On the IP support in IEEE 802.15.4 LR-WPANs: self-configuring solutions for real application scenarios

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}@telecomitalia.it

***Abstract*—Due to the lack of IP support, the ZigBee protocol stack for IEEE 802.15.4 low rate wireless personal area networks (LR-WPANs) is the perfect solution for closed, ad-hoc environments. However, we can envisage also open and interconnected application scenarios, such as the Internet of Things or the integration of ZigBee/IEEE 802.15.4 nodes in mobile/smart phones, that would greatly benefit from the IP support.**

The LoWPAN adaptation layer proposed by the 6LoWPAN IETF working group to enable the transmission/reception of IPv6 data in IEEE 802.15.4 networks may be regarded as the first step towards the direction of IP direct support into IEEE 802.15.4 nodes. However, a fully fledged solution to be used in the scenarios outlined above still lacks the following functionality: i) self-configuring address assignment mechanism; ii) multi-hop routing protocol; iii) support of Internet-compliant transport protocols; iv) self-configuring network and service discovery. It may seem trivial to provide such functionality in an IP world, as off the shelf solutions could be adopted; however, this is not an easy task, given the very limited hardware capabilities of IEEE 802.15.4 devices.

In this paper we define a complete protocol architecture and specific self-configuring mechanisms to support the aforementioned functionality over IEEE 802.15.4 devices, taking into account their hardware constraints.

We also present an implemented test-bed which demonstrates the functionality of the proposed solution in a real application scenario. [1].

I. Introduction

Low rate wireless personal area networks (LR-WPANs) are the ideal candidate for enabling applications with simple wireless connectivity, relaxed throughput and relaxed latency requirements in fields such as sensor network applications among which home automation, industrial control, personal medical assistance and remote control and monitoring.

The most important and currently available reference standard 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 the IEEE 802.15.4 physical and MAC layers will be denoted as IEEE 802.15.4 LR-WPANs.

IEEE 802.15.4 physical and MAC layers are also used in conjunction with the network layer and the application framework defined by the ZigBee Alliance association in the so said ZigBee protocol stack [3].

Unfortunately, the network layer of the ZigBee protocol stack is not interoperable with the IP network layer. On the one hand, this makes the ZigBee protocol stack the perfect solution for closed and not interconnected ad hoc environments. On the other hand, the lack of IP support in ZigBee WPANs is a serious drawback for open and connected environments, which would clearly benefit from built-in inter-networking functionalities. As a matter of fact, we can imagine several application scenarios, going beyond closed automation, control and monitoring networks, which would take great advantage from an IP support, such as:

- integration of ZigBee devices in mobile/smart phones: Telecom Italia, the major Italian operator and member of ZigBee Alliance, envisages in [4] a scenario comprising mobile phones equipped with either the so-called ZSIMs, which are SIMs integrating a ZigBee node, or the so called ZSDs, which are micro or mini SD cards integrating a ZigBee chip-set. In such a scenario, IP-enabled ZigBee nodes would be very convenient to complement GSM/UMTS with a near field technology for data exchange (chat and advertisements in a commercial center, configuration data, micro-payments, access control, etc.).
- pervasive networking: in an Internet of Things [5] ZigBee could be a very convenient technology to support sensor networks and in general to support communication among networked devices; however, since the IP protocol is the most natural and obvious candidate for building an Internet of Things, the wireless technology of choice should include easy internetworking function with IP networks.

In the literature we can find several solutions to interconnect ZigBee WPANs to IP networks (see II for a description of

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

such solutions). Most of these proposals exploit a complex, stateful gateway between IEEE 802.15.4 LR-WPANs nodes and IP networks; the gateway establishes and handles the communication between the Internet and the IEEE 802.15.4 WPANs and manages all the necessary validity checks and message format translations. In addition, to allow ZigBee nodes to communicate with IP hosts in the Internet, it is necessary to either add a suitable application layer protocol to ZigBee nodes or to assign a further suitable identifier/address to both ZigBee nodes and IP nodes implementing an ad-hoc abstraction layer. These approaches could be viable in a limited set of scenarios where flexibility and openness are not important, but they are less appealing for more general and future-proof applications.

Obviously, if we want to interconnect two different complete protocol stacks, there is not an easy way out. A more radical approach is to replace the network and upper layers of ZigBee with a TCP/IP stack, keeping only the physical and MAC layer of IEEE 802.15.4.

In this way, all WPANs nodes would naturally support IP, and data transfer to/from the Internet would become trivial.

However, some limitations of IEEE 802.15.4 do not allow to simply put the classical TCP/IP stack directly over the MAC layer, but require a suitable adaptation layer. These limitations include a too small data transfer unit (of 128 bytes), and the limited memory and processing capabilities of current IEEE 802.15.4 devices.

The IETF 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) working group has recently proposed a solution to overcome such limitations by defining an adaptation layer ([6]), the so-called LoWPAN adaptation layer, which enables the transmission/reception of IPv6 packets over LR-WPANs based on the IEEE 802.15.4 standard, and is located between the IEEE 802.15.4 MAC layer and the IPv6 layer.

The LoWPAN adaptation layer solves the issue of the small size of the IEEE 802.15.4 data transfer unit by offering compression and fragmentation/de-fragmentation capabilities. However, a fully fledged solution to be used in the scenarios outlined above still lacks the following functionality: i) self-configuring address assignment mechanism; ii) multi-hop routing protocol; iii) support of Internet-compliant transport protocols; iv) self-configuring network and service discovery. It may seem trivial to provide such functionality in an IP world, as off the shelf solutions could be adopted; however, this is not an easy task, given the very limited hardware capabilities of IEEE 802.15.4 devices.

In this setting, the contribution of this paper is twofold: i) to define a complete protocol stack integrating a specific self-configuring mechanisms to support the aforementioned functionality over IEEE 802.15.4 devices, taking into account their hardware constraints; ii) to present a test-bed, which we implemented to demonstrate the functionality of our proposed solution and of selected applications.

The paper is organized as follows. Section II discusses the related work. Section III provides the necessary background information on 6LoWPAN specification. Section IV presents the hardware constraints of typical IEEE 802.15.4 devices. Section V identifies solutions for routing, address assignment, transport, network and service discovery that are suitable for the hardware limitations of typical 802.15.4 devices. Section VI presents a possible protocol architecture. Such architecture has been implemented in a real test-bed which is described in section VII. Finally, we conclude the paper in section VIII.

II. Related work

TinyREST [7] uses an HTTP-based approach. It assigns a URL address to all resources, and it uses HTTP methods and extensions to request data or send command to the IEEE 802.15.4 WPANs. However, HTTP messages are not encapsulated within TCP packets, and they are transmitted via the TinyOS network stack. Since this stack is not IP-compliant, a HTTP-2-TinyREST gateway is necessary for establishing and handling the communication to/from the Internet and the IEEE 802.15.4 WPANs, including all the necessary validity checks and message format mappings.

TinySIP [8] uses the SIP protocol and a subset of SIP methods to enable the communication between IEEE 802.15.4 WPANs and the Internet. As in TinyREST, a sort of communication abstraction layer is provided in such a way that TinySIP layer can be used upon an IP-not-compliant already existing protocol stack. This implies that a TinySIP gateway, mapping SIP methods to TinySIP methods and disseminating the messages among the IEEE 802.15.4 nodes, plays a fundamental role.

Sakane et al. [9] focus on the interconnection between IEEE 802.15.4 WPANs and the public IPv6 Internet too. They present i) a translation method which makes use of a common addressing layer by defining a device identifier called *devid* to identify 802.15.4 nodes and IPv6 nodes in a similar way, and ii) an interconnection node called *translator* to interconnect IEEE 802.15.4 WPANs and the public IPv6 Internet. In more detail, the translator achieves virtual IEEE 802.15.4 addresses for IPv6 nodes and keeps trace of the correspondence between the virtual IEEE 802.15.4 address, IPv6 address and *devid*. When an IEEE 802.15.4 node has to communicate with an IPv6 public node, it sends data to the virtual IEEE 802.15.4 address of the IPv6 node. In such a way, the translator can intercept data, translate properly the format and forward data to the IPv6 node. Viceversa, when an IPv6 public node wants to communicate with an IEEE 802.15.4 node, it sends data directly to the translator by indicating explicitly the destination *devid*, so that the translator forwards data to the actual IEEE 802.15.4 destination node.

A similar internetworking mechanism to interconnect ZigBee/802.15.4 WPAN and IPv6 networks has been proposed in [10]. Specifically, every ZigBee node is assigned with a Global Unicast IPv6 address, and each IPv6 node is also assigned with a ZigBee short address. In addition, IPv6 nodes are organized in Multicast Groups to enable the reception of broadcast messages from ZigBee networks. The key component of the system is a stateful gateway which i) assigns ZigBee nodes and IPv6 nodes with the corresponding IPv6/ZigBee addresses,

ii) stores the pairs (ZigBee address, IPv6 address), iii) keeps trace of all the on going communications and executes all the necessary format translation operations.

Other solutions are instead based on the LoWPAN adaptation layer defined by the 6LoWPAN IETF working group in RFC 4944 [6]. For instance, the protocol stacks blip [11] or uIPv6 [12] implement the LoWPAN adaptation layer. The first one includes also the header compression scheme suggested in the RFC 4944, IPv6 neighbour discovery, default route selection, point-to-point routing, UDP and a prototype of TCP. The second one includes Duplicate Address Detection (DAD) to avoid collisions between auto-assigned addresses, routing functionalities, TCP and UDP.

It is worth mentioning also the activity of ROLL (Routing Over Low power and Lossy networks) IETF working group, which presents in [13] the routing requirements which should be taken in account in the design of an ad-hoc routing protocol for Low Power and Lossy Networks (LLNs). These are typically composed of many embedded devices with limited power, memory, and processing resources and are interconnected by a variety of links, such as IEEE 802.15.4, Bluetooth, Low Power Wi-Fi. As a consequence, there is a very high number of similarities between LLNs and IEEE 802.15.4 networks, and 6LoWPAN and ROLL working groups are working together to enable a Wireless Embedded Internet composed of IEEE 802.15.4 devices. The problem of how to make applicative protocols (such as HTTP) inter-operate with lightweight underlying protocols, exploiting LoWPAN adaptation layer and designed for low-rate and lossy networks, is formulated in the Internet Draft [14].

III. Background information on 6LoWPAN specification

This section provides the reader with the necessary background information on 6LoWPAN specification [6]. This proposes i) IPv6 Stateless Address Auto-configuration [15] for IPv6 address assignment, and ii) the LoWPAN adaptation layer between the IEEE 802.15.4 MAC sub-layer and the IPv6 layer.

According to the IPv6 Stateless Address Auto-configuration, first nodes generate an IPv6 link-local unicast address by placing the prefix FE80::/64 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 IEEE 802.15.4 PAN coordinator[2] may assign after the association (see [2] for more details). Then, if nodes are informed by the coordinator about the presence of a global link prefix, they generate a global IPv6 unicast address by appending their interface identifier to the 64 bits of the global link prefix.

The LoWPAN adaptation layer 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. The following possible header types are defined in [6]: mesh header, broadcast header, fragmentation header, compressed or not compressed IPv6 header. The only header that has to be necessarily included is the IPv6 header. In case of inclusion of more than one header, the order of appearance is the one in figure 1, which illustrates the case in which all the headers are included.

Fig. 1. IEEE 802.15.4 data frame including 6loWPAN possible headers and IPv6 payload.

The mesh header has been introduced to forward 6LoWPAN datagrams over multiple radio hops and support layer-two forwarding. The broadcast header allows to support multi-hop forwarding capability in case of broadcast/multicast communication. The fragmentation header has been introduced since i) the maximum physical layer packet size of IEEE 802.15.4 is only 128 bytes, ii) according to IPv6 specification [16], every link in the network must support a Maximum Transfer Unit (MTU) size at least equal to 1280 bytes, and on any link that cannot transport a 1280 bytes packet in one piece, link-specific fragmentation and reassembly must be provided at a layer below IPv6, iii) IEEE 802.15.4 MAC layer does not support any fragmentation/reassembly function. Finally, the possibility of compressing the IPv6 header has been introduced since i) at most 102 bytes are available for the IEEE 802.15.4 payload, ii) being IPv6 headers 40 bytes long, the space available for upper-layer protocols would be at most 62 bytes without any compression scheme. The rationale behind the compression solution adopted by 6LoWPAN consists in omitting redundant information that i) either can be derived from the IEEE 802.15.4 header (e.g. interface identifiers, packet length) or ii) is common for all nodes (e.g. protocol version, link-local or global prefix, traffic class, flow label). Under this way of operation, the only field that can not be compressed is the Hop Limit, which needs to be decremented for each forwarding hop.

IV. Hardware Constraints of Typical IEEE 802.15.4 Devices

This section presents the hardware constraints of typical IEEE 802.15.4 devices. Such constraints pose several challenges in the identification of basic services and functionalities to be used in conjunction with 6LoWPAN solution to fully enable the interaction between IEEE 802.15.4 LR-WPANs and IP networks.

[2]We recall that there are three logical roles a device can play in an IEEE 802.15.4 LR-WPAN: i) coordinator, which provides synchronization services through the transmission of beacons, ii) PAN coordinator, which is the principal coordinator of the PAN (each IEEE 802.15.4 network has exactly one PAN coordinator) and it always starts a new PAN after selecting a proper frequency channel on which to operate, iii) end device, which is a simple device. The logical role a node plays depends also on whether it is a full-function device (FFD) or a reduced-function device (RFD). FFDs implement the complete protocol set, so that only FFDs may play the role of PAN coordinator and coordinator. RFDs implement only minimal protocol functionalities, and they can only play the role of simple end devices.

In what follows, we assume as reference hardware a commercial and wide-deployed 802.15.4 chip, the Texas Instruments' System-on-Chip CC2430 [17], that may be regarded as representative of the whole category.

This chip has a 2.4 GHz IEEE 802.15.4 RF transceiver, an 8051 micro-controller, a flash memory of 128 Kb, 8 Kb of RAM memory (able to maintain only 4 kb of data when the system goes in sleep mode). The extension area of the SoC is $7mm \times 7mm$. Its power consumption is $27mA$ when transmitting or receiving data, while it is limited to $0.5\mu A$ in sleep mode. According to the IEEE 802.15.4 standard, the theoretical raw data rate is 250 Kbps.

Two closed-source protocol stacks are available for the SoC CC2430: i) "TIMAC", that implements the IEEE 802.15.4 MAC sublayer and requires about 40 Kb of the total flash memory, and ii) "ZStack", that implements the IEEE 802.15.4 MAC sublayer and the upper ZigBee layers, requires about 100 Kb of the total flash memory and leaves less than 30 Kb for applications and services.

The flash memory occupation is not just an implementation issue, as it makes impossible the coexistence of both IPv6 and ZigBee stacks.

Moreover even if forthcoming devices will be reasonably more powerful in terms of memory storage (for instance TI's SoC CC2530 supports up to 256 Kb of flash memory), it is necessary to take a natural design trade-off into account: advances in technology could indeed impact *both* system capabilities (e.g. increasing the storage memory) *and* system integration (e.g. reducing the surface/power consumption of the chip and a s a consequence reducing storage capabilities).

V. Basic protocol/functionalities to be used in conjunction with 6LoWPAN

In this section we focus on a limited set of basic functionalities, to be used in conjunction with LoWPAN adaptation layer, and after critically reviewing existing literature solutions, we try to identify the most suitable one for the hardware constraints and limitations which typically 802.15.4 devices exhibit.

The section is organized in four subsections, which address the issue of what multihop intra-PAN routing protocol (subsection V-A), automatic address assignment mechanism (subsection V-B), transport protocol (subsection V-C), automatic network and service discovery mechanisms (subsection V-D) should be selected to operate in conjunction with 6LoWPAN in resource-constrained devices.

A. Multihop Intra-PAN Routing

The 6LoWPAN IETF working group assumes that each node is equipped with a proper routing table to support multihop forwarding capabilities, but it does not mention how to fill such routing tables. So a multihop routing protocol has to be introduced.

On the one hand, the selection of the multihop intra-PAN routing protocol has to be done according to suitable requirements, defined by ROLL working group in [13], which take into account the main characteristics of IEEE 802.15.4 LR-WPANs, such as low bandwidth, short range, scarce memory capacity, limited processing capability and other attributes typical of inexpensive hardware. On the other hand, 6LoWPAN IETF working group has recommended in [18] other reasonable routing requirements. Among these, we cite small code size and routing state to fit the typical 6LoWPAN node capacity (RAM size from 2KB to 8KB and flash memory size from 48 KB to 128 KB), efficient use of routing control packets to minimize power consumption, small size of routing control messages to avoid fragmentation of physical layer frames, robustness against packet losses, reliability in presence of unresponsive nodes, scalability.

According to the above requirements and after analysing the different routing protocol available for LR-WPANs and MANET, our opinion is that AODV is to be preferred to the other protocols. The reasons are the following: i) simplicity, ii) lower signalling overhead and memory load than in proactive routing protocols, iii) AODV has been evaluated and selected by ZigBee Alliance to be included in the ZigBee Protocol stack, iv) as it will be explained in subsection V-B, AODV route requests and replies may be exploited for address auto-assignment and duplicate address detection.

B. Automatic address assignment

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 LoWPAN adaptation layer.

Several address auto-configuration solutions have been proposed in the literature for mobile ad hoc networks (MANETs). However, are these solutions well-suitable also for 6LoWPAN-enabled IEEE 802.15.4 LR-WPANs case?

Generally speaking, address auto-configuration protocols for ad hoc networks can be classified as stateful and stateless.

In the stateful approach, each node has to maintain detailed state information about the utilization of the MANET address space. This state information is usually represented by an address allocation table that contains the addresses currently in use within the ad hoc network. The main challenge of such approach is the high signalling overhead necessary for the maintenance of the allocation table consistency, which is not guaranteed at all in presence of packet losses and network merging.

In the stateless approach each node selects autonomously and randomly its own address and performs a Duplicate

Address Detection (DAD) procedure to verify its uniqueness and resolve conflicts.

According to a taxonomy introduced in [19], DAD procedures can be classified as Strong and Weak. Strong DAD is used by new nodes joining the network to check if the selected address is already in use. The main drawback of these solutions is that they do not quickly respond to network split/join, even if the procedure is repeated periodically. Perkins et al. provide in [20] an example of Strong DAD. Specifically, when a node joins the network, it selects two addresses: a random address and a temporary address. The last one is selected from a "reserved" range of addresses. In this way, it floods the network with an address request (AREQ) message originated from the temporary IP address. If the selected address has already been assigned, the colliding node sends back to the temporary address of the joining node an address reply (AREP) message. If no AREP message is received, after a timeout the selected random address may be assigned; otherwise, the joining node retries with another randomly selected address. In the above solution the DAD procedure may fail in case of high-delay networks and message losses

Weak DAD copes with unbounded delay messages and avoids the use of time outs. To circumvent the chicken-and-egg problem of "how to signal that an address duplication occurs if IP is not usable because of the address duplication", in [19] Vaidya proposes to make every node select an unique ID string to be associated with the IP address in routing packets. In such a way, i) each entry in the routing table is unambiguously identified by the pair $< address, key >$, ii) address conflicts are detected when multiple keys are associated with the same address, and iii) duplicate address detection fails only when different nodes select the same address and the same key.

With reference to IEEE LR-WPANs, the stateless approach is the most suitable for IEEE 802.15.4 LR-WPANs, since it does not require to store the address allocation table. This represents a desirable memory saving especially if we consider that RAM size of typical IEEE 802.15.4 devices is about 8 KB, that can even half itself in sleeping mode. Theoretically, both solutions in [20] and [19] can operate in IEEE 802.15.4 networks. In fact, a solution like the one in [20] could take advantage from reactive routing protocols, whose route discovery requests and replies could be used as address request (AREQ) and address reply (AREP) in the DAD procedure. Likewise, a solution like the one in [19] could use the 64-bit MAC address as unambiguous key. In the last case, however, it is necessary to store for each routing table entry also the 64-bit MAC address and, as a consequence, routing tables would require more memory space. This is the main reason why we consider more suitable a solution like the one in [20] for the self-configuring address assignment in the proposed protocol stack.

C. Transport protocol

The best choice would be certainly to include both UDP and TCP.

UDP is for sure simpler to implement than TCP. In addition, the authors of [6] mention only UDP as possible transport protocol and suggest a possible compression scheme for UDP headers to reduce UDP header size from 8 bytes to just 4 bytes.

As regards TCP, we can identify two main factors that could prevent its use: i) limited flash ROM memory of a device and ii) limited RAM memory.

The first one could prevent the implementation of a protocol stack including TCP as transport protocol. Taking in account our implementation, we can claim that the IEEE 802.15.4 and 6LoWPAN functionalities have a code size of about 60 Kb (out of the 128 Kb totally available on the reference hardware) in the implemented test-bed. The implementation of TCP in μC/TCP-IP [21], 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 25 to 35 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 dedicated 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.

According to the above reasoning, we can state that TCP over LoWPAN over IEEE 802.15.4 would not perform well and that simpler and more lightweight transport protocols, such as UDP, would be more suitable, given the limited memory resources of IEEE 802.15.4 nodes. In addition, loss recovery mechanisms could be provided by the application and the IEEE 802.15.4 link layer and operate in conjunction with the (unreliable) UDP transport.

D. Network and service discovery

According to the current IEEE 802.15.4 standard, the network discovery process exploits the transmission of beacon frames from coordinators, both in beacon-enabled mode and in non beacon-enabled mode. The information that IEEE 802.15.4 nodes can deduce from the examination of the beacon frames during the network discovery phase mainly relates to i) the (64 bit) PAN identifier of the network, ii) the logical channel occupied by the network, iii) the capability of the network to accept joining requests. As a consequence, IEEE 802.15.4 nodes can choose a network on the basis of the

PAN identifier only and cannot automatically (e.g. with zero-configuration) select a network based on the offered services. Thus, IEEE 802.15.4 nodes can only exhaustively associate with all the available networks, until services of interest are discovered in one of these networks.

To allow self-configuring network discovery procedures in IEEE 802.15.4 LR-WPANs, we propose the creation of a "default discovery WPAN", with which each new joining node initially associates to find out information about the services offered by available WPANs in the node service area. As regards how the default discovery WPAN collects information about the existing WPANs in the surrounding area and the offered services and how a joining node take advantage of the presence of the default discovery WPAN, we propose a solution based on the interaction between 6LoWPAN/IPv6 gateways and the joining node. A 6LoWPAN/IPv6 gateway is the interconnection node between IEEE 802.15.4 6LoWPAN and IPv6 networks and its goal is to translate the packet format in case of compression and to cope with possible fragmentation/reassembly operations acting in stateless and lightweight manner since it has not to store any state information about address translations and on going communications.

As depicted in figure 2, each 6LoWPAN/IPv6 gateway announces its WPAN (PAN identifier, offered service, IPv6 address prefix of the network and other useful information to the service discovery) by transmitting DNS-SD/mDNS [22] messages to a multicast group which also the gateway of the default discovery WPAN belongs to (on the backbone that interconnect the WPAN gateways). When DNS-SD/mDNS messages with the discovery information are received by the gateway of the default discovery WPAN, a database of available WPANs and their offered services is created. The requirements for the creation of the database in the gateway of the default WPAN is that both WPAN gateways and default WPAN gateway are associated with the same multicast group to send and receive DNS-SD/mDNS messages. A new joining IEEE 802.15.4 6LoWPAN device i) uses the IEEE 802.15.4 facilities for channel scanning and finds out the WPANs available in its neighbourhood, ii) joins the default discovery WPAN and queries the 6LoWPAN/IPv6 gateway about the discovered WPANs, iii) selects a suitable WPAN based on the offered services.

From the IPv6 point of view, when a node joins the default discovery WPAN, it can communicate with the default discovery WPAN gateway by using link local addressing. In this way, if the default discovery WPAN gateway is assigned the address $FE80 :: 1$ (or any other IPv6 anycast address), not only new joining nodes know a priori the address of the default discovery WPAN gateway, but also gateways of different WPANs can have the same IPv6 address, since an IPv6 local address is valid only in the subnetwork where it is used.

The above solution calls for: i) a reserved PAN ID, in such a way that each node is able to recognize the default discovery WPAN, ii) a pre-existing network infrastructure that is responsible of creating and maintaining the default discovery

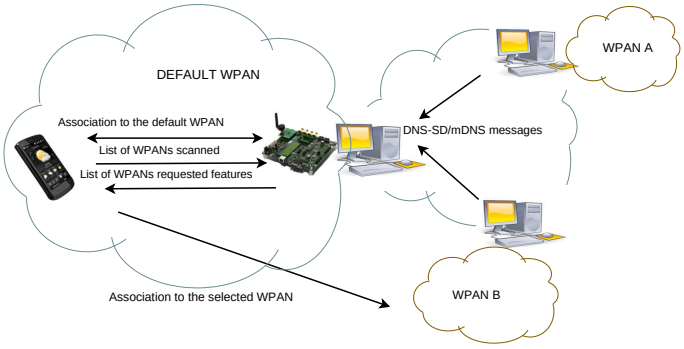

Fig. 2. Interaction between 802.15.4 nodes and default discovery WPAN gateway.

WPAN. The last requirement is due to the fact that an ordinary IEEE 802.15.4 node is not allowed to simultaneously join different WPANs. Thus, if the first (ordinary) joining node does not find any default discovery WPAN and creates such PAN by using the reserved PAN ID, later on it would not be able to join other WPANs of interest.

Besides the default discovery WPAN, we propose the introduction of a default free WPAN, with which nodes may associate with and where nodes announce their identity, if they want to communicate with other nodes, even in absence of pre-existing default discovery WPAN and other WPANs. A similar WPAN would allow the automatic discovery and communication among nodes and would be extremely useful as a social networking facility. Like the default discovery WPAN, the default free WPAN should be assigned a reserved WPAN ID. However, differently from what happens for the default discovery WPAN, each node is allowed to create and maintain a default free WPAN.

VI. A Possible Protocol Architecture

A high level overview of a possible protocol architecture is depicted in figure 3.

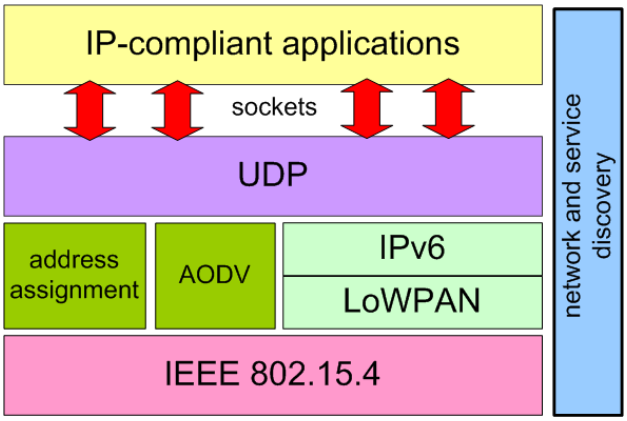

Fig. 3. High level overview of the proposed protocol architecture.

This architecture integrates the different solutions we discuss in the previous section. The standard IEEE 802.15.4 is used at MAC sublayer. The protocol stack in figure 3

includes also AODV for the routing algorithm, the automatic address assignment module, the 6LoWPAN adaptation layer. The availability of IPv6 at network layer allows to use UDP as transport protocol. The stack offers standard socket APIs to application developers. Thus, as regards the application layer, applications may take advantage from socket APIs to interact with the underlying transport protocol. Finally, the protocol stack in figure 3 includes the module for network and service discovery described in the previous section; this is depicted as orthogonal with respect to the protocol stack, since it requires support from all the protocol stack layers.

A. 6loWPAN/IPv6 gateway architecture

Figure 4 depicts a high level overview of the architecture of a 6LoWPAN/IPv6 gateway. As it can be seen, a 6loWPAN/IPv6 gateway may be regarded as the union of two separate components: the IEEE 802.15.4 node and the IPv6 node. The architecture of the IEEE 802.15.4 node is almost identical to the one presented in figure 3, with the only exception of two specific modules for decompression/compression of packet format and the transmission/reception of packets to/from the serial line. The IPv6 node plays instead the twofold role of i) forwarding the packets received through the serial line toward the public Internet and viceversa and ii) providing the network and service discovery functionalities described in the previous section.

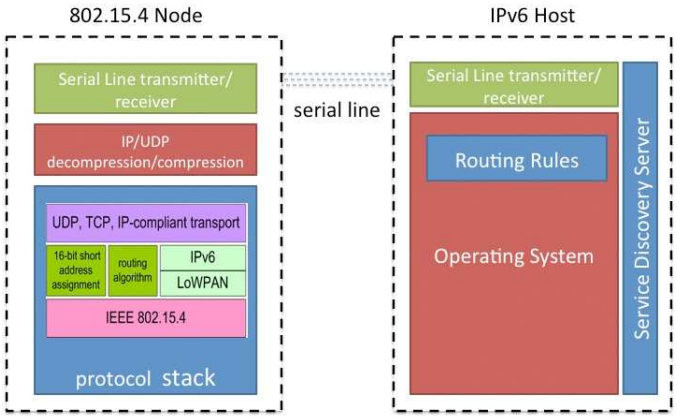

Fig. 4. High level overview of a 6LoWPAN/IPv6 gateway protocol architecture.

VII. Description of the implemented test-bed

The protocol architecture presented in section VI has been implemented in a real test-bed in two different phases.

During the first one, described in subsection VII-A, we tested the basic functionalities of the proposed protocol architecture (transmission/reception of IPv6 datagrams using LoWPAN adaptation layer, automatic address assignment, interconnection with the public Internet, simple service discovery). For this purpose, we used a development kit composed of very simple nodes equipped at most with a simple display.

In the second phase of the implementation, described in subsection VII-B, we developed a more realistic application scenario, where PDAs equipped with Secure Digital Input/Output (SDIO) cards are able to run the firmware implementing the proposed protocol stack.

A. Implementation of the basic functionalities of the proposed solution

The general reference scenario of the basic test-bed is depicted in figure 5. 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 5 shows also that such kind of interaction between IEEE 802.15.4 6LoWPAN networks and the Public Internet calls for a 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.

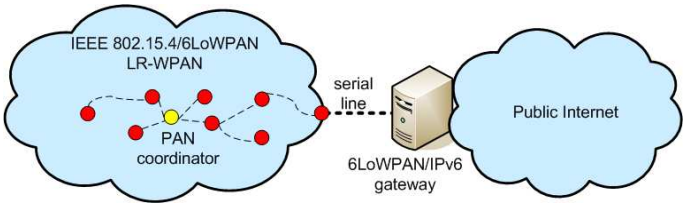

Fig. 5. Reference scenario for the implemented basic test-bed.

As regards the test-bed hardware, we used the Texas Instruments' CC2430ZDK [17] 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 microcontroller and the flash memory. The kit includes three simple minimal boards (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 6.

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.

1) IEEE 802.15.4 6LoWPAN nodes: All IEEE 802.15.4 nodes implements the proposed protocol stack. This holds for both the PAN coordinator or end device.

In more detail, we used the so called TIMAC, that is Texas Instruments' implementation of the IEEE 802.15.4 MAC protocol, as lowest layer and starting point of the proposed protocol stack. The upper layers have been instead implemented from scratch.

Whereas the interaction between the implemented stack and the TIMAC is based on the APIs provided by Texas Instruments, the interaction between the implemented stack and the application layer are based on Socket compliant APIs,

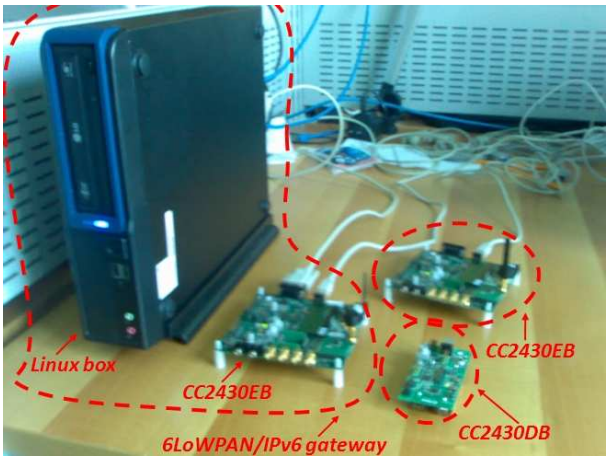

Fig. 6. Test-bed hardware components.

whose prototypes are almost equal to the standard IPv6 socket functions used by the most used operating systems, in order to simplify the learning process for application developers.

As regards the packet creation/processing during the transmission and reception operation, each packet is dealt as a sequence of bytes. In this process, the *socket_buf* data structure plays a fundamental role. In the reception phase, the *socket_buf* data structure allows to parse the received packet by storing the values of the packet fields. In the transmission phase, the *socket_buf* data structure allows to derive the packet to be transmitted by storing the values that have to be included in the packet fields. In doing this, we took free inspiration from the homonym *socket_buf* kernel Linux implementation and we adapted it to the context of low-energy and low-power devices. Thus, the implemented *socket_buf* data structure is much simpler and lighter, and it is about 100 bytes sized. We also used the *socket_buf* data structure to store some state information that must be shared among different stack layers (for instance, information on eventual UDP header compression has been included in the 6LoWPAN header).

2) 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 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 and can operate in stateless manner. So, unlike the solutions presented in section II, where a stateful gateway is used to permit the interconnection between IEEE 802.15.4 LR-WPANs and the whole IPv6 public Internet, the gateway in the proposed solution acts as a simple layer-3 router and does not execute any protocol translation operation. The only operation performed by the gateway is the compression/decompression of IPv6 and UDP headers for packets directed to/from the 6LoWPAN network, which means simply adding/striping the 6LoWPAN adaptation header.

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 [23]. With regard to the tunnelling protocol, we chose a non-standard solution based on *AYIYA* [24] in place of the standard proto-41-like solutions [25], 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) [26] 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) [27] protocol to carry data over the serial line. Even if SLIP supports the only transmission of IPv4 datagrams and as such it calls for encapsulating data into IPv4 packets, we preferred SLIP to other standard and custom protocols to transmit IP packets over a serial line, because 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 and this is just the case of SLIP. SLIP limits indeed itself to define two special characters, END (0xC0) and ESC (0xDB), in such a way that i) each IPv4 packet is terminated with an END character, ii) if the IPv4 packet contains an END character, this is replaced by the two byte sequence of ESC and octal 334, iii) if the IPv4 packet contains an ESC character, this is replaced by a two byte sequence of ESC and octal 335. 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.

B. Deploying the proposed solution in a real scenario

We conceived an application scenario where in a shopping center are deployed two different IEEE 802.15.4 LR-WPANs and a default discovery WPAN. With respect to the basic implementation described in subsection VII-A, where TI's minimal board (CC2430DB) and evaluation boards (CC2430EB) play the role of simple IEEE 802.15.4 devices, we used PDAs equipped with Secure Digital Input/Output (SDIO) cards, where the TIMAC and the implemented protocol stack run in

form of firmware. In such a way, we were able to fully exploit the PDAs equipment (display, keypad, etc.) and to develop applications based on the IPv6/6LoWPAN capabilities of the proposed protocol stack.

As depicted in figure 7, in order to enable the interaction between the SDIO and the applications on the PDA, we developed an API inspired by the Berkeley socket interface: using this API the application developer can simply re-use/adapt existing applications or develop from scratch IP-enabled applications with minor effort.

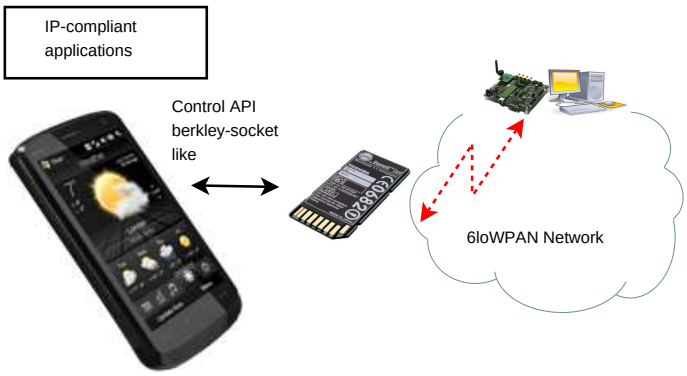

Fig. 7. Interaction between the mobile node and the 6loWPAN-enabled SDIO.

More precisely, we implemented an application that interacts with the firmware loaded on the SDIO cards and allows users i) to visualize on the PDA display the available WPANs and information about the offered services, exploiting the capability of the network and service discovery framework, ii) to chat with other users, even if connected to different WPANs, iii) receive advertisement messages.

The reference scenario for the application is depicted in figure 8. WPAN1 and WPAN2 denote the two WPANs in the shopping centres, and G1 and G2 denote the corresponding IPv6/6LoWPAN gateways. The default discovery WPAN makes possible the network and service discovery, when a PDA wants to join a WPAN, and operates as described in subsection V-D. As regards the chat function, we developed a single room client-server chat as a simple proof-of-concept of the protocol stack functionalities. The remote chat server depicted in figure 8 manages user connections and disconnections, buddy lists, and it allows buddy users to discover each other. In addition, in case of users connected to different WPANs, the chat server makes possible the communication by receiving messages from the IPv6/6LoWPAN gateway of the source WPAN and forwarding messages to the IPv6/6LoWPAN gateway of the destination WPAN. As regards the advertisement server, it interacts with the IPv6/6LoWPAN gateways to send advertisements and promotional messages, that PDAs receive in form of pop-up messages. These messages can also contain URLs that users can access through the default web browser by using the GSM/UMTS and/or Wi-Fi connectivity.

The above application was developed for Windows Mobile 5.0 and PocketPc 2003, and it is based on Microsoft Foundation Classes (MFCs).

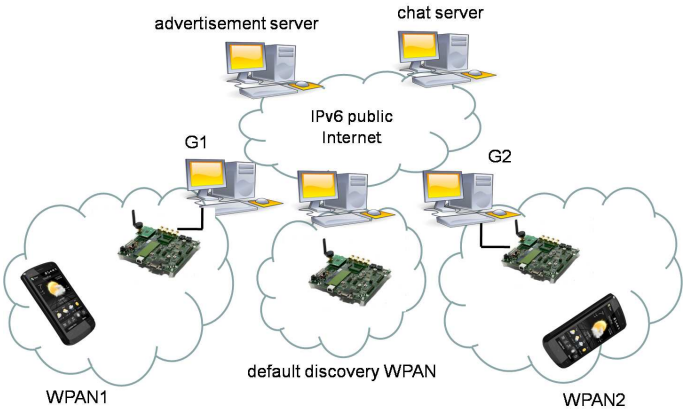

Fig. 8. Reference application scenario.

Figure 9 shows three screenshots of the test-bed applications.

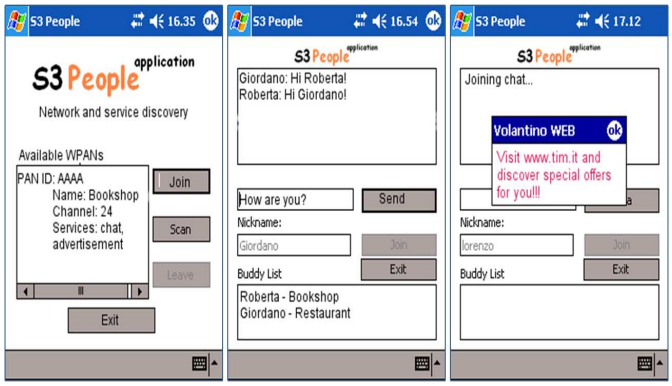

Fig. 9. Screenshots of the GUI of the developed application.

VIII. Conclusions and Future Work

In this paper we presented a protocol stack for fully IP-enabled and self-configuring IEEE 802.15.4 LR-WPANs. On the one hand, the stack includes the LoWPAN adaptation layer to support the transmission/reception of IPv6 packets. On the other hand, the stack includes also AODV as routing protocol, UDP as transport protocol, a protocol for 16-bit short address auto-configuration and assignment, a mechanism for network and service discovery. We also presented a test-bed, which implements the proposed protocol stack on a SDIO firmware and describes a use case where mobile nodes communicates in a simple application scenario.

While in this work we took into account only the 6loWPAN specifications there are many others recently proposed IETF internet draft: future work will integrate in the stack the new compression format for IPv6 datagram [28] and the RPL routing protocol [29] designed by the ROLL working group especially for Low power and Lossy Networks (LLNs).

The energy saving features of the IEEE 802.15.4 standard allow nodes to go temporarily down during a sleeping period.

This functionality makes impossible the reachability of a sleeping node and could lead to temporary network splits. This issue will be taken in account in future works by i) the introduction of the publish/subscribe paradigm in IEEE 802.15.4 WPANs and ii) the integration of MANET and DTN (Delay Tolerant Network) routing functionalities to cope with network partitioning.

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] M. Turolla, E. Alessio, *ZSIM enabling innovative services to improve quality of life*, white paper on line available at www.zigbee.org/imwp/download.asp?ContentID=10403

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

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

[7] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulo, K. Kim, *TinyREST: A Protocol for Integrating Sens Networks into the Internet*, in Proceedings of REALWSN, 2005.

[8] S. Krishnamurthy, *TinySIP: Providing Seamless Access to Sensor-based Services*, in Proceedings of Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services, 2006.

[9] S. Sakane, Y. Ishii, K. Toba, K. Kamada, N. Okabe, *A translation method between 802.15.4 nodes and IPv6 nodes*, in Proceedings of International Symposium on Applications and the Internet (SAINT), 2006.

[10] R. Wang, R. Chang, H.Chao, *Internetworking Between ZigBee/802.15.4 and IPv6/802.3 Network*, in Proceedings of SIGCOMM 2007 Workshop ”IPv6 and the Future of the Internet”, 2007.

[11] blip project web site, http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip

[12] M. Durvy, J. Abeill, P. Wetterwald, C. OFlynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, A. Dunkels, *Making sensor networks IPv6 ready*, in Proceedings of the 6th ACM conference on Embedded network sensor systems, 2008.

[13] M. Dohler et al., *Routing Requirements for Urban Low-Power and Lossy Networks*, RFC 5548, May 2009.

[14] C. Bormann, D. Sturek, Z. Shelby, *6LowApp: Problem Statement for 6LoWPAN and LLN Application Protocols*, Internet Draft.

[15] RFC 4862, *IPv6 Stateless Address Autoconfiguration*.

[16] RFC 2460, *Internet Protocol Version 6 (IPv6) Specification*

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

[18] Internet Draft, *Problem Statement and Requirements for 6LoWPAN Routing*

[19] N. Vaidya, *Weak Duplicate Address Detection in Mobile Ad Hoc Networks*, in Proceedings of ACM MobiHoc, 2002.

[20] C. Perkins, J. Malinen, R. Wakikawa, E. Belding-Royer, Y. Sun, *IP address autoconfiguration for ad hoc networks*, expired Internet Draft, 2001.

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

[22] Multicast DNS (mDNS) reference web site, http://www.multicastdns.org/

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

[24] 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

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

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

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

[28] draft-ietf-6lowpan-hc-07, *Compression Format for IPv6 Datagrams in 6LoWPAN Networks*, April 8, 2010

[29] draft-ietf-roll-rpl-07, *RPL: IPv6 Routing Protocol for Low power and Lossy Networks*, March 8, 2010