Announcements

• Upcoming dates
  – Wed, 4/30: Lab 3 due, Lab 4 out
  – Fri, 5/2: Midterm Review
  – Mon, 5/5: In-class midterm
  – Wed, 5/14: Lab 4 due

• Lab 3 is more complex than Lab 1 or Lab 2, so start now
TCP Overview

• Network layer protocol
• Properties
  – Full-duplex connection
    • Two-way communication between \((IP, port)_{src}\) and \((IP, port)_{dst}\)
    • Connection setup before any transfer
    • Connection teardown after transfer finishes
    • Each connection creates state in sending and receiving hosts
    • How is this different than with a VC network?
  – Reliable: resends lost/corrupted segments
  – In-order: buffers at sender and receiver
  – Stream of bytes: looks like a file you can R/W to
TCP Segments

- Provide illusion of a stream of bytes, but we actually are going over a datagram network using packets (IP)
- Data is carried in TCP segments and placed into an IP packet
Sequence Numbers

ISN (initial sequence number)

Seq number = First byte of segment

Ack seq number = next expected byte

Credit: CS244A Handout 8
Three-Way Handshake

- Exchange initial sequence numbers at connection startup
  - Client’s ISN = x
  - Server’s ISN = y
- Send a special segment with SYN bit set ("synchronize")
- SYN takes up one “byte”
Shutdown

• Either side can initiate shutdown
• Can shutdown only one side of connection, if desired
• TIME_WAIT state to handle case of whether last ACK was lost
Sockets and TCP

- `socket, bind`
- `connect`
- `send/recv`
- `shutdown (SHUT_RDWR)`

Communication flow:
- SYN
- SYN/ACK
- ACK
- ::
- FIN
- ACK
- FIN
- ACK

- `socket, bind`
- `listen`
- `accept`
- `send/recv`
- `shutdown (SHUT_RDWR)`
Sender Window

- Window size: maximum amount of unacknowledged bytes/segments
- Usually dynamically adjusted in response to congestion
- Must be smaller than receiver window
- Local state maintained at sender
Example: Ideal TCP Transfer Rate

• Assume an ideal TCP connection between two hosts A and B. What is the maximum transmission rate between the two hosts in terms of:
  – $W$, the window size in bytes
  – $RTT$, the round trip time
  – $R$, the transmission rate of the link
Solution: Ideal TCP Transfer Rate

- So ideal transfer rate is $W/RTT$—independent of link BW!
Receiver Window

• Advertised to sender in TCP header
• Amount of out-of-order bytes the receiver will buffer
• Sender window cannot be larger than advertised receiver window
• Example
  – \( \text{RecvWind} = \) receiver window in bytes
  – Last ack to sequence number \( x \)
  – Then receiver will buffer any bytes in the sequence number range \([x, x+\text{RecvWind})\)
Example: TCP RST Attack

• Suppose we have a long-lived TCP connection (like a BGP session), and we want to maliciously terminate it
  – Suppose we know the IP and port numbers for both sides of the connection
  – Then sending a TCP RST packet will immediately terminate the session
• Given a receiver window size of 8K, what is the chance that a RST packet with a random sequence number will terminate the connection?
• How many RST packets are needed to span the entire sequence number space?
• Using 58 byte RST packets on a 10 Mbps link, how long does it take to generate this number of packets?
Solution: TCP RST Attack

• Given a receiver window size of 8K, what is the chance that a RST packet with a random sequence number will terminate the connection?
  – $2^{13}/2^{32} = 2^{-19} = 1$ in half a million chance

• How many RST packets are needed to span the entire sequence number space?
  – $2^{19}$ packets

• Using 58 byte RST packets on a 10 Mbps link, how long does it take to generate this number of packets?
  – $2^{19}$ packets * 58 bytes/packet * 8 bits/byte / 10 Mbps = 24 seconds
Flow Control

• Don’t want to overwhelm the network or the receiver with packets
• Adjust cwnd (congestion window) dynamically in response to loss events
  – Sender window = min(cwnd, rwnd)
• Congestion window resized using AIMD
  – When connection starts, start with window size of 1
  – As long as segments are acked:
    • Increase window size by 1 segment size every RTT (additive increase)
  – If loss is detected:
    • Halve window size (multiplicative decrease)
TCP Sawtooth

Window Size vs. Time (t)

Timeouts

Src

Dest

Credit: CS244A Handout 8
Optimizations

• Slow start initialization
  – Increase cwnd by MSS for every ack (doubles cwnd for every RTT)
  – Suppose we detect first loss at window size $W$
    • Set $ssthresh := \frac{W}{2}$
    • Set $cwnd := 1$
    • Use slow start until our window size is $ssthresh$
    • Then use AIMD (congestion avoidance mode)

• Fast retransmit and fast recovery if we get three duplicate acks during slow start
  – Suppose we send 1, 2, 3, 4, 5, ... , 8, 9, 10
  – Get acks 1, 2, 3, 4, 5, ..., 8, 8, 8
  – Probably 9th segment has been lost, so:
    • Resend it before retransmit timer expires (fast retransmit)
    • Set $cwnd := ssthresh$ rather than 1 and go into AIMD (fast recovery)
TCP Sawtooth With Optimizations

Credit: CS244A Handout 8
Example: Reaching Maximum Congestion Window Size with Slow Start

• Assume this TCP implementation:
  – MSS = 125 bytes
  – RTT is fixed at 100 ms (even when buffers start filling)
  – Uses slow start with AIMD
  – Analyze one flow between A and B, where bottleneck link is 10 Mbps
  – Ignore receiver window

• What is the maximum congestion window size?
  – For one flow (ideally), $W/RTT = \text{rate}$
  – $W = (100 \text{ ms} \times 10 \text{ Mbps}) / (8 \text{ bits/byte}) = 125000 \text{ bytes}$

• How long does it take to reach this size?
  – Slow start grows $cwnd$ exponentially, starting from one MSS
  – Find $n$ s.t. $125 \times 2^n \geq 125000$
    • $n = 10$
  – Then it takes $n \times RTT = 1 \text{ s}$ to reach the max $cwnd$ size
Detecting Losses

- Each segment sent has a retransmit timer
- If a segment’s retransmit timer expires before ack for that segment arrive, assume loss
- Retransmission timeout (RTO) for timer based on exponential weighted moving average of the previous RTTs and variance between RTT samples
  - \( \text{EstRTT}_k = (1 - \alpha) \cdot \text{EstRTT}_{k-1} + \alpha \cdot \text{SampleRTT}_k \)
    - Recommended \( \alpha \) is 0.125
    - \( \text{EstRTT} \) is an EWMA of the \( \text{SampleRTT} \)
  - \( \text{DevRTT}_k = (1 - \beta) \cdot \text{DevRTT}_{k-1} + \beta \cdot |\text{SampleRTT}_k - \text{EstRTT}_k| \)
    - Recommended \( \beta \) is 0.25
    - \( \text{DevRTT} \) is an EWMA of the difference between sampled and estimated RTT
- \( \text{RTO} = \text{EstRTT} + 4 \cdot \text{DevRTT} \)
Lab 3 Operation

ACK packets

STDIN

handle_pkt

handle_ack

Send Buffer

timer

Data packets

Server

Receiver State

handle_pkt

STDOUT

Client