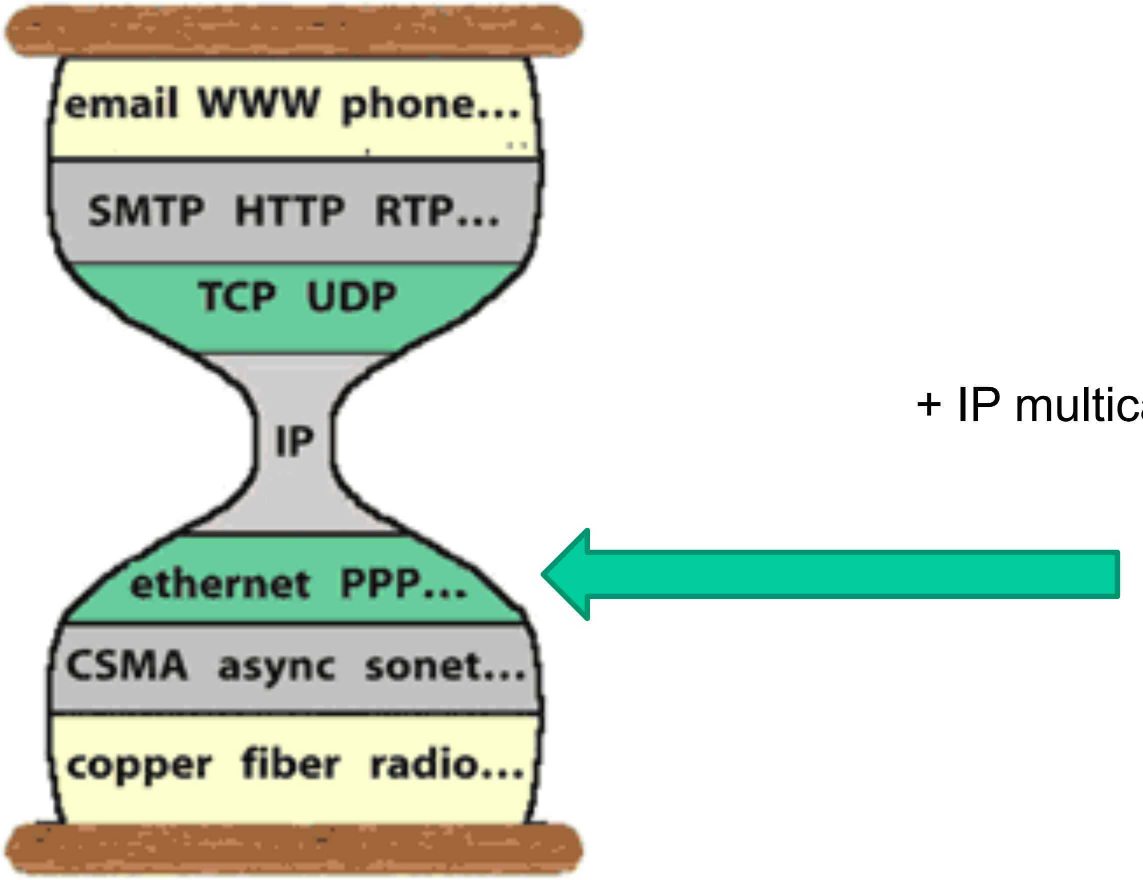


+ IP multicast and IGMP protocol (Ch.4.7)

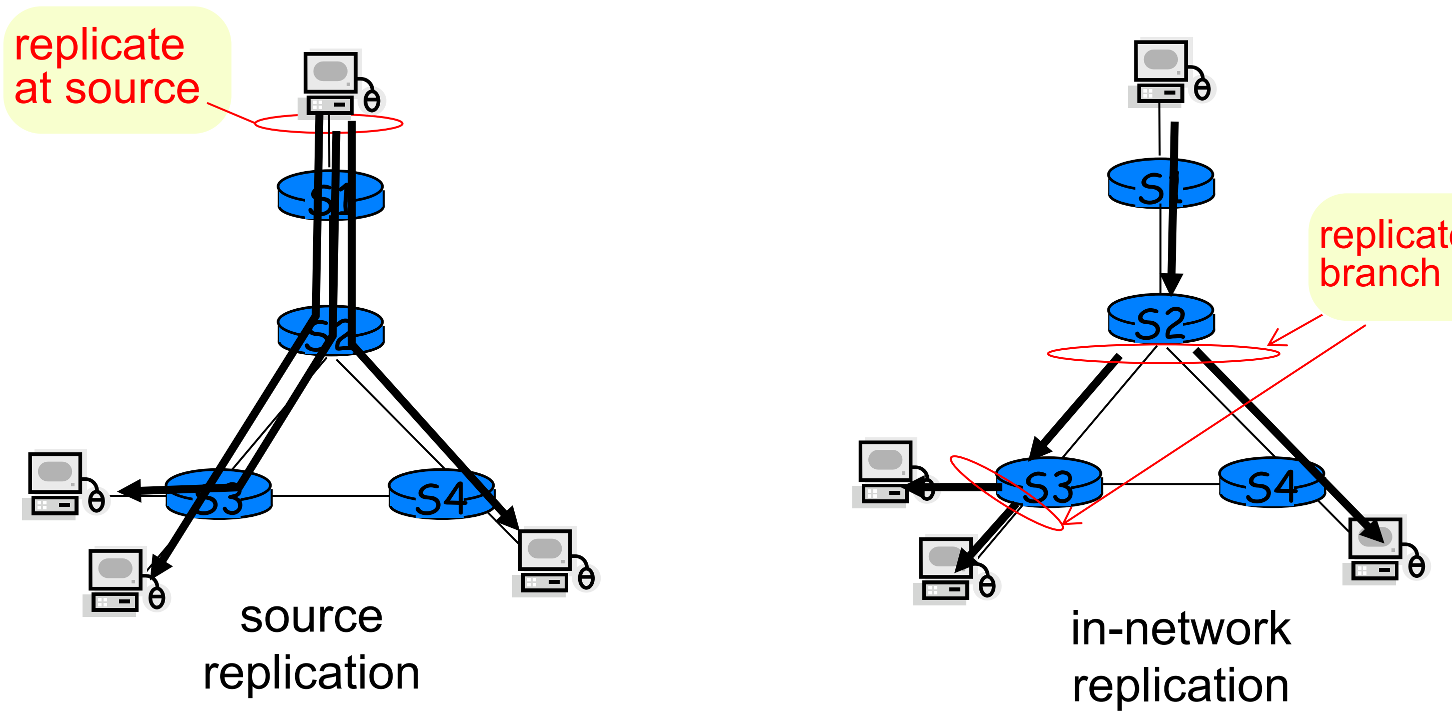
Link layer (Ch. 5)

Delivering Packets (at the Link Layer)

- Unicast
  - "quietly talking to a single person in the room"
- Broadcast
  - "shouting to everybody"
    - how to properly scream so your message can be heard in different rooms?
- Multicast
  - "forming small groups and quietly shouting"
    - how to form a group?
    - how to ensure group can spread in multiple rooms?

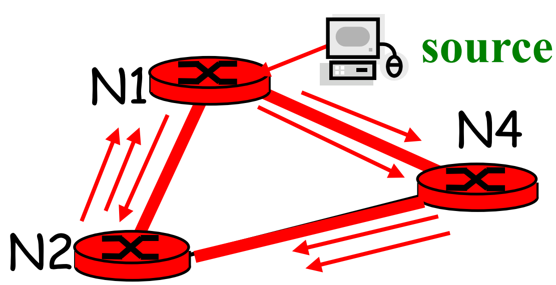
Broadcast

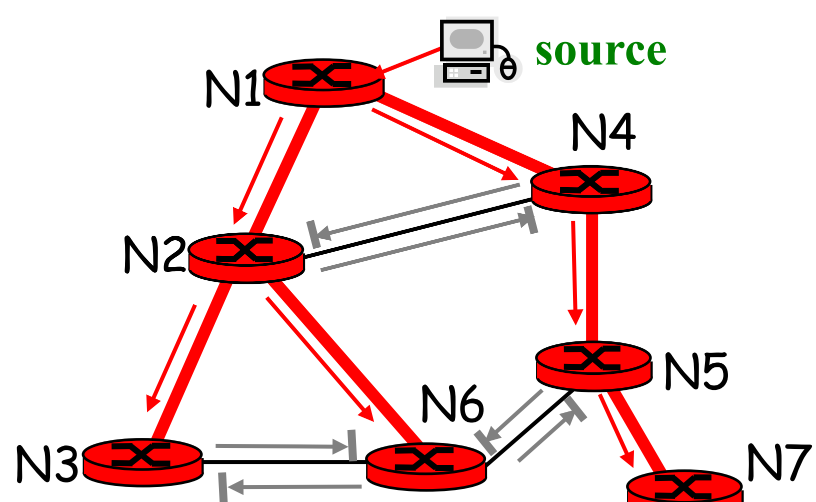
- Packets from one host go to all other hosts
- 2 ways to implement



Broadcast by in-network replication

- Flooding: when a node receives a broadcast packet, sends a copy to all its neighbors
  - Problem: packet looping
- Controlled flooding: node broadcasts a packet only if it hasn't seen the same packet before
  - Keep track of all packets already seen
  - Reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source

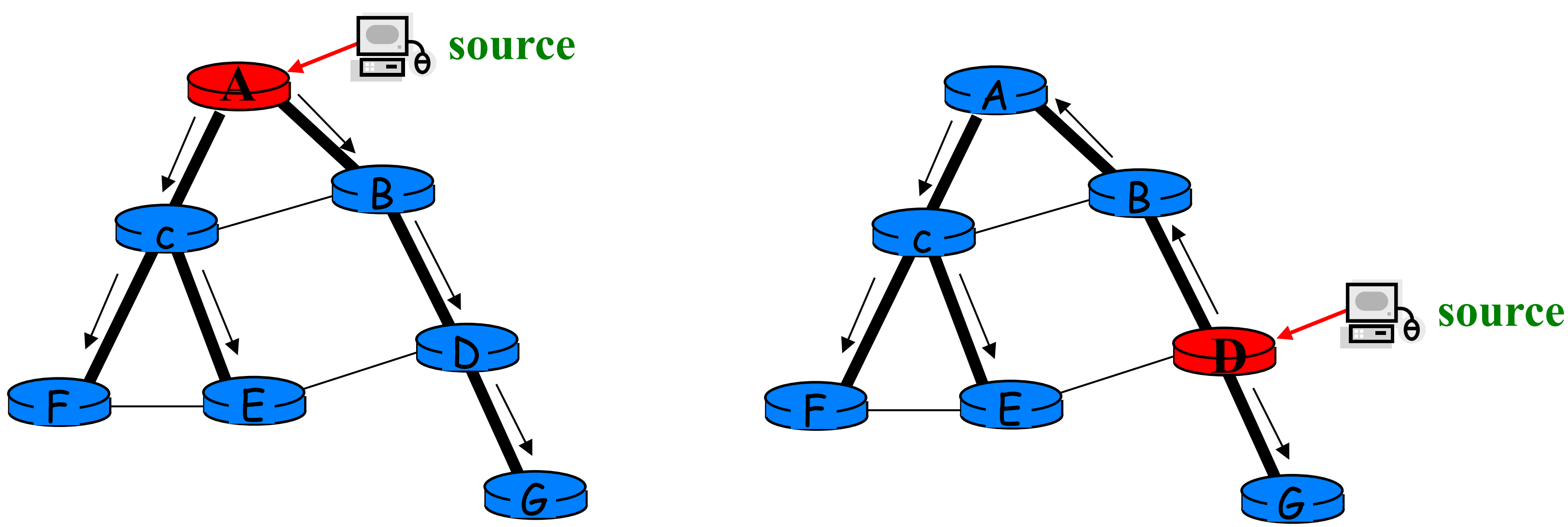




Duplicate elimination: Spanning Tree

- First construct a spanning tree
- Nodes forward copies only along spanning tree
- All sources send data along **the same tree**

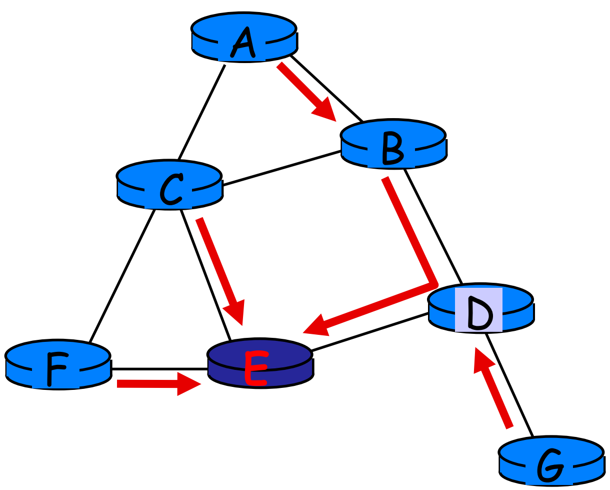
Q: how to choose the root of the tree?



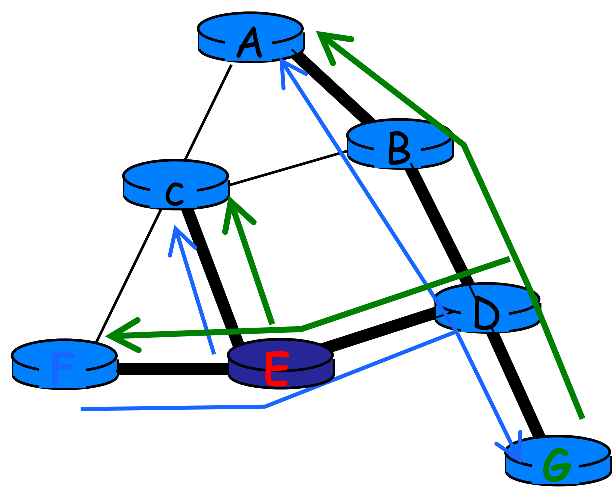
(a) Broadcast initiated at A **(b) Broadcast initiated at D**

Answer 2: Center-based Spanning Tree

- First pick a center node (***E*** in this example)
- *Each node* sends *unicast* join message *towards* the center node
  - A join message is forwarded until it arrives at a node already on the spanning tree



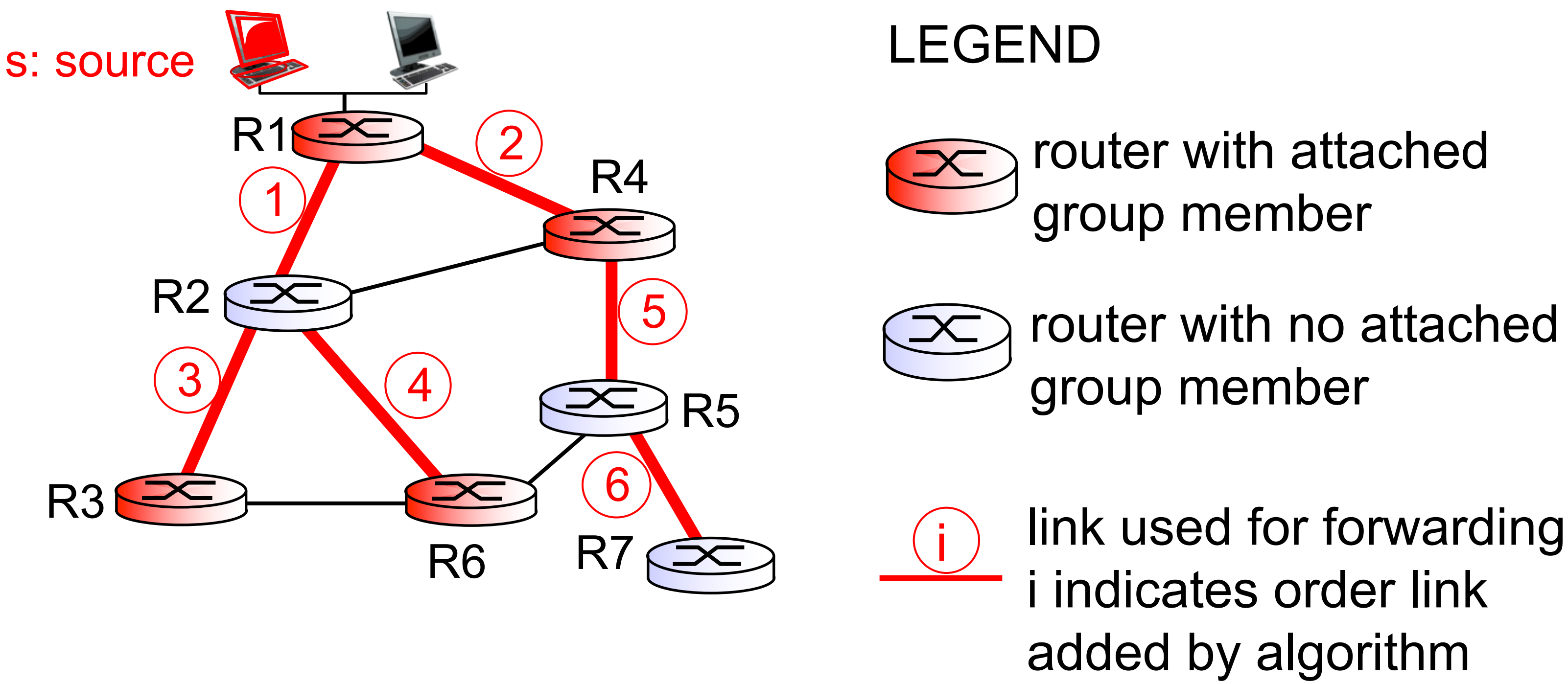
(a) Stepwise construction of spanning tree



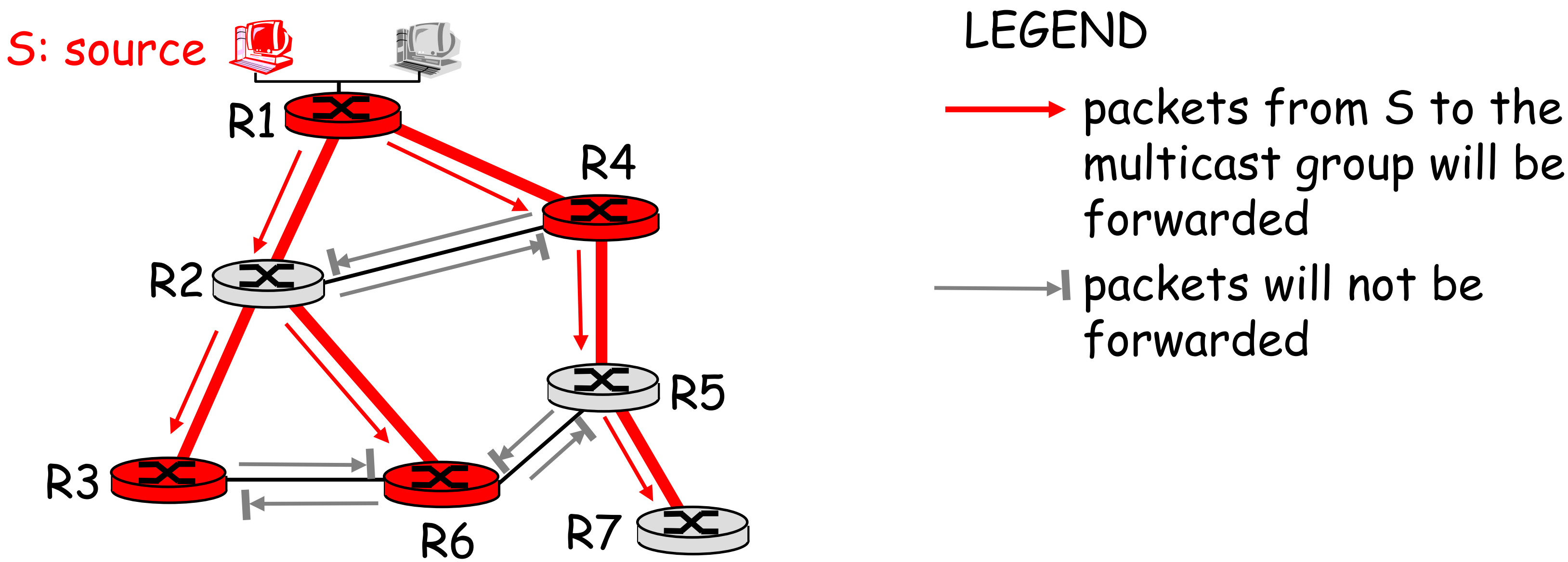
(b) Resulting spanning tree

Building Shortest path tree

- Build a tree of shortest path routes from each source to all receivers
- Solution 1: Router calculates the tree using Dijkstra's algorithm

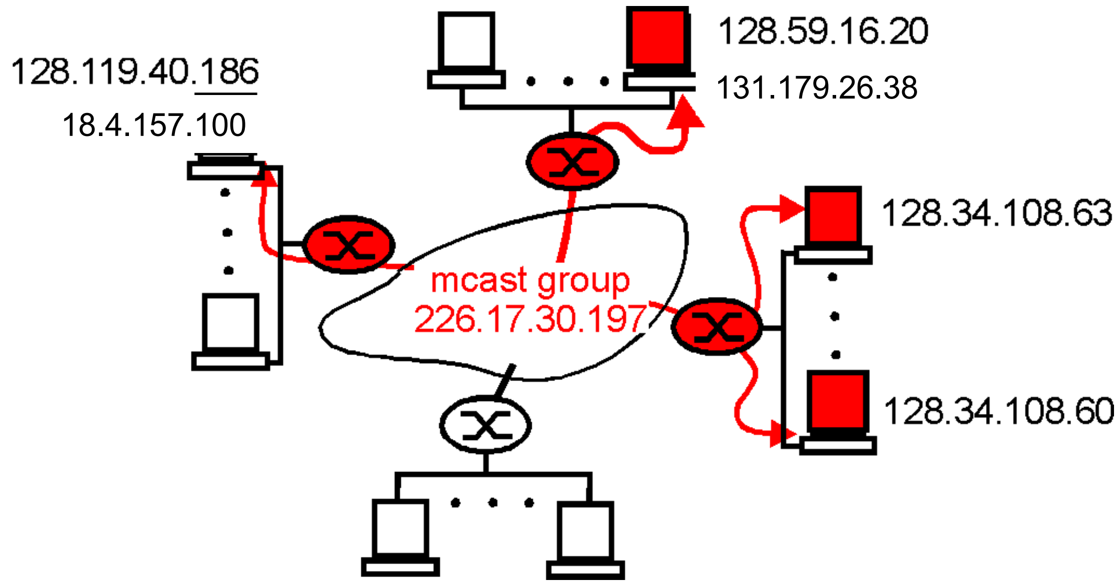


Solution 2: using Reverse Path Forwarding

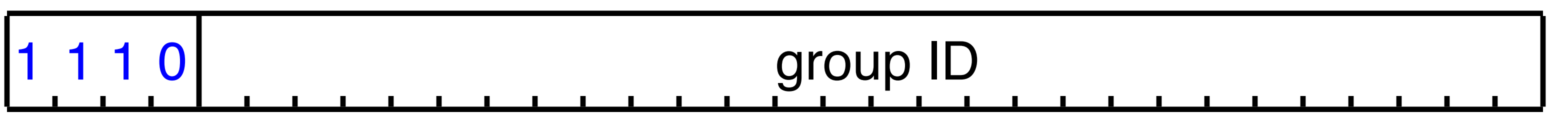


- a node N forwards packet from source **S** if it arrived on shortest path from N to **S**
- Assuming symmetric link cost, RPF builds a source-specific shortest path broadcast tree
  - But we need a multicast tree

IP Multicast Address



Class D IP addresses:



in “dotted decimal” notation: **224.0.0.0 — 239.255.255.255**

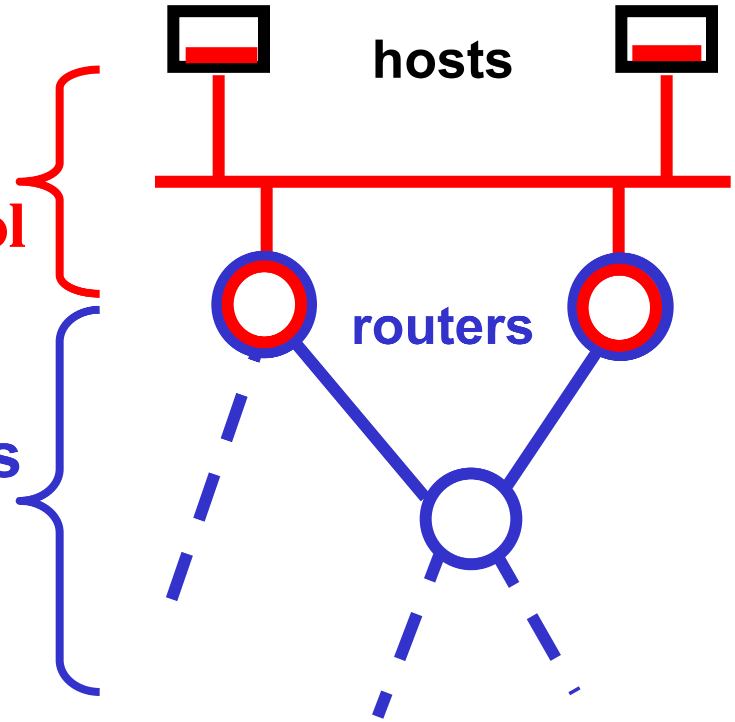
Two administrative categories:

- *well-known* multicast addresses, assigned by IANA
  - http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml
- the rest: *transient* addresses, assigned & reclaimed dynamically

Components of IP Multicast Architecture

host-to-router protocol
Internet Group Management Protocol

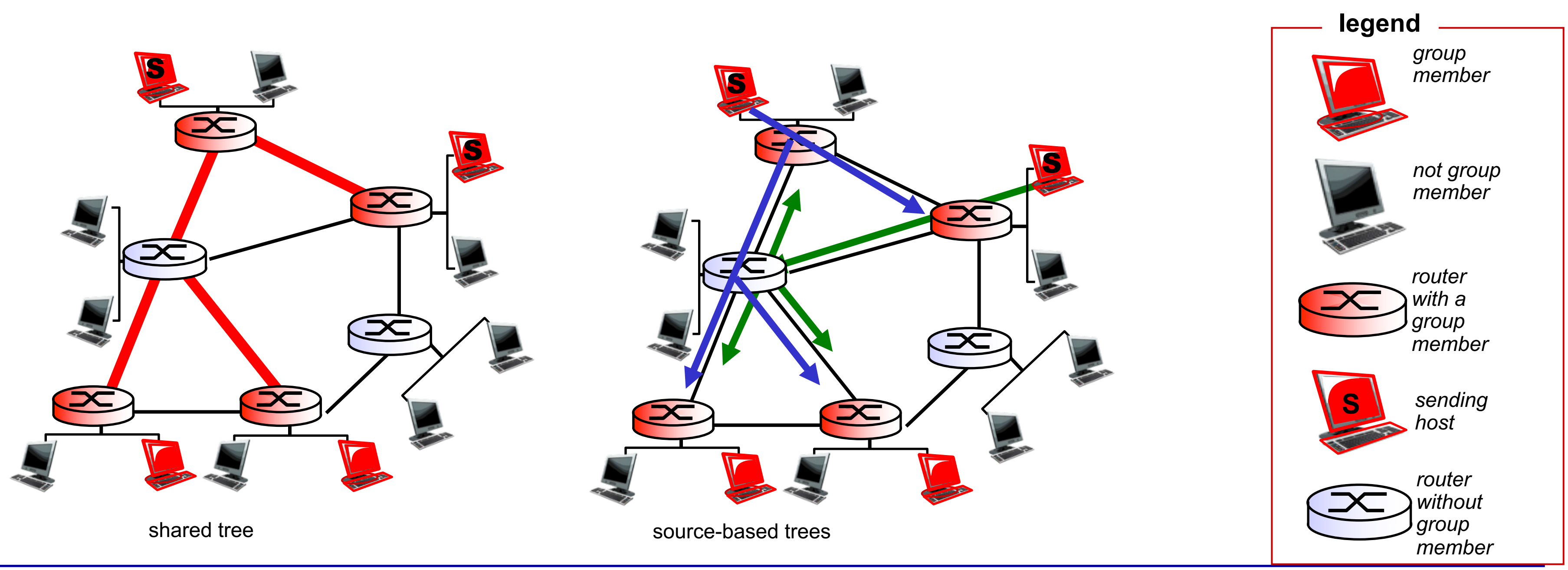
multicast routing protocols
(MOSPF, DVMRP, PIM)



- IGMP operates between a router and local hosts on the *same* network (e.g., WiFi, Ethernet)
  - Router queries local hosts for mcast group membership info
  - Hosts respond with membership reports

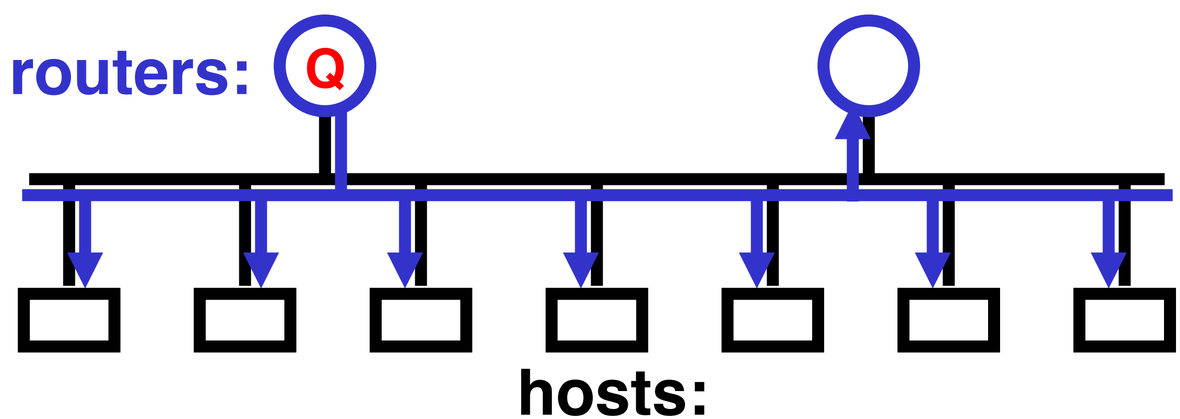
Multicast Routing: the basic idea

- IGMP
  - Let local routers know that there are members in the group
    - “Manager periodically running around and screaming: are you interested in any conversations?”
    - “People who are interested scream back list of conversations they are interested”
- Multicast routing protocols (DVMRP, PIM)
  - Maintain a tree (or trees) connecting all routers having local mcast group members (“between buildings”)
    - shared-tree: *same* tree used by all group members
    - source-tree: one tree from each sender to all receivers



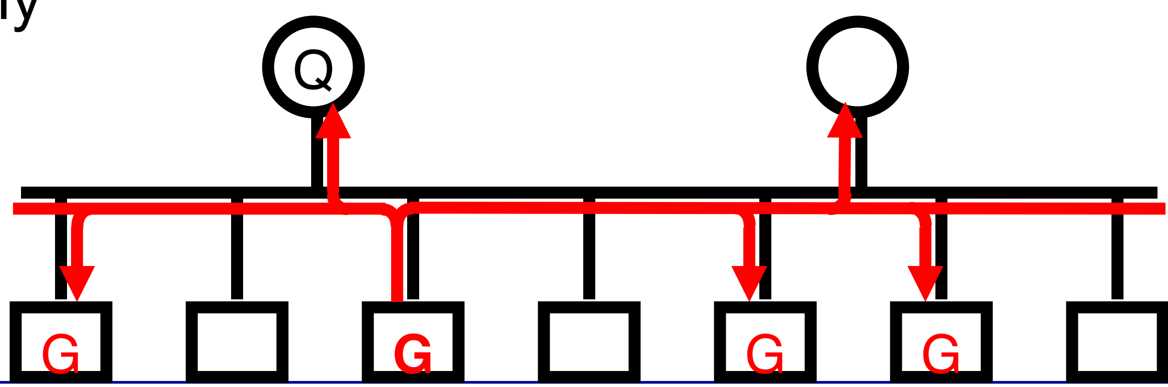
How IGMP Works: Router Query

- One router is elected the "querier" on each local/physical network
- querier *periodically* sends Membership Query message to "all-systems group" (224.0.0.1) with TTL=1



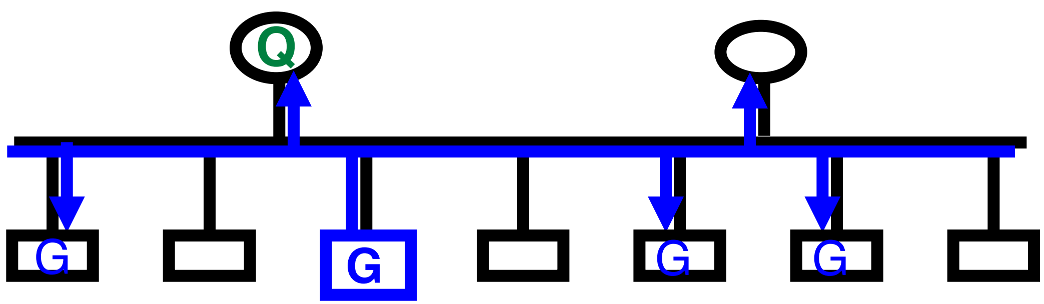
How IGMP Works: Host Reply

- On receipt, a host starts a *random timer* [0–10 sec] for *each* multicast group it wants to join
- when a host's timer for group G expires, it sends a Membership Report to group G (TTL = 1)
  - other members of G hear the report, stop their timers
  - routers hear all reports
- Normal case: one report message *per group* is sent in response to a query
- when a host first joins a group: may send unsolicited reports immediately



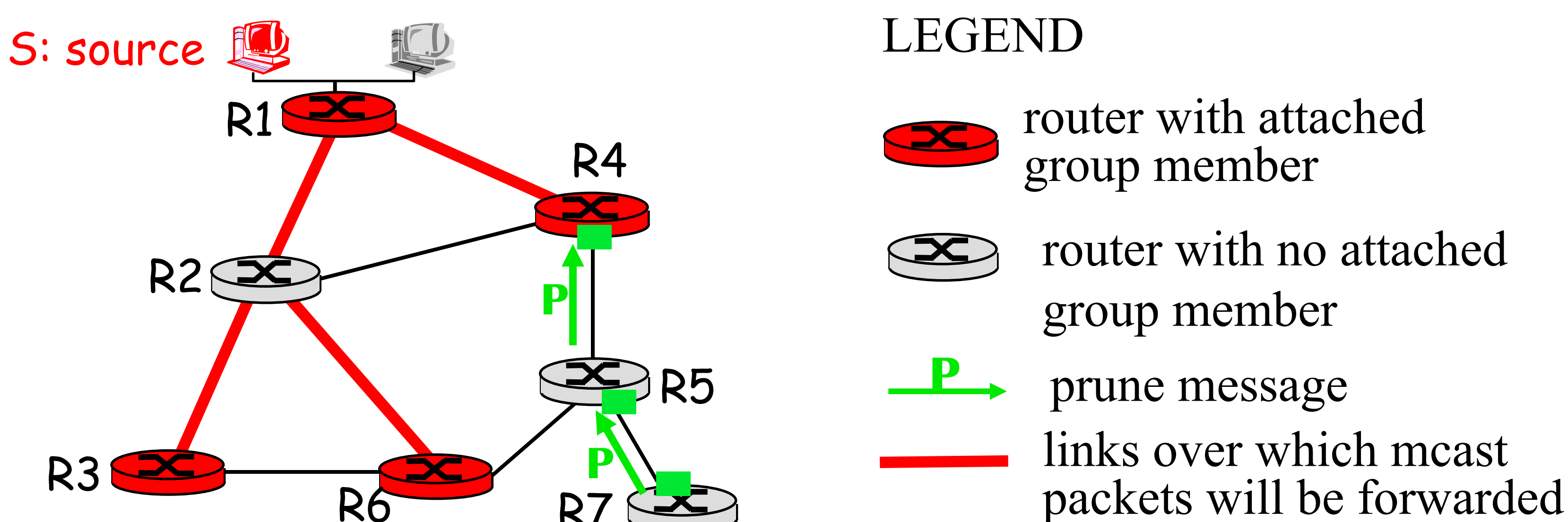
Leaving a Multicast Group

- host sends a Leave Group msg to group address G *if and only if* it was the most recent host to report membership in that group
- Upon receiving Leave Group msg: Q-router sends a few queries to group G with a small max-response-time
  - if no Membership Report heard, stop data forwarding



Pruning: Trim broadcast tree to mcast tree

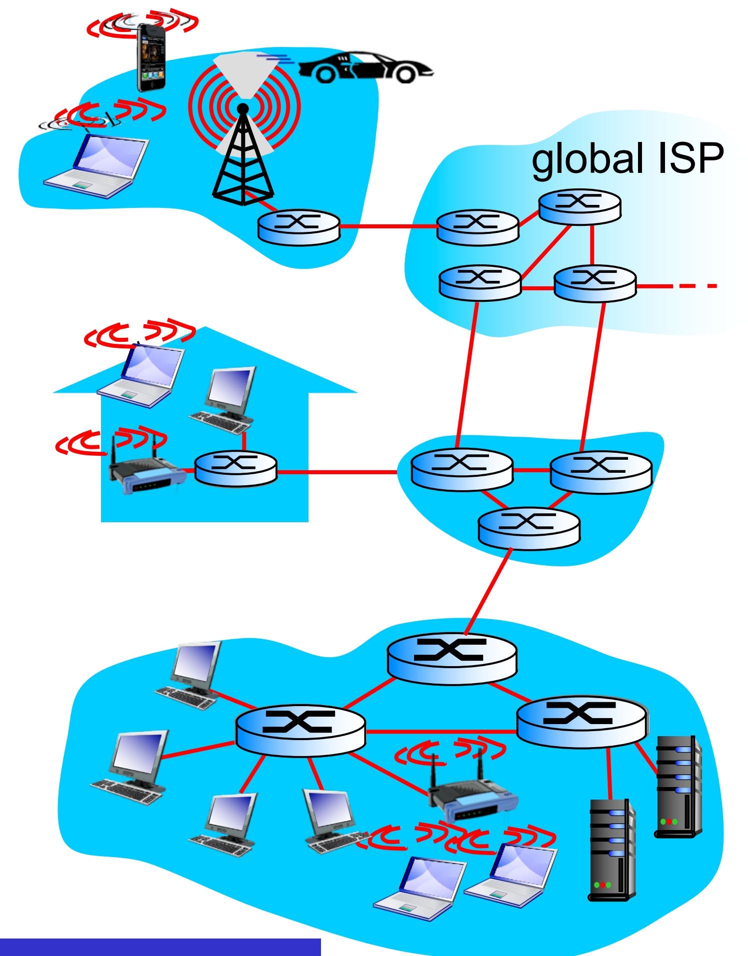
- no need to forward packets down branches *which has no mcast group members*
- router with no downstream group members sends "prune" message upstream
  - Routers keep ***state*** to remember prune msgs

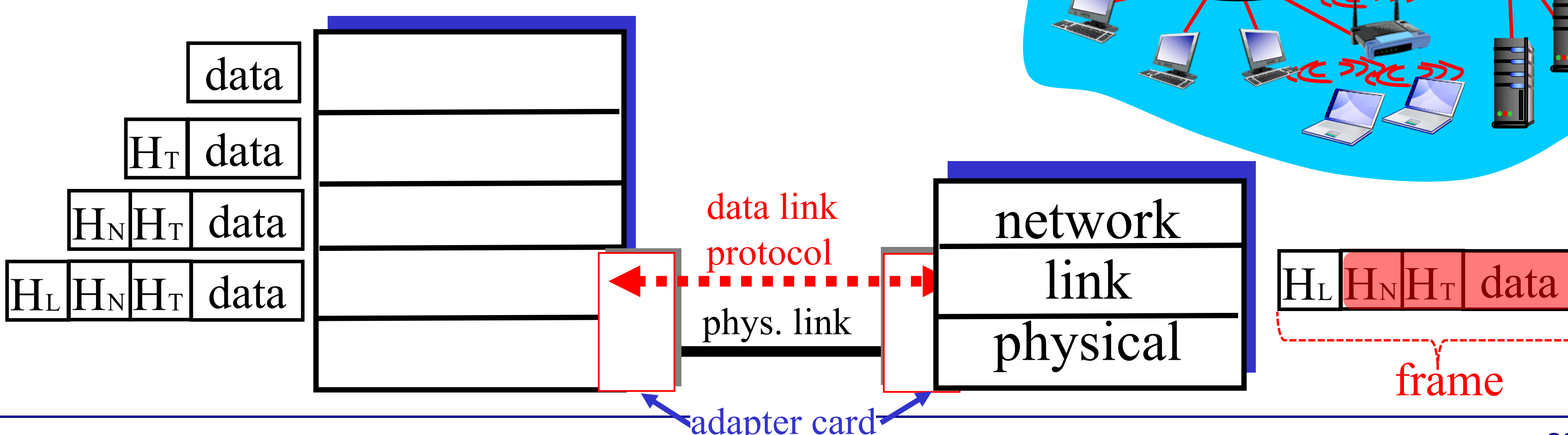


LINK LAYER

Link Layer: overview

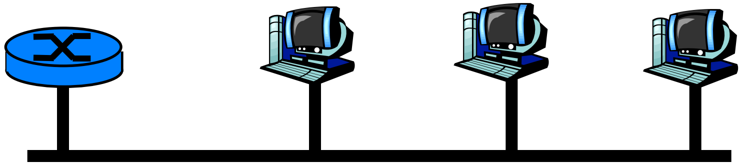
- Link layer transfers packets from one node to physically adjacent node over a link
  - Nodes: Routers, hosts
- implementation of various link layer technologies:
  - Ethernet, wireless LANs
- Encapsulate IP packet in layer-2 frame





Link Layer functions

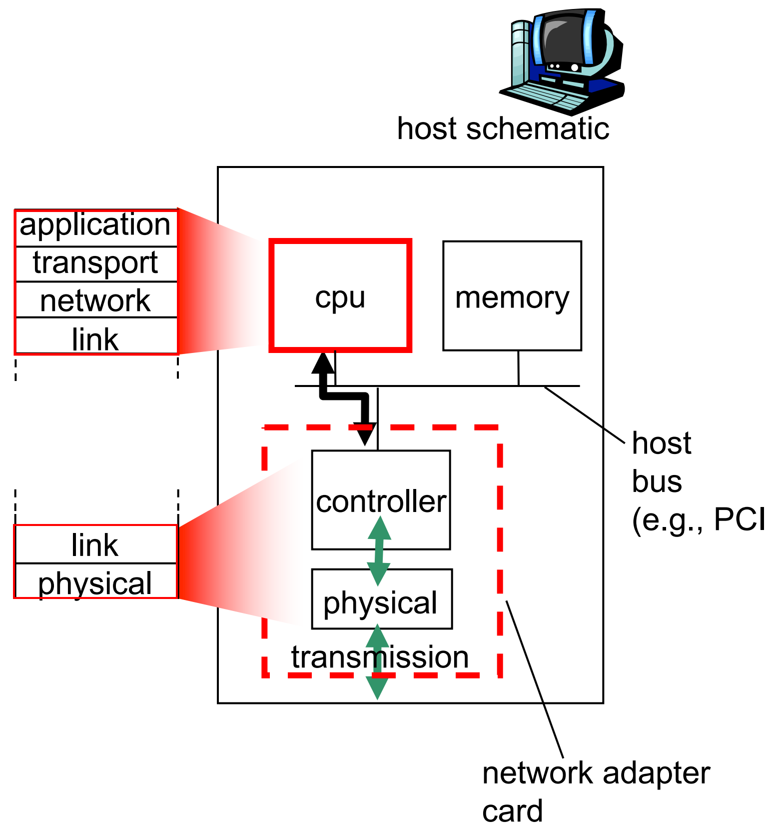
- Link type: simplex, Half-duplex, full-duplex
  - Multi-access links, e.g, Ethernet, WiFi
- Link layer address: MAC (Medium Access Control) addresses
- Link layer functions:
  - Data framing
    - **link layer receives just a sequence of bits from physical layer**
    - the beginning/end of a data chunk needs to be demarcated
  - Error detection
  - Channel access protocols



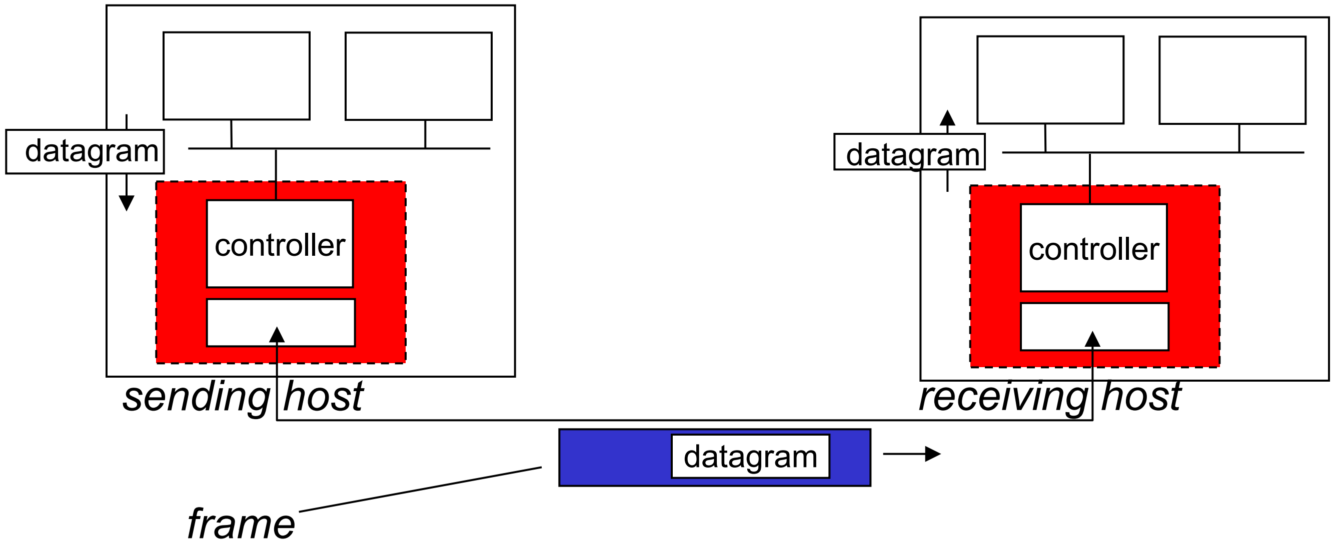
Where is the link layer implemented?

FYI

- implemented in adaptor (aka *network interface card*, NIC) or on a chip
  - Ethernet card, Thunderbolt adapter, etc.
  - Implements link & physical layer
- Attached to host's system buses
- Combination of hardware, software, firmware



Adaptors communicating



sending side:

- encapsulates IP packet in frame
- adds error checking bits
- Following access control protocol to send frame out

receiving side

- looks for errors
- If OK , extracts datagram, passes to upper layer at receiving side

Data Framing

- For a block of data
  - at link layer: a data frame
  - at network layer: a packet
  - at transport level: TCP — a segment; UDP — a datagram
- A frame has a header field
  - optionally there may be a trailer field as well

- Byte-Oriented Framing Protocol: delineate frame with a byte of special bit sequence: 01111110

Q: What if the bit sequence 01111110 occurs in data stream?

Byte Stuffing Ideas

- Input to the sender
  - 01111110 01010101 01111110 01111110 01111101 ...
- Input to the receiver
  - **01111110** 011111**0**10 01010101 011111**0**10 01111110 ... 011111**0**01 <tail> **01111110**

Byte stuffing

- HDLC Byte Stuffing
  - bit-oriented
  - adds (“stuffs”) extra <u>0</u> bit after it sees a sequence of five <u>1</u> bits
  - the worst-case overhead 20%
- PPP Byte Stuffing
  - byte-oriented
  - adds
  - the worst-case overhead 200%
- COBS: Consistent Overhead Byte Stuffing
  - byte-oriented
  - fixed and trivial overhead (<1%)

HDLC Byte Stuffing

- Sender: adds ("stuffs") extra 0 bit after it sees a sequence of five 1 bits
- Receiver:
  - Whenever it sees a sequence of five 1
    - if it follows by 0, remove it and process data
    - if it follow by 1, then it should be a frame boundary
- Input to sender
  - 01111110 01010101 01111110 ...
- Input to receiver
  - **01111110**011111**0**1001010101011111**0**10**01111110**...

PPP Byte Stuffing

- Sender:
  - replaces 0x7e (01111110) with “control escape” sequence **0x7d 0x5e**
  - replaces 0x7d with **0x7d 0x5d**
- Receiver:
  - if it sees 0x7d, it
    - removes 0x7d from stream
    - adds value of (next byte) XOR 0x20 to the stream
- Input to sender
  - 0x11 0x22 0x7d 0x7e 0x33
- Input to receiver
  - **0x7e** 0x11 0x22 **0x7d 0x5d 0x7d 0x5e** 0x33 **0x7e ...**

Consistent Overhead Byte Stuffing (COBS)

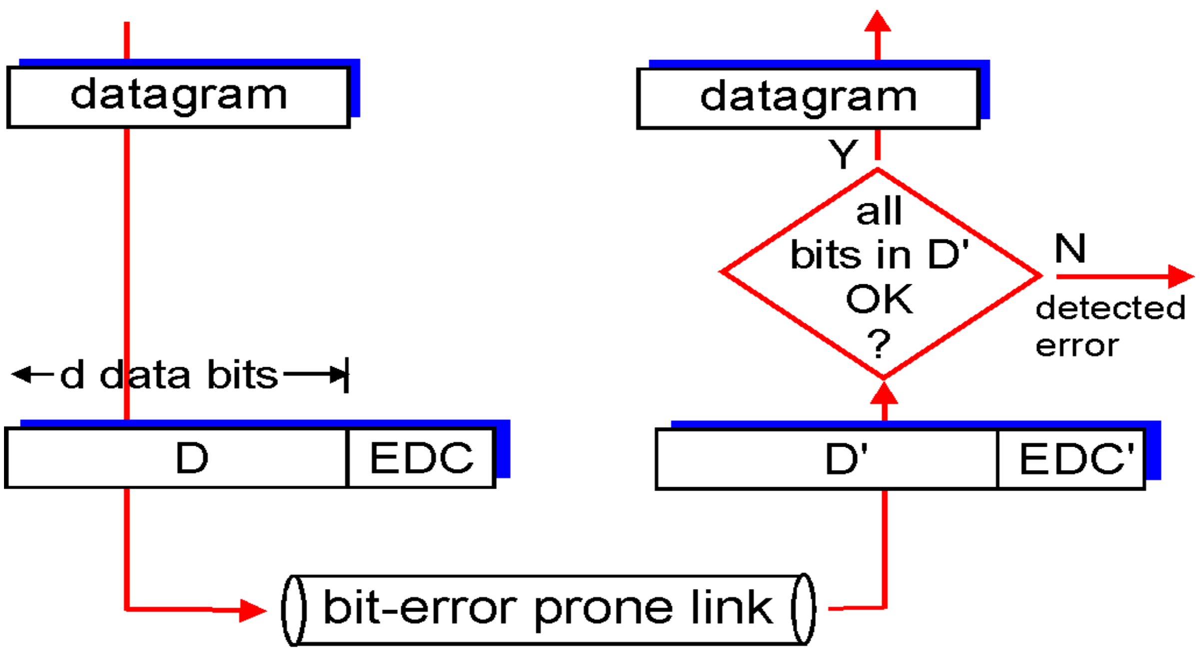
- User 0x00 as a frame delimiter
- Sender:
  - Adds byte representing the number non-zero bytes in the stream + 1
  - Adds non-zero bytes
  - Adds byte representing the number of non-zero bytes to follow + 1
  - Adds non-zero bytes
  - ...
- Receiver:
  - Reads the first byte X and processes the next X - 1 bytes as is
  - **If input length is not 0**
    - **Insert 0x00**
    - Reads the next byte X and processes the next X -1 bytes as is
    - **repeat**
- Input to sender
  - 0x11 0x22 0x00 0x33 0x00 0x00 ...
- Input to receiver
  - **0x00 0x03** 0x11 0x22 **0x02** 0x33 **0x01 0x01 0x00 ...**

Consistent Overhead Byte Stuffing (COBS)

- User 0x00 as a frame delimiter
- Sender:
  - Adds byte representing the number non-zero bytes in the stream + 1
  - Adds non-zero bytes
  - Adds byte representing the number of non-zero bytes to follow + 1
  - Adds non-zero bytes
  - ...
- Receiver:
  - Reads the first byte X and processes the next X - 1 byes as is
  - Insert 0x00
  - Reads the next byte X and processes the next X -1 bytes as is
  - ...
- Input to sender
  - 0x11 0x22 0x00 0x33 0x00 0x00 ...
- Input to receiver
  - **0x00 0x03** 0x11 0x22 **0x02** 0x33 **0x01 0x01 0x00 ...**

Error Detection

- EDC= Error Detection and Correction bits
- D = Data protected by error checking
- Error detection not 100% reliable!
  - protocol may miss some errors, though rarely
  - larger EDC field offers better detection and correction



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?*

Error Detection:

Cyclic Redundancy Check (CRC)

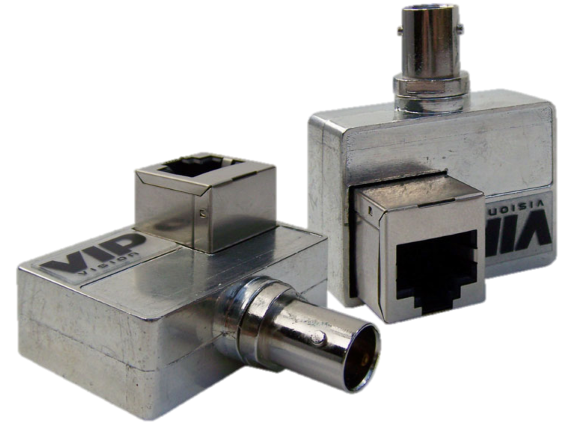
FYI

- Powerful error detection scheme
- Rather than addition, binary division is used → Finite Algebra Theory (Galois Fields)
- Can be easily implemented with small amount of hardware
  - Shift registers
  - XOR (for addition and subtraction)
- Typically an n-bit CRC applied to a data block of arbitrary length will detect any single error burst not longer than n bits and will detect a fraction $1 - 2^{-n}$ of all longer error bursts
  - **Does not** protect against intentional alteration of data

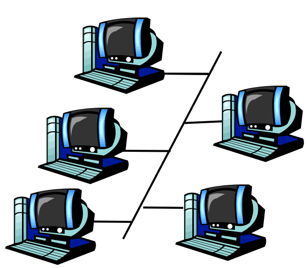
Multi-Access Links and Protocols

Multiple Access Links and Protocols

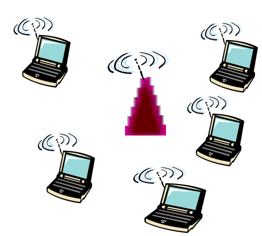
- 2 types of "links":
  - point-to-point
    - Ethernet over twisted pair cable
  - broadcast (shared wire or medium)
    - Ethernet over coax cable
    - 802.11 wireless LAN



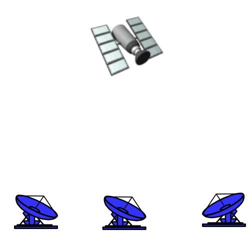
- For broadcast medium:
  - Need unique address for each interface
  - Need access control to the shared channel



shared wire (e.g., cabled Ethernet)



shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

Multiple Access Control

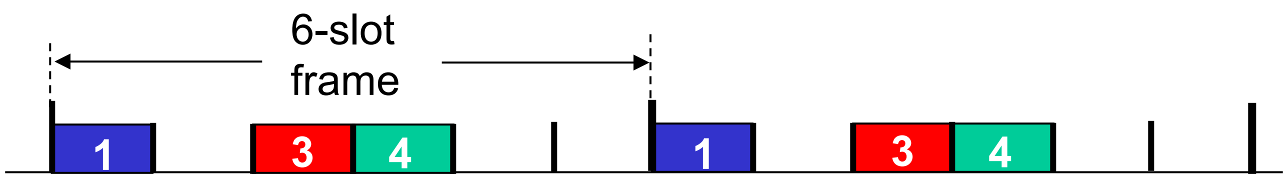
- determines *which* node can transmit *when*
- communication about channel sharing must use channel itself
- Ideal solution: *given* broadcast channel of rate R bps
  - If only one node wants to send: can send at rate R
  - If M nodes want to send: each can send at rate R/M
  - simple, no central controller
- 3 classes of solutions:
  - Channel partitioning
  - Taking turns: coordinated access to avoid collision
  - Random Access: no coordination; avoid collisions if possible, detect and resolve collisions in case them occur

(1) Channel Partitioning: TDMA, FDMA

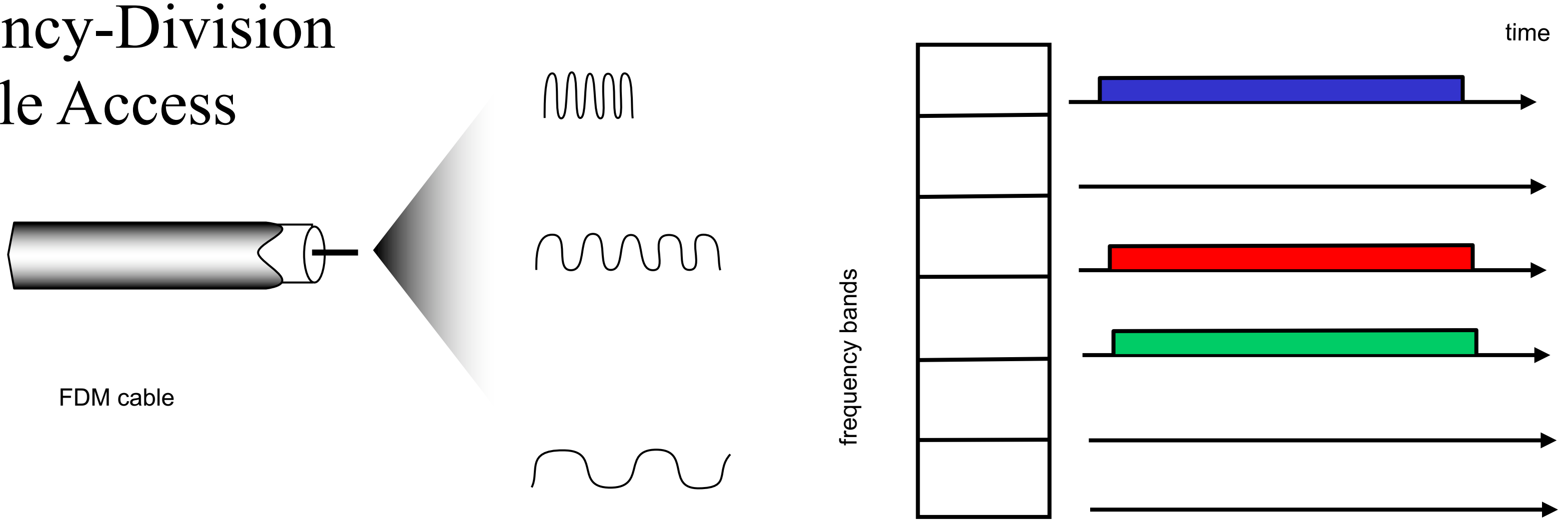
FYI

- Assuming 6 transmitters sharing one channel; 1, 3, 4 are sending; 2, 5, 6 idle

Time-Division Multiple Access

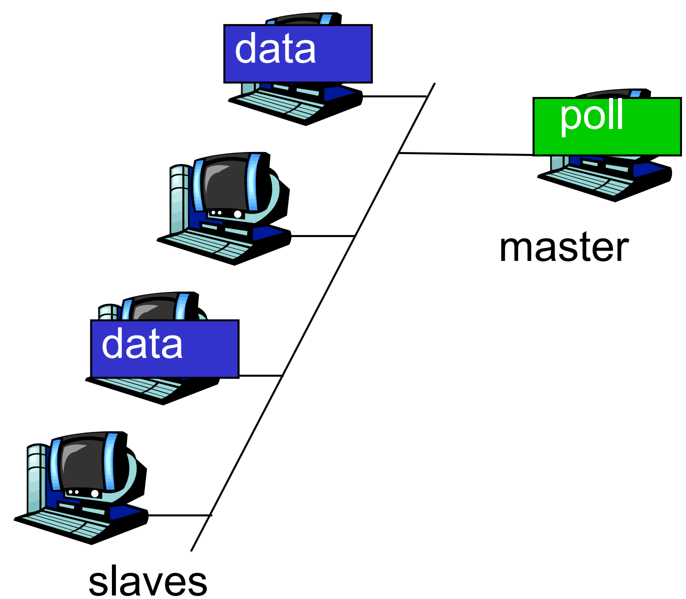


Frequency-Division Multiple Access



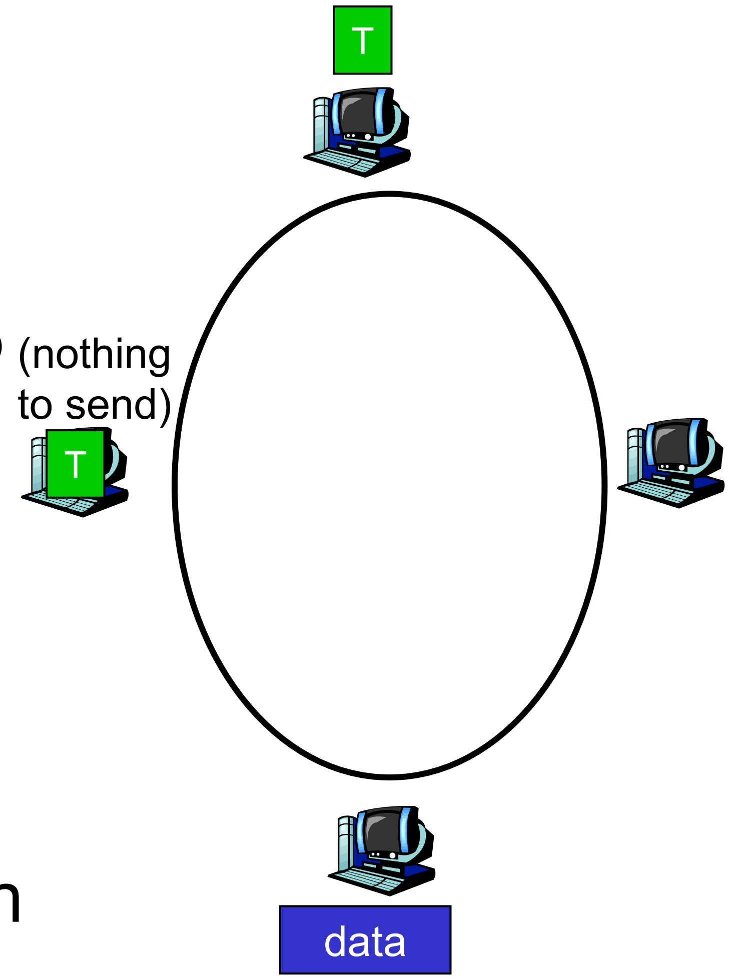
(2) "Taking Turns" MAC protocol (I)

- *On-demand* channel allocation
- Polling:
  - master node asks slave nodes to transmit in turn
  - Concerns
    - polling overhead
    - Latency
    - single point of failure (master)



(2) "Taking Turns" MAC protocol (II)

- Token passing
  - One token message passed from one node to next sequentially
  - whoever gets the token can send one data frame, then pas token to next node
- Concerns:
  - latency
  - single point of failure (token)
- A master station generates the token and monitors its circulation
  - If token is lost, generate a new one

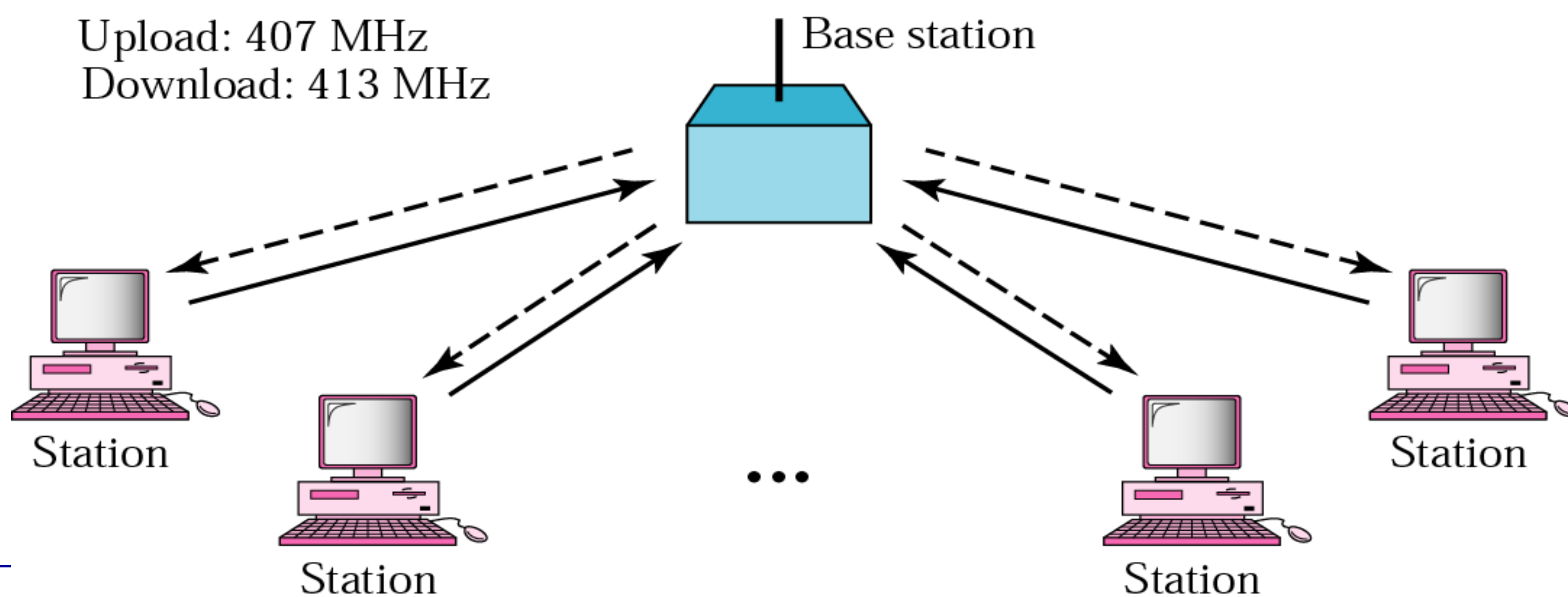


(3) Random Access protocols

- When node has packet to send
  - transmit at full channel data rate R
  - *no a priori coordination among nodes*
- When collide (2 or more nodes transmitting at the same time), random access protocol specifies:
  - how to detect a collision
  - how to recover from a collision
- Examples of random access MAC protocols:
  - ALOHA, slotted ALOHA
  - CSMA/CD, CSMA/CA

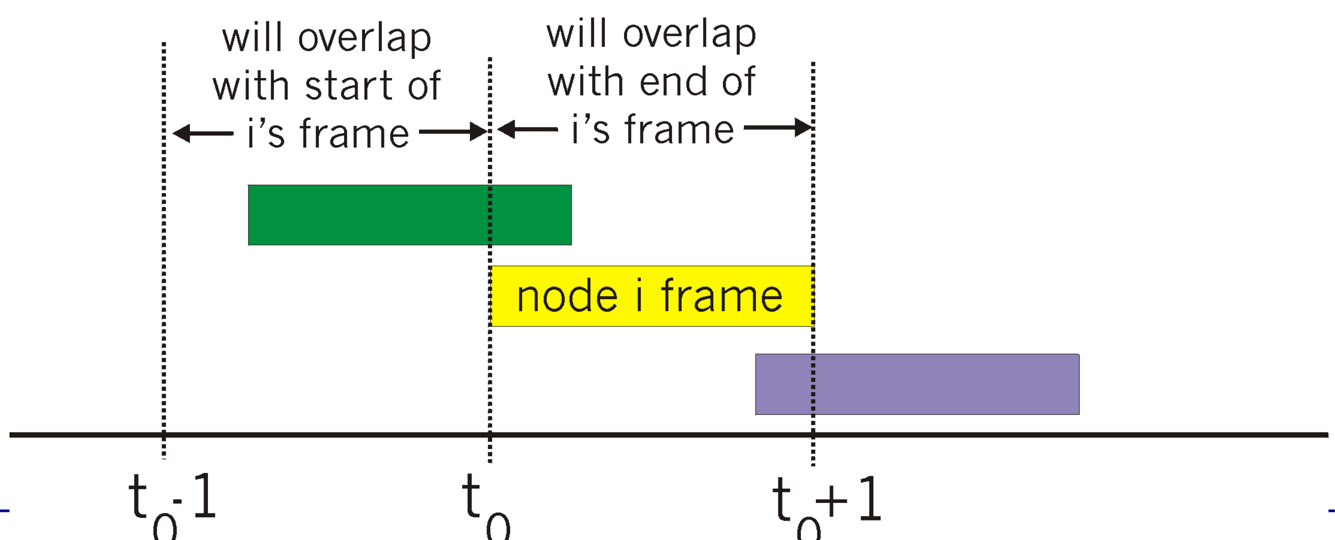
ALOHA History

- Developed by Norm Abramson at Univ. of Hawaii in 1970
  - The world's first wireless packet-switched network
- Why ALOHA
  - mountainous islands → wire-based network infeasible
  - Radio channel → high error rate → centralized control infeasible
- Upload channel: contention-based random access
- Download channel: rebroadcasting all received packets



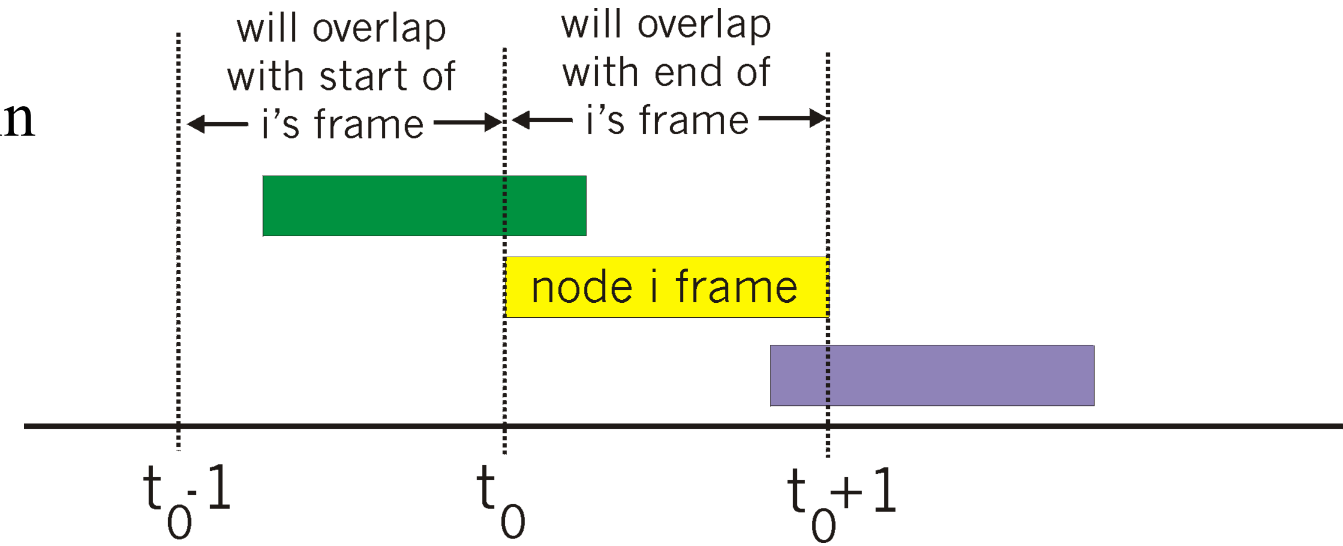
ALOHA

- If a node has data to send, send the whole frame immediately
  - If collision: retransmits the frame again with the probability *p*
- collision probability: assume all frames of same size, frame sent at t_0 may collide with other frames sent in $[t_0-1, t_0+1]$



probability of a successful transmission

Assuming N nodes in the network:
p = probability of a node transmitting



P(success by a given node) = P(node transmits) •

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

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

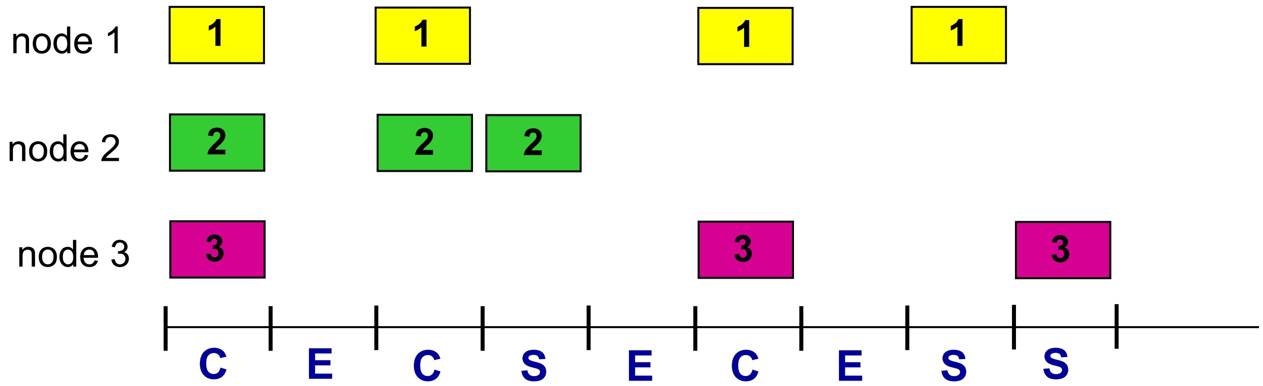
$$= p \bullet (1\text{-}p)^{N\text{-}1} \bullet (1\text{-}p)^{N\text{-}1}$$

P (success by any node) = $N\, p \cdot (1\text{-}p)^{2(N\text{-}1)}$, choosing optimum p as n -> ∞

$$= 1/(2e) = 0.18$$

Slotted Aloha

- Assumptions:
  - Divide time into equal size slots (= frame transmission time)
  - clocks in all nodes are synchronized
  - If 2 or more nodes collide in one slot, all nodes detect collision
- Operations:
  - Each node transmits only at <u>beginning</u> of next slot
  - If no collision, node can send new frame in next slot
  - If collision, retransmit in each subsequent slots with probability p, until succeed



S(uccess), C(ollision), E(mpty) slots

Slotted Aloha efficiency

Q: what is the max fraction of successful slots?

- N nodes, each transmits in a slot with probability p

prob. successful transmission S:

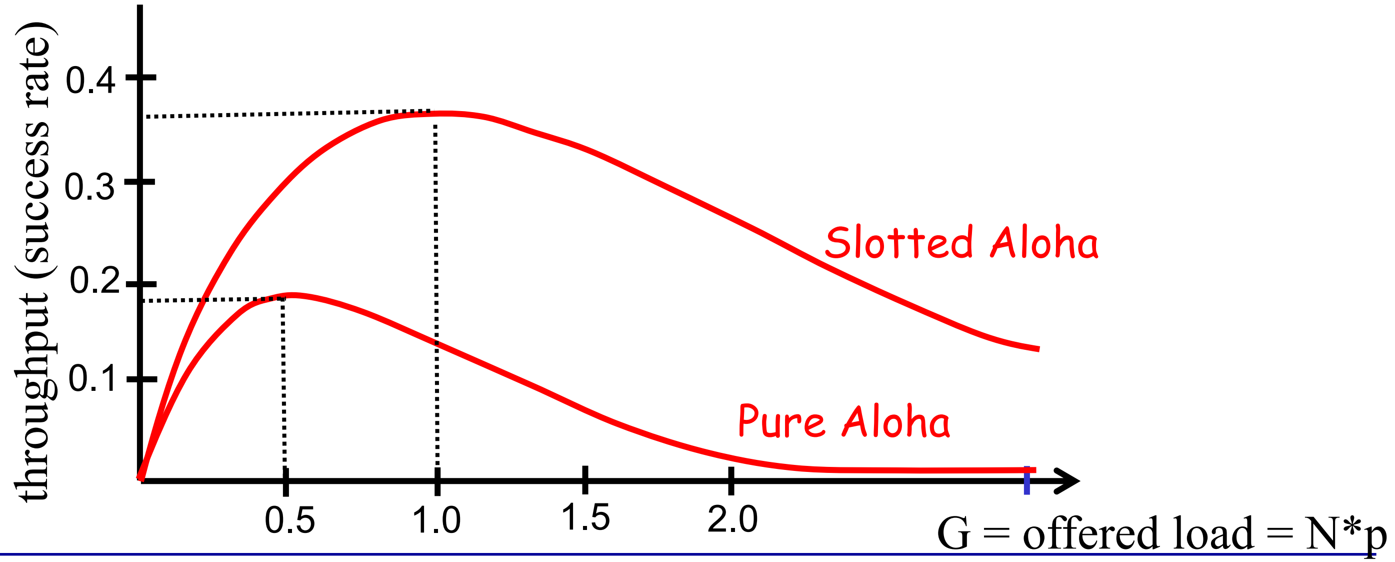
for a given node: S= p (1-p)(N-1)

by any of the N nodes:

S = P(only one transmits)

= N p (1-p)(N-1), choosing optimal p as $n \rightarrow \infty$

= 1/e = 0.37



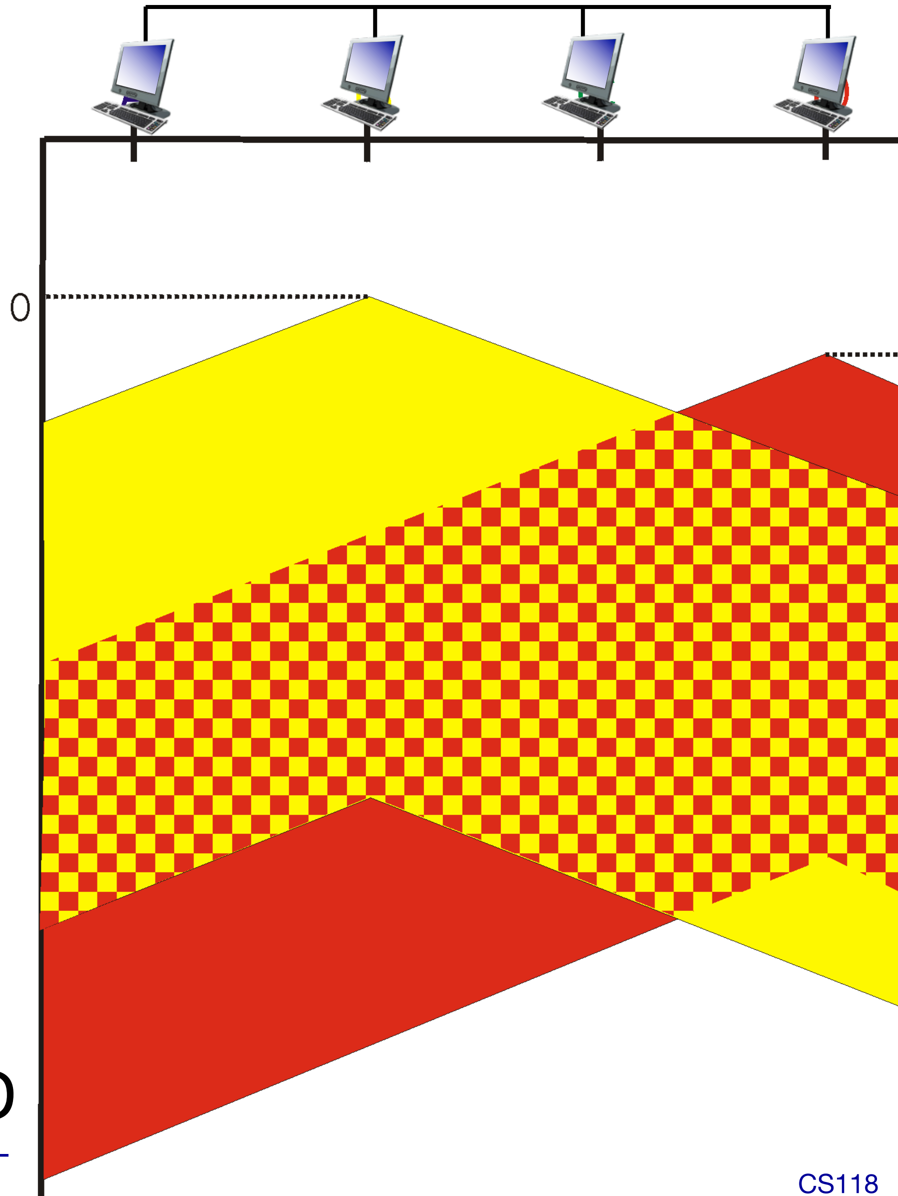
CSMA: Carrier Sense Multiple Access

- listen before transmit
- If channel *sensed* idle: transmit
- If channel sensed busy, wait until it becomes idle:

  1-persistent CSMA: retry immediately

  p-persistent CSMA: retry immediately with probability p

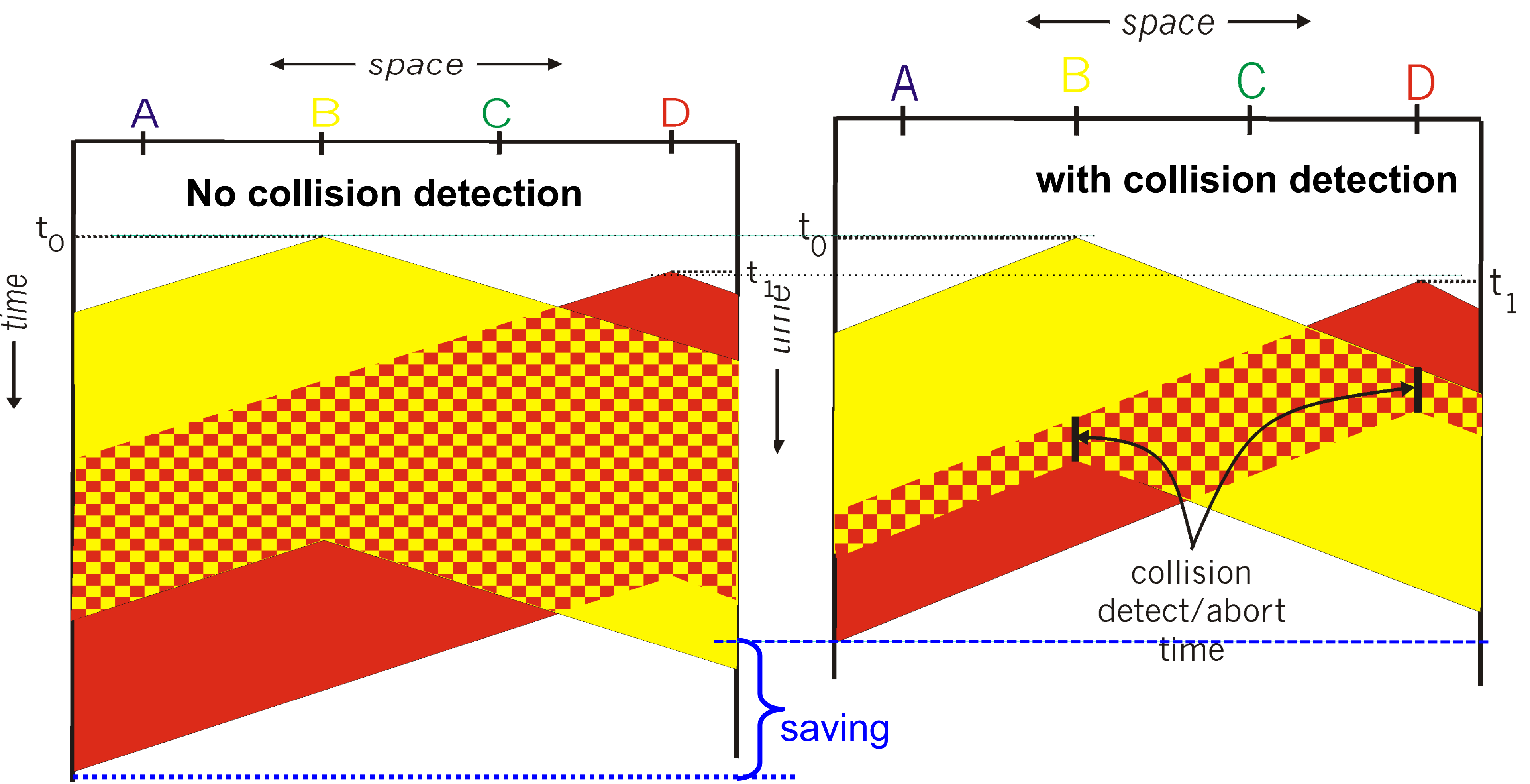
  Non-persistent CSMA: retry after a random interval
- collisions still possible:
  - Chance of collision goes up with distance between nodes



To cut the loss early: CSMA/CD

CSMA/CD (Collision Detection)

- Collision Detection: compare transmitted with received signals
- Abort collided transmissions



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 $\{0,1,2, \ldots, 2^m-1\}$.
   - NIC waits K × 512 bit times, returns to Step 2
   - more collisions, exponentially longer backoff interval

CSMA/CD efficiency

important

- T_{prop} = max. propagation delay between any 2 nodes
- T_{trans} = time to transmit a max-size frame

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

- Efficiency goes to 1
  - as T_{prop} goes to 0
  - as T_{trans} goes to infinity
- much better performance than ALOHA, and still decentralized and simple

Summary of MAC protocols

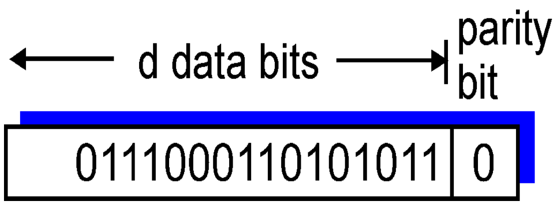
- channel partitioning
  - Time Division, Frequency Division
- taking turns
  - polling from central site, or token passing
- random access
  - ALOHA, S-ALOHA
  - CSMA: Carrier Sensing in MultiAccess: easy in some case (wire), harder in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11

Error Detection: Parity Checking

FYI

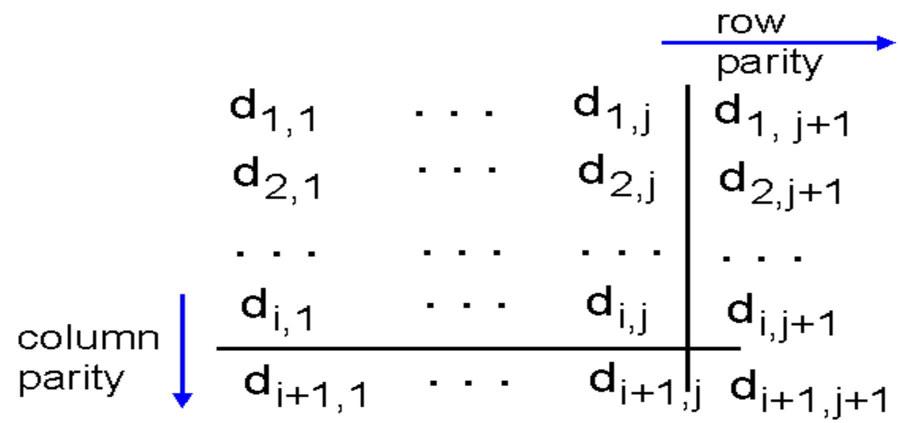
Single Bit Parity:

Detect single bit errors



Two Dimensional Bit Parity:

Detect and *correct* single bit errors



consider a data frame as a m×n matrix

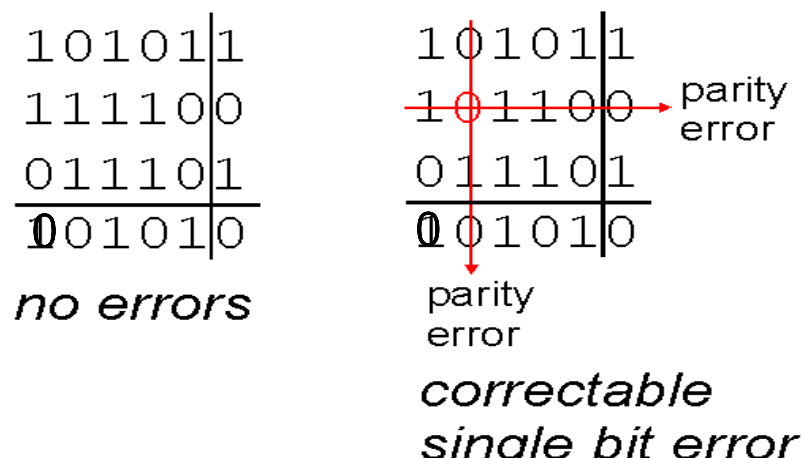
A parity bit for each row

→ n-bit checksum

and

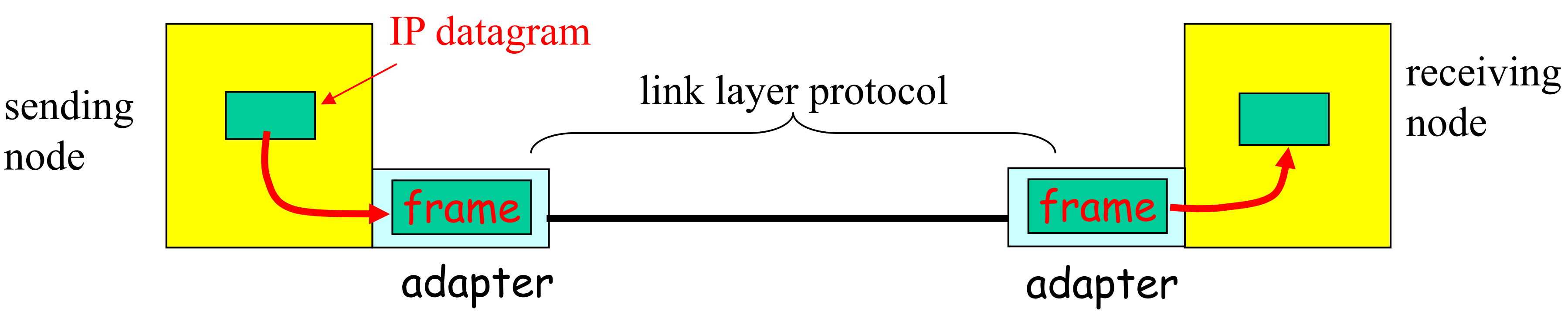
A parity bit for each column

→ m-bit checksum



Sending data over a physical link

- Framing: marking the beginning & end of a data chunk
- bit error detection
- (in some layer-2 protocols) Flow Control: pacing between sender and receivers
- (in some layer-2 protocol) reliable transmission



Ethernet MAC protocol: CSMA/CD

A: sense channel, wait if necessary, when channel is idle, transmit and monitor the channel;

If detect collision

then {

abort and send jam signal;
update collision-count (n++);
delay for K slots (1 slot = 512bits transmission time)
goto A
}

else {finish sending the frame; reset collision-count (n = 0)}

Jam Signal (48 bits): make all other transmitters aware of collision

Exponential Backoff algorithm:

- first collision (n=1): choose K from {0, 1}
- second collision (n =2): choose K from {0, 1, 2, 3}…
- after 10 collisions (n=10), choose K from {0,1,2,3,4,…,1023}

Point to Point Data Link Control

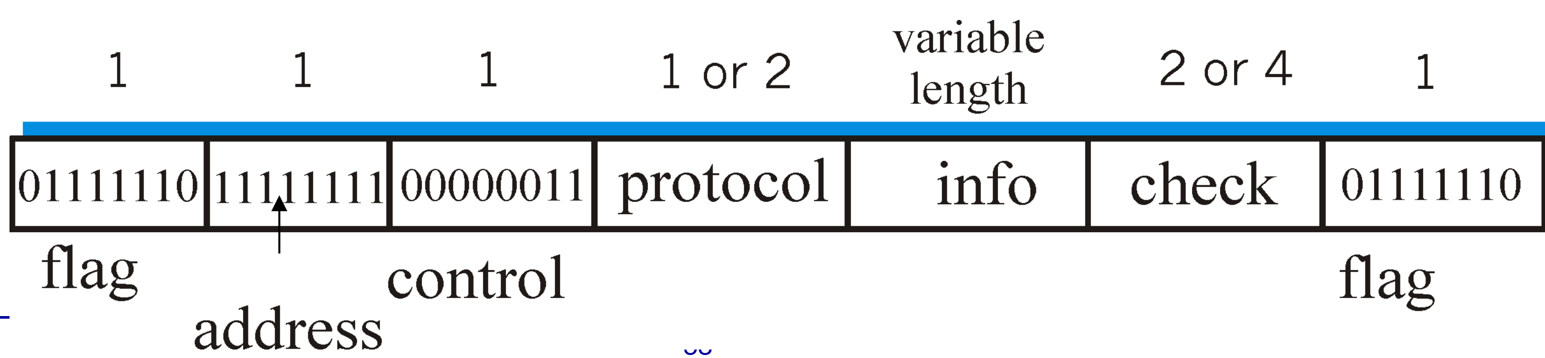
- One sender, one receiver, one link
- One example: PPP (point-to-point protocol)

  [RFC 1661, 1662]

  - Can carry data from any layer-3 protocol; ability to de-multiplex upwards
  - error detection
  - connection liveness: detect & signal link failure to network layer
  - network layer address negotiation: endpoint can learn/configure each other's network address

PPP Data Frame

- Flag: delimiter (framing)
- Address: does nothing (only one value)
- Control: does nothing
  - possible multiple control values in the future
- Protocol: upper layer protocol to which frame is delivered (e.g. PPP-LCP, IP, etc.)
- info: upper layer data being carried
- check: cyclic redundancy check for error detection



Ethernet Frame length limit

- All packets/frames have an upper bound on length
- Ethernet has a lower bound for reliable collision detection:
  - cable max length 2500m ⇨ $2P_{delay} = 51.2\mu\sec$
- transceiver can send & listen at the same time; if received data stream differs from the one transmitted → collision
  - to detect collision: sender must be transmitting when garbled signal propagates back
  - minimum frame = 64bytes = 512bits, take 51$\mu\sec$ to transmit at 10Mbps