Solution to COMP3331 (2010 Session 1) mid-session test

Question 1

Answer is (e). One HTTP request message is used to obtain the HTML document and one message is used for each image. One HTTP response message is sent per HTTP request.

Note: The answer is independent of whether the browser uses pipelining or not, or whether HTTP 1.0 or 1.1 is used.

Question 2

(a) The answer can be TCP or UDP. If the answer is TCP, the student should mention that TCP provides reliable delivery to the application. If the answer is UDP, then the reliable delivery mechanism should be embedded in the application layer.

(b) UDP. UDP is faster because it doesn't need connection setup and it doesn't guarantee reliable delivery.

Question 3

Since Anna uses web-based mail, her email message is delivered from Anna's host to her mail server using HTTP. DNS may also be used for domain name resolution.

The delivery from Anna's mail server to Beatrice's mail server uses SMTP. DNS may also be used.

For Beatrice, she can be using POP3 or IMAP to read her email. DNS may be also be used.

Question 4

Let us begin with number of basic calculations:

- Propagation delay in link type 1 $(d_{p1}) = \frac{2000 \times 10^3}{2 \times 10^8} = 0.01$ s
- Transmission delay in link type 1 $(d_{t1}) = \frac{8 \times 1024}{100 \times 10^3} \approx 0.08$ s
- Propagation delay in link type 2 $(d_{p2}) = \frac{4000 \times 10^3}{2 \times 10^8} = 0.02$ s
- Transmission delay in link type 2 $(d_{t2}) = \frac{8 \times 1024}{50 \times 10^3} \approx 0.16$ s

(a) From the timing diagram (see the scanned pages at the back of this document), the time required = $2d_{p1} + 5d_{t1}$ = 2 (0.01) + 5 (0.08) = 0.42s
(b) From the timing diagram (see the scanned pages at the back of this document), the time required = $d_{p1} + d_{t1} + d_{p2} + 4d_{t2}$ = 0.01 + 0.08 + 0.02 + 4(0.16) = 0.75s

Question 5

In order for all the peers to receive the file, the file must have been uploaded N times by the server and the non-selfish peers. The total number of bits that are uploaded is NF. The total bandwidth for uploading is $u_s + \sum_{i=k+1}^{N} u_i$ because only the server and the non-selfish peer perform upload. A lower bound for the download time is: $\frac{NF}{u_s+\sum_{i=k+1}^{N} u_i}$.

By combining with the two expressions that are given in the hint, we have

$$D_{\text{P2P}}^{\text{selfish users}} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=k+1}^{N} u_i}\right\} \tag{1}$$

Since $\frac{NF}{u_s+\sum_{i=k+1}^{N} u_i} \geq \frac{NF}{u_s+\sum_{i=1}^{N} u_i}$, the delay when there are selfish users will be no less than when all users upload.

Remark: The aim of this question is to test whether you understand how the expression $\frac{NF}{u_s+\sum_{i=1}^{N} u_i}$ is derived. If you do, you will be able to adapt the original solution to the new situation suggested in the question.

Question 6

(a) To begin with, the sender is at the "wait for ACK0" state and the receiver is at the "wait for 0 from below" state. Since the packet is lost, the receiver doesn't receive any packet and takes no action. When timeout occurs at the sender, it re-sends the packet and starts the timer. When the receiver receives this packet, it finds that the packet is not corrupted and has the correct sequence number, so it extract the data and deliver it to the upper layer. The receiver also makes an ACK packet with sequence number 0 to be delivered to the sender. The receiver then advances to the state "Wait for 1 from below." When the sender receives the ACK(0) packet, it advances to the "Wait for 1 from above state."

(b) The sender will be in the "wait for ACK 1" state and the receiver will be in the "wait for 1 from below" state after the sender has send the second packet the first time. When the receiver receives a corrupted packet, it sends an ACK packet with sequence number 0 to the sender. When the sender receives this ACK(0) packet, it finds that it is not corrupted and has sequence number 0, it does nothing. When timeout occurs later, and the sender resends the packet with sequence number 1. When the receiver receives the packet, it checks that it is not corrupted and the correct sequence number, so it extract the data and delivers it to the upper layer. The receiver also makes an ACK packet with sequence number 1 to be delivered to the sender. The receiver then advances to the state "Wait for 0 from below." When the sender receives the ACK(1) packet, it advances to the "Wait for 0 from above state."

Remarks: The aim of parts (a) and (b) is to test whether you can read finite state machine diagrams.

(c) Sequence number will not be needed in this case because no packets will be duplicated. To see this, there are only two possible scenarios here:

Scenario 1: the packet is delivered and ACK is received correctly.

Scenario 2: The data packet is lost and when the sender times out, the data packet is re-sent.

Note that under these scenarios, the receiver will never receive a duplicate packet, therefore, sequence number is not necessary.

Question 7

Go-back N: 15, 15, 15, 16

Selective repeat: 15, 18, 17, 16

Question 8

Source port: 2121. Destination port: 5123.

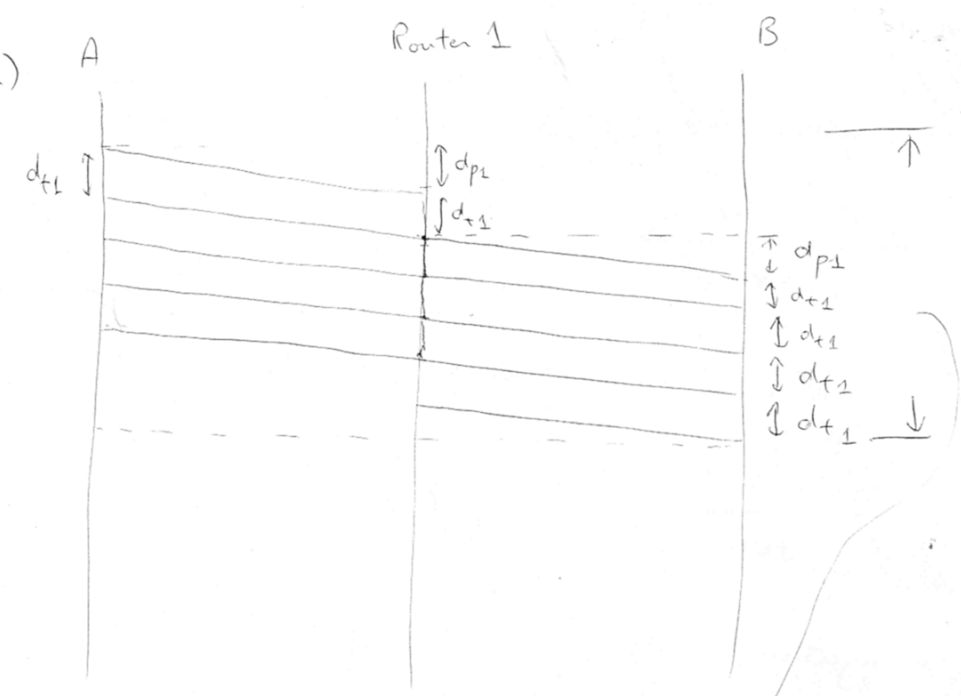
Question 9

(a) We know from the principle of checksum that the sum of all 16-bit words (when they are treated as ones complement numbers) in the entire segment is 0. If the source port is 1025, then the sum of all the other 16-bit words will be -1025. Since the binary representation of 1025 is 0000 1000 0000 0001, therefore if the least significant two bits are flipped, the received 16-bit word for the source port is 0000 1000 0000 0010, which is 1026. By adding 1026 and -1025, we get -1, which is not zero. Therefore the segment is corrupted and checksum is able to detect that.

(b) The sum of the source port number and destination port number for the uncorrupted packet is 1025+1026 = 2051. From the principle of checksum, we know that the sum of all the other 16-word integers in the segment should be -2051. If the least significant bit of 1025 is flipped, it becomes 1024. If the least significant bit of 1026 is flipped, it becomes 1027. At the destination, the checksum will check whether 1024 + 1027 - 2051 is zero. Indeed it is. Therefore, the destination host will think that this packet is uncorrupted. Therefore, checksum is not able to detect the error.

Remarks: Note that when there are two bit errors, the checksum can detect the error in some situations but it cannot detect it in some others. This question is test whether you can do that. The idea is that you start from the basic principle and work your way through the problem setting to answer the question.

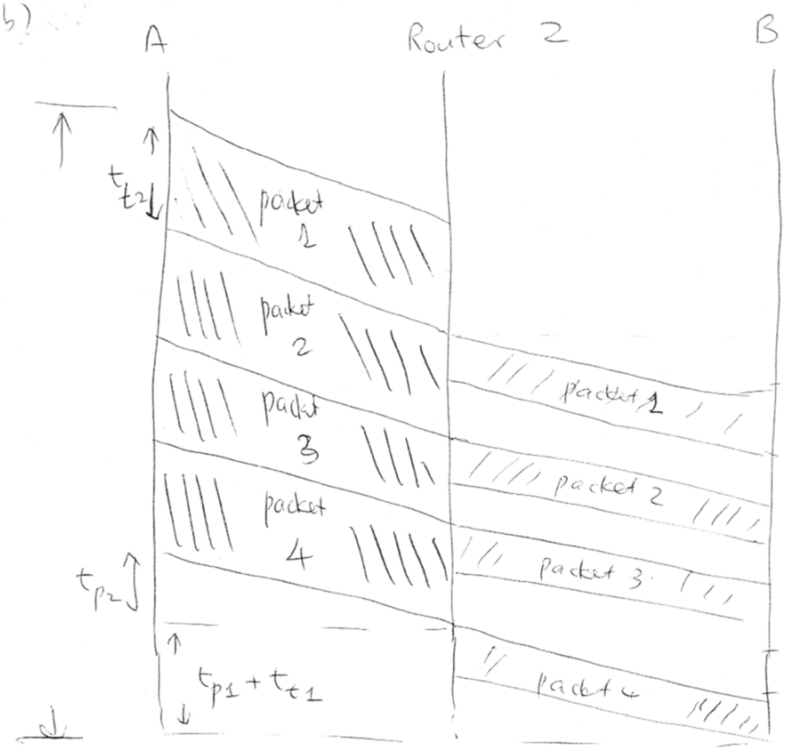
(a)



The time required

$= 2 d_{p1} + 5 d_{t1}$

(b)



The time required

$$= d_{p1} + t_{t1} + d_{p2} + 4\, t_{t2}$$