Report

Assignment 3

Author: LOIC GALLAND / LEONARDO PEDRO
Email: lg222sv@student.lnu.se / lr222qp@student.lnu.se
Semester: Spring 2020
Area: Computer Science
Course code: 1DV701

Contents

1 Summary of workload

The workload for this assignement was done [**50% lg222sv**] / [**50% lr222qp**]

2 Problem 1

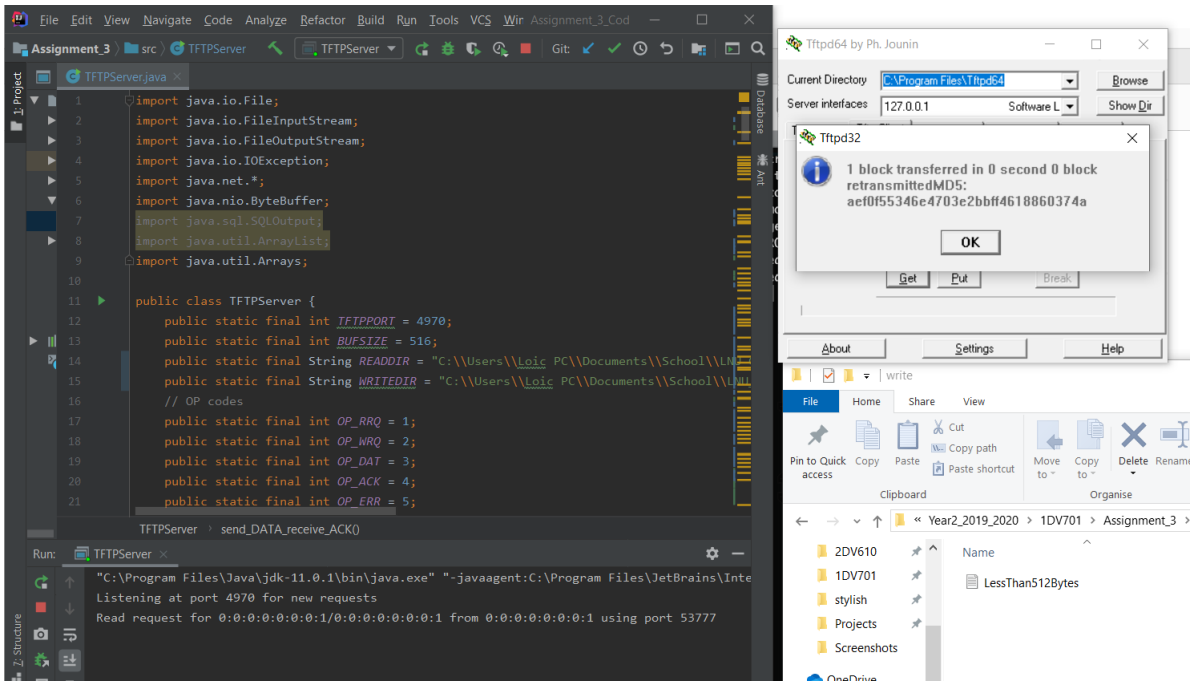

Figure 1: Read request to the TFTP server

2.1 Discussion

After successfully reading the requests, examine the TFTPServer starter code once more and explain in your report why it uses both ***socket*** *and* ***sendSocket***

The **socket** is being used to create a connection between server and client. **sendSocket** is used to send packets between the client and the server.

3 Problem 2

The first two screenshots represent the READ request with files that are larger than 512 Bytes.

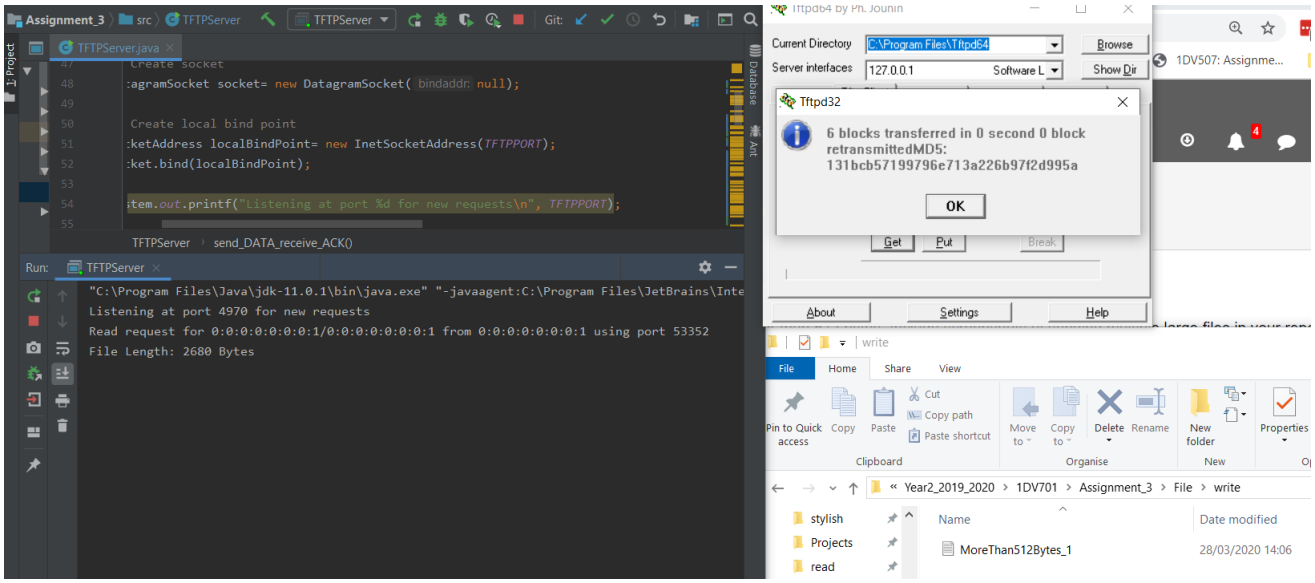

Figure 2: Read request to the TFTP server with file larger than 512 Bytes

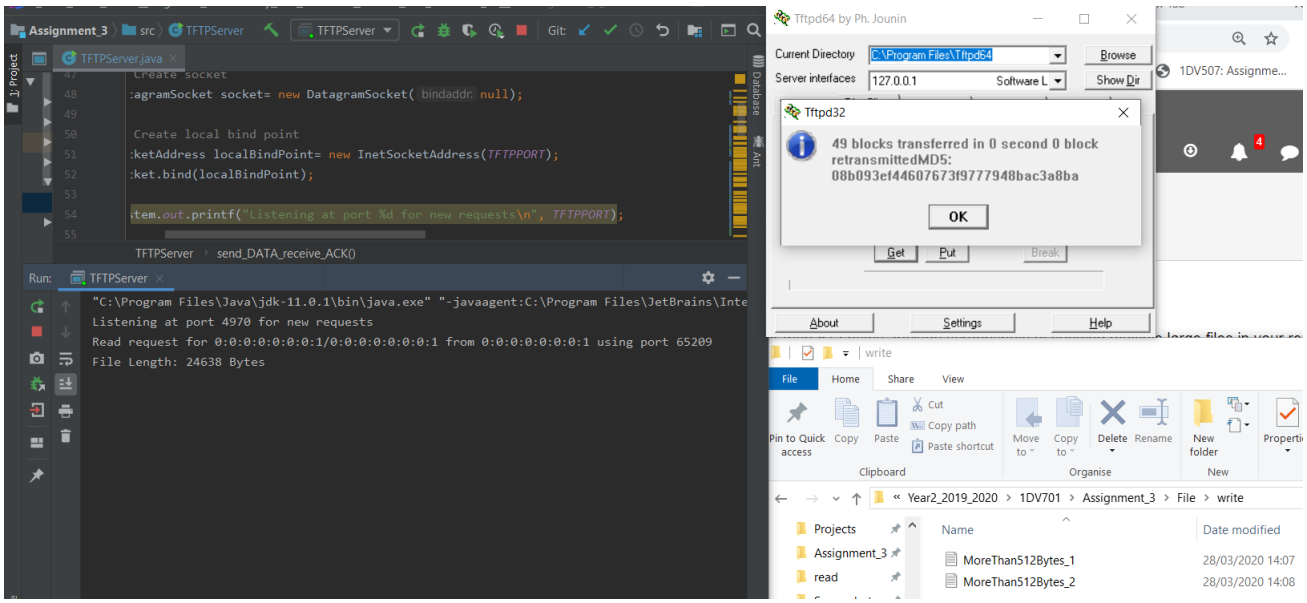

Figure 3: Read request to the TFTP server with file larger than 512 Bytes

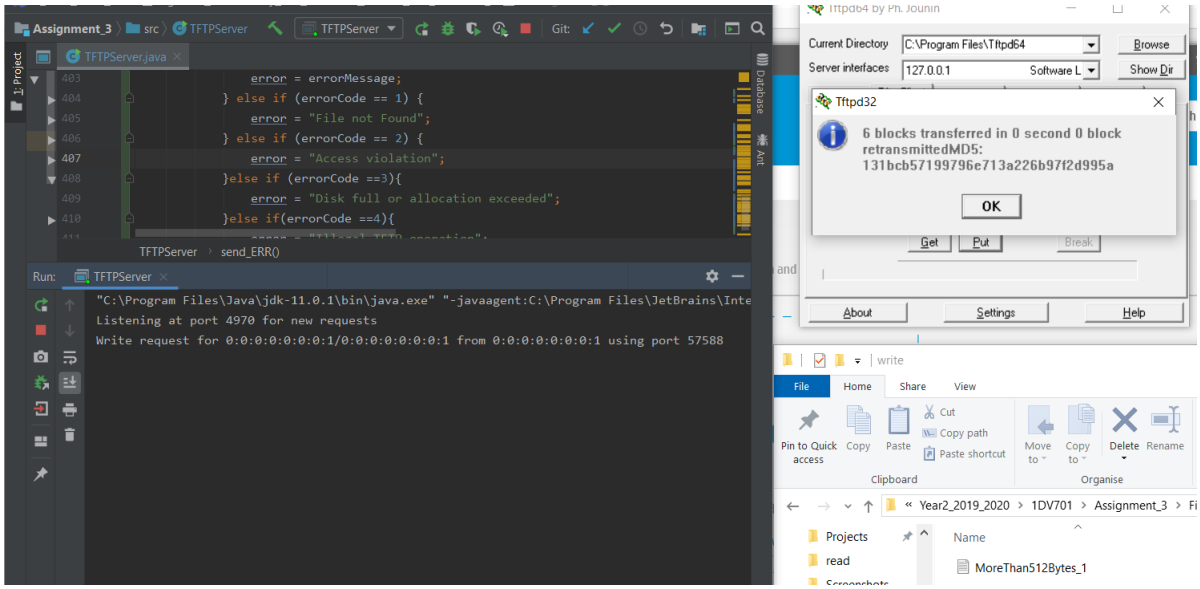

Figure 4: Write request to the TFTP server with file larger than 512 Bytes

3.1 Discussion

Describe how you tested timeouts and retransmissions!

To be able to handle the timeout, we have used the method *setSoTimeout()* (see below) on the DatagramSocket. It sets a timeout after 5 seconds and retransmits the packet.

```
//To check the response (ACK) after sending
byte[] responseBytes = new byte[BUFSIZE];
DatagramPacket response = new DatagramPacket(responseBytes, responseBytes.length);
sendSocket.setSoTimeout(5000);//Set timeout after 5s
sendSocket.receive(response);
//System.out.println("Got the response");
```

Figure 5: Snapsnot of code to show handling of timeouts

The two tests below are used to test timeouts and retransmissions. The methods are first failing the first ACK they are supposed to send, which would trigger the system and start the retransmissions. While the system is doing the retransmission it also activates the timeout. After 5 seconds without response, the data will be re-sent. The screenshot below

```
# Get a large file and fail the first ACK every time
def test_GMBFail1stAck(client):
    assert client.getMultiBlockFileFailAck(b'f3blks.bin', 1)


# Get a large file and fail the first two ACKs every time
def test_GMBFail2ndAck(client):
    assert client.getMultiBlockFileFailAck(b'f3blks.bin', 2)
```

Figure 6: Test created by the Teacher

shows that our implementation of the TFTP server is passing all of the tests created by the teacher. Therefore they pass the two tests shown above that test the retransmission of data.

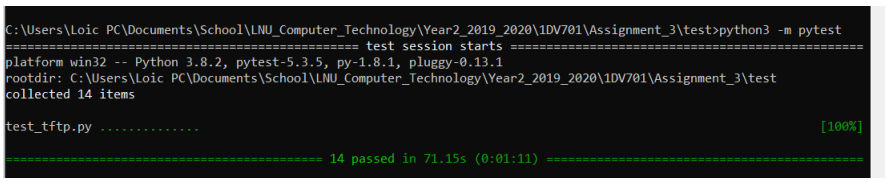

Figure 7: Results of the test given by the professor

4 Problem 3

- **Error code 0 - Not defined, see error message (if any).**

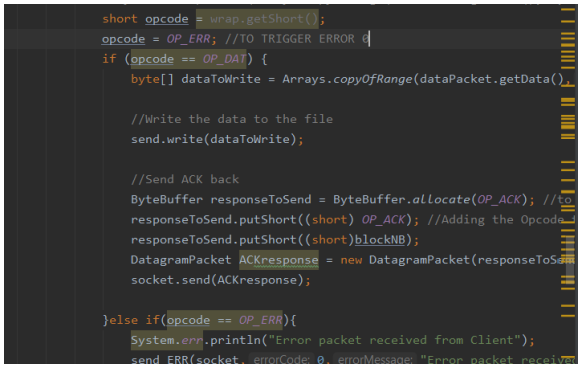

Figure 8: Code to show how the Error 0 was triggered

We added the extra line of code "opcode = OP-DAT" (where OP-DATA = 5) to be able to trigger the error code 0. The program will think it received an error packet and will , therefore, send back an error packet with error code 0.

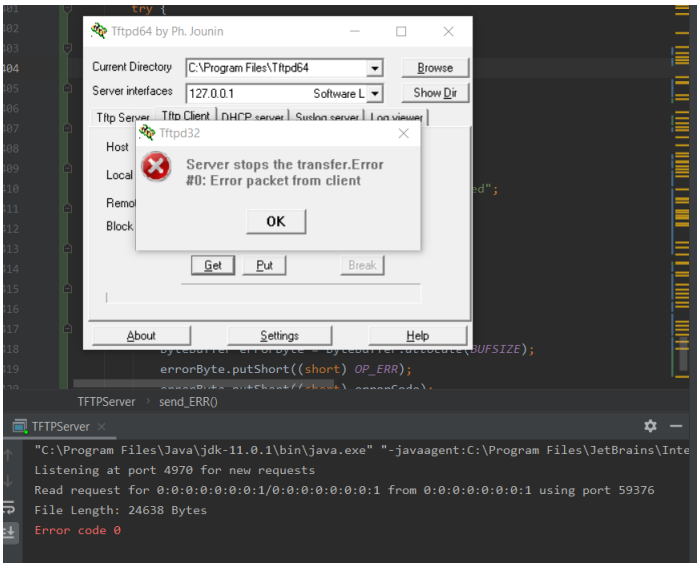

Figure 9: Received error packet from client - Error Code 0

- **Error code 1 - File not found.**

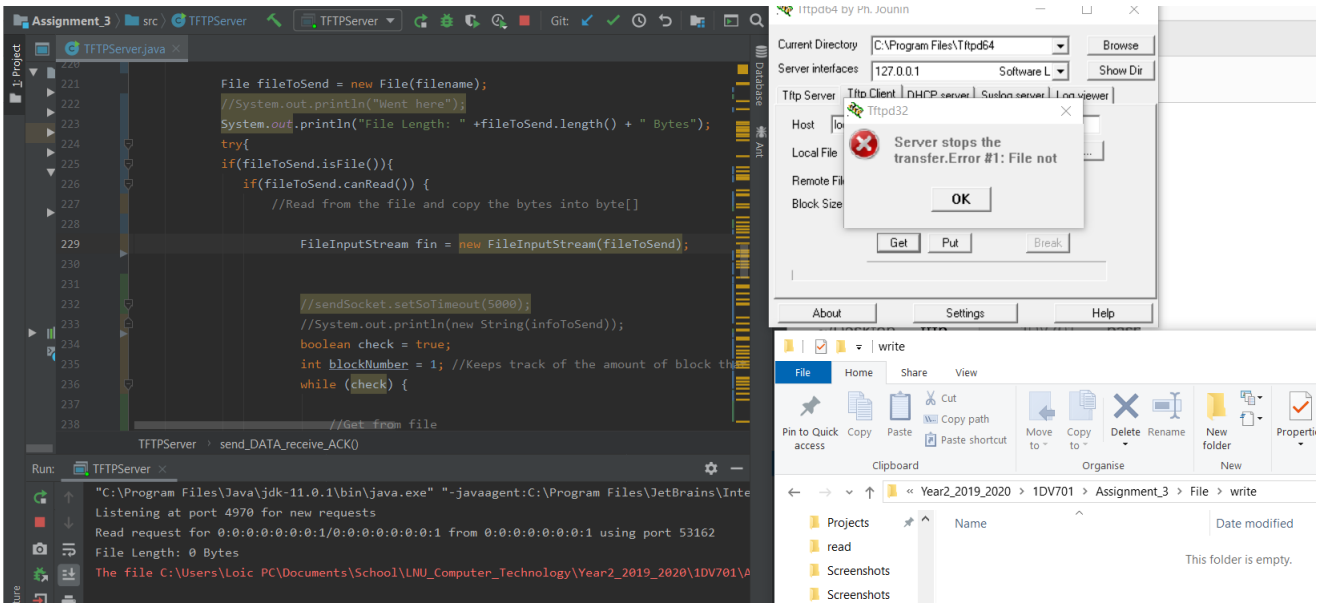

Figure 10: GET request of a file that does not exists - Error Code 1

We tried to GET fakefile.txt which is a file that does not exist and therefore it sent back an Error packet with error code 1 to the client.

- **Error code 2 - Access violation.**

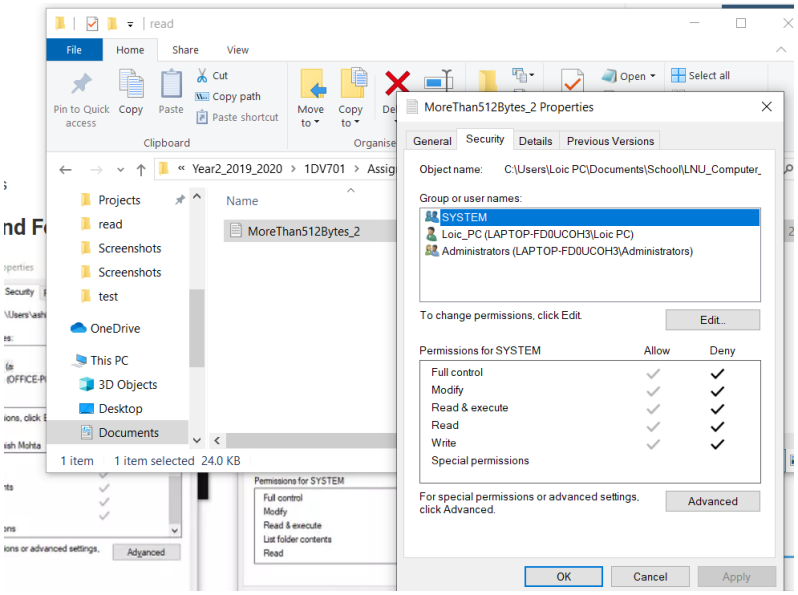

Figure 11: Removed all permission for the user over this File - Error Code 2

To be able to get this error, We had to manually remove all the permission on that text file for my user as you can see above. Whenever we use the GET command for a file, we are

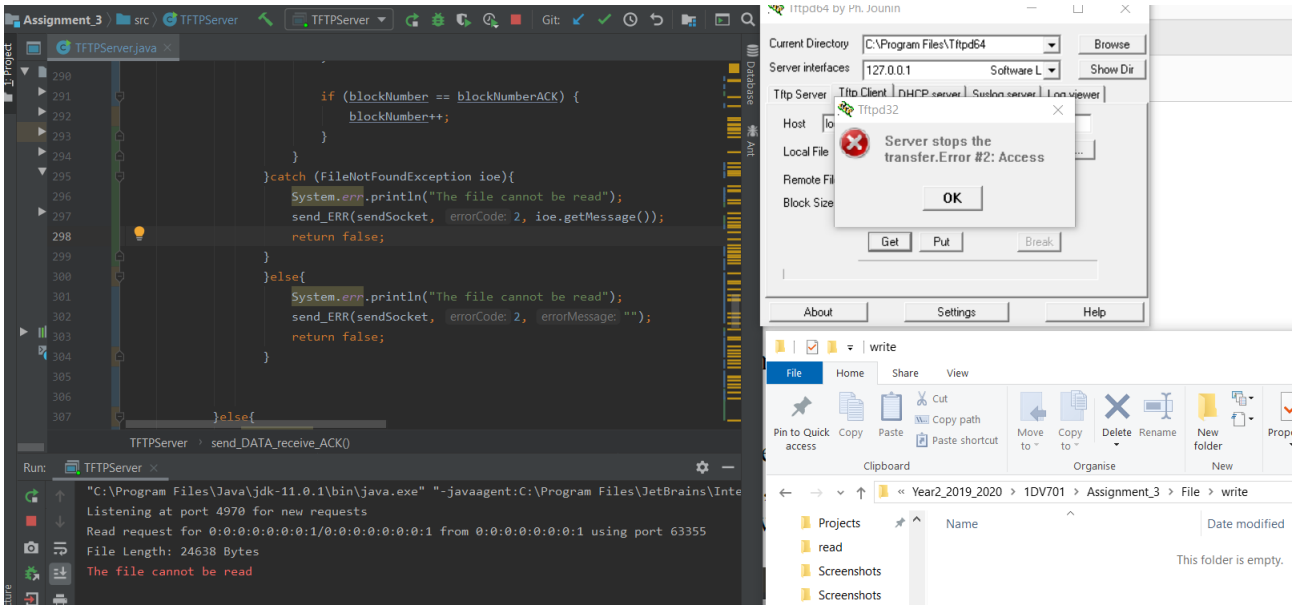

Figure 12: Error Code 2 - Access violation

checking if the file can be read (with .canRead() on File object), in this case, because we removed all the permission on the text document, it cannot read the file and therefore we send back the error packet with error code 2

- **Error code 6 - File already exists.**

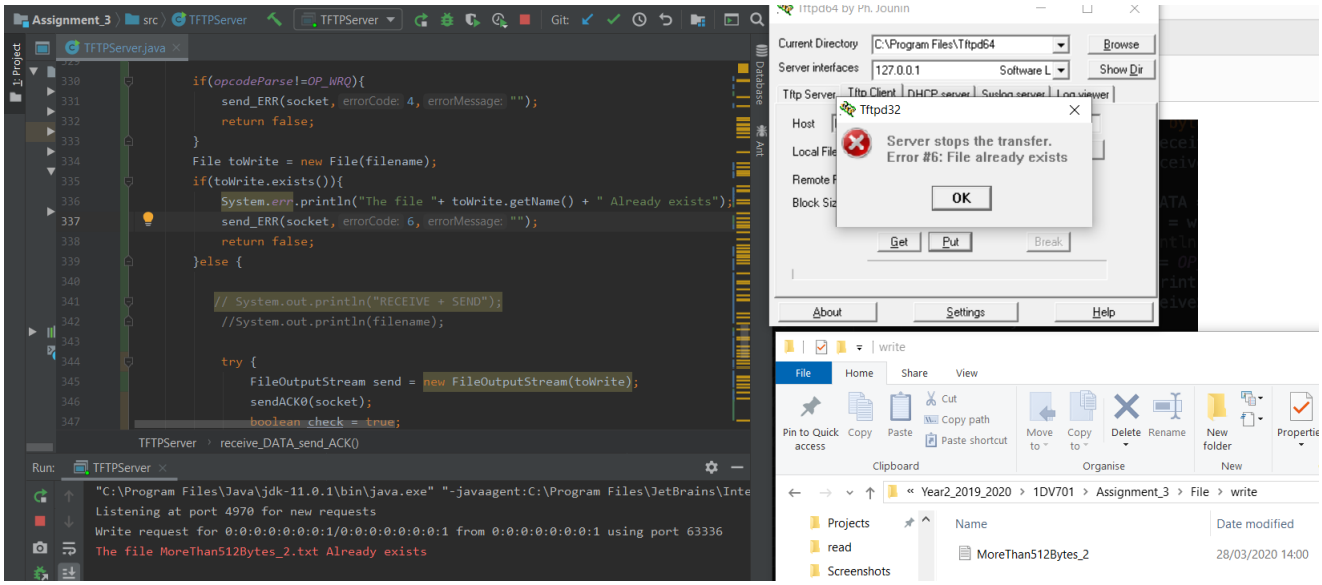

Figure 13: File already exists - Error Code 6

When getting a PUT request, we check first if the file already exists. If it does already, then we just send back an error packet with error code 6