



**Linnéuniversitetet**

Kalmar Våxjö

Report

# Assignment 3

*1DV701*



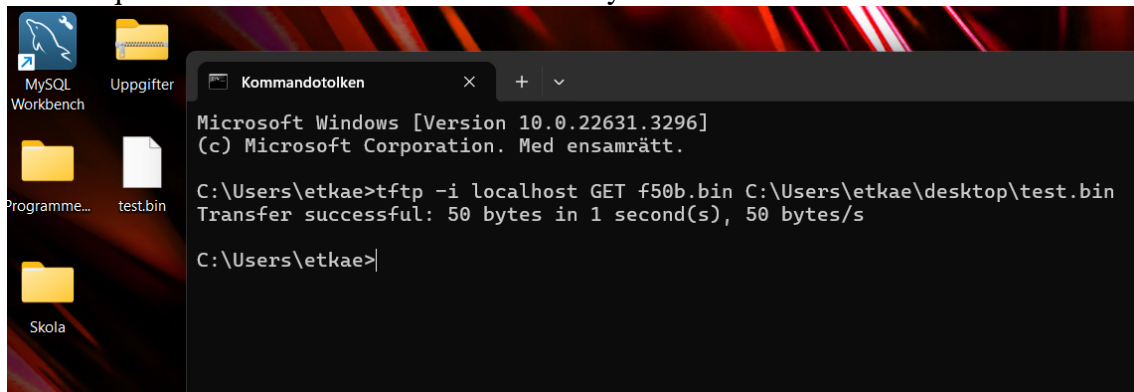
*Author: Etkä Kocak*  
*Semester: Spring 2024*

Contents

<b>1 Problem 1</b>	<b>I</b>
1.1 Discussion	I

# 1 Problem 1

Read request of a file that is shorter than 512 bytes:



```
Kommandotolken
Microsoft Windows [Version 10.0.22631.3296]
(c) Microsoft Corporation. Med ensamrätt.

C:\Users\etkae>tftp -i localhost GET f50b.bin C:\Users\etkae\desktop\test.bin
Transfer successful: 50 bytes in 1 second(s), 50 bytes/s

C:\Users\etkae>
```

```
Listening at port 69 for new requests
Read request for f50b.bin from 0:0:0:0:0:0:0:1 using port 63485
```

Write request of a file that is shorter than 512 bytes:

```
C:\Users\etkae>tftp -i localhost PUT C:\Users\etkae\desktop\test.bin f50b.bin
Transfer successful: 50 bytes in 1 second(s), 50 bytes/s

C:\Users\etkae>
```

```
Listening at port 69 for new requests
Read request for f50b.bin from 0:0:0:0:0:0:0:1 using port 63485
Write request for f50b.bin from 0:0:0:0:0:0:0:1 using port 52160
```

Test results of the TFTP Server:

```
PS C:\Users\etkae\Desktop\Skola\Datornät\assignment3> python -m pytest
===== test session starts =====
platform win32 -- Python 3.11.2, pytest-8.1.1, pluggy-1.4.0
rootdir: C:\Users\etkae\Desktop\Skola\Datornät\assignment3
collected 14 items

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

----- 14 passed in 34.44s -----
PS C:\Users\etkae\Desktop\Skola\Datornät\assignment3>
```

## 1.1 Discussion

### Why both socket and sendSocket is used?

The TFTPServer uses both "socket" and a "sendSocket" to manage its network communication, which is necessary for handling TFTP operations correctly.

The socket is created to listen for incoming TFTP requests from clients. This socket stays open and listens in a loop for any client that wants to launch a read (RRQ) or write (WRQ) request.

For each request from a client, the server spawns a new thread to handle it. Inside this thread, a sendSocket is created for the handling of the current request of the client. This allows multiple requests to be processed at the same time, as each request gets its dedicated sendSocket.

### **How was the problem solved and the server developed?**

The TFTPServer class implements a basic TFTP server with the following methods that work together:

**main:** The entry point that initializes the server.

**start:** Sets up a socket to listen for incoming TFTP requests on the default TFTP port 69. It loops indefinitely, receiving client requests and spawning new threads to handle them.

**receiveFrom:** Receives a datagram packet from a client. It's used to get the initial request from the client.

**ParseRQ:** Parses the read or write request from the client, extracting the requested filename and the operation code (RRQ or WRQ).

**HandleRQ:** Depending on the request type (read or write), it either sends the requested file to the client or receives a file from the client. It calls different methods based on the required operation.

**send\_ACK:** Sends an ACK packet to the client, indicating the reception of a block of data.

**sanitizeFilePath:** Ensures the requested file path is secure and doesn't allow path traversal to unauthorized directories.

**send\_DATA\_receive\_ACK:** In case of a read request from the client, it reads the requested file and sends it in blocks to the client, waiting for ACKs for each block sent.

**receive\_DATA\_send\_ACK:** For write requests, it receives data blocks from the client, writes them to the specified file, and sends ACKs for each received block.

**send\_ERR:** Sends an error packet to the client, indicating issues like file not found, access violation, or other errors as per the TFTP specification.