International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 16, No.1 (Aug-24)

https://dx.doi.org/10.12785/ijcds/160162

Microservices for Asset Tracking Based on Indoor Positioning System

Dondi Sasmita[1] **and Gede Putra Kusuma**[2]

[1,2]*Computer Science Department, BINUS Graduate Program – Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia. 11480*

Received 06 Feb. 2024, Revised 19 Apr. 2024, Accepted 05 May 2024, Published 10 Aug. 2024

Abstract: Indoor positioning systems (IPS) are widely used for different use cases, but most of them are for asset tracking and indoor navigation. Asset tracking, for instance, might help industries be more efficient, such as warehouses, stock recording, guest trackers, and many more. Implementation of asset tracking needs to have the IPS, such as trilateration and fingerprinting. To have an accurate location, it is not just precise; the data that will be consumed also needs robust services to process all that data in almost real time. Bluetooth low energy (BLE) is used to send the received signal strength indicator (RSSI) to the microservices-based server. To support this, microservices architecture (MSA) is designed with a Service-Oriented Modeling and Architecture (SOMA) framework to translate business goals into necessary services. We were implementing and comparing both MSA implementation strategies, which are orchestration and choreography strategies, on cloud computing with the Kubernetes platform. These strategies were compared to find the most resource-efficient with the biggest number of served requests. The bigger the served request number, the more assets will be tracked in real time. Less resource usage could also mean the computational cost is inexpensive. The study finds that the choreography strategy in MSA is better for IPS since the number of served requests is five times higher with similar resource usage.

Keywords: Indoor Positioning System, Bluetooth Low Energy, Asset Tracking, Microservices Architecture, SOMA Framework

1. Introduction

The Global Positioning System (GPS) is being used everywhere for daily tasks, e.g., finding addresses, tracking packages, and others. GPS is powerful since it uses satellite technology, but it cannot be used for specific scopes such as indoor navigation or tracking assets inside a building or underground. Many industries, including healthcare, supply chain management, retail, construction, manufacturing, and many more, can utilize these scenarios. For instance, the military uses asset tracking for target tracking, where the system estimates and detects an object's behavior, even predicting its future location [1]. In the supply chain industry, asset tracking serves as a tool for logistic tracking and inventory management. With this, the company can track and monitor the shipment progress and conduct inventory counts more effectively in real time. Additionally, clothing stores could use the asset tracker as an anti-theft and counterfeiting device, triggering an alarm when the clothes pass through exit points [2]. Asset tracking is also useful for companies that need to track their employees' locations. The tracker could be placed on an employee's laptop or a construction worker's safety helmet to check whether their employees are on the office or construction sites [3]. With asset tracking that can be used as other sensors, companies can monitor their construction workers' conditions and safety at the same time. Safety and healthy workers will lead to better productivity [4]. Similar to the others, in hospitals, asset tracking could be implemented to track their patients, medical staff, medical equipment, and vehicles. The goals are to improve patient care, productivity, and operational efficiency [5]. Asset tracking in manufacturing can monitor the location and condition of products and materials. Sensitive industries like pharmaceuticals and food could standardize their products' quality by monitoring their surroundings, equipment, and environment [6].

While asset tracking is useful to improve business productivity, there are some challenges to implementing it. The cost of the device or tracker is really dependent on the number of devices, use cases, and devices' lifecycle plans [2]. The algorithm for asset tracking is also critical. A fully autonomous factory full of robotics, for instance, will need better accuracy compared to a factory that still has people do manual procedures. Internet of Things (IoT)

E-mail address: dondi.sasmita@binus.ac.id, inegara@binus.edu

energy efficiency and interoperability are also some of the challenges. These IoT devices need a constant supply of power, which could be expensive or limited in some cities or countries. Also, the lack of interoperability of IoT devices from different vendors will create challenge as well. The more custom the IoT interfaces or protocols, the longer it takes to integrate with existing or new devices [6].

The technology to get user-specific location inside a building or room is called an indoor positioning system (IPS), and it could be implemented using different technologies such as wireless fidelity (WiFi) [7], [8], [9], Bluetooth low energy (BLE) [10], [11], [12], [13], [14], radio frequency identification (RFID) [2], [5], etc.

BLE is a wireless technology that runs at 2.4 GHz frequency, just like wireless fidelity (WiFi), but BLE's size is smaller, the price is cheaper, and it consumes less energy, which makes BLE perform better in the long run and scale up efficiently in terms of cost. Batteries power most BLEs, but some devices, such as BLE Gateway, require more power than BLE Beacon and primarily rely on electricity.

Indoor positioning systems (IPS) commonly use several techniques, such as trilateration, fingerprinting [9], [12], [14], [15], and hybrid approaches. Trilateration is a technique that uses three devices to estimate distance. Fingerprinting is estimating distance by creating a radio map database to collect data from several points and create a model using that data. While hybrid combines both trilateration and fingerprinting.

There are several methods to estimate an asset's location, like weighted k-nearest neighbor (WK-NN), machine learning, and deep learning-based. Collecting fingerprinting data or a radio map database can complement these methods. One of the neural network-based algorithms is artificial neural networks (ANN).

There are several methods to estimate an asset's location. The great system behind the IPS is crucial for real-time data retrieval. There are few architectures to build systems, such as monolith [16] and microservices [17], [18], [19], [20]. Monolith's modules are dependent on each other, while modules in microservices are independent. Since the modules are independent, services in microservices architecture (MSA) could be written in different programming languages and use messaging protocols to communicate, like Protocol buffers (Protobuf) [21] and JavaScript Object Notation (JSON). Additionally, the IoT itself can use message queuing telemetry transport (MQTT) [22], [23] to send telemetry information to the server.

To start sending the telemetries, the public must be able to access the IPS's system anytime and anywhere. One of the easiest and most common is to host the system in a public cloud instead of on-premise. There are several global cloud hosting companies with reputable names, such as Google Cloud Platform (GCP), Amazon Web Services (AWS), Alibaba Cloud, Microsoft Azure, Digital Ocean, and many more.

In this experiment, BLE will be used as a beacon and gateway since it has several advantages, like its size, price, and efficiency. We will send the data from BLE to a microservices-based server. Researchers will design a microservices architecture with the Service-Oriented Modeling and Architecture (SOMA) framework [17] that suits asset tracking on IPS. The design will use both MSA's implementation strategies, which are orchestration and choreography strategies. We will deploy this MSA in a cloud computing environment featuring Kubernetes services. Finally, the comparison of served requests and resource usage between those strategies will be done to find the better approach for IPS.

2. Related Works

To implement IPS in real time, there are two things that need to be considered, which are indoor positioning methods and microservices architecture.

A. Indoor Positioning Methods

Implementation of IPS is dominated by Bluetooth and WiFi technologies, especially when Bluetooth low energy (BLE) is introduced because of its power consumption. Advertisement packets, a feature of BLE, report the Received Signal Strength Indicator (RSSI).

In 2021, Suseenthiran et al [10] also did research regarding BLE. They used the RSSI, calculated the information using trilateration, and then sent it via LoRa (long range) technology. Their research can be continued in the future for IoT (Internet of Things) purposes since LoRa can send signals over a long range.

Other than Trilateration, Fingerprinting is also becoming more popular. This method can be used for BLE and WiFi technologies as well [8], [9], [12]. Riady & Kusuma used Fingerprinting and pedestrian dead reckoning (PDR) combined with ANN.

Researchers are increasingly using neural networks as part of their methods. Sulaiman et al, in 2022, were using generalized regression neural networks (GRNN) [8]. Combined with the fingerprinting technique, their research divides the phase into two: offline and online. The offline phase is to collect data from radio map fingerprinting and create a neural network model.

Cha & Lim also proposed a new framework based on neural network called the hierarchical auxiliary deep neural network (HADNN) in 2022 [9]. The idea is to simplify the calculation of building, floor, and users' locations at once. Compared with the same dataset from TUT2017 and TUT2018, HADNN gave a better result and improved the floor's accuracy to 94.58%.

When trilateration or fingerprinting alone is not enough, there is research that combines those two to get a better

result. Lie & Kusuma are two of them. They proposed the coarse-to-fine algorithm in 2021 [24]. The mean position error (MPE) is 0.874 meters, and the computation to process it is between 0.4 and 0.7 milliseconds.

Mehrabian & Ravanmehr, in 2023, also used a hybrid approach that combined PDR and fingerprinting [25]. They used a novel filter called weight-based optimization (WBO) to optimize the RSSI value. Then they optimized the path loss parameter with particle swarm optimization (PSO) to convert RSSI to a distance value. The mean absolute error (MAE) value is 0.68 meters from a room with a size of 4.25 x 10.7 $meter^2$.

From the IPS work that has been done in the last couple of years, there are trends to use BLE over other technologies because its performance gets better each year. Also, BLE is one of the cheapest options to mass produce. Hybrid is also becoming a way to find a better result by combining the advantages of each methodology.

B. Microservices Architecture

"A monolith is a software application whose modules cannot be executed independently" [16]. One of the reasons organizations moved from monolith was the monolith's flexibility. Organizations were unable to selectively scale up modules or services, as they were required to expand the entire architecture. As the system scales up, so does the cost.

"A microservices is a cohesive, independent process interacting via messages. While A microservices architecture (MSA) is a distributed application where all its modules are microservices" [16].

A microservices architecture is totally different from a monolith, but there is no definitive answer to which architecture is the best. Both architectures have their strengths and weaknesses. Velepucha Flores did research in 2023 regarding the architecture's strengths and weaknesses [26]. Small applications demonstrate the strength of monolithic architecture, as it simplifies development and deployment. Maintenance, reusing code, and tracking code changes are also simpler. However, as the application grows, so does the cost of maintenance and scaling. It is also difficult to use a different programming language. This case will arise when there is a limitation on the programming language used. Where a microservices architecture has strengths, the system could always be online. This happened since during updates and deployments, we don't need to restart the whole application. We can assign the team specifically for each service, making maintenance and testing are easier.

In 2004, Ali Arsanjani introduced techniques to identify, specify, and realize services in a service-oriented architecture (SOA), their flows and composition, as well as the enterprise-scale components needed to realize and ensure the quality of services required by SOA [27]. Then, in 2008, Arsanjani et al proposed a method for developing service-oriented solutions called service-oriented modeling and architecture (SOMA) [17]. SOMA is used for different sizes of scopes in multiple industries worldwide. The idea of SOMA is to standardize on how to analyze, design, implement, and deploy SOA effectively.

In 2021, Wang et al surveyed and interviewed professionals with different backgrounds and positions, from software engineers to devops [28]. Wang et al conducted this survey in two ways: face-to-face and online, focusing on three main concepts: architecture, infrastructure, and code management. 81% of respondents cite the business process as the primary factor in determining the MSA granularity of the services under development. While 43% refer to data access and 24% to the team's structure.

Still in the same year, de Toledo et al also interviewed 25 practitioners related to 16 architectural technical debts (ATD). There's a potential issue when MSA is developed with shared databases if it is not well designed. A database per service and a saga pattern [29] could solve this issue, ensuring data consistency for each service.

Other than data consistency, there are distribution, portability, availability, and robustness, which are a few key characteristics of MSA [30]. There are several ways to implement these, including orchestration using Docker. This will isolate problems in each service without affecting other services.

Docker is one of many tools to contain the microservices. The goal is to simplify the lives of developers by automating scale-up and scale-down processes. Dragoni et al did research to show how powerful containerization and orchestration microservices are with Kubernetes [30]. With the tool, developers can focus on the service's performance.

In MSA, performance can be affected by its design. Shadija et al, in 2017, did research that shows services' granularity could affect MSA performance if their design is too deep [31]. The granularity could be affected by the size of the application, business processes, number of software engineers, database design, and the re-use of the services. Infrastructure also affects performance, like the latency and the service's location in the container.

When designing the MSA, other than performance, we need to look at the business model. If the product will be software as a service (SaaS), then the approach might differ from internal use. In 2020, Song et al researched a SaaS that can be modified [32]. Normally, customers can only customize a small part of the SaaS, but in this research, they proposed a method known as deep customization. The idea is to group the granularity into four, which are class/function, components, services, and languages.

MSA is also advantageous in terms of portability and service maintenance. But with the services that are growing and getting bigger, sometimes it would be tricky to find the

root cause of issues; therefore, Brandon et al researched a topic related to root cause analysis (RCA) in MSA. They were using graphs to help visualize the RCA [33]. From their findings, the graph solution is better than the machine learning approach, with a difference of around 19.41%. They used Kafka and Spark to detect system anomalies and save them. The solution is trained by using the data's patterns and shows the RCA in a graph. Their solution allows users to set priority at the service, component, or database level.

You can implement a microservices-based architecture in on-premise, cloud, and edge computing. Tusa et al compared the microservices-based system with serverless functions in edge computing to support IoT analytics [18]. Since edge computing resources are close to the data sources, the latency will be small and suitable for real-time data processing. According to this research, microservices with auto-scaling enabled are better than static ones.

Cota et al proposed a system called BHiveSense, which is an information system for remote monitoring and management of apsiaries based on microservices and IoT [20]. The goal is to "monitor honeybee colonies to promote more sustainable resource usage and maximize productivity.". The system itself is divided into four groups: a prototype of commercial off-the-shelf hive sensing, a Representational State Transfer (REST) backend API, a web application, and a mobile application. The backend is implementing a microservices architecture with ten modules, all of which are transparent to the users. The information is then availbale to the public via the API gateway module.

Atitallah et al, in 2023, researched how to implement microservices with entities that are fine-grained and loosely connected for deep-learning-based disease diagnosis applications [19]. They used microservices to decouple and distribute their federated learning model. They are breaking down the data analytic functions into smaller and more specific microservices. With these fine-grained services, they are able to improve scalability by making sure the service is optimized and resource-efficient. This proposed approach shows that it is better than traditional monolithic architecture with centralized learning.

3. Theory and Methods

A. Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a wireless technology that runs on an unlicensed frequency of 2.4 GHz. The price of BLE is cheaper compared to other technologies, and it is supported by most smartphones, making BLE quite common to be used [34]. BLE power consumption is also quite efficient, and its relatively small size makes this technology easier to deploy [10]. BLE also has higher sample rates, a smaller network latency, and better accuracy than WiFi [11].

For several scenarios, BLE can be grouped into two: a beacon and a receiver. A BLE beacon is a Bluetooth-based device that can send signals and is powered by batteries [13]. A receiver is a Bluetooth-based device that can receive signals from transmitters or beacons that will be processed or sent to a server.

BLE has a feature that reports the RSSI, which will be used to calculate the signal's strength between the beacon and receiver. The power of the signal is dependent on distance and the broadcast power value. From RSSI data, the distance between beacon and receiver can be calculated using several methods, like Lateration and Fingerprinting.

B. Fingerprinting

Fingerprinting is divided into two phases: offline and online. During the offline phase, RSS data will be collected from several reference points, as shown in Figure 1. With this information, a radio map database will be created. Each row of the database contains two dimensions, x and y. The offline phase's goal is to form a model [34]. Later, in the online phase, this data will be used to estimate the asset location using WK-NN.

To get the weight for this method, we should first calculate (1)

$$W_n = \sum_{i=1}^{n} \frac{1}{d_i} \tag{1}$$

Where W is the weight's value and d is the result of the calculation between the asset location's RSS and the RSS from the radio map, which will be calculated using Euclidian distance as shown in (2)

$$d_i = \sqrt[2]{\sum_{j=1}^{n} \left(p_{ij} - \hat{p}_{ij}\right)^2} \tag{2}$$

Where p is the RSSI value that saved in radio map and $\hat{p}$ is the measured RSSI. Once we have all the distances and weights, we should normalize the weights using (3)

$$NW_i = \frac{W_i}{\sum_{j=1}^{n} W_j} \tag{3}$$

By calculate each normalized weight, asset's location's estimation can be calculated using (4)

$$(x, y) = \sqrt{\sum_{j=1}^{n} \left(NW_j \cdot \left(x_j, y_j\right)\right)} \tag{4}$$

Where (x, y) is estimated position while $\left(x_j, y_j\right)$ is reference point.

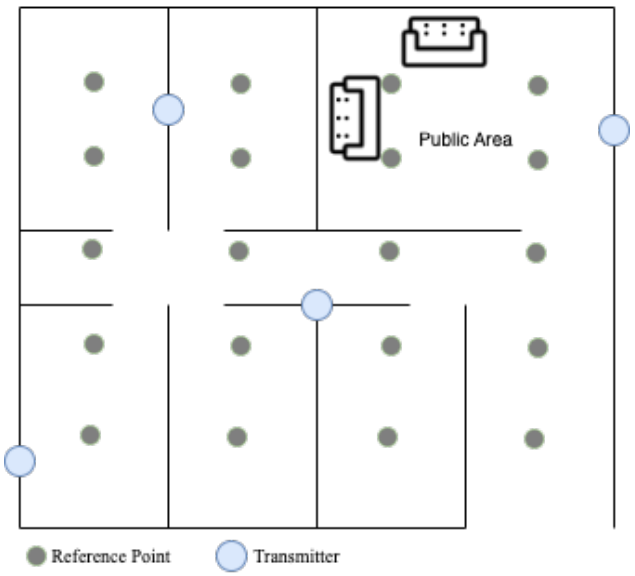

Figure 1. Fingerprinting Method

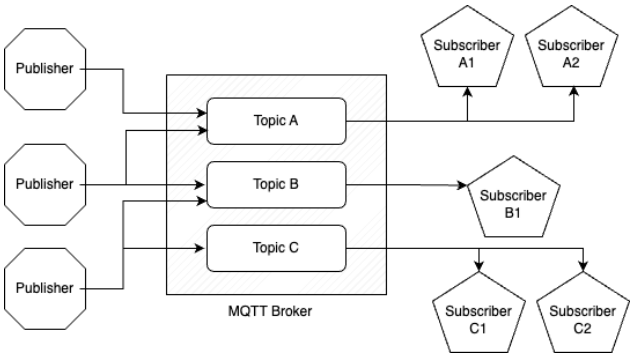

Figure 2. Publisher Subscriber Pattern

C. Message Queuing Telemetry Transport

This protocol is designed specifically for operation on low-cost and low-power sensor or actuator devices. MQTT runs over TCP with a small data packet, so the power consumption is minimized [23].

MQTT uses a publish/subscribe pattern. The publisher will send the message, while the subscriber will receive it. In order to filter messages, MQTT uses topic to group the messages, so the subscribers only get messages from the topic which they subscribed, as shown in Figure 2.

MQTT also provides quality of service (QoS), which guarantees the reliability of message delivery. There are three levels, level 0 messages are sent only once. Level 1 messages are sent at least once, so if the subscribers are not ready, the publishers could re-send the messages again. Lastly, Level 2 messages need to be received. Important or chained data might use level 2 QoS.

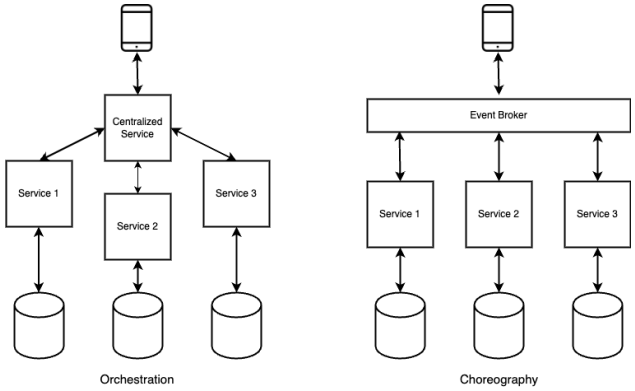

Figure 3. Orchestration and Choreography

D. Microservices

A microservice is a cohesive, independent process interacting via messages. And a microservice architecture is a distributed application where all its modules are microservices [16].

A few characteristics of MSA that make it more favorable than monolith architecture are flexibility, modularity, size, and independence. Flexibility means the system should be aligned with business dynamics and support the modifications needed by the organization. The system's components need to be isolated, where each component contributes to the whole function. In MSA, the service's size should be small, if it is too big, the service needs to be broken down into two or more services so each service can focus on one thing only. Lastly, services in MSA should be operated independently from other services and can only communicate through published interfaces.

MSA can be implemented using two strategies, which are orchestration and choreography, as shown in Figure 3. Orchestration needs a conductor, a centralized service that will send requests to other services until a response is received. While choreography assumes that there is no centralized service, the strategy will use publisher subscriber mechanisms to collaborate between services.

Figure 4 shows the deployment era from traditional to container. In traditional deployment, the application is running on top of the operating system, which will be difficult if there are applications that need to run on different stacks. This issue was solved in the virtualization era, where the application will be running inside a virtual machine. But this deployment also has an issue where there are multiple applications that run on different versions of libraries. For instance, application A needs to run on Java 8, while the other is using Java 11. These Java versions might need different libraries, and if the system was updated, the application might break. That is where the containerization deployment tries to address this issue, where each application will have its own environment. With this scope, developers would not worry about breaking applications

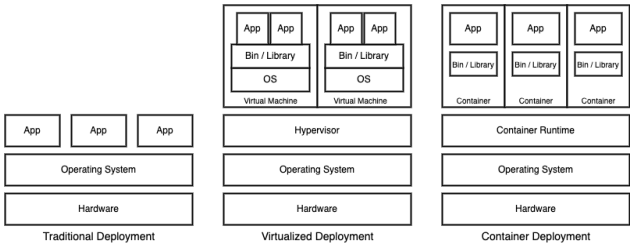

Figure 4. Comparison Between Deployments

when updates are applied.

E. Google Protocol Buffers and Remote Procedure Call

Protocol buffers (Protobuf) were developed by Google and introduce a mechanism to encode structured data in an efficient and extensible format [21]. Protobuf is both human- and machine-friendly because it allows developers to create a Proto file, which will later be compiled into a native language. Protobuf is language- and platform-independent, which is in line with microservices behavior. It supports Go, Python, Kotlin, and other programming languages.

Google Remote Procedure Call (gRPC) is an open-source remote procedure call framework that utilizes protobuf as an interface description language and uses the next version of the hypertext transfer protocol (HTTP), which is HTTP/2 [35]. gRPC can use a single HTTP/2 connection for bi-directional communication between client and server. This key feature allows users to stream messages in both directions.

Figure 5 shows the implementation of gRPC with Protobuf. The gRPC server can be written in Go and will be called by the gRPC stub written in Python or Kotlin. These gRPCs use Protobuf to send requests and receive responses.

F. Cloud Computing

Cloud computing is a model that allows networks to demand configurable resources like networks, databases, servers, applications, and services that can be provided and deleted instantly using the user interface [36].

There are five elements in cloud computing that is on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service.

On-demand self-service gives customers the flexibility to instantiate new resources (Central Processing Unit, Storage, etc.) automatically without any interactions with other people who provided the services. With broad network access, the resources can be accessed from public devices such as laptops, personal computers (PCs), and smartphones. The goal of resource pooling is to give economic scale to customers by implementing multi-tenancy and model virtualization. The next element is rapid elasticity, which makes sure customers can add and remove resources without commitment or a long-term contract. And to give customers a way to evaluate their resources, cloud computing

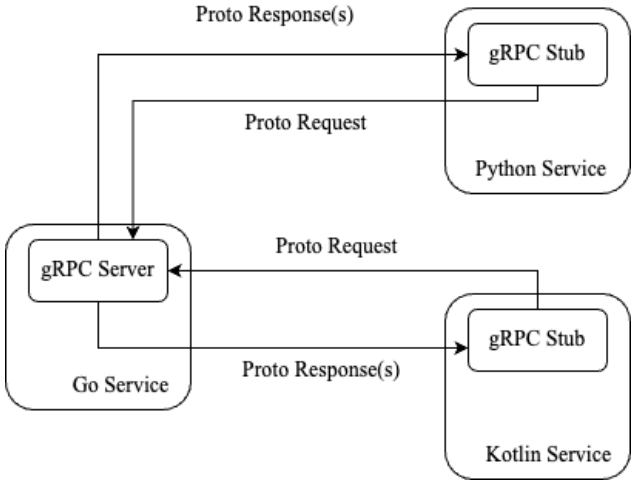

Figure 5. Google Remote Procedure Call

hosting gives information based on measured resources that customers have subscribed to.

In terms of services, there are four groups that communities are aware of, which are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), and Data Storage as a Service (DaaS) [36].

4. Proposed Microservices Design Framework

We used the Service-Oriented Modeling and Architecture (SOMA) framework to interpret indoor positioning's business use case into services. There are six phases in SOMA, which are business and modeling transformation, identification, specification, realization, implementation, and deployment & monitoring.

The research is focusing on estimating the location of assets inside a room within a building. During identification, we used goal-service modeling (GSM) to transform business goals into services based on their key performance index (KPI) and the metrics that needed to be measured. The goal-service modeling can be seen in Table I.

Once we identified basic services through GSM, we detailed the services with a technique called domain decomposition. This technique is a top-down analysis of business models and business process modeling to identify services, components, and flows. The result is in Table II.

During domain decomposition, we focused on account management and asset location only. Here we were detailing the process of account registration, where the flow is enhanced by email verification and notification. We did the same thing with the office registration, BLE registration, and asset location. We added more flows to those services.

Once the services are fixed, we then group those functions into services and explore their technical feasibility through early prototypes designed and developed. These

TABLE I. Goal-Service Modeling

Goal	KPIs	Metrics	Services
1. Attract and retain customers	Increase new and retain existing users		
1.1. Enable assets tracking service	Increase new and retain existing users with indoor positioning system		
1.1.1. Enable assets tracking service in office complex	Increase new and retain existing users with indoor positioning system for office complex	Number of indoor positioning system users in office complex	• Account Registration • Account Management
		Number of indoor positioning system tracked assets in office complex	• Office management • BLE management • Asset management • Assets' estimation's location • Assets' location histories • Algorithm management
	Improve indoor positioning system's quality	Number of issues and solved issues of indoor positioning system	• Store log information • Monitoring response time

TABLE II. Domain Decomposition

1. Account Managemet	1. Account Managemet
1.1. Account Registration	1.1. Account Registration
1.2. Office Registration	1.1.1. Email Verification
1.3. BLE Registration	1.1.2. Notification
2. Assets Location	1.2. Office Registration
2.1. Location Estimation	1.2.1. Building Registration
2.2. Location Histories	1.2.2. Floor Registration
	1.3 BLE Registration
	1.3.1. BLE Beacons
	1.3.2. BLE Gateways
	2. Assets Location
	2.1 Location Estimation
	2.1.1. RSSI Data Checking
	2.1.2. Estimation Methods
	2.1.3. Estimation Process
	2.2. Location HIstories

steps are part of specification and realization. The designed architecture can be seen in Figure 6.

5. Proposed Method

A. *Indoor Positioning System*

We used an Espressif board named ESP32-S3, as shown in Figure 7. It has BLE 5.0 and Wi-Fi modules embedded. We will need to write a custom firmware so it will scan the BLE Beacon with a universally unique identifier (UUID) that is predefined. This device will be our gateway to receive beacon RSSI, then assemble a payload with several other pieces of information and send it to the server. We will put six devices in a room with a size of 19 x 12 $meter^2$.

The fingerprinting method with WK-NN will be used to estimate the location of the BLE beacon. With fingerprinting, there will be two phases: online and offline. For the offline phase, we will collect RSSI data based on 46 reference points and 45 testing points. The floor plan can be seen in Figure 8. The blue color is the BLE gateway, the green color is the reference point, and the red color is the testing point. We will pick 45 random testing points from the plan. The number of K that will be used is 6.

Once we get the radio map, we will estimate the location with WK-NN (4) and calculate the mean position error (MPE) with (5).

$$MPE = \frac{1}{n}\sum_{i=1}^{n} error_i \tag{5}$$

where error is a Euclidean distance calculated from the real location of the beacon (x, y) with the estimated location $\left(x_p, y_p\right)$ as shown in (6)

$$error = \sqrt{\left(x - x_p\right)^2 + \left(y - y_p\right)^2} \tag{6}$$

Together with the error, we will benchmark the estimate's process time, Central Processing Unit (CPU) and Random Access Memory (RAM) usage as well.

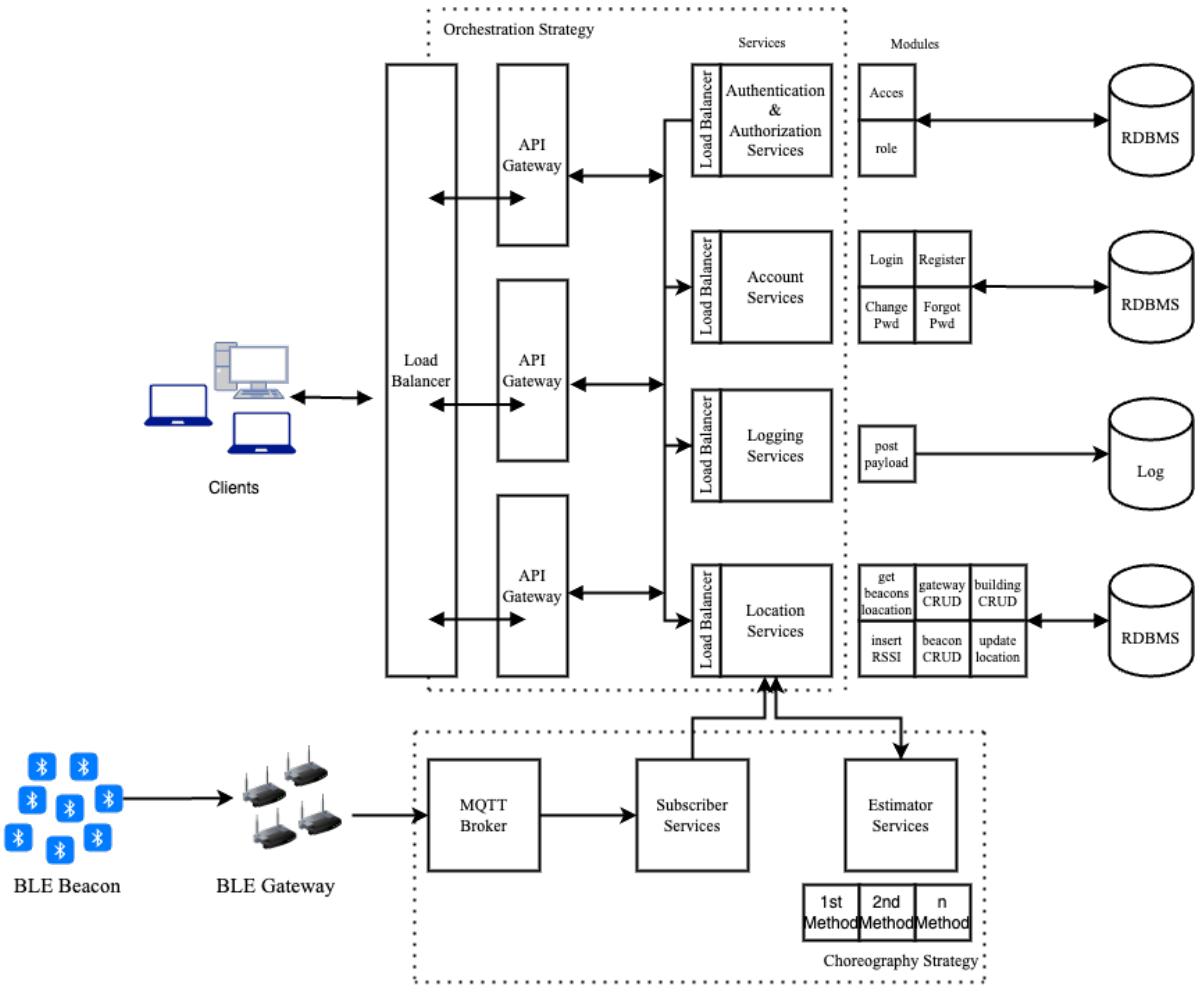

Figure 6. Proposed Microservices Architecture

Figure 7. Espressif ESP32-S3

B. *Microservices*

With SOMA, we have specified and decided on the IPS functionalities. We translated those features and services into a chart shown in Figure 6. There will be seven services, which are account, authentication, log, location, subscriber, estimator, and API gateway services. To support those services, third-party software will be installed as well. They are PostgreSQL as a relational database management system, MongoDB as a document data model, and EMQX as an MQTT broker.

The subscriber service is the first service that receives RSSI information from BLE gateways. The information will be sent by BLE gateways using the MQTT protocol through the MQTT broker. The main program of the subscriber service is to establish a connection with an MQTT broker and subscribe to a certain topic. Every time a payload is sent to a topic, the subscriber service will receive it and process the payload. Once the payload is processed, the subscriber will call the location service using a Google remote procedure call (gRPC) with a protocol buffer.

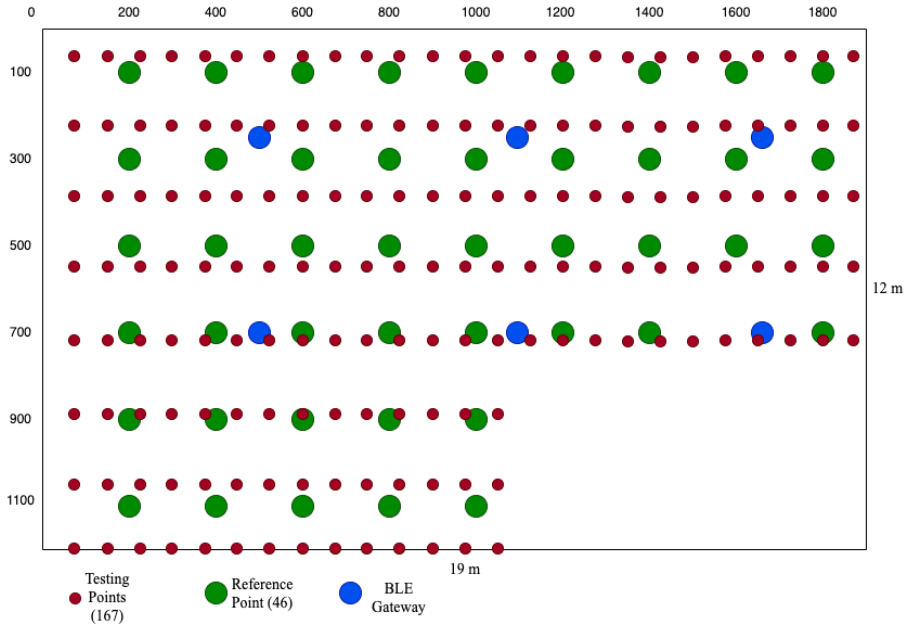

Figure 8. Fingerprinting Floor Plan

The location service is designed to store information related to the location of the asset. Based on our business flow, we stored several pieces of information, e.g., buildings, beacons, gateways, estimation algorithms, RSSI, and the assets' locations themselves.

Even though the estimated location is stored in the location service, the process of estimation itself happens in a different service, which is the estimator service. This service job is only to estimate assets' locations based on methods or models that have been assigned to a building or a floor. There are possibilities that different buildings or floors might need different algorithms. This service will call the location service's function periodically to check whether there are assets that need to be estimated. If there is, the estimator service will check the assigned algorithm, use it to estimate location, and return the coordinates of X and Y to the location service.

For security reasons, all services are inaccessible from outside the private network, except the API gateway service. The API gateway service is built as an orchestrator, hence the name of the strategy. As an orchestrator, the flow inside microservices is transparent to the client. For instance, when clients call an endpoint to retrieve a list of buildings, the API gateway communicates with several services, which are account, auth, log, and location.

The other services, which are account, auth and log services, act as supported services in this research. Those services are developed to do basic things such as users' registration, checking passwords, storing log information in databases, and other things that are not directly affected by the location service's performance.

To host these services, we used a cloud provider named Digital Ocean in the Singapore region for this research's infrastructure. They have a managed Kubernetes service, so we could focus on developing the services instead of maintaining the Kubernetes platform. We will use their load balancers as well to balance the traffic for both orchestration and choreography strategies.

Digital Ocean has a dashboard that shows metrics related to their servers and Kubernetes platform. This will help us measure our services' performance. During the test, the metrics that will be measured are CPU and RAM. Other than those, we will measure the number of served requests too.

C. *Evaluation*

The microservices performance's test scenarios are to compare between two strategies: orchestration and choreography. For orchestration, there will be two sub-scenarios where we will test the API gateway with the Post method only and the Get & Post methods together. So, there will be three scenarios in total. Each scenario will be tested with the same settings. The settings are to test with different numbers of nodes and pod replication. We will test with 1 node & 3 pods, 1 node & 5 pods, 2 nodes & 5 pods, and 2 nodes & 8 pods. In terms of the number of connections, the test will use 100, 500, and 1,000 concurrent connections.

The specification of the Kubernetes cluster's machine is 8 vCPU and 16 GB of RAM per node. For the load balancer, we will use two nodes so it can accommodate up to 20,000 connections, 20,000 requests per second, and 500 SSL connections concurrently.

TABLE III. Fingerprinting Radiomap

X (cm)	Y (cm)	G1	G2	G3	G4	G5	G6
200	100	-61	-63	-65	-73	-70	-76
200	300	-59	-69	-68	-73	-71	-76
200	500	-57	-55	-61	-67	-75	-73
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
1800	500	-68	-62	-54	-54	-40	-47
1800	700	-66	-66	-59	-54	-54	-31

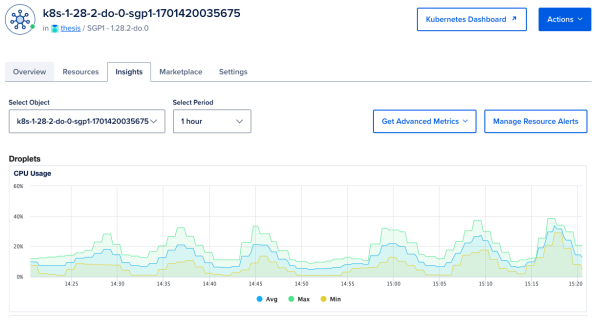

Figure 9. CPU Usage on Kubernetes Platform

6. Result and Discussion

A. Data Acquisition

After installing the BLE gateways and deploying the microservices on the cloud, we initiated the data collection process. Using fingerprinting, we collected RSSI based on reference points (RP). There are 46 RPs, each with 100 sample data points. Total RSSI data is 4600 records. From these records, we calculate the average RSSI for each RP, so by the end, the radio map based on reference points only has 46 records. This radio map was then uploaded to the estimator service for the online fingerprinting phase. To calculate the accuracy, we collected another radio map based on testing points (TP), where the coordinates are different from RP.

The radio map format for both RP and TP is the same. It has eight columns that represent beacon coordinates (x and y). The coordinates are in centimeters and started from the base point on the floor plan's top left, as shown in Figure 8. The other six are RSSIs from different BLE gateways. Table III displays the format..

With these radio maps, we calculated the WK-NN method's accuracy and resource usage. Using (4) and (4), the method yields an MPE of 6.32 meters. The estimation process took about 1.98 milliseconds with CPU usage around 1.2% and 0.5% RAM.

Compared to several other indoor positioning methods, the WK-NN result in this study is underperforming. Sulaiman et al with the same fingerprinting method, show a better result with an error of 0.48 meter. The differences are that they were using WiFi instead of BLE and using more complex method compared to WK-NN, which is neural network based called GRNN. Riady & Kusuma used 23 BLE beacons with fingerprinting, and ANN got a better result with an MPE of 1.11 meters. Aranda et al with 360 BLE beacons with fingerprinting and WK-NN, got a normalized error of 0.68 meters.

The number of beacons used in related works seems to affect the IPS accuracy since Riady & Kusuma and Aranda et al got a much better result with the same technologies as our research. When the location requires precision, such as in a fully autonomous factory, it is important to consider the number of beacons. Despite the BLE's lower cost compared to other options, the implementation of IPS in larger buildings or factories will ultimately result in higher costs. While the method or model might be dependent on the room's layout and materials, WK-NN could achieve similar accuracy with other methods like neural network.

B. Microservices Architecture

Once the fingerprinting data was collected, we tested the microservices' performances. There are three scenarios: orchestration strategy for the GET and POST methods; orchestration strategy for the POST method; and choreography strategy.

We conducted load testing on the orchestration strategy using a modern benchmark tool named Bombardier. It works by calling an endpoint for a specific period of time with a certain number of concurrent connections. We called the endpoints for 120 seconds with 100, 500, and 1,000 connections to find the number of served requests and their CPU & RAM usage.

For choreography strategy, we wrote a custom Python script to imitate a real-life scenario where the BLE gateway pushes the RSSI payload through the MQTT broker for each interval time. In this scenario, we sent six payloads per second for each 120-second connection. We used the same number of connections as the orchestration, which are 100, 500, and 1,000 concurrent connections.

We used four settings to test the microservices' performance. 1 node with 3 replications, 1 node with 5 replications, 2 nodes with 5 pod replications, and 2 nodes with 8 pod replications. Table IV, Table V, and Table VI displays the results. The numbers are available from the cloud dashboard, as shown in Figure 9.

Table IV shows the results for the orchestration strategy for the GET & POST methods. This scenario demonstrates that the largest served request per second (RPS) occurred when there were 1,000 concurrent connections, referred to as the API. The RPS is 703. The average CPU usage is 33.3%, and the RAM is 15.2%.

Table V, the second scenario, shows that calling the Post

TABLE IV. Orchestrations Strategy Result (Get & Post)

No. of connections	No. of Nodes	No. of Pods	Average CPU (%)	Peak CPU (%)	Average RAM (%)	Peak RAM (%)	Total Served Requests	Served Requests Per Second
100	1	3	22.2	36.4	20.0	20.2	26,627	221
100	1	5	33.4	55.2	21.1	21.2	35,702	297
100	2	5	15.1	29.1	14.1	15.1	36,239	301
100	2	8	23.9	46.9	13.9	17.3	55,905	465
500	1	3	36.1	49.0	20.2	20.4	37,310	310
500	1	5	32.7	47.6	22.0	22.0	51,721	431
500	2	5	17.1	35.1	14.4	15.5	44,829	373
500	2	8	25.1	46.0	14.1	17.5	79,120	659
1000	1	3	27.1	33.9	21.4	21.8	31,926	266
1000	1	5	36.5	41.2	22.4	22.9	51,876	432
1000	2	5	16.5	26.0	14.5	15.4	47,287	394
1000	2	8	33.3	63.6	15.2	17.2	84,367	703

TABLE V. Orchestrations Strategy Result (Post)

No. of connections	No. of Nodes	No. of Pods	Average CPU (%)	Peak CPU (%)	Average RAM (%)	Peak RAM (%)	Total Served Requests	Served Requests Per Second
100	1	3	31.2	38.1	20.5	21.0	23,638	196
100	1	5	42.7	57.2	22.1	22.8	34,538	287
100	2	5	23.0	35.3	14.7	14.9	31,216	260
100	2	8	31.6	46.8	15.5	16.8	38,292	319
500	1	3	32.1	40.1	20.5	21.1	29,346	244
500	1	5	42.4	59.9	22.2	23.1	27,143	226
500	2	5	23.7	39.2	14.6	15.1	33,844	282
500	2	8	32.1	57.3	15.5	16.9	40,784	339
1000	1	3	33.4	46.1	21.4	21.8	18,310	152
1000	1	5	44.5	63.0	22.3	23.0	23,652	197
1000	2	5	25.5	41.4	14.4	15.4	34,984	291
1000	2	8	34.3	65.6	15.3	17.1	43,580	363

method results in a smaller RPS. The RPS is 363, with an average CPU of 34.3% and RAM of 15.3%. Compared to the first scenario, this scenario used more CPU power. It seems the Post method mainly uses CPU since the RAM of 100, 500, and 1,000 connections is quite the same.

In the last scenario, the choreography strategy on Table VI gave us the best result with 3,816 RPS. With 1,000 connections, this scenario served 457,970 requests for 120 seconds, with an average CPU of 26.5% and RAM of 21.2%.

The choreography strategy shows a better result since the telemetry and estimation location services are separated into fine-grained services. This strategy aligns with research done by Atitallah et al where fine-grained service affects the performance. The event-driven pattern also plays a significant role, enabling the processing of telemetry data in parallel with multiple services or pods. It differs from what Cota et al did, where they used API Gateway with REST for their backend while we divided the backend into two, which are an orchestration strategy with the API Gateway for system information management only and an event-driven pattern in the choreography strategy to process the telemetry.

Choosing the correct region for cloud computing could also help achieve better performance. This is similar to the idea of edge computing research done by Tusa et al where the closer the server, the better the latency, and it will increase the MSA performance. For the IPS in this study, cloud computing is suitable since the telemetry's size is quite small compared with other IoT systems that have more data and are sensitive to real-time information. There is no need to use edge computing because the cost will not outweigh the performance.

7. Conclusion and Future Work

We have implemented an indoor positioning system using BLE and microservices in the cloud environment. We designed the microservices using the SOMA framework and implemented them in choreography and orchestration strategies. We then deployed this MSA on a Kubernetes platform. After services were ready, both strategies were called with multiple scenarios, and the choreography strategy showed a better result with 457,970 served requests, or five times more than the other scenarios. The CPU usage is smaller compared to the orchestration strategy's scenarios, but RAM usage is 50% higher. We achieved this result by keeping the number of nodes and pods static. In the future, with cloud computing's prices and technologies evolving quickly, future research related to auto-scaling the architecture could improve the MSA's performance with reduced cost.

The location estimation method used in this research is a simple WK-NN since our focus is more on the microservices architecture. As the neural network flourishes, we could enhance the IPS by incorporating other location

TABLE VI. Choreography Strategy Result

No. of connections	No. of Nodes	No. of Pods	Average CPU (%)	Peak CPU (%)	Average RAM (%)	Peak RAM (%)	Total Served Requests	Served Requests Per Second
100	1	3	21.3	25.9	20.9	21.0	54,835	456
100	1	5	19.1	23.1	25.2	25.4	59,805	498
100	2	5	14.7	28.3	16.8	22.8	63,570	529
100	2	8	20.0	32.1	20.4	25.7	68,990	574
500	1	3	21.2	28.8	21.9	21.9	151,419	1,261
500	1	5	21.6	49.6	25.5	25.7	202,211	1,685
500	2	5	18.8	32.5	17.4	23.2	237,448	1,978
500	2	8	23.8	37.3	20.4	25.9	233,798	1,948
1000	1	3	23.4	33.7	22.0	22.0	166,682	1,389
1000	1	5	44.8	62.8	26.1	26.3	313,122	2,609
1000	2	5	19.1	33.5	17.8	23.5	338,781	2,823
1000	2	8	26.5	38.6	21.2	25.9	457,970	3,816

estimation methods, like the Graph Neural Network (GNN), to ensure a more precise estimated location in the future. The number of devices, building's layouts, building's materials, and multi-floor should be taken into consideration when choosing the right methods or model to get better accuracy and performance.

References

[1] M. Alhafnawi, H. A. B. Salameh, A. Masadeh, H. Al-Obiedollah, M. Ayyash, R. El-Khazali, and H. Elgala, "A survey of indoor and outdoor uav-based target tracking systems: Current status, challenges, technologies, and future directions," *IEEE Access*, vol. 11, pp. 68 324–68 339, 2023.

[2] X. Su, "Application analysis of rfid in supply chain management," *Highlights in Business, Economics and Management*, vol. 24, pp. 122–128, 1 2024.

[3] V. Gujar and R. P. Singh, "Innovative indoai's smart asset tracking: Securing efficiency, security compliance in mobile work environments," *IARJSET*, vol. 11, 12 2023.

[4] M. A. Musarat, W. S. Alaloul, A. M. Khan, S. Ayub, and N. Jousseaume, "A survey-based approach of framework development for improving the application of internet of things in the construction industry of malaysia," *Results in Engineering*, vol. 21, p. 101823, 3 2024.

[5] K. Saritha, C. Abhiram, P. A. Reddy, S. S. Sathwik, N. R. Shaik, L. H. Alzubaidi, V. H. Raj, A. Dutt, and D. K. Yadav, "Iot enabled hospital asset tracking using advanced interdisciplinary approaches," *E3S Web of Conferences*, vol. 507, p. 01007, 3 2024.

[6] M. Soori, B. Arezoo, and R. Dastres, "Internet of things for smart factories in industry 4.0, a review," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 192–204, 2023.

[7] O. I. Mustafa, H. L. Joey, N. A. AlSalam, and I. Z. Chaloob, "Accurate indoor positioning system based on modify nearest point technique," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, p. 1593, 4 2022.

[8] B. Sulaiman, E. Natsheh, and S. Tarapiah, "Towards a better indoor positioning system: A location estimation process using artificial neural networks based on a semi-interpolated database," *Pervasive and Mobile Computing*, vol. 81, p. 101548, 4 2022.

[9] J. Cha and E. Lim, "A hierarchical auxiliary deep neural network architecture for large-scale indoor localization based on wi-fi fingerprinting," *Applied Soft Computing*, vol. 120, p. 108624, 5 2022.

[10] K. Suseenthiran, A. S. Ja'afar, K. W. Heng, M. Z. A. A. Aziz, A. A. M. Isa, S. H. Husin, and N. M. Z. Hashim, "Indoor positioning utilizing bluetooth low energy (ble) rssi on lora system," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, p. 927, 8 2021.

[11] G. P. Kusuma and M. M. K. Lie, "A review of indoor positioning system techniques using bluetooth low energy," *ICIC Express Letters*, vol. 13, pp. 1139–1147, 2019.

[12] A. Riady and G. P. Kusuma, "Indoor positioning system using hybrid method of fingerprinting and pedestrian dead reckoning," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, pp. 7101–7110, 2022.

[13] G. Pau, F. Arena, M. Collotta, and X. Kong, "A practical approach based on bluetooth low energy and neural networks for indoor localization and targeted devices' identification by smartphones," *Entertainment Computing*, vol. 43, p. 100512, 8 2022.

[14] F. J. Aranda, F. Parralejo, F. J. Álvarez, and J. A. Paredes, "Performance analysis of fingerprinting indoor positioning methods with ble," *Expert Systems with Applications*, vol. 202, p. 117095, 9 2022.

[15] M. Nabati and S. A. Ghorashi, "A real-time fingerprint-based indoor positioning using deep learning and preceding states," *Expert Systems with Applications*, vol. 213, p. 118889, 3 2023.

[16] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow," 6 2016. [Online]. Available: http://arxiv.org/abs/1606.04036

[17] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, "Soma: A method for developing service-oriented solutions," *IBM Systems Journal*, vol. 47, pp. 377–396, 2008.

[18] F. Tusa, S. Clayman, A. Buzachis, and M. Fazio, "Microservices and serverless functions—lifecycle, performance, and resource utilisation of edge based real-time iot analytics," *Future Generation Computer Systems*, vol. 155, pp. 204–218, 6 2024.

[19] S. B. Atitallah, M. Driss, and H. B. Ghézala, "Revolutionizing disease diagnosis: A microservices-based architecture for privacy-preserving and efficient iot data analytics using federated learning," *Procedia Computer Science*, vol. 225, pp. 3322–3331, 2023.

[20] D. Cota, J. Martins, H. Mamede, and F. Branco, "Bhivesense: An integrated information system architecture for sustainable remote monitoring and management of apiaries based on iot and microservices," *Journal of Open Innovation: Technology, Market, and Complexity*, vol. 9, p. 100110, 9 2023.

[21] S. Popic, D. Pezer, B. Mrazovac, and N. Teslic, "Performance evaluation of using protocol buffers in the internet of things communication." IEEE, 10 2016, pp. 261–265.

[22] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-s — a publish/subscribe protocol for wireless sensor networks." IEEE, 1 2008, pp. 791–798.

[23] R. A. Atmoko, R. Riantini, and M. K. Hasin, "Iot real time data acquisition using mqtt protocol," *Journal of Physics: Conference Series*, vol. 853, p. 012003, 5 2017.

[24] M. M. K. Lie and G. P. Kusuma, "A fingerprint-based coarse-to-fine algorithm for indoor positioning system using bluetooth low energy," *Neural Computing and Applications*, vol. 33, pp. 2735–2751, 4 2021.

[25] H. Mehrabian and R. Ravanmehr, "Sensor fusion for indoor positioning system through improved rssi and pdr methods," *Future Generation Computer Systems*, vol. 138, pp. 254–269, 1 2023.

[26] V. Velepucha and P. Flores, "A survey on microservices architecture: Principles, patterns and migration challenges," *IEEE Access*, vol. 11, pp. 88 339–88 358, 2023.

[27] A. Arsanjani, "Service-oriented modeling and architecture," *IBM developer works*, 3 2004.

[28] Y. Wang, H. Kadiyala, and J. Rubin, "Promises and challenges of microservices: an exploratory study," *Empirical Software Engineering*, vol. 26, p. 63, 7 2021.

[29] H. Garcia-Molina and K. Salem, "Sagas," *ACM SIGMOD Record*, vol. 16, pp. 249–259, 12 1987.

[30] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, "Microservices: How to make your application scale," 2 2017. [Online]. Available: http://arxiv.org/abs/1702.07149

[31] D. Shadija, M. Rezai, and R. Hill, "Microservices: Granularity vs. performance," 9 2017. [Online]. Available: http://arxiv.org/abs/1709.09242

[32] H. Song, F. Chauvel, and P. H. Nguyen, *Using Microservices to Customize Multi-tenant Software-as-a-Service*. Springer International Publishing, 2020, pp. 299–331.

[33] Álvaro Brandón, M. Solé, A. Huélamo, D. Solans, M. S. Pérez, and V. Muntés-Mulero, "Graph-based root cause analysis for service-oriented and microservice architectures," *Journal of Systems and Software*, vol. 159, p. 110432, 1 2020.

[34] R. P. Ghozali and G. Putra, "Indoor positioning system using regression-based fingerprint method," *International Journal of Advanced Computer Science and Applications*, vol. 10, 2019.

[35] C. Nimpattanavong, I. Khan, T. V. Nguyen, R. Thawonmas, W. Choensawat, and K. Sookhanaphibarn, "Improving data transfer efficiency for ais in the darefightingice using grpc," 3 2023.

[36] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges." IEEE, 2010, pp. 27–33.

Dondi Sasmita received B.Sc. degree in Computer Science major from Bina Nusantara University in 2009 and he is currently pursuing his Master's Degree from Bina Nusantara University. System architecture, cloud computing and machine learning are his research interests.

Gede Putra Kusuma received PhD degree in Electrical and Electronic Engineering from Nanyang Technological University (NTU), Singapore, in 2013. He is currently working as a Lecturer and Head of Department of Master of Computer Science, Bina Nusantara University, Indonesia. Before joining Bina Nusantara University, he was working as a Research Scientist in I2R – A*STAR, Singapore. His research interests include computer vision, deep learning, face recognition, appearance-based object recognition, gamification of learning, and indoor positioning system.