Evaluating the Performance of MQTT Brokers

Biswajeeban Mishra

Abstract: Internet of Things (IoT) is a rapidly growing research field, which has enormous potential to enrich our lives for a smarter and better world. Significant improvements in telemetry technology make it possible to quickly connect things (i.e. different smart devices) that are present at different geographical locations. Telemetry technology helps to monitor and measure the devices from remote locations, making them even more useful and productive at a low cost of management. MQTT (MQ Telemetry Transport) is a lightweight messaging protocol that meets today's smarter communication needs. The protocol is used for machine-to-machine communication and plays a pivotal role in IoT. In cases when the network bandwidth is low or a network has high latency, and for devices having limited processing capabilities and memory, MQTT is able to distribute telemetry information using a publish/subscribe communication pattern. It enables IoT devices to send, or publish information on a topic head to a server (i.e. MQTT broker), then it sends the information out to those clients that have previously subscribed to that topic. This paper investigates the performance of several publicly available brokers and locally deployed brokers by subscription throughput i.e., in how much time a broker pushes a data packet to the client (the subscriber), or how much time a data packet takes to reach the client from the broker, and how the same brokers' performance varies when they are put under stress test? The research question was *"In standard domestic deployment use case, is there any difference in performance of different MQTT broker distributions at standard TCP/IP level?"* MQTT brokers having version v3.1.1 have been evaluated in this study. For the evaluation we use the mqtt-stresser and mqtt-bench stress test tools to evaluate the brokers both locally, and through their publicly deployed brokers.

Keywords: Internet of Things, MQTT, MQTT Brokers, Cloud Computing

Introduction

With the rise of the Internet of Things (IoT), billions of embedded smart devices and sensors are interconnected, and they exchange data using the existing Internet infrastructure. They are enormously impacting and improving our life. In today's fast growing world, many IoT application areas exist, starting from manufacturing, automobile, agriculture, energy management, environmental monitoring to smart cities and the defense sector. For example, a supplying company can trace leakage in oil and gas pipelines from a central control room, and supply can immediately be stopped to avoid accidents [1]. Trace-passing events across the borders of a nation can be detected and sent to the appropriate authorities for necessary action. All these IoT networks use several radio technologies such as RFID (radio-frequency identification), WLAN (wireless local area network), WPAN (wireless personal area network) or WMAN (wireless metropolitan area net-works) to create a Machine-to-Machine (M2M) network. No matter which radio technology is used to operate an M2M network, the end device or machine must make their data available to the Internet. There are many M2M data transfer protocols are available for IoT systems, amongst them, MQTT, CoAP, AMQP, and HTTP are the widely accepted one. Considering message size vs. message overhead, power consumption vs. resource requirement, reliability/QoS vs. interoperability, bandwidth vs. latency, security vs. provisioning, or M2M/IoT usage vs. standardization MQTT, this latest stands tall among all [2]. It is a very lightweight TCP based M2M protocol designed for lightweight publish/subscribe messaging transport. TCP port numbers 8883 and 1883 are registered with IANA for MQTT TLS and non-TLS communication respectively [3]. An MQTT client provides three Quality of service levels for delivering messages to an MQTT Broker and to any client (ranging from 0 to 2). At Qos 0 a message will not be acknowledged by the receiver or stored and delivered by the sender. This is often called "fire and forget." It is the minimal level and guarantees the best delivery effort.At QoS 1 acknowledge is assured. Data loss may occur. At least once delivery is guaranteed. At QoS 2 data delivery is assured. Exactly once delivery is guaranteed. In this paper we investigate the performance of several publicly available brokers and locally deployed brokers , and compare their properties concerning subscription throughput.

Evaluation of Publicly Available Broker Servers

In this section we introduce our evaluation of publicly available MQTT brokers [4], the considered ones are summarized in Table 1. In our test scenario, live environment event data were sent from a Raspberry PI3 Board to the cloud using the MQTT Protocol. So, on Raspberry PI Board side, an MQTT client, called "publisher" was created using a Node-Red programming language to read environment event values from on-board temperature, humidity, and pressure sensors, and publish them on a given topic-tag to an MQTT message broker server at a rate of approximately one message per second. On the receiving end, another MQTT client, called "subscriber" was created to subscribe to the publishing topic, and receive data. Eclipse Foundation recommended MQTT data capture tool MQTT Spy was used to capture, save and analyze the received data. The goal was to evaluate overall topic-specific message load and broker payload of each broker. The evaluation parameters are depicted in Table 2.

Table 1: Publicly hosted MQTT brokers

Type	Mosquitto	HiveMQ	Bevywise
Address	test.mosquitto.org	broker.mqttdashboard.com	mqttserver.com
Port	1883	1883	1883 (TCP)
Sign up Needs	No	No	Yes (name and pwd)

Table 2: Publishing condition parameters for the public experiment

QoS:	0 / 1 / 2
Payload:	14 Bytes
Keep alive time (in seconds)	60
Clean Session	True
Total number of messages published	1000

Evaluation of Locally Deployed Broker Servers

The following MQTT brokers were deployed and evaluated in a local environment with default server configurations: ActiveMQ v5.15.2, Bevywise MQTT Route, HiveMQ 3.3 Evaluation Version, Mosquitto-1.4.14, and RabbitMQ v3.7.2. Concerning the local test environment, all servers (broker and client instances) were deployed in a hardware and software setup shown in Table 3.

Table 3: Hardware and Software Configurations

PROCESSOR:	Intel Core i5-5200U CPU@2.20GHz*4
RAM:	8 GB
DISK:	SATA 3.0, 6.0 Gb/s 5400 rpm
OS:	Ubuntu 16.04.3
OS TYPE:	64-bit
KERNEL VERSION:	4.10.0-42-generic
JAVA VERSION:	Java version "9.0.1"

In the locally deployed MQTT brokers' test scenario, the goal was to evaluate overall message rate and broker payload with a specific test case scenario (i.e: QoS 0/1/2, 1 topics, 1 client). The publishing condition parameters for the locally deployed brokers remained the same as the publishing conditions for public experiment See Table 2; only the message payload was raised to 31 Bytes. A javascript program was created to simulate the sensors in a house. The script simulated and published temperature and humidity sensor values of a room on a unique topic, 1000 times (1 message/second) and thus a total number of 1000 messages were taken into account to calculate performance statistics of a given broker server.

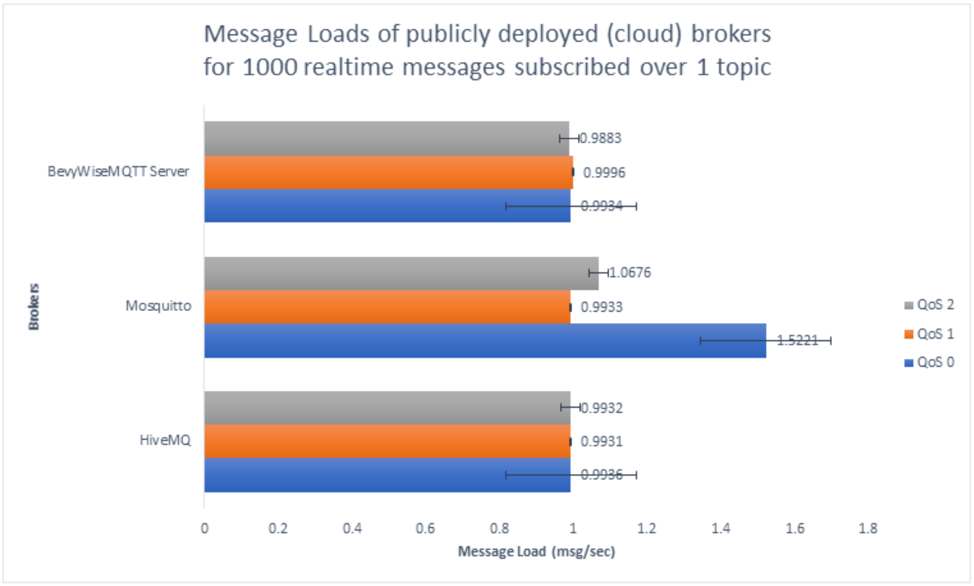


Figure 1: Message Load Rate Public Brokers

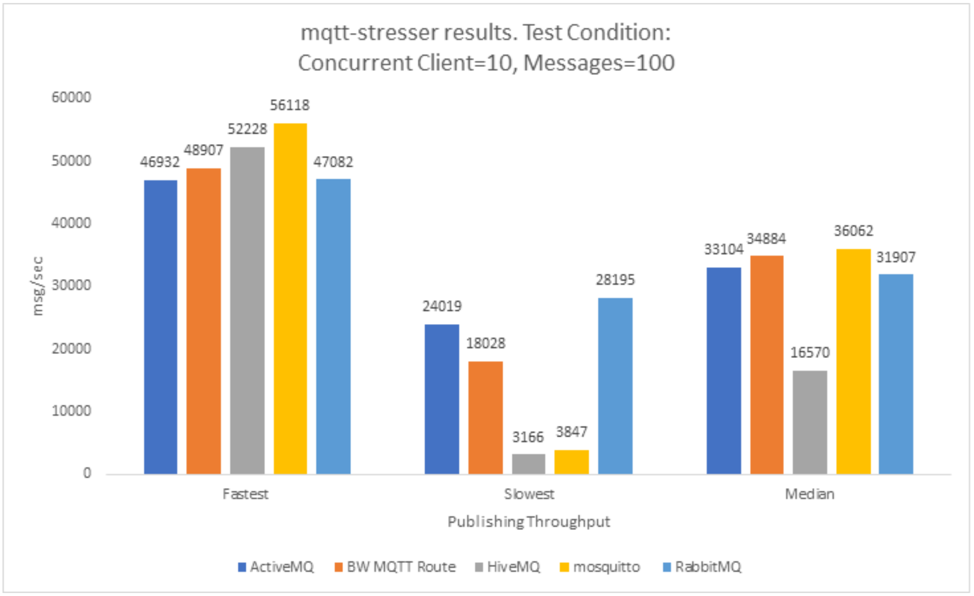
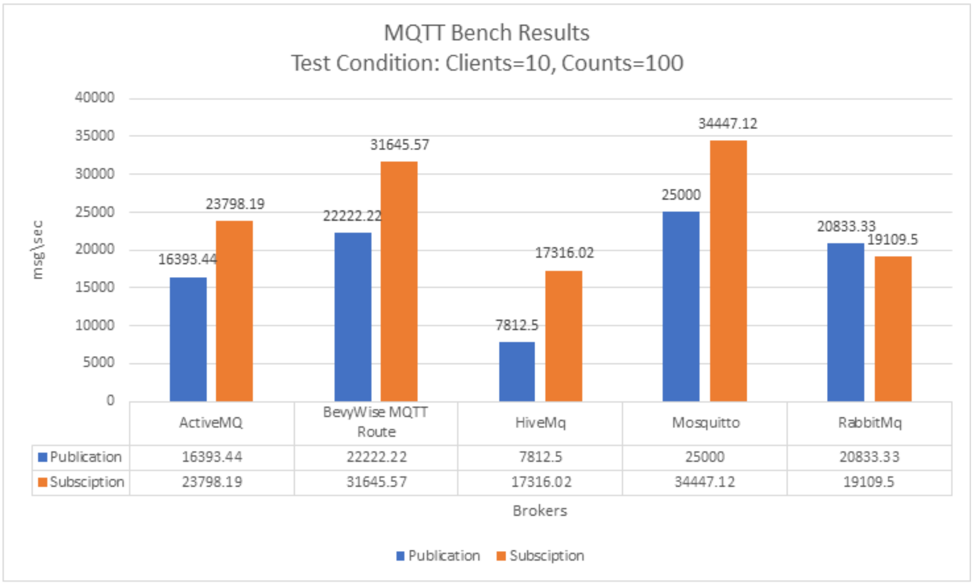


Figure 2: MQTT Stresser Results of Local Brokers

Evaluation results

The evaluation results of the public brokers can be seen in Figure 1, while we depicted the results of the locally deployed brokers in Figure 2 and Figure 3. Based on these experiments, we found that at standard Transport Layer level over TCP/IP there is insignificant difference in the performance of various MQTT brokers in standard domestic deployment use case i.e.: all MQTT broker distributions performed almost identically.



	ActiveMQ	BevyWise MQTT Route	HiveMq	Mosquitto	RabbitMq
🟦 Publication	16393.44	22222.22	7812.5	25000	20833.33
🟧 Subsciption	23798.19	31645.57	17316.02	34447.12	19109.5

Figure 3: MQTT Bench Results of Local Brokers

Conclusion

The Internet of Things paradigm represents a rapidly growing research field, which has a great potential to ease our lives and to create a better world. There are many M2M data transfer protocols are available for IoT systems, including MQTT, CoAP, AMQP, and HTTP. In this paper we investigated the performance of MQTT brokers and compared their properties by subscription throughput under a specific publishing condition. Our results showed very little difference in the performance of MQTT brokers in standard domestic deployment use case. On the other hand, we found stress testing results of MQTT brokers very inspiring, and we plan to continue this research line in the future with increased number of load conditions.

References

[1] V. Lampkin et al., Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry, IBM Redbooks publication, 268 pages, 2012.

[2] Nitin Naik, Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. IEEE International Systems Engineering Symposium, Vienna, pp. 1-7, 2017.

[3] MQTT Version 3.1.1, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Accessed in March, 2018.

[4] Public Brokers, https://github.com/mqtt/mqtt.github.io/wiki/public_brokers. Accessed in March, 2018.