Linnéuniversitetet
Kalmar Växjö

Report

Assignment 3

1DV701

Linnéuniversitetet
Kalmar Växjö

Author: Fabian Dacic, Yuyao Duan
Semester: Spring 2022
Email fd222fr@student.lnu.se, yd222br@student.lnu.se

Contents

1 Introduction

In this assignment, the server is designed via running the main method to initialize the service functions.

TFTPServer: this is the main class for runing the all related services. The user can configure which specified port to use for incoming requests as well as creating server thread object for new request.
ErrorCode: this is the inner enum class which is used for holding all error codes for TFTP protocols.

2 Problem 1

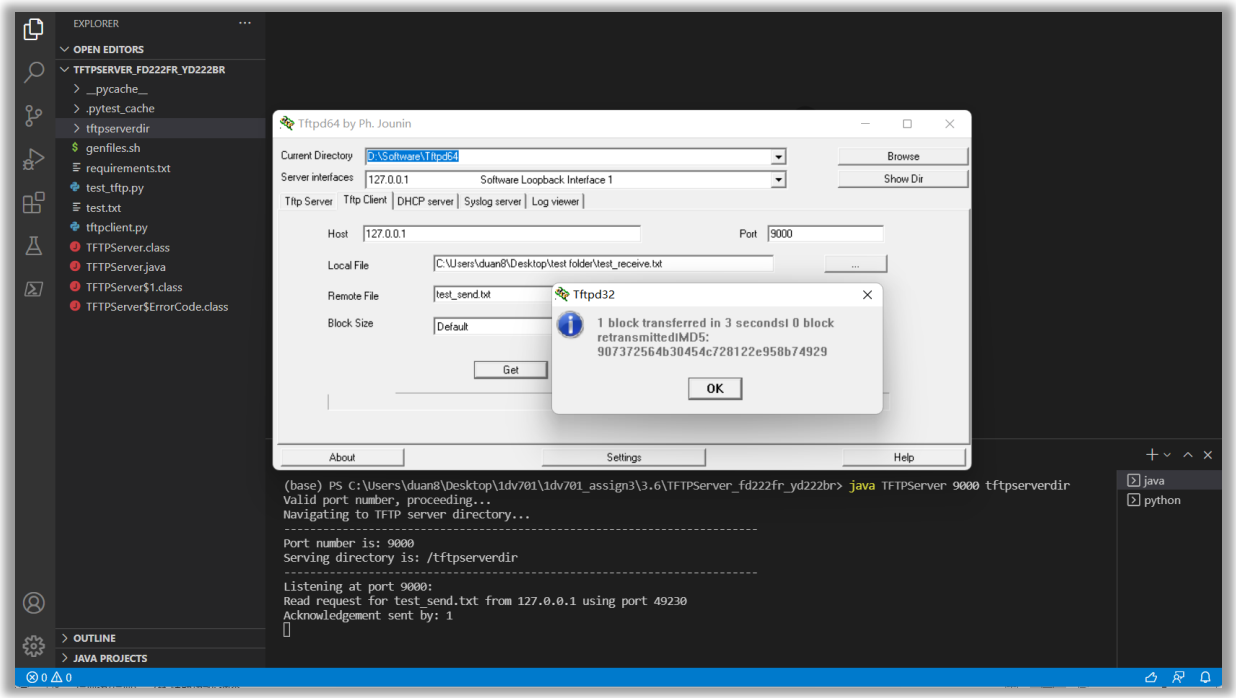

Figure 1. Read request from server shorter than 512 bytes

2.1 Discussion

According to above figure, we can see the TFTP get request interaction between a client and a server. The transmission is finished within 3 seconds. The file test_send.txt which is lower than 512 bytes is used as server data for sending and the file test_receive.txt is used for receiving the data that the client's want. The client can contact the server via the pre-configured port number, in this test is 9000. The server will create a new socket, i.e. the sendSocket which will be used for transferring the file.The server will assign a free port for this specific connection, in this case, it is port 49230. Therefore, the listening port can continue be used for other requests.

3 VG-Problem 2

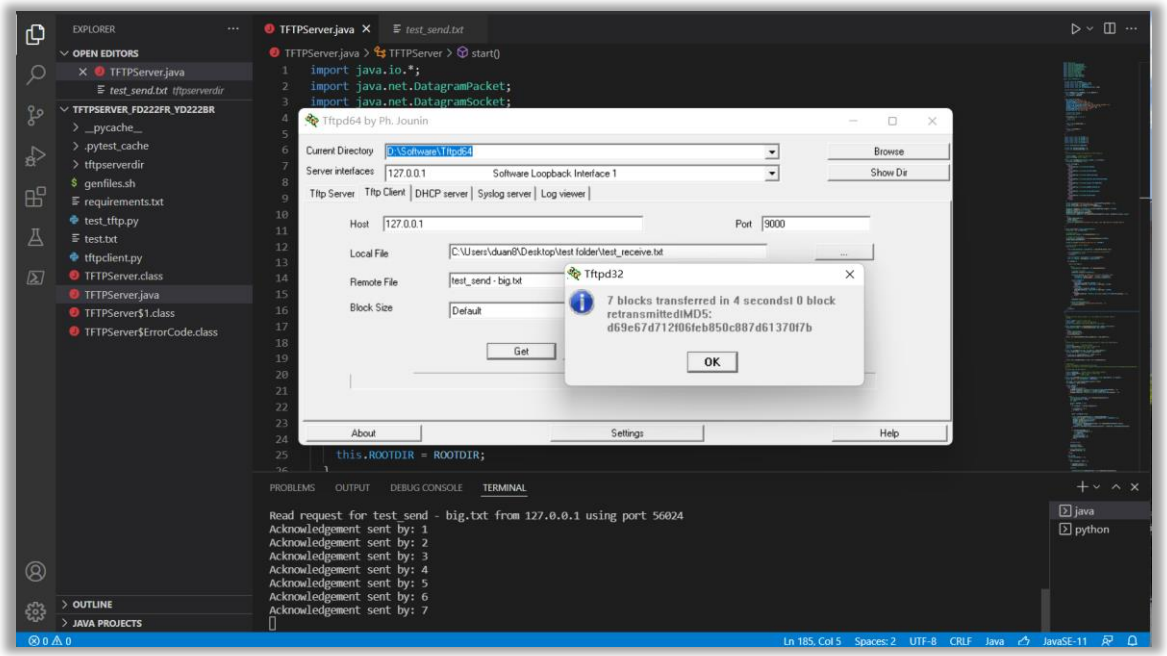

Figure 2. Read request from server larger than 512 bytes (test 1)

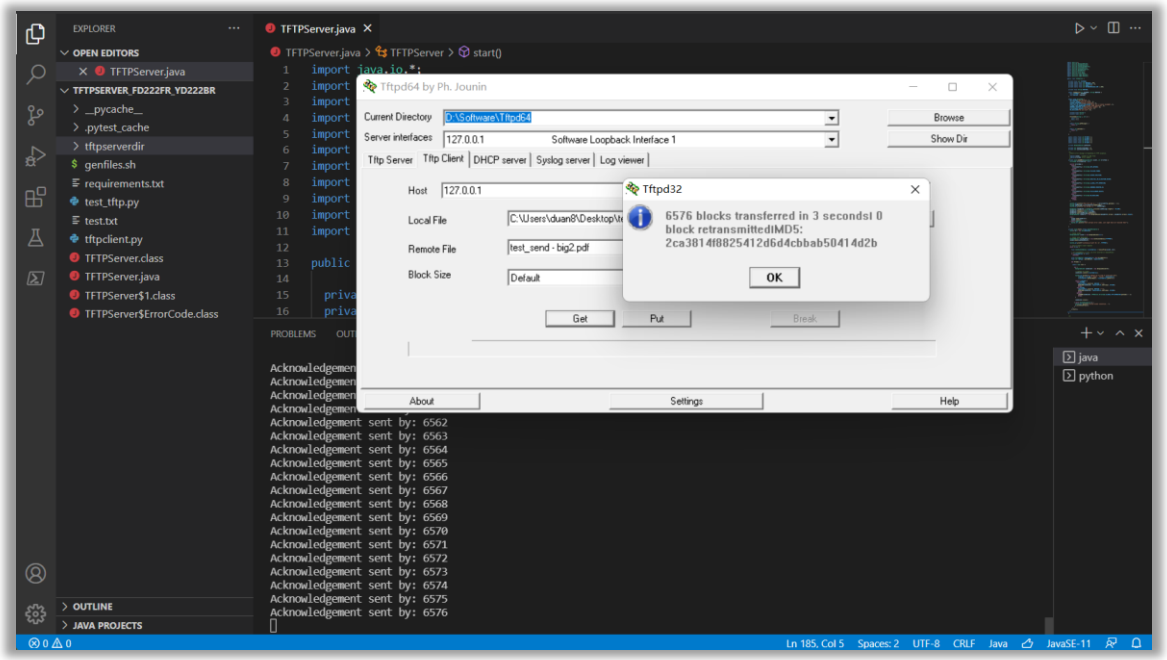

Figure 3. Read request from server larger than 512 bytes (test 2)

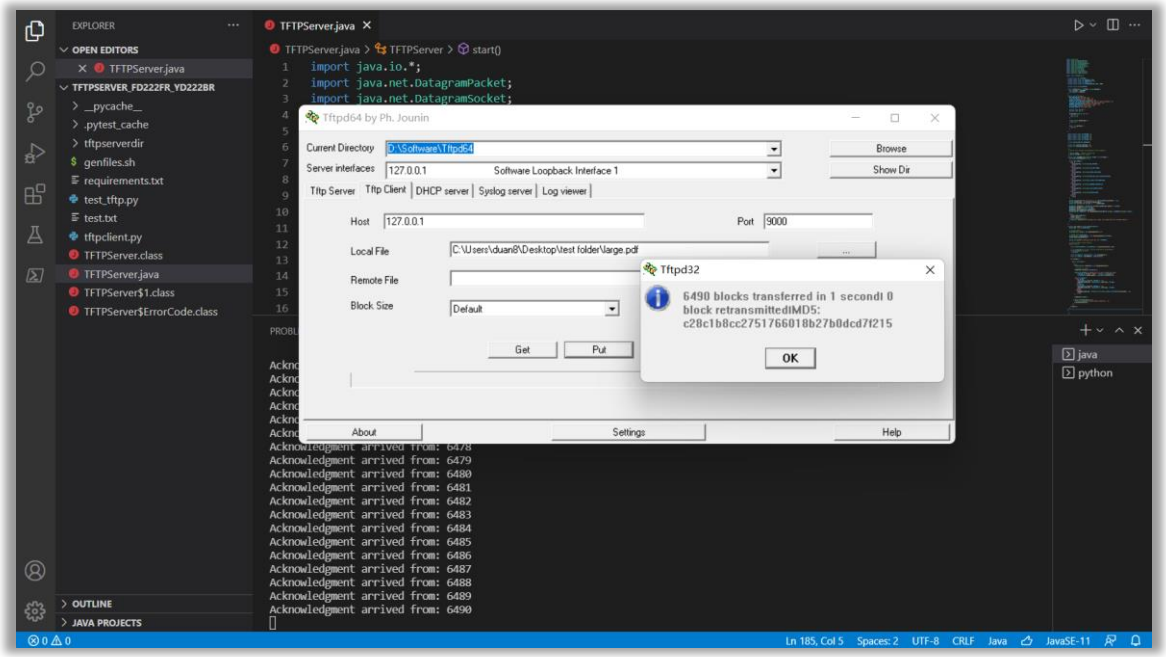

Figure 5. Write request to server larger than 512 bytes (test 1)

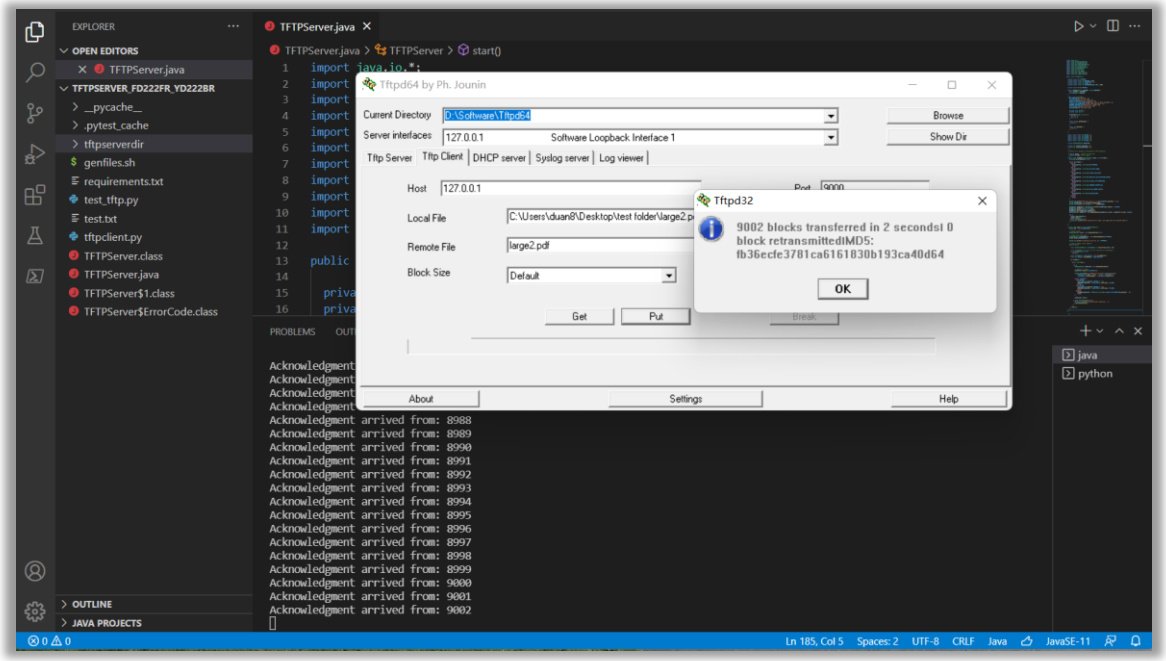

Figure 6. Write request to server larger than 512 bytes (test 2)

3.1 Discussion

Initially after having not implemented the timeout functionality correctly i.e the socket timeout exception was caught and in it the socket connection was closed, resulted in the failure of the last two tests provided for this assignment which lead to us believe that perhaps socket timeouts need to ignored and the previous packet to be re-transmitted which is evident in the respective methods that handle get and put requests. We tested the timeouts in different ways, one of them including an unintentional method where if the server has a lower timeout than the client (based on the tests provided) can result in a timeout of the server before the previous packet is re-transmitted by the client however

to keep in mind after numerous changes to the server and the different machines used, results may vary.

4 VG-Problem 3

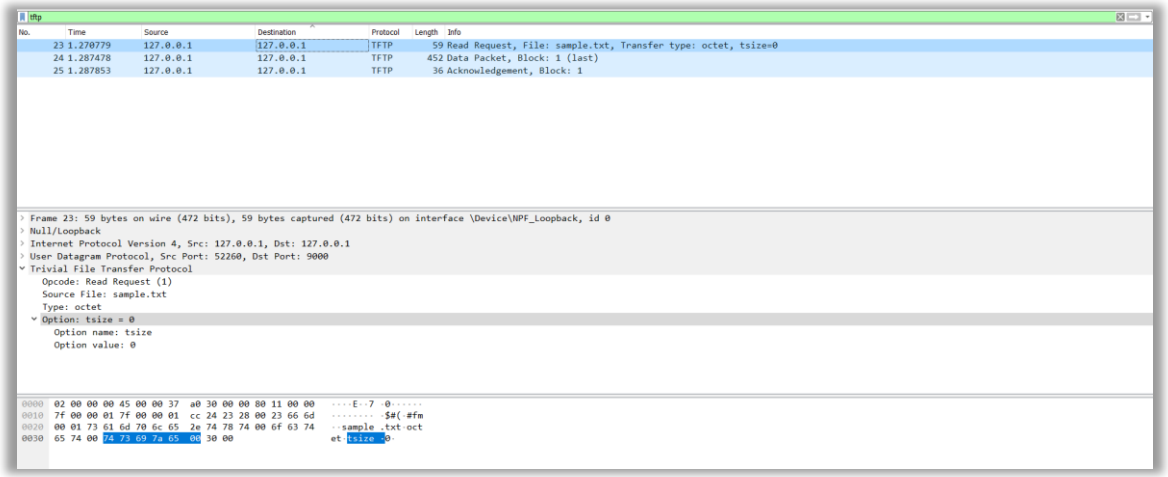

Figure 7. Read request first contact

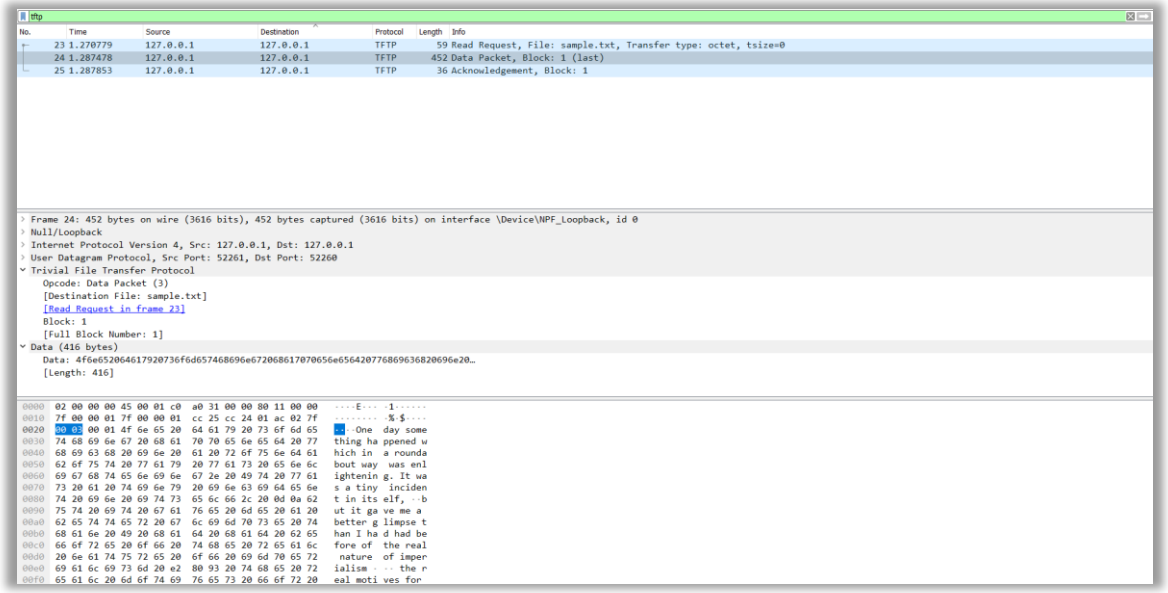

Figure 8. Read request second contact

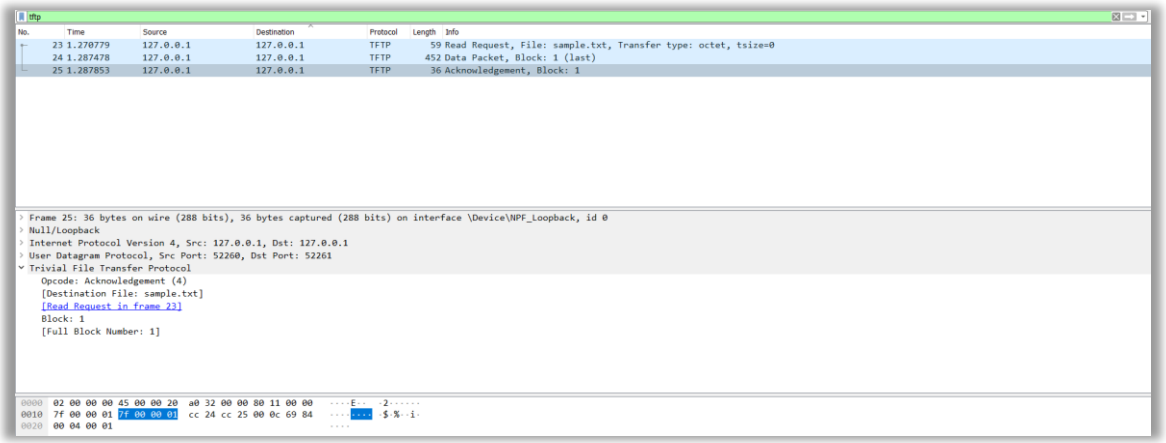

Figure 9. Read request third contact

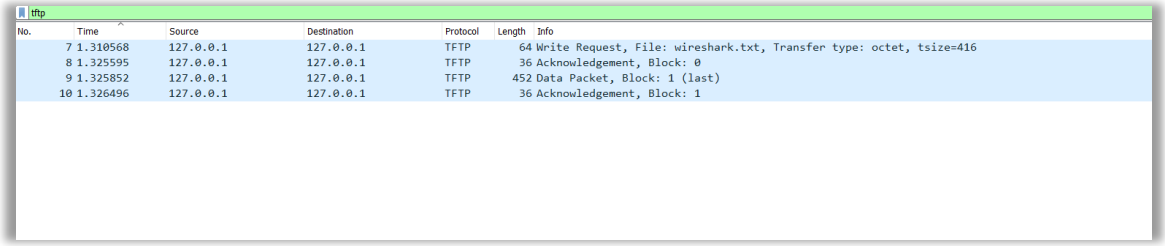

Figure 10. Write requests

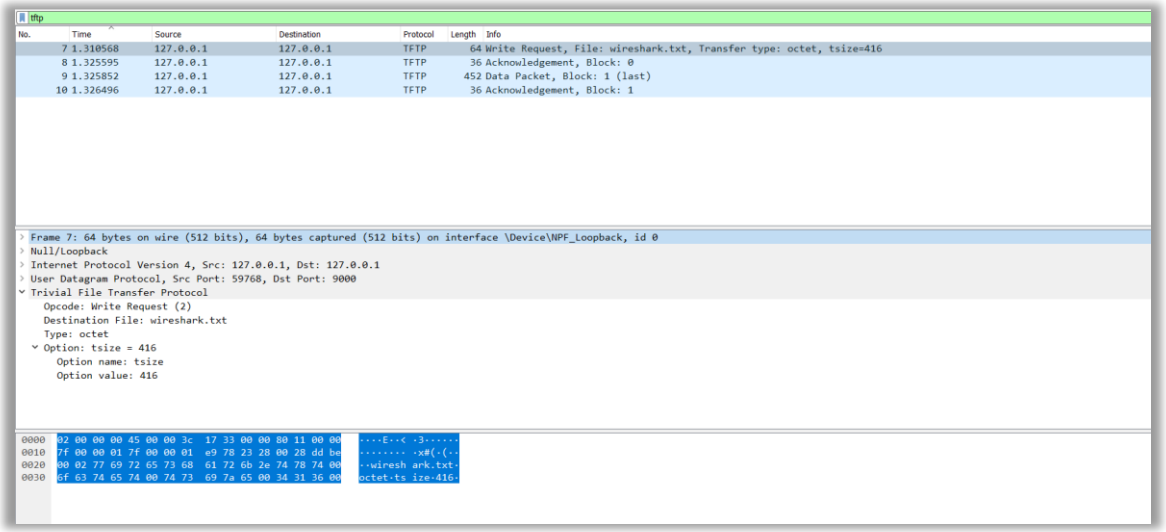

Figure 11. Write request first contact

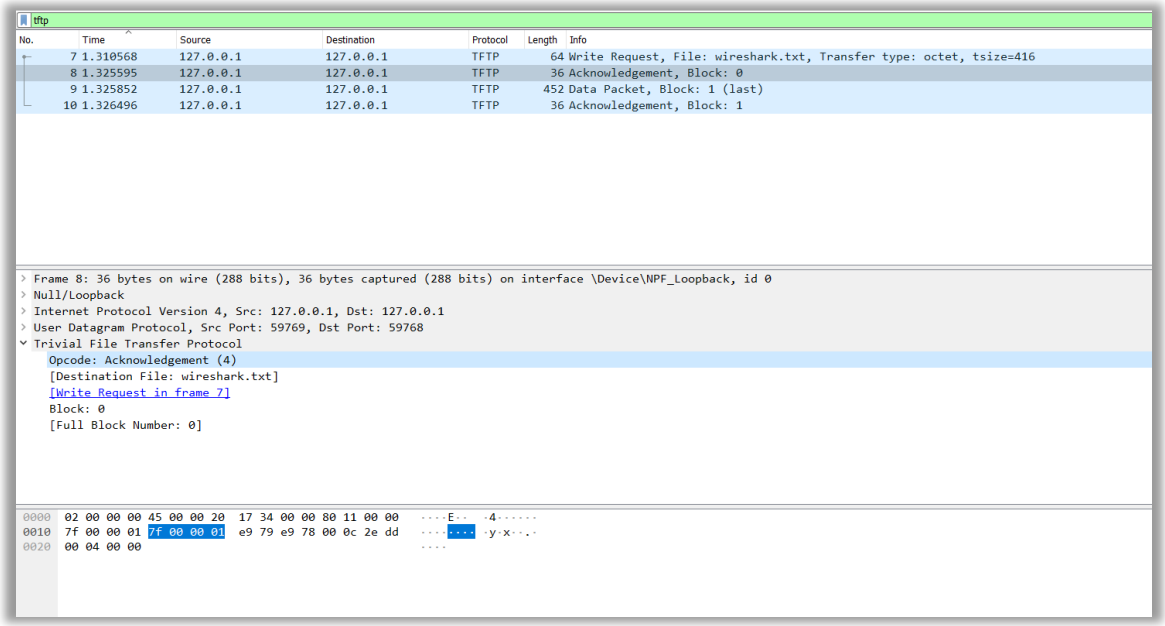

Figure 12. Write request second contact

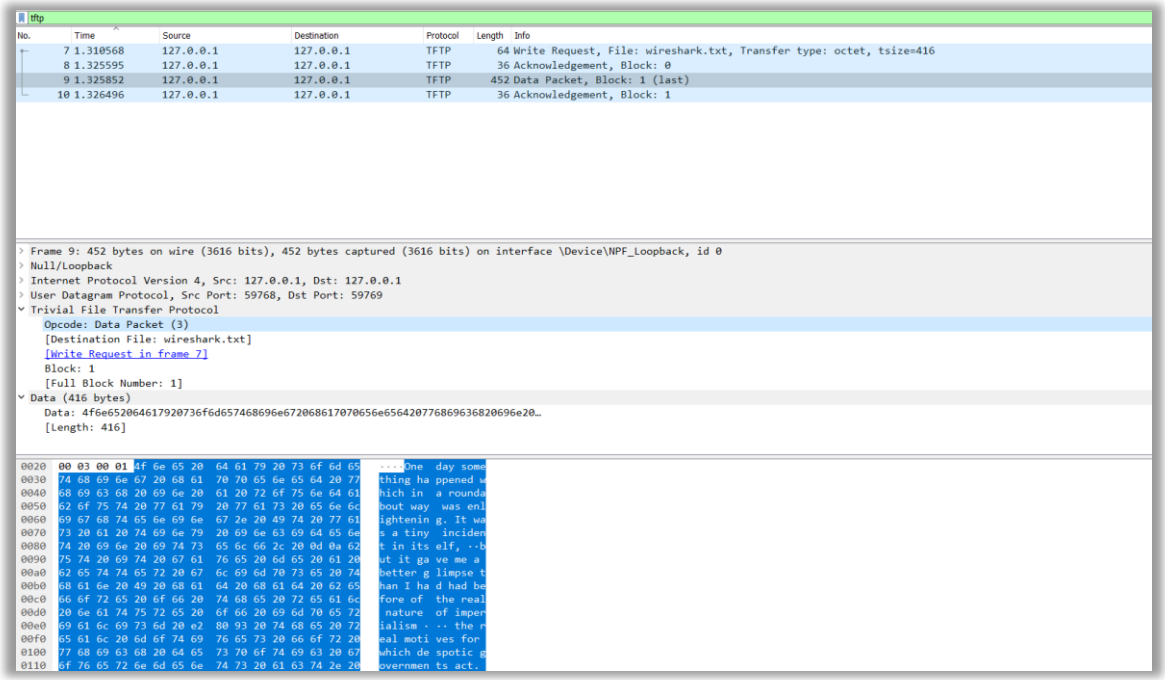

Figure 12. Write request third contact

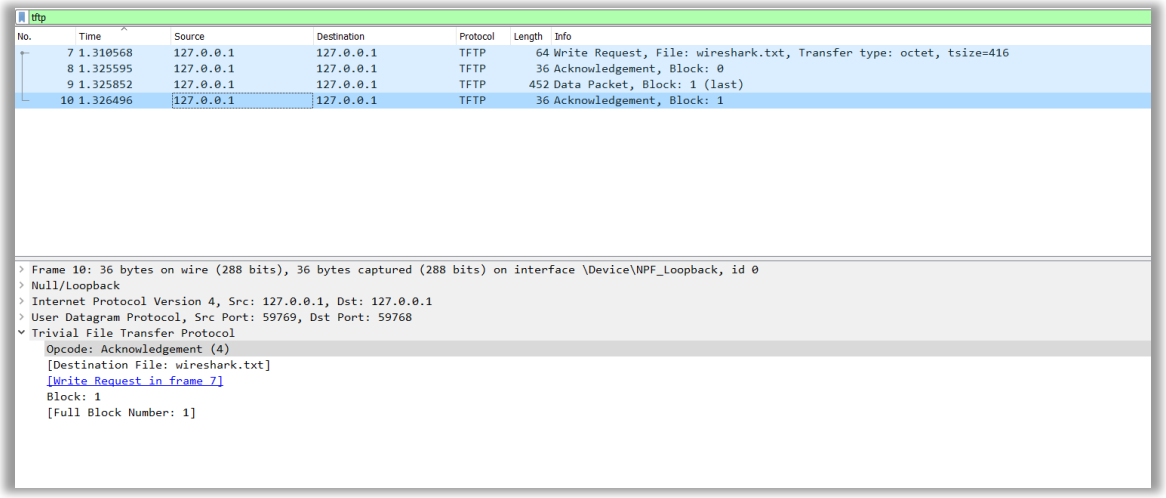

Figure 12. Write request fourth contact

4.1 Discussion

Wireshark has captured three requests related to a read request. The first is related to the server picking up a read request while listening to UDP port 9000 which tells the opcode being a read one, the source file inquired and the mode field which indicates that it is octet mode. As shown in the packet capture above, the first TFTP request is made to port 9000 (the TFTP server), but after that, the TFTP server selects another high-number port (or TID) to send its responses. Once received by the client, the client sends back an acknowledgment to the server on the TID where the data was received. Each data block received by the client is acknolwedged back to the server before the next byte is transmitted. The process continues until a partial data block or an empty data block is received, which indicates the end of the transfer.

A write request was sent from the client to the server on UDP Port 9000, the server listens and accepts the TFTP request and sends back an acknowledgment to the client on a randomly assigned ephemeral port (TID, visible on the screenshots). Received by the client, the first of the data is sent to the server on the port which the acknolwedgement was received. Each data block received by the server is acknolwedged with an ACK message back to the host before the next byte is transmitted. The process continued until a partial data block or an empty data block was received, indicating the end of the transfer.

From above discussion we can find that: for write, each data block received by the server is acknolwedged back to the client before the next byte is transmitted; where as for read, they are acknowledged back to the server.

5 VG-Problem 4

In this section the screenshots of the exceptions of error code 0, 1, 2, 6 are presented.

Due to the ambiguous nature of the not defined error, we tried to handle it in a way where if the server encounters a situation that it does not how to handle or what response to send, not defined error will appear. The way it was triggered was through a poorly made get request however instead of the data being transferred in octet mode, it was edited to be an unknown mode. This was done with the help of python tests, and produced the following results:

```
Listening at port 9000:
Read request for thisistest from 127.0.0.1 using port 52814
The error num is: 1
The message is: File not found
The error num is: 0
The message is: Not defined
Sent multiple error codes, user might have not received them!
```

Figure 13. Error code 0 - Not defined

This error occurred when the requested file does not exist on the server:

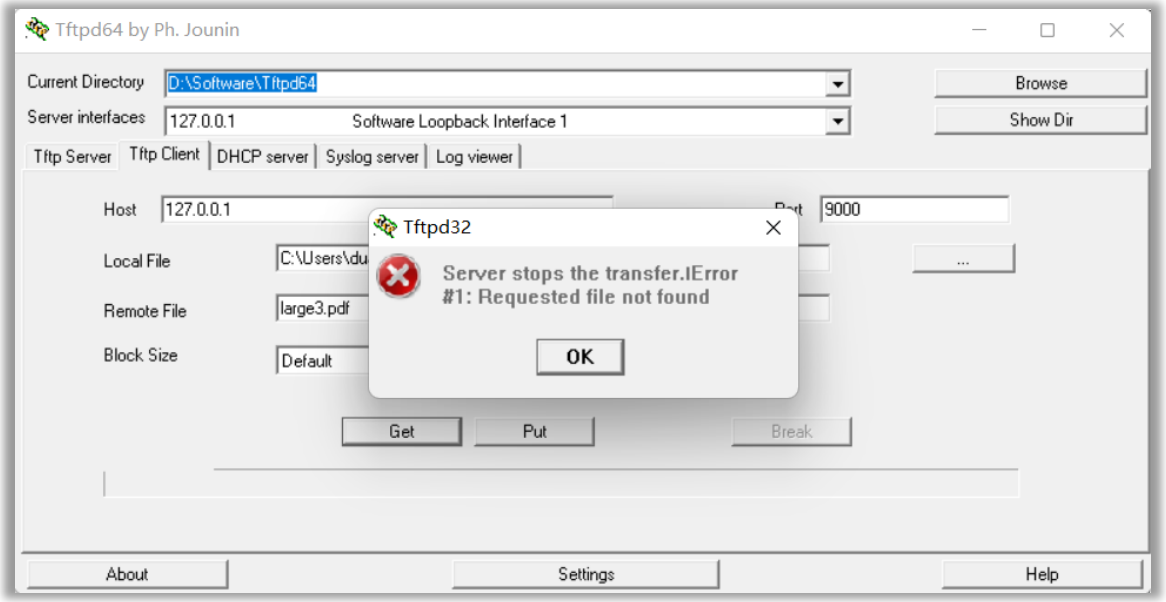

Figure 14. Error code 1 - Requested file not found

In order to test the following error, we set up one file with the property “Read only” and then when we try to get the file, the access violation will occur:

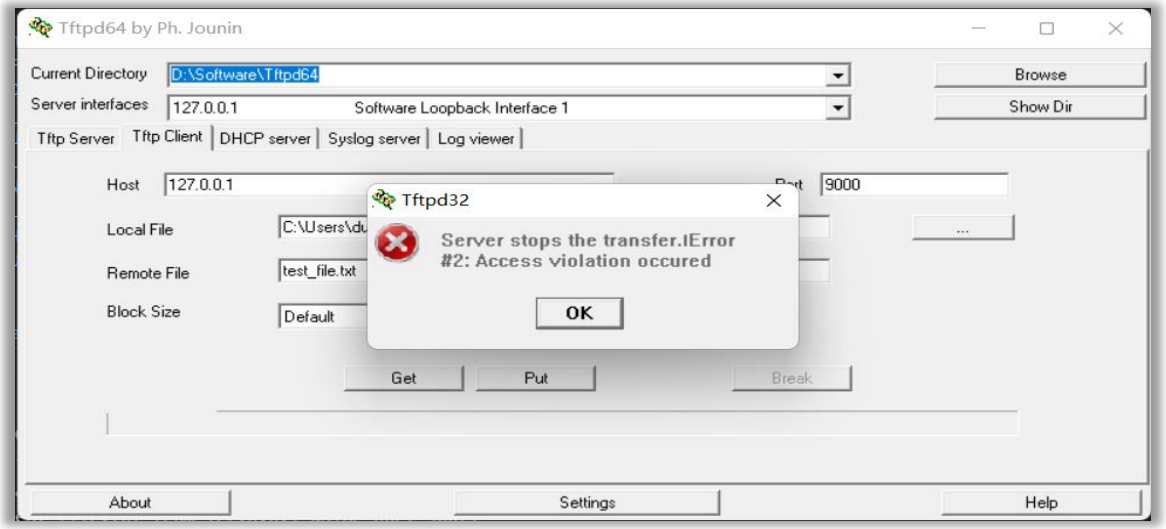

Figure 15. Error code 2 - Access violation occurred

In order to test the following error, we upload a file which is already existed in the server folder:

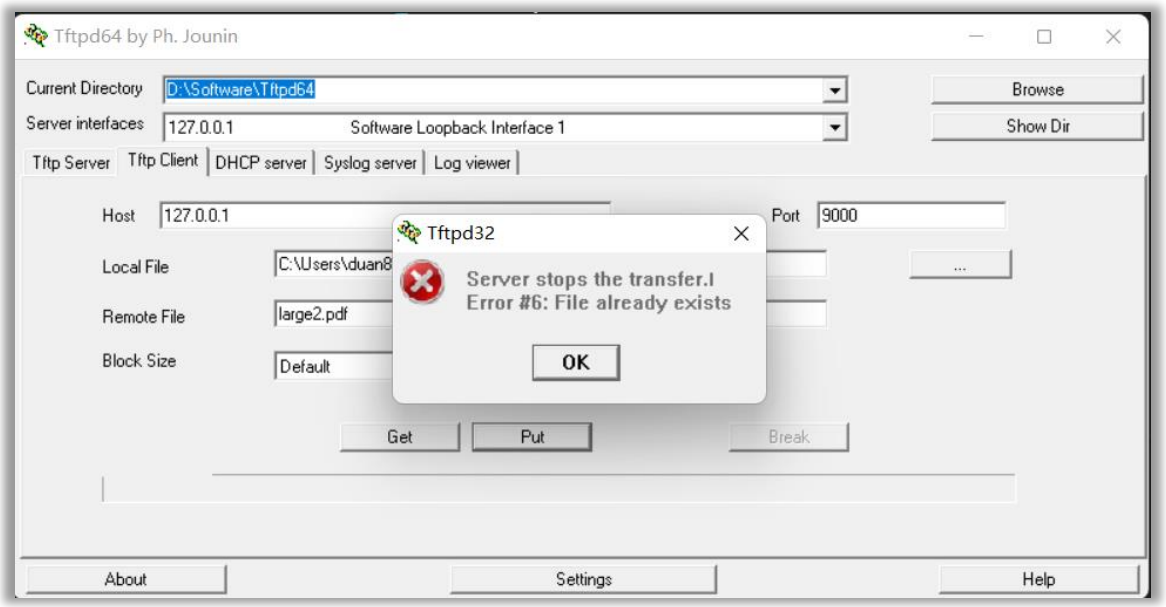

Figure 16. Error code 6 - File already exists

6 Summary

During this assignment, the cooperation was conducted via intensive online meeting and screensharing due to the fact of implementing TFPT server is new for both of us. The work was split between 50% for Yuyao Duan and 50% for Fabian Dacic. Both members were involved in all of the tasks due to the demand of logic logical coherence of this task.