FTP protocol (File Transfer Protocol)

Introduction to FTP protocol

FTP protocol (*File Transfer Protocol*) is, as its name indicates a protocol for transferring files.

The implementation of FTP dates from 1971 when a file transfer system (described in RFC141) between MIT machines (*Massachusetts Institute of Technology*) was developed. Many RFC have since made improvements to the basic protocol, but the greatest innovations date from July 1973.

The FTP protocol is currently defined by RFC 959 (*File Transfer Protocol (FTP) - Specifications*).

The role of FTP protocol

FTP protocol defines the way in which data must be transferred over a TCP/IPnetwork.
The aim of FTP protocol is to:

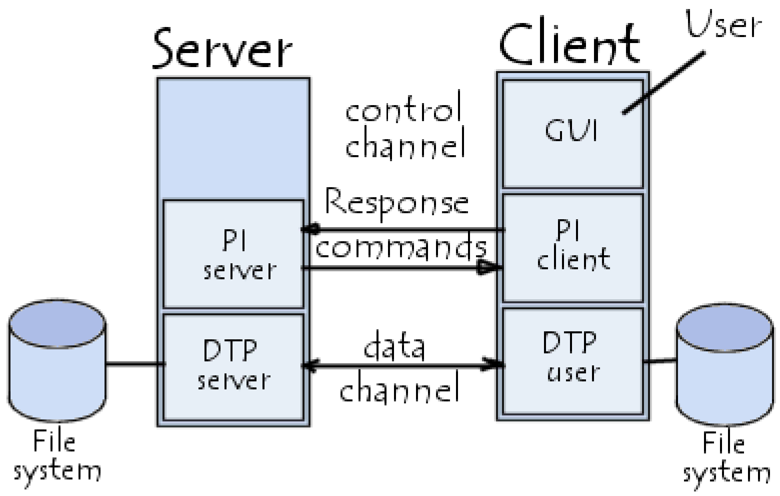
- allow file sharing between remote machines
- allow independence between client and server machine system files
- enable efficient data transfer

The FTP model

FTP protocol falls within a client-server model, i.e. one machine sends orders (the client) and the other awaits requests to carry out actions (the server).

During an FTP connection, two transmission channels are open:

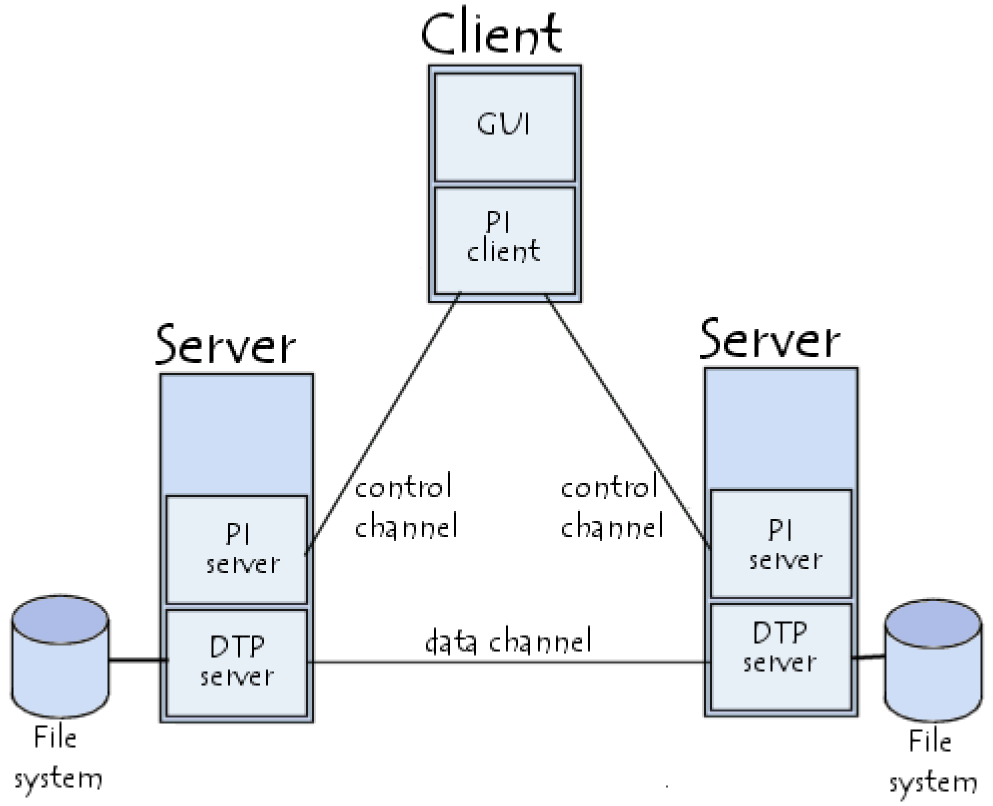
- A channel for commands (control channel)
- A channel for data



So, both the client and server have two processes allowing these two types of information to be managed:

- **DTP** (*Data Transfer Process*) is the process in charge of establishing the connection and managing the data channel. The server side DTP is called*SERVER-DTP*, the client side DTP is called *USER-DTP*
- **PI** (*Protocol Interpreter*) interprets the protocol allowing the DTP to be controlled using commands received over the control channel. It is different on the client and the server:
  - The SERVER-PI is responsible for listening to the commands coming from a USER-PI over the control channel on a data port, establishing the connection for the control channel, receiving FTP commands from the USER-PI over this, responding to them and running the SERVER-DTP.
  - The USER-PI is responsible for establishing the connection with the FTP server, sending FTP commands, receiving responses from the SERVER-PI and controlling the USER-DTP if needed.

When an FTP client is connected to a FTP server, the USER-PI initiates the connection to the server according to the Telnet protocol. The client sends FTP commands to the server, the server interprets them, runs its DTP, then sends a standard response. Once the connection is established, the server-PI gives the port on which data will be sent to the Client DTP. The client DTP then listens on the specified port for data coming from the server.
It is important to note that since the control and data ports are separate channels, it is possible to send commands from one machine and receive data on another. So, for example it is possible to transfer data between FTP servers by passing through a client to send control instructions and by transferring information between two server processes connected on the right port.



In this configuration, the protocol imposes that the control channels remain open throughout the data transfer. So a server can stop a transmission if the control channel is broken during transmission.

The FTP commands

All communication conducted on the control channel follows Telnet protocol recommendations. So, the FTP commands are Telnet character strings (in NVT-ASCII code) ending in the Telnet end of line code (i.e. the sequence <CR>+<LF>,**Carriage Return** followed by the **Line Feed** character, noted <CRLF>). If the FTP command has a parameter, this is separated from the command by a space (<SP>).
FTP commands make it possible to specify:

- The port used
- The method of data transfer.
- Data structure
- The nature of the action to be conducted (Retrieve, List, Store, etc.)

There are three different types of FTP commands:

- Access control commands
- Transfer parameter commands
- FTP service commands

Access control commands	
Command	**Description**
USER	Character string allowing the user to be identified. User identification is necessary to establish communication over the data channel.
PASS	Character string specifying the user's password. This command must immediately precede the *USER* command. It falls to the client to hide the display of this command for security reasons.
ACCT	Character string representing the user's account. The command is generally not necessary. During the response accepting the password, if the response is 230 this stage is not necessary, if the response is 332, it is.
CWD	*Change Working Directory*: this command enables the current directory to be changed. This command requires the directory's access path to be fulfilled as an argument.
CDUP	*Change to Parent Directory*: this command allows you to go back to the parent directory. It was introduced to solve problems of naming the parent directory according to the system (generally "*..*").
SMNT	*Structure Mount*:
REIN	*Reinitialize*:
QUIT	Command enabling the current session to be terminated. The server waits to finish the transfer in progress if the need arises, then supplies a response before closing the connection.

Transfer parameter commands	
Command	**Description**
PORT	Character string allowing the port number used to be specified.
PASV	Command making it possible to indicate to the DTP server to stand by for a connection on a specific port chosen randomly

	from among the available ports. The response to this command is the IP address of the machine and port.
TYPE	This command enables the type of format in which the data will be sent to be specified.
STRU	Telnet character specifying the file structure (F for *File*, R for *Record*, P for*Page*).
MODE	Telnet character specifying data transfer method (S for *Stream*, B for *Block*, C for *Compressed*).
FTP service commands	
Command	**Description**
RETR	This command (*RETRIEVE*) asks the server DTP for a copy of the file whose access path is given in the parameters.
STOR	This command (*store*) asks the server DTP to accept the data sent over the data channel and store them in a file bearing the name given in the parameters. If the file does not exist, the server creates it, if not it overwrites it.
STOU	This command is identical to the previous one, only it asks the sever to create a file where the name is unique. The name of the file is returned in the response.
APPE	Thanks to this command (*append*) the data sent is concatenated into the file bearing the name given in the parameter if it already exists, if not, it is created.
ALLO	This command (*allocate*) asks the server to plan a storage space big enough to hold the file whose name is given in the argument.
REST	This command (*restart*) enables a transfer to be restarted from where it stopped. To do so, the command sends the marker representing the position in the file where the transfer had been interrupted in the parameter. This command must immediately follow a transfer command.
RNFR	This command (*rename from*) enables a file to be renamed. In the parameters it indicates the name of the file to be renamed and must be immediately followed by the *RNTO* command.

RNTO	This command (*rename to*) enables a file to be renamed. In the parameters it indicates the name of the file to be renamed and must be immediately followed by the *RNFR* command.
ABOR	This command (*abort*) tells the server DTP to abandon all transfers associated with the previous command. If no data connection is open, the DTP sever does nothing, if not it closes it. The control channel however remains open.
DELE	This command (*delete*) allows a file to be deleted, the name of which is given in the parameters. This command is irreversible, confirmation can only be given at client level.
RMD	This command (*remove directory*) enables a directory to be deleted. The name of the directory to be deleted is indicated in the parameters.
MKD	This command (*make directory*) causes a directory to be created. The name of the directory to be created is indicated in the parameters.
PWD	This command (*print working directory*) makes it possible to resend the complete current directory path.
LIST	This command allows the list of files and directories present in the current directory to be resent. This is sent over the passive DTP. It is possible to place a directory name in the parameter of this command, the server DTP will send the list of files in the directory placed in the parameter.
NLST	This command (*name list*) enables the list of files and directories present in the current directory to be sent.
SITE	This command (*site parameters*) causes the server to offer specific services not defined in the FTP protocol.
SYST	This command (*system*) allows information on the remote server to be sent.
STAT	This command (*status*) makes it possible to transmit the status of the server, for example to know the progress of a current transfer. This command accepts an access path in the argument, it then returns the same information as LIST but over the control channel.

HELP	This command gives all the commands understood by the server. The information is returned on the control channel.
NOOP	This command (*no operations*) is only used to obtain an OK command from the server. It can only be used in order not to be disconnected after an excessive period of inactivity.

The FTP responses

The FTP responses make it possible to ensure synchronization between the client and FTP server. So, at each command sent by the client, the server will potentially carry out an action and systematically send back a response.

The responses are made up of a 3 digit code indicating the way in which the command sent by the client has been processed. However, since this 3 digit code is hard to read for humans, it is accompanied by a text (Telnet character string separated from the numeric code by a space).

The response codes are made up of 3 numbers the meanings of which are as follows:

- The first number indicates the status of the response (success or fail)
- The second number indicates what the response refers to.
- The third number gives a more specific meaning (relative to each second digit)

First number		
Digit	**Meaning**	**Description**
1yz	Preliminary positive response	The action requested is in progress, a second response must be obtained before sending a second command
2yz	Positive fulfilment response	The action requested has been fulfilled, a new command can be sent
3yz	Intermediary positive response	The action request is temporarily suspended. Additional information is awaited from the client
4yz	Negative fulfilment response	The action requested has not taken place because the command has temporarily not been accepted. The client is requested to try again later
5yz	Permanent negative response	The action requested has not taken place because the command has not been accepted. The client is requested to formulate a different request

Second number		
Digit	**Meaning**	**Description**
x0z	Syntax	The action has a syntax error, or is a command not understood by the server
x1z	Information	This is a response sending back information (for example a response to a STAT command)
x2z	Connections	The response relates to the data channel
x3z	Authentication and accounts	The response relates to the (USER/PASS) login or the request to change the account (CPT)
x4z	Not used by the FTP protocol	
x5z	File system	The response relates to the remote file system

Source: http://en.kioskea.net/contents/272-ftp-protocol-file-transfer-protocol