**Exercise I.a.1** The listening socket is bound to a specific address. What address is this? (Give both the symbolic name used in the code, and the corresponding IPv4 address in numeric or dotted notation).

Ans. The listening socket is bound to the INADDR\_ANY which is the symbolic name of the address, to which the server is bound. The corresponding IPv4 address is 0.0.0.0. It is not a routable IP address, but it ask the servers to bind all network interfaces, which might have different IP addresses and start listening on them.

**Exercise I.a.2** In the code, there is a call to recv() as follows:

ret = recv( cd.sock, cd.buffer, kTransferBufferSize, 0 );

The return value ret will be one of the following:

(a) ret = −1 : Error has occurred on receiving a packet and will lead to socket shutdown.

(b) ret = 0 : Socket has been closed by the client.

(c) 0 < ret <kTransferBufferSize : The operation of the function recv() worked as expected, however the size of the message is lesser than the total available message size. Due to this the remaining bytes until (kTransferBufferSize - ret) bytes will contain garbage values.

(d) ret = kTransferBufferSize : The operation of the function recv() work as expected and the message is read from the buffer.

Describe the implications of each case! Additionally, why is cd.buffer (see ConnectionData declaration) defined to be of size kTransferBufferSize+1 rather than just plain kTransferBufferSize?

And: cd.buffer is zero terminated which is not counted by ret due to which we need to manually update the count.

**Exercise I.a.3** Sending is performed using the send() method as follows: ret = send( cd.sock,

cd.buffer+cd.bufferOffset, cd.bufferSize-cd.bufferOffset, MSG\_NOSIGNAL

);

How does the send() method indicate that the connection in question has been closed/reset? How does MSG\_NOSIGNAL relate to this (on linux machines)?

Ans: If the connection has been closed or reset the send() method will return the value -1 and the sets the error to ECONNRESET.

MSG\_NOSIGNAL requests not to send SIGPIPE(A SIGPIPE is sent to a process if it tried to write to a socket that had been shutdown for writing or isn't connected (anymore)), when the connection has been closed or reset. A SIGPIPE will kill the entire server process. However an EPIPE error will still be received.

**Exercise I.a.4** Discuss the reasons for this behaviour with your partner. Why are these two strategies used?

Also, quickly look through the error codes (values of errno) possible after accept(), send(), and recv() (check the man-pages!). Under which conditions attempting to continue execution might be unreasonable?

Ans: There is no point for the server to even start listening to the socket if a socket bind fails, possibly because the port is already in use or if the IP address is incorrect, as no data would ever reach the process. It is advantageous for the server to close the socket while receiving packets if the client closed the socket on its own or if it is sending packets that are producing problems on the server in order to save resources and prevent the errors from crashing the entire system. It is unreasonable to continue execution when the network route is missing, the host is not reachable, or the provided address is not available.

**Exercise I.b.1** Discuss with your partner: How is the program notified that a connection attempt has failed or succeeded?

Hint: the process is described in the course book!

ANS: When the connection to server fails, the program notifies us with the response connect() failed. If the socket is terminated prematurely prior to establishing the connection, the program will notify us with socket() failed ”socket() failed”.

**Exercise I.c.1** Try to send messages with each of the clients. Describe the results – do you receive a response immediately?.

Check with *netstat* and document the status of the connection from each client.

When two clients are connected, the message sent from first client(desktop-0q665bm.59986) is sent immediately and the response is received. However for the message from the second client(souptiks-mbp.49256), the response is not received.

Proto Recv-Q Send-Q Local Address Foreign Address (state)

tcp4 2 0 souptiks-mbp.5703 souptiks-mbp.49256 ESTABLISHED

tcp4 0 0 souptiks-mbp.49256 souptiks-mbp.5703 ESTABLISHED

tcp4 0 0 souptiks-mbp.5703 desktop-0q665bm.59986 ESTABLISHED

Exercise I.c.2 When you disconnected the first client, what happened? Explain why.

When the first client (desktop-0q665bm.59986) is disconnected the, the message from the second client is sent and the response is received immediately. As both the clients are connected concurrently in the same socket, the server is only able to communicate with one client at a time.

**Exercise I.c.3** Measure the round trip time when the client and server are running on the same machine. Also measure the round trip time when they are on different machines.

Can you observe any differences? Write down the times. (Note: take the average of a few (> 5) attempts.)

Client and Server running on the same machine:

Average time(5 attempts) on sending a message on 5 bytes: 0.1975332 ms

Client and Server running on the different machine:

Average time(5 attempts) on sending a message on 5 bytes: 70.689747 ms

The a considerable difference between the response times when the client and server are running on the same machine and then on different systems.

**Exercise I.c.4** Measure the round trip times for two concurrently connected simple clients (similar to exercise I.c.1 ).

Discuss with your partner: What is the largest factor in the measured round trip time of the second client?

ANS : When two client are connected concurrently, the round trip time for the first client was 76.321878 ms and the round trip time for the second client was 97184.621615. The largest factor in the round trip time for the second client is the WAIT TIME for receiving the response, after the first client is disconnected.

**Exercise I.d.1** Run the above command (make sure that the server is still running), and note the results.

souptikpaul@Souptiks-MBP VS Code % ./client-multi 192.168.0.107 5703 7 255

Simulating 7 clients.

Establishing 7 connections...

successfully initiated 7 connection attempts!

Connect timing results for 7 successful connections

- min time: 0.365417 ms

- max time: 0.547042 ms

- average time: 0.416655 ms

(0 connections failed!)

Roundtrip timing results for 7 connections for 255 round trips

- min time: 22.617541 ms

- max time: 61.421916 ms

- average time: 43.938726 ms

**Exercise I.d.2** Take note of the timing results. You will want to compare them to results in the next Lab/Exercise.

ouptikpaul@Souptiks-MBP VS Code % ./client-multi 192.168.0.107 5703 10 255

Simulating 10 clients.

Establishing 10 connections...

successfully initiated 10 connection attempts!

- conn 0 : error in recv() : Connection reset by peer

- conn 1 : error in recv() : Connection reset by peer

Connect timing results for 10 successful connections

- min time: 0.446792 ms

- max time: 0.583333 ms

- average time: 0.507950 ms

(0 connections failed!)

Roundtrip timing results for 8 connections for 255 round trips

- min time: 23.243625 ms

- max time: 66.467292 ms

- average time: 47.015510 ms

souptikpaul@Souptiks-MBP VS Code % ./client-multi 192.168.0.107 5703 15 255

Simulating 15 clients.

Establishing 15 connections...

successfully initiated 15 connection attempts!

- conn 3 : error in recv() : Connection reset by peer

- conn 4 : error in recv() : Connection reset by peer

- conn 5 : error in recv() : Connection reset by peer

- conn 6 : error in recv() : Connection reset by peer

- conn 7 : error in recv() : Connection reset by peer

- conn 8 : error in recv() : Connection reset by peer

Connect timing results for 15 successful connections

- min time: 0.372083 ms

- max time: 0.706917 ms

- average time: 0.463097 ms

(0 connections failed!)

Roundtrip timing results for 9 connections for 255 round trips

- min time: 22.510583 ms

- max time: 71.094250 ms

- average time: 49.113546 ms

(You don’t have to hand in the results, though.)

**Exercise I.d.3** How long did it take for the connection attempts to time out?