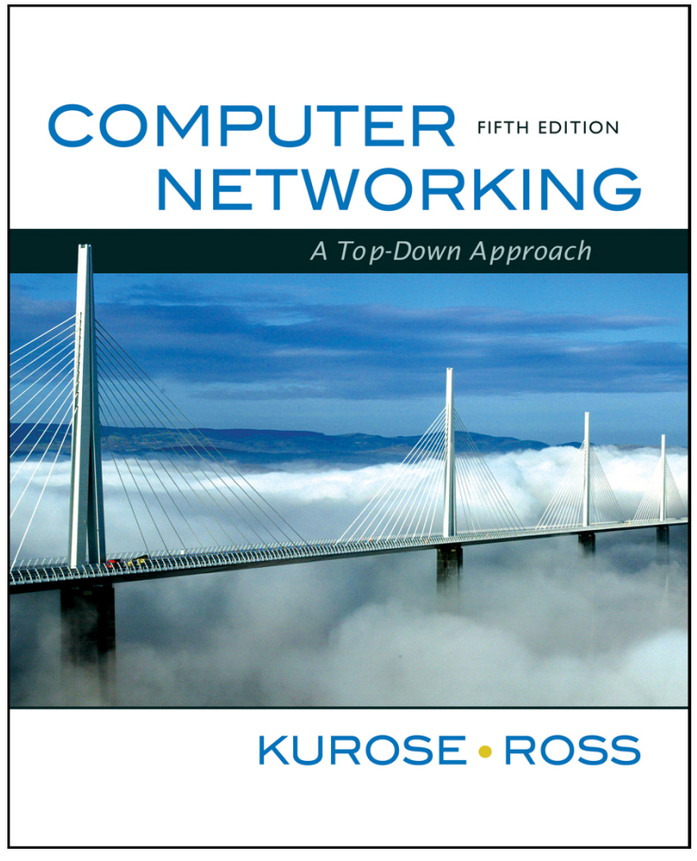
Module 4
Data Link Layer

Please note: Most of these slides come from this book. Note their copyright notice below…



A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you 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) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form 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.

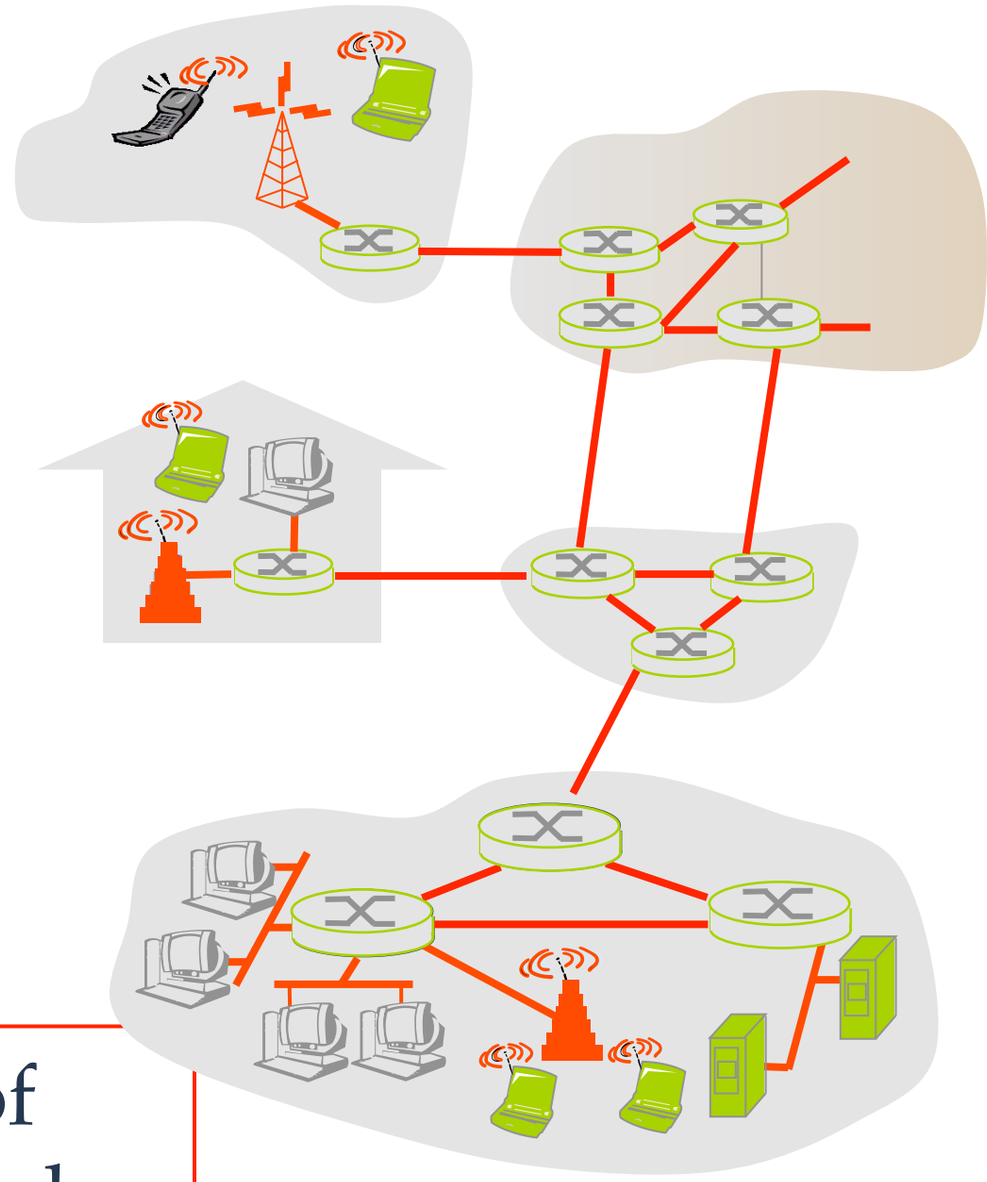
Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach 5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.

Link Layer: Introduction

Terminology:

- hosts and routers are nodes
- communication channels that connect adjacent nodes along communication path are links
  - ➡ wired links
  - ➡ wireless links
  - ➡ LANs
- layer-2 packet is a frame, encapsulates datagram



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

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

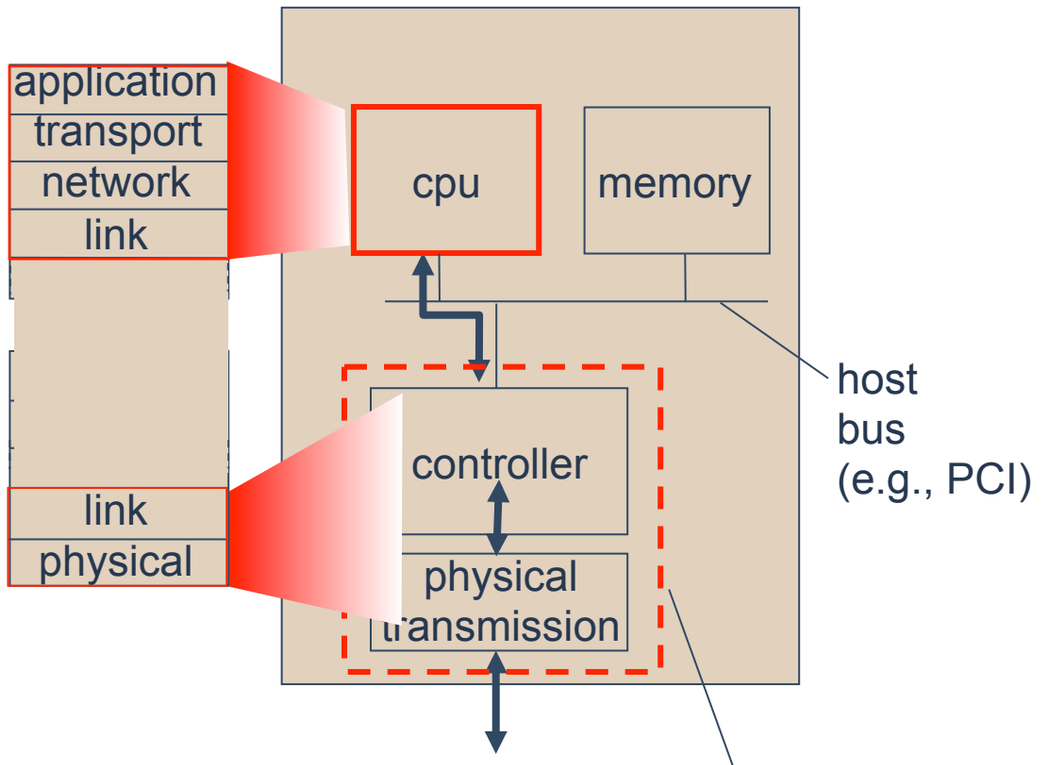
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - Similar techniques to transport layer – ack and retransmit
  - 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 (2)

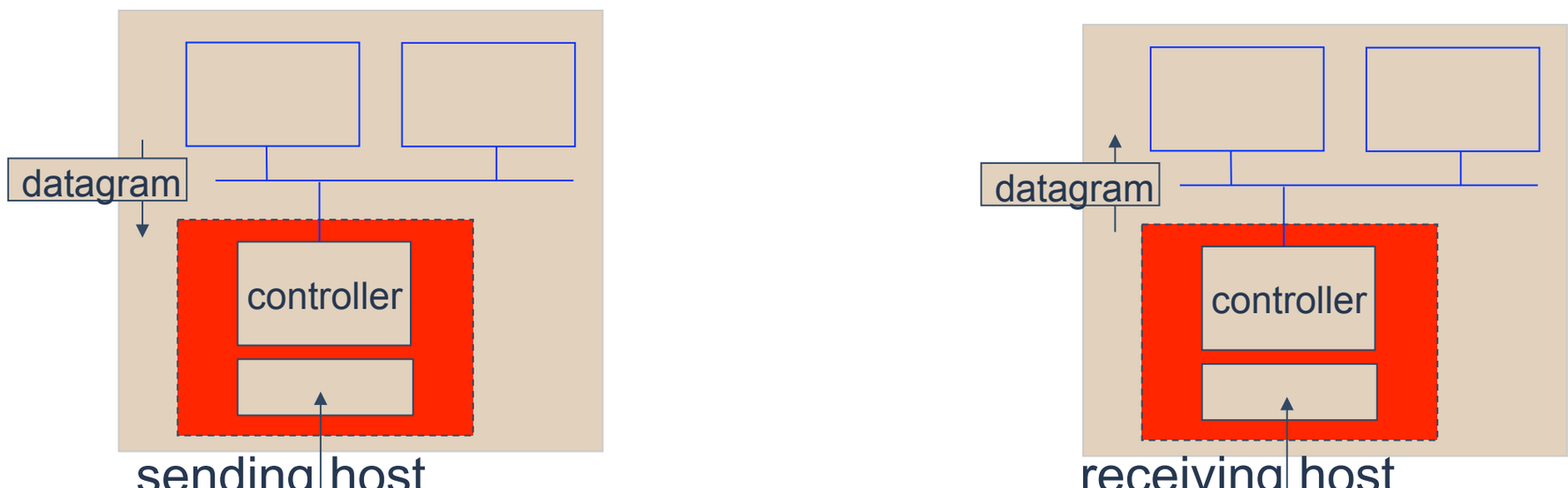
- *flow control:*
  - ➡ pacing between adjacent sending and receiving nodes
- *error detection*:
  - ➡ errors caused by signal attenuation, noise.
  - ➡ receiver detects presence of errors:
    - ✦ signals sender for retransmission or drops frame
- *error correction*:
  - ➡ receiver identifies *and corrects* 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

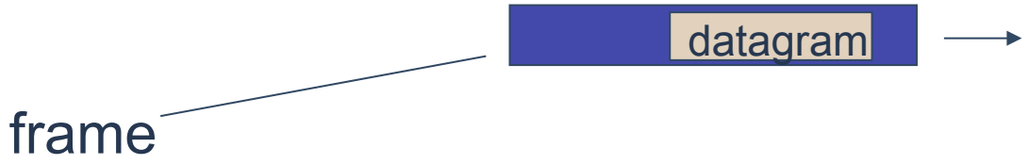
Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC)
  - ➡ Ethernet card, PCMCI card, 802.11 card
  - ➡ implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



Adaptors Communicating



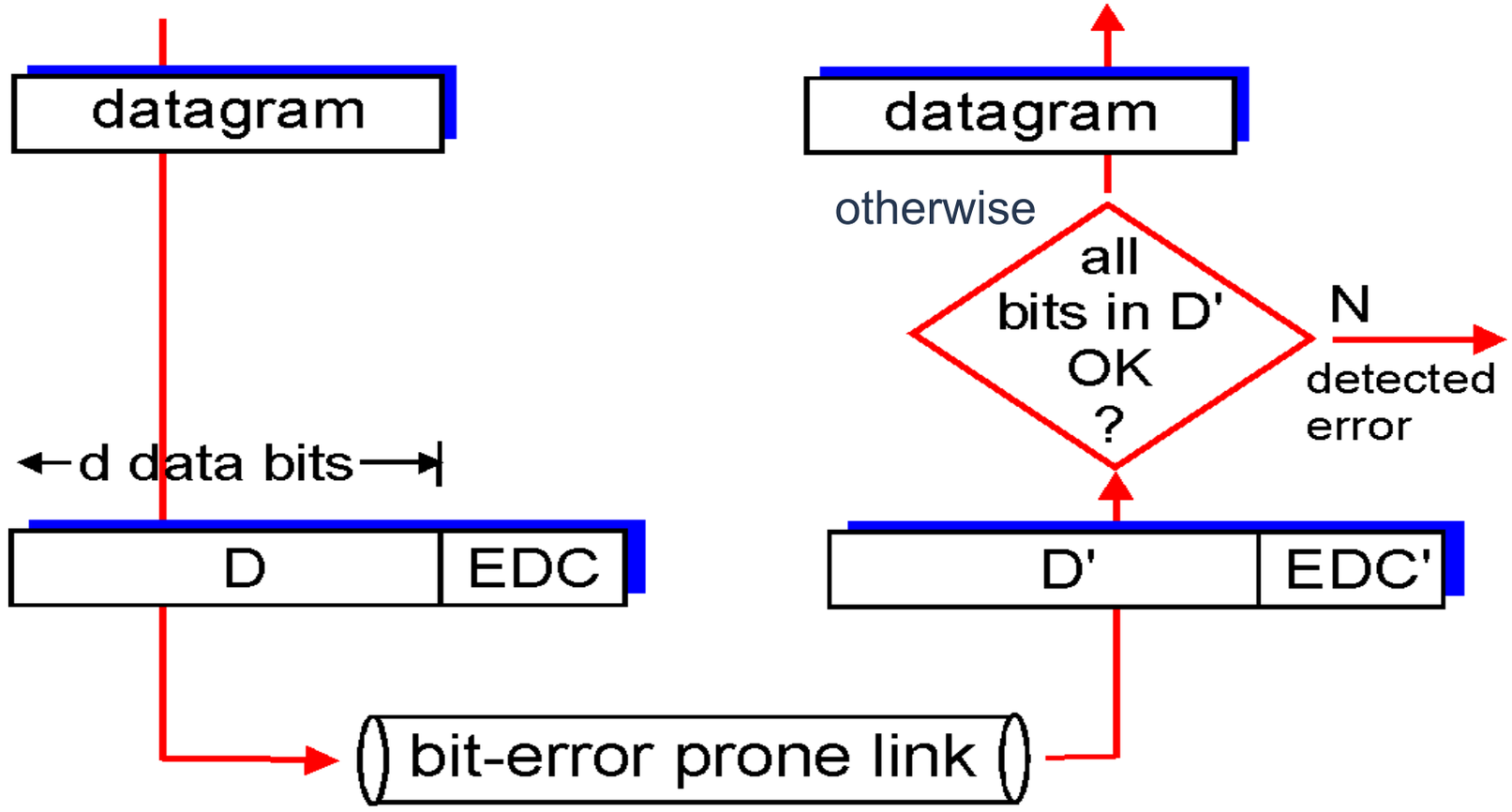


- sending side:
  - ➡ encapsulates datagram in frame
  - ➡ adds error checking bits, rdt, flow control, etc.
- receiving side
  - ➡ looks for errors, rdt, flow control, etc
  - ➡ extracts datagram, passes to upper layer at receiving side

Error Detection

EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

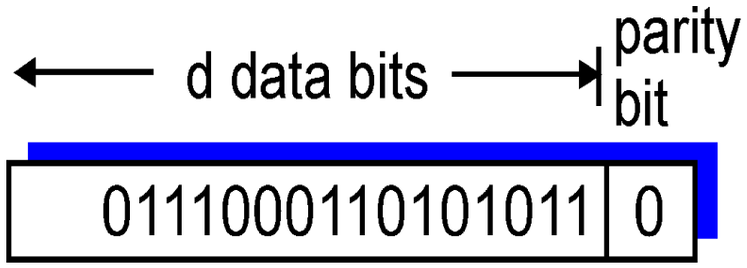
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



Parity Checking

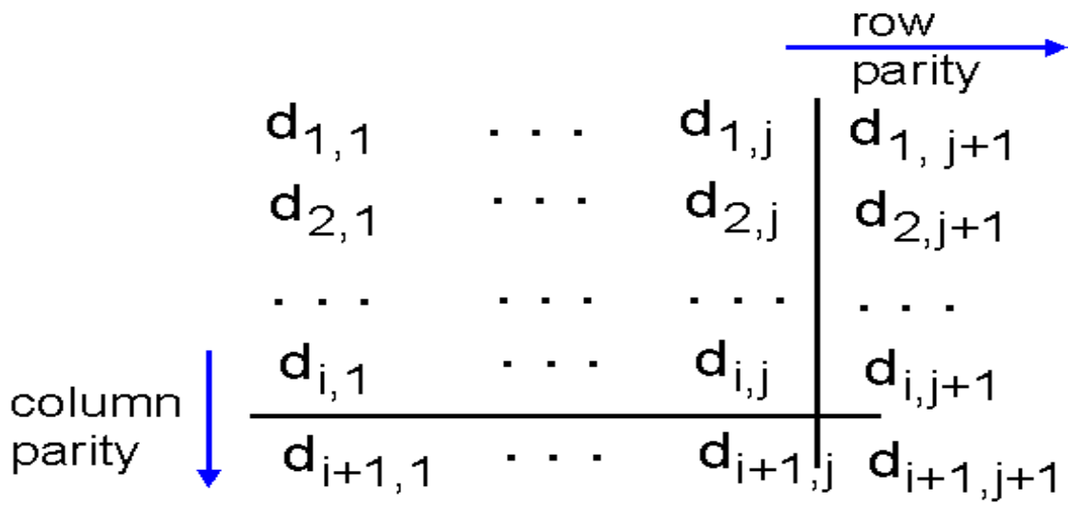
Single Bit Parity:

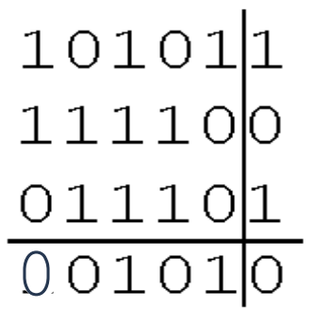
Detect single bit errors



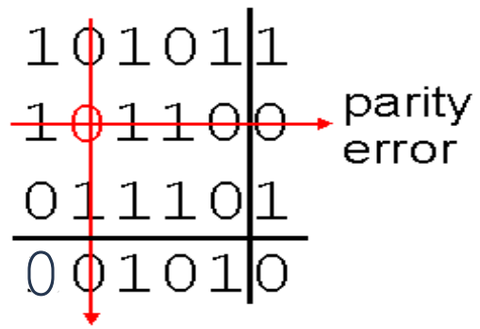
Two Dimensional Bit Parity:

Detect and correct single bit errors





no errors



correctable single bit error

Internet checksum (review)

Goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer only)

Sender:

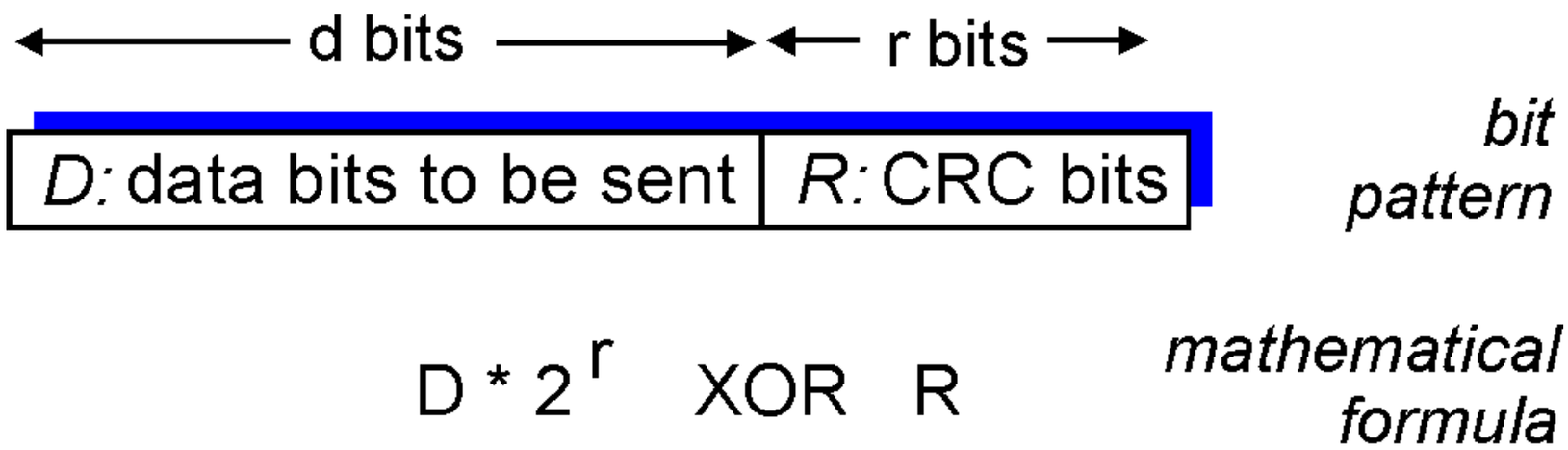
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

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?*

Checksumming: Cyclic Redundancy Check

- 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!
  - ➡ can detect all burst errors less than *r*+1 bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)



CRC Example

Want:

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

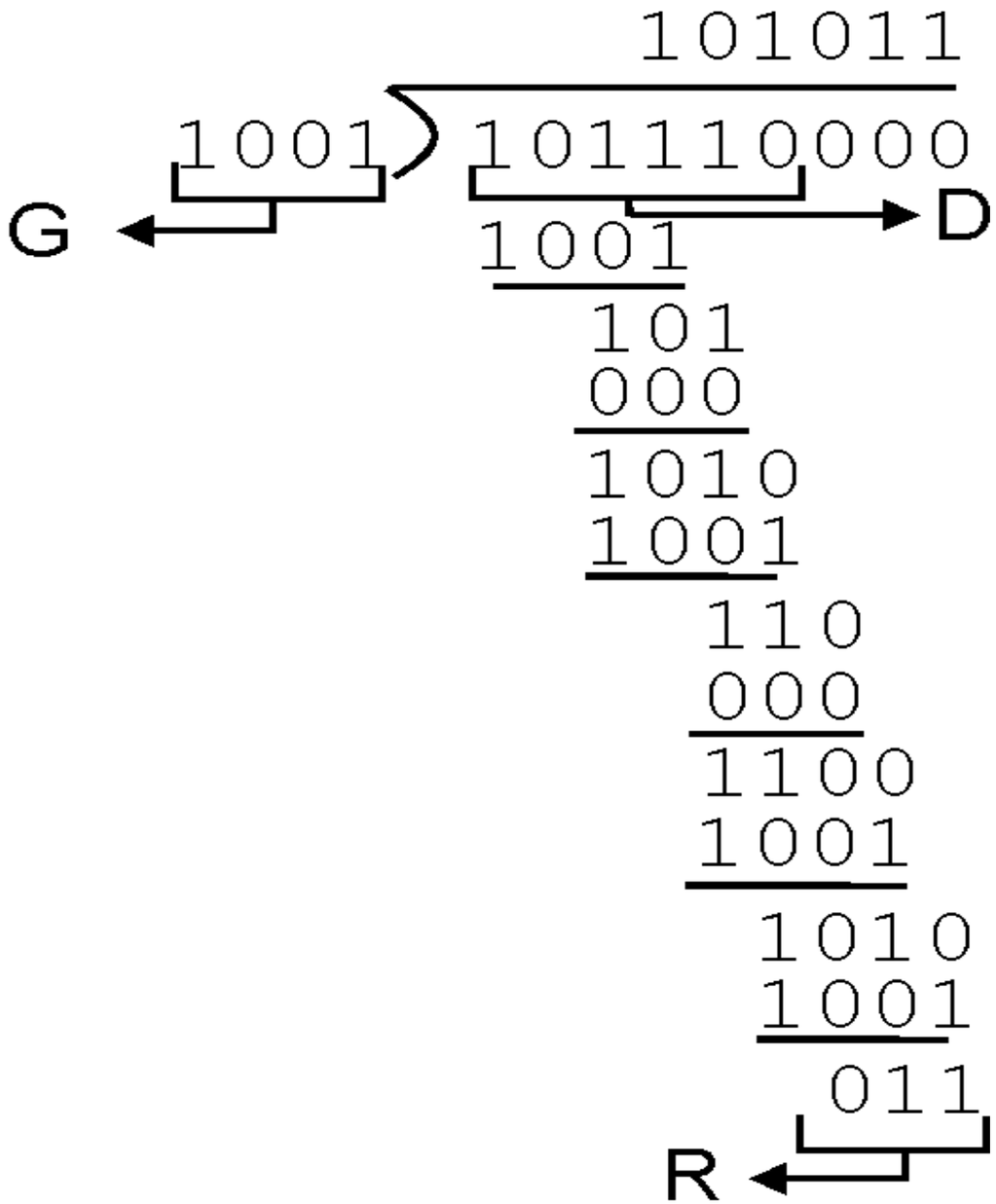
equivalently:

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

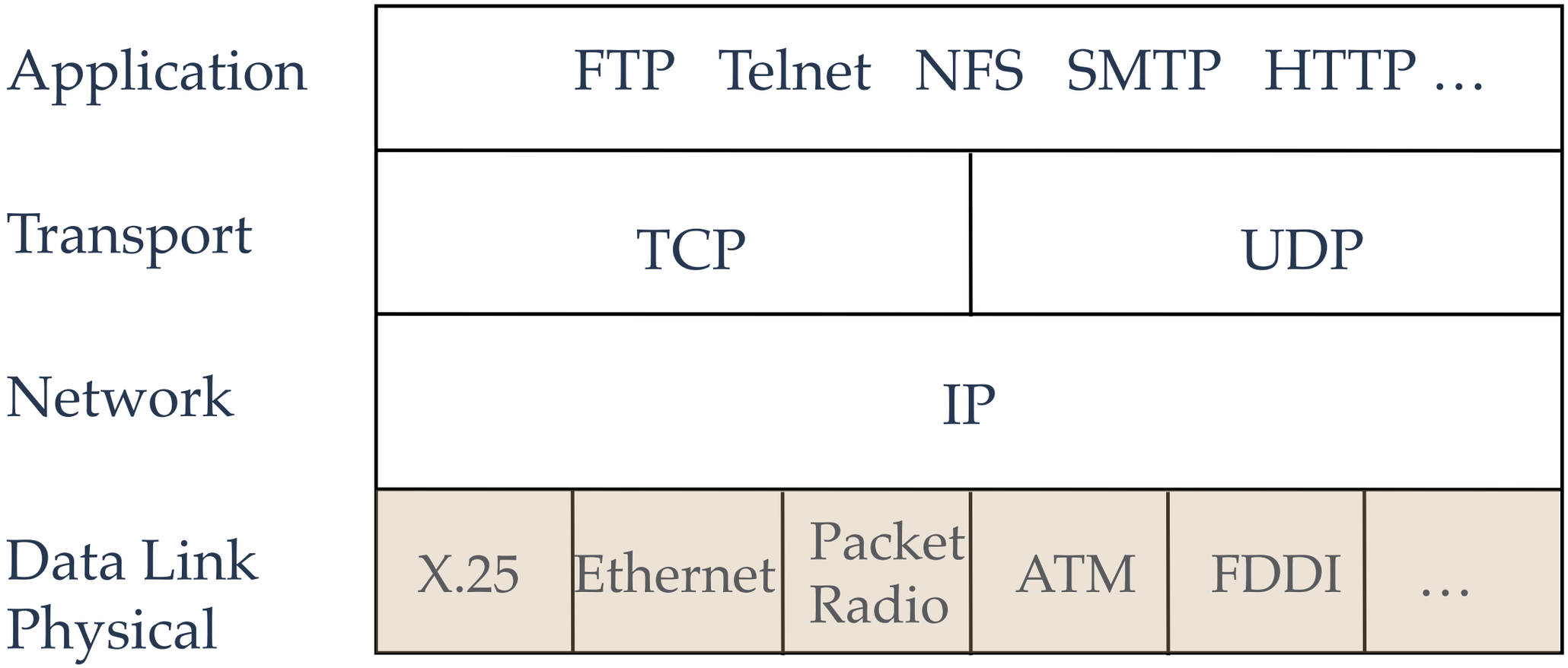
equivalently:

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

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



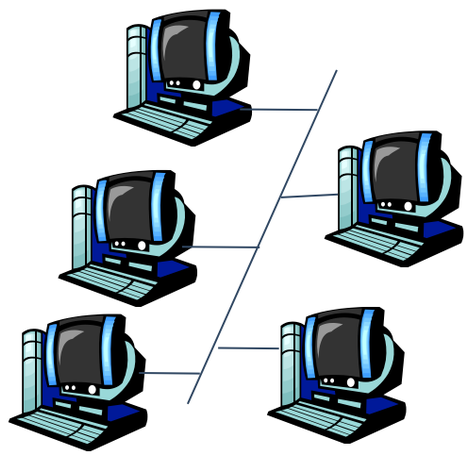
Internet Protocols



Multiple Access Links and Protocols

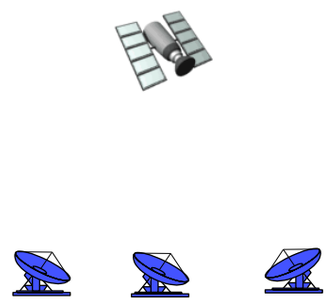
Two types of "links":

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

Ideal Multiple Access Protocol

Broadcast channel of rate R bps

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: a taxonomy

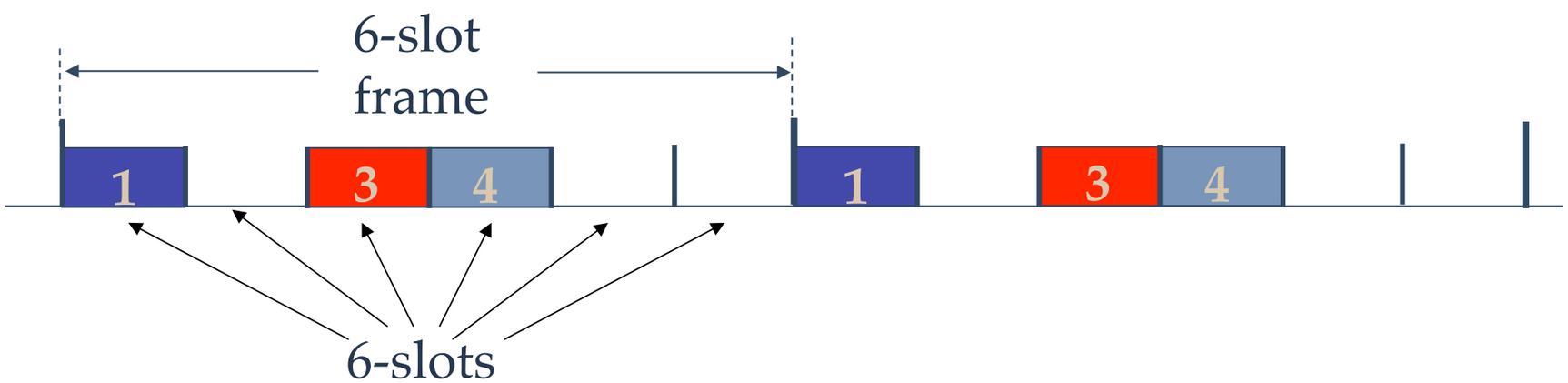
Three broad classes:

- 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

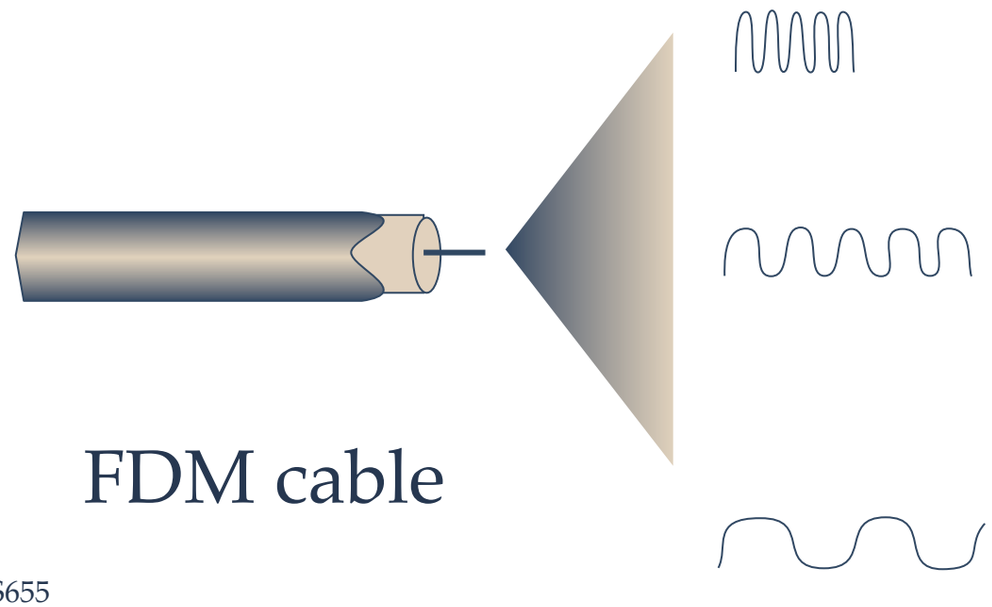
- Access to channel in "rounds"
- Each station gets fixed length slot (length = single frame transmission time) in each round
- Unused slots go idle
- Example: 6-station LAN; 1, 3, 4 have frames, slots 2, 5, 6 idle

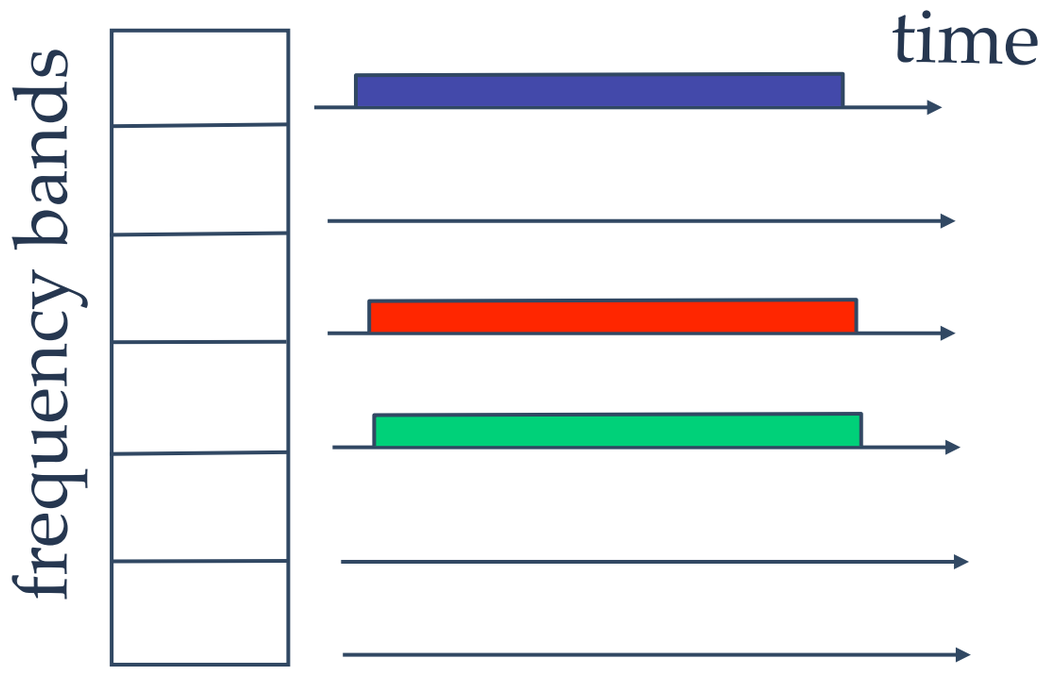


Channel Partitioning MAC Protocols: FDMA

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 frames, 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
- When collisions occur, keep retransmitting
  - Don't retransmit immediately, transmit after a *random* delay
- 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

CSMA (Carrier Sense Multiple Access)

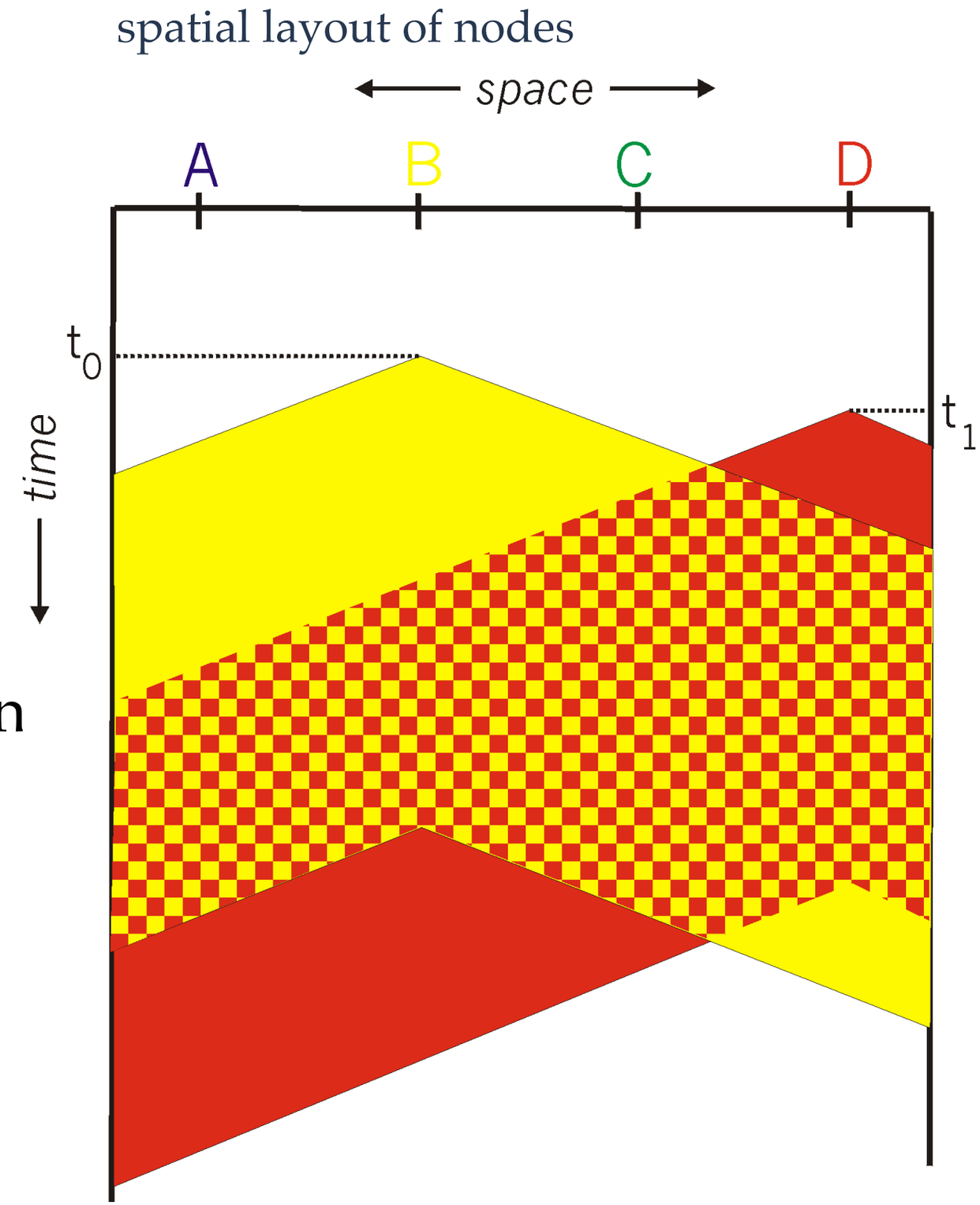
CSMA: listen before transmitting:

- If channel sensed idle, transmit entire frame
- If channel sensed busy, defer transmission for a random amount of time and then sense again

- 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
- Role of distance & propagation delay in determining collision probability



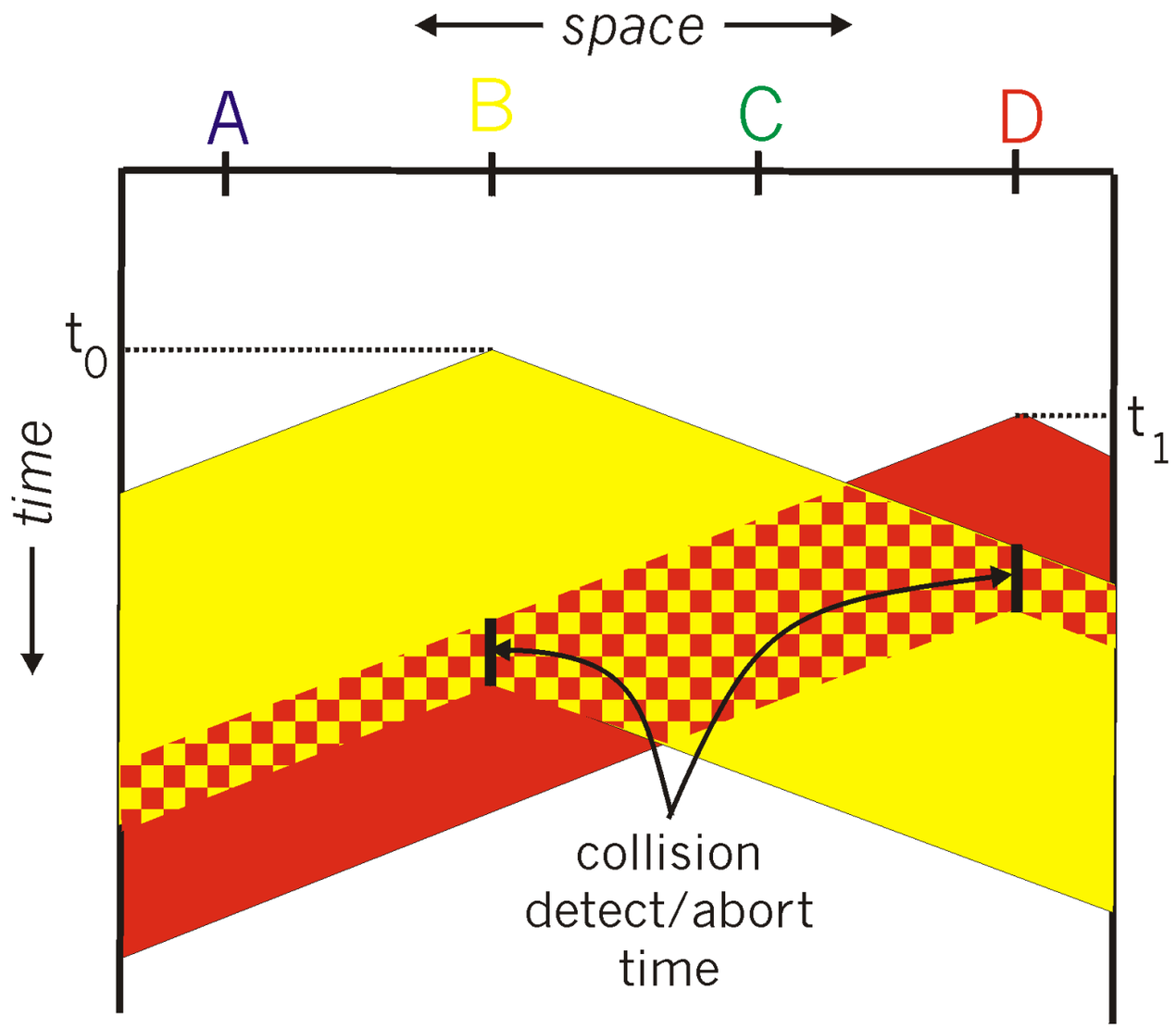
CSMA/CD (Collision Detection)

CSMA/CD: Carrier Sense Multiple Access/Collision Detection

- Carrier sensing, deferral as in CSMA
- Listen while you transmit
  - 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

CSMA/CD Collision Detection



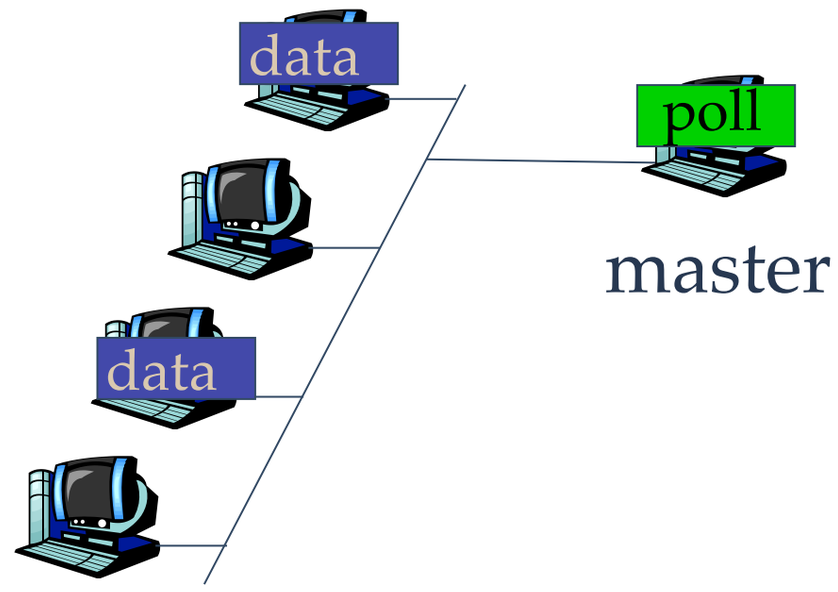
Taking Turns MAC protocols

- Taking turns protocols encapsulate the best of channel partitioning and random access protocols
  - ➡ Channel partitioning 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 protocols
    - ✦ Efficient at low load: single node can fully utilize channel
    - ✦ High load: collision overhead
  - ➡ Take turns and use the entire channel when it is your turn
    - ✦ There are many alternatives
    - ✦ Bluetooth, FDDI, IBM Token Ring

Taking Turns MAC Protocols

Polling:

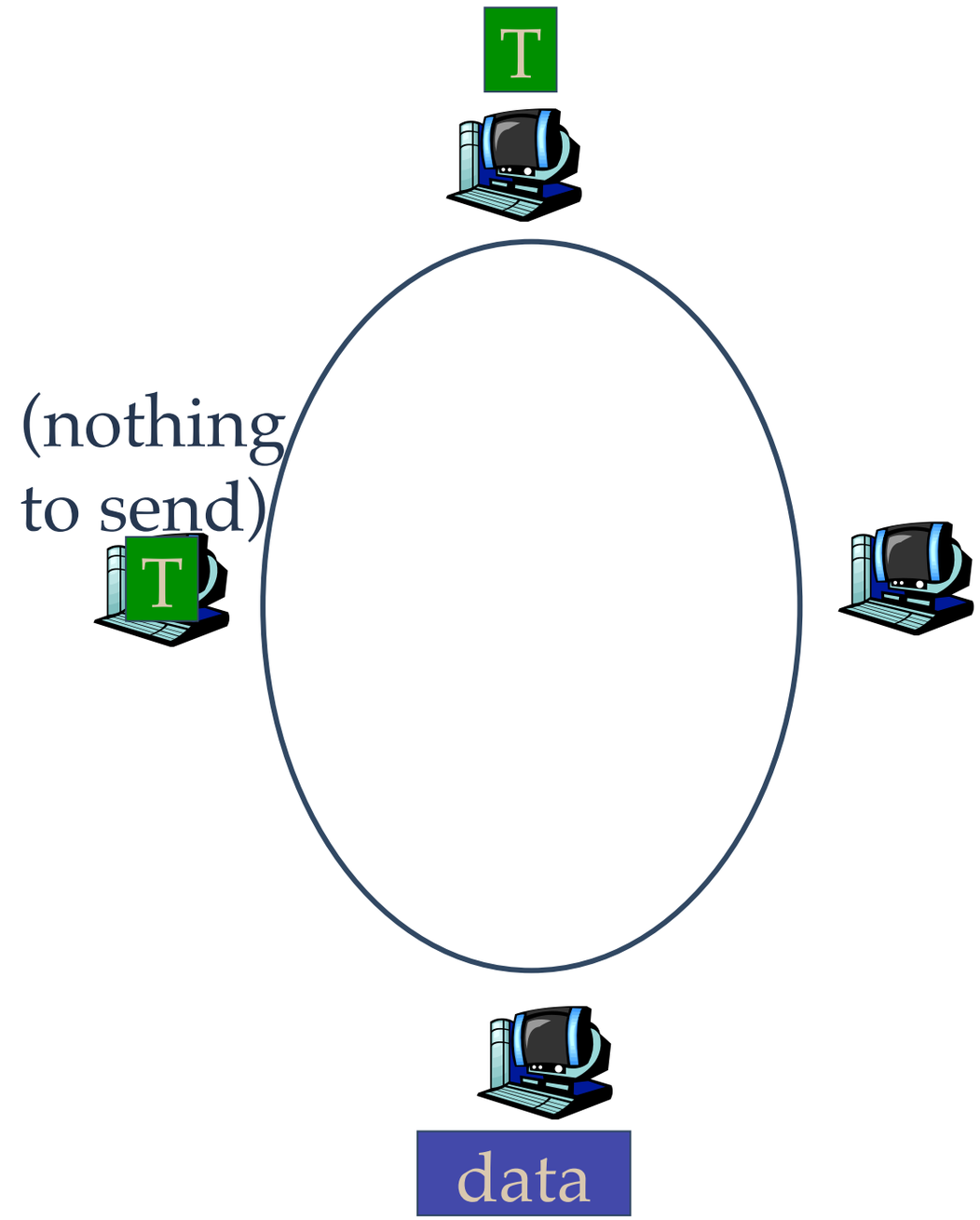
- Master node "invites" slave nodes to transmit in turn
- Typically used with "dumb" slave devices
- Concerns:
  - polling overhead
  - latency
  - single point of failure (master)



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)

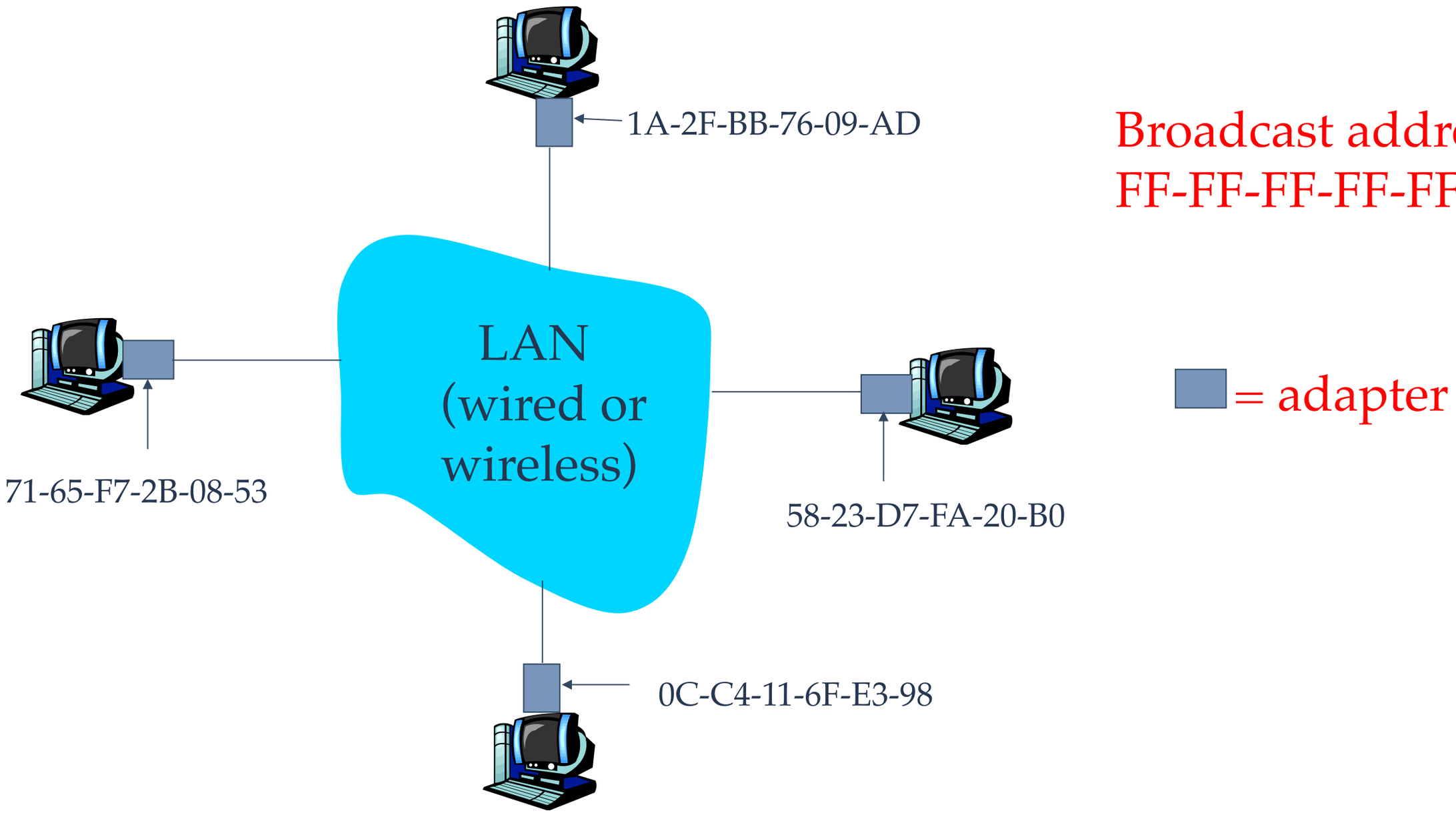


MAC Addresses and ARP

- Nodes (hosts and routers) have link-layer addresses
  - Strictly speaking, the node's adapter has a link-layer address
  - Thee are known as MAC addresses
- MAC (or LAN or physical or Ethernet) address:
  - Function: *get frame from one interface to another physically-connected interface (same network)*
  - 48 bit MAC address (for most LANs)
    - Burned in NIC ROM, also sometimes software settable
- Contrast this with network-layer address (32-bit IP address):
  - *Network-layer* address
  - Used to get datagram to destination IP subnet

LAN Addresses and ARP

Each adapter on LAN has unique LAN address

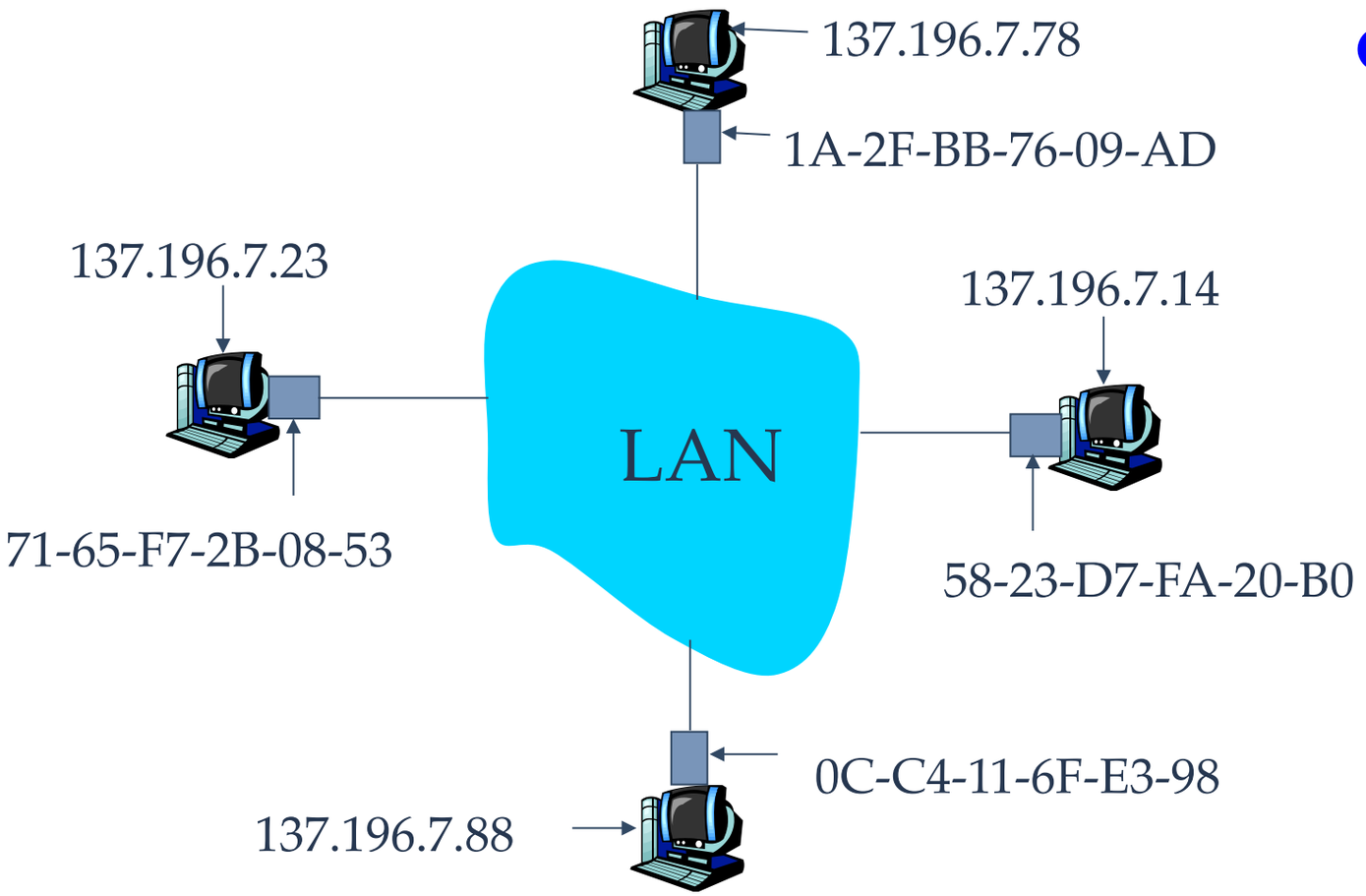


Broadcast address =
FF-FF-FF-FF-FF-FF

= adapter

ARP: Address Resolution Protocol

Question: how to determine MAC address of B knowing B's IP address?



- Each IP node (host, router) on LAN has ARP table
- ARP table: IP/MAC address mappings for some LAN nodes

< IP address; MAC address; TTL>

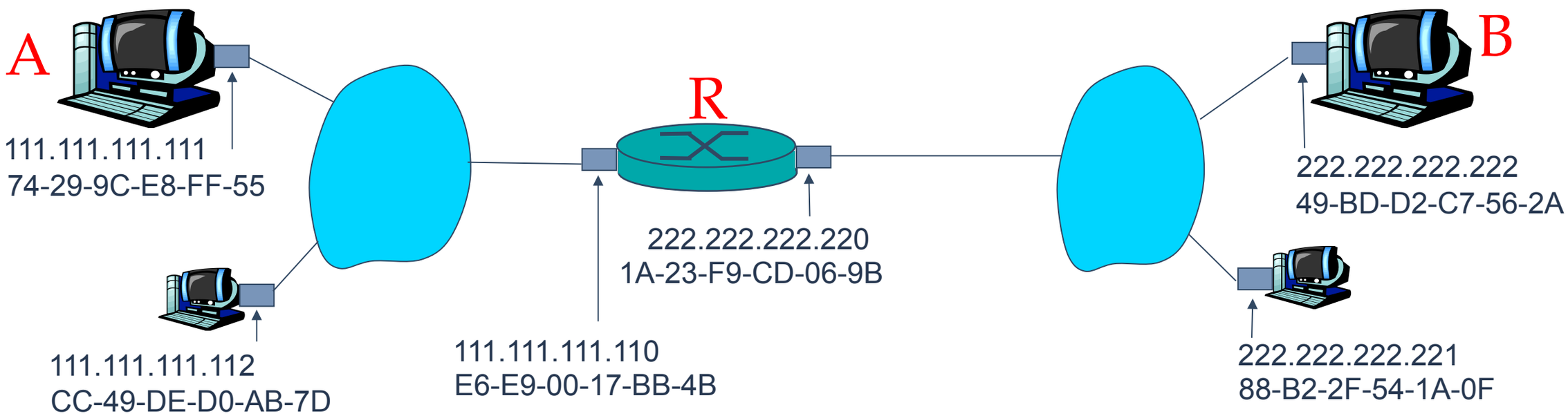
➡ TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

ARP protocol: Same LAN (network)

- A wants to send datagram to B, and B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - destination MAC address = FF-FF-FF-FF-FF-FF (broadcast address)
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

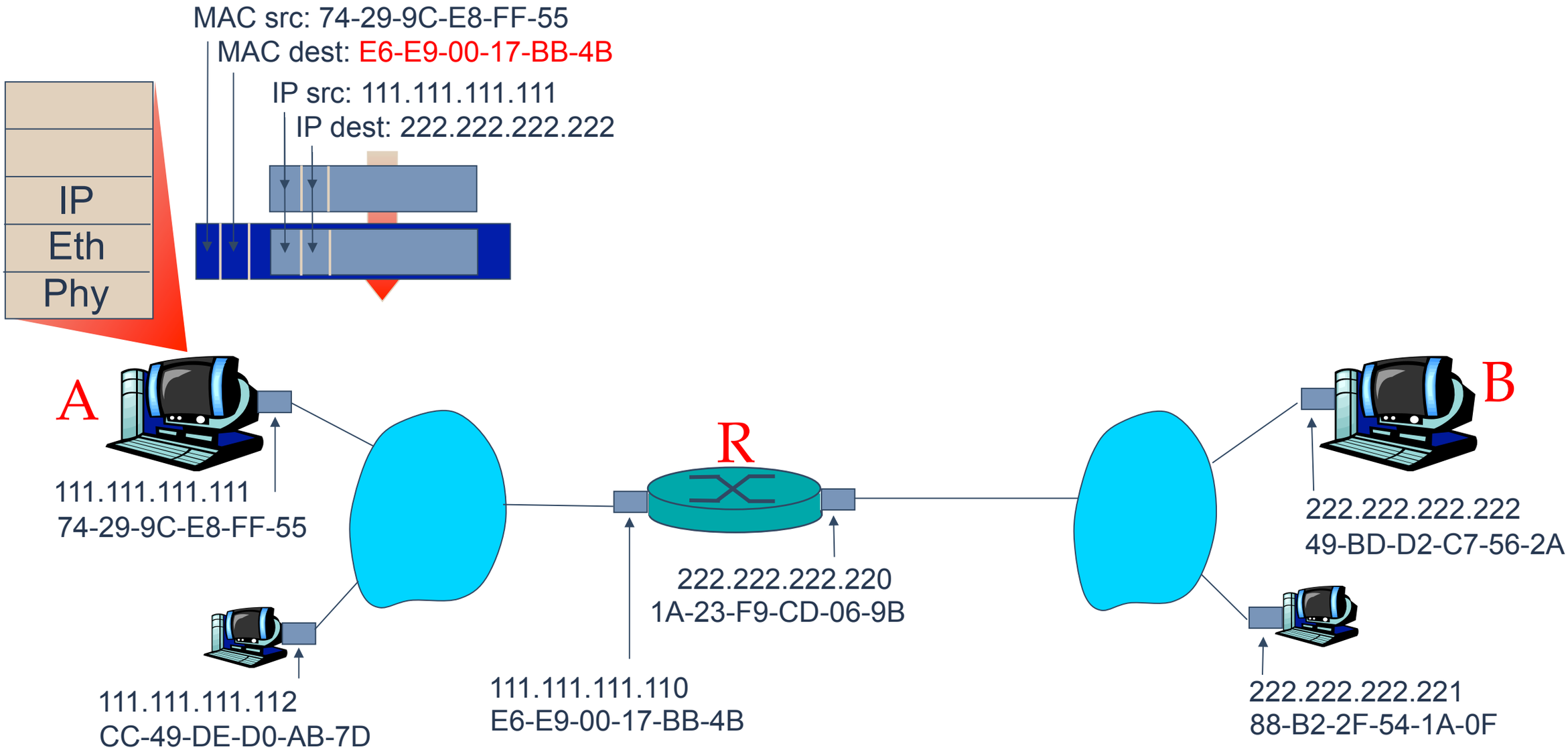
Addressing: routing to another LAN

- walkthrough: send datagram from A to B via R.
  - ➡focus on addressing - at both IP (datagram) and MAC layer (frame)
  - ➡assume A knows B's IP address
  - ➡assume A knows IP address of first hop router, R (how?)
  - ➡assume A knows MAC address of first hop router interface (how?)



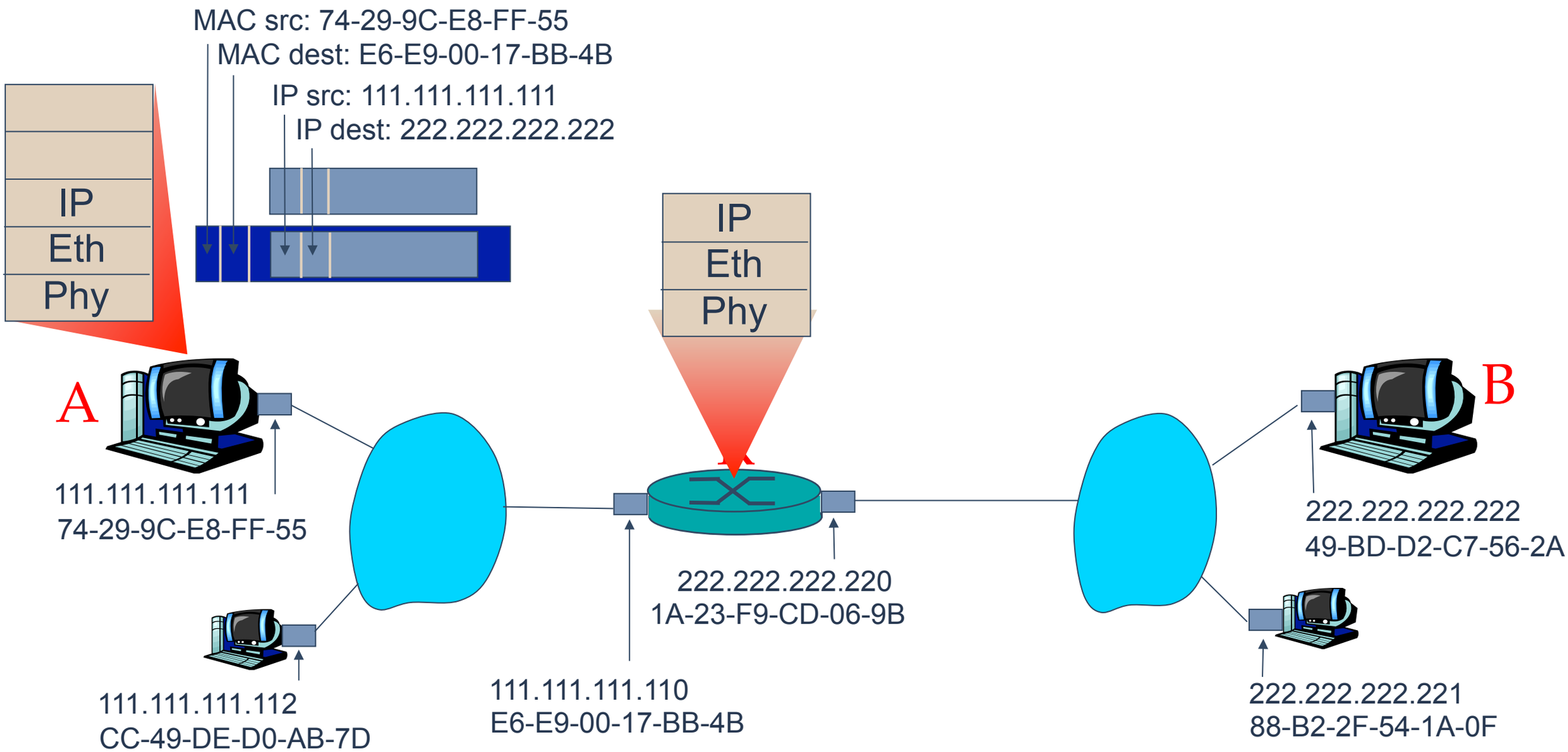
Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as destination, frame contains A-to-B IP datagram



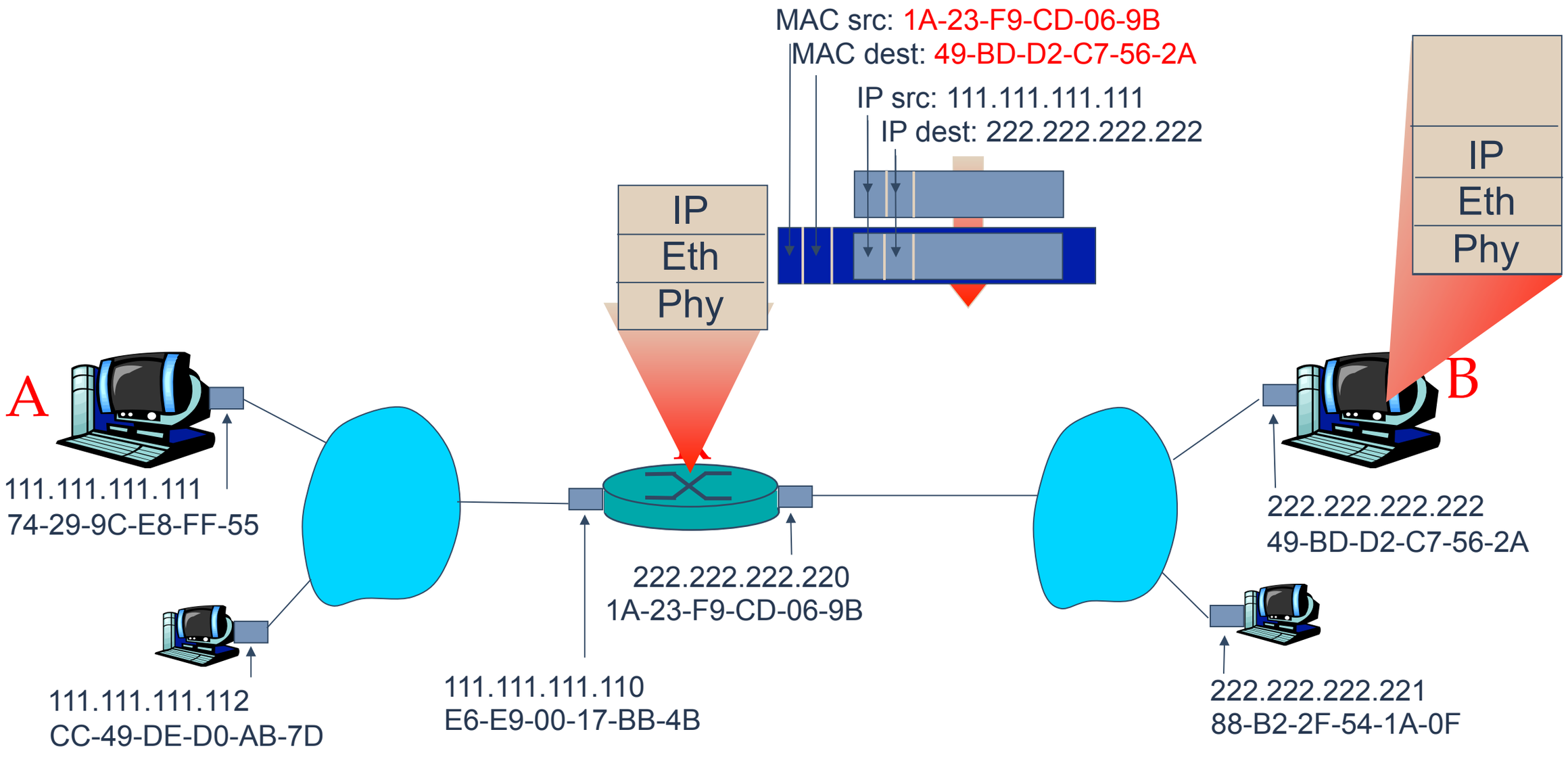
Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



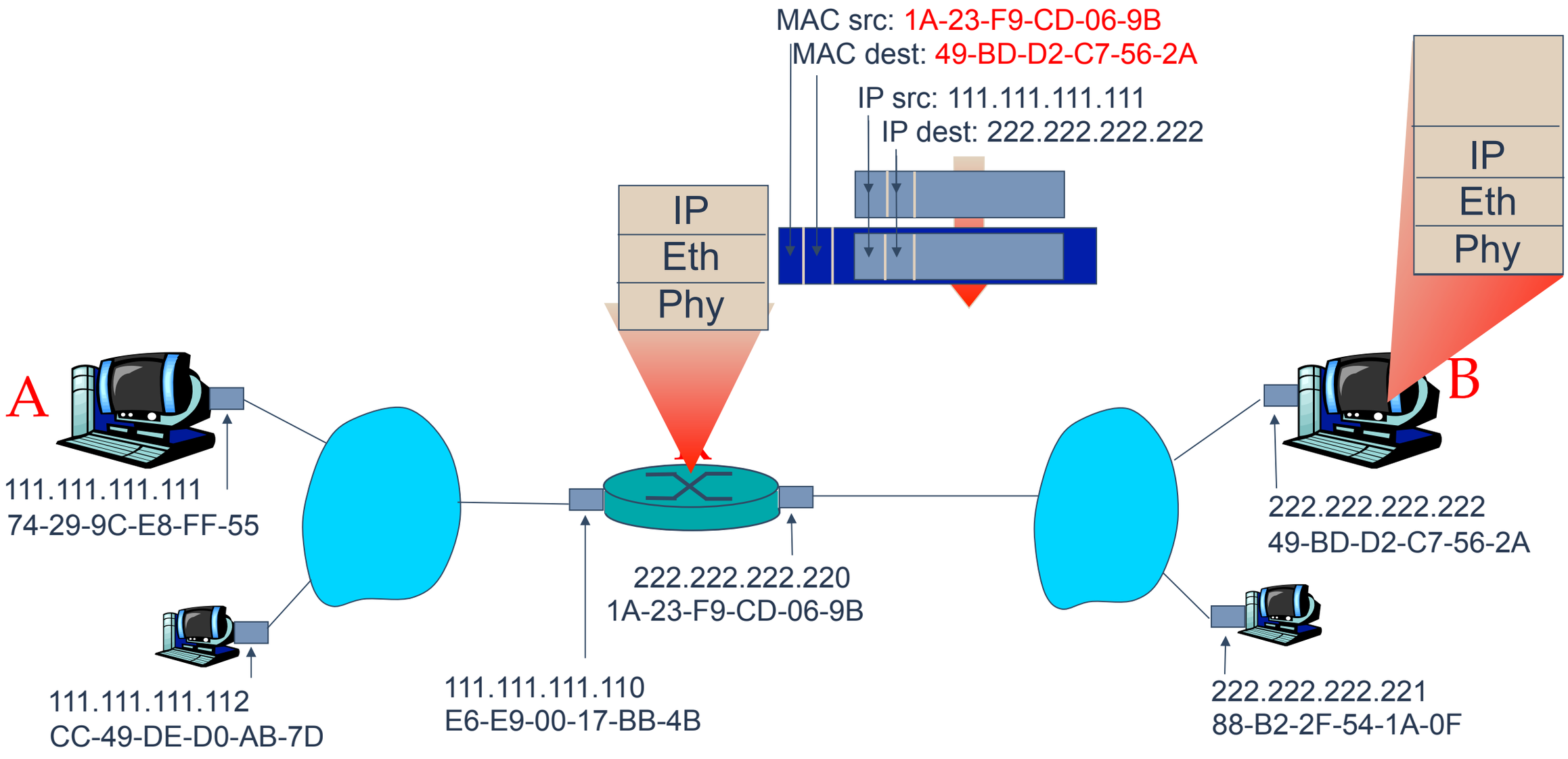
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as destination, frame contains A-to-B IP datagram



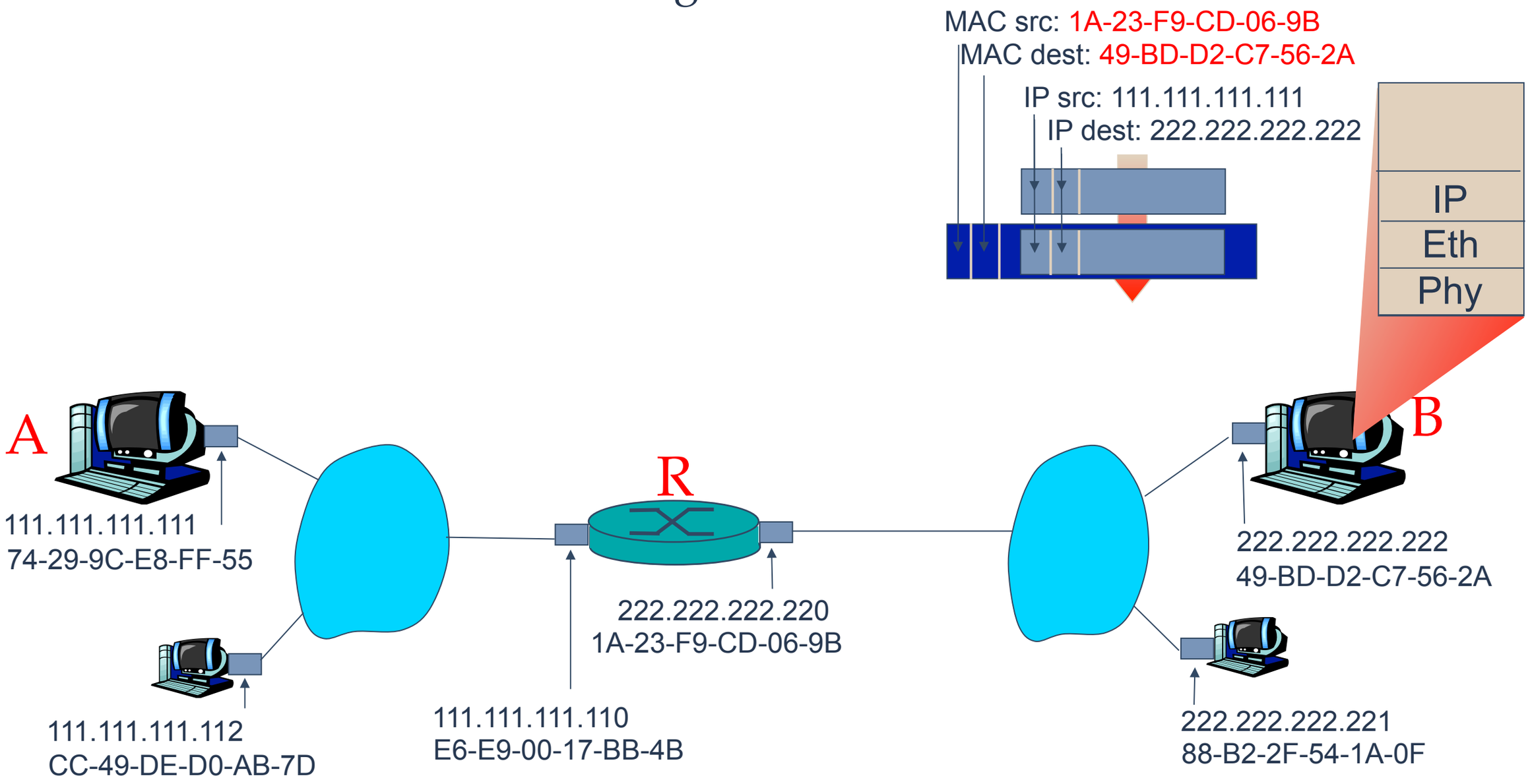
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Addressing: routing to another LAN

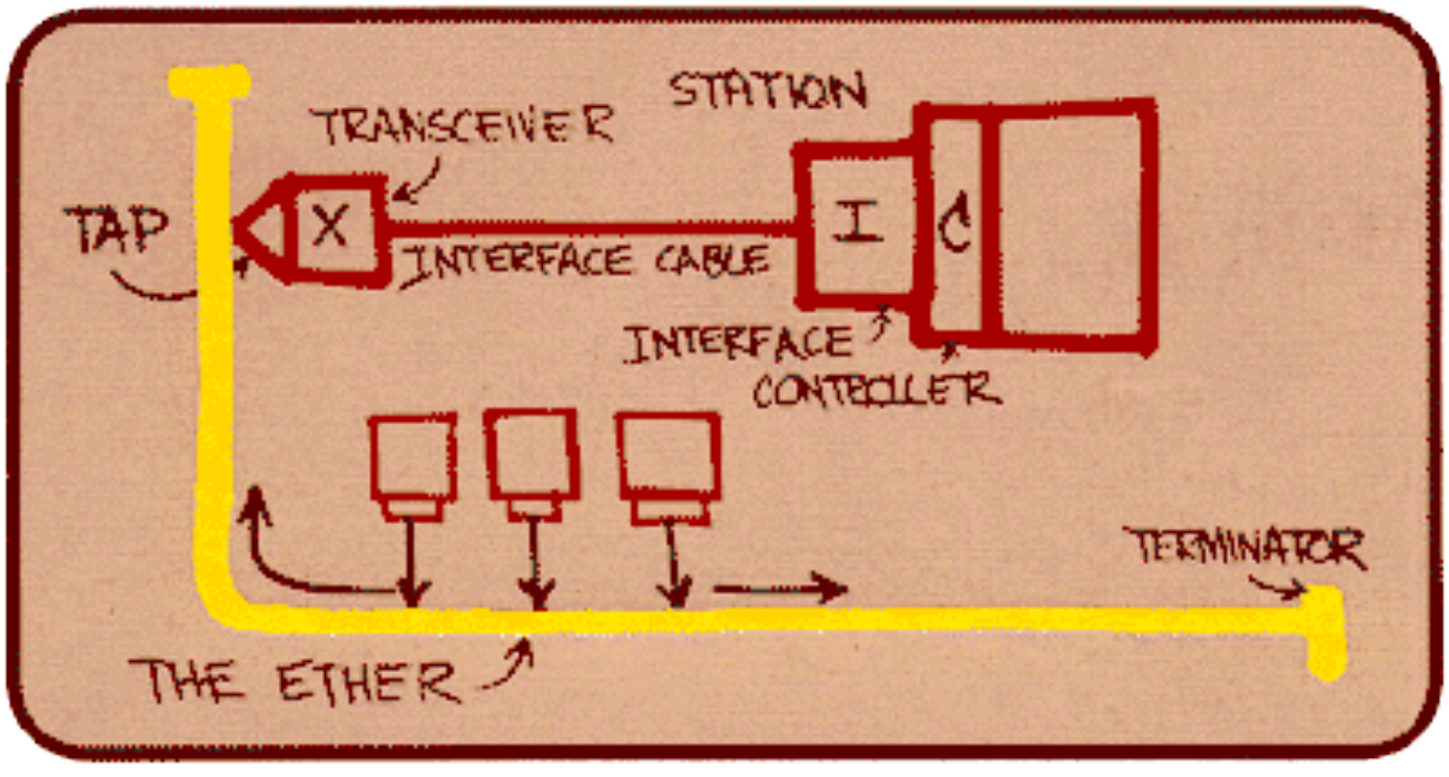
- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as destination, frame contains A-to-B IP datagram



Ethernet

"dominant" wired LAN technology:

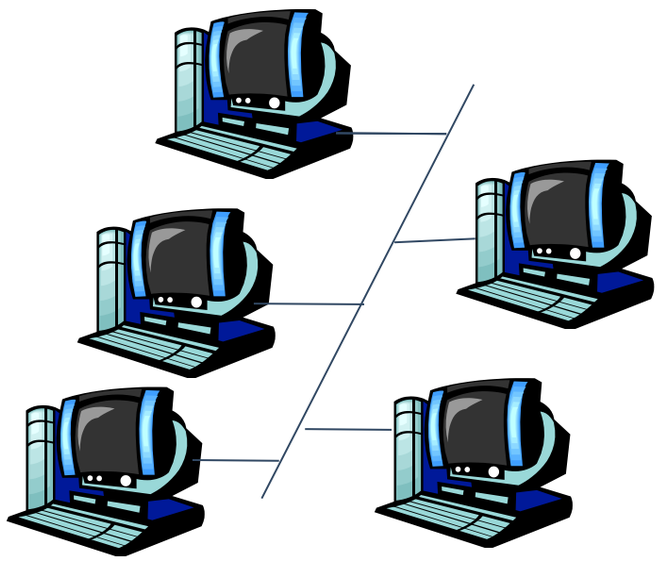
- cheap $20 for NIC
- first widely used LAN technology
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 10 Gbps



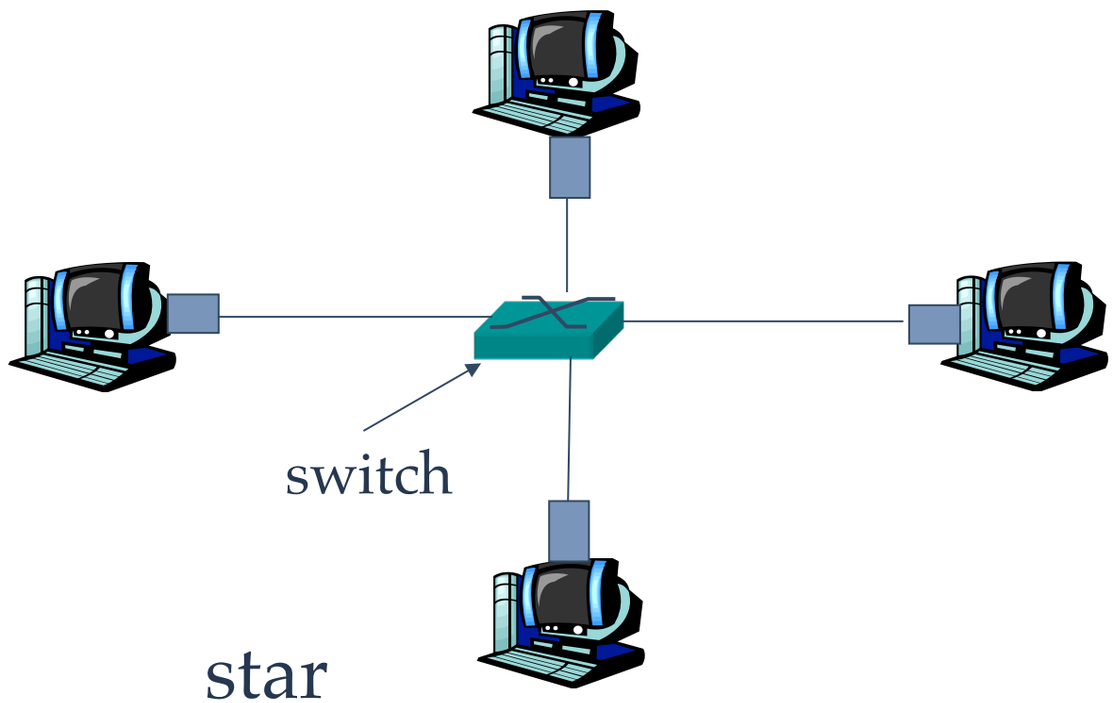
Metcalfe's Ethernet sketch

Star topology

- bus topology popular through mid 90s
  - ➡ all nodes in same collision domain (can collide with each other)
- today: star topology prevails
  - ➡ active *switch* in center
  - ➡ each "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)



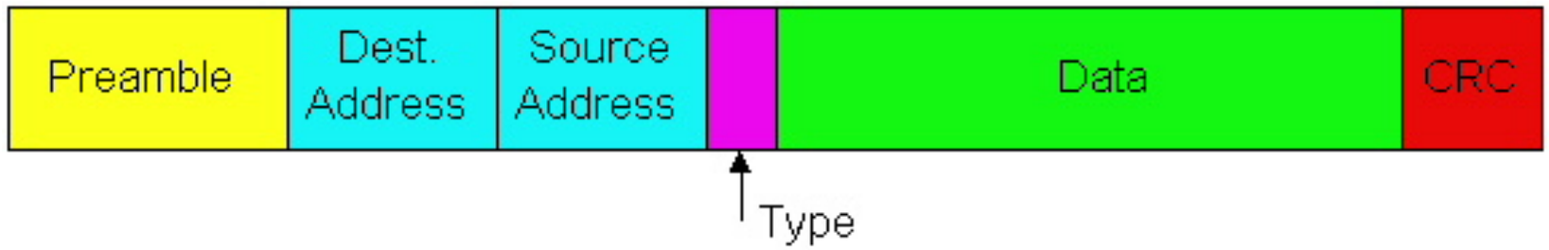
bus: coaxial cable



star

Ethernet Frame Structure

Sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

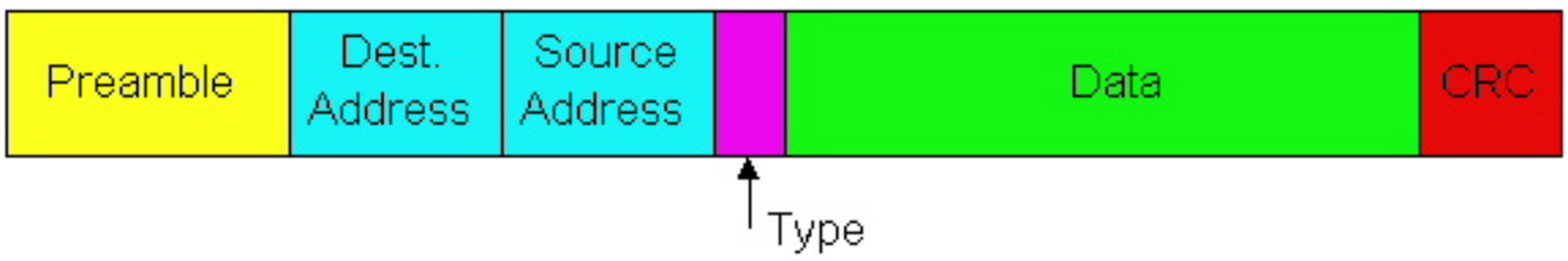


Preamble:

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

Ethernet Frame Structure (more)

- Addresses: 6 bytes
  - ➡ 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: checked at receiver, if error is detected, frame is dropped



Ethernet: Unreliable, connectionless

- Connectionless: No handshaking between sending and receiving NICs
- Unreliable: receiving NIC doesn't send acks or naks to sending NIC
  - ➡ stream of datagrams passed to network layer can have gaps (missing datagrams)
  - ➡ gaps will be filled if app is using TCP
  - ➡ otherwise, app will see gaps
- Ethernet's MAC protocol: unslotted CSMA/CD

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
5. After aborting, NIC enters exponential backoff: after *n*th collision, NIC chooses *K* at random from {0,1,2,…,2^m-1} (*m*=min(*n*,10)). NIC waits K×512 bit times, returns to Step 2

Ethernet's CSMA/CD (more)

Jam Signal: make sure all other transmitters are aware of collision; 48 bits

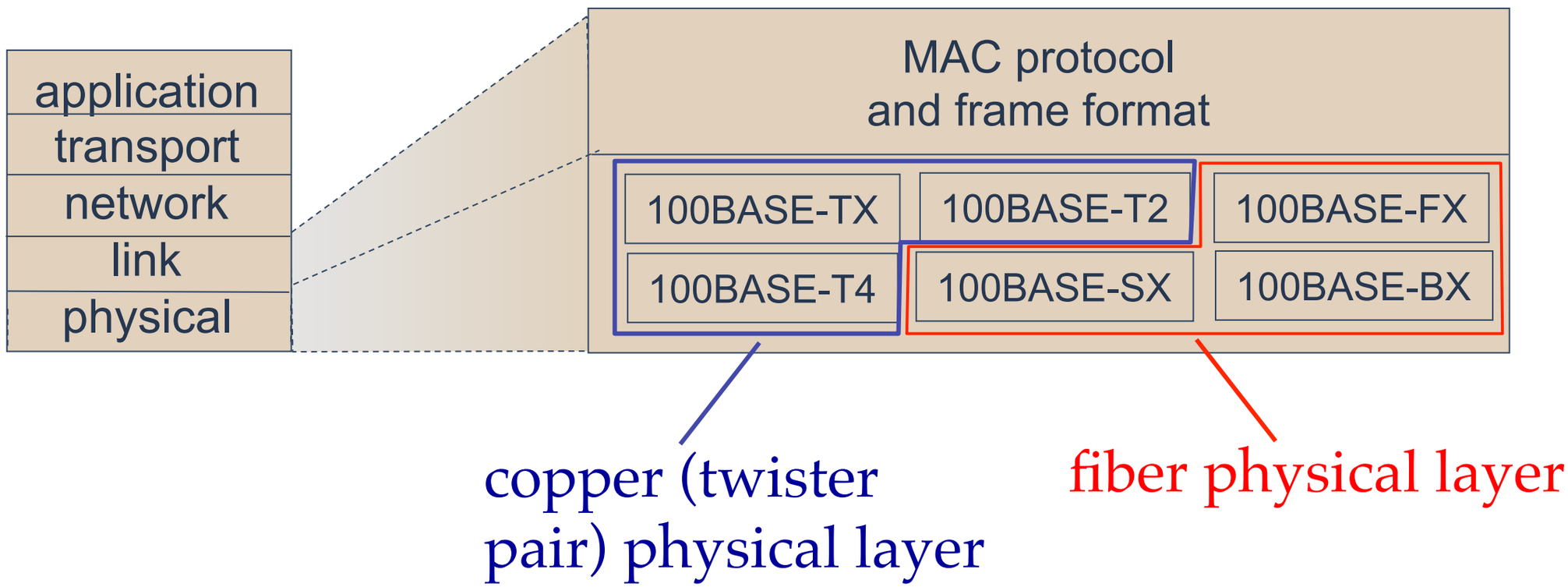
Exponential Backoff:

- *Goal*: adapt retransmission attempts to estimated current load
  - ➡ heavy load: random wait will be longer
- first collision: choose K from {0,1}; delay is $K \times 512$ bit transmission times
- after second collision: choose K from {0,1,2,3}…
- after ten collisions, choose K from {0,1,2,3,4,…,1023}

Bit time: .1 microsec for 10 Mbps Ethernet ;
for K=1023, wait time is about 50 msec

802.3 Ethernet Standards: Link & Physical Layers

- *many* different Ethernet standards
  - ➡ common MAC protocol and frame format
  - ➡ different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10G bps
  - ➡ different physical layer media: fiber, cable



Point to Point Data Link Control

- One sender, one receiver, one link: easier than broadcast link:
  - No Media Access Control
  - No need for explicit MAC addressing
  - E.g., dialup link, ISDN line
- Popular point-to-point DLC protocols:
  - PPP (point-to-point protocol)
  - HDLC: High level data link control (Data link used to be considered "high layer" in protocol stack!

PPP Design Requirements

- Packet framing: encapsulation of network-layer datagram in data link frame
  - carry network layer data of any network layer protocol (not just IP) *at same time*
  - ability to demultiplex upwards
- Bit transparency: must carry any bit pattern in the data field
- Multiple network-layer protocols
- Multiple link types
- Error detection (no correction)
- Connection liveness: detect, signal link failure to network layer
- Network layer address negotiation: endpoint can learn/ configure each other's network address

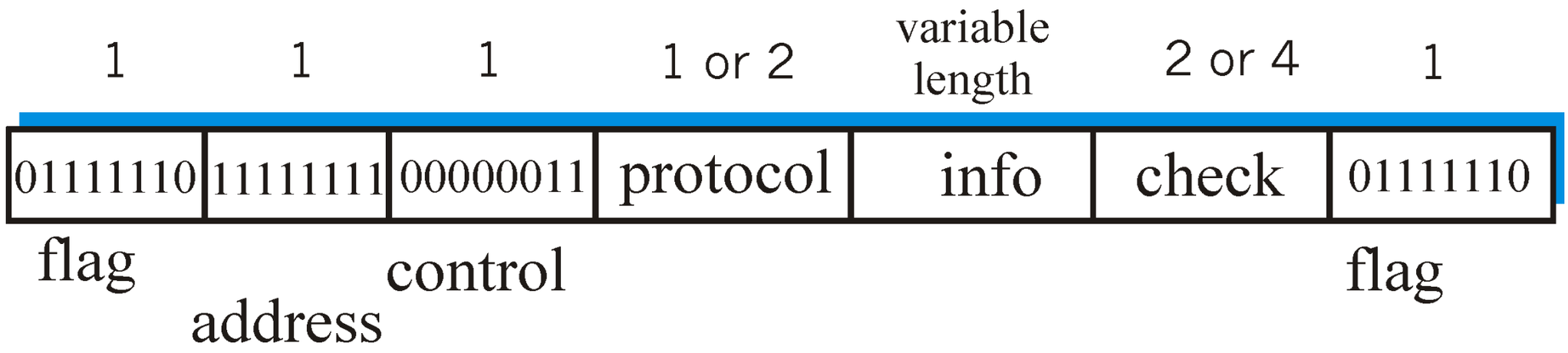
PPP non-requirements

- No error correction/recovery
- No flow control
- Out of order delivery OK
- No need to support multipoint links (e.g., polling)

Error recovery, flow control, data re-ordering
all relegated to higher layers!

PPP Data Frame

- flag: delimiter (framing)
- address: does nothing (only one option)
- control: does nothing; in the future possible multiple control fields
- protocol: upper layer protocol to which frame delivered (e.g., PPP-LCP, IP, IPCP, etc)
- info: upper layer data being carried
- check: cyclic redundancy check for error detection



Byte Stuffing

- "Data transparency" requirement: data field must be allowed to include flag pattern <01111110>
- Sender:
  - ➡ adds ("stuffs") extra <01111110> byte after each <01111110> *data* byte
- Receiver:
  - ➡ two 01111110 bytes in a row: discard first byte, continue data reception
  - ➡ single 01111110: flag byte

Byte Stuffing