Hindawi
Mobile Information Systems
Volume 2020, Article ID 8896252, 12 pages
https://doi.org/10.1155/2020/8896252

Research Article

Smart Edge Broker for Location-Based Transfer between Services and Distributed Data in IoT Smart Services

Junguk Ahn[1] **and Byung Mun Lee**[2]

[1]*Department of IT Convergence Engineering, Gachon University, Seongname-Daero 1342, Seongnam, Republic of Korea*
[2]*Department of Computer Engineering, Gachon University, Seongname-Daero 1342, Seongnam, Republic of Korea*

Correspondence should be addressed to Byung Mun Lee; bmlee@gachon.ac.kr

Received 20 April 2020; Revised 3 June 2020; Accepted 17 June 2020; Published 7 July 2020

Academic Editor: Jinan Fiaidhi

Various kinds of smart sensors are being developed and released with the growth of sensor technology and Internet of Things (IoT) technology. IoT smart service can provide convenience in our daily lives with these smart sensors. However, the increasing number of mobile sensors and amount of data may increase latency. It may cause network congestion on a particular network. Therefore, we propose a Smart Edge Broker (SEB) to intelligently transmit data traffic generated by a smart city in this paper. SEB can prevent the traffic congestion from being transmitted or bypassed to a location where traffic is not necessary. In addition, SEB can prevent overload in a specific area between services through a location-based transfer. Plus, SEB is suitable to operate it as a fog computing model by placing it at the edge of a smart city network. We conducted a latency measurement experiment and load measurement experiment to evaluate the effectiveness of the proposed Smart Edge Broker. As a result, we found that the latency was reduced by 72%, and the CPU usage was reduced by 63% compared to when the Smart Edge Broker was not used.

1. Introduction

To provide convenience in our daily lives, it is possible to configure a smart home using various IoT technologies [1–6]. For example, if we attach health sensors such as an ECG recorder and blood glucose meters to our body and link them to a smart home system, it is possible to remotely monitor our health [7]. Alternatively, by linking a sleep care device to a smart home system, it is possible to measure sleep and create an environment for improving sleep [8]. If smart home services are linked to a smart city platform, it can provide a variety of services [9–12]. For example, a smart home system can monitor the activities of elderly people and people with dementia at home and provide more complex medical support services through a smart city platform in the event of an emergency [13]. However, as the coverage of the network becomes broader, the number of sensor devices increases, which may cause several problems [14, 15].

First, latency can be prolonged. This means that it takes a long time to transmit data because the transmission distance of sensor data is too far. If the distance between smart home sensors that generate data and a service that consumes the data increases, there will be latency in transmission. This will impair the immediateness of sensor data, thereby lowering the quality of the service. Second, overload could happen. When measuring by increasing the number of sensors and widening the range, the amount of the data that needs to be processed increases, which may lead to bottlenecks due to an excessive concentration of traffic in a specific data center. If this problem happens, it will delay data processing and service provision. The third is service scalability. When new services are added to a smart home and smart city, all the related services need to be changed, increasing the cost as a result. These problems need to be solved when a smart city is configured. To that end, an intelligent distributed computing device is needed between sensors and services.

In this paper, we propose a Smart Edge Broker [16] as an intelligent broker for distributed computing. The Smart Edge Broker is located at the edge between sensors, services, and data centers across the city to intelligently route and

broker data transmission. Through this process, the data can be directly transmitted between sensors, thereby reducing the distance and latency. As it also processes the data transmission between sensors and services instead of data centers, it naturally solves the overload problem. If the Smart Edge Broker can filter only the necessary data, scalability can be improved.

In Section 2, we examine smart homes and smart cities to identify likely problems. In Section 3, we define and design a Smart Edge Broker that can solve the problems of a smart city. In Section 4, we implement the Smart Edge Broker and evaluate it by conducting experiments according to scenarios. In Section 5, we conclude based on the results of experiments and evaluations.

2. Related Studies

2.1. Smart Homes for Smart Cities. A smart home is an automation system that not only remotely controls home appliances in a house by connecting them to a network but also detects and actively controls situations using sensors [17]. Figure 1 shows an example of a sleep care service.

The sleep environment and biometric sleep information measured from sensors in each house are transmitted to an IoT sleep care service provided by the cloud [18]. Each home sensor A, B, C, and D has a sound sensor and a temperature sensor and a service gateway that collects the measured data and transmits it to IoT Sleepcare Service. This service collects and analyzes the data from each smart home and provides sleep care services to users. If a smart city infrastructure can be included in the range, it can also provide other services to the sleep care service [19]. In this case, the smart home functions as one sensor in a smart city [20]. Also, an urban sensor [21], which globally measures the urban environment, such as air temperature and noise, can be used. Home sensor A and home sensor C are in smart city_1, and they can utilize city sensor_1. Similarly, home sensor B and home sensor D are in smart city_2, and they can utilize city sensor_2 [22]. As an IoT sleep care service can utilize not only the sound and temperature sensor data from each smart home but also the temperature and environmental noise data of the entire city, it can provide area-based sleep care services.

2.2. Smart City from Smart Homes. A smart city is a system that provides support for efficient linkages between urban infrastructure and smart servers using ICT technologies [23]. A smart city platform consists of a sensor for measuring the environment, a network for transmitting measured data, and a data center for storing and managing the transmitted data [24]. Here, sensors include urban sensors that measure an urban environment and virtual sensors that transmit the data measured in each smart home. In OneM2M, these elements of a smart city were proposed by configuring them as one platform in the technical document "Smart Cities Done Smarter," as shown in Figure 2 [25].

As shown in Figure 2, the sensor at the bottom measures data and transmits it to a gateway. The gateway transmits it to a city application through a smart city frontend or transmits it to a smart city backend through a broker. The smart city backend acts as a data center to store and manage the data. After that, it transmits the data to a third-party application or analytics application through the smart city backend. In this process, network protocols such as Hyper Text Transfer Protocol (HTTP), Message Queuing Telemetry Transport (MQTT), and Constrained Application Protocol (CoAP) are used.

2.3. Application Protocol for Smart City. HTTP has high reliability because it communicates synchronously through the request-response method, but it is slow and consumes a lot of resources. In contrast, MQTT is fast and light because it asynchronously communicates through the publish/subscribe method, but it has low reliability because it cannot guarantee sequential data processing.

In Figure 3(a), a smart city platform of an Array of Things (AoT) project underway in the city of Chicago is shown [26]. The temperature, humidity, atmospheric pressure, light intensity, three-axis acceleration, and instantaneous sound of the city are measured and collected by Rabbit MQ, and the users are provided with measurement data in the form of REST APIs using Plenar.io of a Beehive data server. Plenar.io is a REST API interface to provide data from the Beehive data server [27].

In Figure 3(b), an example of using MQTT, which is a framework for a smart city platform called Fog Flow, is given [28]. In the study that proposed this, the data measured in the sensor was processed in Worker. After that, it was transmitted to a service through an MQTT broker so that it could be used in the platform.

In the previous two studies, the smart city platform was configured using HTTP and MQTT, respectively. However, in both studies, the edge where sensors and services are in contact was not fully utilized. The problems mentioned in Section 1, such as latency, overload, and service scalability, occur in the process of transmitting sensor data to a service through the edge. If a device located at the edge analyzes the data and automatically routes it, this will not only solve problems but also enable a smart city network to be configured more efficiently. In addition, as a smart city sensor corresponds to a smart home service gateway, it can improve the quality of smart home services. Therefore, in Section 3, we propose and design a Smart Edge Broker that can recognize the payload of transmission traffic and actively distribute data.

3. Smart Edge Broker

3.1. Smart Edge Broker for Smart City. The Smart Edge Broker is an intelligent broker, and it is located at an edge between a sensor network and service server to distribute traffic efficiently. It receives all the data from the sensors and analyzes who requested the data and how it routes to the appropriate location. Figure 4 is an example of a smart city platform when the Smart Edge Broker is introduced.

The boundary points between the sensor area and service area are divided by the edge. Eight smart homes are

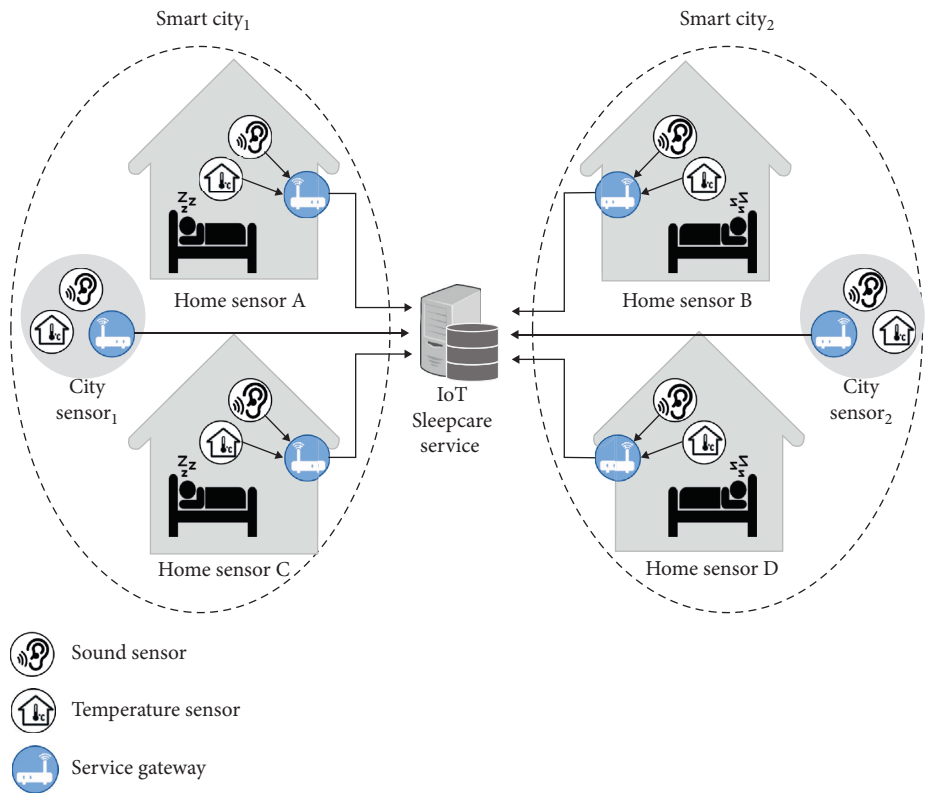

Figure 1: Example of a smart home system: IoT sleep care service.

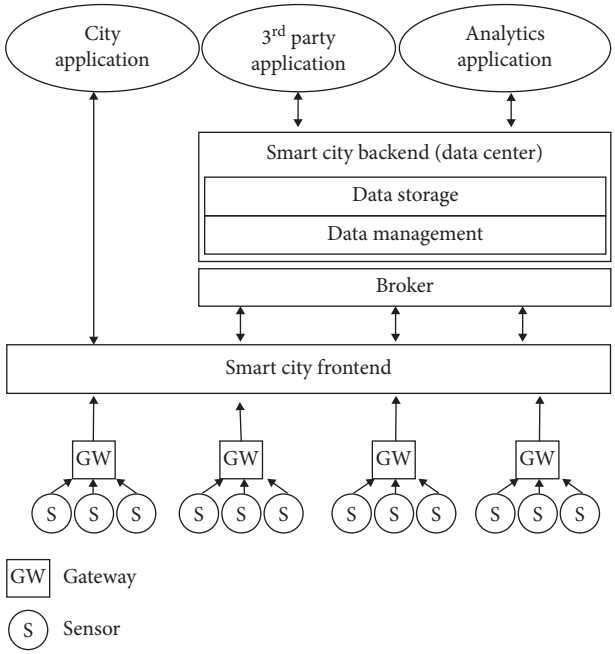

Figure 2: OneM2M smart city blueprint.

distributed in two areas in the sensor area, and the service area consists of one data center and two services. It is assumed that Svc_1 and Svc_2 use the sensors installed in the smart home system to provide different services and share the sensors in the area below. It is assumed that, at this time, Svc_1 needs the data measured in S_1, S_3, S_5, and S_7, and Svc_2 needs the data measured in S_2, S_4, S_6, and S_8. As the data center needs to collect and manage all data, it receives data from all sensors. The Smart Edge Broker mediates the exchange of data between sensors and services. It has features that reduce delays and overloads in the process.

The distance to the data center is far greater than the distance between the service and the sensor, so using sensors in multiple locations can provide efficient data exchange. Smart cities are often composed of services beyond the urban range, so the distance between services and sensors is often more than several kilometers. In this case, it naturally takes longer to send and receive data between the sensor and the service. Additionally, the response time to provide the service is long, so the quality of the service is reduced. Also, the wider the range, the more the sensors that are needed to cover the range, and the burden on the data center to collect and transmit data becomes greater. Furthermore, once a smart city has been configured, if you want to add new services or reconfigure existing services, you need to fix or modify the sensors installed around the city, which means new expenses. The Smart Edge Broker has an intelligent routing function to solve these problems.

It receives requests for desired data from each service and collects and analyzes the data from all sensors based on it. Based on the analysis results, it classifies the data requested by each service and routes it and transmits all the collected data to a data center. Figure 5 shows this process. Figure 5(a) shows the process by which a service receives data from a smart home and provides services.

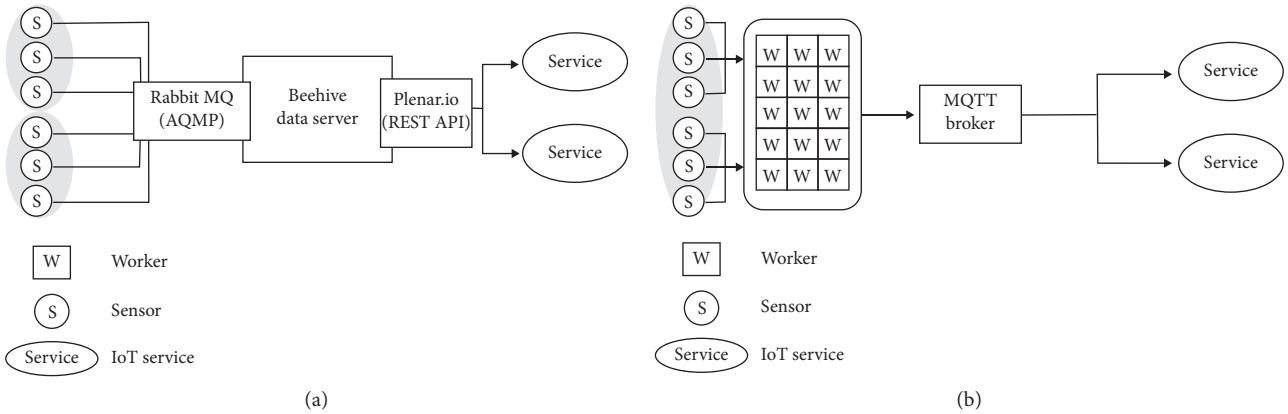

Figure 3: Examples of a smart city: (a) "Array of Things (AoT)" smart city platform; (b) "Fog Flow" smart city platform.

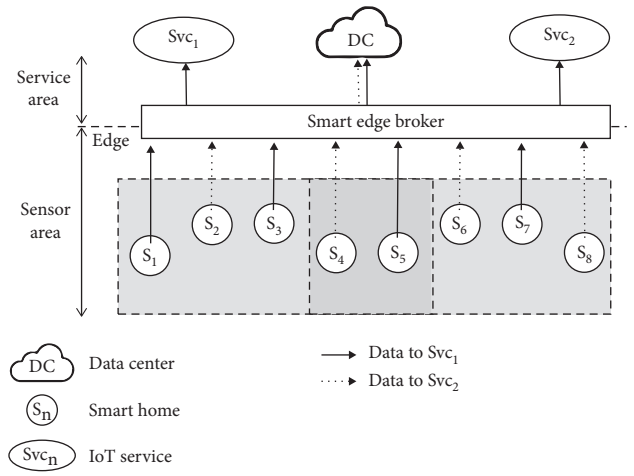

Figure 4: Smart city platform with a Smart Edge Broker.

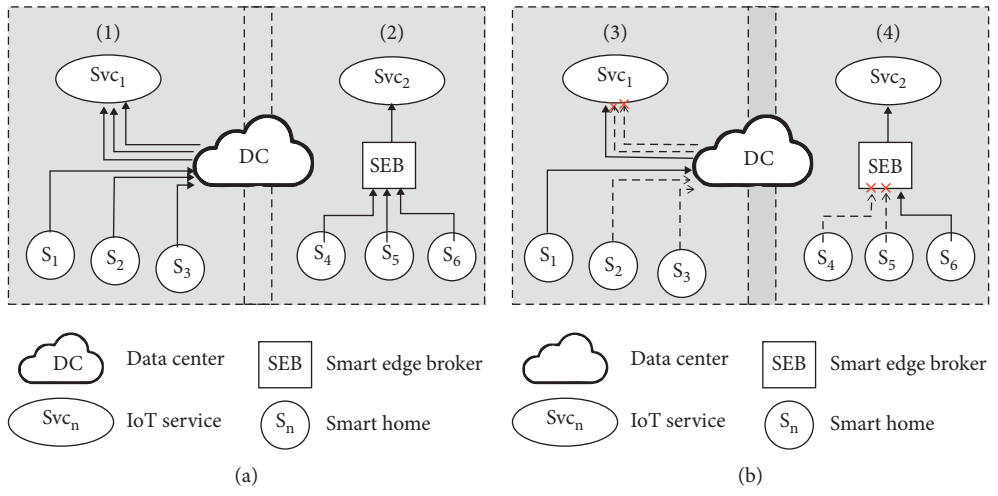

Figure 5: Scenarios for service routing in the Smart Edge Broker: (a) scenario for collecting data from sensors; (b) scenario for collecting selective data from sensors.

In the case of (1), if Svc_1 receives the data of S_1, S_2, and S_3, the smart home transmits data to the data center and the service requests data from the data center and receives it. At this time, if the Smart Edge Broker brokers the data between the smart home and service, as shown in (2), there is no need to purposely transmit the data to a remote data center. Svc_2

in (2) receives data of S_4, S_5, and S_6 in the same way as Svc_1 in (1), but the Smart Edge Broker does not need to go through a data center in place of the data center. In this case, the communication distance is reduced by the distance from the data center, which in turn reduces the burden on both ends of the smart home and service. Also, as the data center does not need to collect data, there is no traffic centralization.

Figure 5(b) shows a case where a service needs only some data. (3) is similar to (1), but Svc_1 requests only the data of S_1. In this case, Svc_1 has no choice but to filter only the necessary data after receiving all the data. In (4), Svc_2 requests the data of S_6, but unlike (3), the data of S_4 and S_5 are filtered in advance so that the Smart Edge Broker can provide services more efficiently because there is no need to consume resources to reconstruct the data in Svc_2.

If this is applied to the IoT sleep care service, a smart home can correspond to a sleep care device, and a service can correspond to IoT sleep care service. It collects the sleep data measured in each smart home and transmits it to the IoT sleep care service to provide services. At this time, the Smart Edge Broker provides intelligent data routing functions at the edge between the smart home and services, thereby transmitting data more efficiently. Furthermore, the advantages of the Smart Edge Broker are highlighted when providing a wide range of IoT services.

The Smart Edge Broker functions as part of the configuration of the smart city as described above. Figure 6 illustrates the specific benefits of the Smart Edge Broker. The Smart Edge Broker is installed on the left, the gateway is installed on the right, and a number of sensors and services are arranged around it.

First, the latency between the sensor and the service can be reduced. In general, the sensor must send data through a central data center to the service. For example, in Figure 6(a), when S_5 delivers data to the data center, Svc_2 finds and requests data from S_5, and the data center must transmit it. However, if you use the Smart Edge Broker on the left, the Smart Edge Broker automatically passes the data from the sensor to the service. Data from sensors do not have to go through the data center. In this case, the movement path of that much data is reduced. When sending data from S_1 to Svc_1 without knowing each other's addresses as usual, S_1 sends data to the Smart Edge Broker, not to the data center. Svc_1 can benefit greatly from the distance by receiving data using the Smart Edge Broker, which is much closer than the data center. This leads to a reduction in data latency and an improvement in response time.

Secondly, the Smart Edge Broker can distribute the load on the data center. For example, as shown on the right side of Figure 6(b), when multiple sensor data are used in a service, multiple sensors transmit data to the data center, and the data center must also aggregate the data and transmit it to the service. The data center is responsible for the computing power that receives and processes the data from S_4, the data from S_5, and the data from S_6 and sends them to Svc_2. However, if there is a Smart Edge Broker, as shown on the left side of Figure 6(b), all the burden is borne by the Smart Edge Broker, which saves computing power in the data center.

Finally, the scalability of the service can be further extended. Unlike Figure 6(a) or 6(b) if a sensor knows the address of a service, it can send data directly to the service without having to go through the data center. This saves you a round trip to the data center, but it can be fatal when you change services. For example, in the case of the right side of Figure 6(c), Svc_2 originally used data of S_4, S_5, and S_6. But, recently, data from S_4 and S_5 are no longer needed. In this case, it is necessary to remove S_4 and S_5 or filter the data from Svc_6. However, the Smart Edge Broker can filter out unnecessary data from the Smart Edge Broker. On the left side of Figure 6(c), Svc_1 originally used all data from S_1, S_2, and S_3, but now uses only S_1; thus, the unnecessary data is filtered by the Smart Edge Broker.

3.2. Message Definition for the Smart Edge Broker. The Smart Edge Broker receives data requests or receives data from services and sensors through messages. A query is a criterion to classify the data transmitted from the sensor and is a medium to select the data to receive. Therefore, a query is the most important element among the elements of the Smart Edge Broker. A topic also explains data information, so it becomes an important clue for routing data together with a query.

When processing, the query attached to a topic is stored separately. When checking receivers, the separately stored query conducts an additional search for whether it is suitable for the query in the message and filters data according to the query. At this time, it must transmit the data only if the data is suitable. If this is expanded further, it is possible to configure the data transmission process of an IoT Sleep care service, as shown in Figure 7.

As shown in Figure 7, there are three sleep sensors on the left and the server and application that provide the IoT sleep care service on the right. The IoT sleep care service provides a tailored sleep care service by receiving personal sleep data from sleep sensors in a smart home. To that end, the server and application first transmit a QueryReq message to the Smart Edge Broker requesting data. The server needs the data for "KIM" in the topic called "/smarthome/sleep," and the application needs the data for "LEE" on the same topic. This need is transmitted to the Smart Edge Broker by containing the topic and query of the desired data in the QueryReq message. After that, when personal sleep data is transmitted through a DataPub message, the Smart Edge Broker compares the topic and query of the transmitted QueryReq message with the topic and data of the DataPub message and filters and delivers them to the appropriate target.

There is a fixed rule in the form of a query statement in each message. According to this rule, the Smart Edge Broker analyzes the payload and processes it. At the end of the topic, "query?" is added after the identifier. For example, only the data for the user called "KIM" are wanted in the topic called "/RFID," and the topic type is defined in the form of "/smarthome/sleep/query?user = kim." DataPub messages are configured in the JSON form to search data. In other words, it must be in the form of {"Key": "Value"}, but Key

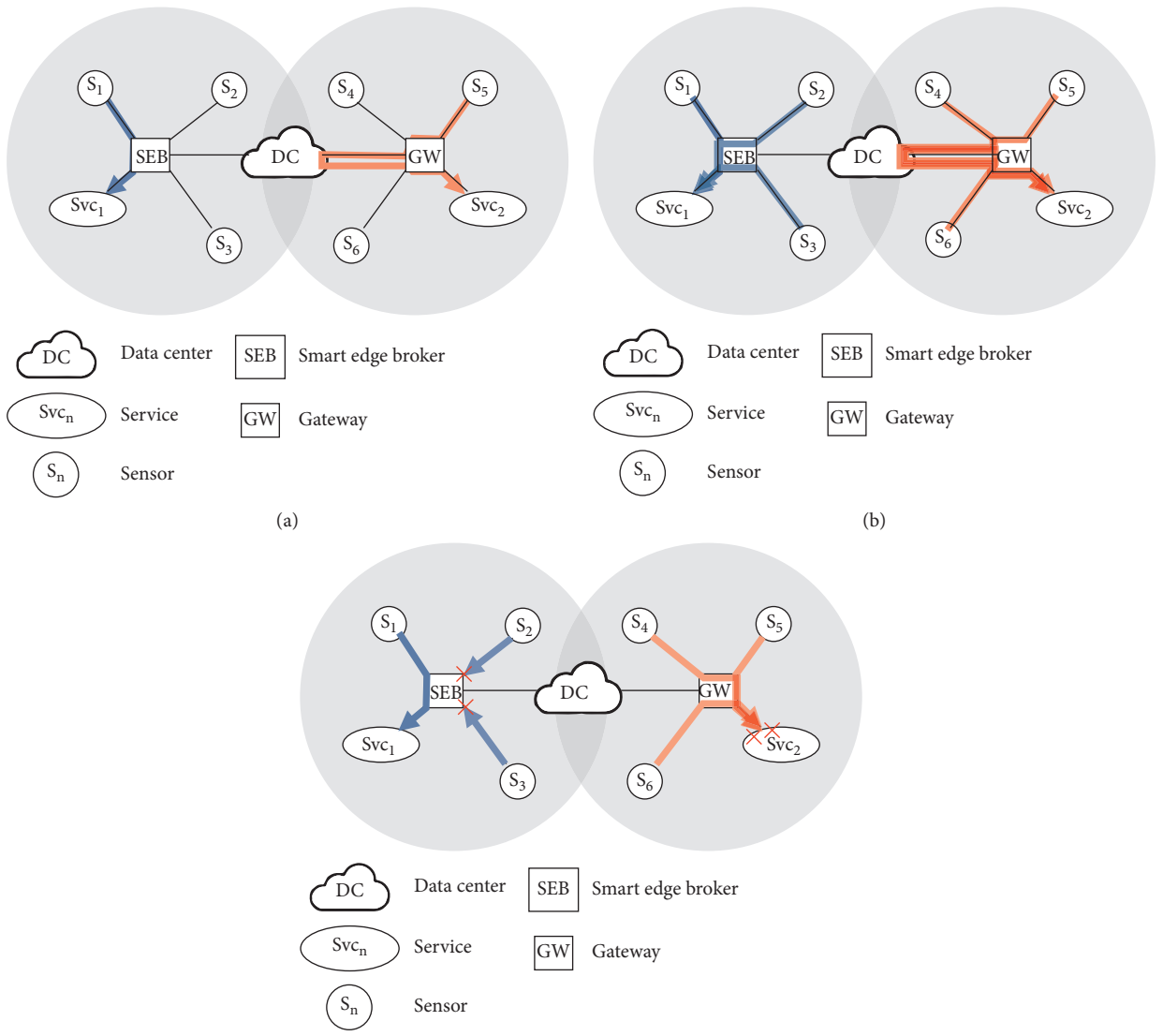

Figure 6: Scenarios for service routing in the Smart Edge Broker: (a) scenario for collecting data from sensors; (b) scenario for collecting selective data from sensors; and (c) scenario for filtering unnecessary data from sensors.

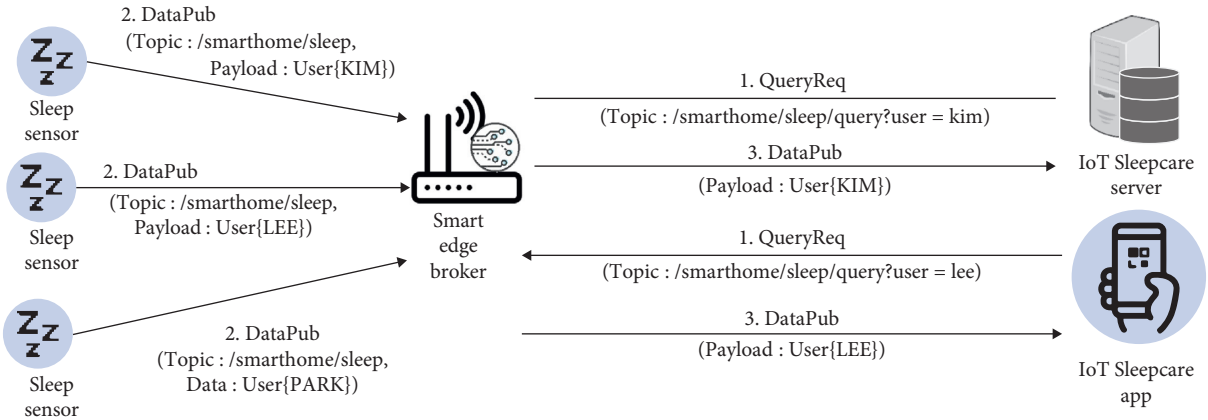

Figure 7: Function of the Smart Edge Broker.

specifies the search target of the query so that Value has search values. To process such queries, the following specific process is needed. If an IoT sleep care service transmits data through the Smart Edge Broker, it must transmit messages in the form as shown in Tables 1 and 2.

As shown in Tables 1 and 2, the message transmitted from the Smart Edge Broker consists of a fixed header, variable header, and payload. Also, the message is divided into two types and is defined as QueryReq transmitted when a service requests data from the sensor and DataPub, which transmits data from the sensor to the service. A fixed header consists of two types: an MQTT control packet type for using the MQTT protocol and the remaining length that indicates the length of a message. For the variable header, there is a packet identifier in the QueryReq message, and there are topics with packet identifiers in a DataPub message according to the message type. The payload also varies depending on the message. There are topics and queries in a QueryReq message, and there is the payload of the data to be transmitted in a DataPub message.

3.3. Smart Edge Broker Algorithm. To express the process in the previous example in detail, the operating algorithms of the IoT sleep care service, the Smart Edge Broker, and smart home system can be configured, as shown in Figure 8.

First, the IoT sleep care service on the right transmits the data request message to the Smart Edge Broker through a QueryReq message. The smart home system on the left is in the home of a user who receives an IoT sleep care service and transmits the sleep environment measured by sleep sensors to the Smart Edge Broker in the middle through a DataPub message. The Smart Edge Broker analyzes the message in the middle and routes the data message to the IoT sleep care service through the DataPub message. The Smart Edge Broker consists of two modules: a query analyzer and a context-aware filter. These two modules process QueryReq and DataPub messages, as shown in Figure 9.

The query analyzer separates the payload of a topic and query from a QueryReq message transmitted from an IoT sleep care service and stores them in a message database so that they can be used for filtering later. The context-aware filter receives a DataPub Message from the smart home system and analyzes the topic and payload data to check whether the data are suitable.

After receiving a QueryReq message, it checks whether the query is included in the topic name. If it is included, it separates the topic name into topic and query and stores them in a message database. This is to use it as a criterion for filtering when a DataPub message is received later. However, if the query is not included, it does not perform any task.

Also, when a DataPub message is received, the context-aware filter filters data by checking whether the topic name of the message is in the message database. It retrieves the query and topic stored in the message database during the filtering process. This is carried out to use the query and topic as a criterion for analyzing the payload of a received DataPub message. If the two payloads do not match, it transmits a message to QueryReq. If filtering is not necessary because the payload is not registered in the message database, it transmits a message to all those registered. If it is registered but not suitable for the query, it does not transmit a message.

Table 1: Smart Edge Broker message definition: QueryReq message.

Header	Byte	Description
Fixed	Byte 1	MQTT control packet type (8)
	Byte 2	Remaining length
Variable	Byte 1	Packet identifier MSB
	Byte 2	Packer identifier LSB
	Byte 3	No properties
Payload	Byte 1	Payload length MSB
	Byte 2	Payload length LSB
	Byte 3. . . n	**Topic filter + query**
	Byte $n+1$	Subscription options (1)

Table 2: Smart Edge Broker message definition: DataPub message.

Header	Byte	Description
Fixed	Byte 1	MQTT control packet type (3)
	Byte 2	Remaining length
Variable	Byte 1	Packet identifier MSB
	Byte 2	Packer identifier LSB
	Byte 3. . .n	**Topic**
	Byte $n+1$	Payload length MSB
	Byte $n+2$	Payload length LSB
	Byte $n+3$	Topic filter + query
Payload		Payload

For a better understanding, the operation of the Smart Edge Broker is shown in a sequence diagram in Figure 10. This sequence diagram shows the data exchange process between the IoT Sleepcare service and the smart home system that provides the Sleepcare service.

First, the IoT Sleepcare service on the left sends a QueryReq message to the Smart Edge Broker. This message is for requesting data of the user "kim" among data with the topic "sleep." The Smart Edge Broker receives the message and analyzes the message using a query analyzer. It analyzes the topic filter and query contained in the message and saves it in the message database. The saved topic filter and query are used to screen out data later.

After that, the smart home system sends a DataPub message to the Smart Edge Broker to send the measured data to the service. The DataPub message contains a payload along with a topic and is divided into a "User" item and "Data" item to use Query. The "User" item in the payload contains the user's name "kim," and the "Data" item contains the sleep information measured during the user's sleep.

The Smart Edge Broker analyzes the received Datapub message through the context-aware filter. The context-aware filter loads topics and queries from the message database to analyze the received messages. These topics and queries are extracted from previously received QueryReq messages, and they can be mixed with topics and queries received from other services besides the IoT Sleepcare Service. The context-aware filter analyzes the Datapub message and checks whether there

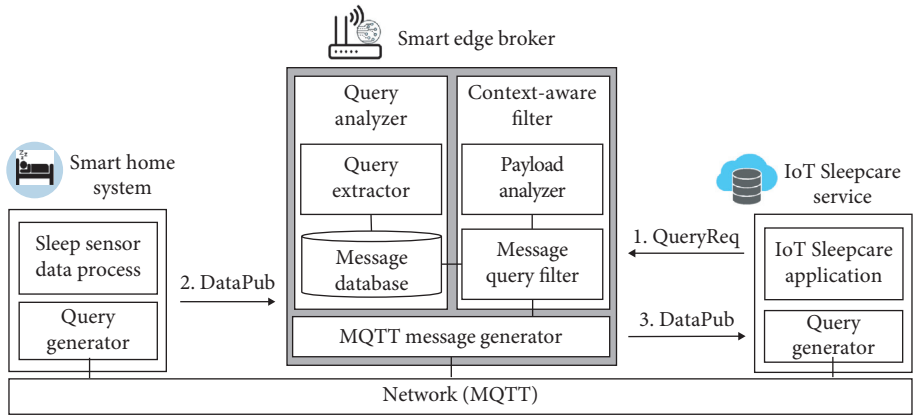

Figure 8: Function of the Smart Edge Broker.

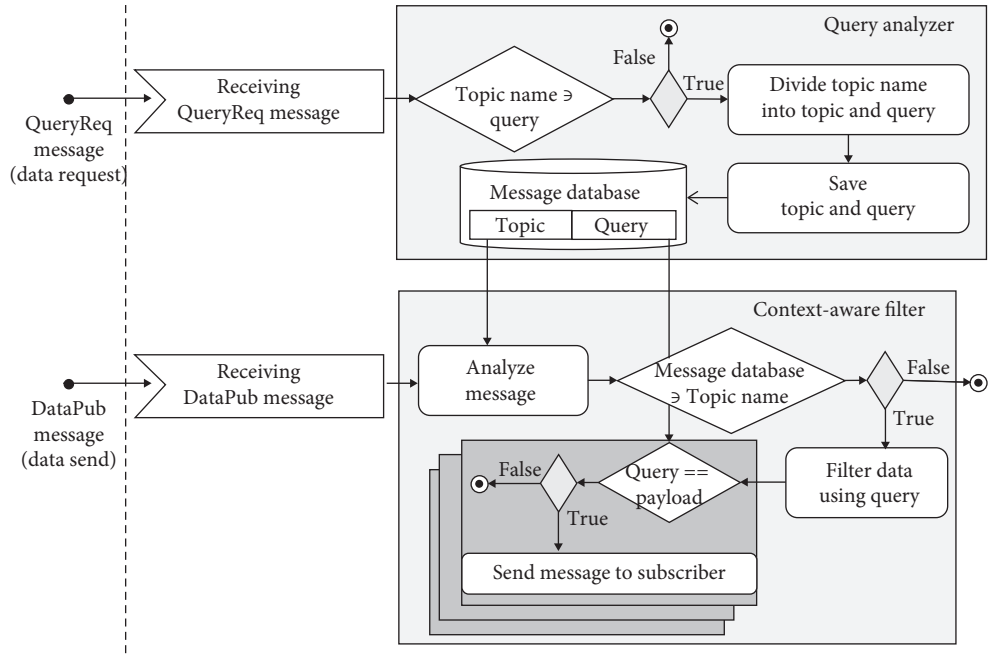

Figure 9: Smart Edge Broker activity diagram.

is a corresponding request among the services that sent the QueryReq message. If there is a QueryReq request that matches the DataPub message, the Datapub message is delivered to the service that sent the QueryReq message. In this process, the data in the message can be processed or sent as is.

This series of processes allows the IoT Sleepcare service to collect sleep data from the smart home system and provide sleep care services to users through the collected data. This can be performed in 1: n, and the Smart Edge Broker can process multiple messages. The Smart Edge Broker mediates this data delivery and takes over the role of the data center. Additionally, a sensor or smart home that transmits data through the Smart Edge Broker can reduce the burden on the communication process since there is not a concern about data transmission, and the IoT service also needs to connect with multiple sensors to collect data. It can be connected to one Smart Edge Broker without the burden of data communication.

We implemented and tested the Smart Edge Broker in the next chapter to see the abovementioned effects in the smart city environment.

4. Experiment and Evaluation

4.1. Testbed and Its Scenarios. To check whether the Smart Edge Broker can solve the problems that may occur when it is applied to the configuration of a smart city, we configured two experiments: a latency measurement experiment and a load measurement experiment.

These experiments were conducted under two conditions in the same experimental environment. Under the first condition, the IoT sensor transmitted data to the IoT service server through the Smart Edge Broker. At this time, the data to be transmitted were randomly set so that the Smart Edge Broker could filter 30% of data. Under the second condition, the IoT sensor transmitted data to the IoT service server through the data

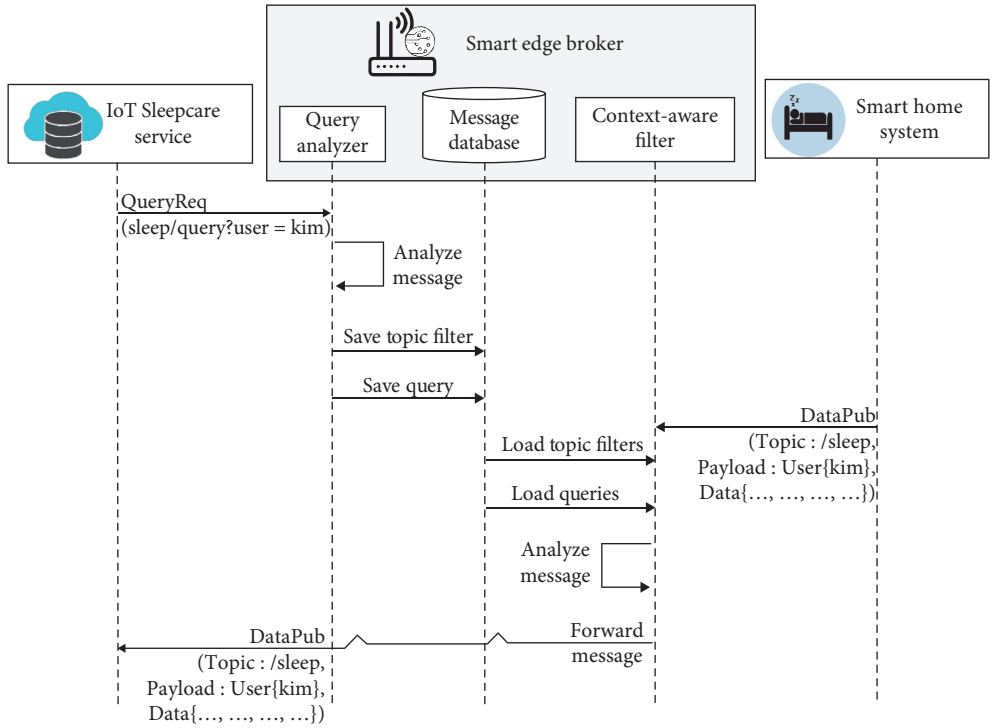

Figure 10: Smart Edge Broker sequence diagram.

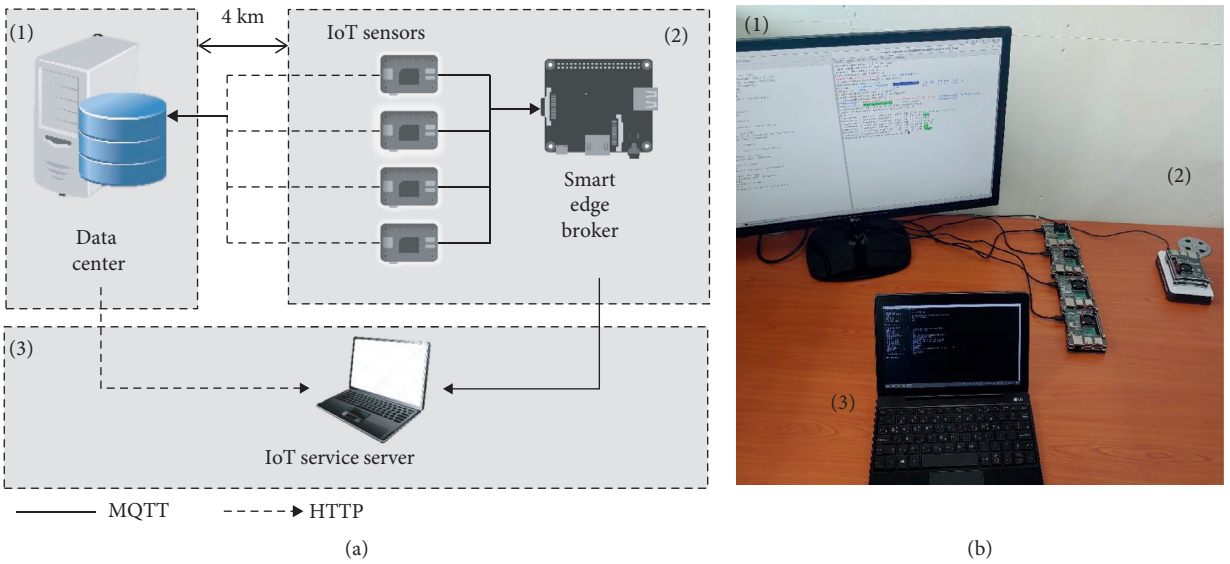

Figure 11: Smart Edge Broker experimental testbed: (a) configuration of the testbed; (b) actual configuration of the testbed.

center. In both cases, four IoT sensors simultaneously transmitted 140 bytes of data, 10 times per second, and a total of 4000 data were transmitted for 100 seconds. The measurement started when all the sensors started to transmit the data.

Experiments are proceeded in an environment as shown in Figure 11.

First, we divided the experimental environment into three areas, as shown in Figure 10: an area with a data center (1), an area with IoT sensors and the Smart Edge Broker (2), and an area with an IoT service server (3), and the network was divided by each area. For the data center, a desktop PC was installed about 4 km away geographically, and CentOS 7.6 was used. For the IoT sensor and Smart Edge Broker, a Raspberry Pi 3 B + model was used, and Raspbian Buster 4.19 was used as the operating system. For the IoT service server, a laptop PC with the Windows 10 operating system was used.

5. Results and Evaluation

The experimental results are as follows. Latency measurement experience measured Round Trip Time (RTT), which is the time when each IoT sensor sends data to the IoT service

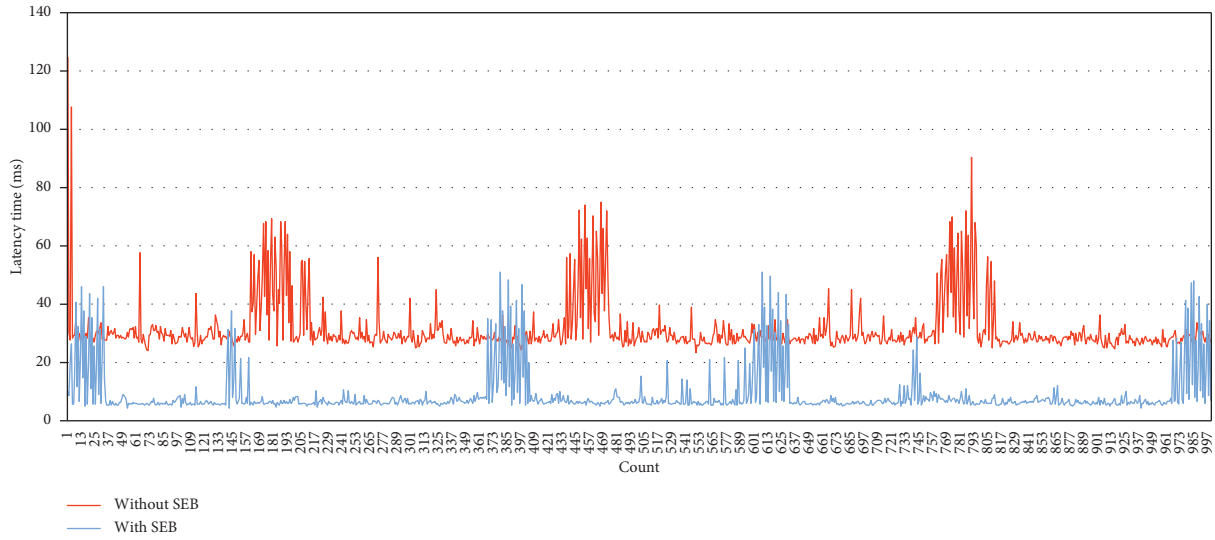

Figure 12: Smart Edge Broker experiment results: latency time between the source sensor and destination sensor.

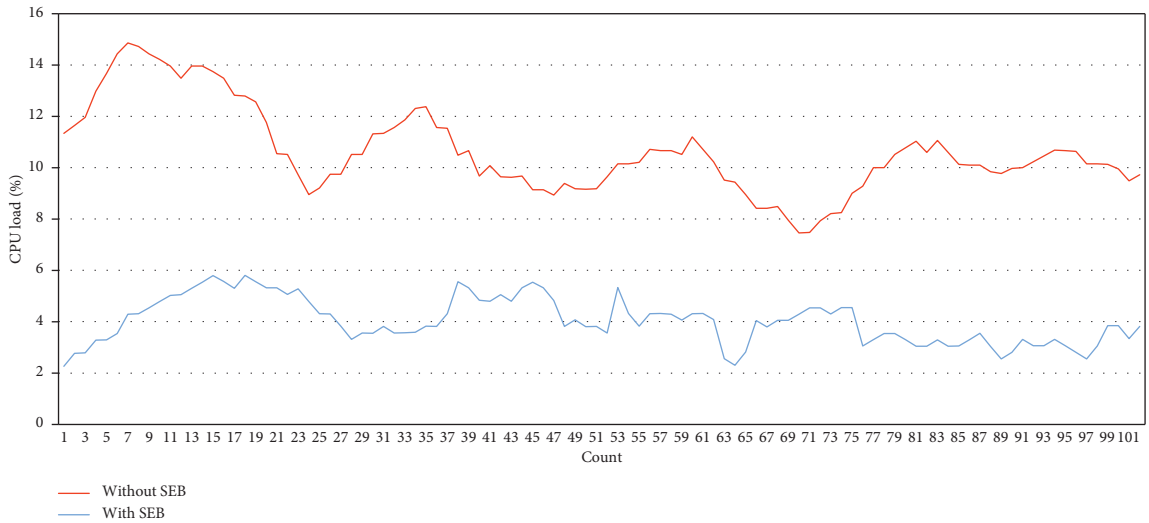

Figure 13: Smart Edge Broker experiment results: CPU load rate for the data center.

server and receives responses. Graphs and representation of the measuring results are shown in Figure 12. The *Y*-axis on the left is latency time for RTT, and the *X*-axis represents the count of the message.

For latency, the time spent until each message reached the service was measured. When the Smart Edge Broker was not used (without the SEB), the average latency was 31.41 ms. In contrast, when the Smart Edge Broker (with the SEB) was used, the average latency was 8.83 ms, with a decrease of about 72%. This was mainly caused by network latency that occurred in the process of transmitting data to the IoT service server through the data center, which was 4 km away from the IoT sensor. In both experiments, the latency instantaneously increased due to the declining function of Raspberry Pi 3, which was used as an IoT sensor. As a result of analysis, it was found that performance throttling was activated during the generation and transmission of messages due to the increased temperatures of the Cortex-A53 CPU cores mounted on Raspberry Pi 3. Since this was not a network problem but a device performance problem, it did not affect the experiment's results.

For CPU usage of the data center, the CPU usage of the server process was measured at 1-second intervals during message transmission. As seen in Figure 13, without the SEB, 10.63% was recorded on average, indicating that the usage was high because the data center received sensor data and simultaneously transmitted it to the IoT service server. In contrast, the SEB did not go through the data center to transmit data to the IoT service center, so the roles of the data server were reduced by one. As a result, 4.02% was recorded on average, and consumption was reduced by 63% in the same environment.

For the CPU usage of the service server, the CPU usage of the server process was measured at 1-second intervals during message transmission. As shown in Figure 14, without the SEB, 15.95% was recorded and with the SEB,

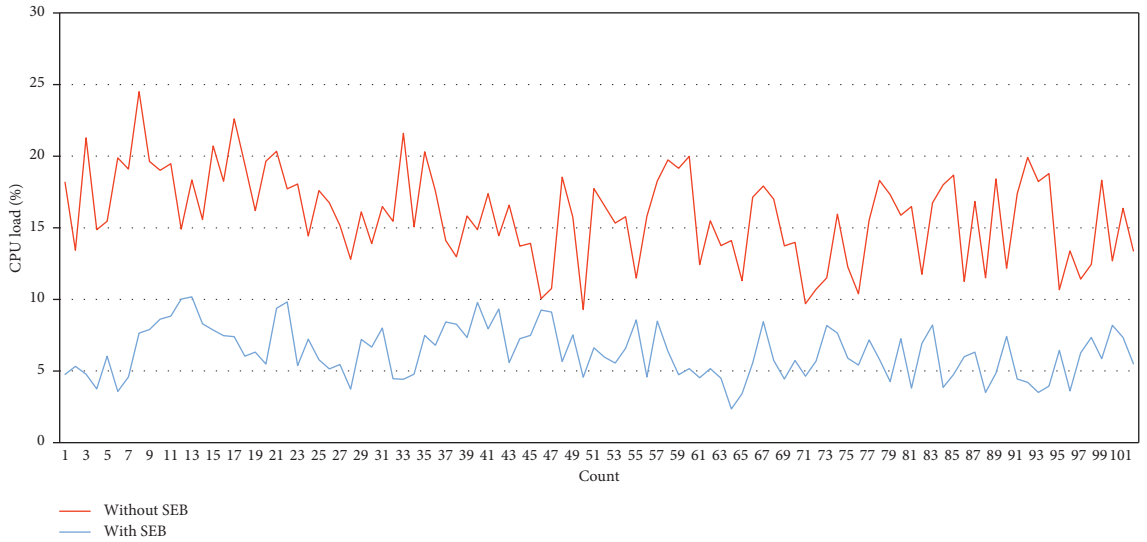

Figure 14: Smart Edge Broker experiment results: CPU load rate for the service server.

6.28% was recorded on average, indicating a reduction effect of 60%. The amount of data that needs to be processed in the IoT service server was reduced because the Smart Edge Broker filtered 30% of the data.

6. Conclusions

In this paper, by proposing a Smart Edge Broker, we could solve the latency and overload problems that may occur during the process of using an IoT service through a smart home system and smart city platform. This is a method of analyzing and filtering the data using topic and query and routing it to the proper location.

To find out whether the proposed Smart Edge Broker could control network traffic by analyzing the payload of data, we conducted experiments to compare two cases: when the Smart Edge Broker was used and when it was not used. In the latency measurement experiment, the latency was reduced by 72% when the Smart Edge Broker was used compared to when it was not used, and the CPU load rate was reduced by 63% in the load measurement experiment. It was also found that sensors or services could be conveniently added to the existing system using the characteristics of the Smart Edge Broker, even if a sensor or a service was added to an existing system, which indicates service scalability. In addition, the Smart Edge Broker works on the application layer, so that engineers can easily handle it.

However, further studies on security are needed. When searching for items corresponding to the Smart Edge Broker according to the 10 recommendations for security specified by the OWASP [29], there may be the possibility of confidentiality and integrity breaches for injection, broken authentication, and sensitive data exposure. In the case of a query used in the Smart Edge Broker, as it is somewhat similar to SQL, it is possible to acquire data that are not allowed or the data of a system area. Therefore, there is a need to improve the Smart Edge Broker with continuous studies in the future.

Data Availability

The data used to support the findings of this study are available at https://github.com/woodencatty/SmartEdgeBrokerExperiment.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Technology development Program funded by the Ministry of SMEs and Startups (MSS, Korea) (Grant no. S2798371). This work was supported by the Gachon University Research Fund of 2019 (GCU-20190787).

References

[1] B. Li and J. Yu, "Research and application on the smart home based on component technologies and Internet of things," *Procedia Engineering*, vol. 15, pp. 2087–2092, 2011.

[2] Y. Pang and S. Jia, "Wireless smart home system based on zigbee," *International Journal of Smart Home*, vol. 10, no. 4, pp. 209–220, 2016.

[3] Y. Yin, L. Chen, Y. Xu, and J. Wan, "Location-aware service recommendation with enhanced probabilistic matrix factorization," *IEEE Access*, vol. 6, pp. 62815–62825, 2018.

[4] Y. Yin, Y. Xu, W. Xu, M. Gao, L. Yu, and Y. Pei, "Collaborative service selection via ensemble learning in mixed mobile network environments," *Entropy*, vol. 19, no. 7, p. 358, 2017.

[5] S. Yoon and J. Kim, "A study on the user's value of the smart home service in the Internet of things technology," *International Journal of Future Generation Communication and Networking*, vol. 10, no. 6, pp. 65–80, 2017.

[6] W. Jin and D. Kim, "Seamless multimedia service mechanism based on content profile using user and device ID for personal mobility in smart home," *International Journal of Grid and Distributed Computing*, vol. 11, no. 1, pp. 45–56, 2018.

[7] M. Talal, A. Zaidan, B. Zaidan et al., "Smart home-based IoT for real-time and secure remote health monitoring of triage and priority system using body sensors: multi-driven systematic review," *Journal of Medical Systems*, vol. 43, no. 3, 2019.

[8] H. Han, J. Jo, Y. Son, and J. Park, "Smart sleep care system for quality sleep," in *Proceedings of the 2015 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 393–398, Jeju, Korea, October 2015.

[9] K. Skouby and P. Lynggaard, "Smart home and smart city solutions enabled by 5G, IoT, AAI and CoT Services," in *Proceedings of the International Conference on Contemporary Computing and Informatics (IC3I)*, pp. 874–878, Mysore, India, November 2014.

[10] G. Jia, G. Han, H. Rao, and L. Shu, "Edge computing-based intelligent manhole cover management system for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1648–1656, 2018.

[11] Y. Yin, F. Yu, Y. Xu, L. Yu, and J. Mu, "Network location-aware service recommendation with random walk in cyber-physical systems," *Sensors*, vol. 17, no. 9, p. 2059, 2017.

[12] F. Li and B. Zheng, "Design of the smart city planning system based on the Internet of things," *International Journal of Smart Home*, vol. 10, no. 11, pp. 207–218, 2016.

[13] A. Gaur, B. Scotney, G. Parr, and S. McClean, "Smart city architecture and its applications based on IoT," *Procedia Computer Science*, vol. 52, pp. 1089–1094, 2015.

[14] S. Sarkar and S. Misra, "Theoretical modelling of fog computing: a green computing paradigm to support IoT applications," *IET Networks*, vol. 5, no. 2, pp. 23–29, 2016.

[15] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog computing: principles, architectures, and applications," in *Internet of Things*, pp. 61–75, Morgan Kaufmann, Burlington, MA, USA, 2016.

[16] J. Ahn and B. M. Lee, "Enhanced smart edge broker using fog computing for smart homes," *International Journal of Artificial Intelligence and Applications for Smart Devices*, vol. 7, no. 1, pp. 1–6, 2019.

[17] V. Ricquebourg, D. Menga, D. Durand et al., "The smart home concept: our immediate future," in *Proceedings of thr 1st IEEE International Conference on E-Learning in Industrial Electronics*, pp. 23–28, Hammamet, Tunisia, December 2006.

[18] J. H. Choi, U. G. Kang, and B. M. Lee, "Sleep information gathering protocol using CoAP for sleep care," *Entropy*, vol. 19, no. 9, p. 450, 2017.

[19] D. h. Lim and B. H. Rhee, "Design study of the U-city home network architecture of cloud computing," *International Journal of Smart Home*, vol. 7, no. 6, pp. 145–156, 2013.

[20] H.-J. Lee, K.-H. Kim, and Y.-H. Kim, "Wireless sensor network-based 3D home control system for smart home environment," *International Journal of Smart Home*, vol. 10, no. 1, pp. 159–168, 2016.

[21] Y. Choe, M. Jang, and S. Kim, "System design for a urban energy monitoring and visualization environment using ubiquitous sensor network and social sensor networking," *Journal of the HCI Society of Korea*, vol. 5, no. 2, p. 7, 2010.

[22] S. Vergura, "Smart city, sustainable mobility, home-work mobility: data analysis and actions," *Renewable Energy and Power Quality Journal*, vol. 13, pp. 768–773, 2015.

[23] S. Pellicer, G. Santa, A. Bleda et al., "A global perspective of smart cities: a survey," in *Proceedings of the Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 439–444, Taichung, Taiwan, July 2013.

[24] M. Battarra, M. Consonni, S. Domenico, and A. Milani, "Storm clouds platform: a cloud computing platform for smart city applications," *Journal of Smart Cities*, vol. 2, no. 1, pp. 14–25, 2016.

[25] O. Elloumi, T. Carey, J. Blanz et al., "OneM2M white paper: smart cities done smarter," OneM2M, 2018.

[26] C. Catlett, P. Beckman, R. Sankaran, and K. Galvin, "Array of things: a scientific research instrument in the public way: platform design and early lessons learned," in *Proceedings of the 2nd International Workshop on Science of Smart City Operations and Platforms Engineering SCOPE'17*, pp. 26–33, Pittsburgh, PA, USA, April 2017.

[27] C. Catlett, I. Foster, T. Malik et al., "Plenario: an open data discovery and exploration platform for urban science," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 37, 2014.

[28] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa, "FogFlow: easy programming of IoT services over cloud and edges for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 696–707, 2018.

[29] OWASP, *Category: OWASP Top Ten Project—OWASP*, OWASP, MA, USA, 2017.