Chapter 3, Chapter 4

**Part 2. Textbook questions**

**Chapter 3.** [40 points]

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

An application developer would choose TCP over UDP when they need a reliable data transfer. The inner workings of TCP use flow control, sequence numbers, acknowledgments, and times to ensure that data is delivered and intact. So if the information is critical and needs a guaranteed reliable transfer TCP is best. Whereas a connection with UDP would not be guaranteed. Another concept would be congestion control, this would help an application developers project not eat up all the bandwidth.TCP allows for congestion control which regulates an applications bandwidth, essentially leaving wiggle room for other packets. UDP does not provide congestion control and quickly congest network bandwidth. You would also use TCP when data is not time sensitive and can take the time to be propagated for transport.

3.2 Suppose host A is sending host B a large file over a TCP connection. If the acknowledge number for a segment of this connection is *y*, then the acknowledge number for the subsequent segment will necessarily be *y+ 1*. Is this true or false? Why?

False, all though it is true that the acknowledgment will be up one from the prior packets sequence number. The next sequence number will be comprised of a totally new number and its corresponding acknowledgement packet will be up one from that new sequence number as well. For example, packet#1 sends SEQ 100 and ACK 0, packet#2 returns a SEQ110 and ACK 101.Then packet number#3 could have SEQ125 and ACK111. This does not follow the y+1 rule because the next packets sequence number effects the next packets acknowledgment number. Not the previous packets ACK effecting the next packets ACK as the question implies. Although it should be noted that in specific instances the y+1 rule could be applied. For instance, if the date field of the response is a length of 1, while there is no cumulative acknowledgement mechanism, and there are no retransmitted packets. If all three of these are met, then yes it will meet the y+1 rule.

3.3 Suppose 5 TCP connections are present over some bottleneck link of rate X bps. All 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 is the transmission rate that TCP would like to give to each of the connections?

The formula is (Rate / Connections). So at a rate of (X Bps / 5 Connections) = X/5

3.4 How to identify a UDP socket? How to identify a TCP socket? Are these data fields same? Why?

To properly identify a UDP socket you only need a destination IP address and destination port number. This is because UDP is connectionless with no guaranteed response so source packet information is not needed. TCP not only needs a destination IP and Port, but also needs a host IP and Port as well. This is to provide retransmissions and acknowledgments.

3.5 UDP and TCP use 1's complement for their checksums. Suppose you have the following three 8-bit words: 10010101, 01111000, 10101011. What is the 1's complement of the sum of these words? Show all work. Why UDP takes the 1's complement of the sum, that is, why not just use the sum?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First find sum of 10010101 and 01111000 | | | | | | | | | | | | The take sum of previous and add it to 10101011 | | | | | | | | | | | | | |
|  | 1 | 0 | 0 | 1 | | 0 | 1 | 0 | 1 | | 🡪 |  | | 0 | 0 | 0 | | 0 | 1 | 1 | | 0 | 1 | |  |
| + | 0 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | | + | | 1 | 0 | 1 | | 0 | 1 | 0 | | 1 | 1 | |  |
| = | 0 | 0 | 0 | 0 | | 1 | 1 | 0 | 1 | | = | | 1 | 0 | 1 | | 1 | 1 | 0 | | 0 | 0 | |  |
| Finding ones compliment of sum of words (10111000) | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Sum | | | | | 1 | | | | | 0 | | | 1 | | | | 1 | | | | 1 | | | 0 | | | 0 | 0 |
| Ones Complement | | | | | 0 | | | | | 1 | | | 0 | | | | 0 | | | | 0 | | | 1 | | | 1 | 1 |

Ones Complement = 01000111

The reason why UDP takes the 1’s complement of the sum is that it is endian independent and does not need to compare the checksums of both the sending and receiving packets. What UDP does is the receiver just verifies that the entire packets checksum equals 0. You would not use the sum with UDP because it is connectionless.

3.6 Suppose Client A initiates a HTTPS session with server S. Provide possible source and destination port numbers for:

a. The segment sent from S to A.

Source =443, Destination = 8443.

b. The segment sent from A to S.

Source = 8443, Destination = 443.

3.7 Compare two pipelining protocols shown in the textbook – go-back-N and selective repeat.

The GBN protocol essentially allows you to set a variable on how many packets you can transmit without receiving acknowledgments. This allows you to set a threshold on your outgoing traffic effectively controlling network congestion.One issue with GBN is that when a single packet error occurs it can cause all the packets in the pipeline to get retransmitted and fill up. Selective repeat avoids this by only retransmitting the packets it suspects it received in error. Unlike GBN, SR will have already received ACKs for some the packets. SR will acknowledge a packet whether or not it is in order. It will que that acknowledged packet and will be buffered till the correctly sequenced packets are received. At this point they will all be added to the pipeline.

3.8 In our textbook, protocol rdt 3.0 shows a data transfer protocols that uses only acknowledges. As an alternative, consider a reliable data transfer protocol that uses negative acknowledgements. Suppose the sender sends data only infrequently. Will a NAK-only protocol be preferable to protocol that uses ACKs? Why? Suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In the second case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

If the sender sends data infrequently then the NAK-only protocol is NOT preferred. This is because NAK-only protocols only detect packet loss when the next sequential packet is received. If the packets are sent frequently and with few losses then a NAK-only protocol is preferred. In the case of the end-to-end connections having a lot of data and with few losses you would want to use a NAK-only protocol since packets could receive their next sequential packet fast and reliably to verify there was no loss of a packet.

3.9 Let us assume 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? Please explain.

Assuming that a roundtrip delay is known, a timer would still be beneficial and necessary. When setting a timer in rdt 3.0, you must consider the known delay already in your network. You should overcompensate the timer to encompass your delay, plus what you deem is acceptable as threshold for your timer. You will want to implement this to verify packets haven’t been lost after your delay has been met.

3.10 Briefly discuss the basic mechanisms adopted by TCP congestion control.

TCP uses additive-increase, multiplicative decrease (AIMD) for congestion control. Modeling this out you will see it creates a saw tooth patterned chart that essentially probes the network for bandwidth. Essentially as ACKs are received the congestion window increases in a linear (additive) fashion. When three duplicate ACKs are received the fast recovery state is entered. This makes the threshold smaller in an exponential (multiplicative) fashion. AIMD cautiously allows more bandwidth when needed and aggressively takes it away in signs of distress. This is what causes the saw tooth pattern mentioned above in reference to its congestion control.

**Chapter 4** [40 points]

4.1 Describe two major network-layer functions in a datagram network.

The two major network-layer functions are Forwarding and Routing. Forwarding is done in the router, transferring packets from interface to interface and does not mess with the overall network process. Routing will actually determine the end-to-end paths as it traverses the network determining the next hop outside of the routers forwarding table.

4.2 Describe how packet loss can occur at input and outputs of a router. Is it possible to eliminate packet loss at these ports? If so, how? If not, please explain.

Packet loss can occur at input if the queue size gets larger from slow switching speed. This exhausts the buffer space causing packet loss. This can be prevented if the switching speed is at least as fast the input speed multiplied by the number of input ports. (To prevent packet loss >= (number of input ports x input line speed.)) On the outputs, if the outgoing speed is to slow the queue will fill up causing packet loss.

4.3 Suppose an application generates chunks of 1360 bytes of data every 20 msec, and each chunk gets encapsulated in a TCP segment and then an IP datagram. What percentage of each datagram will be overhead, and what percentage will be application data?

*Header field = (TCP header + IP header) = (20 Bytes + 20 Bytes) = 40 Bytes.*

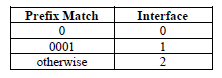
*Datagram = (40 Bytes + 1360 Bytes) = 1400 Bytes.*

*Overhead = (Header /Datagram) = (40 / 1400) = (.028571)*

*Application Data = (Data / Overhead) = (1360 / 1400) = (.97142)*

So rounding up Overhead is about 03%, while rounding down Application Data will be about 97%.

4.4 Consider a datagram network using 8-bit host addresses. Suppose a router uses longest prefix matching and has the following forwarding table:



For each of the 3 interfaces, give the associated range of destination host addresses and the number of addresses in the range.

Interface 0 = (00000000 through 011111111) = 255 networks.

Interface 1 = (00010000 through 00011111) = 16 networks.

Otherwise = (00000000 through 10111111) = 192 networks.

**3. Practical assignment** [20 points]

(Please see attached ZIP file in Canvas.)

**import** java.io.\*;

**import** java.net.\*;

**public** **class** EmailAgent {

**public** **static** **void** main(String[] args) **throws** Exception {

// Establish a TCP connection with the mail server.

Socket socket = **new** Socket("localhost", 25);

//Create a BufferedReader to read a line at a time.

InputStream is = socket.getInputStream();

InputStreamReader isr = **new** InputStreamReader(is);

BufferedReader br = **new** BufferedReader(isr);

//Read greeting from the server.

String response = br.readLine();

System.***out***.println(response);

**if** (!response.startsWith("220")) {

**throw** **new** Exception("220 reply not received from server.");

}

//Get a reference to the socket's output stream.

OutputStream os = socket.getOutputStream();

//Send HELO command and get server response.

String command = "HELO Eric\r\n";

System.***out***.print(command);

os.write(command.getBytes("US-ASCII"));

response = br.readLine();

System.***out***.println(response);

**if** (!response.startsWith("250")) {

**throw** **new** Exception("250 reply not received from server.");

}

//Send MAIL FROM command.

command = "MAIL FROM: webberic92@yahoo.com\r\n";

System.***out***.println(command);

os.write(command.getBytes("US-ASCII"));

response = br.readLine();

System.***out***.println(response);

**if** (!response.startsWith("250")) {

**throw** **new** Exception("250 reply not received from server. MAIL");

}

//Send RCPT TO command.

command = "RCPT TO: eric@webby.com\r\n";

System.***out***.println(command);

os.write(command.getBytes("US-ASCII"));

response = br.readLine();

System.***out***.println(response);

**if** (!response.startsWith("250")) {

**throw** **new** Exception("250 reply not received from server.");

}

//Send DATA command.

command = "DATA\r\n";

System.***out***.println(command);

os.write(command.getBytes("US-ASCII"));

response = br.readLine();

System.***out***.println(response);

**if** (!response.startsWith("354")) {

**throw** **new** Exception("data 354 data not received from server.");

}

//Send message data.

os.write("SUBJECT: JAVA SMTP \r\n".getBytes("US-ASCII"));

os.write("Welcome to Webbys SMTP service! \r\n".getBytes("US-ASCII"));

os.write("Ending Email now \r\n".getBytes("US-ASCII"));

//End with line with a single period.

os.write(".\r\n".getBytes("US-ASCII"));

response = br.readLine();

**if** (!response.startsWith("250")) {

**throw** **new** Exception(" period 250 reply not received from server.");

}

//Send QUIT command.

os.write("QUIT\r\n".getBytes("US-ASCII"));

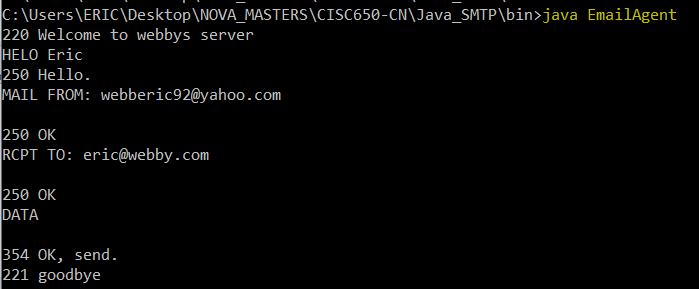
response = br.readLine();

System.***out***.println(response);

}

}

From Command Line.





**Certificate of Authorship**

Submitted to: Professor Wei Li

Student’s Name: Eric Webb

Date of Assignment: 09/29/2019

Title of Assignment: Assignment No. 2

Certification of Authorship: I hereby certify that I am the author of this document and that any assistance I received in its preparation is fully acknowledged and disclosed in the document. I have also cited all sources from which I obtained data, ideas, or words that are copied directly or paraphrased in the document. Sources are properly credited according to accepted standards for professional publications. I also certify that this paper was prepared by me for this course.

Student's Signature: ERIC WEBB