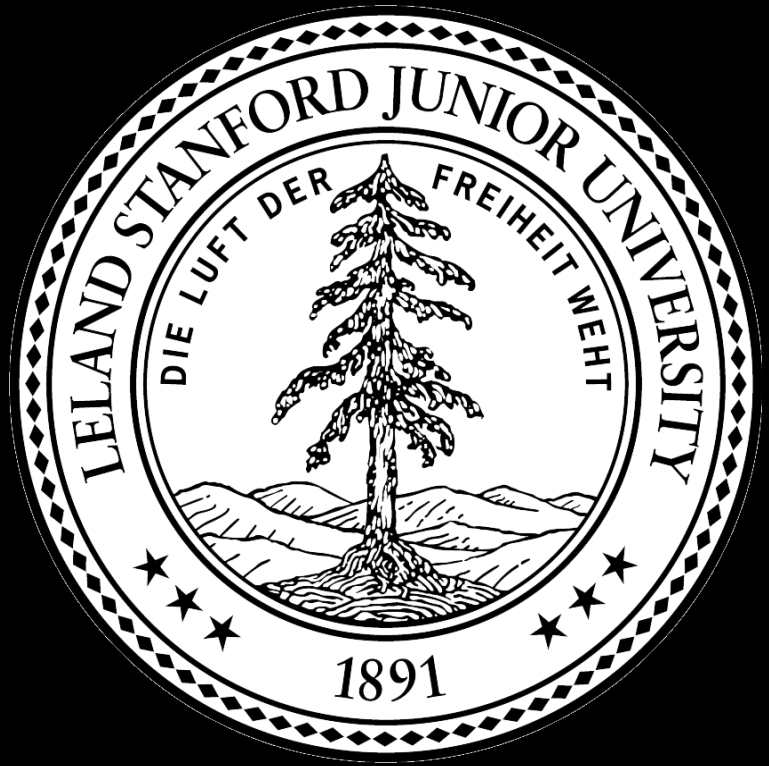
Welcome to CS244 Spring 2020!

Class will start shortly



CS244

Advanced Topics in Networking

Lecture 4: Congestion Control

Nick McKeown

"Congestion Control and Avoidance"
[Van Jacobson and M. Karels, 1988]

Spring 2020

Context

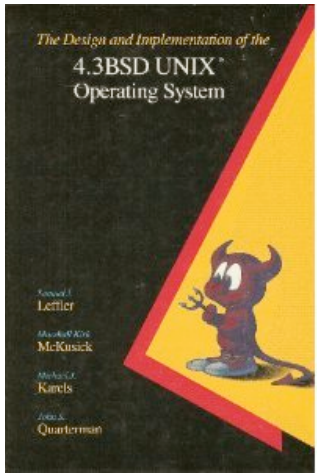
Van Jacobson

- Formerly at LBL
- Internet pioneer
- Now at Google Cloud
- Inventor tcpdump, traceroute, BBR

Michael J. Karels

- Very involved in BSD development
- Replaced Bill Joy as developer

Paper cited over 8,500 times.



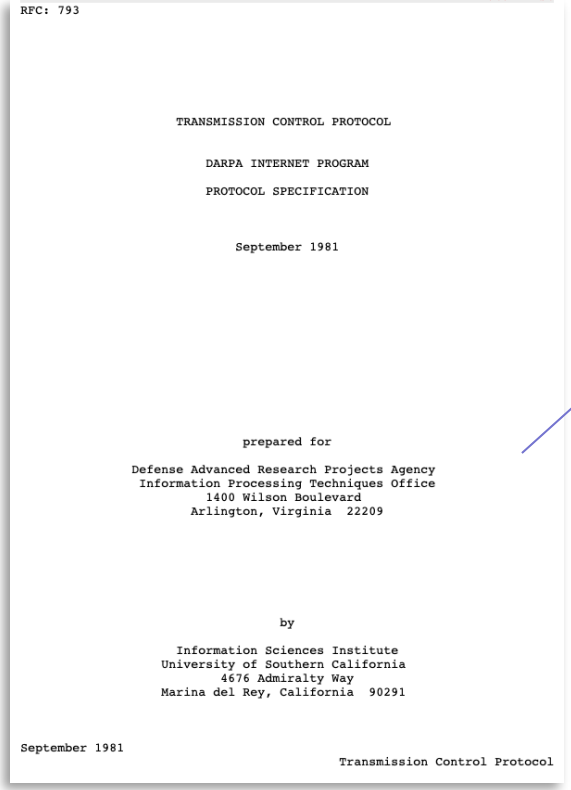
The Problem

Congestion collapse:

- Breakdowns in performance noted in 1986 on NSFNet.
- NSFNet was forerunner of today's Internet backbone (1986 to 1995). NSF = National Science Foundation.
- 40Kb/s links operating as slow as 32b/s.

Q: How did TCP control congestion before 1988?

How did TCP work in 1988?



RFC: 793

TRANSMISSION CONTROL PROTOCOL

DARPA INTERNET PROGRAM

PROTOCOL SPECIFICATION

September 1981

prepared for

Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

by

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, California 90291

September 1981

Transmission Control Protocol

1.5. Operation

As noted above, the primary purpose of the TCP is to provide reliable, securable logical circuit or connection service between pairs of processes. To provide this service on top of a less reliable internet communication system requires facilities in the following areas:

- Basic Data Transfer
- Reliability
- Flow Control
- Multiplexing
- Connections
- Precedence and Security

Q: Why did the old TCP lead to congestion?

Q: What do we *really* mean by Internet "congestion"?

https://tools.ietf.org/html/rfc793

How I think about it

- The path of a TCP connection has “space” for a certain number of bits on the wires and in the buffers.
- I think of it like a paper bag that has room for our bits. The bag changes size depending on other traffic.
- Add too many bits, and our bag overflows.
- Add too few, and our flow runs slower than it could.
- TCP tries to keep the bag just full.
- It is less about congestion control than resource allocation: *How to wisely allocate space to all connections*.
- “Packet conservation” is key to this idea…

Packet Conservation

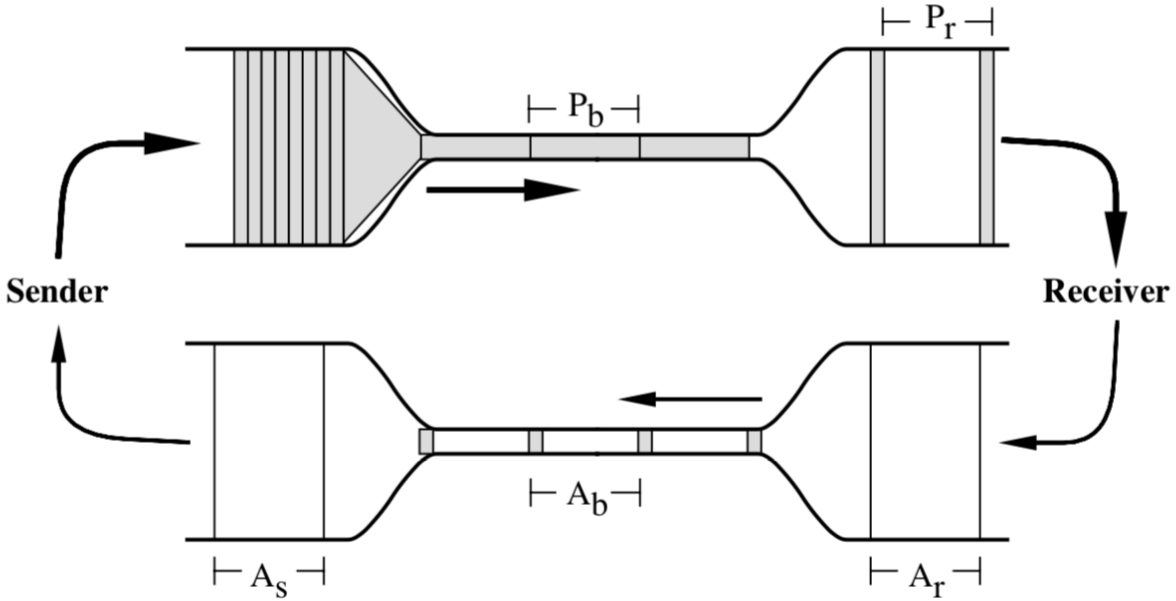
Packet Conservation principle:

- 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.

Q: How does TCP accomplish this?

Self-Clocking

Figure 1: Window Flow Control 'Self-clocking'



Hence, once a flow reaches its equilibrium rate, new packets are naturally sent

1. When a packet leaves, and
2. At the bottleneck rate.

"So, if packets after the first burst are sent only in response to an ack, the sender's packet spacing will exactly match the packet time on the slowest link in the path."

Key observation: $A_s = A_b = A_r = P_r = P_b$

A fun video of how packets flow in a data center network.
I am showing this to give you a feeling for the difference between the propagation delay and the packetization (or serialization) delay.
Watch the whole thing, or just starting around 2:00mins

https://www.youtube.com/watch?v=BO0QhaxBRr0

A subtlety about stability

[1] A conservative flow means that for any given time, the integral of the packet density around the sender–receiver–sender loop is a constant. Since packets have to 'diffuse' around this loop, the integral is sufficiently continuous to be a Lyapunov function for the system. A constant function trivially meets the conditions for Lyapunov stability so the system is stable and any superposition of such systems is stable. (See [3], chap. 11–12 or [21], chap. 9 for excellent introductions to system stability theory.)

Q: Why isn't packet conservation enough on its own?

1. A flow needs to reach equilibrium in the first place. Hence, Slow-Start.
2. As the "bag" changes size, we need to adapt the number of outstanding bits in the bag accordingly. Hence, AIMD.

TCP already had a sliding window mechanism in place,
so they overloaded it with AIMD and slow-start for congestion avoidance.

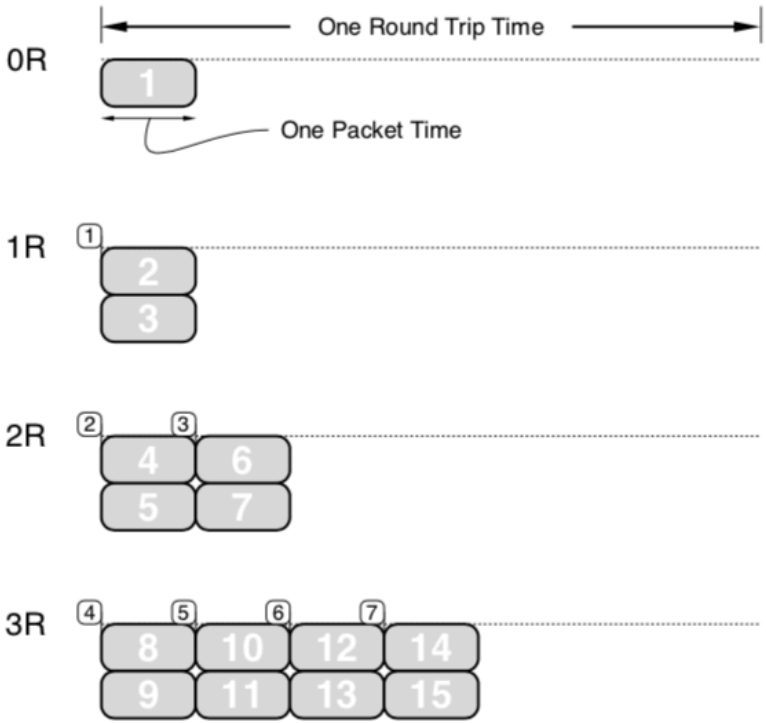
Packet conservation

"There are only three ways for packet conservation to fail:

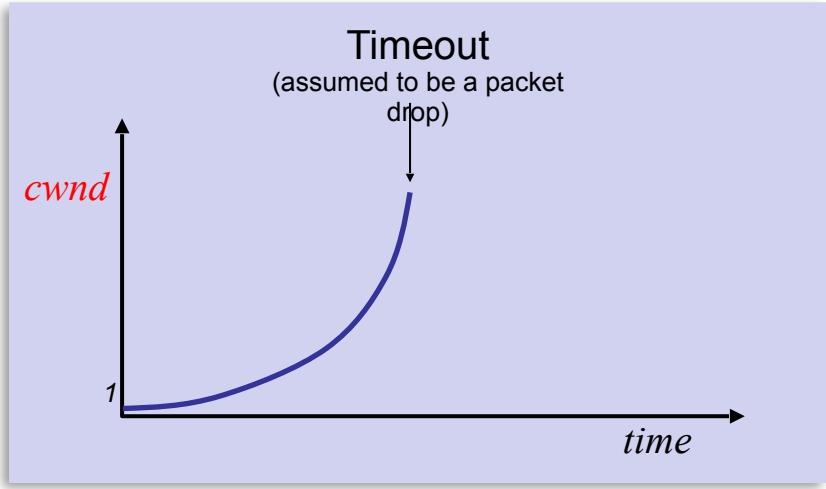
1. The connection doesn't get to equilibrium, or
2. A sender injects a new packet before an old packet has exited, or
3. The equilibrium can't be reached because of resource limits along the path."

Getting started: Slow-start

Figure 2: The Chronology of a Slow-start



Window size = min(*rwnd*, *cwnd*)



Q: How long does it take to reach a window of size W?

$$RTT * \log_2 W$$

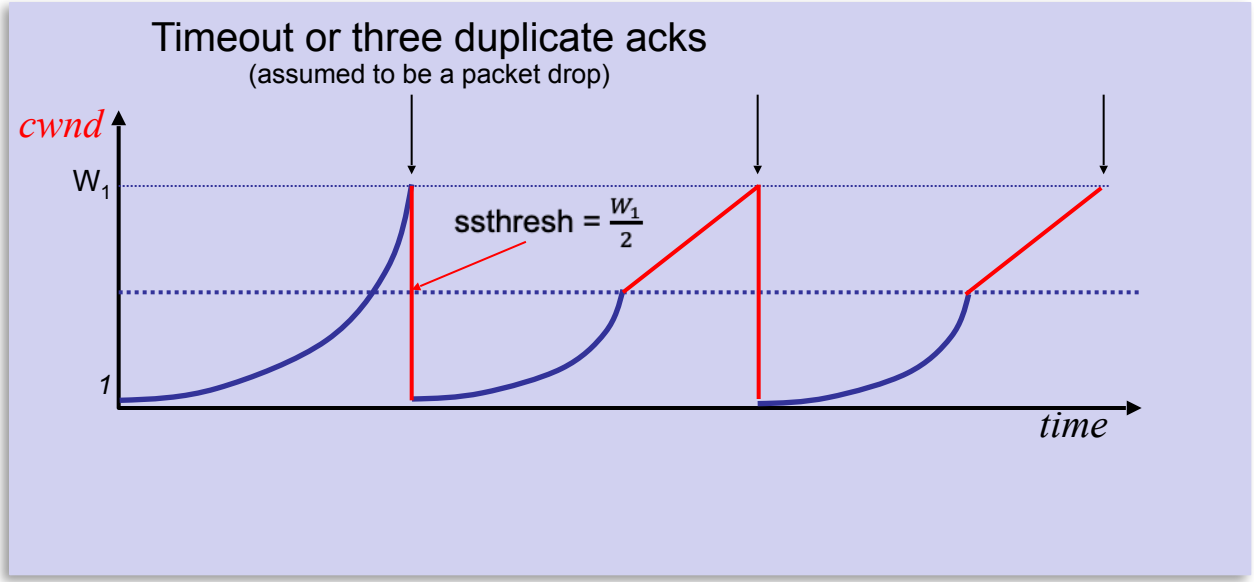
You said

Peter McEvoy:

I know that the paper mentioned that John Nagle coined the term slow-start in a message to the IETF mailing list in 1987, but I'm wondering why the name slow-start stuck? The window increase doesn't seem slow at all.

Entering AIMD “Congestion Avoidance”

Window size = min(*rwnd*, *cwnd*)




```
if (cwnd < ssthresh)
        /* if we're still doing slow-start
         * open window exponentially */
        cwnd += 1;
else
        /* otherwise do Congestion
         * Avoidance increment-by-1 */
        cwnd += 1/cwnd;
```

Called “TCP Tahoe”

Q: What happens if the timeout happens at a different window size?

You Said

Amalee Dianne Wilson:

…on page 9, "There is reason to believe on might eventually need a three term, second order model, but not until the internet has grown substantially." Over 30 years later, the internet has certainly grown substantially. Was a three term, second order model ever used?

TCP Reno: A slight enhancement

Original: TCP Tahoe

If three duplicate ACKs

1. Retransmit
2. Set ssthresh = cwnd/2
3. Set cwnd =1
4. Re-enter slow-start.

Tweak: TCP Reno

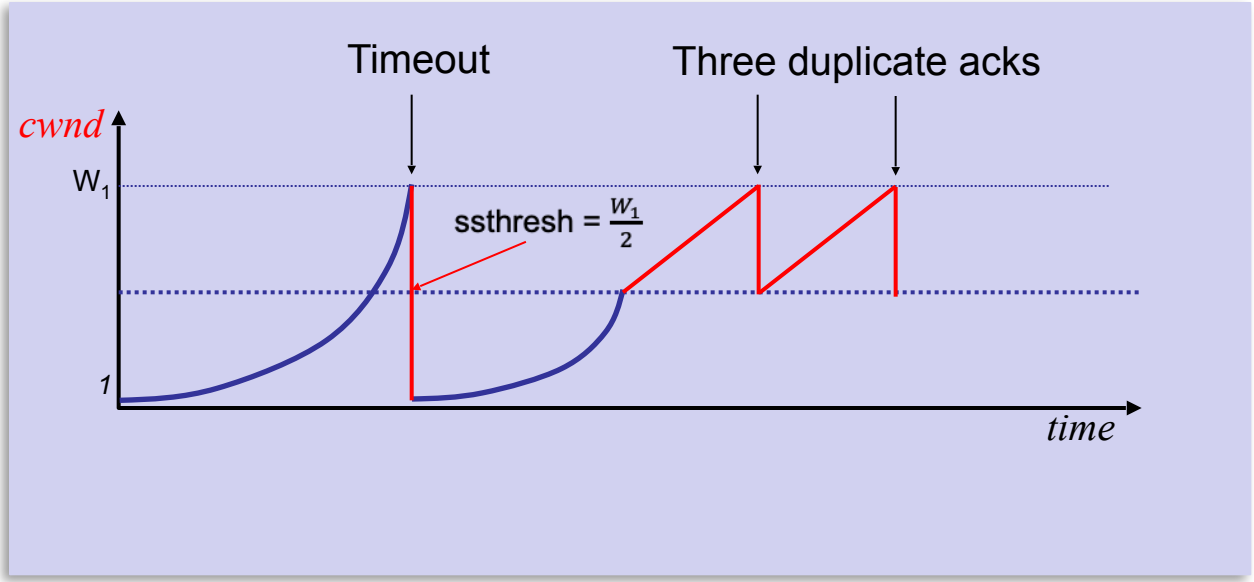
If three duplicate ACKs

1. Retransmit
2. Set ssthresh = cwnd/2
3. Do not re-enter slow-start.

```
if (cwnd < ssthresh)
        /* if we're still doing slow-start
         * open window exponentially */
        cwnd += 1;
else
        /* otherwise do Congestion
         * Avoidance increment-by-1 */
        cwnd += 1/cwnd;
```

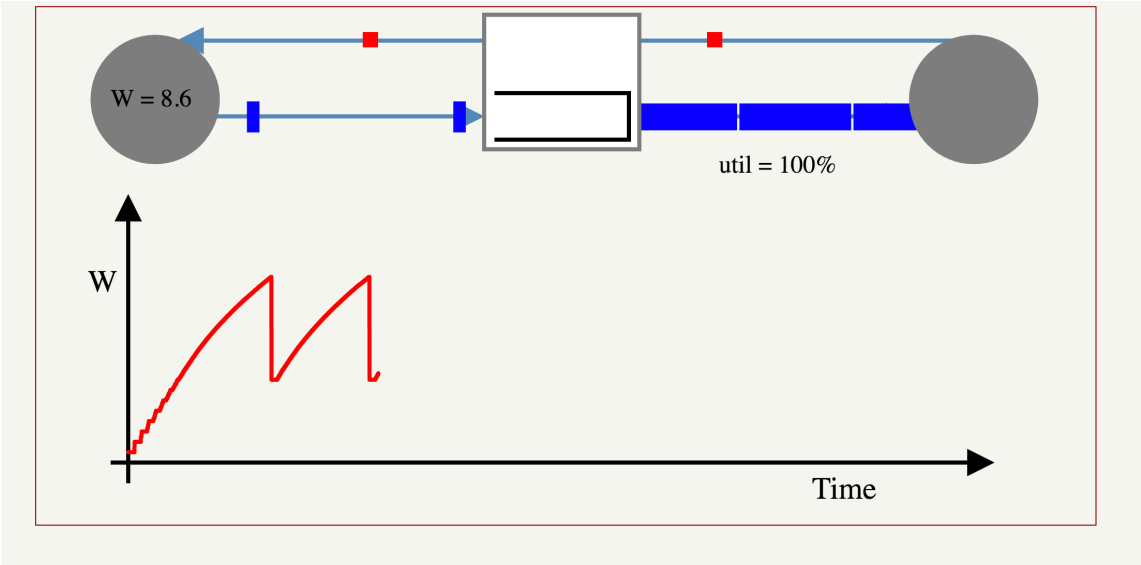
TCP Reno: A slight enhancement

Window size = min(*rwnd*, *cwnd*)




```
if (cwnd < ssthresh)
        /* if we're still doing slow-start
         * open window exponentially */
        cwnd += 1;
else
        /* otherwise do Congestion
         * Avoidance increment-by-1 */
        cwnd += 1/cwnd;
```

Animations



To learn more…

If you didn't take CS144 at Stanford (and even if you did!), consider watching:

1. Video 4-3 "Dynamics of a Single AIMD Flow"
2. Video 4-4 "Multiple AIMD Flows"

Packet Conservation led to New Algorithms

1. RTT variance estimation
2. Exponential retransmit timer backoff
3. Slow-start
4. More aggressive receiver ack policy
5. Dynamic window sizing on congestion
6. Karn’s algorithm
7. Fast retransmit

You Said

Angela Montemayor:

In the future work section, they discuss the creation of the gateway "congestion detection" algorithm" as the "next big step" for the study of congestion control. Has this algorithm been solidified, and how long did it take to establish it? What is the algorithm?

TCP properties

Q: What is TCP congestion control trying to accomplish?

What are its goal for:

- Long-lived flows
- Short-lived flows
- The network operator
- The end user

Q: How well does it accomplish these goals?

Meta Comments

Style

- Q: What do you think of the style of the paper
- Q: It is rigorous or intuitive?

Involving the “Gateway”

Q: How do they propose the router (“gateway”) gets involved in identifying congestion early?

Q: Why might an early detection be helpful?

Q: What methods have since been proposed and tried?

End.