A Living Smart City: Dynamically Changing Nodes Behavior Through Over the Air Programming

Jose A. Galache, Pablo Sotres, Juan R. Santana, Veronica Gutierrez, Luis Sanchez, Luis Muñoz
Network Planning and Mobile Communications Laboratory
Universidad de Cantabria
Santander, Spain
{jgalache,psotres,jrsantana,veronica,lsanchez,luis}@tlmat.unican.es

***Abstract*— The Future Internet Research and Experimentation (FIRE) initiative aims at achieving experimentation and testing in large-scale environments, through the creation of a multidisciplinary research environment for investigating and experimentally validating highly innovative and revolutionary ideas for new networking and service paradigms. SmartSantander FP7 project aims at the creation of an experimental test facility for the research and experimentation of architectures, key enabling technologies, services and applications for the Internet of Things (IoT) in the context of a city. In this sense, this paper presents the deployed facility, emphasizing the capacity of experimenting over this large-scale testbed, through the reprograming of the deployed IoT devices with different code images. For this purpose, the implementation and validation of an Over the Air programming (OTAP) scheme has been carried out, coexisting with the service provision and experimentation ability simultaneously offered over the deployed facility.**

Keywords- FIRE; IoT; OTAP; Smart City.

I. INTRODUCTION

Nowadays, it is a fact that present and future wireless technologies will be predominantly impacted by the massive deployment of IoT devices. Hence, it is mandatory to make available to the research community, infrastructures which allow analyzing and assessing the performance of mechanisms aiming at integrating IoT world in the Future Internet (FI) infrastructure.

Smart Cities stand as the meeting point to align the creation of large scale experimentation facility fostered by FIRE initiative, and the user involvement in ICT-based innovation targeted by the Living Labs community. In this sense, Wireless Sensor Networks (WSNs) place as key-enabling technology to gather information associated to these smart environments.

The SmartSantander project [1] aims at the creation of an experimental test facility for the research and experimentation of architectures, key enabling technologies, services and applications for the Internet of Things in the context of a city. The envisioned facility is conceived as an essential instrument to achieve the European leadership on key enabling technologies for IoT, and to provide the European research community with a one-and-only platform of its characteristics, suitable for large scale experimentation and evaluation of IoT concepts, under real-life conditions.

In order to cover this twofold approach [2] (experimentation and service provision), deployed nodes within SmartSantander project must present, not only the aforementioned characteristics associated to WSNs, but also they must be able to be remotely flashed with different code images in order to run different experiments and services. This remote programming is known as Over The Air Programing (OTAP) or multihop OTAP, called (M)OTAP, for nodes more than one hop away from the source.

This paper describes the implementation and operation of a (M)OTAP protocol over a real deployment carried out in the city of Santander, within the SmartSantander project framework.

The paper is structured as follows: Section II presents the related work to the existing OTAP implementations. In section III, it is shown the architecture and deployment on which the performed implementation is developed. Section IV describes in a detailed way the implementation carried out, showing in section V the corresponding measurements that assess and validate the correct operation of the implemented protocol. Finally, Section VI indicates the main conclusions derived from this work.

II. RELATED WORK

In order to make WSNs as dynamic and flexible as possible, several protocols for remotely programming nodes over the air have been proposed. XNP [3] was the first network reprogramming protocol proposed by TinyOS, for WSNs. It operates only over a single hop and does not support incremental updating of the program image. In order to support multihop over-the-air programming, it was developed MOAP [4], implementing a simple windowed retransmission tracking scheme. Unlike XNP and MOAP, in [5] an energy-efficient code distribution scheme to wirelessly update the code running in a sensor network is shown. In this sense, only the changes to the currently running code are distributed.

Manufacturers have also developed their own solutions, such as BitCloud OTAU [6], which implements the support of over the air upgrade in ZigBee networks on top of the Atmel BitCloud, a full-featured ZigBee PRO stack; or

Jennic OAD [7] that allows flashing Jennic nodes, but only in a unicast fashion and for nodes one-hop away from the source.

Currently, more complex and efficient protocols are being implemented; those like MNP [8] presents an energy efficient protocol, based on reducing the problem of collision and hidden terminal, trying to guarantee that in a neighborhood there is at most one source transmitting the program at a time. SYNAPSE [9], on the one hand, presents a protocol based on Fountain Codes featuring a hybrid ARQ (HARQ) solution, where data are encoded prior to transmission and incremental redundancy is used to recover from losses, thus considerably reducing the transmission overhead. On the other hand, Deluge [10] bases on a density-aware, epidemic maintenance protocols for propagating large data objects from one or more source nodes to many other nodes over a multihop wireless sensor network.

From the operation point of view, different protocols have been presented, both unicast and multicast approaches, based on fountain codes and on different transmission schemes, or solutions associated to different manufacturers such as Jennic and Atmel. As a commonalty, most of these protocols, based on measurement testbeds bounded to determined grids and configurations, composed of a reduced set of nodes, turning to simulation tools for complex topologies and large quantity of nodes.

III. SmartSantander Architecture

Within SmartSantander project framework, around 3000 sensor nodes have been deployed in several zones of the city of Santander [11]. Among these nodes, environmental sensors measuring temperature, humidity, pressure, light, noise, CO, as well as parking sensors, have been deployed, with their corresponding gateways. Taking into account the large number of sensors installed, they have been grouped in several clusters in order to manage them in an easy way. In this sense, gateway will act as head of the cluster gathering and processing information retrieved from the nodes hanging from this gateway.

As described in [2], all the deployed gateways (always powered) and repeaters (fed with rechargeable batteries) are intended to support both experimentation and service provision, unlike to parking sensors that due to their battery constraints are not intended for running experimentation and being flashed over the air.

The simultaneous service-provision support is carried out through two independent radio interfaces: one 802.15.4 native interface for transmission/reception of data associated to the experiments, and another 802.15.4 interface implementing Digimesh routing protocol, for sending information derived from service provision (environmental and car presence) and network management (transmission/reception of commands and (M)OTAP).

Taking into consideration this independence between service and experimentation operation, together with the fact of being able to flash nodes over the air, gives the network the capacity to offer a twofold flexibility in terms of:

- Service provision: Different services can be implemented on the same nodes, just modifying the code installed in a determined set of them in order to fulfill different requirements associated to a certain service.
- Experimentation: Different routing protocols, data mining techniques, network coding schemes can be tested on the deployed network, flashing nodes accordingly.

Apart from the experimentation and the service provision joint approach, network can be managed by sending commands to the nodes, as well as changing their behavior (through (M)OTAP) according to application requirements.

IV. (M)OTAP Operation

In order to make deployed platform as dynamic and reconfigurable as possible, a reliable (M)OTAP protocol has been implemented for flashing nodes over the air either in unicast, multicast or broadcast fashions, as many times as needed. The protocol implemented in this paper has been developed over nodes provided by the Spanish company Libelium [12]. Although Libelium offers its own (M)OTAP protocol, as most of existing solutions, it is mainly constrained to small deployments within reliable and non-interferring environments. In this sense, firstly in order to adapt the protocol operation to the deployment in the city of Santander, providing a reliable operation in outdoor dense deployments, as well as to fulfill main project objective (service-provision duality), a new scheme has been proposed and implemented, as shown in Fig. 2.

Before explaining the (M)OTAP protocol itself, it is important to describe the importance of the commServer module that consists of a port multiplexer/demutiplexer, offering several virtual communication ports towards a one physical serial port.

Fig. 1 shows from a bottom up perspective, two available physical nodes in each gateway for receiving packets associated to experimentation (ttyUSB0), as well as those associated to the provided services and the network management (ttyUSB1). The commServer module offers several virtual ports, ones for experimentation that take data from the physical experimentation port (ttyUSB0), others for different services and, in this case, one associated to network management (OTAP), taking data also from the physical service port (ttyUSB1). Through this virtual port, all data related to flashing procedure is transmitted/received through the OTAP virtual port.

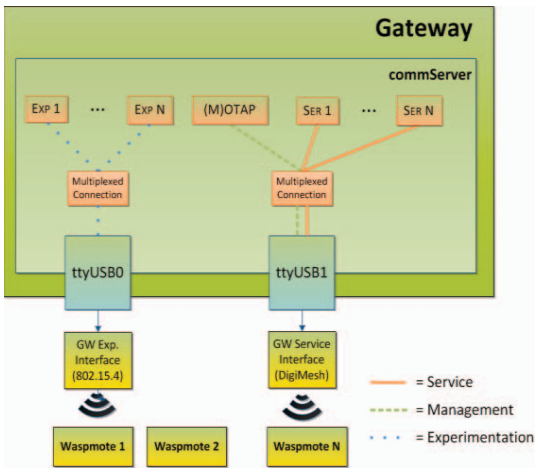

Figure 1. CommServer Architecture

Regarding to the OTAP protocol, it is implemented at both server and client sides. OTAP server is implemented in all the gateways (called as meshliums [12]), whilst OTAP client is implemented in all repeaters, which are composed of a waspmote node [12] and two Xbee [13] radio modules. For (M)OTAP operation, the code images to be flashed, are stored in the Meshlium and sent (using the OTAP virtual port), towards the corresponding nodes, through the following process:

1) (M)OTAP begins sending a start frame, including the corresponding OTAP key (used for security issues), so as to inform the node/s to be flashed, the beginning of a reprogramming process. For multicast communications, in order not to disrupt, both service provision and experimentation of nodes not affected by reprogramming process, it is sent a former message to the target nodes for changing the OTAP key to be used for flashing them, thus not reprograming nodes not involved in flashing procedure. Other parameters included in the start frame are the number of fixed-size fragments (chunks) in which the server will divide the image to be flashed, the program name and the compilation date. Each node will process this start frame and send back an acknowledgement, entering in a reprogramming state that will last until the whole program has been correctly received or an error occurs (erroneous access to SD card, reply timeout exceeded, etc).

2) Once the server receives the acknowledgement, it starts sending the image towards the destination node/s. Taking in consideration the big constraints in terms of internal memory (8KB), and that the average size of a code running on the waspmote occupies ranges around 5KB (depends on the functionalities and libraries used), the code must be stored in an additional memory. Waspmote includes the possibility of connecting a 2 GB SD card for extra storing. This card will be used by nodes to store the received code.

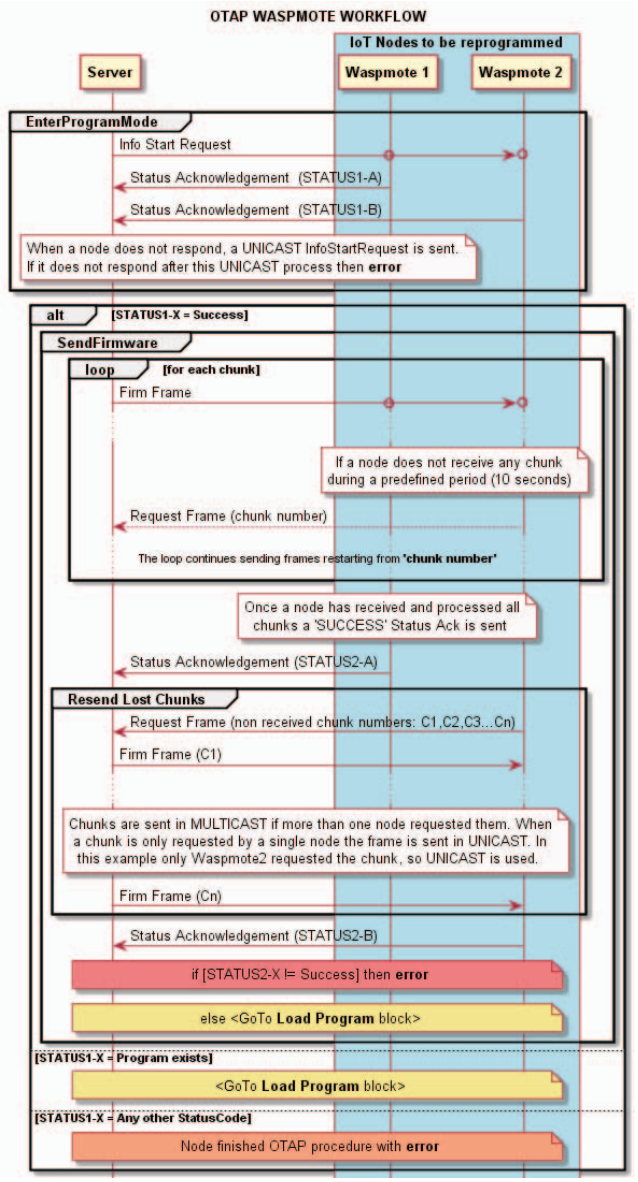

Figure 2. OTAP procedure

3) Server sends chunks separated by a determined time interval, whose value will depend on the type of reprogramming process; either unicast or multicast/broadcast. If ACK confirmation at MAC layer (provided by Digimesh protocol) is not received within this period, then server will wait until the reception of the ACK within a fixed timeout. If not received or a route discovery error is thrown, the server will retransmit the fragment again a determined number of times. Nodes will allocate the program in a determined position of the SD card. This will allow them to process out of order packets. This is very important in multicast/broadcast communications, as a node can lose a fragment, but the remaining nodes can have

received it correctly, thus receiving it twice if the source node had to retransmit it, making the protocol less efficient.

4) Nodes will request frames in two situations: the first one occurs when a node has not received a frame from the last 10 seconds, thus forcing the server to retransmit remaining frames from this last frame. Second one occurs when server has finished the transmission of all packets, and nodes ask for retransmission of lost chunks (which ID has been previously recorded in a linked list). Server will send again the lost packets in a unicast or a multicast/broadcast way depending on the type of reprograming, unicast or multicast/broadcast respectively. In order to improve the protocol performance, for multicast/broadcast transmissions, if only one node has lost a specific fragment, the server will retransmit it in a unicast fashion.

5) Once new code image is stored in the SD card, another command must be sent from the server to start running the new program in the node.

Considering the whole OTAP process and the different components involved, main advantages comparing to the previous work carried out by Libelium can be summarized as follows:

- **Over the air reprogramming, service provision and experimentation under the same cluster simultaneously**: Commserver component allows server node to flash a set of nodes at the same time that it continues receiving, both service and experimentation data, from the remaining nodes of the cluster. Furthermore, new virtual serial ports can be easily introduced to deal with new possible applications working at the server side. On the contrary, previous OTAP utility forced to suspend the service provision while reprogramming nodes.
- **Packet loss-aware design**: Former OTAP design does not consider recovery mechanisms from packet losses, being not efficient under interfering environments. In order to solve it, approach developed in this paper allows the retransmission of lost packets at the end of the OTAP process, which improves performance as lost packets can be retransmitted in multicast fashion if there is more than one node requesting them.
- **Disordered packets management**: Mainly in multicast flashing procedures, some nodes (especially those more than one hop away), could receive disordered packets. Former protocol only deals with a maximum of two packets out of order, whilst new approach allows processing any number of disordered packets, discarding those received twice.
- **Concurrent OTAP and service/experimentation management at node level**: As previously commented, both service provision and network management run over the Digimesh interface, whilst experimentation data is transmitted over 802.15.4 interface. This translates into the fact that node has to attend to both interfaces, in order to manage the two data types. Considering that the waspmote uses a mono-threaded processor, watchdog interruptions from the microcontroller are used, thus checking periodically if it has been received a command (start of reprograming process) on Digimesh interface, at the same time as service and experimentation frames are sent.

V. MEASUREMENTS

Besides the outdoor scenario, and in order to avoid uploading erroneous or malicious codes to the outdoor nodes, an indoor scenario has been also installed within the University of Cantabria premises.

Fig. 3 shows, on the left side, the outdoor scenario composed of nodes belonging to cluster managed by Meshlium 6 (installed in the city centre). This Meshlium composes of 27 nodes. On the right side, the indoor testbed groups 15 nodes (connected through USB wires to feed their rechargeable batteries and for debugging issues), placed in the ceiling of different rooms and corridors at University premises.

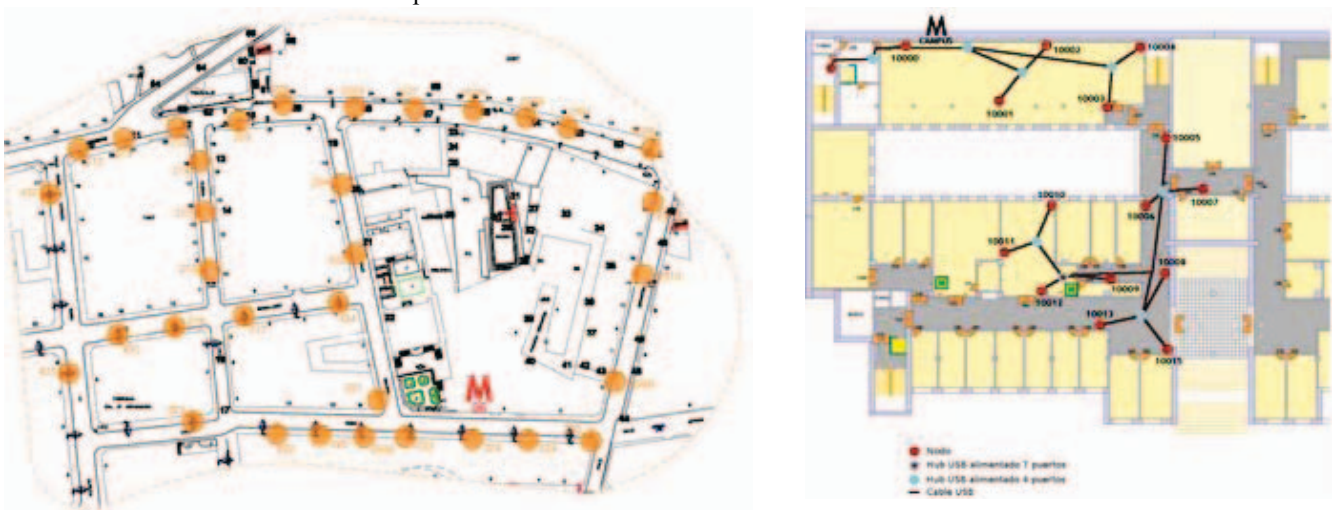

Figure 3. Outdoor scenario (l) and Indoor Testbed (r)

Regarding to the measurements, they must be taken into consideration the following issues:

- Fragment size: Maximum size of a fragment sent in a Digimesh communication is 100 bytes, 27 of them are Digimesh header, whilst the remaining 73 bytes are used for storing payload information. In our case, from these 73 bytes, 6 bytes associates to waspmote API, 2 to data length and 8 to OTAP key, so finally just 57 bytes of each packet contains information on the code image to be flashed.
- Data rate: Waspmote nodes work at a maximum data rate of 38.4 Kbps.
- File size: Files of different sizes ranging from 50 KB to 90 KB (60 KB, 70 KB and 80 KB) have been selected, as they are the typical code image sizes.
- Route establishment: Digimesh [13] is a proprietary routing protocol that allows installing/removing a node in/from a network in a transparent and reliable way. As a proprietary protocol, it does not offer information about the obtained routes, but they can be inferred the number of hops distance from a node to the meshlium. For this purpose, it is used the broadcast radius parameter, that limits the number of hops a broadcast communication can reach. Gradually increasing the value of this parameter, nodes at different number of hops are discovered. TABLE I. shows the hops distance of the nodes in the indoor and outdoor testbeds.

TABLE I. HOPS DISTANCE IN INDOOR/OUTDOOR SCENARIOS

Scenario	*Hops*	*Nodes*
Indoor (CAMPUS)	1	10000, 10001, 10002, 10003, 10004, 10005, 10007, 10009, 10010, 10011, 10012
	2	10006, 10008, 10013, 10015
Outdoor (M06)	1	221, 222, 223, 225, 226, 228, 229, 230, 434, 2505, 2510
	2	208, 209, 214, 215, 216, 217, 218, 227, 231, 430, 431, 2506, 2508, 2509
	3	210, 432

It must be taken into consideration that, for sevral nodes, distance in terms of hops oscillates between 1 and 2 hops or 2 and 3 hops, depending on the network conditions. It is also important to highlight that, as number of hops is being calculated through a broadcast communication, the influence of routes calculated by Digimesh protocol is avoided.

- Information encryption: In order to preserve the information transmitted among the nodes, communications have been encrypted using AES128 protocol.
- Interval between fragments: Considering that no flow control, neither hardware nor software, is implemented on the gateway side, it is included a time interval between the transmission of consecutive frames. In order to estimate this interval time, they are considered the following parameters for radio modules:

  NN (Network Delay Slots) = 3
  NH (Network Hops) = 7
  RR(Mac Retries) = 3

unicastOneHopTime (for RR = 3) = 189 ms

Measurements indicated beforehand consider the worst conditions in the transmission procedure. After characterizing the measurement scenarios, there were selected a time of 150 ms for unicast transmissions (a bit lower than unicastOneHopTime) and a bigger one of 250 ms for multicast/broadcast transmissions (possibility of error/collision is greater). It must be indicated that these are minimum intervals, so if the ACK (at Digimesh level) associated to a fragment, is received in a higher time, then this time will be the interval among fragments; otherwise it will be used the fixed interval.

- Simultaneous service/experimentation: At the outdoor scenario all nodes are sending data (around 35 byte length packets, depending on the type of service) associated to the environmental monitoring and parking management services, each 5 minutes. At the indoor testbed, service packets are being sent each minute (instead of each 5). During the (M)OTAP process, server will keep on receiving, both experimentation and service data, from nodes not involved within the flashing process.

Over the described scenarios and considering the aforementioned considerations, there have been carried out different measurements on different parameters, latency, throughput and Packet Error Rate (PER) for characterizing the implemented protocol:

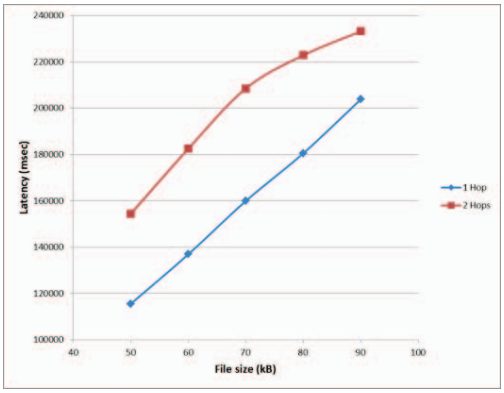

Figure 4. Indoor latency measurements

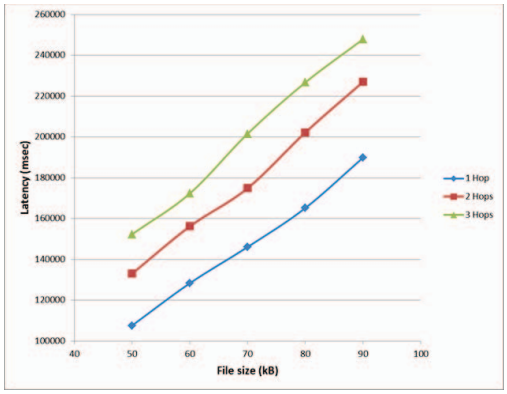

Figure 5. Outdoor latency measurements

In Fig. 4 and Fig. 5, it is shown the latency and throughput associated to nodes from both outdoor scenario (Meshlium 6) and indoor testbed (CAMPUS). These measurements have been made in a unicast way, sending files of different size to all nodes within both scenarios, calculating the average values as the representative ones to characterize the protocol operation. From these figures, it can be observed that values in outdoor scenario are slightly lower than in indoor one, but pretty similar for the different file sizes and number of hops. Regarding to the throughput value, the larger the file, the bigger the throughput value (mainly for several hops where the influence of headers is more negligible). In this sense, as an average of the throughput for all different file sizes, values obtained are 3.7 Kbps, 3 Kbps and 2.7 Kbps, for 1, 2 and 3 hops respectively. To compare these values with those offered by Digimesh documentation, it is calculated the estimated rate through:

$$Est_{DigimeshRate} = Est_{PayloadRate} \frac{Digi\_Payload}{Data\_Payload} \quad (1)$$

TABLE II. DIGIMESH ESTIMATED THROUGHPUT FOR 38.4 KBPS

Hops	*Theoretical*	*Estimated*
1	6.83	5.64
3	3.27	3.45

As derived from TABLE II. ,throughput for 1 hop is lower than theoretical one, but for 3 hops it is a bit higher. This can be due to the fact that the two nodes at 3-hop distance behave in an unstable way oscillating their distance between 2 and 3 hops.

Regarding to the packet error rate, considering that network topology is over-meshed in order to assure no disruption in service provision, the number of erroneous packets received can be considered negligible in terms of protocol operation (maximum of 6 packets in some transmissions at 3 hops), in both outdoor and indoor environments. In this sense, service running at the same time as MOTAP process is running does not produce considerable packet losses. A high demanding service and/or experiment running on the nodes not involved in flashing procedure could imply a worse performance of the protocol.

In order to compare unicast measurements with multicast/broadcast ones, multicast flashing has been made to three nodes (at 1, 2 and 3 hops),just getting an average value of 330 s (70 KB size file), which means a gain of 40% compared with separated flashing procedures. Regarding to broadcast flashing, 15 nodes of indoor testbed have been flashed with a 70 KB size file, lasting an average of 350 s (similar to multicast as no 3 hops distance node is involved), which translates in around 10 % of the global time for separated flashing procedures.

VI. CONCLUSIONS

Unstoppable growth of WSNs joined to the adaptation to the market of new technologies, has led IoT to become an integral part of the FI. Under the SmartSantander project umbrella, applications and services for the smart city are to be deployed and used by real users, as well as a facility for realistic IoT experimentally-driven research is provided.

In order to make the research testbed and service provision facility, as flexible and dynamic as possible, this paper presents a (M)OTAP protocol to remotely flash nodes as many times and with as many code images as needed, in unicast, multicast and broadcast fashions. This translates into the continuous adaptation of provided services according to users' feedback, as well as, the capacity of running different experiments in a set of nodes within a real scenario.

It is important to highlight that, unlike current OTAP solutions, which are bounded to laboratory-tailored scenarios with a small number of nodes located according to determined topologies; the protocol presented in this paper has been tested in a real deployment (with real traffic associated to service provision), within a smart city environment. The implemented protocol has been characterized through different measurements on latency, throughput and packet error rate, in both an indoor test deployment, as well as an outdoor real scenario.

ACKNOWLEDGMENT

This work is partially funded by research project SmartSantander (http://www.smartsantander.eu), under FP7-ICT-2009-5 of the FP7 of the European Community.

REFERENCES

[1] SmartSantander project website. www.smartsantander.eu

[2] J. A. Galache, Juan R. Santana, Verónica Gutiérrez, Luis Sánchez, Pablo Sotres and Luis Muñoz. "Towards Experimentation-Service duality within a Smart City scenario", presented at Wireless On-demand Network Systems and Services (WONS 2012), Courmayeur, Italy, Jan. 9-11.

[3] J. Jeong, S. Kim, and A. Broad, "Network Reprogramming", Berkeley, California, USA, Aug. 2003. [Online]. Available: http://www.tinyos.net/tinyos-1.x/doc/

[4] T. Stathopoulos, J. Heidemann, and D. Estrin. "A remote code update mechanism for wireless sensor networks". Technical report, UCLA, 2003.

[5] Niels Reijers and Koen Langendoen. Efficient Code Distribution in Wireless Sensor Networks. In Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, pages 60{67. ACM Press, 2003.

[6] BitCloud OTAU User Guide. http://www.atmel.com/Images/doc8426.pdf

[7] Jennic OAD Application Note. http://www.jennic.com/files/support_documentation/JN-AN-1057-Over-Air-Download.pdf.

[8] S. S. Kulkarni and L. Wang, "MNP: Multihop Network Reprogramming Service for Sensor Networks," in IEEE ICDCS, Columbus, Ohio, USA, Jun. 2005.

[9] Rossi, M., Zanca, G., Stabellini, L., Crepaldi, R., Harris, A., Zorzi, M.: SYNAPSE: A network reprogramming protocol for wireless sensor networks using fountain codes. In: Proc. of SECON, San Francisco, CA (2008) 188{196

[10] J. W. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," in ACM SenSys, Baltimore, Maryland, USA, Nov. 2004.

[11] SmartSantander deployed facility in the city of Santander. http://maps.smartsantander.eu/

[12] Libelium. Wireless Distributed Communications. http://www.libelium.com/

[13] Digi. Wireless machine-to-machine (M2M) device networking products. http://www.digi.com/