Mobile Information Systems 10 (2014) 19–35
DOI 10.3233/MIS-130170
IOS Press

Ubiquitous monitoring solution for Wireless Sensor Networks with push notifications and end-to-end connectivity

Luis M.L. Oliveira[a], Joel J.P.C. Rodrigues[a,*], André G.F. Elias[a] and Bruno B. Zarpelão[b]
[a]*Instituto de Telecomunicações, University of Beira Interior, Covilhã, Portugal*
[b]*Pontifícia Universidade Católica do Paraná (PUC-PR), Londrina, Brazil*

Abstract. Wireless Sensor Networks (WSNs) belongs to a new trend in technology in which tiny and resource constrained devices are wirelessly interconnected and are able to interact with the surrounding environment by collecting data such as temperature and humidity. Recently, due to the huge growth in the use of mobile devices with Internet connection, smartphones are becoming the center of future ubiquitous wireless networks. Interconnecting WSNs with smartphones and the Internet is a big challenge and new architectures are required due to the heterogeneity of these devices. Taking into account that people are using smartphones with Internet connection, there is a good opportunity to propose a new architecture for wireless sensors monitoring using push notifications and smartphones. Then, this paper proposes a ubiquitous approach for WSN monitoring based on a REST Web Service, a relational database, and an Android mobile application. Real-time data sensed by WSNs are sent directly to a smartphone or stored in a database and requested by the mobile application using a well-defined RESTful interface. A push notification system was created in order to alert mobile users when a sensor parameter overcomes a given threshold. The proposed architecture and mobile application were evaluated and validated using a laboratory WSN testbed and are ready for use.

Keywords: Wireless Sensor Networks, Internet of Things, ubiquitous computing, network monitoring, mobile computing, RESTful Web Services, push notifications, Android applications

1. Introduction

Wireless sensors are tiny devices that are able to measure several environmental and crucial data. Recently, these devices have been used in areas such as environmental monitoring, home automation, and war scenarios. In this context, a new emerging technology called wireless sensor networks (WSNs) has become a trend in technological research [12,20]. This technology combines hundreds or even thousands of tiny and resource constrained sensor devices that communicate wirelessly in order to accomplish a common task. These devices are spatially distributed in the environment in order to collect data about surrounding environmental variables [18,28]. Each device has several sensor modules capable of measuring parameters such as temperature, humidity, and luminosity.

The main challenges regarding wireless sensor networks are power consumption of sensor devices and their connection to the Internet [8]. Power consumption is highly affected by the communication

*Corresponding author: Joel J.P.C. Rodrigues, Instituto de Telecomunicações, University of Beira Interior, Covilhã, Portugal. E-mail: joeljr@ieee.org.

with nodes. A solution to this problem is reducing the communication among nodes using better routing algorithms and shutting down (hibernate state) the nodes when they are not required [19,24].

Connecting these limited devices to the Internet is a big challenge because the use of the TCP/IP stack, as initially conceived, is too heavy in the context of WSNs. However, the use of IPv6 over Low-power Personal Area Networks (6LoWPANs) allows the transmission of IPv6 packets over IEEE 802.15.4 wireless links and enables WSNs to communicate with the Internet more efficiently [32]. Basically, 6LoWPAN adds an adaptation layer below network layer that fragments the packets and compresses the IPv6 transport layer headers. These adjustments allow the IPv6 to be used in low-power networks such as WSNs. The connectivity between 6LoWPAN-enabled WSNs and IPv6 networks is not straightforward because devices with IPv6 support are not able to handle the 6LoWPAN compression and fragmentation [2]. One solution to this issue is using an intermediary element such as a gateway that allows data exchange between WSNs and IPv6 hosts through 6LoWPAN [7,26].

The introduction of 6LoWPAN turn IPv6 suitable for resource-constrained devices enabling their connectivity to the Internet. Smartphones are becoming current personal computers, but it is a big challenge to establish end-to-end connectivity (between an end-user device and a sensor node) with all sorts of smart objects as proposed by the Internet of Things vision [16]. To solve this issue, an end-to-end connectivity solution is proposed in order to interconnect smartphones and sensor devices allowing real-time mobile monitoring.

The integration between mobile devices, specifically smartphones, and future wireless networks is crucial for the Internet growth. In the context of WSNs, the use of smartphones to control and monitor these kinds of networks represents a new trend in ubiquitous computing research [33]. The growing diversity of mobile operating systems and hardware platforms is developing new and heterogeneous network scenarios. Therefore, the construction of new models and architectures to enable the interaction between these devices in a platform independent way is essential [13].

Due to the heterogeneity of recent mobile devices and platforms, the construction of a Web service to interconnect these devices in a platform independent way require open technologies and protocols. The interaction between smart phones and WSN devices is possible by using Web technologies such as Web services and IP-enabled devices. The Representational State Transfer (REST) architecture is commonly used in the construction of Web services since it is based on HTTP that is supported by all smartphones [30]. The REST architecture is based on client-server communication, where clients request available resources from the Web service. A REST resource is defined as a set of data that is available through a well-defined RESTful interface [11]. Mobile clients request these resources using well-known HTTP methods such as GET and POST. To exchange information between the REST Web service and the mobile device, XML, and JSON media types are commonly used. Furthermore, since a WSN involves large amounts of data, the exchange of information needs to be optimized.

Wireless sensor networks are strictly related with monitoring solutions and the information collected by the sensors is more important than the sensor itself. In order to increase the efficiency of wireless sensor network monitoring, the user should be alerted when there are significant changes in sensed data. In a mobile environment, it is becoming natural for us to receive alerts or notifications from several services such as eMail and newsletters. The integration of push-notifications in ubiquitous wireless sensor networks makes sense since it is crucial for one to be alerted when a value collected by a sensor exceeds a threshold. Recent mobile operating systems are also capable to receive and presente this type of notifications seamlessly. Pushing notifications to the mobile device represents significant energy saving when compared with always-on solutions based on polling requests [29].

The main contributions of this paper are the following: (1) the introduction of a system architecture specifically designed to collect, store, and present real-time wireless sensor networks data and historical

measures in mobile environments; (2) a push notification system that allows mobile users to receive an alert over the Internet when a sensor value overcomes a given threshold; (3) an Android mobile application that presents the latest data collected by the WSN and historical measures in scalar and graphical ways; (4) and the testbed architecture definition to evaluate, demonstrate, and validate the proposed approach.

In the proposed architecture, users have three different ways of accessing sensors data. In the first one, users request up-to-date and historical data to a RESTful service that retrieves sensor data stored in a relational database. A format is also defined for the messages exchanged between the mobile devices and the RESTful service. Adoption of REST, XML, and JSON allow heterogeneous mobile clients to exchange information with the server independently of their platforms. In the second way of accessing sensors data, users request real-time data directly to the WSN gateway. Finally, in the third way, a push service notifies users when sensor data overcomes a given threshold. It is another possibility to users that do not want to keep requesting data to the RESTful service or WSN gateway, allowing client devices to optimize power consumption.

This paper is organized as follows. Section 2 reviews the available related literature about the topic. Section 3 introduces and describes the overall system architecture and the interaction between its modules. Section 4 addresses the implementation of the proposed architecture while Section 5 presents the design and construction of the mobile application. Section 6 evaluates and demonstrates the proposed solution considering its architecture and mobile application with performance measurements considering different scenarios. Finally, conclusion and further works are addressed in Section 7.

2. Related work

Wireless sensor networks (WSNs) monitoring applications demand new functionalities and design patterns to address recent challenges in efficiency, interoperability and user interaction. The increasing heterogeneity of mobile devices, operating systems, and communication interfaces requires modern architectures and mobile applications that access the information in a platform independent way. This section presents some available solutions regarding WSN monitoring. Some projects address the used communication protocols and their architectures while others focus on the application layer.

Munawar et al. [3] proposed an Open Sensor Platform to interconnect mobile devices and sensors. The solution uses available commercial hardware and software tools such as a proprietary data acquisition device. This device receives sensed data and sends the information to a host PC using a proprietary PC application. The information is then requested by the mobile device through a Symbian OS application. Since the presented solution does not use the Internet to retrieve data from the sensors, the mobility of the user is highly reduced. Also, using proprietary hardware and software is a limitation due to interoperability and costs. The use of Symbian OS is also a limitation, since it is becoming obsolete when comparing to modern mobile operating systems such as Android and iOS from Apple.

Herrera et al. [17] present an approach to wireless sensor networks monitoring using Zigbee and the iPhone platform. The focus of this solution is to collect environmental and weather data from remote stations and present the information on the mobile device. The remote stations are equipped with a microcontroller that periodically gathers data from the sensors such as wind speed and rainfall. The sensed data is then parsed into a readable format and sent to a proprietary gateway coordinator. This coordinator interconnects the Zigbee personal area network and other network with access to database servers. The data is stored in a MySQL database through PHP scripting and a web page. The mobile application sends requests to the web page in order to retrieve last sensed data. The use of PHP and

HTML instead of a structured Web Service is a limitation of this approach in terms of performance and scalability.

Hornsby et al. [1] proposed an architecture that is based on the XMPP-protocol and wireless sensor networks that support a push based notification functionality. The proposed architecture uses Atom feeds [22] to communicate with the wireless sensor network in a Web Service-like way and an UPnP gateway was used to interconnect the XMPP-based WSN and UPnP media-oriented networks. Therefore, messages are exchanged over IP protocol using the XML media type. In order to interact with the WSN, a solution was implemented in Internet tablet devices. This architecture has some drawbacks related to the use of XMPP-protocol that is not yet supported by many mobile operating systems. This protocol is also restricted to XML media type that is a limitation in modern mobile environments where the use of JSON is becoming a standard.

Cardei et al. [21] presented a RESTful architecture for healthcare patient monitoring using heterogeneous wireless sensor networks. This innovative approach uses an Android smartphone as a gateway in order to interconnect the WSN and the Internet. The heterogeneous WSN used in this system interconnects environmental, medical, and smartphone internal sensors. The sensed data is processed on the smartphone and sent to the Internet Web Service using the cellular network. This approach represents a limitation when using the mobile device as a gateway due to the limited resources of these devices, such as processing power and battery consumption.

A monitoring platform, called WSN Monitor, was proposed by Vajsar and Rucka [27]. The platform is based on client-server architecture and is focused on monitoring and managing wireless sensor networks. The collected data is stored in a MySQL database and the server processes requests from the client application using the data available in the database. The client application was developed using the proprietary Adobe Flex framework [4] that has some limitations in terms of performance and support when compared with native mobile applications.

Moreira et al. work [25] focuses on the design and construction of a mobile monitoring application for wireless sensor networks. The proposed architecture is based in REST interfaces and XML messaging. The mobile application was built on the top of the Android platform but no performance tests were conducted to validate the architecture and the mobile application. The proposed approach also features an alert functionality that alerts the user when a sensor threshold value exceeds a defined limit. The alerts are sent via SMS or e-mail and not as native push-notifications that are standard across the mobile operating system.

Tudose et al. [10] proposed a solution for home automation using a 6LoWPAN wireless sensor network and a mobile monitoring application. The architecture is based on a wireless sensor and actuator network with energy harvesting capabilities that minimize node power consumption. In order to interconnect the WSAN and the Internet, a gateway was developed. Periodically, sensed values are transmitted wirelessly to the gateway over a UDP over IPv6 connection. The gateway receives and stores the collected data sensed by WSN devices. The Android mobile application sends REST-like requests to the gateway that responds with data in JSON format. The inexistence of a structured Web Service on the proposed architecture has limitations in terms of scalability and performance. Moreover, since no database was constructed in order to store collected data, the access to historical measures is limited.

There are also commercial solutions that aim to gather, store, and display sensor data. Cosm [9], formerly named Pachube, is a solution based on a Web portal that allows users to visualize and manipulate data collected by sensors. Moreover, Cosm offers a notification system that sends HTTP POST requests to a URL selected by the user.

In the past few years, several solutions were proposed to control and monitor wireless sensor networks. This paper proposes a reliable architecture to collect, store, and present data gathered by wireless sensors

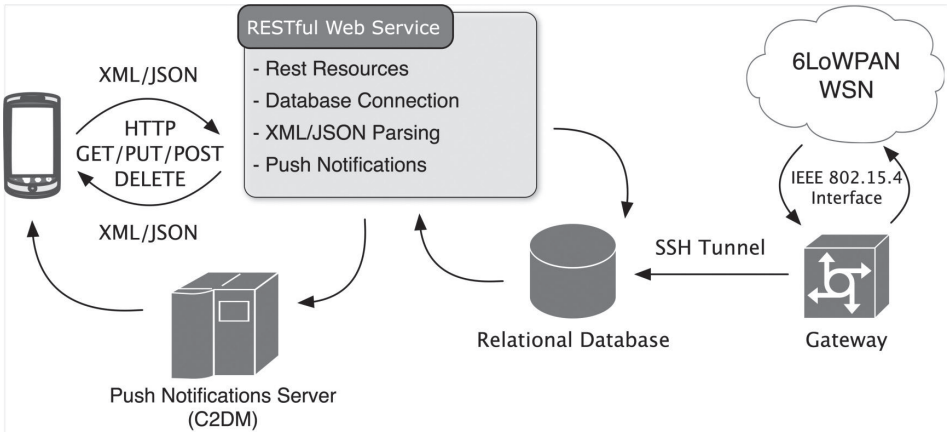

Fig. 1. Illustration of the system architecture diagram.

in a ubiquitous environment. The proposed architecture is designed for being a platform independent on a client and server sides since it is based on REST interfaces and XML/JSON messages. Besides, the native push-notification functionality presented sends a notification to the mobile user when a sensor reading overcomes a given threshold.

3. System architecture

The proposed four-tier architecture is based on the following main components: a Web service, a gateway, a relational database, and a mobile client application. The entire system architecture is illustrated in Fig. 1. The communication between these four components allows mobile client devices to present data collected by the sensors in a ubiquitous environment. The architecture was designed to enable the mobile application to present real-time data as well as stored historical measures. A push notification system integrated into the Web service and the mobile application is used to alert users if a sensor reading overcomes a given threshold.

Sensor devices send collected data to the gateway in pre-defined (and customizable) time intervals according to the variation of the physical sensed phenomenon. The software in the gateway parses the raw data and stores it in the relational database. Then, the mobile client application sends requests to the Web service that runs a query in the database and returns the result to the mobile application through a specified file format. Besides that, the mobile application can also request real-time data from the gateway. When a sensor reading overcomes a given threshold, the Web Service sends a push notification to the mobile application in order to alert the user that an event occurred. The push notification system is built for the Android operating system and is integrated in Google Cloud to Device Message servers that enable the mobile device to receive push notifications in a mobile environment.

The Web service ensures the communication between the relational database and the mobile application, using HTTP over an IPv4-based network. In order to communicate with the server, an Internet connection is required at the mobile device. This interconnection between Internet-based devices is crucial in the near future and is one of the biggest challenges of the Internet as defended by the Internet of Things vision. Designing and building the Web Service on top of industry open standards was a main requirement. Simplicity, scalability, and interoperability were also key requirements in the Web

service development process. The interoperability ensures the ubiquitous access to the database from multi-platform client applications and also the use of different file formats in the information exchange.

Nowadays, two main approaches are used in the construction of Web Services, namely Simple Object Access Protocol (SOAP) and REST. Both approaches have advantages and disadvantages depending on the deployment environment.

The REST approach uses a standard Uniform Resource Identifier (URI) that requests a unique resource in the Web service interface. The approach is simple and can be executed on any client or server that provides HTTP/HTTPS support. REST allows many different data formats while SOAP only supports XML.

SOAP uses XML in the definition of the envelope, that defines what is in the message and how to process it, in the encoding rules for data types, and finally, in the definition of the procedure calls and responses. The envelope is sent via HTTP and a Remote Procedure Call (RPC) is executed and the envelope is returned with the corresponding response in a XML formatted document. In comparison, SOAP presents some additional overhead that is not found in the REST approach.

In REST, the use of multiple data types offers some benefits since it adds support for all sorts of platforms. JSON has been widely used in REST architectures when less overhead is needed to exchange large amounts of data and also performs a faster parsing.

Each technology approach has its own characteristics and these both Web services solutions have issues related with security and transport layers. Summarizing, REST presents some advantages in scenarios with limited bandwidth, resources, and client platforms. Then, taking into account the relative performance of both approaches, the Representational State Transfer (REST) was chosen for the proposed solution.

The RESTful architecture is being widely adopted by major technology companies. Most of these companies rely on REST for sharing information and expose Web services and applications. REST architecture is based on client-server communication where clients initiate requests to servers that process these requests and return the appropriate results. These results are defined as resources and represent the information exposed by the service. RESTful architectures are based on HTTP to communicate over the network. HTTP is the Web protocol and it has a set of tools that simplify communications such as Uniform Resource Identifiers (URI), request and response headers, and Internet media types. These functionalities allow the mobile client application to use the HTTP methods GET, POST, PUT, and DELETE to communicate with the Web service and also to exchange information in several file formats such as XML and JSON.

In a mobile environment, 3G Internet connections have several limitations such as bandwidth, speed, and cost. Therefore, it is crucial to minimize data traffic in mobile applications and optimize communication protocols. The use of XML and JSON in Web services is the standard and these file structures are widely supported in both client and server architectures. A uniform file structure was defined for both XML and JSON media types in order to be parsed independently by the mobile application. The *result* tag or name was chosen to define the returned result set by the Web service and the *row* tag is used to define each individual row from the result set.

The historical information presented in the mobile application is stored in a relational database. The database was designed specifically to store all the information about a WSN and also user credentials to enable access control and manage user permissions. The scalability and flexibility were key requirements in the design of the database in order to allow its implementation across multiple server platforms and architectures. The structure of the database is a result of several iterations, from the analysis of different WSNs and the study of available solutions. Based on this analysis, several fields of the database were

defined to store all the relevant data needed to remotely access and monitor a WSN. Some of these fields are the following: MAC address, IP address, GPS coordinates; name, value, unit, and timestamp for each mote parameter; mote manufacturer and country of origin; information about the localization and the environment of the WSN deployment; and information about user credentials.

In order to get the collected data from the 6LoWPAN WSNs and store it in the database, a gateway is needed. The gateway can have more than one IPv6 interface and, at least, one 6LoWPAN interface to allow the communications between the regular IPv6 node and the WSN. The requests destined to WSN nodes are forwarded to one of the IPv6 interfaces and then sent to the 6LoWPAN adaptation layer. The 6LoWPAN adaptation layer is responsible for the packet fragmentation and reassembly in order to support the IPv6 minimum MTU, and for IP and UDP header compression. The gateway communicates with the 6LoWPAN WSN through IEEE 802.15.4. Sensed data received by the gateway is stored in the database through a JDBC connection over an SSH tunnel.

The proposed architecture joins generic messages in XML and JSON, and well-defined REST interfaces to build a communication protocol that enables clients and servers to exchange messages in a platform-independent way. A push notification system, in the context of WSNs, is also proposed and it allows mobile users to receive alerts without requesting them to the server continuously. Besides, the user can also access real-time data instead of getting latest collected values from the database. For real-time monitoring, the mobile application sends requests over the Internet to a HTTP service running on the gateway computer that queries the WSN and responds directly to the smartphone. The real-time HTTP service acts as an intermediary between the 6LoWPAN wireless sensor network and the IPv4-enabled smartphone.

4. Construction of the proposed model

The proposed model architecture was constructed in a real environment with all the needed components for a full WSNs monitoring solution. In this section, the development process of the server-side components is described, including a database, a Web service, a push notifications system, and an end-to-end connectivity between mobile devices and sensor nodes.

4.1. Database design

Based on the requirements analysis, the MySQL Database Management System (DBMS) was chosen for data storage [23]. MySQL provides scalability, flexibility, and is open source. It has support for almost all the operating systems and also provides drivers, plugins, and connectors for the majority of platforms and programming languages.

In the context of WSNs, the performance of the database is an important issue due to the large amount of data collected by sensors. The number of records and queries increases exponentially depending on the frequency of sensor readings. Tables and relations were defined according to the requirements analysis based on several WSNs and available monitoring solutions. Reducing redundant and null values was a main requirement in order to optimize performance and consistency.

The database structure is based on eleven related tables that can be divided into three groups according to the type of stored data. The entity-relationship diagram of the database is illustrated on Fig. 2. The tables *group*, *user*, and *credential* are grouped together because of the relation to user credentials and their permissions. The *user* table stores information about user credentials, such as username and password.

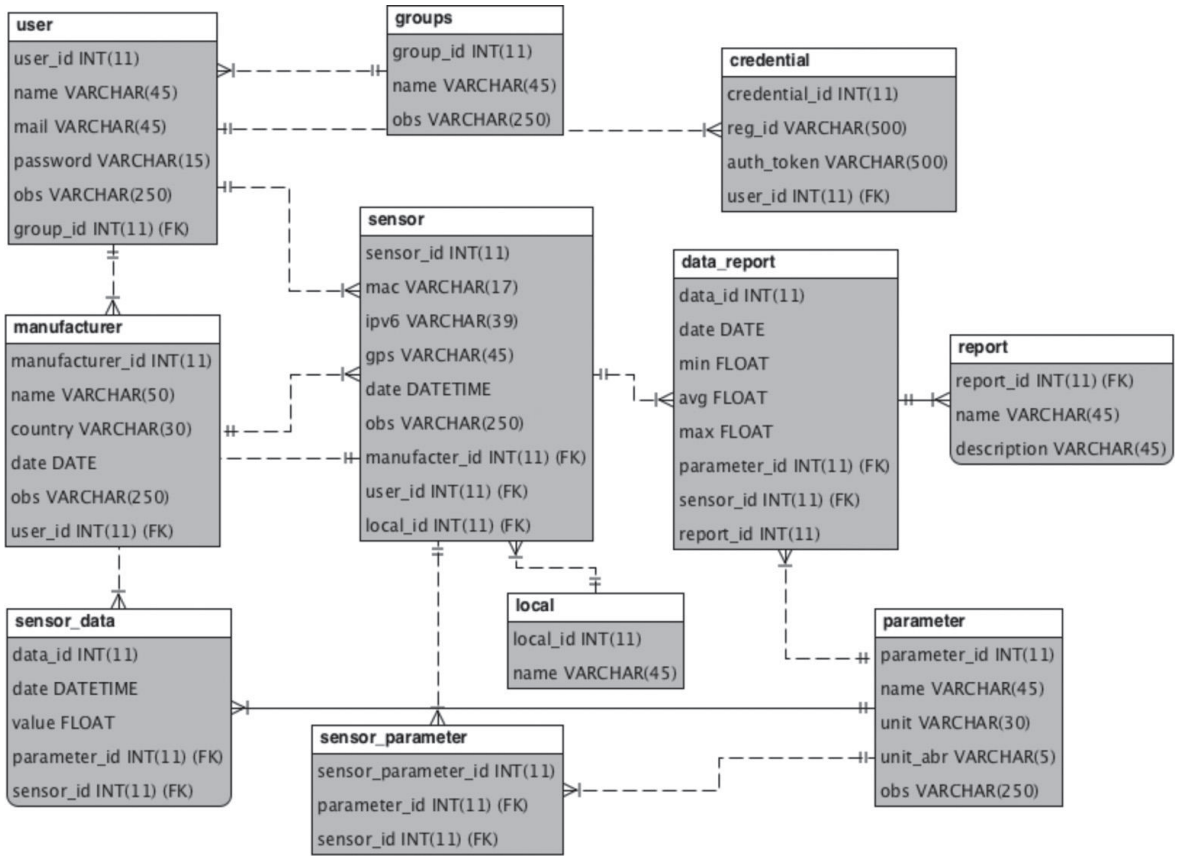

Fig. 2. Database Entity-Relationship diagram.

The *group* table stores information about groups of users that have different access permissions to available WSNs while the table *credential* stores the registration ID and the authentication token that enables the mobile application to receive push notifications. The following tables represent another group: *manufacturer, local, sensor, sensor_ parameter*, and *parameter*. These tables store information about sensor nodes, their location, specifications, and the parameters available in each mote. The *manufacturer* table holds information related to each mote specifications as well as the mote manufacturer.

The *local* table stores information about the geographical location of the WSN. Similarly, *sensor* and *parameter* tables store data related to each individual mote such as the mote's IP address, GPS coordinates, and the type of sensor parameters available at each mote. The third group of tables is formed by the tables *sensor_data, report*, and *data_report* that store the collected data by the sensors for each parameter as well as the maximum, minimum, and average values for each sensor parameter grouped by day, month, or year. The data stored in the table *data_report* are added through the MySQL event scheduler that calculates the minimum, maximum, and average values for each sensor parameter. These events are scheduled to work after each day, month, and year.

4.2. RESTful web service

To enable the information exchange between the database and the mobile devices, a RESTful Web service was created. The Web service presents a modular architecture and generic deployment in order

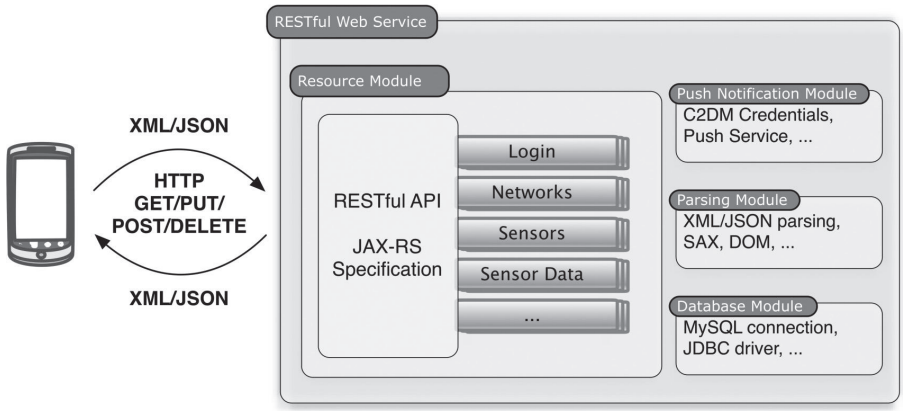

Fig. 3. RESTful Web service architecture.

to be scalable and accessed through several mobile platforms. A RESTful Web service can be defined as a set of resources, available through HTTP interfaces accessed using well-defined HTTP methods, such as GET and POST. Therefore, any client device with an Internet connection and HTTP support can send requests to the Web service in a complete platform independent way. In the construction of the Web service the Jersey open-source framework [15] was used. Jersey is the reference implementation for the JAX-RS specification [14] provided by the Java EE 6. It implements the annotations presented on the specification providing a Java API for RESTful Web services development. The modular architecture of the Web service provides scalability to the entire model and it can be further expanded by adding new modules and functionalities without changing the existing ones.

The Web service considers a four-tier approach with the following modules: the database module, the resource module, the parsing module, and the notifications module. The Web service architecture may be seen in Fig. 3. The database module manages all the connections to the database using the Java Database Connectivity (JDBC) API. The RESTful resources are available in the resource module and a unique Uniform Resource Identifier (URI) identifies each resource. A representation of the current state of each resource and a data format known as media type are used to exchange information in the RESTful environment. For example, the following uniform resource identifier (URI), "http://[server-ip]/rest/sensors" is used to request a list of the available sensors in a chosen WSN. Several unique resources were defined in the Web service, such as, the networks and sensors resources, the historical data resource, and the sensor value resource. When the Web service receives a request of an available resource, an SQL query is sent to the database and, then, the parsing module of the Web service converts the result set into the requested media type and returns the resource representation to the client. The parsing module is responsible for the construction and deconstruction of all the documents and objects exchanged between the Web service and the mobile application. When the mobile application sends an HTTP request to the Web service, the requested media type is specified in the payload to inform the Web service and the parsing module about the data type to return. This parsing module is able to create and parse XML and JSON documents based on the requested media type. For example, the XML structure is defined based on the result set returned by the database, using the names of the tables and columns to define each XML tag.

The <result> tag defines the result set returned by the server and the <row> tag is used to define each individual row of the result set. All the other tags derived from the name of each attribute in the database.

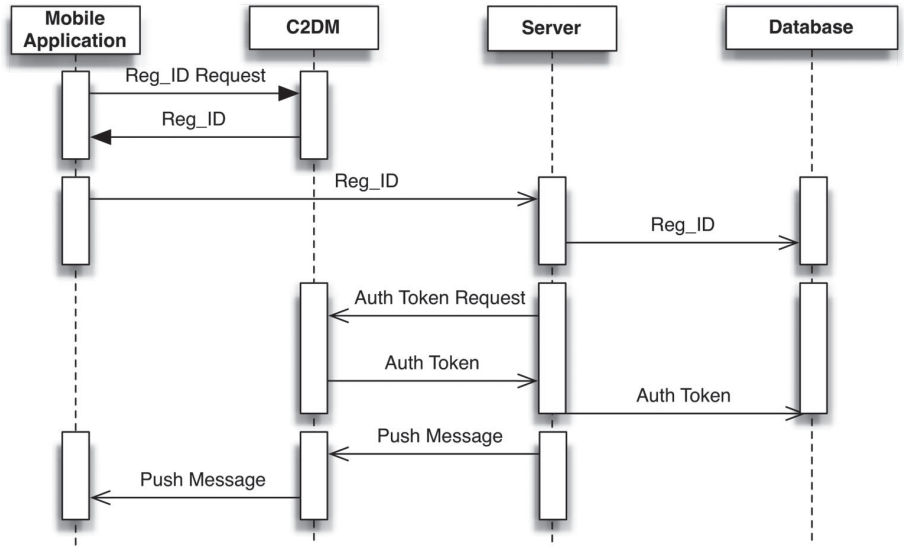

Fig. 4. Sequence diagram of the push notification system.

Thus, an example of an XML file with information about the last temperature reading on a given sensor node would be the following:

```
<?xml version=”1.0”?>
<result>
    <row>
        <value>27.14<value/>
    <row/>
<result/>
```

4.3. Push notifications

One of the main features of the proposed WSN monitoring architecture is the push notification system. This technology is capable to send messages to the mobile device without constantly poll the server for updates. Polling the server for updates has several well-known issues including the adjustment of the requests frequency, energy consumption, and high data traffic. These limitations have more impact when working with mobile devices due to limited battery life and network coverage.

The push notification system is focused on the Android operating system because it is an open platform and integrates very well the RESTful Web service and the mobile application. A new technology called Cloud to Device Messaging (C2DM), developed by Google was used in the construction of the push notification system [5]. The C2DM technology is part of the Android platform and provides libraries and APIs for developing push-enabled applications.

The integration of push notifications in the WSN monitoring model enables the mobile Android application to receive messages and alert the user when a sensor reading overcomes a predefined threshold. The push messaging technology follows a publish/subscribe model where mobile users register at the server in order to receive push messages on the mobile device. Figure 4 presents a sequence diagram of the push notification system.

To use C2DM, the mobile application must register at the Google authentication servers using a Google account. Then, a unique registration ID is generated by the authentication server and sent to the mobile device. This registration ID is forwarded to the Web service and stored in the database. The Web service also requests an authentication token that is used to send notifications to registered mobile devices. Both device registration and server authentication must be completed before sending push messages. Since the push notification system is integrated in the RESTful Web service, it polls the database continuously and when a sensor reading overcomes a predefined threshold, a push message is sent to Google C2DM servers and then to mobile clients. By doing this, the computational costs of polling are on the server side instead of stressing the mobile devices, resulting in significant energy and bandwidth savings. The push notification module was developed in Java language in order to be platform independent as well as the Web server. Therefore, several classes were added to the Web service to support three main functionalities: receiving the registration ID from mobile devices, server authentication and sending push messages.

4.4. End-to-end connectivity

In order to access real-time data from the wireless sensor networks, a multiplatform software application was developed and deployed in the gateway enabling the mobile users to request sensor measures. When the mobile client application requests data from the 6LoWPAN WSN, the software application deployed at the gateway handles the HTTP request and retrieves data directly from the WSN using the UDP transport protocol. The requested data is transmitted through IEEE 802.15.4. Then, the gateway application converts the collected data to XML format and forwards it to the smartphone over HTTP using an IEEE 802.11g wireless network with Internet connection. On the client side, the Android application parses the received data and presents it to the user. With this solution, the access to the database is not needed and the mobile application could request, on demand, data directly from the WSN bypassing the database. This performs the information exchange between the WSN and the smartphone even faster because there is no need to query the database for information. On the other hand, it is not possible to request historical sensors data.

5. Android application

The Android OS is an open mobile operating system, developed and supported by Google. It was built from scratch, specifically for mobile devices and is based on Linux kernel. The Android platform is supported by a wide range of mobile devices, from smartphones to tablets. The Android System Development Kit (SDK) provides libraries and APIs that enable developers to create Android applications and take advantage of hardware capabilities available on the devices using Java programming language. Through the APIs, developers can use functionalities such as text messaging or accelerometers in order to build richer and immersive applications. Since Android is an open platform, it integrates well with emerging technologies and Web services.

5.1. Android User Interface

The user interface was designed following the Android User Interface Guidelines [6] in order to be consistent with the operating system interface and other Android applications. A user-friendly and organized interface was a main requirement in the design process. The application follows two Android

Fig. 5. Login screen.

Fig. 6. Data visualization screen.

navigation guidelines: tabbed navigation and hierarchical navigation. Three fixed tabs at the top of the screen represent the tabbed navigation and within each tab the relationship between different screens is hierarchical.

The initial screen of the application is the login screen, as may be seen in Fig. 5. In this screen, the user must authenticate with username and password in order to access the application's main functionalities. If the user authentication is successful, a new screen is presented with the tab bar at the top where the user could choose between three tabs: Sensors tab, History tab, and Settings tab. By default, the Sensors tab is selected and presents a list view with the available WSNs for the current user. On this screen, if the user selects one of the available WSNs, a new list view is loaded with the mote names that belong to the selected network. When a unique mote is selected, the main data visualization screen is presented under the tab bar as shown in Fig. 6. This screen displays the latest sensor readings of the selected mote in both numerical and graphical modes. Following Fig. 6, "1" indicates the selected Sensors tab, while "2" points out the numerical data presentation where four sensor parameters are presented as well as each parameter unit. If more than four sensor readings are available for each mote, the user can select the four sensor readings to present simultaneously on the screen by using the button indicated by "3". At the bottom of the screen, sensed data is presented graphically as indicated by "5". Using the button pointed out by "4", the user may choose the parameter to visualize in the graphical representation. These parameter readings are presented as a line graph that is updated as new values are received by the mobile device. The graph line represents sensed values over time and the y-axis scale is constantly adjusted in order to center vertically the line graph.

If the user switches to the History tab, a new screen is shown, which presents an interface that allows the user to choose a time interval as indicated by Fig. 7. When the user defines a time interval, a

Fig. 7. Historic data screen.

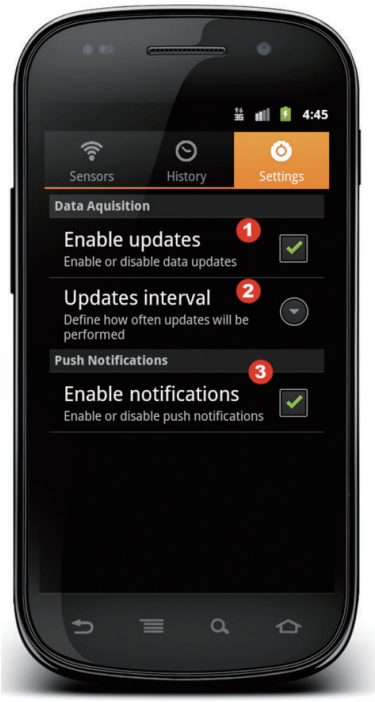

Fig. 8. Settings screen.

full-screen line graph is displayed in landscape orientation with all the sensor readings for the current parameter in that time interval. The chosen time interval may be days, hours or only a few minutes. Following Fig. 7, "1" indicates the selected History tab. Labels "2" and "3" represent the buttons that are used to define the limits of the time interval while "4" points out the button to draw the line graph.

The default application settings may be changed in the Settings tab as shown in Fig. 8. The settings menu provides the following application options: enable or disable data updates; define the updates frequency; and enable or disable push notifications. Following Fig. 8, label "1" points out the option used to enable or disable data updates. This option is enabled by default and is used to reduce battery consumption and data traffic. The frequency of data updates can be defined from 1 second to 1 minute by using the option indicated by "2". When the data updates are disabled, this option is blocked automatically. Label "3" points out the option that allows the user to enable or disable push notifications.

6. Performance evaluation and demonstration

In order to evaluate and demonstrate the architecture and mobile application, a 6LoWPAN wireless sensor network laboratory testbed was constructed. In the design of the 6LoWPAN WSN, several Telos B motes running the TinyOS operating system were used [31]. This network may be seen in Fig. 9. These motes communicate through IEEE 802.15.4 and the 6LoWPAN protocol stack is provided by the TinyOS Blip 1.0 implementation. The motes are capable of sensing air temperature and humidity, luminosity and battery voltage readings. A 6LoWPAN gateway is used to provide IPv6 end-to-end connectivity between the sensor network and the Internet. The 6LoWPAN gateway runs on Ubuntu 10.0.4 and it has multiple communication interfaces technologies, including IEEE 802.15.4, Ethernet and IEEE 802.11a/b/g. To

Table 1
Relation between the downloaded data, battery consumption, and missed updates with changing polling rates and push notifications

Polling rate	3 sec	10 sec	30 sec	Push	Idle
Downloaded data/hour	501 KB	149 KB	49 KB	301 KB	_
Battery consumption/hour	19%	13%	9%	4%	2%
Missed updates	0 %	50.4%	83.6%	_	_

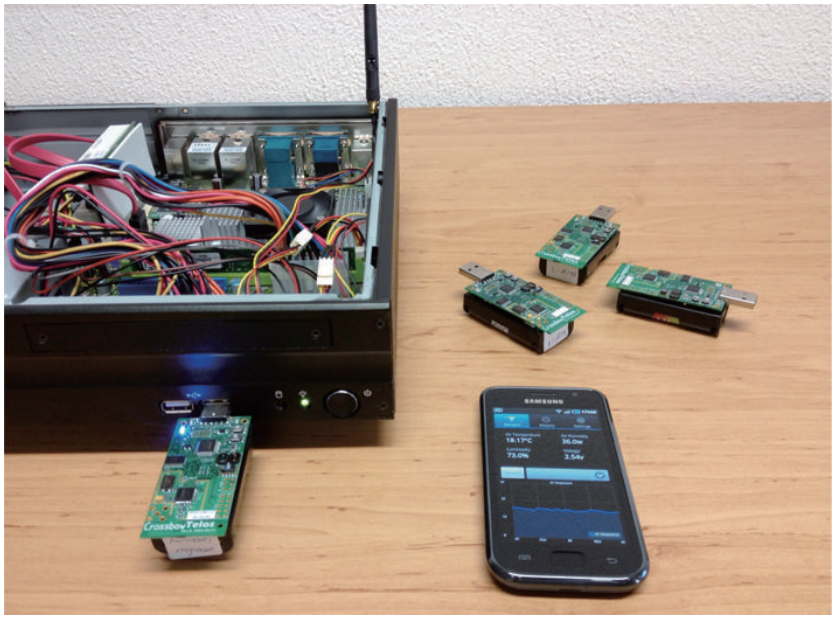

Fig. 9. 6LoWPAN wireless sensor network laboratory testbed.

implement the IEEE 802.15.4 interface in the gateway device, a TelosB mote connected to an USB port was used. An Intel desktop board D945GCLF with a 1.6 Ghz Intel Atom processor has been used to be the motherboard of the gateway. The application IP-driver compliant with RFC 4944, provided by TinyOS 2.1, acts as the 6LoWPAN adaptation layer in the gateway. The 6LoWPAN gateway is also responsible for sending ICMPv6 router advertisement messages to announce the IPv6 prefix and the default gateway address to all sensor nodes.

The smartphone used to evaluate the proposed architecture and mobile application was the Samsung Galaxy S, running Android 2.3 with a 1.0 GHz CPU and a Li-Ion 1500 mAh battery. All the tests were performed over an IEEE 802.11 g connection with Internet access. During the experiments, the phone only used essential core services, the Wi-Fi adapter and the constructed application in foreground.

Measuring energy consumption on mobile devices is not easy. There are several factors that influence the energy consumption in a mobile device, specifically a smartphone. The energy consumption is different from device to device and it also depends of the operating system version and network specifications. For each experiment, the Android battery manager was used to check the current battery level that runs from 100% when fully charged to 0%.

Table 1 shows the relation between the amount of downloaded data, the battery consumption and the missed updates for a varying polling rate with a fixed update rate. In Table 1, “Polling Rate” refers to the time interval between two requests from the mobile device to the RESTful service. A polling rate

equals to "push" means that the mobile device did not request data to the RESTful service, but waited for notifications from the push service. "Downloaded data/hour" refers to the amount of data the RESTful service transferred to the mobile device in one hour. "Battery consumption/hour" refers to the amount of energy consumed in the mobile device battery in one hour. "Missed updates" refers to the updates that were sent by the WSN gateway to the relational database but were not retrieved by the mobile device. The update rate of the wireless sensor network values was fixed in 5 seconds for testing purposes while the polling rate of the Android application varies between 3, 10 and 30 seconds. Furthermore, the same experiment was conducted with push notifications and also with the smartphone in idle for comparison purposes. All the experiments were performed during 1 (one) hour of monitoring for each polling rate.

As expected, the lower polling rate presents the lower battery consumption of the smartphone, but more updates missed by the monitoring application. The experiments also shown that if the polling rate is lower than the update rate of the WSN, none of the updates is missed but the amount of downloaded data is very high resulting in higher energy consumption. On the other hand, if the polling rate is too high, the amount of downloaded data and the battery consumption are reduced but the percentage of missed updates is also high. When the push notification system is used, the energy costs are significantly reduced. If push notifications are enabled, the mobile application is not constantly sending requests to the server and checking if there are any updates resulting in significant energy saving. As a result, ubiquitous wireless sensor networks monitoring is much more energy efficient when push technologies are used on mobile client applications.

The end-to-end connectivity between the smartphone and the 6LoWPAN WSN was also experiment in detail. Real-time temperature readings were collected and presented successfully in the mobile application.

7. Conclusion and future work

This paper proposed a ubiquitous wireless sensor networks monitoring solution allowing users to receive latest sensor readings as well as historical measures on their smartphones. The architecture was designed to be modular and was constructed based on open standards to ensure scalability and reusability. Since it is based on REST interfaces and XML/JSON messaging, the architecture is platform independent and supported in the majority of current mobile devices.

A push notification system was constructed specifically for mobile devices and is able to send push messages to smartphones if a sensor reading overcomes a given threshold. The smartphone application is also able to access real-time data over the Internet through a gateway software application. The proposed architecture was evaluated and demonstrated using a real wireless sensor testbed and an Android mobile application. The experiments showed that the solution work as planned and the push notification system has a significant impact on smartphone's energy savings.

As future work, the proposed solution may be deployed in a real environment, deploying the wireless sensor network testbed outside the laboratory. In an outdoor environment, factors such as energy management, security and weather conditions should be considered. Furthermore, the mobile application could be extended to other mobile platforms such as the iPhone and Windows Phone. With respect to storage of sensor data, NoSQL solutions may be adopted, such as document-oriented databases. Furthermore, the development of algorithms to search and locate sensor resources in the proposed REST environment may be considered.

Acknowledgments

This work has been partially supported by the *Instituto de Telecomunicações*, Next Generation Networks and Applications Group (NetGNA), Portugal, by National Funding from the FCT – *Fundação para a Ciência e Tecnologia* through the Pest-OE/EEI/LA0008/2013, and by the AAL4ALL (Ambient Assisted Living for All), project co-financed by COMPETE under FEDER via QREN Programme.

References

[1] A. Hornsby, P. Belimpasakis and I. Defee, XMPP-based wireless sensor network and its integration into the extended home environment, in ISCE, IEEE 13th International Symposium on Consumer Electronics, May 25–28, 2009.
[2] A. Ludovici, A. Calveras and J. Casademont, Forwarding Techniques for IP Fragmented Packets in a Real 6LoWPAN Network, *Sensors* **11**(1) (2011), 992–1008.
[3] A. Munawar, A. Masood and F. Bangash, Open sensor platform: Integration of sensors and mobile phones, in IBCAST, International Bhurban Conference on Applied Sciences and Technology, January 9–12, 2012.
[4] Adobe Flex Framework, http://www.adobe.com/devnet/flex.html.
[5] Android Cloud To Device Messaging. http://developers.google.com/android/c2dm/. Accessed Jan 2012.
[6] Android User Interface Guidelines, http://http://developer.android.com/guide/practices/ui_guidelines/index.html. Accessed Jan 2012.
[7] B. da Silva Campos, J.J.P.C. Rodrigues, L.M.L. Oliveira, L.D.P. Mendes, E.F. Nakamura and C.M.S. Figueiredo, *Design and construction of a wireless sensor and actuator network gateway based on 6LoWPAN*, in EUROCON, International Conference on Computer as a Tool, April 27–29, 2011.
[8] C. Alcaraz, P. Najera, J. Lopez and R. Roman, Wireless Sensor Networks and the Internet of Things: Do We Need a Complete Integration? 1er International Workshop on the Security of The Internet of Things, 2010.
[9] Cosm, https://cosm.com/. Accessed Mar 2013.
[10] D.S. Tudose, A. Voinescu, M. Petrareanu, A. Bucur, D. Loghin, A. Bostan and M. Tapus, *Home automation design using 6LoWPAN wireless sensor networks*, in DCOSS, International Conference on Distributed Computing in Sensor Systems and Workshops, June 27–29, 2011.
[11] F. Belqasmi, R. Glitho and F. Chunyan, RESTful web services for service provisioning in next-generation networks: a survey, *IEEE Communications Magazine* **49**(12) (2011), 66–73.
[12] I.F. Akyildiz, S. Weilian, Y. Sankarasubramaniam and E. Cayirci, A survey on sensor networks, *IEEE Communications Magazine* **40** (2002), 102–114.
[13] J. M. Corchado, J. Bajo, D.I. Tapia and A. Abraham, Using Heterogeneous Wireless Sensor Networks in a Telemonitoring System for Healthcare, *IEEE Transactions on Information Technology in Biomedicine* **14**(2) (2010), 234–240.
[14] Java API for RESTful Services. http://jax-rs-spec.java.net/. Accessed Nov 2011.
[15] Jersey Open-source Framework. http://jersey.java.net/. Accessed Nov 2011.
[16] L. Atzori, A. Iera and G. Morabito, The Internet of Things: A survey, *Computer Networks* **54**(15) (Elsevier, 2010), 2787–2805.
[17] L. Herrera, B. Mink and S. Sukittanon, Integrated personal mobile devices to wireless weather sensing network, Proceedings of the IEEE SoutheastCon, March 18–21, 2010.
[18] L.M.L. Oliveira and J.J.P.C. Rodrigues, Wireless Sensor Networks: a Survey on Environmental Monitoring, *Journal of Communications* **6**(2) (2011), 143–151.
[19] L.M.L. Oliveira, A.F. de Sousa and J.J.P.C. Rodrigues, Routing and mobility approaches in IPv6 over LoWPAN mesh networks, *International Journal of Communication Systems* **24**(11) (Wiley, 2011), 1445–1466.
[20] L. Mottola and G.P. Picco, *Programming Wireless Sensor Networks: Fundamental Concepts and State of the Art* **43**(3), ACM Computing Surveys, 2011.
[21] M. Cardei, A. Marcus, I. Cardei and T. Tavtilov, *Web-based heterogeneous WSN integration using pervasive communication*, in IPCCC, IEEE 30th International Performance Computing and Communications Conference, November 17–19, 2011.
[22] M. Nottingham and R. Sayre, *The Atom Syndication Format*, RFC 4287, December 2005.
[23] MySQL Database Management System. http://dev.mysql.com/doc/. Accessed Out 2011.
[24] N. Aslam, W. Phillips, W. Robertson and S. Sivakumar, A multi-criterion optimization technique for energy efficient cluster formation in wireless sensor networks, *Information Fusion* **12**(3) (Springer, 2011), 202–212.
[25] N. Moreira, M. Venda, C. Silva, L. Marcelino and A. Pereira, @Sensor – Mobile application to monitor a WSN, in CISTI, 6th Iberian Conference on Information Systems and Technologies, June 15–18, 2011.

[26] P. Sanyal, S. Das, S.S. Bhunia, S. Roy and N. Mukherjee, An experience of implementing IPv6 based data retrieval system for Wireless Sensor Networks, on RACSS, International Conference on Recent Advances in Computing and Software Systems, April 25–27, 2012.
[27] P. Vajsar and L. Rucka, Monitoring and management system for wireless sensor networks, 34th International Conference on Telecommunications and Signal Processing, August 18–20, 2011.
[28] P. Wang, Z. Sun, M.C. Vuran, M.A. Al-Rodhaan, A.M. Al-Dhelaan and I.F. Akyildiz, On network connectivity of wireless sensor networks for sandstorm monitoring, *Computer Networks* **55**(5) (Elsevier, 2011), 1150–1157.
[29] R. Kemp, N. Palmer, T. Kielmann and H. Bal, Energy Efficient Information Monitoring Applications on Smartphones through Communication Offloading, *Mobile Computing, Applications, and Services* **95**(2) (Springer, 2012), 60–79.
[30] R.T. Fielding, REST: architectural styles and the design of network-based software architectures, Doctoral dissertation, University of California, Irvine, 2000.
[31] Tiny OS Documentation Wiki, http://docs.tinyos.net/tinywiki/index.php/. Accessed Feb 2012.
[32] V. Kumar and S. Tiwari, Routing in IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN): A Survey, Journal of Computer Networks and Communications, 2012.
[33] Z. Li, Y. Liu, M. Li, J. Wang and Z. Cao, Exploiting Ubiquitous Data Collection for Mobile Users in Wireless Sensor Networks, *IEEE Transactions on Parallel and Distributed Systems*, Issue 99, 2012.

Luís M. Oliveira (loliveira@ipt.pt) is a PhD student of Informatics Engineering at the University of Beira Interior, Covilhã, Portugal, under supervision of Professors Joel Rodrigues and Amaro de Sousa. He received his 5-year BS degree (licentiate) in Electronics in 1998 and the MSc degree in Electronics and Telecommunications Engineering in 2004, both from the University of Aveiro. He also teaches at the Polytechnic Institute of Tomar, Portugal. He is a PhD student member of the Institute of Telecommunications, Portugal. His research interests include routing on wireless sensor mesh networks. He has authored or co-authored over ten papers in international refereed journals and conferences.

Joel J. P. C. Rodrigues (joeljr@ieee.org) is a professor at the University of Beira Interior (UBI), Covilhã, Portugal, and researcher at the *Instituto de Telecomunicações*, Portugal. He received a PhD degree in informatics engineering, an MSc degree from the University of Beira Interior, and a five-year BSc degree (licentiate) in informatics engineering from the University of Coimbra, Portugal. He is the Director of the Master degree in Informatics Engineering at UBI. He is the leader of NetGNA Research Group (http://netgna.it.ubi.pt), the Vice-chair of the IEEE ComSoc Technical Committee on Communications Software, the Vice-Chair of the IEEE ComSoc Technical Committee on eHealth, and Member Representative of the IEEE Communications Society on the IEEE Biometrics Council. He is the editor-in-chief of the International Journal on E-Health and Medical Communications, the editor-in-chief of the Recent Patents on Telecommunications, and editorial board member of several international journals. He has been general chair and TPC Chair of many international conferences. He is a member of many international TPCs and participated in several international conferences organization. He has authored or coauthored over 250 papers in refereed international journals and conferences, a book, and 2 patents. He had been awarded the Outstanding Leadership Award of IEEE GLOBECOM 2010 as CSSMA Symposium Co-Chair and several best papers awards. Prof. Rodrigues is a licensed professional engineer (as senior member), member of the Internet Society, an IARIA fellow, and a senior member of ACM and IEEE.

André Gaudêncio F. Elias is an MSc student of Informatics Engineering at the University of Beira Interior under the supervision of Prof. Joel Rodrigues and Prof. Bruno Zarpelão. He received his 3-year BS degree in Informatics Engineering from the University of Beira Interior, in 2010. In 2011, he studied in the University of Campinas, Brazil, during one MSc semester. He is an MSc student member of the Next Generation Networks and Applications Group (NetGNA) at Instituto de Telecomunicações, Portugal. His main research areas include wireless sensor networks, and mobile and ubiquitous computing.

Bruno B. Zarpelão received his B.S. degree in Computer Science from State University of Londrina, Brazil, and the Ph.D. degree in Electrical Engineering from University of Campinas, Brazil. He is currently a professor at the *Pontifícia Universidade Católica do Paraná* (PUC-PR), Londrina, Brazil and an associate researcher at the Next Generation Networks and Applications Group (NetGNA), University of Beira Interior, Covilhã, Portugal. His research interests include Smart Cities, eGov, Open Access MAN, Communication Network Management and Information Security.