A Method of Distributing Clients to MQTT brokers Using Server Redirection

1st Keisuke Yoshimura
Kogakuin University
Tokyo, JAPAN
em23046@ns.kogakuin.ac.jp

2nd Ryohei Banno
Kogakuin University
Tokyo, JAPAN
banno@computer.org

***Abstract*—MQTT is a simple message protocol with the publish/subscribe model, which enables loosely coupled communication. In large-scale systems, load balancing is required because of the problem of concentrated load on brokers and subscrebers. However, existing load balancing methods have difficulty in distributing the load according to the load status of broker and the sending frequency of publisher, and there is a problem of a single point of failure.**

In this study, we propose a load balancing method using server redirection specified in MQTT v5.0 to improve the problems of existing methods. In order to verify the effectiveness of the proposed method, we conducted comparison experiments with DNS round-robin, an existing method, and measured the throughput and CPU utilization of each broker as evaluation indices.

***Index Terms*—MQTT, Publish/Subscribe, Load distribution**

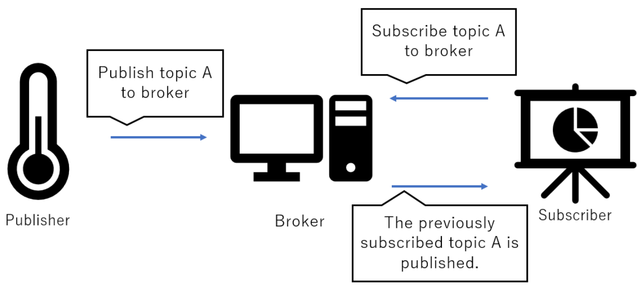

Publish topic A to broker
Subscribe topic A to broker
Publisher
Broker
Subscriber
The previously subscribed topic A is published.

Fig. 1. Overview of MQTT protocol

I. Introduction

IOT devices have been spreading rapidly in recent years, and according to the Ministry of Internal Affairs and Communications' 2022 White Paper on Information and Communications [1], the number of IoT devices is expected to grow to 39.85 billion by 2024. HTTP [2], WebSocket [3], and other protocols are examples of protocols used to collect information using IoT devices, These protocols are unsuitable for some large-scale IoT systems because they are one-to-one communication protocols. In addition, their large header can cause a bandwidth squeeze.

In recent years, the MQTT protocol [4] has been attracting attention. It is multicastable and allows many-to-many communication. MQTT is more suitable for large-scale IoT information collection than HTTP or other protocols because the small header size does not squeeze the communication bandwidth. IoT information collection with MQTT has a problem where the load concentrates on the broker and the subscriber, and various methods have been proposed to distribute the load. However, existing load balancing methods have difficulty in distributing the load according to the broker's load status and the publisher's sending frequency.

This study aims to improve the issue of the single point of failure and to realize the allocation according to the sending frequency of publishers and the load status of brokers, which are problems that have occurred in the existing load balancing methods.

This paper consists of five sections: Chapter 2 describes existing studies and related techniques, Chapter 3 describes the proposed method, and Chapter 4 presents the results from the experimental environment and discusses the results. Finally, Chapter 5 summarizes the paper and discusses future prospects and issues.

This work was supported in part by JST PRESTO Grant Number JPMJPR21P8.

II. Existing Techniques and Related Research

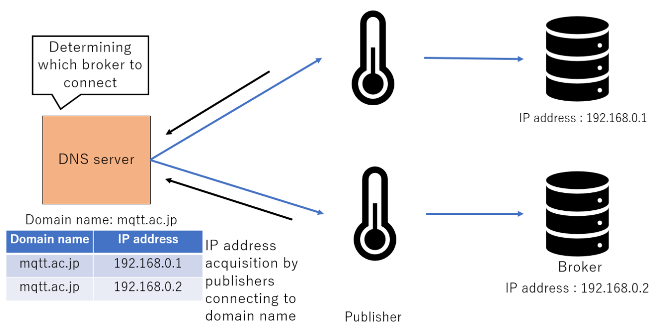
A. MQTT

MQTT consists of three roles: publisher, broker, and subscribe. MQTT is characterized by its simple and lightweight message protocol, which means that it consumes little power and does not overwhelm the communication bandwidth. It also provides a loosely coupled nature due to its Publish/Subscribe model. Figure 1 shows how MQTT works. In MQTT, the publisher sends a message with a topic to the broker and the broker stores the message. Then the broker forwards a it to the subscribe who has subscribed to its topic.

B. Load Balancing Method Using DNS Round Robin and Shared Subscription

When a publisher sends a large amount of data to a broker, there is a possibility of increased latency and packet loss. A method proposed by Banno et al. [5] successfully produced high throughput by using DNS round-robin to distribute the broker's receiving load and Shared Subscription to distribute the subscribe's receiving load.

1) DNS Round Robin: DNS round robin is also available as a product [6] and is one of the major load balancing systems. Figure 2 shows an overview of the DNS Round Robin mechanism. In DNS round robin, a DNS server keeps a table of domain names and IP addresses, and returns the IP address corresponding to the domain name in question to a client query. Clients connect to the IP address returned by the DNS server to distribute the load. Advantages include ease of


Determining which broker to connect
DNS server
Domain name: mqtt.ac.jp
IP address acquisition by publishers connecting to domain name
Publisher
IP address : 192.168.0.1
Broker
IP address : 192.168.0.2

Domain name	IP address
mqtt.ac.jp	192.168.0.1
mqtt.ac.jp	192.168.0.2

Fig. 2. Allocating publishers by DNS round robin.

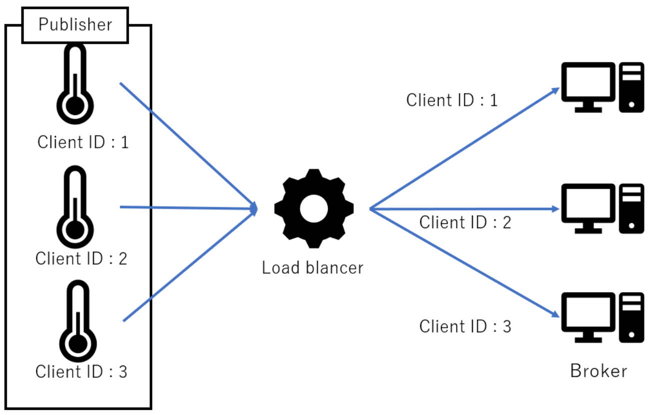


Fig. 3. Communication Between Publishers and Brokers with a Load Balancer

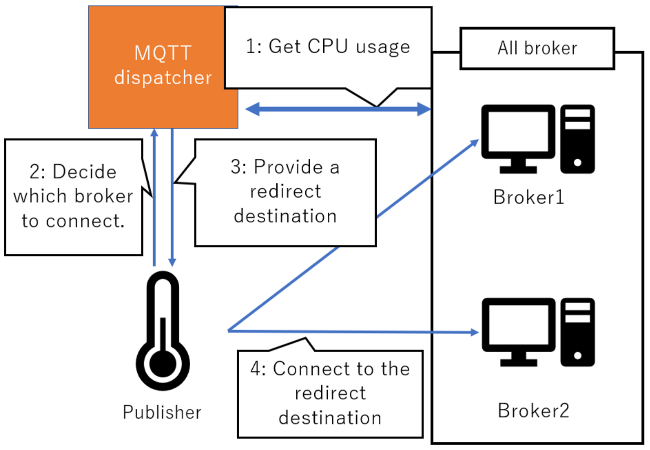

MQTT dispatcher
1: Get CPU usage
All broker
2: Decide which broker to connect.
3: Provide a redirect destination
Broker1
Publisher
4: Connect to the redirect destination
Broker2

Fig. 4. Overview of the Proposed Method

implementation and the fact that the DNS server configuration file only needs to be changed when adding new servers. The disadvantage is that DNS servers can only distribute clients according to the order in the table corresponding to domain names, so it is not possible to distribute clients according to the broker's load status. In addition, when a problem occurs in the broker, communication continues without detection. A time lag due to caching occurs when DNS settings are changed due to the addition of new servers.

C. Load Balancing Method Using Load Balancers and Broker Clustering

To improve the aforementioned problems such as DNS round robin, Jutadhamakorn's method [7] uses a load balancer between the publisher and broker, and clusters the brokers together to achieve more scalable load balancing.

1) Load Balancer: Load balancers are one of the major load balancing systems available in the market [8]. Figure 3 shows an overview of the load balancer mechanism. When a client connects to the load balancer, the load balancer stores the client ID as a hash and assigns it to the broker for each client ID. Messages sent from clients then pass through the load balancer to the broker allocated earlier for load balancing. The advantage of this method is that it provides higher performance load balancing than DNS round-robin. In addition, some load balancers have fault detection. The disadvantage is that the load balancer is the single point of failure because all communications go through the load balancer. The larger the network, the higher the required performance of the load balancer is. Besides, the latency could be larger since it increases the relay points between publishers and brokers.

III. Proposed Method

Figure 4 shows an overview of the proposed method, in which a dispatcher implementing server redirection is provided. Hereinafter, it is referred to as MQTT Dispatcher.MQTT Dispatcher obtains CPU utilization from each broker and directs the publisher to connect to the broker with the lowest load by using Server redirection. The CPU utilization data obtained from each broker is sent to MQTT Dispatcher using socket communication, and a 5-second average is calculated by MQTT Dispatcher. When a client connects, the broker with the lowest average value is set as the redirection destination for Server redirection. Server redirection is a new functionality defined in MQTT version 5.0 [9]. Figure 5 shows an overview of the server redirection mechanism. Server redirection is a mechanism that includes information on other brokers in the response message when a client connects to a broker.

IV. Experimental Environment

Table 1 shows the specifications of the machines used in this experiment. A total of eight machines were used to run the following process. Two types of publishers were implemented.Publisher 1 has 9 units.Publisher 2 one unit. Broker has two units. MQTT dispatcher has one unit. Five machines were used to implement the publishers. One machine each was used for the broker and MQTT dispatcher implementation. We omit deploying subscribers since the experiment focuses on allocating publishers. MQTTLoader v0.8.2 [10] was used as the measurement tool, and Mosquitto [11] was used to implement the broker. Since the brokers run on Ubuntu, the CPU utilization was calculated every second using the output information from /proc/stat in Linux. Table 2 shows the parameter settings for the two types of publishers. In this experiment, we evaluated the throughput between publisher and broker and the CPU utilization of each broker.

A. Experimental Methods

We compared the proposed method and a method using DNS round-robin (hereinafter referred to as the "existing

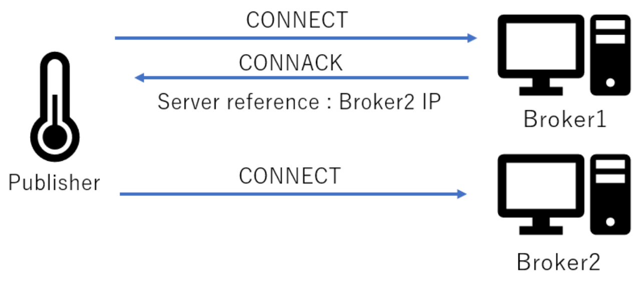


Fig. 5. An Example of Server Redirection.

TABLE I
MACHINE SPEC

	publisher, broker
CPU	Celeron N3350
Memory	4GB
OS	Ubuntu 20.04

method"). The experimental configuration of the proposed method is shown in Figure 4, and that of the existing method is shown in Figure 6. The existing method omitted the implementation of DNS servers and publishers were alternately connected to the broker. publishers were added at intervals of approximately 5 seconds, and the order of addition was randomly determined.

B. Result

The average throughput betweenthe publishers and the brokers is shown in Figure 7. Figure 8 and 9 show the results of CPU transition for each broker where Publisher2 was the second to be added, and Figure 10 and 11 show the results where Publisher2 was the fourth to be added. In terms of throughput, the proposed method shows less difference in throughput among brokers compared to the existing methods.

C. Discussion

The difference in throughput between the two brokers is about 8000msg/sec for the existing method, while the difference is about 3000msg/sec for the proposed method. These indicate that the load on each broker is more uniform than that of the existing method, and thus the proposed method is more efficient in load balancing.

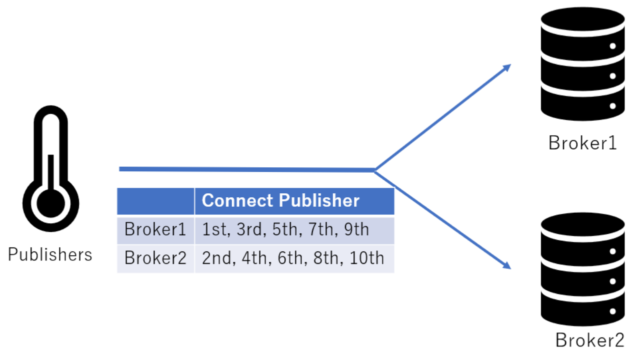
Comparing Figures 8 and 9, the proposed method has a smaller difference in CPU utilization between the two brokers than the existing method during the entire measurement time. In other words, the proposed method could distribute the load on the brokers more evenly than the existing method. The reason is that the addition of the high-load publisher was early, and the addition of the following publishers gradually smoothed the biased load.

The proposed method seems to be less effective when comparing Figures 10 and 11. Its difference between the two brokers was smaller than the existing method for the entire measurement time.

However, in Figure 10, the load on Broker2 increases rapidly at around 20 seconds and then decreases quickly at around 120 seconds. This period of about 100 seconds is considered the operating time of Publisher2. Since the load of Broker2 was down to almost zero after that period, it is

TABLE II
CONFIGURATION OF PUBLISHERS

	publisher1	publisher2
Number	9	1
Payload size	600byte	600byte
Sending frequency	2000msg/sec	10000msg/sec
Operating Time	100sec	100sec



	Connect Publisher
Broker1	1st, 3rd, 5th, 7th, 9th
Broker2	2nd, 4th, 6th, 8th, 10th

Fig. 6. The existing method in the experiment.

considered that no publishers were added to Broker2 after the addition of Publisher2. That is, the CPU monitoring system reflects the addition of the high-load publisher, which has a high frequency of sending messages. From these, we can say that load balancing according to the sending frequency of publishers and the load status of brokers, which is the objective of this study, can be achieved by the proposed method.

On the other hand, only monitoring the CPU utilization might be insufficient. Comparingthe results of the CPU utilizations of the existing method (Figures 9 and 11), they differ greatly, even though the load on the brokers is logically almost the same from 50 seconds to 100 seconds.From this, considering other metrics might make more effective load distribution.

V. CONCLUSIONS

In this study, we proposed a load balancing method using server redirection specified in MQTT v5.0 to balance the load according to the load status of brokers and the sending frequency of publishers. To verify the effectiveness of this method, we compared it with an existing method based on DNS round-robin, measured the throughput between publisher and broker and the CPU utilization of each broker, and confirmed that the proposed method is superior to the existing method.

Since the CPU utilization might not fully represent the load by publishers, it is necessary to consider load indicators other than CPU utilization. In addition, the protocol used to monitor the load status of the brokers was proprietary, so introducing an open-source monitoring system [12] could improve the practicality. Finally, since the implementation of server redirection this time was only for CONNACK packets to the publishers, it is necessary to implement it for subscribers and DISCONNECT packets to achieve more flexible load balancing.

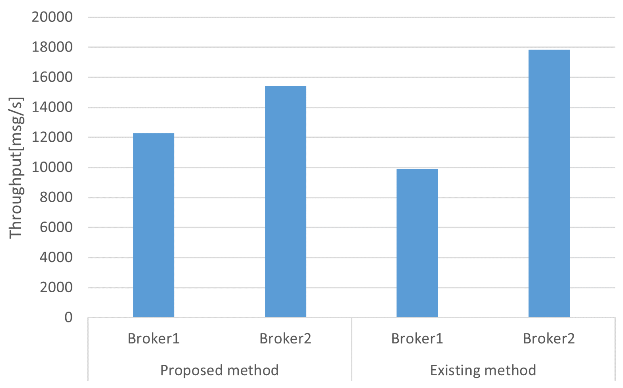


Fig. 7. Throughput between the publishers and the brokers

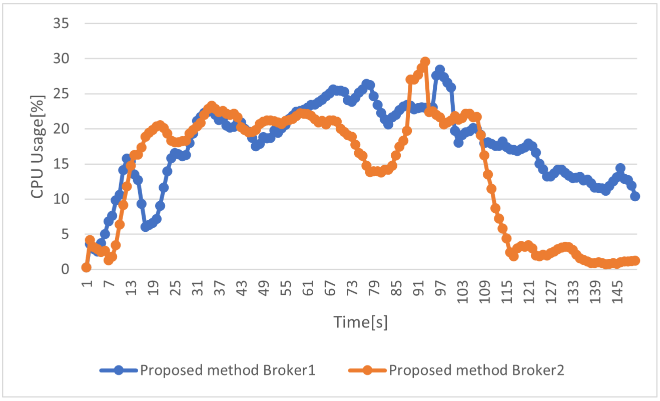


Fig. 8. CPU utilization of proposed method where Publisher2 was second.

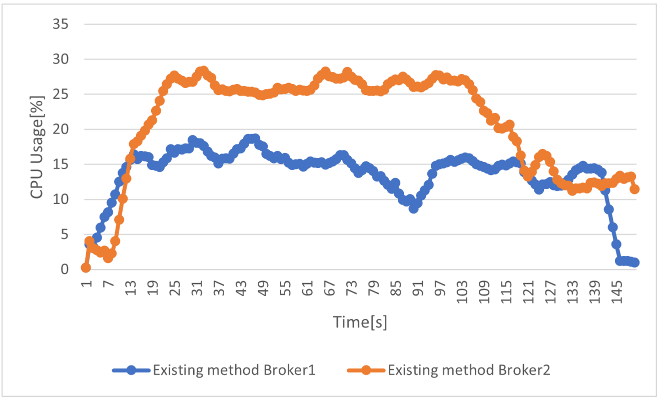


Fig. 9. CPU utilization of existing method where Publisher2 was second.

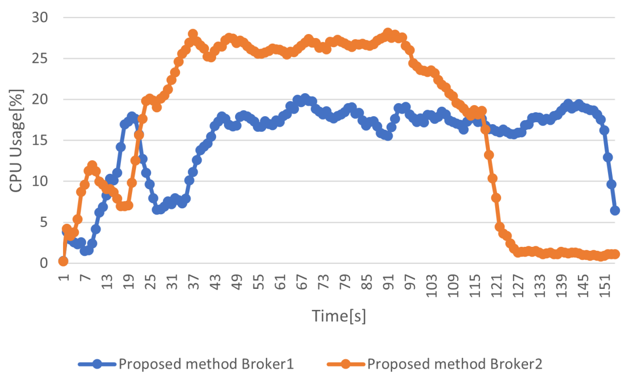


Fig. 10. CPU utilization of proposed method where Publisher2 was fourth.

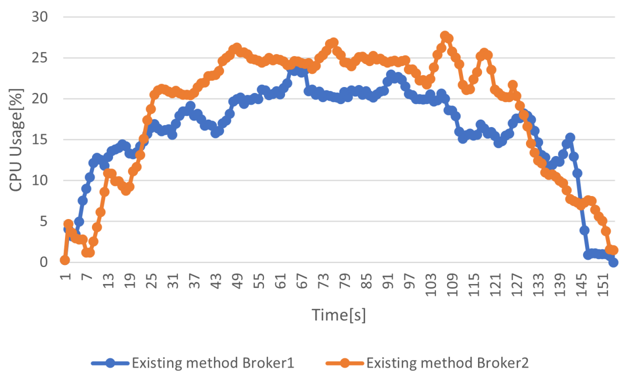


Fig. 11. CPU utilization of existing method where Publisher2 was fourth.

References

[1] Ministry of Internal Affairs and Communications, "2022 white paper on information and communication technology," https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r04/pdf/index.html (accessed May 19, 2023).

[2] HTTP Working Group, "HTTP Documentation," https://httpwg.org/specs/ (accessed May 19, 2023).

[3] "RFC 6455 - The WebSocket Protocol," https://datatracker.ietf.org/doc/html/rfc6455/ (accessed Mqy 19, 2023).

[4] "MQTT - The Standard for IoT Messaging," https://mqtt.org/ (accessed May 19, 2023).

[5] R. Banno and T. Yoshizawa, "A scalable iot data collection method by shared-subscription with distributed mqtt brokers," *EAI International Conference on Mobile Networks and Management*, pp. 218–226, 2021.

[6] "AdGuard DNS — ad-blocking DNS server," https://adguard-dns.io/en/welcome.html (accessed May 19, 2023).

[7] P. Jutadhamakorn, T. Pillavas, V. Visoottiviseth, R. Takano, and D. Kobayashi, "A scalable and low-cost mqtt broker clustering system," *2017 2nd International Conference on Information Technology(INCIT)*, 2017.

[8] Liam Crilly of F5, "NGINX Plus for the IoT:Load Blancing MQTT," https://www.nginx.com/blog/nginx-plus-iot-load-balancing-mqtt/ (accessed May 19, 2023).

[9] OASIS, "MQTT version 5.0," https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html (accessed May 19, 2023).

[10] "MQTTLoader," https://github.com/dist-sys/mqttloader/ (accessed May 19, 2023).

[11] "Eclipse Mosquitto," https://mosquitto.org/ (accessed May. 19, 2023).

[12] "Zabbix ;; The Enterprise-Class Open Sourse Network Monitoring Solution," https://www.zabbix.com/jp/ (accessed May 19, 2023).