TCP Congestion Control

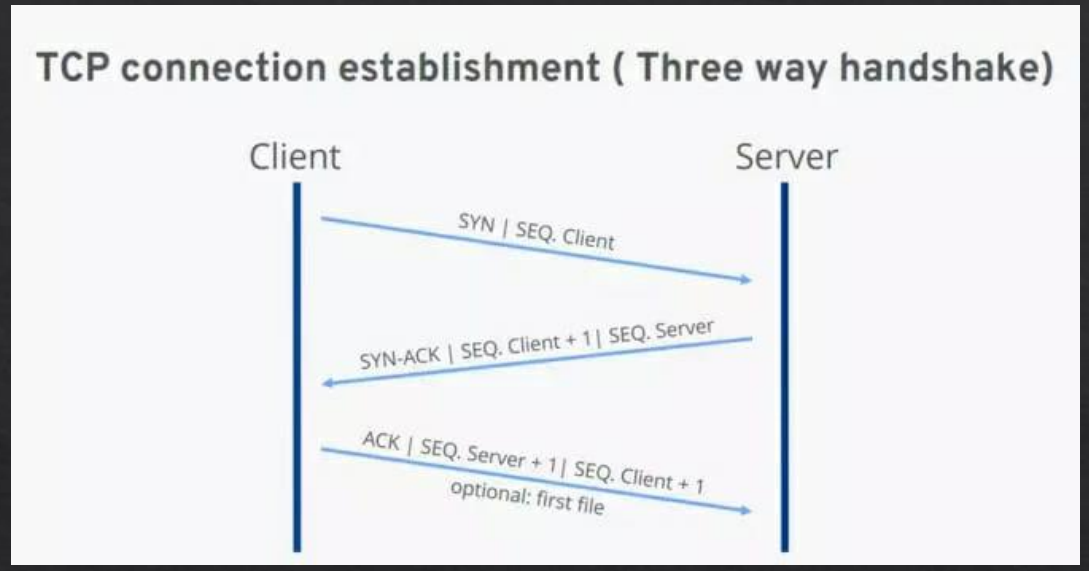
M. Umar 2021-SE-28

Abdul Haseeb 2021-SE-22

Introduction

TCP

- Widely Used Internet Protocol
- Reliable
- Transfer data in An Ordered Way



Typical TCP Connection

What Is Congestion?

- Overcrowding
- Blockage
- congestion in TCP (Transmission Control Protocol) occurs when there's too much data trying to flow through a network at the same time, causing a kind of traffic jam.

Congestion Window

- Purpose?
- imposes a constraint on the rate at which a TCP sender can send traffic into the network
- Denoted by “cwnd”

Equation:

$$LastByteSent - LastByteAcked \leq \min\{cwnd, rwnd\}$$

Where

- **LastByteSent:** the sequence number of the last byte that the sender has sent.
- **LastByteAcked:** the sequence number of the last byte that has been successfully acknowledged by the receiver.
- **cwnd (Congestion Window)**
- **rwnd (Receiver Window)**

Equation can also be Written as:

“ Unacknowledged Data ≤ min{cwnd, rwnd} ”

The equation essentially says that the difference between LastByteSent and LastByteAcked, which represents the amount of unacknowledged data in transit, should not exceed the minimum of the congestion window (cwnd) and the receiver window (rwnd).

What If
the TCP receive buffer is so large that the receive-window constraint can be ignored

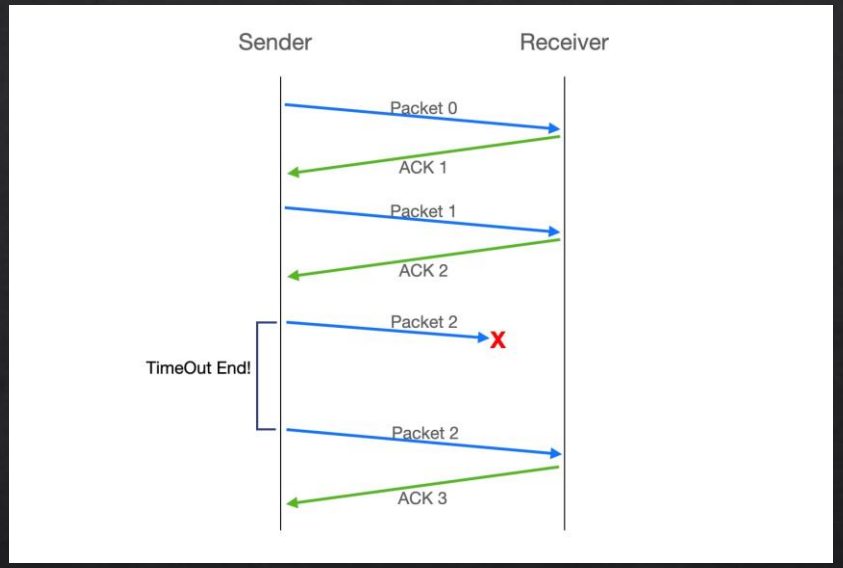
Then the equation will be:

“ Unacknowledged Data ≤ min{cwnd} ”

Therefore,
the amount of unacknowledged data at the sender is solely limited by cwnd.

How a TCP sender perceives that there is congestion on the path?

- ➢ Loss Event
- Timeout
- the receipt of four acknowledgments
  (three duplicate ACKs followed by an additional original ACK)



- ➢ Excessive Congestion
- ➢ Router Buffers
- ➢ Datagram

- ✓ Congestion-Free Network
- No Any Type Of Loss

TCP is said to be “ Self Clocking ”
Adjusts its sending rate based on the feedback it receives from the network, without relying on external timing mechanisms

How should a TCP sender determine the rate at which it should send?

If too fast,
(network will be congested)
If too slow,
(could under utilize the bandwidth in the network)

Question:

How then do the TCP senders determine their sending rates such that they don't congest the network but at the same time make use of all the available bandwidth?

Guiding Principles:

A. Lost segment implies congestion
B. An acknowledged segment
C. Bandwidth probing

❖ Lost segment implies congestion

i. Decrease congestion window size

ii. Decrease sending rate

❖ An acknowledged segment

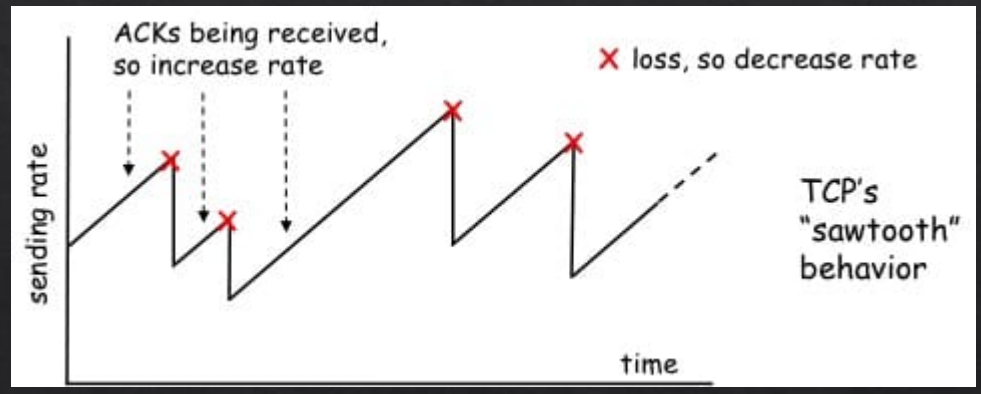
i. congestion window size can be increased

❖ Bandwidth probing

TCP congestion-control algorithm

The algorithm has three major components:

1. Slow start
2. Congestion avoidance
3. Fast recovery



Sawtooth behavior to find a balance between utilizing available bandwidth and avoiding congestion.

Slow Start

- to efficiently probe the network's capacity and avoid congestion

- cwnd is typically initialized to a small value of 1 MSS (Maximum Segment Size)
- initial sending rate of roughly MSS/ RTT.

For Example:
if MSS = 500 bytes,
RTT = 200 msec

Result:
the resulting initial sending rate is only about 20 kbps.

Slow Start State
“Since the available bandwidth to the TCP sender may be much larger than MSS/RTT, the TCP sender would like to find the amount of available bandwidth quickly. This state is known as Slow Start State.”

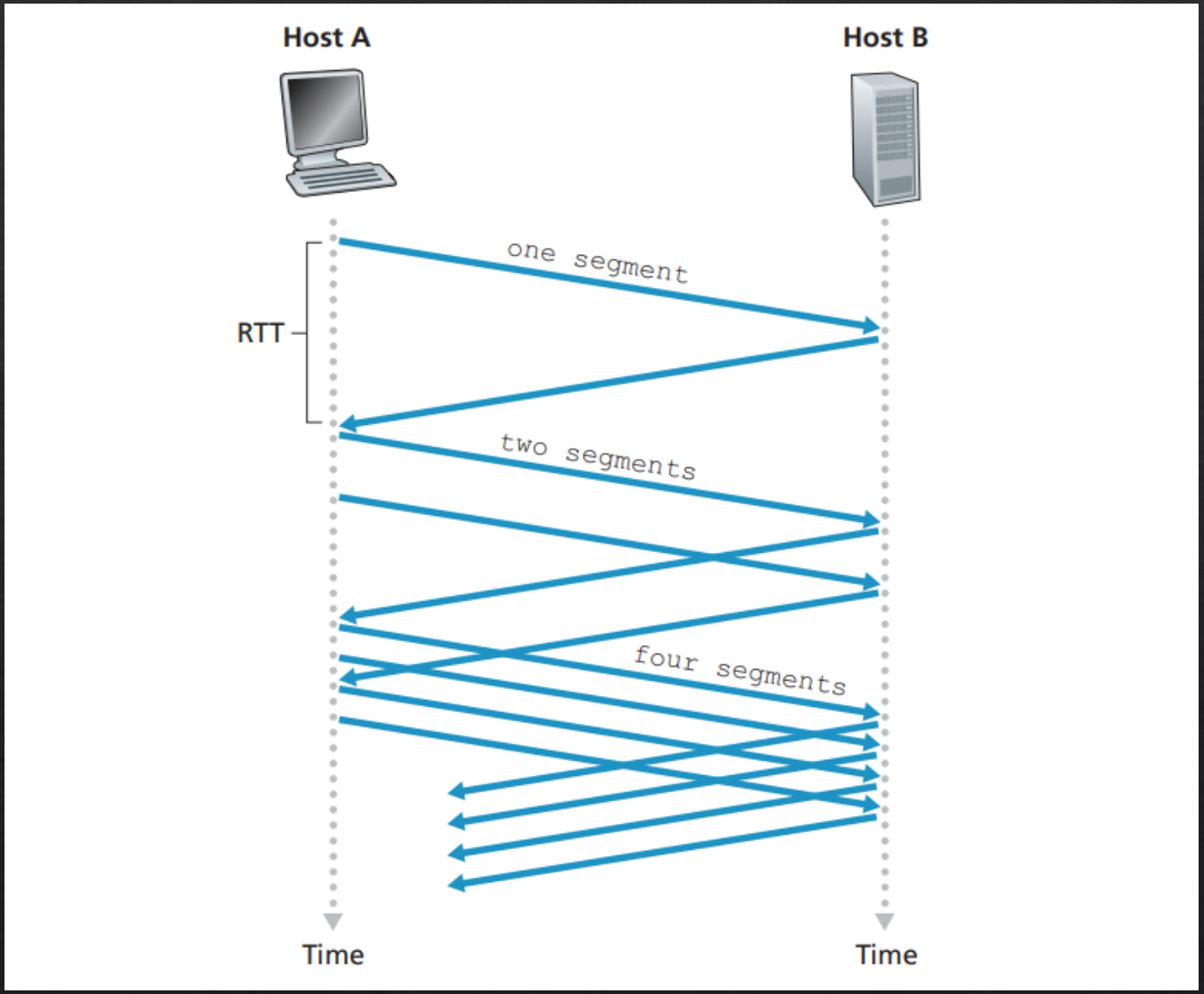
Grows Exponentially

Like;

```
Initially cwnd = 1

After 1 RTT, cwnd = 2^(1) = 2

2 RTT, cwnd = 2^(2) = 4

3 RTT, cwnd = 2^(3) = 8
```

After 3 RTT, cwnd will be 8.

Slow Start


Host A
Host B
one segment
RTT
two segments
four segments
Time
Time

But when should this exponential growth end?

i. **Loss Event(Timeout)** ssthresh is updated to half the value of cwnd when the loss event occurred.
ii. **Reaching Slow Start Threshold (*ssthresh*)**
iii. **Fast Retransmit and Three Duplicate ACKs**

Example:

Let's consider an example where TCP is in slow start and has just transmitted a segment. With each acknowledgment received, *cwnd* doubles.

- Suppose *cwnd* is currently 8 and the slow start threshold (*ssthresh*) is set to 16.
- If the next acknowledgment is received without any loss events or reaching the threshold, *cwnd* becomes 16 (8 * 2).
- If the next acknowledgment indicates a loss event (e.g., due to three duplicate ACKs), TCP performs a fast retransmit, and the congestion window may be adjusted based on the congestion control algorithm.
- If, instead, the next acknowledgment is received without loss and *cwnd* reaches or exceeds 16, slow start ends, and TCP transitions into congestion avoidance mode.

Summary:

Initial Rate is Slow but ramps up exponentially fast.

Congestion Avoidance

- follows the Slow Start phase
- Linear Increase

Objective:
"The objective of congestion avoidance is to have a more cautious increase in *cwnd* to avoid potential congestion".

Action:
"Instead of doubling *cwnd* every RTT, TCP adopts a linear increase by adding just one MSS per RTT when a new acknowledgment arrives. This ensures a more gradual and conservative growth in *cwnd*".

For Example:
if MSS is 1,460 bytes and cwnd is 14,600 bytes, then 10 segments are being sent within an RTT.
Each arriving ACK (assuming one ACK per segment) increases the congestion window size by 1/10 MSS, and thus, the value of the congestion window will have increased by one MSS after ACKs when all 10 segments have been received.

Example:

```
Initially cwnd = i

After 1 RTT, cwnd = i+1

2 RTT, cwnd = i+2

3 RTT, cwnd = i+3
```

If we assume cwnd = 1 then after 3 RTT, cwnd will be 4

Question:
when should congestion avoidance's linear increase (of 1 MSS per RTT) end?

Answer:
depend on the specific TCP congestion control algorithm in use.
However, the common condition is typically based on detecting network congestion.
Mostly Same As Slow Start:

- Timeout (ssthresh is updated to half the value of cwnd when the loss event occurred)
- Packet Loss
- Triple Duplicate ACKs

Question:
When should the exponential increase switch to linear?

Answer:
When CongWin gets to 1/2 of its value before timeout.

Summary:

- When CongWin is below Threshold, sender is in slow-start phase, window grows exponentially.
- When CongWin is above Threshold, sender is in congestion-avoidance phase, window grows linearly.
- When a triple duplicate ACK occurs, Threshold set to CongWin/2 and Congwin set to Threshold.
- When timeout occurs, Threshold set to CongWin/2 and Congwin is set to 1 MSS.

AIMD

- Additive Increase, Multiplicative Decrease
- Happens in Congestion Avoidance phase
- Increase linearly by 1 MSS
- Decrease by factor of 2
- Illustrates the earlier "probing" concept [Sawtooth]

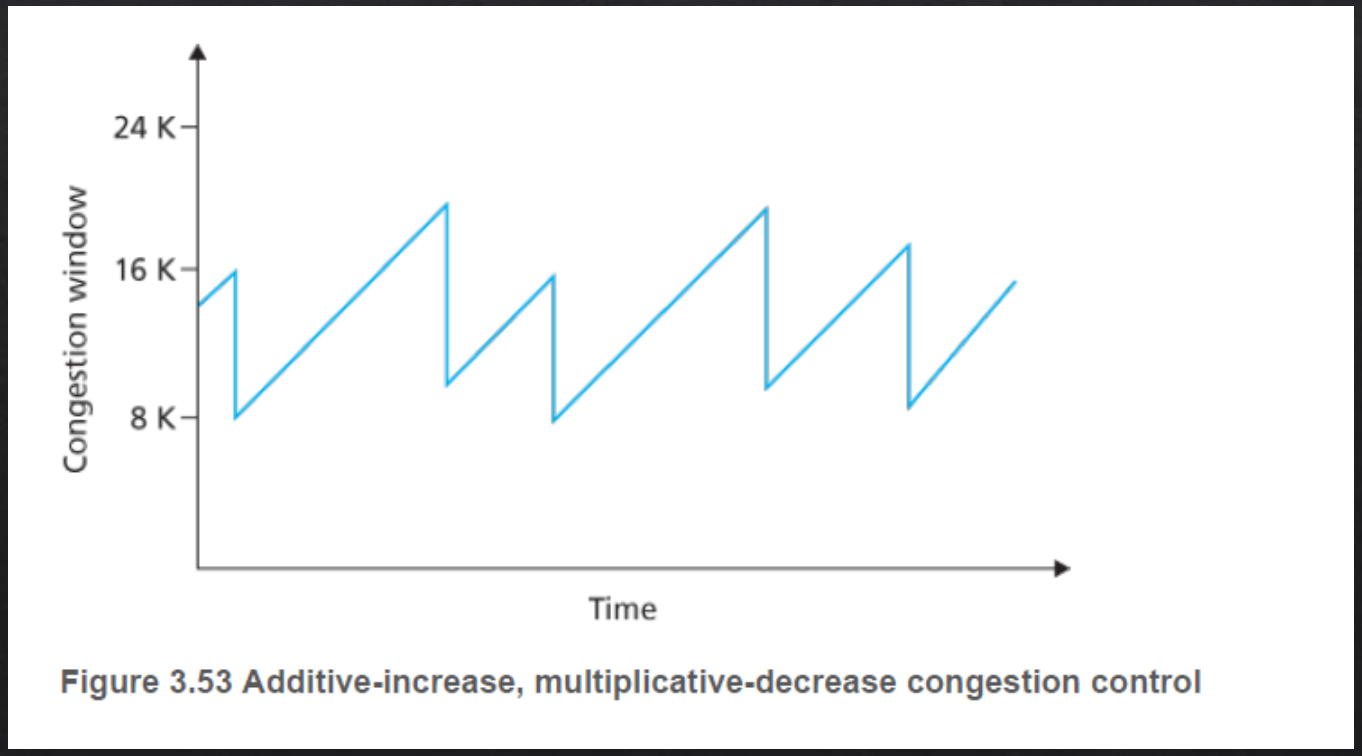


Figure 3.53 Additive-increase, multiplicative-decrease congestion control

Fast Recovery

- The actual recovery in case of packet loss
- Can be initiated through both Slow Start and Congestion avoidance phases

When does it occur?

- 3 duplicate ACK messages are received

Actions:

- Increase cwnd by 1 MSS for retransmission

    - **Case 1:** Timeout
        - ➢ Go back to Slow Start phase

    - **Case 2:** new ACK received
        - ➢ Go back to Congestion Avoidance

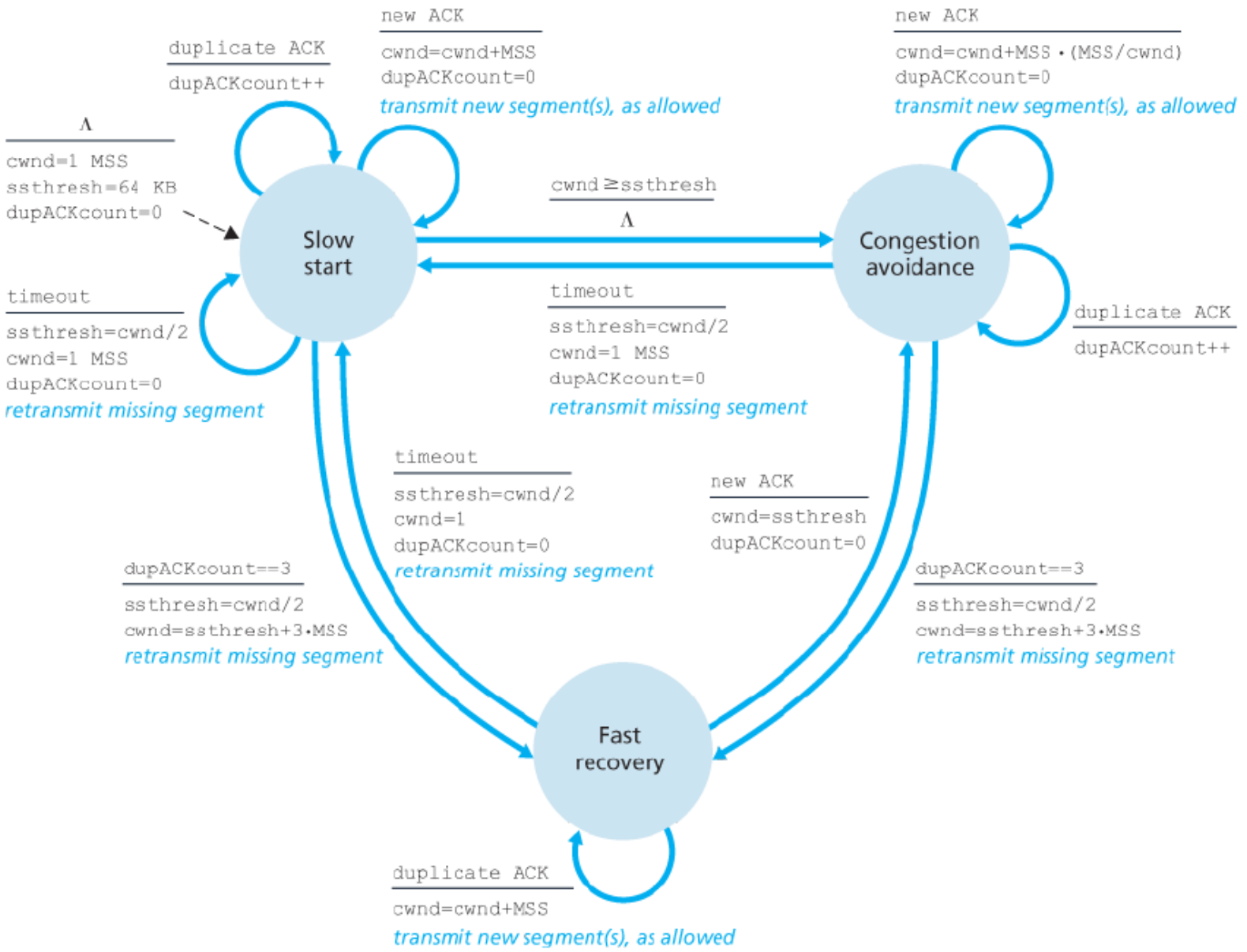


Figure 3.51 FSM description of TCP congestion control

Different versions of TCP

TCP Tahoe
- Older; No Fast Recovery, went straight into Slow Start

TCP Reno
- Newer; Implemented Fast Recovery

Initial ssthresh = 8 MSS
Loss event at 12 MSS
Fast recovery: cwnd = (ssthresh/2) + 1

w/o fast recovery: cwnd = 1, ssthresh = cwnd/2

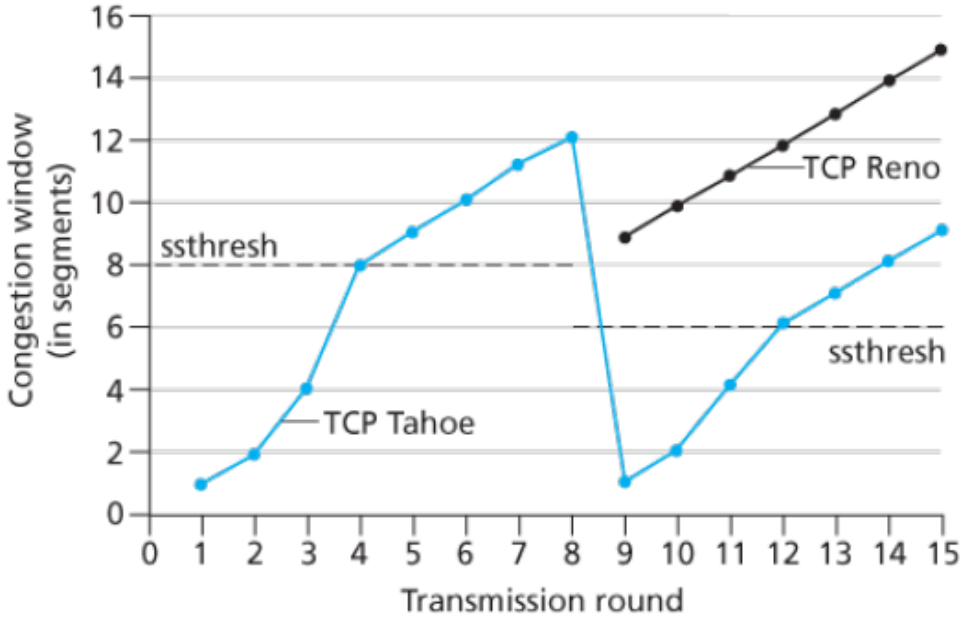


Figure 3.52 Evolution of TCP's congestion window (Tahoe and Reno)

Fairness

- Consider K TCP connections, bottleneck link of R bps
- The mechanism would be fair if average transmission rate [approximately] = R/K
- i.e., each connection gets equal share of the bandwidth

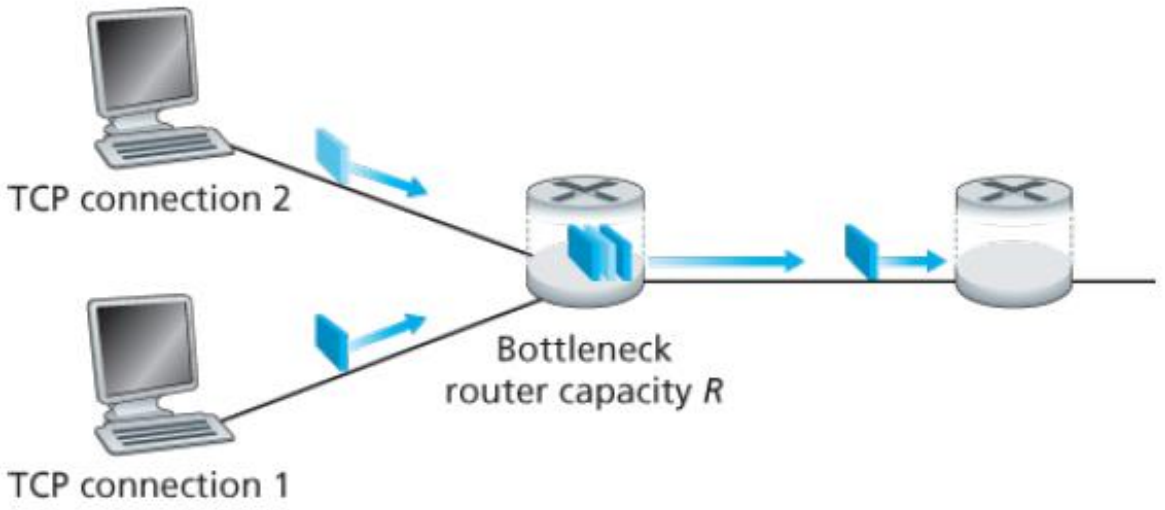


Figure 3.54 Two TCP connections sharing a single bottleneck link

Is TCP congestion control fair?

- Consider K TCP connections, bottleneck link of R bps, ignore slow start
- Ideal sum of throughputs = R
- Achieved throughputs should be near the intersection
- A to B to C to D and so on
- The connections eventually converge to equal sharing of bandwidth

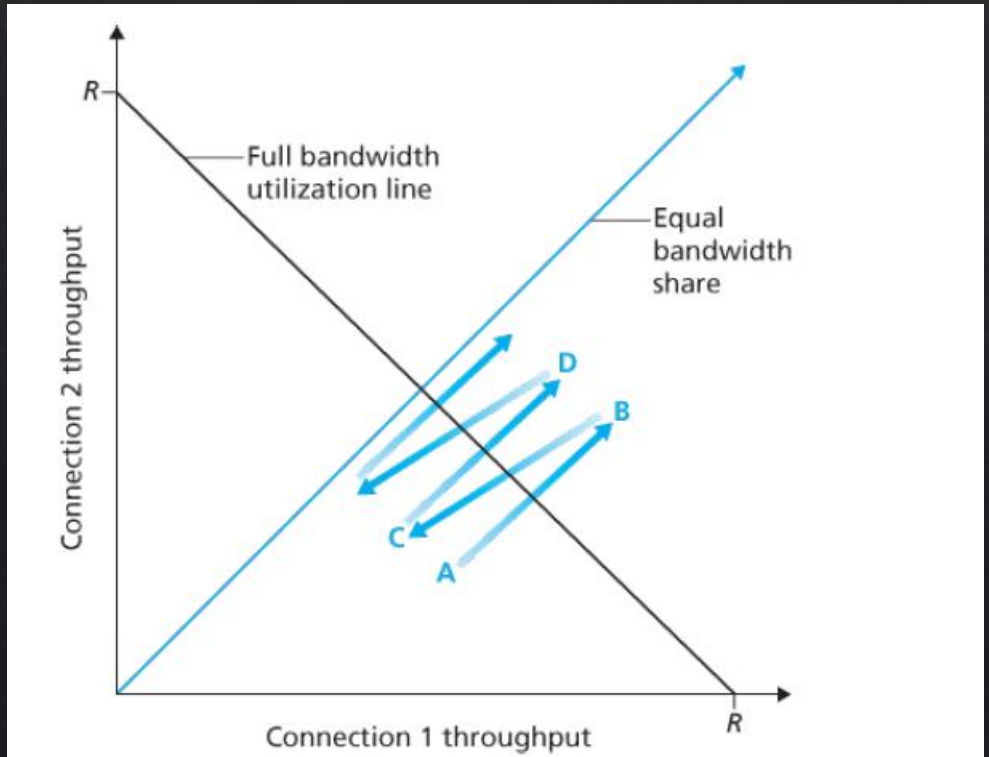


Figure 3.55 Throughput realized by TCP connections 1 and 2

- Only an ideal scenario
- In reality, multiple parallel connections are used

Explicit Congestion Notification (ECN)

- "TCP sender receives no explicit congestion indications from the network layer, and instead infers congestion through observed packet loss."
- At the network layer, two bits in the Type of Service field of the IP datagram header are used for ECN

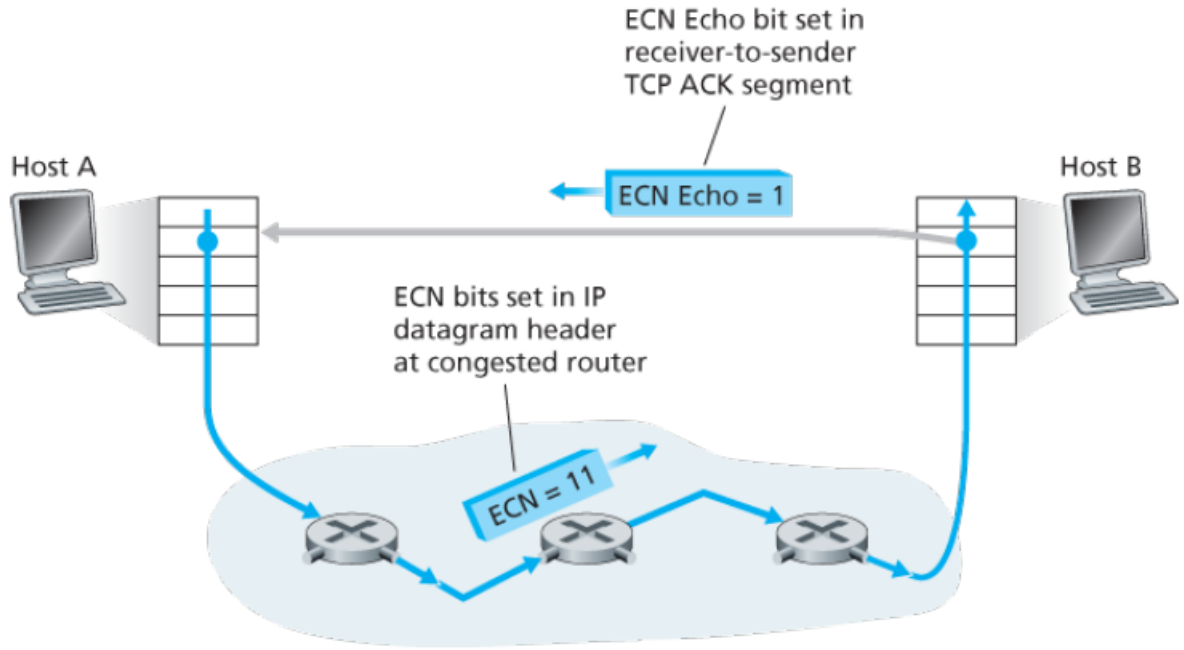


Figure 3.56 Explicit Congestion Notification: network-assisted congestion control

- One setting of the bits indicates congestion
- These are carried to the destination host
- The sender is sent an Echo bit with the TCP ACK
- The sender reacts by halving the cwnd [to prevent further congestion]