1. **What causes congestion collapse to occur?**
   1. An increase in traffic load suddenly results in a decrease in useful work done. At some point the network reaches saturation at which point increasing the load no longer results in getting useful work done.
   2. One possible cause is the spurious retransmissions of packets still in flight. When senders don’t receive acknowledgements for packets in a timely fashion, they can spuriously retransmit. Thus resulting in many copies of the same packet being outstanding.
   3. Another cause of congestion collapse is simply undelivered packets, where packets consume resources and are dropped somewhere else in the network. The solution to this is to apply congestion control.
2. **What is the difference between fairness and efficiency in a congestion control scheme?**
   1. Both of these are goals of congestion control.
   2. The first goal is to use network resources efficiently. We should never have spare capacity in the network.
   3. Preserve fair allocation of resources. Everyone gets their fair share.
   4. X1+x2 = c in phase plot. A crossing line between the first line represents fairness. The first represents efficiency.
3. **Assuming traditional TCP RENO with AIMD behavior (i.e. the version presented in the lecture videos), suppose a TCP flow’s bottleneck link has a 1gbps capacity, and that link is not being shared with any other flows. What will the average throughput of that flow be in megabits per second?** 
   1. Afae
4. **What circumstances lead to the incast problem? (in other words, what factors must be present for incast to occur?)**
   1. The presence of many parallel requests coming from servers and the fact that the switches that the servers are connected to have small buffers, leads to a problem (throughput collapse / incast problem). TCP Incast is a drastic reduction in application throughput that results when servers using TCP all simultaneously request data leading to a gross underutilization of network capacity in many to 1 communication networks like a data center. The filling of the buffers on the switches results in bursty retransmissions that overfill the switch buffers. And these bursty retransmissions are caused by TCP timeouts. They can laster hundreds of milliseconds. But the RTT in a data center network is generally less than a millisecond. Because the RTT are so much less than the timeouts, the senders will have to wait for the TCP timeout before they retransmit. And application throughput can results in 90% in reduction.
5. **Suppose you are working on some live video call software (think skype or Google hangouts) and decide to build your application protocol on top of UDP (rather than TCP). Give as many different points as you can (minimum two) that help justify that decision.** 
   1. TCP is not a good fit for congestion control for streaming video. TCP transmits lost packets and retransmissions are not always useful. TCP also slows down its sending rate after packet loss. Which may cause starvation at the client. A TCP header of 20 bytes for every packet is large for audio samples. And sending acknowledgements for every other packet is overkill. UDP would be better because it does not retransmit lost packets or adjust the sending rate. This leaves the decisions of whether or not to transmit and whether to adapt the sending rate to the application.
6. **Why does the linear growth rate of TCP-RENO (1/RTT) perform poorly for short lived flows in networks with large bandwidth and delay products?**
7. **Describe the operation of BIC-TCP’s binary search algorithm for setting the congestion window. Once stable, how does BIC-TCP react to changes in available bandwidth i.e. what happens when there is a sudden increase or decrease in available bandwidth?** 
   1. BIC-TCP uses a binary search algorithm where the window grows to the mid-point between the last window size (i.e. the max) where the TCP has a packet loss and the last window size (i.e., min) it does not have a loss for one RTT period. This search makes sense because the capacity of the current path must be somewhere between the min and max window sizes if the network conditions do not change too quickly since the last congestion signal (which is the last packet loss). After the window grows to the mid-point, if the network does not have packet losses, then it means that the network can handle more traffic and thus BIC-TCP sets the mid-point to be the new min and performs another “binary-search” with the min and max windows. This has an effect of growing the window really fast when the current window size is far from the available capacity of the path, and furthermore, if it is close to the available capacity (where we had the previous loss), it slowly reduces its window increment. It has the smallest window increment at the saturation point and its overshoots amount beyond the saturation point where losses occur very small.
8. **How does the replacement of this congestion control algorithm with a cubic grown function in CUBIC-TCP improve on BIC-TCP? Discuss.**
9. **What is the purpose of the following regions of the CUBC growth function:**
   1. **Concave –** After the flow enters into congestion avoidance from fast recovery, it starts to increase the window using the concave profile of the cubic function.
   2. **Plateau** – The cubic function is set to have its plateau at Wmax so the concave growth continues until the window size becomes Wmax.
   3. **Convex** – After the plateau, the cubic function turns into a convex profile and the convex growth begins.
10. **How does CUBIC’s fast convergence mechanism detect a reduction in available bandwidth (i.e a new flow for competing bandwidth)?** 
    1. When a loss event occurs, before a window reduction of the congestion window, the protocol remembers the last value of Wmax before it updates Wmax for the current loss event. At a loss event, if the current value of Wmax is less than the last value of it, Wlast\_max, this indicates that the saturation point experienced by this flow is getting reduced because of the change in available bandwidth.