Spring 2025, CS 3611: Computer Networks

Solution to Homework 3

Solution to problem 1

Yes, both segments will be directed to the same socket. For each received segment, at the socket interface, the operating system will provide the process with the source IP addresses to determine the origins of the individual segments.

Solution to problem 2

1) 4467*, 22

2) 5513*, 23

3) 22, 4467*

4) 23, 5513*

* Note here A and B can have any possible port numbers larger than 1024.

5)Yes.

6)No, it's impossible that A and B have the same port number.

Solution to problem 3

The 32-bit sequence number can have up to 2^{32} sequence numbers. So in TCP case, the whole sequence number space can send up to 2^{32} bytes before wrapping around.

a) If the line is 224-kbps, the time to transmit the 2^{32} bytes is: $2^{32} * 8/224k = 42.6$ hours

b) If the line is 20-Mbps, the time to transmit the 2^{32} bytes is: $2^{32} * 8/20M = 28.6$ minutes

c) If the line is 4-Gbps, the time to transmit the 2^{32} bytes is: $2^{32} * 8/4G = 8.59$ seconds

d) Suppose a packet can stay in Internet for 40 seconds at most, 32-bit sequence number is not enough for 4Gbps network, i.e., there might be two different TCP segment with same sequence number. To deal with this problem, TCP introduces a TIMESTAMP field in TCP option. Sequence number and Timestamp together uniquely identify a segment.

Solution to problem 4

Suppose packets n, n+1, and n+2 are sent, and that packet n is received and ACKed. If packets n+1 and n+2 are reordered along the end-to-end-path (i.e., are received in the order n+2, n+1) then the receipt of packet n+2 will generate a duplicate ack for n and would trigger a retransmission under a policy of waiting only for second duplicate ACK for retransmission. By waiting for a triple duplicate ACK, it must be the case that two packets after packet n are correctly received, while n+1 was not received. The designers of the triple duplicate ACK scheme probably felt that waiting for two packets (rather than 1) was the

right tradeoff between triggering a quick retransmission when needed, but not retransmitting prematurely in the face of packet reordering.

Solution to problem 5

a) Consider sending an application message over a transport protocol. With TCP, the application writes data to the connection send buffer and TCP will grab bytes without necessarily putting a single message in the TCP segment; TCP may put more or less than a singe message in a segment. UDP, on the other hand, encapsulates in a segment whatever the application gives it; so that, if the application gives UDP an application message, this message will be the payload of the UDP segment. Thus, with UDP, an application has more control of what data is sent in a segment.

b) With TCP, due to flow control and congestion control, there may be significant delay from the time when an application writes data to its send buffer until when the data is given to the network layer. UDP does not have delays due to flow control and congestion control.

Solution to problem 6

a)

1. GoBackN:

   A sends 10 segments in total. They are initially sent segments 1, 2, 3, 4, 5, 6 and later re-sent segments 3, 4, 5 and 6.

   B sends 9 ACKs. They are ACKS with sequence numbers 1, 2, 2, 2, 2, and then ACKS with sequence numbers 3, 4, 5, 6.

2. Selective Repeat:

   A sends 7 segments in total. They are initially sent segments 1, 2, 3, 4, 5, 6 and later re-sent segments 3.

   B sends 6 ACKs. They are 5 ACKS with sequence number 1, 2, 4, 5, 6. And there is one ACK with sequence number 3.

3. TCP:

   A sends 7 segments in total. They are initially sent segments 1, 2, 3, 4, 5, 6 and later re-sent segments 3.

   B sends 6 ACKs. They are 5 ACKS with sequence number 2, 3, 3, 3, 3, and one ACK with sequence numbers 7. Note that TCP always send an ACK with expected sequence number.

b) TCP. This is because TCP uses fast retransmit without waiting until time out.

Solution to problem 7

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

b) After the 16th transmission round, packet loss is recognized by a triple duplicate ACK. If there was a timeout, the congestion window size would have dropped to 1.

c) After the 22nd transmission round, segment loss is detected due to timeout, and hence the congestion window size is set to 1.

d) The threshold is initially 32, since it is at this window size that slow start stops and congestion avoidance begins.

e) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 16, the congestion windows size is 42. Hence the threshold is 21 during the 18th transmission round.

f) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 22, the congestion windows size is 29. Hence the threshold is 14 during the 23th transmission round.

g) During the 1st transmission round, packet 1 is sent; packet 2-3 are sent in the 2nd transmission round; packets 4-7 are sent in the 3rd transmission round; packets 8-15 are sent in the 4th transmission round; packets 16-31 are sent in the 5th transmission round; packets 32-63 are sent in the 6th transmission round; packets 64 – 96 are sent in the 7th transmission round. Thus packet 80 is sent in the 7th transmission round.

h) The threshold will be set to half the current value of the congestion window (8) when the loss occurred. Thus the new values of the threshold and will be 4. The congestion window size is 4+3=7.

i) Threshold is 21, and congestion window size of 17th round is reset 1 then do slow start, so the congestion window size of 18th round is 2.

Solution to problem 8 In this problem, there is no danger in overflowing the receiver since the receiver's receive buffer can hold the entire file. Also, because there is no loss and acknowledgements are returned before timers expire, TCP congestion control does not throttle the sender. However, the process in host A will not continuously pass data to the socket because the send buffer will quickly fill up. Once the send buffer becomes full, the process will pass data at an average rate or R << S.