

Multiple access links, protocols (Access Control)

two types of “links”:

- point-to-point
 - point-to-point link between Ethernet switch, host
- **broadcast (shared wire or medium)**
 - old-fashioned Ethernet
 - 802.11 wireless LAN, 4G/5G, satellite



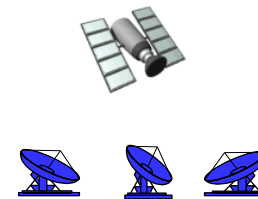
shared wire (e.g.,
cabled Ethernet)



shared radio: 4G/5G



shared radio: WiFi



shared radio: satellite



humans at a cocktail party
(shared air, acoustical)

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: multiple access channel (MAC) of rate R bps

Requirement:

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**

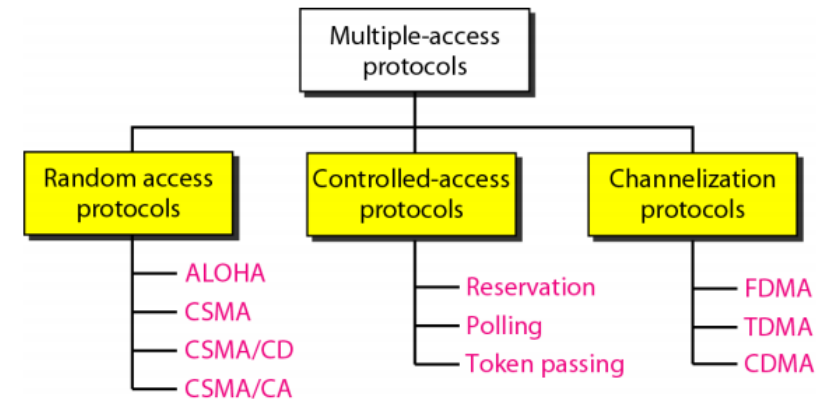
- divide channel into smaller “pieces” (time slots, frequency, code)
- allocate piece to node for exclusive use

- **random access**

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

- **“taking turns”**

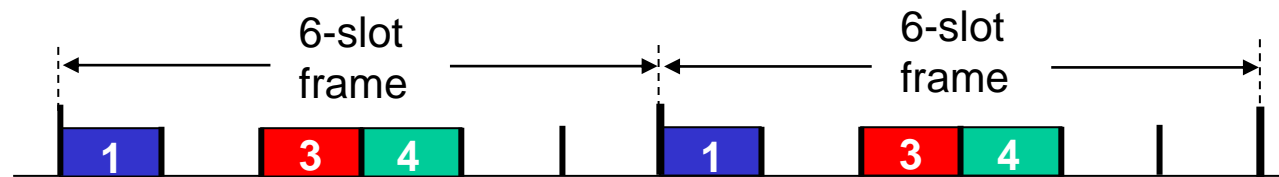
- nodes take turns, but nodes with more to send can take longer turns



Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

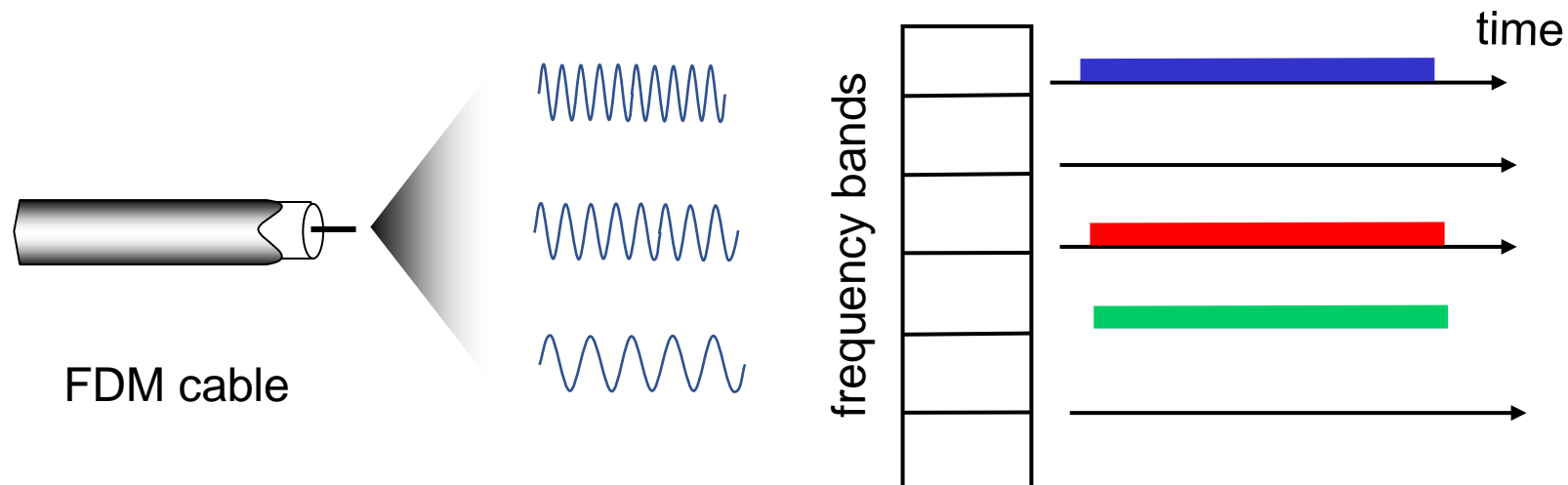
- access to channel in “rounds”
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



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 packet to send, 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:
 - ALOHA, slotted ALOHA
 - CSMA, CSMA/CD, CSMA/CA (**Important and in use**)

Slotted ALOHA

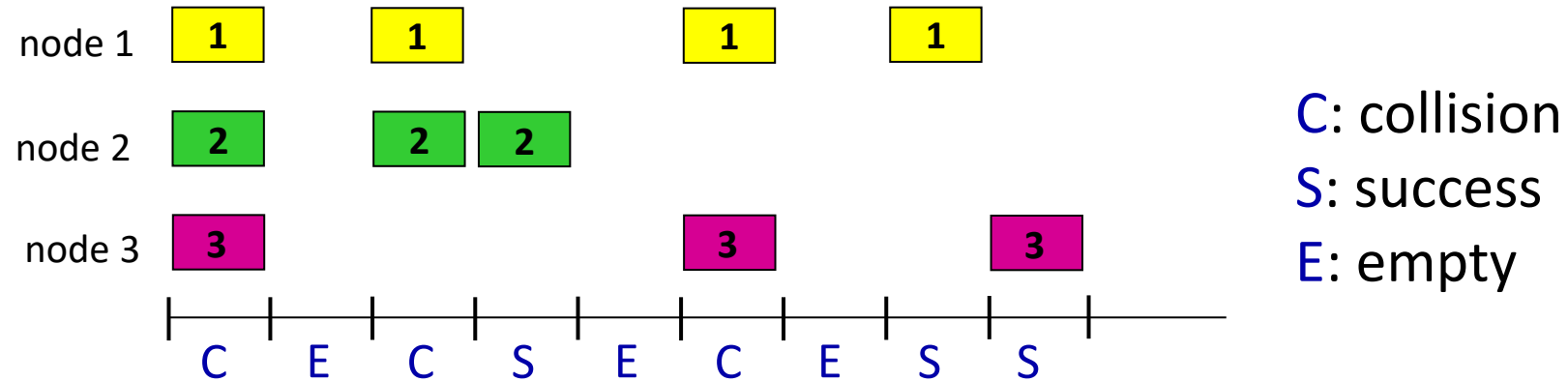
assumptions:

- 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 probability p until success

Slotted ALOHA



Pros:

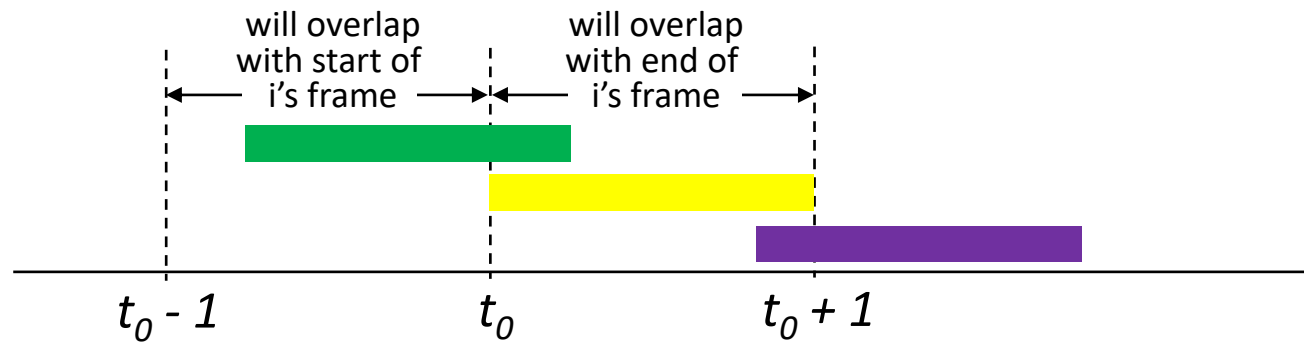
- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons:

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

Pure ALOHA

- 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 $[t_0-1, t_0+1]$



CSMA (carrier sense multiple access)

simple **CSMA**: listen before transmit:

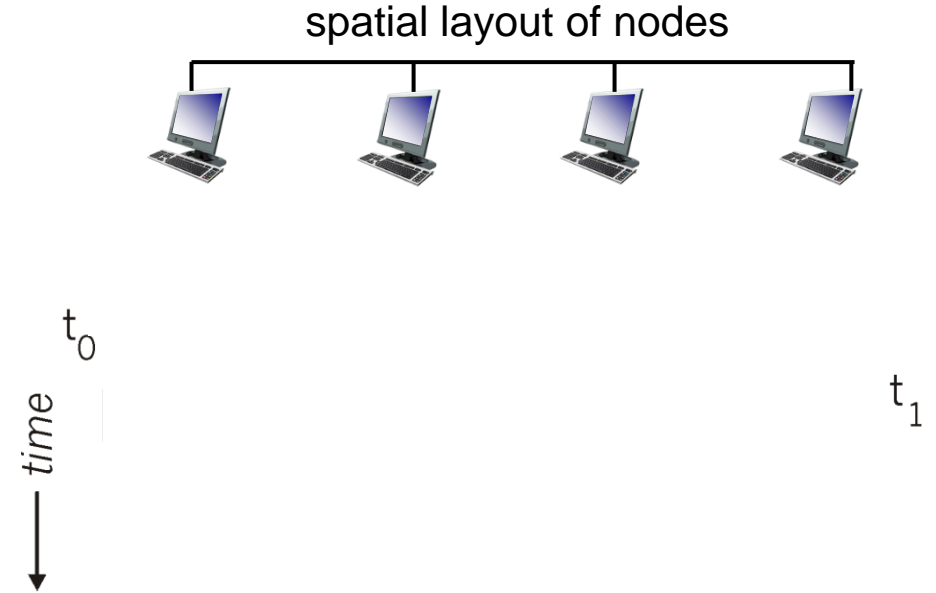
- if channel sensed idle: transmit entire frame
 - if channel sensed busy: defer transmission
- human analogy: don't interrupt others!

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

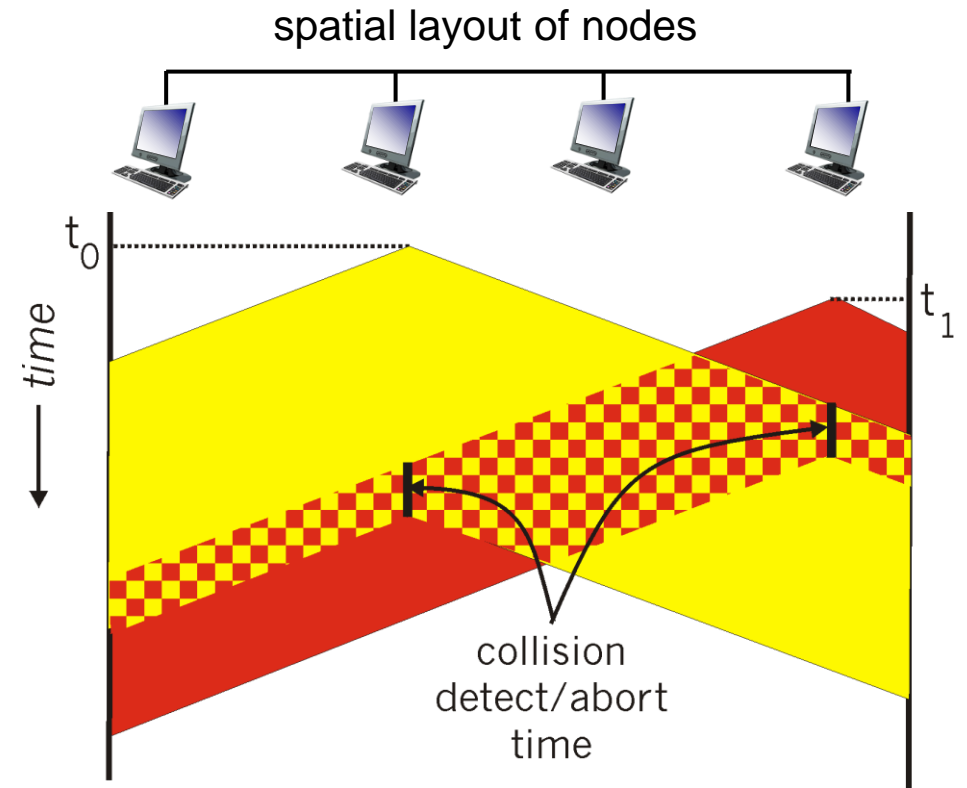
CSMA: collisions

- 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/CD:

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



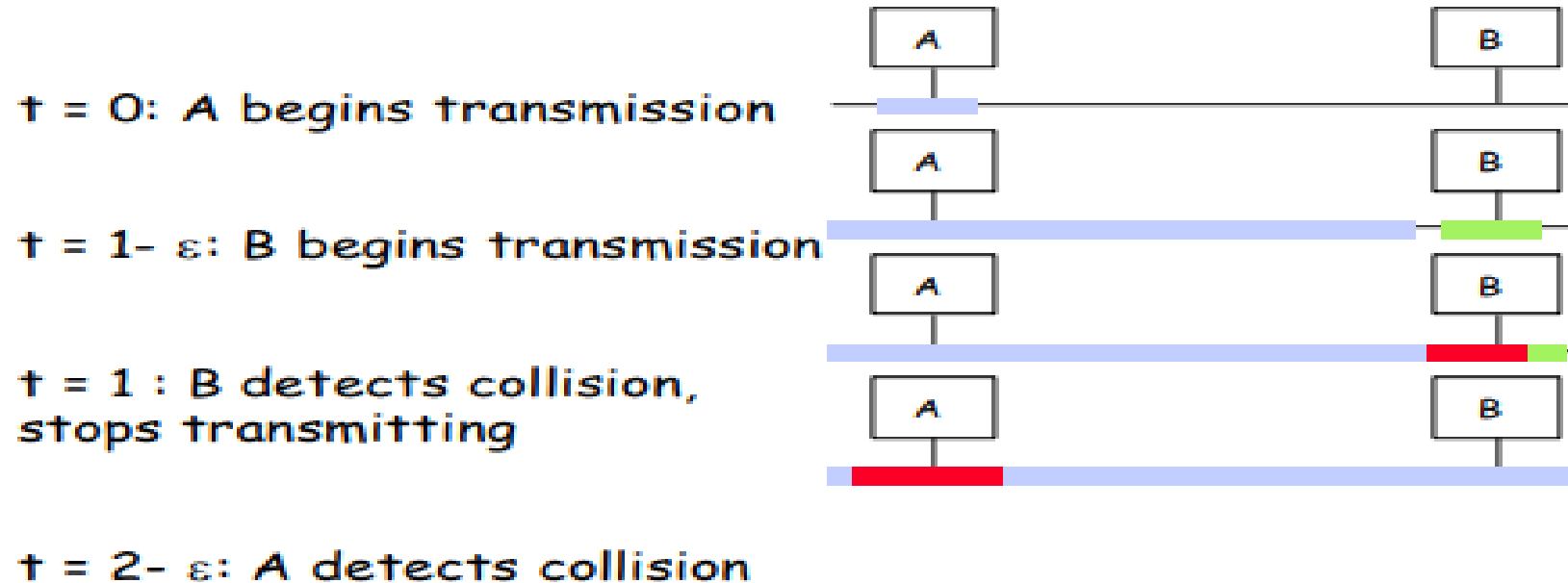
Ethernet CSMA/CD algorithm

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

Minimum Packet Size

- Why enforce a minimum packet size?
- Give a host enough time to detect collisions
- In Ethernet, minimum packet size = 64 bytes (two 6-byte addresses, 2-byte type, 4-byte CRC, and 46 bytes of data)
- If host has less than 46 bytes to send, the adaptor pads (adds) bytes to make it 46 bytes
- What is the relationship between minimum packet size and the length of the LAN?

Minimum Frame Size



- a minimum frame size equal to number of bits transmitted during one round trip is required to detect all collisions.
- Min ethernet frame size is 512 bits i.e., 64 bytes for 10Mbps/100Mbps (51.2us at 10Mbps)
- 512 bytes for 1000Mbps (Why Increased?)
- LAN Cable length = Speed of light (in twisted/coax/fiber) * time to transmit minimum frame size/2
- LAN length = (min_frame_size)*(light_speed)/(2*bandwidth)