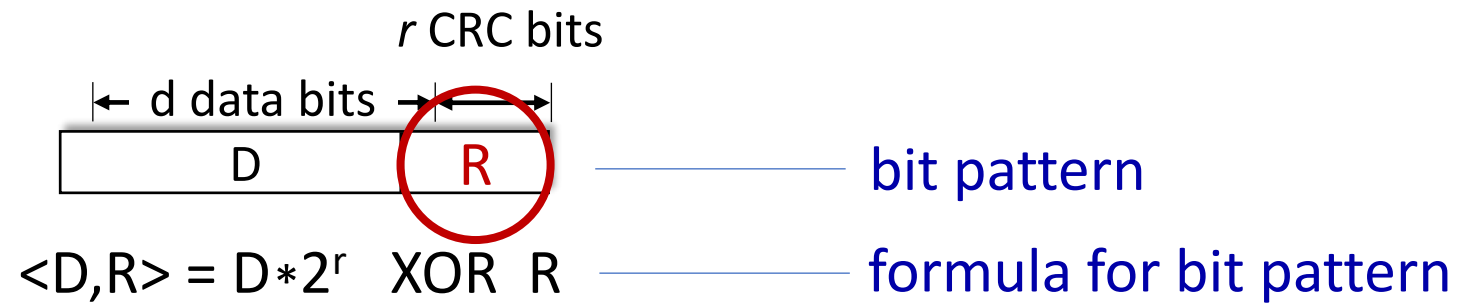
receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
 - not equal - error detected
 - equal - no error detected. *But maybe errors nonetheless?* More later

Cyclic Redundancy Check (CRC)



- more powerful error-detection coding
- **D**: data bits (given, think of these as a binary number)
- **G**: bit pattern (generator), of $r+1$ bits (given)



goal: choose r CRC bits, **R**, such that $\langle D, R \rangle$ exactly divisible by $G \pmod{2}$

- receiver knows G , divides $\langle D, R \rangle$ 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)

Cyclic Redundancy Check (CRC): example



We want:

$$D \cdot 2^r \text{ XOR } R = nG$$

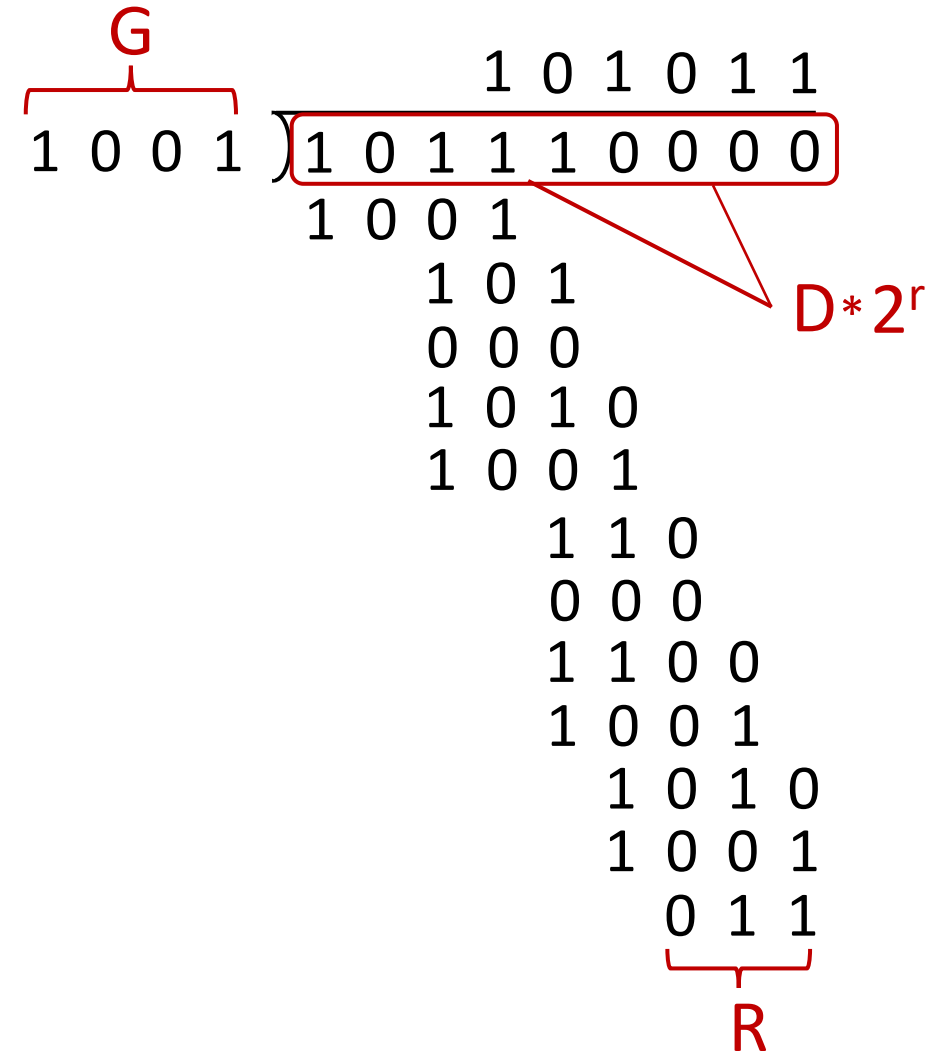
or equivalently:

$$D \cdot 2^r = nG \text{ XOR } R$$

or equivalently:

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

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



- introduction
- error detection, correction
- **multiple access protocols**
- LANs
 - addressing, ARP
 - Ethernet
 - switches
 - VLANs
- link virtualization: MPLS
- data center networking



- a day in the life of a web request

Multiple access links, protocols



two types of “links”:

- point-to-point
 - point-to-point link between Ethernet switch, host
 - PPP for dial-up access
- **broadcast (shared wire or medium)**
 - old-fashioned Ethernet
 - upstream HFC in cable-based access network
 - 802.11 wireless LAN, 4G/4G. satellite



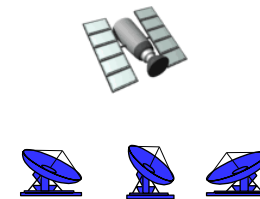
shared wire (e.g.,
cabled Ethernet)



shared radio: 4G/5G



shared radio: WiFi



shared radio: satellite



humans at a cocktail party
(shared air, acoustical)

- 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

given: multiple access channel (MAC) of rate R bps

desiderata:

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

three broad classes:

- **channel partitioning**

- divide channel into smaller “pieces” (time slots, frequency, code)
- allocate piece to node for exclusive use

- ***random access***

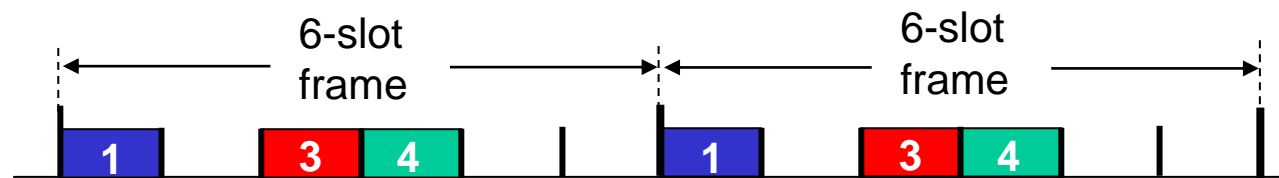
- channel not divided, allow collisions
- “recover” from collisions

- **“taking turns”**

- nodes take turns, but nodes with more to send can take longer turns

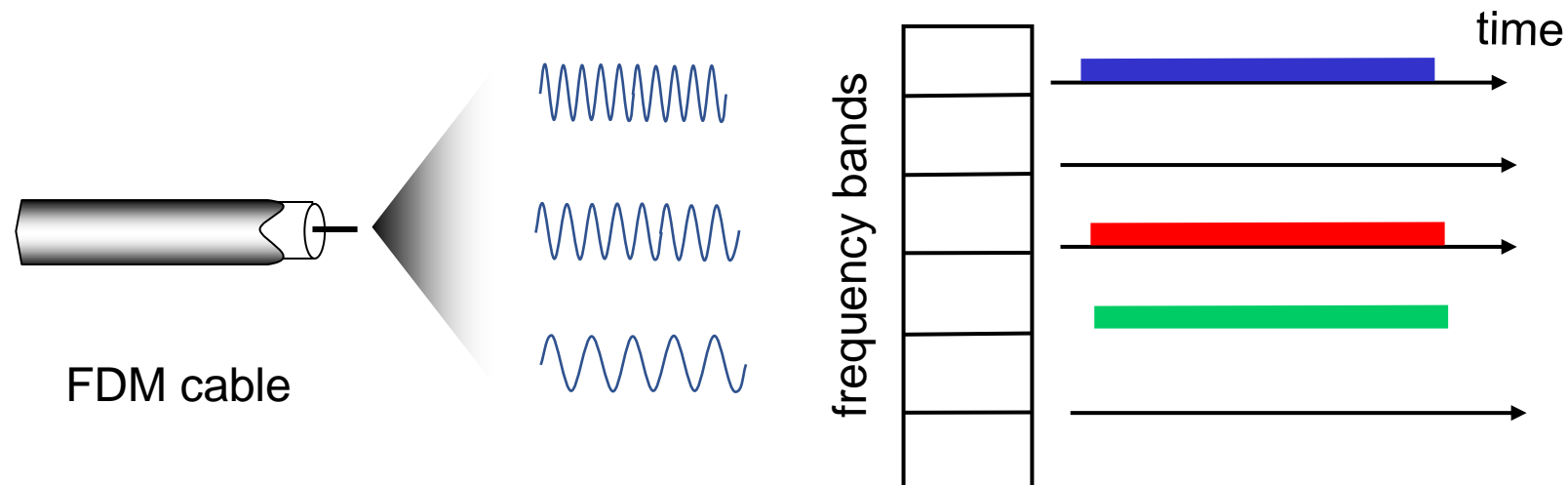
TDMA: time division multiple access

- access to channel in “rounds”
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



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 packet to send, frequency bands 2,5,6 idle



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

assumptions:

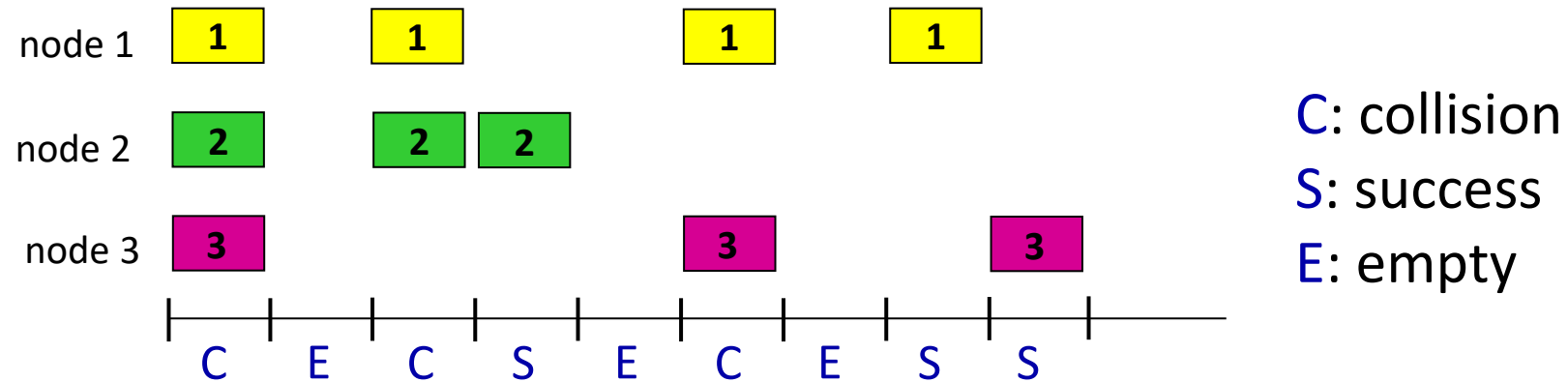
- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

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 probability p until success

randomization – why?

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

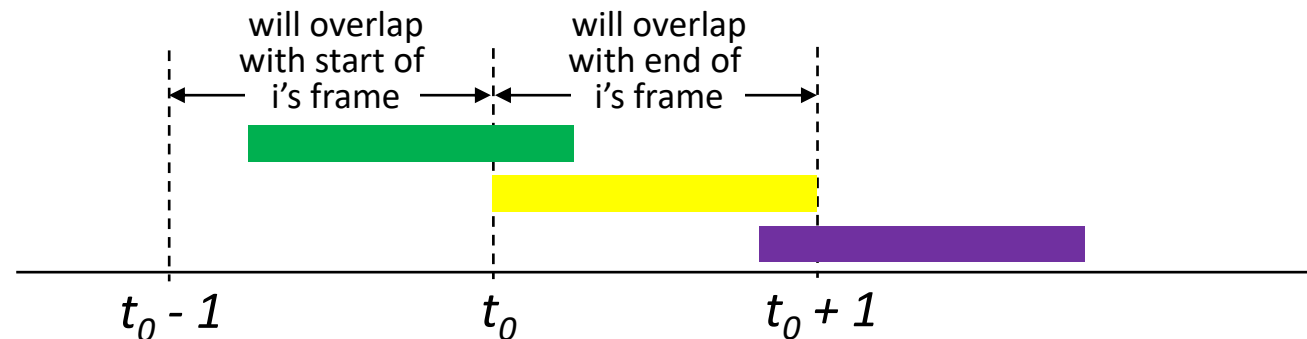
efficiency: long-run fraction of successful slots (many nodes, all with many frames to send)

- *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!

- unslotted Aloha: simpler, no synchronization
 - when frame first arrives: transmit immediately
- collision probability increases with no synchronization:
 - frame sent at t_0 collides with other frames sent in $[t_0-1, t_0+1]$



- prob that *any* node has a success = $p(1-p)^{2(N-1)}$
- max efficiency: find p^* that maximizes $p(1-p)^{2(N-1)}$
- pure Aloha efficiency: 18% ($1/(2e)$) when N goes to infinity.

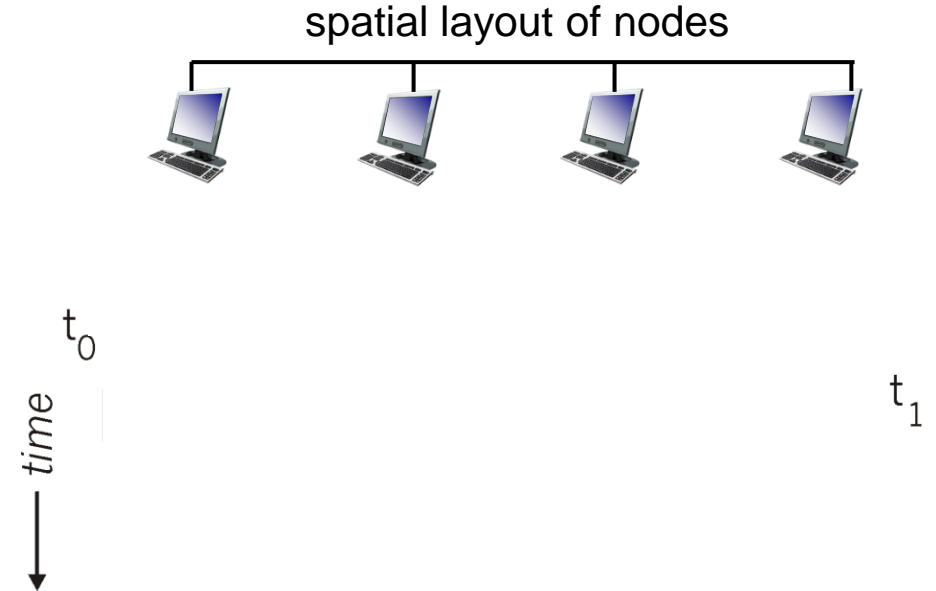
simple **CSMA**: listen before transmit:

- if channel sensed **idle**: transmit entire frame
 - if channel sensed **busy**: defer transmission
- human analogy: don't interrupt others!

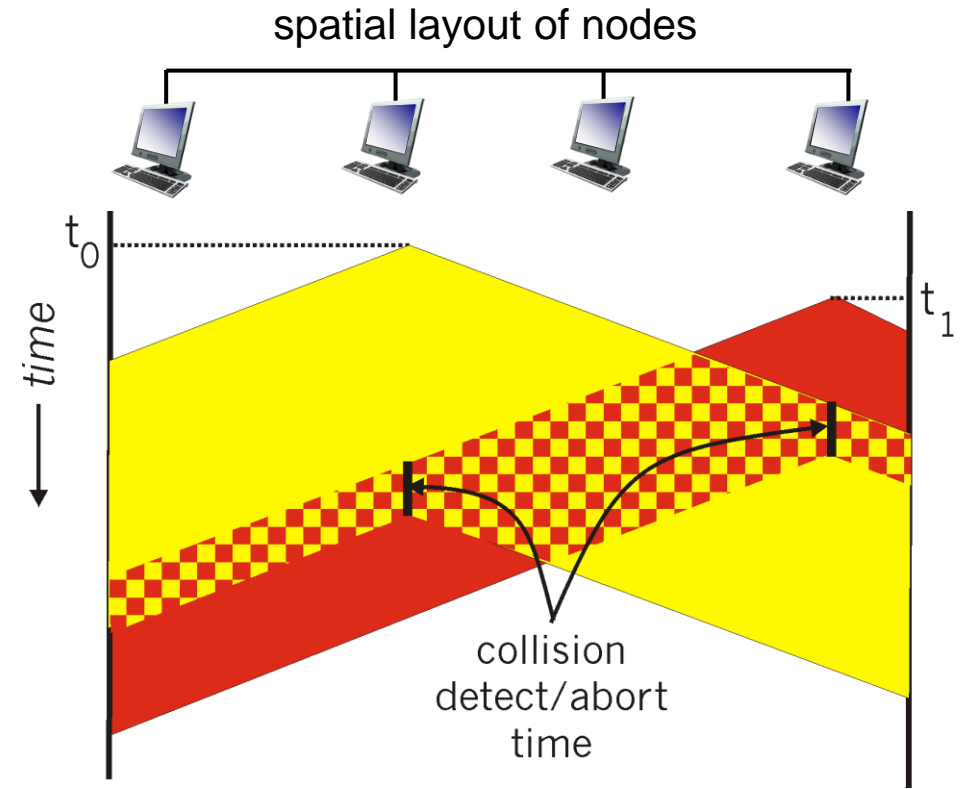
CSMA/CD: CSMA with *collision detection*

- collisions *detected* within short time
 - colliding transmissions aborted, reducing channel wastage
 - collision detection easy in wired, difficult with wireless
- human analogy: the polite conversationalist

- collisions *can* still occur with carrier sensing:
 - propagation delay means two nodes may not hear each other's just-started transmission
- **collision**: entire packet transmission time wasted
 - distance & propagation delay play role in determining collision probability



- CSMA/CS reduces the amount of time wasted in collisions
 - transmission aborted on collision detection



1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel:
 - if **idle**: start frame transmission.
 - if **busy**: wait until channel idle, then transmit
3. If NIC transmits entire frame without collision, NIC is done with frame !
4. If NIC detects another transmission while sending: abort, send jam signal
5. After aborting, NIC enters *binary (exponential) backoff*:
 - after m th collision, NIC chooses K at random from $\{0, 1, 2, \dots, 2^m - 1\}$. NIC waits $K \cdot 512$ bit times, returns to Step 2
 - more collisions: longer backoff interval

- 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!

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

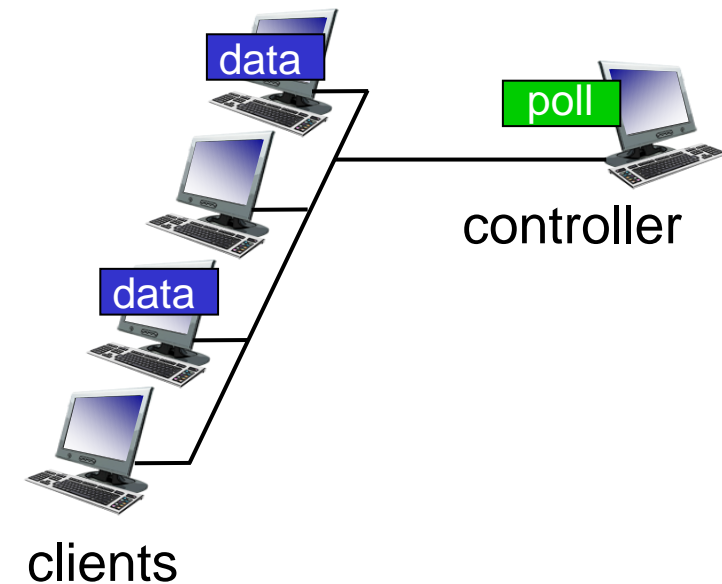
- look for best of both worlds!

“Taking turns” MAC protocols



polling:

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

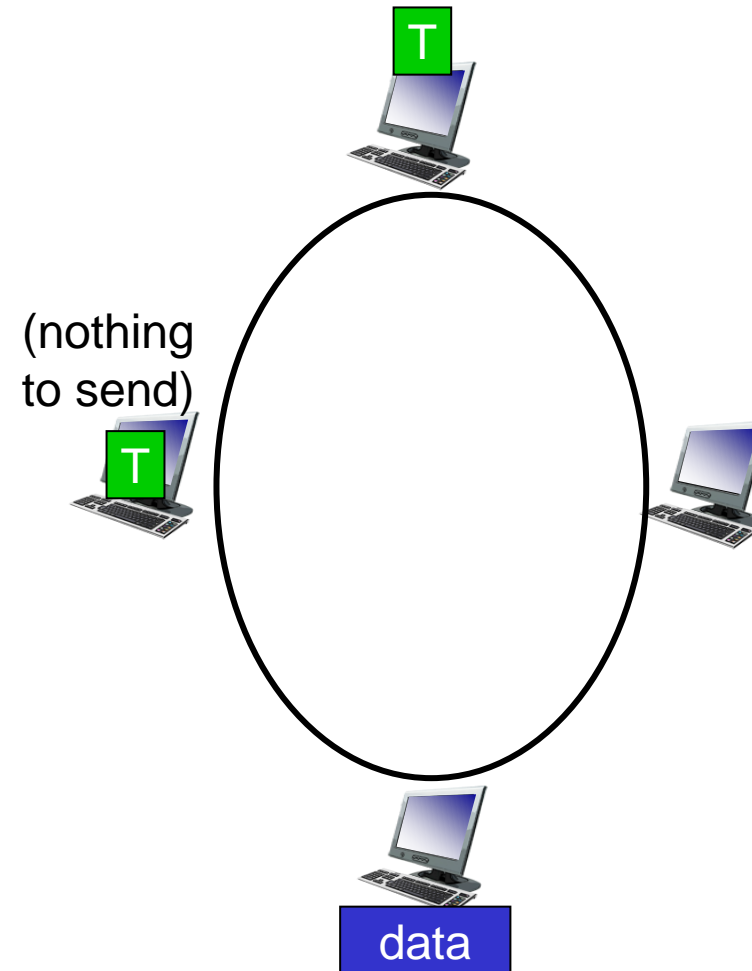


“Taking turns” MAC protocols

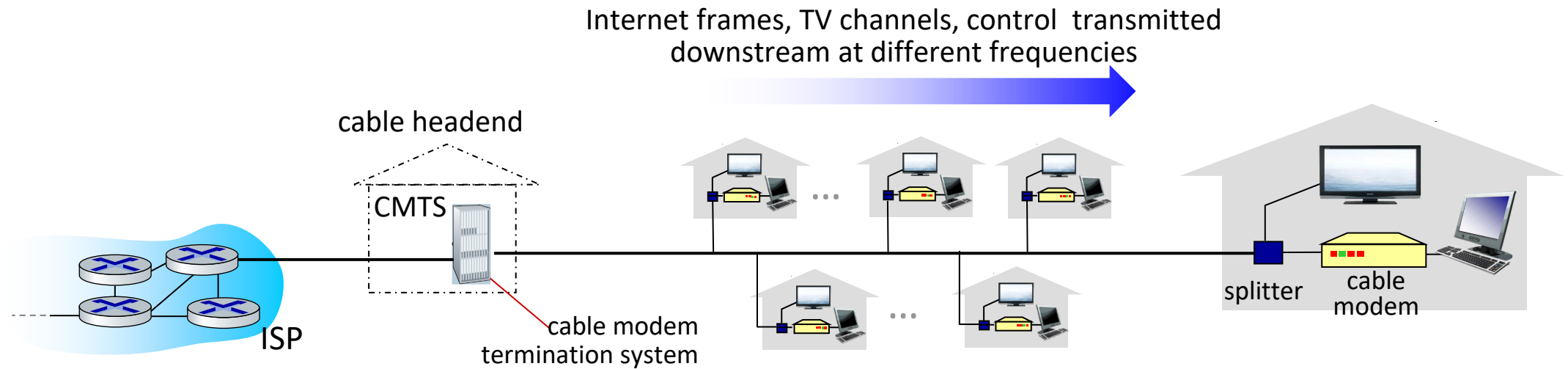


token passing:

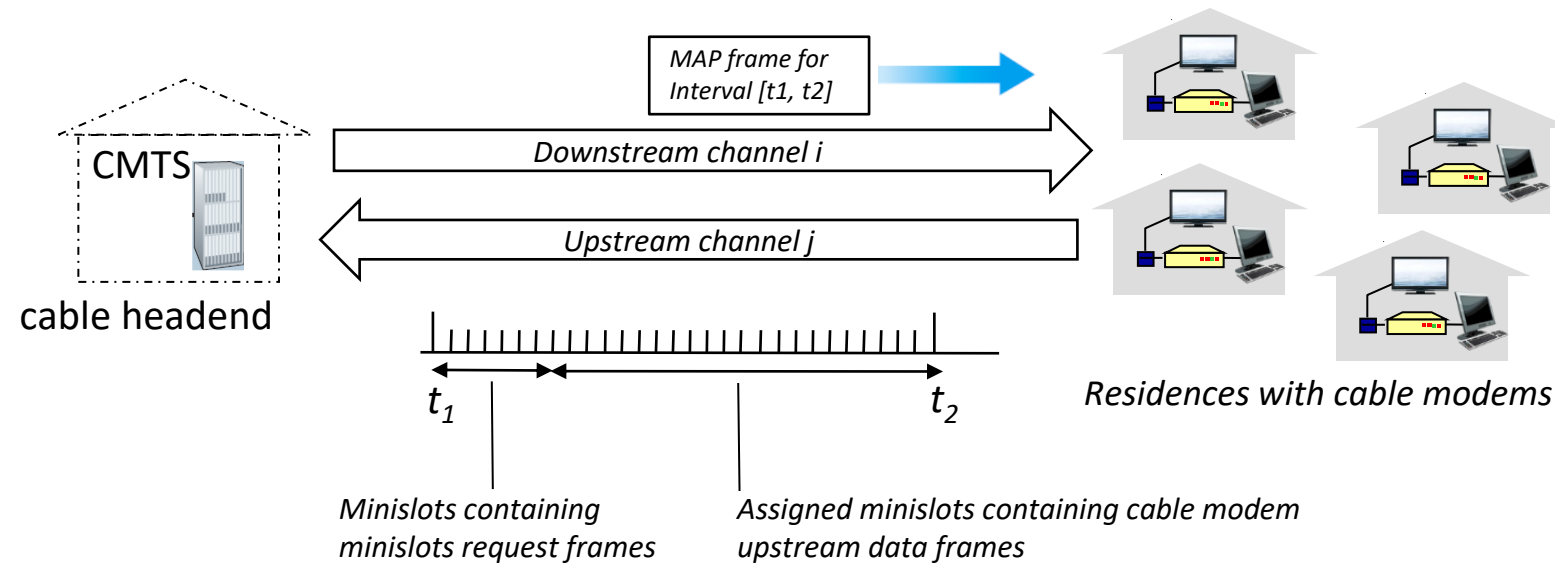
- control *token* passed from one node to next sequentially.
- token message
- concerns:
 - token overhead
 - latency
 - single point of failure (token)



Cable access network: FDM, TDM *and* random access!



- **multiple** downstream (broadcast) FDM channels: up to 1.6 Gbps/channel
 - single CMTS transmits into channels
- **multiple** upstream channels (up to 1 Gbps/channel)
 - **multiple access**: all users contend (random access) for certain upstream channel time slots; others assigned TDM



DOCSIS: data over cable service interface specification

- FDM over upstream, downstream frequency channels
- TDM upstream: some slots assigned, some have contention
 - downstream MAP frame: assigns upstream slots
 - request for upstream slots (and data) transmitted random access (binary backoff) in selected slots

- **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