Towards an MQTT5 geo-location extension for location-aware applications

Felicien Ihirwe
Innovation Technology Services
Intecs Solutions Spa
Pisa, Italy
felicien.ihirwe@intecs.it

Giovanni Iovino
Fintech Business Unit
Intecs Solutions Spa
Pisa, Italy
giovanni.iovino@intecs.it

Davide Di Ruscio
Department of Information Engineering
Computer Science and Mathematics
University of L'Aquila
L'Aquila, Italy
davide.diruscio@univaq.it

2021 44th International Conference on Telecommunications and Signal Processing (TSP) | 978-1-6654-2933-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/TSP52935.2021.9522590

***Abstract*—Location-aware applications are becoming popular nowadays because of the increasing adoption of edge and fog computing. This poses novel difficulties to the conventional applications that utilize the Message Queuing Telemetry Transport (MQTT) as their prevalent communication channel. Also, developers are obliged to figure out the code at whatever point they need to build up MQTT-based location-aware applications. This can causes various issues, for example, protocol standard infringement or problems in data acquisition. This paper proposes a novel extension of the recently released MQTT5 protocol that will permit message delivery by not only respecting the topics of interest but also in addition to geo-referenced information provided by both publishers and subscribers.**

***Keywords*—MQTT5; wireless networks; location-aware applications**

I. Introduction

The increasing adoption of High-performance computing (HPC) systems and of 5G networks demands effective and well-designed data querying mechanisms to handle a large volume of real-time requests from different players [1]. Because of the current trend of edge/fog computing, in the case of moving nodes, their location will be one of the most important characteristics for node identifications in the entire considered ecosystem. In application domains like the Internet of things (IoT) and Smart Cities, large quantities of data must be shared across the network in which geo-location information will come as an additional burden to the message itself.

Message Queuing Telemetry Transport (MQTT) protocol [2] has demonstrated an extraordinary accomplishment as the communication channel among different IoT segments and is considered a promising protocol for 5G systems to enhance the connectivity between the physical and digital worlds. This is due to its simplicity, robustness, and security [3]. MQTT is a topic-based protocol, and even though it presents solid handling and messaging mechanisms, MQTT clients can experience high volumes of data pattern encoding/decoding in the message payloads. This may increase the client side's execution time due to the enormous information to be decoded, including real-time location data.

This work has received funding from the Lowcomote project under European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement n° 813884.

To address the above challenges, we propose to separate the geo-referenced data from the message payload. The geo-location extension of MQTT5 [4] will primarily target location-aware applications. The main potential contributions of the paper are threefold: *i)* it proposes a new extension of MQTT5 based approach, which will consider geo-location information from the publisher and the subscriber; *ii)* it presents the operational rules to guarantee back-compatibility with clients that do not use the proposed extension; *iii)* it presents our proposed approach's technical specifications to ensure its scalability and robustness.

Once implemented, the proposed approach will simplify IoT systems' design and development, where client positions' management is crucial. The new proposed approach will trigger the following advantages: *(1)* Discover a usable and straightforward mechanism to the applications that utilize standard MQTT5 communication protocol. *(2)* Simplify the integration of location-aware sub-system within the application. *(3)* Support of the previous version MQTT3.1.1 [2] in which any application which utilizes it can still receive and send messages as usual.

The remainder of the paper is organised into four sections. Section II examines the related work. Section III gives a concise overview of MQTT and the recent MQTT5 in particular. Section IV presents the technical specification of the proposed extension. Lastly, Section V highlights future work and concludes the paper.

II. Related work

Several approaches have been proposed to extend the Publish/Subscribe paradigm. For instance, in [5], [6], [7], [8], [9], [10] the authors propose approaches to accommodate location awareness for better service delivery. In [11] the authors propose *MQTT-G*, an extension of the MQTT protocol to include geo-location data. This extension utilizes the unused binary flags in the protocol definition itself and by alternatively inserting geo-area information between the header and payload. Differently from the work presented in this paper, in [11] the authors leveraged the third bit of the MQTT Control Packet-type definition. However, to adapt with the PUBLISH packet, the authors had to re-implement another packet called

PUBLISHg, which can make the approach not working with existing MQTT implementations.

In [12], the authors present the *GeoMQTT* extension, a full extension of the MQTT protocol to support the specification of spatial and temporal events. In this case, the authors introduced re-designed control packet types of MQTT and renamed them by adding a *Geo* prefix to reflect the new use. Even though this approach has been implemented and tested, it can have conflicts with the original MQTT. For instance, a client is subscribed to an MQTT topic filter of a GeoMQTT broker and receives GeoPublish message whose topic name matches the filter [13]. In this case, the client-side handling mechanism discards the additional information relating to locations.

In [14], [15] authors propose *GeoBroker*, a topic-based server-side data distribution service to control data distribution based on publisher or subscriber geo-context information. To achieve better message matching, the authors created an indexing structure to store all client's pub/sub information. When a request is issued, a three-level payload message checking is performed to determine which consumer or producer is intended to. *ContentCheck* matches the pub/sub topics, *Consumer GeoCheck* matches subscriptions' geofence information with the producer location whereas *Producer GeoCheck* checks whether the producer geofence of the message contains the corresponding consumer locations. An open-source proof-of-concept implementation is available.

However, the authors use the payload field to specify the geofence information, this can increase the execution time as every single message have to be decoded and checked for location information. Different from our approach, the message payload is not checked for pub/sub client matching but just the user property values in the variable header. This can extremely diminish the matching time and increase operational efficiency. The authors also did not explicitly clarify the message structure and how the users define their geofence shapes of interest.

III. MQTT PROTOCOL

MQTT protocol was established in 1999, as one of the primary IoT communication protocols. Created by IBM, MQTT aimed to empower a lightweight communication between different heterogeneous situations between nodes [16]. In 2013 IBM submitted the protocol to the OASIS specification body and in 2014, the MQTT 3.1.1 was officially released and referred to as the standard version of the protocol. MQTT utilizes a publish/subscribe mechanism, unlike other conventions that utilize the request/response model. To guarantee the best nature of its correspondence, it utilizes 3 unique levels to indicate the nature of administrations (fire and forget, conveyed only a single time, or conveyed at least once). MQTT is very fascinating contrasted with other correspondent protocols such as HTTP. It empowers a better message conveyance in a wide scope of IoT use cases with constrained environments and restricted transmission capacities. Besides, MQTT can scale up to billions of real-time connected devices [17].

To implement the publish/subscribe paradigm, MQTT is mainly based on the concept of *topics* that are used as links each time a client is interested in a particular data. The *broker* is a server-side application responsible for filtering topics and keep track of *publishers* and *subscribers*. It is also responsible for sending messages to subscribers, filtering the messages, and classifying the subscribers based on the topics that they have subscribed to. The broker also holds the sessions of all persisted clients, including subscriptions and missed messages [18].

Due to some pitfalls found in MQTT 3.1.1 related to robustness, transparency, and feedback of the users [19], the MQTT committee released a new version of the protocol in early 2018 known as MQTT5 [4] as free and open for contribution. The new version of the protocol offers backward compatibility with the previous version, and the increase in robustness, transparency, and ease of communication between devices has been addressed. The main objective of MQTT5 includes the enhancements for scalability, extensibility to large-scale systems, improved error reporting, and formalization for client discovery [4]. More than 20 major updates were released under the MQTT5 version, including session and message expiry, the format of the payload data specification, user properties, enhanced authentication, and authorization mechanism, shared subscription, etc. For the sake of space, we do not go over the details, and interested readers can refer to the online specification of the MQTT5 protocol.[1]

IV. PROPOSED APPROACH

The MQTT protocol works in such a way it shares message sequences of packets referred to as *MQTT Control Packets*. There are 15 standard types of control packets in MQTT. As previously mentioned, MQTT is a topic-based publish/subscribe protocol, in which the list of topics is usually known in advance, e.g., during the design phase of an application, even though the user can create them at run-time [20]. MQTT5 has provided clients with the means for sending or receiving the application layer name-value information whose meaning and translation are known distinctly by the application programs responsible for sending and accepting messages [4]. The MQTT Control Packet is divided into three main parts as represented in Table I.

TABLE I. MQTT CONTROL PACKET TOP-LEVEL STRUCTURE

Part name	Size	Mandatory
Fixed Header	Fixed	Yes
Variable Header	Not fixed	No
Payload	Not fixed	No

The *fixed header* is used to specify the packet's mandatory behavioural information such as the packet type, length, etc. This part is obligatory and fixed for all packets and should be satisfied as described in the MQTT5 wiki page [4]. The *variable header* is used to specify the packet's optional behaviour

[1] https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

TABLE II. THE TYPICAL STRUCTURE OF THE VARIABLE HEADER

Byte	Name	Typical presence in packets
1	Packet Identifier MSB	Yes
2	Packet Identifier LSB	Yes
3	Property length	Yes
3	Properties byte 1	No
4	...	...
..	...	...
N	Properties byte N	No

in the network. This part of the protocol is variable and can also be used to specify the payload's content characteristics. The *payload* is reserved for conveying application messages. For this research, we will only focus on the variable header spec as it is the basis of our approach.

As for the last release of the protocol, the variable header structure varies depending on the packet type. The current and typical structure of the variable header is shown in Table II. MQTT5 has defined several types of properties, and each of them has its standard identifier. *Payload Format Indicator(0X01)*, *Message Expiry Interval(0X02)*, *User Properties(0X26)*, *Authentication Method(0X15)* are examples of possible properties. Our approach lies in investigating the possible extensions of the variable header's field called *User property* to help carry the additional information such as geo-referenced data that the client can use as one of the facts when receiving or sending a message alongside the topic of interest.

A. The proposed User Property field

As previously mentioned, MQTT5 accepts the presence of additional properties according to the client's specific needs. We propose the adoption of a *User Property* field by passing various UTF-8 string pairs containing information related to geo-localized data (geo-localizing payload or *GLP*). Note that each property type in MQTT has a unique identifier to make it easily recognizable in the network. For instance, the *User property* identifier in hexadecimal format is *0x26*. So, this means that each key-value pair entry will correspond to one user property. Table III shows the format of the proposed user property structure.

TABLE III. PROPOSED EXTENSION

Byte number	Value
K	length
K+1	0x26
K+2 to K+p	GLP1
K+p+1	0x26
K+p+2 to K+q	GLP2
K+q+1	0x26
...	...
K+z to K+w	GLPn

(where K,p,z,w,n are integer)

The length of packets is calculated as the sum of *n+size(GLP1)+size(GLP2)+...+size(GLPn)*. The actual format of GLP is explained in Section IV-C.

The proposed approach allows for defining the specific geo-localized information from each location in which the client is interested. Note that such information is considered in addition to the topic when the broker selects the message to send to the subscribers. This approach still supports all MQTT5 based communication if the client is not interested in using our approach.

B. Operation rules

To guarantee back-compatibility with a client not using the proposed extension, the following rules are defined:

- A client which is interested in receiving messages from a specific geographical area **MUST** subscribe to the topic of interest including the addition of the data related to the area of interest.
- The client uses the GLP payload to announce its position.
- The information provided in the GLP is stored by the MQTT broker.
- When a client is interested in publishing the message to a specific topic by using our approach **SHOULD** use the GLP payload to provide information about the geographic area for which the message is relevant (if any).
- When the MQTT broker receives a publish message by a client, it **MUST** use the information provided in the GLP payload provided in the variable header property field, if present, to identify the clients to which the message should be forwarded, according also to the topic name and the QoS settings as in a normal MQTT communication.
- If a message is published by a client without a GLP payload the message MUST be forwarded to all the clients subscribed to the topic regardless of their position of interest.
- If a client subscribes to a topic without specifying any GLP payload, all the messages published to the topic **MUST** be delivered to the client, despite they include geo-localized information as defined in this extension or not.
- The updated position of a client can be retrieved by:
  - A renewal of the subscription.
  - The GLP payload when a client publishes a message to a topic.
  - The publish of a message to the dedicated topic position.
  - Any request for update done by the client.

C. The GLP payload

According to the *User Property* field, as defined in the recently released MQTT5 version, the User property can be defined as a set of UTF-8 string pairs holding name-value tags and in which it can appear multiple times. From this point, we choose to define the payload in a series of `key:value` strings format in which keys are standardised and known in advance. In this section, we give details on different area definitions and illustrate different GLP command examples. The proposed GLP payload is defined as shown in Listing 1.

```
{
    "gpl_current_pos":"lat;lon",
    "gpl_areaType":"1;unit;or;areaData",
    "gpl_areaType":"...",
    "gpl_areaType":"...",
    "gpl_areaType":"n;unit;or;areaData"
```

```
}
```

Listing 1. GLP payload format

The only mandatory entry is `gpl_current_pos`. The `gpl_areaType` varies depending on the area types as explained later in this section. Technically, in order to facilitate the message entry discovery, each key-value user property entry has always its own identification, i.e. 0X26 as identifying byte.

`gpl_current_pos` is a mandatory entry for all client-server communication. It gives the possibility for a client to send its current position in terms of latitude and longitude coordinates by following the format `"gpl_current_pos":"lat;lon"`.

`gpl_areaType` is a variable-length key-value pairs which defines the description of a geo-location area of interest. One operation can contain a combination of as many `gpl_areaType(s)` as possible according to the user interest. The following types are available as standard area shapes which can be used to define any complex shape.

- `glp_circleStatic` is used to describe a static circular area on a map with a fixed center.
- `glp_circleDynamic` is used to describe a dynamic circular area on a map with a dynamic center (client position).
- `glp_rectangleStatic` is used to describe a static forward oriented rectangle area on a map with a fixed center.
- `glp_rectangleDynamic` is used to describe a dynamic forward oriented rectangle area on a map with a variable center.
- `glp_line` is used to describe a linear path on a map with a corresponding thickness.

Each area entry value has the mandatory specification to abide by, in addition to other specific values to be provided according to its type as explained below:

- `id`: this mandatory area entry value defines the area ID in the whole listing of the areas. This entry should be defined as an integer and possibly auto-incremented.
- `unit`: this mandatory area entry value is used to define the unit used in the encoded area. Possible values are:
  - *km:* Integer number are expressed in meters;
  - *m:* Integer number are expressed in kilometers;
  - *dm:* Integer number are expressed in decimeters;
  - *cm:* Integer number are expressed in centimeters.
- `areaData`: this value entry is variable depending on the type of the area corresponding to the `gpl_areaType` stated above. The full area data definition model is described in detail in Section IV-D.
- `or`: it is used to specify if the client is interested in messages related to the area or not:
  - *1:* The client wants to receive messages from the area.
  - *0:* The client does not want receive messages from that area.

This entry is very important when defining complex shapes. This can be done by just sending an updated area with toggled `or` value. A simple related example is presented in Section IV-E

D. AreaData definition

The `areaData` value entries are defined differently depending on shape types. Figure 1 shows the kinds of areas that can be used to define complex shapes and that are described in the following.

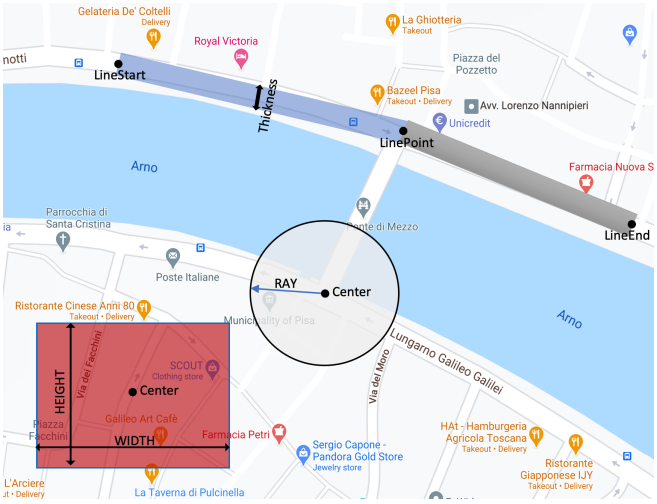

Fig. 1. Area encoding

Circle with an arbitrary center point: it is defined with the `gpl_circleStatic` type and it consists of a cyclic area with a remote center point (`center_lat` and `center_lon`) and a radius integer value. The `unit` value entry specifies radius's used unit (e.g., *km, m, cm, dm*). The definition of such a shape follows the following format

```
"gpl_circleStatic":"id;unit;or;center_lat;
                    center_lon;radius"
```

Listing 2. Circle with a remote center point

Circle with client position as center: it is defined with the `gpl_circleDynamic` type and it is represented by a point for the center (current client position), and an unsigned integer for the radius. The `unit` key entry indicates the unit used for the radius (e.g., *km, m, cm, dm*). The definition of such a shape follows the following format

```
"gpl_circleDynamic":"id;unit;or;radius"
```

Listing 3. Circle with the client position as center

Rectangle with an arbitrary center point: it is defined with the `gpl_rectangleStatic` and it is represented by a point for the center (`center_lat` and `center_lon`), plus two unsigned integer values to specify the height and the width of the rectangle. The `unit` key entry indicates the unit used for the dimensions of the rectangle (e.g., *km, m, cm, dm*). The definition of such a shape follows the following format

```
"gpl_rectangleStatic":"id;unit;or;center_lat;
                       center_lon;width;height"
```

Listing 4. Rectangle with an arbitrary center point

Rectangle with client position as center: it is defined with `gpl_rectangleDynamic` and it is represented by two integer values to specify the height and width of the rectangle. The `unit` key entry indicates the unit used for the dimensions of the rectangle (e.g., *km, m, cm, dm*). The definition of such a shape follows the following format

```
"gpl_rectangleDynamic":"id;unit;or;height;width"
```

Listing 5. Rectangle with the client position as center

Line: it is defined as a set of points and a corresponding thickness value. As far as our approach is concerned, a line is defined in terms of three point types:

- `gpl_lineStart` to define a unique starting point of the line and the general thickness of the line. It is represented as follows:

```
"gpl_lineStart":"id;unit;or;point_lat;point_lon;
    thickness"
```

Listing 6. Starting point of the line

- `gpl_linePoint` to define the intermediate points of the line. Such points are optional depending on the line to be defined and are represented as follows:

```
"gpl_linePoint":"id;unit;or;point_lat;point_lon"
```

Listing 7. Intermediate point of the line

- `gpl_lineEnd` to define the ending point of the line being specified and it is represented as follows:

```
"gpl_lineEnd":"id;unit;or;point_lat;point_lon"
```

Listing 8. End point of the line

E. Basic Command scenarios

In this section, we present different commands to describe several basic and complex scenarios, which can be covered by the proposed approach.

Area subscription: Figure 2 represents explanatory subscription scenarios. In particular, let us consider a client located at point ***O_0(O_0_lat,O_0_lon)*** and she wants to subscribe to all the indicated colored areas consisting of the circle C_2 having as center the client position O_0 and r_2 as radius. Moreover, the wanted subscription excludes the area in the circle C_1 with radius r_1. The subscription also includes the rectangle R_1 with the client position O_0 as center, and with height and width H_1 and W_1, respectively. The area represented by the circle C_1 is excluded.

The subscription also includes the circle C_3 with center ***O_3(O_3_lat,O_3_lon)*** and radius r_3. The user also subscribes to a static rectangle R_4 with center at ***O_4(O_4_lat,O_4_lon)*** and with height and width of H_4 and W_4 respectively. Finally the user subscribes to the lines, one from point A_1 to B_1 with thickness T_1 and the other line from A_2 through B_2 ending at the point C_2. The second line has the thickness T_2.

The above scenario can be specified by means of the commands shown in Listing 9. Note that all the distance entities, such as *line thickness*, *radius*, *height*, and *width* are expressed in meters.

```
{
    "gpl_current_pos":"O0_lat;O0_lon",
    "gpl_circleDynamic":"1;m;1;r2",
    "gpl_rectangleDynamic":"2;m;1;H1;W1",
    "gpl_circleStatic":"3;m;1;O3_lat;O3_lon;r3",
    "glp_rectangleStatic":"4;m;1;O4_lat;O4_lon;H4;W4"
    ,
    "gpl_lineStart":"5;m;1;A1_lat;A1_lon;T1",
    "gpl_lineEnd ":"6;m;1;B1_lat;B1_lon",
    "gpl_lineStart":"7;m;1;A2_lat;A2_lon;T2",
    "gpl_linePoint ":"8;m;1;B2_lat;B2_lon",
    "gpl_lineEnd ":"9;m;1;B2_lat;B2_lon",
    "gpl_circleDynamic":"10;m;0;r1"
}
```

Listing 9. Example of subscription commands

Note the last areas command expression have the `or` value entry sets to zero in order to unsubscribe from area C_1.

Location update: In this case the client wants to update their current location, the client will have to send the `PINGREQ` packet type with the GLP payload's *gpl_current_pos* set with the new *lat;lon* values. This scenario can be configured as a periodic event in case of real-time scenarios. An example of the packet that can be sent in this case is shown in Listing 10. Note that according to MQTT broker communication architecture, the broker has to send a `PINGRESP` packet to acknowledge the ping request.

```
    "gpl_current_pos":"new_lat;new_lon"
```

Listing 10. Update location command example

Update of the area of interest: This case can happen when the client needs to update the area of interest. In this case, the client has to send new PUBLISH/SUBSCRIBE commands with new values even though the client has to use the same centers as before otherwise, it will look like a brand new client request. In case of center changes i.e., moving client node, the broker should be able to adapt PINGREQ packets sent by the client accordingly.

Area unsubscription: This scenario can happen when the client wants to unsubscribe one or more areas that are no longer of interest. In that case, the client sends the same `SUBSCRIBE` command with area data defined but with the `or` entry value set to zero. In case the client wants to make a full unsubscription to all the previously subscribed areas, the client can use an `UNSUBSCRIBE` commands with no `gpl_areaType` defined. The full unsubscription can be specified as follows

```
    "gpl_current_pos":"lat;lon"
```

Listing 11. Full unsubscription command

V. Conclusion

Developers interested in a location-aware application whose communication depends on MQTT need to crack a robust client-side service to discern location information from the real payload message.

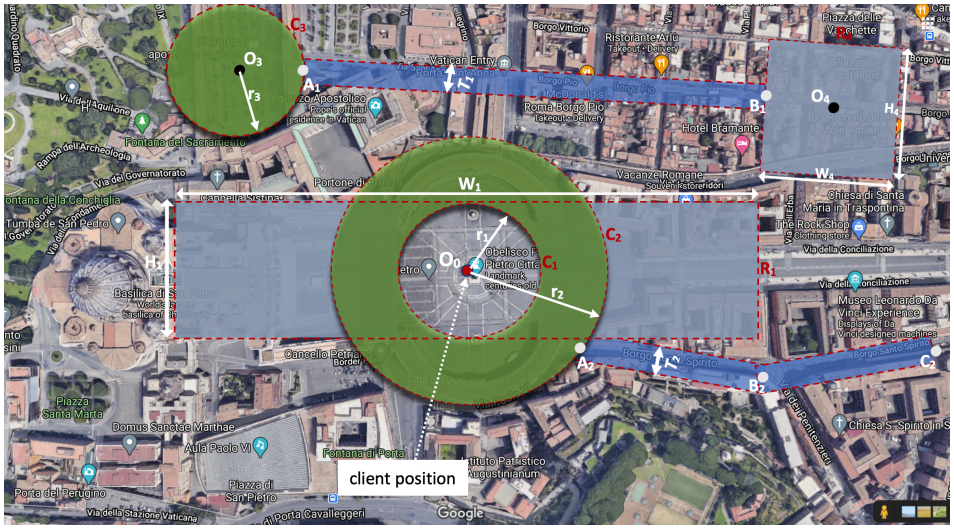

Fig. 2. Multiple area subscription example

In some cases, the client might be interested in subscribing only to publishers that are at a certain location. This may increase the implementation time and efforts, and in some cases disregards the data conveyance efficiency enabled by MQTT. To guarantee data decoupling and better transport, in this paper, we proposed an extension to the standard MQTT5 to maintain geo-location information by using user property fields. Employing the proposed approach, a customer can publish and subscribe by using the original MQTT topic mechanisms as well as based on his/her current location or a self-assertive point. We've also depicted the conceivable area encoding mechanisms considering different basic shapes of interest. As future work, we plan to focus more on the implementation of this approach by embedding it in a real use case. Finally, we would like also to focus on improving the efficiency of the encoding mechanisms for the irregular areas.

REFERENCES

[1] N. Panwar, S. Sharma, and A. K. Singh, "A survey on 5g: The next generation of mobile communication," *Physical Communication*, vol. 18, pp. 64–84, 2016.

[2] B. Andrew and G. Rahul. (2015, December) Mqtt version 3.1.1 plus errata 01 [online]. Last Accessed: September 2020. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

[3] A. Rahman, S. Roy, M. S. Kaiser, and M. S. Islam, "A lightweight multi-tier s-mqtt framework to secure communication between low-end iot nodes," in *2018 5th International Conference on Networking, Systems and Security (NSysS)*, 2018, pp. 1–6.

[4] B. Andrew, B. Ed, B. Ken, and G. Rahul. (2019, March) Mqtt version 5.0 [online]. Last Accessed: September 2020. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[5] L. Fiege, F. C. Gartner, O. Kasten, and A. Zeidler, "Supporting mobility in content-based publish/subscribe middleware," in *Middleware 2003*, M. Endler and D. Schmidt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 103–122.

[6] A. Devlić and I. Podnar, "Location-aware content delivery service using publish/subscribe," in *Proceedings of the tcmc2003, Workshop on Nomadic Data Services and Mobility*. Citeseer, 2003.

[7] P. Zhao, H. Jiang, J. Xu, V. S. Sheng, G. Liu, A. Liu, J. Wu, and Z. Cui, "Location-aware publish/subscribe index with complex boolean expressions," *World Wide Web*, vol. 20, no. 6, pp. 1363–1384, 2017. [Online]. Available: https://doi.org/10.1007/s11280-017-0439-z

[8] P. T. Eugster, B. Garbinato, and A. Holzer, "Location-based publish/subscribe," in *Fourth IEEE International Symposium on Network Computing and Applications*, 2005, pp. 279–282.

[9] G. Cugola and J. E. M. de Cote, "On introducing location awareness in publish-subscribe middleware," in *25th IEEE International Conference on Distributed Computing Systems Workshops*, 2005, pp. 377–382.

[10] D. Frey and G.-C. Roman, "Context-aware publish subscribe in mobile ad hoc networks," in *Coordination Models and Languages*, A. L. Murphy and J. Vitek, Eds., 2007, pp. 37–55.

[11] R. Bryce, T. Shaw, and G. Srivastava, "Mqtt-g: A publish/subscribe protocol with geolocation," in *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, 2018, pp. 1–4.

[12] S. Herle and J. Blankenbach, "Geopipes using geomqtt," in *Geospatial Data in a Changing World*, T. Sarjakoski, M. Y. Santos, and L. T. Sarjakoski, Eds. Springer International Publishing, 2016, pp. 383–398.

[13] S. Herle, R. Becker, and J. Blankenbach, "Bridging GeoMQTT and REST," in *CEUR Workshop Proceedings*, vol. 1762. CEUR-WS, 2016.

[14] D. Bermbach and J. Hasenburg, "Towards geo-context aware iot data distribution," in *Service-Oriented Computing – ICSOC 2019 Workshops*. Cham: Springer International Publishing, 2020, pp. 111–121.

[15] J. Hasenburg and D. Bermbach, "Geobroker: Leveraging geo-contexts for iot data distribution," *Computer Communications*, vol. 151, pp. 473–484, 2020.

[16] F.-J. Ferrández-Pastor, H. Mora, A. Jimeno-Morenilla, and B. Volckaert, "Deployment of IoT Edge and Fog Computing Technologies to Develop Smart Building Services," *Sustainability*, vol. 10, no. 11, pp. 1–23, October 2018.

[17] A. Hakiri, P. Berthou, A. Gokhale, and et al, "Publish/subscribe-enabled software defined networking for efficient and scalable iot communications," *IEEE Communications Magazine*, vol. 53, pp. 48–54, 2015.

[18] J. Kotak, A. Shah, A. Shah, and P. Rajdev, "A comparative analysis on security of MQTT brokers," in *IET Conference Publications*, 2019.

[19] S. Lee, H. Kim, D. Hong, and H. Ju, "Correlation analysis of mqtt loss and delay according to qos level," in *The International Conference on Information Networking 2013 (ICOIN)*, 2013, pp. 714–717.

[20] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks," in *3rd COMSWARE '08*. IEEE, jan 2008, pp. 791–798.