A Prototype to Integrate a Wireless Sensor Network with Civil Protection Grid Applications

António Esteves, António Pina,
Vítor Sá, Marco Caldas, Luiz Lopes, Nuno Lebreiro
Department of Informatics, University of Minho
{esteves , pina , vitorsa , marcocaldas , luiz}@di.uminho.pt , nflebreiro@gmail.com

Abstract : The present work was performed in the context of the CYCLOPS project, which aimed to exploit the Grid capabilities for Global Monitoring for Environment and Security (GMES) applications. The scenario exploited in the present work was the existence of remote wireless sensor networks, which could monitor and transmit real-time data from remote places, in order to prevent or react more accurately to situations of natural disasters. Considering a Wireless Sensor Network (WSN) as an instrument, we used the DORII middleware to integrate this instrument with *gLite*-based Grid computing and storage, allowing an effective and user friendly access to the instrument, as it is required by Civil Protection applications. The mentioned goal was achieved by (i) implementing an Instrument Element and several Instrument Managers, which virtualize the WSN; (ii) developing a Custom Java Interface to connect the Instrument Managers with sensors, performing the translation of the commands/data exchanged between them; (iii) implementing additional modules to permit a long duration (or offline) monitoring, saving the observed data in a database; (iv) implementing a Sensor Observation Service, following the OGC standards, providing the users with access to the database.

Keywords : Civil Protection, Grid middleware, wireless sensor network, Web service, OGC standards.

1 Introduction

Civil Protection (CP) applications require the access to acquisition systems (single sensors, sensors networks and airborne and satellite-based remote sensing systems). These applications need to read data from different systems, and also to control the data acquisition to be able to adopt the best strategy for monitoring events. For example a CP application for Flash Flood management could provide access to data from different sensors like weather radars and satellites, and could provide also the possibility of modifying the sampling time of a rain gauges network depending on the characteristics of the observed phenomenon.

A complete suite of services for integrating sensors in an e-infrastructure should include: (i) discovery of sensor systems basing on the user's needs, (ii) description of a sensor's capabilities and quality of measurements, (iii) access to sensor parameters, (iv) access to observations and coverages, (v) planning of sensors observations and (vi) notification of alerts from sensors. One important point is that not all of the sensors are accessible using the same technologies, and this architectural heterogeneity cannot be reduced. In fact, data acquisition systems are managed by different organizations and they are often part of existing data sharing systems based on several technologies (Web, Grid, proprietary or legacy technologies). Even if many initiatives are putting efforts in an architectural harmonization, it should be considered that at least in the next years, many acquisition systems will be accessible only using different technologies and systems. In particular concerning the CYCLOPS project [1], it is possible that some sensors are integrated in the Grid as Instrument Element, while others are not.

The work presented in this paper was performed in the context of the CYCLOPS project, with the goal of joining Grid and **G**lobal **M**onitoring for **E**nvironment and **S**ecurity (**GMES**) communities, in order to fully exploit the Grid capabilities for GMES applications. One of the GMES scenarios is the existence of **R**emote **W**ireless **S**ensor **N**etworks (**RWSN**), which could monitor and transmit real-time data from those remote places, like temperature, humidity or wind speed, in order to prevent or react more accurately to situations of natural disasters like floods, fires or earthquakes.

After evaluating several alternatives, we decided to use the **DORII** middleware, the successor of the GRIDCC project, to control, monitor and collect data from a RWSN. DORII provides a well-proven technology that can be deployed on top of existing Grid middleware, extending the grid e-infrastructure to the control and monitoring of remote instrumentation. EGEE *gLite* is the reference Grid middleware, both for

DORII and the CYCLOPS projects, and the EGEE e-infrastructure is the natural framework in which to deploy and integrate the instrument with Grid technology. The core and novel element of the DORII middleware [2] is the **I**nstrument **E**lement (**IE**), functioning as a Web service interface to integrate instruments and sensors into the Grid. The second key component is the **V**irtual **C**ontrol **R**oom (**VCR**) that has been introduced to provide remote users with a virtual area from where they can control and monitor the instrumentation, and where they can collaborate with each others, even if located in different physical places. The third main component is the **E**xecution **S**ervice, which provides a workflow engine able to handle BPEL workflows interacting both with the new features of DORII and with traditional computational and storage Grid services.

The paper is organized in the following manner. Section 2 presents the related approaches. Section 3 describes the Imote2-based WSN instrument, used as a case study to integrate sensors into the Grid, and the idea of extending this instrument with emulated sensors. Section 4 details a prototype, based on the DORII middleware, developed to integrate a WSN instrument into the Grid. Section 5 introduces a second version of the prototype, developed to overcome some limitations of the DORII version, which is based on the OGC standards. Finally, section 6 presents some conclusions and the future work.

2 Related Work

The **DORII** project [2] aims to deploy an e-Infrastructure for new scientific communities, in one hand ICT technology is still not present at the appropriate level today, in other hand it is required to improve the communities' daily work. The main focus of the DORII project is on groups of scientific users with experimental equipment and instrumentation that are currently not or only partially integrated in the European e-Infrastructures. At present, the following selected scientific areas are represented in the project: earthquake community (with various sensor networks), environmental science community, experimental science community (with synchrotron and free electron lasers).

The **GEO Grid** [3] project integrates virtually a wide variety of data sets, such as satellite imagery, geological data, and ground sensed data. The integration is enabled by Grid technology, and data can be accessed and processed on user demand through standardized web services interfaces on basic OGSA services. GEO Grid is based on four layers: hardware, virtual storage, application and data services, and user interface. The GEO Grid project does not integrate sensors into the Grid, instead they suppose data is archived on a filesystem.

The **Cowbridge** project [4], at Lancaster University, employs the pervasive management support (PMS) approach to predict and manage flooding in a river valley. The system incorporates an adaptive, resilient sensor network that feeds real-time sensor data to a computationally intensive flood-prediction algorithm running on a general-purpose computational cluster in a customized Grid middleware environment. The system enables real-time prediction of flooding including detailed predictions of what areas flooding will most likely affect. It also provides timely alerts to local stakeholders when it perceives flooding to be imminent. An interesting technical aspect of this system is that it supports selectively and dynamically assigning computation tasks to either the sensor network itself or the remote cluster. This approach is not concerned with wireless **s**ensor **n**etworks (**SN**), only wired SNs, which raise new challenges such as power management, sensors discovery, communications safety, routing protocols.

Sensors Anywhere (**SANY**) project [5] deals with sensor networks research for environmental applications, and tries to improve the interoperability of insitu sensors and sensor networks, and allowing quick and cost-efficient reuse of data and services from currently incompatible sources in future environmental risk management applications. To achieve this goal they (i) specified a standard open architecture for fixed and moving sensors and SNs, (ii) developed and validated re-usable data fusion and decision support service building blocks and a reference implementation of the architecture, (iii) tried to contribute to future standards applicable to GMES. All the architecture specifications conform to the OGC and INSPIRE standards. SANY inherits and extends the results of ORCHESTRA and SSE (Services Support Environment) projects. SANY shares the same goal with our project, the use of OGC/SWE standards, but it is not targeted to feed Grid running applications. They also do not consider the use of wireless SNs, but instead they deal with wired ones.

The Open Grid Services Architecture (**OGSA**) [6] describes an architecture for a service-oriented Grid computing environment and is based on several Web-service technologies, notably WSDL and SOAP. In the OGC-based prototype, discussed in section 5, we defined an architecture similar to the OGSA. The **Globus** project [7] offers an open source collection of Grid services that follow OGSA architectural principles. The

Globus Toolkit is an open source software toolkit used for building Grids. It also offers a development environment for producing new Grid services that follow OGSA principles. The Common Instrument Middleware Architecture (**CIMA**) [8], also based on the emerging OGSA, provides a framework that answers the need for re-usable middleware architectures, with the ability to connect instruments with minimal configurations. CIMA concentrates its efforts on the development of a middleware that allows a uniform access to different instruments, but a few words are said about integrating these instruments into the Grid. Another approach is **SensorWeb** [9] that implements standards compliant platform and middleware to integrate sensor networks with emerging distributed computing platforms such as Grids. That approach uses a different GRID middleware, **gridbus**, but follows the same guidelines as CIMA.

In [10] it is proposed a middleware, called **Grid-M**, to integrate WSNs and mobile devices into the Grid. The Grid-M middleware allows management, data communication, services directory, resources discovery and security, providing the developer with a set of reused and homogeneous resources. They provide an API to connect Java applications, running on mobile device, with Grid computing. Instead of trying to integrate the instruments on a *gLite*-based Grid middleware, which is the goal of the CYCLOPS project, they develop their own middleware.

In [11] it is presented an architecture, called Scalable Proxy-based aRchItecture for seNsor Grid (SPRING), to combine WSNs with Grid computing. They also developed a sensor grid testbed to study the design issues of sensor grids and to improve the sensor grid architecture design.

3 Extending an Imote2 Wireless Sensor Network with Emulated Sensors

The instrument used as a case study, to integrate sensors into the Grid, is a **W**ireless **S**ensor **N**etwork (**WSN**) based on **Imote2** modules [12]. A WSN can be seen as a network of small sensor devices, connected without wires, composed by a large number of immovable devices planted in an ad hoc manner, and used to detect and transmit physical characteristics of the environment, known as phenomena. Associating an increasing popularity of the wireless world with the development of cheaper, smaller and embedded processors, WSNs have grown as an important field. The Imote2 module includes an Intel XScale 32-bit processor and a Radio Frequency (**RF**) unit based on the 802.15.4 standard (figure 1). Each Imote2 can be connected to a sensor board that includes a 3D accelerometer, and temperature, humidity, and light sensors. After studying the alternatives for sensor operating system, we selected **TinyOS** 1.x.

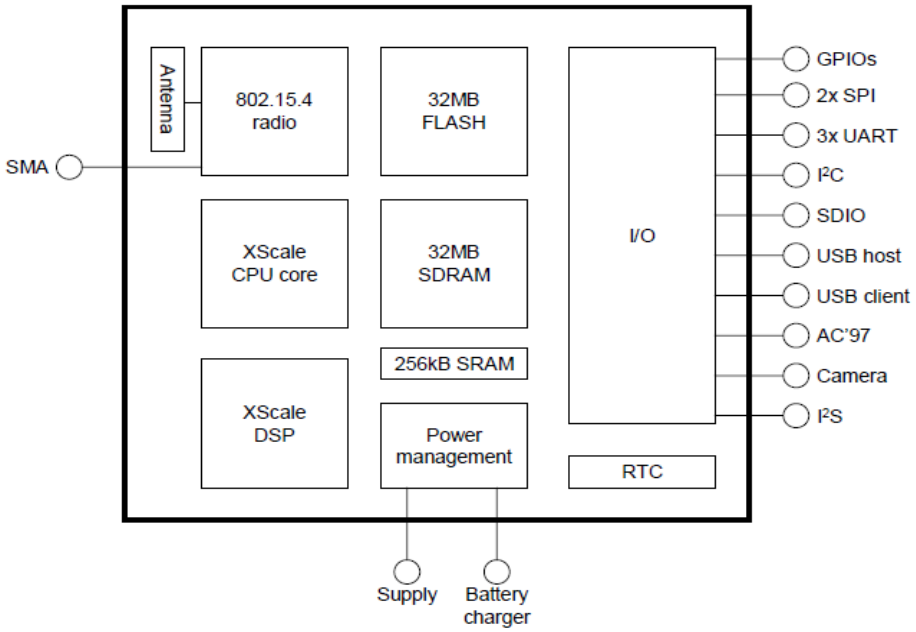

Figure 1. Imote2 radio/processor module.

The present work planned to use a network composed of a small number of real sensor modules extended with a larger number of emulated (simulated) modules. In the context of integrating WSNs into Grid civil protection applications, simulation is relevant due to the fact that (i) it can replace real sensors with a simulated version, which makes it easier, faster and cheaper to test applications without real hardware, (ii) it makes it possible to scale a real network (Imotes2) and test applications with a larger input data set, (iii) it aids on the choice of the best topology and localization of the network, based on type of soil and other geographic criteria. For these reasons, it was evaluated a large set of network simulators, such as GloMoSim, OPNET [13], TOSSIM [14], OMNet [15], EmStar, SensorSim [16], ns-2 [17], SENSE [18], ATEMU [19], SENS [20], DISENS and AVRORA [21]. The comparison of simulators focused on characteristics such as the scalability, the completeness, the fidelity, the GUI, the platforms where it runs and the license of utilization.

The four key elements required for a good TinyOS simulator are: the scalability (ability to handle large networks), the completeness (capability of simulating complete applications), fidelity (capture the network behaviour with fine detail), and bridging (bridge the gap between algorithm and its implementation, allowing developers to test and verify the code that will run on real hardware). The selected simulator is **TOSSIM**, since it is the only one presenting these four key characteristics.

The TOSSIM architecture is composed of five parts: the support for compiling TinyOS component graphs into the simulation infrastructure, a discrete event queue, a small number of re-implemented TinyOS hardware abstraction components, the mechanisms for extensible radio and ADC models, and the communication services for external programs to interact with a simulation.

4 DORII-based Prototype

The architecture of the prototype we propose to connect an Imote2 based WSN instrument into the Grid is synthesized in figure 2. The main components of this architecture are: (i) the WSN itself and its base station; (ii) the instrument access services host with the Custom Java Interface (CJI) and the observations database; (iii) the host of the Instrument Element (IE), which may contain several Instrument Managers (IMs) and associated custom plugins to interface with CJI; (iv) the Virtual Control Room (VCR) host that allows users to access the instrument.

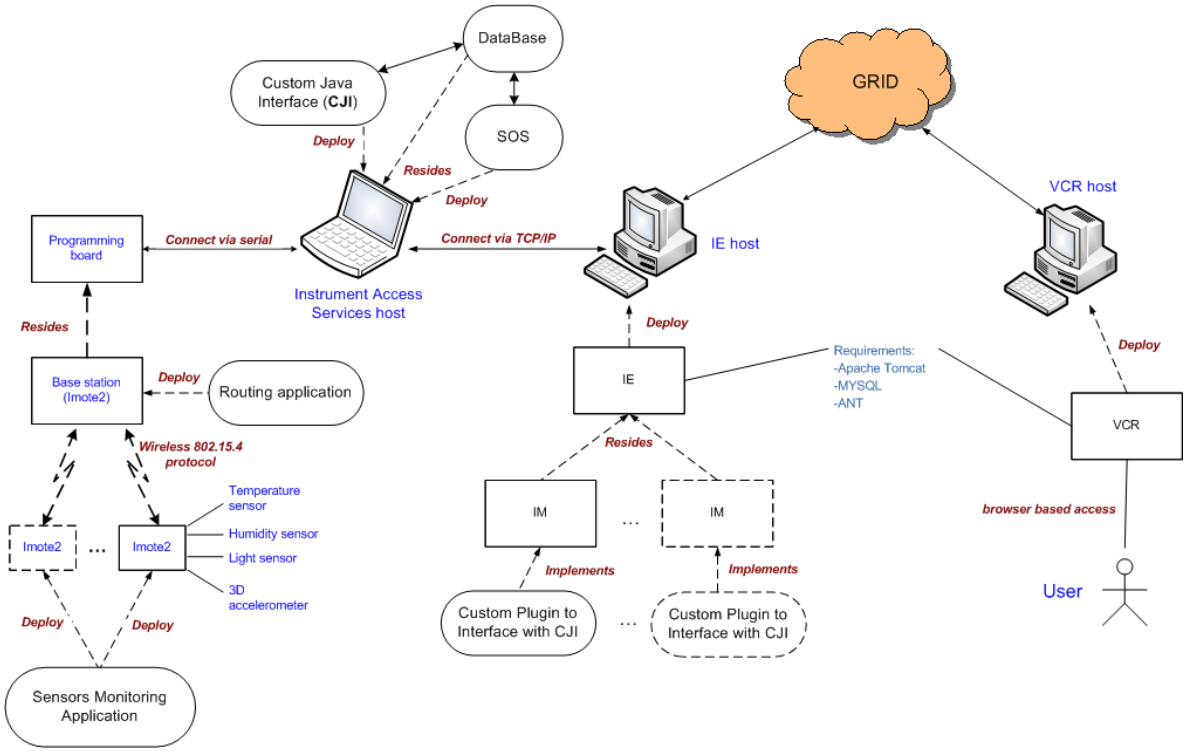

Figure 2. The architecture proposed to integrate a WSN instrument into the Grid.

Each remote Imote2 module, with associated sensors, will run a customized nesC application that monitors temperature, light, humidity and 3D acceleration. All Imote2 nodes of the network are connected to an Imote2 base-station through RF. The Imote2 base-station is connected via USB to a laptop, the main task of the application it runs being to perform packet routing between the remote Imote2 and the laptop. The laptop Instrument Access Services Host (IASH) will have a Custom Java interface for sensor operations and TCP/IP connection to the correspondent IM. The IASH also hosts the Observations Database. The remaining components of the architecture are the VCR and IE hosts, which in our implementation are installed in different desktops, both with Grid connection. There is a single IE for the instrument (WSN), but several VCR users may connect to the IE at the same time. The implementation of the IM, for an Imote2 sensor, includes a custom interface to communicate with the CJI. The main software components of the architecture are presented in figure 3.

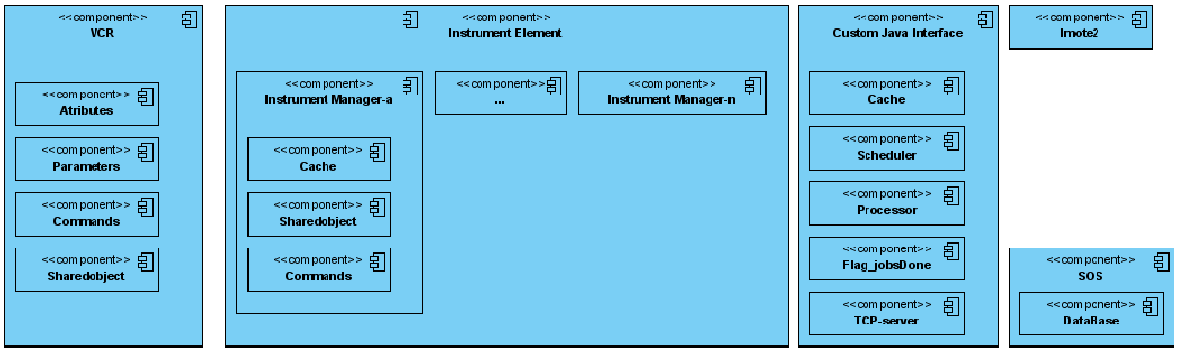

Figure 3. The software components of the architecture used to integrate a WSN into the Grid.

So, there is an Internet portal (VCR) where scientists or civil protection people may log in and see the available WSN, choose the one they are interested in and monitor or interact with it, using defined commands. The VCR allows us to save the data, share it, or send it to grid-running simulations. The WSN must be virtualized in order to be visible on the VCR. This virtualization is done by an Instrument Element that will integrate the specifications and capabilities of the WSN using one or more Instrument Managers. The IE is basically a container for Instrument Managers. The developed prototype uses one IM for each sensor, due to DORII constraints, namely regarding time management and duration of the jobs. Another particularity that brings complexity to this case study is that the real instrument is the single sensor, not the network. As consequence of the constraints imposed by the DORII middleware, our instrument must operate in one of two modes: real time mode or user interaction mode. When there are no users sending commands to the instrument, the instrument performs stand alone monitoring and the users automatically watch the real time observations. In this mode, the users can only handle real time observations: save, disseminate to other users or send to Grid application (figure 4). When the user wants to submit a command, to request data or schedule observations, the instrument operates in user interaction mode (figure 5).

The last block of the system includes the real WSN and its connection to the IE. A Custom Java Interface (CJI) has been developed to perform the translation of the commands and data between the IM and the sensors and to manage the connection itself using TCP/IP. The CJI gains some major features in the prototype, because it is necessary to guarantee long-time jobs (sensor monitoring), including when users are offline, save the results in a file, and signal the jobs that are complete. It was decided to adopt a database format in agreement with the OGC Sensor Observation Service (SOS) standard. The CJI also receives the user requests and processes them into multiple single sensor readings, marking those readings with a tag that will identify the user and job to which they belong.

The Virtual Control Room (VCR) provides the user with several commands to interact with the sensors, and show the state of some defined attributes and parameters. This state is updated every 5 seconds by the VCR from the Instrument Manager. The available attributes are temperature, humidity, light, accelerometer, battery level, flag *JobsDone*, File URL. The parameters are the ID, latitude, and longitude. Finally, the available commands are *Get(<temperature|humidity|light|accelerometer>,each_time,during_time,job_name)*, and *GetJob(job_name)*. When a job is complete, the flag *JobsDone* is updated with its identification, so the user who ordered the job knows that the associated file is ready.

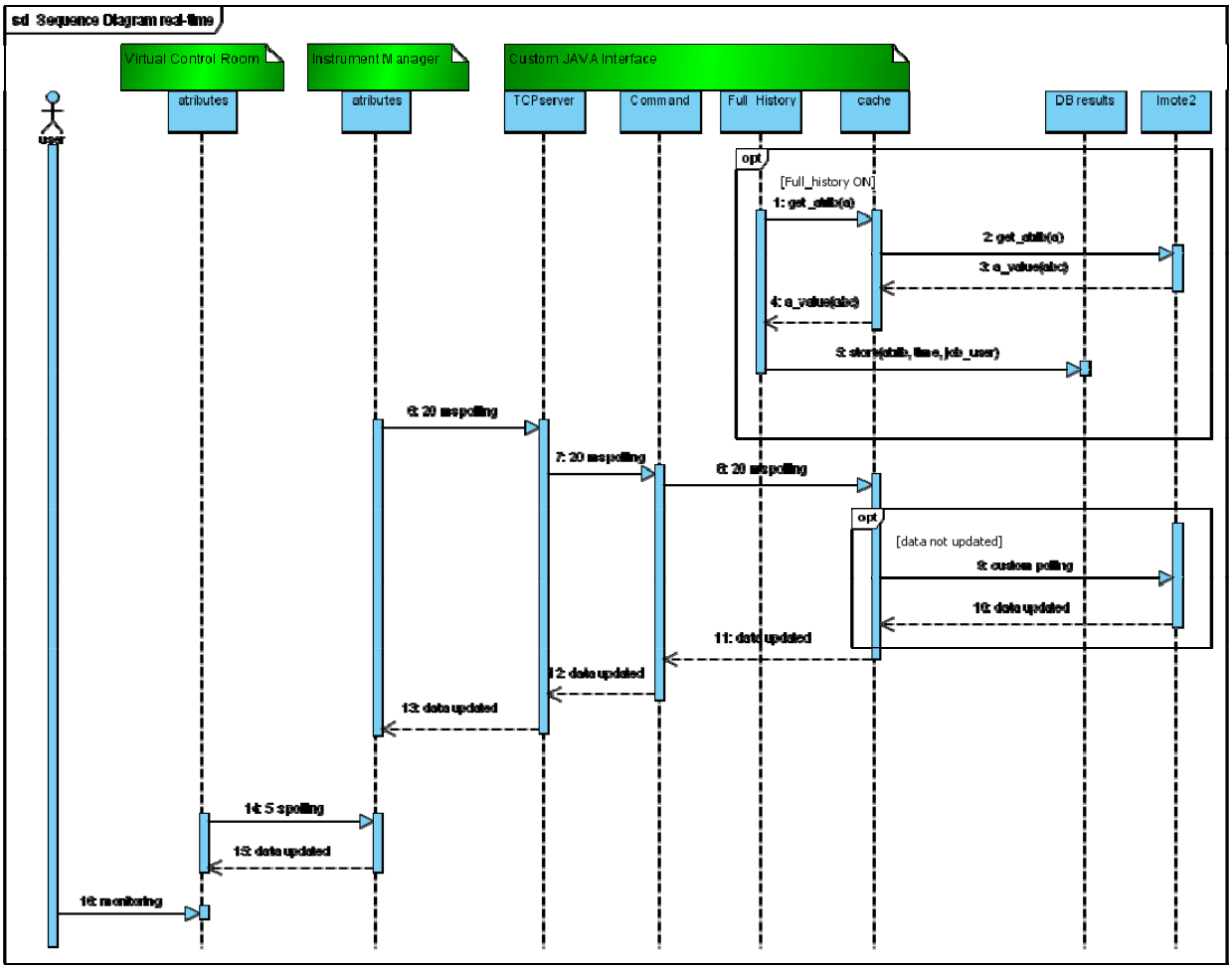

Figure 4. Real time mode timing.

The IM is an object that virtualizes an instrument (an Imote2 node), and performs the required translation between the instrument and the user-VCR, in order to operate and monitor the sensors. The IM has a cache, where the values for the VCR attributes and parameters are kept, which is automatically updated every 20 milliseconds with values from the instrument. The CJI implements another cache that it used to answer the IM, therefore avoiding unnecessary and possibly problematic interactions with the sensors, due to energy and bandwidth management issues. The CJI has several components in order to manage the system needs:

- A TCP server to manage the connection to the IM;
- A command block to parse the commands and redirect them to the adequate recipient;
- A scheduler to split a periodic job into several unique requests and insert them in a job schedule table;
- A processor of requests coming from the table and save the results in a database;
- A cache to keep the sensor values and update them when their lifetime ends;
- A historical module to keep a complete log of all the attributes in the database.

The CJI has two logging modes: “history on” and “history off”. The “history off” mode only saves the results from the user requests on the database, while the “history on” mode saves all the readings made by the sensors, according to the lifetime of each attribute. The attribute lifetime, the logging mode and the battery management are configurable and controlled by the IASH administrator using a XML configuration file.

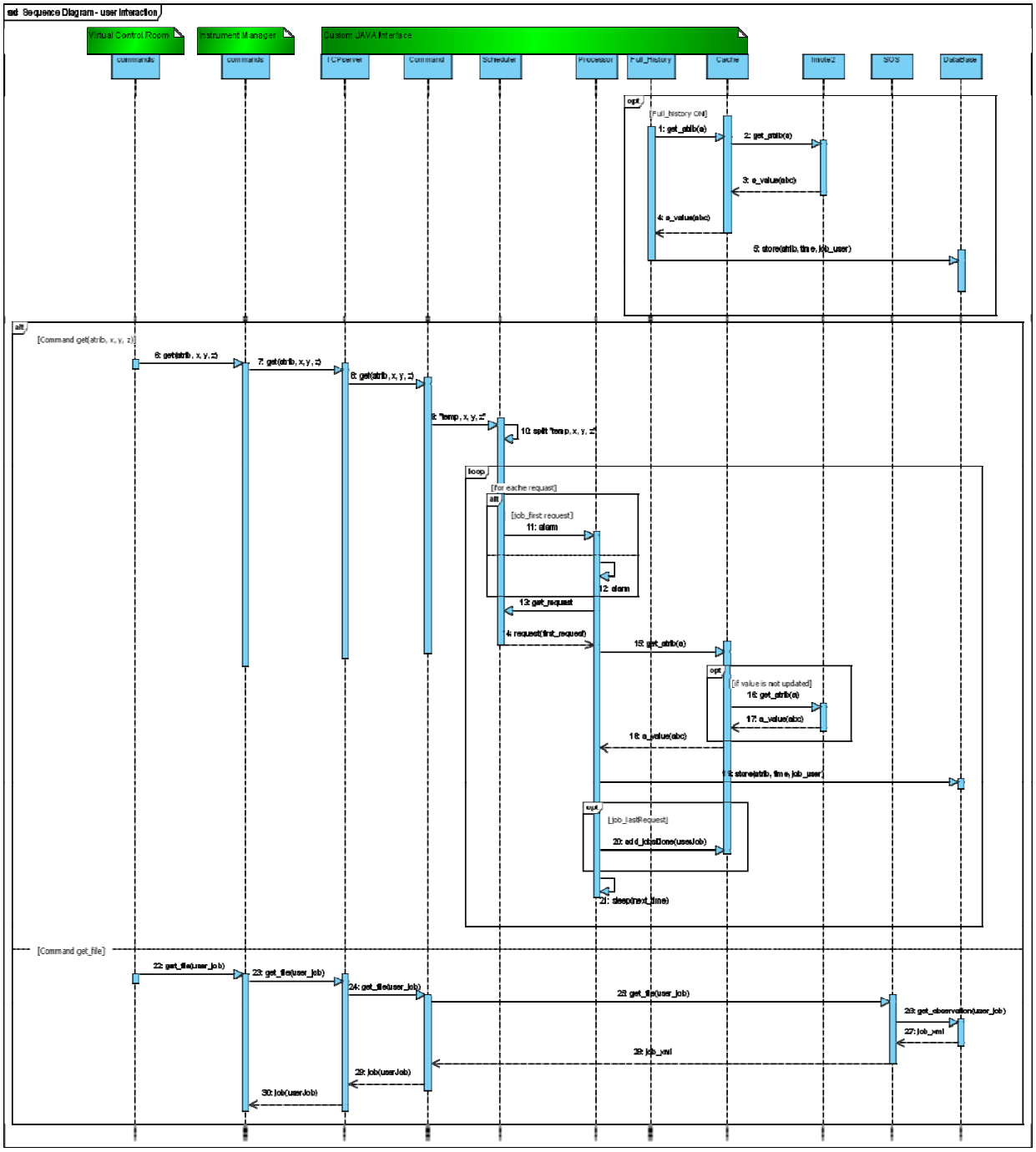

Figure 5. User interaction mode timing (*get* commands).

5 OGC-based Prototype

To overcome some limitations of DORII instrument access, it was carried out a second version of the prototype: the implementation of a Web service, to access Imote2 sensors, based on the **OGC** standards [22]. In the OGC **S**ensor **W**eb **E**nablement (**SWE**) activity, members of the OGC are defining, testing, and documenting a consistent framework of open standards for exploiting Web-connected sensors and sensor

systems of any type. To develop this prototype, we analyzed two relevant implementations of OGC standards: one from NICTA Melbourne University [23], and the other from 52º North [24].

The basic standards for modeling and codification of the information involved in the access to sensors are **O**bservation & **M**easurements (**O&M**) and **SensorML**. SensorML [25] provides a functional model of the sensor system, rather than a detailed description of its hardware. O&M [26] defines a framework and a conceptual model, where the encoding is formalized as an application schema. O&M binds a result to a feature of interest, upon which the observation was made using a procedure to determine the value of the result.

The standards that define the essential services to access the sensors data are **S**ensor **O**bservation **S**ervice (**SOS**) and **S**ensor **P**lanning **S**ervice (**SPS**). The SOS [27] provides an interface to make sensors and sensor data archives accessible via an interoperable Web based interface. The basic SOS allows the operations *GetCapabilities* (request a self-description of the service), *DescribeSensor* (request information about the sensor itself in a SensorML document) and *GetObservation* (request the sensor data, encoded in an O&M document). The SPS [28] is an interface to task an asset, or an asset system. The SPS allows the operations *GetCapabilities* (retrieves the metadata about the capabilities of an SPS), *DescribeTasking* (gets the information that is required to prepare a tasking request), *GetFeasibility* (obtains the feasibility details about a tasking request), *Submit* (submits a tasking request) and *DescribeResultAccess* (retrieves the information about where observed data can be access from).

The proposed architecture is based on reading, controlling and analyzing data from a SN through an environment Grid, to support CP applications. As in the previous version of the prototype, the DORII middleware was also used for accessing the Grid in conjunction with the OGC standard sensor observation service. In this architecture, the access to a controllable database is carried out by the previously presented CJI. All the services of the OGC, as well as the control service CJI and the database, are available through the IM of the DORII whose access is done by means of the VCR. An OGC-based prototype, not fully implemented, is illustrated by the layered architecture included in figure 6. Service and sensor access layers can access the database. In this case, the CJI functionalities would be spread between the service layer (observations scheduling), the sensor access layer (instrument control, access and management) and the storage access layer (observations saving).

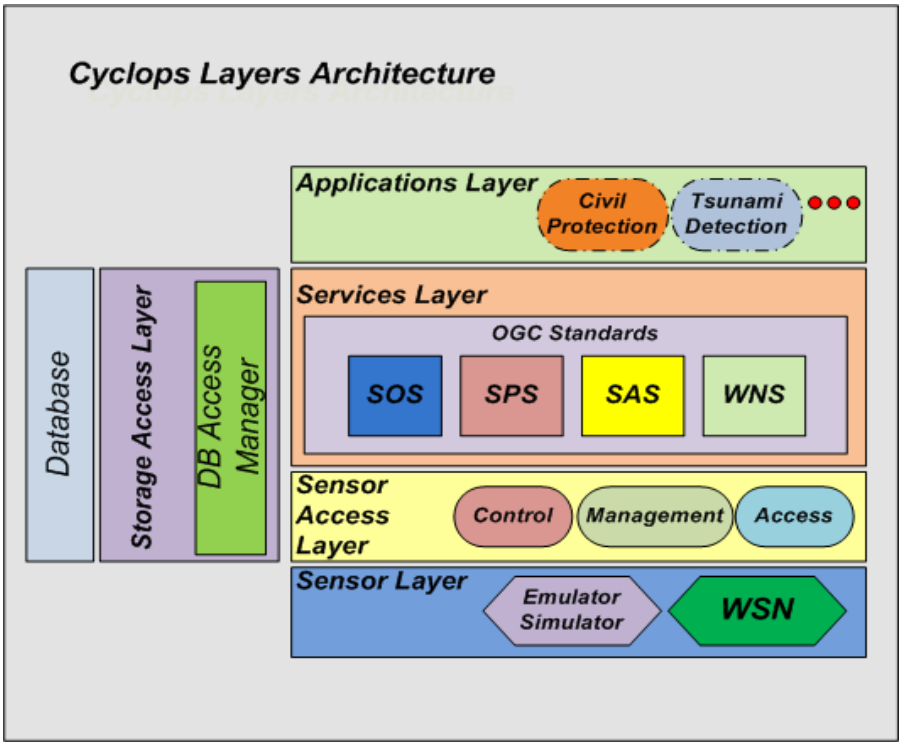

Figure 6. The layered architecture of 2nd version of the prototype, following the OGC approach.

The used **database schema** is based on the 52º North SOS implementation. The core tables and definitions are:

- ***feature_of_interest***, the table that stores data about the features or feature of interest. It represents the identifiable object(s) on which the sensor systems are making observations.
- ***observation,*** the table that aggregates the data of an observation event like air temperature, procedure (sensor or group of sensors), the feature of interest and the observation value, which is stored in a separate table. An observation is an action with a result which has a value describing some phenomenon. An observation uses a procedure to determine the value of the result, which may involve a sensor or observer, analytical procedure, simulation or other numerical process.
- ***procedure***, it is the reference to one or more procedures, including sensor systems, instruments, simulators, etc, that supply observations in this offering.
- ***offering,*** the table that stores each offering of the SOS. It is only used when the SOS is initialized to read in the offerings of the SOS and the phenomena which are related to each offering. An SOS organizes collections of related sensor system observations into Observation Offerings. For example, temperature in a station weather placed in a city.
- ***phenomenon***, it represents the phenomena (temperature, humidity, etc).

The SOS will use these tables to answer incoming requests, but it is not SOS responsibility to update the values in the tables. This task has to be fulfilled externally or using the CJI.

6 Conclusions and Future Work

A functional prototype to provide Grid civil protection applications with data from a remote wireless sensor network was developed. A first version of the prototype is based on the utilization of the DORII middleware. This prototype was later improved with observations handling and services which are in conformity with the OGC SWE standards, O&M/SensorML and SOS respectively. SOS was implemented successfully, working in Standalone mode, receiving XML requests and returning XML documents with the response.

From our point of view, the main weaknesses of the present DORII middleware evolution are:

- The necessity of having the VCR online in order to receive alarms or to carry out planned observations;
- Limited support for the virtualization of an instrument that is a network of devices;
- The control needs to be shared simultaneously by several different users while maintaining the integrity of their individual jobs.

These problems will be overcome if an improved interface is used in the instrument side, but this will result in custom and non-normalized libraries, and maybe some problems regarding Grid protocols. In order to provide a feasible tool, these points should be corrected in DORII, as well as a few others of minor functionality already discussed with the DORII project members. In the present stage of development, the IE concept has shown itself to be a little too restrictive to be used by CP people; however, it seems that there are good chances of it evolving in the right direction in the near future.

The Web-based access to sensors seems to be an obvious valuable improvement to the local and traditional access to heterogeneous sensors. Further studies are needed to evaluate if Grid enabled access to sensors is adequate for Civil Protection purposes. It is necessary to identify and balance the real necessities of CP people, the benefits brought by this technology, the training overhead needed to use it, and its fidelity in critical situations usage.

In the future, a QoS could be defined to guarantee that Grid-enabled applications and sensor accessing fulfill Civil Protection requirements. QoS can be used to guarantee that applications perform in real-time, or near real-time. Another issue to address in the future is power consumption control in sensors. In WSNs, like those based in Imote2s, power is a critical resource that must be saved as much as possible. This can be crucial to keep sensors running when an intensive access, such as a full time access, is being performed by CP applications in an emergency situation.

References

[1] The Cyclops Project web site. http://www.cyclops-project.eu

[2] The DORII project: Deployment of Remote Instrumentation Infrastructure. http://www.dorii.eu

[3] Naotaka Yamamoto, Ryosuke Nakamura, Hirokazu Yamamoto, Satoshi Tsuchida, Isao Kojima, Yoshio Tanaka, Satoshi Sekiguchi. "GEO Grid: Grid infrastructure for Integration of Huge Satellite Imagery and Geoscience data sets". *Proceedings of the 6th IEEE International Conference on Computer and Information Technology (CIT'06), 2006*. www.geogrid.org

[4] Geoff Coulson, Dean Kuo, and John Brooke, "Sensor Networks + Grid Computing = A New Challenge for the Grid?". *IEEE Distributed Systems Online, vol. 7, no. 12, 2006, art. no. 0612-oz002*. www.comp.lancs.ac.uk/~geoff/Publications/DSO06a.pdf

[5] D. Havlik, G. Schimak, R. Denzer, B. Stevenot. "Introduction to SANY (Sensors Anywhere) Integrated Project", Shaker, Graz, Austria, pp.541-546, 2006. sany-ip.eu/filemanager/active?fid=19

[6] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration". www.globus.org/research/papers/ogsa.pdf

[7] The Globus Alliance project. www.globus.org

[8] Tharaka Devadithya, Kenneth Chiu, Kianosh Huffman, Donald F. McMullen. "The Common Instrument Middleware Architecture: Overview of Goals and Implementation". *Proceedings of the First International Conference on e-Science and Grid Computing (e-Science'05)*, 2005.

[9] Xingchen Chu and Rajkumar Buyya, "Service Oriented Sensor Web, Sensor Network and Configuration: Fundamentals, Standards, Platforms, and Applications", N. P. Mahalik (ed), pp.51-74, *Springer-Verlag*, ISBN: 978-3-540-37364-3, January 2007.

[10] Hans A. Franke, Douglas O. Balen, Carlos O. Rolim, Fernando Koch, Kleber Vieira, and Carlos Westphall. "Grid-M: Middleware para Integrar Dispositivos Móveis, Sensores e Grids". *XII Workshop de Gerência e Operação de Redes e Serviços (WGRS)*, 2007.

[11] Hock Beng Lim, Yong Meng Teo, Protik Mukherjee, Vinh The Lam, Weng Fai Wong, and Simon See. "Sensor Grid: Integration of Wireless Sensor Networks and the Grid". *Proceedings of the IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)*, 2005. ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01550845

[12] Crossbow. "Imote2 hardware reference manual". September, 2007.

[13] Xinjie Chang. "Network Simulations with OPNET". *Proceedings of the 31st Conference on Winter Simulation: Simulation - a Bridge to the Future*, Volume 1, pp. 307-314,1999.

[14] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications". *Proceedings of SenSys'03, First ACM Conference on Embedded Networked Sensor Systems, New York*, ACM Press, 2003.

[15] C. Mallanda, A. Suri, V. Kunchakarra, S.S. Iyengar, R. Kannan, and A. Durresi. "Simulating Wireless Sensor Networks with OMNeT++", *Sensor Network Research Group, Department of Computer Science, Louisiana State University*, 2005.

[16] Sung Park, Andreas Savvides, and Mani B. Srivastava. "Sensorsim: a simulation framework for sensor networks", in *MSWIM'00: Proceedings of the 3rd ACM International Workshop on Modelling, Analysis and Simulation of Wireless and Mobile Systems*, (New York, NY, USA), pp. 104–111, ACM Press, 2000.

[17] "The Network Simulator ns-2". http://www.isi.edu/nsnam/ns.

[18] Gilbert Chen, Joel Branch, Michael Pflug, Lijuan Zhu, and Boleslaw Szymanski. "SENSE: A Sensor Network Simulator. *Advances in Pervasive Computing and Networking"*, 2004.

[19] Jonathan Pollet, et al. “ATEMU: A Fine-Grained Sensor Network Simulator”. *Proceedings of SECON’04, First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[20] Sameer Sundresh, Wooyoung Kim, and Gul Agha. “SENS: A Sensor, Environment, and Network Simulator”. *Proceedings of 37th Annual Simulation Symposium*, 2004.

[21] Ben L. Titzer, Daniel K. Lee, Jens Palsberg. “Avrora: Scalable Sensor Network Simulation With Precise Timing”. *Proceedings of IPSN'05,Fourth International Conference on Information Processing in Sensor Networks*, 2005.

[22] Open Geospatial Consortium (OGC). www.opengeospatial.org/ogc

[23] Xingchen Chu. “Open Sensor Web Architecture**:** Core Services”. Thesis of Master of Information Technology. *Grid Computing and Distributed Systems Laboratory. Department of Computer Science and Software Engineering. University of Melbourne, Australia*, 2005.

[24] 52North Website: http://52north.org

[25] Mike Botts. “OpenGIS Sensor Model Language (SensorML) Implementation Specification”. OGC Draft 05-086, Open Geospatial Consortium, 2005.

[26] Simon Cox (Ed). “Observations and Measurements”. OGC Draft 05-087r3. Open Geospatial Consortium. 24 February, 2006.

[27] Arthur Na, and Mark Priest, “Sensor Observation Service Implementation Specification”. OGC 06-009r1, 2006.

[28] Ingo Simonis. “OpenGIS Sensor Planning Service Implementation Specification”. OGC 07-014r3, 2007.