Linnéuniversitetet
Kalmar Växjö

Report

Assignment 3
1DV701

Author: Katerina Ioannidou
Semester: Spring 2023
*Ema*il: ki222ea@student.lnu.se

Author: Katarina Simakina
Semester: Spring 2023
Email: es225hi@student.lnu.se

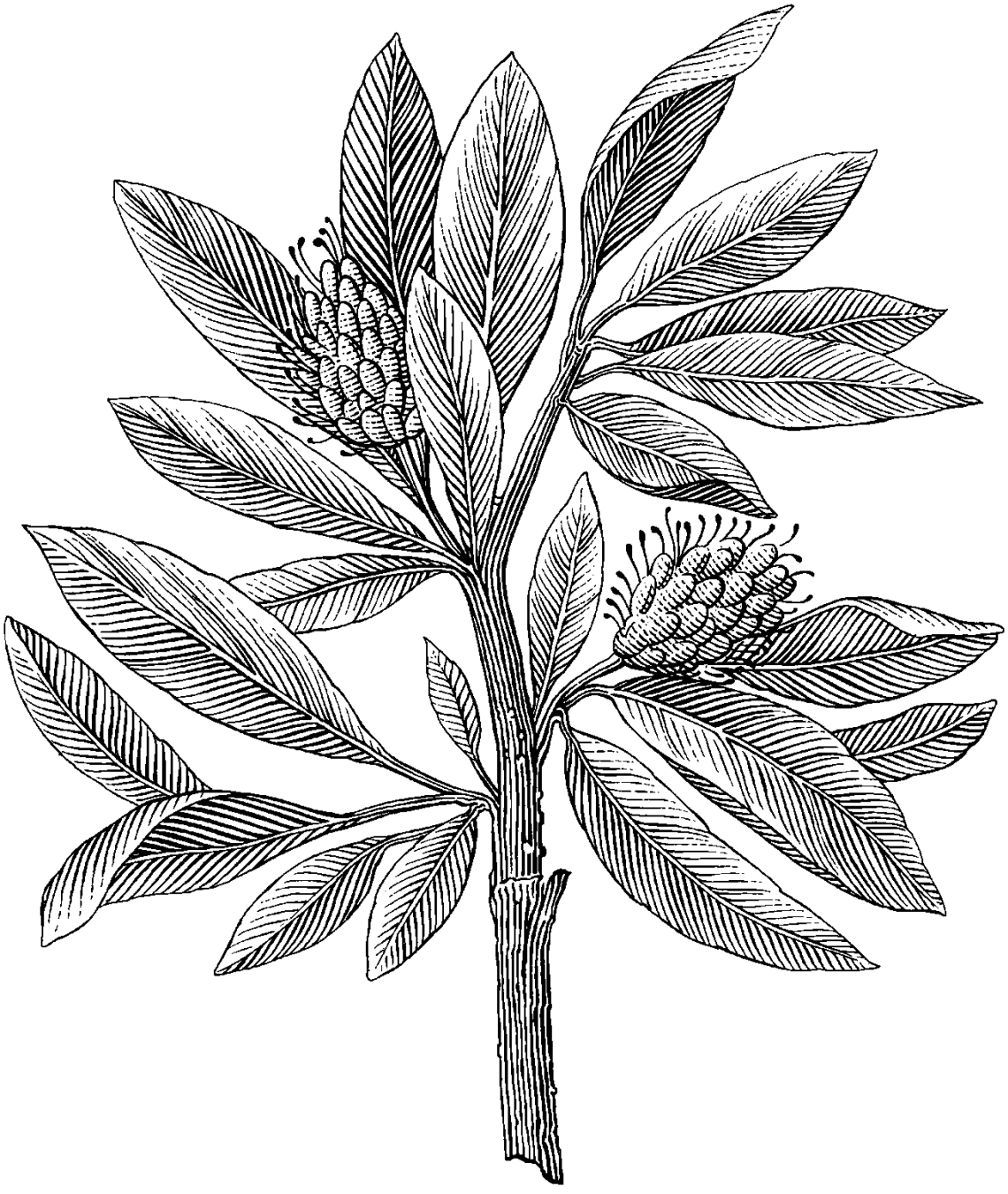
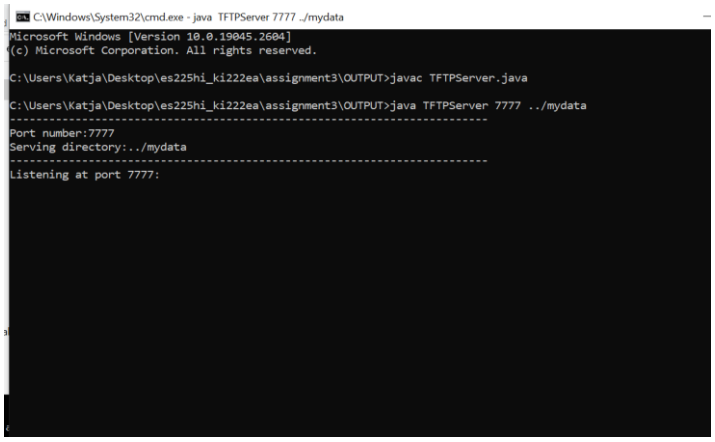


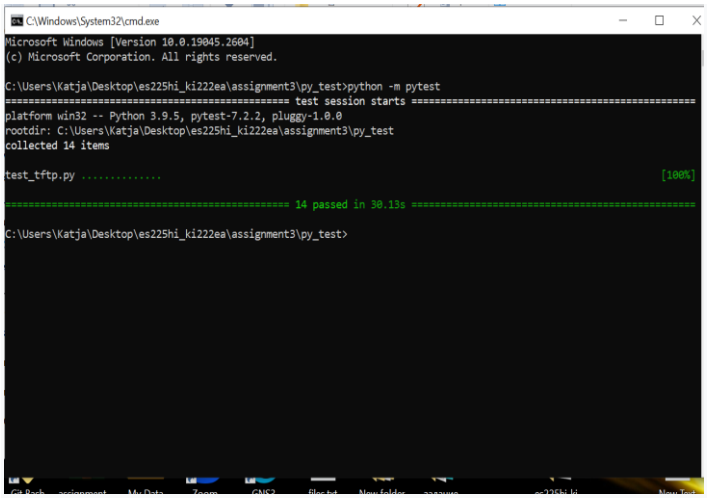
Table of Contents

Problem 1.21.1 Discussion 2
Problem 2. 5
2.1 Discussion 5
Problem 3. 6
3.1 Discussion 6
Problem 4. 12
4.1Discussion 12

Summary 14

Before describing problems from assignment 3 we would like to present the results of the testing of the code in accordance with tasks from this assignment. Please look below.

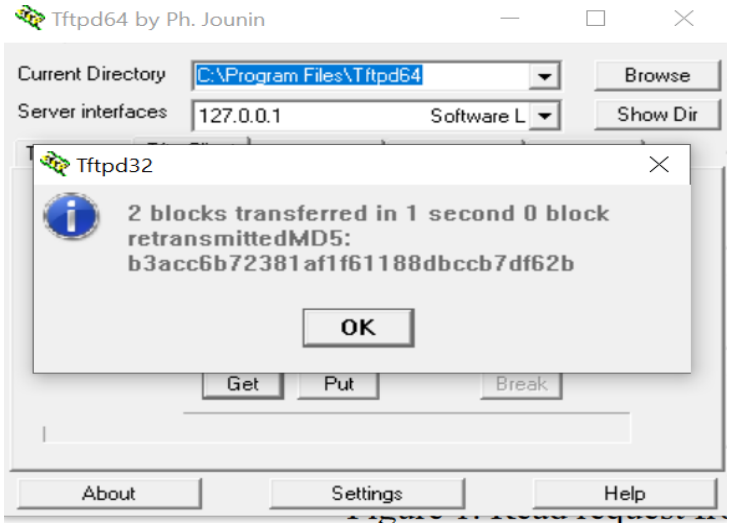




Problem 1. Implementing TFTP server according to RFC1350.

The TFTP server has been implemented according to RFC1350.

The client sends a request to the TFTP server to read or write a file.
The server grants the request and opens the connection. It sends the file in the form of data packets, each packet containing 512 bytes. Each packet should be acknowledged by the client to the server using the TFTP packet block number. A data packet with less than 512 bytes content signals that it is the last packet in the transmission for this particular file. If the TFTP server does not receive an acknowledgement due to packet loss, it retransmits for a specified period of time and then times out. If an error occurs, an error packet is sent with a message describing the problem. If an error occurs, the connection is terminated. TFTP data packets are sent one at a time; once an acknowledgement is received for one packet, the next packet is sent.



Read request (the port 7777 has been defined). It takes 1 second for this operation, transmission.

To make this implementation we have done the following steps:

1. Implementing the method receiveFrom(). This function takes in a DatagramSocket object and a byte array as its arguments. The DatagramSocket is a Java class used to implement the Datagram communication protocol, which is a connectionless protocol used for transmitting data over the internet. The purpose of the receiveFrom function is to receive a datagram packet on the given socket, and return the InetSocketAddress of the sender.
2. Implementing the method private int parseRQ(). This function parses a Request packet, extracting the opcode and filename from the input buffer. The filename is stored in a StringBuffer object called requestedFile.
3. Implementing the method private boolean receive_DATA_send_ACK(). The function receives a block of data from the server and sends an acknowledgment (ACK) to the server indicating that the block has been received successfully.
4. Implementing the method private boolean send_DATA_receive_ACK(). This function is responsible for sending a data packet and waiting for an acknowledgement packet in response.
5. Implementing the method public void readRequestOperationProcessing().This function reads data from a file and sends it over the network using the DatagramSocket object. It also ensures that the data is successfully transmitted by using a block counter and acknowledgement mechanism.
6. Implementing the method public void writeRequestOperationProcessing().This method handles the write request operation processing. It receives a DatagramSocket instance and an OutputStream instance, and uses them to establish a socket connection and write data to the output stream.
7. Implementing the method private void handleRQ(). This method handles the read and write requests sent by the client.
8. The Class TFTPServer, method private void start (). This method starts the TFTP server and listens for incoming requests on a specified port. It receives the requests and spawns a new thread to handle the request.

The TFTP server implemented in this code is capable of handling write request operations, where a client sends a request to write a file to the server. The server receives

the request, establishes a socket connection with the client using a DatagramSocket, and then begins to transfer the requested file.

The TFTP server listens on a specified port for incoming requests and receives the requests by creating a DatagramSocket. The server then parses the received request and spawns a new thread to handle the request. The request is handled based on the request type (read or write).

The "start" method initializes the server and continuously listens for incoming requests. When a request is received, the server creates a new thread to handle the request. The "receiveFrom" method receives data from the DatagramSocket and returns the InetSocketAddress of the sender. The "parseRQ" method extracts the requested file name from the received request message.

The "handleRQ" method handles the read and write requests sent by the client. The method checks if the requested file exists and if it can be read or written. If the file exists and can be read or written, the method handles the request by sending the requested file or receiving the file to be written. If the file does not exist or cannot be read or written, the method sends an error packet with the appropriate error code.

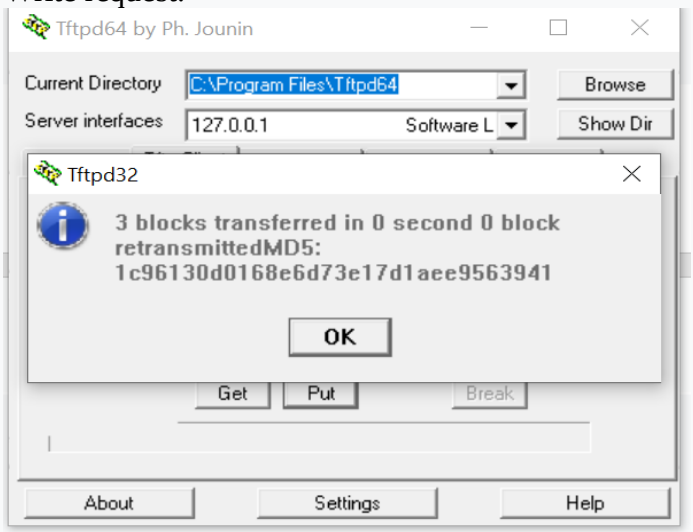
We have used socket and sendSocket as two sockets are used to separate the responsibilities of receiving and sending datagram packets in the TFTP protocol. The first socket is used to receive packets and the second socket is used to send packets back to the client.

The first socket is created to listen for incoming requests on the specified port using DatagramSocket socket = new DatagramSocket(null); and then bound to the local bind point using socket.bind(localBindPoint);. This socket is used to receive datagram packets from the client and parse the request by calling receiveFrom(socket, buf) and parseRQ(buf, requestedFile) respectively.

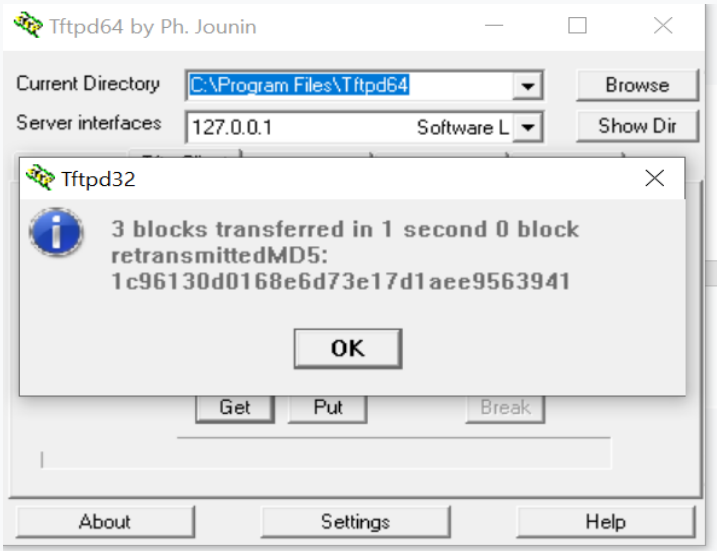
The second socket is created inside the new thread to handle the request, using DatagramSocket sendSocket = new DatagramSocket(0); and then connected to the client's address using sendSocket.connect(clientAddress);. This socket is used to send the response back to the client, using sendSocket.send(sendPacket); in the TFTPService class.

Problem 2.
Discussion 2.1

We created the file which more than 512 bytes. It was tested timeouts and retransmissions.
Write request.

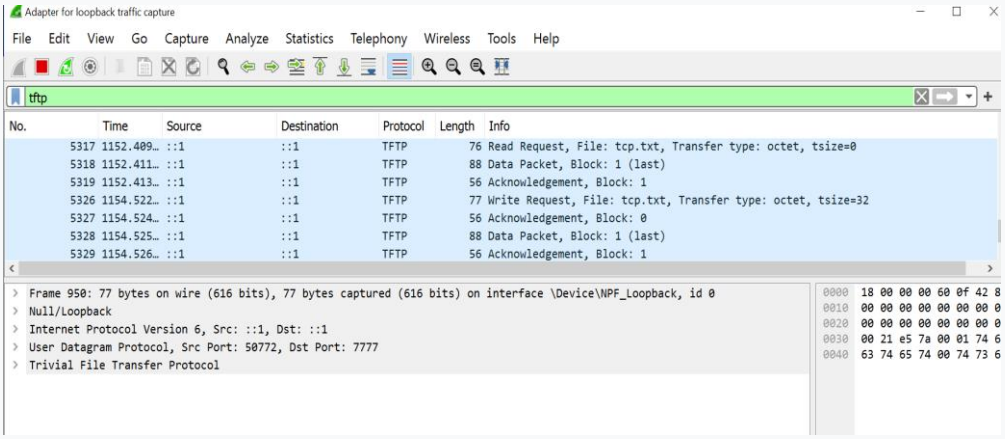


Read request.

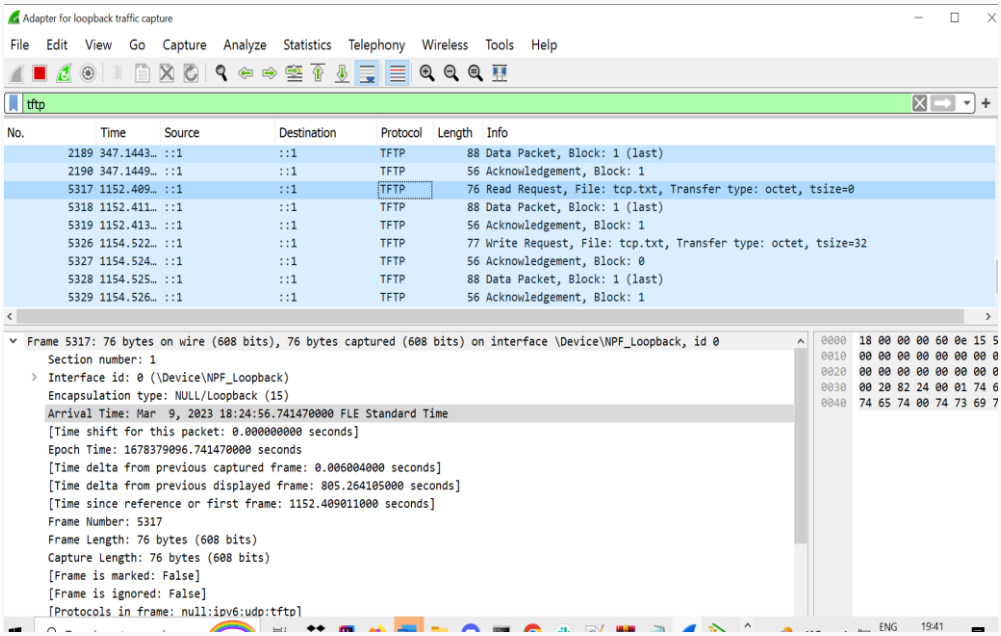


Problem 3.

Discussion 3.1.



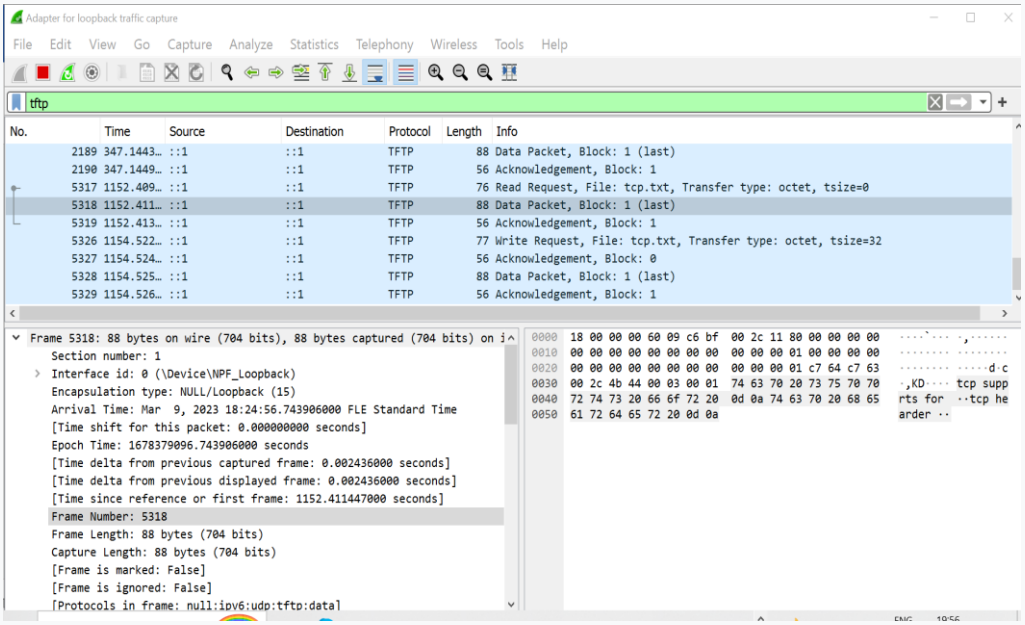
Read request, first line.

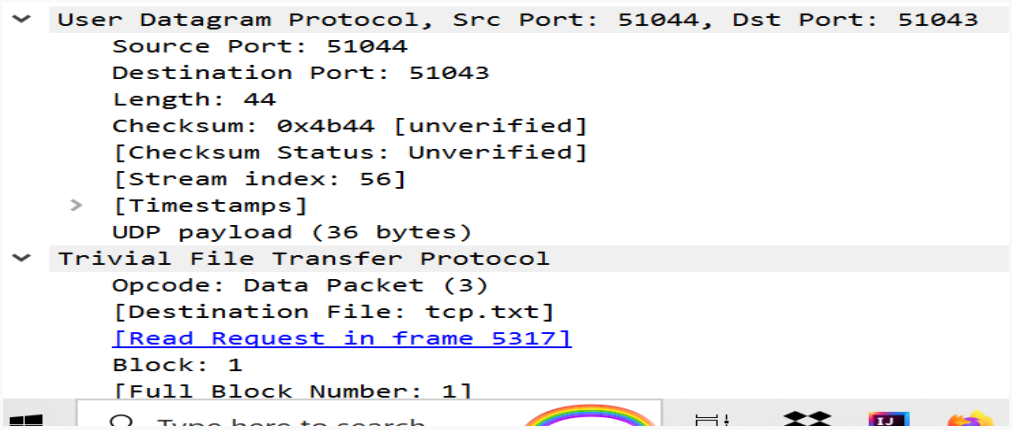


Wireshark captures the read request. Three lines belong to read request.
First line gives the following information about this reading request:

-Name of the file is tcp.txt,
-Transfer type is octet.
-tsize=0.
As well in bottom part:
-number of the frame is 5317,
-it was captured 76 bytes (608 bits),

Read request, second line.



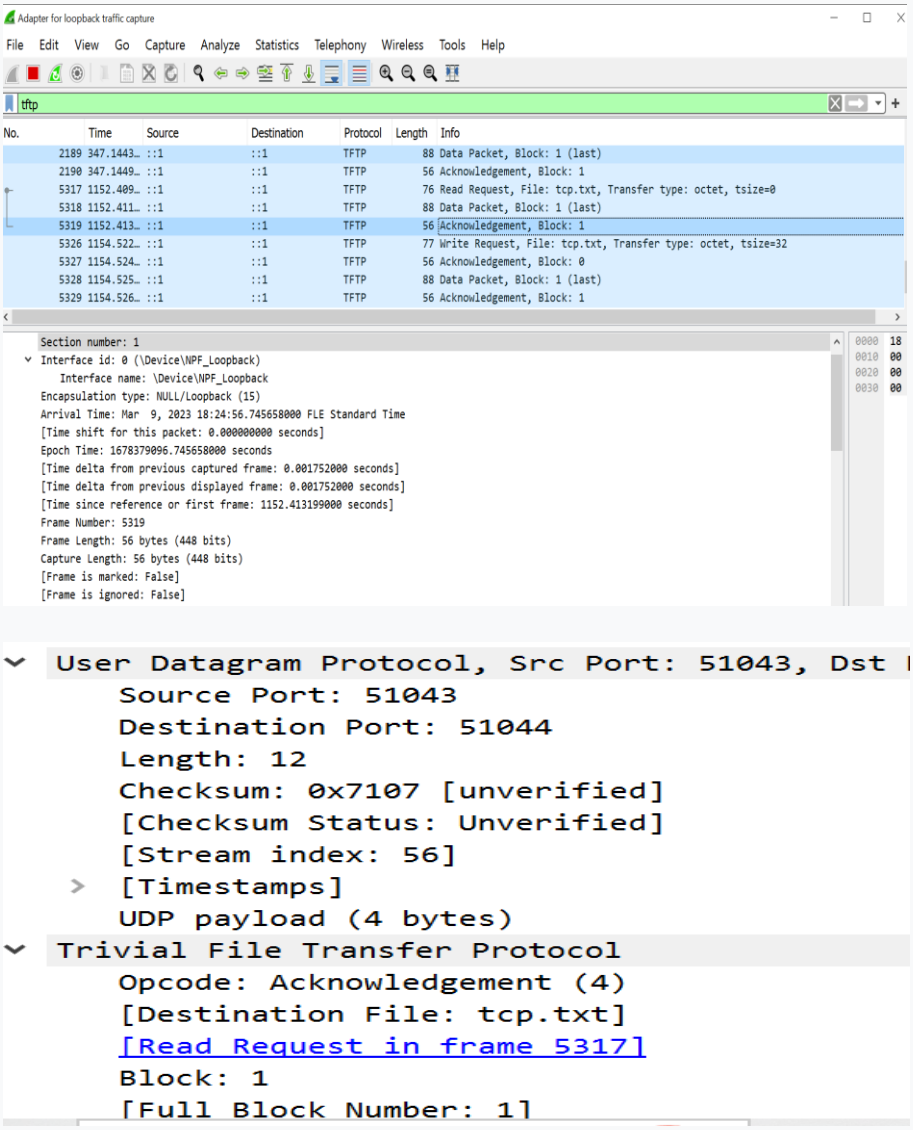


This line gives the following information:

- It is data packet, block 1.
- Source port: 51044:
- Destination port: 51043
- Opcode: data packet (3)

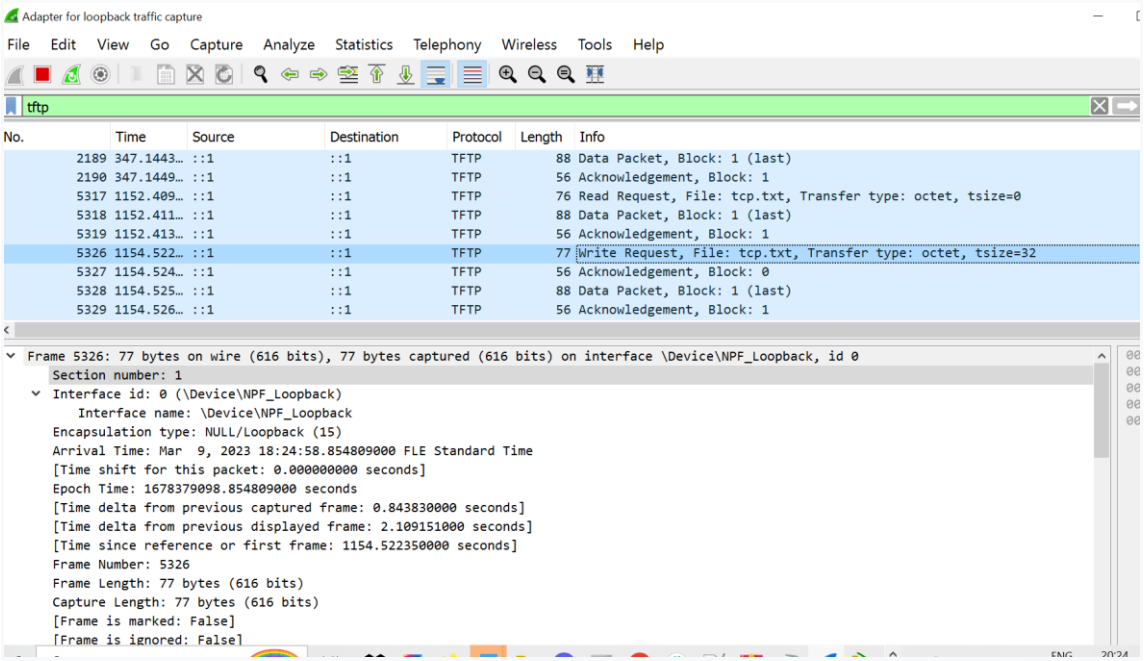
In accordance TFTP protocol data packets have opcode 3, block number and data field. The blocks numbers on data packets begin with one and increase by one for each new block od data. So, we can say that we get correctly.

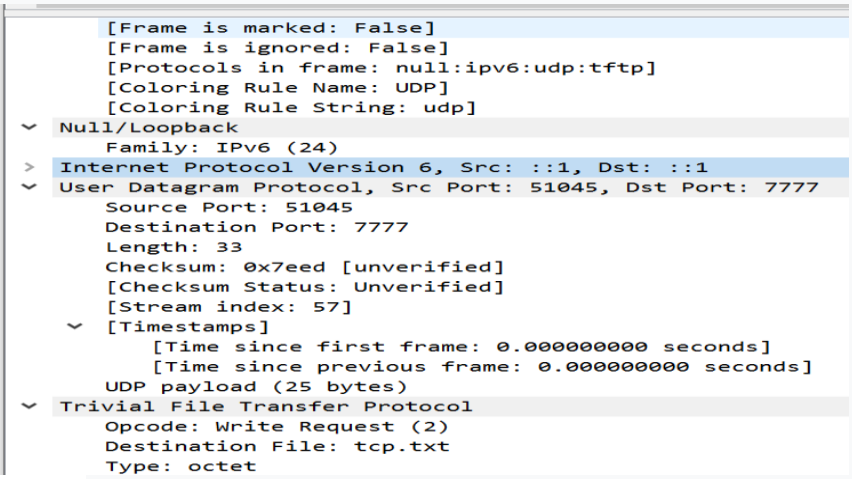
Read request, third line.



Third line is acknowledgment request which gives the following information:
- block 1. In accordance the TFTP protocol the block number in ACK repeats the block number of the data packet being acknowledged.
-source port 51043,
-destination port 51044,
-opcode: acknowledgment 4. In accordance the TFTP protocol the opcode for the ACK packet is 4. So, we got correct information.

Write request has been captured by Wireshark. Four lines belong to write request which are used it to ensure the successful transfer of data.

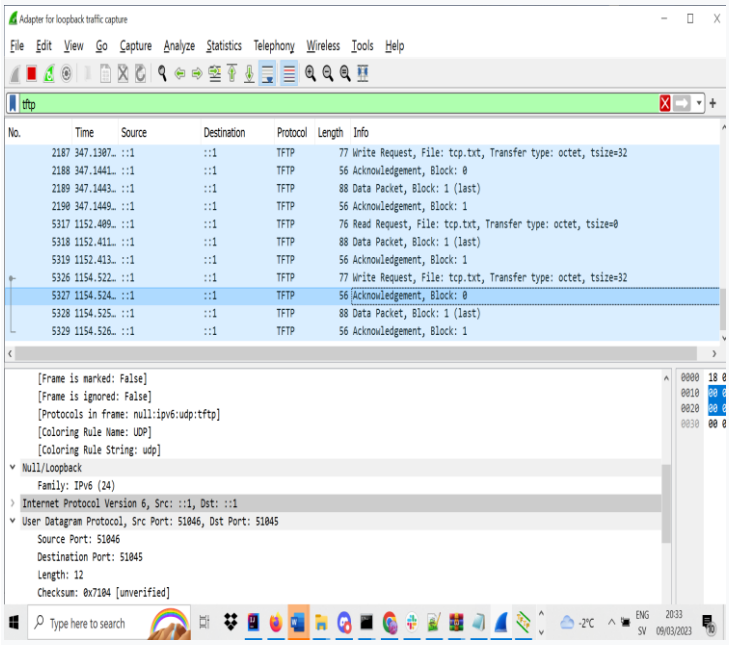




First line gives the following information:

- Name of the file is tcp.txt
- Transfer type: octet. This specifies the type of data being transferred: "octet" which means the data is being transferred as a sequence of bytes.
- Destination port 7777.
- Opcode: write request (2). It is correct in accordance tftp protocol. We got correct information.

Write request, second line.



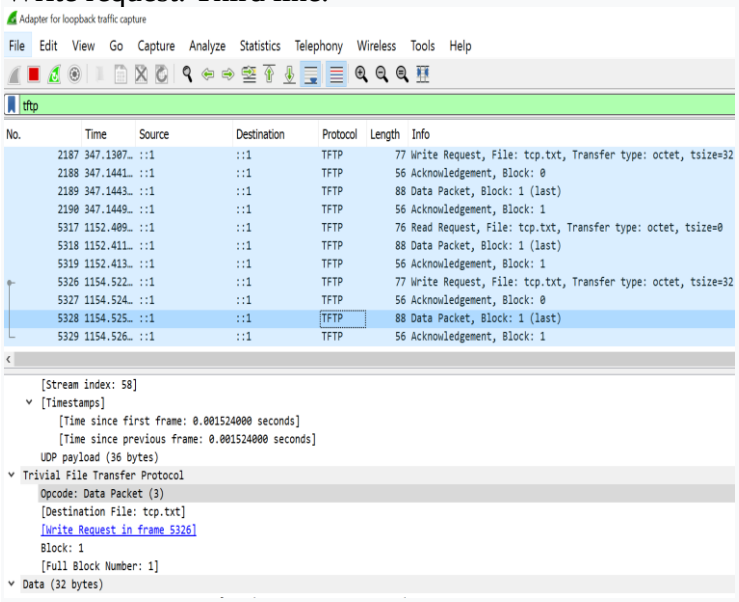
Third line gives the following information:

- It is acknowledgment,
- Block 0. Here we see difference between write and read request.

In above we write that acknowledgment for read request, block 0.

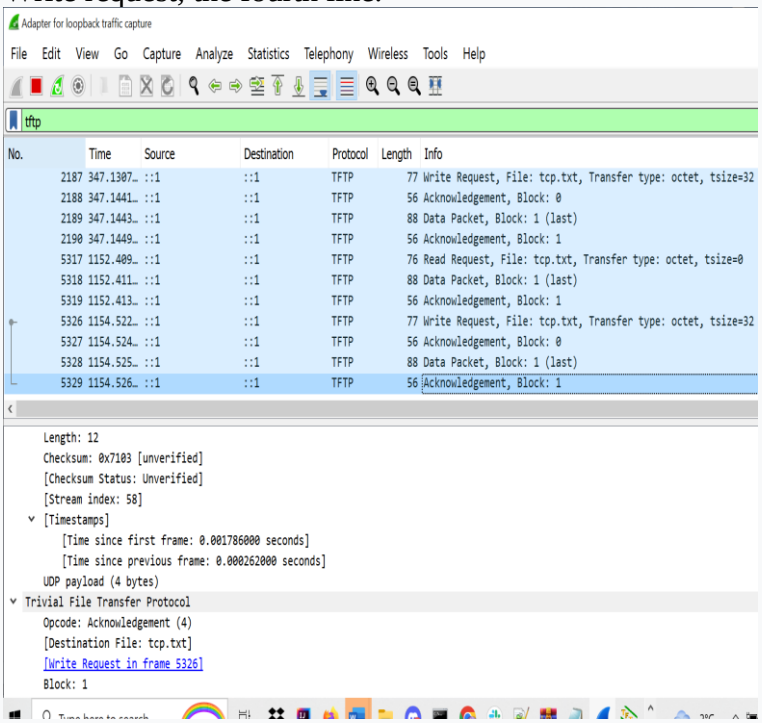
- Opcode ACK is 4.
- Tsize 32. This specifies the total size of the file being transferred. In this example, the file being transferred is 32 bytes in size.

Write request. Third line.



Third line gives the following information. It is data packet, block 1 (last). It means last.

Write request, the fourth line.

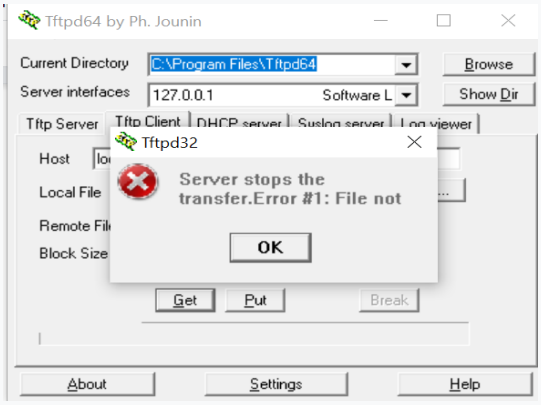


This line gives us the following information:
It is acknowledgment, Block 1, opcode: Ack (4). It is correct in accordance TFTP protocol.

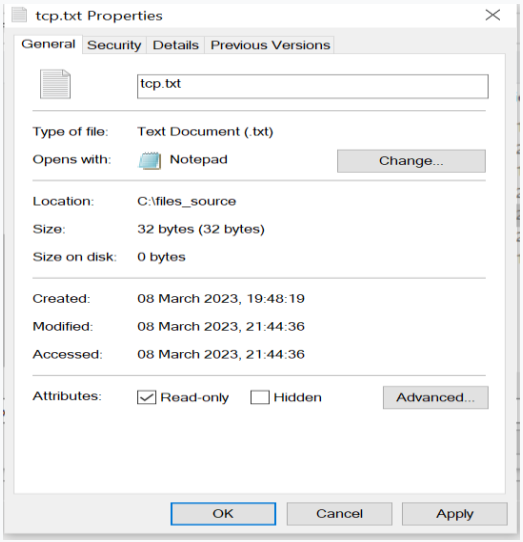
Difference between read and write request is the write request protocol is used to send data from one device to another, while the read request protocol is used to request and receive data from another device, as well acknowledgment for read request, block 0 but wrote request block 1.

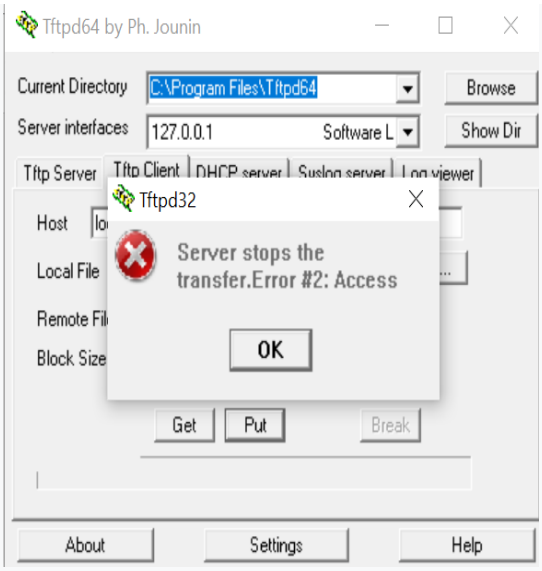
Problem 4.
Discussion 4.1.

Error 1, file cannot be found.

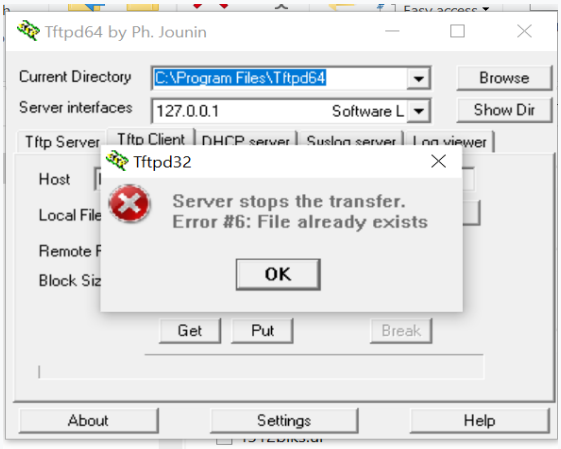


The error number 2, access violation. We have created the file and implemented the rights only for reading.

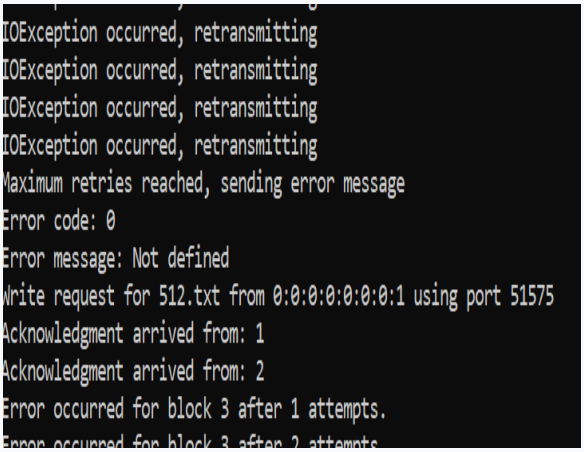




Error 6, file already exists.



Error 0, not defined. In out code we have conditions that if maximum attempts reached (without response), send error and return false.



Summary:

Our group: Katerina Ioannidou and Katarina Simakina.
We worked together, 50 % one person and 50 % other.
Working process was going well. We have used github, meetings for contribution.
The java file is located in the folder is called OUTPUT. One file TFTP Server has all methods In the beginning we have done the code using the program IDE using Maven. However, we decided to remove maven to run this code in different devices. So, this code is runnable in different computers (windows, mac etc).