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).

INADDR_ANY -> 0.0.0.0

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:

- ret = -1 : error while performing recv()
- ret = 0 : no message available
- ret = 0 < ret < bufferSize : length of message in bytes
- ret = bufferSize : : length of message is buffer size, maybe there is an other recv() call needed because the received message is too long

bufferSize+1 : ConnectionData.buffer is 1 byte longer than the real buffer size to store the '\0' character at the end of the message (this is the « null » character, it is used to signify the end of a string).

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)?

SIGPIPE is a socket signal which, if not handled, will cause the process to exit. Then, the process must catch the signal to avoid being involuntary terminated. So there is specific ways to do this:

- On Linux, you can add the « MSG_NOSIGNAL » parameter to the send() function : it requests not to send the SIGPIPE signal, and you can still catch the error by checking the errno value of the socket which must be « EPIPE », then you can admit that the connection is finished.
- On macOS, you can add the « SO_NOSIGPIPE » parameter to the setsockopt() function (which is handled while the socket setup).

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?

If the server crashes during setup, the server won't work at all, there is no sense if the process stays alive. However, if an error occurred while a client connection has been established, the server can probably continue its communications with other clients. It might be unreasonable to continue execution if the errno values of the socket are EBADF, EFAULT, EINTR, EMFILE, ENFILE, ENOBUFS, ENOMEM, ENOTSOCK, EOPNOTSUPP, EPERM, or EPROTO.

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

If a connection attempt has failed, the file descriptor of the connection socket will be equal to « -1 », you can find the error on the « errno » string variable. Otherwise, if the connection succeeded, the socket file descriptor will be equal to 0 (no message), or a value between 1 and the « bufferSize » variable (there is a message).

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

The first connected client is able to immediately receive a response, the second connected client have to wait the first client to be disconnected to get a response. Each client connection in « netstat » is marked as an « established » connection state.

Exercise I.c.2 When you disconnected the first client, what happened? Explain why. After disconnecting the first client, the second client received the answer he was waiting for. The server cannot communicate with multiple clients due to the socket being in « blocking mode » by default, so the second client was waiting for his connection to be accepted.

Exercice 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.)

When they running on the same machine, the round trip time is between 0.1 and 0.15 ms.
I setup the server on Raspberry Pi and the client on PC, and the PC and Raspberry Pi are both connected to a router. In this case, the round trip time is around 3 ms in average, but sometimes up to tens of ms.

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?

The largest factor is the time interval from the second client sending out the message to the first client closing the connection.

Exercise I.d.1
Simulating 7 clients.
Establishing 7 connections...
  successfully initiated 7 connection attempts!
Connect timing results for 7 successful connections
  - min time: 0.124623 ms

- max time: 0.291248 ms
- average time: 0.180483 ms

(0 connections failed!)
Roundtrip timing results for 7 connections for 255 round trips
- min time: 25.229094 ms
- max time: 146.315785 ms
- average time: 85.847773 ms

Exercise I.d.2
Simulating 100 clients.
Establishing 100 connections...
successfully initiated 100 connection attempts!
Connect timing results for 100 successful connections
- min time: 0.950009 ms
- max time: 1032.983030 ms
- average time: 929.572446 ms

(0 connections failed!)
Roundtrip timing results for 100 connections for 255 round trips
- min time: 22.867836 ms
- max time: 54676.523240 ms
- average time: 7553.827229 ms

Simulating 7 clients.
Establishing 7 connections...
successfully initiated 7 connection attempts!
Connect timing results for 7 successful connections
- min time: 0.090800 ms
- max time: 0.198374 ms
- average time: 0.123030 ms

(0 connections failed!)
Roundtrip timing results for 7 connections for 10000 round trips
- min time: 942.050240 ms
- max time: 5835.153086 ms
- average time: 3468.632647 ms

Exercise I.d.3
In 2 minutes.