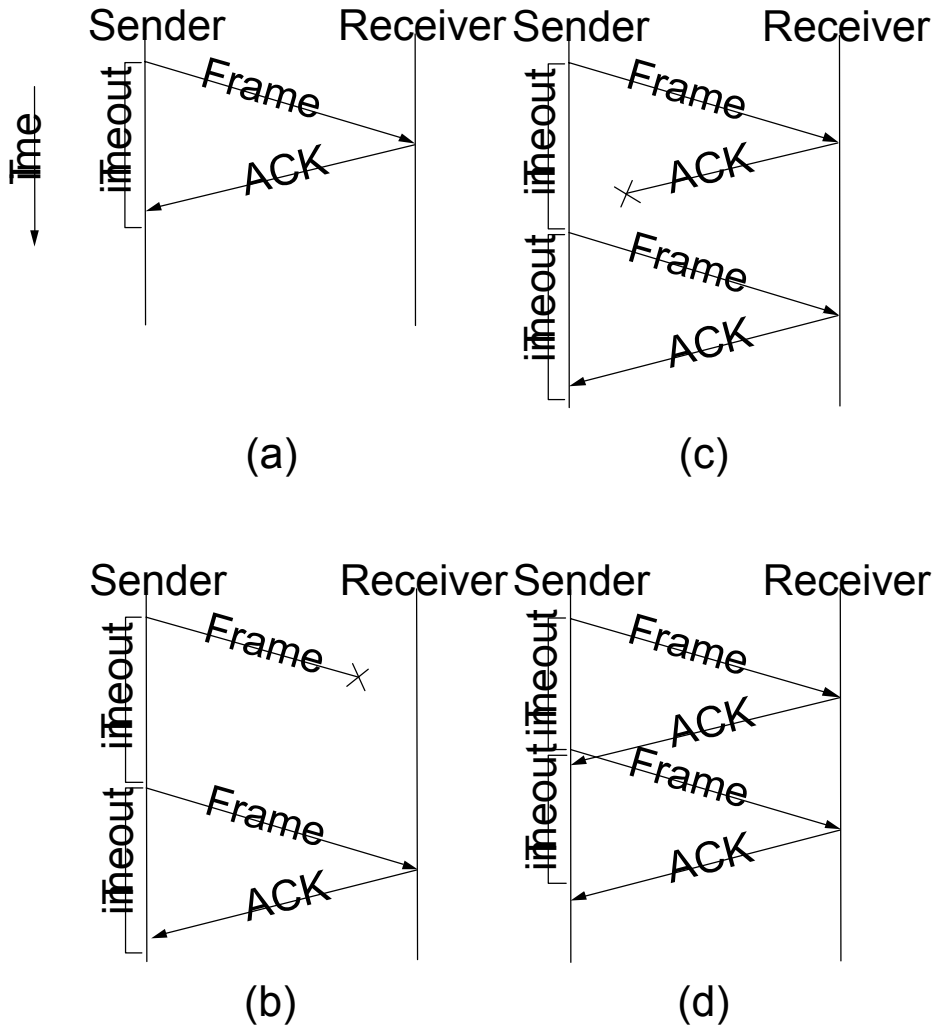
Reliable Transmission

Acknowledgements and Timeouts

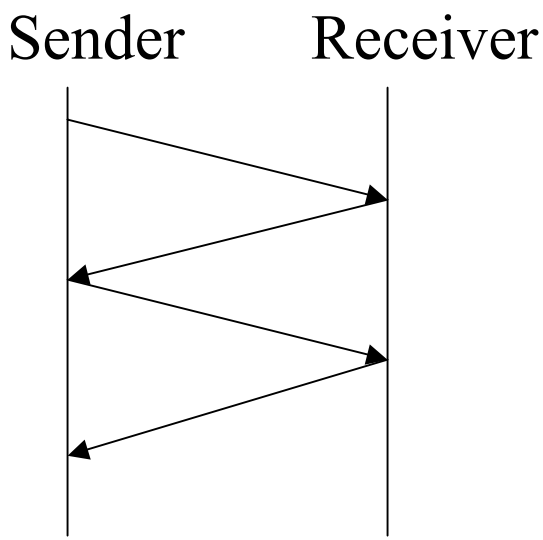
- ACK – Control frame sent back to peers saying that it successfully received an earlier frame
- If no ack without reasonable time, timeout and retransmit packet
- Automatic Repeat Request - ARQ

Acknowledgements & Timeouts



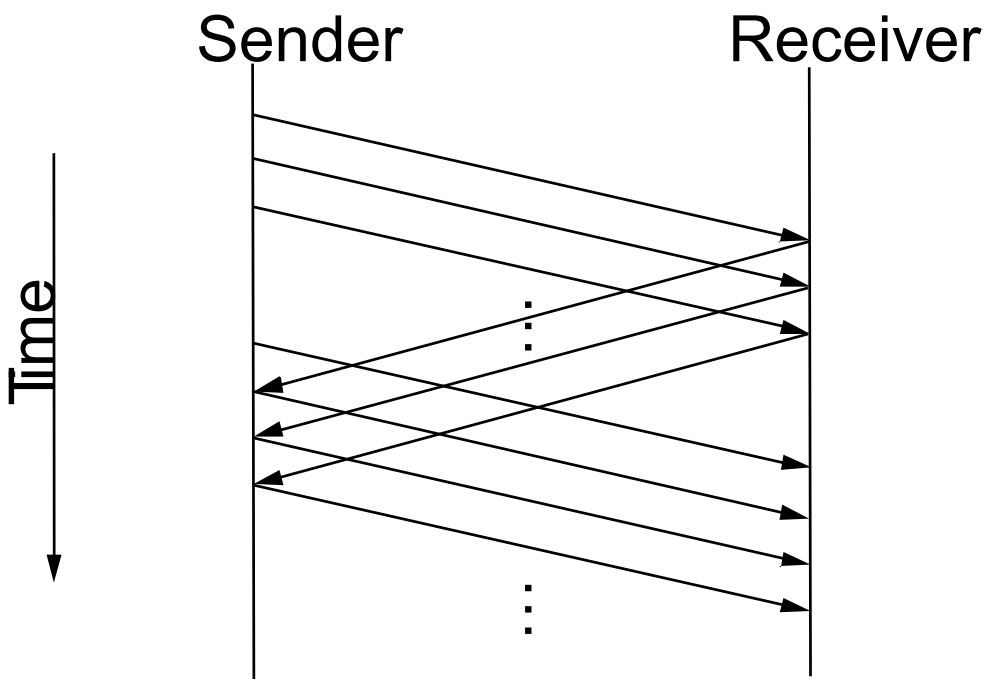
Stop-and-Wait

- Problem: keeping the pipe full
  - (bandwidth delay product)
- Example
  - 1.5Mbps link x 45ms RTT = 67.5Kb (8KB)
  - 1 KB frames imples 1/8th link utilization



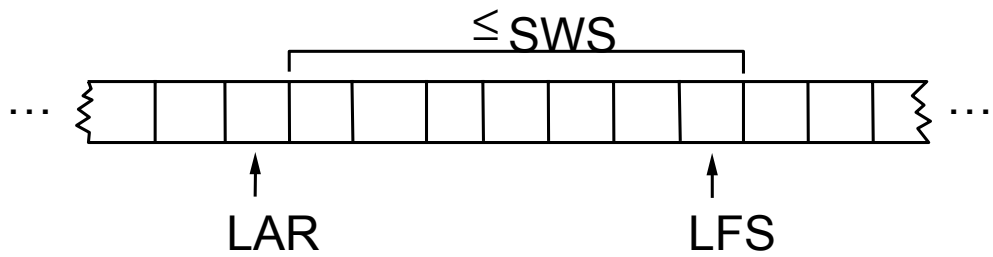
Sliding Window

- Allow multiple outstanding (un-ACKed) frames
- Upper bound on un-ACKed frames, called window



SW: Sender

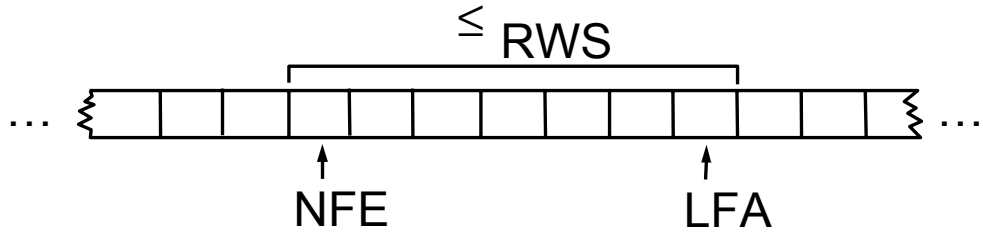
- Assign sequence number to each frame (SeqNum)
- Maintain three state variables:
  - send window size (SWS)
  - last acknowledgment received (LAR)
  - last frame sent (LFS)
- Maintain invariant: LFS - LAR <= SWS



- Advance LAR when ACK arrives
- Buffer up to SWS frames

SW: Receiver

- Maintain three state variables
  - receive window size (RWS)
  - largest frame acceptable (LFA)
  - last frame received (NFE)
- Maintain invariant: LFA - LFR <= RWS



- Frame SeqNum arrives:
  - if LFR < SeqNum < = LFA → accept
  - if SeqNum < = LFR or SeqNum > LFA → discarded
- Send cumulative ACKs

Sequence Number Space

- SeqNum field is finite; sequence numbers wrap around
- Sequence number space must be larger then number of outstanding frames
- SWS <= MaxSeqNum-1 is not sufficient
  - suppose 3-bit SeqNum field (0..7)
  - SWS=RWS=7
  - sender transmit frames 0..6
  - arrive successfully, but ACKs lost
  - sender retransmits 0..6
  - receiver expecting 7, 0..5, but receives second incarnation of 0..5
- SWS < (MaxSeqNum+1)/2 is correct rule
- Intuitively, SeqNum “slides” between two halves of sequence number space

Concurrent Logical Channels

- Multiplex 8 logical channels over a single link
- Run stop-and-wait on each logical channel
- Maintain three state bits per channel
  - channel busy
  - current sequence number out
  - next sequence number in
- Header: 3-bit channel num, 1-bit sequence num
  - 4-bits total
  - same as sliding window protocol
- Separates reliability from order

Shared Access Networks

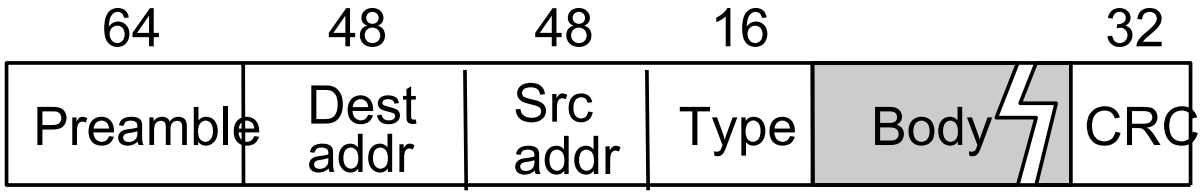
Bus (Ethernet)

Token ring (FDDI)

Wireless (802.11)

Ethernet Overview

- History
  - developed by Xerox PARC in mid-1970s
  - roots in Aloha packet-radio network
  - standardized by Xerox, DEC, and Intel in 1978
  - similar to IEEE 802.3 standard
- CSMA/CD
  - carrier sense
  - multiple access
  - collision detection
- Frame Format



Ethernet (cont)

- Addresses
  - unique, 48-bit unicast address assigned to each adapter
  - example: 8:0:e4:b1:2
  - broadcast: all 1s
  - multicast: first bit is 1
- Bandwidth: 10Mbps, 100Mbps, 1Gbps
- Length: 2500m (500m segments with 4 repeaters)
- Problem: Distributed algorithm that provides fair access

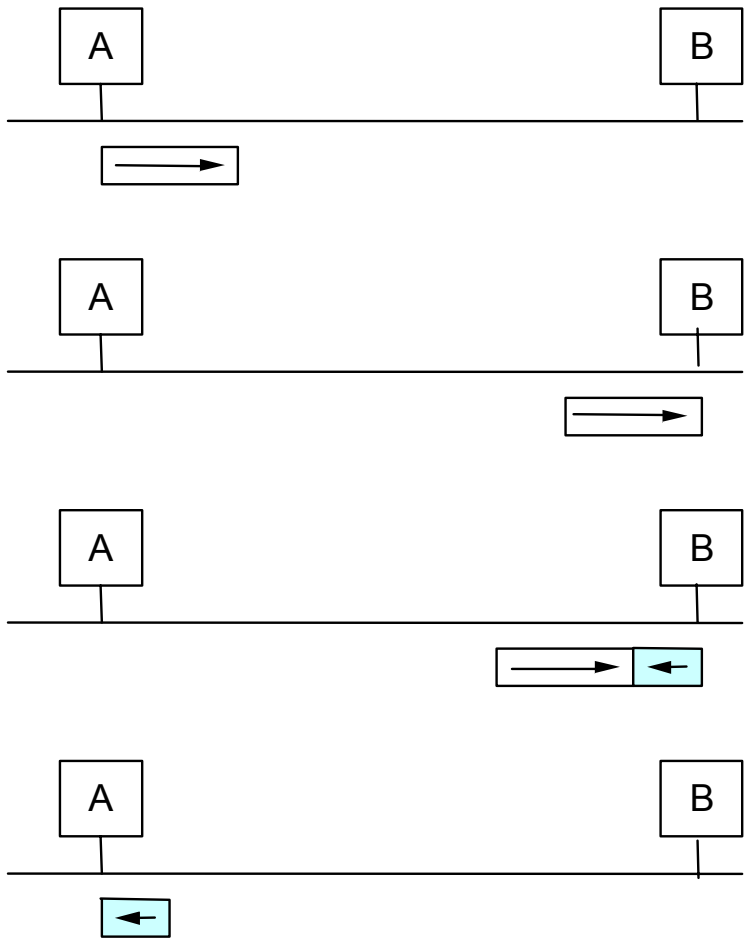
Transmit Algorithm

- If line is idle…
  - send immediately
  - upper bound message size of 1500 bytes
  - must wait 9.6us between back-to-back frames
- If line is busy…
  - wait until idle and transmit immediately
  - called 1-persistent (special case of p-persistent)

Algorithm (cont)

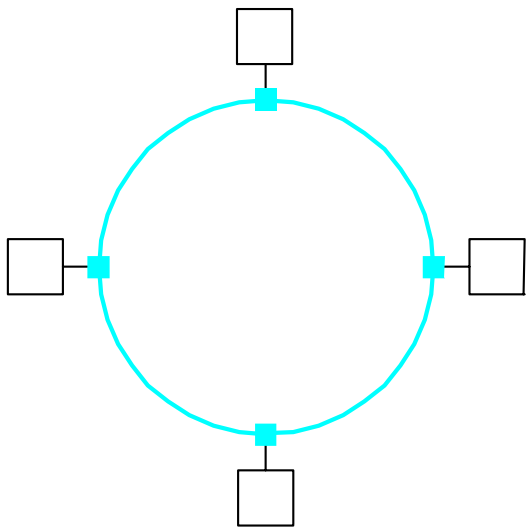
- If collision…
  - jam for 32 bits, then stop transmitting frame
  - minimum frame is 64 bytes (header + 46 bytes of data)
  - delay and try again
    - 1st time: 0 or 51.2us
    - 2nd time: 0, 51.2, or 102.4us
    - 3rd time51.2, 102.4, or 153.6us
    - nth time: k x 51.2us, for randomly selected k=0..2n - 1
    - give up after several tries (usually 16)
    - exponential backoff

Collisions



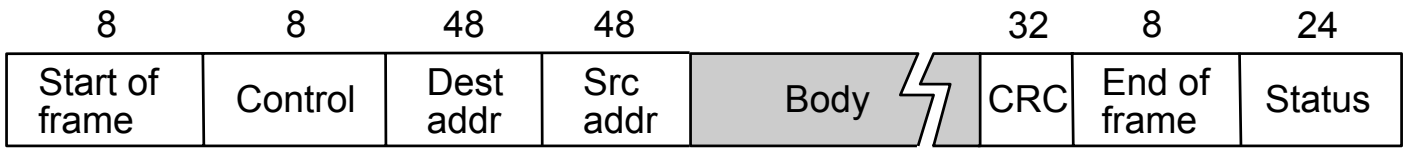
Token Ring Overview

- Examples
  - 16Mbps IEEE 802.5 (based on earlier IBM ring)
  - 100Mbps Fiber Distributed Data Interface (FDDI)



Token Ring (cont)

- Idea
  - Frames flow in one direction: upstream to downstream
  - special bit pattern (token) rotates around ring
  - must capture token before transmitting
  - release token after done transmitting
    - immediate release
    - delayed release
  - remove your frame when it comes back around
  - stations get round-robin service
- Frame Format



Timed Token Algorithm

- Token Holding Time (THT)
  - upper limit on how long a station can hold the token
- Token Rotation Time (TRT)
  - how long it takes the token to traverse the ring.
  - TRT <= ActiveNodes x THT + RingLatency
- Target Token Rotation Time (TTRT)
  - agreed-upon upper bound on TRT

Algorithm (cont)

- Each node measures TRT between successive tokens
  - if measured-TRT > TTRT: token is late so don't send
  - if measured-TRT < TTRT: token is early so OK to send
- Two classes of traffic
  - synchronous: can always send
  - asynchronous: can send only if token is early
- Worse case: 2xTTRT between seeing token
- Back-to-back 2xTTRT rotations not possible

Token Maintenance

- Lost Token
  - no token when initializing ring
  - bit error corrupts token pattern
  - node holding token crashes
- Generating a Token (and agreeing on TTRT)
  - execute when join ring or suspect a failure
  - send a claim frame that includes the node's TTRT bid
  - when receive claim frame, update the bid and forward
  - if your claim frame makes it all the way around the ring:
    - your bid was the lowest
    - everyone knows TTRT
    - you insert new token

Maintenance (cont)

- Monitoring for a Valid Token
  - should periodically see valid transmission (frame or token)
  - maximum gap = ring latency + max frame < = 2.5ms
  - set timer at 2.5ms and send claim frame if it fires