

# Computer Networks

## CS3001

### (Section BDS-7A)

## Lecture 27

Instructor: Dr. Syed Mohammad Irteza  
Assistant Professor, Department of Computer Science  
30 November, 2023

# Link layer, LANs: roadmap

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

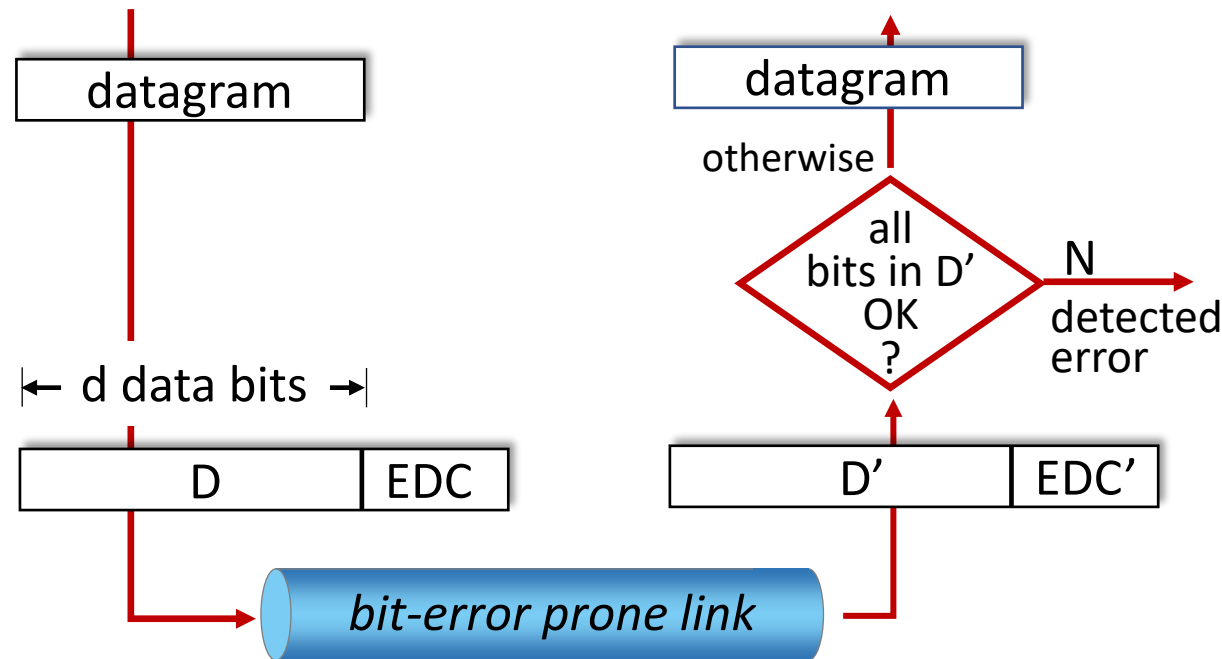


- a day in the life of a web request

# Error detection

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

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



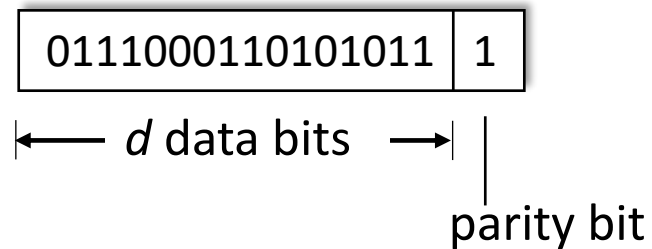
Error detection not 100% reliable!

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

# Parity checking

## single bit parity:

- detect single bit errors



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

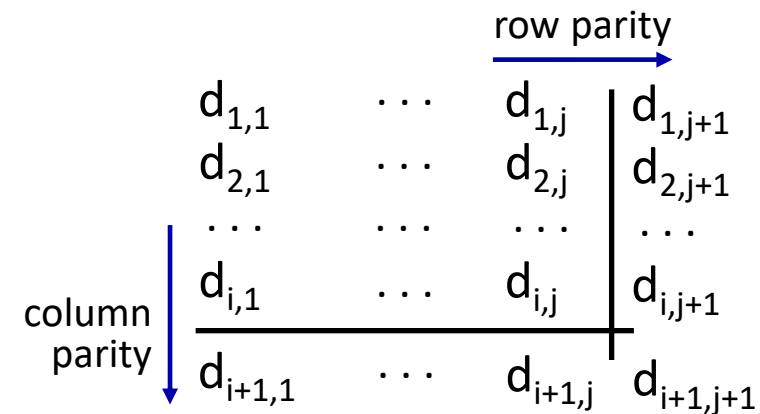
## At receiver:

- compute parity of  $d$  received bits
- compare with received parity bit – if different than error detected



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

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



no errors:

1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

**detected and correctable single-bit error:**

1	0	1	0	1	1
<del>1</del>	<del>0</del>	<del>1</del>	<del>1</del>	<del>0</del>	<del>0</del>
0	1	1	1	0	1
0	0	1	0	1	0

parity error  $\rightarrow$

$\downarrow$   
parity error

# Internet checksum (review, see section 3.3)

*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, specified in CRC standard)



- sender:* compute  $r$  CRC bits, **R**, such that  $\langle D, R \rangle$  *exactly* divisible by  $G \pmod{2}$
- receiver knows  $G$ , divides  $\langle D, R \rangle$  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

Sender wants to compute R  
such that:

$$D \cdot 2^r \text{ XOR } R = nG$$

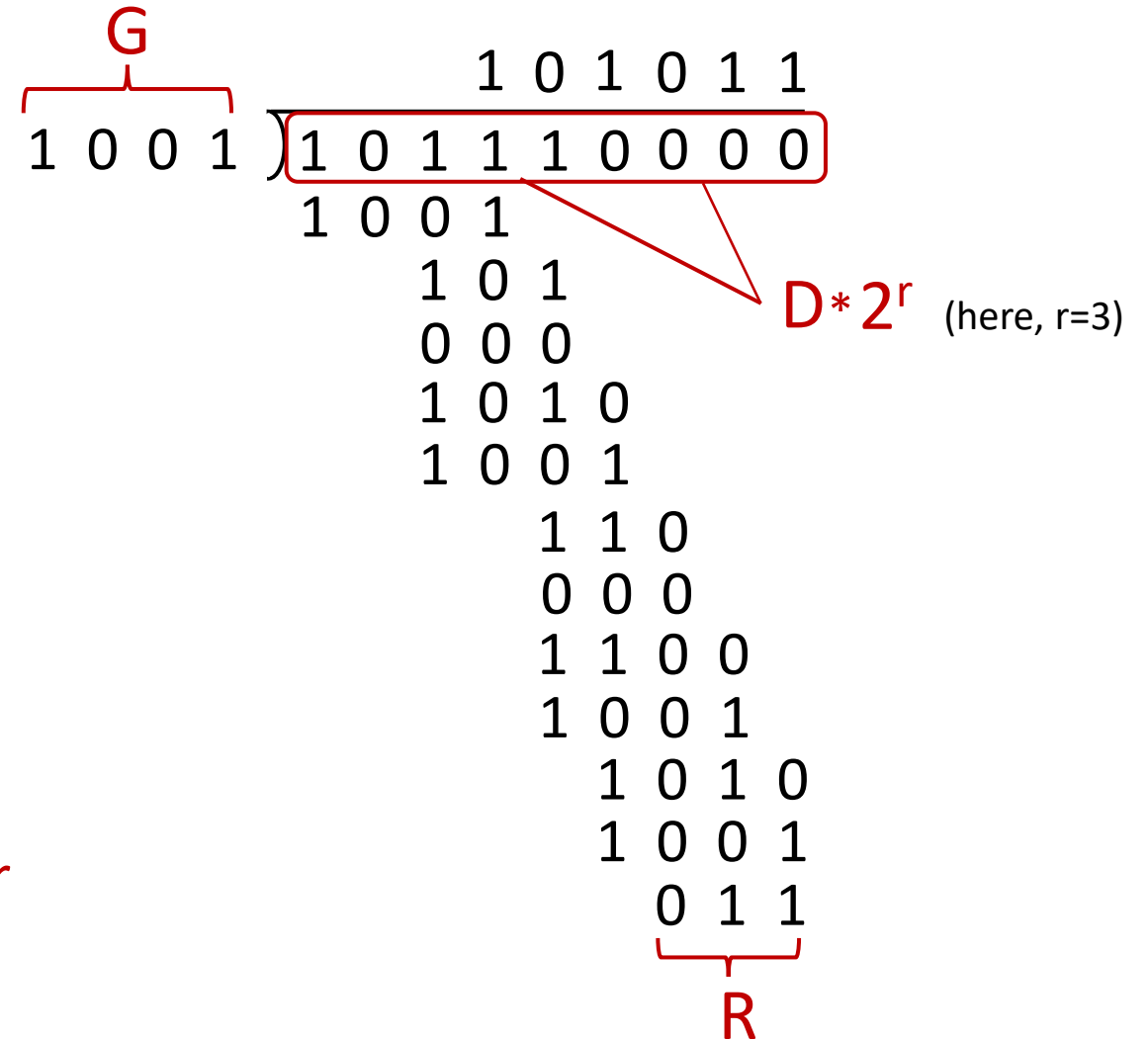
... or equivalently (XOR R both sides):

$$D \cdot 2^r = nG \text{ XOR } R$$

... which says:

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

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



\* Check out the online interactive exercises for more examples: [http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)





# 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



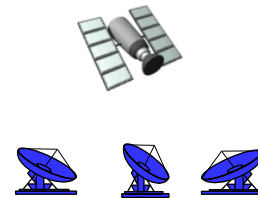
shared wire (e.g.,  
cabled Ethernet)



shared radio: 4G/5G



shared radio: WiFi



shared radio: satellite



humans at a cocktail party  
(shared air, acoustical)

# Multiple access protocols

- 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

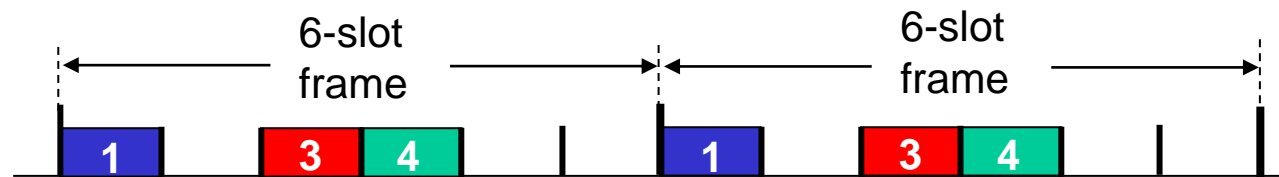
- **“taking turns”**

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

# Channel partitioning MAC protocols: TDMA

## TDMA: time division multiple access

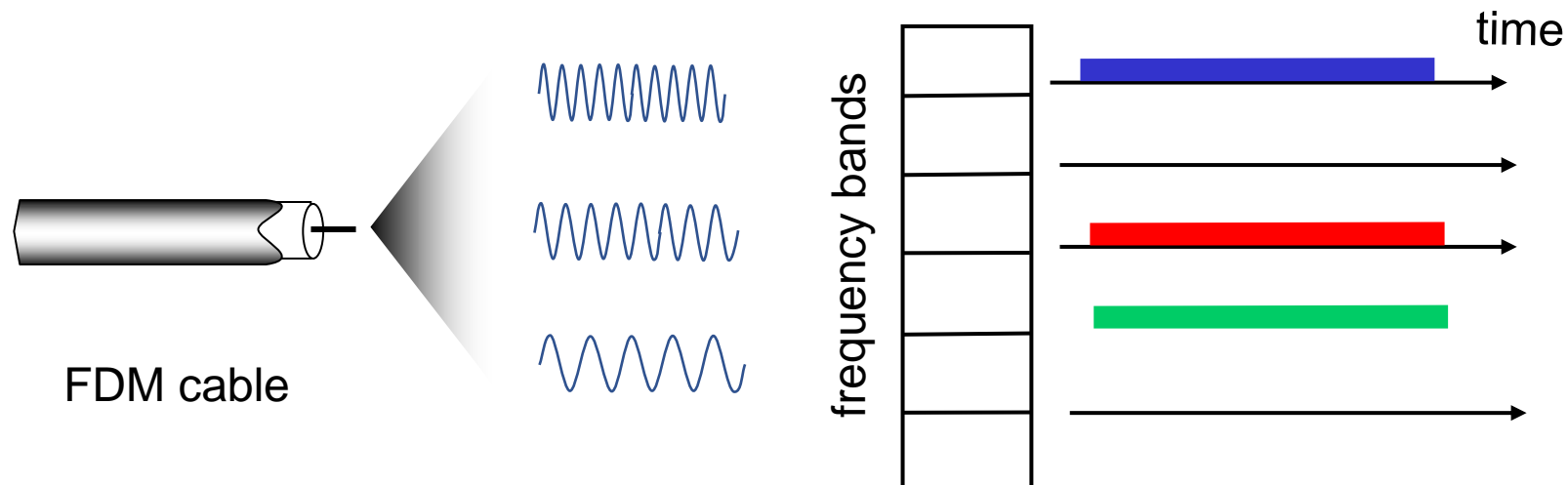
- access to channel in “rounds”
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



# 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

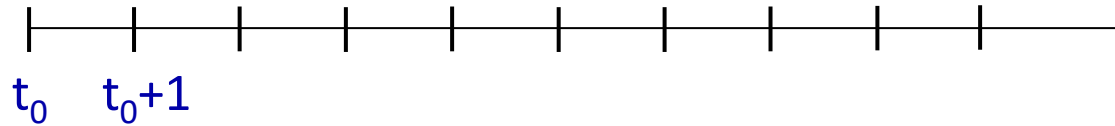


# Random access protocols

- 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 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:

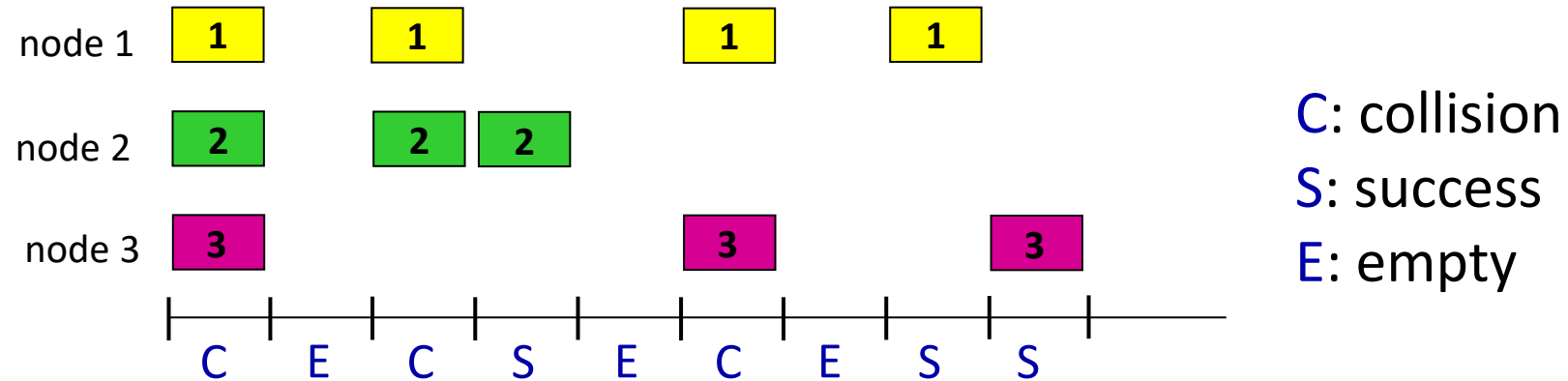
- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

## 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



## 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!

