OPEN ACCESS

International Electronic Conference on Sensors and Applications
1 - 16 June 2014

www.mdpi.com/journal/sensors

Conference Proceedings Paper – Sensors and Applications

6LoWPAN – Towards Zero-Configuration For wireless Building Automation: System Architecture

Matthias Carlier [1,*], An Braeken [1], Bart Lemmens [2], Ruben Smeets [3], Laurent Segers [1], Kris Steenhaut [2], Abdellah Touhafi [1,2] and Nele Mentens [3]

[1] Department of Industrial Sciences and Technology (INDI), Vrije Universiteit Brussel, Pleinlaan 2, Elsene 1050, Belgium; E-Mails: an.braeken@vub.ac.be (A.B.); laurent.segers@vub.ac.be (L.S.); abdellah.touhafi@vub.ac.be (A.T.)

[2] Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel, Pleinlaan 2, Elsene 1050, Belgium: blemmens@etro.vub.ac.be (B.L.); ksteenha@etro.vub.ac.be (K.S.)

[3] Technologiecluster Elektrotechniek (ESAT), Campus Diepenbeek (@ KHLim), Wetenschapspark 21,3590 Diepenbeek: ruben.smeets@kuleuven.be (R.S.); nele.mentens@esat.kuleuven.be (N.M.)

* Author to whom correspondence should be addressed; E-Mail: matcarli@vub.ac.be (M.C.); Tel.: +3-226-292-976.

Published: 1 June 2014

Abstract: In this paper, we describe the techniques and technologies used in a home automation system based on 6LoWPAN. This IP-based protocol has the advantage that no extra layer or logic is required for communication with a node in or outside the network. The 6LoWPAN network is divided into three main parts: the central server, the border-routers and the embedded motes. The central server can access all the different nodes of all the different connected networks. Therefore, the server runs a database and hosts a web application, allowing to control and interact with the different resources in the network. As a consequence, we demonstrate how a network can be built with a minimum of configuration.

Keywords: 6LoWPAN; Sensor Networks; Contiki; Home Automation

1. Introduction

A lot of research has been done in the field of home automation. The ability to automate buildings or houses is endless with respect to monitoring, controlling, reporting, and alerting. The creation of a Java-based home automation system is described in [1]. The major disadvantage of this system is that the different devices have to be connected through a wire, which considerably reduces the ease of deployment into existing infrastructures. Others describe a system that uses wireless technologies

to reduce the amount of physical wiring required [2, 3]. They mostly suffer from other problems, like scalability or access delays. These systems also need some kind of translation from the used technology to existing technologies, when access to the network from the outside is desired.

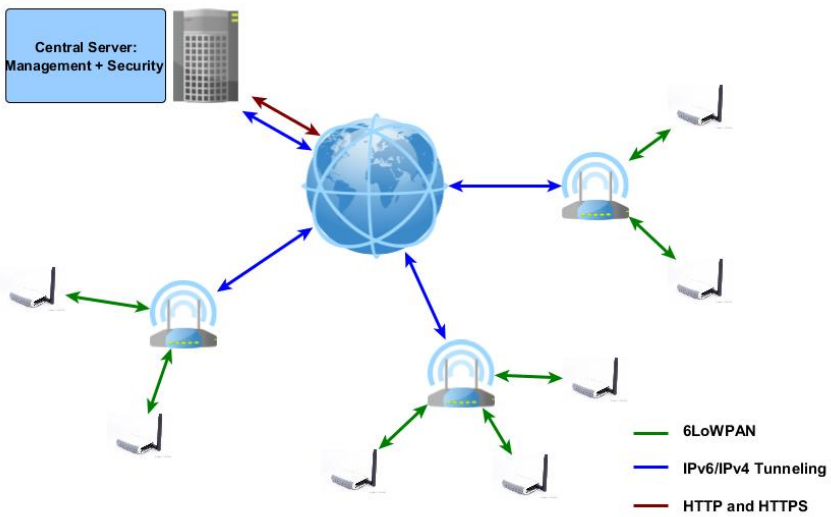
This is to a large extent due to the wide range of communication protocols. The 6LoWPAN protocol is a rather new and very promising protocol [4]. It is IP based, facilitating the integration into existing network. Each component of the system (i.e. plug, light, heating, etc.) has its own IP address, and can be interconnected with several devices in a wireless fashion. As a result, it becomes possible for instance to monitor the energy consumption at all times or to control devices at any place without having to switch between different protocols, but just by using the ubiquitous Internet Protocol. In [5], the authors of the article use this protocol to solve the previously mentioned problems, but don't take user friendliness into account.

In this paper, we describe the required techniques and technologies in order to build a wireless home automation system. This system is based on 6LoWPAN, which has great scalability, while minimizing access delays. We describe how we used different existing techniques, to build a wireless network that can be distributed over different locations with a minimum of configuration and installation cost. The paper is organized as follows: Section 2 deals with an overview of the architecture. In Section 3, the different components of the architecture are described. Finally, we end with some conclusions and future work.

2. System Architecture and Implementation

The setup of our system is divided into three main parts: the central server, the border-routers and the embedded motes (Figure 1). These parts will be thoroughly detailed in Section 3. First, we explain the implementation of the communication between the different parts of the network.

Figure 1. Overview of the system architecture



The network is divided into several different subnetworks, that can be at different geographical locations. Each subnetwork exists of at least one sensor node and exactly one BeagleBone, connected to a border-router. This border-router is the network's gateway to the rest of the Internet. It ensures that the nodes in a subnetwork can reach the rest of the network by forwarding packages to their correct destination. Each BeagleBone is connected to the central server, which will collect all information of the nodes in a relational database. This central server also hosts the web application, that allows to access the data, by means of a user friendly interface.

As one can see on figure 1, the communication between a sensor node and other nodes in the same network or the border-router uses a version of the IP-protocol, designed for low power devices, i.e. 6LoWPAN[6][7]. The 6LoWPAN protocol uses compressed IPv6 packets to transfer data. To get these packets to the central server, they need to be sent over the Internet, which in most current networks only supports IPv4. A small IPv6/IPv4 tunneling application ensures the tunneling between the Zolertia nodes, the border-routers and the central server. The application has two purposes. At one hand it provides the appropriate IPv6 addresses to the Beagle bones and, at the other hand it delivers unique IPv6 network prefixes for the 6LoWPAN networks. Both of these techniques are referred to the Dynamic Host Configuration Protocol (DHCP).

To make all the information stored on the server available to the users, a web page is created. This page can be used to monitor all the nodes and to control certain devices, connected to the nodes (i.e. lights, heating,etc.). The communication between the server and the web page will be handled the HTTP(S).

3. Components of the architecture

3.1. Sensor nodes

Each node is attached to different kinds of sensors and actuators, which need to be made available to the network. In order to provide the server with the information and controls of the nodes, each node is accessed through a RESTful interface, which offers great scalability[8]. It is very useful since in many cases, the system has no knowledge of which sensors and actuators are connected to a given node. We have chosen to use the Constrained Application Protocol (CoAP) as a RESTful interface. This protocol is an application layer protocol, based on HTTP, that allows certain devices to access several resources on a particular device[9]. Since CoAP is using UDP as a transport layer service, we also need to ensure that UDP-packets can be exchanged.

One of the main goals of the project is to allow users to deploy the sensor nodes with a minimum of configuration. This means that a node should be able to enter the network automatically and inform the server with information like its address or which resources these nodes will make available. This process is performed by sending a simple UDP packet to the server, which has a fixed IP address. The UDP packet contains all the information the server needs to add to its database, so it can start monitoring all the available resources. The data of the different UDP packets is transferred using the 6LoWPAN-protocol, as explained before. This protocol uses RPL as its routing protocol, which is a protocol designed for low power and lossy networks. This protocol allows data, in our case UDP packets, to be transferred over multiple hops, to its destination or default gateway.

The current state of all of the resources of a node can be checked by sending a CoAP GET request to the node. For some resources, the node can be asked to send a notification, each time the state of a resource changes, by sending an OBSERVE request to the node. The state of certain resources, like resources that represent lamps, can be changed by sending a POST request to the node.

As platform for the sensor nodes, we work with the Zolertia motes. These motes are equipped with an MSP430F2617 micro-controller from Texas Instruments, which runs the Contiki embedded operating system. Contiki is an open source operating system for the Internet of Things[10][11]. It can be used in combination with several different types of nodes. Another advantage of this operating system is that it already includes a lot of protocols for all the different layers, like UDP and RPL. Erbium is an implementation of CoAP for the Contiki operating system that uses the Contiki network stack to provide power efficiency. The tunneling has also been adapted to encapsulate the existing tunslip6 application of Contiki. In that way, the server is able to directly communicate with each of the Zolertia nodes (and vice versa).

Although the core of the Contiki-OS has fully been ported to the MSP430F2617 micro-controller, some drivers were not available yet. Two drivers were implemented during the project: PWM driver

for LED-light dimming (over I2C) and a generalized method for remote button sensing. With the latter one, it is possible to interrupt the system with the desired interruptible IO-pins. Moreover, in the scope of the project, more different resources are required then the ones available on the Zolertia, like sensors that measure the light intensity in a room or relays that control an actual external light. Therefore two new sensors and one new actuator were developed/purchased and tested: an external LED, a LUX sensor and a PIR-based motion sensor. For all these sensors, a driver has been written. As a consequence, they can be used in combination with the CoAP server on the nodes.

Finally, we want to note an important problem we had to overcome. The MSP430F2617 has a total amount of 92kB of addressable memory space. The older version of the open source MSP430 compiler (4.6.3) lacks of support for applications, which exceed the 64kB memory boundary. This compiler compiles only in 16-bit addressing mode, which limits the programmer to only 64kB of memory. The amount of memory, needed for the CoAP server is around 58kB. Together with the security aspects and automatic node registration, the total amount of memory space needed exceeds the 64kB boundary. A newer version (4.7.0) of the MSP430 compiler supports 20-bit addressing. Special care must however be taken in order to fully utilize the 20-bit capabilities. Aside of the compiling issues, another problem arose when trying to program the MSP430F2617 with the regular USB Boot Script Loader (BSL). This bootloader is limited to only 64kB of machine-code. To overcome this, the MSP-FET430UIF JTAG programmer is necessary to access the complete memory space. For ease of use a JTAG-programmer-adapter system has been designed[12].

3.2. Border-Router & BeagleBone

When a node sends a packet to a destination outside the network (i.e. the central server), which is only accessible through the Internet, the RPL-protocol can't directly reach that destination. Then the RPL-protocol will make sure the packet gets to the border-router. Whenever the border-router receives a packet, with a destination outside the RPL-network, it will transfer the packet to the BeagleBone, which will send the packet to the server over the Internet, using a tunneling program. This means the border-router is actually the default gateway, where nodes send their packages with a destination they can’t directly reach. The transmission of packets between the border-router and the BeagleBone is handled by a protocol called SLIP, which allows the transmission of packets over a serial line, like USB[13].

This means the border-router must be able to decide if a packet should be send to one of its neighboring nodes, or to the BeagleBone using SLIP. This decision is done by consulting the routing tables. These tables contain information about where to send packets destined for all reachable nodes.

The work demand of the BeagleBone is much less complicated. It just has to forward a packet from one interface to the other. The BeagleBone is connected to the Internet via an Ethernet connection and to the sensor network through a USB connection with the border-router. The BeagleBone will just send all packets it receives on its Ethernet connection to the border-router over SLIP. All packets coming from the border-router, will be processed by the tunneling program, which will send it to the correct destination on the Internet.

3.3. Central Server

The central server runs a CoAP client, which is able to access the data from all the different nodes in all the different subnetworks, after sharing its information with the server. The server is able to get the current state of a known resource on command. The server is also able to observe certain resources that provide this service. This means that a node will notify the server when the state of a resource changes. The information received from the nodes will be used to update a database, also running on the server.

The database on the central server holds the necessary information about the complete network. The database is subdivided into 5 major parts: users/roles, rules, embedded nodes, security and locations. Users may access the system (with roles), and decide which actions to take on a given area (nodes, location and rules). User roles are used to deny certain users to take certain actions. In the “nodes” part of the database, all the information about the nodes is stored, like the available resources or a link to the location, associated with this node. The information about the different locations is stored in the “location” part. The “rules” part is used to connect resources to each other, like a motion sensor, that is connected to the lighting in the same room.

The central server is also able to forward all incoming packets to their correct destinations in the different subnetworks. This is done by means of the tunneling program, which sets up virtual connections to these networks.

The central server hosts a web application to interact with these resources. The required functionalities include network administration, visualization of the state of the network's sensors and actuators as well as interaction, and implementation of automation rules. A dedicated server daemon is used to handle all CoAP communication. The frontend of the web application is implemented with the Google Web Toolkit for cross-browser compatibility. In order to present a consistent view when multiple users are utilizing the web application simultaneously, the server-side pushing techniques of the Atmosphere framework are used.

4. Conclusions and future work

We now have a system containing a network of Zolertia Z1 nodes, which are able to send messages to each other. These nodes form a network, which allows messages to be sent over multiple hops before reaching their destination. When a node is turned on, no additional configuration is needed for the node to enter the network, thanks to the use of the 6LoWPAN protocol. Each network is equipped with a border-router. This is a node with a special purpose. These nodes do not have a CoAP server, but just act as a gateway between the wireless sensor network and the rest of the Internet. This, in combination with a tunneling program, allows devices from outside the sensor network to access the different nodes. This functionality allows to connect sensor networks with other (sensor) networks. The central server can access all the different nodes on all the different connected networks. On this server, a database is maintained that contains information about all the different nodes and their resources.

We want to stress that this work is part of a larger project in which also the required security aspects and implementations are coupled with the above-described architecture. In a further phase, we plan a detailed performance analysis of the system with respect to scalability, energy consumption range and reaction speed.

Acknowledgments

This research has been made possible thanks to a Tetra grant of the Flanders agency for Innovation by Science and Technology. The authors would also like to acknowledge the contribution of the COST Action IC1303 - AAPELE, Architectures, Algorithms and Platforms for Enhanced Living Environments.

Author Contributions

The research described in this paper has been performed by Matthias Carlier, Laurent Segers, Ruben Smeets and Bart Lemmens. The project is followed up by Nele Mentens, Kris Steenhaut, Abdellah Touhafi, and An Braeken, where the latter is project leader.

Conflicts of Interest

The authors declare no conflict of interest.

References and Notes

1. 1. Al-Ali, A. R.; Al-Rousan, M. "Java-based home automation system" *IEEE Transactions on Consumer Electronics, vol. 50*, 2004, 2, pp. 498–504.
2. Sriskanthan, N.; Tan, F.; Karande, A. "Bluetooth based home automation system" *Microprocessors and Microsystems, vol. 26*, 2002, 6, pp. 281–289.
3. Gill, K.; Yang, S.-H.; Yao, F.; Lu, X. "A ZigBee-based home automation system" *IEEE Transactions on Consumer Electronics, vol. 55,* 2009, 2, pp. 422–430.
4. Toscano, E.; Lo Bello, L. "Comparative assessments of IEEE 802.15.4/ZigBee and 6LoWPAN for low-power industrial WSNs in realistic scenarios", *9th IEEE International Workshop on Factory Communication Systems (WFCS)*, Lemgo, Germany, 21-24 May 2012, pp. 115-124.
5. Efendi, A. M.; Oh, S.; Negara, A. F. P.; Choi, D. "Battery-Less 6LoWPAN-Based Wireless Home Automation by Use of Energy Harvesting" *International Journal of Distributed Sensor Networks*, 2013.
6. Shelby, Z.; Bormann, C. *6LoWPAN: the Wireless Embedded Internet*, 1st ed.; John Wiley & Sons Ltd: Chichester, United Kingdom, 2009.
7. Afanasyev, M.; O'Rourke, D.; Kusy, B.; Hu, W. "Heterogeneous Traffic Performance Comparison for 6LoWPAN Enabled Low-Power Transceivers", *HotEMNETS 2010 Workshop on Hot Topics in Embedded Networked Sensors*, Killarney, Ireland, 28-29 June 2010, pp. 10-20.
8. Fielding, R. "Principled Design of the Modern Web Architecture" *ACM Transactions on Internet Technology, Volume 2 Issue 2,* 2002, pp. 115-150.
9. Kuladinithi, K.; Bergmann, O.; Pötsch, T.; Becker, M.; Görg, C. "Implementation of CoAP and its Application in Transport Logistics", In *Proceedings of the Workshop on Extending the Internet to Low power and Lossy Networks*, Chicago, USA, April 2011.
10. Dunkels, A.; Gronvall, B. ; Voigt, T. "Contiki - a lightweight and flexible operating system for tiny networked sensors", *29th Annual IEEE International Conference on Local Computer Networks*, Tampa, USA, 16-18 November 2004. pp. 455-462.
11. Dunkels, A.; "Contiki: Bringing IP to Sensor Networks", *Ercim news*, 2009, Vol. 76, pp. 59-60.
12. Segers, L.; Smeets, R.; Carlier, M.; Braeken, A.; Touhafi, A.; Steenhaut, K.; Mentens, N.; "Pogo-Pin-JTAG-Programmer-Box: A Low-Cost JTAG Programmer Interface for the Wireless Embedded Zolertia-Z1 Platform", ECSA, 1-16 June 2014.
13. Miller, P.; "TCP/IP: The Ultimate Protocol Guide, Volume 2", Brown Walker Press: Boca Raton, Florida, USA, 2009.

© 2014 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/).