Informe del Proyecto Final de Redes de Computadoras

Richard Alejandro Matos Arderí
Mauricio Sunde Jiménez

Grupo 311, Ciencia de la Computación.

Facultad de Matemática y Computación

Universidad de La Habana.

2025

Índice

1. Introducción Teórica sobre el Protocolo FTP

El Protocolo de Transferencia de Archivos (FTP, por sus siglas en inglés) es un protocolo de red estándar utilizado para la transferencia de archivos entre un cliente y un servidor en una red TCP/IP. FTP fue uno de los primeros protocolos desarrollados para la transferencia de archivos y sigue siendo ampliamente utilizado hoy en día.

FTP opera en un modelo cliente-servidor, donde el cliente inicia la conexión y el servidor responde a las solicitudes del cliente. El protocolo utiliza dos canales de comunicación: un canal de control para enviar comandos y recibir respuestas, y un canal de datos para la transferencia de archivos.

FTP fue especificado por primera vez en 1971 y ha evolucionado a lo largo de los años. La versión más utilizada del protocolo está definida en el RFC 959, publicado en 1985.

2. RFC 959: Especificaciones del Protocolo FTP

El RFC 959, titulado "File Transfer Protocol (FTP)", es un documento que especifica las reglas y estándares para la implementación del protocolo FTP. Fue publicado en octubre de 1985 por Jon Postel y Joyce Reynolds como parte de la serie de Request for Comments (RFC) de la Internet Engineering Task Force (IETF).

El RFC 959 define las especificaciones del protocolo FTP, incluyendo los comandos y respuestas que deben ser implementados por los clientes y servidores FTP. Algunos de los comandos más importantes incluyen:

- **USER**: Enviar el nombre de usuario al servidor.
- **PASS**: Enviar la contraseña del usuario al servidor.
- **STOR**: Subir un archivo al servidor.
- **RETR**: Descargar un archivo del servidor.
- **LIST**: Listar los archivos en el directorio actual del servidor.
- **QUIT**: Cerrar la conexión con el servidor.

El RFC 959 también define los códigos de respuesta que el servidor debe enviar en respuesta a los comandos del cliente. Estos códigos de respuesta indican el estado de la solicitud y pueden ser utilizados por el cliente para determinar el siguiente curso de acción. Los códigos de respuesta están organizados en categorías, como:

- **1xx**: Respuestas positivas preliminares.
- **2xx**: Respuestas positivas de finalización.
- **3xx**: Respuestas positivas intermedias.
- **4xx**: Respuestas negativas transitorias.
- **5xx**: Respuestas negativas permanentes.

Además de los comandos básicos, el RFC 959 especifica una serie de comandos adicionales para manejar diversas funcionalidades, como la navegación por directorios, la gestión de permisos de archivos y la configuración de modos de transferencia. Algunos de estos comandos adicionales incluyen:

- **CWD**: Cambiar el directorio de trabajo.
- **PWD**: Imprimir el directorio de trabajo actual.
- **MKD**: Crear un nuevo directorio.
- **RMD**: Eliminar un directorio.
- **DELE**: Eliminar un archivo.
- **RNFR**: Renombrar un archivo (parte 1).
- **RNTO**: Renombrar un archivo (parte 2).

El RFC 959 también aborda aspectos de seguridad, aunque de manera limitada. Por ejemplo, menciona la necesidad de proteger las credenciales de usuario (nombre de usuario y contraseña) durante la autenticación, pero no especifica mecanismos de cifrado. En la práctica, se utilizan extensiones como FTPS (FTP sobre SSL/TLS) y SFTP (FTP sobre SSH) para proporcionar seguridad adicional.

Para más información sobre el RFC 959, se puede consultar el documento completo en el sitio web de la IETF: `https://tools.ietf.org/html/rfc959`.

3. Estructura del Proyecto

El proyecto está organizado en dos partes principales: el cliente FTP y el servidor FTP. Cada parte está implementada en su propio directorio y contiene varios módulos para manejar diferentes aspectos de la funcionalidad.

3.1. Cliente FTP

El cliente FTP está implementado en el directorio `client_lib` y contiene los siguientes módulos:

- `ftp_client.py`: Implementa la lógica principal del cliente FTP, incluyendo la conexión al servidor, el envío de comandos y la recepción de respuestas.
- `utils.py`: Contiene funciones utilitarias para manejar archivos, como obtener el tamaño de un archivo, leer un archivo y escribir datos en un archivo.
- `security.py`: Implementa funciones de seguridad, como la generación y verificación de hashes SHA-256.
- `test/test_ftp_client.py`: Contiene pruebas unitarias para verificar el correcto funcionamiento del cliente FTP.

3.2. Servidor FTP

El servidor FTP está implementado en el directorio `server` y contiene los siguientes módulos:

- `run_server.py`: Script para ejecutar el servidor FTP.
- `app/ftp_server.py`: Implementa la lógica principal del servidor FTP, incluyendo la aceptación de conexiones de clientes y el manejo de comandos.
- `app/handlers.py`: Contiene manejadores para los comandos FTP, como `USER`, `PASS`, `STOR`, `RETR`, `LIST` y `QUIT`.
- `app/security.py`: Implementa funciones de seguridad, como la generación y verificación de hashes SHA-256.
- `app/config.py`: Archivo de configuración para el servidor FTP.

3.3. Interfaz Visual con React Vite

La interfaz visual del cliente FTP está implementada utilizando React y Vite. React es una biblioteca de JavaScript para construir interfaces de usuario, mientras que Vite es una herramienta de construcción rápida para proyectos de frontend. La interfaz visual permite a los usuarios interactuar con el cliente FTP de manera intuitiva y moderna.

El directorio `client_app` contiene los siguientes módulos:

- `App.jsx`: Componente principal de la aplicación React.
- `components/Home.jsx`: Componente que representa la página principal de la interfaz del cliente FTP.
- `styles/GlobalStyle.js`: Archivo que define los estilos globales de la aplicación utilizando `styled-components`.

4. Tecnologías Utilizadas

El proyecto utiliza las siguientes tecnologías:

- **Python**: Lenguaje de programación utilizado para implementar tanto el cliente como el servidor FTP.
- **Sockets**: Utilizados para la comunicación entre el cliente y el servidor a través de la red.
- **Threads**: Utilizados en el servidor para manejar múltiples conexiones de clientes simultáneamente.
- **Unittest**: Biblioteca de pruebas unitarias de Python utilizada para verificar el correcto funcionamiento del cliente FTP.

- **React**: Biblioteca de JavaScript utilizada para construir la interfaz de usuario del cliente FTP.
- **Vite**: Herramienta de construcción rápida utilizada para el desarrollo del frontend.
- **Styled-components**: Biblioteca utilizada para aplicar estilos a los componentes de React.
- **React-icons**: Biblioteca utilizada para incluir iconos en la interfaz de usuario.

5. Funcionamiento del Cliente y Servidor FTP

El cliente FTP se conecta al servidor utilizando sockets y envía comandos FTP para realizar operaciones como subir, descargar y listar archivos. El servidor FTP escucha en un puerto específico y maneja las conexiones de los clientes, respondiendo a los comandos y realizando las operaciones solicitadas.

El servidor utiliza threads para manejar múltiples conexiones de clientes al mismo tiempo, lo que permite que varios clientes se conecten y realicen operaciones simultáneamente.

La interfaz visual del cliente FTP permite a los usuarios realizar estas operaciones de manera intuitiva a través de botones y listas de archivos, proporcionando una experiencia de usuario moderna y práctica.