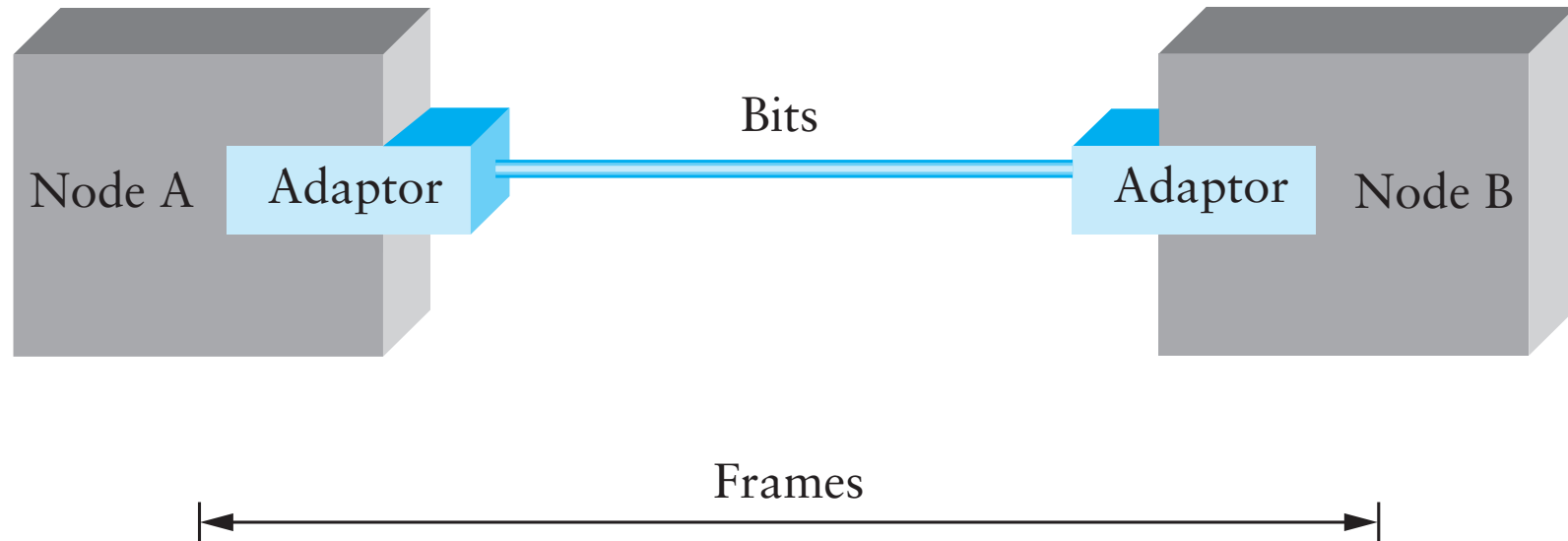


Link-Layer

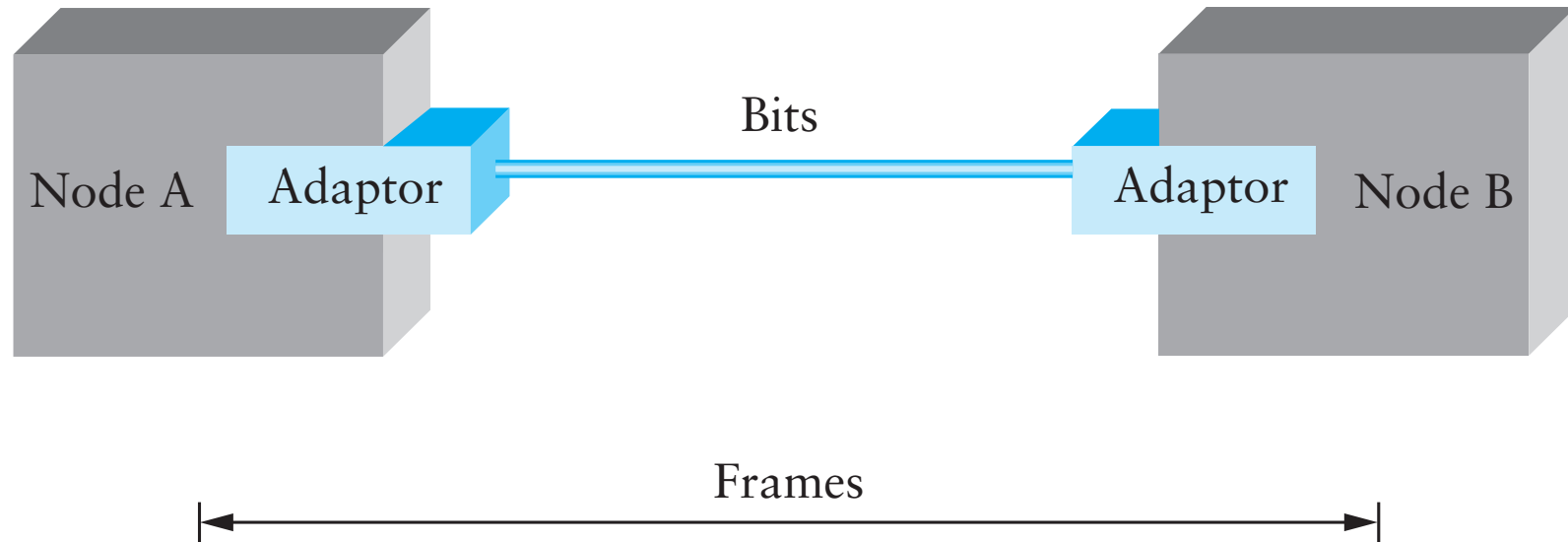
- **Framing – How to delineate frames?**
- **Error detection**
- **Flow control**
- **Reliability – mostly for point-to-point**
- **Media Access Control (MAC) – for shared media**

Framing



- Break sequence of bits into a frame.
- Typically implemented by network adaptor.
- Given pure bit strings, how can boundaries be represented?

Representing boundaries

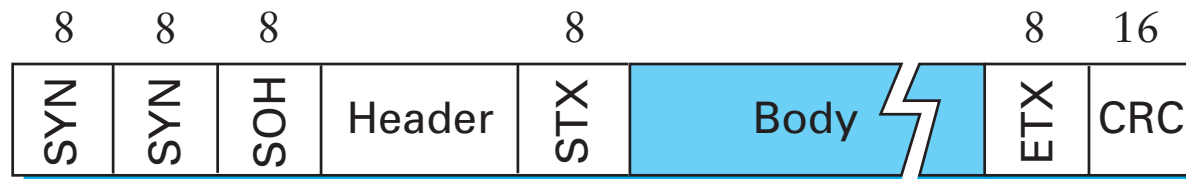


- **Sentinels**
- **Length counts**
- **Clock-based**

Sentinel-based framing

- **Byte-oriented protocols (e.g., BISYNC, PPP)**

- Place special bytes (SOH, ETX) at beginning, end of message

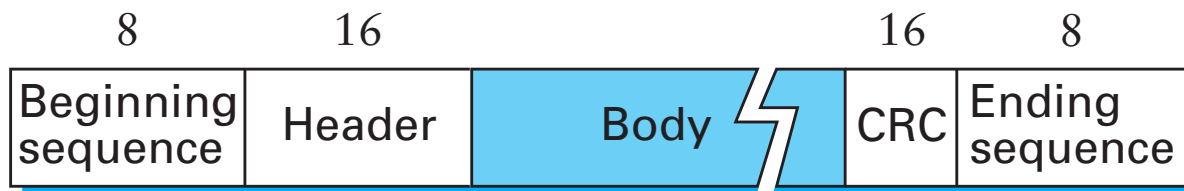


- **What if ETX appears in the body?**

- Escape ETX byte by prefixing DEL byte
- Escape DEL byte by prefixing DEL byte
- Technique known as *character stuffing*

Bit-oriented protocols

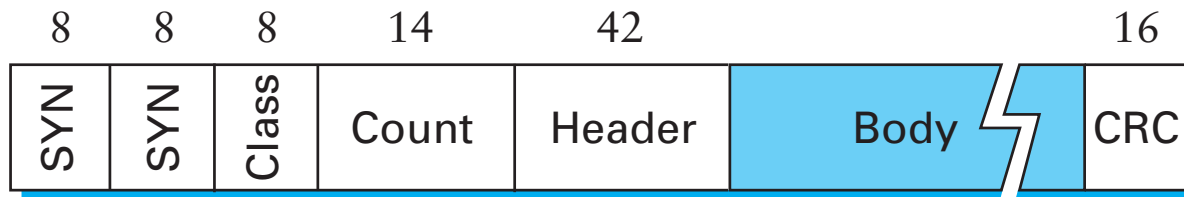
- View message as stream of bits, not bytes
- Can similarly use sentinel (e.g., HDLC)



- HDLC begin/end sequence 01111110
- Use *bit-stuffing* technique to escape 01111110
 - When transmitting message, always append 0 to five consecutive 1s
 - Receiver uses bit to distinguish from begin/end sequence, strips bit out of payload

Length Counts

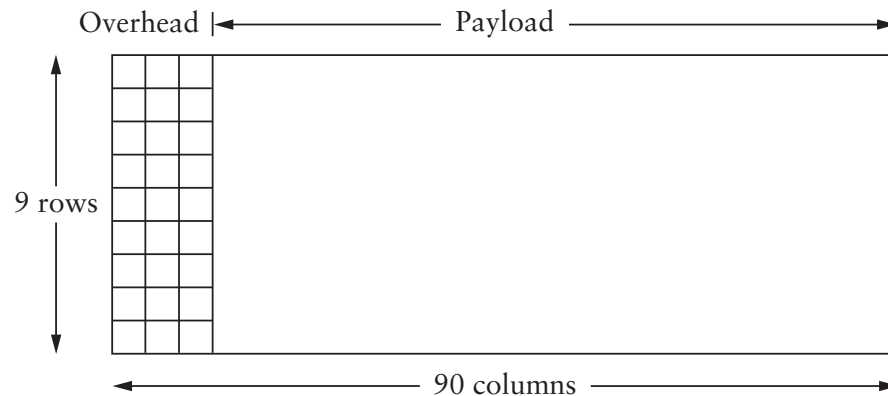
- **Drawback of “stuffing techniques”**
 - Length of frame depends on contents of payload
- **Alternative: Put length in header (e.g., DDCMP)**



- **Danger: Framing errors**
 - What if high bit of count gets corrupted
 - Adds 8K to length of frame, may lose many frames
 - CRC checksum helps detect error

Clock-based framing

- *E.g.*, **SONET: Synchronous Optical Network**
 - Each frame is $125\mu\text{s}$ long
 - Look for header every $125\mu\text{s}$
 - Encode with NRZ, but XOR payload with 127-bit string to help ensure lots of transitions.



Error detection

- **Basic idea: Use a checksum**
 - Compute small checksum value, like a hash of the packet
 - If packet changes, checksum is likely to be wrong
- **Good checksum algorithms**
 - Want several properties, *e.g.*, detect any single bit error
 - Details in a later lecture
- **Next problem: If discarding bad packets, how to ensure reliable delivery?**

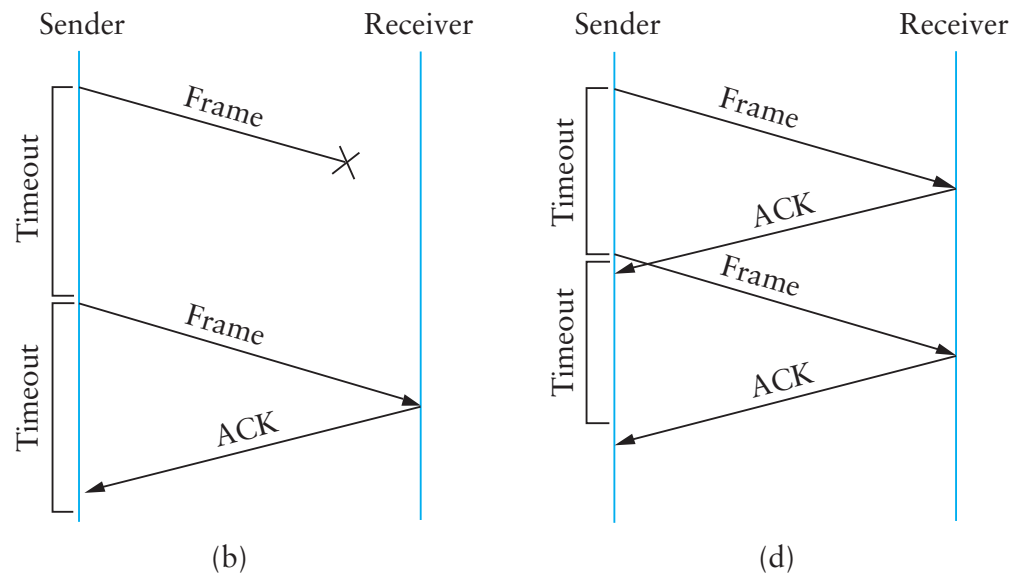
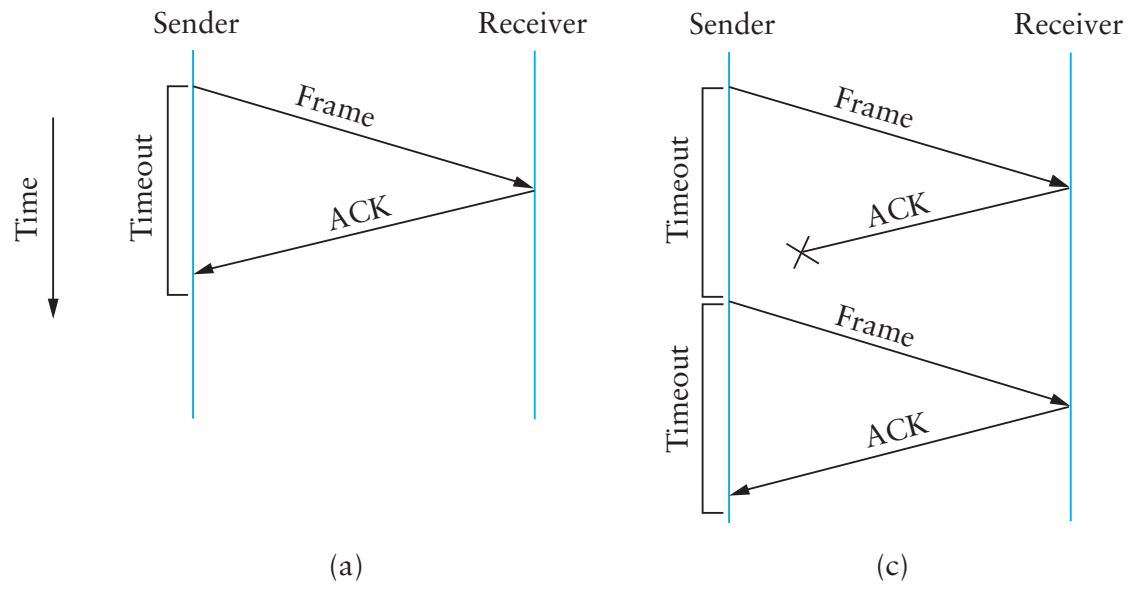
Acknowledgements and Timeouts

- **Stop and wait approach**

- Send packet, wait
- Receive packet, send ACK
- Receive ACK, send next packet
- Don't receive ACK, timeout and retransmit

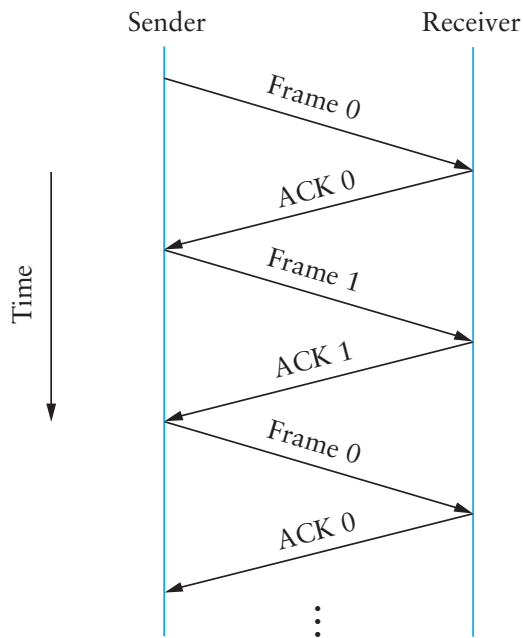
- **Problems**

- Might duplicate packet
- Can't keep pipe full (remember bandwidth-delay product)
- Difficult to set length of timeout



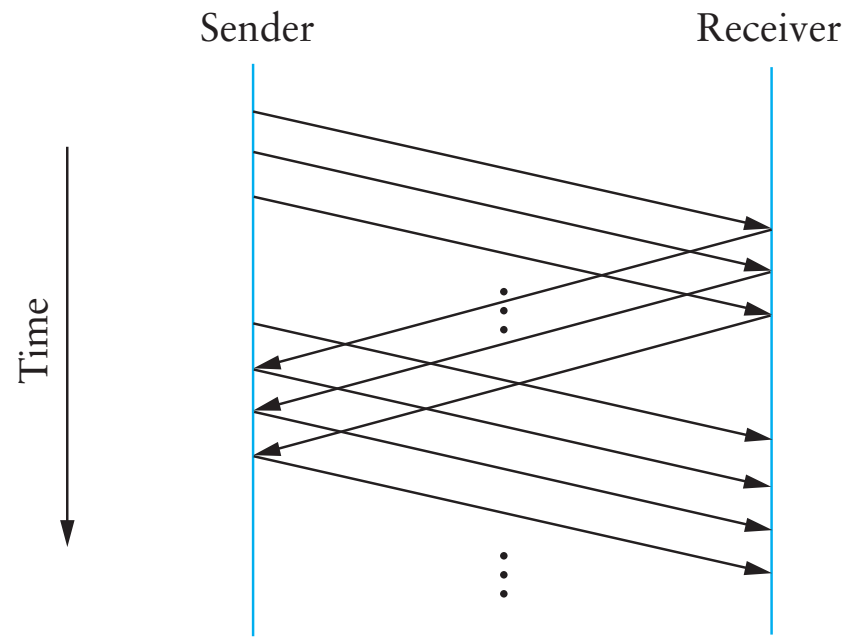
Duplicates

- Can avoid duplicates with 1-bit flag
 - Placed in both Frame and ACK.
 - Receiver knows if duplicate of last frame.



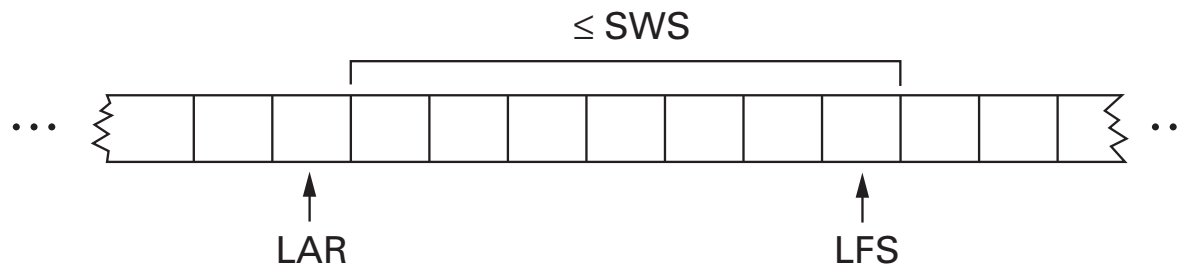
Sliding window protocol

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



SW sender

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

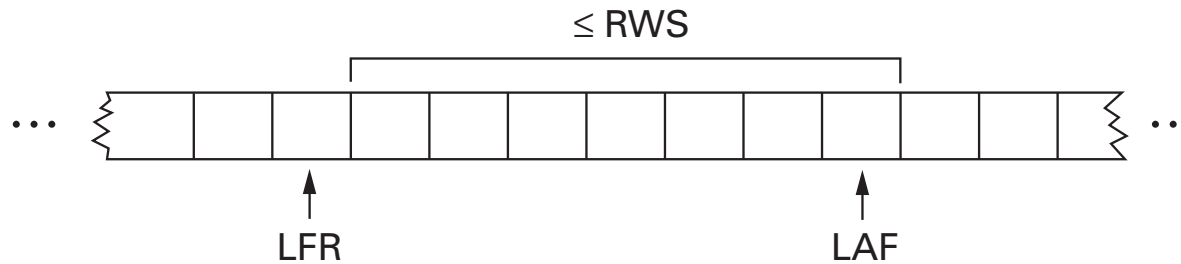


- Maintain invariant: $LFS - LAR \leq SWS$
- Advance LAR when ACK arrives
- Buffer up to SWS frames

SW receiver

- **Maintain three state variables**

- receive window size (RWS)
- largest acceptable frame (LAF)
- last frame received (LFR)



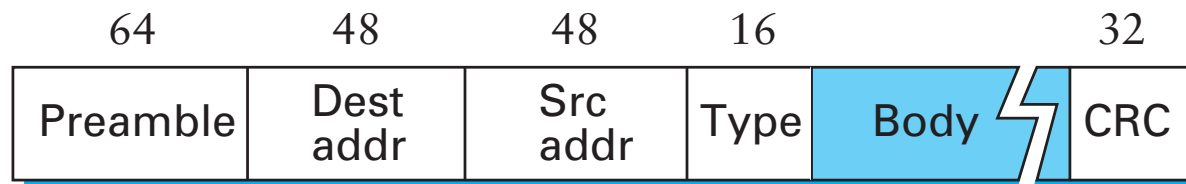
- **Maintain invariant: $LAF - LFR \leq RWS$**
- **Frame SeqNum arrives:**
 - if $LFR < SeqNum \leq LFA$ accept
 - if $SeqNum \leq LFR$ or $SeqNum > LFA$ discarded
- **Send cumulative ACKs**

Sequence number space

- **How big should RWS be?**
 - At least 1, no bigger than SWS
- **How many distinct sequence numbers needed?**
 - If $RWS=1$, need at least $SWS+1$
 - If $RWS=SWS$, need at least $2SWS+1$
 - (Imagine resending frames because of lost ACKs.)

Ethernet (Framing)

- One of the most popular LAN network types
- CSMA/CD
 - carrier sense
 - multiple access
 - collision detection
- Frame format (Manchester encoding)



Ethernet (MAC)

- **Addresses**

- unique, 48-bit unicast address assigned to each adapter
- example: 00:07:e9:52:c3:93
- broadcast: all 1s
- multicast: first bit is 1

- **Bandwidth: 10Mbps, 100Mbps, 1Gbps**

- **Length: 2500m (500m segments with 4 repeaters)**

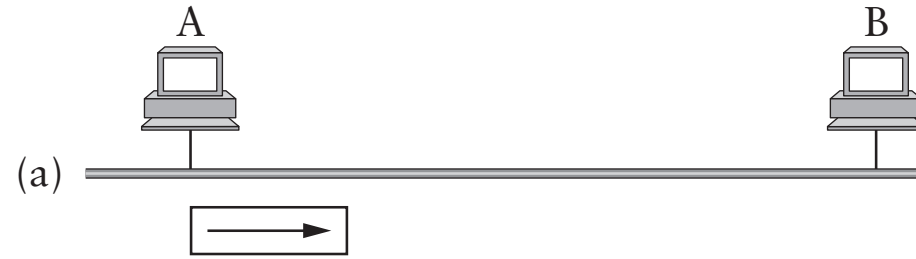
- **Problem: Distributed algorithm to provide fair access**

Transmit algorithm

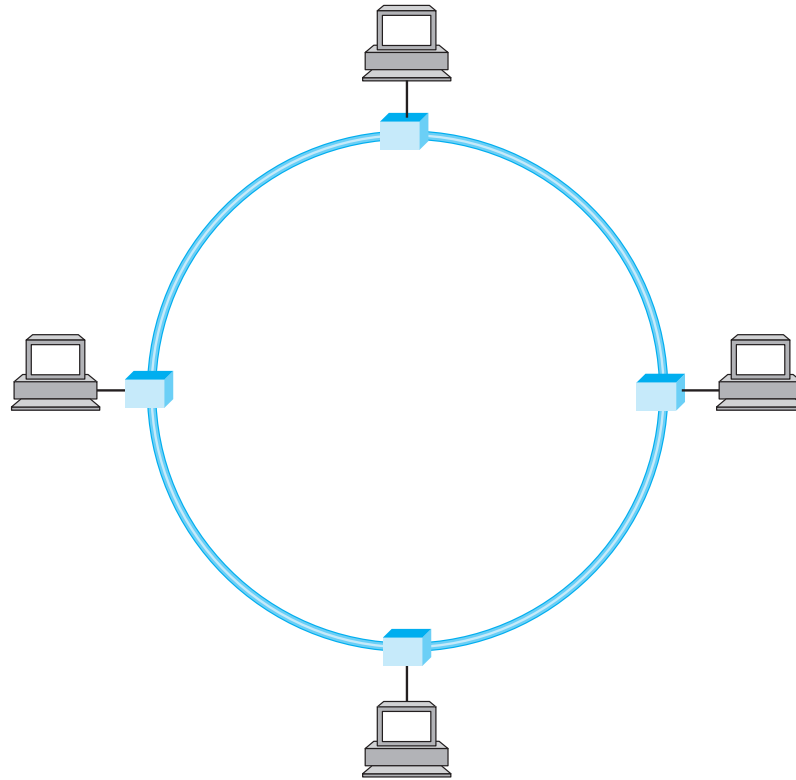
- **If line is idle**
 - send immediately
 - upper bound message size of 1500 bytes
 - must wait $9.6\mu\text{s}$ between back-to-back frames
- **If line is busy**
 - wait until idle and transmit immediately
 - called 1-persistent (special case of p-persistent)

Handling collisions

- **If collision**
 - jam for 32 bits, then stop transmitting frame
 - minimum frame is 64 bytes (header + 46 bytes of data)
 - delay and try again
- **n th time: $k \times 51.2\mu\text{s}$, for $k \leftarrow_R \{0 \dots 2^{\min(n,10)} - 1\}$**
 - 1st time: 0 or $51.2\mu\text{s}$
 - 2rd time: 0, 51.2, 102.4, or $153.6\mu\text{s}$
- **give up after several tries (usually 16)**
- **Min packet size determines max net length...**

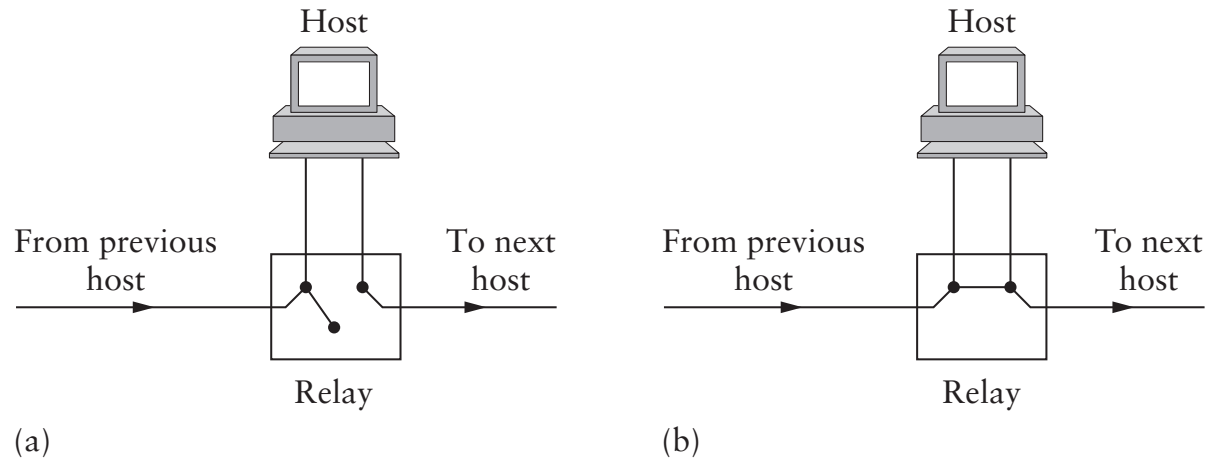


Token ring



- Idea: Frames flow around ring
- Capture special “token” bit pattern to transmit

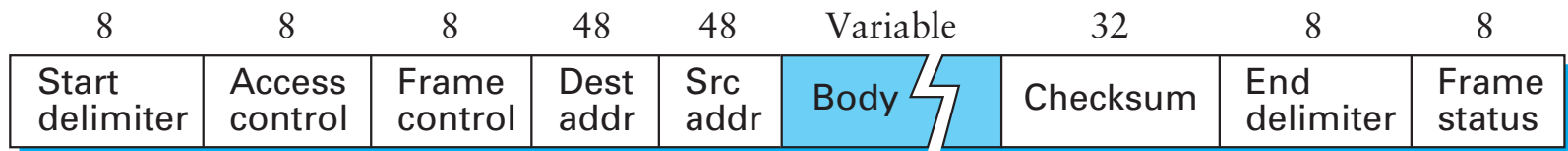
Interface cards



- **Danger if host dies, can wedge net**
- **Hardware typically has relays**
 - In interface cards, or multi-station access units (MSAUs)

Token ring frames

- **Frame format (Manchester encoding)**



- **First two bits of token differ from preamble by one bit**

- Can just flip bit as you grab the token, so don't need to buffer more than one byte at host

- **Token maintenance**

- Various complications required to recover from lost token, etc.

Coming up

- **Today: Add Date (Banner IDs for grads)**
- **Today, until 2pm: Free Grand Slam [Denny's]**
- **Thu: Switching**
- **Fri, 5pm: Snowcast due**
- **Mon: Apple info session**
- **Tue: Homework 1 out**
- **Thu: GTECH info session**