Distributed Brokers in Message Queuing Telemetry Transport: A Comprehensive Review

Snowlin Preethi Janani
Research Scholar, CSE
Karunya Institute of Technology and Science, Coimbatore, India
jananifrn@gmail.com

Immanuel JohnRaja Jebadurai,
Professor, CSE
Karunya Institute of Technology and Science, Coimbatore, India
immanueljohnraja@gmail.com

Getzi Jeba Leelipushpam Paulraj,
Associate Professor, CSE
Karunya Institute of Technology and Science, Coimbatore, India
getzi@karunya.edu

Jebaveerasingh Jebadurai
Assistant Professor, CSE
Karunya Institute of Technology and Science, Coimbatore, India
jebaveerasingh@karunya.edu

2022 International Conference on Computer Communication and Informatics (ICCCI) | 978-1-6654-8035-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCCI54379.2022.9740815

***Abstract* - IoT (Internet of Things) is a system of interconnected devices used for collecting and sharing data across the Internet. IoT has enabled people to live and work smarter in daily living. IoT has driven the world to the next pace of progress. IoT plays a major role in the tremendous development of Automation. It has made our life work with ease. Lower Power Sensor Technology, Connectivity, Cloud Computing Platforms, Machine Learning and Analytics, and Artificial Intelligence are the enabling technologies that lead to the tremendous growth of this technology. Message Queuing Telemetry Transport is an important messaging protocol in IoT that offers services as brokers, subscribers, and publishers. This paper reviews various applications of IoT, Communication Protocols for IoT and it reviews in detail various MQTT architecture. Software implementation of the broker is also discussed and performance analysis of those brokers have been presented.**

Keywords – Internet of Things, Message Queuing Telemetry Transport, Broker, Subscriber, Publishers

I. INTRODUCTION

Internet of Things allows devices to connect themselves enabling easier monitoring and automation[1]. Applications viz., smart home, smart health, smart transport, and smart environment have emerged out of IoT. IoT offers connectivity and control through the existing network such as Wi-Fi, Ethernet, Bluetooth, and ZigBee. Similarly, it uses application protocols such as Message Queuing Telemetry Transport, Constrained Application protocol, and Extensible Messaging protocol for enabling data transfer between them[2]. Among these application protocols, Message Queuing Telemetry Transport Protocol has been widely used and has its application in many fields that including healthcare, industrial automation, smart home[3]–[5]. Message Queuing Telemetry Transport (MQTT) is a lightweight publish/subscribe protocol that has entities viz., Publisher, Subscriber, and Broker. The subscriber subscribes to the broker, the sensor value that is required is named TOPIC. The publisher sends all the sensor values to the broker. The broker filters and regulates the communication of sensor value to suitable subscribers. This protocol offers less energy usage and minimal bandwidth. The broker plays a very important role in MQTT communication. However, a single broker leads to a single point of failure and load issues. Various literature has suggested distributed broker architecture. This paper reviews various distributed broker architectures.

The remaining of the paper is organized as follows: Section II explains the architecture of MQTT. Section III reviews various distributed MQTT broker architecture and challenges in distributed broker have been presented. Section IV, discuss the software-based broker implementation and its performance analysis, and Section V, concludes the paper.

II. ARCHITECTURE OF MQTT Protocol

MQTT is a lightweight publish/subscribe protocol with the least code and utilizes optimal network bandwidth invented in 1999 by Andy Stanford and Arlen Nipper. It is suitable for resource-constrained applications for automating the food, health, transport, and manufacturing industry[6]. The MQTT protocol architecture is depicted in Fig.1.

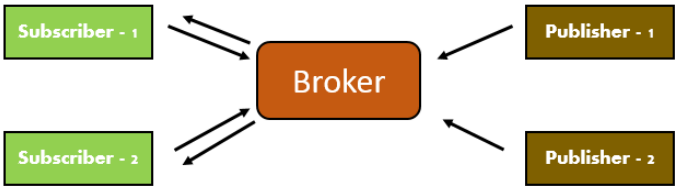

Fig. 1. MQTT Architecture

The protocol has various entities viz., Subscribers, Broker, and Publishers. The clients or subscribers communicate through a broker. The publishers are sensing devices in the network. They collect data from the environment and publish the data to the broker. The published data are grouped by the name “topic”. The subscribers subscribe to the topic of interest and the broker filters the topic and publishes it to the subscribers[7],[8]. An example

implementation of MQTT for smart home applications is shown in Fig. 2.

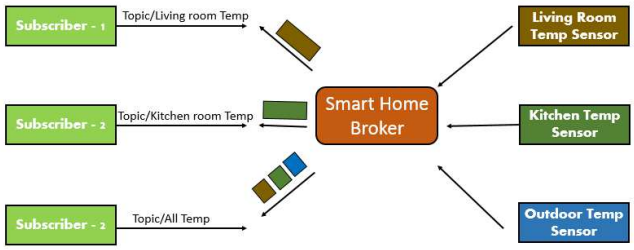

Fig. 2. Example Implementation of MQTT in Smart Home Implementation

In Fig.2. Three sensors viz., Living room-temperature sensor, Kitchen temperature sensor, and outdoor temperature sensors are connected in the smart home environment. This scenario has three subscribers in which subscriber – 1 has subscribed to receive notification regarding living room sensors only. Similarly, subscriber – 2 has subscribed to kitchen temperature sensor notification. However, subscriber – 3 has subscribed to all the sensor's values. From Fig. 2, it is understood that the MQTT broker collects the data from the sensors and publishes data to the subscribers according to their requests. Interaction between the subscribers, broker, and publishers is carried out using 14 message types. Having a single broker in the network may lead to bottlenecks and a single point of failure. The next section reviews various literature on distributed broker architectures[9].

III. DISTRIBUTED BROKER ARCHITECTURES

This section reviews various literature that used distributed brokers for various IoT applications. A distributed broker for location-based IoT applications has been proposed in [10]. Z-ordering that maps two order geographical information into a single dimension is used. Topics are published using geographical information embedded with data. The proposed system has been tested using a real-time prototype and performance analysis is carried out on the number of messages transmitted and received for unit time. A distributed MQTT-based ring topology has been proposed in [11]. Here the brokers are connected in a ring topology using VLAN architecture. Sensors are actuators are connected to the MQTT brokers in the link. Data transfer between the MQTT brokers is controlled by access control block and shared memory. Sensors communicate their data to the MQTT broker which is then written in shared memory. The shared memory has two areas viz., Read area and write area. The sensed data is written in the writing area of the attached MQTT broker. The same information is transferred to the read memory of the other broker in the next routed frame. Based on queuing analysis, it is observed that the proposed ring-based approach has less delay compared to the traditional flooding approach. A prototype for the above ring topology-based distributed MQTT broker has been proposed in [12]. A PC has been virtualized to accommodate subscribers, publishers, and a broker. There are 100 brokers in the ring accommodating one subscriber and one publisher each. There are 10 topics in the network. The system was tested to show a 99% of confidence level with a latency of less than 44ms.

A Fog-based MQTT publish-subscribe system has been proposed in [13]. The author aims at reducing the latency and excess data transmitted in the network. The edge brokers form a broadcast group in which data transmission within the group takes place through flooding. Data transmission relays through the cloud between the groups through a group leader. The proposed technique is evaluated through simulation measuring vital parameters viz., overhead for group formation, the effectiveness of broadcast group and message delivery latency, and excess data.

Byzantine fault-tolerant distributed MQTT broker architecture is presented in [14]. The components of blockchain are combined with the publish-subscribe model. Using Byzantine fault-tolerant consensus protocol, a novel technique Trinity has been proposed to avoid a single point of failure. This offers immutable storage, message ordering, and fault tolerance. The technique has been incorporated into the MQTT protocol and Ethereum. In this, multiple domains contain publish-subscribe brokers and clients. They also contain consensus nodes that maintain a distributed ledger. If the broker or the consensus node shows byzantine faults, then the domain is classified as the faulty domain. Performance analysis shows that the network overhead increases with less number of transactions and the CPU usage were proved to be 51% for HyperLedger fabric.

To provide fault tolerance, redundancy, and self-healing, a swarm-based architecture has been proposed in [15]. Swarm architecture is used in which a single swarm unmanned aerial vehicle (UAV) will behave as a master and all others will be the slave. Each UAV is equipped with software-defined planes viz., application, control, and forwarding. Applications and control flows are filtered and communicated using MQTT. Quality of Service enabled multipath routing framework is used using disjoint paths. The proposed framework is discussed with use cases that include battlefield military operations.

Internetworking layers for improving communication between MQTT brokers have been proposed in [16]. In this scenario, there are multiple brokers and internetworking layers for distributed brokers (ILDM) nodes are implemented. Standing in between the broker and client, the ILDM node serves as a proxy connecting the client to multiple brokers and nodes. An API enables the nodes to select a suitable algorithm for the deployment of the application. The throughput has increased two to four times by using cooperative brokers than using only Mosquitto broker.

Spanning tree protocol to address redundancy has been proposed in [17]. Uses in-band signaling to architect a distributed and robust broker environment. Initially, nodes exchange bridge protocol data units and elect the root bridge. The redundant ports are blocked by selecting the designated and root port. MQTT-ST has various phases viz., Connection phase, signaling phase, root selection, path computation, runtime behavior, and reaction to failures. The protocol was simulated using a varied number of subscribers and publishers. Performance metrics viz., throughput and latency have been measured. Experimental studies show that delay is minimum to the tens of milliseconds and throughput is maximum.

Challenges with Distributed Broker Environment:

Distributed broker environment avoids a single point of failure, congestion and provides available service in a MQTT-based IoT environment. However, In [18], the authors have highlighted a few challenges in implementing distributed broker in MQTT+ protocol-based multi-access edge environment. Few Challenges includes:

- Automatic broker discovery
- Broker vertical clustering
- Horizontal clustering and distribution of brokers

Several approaches for distributed brokering are also discussed. Static broker bridging in which brokers connect among themselves to publish the available data. However, static bridging does not scale well for a large-scale IoT environment. Rendezvous-based routing in which rendezvous broker is selected and publishing occurs through such brokers. A suitable optimization function must be devised for the selection of rendezvous brokers. Information-based communication model in which the topology manager provides the delivery tree, the rendezvous point concentrate on subscription and billing and forwarding function does the publishing function. Issues such as resource discovery, load balancing, and many-many paradigms have been still a research challenge in a distributed broker environment. Optimized routing strategies and implementation of MQTT-based distributed environments in edge and fog environments still need to be explored.

IV. COMPARISON OF VARIOUS BROKER ARCHITECTURES

Broker plays a very major role in an MQTT-based environment. It acts as a relay node transmitting messages between the subscribers and the publishers. It used filters to publish appropriate messages to the subscribers. There are more than 15 brokers in the market. This section presents a comparison of various brokers from the literature.

In [19], the authors have compared RabbitMQ and Mosquitto MQTT brokers. The message packages of 10Kb with 4 packages (Level-1) and 2kb with 8 packages (Level – 2) have been used for experimenting with the broker under the environment depicted in Fig. 3.

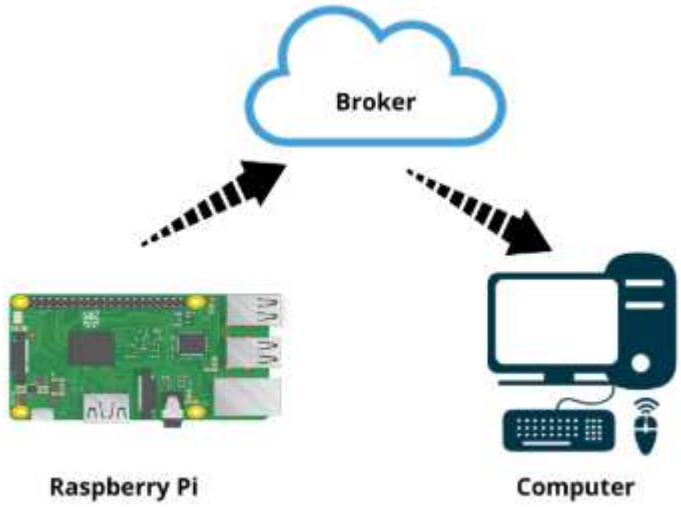

Fig.3. Testbed Environment [19]

The Raspberry pi with 1 GB memory is used as a publisher which connects to the publishers viz., computer- 1 (C1) with 2 GB RAM, i5 Intel processor, and computer-2 (C2) with 1.6GHZ and Dual-core processor through the cloud brokers. From the analysis, it is observed that the Cloud AMQP with both the brokers works well for large-scale connected devices and IoT eclipse works well for a small-scale project. The authors have suggested that the RabbitMQ can be used for smart grid and meters applications while the Mosquitto broker is suitable for home energy management and smart home applications.

Comparative analysis of three brokers viz., Erlang/Enterprise/Elastic MQTT broker (EMQX^2), HiveMQ, and VerneMQ have been performed in [20]. EMQX is an open-source broker that started in early 2013. It scales ten million connected devices. It supports EMQX-EDGE to support resource-constrained devices and suits well for applications running in a single server to the cloud. Started by a German company in the year 2012, HIveMQ provides an open-source platform for implementing run-time brokers using MQTT. Written in JAVA the platform runs famous applications that include BMW-connected cars and flight status of drones in Mattenet. To overcome the scalability issue of AMQP and XMPP, VerneMQ was developed in the year 2015 in Switzerland. Microsoft and Volkswagen run its application on this platform. These three brokers are evaluated for various metrics viz., Maximum sustainable throughput (M1), average latency (M2), supported concurrent connections (M3), New broker starts time on heavy loads (M4), Message loss count (M5), Security (M6). The testbed information is detailed in the paper.

Performance evaluation of Mosquitto, HiveMQ, and Bevywise brokers has been done on [21]. At the rate of one message per second, values from temperature, humidity, and pressure sensor are published from raspberry pi to the MQTT broker. MQTT spy is used at the receiving end to receive and analyze the data. The brokers are analyzed for 1000 messages with Quality of service levels 0, 1, 2. On comparing the brokers, Mosquitto has taken less time in sending the messages with QoS – 2, and in terms of message delivery Bevywise performed with a good message load rate. The experimentation is also performed in locally deployed broker viz., Active MQ, Bevywise Active Route, HiveMQ, Mosquitto, and RabbitMQ. From the paper, it is understood that the HiveMQ has a better message delivery rate compared to other brokers and has less time to deliver than other brokers. In QOS-2 the RabbitMQ automatically downgrades to QOS-1[22].

A comparison of six well-known brokers has been performed in [23]. A local and global testing environment has been set up to evaluate the performance of the brokers. The local environment includes three PCs acting as publisher, broker, and subscriber each. The publisher shoots the 8 hardware threads at a higher rate. The system configuration and specifications are well explained in the paper. The cloud environment used the Google Cloud Platform in which three standard virtual machines are created with 32GB of memory. Benchmarking tool MQTTblaster is used for sending messages at a higher rate. 3 publisher topic has been generated via three publisher thread with 15 subscribers subscribing to all 3 topics. The payload size is 64 bytes.

V. CONCLUSION

Message Queuing Telemetry Transport is a widely-used communication protocol for Internet of Things (IoT) applications. MQTT operates using subscribers, publishers, and a broker. The broker accepts the published data from the publisher and communicates it to the subscriber. Single brokers may lead to a single point of failure and bottleneck for the broker. Hence distributed broker has been implemented for many applications viz., smart home, health, transport, and disaster management. This paper review various MQTT papers based on distributed papers. It also throws insight into challenges faced by distributed brokers. Various software-based broker implementation is also discussed and performance analysis is presented.

VI. References

[1] G. J. L. Paulraj, S. A. J. Francis, J. D. Peter, and I. J. Jebadurai, "Resource-aware virtual machine migration in IoT cloud," *Futur. Gener. Comput. Syst.*, vol. 85, no. 2018, pp. 173–183, 2018, doi: 10.1016/j.future.2018.03.024.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.

[3] M. Mukhandi, D. Portugal, S. Pereira, and M. S. Couceiro, "A novel solution for securing robot communications based on the MQTT protocol and ROS," *Proc. 2019 IEEE/SICE Int. Symp. Syst. Integr. SII 2019*, no. March, pp. 608–613, 2019, doi: 10.1109/SII.2019.8700390.

[4] M. Pradhan, "Federation Based on MQTT for Urban Humanitarian Assistance and Disaster Recovery Operations," *IEEE Commun. Mag.*, vol. 59, no. 2, pp. 43–49, 2021, doi: 10.1109/MCOM.001.2000937.

[5] P. Jamborsalamati, E. Fernandez, M. Moghimi, M. J. Hossain, A. Heidari, and J. Lu, "MQTT-Based Resource Allocation of Smart Buildings for Grid Demand Reduction Considering Unreliable Communication Links," *IEEE Syst. J.*, vol. 13, no. 3, pp. 3304–3315, 2019, doi: 10.1109/JSYST.2018.2875537.

[6] J. Sidna, B. Amine, N. Abdallah, and H. El Alami, "Analysis and evaluation of communication protocols for iot applications," *ACM Int. Conf. Proceeding Ser.*, no. December, pp. 257–262, 2020, doi: 10.1145/3419604.3419754.

[7] J. Samer, "MQTT for IoT-based Applications in Smart Cities * Dr . Samer Jaloudi **," *Palest. J. Technol. Appl. Sci.*, vol. 2, 2019.

[8] C. Rattanapoka, S. Chanthakit, A. Chimchai, and A. Sookkeaw, "An MQTT-based IoT Cloud Platform with Flow Design by Node-RED," *RI2C 2019 - 2019 Res. Invent. Innov. Congr.*, no. December 2019, 2019, doi: 10.1109/RI2C48728.2019.8999942.

[9] A. A. Alam Kazi Masudul, "A Survey on MQTT Protocol for the Internet of Things," p. 5, [Online]. Available: http://cseku.ac.bd/faculty/~kazi/files/cn_msc/MQTT.pdf.

[10] R. Kawaguchi and M. Bandai, "A Distributed MQTT Broker System for Location-based IoT Applications."

[11] T. Yokotani, S. Ohno, H. Mukai, and K. Ishibashi, "IoT Platform with Distributed Brokerson MQTT," *Int. J. Futur. Comput. Commun.*, vol. 10, no. 1, pp. 7–12, 2021, doi: 10.18178/ijfcc.2021.10.1.572.

[12] S. Ohno, K. Terada, T. Yokotani, and K. Ishibashi, "Distributed MQTT broker architecture using ring topology and its prototype," *IEICE Commun. Express*, vol. 1, pp. 70–72, 2021, doi: 10.1587/comex.2021xbl0096.

[13] J. Hasenburg, F. Stanek, F. Tschorsch, and D. Bermbach, "Managing Latency and Excess Data Dissemination in Fog-Based Publish/Subscribe Systems," *Proc. - 2020 IEEE Int. Conf. Fog Comput. ICFC 2020*, no. April 2020, pp. 9–16, 2020, doi:

10.1109/ICFC49376.2020.00010.

[14] G. S. Ramachandran *et al.*, "Trinity: A byzantine fault-tolerant distributed publish-subscribe system with immutable blockchain-based persistence," *ICBC 2019 - IEEE Int. Conf. Blockchain Cryptocurrency*, pp. 227–235, 2019, doi: 10.1109/BLOC.2019.8751388.

[15] E. Ciklabakkal, A. Donmez, M. Erdemir, E. Suren, M. K. Yilmaz, and P. Angin, "ARTEMIS: An intrusion detection system for mqtt attacks in internet of things," *Proc. IEEE Symp. Reliab. Distrib. Syst.*, pp. 369–371, 2019, doi: 10.1109/SRDS47363.2019.00053.

[16] R. Banno, J. Sun, S. Takeuchi, and K. Shudo, "Interworking layer of distributed MQTT brokers□," *IEICE Trans. Inf. Syst.*, vol. E102D, no. 12, pp. 2281–2294, 2019, doi: 10.1587/transinf.2019PAK0001.

[17] E. Longo, A. E. C. Redondi, M. Cesana, A. Arcia-Moret, and P. Manzoni, "MQTT-ST: A Spanning Tree Protocol for Distributed MQTT Brokers," *IEEE Int. Conf. Commun.*, vol. 2020-June, 2020, doi: 10.1109/ICC40277.2020.9149046.

[18] A. E. C. Redondi, A. Arcia-Moret, and P. Manzoni, "Towards a Scaled IoT Pub/Sub Architecture for 5G Networks: The Case of Multiaccess Edge Computing," *IEEE 5th World Forum Internet Things, WF-IoT 2019 - Conf. Proc.*, pp. 436–441, 2019, doi: 10.1109/WF-IoT.2019.8767268.

[19] D. L. De Oliveira, A. F. Da Veloso, J. V. V. Sobral, R. A. L. Rabelo, J. J. P. C. Rodrigues, and P. Solic, "Performance evaluation of mqtt brokers in the internet of things for smart cities," *2019 4th Int. Conf. Smart Sustain. Technol. Split. 2019*, 2019, doi: 10.23919/SpliTech.2019.8783166.

[20] H. Koziolek, S. Grüner, and J. Rückert, "A comparison of mqtt brokers for distributed iot edge computing," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12292 LNCS, pp. 352–368, 2020, doi: 10.1007/978-3-030-58923-3_23.

[21] B. Mishra, "Performance evaluation of MQTT broker servers," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10963 LNCS, no. July 2018, pp. 599–609, 2018, doi: 10.1007/978-3-319-95171-3_47.

[22] "RabbitMQ Documentation." https://www.rabbitmq.com/documentation.html.

[23] P. Measurements, "Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements," pp. 1–20, 2021.