Computer Networks Project

4rd Semester, Jan – May 2025

Title of Project: **Lightweight Telnet and FTP implementation in python**				
Sl No	Student Name	SRN:	Section	Group Number
1	Pranav Hemanth	PES1UG23CS433	G	-
2	Nishant K Holla	PES1UG23CS401	G	

ProtoMUX

Abstract

ProtoMUX is a Python-based project that demonstrates the use of socket programming to facilitate inter-device communication over a local network using FTP and Telnet protocols. Built with a focus on educational value for the Computer Networks lab at PES University, the project also features a user-friendly Tkinter GUI for enhanced usability. Users can send and receive files using FTP or establish a remote shell session using Telnet. The implementation helps students understand how protocol-based communication works under the hood.

Introduction

The internet consists of a multitude of protocols for data exchange and remote access. Two foundational protocols in this space are FTP (File Transfer Protocol) and Telnet (Telecommunication Network). While largely deprecated in favor of more secure alternatives, understanding their inner workings is essential to grasping core networking concepts.

ProtoMUX provides a hands-on experience in:

- Socket programming
- Network communication models
- Process and thread management
- GUI design using Tkinter

Objectives

- Implement FTP-like functionality for file transfers between devices on the same LAN.
- Emulate Telnet-like remote shell access using PTY and socket programming.
- Provide a cross-platform GUI interface using Python's standard libraries.
- Ensure modular and extensible code design for learning and demonstration purposes.

System Design

Architecture

The system follows a modular structure with separate folders for FTP and Telnet functionality. A unified GUI in `main_gui.py` allows users to interact with the protocols without needing to access the command line.

- **FTP Module**: Contains sender and receiver scripts that communicate using TCP sockets on port 2121.
- **Telnet Module**: Emulates a PTY shell using `os.fork`, `pty`, and `select`, enabling a remote terminal interface.
- **GUI Layer**: Tkinter interface that allows users to:
    - Enter IP addresses
    - Choose protocol (FTP or Telnet)
    - Select role (Sender or Receiver)
    - Choose files (for FTP senders)

Tools and Technologies

- **Language**: Python 3.x
- **Libraries**: `socket`, `os`, `pty`, `select`, `tkinter`, `threading`
- **Platform**: Cross-platform (tested on macOS and Linux)

Implementation Details

FTP

- Uses TCP sockets to transfer files over the network.
- The sender reads a file in binary chunks and sends it with a header indicating the file name.
- The receiver listens on a specified port, accepts incoming connections, and saves the received file.

Telnet

- The server launches a PTY (pseudo-terminal) and binds a socket to port 8023.
- Incoming client connections are attached to the PTY, enabling real-time remote shell access.
- This implementation does not use authentication and is intended for LAN-based demo purposes only.

GUI

- Built with `tkinter` and `ttk` to ensure native look and feel.
- Dynamically adjusts input fields based on user selections.
- Validates inputs before starting any session.

Testing and Results

The project was tested using two devices connected to the same Wi-Fi network. The following observations were made:

- FTP file transfers worked reliably for files up to ~100MB.
- Telnet shell sessions were responsive with minimal latency.
- The GUI worked consistently across test scenarios, enhancing usability.

Working

Limitations

- Telnet and FTP are not secure; this project is not intended for use outside trusted networks.
- No encryption or authentication mechanisms are implemented.
- File integrity and error checking mechanisms are minimal.

Conclusion

ProtoMUX achieves its goal of demonstrating protocol-based communication using core Python libraries. It serves as a practical educational tool for students learning computer networks and socket programming.

Future Enhancements

- Add support for secure protocols (e.g., SFTP, SSH).
- Implement file integrity checks using hashing.
- Enable multi-client support and better concurrency handling.
- Add logging and error reporting features.

Team Members

- **Nishant Holla** - PES1UG23CS401
- **Pranav Hemanth** - PES1UG23CS433

References

- Python Documentation: https://docs.python.org/3/
- Computer Networking: A Top-Down Approach by Kurose and Ross
- Tkinter GUI Programming by John Elder