Computer Networking: Socket programming assignment

Sup. by Dr. Bazmi and implemented by Amirreza Yarahmadi

Introduction

This assignment is about creating a client and a server that talk to each other using sockets. The client sends a name and number to the server, and the server replies with its name and an integer number, showing both numbers and their sum. If the client sends a number outside the range, the server shuts down. The main purpose is to validate the data recieved from the server, the sum of the numbers that are logged in server and client should be the same.

The assignment folder structure looks like:

```
.
├── __socket_client
│   ├── api.py
│   ├── __init__.py
│   ├── logger.py
├── __socket_server
│   ├── api.py
│   ├── __init__.py
│   ├── logger.py
├── start_socket_client.py
└── start_socket_server.py
```

The `__socket_client` and `__socket_server` packages that contains the `api.py` files for starting and handling the socket server and client. The `logger.py` file contain the custom utils needed for logging the output. The `start_socket_client.py` and `start_socket_server.py` files as those sounds starts the client and server.

Next I will explain the details of this program.

Server Side:

The server waits for the client to connect. When the client connects, the server gets the client's name and a number. If the number is between 1 and 100, the server picks a random number and sends it back to the client, along with the server's name. The server also sends the sum of both numbers. If the client sends a number that is too big or too small, the server stops and closes the connection. Also it print logs for the user to see behind of the proccess. The logs that are indicated as "DEBUG" are the programming logs (ex. The server started on 0.0.0.0:8000) and "INFO" logs are the main output of these program. The server starts by running `start_socket_server.py` file. The logged output of this program after interacting with user looks like:

```
[INFO] [start_server]: Amirreza Yarahmadi Server started and listening on
0.0.0.0:8000
[DEBUG] [start_server]: Connection accepted from ('127.0.0.1', 48528)
[DEBUG] [handle_client]: Message recieved from client
[INFO] [handle_client]: Client Name: Amirreza Yarahmadi Client | Server Name: Amir
reza Yarahmadi Server
[INFO] [handle_client]: Client Number: 25 | Server Number: 10 | Sum: 35
[DEBUG] [handle_client]: Sending client response
[DEBUG] [handle_client]: Client connection closed
[DEBUG] [start_server]: Connection accepted from ('127.0.0.1', 40484)
[DEBUG] [handle_client]: Message recieved from client
[WARNING] [handle_client]: Client int out of range: 0. Terminating the server.
```

As this log shows, a client with IP and Port `127.0.0.1:48528` created a connection with the server `0.0.0.0:8000` . The client sent its name as `Amirreza Yarahmadi Client` and its integer value `25` . The server logged its name with client name as `Client Name: Amirreza Yarahmadi Client | Server Name: Amirreza Yarahmadi Server` and its int value, the client int value and sum of those as `Client Number: 25 | Server Number: 10 | Sum: 35` . Then as supposed the client connection closed. Then another client `127.0.0.1:40484` connected to the server and submited a value of `0` which is out of desired range `[1, 100]` . So as desired the server terminated itself.

Client Side:

The client asks the user to enter a number, connects to the server, and sends its name and the number. The client then waits for the server's response, which has the server's name, number, and the sum of both numbers. If the number is not between 1 and 100, the client stops and exits. The printed logs are the same as the server logs. A logged interaction with server looks like:

```
Enter an integer between 1 and 100: 25
[DEBUG] [send_message]: Connected to server at 0.0.0.0:8000
[INFO] [send_message]: Amirreza Yarahmadi Client | Amirreza Yarahmadi Server
[INFO] [send_message]: Client Number: 25 | Server Number: 10 | Sum: 35
```

This is the default connection of client and a value inside `[1, 100]` and The server respond according to the client submition. The bellow happens when the user enter an int value out of the desired range:

```
Enter an integer between 1 and 100: 0
[DEBUG] [send_message]: Connected to server at 0.0.0.0:8000
[WARNING] [send_message]: The integer input out of range. Terminating the server
```

Summary:

This program is about how the client and server talk to each other using sockets. The server checks the number from the client and sends back a response. If anything is wrong, the server or client will stop. This program is a demo of the conversation between client and server and also validating the values transmitted.

---

amirreza-yar | yar.amirreza@gmail.com