Manish Yadav
3836-6483
m.yadav@ufl.edu

Homework 3

CNT5106C : Fall 2020 : Dr. Ye Xia
Due Fri, Oct 9, 2020

Contents

Problem 1

P5

Suppose that the UDP receiver computes the Internet checksum for the received UDP segment and finds that it matches the value carried in the checksum field. Can the receiver be absolutely certain that no bit errors have occurred? Explain.

Answer

No. The receiver cannot be certain that no bit errors have occurred.

It may be possible that both data and checksum have been corrupted that could result in checksum matched at the receiving end [2].

Also, 2-bit error changes in different word (i.e $1 \to 0$ and $0 \to 1$) will not change the sum. Because checksum is merely an addition. Consider two bytes (trimmed down example of WORD) 00000101 and 00000011, this would result in a sum of 00001000. But if 1 bit from each of the bytes is flipped, i.e 00000111 and 00000001 would result in a sum of 00001000 which is the same.

It could also fail to detect if there is a transposition of the word i.e. 0011 and 0101 is being transmitted instead of 0101 and 0011

Problem 2

P6

Consider our motivation for correcting protocol `rdt2.1`. Show that the receiver, shown in **Figure 3.57**, when operating with the sender shown in **Figure 3.11**, can lead the sender and receiver to enter into a deadlock state, where each is waiting for an event that will never occur

Answer

The real problem lies with the fact that the sender has to receive an acknowledgment from the receiver in order to send the next packet.

To best illustrate this scenario, consider a scenario where the receiver received a packet, sent an ACK, and now has moved to wait for another packet 0. However, the ACK sent is corrupted and the sender sends back packet 1 again. Now, both the sender and receiver have entered a state where they keep sending packet 1 and the receiver keeps sending NACK since it is now waiting for packet 0 to arrive.

Problem 3

P12

The sender side of `rdt3.0` simply ignores (that is, takes no action on) all received packets that are either in error or have the wrong value in the `acknum` field of an acknowledgment packet. Suppose that in such circumstances, `rdt3.0` were simply to retransmit the current data packet. Would the protocol still work? (Hint: Consider what would happen if there were only bit errors; there are no packet losses but premature timeouts can occur. Consider how many times the n^{th} packet is sent, in the limit as n approaches infinity.)

Answer

Yes. The protocol would still work. There would be packet re-transmission in case if the ACK was completely lost or had errors.

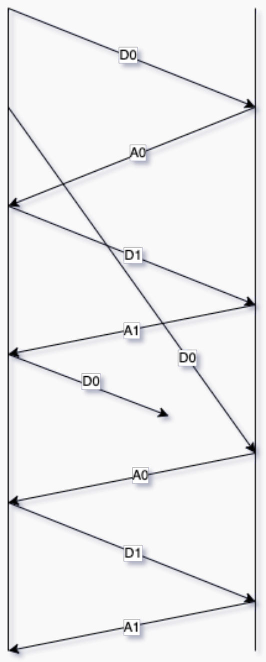
Consider a case where packet 1 was sent. In case of premature timeout, an extra copy would be sent that would be ACK'd by the receiver. This would cause the sender to send an extra copy of packet 2. This extra copy would be ACK'd along with the previous one causing packet 3 to be sent twice. This process continues to repeat on and on. Since n approaches infinity, the number of the packet sent would tend towards infinity as well.

Problem 4

P13

Consider the `rdt 3.0` protocol. Draw a diagram showing that if the network connection between the sender and receiver can reorder messages (that is, that two messages propagating in the medium between the sender and receiver can be reordered), then the alternating-bit protocol will not work correctly (make sure you clearly identify the sense in which it will not work correctly). Your diagram should have the sender on the left and the receiver on the right, with the time axis running down the page, showing data (D), and acknowledgment (A) message exchange. Make sure you indicate the sequence number associated with any data or acknowledgment segment.

Answer



`rdt 3.0` works correctly under the assumption of a FIFO based channel. In case of packet reordering, it causes `rdt 3.0` to deliver incorrect data. As in the above case, the older of D0 gets accepted.

Problem 5

P15

Consider the cross-country example shown in **Figure 3.17** . How big would the window size have to be for the channel utilization to be greater than 98 percent? Suppose that the size of a packet is 1,500 bytes, including both header fields and data.

Answer

$$d_{trans} = \frac{L}{R} = \frac{1500 * 8}{10^9} = 0.012ms$$

In order to get channel utilization,

$$\begin{aligned} U &= N * \frac{L/R}{RTT + L/R} = \frac{N * 0.012}{30 + 0.012} \\ \frac{98}{100} &= \frac{N * 0.012}{30.012} \\ N &= 2451 \end{aligned}$$

Therefore the window size should be at least 2451.

Problem 6

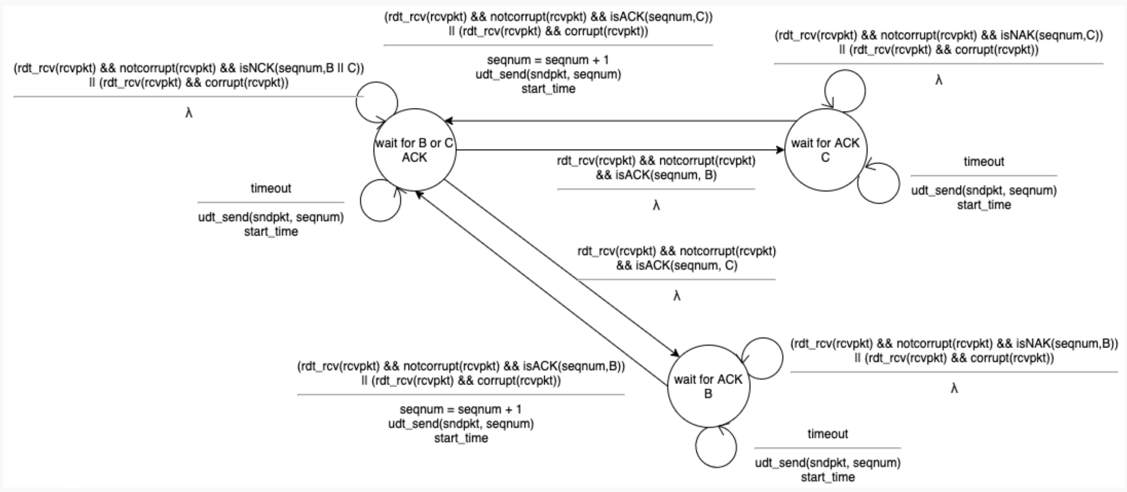
P19

Consider a scenario in which Host A wants to simultaneously send packets to Hosts B and C. A is connected to B and C via a broadcast channel—a packet sent by A is carried by the channel to both B and C. Suppose that the broadcast channel connecting A, B, and C can independently lose and corrupt packets (and so, for example, a packet sent from A might be correctly received by B, but not by C). Design a stop-and-wait-like error-control protocol for reliably transferring packets from A to B and C, such that A will not get new data from the upper layer until it knows that both B and C have correctly received the current packet. Give FSM descriptions of A and C. (Hint: The FSM for B should be essentially the same as for C.) Also, give a description of the packet format(s) used.

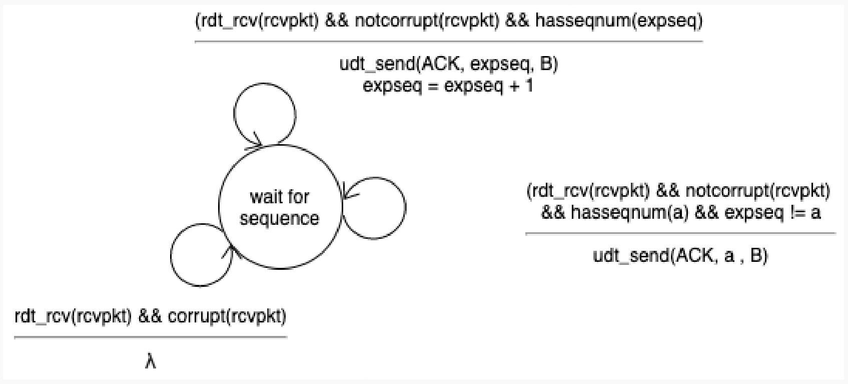
Answer

The below diagrams illustrates three host A, B and C. The first diagrams illustrates the sender state indicating where the channel may lose message on B/C or both. In order to make sure the receiver has received message we make use of sequence number that ensures that receiver does not get the same message again. Hence with sequence number, the sender mechanism should be able to work.

Sender


(rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(seqnum,C)) || (rdt_rcv(rcvpkt) && corrupt(rcvpkt))
seqnum = seqnum + 1
udt_send(sndpkt, seqnum)
start_time
(rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isNAK(seqnum,C)) || (rdt_rcv(rcvpkt) && corrupt(rcvpkt))
λ
(rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isNCK(seqnum,B || C)) || (rdt_rcv(rcvpkt) && corrupt(rcvpkt))
λ
wait for B or C ACK
wait for ACK C
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(seqnum, B)
λ
timeout
udt_send(sndpkt, seqnum)
start_time
timeout
udt_send(sndpkt, seqnum)
start_time
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(seqnum, C)
λ
(rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isNAK(seqnum,B)) || (rdt_rcv(rcvpkt) && corrupt(rcvpkt))
λ
(rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(seqnum,B)) || (rdt_rcv(rcvpkt) && corrupt(rcvpkt))
seqnum = seqnum + 1
udt_send(sndpkt, seqnum)
start_time
wait for ACK B
timeout
udt_send(sndpkt, seqnum)
start_time

Receiver


(rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && hasseqnum(expseq)
udt_send(ACK, expseq, B)
expseq = expseq + 1
wait for sequence
(rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && hasseqnum(a) && expseq != a
udt_send(ACK, a , B)
rdt_rcv(rcvpkt) && corrupt(rcvpkt)
λ

Problem 7

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:

P22 a

What are the possible sets of sequence numbers inside the sender's window at time t? Justify your answer.

Answer

At the time t, the receiver would get packet k, meaning packets from $[k-4...k-1]$ has been received already. If all of these packets are ACK'd then the window is $[k, k+3]$. Also packet k to packet $k+3$ maybe already sent from sender side.
Therefore sender's window is $[k-4, k]$. In case the beginning of the window is at k, then packet sequence will be from $[k, k+3]$

P22 b

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.

Answer

Same as above, ACK $k-5$ to ACK $k-1$ has already sent from receiver. Additionally, the receiver may be just received packet $k-1$ at time t, in this case, the previous ACKs are still propagating.

Problem 8

P23

Consider the GBN and SR protocols. Suppose the sequence number space is of size k. What is the largest allowable sender window that will avoid the occurrence of problems such as that in Figure 3.27 for each of these protocols?

Answer

The largest allowable sender window that will avoid the occurrence of the problem in Figure 3.27 would have to be less than or equal to half of the sequence number space size [1].

Problem 9

Answer true or false to the following questions and briefly justify your answer:

P24 a

With the SR protocol, it is possible for the sender to receive an ACK for a packet that falls outside of its current window

Answer

True. This is possible in case the receiver's ACKs are received after the sender is timed out. As the second group of ACKs for the would-be resent even if the window has advanced past those packets.

P24 b

With GBN, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

Answer

True. Same as P24 a.

P24 c

The alternating-bit protocol is the same as the SR protocol with a sender and receiver window size of 1.

Answer

True.

P24 d

The alternating-bit protocol is the same as the GBN protocol with a sender and receiver window size of 1.

Answer

True. A window size of 1 avoids the packet from being out of order.

Problem 10

Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A.

P27 a

In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?

Answer

Sequence number = 127+80 = 207
Source port number = 302
Destination port number = 80

P27 b

If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number, the source port number, and the destination port number?

Answer

Acknowledgment number = 207
Source port number = 80
Destination port number = 302

P27 c

If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?

Answer

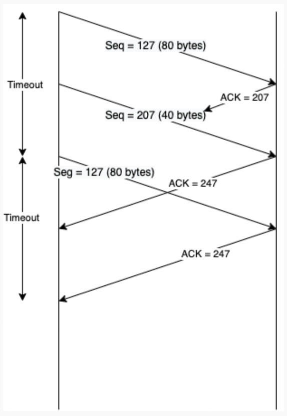
In the event of the second segment arriving before the first segment, the acknowledgment number would be 127 since it is missing a segment of 80 bytes which has to be delivered to maintain the order.

P27 d

Suppose the two segments sent by A arrive in order at B. The first acknowledgment is lost and the second acknowledgment arrives after the first timeout interval. Draw a timing diagram, showing these segments and all other segments and acknowledgments sent. (Assume there is no additional packet loss.) For each segment in your figure, provide the sequence number and the number of bytes of data; for each acknowledgment that you add, provide the acknowledgment number.

Problem 10 [P27 d] continued on next page. . .

Answer



Left side = Host A, Right side = Host B

References

[1] James Kurose and Keith Ross. Computer networks: A top down approach featuring the internet. *Peorsoim Addison Wesley*, 2010.

[2] Jonathan Stone and Craig Partridge. When the crc and tcp checksum disagree. *ACM SIGCOMM computer communication review*, 30(4):309–319, 2000.