Network Protocols

Transmission Control Protocol (TCP)
IP review

• IP provides just enough connected-ness
  • Global addressing
  • Hop-by-hop routing
• IP over everything
  • Ethernet, ATM, X.25, fiber, etc.
• Minimizes network state
• Unreliable datagram forwarding
TCP key features

- Sequencing
- Byte-stream delivery
- Connection-oriented
- Reliability
- Flow-control
- Congestion avoidance
TCP feature summary

Provides a completely reliable (no data duplication or loss), connection-oriented, full-duplex byte stream transport service that allows two application programs to form a connection, send data in either direction simultaneously and then terminate the connection.
Apparent contradiction

- IP offers best effort (unreliable) delivery
- TCP uses IP
- TCP provides completely reliable transfer
- How is this possible?
Achieving reliability

• Reliable connection start-up
• Reliable data transfer
  • Sender starts a timer
  • Receiver sends ACK when data arrives
  • Sender retransmits if timer expires before ACK is returned
• Reliable connection shutdown
Reliability illustrated
When do you retransmit?

- The time for an ACK to return depends on:
  - Distance between endpoints (propagation delay)
  - Network traffic conditions (congestion)
  - End system conditions (CPU, buffers)
- Packets can be lost, damaged or fragmented
- Network traffic conditions can change rapidly
Solving retransmission problem

- Keep running average of round trip time (RTT)
- Current average determines retransmission timer
- This is known as adaptive retransmission
- This is key to TCP's success
- How does each RTT sample affect the average?
  - What weight to you give each sample?
  - Higher weight means timer changes quickly
  - Lower weight means timer changes slowly
Adaptive retransmission illustrated
Flow control

• Match the sending rate with allowable receiver rate
• TCP uses a sliding window
  • Receiver advertises available buffer space
  • Also known as the window
  • Sender can transmit a full window without receiving an ACK for that transmitted data
• Ideally the window size allows pipe to remain full
Window size advertisement

- Each ACK carries receiver's current window size
  - Called the window advertisement
  - If zero, window is closed, no data can be sent
- Interpretation of window advertisement:
  - Receiver: I can accept $X$ octets or less unless I tell you otherwise
Window size illustrated
Window size: another picture
Byte stream sequencing

- Each segment carries a sequence number
- Sequencing helps ensure in order delivery
- TCP sequence numbers are fixed at 32 bits
  - Byte stream is not limited to $2^{32}$ bytes
  - Sequence number space can wrap
- Each side has an initial sequence number (ISN)
  - Exchanged during connection establishment
- Receiver ACKs cumulative octets (bytes)
TCP segment illustrated
Application multiplexing

- OS independent identifier for a network process
- Each application assigned a unique 16-bit integer
  - Called a port number
- Server applications
  - Use standard, well-known port numbers
  - Usually low numbered port numbers
- Clients
  - Obtain unused number from protocol software
  - Usually uses high numbered port numbers
TCP connection start-up

- The three-way handshake used
- Servers use a passive open
  - Application sits waiting on an open port
- Clients use an active open
  - Application requests a connection to server
- Initial sequence number (ISN) exchange is the primary goal
- Other parameters/options can also be exchanged
  - e.g. Window scale, maximum segment size, etc.
3-way handshake illustrated
Connection shutdown illustrated
Congestion principles

- Flow control
  - Matching the sending and receiving rates
- Congestion control
  - Active response to network overload conditions
  - End hosts cannot control congestion per se
  - Network devices (routers) do this
- Congestion avoidance
  - Cautionary response to presumed conditions
  - TCP does this
TCP congestion control

- Recall sliding window (advertised window)
  - Receiver based control of sending rate
- Congestion window is sender based control
- Sender transmits min(cwnd, advertised window)
  - This value is the transmission window
- TCP sender infers network conditions and adjusts
TCP retransmission

• TCP starts timer after sending a segment
• If ACK returns, reset timer
• If time-out occurs, retransmit and increase timer
  • This is a back-off process
• Can't retransmit forever, need some upper bound
• Eventually TCP would give up
  • Maximum time-out must be at least 60 seconds
Estimating round trip time (RTT)

• TCP measures RTT for which to calculate timers
• If ACKs return quickly, timers should be short
  • If loss occurs, recovery happens quickly
• If ACKs return slowly, timers should be long
  • If delays occur, retransmits not sent needlessly
• Keep a smoothed running average of RTT
  • Smoothed RTT used to adjust retransmit timer
  • Karn's algorithm says ignore ACKs of retransmits
TCP slow start

• Recall that min(cwnd, awnd) = transmission window
• Rather than sending a full window at start-up...
• Initialize cwnd to 1 maximum segment size (MSS)
• Increase cwnd by 1 MSS for every ACK returned
• Obviously don't go past advertised window!
• This can actually be quite fast, exponential!
TCP slow start illustrated
TCP congestion avoidance

• If a retransmission timer expires, slow down
• Set slow start threshold = transmission window x ½
  • This is sshthresh
• Set cwnd back to 1 MSS
• Transmit min(cwnd, advertised window) as usual
• Do slow start until transmission window = sshthresh
• Thereafter, increase cwnd by 1/cwnd per ACK
  • Linear increase instead of exponential
Congestion avoidance illustrated
Duplicate ACKs

• Recall ACKs acknowledge cumulative octets
• TCP receiver sends an immediate ACK if it receives an out-of-order segment
• This is a duplicate ACK
• This dupe ACK informs the sender and tells it what sequence number the receiver expected
• Its unclear whether dupe ACKs indicate loss or simply packet re-ordering on the network
• But, multiple duplicate ACKs probably indicate loss
TCP fast retransmit

• If sender gets $\geq 3$ dupe ACKs, assume loss
• Immediately retransmit, don't wait for timer to expire
• Goto fast recovery
TCP fast recovery

• Duplicate ACKs indicate data is still flowing
• If there was a loss event, it was probably temporary
• Go directly to congestion avoidance
  • Not all the way into slow start!
  • Don't want to start off with just a 1 MSS window
• This is the fast recovery algorithm
  • Minus a few minor details
Other TCP stuff

• Selective ACK (SACK) option
• Window scale option
• Timestamp option
• Persist timer (window probes)
• Silly window syndrome
• Keepalive timer
• Nagle algorithm