Esra Eryılmaz
171044046

CSE 344 – System Programming

Final Project Report

16.06.2023

1) System Architecture

- In this project we were suppose to implement simplified version of Dropbox.

- The system consists of two main components: a server and clients.

- **The server** listens for incoming connections, reads client requests and writes back to the clients via sockets.

- **Clients** connect to the server using sockets and each client is assigned a separate thread for communication.

Server and Clients
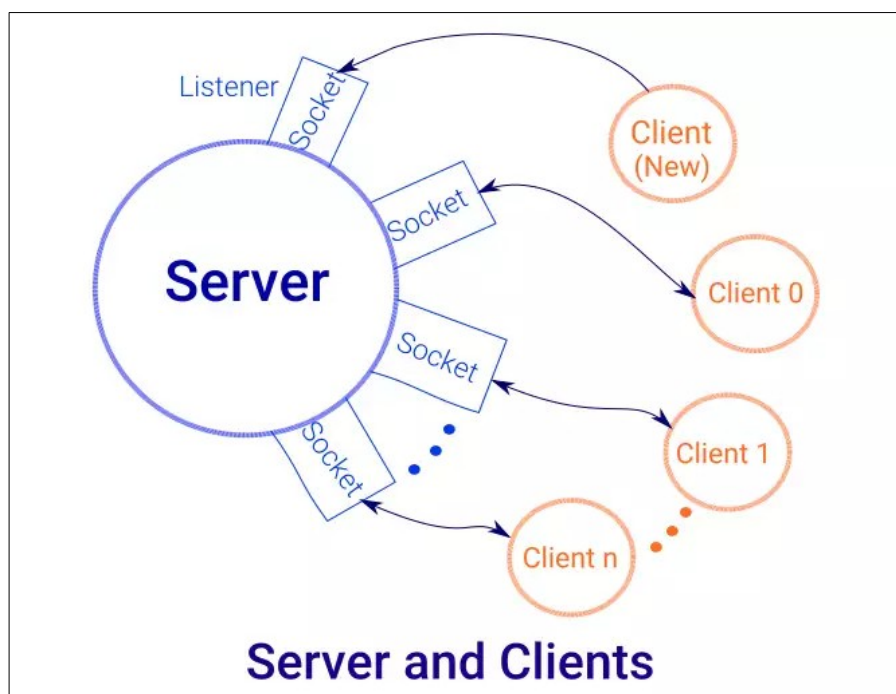
2) Design Decisions and Implementation Details

General Information :

- On both sides, I am keeping the files inside the directories using the following file struct :

```c
typedef struct{
    char name[512];
    char content[4096];
    mode_t mode;    // Permissions
    long size;
    time_t modifiedTime;
} file;
```

- The algorithm I implemented to ensure that the directories be synchronized can be summarized as follows:
  - In both sides, I continuously perform read() and write() operations through sockets within a loop.
  - Firstly, I check if there are any changes in the server directory. If there are no changes, I send -1 to indicate that. If there are changes, I send the byte count of the directory content and then send all the files within the directory to the client via the socket. After the client finishes reading everything, it sends an "ok" message, which the server reads, and the program continues.
  - Then, the same process is performed by the client. This way, they synchronize their directories with each other.

    (You can review this structure based on the draft design I have drawn on the two pages later.)

BibakBOXServer.c

- It takes input from the user,
- Creates a socket and starts listening for incoming connections,
- When a connection is received, it sends all the files in the server directory to the client initially for synchronization at the beginning,
- The respective thread terminates when the client disconnects.
- When a SIGINT signal is received, the server shuts down.
- The server is responsible for writing the log file.

BibakBOXClient.c

- It takes input from the user,

- It has an optional parameter [server_address] which allows connecting to a different computer by providing the IP address,

- It creates a socket and connects to the server,

- When a SIGINT signal is received or the server is closed, the client also shuts down.

Makefile

- It compiles the project as a whole.

3) Which requirements I met / did not meet?

- A thread pool structure has been implemented in the project.

- The signal SIGINT is working on both sides.

- When there is a connection, the connection address is printed to the screen on both sides.

- I have performed error checks as much as possible. The usage is printed to the screen.

- It works for the created, deleted and updated changes of the files on both sides.

- I keep the files in the directory in a struct file that I created on both sides. I send the files through sockets mutually. That's why it works on different computers as well.

- The program works fine with a single client, but when multiple clients are connected, it may not always work properly. (I used a mutex as a synchronization mechanism but sometimes certain clients fail to access the necessary information coming from server.)

- If directories contain subfolders, I couldn't take that part into account due to the time constraints.

- There is logging, but not exactly as requested. I keep the log file directly in the project's working directory, not in the server or client directory. And inside the log file, I only write that a change occurred on the side where the change occurred whenever there is a change in any directory.

threadMark

Server

```
while( ){
    accept()
    totalClient++;
    readFolderContent()
    fileCount
    write(& total_bytes)

    while(remaining >>){
        write(fileList---)
    }
    read("ok")

    while( ){
        compare Files(--
        if(choice){            no
            write(-1);
            read("ok");
        else{
            memset(FileList)
            fileCount = read Folder
            write(& total_bytes)
            while(remaining- >>){
                write(fileList)
            }
            read("ok");
        }

        /*----------------*/

        read(& received-byte)
        if(received == -1){
            write("ok");
        }
        else{
            while(remaining >0){
                read(receive Buff)
            }
            re create Folder()
            fileCoun = read Folder Conter
            write("ok");
        }
    }

    totalClient--

}
```

server
choice 109

Client

Main

```
empty-dir()
socket()
connect()
read(& received-bytes)
while(remaining-){
    read(receive Buf + tota...)
}

re Create Folder
fileCount = read Folder Content
write("ok")

while(){
    read(& received-byte)
    if(received == -1){
        write("ok");
    }
    else{
        while(remaining -bytes >0){
            read(receive Buff)
        }
        write("ok");
    }

    /*---------------------*/

    compare Files(---)
    if(change){          no
        write(-1);
        read("ok");
    }
    else{
        memset(File..)
        fileCount = read Folder Content(
        write(& total bytes)
        while(remaining byte){
            write(fileList + tot--)
        }
        read("ok");
    }
}
```
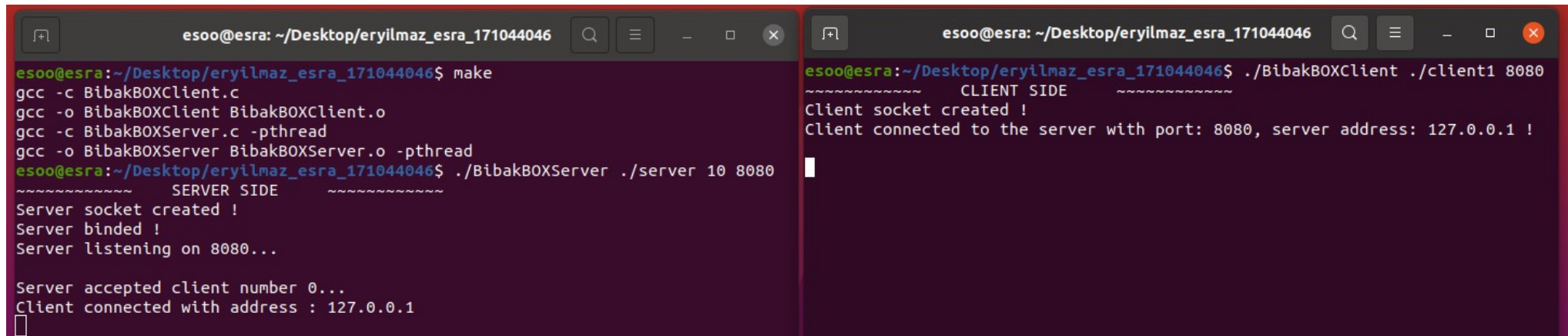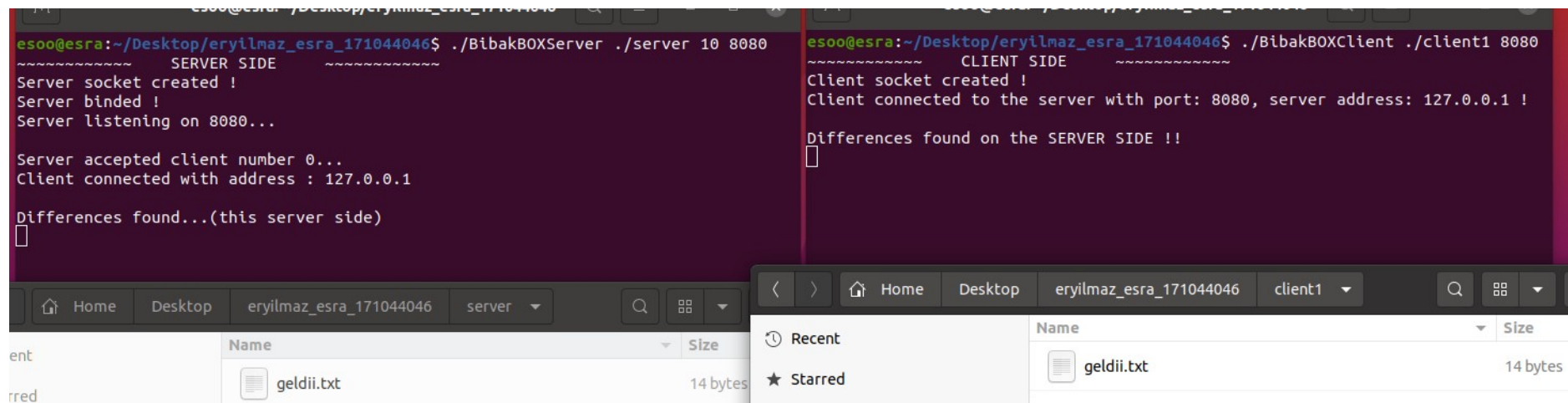
V2

4) Tests

- Usage :

- Test case 1 : The server was started and then 1 client connected. A file was created in the server directory and it was also reflected in the client directory.

- Test case 2 : The server was started and then 1 client connected. A file was deleted in the client directory and it was also reflected in the server directory.

- Test case 3 : The server was started and then 1 client connected. A file was updated in the server directory and it was also reflected in the client directory.

- Test case 4 : The server was run with a thread pool size of 10 and 10 clients were run in parallel. Communication has been established but the clients do not always communicate properly with the server.


```
esoo@esra:~/Desktop/eryilmaz_esra_171044046$ ./BibakBOXServer ./server 10 8080
~~~~~~~~~~~     SERVER SIDE      ~~~~~~~~~~~
Server socket created !
Server binded !
Server listening on 8080...

Server accepted client number 0...
Client connected with address : 127.0.0.1

Server accepted client number 1...
Client connected with address : 127.0.0.1

Server accepted client number 2...
Client connected with address : 127.0.0.1

Server accepted client number 3...
Client connected with address : 127.0.0.1

Server accepted client number 4...
Client connected with address : 127.0.0.1

Server accepted client number 5...
Client connected with address : 127.0.0.1

Server accepted client number 6...
Client connected with address : 127.0.0.1

Server accepted client number 7...
Client connected with address : 127.0.0.1

Server accepted client number 8...
Client connected with address : 127.0.0.1

Server accepted client number 9...
Client connected with address : 127.0.0.1
```
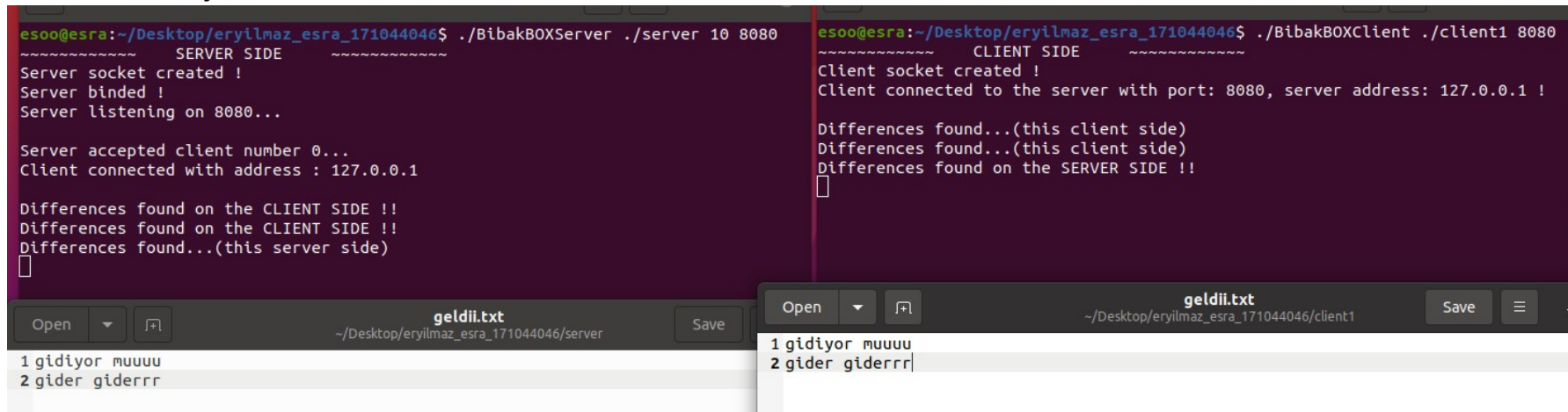
```
esoo@esra:~/Desktop/eryilmaz_esra_171044046$ bash test.sh
~~~~~~~~~~~     CLIENT SIDE      ~~~~~~~~~~~
Client socket created !
~~~~~~~~~~~     CLIENT SIDE      ~~~~~~~~~~~
Client socket created !
Client connected to the server with port: 8080, server address: 127.0.0.1 !

~~~~~~~~~~~     CLIENT SIDE      ~~~~~~~~~~~
Client connected to the server with port: 8080, server address: 127.0.0.1 !

Client socket created !
~~~~~~~~~~~     CLIENT SIDE      ~~~~~~~~~~~
Client socket created !
~~~~~~~~~~~     CLIENT SIDE      ~~~~~~~~~~~
Client socket created !
~~~~~~~~~~~     CLIENT SIDE      ~~~~~~~~~~~
Client socket created !
~~~~~~~~~~~     CLIENT SIDE      ~~~~~~~~~~~
Client socket created !
esoo@esra:~/Desktop/eryilmaz_esra_171044046$ ~~~~~~~~~~~      CLIENT SIDE~
~~~~~~~~~~~
Client socket created !
~~~~~~~~~~~     CLIENT SIDE      ~~~~~~~~~~~
~~~~~~~~~~~     CLIENT SIDE      ~~~~~~~~~~~
Client socket created !
Client socket created !
Client connected to the server with port: 8080, server address: 127.0.0.1 !

Client connected to the server with port: 8080, server address: 127.0.0.1 !

Client connected to the server with port: 8080, server address: 127.0.0.1 !

Client connected to the server with port: 8080, server address: 127.0.0.1 !

Client connected to the server with port: 8080, server address: 127.0.0.1 !

Client connected to the server with port: 8080, server address: 127.0.0.1 !

Client connected to the server with port: 8080, server address: 127.0.0.1 !

Client connected to the server with port: 8080, server address: 127.0.0.1 !
```
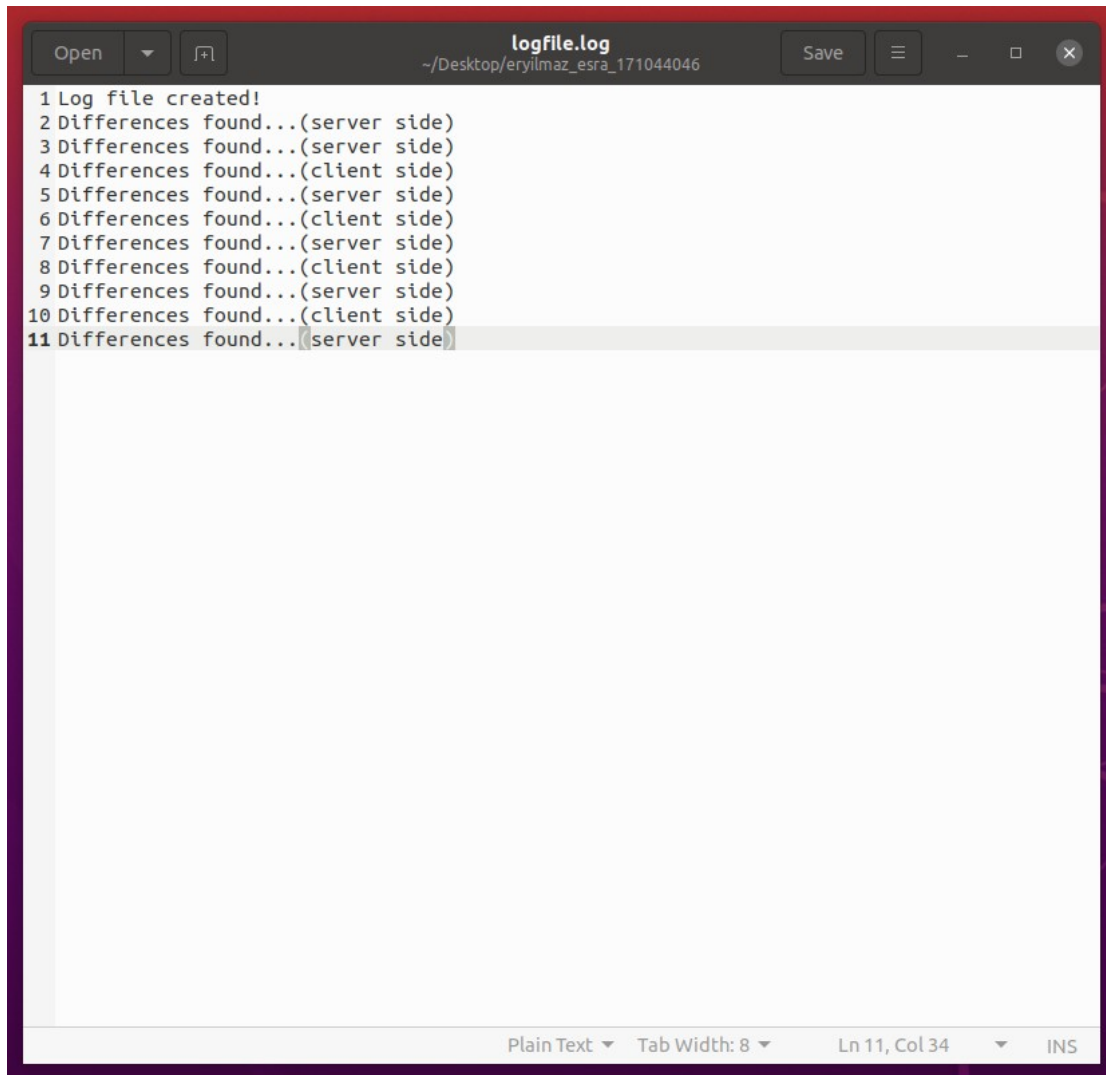
test.sh
```bash
1  #!/bin/bash
2
3  for i in {1..10}
4  do
5      ./BibakBOXClient client"$i" 8080 &
6  done
```

- Test case 5 : The server was run with a thread pool size of 20, and 2 clients were run consecutively. Communication was established and it was observed that the file operations were done correctly.

- Test case 6 : The server was started and then 1 client connected. Various operations were performed in both the server and client directories and the contents of the log file at the end are as follows:
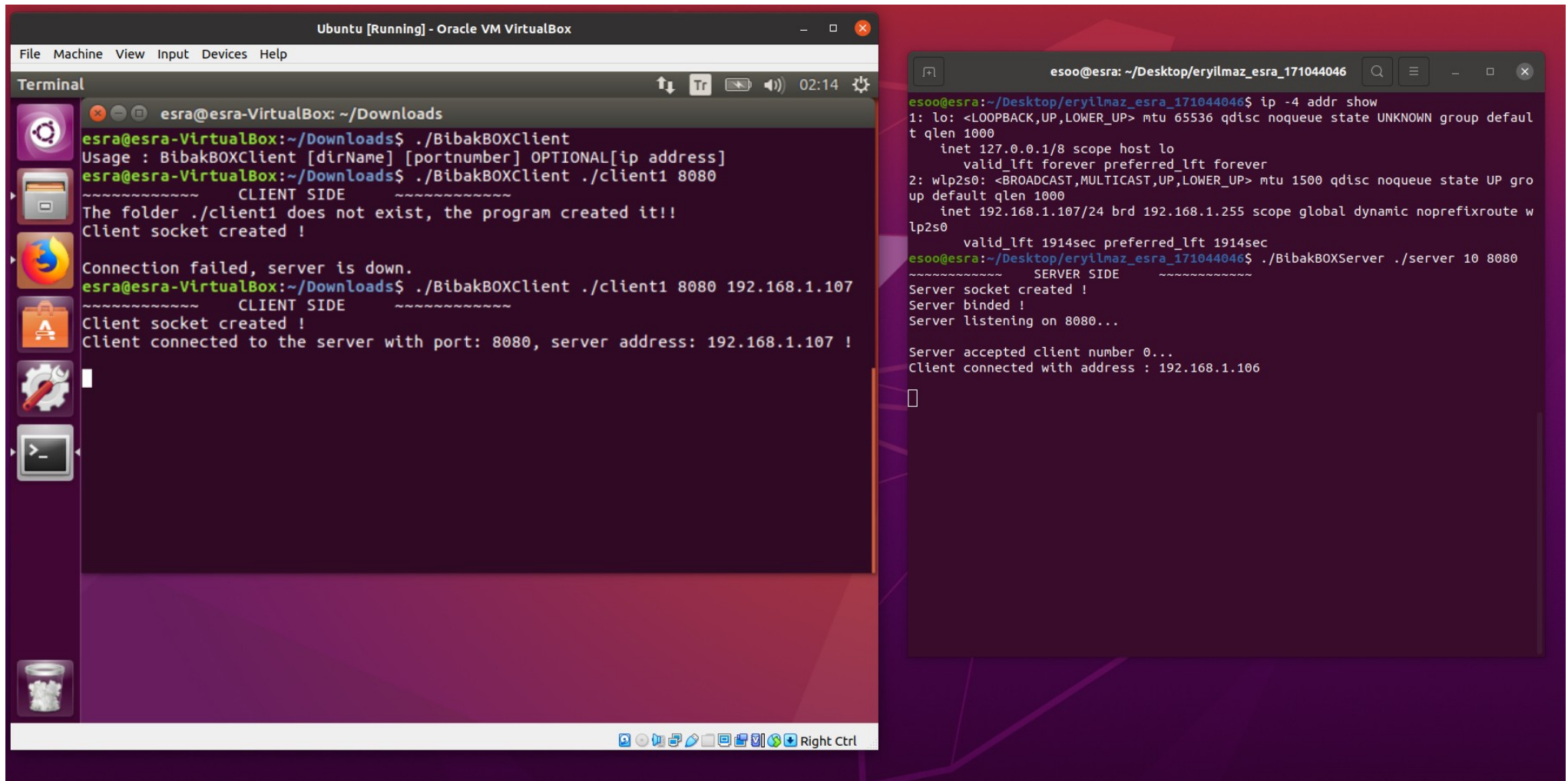

```
1 Log file created!
2 Differences found...(server side)
3 Differences found...(server side)
4 Differences found...(client side)
5 Differences found...(server side)
6 Differences found...(client side)
7 Differences found...(server side)
8 Differences found...(client side)
9 Differences found...(server side)
10 Differences found...(client side)
11 Differences found...(server side)
```

- Test case 7 : The server and client were run on different computers:

Firstly, the connection was established :

Afterwards, a file was created on the client side and it was tested whether the file reached the server :

- Test case 8 : Signal test