**R1. Suppose the network layer provides the following service. The network layer in the source host accepts a segment of maximum size 1,200 bytes and a destination host address from the transport layer. The network layer then guarantees to deliver the segment to the transport layer at the destination host. Suppose many network application processes can be running at the destination host.**

1. **Design the simplest possible transport-layer protocol that will get application data to the desired process at the destination host. Assume the operating system in the destination host has assigned a 4-byte port number to each running application process.**
   1. Call this protocol Simple Transport Protocol (STP). At the sender side, STP accepts from the sending process a chunk of data not exceeding 1196 bytes, a destination host address, and a destination port number. STP adds a four-byte header to each chunk and puts the port number of the destination process in this header. STP then gives the destination host address and the resulting segment to the network layer. The network layer delivers the segment to STP at the destination host. STP then examines the port number in the segment, extracts the data from the segment, and passes the data to the process identified by the port number.
2. **Modify this protocol so that it provides a “return address” to the destination process.**
   1. The segment now has two header fields: a source port field and destination port field. At the sender side, STP accepts a chunk of data not exceeding 1192 bytes, a destination host address, a source port number, and a destination port number. STP creates a segment which contains the application data, source port number, and destination port number. It then gives the segment and the destination host address to the network layer. After receiving the segment, STP at the receiving host gives the application process the application data and the source port number.
3. **In your protocols, does the transport layer “have to do anything” in the core of the computer network?**
   1. No, the transport layer does not have to do anything in the core; the transport layer “lives” in the end systems.

**R2. Consider a planet where everyone belongs to a family of six, every family lives in its own house, each house has a unique address, and each person in a given house has a unique name. Suppose this planet has a mail service that delivers letters from source house to destination house. The mail service requires that (1) the letter be in an envelope, and that (2) the address of the destination house (and nothing more) be clearly written on the envelope. Suppose each family has a delegate family member who collects and distributes letters for the other family members. The letters do not necessarily provide any indication of the recipients of the letters.**

1. **Using the solution to Problem R1 above as inspiration, describe a protocol that the delegates can use to deliver letters from a sending family member to a receiving family member.**
   1. For sending a letter, the family member is required to give the delegate the letter itself, the address of the destination house, and the name of the recipient. The delegate clearly writes the recipient’s name on the top of the letter. The delegate then puts the letter in an envelope and writes the address of the destination house on the envelope. The delegate then gives the letter to the planet’s mail service. At the receiving side, the delegate receives the letter from the mail service, takes the letter out of the envelope, and takes note of the recipient name written at the top of the letter. The delegate then gives the letter to the family member with this name.
2. **In your protocol, does the mail service ever have to open the envelope and examine the letter in order to provide its service?**
   1. No, the mail service does not have to open the envelope; it only examines the address on the envelope.

**R3. Consider a TCP connection between Host A and Host B. Suppose that the TCP segments traveling from Host A to Host B have source port number x and destination port number y. What are the source and destination port numbers for the segments traveling from Host B to Host A?**

Source port number y and destination port number x

**R4. Describe why an application developer might choose to run an application over UDP rather than TCP.**

An application developer may not want its application to use TCP’s congestion control, which can throttle the application’s sending rate at times of congestion. Often, designers of IP telephony and IP videoconference applications choose to run their applications over UDP because they want to avoid TCP’s congestion control. Also, some applications do not need the reliable data transfer provided by TCP.

**R5. Why is it that voice and video traffic is often sent over TCP rather than UDP in today’s Internet? (Hint: The answer we are looking for has nothing to do with TCP’s congestion-control mechanism.)**

Since most firewalls are configured to block UDP traffic, using TCP for video and voice traffic lets the traffic though the firewalls.

**R6. Is it possible for an application to enjoy reliable data transfer even when the application runs over UDP? If so, how?**

Yes. The application developer can put reliable data transfer into the application layer protocol. This would require a significant amount of work and debugging, however.

**R7. 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. Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts?**

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.

**R8. 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. Are all of the requests being sent through the same socket at Host C? If they are being passed through different sockets, do both of the sockets have port 80? Discuss and explain.**

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.

**R9. In our rdt protocols, why did we need to introduce sequence numbers?**

Sequence numbers are required for a receiver to find out whether an arriving packet contains new data or is a retransmission.

**R10. In our rdt protocols, why did we need to introduce timers?**

To handle losses in the channel. If the ACK for a transmitted packet is not received within the duration of the timer for the packet, the packet (or its ACK or NACK) is assumed to have been lost. Hence, the packet is retransmitted.

**R11. Suppose that the roundtrip delay between sender and receiver is constant and known to the sender. Would a timer still be necessary in protocol rdt 3.0 , assuming that packets can be lost? Explain.**

A timer would still be necessary in the protocol rdt 3.0. If the round trip time is known then the only advantage will be that, the sender knows for sure that either the packet or the ACK (or NACK) for the packet has been lost, as compared to the real scenario, where the ACK (or NACK) might still be on the way to the sender, after the timer expires. However, to detect the loss, for each packet, a timer of constant duration will still be necessary at the sender.

**R12. Visit the Go-Back-N Java applet at the companion Web site.**

1. **Have the source send five packets, and then pause the animation before any of the five packets reach the destination. Then kill the first packet and resume the animation. Describe what happens.**
   1. The packet loss caused a time out after which all the five packets were retransmitted.
2. **Repeat the experiment, but now let the first packet reach the destination and kill the first acknowledgment. Describe again what happens.**
   1. Loss of an ACK didn’t trigger any retransmission as Go-Back-N uses cumulative acknowledgements.
3. **Finally, try sending six packets. What happens?**
   1. The sender was unable to send sixth packet as the send window size is fixed to 5.

**R13. Repeat R12, but now with the Selective Repeat Java applet. How are Selective Repeat and Go-Back-N different?**

1. When the packet was lost, the received four packets were buffered the receiver. After the timeout, sender retransmitted the lost packet and receiver delivered the buffered packets to application in correct order.
2. Duplicate ACK was sent by the receiver for the lost ACK.
3. The sender was unable to send sixth packet as the send window size is fixed to 5 When a packet was lost, GO-Back-N retransmitted all the packets whereas Selective Repeat retransmitted the lost packet only. In case of lost acknowledgement, selective repeat sent a duplicate ACK and as GO-Back-N used cumulative acknowledgment, so that duplicate ACK was unnecessary.

**R14. True or false?**

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.**
   1. False
2. **The size of the TCP rwnd never changes throughout the duration of the connection.**
   1. False
3. **Suppose Host A is sending Host B a large file over a TCP connection. The number of unacknowledged bytes that A sends cannot exceed the size of the receive buffer.**
   1. True
4. **Suppose Host A is sending a large file to Host B over a TCP connection. If the sequence number for a segment of this connection is m, then the sequence number for the subsequent segment will necessarily be m+1.**
   1. False
5. **The TCP segment has a field in its header for rwnd**
   1. True
6. **Suppose that the last SampleRTT in a TCP connection is equal to 1 sec. The current value of TimeoutInterval for the connection will necessarily be sec.**
   1. False
7. **Suppose Host A sends one segment with sequence number 38 and 4 bytes of data over a TCP connection** to **Host B. In this same segment the acknowledgment number is necessarily 42.**
   1. False

**R15. 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.**

1. **How much data is in the first segment?**
   1. 20 bytes
2. **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?**
   1. 90

**R16. Consider the Telnet example discussed in Section 3.5 . A few seconds after the user types the letter ‘C,’ the user types the letter ‘R.’ After typing the letter ‘R,’ how many segments are sent, and what is put in the sequence number and acknowledgment fields of the segments?**

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

**R17. 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. What transmission rate would TCP like to give to each of the connections?**

R/2

**R18. True or false? Consider congestion control in TCP. When the timer expires at the sender, the value of ssthresh is set to one half of its previous value.**

False, it is set to half of the current value of the congestion window.

**R19. In the discussion of TCP splitting in the sidebar in Section 3.7 , it was claimed that the response time with TCP splitting is approximately Justify this claim.**

Let X = RTTFE, Y = RTTBE and ST = Search time

TCP packet exchange diagram between a client and a server (Back End) with a proxy (Front End) between them. From this diagram we see that the total time is 4X + Y+ ST = 4\*RTTFE + RTTBE + Search time