Impact of Mobility on Message Oriented Middleware (MOM) Protocols for Collaboration in Transportation

Jorge E. Luzuriaga*, Miguel Perez†, Pablo Boronat†, Juan Carlos Cano*, Carlos Calafate*, Pietro Manzoni*
*Department of Computer Engineering
Universitat Politècnica de València, Valencia, SPAIN
jorlu@upv.es, {jucano,calafate,pmanzoni}@disca.upv.es
†Universitat Jaume I, Castelló de la Plana, SPAIN
{mperez,boronat}@uji.es

***Abstract*—Message Oriented Middleware (*MOM*) refers to the software infrastructure that support ubiquitous information delivery among software and hardware systems. Two of the most relevant protocols in this context are AMQP and MQTT. Lately, they have been extensively used to exchange messages conserving network bandwidth, device memory and batteries. These protocols provide an abstraction of the communication programming details of the different participating system entities, alleviating their coordination and collaboration.**

However, these protocols have not been thoroughly tested focused on studying the impact of node mobility. In this paper we present an experimental evaluation of both protocols quantifying the effect of the node mobility in terms of message loss, latency, jitter and saturation boundary values. Based on the results obtained, we provide criteria of applicability of these protocols. This evaluation is of interest for the upcoming applications that can be supported by MOM, and in especial for communication in Machine to Machine (*M2M*) and Internet of Things (*IoT*).

I. Introduction

Communication is essential to node coordination. Modern Internet-based applications are becoming more oriented towards the interaction of wireless and mobile devices with cloud resources and services. For many years HTTP has been used as the reference communications protocol in this context. HTTP is a very wide spread protocol, and APIs for its use are available basically for every programming language.

However, more flexible middleware systems have been developed to ease the design of cloud-based applications. Significant research efforts have been dedicated to define new communication systems that connect distributed components via message passing; one of them is the *Message-Oriented Middleware* (MOM). The basic idea of MOM are publish/subscribe systems to deliver messages from publishers via the message-brokers to subscribers based on their interests. This architectural model is a key components for real time collaboration experience. Based on this model many protocols have been developed, e.g. AMQP, MQTT, STOMP, DDS, XMPP. Here, we focus on two open protocols that have become in standards (a) the *Advanced Message Queuing Protocol* (AMQP), and (b) the *Message Queuing Telemetry Transport* (MQTT).

AMQP was created in 2003 by John O'Hara of JPMorgan Chase [8] and MQTT was originated in 1999 by Andy Stanford-Clark of IBM [18]. Both protocols ensure that the information is safely transported between systems, offering a number of advantages including efficiency (in use of resources), flexibility (content-based publish/subscribe), fault-tolerance (support to off-line operations), and ease to use (there are libraries available for the most popular programming languages, and there are implementations for the most common operating systems and platforms). In addition, they take into consideration security and confidentiality issues without affecting significantly the communications performance.

AMQP is an open standard for enterprise messaging, designed to support messaging for almost any distributed or business application [5]. It works like instant messaging or email the difference towards these available solutions is that AMQP comprises both (a) the network protocol, which specifies the entities (producer, consumer, broker) to interoperate with each other, and (b) the protocol model, which specifies the message representation and the methods to interoperate among the entities. Furthermore, the data content in an AMQP message is opaque, immutable and self-contained. There is no limit for the size of a message, it can be as large as or greater than several gigabytes [16]

MQTT is a lightweight machine-to-machine messaging protocol [2]. It fits perfectly for constrained devices with very limited resources, it has a clear focus on the mobile sector. Its information exchange procedure is resource-efficient, and it does not specify a particular data format. Additionally, it provides security that all messages transmitted even if the connection breaks off briefly, solving problems that arise upon unreliable communications.

The mechanisms to support message delivery allow a wide variety of usage scenarios, AMQP offers possibilities such as *point-to-point*, *store-and-forward*. However, the MQTT protocol provides a basic messaging based on specified topics (subscription's name) without possibility of *store-and-forward* [12]

In this article, we evaluate experimentally AMQP and MQTT protocols fundamentally focused on compare their capabilities and capacities through measurements in situations of node movement. We consider that among the major factors that effect the wireless network reliability and computation quality are situations such as significant path loss, attenuation of transmitted signals, loss of line-of-sight, interference among adjacent wireless networks occur among other things resulting from node's movement. Thus this study that will be replicated in any wireless network (in general BAN, PAN and WLAN) where the collaborative use of the information collected by individual nodes will form an enhanced dataset to improve any situation of our day to day life, like in many applications of physiological monitoring, inter-vehicular safety, remote sensing of environmental conditions, and transportation planning, analysis and monitoring. Our goal is to determine whether these protocols provide a satisfactory service depending of the applications' load needs in terms of message size and communication rates.

In order to emulate applications of remote sensing in transportation infrastructure, we use a simple scenario composed by one producer/publisher node (sensors to acquire information about objects or phenomenons), one consumer/subscriber node (to view, analyse and make decisions about the environment) and one message-broker (for matching subscriptions with publications) using specific hardware devices to observe the effect of the producer/publisher node moving from one Wi-Fi access point (AP) to another in the same IP network. Then, we present the evaluation results on the messages' jitter, and the saturation boundary values. With this purpose, we have been designed, developed, tested, and evaluated a synthetic load generator, called *momperf* to generate and publish messages with different load patterns. A message generated with this tool in its payload includes: a sequence number (to detect messages that can be lost, delivered out-of-order, or duplicated), the size of the current message and the interval of time in which they are sent by the producer/publisher node.

The rest of the article is structured as follows: Section II presents a literature review related to the topic. Section III provides a description of the methodology used in this work, showing how measurements have been done in order to be reproducible. Section IV presents the results and, finally, Section V concludes the paper, highlighting the next steps to follow in this research line.

II. Related work

The mobility concept has been profoundly researched in the last decade, many research groups and individuals have focused on node mobility support from many different points of views and providing many useful approaches. For example, in [9], the authors propose a middleware sits between the application and transport layers that use the store-and-forward feature of delay-tolerant network technology (DTN) and concepts of publish and subscribe employing AMQP protocol and Qpid message-broker, to provide resilient message dissemination even without network support. As a proof of concept, they present a prototype to Android devices inspired in a neighbourhood watch system with watch volunteers reporting incidents in order to reduce burglaries.

In [1] a theoretically middleware service is used to bridge things (the low level interfaces) and applications. To support devices that quickly moving between networks, re-associating with public access points, connecting to multiple networks and multiple technologies. Using a global unique identifier to separate the naming and addressing for every networked object (like HIP protocol where a device's identity is separate from its network address). And introducing tree network services: (a) a service of name resolution, (b) a routing module based in the identifier, and (c) delay-tolerance data delivery. Thus they handle mobile devices and context services supporting its mobility and seamless hand-off, but large scale evaluations of this proposal are pending.

The AMQP and MQTT protocols have proved to be useful in production scenarios, and have been used in challenging applications, including Autonomous Computing [3], Cloud computing [13] or in aspects related to the IoT [14].

There are several works in which the AMQP and MQTT protocols are evaluated separately. In [10] the performance of AMQP is assessed using Infiniband and Gigabit Ethernet networks with Qpid as AMQP middleware. Five simple synthetic benchmarks modeled after the OSU Micro-benchmarks for MPI were used. They exercise the number of Publishers, the number of Consumers, and the Exchange type. Each benchmark measures performance for data capacity (the amount of raw data in MegaBytes per second), message rate (the number of discrete messages transmitted), and speed (average time one message takes to travel from the publisher to the consumer).

In [4] a performance comparison between AMQP and RESTful *web services* is presented. Three different tests are performed, which consist of several client applications sending messages during 30 minutes to the broker or the web server, respectively; once the messages arrive to the server they are stored in a database. Then, the average number of messages per second that have been sent is compared to the total number of messages stored in the database. They conclude that, when the AMQP protocol is used to exchange messages, a larger number of messages per second is supported.

A study about MQTT, a "light weight" publish-subscribe based messaging protocol, is presented in [6]. The correlation between the end-to-end latency and loss of system messages is studied. Three different QoS levels with different sizes of payload (from 1 to 16 Kbytes) are tested on a real world scenario with both wired and wireless clients using 3G. They prove that there is a strong correlation between these two variables.

However, few studies have focused on the mobility impact in the effectiveness of both protocols

III. Methodology of the experiments

The *publish/subscribe* topology allows us to use a simple scenario with decoupled components in order to inter-operate among them. In our experiments, we use a message *producer/publisher* node located at the edge of the network, which at a given period of time, produce and publish AMQP/MQTT messages of a prefixed size to the message-broker.

In the case of AMQP, the message-broker accepts incoming messages from a producer node in an exchange (an exchange is essentially a router [11]) and, based on a set of criterions routes the messages to a specific queue. In the case of MQTT, the message-broker forwards the incoming messages from publisher nodes directly to the subscriber nodes. A subscription is initially created by a client application with a simple subscription name or a predefined topic.

In our scenario, the message-broker and the *consumer/subscriber* node are executed on the same computer

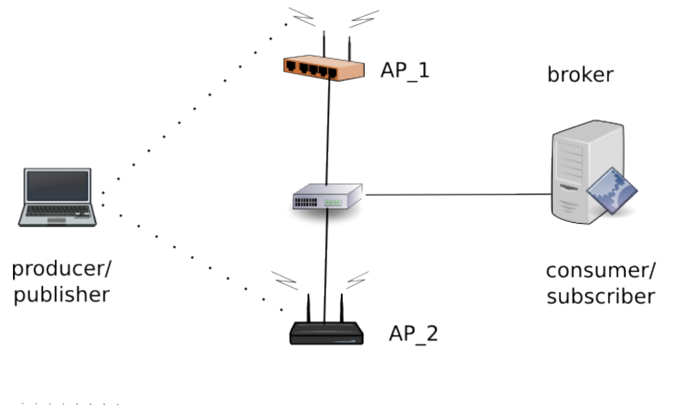


Fig. 1: A picture of the scenario.

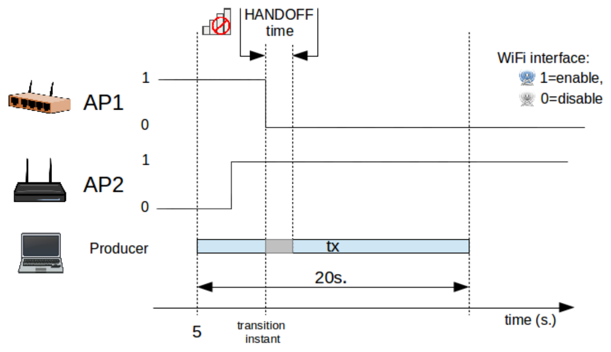


Fig. 2: Simulating the node mobility in an indoor scenario.

in order to avoid the latency that can be introduced by the message trip from the message-broker to the consumer node. The *producer/publisher* node is connected to a one of the two Wi-Fi access points. Both Wi-Fi access points offer paths to and from the message-broker. All these entities are connected within the same network. The diagram of the scenario configuration is depicted in Figure 1.

In the testbed, to generate workloads for the message queuing system with both MOM protocols we have been developed a testing application (which we call *momperf*). *Momperf* works as a message *producer/publisher* and *consumer/subscriber*.

Notice that *momperf* uses the *RabbitMQ* [15] and *Paho* [17] libraries, which are open source implementations of AMQP and MQTT protocols respectively. These current implementations are using TCP/IP connections for these communications. Indeed, this is needed to enhance reliability.

If the producer/publisher's connection suffers an interruption, the client (producer's AMQP or publisher's MQTT) stores the messages in their local buffers and then transmit them when the connection that reaches the message-broker is re-established. The problem appears when, the storage buffer capacity of the *producer/publisher* device is depleted, thereby causing message losses.

In the arrival event of each message, the sequence number and the production timestamps are recorded in a log file together with the reception time stamp. When there are changes in the *producer/publisher* link, the regularity of message reception is affected. Since the inter-message times are modified, message bursts can be delivered to the consumer, and even the sending order can be changed.

The duration of each test was about 20 *seconds*, which is enough time to check the access point migration of the message producer. During the tests we checked whether there were messages losses or if messages arrived out of order.

To the simulation of the node mobility in an indoor scenario, we used a set of scripts that shut down and activate the routers' radios, in this way we get disassociation/association of all client devices. A schema of this approach is shown in Figure 2.

Due to the asynchrony among the internal clocks of the different entities of a distributed system we cannot obtain an accurate latency value [7]. Instead, we have calculated the variation in the delay of the received messages, i.e., the *jitter*. Let us consider two consecutive messages that have been received by the consumer node, for message n, t'_n is its arrival time. T is the inter-message production period and it is one of the variables fixed for each experiment. To the n^{th} message, its inter-arrival jitter time is computed through the following formula:

$$J_n = t'_n - t'_{n-1} - T$$

Note that, with this formula, we are not concerned by a possible asynchrony among the producer, broker and consumer.

In our tests, we have taken as a reference bandwidth, the bandwidth needed to support high definition video streaming, that is about 5 Mbps. With *momperf* tool this value can be reached, for instance, by transmitting messages of $12500B$ ($12.5KBytes$) every 0.02 seconds. To detect the point at which messages start being lost, we made some tests in both cases: with and without mobility of the producer node. The obtained values are detailed in the following section.

The technical details of our testbed are: the message broker was deployed on a server with an AMD 8-core processor and $16GBytes$ of RAM memory. The mobile client had an Atom N450 processor and $1GByte$ of RAM memory. Both of them were running Ubuntu 12.04 GNU/Linux distribution. For the wireless network, we have used the OpenWRT GNU/Linux distribution with *Attitude Adjustment* version on an Alix PC-Enginees (alix2d2) and a Tplink (TL-WDR3600) routers.

IV. Results

In this section, we present our experimental results. The scenario is focused on the pattern of devices quickly moving between networks (from one access point to another maintaining the same IP address) while producing and publishing messages with stream-based applications. Indeed the TCP connection between the *producer/publisher* and the message broker is slightly affected.

For the experiments, we have used a completely dedicated network without external traffic. Each test was repeated 100 times for each combination of inter-message period and message size. The data message size is between 0.5 *KBytes* and

6 *KBytes*, and message production intervals of 10, 50, 100, 500 and 1000 *ms*. Then, we analyse the behaviour for each scenario, the results were generalized through a cumulative distribution function.

A. *Behaviour during access point transition*

When the communication link is stable and reliable, the jitter values to each message are around zero. In the mobility case, when a producer is migrating from one access point to another, the delay jitter has a considerable value.

Figure 3 show the typical jitter behaviour of a message received by the *consumer/subscriber*. These figures were obtained producing/publishing messages of different sizes with different inter-message period from 10 to 1000]*ms* during 20 *seconds*. As can be seen both protocols seem to perform very similarly.

In Figures 3, we can see certain points that reach values between 3.1 and 3.3 *seconds*, these points reveal the value of delay jitter that is associated with the corresponding hand-off time in each transition among the Wi-Fi access points. Also, we can see some negative values that belong to the reception of a burst of messages which the producer retained during the communication's interruption. The number of burst messages (that present a negative jitter) can be approximately calculated by the value of the maximum jitter (hand-off time) divided by the message producing period.

We have found that during the message burst, using the AMQP protocol (the Figure 3 *left*) the messages are consumed in inverted order. The delivery no guarantees the message delivery in order. It follows a LIFO (last-input first-output) order. This does not occur with MQTT protocol where the burst delivery of message retained in the transition do not affect the order. The LIFO behaviour during the burst of messages following the hand off may be due to implementation details, because it is not specified as a feature of the protocol.

B. *Jitter analysis*

The messages were sent with exactly the right intervals between them at the broker, depending on the conditions to transit in the network, these messages reach the consumer/subscriber at different amounts of time. Taking the difference of the arrival timestamps according with the previous formula, the jitter values of each message with a fixed node are around zero. We know that the maximum jitter value in our tests is the consequence of the node movement, given that the network had no external traffic, and that the workloads used in these tests do not saturate the system.

Using the Cumulative Distribution Function on the set of experimental data, we have analysed the behaviour of the message jitter focusing on the instant when the producer makes an access point migration.

Due the space limitations, in Figure 4 we show only a few cases of the distribution function of the jitter. In example using a period of 10*ms* for message production and message sizes of 512 *Bytes* and 6 *KBytes* for each protocol tested. When message size is 512 *Bytes* (Figures 4a and 4b), we observe that the jitter value is concentrated around 3.3 *seconds*, with sporadic cases of jitters of 7 *seconds*, without significant differences between these protocols, when the number of bytes sent increases (Figures 4c and 4d) the message rate drops. About half of the cases present jitter values close to 3.3 *seconds* while the other half of the cases double this value. We consider that this behaviour for bigger messages is due to the fragmentation of their payload by both protocols.

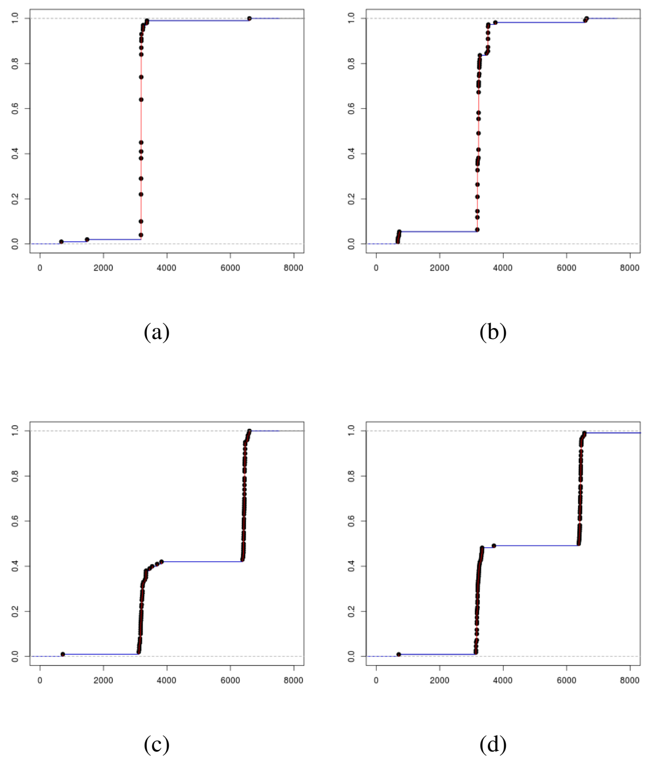


Fig. 4: Cumulative Distribution Function of the maximum jitter value (on *x*-axis). Using: AMQP (left) and MQTT (right). With a inter-message production period of 10*ms*, and messages size of 512*Bytes* (*a* and *b*) and 6*KBytes* (*c* and *d*).

To study the jitter evolution, we have used the statistical analysis for rounding mode values (rounded to the nearest hundred) to fit the most representative value instead of using the value that appears most often in the data sets.

We have represented the jitter mode value in two ways: as a function of the message size (Figure 5a) and as function of the inter-message period (Figure 5b). Where for different message lengths the jitter value is about constant, except when the periods between each published message are of 10*ms*, because in the measurements show the deleterious effects of mobility with jitter mode values was around the double.

C. *Workload boundary*

In order to know the capacity of the messaging system to handle heavy workloads, we executed the experiments without node movement. There is, therefore, no interruption in the wireless link between the *producer/publisher* node and messaging broker. Note that these saturation boundary values can be dependent on the platform used, and even on their configuration.

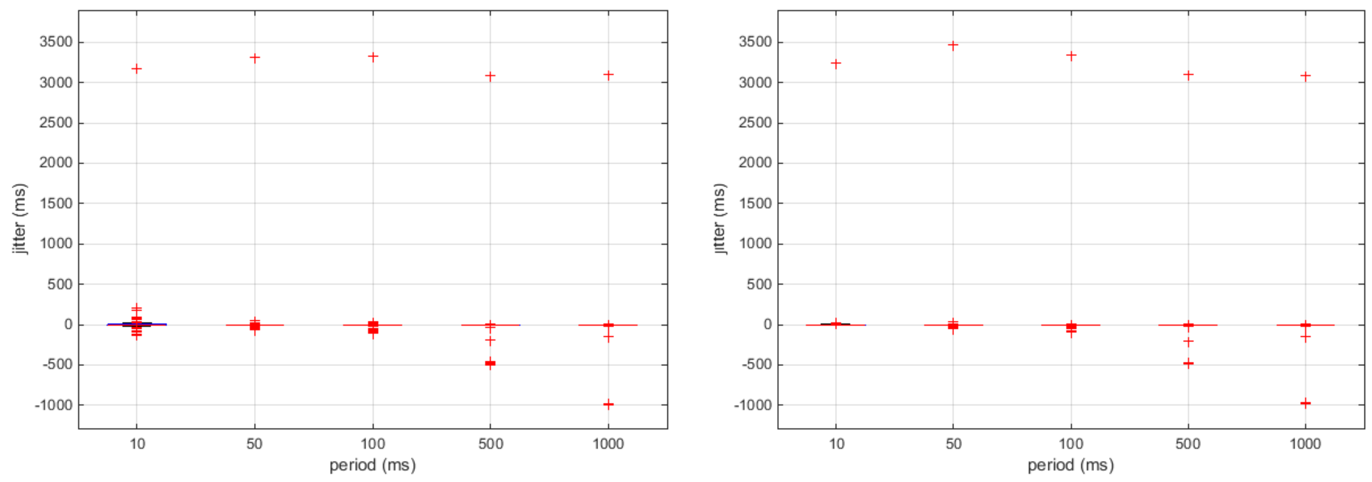
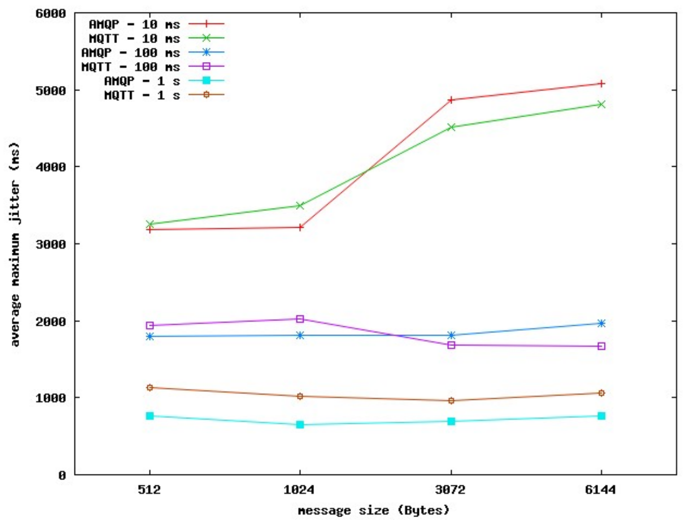
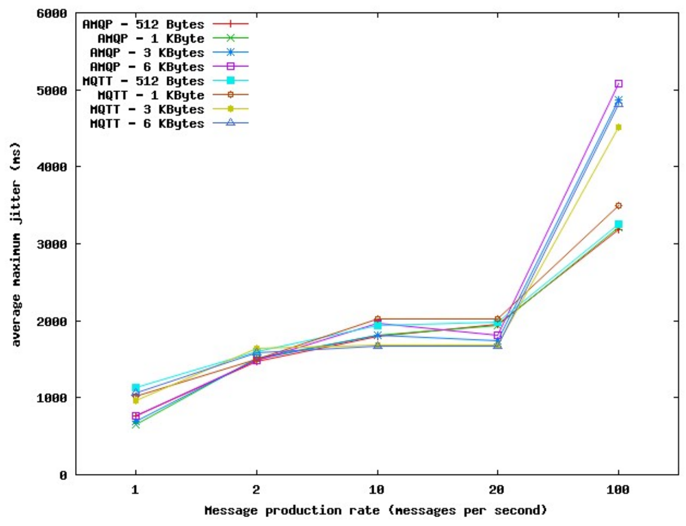


Fig. 3: Jitter behaviour on the producer migration access points, using (*left*) AMQP and (*right*) MQTT protocol



(a)



(b)

Fig. 5: Evolution of maximum jitter as a function of (a) message size, and (b) inter-message period

Typically a user application sends a few messages per second, with average load below 5 Mbps, which is well managed both by message protocols and the network. Performing this exhaustive delimitation of the workloads, in Figure 6 we show an approximation of the capacity of the system in terms of message size and number of messages produced per second.

The system is saturated, for loads above 20 Mbps, which is near to the bandwidth that we have obtained with the *iperf* tool for the TCP test in each case. If the load exceeds this limit value a certain proportion of all produced messages will not arrive to consumers.

Indeed, we note that the payload limit of a message in the MQTT protocol is greater than for AMQP. We consider that is mainly caused by the difference between the frame header: AMQP has a fixed size header of 8 *Bytes* while MQTT has only a 2 *Byte* header.

V. Conclusions and Future Work

In this paper, we presented an experimental analysis of the behaviour of two of the most relevant application level protocols based on Message-Oriented Middleware standard (AMQP and MQTT), focussed on the impact on the information exchange while a node is moving between different access points within the same domain in order to provide a characterization of the *publish-and-subscribe* models to scenarios where the collaborative use of the information collected by individual nodes will improve any situation of our day to day life, like in applications of physiological monitoring, inter-vehicular safety, remote sensing of environmental conditions,

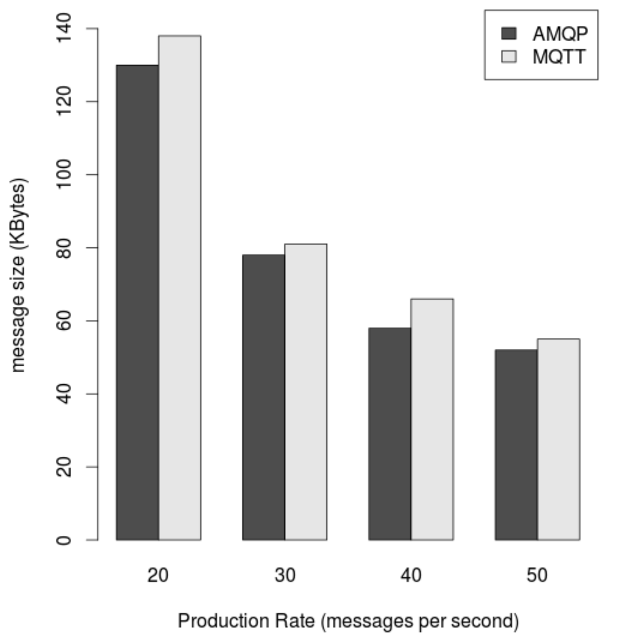


Fig. 6: Maximum message size before message losses start to appear increasing the inter message production period.

and transportation planning, analysis and monitoring.

With a simple workload model of one *producer/publisher* and one *consumer/subscriber* with several access points conforming a same Service Set, we have evaluated scenarios where the *producer/publisher* suffers a handover process measuring the effect in the variability of the jitter and the information loss, Indeed we have observed that both approaches of these protocols seem to perform very similarly with the mean jitter values that tend to oscillate between 1 and 4.5 *seconds*.

We have demonstrated that there is no information loss during the hand-off period and after the node recover its connection with the message-broker using the same IP address. We can say that the messaging system of these protocols is reliable and robust. The message losses are present only when load is higher than its system buffer capacity in the producer side.

We have found that after the attachment point, during message bursts the delivery with AMQP protocol follows a LIFO (last-input first-output) order, which results in messages consumed in inverted order. But this is not done MQTT protocol, with MQTT the delivery is always in order.

In order to select the right protocol to build systems and applications with mobile communications over node mobility scenarios, both of these protocols can be used. The application/system architect's decision to choose one of them, will be determined according to different criteria, such as security and energy efficiency. AMQP offers more aspects related to security [16], and MQTT is more energy efficient [6].

We recommend the use of AMQP protocol to build reliable, scalable, and advanced clustering messaging infrastructures over an ideal WLAN, and the use of MQTT protocol to support connections with edge nodes (simple sensors/actuators) under constrained environments (low-speed wireless access).

As future work, we plan develop techniques and strategies to reduce the problems arising from mobility, multi-homing and temporary disconnection of the nodes in IoT applications.

ACKNOWLEDGMENTS

This work was partially supported by the *Ministerio de Ciencia e Innovación*, Spain, under Grant TIN2011-27543-C03-01.

REFERENCES

[1] Li, Jun and Shvartzshnaider, Yan and Francisco, John Austen and Martin, Richard P. and Nagaraja, Kiran and Raychaudhuri, Dipankar, *Delivering internet-of-things services in MobilityFirst future internet architecture*, IEEE Proceedings of 2012 International Conference on the Internet of Things, IOT 2012, pp 31–38, 2012.

[2] Chen, Whei-Jen and Gupta, Rahul and Lampkin, Valerie and Robertson, Dale M. and Subrahmanyam, Nagesh, *Responsive Mobile User Experience Using MQTT and IBM MessageSight* IBM Corp., 2014.

[3] Gluhak, A. and Krco, S. and Nati, M. and Pfisterer, D. and Mitton, N. and Razafindralambo, T., *A survey on facilities for experimental internet of things research*, IEEE, Communications Magazine, pp. 58-67, 2009.

[4] Fernandes, J.L. and Lopes, I.C. and Rodrigues, J.J.P.C. and Ullah, S., *Performance evaluation of RESTful web services and AMQP protocol*, IEEE ICUFN, pp. 810–815, 2013.

[5] Kramer, Joshua *Advanced Message Queuing Protocol (AMQP)*, Linux Journal 187, Houston, TX, 2009.

[6] Lee, Shinho and Kim, Hyeonwoo and Hong, Dong Kweon and Ju, Hongtaek *Correlation analysis of MQTT loss and delay according to QoS level*, International Conference on Information Networking, pp. 714-717, 2013.

[7] Luzuriaga, J. and Perez, M. and Boronat, P. and Cano, J. C. and Calafate, C. and Manzoni, P. *Testing AMQP Protocol on Unstable and Mobile Networks*, Internet and Distributed Computing Systems, 7th International Conference, IDCS 2014, Italy, Proceedings Lecture Notes in Computer Science, pp. 250-260.

[8] O'Hara, John, *Toward a Commodity Enterprise Middleware*, ACM, Communications Magazine, 2007

[9] Peng Jiang; Bigham, J.; Bodanese, E.; Claudel, E., *Publish/subscribe delay-tolerant message-oriented middleware for resilient communication*, Communications Magazine, IEEE , vol.49, no.9, pp.124,130, September 2011.

[10] Subramoni, Hari and Marsh, Gregory and Narravula, Sundeep and Lai, Ping and Panda, Dhabaleswar K. *Design and evaluation of benchmarks for financial applications using advanced message queuing protocol (AMQP) over infiniband*, Workshop on High Performance Computational Finance, WHPCF 2008.

[11] Vinoski, S, *Advanced Message Queuing Protocol*, IEEE Internet Computing, vol.10, 2006.

[12] Cohn, Raphael *StormMQ: A Comparison of AMQP and MQTT*, Available: www.stormmq.com, 2011.

[13] OpenStack, Foundation *AMQP and Nova*, Available: http://docs.openstack.org/developer/nova/devref/rpc.html, 2014.

[14] IMatix, Corporation *Security and Robustness*, Available: http://zeromq.org/docs:welcome-from-amqp, 2014.

[15] Pivotal Software, Inc., *Messaging that just works*, Available: https://www.rabbitmq.com/reliability.html, 2014.

[16] OASIS, Standard *OASIS Advanced Message Queuing Protocol AMQP Version 1.0*, 2014.

[17] Eclipse, Org., *Paho is an iot.eclipse.org project*, Available: http://eclipse.org/paho/, 2014.

[18] MQTT.org., *MQTT faq website*, Available: http://mqtt.org/faq, 2014.