Computer Networks
Exercise 2

<u>Problem 1</u>

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*106 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 <u>non</u>-persistent HTTP, with <u>no</u> 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 <u>non</u>-persistent HTTP, but now assume that, after the browser gets the HTML page, it <u>can</u> 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 <u>no</u> pipelining and with <u>no</u> parallel connections. What is the response time?
d. Now assume persistent HTTP <u>with</u> pipelining and with <u>no</u> parallel connections. What is the response time?

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

זמן תגובה	סעיף
3000.27 ms	א
1400.07 ms	ב
2200.07 ms	ג
1400.27 ms	ד

<u>Explanations & Calculations:</u>

a. HTML fetch time:
   2 RTT to server A = $2 * 500 = 1000\,ms$
   L/R = $20,000/1,000,000,000 = 0.02\,ms$
   Image fetch time:
   2 RTT to server B = $2 * 200\,ms = 400\,ms$
   L/R = $50,000/1,000,000,000 = 0.05\,ms$
   HTML fetch time + 5 * (image fetch time) =
   $1000 + 0.02 + 5 * (400 + 0.05) = 3000.27\,ms$

b. HTML fetch time:

    2 RTT to server A = $2 * 500 = 1000\,ms$

    L/R = $20,000/1,000,000,000 = 0.02\,ms$

  Image fetch time:

    2 RTT to server B = $2 * 200\,ms = 400\,ms$

    L/R = $50,000/1,000,000,000 = 0.05\,ms$

  HTML fetch time + image fetch time (5 in parallel) =

  $1000 + 0.02 + 400 + 0.05 = 1400.07\,ms$

c. HTML fetch time:

    2 RTT to server A = $2 * 500 = 1000\,ms$

    L/R = $20,000/1,000,000,000 = 0.02\,ms$

  RTT to server B: $200ms$

  Image fetch time:

    RTT to server B = $200\,ms$

    L/R = $50,000/1,000,000,000 = 0.05\,ms$

  HTML fetch time + RTT to server B + 5 * (image fetch time) =

  $1000 + 0.02 + 200 + 5 * (200 + 0.05) = 2200.27\,ms$

d. HTML fetch time:

    2 RTT to server A = $2 * 500 = 1000\,ms$

    L/R = $20,000/1,000,000,000 = 0.02\,ms$

  Image fetch time:

    2 RTT to server B = $2 * 200 = 400\,ms$

    L/R = $5 * (50,000/1,000,000,000) = 5 * 0.05 = 0.25\,ms$

  HTML fetch time + Image fetch time =

  $1000 + 0.02 + 400 + 0.25 = 1400.27\,ms$

<u>Problem 2</u>

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.

a.

1. RFC 868 specifies the Time Protocol.
2. The server defined in RFC 868 listens over TCP on port 37.
3. The server defined in RFC 868 listens over UDP on port 37.
4. Yes, it is possible for two servers to use the same port number concurrently on the same computer over different protocols. This is because the protocols are different. Therefore, a server using TCP on a specific port and another using UDP on the same port can coexist without any conflict.

b. The NOOP POP3 command stands for "No Operation”. It serves as a simple way to check if the server is still responsive and maintains the connection alive.

c.

1. RFC 821 specifies the Simple Mail Transfer Protocol (SMTP).
2. The handling of multiple recipients of the same message is specified in Section 4.1.1.3, under the RECIPIENT (RCPT) command.
   Sending a message to multiple recipients is done by including multiple RCPT commands in the SMTP conversation, with each command specifying a different recipient.
   <u>RFC 821 RECIPIENT (RCPT)</u>

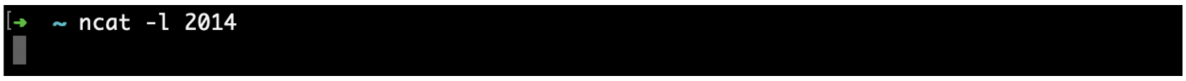
Problem 3

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.

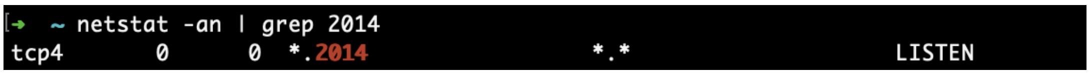
C. Get **ncat** listening on port 3333 over **TCP** using the command **"ncat –v –k –l 3333"**. On the some machine, use your browser to access the URL: http://127.0.0.1:3333/ from <u>two</u> different **Tabs**. Your browser should now send <u>two</u> HTTP GET request messages to the **ncat** server, 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. Listen to port 2014 over TCP.



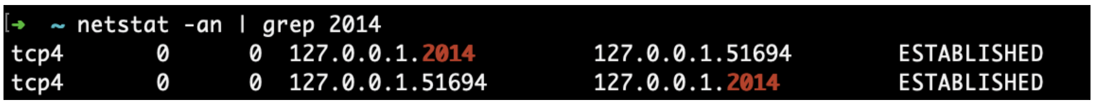
```
➜  ~ ncat -l 2014
```

   i. The server is listening over TCP in port 2014.



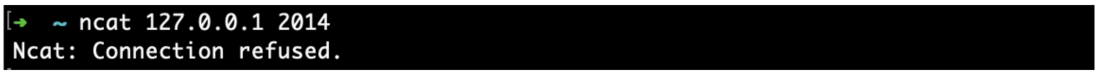
```
➜  ~ netstat -an | grep 2014
tcp4       0      0  *.2014                 *.*                    LISTEN
```

   ii. The first line is the server side of the TCP connection.
   The second line is the additional connected client.
   The TCP connection succeeded, hence the status is ESTABLISHED.



```
➜  ~ netstat -an | grep 2014
tcp4       0      0  127.0.0.1.2014         127.0.0.1.51694        ESTABLISHED
tcp4       0      0  127.0.0.1.51694        127.0.0.1.2014         ESTABLISHED
```

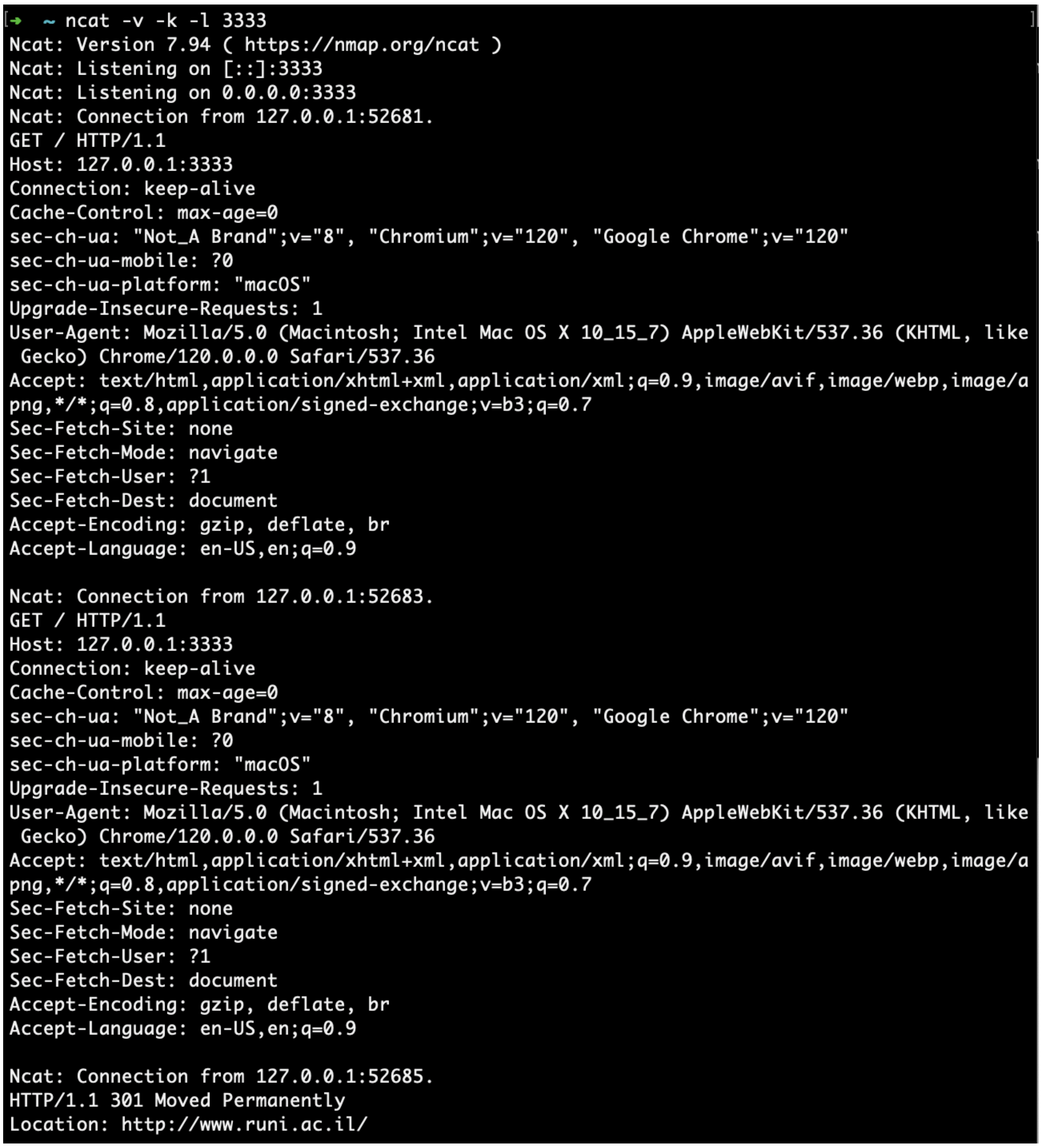
   iii. The connection refused by Ncat, which means we couldn't add another netcat to the server.
   This is because netcat in its basic form does not support handling multiple connections concurrently.



```
➜  ~ ncat 127.0.0.1 2014
Ncat: Connection refused.
```

B. No, the data sent by the UDP client will not be shown in the TCP server, and the communication will not be successful. This is because TCP and UDP are different protocols with distinct methods of handling data. For the communication to be successful, both the server and client should use the same protocol.

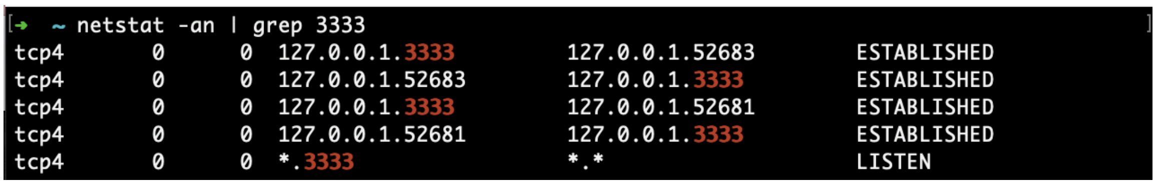
C. There are two HTTP GET request messages to the ncat server.
The HTTP requests were sent from different ports (52681, 52683).
Only one tab will get the response, and it will be the tab that has the last HTTP request that was received.



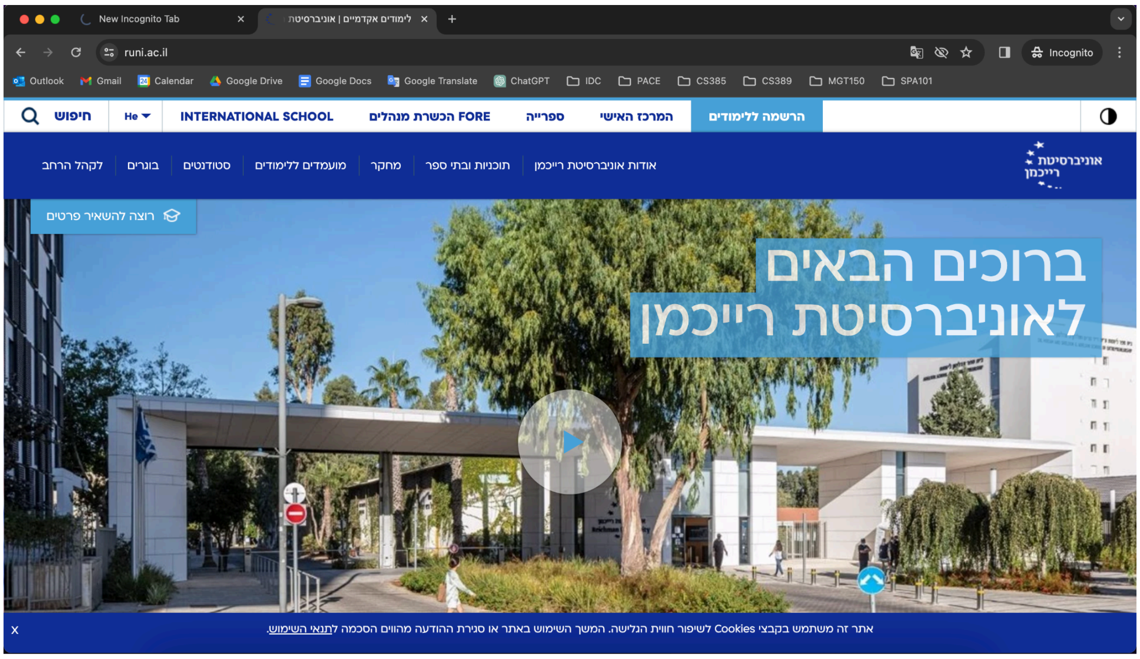
```
➜  ~ ncat -v -k -l 3333
Ncat: Version 7.94 ( https://nmap.org/ncat )
Ncat: Listening on [::]:3333
Ncat: Listening on 0.0.0.0:3333
Ncat: Connection from 127.0.0.1:52681.
GET / HTTP/1.1
Host: 127.0.0.1:3333
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120", "Google Chrome";v="120"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/120.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

Ncat: Connection from 127.0.0.1:52683.
GET / HTTP/1.1
Host: 127.0.0.1:3333
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120", "Google Chrome";v="120"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/120.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

Ncat: Connection from 127.0.0.1:52685.
HTTP/1.1 301 Moved Permanently
Location: http://www.runi.ac.il/
```



```
➜  ~ netstat -an | grep 3333
tcp4       0      0  127.0.0.1.3333         127.0.0.1.52683        ESTABLISHED
tcp4       0      0  127.0.0.1.52683        127.0.0.1.3333         ESTABLISHED
tcp4       0      0  127.0.0.1.3333         127.0.0.1.52681        ESTABLISHED
tcp4       0      0  127.0.0.1.52681        127.0.0.1.3333         ESTABLISHED
tcp4       0      0  *.3333                 *.*                    LISTEN
```



ברוכים הבאים
לאוניברסיטת רייכמן

אתר זה משתמש בקבצי Cookies לשיפור חווית הגלישה. המשך השימוש באתר או סגירת ההודעה מהווים הסכמה לתנאי השימוש.

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.

1. DNS lookup to obtain the IP address: $RTT_1 + RTT_2 + RTT_3 + RTT_4$
2. TCP between the localhost and the server: RTT_0
3. HTTP for HTML: RTT_0 (persistent HTTP).
4. HTTP for 5 objects: $5 * RTT_0$ (persistent HTTP).

Total time: $7 * RTT_0 + RTT_1 + RTT_2 + RTT_3 + RTT_4$

<u>Problem 5</u>
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.

1. TCP with S_0.
2. HTTP for HTML.
3. HTTP for GIF_1 from S_0.
   TCP with S_2.
   TCP with S_3.
4. HTTP for GIF_2 from S_0.
   HTTP for GIF_1 from S_2.
   HTTP for GIF_1 from S_3.
5. HTTP for GIF_3 from S_0.
   HTTP for GIF_2 from S_2.
   HTTP for GIF_2 from S_3.
6. HTTP for GIF_4 from S_0.
   HTTP for GIF_3 from S_3.
   TCP with S_1.
7. HTTP for GIF_5 from S_0.
   HTTP for GIF_4 from S_3.
   HTTP for GIF_1 from S_1.
8. HTTP for GIF_2 from S_1.
   HTTP for GIF_3 from S_1.

In the best case, 8 RTTs are needed.

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!