

# DATA LINK LAYER

On the Wire

# Link layer

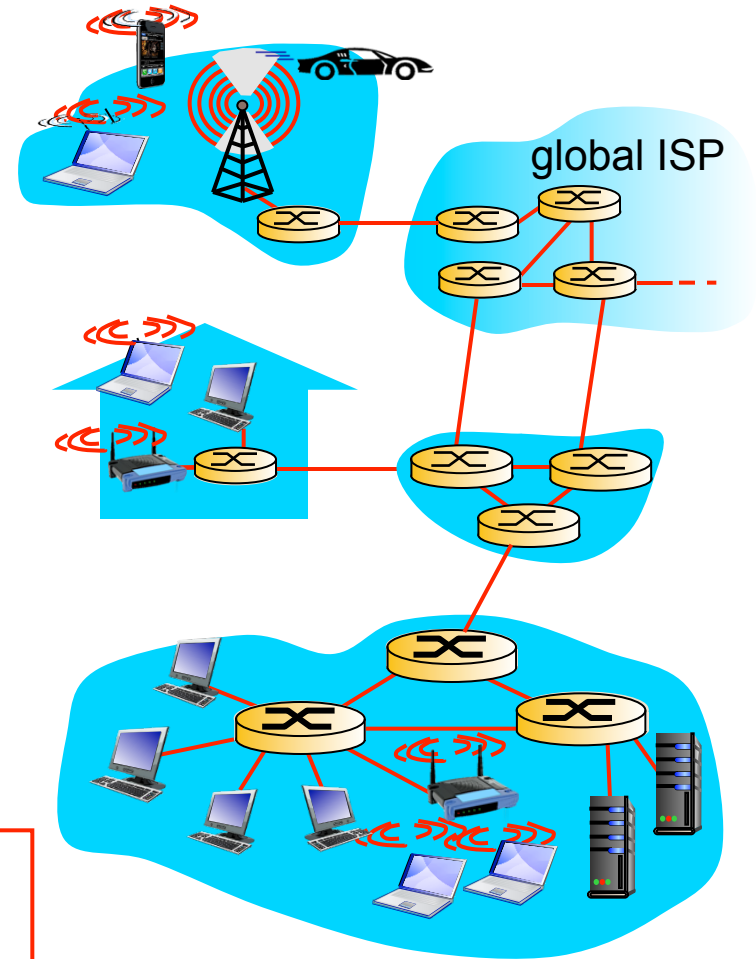
- understand principles behind link layer services:
  - ▣ error detection, correction
  - ▣ sharing a broadcast channel: multiple access
  - ▣ link layer addressing
  - ▣ local area networks: Ethernet, VLANs
- instantiation, implementation of various link layer technologies

# Link layer: introduction

## *terminology:*

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

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



# Data Link Layer

- Bridging Protocol
  - ▣ Network Layer
  - ▣ Physical Layer
- Places data (**Frames**) onto the Network
- Receives data from the Network

# Transit Example

*UK*



*DE*



# How does it work?

---

*Data Link Layer*



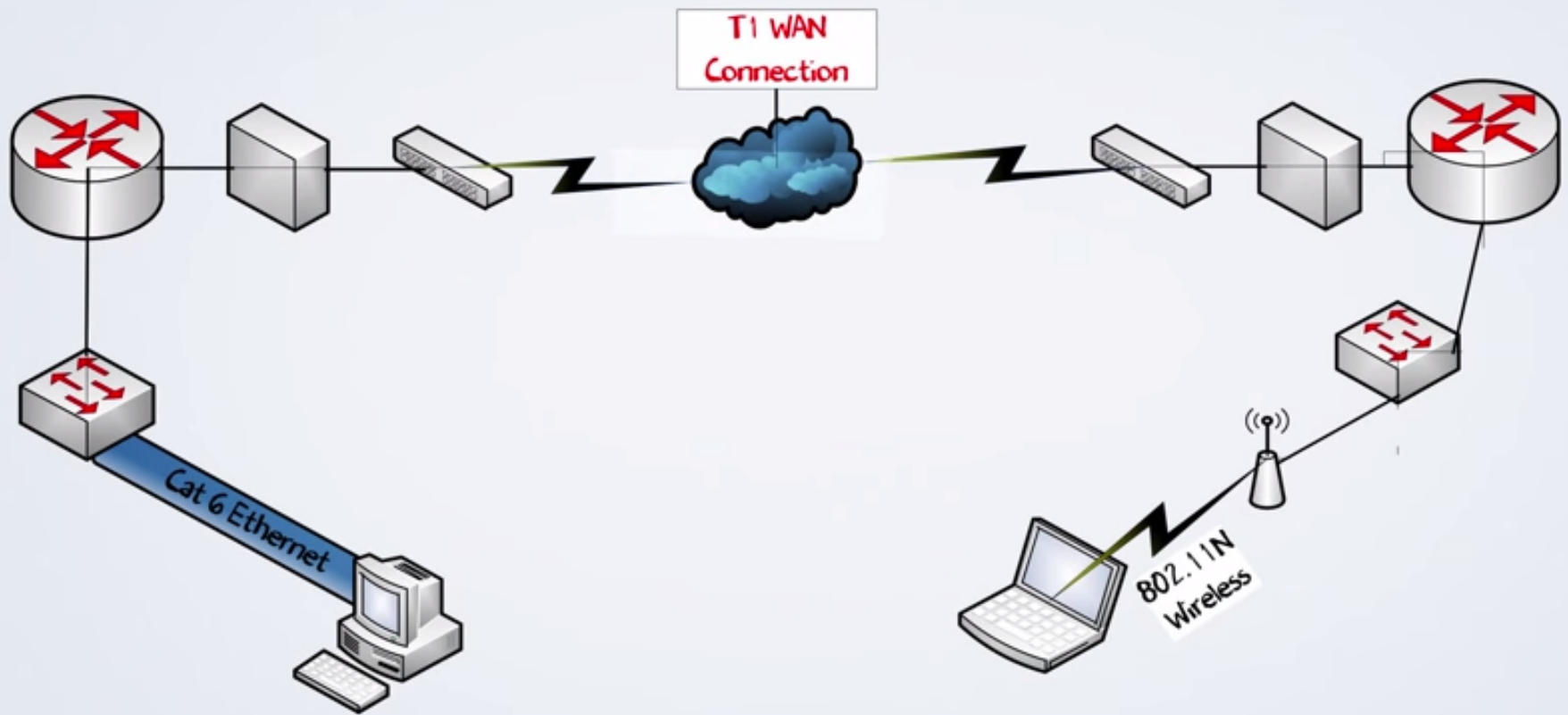
```
graph TD; A[Data Link Layer] --> B[Logical Link Control]; A --> C[Media Access Control];
```

A large blue arrow points from the 'Data Link Layer' box to the 'Logical Link Control' box, indicating a hierarchical or functional relationship.

*Logical Link Control*

*Media Access Control*

# Medium Independent



# Link layer: context

- datagram transferred by different link protocols over different links:
  - ▣ e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- each link protocol provides different services
  - ▣ e.g., may or may not provide rdt 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 used in frame headers to identify source, dest
  - different from IP address!

## □ *reliable delivery between adjacent nodes*

- we learned how to do this already (chapter 3)!
- seldom used on low bit-error link (fiber, some twisted pair)
- wireless links: high error rates
  - *Q*: why both link-level and end-end reliability?

# Link layer services

## □ *flow control:*

- ▣ pacing between adjacent sending and receiving nodes

## □ *error detection:*

- ▣ errors caused by signal attenuation, noise.
- ▣ receiver detects presence of errors:
  - signals sender for retransmission or drops frame

## □ *error correction:*

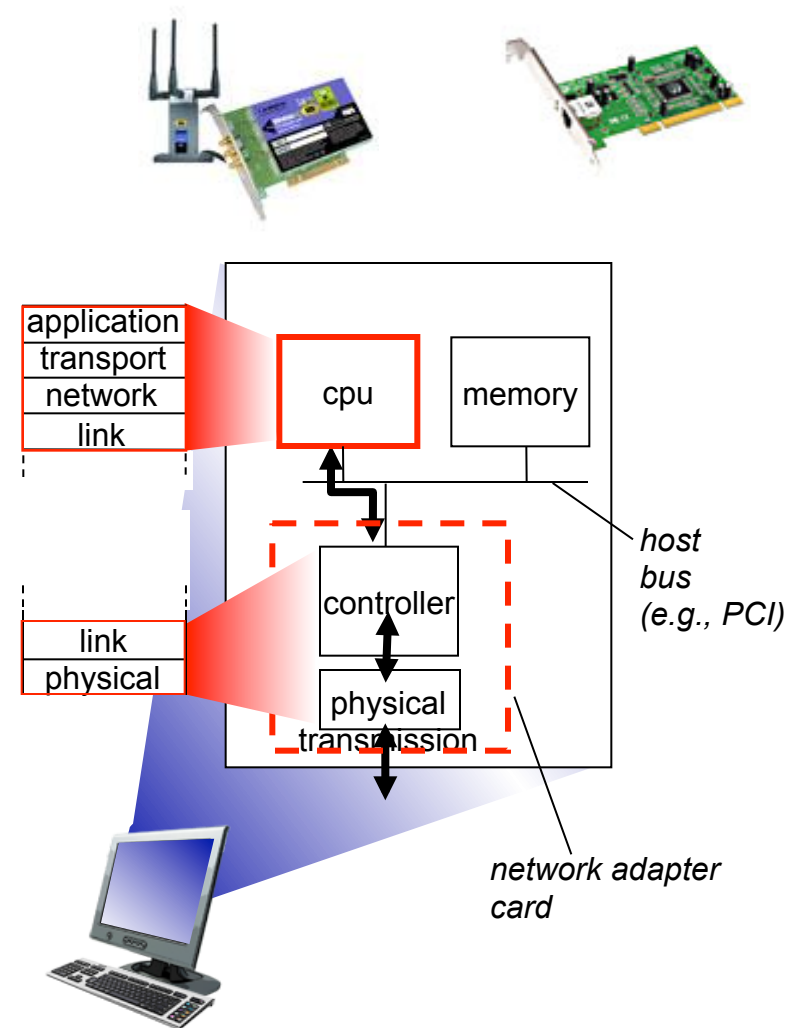
- ▣ receiver identifies *and corrects* bit error(s) without resorting to retransmission

## □ *half-duplex and full-duplex*

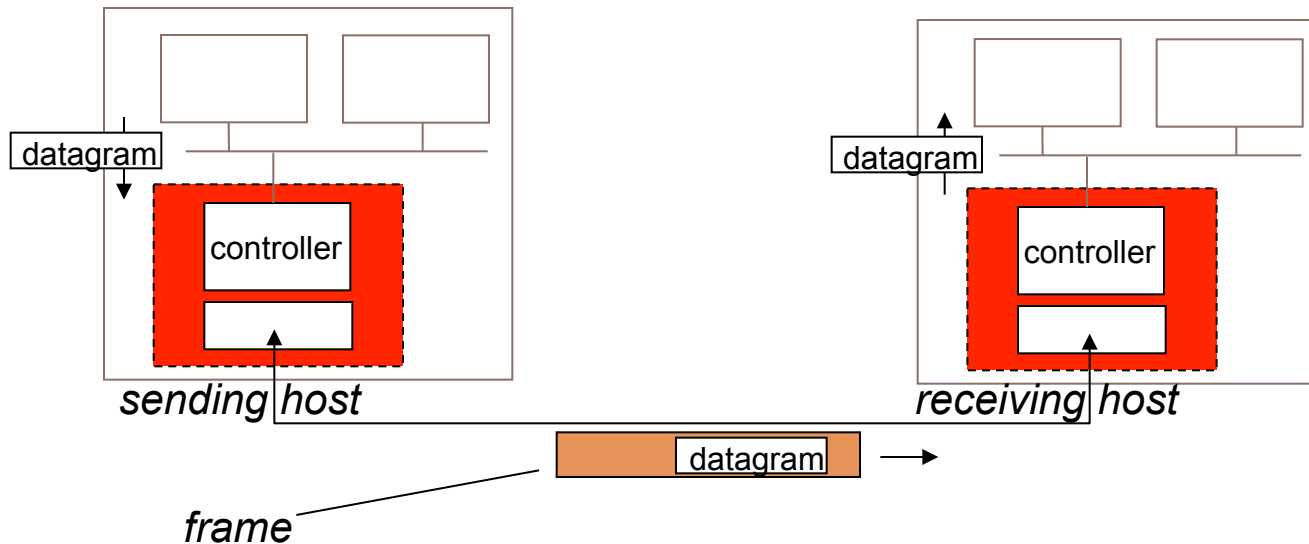
- ▣ with half duplex, nodes at both ends of link can transmit, but not at same time

# Where is the link layer implemented?

- in each and every host
- link layer implemented in “adaptor” (aka *network interface card* NIC) or on a chip
  - ▣ Ethernet card, 802.11 card; Ethernet chipset
  - ▣ implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



# Adaptors communicating



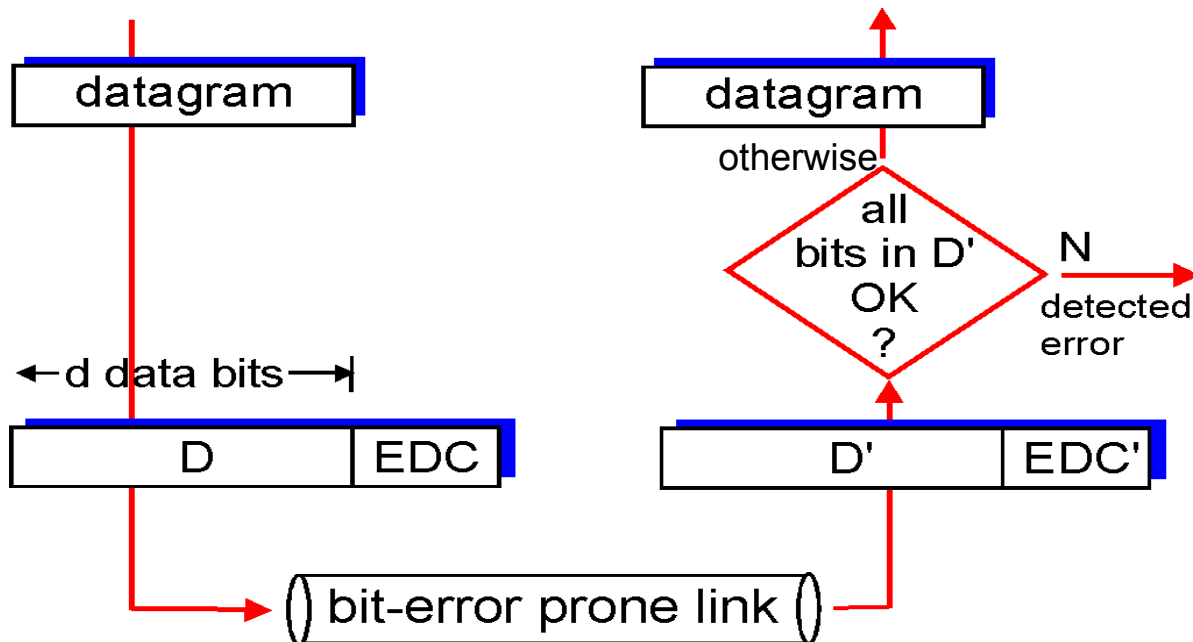
- sending side:
  - ▣ encapsulates datagram in frame
  - ▣ adds error checking bits, rdt, flow control, etc.
- receiving side
  - ▣ looks for errors, rdt, flow control, etc
  - ▣ extracts datagram, passes to upper layer at receiving side

# Error detection

EDC= Error Detection and Correction bits (redundancy)

D = Data protected by error checking, may include header fields

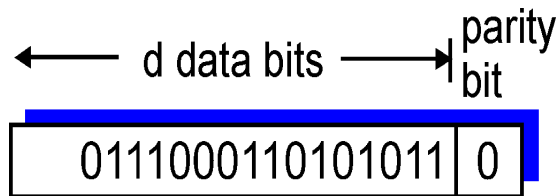
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



# Parity checking

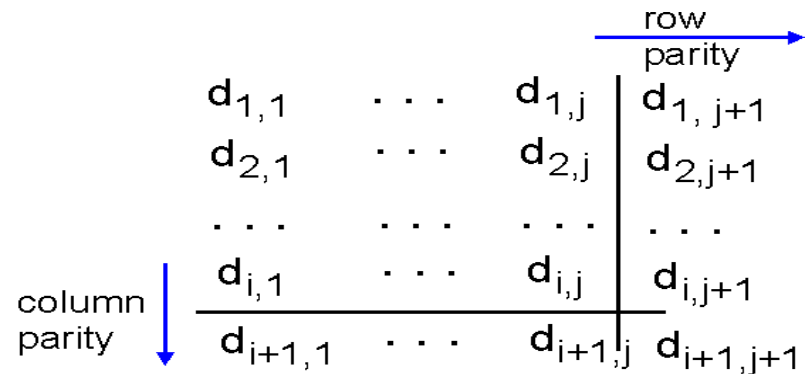
## *single bit parity:*

- ❖ detect single bit errors



## *two-dimensional bit parity:*

- ❖ detect and correct single bit 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

*no errors*

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

parity error

*correctable  
single bit error*

# Internet checksum (review)

**goal:** detect “errors” (e.g., flipped bits) in transmitted packet  
(note: used at transport layer *only*)

## **sender:**

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

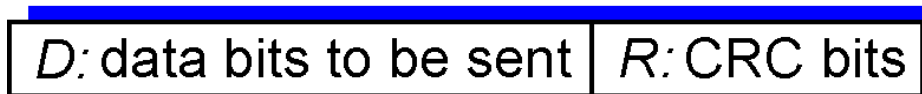
## **receiver:**

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - ▣ NO - error detected
  - ▣ YES - no error detected. *But maybe errors nonetheless?*

# 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
  - ▣  $\langle D, R \rangle$  exactly divisible by  $G$  (modulo 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, ATM)

← d bits → ← r bits →



*bit  
pattern*

$$D * 2^r \text{ XOR } R$$

*mathematical  
formula*



# CRC example

want:

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

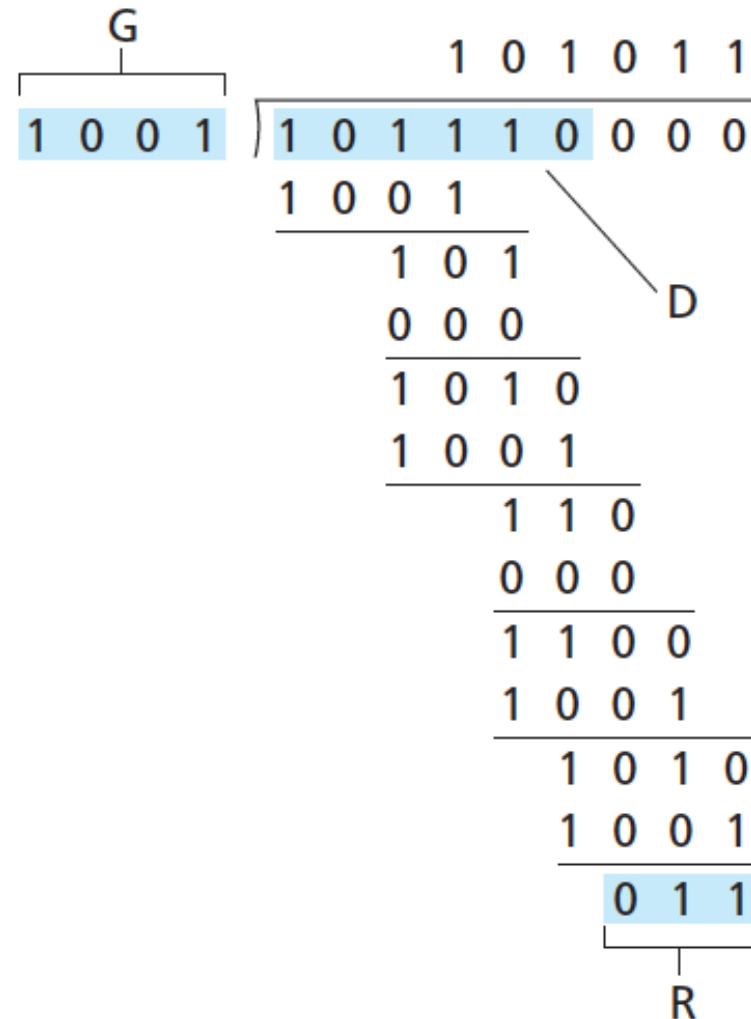
*equivalently:*

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

*equivalently:*

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

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



# Multiple access links, protocols

two types of “links”:

- point-to-point

- PPP for dial-up access
- point-to-point link between Ethernet switch, host

- *broadcast (shared wire or medium)*

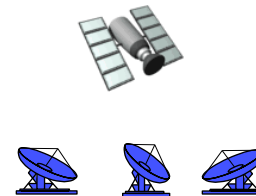
- old-fashioned Ethernet
- upstream HFC
- 802.11 wireless LAN



shared wire (e.g.,  
cabled Ethernet)



shared RF  
(e.g., 802.11 WiFi)



shared RF  
(satellite)

# 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:* broadcast channel 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

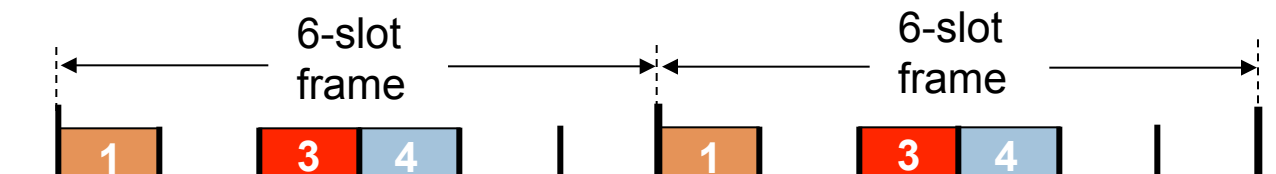
- *“taking turns”*

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

# Channel partitioning MAC protocols: TDMA

## TDMA: time division multiple access

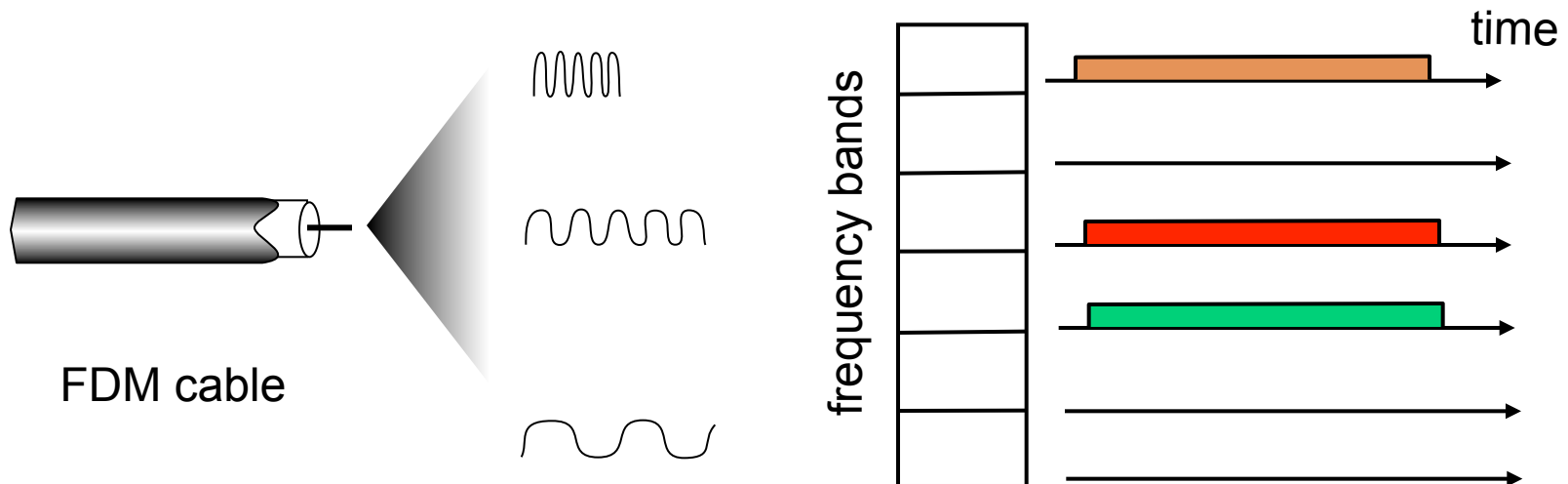
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



# 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 pkt, 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:
  - ▣ slotted ALOHA
  - ▣ ALOHA
  - ▣ CSMA, CSMA/CD, CSMA/CA



# Slotted ALOHA

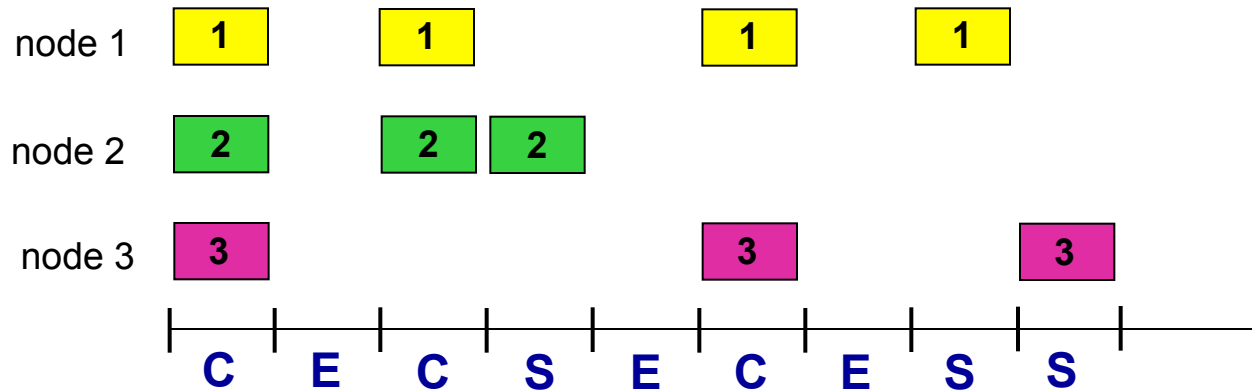
## *assumptions:*

- 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 prob.  $p$  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:

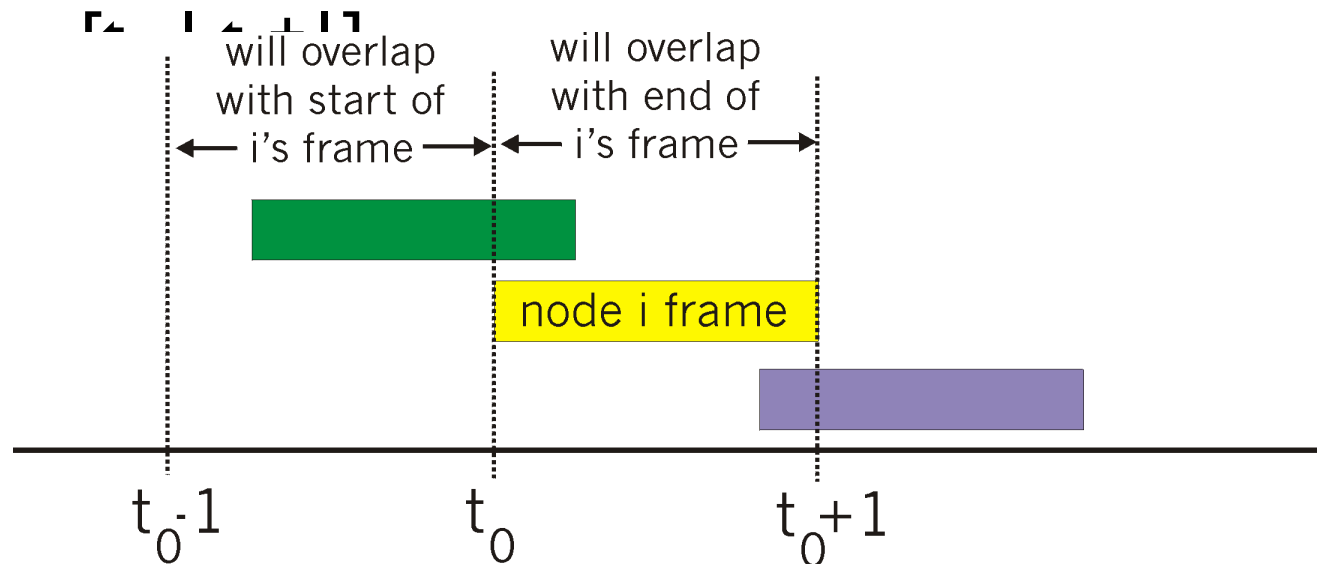
$$\text{max efficiency} = 1/e = .37$$

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



# Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives
  - ▣ transmit immediately
- collision probability increases:
  - ▣ frame sent at  $t_0$  collides with other frames sent in



# Pure ALOHA efficiency

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

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

$$P(\text{no other node transmits in } [t_0 - 1, t_0])$$

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

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

... choosing optimum  $p$  and then letting  $n$

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

**even worse than slotted Aloha!**

# CSMA (carrier sense multiple access)

**CSMA:** listen before transmit:

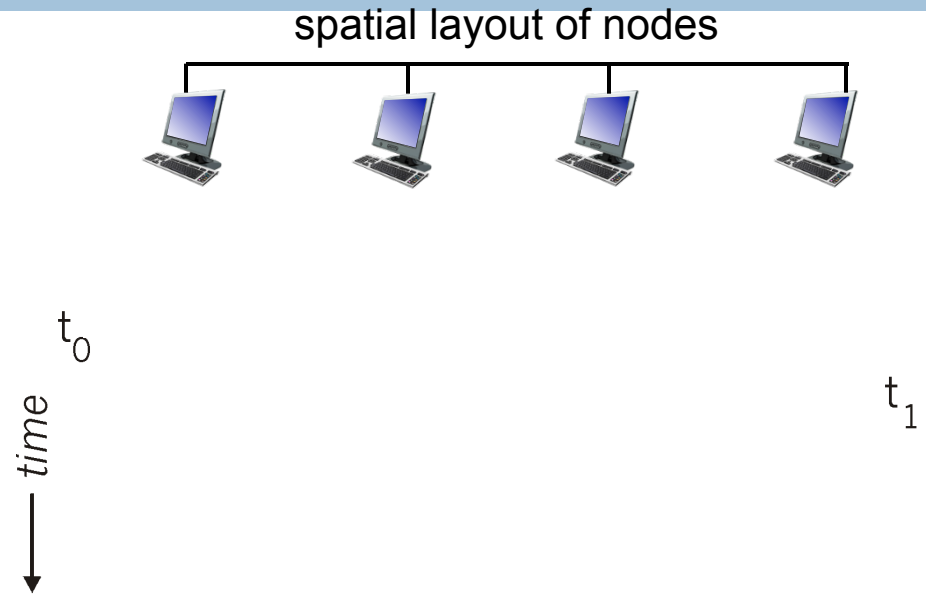
if channel sensed idle: transmit entire frame

- 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 & propagation  
delay play role in in  
determining collision  
probability



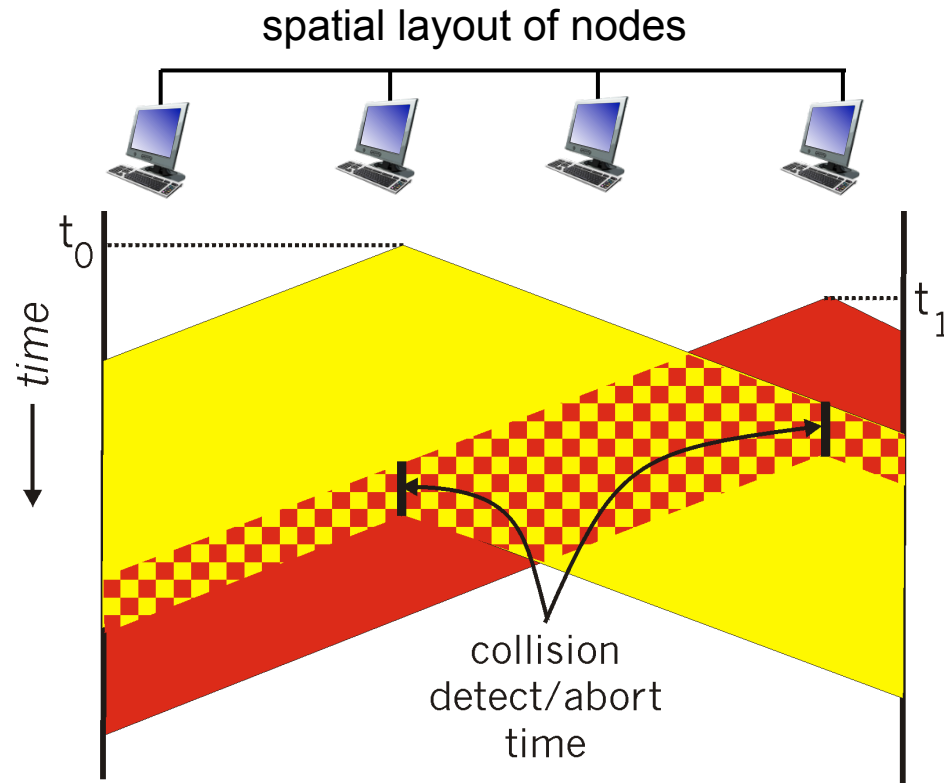
# CSMA/CD (collision detection)

**CSMA/CD:** carrier sensing, deferral as in CSMA

- ▣ 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. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, then transmits.
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  $\{0, 1, 2, \dots, 2^m - 1\}$ . NIC waits  $K \cdot 512$  bit times, returns to Step 2
  - ▣ longer backoff interval with more collisions

# CSMA/CD efficiency

- $T_{\text{prop}}$  = max prop delay between 2 nodes in LAN
- $t_{\text{trans}}$  = time to transmit max-size frame

$$\text{efficiency} = \frac{1}{1 + 5t_{\text{prop}}/t_{\text{trans}}}$$

- efficiency goes to 1
  - ▣ as  $t_{\text{prop}}$  goes to 0
  - ▣ as  $t_{\text{trans}}$  goes to infinity
- better performance than ALOHA: and simple, cheap, decentralized!