T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE 4074 Programming Assignment

Remote Sensing Application

Group Members

Yağmur Koçoğlu- 150119715

Abdullah Enes Dizer- 150119880

Tarık Işın-150120526

Instructor

Doç.Dr. ÖMER KORÇAK

Project Summary

This Java project, named SensorHandler, simulates a sensor system consisting of temperature and humidity sensors. It communicates with a gateway using both TCP and UDP protocols to collect and process sensor data. The project also includes a basic server that serves HTML pages to display the collected sensor data. The system runs on a multi-threaded architecture, allowing concurrent operation of various components.

Solution Approach

- Sensor Simulation:

Simulated temperature and humidity sensors generate random data at regular intervals. Temperature data is sent over TCP to the gateway. Humidity data is sent over UDP to the gateway, with additional logic to filter data based on a threshold.

Temperature Sensor Application: Generates random values between 20-30°C every second and sends them to the gateway via TCP, along with a timestamp.

Humidity Sensor Application: Generates random values between 40-90% every second. It sends data to the gateway via UDP only if the value exceeds 80%. Additionally, it sends an 'ALIVE' message every 3 seconds.

- Gateway:

The gateway listens for incoming TCP connections from the temperature sensor and UDP packets from the humidity sensor. It processes and logs incoming sensor data, keeping track of the last humidity value. The gateway also handles client requests through a basic server.

- Server:

A simple server serves HTML pages for temperature and humidity data, as well as the last humidity value. The server runs on a separate thread and handles client requests concurrently.

Encountered Problems and Solutions

- Logging:

Configured Java's logging system to log temperature, humidity, and alive messages separately. Solved by setting up different loggers and handlers for each type of data.

- Concurrency:

Implemented multi-threading to handle concurrent operations of temperature and humidity sensors, the gateway, server, and the alive message sender. Ensured proper synchronization to avoid race conditions.

- Protocol Implementation:

Defined a simple protocol for communication between the sensors and the gateway (e.g., "TEMPERATURE|value|timestamp"). Ensured proper data parsing and handling in both the sensors and the gateway.

- Ensuring Reliable Communication:

Implemented a handshake protocol for reliable communication between the server and gateway.

Unresolved Issues

- Getting Sensor Off Messages:

We cannot take any Off messages when sensors off.

- "Gateway - TEMP SENSOR OFF"

- "Gateway - HUMIDITY SENSOR OFF"

Usage Explanation

- Running the System:

Compile and run the SensorHandler class. The system starts the temperature and humidity sensors, the gateway, the server, and the alive message sender concurrently.

- Accessing Sensor Data:

Access sensor data through the server by visiting:/temperature for temperature data, /humidity for humidity data, /gethumidity for the last humidity value.

- Interaction with Sensors:

The system simulates the continuous generation of sensor data until the program is terminated.

- Advanced Data Visualization:

The web interface currently displays data in a simple text format. Implementing graphical representations of data could enhance user experience.

- Scalability and Performance:

The current setup might not efficiently handle a significantly higher number of sensor data streams or web requests.

For code readability and maintainability, each module (temperature sensor, humidity sensor, gateway, server) can be organized same class. More detailed and understandable information can be provided for log files and messages displayed on the screen.

- Viewing Data:

Access 'http://localhost:8080/temperature' for temperature data and 'http://localhost:8080/humidity' for humidity data.

- Requesting Last Humidity Value:

Access 'http://localhost:8080/gethumidity' to get the last measured humidity value.

Protocol Details

- Handshake Protocol:

Ensures reliable initial communication between the gateway and server.

- Temperature Sensor (TCP):

Protocol: TEMPERATURE|value|timestamp

- Humidity Sensor (UDP):

Protocol: HUMIDITY|value|timestamp

- Alive Message (UDP):

Protocol: ALIVE|timestamp

Logging and Display

- All processes display sent and received messages on the screen.
- Log files are maintained for each component, recording these messages for review.

Note: This project provides a foundational framework for a networked sensor system, demonstrating key concepts in network programming, data management, and web server implementation. It offers valuable insights into handling different communication protocols and managing sensor data in a simulated IoT environment.

Handshake Protocol

Purpose: Establishes a reliable connection between the server and the gateway.

Process:

1. **Connection Initiation:** The gateway opens a TCP connection to the server.

2. **Handshake Request:** The gateway sends a "HANDSHAKE_REQUEST" message.

3. **Handshake Acknowledgment:** The server, upon receiving this message, responds with "HANDSHAKE_ACCEPTED".

4. **Confirmation and Protocol Start:** The gateway, upon receiving the acknowledgment, confirms a successful handshake and begins regular data transmission.