Assignment 8: Build an End-to-End IoT System Report

Title:
Building an End-to-End IoT System

Course:
CECS 327: Introduction to Networking and Distributed Systems

Group Number 3:

- Member 1: Medhanie Meshesha, 028018976, medhanie.meshesha01@student.csulb.edu
- Member 2: Ian Lee, 018778017, ian.lee@student.csulb.edu

Submission Date:
November 21, 2024

- GitHub Link: https://github.com/peacemeda/CECS327_Assignment8_gr3.git

---

1. Introduction

The goal of this project was to design and implement an **End-to-End IoT System** that integrates IoT sensors, a TCP client-server architecture, and a MongoDB database to process and analyze user queries. Using metadata from Dataniz, we enhanced the system's functionality for flexibility and accuracy. This assignment extended our previous work on IoT devices and client-server communication. We used the MQTT communication protocol.

Key Objectives:

1. Integrate IoT sensors, metadata, and databases into a unified system.
2. Enhance TCP client-server communication to handle real-world IoT queries.
3. Utilize metadata for device management and query processing.
4. Perform unit conversions and data analysis to meet user requirements.
5. Demonstrate hands-on experience with cloud databases and system integration.

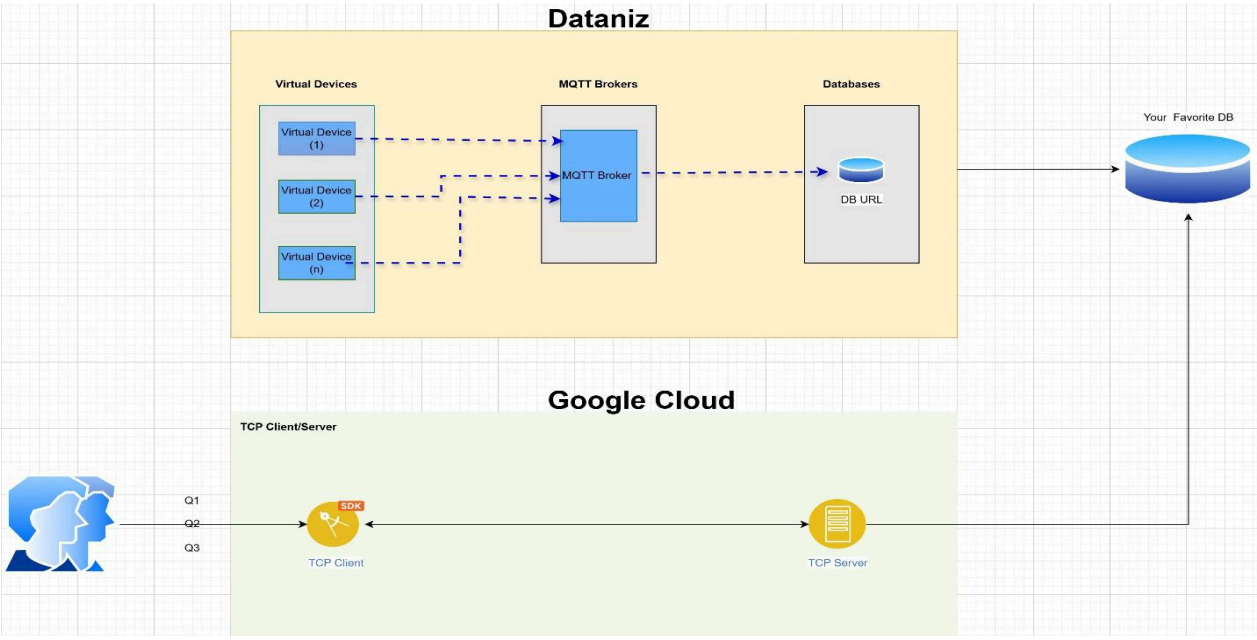
---

2. System Architecture

Overview: The system includes:

1. **TCP Client**:
    - Handles user queries and communicates with the server.
    - Sends queries such as:
        - Average moisture inside the fridge in the kitchen(past 3 hours).
        - Average water consumption per dishwasher cycle.
        - Device with the highest electricity consumption.
2. **TCP Server**:
    - Retrieves data from the `table1_virtual` collection in MongoDB.
    - Uses metadata from `table1_metadata` for device identification and data interpretation.
    - Performs calculations.
3. **MongoDB Atlas Database**:
    - **`table1_metadata`**: Stores metadata for devices (e.g., `SmartFridge`, `SmartWasher`).
    - **`table1_virtual`**: Stores sensor data (e.g., `Moisture Meter1`, `Ammeter3`).

Diagram:



3. IoT Sensor Research

Sensors Used:

1. **Moisture Meters**:
    - **Type of Data**: Measures internal fridge moisture levels.
    - **Unit**: RH% (Relative Humidity).
    - **Precision**: ±2% RH.
    - **Time Zone**: Configured for PST using metadata.
2. **Water Sensor**:
    - **Type of Data**: Records water usage per dishwasher cycle.
    - **Unit**: Gallons.
    - **Precision**: ±0.5 gallons.
    - **Time Zone**: Adjusted to PST using metadata.
3. **Ammeter (Current Sensor)**:
    - **Type of Data**: Tracks electricity usage of all devices.
    - **Unit**: Kilowatt-hours (kWh).
    - **Precision**: ±0.1 kWh.
    - **Time Zone**: Converted to PST.

---

4. Implementation Details

TCP Client:

- Sends predefined queries to the server.
- Rejects invalid queries with a user-friendly message.

TCP Server:

- Connects to MongoDB Atlas.
- Processes the following queries:
    1. **Average Moisture (Fridges)**:
        - Retrieves data for `SmartFridge1` from `table1_metadata`.
        - Uses `Moisture Meter1` value from `table1_virtual` for the past 3 hours.
        - Calculates and returns the average moisture.
    2. **Average Water Consumption (Dishwasher)**:
        - Uses the `WaterSensor` field in `table1_virtual` for `SmartWasher` data.
        - Calculates average water consumption per cycle.
    3. **Electricity Consumption**:
        - Uses three `Ammeter` values to compute electricity usage for all devices.
        - Returns the device with the highest consumption.

Database:

- `table1_metadata`:
  - Metadata fields include `assetUid`, `customAttributes.name`, and device types.
- `table1_virtual`:
  - Sensor fields include `Moisture Meter1`, `Moisture Meter2`, `WaterSensor`, and `Ammeter3`.

---

5. Challenges and Solutions

1. **Challenge**: Aggregating moisture data for the fridge in the kitchen.
   - **Solution**: Used `parent_asset_uid` to link `table1_virtual` data to metadata in `table1_metadata`.
2. **Challenge**: Handling inconsistent sensor data.
   - **Solution**: Filtered out entries missing required sensor fields.
3. **Challenge**: Converting timestamps to PST.
   - **Solution**: Used Python's `datetime` module to adjust all timestamps.

---

6. Metadata Usage

Metadata Integration:

- **Device Identification**:
  - `assetUid` links each device's metadata to its sensor data in `table1_virtual`.
- **Unit Standardization**:
  - `customAttributes.name` and device type ensure consistent interpretation of sensor values.

Justification:

- Metadata simplifies device management and ensures accurate query results.
- Without metadata, distinguishing devices and interpreting data would be error-prone.

---

7. Feedback for Dataniz

Strengths:

- The platform's metadata capabilities simplify IoT device management.

- Provides realistic virtual devices for testing.

Suggestions for Improvement:

1. Add built-in unit conversion for common sensor types.
2. Include a time zone configuration interface for data consistency.
3. Provide export options for virtual data directly to database formats like JSON or CSV.

---

8. Conclusion

At the end this end to end IoT project was a very inciting and challenging project. When implementing this MQTT protocol project it looks more of the same with the real world. Even Though we didn't deployed to web server it has everything to be near to the real world application.This project successfully implemented an end-to-end IoT system that:

1. Integrated IoT sensors, a TCP client-server architecture, and a MongoDB database.
2. Processed user queries efficiently using metadata and real-time sensor data.
3. Provided accurate results with appropriate unit conversions.

Future Work:

- Add real-time data visualization via dashboards.
- Extend support for more IoT devices.