INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 3 - V7I3-1801)
Available online at: https://www.ijariit.com

Simulation of Message Queuing Telemetry Protocol in IoT Environment

Sharadadevi Kaganurmath
sharadadeviks@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

Siddharth Sharma
siddharths.is17@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

Mohammed Asif
mohammedasif.is17@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

Sheikh Shehzad Ahmed
sheikhshehzada.is17@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

ABSTRACT

IoT has recently emerged as the most popular Internet advancement and has become a trending technology that incorporates M2M communication. Device-to-device communications in IoT are handled via the Pushing or Polling protocols. There are numerous protocols available for IoT, including AMQP, MQTT, and XMPP, with MQTT being the mostwidely used. MQTT's main advantage is its lightweight and high bandwidth efficiency. This paper mainly simulates MQTT and describes the importance of MQTT in IoT, and the terminologies related to it. Finally, we analyzed the performance of MQTT in terms of data transmission rate.

Keywords— MQTT, Broker, Client, Publisher, Subscriber, TCP,Internet of Things (IoT).

1. INTRODUCTION TO IOT

The Internet of Things (IoT) is a networked collection of many devices that can transfer data without human interventionover the internet. The IoT devices range from household devices to high-tech industrial equipment. Devices and systemsin the Internet of Things are able to dynamically adapt to changing circumstances. IoT consists of smart devices that gather data from the surroundings and send it over the network using embedded systems such as CPU, sensors, etc. In IOT lightweight data transfer would be provided through the IETF's Constrained Application Protocol, ZeroMQ, and MQTT. MQTT is an OASIS-recognized IoT communications protocolthat is intended to be a lightweight publish/subscribe message transport that's suitable for integrating remote devices with minimum code and network resources.

The IoT functional block provides the system the various features of IoT such as identification, sensing, actuation, communication and management. The various IoT communication models that are used in IoT are:

1) *Request-Response Model:* This is a stateless architecture in which the client requests to the server, and the server responds. Each request-response pair is independent of the others in this architecture [1].
2) *Publish-Subscribe Model:* Publishers are data sources. Publishers contribute content to the broker's subjects, which aremanaged by the broker. Publishers have no idea whothe subscribers are. Subscribers subscribe to the broker-managed topic. When the broker gets data from the publisher for a topic, it distributes it to all subscribers of that topic [1].
3) *Push-Pull Model:* Producers of data send data into queues, while consumers extract data from the queues. Consumers are not required to be informed of the producers. Queues assist in the decoupling of the communication from the producers to theconsumers [1].
4) *Exclusive Pair Model:* It is a bi-directional, full-duplex communication architecture in which the client and server maintain a permanent connection. Once a connection has beenestablished, it will stay open until the client requests that it be closed. It is a stateful communication approach in which the server keeps track of all open connections [1].

I. 2. IOT APPLICATION LAYER PROTOCOLS

HTTP and HTTPS are frequently application layer protocolsused in internet applications, and this is also true in the IoT. CoAP (Constrained Application Protocol) is a lightweight HTTP protocol that operates over UDP with 6LoWPAN (IPv6 Low Power Wireless PAN). The application layer specifies how applications interact with lower-layer protocols in order todeliver data over the network. It enables process-to-process communication through the use of ports. MQTT, AMQP, and XMPP are some of the common messaging protocols used.

1) *MQTT:* Message Queue Telemetry Transport (MQTT) is a publish/subscribe messaging protocol designed for low-bandwidth applications on unreliable networks, such as

© 2021, www.IJARIIT.com All Rights Reserved

sensors and mobile devices [2]. It is well suited for a constrained environment.

2) *AMQP:* Advanced Message Queuing Protocol (AMQP) isa protocol that is used for message-oriented middleware communication. It is used for business messaging. AMQP supports both point-to-point and pub/sub model and is implemented by RabbitMQ [2].
3) *XMPP:* XMPP (Extensible Messaging and Presence Protocol) was created for real-time human-to-human communication, such as instant messaging which was modified for M2M communication in order to provide lightweight middleware and to route XML data. Smart appliances primarilyuse XMPP [2].
4) *CoAP:* Constrained Application Protocol (CoAP) is a web transfer protocol designed for constrained nodes and networksin the IoT. It is intended to allow devices to connect via constrained networks with limited bandwidth and availability [2]. Smart energy and building automations are some applications where CoAP is used.

3. INTRODUCTION TO MQTT PROTOCOL

MQTT (Message Queuing Telemetry Transport) is amessaging protocol for IoT [3]. Andy Stanford-Clark and Arlen Nipper invented it in 1999 as an effort to design a lightweight and bandwidth-efficient protocol with support for differentlevels of Quality of Service (QoS). MQTTv3.1.1 is the most often used version. It operates on TCP/IP or other network protocols that allow ordered, two-way, losslesscommunication.

A. Basic Terminologies of MQTT

1) *Publish/Subscribe pattern:* The publish/subscribe pattern (pub/sub) allows for the separation of the client that sends a message (publisher) from the client or clients who receive the messages (subscribers) [5]. Clients, on the other hand, can publish messages to topics which can be accessed by all subscribers to those topics [6] as presented in section I.
2) *Topics/Subscriptions:* Clients can subscribe to their interested topic(s) to which the messages get published. The following are the best practices: Keep the topic short and concise, using only ASCII characters, using specific topics, and avoiding extensibility [7]. Fig. 1 depicts the publish/subscribe process in MQTT.

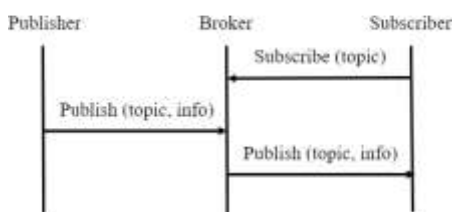

Fig. 1: Publish/Subscribe process in MQTT [8]

3) *Quality of Service Levels:* The QoS (Quality of Service) level is an agreement between a sender and receiver that defines the certainty that a particular message will be delivered [5]. It defines the reliable delivery of messages. MQTT establishes three levels of QoS:

i. QoS 0 is defined as "at most once" delivery and is the lowest reliability level. QoS 0 messages are the fastest to send but delivery of messages is not guaranteed. The receiver does not acknowledge the reception of message. It is also known asthe "fire and forget" level. It is only recommended to use this level if the sender and receiver have a stable connection.

ii. QoS 1 is called "at least once" delivery and the messagesare guaranteed to be delivered but may get delivered multiple times after replication. It is suggested when the duplicates can be managed effectively.

iii. QoS 2 is defined as "exactly once" delivery and it is the highest reliability level. QoS 2 messages are guaranteed to be delivered and guaranteed not to be duplicated.

4) *Clean and persistent sessions:* If a client disconnects froma clean session, the broker will delete the client's subscriptions, and the client must re-create the subscriptions whenever it reconnects. In case a client disconnects from an ongoing session, the broker queues the client's subscriptions, and the broker will store all non QoS0 messages that were published whilst the disconnection of client. When the client reconnects, its subscriptions would be automatically re-established.
5) *Retained messages:* MQTT permits the messages to be retained in the broker after distributing it to all its subscribed clients. After gaining another membership on the same topic, hidden messages in these topics will be forwarded to the new client [9].

B. MQTT Architecture

Fig. 2 depicts the core components of MQTT architecturewith a brief description of each component:

1) *Broker (Server):* The MQTT broker is the central server that MQTT clients can connect to. The MQTT broker regulatesthe message subjects (topics). When a client sends a message about a topic, the broker sends a copy of the message to each subscribed client. It can accept and process client requests such as subscribe, unsubscribe, and so on.

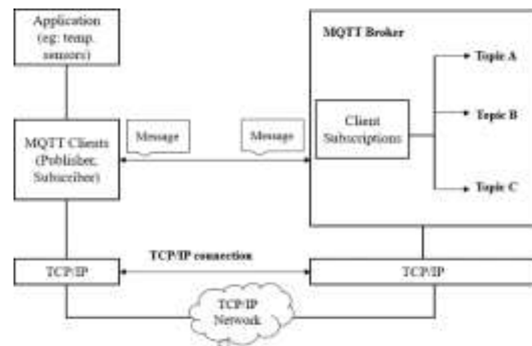

Fig. 2: MQTT Architecture [6]

2) *Client (publisher/subscriber):* To publish and receive messages, the client subscribes to topics of interest. Hence, the client can act as publisher and subscriber, the specifics of which are covered in Chapter 3. It has the ability to publish messages to interested users, subscribe to and unsubscribe from relevant topics, as well as disconnect from the Broker.

C. MQTT Message Format

MQTT message has a header of 2 bytes (fixed length) and an optional message-specific variable length header and message payload. Protocol processing is usually hampered by optional fields. However, MQTT is optimized for constrained network conditions, so it is used to keep transmissions as shortas possible. It employs network-standard ordering of bit and byte.

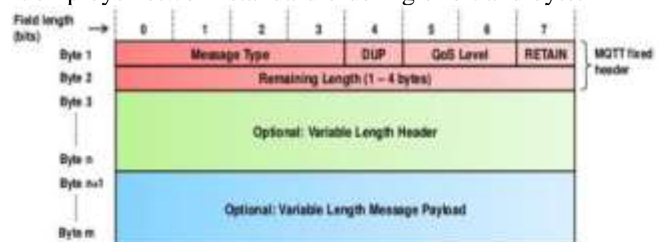

Fig. 3: Message Format [10]

The fixed header contains the following fields: Message type(15 values such as CONNECT, CONNACK, PUBLISH, PUBACK, and so on), Duplicate message flag (DUP), QoS level (values 0-

© 2021, www.IJARIIT.com All Rights Reserved

2), Retain flag (keeps the last message), and theremaining length encodes the total lengths of the variable length header and payload.

4. IMPLEMENTATION AND SIMULATION

To investigate MQTT-S implementation in theory, we created a MQTT server and two clients in LabVIEW. One of these was supposed to act as publisher client and the other one as subscriber. Following that, we tried to introduce TCP/SSL by creating new security certificates.

A. About LabVIEW

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a National Instruments system-design platform and development environment [11]. It is systems software designed for applications that require test, measurement, and control as well as quick access to hardware and data insights [12]. As a result, it's nearly ideal for simulating MQTT in a virtual environment.

B. MQTT Server

We started by creating our first block diagram and placing the server class in it from 'addons' segment of LabVIEW. We connect it to the reader of public events which is prompted by user interaction. When the user starts the server, case structure gets started. We do that by pacing a while loop inside the case structure. The reader of public events has 'server out' wire running through the while loop and the event structure in the while loop which is what keeps the server running. The event structure also shows the number of connections established i.e.; the number of clients connected through it and also what kind of connection it is; in this case it is a 32-bit connection. The server out wire further leads to the 'MQTT Stop' base class placed outside the case structure. As the name suggests, its purpose is to stop the server function when user wants. The two other base classes associated with the entire structure are:

1) Register/Unregister from event: When using dynamic registration, make sure each Event structure has a 'Register for Events' function. Each event source input refers to a specific application, VI, control, or user event. Each type of event source can generate a wide range of events.
2) Error Server: The error out cluster sends error or warning information from one VI to another. The pop-up explain error window provides additional information about the displayed error.

C. MQTT Client

In a new VI, we start creating the first client which is supposedto send the published data to the broker. We place the client library class and give its output to 'Client Read Public Events'. That, together with 'Read MQTT Public Events' is used to keeptrack of all references using their reference id number. Public Events class takes the data and sends it to SetSerializer in order to convert the data form object into stream of bytes. The next part is similar to server where a while loop is used to keep running the client as long as user wants. Case structure inenclosed within the while loop which executes the contents of event structure within it (exactly once in every while loop). Inside the event structure we have a random number generator which generates random decimal values between 0 & 1. This is the data which we wish to send via the publisher. It is all published by the publisher class within the event structure whichis connected by wires to the MQTT Client outside case structure.The waveform chart is also connected to the publisher to displaythe values of sent data. Apart from those the other two base classes same as server are present in this first client which perform their task in similar fashion.

As for the subscriber client, we made in in a new VI. Its structure is different form publisher as in:

1) It has the event id of the first client stored so the client library class subscribes to it when prompted.
2) Variant data is converted to LabVIEW data type so LabVIEW can display or process the data.

D. TCP/SSL

Netscape created Secure Socket Layer (SSL) in 1994, and ithas since evolved into a standard with input from the Internet community [13]. The 'Certification Authority' tool in LabVIEW allows us tocreate root certificates and keys. They are important if someoneneeds to create server certificate files. We send a server name indication (SNI) message, which contains the hostname (domain name) of the service to which we are seeking to connect, along with a list of cipher suites. This will be used by the server to return an SSL certificate that matches this hostname. In the VI diagram of both clients, we specify the location (path) of these certificates on the system using 'BuildPath' to the Secured TCP Client. In this case, we named it 'server1' and the remote port used for communication is 1883 (standard for MQTT).

After assembling everything in the order above we started MQTT protocol by doing the following in order: -

1) Running the server VI.
2) Running the publisher client and have it connected to thebroker.
3) Running the subscriber client, connecting it to broker and subscribing to the first client.

E. Mosquitto MQTT

Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that supports MQTT versions 5.0, 3.1.1, and 3.1 [14]. It can be used on a wide range of devices, from low-power computers to servers.

We used it to oversee the data transfer and calculate the rateat which it had been sent. It was done via the terminal and by creating and running an MQTT broker. After connecting it to the publisher client, it displayed every activity taking place – every time a new client connected, every data packet sent, etc.

II. 5. RESULTS OBTAINED

We confirmed the results by using the waveform chart in both clients. The publisher client showed the value chart that were generated by the Value Generator and ultimately sent to the broker. The number of clients connected to the broker is shown in fig. 4.

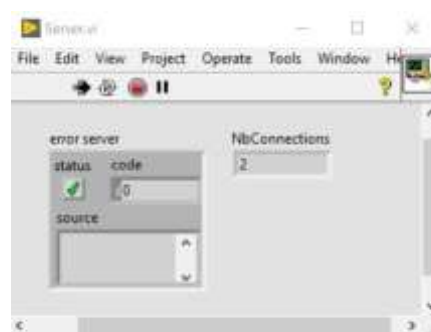

Fig. 4: MQTT server and the number of clients connected to it

We kept the frequency of the data being sent at 500 milli-seconds, i.e., two decimal values being sent every second by the publisher (Client1) to the broker (server). The fig. 5 showsthe publisher client communicating data values to the broker. The

© 2021, www.IJARIIT.com All Rights Reserved

subscriber client (Client2) printed the output by creating the same chart values sent by publisher.

Fig. 5: Publisher Client communicating data values to the broker

The fig. 6 shows subscriber client showing the 'subscribed' status and displaying the values received in waveform chart.

Fig. 6: Subscriber Client displaying waveform chart

Latency: The only delay made by subscriber was when the user didn't subscribe to the broker before connecting the publisher client. In that case the subscriber didn't receive the values which were sent before it subscribed to the broker.

Data Transfer Rate: The decimal values being transferred havea size of 44 bytes. Keeping the frequency at 500 milli-seconds,the transfer rate is (44 / 500) * 1000 = 88 bytes-per-second.

The fig. 7 shows Mosquitto broker displaying the size of each data packet sent.
\

Fig. 7: Data packets being sent in Mosquitto broker

III. 6. CONCLUSION

This paper overviews the terminologies and the working of MQTT protocol of IoT. We analyzed the performance of MQTT. In the simulation, we were able to run MQTT clients using mosquitto server at any frequency ranging from 1 to 1000 milliseconds. The packets of data sent out were of size range 44to 48 bytes – depending on the nature of data, without any changes the size of data packets remained exactly the same. One would have to make sure that the 1883 port – which is usedfor MQTT – was free every time the system is booted. Ports only work when freed from their previous use. Some of the limitations of MQTT are: centralized broker can limit scale, security issues and the overhead involved with TCP.

The future of MQTT is very promising due to the benefits it offers to its users like reliability, fast response time, simple architecture, adequate documentation, etc. The simulation of MQTT is accomplished which is the fundamental purpose of our study. In the future, we hope to add measures to change the frequency of data transfer, introduce certificate and security keys for all clients and add support for performance evaluation on different constraints. Furthermore, additional evaluations of performance metrics and appropriate qualitative interpretations for additional protocols that can be used in IoT are recommended. Potential candidates include AMQP, XMPP, and other protocols that may be of interest as IoT develops and becomes the ubiquitous focus of technology.

7. REFERENCES

[1] Dr. Chukka Santhaiah, N.M Deepika, "Internet of Things", presented tobatch R16 Information Technology, Institute of Aeronautical Engineering, Hyderabad, India [Online] Available: https://iare.ac.in/sites/default/files/IARE_IOT_LECTURE_NOTES.pdf

[2] , pg. 11-13.

[3] M. Anantha Guptha, "Internet of Things & its applications", Malla Reddy College of Engineering & Technology,pg. 13, Telangana, India

[4] "MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. OASIS Standard", Oct. 2014, [Online] Available: http://docs.oasis- open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[5] Raj Jain. (2015), Messaging Protocols for Internet of Things: for Internet of Things: MQTT [Online] Available: https://www.cse.wustl.edu/~jain/cse570-15/ftp/m_14mqt.pdf

[6] (2015, Feb) "MQTT Essentials" [Online] Available: https://www.hivemq.com/mqtt-essentials/

[7] Dipa Soni and Ashwin Makwana, "A Survey On MQTT: A Protocol Of Internet Of Things (Iot)", presented at ICTPACT – 2017, Selaiyur, Chennai, India, April 12, 2017.

[8] (2019, Aug) "MQTT Topics & Best Practices - MQTT Essentials: Part 5" [Online] Available: https://www.hivemq.com/blog/mqtt-essentials- part-5-mqtt-topics-best-practices/

[9] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, and Mohammed Aledhari, "Internet of Things: A Survey on Enabling Technologies, Protocols and Applications", in IEEE Communications Surveys & Tutorials 17(4), 2015, pp. 2347 – 2376, doi: 10.1109/COMST. 2015.2444095.

[10] Lampkin, V, Leong, W. T, Olivera, L, et al, (2012). Building SmarterPlanet Solutions with MQTT and IBM Websphere MQ Telemetry, IBM Redbooks [Online] Available: https://www.redbooks.ibm.com/redbooks/pdfs/sg248054.pdf

[11] Peter R. Egli (2017), MQTT - MQ Telemetry Transport for Message Queueing [Online] Available: https://www.slideshare.net/PeterREgli/mq-telemetry-transport

© 2021, www.IJARIIT.com All Rights Reserved

[12] (2015, May). LabVIEW [Online] Available: https://en.wikipedia.org/wiki/LabVIEW
[13] National Instruments "What Is LabVIEW?" [Online] Available: https://www.ni.com/en-in/shop/labview.html [accessed 17 May, 2021].
[14] IBM Documentation, "TLS protocol overview" [Online] Available: https://www.ibm.com/docs/en/sdk-java-technology/7.1?topic=provider
[15] -tls-protocol-overview
[16] Eclipse foundation, "Eclipse Mosquitto™ An open source MQTTbroker" [Online] Available: https://mosquitto.org/

© 2021, www.IJARIIT.com All Rights Reserved