Computer Networks

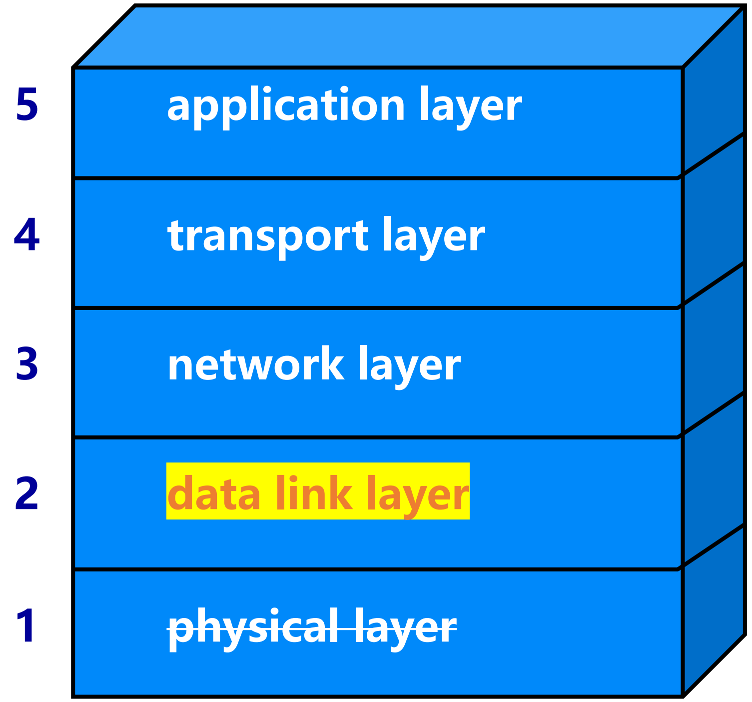
Lecturer: ZHANG Ying

Fall semester 2022

FINAL EXAM

- Time: November 17th, Exam time will be 16:00 – 18:00 (Next Thursday), all students should enter platform at 15:30 for identity check with your student card and passport.
- Platform: Tencent Meeting (Meeting code will be provided 35 min before)
- Closed-book paper-based exam.
- Setting requirements:
  - Camera-based live steaming. The camera should cover NOT ONLY the face, BUT ALSO the desk and the computer screen in a light- sufficient environment.
  - Mic should be OPEN all the time.
  - Ask question ONLY with mic
  - Strictly no other communication
- The detailed policy will be shared for everyone soon and we may have an environment check before exam next week. (Tentative time: 15th Nov, 16:00)

Chapter 6
Data Link Layer



Outline

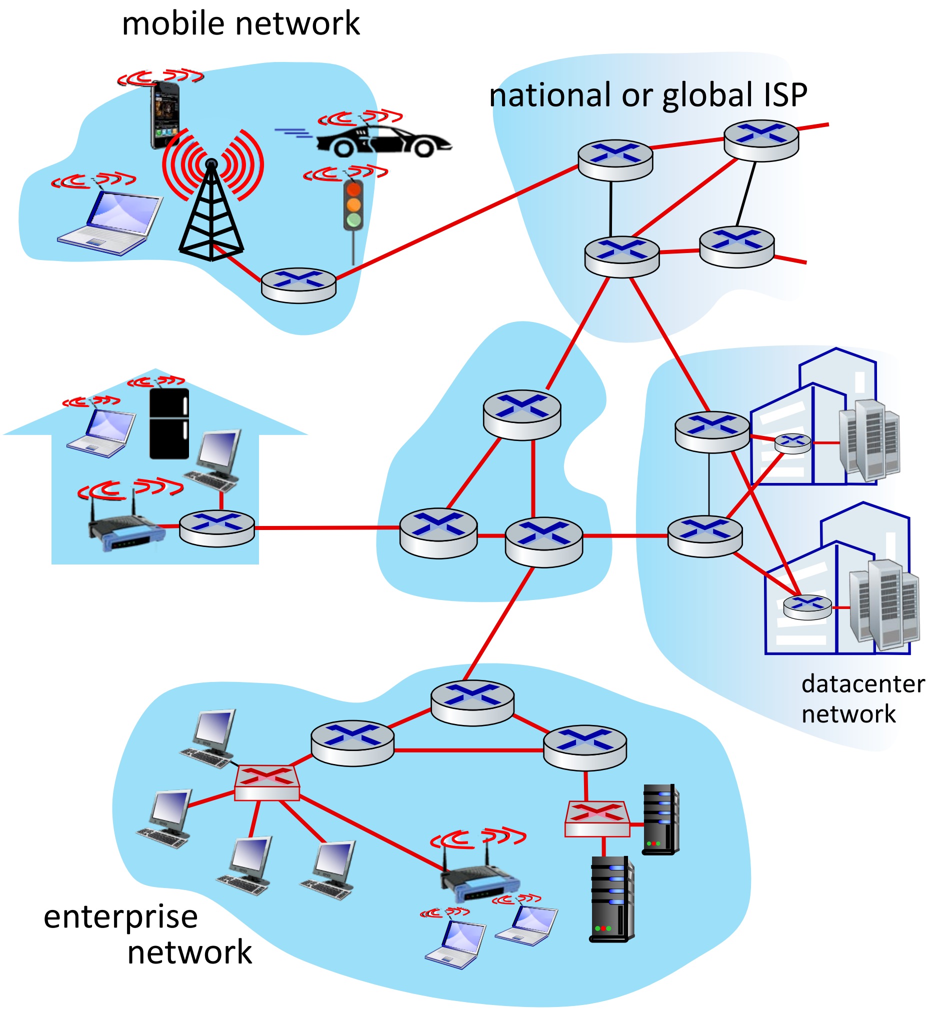
- introduction
- error detection, correction
- multiple access protocols
- LANs

Link layer: introduction

terminology:

- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired
  - wireless
  - LANs
- layer-2 packet: *frame*, encapsulates datagram

link layer has responsibility of transferring datagram from one node to physically adjacent node over a link



Link layer: context

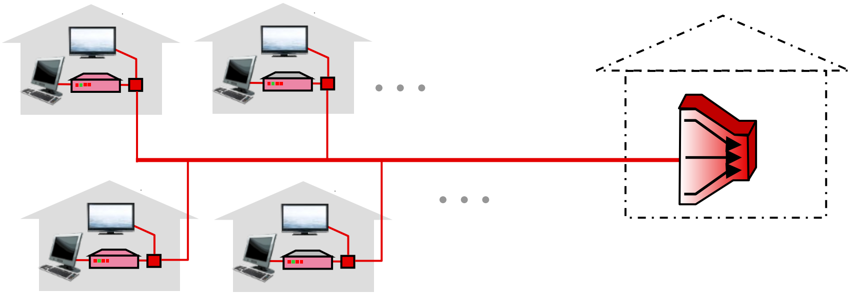
- datagram transferred by different link protocols over different links:
  - e.g., WiFi on first link, Ethernet on next link
- each link protocol provides different services
  - e.g., may or may not provide reliable data transfer over link

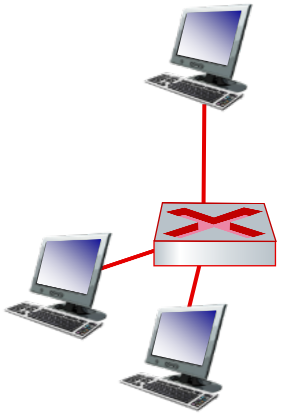
transportation analogy:

- 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

Link layer: services

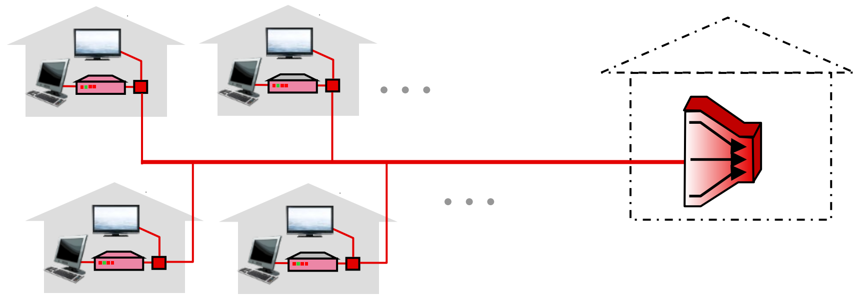
- framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - “MAC” addresses in frame headers identify source, destination (different from IP address!)
- reliable delivery between adjacent nodes
  - we already know how to do this!
  - seldom used on low bit-error links
  - wireless links: high error rates
    - *Q:* why both link-level and end-end reliability?

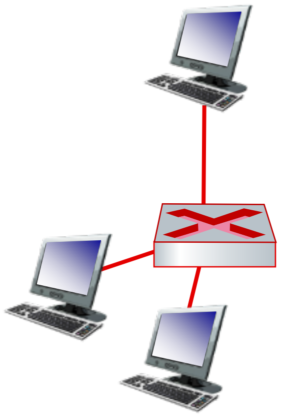




Link layer: services (more)

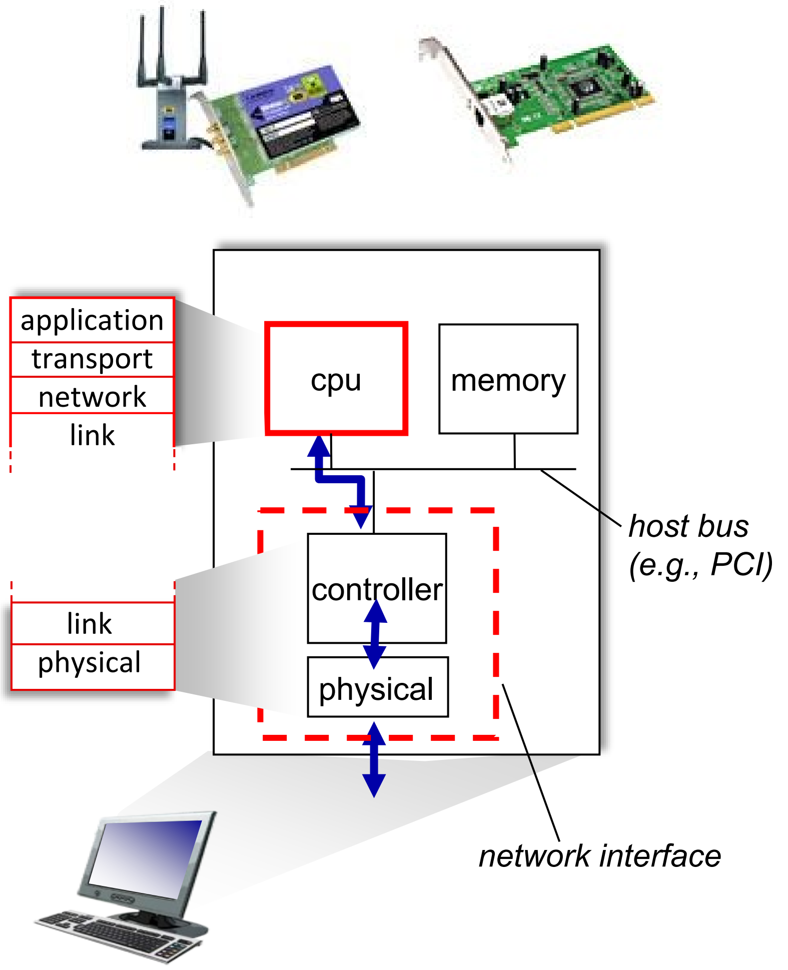
- **flow control:**
  - pacing between adjacent sending and receiving nodes
- **error detection:**
  - errors caused by signal attenuation, noise.
  - receiver detects errors, signals retransmission, or drops frame
- **error correction:**
  - receiver identifies *and corrects* bit error(s) without retransmission
- **half-duplex and full-duplex:**
  - with half duplex, nodes at both ends of link can transmit, but not at same time



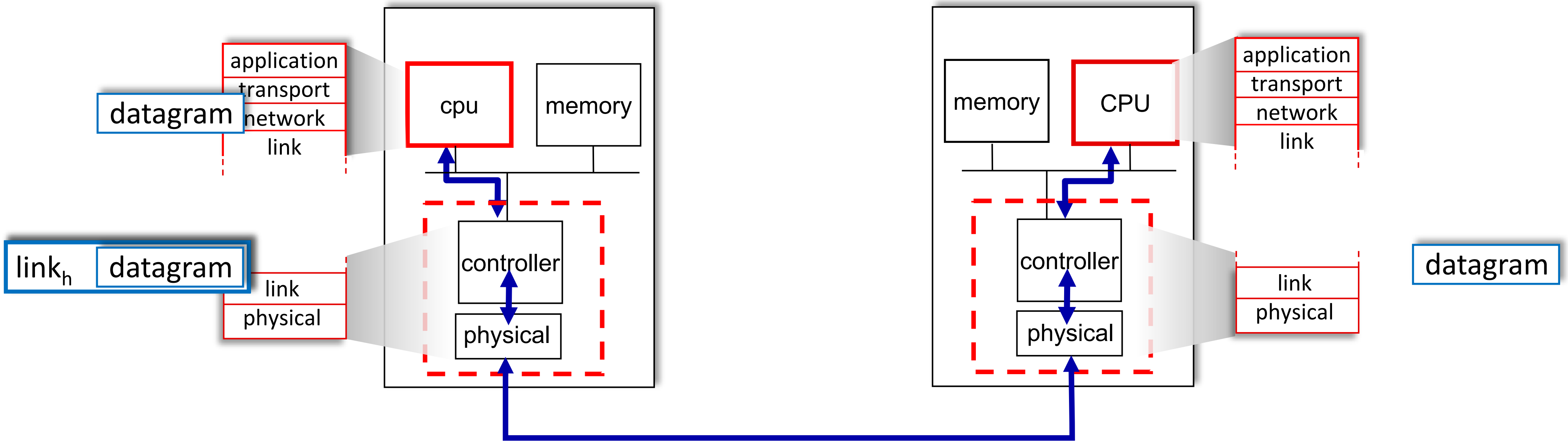


Where is the link layer implemented?

- in each-and-every host
- link layer implemented in *network interface card* (NIC) or on a chip
  - Ethernet, WiFi card or chip
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



Interfaces communicating



sending side:

- encapsulates datagram in frame
- adds error checking bits, reliable data transfer, flow control, etc.

receiving side:

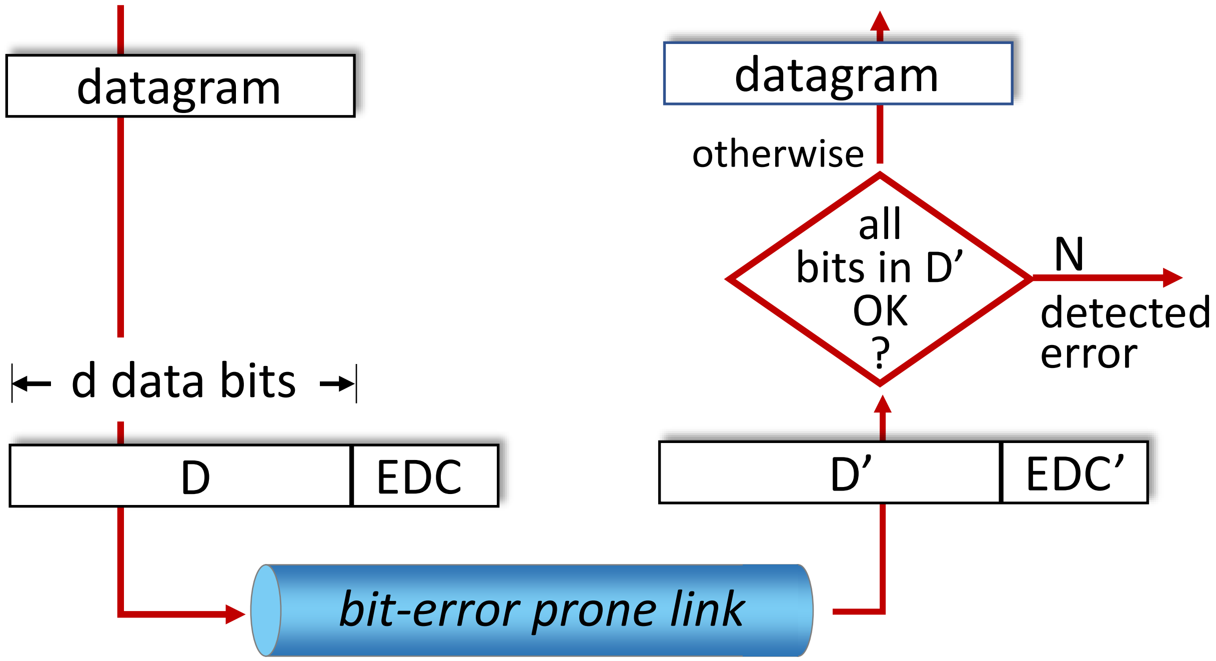
- looks for errors, reliable data transfer, flow control, etc.
- extracts datagram, passes to upper layer at receiving side

outline

- introduction
- error detection, correction
- multiple access protocols
- LANs

Error detection

EDC: error detection and correction bits (e.g., redundancy)
D: data protected by error checking, may include header fields



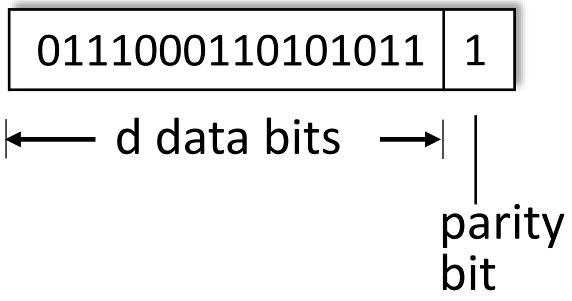
Error detection not 100% reliable!

- 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

Parity checking

two-dimensional bit parity:

- detect *and correct* single bit errors

row parity →

column parity ↓

$$
\begin{array}{cccc|c}
d_{1,1} & \cdots & d_{1,j} & & d_{1,j+1} \\
d_{2,1} & \cdots & d_{2,j} & & d_{2,j+1} \\
\cdots & \cdots & \cdots & & \cdots \\
d_{i,1} & \cdots & d_{i,j} & & d_{i,j+1} \\
\hline
d_{i+1,1} & \cdots & d_{i+1,j} & & d_{i+1,j+1}
\end{array}
$$

no errors:

$$
\begin{array}{ccccc|c}
1 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 0 & 1 & 0
\end{array}
$$

detected and correctable single-bit error:

$$
\begin{array}{ccccc|c}
1 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 0 & 1 & 0
\end{array}
$$

parity error

parity error

Internet checksum (review)

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

sender:

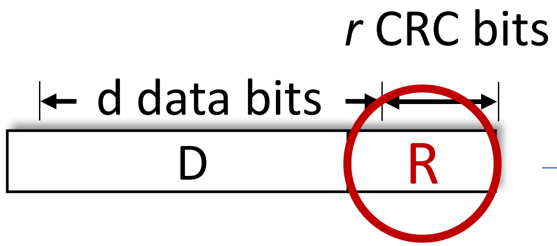
- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

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)



bit pattern

$<D,R> = D*2^r$ XOR R — formula for bit pattern

goal: choose *r* CRC bits, R, such that <D,R> exactly divisible by G (mod 2)

- receiver knows G, 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 (Ethernet, 802.11 WiFi)

Cyclic Redundancy Check (CRC): example

Steps:

1. 1 find the length of the divisor
2. Append L-1 bits to the original message
3. Perform binary division operation.
4. Reminder of the division = CRC

We want:

$$D \cdot 2^r \ XOR \ R = nG$$

or equivalently:

$$D \cdot 2^r = nG \ XOR \ R$$

or equivalently:

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

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

```
   G
                    1 0 1 0 1 1
 1 0 0 1 ) 1 0 1 1 1 0 0 0 0      D*2^r
           1 0 0 1
               1 0 1
               0 0 0
               1 0 1 0
               1 0 0 1
                   1 1 0
                   0 0 0
                   1 1 0 0
                   1 0 0 1
                     1 0 1 0
                     1 0 0 1
                       0 1 1
                         R
```

outline

- introduction
- error detection, correction
- multiple access protocols
- LANs

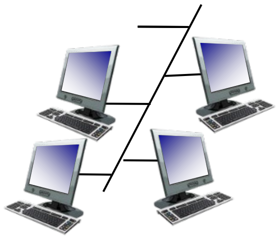
Multiple access links, protocols

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

humans at a cocktail party
(shared air, acoustical)

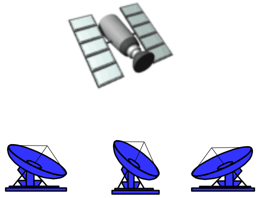
Multiple access links and protocols



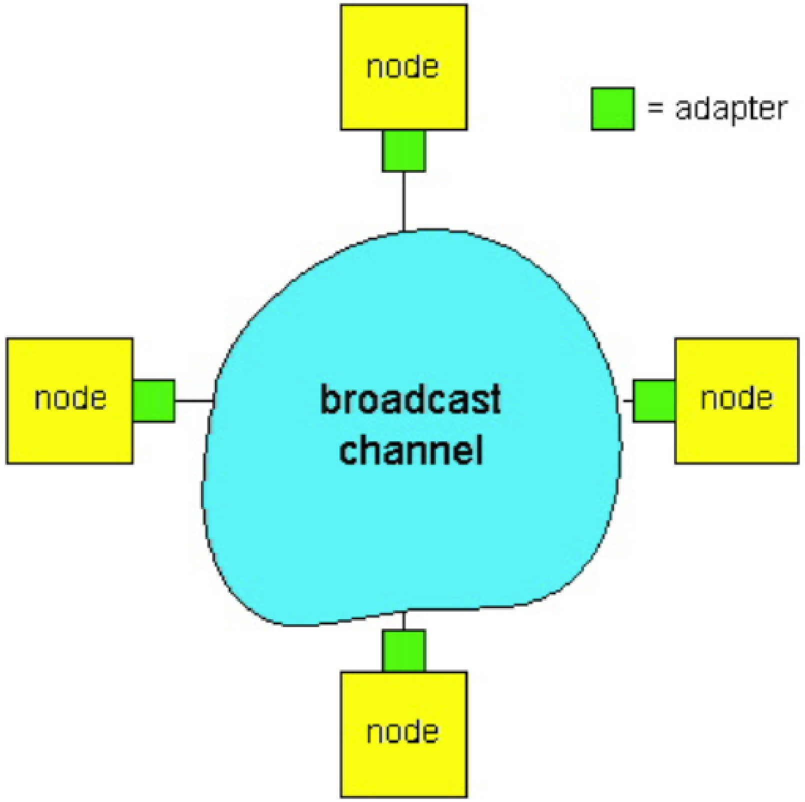
shared wire (e.g., cabled Ethernet)

shared radio: 4G/5G

shared radio: WiFi



shared radio: satellite



A broadcast channel interconnecting four nodes.

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

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

MAC protocols: taxonomy

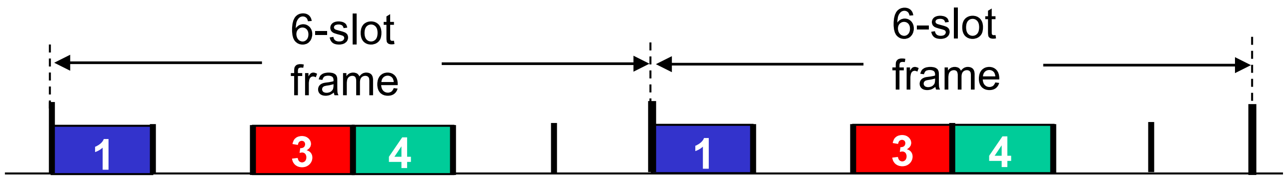
three broad classes:

- 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

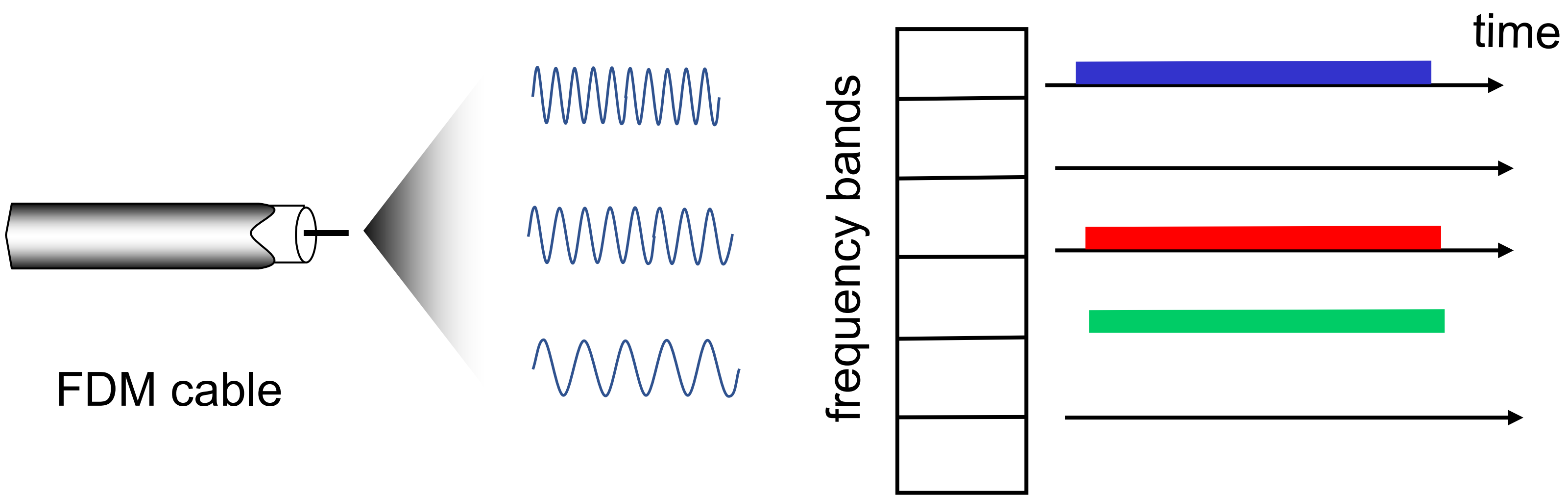
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



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

Slotted ALOHA

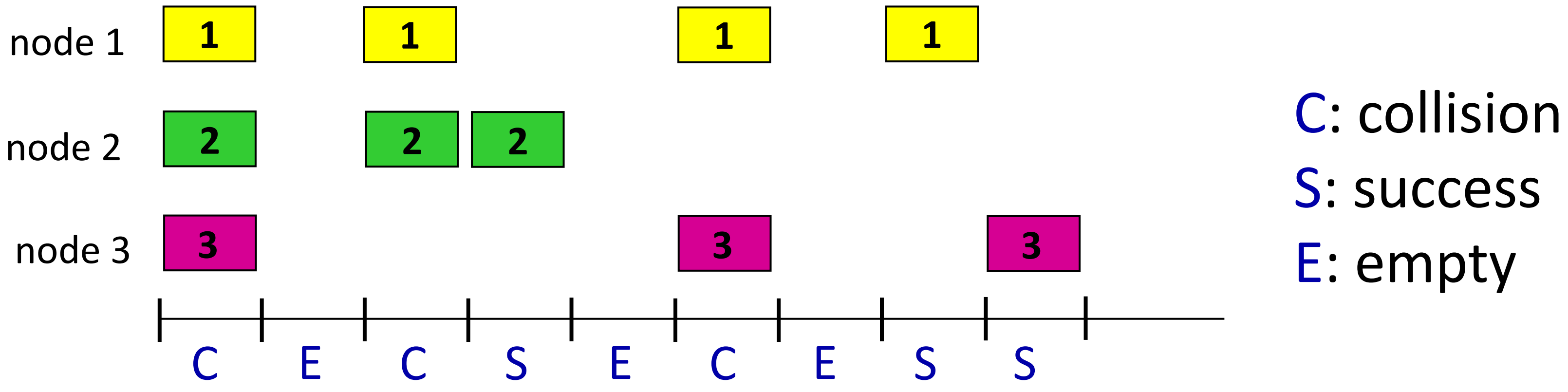
assumptions:

- 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

Slotted ALOHA: efficiency

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

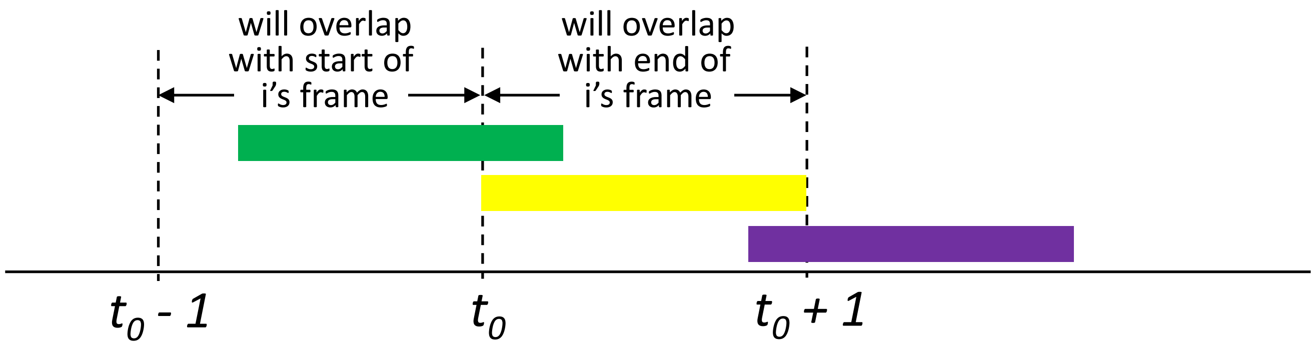
- *suppose: N* nodes with many frames to send, each transmits in slot with probability *p*
  - prob that given node has success in a slot = $p(1-p)^{N-1}$
  - prob that *any* node has a success = $Np(1-p)^{N-1}$
  - max efficiency: find $p*$ that maximizes $Np(1-p)^{N-1}$
  - for many nodes, take limit of $Np*(1-p*)^{N-1}$ as N goes to infinity, gives:

  max efficiency = 1/e = .37

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

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 $[t_0-1, t_0+1]$



- pure Aloha efficiency: 18% !