# CAN201: Introduction to Networking

## Lecture 9 - The Link Layer 1

Lecturers: Dr. Gordon Boateng and Dr. Fei Cheng

# Important Information

- **Contact:**
  - Email: Gordon.Boateng@xjtlu.edu.cn
  - Office No.: SC554A

- **Office Hours (Strictly via appointment)**
  - **Tuesday: 14:00-15:00**
  - **Wednesday: 14:00-15:00**

# Recap: Top-Down Approach

| |
|---|
| Application Layer |
| Transport Layer |
| Network Layer |
| Link Layer |
| Physical Layer |

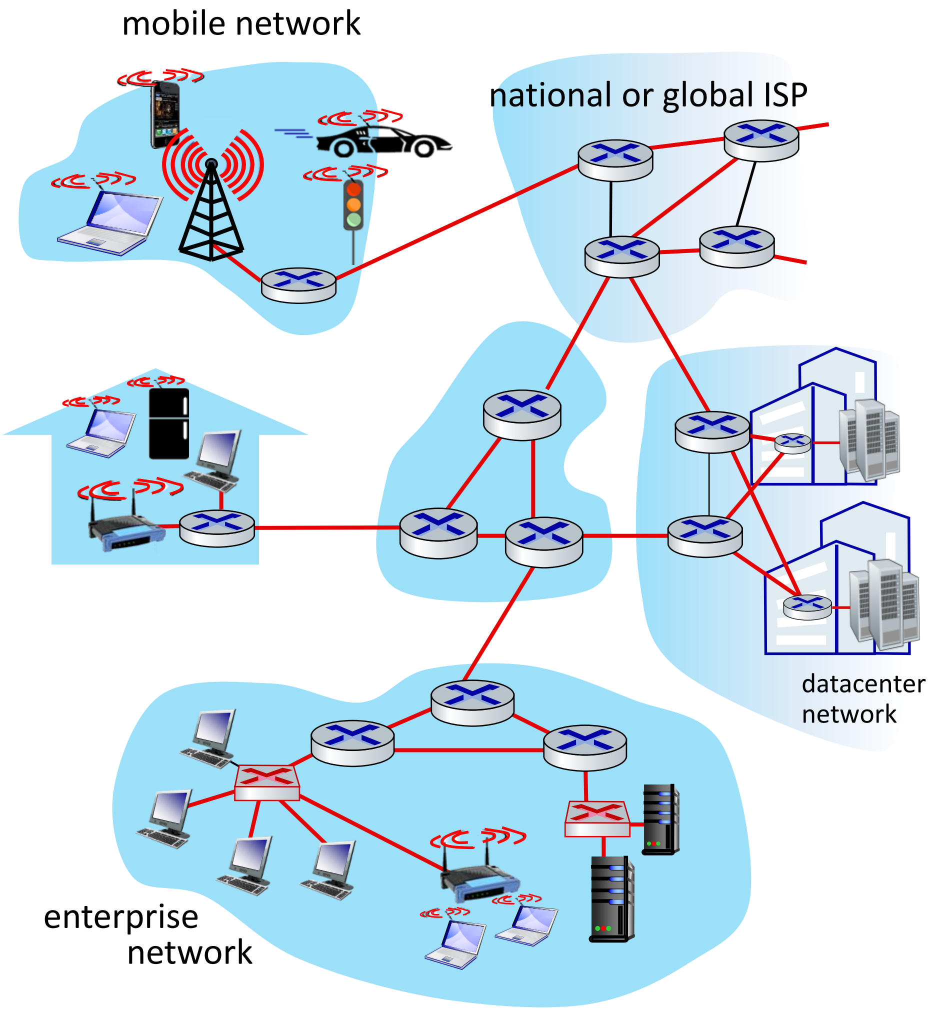
# The Link Layer

- Introduction to the Link Layer
- Error-Detection and -Correction Techniques
- Multiple Access Links and Protocols

# Link layer: Introduction

terminology:

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




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

# Data Unit

Application Layer
Transport Layer
Network Layer
Link Layer
Physical Layer

?

Datagram
Frame
Message
Segment
Bit (or Signal)

# Link Layer: Context

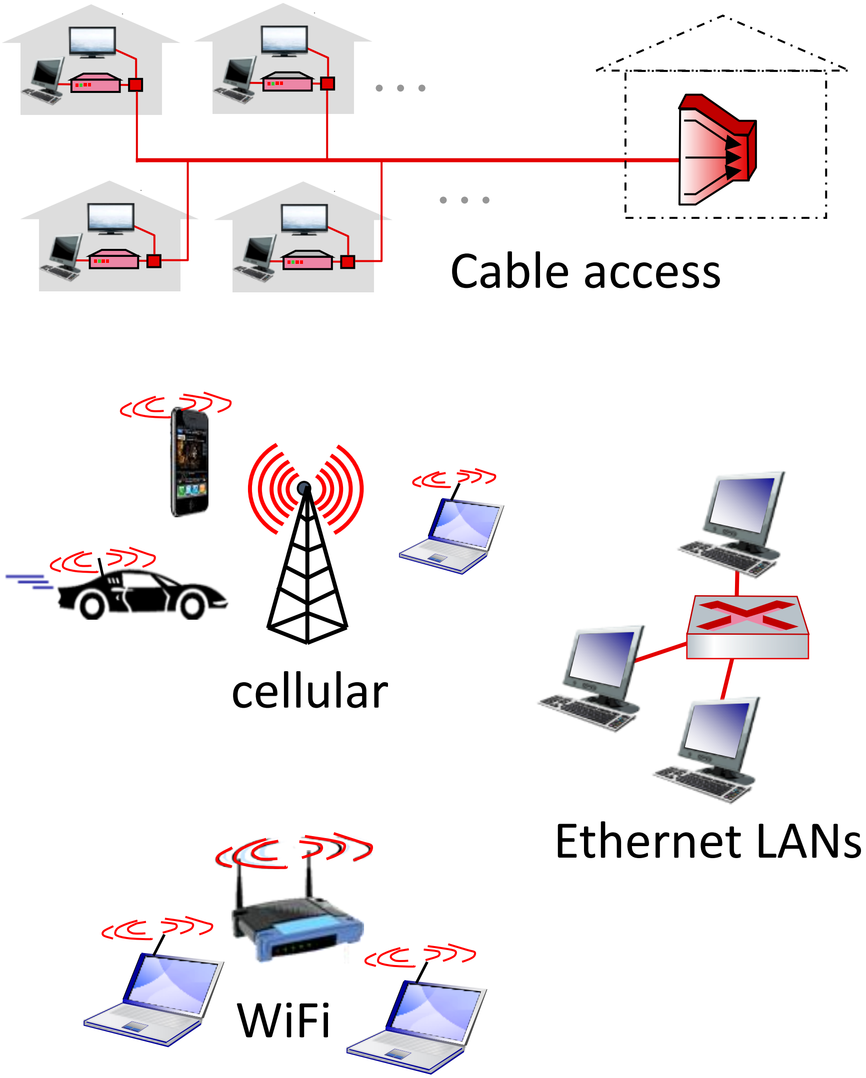
- datagram transferred by different link protocols over different links:
  - e.g., WiFi on first link, Ethernet on next link
- each link protocol provides different services
  - e.g., may or may not provide reliable data transfer over link

transportation analogy:

- trip from Suzhou to Osaka
  - limo: Suzhou to Shanghai Airport
  - plane: Shanghai Airport to Tokyo
  - train: Tokyo to Osaka
- tourist = datagram
- transport segment = communication link
- transportation mode = link-layer protocol
- travel agent = routing algorithm

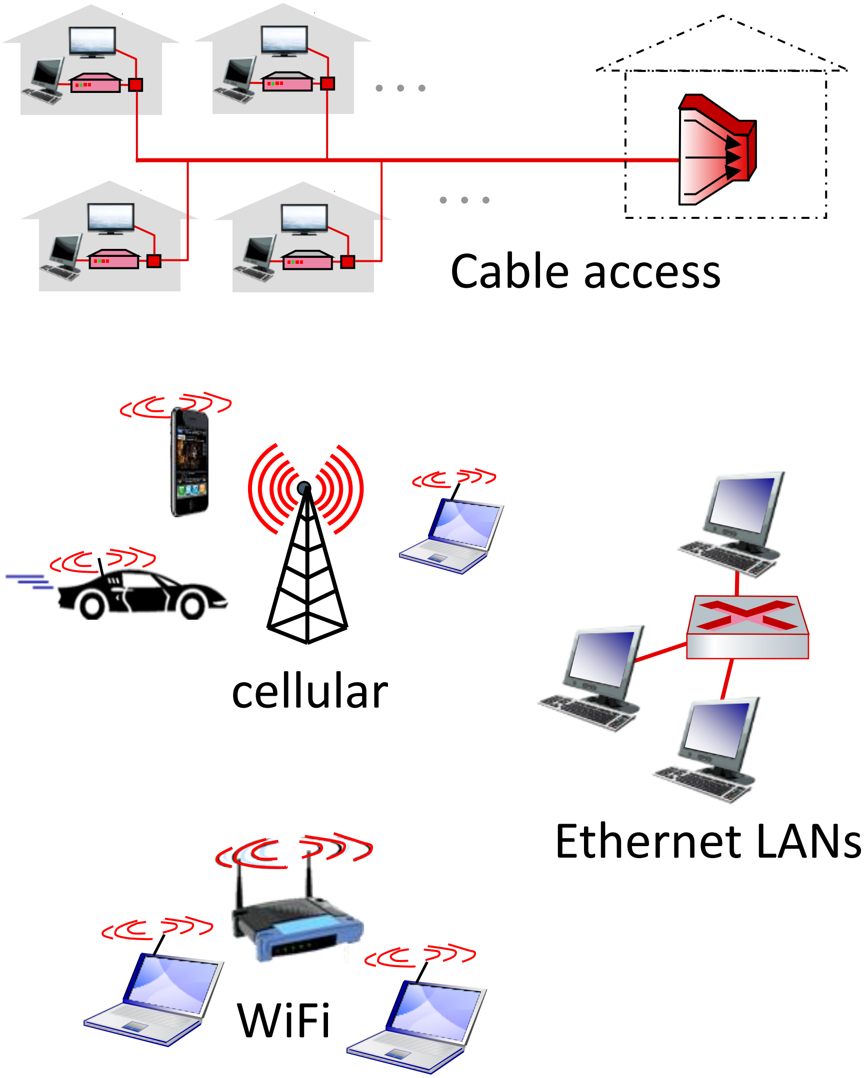
# Link Layer: Services

- 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?



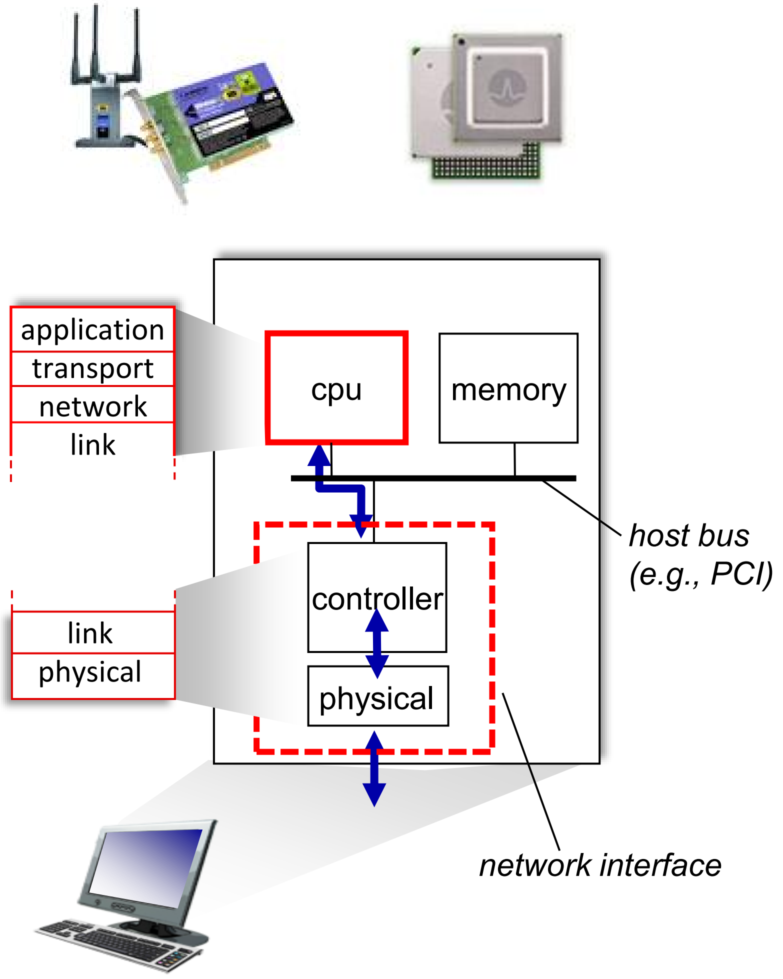
# Link Layer: Services (cont'd)

- 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 OR receive, but not at same time
  - with full duplex, end and receive simultaneously

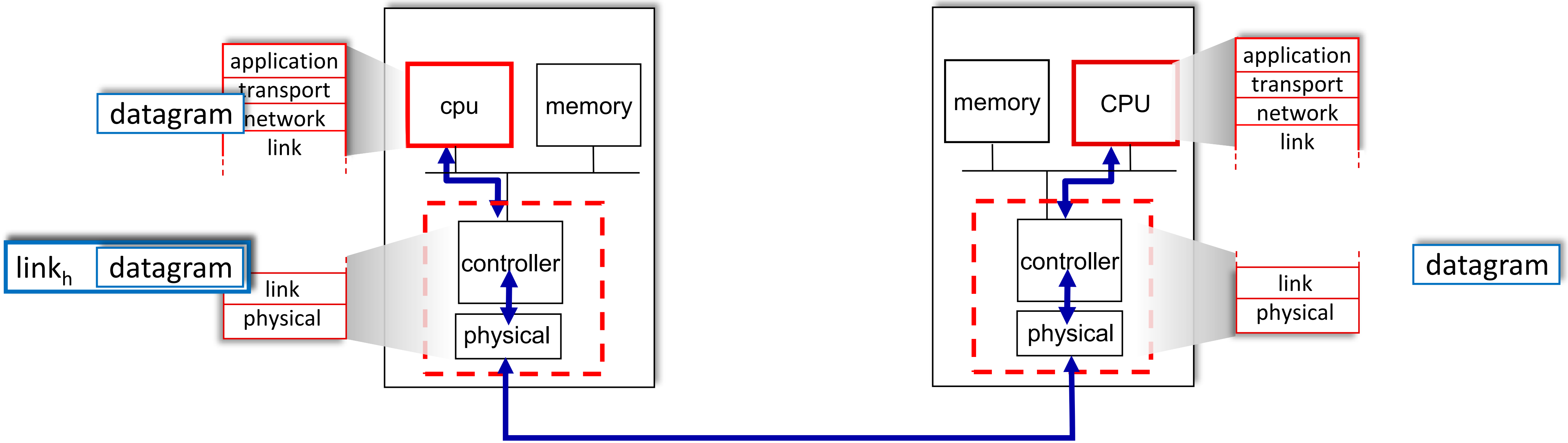


# Host Link-Layer Implementation

- **In each and every host**
- **Link layer implemented in “Adapter” or on a chip**
  - Ethernet card, Wifi 802.11 card, or chipset
  - For link and physical layers
- **Attached into host’s system (motherboard) buses**
  - USB / PCI / Thunderbolt …
- **Hardware / software / firmware**



# Adaptors 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

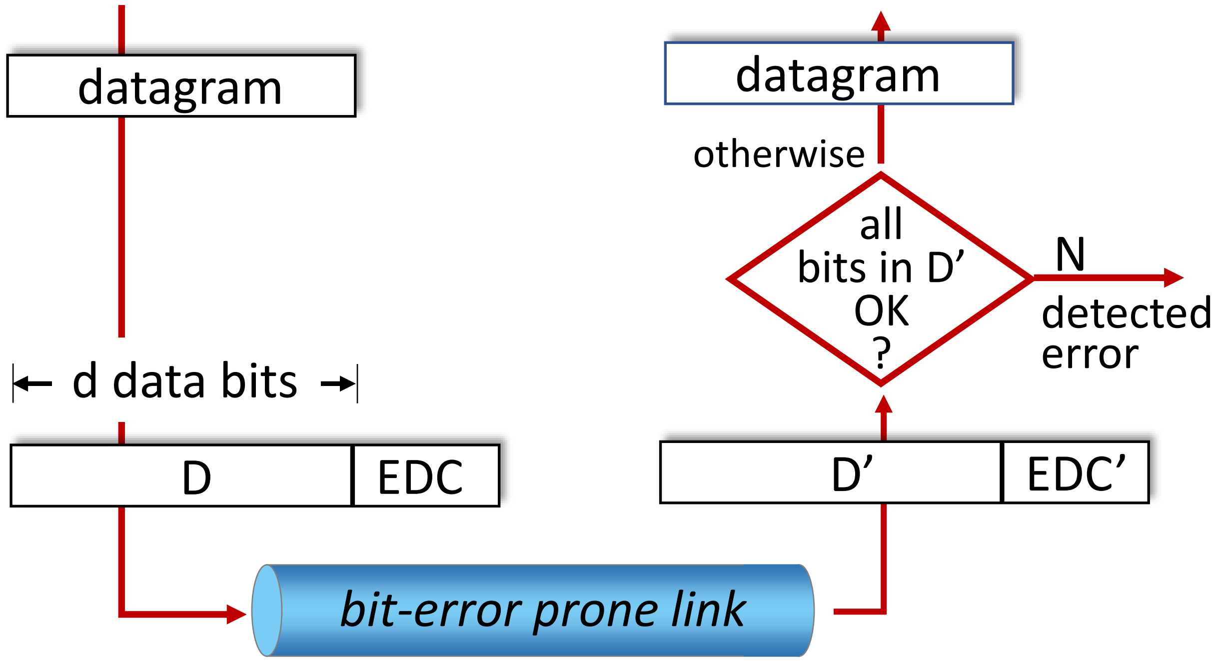
# The Link Layer

- Introduction to the Link Layer
- Error-detection and -correction Techniques
- Multiple Access Links and 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

# Three techniques for detecting errors

- **Parity Checks**
  - basic ideas (simple, detect odd or even numbers of errors)
- **Checksum**
  - used in transport layer ( but link layer can use variants)
- **Cyclic Redundancy Checks (CRC)**
  - used in link layer in an adapter (most powerful)

# Parity Checking

## single bit parity:

- detect single bit errors

| 0111000110101011 | 1 |
|---|---|

← *d* data bits → | parity bit

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

Odd parity: set parity bit so there is an odd number of 1's

### At receiver:

- compute parity of *d+1* received bits, if not even, then error detected
- can detect odd number of bit flips (if two bits flip, it may fail)

## 2-D parity:

row parity →

column parity ↓

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

- detect two-bit errors (adds row and column parity bits)
- detect *and correct* single bit errors without retransmission!

no errors:

$$
\begin{array}{ccccc|c}
1 & 0 & 1 & 0 & 1 & 1 \text{ (even)} \\
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 0 & 1 & 0
\end{array}
$$

(odd)

detected and correctable single-bit error:

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

parity error

parity error

E.g. 1: (Even parity) D=1000001; detect 1-bit error and 2-bit error

E.g. 2: (odd parity) D=4x7 data block; check parity

# Internet Checksum

*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 ....

# **Internet checksum:** 1s complement

## Example: add two 16-bit integers

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| | | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| wraparound | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| sum | | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| checksum | | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

*Note:* wraparound - when adding numbers, a carryout from the most significant bit needs to be added to the result

# Cyclic redundancy check (CRC)

- **more powerful error-detection coding**
- **widely used in practice (Ethernet, 802.11 WiFi, ATM)**
- **all CRC calculations are done in modulo-2 arithmetic (without carries in addition or borrows in subtraction).**
- **modulo-2 Arithmetic**
  - Addition
  - Subtraction
  - Bitwise exclusive-or (XOR)

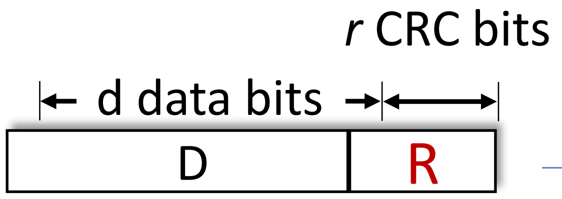
1011 + 0101 = ?

1011 - 0101 = ?

1011 XOR 0101= ?

# 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)




bits to send

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

*sender:* Given D and G(r+1) bits, append r bits R such that $D \cdot 2^r$ *XOR R is divisible by G* (mod 2)

*receiver:* knows G, divides <D,R> by G. If remainder not equal to zero, error detected!

- can detect single-bit-error, double-bit error, 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:

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

... 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
```

# The Link Layer

- Introduction to the Link Layer
- Error-detection and -correction Techniques
- Multiple Access Links and Protocols

# Multiple Access Links, Protocols

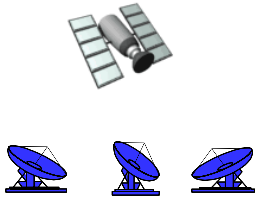
two types of "links":

- point-to-point
  - point-to-point link between Ethernet switch, host
  - PPP for dial-up access
- broadcast (shared wire or medium)
  - old-school Ethernet
  - upstream HFC in cable-based access network
  - 802.11 wireless LAN, 4G/5G, 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

*Requirements:*

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

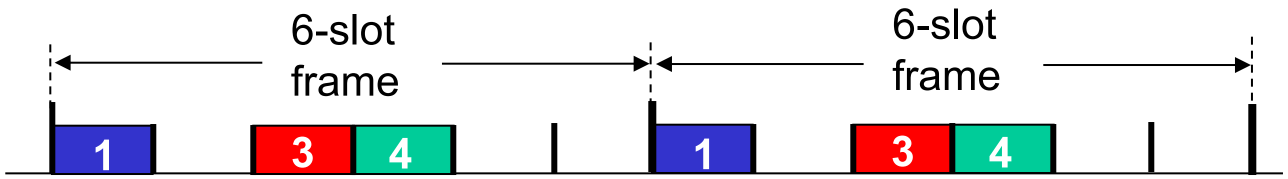
three broad classes:

- 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

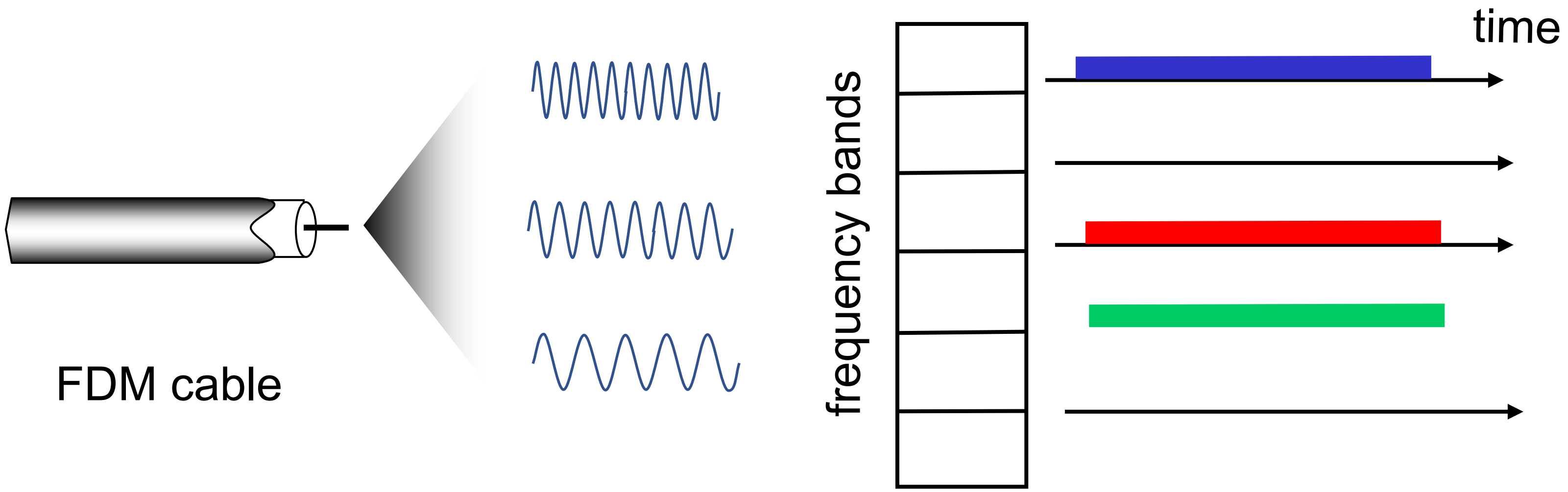
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



# 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

# 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!

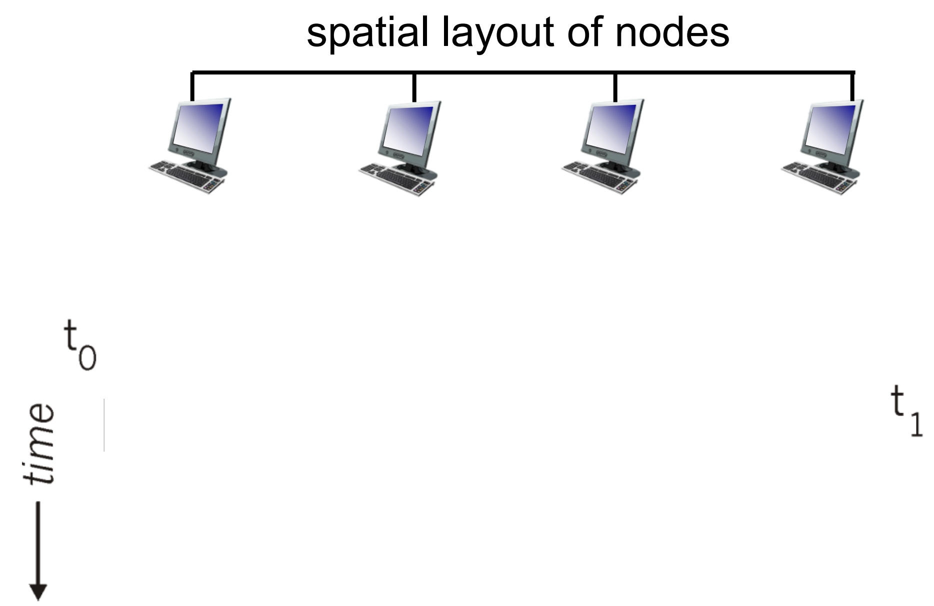
CSMA/CD: CSMA with *collision detection*

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

- human analogy: the polite conversationalist

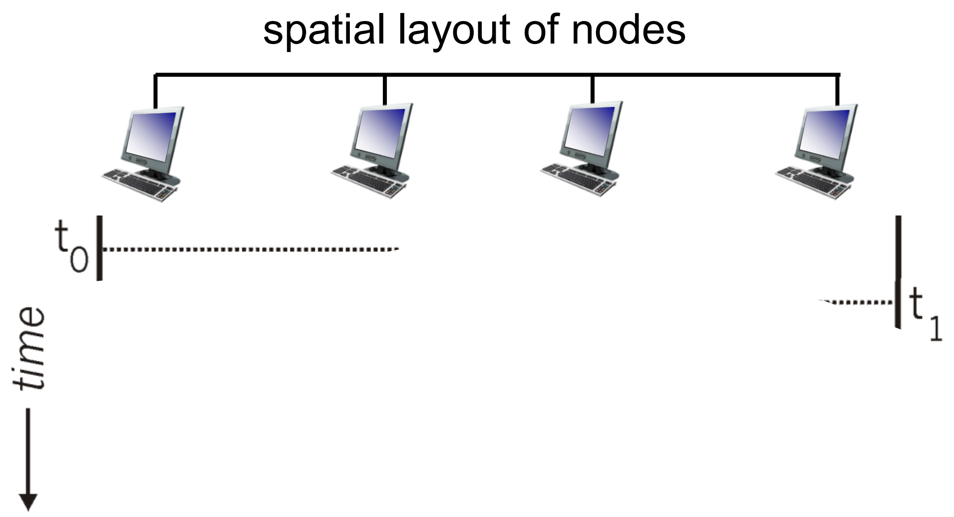
# CSMA: Collisions

- 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 determining collision probability



# CSMA/CD:

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



# “Taking Turns” MAC Protocols

channel partitioning MAC protocols:

- **Pro:** share channel *efficiently* and *fairly* at high load
- **Con:** inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!

random access MAC protocols

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

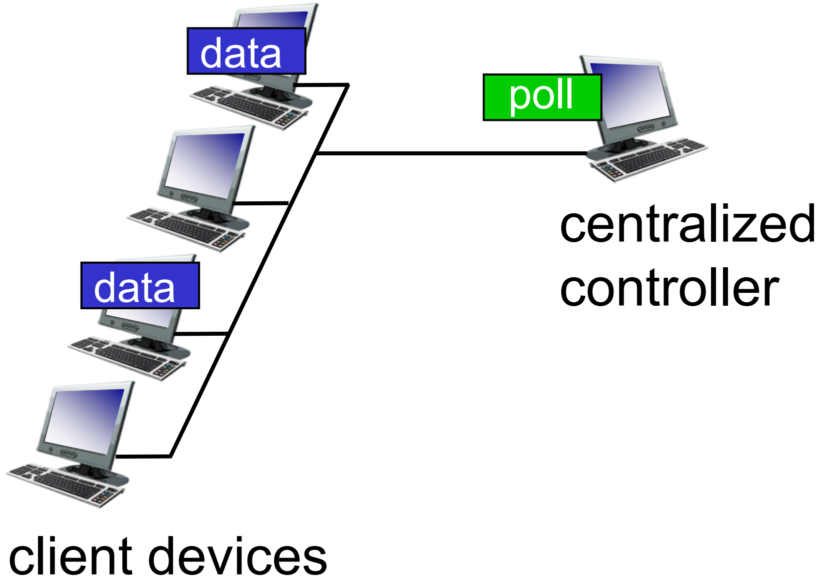
“taking turns” protocols

- look for best of both worlds!

# "Taking Turns" MAC Protocols (cont'd)

polling:

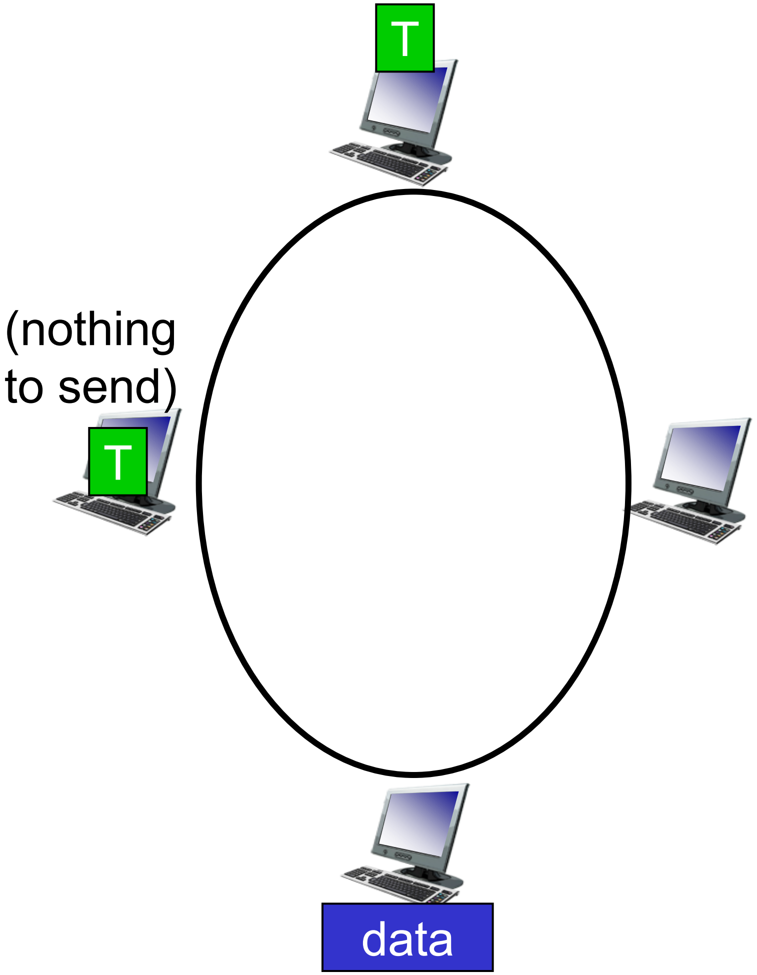
- centralized controller "invites" other nodes to transmit in turn
- typically used with "dumb" devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)
- Bluetooth uses polling



# “Taking Turns” MAC Protocols (cont’d)

token passing:

- control *token* message explicitly passed from one node to next, sequentially
  - transmit while holding token
- concerns:
  - token overhead
  - latency
  - single point of failure (token)



# Link layer, LANs: Roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs

# MAC Addresses

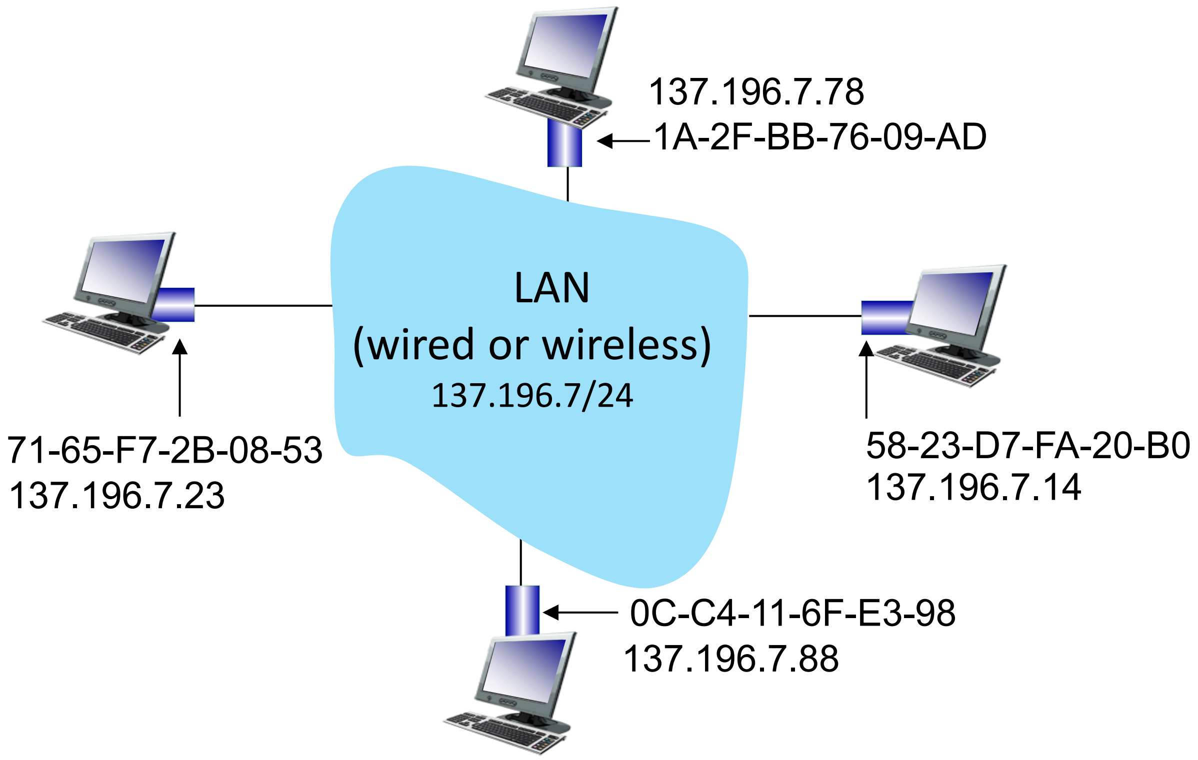
- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
  - e.g.: 128.119.40.136
- MAC (or LAN or physical or Ethernet) address:
  - function: used "locally" to get frame from one interface to another physically-connected interface (same subnet, in IP-addressing sense)
  - 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 Addresses

each interface on LAN

- has unique 48-bit MAC address
- has a locally unique 32-bit IP address (as we've seen)

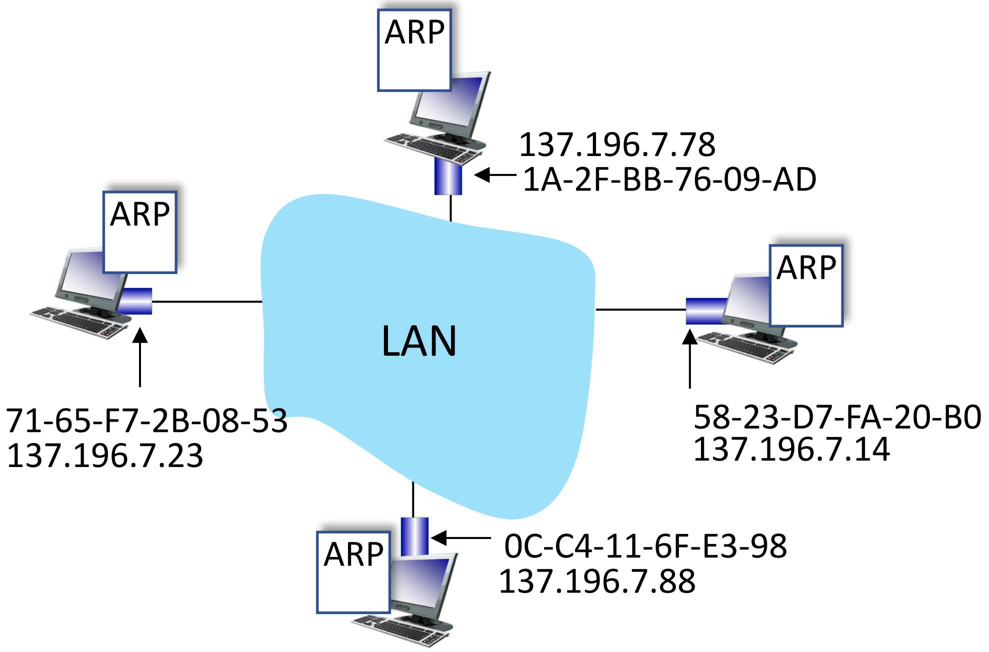


# MAC Addresses

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
- MAC flat address: portability
  - can move interface from one LAN to another
  - recall IP address *not* portable: depends on IP subnet to which node is attached

# ARP: Address Resolution Protocol

*Question:* how to determine interface's MAC address, knowing its IP address?




ARP table: each IP node (host, router) on LAN has table

- IP/MAC address mappings for some LAN nodes:

  < IP address; MAC address; TTL>

- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

# ARP Protocol in Action

## example: A wants to send datagram to B

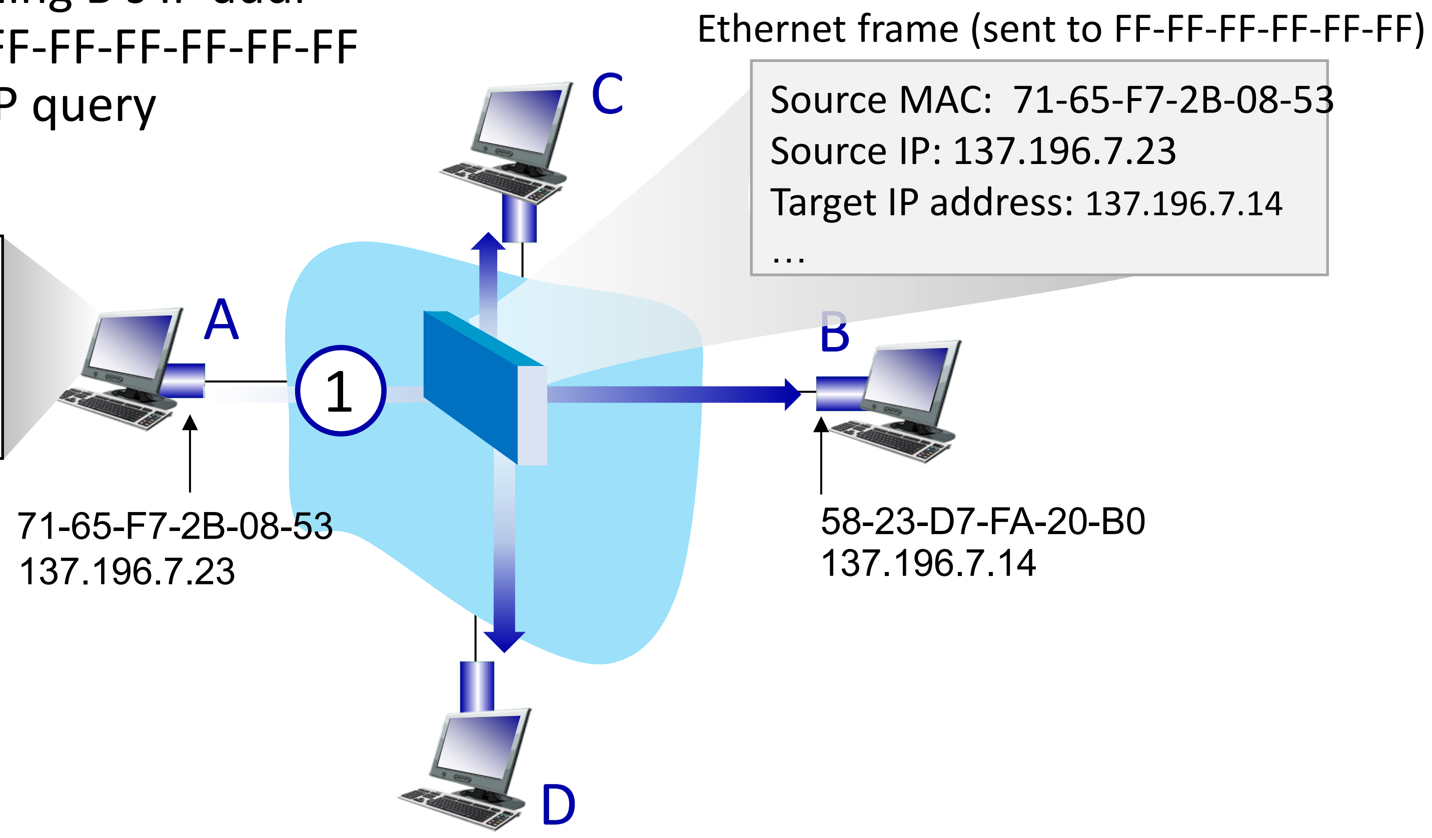
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

1 A broadcasts ARP query, containing B's IP addr

- destination MAC address = FF-FF-FF-FF-FF-FF
- all nodes on LAN receive ARP query

ARP table in A

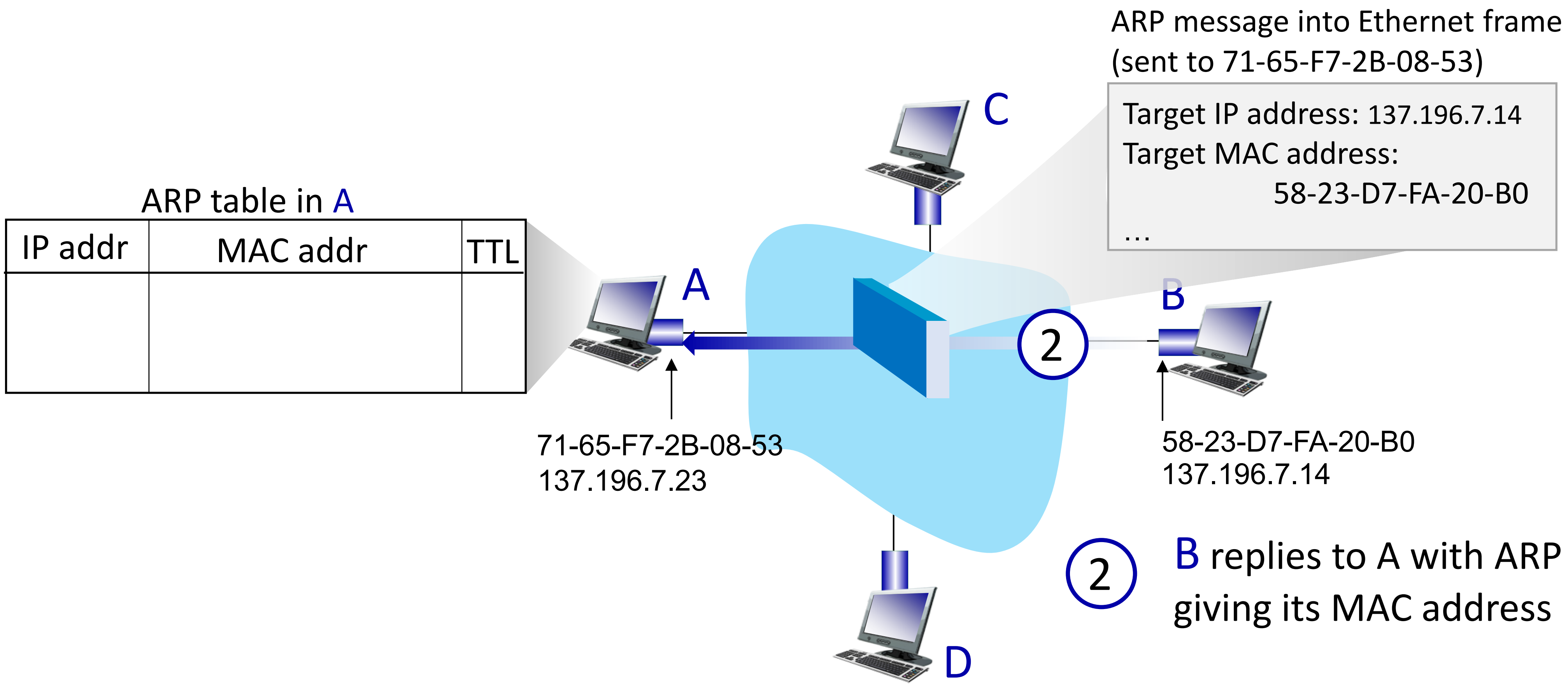
| IP addr | MAC addr | TTL |
| --- | --- | --- |
| | | |



# ARP protocol in Action (cont'd)

example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address




2 B replies to A with ARP response, giving its MAC address

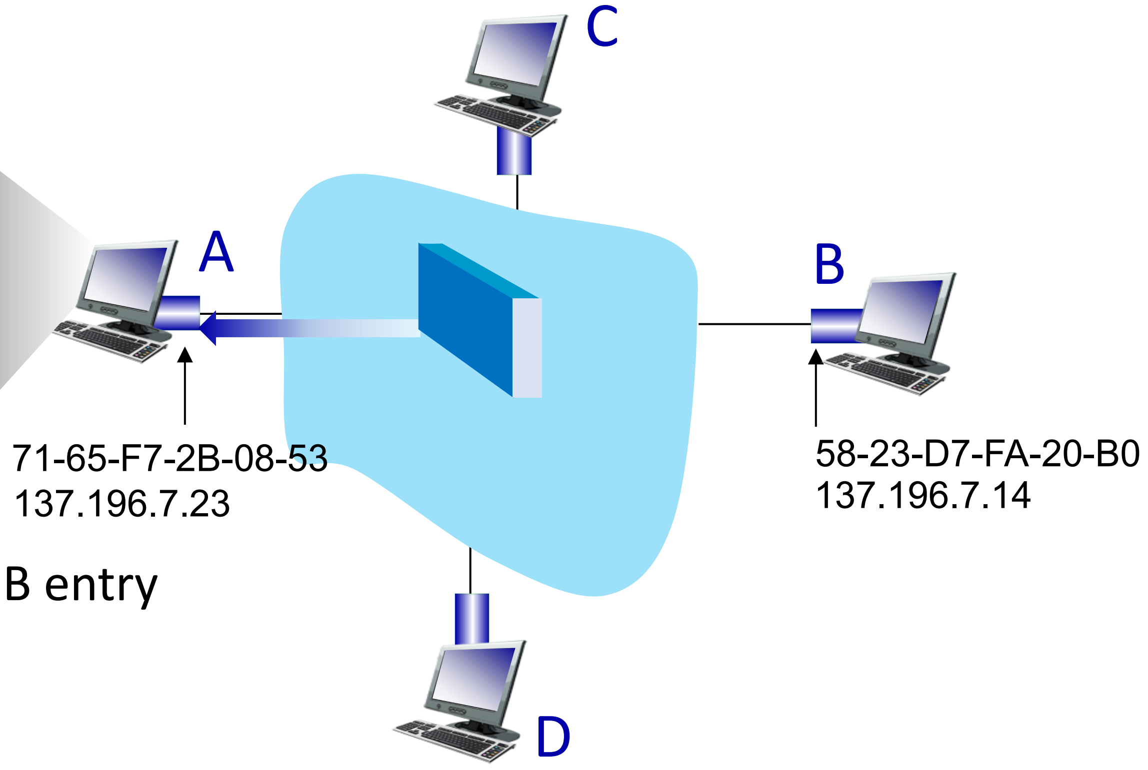
# ARP protocol in Action (cont'd)

## example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

ARP table in A

| IP addr | MAC addr | TTL |
|---|---|---|
| 137.196.7.14 | 58-23-D7-FA-20-B0 | 500 |

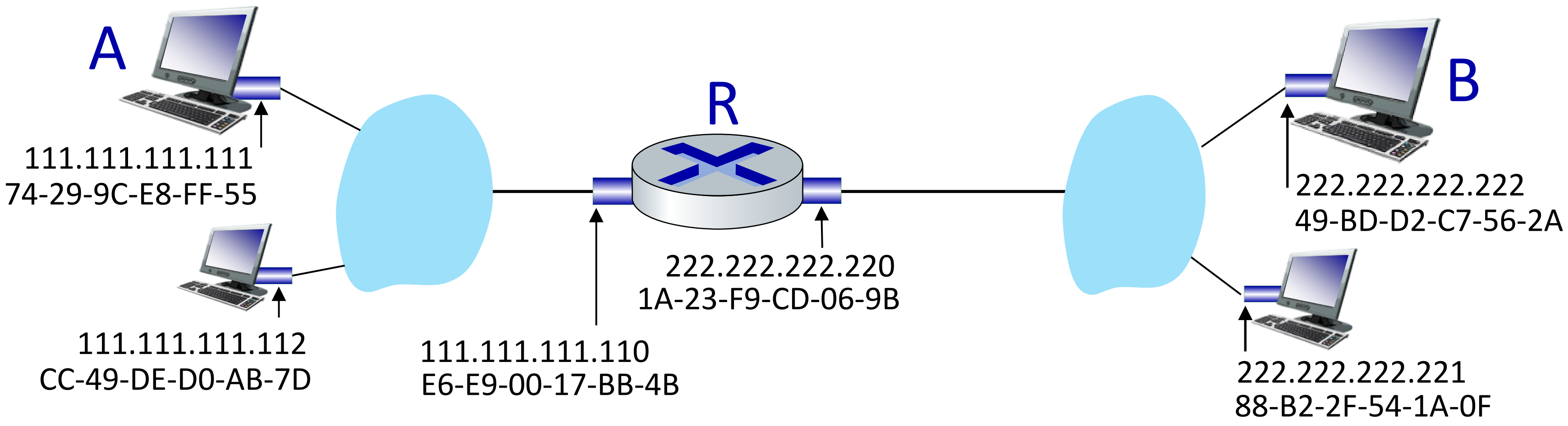



3 A receives B's reply, adds B entry into its local ARP table

# Routing to Another Subnet: Addressing

walkthrough: sending a datagram from *A* to *B* via *R*

- focus on addressing – at IP (datagram) and MAC layer (frame) levels
- assume that:
  - A knows B's IP address
  - A knows IP address of first hop router, R (how?)
  - A knows R's MAC address (how?)



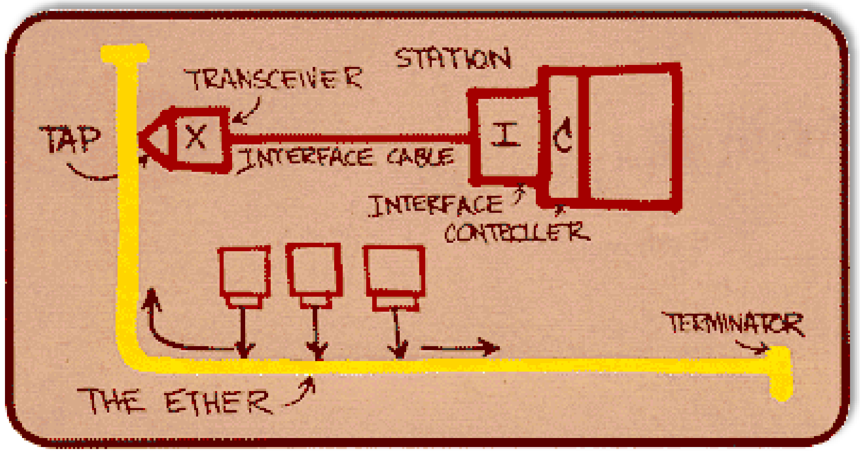
# Link layer, LANs: Roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs

# Ethernet

"dominant" wired LAN technology:

- first widely used LAN technology
- simpler, cheap
- kept up with speed race: 10 Mbps – 400 Gbps
- single chip, multiple speeds (e.g., Broadcom BCM5761)

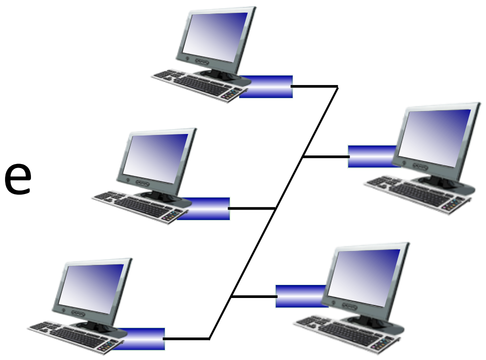


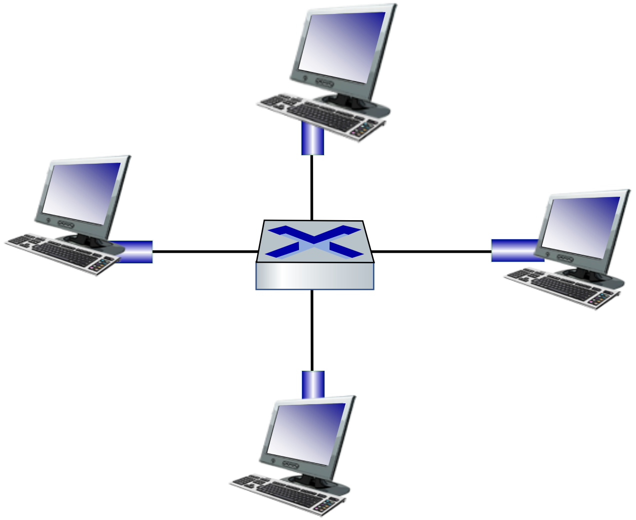

*Metcalfe's Ethernet sketch*

# Ethernet: Physical Topology

- bus: popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- switched: prevails today
  - active link-layer 2 *switch* in center
  - each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other)

bus: coaxial cable





switched

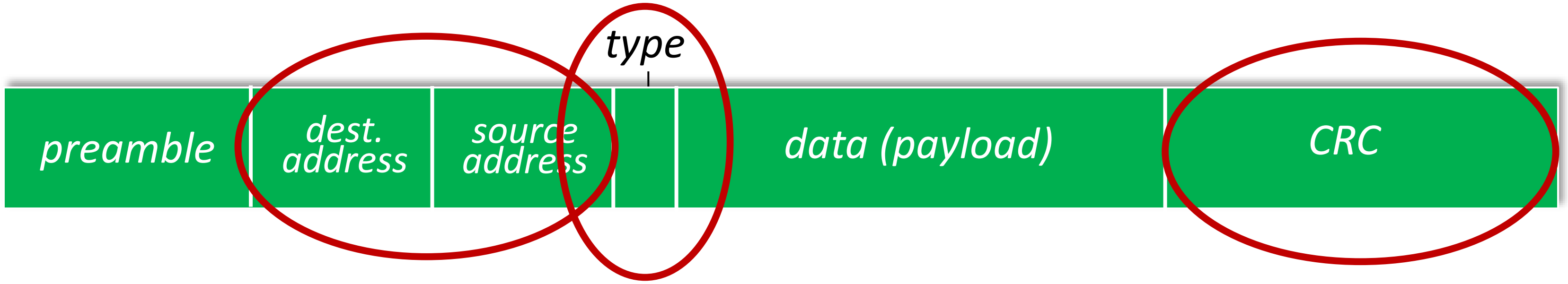
# Ethernet Frame Structure

sending interface encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame


*preamble:*

- used to synchronize receiver, sender clock rates
- 7 bytes of 10101010 followed by one byte of 10101011

# Ethernet Frame Structure (cont'd)




- addresses: 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (e.g., ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- type: indicates higher layer protocol
  - mostly IP but others possible, e.g., Novell IPX, AppleTalk
- CRC: cyclic redundancy check at receiver
  - error detected: frame is dropped

# Ethernet: Unreliable, Connectionless

- connectionless: no handshaking between sending and receiving NICs
- unreliable: receiving NIC does not send ACKs or NAKs to sending NIC
  - data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost
- Ethernet's MAC protocol: unslotted CSMA/CD with binary backoff