Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 5, May 2015, pg.777 – 785

RESEARCH ARTICLE

FTP UPLOAD & DOWNLOAD WITH DOWNLOAD RESUMPTION USING BYTE-ROLLBACK

Ritika Kamboj, Mauli Joshi
Maharishi Ved Vyas Eng. College, Jagadhri, Yamunanagar, Kurukshetra University, Haryana
Ritikakamboj051@gmail.com

I INTRODUCTION

FTP:- The File Transfer Protocol (FTP) is a standard network protocol used to transfer computer files from one host to another host over a TCP-based network, such as the Internet.

FTP is built on client-server architecture and uses separate control and data connections between the client and the server. FTP users may authenticate themselves using a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS). SSH File Transfer Protocol (SFTP) is sometimes also used instead, but is technologically different.

The first FTP client applications were command-line applications developed before operating systems had graphical user interfaces, and are still shipped with most Windows, UNIX and Linux operating systems. Many FTP clients and automation utilities have since been developed for desktops, servers, mobile devices, and hardware, and FTP has been incorporated into productivity applications, such as Web page editors.

FTP may run in active or passive mode, which determines how the data connection is established. In both cases, the client creates a TCP control connection from a random, usually an unprivileged, port N to the FTP server command port 21. In active mode, the client starts listening for incoming data connections from the server on port M. It sends the FTP command PORT M to inform the server on which port it is listening. By default, M=N+1. The server then initiates a data channel to the client from its port 20, the FTP server data port. In situations where the client is behind a firewall and is unable to accept incoming TCP connections, passive mode may be used. In this mode, the client uses the control connection to send a PASV command to the server and then receives a server IP address and server port number from the server, which the client then uses to open a data connection from an arbitrary client port to the server IP address and server port number received. Both modes were updated in September 1998 to support IPv6. Further changes were introduced to the passive mode at that time, updating it to extended passive mode.

The server responds over the control connection with three-digit status codes in ASCII with an optional text message. For example "200" (or "200 OK") means that the last command was successful. The numbers represent

the code for the response and the optional text represents a human-readable explanation or request (e.g. <Need account for storing file>). An ongoing transfer of file data over the data connection can be aborted using an interrupt message sent over the control connection.

While transferring data over the network, four data representations can be used:

1. **ASCII mode:** Used for text. Data is converted, if needed, from the sending host's character representation to "8-bit ASCII" before transmission, and (again, if necessary) to the receiving host's character representation. As a consequence, this mode is inappropriate for files that contain data other than plain text.

2. **Image mode (commonly called Binary mode):** The sending machine sends each file byte for byte, and the recipient stores the byte stream as it receives it. (Image mode support has been recommended for all implementations of FTP).

3. **EBCDIC mode:** Used for plain text between hosts using the EBCDIC character set.

4. **Local mode:** Allows two computers with identical setups to send data in a proprietary format without the need to convert it to ASCII.

For text files, different format control and record structure options are provided. These features were designed to facilitate files containing Telnet or ASA.

Data transfer can be done in any of three modes:

1. **Stream mode:** Data is sent as a continuous stream, relieving FTP from doing any processing. Rather, all processing is left up to TCP. No End-of-file indicator is needed, unless the data is divided into records.

2. **Block mode:** FTP breaks the data into several blocks (block header, byte count, and data field) and then passes it on to TCP.

3. **Compressed mode:** Data is compressed using a single algorithm (usually run-length encoding).

One of the most common activities on a network is accessing a file located on a remote location normally called a file server. Very often the file (of any format) is downloaded on the local system from that remote location using some network protocol like FTP. A very common problem during downloading is interruption of download due to link failure. A common complaint by FTP clients is failure to resume an interrupted download. Imagine the frustration when a download is interrupted after significant completion and the file has to be re-downloaded (i.e. downloaded from start).

Our application implements FTP which is used to exchange files over client-server architecture. This application utilizes data connections between the client and server applications and is capable of resuming interrupted downloads before rollback of last few bytes. The client makes a connection to the server using TCP port 21 from a port N and then establishes a data connection on port N+1.

In this application, we have planned to develop the FTP client whereas the server would be a 3rd party application which we would be configuring in our system. During the configuration, we will create a shared directory on the server from/to where, the files would be uploaded/downloaded. The client application will obtain the inputs regarding ip address and directory location from the user to effectively establish a connection to the server. The stoppage of file transfer in progress will be done through manually stopping the server. The resumption of download will be done from the point where the download was interrupted after rolling back a few bytes of data previously transferred.

When the Program gets executed, a GUI appears for the end user. Here the user provides the IP address of the FTP server, the location on the server and the location on client side. Then the file name along with download or upload option is provided by the user. All this things are taken as a string in the main program and ftpconnect() is called. In ftpconnect(), connection is made with the server using Connection pipe provided by JAVA. Once the

connection is made with the server on the default port, user name and password is provided to the server for the authenticated login. If authentication is successful, a passive mode is activated for file upload and download.

In passive mode FTP the client initiates both connections to the server, solving the problem of firewalls filtering the incoming data port connection to the client from the server. When opening an FTP connection, the client opens two random unprivileged ports locally (N > 1023 and N+1). The first port contacts the server on port 21, but instead of then issuing a PORT command and allowing the server to connect back to its data port, the client will issue the PASV command. The result of this is that the server then opens a random unprivileged port (P > 1023) and sends the PORT P command back to the client. The client then initiates the connection from port N+1 to port P on the server to transfer data.

From the server-side firewall's standpoint, to support passive mode FTP the following communication channels need to be opened:

FTP server's port 21 from anywhere (Client initiates connection)
FTP server's port 21 to ports > 1023 (Server responds to client's control port)
FTP server's ports > 1023 from anywhere (Client initiates data connection to random port specified by server)
FTP server's ports > 1023 to remote ports > 1023 (Server sends ACKs (and data) to client's data port)

After all this procedure, Directory provided by the user is searched on the server and that location is made the current location on the server. In that directory file provided is searched one by one, when that file is encountered a file with same name is made on the current client side location. Now, the file is read byte by byte and transferred to the client side through the connection pipe made earlier. In between a thread has been created which count the number of bytes transferred every 0.5 second. And convert that in to percentage transfer. In between if connection failure happens, and user tries to reattempt the procedure, then a check has been created so that if file exists on the client side then we resume the further download from size of file at client side (– 500 bytes).

Same thing also happens while uploading the file. The main concept that comes into the picture is: PIPE created between Client and server through which the data get transferred. To exist the firewall on client side, its client that starts all the connection (passive mode) timers to check on the transferred data per unit time under consideration. This is maintained through the JAVA thread programming where a thread is involved in monitoring the bytes transferred through the pipe, apart from the main thread.

This idea is developed using Java. It connects ‘n’ number of laptops to each other in a wired or wireless fashion. One becomes the FTP server which is 3rd party software (Filezilla FTP Server) that we install and configure on one system. On others, we install the FTP client application which we have coded by applying Java concepts like, Socket/Port Programming, Multithreading, I/O Handling, etc. Using the client application, data files in any format could be uploaded and downloaded to and from the FTP server. During download, we deliberately disconnect the network in order to demonstrate or simulate the situation of abrupt network link failure. During this event, we prove that the ongoing download of the last file was not completely hampered. Instead, if we download the same file again using our application, the process resumes from exactly the same point from where it got disrupted. We create an intranet in order to illustrate this concept. It is proved with the fact that, for ex: we choose to download a file of size 2 GB from the server. The total download time taken is, let’s say 1 minute. Now, we delete this file from the client and download it again from the FTP server. This time, at 50% download progress, we take out the LAN wire from laptop’s Ethernet port (in case of a wired network). Now, if we download the same file again, the entire process completes in 30 seconds proving the fact that it did not take 1 minute to start the entire process all over. Instead, it resumed the process after 50% to download the remaining 50%. And our application is able to resume the download of data in ANY format unlike other applications which distinguish the type of files that can be resumed or not resumed.

II ABSTRACT

Our application implements a network protocol which is used to exchange files over client-server architecture. This application utilizes data connections between the client and server applications.

The client makes a connection to the server using TCP port 21. This connection, called the control connection, remains open for the duration of the session, with a second connection on port 20 opened as required to transfer file data. The server responds on the control connection and in our application, the stoppage of file transfer in progress will be done through manually stopping the server.

In this application, we have planned to develop the FTP client whereas the server would be a 3rd party application which we would be configuring in our system. During the configuration, we will create a shared directory on the server from/to where, the files would be uploaded/downloaded.

III DESIGN

A. Architectural Design

Under architectural design, after defining the whole system into a set of objectives & further subdividing them into functions, we defined the basic dependency & communication between them.

This means that all the prime functions, their required inputs, expected output/behaviour & interdependency between other functions were clearly defined. The corresponding interfaces for the user for each function were designed to ensure user-friendliness.

We actually addressed the system-level problems here and made a conscious effort to build a robust design which can result in an effective communication within itself and with the system in terms of raw data or processed information.

All the primary database design for data storage was also done in this phase.

B. Detailed Design

In this phase, we further subdivided every function into a set of modules & defined required inputs & expected behaviour for each of them. All the minute correlations, interdependencies, communication between the modules were clearly defined. The source, usage & processing of data for every module was carefully done. The database design was also normalized at this stage to ensure that the data is efficiently stored & retrieved.

Detailed design helped us to exactly concretize every problem into inputs & outputs and visualize them in terms of their communication with each other. We focused on interdependency & interoperability between the broken modules here.

It was this design phase where the factors like user-friendliness ease of use, scalability and self-explanation of interfaces & outputs were actually realized. For all the modules, the placement of controls,

passing of information, communication of different interfaces, user messages, data transfer to databases was defined.

IV ANALYSIS

The analysis model must achieve three primary objectives:

1. To describe the requirements of the customer.
2. To establish a basis for the creation of a software design.
3. To define a set of requirements that can be validated once software is built.

An Overview

The system analysis phase is considered to be one of the most important phases in the system development life cycle. It is immensely important that the software developer make through study of the existing system.

Thorough study of the system is made and need i.e. features that are critical to system success and users wants i.e. features that would be good but not essential are brought out. The study will enable the developer to know the intricacies of the existing system.

Requirements analysis is done in order to understand the problem which the S/W system is to solve e.g., the problem could be automating the existing manual system or developing a completely new automated system or a combination of the two. For large systems having a large number of features and the need to perform many different tasks, understanding the requirement of the system is a major task. The emphasis in requirement analysis is on identifying what is needed from the system.

The analysis of the system was done rigorously because this is such a phase where all the loopholes had to be discovered keeping company's objectives & challenges in mind. We performed the analysis in 02 parts i.e. Feasibility Analysis & Requirements Analysis.

Feasibility Analysis

1) We studied the whole system & its objectives. Calculated the total time & resources incurred on every function being done manually.

2) Bifurcated the complete system into a list of functions & the users who operate on them.

3) Further subdivided all the functions into a list or source of requirements/inputs & clearly defined the output/expectation from each function.

4) The interaction, communication & dependency of all the functions between each other were carefully analyzed in terms of sequence & information.

5) The source & flow of the information was determined & how would it be processed & used was considered.

6) Finally, we visualized the complete system with automated functions & compared the total time & resources being incurred to check the feasibility & see whether it is fulfilling all the necessary objectives.

Requirement Analysis

1) This was a subset of feasibility analysis in which we defined a set of objectives for the complete system after thoroughly analyzing it.

2) All the objectives were further subdivided into a set of function(s).

3) The input(s) required by each function & the expected output(s)/behaviour was/were clearly defined.

4) The source of information/input to every function was determined & its corresponding processing, usage & storage were also taken into account.

5) After this the interdependency & communication was finalized.

V Figures

FIG. 1

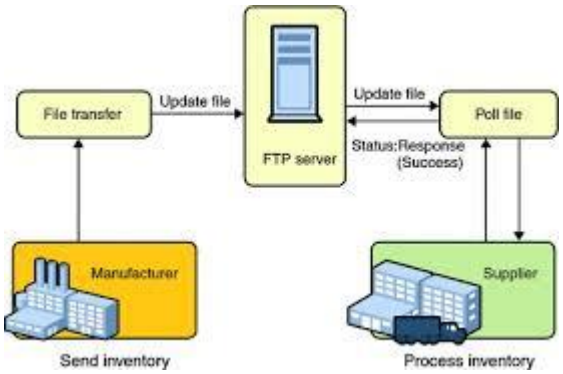

FIG. II

Figures Used:

Fig 1: http://www.google.co.in/imgres?imgurl=http://mrgoogleglass.com/wp-content/uploads/2014/09/what-is-FTP.jpg&imgrefurl=http://mrgoogleglass.com/transferring-files-server-ftp/&h=239&w=554&tbnid=jos_p7qWqgugsM:&zoom=1&docid=jNWe161KPW-DJM&ei=uYVgVZ-vCJGjugSvjYGwDg&tbm=isch&ved=0CEQQMygTMBM

Fig 2: http://www.google.co.in/imgres?imgurl=http://docs.oracle.com/cd/E19182-01/820-6326/images/FTP_InventoryNew.gif&imgrefurl=http://docs.oracle.com/cd/E19182-01/820-6326/fbindftpctut_intro/index.html&h=303&w=462&tbnid=yJUKyEpE23lSYM:&zoom=1&docid=c3nNg0s88ZQFUM&ei=uYVgVZ-vCJGjugSvjYGwDg&tbm=isch&ved=0CDUQMygEMAQ

VI CONCLUSIONS

We can hereby conclude that:

- This application fulfills the desired objective of uploading & downloading files from the FTP server.
- It effectively resumes the download process of files through *byte-rollback* concept which suggests that if the download of any file gets interrupted due to any reason, we always rollback a few bytes & then start (resume) the download process till the last byte of the file (EOF). In this way, the file-state remains intact because we recover all the lost bytes and duplicate bytes get overwritten.
- Through this application, we can upload & download the files of any format & size.
- The download-resumption functionality effectively works for all file sizes and formats unlike other applications which are restricted to certain sizes & formats only.
- For a clear visibility of resumption feature, it is essential that the file-size to be downloaded is greater than 700 MB because the server & client are executing on the standalone system due to which the transfer of bytes is very swift. For small-size files, this functionality would not be visible for obvious reasons.

VII FUTURE SCOPE

Although, we have attempted to include & implement the necessary functionality in the application, it still holds a great scope of improvement.

Few points which are worth consideration for the betterment of the application's functionality are as mentioned below:

- We can further enhance this application to work on different networking protocols in addition to FTP for the purpose of uploading, downloading & download-resumption.

- The efficiency of this application can be improved further through the implementation of network route optimization during the upload, download & resumption process in order to reduce & save time during the process of data packets transmission between clients & server.

- Advanced concepts like detecting a network resource deadlock & collision detection of data packets can be implemented for an accurate & efficient upload, download & resumption process.

- We can enhance this application to work on multi-user environments spread over geographically diverse locations.

VIII ACKNOWLEDGEMENT

I would like to give my heart-felt sincere-most thanks to our guides, teachers and all lab faculty members who always stood as a guiding spirit behind me and gave their invaluable supervision, guidance and support without which this project wouldn't have reached its final destination.

REFERENCES

Teach Yourself Java 1.1 Programming in 24 Hours
by Rogers Cadenhead
- Publisher: Sams.net
- ISBN: 1-575-21270-6
- Publication Date: May, 1997
- Pages: 382

Java Now!
by Kris Jamsa
- Publisher: Jamsa Press
- ISBN: 1-884-13330-4
- Publication Date: July, 1996
- Pages: 350

The Java Programming Language
by Ken Arnold, James Gosling
- Publisher: Addison-Wesley
- ISBN: 0-201-63455-4
- Publication Date: May, 1996
- Pages: 380

Kickass Java Programming
by Tonny Espeset
- Publisher: The Coriolis Group
- ISBN: 1-883-57799-3
- Publication Date: August, 1996
- Pages: 450

Inside the Java Virtual Machine
by Bill Venners
- Publisher: McGraw-Hill
- ISBN: 0-079-13248-0
- Pages: 384
- Publication Date: June, 1998

Inside Java
by Karanjit S. Siyan , James L. Weaver
- Publisher: New Riders Publishing
- ISBN: 1-562-05664-6
- Pages: 928
- Publication Date: January, 1998

Java Programming With Corba
by Andreas Vogel , Keith Duddy
- Publisher: John Wiley & Sons
- ISBN: 0-471-17986-8
- Pages: 352
- Publication Date: March, 1997

Not Just Java
by Peter Van Der Linden
- Publisher: Prentice Hall
- ISBN: 0-138-64638-4
- Publication Date: April, 1997
- Pages: 332

Special Edition Using Enterprise Java
by Jeff Schneider, Rajeev Arora
- Publisher: Que
- ISBN: 0-789-70887-6
- Pages: 496
- Publication Date: June, 1997

Web References:

1. https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0CCoQFjAC&url=http%3A%2F%2Fcdts.genomics.org.cn%2FcustomerSupport%2FHowToDownloadDataFromCDTS.pdf&ei=qkBXVc_lN4W78gXmwILQAg&v6u=https%3A%2F%2Fs-v6exp1-ds.metric.gstatic.com%2Fgen_204%3Fip%3D101.57.202.2%26ts%3D1431781136721951%26auth%3Dtytdye3dxubg4biqr26cabygmcousb7z%26rndm%3D0.7354220028501004&v6s=2&v6t=424423&usg=AFQjCNEMdKOApsPYkTZ0DQMxVj4ayQwG3Q&bvm=bv.93564037,d.dGc

2. https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&cad=rja&uact=8&ved=0CEEQFjAG&url=ftp%3A%2F%2Fftp.ipswitch.com%2Fipswitch%2FWhite_Papers%2Fsecure_ftp.pdf&ei=qkBXVc_lN4W78gXmwILQAg&usg=AFQjCNEOYEaAfDGhCPPDKteY23e1Rmr9sw&bvm=bv.93564037,d.dGc

