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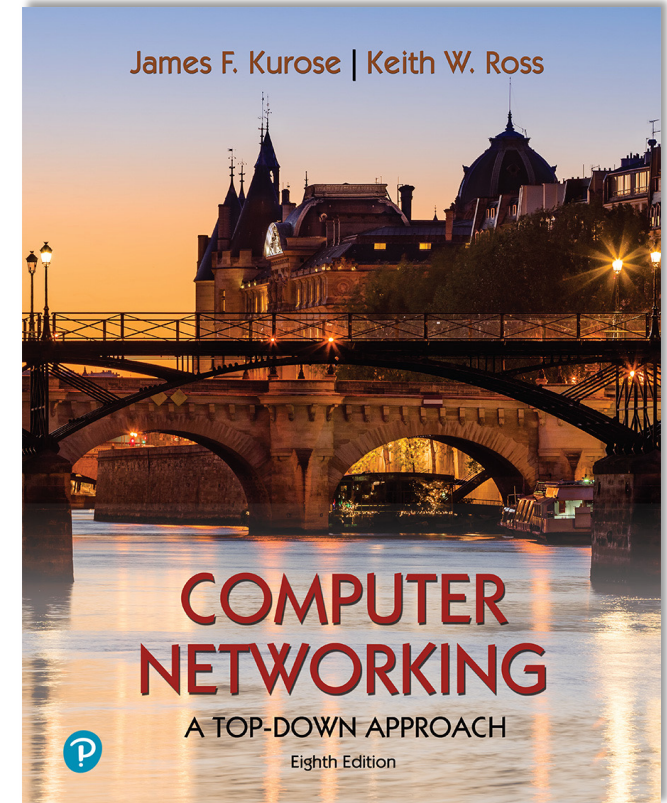
- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

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

Objective

Build a *system* to transfer data from one node to a physically connected node

Let's turn back time …

1. Addressing
2. Error detection
3. Medium access control

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
    - WiFi

Link layer: roadmap

- **introduction**
- addressing
- error detection, correction
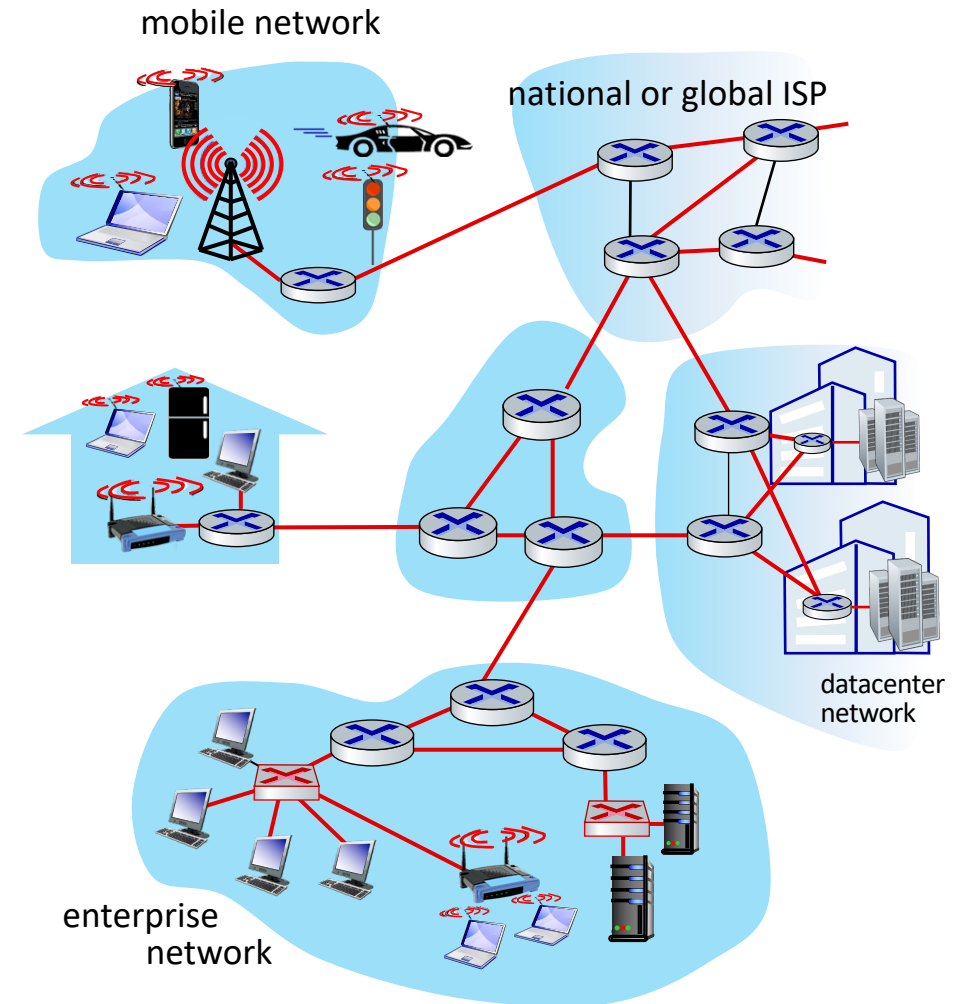- multiple access protocols
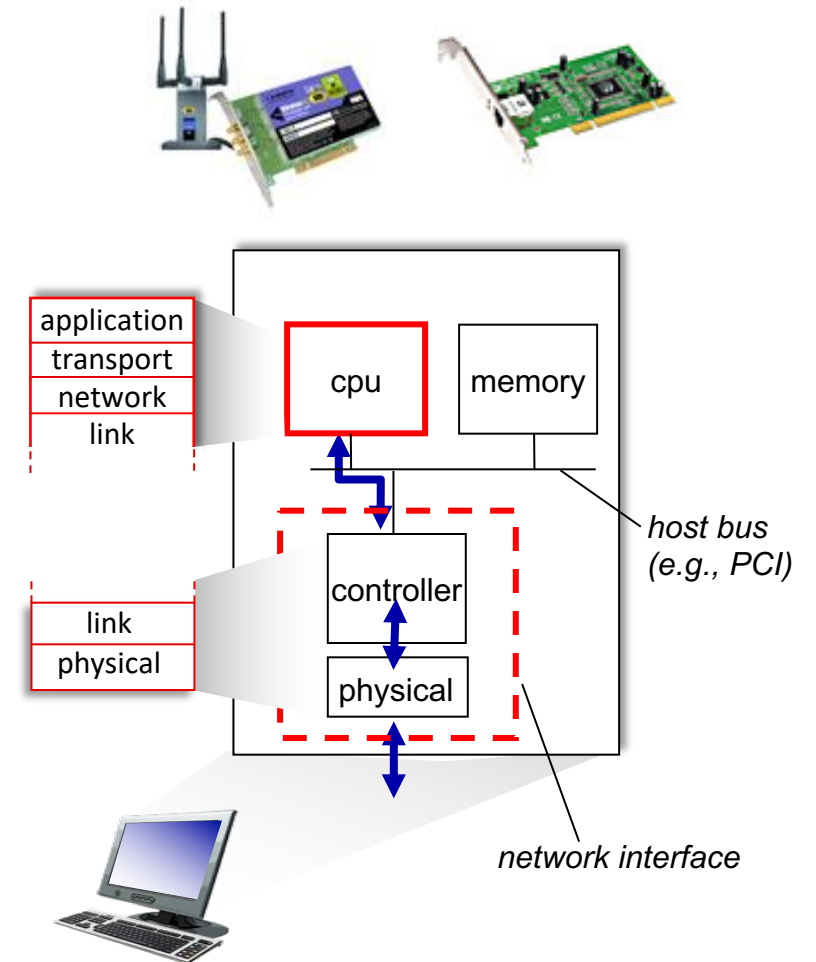
Link layer: introduction

terminology:

- communication channels that connect adjacent nodes along communication path: links
  - wired
  - wireless
- layer-2 packet: *frame*
- LAN: local area network

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

mobile network

national or global ISP

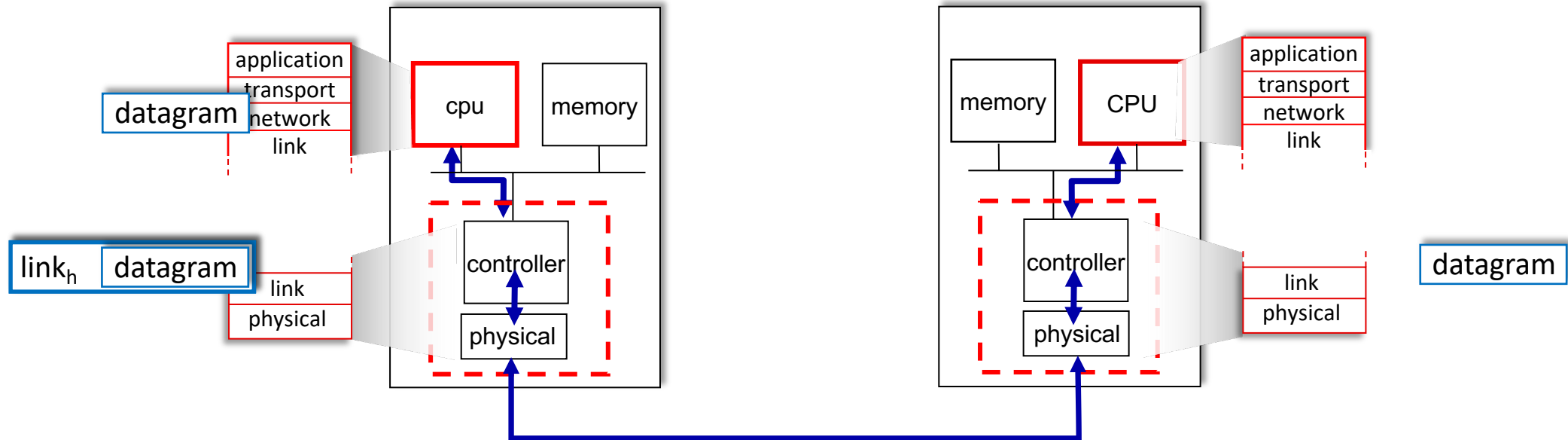datacenter network

enterprise network

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 and physical layer
- attaches into host's system buses
- combination of hardware, software, firmware

application
transport
network
link

link
physical

cpu

memory

host bus (e.g., PCI)

controller

physical

network interface

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: roadmap

- introduction
- **addressing**
- error detection, correction
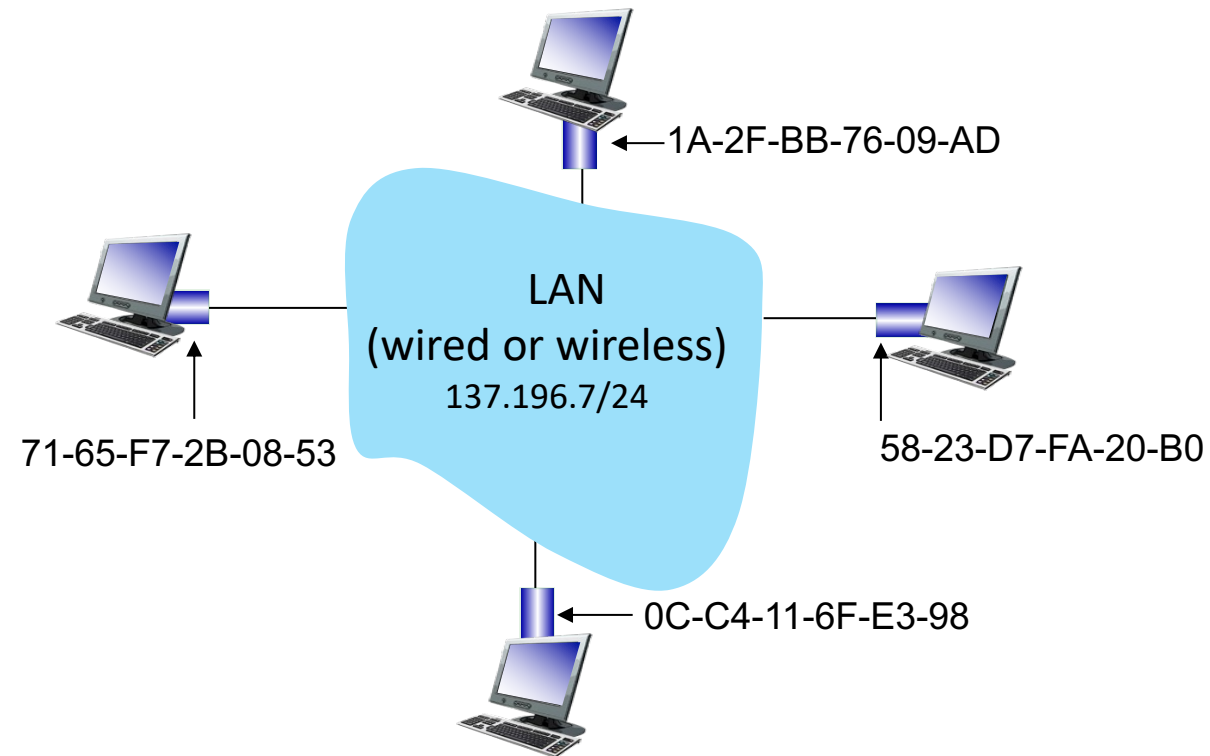- multiple access protocols

MAC addresses

- MAC (or LAN or physical or Ethernet) address:
  - function: used "locally" to get frame from one interface to another physically-connected interface
  - 48-bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
  - e.g.: 1A-2F-BB-76-09-AD

    hexadecimal (base 16) notation (each "numeral" represents 4 bits)

- MAC address allocation administered by IEEE
  - manufacturer buys portion of MAC address space (to assure uniqueness)
- MAC flat address: portability
  - can move interface from one LAN to another

MAC addresses

each interface on LAN
- has unique 48-bit MAC address

1A-2F-BB-76-09-AD

LAN
(wired or wireless)
137.196.7/24

71-65-F7-2B-08-53

58-23-D7-FA-20-B0

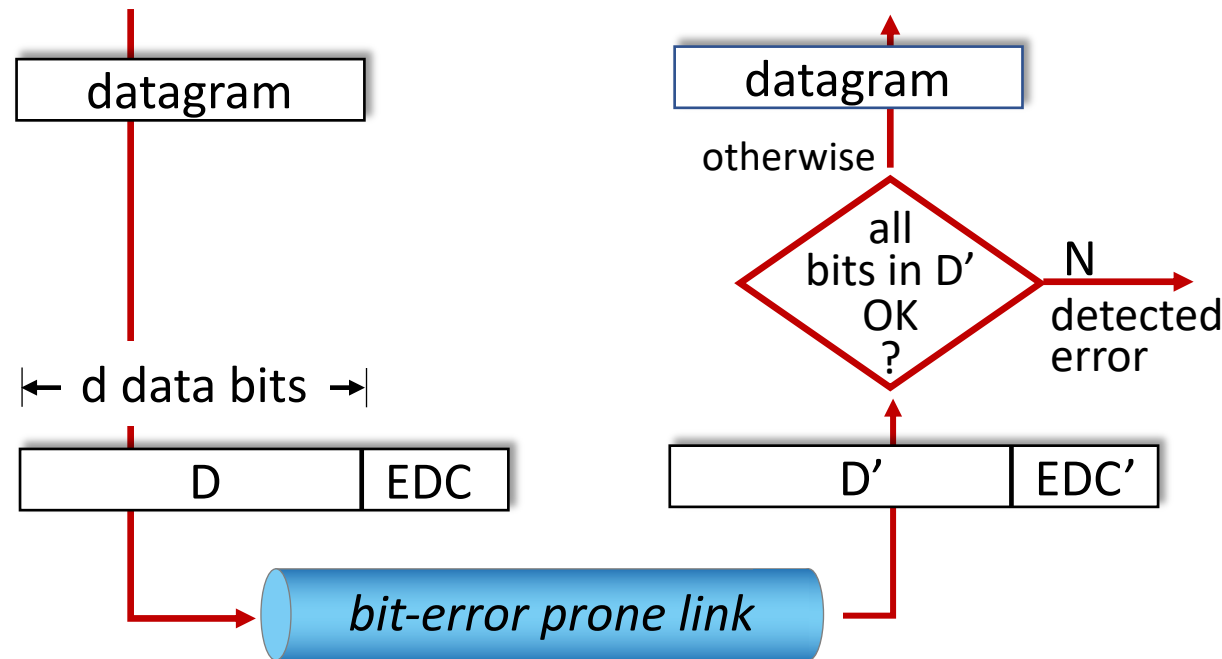0C-C4-11-6F-E3-98

Link layer: roadmap

- introduction
- addressing
- **error detection, correction**
- multiple access protocols

Error detection

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

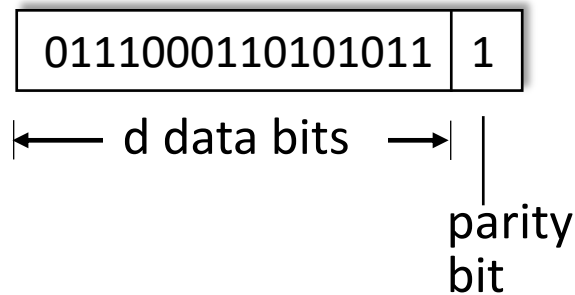D: data protected by error checking, may include header fields

Error detection not 100% reliable!

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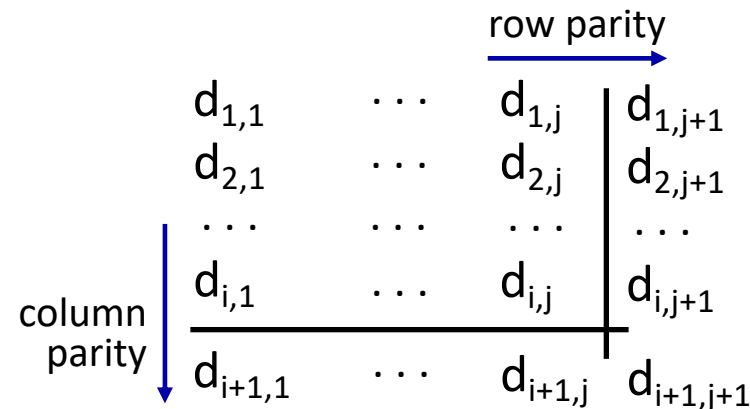
Parity checking

single bit parity:
- detect single bit errors

$$01110001101010110 \mid 1$$

$\longleftarrow$ d data bits $\longrightarrow$

parity bit

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

two-dimensional bit parity:
- detect *and correct* single bit errors

row parity $\longrightarrow$

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

column parity $\downarrow$

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) $\longrightarrow$

parity error (column)

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

Internet checksum

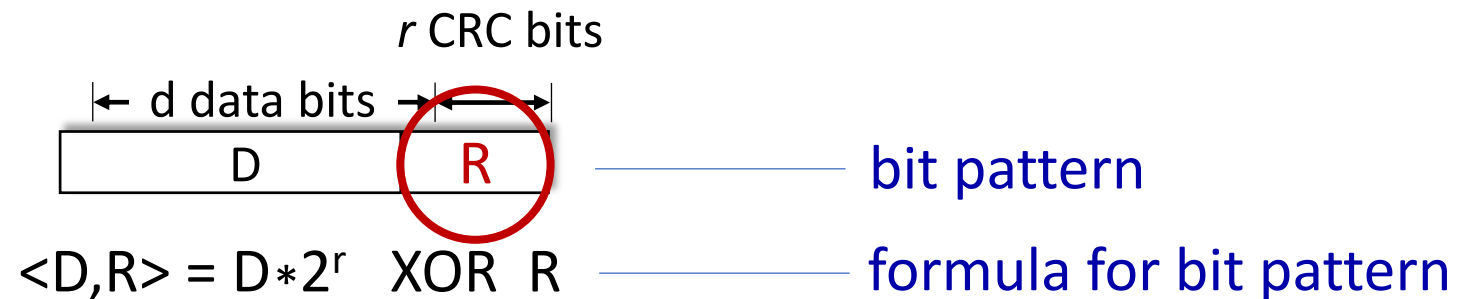*Goal:* detect errors (*i.e.,* flipped bits) in transmitted segment

sender:

- treat contents as sequence of 16-bit integers
- checksum: addition (one's complement sum) of pdu content
- checksum value put into checksum field

receiver:

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

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)

r CRC bits

← d data bits →|← →|

| D | R |

bit pattern

$$\langle D,R \rangle = D * 2^r \ \text{XOR} \ R$$ —————— 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
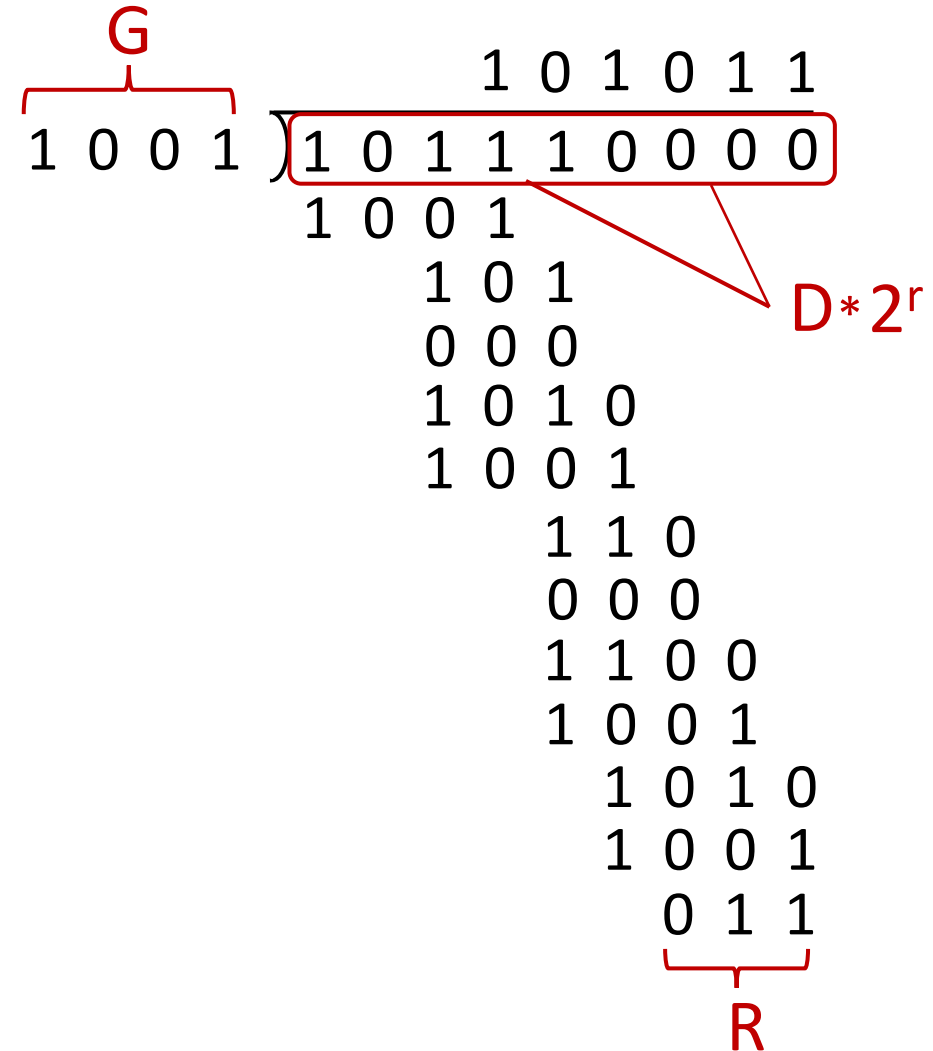
We want:

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

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:

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

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

Link layer: roadmap

- introduction
- addressing
- error detection, correction
- **multiple access protocols**

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
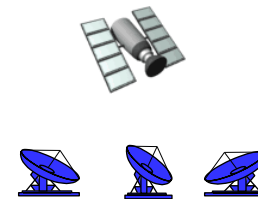 - 802.11 wireless LAN, 4G/4G. satellite

shared wire (e.g., cabled Ethernet)

shared radio: 4G/5G

shared radio: WiFi

shared radio: satellite

humans at a cocktail party (shared air, acoustical)

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

- **"taking turns"**
  - nodes take turns, but nodes with more to send can take longer turns
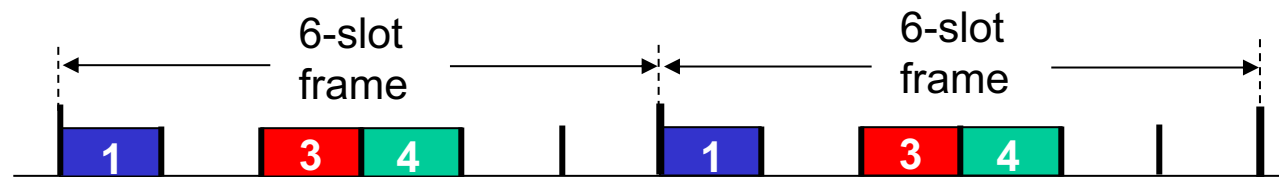
- **random access**
  - channel not divided, allow collisions
  - "recover" from collisions

Channel partitioning MAC protocols: TDMA
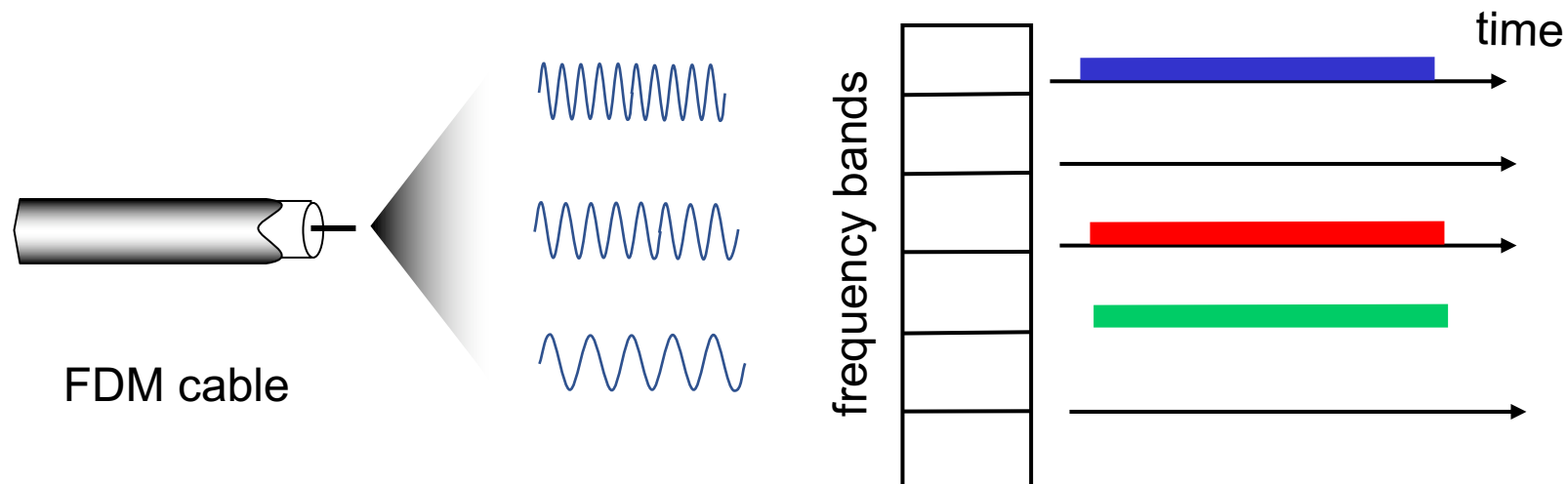
TDMA: time division multiple access

- 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

FDM cable

frequency bands

time

"Taking turns" MAC protocols

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!
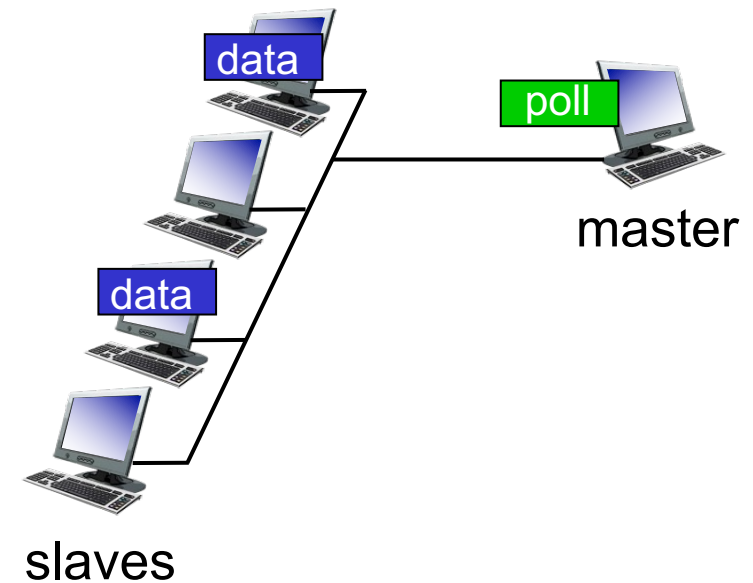
"taking turns" protocols

- look for best of both worlds!

random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead
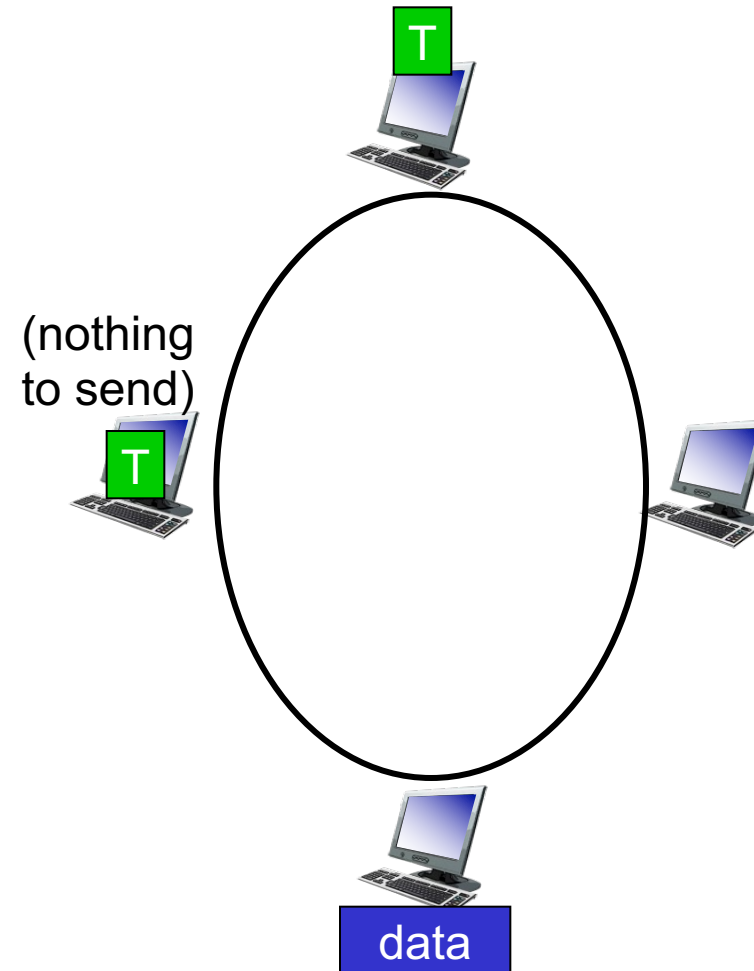
"Taking turns" MAC protocols

polling:

- master node "invites" other nodes to transmit in turn

- typically used with "dumb" devices

- concerns:

  - polling overhead

  - latency

  - single point of failure (master)

data

data

poll

master

slaves

"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)

"Taking turns" MAC protocols

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!

"taking turns" protocols
- look for best of both worlds!

random access MAC protocols
- efficient at low load: single node can fully utilize channel
- high load: collision overhead

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
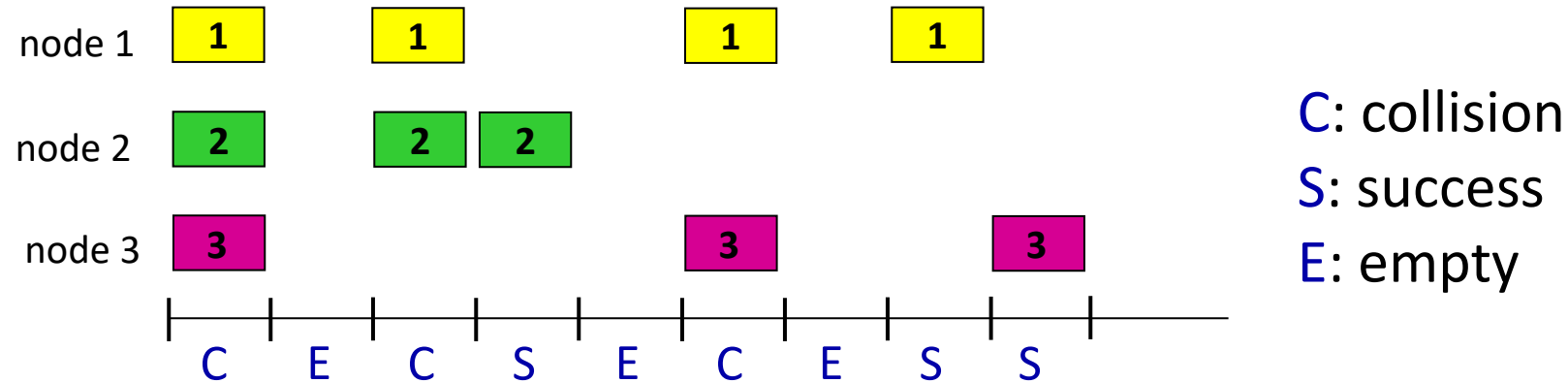
Slotted ALOHA

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

randomization – *why*?

Slotted ALOHA

node 1 | 1 | | 1 | | 1 | | 1 |

node 2 | 2 | | 2 | 2 |

node 3 | 3 | | 3 | | 3 |

C   E   C   S   E   C   E   S   S

C: collision
S: success
E: empty

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
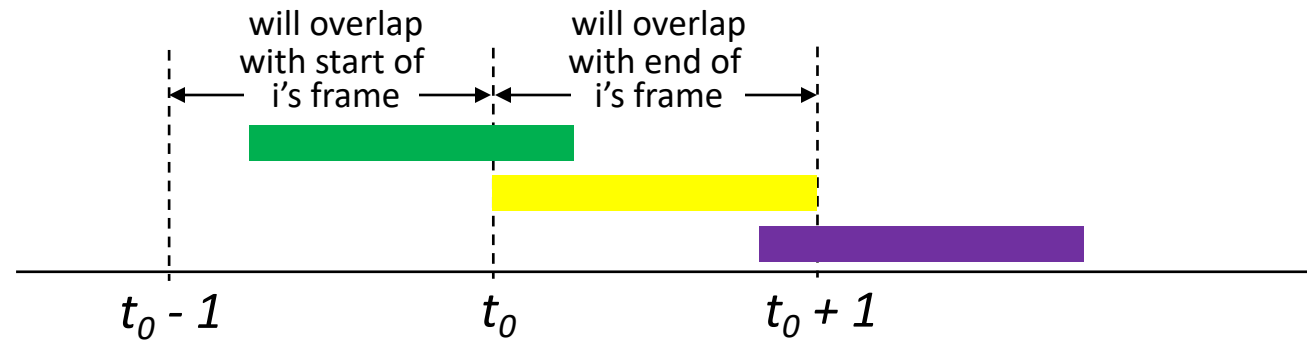- clock synchronization

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 $[t_0-1, t_0+1]$

will overlap with start of i's frame

will overlap with end of i's frame

$t_0 - 1$ t_0 $t_0 + 1$

- pure Aloha efficiency: 18% !

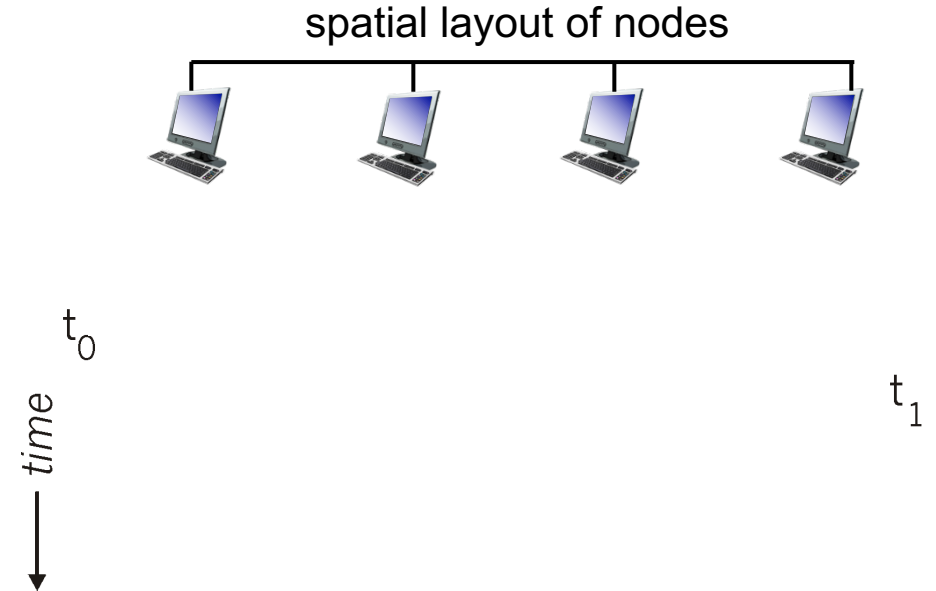# CSMA (carrier sense multiple access)

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
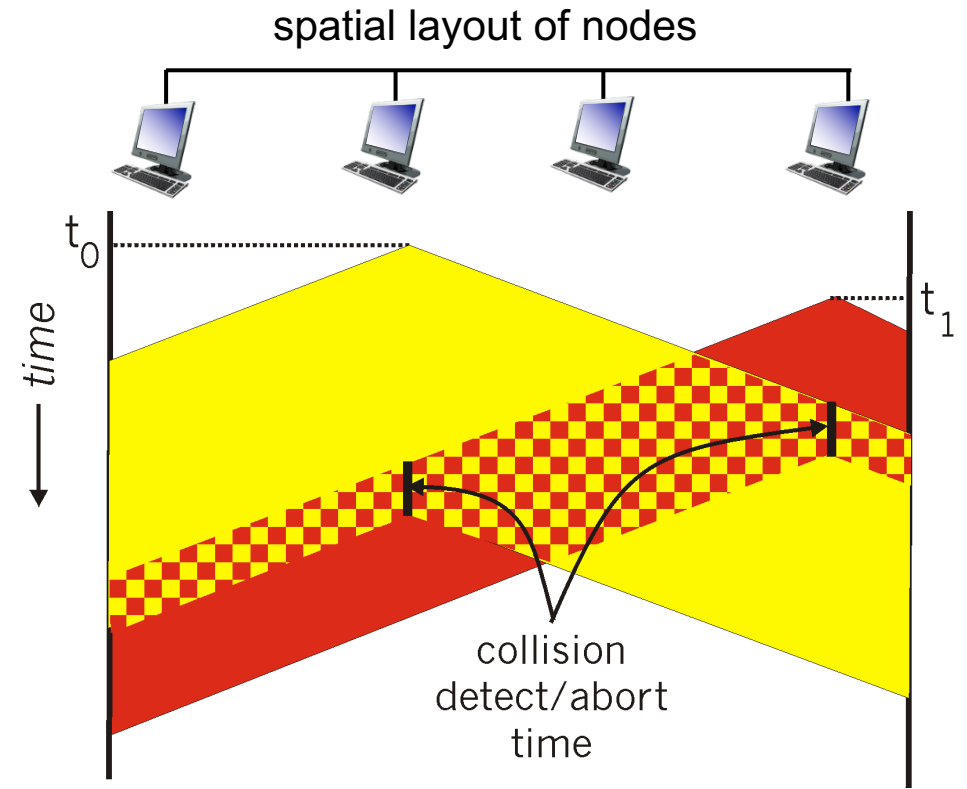
▪ human analogy: the polite conversationalist

CSMA: collisions

spatial layout of nodes

- **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

t_0

time

t_1

CSMA/CD:

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

spatial layout of nodes

collision detect/abort time

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 mth collision, NIC chooses K at random from *{0,1,2, …, 2^m-1}*. NIC 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
  - as t_{trans} goes to infinity
- better performance than ALOHA: and simple, cheap, decentralized!

Summary of MAC protocols

- **channel partitioning,** by time, frequency or code
  - Time Division, Frequency Division
- **taking turns**
  - polling from central site, token passing
  - Bluetooth, FDDI, token ring
- **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
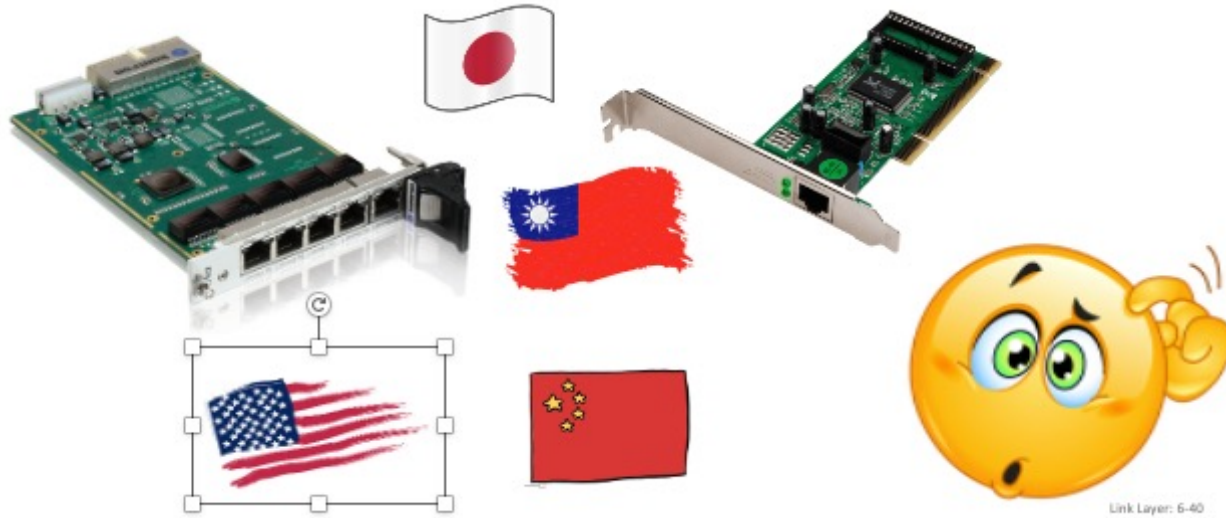 - CSMA/CA used in 802.11

Link Layer I: Summary

- principles behind data link layer services:
    - link layer addressing
    - error detection, correction
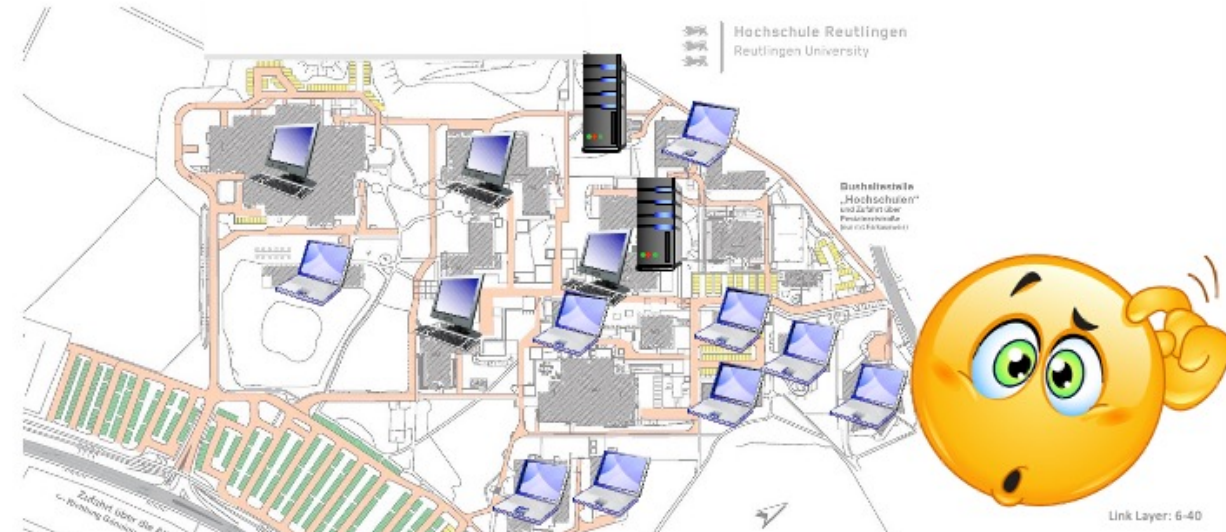   - sharing a broadcast channel: multiple access

More problems and solution ideas

- What problems remain?

- What cannot be done with the shown building bricks?
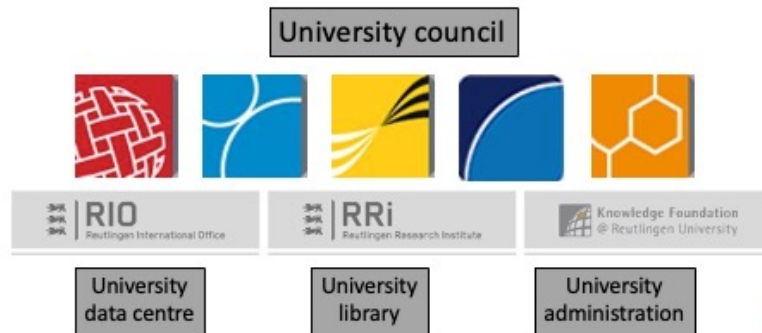 - Discussion in small groups (10 minutes)

Real world application – interoperability
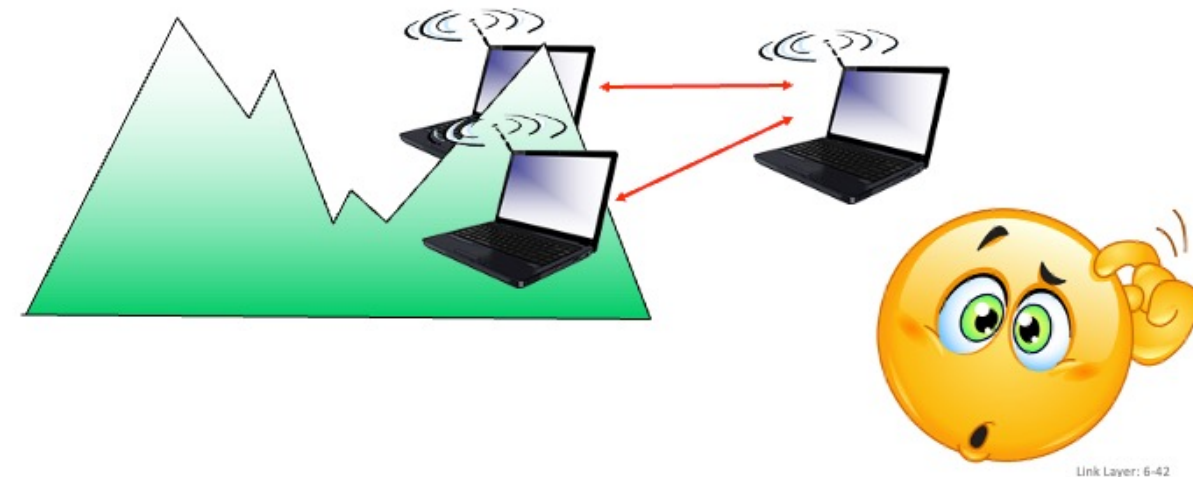

Real world application – campus network

Real world application – multiple stakeholders

Real world application – wired vs. wireless

Topics for next lecture

1. Protocol specification
2. Building large local networks
3. Segregation of stakeholders
4. CSMA in wireless networks