International Journal of Research in Engineering, Science and Management
Volume-1, Issue-10, October-2018
www.ijresm.com | ISSN (Online): 2581-5782

MQTT-SN Protocol - A Review

Maria K Tom
Student, Department of Computer Science, School of Computer Sciences, Kottayam, India

***Abstract*—Traditional TCP/IP (Transmission Control Protocol/ Internet Protocol) does not cater the requirements of the IoT (Internet of Things) application. This invoked development of different protocols for IoT devices that satisfy specific purposes. Within its domain, sensor network has constrained devices and limited network bandwidth. Even though Message Queue Telemetry Transport (MQTT) is light weight they need TCP/IP stack to operate. They are heavy to be operated by small sensors that requires big payload, retry mechanism, time to live for the packets etc.**

MQTT for Sensor Networks (MQTT-SN) is adapted to incorporate devices with limited processing and storage resources. It's a publish-subscribe protocol that supports any network having bi-directional data transfer service that does not need TCP/IP. It is designed to work in similar way as MQTT but uses UDP which is a connectionless protocol. The data are transmitted based on function of contents and interests rather than network addresses. When the communications to other networks are required a gateway with MQTT-SN at the one side and the MQTT at another end will suffice.

***Index Terms*— MQTT, MQTT-SN**

I. Introduction

In the world of M2M (Machine to Machine) and IoT protocols, there are many competing protocols vying for attention. These protocols are designed to be light-weight for the low power devices to take advantage of low bandwidth constraints of the M2M world. One such protocol which is specifically designed for very low power M2M devices is MQTT-SN. The SN in the name indicates that this protocol is specifically designed for sensor networks. The fact that the headers and footers of the existing Internet based protocols is a huge overhead is considered while designing this MQTT-SN protocol.

As TCP-IP protocol is rather heavy weight for the M2M sensor networks. The solution seems to be a protocol which can ride on top of any other light-weight protocol. When the base protocol does not guarantee that robustness the requirements and implementation doesn't fit in. In case of MQTT-SN these are facilitated as and when required in the application layer. The communication with different entities in a network will take place without TCP-IP and when the communication with the cloud or certain such entity is required, then a gateway with MQTT-SN at the one side of the suite and the TCP-IP and the other end of the suite will be the requirement of the day.

MQTT-S or MQTT-SN is an extension for sensor networks of Message Queuing Telemetry Transport (MQTT). Both MQTT and MQTT-SN are client-server protocols, for which a server is needed to distribute message between the client applications. To enable the server to run on machines which do not have capacity for running a JVM, MQTT and MQTT-SN are written in C and sensors and actuators, which are very small and lacking in power, are often the sources and destinations. MQTT and MQTT-SN also using the publish-subscribe paradigm, rather than queuing mechanism.

MQTT-SN improves upon the base MQTT by adding many new features such as error status, concise message header etc. The broker acts as a conduit and controls the message to and from the client based on pub-sub mechanism and registration of topics. Even if very few clients are connected, the above said mechanism is necessary to support robustness. [3]

To reduce the size of the payloads, the data packets are numbered by numeric topic ids rather than long topic names. This particular feature which is different from the MQTT reduces the readability of the topics, but elegantly reduces the size of the packets. The negotiation for the topic ids has to happen from the client side. The protocol does not guarantee any kind of operation when a restart happens, which might have erased all the topics. These kinds of operations have to be taken care at the application level.

In place of topic name that is used by MQTT, MQTT-SN supports topic ID. The client can give a registration request to a broker with topic name and topic ID (2 octets). After registration, instead of topic name which is larger the client can use topic ID that refer to topic name. This helps to saves device memory and media bandwidth as it is quite expensive to keep and send topic name e.g.: home/livingroom/socket2/meter in memory for each publish message [2].

Topic ID can be preconfigured in MQTT-SN gateway in place of Topic name which helps to avoid even registration message before publish. Rather than TCP/IP stack, MQTT-SN can be used over a serial link, with simple link protocol. It is needed to differentiate devices on the line that results in very small overhead. Alternatively, it can be executed over UDP but it need gateway, acts as a transparent link between a sensor network of low power devices and a MQTT broker (like mosquitto). This allows to seamlessly integrate sensor devices with existing MQTT applications and libraries.

II. Literature Review

The document [1] specifies basic open and lightweight publish/subscribe protocol that is designed specifically for

machine-to-machine and mobile applications. It is optimized for communications over networks where bandwidth is at a premium or where the network connection could be intermittent.

MQTT-SN is designed in such a way that it is agnostic of the underlying networking services. Any network which provides a bi-directional data transfer service should be able to demonstrate MQTT-SN. For example, a simple datagram service which allows a source endpoint to send a data message to a specific destination endpoint should be sufficient. A broadcast data transfer service is only required if the gateway discovery procedure is employed. To reduce the broadcast traffic created by the discovery procedure, it is desirable that MQTT-SN could indicate the required broadcast radius to the underlying layer.

In the research [2], they implemented an IPv6 over BLE experimental environment, and then run MQTT and MQTT-SN on top of IPv6/BLE. Specifically, build an IPv6 over Low Power Wireless Personal Area Network, in which the 6LoWPAN Border Router (6LBR) distributes IPv6 addresses to all 6LoWPAN Nodes (6LN), as well as plays the role of a heterogeneous gateway to bridge the BLE subnet and the Internet. On top of that, IoT devices collect environmental data, and then send the data to the MQTT broker using MQTT-SN protocol.

The network architecture in experimental system in The BLE network consists of three Raspberry Pi 3 development boards: one acts as the 6LBR which plays the role of a gateway between the BLE network and the Internet, the other two are used as 6LNs which are connected with environmental sensors. It also set up a web server to collect the environmental data from the sensor nodes.

In paper [3] authors have discussed about MQTT-SN, a pub/sub protocol developed based on the following design points:

1) As close as possible to MQTT: This allows a seamless connection of the SA devices to an MQTT broker, thus enabling a smooth integration of the WSNs with the existing communication infrastructure. This also enables a very simple and lossless implementation of the gateways.
2) Optimized for tiny SA devices: The protocol is designed in such a way that it can be implemented for low- cost, battery-operated devices with limited processing and storage. Whenever complexities are required, they reside on the gateway/broker's side; the client running on the SA devices is kept as simple as possible.
3) Consideration of wireless network constraints such as high link failure rates, low bandwidth, and short message payload.

Procedures should be defined to reduce the risk of having SAs disconnected from the infrastructure owing to link failures or network congestion. Moreover, to be resistant against transmission errors, wireless networks have a much shorter packet length than wired networks.

4) Network independent: MQTT-S is designed to run on any network that provides the two following services:
   a) Point-to-point data transfer service: A datagram service that allows the transport of messages between any two points based on their network address. The two points involved may be multiple hops away from each other.
   b) One-hop broadcast data transfer service: This is in principle supported by all wireless networks; messages sent by a node can be received by all nodes within the transmission range. In contrast to MQTT, MQTT-S does not assume a connection-oriented service, and does not rely on message segmentation, nor in-order delivery of those segments.

In the paper [4] it modelled an exemplary end-to-end e-health system and analysed the service performances; the content delivery delay and content delivery probability, This derived the delivery probability and delay theoretically as a function of MQTT protocol and other system parameters, This study unveils the impact of various parameters, e.g." content publishing/request process, content/request possession time, etc." on the end-to-end delivery performances, This analysis will give system designer a flexibility to devise various admission , and service control policies . This modelling also facilitates the system designer on designing the server to meet the service assurances, Real time implementation of the MQTT SN and comparing the theoretical and simulation results against real time experimental results.

III. MQTT

It is lightweight broker-based publish/subscribe messaging protocol designed to be open, simple, lightweight and easy to implement. It is ideal for use in constrained environments with low or unreliable expensive network as it has minimal overhead. It is easy to implement on the client side that utterly applies to embedded devices with limited processor or memory resources. It is useful for use with low power sensors, but is applicable to many scenarios. [4]

The MQTT protocol is based on the principle of publishing messages and subscribing to topics, or "pub/sub". Multiple clients connect to a broker and subscribe to topics that they are interested in. Clients also connect to the broker and publish messages to topics. Many clients may subscribe to the same topics and do with the information as they please. The broker and MQTT act as a simple, common interface for everything to connect to. Messages in MQTT are published on topics. There is no need to configure a topic, publishing on it is enough. Topics are treated as a hierarchy, using a slash (/) as a separator. This allows sensible arrangement of common themes to be created, much in the same way as a file system. For example, multiple systems may all publish their hard drive utilization information on the following topic, with their own computer and hard drive name being replaced as appropriate: sensors/ COMPUTER_NAME/ utilization/ HARDDRIVE_ NAME.

Clients can receive messages by creating subscriptions. A subscription may be to an explicit topic, in which case only messages to that topic will be received, or it may include wild cards. Two wild cards are available, + or #. + can be used as a wild card for a single level of hierarchy.

IV. Why MQTT-SN?

In the world of M2M and IoT protocols, there are many competing protocols vying for attention. These protocols are designed to be light-weight for the low power devices to take advantage of low bandwidth constraints of the M2M world. One such protocol which is specifically designed for very low power M2M devices is MQTT-SN. The SN point out that it is specifically devised for sensor networks.

MQTT protocol is implemented over TCP/IP which can be used for LAN network or over Internet. MQTT-SN protocol is more fitted for sensors network like ZigBee, Z-Wave and so on. When we try to omit TCP to cut down on costs of small devices, there is currently only a small selection of non-proprietary protocols available. One of TCP-less communication protocol is MQTT-SN that is designed for constrained sensor networks. MQTT-SN allows building up a network of constrained devices with a central broker connected to many clients. Message distribution is controlled by a message bus like pub-sub mechanism and topic registration [5].

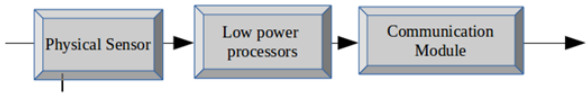

Fig. 1. Depiction of the flow from the sensor to the processor and to the network

Hence a need was felt to create a protocol which does not rely on TCP-IP and still provides value in terms of being robust enough to carry data in the M2M domain.

The guarantees provided by TCP-IP protocol is rather heavy weight for the M2M sensor networks. So, the solution seems to be a protocol which can ride on top of any other light-weight protocol. But there is a catch. Where will be the requirements fit in and how they will be implemented when the base protocol does not guarantee that robustness? The answer is that these will be implemented as and when required, in the application layer. The communication with different entities in a network will take place without TCP-IP and when the communication with the cloud or certain such entity is required, then a gateway with MQTT-SN at the one side of the suite and the TCP-IP and the other end of the suite will be the requirement of the day.

MQTT-SN is designed to be as close as possible to MQTT, but is adapted to the particular behaviour of wireless communication network such as low bandwidth, high link failures, short message length etc. Second, MQTT-SN is optimized for the implementation on low cost, battery-operated devices with limited processing and storage resources. Third, MQTT-SN demands a bridge to translate MQTT-SN data messages to MQTT messages that can be used further.

MQTT-SN, formerly known as MQTT-S is aimed at embedded devices on non-TCP/IP networks, such as Zigbee. It follows the same publish-subscribe method but works on UDP, rather than requiring a TCP connection.

MQTT−SN consists of MQTT-SN client which is a Publisher device which sends the messages to a Gateway which in turn converts the MQTT-SN message to a MQTT message and forwards it to Broker. Subsequently, Broker delivers message to the MQTT-SN clients (which are Subscribers from another Gateway). Here Gateway acts as a protocol converter from MQTT to MQTT-SN and vice versa [5].

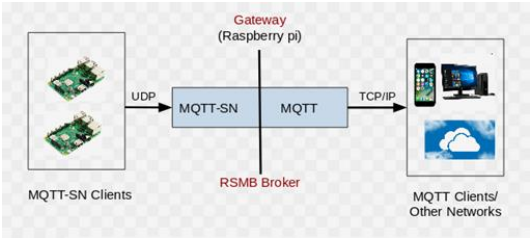

Fig. 2. Client-Server Communication over MQTT-SN using Raspberry Pi as MQTT-SN Gateway

MQTT-SN has been developed for wireless sensor networks. It can be considered as a version of MQTT adapted to the peculiarities of wireless communication environment. Unlike MQTT, in MQTT-SN system, the end nodes are connected to Gateway using MQTTSN protocol over wireless and then the Gateway is connected to Server/broker using MQTT protocol over wired network. In addition, MQTT-SN uses UDP connection mainly because UDP best suits for the simple message transmission requirement of MQTT-SN. In addition, UDP is faster, simpler, more efficient and much light weight than TCP over a wireless link, with the compromise of reliability suited for sensors applications.

MQTT-SN supports topic ID instead of topic name that saves media bandwidth and device memory. Topic name to topic ID can be preconfigured in MQTT-SN gateway, so that even registration message can be skipped before publish. MQTT-SN does not require TCP/IP stack. Alternatively, it can be used over UDP, which is less hungry than TCP.

But this needs any sort of gateway, which is nothing else than a TCP or UDP stack moved to a different device. Also, MQTT-SN is not well supported.

MQTT-SN is distinguished by given differences:

1) MQTT-SN accepts topic ID to use instead of topic name in MQTT. Firstly, client sends a registration with topic name and topic ID (2 octets) to the broker. After registration, these clients can referrer these topic name by topic ID. It saves bandwidth and device memory. The usage of topic names like home/room1/section1/meter are expensive to keep and send in each publish messages [6].
2) The pre-configuration allows to set topic ID instead of Topic name that help to avoid even the registration message before publishing.

3) If the clients are not pre-configuring the server/gateway's address, discovery procedure support clients to find the network address of operating server/gateway. More than one gateway will ru n at the same time in single wireless system with sharing of load or stand-by mode.
4) The broker to client connection is established through a gateway device, which is located inside the same network.

V. Conclusion

To make IoT applications feasible and pervasive, one of the most important considerations is the low power consumption of IoT devices so they can last a long time without replacing batteries. MQTT is a protocol which is designed to be light weight, but needs TCP/IP stack to operate. But if the sensor nodes cannot hold the TCP/IP stack then MQTT-SN which does not depend on TCP/IP is optimum. .MQTT-SN is exclusively designed for sensor networks. They are using UDP which is connectionless that reduced connection overhead compared to TCP/IP. Similarly, auto discovery of brokers and use of topic ID's, reduces the size of every packet that is transferred using MQTT-SN. The packet size reduction highly reduces the amount of power required to create and communicate data. The use of Gateway in addition to the broker will enable the sensor network to communicate with other MQTT clients and even to other networks.

References

[1] A. Stanford-Clark and H. L. Truong," MQTT for sensor networks (MQTTs)", http://www.mqtt.org/MQTT-SN_spec_v1.2.pdf, Oct. 2013.
[2] Kai-Hung Liao; Chi-Yi Lin: "Implementation of IoT Applications based on MQTT and MQTT-SN in IPv6 over BLE", International Journal of Design, Analysis and Tools for Integrated Circuits and Systems, Vol. 6, No. 1, October 2017
[3] U. Hunkeler; H.-L. Truong; and A. Stanford-Clark; "MQTT-S: A publish/subscribe protocol for wireless sensor networks", In Workshop on Information Assurance for Middleware Communications (IAMCOM '08), Bangalore, India, January 2008.
[4] Kannan Govindan; and Amar Prakash Azad; "End-to-end Service Assurance in loT MQTT-SN",12th Annual IEEE Consumer Communications and Networking Conference (CCNC)2015
[5] Deepsubhra Guha Roy; Bipasha Mahato; Debashis De; Rajkumar Buyya;" Application-aware end-to-end delay and message loss estimation in Internet of Things (IoT) — MQTT-SN protocols", Future Generation Computer Systems 89 (2018) 300–316
[6] https://www.hivemq.com/
[7] https://jpinjpblog.wordpress.com
[8] https://www.engineersgarage.com/