Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 4015625, 11 pages
http://dx.doi.org/10.1155/2016/4015625

*Research Article*

# Improving MQTT Data Delivery in Mobile Scenarios: Results from a Realistic Testbed

**Jorge E. Luzuriaga,[1] Miguel Perez,[2] Pablo Boronat,[2] Juan Carlos Cano,[1] Carlos Calafate,[1] and Pietro Manzoni[1]**

[1]*Department of Computer Engineering, Universitat Politècnica de València, Camí de Vera, s/n, 46022 València, Spain*
[2]*Universitat Jaume I, Castelló de la Plana, Avenida de Vicent Sos Baynat, s/n, 12071 Castelló, Spain*

Correspondence should be addressed to Jorge E. Luzuriaga; jorlu@upv.es

Received 5 February 2016; Accepted 11 July 2016

Academic Editor: Wenfeng Li



MQTT is being widely used for data delivery in IoT applications but its architecture does not properly handle mobility when disconnection periods tend to be large. In this paper we describe an experimental evaluation, made in a real environment, of a solution that guarantees that there is no information loss when variable length hand-offs appear due to the movement of a node. Our proposal modifies the classical publish/subscribe scheme by introducing an intermediate buffer that takes care of message transfer. Finally, we study the impact related to the connectivity of mobile devices of the use of the standard Linux *Network Manager*. We propose a cross-layer solution that improves the device connectivity in conjunction with the data layer management. We show that our solution improves the data delivery guaranteeing that no information is lost.

## 1. Introduction

The Internet revolutionised how people communicate and work together. It led the way to a new era of free information for everyone, transforming life in ways that were hard to imagine in its early phases. But the next wave is not about people, it is about intelligent, connected devices. To interact successfully with the real world, these devices must work together with speeds, scales, and capabilities far beyond what individual users may need. The Internet of Things (IoT) will change the world, perhaps more profoundly than today's human-centric Internet.

MQTT was developed in 1999 for the monitoring of an oil pipeline through the desert. The goals were to have a bandwidth-efficient protocol that used little battery power, because the devices were connected via satellite link and this was extremely expensive at that time [1, 2]. The protocol uses publish/subscribe architecture in contrast to HTTP with its request/response paradigm. Publish/Subscribe architecture is event driven and enables messages to be pushed to clients. The central communication point is the MQTT broker. It is in charge of dispatching all messages between the senders and the rightful receivers. This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers.

But MQTT was designed for fixed architectures and nowadays mobile devices are really the clear choice for our interactions with sensors and devices [3]. Smartphones diffusion rates in emerging and developing nations are rising at an extraordinary rate. Thus integrating mobile devices in a MQTT publish/subscribe architecture is a critical task and the one that we cover in this paper.

In this work we analyse, using a real scenario, the behaviour of a mobile smart device that is continuously sending information to servers while travelling through the coverage area of different access points. The solution we evaluate is capable of handling problems such as the expiration of TCP connections, the variable quality of network conditions, and applications misbehaviour due to changes of the assigned IP address. The proposed framework supports disconnected operation and tolerates spontaneous communications without data loss by caching messages to be sent and delivering them as soon as a path to the broker becomes available. The results show that our framework can support intermittent

connectivity using an asynchronous communication scheme very common in IoT networks.

The rest of the paper is organised as follows: a review of related work is offered in Section 2. We briefly describe the proposal in Section 3 and the methodology used to evaluate it in Section 4. In Section 5 we present the performance evaluation and result analysis. Eventually, some concluding remarks are available in Section 6.

## 2. Related Work

Acquiring data through remote sensing especially in the context of smart environment has been an important topic of research and development referred to and discussed concisely in [4–8].

Focussing on network layer provided mobility, a seamless handover for a hotspot network using a buffering technique is proposed in [9]. In this work the authors make a packet forwarding of the messages that during the handover of the mobile terminal are stored in the buffer placed on each access point. Thus, when a mobile terminal has moved from one access point to another the messages that were sent by the server to it are buffered in the first access point. After the handover the mobile terminal sends a movement notification packet to the new access point and it is resent to the rest of the access points where its tables are updated with the new address of the terminal. Then, all the packets that use the buffering technique are forwarding to the new access point and finally to the mobile terminal. From our point of view this approach has different drawbacks. In the first place it shows a problem of scalability since, with a high number of terminals moving around, the memory of each access point would be overload. Our approach alleviates this problem thanks to the buffer that is located on each mobile terminal. Secondly, the implementation of this proposal requires a modification of the drivers of the Linux bridge interface; in our proposal there is no need for such modifications.

In [10] the authors propose a link-layer packet forwarding scheme that can recover almost all packet loss during a hand-off. They implement "store and forwarding" and "hand-off detection" units in the device-driver of each access point, since all access points NICs have internal queues built in to hold many packets that have to be forwarded for each hand-off terminal after the reassociation. The store and forwarding unit is made to maintain an image queue in the driver to mirror the queue in the card. Thus every packet sent to the card has a copy in the driver and after a hand-off of a terminal the old AP will forward all packets including the packets of the driver buffer and the packets held in the image queue. The hand-off detection unit detects the hand-off process by a consecutive packet transmission failure and starts to buffer unsuccessfully transmitted packets for the hand-off terminal. Upon the reassociation of a hand-off terminal, the new AP sends out a broadcast update frame with the address of the terminal; thus all the devices update their routing table. The hand-off detection of the old AP unit also detects the hand-off completion when it checks the source address of the update frame and identifies if the terminal was once in its BSS. When the hand-off detection is complete the buffered packets are forwarded to the new destination of the terminal through the new AP.

In a lab testing environment [11] the performance of the protocols COAP, MQTT, and OPC UA is evaluated, measuring the transmission time of several messages with different length between two devices acting as a data-source and data-sink, respectively. Both devices are connected to the cellular and wired interfaces of the network simulator. This simulator supports the emulation of several radio technologies such as EDGE, UMTS, and LTE cellular network. They have found the following. [a] OPC UA has the lowest transmission time. [b] The protocols MQTT and OPC UA based on TCP achieve a better performance. [c] Reliable data exchange is not suitable for the transmission of large payloads over cellular networks. [d] In LTE the IP packets are concatenate in the same transport block until the transport block size is reached and then are sent as TCP frames.

Finally, the authors of [12, 13] consider similar issues also with smart objects-based IoT cyber-physical systems.

However, few studies propose an overall framework that increases the mobility support guaranteeing message delivery also improving the wireless connections quality, enabling the development of IoT applications where the mobility of the device is an issue without requiring network support through protocols like MobileIP or LISP [14] allowing that developers do not have to explicitly consider the changes in the point of attachment to the network.

## 3. Our Evaluated Proposal

Our proposal [15] maintains the *publish/subscribe* approach but decouples the pure data generation process by the data sending process by means of a technique called *intermediate buffering*. This decoupling allows for recovery when the communication channel presents disruption periods, even if they are very frequent and with length of various seconds, that is, in situations where TCP fails to recover.

We suppose having a message producer that continuously generates messages with a given frequency. A MQTT publisher takes the produced messages [16] and turns them it into MQTT messages, to be published with the same given periodicity to a predefined MQTT broker that will forward the incoming messages directly to the subscribers. A subscription is initially created by a client application on a predefined topic (simple subscription name). A basic diagram of the proposal can be seen in Figure 1.

When the connection between the publisher node and the message-broker suffers an interruption, the node enters in *roam mode*. The in-flight published messages (messages that have not received the acknowledgement from the message-broker) are stored in the MQTT internal buffer that is constrained with a very limited space available. These messages are delivered making a push diffusion [17] only when the node recovers the connection with the last access point (that means recover the last IP address); otherwise these messages are lost.

When longer disruption appears, our intermediate buffer takes charge of storing all the published messages that have not received the acknowledgement. Meanwhile, the MQTT network control mechanism manages the creation of the new

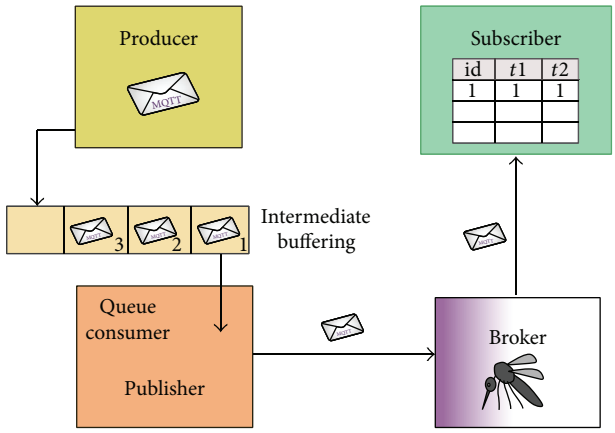


Figure 1: Diagram of the intermediate buffering proposal based on the publish/subscribe pattern of MQTT protocol.

connection and the correct closing of the aborted session. With the new connection, independently from the IP address that the node obtains, once the connection with the MQTT broker is reestablished, we can guarantee the delivery of all these messages in the same order that was published, followed by the messages which continue to be generated.

We evaluated our proposal using two software tools to manage the network connections. The first was the standard *Network Manager* version 1.0 [18] which is included in most of the Linux distributions and the second was our own tool called *signalBased Manager* (sBM) which we developed to allow faster handovers.

*3.1. Network Manager.* The Network Manager (NM) is an open source software project that enables the automatic configuration of the network interfaces of a Linux-based device as well as their network connections via the D-Bus interface. The NM consists of a system daemon that receives network settings from a pair of setting services by placing Connection-Objects on the system bus and a client application known as "nm-applet" that sends commands to the services to activate these connections. For the establishment of a connection with a wireless network, NM does an initial scan of available wireless networks if there is previously used network on the list connecting the device to it; otherwise it makes a selection based on an opportunistic approach attempting to use the best one.

When a device is moving around, the establishment of a connection using the wireless network is an issue to NM which causes that, most of the time during the displacements, the device remains in a disconnected state. If the device is moving in a network with several access points configured to offer a connectivity with the same service set identifier (ESSID), NM works fine connecting the device to the network because it tries to connect to the ESSID of the previous AP for three times and resumes the network connection successfully in a few seconds. But when the device is moving in a network with several access points configured with several ESSID NM does not work well. It attempts to reestablish the connection up to three times with a nonavailable access point; after this it tries to get a connection with a new detected network. This process takes around 5 minutes to get the device again online using a different service set identifier. Thus, we can say that the default behaviour of NM is not designed to support a device mobility between networks with different ESSID.

*3.2. The signalBased Manager.* We have developed this network which manages to improve the establishment of the wireless network connection of mobile users in highly dynamic scenarios. We observed that the standard Network Manager is not suitable for these environments since it decides to stay on a network even if the signal strength is very poor [19]. Moreover, when it gets to a total disconnection it keeps trying to reestablish the previous connection, even if it is not available in the current location.

We propose a mechanism that chooses in real time the best available radio based on signal strength measurements. This mechanism is included in the framework as a Network Manager tool called "*signalBased Manager*" (sBM) that support both handovers and hand-offs of a node moving around the coverage of different wireless networks especially to mesh or collaborative networks.

The mechanism is based on three phases: detection, discovery, and execution [20]. It starts the handover process on the client when its connection quality degrades to a predefined threshold (detection). It decides handover to a different AP based on the information of all the available access points in order to choose the best candidate (discovery); finally, the handover is completed with the client establishing a connection with the new access point (execution).

## 4. Methodology of the Experiments

This section presents the setup of the empirical evaluation we made. The indoor tests have been performed to study the behaviour of the MQTT protocol against an intermittent connectivity and the outdoor tests evaluated the behaviour of our proposal in a walking itinerary in Jaume I University Campus [21]. The walk imitated a common journey took by UJI students to reach the "*Espaitec center*" from the bus stop; these walks are hereafter referred to as AB and BA journeys. They are represented in Figure 2. All the outdoor experiments were performed while the *Guifi.net* nodes [22] were uniquely dedicated to be used with our generated traffic. In order to obtain a representative data set we have performed 32 tests with a duration of about 5 minutes each, generating four repetitions for each configuration and with each of the two Network Managers.

In general the traffic parameters used are a constant generation rate of messages fixed to 1 mps; a fixed publication periodicity of 1 second between each publication; two fixed messages size used of 512 Bytes and 6 Kbytes.

Our measurements were oriented to show different performance metrics calculated using the reception time-stamp of each message [23], that is, the maximum and mean disconnection times, the maximum amount of messages stored in the buffer, the amount of messages losses (packet loss rate), and the interdelivery regularity (jitter analysis).

Table 1: Summary of equipment used.

| Device | Role | Radio | Protocols | OS |
|---|---|---|---|---|
| CS-UJInuvolguifi1 | Mesh node | AR9582 | 802.11b/g/n | qMp 3.1 Clearance |
| | Access point | AR9341 | 802.11a/n | |
| CS-UJInuvolguifi2 | Mesh node | Nanostation M5 | 802.11a/n | qMp 3.1 Clearance |
| | Access point | AR9341 | 802.11b/g/n | |
| CS-UJInuvolguifi3 | Mesh node | AR9582 | 802.11a/n | qMp 3.1 Clearance |
| | Access point | Nanostation M2 | 802.11b/g/n | Airos (xs2.ar2316) |
| CS-UJInuvolguifi4 | Mesh node | Nanostation M5 | 802.11a/n | qMp 3.1 Clearance |
| | Access point | Nanostation M2 | 802.11b/g/n | Airos (xs2.ar2316) |
| CS-UJInuvolguifi5 | Mesh node | Nanostation M5 | 802.11a/n | qMp 3.1 Clearance |
| | Access point | Nanostation M2 | 802.11b/g/n | Airos (xs2.ar2316) |
| Samsung NC10 | Client | Qualcomm Atheros | 802.11b/g | Ubuntu 14.04 |
| Desktop computer | Server | 100 Base ethernet | 802.3 | Ubuntu 14.04 |

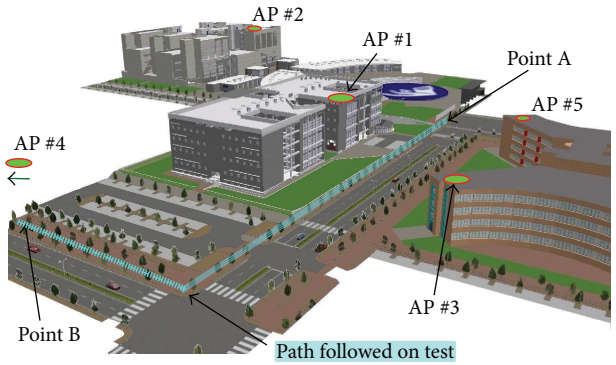


Figure 2: Testing scenario inside Jaume I University.

*4.1. Experimental Scenario.* Inside University Jaume I there is deployed a part of the wireless infrastructure of *Guifi.net* Community Network [22]. To ensure that the clients (students and staff) can roam smoothly, multiple nodes have been installed outdoors on the terraces of the principal buildings for a full coverage throughout the entire University Campus. In Figure 2 we can appreciate a scenario representation; more specifically we used the nodes *CS-UJInuvolguifi* (from 1 up to 5) [24]. The clients use IEEE802.11n links at 2.4 GHz, while fixed nodes are interconnected as a mesh at 5 GHz. A list of equipment types used is in Table 1.

These nodes are based on antennas and integrated routers such as Ubiquiti or tp-link running an open source community distribution based on the OpenWRT Linux distribution [25].

In these scenarios, we have travelled on a bidirectional pedestrian walkway that is around 500 m long, carrying a laptop trying to keep a constant pace of about 6 kph. The forward and return paths walkway are coincident but as we will see in the coming sections the connection establishment order with the access points and the behaviour related to the message delivery were quite different.

The duration of each test was 5 minutes, that is, the time necessary to move between these points.

Table 2: Summary of test setup parameters.

| Parameter | Value(s) |
|---|---|
| Generation rate | 1 mps |
| Publication rate | 1 mps |
| Net manager | GNU Network Manager (NM), *signalBased Manager* (sBM) |
| Message size | 512 Bytes, 6 Kbytes |
| Trips | from A to B, from B to A |
| Walkway length | ~500 m |
| Velocity | ~6 kph |

During this period of time our mobile device generates several MQTT messages with different payload sizes. These messages were sent to the broker and then were delivered to the subscribers. Table 2 shows the parameters setup for MQTT measurements on the proposed system.

## 5. Evaluation and Results

This section includes the performance evaluation of the proposed framework and shows results obtained through different experiments on the field followed by a statistical analysis. The measured results are presented for the MQTT protocol with and without the proposed prebuffering technique and with and without the Network Manager improvement.

*5.1. Using the MQTT Protocol without Prebuffering.* To obtain some reference value about the behaviour of MQTT, in this section we evaluate its performance both in an outdoor environment without moving the devices ("static case") and over intermittent connections ("with disconnections case") to illustrate the weaknesses of the MQTT protocol. We simulated an unstable environment with disconnections using an indoor testbed where the mobility of a device was obtained turning off and on the wireless radio at intervals that were varied between 1, 5, and 30 seconds.

Table 3: Static case, without mobility.

| Size (Bytes) | Mean (ms) | St dev | Min (ms) | Max (ms) |
|---|---|---|---|---|
| 512 | 0.926 | 30.454 | −250 | 240 |
| 6144 | 1.416 | 46.456 | −344 | 301 |

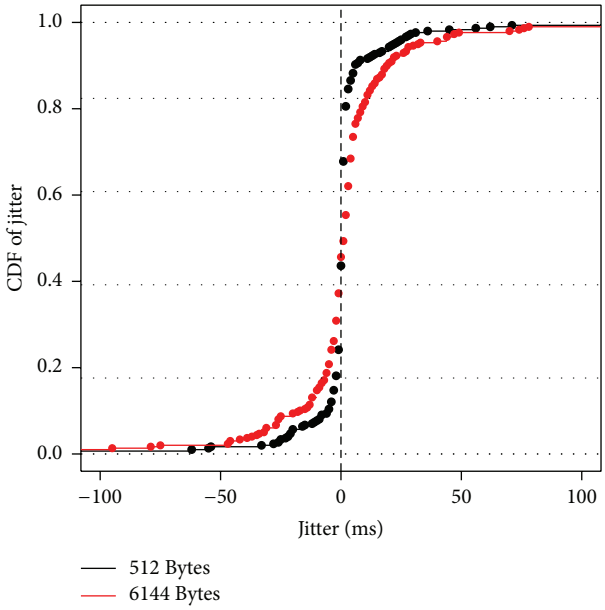


Figure 3: Static case.

*5.1.1. Static Case.* A set of tests were executed without any movement of the mobile devices. Specifically, we have executed the test ensuring a direct line-of-sight (LOS) link between the access point AP #5 and the mobile devices placed on a fixed spot of the central boulevard of the UJI, while the mobile device is publishing our testing messages.

Figure 3 shows the CDF of the jitter for each of the two message sizes. The jitter is basically very reduced and similar independently from the size of the messages. In Table 3, we can see that the mean values of the jitter with both message sizes are around 1 ms.

*5.1.2. With Disconnections Case.* To control disconnection periods we built a testbed in the laboratory that allowed us to control most of the factors that may influence an experiment. We have tested several parameter combinations, for instance, other message sizes, higher publishing frequencies (100 and 500 ms), and different on-off periods using the values of 1, 5, and 30 seconds.

With an off period of 30 s, Figure 4, we can see that slightly less than 50% of the cases have a negative values and less than 95% of the cases have values smaller than 100 ms. The same behaviour is observed with both message sizes used and even with messages of a smaller size (120 Bytes) as we can see from Table 4. We notice that lower standard deviation values are obtained with a higher messages sending frequency (100 ms).

We also evaluated the message loss values considering a message loss when a message sent by a publisher was not delivered to a subscriber or not even received by the broker. We used the MQTT quality of service set to "at least once" where the protocol ensures that a message arrives at the server at least once. When a message is published, a copy is stored in the publisher internal buffer until the reception of the ACK packet. When the acknowledgement is received it indicates a successful delivery and the copy of the message is discarded from the buffer. By default this internal buffer has been defined as the maximum number of in-flight messages to 10. Once this value is reached the buffer will overflow and all the outstanding MQTT messages sent to the broker will be lost. The reduced space available on the buffer allows that only a few messages can be stored. This is a problem in high data traffic environments where this value can be reached easily and quickly. In addition, the content of the buffer is delivered only if the MQTT session is active and only if the client maintains the same id, a problem that appears when reestablishing broken TCP/IP connections. Table 5 shows statistical information about the number of lost messages with different disconnection periods.

Table 4: Jitters obtained with a wireless radio off time of 30 seconds.

| Size (B) | Period (ms) | Mean (ms) | St dev | Min (ms) | Max (ms) |
|---|---|---|---|---|---|
| 1024 | 100 | 307.17 | 4,293.83 | −100 | 62,353 |
| 1024 | 500 | 1,318.85 | 8,218.52 | −465 | 51,930 |
| 1024 | 1000 | 2,691.85 | 11,871.21 | −561 | 56,385 |
| 512 | 100 | 287.62 | 4,206.89 | −100 | 63,636 |
| 512 | 500 | 988.59 | 7,568.70 | −500 | 56,204 |
| 512 | 1000 | 1,542.55 | 9,953.38 | −1,000 | 56,437 |
| 120 | 100 | 276.80 | 4,054.28 | −100 | 62,601 |

Table 5: Number of message losses.

| Radio turnoff time (sec.) | Median | Mean | St dev | Min | Max |
|---|---|---|---|---|---|
| 1 | 125.5 | 116.35 | 24.68 | 60 | 154 |
| 5 | 125.0 | 125.82 | 3.98 | 121 | 134 |
| 30 | 521.0 | 530.40 | 19.38 | 511 | 560 |

As can be seen, even with not so high disconnection time (30 sec) the number of messages lost increases in a clear manner, thus reinforcing the need of an improvement of the architecture.

*5.2. Using the MQTT Protocol with Buffering.* In this subsection we present the results with our buffering proposal. First, we show the results using the default connection manager (i.e., Network Manager) which is integrated in most of the Linux distributions. Secondly, the results with our own connection manager are shown where we can observe an improvement in the overall results.

We focus our attention on the situations where the devices get disconnection, where we have identified the percentage of time among several reasons like (a) loss of coverage situations [26], (b) association to an access point without an IP address, or (c) keeping an old IP address assigned by the previous

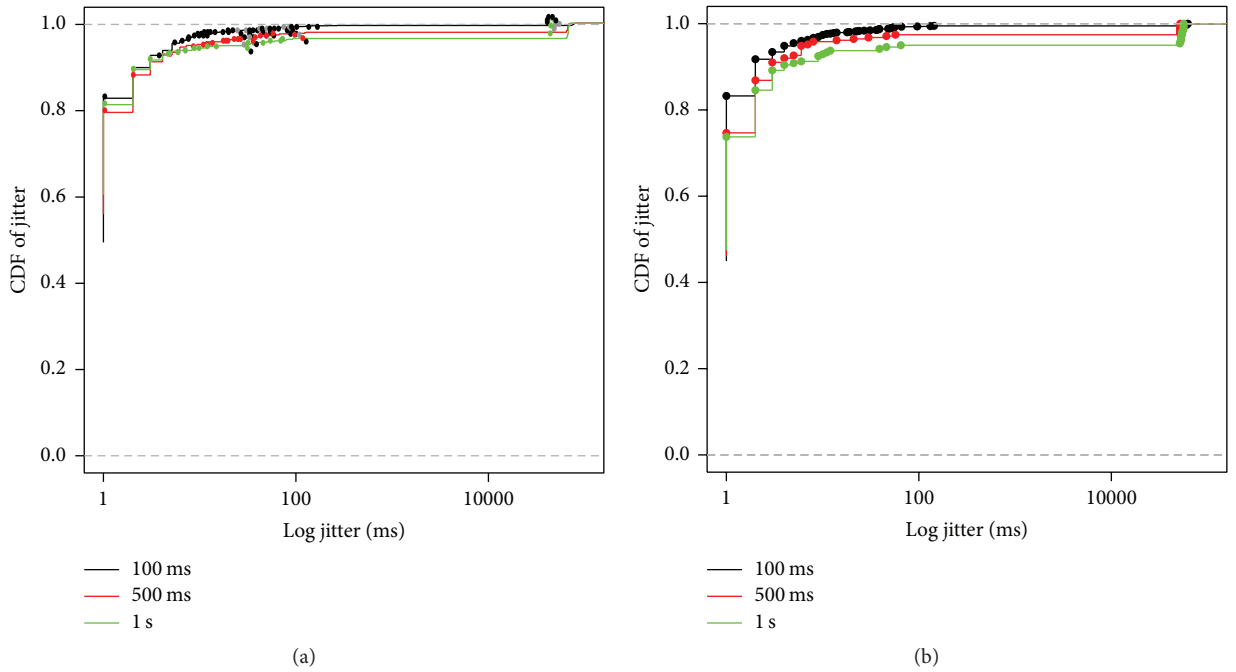


Figure 4: The CDF of jitter with a wireless radio off time of 30 sec. and message size of 512 Bytes (a) and 1024 Bytes (b).

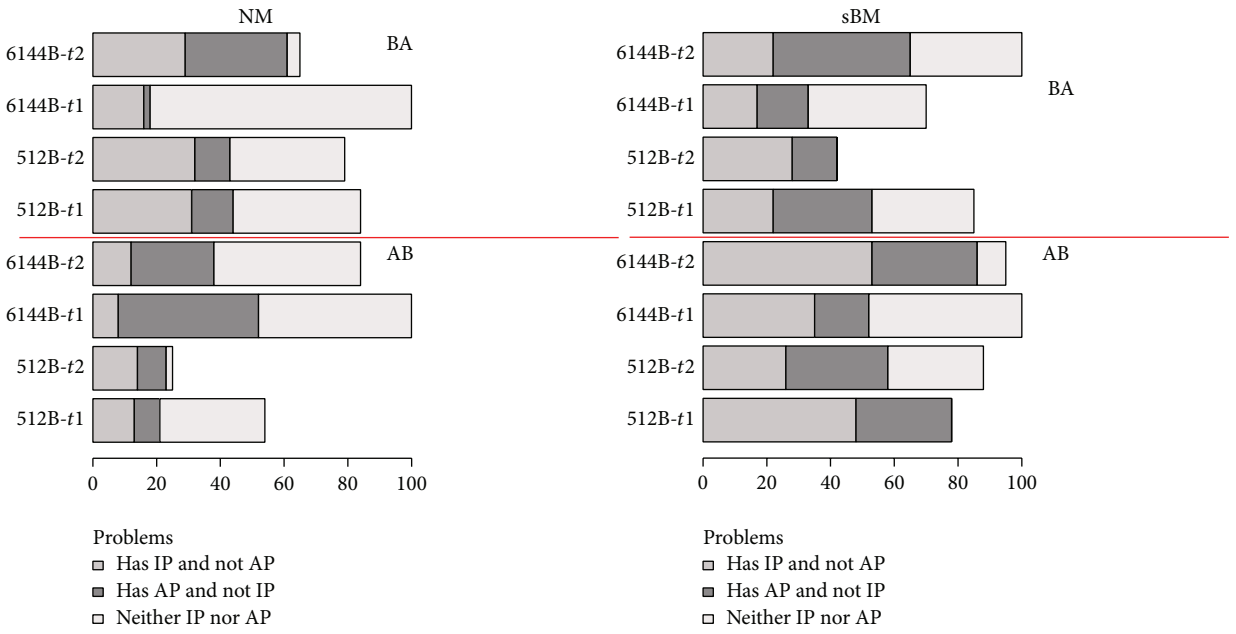


Figure 5: Connection problems with NM and sBM.

access point without getting a successful association with the new access point.

Figure 5 shows the percentage of time that the device had problems transmitting MQTT messages due to the connection breakdown during the execution of the test.

The size of each bar is proportional to the disconnection time presented by the device in each test. When NM is used the principal problem is the total isolation of the device, because the device most of the time was trying to connect to an unexcited network. While sBM is used the principal problem is related with the DHCP time out to the IP assignation.

We can say that the behaviour of the connection is very variable even when the same trajectory is repeated in another time, due an innumerable number of reasons making the results very dependent on a particular test.

Table 6: Jitter values using our buffering proposal with the standard Linux Network Manager.

| Journey | Size (Bytes) | Median (ms) | Mean (ms) | St dev | Max (ms) |
|---|---|---|---|---|---|
| AB | 512 | 201 | 1,063.82 | 6,320.82 | 102,413 |
| AB | 6 K | 338.5 | 1,552.02 | 14,216.05 | 290,851 |
| BA | 512 | 865 | 1,511.91 | 17,252.24 | 481,502 |
| BA | 6 K | 865 | 1,401.62 | 11,609.68 | 285,475 |

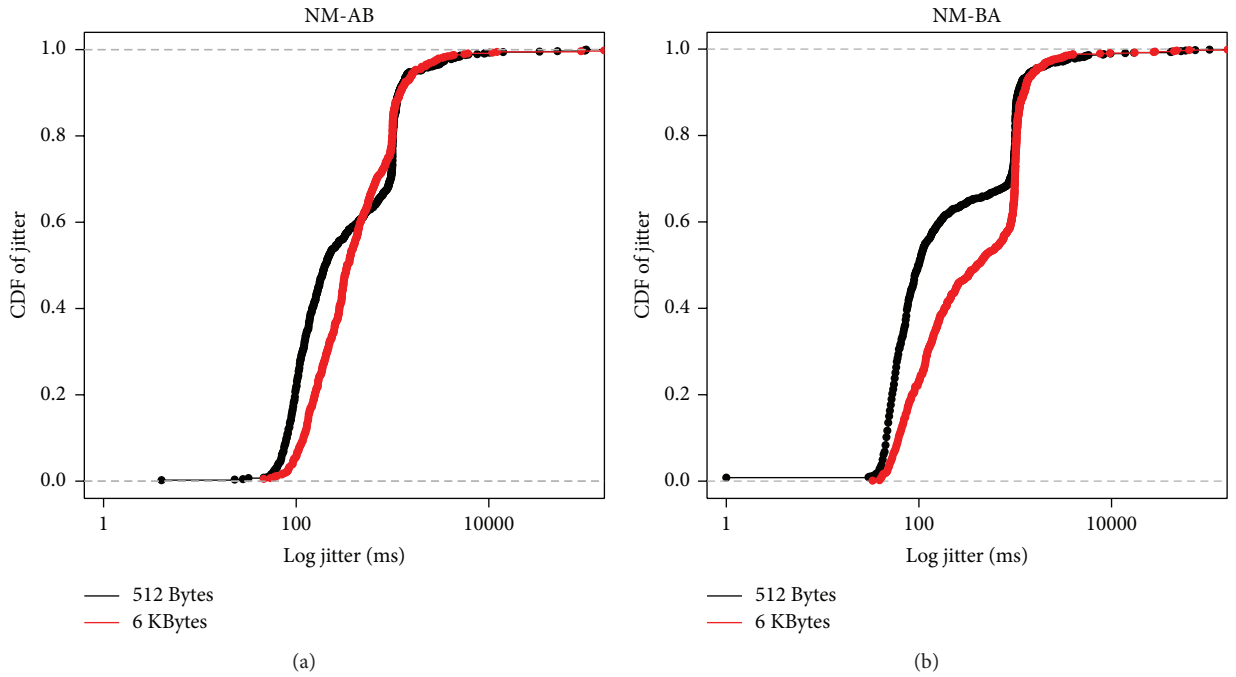


Figure 6: CDF of jitter in logarithmic scale using our buffering proposal, sending messages of two sizes 512 Bytes and 6 Kbytes on two journeys (a) AB and (b) BA trip, respectively.

*5.2.1. Using the Standard Network Manager.* In these tests we used the standard Linux Network Manager (NM) service. As we said before, the NM has a problem when the devices are moving around and getting out of a coverage area, since, for example, it tries to recover the connection with the old access point, making up to three repetitions to reestablish the previous connection. While this could be reasonable for a static user, it leads to bad performance for a mobile user.

As we can see from Figure 6, in both plots the small messages have a smaller jitter value. The 80% of probability of the values is around 1 second that basically corresponds to the sending rate. Then the biggest values in the graph are the disconnection times of each test.

In Table 6, we can see that in the AB journey with the biggest message size values of 290 seconds are reached. That means that almost all of the trip the client was disconnected. In the BA journey, we observe a chaotic situation in the network establishment during the walking test independently of the message size used; that is, there are cases where the client was disconnected along all the journey, and all the messages were stored in the buffer and transmitted uniquely at the arrival end point.

*5.2.2. Using Our signalBased Manager.* In the following tests we have used our proposal of Network Manager that tries to get the best available connection with an access point with a client moving through different coverage areas during data transmission.

Table 7 summarizes the obtained results, where we can see that standard deviation values are lower than the obtained ones with the standard Network Manager; indeed the maximum disconnection values are between 103 and 133 seconds and in the worst case a value of 256 seconds was obtained; that means that during the walking test 34 up to 44% of the time the device was disconnected, and in the worst case up to 85% was reached but never a total isolation as occurs with the standard manager.

Figure 7 shows that the biggest message has more regular jitter values than the smallest one, especially from 60 to 80%, that is, the median values. It is also evident that from 74 up to 86% of the jitter, values are close to 1 second.

To a better comparison between two managers, we have joined the results obtained of the two message sizes and then subtracted the generation periodicity for each message. By comparing these results in Figure 8, we can see that slightly

Table 7: Jitter values using the protocol with the buffer using *signalBased* Manager.

| Journey | Size (Bytes) | Median (ms) | Mean (ms) | St dev | Max (ms) |
|---|---|---|---|---|---|
| AB | 512 | 386.5 | 1,027.88 | 5,061.39 | 102,730 |
| AB | 6 K | 249 | 1,016.23 | 6,592.85 | 122,835 |
| BA | 512 | 1,154 | 1,105.68 | 9,158.91 | 255,958 |
| BA | 6 K | 576 | 1,189.65 | 8,472.11 | 133,159 |

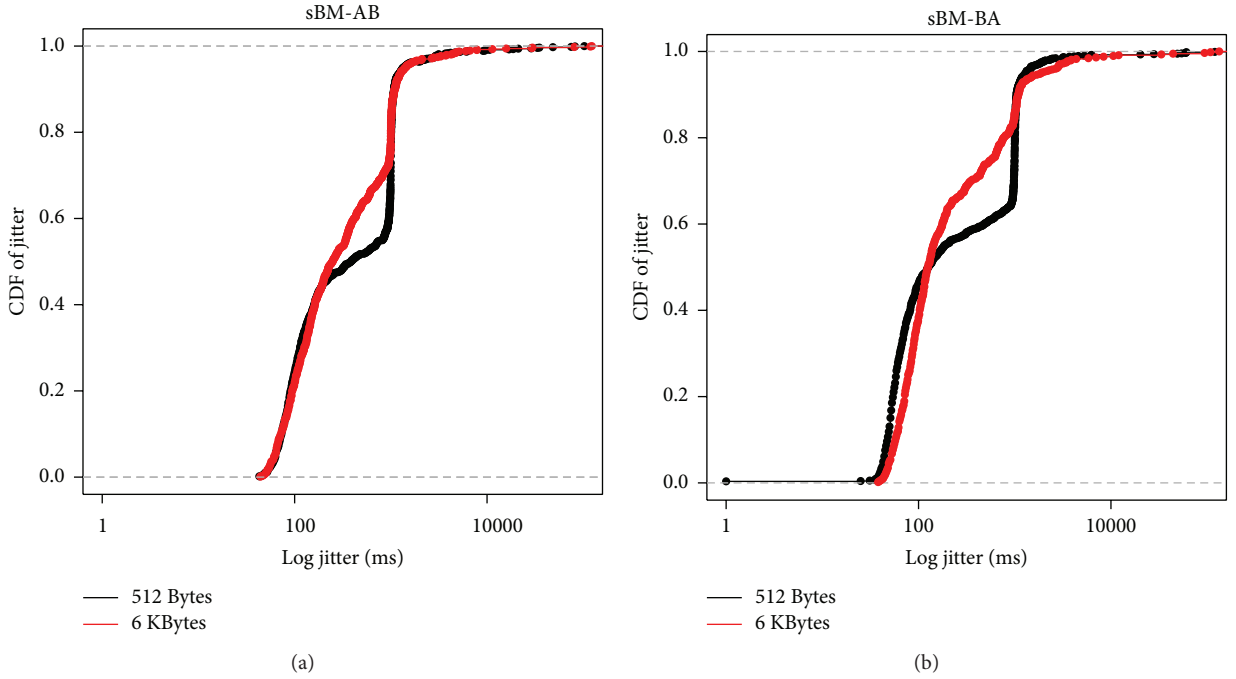


Figure 7: CDF of jitter in logarithmic scale using the buffer proposal sending messages of two sizes 512 Bytes and 6 Kbytes on two journeys (a) AB and (b) BA trip, respectively.

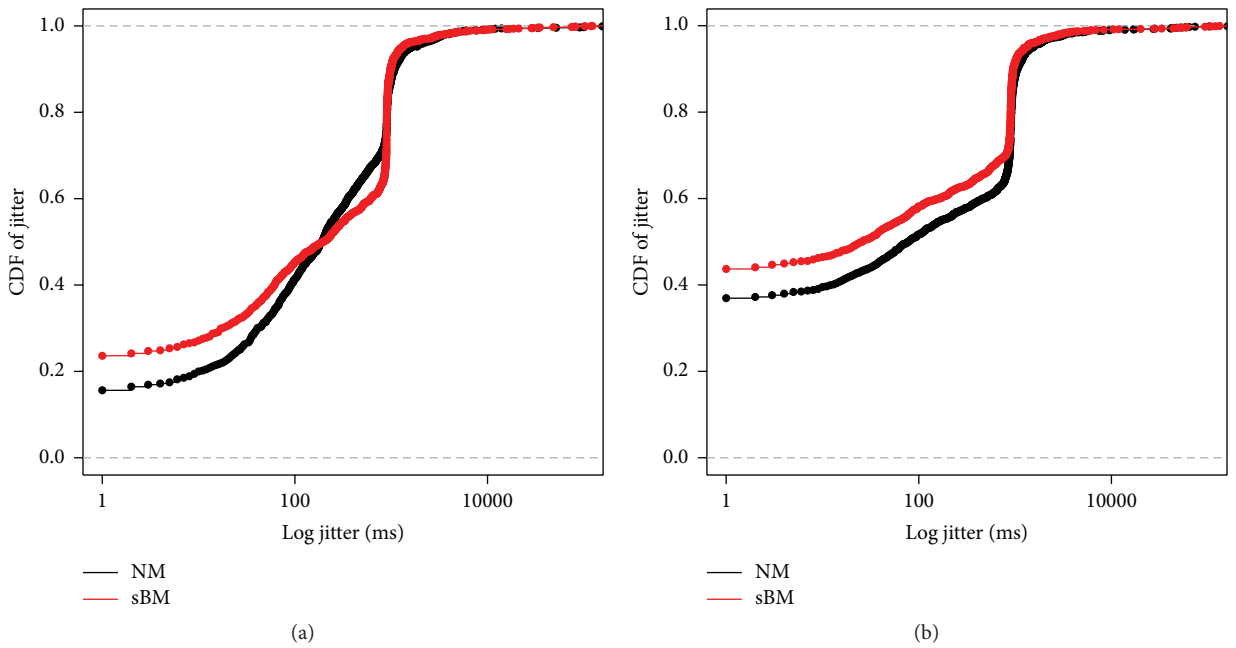


Figure 8: CDF of jitter in logarithmic scale using the buffer in (a) AB trip and (b) BA trip.

Table 8: Network manager comparative.

| Journey | Manager | Median (ms) | Mean (ms) | St dev | Max (ms) |
|---|---|---|---|---|---|
| AB | NM | 192 | 1,158.96 | 10,233.62 | 290,751 |
| AB | sBM | 197 | 922.89 | 5,765.98 | 122,735 |
| BA | NM | 79 | 1,356.77 | 14,700.01 | 481,402 |
| BA | sBM | 26.50 | 1,033.64 | 8,933.74 | 255,858 |

Table 9: Disconnection times in seconds.

| Journey | Manager | Median (s) | Mean (s) | Min (s) | Max (s) |
|---|---|---|---|---|---|
| AB | NM | 113.5 | 97.0 | 35.0 | 126.0 |
| AB | sBM | 60.50 | 64.00 | 49.00 | 86.00 |
| BA | NM | 49.00 | 56.25 | 32.00 | 95.00 |
| BA | sBM | 65.50 | 58.25 | 35.00 | 67.00 |

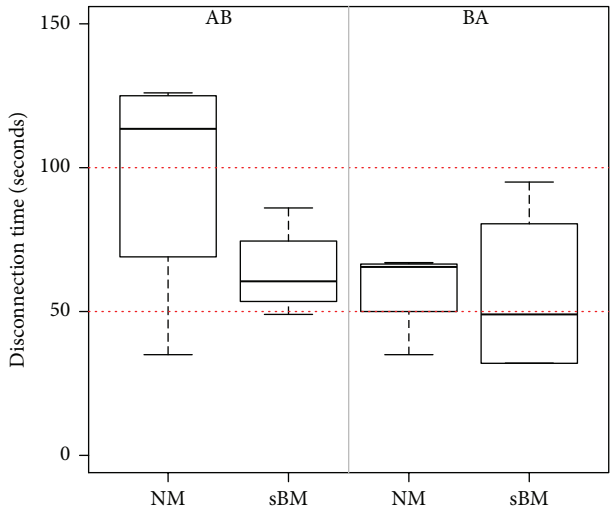


Figure 9: Disconnection times.

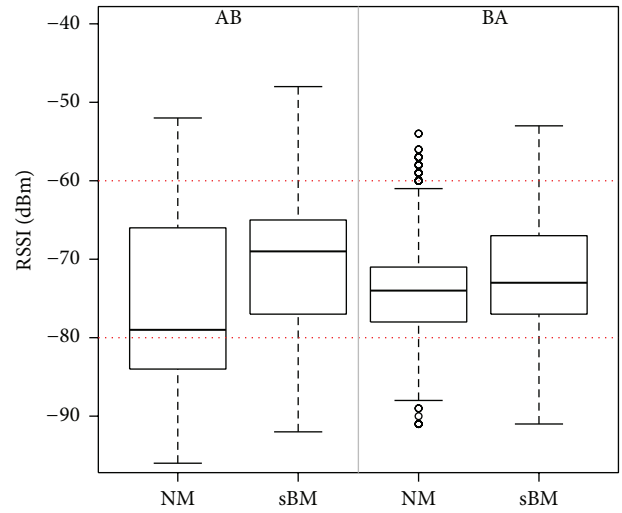


Figure 10: RSSI by journey and manager.

less than 20 and 40% in AB and BA journeys, respectively, have jitter values around zero, while more probability to get a less jitter is obtained using the *signalBased Manager*.

From this test, we know that the buffer proposal to the MQTT protocol in the client is working successfully with a high production rate in regard to IoT applications with the target devices in the real networking environments. Table 8 provides a comparison of both Network Managers. With these results we see that the maximum jitter value obtained in both journeys is lower with sBM than with NM.

Finally, we present how long the disconnection periods were during the execution of each testbed and the variation of their RSSI values in each one.

Regarding the maximum value of the network disconnection time of a device in a mobility case, we observed that it is highly correlated with the jitter measurements. As we can see in Table 9 the disconnection times are from the 32 seconds up to a maximum of 126 seconds. Figure 9 illustrates the disconnection times in the test with each manager where we confirm that the minimum disconnection values are obtained using *signalBased Manager*.

Regarding the RSSI observed with a user moving with a mobile device in outdoor environment during the tests, we observe that the received radio signals in general terms are weak due to various factors such as distance, mobility, and obstacles. We can see in Figure 10 that most of the values of the measurements of the power of the received radio signal are between −84 and −66 dBm. Specifically in Table 10 the mean values obtained are close to −70 and −75 dBm, with values of 35 dBm for really good connections and in the worst case a connection with a value of 126 dBm.

## 6. Conclusions

The Internet of Things (IoT) is already connecting computing devices, appliances, humans, and other living beings through the Internet. Accumulating data and knowledge through these things would improve a vast array of items and experiences throughout the world. The IoT is made of events and signals of many different kinds and require a standardised mode of communication. MQTT is IoT connectivity protocol so lightweight that it can be supported by some of the smallest

Table 10: RSSI values in decibel-milliwatts (dBm).

| Journey | Manager | Median | Mean | St dev | Min | Max |
|---|---|---|---|---|---|---|
| AB | NM | −79 | −75.75 | 10.06 | −96 | −52 |
| AB | sBM | −69 | −70.36 | 8.04 | −92 | −48 |
| BA | NM | −74 | −74.14 | 6.02 | −91 | −54 |
| BA | sBM | −73 | −71.94 | 7.21 | −91 | −53 |

measuring and monitoring devices, and it can transmit data over far reaching sometimes intermittent networks. Its architecture anyway does not properly handle mobility when disconnection periods tend to be large.

In this paper we described an experimental evaluation, made in a real environment, of a solution that guarantees that no information loss appears in the presence of variable length hand-offs due to the nodes movement. Our proposal modifies the classical publish/subscribe scheme by introducing an intermediate buffer that takes care of message transfer.

We showed that our solution allows the development of IoT applications where the mobility of the device is no more an issue. Developers do not have to explicitly consider the changes in the point of attachment to the network and no network support is required through protocols like MobileIP or LISP.

Moreover, we studied the impact related to the connectivity of mobile devices of the use of the standard Linux Network Manager. We proposed a cross-layer solution that improves the device connectivity in conjunction with the data layer management. From the experimental results related to the connection managers, the *signalBased Manager* is verified to show a slightly better performance compared to the standard Linux Network Manager, due to the fact that improves the RSSI and extends the connections' duration.

## Competing Interests

The authors declare that they have no competing interests.

## Acknowledgments

This work was partially supported by *Ministerio de Economía y Competitividad, Programa Estatal de Investigación, Desarrollo e Innovación Orientada a los Retos de la Sociedad, Proyectos I+D+I 2014*, Spain, under Grant TEC2014-52690-R.

## References

[1] W. Chen, R. Gupta, V. Lampkin, D. Robertson, and N. Subrahmanyam, *Responsive Mobile User Experience Using MQTT and IBM MessageSight*, IBM Corp, 2014.

[2] G. Owojaiye and Y. Sun, "Focal design issues affecting the deployment of wireless sensor networks for pipeline monitoring," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1237–1253, 2013.

[3] Network Working Group, *DNS Name Auto-Configuration for Internet of Things Devices*, 2016, https://tools.ietf.org/html/draft-jeong-homenet-device-name-autoconf-03.

[4] J.-A. Jiang, C.-L. Tseng, F.-M. Lu et al., "A GSM-based remote wireless automatic monitoring system for field information: a case study for ecological monitoring of the oriental fruit fly, *Bactrocera dorsalis* (Hendel)," *Computers and Electronics in Agriculture*, vol. 62, no. 2, pp. 243–259, 2008.

[5] M. Di Francesco, S. K. Das, and G. Anastasi, "Data collection in wireless sensor networks with mobile elements: a survey," *ACM Transactions on Sensor Networks*, vol. 8, no. 1, article 7, 2011.

[6] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 14, no. 2, pp. 70–87, 2007.

[7] Q. Gao, J. Zhang, X. Shen, and B. Larish, "A cross-layer optimization approach for energy efficient wireless sensor networks: coalition-aided data aggregation, cooperative communication, and energy balancing," *Advances in Multimedia*, vol. 2007, Article ID 56592, 12 pages, 2007.

[8] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: a survey," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.

[9] F. Yamagata, D. Komaba, H. Oguma et al., "Seamless handover for hotspot network using buffered packet forwarding method," in *Proceedings of the 10th IEEE Singapore International Conference on Communications Systems (ICCS '06)*, pp. 1–5, Singapore, November 2006.

[10] M. Portolés, Z. Zhong, S. Choi, and C.-T. Chou, "IEEE 802.11 link-layer forwarding for smooth handoff," in *Proceedings of the 14th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '03)*, vol. 2, pp. 1420–1424, Beijing, China, September 2003.

[11] L. Dürkop, B. Czybik, and J. Jasperneite, "Performance evaluation of M2M protocols over cellular networks in a lab environment," in *Proceedings of the 2015 18th International Conference on Intelligence in Next Generation Networks (ICIN '15)*, pp. 70–75, IEEE, Paris, France, February 2015.

[12] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, "Middlewares for smart objects and smart environments: overview and comparison," in *Internet of Things Based on Smart Objects: Technology, Middleware and Applications*, pp. 1–27, Springer, Berlin, Germany, 2014.

[13] G. Fortino, A. Guerrieri, M. Lacopo, M. Lucia, and W. Russo, "An agent-based mid-dleware for cooperating smart objects," in *Highlights on Practical Applications of Agents and Multi-Agent Systems: International Workshops of PAAMS 2013, Salamanca, Spain, May 22–24, 2013. Proceedings*, pp. 387–398, Springer, Berlin, Germany, 2013.

[14] T. Taleb and A. Ksentini, "Follow Me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013.

[15] J. Luzuriaga, J. C. Cano, C. Calafate, P. Manzoni, M. Perez, and P. Boronat, "Handling mobility in IoT applications using the MQTT protocol," in *Proceedings of the Internet Technologies and*

*Applications (ITA '15)*, pp. 245–250, Wrexham, UK, September 2015.

[16] Y. Gu, F. Ren, Y. Ji, and J. Li, "The evolution of sink mobility management in wireless sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 507–524, 2016.

[17] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: a survey," *IEEE Communications Surveys and Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.

[18] The GNOMEProject, NetworkManager, 2016, https://wiki.gnome.org/Projects/NetworkManager.

[19] P. Fazio, M. Tropea, C. Sottile, and A. Lupia, "Vehicular networking and channel modeling: a new Markovian approach," in *Proceedings of the 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC '15)*, pp. 702–707, IEEE, Las Vegas, Nev, USA, January 2015.

[20] H. Velayos and G. Karlsson, "Techniques to reduce the IEEE 802.11b handoff time," in *Proceedings of the 2004 IEEE International Conference on Communications*, vol. 7, pp. 3844–3848, June 2004.

[21] UJI International, Universitat Jaume I of Castello, http://ujiapps.uji.es/perfils/internacional/.

[22] Guifi.net, "Telecommunications Networks as a Commons," http://guifi.net/en/what_is_guifinet.

[23] J. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni, "Testing AMQP protocol on unstable and mobile networks," in *Proceedings of the 7th International Conference on Internet and Distributed Computing Systems (IDCS '14), Calabria, Italy, September 2014*, Lecture Notes in Computer Science, pp. 250–260, Springer, 2014.

[24] Guifi.net, CastelloUJISolicom, http://guifi.net/en/uji-biblioteca.

[25] D. Vega, L. Cerdà-Alabern, L. Navarro, and R. Meseguer, "Topology patterns of a community network: Guifi.net," in *Proceedings of the IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob '12)*, pp. 612–619, IEEE, Barcelona, Spain, October 2012.

[26] B. Wang, "Coverage problems in sensor networks: a survey," *ACM Computing Surveys*, vol. 43, no. 4, article 32, pp. 1–53, 2011.