Homework 3 **Zhanghao Wu (516030910593)**
CS391 Computer Networking ACM Class, Zhiyuan College, SJTU
Prof. **Yanmin Zhu** Due Date: Oct 25, 2018
TA. **Haobing Liu** Submit Date: October 24, 2018

P3

Solution: Firstly, we add up the first two 8-bit bytes and get 10111001. Then we add it to the third 8-bit byte (wrap around the overflow) and get 00101110. Therefore, the 1s complement of the sum of these 8-bit bytes is 11010001.

The reason why the UDP takes the 1s complement of the sum rather than just use the sum is that for the receiver, it can simply add up all of the things (bytes and the checksum) to detect an error.

To detect errors, the receiver checks the sum of the bytes and the checksum. If the sum contains zero bit, there must be errors.

All 1-bit errors will be detected, as the sum must be different from the origin one when only 1-bit changes.

However, 2-bit error may not be detected. For example, when the last bit of the first byte changes to 0 and the last bit of the second byte changes to 1, the sum of the three bytes will not change. Therefore, it cannot be detected.

P7

Solution: In the protocol rdt3.0, the packet contains a sequence number so that the receiver can tell if a data packet is a resent packet of an already received packet. However, for an ACK, the sender do need the sequence number of the packet the ACK is acknowledging to prevent the wrong behavior in *Premature timeout* situation caused by receiving a ACK for the former packet when a new packet is just sent, but the sequence number of the ACK itself is never needed, since the sender can easily detect the duplicate ACK by the sequence number of packet and ignore it.

On the other hand, if the maximum delay time is known and the timeout is not smaller than it, then the sequence of packet on the ACK will never be needed either. In this case, because of the stop-and-wait model, the ACK for the same packet will be received exactly once by the sender, no matter what exception occurred under the assumption of the protocol rdt3.0. Therefore, there is no need to find out which packet the ACK is acknowledging by the sequence number.

P10

Solution: Figure 1 shows the sender's finite-state machine of the modified protocol rdt2.1 including sender timeout and retransmit, where the timer's value is set to be larger than maximum delay. And the receiver, Figure 2, remains unchanged in the new protocol.

Now we show that why it works. There are two kind of packet loss: data packet loss and ACK/NAK loss.

1. **Data packet loss.** For a data packet loss, the receiver never receives that data packet and no ACK or NAK will be received by the sender. So the sender will timeout, and retransmit the data packet, just like what the states, 'Wait for ACK or NAK 0' and 'Wait

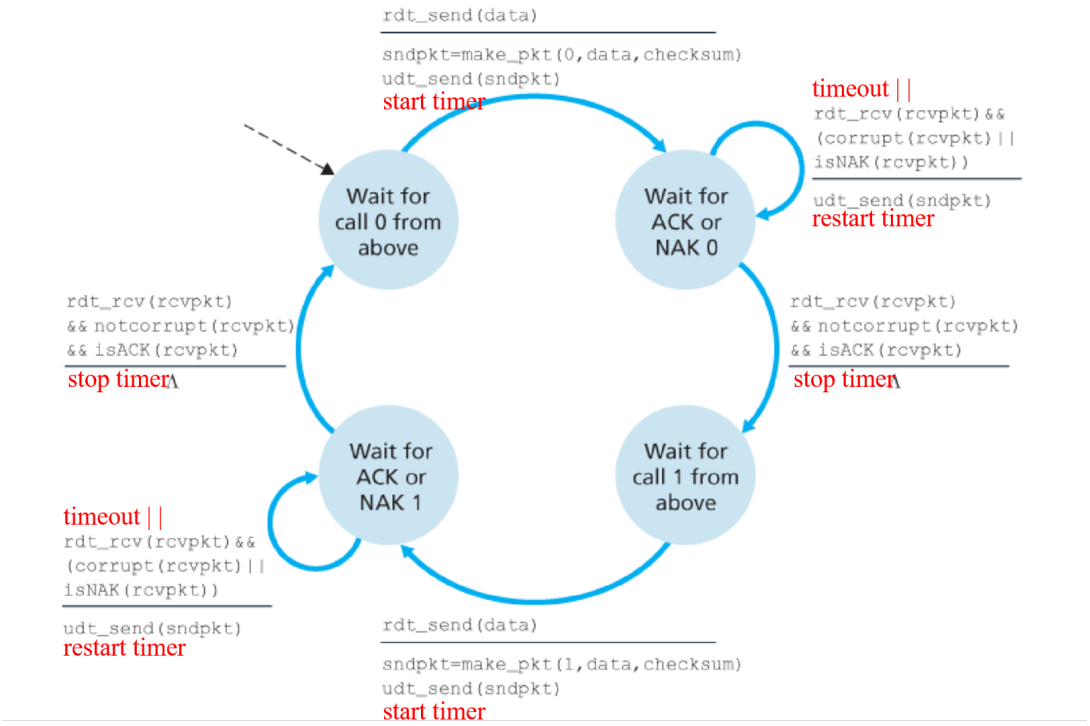

Figure 1: The the sender's FSM of modified rdt2.1

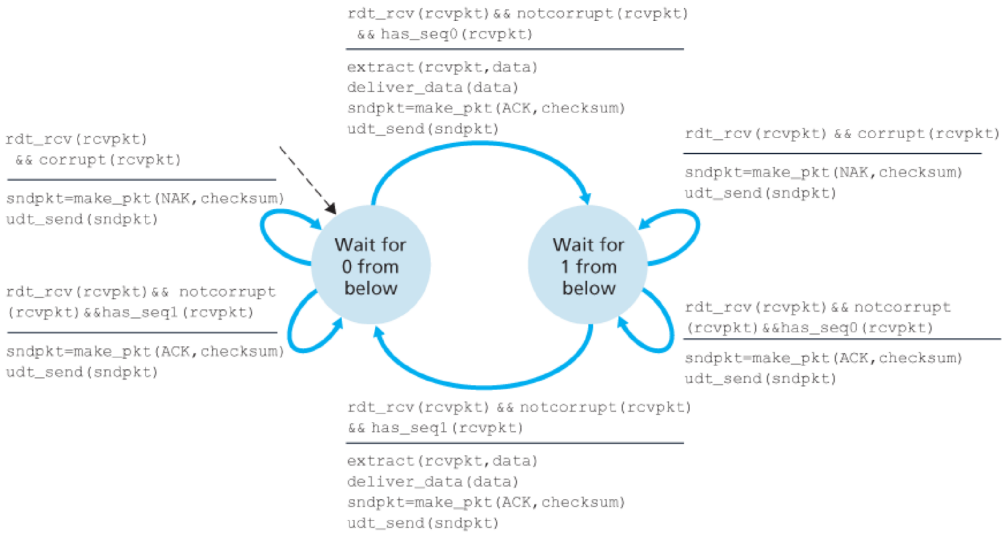

Figure 2: The the receiver's FSM of modified rdt2.1

for ACK or NAK 1' show. And now the situation is exactly the same as before the data loss happened.

2. **ACK/NAK loss.** For a ACK/NAK loss, the sender will wait until timeout and resend the data packet (without loss of generality assume that the packet sequence number is 0). If NAK is lost, the receiver is still waiting for the packet 0 in state 'Wait for 0 from below' and the sender will retransmit packet 0, which is same as the case where NAK is not lost. If ACK is lost, though the receiver has correctly received the packet 0, and move to the state 'Wait for 1 from below', the arrival of the duplicate data packet 0 will make the receiver send a ACK again. And now the situation is exactly the same as before the ACK loss.

P17

Solution: Figure 3 and Figure 4 shows the FSM for A and B in the protocol respectively. And the initial state of each entity is shown in black, i.e. A starts at 'Wait for call from above' and B starts at 'Wait for packet from A'.

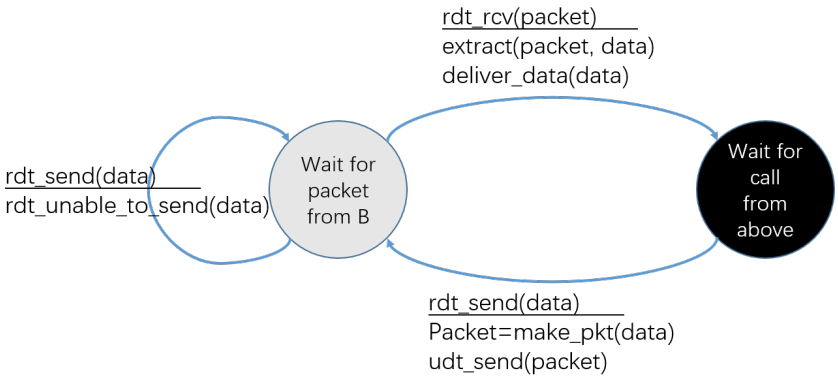

Figure 3: FSM for A

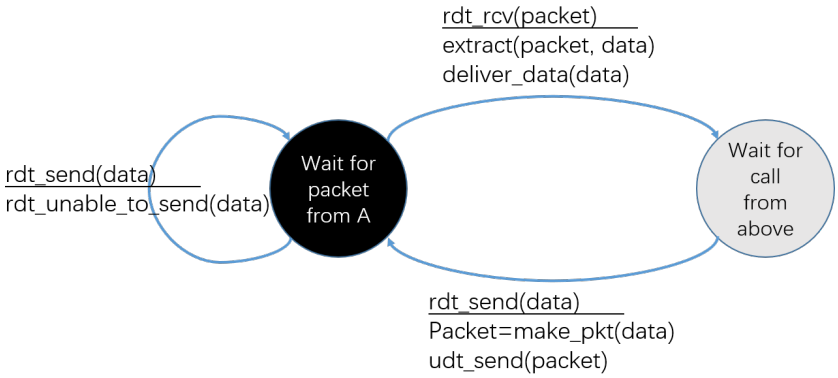

Figure 4: FSM for B

P33

Solution: For a retransmitted segment, there may be an delayed ACK of a same packet sent before the retransmission, so the SampleRTT measured may be much shorter than the correct

one.

P40

Solution:

a. The intervals of time when TCP slow start is operating are [1,6] and [23, 26]. In these intervals the MSS grows exponentially.

b. The intervals of time when TCP congestion avoidance is operating are [6,16] and [17, 22]. In these intervals the MSS grows linearly.

c. The segment loss is detected by a triple duplicate ACK, because if otherwise, the congestion window size will drop to 1.

d. The segment loss is detected by a timeout, because the congestion window size drop to 1.

e. The initial value of ssthresh at the first transmission round is 32, as the slow start stop at 32.

f. The value of ssthresh at the 18th transmission round is 21. Because each time the segment loss detected the ssthresh will be set to half of current congestion window size. And the former congestion window size exactly before the segment loss detected (16th transmission round) is 42.

g. The value of ssthresh at the 24th transmission round is 14. Because each time the segment loss detected the ssthresh will become the half of current congestion window size. And the former congestion window size exactly before the segment loss detected (22th transmission round) is 29.

h. During the 7th transmission round, the 70th segment is sent. Because segment 1 is sent in round 1, segment 2-3 are sent in round 2, segment 4-7 are sent in round 3, segments 8-15 are sent in round 4, segments 16-31 are sent in round 5, segment 32-63 are sent in round 6, and 64-96 are sent in round 7. Therefore, the 70th segment is sent in transmission round 7.

i. The value of the congestion window size is 8, according to the graph. The value of ssthresh is 4, as the ssthresh is set to half the current value of the congestion window when a packet loss is detected.

j. The value of the congestion window size at the 19th round will be 4 and the ssthresh is 21. As the congestion window size will drop to 1 when a packet loss detected and the ssthresh will be half in TCP Tahoe.

k. In TCP Tahoe the congestion window size will drop to 1 at 17th round and the ssthresh will be 21. Then 1 packet is sent in round 17th, 2 packets are sent in round 18th, 4 packets are sent in round 19th, 8 packets are sent in round 20th, 16 packets are sent in round 21th, 21 packets (meet the ssthresh) are sent in round 22th. Therefore, the total number of packets sent from 17th round to 22th round is 52.

P42

Solution: When discussing the doubling of the timeout interval, we were focusing on the stop-and-wait model which only send one packet at a round and doubling timeout interval is enough to deal with congestion. However, for TCP, it is a pipelining model, which can send a lot of packets at a round. Doubling the timeout interval can reduce the frequency for the sender to send packets, but it can still send a lot of packets at a round and make the network congested. Therefore, a window-based congestion-control mechanism is need in addition to the doubling-timeout-interval mechanism.

P50

Solution:

a. Table 1 shows the changes in the first 1000 msec of C_1's and C_2's congestion window sizes. Therefore C_1's and C_2's congestion window sizes are both 1 after 1000 msec.

Time (msec)	C1 segment window size	C1 average sending rate	C2 segment window size	C2 average sending rate
0	10	200	10	100
50	5	100		100
100	2	40	5	50
150	1	20		50
200	1	20	2	20
250	1	20		20
300	1	20	1	10
350	2	40		10
400	1	20	1	10
450	2	40		10
500	1	20	1	10
550	2	40		10
600	1	20	1	10
650	2	40		10
700	1	20	1	10
750	2	40		10
800	1	20	1	10
850	2	40		10
900	1	20	1	10
950	2	40		10
1000	1	20	1	10

Table 1: Changes of C_1's and C_2's congestion window size

b. In the long run these two connections will not get the same share of the bandwidth of the congested link. As C_1's RTT is half of C_2's, C_1's adjust its congestion window size twice as fast as C_2. C_1's average sending rate is 30 segments per second but C_2's is only 10 segments per second. Though they all experience segment loss, C_1 still has the larger bandwidth.