TCP
Congestion Control

TCP Congestion Control

- Essential strategy :: The TCP host sends packets into the network without a reservation and then the host reacts to observable events.
- Originally TCP assumed FIFO queuing.
- Basic idea :: each source determines how much capacity is available to a given flow in the network.
- ACKs are used to 'pace' the transmission of packets such that TCP is "self-clocking".

AIMD
(Additive Increase / Multiplicative Decrease)

- CongestionWindow (cwnd) is a variable held by the TCP source for each connection.

MaxWindow :: min (CongestionWindow , AdvertisedWindow)

EffectiveWindow = MaxWindow – (LastByteSent -LastByteAcked)

- cwnd is set based on the perceived level of congestion. The Host receives implicit (packet drop) or explicit (mark) indications of internal congestion.

Multiplicative Decrease

* The key assumption is that a dropped packet and the resultant timeout is due to congestion at a router or a switch.

Multiplicate decrease:: TCP reacts to a timeout by halving cwnd.

- Although cwnd is defined in bytes, the literature often discusses congestion control in terms of packets (or more formally in MSS == Maximum Segment Size).
- cwnd is not allowed below size of a single packet.

Additive Increase

- Additive Increase is a reaction to perceived available capacity.
- Linear Increase basic idea:: For each "cwnd's worth" of packets sent, increase cwnd by 1 packet.
- In practice, **cwnd** is incremented fractionally for each arriving ACK.

increment = MSS x (MSS /cwnd)

cwnd = cwnd + increment

AIMD
(Additive Increase / Multiplicative Decrease)

- It has been shown that AIMD is a necessary congestion for TCP congestion control to be stable.
- Because the simple CC mechanism involves timeouts that cause retransmissions, it is important that hosts have an accurate Timeout mechanism.
- Timeouts set as a function of average RTT and standard deviation of RTT.
- However, TCP hosts only sample round-trip time once per RTT using coarse-grained clock.

Slow Start

- Linear additive increase takes too long to ramp up a new TCP connection from cold start
- Beginning with TCP Tahoe, the slow start mechanism was added to provide an initial exponential increase in the size of cwnd.

Remember mechanism by: ***slow start prevents a slow start. Morevoer, slow start is slower than sending a full advertised window's worth of packets all at once.***

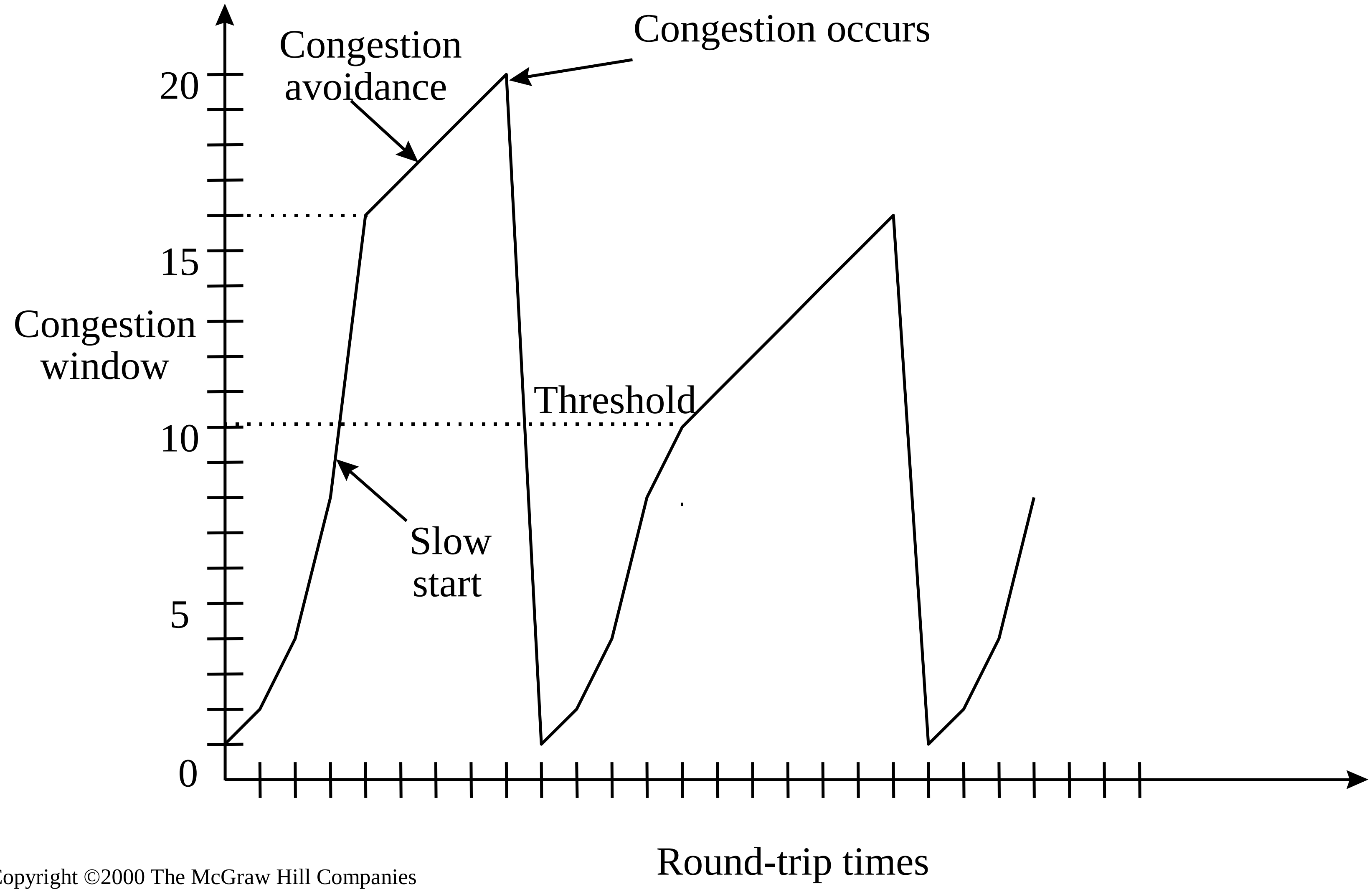
Slow Start

- The source starts with cwnd = 1.
- Every time an ACK arrives, cwnd is incremented.

➔cwnd is effectively doubled per RTT "epoch".

- Two slow start situations:
  - At the very beginning of a connection {cold start}.
  - When the connection goes dead waiting for a timeout to occur (i.e, advertized window goes to zero!)
- However, in the second case the source has more information. The current value of cwnd can be saved as a **congestion threshold**. This is also known as the "slow start threshold" **ssthresh**.

TCP Congestion Control



Leon-Garcia & Widjaja: *Communication Networks*

Figure 7.63

Fast Retransmit

- Coarse timeouts remained a problem, and Fast retransmit was added with TCP Tahoe.
- Since the receiver responds every time a packet arrives, this implies the sender will see duplicate ACKs.

Basic Idea:: *use **duplicate ACKs** to signal lost packet.*

Fast Retransmit

Upon receipt of three duplicate ACKs, the TCP Sender retransmits the lost packet.

Fast Retransmit

- Generally, fast retransmit eliminates about half the coarse-grain timeouts.
- This yields roughly a 20% improvement in throughput.
- Note – fast retransmit does not eliminate all the timeouts due to small window sizes at the source.

Fast Recovery

- Fast recovery was added with TCP Reno.

Basic idea:: When fast retransmit detects three duplicate ACKs, start the recovery process from congestion avoidance region and use ACKs in the pipe to pace the sending of packets.

Fast Recovery

After Fast Retransmit, half cwnd and commence recovery from this point using linear additive increase 'primed' by left over ACKs in pipe.

Modified Slow Start

- With fast retransmit, **slow start** only occurs:
  - At cold start
  - After a coarse-grain timeout
- *This is the difference between TCP Tahoe and TCP Reno!!*