CS305: Computer Networking
2024 Fall Semester Written Assignment # 2

Q 1. The UDP checksum provides for error detection. Consider the following word with 32 bits:

$$10011001100111111010101010101010 \tag{1}$$

(a) Compute the checksum. (Recall that UDP computes checksum based on 16-bit word.)

(b) How does the receiver check whether the message was transmitted with errors?

(c) If the receiver does not detect any error using the checksum, does it mean that the message was transmitted without any error? Please explain the reason and provide an example.

Solution:

(a) Break the 32-bit word into two 16-bit words, and sum their up:

$$\begin{array}{rr} & 1001100110011111 \\ + & 1010101010101010 \\ \hline & 10100010001001001 \end{array} \tag{2}$$

Then, wrapping up the highest bit:

$$\begin{array}{rr} & 0100010001001001 \\ + & 1 \\ \hline & 0100010001001010 \end{array} \tag{3}$$

Compute the 1s complement of 0100010001001010. The checksum is 1011101110110101.

(b) Add up all 16-bit word, including the checksum. If the sum is equal to 1111111111111111, then no error detected.

(c) No. Consider the following example: the first two rows correspond to the original two 16-bit words, with bit flipping; the third row is the obtained checksum, with bit flipping. The flipped bits are marked in bold text:

$$\begin{array}{rr} & 1001100110011111 \\ & 1010101010101011 \\ + & 1011101110110100 \\ \hline & 11111111111111110 \end{array} \tag{4}$$

In this example, the lowest bit of the second 16-bit word is flipped from zero to one, and that of the checksum is flipped from one to zero. Then, wrapping up the highest bit, the sum of all 16-bit words of the received segment is still 1111111111111111. In this example, there exists bit error, but such an error cannot be detected.

Q 2. List three main differences between go-back-N and selective repeat.

Solution: Their main differences are listed as follows:
Go-back-N:

1. Cumulative ACK;
2. One timer for the oldest packet in the window;
3. When timer times out, the sender retransmits all un-ACKed packets in the window;
4. Receiver does not buffer the out-of-order packets;

5. ... (other reasonable answers).

Selective repeat:

1. Individual ACK;
2. One timer for each packet;
3. If a timer times out, the sender retransmits only the packet triggered the time out.
4. Receiver buffers the out-of-order packets;
5. ... (other reasonable answers).

Q 3. The following figure illustrates the convergence of TCP's additive-increase multiplicative-decrease (AIMD) algorithm.

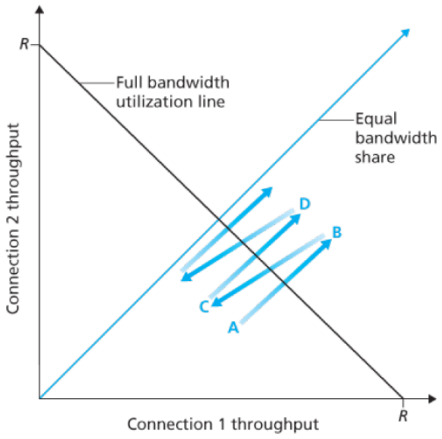
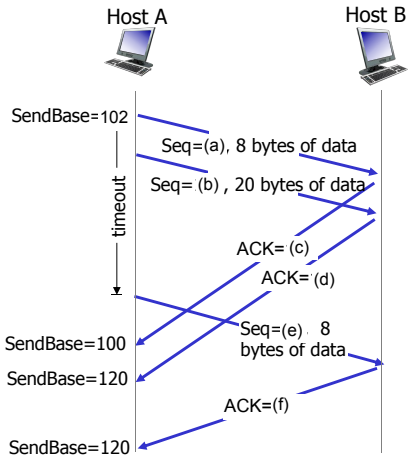


Figure 3.55 Throughput realized by TCP connections 1 and 2

Suppose that instead of a multiplicative decrease, TCP decreased the window size by a constant amount. Would the resulting additive-increase additive-decrease (AIAD) algorithm converge to an equal share algorithm? Justify your answer using a diagram similar to the above figure. (Note: Simply draw the diagram is not sufficient. You need to explain what the diagram shows.)

Solution: No. If the window size is decreased by a constant amount, then at point B (i.e., when packet loss occurs), the window size will be reduced along the line defined by points A and B. As a result, the window size will keep remaining along the aforementioned line, which will not get closer to the equal bandwidth share line across time.

Q 4. Consider the following figure. Please fill in blanks (a)–(f).



Solution: (a) 102; (b) 110; (c) 110; (d) 130; (e) 102; (f) 130.

Q 5. Suppose hosts A and B has already established a TCP connection. The maximum segment size (MSS) is 2KB, and round-trip time (RTT) between A and B is 4ms. Suppose there is no congestion occurs. It is recently at congestion avoidance state. How long it takes for the congestion window to increase from 8KB to 32KB.

Solution: In congestion avoidance state, the window size increases 1 MSS per RTT. That is, the window size increases 2KB per 4ms, i.e., 0.5KB per ms. From 8KB to 32KB, the time it takes is equal to $(32KB - 8KB)/0.5KB/ms = 48ms$.

Q 6. Consider a datagram network using 8-bit host addresses. Suppose a router has four links, numbered 0 through 3, and uses longest prefix matching. It has the following forwarding table:

Prefix Match	Interface
1110****	0
111000**	1
111111**	2
otherwise	3

Consider the longest prefix matching, for each interface,

a) provide the range of destination host address that will be forwarded to each interface, i.e., fill in (A), (B), (C), (D);

b) provide the number of addresses in each range, i.e., fill in x, y, z, w.

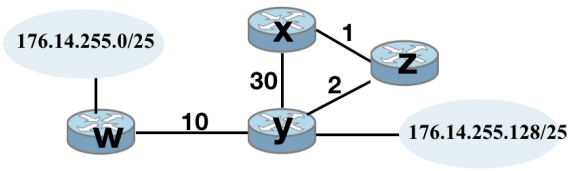
Note: you do NOT need to remove the network address, i.e., the address with "*"'s all being zeros. In other words, (A), (B), (C), (D) should cover the entire address range, and $x + y + z + w$ should be equal to 2^8.

Prefix	Range of destination host addresses	Number of addresses in the range
1110****	(A)	x
111000**	(B)	y
111111**	(C)	z
otherwise	(D)	w

Solution:

Prefix	Range of destination host addresses	Number of addresses in the range
1110****	11100100 through 11101111	$2^4 - 2^2 = 12$
111000**	11100000 through 11100011	$2^2 = 4$
111111**	11111100 through 11111111	$2^2 = 4$
otherwise	00000000 through 11011111 11110000 through 11111011	$(2^3 - 1)2^5 = 224$ $2^3 + 2^1 + 2^0 + 1 = 12$ $224 + 12 = 236$ or $2^8 - 4 - 4 - 12 = 236$

Q 7. Consider a network as shown in the figure below. Distance vector algorithm is used to calculate the forwarding table.



a) Draw the distance vector table of node z after convergence WITHOUT poisoned reverse. You do not need to draw those tables before convergence.

b) Draw the distance vector table of node z after convergence WITH poisoned reverse.

c) Based on b), draw the forwarding table of node z that contains two entries related to network prefix 176.14.255.0/25 and 176.14.255.128/25. Note: To denote an interface, you can use notation (a,b) to indicate the interface of a connected to b.

d) Based on c), can the two entries be combined into one entry? If yes, what is the combined entry?

e) Consider the case WITHOUT poisoned reverse. Suppose $cost(y, z) = cost(z, y)$ has been changed to 100. (i) Which node(s) will update their distance vector table(s) immediately after the link change? (ii) What is the updated table of node z in the first iteration? (iii) Use this example to explain why "bad news travels slowly".

Solution:

a) The distance vector table of node z after convergence without poisoned reverse:

Node z					
		Cost to			
		w	x	y	z
from	x	13	0	3	1
	y	10	3	0	2
	z	12	1	2	0

b) The distance vector table of node z after convergence with poisoned reverse:

Node z					
		Cost to			
		w	x	y	z
from	x	∞	0	∞	1
	y	10	∞	0	2
	z	12	1	2	0

c) The forwarding table of node z

Destination	Interface
176.14.255.0/25	(z,y)
176.14.255.128/25	(z,y)

d) Yes; Destination: 176.14.255.0/24; Inferface (z,y) .

e) Node y and z will update their distance vector tables. The updated table of node y:

Node z					
		Cost to			
		w	x	y	z
from	x	13	0	3	1
	y	10	3	0	2
	z	14	1	4	0

As shown in this example, z believes x has a better way to w and y, so it decides to forward the data destinated towards w and y to x as the next hope. However, such a believe is incorrect, and it takes iterations for z to notice the fact that x does not have a better way to w and y.