5617, Spring 2019 computer networking and communication

anduo wang, Temple University
TTLMAN 401 A, R 17:30-20:00

to do

paper review

- Congestion Control for High Bandwidth-Delay Product Networks
  - https://conferences.sigcomm.org/sigcomm/2002/papers/xcp.pdf
- submit review online
  - https://www.dropbox.com/request/0s8Y4HAL6liZuWzHzEo6

to do

homework 3

- Due Feb 28
- Submit in class

Congestion Avoidance and Control

https://ee.lbl.gov/papers/congavoid.pdf

Cited by 8130, as of Feb 21, 2019

the congestion collapse problem

problem — congestion collapse

- data throughput drops dramatically
- make congestion collapse exception rather than the rule

the congestion collapse problem

problem — congestion collapse

- data throughput drops dramatically
- make congestion collapse exception rather than the rule

solution — flow on a TCP connection must obey a "conservation of packets" principle

- a connection in equilibrium
- running stably with a full window of data in transit

the solution(s)

three ways packets conservation can fail

- connection not get to equilibrium

- sender injects a new packet before an old packet has exited
- equilibrium can’t be reached because of resource limits along the path

get to equilibrium
by slow start

the solution(s)

three ways packets conservation can fail

- connection not get to equilibrium
- sender injects a new packet before an old packet has exited
- equilibrium can't be reached because of resource limits along the path

get to equilibrium by slow start

stay at equilibrium by a good timer

the solution(s)

three ways packets conservation can fail

- connection not get to equilibrium
- sender injects a new packet before an old packet has exited
- equilibrium can't be reached because of resource limits along the path

get to equilibrium by slow start

stay at equilibrium by a good timer

adapt to the path by congestion avoidance

slow start

why not get to the equilibrium?

- the congestion control protocol is self-chocking
- sender uses acks as a "clock" to strobe new packets into the network
- but, the receiver can generate backs no faster than data packets can get through the network

slow start

self clocked system

- stable automatically adjust to bandwidth and delay variation
- but the same thing that makes it stable when it’s running makes it hard to start

to get data flowing
there must be acks to clock out packets
but to get acks there must be data flowing

a slow start algorithm to start the “clock”

- add a congestion window, “cwnd”, to the per-connection state
- when starting or restarting after loss, set cwnd to 1 packet size
- on each ack for new data, increase cwnd by 1 packet size
- when sending, sending at the rate of min(rwnd,cwnd)

retransmission timer

want to stay at the equilibrium, but what if sender injects a new packet before an old one has exited?

retransmission timer

want to stay at the equilibrium, but what if sender injects a new packet before an old one has exited?

↓

must be a failure of sender’s
retransimmision timer

retransmission timer

want to stay at the equilibrium, but what if sender injects a new packet before an old one has exited?

↓

must be a failure of sender's retransimmision timer

↓

solution: a good RTT (round trip time) estimator

timeout?

timeout indicates packet loss

- when the retransmission timer is in good shape

packets get lost because

- they are damaged in transit — rare (<<1%)
- network is congested
  - somewhere on the path there was insufficient buffer capacity

timeout?

timeout indicates packet loss

- when the retransmission timer is in good shape

packets get lost because

- they are damaged in transit — rare (<<1%)
- network is congested
  - somewhere on the path there was insufficient buffer capacity

solution — adapting to the path
by congestion avoidance

congestion avoidance strategy

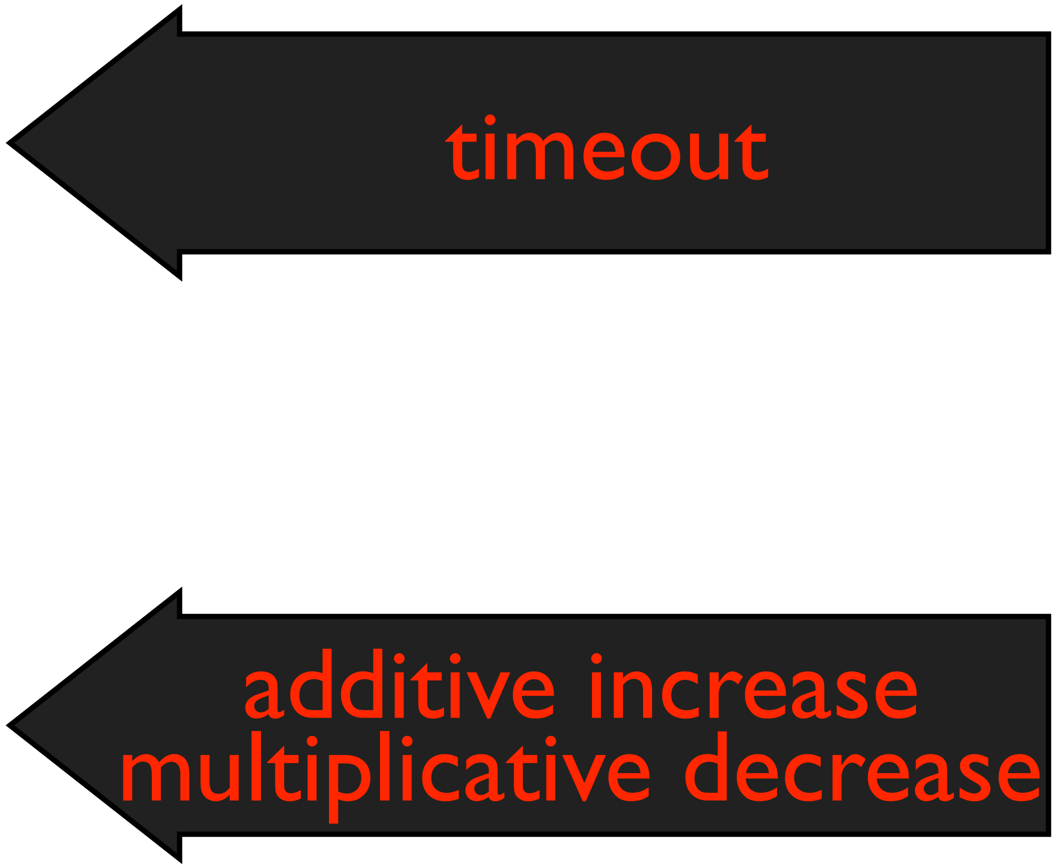
- the network must be able to signal the transport endpoints that congestion is occurring (or about to occur)

- the endpoints have a policy that decreases utilization if this signal is received and increase utilization if the signal is not received

congestion avoidance strategy

- the network must be able to signal the transport endpoints that congestion is occurring (or about to occur)

- the endpoints have a policy that decreases utilization if this signal is received and increase utilization if the signal is not received



timeout

additive increase
multiplicative decrease

decrease policy on congestion

adjust sender window size, W, on congestion

$$W_i = d\,W_{i-1}\ (d<1)$$

no congestion

the network says nothing if a connection is using less than its fair share

- a connection has to increase its utilization to find out the correct limit
- need an increase policy

increase policy on no congestion

a first attempt

- symmetric, multiplicative increase
  - oscillate wildly, poor throughput
- overestimating the available bandwidth is costly

best increase policy

- $W_i = W_{i-1} + u \ (u << W_{max})$

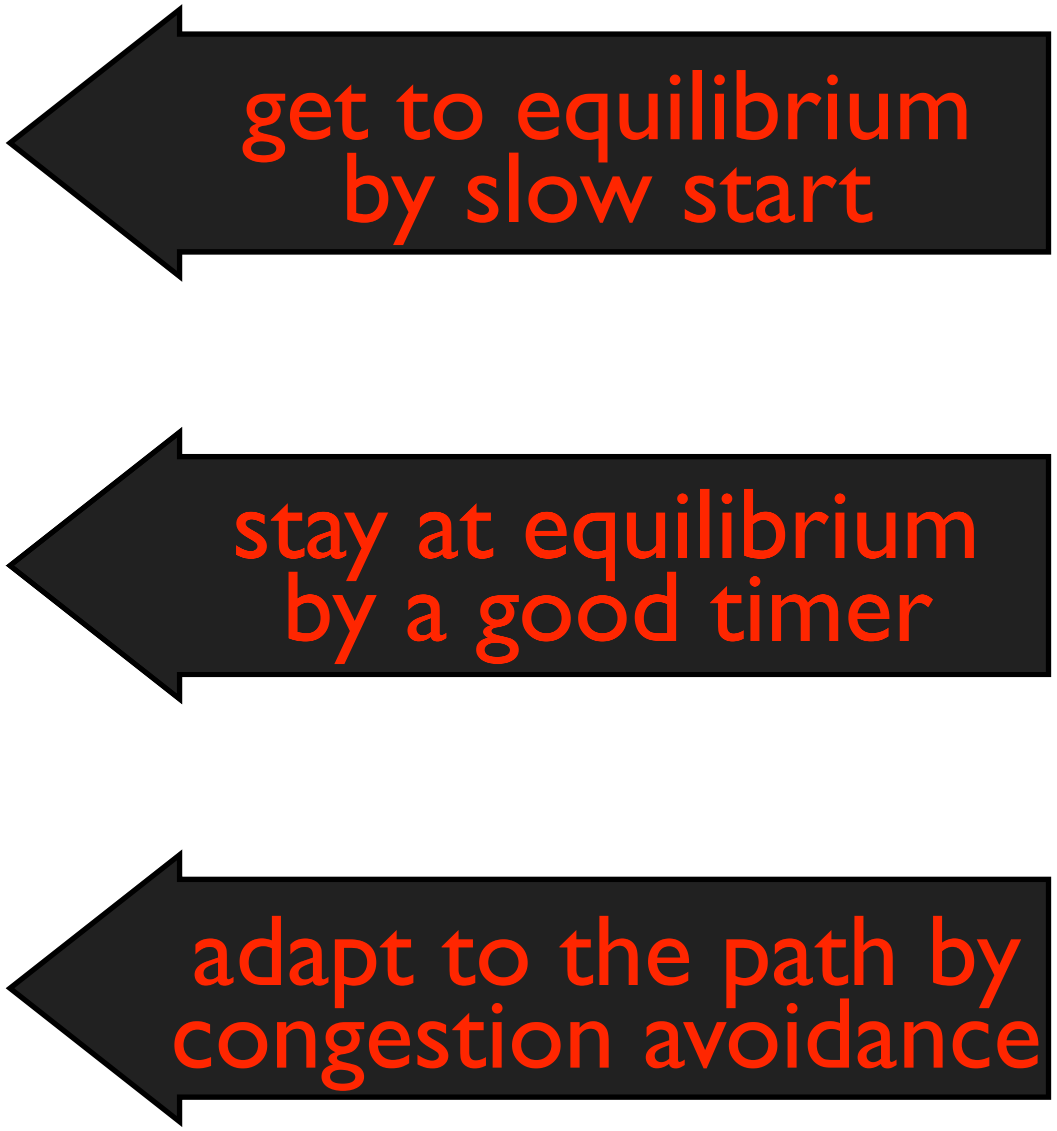
a congestion avoidance algorithm

- on any timeout, set cwnd to half the current cwnd
- on each ack for new data, increase cwnd by 1/cwnd
- when sending, sending at min(cwnd, rwnd)

recap: congestion avoidance and control

three ways packets conservation can fail

- connection not get to equilibrium
- sender injects a new packet before an old packet has exited
- equilibrium can’t be reached because of resource limits along the path



get to equilibrium by slow start

stay at equilibrium by a good timer

adapt to the path by congestion avoidance