SYSC 5201 2019 Fall Assignment 3 Solution

1. 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:

   Note, wrap around if overflow.

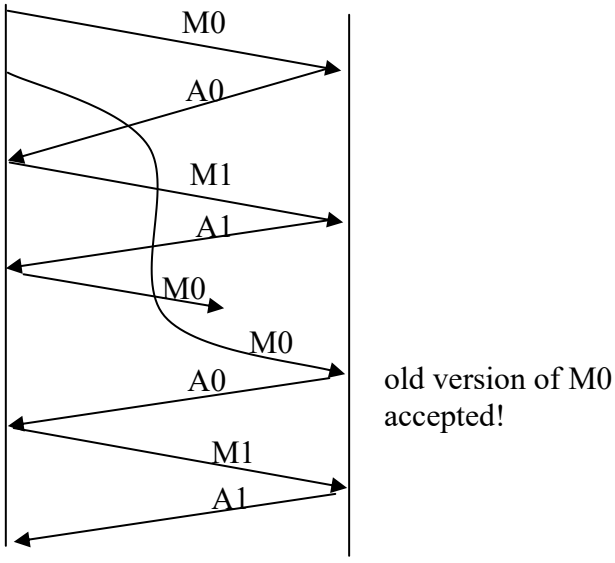
$$\begin{array}{rcccccccc} & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ + & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ \hline & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}$$

$$\begin{array}{rcccccccc} & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ + & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ \hline & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{array}$$

   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).

2. Consider the rdt 3.0 protocol. Draw a diagram (MSC) 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.

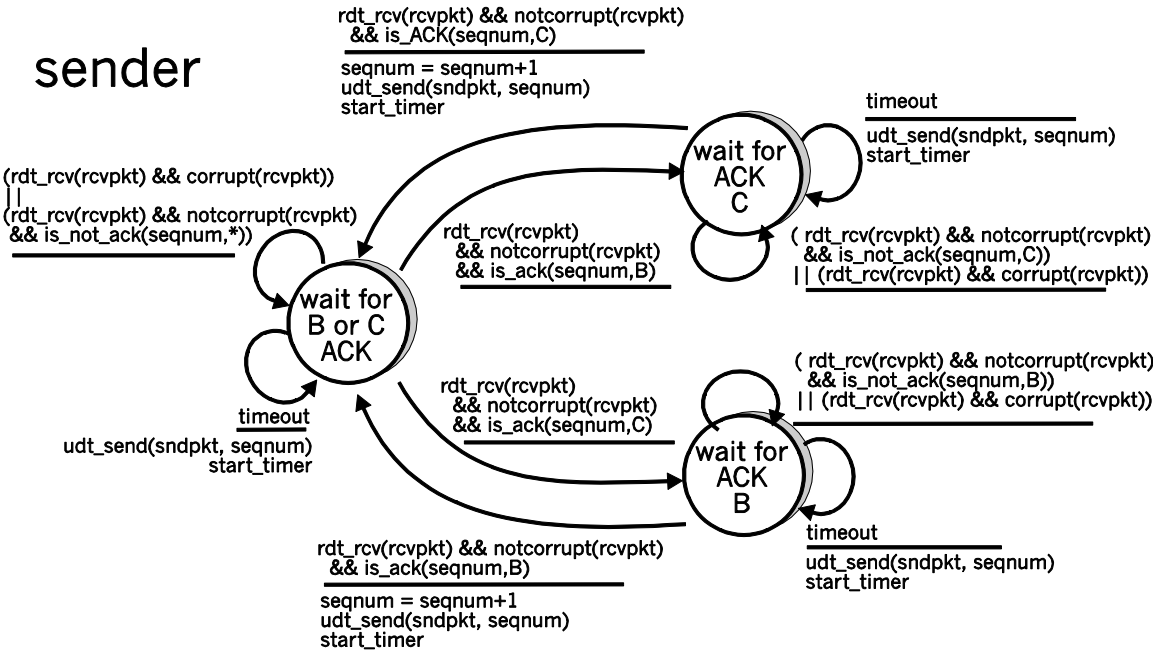
Solution:

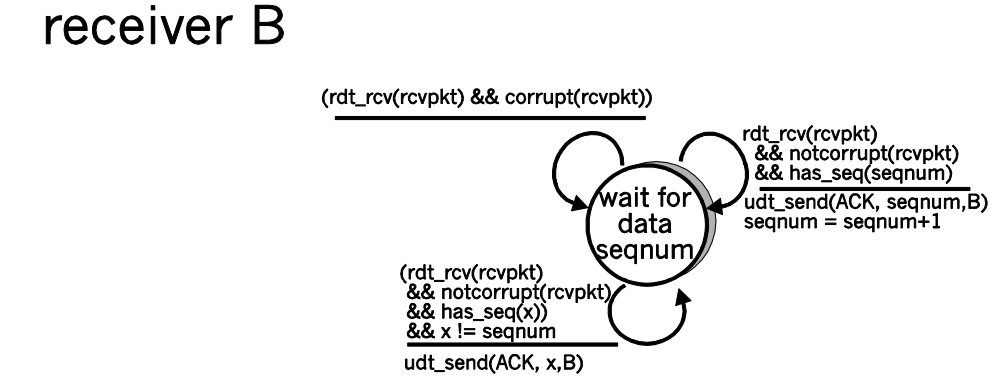


3. Consider a scenario in which Host A wants to simultaneously send packets to Hosts B and C. A is connected to B and C via a broadcast channel—a packet sent by A is carried by the channel to both B and C. Suppose that the broadcast channel connecting A, B, and C can independently lose and corrupt packets (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.

   Solution:

   This problem is a variation on the simple stop and wait protocol (rdt3.0). Because the channel may lose messages and because the sender may resend a message that one of the receivers has already received (either because of a premature timeout or because the other receiver has yet to receive the data correctly), sequence numbers are needed. As in rdt3.0, a 1-bit sequence number will suffice here. The sender and receiver FSM are shown in the following figure. In this problem, the sender state indicates whether the sender has received an ACK from B (only), from C (only) or from neither C nor B. The receiver state indicates which sequence number the receiver is waiting for.





4. Suppose that the five measured SampleRTT values are 106 ms, 120 ms, 140 ms, 90 ms, and 115 ms. Compute the EstimatedRTT after each of these SampleRTT values is obtained, using a value of $\alpha = 0.125$ and assuming that the value of EstimatedRTT was 100 ms just before the first of these five samples were obtained. Compute also the DevRTT after each sample is obtained, assuming a value of $\beta = 0.25$ and assuming the value of DevRTT was 5 ms just before the first of these five samples was obtained. Last, compute the TCP TimeoutInterval after each of these samples is obtained.

   Solution:

$$EstimatedRTT = xSampleRTT + (1-x)EstimatedRTT$$

$$DevRTT = y|SampleRTT - EstimatedRTT| + (1-y)DevRTT$$

$$TimeoutInterval = EstimatedRTT + 4 * DevRTT$$

After obtaining first sampleRTT is

$$EstimatedRTT = 0.125 * 106 + 0.875 * 100$$

$$= 100.75ms.$$

$$DevRTT = 0.25 * |106 - 100.75| + 0.75 * 5$$

$$= 5.06ms.$$

$$TimeoutInterval = 100.75 + 4 * 5.06$$

$$= 120.99ms.$$

After obtaining second sampleRTT = 120ms:

$$EstimatedRTT = 0.125 * 120 + 0.875 * 100.75$$

$$= 103.15ms.$$

$$DevRTT = 0.25 * |120 - 103.15| + 0.75 * 5.06$$

$$= 8ms.$$

$$TimeoutInterval = 103.15 + 4 * 8$$

$$= 135.15ms.$$

After obtaining Third sampleRTT = 140ms:

$$EstimatedRTT = 0.125 * 140 + 0.875 * 103.15$$

$$= 107.76ms.$$

$$DevRTT = 0.25 * |140 - 107.76| + 0.75 * 8$$

$$= 14.06ms.$$

$$TimeoutInterval = 107.76 + 4 * 14.06$$

$$= 164ms.$$

After obtaining fourth sampleRTT = 90ms:

$$EstimatedRTT = 0.125 * 90 + 0.875 * 107.76$$

$$= 105.54ms.$$

$$DevRTT = 0.25 * |90 - 105.54| + 0.75 * 14.06$$

$$= 14.42ms.$$

$$TimeoutInterval = 105.54 + 4 * 14.42$$

$$= 163.22ms.$$

After obtaining fifth sampleRTT = 115ms:

$$EstimatedRTT = 0.125 * 115 + 0.875 * 105.54$$

$$= 106.71ms.$$

$$DevRTT = 0.25 * |115 - 106.71| + 0.75 * 14.42$$

$$= 12.88ms.$$

$$TimeoutInterval = 106.71 + 4 * 12.88$$

$$= 158.23ms.$$

5. Recall the macroscopic description of TCP throughput. In the period of time from when the connection's rate varies from $W/(2 \cdot \text{RTT})$ to W/RTT, only one packet is lost (at the very end of the period).
    a. Show that the loss rate (fraction of packets lost) is equal to

$$L = \text{loss rate} = \frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}$$

    b. Use the result above to show that if a connection has loss rate L, then its average rate is approximately given by

$$\approx \frac{1.22 \cdot MSS}{RTT\sqrt{L}}$$

Solution:

a) The loss rate, L, is the ratio of the number of packets lost over the number of packets sent. In a cycle, 1 packet is lost. The number of packets sent in a cycle is

$$\frac{W}{2}+\left(\frac{W}{2}+1\right)+\cdots+W=\sum_{n=0}^{W/2}(\frac{W}{2}+n)$$

$$=\left(\frac{W}{2}+1\right)\frac{W}{2}+\sum_{n=0}^{W/2}n$$

$$=\left(\frac{W}{2}+1\right)\frac{W}{2}+\frac{W/2(W/2+1)}{2}$$

$$=\frac{W^2}{4}+\frac{W}{2}+\frac{W^2}{8}+\frac{W}{4}$$

$$=\frac{3}{8}W^2+\frac{3}{4}W$$

Thus the loss rate is

$$L=\frac{1}{\frac{3}{8}W^2+\frac{3}{4}W}$$

b) For W large, $\frac{3}{8}W^2>>\frac{3}{4}W$. Thus $L\approx 8/3W^2$ or $W\approx\sqrt{\frac{8}{3L}}$. From the text, we therefore have

$$\text{average throughput}=\frac{3}{4}\sqrt{\frac{8}{3L}}\cdot\frac{MSS}{RTT}$$

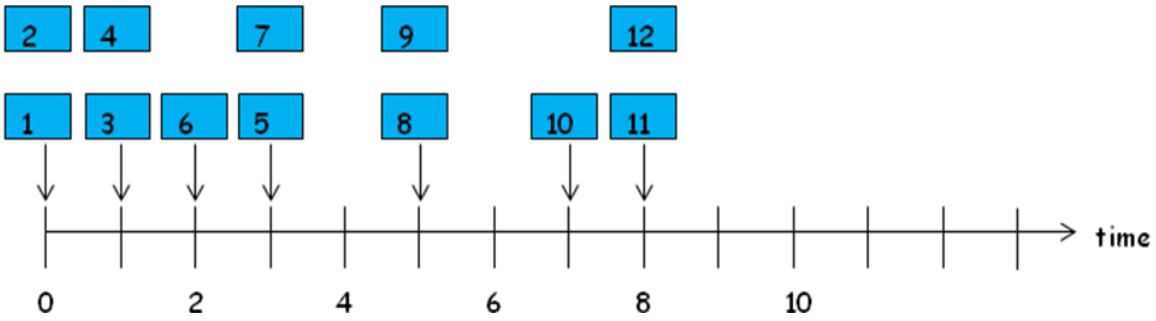
$$=\frac{1.22\cdot MSS}{RTT\cdot\sqrt{L}}$$

6. Consider a router that interconnects three subnets: Subnet 1, Subnet 2, and Subnet 3. Suppose all of the interfaces in each of these three subnets are required to have the prefix 223.1.17.0/24. Also suppose that Subnet 1 is required to support up to 63 interfaces, Subnet 2 is to support up to 95 interfaces, and Subnet 3 is to support up to 16 interfaces. Provide three network addresses (of the form a.b.c.d/x) that satisfy these constraints.

Solution:

223.1.17.0/25
223.1.17.128/25
223.1.17.64/27

7. Consider the following figure which indicates the arrival time of all the bits of each packet. Assume the transmission time of each packet be 1 unit time. Answer the following questions:
    a. Assuming FIFO service, indicate the time at which packets 1 through 12 each leave the queue. For each packet, what is the delay between its arrival and the beginning of the slot in which it is transmitted? What is the average of this delay over all 12 packets?
    b. Now assume a priority service, and assume that odd-numbered packets are high priority, and even-numbered packets are low priority. Indicate the time at which packets 1 through 12 each leaves the queue. For each packet, what is the delay between its arrival and the beginning of the slot in which it is transmitted? What is the average of this delay over all 12 packets?
    c. Now assume round robin service. Assume that packets 1, 2, 3, 6, 11, and 12 are from class 1, and packets 4, 5, 7, 8, 9, and 10 are from class 2. Indicate the time at which packets 1 through 12 each leaves the queue. For each packet, what is the delay between its arrival and the beginning of the slot in which it is transmitted? What is the average of this delay over all 12 packets?
    d. Now assume weighted fair queuing (WFQ) service. Assume that odd-numbered packets are from class 1, and even-numbered packets are from class 2. Class 1 has a WFQ weight of 2, while class 2 has a WFQ weight of 1. Indicate the time at which packets 1 through 12 each leaves the queue. For each packet, what is the delay between its arrival and the beginning of the slot in which it is transmitted? What is the average of this delay over all 12 packets?
    e. What do you notice about the average delay in all four cases discussed above?



Solution:

a)

Packet	Time leaving the queue	Delay
1	0	0
2	1	1
3	2	1
4	3	2
5	5	2
6	4	2
7	6	3
8	7	2
9	8	3
10	9	2

11	10	2
12	11	3
Average Delay		**1.91**

b)

Packet	Time leaving the queue	Delay
1	0	0
2	2	2
3	1	0
4	6	5
5	3	0
6	7	5
7	4	1
8	9	4
9	5	0
10	10	3
11	8	0
12	11	3
Average Delay		**1.91**

c)

Packet	Time leaving the queue	Delay
1	0	0
2	2	2
3	4	3
4	1	0
5	3	0
6	6	4
7	5	2
8	7	2
9	9	4
10	11	4
11	8	0
12	10	2
Average Delay		**1.91**

d)

Packet	Time leaving the queue	Delay	Note
1	0	0	WFQ
2	2	2	WFQ
3	1	0	WFQ
4	5	4	WFQ
5	3	0	WFQ
6	7	5	Idealized WFQ scheduling
7	4	1	WFQ
8	9	4	WFQ
9	6	1	Idealized WFQ scheduling
10	10	3	WFQ
11	8	0	WFQ

12	11	3	WFQ
Average Delay		**1.91**	

e) It can be noticed that the average delay for all four cases is the same (1.91 seconds).

8. Suppose a company have the numbers of users listed in the following table. The company owns the network address block 129.99.0.0/16. For minimizing the sizes of forwarding tables, the following conditions are imposed:
    a) Each AS can be identified by a single prefix;
    b) Each location can be identified by a single prefix;
    c) Each department can be identified by a single prefix;
    d) While satisfying the above three conditions, the block size allocated for each department should be minimized.

    Develop a variable-length addressing scheme that best fits the above requirements and provide all the prefixes used by each AS, location, and department. Also provide the associated block sizes for the prefixes used by all departments.

AS Number	Location	Department	Users
1	Chicago Campus Building 1	Legal	120
		Accounting	370
	Chicago Campus Building 2	HQ	1580
		Engineering	200
2	Toronto	Sales	75
	Boston	Sales	110
3	Philadelphia	Operations1	2150
		Operations2	975
		Sales	575

Solution:

1. 129.99.0/19,

1.1 129.99.0/22 (1024)

1.1.1 129.99.0.0/25 (128)

1.1.2 129.99.2/23 (512)

1.2 129.99.16/20 (4096)

1.2.1 129.99.16/20 (3584)

1.2.2 129.99.16/24 (256)

2. 129.99.17/24 (256)

2.1 129.99.17.0/25 (128)

2.2 129.99.17.128/25 (128)

3. 129.99.32/19

3.1 129.99.32/20 (4096)

3.2 129.99.48/22 (1024)

3.3 129.99.52/22 (1024)

9. RFC 1141 states that when an IP header with checksum *HC* is modified by changing some 16-bit field value (such as the TTL field) *m* to a new value *m*', then the new checksum should become *HC* + *m* + ~ *m*', where ~*X* denotes the 1's complement of *X*. This allows faster incremental checksum computation to avoid recomputing checksum completely. While this works most of the time, the right equation, described in RFC 1624, is to compute ~(~*HC* + ~ *m* + *m*'). This is slightly more inefficient but correct. To see the difference between these two implementations, consider an example given in RFC 1624 with an IP header in which a 16-bit field *m* = 0x5555 changes to *m*'= 0x3285. The 1's-complement sum of all the remaining header bytes is 0x*CD*7*A*. Compute the checksum both ways and show that they produce different results. Given that these two results are really the same in 1's complement notation (different representations of zero), why might it cause trouble at the receiver?

   Solution:

   Consider an IP packet header in which a 16-bit field *m* = 0x5555 changes to *m*' = 0x3285. Also, the one's complement sum of all other header octets is 0xCD7A.

   Then the header checksum would be:

   HC = ~ (0xCD7A + 0x5555)
   = ~0x22D0
   = 0xDD2F

   The new checksum via recomputation is:

   HC' = ~ (0xCD7A + 0x3285)
   = ~0xFFFF
   = 0x0000

   Using the equation as specified in RFC 1141, the new checksum is computed as:

   HC' = HC + m + ~m'
   = 0xDD2F + 0x5555 + ~0x3285
   = 0xFFFF

which does not match that computed from scratch.

Applying the new equation in RFC 1624 to the example above, we get the correct result:

HC' = ~ (C + ~m + m')
= ~ (0x22D0 + ~0x5555 + 0x3285)
= ~0xFFFF
= 0x0000

In one's complement, there are two representations of zero: the all zero and the all one bit values, often referred to as +0 and -0. One's complement addition of non-zero inputs can produce -0 as a result, but never +0. Since there is guaranteed to be at least one non-zero field in the IP header, and the checksum field in the protocol header is the complement of the sum, the checksum field can never contain ~(+0), which is -0 (0xFFFF). It can, however, contain ~ (-0), which is +0 (0x0000).