Adding QoS Support For Timeliness To The Observe Extension Of CoAP

A. Ludovici, E. Garcia, X. Gimeno, A. Calveras Augé
Wireless Network Group (WNG), Dept. Telematics Engineering
Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain
anna.calveras@entel.upc.edu

Abstract—**In this paper, we propose a modification of the observe extension of CoAP to include Quality of Service (QoS) support for timeliness. The proposed QoS support is based on delivery priority. Nodes are able to express the priority order with which they wish to be notified. Furthermore, they are also able to specify which notification they want to receive. We classify the notifications in two categories: Critical and Non-Critical. The provided QoS is the result of a negotiation between the client and server. The client demands a certain degree of QoS according to its role. The server could accept or negotiate it. This choice depends on the availability of server and network resources. We evaluate our proposal in a real Wireless Sensor Network. An e-health application has been chosen as target of our tests. The performance evaluation is done in terms of average delay, energy consumption and delivery ratio.**

Keywords-component; CoAP; QoS; WSNs; Publish/Subscribe; Observe;

I. INTRODUCTION

Wireless Sensor Networks (WSNs) are data-centric, large-scale and event-driven networks. The information flow is mainly constituted by data collected by sensors. A sensor can collect data periodically or in occurrence of an event. In either case these data have to be transferred elsewhere in the WSN or to external networks. Nodes of the WSN could be interested in these data to perform local calculations, data aggregation or to take action to react to the occurrence of specific events. External networks could be interested in monitoring an environment, spreading data among heterogeneous users or in receiving data for computational purpose. WSNs can be composed by a large number of nodes or could interact with many external networks.

The data transfer model to adopt should, therefore, be compliant with these characteristics. Moreover, it should be as simple and as lighter as possible. In particular, its design has to minimize the resource consumption. WSNs, in fact, are distinguished to traditional networks by having constrained resources. Sensor nodes are usually battery-powered and equipped with few kilobytes of RAM memory. Another problem is that bandwidth is limited. Thereby, the number of messages exchanged between nodes should be reduced as well as the protocol overhead.

In this context, the traditional request/response model is inefficient. Clients have to periodically poll the server to receive updates of the state of a monitored resource or event. As a consequence, the traffic will increase dramatically congesting and overloading the WSN. Furthermore, the resource consumption would grow. Instead, an asynchronous model would be more suitable for these characteristics. In this sense, the publish/subscribe model would be an optimal solution. Sensor nodes would transfer data only when these are available and not in response to a request. Clients, which are called subscribers, can register to services offered by servers and receive a notification when new data is available. The term publisher is used to refer to the server. The adoption of this model would therefore reduce the traffic by avoiding sending useless messages. As a consequence, the consumption of nodes resources would be reduced.

The IETF Constrained RESTful Environments (CoRE) work group [1] has defined a new Web transfer protocol called Constrained Application Protocol (CoAP) [2]. CoAP seeks to apply the same application transfer paradigm and basic features of HTTP to WSNs. In contrast with HTTP, CoAP is designed to meet the requirements of constrained networks. Therefore, it is able to reduce protocol overhead and resource consumption. CoAP is a promising protocol for implementing Web applications in WSNs and, therefore, realizing the "Web of things" model. An extension of CoAP, which is called observe, has been proposed in [3] to implement the observer design pattern [4]. Although its communication paradigm is similar to that of the publish/subscribe model, the observer implies a simpler architecture and a less complex data transaction.

The peculiarities of both, the publish/subscribe model and the WSN, require providing QoS support to applications. For instance, many applications could have real-time requirements. That implies that a notification could be no more valid if received after a given time. Moreover, the correct delivery of a notification cannot be guaranteed. Wireless links are, in fact, prone to packet loss. Common solutions used in publish/subscribe systems provides reliability and timeliness as QoS parameters. The first is used to ensure to subscribers the reception of notifications. The timeliness is used to guarantee the on-time delivery of packets.

The observe extension provides end-to-end reliability by using the confirmable (CON) messages of CoAP. Regarding timeliness, it only provides the option of indicating the validity of a notification over a period of time. This is achieved using the freshness model of CoAP caching. However, it does not specify how to guarantee on-time delivery. Meeting the deadline requirements of a notification depend in large part to the delivery process. In this sense, the order in which the nodes are notified is of paramount importance. The current observe extension model gives to the server the faculty of choosing in which order clients are notified. The current delivery model could be inefficient for many application domains. We argue that the server should be able to prioritize the delivery of notifications to nodes requiring it. Our approach is motivated by the fact that WSN nodes could have distinct roles and requirements. As a consequence, timeliness requirements could vary depending on the characteristics of a node. For instance, in e-emergency applications the notification of a critical event must be prioritized to those nodes in charge of reacting to it. Furthermore, these nodes could be interested only in being notified in case a critical event occurs. Therefore, the server should avoid sending them each notification. Other nodes, however, could be interested in receiving all of them. Therefore, a server should be able to distinguish which information sends to a particular client.

In this paper, we propose a modification of the observer extension to enable QoS support for timeliness. The proposed QoS support is based on delivery priority. Clients are able to express the priority with which they wish to be notified. Furthermore, they can specify which notification they want to receive. We classify notification in two categories: Critical and Non-Critical. The provided QoS is the result of a negotiation between the client and server. The client demands a certain degree of QoS according to its role. The server could accept or negotiate it. The server, in fact, could not be able to provide the required QoS. This choice depends on the availability of server and network resources. We demonstrate the effectiveness of our proposal by a performance evaluation in a real WSN. This evaluation is performed in terms of delay, delivery ratio and energy consumption.

The paper is organized as follows: First, in the following section, we present the design of the observe extension. Then, in Section 3, we present related works on QoS support for publish/subscribe systems. Our proposal is illustrated in Section 4. The experiment setup is described in Section 5. Preliminary results and discussion of a performance evaluation are reported in Section 6. Finally, in Section 7 we conclude the paper and give guidelines for future work.

II. Observe Extension for CoAP

The observe model is formed by two components: the observer and the subject. The observer is a client interested in being notified by changes of the state of a resource while the subject is the server that provides it. The conventional architecture of publish/subscribe systems for WSNs has been based on the presence of a broker [1]-[11]. This is in charge of coordinating and distributing notifications and subscriptions requests that it receives from publishers and subscribers. Differently from publish/subscribe systems, the observe model avoids the use of a broker allowing the subject and the observer to communicate directly. Therefore, the resulting architecture is less complex and it is able to reduce latency. The presence of intermediaries, however, is expected in the observe model to enhance scalability. An observer can register its interest to an intermediary node. This will register itself to the subject and then it will forward to the observer the notifications it receives. All the process is transparent and it is particularly suitable for multi-hop or large-scale networks where the subject is located many hops away from the observer. Fig. 1 shows the architecture of a CoAP based WSN adopting the observe protocol
extension.

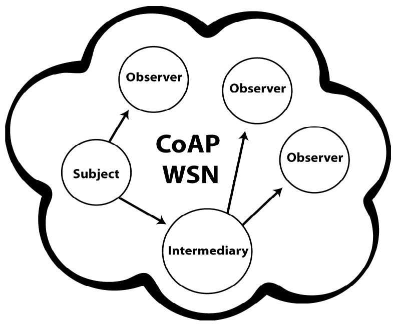


Figure 1 CoAP based WSN implementing the observer extension. An intermediary can be used for scalability purposes.

A. Registration process and notifications

The registration procedure followed by the observer is kept as simple as possible. The observer shows its interest by sending an extended GET request message to the subject. As a consequence, the subject registers it as an observer and then notifies it when the state of the resource changes. To confirm the successful registration, the subject sends an extended response with the current state of the resource. The term extended means that the CoAP request or response contains the observer option as defined in [10]. The value of this option will be always zero if it is contained in a request. In a response it is different from zero and it is used as a sequence number. A token option [10] is used to match the notifications received by the observer with its registration. The token value specified in the registration message is used for all the notifications. The subject, however, may decline the observer request by answering with a response not containing the observe option. This indicates to the observer that its request has been rejected.

An observer has the option to be removed from the list of observers of a resource. Should an observer respond to a notification with a RST message, the subject will remove it from this list. An observer is also cancelled if it sends a simple GET request message to the observed resource. Fig. 2 shows an observer registering to a resource, then receiving a notification and the deleting its interest.

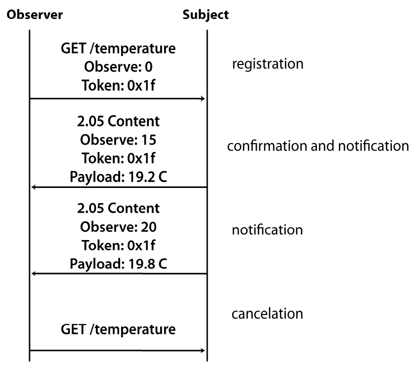


Figure 2 The observer delete its interest sending a GET request without the observe option

B. *QoS in the observe extension*

The observe extension supports reliability. It uses the mechanism defined in CoAP. This provides end-to-end reliable data transaction using CON messages [2]. A node receiving a CON message must acknowledge its reception to the sender. In case the sender does not receive an acknowledgment (ACK) then it retransmits the request. The retransmission process follows a stop-and-wait model with exponential back-off. A node could also send unreliable messages by using non-confirmable messages (NON). In the observer extension the use of NON or CON messages depends exclusively on the subject. For each individual notification the subject decides using or not reliability. The QoS support for reliability can therefore be divided in two categories: best-effort using NON messages or persistent when using CON ones.

Regarding the timeliness parameter, the observe extension allows to specify the validity of a notification but not to guarantee its on-time delivery. The indication of the validity is achieved using the caching model of CoAP [2]. This defines a freshness model to indicate the period of time in which the cached response is valid. The subject is responsible for deciding the length of this period. Its value is contained in a CoAP option, which is called max-age. An observer can cache and re-use a notification until it is not expired. Moreover, observers can use the max-age value to control if they are still registered to the subject. If an observer does not receive a notification after the max-age has expired, it may assume that it has been cancelled.

The lack of support for on-time delivery could not allow meeting the different delay requirements of observers. As previously mentioned, the delay of a notification experienced by an observer is strongly affected by the delivery order followed by the subject. The current observer definition leaves the decision of the order to the subject. In this paper, we argue that this approach is inefficient for many applications wishing to use the observe extension. The delivery order should, in fact, be differentiated depending on the requirements of each observer. In this sense, we present a proposal of modification of the observe extension for supporting timeliness. Our contribution seeks to define a simple mechanism for establishes the delivery order based on the priority expressed by observers. Furthermore, the observers can express their interest in which kind of notifications they want to receive.

In the next section we present related works on QoS support in publish/subscribe systems.

III. Related Works

As previously mentioned, the most accepted QoS parameters considered in publish/subscribe are reliability and timeliness. The negotiation of their level between the publisher and the subscriber is not common. Although important, few protocols have this characteristic.

The authors of [12] provide a mechanism for negotiating QoS in terms of timeliness. In this model, a subscriber can specify the delay constraint as an attribute of its subscription request. The broker will then select the best path to route the notification and, therefore, meet the delay requirement.

Reliability can be provided with best effort or persistence mode. In best effort the retransmission of lost notifications is not provided. In persistence mode the publisher is able to retransmit notifications. The solutions presented in [13]-[15] provide reliability only with best effort. Persistence is provided in [16]-[18]. These systems, however, only provide reliability from the fault-tolerance point of view. They use mechanisms to enable brokers to recover after a failure or, to route information towards active brokers. They do not negotiate the reliability with subscribers.

Regarding the protocols designed for WSNs, MQTT-s [9] allows to subscribers to define three QoS levels for reliability. The first level offers a best effort delivery service. The second allows the retransmission of a notification until receiver acknowledges it. Retransmitted messages may arrive duplicated at the destination. To overcome this problem, the third level ensures that the same message is received only once.

The authors of [19] define QoS support for real-time publish/subscribe in WSNs. A Subscriber can specify the maximum tolerated delay for receiving notifications. A dispatcher is used to meet the delay requirement. Depending on the required delay, the notification can be buffered in a QoS or in a non-QoS queue. The dispatcher gives priority to the sending of notifications with real-time constraints. In [20], the same authors propose a publish/subscribe middleware for WSNs that provides a mechanism for supporting fault-tolerance and real-time requirements. This is achieved through a cluster-based organization of the WSN. The middleware receives QoS requirements from the cluster-head nodes. These requirements are about the services of the WSN and their default operation conditions in terms of delay, data rate and energy. Once the middleware has this information it is able to provide the required QoS.

IV. Adding QoS differentiation to the observe extension

As previously mentioned, the observe extension has drawbacks in QoS support for timeliness. In particular, the subject is the only node able to manage the delivery order of notifications. Therefore, its choice can be based only on the information available at server side. Commonly, it concerns the characteristics of the resource that the subject is monitoring. The requirements of observers are not considered in the current model. Although the deadline of a notification depends on the type of data that are contained in it, its delivery should be differentiated depending on the particular observer. In fact, the total delay of a notification undergoes a pronounced variation depending on the delivery order. Varying with the WSN application domain, each observer could have a different role that could imply a distinct delay requirement. In a WSN used for fire detection, for instance, a node in charge of the fire-extinguishing process should be the first to be notified in case of a fire is detected. Furthermore, an observer could only be interested in receiving only critical notifications. For instance, in the e-health domain an alarm is only interested in knowing critical states of the patient rather than each single state. Furthermore, it could require to be notified only when the critical situation is detected and when it finishes. The adoption of a mechanism that allows an observer to explicit the notification it needs, allows minimizing the resource consumption of subjects and observers.

In our proposal, observers can request a priority level in the delivery of notifications. Moreover, they can indicate which kind of notifications they wish to receive. As previously mentioned, a notification can be classified as critical or non-critical. The subject is the only component that has the authority to make this decision. This choice depends exclusively on the application domain of the WSN. Usually, these are not multi-purpose networks. They are deployed to perform a specific task. Each node is aware of its role and of the resource it is monitoring. Therefore, it is expected that the subject will have the necessary information to establish the criticality of a notification.

The modification that we propose defines the support for four levels of QoS: low, medium, high and highest. The subject will prioritize the delivery to observers requiring the highest level. This level is intended for observers that are interested in being notified when a critical event is detected and when it finishes. The high level is required for those nodes that show interest in receiving each critical notification. They are notified immediately after the observers with highest priority. Observers interested in receiving both critical and non-critical notifications can demand a low or medium level. The medium level has priority on the low one. Should more than one observer require the same level, the subject should order them following the order at which their requests arrived.

The value of the QoS level is specified using the two most significant bits of the observe option value. We will refer to those bits as QoS field. This modification affects only the requests sent by observers for registering and the consequent response of the subject. The bits used by the QoS field do not affect the subsequent notifications and, therefore, the maximum value of the observe option does not change. The definition of the four QoS level and the relative QoS field values is the following:

- Level 1: The subject sends non-critical and critical notifications with low priority. The value of the QoS field is 00.
- Level 2: Both non-critical and critical notifications are sent with medium priority. The value of the QoS field is 01.
- Level 3: The subject sends only critical notification with high priority. The value of the QoS field is 10.
- Level 4: The subject notifies with the highest priority only the start and the end of a critical state. The value of the QoS field is 11.

The model we propose expects that the observer and the subject negotiate the QoS. A subject, in fact, should have the faculty to reject or negotiate the demanded QoS. It could not have enough resources to satisfy all the requests. An observer could demand the highest priority but the subject could have already other observers with the same level. Therefore, the subject could not meet the delay requirement of the observer and could offer a lower priority level. Furthermore, the subject could recognize that the energy level of its batteries is under a threshold. Therefore it could not satisfy a new request from an observer that requires each notification. Should the subject accepts the request, it replies with the observe option containing the QoS value requested. The subject expresses its willingness to negotiate the QoS by replying with the offered QoS. Should the observer accepts, it would send a response containing the offered QoS. The observer can reject the offer by ignoring it.

The definition of a routing strategy that a subject can use to reduce the delay is out of the scope of this paper. Our proposal focuses in reducing delay by defining a simple mechanism for establishing the delivery order of notifications. A further optimization of the delay is achieved by allowing to observers to register their interest according to the kind of notifications they want to receive.

V. Experimental Setup

The effectiveness of our proposal is evaluated in a real WSN. We run tests to calculate the average delay of notifications, their delivery ratio and the energy consumption of the observer. TelosB motes [21] are used as a hardware platform to develop our test-bed network. The observe extension has been implemented as part of a CoAP implementation for TinyOS, which is called TinyCoAP [22]. The IEEE 802.15.4 standard [23] is used as solution for the MAC and physical layers.

Our test-bed network meets the requirements of an e-health application used to monitor the cardiac rate of a patient. We choose this scenario because of its criticality. E-health applications, in fact, have strong real-time requirements. The results, however, are general and valid for different application domains. We assume that several observers are interested in receiving notifications about the state of a patient. Each observer has different requirements in terms of priority and interest. In our experiments, we consider the presence of six observers: an alarm, a doctor, a nurse, an Intra Venous (I.V.), a general and a patient-specific monitor. The QoS requirements for each observer are the followings:

- Alarm: It needs to be notified when a critical event occur and when it ends. The level of the QoS is 4.
- Doctor: He needs to be notified only when the patient enters or leaves the critical state. The level of the QoS field is 4.
- Nurse: He has a tablet to monitor each state of the patient. The level of the QoS field is 2.
- General monitor: It receives the notifications of many patients. The level of the QoS is 1.
- Personal monitor: It is exclusive for the patient. The level of the QoS is 2.
- I.V.: It is active in case of emergency. It has to receive each critical notification. The level of the QoS is 3.

The characteristics of the notifications sent for monitoring the cardiac rate are specified in Tab.1. The values reported in this table are based on the characteristics of this parameter [24]. The deadline of a notification corresponds to its periodicity. In case of a critical notification this is equal to the sampling rate of the cardiac rate. In non-critical notifications, the data collected is aggregated and sent in a full 802.15.4 frame. Considering the presence of protocol overhead, the maximum payload size available is of 74 bytes. Therefore each notification will contains 37 samples of the cardiac rate and will have a periodicity of 3700 ms. The data aggregation allows to a subject reducing the number of notifications to sent to observer and, therefore, minimize the resource consumption.

TABLE I. CHARACTERISTICH OF CARDIAC RATE NOTIFICATIONS

Parameter	*Patient state*	*Payload*	*Periodicity*
Cardiac Rate	Normal	74 bytes	3700 ms
	Critical	2 bytes	100 ms

In our test-bed the subject and the observers are implemented in motes. The WSN follows a star topology. For simplicity we assume that the observers are equally spaced between each other and at the same distance from the subject. Fig. 3 shows the topology of the WSN. In the next section we present and discuss the results of our performance evaluation.

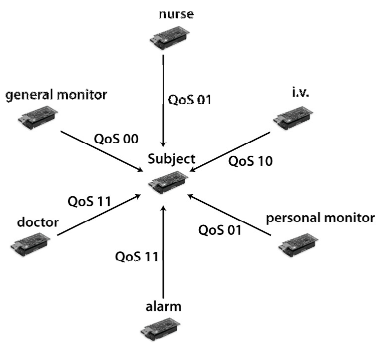
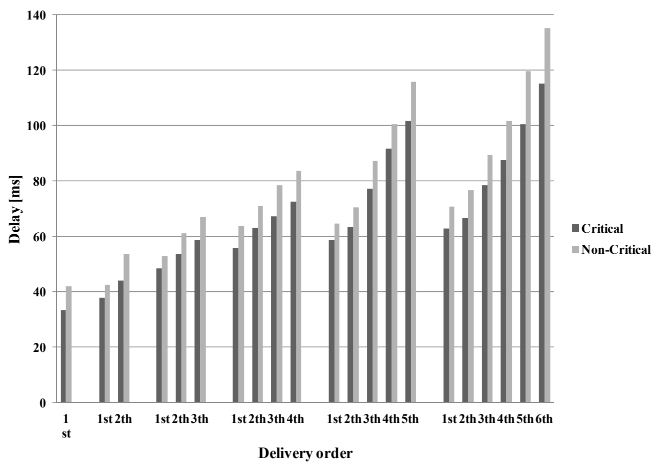


Figure 3 Topology of the test-bed network

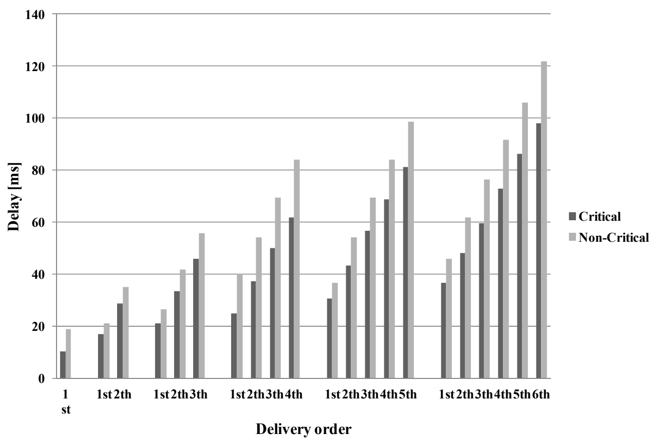
VI. RESULTS

The performance evaluation focuses on evaluating the delay and the delivery ratio of notifications. Moreover, we also evaluate the energy consumption of the subject. Our tests consider a subject sending both critical and non-critical notifications. For each case we differentiate the tests according to the reliability mode used.

A. Delay



(a) Persistence



(b) Best Effort

Figure 4 Delay as a function of the delivery order.

Fig. 4 shows the delay of notifications as a function of the delivery order. Each value represents the average delay of 100 delivered notifications. We define delay as the time elapsed from the moment the notification is created until the moment the observer acknowledges its reception. We consider both the use of best effort and persistence mode. In persistence mode the notification is acknowledged at CoAP layer while in best effort only at MAC layer. The IEEE 802.15.4, in fact, provides hop-by-hop reliability at this layer. The best effort mode relies on it. For each mode we run different tests according to the number of observers that receive the notifications. As previously mentioned, the characteristics of the notifications reflect those of an e-health application for monitoring the cardiac rate of a patient.

As one may observe from fig.4, the delivery order has a strong influence on the delay experienced by an observer. In non-critical notifications, however, this delay is considerable lower than the deadline of the notification. A further reduction of the delay is achieved by enabling the subject to avoid sending non-critical notifications to all the six observers. According to the QoS required by the observers of our test-bed, only three nodes receive non-critical notifications. More observers, however, could require these notifications.

The delivery order is of paramount importance in critical notifications. According to the size and topology of our test-bed network, the delay experienced by observers with lower priority is greater or equivalent to the deadline of a notification. We expect that this could be even greater in case more observers have to be notified. As a consequence, the current delivery mechanism of the observe extension could not guarantee the on-time delivery. When delivering critical notifications, especially in e-emergency applications, the subject has to follow a delivery order based on the characteristics and requirements of observers. As our results show, the use of a priority-based delivery is necessary to meet the timeliness requirements of observers with critical functions. Furthermore, we achieve a further delay reduction by allowing to observers to receive only the first and the last notification of a critical situation. As reported in fig.3, during the time the cardiac rate is in a critical condition the subject send notifications only to four observers. Therefore, the notification delay of each observer is lower enough to meet the deadline requirements.

A further analysis has to consider the reliability support of the observe extension and the effects this has on delay. As one may expect, the use of the persistence mode implies a growth of the delay. In fact, the processing time taken by an observer to reply to a notification with a CoAP ACK is higher than that required for sending a MAC ACK. Besides the packet processing time, a further delay is introduced by the CSMA-CA mechanism of the 802.15.4. In our tests we use its unslotted version. According to its definition [23], a subject wishing to send a notification has to wait for a random number of backoff slots and, then, it performs the clear channel assessments (CCA). Should the subject finds the channel idle, it sends the notification and then switches from the transmitting to receiving mode to receive the MAC ACK. Once received the notification, the destination node acknowledges immediately its reception sending a MAC ACK without using the CSMA-CA. Instead, the CoAP ACK is subjected to the same channel access procedure of a notification. Therefore, in persistence mode the observers and the subject have to compete for accessing the channel. The probability that one of them finds the channel busy could be high. In this case, they could perform several attempts for transmitting the packet. As a consequence, the delay will increase. Due to the high number of nodes competing for the channel, the probability of a CoAP ACK colliding with the transmission of a notification could be high. In case of collision the nodes will retransmit the packets. However, as specified in [3] a subject should avoid retransmitting a notification if a new one is available. Therefore a collision could imply either an extra delay caused by retransmissions or the lost of the packet.

B. *Delivery ratio*

Fig. 5 reports the delivery ratio as a function of the delivery order. The values obtained refer to the transmission of critical notifications using persistence or best effort as reliability support. We define delivery ratio as the probability that an observer receives a notification correctly. We fix the retransmission timeout to 38 ms. The timeout value is equivalent to the sum of the average delay experienced in presence of a single observer and its standard deviation. The retransmission timer is activated when the notification has been sent. Therefore it does not consider the time required to generate it. The results reported in this test are only valid for WSN adopting a star topology. We expect that the persistence mode will achieve better results in multi-hop networks. The study of this topology is out-of-the-scope of this paper.

The results demonstrate that observers notified with high or highest priority have also a high delivery ratio. In persistence mode, a notification sent with these priorities encounters a less congested channel and therefore has less chance to collide with a CoAP ACK. Even if the notification or the acknowledgment does not arrive correctly, the subject has more time to retransmit it before a new one is ready. As mentioned above, a notification is not retransmitted if a new one is available and therefore it is considered as a lost packet. Notifications sent in best effort yield a better performance in terms of delivery ratio. Here, the subject is the only node transmitting packets and therefore it does not compete for the channel. As a consequence, the channel is less congested and the probability of collision is negligible. The performance reached in best effort is more suitable for an application domain as e-health where having reliable communications is of paramount importance.

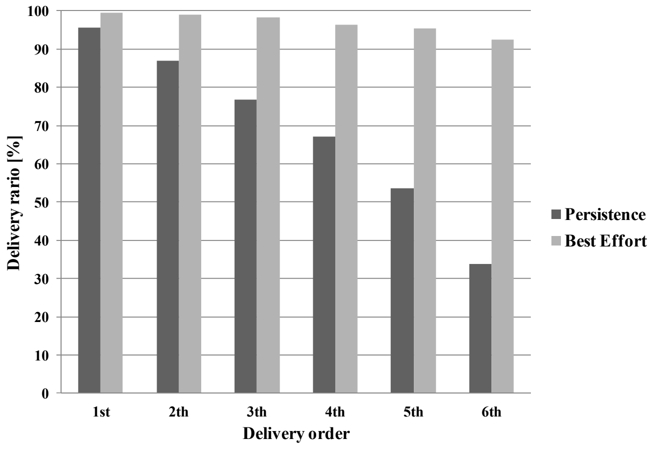


Figure 5 Delivery ratio as a function of the delivery order. A delivery order based on priority allows guarantying high delivery ration to observers requiring high priority.

C. *Energy*

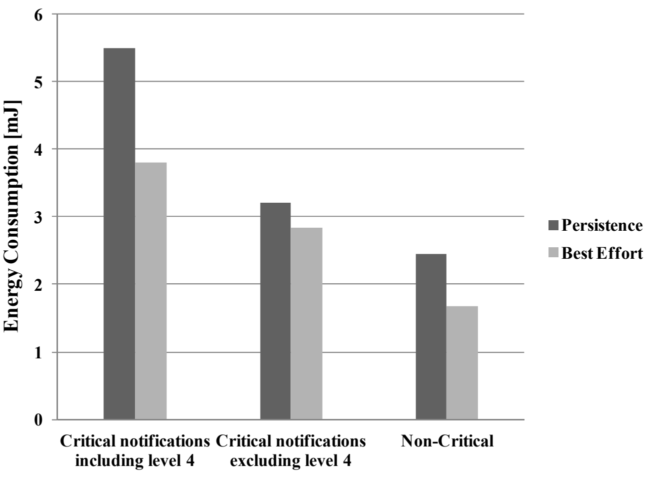


Figure 6 Energy consumption of the subject. The subject saves energy by avoiding sending all the critical notifications to all observers.

Fig. 6 shows the results obtained in the energy consumption test. This is focused in evaluating the energy consumed by a subject when delivering critical or non-critical notifications. The results of this test do not take into account the energy consumed for listening the channel. The measures are only inherent to the energy consumed for processing and sending notifications and receiving the CoAP or MAC acknowledgment from the observers. As a consequence, our evaluation does not need to consider power-saving protocols for radio duty cycling. The energy is sampled each 0.02 ms. The device used for these measures is the Agilent Technologies DC power Analyzer N67705A.

Notifications sent in persistence mode involve a higher number of messages respect to those sent in best effort. As a consequence, the subject consumes more energy. The increased consumption is due mainly to the energy used by the radio chip for receiving the CoAP ACKs and, to a lesser extent by that consumed for processing them. As one may expect, the subject consumes less energy when sending non-critical notifications. In this sense, our proposal of enabling an observer to choose which notifications receive allows to reduce the energy consumption of the subject. Regarding critical notifications, the consumption is significantly reduced by avoiding sending all of them to those observers that choose the QoS level 4.

VII. Conclusion

In this paper we present a proposal for adding QoS support for timeliness to the observe extension of CoAP. In this perspective, we suggest the adoption of a simple notification delivery mechanism based on priority. The level of priority is requested by the observers. This level establishes the order in which the subject send a notification. We define four level of priority. Each level is associated to a class of notifications that the observer wishes to receive. We classify the notifications as critical or non-critical. The subject is the only authority able to distinguish the class of a notification. At present, the definition of the observe extension leaves to the subject the decision of which notifications send and in which order. This approach could have limitations since it does not consider that observers could have different requirements. A subject could have no information on the observers and, therefore, it would be unable to meet their requirements.

We run tests focused on evaluating the effects that our proposal has on a WSN adopting the observe extension. The tests are done in a real WSN. The requirements of the observe extension application are those of an e-health network used for monitoring the cardiac rate of a patient. The performance evaluation is done in terms of delay, delivery ratio and energy consumption. The results confirm that the delivery order of a notification influence both the delay and delivery ratio. The first notifications sent by a subject experience less delay and have a high delivery ratio. A delivery order based on priority is therefore essential for providing on-time delivery to observers requiring it. The energy consumption of the subject is considerably reduced if observers are able to express the notification class of interest.

In conclusion, our proposal offers an effective solution for adding timeliness support to the observe extension of CoAP. The priority based delivery proposed allows to the observe extension to work better with applications with real-time requirements. Furthermore, our proposal is able to maintain the simplicity of the observe extension by requiring only two bits of its option value. These are used only when an observer register its interest to a subject. The option value of the subsequent notifications is not affected by these bits.

Future work will focus on extending and evaluate our proposal in WSNs with multi-hop topology.

VIII. Acknowledgment

This work has been supported by the Spanish Government, MICINN, under research grants TEC2009-11453 and FEDER, by the Catalan Government (Commissionat per a Universitat I Recerca del DIUE), the European Social Budget and the Professional Excellence Program SENACYT-IFARHU-Panama.

IX. References

[1] CoRE IETF Working Group; Available online: http://datatracker.ietf.org/wg/core/charter/ (accessed on 28 June 2011).

[2] Z. Shelby, K. Hartke, C. Bormann, B. Frank, Constrained Application Protocol (CoAP), Available online: https://datatracker.ietf.org/doc/draft-ietf-core-coap/ (accessed on 30 May 2012).

[3] K. Hartke, Observing Resources in CoAP, Available online: https://datatracker.ietf.org/doc/draft-ietf-core-observe/ (accessed on 30 May 2012).

[4] E. Gamma, R. Helm, R. Johnson, J. Vissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994

[5] Choon-Sung Nam, Hee-Jin Jeong, Dong-Ryeol Shin, "Design of wireless sensor networks middleware using the publish/subscribe paradigm", IEEE International Conference on Service Operations and Logistics, and Informatics, IEEE/SOLI 2008. Page(s):559-563

[6] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, J. Kelner, "Mires: a publish/subscribe middleware for sensor networks", Personal and Ubiquitous Computing, Vol. 10, Feb 2006, pp. 1617-4917.

[7] T. Luckenbach, P. Gober, S. Arbanowski, S. Kotsopoulos, "TinyREST: A protocol for integrating sensor networks into Internet", In Proc. Of REALWSN 2005.

[8] S. Krishnamurthy, "TinySIP: Providing Seamless Access to Sensor-based Services", Third Annual International Conference on Mobile and Ubiquitous Systems, pp. 1-6 July 2006.

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

[10] J. H. Schönherr, H. Parzyjegla, and G. Mühl, "Clustered publish/subscribe in wireless actuator and sensor networks", In MPAC '08: Proceedings of the 6th international workshop on Middleware for pervasive and ad-hoc computing, page(s) 60–65, New York, NY, USA, 2008. ACM.

[11] P. Costa, G.P. Picco, S. Rossetto, "Publish-Subscribe on Sensor Networks: A Semi-probabilistic Approach", IEEE International Conference on Mobile Adhoc and Sensor Systems, Nov. 2005

[12] N. Carvalho, F. Araújo, L. Rodrigues, "Scalable QoS-Based Event Routing in Publish-Subscribe Systems", In Proc. of the Fourth IEEE International Symposium on Network Computing and Applications (NCA 2005).

[13] C. Fengyun. P.S. Jaswinder, "Medym: match-early with dynamic multicast for contentbased publish-subscribe networks", In Proceedings of the ACM/IFIP/USENIX '05 International Conference on Middleware, pages 292–313, New York, NY, USA, 2005.

[14] G. Cugola, E. Di Nitto, and A. Fuggetta, "The jedi event-based infrastructure and its application to the development of the opss wfms", IEEE Transactions on Software Engineering, pages 827-850, 2001.

[15] H.A. Jacobsen, A. Cheung, G. Li, B. Maniymaran, V. Muthusamy, R.S Kazemzadeh, "The PADRES Publish/Subscribe System" In Principles and Applications of Distributed Event-Based Systems, pages 164-205, IGI Global, 2010.

[16] B. Gruber, E. Krishnamurthy, E. Panagos, "The architecture of the READY event notification service" Electronic Commerce and Web-based Applications/Middleware, 1999. Proceedings. 19th IEEE International Conference on Distributed Computing Systems Workshops on , pp.108-113, 1999.

[17] P.R. Pietzuch, J. Bacon, "Hermes: A Distributed Event-Based Middleware Architecture", Proceedings of the 22nd International Conference on Distributed Computing Systems, p.611-618, July 02-05, 2002

[18] MQ Telemetry Transport, Available online: http://mqtt.org/ (accessed on 30 May 2012).

[19] M. Sharifi, M.A Taleghan, A. Taherkordi, “A Publish-Subscribe Middleware for Real-Time Wireless Sensor Networks”, International Conference on Computational Science; ICCS 2006, LNCS, 2006, Volume 3991/2006, Page(s): 981-984.

[20] M. Sharifi, M.A Taleghan, A. Taherkordi, “A Middleware Layer Mechanism for QoS Support in Wireless Sensor Networks”, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006, pp. 118, 23-29 April 2006.

[21] Crossbow Technology Inc.TelosB Datasheet: http://www.willow.co.uk/TelosB_Datasheet.pdf (accessed on 30 May 2012).

[22] A. Ludovici, P. Moreno, A. Calveras, “Embedding RESTful Web Services in Wireless Sensor Networks through CoAP”, unpublished.

[23] IEEE, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). In IEEE Standard 802.15.4-2006. Part 15.4; IEEE Computer Society: Los Alamitos, CA, USA, 2006.

[24] O. Gama, P. Carvalho, J. Afonso, P. M. Mendes, “Quality of service in wireless e-emergency: main issues and a case-study,” In proc. of 3rd UCAmI, Salamanca, Spain, Oct. 2008