National University of Computer & Emerging Sciences

CS 3001 – COMPUTER NETWORKS

Lecture 22

Chapter 6

24th April, 2025

Nauman Moazzam Hayat

nauman.moazzam@lhr.nu.edu.pk

Office Hours: 11:30 am till 01:00 pm (Every Tuesday & Thursday)

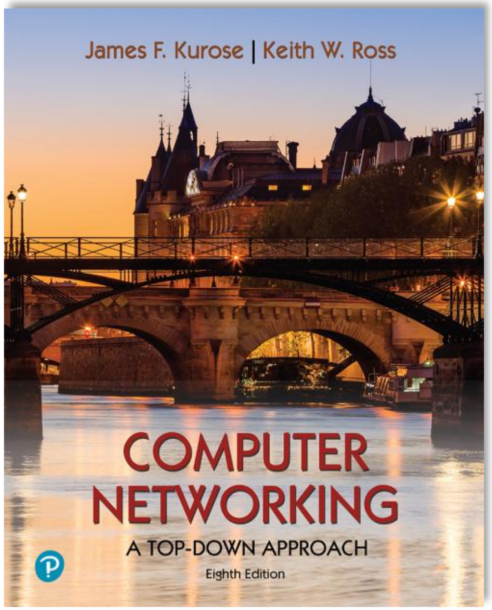
Chapter 6
The Link Layer and LANs

A note on the use of these PowerPoint slides:
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Link layer and LANs: our goals

- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet, VLANs
- datacenter networks

- instantiation, implementation of various link layer technologies

Link layer, LANs: roadmap

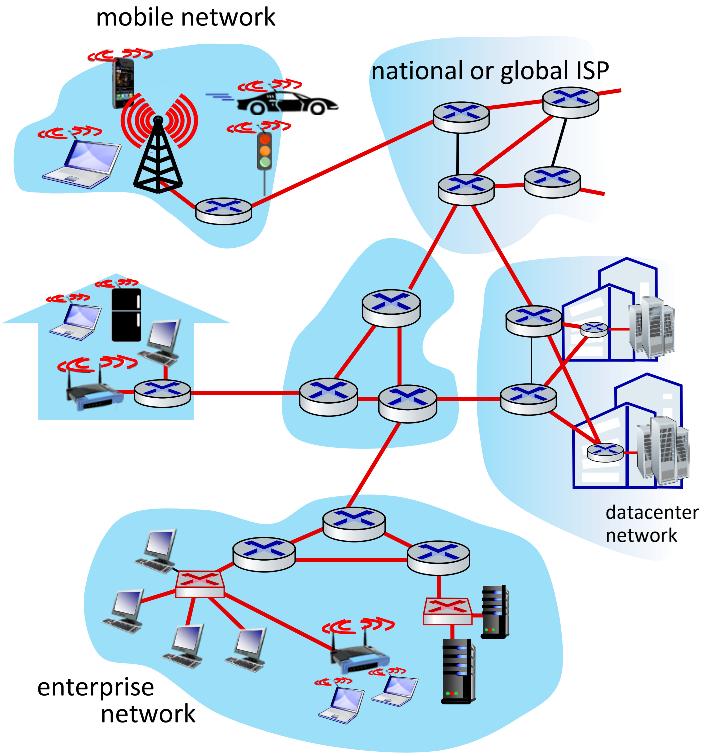
- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - ~~VLANs~~
- ~~link virtualization: MPLS~~
- data center networking

- a day in the life of a web request

Link layer: introduction

terminology:

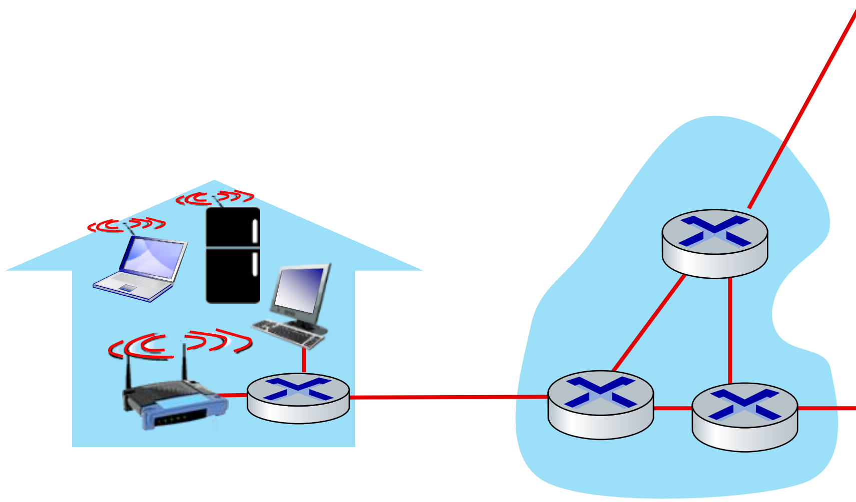
- hosts, routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired , wireless
  - LANs
- layer-2 packet: *frame*, encapsulates datagram



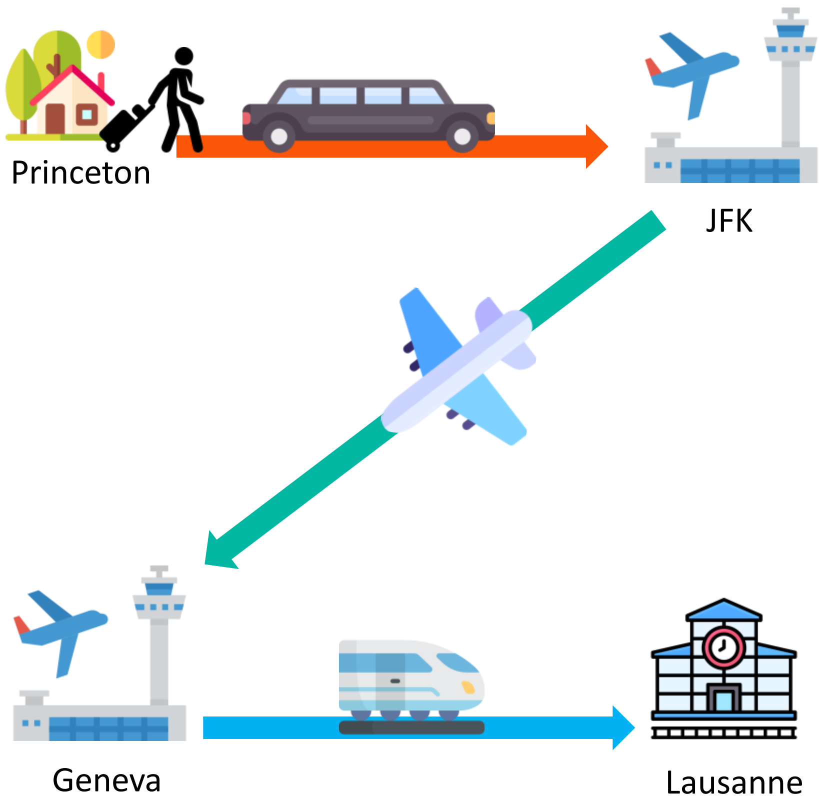
link layer has responsibility of transferring datagram from one node to physically adjacent node over a link

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

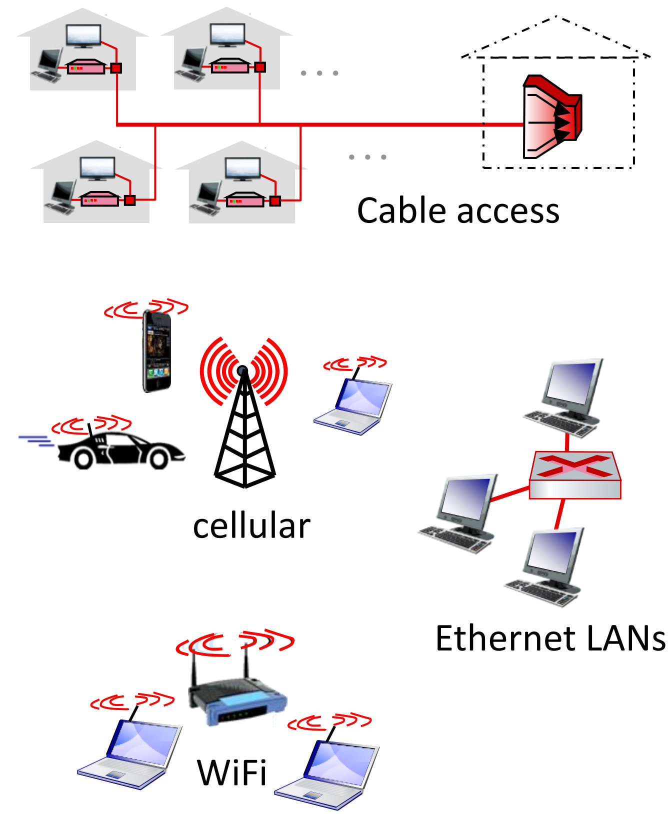


transportation analogy:

- 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

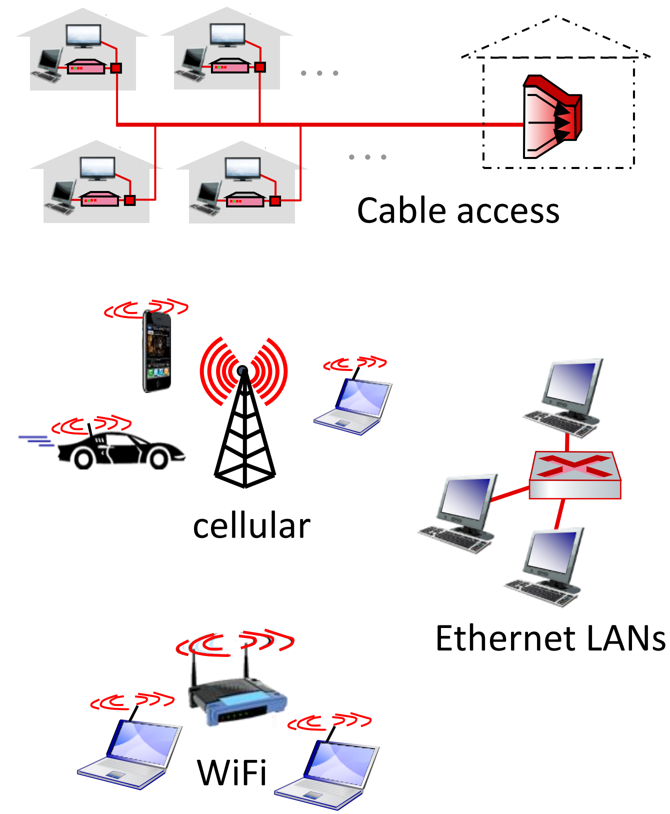
Link layer: services

- **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?**
    - ***A***: Efficiency is improved, i.e. burden taken off of TCP. The goal of correcting an error locally—on the link where the error occurs—rather than forcing an end-to-end retransmission of the data by a transport- or application-layer protocol. Also, even if the link layer is providing reliability, packets could still get lost in intermediate nodes, (e.g. routers etc.)



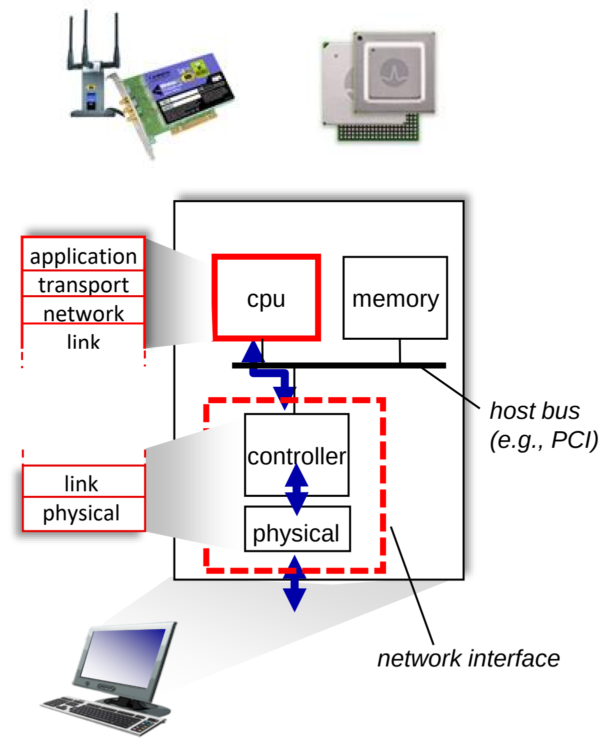
Link layer: services (more)

- 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

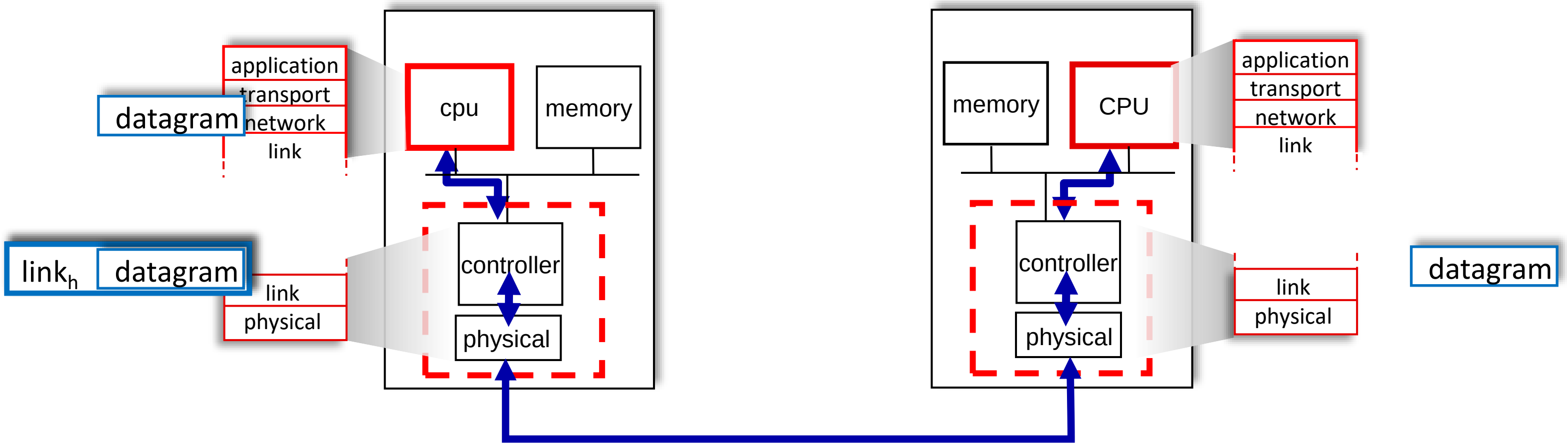


Host link-layer implementation

- in each-and-every host
- link layer implemented on-chip or in network interface card (NIC)
  - 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

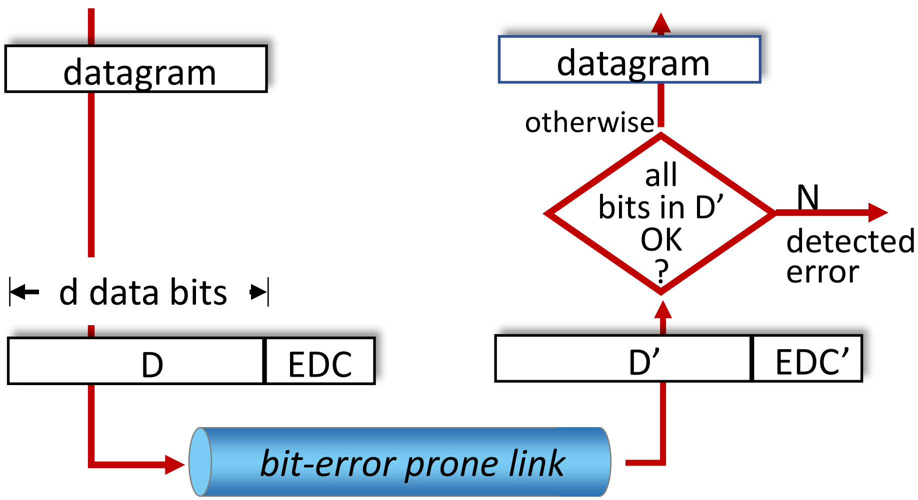
Link layer, LANs: roadmap

- 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

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 (but incur a larger overhead, more computation)

Parity checking

single bit parity:

- detect single bit errors

0111000110101011	1

← *d* data bits → | parity bit

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

At receiver:

- compute parity of *d* received bits
- compare with received parity bit – if different than error detected
- Can only check single bit error & discard if error detected, (not correct the error)

two dimensional parity:

Can detect *and* correct errors (without retransmission!)

- two-dimensional parity: detect *and correct* single bit errors
- For a total of N data bits, You can choose the number of rows (i) and columns (j) such that $i * j \geq N$

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:

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

detected and correctable single-bit error:

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

parity error (row)

parity error (column)

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Internet checksum (review, see section 3.3, primarily performed at the transport layer)

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

sender:

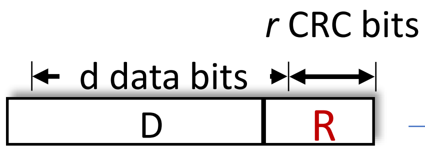
- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

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) (aka Polynomial Codes)

- 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, specified in CRC standard) (the most significant (leftmost) bit of G is required to be a 1; pre agreed between sender & receiver)



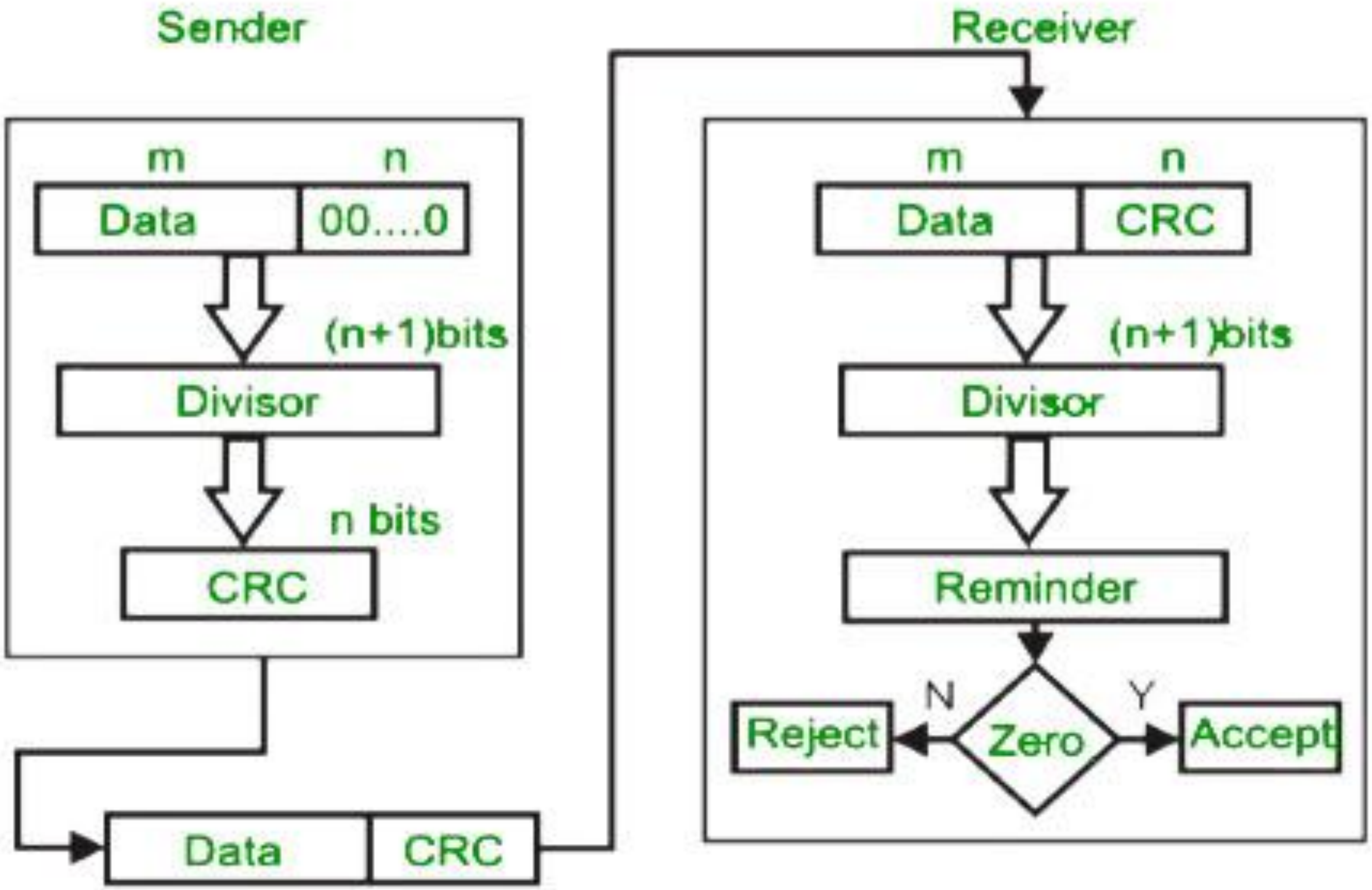
bits to send

$<D,R> = D * 2^r \text{ XOR } R$ formula for these bits

sender: compute *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)

CRC Process Explained



Cyclic Redundancy Check (CRC): example

Sender wants to compute R such that:

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

... or equivalently (XOR R both sides):

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

... which says:

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

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

algorithm for computing R

```
    G
              1 0 1 0 1 1
1 0 0 1 ) 1 0 1 1 1 0 0 0 0     D*2^r  (here, r=3)
          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
```

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Cyclic Redundancy Check (CRC) – Example Video

- For revision of CRC discussed in the Class, please watch and review my video shared via Google Classroom. (Please watch the complete video, where I explain & solve an example of CRC step by step in detail.)

Very Important Example !!!!!!!

Link layer, LANs: roadmap

- 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

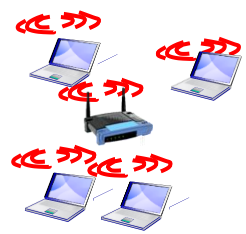
Multiple access links, protocols

two types of "links":

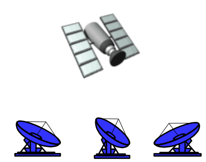
- point-to-point
  - point-to-point link between Ethernet switch, host
  - PPP for dial-up access
- broadcast (shared wire or medium)
  - old-school Ethernet
  - upstream HFC in cable-based access network
  - 802.11 wireless LAN, 4G/4G. satellite

shared wire (e.g., cabled Ethernet)

shared radio: 4G/5G



shared radio: WiFi



shared radio: satellite

humans at a cocktail party (shared air, acoustical)

Multiple access protocols (Solution to Multiple Access Problem)

- 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

desired characteristics of an ideal multiple access protocol:

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

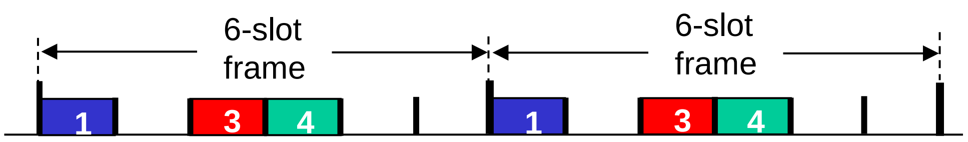
three broad classes:

- 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

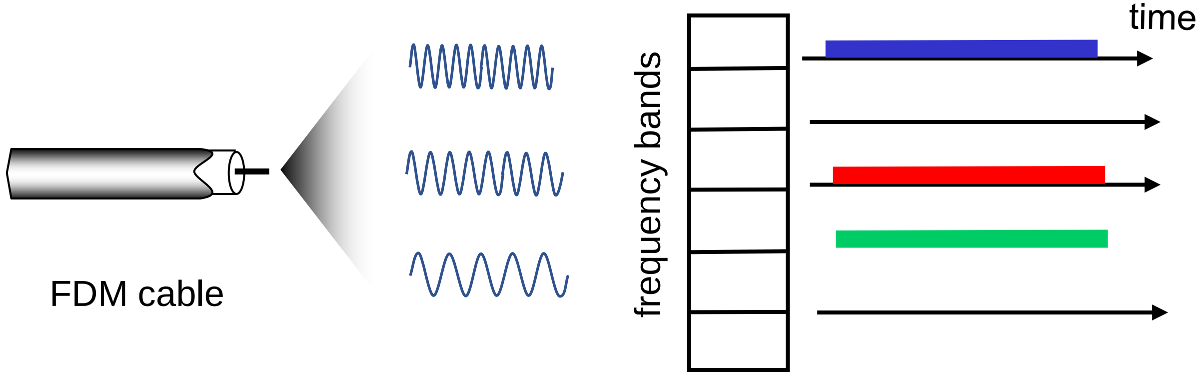
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

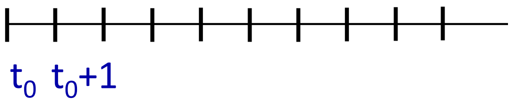
- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have packet to send, frequency bands 2,5,6 idle

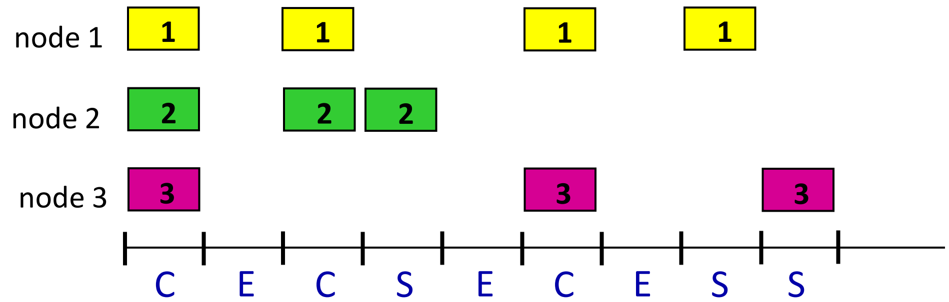


Random access protocols

- 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 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





C: collision
S: success
E: empty

assumptions:

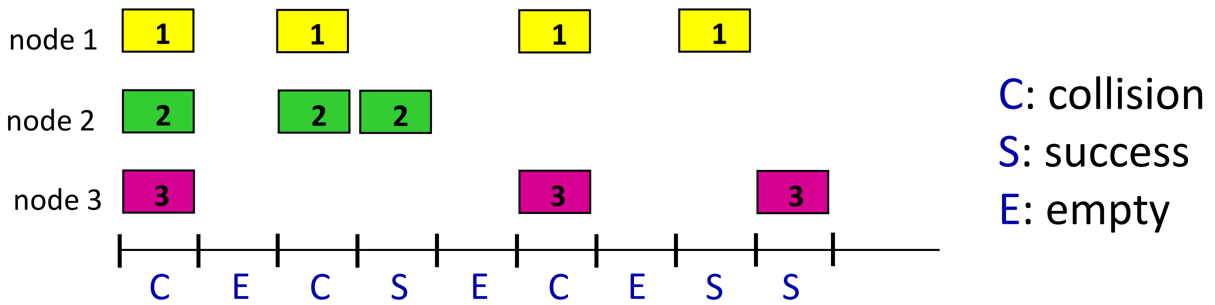
- all frames same size
- time divided into equal size slots (time to transmit 1 frame, i.e. if frame size is L bits, transmission rate is R in bits per second, then time required to transmit one frame = the size of one time slot = L / R seconds)
- nodes start to transmit only at slot beginning
- nodes are synchronized (so that each node knows when the slots begin)
- if 2 or more nodes transmit in the same time 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 (i.e. each node knows when the slots begin so the transmission starts at the beginning of the slot)
- very 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 *(See the below explanation. For further details, review the textbook page 466 till 468.)*

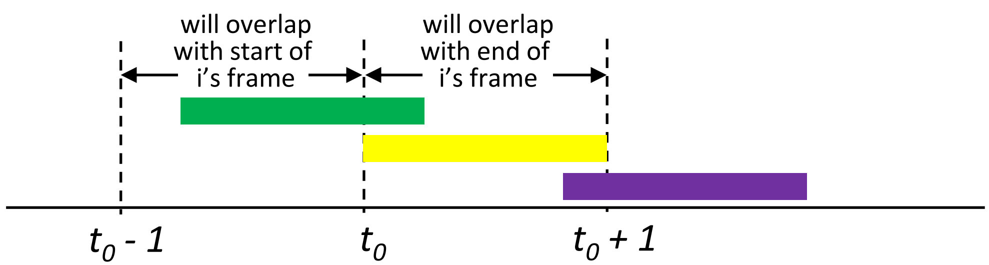
- *at best:* channel used for useful transmissions 37% of time!

(**Explanation**: Suppose there are N nodes. Then the probability that a given slot is a successful slot is the probability that one of the nodes transmits and that the remaining N - 1 nodes do not transmit. The probability that a given node transmits is p; the probability that the remaining nodes do not transmit is $(1 - p)^{N-1}$. Therefore, the probability a given node has a success is $p(1 - p)^{N-1}$. Because there are N nodes, the probability that any one of the N nodes has a success is $Np(1 - p)^{N-1}$.

Only 37 percent of the slots do useful work. Thus, the effective transmission rate of the channel is not R bps but only 0.37 R bps! A similar analysis also shows that 37 percent of the slots go empty and 26 percent of slots have collisions.)

Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization (fully de-centralized)
  - when frame first arrives: transmit immediately (in it's entirety into the broadcast channel)
- 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% ! *(for details, review the textbook page 468.)*

even *worse* than slotted Aloha!

(The price to be paid for a fully decentralized ALOHA protocol)

Pure ALOHA efficiency

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

P(no other node transmits in $[t_0-1,t_0]$ $\cdot$

P(no other node transmits in $[t_0-1,t_0]$

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

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

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

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

even worse than slotted Aloha!

CSMA (carrier sense multiple access)

simple CSMA: listen before transmit:

- if channel sensed idle: transmit entire frame
- if channel sensed busy: defer transmission

- human analogy: <u>don't interrupt others!</u>

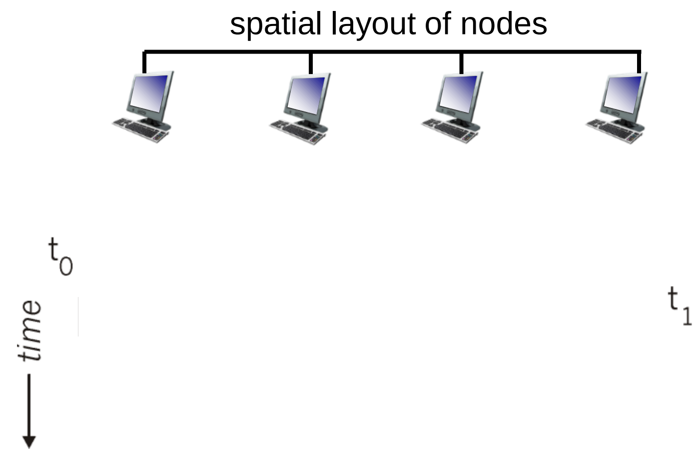
CSMA/CD: CSMA with *collision detection*

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage
- collision detection easy in wired, difficult with wireless

- human analogy: <u>the polite conversationalist</u>

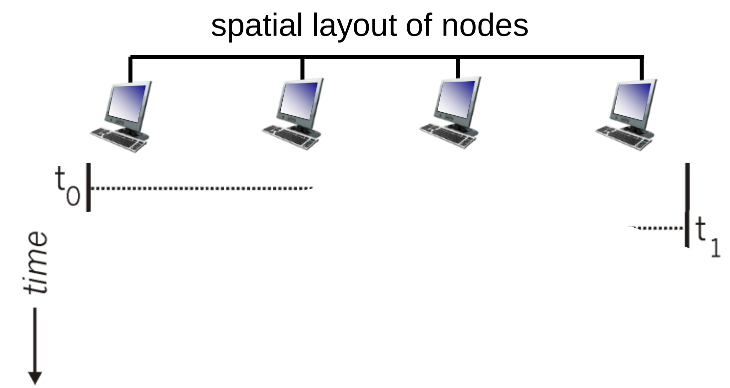
CSMA: collisions

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

CSMA/CD efficiency

- 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 (since colliding nodes will abort immediately without wasting the channel)
  - as t_{trans} goes to infinity (because when a frame grabs the channel, it will hold on to the channel for a very long time; thus, the channel will be doing productive work most of the time)
- better performance than ALOHA: and simple, cheap, decentralized!

Assignment # 5 (Chapter – 5) (Already announced)

- *5th Assignment will be uploaded on Google Classroom on Tuesday, 22nd April, 2025, in the Stream - Announcement Section*

- *Due Date: Tuesday, 29th April, 2025 (Handwritten solutions to be submitted during the lecture)*

- *Please read all the instructions carefully in the uploaded Assignment document, follow & submit accordingly*

Quiz # 5 (Chapter – 5) (Already announced)

- *On: Tuesday, 29th April, 2025 (During the lecture)*

- *Quiz to be taken during <u>own section class only</u>*