TCP Overview

CSci551: Computer Networks
SP2006 Thursday Section
John Heidemann

What does TCP Provide?

- connection establishment: y
- connectionless communication: n
- congestion avoidance: y
- differentiated services: n (type-of-service bits in IP)
- duplicate packet detection: y
- flow control: y
- loss recovery: y
- message or record boundaries: n (wanted by some protocols like RPC)
- ordered data delivery to the application: y
- out-of-order data delivery to the application: n (wanted by streaming media)
- quality-of-service: n
- urgent data indication: y

Where and Why is TCP Used?

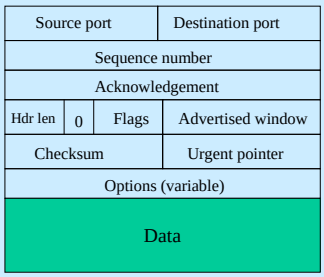
- where: anywhere reliable communication is needed
  - ftp, http, BPG, …
- why?
  - lots of folks want TCP's mix of features (reliability, in-order delivery, connections…)
  - it's already there
  - algorithms
    - window-based flow control
    - congestion control, AIMD
    - reliablity (checksums, retransmissions, etc.)
      - NACK vs. ACK
      - exponential backoff
      - RTT estimation algorithm
      - RTO (Retx timeout) estimation
  - => lot of very careful design and engineering
    - important reason to use TCP is to leveage this prior work

TCP in a Nutshell

- abstraction:
  - reliable
  - ordered
  - point-to-point
  - byte-stream
- mechanisms
  - (prior page)

TCP Header

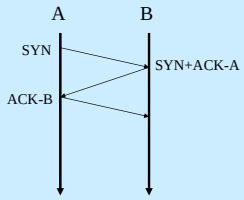
Flags: SYN
FIN
RESET
PUSH
URG
ACK



Agenda

- **connection setup and teardown**
  - **initial sequence number selection**
  - **passive/active open**
  - **time-wait**
- flow control
- congestion control practice and theory
- loss recovery
- security
- performance

Three-Way Handshake



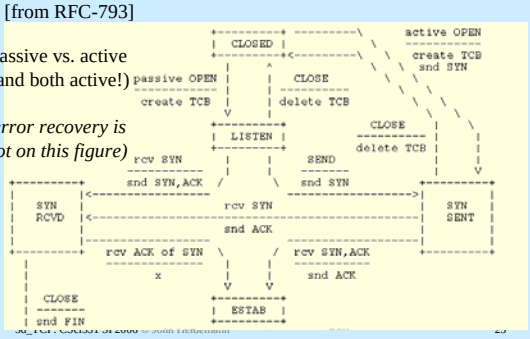
- why?
  - set the initial sequence number
    - need number order packets
    - try to start with different ISNs each time to make accidental replay of packets less likely
  - make sure someone's there
  - and to make sure the *originator*'s there

Connection Setup States

[from RFC-793]

passive vs. active (and both active!)

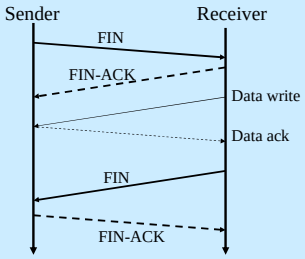
(error recovery is not on this figure)



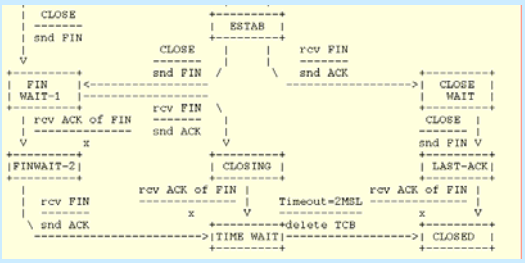
Initial Sequence Number Selection

- Why not just start at 0?
  - xxx
- Approach:
  - xxx

Tear-down Packet Exchange



Connection Tear-down



Connection Tear-down

- either side can close
  - or one side can close and the other stay open
- one side (active side) must maintain state (TIME_WAIT) for 2 minutes, why?
  - need to make sure any segments floating around the net can be disposed of

Agenda

- connection setup and teardown
- **flow control**
  - **setting window sizes**
  - **Nagle's algorithm**
  - **silly window syndrome**
  - **protection against wrap-around**
- congestion control practice and theory
- loss recovery
- security
- performance

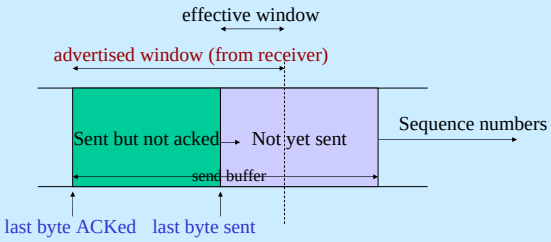
Flow Control

- Window sizes are passed in every packet
  - beware: implementations often have separate TCP and socket buffers
  - effective window is the *minimum* of the two

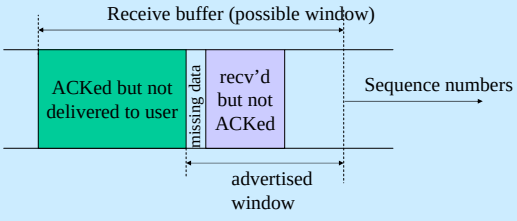
Flow Control

- Why?
  - make sure the receiver can handle whatever data they get
- Solutions
  - sliding window
  - (vs. stop-and-wait)
  - need to keep multiple packets in flight
    - routers are store and forward
    - need to allow for retx
    - need to keep the "pipe full"---a bandwidth-delay-product's worth of packets in flight

Window Flow Control: Sender



Window Flow Control: Receiver



Sending Small Things

- Silly window syndrome (RFC-813)
  - receiver dribbles out small window advances

  ⇒ Silly Window Avoidance: delay ACKing (receiver) or sending small segments (sender)
- Sender who dribbles out data (like telnet)

  ⇒ Nagle's algorithm (RFC-896): send 1st partial packet, but not more until it's ACKed or you have a full packet

Problem: Rapid Wrap-Around

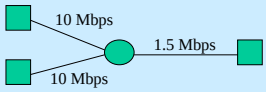
- Wraparound time vs. Link speed:
  - 1.5Mbps: 6.4 hours
  - 10Mbps: 57 minutes
  - 45Mbps: 13 minutes
  - 100Mbps: 6 minutes
  - 622Mbps: 55 seconds
  - 1.2Gbps: 28 seconds

⇒ Protection Against Wrapped Sequences (PAWS extension): Use *timestamp* to distinguish sequence number wraparound

Agenda

- connection setup and teardown
- flow control
- **congestion control theory**
  - **what and why**
  - **how**
- congestion control practice
- loss recovery
- security
- performance

Congestion Collapse



- If both sources send full speed, the router is completely overwhelmed

  ⇒ *congestion collapse:* senders lose data from congestion and they resend, causing *more* congestion (can be self-reinforcing)

  – has been observed many times

Congestion Control vs. Flow Control

- What does flow control do?
  - dest buffer capacity
- What does congestion control do?
  - router buffer (network) capacity
- What mechanism do they use?
  - sliding window (wnd in TCP header)
  - AIMD (wnd in TCP header; cwnd variable at src/dest)
    - also explicit congestion notification (ECN) in the IP header

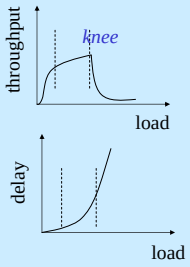
Congestion Control Signals in the Internet

- ECN: bit set by routers if they have congetion
  - fairly new
  - (how widely used?)
- drop packets
  - due to buffer overflows
  - also early packet discard (see RED)
  - (much rarer: link failures or packet corruption)

Congestion Control Goals

- to recover after congestion
- not to make congestion worse
  - i.e., if you're close to full utilization, don't increase
- if you have spare capacity, use it
- (also we forgot fairness)

Power and Load



- throughput and delay change due to load
  - want to optimize *power*

$$power := \frac{(throughput)^{\alpha}}{delay}$$

(From [Ramakrishnan90a])

Fairness

- Also want *fairness*
  - should treat all users equally
- but it's not so easy
  - what is a user? host, flow, person?
  - ⇒ if n flows through a link, each should get n^{-1} of the bandwidth
  - R&J's fairness index: $(\Sigma x_i)^2/n(\Sigma x_i^2)$
  - but what if flows have different needs? different RTTs?

Congestion Control Design

- Avoidance or control? (R&J:)
  - avoidance keeps system at knee of curve
    - requires some congestion signal
  - control responds to loss after the fact
- TCP
  - Which is TCP?
    - both, congestion avoidance and control, but they mean different things
      - be careful that the TCP terms don't mean the same as the R&J terms
  - How does TCP do it?
    - slow start, AIMD… will talk more…

How to Adjust Window?

- When to increase/decrease?
- A *control theory* problem
  - observe network
  - reduce window if congested
  - increase window if not congested
- Constraints:
  - efficency
  - fairness
  - stability (too much oscillation is bad)
  - *out-of-date info*
    - RTT is fundamental limit to how quickly you can react

Linear Control

$$X_i(t+1) = a_i(t,f) + b_i(t,f)\, X_i(t)$$

- Formulation allows for the feedback signal:
  - to change additively: $a_i(t)$
  - to change multiplicatively: $b_i(t)$
  - can consider feedback: *f*
  - all to compute beavhior at time *t+1* based on info at time *t*
- What does TCP do and why?
  - start up: slow start: start by sending a small amount, then increase multipliciatively to get to "steady state"
  - at steady state: increase linearly; TCP calls this "congestion avoidance"
  - after congestion: multiplicative decrease
- Types of feedback in Internet?
  - drop packets: lack of ACKs or ACKs that don't advance
  - or ECN

Agenda

- connection setup and teardown
- flow control
- congestion control theory
- **congestion control practice (in TCP)**
- **loss recovery**
- **security**
- **performance**

=> next sets of slides