Tutorial 3 Solution

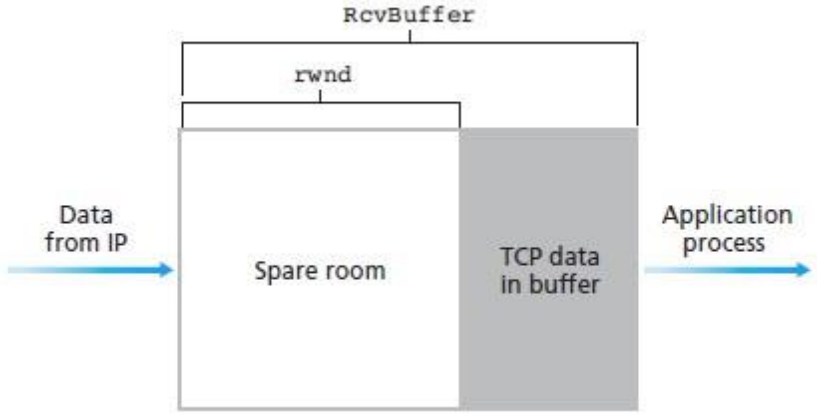
1 TCP Flow Control

Host A and B are directly connected with a 100 Mbps link. There is one TCP connection between the two hosts, and Host A is sending to Host B an enormous file over this connection. Host A can send its application data into its TCP socket at a rate as high as 120 Mbps but Host B can read out of its TCP receive buffer at a maximum rate of 50 Mbps.
Describe the effect of TCP flow control over the rate that the Application Layer of A sends data through its TCP socket.

Solution

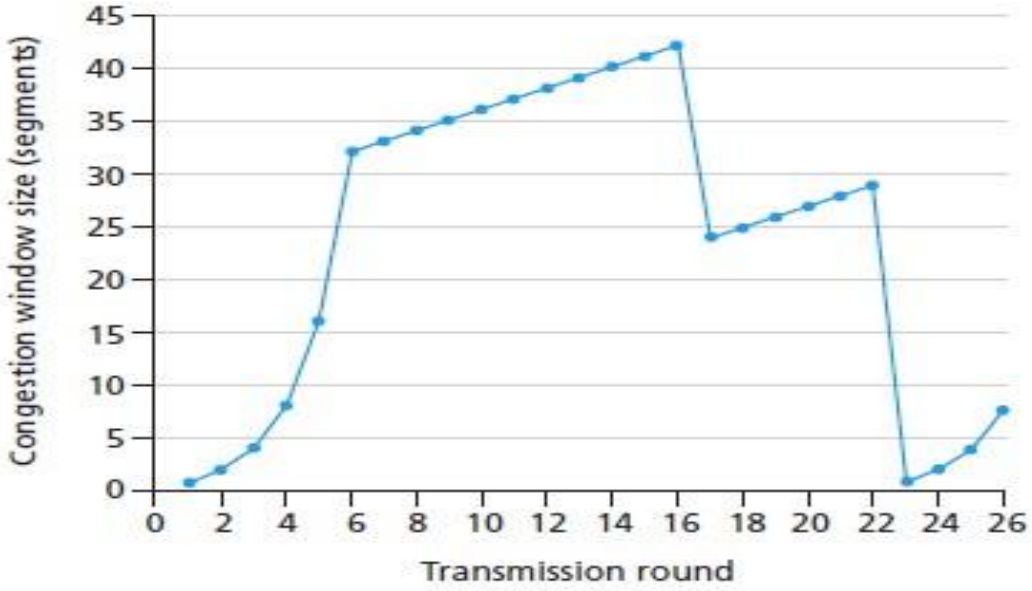
In this exercise, the main issue is that there are two bottlenecks in the process of delivering the information from Host A to Host B. The first bottleneck is between Host A and the link, because Host A can send data at a rate up to 120 Mbps, but the link can only transfer data up to 100 Mbps. The second bottleneck is between the link and Host B, because the link can transfer data up to 100 Mbps but Host B cannot receive data at a rate higher than 50 Mbps.
Since the link capacity is only 100 Mbps, Host A's sending rate can be at most 100 Mbps. However, this still means that Host A sends data into the receive buffer faster than Host B can remove data from the buffer. The receive buffer fills up at a rate of roughly 50 Mbps. TCP flow control determines the highest rate at which the receiver can receive the data. This rate is indicated in the TCP segments directed from the receiver to the sender, such that the sender knows at which rate the data can be sent at any moment.

When the buffer is full, Host B signals to Host A to stop sending data by setting the receive window $rwnd = 0$. Host A then stops sending until it receives a TCP segment with $rwnd > 0$. Host A will thus repeatedly stop and start sending as a function of *rwnd* values it receives from Host B.
On average, the long-term rate at which Host A sends data to Host B as part of this connection is no more than 50 Mbps.



2 TCP CONGESTION CONTROL

A. Consider the following plot of TCP window size as a function of time.



Assuming TCP Reno is the protocol experiencing the behavior shown in the above figure, answer the following questions. In all cases, you should provide a short discussion justifying your answer.

a. Identify the intervals of time when TCP slow start is operating.

b. Identify the intervals of time when TCP congestion avoidance is operating.

c. After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
d. After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?

e. What is the initial value of *ssthresh* at the first transmission round?

f. What is the value of *ssthresh* at the 18th transmission round?

g. What is the value of *ssthresh* at the 24th transmission round?

h. During what transmission round is the 60th segment sent?

i. During what transmission round is the 65th segment sent?

j. Assuming a packet loss is detected after the 26th round by the receipt of a triple duplicate ACK, what will be the values of *cwnd* and *ssthresh*?

SOLUTION

a. TCP slow start is operating in the intervals [1, 6] and [23, 26].

b. TCP congestion avoidance is operating in the intervals [6, 16] and [17, 22].

c. After the 16th transmission round, segment loss is recognized by a triple duplicate ACK because the value of *cwnd* was halved. If there was a timeout, *cwnd* would have dropped to 1.

d. After the 22nd transmission round, segment loss is detected due to timeout, and hence, *cwnd* is set to 1.

e. *ssthresh* is initially 32, since it is at this window size that slow start stops and congestion avoidance begins.

f. *ssthresh* is set to half the value of *cwnd* when packet loss is detected. When loss is detected during the 16th transmission round, *cwnd* is 42. Hence, *ssthresh* is 21 during the 18th transmission round.

g. *ssthresh* is set to half the value of *cwnd* when packet loss is detected. When loss is detected during the 22nd transmission round, *cwnd* is 26. Hence, *ssthresh* is 13 during the 24th transmission round.

h.
(Refer to the graph from the [1,6] transmission round, slow start is operating and *cwnd* increases exponentially).
During the 1st transmission round, packet 1 is sent.
During the 2nd transmission round, packets 2-3 are sent.
During the 3rd transmission round, packets 4-7 are sent.
During the 4th transmission round, packets 8-15 are sent.
During the 5th transmission round, packets 16-31 are sent.

During the 6th transmission round, packets 32-63 are sent. Thus,
packet 60 is sent in the 6th transmission round.

i.
(Refer to the graph after the 6th transmission round, cwnd increases linearly)
During the 7th transmission round, packet 64 is sent.
During the 8th transmission round, packet 65 is sent.

B. Consider sending a large file from a host to another over a TCP connection that has no loss. Assume that the initial value of *cwnd* is 5 MSS.

a. Suppose TCP uses AIMD for its congestion control without slow start. Assuming *cwnd* increases by 1 *MSS* every time a batch of ACKs is received and assuming approximately constant round-trip times, how long does it take for cwnd to increase from 6 *MSS* to 12 *MSS* (assuming no loss events)?

b. What is the average throughput (in terms of *MSS* and *RTT*) for this connection up through time = 6 *RTT*?

SOLUTION

a. Since we are considering the case of AIMD without slow start, TCP directly enters the congestion avoidance phase, during which the value of *cwnd* increases by just 1 *MSS* every *RTT*.

It takes:

1 *RTT* to increase *cwnd* to 6 *MSS*.
2 *RTT*s to increase to 7 *MSS.* 3 *RTT*s to increase to 8 *MSS*.
4 *RTT*s to increase to 9 *MSS*.
5 *RTT*s to increase to 10 *MSS*.
6 *RTT*s to increase to 11 *MSS*.
Finally, it takes 7 *RTT*s to increase to 12 *MSS*.

b.
In the 1st *RTT*, 5 *MSS* was sent.
In the 2nd *RTT*, 6 *MSS* was sent.
In the 3rd *RTT*, 7 *MSS* was sent. In
the 4th *RTT*, 8 *MSS* was sent.
In the 5th *RTT*, 9 *MSS* was sent.
In the 6th *RTT*, 10 *MSS* was sent.

Thus, up to time 6 *RTT*, 5 + 6 + 7 + 8 + 9 + 10 = 45 *MSS* were sent and acknowledged.
Therefore, we can say that the average throughput up to time 6 *RTT* = (45 *MSS*) / (6 *RTT*) = 7.5 *MSS*/*RTT*

3 TCP FLOW CONTROL VS CONGESTION CONTROL

Host A is sending an enormous file to Host B over a TCP connection. Over this connection, there is never any packet loss and the timers never expire.
Denote the transmission rate of the link connecting Host A to the Internet by *R* bps. Suppose that the process in Host A is capable of sending data into its TCP socket at a rate *S* bps, where *S* = 10 . *R*. Further suppose that the TCP receive buffer is large enough to hold the entire file, and the send buffer can hold only one percent of the file.
What would prevent the process in Host A from continuously passing data to its TCP socket at rate *S* bps? TCP flow control? TCP congestion control? Or something else? Elaborate.

SOLUTION

In this problem, there is no danger in overflowing the receiver since the receiver's receive buffer can hold the entire file. As such, TCP flow control does not play a role in this particular scenario.
Also, because there is no loss and acknowledgments are returned before times expire, TCP congestion control does not throttle the sender.
However, the process in Host A will not continuously pass data to its TCP socket because the send buffer will quickly fill up. Once the send buffer become full, the process will pass data at any average rate of $R << S$, because the sender has to adapt to the bottleneck imposed by the link's much lower transmission rate.

4

A. TRUE OR FALSE

a. Suppose Host A is sending a large file to Host B over a TCP connection.
    1. If the sequence number for a segment of this connection is *m*, then the sequence number for the subsequent segment will necessarily be *m* + 1.

    2. The number of unacknowledged bytes that A sends cannot exceed the size of the receive buffer.

b. Suppose Host A sends one segment with sequence number 38 and 4 bytes of data over a TCP connection to Host B. In this same segment the acknowledgment number is necessarily 42.

c. The size of TCP *rwnd* never changes throughout the duration of the connection.

d. The TCP segment has a field in its header for *rwnd*.

e. Consider TCP congestion control. When the timer expires at the sender, the threshold is set to one half of its previous value.

SOLUTION

a.

1. False

2. True

b. False (because Hosts A and B might be using different sequence numbers to label their packets)

c. False

d. True

e. True

B. Consider that only a single TCP Reno connection uses one 10 Mbps link which does not buffer any data. Suppose that this link is the only congested link between the sending and receiving hosts.

Assume that the TCP sender has a huge file to send to the receiver, and the receiver's receive buffer is much larger than the congestion window.
We also make the following assumptions:

- Each TCP segment size is 1,500 bytes.
- The two-way propagation delay of this connection is 150 msec.
- This TCP connection is always in congestion avoidance phase, that is, ignore slow start.

a. What is the maximum window size (in segments) that this TCP connection can achieve?

b. What is the average window size (in segments) and average throughput (in bps) of this TCP connection?

c. How long would it take for this TCP connection to reach its maximum window again after recovering from a packet loss?

SOLUTION

a. Let W denote the maximum window size measured in segments.

This means that (W) (MSS) / (RTT) = 10 Mbps, as packets will be dropped if the maximum sending rate exceeds link capacity.
Thus, we have (W) (1500 x 8) / (150 x 10^-3) = (10 x 10^6), then W is 125.

b. As window size varies from W/2 to W, then the average window size = 0.75W = (0.75) (125) = 93.75.

Average throughput = (Wavg) (*MSS*) / (*RTT*) = (93.75) (1500 x 8) / (150 x 10^-3) = 7.5 Mbps.

c. Recall that, when packet loss occurs, window size becomes $W/2$. In order to recover from a packet loss, congestion avoidance increases the window size by one in each *RTT*.

As such, the number of *RTTs* needed in order to increase the window size from $W/2$ to W = ($W/2$) *RTTs* = (125/2) (150 x) = 9.375 seconds.