



Linnéuniversitetet

Kalmar Västervik

Report

Assignment 3

IDV701



Linnéuniversitetet

Kalmar Växjö



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

Contents

1 Introduction	1
2 Problem 1	1
2.1 Discussion	1
3 VG-Problem 2	2
3.1 Discussion	3
4 VG-Problem 3	4
4.1 Discussion	7
5 VG-Problem 4	7
6 Summary	9

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 running 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

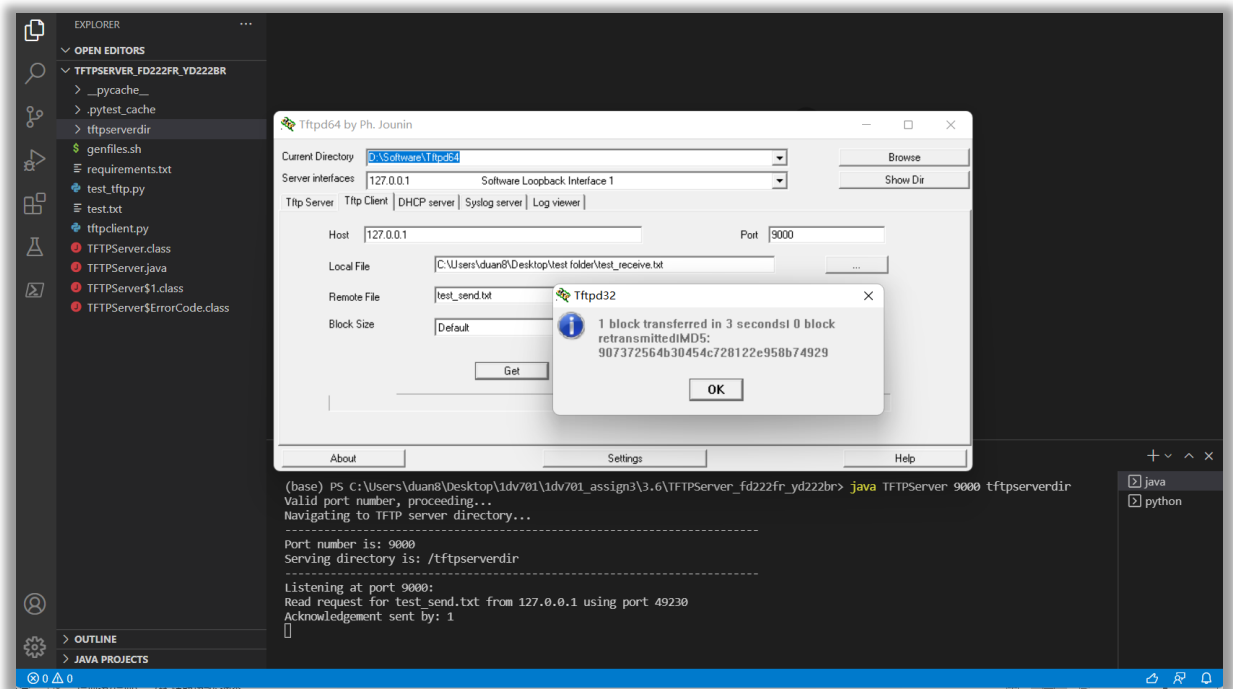


Figure 1. Read request from server shorter than 512 bytes

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

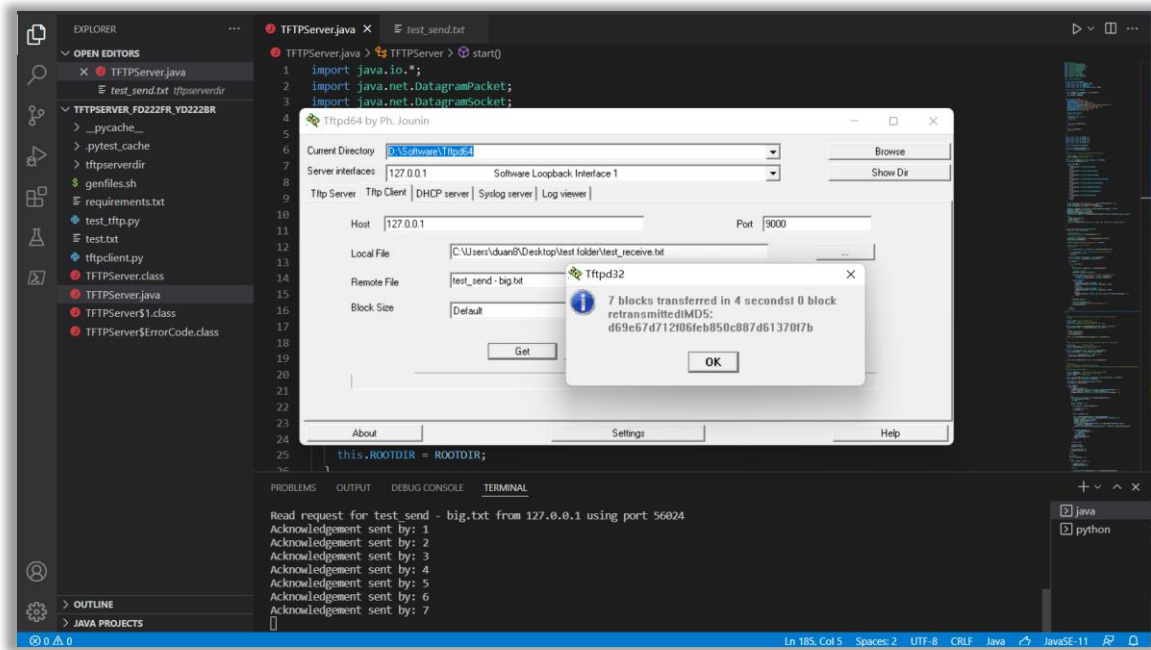


Figure 2. Read request from server larger than 512 bytes (test 1)

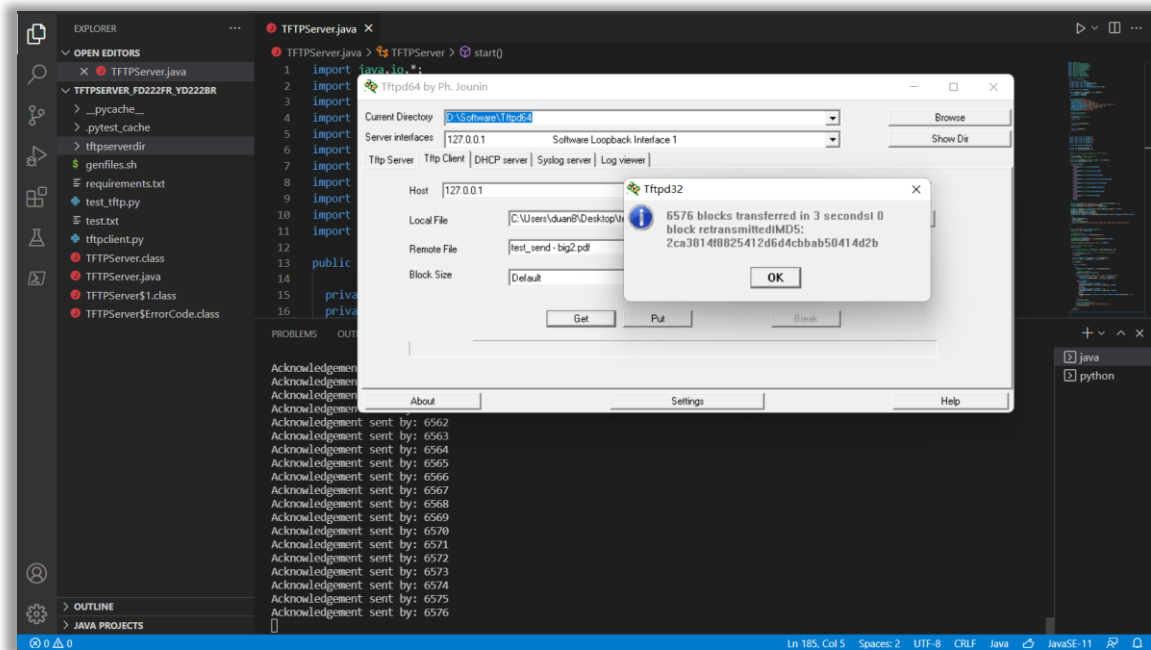


Figure 3. Read request from server larger than 512 bytes (test 2)

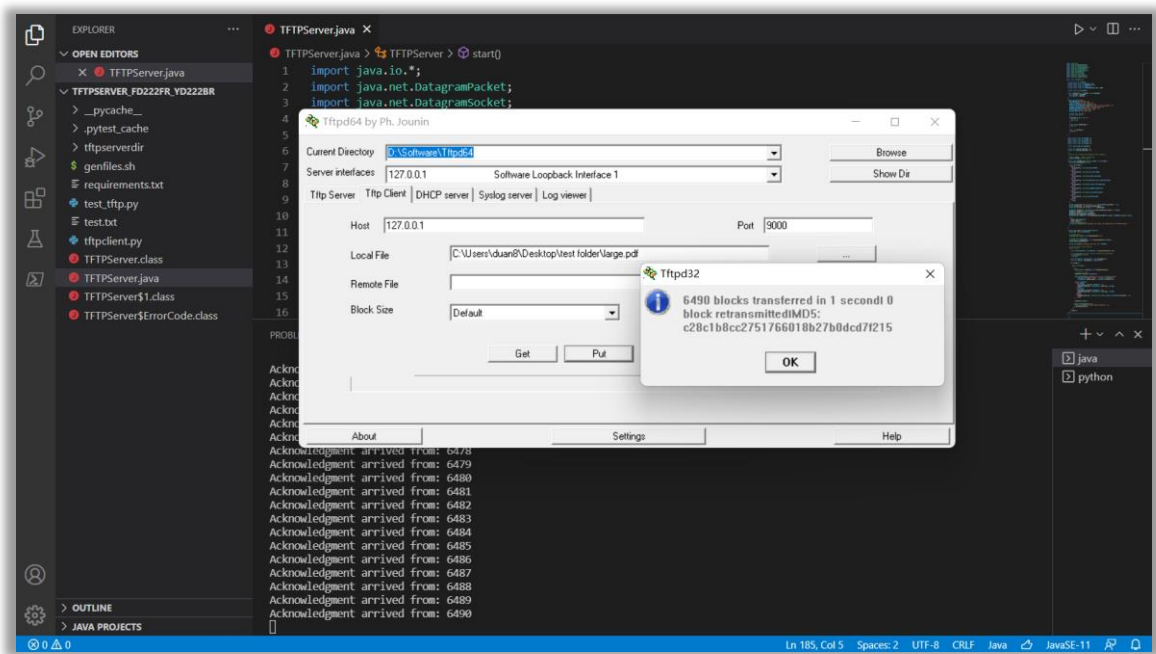


Figure 5. Write request to server larger than 512 bytes (test 1)

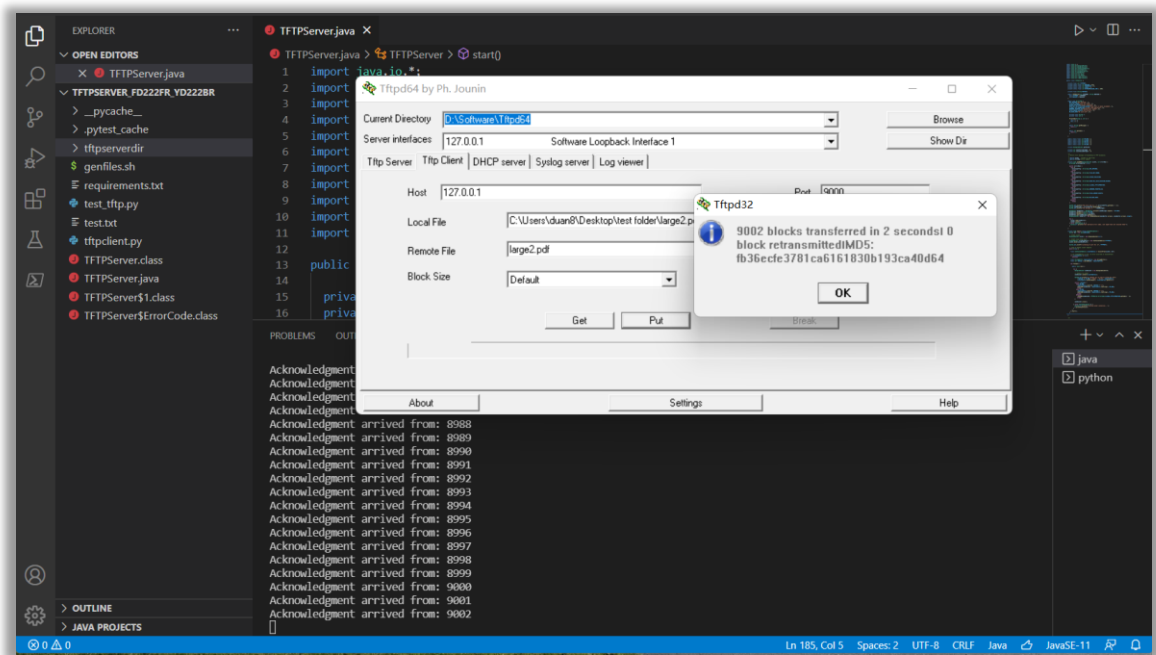


Figure 6. Write request to server larger than 512 bytes (test 2)

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 be 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

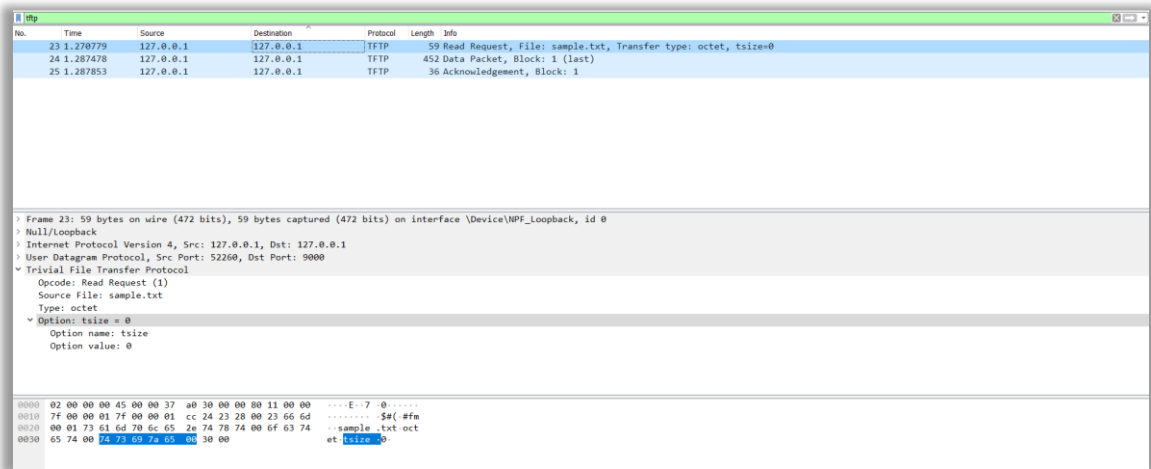


Figure 7. Read request first contact

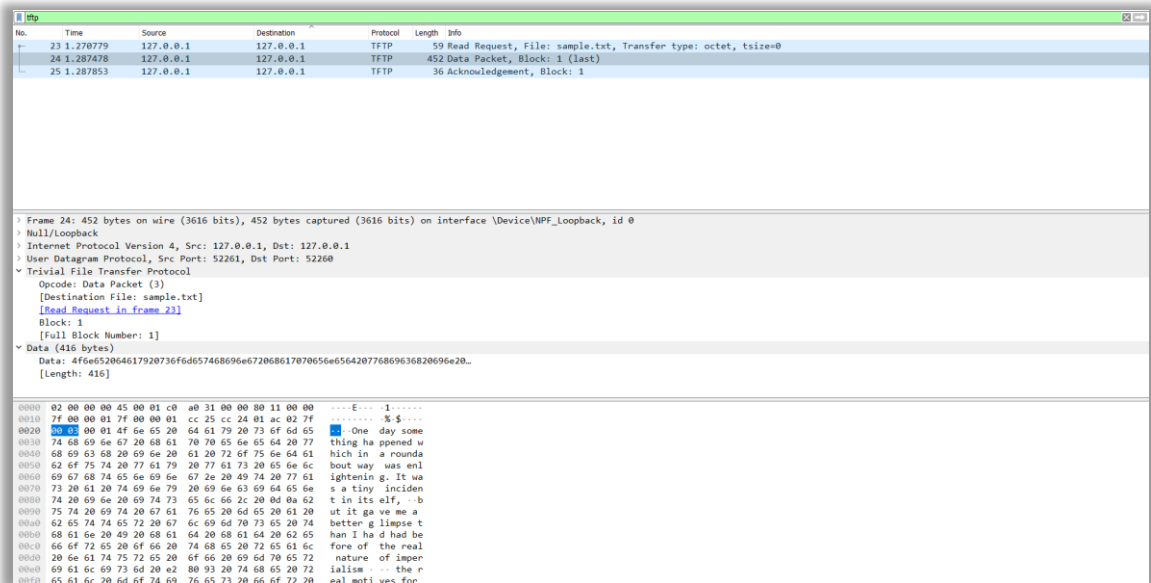


Figure 8. Read request second contact

No.	Time	Source	Destination	Protocol	Length	Info
23	1.270779	127.0.0.1	127.0.0.1	TFTP	59	Read Request, File: sample.txt, Transfer type: octet, tsize=0
24	1.287478	127.0.0.1	127.0.0.1	TFTP	452	Data Packet, Block: 1 (last)
25	1.287853	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 1

> Frame 25: 36 bytes on wire (288 bits), 36 bytes captured (288 bits) on interface \Device\NPF_{loopback}, id 0
 > Null/Loopback
 > Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 > User Datagram Protocol, Src Port: 52260, Dst Port: 52261
 > Trivial File Transfer Protocol

Opcode: Acknowledgement (4)
 [Destination File: sample.txt]
 [Read Request in frame 23]
 Block: 1
 [Full Block Number: 1]


```

0000  02 00 00 00 45 00 00 20 a0 32 00 00 00 11 00 00  ....E...2.....
0010  7f 00 00 01 7f 00 00 01 cc 24 cc 25 00 0c 69 84  ....f...f...$.X..I.
0020  00 04 00 01                                     ....
  
```

Figure 9. Read request third contact

No.	Time	Source	Destination	Protocol	Length	Info
7	1.310568	127.0.0.1	127.0.0.1	TFTP	64	Write Request, File: wireshark.txt, Transfer type: octet, tsize=416
8	1.325595	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 0
9	1.325852	127.0.0.1	127.0.0.1	TFTP	452	Data Packet, Block: 1 (last)
10	1.326496	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 1

Figure 10. Write requests

No.	Time	Source	Destination	Protocol	Length	Info
7	1.310568	127.0.0.1	127.0.0.1	TFTP	64	Write Request, File: wireshark.txt, Transfer type: octet, tsize=416
8	1.325595	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 0
9	1.325852	127.0.0.1	127.0.0.1	TFTP	452	Data Packet, Block: 1 (last)
10	1.326496	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 1

> Frame 7: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface \Device\NPF_{loopback}, id 0
 > Null/Loopback
 > Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 > User Datagram Protocol, Src Port: 59768, Dst Port: 9000
 > Trivial File Transfer Protocol

Opcode: Write Request (2)
 Destination File: wireshark.txt
 Type: octet
 Option: tsize = 416
 Option name: tsize
 Option value: 416


```

0000  02 00 00 00 45 00 00 3c 17 33 00 00 00 11 00 00  ....E...3.....
0010  7f 00 00 01 7f 00 00 01 e9 78 23 28 00 28 dd be  ....f...f...x#(.(.
0020  00 02 77 69 72 65 73 68 61 72 66 2e 74 78 74 00  ..wireshark.txt..
0030  0f 63 74 65 74 00 74 73 69 7a 65 00 34 31 36 00  octet-tsize=416.
  
```

Figure 11. Write request first contact

tftp						
No.	Time	Source	Destination	Protocol	Length	Info
7	1.310568	127.0.0.1	127.0.0.1	TFTP	64	Write Request, File: wireshark.txt, Transfer type: octet, tsize=416
8	1.325595	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 0
9	1.325852	127.0.0.1	127.0.0.1	TFTP	452	Data Packet, Block: 1 (last)
10	1.326496	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 1

> Frame 8: 36 bytes on wire (288 bits), 36 bytes captured (288 bits) on interface \Device\NPF_{Loopback}, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> User Datagram Protocol, Src Port: 59769, Dst Port: 59768

> Trivial File Transfer Protocol

Opcode: Acknowledgement (4)

[Destination File: wireshark.txt]

[Write Request in frame 7]

Block: 0

[Full Block Number: 0]

0000	02 00 00 00	45 00 00 20	17 34 00 00	80 11 00 00E...-4-.....
0010	7f 00 00 01	7f 00 00 01	e9 79 e9 78	00 0c 2e dd-...-y-X-...
0020	00 04 00 00			

Figure 12. Write request second contact

tftp						
No.	Time	Source	Destination	Protocol	Length	Info
7	1.310568	127.0.0.1	127.0.0.1	TFTP	64	Write Request, File: wireshark.txt, Transfer type: octet, tsize=416
8	1.325595	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 0
9	1.325852	127.0.0.1	127.0.0.1	TFTP	452	Data Packet, Block: 1 (last)
10	1.326496	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 1

> Frame 9: 452 bytes on wire (3616 bits), 452 bytes captured (3616 bits) on interface \Device\NPF_{Loopback}, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> User Datagram Protocol, Src Port: 59768, Dst Port: 59769

> Trivial File Transfer Protocol

Opcode: Data Packet (3)

[Destination File: wireshark.txt]

[Write Request in frame 7]

Block: 1

[Full Block Number: 1]

> Data (416 bytes)

Data: 4f6e5206461792073f6d657468696e672068617070656e56420776869636820696e20...

[Length: 416]

0020	00 03 00 01	4f 6e 65 20	64 61 79 20	73 6f 6d 65One day some
0030	74 68 69 6e	67 20 68 61	70 70 65 6e	65 64 20 77	thing ha ppened w
0040	68 69 63 68	20 69 6e 20	61 20 72 6f	75 6e 64 61	nich in a rounda
0050	62 6f 75 74	20 77 61 79	20 77 61 73	20 65 6e 6d	bout way was enl
0060	69 67 68 74	65 6e 69 6e	67 2e 20 49	74 20 77 61	ightenin g. It wa
0070	73 20 61 20	74 69 6e 79	20 69 6e 63	69 64 65 6e	s a tiny inciden
0080	74 20 69 6e	20 69 74 73	65 6e 66 2c	20 0d 0a 62	t in its elf, --b
0090	75 74 20 69	74 20 67 61	76 65 20 6d	65 20 61 20	ut it ge ve me a
00a0	62 65 74 74	65 72 20 67	6c 69 6d 70	73 65 20 74	better g limpose
00b0	68 61 6e 20	49 20 68 61	64 20 68 61	64 20 62 65	han I ha d had be
00c0	66 6f 72 65	20 6f 66 20	74 68 65 20	72 65 61 6c	fore of the real
00d0	20 6e 61 74	75 72 65 20	6f 66 20 69	6d 70 65 72	nature of imper
00e0	69 61 6c 69	73 6d 20 e2	80 93 20 74	68 65 20 72	ialism -- the m
00f0	65 61 6c 20	6d 6f 74 69	76 65 73 20	66 6f 72 20	el motives for
0100	77 68 69 63	68 20 64 65	73 70 6f 74	69 63 20 67	which de spotic g
0110	6f 76 65 72	6e 6d 65 6e	74 73 20 61	63 74 2e 20	overnmen ts act,

Figure 12. Write request third contact

No.	Time	Source	Destination	Protocol	Length	Info
7	1.310568	127.0.0.1	127.0.0.1	TFTP	64	Write Request, File: wireshark.txt, Transfer type: octet, tsize=416
8	1.325595	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 0
9	1.325852	127.0.0.1	127.0.0.1	TFTP	452	Data Packet, Block: 1 (last)
10	1.326496	127.0.0.1	127.0.0.1	TFTP	36	Acknowledgement, Block: 1

Frame 10: 36 bytes on wire (288 bits), 36 bytes captured (288 bits) on interface \Device\NPF_{Loopback}, id 0
 Null/Loopback
 Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 User Datagram Protocol, Src Port: 59769, Dst Port: 59768
 Trivial File Transfer Protocol
 Opcode: Acknowledgement (4)
 [Destination File: wireshark.txt]
 [Write Request in frame 7]
 Block: 1
 [Full Block Number: 1]

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 acknowledged 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 acknowledgment was received. Each data block received by the server is acknowledged 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 acknowledged 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:

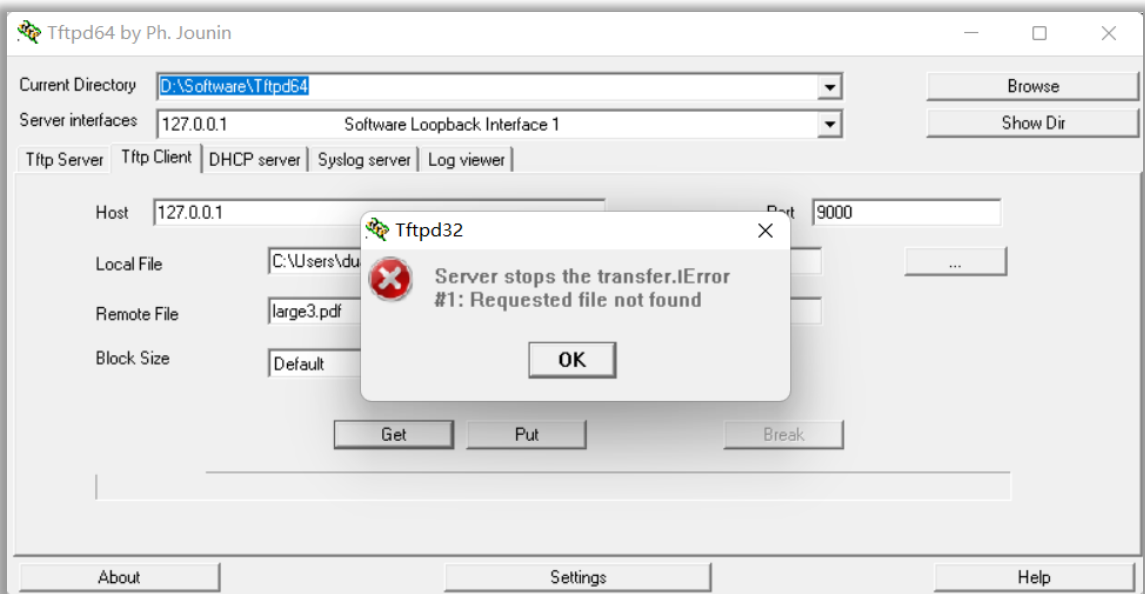


Figure 14. Error code 1 - Requested file not found

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:

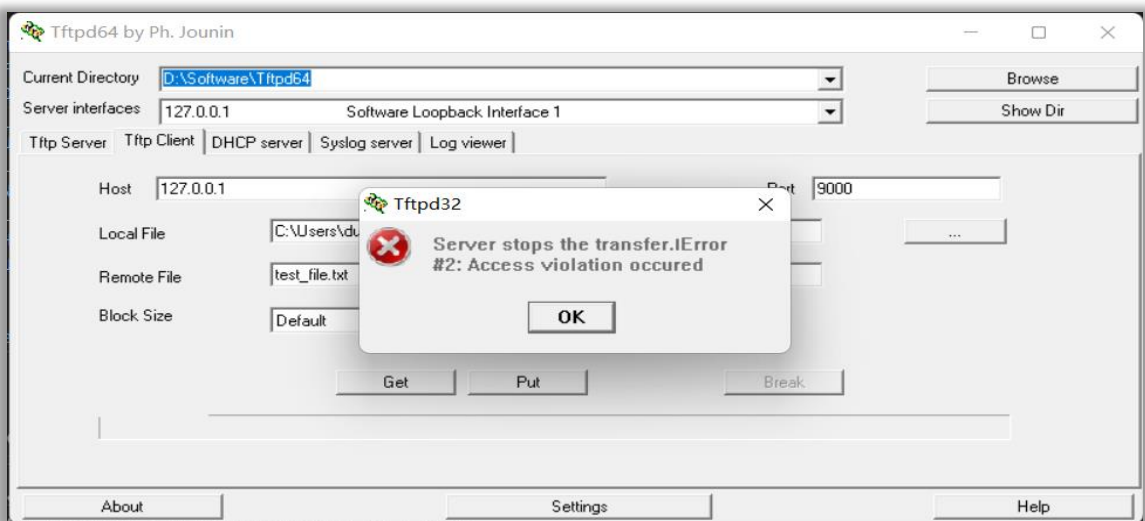


Figure 15. Error code 2 - Access violation occurred

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

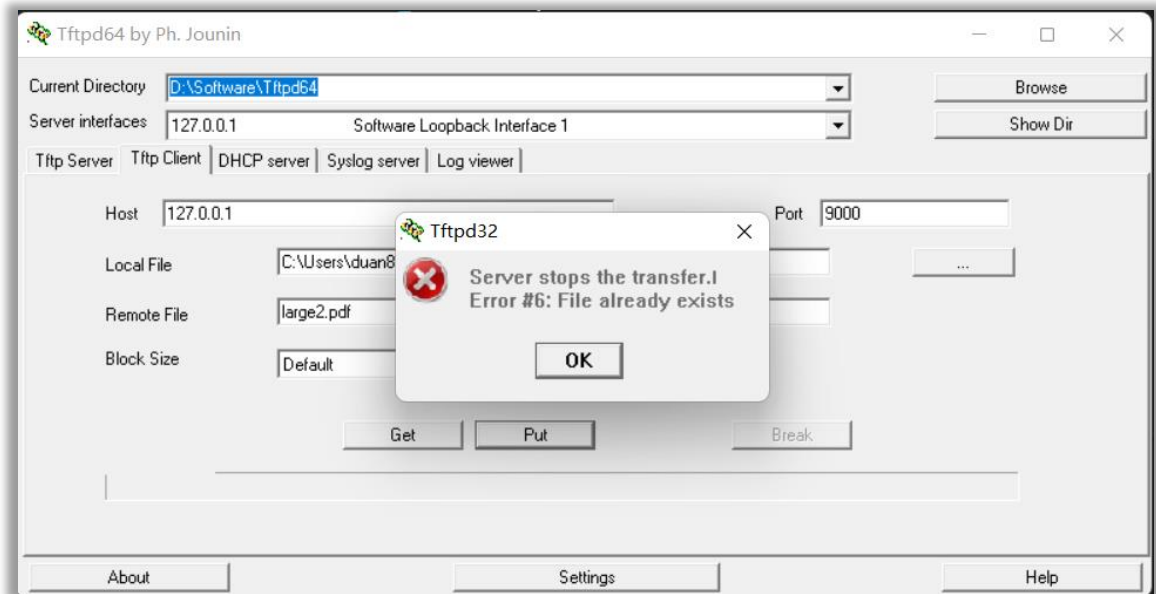


Figure 16. Error code 6 - File already exists

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.