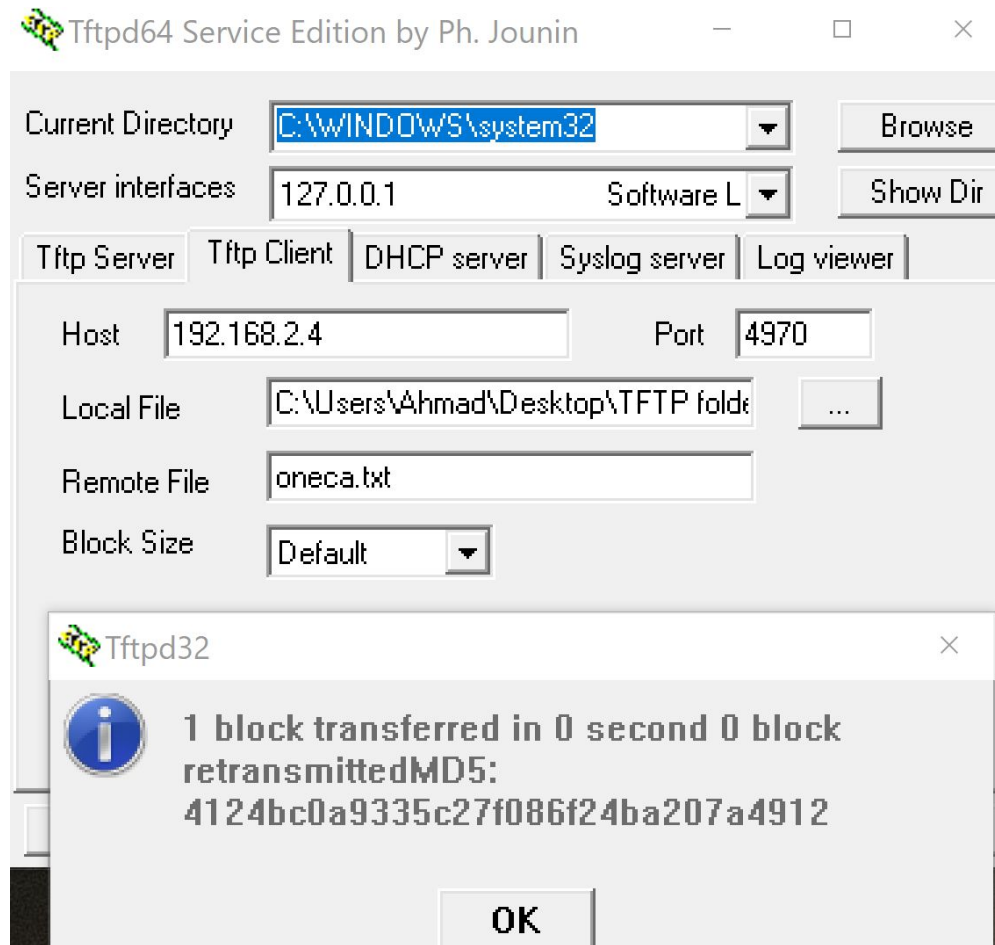


Problem 1

request to read a file that is shorter than 512 bytes.

The screenshot listed below displays the results of a read a file shorter than 512 bytes from the TFTP Client to Server.



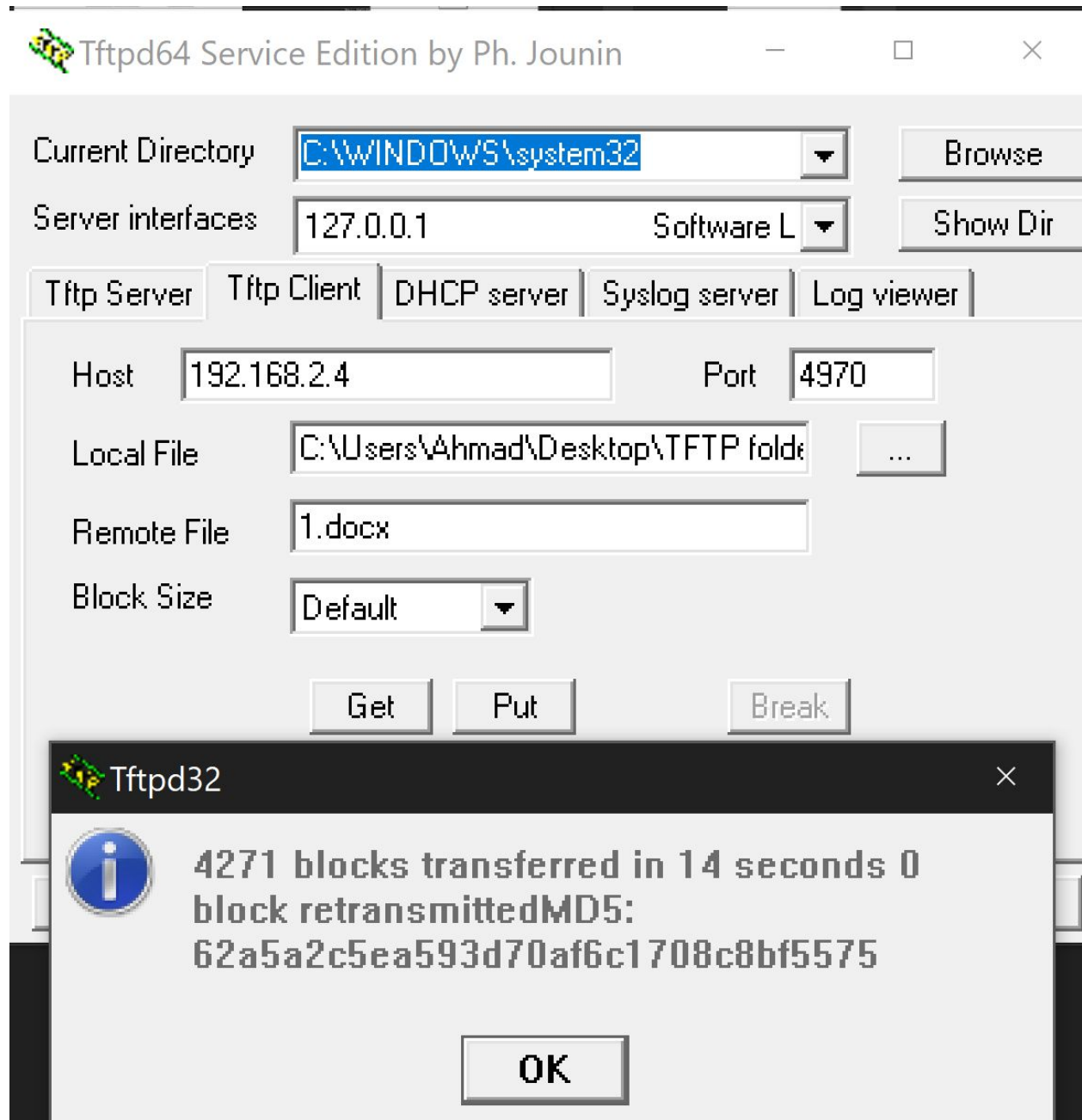
why it uses both socket and sendSocket?

The use of both sockets is for socket one initializes a connection and the other socket is to send packets across the Client and Server.

Problem 2

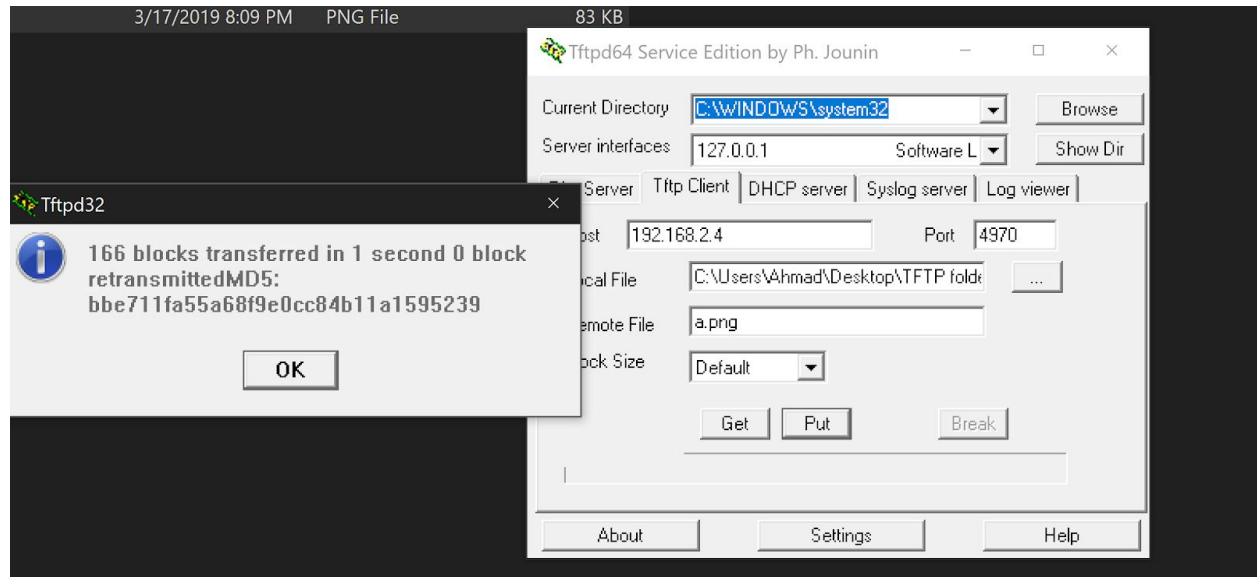
Read files larger than 512 bytes.

The screenshot listed below displays the results of a read a file larger than 512 bytes from the TFTP Client to Server. Since the file is larger than 512 we get 4271 blocks.

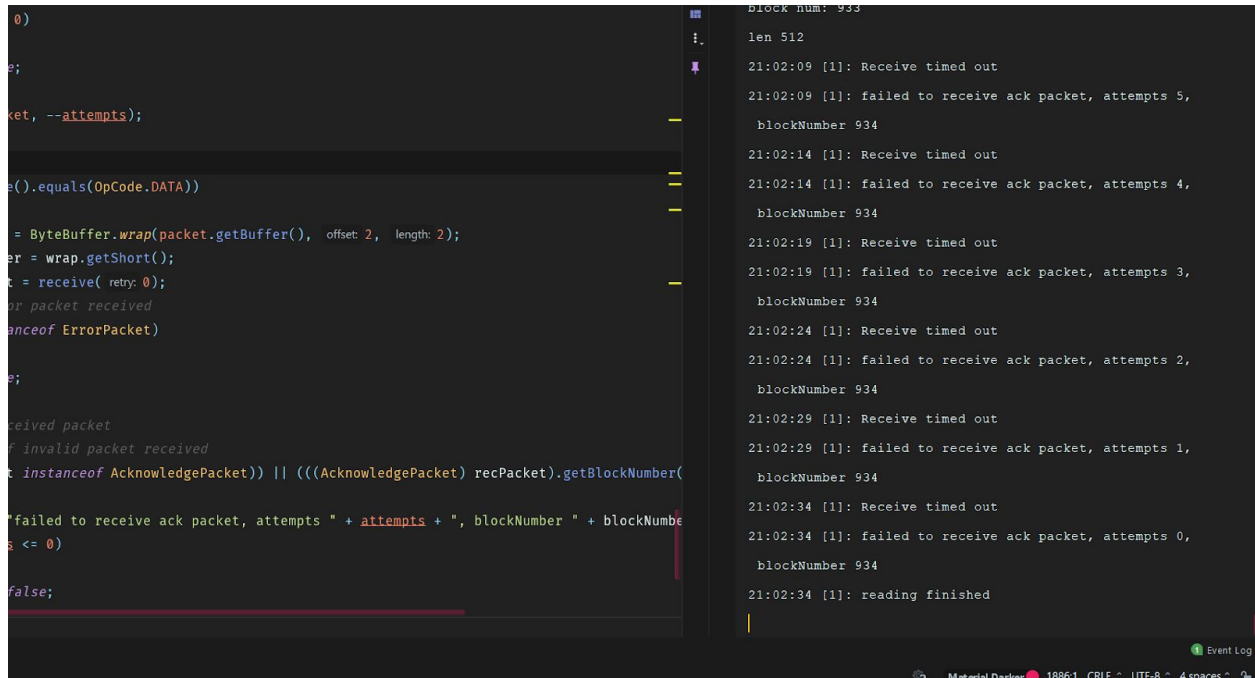


Write files larger than 512 bytes.

The screenshot listed below displays the results of a writing a file larger than 512 bytes from the TFTP Client to Server. Since this is a larger than 512 we get 166 blocks.



How we tested timeouts and retransmissions



```
0)
;
ket, --attempts);

e().equals(OpCode.DATA))

= ByteBuffer.wrap(packet.getBuffer(), offset 2, length 2);
er = wrap.getShort();
t = receive( retry: 0);
or packet received
anceof ErrorPacket)
;

ceived packet
f invalid packet received
t instanceof AcknowledgePacket)) || (((AcknowledgePacket) recPacket).getBlockNumber(

"failed to receive ack packet, attempts " + attempts + ", blockNumber " + blockNumbe
s <= 0)

false;
```

```
Block num: 933
len 512
21:02:09 [1]: Receive timed out
21:02:09 [1]: failed to receive ack packet, attempts 5,
blockNumber 934
21:02:14 [1]: Receive timed out
21:02:14 [1]: failed to receive ack packet, attempts 4,
blockNumber 934
21:02:19 [1]: Receive timed out
21:02:19 [1]: failed to receive ack packet, attempts 3,
blockNumber 934
21:02:24 [1]: Receive timed out
21:02:24 [1]: failed to receive ack packet, attempts 2,
blockNumber 934
21:02:29 [1]: Receive timed out
21:02:29 [1]: failed to receive ack packet, attempts 1,
blockNumber 934
21:02:34 [1]: Receive timed out
21:02:34 [1]: failed to receive ack packet, attempts 0,
blockNumber 934
21:02:34 [1]: reading finished
```

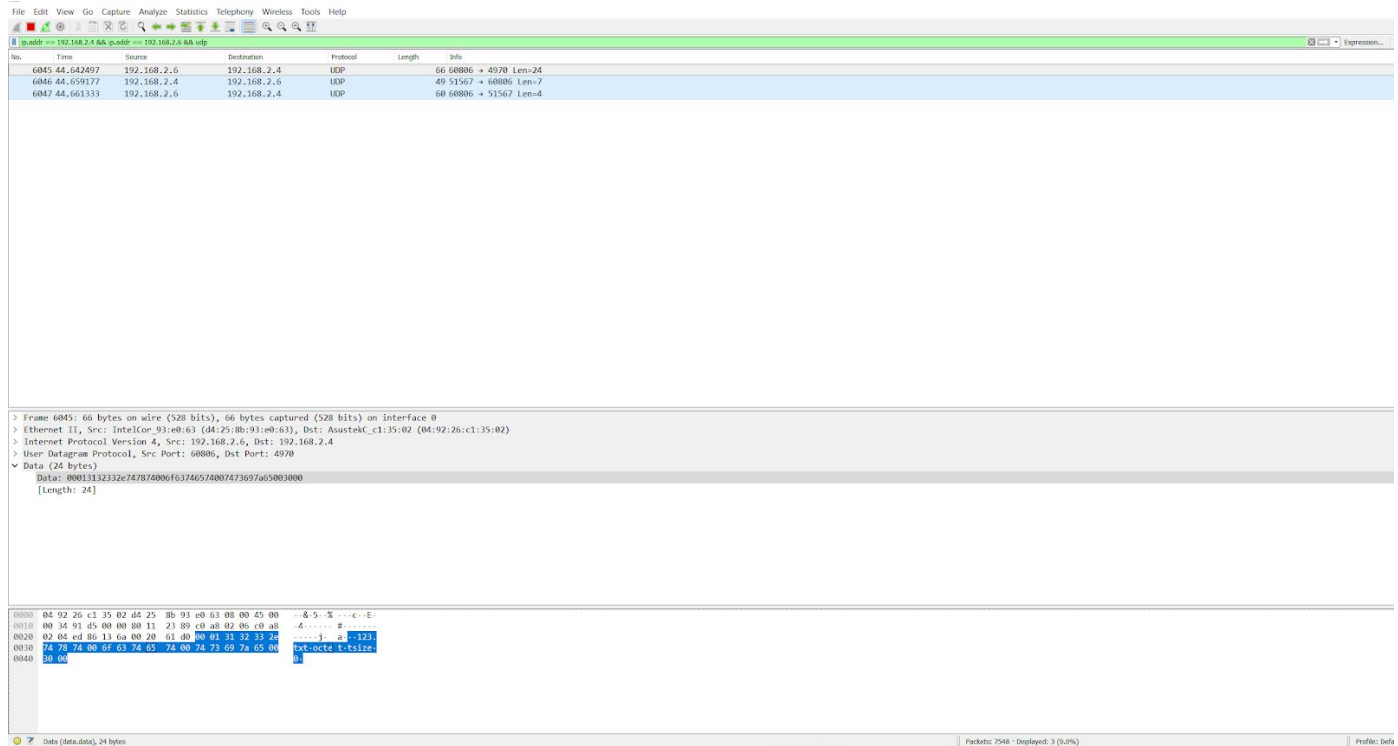
Event Log

Material Darker 1886:1 CRLF : UTF-8 : 4 spaces

To test the retransmission and timeout we had to disconnect the connection manually from the client side from the internet so since it is a UDP Protocol and a connectionless based the server kept sending packets, however client can not receive since it is disconnected and this results in the client not sending Ack to the server and we have the Server repeat it self for 5 attempts and then timeout.

VG task 1

The following Wireshark screenshot captures the traffic between two windows laptops when reading a file 1 KB “123.txt”



Since UDP protocol is under TFTP protocol, I have filtered the packet by UDP and the IP addresses to my IP address (192.168.2.4) and my partner windows laptop IP address (192.168.2.6) client.

In the first line the screenshot displays when the client requested a packet with length 24 byte from his IP address and port 60806 to my computer address (server) on port 4970.

In the second line an **Acknowledgment** packet sent from the client as a reply to the server that the **DATA** packet has been received. Server sent the packet from port 51567 to the client's port 60806.

In the third line, is the last line which the client terminates the connection as an ack packet

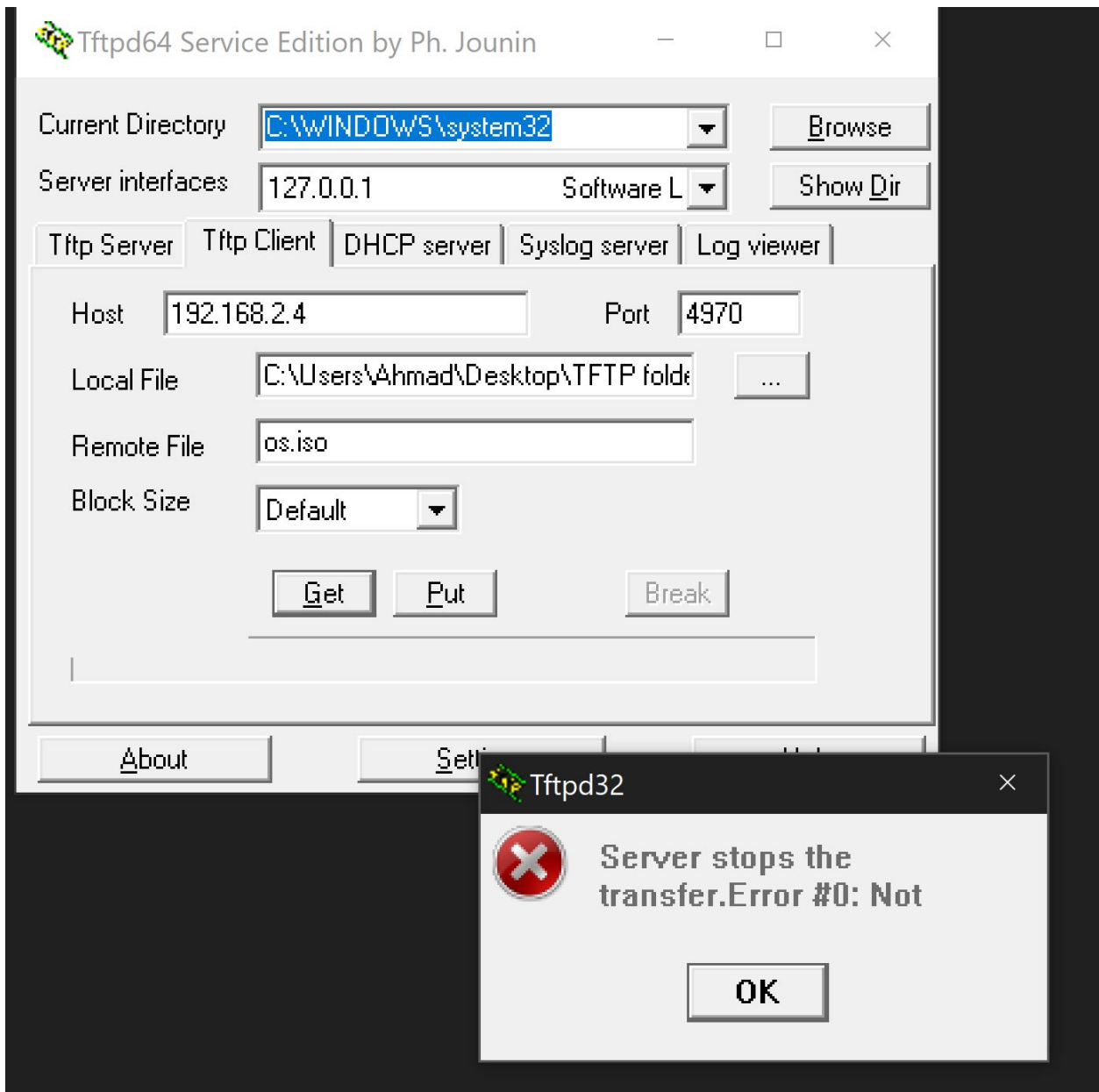
Difference between a read and a write request:**Read Request:**

Uses “get” to make the read request. The server sends a initial Data packet of the specified file to the client after it receives the “get” or the opcode 1 and after sending it the server wait for acknowledgment packet from the client when the server receives the acknowledgment packet. if the last packet of a full block (512 bytes) the server sends a new packet.

Write Request:

Uses “put” to make the write request. The server sends an acknowledgment packet to the client if “put” or opcode 2 is received with a block of the same number. The server waits for a Data packet to be received from the client and when the server receives the packet the server sends the acknowledgment packet back to the client with block of the same number that equals the Data packet received from the client. if the packet that was received was a full block (512 bytes) The process is repeated.

Problem 3
Error Code 0



```
// In case we want to disconnect the client
if (!active) {
    send(PacketFactory.createErrorPacket(ErrorCode.NOT_DEFINED));
    break;
}
```


Error Code 1

Current Directory:

Server interfaces:


Tftp Server | Tftp Client | DHCP server | Syslog server | Log viewer

Host: Port:

Local File:

Remote File:

Block Size:

 Server stops the transfer.Error #1: File not

Error Code 2

Server interfaces:


Tftp Server | Tftp Client | DHCP server | Syslog server | Log viewer

Host: Port:

Local File:

Remote File:

Block Size:

 Server stops the transfer.Error #2: Access

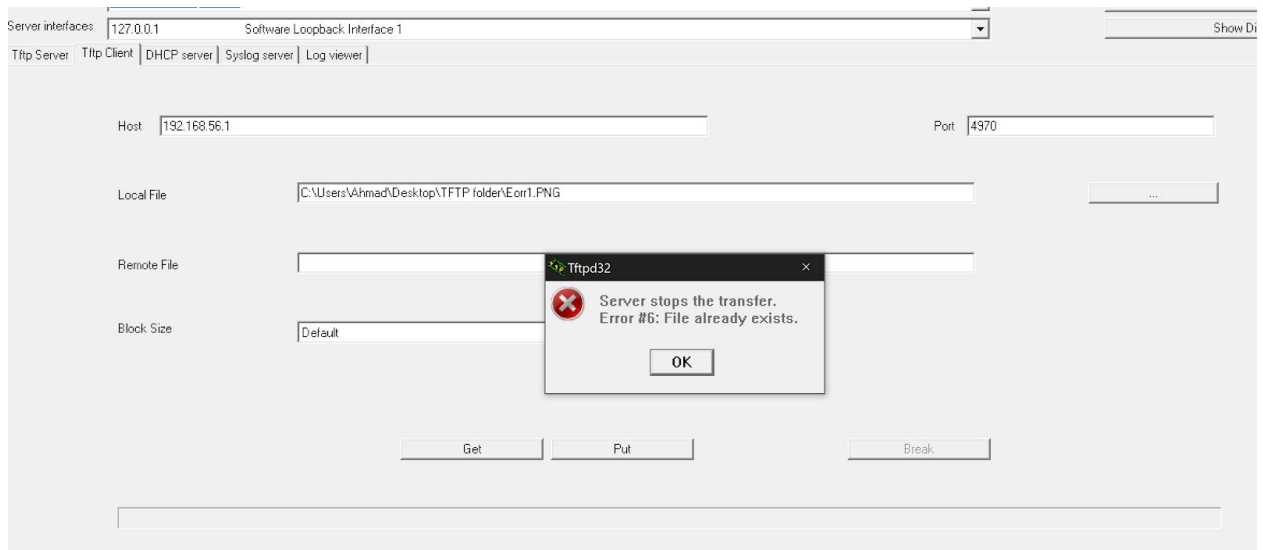
```
156
157
158
159
160
161
162
163
164
165

try
{
    fileInputStream.getChannel().position(totalBytes);
    totalBytes += fileInputStream.read(fileBuf, off: 0, len: 512);
    throw new IOException();
} catch (IOException e)
{
    logMessage("reading failed, " + e.getMessage());
    send(PacketFactory.createErrorPacket(ErrorCode.ACCESS_VIOLATION));
    return;
}
```

Client → handleReadRequest()

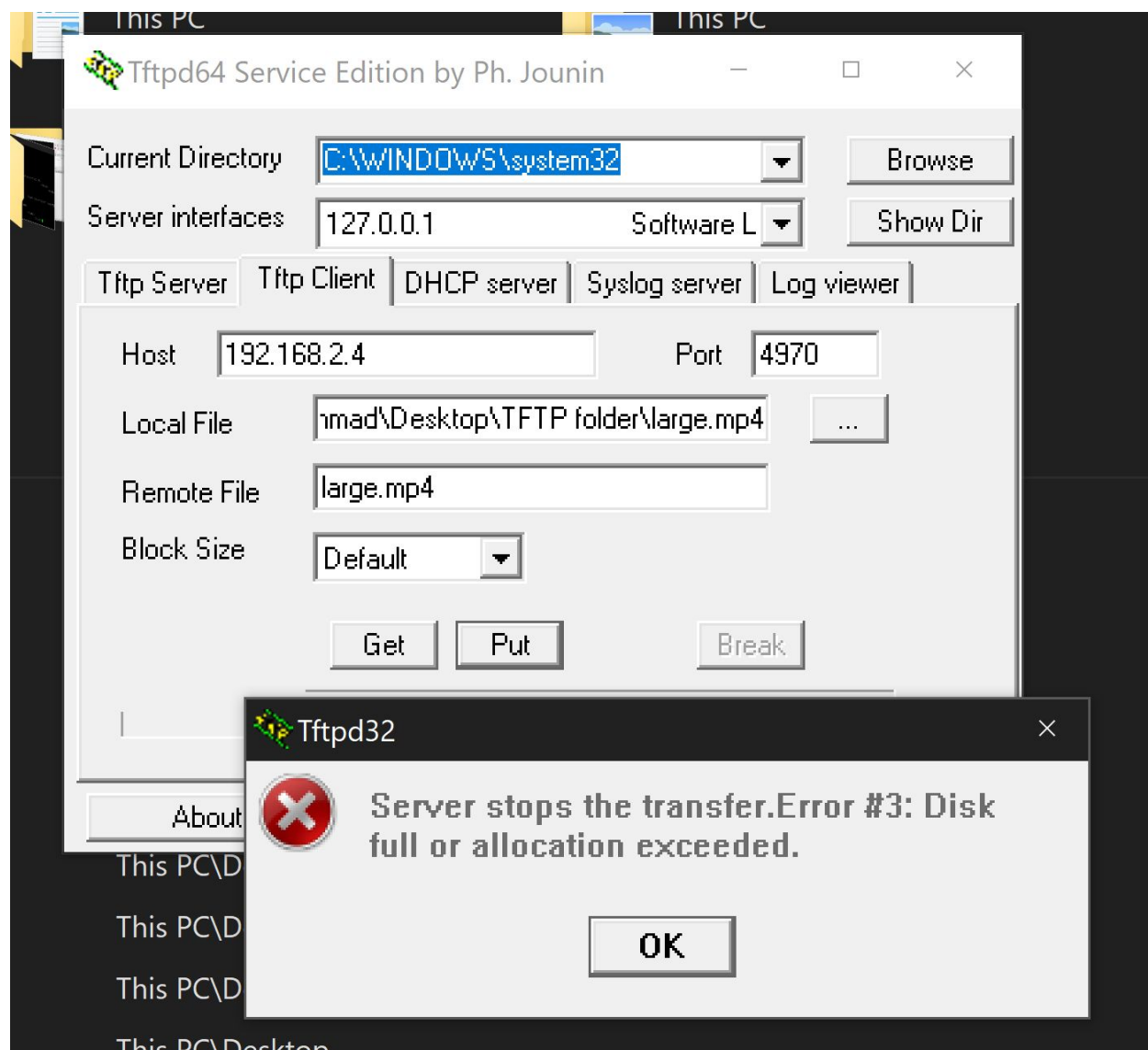
ges g: TODO

Error Code 6



VG task 2

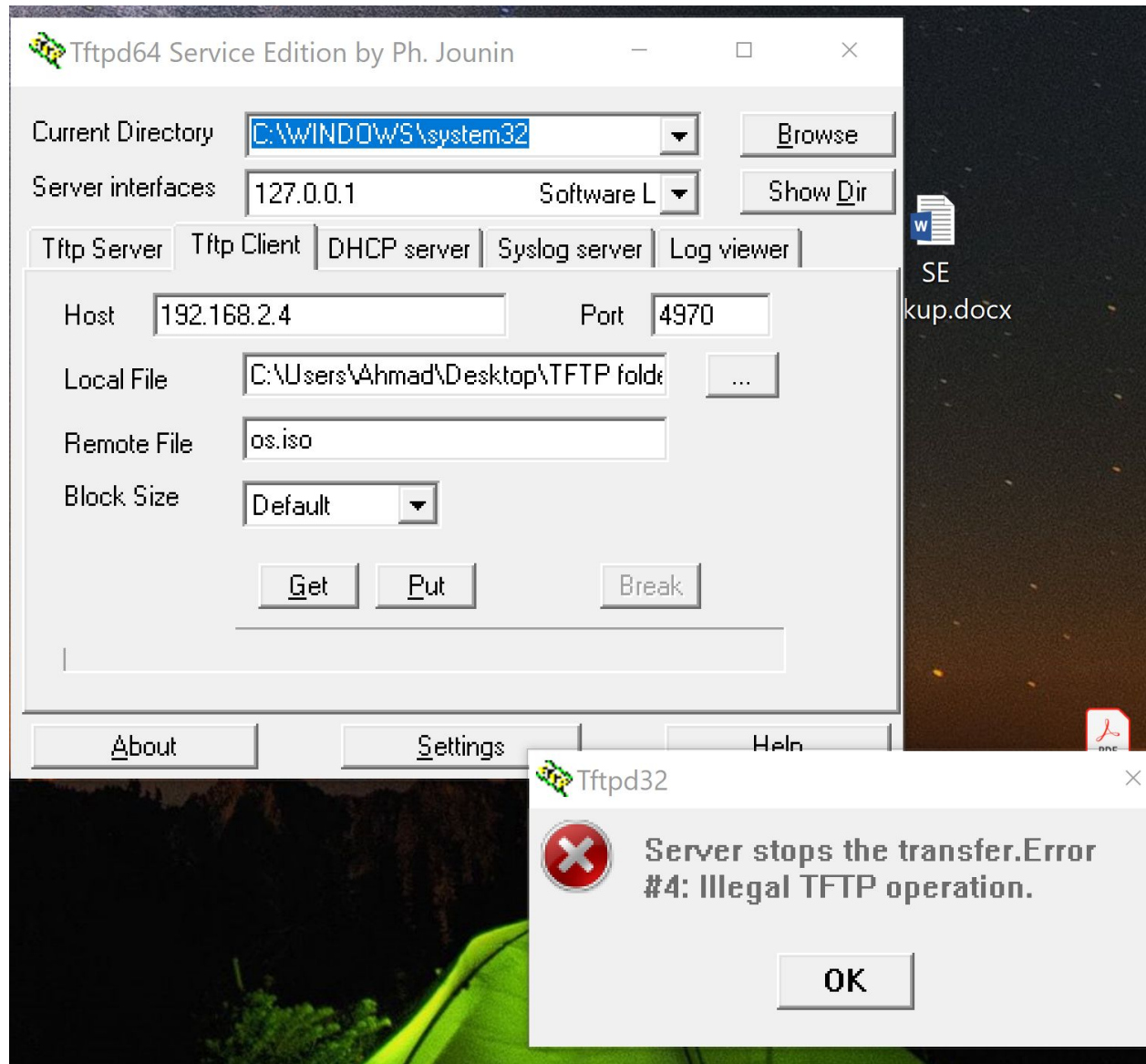
Error Code 3



```
20:39:06 [2]: client started
20:39:06 [2]: writing large.mp4
20:39:25 [2]: writing failed, There is not
           enough space on the disk
20:39:25 [2]: writing finished
```

To generate this error we have used a USB Flash Drive with limited capacity and we made write request with a large file that is bigger than the capacity of the Flash Drive.

Error Code 4



```

    return,
}

switch (packet.getOpCode())
{
    /*
    case RRQ:
        handleReadRequest((RequestPacket) packet);
        break;
    case WRQ:
        handleWriteRequest((RequestPacket) packet);
        break;
    */
    default:
        logMessage("illegal TFTP operation");
        send(PacketFactory.createErrorPacket(ErrorCode.ILLEGAL_TFTP_OPERATION));
        break;
}
socket.close();
}

```

Error Code 5

Expiation

Contribution Summery

We both worked 50% on this assignment