ID:12214731   
NAME: Diyora Bobokulova

Introduction

This report shows the implementation process of a Python-based chat application, along with enhancements for Round Trip Time (RTT) measurement, the integration of simulated sensor data and Packet Loss Handling. The chat application serves as a basic client-server model where clients can send messages and receive messages from other clients, as well as receive sensor data from the server.

**Implementation Steps**

1. Server Setup

The server-side implementation is encapsulated in the `server.py` script. The server is responsible for:

- Creating a socket and setting up the server to listen for incoming connections.

- Handling new client connections.

- Receiving and broadcasting messages.

- Measuring RTT for messages.

- Simulating sensor data and sending it to connected clients.

2. Client Setup

The client-side implementation is encapsulated in the `client.py` script. The client is responsible for:

- Creating a socket and connecting to the server.

- Sending messages to the server.

- Receiving and displaying messages from the server.

- Measuring RTT for messages.

- Receiving sensor data from the server.

3. RTT Measurement

Both the server and client have been modified to measure the RTT for messages:

- **On the server side**, the RTT measurement is performed just before broadcasting a message to clients. The time at which the message is sent is recorded, and the time it is received by the client is used to calculate the RTT, which is then displayed.

- **On the client side,** after sending a message, the client listens for an acknowledgment from the server. The time at which the acknowledgment is received is used to calculate the RTT, which is then displayed.

**4. Data Integration**

To simulate sensor data, both the server and client have been enhanced with the following functionality:

- Integrated enclosed wheel rotation sensor data from an Excel file with the name “data\_xsl”

- On the server side, a `sensor\_data` function has been introduced, which generates simulated sensor data. This data includes a timestamp and a numeric value (e.g., time-based value). The server sends this sensor data to connected clients.

- On the client side, the client receives and displays the sensor data received from the server.

**5. Packet Loss Handling**

Packet loss is addressed using error handling with **try** and **except** blocks. The server captures client disconnections and exceptions, removing disconnected clients from the list of connected clients. ‘errno’ function was imported in both client and server files in order to handle the errors.

**Output and Usage**

When the server and client scripts are executed, the following functionality is observed:

- The server listens for incoming connections and accepts clients.

- Clients can send messages to the server, and the server broadcasts these messages to all connected clients.

- RTT for each message is measured and displayed.

- Sensor data is periodically generated and sent from the server to clients.

- Clients can view the chat messages and receive and display sensor data.

**Conclusion**

The implementation of the chat application, with RTT measurement, simulated sensor data, Packet loss handling, demonstrates the bidirectional communication between clients and the server. This application can be further extended to accommodate real-time sensor data or additional features based on specific use cases. The combination of chat, RTT measurement, and sensor data integration showcases the potential for creating comprehensive networked applications.