# CSCI-1680
# Link Layer

Rodrigo Fonseca

Based partly on lecture notes by David Mazières, Phil Levis, John Jannotti

# Administrivia

- **Where are the policy forms?**
- **Snowcast due on Friday**
- **Homework I out on Thursday**
- **GitHub**
  - brown-csci1680 *organization*
  - Private repositories for each group

# Today

- **Previously…**
  - Physical Layer
    - Encoding
    - Modulation
  - Link Layer
    - Framing
- **Link Layer**
  - Error Detection
  - Reliability
  - Media Access
  - Ethernet
  - Token Ring

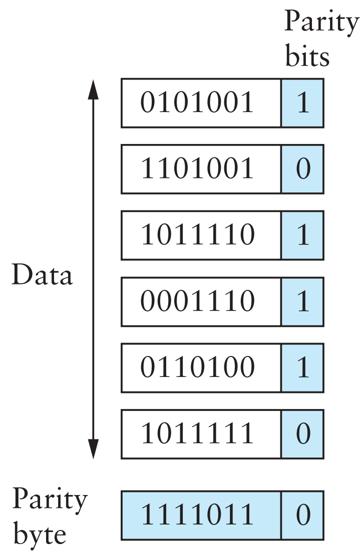
# Error Detection

- **Idea: add redundant information to catch errors in packet**
- **Used in multiple layers**
- **Three examples:**
  - Parity
  - Internet Checksum
  - CRC

# Simplest Schemes

- **Repeat frame**
  - High overhead
  - Can't correct error
- **Parity**
  - Can detect odd number of bit errors
  - No correction

# 2-D Parity




- **Add 1 parity bit for each 7 bits**
- **Add 1 parity bit for each bit position across the frame)**
  – Can correct single-bit errors
  – Can detect 2- and 3-bit errors, most 4-bit errors

# IP Checksum

- **Fixed-length code**
  - n-bit code should capture all but $2^{-n}$ fraction of errors
  - But want to make sure that includes all common errors
- **Example: IP Checksum**

```
u_short
cksum (u_short *buf, int count)
{
  u_long sum = 0;
  while (count--)
    if ((sum += *buf) & 0xffff) /* carry */
      sum = (sum & 0xffff) + 1;
  return ~(sum & 0xffff);
}
```

# How good is it?

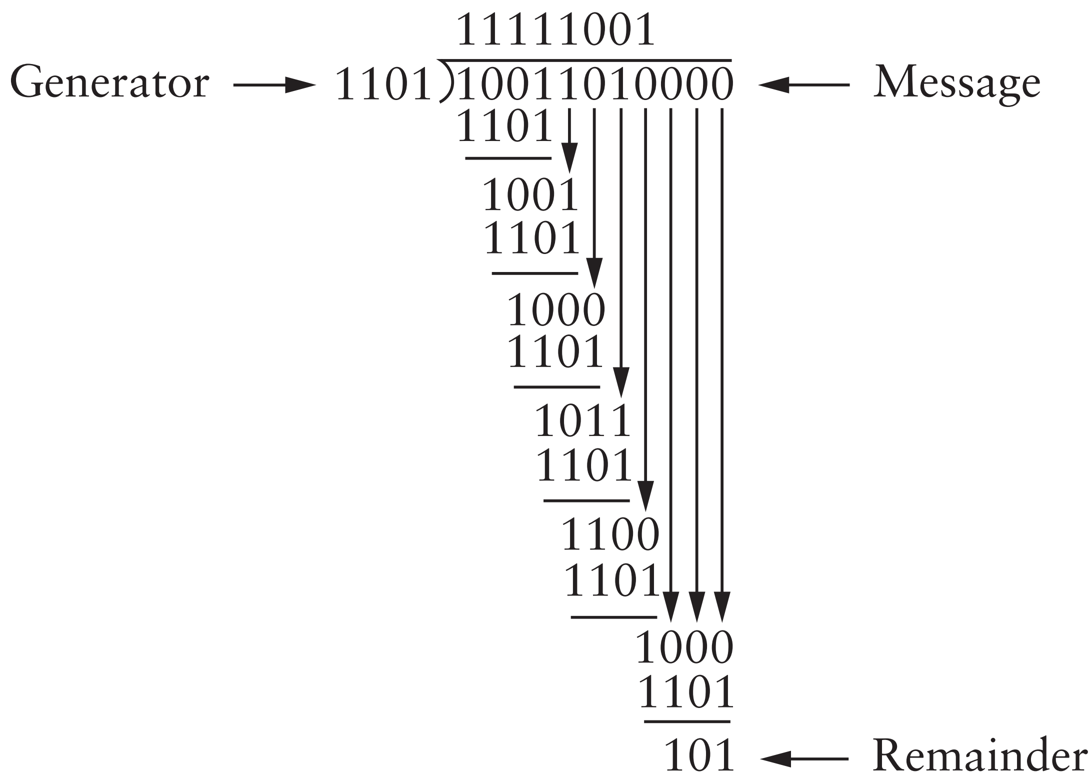
- **16 bits not very long: misses 1/64K errors**
- **Checksum does catch any 1-bit error**
- **But not any 2-bit error**
  - E.g., increment word ending in 0, decrement one ending in 1
- **Checksum also optional in UDP**
  - All 0s means no checksums calculated
  - If checksum word gets wiped to 0 as part of error, bad news

# CRC – Error Detection with Polynomials

- **Consider message to be a polynomial in $Z_2[x]$**
  - Each bit is one coefficient
  - E.g., message 10101001 -> $m(x) = x^7 + x^5 + x^3 + 1$
- **Can reduce one polynomial modulo another**
  - Let $n(x) = m(x)x^3$. Let $C(x) = x^3 + x^2 + 1$
  - Find $q(x)$ and $r(x)$ s.t. $n(x) = q(x)C(x) + r(x)$ and degree of $r(x)$ < degree of $C(x)$
  - Analogous to taking 11 mod 5 = 1

# Polynomial Division Example

- **Just long division, but addition/subtraction is XOR**



# CRC

- **Select a divisor polynomial C(x), degree k**
  - C(x) should be *irreducible* – not expressible as a product of two lower-degree polynomials in $Z_2[x]$
- **Add k bits to message**
  - Let $n(x) = m(x)x^k$ (add k 0's to m)
  - Compute $r(x) = n(x) \bmod C(x)$
  - Compute $n(x) = n(x) - r(x)$ (will be divisible by C(x)) (subtraction is XOR, just set k lowest bits to r(x)!)
- **Checking CRC is easy**
  - Reduce message by C(x), make sure remainder is 0

# Why is this good?

- **Suppose you send m(x), recipient gets m'(x)**
  - E(x) = m'(x) – m(x) (all the incorrect bits)
  - If CRC passes, C(x) divides m'(x)
  - Therefore, C(x) must divide E(x)
- **Choose C(x) that doesn't divide any common errors!**
  - All single-bit errors caught if $x^k$, $x^0$ coefficients in C(x) are 1
  - All 2-bit errors caught if at least 3 terms in C(x)
  - Any odd number of errors if last two terms (x + 1)
  - Any error burst less than length k caught

# Common CRC Polynomials

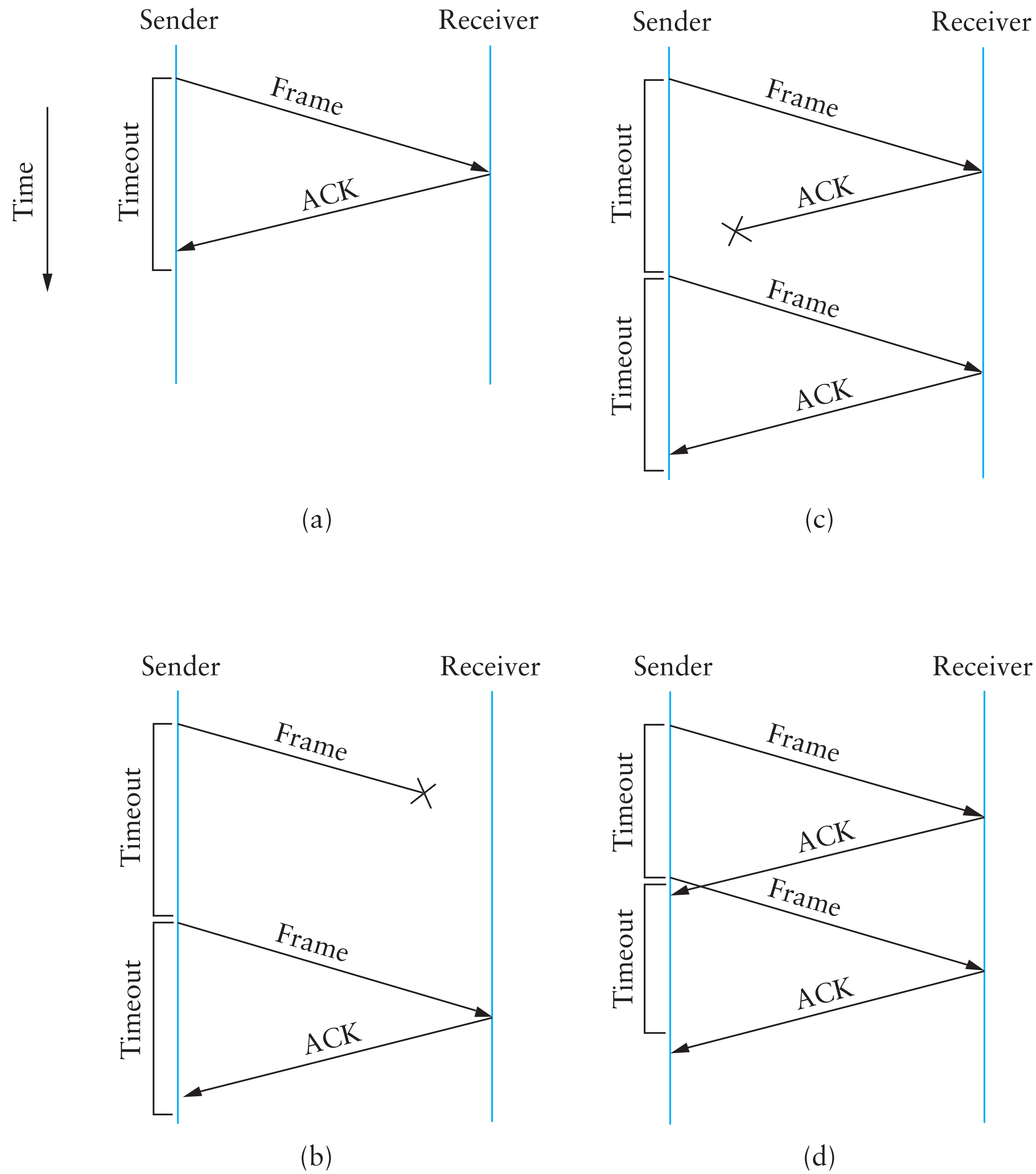
- CRC-8: $x^8 + x^2 + x^1 + 1$
- CRC-16: $x^{16} + x^{15} + x^2 + x^1$
- CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$
- CRC easily computable in hardware

# Reliable Delivery

- **Error detection can discard bad packets**
- **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
- **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

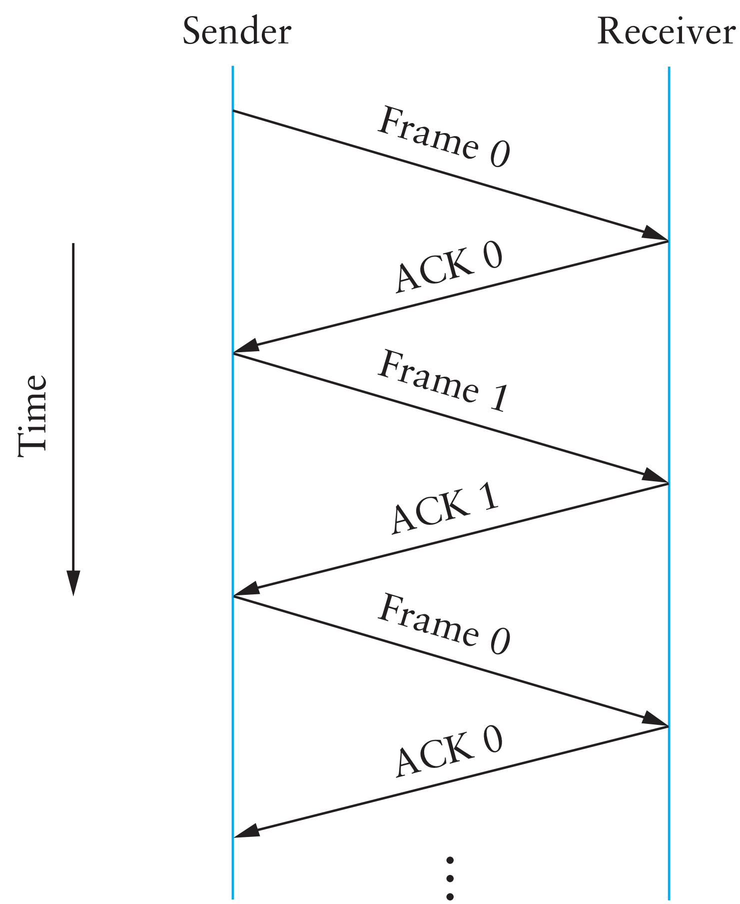

Sender
Receiver
Time
Timeout
Frame
ACK
(a)
(b)
(c)
(d)

# Stop and Wait Problems

- Duplicate data
- Duplicate acks
- Can’t fill pipe (remember bandwitdh-delay product)
- Difficult to set the timeout value

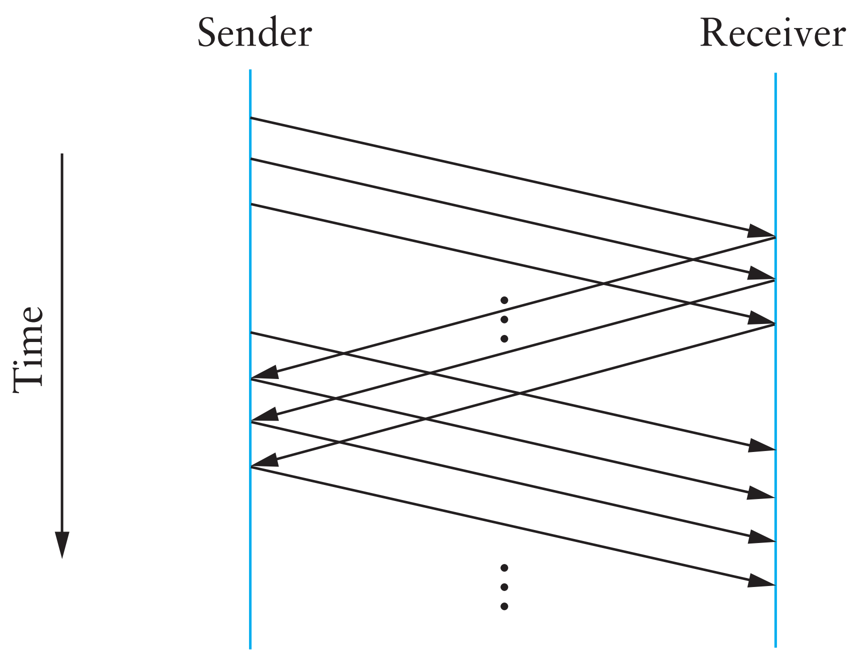
# At Most Once Semantics

- **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?


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

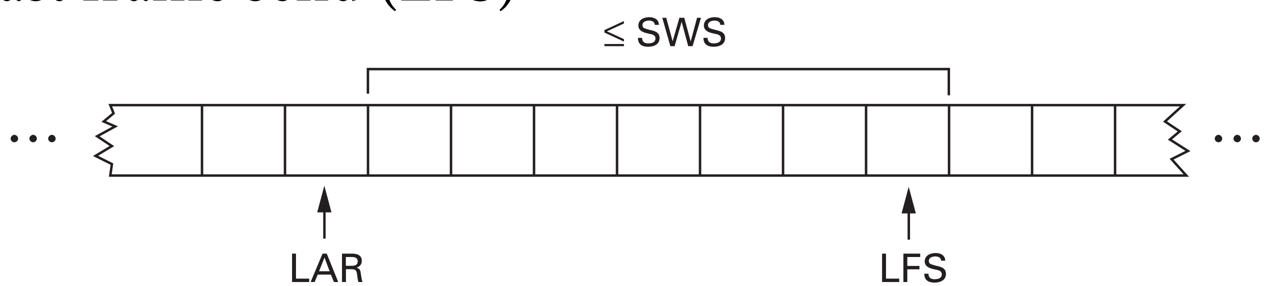
# Sliding Window Protocol

- **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*



# Sliding Window Sender

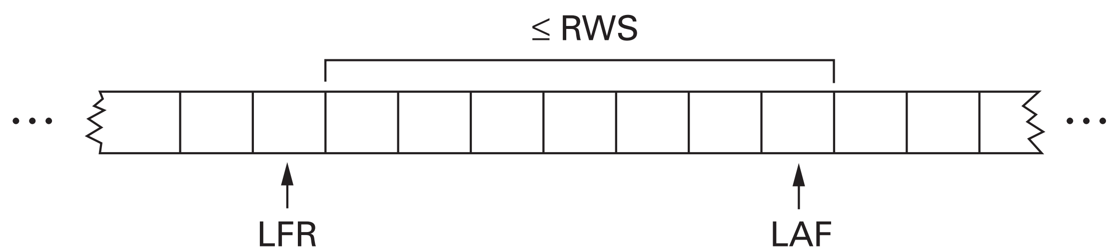
- **Assign sequence number (SeqNum) to each frame**
- **Maintain three state variables**
  - send window size (SWS)
  - last acknowledgment received (LAR)
  - last frame send (LFS)




- **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)



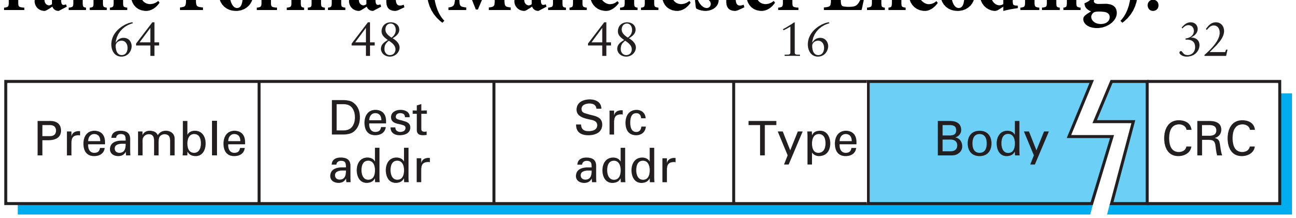

- **Maintain invariant: LAF – LFR ≤ RWS**
- **Frame SeqNum arrives:**
  - if LFR < SeqNum ≤ LAF, accept
  - if SeqNum ≤ LFR or SeqNum > LAF, discard
- **Send *cumulative* ACKs**

# Tuning SW

- **How big should SWS be?**
  - "Fill the pipe"
- **How big should RWS be?**
  - $1 \leq RWS \leq SWS$
- **How many distinct sequence numbers needed?**
  - If RWS = 1, need at least SWS+1
  - If RWS = SWS, SWS < (#seqs + 1)/2

# 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**

# Media Access Control

- **Control access to shared physical medium**
  - E.g., who can talk when?
  - If everyone talks at once, no one hears anything
  - 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

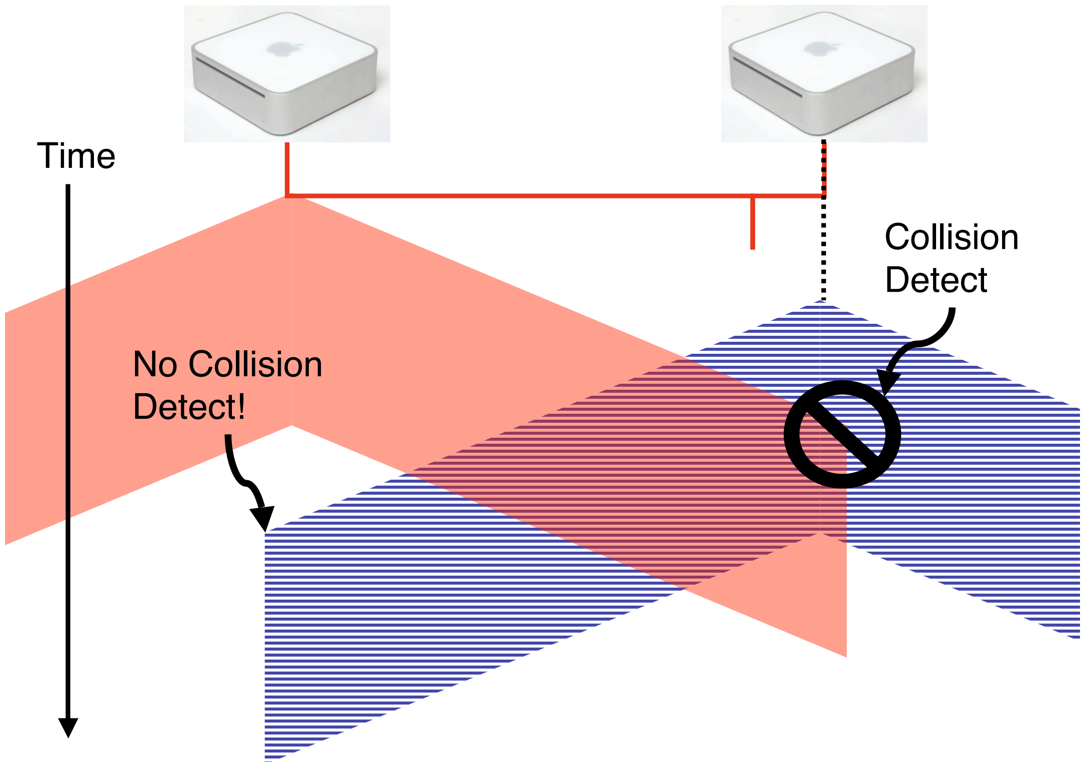
# Ethernet MAC

- **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 between back to back frames
  - If line is busy: wait until idle and transmit immediately

# Handling Collisions

- **Collision detection (10Base2 Ethernet)**
  - Uses Manchester encoding
  - 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)
  - Ensure transmission time ≥ 2x propagation delay (why?)

# Collision Detection




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

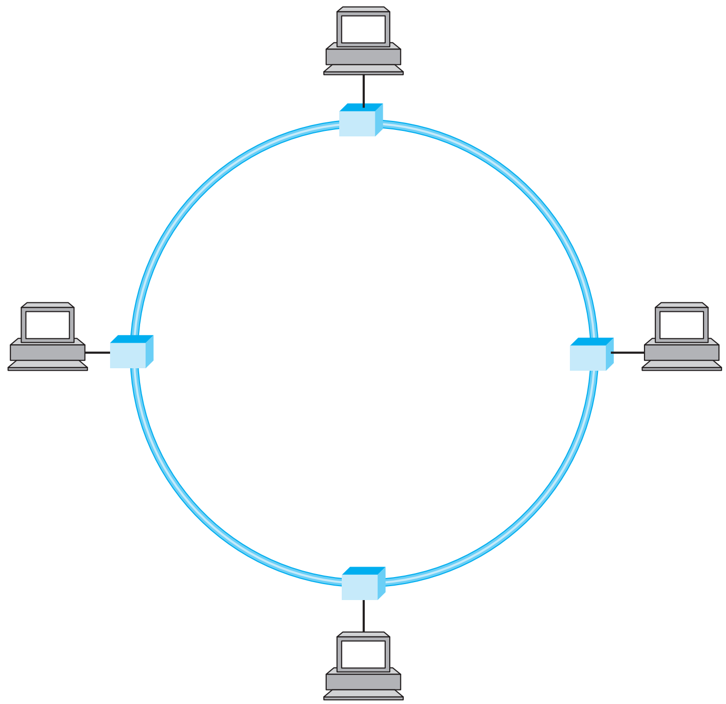
# When to transmit again?

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

# Capture Effect

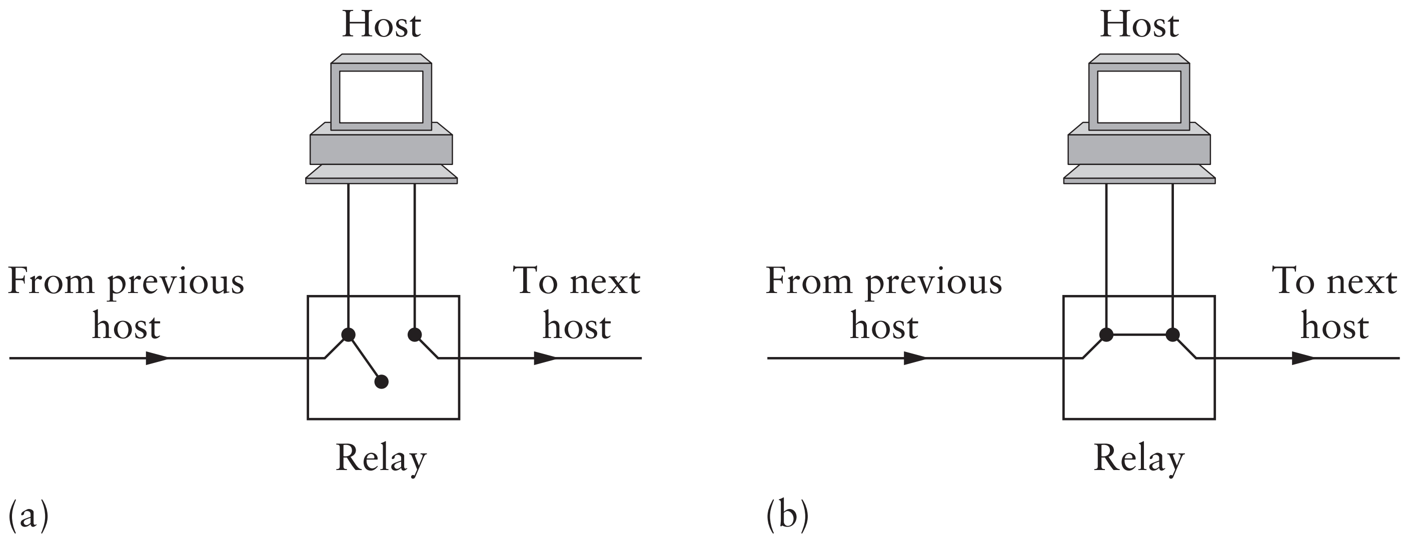
- **Exponential backoff leads to self-adaptive use of channel**
- **A and B are trying to transmit, and collide**
- **Both will back off either 0 or 51.2µs**
- **Say A wins.**
- **Next time, collide again.**
  - A will wait between 0 or 1 slots
  - B will wait between 0, 1, 2, or 3 slots
- …

# Token Ring



- Idea: frames flow around ring
- Capture special "token" bit pattern to transmit
- Variation used today in Metropolitan Area Networks, with fiber

# Interface Cards




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

# Token Ring Frames

- Frame format (Differential Manchester)

| Start delimiter | Access control | Frame control | Dest addr | Src addr | Body | Checksum | End delimiter | Frame status |
|---|---|---|---|---|---|---|---|---|
| 8 | 8 | 8 | 48 | 48 | Variable | 32 | 8 | 8 |

- 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

# Coming Up

- Link Layer Switching