# Tutorial questions for Chapter 3.

Expected time to complete: 1-2 week(s).

Simple questions: Pease answer the following questions:

1. Briefly define the following terms/concepts:

a. Multiplexing

Multiplexing is the process of directing the packet from the source host by attaching a relevant transport header.

b. Demultiplexing

Demultiplexing is the process of directing received packets to the correct destination socket by using the transport header information.

c. UDP

UDP, User Datagram Protocol, is a transport layer protocol that offers datagram-based packet-switched communication without establishing connections.

d. TCP

TCP, Transmission Control Protocol, is a transport layer protocol that offers reliable host-to-host communication on a packet-switched network.

e. TCP congestion control: slow start

Slow start begins initially with a small congestion window size (CWND), increasing the CWND by one SMSS (sender maximum segment size) with each ACK received until a loss is detected or the receiver window size (RWND) is reached.

f. TCP congestion control: congestion avoidance

Congestion avoidance extends the slow start algorithm beyond ssthresh (slow start threshold). The difference is that congestion avoidance MUST NOT increase the CWND by more than SMSS bytes for every RTT (compared to 1 SMSS per ACK for slow start).

g. TCP congestion control: fast recovery

With fast recovery and retransmit, the receiver will send a duplicate ACK when a packet is lost. When the sender receives 3 duplicate ACKs, the data segment indicated by the ACKs is then immediately retransmitted (in contrast to waiting for a timeout).

2. What are the 4-tuple used to identify a TCP socket?

(Source IP address, Source port number, Destination IP address, Destination port number)

3. What information does a UDP segment include?

UDP segment consists of the header and payload. The header includes source port, destination port, length (of the entire packet, i.e. the header length + payload length) and the checksum.

4. What is stop-and-wait operation in rdt3.0?

The sender sends the packet and waits for the ACK (acknowledgement) of the packet. Once the ACK reaches the sender, it transmits the next packet in row. If the ACK is not received by the time the countdown timer has expired, it re-transmits the previous packet again.

5. What are the differences between the following?

a. Go-Back-N and selective repeat in pipelined protocols

| Go-Back-N | Selective Repeat |
|---|---|
| Sender sends a chunk of packets and waits for a cumulative ACK. | Sender sends packets and waits for individual ACKs. |
| In case of loss or damage to packets, the entire chunk (of N packets) must be retransmitted. | In case of loss or damage to packets, only the packets in question must be retransmitted. |
| If retransmission occurs, it will occupy more bandwidth. Conversely, it saves bandwidth in terms of acknowledgements sent. | If retransmission occurs, it will occupy less bandwidth. Conversely, it expends more bandwidth in terms of acknowledgements sent. |

b. TCP retransmission and TCP ‘fast’ retransmit

| TCP retransmission | TPC Fast retransmit |
|---|---|
| TCP retransmission occurs when a packet is damaged (verifiable by the checksum) or lost (when there is a timeout). | Occurs when the sender receives three duplicate ACKs before a timeout. In such a case, the packet with the next higher sequence number is immediately retransmitted and the timeout is also reset. |

6. What are the fields that are not included in UDP segment but included in TCP segment?

In addition to the fields contained in a UDP segment, TCP segments also contain

- 32 bit “Sequence number” and “Acknowledgment number” fields,
- 16 bit “Window”, “Checksum” and “Urgent Pointer” fields,
- 6 bit “Reserved” and “Flags” fields,
- 4 bit “Data offset” field, and
- Variable size “Options” field

Calculation questions:

Please solve the following questions.

7. Consider the two 16-bit words (shown in binary) below. Recall that to compute the Internet checksum of a set of 16-bit words, we compute the one's complement sum [1] of the two words. That is, we add the two numbers together, making sure that any carry into the 17th bit of this initial sum is added back into the 1's place of the resulting sum); we then take the one's complement of the result. Compute the Internet checksum value for these two 16-bit words:

|  |  |  |
|---|---|---|
| a. | 10010111 00011101 | this binary number is 38685 decimal (base 10) |
|  | 11110000 10010111 | this binary number is 61591 decimal (base 10) |

Step 1. Add the two numbers

```
        1 0 0 1 0 1 1 1 0 0 0 1 1 1 0 1
+       1 1 1 1 0 0 0 0 1 0 0 1 0 1 1 1
= (+1)  1 0 0 0 0 1 1 1 1 0 1 1 0 1 0 0
```

Step 2. Carry over the overflow

```
=       1 0 0 0 0 1 1 1 1 0 1 1 0 1 0 1
```

Step 3. Compute one’s complement

= **0 1 1 1 1 0 0 0 0 1 0 0 1 0 1 0**

|  |  |
|---|---|
| b. | 11110000 1001111 |
|  | 10010000 1001110 |

Step 1. Add the two numbers

```
        1 1 1 1 0 0 0 0 1 0 0 1 1 1 1
+       1 0 0 1 0 0 0 0 1 0 0 1 1 1 0
= (+1)  1 0 0 0 0 0 0 1 0 0 1 1 1 0 1
```

Step 2. Carry over the overflow

```
=       1 0 0 0 0 0 0 1 0 0 1 1 1 1 0
```

Step 3. Compute one’s complement

= **0 1 1 1 1 1 1 0 1 1 0 0 0 0 1**

8a. Fill out the table below indicating (i) the state of the sender and the receiver just after the transmission of a new packet in response to the received packet at time t , (ii) the sequence number associated with the data packet or the ACK number associated with the ACK packet sent at time t.

| t | Sender state | Receiver state | Packet type sent | Seq# or Ack# sent |
|---|---|---|---|---|
| 0 | Wait for ACK 0 | Wait 0 from below | data | 0 |
| 1 | Wait for ACK 0 | Wait 0 from below | ACK | 1 |
| 2 | Wait for ACK 0 | Wait 0 from below | data | 0 |
| 3 | Wait for ACK 0 | Wait 1 from below | ACK | 0 |
| 4 | Wait for ACK 1 | Wait 1 from below | data | 1 |
| 5 | Wait for ACK 1 | Wait 0 from below | ACK | 1 |
| 6 | Wait for ACK 0 | Wait 0 from below | data | 0 |

b. How many times is the payload of the received packet passed up to the higher layer at the receiver in the above example? At what times is the payload data passed up?

The payload data is passed up to the higher level at the receivers end exactly two times,

i.e. *t* = *{3,5}* (when data packets are sent to the receiver and received without corruption*).*

9. TCP sequence and ACK numbers with segment loss:

a. Give the sequence numbers associated with each of the three segments sent by the sender.

First segment seq# = **101**

Second segment seq# = first segment seq# + first segment size = 101 + 450 = **551**

Third segment seq# = second segment seq# + second segment size = 551 + 450 = **1001**

b. List the sequence of acknowledgements transmitted by the TCP receiver in response to the receipt of the segments actually received. In particular, give the value in the acknowledgement field of each receiver-to-sender acknowledgement, and give a brief explanation as to why that particular acknowledgement number value is being used.

First segment's acknowledgment # = first segment seq# + first segment size = **551**

Second segment is lost, so there is **no ack** for it.

Third segment arrives at the receiver end and when a seq# equal to the last ack# is expected, instead 1001 is received as seq# (1001 > 551). Therefore, the sender sends a duplicate ack (ack# = **551**) to the sender to alert them of the missing segment.

10. Computing TCP's RTT and timeout values.

Suppose that TCP's current estimated values for the round-trip-time (estimatedRTT) and deviation in the RTT (DevRTT) are 315 msec and 36 msec, respectively (see Section 3.5.3 for a discussion of these variables). Suppose that the next three measured values of the RTT are 240, 300, and 380 respectively.

Compute TCP's new value of estimatedRTT, DevRTT, and the TCP timeout value after each of these three measured RTT values is obtained. Use the values of α = 0.125 and β = 0.25.

Using the formulas below to answer the following questions:

$$\text{EstimatedRTT}^{o} = (1-\alpha)*\text{EstimatedRTT} + \alpha*\text{SampleRTT}$$

$$\text{DevRTT}^{o} = (1-\beta)*\text{DevRTT} + \beta*(\text{abs}(\text{SampleRTT} - \text{EstimatedRTT}^{o}))$$

TimeoutInterval = EstimatedRTT° + 4* DevRTT°

a. After the first RTT estimate is made

- estimatedRTT = 0.875*315+0.125*240 = 305.625
- DevRTT = 0.75*36+0.25*(abs(240-305.625)) = 43.40625
- TimeoutInterval = 305.625 + 4*43.40625 = 479.25

b. After the second RTT estimate is made

- estimatedRTT = 0.875*305.625 + 0.125*300 = 304.9219
- DevRTT = 0.75*43.40625 + 0.25*(abs(300-304.9219)) = 33.78516
- TimeoutInterval = 304.9219 + 4*33.78516 = 440.0625

c. After the third RTT estimate is made

- estimatedRTT = 0.875*304.9219 + 0.125*380 = 314.3067
- DevRTT = 0.75*33.78516+0.25*(abs(380-314.3067)) = 41.7622
- TimeoutInterval = 314.3067 + 4*41.7622 = 481.3555