Name - Kunal Vohra

Date – 1/28/2020

Class – CS-544 Computer Networks

Protocol Title – File Transfer Protocol

Analysis of File Transfer Protocol (RFC 959)

Definition of Service

FTP is a Client-Server protocol used to transfer files between a remote server and a local client machine. It is one of the oldest protocols and has still been in use till today almost 50 years after it was designed. This paper is an attempt to breakdown the protocol into its very basic units provide an in-depth analysis of how a new age FTP could be implemented keeping in mind all the positives and negatives of how we know it today.

History of File Transfer Protocol

File transfer protocol was first written by Abhay Bhushan in 1971 as RFC 114. It was pitched as a simple protocol which would carry exchange operation of files between a remote machine and another computer which the user had access to. "Remote" meaning that a person would be able to achieve indirect access to the storage tapes of a machine without being physically present at the server location via a command line interface like the TELNET protocol RFC 854.

> **"A principal objective of the protocol is to promote the indirect Use of computers on the network. Therefore, the user or his Program should have a simple and uniform interface to the file systems on the network and be shielded from the variations in file and storage systems of different host computers. This is achieved by the existence of a standard protocol in each host." -RFC 114**

RFC-114 guaranteed that the protocol would take care of the transfer between the 2 machines irrespective of their varying file systems. FTP was in experimentation at MIT when this RFC was published. In fact, it was first implemented over the Network Control Program (NCP) on the ARPANET. This FTP was implemented in a way to handle many datatypes when specified in headings but recommended the standardization of representation in ASCII or binary with inclusion of EBIDIC and many other character types.

It was specified that the protocol was carried in terms of transactions with messages containing a description header 72 bits in size, the data and the filler or non-information bits of variable lengths to indicate end of a transaction as well as providing a gap between messages of the transaction. The descriptor field consisted of a transaction type specification, a data type byte and its extension, length spec

ification of the filler and a 24-bit field specifying the length of the data field. Transactions were divided into 4 types namely: Request, Response, Transfer and Terminate and all commands fit into one of each category.

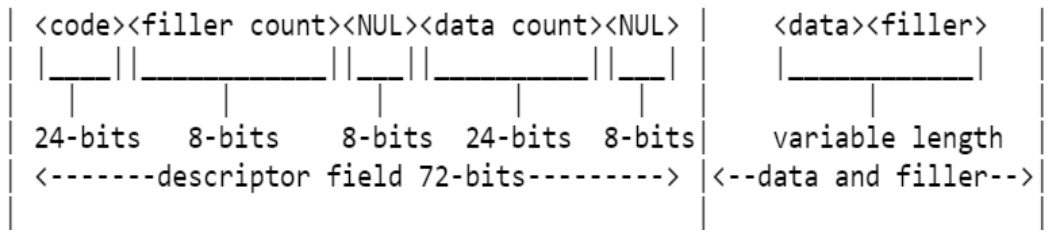

Fig: Syntax of a Transaction (RFC 114)

An “execute” command was added as a functionality to execute a file on the server. The RFC also encouraged that the protocol be extended by anyone who went through it provided they did not change existing structure of the protocol.

Surprising to note that FTP did not even have a name as can be seen by the RFC 114, the heading read “A File Transfer Protocol” suggesting that no name was fixated on this first definition of the protocol.

Initially the protocol was implemented on a duplex channel with transmission of commands and data on the same NCP channel. Although it was tried over dual channels, Abhay preferred the single duplex better due to it being more synchronous just in maintaining a single channel and closing and reopening for file transfers.

RFC 114 was extended by RFC-133 titled “File Transfer and Error Recovery” and authored by R. L. Sunberg of Harvard. The author suggested that the transfers demand a response from the receiver’s end instead of having a Fire and Forget system. Sunberg commented on having an error recover system. If transfers failed and on one end, they could be resumed on request instead of starting over the connection. Also creating an address for individual pieces of a file which could help in cases where large files were to be transferred. And finally, a version number of NCP be added while transacting so that the server and the host were on the same page.

FTP was then formalized with set of important specifications in the subsequent RFCs 141,171 and 172. The proposed points in these RFCs were as follows: -

1) Files not created by FTP should be accessible/usable at users/server
2) Users file list should be available
3) A detailed information dump on failure of a connection
4) Definition of DTP (Data Transfer Protocol): a protocol which would be a part of FTP but without the files system details.
5) Defining of 3 operating modes in DTP:
   - a stream of bits being transmitted,
   - a block transaction mode with a transaction type byte and special codes to specify end of a transaction
   - A new count transaction that consists of a header specifying the length of data in the transaction
6) Error Codes were now used in form of 3-digit numbers instead of ascii codes. These numbers did not have set meanings yet.
7) FTP codes handled at DTP layer.

Since then FTP has been extending to be the current version. The following are a few important updates in the protocol since its implementation: -

RFC 265 – Added functionality of creating file if file and append and create if not existing

RFC 697 – CWD command was implemented

RFC 959 – File Transfer Protocol was standardized

RFC 1579 – Firewall friendly FTP was suggested

RFC 1635 – Using FTP anonymously

RFC 1639 - FOOBAR

RFC 1738 – URL

RFC 2228 – FTP Security Extensions proposed

RFC 2389 –Feature negotiation mechanism

RFC 2428 –Extensions for IPv6, NAT, Extended passive mode implementation

RFC 2577 – FTP Security questioned

RFC 2640 –Internationalization of the File Transfer Protocol.

RFC 3659 –Extensions to FTP

RFC 5797 – (Command and Extension Registry

RFC 7151 - File Transfer Protocol HOST Command for Virtual Hosts.

Description of Service

FTP creates 2 channels between a client and server, both using TCP connections. One of the channels is used to transfer data hence called the data channel. The other is always open during a session. It is called the control channel as the primary purpose of this channel is to exchange commands and replies to and from the FTP server hence providing control over the FTP connection. FTP protocol can be established in two ways namely an Active mode and a Passive mode. The Active mode is where the client provides the server with the port it will be listening on using the PORT <XX> command. The server will then establish a data connection through its port 20 to the XX port specified by the client.

In the Passive mode however, the client sends a PASV command to the server indicating and requesting for the server to send its IP and port numbers for the client to establish a data connection to it. This may be used when the clients firewall blocks incoming connections and it may have to establish a connection manually.

Since the ports until 1023 are reserved on a computer for system required tasks, the client usually uses higher unused ports while establishing connections to the FTP server

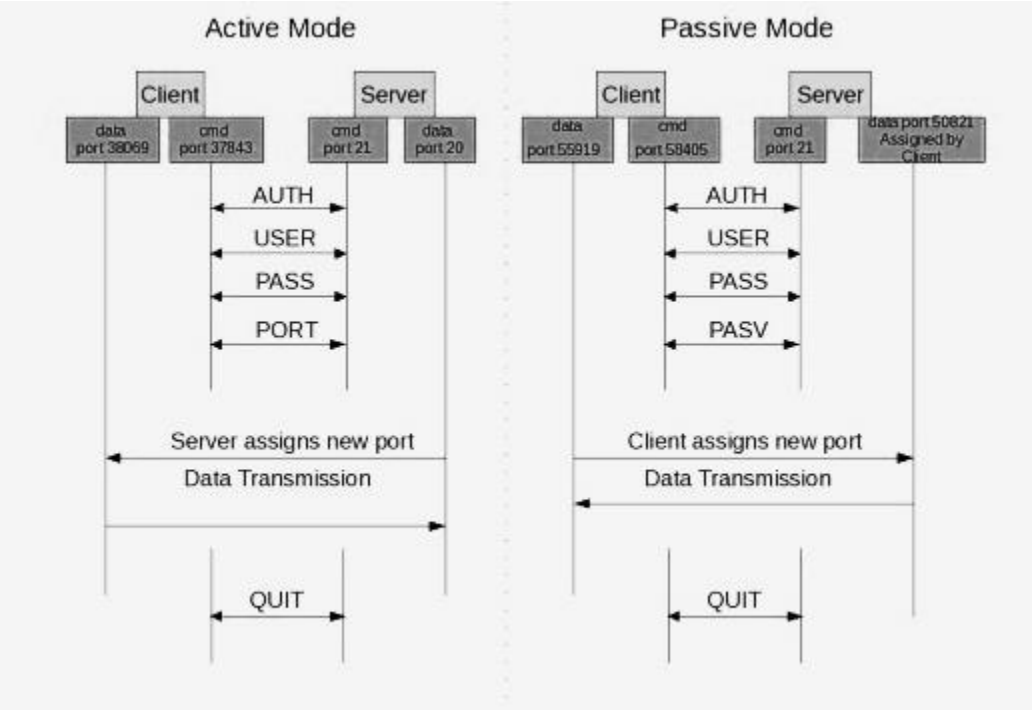

Fig: Active FTP vs Passive FTP (https://www.mybluelinux.com/active-and-passive-ftp-simplified-understanding-ftp-ports/)

The client first sends an FTP request on the server port 21. This will be the control channel. When established, the user will use TELNET to send commands on the channel to the server and get replies on the same channel. The user can use provide specification for a transfer to occur in a way for example MODE, STRU and TYPE commands define the way in which data is exchanged between the client and the server. All commands are 4 ASCII character long.

The replies from the server are 3-digit status codes in some cases followed by a description text if it has been defined by the server. The status codes are defined in RFC 959. The first digit specifies one of three possible outcomes

- 2yz – A command was successful
- 4yz,5yz – indicate a failure
- 3yz, 1yz - indicate error or a wait for next reply

The second digit of the FTP reply code define the type of error encountered if any; -

- x0z – indicate syntax errors.
- x1z – Requests for information.
- x2z – control or data connections error.
- x3z – Authentication and accounting example login and authentication
- x4z – Not defined.
- x5z – File system. These replies relay status codes from the server file system.

The third digit provides any further information regarding the error defined in the second digit.

PDU of FTP

The main link of communication between the server and the client is a command line interface terminal. After the control channel is established between the two ends, the user enters commands which are sent over from the user FTP process to the server-FTP process. The interaction during FTP is active is as shown below.

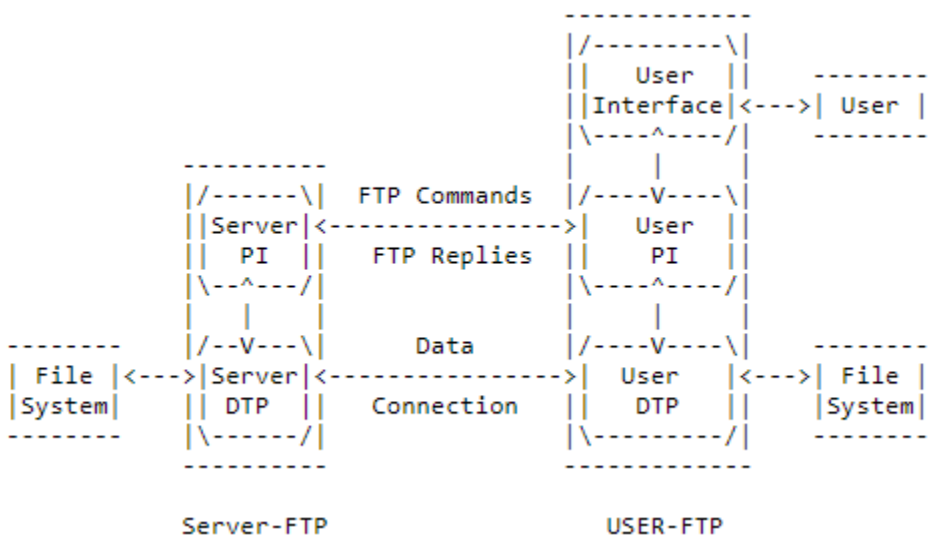

Fig: interaction between the server and client over an FTP service (RFC 959)

The user enters ASCII commands of four or fewer alphabetic characters in the terminal. The commands are not case sensitive. Commands are also sometimes passed with parameters or values which are also not sensitive to case. Arguments and commands are separated by the 'space' character and every command is not received by the server until it encounters the <CRLF> - Carriage Return Line Feed character sequence. FTP commands follow the syntax as below: -

```
<COMMAND> <SPACE> <ARGUMENT> <CRLF>
```

For e.g.: `USER Kunal <CRLF>`

In some cases even a BNF command line can also be used as below: -

```
<username>:: = <string>
 <password>:: = <string>
<account-information>:: = <string> …
```

For Some Useful FTP commands are as follows:-

PWD – Print Working Directory – Prints the current directory the user is in

CWD – Change Working Directory – Change directory or dataset for file storage

REIN – Reinitialize – Resets the connection right to the start. But lets any ongoing transfer complete

RETR – Retrieve – transfers a copy of the file specified to the Users system

STOR – Store – Stores a copy of the specified file (linked in attribute) to be stored in the server

ABOR – Abort – cancels any action taken by the server at the previous command

HELP – Help – Provides information regarding the control connection or providing specific information about a FTP command provided as an argument

For a standardized transfer, the sender of information must transmit data over a specified or a default representation. Hence translation of denotations must be processed at both receiving end and the senders end to the specified standard. Transfer can be done over various formats depending on the common factor between the Server and Client. ASCII, EBCDIC and binary are the few common understandable standards.

There can be a representation scheme selected by the user for carrying out transfer of files using the STRU command on the control channel. Representation can be in terms of File type structure, Record type structure or a Page kind of a structure. File is the default format. Since 2 different machines may have different defaults, it is usually set to one common default which sets the structure of documents transferred over the FTP.

File structure is used for unstructured continuous data byte sequences. In record structures data must be continuous type usually like text files made up sequential records. An End of Record (EOR) marker is used as a delimiter in such data.

A Page structure is used to transfer discontinuous files. In this case the files come with different page sizes and differ in information on each page. Hence a page header is sent with each page containing a Header length, a page index, length of data a flag defining its page type (0 – last page, 1 – simple page, 2 – descriptor page, 3 – access controlled page) another field may exist defining the page access control for each page.

Data transmission can also be in one of these modes (defined by the MODE command): -

1) Stream mode: Data is transmitted in a stream of bytes. Either of the representation structure can be used for transmission. An EOR or EOF is specified in a 2-byte control code. First byte is specified as the escape character.
2) Block mode: Data transmitted in series of blocks with one or more headers. The header consists of a descriptor and a count field. The count indicates total length of the data block in bytes, and the descriptor field consists of an EOF, EOR restart marker (in case of a failed transfer), or a suspect data field which tells if there are errors or the block is unreliable.
3) Compressed data: Is used to obtain a greatly increased bandwidth on a large transmission at small CPU cost.

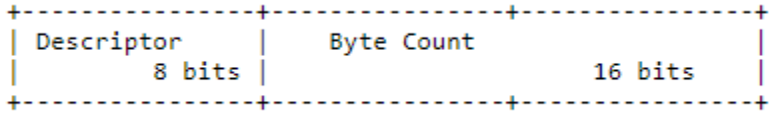

fig: PDU in BLOCK mode transmission (RFC 959)

DFA description

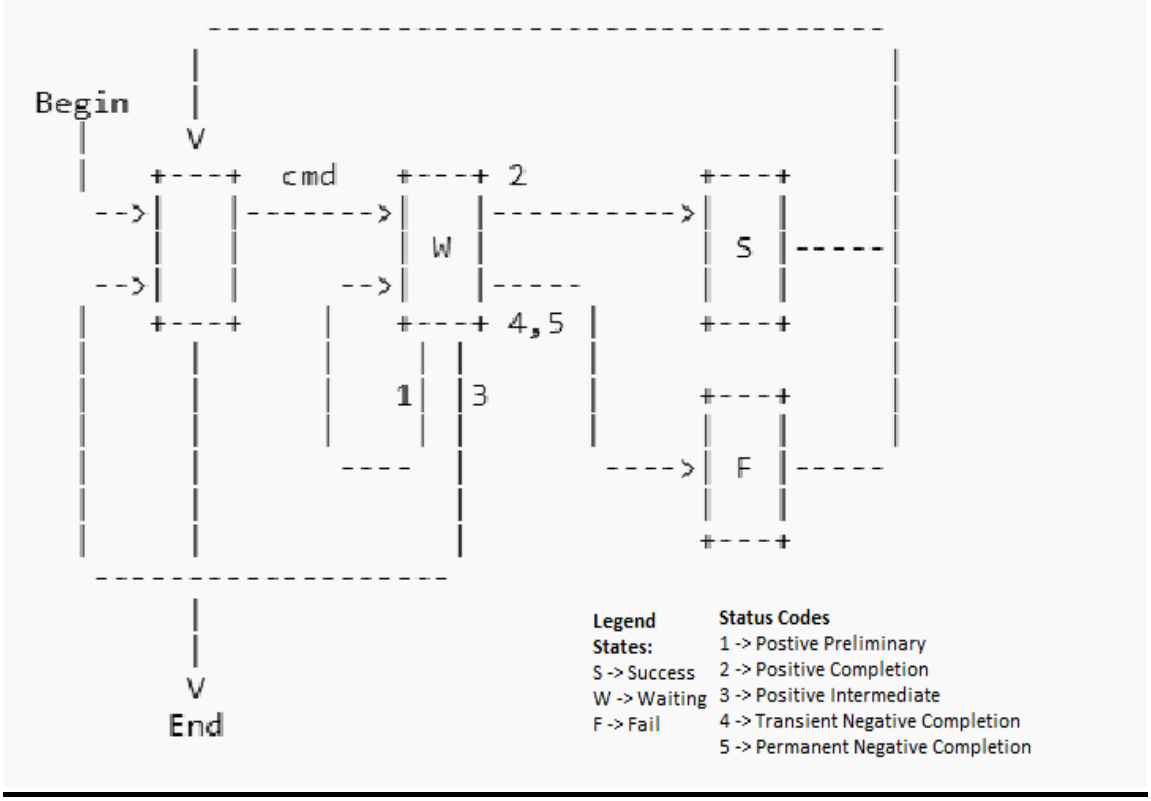

Fig: A generic Deterministic Finite Automata of FTP (RFC 959)

At the beginning, the user inputs a command. For instance, consider the user authentication. So, the USER command is input with the username. in case a valid username is entered, and the process goes through fine, the user code 3 is returned and the state goes back to the initial state with a request for a password from the server end i.e. a positive intermediate reply.

Now the user inputs the PASS command with the password attribute as per the server reply requesting a password. Now when the user enters the password, he is asked to wait for a verification check which returns the status code 1 meaning that the input went through fine and the server is making the user wait till it checks the input password. If the entered password attribute is correct, then the server will respond with status code of 2 which means the authentication is completed.

In case the password entered is wrong, the user is asked to go back and try again which would return status code 4. But if the user, instead of trying to enter the password asks to Print the working directory instead, then he will be shown status code 5 which means he should not try the same sequence of commands again and a command was typed out of sequence.

Common Trends And Subjective Analysis

Security

FTP protocol when it was implemented was used on ARPANET which had a few users and even fewer security threats. Hence FTP was designed with very few security measures. While authenticating a user, the attributes of the password command were entered without any masking from the server. This would have to be implemented at the client end.

When passing any commands or retrieving any data, all transfers were done in plane text. No encryption meant that nothing was protected. It can be safely said that FTP is a minimum-security protocol and it expects its carrier to take care of the security for it which is TCP.

This can lead to serious interruptions or breaching into the connection due to the data being in plain text. It is as good as having a no access control scheme defined in a system.

The data integrity is ensured in FTP by both the server and client meeting halfway by converting data into intermediate type which can be translated back on the receivers end to ensure that the received file is not useless. FTP also ensures that user can write data onto the server and append into the files while operating remotely.

Quality of Service

FTP does not ensure much reliability in terms of transferring data either as there is no conformation of a completed transfer. The file could turn up on the receivers end corrupted, or incomplete but with no way of knowing if it was just the parent file or something went wrong over the transfer process. Even for the reliability part, the protocol is dependent on TCP.

Speed is also not one of FTP's strengths. To transfer only a few files over the server, the client might have to perform too many back and fourths on the CLI to put in requests for each file. Since the data transport connection is terminated after each download, the connection needs to establish repeatedly in such a scenario. It can be said though that for transferring files of greater size, FTP might not be a bad protocol.

The FTP server does not alert the user of any system shutdowns and the user might be totally unaware of the status since there are no ways to check if a server is live.

User commands are not registered if a reply to the last command has not been received yet.

Deleting of a folder recursively is also a problem with FTP as a folder needs to be empty before it can be deleted. This can be a painful process for remote operations.

Extensibility

The FTP protocol can be extended with many functionalities to create an efficient transfer protocol. Since the protocol can connect a wide range of differing computers with varying file structure and its capability of handling various file types. It can be extended to create a kind of a backup safe for a network if its security can be enhanced in a way. By defining more security in the protocol or adding an encryption method. This might create implications like a need for a complex decoding algorithm and make the lightness characteristic to be compromised but would really build a safe Network access terminal.

Further, the protocol is extensible as it has a very simple model which can be easily modified to fit a designer's basic blueprint in creating a transfer protocol. The full duplex channels provide reliability as the user can stay connected and implementing fail safes or alerts might help in increasing the reliability. In general, a more interactive server end could make FTP a more reliable protocol.

References

- RFC 959 (Standard) File Transfer Protocol - https://tools.ietf.org/html/rfc959
- RFC 114 A file transfer protocol - https://tools.ietf.org/html/rfc114
- RFC 133 File transfer and error recovery- https://tools.ietf.org/html/rfc133
- RFC 1579 – Firewall friendly FTP was suggested - https://tools.ietf.org/html/rfc1579
- RFC 1635 – Using FTP anonymously - https://tools.ietf.org/html/rfc1635
- RFC 2228 – FTP Security Extensions proposed - https://tools.ietf.org/html/rfc2228
- RFC 2389 –Feature negotiation mechanism - https://tools.ietf.org/html/rfc2389
- RFC 2428 –Extensions for IPv6, NAT, Extended passive mode implementation - https://tools.ietf.org/html/rfc2428
- RFC 3659 –Extensions to FTP - https://tools.ietf.org/html/rfc3659
- RFC 5797 – Command and Extension Registry - https://tools.ietf.org/html/rfc5797
- RFC 7151 - File Transfer Protocol HOST Command for Virtual Hosts. - https://tools.ietf.org/html/rfc7151