Project Report
Members:

Kirby James Fernandez
Johnny Olswang
Joshua Reyes

The program comprises three main parts: master.py, node.py, and network_analyzer.py. Each serves a distinct function for this Small Distributed System project.

Master Component:

- The master.py script broadcasts messages to all network nodes using UDP over IP.
- Broadcasting ensures messages reach all nodes efficiently, without needing specific IP addresses.

Node Component:

- The node.py script listens for messages broadcasted by the master.
- It utilizes a UDP socket bound to all interfaces, listening on port 12345.
- This continuous listening aligns with a multicast model, allowing multiple recipients for a single message.

Network Analyzer Component:

- network_analyzer.py analyzes network packets using Scapy.
- It extracts packet details like type, source/destination IP, protocol, length, and flags.
- Useful for debugging, monitoring, or security analysis of network traffic.

Network Configuration and UFW:

- The network is configured to enable UFW and allow specific ports for traffic.
- UFW simplifies firewall management on Ubuntu systems.
- The firewall allows port 12345, ensuring seamless communication between master and nodes.

The master.py script is designed to broadcast messages to all nodes in a network. It uses the User Datagram Protocol (UDP) over the Internet Protocol (IP) to send messages. The script is structured as follows:

- broadcast_message(message): This function is responsible for broadcasting a message to all nodes. It creates a UDP socket and enables broadcasting by setting the socket option `SO_BROADCAST` to 1. It then broadcasts the user's message to all sockets bound to the target port (12345 in this case). The message is encoded before being sent, and a log entry is created to indicate that the message has been sent.
- Main Execution: The script's main execution block sets a message to "Hello, Nodes!" and calls the `broadcast_message` function with this message. This results in the message being broadcasted to all nodes in the network.

Node.py Explanation

The `node.py` script is designed to listen for and receive messages from the master server. It uses a threaded function to set up a UDP socket to receive messages. The script is structured as follows:

- receive_messages(): This function sets up a UDP socket and binds it to all available interfaces on port 12345, which matches the port used by the master server. It then enters a loop where it waits for messages. When a message is received, it decodes the message from bytes to a string, logs the received message, and prints it to the console. It also prints a task message indicating that it is processing the received message.
- Main Execution: The script's main execution block starts a new thread that runs the `receive_messages` function. This allows the script to listen for messages in the background while the main thread can perform other tasks.

Network Analyzer.py Explanation

The `network_analyzer.py` script is a utility for analyzing network packets using the Scapy library. It is designed to provide detailed insights into the network traffic. The script is structured as follows:

- analyze_packet(packet): This function takes a network packet as input and checks if it contains an IP layer. If it does, it extracts various details from the packet, such as the packet type, source and destination IP addresses, protocol, length, and flags. It then logs these details. If the packet does not contain an IP layer, a warning is logged.

Network Configuration for UFW and Port Allowance

The network configuration involves enabling UFW (Uncomplicated Firewall) and allowing specific ports for network traffic. This is crucial for ensuring that the messages broadcasted by the master can reach the nodes, and vice versa, without being blocked by the firewall. The specific port used by the master and node scripts (12345) is allowed through the firewall, enabling the interaction between the master and nodes.

Conclusion

The design of the program leverages broadcasting for the master to communicate with all nodes simultaneously. The use of UFW and specific port allowance ensures that the network traffic is not blocked, facilitating the interaction between the master and nodes. The network analyzer component provides a way to inspect the network packets, offering insights into the network

traffic. This setup demonstrates a practical approach to network communication and analysis, showcasing the use of Python for network programming and the importance of firewall configuration in network security.