**FTP Protocol Implementation**

**Team 9**

Yan Zhichao 1001357612

Arjun Lakshmikanth 1001192326

Chanakya Harish 1001267138

**Introduction:**

**FTP Protocol:**

The objectives of FTP are 1) to promote sharing of files (computer programs and/or data), 2) to encourage indirect or implicit (via programs) use of remote computers, 3) to shield a user from variations in file storage systems among hosts, and 4) to transfer data reliably and efficiently. FTP, though usable directly by a user at a terminal, is designed mainly for use by programs.

The attempt in this specification is to satisfy the diverse needs of users of maxi-hosts, mini-hosts, personal workstations, and TACs, with a simple, and easily implemented protocol design.

**Prerequisites for FTP Protocol:**

Transmission Control Protocol and Telnet protocol are basic requirements for understanding or using FTP Protocol.

**High level Design:**

**System Architecture:**

As shown in Figure 1, FTP Protocol has defined two parts, Server-FTP (server) and User-FTP (client), while the former has two main components, namely Server PI (protocol interpreter) and Server DPT (data transfer process); and the latter has three main components, namely User Interface, User PI and User DTP.



**Figure 1. Main Components in FTP**

From the protocol, we at least need to implement these five main components, namely, Server\_PI, Sever\_DTP, User\_Interface, User\_PI and User\_DTP. We will briefly describe the function of each component.

Server\_PI and User\_PI are designed to handle the FTP protocol logic, specifically to take the corresponding actions upon receiving a FTP command and reply the corresponding numbers. Server\_DTP and User\_DTP are designed to transfer the file stream between Server-FTP and User-FTP. It can be implemented by constructing a new socket connection to transfer the corresponding files. Once it finishes the data transferring, we will delete the socket connection. User\_Interface is designed to receive the input commands from users, and translate it as the corresponding FTP commands to the User\_PI logic.

**Implementation Tasks:**

1. Minimum function set

In order to make FTP workable without needless error messages, the following minimum implementation is required for all servers:

TYPE - ASCII Non-print

MODE - Stream

STRUCTURE - File

COMMANDS - USER, QUIT, PORT,

TYPE, MODE, STRU,

for the default values

RETR, STOR,

NOOP.

The default values for transfer parameters are:

TYPE - ASCII Non-print

MODE - Stream

STRU - File

All hosts must accept the above as the standard defaults.

2. Implementation Plan

We plan to finish this minimum standard first and would like to extend it to incorporate more functions.

3. Commands Analysis

Basic commands:

USER: send new user information

QUIT: terminate ftp session and exit

PORT: send the data transfer port

TYPE: set file transfer type (ASCII)

MODE: set file transfer mode (Stream)

STRU: set file transfer structure (File)

RETR: retrieve a file

STOR: accept the data and to store the data as a file at the server site

NOOP: no operation (dummy package, mostly on keeping alive)

Basic procedure:

Client initiates a PI connection to Server

Server asks the user name and password to authorization

Client sets data transfer port

Server responses the setting request

Client sets data transfer type

Server responses the setting request

Client sets data transfer mode

Server responses the setting request

Client sets data transfer structure

Server responses the setting request

Client sends retrieve and store request to server

……

Once receive a command, it usually replies a code to indicate the statues, the reply codes are:

110 Restart marker reply.

In this case, the text is exact and not left to the

particular implementation; it must read:

MARK yyyy = mmmm

Where yyyy is User-process data stream marker, and mmmm

server's equivalent marker (note the spaces between markers

and "=").

120 Service ready in nnn minutes.

125 Data connection already open; transfer starting.

150 File status okay; about to open data connection.

200 Command okay.

202 Command not implemented, superfluous at this site.

211 System status, or system help reply.

212 Directory status.

213 File status.

214 Help message.

On how to use the server or the meaning of a particular

non-standard command. This reply is useful only to the

human user.

215 NAME system type.

Where NAME is an official system name from the list in the

Assigned Numbers document.

220 Service ready for new user.

221 Service closing control connection.

Logged out if appropriate.

225 Data connection open; no transfer in progress.

226 Closing data connection.

Requested file action successful (for example, file

transfer or file abort).

227 Entering Passive Mode (h1,h2,h3,h4,p1,p2).

230 User logged in, proceed.

250 Requested file action okay, completed.

257 "PATHNAME" created.

331 User name okay, need password.

332 Need account for login.

350 Requested file action pending further information.

421 Service not available, closing control connection.

This may be a reply to any command if the service knows it

must shut down.

425 Can't open data connection.

426 Connection closed; transfer aborted.

450 Requested file action not taken.

File unavailable (e.g., file busy).

451 Requested action aborted: local error in processing.

452 Requested action not taken.

Insufficient storage space in system.

500 Syntax error, command unrecognized.

This may include errors such as command line too long.

501 Syntax error in parameters or arguments.

502 Command not implemented.

503 Bad sequence of commands.

504 Command not implemented for that parameter.

530 Not logged in.

532 Need account for storing files.

550 Requested action not taken.

File unavailable (e.g., file not found, no access).

551 Requested action aborted: page type unknown.

552 Requested file action aborted.

Exceeded storage allocation (for current directory or

dataset).

553 Requested action not taken.

File name not allowed.

Command-Reply Sequences

Connection Establishment

120

220

220

421

Login

USER

230

530

500, 501, 421

331, 332

PASS

230

202

530

500, 501, 503, 421

332

Logout

QUIT

221

500

Transfer parameters

PORT

200

500, 501, 421, 530

PASV

227

500, 501, 502, 421, 530

MODE

200

500, 501, 504, 421, 530

TYPE

200

500, 501, 504, 421, 530

STRU

200

500, 501, 504, 421, 530

File action commands

STOR

125, 150

(110)

226, 250

425, 426, 451, 551, 552

532, 450, 452, 553

500, 501, 421, 530

RETR

125, 150

(110)

226, 250

425, 426, 451

450, 550

500, 501, 421, 530

Miscellaneous commands

NOOP

200

500 421

**Minimum Requirements Implementation:**

In order to make FTP workable without needless error messages, the following minimum implementation is required for all servers:

TYPE - ASCII Non-print

MODE - Stream

STRUCTURE - File, Record

COMMANDS - USER, QUIT, PORT,

TYPE, MODE, STRU,

for the default values

RETR, STOR,

NOOP.

The default values for transfer parameters are:

TYPE - ASCII Non-print

MODE - Stream

STRU - File

All hosts must accept the above as the standard defaults.

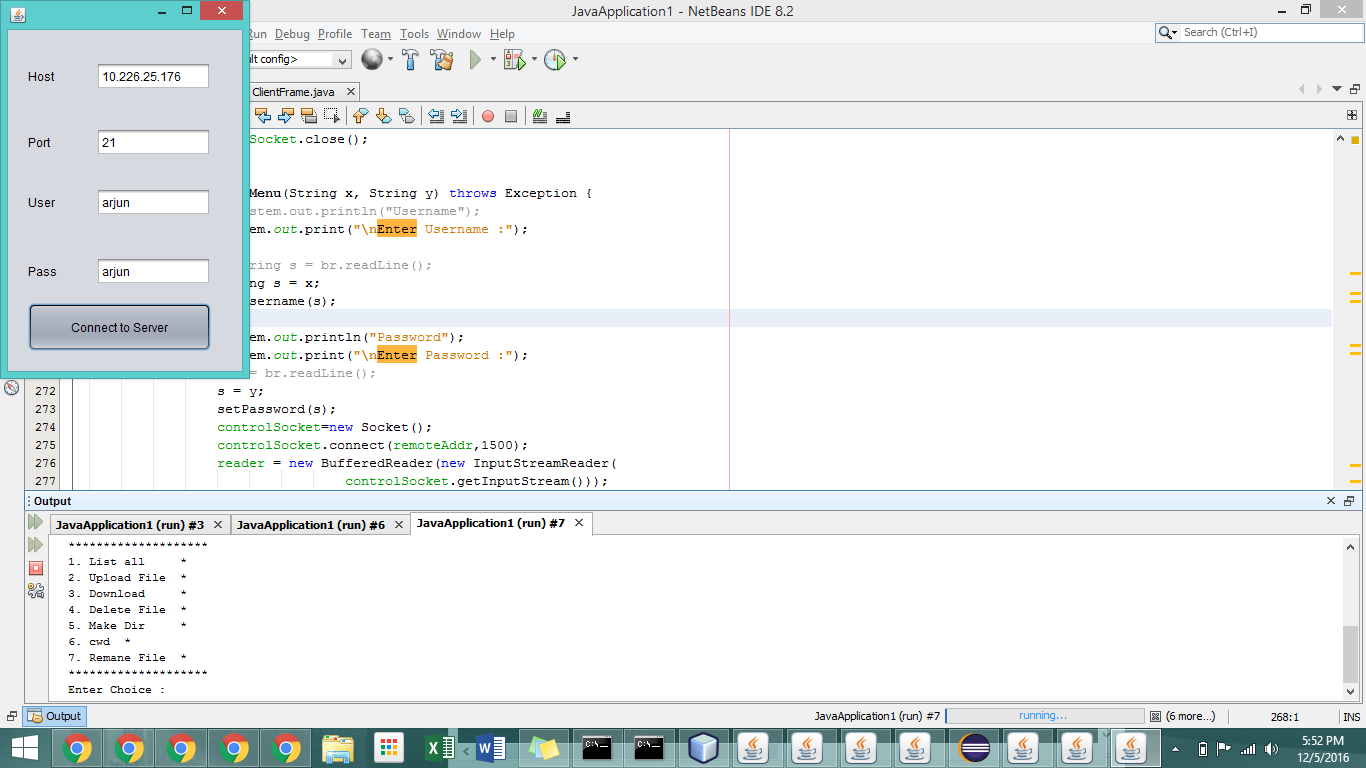
**Features Implemented:**

|  |  |  |
| --- | --- | --- |
| **Mandatory Features**  USER  QUIT  PORT  TYPE  MODE  RETR  STOR  NOOP | **Implemented Features**  USER  QUIT  PORT  TYPE  MODE  RETR  STOR  NOOP | **Extra features done**  LIST  MKDIR  DELETE FILE  CHANGE WORKING DIR  RENAME FILE  USER INTERFACE  HELP |

**Execution Steps:**

For the User Interface to work the FTP client program has another class named Client Frame which will provide the interface for the User Interface part.

Enclosed are two different folders of which one includes the classes with just command prompt execution and the other with User Interface execution.

****

The above screenshot is what it looks like to execute the client frame interface which gives the User Interface as specified above.