Networks: Tutorial 11

Shinnazar Seytnazarov, PhD

Innopolis University

s.seytnazarov@innopolis.ru

Febuary 21, 2022

Topic of the lecture

- Introduction and Link-layer services
- Error detection and correction
- Multiple Access Protocols
- Switched LANs
- Data Center Networking

Topic of the tutorial

- Error detection
  - Checksum
  - Cyclic redundancy check

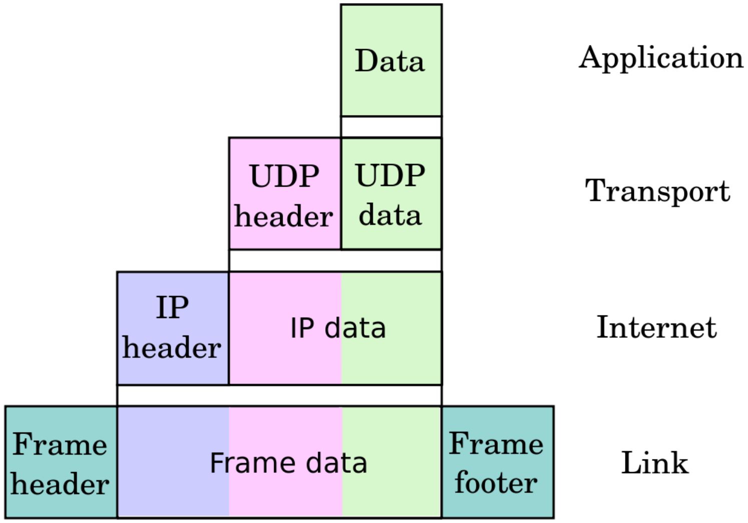
- Multiple Access Protocols
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

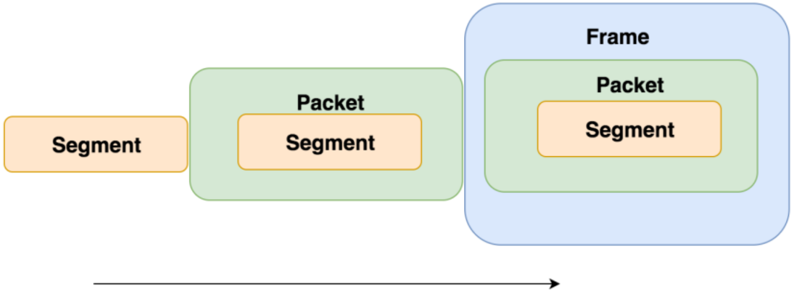
Topic of the lab

- Continuation of the lab practice

Segments, Packets and Frames

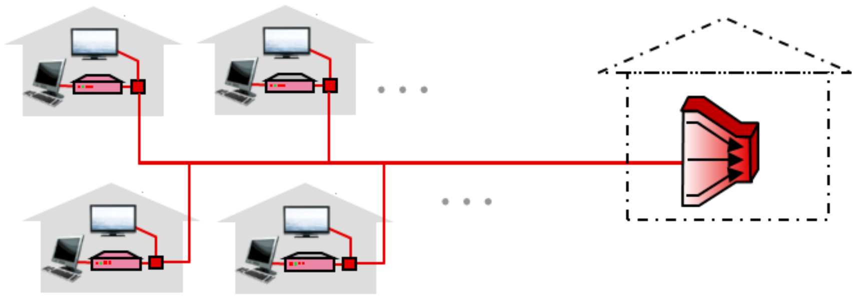
- **Segments** are units of data in the *Transport Layer*
- **Packets** are units of data in the *Network Layer*
- **Frames** are units of data in the *Link Layer*

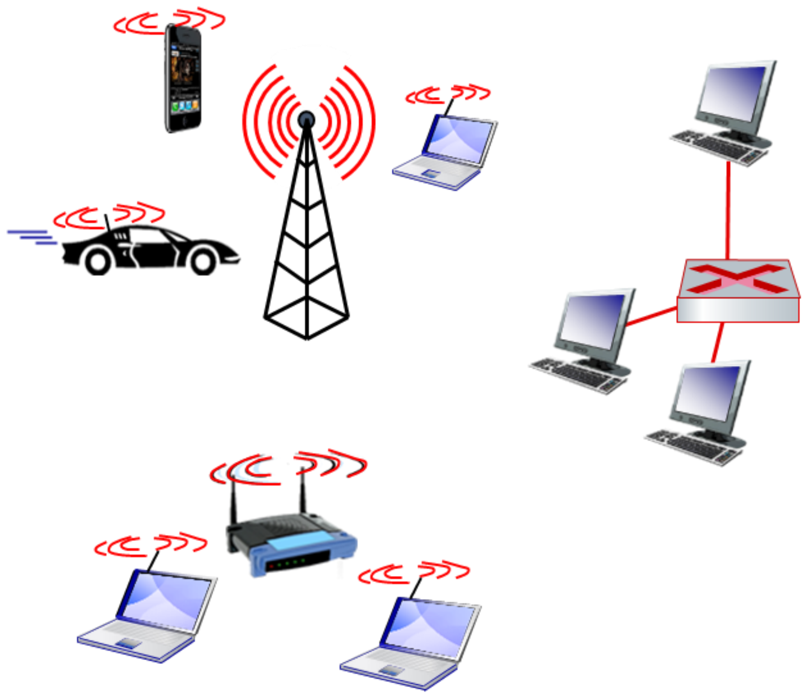




Link layer: services

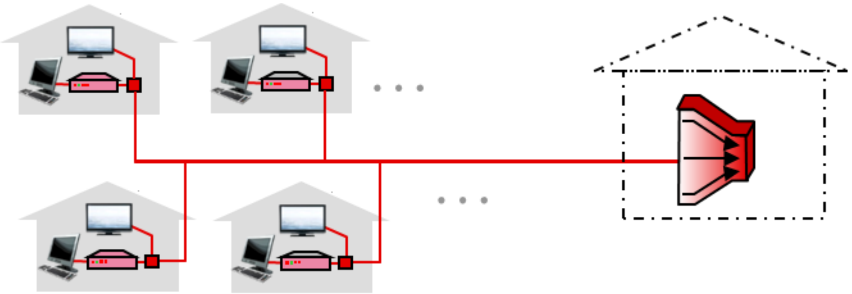
- **framing, link access:**
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - “MAC” addresses in frame headers identify source, destination (different from IP address!)
- **reliable delivery between adjacent nodes**
  - we already know how to do this!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





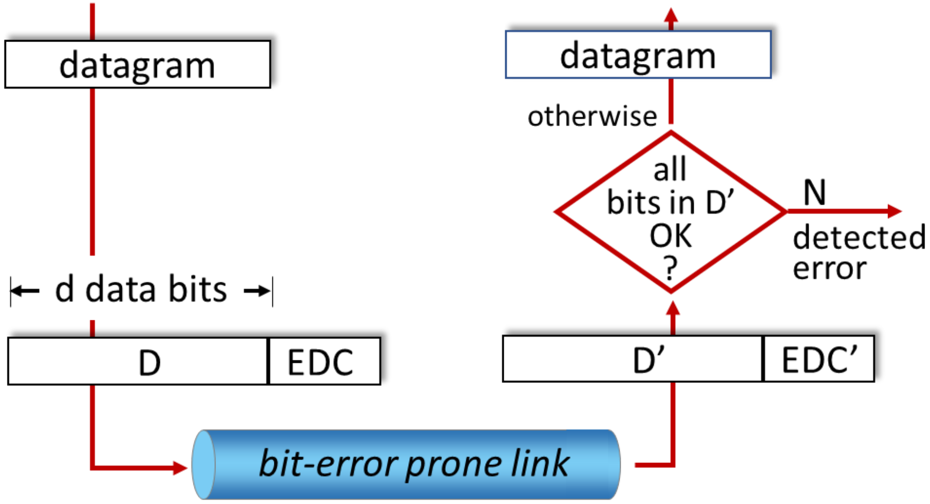
Link layer: services (more)

- **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



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

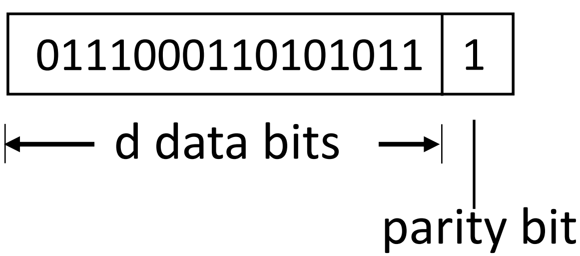
Parity checking

single bit parity:

- detect single bit errors

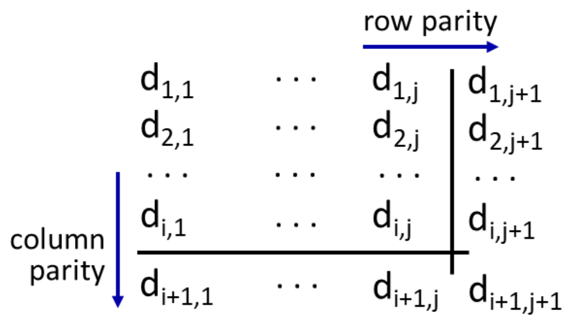
Even parity:

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



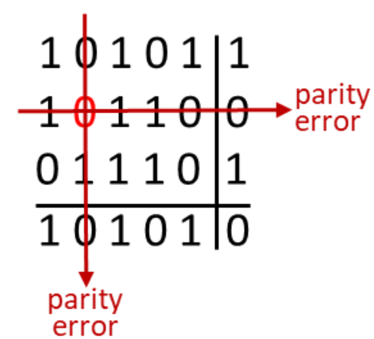
two-dimensional bit parity:

- detect and correct single bit 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:



Checksum

- The data is divided into k segments each of m bits.
- **Sender's End:** The segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum.
- The checksum segment is sent along with the data segments.
- **Receiver's end:** All received segments are added using 1's complement arithmetic to get the sum. The sum is complemented.
- If the result is zero, the received data is accepted; otherwise discarded.

Checksum – Example

- Suppose the following block of 16 bits is to be sent using a checksum of 8 bits.

10101001 00111001

The numbers are added using one's complement

```
           10101001
           00111001
           ------------
     Sum   11100010
Checksum   00011101
```

The pattern sent is 10101001 00111001 00011101

Checksum – Example

- Now suppose the receiver receives the pattern sent and there is no error.

10101001 00111001 00011101

When the receiver adds the three sections, it will get all 1s, which, after complementing, is all 0s and shows that there is no error.

	10101001
	00111001
	00011101
Sum	11111111

Complement 00000000 means that the pattern is OK

Checksum – Example III

- Now suppose there is a burst error of length 5 that affects 4 bits.

10101111 11111001 00011101

- When the receiver adds the three sections, it gets

	10101111
	11111001
	00011101
Partial Sum	**1** 11000101
Carry	**1**
Sum	11000110

Complement 00111001 the pattern is corrupted.

Checksum – Example II

Original Data

10011001	11100010	00100100	10000100
1	2	3	4

k=4, m=8

Sender

```
1          10011001
2          11100010
          ---------
          1 01111011
                   1
          ---------
           01111100
3          00100100
          ---------
           10100000
4          10000100
          ---------
          1 00100100
                   1
          ---------
Sum:       00100101
          ---------
CheckSum:  11011010
```

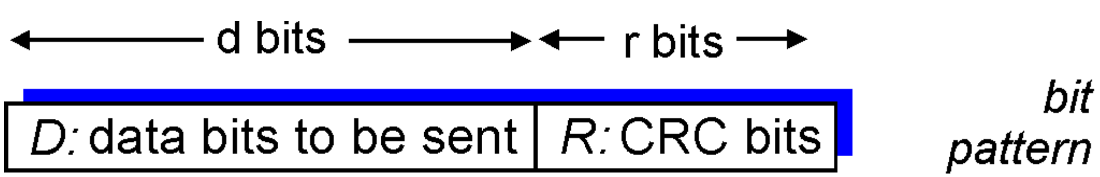
Reciever

```
1          10011001
2          11100010
          ---------
          1 01111011
                   1
          ---------
           01111100
3          00100100
          ---------
           10100000
4          10000100
          ---------
          1 00100100
                   1
          ---------
           00100101
           11011010
          ---------
Sum:       11111111
Complement:00000000
```

Conclusion: Accept Data

Cyclic Redundancy Check

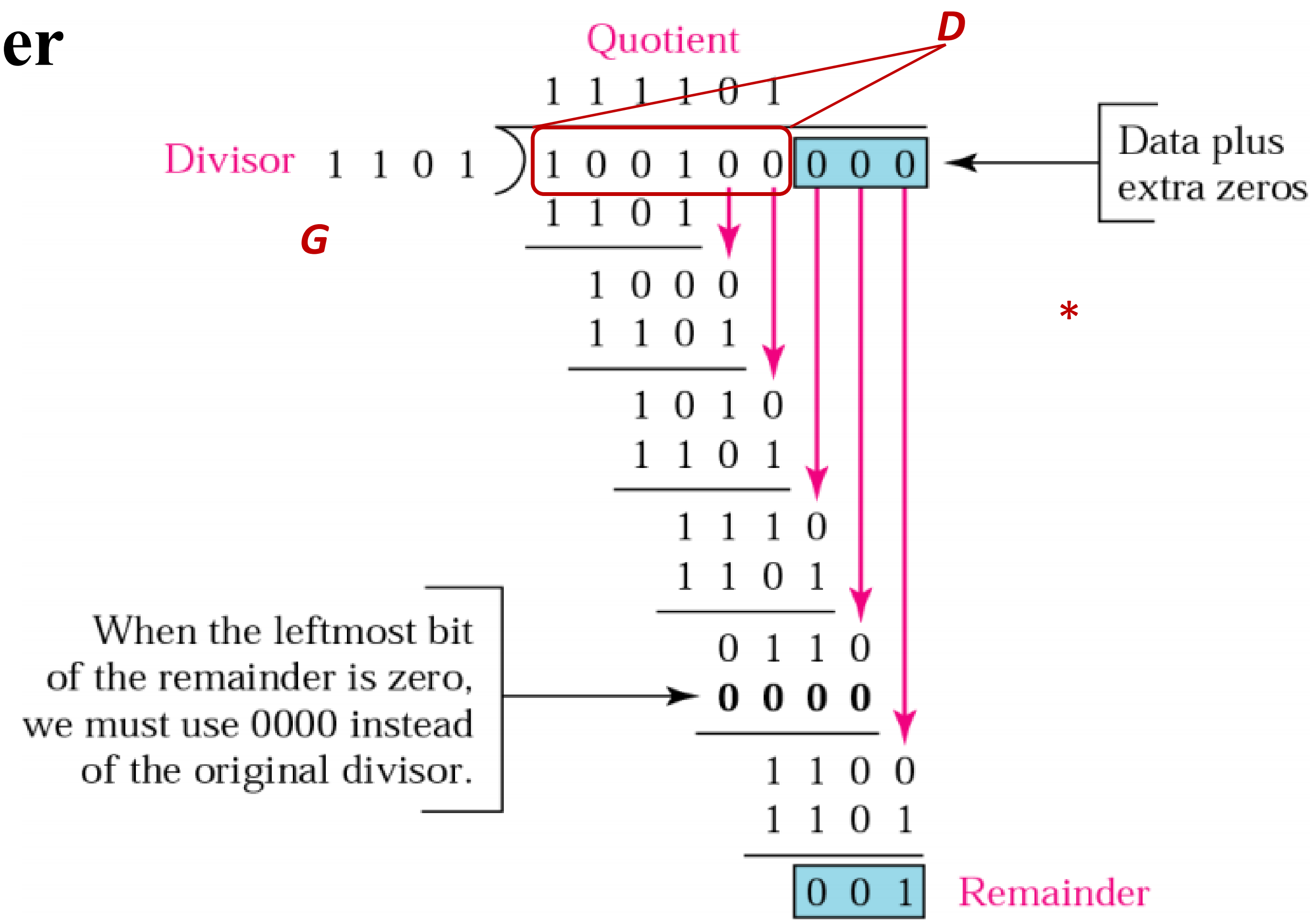
- More powerful error-detection coding
- View data bits, D, as a binary number
- Choose $r + 1$ bit pattern (generator), G
- Goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
- Widely used in practice (Ethernet, 802.11 WiFi, ATM)



$D * 2^r$ XOR R *mathematical formula*

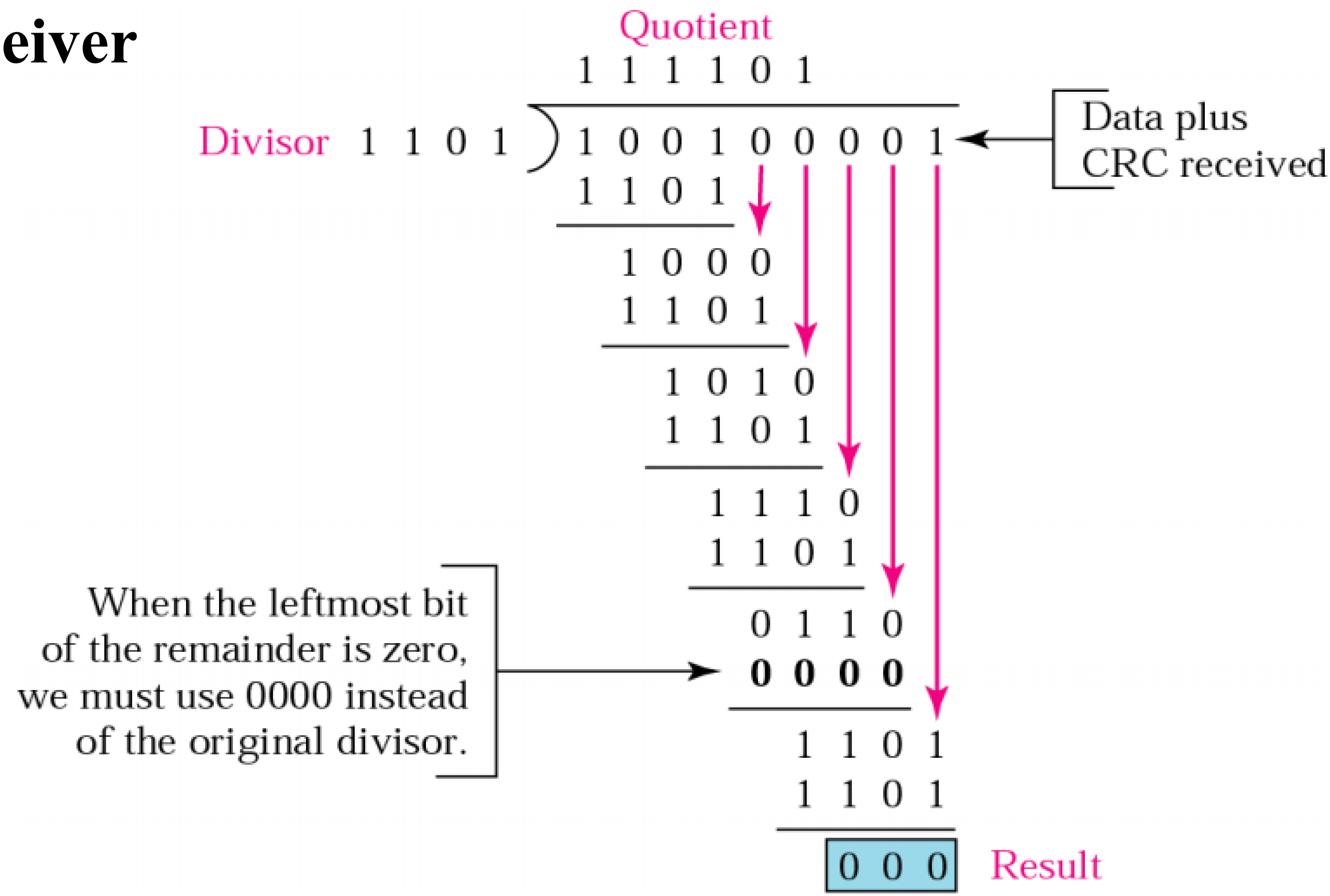
CRC – Example

Sender



CRC – Example

Receiver



Multiple access links, protocols

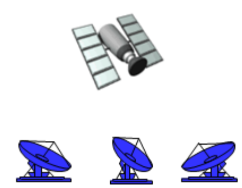
Two types of "links":

- 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
- **Problem:** How to coordinate the access of multiple sending and receiving nodes to a shared broadcast channel?
  - Multiple access problem
- **Answer:** Multiple access protocol, aka Medium access control (MAC) 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*: shared channel (medium) of rate ***R*** bps
- Desired characteristics of MAC 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

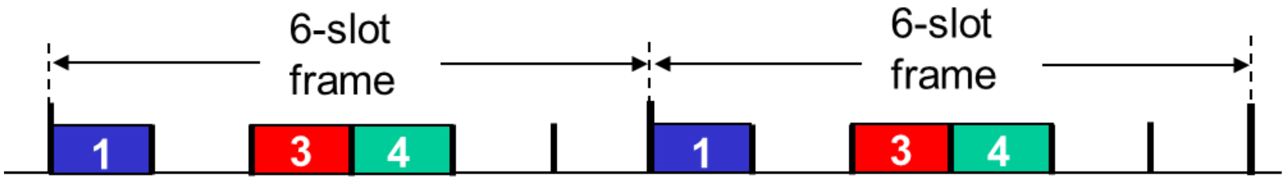
Three broad classes:

1. Channel Partitioning
   - divide channel into smaller “pieces” (time slots, frequency, code)
   - allocate piece to node for exclusive use
2. Random Access
   - channel not divided, allow collisions
   - “recover” from collisions
3. “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



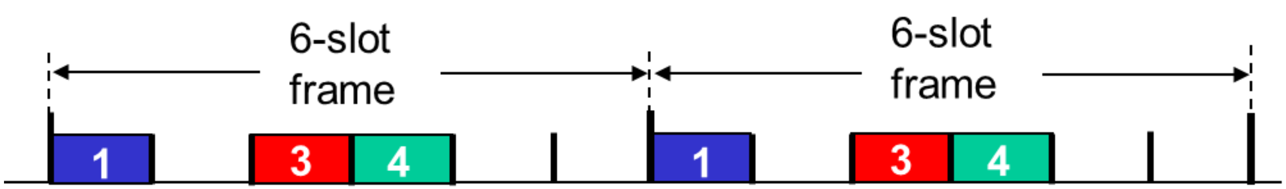
Channel partitioning MAC protocols: TDMA

Advantages

1. No collisions
2. Perfectly fair
   - each node gets the same amount of access time
   - Each node has dedicated ***R/N*** bps bandwidth

Drawbacks

1. Node's bandwidth is limited to ***R/N*** bps even when all other nodes are idle (they have no packet to send)
2. Node always needs to wait to its turn even when others are idle



Random Access MAC Protocols

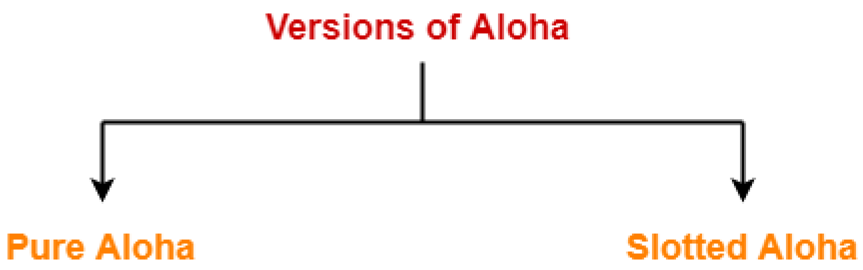
- When node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- Two or more transmitting nodes ➜ "collision"
- **Random access MAC protocol** specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- Examples protocols:
  - ALOHA, slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA

ALOHA

- It was developed at the University of Hawaii in the early 1970s to connect computers situated on different Hawaiian islands.
- The computers of the ALOHA network transmit on the same radio channel whenever they have a packet to transmit.
- From time-to-time packet transmission will collide, but these can be treated as transmission errors, and recovery can take place by retransmission.
- When traffic is very light, the probability of collision is very small, and so retransmissions need to be carried out infrequently.

ALOHA

- There are two different versions of Aloha



Slotted ALOHA

assumptions:

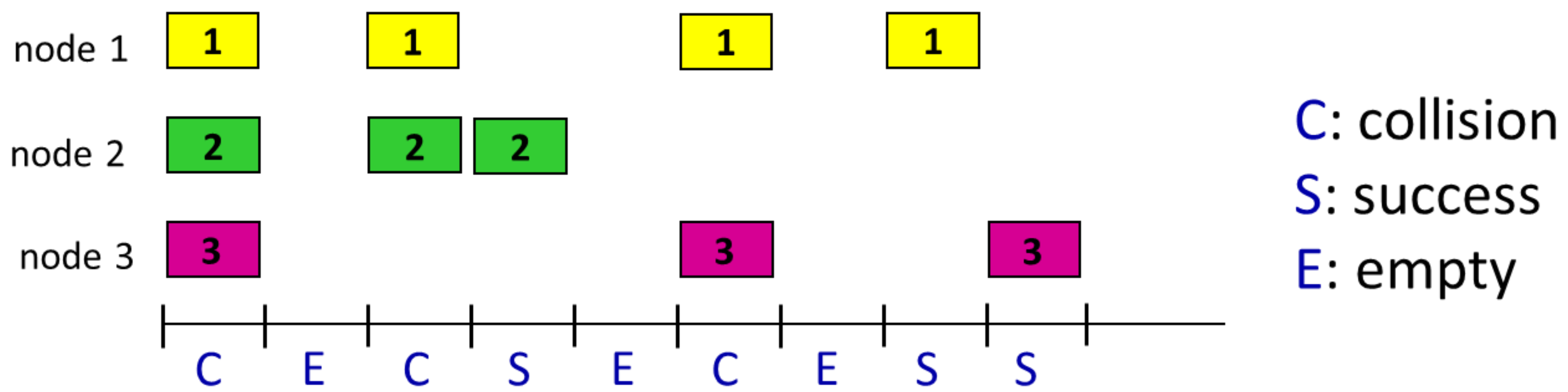
- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

operations:

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:

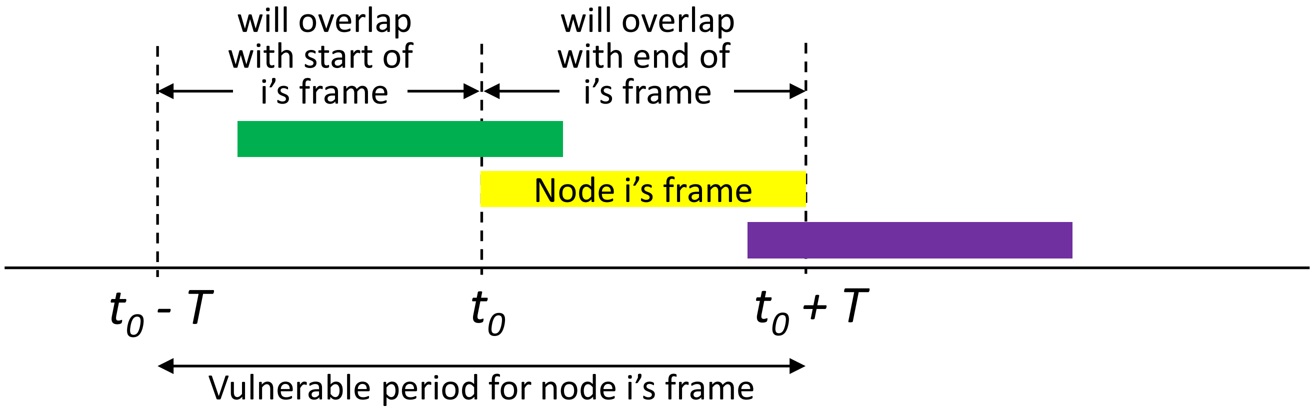
- wasting slots: collisions, idle slots
- nodes may be able to detect collision before finishing packet transmission
- clock synchronization

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 (both new and retry frames)
  - prob that given node has success in a slot: $p(1-p)^{n-1}$
  - prob that any of n nodes has a success, i.e., prob. of success slot:
    $$P_S = \binom{n}{1} p(1-p)^{n-1} = np(1-p)^{n-1}$$
  - Max efficiency → max P_S: find p^* that maximizes P_S
    - Solve $\frac{d}{dp} P_S = 0$ which yields $p^* = \frac{1}{n}$
    - Then $\lim_{n\to\infty} P_S = \lim_{n\to\infty} \left(1-\frac{1}{n}\right)^{n-1} = \frac{1}{e} \approx 0.37$
- **at best**: channel used for useful transmissions 37% of time!

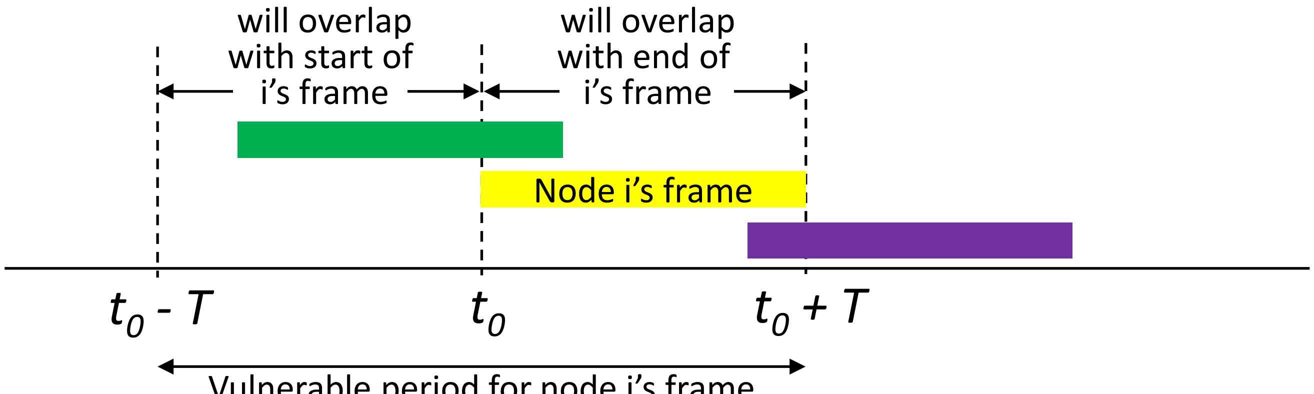
Pure ALOHA

- The sender station
  - transmits frame at any time whenever a new frame arrives
  - after transmitting the frame, it waits for an **ACK frame**.
  - if it receives an ACK within a **timeout**, it can transmit the next frame if there's any.
  - otherwise, it assumes a collision and retransmits the frame after random time.
- The receiver station
  - when receives a data frame, it replies with ACK frame



Pure ALOHA: efficiency

- *suppose: n* nodes with many frames to send, each transmits in interval T with probability p (both new and retry frames)
  - Probability of successful transmission: prob that only one of n nodes transmits and other $n-1$ nodes don't transmit during vulnerable period
    $$P_S = \binom{n}{1} p(1-p)^{n-1}(1-p)^{n-1} = np(1-p)^{2n-2}$$
  - Max efficiency → max P_S: find p^* that maximizes P_S
    - Solve $\frac{d}{dp}P_S = 0$ which yields $p^* = \frac{1}{2n-1}$
    - Then $\lim_{n\to\infty} P_S = \lim_{n\to\infty} \frac{n}{2n-1}\left(1-\frac{1}{2n-1}\right)^{2n-2} = \frac{1}{2e} \approx 0.18$
- **at best**: channel used for useful transmissions 18% of time!



Carrier Sense Multiple Access (CSMA)

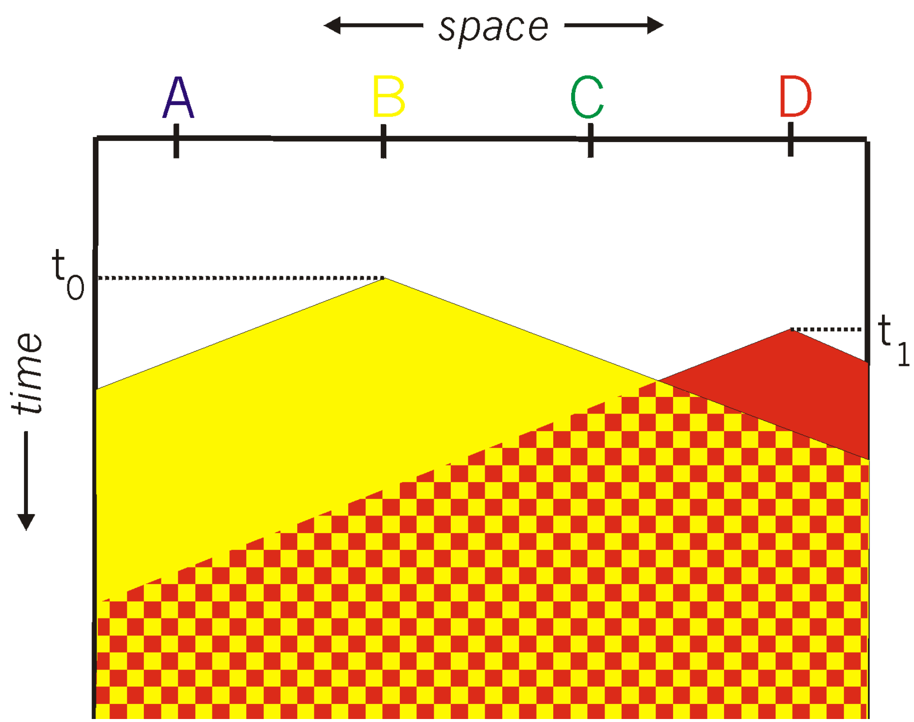
CSMA: listen before transmit:

- If channel sensed idle: transmit entire frame
- If channel sensed busy, defer transmission

human analogy: don’t interrupt others!

CSMA Collisions

- Collisions *can* still occur: Propagation delay means two nodes may not hear each other's transmission
- Collision: Entire packet transmission time wasted
  - Distance and propagation delay play role in determining collision probability



CSMA/CD (Collision Detection)

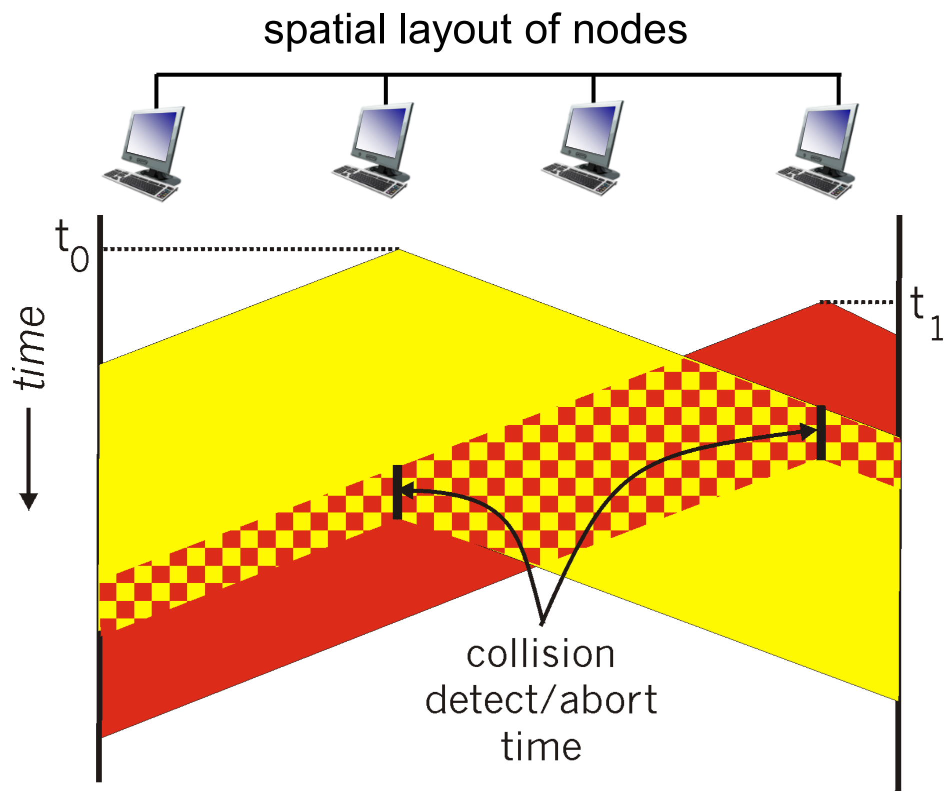
CSMA/CD: CSMA with collision detection

- Collisions *detected* within short time
- Colliding transmissions aborted, reducing channel wastage

- Collision detection:
  - Easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - Difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

Human analogy: the polite conversationalist

CSMA/CD (Collision Detection)



Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. NIC senses the channel
   - If idle: start frame transmission.
   - If busy: wait until channel becomes idle, then transmit.
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame!
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters binary (exponential) backoff:
   - After ***m***-th collision, NIC chooses ***K*** at random from $\{0,1,2, \ldots, 2^m-1\}$
   - and waits ***K***·512 bit times,
   - Then returns to Step 2

More collisions →longer backoff interval

CSMA/CD Efficiency

- T_{prop} = max propagation 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!

“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!

random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

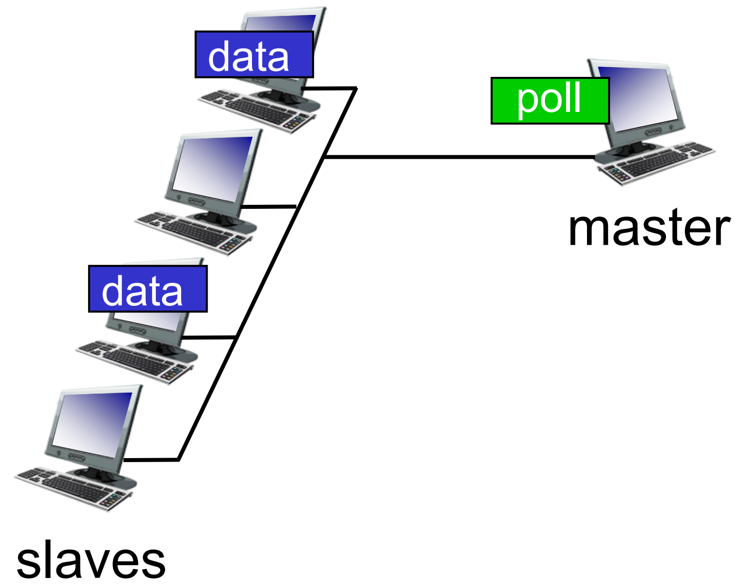
“taking turns” protocols

- look for best of both worlds!

“Taking turns” MAC protocols

polling:

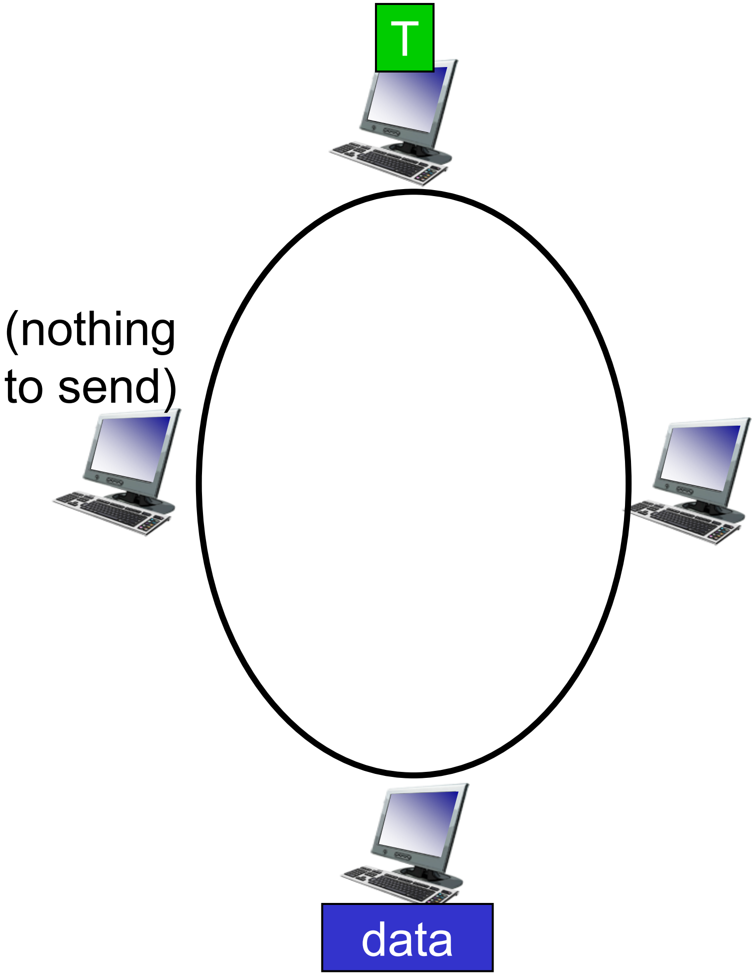
- master node “invites” other nodes to transmit in turn
- typically used with “dumb” devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



"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)



Acknowledgements

- Most part of this tutorial was prepared by M.Fahim, G.Succi, and A.Tormasov

Reference

- This tutorial is based on the on the following resources as well as relevant material over the internet.
  - Chapter 5 of Computer Networking: A Top Down Approach (6th edition) by Jim Kurose and Keith Ross
  - The material is aligned and add/deleted according to the need of the students.