IEEE P21451-1-7: Providing More Efficient Network Services over MQTT-SN

1st Edelberto Franco Silva
Computer Science Department
Federal University of Juiz de Fora
Juiz de Fora, Brazil
edelberto@ice.ufjf.br

2nd Bruno José Dembogurski
Computer Science Department
Federal Rural University of Rio de Janeiro
Rio de Janeiro, Brazil
brunodembogurski@ufrrj.br

3rd Alex Borges Vieira
Computer Science Department
Federal University of Juiz de Fora
Juiz de Fora, Brazil
alex.borges@ice.ufjf.br

4th Francisco Henrique Cerdeira Ferreira
Computer Science Department
Federal University of State of Rio de Janeiro
Rio de Janeiro, Brazil
francisco.ferreira@uniriotec.br

***Abstract*—In the last 20 years, the IEEE 1451 standards family has been presented throughout the changes in the world of smart sensors. One of the most important evolution in this field was the Internet of Things, mainly because of the popularity of light-weight and simple methods to implement communication proocols among human, machines, and sensors. In particular, one of its protocols is very important to enable this interconnection: the MQTT. It has become synonymous with large cloud service providers for the Internet of Things, such as Amazon AWS, IBM Watson and Microsoft Azure. Despite MQTT protocol being traditionally used in controlled networks within server centers, nowadays this protocol is largely used on IoT and It is part of IEEE 1451 family, having the number IEEE 1451.1.6. Following the motivation to adopt IoT' standards protocols in the IEEE 1451 standard family, this paper proposes the use of a specific and potential variant of MQTT, the MQTT for Sensors Networks, or MQTT-SN. It introduces a more efficient protocol for the Common Architectures and Network services found in the IEEE 1451 Family of Standard, and shows how services and applications based on MQTT can be integrated.**

***Index Terms*—MQTT-SN, MQTT, NCAP, TIM, IEEE 1451**

I. Introduction

The IEEE 1451 standards family have been the main motivation to increase the facility of acquisition and dissemination of sensors and actuators data. Since the '90s, when its architecture was introduced, proposals of evolution are being presented, always considering the concepts of interoperability and scalability. To achieve this, in the last decade, continuous innovations in hardware, software, and connection have to lead to the expansion of the Internet of Things (IoT), with the number of connected devices growing by the day [1]. Even today, the sensors network area, namely Wireless Sensors Network (WSN), has been proposing methods which mainly focus on local area networks, worrying about how routing protocols or medium access could be managed. Although, having the IEEE 1451 standard family as our base and motivation, it is possible to think how improvements can be proposed to increase the support of sensors/actuators networks to the new IoT. Thus, it is necessary to research how the new IoT protocols can be integrated/adopted in the context of IEEE 1451' architecture.

Many protocols were created to communicate between devices to devices and devices to applications. These protocols include, but are not limited to HTTP [2], CoAP [3], XMPP, WebService, DDS, AMQP [1], MQTT [4] and MQTT-SN [5]. Each one of these has their own unique features and limits. But, the main characteristic that these protocols must have is the respect to the limitations of the network and resources.

In this paper, it will be introduced the current protocols that enable the communication between sensors/actuators and applications. To complement this, a proposal for a new box in the IEEE 1451 standard will be presented, making it possible the adoption of this protocol in the current scope. Thus, this paper is organized as follow: Section II which present the main protocols responsible to make possible the communication between sensors/actuators and applications and its comparison; Section III shows how can be adopt the communication protocol with better performance and characteristic identified in the II; Section IV shows qualitative results for some protocols, strengthening our proposal; Finally, Section V concludes the paper.

II. Communication Protocols

In this section will be introduced the main protocols responsible to enable the communication between sensors/actuators and applications, they are: HTTP, XMPP, MQTT and MQTT-SN. In this way, only the protocols in the most high-level layer will be introducing, leaving out the protocols of lower-layers (as the network or medium access layers – *e.g.* wireless IEEE 802.11, IEEE 802.15.4 etc).

A. HTTP

Considering today's state of the Web, HyperText Transport Protocol (HTTP) is the most used and compatible protocol. According to [6], its most widely accepted version is HTTP/1.1, where communication between servers and clients

978-1-5386-7713-1/19/$31.00 ©2019 IEEE

occurs via request/response messages. This happens according to the following sequence: first, the client sends an HTTP request message and the server responds if the message is accepted, with a message containing the resource that was requested. In IEEE 1451 family of standards, HTTP is defined by P21451-1-2.

Moving towards the IoT field, there is an effort to combine HTTP with the Representational State Transfer (REST) [7]. This is due to the fact of the success of RESTful Web Services, meaning that devices could potentially offer their state information more easily, considering that there is a standard to perform CRUD operations. Combining REST and HTTP creates a mapping from CRUD operations to HTTP POST, GET, PUT and DELETE methods respectively, this enables a more accessible way of building REST models for IoT devices [8].

Another important topic is the data representation. In the previously presented protocol, considering an HTTP/REST model, there is no enforcement, so any type can be used. Thus, application developers can be flexible enough to fulfill their specific needs. However, the most common representations used are JSON and XML [6], where IoT standardizes using JSON over HTTP.

B. XMPP

The Extensible Messaging and Presence Protocol (XMPP) is an open standard messaging protocol formalized by IETF [9] which consists of a set of open technologies for instant message, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data [10]. In IEEE 1451 family of standards, XMPP is defined by P21451-1-4.

Considering the previously mentioned XMPP characteristics, there are downfalls that should be addressed while working with IoT and fog technologies. The use of XML increases the size of messages making it difficult to work in the networks with bandwidth problems. The protocol depends on a persistent TCP connection and lacks efficient binary encoding [6], thus rendering its use on sensor networks, which is intrinsically unreliable and low-powered, impractical. Despite that, lately, there has been a lot of effort to create a lightweight implementation of XMPP for resource-constrained systems [11], [12].

C. MQTT

MQTT is a connectivity protocol specifically designed for machine-to-machine (M2M) and the Internet of Things [13]. Thus, it was designed to be simple and lightweight considering constrained devices, low-bandwidth, high-latency or unreliable networks. Also, it employs the publish/subscribe pattern decoupling the sending and receiving parties. This enables scalability due to the parallel nature of the publisher and subscriber components.

However, the MQTT depends on a transport layer that delivers an orderly sequence without loss of packets, such as TCP, for its operation. Sensors and actuators are generally devices with low processing capacity, storage and are often powered by batteries. Such devices are not prepared to work with TCP-IP since it is a heavy protocol and with several resources for sending packets, which are generally not necessary for a network of sensors. In IEEE 1451 family of standards, MQTT was proposed under the P21451-1-6.

D. MQTT-SN

The design of MQTT for Sensor Networks (MQTT-SN) is aimed to be as close as possible to the MQTT one [14]. Nevertheless, as the name suggests, it has modifications aiming the specificities of a sensor network, such as: low bandwidth, restraints to the message length, unreliability, link failure, etc. Moreover, it is optimized towards low-cost, battery-operated devices which possess limited processing and storage capabilities, operating on any transport layer, such as ZigBee, Bluetooth, UDP, serial wired communication, among others. For example, Wireless Sensors Networks (WSN) based on the IEEE 802.15.4 standard provide a maximum bandwidth of 250 kbit/s in the 2.4 GHz band. Moreover, to be resistant to transmission errors, their packets have a very short length. In the case of IEEE 802.15.4, the packet length at the physical layer is limited to 128 bytes. Half of these 128 bytes could be taken away by the overhead information required by supporting functions such as MAC layer, networking, security, etc.

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 between any node and a particular one (a gateway) should be able to support 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. In addition, MQTT-SN supports topic ID rather than a topic name, reducing, even more, its packet's header.

The architecture of MQTT-SN is shown in Figure 1. There are three main components in its architecture: (1) MQTT-SN clients, (2) MQTT-SN gateways (GW), and (3) MQTT-SN forwarders. MQTT-SN clients connect themselves to an MQTT server via an MQTT-SN GW using the MQTT-SN protocol. An MQTT-SN GW may or may not be integrated with an MQTT server. In case of a stand-alone GW, the MQTT protocol is used between the MQTT server and the MQTT-SN GW. Its main function is the translation between MQTT and MQTT-SN.

As seen in Figure 1, MQTT-SN can be naturally integrated with MQTT. This integration makes real the communication between the WSN and its constrained nodes, *e.g.* sensors/actuators, to the Internet and cloud services.

MQTT-SN clients can also access a GW via a forwarder in case the GW is not directly attached to their network. Its role is only to encapsulates the MQTT-SN frames received on the wireless side and forwards them unchanged to the GW.

Depending on how a GW performs the protocol translation between MQTT and MQTT-SN, it is possible to differentiate

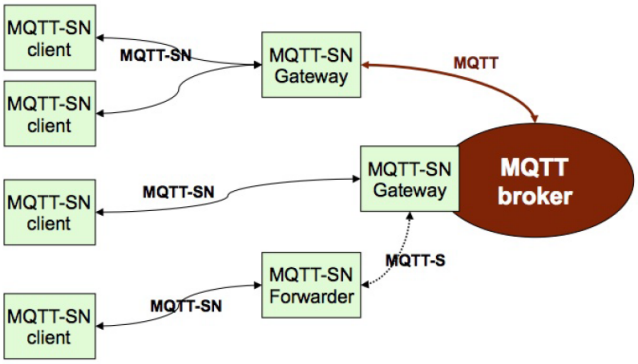

Fig. 1. MQTT-SN Architecture [5].

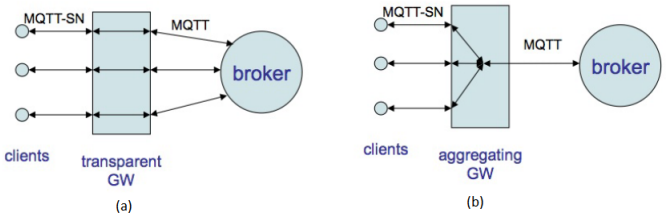

Fig. 2. MQTT-SN's gateway types. Where (a) is the transparent GW and (b) the aggregating GW [5].

between two types of GWs, namely transparent and aggregating GWs, see Figure 2(a) and (b). The transparent GW is responsible to maintain an MQTT connection to the server for each MQTT-SN client, and the aggregating GW has only one connection between the MQTT-SN' clients and the MQTT server, reducing the cost of the message exchanges.

E. Comparison

In this section will show the comparison between and all protocols and will be given a greater focus to MQTT and MQTT-SN. As seen in Table I all protocols studied are part of IEEE 1451' standards family, and the MQTT-SN is introduced as P21451-1-7.

TABLE I
COMPARISON BETWEEN ALL PROTOCOLS AND MQTT-SN.

	Broker	Transport Protocols	QoS	RESTful
HTTP P21451-1-2	–	TCP	–	X
XMPP P21451-1-4	X	TCP	–	X
MQTT P21451-1-6	X	TCP	3	X
MQTT-SN P21451-1-7	X	UDP	3	X

As mentioned, MQTT-SN is optimized for low-cost, battery-operated devices with limited processing and storage resources. Compared to MQTT, MQTT-SN is characterized by the following main differences:

- To cope with the short message length and the limited transmission bandwidth in wireless networks, the topic name in the PUBLISH messages is replaced by a short, two-byte long "topic id".

TABLE II
COMPARISON AMONG MQTT AND MQTT-SN.

Feature	MQTT	MQTT-SN
Communication	TCP	UDP
Network	Ethernet, Wifi, 3G	Zigbee, Bluetooth, RF, etc
Minimum Message size	2 bytes	1 byte
Maximum Message size	<= 24 MB	< 128 bytes
Battery	No	Yes
Sleep mode client	No	Yes
Connectionless mode	No	Yes

- "Predefined" topic IDs and "short" topic names are introduced, for which no registration is required.
- A discovery procedure helps clients without a pre-configured server/gateways address to discover the actual network address of an operating server/gateway. Multiple gateways may be present at the same time within a single wireless network and can co-operate in a load-sharing or standby mode.
- An offline keep-alive procedure is defined for the support of sleeping clients.

To briefly highlight the benefits of MQTT-SN over the MQTT, Table II is presented.

III. Architecture Update

A. Background

The IEEE 1451 family of standards outlines a way to create and support a Smart Transducer Network. To create this network the family of standards defines 3 entities, a client, a Network Capable Application Processor (NCAP), and a Transducer Interface Module (TIM). It also defines how each of these entities interacts. The architecture and connection of these entities can be seen in Figure 3.

B. Client

In the highest level of the architecture we have the Client (*e.g.* end user). The Client through its application can only communicate to the NCAP(s), sending a message about specific functions wanted to be performed. This is one of the main boxes of our interest because it is responsible to make possible the communication between the Client and NCAP under XMPP, HTTP or MQTT.

C. NCAP

Another one entity is the NCAP – Network Capable Application Processor. It is our "broker", managing the messages coming from the TIMs – Transducer Interface Module – and Clients. The structure of the message depends on the standard protocol chosen, *i.e.* XMPP, MQTT etc. Today, the NCAP is most commonly wirelessly connected through Wifi, ZigBee etc.

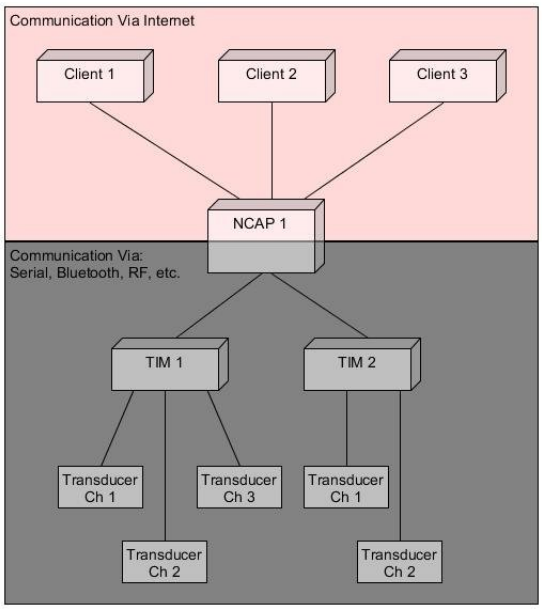

Fig. 3. Basic Architecture of a P21451 Network with multiple Clients talking to a single NCAP [15].

D. TIM

The last entity, TIM, is what handles all of the transducers in the system. The TIM can be a wired or wireless IEEE 1451.X module consisting of up to 255 transducers, signal conversion, and processing electronics.

Briefly, the NCAP is a network node composed of hardware and software that provides the gateway functions between TIM and the user network, and the TIM is a module that contains signal conditioning, analog-to-digital or/and digital-to-analog conversion, frequency, and digital converter and an interface to communicate with NCAP.

E. Proposal

MQTT provides two entities, the Client and the Broker, and the MQTT-SN 3 or 4. But, for MQTT-SN, the "Forwarder" can be nodes with more resources, and "Gateways" can be incorporated to the NCAP. As presented in [15], by conforming MQTT to the family as so it is forcing the 1451 Client and 1451 NCAP to both be MQTT clients and the TIM would remain the same and only be connected to the NCAP. Thus, the MQTT-SN need to follow the same strategy, doing transparent the connection between client and TIM.

An issue needed to think is the same as [15], where the 1451 Client and TIM needs to know previously a topic name/number. For the 1451 Client which is on the MQTT side, with more resources, and the access to the topic is by using his/her topic name. The TIM needs only to know the number ("short" topic ID) to publish/subscribe. Both need to know *a priori* the topic identification, but only the TIM needs to configure it before being sent to the real environment.

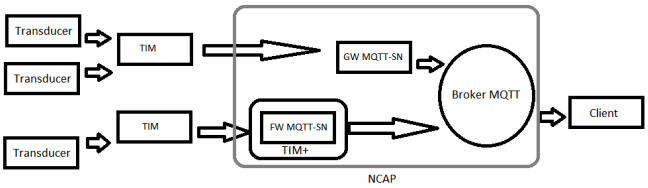

Fig. 4. Use case scenario.

IV. Results

To demonstrate how MQTT-SN can co-exist with MQTT and can improve the IEEE 1451, this section shows a simulation with MQTT-SN nodes the results of memory consumption. The simulator chosen was the Cooja from Contiki-OS[1], the implementation of MQTT-SN was shared on GitHub[2], and the nodes were Z1 motes.

The Figure 4 shows the simulated scenario, where 2 TIMs are connected to different sensors, get its data generated and send it to the next hop. The next hop, in this case, could be a TIM with more resources, called TIM+, which plays the role of "forwarder", or the GW MQTT-SN, responsible only to translate the "short" topic ID in an MQTT topic. The topics chosen for MQTT and MQTT-SN were "/node/home/temp/" and "22", respectively. The environment simulated used DHT11' sensors to get the measures of temperature.

The metric used in this use case to compare both protocols was the memory consumption by the messages' header. As expected, MQTT-SN presents a better performance than the MQTT in approximately 50% in the best case. It was considered only the header size plus a payload of the collected temperature. We believe that this result is because of the different sizes of these two protocols and the string of topic size from MQTT. All mentioned results can be seen in Figure 5, where the y-axis is the consumption in bytes and the x-axis the time evaluated.

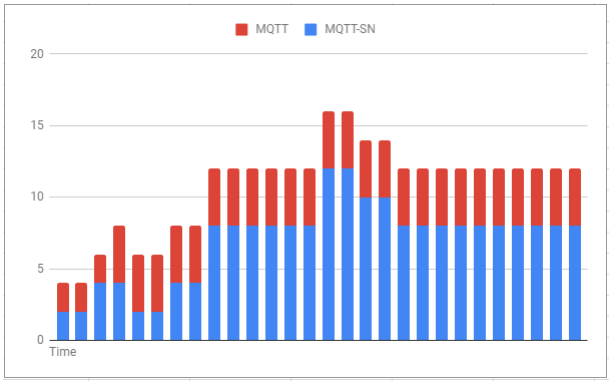

Fig. 5. Memory consumption of MQTT-SN and MQTT's messages.

[1]http://www.contiki-os.org
[2]https://github.com/aignacio/mqtt-sn-contiki_example

V. Conclusion

In this paper, the MQTT-SN, an improvement of IEEE 1451.1.6 - the MQTT in the 1451 family of standards - was introduced. There are many features and solutions that the MQTT can contribute to the family, and the new advances provide strong pieces of evidence to support this statement. MQTT-SN provides more than a new publish/subscribe messaging, yet another level of lightweight systems, enabling the increase of IEEE 1451 family of standards adoption on new environments. As future works, it is necessary to verify the delay and data loss in a real scenario for both protocols, MQTT and MQTT-SN.

References

[1] A. Foster, "A comparison between dds, amqp, mqtt, jms, rest and coap," *Messaging Technologies for the Industrial Internet and the Internet of Things*, vol. 1, no. 7, pp. 1–25, 2014.

[2] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol–http/1.1," Tech. Rep., 1999.

[3] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (coap)," Tech. Rep., 2014.

[4] I. B. M. C. EUROTECH, "Mqtt specification version 3.1," http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html, 2010, last access: 15 nov. 2018.

[5] A. Stanford-Clark and H. L. Truong, "Mqtt for sensor networks (mqtt-sn) protocol specification," *International business machines (IBM) Corporation version*, vol. 1, 2013.

[6] J. Dizdarevic, F. Carpio, A. Jukan, and X. Masip-Bruin, "Survey of communication protocols for internet-of-things and related challenges of fog and cloud computing integration," *CoRR*, vol. abs/1804.01747, 2018. [Online]. Available: http://arxiv.org/abs/1804.01747

[7] C. Severance, "Roy t. fielding: Understanding the rest style," *Computer*, vol. 48, no. 6, pp. 7–9, June 2015.

[8] Z. B. Babovic, J. Protic, and V. Milutinovic, "Web performance evaluation for internet of things applications," *IEEE Access*, vol. 4, pp. 6974–6992, 2016.

[9] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 3920, Oct. 2004. [Online]. Available: https://rfc-editor.org/rfc/rfc3920.txt

[10] J. Miller, "XMPP extensible messaging and presence protocol," 1999. [Online]. Available: https://xmpp.org/about/technology-overview.html

[11] A. Hornsby and E. Bail, "xmpp: Lightweight implementation for low power operating system contiki." in *ICUMT*. IEEE, 2009, pp. 1–5. [Online]. Available: http://dblp.uni-trier.de/db/conf/icumt/icumt2009.html#AdrianB09

[12] H. Wang, D. Xiong, P. Wang, and Y. Liu, "A lightweight xmpp publish/subscribe scheme for resource-constrained iot devices," *IEEE Access*, vol. 5, pp. 16 393–16 405, 2017.

[13] A. S. Clark, "Message queue telemetry transport (MQTT)," 1999. [Online]. Available: http://mqtt.org

[14] A. S. Clark and H. L. Truong, "MQTT for sensor networks (MQTT-SN) protocol specification," 2013. [Online]. Available: http://mqtt.org

[15] J. Velez, R. Trafford, M. Pierce, B. Thomson, E. Jastrzebski, and B. Lau, "Ieee 1451-1-6: Providing common network services over mqtt," in *2018 IEEE Sensors Applications Symposium (SAS)*, March 2018, pp. 1–6.