Journal of Network and Computer Applications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support

Paulo A. Neves [a,b], Joel J.P.C. Rodrigues [a,*], Min Chen [c], Athanasios V. Vasilakos [d]

[a] Instituto de Telecomunicações, University of Beira Interior, Portugal
[b] Polytechnic Institute of Castelo Branco, Castelo Branco, Portugal
[c] Seoul National University, Seoul, Korea
[d] National Technical University of Athens (NTUA), Greece

ARTICLE INFO

Article history:
Received 7 July 2010
Received in revised form
26 December 2010
Accepted 30 March 2011

Keywords:
Wireless sensor and actuator networks
Data gathering on WSANs
IPv6
6LoWPAN
Plug-and-Play
ContikiOS

ABSTRACT

Wireless sensor and actuator networks provide a distributed system composed of wirelessly connected smart sensor and actuator nodes, suitable for cost-efficient control applications. An important research challenge is deployment, where features like node auto-configuration, unattended operation, and Internet connectivity are becoming mandatory. Moreover, on off-the-shelf solutions the user typically must be network technology-savvy to take advantage of sensing and actuation services. This paper presents a novel multi-channel architecture for sensor data gathering and actuation, featuring Plug-and-Play like functionality for node attachment and operation, IPv6 at the node level, and dedicated communication semantic protocols—the ZenSens system architecture. The architecture features the sensor/actuator nodes, a personal computer application (SenseLab), a mobile application (SenseLab mobile), and World Wide Web access (WebSensor), presenting the user with a complete sensing and actuation solution. As a result the user can operate the network without technological background, and near-zero configuration. All developed software and firmware are presented, and validated through a series of experiments on real hardware, namely using a test-bed with TelosB motes running ContikiOS.

1. Introduction

Wireless sensor and actuator networks (WSANs) present a distributed environment of wirelessly connected nodes that can present sensing and/or actuator control capabilities, thus making them suitable for cost-efficient deployments (Xia et al., 2007). Research community has centered much of its efforts on wireless sensor networks (WSNs) (Baronti et al., 2007), leaving WSANs as a relatively new area of research. Typically WSNs are considered for many control applications at the sensing level, leaving the control and actuation planes for other systems to perform.

A WSAN is also composed of small smart nodes that communicate wirelessly with processing core and memory, a battery, one or more base stations (sinks), and sensor nodes. However, WSANs introduce actuator nodes, small smart nodes similar to sensor nodes, but with actuation capabilities. This small difference brings a new plethora of applications and challenges. Small smart nodes can have sensors, actuators, both or none. Sensors capture information, actuators make decisions and take actions based on the input from sensors, and sinks monitor overall network, communicating with both sensor and actuator node types to reach network's goals (Rezgui and Eltoweissy, 2007).

A WSAN can also benefit from Internet connectivity, exposing its sensing services and actuation information worldwide. Two main approaches can be considered when Internet connectivity is mandatory, namely proxy-based and node stack-based (Rodrigues and Neves, 2010). WSN/WSAN with Internet connectivity makes ubiquitous computing realistic (Stankovic, 2008), turning sensor and actuator networks into ubiquitous networks. The node stack-based approach has been gaining popularity (Hui and Culler, 2008; Yang et al., 2008; Silva et al., 2008; Han and Ma, 2007), namely with support from two of the major operating systems for embedded objects—TinyOS (Levis et al., 2004) and ContikiOS (Dunkels et al., 2004).

This paper proposes a system architecture for WSANs, called ZenSens, with user-centric perspective, namely featuring different software access channels, and zero-user configuration. As a result the user can operate and take advantage of the network's services without technological background. The architecture supports Plug-and-Play (PnP) from the ground up, where nodes present themselves to the sink, with automatic network attachment, in a hassle-free and transparent way. With the ZenSens system a

* Corresponding author at: Instituto de Telecomunicações, University of Beira Interior, 6201-001 Covilhã, Portugal.
E-mail addresses: pneves@co.it.pt (P.A. Neves), joeljr@ieee.org (J.J.P.C. Rodrigues), minchen@ieee.org (M. Chen), vasilako@ath.forthnet.gr (A.V. Vasilakos).

1084-8045/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2011.03.033

user can deploy and automatically monitor a WSAN through different channels, putting the user in control without digging into WSAN technology, mobile or personal computer technologies. ZenSens enables the user to focus on the physical system to be controlled, obtaining results in an effective way.

PnP in WSNs/WSANs (Sung et al., 2009) is a fundamental aspect of network operation, enabling automatic configuration of nodes, abstraction from the attachment process, while contributing for dynamic topology designs. Another added benefit is also the capacity to integrate new hardware platforms without system changes or existing node reprogramming.

The software layers have knowledge of network condition and assets through a USB-connected sink that manages and coordinates connected remote nodes. For user access, the system presents a multi-channel approach: computer-based (SenseLab), mobile-based (SenseLab mobile), and Internet-browser based (WebSensor). The personal computer application is also in charge of distributing information obtained from the WSAN to the other two channels through XML files. The mobile application brings portability and convenience to the solution, while World Wide Web access takes the system into a global scale.

The WSAN level is based and demonstrated on a test-bed using TelosB motes (Polastre et al., 2005), running the Contiki operating system. All motes are IPv6-enabled through uIPv6 (Durvy et al., 2008), using the current implementation of 6LoWPAN available on Contiki (sicslowpan). This "future-proof" approach enables seamless Internet connectivity of network nodes.

The remainder of the paper is organized as follows. Section 2 presents some background and related work, while Section 3 elaborates on the ZenSens system architecture. Section 4 presents USB and UDP communication, node's firmware, and PnP support, while Sections 5 and 6 detail SenseLab and SenseLab mobile applications. Section 7 presents WebSensor web solution, while Section 8 elaborates on system tests and validation. Finally, Section 9 presents the conclusions and planned future work.

2. Background and related work

This section elaborates on the system's technological base—IPv6, 6LoWPAN, and ContikiOS—and presents motivation for this work. IPv6 brings several benefits to embedded smart objects that require seamless Internet connectivity (Silva et al., 2008).

The 6LoWPAN specification enables the transmission of IPv6 packets over standard IEEE 802.15.4 networks with support for header compression (Montenegro et al., 2007). This specification defines frame format for transmission of IPv6 packets, establishment of local-link addresses, and stateless auto-configuration. IEEE 802.15.4 defines physical and link-layer communication for small embedded devices, suitable for WSN/WSAN. Moreover, since IEEE 802.15.4 is present in the majority of sensor hardware (motes), the application of 6LoWPAN is almost mandatory for IPv6 node stack approaches. Furthermore, a very recent book presents 6LoWPAN and its benefits in several application scenarios, helping the adoption of IPv6-enabled sensor networks (Shelby and Bormann, 2009). Another work presents a study on performance of 6LoWPAN implementation over TinyOS, with TelosB and MICAz motes, using ICMP payloads (Cody-Kenny et al., 2009).

Contiki is an operating system for embedded smart objects, realizing the vision of "The Internet of Things", by enabling IP communication on very constrained smart sensor node. It uses ANSI C, as opposed to TinyOS nesC, and it is currently adopted by many research teams worldwide. Among the assets of Contiki, it is the uIPv6 communication stack that enables IPv6 communication for smart embedded devices, with MAC and link-layer agnostic features. Namely, uIPv6 stack can potentially run over IEEE 802.15.4, Ethernet, and IEEE 802.11.

Contiki supports an event-driven model with a form of multiprocessing designated as protothreads. Protothreads present a multiprocessing-like environment, but with a shared stack system, thus resulting in a memory-efficient alternative to threads. However, since all protothreads share the same stack, care must be taken with internal variables. Two types of events can be used—system and in-program defined. Event timers are also available, enabling support for periodic events. Protothread processing may be event-driven, thus turning it into an event handler.

The choice of ContikiOS over TinyOS was based in several parameters. ContikiOS presents almost no learning curve for C programmers, when compared to TinyOS, which uses nesC as the development language. When considering IPv6 implementation, ContikiOS is pioneer, offering a set of tools and the uIPv6 stack. Third TelosB, the current test-bed hardware platform, is very well supported in Contiki. Features like the availability of documentation, a pre-configured virtual machine and open source code are common with TinyOS, although we found the virtual machine in ContikiOS, instant-Contiki over Ubuntu, to be more adequate than TinyOS over XUbuntos. Although ContikiOS documentation is far from perfect, the discussion lists are simply phenomenal with real help from the code contributors, including ContikiOS' creator.

Authors believe IPv6 paves the way for the Future Internet, by enabling not only hosts, but also small smart devices to be part of Internet, realizing the "Internet of Things" vision. In this regard 6LoWPAN efforts may accelerate the adoption of IP-enabled WSN designs and deployments. IP-enabled WSN research has been focused on simulation-based approaches, use of test-beds for performance assessment, some real deployments, and commercial solutions. However, application layer, namely user interaction, has been overlooked. To the best of the author's knowledge, no other solution presents multi-channel visualization tools, PnP-like operation, and support for IPv6-enabled WSANs.

Developing over real sensor devices can be difficult and tedious (Ramanathan et al., 2005; Köpke et al., 2008), since unlike personal computer programming, programs must be sent to the target hardware, which is a time consuming "non-productive" task. This time consuming process may lead to simulation-based approaches, thus shortening the development cycle. On ContikiOS two tools that can be used for simulating node behavior, namely COOJA (Österlind et al., 2006) and MSPsim (Eriksson et al., 2009, 2007). The latter one can be used for stand-alone debugging on MSP430 processor-based boards such as TelosB, while COOJA can simulate complete networks. Nevertheless, real hardware deployments always provide a deeper insight, enabling test-bed environments, clearly presenting a better match to real scenario conditions.

The current approach is a step-up on previous work on WSNs (Neves et al., 2010a,b), taking the architecture and assets further to actuator networks. The system is based on a WSN, and as a result does not feature actuator assets. Also the UDP communication inside the network was fully revised on this new version, separating node attachment from the data gathering process. The inclusion of actuators also implied changes on the node's firmware, SenseLab, and WebSense, with the introduction of new features.

The current work is evaluated and demonstrated through the implementation on real devices (motes, computers, and mobile devices), debugging and throughout testing, with development assist from simulation software at early stages. Our motivation is based on bringing WSN/WSANs into mainstream use, namely finding "killer applications" for a technology that is still used on very specific applications, with Internet connectivity to achieve the vision of "Internet of Things". Applications stem from smart

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033

homes, localized monitoring, and actuation (e.g. greenhouse control, machinery control, among others).

Available low-cost components for sensor motes turned accessible the presence of multimedia sensors on the traditional WSNs, including CMOS cameras and microphones (Akyildiz et al., 2007). These wireless multimedia sensor networks include sensor devices that may retrieve multimedia content (images, audio, and video) from the environment performing ubiquitous solutions offering context-aware multimedia services (Zhou et al., 2010a). Furthermore, video streaming may also be supported using distributed scheduling schemes that jointly addresses problems related to channel-assignment, rate allocation, routing, and fairness problems for video streaming over multi-channel multi-radio networks (as wireless multimedia sensor networks), as proposed in Zhou et al. (2010b). The proposed architecture also supports multimedia services over wireless sensor and actuator networks.

Some related approaches can be identified in data gathering for WSN, namely with some data processing on the network itself (Al-Yasiri and Sunley, 2007; Yu et al., 2004; Burri et al., 2007). In terms of related tools for WSN data presentation and management, one can point one open source solution on Contiki, and two solutions from commercial vendors, namely MoteView from Crossbow (2009) and NodeView Pro from Sensinode (2009). The ContikiOS tool, named Contiki-collect features a Java application for sensor data gathering and visualization with Sky/Telos(B) motes. Neither it does support IPv6 at the time of writing, nor multi-channel visualization. However, due to the open source nature it provides a valuable tool for ContikiOS developers.

MoteView features wireless sensor network management and data gathering in a graphical way, with network topology view. Other tools like MoteConfig and XSniffer, and the availability of a dedicated XMesh routing stack API, further complement MoteView. However, Crossbow tools do not provide multi-channel support, nor they can be used directly on IPv6-enabled motes or even non-Crossbow motes.

NodeView Pro is a Java-based tool with sensor network real-time view of node statistics, network topology map updates, network performance assessment and 6LoWPAN support. However, the approach is network-centric, and as far as site information goes no sensed data plotting is performed, nor the tool is compatible with non-sensinode hardware. Moreover, as in the previous approaches, multi-channel access is not contemplated.

On the current work we aim at improving user experience in WSANs, offering multi-channel tools for fast deployment and immediate data gathering on several scenarios, namely on body sensor networks (Neves et al., 2008; Singh et al., 2009) or wireless multimedia sensor networks (Akyildiz et al., 2007; Zhou et al., 2010a,b), with the added convenience of IPv6 at the sensor node level.

3. System architecture

This section elaborates on the system architecture that involves the WSAN, data management, and application levels. This architecture is based on several identified functionalities, leading to a complete system for data storage, monitoring, and WSAN management, demanding minimal user technological knowledge. It also presents the foundations for control applications to be integrated on the existing system, by enabling a sink interface with a personal computer, which can be in charge of applying a suitable control over the existing actuators, based on sensed data. Moreover, smart sensor node firmware may be programmed for a specific control scenario. SenseLab can also be extended for control purposes, providing a real environment for a plethora of applications, from environmental monitoring to smart spaces.

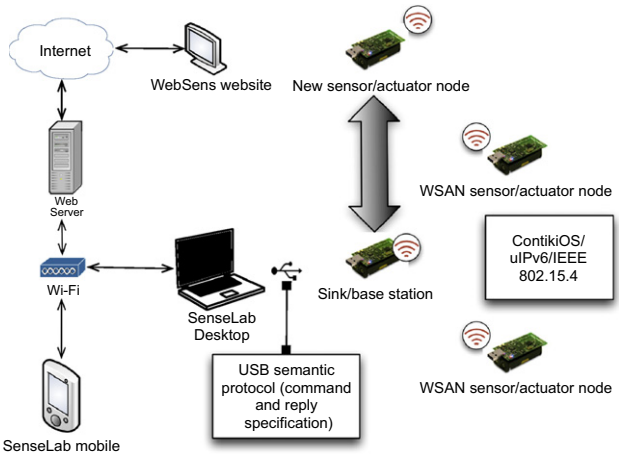

Fig. 1. ZenSens system architecture.

Figure 1 depicts ZenSens system architecture, composed of WSAN level, where sensor/actuator nodes reside, the local monitoring and control level, and the remote monitoring level. The local monitoring and control level is composed of a personal computer running a Java application—SenseLab. Communication with the WSAN level is performed through USB to a sink node. Unlike other sensor/actuator designs that rely on a more powerful sink node, ZenSens uses the same constrained hardware for the sink. The USB interface, together with a dedicated message mechanism (see Sections 4.1 and 4.2 for details), enables SenseLab with the tools required to monitor and control the WSAN level on behalf of the user, with the added benefit of a Graphical User Interface (GUI). The architecture enables multi-sink, through the use of multiple USB ports.

On the WSAN level the sink coordinates a given set of nodes, achieving the desired network goals. The sink and the remote smart nodes run the ContikiOS with IPv6 support through uIP6. Several protothreads were developed to achieve the desired behavior of both node types (see Sections 4.3 and 4.4 for more details), namely communication, data management, and actuation control. WSAN nodes communicate through IEEE 802.15.4 wireless communication, with 6LoWPAN on top to achieve the desired IPv6 compatibility. Smart remote nodes can be only sensor, only actuator or both. Moreover, the sink can also be enabled with sensors and actuators.

The SenseLab Desktop level is the primary interface for the network administrator. It is divided into several areas, namely data acquisition and presentation, actuator data monitoring and control, data communication with the server, and settings. Data acquisition and presentation, the main SenseLab Desktop functionality, is achieved through USB communication with the sink (data acquisition) and a GUI to display data with several visualization options, namely chart format. Actuator data monitoring and control is also achieved through USB communication, sending sink commands accordingly.

Data communication with the server is achieved through sockets, based on a pre-defined set of XML schemas—"networks", "deviceID", and "networkIDtime". Networks XML presents all networks currently attached, showing for each network its ID, name, time, and location. From this XML the software requests the state of network with a certain ID, the networkIDtime XML. This XML has all the information about the network assets at time "time", namely the network nodes with all sensor and actuator

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033

data. Finally the "deviceID" XML contains information about a device profile, namely the correspondence of sensors and actuators specified with abstract generic identification $s1,\ldots,sn$ and $a1,\ldots,an$ on the "networkIDtime" XML into meaningful names like temperature, pressure, LED, among others. This simple mechanism of separating sensor and actuator details from the network description XML presents storage space and bandwidth savings, while enabling hardware abstraction.

The settings part enables the user to define hardware profiles that match the real installed hardware in terms of sensors and actuators on a node. This setting together with the network mechanisms provides hardware abstraction at the WSAN level, where everything is numbered, avoiding communication of symbolic names of sensors and actuators.

The remote monitoring level enables off-site access to collected sensor and actuator data, namely through handheld device (through native application, Windows Mobile and iPhone) on SenseLab mobile, or through a standard Internet Browser such as Firefox or Safari (WebSens). Although enabled with monitoring capabilities, this level is not yet able to access the control features of SenseLab Desktop. Data is obtained through SenseLab Desktop, that pushes data onto the server, while also servicing nearby handheld devices (on iPhone version data is collected directly from the server).

4. Wireless sensor and actuator Level

This section presents the WSAN level of ZenSens, namely USB and UDP communication, sink and remote node firmware, and PnP support. The WSAN layer is the lowest layer of the system, and as such must expose its assets to the PC application, namely number of nodes, sensors and actuators, and use the sink as the "outside world" interface.

4.1. USB communication

Two main communication schemes were developed—wireless inside the WSAN (UDP) and wired (USB) for WSAN interface with the outside world. Two protocols were developed, specifying the semantics of commands that the base station can process and possible replies, both in the wired and wireless world.

Both communication schemes use simple command/response architecture. In USB the command is issued by the PC application to the sink node. Commands are summarized in Table 1, and have two contexts or scope—sink ("BS" — Base Station) commands and remote smart node ("SN" — Smart Node) commands.

Several commands were defined, from obtaining the node's IPv6 address ("ip" command), obtain the current node's sensing values ("sens" command), power (transmission level — "txp", and current consumption — "power"), "reset" that removes all remote nodes information, and the new "setact" and "getact" commands that enable actuation value programming and readout. Finally "info" returns node's current hardware characteristics (mote type, number of sensors, and actuators). Timers setting, such as transmission period of wireless communication from remote nodes to the sink ("txtime"), and sensor-sampling period ("sample"), are also available.

Some commands are only related to the "BS" (sink) context, namely commands that return network status, such as "list" that lists all current network node's IDs, the number of nodes present in the network—"mn"— and "mtl" that returns the mote's types. The "ident" command is dedicated to remote nodes enabling remote node visual identification, by toggling the mote's green led with a 2 Hz frequency for 10 s.

With a plethora of possible commands, the need to identify error conditions is mandatory. As a result seven possible error states were defined, which can occur when interfacing the sink node. These are as follows: "NAC", Not A Command; "CER",

Table 1
USB commands.

Serial command		**Purpose**
Context	*Command*	
BS/SN	ip	Return the node's IPv6 address
BS/SN	sens	Retrieve last sensor's readings, separated by "|"
BS/SN	txp	Retrieve currently programmed IEEE 802.15.4 radio transmission power
BS/SN	power	Retrieve current node power consumption
BS/SN	reset	Clear data structures. BS, all structures; SN, specific node's structures. Returns OK
BS/SN	setact	Set a given actuator value
BS/SN	getact	Get the value programmed into a given actuator
BS/SN	info	Get node type, number of sensors and number of actuators present in the node
BS/SN	sample	Set new sampling period in seconds. If issued on "BS" all nodes are reprogrammed
BS/SN	txtime	Set new transmission period in seconds (only applicable to remote nodes). If issued on BS all remote nodes are reprogrammed
SN	ident	Command used to "find" a remote node in the network by blinking the green LED for 10 s at 2 Hz. Returns "OK" if node exists. Only available if the remote node type supports it, such as Crossbow TelosB nodes
BS	list	List the current network mote's ID, separated by ":"
BS	mn	Return the current number of motes present in the network
BS	mtl	Returns the list of mote types, separated by ":", including the sink
Server reply		**Meaning**
No reply		Sink not connected—wrong USB port, port has other device or sink was physically disconnected
OK		The normal reply from the sink on selected commands that do not provide information, e.g. "reset"
NDA		No Data Available. Command produced no results, e.g. getact when no actuator data is available
NAC		Not a command—signals command not recognized
CER		Command Error—signals command error
INE		Inexistent Node Error—SN context command refers to inexistent node ID
IAE		Inexistent Actuator Error—the referred actuator does not exist in the current node
NSA		No Sensors Available—referred node has no sensors
NAA		No Actuator Available—referred node has no actuators

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033

Table 2
UDP communication commands and replies

Message	Parameters	Purpose
Client to server		
M	Mote type, number of sensor fields and actuators	Client requests WSAN attachment
D	Sensor values	Client sends sensor data to the server
A	Actuator values	Client sends actuator data to the server
Server replies		
NSA		No storage available on the sink to accept node
NAS		Node attachment successful
MFE		Message format error
NNR		Node not recognized
NAA		Node already attached
Server to client commands		
SA	Actuator values to be programmed	Server sends new actuator values to actuator-capable node
TS	Timers setting	Server sends new timer's value to remote node
ID	None	Server requests identification function
GA	None	Server requests actuator values

Command ERror; "INE", Inexistent Node Error; "NDA", No Data Available; "IAE", Inexistent Actuator Error; "NSA", No Sensors Available; and "NAA", No Actuators Available. The first two ("NAC" and "CER") are signal errors on the commands. An "NAC" signals that a non-valid command was received, and a "CER" that the command is correct, but parameters were not correctly given. "INE" is used on the "SN" context and signals that the provided nodeID does not exist, while "IAE" signals that the requested actuator does not exist on the specified node. "NDA" signals that the required information is not available. Finally, since nodes can have only actuators, only sensors, or none, the "NSA" reply signals that the node has no sensors, and "NAA" that the node has no actuators.

If correctly attached to the computer, the sink always replies to a received command, which is used to identify sink nodes from other USB connected devices. The data parameters are speed 115,200 bps, 8 data bits, 1 stop bit and no parity check.

4.2. UDP communication

The sink acts as a UDP server and the remote nodes as UDP clients. Two main communication phases were identified—node attachment and data communication (sensorial and actuation data). A dedicated message format was developed for clients and server communication, as summarized in Table 2.

Communication initiative is always at the client side, and only attached nodes can send sensor and actuator data to the server. In the attachment process the client sends a type "M" message to the server, stating the mote type (a number that corresponds to a hardware profile), the number of sensor fields (may be more than number of sensors, since some sensors may use floating point notation, e.g. temperature), and the number of actuators. Since a node can be actuator, sensor or both, two other message types were defined: "D" for sensor data and "A" for actuator data. By default a sensor and actuator node sends sensor data, sending only actuator data when queried.

The server replies are used to signal normal and error conditions. Since the server will implement a remote node's data cache on its RAM, the "NSA" reply informs the new remote node that it cannot be part of this network or sub-network, due to space restrictions on the sink. "NAS" is the expected outcome of an attachment process, while "MFE" signals that the received message format is not as expected. "NNR" signals a condition where a node tries to send data without being properly attached to the current sink, whereas "NAA" signals that the node is already attached when it tries to send an "M" message.

Moreover, the server may also need to issue commands to the remote node, issuing commands as replies to client communication. One of the four requests can be made, "SA", "TS", "ID", and "GA". "SA" is used by the server to indicate the need to set one or more actuators, while "TS" informs the remote need that it needs to update its timers values. "ID" issues the identification function explained before, and finally "GA" is used to obtain current node's actuator values.

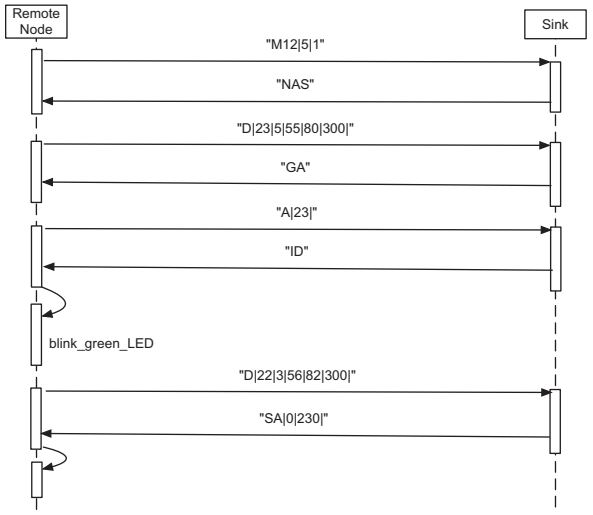

Fig. 2. UDP communication example.

The server commands may be triggered upon requests made by the PC to the sink node, which afterwards triggers the necessary actions on the network. This mechanism enables role separation between the WSAN coordination and management (sink's role), and data/control functions' monitoring and visualization (SenseLab Desktop).

Figure 2 shows a UDP communication example between a remote smart node that wants to join the WSAN and the sink. In the attachment process the node informs the sink that it has hardware profile 12, with five sensors and 1 actuator. As can be seen, message fields are separated by "|" for easier parsing. After successful attachment (received an "NAS" reply), the node starts sending sensor readings periodically through the "D" type message. In the example the sink replies with a request for actuator data ("GA"), which forces the remote node to change the next datagram to actuator data ("A" type). The remote smart node sends actuator data, and sink replies with a request for the identification function ("ID"), which sets an event on the remote node to start flashing the green LED for 10 s. Eventually the remote node sends another datagram with sensor data (type "D")

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033

and receives a command from the sink to enable actuator 0 with the value 230 (SA|0|230).

4.3. Sink firmware

The sink (coordinator) firmware is WSAN's most complex firmware. The firmware is divided into several Contiki protothreads and includes the data structures to store node data to be readily available when queried. Although this "centralized" approach may lead to scalability issues, most complex networks use a multi-sink or hierarchical approach, thus enabling network expansion.

The data structure is also a fundamental part of the PnP approach used in this work. By enabling different sets of sensors and actuators to be part of a given hardware platform, the WSAN processes data without specific knowledge of what it really means, delivering sensing and actuator services to the upper software layers. The option for a linked list system is justified by the expected dynamic nature of the PnP WSAN, where nodes attach dynamically to the network. The linked list uses a data structure that enables node information cache on the sink, namely sensor and actuator data, node type and status, node's IPv6 address, node's ID (sink managed), number of sensors, and number of actuators.

Both server and client firmware are organized into protothreads. The server has five protothreads: serial_command_parser, udp_server, set_actuation, read_sensors, and set_actuators and init, totalizing 860 lines of code (LoC). The init protothread initializes data structures, starts other protothreads and then terminates. The serial_command_parser protothread is responsible for USB communication, receiving commands and replying accordingly, as shown in Fig. 3. The protothread is waiting on a serial event, starting with the analysis of the context as soon as data is available, and then searching for a command. Commands are separated into "complex" and "simple"—a complex command needs parameters, while a simple one does not. As a result further processing must be made on a complex command.

The udp_server_protothread implements the UDP server according to the message formats described previously. When a

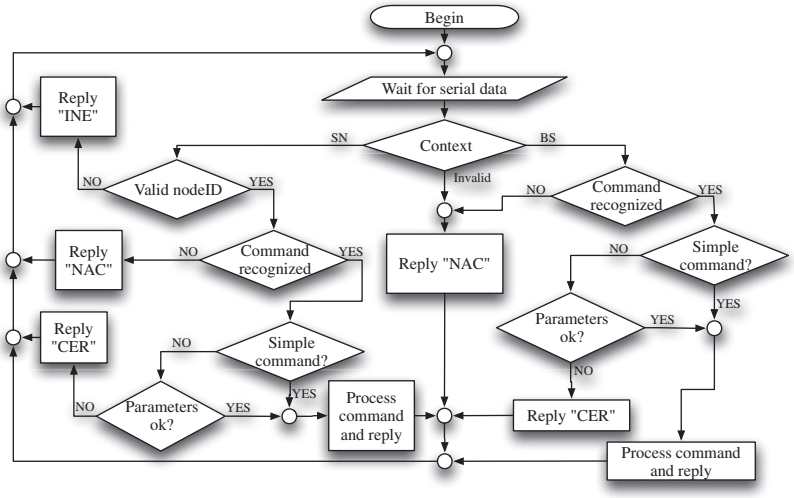

Fig. 3. Serial_command_parser protothread actions.

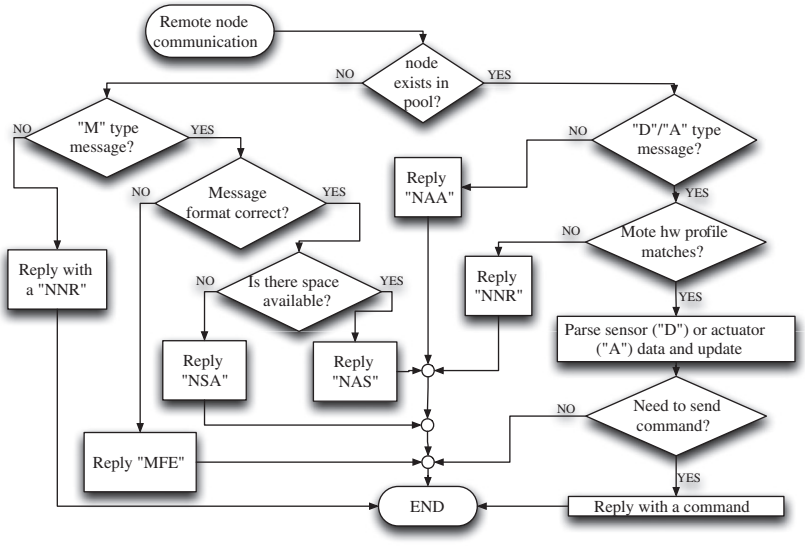

Fig. 4. Sink actions taken based on client communication.

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033

client sends a UDP message to the sink, the sink acts according to Fig. 4. Since each node will has an IPv6 address, the sink uses such information to detect if node already exists in the current pool of attached nodes, by comparing IPv6 addresses.

The protothreads set_actuation and read_sensors enable the sensor and actuator roles inside the nodes. The set_actuation protothread is responsible for actuator processing based on sink input. This protothread is also event-driven, but the event is triggered when an actuator needs to be set (software event). The read_sensors is periodic and controlled by an event timer that dictates the transmission period.

4.4. Remote node firmware

Remote node firmware is able to handle all types of possible remote nodes—sensor, actuator or both—based on the same scheme as the sink, using two protothreads. Another similarity with the sink firmware is the init protothread, which has basically the same behavior. The defined protothreads result in approximately 450 LoC.

However, since remote nodes communicate exclusively through IEEE 802.15.4, the serial_command_parser protothread is absent, while UDP communication is handled by the udp_client protothread, shown in Fig. 5. This protothread uses an event timer for periodic processing. If the node is attached, depending on node's assets and status a sensor or actuator data message is formed and sent to the server. If the mote is not attached, it must request attachment, sending an "M" message. This protothread is also responsible for server reply processing, namely processing server commands.

The nodes have knowledge of sink's IPv6 address hardcoded in the firmware, and use local-link addresses (IPv6 address starting with FE80). UDP is used since 6LoWPAN specification clearly states that TCP header compression is not supported, and TCP on WSN is not desired due to the end-to-end acknowledge mechanism.

In terms of program memory the sink firmware occupies 45,522 bytes from the 48 KB pool, while the sensor and actuator firmware takes 42,464 bytes. According to MSPsim simulator, sink's CPU usage in stand-alone goes around 5%, while remote node's CPU goes around 36%. This difference is related to the udp_client protothread that is periodically sending data through UDP, while the sink is idle accepting connections.

4.5. Plug-and-Play support

Plug-and-Play supports start at the WSAN level on the node's firmware (sink and remote nodes). We face PnP as the mechanism that allows automatic node attachment to the network, where node adverties its capabilities, enabling autonomous operation.

As described earlier, the attachment process uses three information fields: mote type, number of sensor fields, and number of actuators. This information enables the sink to create an information entry for the new node to store sensors and actuators data, status, and mote data such as IPv6 address. Sink's firmware data structure is critical to accommodate different hardware, and to provide support for network heterogeneity. By using an array to store sensor and actuator values inside the data structure, any given mote that needs up to the defined capacity can join the network. Since typically constrained nodes are related to a bounded number of sensors and actuators per node, we chose five of each to be a reasonable value. If RAM is enough, this number can be raised, even at the cost of lower number of maximum nodes per sink (currently ten).

Based on attachment information, the sink is able to manage the node, namely identifying if it can accept actuator data and/or can provide sensing data. The mechanism to identify a given node is based on the remote nodes' IPv6 address, comparing the stored IPv6 address on the sink with the sender's IPv6 address. Data from a node that is not recognized as attached is not considered by the sink, which replies accordingly.

After successful attachment, the node can send data. By default a sensor and actuator node sends sensor data periodically (according to the internal transmission timer), only sending actuator data if queried. This mechanism allows autonomous operation of the remote node without any user interaction. Mote type can be set according to Electronic Product Codes (EPCs) specification.

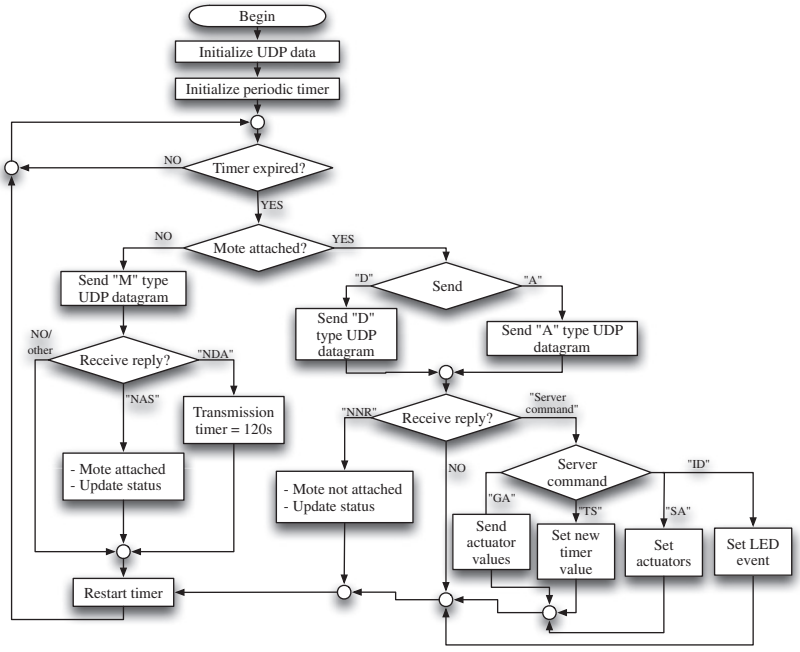

Fig. 5. udp_client protothread actions.

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033

Since the attachment process is hardware-agnostic, any hardware capable of running uIPv6 and support for IEEE 802.15.4 can join a given network. Moreover, the node can operate autonomously without user configuration or intervention.

5. SenseLab application

SenseLab application was developed for a personal computer that is physically connected to the sink(s), using Java programming language. Java guarantees platform independence, running on different operating systems for personal computing. Taking advantage of the virtual machine provided with Contiki, the development was made in Ubuntu Linux 8.04 LTS, and also verified in Mac OS X 10.6.

The application interface is presented in Fig. 6. Part Ⓐ lists all networks found, while part Ⓑ presents the assets of the selected network. At any given time the user may change network name, or select another network. From the Ⓑ view, the user selects a node to monitor, and depending on the node's assets, part Ⓒ may be shown (the sensors/actuators tab), or just the sensors or actuators part. Part Ⓓ presents the sensors or actuators the node has, while part Ⓔ presents information on the asset selected in part Ⓓ. Part Ⓒ also enables node info view, through a dedicated button.

A thread is always searching for possibly connected sinks. Each time a new sink is detected, it creates another thread that is responsible for communicating with the sink according to the protocol presented in Table 1. The application uses two libraries: JfreeChart and JavaComm (RX-TX on Mac OS X). JfreeChart is employed when drawing the 2D graphics, enabling construction of several chart types like histograms, line charts, and pie charts. JavaComm (RX-TX on Mac OS X) is used for USB communication with the sink node. Past data is, in certain situations, more important than current data, so this application also has the functionality to review, in line chart form, past data from both sensors and actuators.

Figure 7 presents the class diagram of SenseLab version 1.5. Class Xmlserver is responsible for implementing an instance of class Connection for each connection established through sockets (mobile devices for instance), while class Connection implements a thread for each connection, accepts requests for XML files and serves as needed. Class MakeCom enables user-defined commands to the sink, while DataTreatment deals with sensor and actuator values parsing, delivering data to other classes that need it. Class XMLParser grabs an array of characters and places it on a user-defined xml schema, while class Popupmenu enables popup for changing some attribute names, such as network names. Nchoose implements the network view and allows selection of a given node for monitoring, while NodeInfo implements node interface and stores node information (this class is instantiated once for each node).

The Main class is responsible for initialization, network choice, graphical elements, and menus. Class NodeReader is responsible for sink communication, it is where USB commands are sent to the sink node, while SinkInfo implements sink interface. Chart classes (ThermoChart, LightChart, PowerChart, TimeChartPanel, BarChartPannel, and HumidityChart) implement the respective graphical representations. Finally, Visualizer has several methods that are used by other classes; Historic implements the interface of historic data chart, and HideSeries is a class for the historic data view that enables hiding data sources from the graph, for instance disable the view of the temperature sensor values. SenseLab is also responsible for managing the XML files, and sending them to

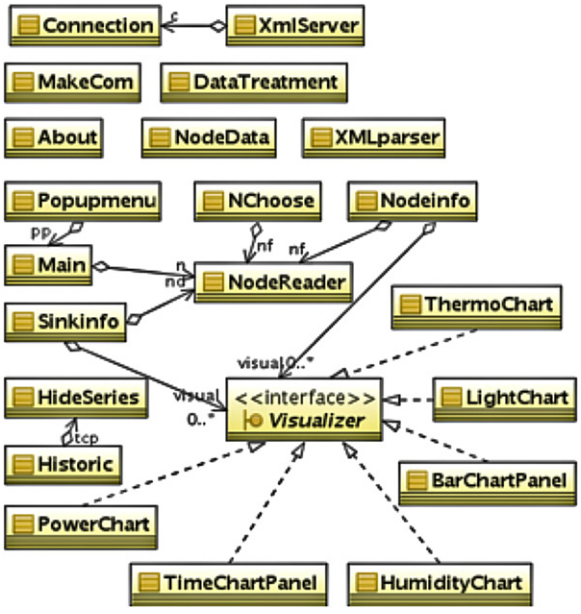

Fig. 7. SenseLab 1.5 class diagram.

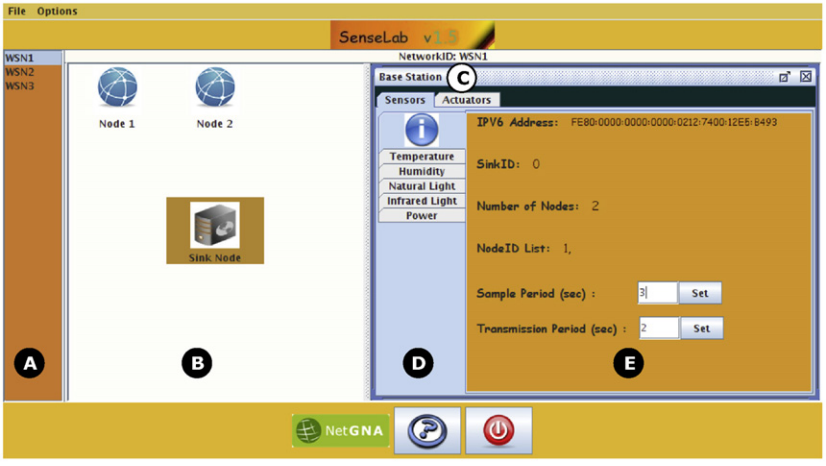

Fig. 6. SenseLab 1.5 Desktop application main view.

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033

the appropriate channels. Each XML file has a time stamp attached to its name, enabling rapid identification. Since connection is made through sockets on a specific port, several clients may be served, e.g. several mobile devices.

6. SenseLab mobile

This section describes the mobile application developed on top of the .NET Compact Framework, using C#. This application enables visualization of both sensor and actuator values in a convenient and adequate way, suitable for the small display and constrained hardware of a mobile device.

The mobile application captures XML data from the personal computer, presenting the detected networks. The user selects one network and a node to visualize current node's attributes, as shown in Fig. 8. Part (a) is the default view when a node is selected, showing node's information—IPv6 address, node type, and node assets, namely sensors and actuators. Part (b) presents the view of sensors, with several different views (number, bar and round gauge) of the detected sensors, and a small point graph of the last received values. Finally part (c) is related to actuators, and presents a similar approach to (b), presenting a listbox with the actuators.

Depending on the node's assets, the sensors and/or actuators tab is selectable. In our example, toggling the green LED of TelosB is the only actuator. At any moment the user may choose another node on the network, or even a node from any of the other detected networks.

The mobile application also enables history view of received values, with time-based search criteria. Figure 9 presents historic data view, namely on part (a) the selected node's information, part (b) the node sensors data and part (c) the node's actuator data. Despite only having an LED as an actuator, the actuator value may be different from the binary values, as the graph shows, since the actuator function on the WSAN may be customized to suit the needs of the connected actuators. In our example if the actuation is 0 the LED is off, and on otherwise. This visualization allows rapid verification of the control function and abnormal operation detection, thus being able to support alarm features if desired.

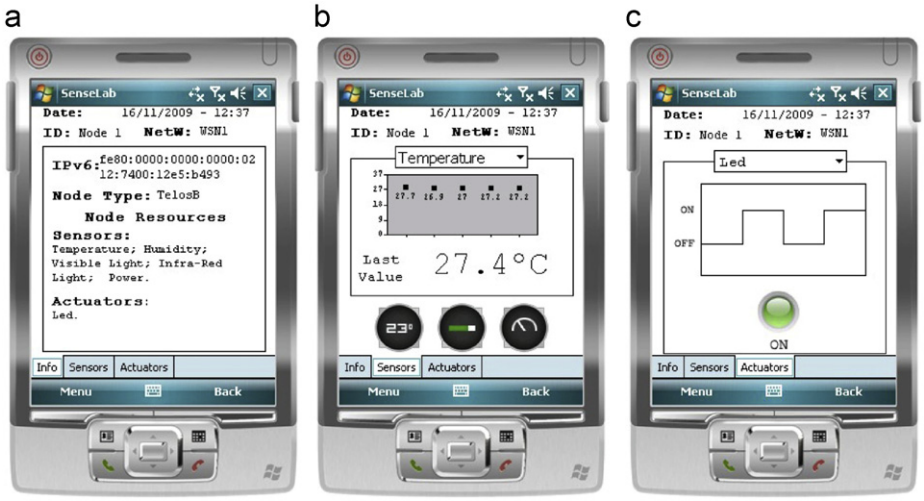

Fig. 8. SenseLab mobile application node visualization.

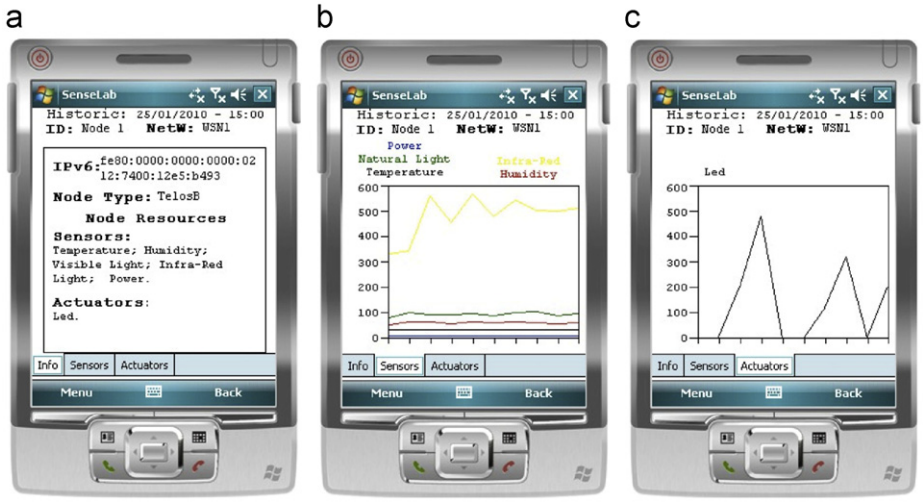

Fig. 9. SenseLab mobile application historic data visualization.

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033

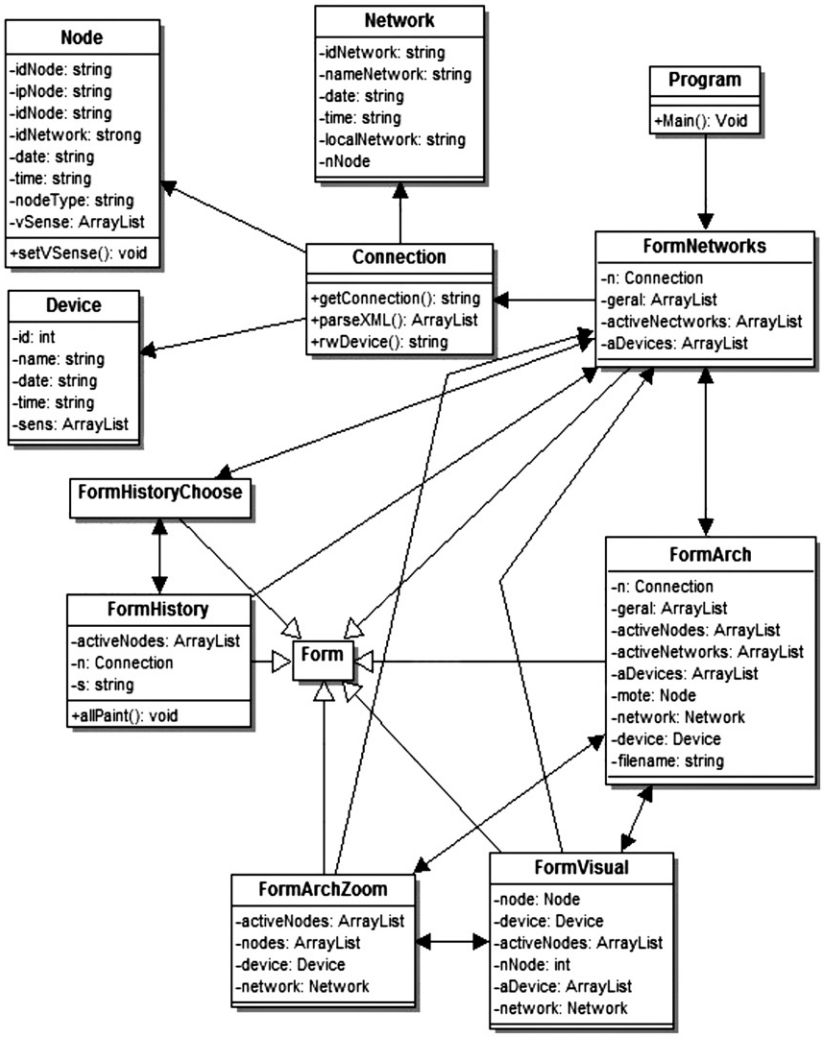

Fig. 10. SenseLab mobile application class diagram.

Figure 10 presents SenseLab mobile application's class diagram. Forms control the guided user interface—FormHistory is responsible for presenting the historical view of data in graphical form (Fig. 9(b) and (c)), FormHistoryChoose enables the selection of a given search criterion of Fig. 9(a), FormVisual presents the information on the selected node, and FormArch presents the current network nodes (sing the Connection class), with FormArchZoom enabling zoom over the network visualization area. The Network class is responsible for receiving and parsing the available networks, using the Connection class that connects to the personal computer application through sockets. The Node class has all nodes' attributes, while Device has device's attributes (sensors and actuators designation, among other data).

A Java application was developed for accepting XML data, parse and transfer into a MySQL local database. This database stores all networks and mote's data, while being accessed by the Joomla! CMS. The database, depicted in Fig. 11 can hold several network's data, along with user authentication and privileges, such as access to certain network's data. The database can hold several networks' data, with user authentication information. With this simple mechanism data access is controlled based on group permissions. The remainder stores sensor and actuator data for a given mote, while it is also possible to associate the hardware with the manufacturer.

Figure 12 shows webpage interface, namely part (a) presents current sensed values, while part (b) presents a historical view.

7. WebSensor—ZenSens Web access

This section presents the Web server application that was developed to disseminate WSAN functionality over the Web: WebSens. Using Joomla! open source dynamic portal and content management system (CMS) for World Wide Web publishing, a suitable Web application for sensor and actuator data visualization was created. The server can accept XML files from several computers and store data on a local MySQL database.

8. Real scenario tests and validation

This section elaborates on system validation by experimenting with the several system layers. By segmenting the test and validation procedures from bottom to top, one can go into the upper layer with confidence about the previous layer functionality.

Starting with the WSAN level and command-line approach to validate the sink's and remote nodes' firmware. On the sink node, the serial communication protocol was tested and validated, by

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033

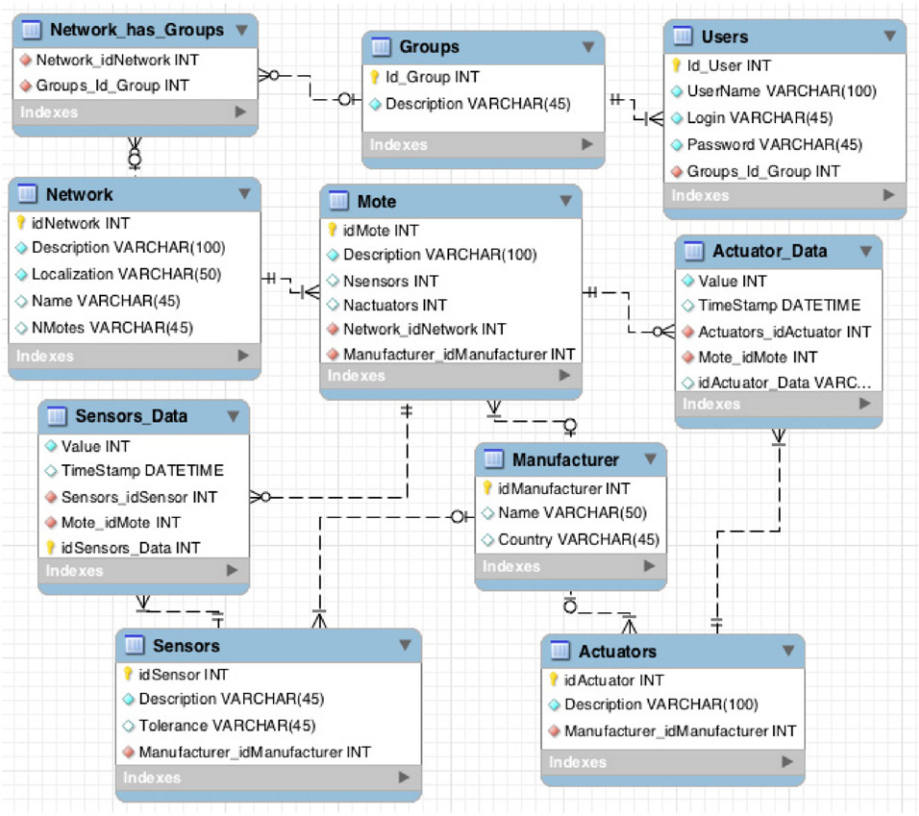

Fig. 11. WebSensor relational database EER diagram.

running a simple program based on serialdump (a tool included in ContikiOS that enables serial communication through USB), and continuously sending commands to a stand-alone sink. To emulate the presence of a single remote node, firmware changes were made to debug without UDP communication.

After successful validation of the serial protocol, focus was given to UDP communication. In this regard, the serial port was used for debugging on both client and server, checking the transmitted and received packets.

We noticed that, from four nodes up, the sink presents some delay in serial command processing. However, we found no erroneous conditions, even when seven motes were used at a transmission period of 2 s (the system default). The LEDs were also used for debugging, namely the blue LED is used for wireless activity (toggling at each sent/received message) and the green LED is used to signal good status (when off it signals a problem was detected). As mentioned previously, the green LED is also used for the identification function on the remote nodes, while the red LED is used for actuator emulation.

The next phase is the introduction of SenseLab. On this application focus was given to features validation and XML file creation and transmission. Tests were issued with the test-bed connected for 24 h and the results logged. By changing sensor readings (increasing temperature, blocking light), system responsiveness and data were checked. Both SenseLab mobile and WebSens draws its data from SenseLab, so testing and validation were done almost in parallel. Figure 13 presents one of the tested scenarios.

For proof of concept, a simple autonomous control system was developed on the sink node, which basically lights the LED of node ID 1 if nodes' temperature is above 25 °C, node ID 2 if humidity is below 80%, node 3 if visible light is below 50, and node 4 if temperature readings from all sensors differ more than 2 °C. The results were logged and verified through a Microsoft® Excel spreadsheet on the personal computer.

9. Conclusions and future work

This paper proposed an approach for IPv6-enabled wireless sensor and actuator networks, featuring multi-channel monitoring tools, Plug-and-Play node operation, and global access to data. We believe that IPv6, namely through 6LoWPAN, will pave the way for adoption of WSANs to power ubiquitous computing and provide sensing and actuator services on a global scale. The WSAN architecture proves that it is possible to create networks with sinks that do not require more computational power than sensor and actuator nodes, paving the way for cost-efficient designs.

This work also enabled us to deal with real devices, and to test and debug with a case study. The firmware present in the WSAN nodes suffered several modifications, while some bugs were corrected on the development phase. In this regard the MSPsim emulator/simulator included in Contiki sped up the development process, specifically on validation of small stand-alone (not wireless communication related) functions and protothreads.

As future work concerns the architecture is going to be expanded through inclusion of more functionalities on both the USB and UDP protocols, and suppression of the portable computer for a dedicated gateway. This approach leads to seamless Internet connectivity as promised by IP over WSN.

SenseLab mobile will be ported to other platforms, namely iPhone, leading to an architectural change, where the mobile device connects directly to the Web server, retrieving data from

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033

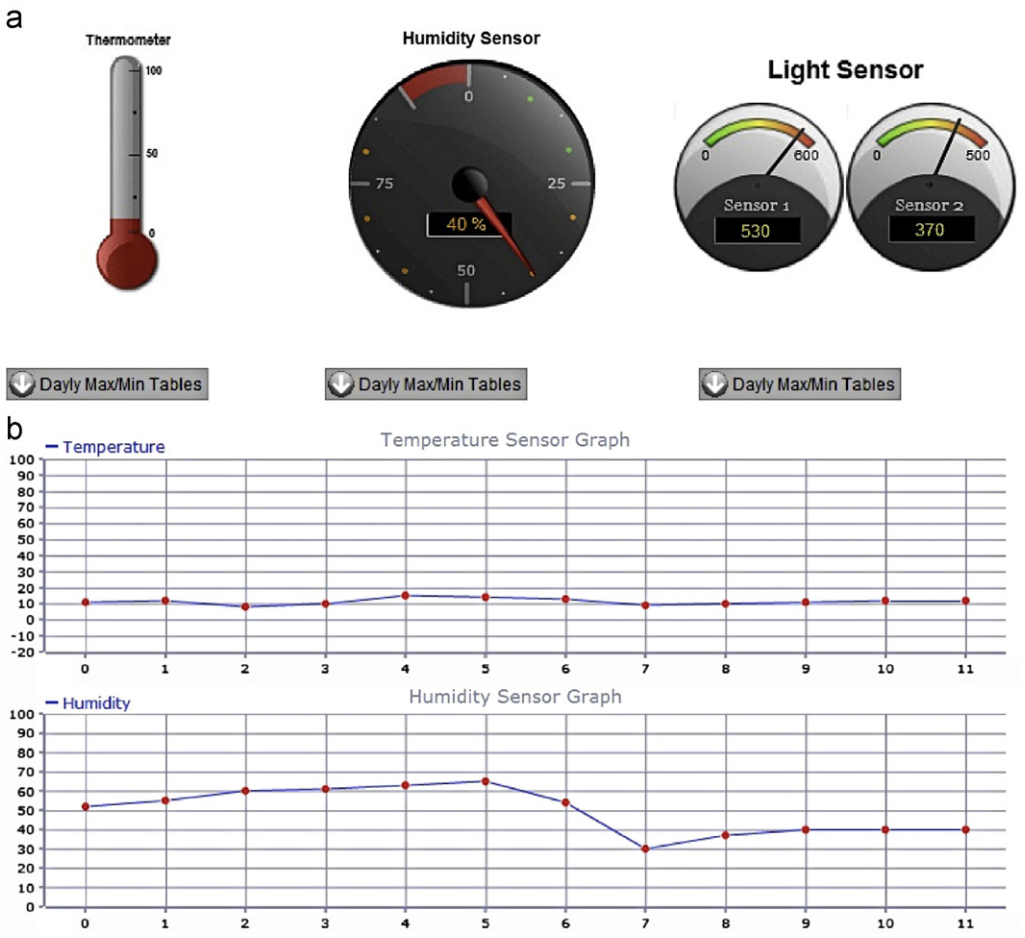

Fig. 12. WebSensor view—(a) current sensor values and (b) historical data.

Fig. 13. Real scenario tests—SenseLab Desktop and Mobile application running.

almost everywhere, removing the limitation of sharing the same Wi-Fi network as the Desktop system.

Acknowledgments

Part of this work has been supported by the Instituto de Telecomunicações, Next Generation Networks and Applications Group (NetGNA), Covilhã, Portugal, in the framework of BodySens Project, and by the Euro-NF Network of Excellence from the Seventh Framework Programme of EU, in the framework of Specific Joint Research Project PADU.

References

Akyildiz IF, Melodia T, Chowdhury KR. A survey on wireless multimedia sensor networks. Computer Networks 2007;51:921–60.

Al-Yasiri A, Sunley A. Data aggregation in wireless sensor networks using the soap protocol. Journal of Physics: Conference Series—Sensors & their Applications XIV (SENSORS07) 2007;76(1).

Baronti P, Pillai P, Chook V, Chessa S, Gotta A, Hu YF. Wireless sensor networks: a survey on the state of the art and the 802.15.4 and zigbee standards. Computer Communications 2007;30(7):1655–95.

Burri N, Rickenbach Pv, Wattenhofer R. Dozer: ultra-low power data gathering in sensor networks. In: Proceedings of the sixth international symposium on information processing in sensor networks, 2007 (IPSN 2007). Cambridge, MA; 2007. p. 450–9.

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033

Cody-Kenny B, Guerin D, Ennis D, Carbajo RS, Huggard M, Goldrick CM. Performance evaluation of the 6lowpan protocol on micaz and telosb motes. In: International workshop on modeling analysis and simulation of wireless and mobile systems. Proceedings of the fourth ACM workshop on performance monitoring and measurement of heterogeneous wireless and wired networks. Tenerife, Canary Islands, Spain; 2009. p. 25–30.

Crossbow. Crossbow moteworks software, 2009. 〈http://www.xbow.com/〉.

Dunkels A, Gronvall B, Voigt T. Contiki—a lightweight and flexible operating system for tiny networked sensors. In: Proceedings of the 29th annual IEEE international conference on local computer networks. IEEE Computer Society; 2004. p. 455–62.

Durvy M, Abeillé J, Wetterwald P, O'Flynn C, Leverett B. Gnoske E, et al. Making sensor networks ipv6 ready. In: Proceedings of the sixth ACM conference on networked embedded sensor systems (ACM SenSys 2008) USA: ACM; Rayleigh, NC; 2008.

Eriksson J. Dunkels A. Finne N. Österlind F. Voigt T. Mspsim—an extensible simulator for msp430-equipped sensor boards. In: Proceedings of the European conference on wireless sensor networks (EWSN 2007). Delft, The Netherlands; 2007.

Eriksson J, Österlind F, Finne N, Tsiftes N, Dunkels A, Voigt T, et al. Cooja/mspsim: interoperability testing for wireless sensor networks. In: Proceedings of the second international conference on simulation tools and techniques (SIMUTools '09). Brussels, Belgium, Rome, Italy: ICST (Institute for Computer Sciences Social-Informatics and Telecommunications Engineering); 2009. p. 1–7.

Han G, Ma M. Connecting sensor networks with ip using a configurable tiny tcp/ip protocol stack. In: Sixth international conference on information, communications and signal processing. Singapore; 2007. p. 1–5.

Hui JW, Culler DE. Ip is dead long live ip for wireless sensor networks. In: Sixth ACM conference on embedded network sensor systems (SenSys '08). Rayleigh, NC, USA: ACM; 2008. p. 15–28.

Köpke A, Swigulski M, Wessel K, Willkomm D, Haneveld PTK, Parker TEV, et al. Simulating wireless and mobile networks in omnet++ the mixim vision. In: Proceedings of the first international conference on simulation tools and techniques for communications, networks and systems & workshops (SIMUTools 2008). Marseille, France; 2008.

Levis P, Madden S, Polastre J, Szewczyk R, Whitehouse K, Woo A, et al. Tinyos: an operating system for wireless sensor networks. Ambient intelligence. Springer-Verlag; 2004.

Montenegro G, Kushalnagar N, Hui J, Culler D. Transmission of ipv6 packets over ieee 802.15.4 networks. September 2007.

Neves P, Stachyra M, Rodrigues JJPC. Application of wireless sensor networks to healthcare promotion. Journal of communications software and systems (JCOMSS), Croatian Communications and Information Society, in cooperation with FESB, University of Split 2008;4(3):181–90.

Neves PACS, Esteves AFF, Cunha RMF, Rodrigues JJPC. User-centric data gathering multi-channel system for ipv6-enabled wireless sensor networks. International Journal of Sensor Networks (IJSNet)—Special Issue on Technologies, Recent Advances in Sensor Integration 2010a;8(3).

Neves PACS, Vaidya B, Rodrigues JJPC. User-centric plug-and-play functionality for ipv6-enabled wireless sensor networks. In: IEEE international conference on communications (ICC 2010). Cape Town, South Africa; 2010b.

Österlind F, Dunkels A, Eriksson J, Finne N, Voigt T. Cross-level sensor network simulation with cooja. In: The 31st IEEE conference on local computer networks (LCN 2006). Tampa, Fl, USA; 2006. p. 641–8.

Polastre J, Szewczyk R, Culler D. Telos: enabling ultra-low power wireless research. In: Proceedings of the fourth international symposium on information processing in sensor networks (IPSN 2005), Los Angeles, CA, USA; 2005. p. 364–369.

Ramanathan N, Chang K, Kapur R, Girod L, Kohler E, Estrin D. Sympathy for the sensor network debugger. In: Proceedings of the 3rd international conference on embedded networked sensor systems (Sensys 2005). San Diego, CA, USA; 2005. p. 255–67.

Rezgui A, Eltoweissy M. Service-oriented sensor–actuator networks: promises, challenges, and the road ahead. Computer Communications 2007;30(13): 2627–48.

Rodrigues JJPC, Neves P. A survey on ip-based wireless sensor networks solutions. International Journal of Communication Systems 2010;23(8):963–81.

Sensinode. Sensinode nodeview pro software, 2009. 〈http://www.sensinode.com/〉.

Shelby Z, Bormann C. 6LoWPAN: the wireless embedded internet. Wiley Series in communications networking & distributed systems. Wiley; 2009.

Silva JS, Ruivo R, Camilo T, Pereira G. Ip in wireless sensor networks—issues and lessons learnt. In: Third international conference on communication systems, software and middleware (COSMWARE 2008). Bangalore, India: IEEE Communication Society; 2008.

Singh D, Singh S, Singh M, Kew H-P, Jeoung D-U, Tiwary US, et al. Ip-based ubiquitous sensor network for in-home healthcare monitoring. In: International conference on multimedia, signal processing and communication technologies (IMPACT 2009). Aligarh, India; 2009. p. 201–4.

Stankovic JA. When sensor and actuator networks cover the world. ETRI Journal 2008;30(5):627–33.

Sung J, Kim Y, Kim T, Kim Y-J, Kim D. Internet metadata framework for plug and play wireless sensor networks. In: Sensors applications symposium (SAS 2009). New Orleans, LA, USA; 2009. p. 320–4.

Xia F, Tian Y-C, Li Y, Sun Y. Wireless sensor/actuator network design for mobile control applications. Sensors Journal 2007;(7):2157–73.

Yang S, Park S, Lee EJ, Ryu JH, Kim B-S, Kim HS. Dual addressing scheme in ipv6 over ieee 802.15.4 wireless sensor networks. ETRI Journal 2008;30(5):674–84.

Yu Y, Krishnamachari B, Prasanna VK. Energy-latency tradeoffs for data gathering in wireless sensor networks. In: Proceedings of the INFOCOM 2004. 23rd annual joint conference of the IEEE computer and communications societies. Hong Kong; 2004. p. 244–55.

Zhou L, Naixue X, Shu L, Vasilakos A, Yeo S-S. Context-aware middleware for multimedia services in heterogeneous networks. IEEE Intelligent Systems 2010a;99(March/April).

Zhou L, Wang X, Tu W, Muntean G-M, Geller B. Distributed scheduling scheme for video streaming over multi-channel multi-radio multi-hop wireless networks. IEEE Journal on Selected Areas in Communications 2010b;28(3):409–19.

Please cite this article as: Neves PA, et al. A multi-channel architecture for IPv6-enabled wireless sensor and actuator networks featuring PnP support. J Network Comput Appl (2011), doi:10.1016/j.jnca.2011.03.033