Design and Implement a Simple Architecture for AI-based Energy Management Systems

1 INTRODUCTION

An **energy management system (EMS)** is a system of computer-aided tools to monitor, control, and optimize energy use as its name implies [1]. One of the applications is the building EMS (BEMS) which monitors and controls energy-powered services, such as air-conditioning or heating, to ensure that a building is efficiently operating while removing wasted energy usage and associated costs [6]. BEMS is necessary as buildings are responsible for more than 40% of global energy use and 1/3 of global greenhouse gases [2].

To make EMS intelligent, there have been several approaches that leverage AI techniques [3]. For the success of AI-based EMS, which may highly rely on high-quality data, EMS can be integrated with *Advanced Metering Infrastructure* (AMI) [8], which is to remotely collect customer usage data in real-time without the need for manual meter reads, or *Supervisory Control and Data Acquisition* (SCADA) [5], which is the gathering of data in real-time from remote locations in order to control equipment and conditions. Simply put, their data collection (or data acquisition) typically relies on the IoT-cloud model, where submeters measure the energy usage and send the collected data to the AMI server on the cloud.

Recently, the notion of edge computing [4, 7] is introduced to make servers closer to users or devices for fast response, privacy, and so on. Usually, edge computing is implemented with an architecture that consists of multiple layers including the Internet-of-Things (IoT) devices, edge devices, and cloud servers. Thanks to the closer distance between an edge device and an IoT device, servers on edge devices can quickly respond to requests, while wireless devices can get into sleep mode faster; thus, it reduces energy consumption. Furthermore, edge devices can aggregate data received from IoT devices and report only the resultant value to cloud servers. It may reduce the number of bytes that cloud servers receive; thus, the possibilities of bottlenecks decrease. Also, it is useful to preserve privacy as edge devices can selectively send data to the cloud servers after aggregating the reported data from sensors. In other words, edge devices can monitor data from IoT devices, and eliminate sensitive information from the data.

Applying the notion of edge computing to EMS, in this project, we consider one EMS company (say, ECOM) that provides energy management service leveraging three-tier architecture. ECOM uses the cloud platform (e.g., Amazon Web Service) to run their EMS servers, parts of which include an AMI server and a SCADA module. The ECOM runs one AI module that forecasts the energy usage of each property to optimize energy usage at a certain point in the future. ECOM sells an edge device that collects the data from the attached sensors, aggregates the data, and sends the data to the cloud server. We assume that there are a bunch of sensors (e.g., 50K devices) in each property of buildings and all are compatible with the edge device in communication. Examples of the sensors are temperature sensors, energy submeters, and humidity sensors. The sensors periodically (e.g., 5 minutes) send the data to an edge device (or an aggregator) installed in the building. Then, the edge device regularly (e.g., 1 day) aggregates the collected data to the cloud server.

You are required to do a team project of designing and implementing a simple architecture for AI-based EMS. The project aims to (1) understand how an AI-based architecture works and (2) how to implement the system. To achieve the goals, you should work together as a team of four menbers.

Task #	Description
1	Design and implement a function that aggregates data and generates a vector instance
2	Design and implement a simple protocol between an edge device and a control server
3	Implement communicating functions between the control server and the AI module
A1	Redesign and implement the structure of the dataset classes
A2	Understand the given implementation

Table 1: A list of tasks. You can get one additional score.

Upon completion of your project, you will demonstrate your implementation between **June 11st** and **June 12nd**. *Tasks are detailed in the subsequent sections* (see section 2 and section 3). You are kindly advised to read through the rest of this material and complete your team project on time.

Data	Plan
May 30th	Development environment setting
June 2nd – June 4th	Project discussion
June 10th	Q&A 1 (Simple practice)
June 11st	Q&A 2 (Simple practice)
June 11st – June 12nd	Demonstration with a simple interview

Table 2: Time-table for the project

Each team member in your team should complete one task (to be detailed in the next section) to implement the system. This project requires your team to be a team leader, a system architect, a protocol designer, an embedded system developer, a server developer, and an AI expert in this project. As a team leader, you should organize your time schedule to complete this project. As a system architect, you should understand how a system works. (There is no outcome for this project as a team leader and a system architect.) As an embedded system developer, you should optimize the code running on the edge device to be lightweight. As a protocol designer, you should design a network protocol executed between the edge device and the cloud server. Also, you should implement the protocol as an embedded system developer and a server developer. As an AI expert, you should consider good features to make an AI model useful for our energy management.

2 GOALS

In this project, we are going to design and implement a simple EMS architecture in which an AI model is used to predict the power usage of each property of buildings in the next time step.

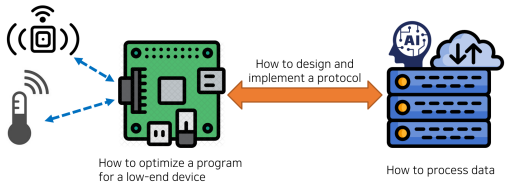

Figure 1: An architecture assumed in our project

You will be given a Raspberry Pi as an edge device and a server as a cloud server. We assume that there are multiple sensors that report their sensor values to the edge device. The sensor values may contain sensitive data such as the name of the owner, the location, and so on. On every given period, the edge device aggregates values and masks sensitive information, and sends the resultant values to the server.

Data aggregation: Typically, edge devices are intermediaries between IoT devices and cloud servers. The edge devices are usually introduced to reduce the number of bytes exchanged with the cloud servers to address the bandwidth problem of the cloud servers, and to avoid privacy issues that may arise due to the sensitive information exposure from the IoT devices. To this end, data aggregation techniques are usually used. You will be given an application that receives sensor values with sensitive information but inefficiently and undesirably processes them before sending the data to servers. Our task is to process the data so that two goals – reducing the number of bytes to be sent to the cloud servers and addressing the privacy issues – are achieved. To this end, you should answer the following questions:

- What aggregation method would be good to achieve the two goals?
- What aggregation method would be good to improve our prediction AI model?

Design and implementation of a protocol: As edge devices are resource-constrained and sometimes rely on wireless communication that consumes lots of power, the protocol executed between the devices and the cloud server should be lightweight. For instance, the bytes that the devices must send in reporting should be small. Also, due to the requirement for privacy, sensitive data from sensor data should be removed. Your goal is to design and implement a lightweight protocol, and to argue why the designed protocol is lightweight in what aspects. To this end, you should answer the following questions:

- What is 'lightweight'?
- Why does your protocol is lightweight?

Improvement of an AI system: As AI is a data-driven approach, its performance highly relies on the quality of data and the selected features, which may have a trade-off relationship in this project. You will be given interfaces for the AI module to generate a prediction model, test it, and get the performance result of the model.

- What features are good for our AI model?
- How do you aggregate the data?

3 TODO

The final outcome of this project include 1) the *final interview* asking some questions such as i) how you design and implement the aggregation function, ii) how you design and implement a protocol between an edge device and a control server, iii) how you communicate with the AI module from your control server, and iv) your analysis on the AI model.

Below are the tasks that you should do to complete this project. The path is based on the structure of the initial project files, which you can get from https://github.com/hw5773/cos-project-2.git. Each member of your team should be assigned one task and report what and why you did on the interview. The demonstration and the interview will be on May 31st.

Task #1. Design and implement a function that aggregates data and generates a vector instance (one person). You are given an application for an edge device that receives around 50K sensor data every day. In the first version of the implementation, the edge device reports all the information from the collected data, not only the power consumption value, the temperature, and the humidity, but also private information such as the owner's name, the address, and the serial number in plaintext. Your role is to reduce the size of the values to be reported to the control server without disclosing private information. For instance, you may aggregate the sensor values to the averaged value and report it with the timestamp and the humidity value, or you can only report the maximum (or the minimum) value to the control server. Please implement at least three types of functions and report why you select those functions, how you implement them.

> You should implement a `processData()` function of a process manager of an edge device, which you can find in `process_manager.cpp` in the directory of `edge`.
> You can practice how to do this task by implementing `tests/test_process.cpp`.

Task #2. Design and implement a simple protocol between an edge device and a control server (one for the client and the other for the server). You are going to design and implement a protocol between an edge device and a data server. With the result of task #2, you are required to send the data to the control server from the edge device. Design a simple and lightweight protocol and implement the protocol. Please try to reduce the number of bytes exchanged between the edge device and the control server. You are required to write down your approach how you encode each instance vector and your strategy to reduce the number of bytes.

Criteria	2	1	0
Level of responsibility in project	Each student proactively contributes to the project by having one's own task	Some students mostly contribute to the project	Some students never contribute to the project
Number of aggregation functions (Task #2)	A team implements at least three types of aggregation functions and reports their idea	A team implements two types of aggregation functions and reports their idea	A team implements less than two types of aggregation functions and report their idea
Design and implementation of the protocol (Task #3)	A team specifies a protocol in detail and in an understandable way (e.g., using a diagram) with the design rationale, and implements the protocol	A team roughly specifies a protocol and implements the protocol	A team does not specify a protocol or implement the protocol
Implementation of the communication with the AI module (Task #4)	A team implements the communication with the AI module working correctly without any error, asking at least one additional feature	A team implements the communication with the AI module working correctly	A team does not implement the communication with the AI module
Interview	A team provides appropriate answers for all the questions	A team somehow provides appropriate answers for the questions	A team does not provide any appropriate answers for the questions
Refactoring the structure of the dataset (optional) (Additional Task #1)	-	A team redesigns the structure of the dataset using the inheritance feature of C++ without modifying the interfaces for the main application	A team incorrectly redesigns the structure of the dataset
Understanding the code (optional) (Additional Task #2)	-	A team inserts line-by-line comments that describe the behavior of the edge device in the C++ code	A team does not inserts comments in the code

Table 3: Evaluation Table. You will have the minimum score of 20 and have scores based on how you achieve your tasks. The additional point is limited to one. That is even if you do both additional tasks, you will have only one point for it.

For a client, you should implement `sendData()` and `receiveCommand()` functions of a network manager of an edge device, which you can find in `network_manager.cpp` in the directory of `edge`.
For a server, you should implement a `handler()` function of a control server, which you can find in `server.py` in the directory of `server`.
You may practice how to implement the protocol by completing the practice about implementing the simple protocol. You can do it by implementing `tests/test_server.py` and `tests/test_client.cpp`. Note that you can make a binary executable of a client by using the command `make`.

Task #3. Implement communicating functions between the control server and the AI module (one person) You are required to communicate with the AI module through HTTP/JSON communication. The AI module provides the REST API so that you can use these APIs to send the dataset and generate the model. You can ask the instructor to add some models, features, or functions to the AI module. Please report what you did and how you specify the instance vector.

You should leverage functions provided by a `requests` package of Python to generate the HTTP/JSON messages to communicate with the AI module. You should insert functions in the `handler()` function of a control server, which you can find in `server.py` in the directory of `server`.
You can practice how to do by completing the practice of a control server (`test_control_server.py`), which you can find in the directory of `tests`).

Additional Task #1 (optional). Redesign and implement the class structure of the dataset. You are given an application for edge devices written in C++. The data receiver of the edge device gets the dataset that consists of three types of data including the temperature, the humidity, and the power (and the timestamp). The classes of these data have common features so it can make a hierarchy. Please redesign the classes of these data so that the common features are managed in one class, without modifying the interfaces used in the main application.

You should read through the header files (`dataset.h`, `house_data.h`, `humidity_data.h`, `info.h`, `power_data.h`, and `temperature_data.h`) in the directory of `data`. Then, you can use the inheritance feature of C++ to restructure them.

Additional Task #2 (optional). Understand source codes. Your task is to understand source codes of our project and summarize how following functions work on the interview:

- `protocol_execution()` in tests/test_client_answer.cpp
- `protocol_execution()` in tests/test_server_answer.py
- `main()` in tests/test_process_answer.cpp
- `processData()` in edge/process_manager.cpp (original version)
- `sendData()` in edge/network_manager.cpp (original version)
- `receiveCommand()` in edge/network_manager.cpp (original version)
- `edge/byte_op.h`
- `connecter()` in server/server.py (original version)
- `listener()` in server/server.py (original version)
- `handler()` in server/server.py (original version)

4 FINAL INTERVIEW AND EVALUATION

On May 31st, we should demonstrate your implementation. You are going to have an interview to ask for the questioins such as 1) how tasks are distributed to each team member, and 2) what each team member does to complete the task assigned. Please refer to the Table 3 to see how the project will be evaluated.

REFERENCES

[1] Charles Arthur. 2021. What is an energy management system? https://www.unido.org/stories/what-energy-management-system. (Accessed on 03/10/2023).
[2] Jane Feeney. 2016. Building Technologies: Who's Who? https://www.dexma.com/blog-en/building-technologies-whos/. (Accessed on 03/10/2023).
[3] Dasheng Lee and Chin-Chi Cheng. 2016. Energy savings by energy management systems: A review. *Renewable and Sustainable Energy Reviews* 56 (2016), 760–777.
[4] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. 2015. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review* 45, 5 (2015), 37–42.

[5] Peter Loshin. [n. d.]. SCADA (supervisory control and data acquisition. https://www.techtarget.com/whatis/definition/SCADA-supervisory-control-and-data-acquisition. (Accessed on 03/10/2023).
[6] Optimised. 2020. What is a BEMS? Building Energy Management Systems Explained. https://www.optimised.net/what-is-a-bems. (Accessed on 03/10/2023).
[7] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge computing: Vision and challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646.
[8] Cristina Tuser. 2022. What is Advanced Metering Infrastructure (AMI)? https://www.wwdmag.com/editorial-topical/what-is-articles/article/10940067/what-is-advanced-metering-infrastructure-ami. (Accessed on 02/25/2023).