15-744: Computer Networking

L-9 TCP Basics

TCP Basics

- TCP reliability
- Assigned reading
  - [FF96] Simulation-based Comparisons of Tahoe, Reno, and SACK TCP

Key Things You Should Know Already

- Port numbers
- TCP/UDP checksum
- Sliding window flow control
  - Sequence numbers
- TCP connection setup

Overview

- TCP introduction
- TCP reliability: timer-driven
- TCP reliability: data-driven

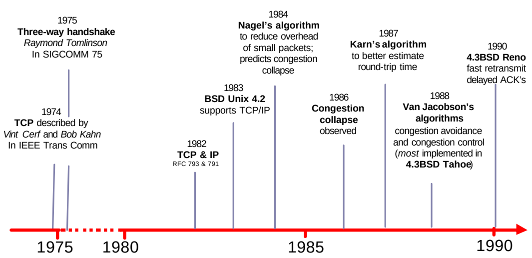
Introduction to TCP

- Communication abstraction:
  - Reliable
  - Ordered
  - Point-to-point
  - Byte-stream
  - Full duplex
  - Flow and congestion controlled
- Protocol implemented entirely at the ends
  - Fate sharing

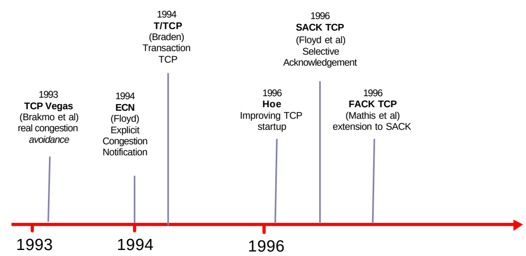
What's Different From Link Layers?

- Logical link vs. physical link
  - Must establish connection
- Variable RTT
  - May vary within a connection
- Reordering
  - How long can packets live → max segment lifetime
- Can't expect endpoints to exactly match link
  - Buffer space availability
- Transmission rate
  - Don't directly know transmission rate

Evolution of TCP



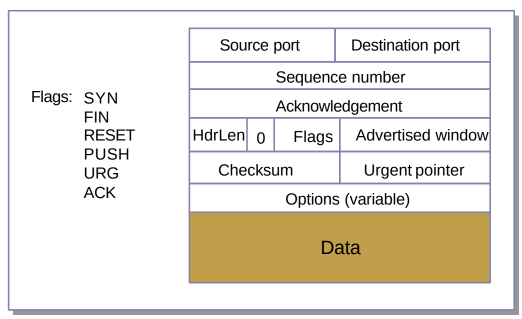
TCP Through the 1990s



Integrity & Demultiplexing

- Port numbers
  - Demultiplex from/to process
  - Servers wait on well known ports (/etc/services)
- Checksum
  - Is it sufficient to just checksum the packet contents?
  - No, need to ensure correct source/destination
    - Pseudoheader – portion of IP hdr that are critical
    - Checksum covers Pseudoheader, transport hdr, and packet body
- UDP provides just integrity and demux

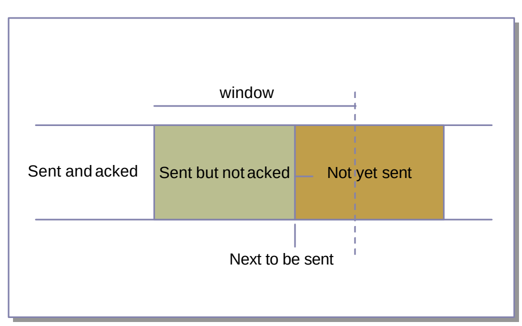
TCP Header



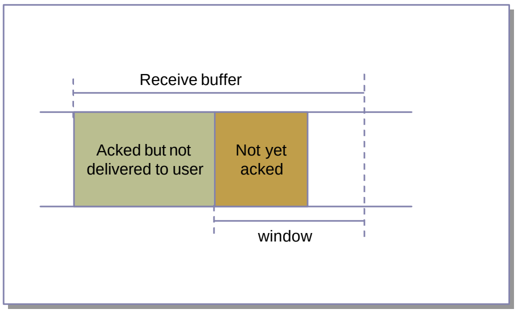
TCP Flow Control

- TCP is a sliding window protocol
  - For window size *n*, can send up to *n* bytes without receiving an acknowledgement
  - When the data is acknowledged then the window slides forward
- Each packet advertises a window size
  - Indicates number of bytes the receiver has space for
- Original TCP always sent entire window
  - Congestion control now limits this

Window Flow Control: Send Side



Window Flow Control: Receive Side

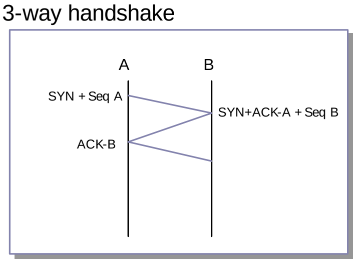


TCP Persist

- What happens if window is 0?
  - Receiver updates window when application reads data
  - What if this update is lost?
- TCP Persist state
  - Sender periodically sends 1 byte packets
  - Receiver responds with ACK even if it can't store the packet

Connection Establishment

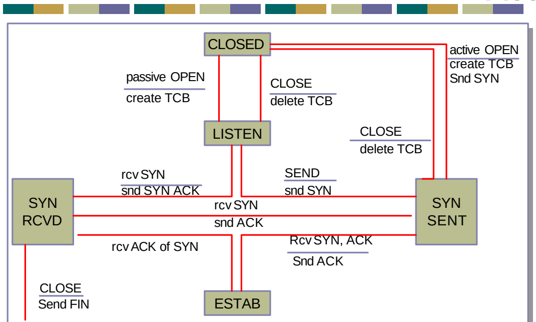
- A and B must agree on initial sequence number selection
  - Use 3-way handshake



Sequence Number Selection

- Why not simply chose 0?
- Must avoid overlap with earlier incarnation

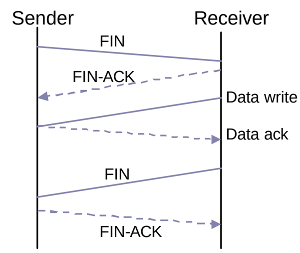
Connection Setup



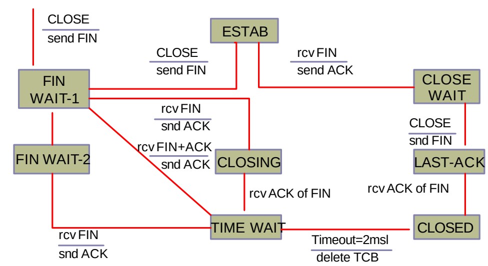
Connection Tear-down

- Normal termination
  - Allow unilateral close
- TCP must continue to receive data even after closing
- Cannot close connection immediately
  - What if a new connection restarts and uses same sequence number?

Tear-down Packet Exchange



Connection Tear-down



Detecting Half-open Connections

	TCP A		TCP B
1.	(CRASH)		(send 300, receive 100)
2.	CLOSED		ESTABLISHED
3.	SYN-SENT → <SEQ=400><CTL=SYN>	→	(??)
4.	(!!) ← <SEQ=300><ACK=100><CTL=ACK>	←	ESTABLISHED
5.	SYN-SENT → <SEQ=100><CTL=RST>	→	(Abort!!)
6.	SYN-SENT		CLOSED
7.	SYN-SENT → <SEQ=400><CTL=SYN>	→	

Observed TCP Problems

- Too many small packets
  - Silly window syndrome
  - Nagel's algorithm
- Initial sequence number selection
- Amount of state maintained

Silly Window Syndrome

- Problem: (Clark, 1982)
  - If receiver advertises small increases in the receive window then the sender may waste time sending lots of small packets
- Solution
  - Receiver must not advertise small window increases
  - Increase window by min(MSS,RecvBuffer/2)

Nagel's Algorithm

- Small packet problem:
  - Don't want to send a 41 byte packet for each keystroke
  - How long to wait for more data?
- Solution:
  - Allow only one outstanding small (not full sized) segment that has not yet been acknowledged

Why is Selecting ISN Important?

- Suppose machine X selects ISN based on predictable sequence
- Fred has .rhosts to allow login to X from Y
- Evil Ed attacks
  - Disables host Y – denial of service attack
  - Make a bunch of connections to host X
  - Determine ISN pattern a guess next ISN
  - Fake pkt1: [<src Y><dst X>, guessed ISN]
  - Fake pkt2: desired command

Time Wait Issues

- Web servers not clients close connection first
  - Established → Fin-Waits → Time-Wait → Closed
  - Why would this be a problem?
- Time-Wait state lasts for 2 * MSL
  - MSL is should be 120 seconds (is often 60s)
  - Servers often have order of magnitude more connections in Time-Wait

Overview

- TCP introduction
- TCP reliability: timer-driven
- TCP reliability: data-driven

Reliability Challenges

- Like reliability on links
  - Similar techniques (timeouts, acknowledgements, etc.)
- New challenges
  - Congestion related losses
  - Variable packet delays
    - What should the timeout be?
  - Reordering of packets
    - Ensure sequences numbers are not reused
    - How long to packets live?
      - MSL = 120 seconds based on IP behavior

Standard Data Transfer

- Sliding window with cumulative acks
  - Ack field contains last in-order packet received
  - Duplicate acks sent when out-of-order packet received
- Does TCP need to send an ack for every packet?
  - Delayed acks

Delayed ACKS

- Problem:
  - In request/response programs, you send separate ACK and Data packets for each transaction
- Solution:
  - Don't ACK data immediately
  - Wait 200ms (must be less than 500ms – why?)
  - Must ACK every other packet
  - Must not delay duplicate ACKs

Round-trip Time Estimation

- Wait at least one RTT before retransmitting
- Importance of accurate RTT estimators:
  - Low RTT → unneeded retransmissions
  - High RTT → poor throughput
- RTT estimator must adapt to change in RTT
  - But not too fast, or too slow!
- Spurious timeouts
  - "Conservation of packets" principle – more than a window worth of packets in flight

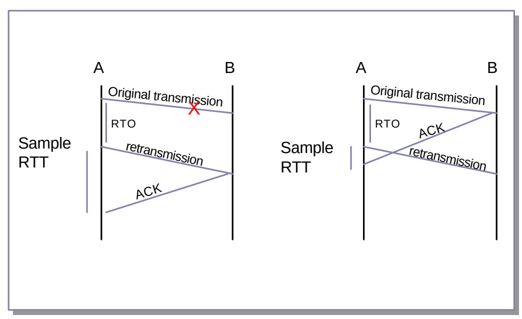
Initial Round-trip Estimator

- Round trip times exponentially averaged:
  - New RTT = α (old RTT) + (1 - α) (new sample)
  - Recommended value for α: 0.8 - 0.9
    - 0.875 for most TCP's
- Retransmit timer set to β RTT, where $\beta = 2$
  - Every time timer expires, RTO exponentially backed-off
  - Like Ethernet
- Not good at preventing spurious timeouts

Jacobson's Retransmission Timeout

- Key observation:
  - At high loads round trip variance is high
- Solution:
  - Base RTO on RTT and standard deviation or RRTT
  - rttvar = χ * dev + (1- χ)rttvar
    - dev = linear deviation
    - Inappropriately named – actually smoothed linear deviation

Retransmission Ambiguity



Karn's RTT Estimator

- Accounts for retransmission ambiguity
- If a segment has been retransmitted:
  - Don't count RTT sample on ACKs for this segment
  - Keep backed off time-out for next packet
  - Reuse RTT estimate only after one successful transmission

Timestamp Extension

- Used to improve timeout mechanism by more accurate measurement of RTT
- When sending a packet, insert current timestamp into option
  - 4 bytes for seconds, 4 bytes for microseconds
- Receiver echoes timestamp in ACK
  - Actually will echo whatever is in timestamp
- Removes retransmission ambiguity
  - Can get RTT sample on any packet

Timer Granularity

- Many TCP implementations set RTO in multiples of 200,500,1000ms
- Why?
  - Avoid spurious timeouts – RTTs can vary quickly due to cross traffic
  - Make timers interrupts efficient

Overview

- TCP introduction
- TCP reliability: timer-driven
- TCP reliability: data-driven

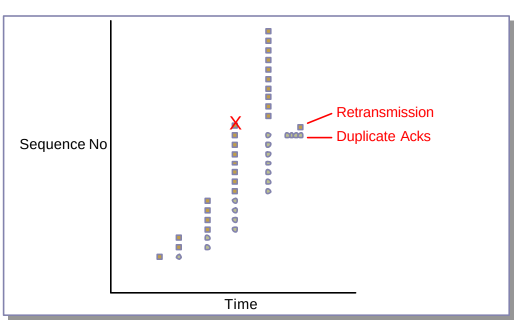
TCP Flavors

- Tahoe, Reno, Vegas
- TCP Tahoe (distributed with 4.3BSD Unix)
  - Original implementation of Van Jacobson's mechanisms (VJ paper)
  - Includes:
    - Slow start
    - Congestion avoidance
    - Fast retransmit

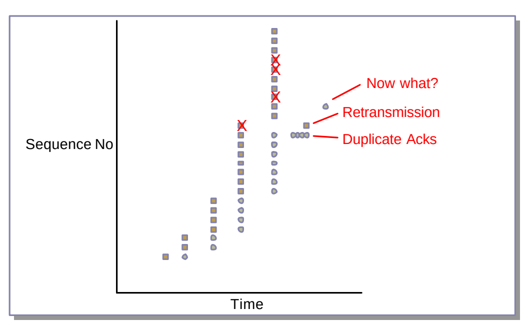
Fast Retransmit

- What are duplicate acks (dupacks)?
  - Repeated acks for the same sequence
- When can duplicate acks occur?
  - Loss
  - Packet re-ordering
  - Window update – advertisement of new flow control window
- Assume re-ordering is infrequent and not of large magnitude
  - Use receipt of 3 or more duplicate acks as indication of loss
  - Don't wait for timeout to retransmit packet

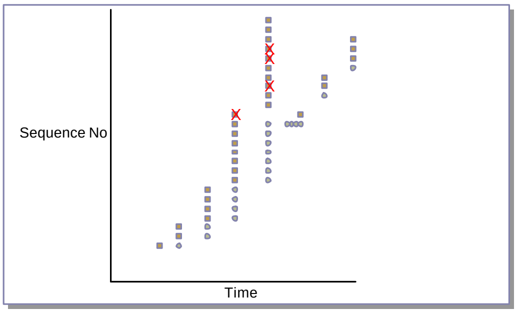
Fast Retransmit



Multiple Losses



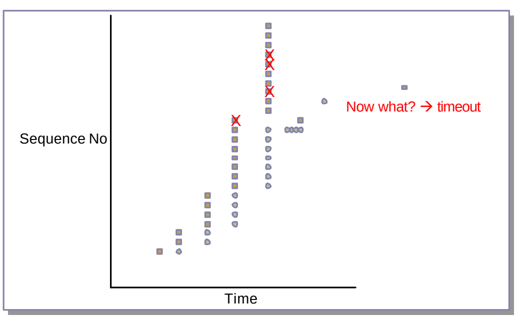
Tahoe



TCP Reno (1990)

- All mechanisms in Tahoe
- Addition of fast-recovery
  - Opening up congestion window after fast retransmit
- Delayed acks
- Header prediction
  - Implementation designed to improve performance
  - Has common case code inlined
- With multiple losses, Reno typically timeouts because it does not see duplicate acknowledgements

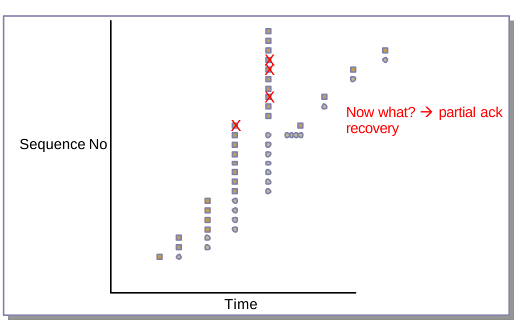
Reno



NewReno

- The ack that arrives after retransmission (partial ack) should indicate that a second loss occurred
- When does NewReno timeout?
  - When there are fewer than three dupacks for first loss
  - When partial ack is lost
- How fast does it recover losses?
  - One per RTT

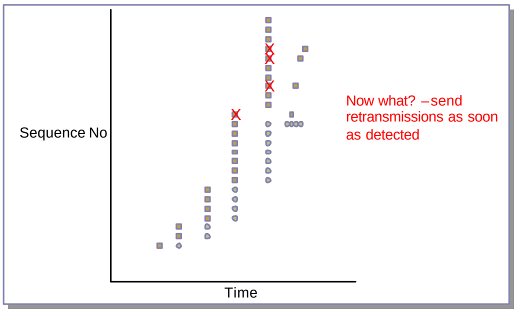
NewReno



SACK

- Basic problem is that cumulative acks only provide little information
  - Ack for just the packet received
    - What if acks are lost? → carry cumulative also
    - Not used
  - Bitmask of packets received
    - Selective acknowledgement (SACK)
- How to deal with reordering

SACK



Performance Issues

- Timeout >> fast rexmit
  - Need 3 dupacks/sacks
  - Not great for small transfers
    - Don't have 3 packets outstanding
  - What are real loss patterns like?
- Right edge recovery
  - Allow packets to be sent on arrival of first and second duplicate ack
  - Helps recovery for small windows
- How to deal with reordering?

TCP Extensions

- Implemented using TCP options
  - Timestamp
  - Protection from sequence number wraparound
  - Large windows

Protection From Wraparound

- Wraparound time vs. Link speed
  - 1.5Mbps: 6.4 hours
  - 10Mbps: 57 minutes
  - 45Mbps: 13 minutes
  - 100Mbps: 6 minutes
  - 622Mbps: 55 seconds → < MSL!
  - 1.2Gbps: 28 seconds
- Use timestamp to distinguish sequence number wraparound

Large Windows

- Delay-bandwidth product for 100ms delay
  - 1.5Mbps: 18KB
  - 10Mbps: 122KB > max 16bit window
  - 45Mbps: 549KB
  - 100Mbps: 1.2MB
  - 622Mbps: 7.4MB
  - 1.2Gbps: 14.8MB
- Scaling factor on advertised window
  - Specifies how many bits window must be shifted to the left
  - Scaling factor exchanged during connection setup

Maximum Segment Size (MSS)

- Exchanged at connection setup
  - Typically pick MTU of local link
- What all does this effect?
  - Efficiency
  - Congestion control
  - Retransmission
- Path MTU discovery
  - Why should MTU match MSS?

Next Lecture: Congestion Control

- Congestion control basics
- TCP congestion control
- Assigned reading
  - [JK88] Congestion Avoidance and Control
  - [CJ89] Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks