Announcements

- Reading
  - Today: 6.1-6.2.6
  - Thursday: 6.3-6.4

Transport Layer

- Goal: provide error free end-to-end delivery of data
  - provide in-order delivery over unreliable network layer
- Issues:
  - checking packet integrity
  - re-transmission of lost of corrupt packets
  - connection establishment and management
  - addresses
    - need to define a host plus process
    - typical abstraction is <host, port>

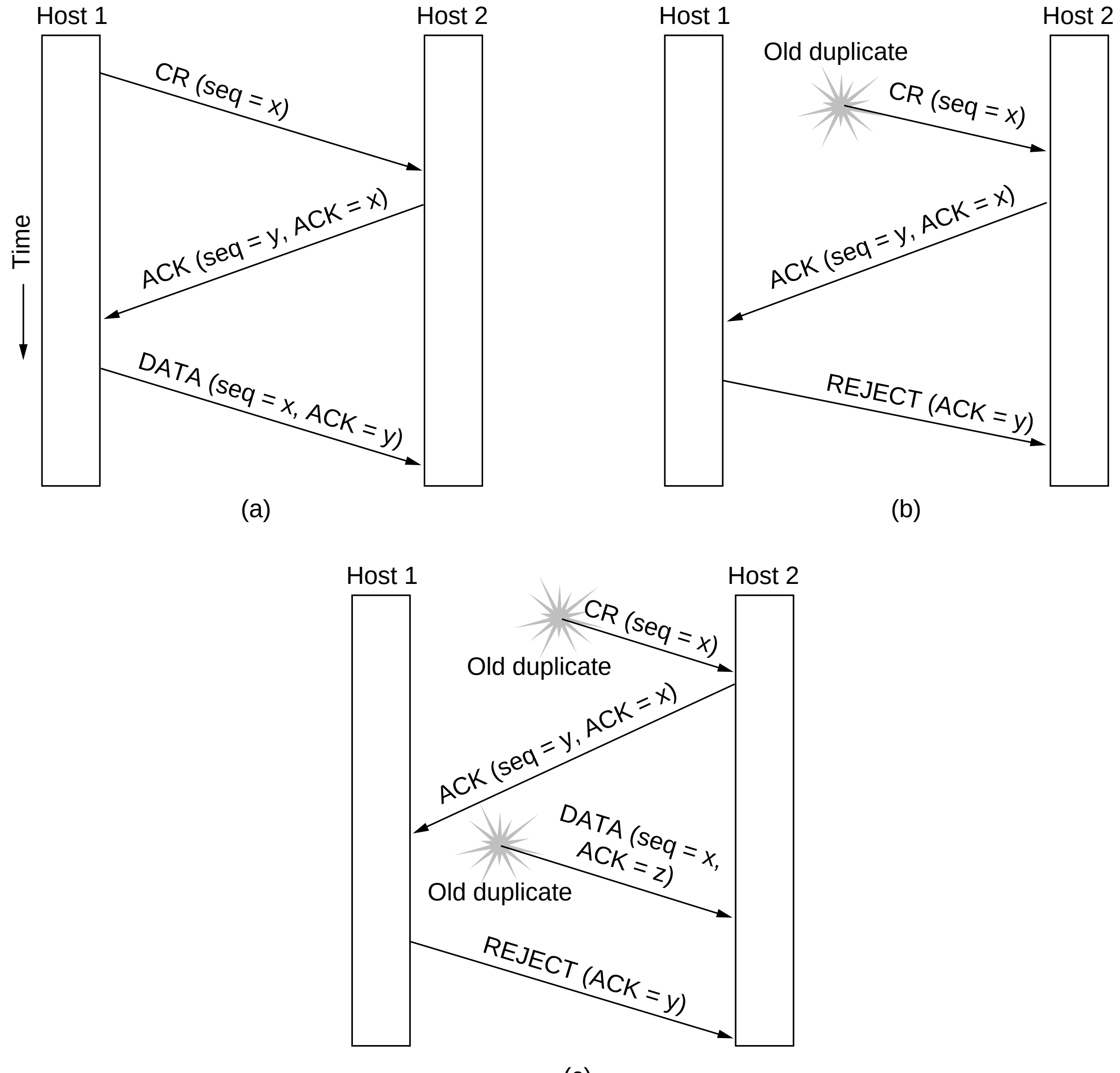
Duplicate Packets

- Issue: packets can be lost or duplicated
  - need to detect duplicates
  - need to re-send lost packets
    - but how do we know they are not just delayed?
- Solution 1
  - use a sequence number
    - each new packet uses a new sequence number
    - can detect arrival of stale packets
  - problem: when node crashes, sequence number resets
- Solution 2
  - use a clock for the sequence number
    - clocks don't reset on reboot, so we never lose sequence #
  - use a max lifetime for a packet
    - permits clocks to roll over
  - can get into **forbidden** region

Three-way Handshake

- Use different sequence number spaces for each direction
- Three messages used
  - Connection Request
    - send initial sequence number from caller to callee
  - Connection Request Acknowledgment
    - send ACK of initial sequence number from caller to callee
    - send initial sequence number from callee to caller
  - First Data TPDU
    - send ACK of initial sequence number from callee to caller
- Each Side Selects an initial number
  - it knows that the number is not currently valid
    - uses time of day
    - limits number of connects per unit time, but not data!

Example of Three-way Handshake



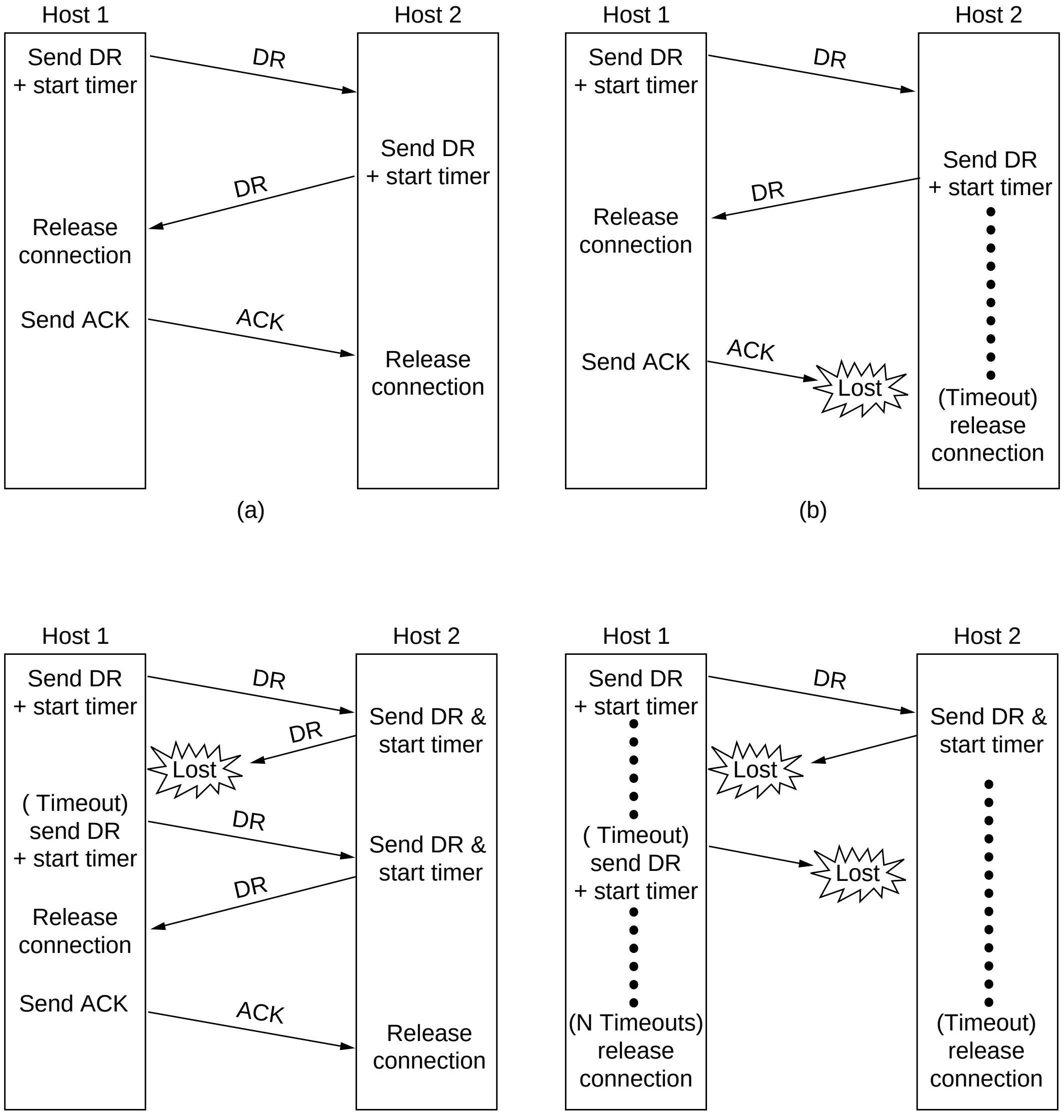
(a) (b) (c)

From: *Computer Networks*, 3rd Ed. by Andrew S. Tanenbaum, (c)1996 Prentice Hall.

Closing a Connection

- To prevent data loss,
  - both sides must agree they are done
- Problem: how to agree
  - possible that “I am done” messages will get lost
  - possible that “I ACK you are done” messages will get lost
- Solution:
  - initiator sends Disconnect Request, start DR timer
  - when initiated party receives DR, send DR and start DR timer
  - when initiator gets DR back, send ACK and release connection
  - when initiated gets ACK, release connection
  - if initiator times out, send new DR
  - if initiated times out, release connection

Connection Close Example



Host 1
Send DR + start timer
Release connection
Send ACK

Host 2
Send DR + start timer
Release connection

DR
DR
ACK

(a)

Host 1
Send DR + start timer
Release connection
Send ACK

Host 2
Send DR + start timer
(Timeout) release connection

DR
DR
ACK
Lost

(b)

Host 1
Send DR + start timer
(Timeout) send DR + start timer
Release connection
Send ACK

Host 2
Send DR & start timer
Send DR & start timer
Release connection

DR
DR
Lost
DR
DR
ACK

(c)

Host 1
Send DR + start timer
(Timeout) send DR + start timer
(N Timeouts) release connection

Host 2
Send DR & start timer
(Timeout) release connection

DR
Lost
Lost

(d)

From: *Computer Networks*, 3rd Ed. by Andrew S. Tanenbaum, (c)1996 Prentice Hall.

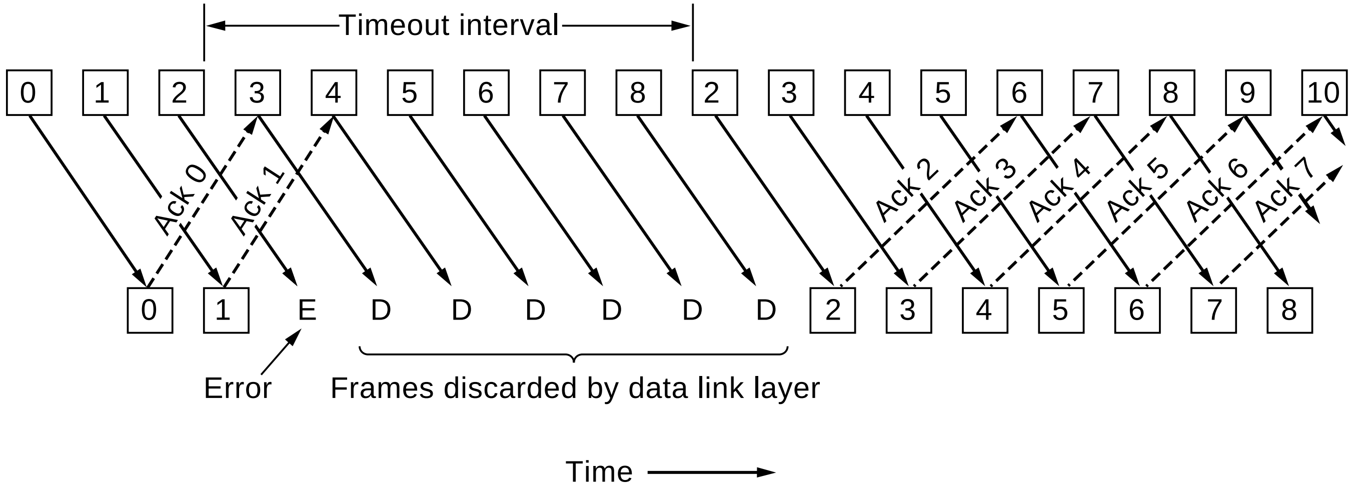
Lingering Half-Duplex Connections

- If a party (or a link) dies
  - can be left with dead connections
- Solution: use keep-alive packets
  - every n seconds, send a packet
  - if no packet is received after n * m seconds, cleanup

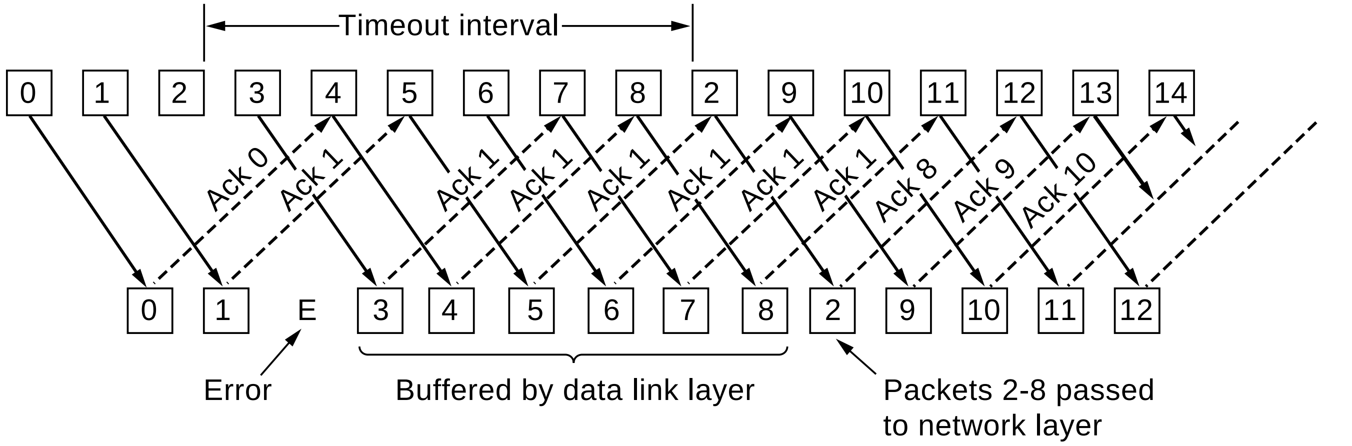
Sliding Window Protocol

- Need to
  - have multiple outstanding packets
  - limit total number of outstanding packets
  - permit re-transmissions to occur
- Sliding Window
  - permit at most N outstanding packets
  - when packet is ACK’d advance window to first non-ACK’d packet
- Retransmission
  - Go-back N
    - when a packet is lost, restart from that packet
    - provides in-order delivery, but wastes bandwidth
  - Selective Retransmission
    - use timeout to re-sent lost packet
    - use NACK as a **hint** that something was lost

Sliding Window Example



(a)



(b)

From: *Computer Networks*, 3rd Ed. by Andrew S. Tanenbaum, (c)1996 Prentice Hall.

Buffer Management

- Unreliable Network
  - sender must buffer all un-acked packets
  - receiver can buffer if space is available
    - if not, drop packet and wait to re-transmission
- Buffer Size
  - does one size fit all?
    - are TPDUs of uniform size?
  - might use a fixed size buffer smaller than max TPDU
    - requires support for multiple buffers per TPDU
- Possible to decouple buffer allocation from window
  - ACKs contain both buffer credits and ACKSs
- Buffer Copies
  - possible for each layer to copy the buffer, but this is slow
  - handoff pointers to data, but requires coordination between layers