Towards fully IP-enabled IEEE 802.15.4 LR-WPANs

Francesca Lo Piccolo *, Donato Battaglino *, Lorenzo Bracciale*,
Andrea Bragagnini†, Maura Santina Turolla† Nicola Blefari Melazzi *
*DIE, Universitá di Roma "Tor Vergata", Rome, Italy
{francesca.lopiccolo,donato.battaglino,lorenzo.bracciale,blefari}@uniroma2.it
†Telecom Italia, Turin, Italy
{andrea.bragagnini,maurasantina.turolla}@telecomitalia.it

***Abstract*—This extended abstract describes a poster and a proposal for demonstration. Its focus is on low rate wireless personal area networks (LR-WPANs) which use IEEE 802.15.4 physical and MAC layers at the lower layers of the protocol stack. More precisely, goal of this extended abstract is to present a real test-bed, whose main goal is to demonstrate how IEEE 802.15.4 LR-WPANs can fully inter-operate with IP networks. To this end, the protocol stack implemented at each IEEE 802.15.4 node includes the so said LoWPAN adaptation layer, which has been defined by the IETF 6LoWPAN working group. Moreover, we will present a possible practical solution for the problem of how to realize a gateway between the IEEE 802.15.4 LR-WPANs and the IP networks[1].**

I. Introduction

Low rate wireless personal area networks (LR-WPANs) are attracting an ever increasing attention for their capability of supporting low-rate and low-power short range wireless communications and, as a consequence, applications with simple wireless connectivity, relaxed throughput and relaxed latency requirements in fields such as home automation, industrial control, personal medical assistance and remote control and monitoring.

One of the most important and currently available reference standards for LR-WPANs is IEEE 802.15.4 [1][2], which defines the physical layer and the medium access control (MAC) sub-layer. In what follows, the LR-WPANs using IEEE 802.15.4 physical and MAC layers will be denoted as IEEE 802.15.4 LR-WPANs.

The first attempt of defining the upper part of the protocol stack for IEEE 802.15.4 LR-WPANs is due to the ZigBee Alliance association, which has defined the network layer and the application framework in the so said ZigBee Alliance specification [3]. Unfortunately, the network layer of the ZigBee protocol stack is not compatible with the IP network layer. This makes impossible the direct interoperability between ZigBee networks and IP networks, and limits the evolution of ZigBee networks towards the emerging future "Internet of Things". In fact, the futuristic vision of an Internet of Things [4] introduces the concept of a network where all the things are able to communicate with each other across the whole world. In such a scenario, the IP protocol with its end-to-end inspiring principle appears the most natural and obvious candidate to building a global network out of things. According to the Internet of Things vision, the IETF 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) working group has recently proposed a solution, described in RFC 4944 [5], for enabling the transmission/reception of IPv6 packets over LR-WPANs based on IEEE 802.15.4 standard.

However, the 6LoWPAN solution basically consists only in a sort of adaptation layer, the so said LoWPAN adaptation layer, to be included between the IEEE 802.15.4 MAC sub-layer and the IPv6 layer, in order to make IEEE 802.15.4 layers interact with IPv6 layer and vice versa. Thus, goal of this extended abstract is to describe a real test-bed which demonstrates and implements the 6LoWPAN functionalities in a complete protocol stack for fully IP-enabled IEEE 802.15.4 LR-WPANs. In addition, we will present a possible solution for the problem of how to realize a gateway between the IEEE 802.15.4 LR-WPANs and the IP networks.

The extended abstract is organized as follows. We present our implemented test-bed in section II. Subsection II-A describes the protocol stack implemented at each IEEE 802.15.4 6LoWPAN node, whereas subsection II-B focuses on the 6LoWPAN/IPv6 gateway that enables the transmission of 6LoWPAN packets towards the Public Internet and vice versa. Finally, we conclude the extended abstract in section III.

II. Description of the implemented test-bed

This section presents the implemented test-bed, whose main goal is demonstrating the 6LoWPAN functionalities.

We assumed for the test-bed the general reference scenario depicted in figure 1. Each node in the IEEE 802.15.4 LR-WPAN implements the LoWPAN adaptation layer and, thanks to this, it is able to transmit/receive IPv6 packets to/from the Public Internet. Figure 1 shows also that such kind of interaction between IEEE 802.15.4 6LoWPAN networks and the Public Internet calls for a special gateway, which is composed by an IEEE 802.15.4 6LoWPAN device connected to a common Linux box through a serial line. In such a way, the gateway is able to communicate with both IEEE 802.15.4 6LoWPAN devices and hosts in the Public Internet.

[1]This work has been carried out in cooperation with and with funding from Telecom Italia.

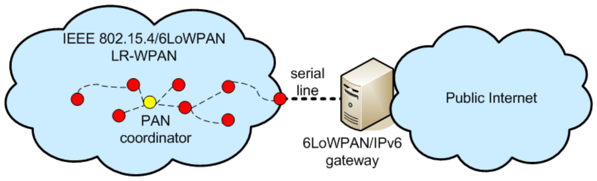

Fig. 1. Reference scenario for the implemented test-bed.

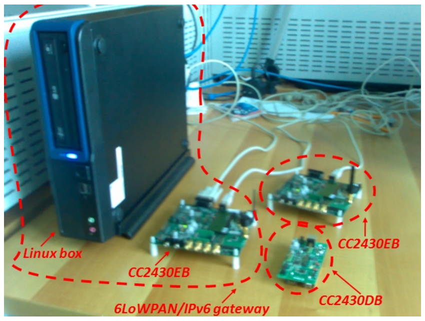

Fig. 2. Test-bed hardware components.

As regards the test-bed hardware, we used the Texas Instruments' CC2430ZDK [6] development kit. This is based on the CC2430 System-on-Chip, which basically includes the 2.4 GHz IEEE 802.15.4 transceiver, an 8051 controller and the flash memory. The kit includes three simple minimal board (CC2430DB) and two evaluation boards (CC2430EB) equipped with joystick, several buttons, USB port, RS-232 port and LCD display. The RS-232 port has been used to connect an evaluation board to the Linux box and to implement the 6LoWPAN/IPv6 gateway. We used, instead, common off-the-shelf hardware for the Linux box. The test-bed hardware components are shown in figure 2.

As nodes in the IEEE 802.15.4 6LoWPAN LR-WPAN and gateway play different roles in the test-bed, in what follows we provide separate descriptions for these components.

A. IEEE 802.15.4 6LoWPAN nodes

Figure 3 depicts the protocol stack implemented in each IEEE 802.15.4 6LoWPAN node. Each node implements the IEEE 802.15.4 MAC protocol at the lowest layer. For this purpose, we used the so called TIMAC, that is the implementation of the MAC protocol provided by Texas Instruments.

As regards the network layer, we implemented from scratch all the 6LoWPAN functionalities. According to 6LoWPAN

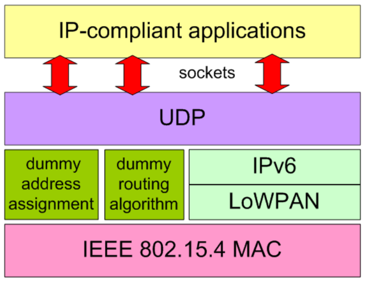

Fig. 3. Protocol stack implemented in each IEEE 802.15.4 6LoWPAN node.

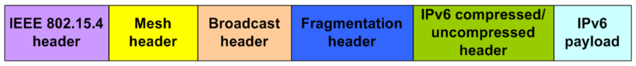

Fig. 4. IEEE 802.15.4 data frame including 6loWPAN headers and IPv6 payload.

specification [5], we implemented i) IPv6 Stateless Address Auto-configuration [7] for IPv6 address assignment, and ii) the LoWPAN adaptation layer between the IEEE 802.15.4 MAC sub-layer and the IPv6 layer. More precisely, each node generates an IPv6 link-local or global unicast address, which is formed by placing the prefix FE80::/64 or a global link prefix before the so-called Interface Identifier. Interface Identifiers can be derived from either the 64-bit MAC address which is assigned to each device by the manufacturer (the so said IEEE EUI-64 identifier) or the 16-bit short address, which the PAN coordinator may assign after the association. As regards the LoWPAN adaptation layer, this may be regarded as a sequence of "header type + header fields" blocks, which, in conjunction with the IPv6 payload, are encapsulated within the payload of an IEEE 802.15.4 data frame. We implemented all the header types defined in [5]: mesh header, broadcast header, fragmentation header, compressed or not compressed IPv6 header. So, if all the headers are included, transmitted/received packets assume the format depicted in figure 4.

We also implemented a dummy multi-hop routing algorithm. According to such algorithm, if a node does not know how to reach the final destination, it forwards the packet to the PAN coordinator. this is the reason why the test-bed currently supports only star topologies with every node directly connected with the PAN coordinator. In addition, the protocol stack in figure 3 includes a module for the 16-bit layer 2 short address assignment. In fact, the specification of how the PAN coordinator assigns 16-bit short addresses is out of the scope of the IEEE 802.15.4 standard, which leaves the responsibility of such assignment to upper layer protocols. However, the 6LoWPAN IETF working group admits the possibility that IPv6 layer-3 addresses for IEEE 802.15.4 nodes are derived from 16-bit layer 2 short addresses, without specifying how 16-bit layer 2 short addresses are generated and assigned. As a consequence, since 16-bit short addresses are highly desirable for the sake of compression, especially for communications within an IEEE 802.15.4 LR-WPAN, a mechanism for 16 bit-address auto-configuration and assignment has to be introduced, to operate in conjunction with 6LoWPAN adaptation layer. This mechanism is referred to as dummy in figure 3, because each node has a direct connection with the PAN coordinator, and this one sequentially generates a 16-bit address for each node joining the PAN.

According to [5], where the authors mention only UDP as possible transport protocol and suggest a possible compression scheme for UDP headers, we implemented only UDP at transport layer. As regards more reliable transport protocols, like TCP, we can identify two main factors that could prevent their use: i) limited flash ROM memory of a device and ii) limited RAM memory. The first one could impede the implementation of a protocol stack including TCP as transport protocol. In fact, the flash memory currently available in the most of commercial IEEE 802.15.4 devices, as well as in the CC2430ZDK development kit used for the test-bed, is at most 128 KB. The code size of both TIMAC and our implemented 6LoWPAN functionalities is about 60 KB. The implementation of TCP in μC/TCP-IP [8], that is one of the currently available minimal TCP/IP stacks, requires an amount of memory ranging from 33 KB to 43 KB. Thus, only an amount of memory ranging from 20 to 40 KB would remain available for applications. The second one could impede the full exploitation of the TCP functionalities. For instance, let us assume that i) the available RAM memory is 8 KB and it reduces to only 4 KB in energy saving mode and ii) 1 KB out of 8/4 KB is devoted to each one of the sending and receiving buffers. This seems a reasonable choice, since it is equivalent to occupy half of the whole RAM memory available in energy saving mode. By subtracting the TCP and LoWPAN headers from the maximum physical data unit size of 128 bytes, we can deduce that 93 bytes is the maximum TCP segment size (MSS). This means that at most 10 not-acknowledged segments can be transmitted in case of one single connection and at most 5 not acknowledged segments can be transmitted in case of two connections. As a consequence, it is likely that i) sending nodes are frequently stopped due to TCP flow control regardless of network congestion, ii) congestion control functionality of TCP is not fully exploited, iii) the available transmission capacities are not fully used.

B. *6LoWPAN/IPv6 gateway*

The 6LoWPAN/IPv6 gateway enables the transmission of 6LoWPAN packets towards the Public Internet and vice versa.

The IEEE 802.15.4 device connected to the Linux box through the serial line is responsible for the interaction between the gateway and the IEEE 802.15.4 6LoWPAN LR-WPAN. With respect to the other IEEE 802.15.4 devices in the LR-WPAN, the IEEE 802.15.4 device in the gateway implements a module for the translation of the 6LoWPAN headers and possible UDP compressed headers into standard IPv6 and UDP headers and vice versa. Such translation does not regard the destination addresses. This means that the 6LoWPAN/IPv6 gateway does not perform NAT-typical operations.

The Linux box is instead responsible for the interaction between the gateway and the Public Internet. More precisely, the Linux box may be regarded as endpoint of an *IPv6-in-IPv4 tunnel* to the SixXS tunnel broker [9]. With regard to the tunneling protocol, we chose a non-standard solution based on *AYIYA* [10] in place of the standard proto-41-like solutions [11], in order to allow the gateway to be placed behind NATs. In such a way, under the assumption that AYIYA data are conveyed by UDP, the resulting tunnel is IPv6-over-AYIYA-over-UDP-over-IPv4. We also used the Unix implementation of *AICCU* (Automatic IPv6 Connectivity Client Utility) [12] for the tunnel configuration. More precisely, AICCU sets up a virtual network interface, which is assigned with a global IPv6 unicast address and uses the other endpoint of the AYIYA tunnel, i.e. the tunnel broker endpoint, as default gateway.

As regards the interaction between the IEEE 802.15.4 device and the Linux box, we chose the *SLIP* (Serial in LIne Protocol) [13] protocol to carry data over the serial line. We preferred SLIP to other standard and custom protocols to transmit IP packets over a serial line, even if SLIP supports the only transmission of IPv4 datagrams. The reason is that the very limited size of the flash memory in CC2430 devices (128 KB) calls for a simple protocol that can be implemented with small code size. This is just the case of SLIP, which limits itself to define and insert two special characters, the termination character END (0xC0) and the escape character ESC (0xDB). In more detail, we used the Unix utility *slattach* in order to create a point-to-point network interface able to transmit IPv4 packets over SLIP through the serial line. Then a point-to-point tunnel was created to connect the network interface created by means of slattach and a new virtual interface assigned with an IPv6 address in the same IPv6 subnet as the IEEE 802.15.4 6LoWPAN LR-WPAN. In such a way, this interface is able both to capture and intercept all the traffic from the Public Internet towards the LR-WPAN and simultaneously to use the serial line and SLIP to transmit that traffic towards the IEEE 802.15.4 part of the gateway.

III. Conclusions and future work

In this extended abstract we presented a test-bed, whose main goal is demonstrating the 6LoWPAN capability of making IEEE 802.15.4 LR-WPANs fully IP-enabled. Whereas the 6LoWPAN functionalities has been fully and completely developed, we developed only a simple implementation of some modules, like the one for the multi-hop routing and 16-bit short address configuration and assignment. Thus, our future work will consist in enriching the test-bed with not-trivial multi-hop routing and address auto-configuration capabilities. Other functionalities such as the management of security infrastructure and the interoperability with ZigBee networks could be also taken into account and developed as future work.

References

[1] IEEE Standard 802.15.4-2003, *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 2003

[2] IEEE Standard 802.15.4-2006, *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 2006

[3] ZigBee Alliance, *ZigBee Specification*, October 2007

[4] N. Gershenfeld, R. Krikorian, D. Cohen, *The Internet of Things*, Scientific American, October 2004

[5] RFC 4944, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*

[6] CC2430 web site, http://focus.ti.com/docs/prod/folders/print/cc2430.html

[7] RFC 4862, *IPv6 Stateless Address Autoconfiguration*

[8] μC/TCP-IP reference web site, http://www.micrium.com/products/tcp-ip/tcp-ip.htm

[9] SixXS web site, http://www.sixxs.net/

[10] J. Massar, *AYIYA: Anything In Anything* , expired Internet Draft, on line available at http://unfix.org/~jeroen/archive/drafts/draft-massar-v6ops-ayiya-02.txt

[11] RFC 1933, *Transition Mechanisms for IPv6 Hosts and Routers*, 1996

[12] AICCU reference site, http://www.sixxs.net/tools/aiccu/

[13] RFC 1055, *A Nonstandard for transmission of IP datagrams over serial lines: SLIP*, 1988

IV. Space requirements

The extended abstract describes a test-bed, which we propose for both poster presentation and demonstration. Poster presentation requires simply a panel. Demonstration requires a table, power availability and public Internet connectivity.