# Congestion control

# Overview

Goals: efficiency and fairness.

Try sending packets.

- If packet dropped, decrease sending rate.
- If not, increase sending rate.

Control rate using congestion window.

- cwnd: number of un-acknowledged packets allowed

# AIMD

Additive increase, multiplicative decrease

| | |
|---|---|
| No loss | cwnd = cwnd + 1 |
| Loss | cwnd = cwnd / 2 |

# Additional features

Slow-start

start with 1 packet and exponentially increase (not really “slow”)

Fast retransmit

retransmit packet after 3 dup acks

Fast recovery

cut congestion window in half after fast retransmit

(rather than returning to slow start)

# DCTCP

# Traffic in data centers

Delay-sensitive short flows

- Contribute to incast: synchronized short flows collide.

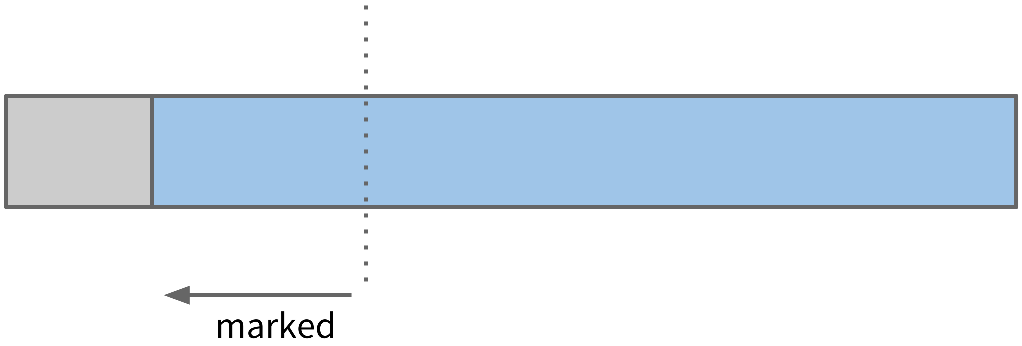
Throughput-sensitive long flows

- Contribute to queue buildup: short flows see high latency.

Applications will time out before operations complete!

# ECN

Packet marked if router queue length > threshold.




Sender sees marks in ACKs and adjusts cwnd.

# DCTCP algorithm

a = current estimate of fraction of marked packets

g = “estimation gain” - tuned parameter (section 3.4)

F = fraction of received ACKs that were marked

In each RTT:

$$a = (1 - g) * a + g * F$$

$$cwnd = (1 - a/2) * cwnd$$

# Bufferbloat

# Overview

Large buffers to prevent packet losses...

but TCP uses packet losses to detect congestion

Results?

- senders do not reduce sending
- potential for long queues to build up
- potential for long delays

# Arguments

Gettys: large data transfers appear to result in high ping latency

Allman: bufferbloat can happen, but modest observed impact