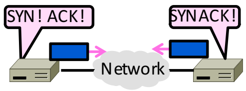
Operating Systems and Networks

TCP Summary

Adrian Perrig
Network Security Group
ETH Zürich

Connection Establishment (6.5.5, 6.5.7, 6.2.2)

- How to set up connections
  - We'll see how TCP does it

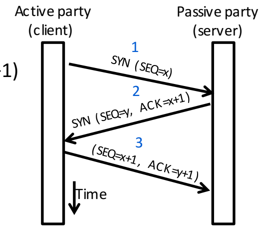


Connection Establishment

- Both sender and receiver must be ready before we start the transfer of data
  - Need to agree on a set of parameters
  - e.g., the Maximum Segment Size (MSS)
- This is signaling
  - It sets up state at the endpoints
  - Like "dialing" for a telephone call

Three-Way Handshake

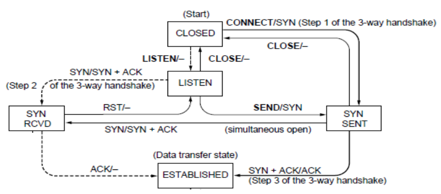
- Three steps:
  - Client sends SYN(x)
  - Server replies with SYN(y)ACK(x+1)
  - Client replies with ACK(y+1)
  - SYNs are retransmitted if lost
- Sequence and ack numbers carried on further segments



TCP Connection State Machine

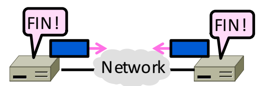
- Captures the states (rectangles) and transitions (arrows)
  - A/B means event A triggers the transition, with action B

Both parties run instances of this state machine



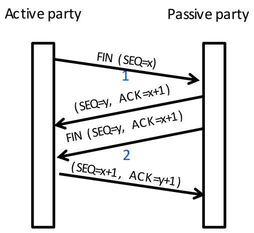
Connection Release (6.5.6-6.5.7, 6.2.3)

- How to release connections
  - We'll see how TCP does it



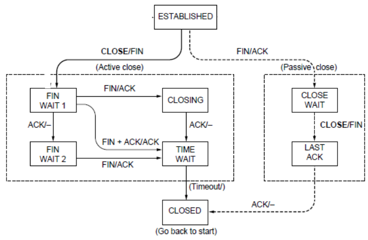
TCP Connection Release

- Two steps:
  - Active party sends FIN(x), passive party sends ACK
  - Passive party sends FIN(y), active party sends ACK
  - FINs are retransmitted if lost
- Each FIN/ACK closes one direction of data transfer



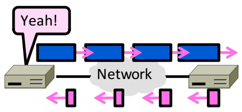
TCP Connection State Machine

Both parties run instances of this state machine



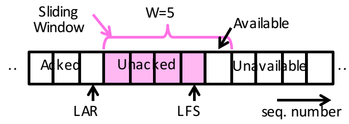
Sliding Windows (§3.4, §6.5.8)

- The sliding window algorithm
  - Pipelining and reliability
  - Building on Stop-and-Wait



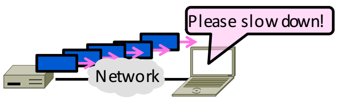
Sliding Window – Sender

- Sender buffers up to W segments until they are acknowledged
  - LFS=LAST FRAME SENT, LAR=LAST ACK REC'D
  - Sends while LFS – LAR ≤ W



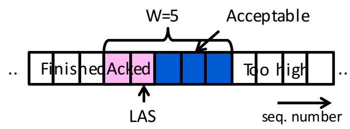
Flow Control (§6.5.8)

- Adding flow control to the sliding window algorithm
  - To slow the over-enthusiastic sender



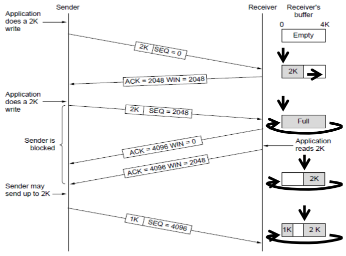
Flow Control

- Avoid loss at receiver by telling sender the available buffer space
  - WIN=#Acceptable, not W (from LAS)



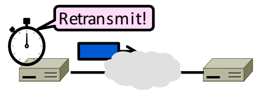
Flow Control (3)

- TCP-style example
  - SEQ/ACK sliding window
  - Flow control with WIN
  - SEQ + length < ACK+WIN
  - 4KB buffer at receiver
  - Circular buffer of bytes



Retransmissions

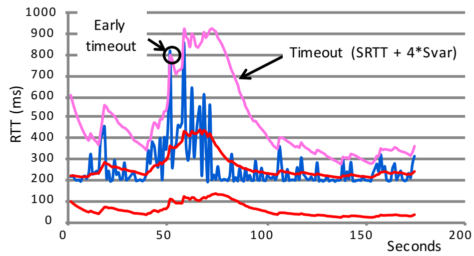
- With sliding window, the strategy for detecting loss is the timeout
  - Set timer when a segment is sent
  - Cancel timer when ack is received
  - If timer fires, retransmit data as lost



Adaptive Timeout

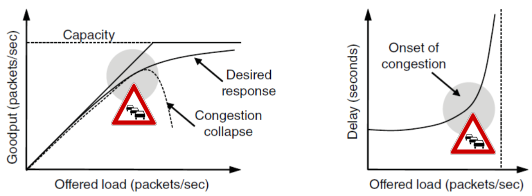
- Keep smoothed estimates of the RTT (1) and variance in RTT (2)
  - Update estimates with a moving average
  1. $SRTT_{N+1} = 0.9*SRTT_N + 0.1*RTT_{N+1}$
  2. $Svar_{N+1} = 0.9*Svar_N + 0.1*|RTT_{N+1} - SRTT_{N+1}|$
- Set timeout to a multiple of estimates
  - To estimate the upper RTT in practice
  - TCP $Timeout_N = SRTT_N + 4*Svar_N$

Example of Adaptive Timeout (2)



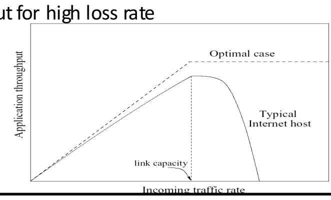
Effects of Congestion

- What happens to performance as we increase the load?



Congestion Characteristics

- Link flooding causes high loss rates for incoming traffic
- Mathis, Semke, Mahdavi, Ott [Sigcomm '97]:
  TCP Throughput ~ $MSS/RTT*c*q^{-1/2}$
  q is loss prob, c is constant close to 1
- Note: very low throughput for high loss rate
- Result
  - Few legitimate clients served during congestion



Bandwidth Allocation

- Important task for network is to allocate its capacity to senders
  - Good allocation is efficient and fair
- Efficient means most capacity is used but there is no congestion
- Fair means every sender gets a reasonable share the network

19

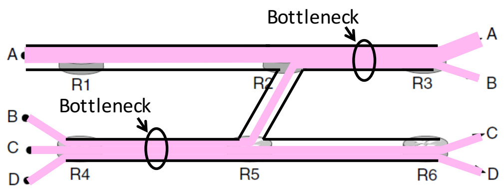
Max-Min Fairness

- Intuitively, flows bottlenecked on a link get an equal share of that link
- Max-min fair allocation is one that:
  - Increasing the rate of one flow will decrease the rate of a smaller flow
  - This "maximizes the minimum" flow

20

Max-Min Example

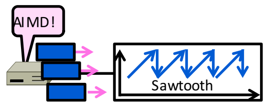
- When rate=2/3, flow A bottlenecks R2—R3. Done.



21

Additive Increase Multiplicative Decrease (AIMD) (§6.3.2)

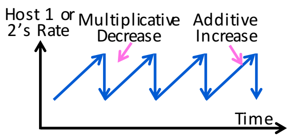
- Bandwidth allocation models
  - Additive Increase Multiplicative Decrease (AIMD) control law



22

AIMD Sawtooth

- Produces a "sawtooth" pattern over time for rate of each host
  - This is the TCP sawtooth (later)



23

AIMD Properties

- Converges to an allocation that is efficient and fair when hosts run it
  - Holds for more general topologies
- Other increase/decrease control laws do not! (Try MIAD, MIMD, AIAD)
- Requires only binary feedback from the network

24

Feedback Signals

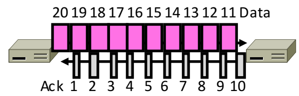
- Several possible signals, with different pros/cons
  - We'll look at classic TCP that uses packet loss as a signal

Signal	Example Protocol	Pros / Cons
Packet loss	TCP NewReno Cubic TCP (Linux)	+Hard to get wrong -Hear about congestion late
Packet delay	Compound TCP (Windows)	+Hear about congestion early -Need to infer congestion
Router indication	TCPs with Explicit Congestion Notification	+Hear about congestion early -Require router support

25

Sliding Window ACK Clock

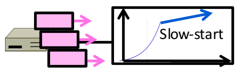
- Each in-order ACK advances the sliding window and lets a new segment enter the network
  - ACKs "clock" data segments



26

TCP Slow Start (§6.5.10)

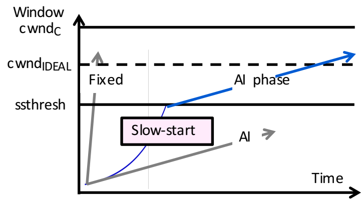
- How TCP implements AIMD, part 1
  - "Slow start" is a component of the AI portion of AIMD



27

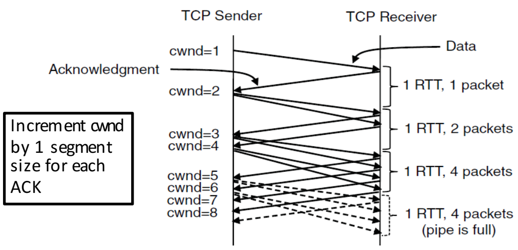
Slow-Start Solution

- Combined behavior, after first time
  - Most time spend near right value



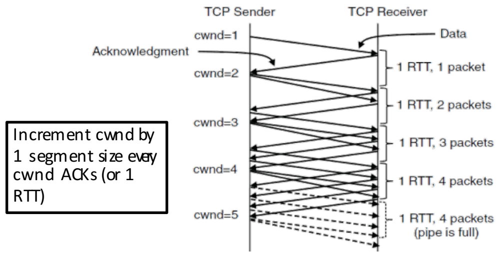
28

Slow-Start (Doubling) Timeline



29

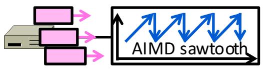
Additive Increase Timeline



30

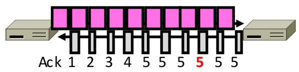
TCP Fast Retransmit / Fast Recovery (§6.5.10)

- How TCP implements AIMD, part 2
  - "Fast retransmit" and "fast recovery" are the MD portion of AIMD

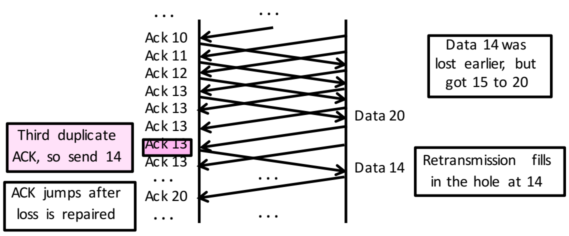


Fast Retransmit

- Treat three duplicate ACKs as a loss
  - Retransmit next expected segment
  - Some repetition allows for reordering, but still detects loss quickly

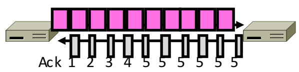


Fast Retransmit (2)

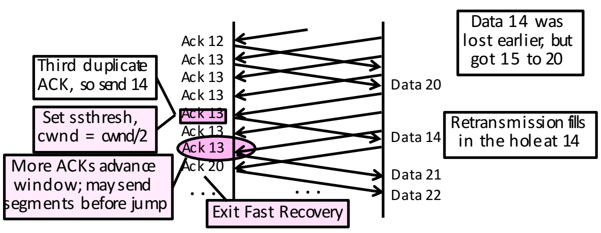


Fast Recovery

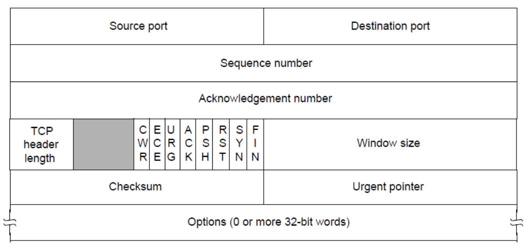
- First fast retransmit, and MD cwnd
- Then pretend further duplicate ACKs are the expected ACKs
  - Lets send new segments for received ACKs
  - Reconcile views when the ACK jumps



Fast Recovery (2)



TCP Header



Interesting Questions

- How is MSS / MTU determined?
- What happens if UDP does not implement congestion control?
  - Do modern UDP applications need to implement congestion control?
  - What is the relationship with network neutrality?
- What if different congestion control schemes are used concurrently? What can go wrong?
- Can a malicious host obtain an unfair advantage?
- Why size would you pick for router buffers? Large or small? Which one will result in better performance if standard TCP is used?

37