CSCI-1680 - Computer Networks

Link Layer II: MAC - Media Access Control

Chen Avin

Based partly on lecture notes by David Mazières, Phil Levis, John Jannotti, Peterson & Davie, Rodrigo Fonseca

Administrivia

- Snowcast deadline tomorrow 10pm
- Homework I out later today, due next Thursday, Sep 26th 4pm

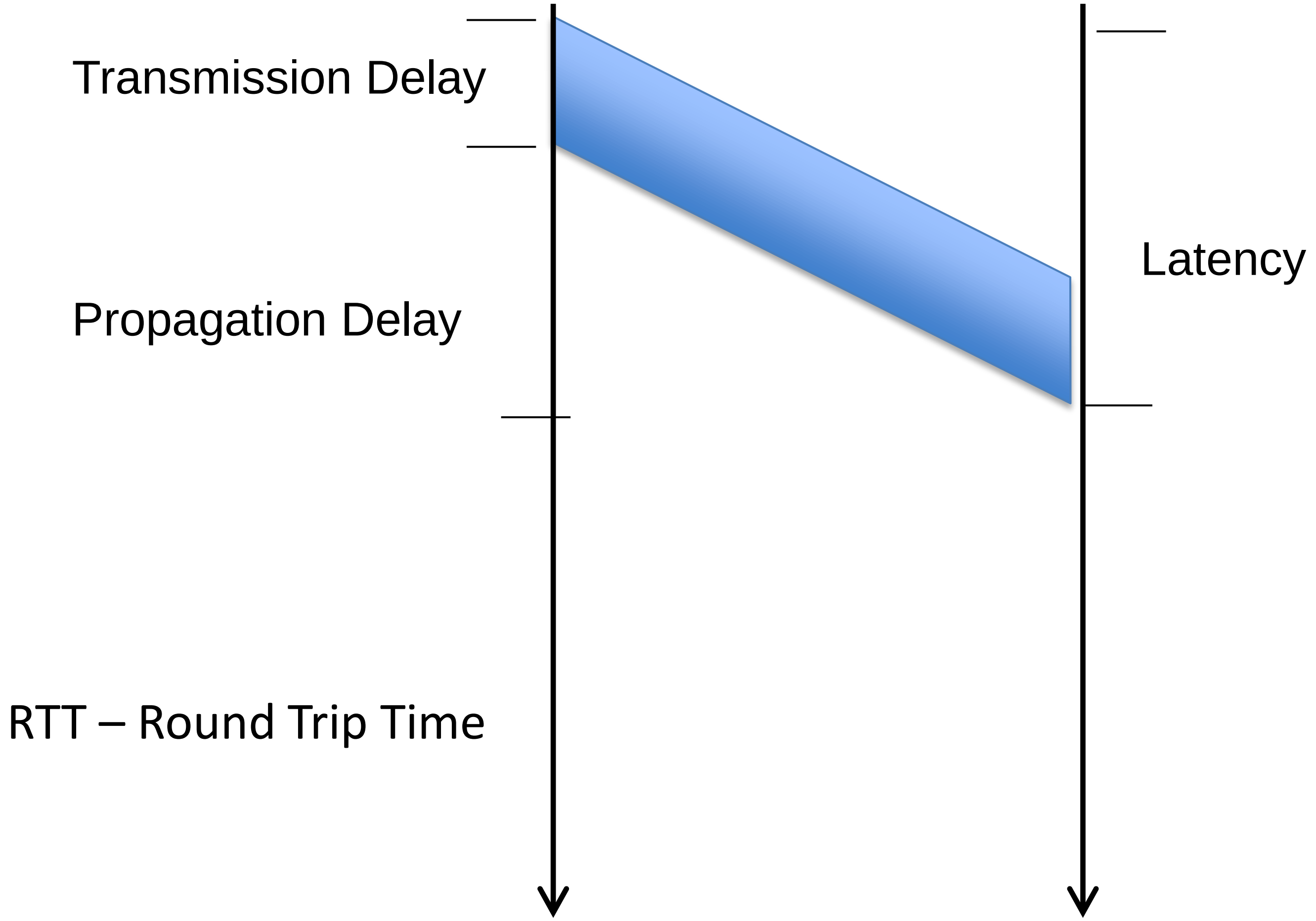
Today: Link Layer (cont.)

- Framing
- Reliability
  - Error correction
  - **Sliding window**
- **Medium Access Control**
- **Case study: Ethernet**

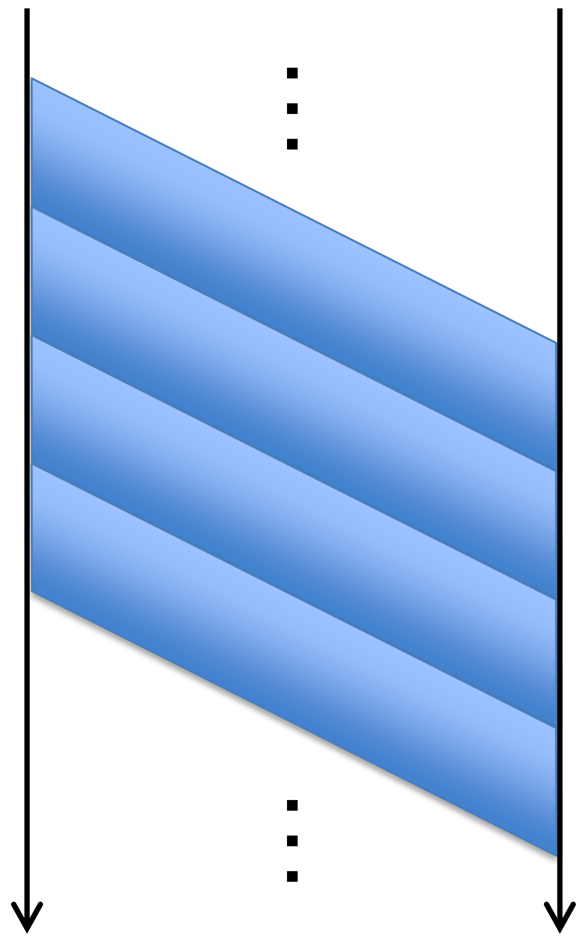
Reliability and Performance

Bandwidth, Throughput, Delay

Sending Frames Across



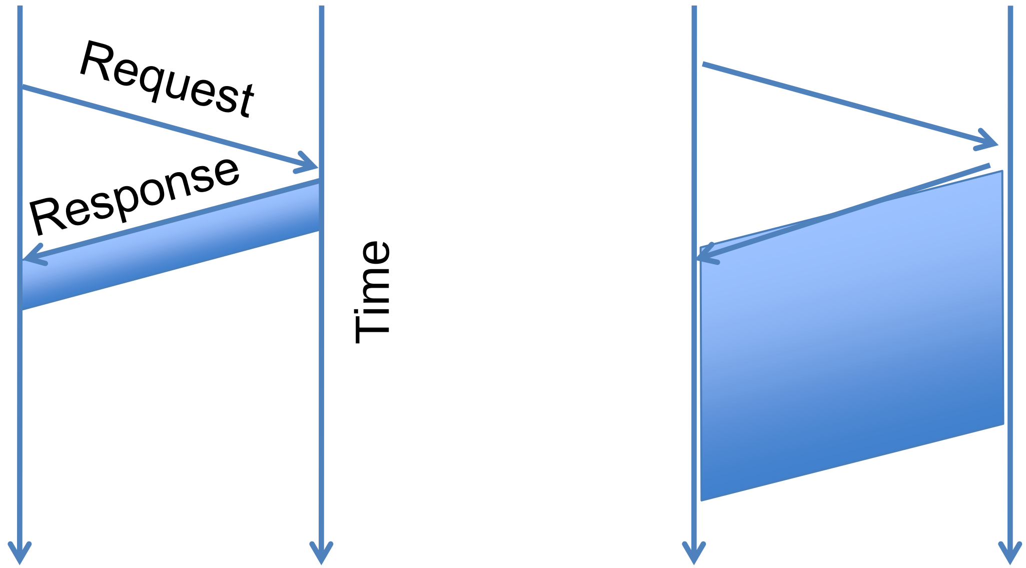
Sending Frames Across



Throughput: bits / s

Which matters most, bandwidth or delay?

- How much data can we send during one RTT?
- *E.g.*, send request, receive file



- For small transfers, latency more important, for bulk, throughput more important

Performance Metrics

- **Throughput - Number of bits received/unit of time**
  - *e.g.* 10Mbps
- **Goodput - *Useful* bits received per unit of time**
- **Latency – How long for message to cross network**
  - Process + Queue + Transmit + Propagation
- **Jitter – Variation in latency**

Latency

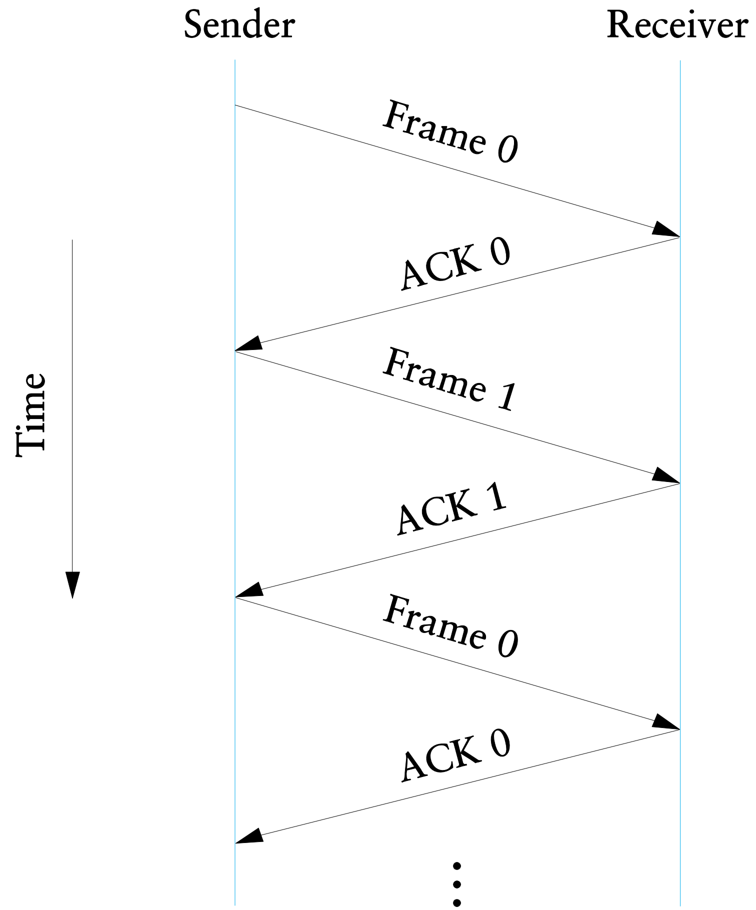
- **Processing**
  - Per message, small, limits throughput
  - *e.g.* $\frac{100Mb}{s} \times \frac{pkt}{1500B} \times \frac{B}{8b} \approx 8{,}333 pkt/s$ or 120μs/pkt
- **Queue**
  - Highly variable, offered load vs outgoing b/w
- **Transmission**
  - Size/Bandwidth
- **Propagation**
  - Distance/Speed of Light

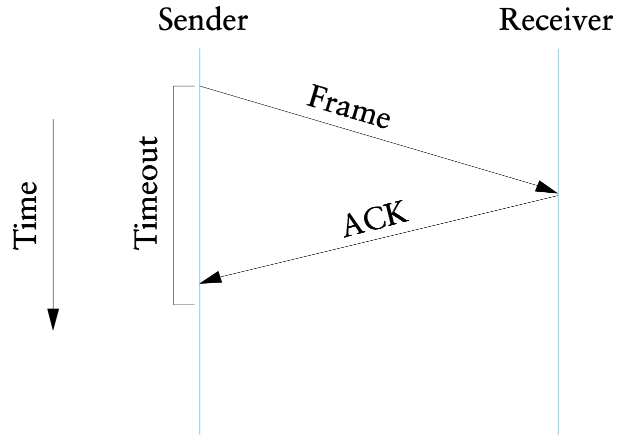
Reliable Delivery

- **Several sources of errors in transmission**
- **Error detection can discard bad frames**
- **Problem: if bad packets are lost, how can we ensure reliable delivery?**
  – Exactly-once semantics = at least once + at most once

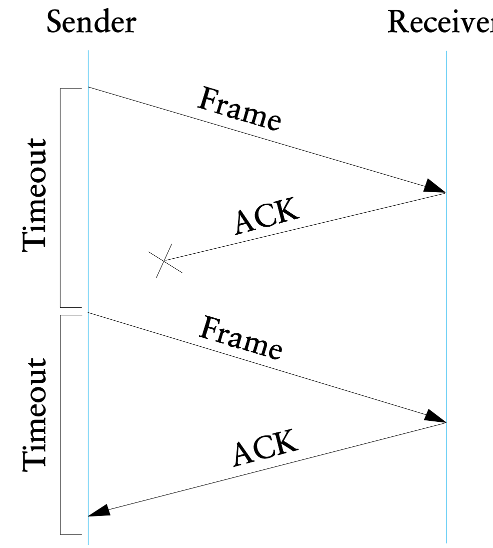
At Least Once Semantics

- **How can the sender know packet arrived *at least once*?**
  - Acknowledgments + Timeout
  - ARQ (automatic repeat request)
- **Stop and Wait Protocol**
  - S: Send packet, wait
  - R: Receive packet, send ACK
  - S: Receive ACK, send next packet
  - S: No ACK, timeout and retransmit

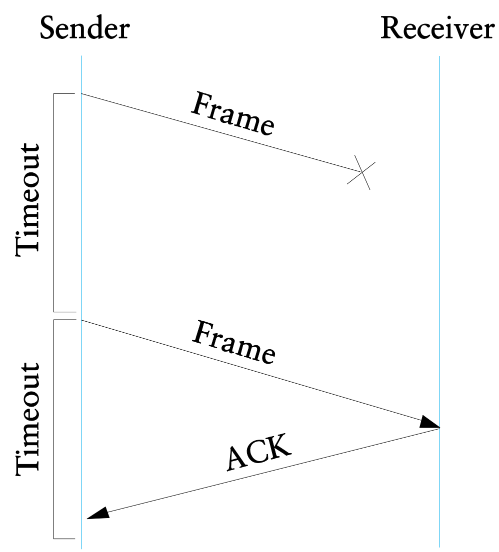

Sender
Receiver
Frame 0
ACK 0
Frame 1
ACK 1
Frame 0
ACK 0
Time


Sender
Receiver
Time
Timeout
Frame
ACK

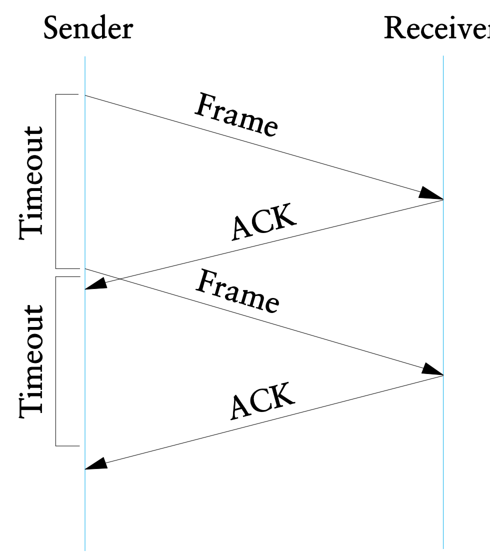
(a)


Sender
Receiver
Timeout
Frame
ACK
Timeout
Frame
ACK

(c)


Sender
Receiver
Timeout
Frame
Timeout
Frame
ACK

(b)

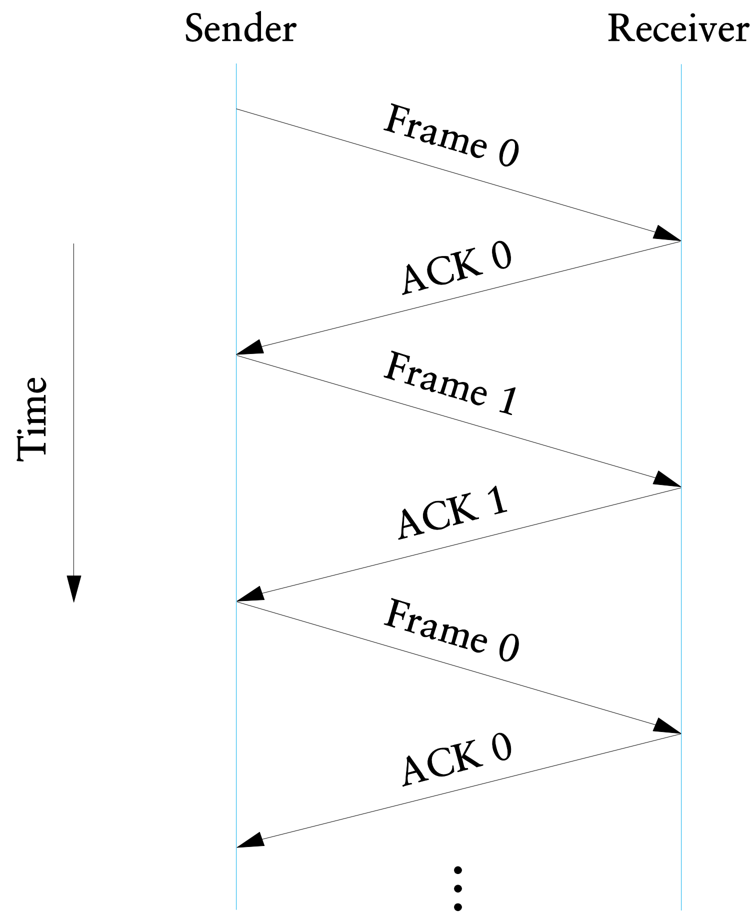

Sender
Receiver
Timeout
Frame
ACK
Timeout
Frame
ACK

(d)

Stop and Wait Problems

- Duplicate data
- Duplicate acks
- Slow (channel idle most of the time!)
- May be difficult to set the timeout value

Duplicate data: adding sequence numbers

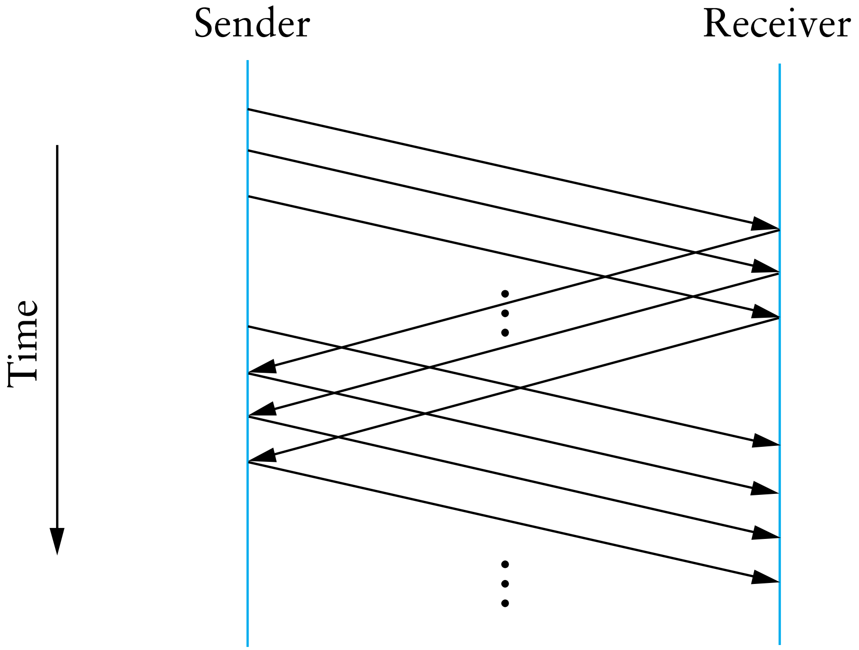


At Most Once Semantics

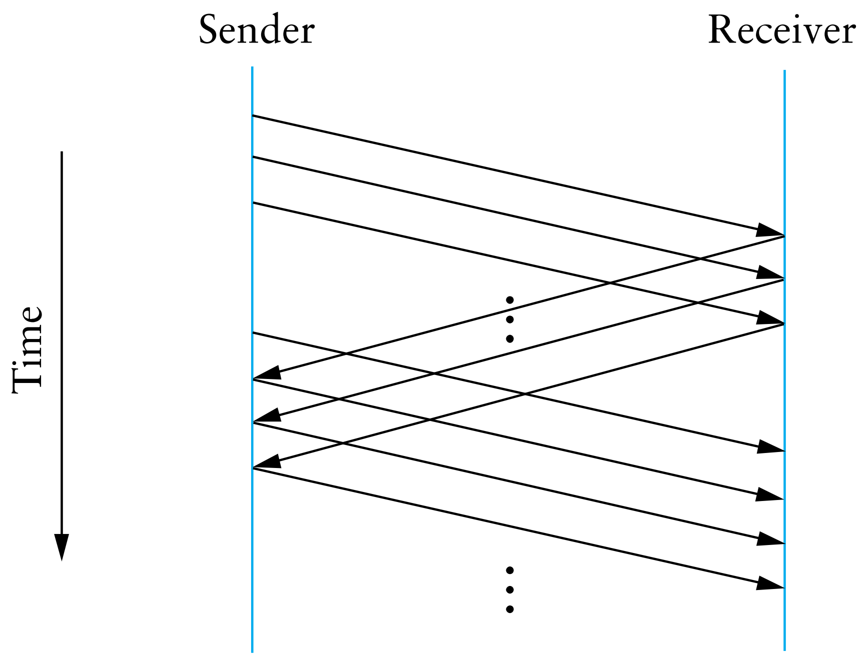
- **How to avoid duplicates?**
  - Uniquely identify each packet
  - Have receiver and sender remember
- **Stop and Wait: add 1 bit to the header**
  - Why is it enough?
- **Do we need to check errors in ACKs?**

Going faster: sliding window protocol

- **Still have the problem of keeping pipe full**
  - Generalize approach with > 1-bit counter
  - Allow multiple outstanding (unACKed) frames
  - Upper bound on unACKed frames, called *window*

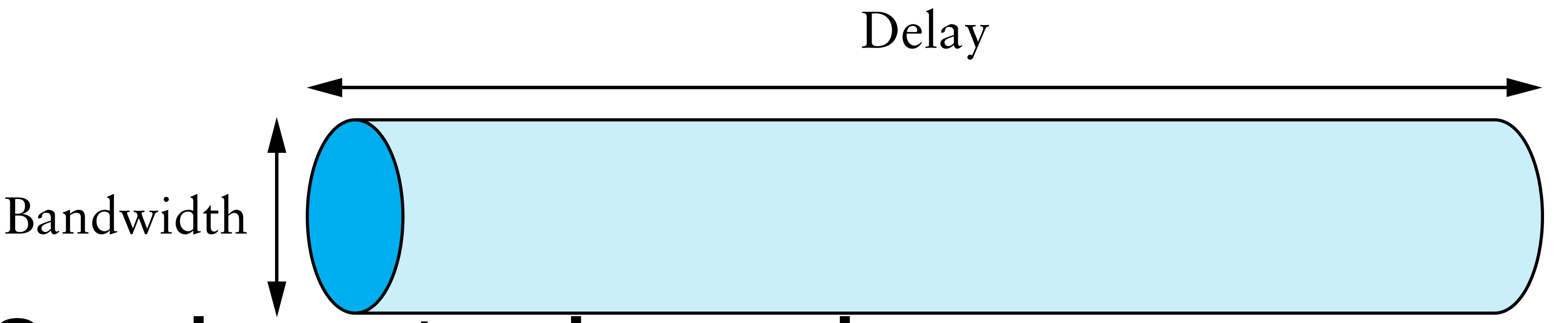


How big should the window be?



- **How many bytes can we transmit in one RTT?**
  – BW B/s x RTT s => "Bandwidth-Delay Product"

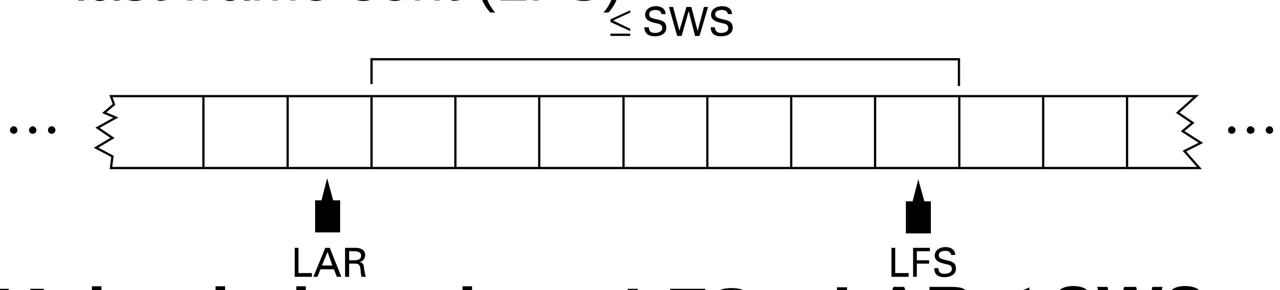
Maximizing Throughput



- **Can view network as a pipe**
  - For full utilization want bytes in flight ≥ bandwidth × delay
  - But don't want to overload the network (future lectures)
- **What if protocol doesn't involve bulk transfer?**
  - Get throughput through concurrency – service multiple clients simultaneously

Sliding Window Sender

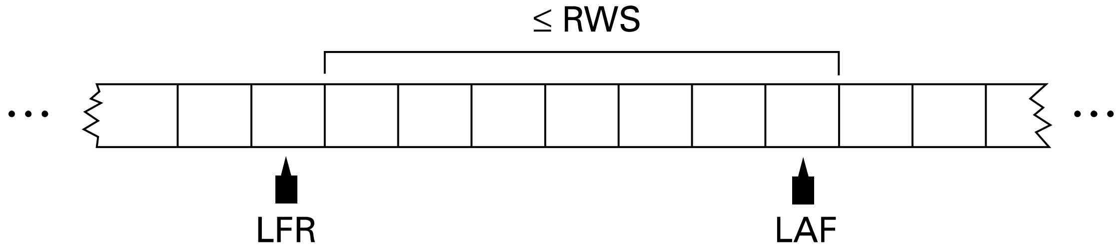
- **Assign sequence number (SeqNum) to each frame**
- **Maintain three state variables**
  - send window size (SWS)
  - last acknowledgment received (LAR)
  - last frame sent (LFS)



- **Maintain invariant: LFS – LAR ≤ SWS**
- **Advance LAR when ACK arrives**
- **Buffer up to SWS frames**

Sliding Window Receiver

- **Maintain three state variables:**
  - receive window size (RWS)
  - largest acceptable frame (LAF)
  - last frame received (LFR)



- **Maintain invariant: LAF – LFR ≤ RWS**
- **Frame SeqNum arrives:**
  - if LFR < SeqNum ≤ LAF, accept
  - if SeqNum ≤ LFR or SeqNum > LAF, discard
- **Send *cumulative* ACKs**
  - other options: selective ACK, NAK.

Tuning Send Window

- **How big should SWS be?**
  - "Fill the pipe"
- **How big should RWS be?**
  - $1 \leq RWS \leq SWS$
- **What to do on a timeout?**
- **How many distinct sequence numbers needed?**
  - SWS can't be more more than half of the space of valid seq#s.

Example

- SWS = RWS = 5. Are 6 seq #s enough?
- Sender sends 0,1,2,3,4
- All acks are lost
- Sender sends 0,1,2,3,4 again
- …

Summary

- **Want exactly once**
  - At least once: acks + timeouts + retransmissions
  - At most once: sequence numbers
- **Want efficiency**
  - Sliding window

MAC: Media Access Control

Media Access Control

- **Control access to shared physical medium**
  - E.g., who can talk when?
  - If everyone talks at once, no one hears anything
  - Two or more transmitting nodes ➜ "collision",
  - Job of the Link Layer
- **Two conflicting goals**
  - Maximize utilization when one node sending
  - Approach 1/N allocation when N nodes sending

Different Approaches

- **Partitioned Access**
  - Time Division Multiple Access (TDMA)
  - Frequency Division Multiple Access (FDMA)
  - Code Division Multiple Access (CDMA)
- **Random Access**
  - ALOHA/ Slotted ALOHA
  - Carrier Sense Multiple Access / Collision Detection (CSMA/CD)
  - Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA)
  - RTS/CTS (Request to Send/Clear to Send)
  - Token-based

Slotted ALOHA

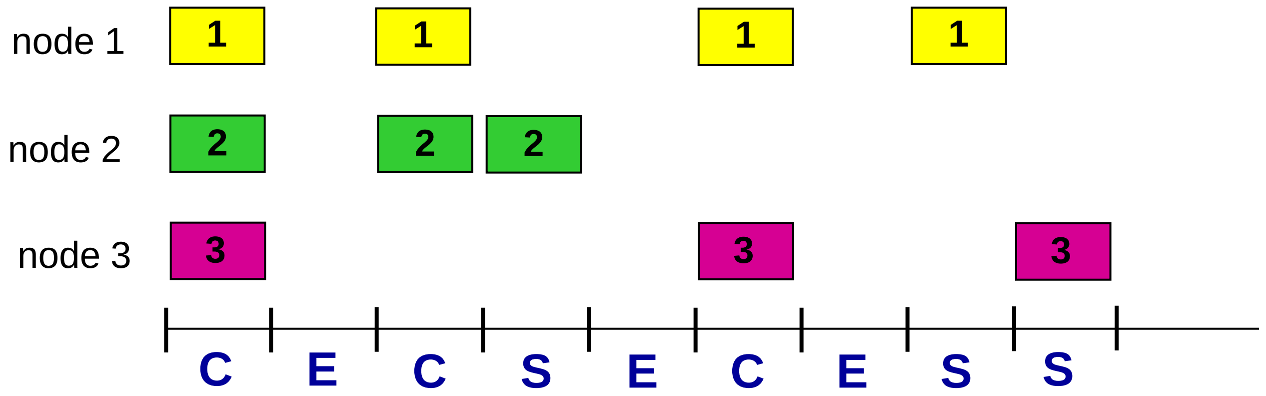
assumptions:

- **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

Slide from: "Computer Networking: A Top Down Approach" - 6th edition

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:

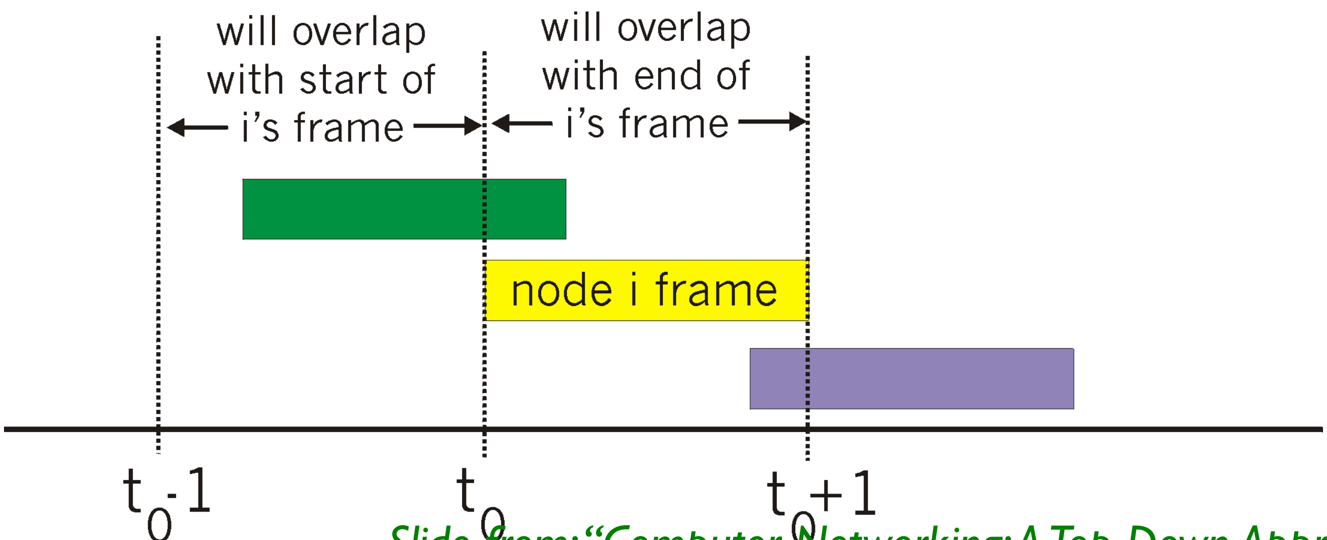
max efficiency = 1/e = .37

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

!

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 $[t_0-1, t_0+1]$

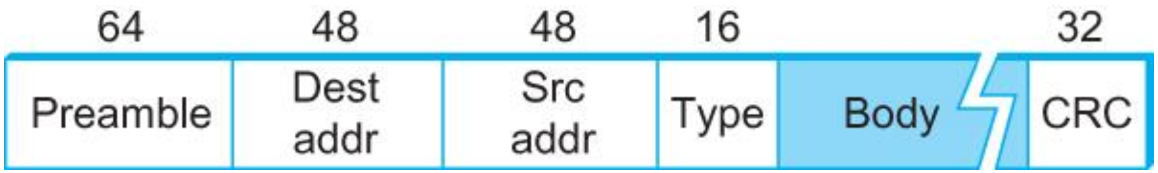


CSMA/CD (collision detection)

- **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**

Case Study: Ethernet (802.3)

- **Dominant wired LAN technology**
  - 10BASE2, 10BASE5 (Vampire Taps)
  - 10BASET, 100BASE-TX, 1000BASE-T, 10GBASE-T,…
- **Both Physical and Link Layer specification**
- **CSMA/CD**
  - Carrier Sense / Multiple Access / Collision Detection
- **Frame Format (Manchester Encoding):**



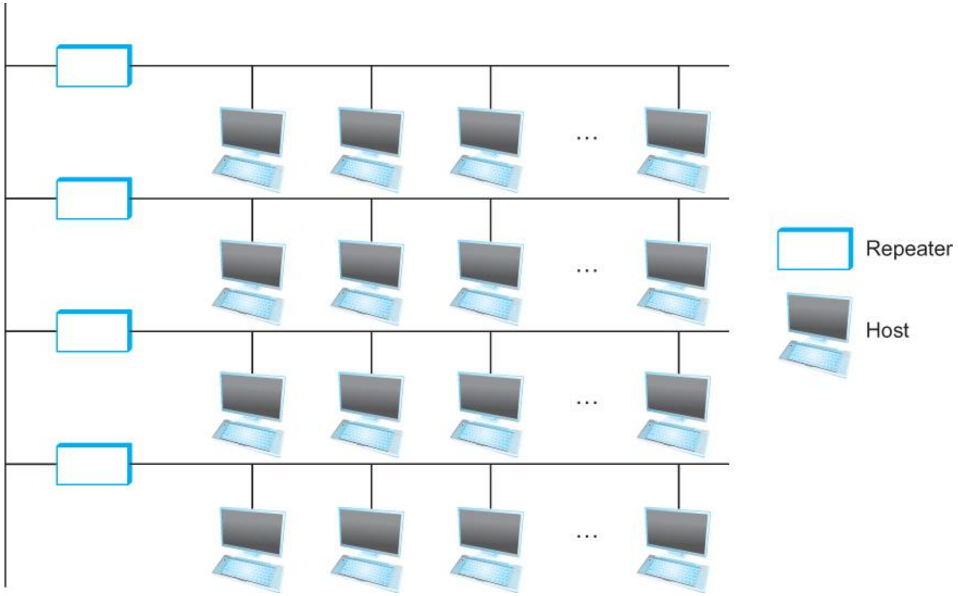
Ethernet Addressing

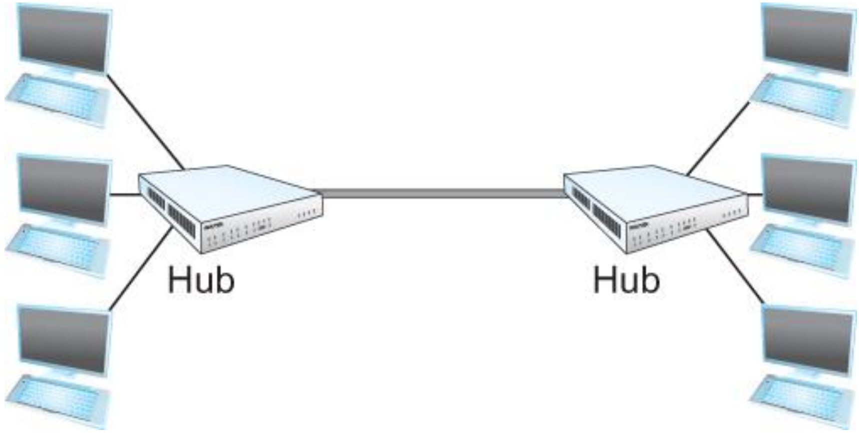
- **Globally unique, 48-bit unicast address per adapter**
  - Example: 00:1c:43:00:3d:09 (Samsung adapter)
  - 24 msb: organization
  - http://standards.ieee.org/develop/regauth/oui/oui.txt
- **Broadcast address: all 1s**
- **Multicast address: first bit 1**
- **Adapter can work in *promiscuous* mode**

Ethernet MAC: CSMA/CD

- **Problem: shared medium**
  - 10Mbps: 2500m, with 4 repeaters at 500m
- **Transmit algorithm**
  - If line is idle, transmit immediately
  - Upper bound message size of 1500 bytes
  - Must wait 9.6μs (96-bit time) between back to back frames
    - (Old limit) To give time to switch from tx to rx mode
  - If line is busy: wait until idle and transmit immediately

Ethernet Topologies

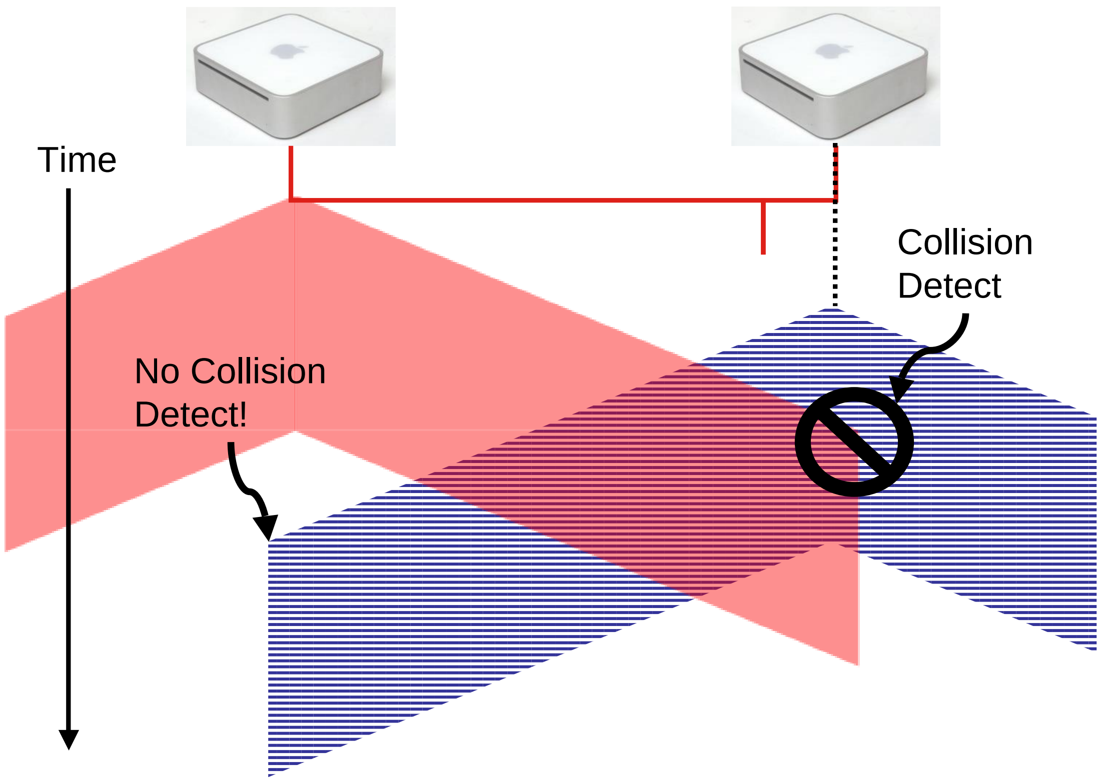




Handling Collisions

- **Collision detection (10Base2 Ethernet)**
  - Uses Manchester encoding. Why does that help?
  - Constant average voltage unless multiple transmitters
- **If collision**
  - Jam for 32 bits, then stop transmitting frame
- **Collision detection constrains protocol**
  - Imposes min. packet size (64 bytes or 512 bits)
  - Imposes maximum network diameter (2500m)
  - Must ensure transmission time ≥ 2x propagation delay (why?)

Collision Detection



- **Without minimum frame length, might not detect collision**

When to transmit again?

- **Delay and try again: exponential backoff**
- ***n*th time: $k \times 51.2\mu s$, for $k = U\{0..2^{\min(n,10)}-1\}$**
  - 1st time: 0 or 51.2μs
  - 2nd time: 0, 51.2, 102.4, or 153.6μs
- **Give up after several times (usually 16)**

Capture Effect

- Exponential backoff leads to self-adaptive use of channel
- A and B are trying to transmit, and collide
- Both will back off either 0 or 51.2µs
- Say A wins.
- Next time, collide again.
  - A will wait between 0 or 1 slots
  - B will wait between 0, 1, 2, or 3 slots
- …

CSMA/CD efficiency

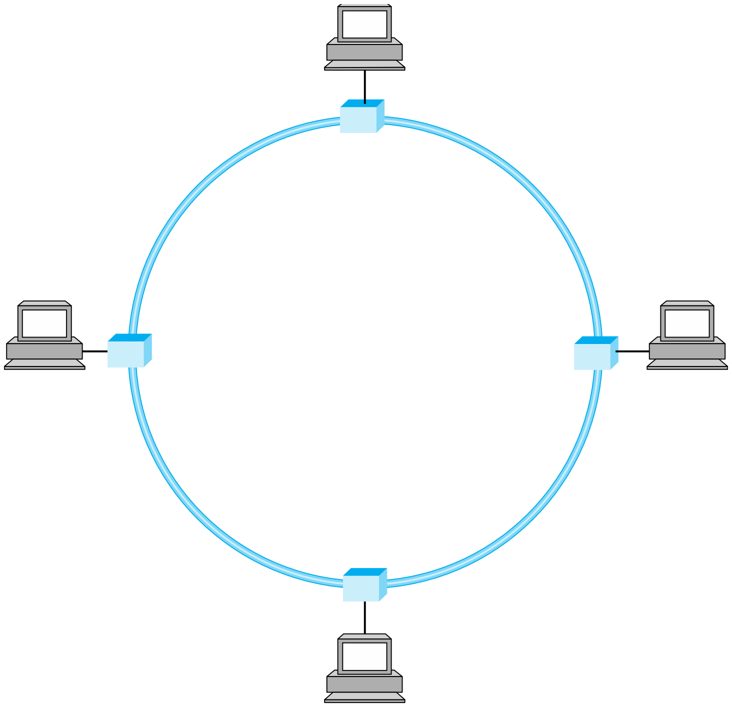
- **T_{prop} = max prop delay between 2 nodes in LAN**
- **t_{trans} = time to transmit max-size frame**

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

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

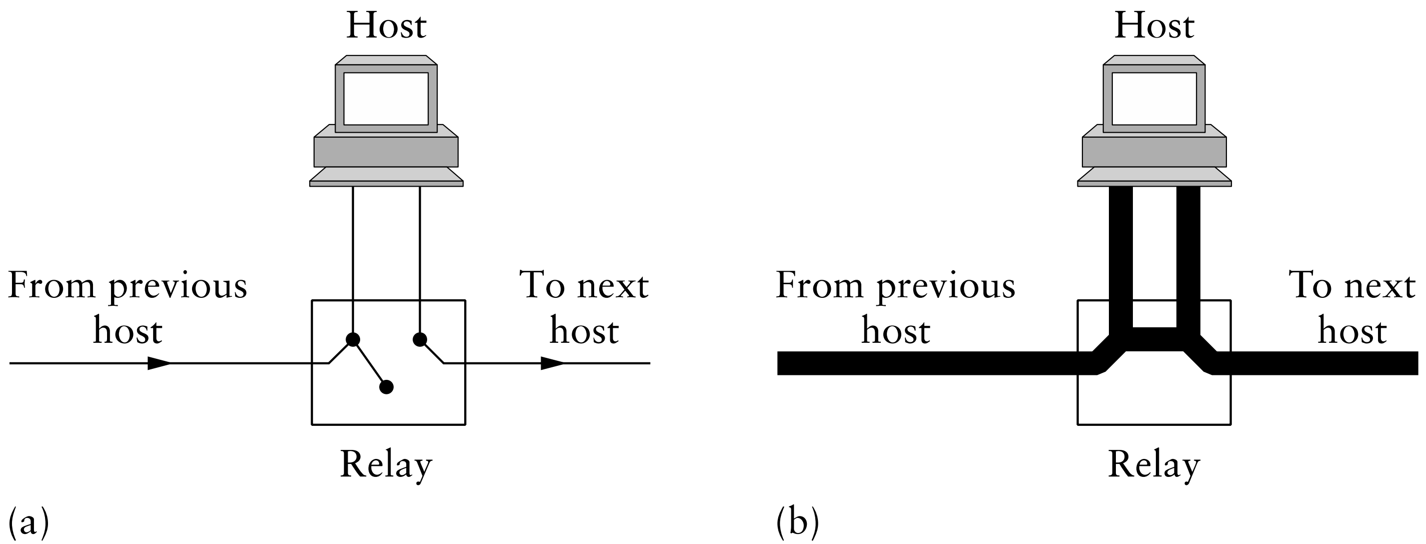
MAC: Token Based

Token Ring/Token Bus



- Idea: frames flow around ring
- Capture special “token” bit pattern to trar
- Variation used today in Metropolitan Area Networks, with fiber, Control Systems

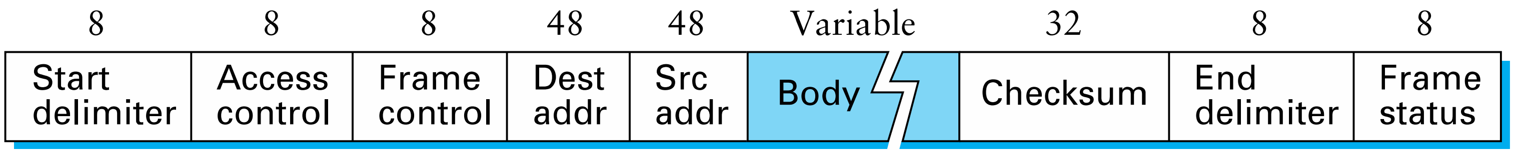
Interface Cards



- **Problem: if host dies, can break the network**
- **Hardware typically has relays**

Token Ring Frames

- Frame format (Differential Manchester)



- Sender grabs token, sends message(s)
- Recipient checks address
- Sender removes frame from ring after lap
- Maximum holding time: avoid capture
- Monitor node reestablishes lost token