Wireless sensor network interconnection protocol

Wireless Sensor Network Interconnection Protocol

Kurt Smolderen[1], Peter De Cleyn and Chris Blondia
Dept. of Mathematics and Computer Science – PATS Research Group
University of Antwerp – IBBT
Middelheimlaan 1, B-2020, Antwerp, Belgium
Email: {firstname.lastname}@ua.ac.be

***Abstract*—Wireless Sensor Networks (WSNs) are at high speed gaining in popularity. Because of this increased adoption, the need for integration with existing technologies becomes more prominent.**

In practice, a WSN is often deployed in environments where both wired and wireless data networks are also present. Consider for instance a WSN which acts as a climate monitoring or fire detection system in a large office building. As WSNs are typically networks with high constraints regarding resource efficiency, it would be beneficial if they could use the available data networks as additional transport medium for their internal data.

In this paper we describe an interconnection protocol and associated sensor gateway design which enables the transparent use of both wired and wireless data networks by WSNs. Through this setup, the WSN will detect the non-sensor devices as virtual sensor nodes and consider them when determining the optimal routing paths.

***Index Terms*—gateway design, integration, internet protocol, protocol architecture, wireless sensor networks**

I. Introduction and related work

In their article *Interconnecting Wireless Sensor and Wireless Mesh Networks: Challenges and Strategies*[1], Bouckaert et al. give an in depth overview of the complexity which should be taken into account when interconnecting a Wireless Sensor Network (WSN) and a Wireless Mesh Network (WMN). A certain set of mesh nodes is converted into a set of sensor gateways by adding the appropriate radio interface to each node. Depending on the needed featureset and allowed complexity, multiple scenarios are possible, each bringing its own level of (transparent) integration between the available networks.

In contrast to wireless sensor nodes, IP based non-sensor nodes do not suffer from the hard constraints regarding energy efficiency. These kinds of nodes are often connected to a power-grid or a much larger battery compared to their sensor equivalents. Offloading some of the network or computational load from a sensor device to one of the latter devices will therefore be beneficial to the WSN's overall power consumption and might improve network latency and throughput due to the higher bandwidth available in the non-sensor networks.

Although interconnecting WSNs with IP based networks has been done before, the main driver herefore was the need to collect the sensor readings on a remote location [2]. Many of these solutions are proxy based. The proxy device acts as a default sink for the sensor nodes and stores the sensor readings. When a remote system asks the proxy for sensor data, cached results are returned [3] or – in case real-time information is needed – a separate request is sent from the proxy to the sensor node [4]. Regardless of the used data retrieval method, no direct connection between the querying system and the sensor node is established.

Lei et al. have developed a Virtual IP Bridge which enables IP integration of WSNs and IP enabled networks [5]. Their solution is manly focused on IP interconnectivity and does not realize route optimizations in the whole virtual sensor network.

The design of the Wireless Sensor Network Interconnection Protocol (WSNIP) presented in this paper allows sensor nodes to route traffic over alternative IP based networks and facilitates direct communication between a sensor node and any host within the network on which the protocol is deployed. To reach this goal, WSNIP transforms hosts into virtual sensor nodes. These virtual sensor nodes are nodes on which an additional network stack is emulated which is fully compatible with the network stack used in the WSN. WSNIP's main benefit is twofold: (1) when finding routes between (virtual) sensor nodes all constraints implied on the WSN routing protocol are met and (2) no specific configuration for the IP backbone network is necessary.

The rest of this paper is structured as follows. In section II we will shortly discuss one of the most popular addressing schemes in WSNs to allow easy integration with existing IPv6 based networks. Section III will highlight the Wireless Sensor Network Interconnection Protocol itself, while section IV will focus on the design of a Wireless Sensor Gateway. We will conclude this paper with some last thoughts in section V.

II. IP based addressing in wireless sensor networks

Sensor devices fit well in the concept of the *Internet of Things* [6]. Their purpose is to gather information and make the collected data accessible to other systems. This information sharing imposes the need for a connection between different devices.

Several communication protocols have been developed to provide constrained wireless devices with a well designed network stack. The *Zigbee specification* [7] was one of the first industry-ready protocols covering end-to-end communication over multiple hops. The Zigbee protocol stack uses a proprietary layer 3 protocol on top of the used IEEE 802.15.4

[1]Kurt Smolderen is also affiliated with Karel de Grote University College, Departement of Applied Engineering, Salesianenlaan 30 B 2660 Antwerp Belgium

(802.15.4) [8] MAC protocol. Changing the proprietary network protocol to the de-facto IPv4 [9] or IPv6 [10] standards would largely simplify the integration of these devices into a globally connected *Internet of Things*.

The disadvantage of IP in WSNs is the overwhelming overhead it brings along. The maximum size allowed for a 802.15.4 frame is 127 bytes. Between nine and 39 bytes are reserved for the MAC header and an additional 20 (IPv4) to 40 (IPv6) bytes would be consumed by the IP header.

To overcome this unacceptable overhead of at least 22.8%, the 6LoWPAN standard was introduced [11]. 6LoWPAN provides us with a mechanism to highly optimize the transport of IPv6 packets over 802.15.4 frames and halves the minimal overhead. The additional compression is achieved by omitting IPv6 fields with fixed or predefined values (such as the IP version field) and by using address compression schemes.

III. The Wireless Sensor Network Interconnection Protocol

In theory, it should be easy to integrate wireless sensor nodes which use IPv6 or 6LoWPAN as addressing scheme with other IP enabled network devices or networks. In practice however, things turn out to be more complex because of a variety of reasons:

1) Sensor nodes are only seldom equipped with multiple interfaces. It is therefore practically impossible to link them with devices using another network technology at the PHY or MAC layer.
2) Processing power is very limited on most sensor devices which limits the possibility to deploy computational intensive or memory consuming network protocols on them.
3) Routing protocols used within WSNs and IP-based backbone networks are most of the time incompatible.
4) Most networks are still IPv4 based as IPv6 penetration is still far from global. Transition methods are therefore needed to route IPv6 traffic across these networks.

A protocol designed for interconnecting WSNs must therefore be able to overcome the above listed obstacles. Not all of these limitations need to be present at the same time however. The solution must thus be easily adaptable depending on the actual requirements. By its modular design, WSNIP supports flexible configurations to adapt to the present environment.

WSNIP focuses on a transparent optimization of wireless sensor routes by using less restricted network paths as a virtual sensor link. Due to its concept, it not only allows to create virtual links between WSNs, but also between a WSN and host systems. As an added gain, using WSNIP may lead to better routes when sensor paths become large or when links within a sensor path appear to be unstable because of external factors (e.g. interference). A graphical representation of the protocol's application area can be seen in Fig.1.

WSNIP is typically deployed on devices acting as a gateway between a WSN and a backbone network. Such WSNIP enabled device is called a Wireless Sensor Gateway (WSG). It must be noted however other deployment scenario's are

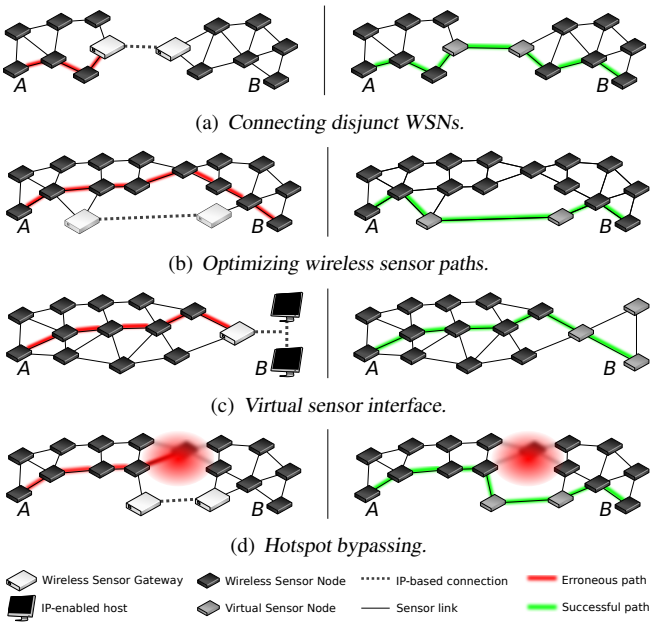

Fig. 1: Scenarios for which WSNIP provides a solution (A tries to communicate with B). The real network topology is shown at the left. The topology as seen by the (virtual) sensor nodes is displayed at the right.

also possible in case WSNIP is used to facilitate direct communication between a computer system and a WSN.

As stated before, the basic idea behind WSNIP is to establish virtual sensor links between all deployed WSGs. From a sensor point of view, each device on which WSNIP is deployed becomes thereby a one hop neighbour of all other available WSNIPs-enabled devices. The difference between the real and virtual topology is shown in Fig.1.

In order for the virtual topology to work, WSNIP must mimic all functionalities available on normal sensor devices with routing support. Routing protocols in sensor networks tend to be highly optimized towards energy consumption and therefore differ from the routing protocols used in less energy constrained IP networks. As a consequence, WSNIP has to provide its own WSN-compatible routing protocol. There should be no conceptual implementation difference between the protocol installed on the sensor devices and the one deployed on the WSGs except for the number of interfaces they have to support. Regular sensor devices have only one radio interface while a WSG should mimic two interfaces: one to the real WSN and the other to the virtual WSN formed by the WSGs.

Sensor packets transported over a virtual sensor link are converted to full IPv6 packets. To forward this converted data over the backbone network, existing protocols such as IPv6 over Ethernet [12], IPv6 over IPv4 [13][14][15], IPv6 over UDP [16] or even IPv6 over IPv6 [17] can be used (Fig.2). The most advisable choice depends both on the characteristics of the backbone network and the protocols used within it and will be discussed in section IV.

Packets arriving at the WSG terminating the virtual sensor link, will be converted back to 6LoWPAN compatible packets.

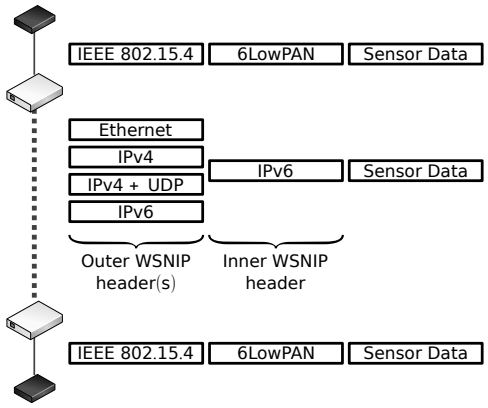

Fig. 2: Overview of the packet headers as used within WSNIP.

IV. The Wireless Sensor Gateway Design

A full WSNIP implementation consists of the following modules:

1) **Sensor Interface Module** Interfaces a real sensor-compatible radio interface.
2) **Sensor Routing Module** Implements the same routing logic used in the WSN the protocol is deployed on.
3) **Address Translation and Tunneling Module C**reates the WSNIP outer headers to transport the plain IPv6 packets between different WSGs.
4) **Wireless Sensor Gateway Discovery Module** Announces itself on the backbone network and detects other WSGs deployed in it.
5) **Virtual Sensor Interface Module** Allows the host on which WSNIP is deployed to communicate directly to other sensor nodes.

A. Sensor Interface Module

The complexity of the *Sensor Interface Module* depends on the used radio interface and the functionality covered by that interface. Depending on the use case, the hardware interface may be limited to a radio-chip or may be a full sensor device itself. In the latter case, it is possible the sensor device takes over responsibility for some of the WSNIP functionalities. If, for example, the sensor routing module is implemented in the sensor device, the corresponding module will be disabled.

Due to its flexibility, the *Sensor Interface Module* does not have any constraints regarding its input format. This means it should be tailored towards each individual use case. The output format however is well defined and constructed as follows:

- An IPv6 header replacing the IEEE 802.15.4 MAC header. MAC addresses are converted into IPv6 addresses according to the methodology explained in [11].
- An IPv6 header. The IPv6 source and destination address are set to the 6LoWPAN originator and destination address if present. In case a 6LoWPAN header is not present, the 802.15.4 originator and destination address is reused. All address conversions are conform to [11].
- The payload of the original received frame. A compressed UDP header should be first converted into a full UDP header. All other bytes are copied without modification.

The address used within the outer IPv6 header might be different from the one used within the original 802.15.4 frame and depends on the functionality the hardware sensor interface takes care of. If it has a built in routing block, the source address of the outer IPv6 header must be set to this node's own IPv6 address. The destination address should be equal to the next hop's IPv6 address. In case WSNIP is responsible for routing all data, the source address of the IPv6 outer header must be set to the originator MAC address of the original frame and the IPv6 destination address must be set to the frame's original (but converted) destination MAC address.

Another approach could have been to use a full (or pseudo) 802.15.4 header instead of the outer IPv6 header. The reason why we have chosen the latter approach is because it simplifies the internal workings of all other WSNIP modules as they have to deal with only one specific address type.

B. Sensor Routing Module

In most cases, the routing algorithm used in the backbone network will be different from the one used in the WSN. Therefore, the next hop node for each packet coming from or going to the *Sensor Interface Module*, must be defined before the packet is passed to the next WSNIP module. If the attached radio interface is more intelligent (i.e. a full functioning sensor device itself), the possibility exists it also acts as a sensor routing manager. The sensor device will then set the addresses in the pseudo MAC header correctly as explained in the previous section.

If a separate *Sensor Routing Module* is needed, it must conform to the following requirements:

- The module must implement the same routing protocol as used within the WSNs.
- Both MAC and network addresses in packets offered to the *Sensor Routing Module* may not be changed beforehand. This means the source address in the outer IPv6 header must correspond to the frame's sender MAC address and the destination address in this header should match this node's sensor MAC address (represented as an IPv6 address). The source and destination address in the inner IPv6 header should match the 6LoWPAN originator and destination address.
- The source and destination addresses in the MAC header should be set to the correct values to represent the next link which should be used to forward the packet towards its final destination. This means the IPv6 source address in the outer IPv6 address must be set to this node's address. The destination address of this header must be set to the address of the next hop.
- The routing module should have at least two interfaces. The first one links to the *Address Translation and Tunneling Module*, the other(s) to the *Sensor Interface Module* or *Virtual Sensor Interface Module*.
- The *Sensor Routing Module* should correctly handle 6LoWPAN's Hop count field and discard the packet if it is no longer allowed to travel over the WSN.

C. Address Translation and Tunnelling Module

The *Address Translation and Tunnelling Module* provides a virtual interface which connects all WSGs with each other. The functionality of this module is slightly different for incoming (from an other WSG) and outgoing (to another WSG) packets.

Packets which need to be forwarded to another WSG must have the outer IPv6 addresses set as explained in the previous sections. Further packet processing differs depending on the network addressing scheme used in the backbone network.

1) Ethernet encapsulation: In case the backbone network (1) is IPv6 addressable, (2) forms one LAN and (3) uses the same IPv6 prefixes as used by the 6LoWPAN to IPv6 address conversion, the IPv6 addresses contained within the outer IPv6 header are directly reachable over the LAN. To forward the packet, the outer IPv6 header is stripped and replaced by an Ethernet header. The MAC addresses corresponding to the Ethernet source and destination address are set to this node's MAC address, respectively the MAC address found by the IPv6 neighbour discovery protocol for the outer header's IPv6 destination address. When receiving an Ethernet encapsulated IPv6 frame, The Ethernet header is replaced again with an outer IPv6 header as described in the previous sections.

2) IPv4 encapsulation: When an IPv4 backbone network is used, a mechanism must be used to transport the IPv6 packet over an IPv4 path. Such protocols can be found in [13],[14],[15] and [16]. As an example, we will briefly discuss 6in4 [15] which encapsulates IPv6 traffic in IPv4 tunnels.

In this mode, the *Address Translation and Tunnelling Module* uses a database which maps IPv4 addresses of all known WSGs on their associated IPv6 sensor addresses. This database is populated by the *Wireless Sensor Gateway Discovery Module* which is discussed in the next section. For each packet, the outer IPv6 header is preceded by an IPv4 header. The IPv4 source address matches this WSG's IPv4 address. The IPv4 destination address is queried from the database from the outer IPv6 destination address.

The actions performed on incoming packets are exactly the opposite as the ones performed on outgoing packets. The IPv4 tunnel header is removed so only the bytes starting from the IPv6 outer header remain. This provides us with a packet representation exclusively used within each WSG and allows the WSG to process the packet further.

Some additional remarks should be noted here:

1) In case no IPv4 address is found in the database for a certain IPv6 address, the packet is dropped. This simulates the behaviour on a real sensor link where packets are transmitted, but no verification can be done at transmission time whether the transmission was successful.
2) Broadcasting as defined for IPv4 is replaced in IPv6 by a link-local multicast. Hence, original broadcasts in the WSN should be converted into an IPv6 link-local multicast packet.
3) The concept of IPv6 multicast addresses does not exist as such in IPv4. If the IPv4 network does not fully support IPv4 multicast, IPv6 multicast streams should be converted to multiple unicast IPv4 streams to each WSG.

Although it might seem inefficient to prepend the original packet with an additional header instead of replacing the outer IPv6 header, there is a good reason for it. The last two items mentioned above show that an original sensor broadcast is eventually split up in multiple IPv4 unicast messages. If the outer IPv6 header would be omitted, there is no way for a WSG to detect if the original packet was broadcasted over the WSN or not. Converting an IPv6 multicast into an IPv4 broadcast is not a valid solution either. The reason for this is because WSNIP does not influence the IPv4 routing table of the WSG. Broadcasting over the IPv4 domain might result in flooding the IPv4 network in some situations which must be avoided at any cost. An example of such a situation is an IPv4 Wireless Mesh Network.

3) IPv6 encapsulation: This mode can be used when the backbone network is IPv6 addressable but uses prefixes different from the ones used during the 6LoWPAN to IPv6 address conversion. The principle behind this operation mode is more or less the same as with IPv4 encapsulation: a database is used which contains the address mappings of each WSG's sensor and backbone address. Instead of prepending the original packet with an additional IPv6 header, the outer header is repopulated with the IPv6 backbone addresses of the WSGs. On packet reception, the opposite action is performed.

D. Wireless Sensor Gateway Discovery Module

The *Sensor Routing Module* and *Address Translation and Tunneling Module* use a database containing the IPv6 sensor addresses of all WSGs and their corresponding IPv4 or IPv6 backbone address if needed. This database can be populated in multiple ways.

1) Auto discovery: In case one can reach all nodes on the backbone network by a single IPv4 broadcast or IPv6 multicast, WSGs can detect each other without any intervention. Each WSG can advertise itself over the network at a regular interval by broadcasting a single packet containing its IPv6 sensor address, an advertisement sequence number and a lifetime for this advertisement. The announced lifetime should be greater than the advertisement interval. In order to deal with packet loss, it is best to take it at least twice as large.

Each WSG receiving the advertisement may only process the information if the sequence number contained within the advertisement is greater than the sequence number of the last received sequence number (using the correct arithmetic). When processing an advertisement, any existing address binding for the sending WSG is updated: the IPv6 address is replaced if needed and the validity of the address association is replaced by the advertised lifetime.

If no new advertisements are received before the lifetime of an association expires, the address association is removed from the WSG's database and sensor packets will no longer be forwarded towards the removed WSG.

2) Fixed configuration: The opposite of *Auto discovery* exists of a fixed, preconfigured database for each node. This approach can be used within networks where you do not have full control over the protocols used within the backbone

network (e.g. because of firewall rules). It is however clear this approach is error prone.

3) WSG coordinator: A third possible implementation could use a so called *WSG coordinator*. The backbone address of this coordinator must be known and reacheable by each WSG. Each WSG will now send its advertisements to this coordinator. The latter will thus have a complete and up to date overview of all associated WSGs.

When the WSNIP service is activated on a WSG, the coordinator is contacted and asked for the latest list of all known WSGs. When an address timeout occurs, the WSG contacts the coordinator before removing the address mapping from its own database. If the coordinator did receive a new advertisement for the corresponding address binding, the new lifetime is sent back towards the requesting WSG which will update its database in its turn. If no newer binding information is found, the WSG will remove the particular binding from its database and no longer forward any data to it.

E. Virtual Sensor Interface module

WSNIP can also be used to connect an IPv6 enabled host to a WSN. For this mode, a new virtual interface is created on the host. The *Virtual Sensor Interface* will bind to this virtual interface at one side. At the other side, applications will bind on this interface and use it as any other 'real' network interface. It thus allows transparent connectivity from the host to a WSN without any hassles for any application. The only limitations applicable are (1) the application should use IPv6 addresses for all communication purposes and (2) the protocols used on top of the network stack must be compatible with the ones used within the WSN. When configured to operate in this mode, WSNIP replaces the *Sensor Interface Module* with the *Virtual Sensor Module*.

Instead of using a virtual interface, the *Virtual Sensor Interface* can also bind to a real IPv6 enabled network interface, converting the host into a WSN-IPv6 gateway. In this operation mode, WSNIP will convert sensor routing messages to IPv6 Neighbor Solicitation and Advertisement messages and vice versa and thus allows bidirectional communication between WSNs and an IPv6 LAN. The only applicable limitation exists out of the used network prefixes: they must be the same for the IPv6 LAN and the WSNs.

V. Conclusion and Future Work

A first proof of concept implementation of WSNIP is used within the IBBT DEUS project [18]. The final network setup of this project is displayed in Fig.3. Use-cases from Fig.1(a), 1(b) and 1(c) were implemented and verified within this network. In the next phase, all use-cases will be evaluated and large scale performance testing will be carried out.

WSNIP optimizes communication within and establishes communication between sensor clouds. However the current protocol does not take in to account any specific routing properties of the used WSN or backbone routing protocol when establishing connections between WSGs. We plan to address this issue by introducing a unified routing scheme. The WSG will be responsible to interpret routing metrics used on each network and possibly transform these metrics between the networks.

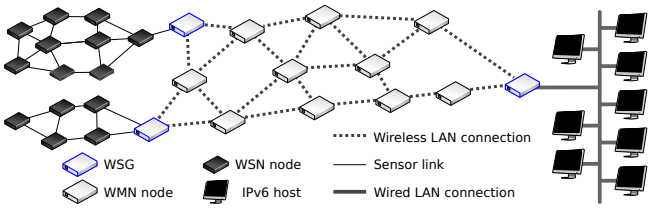

Fig. 3: WSNIP deployment within a DEUS network.

Acknowledgement

The research presented in this paper was partially funded by the IBBT-DEUS project. The authors wish to thank all partners from this project for their valuable feedback.

References

[1] S. Bouckaert, E. De Poorter, P. De Mil, I. Moerman, and P. Demeester, *Interconnecting Wireless Sensor and Wireless Mesh Networks: Challenges and Strategies*. IEEE, november 2009.

[2] P. A. C. D. S. Neves and J. J. P. C. Rodrigues, "Internet Protocol over Wireless Sensor Networks, from Myth to Reality," *Journal of Communications*, vol. 5, no. 3, pp. 189–196, Mar. 2010.

[3] Y. Y. Lim, M. Messina, F. Kargl, L. Ganguli, M. Fischer, and T. Tsang, "SNMP Proxy for Wireless Sensor Network," *Fifth International Conference on Information Technology: New Generations (itng 2008)*, pp. 738–743, Apr. 2008.

[4] A. Kassler, A. Pashalidis, K. Doolin, and T. Mota, "Context-Aware Multimedia Services in a Pervasive Environment - The Daidalos Approach," *Proceedings of the First International Conference on Ambient Media and Systems*, pp. 1–5, 2008.

[5] S. Lei, H. Xu, W. Xiaoling, Z. Lin, J. Cho, and S. Lee, "Vip bridge: Integrating several sensor networks into one virtual sensor network," in *ICISP '06: Proceedings of the International Conference on Internet Surveillance and Protection*. Washington, DC, USA: IEEE Computer Society, 2006, p. 2.

[6] N. Gershenfeld, R. Krikorian, and D. Cohen, "The Internet of Things," *Scientific American*, vol. 291, no. 4, pp. 76–81, October 2004.

[7] "ZigBee Specification, Document 053474r17," ZigBee Standards Organization, San Ramon, CA, USA, 2007.

[8] "IEEE Std 802.15.4-2006, IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networksSpecific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," IEEE Computer Society, New York, NY, USA, sep 2006.

[9] J. Postel, "Internet Protocol," RFC 791 (Standard), IETF, Sep. 1981, updated by RFC 1349.

[10] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, IETF, Dec. 1998, updated by RFC 5095.

[11] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), IETF, Sep. 2007.

[12] M. Crawford, "Transmission of IPv6 Packets over Ethernet Networks," RFC 2464, IETF, Dec. 1998.

[13] C. J. B. Carpenter, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels," RFC 2529, IETF, Mar. 1999.

[14] B. Carpenter, "Connection of IPv6 Domains via IPv4 Clouds," RFC 3056, IETF, Feb. 2001.

[15] R. G. E. Nordmark, "Basic Transition Mechanisms for IPv6 Hosts and Routers," RFC 4213 (Proposed Standard), IETF, 2005.

[16] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," RFC 4380, IETF, Feb. 2007.

[17] S. D. A.Conta, "Generic Packet Tunneling in IPv6 Specification," RFC 2473, IETF, Dec. 1998.

[18] "Deployment and Easy Use of wireless Services," 2008. [Online]. Available: http://www.ibbt.be/en/projects/overview-projects/p/detail/deus