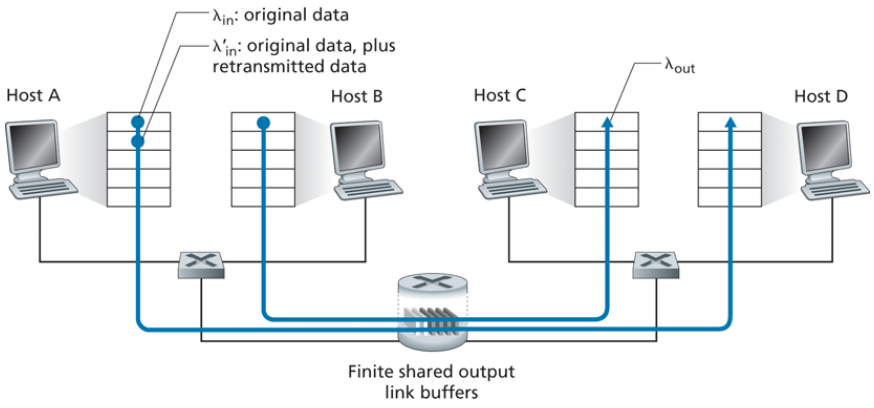
P26. Consider transferring an enormous file of L bytes from Host A to Host B. Assume an MSS of 536 bytes.

a. What is the maximum value of L such that TCP sequence numbers are not exhausted? Recall that the TCP sequence number field has 4 bytes.
b. For the L you obtain in (a), find how long it takes to transmit the file. Assume that a total of 66 bytes of transport, network, and data-link header are added to each segment before the resulting packet is sent out over a 155 Mbps link. Ignore flow control and congestion control so A can pump out the segments back to back and continuously.

P27. Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A.

a. In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?
b. If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number, the source port number, and the destination port number?
c. If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?
d. Suppose the two segments sent by A arrive in order at B. The first acknowledgment is lost and the second acknowledgment arrives after the first timeout interval. Draw a timing diagram, showing these segments and all other segments and acknowledgments sent. (Assume there is no additional packet loss.) For each segment in your figure, provide the sequence number and the number of bytes of data; for each acknowledgment that you add, provide the acknowledgment number.

P30. Consider the network shown in Scenario 2 in **Section 3.6.1**. Suppose both sending hosts A and B have some fixed timeout values.



a. Argue that increasing the size of the finite buffer of the router might possibly decrease the throughput (λ_{out}).
b. Now suppose both hosts dynamically adjust their timeout values (like what TCP does) based on the buffering delay at the router. Would increasing the buffer size help to increase the throughput? Why?

P32. Consider the TCP procedure for estimating RTT. Suppose that $\alpha=0.1$. Let `SampleRTT`$_1$ be the most recent sample RTT, let `SampleRTT`$_2$ be the next most recent sample RTT, and so on.

a. For a given TCP connection, suppose four acknowledgments have been returned with corresponding sample RTTs: `SampleRTT`$_4$, `SampleRTT`$_3$, `SampleRTT`$_2$, and `SampleRTT`$_1$. Express `EstimatedRTT` in terms of the four sample RTTs.
b. Generalize your formula for *n* sample RTTs.
c. For the formula in part (b) let *n* approach infinity. Comment on why this averaging procedure is called an exponential moving average.

P33. In **Section 3.5.3**, we discussed TCP's estimation of RTT. Why do you think TCP avoids measuring the `SampleRTT` for retransmitted segments?

P39. Consider **Figure 3.46(b)**. If λ'_{in} increases beyond $R/2$, can λ_{out} increase beyond $R/3$? Explain. Now consider **Figure 3.46(c)**. If λ'_{in} increases beyond $R/2$, can λ_{out} increase beyond R/4 under the assumption that a packet will be forwarded twice on average from the router to the receiver? Explain.

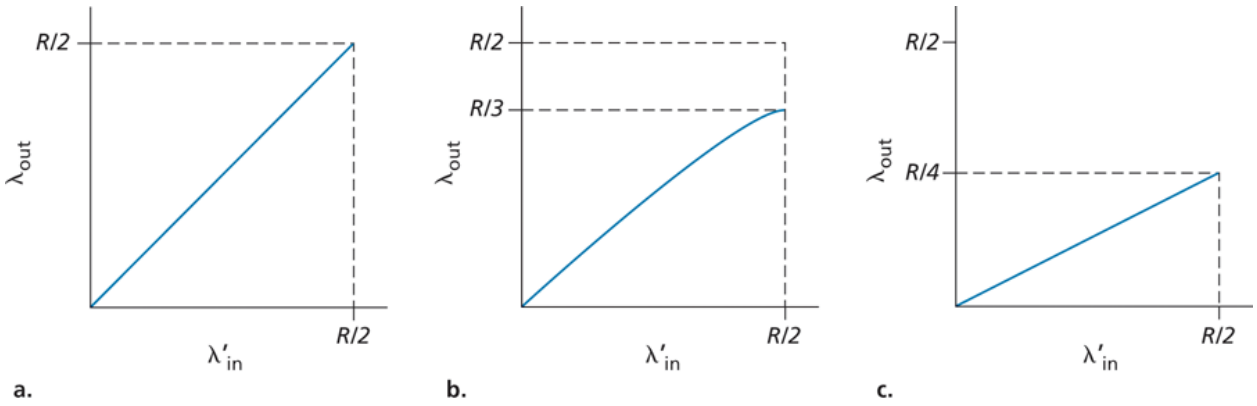
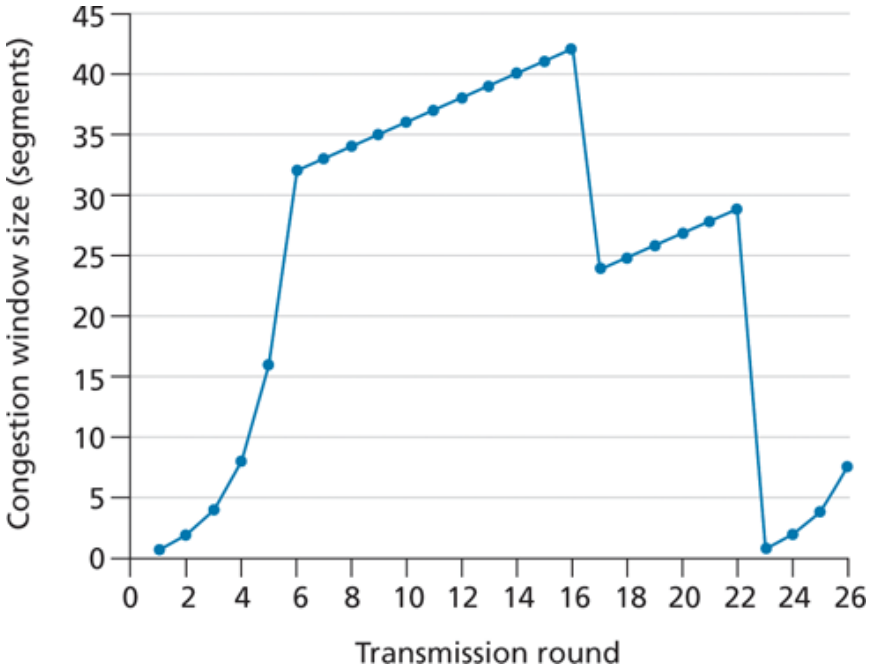


Figure 3.46 Scenario 2 performance with finite buffers

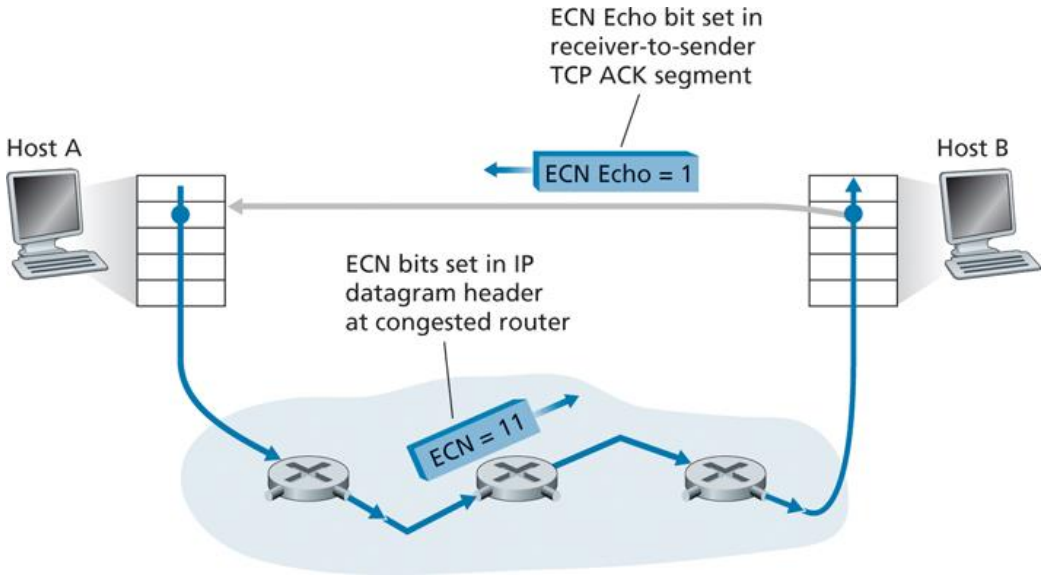
P40. Consider **Figure 3.61**. Assuming TCP Reno is the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying your answer.



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 70th segment sent?
i. Assuming a packet loss is detected after the 26th round by the receipt of a triple duplicate ACK, what will be the values of the congestion window size and of `ssthresh`?
j. Suppose TCP Tahoe is used (instead of TCP Reno), and assume that triple duplicate ACKs are received at the 16th round. What are the `ssthresh` and the congestion window size at the 19th round?
k. Again suppose TCP Tahoe is used, and there is a timeout event at 22nd round. How many packets have been sent out from 17th round till 22nd round, inclusive?

P41. Refer to **Figure 3.55**, which illustrates the convergence of TCP's AIMD algorithm. Suppose that instead of a multiplicative decrease, TCP decreased the window size by a

constant amount. Would the resulting AIAD algorithm converge to an equal share algorithm? Justify your answer using a diagram similar to **Figure 3.55**.



P42. In **Section 3.5.4**, we discussed the doubling of the timeout interval after a timeout event. This mechanism is a form of congestion control. Why does TCP need a window-based congestion-control mechanism (as studied in **Section 3.7**) in addition to this doubling-timeout-interval mechanism?

P43. 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 $S=10 \cdot R$. 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.