Machine Learning for IoT - Report Homework 3

Giuseppe Acquaviva, Mario Capobianco, Marco Mungai Coppolino
Politecnico di Torino, Italy

I. Exercise 1 - Data Collection, Communication, and Storage

A. Implementation of Data Collection using MQTT

The primary goal of this task was to collect temperature and humidity data using a DHT-11 sensor connected to a Raspberry Pi and publish the data to an MQTT broker. The following steps were performed:

- The script `publisher.py` was developed to measure and publish data every 2 seconds.
- JSON messages were structured with fields: MAC address, timestamp, temperature, and humidity.
- The Eclipse MQTT broker was used as a communication endpoint.

B. Data Storage using Redis

To store the data:

- An MQTT subscriber was implemented in `subscriber.ipynb`.
- Data was stored in Redis TimeSeries for efficient retrieval and management.

C. MQTT vs REST for IoT Applications

MQTT is a better choice than REST for this application because:

- **Efficient for frequent updates**: MQTT's publish/subscribe model eliminates the need for continuous client polling, reducing communication overhead.
- **Lightweight**: MQTT requires minimal resources, making it ideal for constrained IoT devices like the Raspberry Pi and DHT-11 sensor.
- **Real-time performance**: MQTT ensures faster delivery of data compared to REST, which involves higher latency due to its request-response model.

In contrast, REST is better suited for less frequent interactions and when resources are not constrained.

II. Exercise 2 - Data Management & Visualization

A. REST API Design

A REST API was implemented in `rest_server.ipynb` to enable retrieval of historical temperature and humidity data. The endpoint `/data/{mac_address}` was designed with:

- HTTP `GET` method for data retrieval.
- Query parameters to specify a date range.
- Response codes for error handling and data availability.

B. REST Client for Data Visualization

A REST client was developed in `rest_client.ipynb` to interact with the API:

- Server status was verified using `GET /status`.
- Sensor node was added using `POST /sensors`.
- Data was retrieved and visualized with Deepnote chart blocks.

C. HTTP Method Justification

The most suitable HTTP method for implementing the `/data/{mac_address}` endpoint is **GET** because:

- **Read-only operation**: GET is designed for retrieving data without altering server resources, aligning with the endpoint's purpose.
- **Idempotency**: Multiple GET requests with the same parameters will yield the same response, ensuring consistent and predictable behavior.
- **Unsuitability of other methods**: `POST` is typically used to create resources, while `PUT` and `DELETE` involve modifying or deleting resources, making them inappropriate for data retrieval.

III. Conclusions

The tasks highlighted the importance of MQTT for real-time IoT communication and REST for structured data retrieval. The implemented solutions met the project requirements and demonstrated efficient data management and visualization.