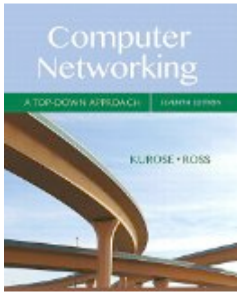
COMP 375: Lecture 24

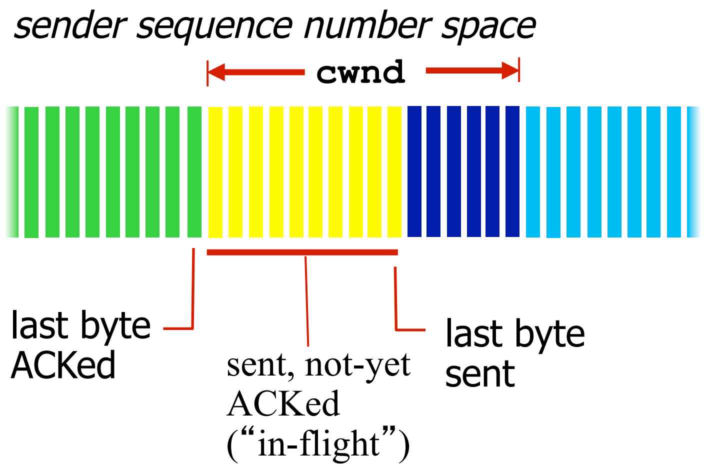


- **News & Notes:**
  - Spring Break... WOOOOO!!!
  - Special presentation at end of class
  - Quiz #6 in class Wednesday, April 4
  - Project #4 due Monday, April 16
- **Reading (Wed, April 4)**
  - Sections 4.3.{0-3} (IPv4 Packets and Addressing, excluding 4.3.3 on DHCP)
  - Note: Older edition chapters get out of sync at this point...

Sections 3.6 – 3.7

CONGESTION CONTROL

Sender's rate is a function of `cwnd` and RTT.


sender sequence number space
cwnd
last byte ACKed
sent, not-yet ACKed ("in-flight")
last byte sent

$$\boxed{\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}}$$

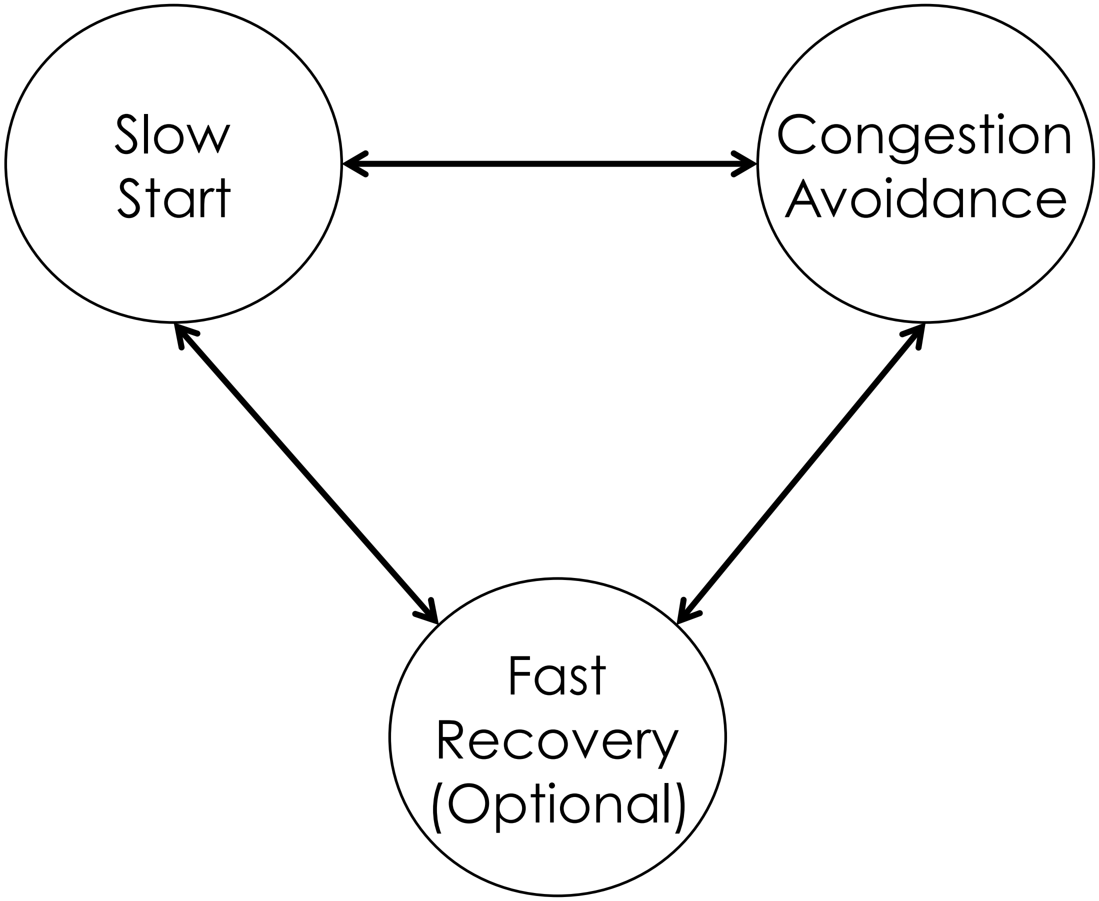
How should we set `cwnd`?

A.	We should keep raising it until a “congestion event”, then back off slightly until we notice no more events.
B.	We should raise it until a “congestion event”, then go back to 1 and start raising it again.
[✓ C]	We should raise it until a “congestion event”, then go back to a median value and start raising it again.
D.	We should send as fast as possible at all times.

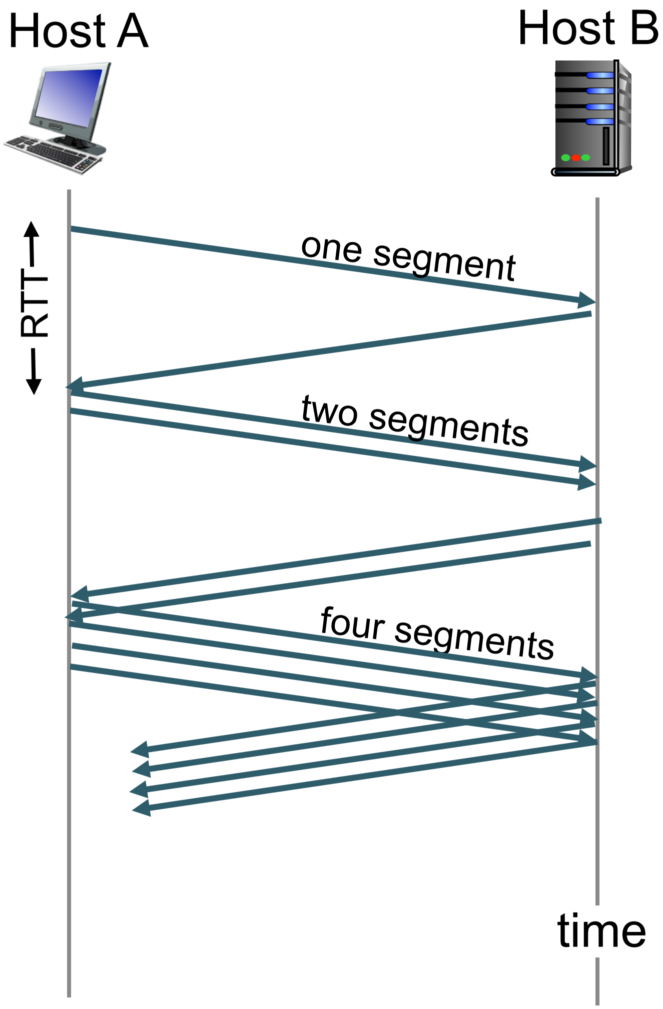
What is a “congestion event”?

A.	A segment loss
B.	Receiving duplicate ACK(s)
C.	Timeout
[✓ D.]	Exactly 2 of the above.
E.	A, B, and C

TCP goes through phases that dictate how `cwnd` changes.



In the **slow start** state, we start slow but increase rate exponentially.

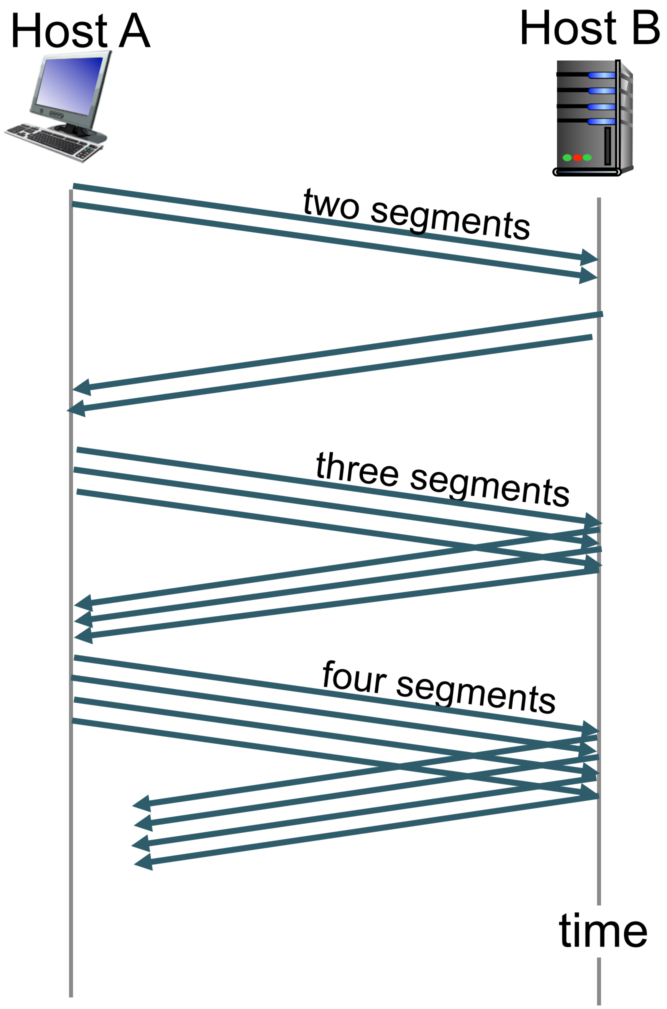


Eventually we will need to transition away from slow start.

- TCP leaves the slow start state for one of **two** reasons...

What are those reasons?

In the **congestion avoidance** state we increase rate linearly.



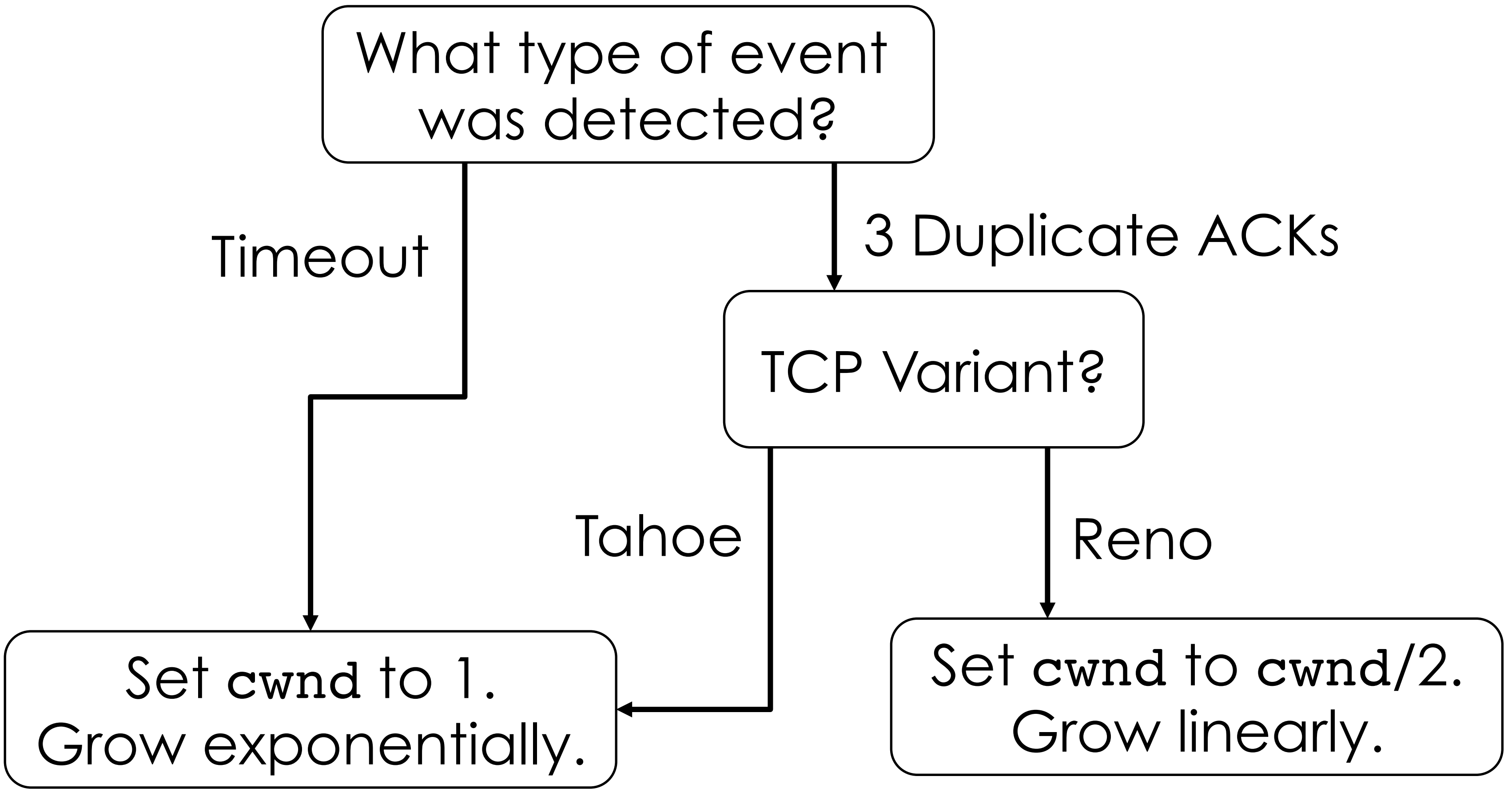
We can detect loss via three duplicate ACKs, or via a timeout.

How should we respond to these two types of loss detection?

[✓ **A.**]	Treat these events differently.
B.	Treat these events the same.

Discuss: Which, if either, of these events are "worse."

TCP variants react differently to loss.



What type of event was detected?

Timeout

3 Duplicate ACKs

TCP Variant?

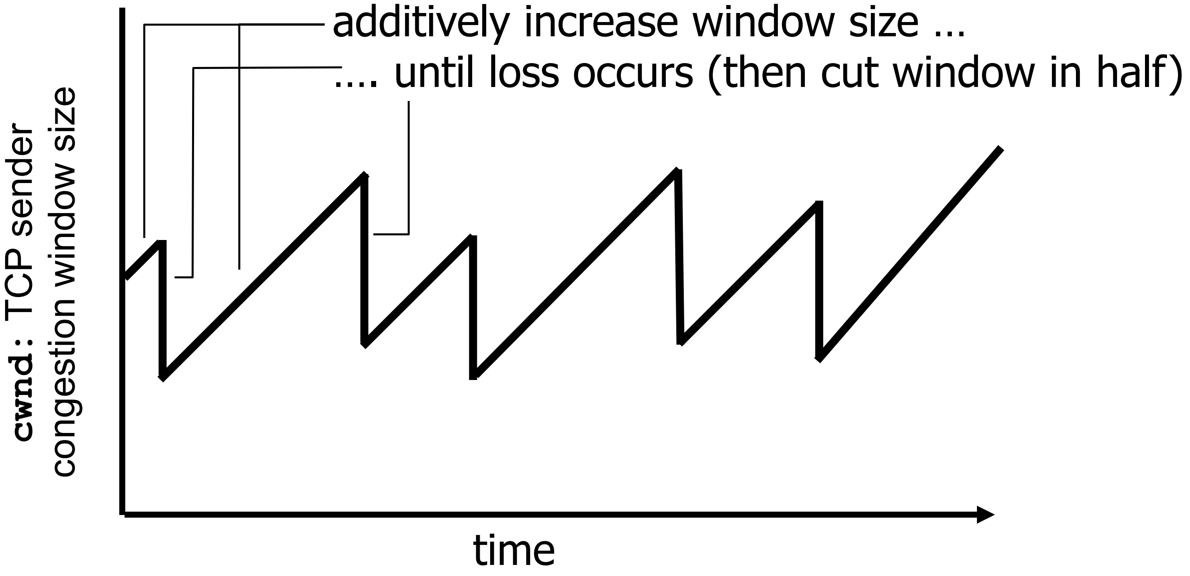
Tahoe

Reno

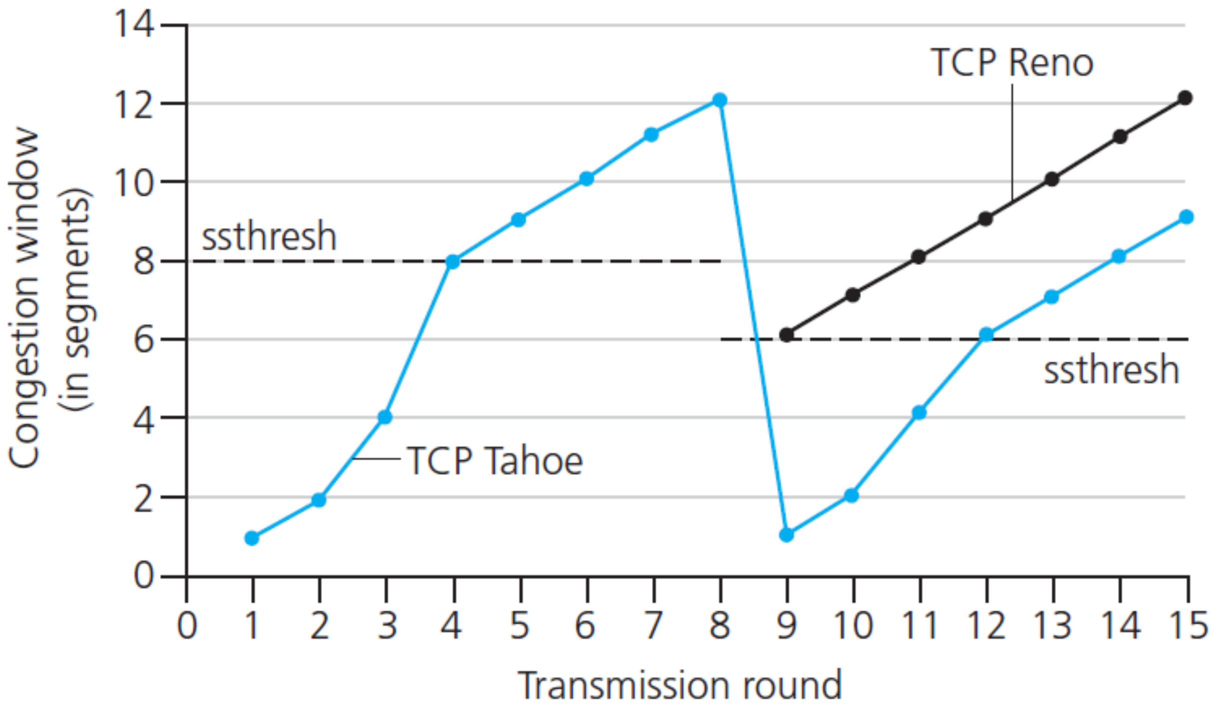
Set **cwnd** to 1.
Grow exponentially.

Set **cwnd** to **cwnd**/2.
Grow linearly.

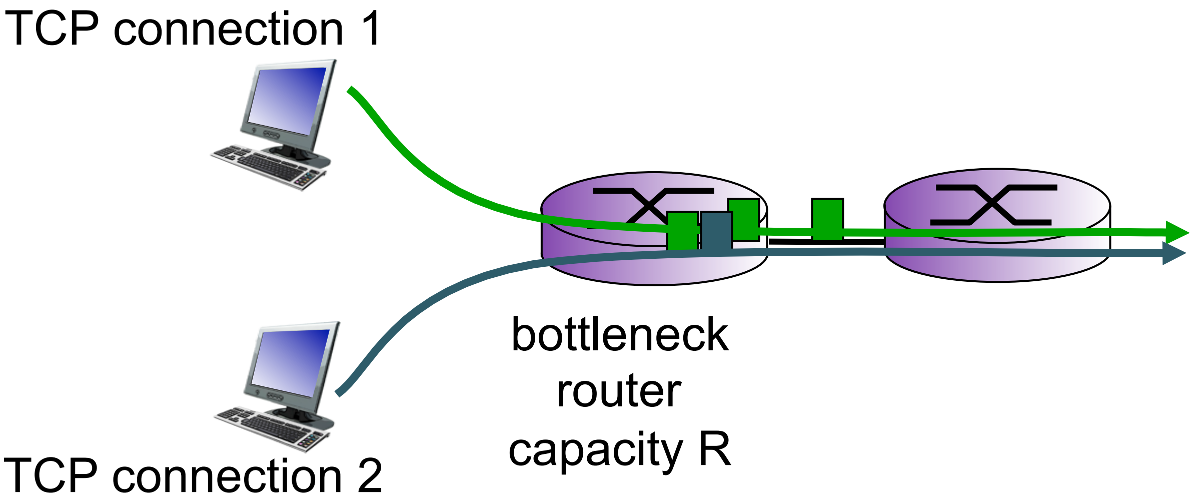
cwnd's sawtooth shape comes from AIMD (Additive Increase, Multiplicative Decrease)



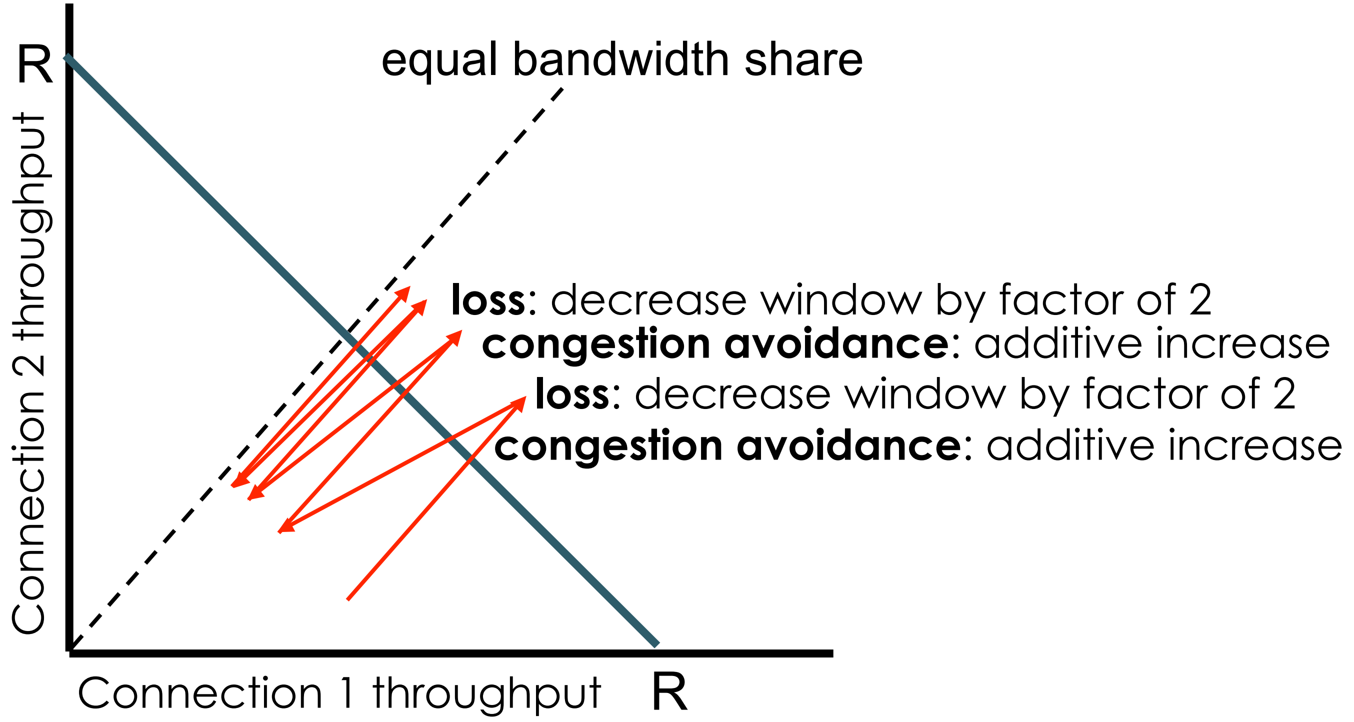
`ssthresh` dictates the transition between slow start and congestion avoidance.



Our goal is for *K* TCP connections to share bottleneck bandwidth equally (i.e. *R*/*K*).



AIMD helps ensure fairness in TCP.


R
equal bandwidth share
Connection 2 throughput
loss: decrease window by factor of 2
congestion avoidance: additive increase
loss: decrease window by factor of 2
congestion avoidance: additive increase
Connection 1 throughput
R