A Disruption Tolerant Architecture based on MQTT for IoT Applications

Jorge E. Luzuriaga*, Marco Zennaro†, Juan Carlos Cano*, Carlos Calafate* and Pietro Manzoni*
* Department of Computer Engineering
Universitat Politècnica de València, SPAIN
† International Centre for Theoretical Physics / ICT4D Lab., ITALY
Email: jorlu@upv.es, mzennaro@ictp.it, {jucano, calafate, pmanzoni}@disca.upv.es

***Abstract*—In the IoT world, establishing a strong mobile network architecture will be critical for organizations to bring together people, processes, data and things. Among the various available protocols and standards to network IoT entities, the Message Queue Telemetric Transport (MQTT) is already a reference solution. It provides a publish/subscribe messaging transport specifically designed to be used in devices with limited resources over constrained networks. MQTT's main limitation is its low resilience with respect to device mobility, so that the connections could suffer frequent and long lasting disruptions or high bit error rates that severely degrade normal communications. In this work we propose an architecture to increase the robustness of MQTT by integrating a Disruption Tolerant Network (DTN) approach. The architecture has been evaluated through several experiments using real devices to validate its feasibility, and to derive some guidelines for its use.**

I. Introduction

Nowadays, more and more smart objects are being incorporated into the Internet. Their management, the traffic that they generate, as well as the effective use of these data are challenges that developers, researchers and enthusiasts of the Internet of Things and wireless sensor networks technologies are facing.

MQTT (Message Queuing Telemetry Transport) is a light publisher/subscriber protocol specifically designed for IoT applications and machine-to-machine communications. It is an open protocol optimized for communication over networks with limited bandwidth [1]. It is also an appropriate choice when dealing with simple, small, low-cost devices with limited resources. Together with CoAP (Constrained Application Protocol) it is one of the most relevant IoT messaging protocols. MQTT works on top of the TCP protocol stack but this stack is too complex for wireless sensors and actuators [1] that basically transfer small data pieces such as measurements, remote commands, or user data. Thus in 2008, Stanford-Clark A. and H. Linh Truong both from IBM published the MQTT for Sensor Networks (MQTT-SN) specifications [2] where TCP is replaced by UDP.

Anyway MQTT, does not behave well in mobile scenarios when sensing devices face periods of disconnection followed by a re-connection (possibly on a different sub-network). This issue may be detrimental to the wide adoption of this standard.

Dealing with connections that suffer frequent and long lasting disruptions and high bit error rates that may severely degrade normal communications is anyway already an active research area. To this end, the so called "Disruption Tolerant Networks" (DTN) paradigm was introduced to provide a general communication architecture in the presence of intermittent connectivity over a wide range of networks [3]. The DTN architecture defines the "Bundle Protocol" (BP) and a "Bundle layer" which operate in-between application and transport layers by exchanging blocks of data (the "bundles") as the basic information unit of variable length.

In this work we focused on these problems and proposed and analyzed the behavior of an MQTT-based architecture that relies on the Disruption Tolerant Network (DTN) paradigm. The overall goal was to increase the flexibility of the system with respect to connectivity. More precisely, we used MQTT for Sensor Networks (MQTT-SN) combined with the IBR-DTN[1] implementation of DTN, and we validated our architecture using real devices in scenarios with multiple motes as publishers and subscribers.

The results we obtained confirm that MQTT is one of the best options in terms of resource use and ease of development for IoT applications, and that, in conjunction with a DTN approach, can be very robust and efficient even in scenarios with unstable links or partitioned networks.

The rest of the article is organized as follows: A review of related work in addition to a short overview of the used technologies on our architecture. are offered in Section II. In Section III we present the architecture and the performance evaluation planning. The results and their statistical analysis are presented in Section IV. Some concluding remarks are available in Section V.

II. Related Work

With the arise of the Internet of Things (IoT), several specific protocols are being widely used by application developers. A basic performance comparison between well known implementations of the DDS, MQTT, CoAP protocols and an UDP program under network emulation using NetEm is presented in [4]. They simulate constrained and low reliability networks with the aim of measuring bandwidth, latency, and packet loss. They experience packet loss with protocols based on UDP and with protocols based on TCP when the link

[1] *https://github.com/ibrdtn/ibrdtn*

conditions imply over 25% of packet loss, and 400 ms of latency.

Another comparison between IoT protocols is offered in [5] where a distributed solution based on ZigBee and 6LoWPAN is compared against a centralized solution based on Software Defined Wireless Network (SDWN). The authors observe performance metrics such as packet loss, RTT and overhead. The obtained results show that the best solution for static or semi-static smart homes and buildings is the SDWN due to the optimal resources' exploitation combined with reduced overload. However in dynamic environments SDWN presents limitations due to the large amount of time required to refresh routes, where ZigBee and 6LoWPAN remain to be the best options.

Also, through the NS2 simulator, authors in [6] evaluate the performance of the MQTT-SN by changing the quality of service levels. In addition, authors also propose a theoretical probability of delivery delay as a function of other parameters is modeled. The authors as future work propose to make the measurements on a real implementation similar to what we have done in this article.

The authors in [7] have experienced problems and difficulties to link the world of internet and the world of things, specially with proprietary solutions on gateways. They propose the use of smartphones as gateways in conjunction with an Android application running to discover and manage things that collects and forwards their data. Through this solution, interoperability of scenarios, interfaces and technology is provided.

Another group of works focuses on handling disruptions without data loss on scenarios with variable network topology due to unstable network links, and based on the Bundle Protocol specifications, in [8] the authors describe an implementation of the bundle protocol for WSN specifically for Contiki called μDTN, outlining the design architectural decisions. As a convergence layer the authors directly use MAC Layer 802.15.4, thus bundles are send directly to 802.15.4 radio frames, without going-through transport or network layers.

In [9] the authors propose the replacement of UDP as the transport protocol used by CoAP in the Bundle Protocol offered by IBR-DTN. The bindings are made via TCP sockets without regarding to the security features. In the evaluation they compare their proposal against a standard CoAP implementation. The results show a slower personal behavior, but instead this proposal allows have no end-to-end communications supporting long expected disconnections.

However all these works do not offer an architecture able to combine these promising standards with a successfully proof of concept that uses real devices.

III. Experimental Set-up and Evaluation Methodology

In this section we describe the experimental setup and the evaluation methodology we used to analyze our proposed architecture.

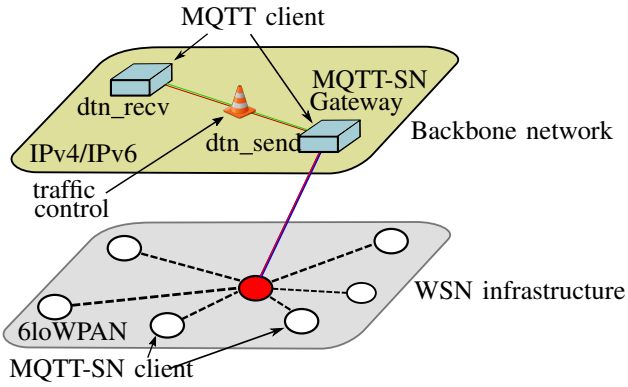

Fig. 1. Diagram of the conceptual integrative architecture for data collection applications.

A. Reference Scenario

Our experimental set-up included two Raspberry Pi 2 Model B (RPi) devices and seven Zolertia Re-Mote Sensor Board (motes) [10]. To simplify our study the scenario was separated into two parts:

On one side we had the WSN infrastructure where the motes (MQTT-SN clients) were wirelessly connected using 6LoWPAN. An intermediate RPi acts as a gateway translating MQTT-SN to MQTT messages and vice versa [11]. We used the Eclipse Paho MQTT-SN gateway implementation[2] and the Mosquitto broker implementation[3] as MQTT message broker to distribute the messages from the publisher to the subscribers.

On the other side, we had the backbone IP-based network where the RPis, acting as DTN nodes, where connected using an Ethernet link. The bandwidth was fixed to 10 Mbps using the *Ethtool* Linux utility. We used IBR-DTN version 1.0.1 as the DTN implementation. For the bundle transmission we used the *dtnsend* and *dtnrecv* [12] tools.

One of the two RPi devices was connected to one of the seven motes to implemented a border router device [13]. This border router interconnected both networks and routed the generated data between them.

In Fig.1 we can see a graphical representation of the scenario while the real set-up is depicted in Fig. 2.

All the source code collected, studied, developed and adapted to our project is under open-source license and can be download from the repository: *http://github.com/jluzuria2001/TS-IT*.

B. Evaluation methodology

The evaluation methodology was based on separately considering the two main components of our proposed architecture: the WSN infrastructure, and the backbone network.

[2] *https://projects.eclipse.org/projects/iot.paho*.
[3] *http://mosquitto.org*.

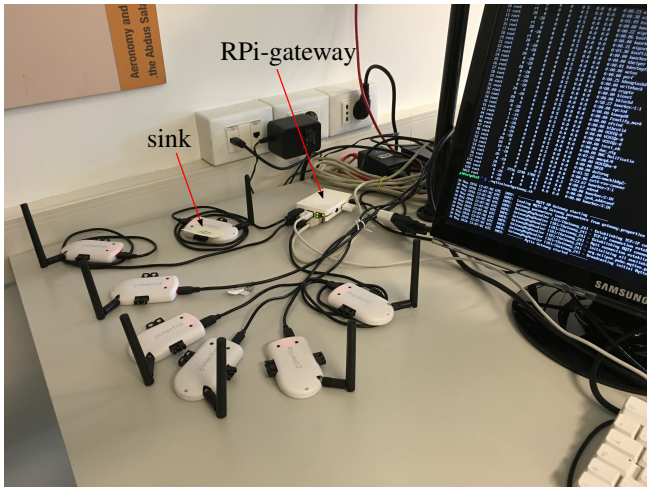

Fig. 2. A picture of the testbed used in our experiments.

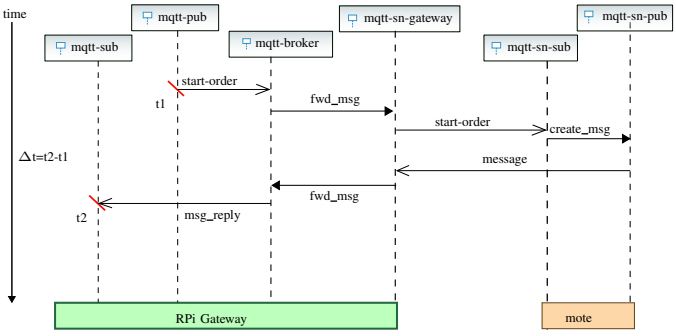

Fig. 3. Sequence Diagram of the Ping-Pong application, totally based on the MQTT-SN protocol.

Regarding the WSN infrastructure, we measured the performance of the connections by using a simple *ping-pong* testing application we developed. Fig. 3 shows we can see the interaction of the involved components as well as the points on which the various time-stamps were measured. The basic behavior of the *ping-pong* application is the following: first, a message is sent from the MQTT publisher on the RPi-gateway to the MQTT broker on the RPi-gateway; then, the MQTT-SN gateway forwards it to the MQTT-SN subscriber on a mote. When the subscriber on the mote receives the message, it creates a new message with the received identifier and publishes it to the MQTT-SN message broker, who then forwards it to the MQTT subscriber at the RPi.

The micro-controllers computation delay and all the other potential delays are considered to be part of the overall channel latency. The obtained results are based on a total of 5000 messages sent at a frequency of 0,1 Hz, and using a message size equal to 20 bytes.

The performance of the DTN-based backbone network was measured configuring the message forwarding on IBR-DTN as follows: 1) maximum lifetime of a bundle of 604800 seconds (i.e., one week), 2) blocks size limit of 1.3Gb, and 3) the non-persistent bundle storage, i.e., all bundles were kept in the RAM memory. To emulate an intermittent channel we used the Linux Traffic Control [14] tool. We modified the communications channel between the transmitter and the receiver varying the percentage of error over the link, specifically 0%, 25%, 50%, and 75%. To ensure at least two connected cycles and one disconnected cycle, and considering that the test length that is around 33 minutes, we fixed the length of each cycle to 12 minutes.

TABLE I
A SUMMARY OF THE PARAMETER'S DETAILS USED IN THE EVALUATION OF EACH SCENARIO.

Parameter	WSN infrastructure	backbone network
Total msg.	5000/2000	2000
Msg. size (bytes)	20	40
Periodicity (seconds)	10	1
Repetitions	10	10

Each test was repeated 10 times, transmitting 2000 messages with an inter-bundle frequency of 1 message/second. The bundle size was 40 bytes (20 bytes of the mote message plus the forwarding timestamps). We emulated the message forwarding in a full connected network, and then in an intermittently connected network where throughout the test, the transmitter network interface was turned off and on with cyclically prefixed time periods

Table I summarizes the configuration parameters used with each scenario.

C. Analyzed Metrics

The data of the message transmissions obtained from the log files were filtered by the ID of each mote. The metrics used in the evaluation of our architecture were: the round-trip times (RTT), the % of message losses, and the messages jitter.

1) Round-trip times: The round-trip times are used to evaluate the channel latency. The *ping-pong* testing application we developed was used specifically to this end. We basically used two log files: one in the MQTT-client at the RPi, storing the timestamp and the ID of each published message; the second, storing the same fields (timestamp, and ID) of each received message. These two logs were merged based on the message ID, and then we computed the difference between the reception and publication timestamp for each message. These values form 10 vectors, one for each test, which are used as the basis of our descriptive and probabilistic statistical analysis.

2) Message Loss: To calculate the message loss (*loss_msg*) we count all the received messages (*rcv_msg*) and subtract it to the total number of sent messages (fixed to 2000). The average values were stored in a vector for each test, eventually providing the following data matrix:

$$[avg\_vector] = (\sum_{i=1}^{n} a_{i1}, \sum_{i=1}^{n} a_{i2}, \dots \sum_{i=1}^{n} a_{im}) \qquad (1)$$

where: a_{kl} are the received messages on a specific test; k is the number of motes ($1 \leq k \leq n=6$), and l is the number of test ($1 \leq l \leq m=10$). The total number of messages sent in each test was 2000*n.

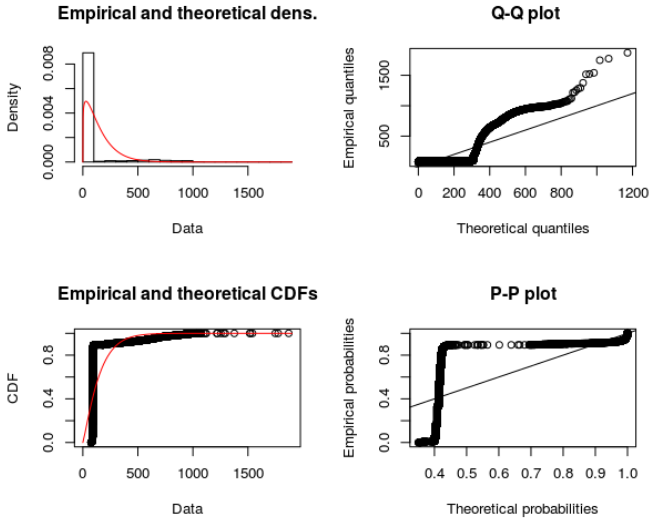

Fig. 4. Empirical and theoretical statistical distributions with the Round-trip time results of the Ping-pong application.

3) Jitter behavior: The jitter of the received messages is computed using the reception timestamps. The timestamps' accuracy was down to milliseconds at the sending and receiving sites. We considered that relative clock drifts during the experiment were negligible, and therefore we used the clock values of the final end-point on the RPi. For every test we take the difference of the arrival values, building a new matrix of values whose *i*th row has the following elements:

$$Jitter_{i,rcv\_ts} = (a_{i+1,rcv\_ts} - a_{i,rcv\_ts}) \qquad (2)$$

where: *a* correspond to the received messages on a specific test, *rcv_ts* represents the reception timestamp, and *i* is the current message ($1 \leq i \leq 2000$).

IV. Analysis of the Results

In this section we present the results that allow evaluating our architecture. In our evaluation we addressed round-trip latency, % of message losses, and the order preservation of the messages when delivered after a network disconnection.

A. Round-Trip Time (RTT)

The first critical metric is the round-trip latency in the WSN infrastructure.

Fig. 4 shows the evaluation of the round-trip time (in milliseconds) for a 6LoWPAN payload size of 20 bytes. We can see that the data distribution of the RTT values does not follow a normal distribution. The density curve within the hump of the histogram is close to zero, with values between 75 and 90 *ms*. Based on quartiles information of the Q-Q plot we can see that only a few values are greater than 1000 *ms*. The regression line intercepts values between 0 and 1700 *ms*. In the cumulative distribution function (CDF) of the distribution plot, 90% of the values are within a few milliseconds, specifically between 75 and 90 *ms*, confirming that the data are not normally distributed. Also, a few unusually high values of 1.7 seconds appear. The P-P plot presents a behavior quite similar to the CDF plot; we can see that the measured values are not aligned along a regression line.

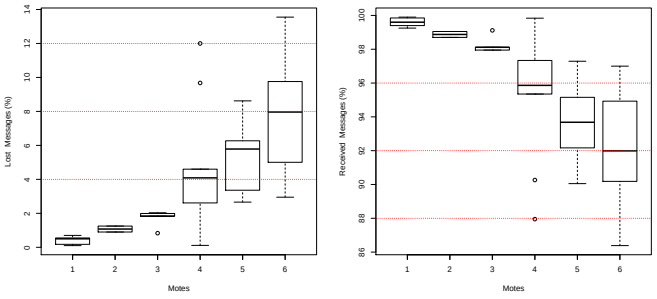

Fig. 5. Results of the Lost (left) and Received (right) Messages respectively, when increasing the number of publisher motes.

B. Varying the number of Publisher and Subscribers

This study was implemented to see the impact of increasing the number of publishers and subscribers, that is a context where several devices are sending/receiving packets simultaneously. To avoid the overloading of the motes memory the inter-message delay was fixed to 1 second, sending a total of 2000 messages. The message size during this test was set to 20 bytes, and each test was repeated 10 times.

We counted a successful transmission process when a published message to the message broker is forwarded and received by the subscribers on the motes or on the other RPi, where they are kept in a log files for statistical analysis. In the first part of the experiment, the publishers are the motes (many-to-one), and in the second part it is the RPi (one-to-many).

1) Message Loss: Fig. 5 on the left shows that message loss goes up from 0 to 10% when we add up to six motes to the scenario, due to the increasing number of collisions in the network in addition to the constrained resources of the motes. On the right part of Fig. 5 we can see how the successful reception of messages drops progressively as the number of motes increases. In all these simulations a message-loss higher than 9% (180 messages) never occurred. The general pattern showed either no packet-losses at all, or up to 100 lost messages with 6 motes.

2) Jitter behavior: On the 6LoWPAN network, the jitter values obtained when increasing the number of motes are shown in Fig. 6. On the left we observed how most of the *maximum* jitter values go from 3 to 9 *seconds*, while on the right part most of the *minimum* jitter values range between 55 to 140 *ms*.

C. Inter-infrastructure Delay

We now present the results of the data delivery delay in the backbone networks. From the set of graphs in Fig.7 we can

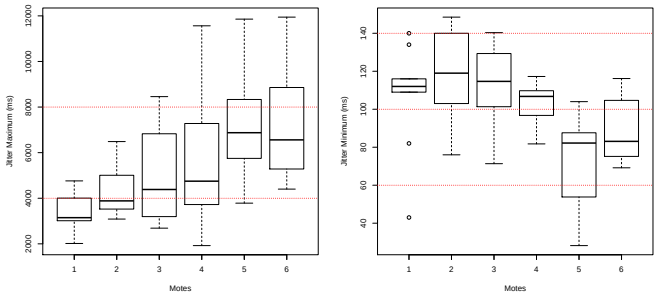

Fig. 6. Obtained Maximum (left) and Minimum (right) Jitter values respectively, when the number of publisher motes is increased.

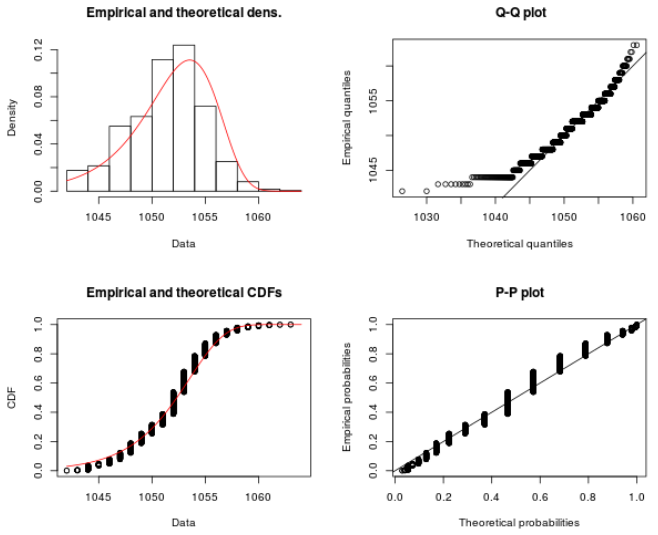

Fig. 7. Processing time requirements at the DTN node with an Inter-Message Sending interval of 1 second.

see that the range of values goes from 1040 and 1065 *ms*. Considering that the bundle generation was fixed on 1000 *ms*, this means that the needed time to handle a bundle with the devices used on our test falls within the range from 40 to 65 *ms* in nearly all the cases. If we compare the density curve hump with the hump of the histogram we can see that the overall data is normally distributed. The data in the Q-Q plot (on the right) also describes a normally distributed process. The values above the line on left of this graph are telling us that most of the values are lower than the medium value. By plotting the cumulative distribution function (CDF), we can see how the data are normally distributed like in the density plot. Normal P-P plots were used to examine whether the residuals are normally distributed. The pattern of the values grouped shows that some values are common on the data distribution.

D. Inter-Message Receiving Delay

This section shows the inter-message gap in the reception of messages in two evaluated cases: (a) without disconnections,

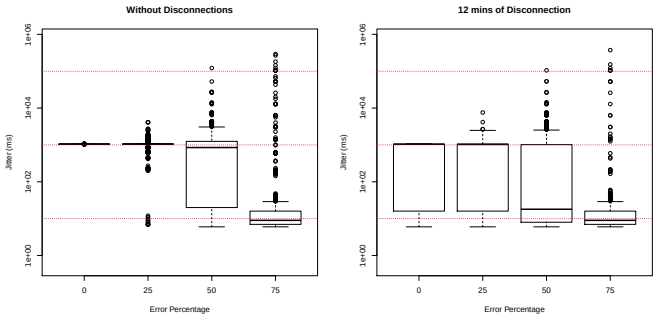

Fig. 8. Variation of the inter-message reception delay in a fully connected scenario (left) and with cyclic connections/disconnections of 12 mins (right) at the DTN node.

and (b) with cyclic disconnections with time intervals of 12 minutes in length. All the scenarios suffer from different percentage of error on the communication channel.

Fig. 8 on the left shows the results achieved under optimal conditions, i.e., with 0% of error over the link. We can see that the values are bounded to a few milliseconds (between 1031 and 1078 *ms*). With a 25% of channel errors, most of the messages are delivered with a few milliseconds around the inter publishing rate with some outliers close to 10 *ms*. When the error increases, most of the values are in tens and hundreds of milliseconds with few outliers with very high delays. Huge outliers behavior is common with 75% of error over the link. However most of the messages are delivered in presence of a minimum connection with an inter delay of tens of milliseconds.

In the presence of disconnections from the network, we can observe on the right part of Fig. 8 that messages are delivered with a delay between 10 *ms* to 1 second. When the error on the link surpasses 50% some outliers with big values appears, while most of the messages are delivered within a inter delay of tens of milliseconds. Basically, When a minimum connection is established it is enough to support the exchange of bundles.

We have observed that the average values of the time needed to re-establish a connection after a period of disconnection, subtracting the bundle generation time is close to 6 seconds in the best conditions; while with a 75% of error in the communication channel it requires about 200 seconds.

E. Order Delivery Analysis

The IBR-DTN module by default uses a non-persistent bundle storage, which means that all bundles are kept in RAM memory and so bundles will not be preserved if a power failure or a service daemon restart takes place.

All the bundles have the same priority, and they are stored and retrieved based on different parameters like their *id*, or the destination, among others. In a full connected network on the nodes' outbound link the FIFO queuing policy is applied. When the network has some constraints, limitations and the nodes are not reachable, the read bundles out of the storage are

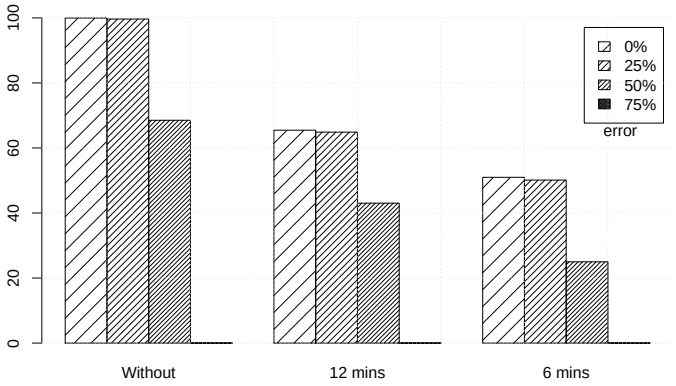

Fig. 9. Percentage of the Delivery order with different scenarios with and without disconnections in addition to the channel manipulation.

re-queued. In these cases the buffer management policy used by IBR-DTN to flush the buffer and send out the buffered bundles to the end node looks like a random policy.

In Fig. 9 we can see, that even in presence of errors on the link up to 25% data delivery order follows a FIFO fashion with a 100% of order in fully connected scenario, 65% in a scenario with 12 minutes of disconnections, and 50% in a scenario with more intermittent connections. When the errors over the channel increase to 50%, the order goes down to 70% in a full connected scenario, while with cyclic disconnections the order goes down to 43%. In presence of a high error % over the channel the delivery order in all the cases is completely broken, being that none of the bundles arrive in the same sequence in which they were sent.

V. Conclusions

We presented a DTN-based architecture to support MQTT for the development of Data Collection IoT Applications. Our proposal focused on offering data transmission in scenarios characterized by link outages, unstable links, split networks, or intermittent connectivity.

We have presented experimental results obtained from different configurations of network parameters and network size to validate the feasibility and effectiveness of our proposal. We showed that in the WSN infrastructure the round-trip time is below 75 *ms*, and the message loss gets up to 5% and 3% with 6 publishers or subscribers respectively. In the backbone network, the time required to handle a bundle is between 40 and 65 *ms*, while the inter-delivery message delay is of few tens of milliseconds, slowly increasing as the degradation of the communication channel grows. We consider that these values can be acceptable for general IoT applications, considering that with this new architecture end-to-end applications do not have to worry about the possible nodes' disconnections periods.

As expected, the most critical parameter has to do with the scalability of the network, considering that IoT systems can easily have hundreds of motes. It is necessary to keep in mind that a large number of motes degrades the network performance as we have demonstrated along the paper. So, it is crucial , when having high density networks to properly design the data collection and distribution solution to generate a load that the network is able to handle. We recommend to keep a very simple data collection scheme, like avoiding simultaneous collection and sending of data from large portions of the WSN infrastructure.

Acknowledgments

This work was partially supported by the *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. In addition, we would like to express our gratitude to the ICTP members for their assistance and support.

References

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

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

[3] V. Cerf, S. Burleigh, A. Hooke, and L. Torgerson, "Delay-tolerant networking architecture," in *RFC4838, April*, no. RFC 4838. IETF, apr 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4838.txt

[4] Y. Chen and T. Kunz, "Performance Evaluation of IoT Protocols under a Constrained Wireless Access Network," in *International Conference on Selected Topics in Mobile Wireless Networking (MoWNeT)*, 2016.

[5] C. Buratti, A. Stajkic, G. Gardasevic, S. Milardo, M. D. Abrignani, S. Mijovic, G. Morabito, and R. Verdone, "Testing protocols for the internet of things on the EuWIn platform," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 124–133, 2016.

[6] K. Govindan and A. P. Azad, "End-to-end service assurance in IoT MQTT-SN," *2015 12th Annual IEEE Consumer Communications and Networking Conference, CCNC 2015*, pp. 290–296, 2015.

[7] G. Aloi, G. Caliciuri, G. Fortino, R. Gravina, P. Pace, W. Russo, and C. Savaglio, "A Mobile Multi-Technology Gateway to Enable IoT Interoperability," *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pp. 259–264, 2016.

[8] G. V. Zengen, F. Büsching, W.-b. Pöttner, and L. Wolf, "An Overview of DTN : Unifying DTNs and WSNs," *The 11th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, no. September, 2012.

[9] M. Auzias, Y. Maheo, and F. Raimbault, "CoAP over BP for a Delay-Tolerant Internet of Things," *Proceedings - 2015 International Conference on Future Internet of Things and Cloud, FiCloud 2015*, pp. 118–123, 2015.

[10] Zolertia , "Zolertia RE-Mote platform Datasheet," vol. 001, no. December, pp. 1–2, 2015.

[11] Z. Shelby and C. Bormann, *Using 6LoWPAN*. John Wiley & Sons, Ltd, 2009, pp. 149–161. [Online]. Available: http://dx.doi.org/10.1002/9780470686218.ch6

[12] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf, "IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation," *Electronic Communications of the EASST*, vol. 37, jan 2011.

[13] A. Linan Colina, A. Vives, M. Zennaro, A. Bagula, and E. Pietrosemoli, "IoT in 5 days," 2016.

[14] B. Hubert, T. Graf, and G. Maxwell, "Linux Advanced Routing & Traffic Control HOWTO," *Ottawa Linux ...*, vol. 1.0.1, p. 631, 2012. [Online]. Available: http://www.lartc.org/lartc.pdf