COMPUTER NETWORKS

Question Bank

Unit – 3
Transport Layer

1. Consider the scenario below:
   Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789.

   Questions:

   a. Will both of these segments be directed to the same socket at Host C?
   b. If so, how will the process a Host C know that these two segments originated from two different hosts?

   Answer:

   Yes, both segments will be directed to the same socket. For each received segment, at the socket interface, the operating system will provide the process with the IP addresses to determine the origins of the individual segments.

2. Consider the scenario below:
   Suppose that a Web server runs in Host C on port 80. Suppose this Web server uses persistent connections, and is currently receiving requests from two different Hosts, A and B.

   Questions:

   a. Are all of the requests being sent through the same socket at Host C?
   b. If they are being passed through different sockets, do both sockets have port 80? Discuss and explain.

   Answer:

   For each persistent connection, the Web server creates a separate “connection socket”. Each connection socket is identified with a four-tuple: (source IP address, source port number, destination IP address, destination port number). When host C receives and IP datagram, it examines these four fields in the datagram/segment to determine to which socket it should pass the payload of the TCP segment. Thus, the requests from A and B pass through different sockets. The identifier for both of these sockets has 80 for the destination port; however, the identifiers for these sockets have different values for source IP addresses. Unlike UDP, when the transport layer passes a TCP segment’s payload to the application process, it does

not specify the source IP address, as this is implicitly specified by the socket identifier.

---

3. Consider the scenario below:

   Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has sequence number 90; the second has sequence number 110.

   Questions:

   a. How much data is in the first segment?
   b. Suppose that the first segment is lost but the second segment arrives at B. In the acknowledgment that Host B sends to Host A, what will be the acknowledgment number?

   Answer:

   a) 20 bytes b) ack number = 90

---

4. Consider the Telnet example , A few seconds after the user types the letter ‘C,’ the user types the letter ‘R.’ After typing the letter ‘R,’

   Questions:

   a. how many segments are sent,
   b. what is put in the sequence number and acknowledgment fields of the segments?

   Answer: 3 segments. First segment: seq = 43, ack =80; Second segment: seq = 80, ack = 44; Third segment; seq = 44, ack = 81

---

5. Consider the scenario below:

   Suppose two TCP connections are present over some bottleneck link of rate R bps. Both connections have a huge file to send (in the same direction over the bottleneck link). The transmissions of the files start at the same time.

   Question:

   a. What transmission rate would TCP like to give to each of the connections?

Answer:

R/2

---

6. Consider the scenario below

   Suppose Client A initiates a Telnet session with Server S. At about the same time, Client B also initiates a Telnet session with Server S.

   Questions:

   Provide possible source and destination port numbers for

a. The segments sent from A to S.
b. The segments sent from B to S.
c. The segments sent from S to A.
d. The segments sent from S to B.
e. If A and B are different hosts, is it possible that the source port number in the segments from A to S is the same as that from B to S?
f. How about if they are the same host?

Answer:

	source port numbers	destination port
a)	A □□ S	23
	46	23
7 b)	B □□ S	46
	51	

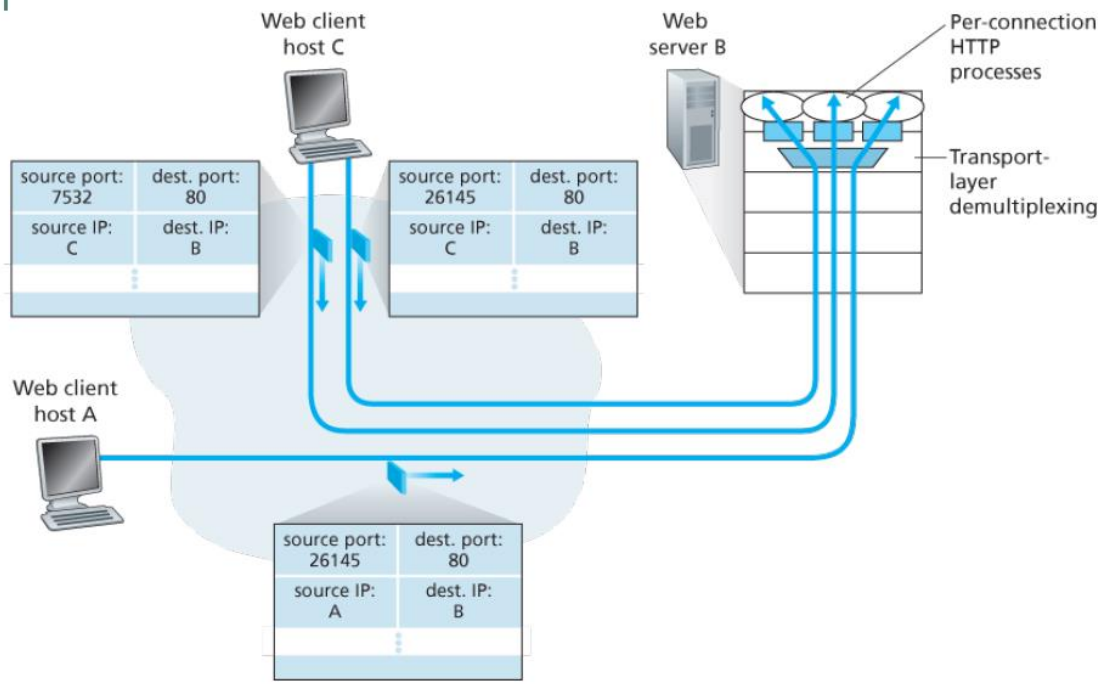
e) Yes.
f) No.

---

7. Consider the figure below,

   Question:

   a. what are the source and destination port values in the segments flowing from the server back to the clients' processes?
   b. What are the IP addresses in the network-layer datagrams carrying the transport-layer segments?



Answer:

Suppose the IP addresses of the hosts A, B, and C are a, b, c, respectively. (Note that a, b, c are distinct.)

To host A: Source port =80, source IP address = b, dest port = 26145, dest IP address = a

To host C, left process: Source port =80, source IP address = b, dest port = 7532, dest IP address = c

To host C, right process: Source port =80, source IP address = b, dest port = 26145, dest IP address = c

---

8. UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, 01110100.

Questions:

a. 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.
b. Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum?
c. With the 1s complement scheme, how does the receiver detect errors?
d. Is it possible that a 1-bit error will go undetected?

e. How about a 2-bit error?

Answer:

Note, wrap around if overflow.

```
    0  1  0  1  0  0  1  1
+   0  1  1  0  0  1  1  0
    1  0  1  1  1  0  0  1

    1  0  1  1  1  0  0  1
+   0  1  1  1  0  1  0  0
    0  0  1  0  1  1  1  0
```

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).

---

9. Consider UDP and TCP Checksum, Solve the following

Questions:

a. Suppose you have the following 2 bytes: 01011100 and 01100101. What is the 1s complement of the sum of these 2 bytes?
b. Suppose you have the following 2 bytes: 11011010 and 01100101. What is the 1s complement of the sum of these 2 bytes?
c. For the bytes in part (a), give an example where one bit is flipped in each of the 2 bytes and yet the 1s complement doesn't change.

Answer:

a) Adding the two bytes gives 11000001. Taking the one's complement gives 00111110.

b) Adding the two bytes gives 01000000; the one's complement gives 10111111.

c) First byte = 01010100; second byte = 01101101.

10. Consider the scenario below

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.

Question:

Can the receiver be absolutely certain that no bit errors have occurred? Explain.

Answer:

No, the receiver cannot be absolutely certain that no bit errors have occurred. This is because of the manner in which the checksum for the packet is calculated. If the corresponding bits (that would be added together) of two 16-bit words in the packet were 0 and 1 then even if these get flipped to 1 and 0 respectively, the sum still remains the same. Hence, the 1s complement the receiver calculates will also be the same. This means the checksum will verify even if there was transmission error.

---

11. Consider the scenario below

In protocol rdt3.0, the ACK packets flowing from the receiver to the sender do not have sequence numbers (although they do have an ACK field that contains the sequence number of the packet they are acknowledging).

Question:

Why is it that our ACK packets do not require sequence numbers?

Answer:

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