Computer Networking

Assignment 2

Problem 1: UDP (20 points)

UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, 01110100. What is the 1s complement of the sum of these 8-bit bytes? (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work. Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum? With the 1s complement scheme, how does the receiver detect errors? Is it possible that a 1-bit error will go undetected? How about a 2-bit error?

Solution 1:

Note, wrap around if overflow.

$$\begin{array}{r} 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0 \\ \hline 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1 \end{array} \qquad \begin{array}{r} 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1 \\ +\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0 \\ \hline 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \end{array}$$

One's complement = 1 1 0 1 0 0 0 1.

To detect errors, the receiver adds the four words (the three original words and the checksum). If the sum contains a zero, the receiver knows there has been an error. All one-bit errors will be detected, but two-bit errors can be undetected (e.g., if the last digit of the first word is converted to a 0 and the last digit of the second word is converted to a 1)

Problem 2: Reliable Data Transfer (10 points)

Consider our motivation for correcting protocol rdt2.1. Show that the receiver, shown in Figure 1, when operating with the sender shown in Figure 2, can lead the sender and receiver to enter into a deadlock state, where each is waiting for an event that will never occur.

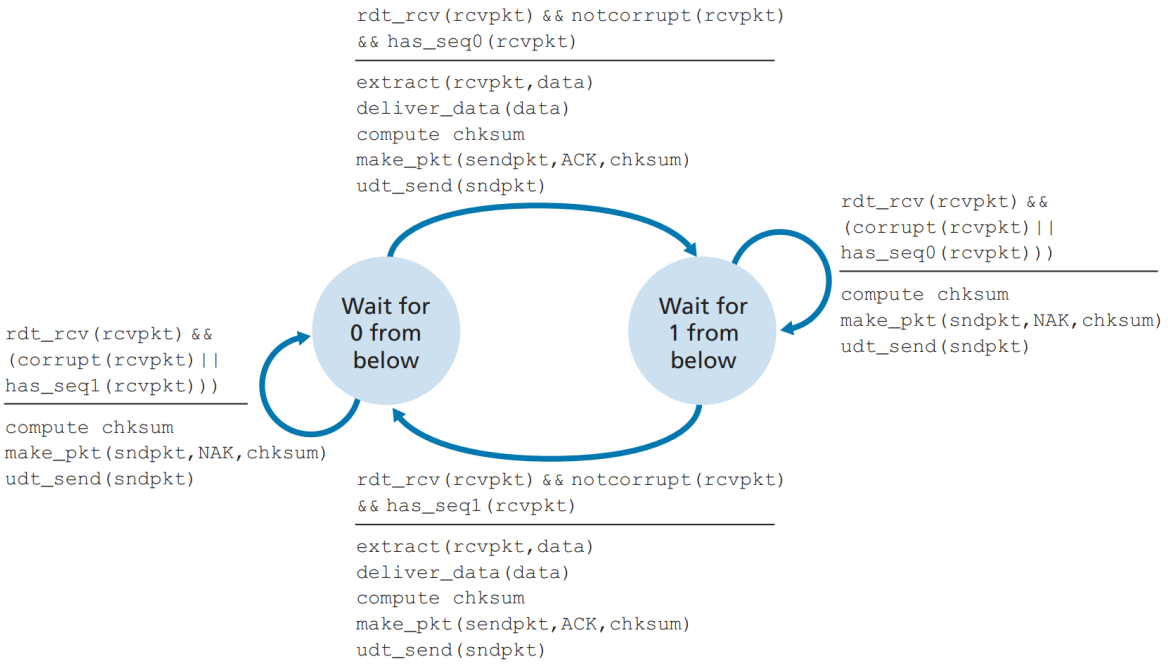

Figure 1: An incorrect receiver for protocol rdt 2.1

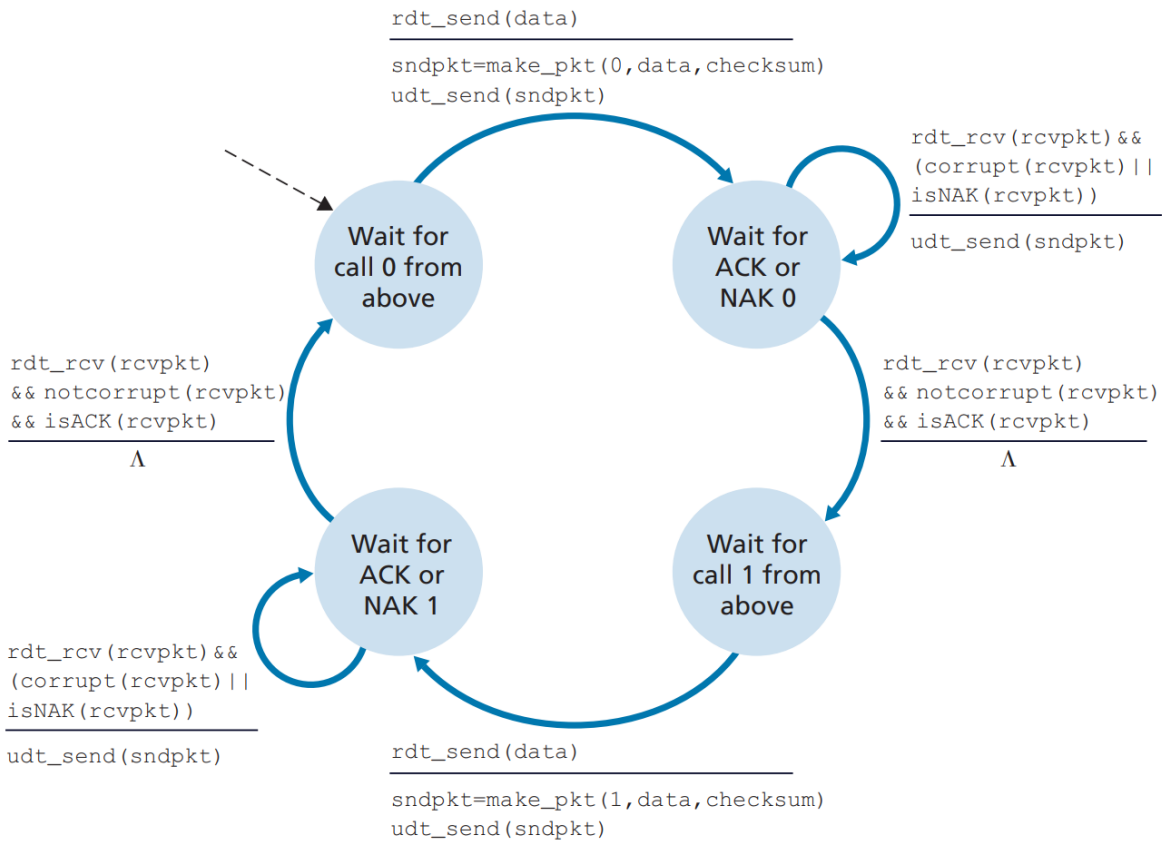

Figure 2: rdt2.1 sender

Solution 2:

Suppose the sender is in state “Wait for call 1 from above” and the receiver (the receiver shown in the homework problem) is in state “Wait for 1 from below.” The sender sends a packet with sequence number 1, and transitions to “Wait for ACK or NAK 1,” waiting for an ACK or NAK. Suppose now the receiver receives the packet with sequence number 1 correctly, sends an ACK, and transitions to state “Wait for 0 from below,” waiting for a data packet with sequence number 0. However, the ACK is corrupted. When the rdt2.1 sender gets the corrupted ACK, it resends the packet with sequence number 1. However, the receiver is waiting for a packet with sequence number 0 and (as shown in the home work problem) always sends a NAK when it doesn't get a packet with sequence number 0. Hence the sender will always be sending a packet with sequence number 1, and the receiver will always be NAKing that packet. Neither will progress forward from that state.

Problem 3: Pipelining (20 points)

Consider the GBN protocol with a sender window size of 4 and a sequence number range of 1,024. Suppose that at time *t*, the next in-order packet that the receiver is expecting has a sequence number of *k*. Assume that the medium does not reorder messages. Answer the following questions:

1. What are the possible sets of sequence numbers inside the sender's window at time *t*? Justify your answer.
2. What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time *t*? Justify your answer

Solution 3:

(1) Here we have a window size of N=4. Suppose the receiver has received packet k-1, and has ACKed that and all other preceding packets. If all of these ACK's have been received by sender, then sender's window is [k, k+3]. Suppose next that none of the ACKs have been received at the sender. In this second case, the sender's window contains k-1 and the 4 packets up to and including k-1. The sender's window is thus [k-4, k-1]. By these arguments, the senders window is [k-4, k+3]

(2) If the receiver is waiting for packet k, then it has received (and ACKed) packet k-1 and the 3 packets before that. If none of those N ACKs have been yet received by the sender, then ACK messages with values of [k-4,k-1] may still be propagating back.

Problem 4: TCP (20 points)

Compare GBN, SR, and TCP (no delayed ACK). Assume that the timeout values for all three protocols are sufficiently long such that five consecutive data segments and their corresponding ACKs can be received (if not lost in the channel) by the receiving host (Host B) and the sending host (Host A) respectively. Suppose Host A sends five data segments to Host B, and the second segment (sent from A) is lost. In the end, all five data segments have been correctly received by Host B.

1. How many segments has Host A sent in total and how many ACKs has Host B sent in total? What are their sequence numbers? Answer this question for all three protocols
2. If the timeout values for all three protocol are much longer than 5 RTT, then which protocol successfully delivers all five data segments in shortest time interval?

Solution 4:

(1) GoBackN:

A sends 9 segments in total. They are initially sent segments 1, 2, 3, 4, 5 and later resent segments 2, 3, 4, and 5.

B sends 8 ACKs. They are 4 ACKS with sequence number 1, and 4 ACKS with sequence numbers 2, 3, 4, and 5.

Selective Repeat:

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

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

TCP:

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

B sends 5 ACKs. They are 4 ACKS with sequence number 2. There is one ACK with sequence numbers 6. Note that TCP always send an ACK with expected sequence number.

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

Problem 5: Congestion Control (30 points)

Consider Figure 3. 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.

1. Identify the intervals of time when TCP slow start is operating.
2. Identify the intervals of time when TCP congestion avoidance is operating.
3. After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
4. After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
5. What is the initial value of ssthresh at the first transmission round?
6. What is the value of ssthresh at the 18th transmission round?
7. What is the value of ssthresh at the 24th transmission round?
8. During what transmission round is the 70th segment sent?
9. 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?
10. 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?
11. 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?

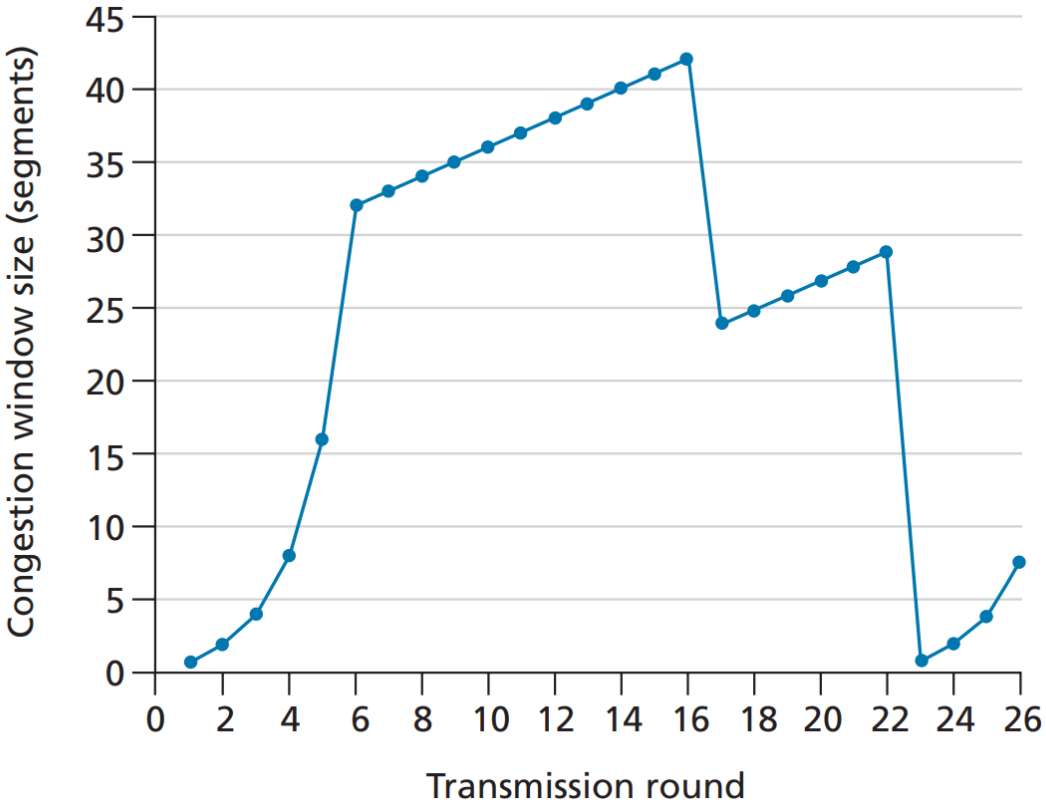

Figure 3: TCP window size as a function of time

Solution 5:

(1) TCP slowstart is operating in the intervals [1,6] and [23,26].

(2) TCP congestion avoidance is operating in the intervals [6,16] and [17,22].

(3) 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.

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

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

(6) 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.

(7) 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 (taking lower floor of 14.5) during the 24th transmission round.

(8) During the 1st transmission round, packet 1 is sent; packet 2-3 are sent in the 2^{nd} 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 70 is sent in the 7th transmission round.

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

(10) threshold is 21, and congestion window size is 4.

(11) round 17, 1 packet; round 18, 2 packets; round 19, 4 packets; round 20, 8 packets; round 21, 16 packets; round 22, 21 packets. So, the total number is 52.