Implementation of VLAN Communication using WiFi and Bluetooth on macOS

Daniel Nuño, *dnuno@cetys.edu.mx, Student, CETYS,*
Kevin Hernandez, *kevin.hernandez@cetys.edu.mx, Student, CETYS,*
Moises Sanchez Adame, *Dr, Professor, CETYS*

***Abstract*—This study focuses on the implementation of a Hybrid VLAN communication system utilizing both WiFi and Bluetooth protocols on macOS devices. WiFi communication was achieved using Python sockets, while Bluetooth communication was handled through CoreBluetooth in Swift for the server and Bleak in Python for the client. The research required a deep understanding of network compatibility constraints and the adaptation of tools to ensure successful bidirectional data transmission. Results confirm that VLAN communication can be established effectively using both technologies, proving the feasibility of hybrid communication methods on macOS.**

***Index Terms*—VLAN, WiFi Communication, Bluetooth Networking, Python Sockets, Swift CoreBluetooth, macOS.**

I. Introduction

Network infrastructures are continuously evolving, leading to an increasing demand for flexible communication solutions. VLANs (Virtual Local Area Networks) enable logical segmentation of networks, improving security and optimizing resource allocation. This project investigates a hybrid VLAN solution integrating WiFi and Bluetooth to establish secure communication between two macOS devices.

II. Theoretical Framework

A. Networking and Data Communication

Data communication enables devices to exchange information efficiently using various protocols and technologies. Two primary wireless communication methods explored in this project are WiFi-based VLAN communication and Bluetooth Low Energy (BLE). These technologies operate at different layers of the OSI model and serve distinct use cases. Understanding the underlying principles of these communication methods is essential for designing a robust client-server architecture.

B. WiFi Communication using Sockets

Sockets are fundamental to networking, enabling communication between devices over TCP/IP. Python's built-in `socket` library provides a simple yet powerful interface for implementing network communication over WiFi VLANs [1].

1) Concept of Sockets: A socket acts as an endpoint for sending and receiving data. In a client-server model, a server socket listens for incoming connections, while client sockets initiate communication. Sockets can use:

- TCP (Transmission Control Protocol): Ensures reliable, ordered, and error-free data transmission, making it ideal for file transfers.
- UDP (User Datagram Protocol): Provides faster but connectionless communication without reliability guarantees.

2) Implementation of Sockets in Python: Python's `socket` module follows a sequence of operations:

1) The server creates a socket and binds it to an IP and port.
2) The server listens for client connections.
3) The client establishes a connection to the server.
4) Data is exchanged between the client and server.
5) The connection is terminated when communication is complete.

This structured approach allows efficient file transfer and command execution between client and server over a VLAN.

3) Virtual LAN (VLAN) Implementation: A VLAN (Virtual Local Area Network) logically groups devices within a network to optimize security and efficiency. By using sockets over a VLAN, devices communicate securely without interference from external networks. This segmentation reduces congestion and enhances security, as unauthorized devices cannot access the VLAN without proper credentials.

C. Bluetooth Communication using CoreBluetooth and Bleak

Bluetooth Low Energy (BLE) is a power-efficient wireless communication standard designed for short-range data transfer. Unlike classic Bluetooth, BLE operates using the Generic Attribute Profile (GATT), which structures how devices exchange data.

1) BLE Server Implementation using CoreBluetooth: On macOS, the CoreBluetooth framework provides a native API for creating Bluetooth-enabled applications [2]. BLE servers (peripherals) advertise services that contain characteristics—specific attributes that store data.

The key CoreBluetooth components include:

- **CBPeripheralManager:** Handles Bluetooth advertisements and manages client requests.
- **CBService:** Defines the service provided by the server.
- **CBCharacteristic:** Represents a data attribute that clients can read or write.

2) BLE Client Implementation using Bleak: The Bleak library is an asynchronous BLE client that allows Python applications to interact with BLE peripherals across multiple platforms [3]. It enables:

- Scanning for BLE devices.
- Connecting to peripherals.
- Reading and writing BLE characteristics.
- Handling asynchronous BLE events.

3) Comparison: WiFi vs. Bluetooth for Data Transfer: While both WiFi sockets and BLE facilitate client-server communication, they differ in range, speed, and power consumption. Table **??** compares their key characteristics.

III. Methods and Materials

A. Materials

The following resources were used:

- Two macOS computers (Server and Client)
- Python (Sockets module for WiFi communication)
- Swift (CoreBluetooth for Bluetooth communication)
- Miniforge (Conda Virtual Environment for dependency management)
- Bleak library (Python BLE client)

IV. Methodology

A. Understanding the System Environment

The first step in this project was to assess the hardware and operating system constraints. The system consisted of two macOS computers, which meant that the approach had to be tailored specifically for macOS compatibility. Unlike Windows, macOS enforces stricter security policies for networking, particularly Bluetooth communication. As a result, selecting the appropriate tools was critical to ensuring successful VLAN communication.

B. Selection of Networking Tools

To establish VLAN communication, we considered multiple options before finalizing the following approaches:

- **WiFi-based VLAN:** Python's `socket` module was chosen due to its efficient handling of TCP/IP communication. It allowed for seamless server-client interaction, including file listing, uploading, and downloading.
- **Bluetooth-based VLAN:** Since Bluetooth networking required platform-specific tools, we opted for CoreBluetooth (Swift) for setting up the Bluetooth server on macOS, while Bleak (Python) was used for the client-side implementation.

C. WiFi-based VLAN Communication Setup

1) Server Configuration (Computer 1): The server setup involved the following steps:

1) Creating a dedicated working directory containing the `serverWifi.py` script.
2) Activating the virtual environment:

```
conda activate socket
```

3) Running the server script:

```
python serverWifi.py
```

Upon execution, the server displayed the message indicating that it was waiting for client connections:

```python
def server_program():
    host = "0.0.0.0"
    port = 8080

    server_socket = socket.socket()
    server_socket.bind((host, port))
    server_socket.listen(5)

    print("Server waiting for connections...")
```

Fig. 1: WiFi Server initialized and awaiting connections.

2) Client Configuration (Computer 2): On the client side, a similar directory structure was created, containing the client script and a test image for transfer.

1) Ensuring the script `clientWifi.py` contained the correct server IP:

IP address	192.168.1.70
Router	192.168.1.254

Fig. 2: Server IP configuration in the client script.

2) Activating the virtual environment:

```
conda activate socket
```

3) Running the client script:

```
python clientWifi.py
```

Upon execution, the client displayed a set of available operations:

```
(socket) (base) daniel.nuno@MXTI1-H249MD6R clientWIFI % python clientWifi.py

Options:
1. List files on server
2. Download file from server
3. Upload file to server
4. Exit
Enter choice: 1
```

Fig. 3: Client operations menu.

A connection request from the client resulted in a successful handshake, as confirmed by the server's logs:

```
Server waiting for connections...
Connected to: ('192.168.1.76', 49552)
```

Fig. 4: Server confirming client connection.

To verify proper communication, the client requested the file list from the server:

Fig. 5: List of server files received by the client.

3) File Transfer via WiFi: For a practical demonstration, the client attempted to download the file `book.jpg`. The process was as follows:

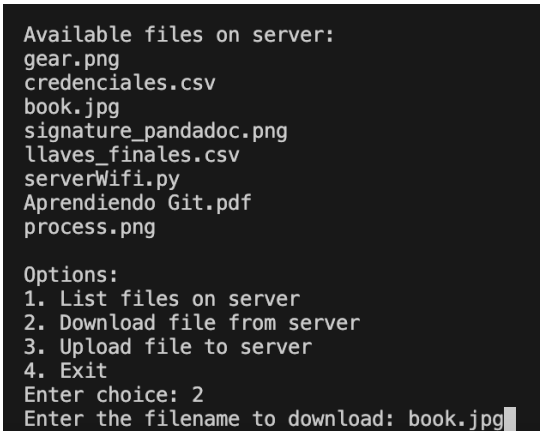

Fig. 6: Client requesting to download book.jpg from the server.

After execution, the server successfully transferred the file:

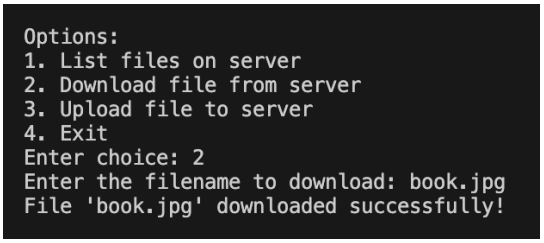

Fig. 7: Image successfully received by the client.

A directory listing on the client verified the presence of the newly downloaded file:

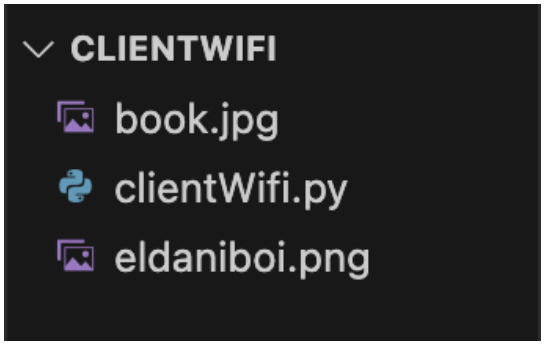

Fig. 8: Updated client directory after receiving the image.

4) Uploading Files from Client to Server: To test the upload function, the client sent the image `eldaniboi.png` to the server:

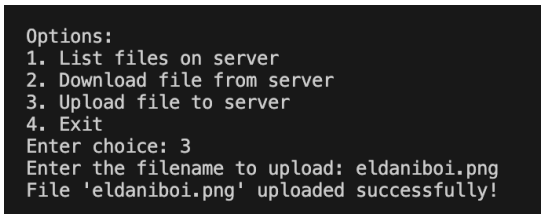

Fig. 9: Client initiating file upload.

A server-side directory listing confirmed that the file was successfully uploaded:

Fig. 10: Server directory after receiving the uploaded file.

D. Bluetooth-based VLAN Communication Setup

1) Server Configuration (Computer 1): Setting up the Bluetooth server required additional configurations due to macOS's strict permissions. The process included:

1) Running the Swift script to start the BLE server:

```
swift server.swift
```

2) Granting necessary Bluetooth permissions:

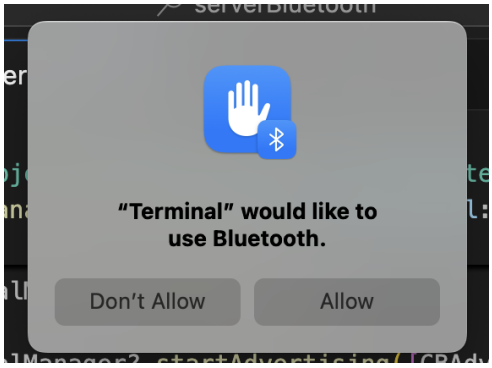

Fig. 11: Bluetooth permission request on macOS.

3) Confirming the server was broadcasting its BLE services:

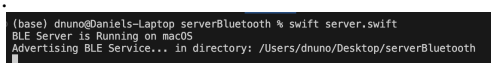

Fig. 12: Bluetooth server active and ready for client connections.

2) Client Configuration (Computer 2): On the client side, the following steps were performed:

1) Scanning for available BLE devices:

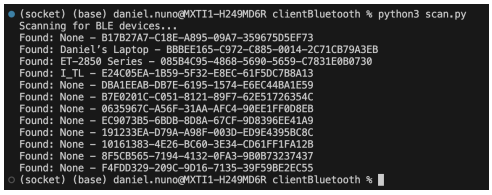

Fig. 13: BLE device scan results.

2) Identifying the correct server address and configuring the client script.

3) Running the client script to list available server files:

Fig. 14: List of files retrieved from the Bluetooth server.

3) Chunk-based File Transfer: Since Bluetooth does not natively support large file transfers, data was divided into chunks:

Fig. 15: File being transferred in chunks via Bluetooth.

A successfully downloaded file was prefixed with "downloaded_":

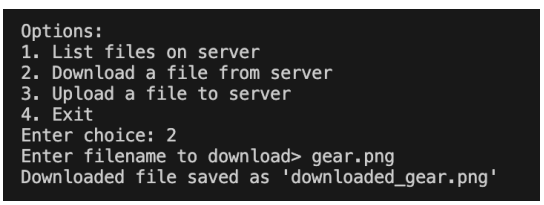

Fig. 16: Client successfully received the file over Bluetooth.

4) Uploading Files to the Bluetooth Server: The client uploaded an image to the server in chunks:

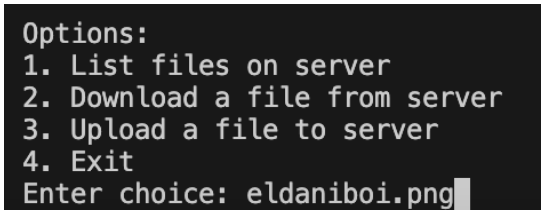

Fig. 17: Client initiating file upload via Bluetooth.

The process involved sequential chunk transfers:

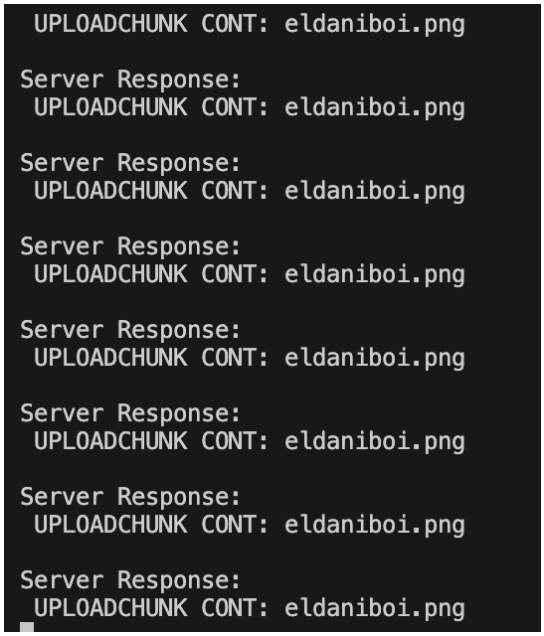

Fig. 18: Image file being divided into chunks for transfer.

On the server side, incoming chunks were received and reassembled:

Fig. 19: Server receiving file chunks from the client.

Finally, a directory listing confirmed that the file was successfully uploaded:

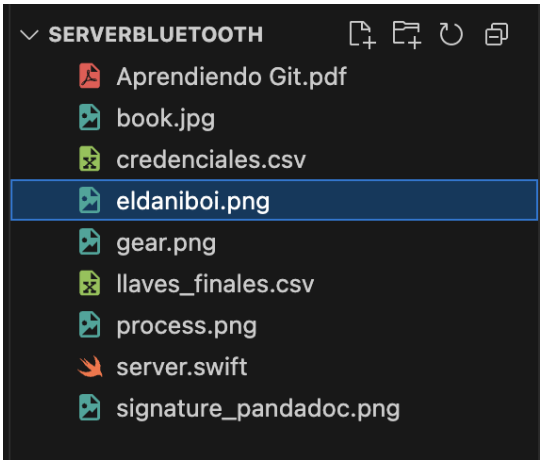

Fig. 20: Updated server directory after successful file upload.

E. Conclusion

This methodology successfully demonstrated VLAN communication over WiFi and Bluetooth. WiFi offered high-speed, reliable transfers, whereas Bluetooth required chunked transfers but proved feasible for small file exchanges.

V. Results

The results of this experiment demonstrated the feasibility and effectiveness of implementing VLAN communication via both WiFi and Bluetooth. Each approach had its advantages and constraints, which were thoroughly analyzed during the testing phase.

A. WiFi-based VLAN Communication

The WiFi implementation successfully established a client-server communication channel, allowing for seamless file exchanges. The process began with the client initiating a request to list the available files on the server. As shown in Figure 21, the client successfully retrieved the server's file directory.

Fig. 21: List of files available on the server through WiFi.

Next, the client requested to download an image from the server. The selection process is depicted in Figure 22, where the user chose to download "book.jpg." After execution, the server successfully transmitted the image to the client.

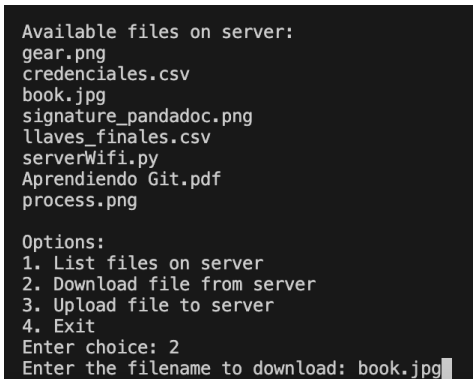

Fig. 22: Client selecting an image to download from the server.

Once the transfer was completed, the image appeared in the client's directory, confirming a successful download (see Figure 23).

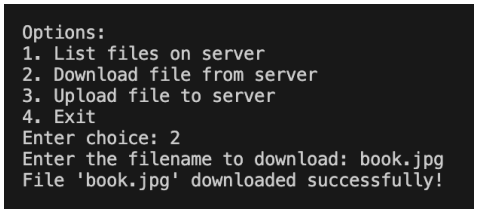

Fig. 23: Image successfully received by the client.

Furthermore, the upload functionality was tested by transferring "eldaniboi.png" from the client to the server. As illustrated in Figure 24, the process was initiated on the client side and confirmed by the server, which saved the uploaded file (Figure 25).

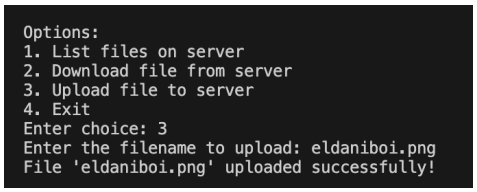

Fig. 24: Client uploading an image to the server.

Fig. 25: Server confirming the received image upload.

The efficiency of the WiFi VLAN was demonstrated through its high-speed file transfer capabilities and reliability in handling multiple client requests.

B. Bluetooth-based VLAN Communication

The Bluetooth VLAN was implemented as an alternative, particularly useful in scenarios where WiFi was unavailable.

Since Bluetooth Low Energy (BLE) has limitations in data transfer speed, a chunk-based transfer mechanism was developed to split files into smaller segments.

The first step involved scanning for available BLE devices and connecting to the server. Figure 26 confirms that the client successfully detected the BLE server and retrieved the list of available files.

Fig. 26: Client listing available files on the Bluetooth server.

A test file, "gear.png," was selected for download. The segmented transmission of this file ensured data integrity despite the lower bandwidth, as shown in Figure 27.

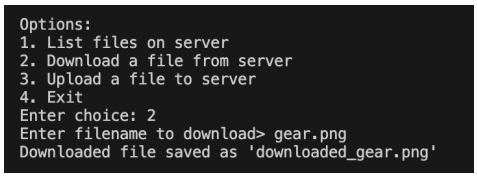

Fig. 27: File successfully downloaded from the Bluetooth server.

Similarly, the upload functionality was tested with the "eldaniboi.png" file. The client's successful upload confirmation is displayed in Figure 28.

Fig. 28: Client successfully uploaded an image to the Bluetooth server.

C. Comparison of WiFi and Bluetooth Implementations

Both implementations were successful in establishing VLAN communication, but their performance characteristics varied significantly:

- **WiFi:** Offered high-speed connectivity, supporting large file transfers with minimal latency.
- **Bluetooth:** Required chunk-based transfers due to lower bandwidth but proved effective for small to medium-sized file transfers.

D. Final Observations

While WiFi proved to be the more efficient solution in terms of speed and ease of implementation, the Bluetooth VLAN served as a viable alternative in network-restricted environments. The experiment successfully demonstrated the ability to create flexible, cross-platform VLAN communication using different networking technologies.

VI. Conclusions

This experiment successfully demonstrated the feasibility and effectiveness of VLAN communication using both WiFi and Bluetooth on macOS devices. The implementation and analysis of both methods provided valuable insights into their capabilities, limitations, and practical applications.

A. Performance Comparison and Network Efficiency

The WiFi-based VLAN implementation proved to be the most efficient in terms of data transfer speed, stability, and ease of implementation. It allowed high-speed communication between the client and server, handling file requests, downloads, and uploads with minimal latency. Given its reliance on existing infrastructure, WiFi is an ideal solution for most scenarios where network connectivity is available. The ability to list, request, and transmit files efficiently validated the reliability of Python's `socket` library in VLAN-based communication [1].

On the other hand, the Bluetooth-based VLAN required a more complex implementation due to the inherent limitations of Bluetooth Low Energy (BLE). The need for chunk-based data transmission was a critical adaptation to mitigate BLE's low bandwidth constraints. While this method successfully facilitated file transfers, it introduced higher latency and required additional processing on both the client and server sides. However, despite these constraints, the Bluetooth VLAN provided a viable alternative for scenarios where WiFi was unavailable, leveraging Apple's CoreBluetooth framework [2] and the Bleak library for cross-platform BLE communication [3].

B. Challenges and Key Learnings

Several challenges were encountered during the implementation:

- **Compatibility Considerations:** Developing the solution for macOS required careful selection of networking libraries, as certain widely used networking tools are optimized for Windows or Linux environments.
- **Security Aspects:** VLAN implementations must consider encryption and authentication mechanisms to ensure secure file transfers, especially over wireless connections.
- **Data Integrity in Bluetooth Transfers:** Given BLE's data packet limitations, additional mechanisms such as chunk verification were necessary to prevent data corruption.

Despite these challenges, the hybrid networking approach enabled a versatile communication framework capable of adapting to different environments.

C. Potential Applications and Future Work

The hybrid VLAN solution developed in this experiment has potential applications in several real-world scenarios:

- **IoT and Embedded Systems:** Bluetooth VLANs can be used in environments where devices lack traditional network connectivity but need to exchange small amounts of data.

- **Offline and Air-Gapped Environments:** Bluetooth communication serves as an alternative when conventional network access is restricted.
- **Cross-Platform Communication:** The integration of Python and Swift allows interoperability between multiple platforms, enhancing the versatility of the solution.

Future improvements could include optimizing the Bluetooth chunk transmission process to further reduce latency, implementing encryption protocols for secure transfers, and expanding the solution to support additional communication protocols.

D. Final Remarks

Ultimately, this experiment showcased the flexibility of VLAN implementations on macOS using different networking technologies. WiFi remains the preferred method for high-speed, stable communications, while Bluetooth serves as a complementary alternative where network infrastructure is limited. The successful execution of both approaches highlights the potential for hybrid networking solutions that leverage multiple protocols to enhance connectivity in diverse environments.

VII. Appendix: Source Code

The complete source code is available on GitHub:

https://github.com/dnuno10/
VLAN-Communication-using-WiFi-and-Bluetooth-on-macOS

References

[1] Python Documentation, ”Socket Programming”, Available: https://docs.python.org/3/library/socket.html.
[2] Apple Developer Documentation, ”CoreBluetooth Framework”, Available: https://developer.apple.com/documentation/corebluetooth.
[3] Bleak Library, ”BLE Communication in Python”, Available: https://github.com/hbldh/bleak.