CSCE 5580: Computer Network

Homework-1

Name: Kishan Kumar Zalavadia

EUID: 11685261

---

1. (10 pts), Please list all the protocol layers in the Internet protocol stack and describe their functionalities.

Ans: Layer Internet protocol stack has five layers, which are as follows

1. **Application layer:**
   This layer provides service directly to the end user. This layer is mainly used by the applications directly to transfer or exchange the data between the applications and one different device. The application can include browser applications, email, etc. SMTP, FTP, HTTP, and DNS are some of the protocols used by the application layer for data transfer.

2. **Transport layer:**
   This protocol is used for communication, ensuring reliable data transfer between different systems or devices. This is an end-to-end protocol. At the senders, the data is transferred from the application layer to the transport layer. The transport layer sends the data to the receiver, and the receiver's transport layer sends the data to the application layer. TCP and UDP are the protocols used by the transport layer.

3. **Network layer:**
   This layer is also an end-to-end service, but it focuses on how data is routed from the source to the destination, for example, how many routers and how many hops are needed for the data transfer to be managed by the network layer. Basically, it decides the path by which the package is sent from sender to receiver. IP (Internet Protocol) is generally used by the network layer. It assigns the IP address, which will help to know which device the data packet should go to.

4. **Link Layer:**
   This layer defines the protocol by which data can be transferred using the physical layer. This layer ensures a reliable data transfer between physically connected machines. It helps in node-to-node communication. Errors are detected (using checksums) and handled by the link layer. This layer provides a MAC address, which is a physical address.

5. **Physical layer:**
   This layer helps in transmitting raw data bits via a physical medium. The physical medium can be capable of wireless signals. This layer deals with hardware specifications. It deals with how the analog signal is transmitted.

---

2. (5+5 pts), Suppose within your Web browser you click on a link to obtain a Web page. The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address. Suppose that *n* DNS servers are visited before your host receives the IP address from DNS; the successive visits incur an RTT of RTT1, . . ., RTT*n*. Further, suppose that the Web page associated with the link contains exactly eight very small objects on the same server. Let RTT0 denote the RTT between the local host and the server containing the object. Assuming zero transmission time of the object, neglecting transmission times, how much time elapses with

a. Non-persistent HTTP connections?

b. Persistent HTTP?

Ans:

RTT1 is the time taken for a signal to go from client to server-1 and come back again to the client.

a. **Non-persistent HTTP connections?**
   - First, we need to establish a TCP connection, which is RTT0.
   - Next, we need to send an HTTP request and response for the main page, which is RTT0.
   - Since it's a non-persistent HTTP connection, we need to establish a connection to each of the eight small objects. Each object needs to have a TCP connection (1 RTT0) and HTTP request and response (1 RTT0).
   - We also need DNS lookup time which is RTT1+RTT2+RTT3+…..RTTn
   - Total time = TCP connection time + HTTP request-response time + (number of objects)*(time for retrieving an object) + DNS lookup time.

   Total time = RTT0 + RTT0 + 8*(RTT0+RTT0) + (RTT1+RTT2+RTT3+…..+RTTn)

   = 2 RTT0 + 16 RTT0 + (RTT1+RTT2+RTT3+…..+RTTn)

   = 18 RTT0 + (RTT1+RTT2+RTT3+…..+RTTn)

b. Persistent HTTP?

In persistent HTTP, the connection is maintained by the single webpage retrieval, which means the client does not need to open a new connection for eight small objects individually.

- First, we need to establish a TCP connection, which is RTT0.
- Next, we need to send an HTTP request and response for the main page, which is RTT0.
- For every object, the request-response: RTT0
- Total time = TCP connection time + HTTP request-response time + (number of objects)*(time for retrieving an object) + DNS lookup time.

Total time = RTT0 + RTT0 + 8*(RTT0) + (RTT1+RTT2+RTT3+…..+RTTn)

= 2 RTT0 + 8 RTT0 + (RTT1+RTT2+RTT3+…..+RTTn)

= 10 RTT0 + (RTT1+RTT2+RTT3+…..+RTTn)

---

3. (5+5 pts) Consider Figure3.5. What are the source and destination port values in the segments flowing from the server back to the client's processes? What are the IP addresses in the network-layer datagrams carrying the transport-layer segments?

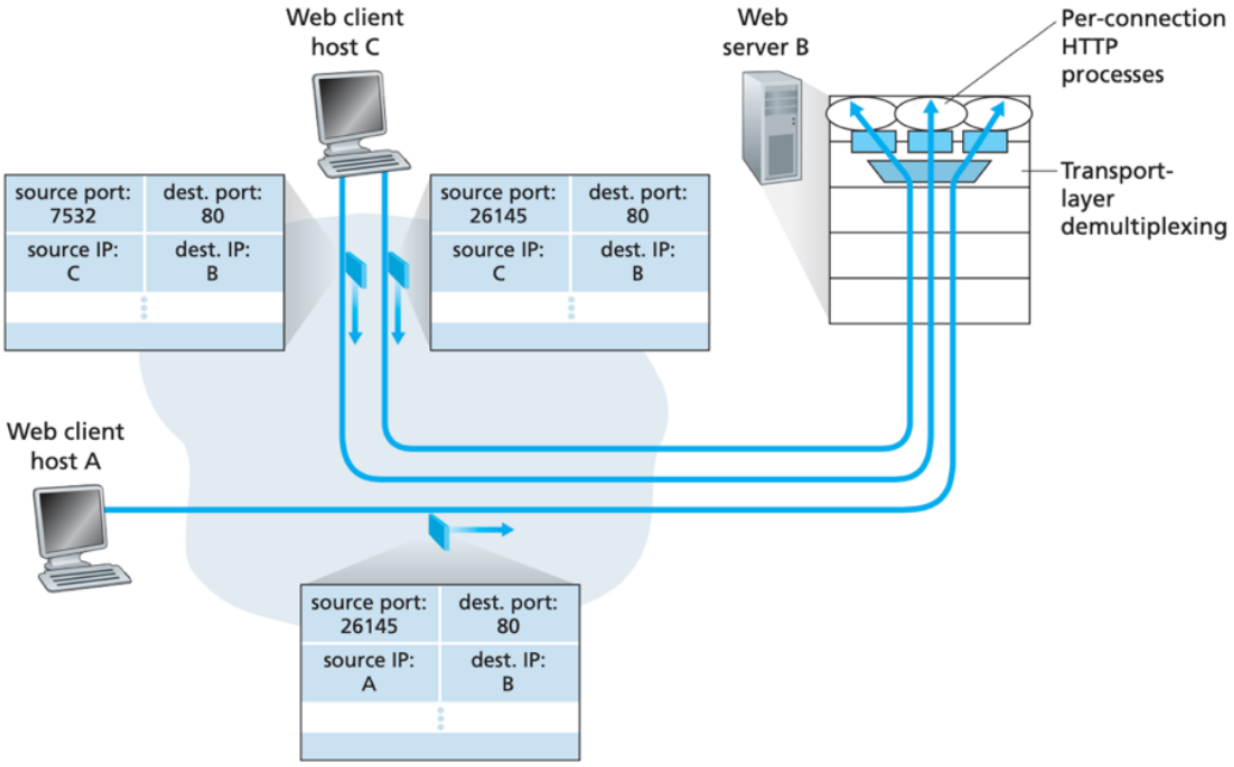


Figure 3.5 ♦ Two clients, using the same destination port number (80) to communicate with the same Web server application

Ans:

Port values:

- When the server responds to the clients, the source port on the server is 80 because the server is running on HTTP.
- The destination port on the server when it responds to clients depends on the client. If the server responds to client A, then the server port number will be 26145. When the server responds to client C, the server port number will be either 7532 or 26145, depending on the socket used.

IP Addresses:

- The source IP address will be the IP address of server B. So, if the IP address of server B is B, then the source IP address will be B.
- The destination IP address will depend on the client. If the server is responding to client B, then the destination IP address will be B (Assuming the IP address of client B is B).
  If the server is responding to client C, then the destination IP address will be C (Assuming the IP address of client C is C).

---

4. (10 pts) 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.

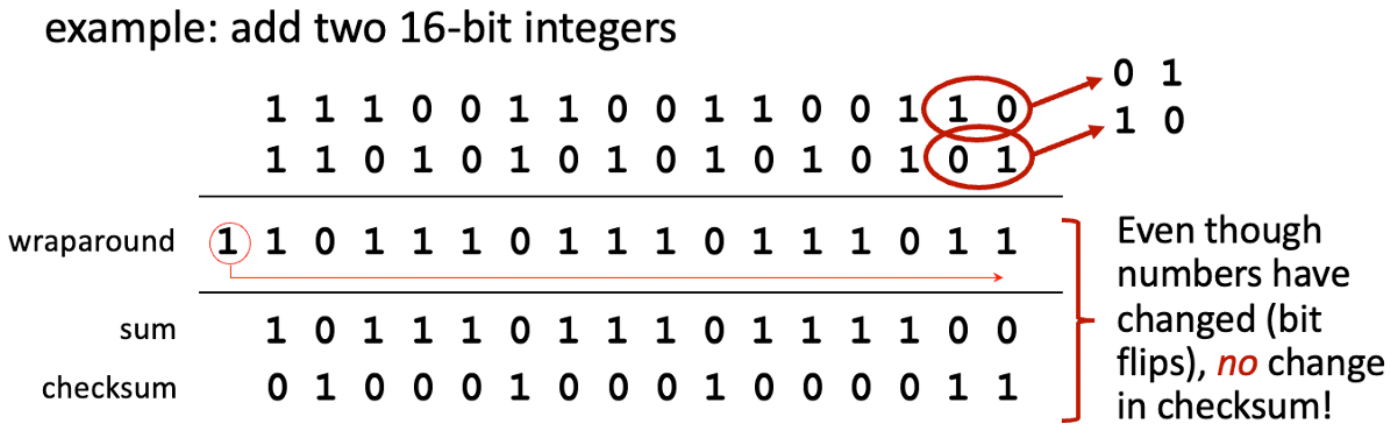
Can the receiver be absolutely certain that no bit errors have occurred? Explain.

Ans: The receiver cannot be absolutely certain that no bit errors have occurred, even if the checksum field matches the receiver UDP segment.

A checksum is used to check the integrity of the message. In UDP, the checksum is a once's complement of the sum of bits (message). It can miss in checking some errors. For example, a bit from message one is flipped from 0 to 1, and the same position bit from message two is flipped from 1 to 0, the checksum still remains the same, but there is an error in the message.

Some multiple-bit errors which have the sum as 0 can be missed in the checkout.

Because the checksum method is simple and easy to calculate, it has fewer error detection techniques; it can only find the most common errors, missing out on the complex errors.



---

5. (15 pts) Consider our motivation for correcting protocol rdt2.1. Show that the receiver, shown in Figure 3.57, when operating with the sender shown in Figure 3.11, can lead the sender and receiver to enter into a deadlock state, where each is waiting for an event that will never occur.

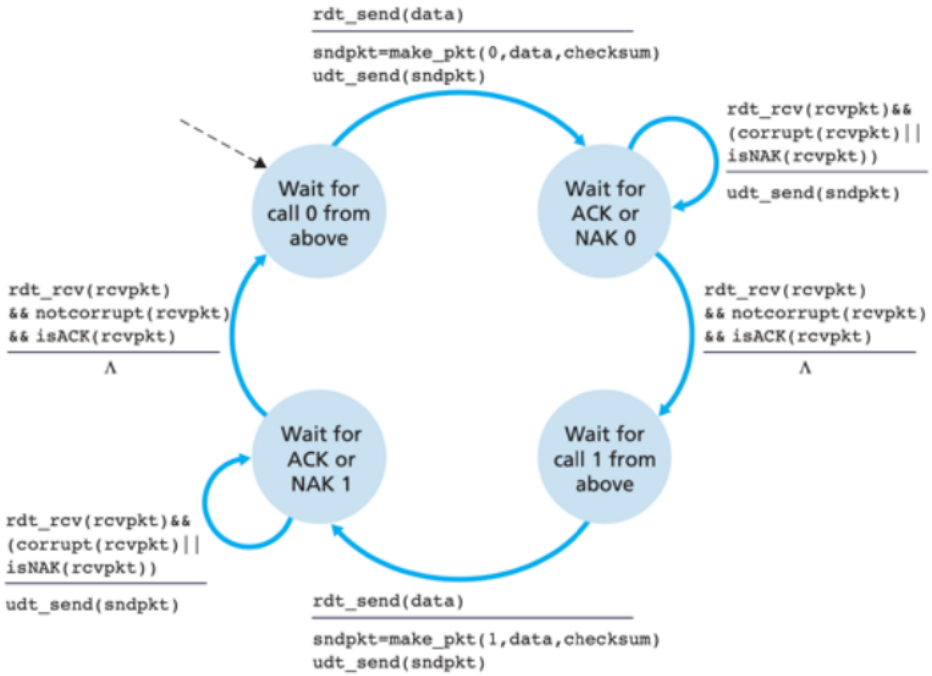


Figure 3.11 ♦ rdt2.1 sender

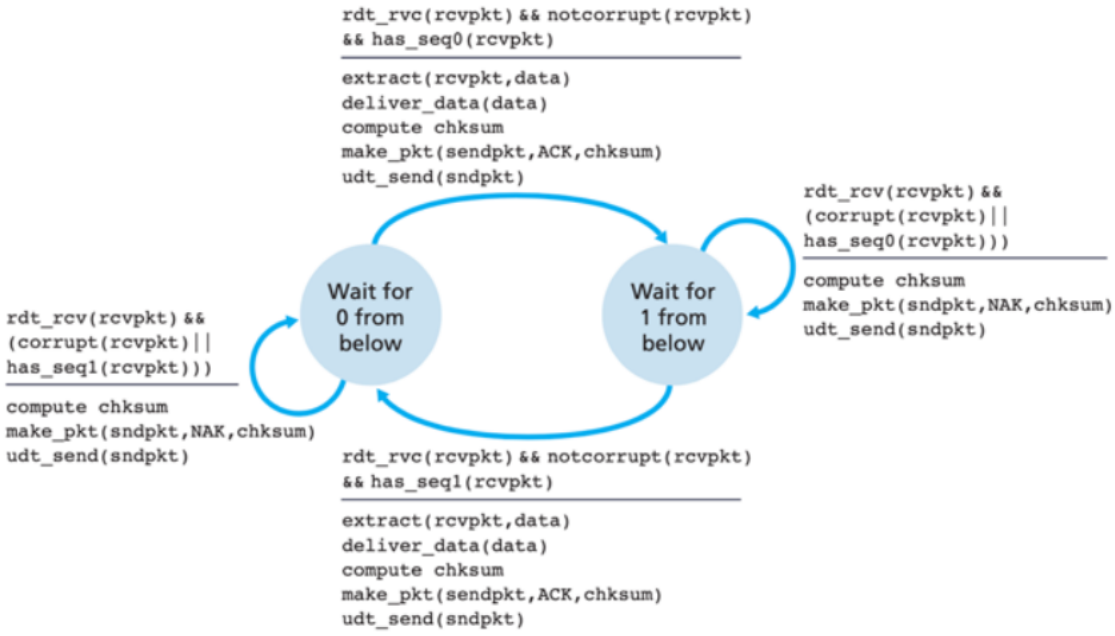


Figure 3.57 ♦ An incorrect receiver for protocol rdt 2.1

Ans:

- First, the sender sends the packet 0 and moves to the "wait for ACK or NAK 0" state.
- The receiver has received the packet correctly. The receiver extracts and delivers the data, sends an ACK for packet 0 to the sender, and moves to the 'wait for 1' state.
- The sender sends the packet with sequence number 1 and moves to 'wait for ACK or NAK 1'.
- The receiver receives packet 1 and sends the ACK to the sender. Then, the receiver moves to the 'wait for 0 from below' state where the receiver is expecting packet 0.
- Let us assume that the sender has received a corrupted ACK sent by the receiver for packet 1.
- Since the sender has received a corrupted ACK, it will resent packet 1.
- The receiver is expecting packet 0, and the sender has sent packet 1, so it sends an NAK.
- The sender sends packet one again, followed by the receiver sending it again.
- Therefore, the sender and receiver are in a deadlock state, where the sender keeps sending packet one, and the receiver keeps sending NAK.

---

6. (20 pts) 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.

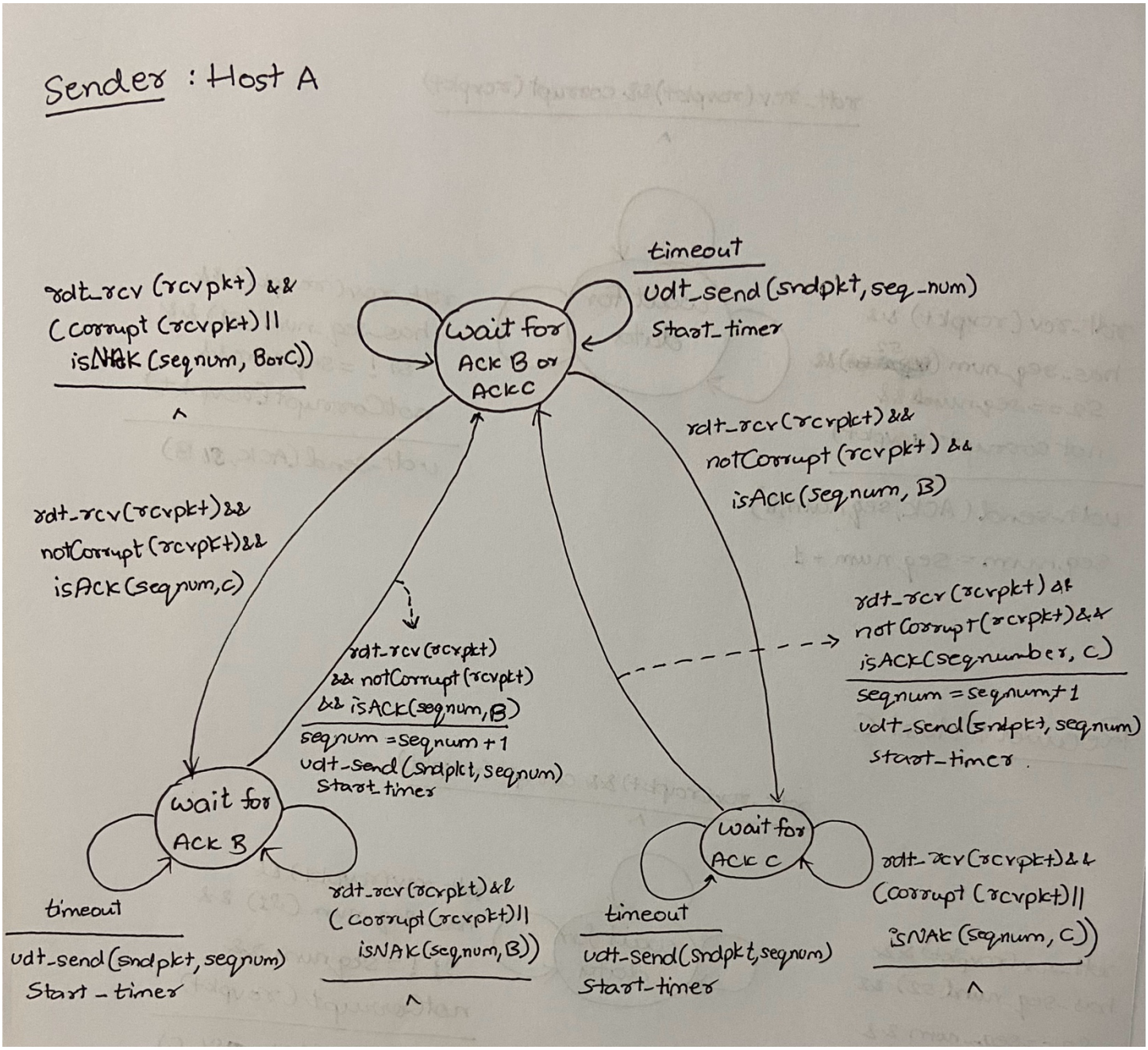
Packet format of Sender: [data | seq number]

Packet format of Receiver back to Sender: [ACK| seq number| from B or C]

Sender should follow have a timer to check timeout for sent package.

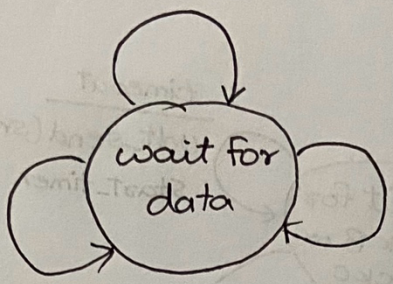
Ans:

Sender : Host A



Receiver : Host B

rdt_rcv(rcvpkt) && corrupt(rcvpkt)



rdt_rcv(rcvpkt) &&
has_seq_num(S2) &&
S2 == seq_num &&
not corrupt(rcvpkt)

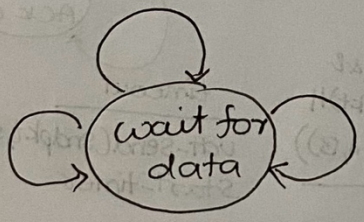
udt_send(ACK, seq_num, B)
seq_num = seq_num + 1

rdt_rcv(rcvpkt) &&
has_seq_num(S1) &&
S1 != seq_num &&
notCorrupt(rcvpkt)

udt_send(ACK, S1, B)

Receiver : Host C

rdt_rcv(rcvpkt) && corrupt(rcvpkt)



rdt_rcv(rcvpkt) &&
has_seq_num(S2) &&
S2 == seq_num &&
not Corrupt(rcvpkt)

udt_send(ACK, seq_num, C)
seq_num = seq_num + 1

rdt_rcv(rcvpkt) &&
has_seq_num(S1) &&
S1 != seq_num &&
notCorrupt(rcvpkt)

udt_send(ACK, S1, C)

Sender: Host A:

- First, in the wait for ACK B or ACKC state, the sender waits for the acknowledgment from either B or C. If there is a time-out, the packet is sent again with the sequence number, the timer is started again, and the sender remains in the same state.
- If at the 'Wait for ACK B or ACK C' state, the sender receives an NAK, or if the packet is corrupt, then the sender waits, and there is no action performed, and it remains in the same state. The packet is sent again when the timeout occurs.
- When the seder receives ACK from Host C, it means that Reciever C has received the packet correctly and we need to wait for receiver B to get the packet. So the sender moves to the 'wait for ACK B' state.
- At this state, if there is a timeout it will resend the packet and start the timer again.
- If it receives NAK from B or the packet is corrput, it performs no action. The packet will get sent when there is a timeout.
- When there is ACK from receiver B and the packet is not corrupt, the sequence number will be incremented by one, a new packet is sent, the timer is stater, and the sender moves to the ' wait for ACK B or ACK C state.
- When the seder receives ACK from Host B, it means that Reciever B has received the packet correctly and we need to wait for receiver C to get the packet. So the sender moves to the 'wait for ACK C' state.
- At this state, if there is a timeout it will resend the packet and start the timer again.
- If it receives NAK from C or the packet is corrupt, it performs no action. The packet will get sent when there is a timeout.
- When there is ACK from receiver C and the packet is not corrupt, the sequence number will be incremented by one, a new packet is sent, the timer is stater, and the sender moves to the ' wait for ACK B or ACK C state.
- This cycle repeats.

Receiver B:

- When the packet is corrupted, there is no action performed. Because there is no acknowledgment sent to the sender, there will be a timeout, and the sender will re-send the packet. And the receiver will stay in the same state.
- If the receiver receives the packet with the wrong sequence number, it will send an ACK with that sequence number and B, which will let the sender know that it has come from receiver B. And the receiver will stay in the same state.
- If the sequence number is correct, the receiver will send ACK with a sequence number B. It will then increment the sequence number by one. And the receiver will stay in the same state.

Receiver C:

- When the packet is corrupted, there is no action performed. Because there is no acknowledgment sent to the sender, there will be a timeout, and the sender will re-send the packet. And the receiver will stay in the same state.
- If the receiver receives the packet with the wrong sequence number, it will send an ACK with that sequence number and C, which will let the sender know that it has come from receiver C. And the receiver will stay in the same state.
- If the sequence number is correct, the receiver will send ACK with a sequence number and C. It will then increment the sequence number by one. And the receiver will stay in the same state.

Packet format:

Packet format of Sender: [data | seq number]

The data is the packet and the sequence number represent which packet is been sent.

Packet format of Receiver back to Sender: [ACK| seq number| from B or C]

The ACK is the acknowledgment which gives the which byte to expect next. There is a sequence number and also an indication that mentions B or C, which means this acknowledgment is from B or C.

---

7. (5+5+5+5 pts) Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A.

a) In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?

Ans:

- The sequence number for the 1st segment is 127, and the segment contains 80 bytes of data. So, for the 2nd segment, the sequence number will be 127 + 80 = 207.
- Source port number and destination port number remain the same as segment one because source and destination are the same as segment 1.
- Therefore,
  Sequence number = 207
  Source Port number: 302
  Destination Port number: 80

b) If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number, the source port number, and the destination port number?

Ans:

- Acknowledge the number, which is the next expected byte to be received by the receiver, which is Host B in this case.
- When the 1st segment arrives, it means all the bytes related to the 1st segment have arrived which is 126 + 80 = 206. So the next byte which it expects is 207.
- Or we can also write that the ack number is 127 (Sequence number) + 80 (Number of bytes of segment 1).
- The acknowledgment is sent from Host B to Host A, So the source port number is 80 (Port number of B) and the destination port number is 302 (Port number of A).
- Therefore,
  Acknowledgement number = 207
  Source Port number = 80
  Destination Port number = 302

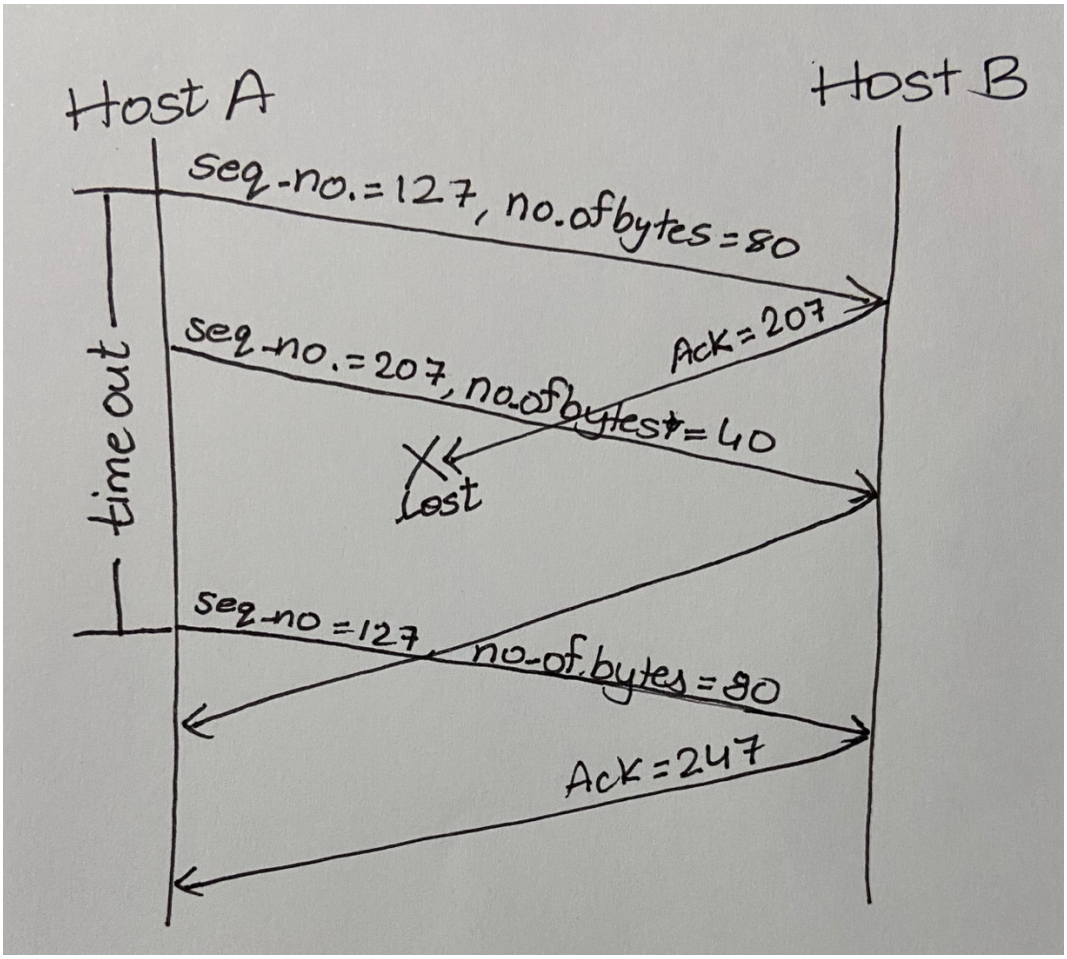
c) If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?

Ans:

- If the second segment arrives before the first segment, the acknowledged number will be 127. Because Host B will still be expecting the 127th byte since it has not received it.
- TCP will not acknowledge a bit until all the preceding bytes have not received.
- Therefore,
  Acknowledge number = 127

d) Suppose the two segments sent by A arrive in order at B. The first acknowledgment is lost, and the second acknowledgment arrives after the first time-out interval. Draw a timing diagram, showing these segments and all other segments and acknowledgments sent. (Assume there is no additional packet loss.) For each segment in your figure, provide the sequence number and the number of bytes of data; for each acknowledgment that you add, provide the acknowledgment number.

Ans:



When the acknowledgment is lost, the sender will again send the packet with sequence number 127 with 80 bytes because when the timeout occurs, the sender will think that the receiver has not received the packet.

---

8. (5 pts) True or False, give reasonable justification

1) Host A is sending Host B a large file over a TCP connection. Assume Host B has no data to send Host A. Host B will not send acknowledgments to Host A because Host B cannot piggyback the acknowledgments on data.

Ans: False

In a TCP connection, to maintain reliability, the receiver will send the acknowledgment to the sender saying that the packet has been received correctly, even if the receiver does not have any data to send. TCK connections do not require data to piggyback the acknowledgment on data.

2) The size of the TCP rwnd never changes throughout the duration of then connection.

Ans: False

TCP receiver window is used by the receiver to indicate how much buffer space is available by the receiver. If the available space is less or more based on the available space, the TCP rwnd needs to be changed.

3) Suppose Host A sends one segment with sequence number 38 and 4bytes of data over a TCP connection to Host B. In this same segment, the acknowledgment number is necessarily 42.

Ans: False

The sequence number in TCP is a number that denotes the byte number that is next expected. It may not be the sum of the current sequence number and the number of bytes of the data.

4) A user requests a Web page that consists of some text and three images. For this page, the client will send one request message and receive four response messages.

Ans: False

If a web page contains some text and three images, the client will send multiple requests to the server. One for the initial web page containing text and three requests for each individual image.

5) HTTP response messages never have an empty message body.

Ans: False

The HTTP response message can have an empty message body.

For example, a response to a HEAD request will require a header but nobody. Some status codes, like 204, which has no content, can also have an empty body.