Congestion Control
TCP is the dominant transport protocol in today’s Internet
- Web page loads, BitTorrent transfers, some video streaming, some of Skype

Embodies some of Internet design principles
- packet switching (i.e., no state on switches)
- smart host, dumb network
TCP Mechanisms

- Flow control: prevent sender from overwhelming the receiver
- Reliable delivery: lost packets are retransmitted
- Congestion control: react to network congestion
Congestion Control

- Best effort delivery on the Internet
  - Let everybody send, try to deliver what you can, and drop the rest
  - If many packets arrive in a short period of time, router buffers fill up and packets are lost
  - Loss indicates congestion; so does increased delay
What should be the goals of congestion control? (Or what is an ideal congestion control protocol?)
Observations

- Congestion is inevitable, and arguably desirable
- If packets are dropped, then retransmissions can make congestion even worse
- When packets are dropped, they waste resources
Original TCP Design

Van Jacobson
- Formerly at LBL
- Internet pioneer
- Now at PARC
- Inventor tcpdump, traceroute

Michael J. Karels
- Very involved in BSD development
- Replaced Bill Joy as developer

Cited more than 6,000 times.
Context

TCP didn’t work well under congestion (circa 1988)

Congestion collapse:

- Breakdowns in performance noted in 1986 on NSFNet.
- 40Kb/s links operating as slow as 32b/s.
- NSFNet was a forerunner of today’s Internet backbone (from 1986 to 1995).
Main contributions

Seven new algorithms:

1. RTT Variance estimation
2. Exponential retransmit timer backoff
3. Slow-start
4. More aggressive receiver ack policy (delayed ACK or not)
5. Dynamic window sizing on congestion
6. Karn’s algorithm (accurate RTT)
7. Fast retransmit
Packet Conservation

‘Conservation of packets’ principle:

For a connection ‘in equilibrium’, i.e., running stably with a full window of data in transit...

A new packet shouldn’t be put into the network until an old packet leaves.
TCP Animation

- http://guido.appenzeller.net/anims/
Slow-start + AIMD (Tahoe)

Window size = min(advertised window, $cwnd$)

Packet Loss

Slow start in operation until it reaches half of previous $cwnd$.

Exponential “slow start”
TCP Reno

- Enhancements include:
  - early detection of packet loss using 3-dupack (where acks are cumulative ACKs)
  - fast retransmit of such packets
  - fast recovery of congestion window
Slow-start + AIMD (Reno)

Window size = \( \min(\text{advertized window, } cwnd) \)

Packet Loss (3-dupack)

Packet Loss (timeout)

Slow start in operation until it reaches half of previous \( cwnd \).

Exponential “slow start”
Getting to equilibrium

Slow-start

Q: What is slow-start trying to accomplish?

Q: How long does it take slow-start to reach equilibrium?

Q: How does AIMD compare to AIAD, MIMD, and MIAD?
RTT Variation Estimate

Q: Why is it important to estimate RTT well?

Q: How can you improve RTT estimation?

Q: Can we do better than cumulative ACKs?
TCP Challenges

- TCP early designs were in 80s and 90s
- What are the new challenges for TCP in today’s world?
TCP Congestion Control

- Allocate resources without requiring network support
- “Try and Backoff” strategy:
  - Start with low transfer rate, ramp up rate
  - FIFO routers drop packets when queues fill up
  - Congestion inferred from packet loss
  - Endpoint responds to packet loss by throttling rate
**Limits of Try-and-Backoff**

- In theory, the link capacity is fully utilized for long flows, but
  - Initial ramp-up takes up most of the response time
  - Channel capacity is left unused
    - If “n” is capacity, takes log(n) steps for the initial ramp-up
    - Wasted capacity during that period: $O(n \log(n))$
  - At the tail of the ramp-up, the rate overshoots the channel capacity
- Could start with higher transfer rates, but could result in higher packet loss/congestion
Network-assisted Congestion Control

1) Routers provide feedback to end-systems
2) Routers explicitly allocate bandwidth to flows

Problem: makes routers complicated and hinders adoption
Feedback Signals

- Delay and router signals can let us avoid congestion

<table>
<thead>
<tr>
<th>Signal</th>
<th>Example Protocol</th>
<th>Pros / Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet loss</td>
<td>Classic TCP</td>
<td>Hard to get wrong</td>
</tr>
<tr>
<td></td>
<td>Cubic TCP (Linux)</td>
<td>Hear about congestion late</td>
</tr>
<tr>
<td>Packet delay</td>
<td>Compound TCP (Windows)</td>
<td>Hear about congestion early</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Need to infer congestion</td>
</tr>
<tr>
<td>Router indication</td>
<td>TCPs with Explicit Congestion Notification</td>
<td>Hear about congestion early</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Require router support</td>
</tr>
</tbody>
</table>
ECN (Explicit Congestion Notification)

- Router detects the onset of congestion via its queue
- When congested, it marks affected packets (IP header)
ECN (2)

- Marked packets arrive at receiver; treated as loss
- TCP receiver reliably informs TCP sender of the congestion
Advantages:
- Routers deliver clear signal to hosts
- Congestion is detected early, no loss
- No extra packets need to be sent

Disadvantages:
- Routers and hosts must be upgraded