# Prototype implementation of downward transfer method by tunneling for a large-scale data collection system using MQTT

Fuya Aoki
*Kanazawa Institute of Technology*
Japan
c6101709@planet.kanazawa-it.ac.jp

Koichi Ishibashi
*Kanazawa Institute of Technology*
Japan
k_ishibashi@neptune.kanazawa-it.ac.jp

Tetsuya Yokotani
*Kanazawa Institute of Technology*
Japan
yokotani@neptune.kanazawa-it.ac.jp

***Abstract*— Message queuing telemetry transport (MQTT) has recently attracted attention as an important communication method in the Internet of Things (IoT). Research and development have been conducted over time to realize IoT services using MQTT; however, when the number of connected IoT devices increase or an IoT system is deployed widely, issues such as scalability and interoperability exist. We have previously proposed an efficient communication method for large-scale data collection systems with multiple MQTT brokers, such as smart street lighting systems, called the downward transfer method by tunneling, in which connection management is by a server and transfer from a server is by tunneling. In other words, the server manages the information of the broker to which the sensor is connected, called sensor location information; then, when the server notifies data to a sensor, it transfers a message with the sensor location information and original data to a specific broker, called a top broker, using tunneling technology. The top broker forwards the received message to a bottom broker according to the extracted sensor location information via de-tunneling. Because the top broker need not to manage a location information for each sensor, the downward transfer method by tunneling can reduce memory resources required for the connection management of the top broker. In addition, it can prevent an increase of traffic due to memory overflow. In this paper, we implemented a prototype of this downward transfer method by tunneling and evaluated the amount of traffic. As a result, we confirmed that the proposed downward transfer method by tunneling is effective and available for a large-scale data collection system.**

***Keywords—IoT, Large-scale data collection, Wireless sensor network, Connection management***

## I. Introduction

Recently, IoT (Internet of Things) devices have spread rapidly, connecting not only Internet-connected terminals such as computers and smartphones but also "things" such as home appliances, automobiles, factory equipment, and streetlights to the network. They have become indispensable in our daily lives as well as in various industries. The number of IoT devices in the world is increasing every year, and is expected to reach several hundred billion in the next few years [1]. Under these trends, it is desirable for IoT services to collect actual data from a wide variety of IoT devices over a wide area and to notify control data to specific devices as needed. To realize such an IoT system which IoT devices are deployed over a wide area, researches and developments are being conducted considering various aspects, such as the construction of a wireless sensor network (WSN) to accommodate IoT devices with communication functions and an IoT core network consisting of gateways for WSNs [2][3][4][5]. In addition, ISO/IEC JTC1 /SC41 classifies the use cases of IoT services and summarizes the requirements for communication platforms in terms of communication type and QoS (Quality of Service). It proposes an IoT data exchange platform for various IoT services to reduce the amount of communication compared with IoT systems built on conventional networks. In the field of IoT, a communication protocol called MQTT (Message Queuing Telemetry Transport) has attracted attention as an important communication method. MQTT is an asynchronous and lightweight communication protocol consisting of publishers that transmit messages, subscribers that receive messages, and a broker that acts as a server to mediate messages. The publisher and subscriber functions operate independently: the former transmits data to the system, and the latter receives data from the system (Fig. 1). However, the current MQTT specification defines an operation using a single broker, which makes its application to a large-scale system over a wide area challenging. Therefore, in order to deploy an IoT system with multiple MQTT brokers, we have previously proposed an efficient downward transfer method by using tunneling [6]. In the proposed method, it is possible to easily migrate from MQTT applications on IoT devices and server by adding some sub-applications for the downward transfer method. In this paper, we implemented a prototype of the downward transfer method by tunneling, and evaluated reduction of traffic on an IoT core network by the proposed method.

The rest of this paper is organized as follows. In Section II, we describe the challenges faced by large-scale data collection systems. Related works are presented in Section III. Section IV gives an overview of our previously proposed downward transfer method by tunneling. The prototype implementation is discussed in Section V, the experiment and evaluation in Section VI, and finally, the conclusion in Section VII.

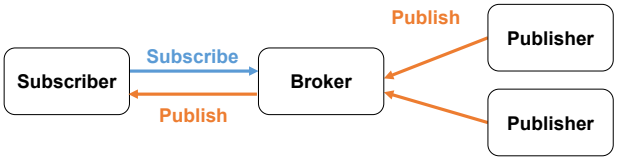


Fig. 1. Message queuing telemetry transport.

## II. Challenges in Large-Scale Data Collection System using MQTT

As an example of a large-scale data collection system, we focus on smart street lighting systems. Recent advances in smart street lights equipped with a means of communication with controllable LEDs have led to the development of remote control systems in which street lights are managed and maintained by a central server [7]. Smart street lighting systems have become an important research field because they

are expected to provide efficient maintenance of street lights as well as additional or novel services. For example, by visualizing the status and keep-alive status of street lights on a central server, it is possible to reduce energy consumption through optimal dimming control that takes into account the effects of buildings, trees, and weather. It is also possible to improve the efficiency of equipment maintenance and reduce maintenance costs such as labor costs. Additional services such as visual navigation are also being considered to control the flow of people smoothly by remotely controlling the lighting color and frequency of street lights. Another candidate for a new service is the visualization of sensor information through environmental monitoring (Fig. 2).

In a smart street lighting system, small data such as sensor information are exchanged between the smart street lights and a central server, but conventional communication has a large data transfer overhead. It is therefore effective to apply MQTT, which is a communication protocol for IoT with a small overhead and lightweight. However, the smart street lighting system consists of several hundred thousand sensors. Therefore, scalability of the broker becomes an issue in systems that utilize MQTT because the current MQTT specification defines an operation using a single broker. In other words, when data collection by the single server is considered, there are resource congestion and overflow issues at the single broker to which the server connects due to the concentration of messages from sensors and from the server on the single broker. Then it is expected for an IoT system with multiple brokers. However, on the IoT system with multiple brokers, issue of memory overflow on the broker to which the server connects to remains because the broker needs to maintain location information of sensors for support of downward transfer from the server to sensors.

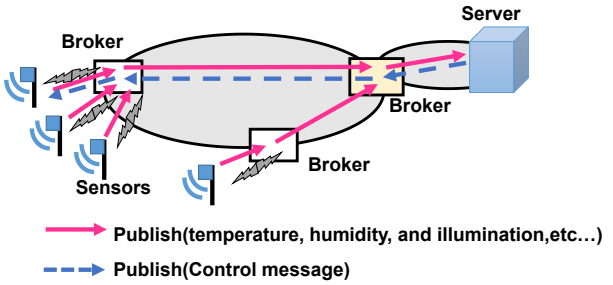


Fig. 2. Smart street lighting system.

## III. Related Work

Regarding the realization of IoT systems using MQTT to deploy large-scale systems, research is being conducted to evaluate the performance and to propose methods of cooperation with multiple brokers. For example, [8][9] evaluate the performance of various MQTT implementations against brokers. According to the benchmark in [8], the number of terminals that can be processed by a single broker is less than 100,000. The smart street light system, which is the target of this research, is a large-scale data collection system, and there are still some issues to be solved before it can be realized with a single broker. MQTT systems with multiple brokers have been investigated in many studies [10][11][12]. In [10], MQTT with a spanning tree of brokers on the network (MQTT-ST) is proposed for building a distributed network with multiple MQTT brokers. MQTT-ST enables the data collection from a wide area. However, MQTT-ST has issues such as traffic overhead due to the need for periodic information exchange with the broker.

In [11], a scalable and low-cost MQTT broker clustering system is proposed to handle a large number of IoT devices. In this clustering system, MQTT clients and multiple MQTT brokers are connected by a load balancer to distribute network traffic to the MQTT brokers. Therefore, compared to a single broker, the load on each broker is reduced and the throughput of the entire clustering system is increased, thereby reducing the CPU utilization of each broker.

In [12], MQTT brokers are placed at each network edge to handle data with the characteristic of "edge heavy," where objects at the network edge of an IoT system generate a large amount of data. To coordinate these multiple MQTT brokers, they propose a new mechanism called the ILDM (Interworking Layer of Distributed MQTT brokers). An ILDM node placed between a broker and a client not only relays MQTT clients and brokers as a proxy but also connects to other ILDM nodes to coordinate multiple brokers. As shown in [11][12], the deployment of systems with multiple brokers is considered in many places for building large-scale systems. However, there is still the issue that the traffic of the entire system will increase.

Research has been conducted in many places to realize a large-scale IoT system using MQTT. However, according to the performance evaluation of a single broker, it is impossible to realize a large-scale data collection system with a single broker. In addition, through research of the coordination of multiple brokers, when we consider communication from the server to the sensor for data notification, the broker becomes overloaded owing to the subscription from many sensors. Increasing the traffic of an entire system consisting of multiple brokers also remains an issue.

Therefore, we have previously proposed an efficient downward communication method for a large-scale data collection system with multiple MQTT brokers [6], called the downward transfer method by tunneling. It has two features: connection management by the server and downward message transfer by tunneling. Connection management by the server reduces the amount of memory used by the brokers connected to the server. In addition, downward transfer by tunneling allows multiple brokers to coordinate. In this paper, we implemented a prototype of the downward transfer method by tunneling and evaluated it experimentally.

## IV. Downward Transfer Method by Tunneling

### A. General Behavior of Tunneling Method

We have previously proposed an efficient communication method for large-scale data collection systems with multiple MQTT brokers, such as smart street lighting systems, called the downward transfer method by tunneling [6]. The system consists of multiple bottom brokers that accommodate IoT devices as sensors at each point, a top broker that accommodates the bottom brokers at each point, and a server (Fig. 3). The targeted system collects data from sensors and notifies control data from the server to the sensors by publish/subscribe using MQTT. The downward transfer method by tunneling has two features: connection management by the server, and message transfer by tunneling (Fig. 3). In other words, the server manages the information of

the bottom brokers to which the sensors connects, called the sensor location information. When the server notifies data to the sensor, it transmits a tunneled message to the top broker. The tunneled message includes the data notified to the sensor, the sensor location information, and the original topic. The top broker forwards the data notified to the sensor by de-tunneling the received message.

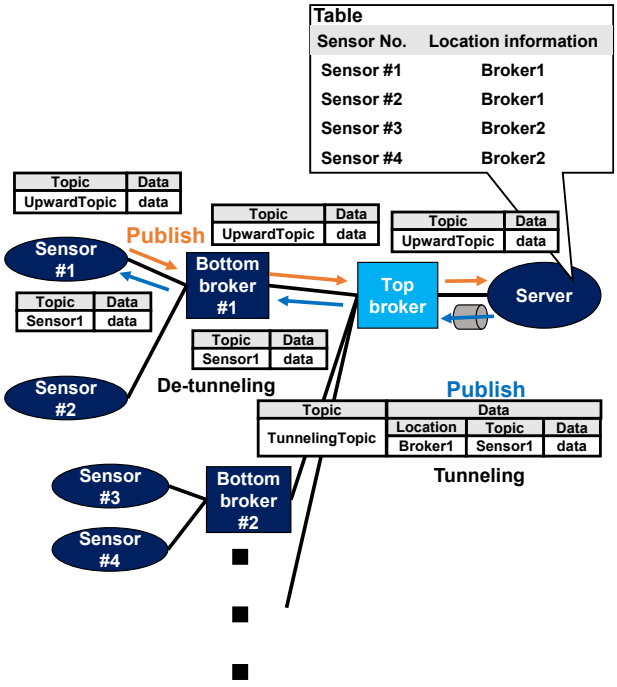


Fig. 3. Structure of the downward transfer method by tunneling.

### B. Connection Management by the Server

Communication from the server to the sensors are via the top broker. Therefore, if the top broker manages the routes or connections to each sensor, amount of memory used by the top broker becomes a major issue. Therefore, the server manages the bottom broker to which each sensor connects, to reduce the amount of memory used by the top broker for connection management. For example, as shown when sensor#1 connects to broker#1, the sensor location information for sensor#1 is "Broker1." In other words, "Broker1" is notified to the server as the sensor location information for sensor#1. Similarly, the sensor location information of sensor#4 connected to broker#2 is "Broker2." According to the notification of information from the sensor, the server maintains a table that shows the relationship between the sensor and the bottom broker to which the sensor connects.

### C. Downward Transfer

The downward transfer of notification data from the server to the sensor via tunneling is described in this sub-clause. To receive messages from the server, each sensor subscribes to a bottom broker with a topic that indicates its own information. When the server notifies data to a sensor, the sensor location information for the sensor and the notification data are tunneled in a MQTT packet. The server then publishes the tunneled data to the top broker.

As an example shown in Fig. 3, when the server notifies data to sensor#1, that is, "data," the server publishes a message with tunneling topic "TunnelingTopic," which contains the sensor location information "Broker1," the notified data and original topic. The topic "TunnelingTopic" is used by the top broker to receive messages for tunneling from the server. When the top broker receives a message with the topic "TunnelingTopic" for tunneling, it de-tunnels the original topic, and transfers the message to the bottom broker according to the extracted topic.

### D. Upward Transfer

The upward transfer system used for registration of sensor location information is described in this sub-clause. First, the server subscribes to the top broker with the topic "UpwardTopic." The topic "UpwardTopic" is used for transferring sensor location information from the sensor to the server. When the top broker connects to each bottom broker, it subscribes with the topic "UpwardTopic." When the sensor notifies to the server, it publishes the sensor location information by the topic "UpwardTopic" to the bottom broker to which it connects. The bottom broker receives this message and publishes it to the top broker to which it has subscribed. The top broker receives the published message from the bottom broker and publishes it to the server with the topic "UpwardTopic" and the payload (sensor location information) contained in the published message.

## V. Implementation of the Downward Transfer Method by Tunneling

### A. Software Architecture

To implement the downward transfer method by tunneling, we used Raspberry Pi 4 Model B 4 GB, Python as the programming language, paho-mqtt as the library, and mosquitto [13] as the broker function. The software architecture is shown in Fig. 4. The server consists of a connection management, a location management, a MQTT subscriber, a MQTT publisher, and a tunneling process. The location management and the tunneling process are specified for the proposed method. The tunneling process is used for transmitting a publish message for downward transfer. The top broker consists of a local broker, a local connection management, a MQTT subscriber, a de-tunneling process, a connection management, and a tx_function. The local broker is a broker that operates within the top broker. The de-tunneling process and the tx_function are specified for the proposed method. The tx_function is used for forwarding a publish message obtained as a result of de-tunneling a message which the top broker receives from the server. The sensor consists of the registration process, a connection management, a MQTT subscriber, and a MQTT publisher. The registration process is responsible for registration of the sensor location. The connection management, the MQTT subscriber, and the MQTT publisher are functions for conventional MQTT applications. The location management and the tunneling process in the server, the de-tunneling process and the tx_function in the top broker, and the registration process in the sensor are newly implemented functions. Therefore, it is possible to easily migrate from MQTT applications on IoT devices and servers by adding some sub-applications for the downward transfer method.

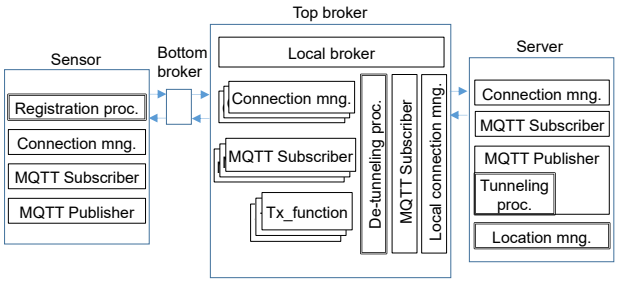


Fig. 4. Software architecture.

### B. Location Management

The location management on the server manages sensor location information including a sensor number and a broker identifier to which the sensor connects to. The location management receives sensor location information from the sensor and maintains the location management table.

### C. Tunneling Process

The tunneling process on the server places a tunneling header in the payload of the message to be transmitted and tunnels the message. A packet in the downward transfer method by tunneling consists of a TCP/IP header, a MQTT header, and a payload (Fig. 5). The MQTT header of the packet contains information such as control type, flags, and topics. The payload part of the packet is divided into the tunneling header, original topic, and data to be notified. The tunneling header contains the location information of the sensor. For example, when transmitting data to sensor#1, which is connected to broker#1, the sensor location information is "Broker1," and the original topic is "Sensor1."

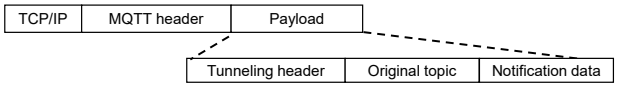


Fig. 5. Packet format.

### D. De-tunneling Process

The de-tunneling process on the top broker extracts sensor location information, the original topic, and notification data from the payload part by de-tunneling the received publish message. Then, it transmits the original topic and notification data to tx_function corresponding to the destination bottom broker based on the sensor location information through inter-process communication. tx_function transfers the notification data to the bottom broker with the original topic.

### E. Registration Process

The registration process on the sensor is a process in which the sensor publishes its own location information to the server via the bottom broker and top broker in order to register it with the server.

### F. Prototype Behavior

The behavior sequence of the prototype is shown in Fig. 6. For notifying a message from the server to the sensor, the behavior sequence of the prototype is described in this sub-clause. As pre-sequence, the top broker connects to each bottom broker by the connection management. MQTT subscriber on the top broker subscribes to each bottom broker with "UpwardTopic" and subscribes to the local broker with "TunnelingTopic."

The sensor firstly registers sensor location information with the server. Therefore, it publishes its location information to the connected bottom broker with "UpwardTopic" by the registration process. At the same time, to receive notifications from the server, the sensor subscribes to a topic that identifies notification data to the bottom broker. The top broker receives a message containing the sensor location information, from the sensor via the bottom broker and transfers it to the server. When the server receives this message, it manages the sensor location information by the location management.

Subsequently, the server tunnels the message to the top broker by means of a tunneling process in order to notify the sensor. In this message, the topic is "TunnelingTopic," and the payload contains the tunneling header, original topic, and notification data. In the top broker, messages from the server are received via the local broker. The de-tunneling process extracts the tunneling header, original topic and notification data from the payload of the published message and de-tunnels it. And it publishes using the tx_function corresponding to each bottom broker. The bottom broker then publishes to the sensor.

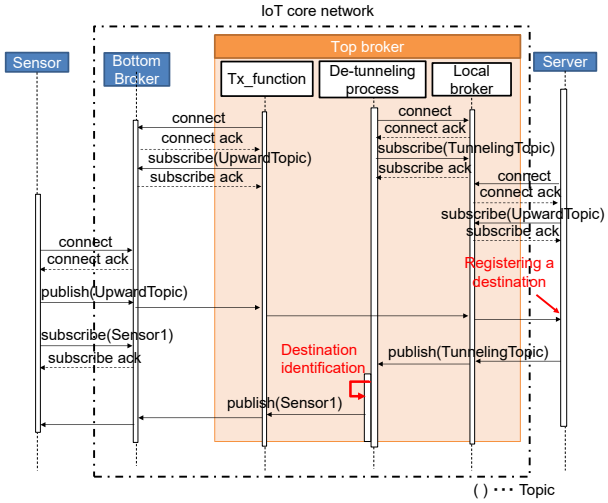


Fig. 6. Prototype behavior sequence.

## VI. Experiment and Evaluation

### A. Experimental Overview

To verify the behavior of the downward transfer method by tunneling, we monitored the traffic data in an experimental environment (Fig. 7) using a Raspberry Pi and compared them with the estimated traffic volume. The experimental environment consisted of eight Raspberry Pi model B 4GB units, which play as one server, one top broker, two bottom brokers, and four sensors. The top broker and bottom brokers are connected by LAN cables via switching hubs. There is a wireless connection between the top broker and server and between the bottom broker and sensor.

In addition, we compared the traffic of the prototype IoT core network of the proposed method with the traffic of the

IoT core network of a system with a single broker. As an IoT core network in a large-scale data collection system, the downward transfer method by tunneling is defined as the range consisting of a top broker and bottom broker, or in the case of a single broker, the range consisting of a broker and an access point.

For the experiment and evaluation, we assumed a large-scale data collection system, such as a smart street lighting system, as a use case. The number of sensors (topics) was increased to 4, 8, 16, 32, 64, and 128, for each iteration of the experiment. Multiple sensors are simulated with a single sensor device, Raspberry Pi. Because this was a large-scale data collection system, the frequency of notifications from the sensor to the server was assumed to be high; therefore, one publication per client was performed every 30 seconds. The frequency of notifications from the server to the sensor was assumed to be low; therefore, one publication was performed every 60 seconds for all clients. The keep-alive value was set to 15 seconds. The average number of messages generated under the above conditions was calculated every 30 seconds. The frequency of data notification from the server to the sensor is expected to be smaller in the actual system than the frequency used for this verification. In this verification, the frequency of data notification from the server to the sensor was increased to confirm behavior and compare traffic volumes.

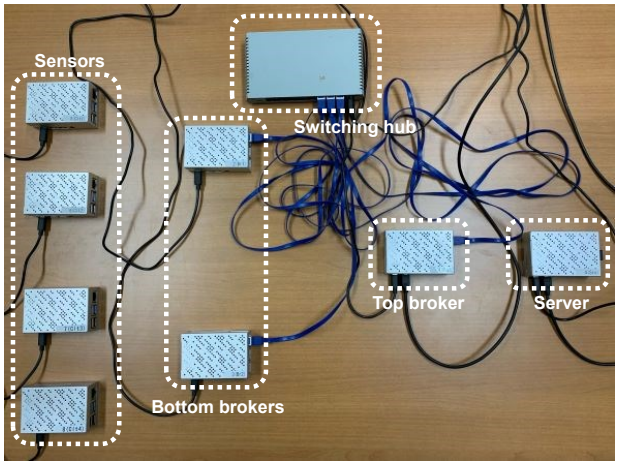


Fig. 7. Experimental environment.

## B. Verification of Prototype Behavior

A prototype of the downward transfer method by tunneling was verified using a simplified analytical model to see if it would behave as per the assumed sequence. The sequence is Fig. 6. Equation (1) shows the traffic (T1) caused by MQTT messages in the IoT core network on the IoT system by the prototype based on a simple analytical model. Here, $n$ is the number of sensors. $p$ is the number of published messages from the sensor to the server. In other words, the number of types of data to be notified to the server (equivalent to the number of topics in MQTT). $t$ is the frequency of data notification from the server to the sensor, compared to the frequency of messages from the sensor to the server. In addition, *ping* is the keep alive frequency for maintaining MQTT connections between the top broker and bottom broker. *broker* is the number of bottom brokers connected to the top broker. In the experimental environment, we monitored the number of MQTT packets in an IoT core network. Fig. 8 compares the traffic of the downward transfer method by tunneling based on a simple model with the data obtained in an experimental environment in terms of the number of messages. The set value at T1 is $p = 1$, $t = 0.5$, $ping = 2$, and $broker = 2$. It can be confirmed that there is no difference between the estimated traffic volume and the traffic volume obtained in the experimental environment. Therefore, we confirmed that the prototype functioned as expected.

$$T1 = n \times (p + (p \times t)) + ping \times broker \qquad (1)$$

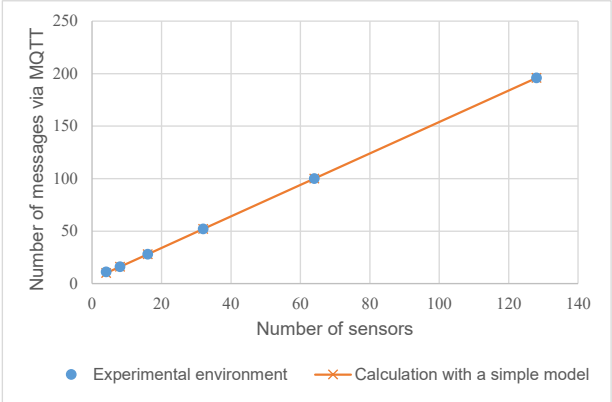


Fig. 8. Comparison of measurement results and simple model.

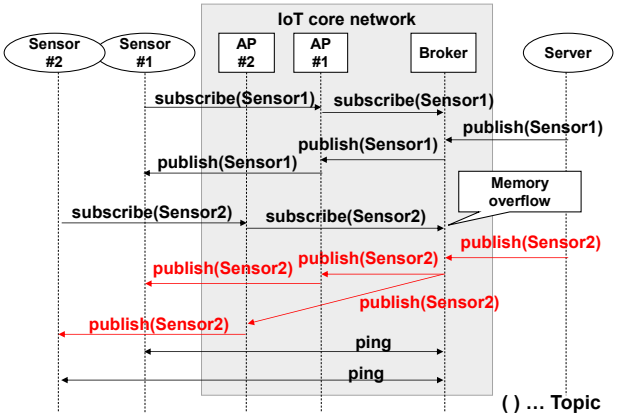


Fig. 9. Single broker evaluation sequence.

## C. Comparison with Single Broker

The traffic of the implemented proposed method prototype was compared with the traffic of an IoT system with a single broker as an existing method. For evaluating the traffic of the IoT system with a single broker, we assumed that publishing messages with unknown topics were broadcast when memory overflow occurred in the broker owing to the increase in topics subscribed by the connected sensors. Fig. 9 shows the sequence of an IoT system with a single broker.

From Fig. 9, in IoT system with a single broker, when the number of sensors exceeds the maximum number of sensors that a single broker can handle, the traffic volume increases sharply due to memory overflow. However, in an IoT system with multiple brokers by the proposed method, the traffic

volume is expected to increase linearly in proportion to the number of sensors, so the proposed method is effective. In the proposed method, the server manages the location information of the sensor. As a result, it is assumed that the memory resources consumed by the top brokers are reduced.

## VII. CONCLUSION

In this paper, we present a prototype implementation of the downward transfer method by tunneling, which is an efficient communication method that we have previously proposed for a large-scale data collection system with multiple MQTT brokers, such as smart street lighting systems. The proposed method consists of connection management by the server and upward and downward transfers by the top broker through tunneling. In other words, the server manages the bottom broker information to which the sensor is connected. When the server notifies data to the sensor, the top broker transfers a message containing the data to the sensor and the sensor information managed by the server, to the bottom broker using tunneling technology.

We implemented a prototype of our proposed method. Its implementation takes into account easy migration from conventional MQTT applications on IoT devices and server by adding some sub-applications for the downward transfer method. To verify the behavior of the downward transfer method by tunneling, we monitored the traffic data in an experimental environment using a Raspberry Pi and compared it with the estimated traffic volume. It was observed that there was no difference between the estimated traffic volume and the traffic volume obtained in the experimental environment. Therefore, we confirmed that the prototype functioned as expected. In addition, the traffic of the prototype was compared with the traffic of an IoT system with a single broker as a control. In an IoT system with a single broker, the traffic volume increases rapidly due to congestion on the single broker if memory overflow on the single broker occurs due to an excessive increase of the number of sensors. However, in the IoT system using the proposed downward transfer method by tunneling, the amount of traffic increased linearly with the number of sensors even if the number of sensors increased excessively. Therefore, we confirm that the memory resources consumed on the top broker could be reduced by management of the sensor location management on the server in the proposed method. And by prototyping the proposed method, the downward transfer method by tunneling can be used in large-scale data collection systems. In the future, we will continue to verify the proposed method in a form closer to actual deployment, including its application to real-world environments.


## ACKNOWLEDGMENT

Part of this work has been supported by the "Strategic International Standardization Promotion Project" of the Ministry of Economy, Trade, and Industry in Japan. The authors would like to heartily thank the members concerned.


## REFERENCES


[1] Jie Ding, Mahyar Nemati, Chathurika Ranaweera, and Jinho Choi, "IoT Connectivity Technologies and Applications: A Survey," IEEE Access (Volume: 8) Apr. 2020.

[2] Luís M. Borges, Fernando J. Velez, and António S. Lebres, "Survey on the Characterization and Classification of Wireless Sensor Network Applications," IEEE Communications Surveys and Tutorials, Vol. 16, No. 4, pp. 1860-1890, Apr. 2014.

[3] Koichi Ishibashi, and Katsunori Yamaoka, "A Study of Network Stability on Wireless Sensor Networks," 2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies, Sep. 2015.

[4] Tetsuya Yokotani, and Yuya Sasaki, "Comparison with HTTP and MQTT on Required Network Resources for IoT," 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC) Sep. 2016.

[5] Yuya Sasaki, Tetsuya Yokotani, and Hiroaki Mukai, "MQTT over VLAN for Reduction of Overhead on Information Discovery," 2019 International Conference on Information Networking (ICOIN), May. 2019.

[6] Fuya Aoki, Koichi Ishibashi, Tetsuya Yokotani, "Proposal of an efficient downward communication method for a large-scale data collection system using MQTT," 15th International Workshop on Informatics (IWIN2021) Sep. 2021.

[7] Koichi Ishibashi, Fuga Nakai, and Tetsuya Yokotani, "A Study of Data Collection Method by Using a Wildcard on a Large-Scale Smart Street Lighting System," The 24th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2020), Sep. 2020.

[8] SCALAGENT "Benchmark of MQTT servers ActiveMQ 5.10.0 Apollo 1.7 JoramMQ 1.1.3 (based on Joram 5.9.1) Mosquitto 1.3.5 RabbitMQ 3.4.2," Jan. 2015.

[9] Biswajeeban Mishra, "Performance Evaluation of MQTT Broker Servers," International Conference on Computational Science and Its Applications (ICCSA2018), pp. 599-609, Jul. 2018.

[10] Edoardo Longo, Alessandro E.C. Redondi, Matteo Cesana, Andres Arcia-Moret, and Pietro Manzoni, "MQTT-ST: a Spanning Tree Protocol for Distributed MQTT Brokers," 2020 IEEE International Conference on Communications (ICC), Jul. 2020.

[11] Pongnapat Jutadhamakorn, Tinnapat Pillavas, Vasaka Visoottiviseth, Ryousei Takano, Jason Haga, and Dylan Kobayashi, "A Scalable and Low-Cost MQTT Broker Clustering System," 2017 2nd International Conference on Information Technology (INCIT), Jan. 2018.

[12] Ryohei Banno, Jingyu Sun, Susumu Takeuchi, and Kazuyuki Shudo, "Interworking Layer of Distributed MQTT Brokers," IEICE Transactions on Information and Systems, Vol. E102-D, No. 12, pp. 2281-2294, Dec. 2019.

[13] Eclipse Foundation, "Eclipse Mosquitto™ An Open Source MQTT Broker," https://mosquitto.org/, May. 2022.