Architecture, functional requirements, and early implementation of an instrumentation grid for the IoT

Andrea Azzarà, Daniele Alessandrelli, Stefano Bocchino
Retis Laboratory - TeCIP institute
Scuola Superiore Sant'Anna
Pisa, Italy
{*a.azzara, d.alessandrelli, s.bocchino*}*@sssup.it*

Paolo Pagano, Matteo Petracca
National Laboratory of Photonic Networks
Consorzio Naz. Interuniv. per le Telecomunicazioni (CNIT)
Pisa, Italy
{*paolo.pagano, matteo.petracca*}*@cnit.it*

***Abstract*—This paper presents an innovative architecture for the instrumentation layer of a pervasive large-scale information system. State of the art examples already include an instrumentation layer in a grid infrastructure; nontheless these systems were not designed for large-scale ad-hoc networks, rapidly deployed, continuously upgraded, and dynamically reconfigured when the system is on-line. The main contribution is that of designing an instrumentation layer coherent with existing infrastructures capable of being federated with others, and already providing robust virtualization services in what concerns user authentication and access, distributed storage, and computing power. The large-scale instrumentation layer is inspired by the Internet of Things, and will consist of a large number of autonomous nodes organized in Wireless Sensor Networks and accessing the Internet following the 6LoWPAN specifications. An early implementation of this architecture is proven to respond to use cases centered on data- and event-driven services; it also permits a large degree of flexibility and adaptiveness in end nodes applications.**

***Keywords*-Internet of Things; Ad-Hoc networks; grid (cloud) computing; virtualization; large-scale infrastructures; 6LoWPAN; CoAP**

I. Introduction

The Internet of Things (IoT) is the future envisioned extension of the current Internet which will include every embedded system, individually discovered and addressed as a resource in the network. The IoT will permit to remotely access such devices thus making available an enormous amount of data about the physical world which was previously inaccessible. Innovative applications can be developed exploiting such data and the new control possibility offered by the IoT.

Various aspects should be considered while designing novel applications leveraging pervasive, topological complex, and large-scale device infrastructures. Connectivity is the first issue and concerns the seamless internetworking of heterogeneous devices, usually operating in the Internet by means of access networks of diversified technologies ranging from copper wired, optical fibers, wireless cellular, wireless ad-hoc, etc. Other issues relate to cooperation and interoperability within a distributed environment, allowing for resource sharing to run global computation tasks.

The usual approach is that of virtualizing users, connections, and resources in a multi-purpose framework, commonly implemented as a middleware. Working solutions for providing transparent access to remote resources from full-fledged and nomadic devices (like PCs and smart phones) have been proposed by the research community and consequently ported to commercial services and final user applications: an example is given by cloud computing software, today available on many platforms.

Nonetheless PCs and smart phones are to be considered as personal equipment so that cloud computing is still confined to the Internet of People. The real challenge is therefore that of extending virtualization to the world of things, specifically to programmable sensors (notably Wireless Sensor Networks) and smart tags (notably RFIDs).

Over the past few years the research community has focused on designing protocols for the IoT. Indeed, protocols for the traditional Internet are often unsuited for embedded devices that are usually constrained in terms of computing power, memory, and network interfaces. The main outcome of this research effort is an adaptation layer for IPv6 allowing to transmit IPv6 packets over IEEE 802.15.4 networks. 6LoWPANs, i.e., an adaptation of IPv6 for low-power devices, have proven to be a valid alternative to traditional WSNs employing proprietary protocols [1]. Indeed, 6LoWPANs can compete with traditional WSNs in terms of power consumption and network throughput, while achieving a seamless integration with the Internet (and therefore a higher interoperability).

Another important step towards the IoT is the on-going drafting of the Constrained Application Protocol (CoAP), an HTTP-like protocol especially designed for constrained devices. CoAP permits to create embedded web services

©2012 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

running on IoT nodes [2], thus extending the successful web architecture, based on the REST paradigm, to the IoT. The basic idea is that IoT nodes are responsible for one or more resources that may represent sensors, actuators, combinations of values or other information. Internet devices can send messages to change and query resources on IoT nodes. IoT nodes can also send notifications about changed resource values to devices that have subscribed to receive notification about changes. An IoT node can also publish or be queried about its resources.

The major power of the Web is to allow the flowing of information through different devices. This result is achieved using a series of common components: a language (the hypertext transfer protocol, HTTP), a scheme for resource identification (the Uniform Resource Identification, URI) and some content descriptions (the Internet Media Type, MIME). The Web is generally accepted as a mean for applications running on a plethora of devices to interoperate, sharing data and resources; in such respect CoAP is responding to the primary objective of supporting a "web of things", by providing a set of services useful for a variety of final user applications. As HTTP, CoAP is a working solution for supporting machine-to-machine (M2M) communication, in the context of Web Services and Semantic Web Services [3].

Technologies like 6LoWPAN and CoAP enable for large scale, ad-hoc networks of things, fully interoperable with full-fledged devices by means of Web services. As the high-end ones, these low-end devices are expected to be fully virtualized in respect of their resources, the running application, and their capability of building up high level information from raw data via cooperation patterns.

IPv6 technology is promoted by strategic research initiatives funded by the European Commission and the National Science Foundation; it plays a key role in the so-called "Future Internet" which envisions the deployment of frontier infrastructures based on the Web in domains like Intelligent Transport Systems (ITS), Smart Ambients (for instance buildings, factories, and energy plants) and Smart Grids.

For instance the societal objectives in the Horizon 2020 programme [4], should enable for (i) "the transition to a reliable, sustainable and competitive energy system, in the face of increasing resource scarcity, increasing energy needs and climate change", (ii) "a European transport system that is resource-efficient, environmentally-friendly, safe and seamless for the benefit of citizens, the economy and society".

The ITS case study will be further investigated in this paper, organized as follows: Section II discusses related work. Section III provides a set of Use Cases roughly defining the requirements of the target system. Section IV describes the system architecture we want to prototype. Section V presents the technical implementation details of the system. In Section VI the experimental validation of the system is discussed. Finally Section VII provides concluding remarks and outlines future work.

II. Related Work

The research community has already proposed a plethora of middleware solutions to hide the complexity of sensor networks, thus simplifying the development of applications on top of them. Most of the proposed solutions can be categorized either as data-centric or as event-centric [5]. Data-oriented techniques are those inherited from Data Base Management Systems, aimed at extracting high level information from elementary records organized in abstract containers (e.g. tables) whose structure, relations, and dependencies are expressed in the data base schema. Usually the protocols are initiated by a query, the network activity performs aggregation, and the high-level information is routed to high-end control points. In event-centric middleware, queries can alternatively be implemented as triggers and produce event notifications in real-time. Events can also provoke actions; an action can be any user-defined function (e.g. a "MailTo" action), a set-point adjustment in a control system, a robotic actuation.

Data- and event-centric middleware solutions simplify the application development and allow for multi-pourpose sensor networks, but constraint the application to a predefined set of functions. Some applications require a higher control on the sensor network and especially on the on-board processing performed by sensor nodes. Such issue can be solved allowing the applications to change the code running on the sensor nodes. Remote code reprogramming is indeed a common feature of operating systems for sensor networks [6], [7]. Its major drawback is the code dependency on the software and hardware platform. To overcome this limitation, other solutions based on Virtual Machines (VM) have been proposed [8], [9], [10]. VM provide hardware abstractions: as hardware evolves new platforms become available. Using virtual machines the same application runs transparently in any platform supporting the interpreter. Virtual machine can also provide a safe execution environment, isolating the user application from the operating system space and preventing direct access to the hardware. The major drawback of the VM approach is the overhead of interpretation. However this issue is disappearing thanks to the emerging new generation of motes providing higher computational power and larger amount of memory, without sacrifying power consumptions [11]. Therefore, nowadays, using a VM is probably a better option than implementing dynamic linking on motes.

Middleware solutions and reprogramming techniques allow for multipurpose sensor networks which can easily support new applications. However, they usually do not solve the problem of interoperating multiple sensor networks. Indeed, this issue has been, so far, greately disregarded by the research community. A remarkable exception is Physicalnet [12], a framework for managing and programming

world-wide distributed heterogeneous sensor and actuator resources in a multi-user and multi-network environment. Physicalnet enables global uniform access to heterogeneous resources and decouples applications from particular resources, locations, and networks. Physicalnet main limitation consists in the lack of support for in-network processing. Moreover it was not designed to exploit the emerging IoT protocols.

Instead of extending Physicalnet to overcome its limitations, we prefered to adopt a different architecture with the final goal of integrating sensor networks with complex infrastructures based on grid technology. Indeed we believe that the grid is the perfect tool for storing and processing the huge amount of data produced by sensor networks. Moreover the grid already offers advanced mechanisms for managing concurrent applications and controlling user access to resources. We modeled our architecture following the DORII example [13]. Indeed the DORII infrastracture is the state of the art in integrating instrumentations with the grid and we believe that a sensor network is functually an instrumentation.

A. Contribution of this paper

The main strength of the IoT is interoperability, which, as already mentioned, will permit to develop innovative applications exploiting the new data becoming available and the extended control possibilities deriving from the interconnection of many embedded systems. The research community has succeeded in achieving such interoperability at the network layer (with 6LoWPAN/IPv6) while, at the application layer, the effort is still incomplete. Indeed, CoAP just defines the paradigm to interact with IoT nodes, while a complete system architecture for easily managing networks of IoT nodes is still undefined.

Leveraging the current IoT design, we propose an innovative architecture for a pervasive infrastructure of embedded systems, inter-connected and running a distributed logic: we call it "instrumentation grid". The embedded systems are functionally organized as a remote layer inter-operated with a set of applications hosted in fixed or mobile equipment, virtually realizing a control room sitting on the Internet. We developed an early implementation of our architecture, which permits to discover available resources, monitor sensor data, handle its storage, control actuators, change the preprocessing performed on nodes, define events and the actions to be performed when they are detected.

Although highly specialized literature about instrumentation grids exists [14], the progress with respect to the state of the art is related to the support of large-scale ad-hoc networks, rapidly deployed, continuously upgraded, and dynamically reconfigured when the system is on-line.

III. Use-Cases for large-scale ITS

Let us consider a frontier Intelligent Transport System allowing for travel planning, transport demand calculation and forecast, traffic management, emergency management, road pricing, and access to parking and transport facilities.

Within the ITS footprint, a certain number of fixed detectors (a.k.a. road sise units) and mobile devices (embedded in personal and vehicular on-board equipment) realize a distributed, heterogeneous, and dynamic collection layer in charge of building up information making use of cooperation patterns.

The collection layer is seamlessly interconnected with a back-end system providing network services and remote computing power. Applications are profiled for final users depending on their identity and their role in the ITS (i.e. citizens, road managers, public officers, safety regulators, system administrators, etc.). ITS applications can be designed in such a way to have a lightweight client running on every device (i.e. a PC in a control room or a smart phone in the hands of an operator) and remote services handling storage, computing, and I/O with the collection layer.

We will show how we must migrate to a virtualized architecture (as shown in Figure 1) when, responding to Horizon 2020 expectations, we want to deploy multi-user, multi-purpose, heterogeneous technologies-based ITS over a large scale.

Grid technologies are already providing well-established solutions for user profiling and access control (through the Virtual Organizations), distributed storage (through the data grid), and distributed calculus (through the computing grid). The Use Cases discussed in the next sections will focus on the virtual I/O showing the benefit of building up an instrumentation grid natively supporting open, scalable, and federated large-scale ITS.

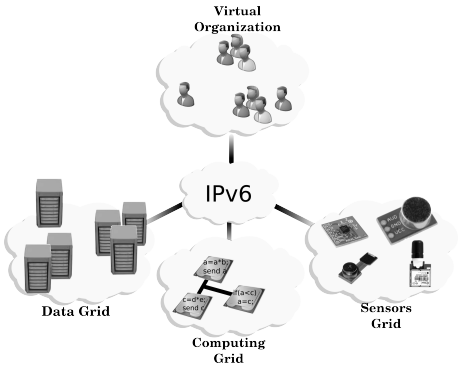

Figure 1. A schematic view of a distributed system providing virtual services to final users.

A. Data and event services

The nodes in the collection layer are heterogeneous; some of them embed cameras, others detect pollutant concentration. The ITS is designed in an open way so that the final

users and the applications set are expected to evolve when the system is already on-line.

Within this network, the nodes regularly send out the sensed variables to those peers which registered for a certain data service. The transmission can also be conditional upon the occurrence of certain events. In our scenario, the surveillance multimedia stream will populate the disk pool of the road operators, the pollutant contamination records will populate the disk pool at the Authority for the Environment.

When the collection layer capabilities are upgraded by adding a set of nodes detecting other metrics, lightweight web applications are capable of discovering these "new resources" and "new services" are eligible to consume these data for instance displaying or storing temporal series into the appropriate places. If in our example sensor nodes classifying the vehicles crossing a road section are installed, the vehicle flow will be recorded by the Transport Authority for on-line criticity management and for off-line mining.

Our goal is such that the only constraint to discover new resources is that they are expected to sit on the Internet and to host a CoAP-enabled network stack. We will show in Section VI that an event service based on CoAP naturally fits the energy constraints and the bandwidth limitation in ad-hoc infrastructures (like WSN) if in-network processing techniques are enforced.

B. Dynamic application deployment

Consider some sensor nodes equipped with video and audio peripherals, generic enough to be able to run a set of diversified functions. In this scenario the function associated to every node is not determined a priori during the deployment itself. A dynamic application deployment presents many advantages compared to a static approach, in terms of: *i*) time and resources necessary to plan the deployment and physically program the nodes; *ii*) flexibility and performance. The functions can be dynamically reassigned to optimize the efficiency of the system, calibrating the performance based on run-time observations. This operation can be easily automatized so that the system will result self organizing and robust with respect to hardware failures.

When the nodes are placed in the environment they do not have any specific task associated but have computation ad sensing resources. We assume that their position is known, either it has been defined statically during installation, or it has been obtained using localization algorithms. The resources are shared by the nodes to the Internet and the actual application deployment is done through a remote web interface on high end systems, which eases the process of selecting the application and sending it to the right node or set of nodes.

We consider the following two applications: *i*) vehicular traffic monitoring; *ii*) acoustic pollution measurement.

As an example of deployment consider the following queries to the system:

- "Install the traffic monitoring application on the nodes along X street, in such a way that the distance between the nodes running the application ranges from 50 to 100 meters."
- "Install the pollution monitoring application on the nodes nearby the major intersections of the city."

Having access to the list of resources both operations can be easily performed querying the database. The information needed as input is just the list of available resources and their geographic position. The output of the operation is a list of URIs identifying a set of computation resources; the application is in turn installed on top of the selected nodes.

C. Conditional behavior of sensor nodes

Suppose that we want to exploit flexibility and adaptiveness in sensor nodes; a scenario requiring a combination of both is given by a collection layer composed by end devices expected to change their application (flexibility) when specific events occurr (adaptiveness).

In this case in our collection layer a set of nodes is equipped with microphones. Ordinarily (as discussed in section III-B) these nodes perform acoustic pollution measurements. Anyway they are configured to detect (eventually collaborating with each others) a certain kind of events (as discussed in section III-A). Whenever an event is detected they download a new application from the Point of Service and run the alternate function.

In our example, a set of nodes equiped with microphones is suited to perform an emergency-type application following the detection of an accident. The downloaded application will permit to stream away the audio from the accident zone to allow rescuers to take action appropriately.

As a realization of the last two Use Cases, we will show in Section VI that we are capable of transmitting executable bytecodes over a complex topology. This fulfills the requirement of large-scale ad-hoc networks where Points of Service (managing the versions of the application code for end devices) can be far away from destination.

IV. System Architecture

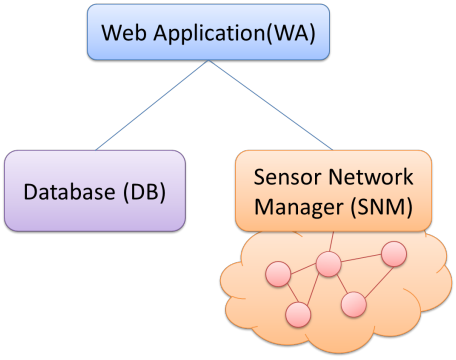

Figure 2. Our proposed architecture.

The system architecture depicted in Fig. 2 is composed by a set of heterogeneous nodes, each one implementing an IP communication stack, allowing direct communication between each peer. Every node is composed by computation and communication resources which are abstracted and shared with other peers in the network making use of the REST communication paradigm. The network layer interconnects transparently both high-end and low power devices, the latter forming resource constrained subnetworks.

More precisely the system components are: a Web Application (WA), a database (DB), and a Sensor Network Manager (SNM) for each sensor networks. The WA runs on a web server and provides users with all the tools they need to control sensor networks and build application on top of them. By simply connecting with a web browser to the WA, users can: *i*) list and query the resources (sensors and actuators) available in the sensor network; *ii*) start the monitoring of a set of sensors (resource subscription); *iii*) access historical sensor data; *iv*) control actuators; *v*) change the on-board processing performed on sensor nodes; *vi*) define events to be detected; *vii*) define the actions (control actuators and/or change on-board processing) to be performed when an event is detected.

The SNM is the Point of Service to the sensor network. Its main task is to perform the translation and forwarding of messages between the resource-constrained environment and the Internet world. The core of the resource constrained network is a 6LOWPAN mesh network, composed of heterogeneous nodes. Nodes can implement different services, providing access to sensors, and actuators in a unified RESTful way.

In our vision of the IoT, by bringing IP connectivity to resource-constrained devices, it is possible to realize simple M2M interaction, each node sharing the representation of their resources, sensors, and actuators. By running a VM all nodes share additional resources, such as memory and computing power. In our architecture the level of abstraction provided by the RESTful application layer is exploited to represent the tasks executing in the network as resources. Those resources can be created, deleted or moved from peer to peer, according to the availability of computation resources on the nodes.

From the application point of view, the SNM keeps track of the nodes and resources available in the 6LOWPAN and updates the related information stored in the DB. Moreover, it manages the sensor data collection and the event detection. If the user subscribes to some sensor data, the SNM monitors the proper resources for changes and adds new sensor values to the DB every time they become available. Events are handled in a similar ways. When the user defines a new event, the SNM starts monitoring the proper resources. Every time a resource changes the SNM checks whether the event is verified, and, in such a case, it adds an event notification to the DB and performs the actions defined by the user, if any. An action can be as simple as an actuator control signal or as complex as a script to be executed on the sensor node.

The DB is used to store historical data, such as event notifications or sensor data, and to keep track of the system status. It indexes the resources available in the sensor networks and it contains the definitions of the events currently monitored, as well as the associated actions to be performed when the event is verified. It also keeps track of the subscriptions currently active and of the scripts installed on the nodes.

As already mentioned, the proposed architecture is inspired by that of the DORII infrastructure. Indeed, as a future work, we plan to integrate sensor networks to the grid infrastructure. To this end, our architecture can be easily mapped to the DORII one: each sensor node can be represented with an Instrumentation Element, such abstraction will be implemented by the SNM, which will be transparent to the user; the tasks currently performed by the SNM (like event detection) can be assigned to Computing Elements, which can be used also for analyzing the acquired sensor data; the functionality of the WA can be provided by the Virtual Control Room server; and the DB can be virtualized with Storage Elements. The integration with the Grid Infrastructure have many advantages. First of all we can virtualize our resources improving robustness and performance. Moreover we can leverage the security and user management mechanisms used by the grid.

V. BUILDING THE PUZZLE UP: MODULES AND SERVICES

A. Modules

In this section we detail our implementation of the modules of the architecture. All the software components are open-source and most of them are based on well-established technologies.

1) Sensor Network Nodes: Every node in the collection layer is a fully Internet enabled device. Moreover nodes run a COAP web server, giving RESTful access to their resources. To realize virtualization services we used PyMite, a lightweight Python interpreter written to execute on microcontrollers with limited resources. PyMite supports a subset of the Python 2.5 syntax and access to native C functions; Python scripts can be compiled, tested and executed on a desktop computer before the actual deployment. Using the native C function support we can provide full access to the software modules of the host operating system, including the communication stack. User-defined scripts can then communicate with other resources on the Internet, performing in-network processing of data received from other sensors, thus reducing the transmission cost (measured in number of messages) following the event detection.

Every end-node of the 6LOWPAN network periodically sends a message to the SNM, notifying its presence. The

SNM can perform resource discovery, retrieving the list of resources available on the host, and updating the database.

2) Database (DB): The Database guarantees the persistence of the information gathered from the instrumentation grid and of the information needed to manage the grid itself. In the database we keep track of:

i) connected hosts; *ii*) available resources, their URIs and the methods supported; *iii*) the messages exchanged; *iv*) subscriptions to the resources; *v*) event handlers; *vi*) high-level scripts which can be dynamically executed by the nodes. We used a simple MySQL DB in our experimental deployment. Other technologies can be used and the DB can be distributed to address scalability issues.

3) Web Application (WA): The Web application provides to the users an interface with all other components, so that: *i*) it allows a platform independent access with little resource requirements to the final user; *ii*) it can be easily extended to provide new services and to interact with other applications through public APIs; *iii*) it is scalable by nature and can be distributed in cloud infrastructure.

We implemented the application using Django, an open source web application framework, written in Python. The application provides a representation of the resources based on their URIs and an interface to perform common operations on them. Actually we defined four different classes to represent respectively the sensors, the actuators, the virtual machines and generic resources. For each specific class we can provide different services, e.g., an on-line processing function on data retrieved on a sensor resource. The application interacts with the SNM using tasks distributed using the AMQP protocol [15]. The tasks are instantiated as a result of user's actions and then allocated to the SNM. Typically they consist of resource representation retrieval, resource observation, resource creation, e.g., the deployment of a new application. This approach enhances the scalability of the system: tasks can be allocated to different SNM, each one managing a different network, in a transparent way. Tasks are asynchronously executed in respect of the web application and the result of the operation is stored in the DB.

4) Sensor Network Manager (SNM): We implemented the SNM on a Linux machine connected to an embedded board, the gateway, which performs routing and translation from and towards the 6LowPan network. The Linux system is responsible for managing the communication, at application level, with the CoAP servers running on the nodes. A task manager is employed to run the tasks spawned by the web application. The tasks can include both CoAP clients and servers monitoring the 6LowPan network, managing subscriptions to resources, storing the information obtained from the DB. The SNM is also responsible for the event handling: when an event is notified, the SNM checks whether any handler is associated with the event and eventually executes it. From the architectural point of view we consider the task manager and the gateway as a single system. Actually we plan to realize the SNM on a single Linux embedded platform having an interface towards the 6LowPan network.

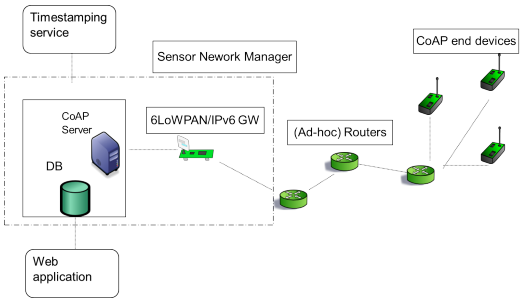

Figure 3. Experimental setup.

B. Services

We implemented the following services in our deployment: *i*) sensor network monitoring; *ii*) data storage; *iii*) dynamic re-tasking of existing virtual machines; *iv*) event notification and automatic event handling.

The sensor network monitoring service allows the users to see the list of resources available on the network. The resources can be accessed through the standard methods provided by RESTful environment (i.e., GET, PUT, POST, DELETE). Resources can be queried with a polling mechanism, performing the GET method periodically. An additional monitoring service is provided by the CoAP Observe protocol, which allows users to subscribe to a specific resource, and to be notified when the resource changes. The SNM saves in the DB every message received from the nodes: data can be analyzed both on-line and off-line.

The dynamic re-tasking service provides a way to rapidly change the application running on a certain host. Through the web interface, users can directly write scripts to be run on virtual machines, realizing a fully dynamic, re-configurable environment. The web application is responsible for creating the bytecode associated with the script and trigger the code transfer to the target VM resource. The state of a node hosting a virtual machine can also be monitored to know what script is actually running and the current state of the execution. Tasks can be created or copied from peer to peer. The automatic event handling service allows users to define some actions to be performed when events occur. We defined two types of handlers performing different actions. In the first case a simple REST operation is performed on a resource. In the second case a script, previously defined by the user, is sent to a resource to be executed. Other handlers, more complex than the ones implemented by us, can be easily included in the automatic event handling service.

VI. Experimental validation

This section is aimed at validating some of the requirements for the instrumentation grid we defined through the use cases discussed in Section III. Through two simple

experiments we want to prove the effectiveness of the architecture, described in Section IV and implemented as discussed in Section V. The first experiment will prove how our instrumentation grid is suited for monitoring events over large scale in a multi-hop topology making use of in-network processing techniques; the second experiment will show how we can dynamically reprogram the end-devices.

In both experiment our nodes are programmed with a real-time operating system, hosting a communication stack compliant with the IEEE802.15.4 specifications for MAC and Physical layers, the 6LoWPAN specifications for the Network layer, and the CoAP draft specifications for the application layer. The routing table is kept static.

In our set-up (shown in Figure 3) three nodes are the sensing peripherals in a WSN; they are connected to a 6LoWPAN/IPv6 gateway via wireless through a route which can be 1, 2, or 3 hops long. The nodes are also connected to the PC via the serial line to implement an event timestamping service making use of a single high resolution clock, that of the PC.

A. Complex event reporting

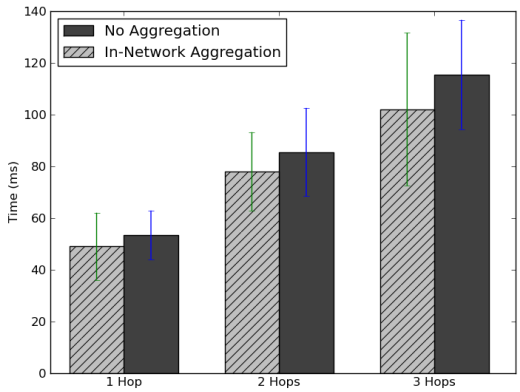

Figure 4. Event detection latency.

By means of this experiment we show how our implementation of the instrumentation grid fits the requirement of a large-scale collection layer having a larger number of detection points than Points of Service (gateways). In such a scenario it is possible that events are localized many hops far from gateways.

If the event is built up by combining the observations coming from more than one node (say, nodes A,B, and C), two options are possible: either the event composition is performed at the gateway (i.e. the gateway individually subscribes to nodes A,B,C and is in charge of combining the partial detections), or one of the nodes (say, node A) subscribes to the others and transmit upstream the event having filtered spurious (partial) detections and performed on-board event composition.

It is obvious that the message cost (in terms of exchanged messages) per unit information is higher in the case of centralized event detection than in the case where in-network processing logic is preferred; this cost increases as the distance between the detection point and the Points of Service gets larger because of the overhead coming from message routing.

The combination of the reports coming from the peripheral nodes in our setup defines the event. The event time is defined as the minimum timestamp given by the individual nodes participating in the detection:

$$T_{true} \;=\; min\;(T_A,\; T_B,\; T_C)$$

whereas the latency is given by:

$$\Delta_T \;=\; T_{SNM} - T_{true}$$

where T_{SNM} is measured at the gateway. In Figure 4 we plot the latency in event detection as a function of the number of hops. We show how the latency is statistically compatible in both the cases of "no aggregation" and "in-network processing"; the latter is the preferred solution for energy-related issues, peculiar of ad-hoc networks.

B. Reprogramming remote devices

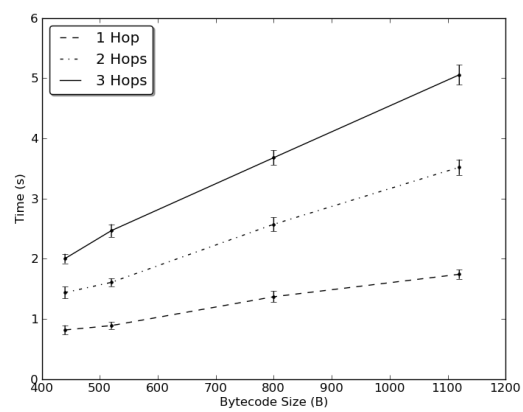

Figure 5. Code update time.

The Use Cases in Sections III-B and III-C presented the requirements of node reprogramming for (i) a rapid set-up procedure and (ii) as an action following an event occurrence.

In a large scale infrastructure, the network access point (where the code is versioned and maintained) can be 1, 2, or 3 hops away from the node which must be reprogrammed. In our experiments we made use of four applications already discussed in two previous papers of ours: two applications of them focusing on video [16] (with and without on-board processing) and two focusing on audio [17] (with and without on-board processing). The audio applications account for 420 (without on-board processing) and 800 Bytes (with on-board processing); the video applications account for 450 and 1,100 Bytes, respectively without and with on-board

processing. Making use of the timestamping service, already discussed in the beginning of this section, we measure the time delay between the start-up of the process and the time when the node application is up and running. In Figure 5 we show that the reprogramming process is always shorter than 5 seconds in all routing conditions and for the full set of implemented codes. For instance switching to the audio application with on-board processing takes a total overhead of about 2 seconds if the code management service is 2 hops away from the reprogrammed node.

The obtained performances should be considered against the specifications of the required service; all in all they demonstrate that an instrumentation grid has a high degree of flexibility and adaptiveness to ease the operations of code versioning and to permit a timely reaction to event occurrence.

VII. Conclusions and outlook

In this paper the architecture and the early implementation of an instrumentation grid suited for large-scale ad-hoc networks is presented. The adopted approach is the one of the Internet of Things aiming at developing virtual services providing connectivity, storage, computing power, and distributed I/O. This work has been inspired by the DORII experience where an instrumentation layer has been imported and operated within a grid infrastructure; our deployment is fully interoperable with state-of-the-art infrastructures based on the grid technology. Moreover it fulfills valuable requirements of ad-hoc networks like reactiveness, flexibility, and adaptiveness; our experiments prove the main functionality required by the use cases.

We are preparing a live, publicly accessible testbed which will be offered to the research community through our web site; through the web interface a user will be able to program a set of device nodes from remote and to be notified upon the occurrence of custom-defined events. Moreover a set of usual grid services like computing power and storage will be offered by the back-end system so that the data will populate a data base and will serve as input to computing demanding applications hosted in PCs.

References

[1] J. W. Hui and D. E. Culler, "Ip is dead, long live ip for wireless sensor networks," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 15–28. [Online]. Available: http://doi.acm.org/10.1145/1460412.1460415

[2] Z. Shelby, "Embedded web services," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 52 –57, december 2010.

[3] D. Fensel, Ed., *Foundations for the Web of Information and Services - A Review of 20 Years of Semantic Web Research*. Springer, 2011.

[4] Commission of the European Communities, "Specific programme implementing horizon 2020 - the framework programme for research and innovation (2014-2020)," *Official Journal of the European Union*, vol. 811, 2011.

[5] S. Hadim and N. Mohamed, "Middleware: Middleware challenges and approaches for wireless sensor networks," *IEEE Distributed Systems Online*, vol. 7, no. 3, 2006.

[6] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ser. SenSys '04. New York, NY, USA: ACM, 2004, pp. 81–94. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031506

[7] A. Dunkels, B. Grnvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proc. of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, Nov. 2004.

[8] P. Levis and D. Culler, "Maté: a tiny virtual machine for sensor networks," in *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*. New York, NY, USA: ACM, 2002, pp. 85–95.

[9] F. Aslam, C. Schindelhauer, G. Ernst, D. Spyra, J. Meyer, and M. Zalloom, "Introducing takatuka: a java virtualmachine for motes," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 399–400. [Online]. Available: http://doi.acm.org/10.1145/1460412.1460472

[10] "Pymite." http://code.google.com/p/python-on-a-chip/.

[11] J. Ko, K. Klues, C. Richter, W. Hofer, B. Kusy, M. Brünig, T. Schmid, Q. Wang, P. Dutta, and A. Terzis, "Low power or high performance? a tradeoff whose time has come (and nearly gone)," in *EWSN*, ser. Lecture Notes in Computer Science, G. P. Picco and W. B. Heinzelman, Eds., vol. 7158. Springer, 2012, pp. 98–114.

[12] P. Vicaire, Z. Xie, E. Hoque, and J. A. Stankovic, "Physicalnet: A generic framework for managing and programming across pervasive computing networks," in *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2010, pp. 269–278.

[13] R. Pugliese, M. Proca, G. Kourousias, A. Del Linz, and A. Curri, "Integrating instruments in the grid for on-line and off-line processing in a synchrotron radiation facility," *Computational Methods in Science and Technology*, vol. 15, pp. 21–30, 2009.

[14] F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, *Remote Instrumentation Services on the E-Infrastructure: Applications and Tools*. Springer, 2010. [Online]. Available: http://books.google.it/books?id=ycT6oZiUa6UC

[15] S. Vinoski, "Advanced message queuing protocol," *Internet Computing, IEEE*, vol. 10, no. 6, pp. 87 –89, nov.-dec. 2006.

[16] D. Alessandrelli, A. Azzarà, M. Petracca, C. Nastasi, and P. Pagano, "Scantraffic: Smart camera network for traffic information collection," in *to appear in the Proceedings of the 9th European conference on Wireless Sensor Networks*, ser. EWSN'12. Berlin, Heidelberg: Springer-Verlag, 2012.

[17] M. Petracca, M. Ghibaudi, R. Pelliccia, and P. Pagano, "Perceptual Voice Communications in IEEE802.15.4 Networks for the Emergency Management Suppor," in *Baltic Congress on Future Internet Communications*, 2012.