Computer Networks

Fall 2023/24
Exercise 2

Submission by Thursday, 21-1-2024. The submission is done by uploading your work to the course Moodle website. >>> No late submissions will be accepted!
The name of the submitted file must be Exercise1_fullname.zip.
For example, Exercise1_israel-israeli.zip. The first and last name of the student must appear.

Problem 1.

We consider the performance of HTTP, comparing non-persistent HTTP with persistent HTTP. Suppose the HTML page your browser wants to download is 20K bits longs, and contains 5 embedded images, each of 50K bits in length. The page is stored on server A and the 5 images are all stored on the server B. The RTT from your browser to server A is 500 msec. The RTT from your browser to server B is 200 msec. We will abstract the network paths between your browser and servers A and B as two 1000Mbps links ($1000*10^6$ bits per second). You can assume that the time it takes to transmit a GET message into the path (write it on the link) is zero, but you should account for the time it takes to transmit the base HTML file and the embedded objects into the link. In your answer below, make sure to take into account the time needed to setup up a TCP connection. Assume that your browser knows the exact IP address of the server in advance.

(a) Assuming non-persistent HTTP, with no parallel connections. What is the response time from the point when the user requests the page to the point in time when the page and its embedded objects are displayed?

(b) Again assume non-persistent HTTP, but now assume that, after the browser gets the HTML page, it can open as many parallel TCP connections to the server as it wants. (Assume that the each parallel connection gets full bandwidth.) What is the response?

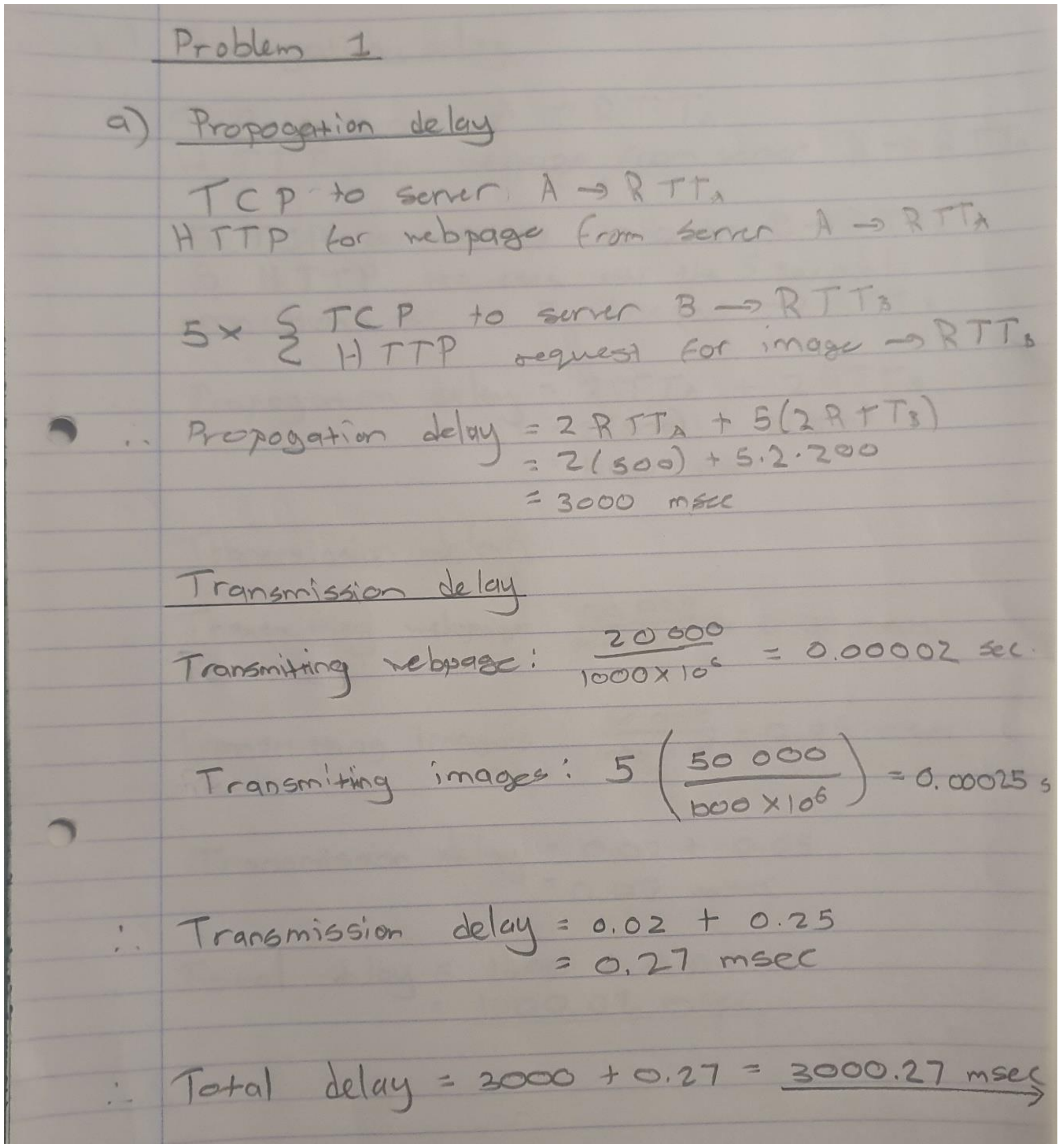
(c) Now assume persistent HTTP with no pipelining and with no parallel connections. What is the response time?

(d) Now assume persistent HTTP with pipelining and with no parallel connections. What is the response time?

Write your answers in the table below and justify your answers.

סעי ף	זמן תגוב ה
א	
ב	
ג	
ד	

Explanations & Calculations:



Problem 1

a) Propogation delay

TCP to server A $\rightarrow RTT_A$
HTTP for webpage from server A $\rightarrow RTT_A$

$5 \times \begin{cases} \text{TCP to server B} \rightarrow RTT_B \\ \text{HTTP request for image} \rightarrow RTT_B \end{cases}$

$\therefore$ Propogation delay $= 2RTT_A + 5(2RTT_B)$
$= 2(500) + 5 \cdot 2 \cdot 200$
$= 3000$ msec

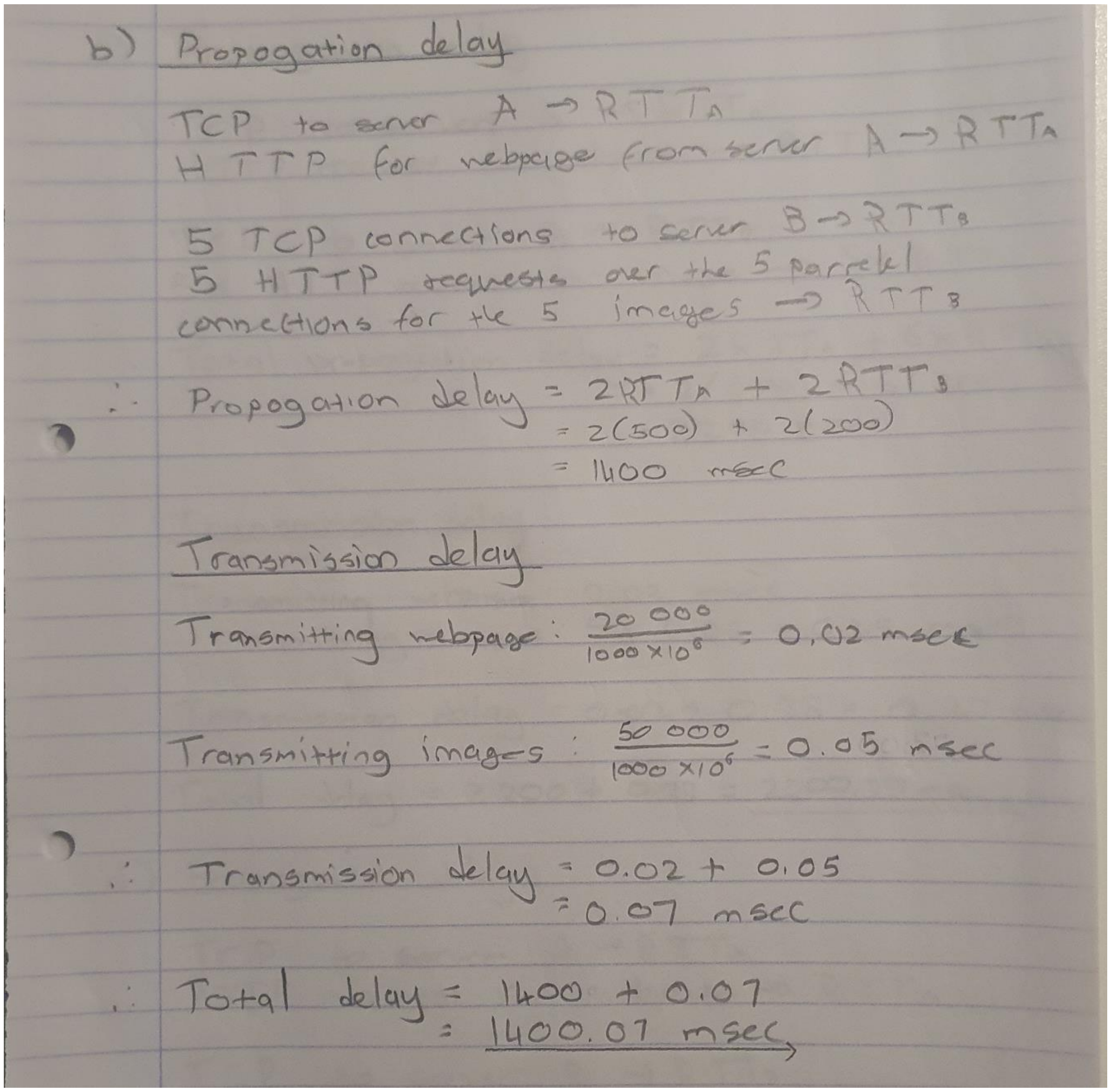
Transmission delay

Transmitting webpage: $\frac{20\,000}{1000 \times 10^6} = 0.00002$ sec.

Transmitting images: $5\left(\frac{50\,000}{1000 \times 10^6}\right) = 0.00025$ s

$\therefore$ Transmission delay $= 0.02 + 0.25$
$= 0.27$ msec

$\therefore$ Total delay $= 3000 + 0.27 = \underline{3000.27 \text{ msec}} \rightarrow$



b) Propogation delay

TCP to server A $\rightarrow RTT_A$
HTTP for webpage from server A $\rightarrow RTT_A$

5 TCP connections to server B $\rightarrow RTT_B$
5 HTTP requests over the 5 parrelel connections for the 5 images $\rightarrow RTT_B$

$\therefore$ Propogation delay $= 2RTT_A + 2RTT_B$
$= 2(500) + 2(200)$
$= 1400$ msec

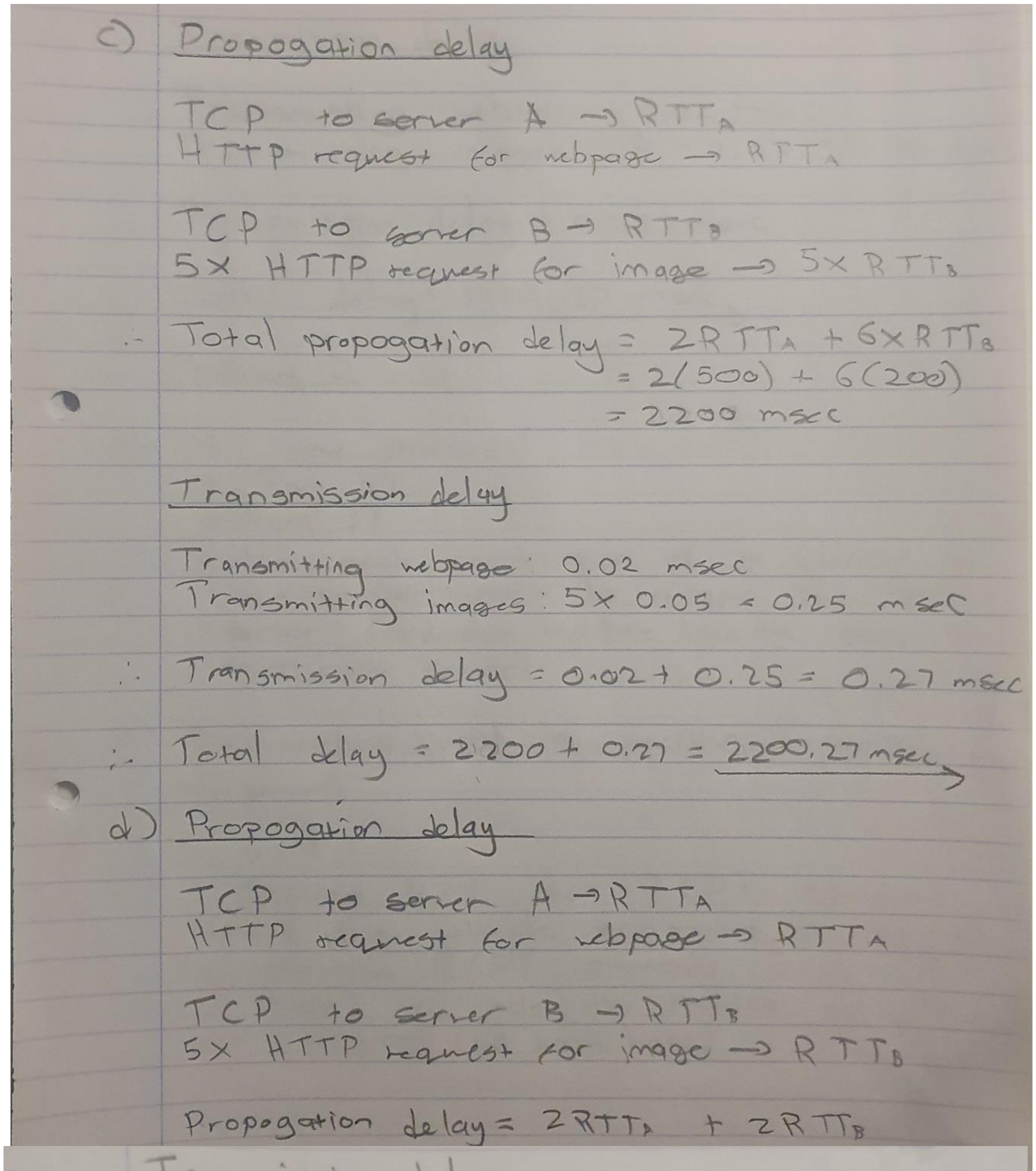
Transmission delay

Transmitting webpage: $\frac{20\,000}{1000 \times 10^6} = 0,02$ msec

Transmitting images: $\frac{50\,000}{1000 \times 10^6} = 0.05$ msec

$\therefore$ Transmission delay $= 0.02 + 0.05$
$= 0.07$ msec

$\therefore$ Total delay $= 1400 + 0.07$
$= \underline{1400.07 \text{ msec}}$



c) <u>Propogation delay</u>

TCP to server A $\rightarrow RTT_A$
HTTP request for webpage $\rightarrow RTT_A$

TCP to server B $\rightarrow RTT_B$
5x HTTP request for image $\rightarrow 5 \times RTT_B$

$\therefore$ Total propogation delay $= 2RTT_A + 6 \times RTT_B$
$= 2(500) + 6(200)$
$= 2200$ msec

<u>Transmission delay</u>

Transmitting webpage: 0.02 msec
Transmitting images: $5 \times 0.05 = 0.25$ msec

$\therefore$ Transmission delay $= 0.02 + 0.25 = 0.27$ msec

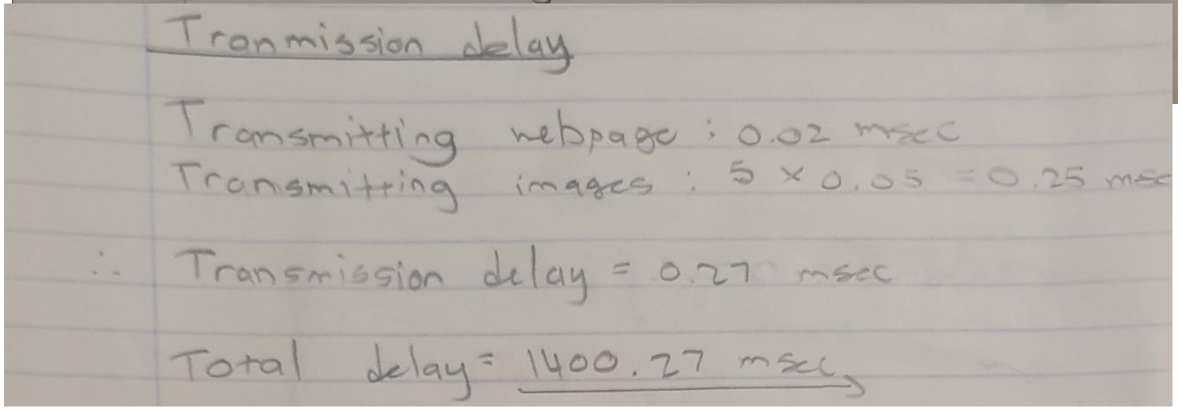
$\therefore$ Total delay $= 2200 + 0.27 = \underline{2200.27 \text{ msec}}$

d) <u>Propogation delay</u>

TCP to server A $\rightarrow RTT_A$
HTTP request for webpage $\rightarrow RTT_A$

TCP to server B $\rightarrow RTT_B$
5x HTTP request for image $\rightarrow RTT_B$

Propogation delay $= 2RTT_A + 2RTT_B$



<u>Tranmission delay</u>

Transmitting webpage: 0.02 msec
Transmitting images: $5 \times 0.05 = 0.25$ msec

$\therefore$ Transmission delay $= 0.27$ msec

Total delay $= \underline{1400.27 \text{ msec}}$

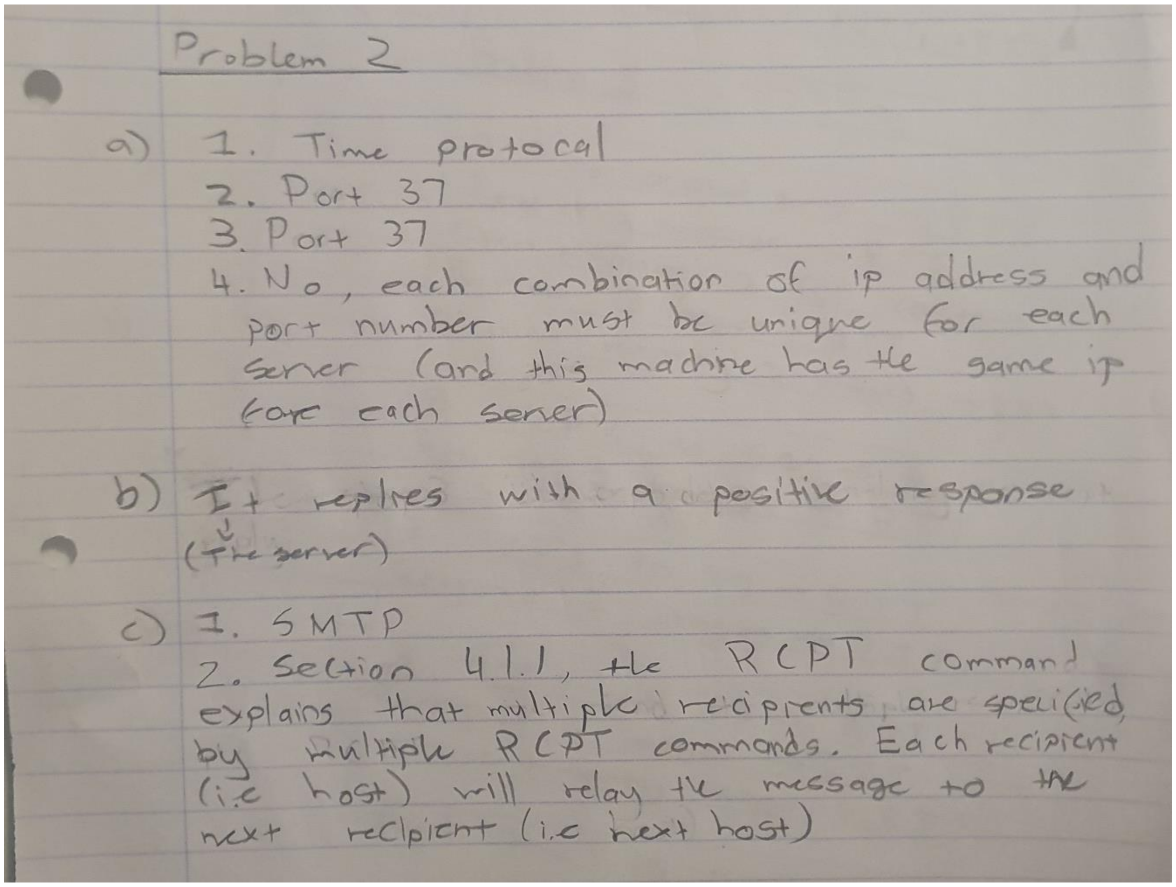
Problem 2.

The goal of this question is to get you to retrieve and read some RFCs.

(a)
1. Which protocol is specified in RFC 868?
2. On which port does the server, which is defined in RFC 868, listen when used via TCP?
3. On which port does the server, which is defined in RFC 868, listen when used via UDP?
4. Is it possible to run two servers on the same computer at the same time, one over UDP and the other over TCP using the same port number?

(b) What does the NOOP POP3 command in RFC 1939 do?

(c)
1. Which protocol is specified in RFC 821?
2. In what section of this RFC it is specified how to handle multiple recipients of the same message? Explain and reference to the RFC.



Problem 2

a) 1. Time protocal
2. Port 37
3. Port 37
4. No, each combination of ip address and port number must be unique for each server (and this machine has the same ip for each server)

b) It replies with a positive response
(the server)

c) 1. SMTP
2. Section 4.1.1, the RCPT command explains that multiple reciprents are specified by multiple RCPT commands. Each recipient (i.e host) will relay the message to the next recipient (i.e next host)

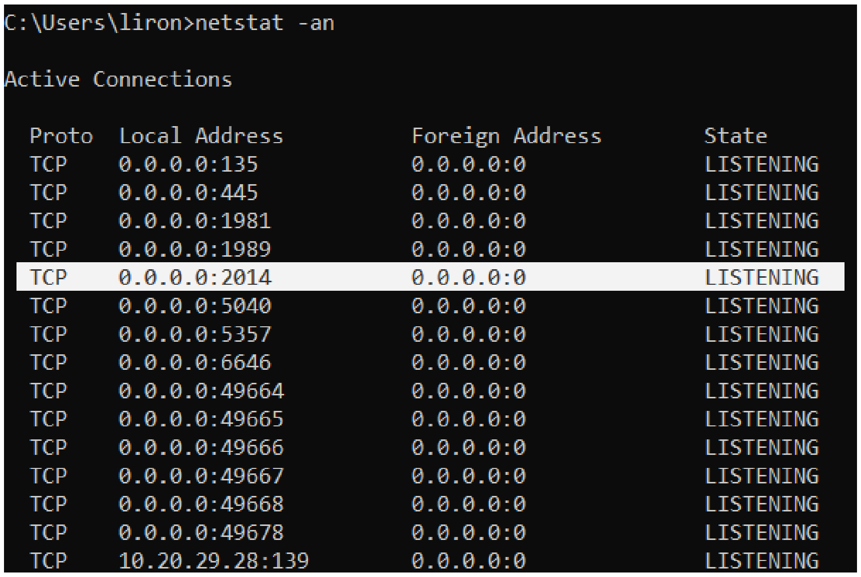
Problem 3.

A. Open netcat (i.e. ncat) and listen to port 2014 over TCP.
   i Use netstat command print the netcat's listening socket and take a print screen. Explain what is shown in the image.
   ii Connect with another netcat client to the netcat server. Use netstat to print the netcats sockets and take a print screen. Explain what is shown in the image.
   iii Connect another netcat to the server. Does it work? If yes, explain how, if not, why not?

B. Suppose you run the netcat server over TCP and connect with netcat client over UDP. The UDP client will send the data, is the data shown in the TCP server? Explain to support your answer.

C. Get ncat listening on port 3333 over TCP using the command ``ncat –v –k –l 3333". On the some machine, use your browser to access the URL:http://127.0.0.1:3333/ from two different Tabs. Your browser should now send two HTTP GET request messages to the ncat server, and your server should display the messages. Use ncat to return an HTTP 301 redirection response to the browser, to redirect the browser to http://www.runi.ac.il/. Are the two HTTP requests sent from the same (source) ports? Which of the two Tabs gets the response? Send us a screen capture of the browser's HTTP requests and ncat's HTTP response. Support your answers using netstat and explain what you see.

A i)



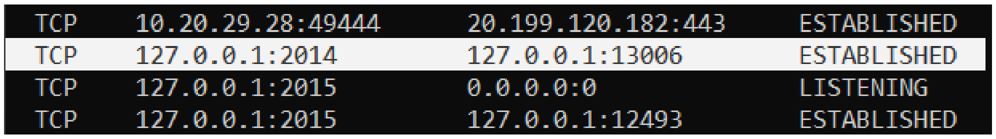
```
C:\Users\liron>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1981           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1989           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2014           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5040           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5357           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:6646           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49665          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49666          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49667          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49668          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49678          0.0.0.0:0              LISTENING
  TCP    10.20.29.28:139        0.0.0.0:0              LISTENING
```

This screenshot shows that there is a TCP connection on my machine (address 0.0.0.0) on port 2014 and is listening for any incoming requests.

ii)



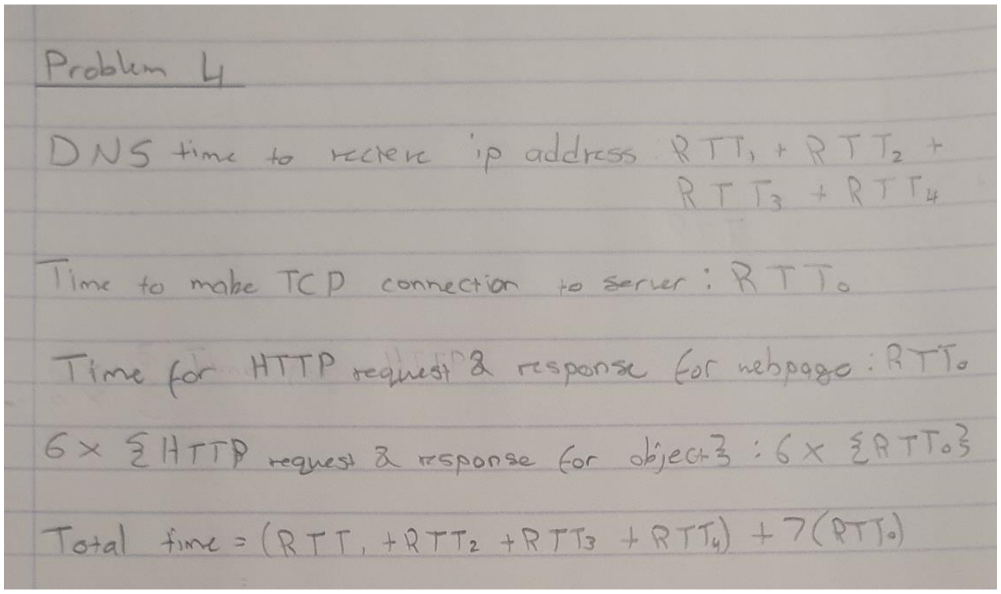
```
TCP    10.20.29.28:49444      20.199.120.182:443     ESTABLISHED
TCP    127.0.0.1:2014         127.0.0.1:13006        ESTABLISHED
TCP    127.0.0.1:2015         0.0.0.0:0              LISTENING
TCP    127.0.0.1:2015         127.0.0.1:12493        ESTABLISHED
```

This screenshot shows that there is an established connection between the netcat server on port 2014 and our client which is on port 13006 (both on ip address 127.0.0.1)
iii) No it does not work. This is because the port the second client is trying to connect to is already in use by another process (which is the active connection between the server and the first client)

B. No the data is not shows in the TCP server (the connection between the client and the server will be closed when attempting to connect with UDP). This is because there is a mismatch of protocol – the server is expecting TCP packets while the client is sending UDP packets. Since UDP and TCP operate differently, the server cannot understand UDP packets, it can only understand TCP packets since it's a TCP connection. For them to communicate they need to be using the same transport layer protocol.

Problem 4.

Suppose within your web browser you click on a link to obtain a Web page. Suppose that the IP address for the associated URL is not cached in your local host, so that a DNS look-up is necessary to obtain the IP address. Suppose that that four DNS servers are visited before your host receives the IP address from DNS; the successive visits incur an RTT of RTT1, ..., RTT4 . Further suppose that the Web page associate with the link contains exactly six objects, a small amount of HTML text file which indexes five very small objects on the same server. Let RTT0 denote the RTT between the local host and the server containing the objects. Assuming zero transmission time of the objects, how much time elapses from when the client clicks on the link until the client receives all the objects, assuming persistent HTTP without pipelining and without parallel TCP connections. Justify your answer.



Problem 4

DNS time to recieve ip address $RTT_1 + RTT_2 + RTT_3 + RTT_4$

Time to make TCP connection to server : RTT_0

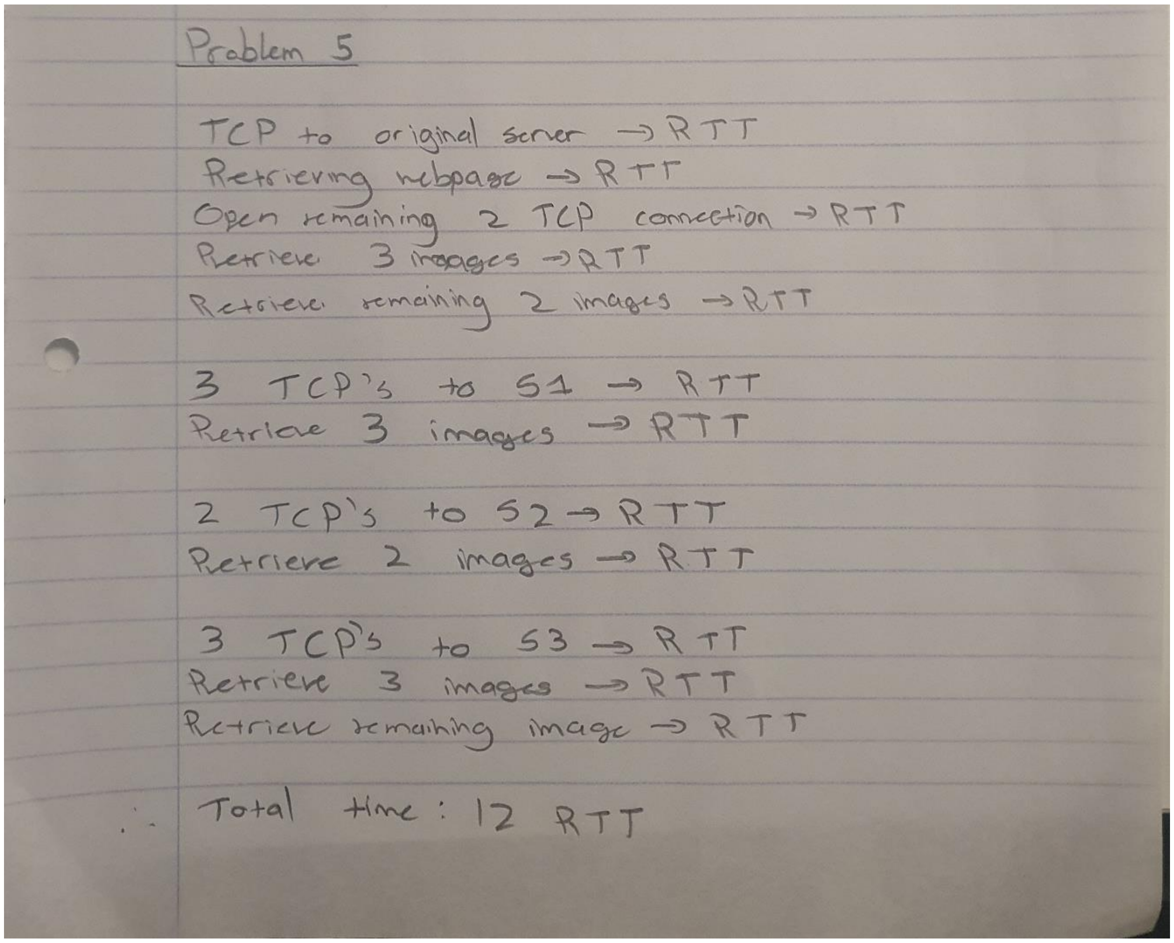
Time for HTTP request & response for webpage : RTT_0

$6 \times$ {HTTP request & response for object} : $6 \times \{RTT_0\}$

Total time = $(RTT_1 + RTT_2 + RTT_3 + RTT_4) + 7(RTT_0)$

Problem 5.

Consider a client that wants to retrieve a Web document at a give URL. The client initially knows the IP address of the Web server. The (html) document has 14 embedded GIF images. Exactly five GIF images reside at the same sever as the original document. Three more GIF image resides on server S1, two GIF images reside on server S2, and four GIF images reside on server S3. All the IP addresses of the servers appears explicitly in the html page. Suppose the time needed to contact and receive a reply from any server is one RTT. Assume that persistent connections without pipelining are used, and it is possible to use at most three parallel connections at the same time. How many RTT's are needed, in the best case, from when the user first enters the URL until the complete document (including the images) is displayed at the client? Justify your answer.



Problem 5

TCP to original server → RTT
Retrieving webpage → RTT
Open remaining 2 TCP connection → RTT
Retrieve 3 images → RTT
Retrieve remaining 2 images → RTT

3 TCP's to S1 → RTT
Retrieve 3 images → RTT

2 TCP's to S2 → RTT
Retrieve 2 images → RTT

3 TCP's to S3 → RTT
Retrieve 3 images → RTT
Retrieve remaining image → RTT

∴ Total time : 12 RTT

Problem 6.

Write a simple multi-threaded chat server using Java that runs over TCP listening to port 9922.

Once a new client connects to the server, the chat client receives from the server the text: "Welcome to RUNI Computer Networks 2024 chat server! There are N users connected." Where N are the number of clients connects (excluding the new one), so for the first client N=0, for the second N=1 and so on. When the user disconnects (i.e. its socket closes) the count decrements by one.
The other connected clients receives the message "[IP of new client] joined". For example, if IP 1.2.3.4 joins the conversation everyone will get the message "1.2.3.4 joined".

Once a client sends data (i.e. message) to the server, the server sends the data to all the other connected clients, adding the sender (IP:port) + colon before the message and CRLF at the end.
For example:
IP 1.2.3.4 sends the data "hey everyone!" from port 5425.

The other clients will see:
(1.2.3.4:5425): hey everyone!

For easy debugging, use netcat as your clients.
Submit a bash script compile.sh that compiles your server.
Submit a bash script run.sh that runs your server.

Important: Make sure your application doesn't crash. Use try-catch!