COMP 431 Internet Protocols & Services

Spring 2017  
Kevin Jeffay

Worksheet 14, March 21 & 23

1. Consider transferring an enormous file of *F* bytes via TCP. Assume an MSS of 1,460 bytes (a common value for TCP when operating over an Ethernet). What is the maximum value of *F* such that TCP sequence numbers are not exhausted? For this value of *F*, how long would it take to transfer the file from a server to a client? Assume 66 bytes of header (for TCP and all lower-level protocols) are added to each segment’s payload and that client and server are connected via a 10 Mbps link. Assume further that flow and congestion control mechanisms do not inhibit the server’s pumping out segments as fast as it can.

2^32 bytes = 4GB

B = Fmax + h\*ceil(Fmax/MSS) = 2^32 + 66\*ceil(2^32/1460)

T = B\*8/(10\*10^6) ~= 3591s

1. Consider a modification to TCP’s additive-increase, multiplicative-decrease congestion control algorithm wherein a connection reduces the threshold by a constant amount upon detecting packet loss rather than by reducing the threshold to one-half of the current congestion window.
2. Would the resulting additive-increase additive-decrease algorithm retain the property that a set of connections sharing a link would receive roughly equal shares of the link’s capacity? Explain.
   1. No, when loss occurs, both connections will experience a loss event, and the connections will never achieve the intersection of ideal bandwidth allocation since the connection with the larger window will always have the larger window.
3. What if the connections reduced their thresholds by different factors? For example, if one connection reduced its threshold by twice as much as another connection, would the two connections be able to fairly share the capacity of the link?
   1. The equilibrium of the two connections would be a function of the ratio of the decreasing factors. The connection with the smaller decreasing factor would have a larger window size and use an unfair amount.
4. The analysis for dynamic congestion windows that we presented in class implicitly assumed that there was only a single network link between the sender and the receiver. Redo the latency analysis for sending an object of *O* bytes assuming that there are *T* links between sender and receiver. To simplify the analysis, assume all links have the same transmission speed and that there is no congestion in the network (*i.e.*, that there are no queuing delays at any network interconnection point). You may also ignore the processing overhead at each network interconnection point.

How long does the server spend waiting?

K = number of windows needed to send the object.

Q = number of windows transmitted before the server no longer stalls

1. Suppose that TCP increased its window size by two segments rather than one for each ACK received during slow-start. In this scenario the first window would consist of 1 segment, the second window would consist of 3 segments, the third window would consist of 9 segments, *etc*. For the variables *K*, *P*, and *Q*, used in the analysis of TCP latency:

*a*) Express *K* in terms of *O* and *S*.

*b*) Express *Q* in terms of RTT, *S*, and *R*.

*c*) Express the latency of a TCP connection in terms of *P*, *O*, *R*, and RTT.

Latency = min\_latency + stall time

Latency with fix-sized window = 2RTT + O/R +(k-1)(S/R+RTT-wS/R)

Latency under slow start = 2RTT + O/R +….

Stall time for kth window = 0 or (TS/R + RTT) – 2^(k-1)S/R

Total Stall Time = sum from k=1 to P (TS/R + RTT -2^(k-1)\*S/R)

Faster slow start

Min\_latency = 2RTT + O/R

Stall time for kth window = 0 or (Time for the return of the ACK for the first segment in the kth window) – (time to transmit the kth window)

Stall time for kth window = 0 or (S/R + RTT) – 3^(k-1)S/R

Total Stall Time = sum from k=1 to P (S/R + RTT -3^(k-1)\*S/R)

Total Stall Time = P (S/R + RTT) – (3^P-1)/2\*S/R

O/S = segments required to send an object

Q is the number of windows requires to send

\*\*\*Should be able to do latency analysis