FILE TRANSFER APPLICATION
ELEN4017: Network Fundamentals

Kishan Y Narotam (717 931) & Matthew van Rooyen (706 692)

School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa

Abstract: The purpose of this document is to provide an easy-to-use template/style sheet to enable authors to prepare papers in the correct format and style for the final year laboratory project. This document may be downloaded from the School of Electrical and Information Engineering web site and can be used as a template. To ensure conformity of appearance it is essential that these instructions are followed. The abstract should be limited to 50-200 words, which should concisely summarise the paper.

Key words: Four to six key words in alphabetical order, separated by commas.

1. INTRODUCTION

2. SYSTEM OVERVIEW

2.1 FTP Server

The FTP server was designed according to the RFC959 standards in order to ensure connection with a standard client is compatible. This allows a number of users to connect to the server through various FTP clients.

The server has been implemented using a number of FTP commands in accordance with RFC959 protocols. In order to facilitate the the file transfer process or processes required by the user FTP process, the server FTP implementation consists of two key components. The first of which is the server protocol interpreter (server-PI). This performs the task of "listening" on a given port number and establishes a controlled connection between the user and the server. This connection first requires authentication from the user to ensure that the communication between the server and the user is legitamate and remains secure. Once this has been achieved, FTP commands can be sent by the user and the response of the server can be sent. This model aims to govern the way in which the DTP layer is operated.

The server data transfer process (DTP) establishes a communication channel with the user through which data can be transferred. This is achieved by creating a channel with the "listening" data port. This then allows the transfer and storage of various file types and instructions to take place according to the commands received by the server PI. The DTP can exist in two states: namely passive and active. Passive state simply listens and for a connection on the server's data port while active state inititates a connection between the data port and the user.

Through this model of an FTP server, a number of FTP commands can be handled in order to obtain functionality and compatibility with the client FTP service. The commands implemented by the FTP server are subdivided into three categories each focusing on the implementation of a specific component of the client-server communication process. The first of these commands are known as **Access Control Commands** which allow the server to authenticate the user by requesting the specification of access control identifiers. Through these commands, user name and password information can be obtained as well as management of the directories established for that particular user. This is followed by **Transfer Parameter Commands** providing the framework through which data will be transferred. The commands involve specifying the port, file structure, mode of transfer and type of file being transferred as well as controlling the state of operation of the server DTP. Finally the **FTP Service Commands** provide the client with

The FTP server was hosted and runs locally using the users local IP address. A server was also hosted using *FileZilla*. The server also implemented multithreading in order to facilitate multiple client connections simultaneously. This allows for concurrent data connections allowing multiple upload and download of files as.

2.2 FTP Client

3. COMMANDS AND REPLIES

FTP replies to client commands are structured in such a way that the all file transfer requests and actions are achieved in a synchronous fashion. The replies also allow the state of the server to be known to the user at all times in order to determine the relative success of each of the requested actions. The details of each reply are made explicit to the user using a 3 digit code followed by text indicating the nature of the reply message. Each digit is is of special significance in the reply with the first digit indicating a successful, unsuccessful or incomplete response. The second digit provides indication as to the type of error that occurred with the 3rd digit providing extra clarification as to the nature of the response and what action should be taken next.

The state of the reply codes are grouped according to the first digit in the following way:

1yx	Positive Preliminary Reply
2yx	Positive Completion Reply
3yx	Positive Intermediate reply
4yx	Transient Negative Completion reply
5yx	Permanent Negative Completion reply

The function groups which the reply codes relate to are encoded in the second digit as follows:

x0z	Syntax
x1z	Information
x2z	Connections
x3z	Authentication and Accounting
x4z	Unspecified as yet
x5z	File System

A number of varying reply codes have been implemented according to the reply code structures given. All implemented messages and the reply codes are given in *Table* **??**. The table also contains a brief description of each reply code.

4. IMPLEMENTATION

4.1 Server Implementation

The FTP server was hosted locally using the user's IP address to allow FTP clients to connect to the server across a range of networks. In order to establish FTP client connections, the server makes use of a `serverListener()` function configured to "listen" across a number of different sockets. In order to allow for multiple connections from a range of clients to be handled simultaneously, each new connection was bound to a thread using the `bind` function. Each thread was initialized in passive mode and is responsible for the data transmission of a given client. A message received across the connected FTP client is decoded and authorized before being executed. Depending on the result of the command, a reply code is then encoded and sent back to the client. The reply codes provide the user with follow on information to each command being passed according to the code structure given in Section 3..

Authentication: A number of commands were implemented in order to control access and configuration of the file structure for each individual client. The server contains a user management system achieved through the authentication of each FTP client. (USER()) and (PASS()) allows the server to accept unique user name and passwords in order to ascertain what file repository's the user has access to. This allows each user to instantiate a file structure that is unique to their credentials. The (QUIT) command provided the client with a method of ending the current session with the wholsale changes made during the period being maintained.

Server Configuration: A range of configuration commands were used in order to facilitate client-server interaction. The server utilities the `os module` to control file and directory operations, providing cross-platform integration. `PORT` command specifies the data port number used by the client in order to create a data connection. `PASV` command was used to instatntiate passive mode with the server listening for the client connections in order to enable the data transfer between the two entities.

File Configuration: `STRU` command allowed the client to customize the file structure within the specified repository. Due to compatibility issues when hosting the server locally, the default file structure was set to page structure (P). The server catered for a number of commands allowing the client to traverse the specified file structure. The `PWD` and `LIST` commands allowed the user to display comprehensive information pertaining to both the remote and local repositories. `CWD` allowed the client to change the path with which the files were being accessed. File transfer types were indicated using the `TYPE` command allowing the user to specify ASCII (TYPE A) or IMAGE (TYPE I) the type of file which needed to be transfered. `STOR` and `RETR` facilitated the uploading and downloading of files respectively. This was achieved through the streaming of data in specified bytes during the transmission of the given file. A ping test was created using the `NOOP` command in order to determine the rate at which a file was being transmitted. Finally, the `HELP` command was added to the server capabilities allowing the user to access the list of commands available to them.

4.2 Client Implementation

5. DIVISION OF WORK

6. RESULTS

6.1 Implemented server and client

6.2 FileZilla server and client

6.3 Free online server and client

7. CRITICAL ANALYSIS

7.1 Successes

7.2 Limitations and Problems

MATT talkl about server issues and limitations

7.3 Future Development

MATT, say what could be improved

8. CODE STRUCTURE AND PREREQUISITES

8.1 Code Structure

8.2 System Prerequisites

MATT, just type the code needed to run your server

9. CONCLUSION

A conclusion may review the main points of the paper, but do not replicate the abstract as the conclusion.

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.