Answers

1-A) false, this statement is true for UDP, but is false for TCP since It is important to note that a UDP socket is fully identified by a two-tuple consisting of a destination IP address and a destination port number but that a TCP socket is identified by a four-tuple: source IP address, source port number, destination IP address, destination port number.

B) True, according to the concept of non-persistent HTTP, in the interaction between client and server, each request/response pair should be sent over a *separate* TCP connection.

C)True.

D)False, Under UDP, as soon as an application process passes data to UDP, UDP will package the data inside a UDP segment and immediately pass the segment to the network layer. TCP, on the other hand, make use of mechanisms for reliable data transfer, congestion control and flow control.

E) true, a server devoted to a particular application can typically support many more active clients when the application runs over UDP rather than TCP.

F)True, that it *is* possible for an application to have reliable data transfer when using UDP. This can be done if reliability is built into the application itself .

G)Flase, the length field in each UDP segment specifies the number of bytes in the UDP segment.

H) False, If the window is full, the sender simply returns the data back to the upper layer, which is an implicit indication that the window is full.

I) True. Suppose the sender has a window size of 3 and sends packets 1, 2, 3 at $t0$. At $t1$ ($t1 > t0$) the receiver ACKS 1, 2, 3. At $t2$ ($t2 > t1$) the sender times out and resends 1, 2, 3. At $t3$ the receiver receives the duplicates and re-acknowledges 1, 2,3. At $t4$ the sender receives the ACKs that the receiver sent at $t1$ and advances its window to 4, 5, 6. At $t5$ the sender receives the ACKs 1, 2, 3 the receiver sent at $t3$. These ACKs are outside its window.

J) True. By essentially the same scenario as in (I).

K) True.

L) True. Note that with a window size of 1, SR, GBN, and the alternating bit protocol are functionally equivalent. The window size of 1 precludes the possibility of out-of-order packets (within the window). A cumulative ACK is just an ordinary ACK in this situation, since it can only refer to the single packet within the window.

2.

a) *Finer application-level control over what data is sent, and when.* Under UDP, as soon as an application process passes data to UDP, UDP will package the data inside a UDP segment and immediately pass the segment to the network layer. TCP, on the other hand, has a congestion-control mechanism that throttles the transport-layer TCP sender when one or more links between the source and destination hosts become excessively congested.

No connection establishment. As we'll discuss later, TCP uses a three-way handshake before it starts to transfer data. UDP just blasts away without any formal preliminaries.

No connection state. TCP maintains connection state in the end systems. This connection state includes receive and send buffers, congestion-control parameters, and sequence and acknowledgment number parameters.

Small packet header overhead. The TCP segment has 20 bytes of header overhead in every segment, whereas UDP has only 8 bytes of overhead.

b) The reason is that there is no guarantee that all the links between source and destination provide error checking; that is, one of the links may use a link-layer protocol that does not provide error checking. Furthermore, even if segments are correctly transferred across a link, it's possible that bit errors could be introduced when a segment is stored in a router's memory. Given that neither link-by-link reliability nor in-memory error detection is guaranteed, UDP or TCP must provide error detection at the transport layer, *on an End-end basis*.

3.

1110011001100110 0110011001100000

\+ +

1101010101010101 0101010101010101

---------- +

0100010001000011 1000111100001100

0100101011000010

0100010001000011

\+

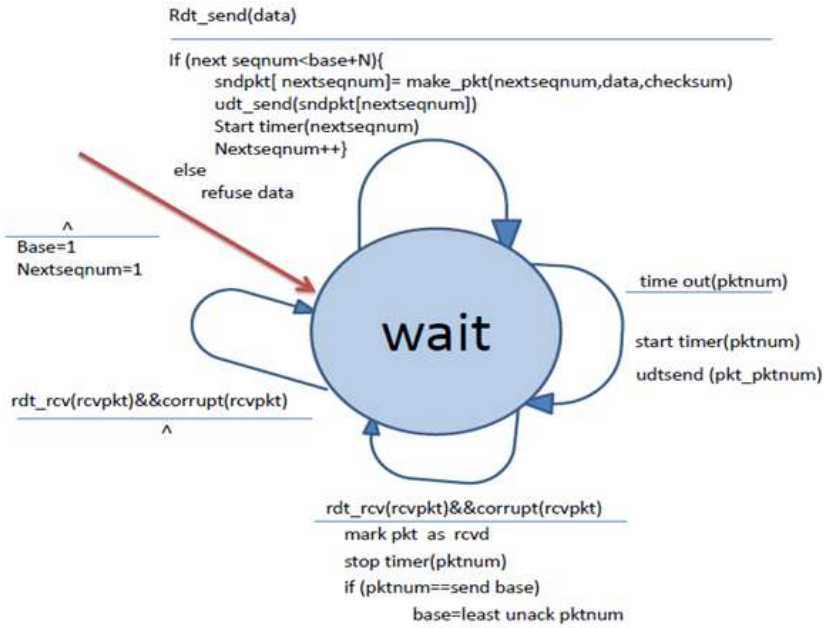
0100101011000010

1000111100000101  checksum field=0111000011111010

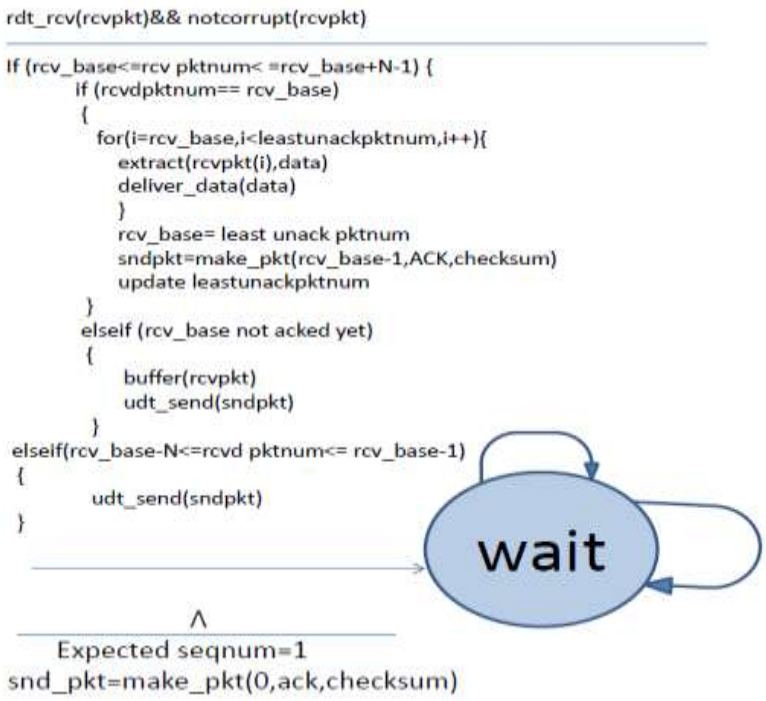
B) The error has been occurred and calculating the sum of headers and data won't result in 1111111111111111.

4)

Extended FSM description of SR sender:



Extended FSM description of SR receiver:



5-assume below situation in which N packet have been completely received and acked by the receiver, but acks have been lost in the way back to receiver, in such a situation when time out happens, the sender would send the packets again and The receiver has no idea about the lost of ack packets so it

would conceive retransmitted packets as new ones. To tackle the problem, it's enough for sequence number to be larger than window, even the sequence number equal to window size+1 could solve the problem.