Measuring Performance of MQTT v5.0 Brokers with MQTTLoader

Ryohei Banno*, Koki Ohsawa*, Yusuke Kitagawa*, Takumu Takada*, Toshinori Yoshizawa*
*Kogakuin University, Tokyo, JAPAN
Email: banno@computer.org

***Abstract*—MQTT is one of the best-known communication protocols for the Internet of Things (IoT). Its light-weight design and loose-coupling nature by publish/subscribe messaging model enable effective information exchange among various devices. Since MQTT brokers could be performance bottlenecks due to the massive amount of data from the increasing IoT devices, their performance holds interest from researchers and engineers. Furthermore, the performance characteristics of MQTT version 5.0, which was released in 2019 and accompanied by new functionalities e.g., shared-subscription, have not become clear. To provide a way to measure MQTT broker performance efficiently and accurately, we have developed an open-source load testing tool MQTTLoader. In this demonstration, we introduce MQTTLoader with its capabilities and features. We also show how it works by using our demonstration environment and describe some results of benchmarking MQTT brokers.**

***Index Terms*—MQTT, IoT, Publish/subscribe, Load testing**

I. Introduction

MQTT [1] is a promising communication protocol for the Internet of Things (IoT). It requires only a small size of header, that contributes to traffic reduction and power saving, as well as providing loose-coupling nature by publish/subscribe messaging model [2]. Due to the increase in demand, MQTT version 5.0 was released in 2019 and accompanied by several new functionalities such as shared-subscription [3].

The performance of MQTT brokers holds interest from engineers and researchers [4]–[8] since they could be bottlenecks if considering the massive amount of data from the increasing IoT devices. However, it has not been clarified what characteristics actual MQTT brokers have and how we can appropriately measure and analyze their performance, especially for the MQTT v5.0.

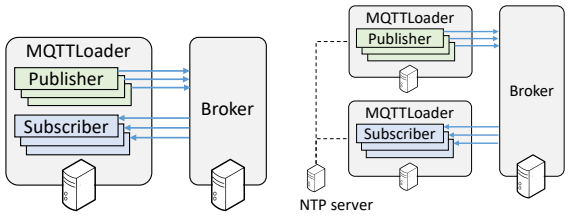
To provide a way to measure MQTT broker performance efficiently and accurately, we have developed an open-source load testing tool MQTTLoader [9]. It supports both MQTT v5.0 and v3.1.1.

In this demonstration, we introduce MQTTLoader with its capabilities and features. We also show how it works by using our experimental environment and describe some results of actual load testing.

II. MQTTLoader

MQTTLoader is a load testing tool for MQTT, implemented in Java. Binary files, source codes, and documents are publicly available on GitHub [9].

This work was supported by JSPS KAKENHI Grant No.19K20253.



(a) Running on single-host (b) Running on multi-host

Fig. 1. Overview of MQTTLoader

TABLE I
Example of parameters

Parameter	Description
-b ⟨*arg*⟩	Broker address.
-v ⟨*arg*⟩	MQTT version.
-p ⟨*arg*⟩	Number of publishers.
-s ⟨*arg*⟩	Number of subscribers.
-ss	Enable shared subscription.
-n ⟨*arg*⟩	NTP server address.

Figure 1 shows an overview of MQTTLoader. It generates multiple MQTT clients (publishers and subscribers) and applies a load on a broker based on specified parameter settings. Extracted parameters are listed in Table I. Besides ones in the list, various parameters are provided such as QoS level, Retain flag, payload size, publish interval, ramp-up/ramp-down time, etc.

We can run MQTTLoader on a single host machine or multiple host machines like Figure 1(a) and Figure 1(b) respectively. Running both publishers and subscribers on a single host may cause mutual influence, e.g., subscribers' receiving load lowers publishers' throughput. By running publishers and subscribers separately on different hosts, we can avoid such mutual influence.

MQTTLoader displays results of measurement on standard output as shown in Figure 2. The results include the following statistic information:

- Maximum throughput of publishers.
- Average throughput of publishers.
- Total number of messages sent by publishers.
- Maximum throughput of subscribers.
- Average throughput of subscribers.
- Total number of messages received by subscribers.
- Maximum latency
- Average latency

```
Measurement started: 2020-11-04 19:18:38.169 JST
Measurement ended: 2020-11-04 19:18:49.925 JST

-----Publisher-----
Maximum throughput[msg/s]: 76122
Average throughput[msg/s]: 57142.86
Number of published messages: 400000
Per second throughput[msg/s]: 41128, 64410, 76122, 71158, 75101, 71518, 563

-----Subscriber-----
Maximum throughput[msg/s]: 83989
Average throughput[msg/s]: 57142.86
Number of received messages: 400000
Per second throughput[msg/s]: 11244, 55569, 76514, 70669, 74768, 83989, 27247
Maximum latency[ms]: 895
Average latency[ms]: 512.50
```

Fig. 2. Example of measurement output

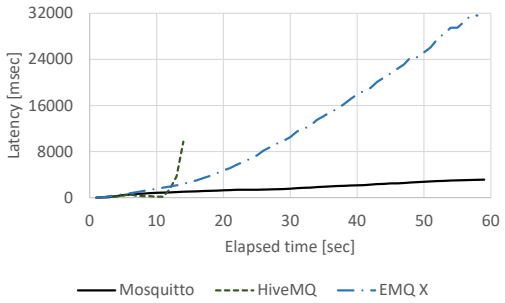


Fig. 3. Temporal change of latency

Latency is the required time from sending out by a publisher to receiving by a subscriber.

To obtain the latency, each message has a timestamp in its payload that indicates the time it was sent out. The latency is calculated by the difference between this timestamp and the time that a subscriber receives the message. In the case of running MQTTLoader on multi-hosts, MQTTLoader acquires time information from the specified NTP server and uses it for timestamps and calculation to avoid the influence of the time offset among the hosts.

MQTTLoader also exports record of sending/receiving messages. By using the record, we can analyze the performance in detail e.g., the temporal change of latency like Figure 3.

III. Measuring Performance of MQTT Brokers

Figure 4 shows our demonstration environment. We use two hosts for running MQTTLoader. MQTTLoader #1 is for publishers, and MQTTLoader #2 is for subscribers. We also use another host for running several kinds of open-source MQTT brokers; Mosquitto [10], HiveMQ [11], and EMQ X [12]. These three hosts connect each other via a 1GbE wire-speed L2 switch and can access an NTP server placed inside the campus network. We use a computer as an SSH client to control the three hosts.

By using the environment, we show how MQTTLoader works. Particularly, we focus on measurement related to the shared-subscription functionality that is one of the new features of MQTT v5.0. Shared-subscription enables to group subscribers so that they share receiving messages of a topic. That is, each message of the topic is delivered to one of the subscribers participating in the group. This could help load distribution and improve the subscribers' throughput. We

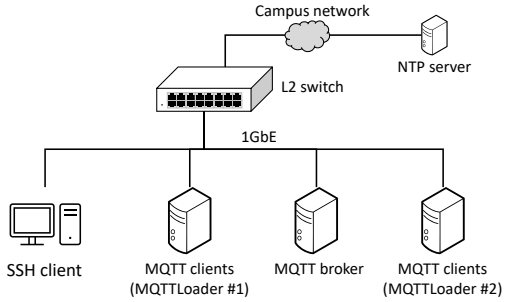


Fig. 4. Demonstration environment

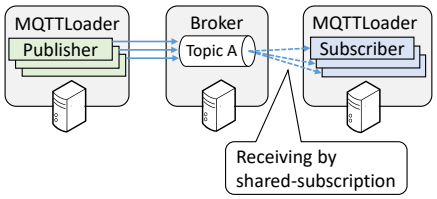


Fig. 5. Load testing with shared-subscription functionality

show how to measure the influence of shared-subscription by changing the number of subscribers and enabling the functionality as shown in Figure 5.

IV. Conclusion

In this paper, we introduced MQTTLoader, an open-source load testing tool for MQTT brokers. MQTTLoader supports the newest version of MQTT and capable of performance evaluation of MQTT functionalities such as shared-subscription. Future work includes analyzing characteristics of actual MQTT brokers in detail and adding the capability of load testing for distributed brokers such as [5]–[8].

References

[1] MQTT, https://mqtt.org/ (accessed Nov. 4, 2020).
[2] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, 2003.
[3] OASIS, *MQTT Version 5.0*. OASIS Standard, 2019.
[4] W. Pipatsakulroj, V. Visoottiviseth, and R. Takano, "mumq: A lightweight and scalable mqtt broker," in *Proc. IEEE International Symposium on Local and Metropolitan Area Networks*, 2017, pp. 1–6.
[5] A. Detti, L. Funari, and N. Blefari-Melazzi, "Sub-linear scalability of mqtt clusters in topic-based publish-subscribe applications," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1954–1968, 2020.
[6] E. Longo, A. E. Redondi, M. Cesana, A. Arcia-Moret, and P. Manzoni, "Mqtt-st: a spanning tree protocol for distributed mqtt brokers," in *Proc. IEEE International Conference on Communications*, 2020, pp. 1–6.
[7] R. Banno, J. Sun, M. Fujita, S. Takeuchi, and K. Shudo, "Dissemination of edge-heavy data on heterogeneous MQTT brokers," in *Proc. IEEE International Conference on Cloud Networking*, 2017, pp. 1–7.
[8] R. Banno, J. Sun, S. Takeuchi, and K. Shudo, "Interworking Layer of Distributed MQTT Brokers," *IEICE Transactions on Information and Systems*, vol. E102.D, no. 12, pp. 2281–2294, 2019.
[9] MQTTLoader, https://github.com/dist-sys/mqttloader (accessed Nov. 4, 2020).
[10] R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," *Journal of Open Source Software*, vol. 2, no. 13, p. 265, 2017.
[11] HiveMQ, https://www.hivemq.com/ (accessed Nov. 4, 2020).
[12] EMQ X, https://www.emqx.io/ (accessed Nov. 4, 2020).