Computer Networks - Exercise 2

Noy Mizrahi, 313296071
Prague Oshri, 204868046

Problem 1

1. Consider a client that wants to retrieve a Web document at a given URL. The client initially knows the IP address of the Web server. The (html) document has 12 embedded images. Exactly two images reside at the same sever as the original document, and the other 10 images reside at 6 different servers. The client initially knows all the IP addresses of the servers. Suppose the time needed to request and receive a reply from any server is one RTT. Assume that persistent connections without pipelining, and without parallel connections are used. How many RTT's are needed from when the user first enters the URL until the complete document (including the images) is displayed at the client?

   Answer:

   We start to open a TCP to server and then request the html page. After we get the html page we know we need to request for the 12 images. Number of RTT according to the requests order:

   (a) TCP to server.
   (b) html from server.
   (c) Image 1 from the same server (no new TCP connection needed).
   (d) Image 2 from the same server (no new TCP connection needed).
   (e) For each server from 1-6 open TCP connection. Total 6 RTT's.
   (f) We request for each image from 3-12 (total 10 images) on TCP connection that already exist at each servers (the 6 from stage (e)). Total 10 RTT's.

   Total we get $1_{(a)} + 1_{(b)} + 1_{(c)} + 1_{(d)} + 6_{(e)} + 10_{(f)} = 20$ RTT's.

2. Consider a client that wants to retrieve a Web document at a given URL. The client initially knows the IP address of the Web server. The (html) document has 12 embedded images. Exactly two images reside at the same sever as the original document. Two more images reside on server S1, two images reside on server S2, five images reside on server S3, and one image resides on server S4. The client initially knows all the IP addresses of the servers. 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 two 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?

Answer:

First we start a TCP connection with the server and get the html page from it. For the best case we implement the parallel connection therefore, we connect two servers at the same time.

num of RTT's	Original Server	S3	S1	S2	S4
1	TCP				
2	html				
3	Image1	TCP			
4	Image2	Image7			
5		Image8	TCP		
6		Image9	Image3		
7		Image10	Image4		
8		Image11		TCP	
9				Image5	TCP
10				Image6	Image12

Total 10 RTT's at the best case.

Problem 2

The goal of this question is to get you comfortable retrieving and reading some RFCs.

a) Which protocol is specified in RFC 7230; in what section upgrade is defined; is it used by the client-side, by the server-side or by both?

Answer: Hypertext Transfer Protocol (HTTP/1.1); upgrade at section 6.7; both client and server used it.

b) Which protocol is specified in RFC 1035; in what section CNAME is discussed?

Answer: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION, know as DNS protocol; CNAME at section 3.3.1.

c) What is the title of RFC 3596; in what section the AAAA record type is defined.

Answer: Title is DNS Extensions to Support IP Version 6; AAAA record type at 2.1 section.

d) Which protocol is specified in RFC 821? In what section of this RFC it is specified how to handle multiple recipients of the same message? Explain.

Answer: SIMPLE MAIL TRANSFER, know as SMTP protocol; this subject at section 4 (also mentioned at section 2 which gives reference to section 4).

e) Which protocol is specified in RFC 8484? What is the minimum recommended version of HTTP, and in what section is it discussed?

Answer: DNS Queries over HTTPS protocol; HTTP/2 is the minimum recommended version of HTTP at section 5.2.

f) Which protocol is specified in RFC 8310; in what section the port 853 is first mentioned. Is the use of port 853 an advantage or a disadvantage?

Answer: Usage Profiles for DNS over TLS and DNS over DTLS protocol; port 853 is not mentioned in RFC 8310 but in RFC 7858 it first mentioned at section 3.1.

g) Which protocol is specified in RFC 7231; in what section the status code that indicates that the client's request was successfully received is discussed?
Answer: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content protocol; the status code that indicates that the client's request was successfully received is discussed at section 6.3.2.

h) What is the full name of RFC 6797? In which section the HSTS Mechanism is discussed?
Answer: The full name is HTTP Strict Transport Security (HSTS); HSTS Mechanism discussed at section 5.

Problem 3

You can download the Windows version of ncat from http://nmap.org/ncat/ (direct link: http://nmap.org/dist/ncat-portable-5.59BETA1.zip). Your Anti-Virus software may identify ncat as a virus; in such a case, disable your anti-virus (ncat is not a virus).
Use ncat (or nc or netcat, whatever variant that your OS supports) to perform the following:

1) Get ncat listening on port 2020 over TCP (this is the server-side).

2) On a different or the same computer, run client ncat to connect to the ncat server that is listening on port 2020 over TCP (127.0.0.1 is IP of the "same computer").

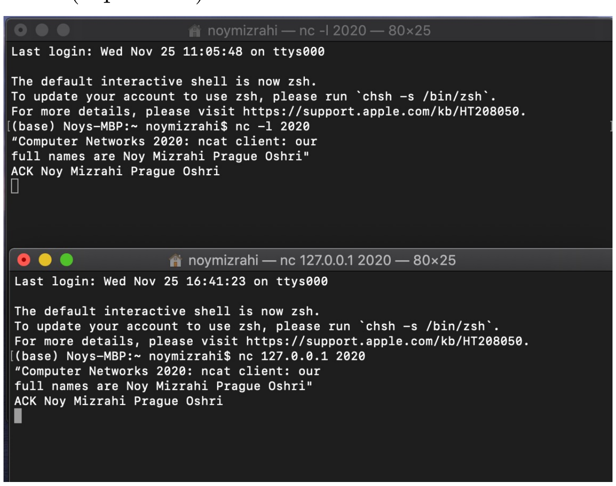
3) From the client, ncat sends the string "Computer Networks 2020: ncat client: ourvfull names are <your-name1><your-name2>" – make sure that the string is being received by the server ncat.

4) From the server, ncat send back the string "ACK <your-name1><your-name2>" – make sure that the string is being received by the client ncat.

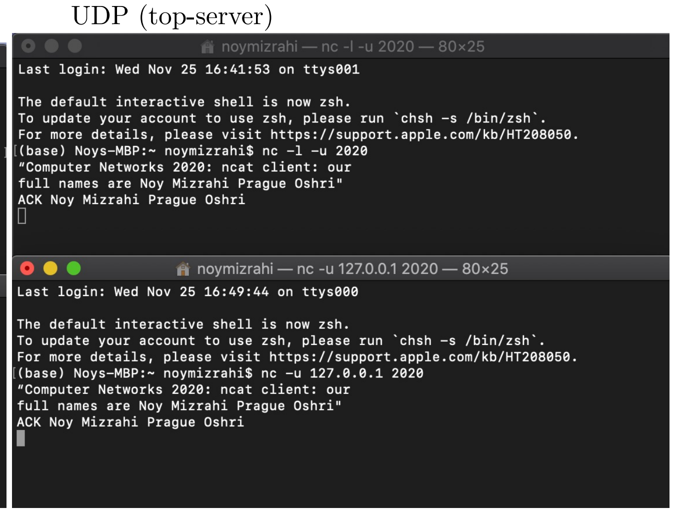
5) Repeat steps 1-4 over UDP.
Tip: use "ncat –help" to see the available command line.

a) Take screen captures of all 4 windows (for both UDP and TCP). The screenshots must contain the ncat commands used and the strings that have been sent and received. Make sure you state which screenshot is of the client and which is of the server.
Answer:

TCP (top-server)

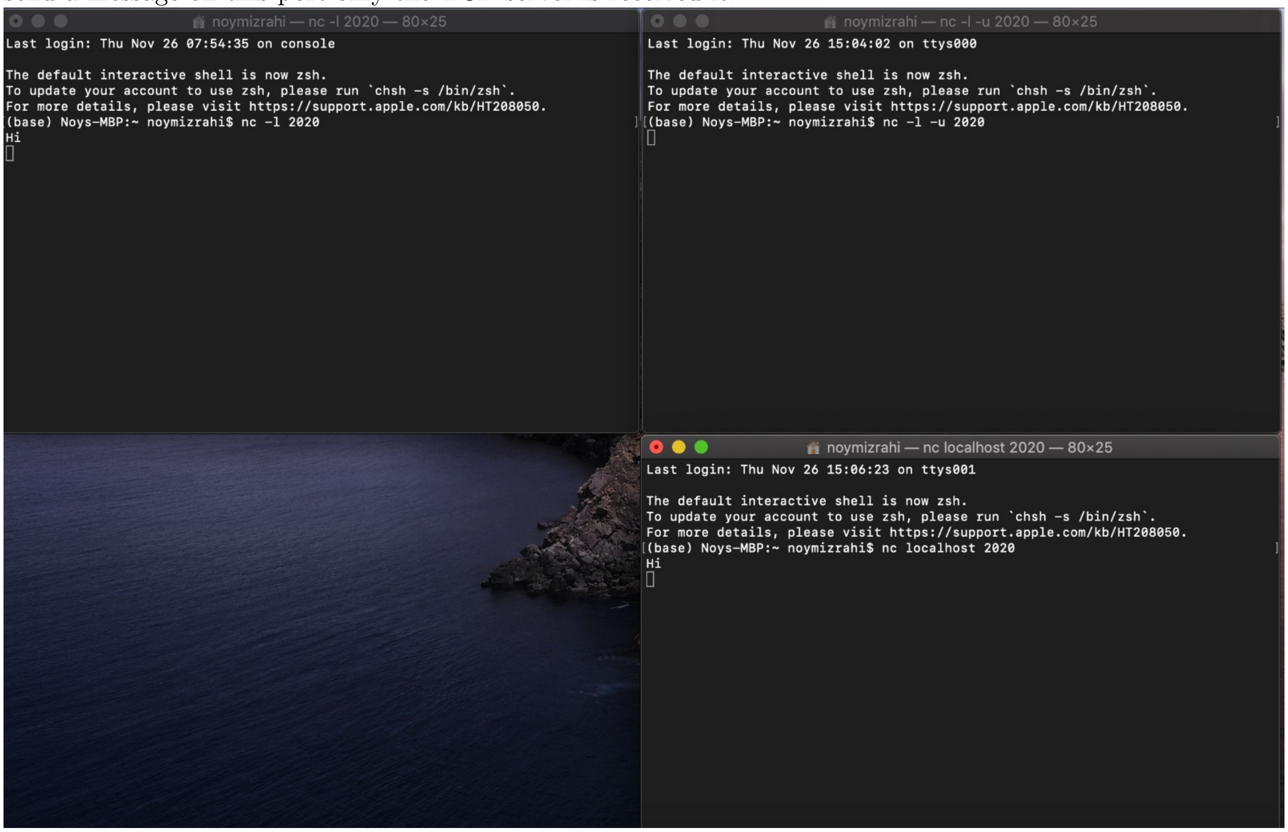


UDP (top-server)



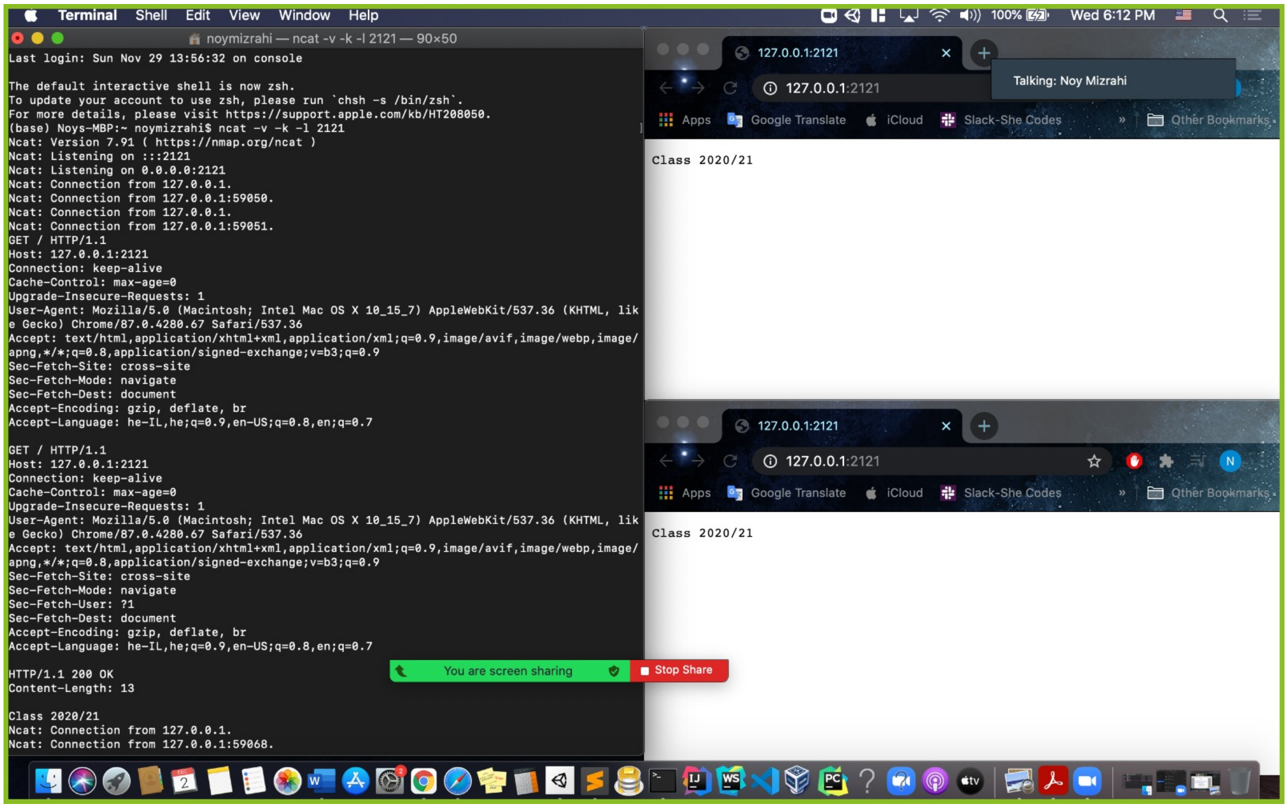
b) Is it possible to run two servers on the same computer at the same time, one over UDP and the other over TCP, using the same port number? Check it using ncat.

Answer: Yes it is possible to listening on using the same port number. However, when a client try to send a message on this port only the TCP server is received it.



c) Get ncat listening on port 2121 over TCP using the command "ncat –v –k –l 2121". On the same machine, use your browser to access the URL:http://127.0.0.1:2121/ 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 the string "Class 2020/21". Are the two HTTP requests are 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.

Answer:



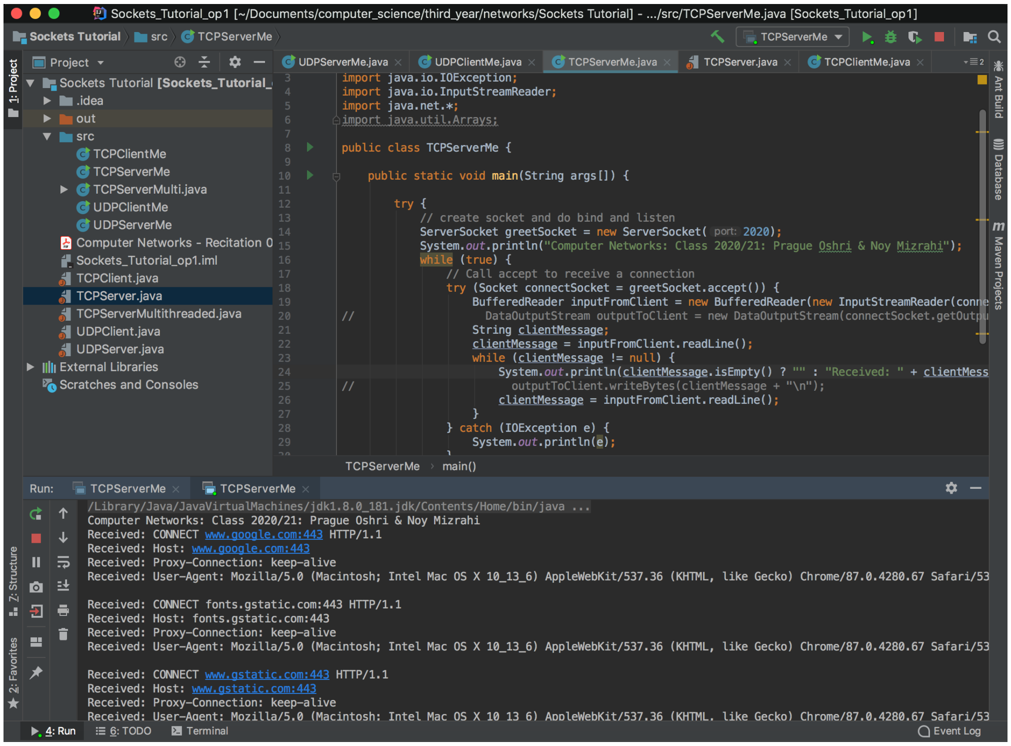
The two HTTP requests didn't have the same port and both responded.

Problem 4

Write a simple server program that runs over TCP using port 2020, that accepts lines of input from a client, and first prints "Computer Networks: Class 2020/21: <YOUR STUDENT names>" and then all the accepted lines onto the server's standard output.

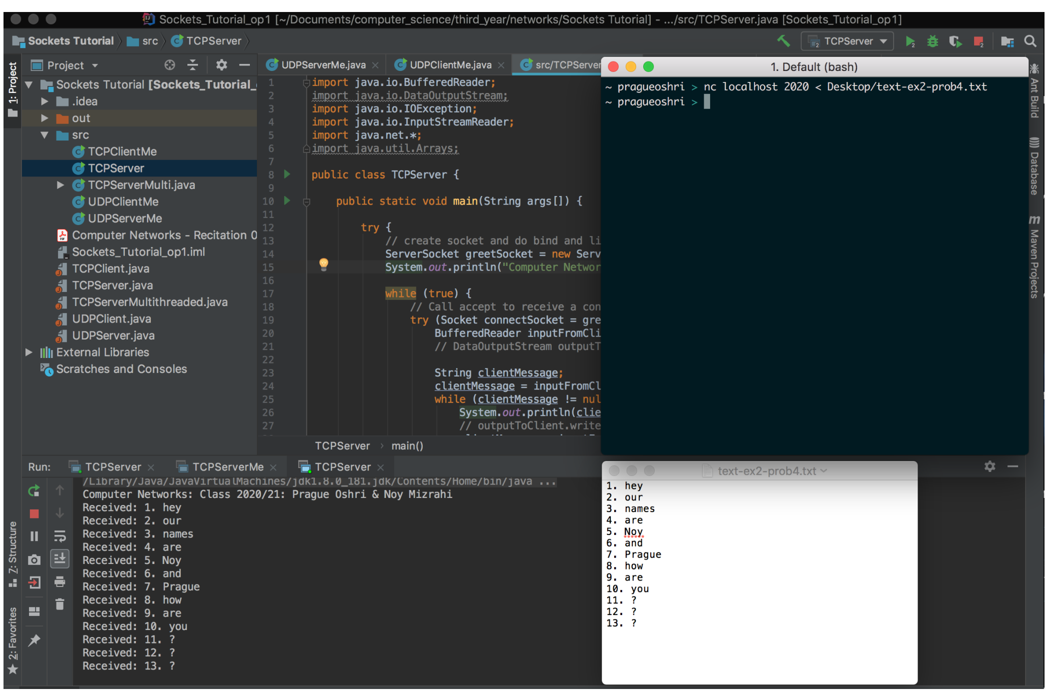
1. Set the proxy server information in your browser to the host that is running your server program, and also configure the port number appropriately. Using your browser, try access the URL: http://www.google.com. Your browser should now send either a GET request message OR a CONNECT request to your server, and your server should display the message. (To understand why your server got a CONNECT message, you may read about HTTP Strict Transport Security (HSTS)).

Answer:



2. On the same machine, run a ncat client to connect to your server listening on port 2020 over TCP. Use the ncat client to send a message of precisely 13 lines to your server. Your server should display the messages.

   Answer:



Problem 5 - Lecture #5

1. Consider the consensus algorithm that solves the consensus problem in a synchronous (message-passing) system in which at most t processes fail only by crashing. The time complexity of the algorithm presented was t + 1 rounds. Recall that a crashing failure is different from a Byzantine failure, and that in the crashing failure case, it is required that the final decision value be the input of some process (i.e., not necessarily the input of a non-faulty process). Consider a synchronous system in which at most t processes fail by almost-clean crashes, that is, in a round a crashed process either sends all its messages, all its messages except one or none (and after it crashes it stops participating in the algorithm). Design an algorithm that solves consensus as fast as possible in this model (i.e., with as few rounds as possible). Try that your answer be short that is, around 7 lines. Start your answer by writing clearly what is the number of rounds of the algorithm.

   Answer:

   Algorithm: **#rounds = 2**

   repeat at each round
     round := round + 1
     if flag = true then send v to all processes wait to receive all messages for this round
     temp := v
     v := minimum among all received values at this round and current value of v
     if v < temp then flag := true else flag := false
   until round = 2
   decide on v

   In this algorithm it is enough to do only 2 rounds. The worst case is when at the first round all the minimum values are fail process by almost-clean crashes, and them send all their messages except to the same process, which its value is not the minimum. Hence, this process doesn't have the minimum value. Therefore, at the second round there two options:
   - Like the first round, we left with only one process.
   - There exist a process with the minimum value which sends the minimum to the only left process without it.

2. Consider the algorithm for binary consensus in the case of byzantine failures. Suppose $n = 29$ and $t = 8$. (Note that in this scenario, it does not hold that $n \geq 4t + 1$). Assume that all the correct processes start with the same input value 0. Is it possible that all the correct processes will reach an agreement at the end and that the value of the agreement will be 1? (Recall that s = n-t = 21.) Justify your answer.

   Answer:

   No, it is not possible since we start when all the correct processes are 0, means that we never choose a leader. According to the algorithm, at step (1) we received s or more messages with the same value - 0 we assign this value to v.