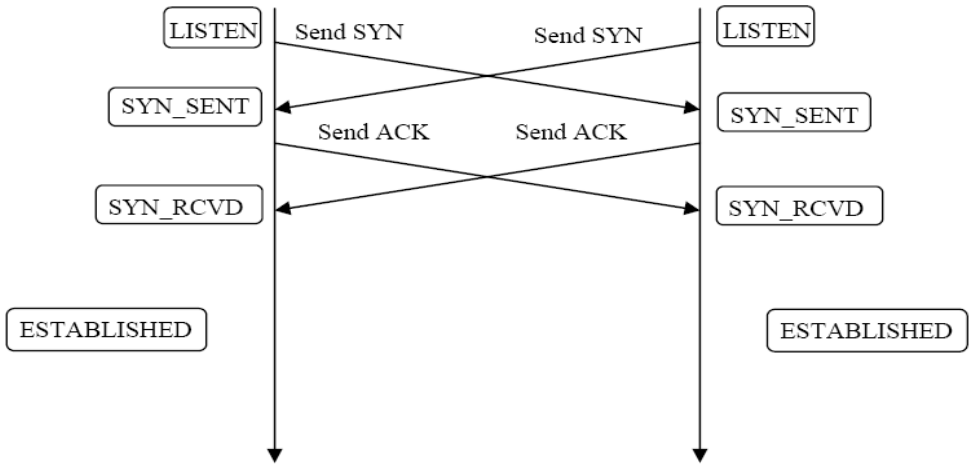
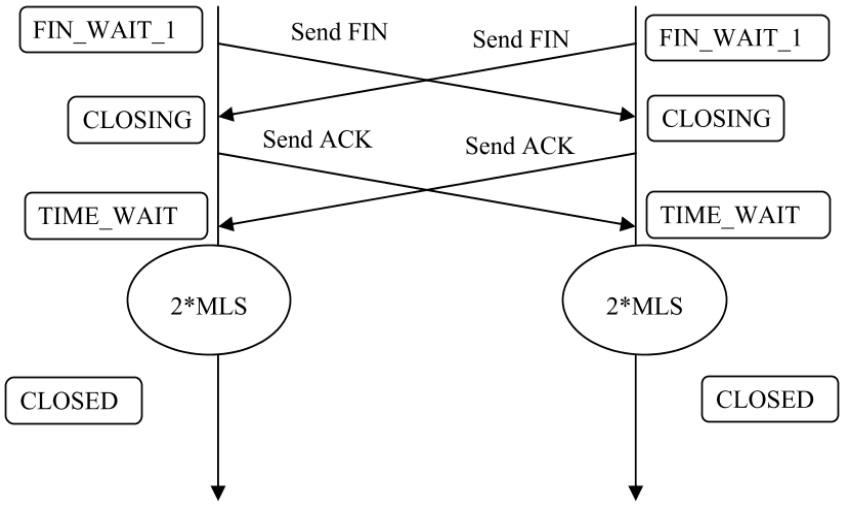
Homework 2

1. Compare and Contrast the following flavors (implementations) of TCP using any and all resources at your disposal, in addition to your textbook. Be sure to cite your references at the end of your answer. Will be graded online – solution not provided here, but answers are in textbook.
   - TCP Reno
   - TCP Tahoe
   - TCP Vegas
   - Fast TCP

2. *Simultaneous Open*: The TCP state transition diagram allows for the case where the two stations issue a SYN segment at nearly the same time. Draw the sequence of segment exchanges and use the TCP state transition diagram to show the sequence of states that are followed by the two stations in this case.



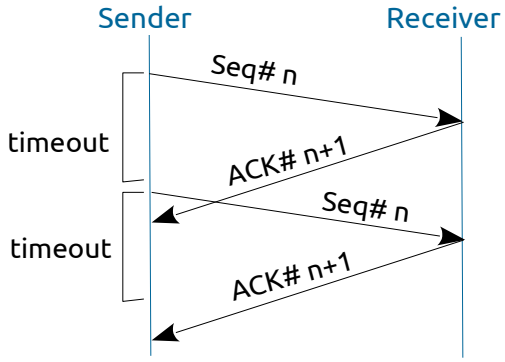
3. *Simultaneous Close*: The TCP state transition diagram allows for the case where the two stations issue a FIN segment at nearly the same time. Draw the sequence of segment exchanges and use the TCP state transition to show the sequence of states that are followed by the two stations in this case.



4. What is the impact on bandwidth utilization if the TCP timeout value is

a. too short

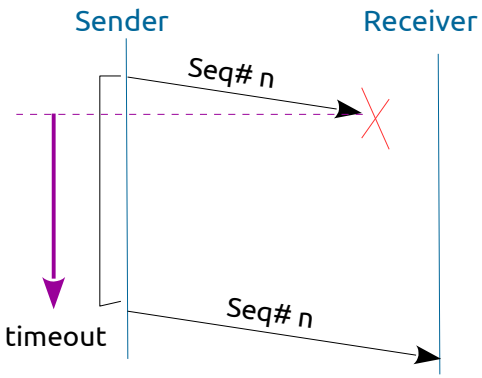
If the timeout is too short, the following is the most likely scenario:

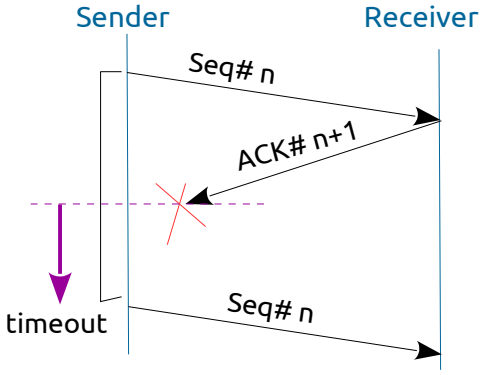


The first ACK# *n*+1 does not arrive within the first timeout interval; this causes the segment with seq# *n* to be retransmitted. Soon thereafter, the first ACK arrives (in the 2nd timeout interval). The sender has no way of knowing that this is the first (delayed) ACK. It matches this first ACK to the 2nd i.e. retransmitted Seq# *n*+1 and therefore accepts it. The receiver, of course, rejects the retransmitted segment since it is a duplicate. Thus, every segment is transmitted twice; but the receiver accepts only one copy. Therefore, the utilization is twice what it should be, and we end up wasting a lot of the bandwidth. Throughput is 50% of the utilization – not a good outcome.

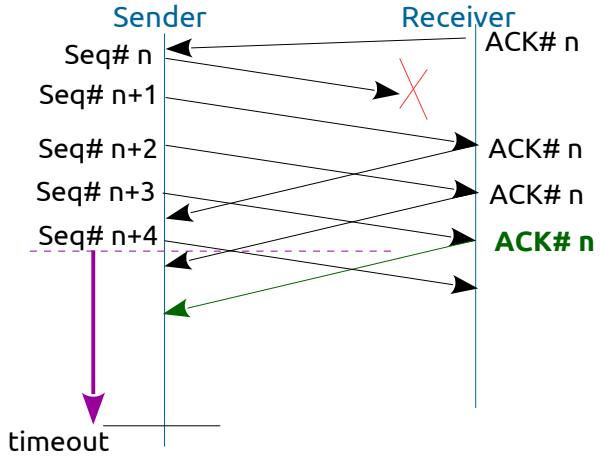
If instead of sending just one segment at a time, the sender sends multiple segments at a time in its effective window, the situation is not much different; each successive ACK arrives after each corresponding segment within the sender's window is retransmitted.

b. too long





The above two figure show what happens when either a segment is lost (left figure), or when the ACK to that segment is lost (right figure). From the moment of loss (shown by the dashed magenta line), the sender must wait for the timeout to occur. The magenta arrow shows the wait for the timeout to occur and hence the time of wastage of the bandwidth. Clearly, the longer the timeout, the greater the wastage of bandwidth. If multiple segments are transmitted within a window by the sender ..



After the 3rd duplicate ACK (shown in green), if the sender's TCP process implements a fast retransmit, then an immediate retransmission of seq# n is done without waiting for the timeout; so the long timeout does not have an adverse consequence.

If fast retransmit is not implemented, then, the sender's TCP waits for a timeout before retransmission. The idle time is right after the seq# n+4 is transmitted, till timeout occurs (shown by the magenta arrow). Thus, if the timeout is too long, we could waste considerable bandwidth.

c. Which is worse (a) or (b)? Why?

(a) is clearly worse since we have high utilization, but half the throughput – thus even though the sender's idle time decreases, its must retransmit every segment twice. That is, it must perform a lot of useless work and waste bandwidth through retransmissions.

Furthermore, in the case of (a), a timeout occurs on ***every*** segment and thus causes a guaranteed adverse effect on throughput. (b) however results in adverse effects ***only when*** a segment or its ACK is lost, not a very frequent occurrence.

d. Would you prefer RFC 793 over the Jacobson-Karels algorithm for timer selection? Explain.

RFC 793 sets a much looser bound on the timeout value – it uses only the sample and estimated RTT values in computing the next-round estimates of RTT, which it then uses in setting the timeout value. It is an inherently simple algorithm that imposes very little computational overhead, per packet. Therefore, for low-cost devices wherein efficiency of bandwidth utilization is a secondary concern, RFC 793 might be preferable.

The J-K algorithm computes the timeout value using not only the sample and estimated RTT, but also the mean deviation of the RTTs. This results in a much more accurate estimate of the timeout value. However, this comes at the cost of increased computational overhead per packet. Therefore, for devices wherein efficiency of bandwidth utilization is a primary concern, but cost (due to hardware/software complexity) is not, the J-K algorithm is preferable.

5. A fast typist can do 100 words a minute, and each word has an average of 6 characters. Demonstrate Nagle's algorithm by showing the sequence of TCP segment exchanges between a client, with input from our fast typist, and a server. Indicate how many characters are contained in each segment sent from the client. Consider the following two cases:

The typist types 100 words per minutes, averaging 6 characters per word. This is equivalent to 600 characters per minute or 10 characters per second. Therefore, the typist can type a character every 100 ms.

a. The client and server are in the same LAN and the RTT is 20 ms.

In this case Nagle's Algorithm is not activated since acknowledgments arrive before the next character is typed. Each client segment is 41 bytes long assuming IP and TCP headers are 20 bytes each.

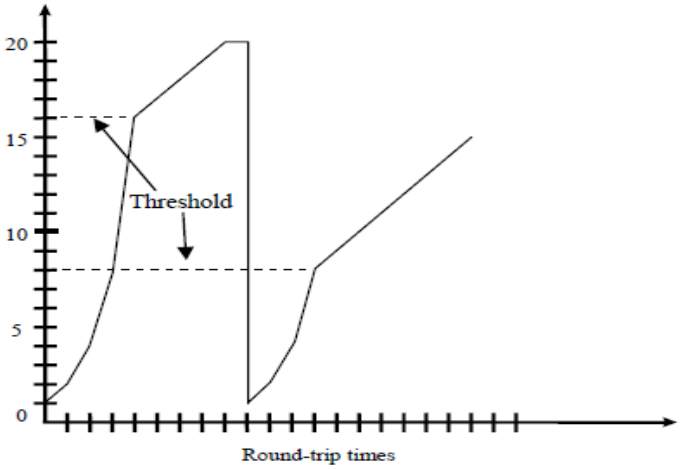
b. The client and server are connected across a WAN and the RTT is 100 ms.

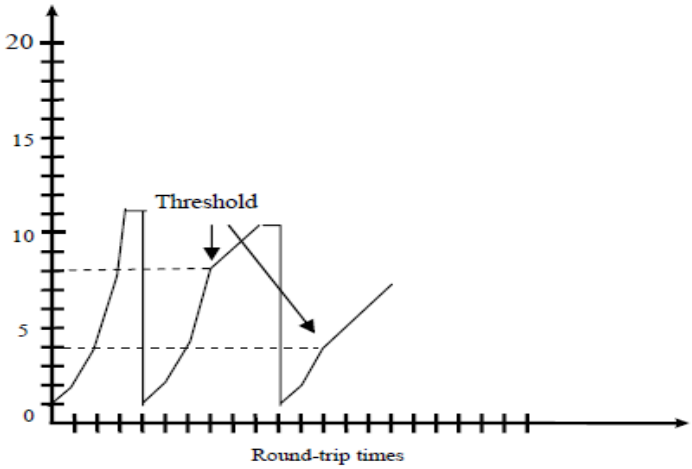
In this case one or two characters are typed before an acknowledgment is received. Therefore, segments are either 41 or 42 bytes long (including the 20 byte TCP header and 20 byte IP header overhead).

Note that the problem only mentions the average character generation rate; it is reasonable to assume (and expected) that the inter-arrival time between characters is not constant Thus, depending on your assumptions on the exact times that the characters are generated by the typist, your diagrammed scenario may differ from the one mentioned in this answer.

6. Suppose that a TCP source (with unlimited amount of information to transmit) begins transmitting onto a link that has 1 Mbps in available bandwidth. Sketch congestion window versus time trajectory. Now suppose that another TCP source (also with unlimited amount of information to transmit) begins transmitting over the same link. Sketch the congestion window versus the time for the initial source.

Initially the TCP source has about 1 Mbps of available bandwidth, so its congestion avoidance behavior will begin as the rate approaches 1 Mbps. When the available bandwidth drops to 500 kbps, the congestion window behavior will begin sooner. The following two figures that follow show the corresponding congestion window trajectories.





Note that if you assumed that the initial slow start threshold (ss_thresh) was set very high, then the plot would show that the first packet drop would occur when TCP was still in the slow start phase; your plot would then be slightly different i.e. more like the 2nd figure in shape, but on a different scale. That is OK.

7. Explain the relationship between advertised window size, RTT, delay-bandwidth product, and the maximum achievable throughput in TCP.

a. Plot the maximum achievable throughput versus delay-bandwidth product for an advertised window size of 65,535 bytes.

The delay-bandwidth product, *DBP* = *RTT* x *B*. That is, *B* = *DBP* / *RTT* .. (I)

Here, *B* is the bandwidth available and *RTT* is the round trip delay. The *RTT* measures the delay (latency) is the minimum time that elapses from when a packet leaves a source to when the acknowledgment is received. It can be approximated as *2 x one-way delay* for a bit traveling from the source to the destination. *RTT* includes not only the propagation delay, but also queuing and transmission delays. For simplicity, if we assume that the transmission delay of a single-bit is being considered, and if the queuing delay is negligible, then the RTT is simply equal (roughly) to the propagation delay of the radio/electrical/optical signal itself.

The delay-bandwidth product *DBP* is then the number of bits (or bytes) that are in the network when the source transmits continuously at the maximum rate and when the first ACK segment returns to the source. The advertised window *W*, places an upper limit on the amount of data that a source can transmit before the first ACK returns. If *W* < *DBP*, then we cannot have 100% bandwidth utilization. If *W* ≥ *DBP* then we have 100% utilization.

Now, throughput is defined as the amount of data delivered correctly to the destination per unit time. Very simply, since we are delivering a window's worth of data (*W*) in one *RTT*, the achieved throughput, ***r = W / RTT***. .. (II)

Also, since ***r = B* x (fraction of time bandwidth *B* is in use) – (total losses due to errors/drops)**

if we use our bandwidth 100% of the time, AND if no error-related packet losses or congestion-related packet drops occur, then we can achieve maximum throughput. Accordingly, by setting "fraction of time" = 1 and "total losses" = 0 in the above equation, we get:

$r_{max} = B$. Therefore, throughput is hard-limited by our available bandwidth, B. .. (III)

Thus, throughput, $r = \min\{B, W/RTT\}$ .. from (II) and (III)

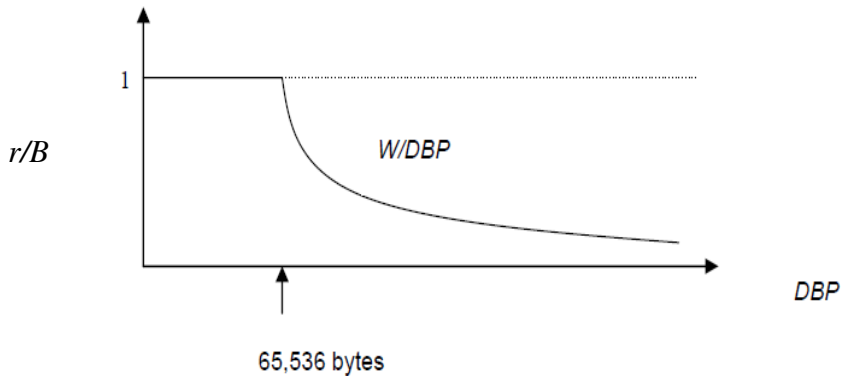
$= \min\{DBP/RTT, W/RTT\}$ .. by substituting from (I)

In the above equation, if $W < DBP$, then $r = W/RTT$ .. (IV)
But if $W \geq DBP$, then $r = DBP/RTT = B$ .. (V)

Also note from above equations that since B and W are fixed, should the RTT double, the DBP doubles too, thus halving r (see eqn. IV). If the DBP triples, the RTT triples too, and r is reduced to a third. And so on. In the other direction, if DBP decreases, then according to eqn. (V), r cannot increase beyond the maximum, B. This is reflected in the plot that follows.

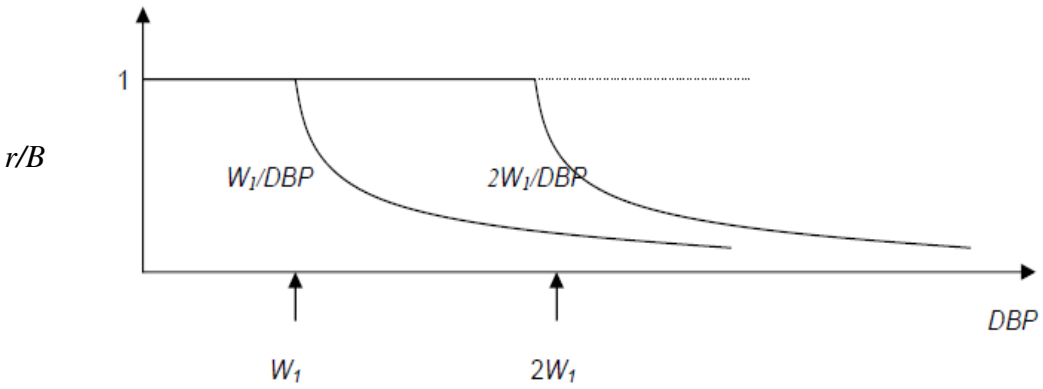
For the plot:

You can choose to plot r (on the Y-axis) against DBP (on the X-axis), OR, you can plot the normalized throughput (r/B) against DBP. Using the normalized throughput makes it a bit easier to observe the nature of the curves without worrying about the scale. Note that, since $r_{max} = B$ (from III), on the Y-axis, the maximum value of r/B is 1.



b. In the above plot include the maximum achievable throughput when the above window size is scaled up by a factor of 2^K, where $K = 4, 8, 12$.

The following figure shows the case where the window size is doubled.



c. Place the following scenarios in the plot obtained in part (b):

c.1 Ethernet with 1 Gbps and distance 100 meters

c.2 2.4 Gbps and distance of 6000 km

c.3 Satellite link with speed of 45 Mbps and RTT of 500 ms

c.4 40 Gbps link with distance of 6000 km.

Note: We are assuming that the RTT is approx. equal to the propagation delay and that the velocity of the signal is the same (= 2.5×10^8 m/s) in all instances. In reality, the signal velocity varies based on the PHY medium.

Case	*DBP* implied
Ethernet R = 1 Gbps D = 100 m	$T_p = 100 / 2.5 \times 10^8$ $T_p = 4 \times 10^{-7}$ $DBP = 2 * T_p * R$ $DBP = 8 \times 10^{-7} * 1 \times 10^9$ $DBP = 8 \times 10^2$ **DBP = 800 bits**
Link R = 2.4 Gbps D = 6000 km	$T_p = 6 \times 10^3 / 2.5 \times 10^5$ $T_p = 2.4 \times 10^{-2}$ $DBP = 2 * T_p * R$ $DBP = 2 * 2.4 \times 10^{-2} * 2.4 \times 10^9$ $DBP = 11.52 \times 10^7$ **DBP = 115.2 Mbits (14.4 Mbytes)**
Satellite link R = 45 Mbps RTT = 500 ms (5×10^{-1} sec)	$DBP = RTT * R$ $DBP = 5 \times 10^{-1} * 45 \times 10^6$ $DBP = 225 \times 10^5$ bits **DBP = 22.5 Mbits (2.85Mbytes)**
link R = 40 Gbps D = 6000 km (6×10^6 m)	$T_p = 6 \times 10^3 / 2.5 \times 10^5$ $T_p = 2.4 \times 10^{-2}$ $DBP = 2 * T_p * R$ $DBP = 2 * 2.4 \times 10^{-2} * 40 \times 10^9$ $DBP = 192 \times 10^7$ **DBP = 1.92 Gbits (240 Mbytes)**