Topics for This Week

• Transport Layer Services
• Connection Set-up and Tear-down

• Readings
  – Sections 6.2.2-6.2.3
Transport Layer

• Provides transport services to applications
• **End-host to end-host delivery** of data
• Aspects of transport services
  – Error detection and recovery
    • Errors (lost or corrupted data) detected at receiver?
    • Detected error corrected?
  – Timing
    • Timing between data preserved when delivered at receiver?
  – Framing
    • Data unit boundaries (e.g., message) preserved?
Common Transport Service Models

• Connectionless datagram
  – No promises, no timing
  – Error detection optional, no error recovery

• Connection-oriented
  – Error recovery, no timing

• Circuit-like
  – Timing preserved
  – No error recovery, optional error detection
Connection Paradigms

• Connection-oriented
  – Explicitly setup/tear down connections
  – Setup connection context ("connection state")
  – Initial sequence number, flow control window size
  – Exchange data within context of connection

• Connectionless service
  – Pure datagram
    • One-time unreliable send
  – Transaction oriented
    • Single request from sender, single reply from receiver
End-to-End Issues

• How to build a reliable, in-order delivery?
  – On top of an unreliable network layer service

• Potentially connects different hosts
  – Need explicit connection establishment and termination

• Potentially different RTT (round-trip time)
  – Need adaptive timeout mechanisms

• Potentially long delay in network
  – Need to be prepared for arrival of very old packets
  – Connection management
End to End Issues

- Potentially unreliable network service
  - Need to be prepared for
    - corrupted/lost and out-of-order packets
  - Error control
- Potentially different capacity at destination
  - Need to avoid overrunning receivers
  - flow control
- Potentially different network capacity
  - Need to be prepared for network congestion
  - Congestion control
Connection Management Issues

• How to identify a connection between end hosts?
  – Source/destination IP addresses + port numbers
  – Is it sufficient?
    • What about different incarnations?

• Potential problems
  – Network can delay, reorder, lose packets
  – Time-out/retransmission introduces duplicates of
    • Data, acknowledgement, connect, close packets
Connection Management Issues

• On packet arrival: is it real or memorex?
  – New connection request/release or an old one?
  – Transport protocols must create/maintain/destroy
    • Enough state to answer the memorex question

• Major issues
  – How to choose an identifier for each packet
    • So that no other packets currently in the network
      – Associated with this host have the same identifier
  – How to deal with old or duplicate (connect) messages
    • Delayed duplicate problem
Choosing Unique Identifier

• Choose an identifier (or sequence number) to
  – Distinguish any two outstanding packets/connections
    • associated with a host
  – Connection identified by
    • <host id., port no.> pairs plus a “unique” initial seq number
• Host id unique globally, why not sufficient to use
  – <host id., port no.> pairs to identify connections?
• Time stamp each pkt using a time-of-day clock?
• Solution: Assume maximum lifetime (T) for a pkt
  • Network layer kills packets when they reach max lifetime
Choosing Unique Identifier

• Approach 1: maintain state
  – Keep list of all values used in last 2T (why 2T?)
    • Don’t reuse value in list
    • If list lost: wait 2T
  – Concerns?

• Approach 2: what me worry?
  – Choose at random from large set (e.g. $2^{32}$) of numbers
    • Unlikely to choose new number previously chosen in last 2T
    • Can be combined with used value list for more protection
  – Good enough for many people (except academics)
Connection Setup

• Exchange control messages between two end hosts
  – To setup (or reject) connection before sending data

• Issues to be considered
  – How to handle lost messages
    • Use timer and retransmission
  – How to identify and handle old/duplicate messages
    • Keep some state info (e.g. seq. no. of packets sent etc)

• Two basic approaches
  – Two-way handshake (with timers)
  – Three-way handshake
Two-Way Handshake

• Connection initiator (client)
  – Choose unique identifier, x
  – Send `req-conn(x)` msg to the other side (server)

• Connection respondent (server)
  – Accepts connection with `acc-conn(x)` reply
**State Transition Diagram**
Two-Way Handshake (cont’d)
Two-Way Handshake (cont’d)

Client

Choose x

Timeout Retransmit

Conn. x Estab.

Server

req-conn(x)

ack-conn(x)

data(x+1)

Conn. x already estab, resend ack

Conn. x Estab.

data(x+1) rcvd

ack(x+1)

Ignore

duplicate message
Two-Way Handshake with Timer

- Connection initiator (the client)
  - Chooses a unique seq. no. x and send req-conn(x)

- Connection respondent (the server)
  - Sends ack-conn(x)

- What’s more?
  - After connection tear-down from client
    - Server \texttt{wont delete} connection record for x
      - \texttt{Until} it is sure that \texttt{no more} req-conn(x) in the network
    - \texttt{Hold} record until \texttt{T} (max pkt lifetime) after connection close
Client

Choose $x$

$\text{req-conn}(x)$

Server

$\text{Conn. } x$

$\text{Estab.}$

state: source/dest TSAPs
seq. no. $x$
status: conn. estab.

$\text{Conn. } x$

$\text{Estab.}$

Exchange Data

$\text{data}(x+1)$

$\text{ack}(x+1)$

$\text{close-conn}(x+n)$

$\text{Conn. } x \text{ closed}$

$\text{ack}(x+n)$

$\text{Conn. } x \text{ closed}$

server holds state record until $T$ after conn. close

$T$: max. packet life time

Release $\text{Conn. } x$ state reco
Three-Way Handshake

- **Connection initiator (the client)**
  - Chooses unique seqno x and sends req-conn(x)

- **Connection respondent (the server)**
  - Upon receiving req-conn(x)
    - Chooses *its own unique identifier*, y
    - Sends ack-conn(y,x)
  - Upon receiving ack-conn(y,x), client responds
    - With ack-conn(x+1,y)

- Why does server need to choose unique seqno y?
Three-Way Handshake

Client

Server

Conn. Estab.

req-conn(x)

ack-conn(y,x)

ack(x+1,y)

Conn. Estab.

Exchange Data

data(x+2)

data(y+1)

Exchange Data
Connection Setup: Summary

• How to deal with old and duplicate messages?
  – Two-way handshake with timer
    • Sender chooses unique identifier x
      – Allows receiver to detect old sender messages (with timer on x)
    • Receiver replies with x
      – Allows sender to detect old replies from receiver
  – Three-way handshake
    • Receiver also chooses its own unique identifier y
      – Requires sender to reply back using y
      – Allows receiver to detect old sender messages without timers
Closing a Connection

• Reaching agreement: two approaches
  – Abort: send close msg to peer, delete state info
    • What if close() message lost?
  – Graceful: send close msg, but before deleting state
    • Wait for peer to acknowledge close()

• Problem solved?
  – Can I decide to close, knowing that
    • Other entity also agreed to close and knows that I will close

• Can two armies coordinate their attacks
  – If communication is unreliable?
Two-Army Problem