IT DEPARTMENT

Formative Cycle: WEB APPLICATIONS DEVELOPMENT **Module**: DEVELOPMENT AW

Course: SECOND **Academic year**: 2023-24 **Unit 1** : The file transfer protocol. FTP

Evaluation: FIRST

Approval date: 09/30/23 **Completion date**: 09/30/2023

Unit prepared and approved by: Juan Antonio Gascón Sorribas

The FTP Service (File Transfer Protocol).

Summary

1 The FTP Service

FTP is an **application** layer protocol in the **OSI** and **TCP/IP** models, designed to provide a standard file transfer service between systems connected to **TCP/IP** networks.

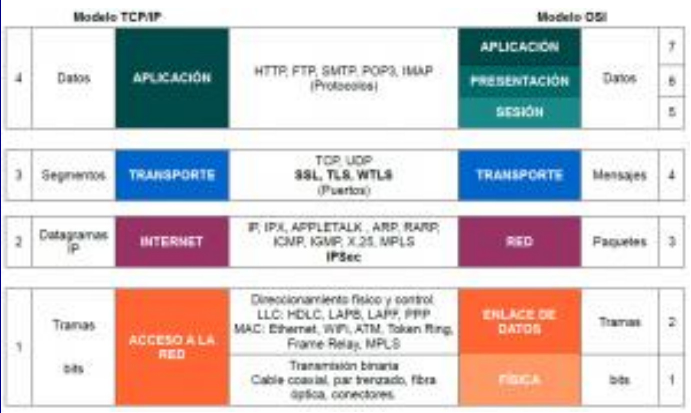

Modelo TCP/IP
Modelo OSI
4
Datos
APLICACIÓN
HTTP, FTP, SMTP, POP3, IMAP (Protocolos)
APLICACIÓN
PRESENTACIÓN
SESIÓN
Datos
7
6
5
3
Segmentos
TRANSPORTE
TCP, UDP SSL, TLS, WTLS (Puertos)
TRANSPORTE
Mensajes
4
2
Datagramas IP
INTERNET
IP, IPX, APPLETALK , ARP, RARP, ICMP, IGMP, X.25, MPLS IPSec
RED
Paquetes
3
1
Tramas
bits
ACCESO A LA RED
Direccionamiento físico y control LLC: HDLC, LAPB, LAPF, PPP MAC: Ethernet, WiFi, ATM, Token Ring, Frame Relay, MPLS
Transmisión binaria Cable coaxial, par trenzado, fibra óptica, conectores
ENLACE DE DATOS
FÍSICA
Tramas
bits
2
1

Figure 1Diagram of the TCP/IP and OSI models

1.1 Characteristics.

Although **FTP was created in 1971**, it is still used on **the Internet** and in **internal networks**.

By means of **FTP** we can:

- List directories and files on remote systems.
- Transfer files to and from the remote system, i.e. upload or download files.
- Perform other actions on the remote system such as renaming, deleting, creating files and directories, changing permissions, unzipping, etc.

Its operation is based on **client/server** architecture:

- **FTP Client:** They establish connections with **FTP servers** and allow us to upload or download files to or from the **FTP server**. There are multiple **FTP clients**. Depending on the user interface they can be:
  - **Command-line clients:** This type of client is integrated in almost all operating systems and is invoked from the command line interface (CLI: Command-Line Interface), with the command ftp server where server corresponds to the hostname or IP of the **FTP server**.
  - **Graphical clients:** They offer the user a graphical interface that facilitates the establishment of connections with servers and the transfer or sending of files. Some of the most widely used are: **Filezilla client**, **gFTP**, **CuteFTP**, **WinSCP**, etc.
  - **Browsers/Explorers:** Browsers such as **Firefox**, **Internet Explorer**, **Safari**, **Google Chrome**, etc, and file explorers such as **Nautilus**, **Dolphin**, **Explorer**, etc, can act as **FTP clients**. To use them you must indicate in the address that you are going to use the

FTP protocol.
The format for the address would be the following: ftp://[user][:password]@server
Some examples: ftp://ftp.zeppelinux.es, ftp://usuario@192.168.1.10.
They are limited but easy to use **FTP clients.**

- **FTP Server**: They control the connections of the **FTP clients** and depending on the defined privileges, they allow the download or upload of files. They offer multiple configuration options to establish user privileges, upload and download limits, connection and waiting times, etc.

  Probably the most used and well-known servers are the following:
  - For **Linux/Unix** systems: **vsftpd** or **ProFTPD**.
  - For **Windows** systems: **FTP server included in IIS**, **Filezilla Server** or **Server-U**.

2 The FTP Protocol

The FTP protocol is the set of standards and rules by which **FTP clients and servers** communicate. Communication between them is based on sending text messages containing commands and responses. **The FTP protocol** uses **TCP** as the transport protocol.

2.1 Types of access

FTP servers, depending on their configuration, allow two types of access from **FTP clients**:

2.1.1 Anonymous access:

The FTP client connects to a special user whose standard name is anonymous or ftp. As a rule, the anonymous user, unless the FTP server administrator decides otherwise, can only download files and his access is limited to a special directory on the server. He is caged (chrooted) or confined to that special directory.

2.1.2 Authorized access**:**

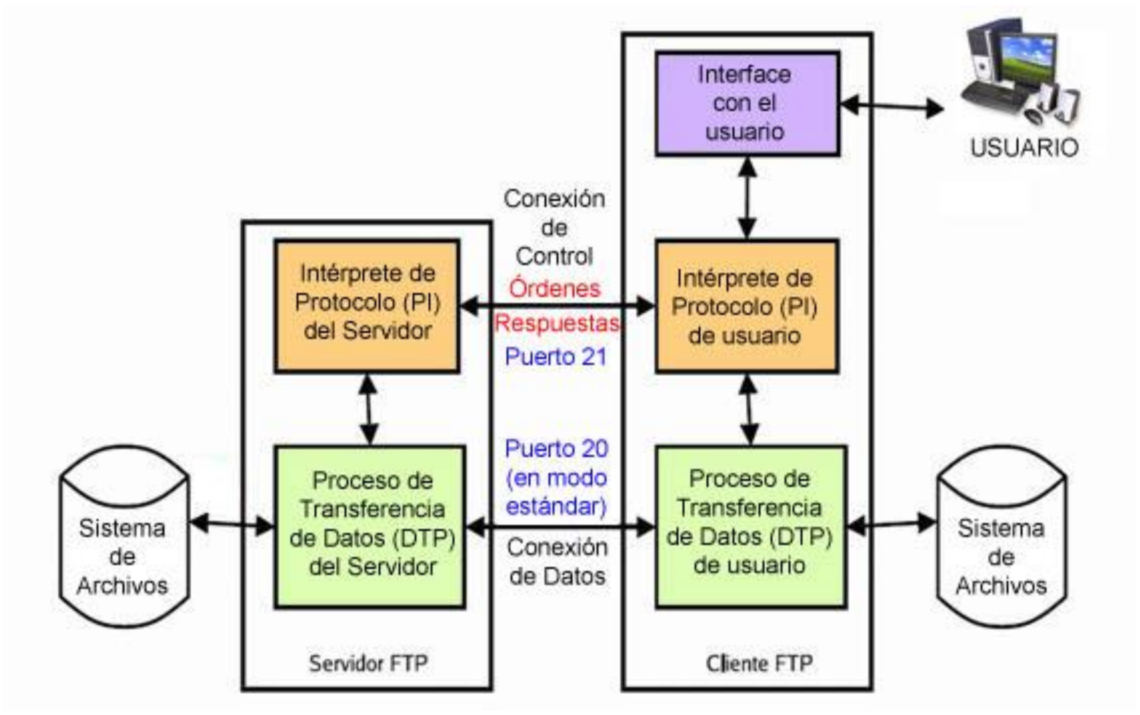
The **FTP client** connects to an existing user on the server. This user can be:

- **A user of the operating system (Local Users)** where the **FTP server** is installed.
- **A virtual user**, created for **FTP** access. Its credentials can be stored in a **database**, in a **directory service (LDAP)**, in a **text file**, etc.

Once authenticated, the user accesses a directory on the server in which he/she may or may not be **caged (chroot)**. In addition, the **FTP server** administrator will assign the privileges that each user will have, such as: download, upload, delete, limit space, access to certain directories, limit speed, etc.

3 Connections

FTP is based on **TCP** and uses various **connections** and **ports**. There are control connections and data connections. **FTP servers** and **clients** establish **TCP** connections, some for control and some for data transfer.


Interface con el usuario
USUARIO
Conexión de Control
Órdenes
Respuestas
Puerto 21
Intérprete de Protocolo (PI) del Servidor
Intérprete de Protocolo (PI) de usuario
Puerto 20 (en modo estándar)
Conexión de Datos
Sistema de Archivos
Proceso de Transferencia de Datos (DTP) del Servidor
Proceso de Transferencia de Datos (DTP) de usuario
Sistema de Archivos
Servidor FTP
Cliente FTP

3.1 Control connection.

The FTP client establishes a connection with the FTP server to dialogue with it. It sends commands to list (`ls`), download (`get`), upload (`put`), etc., and receives responses from the servers informing how they handle the requests. The control connection is active until the user logs off or the server terminates the session because of inactivity timeout. Servers can handle multiple control connections simultaneously, as many as the server configuration allows. No data is ever sent over the control connection.

3.2 Data connection.

When the **FTP client** requests a data transfer, a data connection is created, which will be closed at the end of the transfer. Associated with a control connection there can be multiple simultaneous data connections, as many as simultaneous transfers and up to a maximum set in the server configuration. **Control commands are never sent over data connections**.

In principle, **FTP servers** use **port 21/TCP** for control connections and **port 20/TCP** for data connections, but because servers can work in **active or passive mode**, they do not always use **port 20/TCP** for data connections. **FTP clients** do not use **well-known ports** to initiate or service connections, they always use **ports higher than 1023**.

4 Modes

FTP clients can connect to **FTP servers** in two different ways, in **active mode** and in **passive mode**.

4.1 Active mode

1 The client establishes a **control connection** to the server:

- Open local port 1500 (**port higher than 1023**).
- Establishes a connection to **port 21/TCP** of the **FTP server**.

2 The customer requests, for example, a file transfer:

- The client sends the **PORT command** to the server indicating its IP address and a port number to be opened for use in the **data connection**, in this example, port 1501 (**port greater than 1023**).
- The **FTP server** initiates a TCP connection from its port 20 to port 1501.
- **The data connection** is used to transfer information.

In this mode, it is the **FTP server that** initiates the data connections and forces the client to open ports to handle these connections.

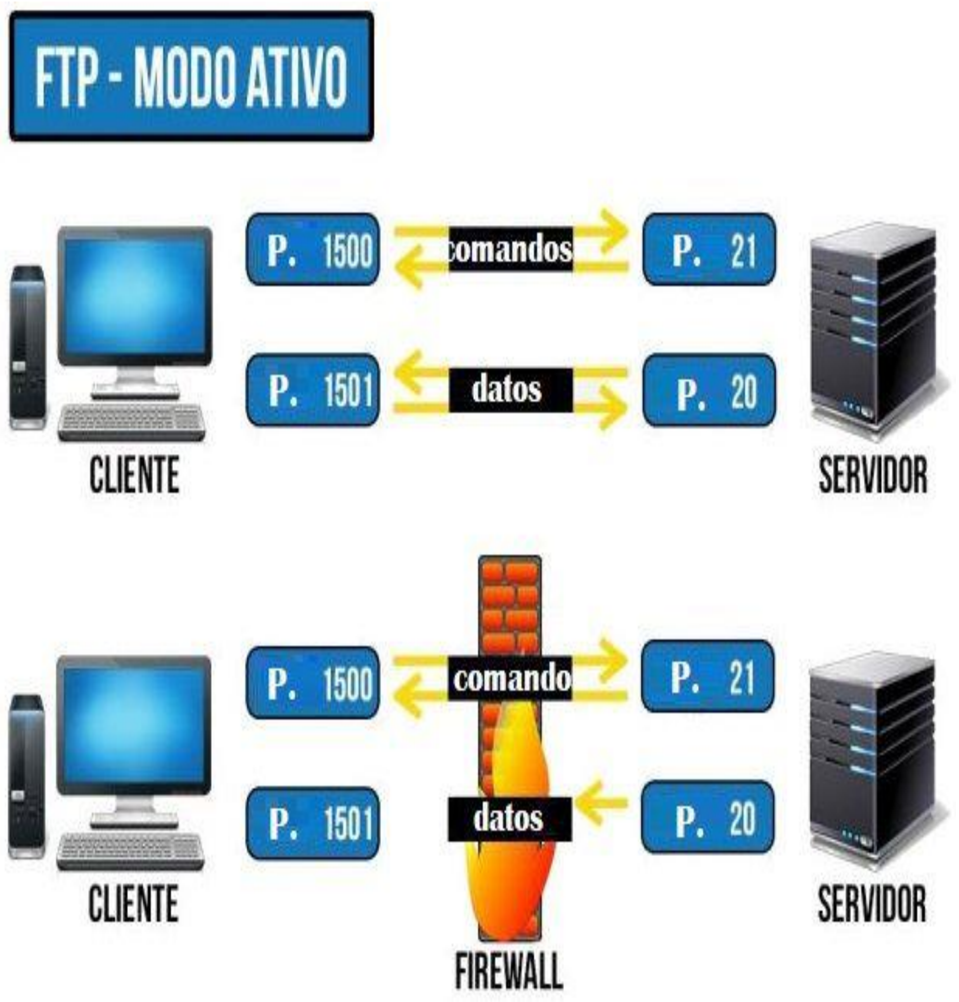


Figure 2FTP active mode diagram

4.2 Passive mode

In **passive mode** it is always the **FTP client that** initiates connections to the **FTP server**, both control and data connections. **Port 20/TCP** of the server is not used.

1 The client establishes a **control connection** with the server:
   - Open local port 4000 (**port higher than 1023**).
   - Establishes a connection to **port 21/TCP** of the **FTP server**.

2 When, for example, a file transfer is requested:
   - The client sends the **PASV command** to activate **passive mode**. In response to this command, the **FTP server** sends a port number it has available, for example 5000.
   - The **FTP client** initiates a TCP connection, opens a **local port greater than 1023**, for example 4001, to the port specified by the **FTP server** (in our example it is 5000).
   - **The data connection** is used to transfer information.

- **Passive mode** avoids the security problem of the **FTP client** having to accept connections on **ports greater than 1023** but shifts this security problem to the **FTP server**. The machine running the **FTP server** has to accept connections on multiple ports and this is a threat to the security of the machine.

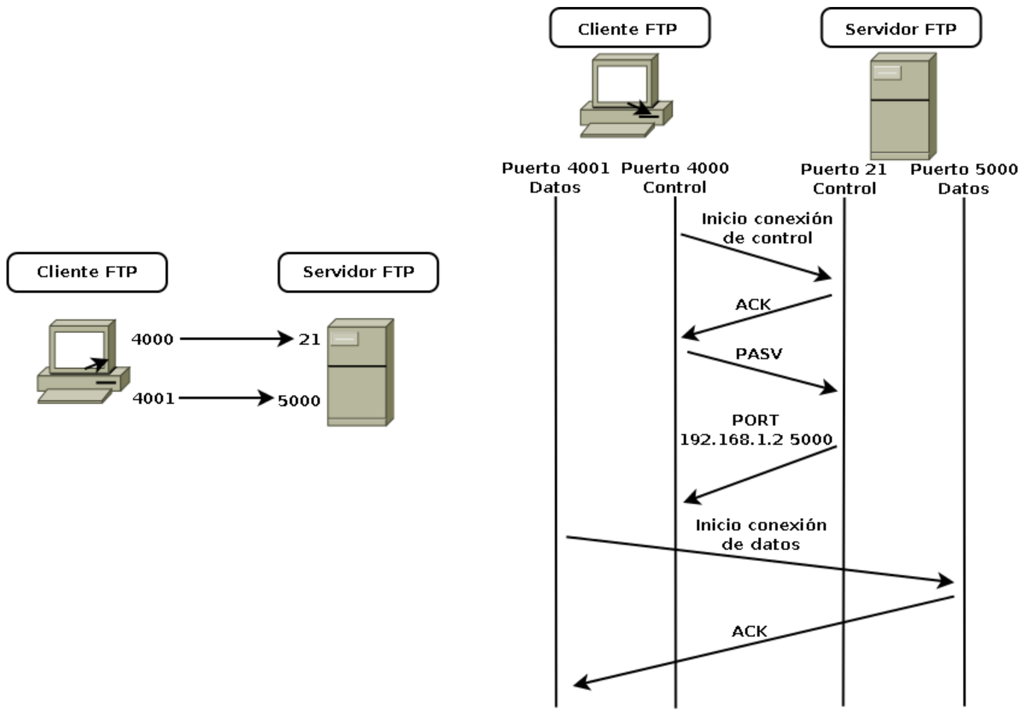


Figure 3Passive FTP mode diagram

5 File transfer types

In the **FTP protocol** there are two modes of file transfer, **ASCII** and **binary**.

- **ASCII mode (type ascii):** Transmitted **byte by byte**. Optimal for text files such as .txt, .html, etc.
- **Binary mode (type bin):** transmitted **bit by bit**. Optimal for non-text files such as multimedia files, executables, etc.

We can define in which mode we transfer files from the **FTP client**. Most clients also offer **automatic mode,** which detects the type of file and sets the appropriate transfer type.

6 Protocol not secure

6.1 Security

Natively, the **FTP protocol** is not secure. It was designed for speed but not security. It implements user authentication mechanisms to determine access and transfer privileges on the **FTP server**, but does not guarantee it:

1. **Authenticity**. That the computers involved in the connection are who they say they are. **It is vulnerable to phishing attacks.**
2. **Confidentiality**. The exchange of information, both the user and password as well as the transfer of any data, is done in **plain text**, **without encryption**. **It is vulnerable to network traffic analysis attacks.**

Although FTP is not natively a secure protocol, there are mechanisms to secure connections between clients and FTP servers such as FTPS or sending the **FTP protocol through** an **SSH tunnel**.

FTPS and secure alternatives to **FTP**, such as SFTP (SSH File Transfer Protocol).

6.2 FTPS or FTP/SSL

FTPS encapsulates **FTP** over SSL (Secure Socket Layer) or TLS (Transport Layer Security) to encrypt the exchange of information. Thanks to cryptographic algorithms and digital certificates, the **confidentiality** and **integrity** of the information transmitted can be guaranteed, as well as the **authenticity of the** equipment involved in the connection.
There are two ways to implement **FTPS**: **Explicit FTPS (FTPES)** and **Implicit FTPS**.

6.2.1 Implicit FTPS

- The client establishes a **control connection** and **the SSL/TLS connection is established**.
- **If the server does not support FTPS**, the connection is closed.
- **All communications**, control connection and data connections **are encrypted**.
- **The client and server do not negotiate**.

To maintain compatibility with **FTP clients** that do not support **SSL/TLS**, other ports are used to handle **FTPS** requests. As standard ports **990/TCP for control** and **989/TCP for data** are used.

6.2.2 FTPS Explicit (FTPES)

- The client establishes a **control connection to port 21**, explicitly requests that the communication be secure by sending the **AUTH SSL** or **AUTH TLS** command and if the server supports it, an **SSL** or **TLS** connection is established based on cryptographic algorithms and digital certificates.

- If the server does not support **FTPS the** client is offered the possibility to use **unsecured FTP**.
- The client and server can **negotiate** which part of the communications, control connection or data connections, will be encrypted.
- This is the recommended method because it allows greater control over communication.

FTPS is not to be confused with SFTP (SSH FTP) which is a file transfer protocol based on SSH, nor with **Secure FTP** which is sending the **FTP** protocol through an **SSH** tunnel.

1. SFTP/SCP, secure alternatives to FTP SSH (Secure Shell) is an application layer protocol designed to provide terminal access to remote computers. It is based on the **client/server** model. The **SSH client** establishes connections to terminals on the computer where the **SSH server** is installed. **SSH servers** use port **22/TCP** as the standard port. **SSH** provides **authentication**, **confidentiality** and **integrity**.

- **Both ends of the connection are authenticated:** The server authenticates to the client with a key and the client authenticates to the server.

Transmitted data is encrypted: Usernames and passwords and all transmitted information are encrypted.

6.2.3 SSH

In addition, it integrates mechanisms for file transfer, also guaranteeing **authentication**, **confidentiality** and **integrity**. These mechanisms are SFTP (SSH FTP) and SCP (Secure Copy Protocol).

- **SFTP**:
  - Allows the transfer of files between remote systems.
  - Allows you to list files and directories on the server.
  - Allows you to perform additional functions on the server such as renaming, deleting, creating files and folders, changing permissions, unzipping, etc.
- **SCP**:
  - Allows copying files between remote systems

Some graphical SCP clients add functionality such as list, delete, etc. These are not pure SCP clients.

6.2.4 FXP Protocol

FXP(File eXchange Protocol) is a direct data transfer protocol between **FTP servers**.

An **FTP client** with **FXP** support establishes the connection between **FTP servers**. That is, the client uses its bandwidth only to establish the initial connection between the **FTP servers**, not to transfer files, the latter is done directly between the servers. For this to be possible, the servers have to allow it.

6.2.5 The TFTP Service

TFTP (Trivial FTP) is an application layer protocol for simple and fast file transfer. Based on the **client/server**
model, there are **TFTP clients** and **TFTP servers**.

Its main characteristics are:

- **TFTP uses UDP as transport protocol**. **TFTP servers** listen on port **69/UDP** as a standard port. **UDP** does not guarantee the integrity of the transmitted information but the transfer speed is higher than in **FTP**.
- There is no authentication or encryption mechanism. It is a service mainly used by network devices or computers to upload or backup the operating system, configuration files, etc.