Limitations of the Pub/Sub Pattern for Cloud Based IoT and Their Implications

Daniel Happ, Adam Wolisz
Technische Universität Berlin, Telecommunication Networks Group (TKN)
{happ, wolisz}@tkn.tu-berlin.de

***Abstract*—The current approach to roll out large scale IoT systems is to outsource the crucial parts of the system to cloud based services, such as message brokerage, devices management, or sensor data storage and processing. One core protocol for messaging in those settings is the widely adopted publish/subscribe protocol MQTT. Pub/sub protocols, however, were not designed for this particular scenario and have decoupling properties that make some common task in IoT settings more challenging to achieve. It is, for instance, not straightforward to discover potential publishers of sensor data or to give guarantees that all, a certain number or at least one subscriber of a certain set of possible subscribers will received a message. Because they are missing in the standard, different approaches and implementations tackling those challenges will lead to incompatibilities between users of different systems. In this work, we therefore give an overview of the challenges with discovery and guaranteed delivery to a certain number of subscribers over pub/sub networks in IoT settings and present different possible solutions. We give advice on which implementation is useful under which circumstances and provide a proof-of-concept that can be used with little adaption for enabling discovery and reliability in MQTT.**

I. Introduction

The Internet of Things (IoT) is the vision of an upcoming ubiquitous network of interconnected physical objects, with a number of connected devices estimated to be in the order of billions [1], [2]. These objects can sense and interact with the physical environment, enabling deep-real time awareness of the world around us and enabling new upcoming applications like smart home, smart enterprise or smart city. The current IoT landscape is characterized by a large heterogeneity in various aspects: different vendors; different hardware specifications and capabilities, including processing power and means of communication with the outside world; diverse communication protocols; and distinct data formats and semantics.

Another aspect of this heterogeneity is the distributed control over these independent sensing and actuating devices by different operators. Since traditional sensor networks were tailored to a specific application, this has led to a plethora of existing silo solutions implemented mostly independently without any reference standard which now need to be connected and integrated into an internetwork of things. We rather envision an open system that overcomes those silo solutions.

The challenge to overcome this heterogeneity is commonly tackled by academics and industry using a cloud centric architecture similar to the one depicted in Figure 1 [2], [3]. Devices connect and send sensor data to a cloud tier using standardized protocols via a gateway. The cloud tier can operate additional services on the data, e.g. data storage, aggregation, or analysis. Individual components are connected using a message bus. User facing applications can get access to the data through the services or directly using the bus. Publish/subscribe has proven to be a suitable candidate for IoT messaging [3]–[5]. However, it was not specifically design for this use-case. Because of that, the decoupling properties deeply associated with pub/sub systems do not allow an easy discovery of sensor streams or reliable end-to-end transfer. This may lead to vendor-specific solutions, limiting interoperability. In this work, we outline the limitations of pub/sub systems, show different approaches to these limitations and show how they can be overcome using the MQTT protocol as an example.

The first challenge is the discovery of publishers, more specifically topics, to subscribe to. As the subscriber has no knowledge about potential publishers, it is unlikely to successfully guess the topic a publisher will choose to publish its messages. Common pub/sub systems, such as MQTT, do not offer discovery of publishers or topics, so topics to subscribe to have to be known in advance or negotiated over another channel. The lack of well-defined schemes of publisher discovery is a key challenge that has to be overcome to enable the interoperability of different IoT data providers and producers.

Another challenge is the guaranteed delivery when publishing to a distributed database. Guarantees are usually given for the delivery to the broker and from the broker to the subscriber; in case of MQTT at-least-once, at-most-once and exactly-once semantics are available. However, as the publisher does not know if or how many subscribers are subscribing to a given topic, the publisher cannot be sure if certain messages reach all, a certain fraction or at least one of the subscribers. This

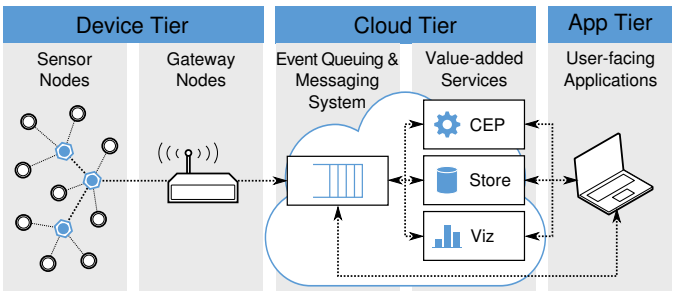


Fig. 1. General architecture of a cloud-centric IoT platform with cloud-based message bus and value-added services.

can be a mayor drawback if guaranteed delivery to a certain set of subscribers is important.

We illustrate those drawbacks with an example. Let us assume a car-sharing business, that has several cars that have sensors for location, availability and level of fuel. Cars are added and temporarily removed from the car pool because of repair work all the time. To keep the manual intervention minimal while provisioning new cars, the cars themselves should register with a service and be discovered by entities needing their information, such as web interface, database, and so on. That way not one of the service depending on the information would have to be changed. However, in many pub/sub systems, such as MQTT, there is no standard way of advertising that a sensor device, in our example car, will start publishing information on a certain topic. Likewise, if the car-sharing business intends to evaluate where cars are needed most to provide additional cars in certain areas, it has to ensure that all location data points are stored so that they can be analyzed later; for example by big data analytics. Database servers would subscribe to the location information, but in case the subscriber looses its connection to the pub/sub system, the publisher will not notice. There is no standard way in pub/sub systems to ensure at least one subscriber receives a message.

We try to trigger additional work on that topic by contributing in the following ways:

1) We emphasize on the limitations of the pub/sub pattern that arise from its symbolic addressing scheme with regard to IoT interoperability, namely the challenges of discovery and reliable message storage.
2) We analyze different approaches to cope with these limitations and to realize discovery and reliable storage on top of pub/sub systems.
3) We demonstrate how one of the identified solutions can be integrated into the existing de-facto standard protocol for IoT messaging MQTT.

We begin in Section II by motivating the use of pub/sub messaging for cloud based IoT platforms and emphasize the limitations of the pub/sub pattern. We give an overview of possible solutions in Section III. In Section V, we demonstrate how one of those solutions can be integrated into the MQTT protocol. We review related work in Section VI. Finally, in Section VII, we conclude our work.

II. Publish/Subscribe for IoT platforms

Usually, a message-oriented middleware provides pub/sub messaging. It offers distributed, asynchronous, loosely coupled many-to-many communication between message producers (publishers) and message consumers (subscribers) [6], [7]. Sensor data producers publish messages about events they have observed to the middleware. Consumers use the middleware to subscribe to event notifications they are interested in. The middleware matches messages against subscriptions and delivers messages accordingly.

Pub/sub enables the monitoring of sensor devices, but pure pub/sub protocols do not support direct messaging, for example to address and control actuators. Different types of filtering messages exist. The most widely adopted type in IoT deployments is topic-based filtering, where publishers address their messages to topics, which are usually strings which can also have a hierarchical structure. Subscriptions are likewise expressed as topic strings, optionally with wildcard characters to subscribe to multiple topics. QoS semantics are not mandatory, but commonly supported to varying degrees.

Three decoupling properties make pub/sub particularly appealing for IoT applications: 1) The asynchronous, non-blocking messaging decouples producer and consumer with regard to their synchronization. 2) The topic acts as a symbolic address, so that producers and consumers do not have to know or care about the addresses of other participants (also called loosely coupled in space). 3) Brokers further decouple publishers and subscribers in time, i.e. they do not have to be connected at the same time.

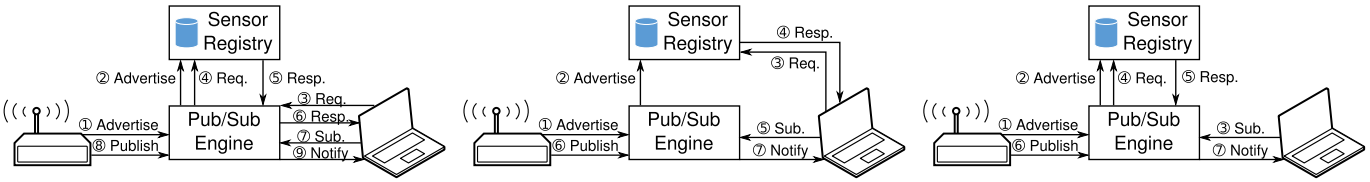
We see a trend to use pub/sub messaging to distribute sensor data to multiple interested applications in IoT settings [3]. MQTT is emerging as the de-facto standard for IoT messaging. Message Queue Telemetry Transport (MQTT) [8] is a pub/sub protocol specifically designed for constrained devices and high-latency, low-bandwidth, unreliable links, as omnipresent in IoT settings. Initially developed by IBM, it was standardized by OASIS in 2014. It can be used royalty-free, so that open source implementations are widely available, such as mosquitto broker and client library [9], Eclipse paho client library [10], or Apache ActiveMQ [11] broker. To run MQTT directly on sensor nodes, the slightly flattened MQTT-S was introduced [12].

While we consider pub/sub a useful pattern for IoT settings, we identify two mayor limitations of the pattern. Both drawbacks are a side-effect of the decoupling properties, in particular the decoupling in space. In pub/sub systems, the subscriber does not know if there are any publishers publishing on a given topic, how many of them there might be or who the publisher of a given message might be. Likewise, the publisher does not know if there are any subscribers actually interested in messages on a certain topic.

In MQTT, will-messages can be used to send a message to all subscribers in case the publisher disconnects unexpectedly. While this enables the subscriber to detect offline publishers, it is not defined in the standard how a subscriber should react to offline publishers or how guaranteed delivery can be achieved. This lack of a standardized way of guaranteeing at least one subscriber may pose a mayor challenge for the interoperability of systems in diverse administration domains, which may implement different approaches to achieve guaranteed delivery on top of existing pub/sub systems.

III. IoT Discovery Schemes for Pub/Sub

A common approach in pub/sub theory is for publishers to advertise their willingness to publish on a certain topic using a special advertisement message [6], [13]. We envision a similar advertisement for IoT devices, where the gateway device by proxy advertises its sensors on startup, on updates and in fixed intervals. It advertises not only its topic, but additional meta-data, like type of sensor, owner, sampling interval, cost. It would



(a) Option 1: Sensors advertise themselves to a sensor registry upon power up. Potential Clients query the sensor registry over the pub/sub system.

(b) Option 2: Sensors advertise themselves to a sensor registry upon power up. Potential Clients query the sensor registry using external mechanisms.

(c) Option 3: Sensors advertise themselves to a sensor registry upon power up. Potential Clients do not search for sensors themselves, but issue a subscription to the pub/sub system. The system queries the sensor registry and subscribes the subscriber to relevant sensor streams.

Fig. 2. Overview of three different options to enable sensor discovery over pub/sub networks.

not be feasible for potential subscribers to subscribe to those advertisements directly, as the number of devices would lead to a massive number of advertisements that would be flooded to all potential subscribers. Instead, the system provides a central registry that can be queried to find appropriate sensors to subscribe to. While logically centralized in the Cloud, this database would be distributed over multiple nodes to spread the load.

While advertisements are part of some pub/sub protocols, the protocols commonly used for IoT (MQTT, XMPP, AMQP) do not implement these messages. However, with those protocols, advertisements can be sent on a predefined topic or topic subtree. Based on the principle of advertisements, we differentiate between three distinct approaches, which we show in Figure 2 and present in the following.

All approaches share the same advertisement mechanism: Sensors advertise themselves regularly on a special topic. At least one of the possibly distributed repository servers subscribes to each of the topics intended for advertising. On receiving an advertisement, the database is updated accordingly. If no advertisement is received for a predefined timeout value, the sensor is considered offline and deleted from the repository.

The approaches differ in the way potential subscribers search for publishers and how they receive the response. The first approach is shown in Figure 2a. Potential Clients query the sensor registry using a request/response pattern over the pub/sub system. The advantage of this approach is that it does not rely on external protocols and just uses the available pub/sub system for sensor search. In pure pub/sub systems, no request/response messaging is offered, so dedicated topics would have to be used to emulate a request/response pattern, which is a clear disadvantage, or the protocol would have to be extended with a request/response pattern.

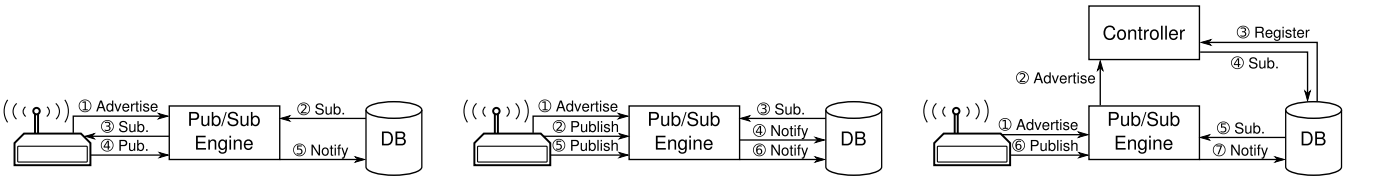
The second option is shown in Figure 2b. While advertisements are sent using the pub/sub sytstem, potential subscribers query the sensor registry using an external mechanisms, e.g. a RESTful interface. With this approach, the pub/sub system would not have to be modified. For the external search queries, a suitable request/response protocol can be used. A disadvantage is the usage of two distinct protocols for search and distribution of sensor data streams, so every application or service using sensor data would have to implement both interfaces.

The last approach is shown in Figure 2c. Both sensor advertisements and sensor queries are sent using the pub/sub system. The approach differs from the first approach in that potential subscribers do not search for sensors themselves and subscribe in a second step. Instead, they just issue a subscription to the pub/sub system including the search query. The pub/sub system implicitly queries the sensor registry and subscribes the subscriber to relevant sensor streams. The pub/sub system could also seamlessly query the registry again after a predefined time to update sensor sets the subscriber is subscribed to. A clear advantage is that subscribers and publishers exclusively use the pub/sub system, although available systems would have to be modified to incorporate search in subscription messages, which is a major disadvantage. The subscriber, similar as in traditional pub/sub, does not know who will be the producer of sensor readings it will get after a subscription. This can be an advantage if an abstract set of sensors with common characteristics is requested rather than particular sensors, especially if the set of sensors of interest is large or changing often. Likewise, the subscriber does not have full control over which sensors readings to get, which can be a disadvantage.

In real deployments, a request/response messaging as in option 1 is often used in pure pub/sub systems for commands or actuation. Some protocols already offer request/response messaging, e.g. XMPP or AMQP. For pure pub/sub protocols, such as MQTT, special topics are used. Option 2 is often used for sensor search as presented, e.g. over a web interface or HTTP request. Option 3 would correspond to a content-based pub/sub system, which none of the major protocol in use for IoT currently supports. As it only requires one protocol, can be used as is with available pub/sub systems and is already widely in use, we recommend using option 1 for sensor discovery in IoT settings.

IV. End-to-End Delivery in Pub/Sub Systems

To achieve guaranteed end-to-end delivery of messages, the straightforward approach is to introduce acknowledgments for successfully received messages. This is supported by MQTT, so that the broker can guarantee that messages reach each subscriber. Still, there is currently no way to enforce the policy to have messages delivered to a certain number of subscribers.



(a) Option 1: Broker indicates to publisher if subscriptions exist and publish successful; publisher stores data if no subscriber is present.

(b) Option 2: Publisher indicates to broker that data is to be stored; broker stores data until subscriber is present.

(c) Option 3: A dedicated controller is responsible for ensuring that at least one subscriber is subscribed to each topic.

Fig. 3. Overview of three approaches to guaranteed end-to-end delivery over pub/sub systems.

The publisher has no knowledge about subscribers and the broker cannot distinguish topics that no subscriber is interested in and disconnected subscribers that are supposed to store data.

The challenge is twofold: 1) One entity has to know how many subscribers are supposed to get a message and how many actually receive the message successfully; 2) One entity has to act accordingly when too few subscribers are receiving the message, usually by temporarily storing the messages and waiting for subscribers to return. We will assume in the rest of the paper that we want to ensure at least one subscriber is present. Additionally, we assume that all subscribers that subscribe to the topic are equally sufficient to fulfill this condition. In MQTT, without a specially configured broker, anyone can subscribe to any topic, so unrelated clients could trick the publisher into assuming that a message reached at least one of the intended recipients by subscribing. To ensure that all subscribers are really intended recipients, additional approaches have to be developed, such as advertising an access control list from publisher to broker.

One option for guaranteed delivery is depicted in Figure 3a. The publisher is responsible for ensuring that at least one subscriber is subscribed to its data at any time. In the event no subscriber is present, the publisher has to cache the data until a subscriber becomes available. For this, the publisher has to be informed when a subscriber subscribes to the published data stream. The approach we envision is to give the publisher the option to demand indications for subscribers in the aforementioned advertisement messages. The broker forwards the first subscription message on the publisher's sensor data stream to the publisher and indicates when the last subscriber has unsubscribed. Messages are acknowledged to the publisher only when one subscriber has received the message. This way, the publisher can cache data that would not be delivered to any subscriber.

An advantage of this approach is that publishers are aware of subscribers and can decide how to react on missing subscribers. One possibility to save energy, which is usually scarce on sensor nodes, would be to disable sampling altogether when no subscriber is interested in the data. It would also be possible to adapt the sampling frequency if no subscriber is active. A disadvantage is that the publisher, which is usually a more constraint device than both broker and subscriber is responsible for its sensor data and may have to store it for an indefinite period. The approach would need a definition of the advertisement messages that would be needed. As subscriptions are usually not forwarded to publishers, publishers would have to be changed to accept subscriptions and act accordingly. The message format of subscriptions would not have to be changed, though, as subscriptions already incorporate all the information necessary.

Another approach is shown in Figure 3b. Here, the publisher can also indicate that sensor data should reach at least one subscriber, but the broker is responsible for storing the data until a subscriber becomes available. This approach is the one that is most consistent with the original pub/sub model, because publishers and subscribers are still strongly decoupled in space, so do not have to know each other, and the broker is responsible for the distributions of the data.

A clear advantage of this scheme is that the messaging protocol does not have to be changed, apart from the introduction of an advertisement message that would also be needed for discovery. The storage is done on the broker side, where cloud storage is more easily upgradeable. A drawback of this approach is that publishers are still not aware of subscribers and may sample and send data that will never be needed.

The last approach is to let subscribers be responsible for getting all relevant sensor readings. This is depicted in Figure 3c. To ensure that the sensor data of all potential publishers are stored, it has to be ensured that at least one subscriber subscribes to the data with a QoS level that ensures guaranteed delivery between broker and subscriber. To achieve this, a controller would have to subscribe to the stream of advertisement messages and order one or more subscribers to subscribe to topics where storage is necessary. An advantage would be that this approach works with existing infrastructure, so it could be used on top of some of the existing IoT platforms in use. A disadvantaged is that it requires a reliable connection from controller to broker.

We propose a hybrid approach based on option 1 and 2. The reason is that it is desirable for the publisher to request some indication if subscribers are actually interested in the data provided. As most sensor devices operate on batteries, the knowledge that no subscriber is present could be exploited for example to save energy by not sensing and send the data at all. The broker on the other hand should be responsible for guaranteed delivery of messages, as publishing devices might be too constrained to cache messages for an indefinite period.

V. Integration into MQTT

We now show a potential integration of the schemes mentioned before into the MQTT protocol. We aim not to modify the core message format to ensure compatibility with legacy clients and brokers and only do optional extensions to the protocol, as developers would continue using available MQTT libraries.

The message format of MQTT is depicted in Figure 4. The wire-level protocol uses 4 bits to determine the message type. 14 message types are defined in the MQTT standard, the other two are reserved for future use. The DUP field indicates duplicate messages, the QoS level can be set to at-least-once, at-most-once and exactly-once delivery, and the retain flag obligates the broker to store the message and deliver it to new subscribers. The rest of the message consists of the remaining length and headers and payload depending on message type.

A. Discovery with Advertisements

We present one solution for sending advertisements over the MQTT protocol, as well as a solution to query a database. To comply with the standard, we propose to use a MQTT publication message for advertising on a reserved topic. We propose topics starting with the string "`$ADV`". This fits in with other reserved topics used by the mosquitto [9] MQTT broker. It already uses topics starting with "`$SYS`" for statistics.

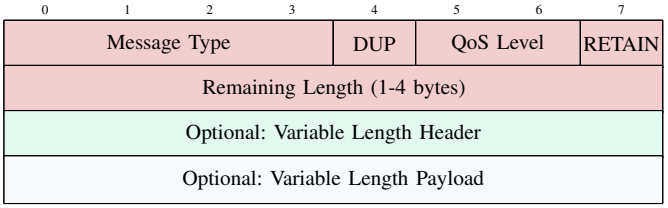


Fig. 4. MQTT message format: a mandatory fixed-length header (2 bytes) is followed by an optional message-specific variable length header and the actual message payload.

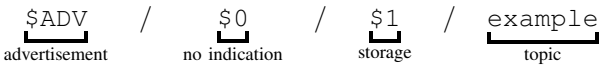


Fig. 5. Example of a special advertisement topic.

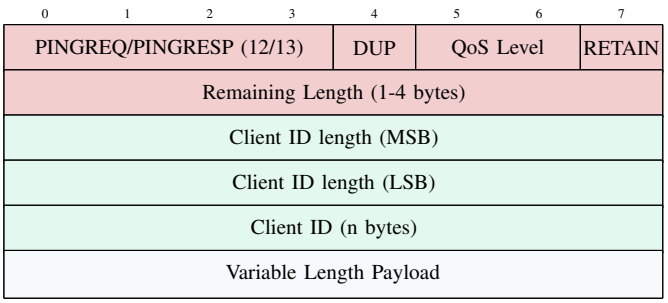


Fig. 6. MQTT PINGREQ/PINGRESP message format extension: client id and variable payload is added to otherwise empty PINGREQ/PINGRESP messages.

The publisher appends the topic string he is going to publish on to this string. Subscribers can now get those advertisement messages by subscribing on a the wildcard topic "`$ADV/#`". An extended MQTT broker should only let a predefined set of trusted registry servers issue such a subscription. The proposed scheme is payload agnostic. That means the payload of the advertisement does not affect the brokerage.

To enable clients to request meta-data about sensors from the registry, the request/response pattern is needed. For systems that have to use the unaltered standard MQTT, one solution is to have a topic the sensor registry is subscribed to. Client can publish requests on this topic. Requests include a response topic in the payload that the registry can use to reach the client. This scheme is complex and error-prone for a simple request/response message exchange. Because of that, we rather propose extending one of the existing message types defined in the standard. PINGREQ/PINGRESP messages are defined as keep-alive messages between broker and connected clients. They have no payload and therefore a length-field of zero. We generalize this message type to optionally include a destination user and payload field and to set the length field accordingly. The resulting message type fits into MQTT as a mixture of a publish message and the original ping message and can be used to reach a client directly by its identifier. The response uses the PINGRESP type. This solution is not compliant with the current standard, but keeps the approach backward compatible. That means an extended broker will interpret PINGREQ/PINGRESP messages from legacy client without a problem. The new messages can be used to query the registry with existing query languages, such as SQL, and get a direct response.

B. Indication & reliable distribution

The goal of this extension is to enable the client to request subscription forwarding from the broker to the client and to request storage for individual messages if no subscriber is present. We propose to add this functionality to the advertisement messages defined above. Basically, the client can request two modes of operation for the subscription forwarding and two modes for the message storage: both can be switched on or off for each topic that is advertised. The default state would be off and represent the standard MQTT behavior. Message storage on the broker may also be required on a per message basis, so in the storage enabled mode, the client would further need to indicate which message to store.

We propose to extend the advertisement scheme by introducing two switches after the "`$ADV`" string, the first for subscription forwarding mode and the second for message storage mode, both being either "`$0`" or "`$1`". If those special markers are found by the broker, the appropriate modes are set if supported. In the message storage mode, we propose that the client indicates a wish to store a certain message using the existing retain field. The field would have a different meaning than in the standard, as not only the last retained message would be sent to every newly connecting subscriber, but all retained messages would only be stored if there is no subscriber and sent only to the next client subscribing to the topic.

This scheme is backward compatible in the sense that both legacy clients and legacy servers would accept all messages used. Legacy brokers would not interpret the messages, so would not offer the subscription indication and message storage service outlined in this section.

VI. Related Work

Our work fits into existing literature focusing on cloud based IoT platforms. The authors of [2] present their vision of IoT and coin the term cloud centric IoT for a system offering data distribution and additional services in a cloud based layer. In [3], the authors find in an analysis of current IoT platforms that those systems share this architecture.

Publish/subscribe-based systems prove to be a suitable data dissemination pattern for IoT platforms, their services and applications. The Sensor Andrew project [4] is an IoT platform using XMPP for data dissemination. The more recent OpenIoT project [14] uses the CUPUS pub/sub system [5], a specifically designed content-based pub/sub with the ability to have mobile brokers, i.e. mobile pub/sub enabled gateways. The authors of [3] find that most currently deployed commercial and academic IoT platforms use (topic-based) pub/sub for messaging.

MQTT is widely accepted as one of the de-facto standard protocol for IoT applications. In [15], the authors emphasize on the wide availability of implementations, e.g. [9], and the recognition by standardizing entities, e.g. the binding of the oneM2M protocols to MQTT [16]. With MQTT-S [12], there is a stripped-down version of MQTT available, which is optimized for the small frame sizes commonly found in traditional wireless sensor networks, enabling sensor devices to directly participate in IoT pub/sub systems.

In [17], the authors argue that a self-configuring discovery service for sensors streams is needed and present a system based on DNS. The authors of [13] present a formal definition of advertisement messages in pub/sub systems. In [6], advertisement messages are defined as a way of informing the subscriber of new information. While advertisements are not part of MQTT, MQTT offers QoS semantics, so enables guaranteed delivery between client and broker [8]. However, as publishers and subscribers are still decoupled in space, they have no knowledge of potential publishers or subscribers.

VII. Conclusion

As the vast majority of IoT systems use pub/sub systems, in particular MQTT, and discovery and guaranteed delivery are key features of an IoT platform, new approaches to overcome these limitations on top of existing pub/sub systems certainly require further investigation. In this paper, we try to overcome this gap by further investigating the prevailing challenges in the IoT landscape currently deployed.

In this work, we argue that the pub/sub message pattern is in general suitable for IoT messaging. We also show that pub/sub protocols currently used do not meet all requirements necessary for seamless interoperability of existing silo solutions. We summarize the limitations of the pub/sub pattern that arise from its symbolic addressing and analyze various approaches to enable discovery and guaranteed end-to-end message delivery.

Nevertheless, we show that missing functionality can be added to existing pub/sub protocols using MQTT as an example. At present stage, we have implemented as a proof of concept the request/response pattern needed for registry lookups on top of the mosquitto MQTT broker and the corresponding client library [9]. We are in the process of implementing the proposed schemes for subscription forwarding and message storage at the broker.

References

[1] M. Chui, M. Löffler, and R. Roberts, "The internet of things," *McKinsey Quarterly*, vol. 2, pp. 1–9, Mar. 2010.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[3] T. Menzel, N. Karowski, D. Happ, V. Handziski, and A. Wolisz, "Social sensor cloud: An architecture meeting cloud-centric iot platform requirements," Apr. 2014, 9th KuVS NGSDP Expert Talk on Next Generation Service Delivery Platforms.

[4] A. Rowe, M. E. Berges, G. Bhatia, E. Goldman, R. Rajkumar, J. H. Garrett, J. M. Moura, and L. Soibelman, "Sensor Andrew: Large-scale campus-wide sensing and actuation," *IBM Journal of Research and Development*, vol. 55, no. 1.2, pp. 6:1–6:14, Jan. 2011.

[5] A. Antonic, K. Roankovic, M. Marjanovic, K. Pripuic, and I. P. Zarko, "A mobile crowdsensing ecosystem enabled by a cloud-based publish/subscribe middleware," in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*. IEEE, 2014, pp. 107–114.

[6] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, Jun. 2003.

[7] E. Curry, "Message-oriented middleware," in *Middleware for Communications*, Q. H. Mahmoud, Ed. John Wiley & Sons, 2005, ch. 1, pp. 1–28.

[8] A. Banks and R. Gupta, "MQTT Version 3.1.1," Oct. 2014, OASIS Standard.

[9] Mosquitto, "An open source mqtt v3.1/v3.1.1 broker." [Online]. Available: http://mosquitto.org/

[10] Eclipse Foundation, "Paho." [Online]. Available: https://eclipse.org/paho/

[11] Apache Software Foundation, "ActiveMQ." [Online]. Available: http://activemq.apache.org/

[12] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S – A publish/subscribe protocol for Wireless Sensor Networks," in *3rd Int. Conf. on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*, Bangalore, India, Jan. 2008, pp. 791–798.

[13] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, no. 3, pp. 332–383, Aug. 2001. [Online]. Available: http://doi.acm.org/10.1145/380749.380767

[14] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, I. P. Žarko *et al.*, "Openiot: Open source internet-of-things in the cloud," in *Interoperability and Open-Source Solutions for the Internet of Things*. Springer, 2015, pp. 13–25.

[15] A. Antonic, M. Marjanovic, P. Skocir, and I. Zarko, "Comparison of the cupus middleware and mqtt protocol for smart city services," in *Telecommunications (ConTEL), 2015 13th International Conference on*, July 2015, pp. 1–8.

[16] oneM2M, "Mqtt protocol binding," Jan 2015, version: TS-0010-V1.0.1.

[17] R. Klauck and M. Kirsche, "Bonjour contiki: A case study of a dns-based discovery service for the internet of things," in *Ad-hoc, Mobile, and Wireless Networks*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7363, pp. 316–329.

[18] IDABC, "European interoperability framework for pan-european egovernment services," 2004, version 1.0.

[19] World Economic Forum and Accenture, "Industrial Internet of Things: Unleashing the Potential of Connected Products and Services," http://www3.weforum.org/docs/WEFUSA_IndustrialInternet_Report2015.pdf, World Economic Forum, Tech. Rep., Jan. 2015.