Linnéuniversitetet
Kalmar Växjö

Report

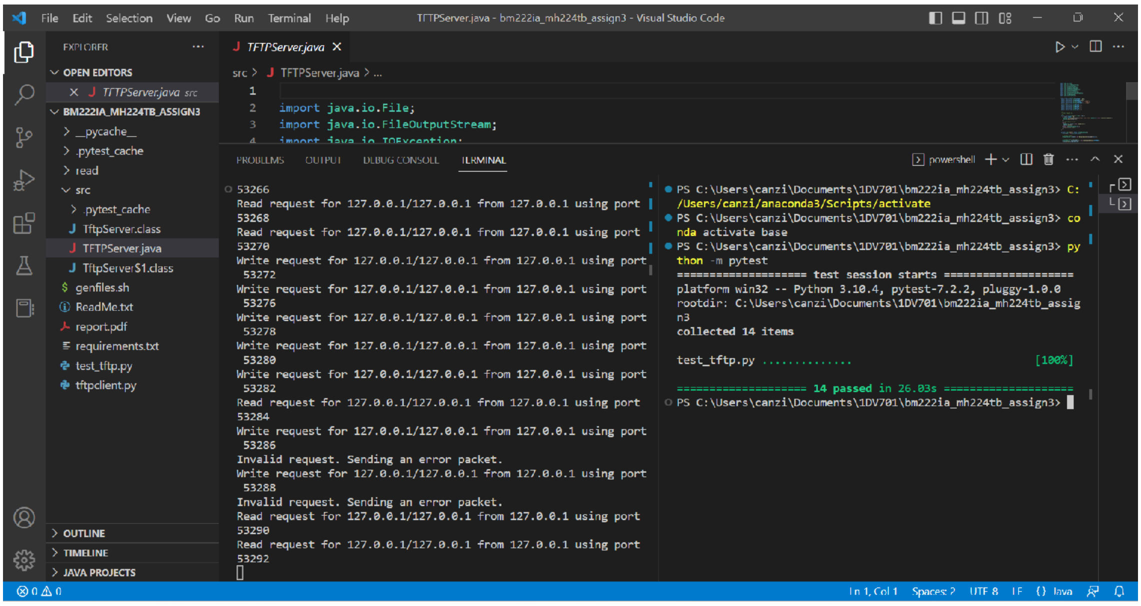
Assignment 3

1DV701

Author1: Baker Mohamad
Email bm225ia.
Author2: Mustafa Habeb
Email mh224tb.
Semester: Spring 2023

Contents

1 Problem 1



```java

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
```

```
53266
Read request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
53268
Read request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
53270
Write request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
 53272
Write request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
 53276
Write request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
 53278
Write request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
 53280
Write request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
 53282
Read request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
53284
Write request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
 53286
Invalid request. Sending an error packet.
Write request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
 53288
Invalid request. Sending an error packet.
Read request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
53290
Read request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port
53292
```

```
PS C:\Users\canzi\Documents\1DV701\bm222ia_mh224tb_assign3> C:
/Users/canzi/anaconda3/Scripts/activate
PS C:\Users\canzi\Documents\1DV701\bm222ia_mh224tb_assign3> co
nda activate base
PS C:\Users\canzi\Documents\1DV701\bm222ia_mh224tb_assign3> py
thon -m pytest
==================== test session starts ====================
platform win32 -- Python 3.10.4, pytest-7.2.2, pluggy-1.0.0
rootdir: C:\Users\canzi\Documents\1DV701\bm222ia_mh224tb_assig
n3
collected 14 items

test_tftp.py ..............                             [100%]

==================== 14 passed in 26.03s ====================
PS C:\Users\canzi\Documents\1DV701\bm222ia_mh224tb_assign3>
```

1.1 Discussion

At the beginning of working on the problem, we examined each part of the code to determine what we could achieve and what was needed to get a high grade. As a result, we implemented required functions in the code to enable it to handle more than 512 bytes from the outset.

The initial implementation of the first two functions was straightforward. We were able to retrieve the necessary data from the datagram socket and then return it without difficulty. For parsing the request, only the first short was needed, as it served as the opcode for this protocol.

Implementation of the "send_DATA_receive_ACK" function:

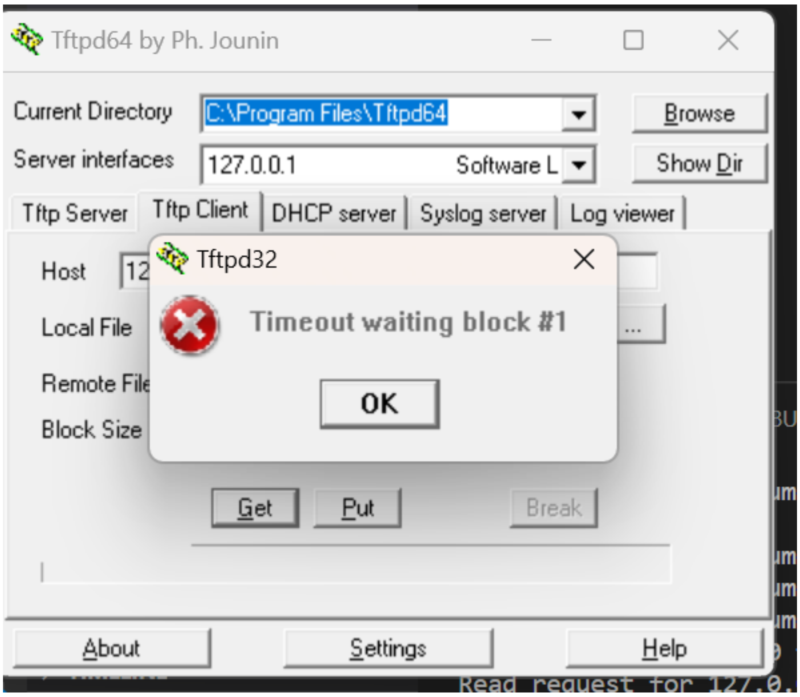
The first step is to check for the existence of the requested file. If it exists, the size of the data is checked to determine if it exceeds 512 bytes. If so, the data is divided into blocks, the size of the bytearray, and the number of blocks required is calculated. Then, the data is sent in 512-byte chunks until all the data has been transmitted.

For the "receive_DATA_send_ACK" function, we made an ack bytearray that we send every time a block is received. This array includes the block number and the ack opcode.

Socket: We use the socket to open the server and allow it to accept incoming connections via the TFTP port that we've designated. This enables any information that comes through that port to be directed to the server.

sendSocket : The sendSocket is used for communication with the connected client. The inet address is bound to the sendSocket, which is then connected to the client. This allows data to be sent and received from the client through the socket.

2 Problem 2



Tftpd64 by Ph. Jounin

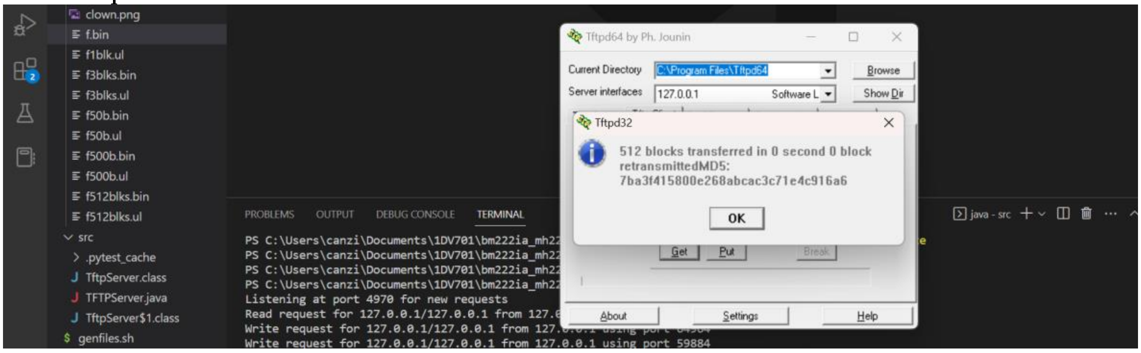
Current Directory: C:\Program Files\Tftpd64

Server interfaces: 127.0.0.1 Software L

Tftpd32

Timeout waiting block #1

Read request for 512 blocks:



Tftpd64 by Ph. Jounin

Current Directory: C:\Program Files\Tftpd64

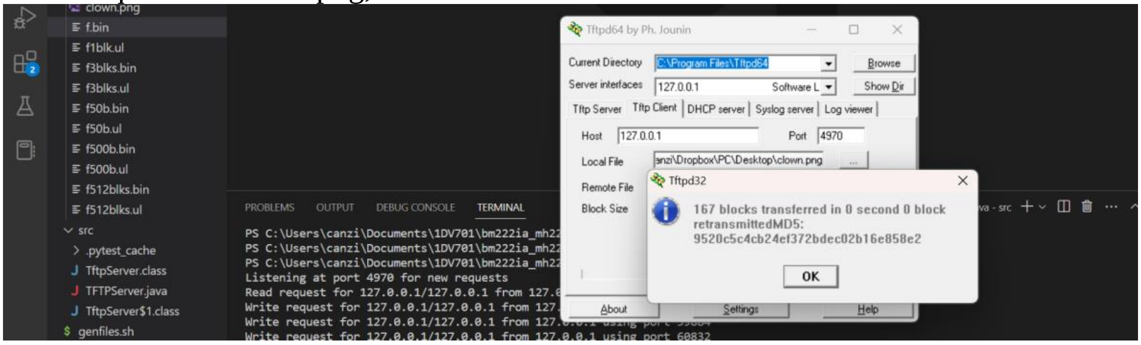
Server interfaces: 127.0.0.1 Software L

Tftpd32

512 blocks transferred in 0 second 0 block retransmittedMD5: 7ba3f415800e268abcac3c71e4c916a6

```
PS C:\Users\canzi\Documents\1DV701\bm222ia_mh22
PS C:\Users\canzi\Documents\1DV701\bm222ia_mh22
PS C:\Users\canzi\Documents\1DV701\bm222ia_mh22
PS C:\Users\canzi\Documents\1DV701\bm222ia_mh22
Listening at port 4970 for new requests
Read request for 127.0.0.1/127.0.0.1 from 127.
Write request for 127.0.0.1/127.0.0.1 from 127.
Write request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port 59884
```

Write request for clown.png, 14.3 MB.



Tftpd64 by Ph. Jounin

Current Directory: C:\Program Files\Tftpd64

Server interfaces: 127.0.0.1 Software L

Host: 127.0.0.1 Port: 4970

Local File: anzi\Dropbox\PC\Desktop\clown.png

Tftpd32

167 blocks transferred in 0 second 0 block retransmittedMD5: 9520c5c4cb24ef372bdec02b16e858e2

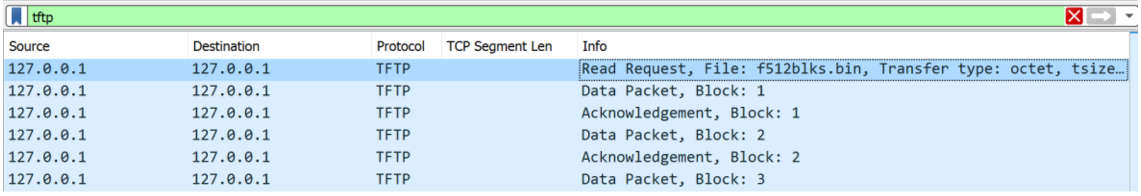
```
PS C:\Users\canzi\Documents\1DV701\bm222ia_mh22
PS C:\Users\canzi\Documents\1DV701\bm222ia_mh22
PS C:\Users\canzi\Documents\1DV701\bm222ia_mh22
Listening at port 4970 for new requests
Read request for 127.0.0.1/127.0.0.1 from 127.
Write request for 127.0.0.1/127.0.0.1 from 127.
Write request for 127.0.0.1/127.0.0.1 from 127.
Write request for 127.0.0.1/127.0.0.1 from 127.0.0.1 using port 60832
```

2.1 Discussion

Actually, we didn’t figure out how to make the file transfer retransmit. But we used the given Python test to check the timeouts and the retransmission, and we purposely caused some timeouts and then attempted to retransmit the file.

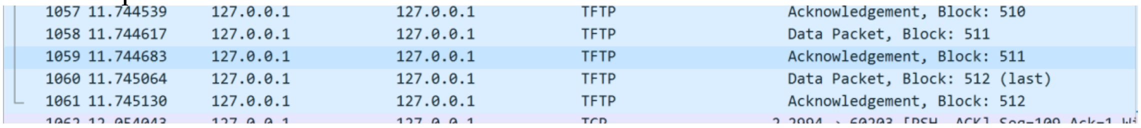
3 Problem 3

Read Request with the first block.



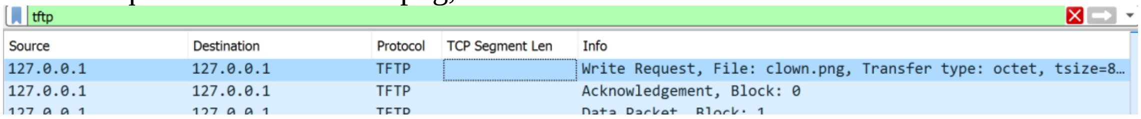
Source	Destination	Protocol	TCP Segment Len	Info
127.0.0.1	127.0.0.1	TFTP		Read Request, File: f512blks.bin, Transfer type: octet, tsize…
127.0.0.1	127.0.0.1	TFTP		Data Packet, Block: 1
127.0.0.1	127.0.0.1	TFTP		Acknowledgement, Block: 1
127.0.0.1	127.0.0.1	TFTP		Data Packet, Block: 2
127.0.0.1	127.0.0.1	TFTP		Acknowledgement, Block: 2
127.0.0.1	127.0.0.1	TFTP		Data Packet, Block: 3

Read Request with the last block.



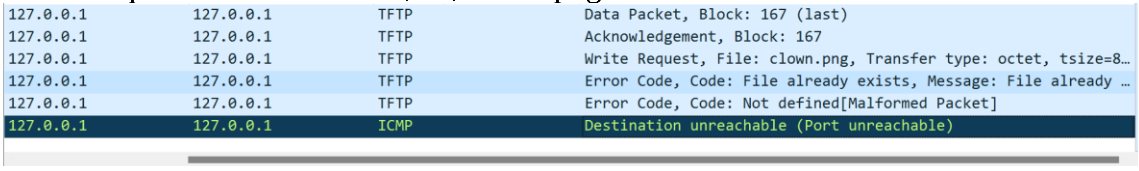
```
1057 11.744539   127.0.0.1   127.0.0.1   TFTP     Acknowledgement, Block: 510
1058 11.744617   127.0.0.1   127.0.0.1   TFTP     Data Packet, Block: 511
1059 11.744683   127.0.0.1   127.0.0.1   TFTP     Acknowledgement, Block: 511
1060 11.745064   127.0.0.1   127.0.0.1   TFTP     Data Packet, Block: 512 (last)
1061 11.745130   127.0.0.1   127.0.0.1   TFTP     Acknowledgement, Block: 512
1062 12.054043   127.0.0.1   127.0.0.1   TCP    2 2994 → 60203 [PSH, ACK] Seq=100 Ack=1 Wi
```

Write Request for a file clown.png, 14.3 MB.



Source	Destination	Protocol	TCP Segment Len	Info
127.0.0.1	127.0.0.1	TFTP		Write Request, File: clown.png, Transfer type: octet, tsize=8…
127.0.0.1	127.0.0.1	TFTP		Acknowledgement, Block: 0
127.0.0.1	127.0.0.1	TFTP		Data Packet, Block: 1

Write Request for the same file, i.e, clown.png.



```
127.0.0.1   127.0.0.1   TFTP   Data Packet, Block: 167 (last)
127.0.0.1   127.0.0.1   TFTP   Acknowledgement, Block: 167
127.0.0.1   127.0.0.1   TFTP   Write Request, File: clown.png, Transfer type: octet, tsize=8…
127.0.0.1   127.0.0.1   TFTP   Error Code, Code: File already exists, Message: File already …
127.0.0.1   127.0.0.1   TFTP   Error Code, Code: Not defined[Malformed Packet]
127.0.0.1   127.0.0.1   ICMP   Destination unreachable (Port unreachable)
```

3.1 Discussion

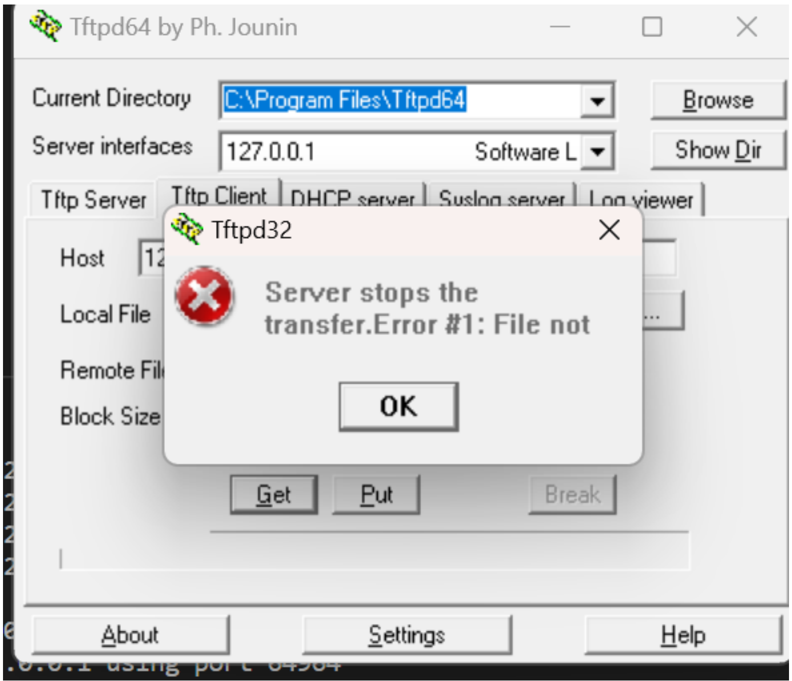
We initiated a read request for the file named "f512blks.bin" with a type and tsize of 0, since we were retrieving data rather than transmitting it. We transmitted data packets and received acks. This process was repeated for all 512 blocks.

3.2 Discussion

Similar to the read request, first an usual connectionwas established. Then client sends a write request that includes the name, type, and size of the file being transferred. The next step is to send the first ack opcode to acknowledge the received message, and then begin receiving the data.
If a file with the same name already exists in our read folder, we send an error code back to the server indicating that the file already exists and the connection will be terminated.
Differences between read, and write request

- The client does not send a tsize value when sending a read request because they do not have information about the size of the file they are requesting.
- in contrast to read requests, when the client sends a file to the server, we receive the tsize value.
- When receiving a read request, there is no need to send an acknowledgement message before starting to receive the file. Regarding write requests, we need to acknowledge the client's request to send a file.

4 Errors



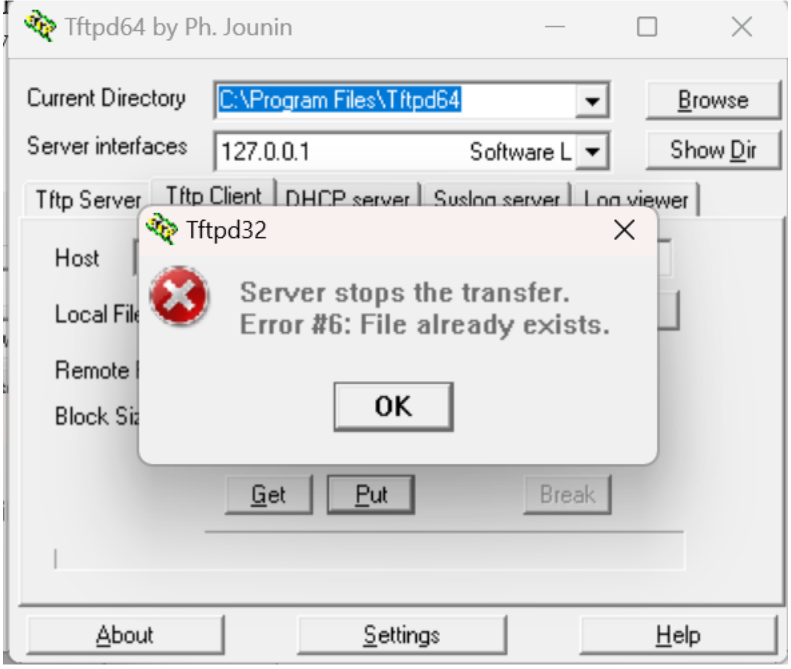
Current Directory: C:\Program Files\Tftpd64

Server interfaces: 127.0.0.1 Software L

Tftpd32

Server stops the transfer.Error #1: File not

.0.0.1 using port 64984



Current Directory: C:\Program Files\Tftpd64

Server interfaces: 127.0.0.1 Software L

Tftpd32

Server stops the transfer. Error #6: File already exists.