Project 1 Report

Kirby Fernandez

Jonny Olswang

Joshua Reyes

Instructor: Hailu Xu

California State University, Long Beach

College of Engineering

CECS 327, Sec 02, Spring 2024

March 10, 2024

Video Demonstration: https://www.youtube.com/watch?v=cfxqBsm3fG4

master.py

run_server() - Server interface for users to construct cast commands and send messages.

1. Creates a UDP socket, enables broadcasting with `setsockopt()`, and sets the socket timeout for 0.5 seconds
2. Runs user interface to prompt which type of message they wish to send (broadcast, unicast, or close connection).
3. Prompts user for desired message to send alongside cast.
4. Calls function for chosen cast type and runs loop until user decides to close connection.
5. Close connection will shut down both the server and all the clients listening to the server.

broadcast_message(message) - Broadcast message to all nodes.

1. Broadcasts the user message to all sockets bound to the target port using built-in socket functionality.

unicast_message(message) - Send a message to a specified node.

1. Since the server does not have information on listening sockets, it sends the connection verification request as broadcast to all listening clients.
2. Receives all the response within the timeout, then stores and displays all the connected node addresses. We also run any verification messages received through the `anlyze_recieved_data` to save the info into a txt file for later. (function design is broken down further down in the doc)
3. User selects which node to send the unicast message to.

node.py

recieve_messages() - Threaded function that sets up a UDP socket to receive multiple types of messages for the client.

1. Creates a UDP socket.
2. Binds socket to all available interfaces on port 12345 to match the master server and waits for messages.

3. Any messages received are automatically run through the `analyze_recieved_messages` function to save the info in a txt file. (function design is broken down further down in the doc)
4. When a message is received, function checks for broadcast, unicast, closing request, or verification flags. Broadcast and unicast will record the message sent from the server, closing request will record the message and terminate the client, and verification will record the message and send back an arbitrary message, which the server will use to determine the client's address.

master.py and node.py

analyze_received_data(data, source_addr, recv_socket) - takes in data with the source address and receiving socket

1. Open the file in which we wish to save this info with with `open("/app/data/network_data.txt", 'a') as f:`
2. Decode the data by getting the first character in order to identify the type of message. Through if statements we then assign the cast_type value to match the flag.

Flag	Meaning
B	Broadcast
U	Unicast
V	Connection Verification/Confirmation (via Broadcast)
Q	Close/Quit Connection (via Broadcast)

3. From the source address we then extract the source ip from `source_addr[0]` and source port from `source_addr[1]`
4. Do the same to the socket getting `destination_ip` from `recv_socket.getsockname()[0]` and the destination port from `recv_socket.getsockname()[1]`.
5. Get the socket protocol from `recv_socket.type` checking for UDP (what we use), TCP, or Unknown.
6. Lastly, write down all the info we have gathered into the file we opened in a specific format so we can look at it later: `(f"Type: {cast_type} | Time: {time} | Source IP: {source_ip} | Destination IP: {destination_ip} | Source Port: {source_port} | Destination Port:`

```
{destination_port} | Protocol: {protocol} | Length: {length} |
Flags: {flags}\n")
```

Group Contributions

Kirby Fernandez (): Wrote broadcast protocol, and contributed to driver code of master.py and node.py. Wrote README.md file.

Jonny Olswang (): Contributed to unicast protocol and network analysis, and driver code of master.py and node.py. Contributed to the report and README.md file.

Joshua Reyes (): Contributed to unicast protocol and network analysis. Contributed to the report.