ROYAL INSTITUTE OF TECHNOLOGY
Telecommunication Systems Laboratory (TSLab)

AALTO SCHOOL OF SCIENCE AND TECHNOLOGY
Telecommunications Software and Multimedia Laboratory (TML)

Jaime Antonio Jiménez Bolonio

Adapting a DHT (Distributed Hash Table) to a Self-Reliant M2M (Machine-to-Machine) Network

Master's thesis
Helsinki, July 25, 2011

Supervisors:

Prof. Antti Ylä-Jääski
Aalto School of Science and Technology, Finland
Prof. Markus Hidell
Kungliga Tekniska Högskolan (KTH), Sweden

Instructor:

D.Sc. (Tech.) Jani Hautakorpi
Ericsson Research NomadicLab, Finland

KTH - AALTO UNIVERSITIES

ABSTRACT OF THE MASTER'S THESIS

Author: Jaime Antonio Jiménez Bolonio

Name of the thesis: Anpassande av ett DHT (Distributed Hash Table) till ett självständigt M2M (Machine-to-Machine) nätverk.

Date: July 25, 2011 **Number of pages:** 79

Faculty: Faculty of Information and Natural Sciences

Supervisors: Antti Ylä-Jääski
Markus Hidell

Instructor: Jani Hautakorpi

"Machine-to-machine" (M2M) kommunikation är ett forsknings område som förväntas växa inom de närmaste åren. Inom det uppstår nya affärsmöjligheter som är något som t.ex. Ericssons "Future Internet" projekt fokuserar på. Detta innebär att man måste definiera nya protokoll och arkitekturer för att stöda dessa M2M scenarier.

P2P nätverk, och speciellt DHT baserade sådana, är en teknologi som kombinerat med M2M kan resultera i nya sätt att lösa problem inom M2M och dess användningsområde. Detta gäller speciellt inom nät bestående av trådlösa sensorer, Wireless Sensor Networks (WSN). M2M scenarier där sensorerna är mera autonoma och oberoende av centraliserad kontroll kan dra nytta av DHTn.

Det här examens arbetet fokuserar på sätt att använda sig av dagens DHT procedurer inom M2M. Dessutom kommer arbetet att presentera ett sätt att implementera ett M2M kommunikations lager som fungerar över ett existerande DHT. De senaste sensor och P2P teknologierna kommer också att presenteras och analyseras. På basen av detta kommer vi att ge en förklaring för varför ett sådant M2M kommunikations lager behövs och vilka fÃűrdelar det för med sig. På basen av den presenterade lösningen implementeras en fungerande prototyp och några use case definieras.

Till slut presenteras slutsatserna av arbetet och möjliga framtida forskningsmöjligheter och riktningar inom projektet lyfts fram.

Keywords: CoAP, DHT, IoT, M2M, P2P, RELOAD, Zigbee.

KTH - AALTO UNIVERSITIES

ABSTRACT OF THE MASTER'S THESIS

Author: Jaime Antonio Jiménez Bolonio

Name of the thesis: Adapting a DHT (Distributed Hash Table) to a Self-Reliant M2M (Machine-to-Machine) Network

Date: July 25, 2011 **Number of pages:** 79

Faculty: Faculty of Information and Natural Sciences

Supervisors: Antti Ylä-Jääski
Markus Hidell

Instructor: Jani Hautakorpi

Machine-to-machine (M2M) communications is a field of research expected to grow in the following years. New business opportunities arise in this area, for instance the 50 Billion Project and the Future Internet Project at Ericsson. Thus new protocols and architectures need to be defined for the different scenarios where this technology is applicable.

At the same time well known structured P2P networks, for instance by means of a Distributed Hash Table (DHT), present great synergy possibilities with M2M, in particular in the Wireless Sensor Networks (WSN) Area. M2M scenarios in which sensors become more autonomous and self-reliant, independent from a centralized decision-making entity can benefit from the use of DHTs.

This thesis aims at adapting current DHT (Distributed Hash Table) procedures to a M2M (Machine-to-Machine) environment. Moreover it will consist on implementing a layer for M2M communication on top of an existing DHT. We analyze the state of the art in both sensor and P2P technologies. Based on that, we explain the motivations to create such a layer and the its benefits. Following the design we implement a fully working prototype and prepare some use case scenarios. Finally, we draw conclusions from the experience and trace future paths of research for our project.

Keywords: CoAP, DHT, IoT, M2M, P2P, RELOAD, Zigbee.

Acknowledgments

This thesis was conducted at NomadicLab in Ericsson LMF Finland, where I have been working the last couple of years.

I would like to express my gratitude to my supervisors Antti Ylä-Jääski and Markus Hidell for their suggestions and support during the thesis. I would also like to thank my instructor Jani Hautakorpi for his guidance at all stages of my thesis work.

Special thanks to my colleagues at NomadicLab; Daoyuan Li and Nalin Gupta for their friendship, expertise and teamwork during this months and our studies. Thanks too to my manager at LMF Jouni Mäenpää for his continuous work and involvement in the project.

Needless to say, this thesis would not have been possible without the unconditional support of my loved ones.

Helsinki, July 25, 2011

Contents

Abbreviations and Acronyms

(A-G)

3G	Third Generation
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ACK	Acknowledgment
AES	Advanced Encryption Standard
AP	Admitting Peer
APDU	Application layer Protocol Data Unit
API	Application Programming Interface
APL	Application Layer
APO	Application Object
APS	Application Support Sub-layer
APSDE	Application Support Sub-layer Data service Entity
ASP	Address Settlement by Peer-to-Peer
BP	Bootstrap Peer
CAN	Content Addressable Network
CoAP	Constrained Application Protocol
CoRE	Constrained RESTful Environments working group
DIEM	Devices and Interoperability EcosysteM
DDNS	Distributed Domain Name Service
DNS	Domain Name Service
DNS-SD	Domain Name Service Service Discovery
DTLS	Datagram Transport Layer Security
DoS	Denial of Service
EPC	Electronic Product Code
EEPROM	Electrically Erasable ProgrammableRead-Only Memory
DHT	Distributed Hash Table
FFD	Full-Function Device
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications

(H-S)

HR-WPAN	High-Rate Wireless Personal Network
HTTP	Hypertext Transfer Protocol
ICE	Interactive Connectivity Establishment
ICMPv6	Internet Control Message Protocol version 6
ICT	Information and Communication Technologies
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol version 6
IPC	Inter Process Communication
LN	Local Node
LoWPAN	Low-power Wireless Personal Network
LR-WPAN	Low-Rate Wireless Personal Network
M2M	Machine-to-machine
M2MCE	Machine-to-machine Communication Enabler
MAC	Medium Access Control
MIC	Message Integrity Code
MCN	Monitoring and Controlling Node
MR-WPAN	Medium-Rate Wireless Personal Network
NAT	Network Address Translation
NWK	Network Layer
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
P2PP	Peer-to-Peer Protocol
P2PSIP	Peer-to-Peer Session Initiation Protocol
PHY	Physical Layer
PN	Proxy Node
PDU	Protocol Data Unit
RELOAD	REsource LOcation And Discovery
REST	Representational State Transfer
RFC	Request For Comments
RFD	Reduced-Function Device
RFID	Radio-Frequency IDentification
RMI	Remote Method Invocation
SEP	Service Extensible P2P Protocol
SHA	Secure Hash Algorithm
SIP	Session Initiation Protocol
SNMP	Simple Network Management Protocol
SRAMS	tatic Random-Access Memory

(T-Z)

TCP	Transmission Control Protocol
TEKES	Teknologian ja Innovaatioiden Kehittamiskeskus
TLS	Transport Layer Security
TURN	Traversal Using Relays around NAT
UA	User Agent
UDP	User Datagram Protocol
URI	Universal Resource Identifier
USB	Universal Serial Bus
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
VLS	Variable Speed Limit
WAN	Wide Area Network
WN	Wide Area Node
WSN	Wireless Sensor Network
WWAN	Wide Wireless Area Network
XPP	Extensible Peer Protocol
ZDO	ZigBee Device Object

List of Figures

Chapter 1

Introduction

1.1 Overview

This thesis is part of a program to adapt current DHT (Distributed Hash Table) procedures to a M2M (Machine-to-Machine) environment. The program participates in Ericsson's 50 billion M2M vision [29]. This work was a part of the Devices and Interoperability EcosysteM (DIEM) program [1] and is sponsored by the Finnish Funding Agency for Technology and Innovation (TEKES) [2] as well as other industrial and research partners. The project is done in collaboration with the ICT Future Internet SHOK program [3] that aims to: *"Bring together the key research resources to develop future Internet networking technologies and to create new global ICT based business ecosystems."*

This project began by evaluating the different use cases that the M2M technologies will have to face. Although most of the scenarios are related to the Ericsson 50 billion vision, only the ones more likely to benefit from a distributed approach were chosen. Then the most suitable hardware was chosen, following the principles of small size, low power consumption and enabled connectivity. From the functionality perspective, the M2M layer was designed by first determining the functions needed for all possible types of cases, making it generic enough to be adaptable to different use cases.

In this chapter we will introduce the problem this thesis tries to solve. We will also present the general structure of the thesis.

[1] http://www.diem.fi/programme
[2] http://www.tekes.fi/en/
[3] www.futureinternet.fi

1.2 Problem Statement

Current Machine-to-Machine networks are often organized in a hierarchical or mesh topology (see Figure 1.1) communicating with low power consuming protocols like Zigbee (see Section 2.2.2). In this mesh there is a central coordination gateway that gathers the data from the densely deployed sensors and analyzes it a posteriori [9]. This kind of network is very well suited for data aggregation and for data analysis. Sensors can operate and retrieve data during long periods of time, weeks, months or even more than a year, due to their extremely low battery consumption. Some of these sensors operate indefinitely if they are connected to a, often renewable, power source. These sensors need human interaction when it comes to actuating over their environment, since the flow of information usually goes from the sensors to the human but not vice-versa.

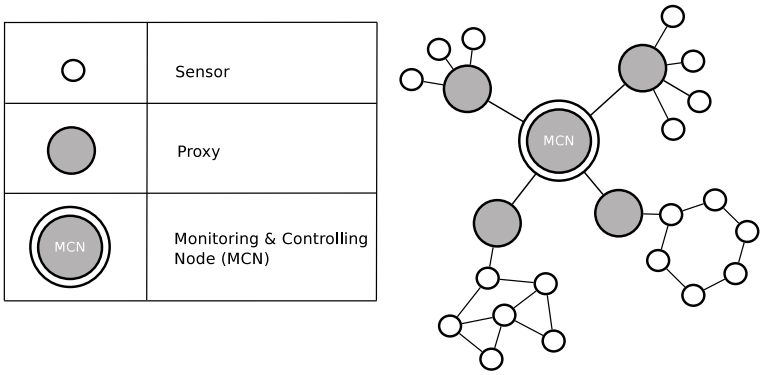

Figure 1.1: Example of Traditional Wireless Sensor Network.

In this thesis a different approach is proposed. This approach implies a different network topology (see Figure 1.2) that relies on a Distributed Hash Table (DHT) (see Section 2.4.3) to organize the nodes. These nodes are not just sensors but also actuators, they share the information about the environment where they have been placed with a Wide Wireless Area Network (WWAN) in order to make independent decisions and actuate over that environment. Therefore these nodes or proxies are both sensors and actuators. This Peer-to-Peer (P2P) network (see Section 2.4) of proxies can be complemented by other traditional Wireless Sensor Networks (WSN) that will feed data to the nodes of the DHT. Note that the Proxy Nodes (PN) and Wide Area Nodes (WN) can act as sensor, actuator or both.

Providing that a suitable DHT is found, there is a necessity for a Machine-to-machine "layer" on top of existing DHT. Such a layer should provide the

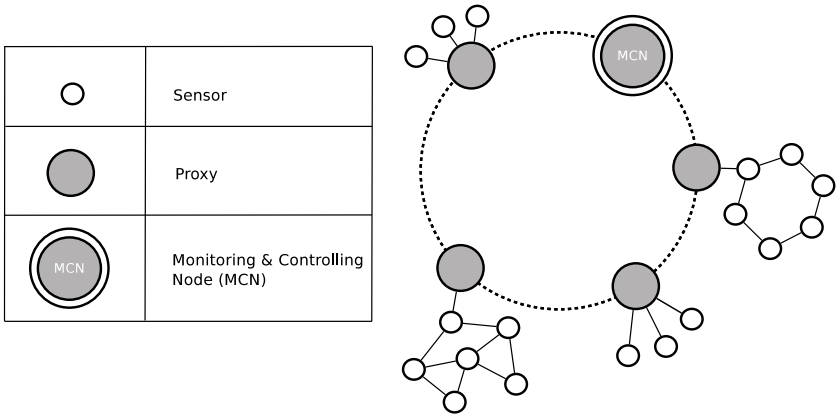

Figure 1.2: Example of Distributed M2M Sensor Network.

functions needed for all possible types of messaging in the M2M network depending on the use case.

Possible use cases involve large area networks, in the range of kilometers, in which there is a large amount of nodes and the decisions to be made by these nodes are usually not extremely complex but depend on the information provided by rest of the nodes.

1.3 Structure of the Thesis

Chapter 2 explains different technologies related to this thesis.

Starting from concepts from The Internet of Things (Section 2.1), M2M (Section 2.1.1) and Current Sensor Technologies (Section 2.2). Some of these technologies being the IEEE 802.15.4 standard (Section 2.2.1), Zigbee (Section 2.2.2), 6LoWPAN (Section 2.2.3)and CoAP (Section 2.2.4). The next section deals with network monitoring and SNMP 2.3.

The next Section 2.4 treats different P2P technologies. It continues by giving an overview on the use of Distributed Hash Tables (Section 2.4.3), with emphasis in the Chord protocol (Section 2.4.3) and RELOAD (Section 2.4.4).

Chapter 3 describes the different guidelines followed in order to design the architecture of the M2M layer as well as different use cases in which the M2M network could be deployed. In Section 3.3 we draw the main architecture. Section 3.1 sets the main principles and in Section 3.2 the principles followed in the design. Section 3.4 explains in detail the CoAP Communication, joining and leaving procedures of the different nodes of the network, the CoAP

Name Registration Service and the Bookkeeping mechanism. Section 3.4.6 addresses some of the security concerns for WPAN, WWAN, CoAP, SNMP, Name Service, the DHT and Bookkeeping. Finally in Section 3.5, we deal with two proposed scenarios for this new technology: Dynamic traffic Signaling and Water System Automation.

Chapter 4 describes the hardware and software used for the implementation of the M2M layer. Section 4.2 shows the architecture for our prototype.

Chapter 5 sets the different conclusions that can be extracted from the thesis and a summary of what it was achieved. It also sets some lines for future work and improvements.

Chapter 2

Background

2.1 The Internet of Things

The European Commission [30] defines the Internet of Things as "things having identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environmental, and user contexts". The book "Interconnecting Smart Objects with IP - The Next Internet" [89] defines IoT as "a loosely coupled, decentralized system of smart autonomous objects augmented with sensing, processing, and network capabilities".

The phrase Internet of Things (IoT) is often used when representing the concept of global network of connected devices. It is inspired by the extensive use of Radio Frequency Identification (RFID) tags in transportation and logistics. RFID chips are small, uniquely addressable chips used to identify objects. They are as small as 0.3 millimeters and the cost of a passive RFID tag is less than 15 cents of an Euro [63].

IoT is the logical evolution of RFID tagging. IoT shares similarities with RFID since it will also involves very small devices embedded into other objects, it is low-cost and ubiquitous. Both are uniquely addressable, RFID with the Electronic Product Code (EPC) and IoT most likely with Internet Protocol v6 (IPv6) and both can interact with other devices like mobile phones [81].

But IoT implies more than just addressing and retrieving data, IoT implies that the devices are smart. The IoT vision foresees an enhancement of current devices - sensors and actuators for instance - with connectivity to other available networks in order to interact with the world [71]. Smart objects can sense and process data, cooperate, make informed interpretations of their surrounding environment [49]. Later on this data can be used for data mining purposes

so that the users or other devices can undertake the appropriate actions upon the environment (adaptation) or themselves (self-configuration), all this done via Internet.

In the case of IoT the types of device can be anything, whether RFID tags, sensors, mobile phones, personal computers, portable game consoles, television sets, cameras, home appliances and many others not envisaged yet. If the types of devices are many, the amount of them can be staggering, some estimates at Ericsson suggest that more than 50 billion devices will be connected by 2020 [29]. The number of devices will outnumber the number of personal computers and servers available in today's Internet. The approximation is taken from the assumption that each telephony subscriber will have from 1 to 5 devices, each vehicle too, different utilities (electricity, water and gas) would be automated, and so on.

As seen in Figure 2.1, the IoT lies as a consequence of the growth of Wireless Sensor Networks, Embedded Systems, Mobile computing, Mobile Telephony and other computing and telephony areas that have thrived in recent times [89].

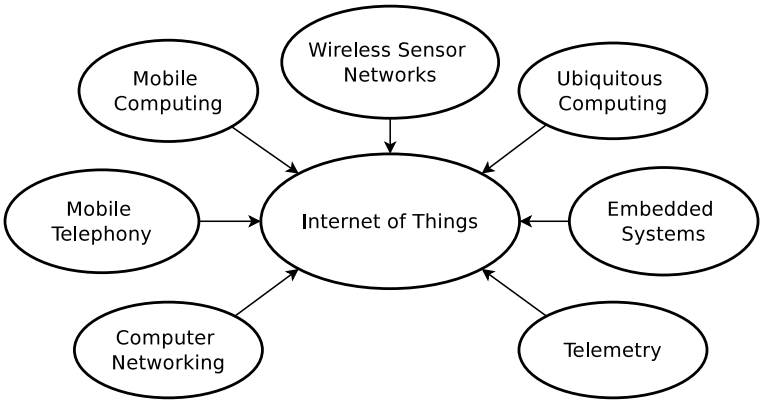

Figure 2.1: Smart objects are at the intersection of current technologies.

Examples of this future scenario is that doors could be programmed to send a signal when they are opened after some time and connect to a home automation system. A person's keys not only would be traceable [91] but they would themselves inform of their location if they are lost. A patients recovery process could be monitored and modified to his needs during the process. Buildings and objects (furniture, appliances, vehicles,...) would adapt to each persons characteristics, remember them, and share them with other objects depending on context. In the industry, water systems could be fully automated and dynamically leak-proofed. So would be distribution chains, with a full tracking

of every single object from manufacture to delivery and even after the delivery. All these are examples of the radical impact that this kind of connectivity would have.

To make these examples of the IoT become a reality, there are some factors spotted in [41] and [11] that have to be addressed. Energy consumption needs to be improved, therefore it is necessary to find better embedded power sources and energy harvesting systems as well as to minimize the use of battery power. It is important to consider the cost of each device, since small cost savings per device have great impact overall. The size of these devices needs to decrease, therefore better miniaturization and space optimization is required. Global standardization is fundamental since we want these devices to interoperate, specially since they will be produced by many different parties.

In order to make this scenario possible, machine-to-machine technologies need to be studied and developed.

2.1.1 Machine-to-Machine

We will see that IoT and M2M are in many ways interwoven concepts. If IoT comprehends a future scenario with billions of interconnected things, Machine-to-Machine (M2M) is one of the enabling technologies to get to that world view. Taking the M2M definition in [27] an example of M2M network would be home automation.

In home automation, a sensor captures the temperature of the room, it sends the information to another node connected to an actuator. Depending on the information, that event can trigger an action that is carried out by the device itself or in coordination with other devices of the home network. All this is done without the need of human intervention.

Until now the Internet has revolved around human activities and human interactions, but it is likely that in the future most of the active participants in these interactions will be things instead. M2M communications have already existed for some time, either in the form of classical client-server technologies or as a byproduct of other human interactions with computers or other machines. Still, M2M as a defined technology is blossoming and it is mostly in smart metering systems, fleet and asset management and telemetry.

Constrained Devices

Sensor devices are indeed constrained. They carry small packets due to the limitations at the physical layer, which allows for a maximum of 127 bytes.

If they follow current IEEE standards for constrained devices, they will have bandwidth with data rates of 250 kpbs at maximum for 2.4 GHz transmission. Their minimum bandwidth will be 20 kbps for 868 MHz. They might be movable and are usually deployed in ad hoc fashion since their location is typically not predefined.

Due to the previous reasons, devices will often offer unreliable connectivity. Also, devices will usually be in sleep mode in order to prevent battery drain, and will not communicate during those periods.

These physical limitations will deeply affect the technologies that can be developed on them. Those technologies are studied in the following chapter.

2.2 Current Sensor Technologies

The following sections will deal with current sensor-related technologies. Arguably, once a constrained environment was imposed, it provoked the creation of adequate link layer protocols, for instance 802.15.4. Then different companies allied to develop on top of it a protocol stack, that is Zigbee. Current efforts are placed towards the standardization of these technologies. The IETF developed 6LoWPAN to align it with IPv6 and now CoAP is being standardized as Application protocol aligned with HTTP.

2.2.1 IEEE 802.15.4

One of the approaches for the use of sensors, whether on home automation, environmental and industrial monitoring or in any other scenario is that they will be battery powered, they will have low throughput and low data rate, in the order of few kbps. Therefore technologies like traditional Wireless Area Network (WLAN) or Bluetooth are inadequate since the first is designed to be used for traditional file transfer and multimedia applications and the second for moderately high data rates. Moreover, they are not very energy efficient. MAC protocols used by those technologies, IEEE 802.11 [4] and IEEE 802.15.1 [6] are thus also inadequate since they were designed for relatively high-end devices able to handle seamless roaming, message forwarding, and a data throughput of 2-11 Mbps, not for constrained sensors. This necessity for communication between sensors with constrained capabilities drove the creation of a Media Access Control (MAC) standard that focus on low-power, short-range and low-cost wireless communications, the IEEE 802.15.4 in 2003 and its revision in 2006 [5, 7].

Within the standardization body tasked with the creation of 802.15.4, the

IEEE 802.15 working group [28], there are seven Task Groups (TG) with various focuses. For instance the IEEE 802.15.3 for High Rate WPAN (HR-WPAN), the IEEE 802.15.1 for Medium Rate WPAN (MR-WPAN) and the IEEE 802.15.4 for Low Rate WPAN (LR-WPAN)[1]. The main design characteristics of 802.15.4 as defined by [5] are simplicity to install, reliability, short-range, low-cost and long battery life. All this while maintaining a simple and flexible protocol.

The standard specifies two possible types of device participating in a network: a full-function device (FFD) and a reduced-function device (RFD). The network must include at least one FFD, that will operate as a coordinator of the Personal Area Network (PAN). This is necessary since a FFD has a higher load, being able to communicate to both FFD and RFD. The RFD - usually battery powered - can only communicate to an FFD.

This holds for the two possible topologies that are defined: star and peer-to-peer. Star topology places the FFD as the center of the communication flow and as a sink for RFD devices. In the peer-to-peer topology, several FFDs form a mesh, the RFDs are connected to the mesh via an FFD PAN coordinator. It is the latter topology that applies for our project since message routing and peer-to-peer networking are implemented on higher layers.

The IEEE 802.15.4 follows the Open Systems Interconnection (OSI) seven-layer model, defining communication at the physical and data layers. In Figure 2.2 we can see the whole ZigBee stack of which the two first layers are the MAC and Physical layers defined by the IEEE 802.15.4. Without going into specifics of the standard itself, since they are out of the scope of this thesis, it is relevant to know that it provides 20-250 kbit/s data rate [5, 79] and it operates in the 868 MHz frequency band, its maximum conductive power is of in 25 mW [2] link-layer security is provided with 128-bit Advanced Encryption Standard (AES) encryption.

Addressing can be done using 64 or 16 bits. This distinction depends on the device within each WPAN, communications will probably use short identifiers since the amount of nodes (sensors) per WPAN will not be more than 2^{16}. Data transport is achieved by using frames as basic unit, these can be data, ACK, beacon and MAC command frames. The standard also specifies a superframe defined by the coordinating node. When the superframe is used, two beacon frames act as its limits and provide synchronization to other devices as well as configuration information. IEEE 802.15.4 has been shown to be efficient when it comes to energy saving [51] at the cost of higher latency and lower

[1]More Information of these standardization efforts is available at IEEE 802.15 working group site: http://ieee802.org/15/

[2]Note that the frequency and maximum conductive power values apply for Europe only.

bandwidth. In the case of battery powered sensors this is not too high a cost.

Two limitations of 802.15.4 are that, although it enables the possibility of two different topologies, it does not provide multi-hop networking nor mesh networking. Therefore it is the basis upon which other protocols have been built, among others ZigBee™.

2.2.2 ZigBee™

To make a functional WPAN node, it is necessary to build the remaining layers on top of the physical and data layers that are covered by IEEE 802.15.4. ZigBee™ is a specification made by the ZigBee Alliance to build such a protocol stack based on the IEEE 802.15.4, it defines the network layer and provides a framework for application programming in the application layer. Since the ZigBee protocol is availed by many different partners [3] it focuses on defining a general-purpose protocol that can be used either in industrial control, home automation, medical data mining or any kind of sensing devices with similar requirements.

The ZigBee protocol is well-known in the wireless sensors world [10, 31, 87] and it can be considered a de facto standard until other standards and applications such as 6LoWPAN are further developed [21]. As it was mentioned in 2.2.1, being build on top of IEEE 802.15.4 [5], ZigBee fulfills the requirements of low power consumption, low throughput and low data rate. It can also accommodate as many as 254 nodes [28, 67], while alternatives like WLAN and Bluetooth can only have a network size of 32 and 7.

ZigBee (see Figure 2.2) provides two main new layers, the ZigBee™ Network Layer (NWK) and a framework for the Application Layer (APL) [67]. The application layer framework consists of the Application Support Sub-layer (APS), responsible for providing a data service to applications and device profiles, and the ZigBee™ Device Object (ZDO), tasked with defining the role of a device and initiating or responding to discovery requests. The ZDO also communicates with the ZDO Management Plane to deal with application requests for network access and security.Manufacturers will in turn use the Application Framework to create their own Application Objects (APO), up to 240 different ones. An APO represents different application attributes or profiles that can be defined on a single Zigbee device. A profile is a set of common messages or actions that enable interoperability between ZigBee nodes. Some example of profiles are those for heating, ventilation, lighting or industrial process control.

[3] Some of the promoters are Texas Instruments, Philips and STMicroelectronics among hundreds of participants and adopters such as Cisco, Samsung, Huawei, LG, etc. More at http://www.zigbee.org

For instance a temperature sensor on one ZigBee node can communicate with a heating device in another node, and form a heating application profile. Being dependent on the vendor these profiles might not be open like the rest of the ZigBee standard. To interact between APOs, ZigBee allows forming clusters that group related attributes.

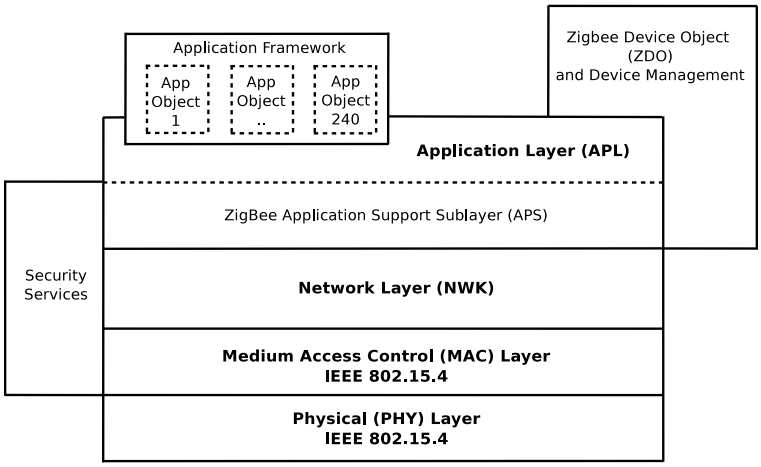

Figure 2.2: Outline of the ZigBee Stack Architecture.

The *NWK layer* handles network address and routing and provides support for initial device configuration, network bootstrapping, neighbor discovery, route discovery and receiver activation. This layer defines three device roles:

1. A Zigbee end device that corresponds to an RFD or FFD in IEEE 802.15.4 (see Section 2.2.1) in the role of a simple device that only can join and leave a network.

2. A ZigBee router that corresponds to a FFD with routing capabilities.

3. A ZigBee coordinator that corresponds to an FFD tasked with the management of the whole network and is the only device capable of creating a new network.

Both ZigBee coordinators and routers shall also participate in the assignment of logical network addresses and maintain a list of neighboring devices.

The *APL layer* contains the previously mentioned APS, ZDO and the Application Framework containing manufacturer defined APOs. The APS provides an interfaces between the NWK and the APL layers, which details are out of the scope of this thesis. Nevertheless it is relevant to know that it creates

the APSDE-SAP that provides a data service allowing the ZigBee device to execute standard network functions such as request, confirm, response and indication primitives for data transfer.

2.2.3 6LoWPAN

Although not used in our prototype, since there are no capable devices in the market (See Section 4.1). It is important to refer to the current standardization efforts made by the Internet Engineering Task Force (IETF) on 6LoWPAN. 6LoWPAN, as defined in [79] enables the use of IPv6 on constrained devices in an efficient manner. It does so by providing an adaptation layer and optimizing protocols related to IPv6.

6LoWPAN follows the trend that low-power devices and low-power technologies are setting by enabling IPv6 addressing capabilities for the IoT [4]. At lower layers wireless technologies have adapted to provide low-power radios with limited frame size and data rates such the ones used by 802.15.4 (see Section 2.2.1), it is therefore the logical next step to provide adaptation at higher layers too.

The main differences between 6LoWPAN and ZigBee is that ZigBee specifies a whole protocol stack up to the Application Layer while 6LoWPAN "only" aims at adapting IPv6 to restricted environments. In that sense, ZigBee does not yet integrate with Internet applications and is very close to vendor specific configurations. 6LoWPAN instead serves as a reference for future work and for future applications that search for high scalability and communication with Internet-based devices, regardless of vendor or of upper-layer protocol. In fact, the ZigBee Alliance announced that it would start integrating 6LoWPAN (i.e. IPv6) within its own ZigBee protocol stack [79] and there is previous work that shows how ZigBee application profiles can be carried over 6LoWPAN [88].

As it is shown in figure 2.3, 6LoWPAN protocol stack is extremely similar to the IP protocol stack. One of the main differences is that 6LoWPAN stack is more constrained, 6LoWPAN has been defined to use IPv6 as network protocol and is optimized for 802.15.4 as link protocol. Issues regarding error checking, reliability and link layer compatibility are further explained in [5, 79] and out of the scope of this thesis. Nevertheless it is relevant to know that 6LoWPAN's most basic requirements are framing and unicast transmission and addressing, requirements covered by the current 802.15.4 standard. In the transport layer, User Datagram Protocol (UDP) is the protocol that better adjusts for constrained environments,while using Transmission Control Protocol (TCP)

[4] A 128 bit address spaces allows for 340282366920938463463374607431768211456 addresses

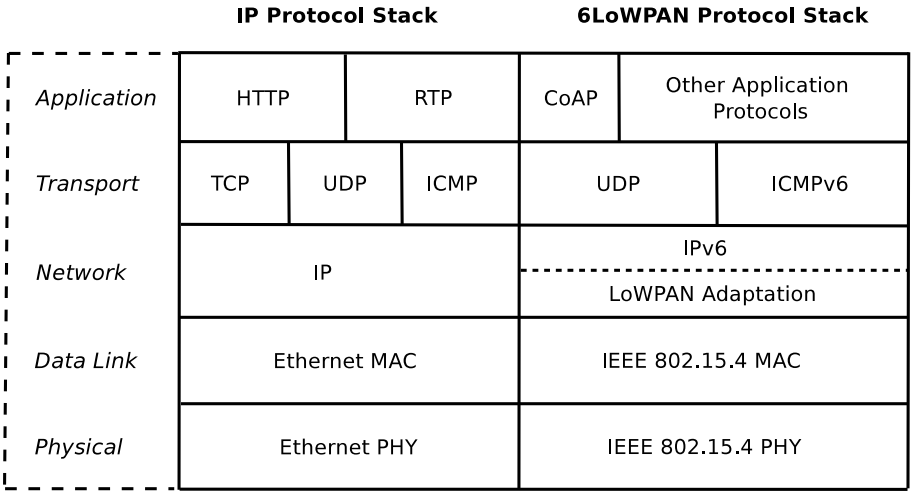

Figure 2.3: Outline of the IP and 6LoWPAN protocol stacks.

would cause low performance and would increase the complexity of the protocol [79]. ICMPv6 is used in the same way as in IP, for control messaging and network reachability and discovery. Transformation between full IPv6 and the LoWPAN adaptation is done in routers at the edge of the WPAN islands (edge routers) that have both Ethernet and 802.15.4 below IPv6, this implies that they are likely to be a full-functioning 802.15.4 device (FFD). This transformation obviously has to be transparent for the nodes in the network and for the IPv6 stack itself. The core of 6LoWPAN is explained in two RFCs of the IETF [53, 64].

2.2.4 CoAP

Nowadays, the use of web services on the Internet is common in most applications. Web services services depend on the basic Representational State Transfer (REST) architecture [33]. This architecture is well suited for M2M, therefore some standardization efforts have started.

The IETF Constrained RESTful Environments (CoRE) working group has as its main task to employ this REST architecture in constrained environments. Their proposed standard is Constrained Application Protocol (CoAP) [78].

CoAP is not just a compression of Hypertext Transfer Protocol (HTTP) [32]. It has a subset of the HTTP functionalities that have been modified to make the protocol suitable to IoT and M2M applications. Since it is fully compatible with HTTP, CoAP can be used not only between nodes on the same WSN but

also between constrained nodes and nodes in the Internet. Some new functionalities such as multicast, asynchronous communication and subscriptions have been included.

CoAP is built on top of UDP and therefore has significantly lower overhead and multicast support. This overhead reduction has a strong impact on the battery life, drastically reducing sensor's power consumption [18].

On top of UDP there are two layers that conform CoAP: Transaction and Request/Response Layer.

The Transaction layer is used for communication between endpoints. It has 4 types of messages: Acknowledgment, Reset, Confirmable and Non-Confirmable. If the message has to be acknowledged, then it is Confirmable. If it does not, then it is Non-confirmable. Reset indicates that a Confirmable message was received but cannot be processed. Confirmable messages are retransmitted after a default timeout until the recipient acknowledges the message. The transaction layer also provides support for multicast and congestion control [26]

The Request/Response layer is responsible for the transmission of requests and responses for resource manipulation and transmission. The REST request is piggybacked on the Confirmable or Non-confirmable message and the REST response on the Acknowledgment message.

CoAP is intended to be used in WSN, in these type of networks resources likely change over time. CoAP therefore allows a client to constantly observe the resources by means of observations: a client can register to a resource by using a modified GET request sent to the server. When the value of the resource changes and the sensor finds it relevant, it informs the server that notifies each client having an observation relationship with the resource. The duration of the observation relationship is negotiated during the registration procedure [40].

With CoAP we have seen most of the state of the art protocols regarding constrained devices. In our network we expect those devices to be managed by one single entity. It will be necessary to develop a protocol to manage those devices. Such a protocol will be inspired in SNMP. We will see more about SNMP in the next section.

2.3 SNMP

The Simple Network Management Protocol (SNMP) is the most widely-used network management tool for TCP/IP based networks. There are three ver-

sions, SNMPv1 in 1990 [16], SNMPv2 including applications [55] and SNMPv3. Most of the specification was done in the first two, SNMPv3 mainly includes new security and administration functions [84].

SNMP defines a protocol for the exchange of management information. It represents management information by dividing nodes into two types: agents and managers. Any node in a network includes one of these alternatives.

An Agent is responsible for collecting and maintaining information about its local environment, providing that information to a manager, either in response to a request or in an unsolicited fashion when something noteworthy happens and responding to manager commands to alter the local configuration or operating parameters. In the context of M2M, most nodes that gather sensor information would be agents.

In the network we would have one or more managers. Managers usually provide some interface towards the human user, often this interface is web. The interface is needed so that a human network manager can monitor and control the different nodes of the network. The control comes by the issuing of commands towards the nodes. In the context of M2M this commands will often be associations between nodes and thresholds for sensors.

In order to control the nodes, SNMP provides four basic types of functions: Get, used by a manager to retrieve an item from an agent; Set, used by a manager to set a value in an agent; Trap, used by an agent to send an alert to a manager; Inform, used by a manager to send an alert to another manager.

Those four functions are implemented in seven SNMP PDU (Protocol Data Unit), used for communication between the SNMP entities:

1. GetRequest. A Manager-to-agent request to retrieve the value of a variable stored by the agent.

2. SetRequest. A Manager-to-agent request to change the value of a variable or list of variables.

3. GetNextRequest. A Manager-to-agent request to discover available variables and their values.

4. GetBulkRequest. A Manager-to-agent request for multiple iterations of GetNextRequest.

5. Response. An Agent-to-manager response that returns variable bindings and acknowledgements for GetRequest, SetRequest, GetNextRequest, GetBulkRequest and InformRequest.

6. Trap. An Agent-to-manager asynchronous notification.

7. InformRequest. A Manager-to-manager acknowledged asynchronous notification.

Current standardization work is being done in adapting SNMP to a distributed environment. For instance, by using SNMP with RELOAD for network management purposes [68, 69]. We will see more about distributed networks and P2P in the following section.

2.4 Peer-to-Peer (P2P)

In traditional client/server architecture, clients request information from a single server entity, this entity in turn provides the information to one or several clients. The typical characteristics of the client are that it initiates requests and waits for replies and that it does not connect to a large number of servers at once. On the other hand, the server entity can comprise one or many servers that serve the same purpose or task and it never initiates the communication. The server only waits to receive a request and then replies to the client. In this model, the information concentrates on the server side instead of the clients. If the server stops working, then the whole network loses its source of information and therefore becomes useless. The need to overcome this weakness lead to the creation of new ways to distribute the information without entirely depending on one single entity, P2P is one of them.

Peer-to-Peer (P2P) architectures became popular in the late nineties, introducing significant advantages over Client-Server models. In a pure Peer-to-Peer (P2P) architecture each of the machines (nodes) operates at the same time as both client and server, having the same level of responsibilities as the rest of the nodes. The definition of P2P is the *"shared provision of distributed resources and services"* [77]. According to [77] P2P main features are:

1. Sharing of distributed resources and services, i.e. each peer acts as both client and server.

2. Decentralization, although depending on the P2P topology there might be a central entity for indexing or for security (i.g providing the security certificates), in P2P networks generally every node is equivalent to any other and there is no node that knows the whole network topology. Nevertheless between fully centralized and fully decentralized structures, there is a range of different structures, which boundaries are not clearly delimited.

3. Autonomy, nodes can decide what to share with the rest of the P2P network.

The rise of public cloud computing services [54] and other similar technologies are steadily concentrating the information and the control on the server side, which is now named cloud. But it could be argued that often in those cases where the overall functionality is that of a server-client model, the server side is a network arranged following an architecture similar to that of P2P (see Figure 2.4.1).

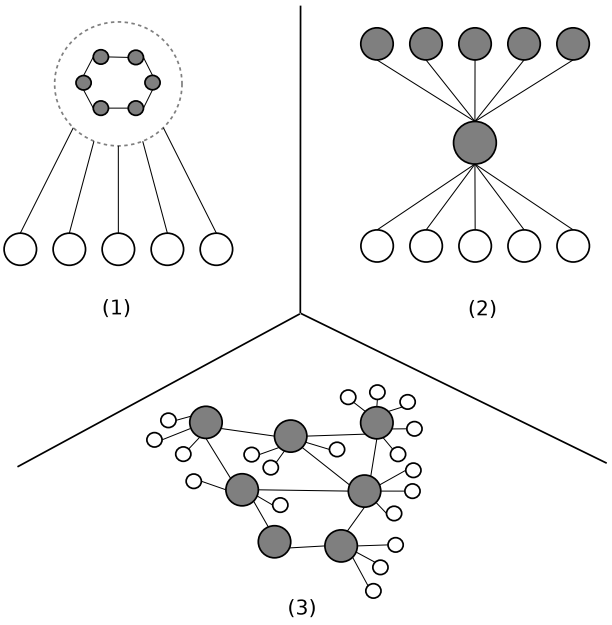

Figure 2.4: Some composed network topologies: (1) Centralized/Ring, (2) Centralized/Centralized, (3) Centralized/Decentralized.

Nevertheless P2P has followed an evolution and currently there are not many examples of P2P-only or Client-Server-only topologies. Indeed, depending on how we want to organize the information we will have different topologies that adapt more efficiently to the different scenarios [52]. For example the *Centralized/Ring Topology* is used in robust web servers (see Figure 2.4) combining the simplicity of a Client-Server system with the robustness of a ring. *Centralized/centralized* topologies are also used in some web applications, when there is the need for a central control but the features of a central server are distributed among other machines. This is the case of Google as for instance, the web server contacts a distributed database of links. *Centralized/decentralized* topologies are used to enable fault tolerance and centralization at the

same time. Often used in the advanced hybrid unstructured P2P systems like Gnutella 0.6, Spotify or Skype [38]. P2P can also be classified in structured and unstructured according to the way the nodes are organized.

2.4.1 Unstructured P2P

An unstructured P2P network is one in which there is no clear definition of what the topology of the network should be. Since there is no definition of the location of the nodes, the searching process sometimes consumes a lot of bandwidth and processing power. These type of networks were used in the first P2P applications. According to their degree of centralization, they can be centralized, pure and hybrid P2P (see Figure 2.5).

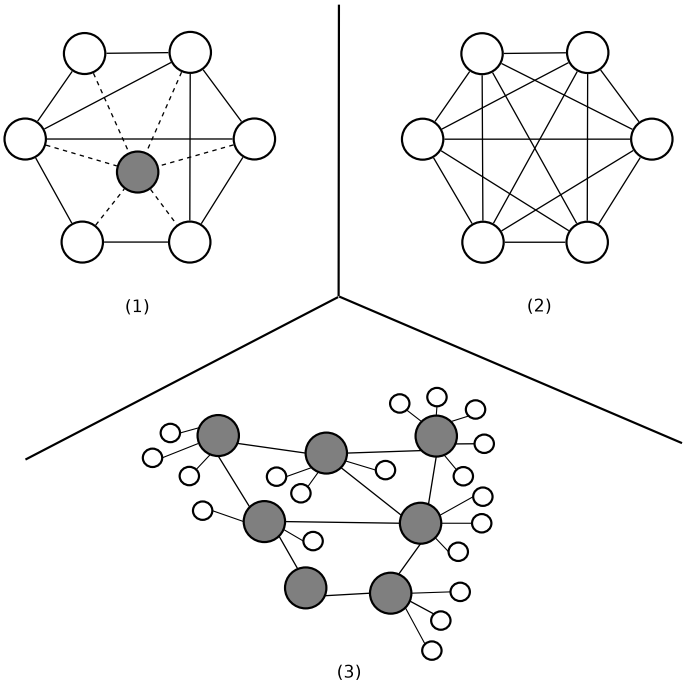

Figure 2.5: Different types of P2P: (1) Centralized, (2) Pure and (3) Hybrid.

Centralized P2P

In the early stages of P2P simplicity was key. This and the ability to control motivated the creation of simple centralized schemes that contained all features of P2P systems but that required a central entity to provide indexing or security. A typical example of this type of P2P is the original Napster which

used a central server for the lookup of content in the P2P network. This architecture provided low message overhead $O(1)$ and 100% success rate in case the content is on the network. Some disadvantages of this networks were that they were not very scalable nor robust, since by taking out the central server the network would become unusable, as it happened.

Pure P2P

Pure P2P networks solved the dependency on a central entity, in them any node could be removed without loss of functionality in the network. These networks are fully decentralized and very robust. Typical examples of pure P2P networks are Freenet and Gnutella 0.4. However, these networks have excessive communication overhead $O(N^2)$ and connections between peers since requests are flooded in the network. Neighbors recursively forward requests in order to (eventually) find a node with the content that was looked up. Taking as an example a Freenet network, if a peer needs to find one particular resource, it has to send a request to one or multiple adjacent nodes which will forward the request to their adjacent peers and so on. This can cause many security-related problems since an exponential request attack can be easily carried out. Therefore, each message carries a time-to-live (TTL) timestamp, which forces the elimination of each request after a certain number of hops. Moreover, this architecture does not guarantee that the search will be successful at all.

Hybrid P2P

Second generation unstructured P2P networks took core concepts of both centralized and pure P2P. Hybrid P2P aims at tackling the problem of flooding by having some peers manage more signaling traffic. These new peers are called super-peers, the rest are leaf peers. Super-peers are usually elected depending on speed, network congestion and other parameters among the leaf peers. For instance Gnutella 0.6 uses this system to provide better search. All the same, even in hybrid networks, where there is no single point of failure, there is still high signaling traffic because of decentralization.

2.4.2 Structured P2P

Structured P2P networks and DHT-based ones in particular, aim at diminishing signaling traffic - a problem that is endemic in unstructured P2P. The solution it presents is to order the nodes in the overlay. In DHT-Based P2P, like in Chord [86, 3] or CAN [2], peers are organized and indexed, position is

predetermined by an algorithm, thus each peer can infer where other peers are located. In order to provide lookup, Distributed Hash Tables (DHTs) are used. DHTs used in P2P assign an unique ID when a peer joins the P2P overlay and another ID or key to the data it wants to store. The key indicates the position where the data is stored, and it is created by means of a hashing function.

When a peer wants to retrieve some information it will perform the same basic function, mapping the requested content into a value and consult its hash table. Chord for instance maps content into a linear space, others map it into different forms. A hash table contains a range of keys that indicate which nodes in the overlay are responsible for the queried data. If the node does not find the responsible node among the peers, the request will be forwarded throughout the overlay. The way it is forwarded is determined by the specific overlay algorithm of the network. The three steps common to most DHTs are:

1. Mapping content or node into the overlay space by means of a hash function.
2. Routing to a (Key Value) pair, by starting the lookup at an arbitrary node of the DHT and forwarding it until finding the right peer.
3. Making a direct connection and retrieving the content.

DHT-Based P2P networks provide the same robustness and scalability as other P2P networks. Additionally, they have the advantage of less communication overhead $O(logN)$ and no false positives when doing a search. However, these networks lose the possibility of doing wild-card searches for content in the P2P overlay network.

The topics of the Distributed Hash Tables and Chord are more thoroughly explained in the next section 2.4.3.

2.4.3 Distributed Hash Tables (DHT)

As it was previously stated, structured P2P networks offer high scalability and reliability. One of the problems though, is that they have high latency for content lookup, especially when the network is large. In those cases it is becoming customary to adopt a Distributed Hash Table (DHT) model in order to organize information for better lookup and less traffic overhead $O(logN)$.

A hash function or hash algorithm is a mathematical function that transforms a large amount of data into a small value, which usually is also encrypted. These functions are used on hash tables or hash maps, which are structures

made for referencing, as they associate the hashes or keys with real values. Hash tables have various purposes; one of them is indexing, keeping an array of *(key, value)* data types. They are useful for lookup of information since the indexing makes the process very efficient.

On P2P networks, the information is stored at the endpoints of the network (i.e. peers) rather than on a single or various servers. The hash table approach is not possible here since the data has to be distributed more or less equally among the peers. Instead, Distributed Hash Tables (DHTs) are used. They are created by storing the contents of the hash table across a set of peers on a P2P network according to certain algorithms to ensure optimal distribution. Each of the peers will be responsible for part of the key space.

Chord

One of the best known DHTs is Chord, a fully distributed peer-to-peer lookup algorithm [86]. Chord has just one operation: *providing that you give a key, Chord can map it onto a node.* Chord uses a distributed hash table for routing. If we have N nodes and K keys, each node is responsible at most for $(1 + O(logN))K/N$ keys. It was demonstrated in [86] that Chord can solve any given lookup by sending information to a maximum of $O(log\ N)$ nodes. This means that even when N is a large number, the number of messages sent by a node using Chord will still be relatively small.

The nodes are arranged on a ring. On the Chord ring each node has a successor and a predecessor. The predecessor of a node is the peer in front of it when traversing the ring clockwise. In the same way, the successor of a node is the peer following it. Since P2P nodes join or leave the network freely, nodes maintain multiple successor pointers to improve robustness.

In Figure 2.6, we can see an example of a Chord ring. In Figure 2.6(a) Nodes 0,1 and 3 are connected while nodes 2,4,5,6,7 are not. The size of the network is 8 and the currently available files/keys map to nodes 1,2 and 6. Since node 6 does not exist, the file (key=6) is mapped to the first available node, in this case node 0. The file (key=2) is mapped onto node 3. Chord continually maps the files along the Chord ring as peers join and leave the network. For instance, In Figure 2.6(b), if the Node 6 joined the overlay, the file (key=6) would be stored in that node, and also the successors of the nodes would vary according to this new topology. In Figure 2.6(c) when node 1 leaves the overlay, the finger tables are adjusted. And the key node 6 is responsible for are assigned to the next peer, in this case Node 3.

In Chord, there are two routing modes: *iterative and recursive.*

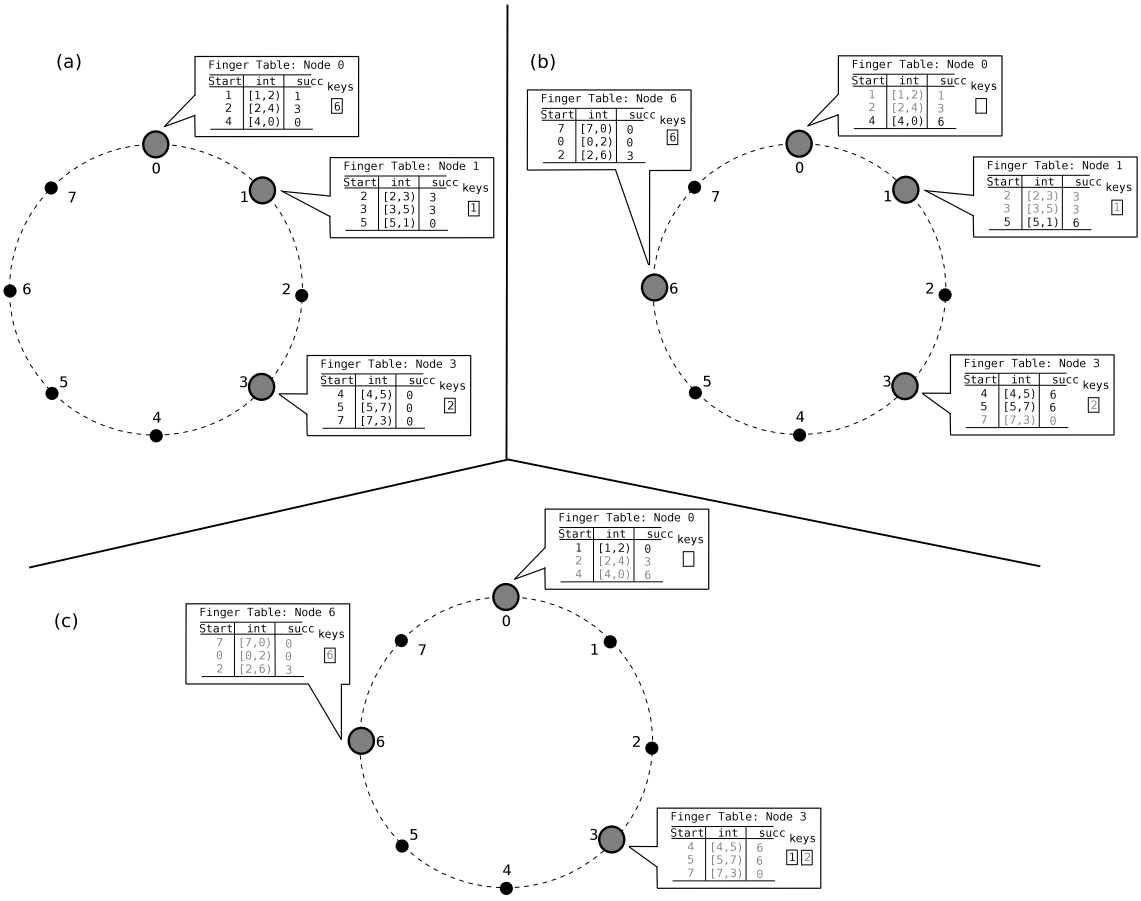

Figure 2.6: Example of a Chord ring (a), with the finger tables of the connected nodes. Examples of a node joining (b) and leaving (c) the overlay

1. In *recursive routing*, the request is forwarded to the next hop until it reaches the responsible node who will reply to the initial node. In Chord, the response is not returned directly, but along the reverse path followed by the query. In Figure 2.7(a) we can see an example in which node 1 asks for key 12. Node 1 first checks its own table but is not responsible for that key so it forwards the request to the next hop, which is node 10. Node 10 checks its routing table, and forwards the request to the node that is supposed to have the information. This node is node 14. Finally, node 14 replies to node 1 with the value associated with key 12.

2. Figure 2.7(b) shows an example of *iterative routing*. The request involves the very same nodes but the message is not forwarded to the next hop. Instead, each node along the path returns a response to node 1. Although recursive routing has a better performance, iterative is usu-

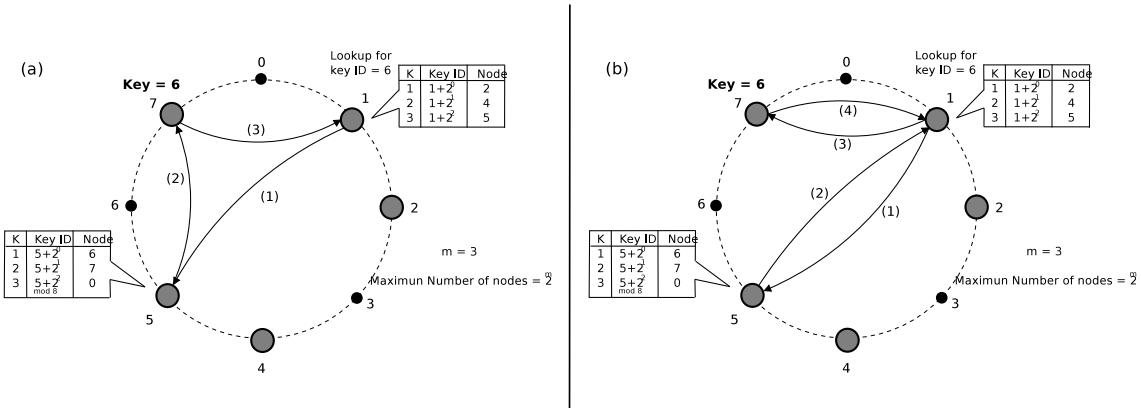

Figure 2.7: Examples of recursive (a), and iterative routing (b)

ally recommended for security reasons, since the message flow can be properly controlled.

2.4.4 RELOAD

P2P technologies are often used in combination with other protocols. A common trait is to take the lookup properties of a DHT based P2P network as a rendezvous mechanism to establish a connection. Once the connection is established, another application protocol can be used. Some research and standardization efforts are exploring the use of DHTs for communications. In particular, Peer-to-peer Session Initiation Protocol (P2PSIP).

In traditional Session Initiation Protocol (SIP) [75] architectures, there is a hierarchy of SIP routing proxies and SIP User Agents (UA) that follow a client/server structure. To start a communication session using SIP, a SIP user agent will send a SIP invite request to a proxy which will send it to the destination UA of the message.

In P2P networks, this type of communication is not possible, since there are no central servers to centralize the mapping. Instead, the mapping function is distributed among the peers of the network using a DHT.

The IETF P2PSIP Working Group [1] is chartered to develop a P2PSIP standard. In 2007, there were several competing proposals for the P2PSIP peer protocol; RELOAD [43], Peer-to-Peer Protocol (P2PP) [12], Address Settlement by Peer-to-Peer (ASP) [44], Service Extensible P2P Peer Protocol (SEP) [45], Extensible Peer Protocol (XPP) [61] and Host Identity Protocol HOP (HIPHOP) [20].

At the end of the year, RELOAD and ASP were merged and by February 2008 also P2PP was merged to the combined RELOAD/ASP protocol. Currently the P2PSIP WG has focused its efforts on the development of the REsource LOcation And Discovery (RELOAD) protocol, which is the peer protocol to be used in P2PSIP networks. Its main characteristics are defined in [43, 14, 15] among many other documents.

RELOAD provides NAT traversal mechanisms with Interactive Connectivity Establishment (ICE) [73], it can support various applications and provides a security framework. Moreover, RELOAD also allows the use of various DHT algorithms in the form of topology plugins. The one that currently used is Chord [60].

RELOAD supports two types of nodes: peers and clients, both of them are identified by NodeIDs. Peers are nodes that run the DHT algorithm and can route and store data. Clients do not provide any of those functions but can access the overlay services by connecting to a peer. Data stored in RELOAD is referred to as resources, identified by resource-ids. Its rendezvous features make it very suitable for locating nodes in a distributed environment such as M2M.

Its architecture is divided in three main parts, as shown in figure 2.8: Usage Layer, Topology Plugin, Overlay Link Layer.

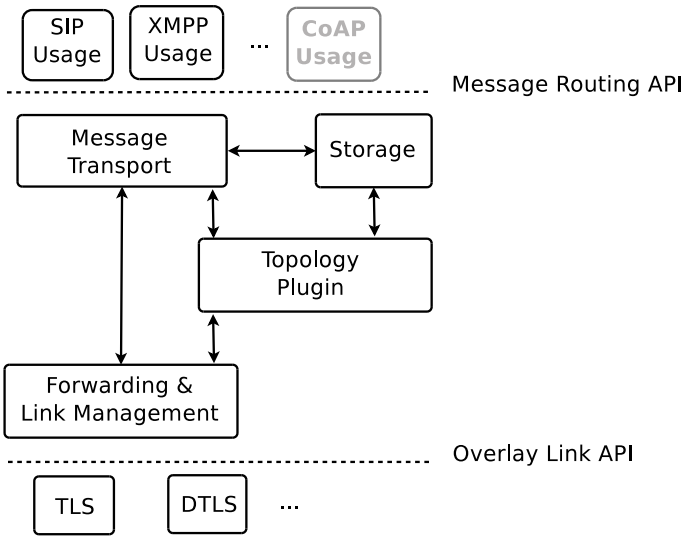

Figure 2.8: Major components of RELOAD.

Usage Layer

RELOAD is intended to be easily extensible. It provides the concept of usages. Each application that wishes to use RELOAD defines a RELOAD usage and interacts with RELOAD by a common Message Transport API , leaving the core of the P2P protocol untouched. This is why it would be realtively easy to design a CoAP usage for RELOAD for M2M, similar to the one presented in this thesis. Applications can use RELOAD to store and retrieve data, as a service discovery tool or to form direct connections in P2P environments. Some already defined usages are the SIP usage [42], the certificate store usage, the Traversal Using Relays around NAT (TURN) [74] server usage and so on. Message Transport layer provides a generic message routing service for the overlay, that is sending and receiving messages from peers. The Storage component is responsible for processing messages relating to the storage and retrieval of data.

Topology Plugin

Following again the extensibility principles, pluggable topologies are designed. Although in order to interoperate with other implementations it is mandatory to implement Chord. The topology plugin defines the content of the messages that will be used in RELOAD, the various procedures to join and leave an overlay, hash algorithm, etc. To set up connections in the open Internet, there is a layer to be used for NAT Traversal mechanism, the Forwarding and Link Management Layer. This layer uses Interactive Connectivity Establishment (ICE) [73] to traverse NATs.

Overlay Link Layer

At the moment both TCP and UDP are used at the link layer. Security is ensured by using Transport Layer Security (TLS) [23] over TCP and and Datagram Transport Layer Security (DTLS) [72] over UDP.

Chapter 3

Design

The goal of this project is to provide a M2M communication enabler (M2MCE) to provide functionalities to enable M2M communication. Such functionalities are resource translation, bookkeeping and addressing. Also as part of this thesis the messaging between nodes in the M2M network needs to be solved. A generic scenario sample is shown in figure 3.1.

The overal architecture we have designed consist on a set sensors or *local nodes* (LN) that form different LR-WPANs. The different LR-WPANs cannot intercommunicate with each other since they are placed several kilometers apart. LNs are also constrained in power and computational capability.

LNs gather data from their surrounding environment and share it with other nodes of the network via a *proxy node* (PN) that acts both as a WPAN coordinator and as an Internet enabled device. The use of a PN is necessary to connect the different LR-WPANs together. PNs may also have a sensor or actuator module attached to them and are not as constrained in resources as the LN devices are. Still PNs consume much less power than, for instance, personal computers do. A stripped down version of the PN is the *wide area node* WN, a device that is part of the DHT overlay, but is not responsible for any subset of LNs.

A *Monitoring and Controlling Node* (MCN) is used to monitor both WNs and PNs as well as LNs and their resources. This node is sporadically connected to the overlay and networkwise does not play a role more significant than the one played by a WN. Its main task is to create associations between the different devices to respond depending on the use case.

The *M2M Communication Enabler* layer (M2MCE) is used to interconnect all the parties: PNs, WNs, LNs, MCN and actuators. M2MCE bridges WPAN networking, WWANs and MCN monitoring. All PNs have to be addressable

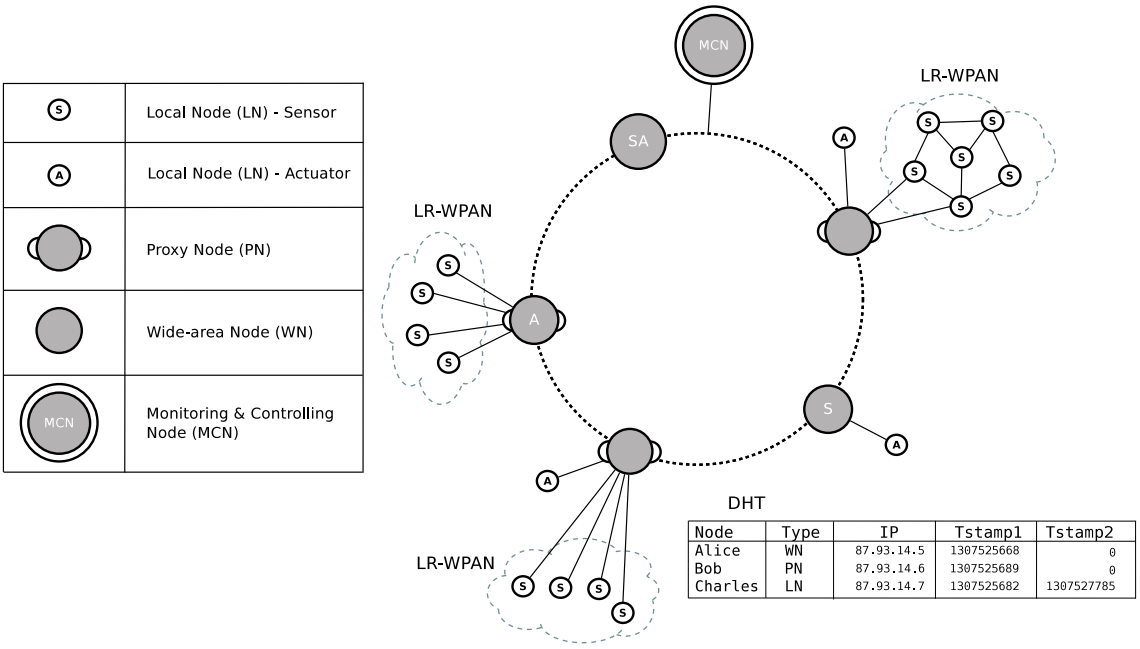

Figure 3.1: Architecture of the system

and must be able to create associations between each other depending on the scenario. In the case of the LNs, we would like to have used IPv6, 6LoWPAN and CoAP on top of IEEE 802.15.4 but due to higher availability we have chosen ZigBee. To enable wide are deployment and wirelessness, we have chosen 3G and GSM to connect PNs, WNs and the MCN. The application protocol to be used to address and retrieve resources in the overlay is CoAP. To monitor and control de nodes we have chosen SNMP, being as it is the mainstream standard. The hardware we used is discussed in the following section.

The network must be easily scalable and robust against failures, therefore a DHT overlay is used to connect the WNs and PNs. LNs share the information to the network via the PNs. The DHT algorithm we chose is Chord, since it is compulsory to be implemented in RELOAD and RELOAD has the main architectural principles we are looking for. The DHT is used for lookup of the nodes of the network in a similar way as a DNS does. The DHT is part of the M2M communication enabler (M2MCE) that is also used for Universal Resource Identifier (URI) translation and as a bookkeeping mechanism, storing the information of the different nodes of the overlay at any given time. Messaging between the MCN, the PNs and WNs is also enabled by the M2MCE. A simple communication scenario between the MCN, PN and LN is shown in

figure 3.2).

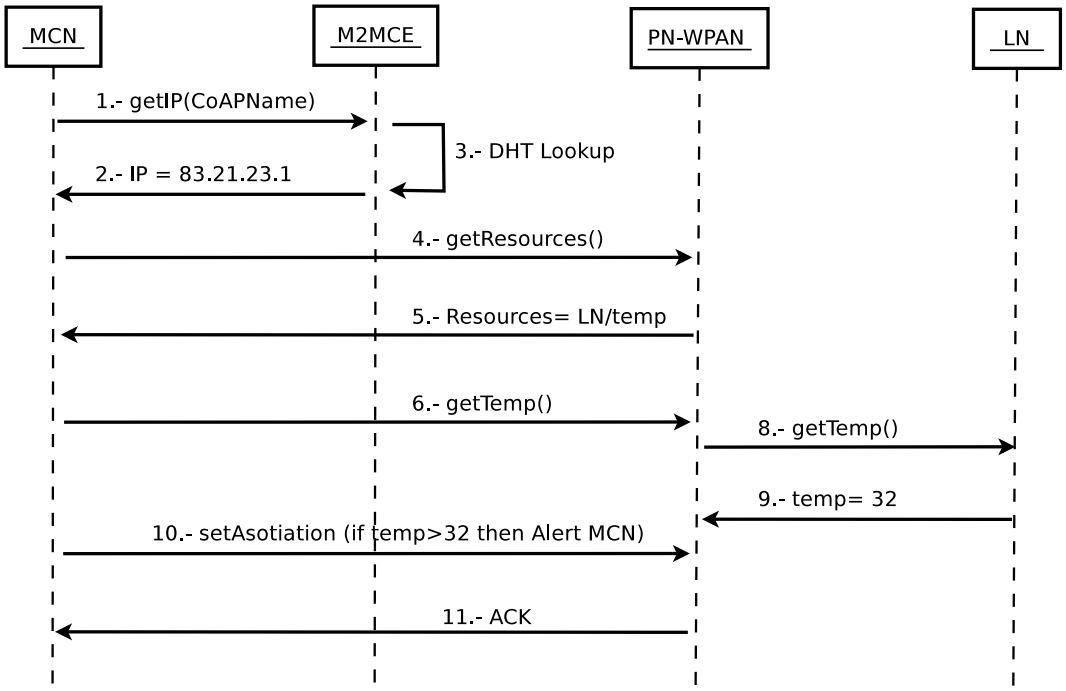

Figure 3.2: Sample sequence diagram of an MCN creating an association with a PN. The MCN first retrieves the temperature, then it sets an alert in case the temperature rises above a predetermined threshold.

3.1 Motivation

The M2MCE acts as a layer to enable different functionality for both nodes and MCN. M2MCE is necessary for the following reasons:

1. The M2MCE main task is to bridge both MCN and sensor networks and allow the sensor information to be reachable for the MCN and for other nodes. For this PNs and WNs have Internet connectivity and are IP addressable.

2. A P2P distributed environment is more robust against failure than a single server case. We believe robustness pays of over message overhead due to P2P networking.

3. The M2MCE has to run on proxy nodes. This is because LNs are too constrained to run our P2P implementation on them. In fact as we show in [57], although running RELOAD on mobile phones is entirely possible, it consumes too much battery power draining it completely in just few hours.

4. It is necessary to keep a database of the nodes of the overlay, as well as different parameters. For that the M2MCE provides a bookkeeping mechanism that efficiently stores this information. Such mechanism must provide cohesive results and avoid unnecessary replication. Data aggregation as well as functionalities similar to the ones provided by a server are also necessary.

5. The system must provide functionality to enable nodes to address each other using standardized protocols. Since sensors can be heterogeneous in nature and PNs too. It is important to align to current standard communication protocols to ensure compatibility. For this reason we chose CoAP, Chord and SNMP since they are the standardized proposals.

6. The M2MCE should allow nodes to make independent decisions based on the information in the network. In traditional WSN there is a single server or central point of control that gathers data, makes decisions and issues commands. In our model any node can retrieve important information from other nodes in order to make an informed decision and send an order to the actuator. These decisions are set by the the MCN and by state changes in the network.

3.2 Design Principles

The M2MCE layer is designed following different principles drawn from the constrains made by the technology and application scenarios. These principles subsequently affect the design and implementation of the M2MCE. The principles are:

1. Distributed Environment: The M2MCE should enable the distribution of the storage and node location service.

2. Scalability: The M2MCE should be operative at all times, regardless of the number of nodes in the overlay. This is particularly critical for the location and bookkeeping functionalities.

3. Security: The M2MCE should provide security for connections between peers, for messages sent in the network and for bookkeeping functionality.

4. Transparency: From the perspective of the other layers, M2MCE should hide the complexity of the DHT and present only the functions that are required.

5. Power efficiency: Although the nodes of the overlay are connected to a power source, they might be submitted to periods in which they will run on battery. Therefore unnecesary signaling should be avoided when possible.

6. Interoperability: The M2MCE should enable heterogeneous devices to communicate between each other. Currently there are three different types of devices intended to use the M2MCE. It should also be assumed that future nodes may have different architectures and functionalities, the M2MCE should therefore provide a transparent interface towards its different functions.

3.3 Architecture

Due to the previous motivation and principles the architecture is as shown in figure 3.3, using a WN as example. The main API to the M2MCE provides access to the different functionalities. It provides access to DHT forwarding, joining and leaving of the overlay. It also allows access to the Storage component, which is responsible for processing data, retrieving messages and has the bookkeeping functionalities. It provides name resolution for both the Proxy Node, Monitoring and Controlling Node and the Wide Area Node intelligence.

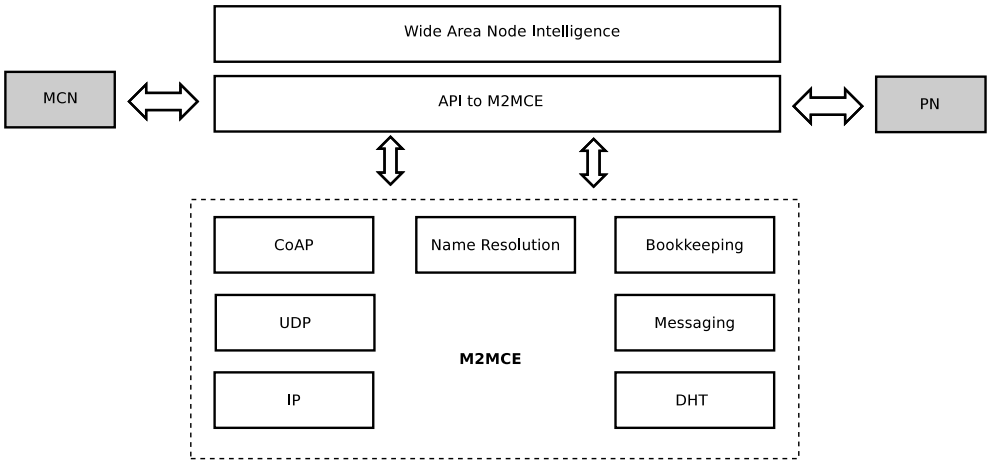

Figure 3.3: Architecture of the M2MCE in a WN.

The Proxy Node aggregates the data from the WPAN and acts as a bridge with the network. To locate the appropriate node it will use the M2MCE and the DHT. Wide Area Nodes, Proxy Nodes and the MCN use common protocols based as IP, like UDP, SNMP and CoAP. In figure 3.4 we can see the ZigBee transceiver and the associated stack towards the WSN, also the Gumstix Board and the protocol stack used towards the Internet, which is the same for the Wide Area Nodes and only changes at the application level.

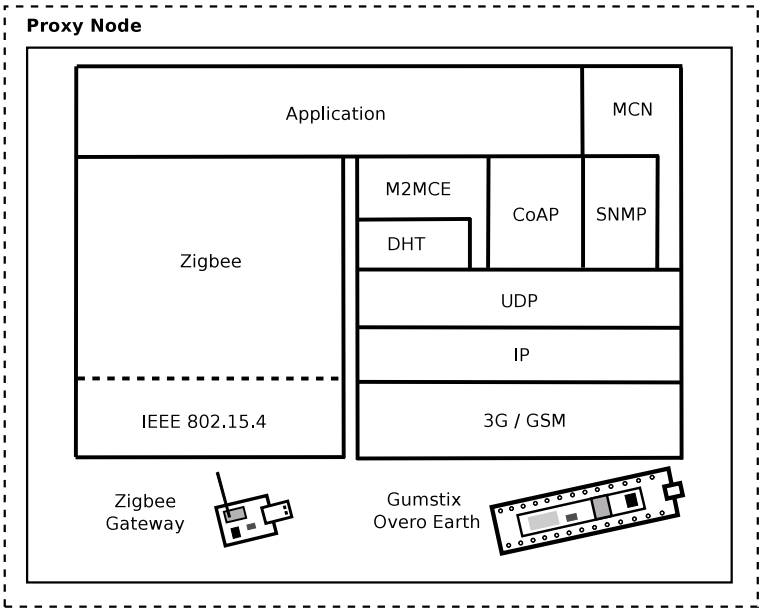

Figure 3.4: Protocol Stack of the Proxy Node (PN).

In this architecture the M2MCE contains a DHT, in our case we have selected Chord. The design is intended to use a subset of the RELOAD protocol for P2P signaling. Nevertheless, in the implementation we used Peer-to-peer Protocol (P2PP). RELOAD is intended to use Session Initiation Protocol (SIP) as application level, in our case we will use CoAP instead.

3.4 Design Details

3.4.1 CoAP communication

In our scenario communication between sensors and PN uses ZigBee due to its availability. Nevertheless, we believe that a standardized solution would be a better approach. Therefore we consider the practical use of CoAP.

CoAP provides four types of messages Confirmable, Non-Confirmable, Acknowledgement and Reset. For communication between overlay nodes we will use Confirmable messages when acknowledgment is required. PNs and WNs are connected to a power source and running a DHT, therefore they can run a CoAP server with no major problems.

Between LNs and PNs we will have two types of messages: Periodic updates and Direct queries.

1. Periodic updates - They are the most common type of message. We have designed our LNs to often be in sleep mode and only transmit during short periods of time, when the value they are registering goes over a pre-established threshold. In those cases the LN will go out of sleep mode and will send the new value of the resource to the PN. The PN in turn will update the value in its cache. This periodic updates do not need to be confirmable, since the sensor node should go to sleep mode as soon as possible, and listening for an ACK would consume battery. Moreover Confirmable messages would create unnecessary overhead, since they are retransmitted until the recipient acknowledges them.

2. Direct queries - Ideally they are not very frequent. Since LNs are on sleep mode most of the time when the PN receives a query for a resource stored in the LN, it will attempt to retrieve the latest information. If that is not possible it will reply with the latest cached value. If the LN is awake at that time then it will reply with a Confirmable message, wait for the ACK or Reset and go back to sleep mode. Waking times for LNs can be set by the MCN, the PN or agreed upon the respective parties. We will use modified Confirmable messages, retransmitting a minimal amount of times and increasing the timeout between retransmissions.

3.4.2 Joining of PN and LN

According to our architecture, before a WSN is accessible by the nodes of the overlay, the PN itself has to be part of that overlay. To join the overlay, it needs a CoAP Name, that will be either set on a configuration file or configured via the MCN. For our ideal M2M network, we are assuming that nodes are using IPv6 and that no NAT traversal mechanisms are required. If there is a need for NAT traversal, ICE should be used as it is recommended in RELOAD. Note that in the practice, most operators still assign only IPv4 addresses and that ICE is usually a must.

Joining of PN

The following joining procedure applies for all overlay nodes and is based on the joining procedure specified in RELOAD. In our design we assume that when a PN joins the overlay it should receive a NodeID from the enrollment server or, in case there is no such server, a Bootstrap Peer (BP) can play that role. For security purposes, it must also have a private key certificate that matches its NodeID and the public key certificate of the MCN.

In general, there are three steps to join the overlay: Forming connections to other peers, acquiring the data to be stored and updating other peers of that fact. In the case of the PN, the steps are shown in figure 3.5:

1. First, the Proxy Node must connect directly to the Bootstrap Peer (BP). In order to do so, the BP should have a public IP address. This is because this is the first connection for the PN.

2. After that, the PN sends a series of probe messages to the BP to populate its routing table.

3. The PN sends requests to initiate connections with other peers in the overlay in order to populate the routing and the finger tables. The latter is necessary for the Chord based DHT. This connections are routed via the overlay -they are not direct connections- since the PN is not yet part of the overlay.

4. The PN enters all the peers it contacted into its routing table. The peers that the PN has contacted can be used in the future as BP, since their IP addresses are also stored.

5. The PN then needs to get a copy of the data it is now responsible for storing. For that, it will try to contact the Admitting Peer (AP) that was until that moment responsible for that data. The PN will request to join to the AP. The AP should in turn respond.

6. The AP will then store in PN the overlay data that PN will be responsible for.

7. The AP sends the PN an update explicitly labeling PN as its predecessor. It also sends an update to all of its neighbors with the new values of its neighbor set (including the PN). At this point, PN is part of the overlay and responsible for a section of it. AP can now forget any data assigned to PN and not itself.

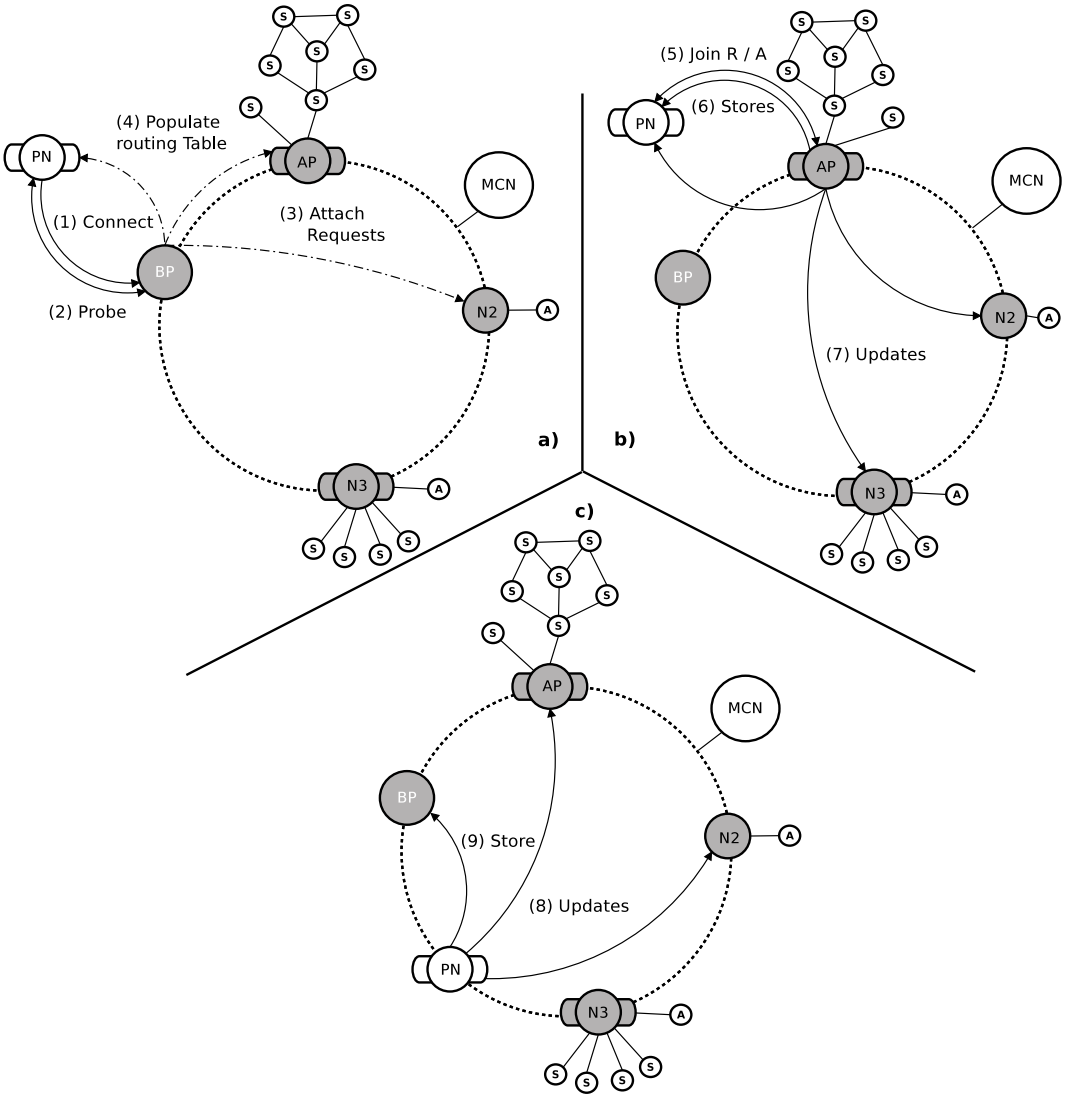

Figure 3.5: PN joining the overlay.

8. The PN now knows the addresses of its predecessors (the same as AP's) so it connects to them. The PN now has a copy of the data and is capable of routing messages and receive them. Therefore it sends updates to all the peers in its routing table telling them it is ready to go.

9. The PN updates its bookkeeping information in the overlay, such as a timestamp indicating the time it joined, battery status, IP address, name and so on.

At this point the PN is part of the overlay. For our prototype the MCN is

controlling the addition of nodes to the network. It is then the MCN who switches on a sensor and permits it to make its resources available. From a design perspective it is more convenient if the WPAN islands are independent. Therefore in the design the MCN has been excluded from the LN addition procedure.

Joining of LN

According to our proposal, once the PN has updated its bookkeeping information, the next step is to make the WPAN nodes it is responsible for accessible from the DHT. PN also is the coordinator of the WPAN. It is therefore a FFD device, responsible of the initial configuration of the WPAN network. The WPAN joining procedure is based on the one described in ZIGBEE and most of its details related to WPAN Management are out of the scope of this thesis. As shown in figure 3.6, the relevant steps are:

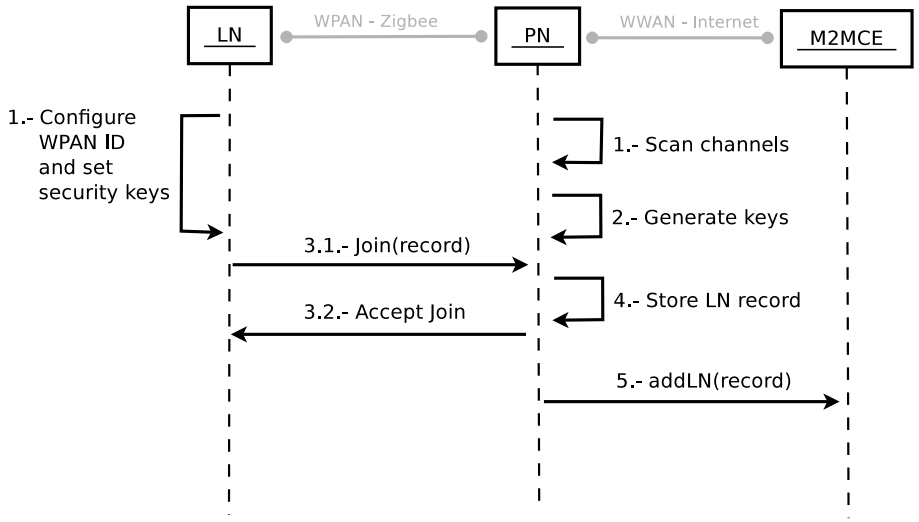

Figure 3.6: LN joining the WPAN.

1. First, the PN will choose a free WPAN ID by scanning for potential channels and discarding the ones in use.

2. Then, if encryption is required, the PN will generate the relevant network security keys and encryption options. Now a LN is capable of joining the WPAN network if it conforms to the security policies and operates in the same WPAN channel.

3. The LN can join the network if the PN permits it.

4. The PN will store a record of the LN as well request from it the different services it provides. The information stored is identity, service type, supported access methods, and availability.

5. This information can now be stored in the overlay via the M2MCE in the PN.

Now the resources/services provided by the LN are available to other nodes of the overlay. At this point we have a PN that is part of a DHT and WPAN coordinator. The next steps is to make its resources addressable. Their information is stored in three places: the LNs themselves, the proxy node and the DHT. Each node must store its CoAP Name and the IP address, see Section 3.4.4 for more information about this. Apart from that, different services information are stored, such as join/leave timestamps and various status values (battery, connectivity or update frequency). Details related to bookkeeping will be explained in greater detail in Section 3.4.5.

3.4.3 Leaving of PN and LN

The following leaving procedure applies for all overlay nodes and is based on the leaving procedure specified in RELOAD. In the case of the PN, leaving the overlay is relatively simple. As shown in figure 3.7, there are two ways a node would leave: gracefully and ungracefully.

Graceful leaving is performed when a node departs from it and wants to update the overlay of that fact. The steps are:

1. The PN updates its bookkeeping information in the overlay, it will also update the information that the LNs it is responsible for.

2. The PN sends a leave message to all its neighbors informing them about its departure from the overlay. They will in turn acknowledge it and reply back.

3. Upon receiving a leave request, peers will update their own routing table, store and update to re-estabilize the overlay.

4. The nodes remove the PN from their own table

Ungraceful leaving implies disconnecting without storing the data that the node is responsible for in another peer nor updating other peers. This data loss is paliated by the automatic replication mechanisms of the DHT.

In both scenarios, the periodic overlay stabilization will afterwards update the rest of the nodes [43, 59].

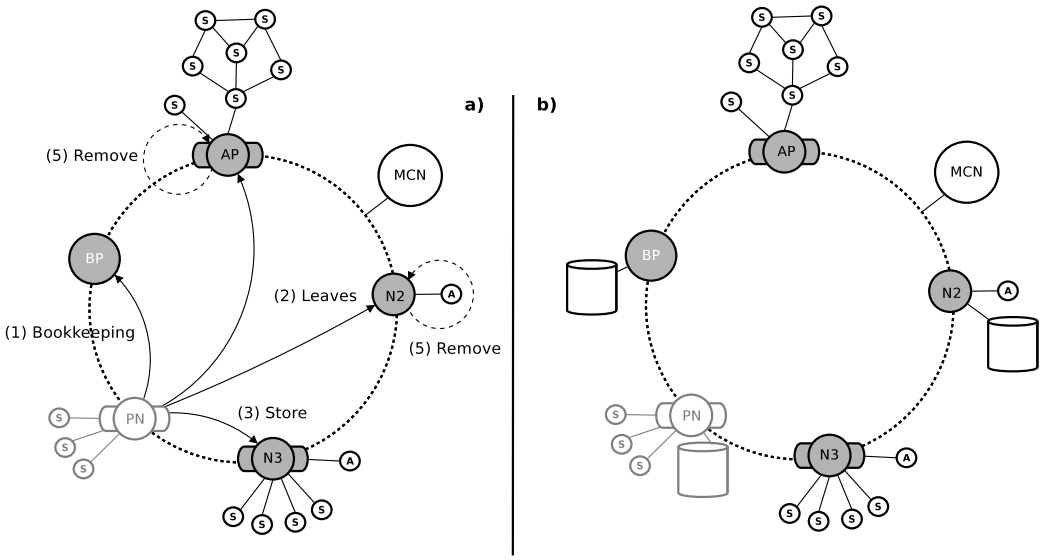

Figure 3.7: Proxy Node and Local Nodes leaving the Overlay in a graceful (a) and ungraceful (b) fashion.

3.4.4 CoAP Name Registration Service

We propose a similar service to a Distributed DNS (DDNS). In our case the resource ID is calculated by hashing the CoAP name of the storing node. That hash will give the location where the value should be stored in the overlay. The value to store is the IP address of the storing node in the case of the PN, WN and MCN. In the case of LNs, the PN will store the information on their behalf, the IP address will be its own, since ZigBee devices do not have IP. In the future, we will use CoAP and IP, and the stored value would be the actual address of the LN. The value can be overwritten at anytime by the owner of the data.

As an example, a proxy node could register its IP address "123.45.67.89" under its CoAP Name, "pn_23". When a node needs "pn_23" IP address to retrieve some resources, it queries the overlay for "pn_23" and gets back IP address "123.45.67.89". The node can then start a CoAP direct connection with "pn_23".

3.4.5 Bookkeeping

Retrieving node information from the overlay can be done in three different forms:

1. Accessing the LN information directly. In this case a node in the overlay will do a lookup of a CoAP name that will return the IP address of the responsible node. Then it will form a direct connection and send a CoAP get request. The receiving node (PN or WN) will then translate and forward the request to the actual sensor. Then sensor will reply and the PN will forward the message via CoAP to the requester.

2. Accessing the cached information in the PN. Same procedure as the previous one but when the PN or WN gets the request, it will access the latest cached copy of it. This is done to save battery on the sensor node. The trade-off is that the sensor data might not be up to date.

3. Accessing the cached information in the Bookkeeping mechanism of the M2MCE. For our use case it is necessary to access individual node information, for that we do a lookup in the DHT of the particular node. Often, it would be necessary to know the status of all the nodes in the network at the same time, what is more to keep track of which nodes are there and which have departed. Moreover it would often be required to keep initial sensor information in the overlay and the latest list of nodes that have joined or left the network.

Bookkeeping can not be left to the DHT alone. A large increase in nodes would cause a message overhead and that the system's topology over time will eventually offset the benefits of having such cache.

The solution we propose is a two layered DHT. As it is shown in figure 3.8 we would have the traditional Chord overlay for lookup functionality of node information. We will also have a second DHT that will store the bookkeeping information.

Similar approaches have already been taken, for instance in [80] a structured DHT is used on top of an unstructured one, being the structured one used for lookup. In fact, the problem of managing shared resources and their access policies is being currently standardized [48, 70] to allow users to share their write access to specific resources with others peers.

Nodes that perform database tasks are called database nodes. Database nodes are tasked to store the bookkeeping information. In our network, all peers have similar computational power and network bandwidth. Therefore peers will become database peers on a random basis, depending on network conditions. We favor the hybrid P2P architecture since we aim at making the system fully distributed and at retrieving all data in just one query. In the future, data management techniques such as these can be used as a core part of any M2M network when we need to retrieve overlay status information [37]

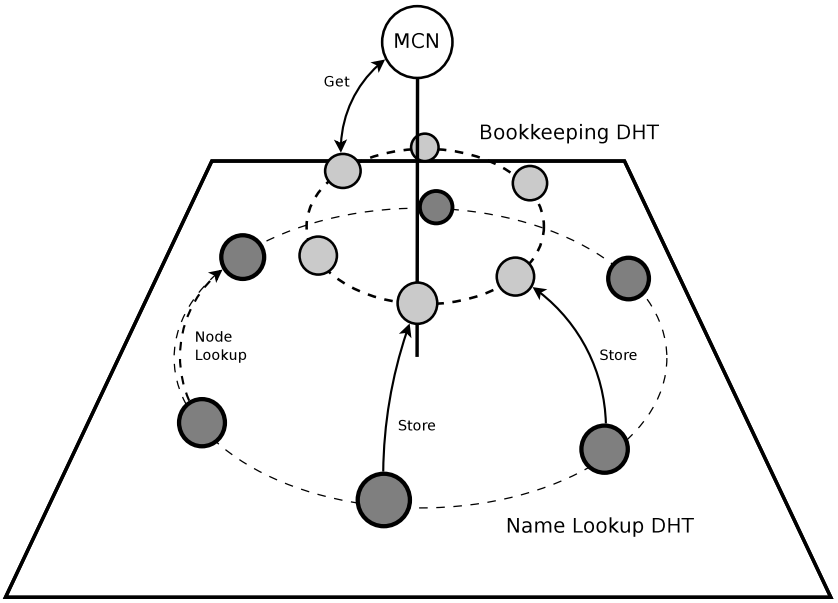

Figure 3.8: Architecture of the bookkeeping mechanism

PNs should not inform the overlay about all changes that occur in the LNs. That would create a large message overhead, since theoretically there would be several thousands nodes. Moreover it would drain the PN's battery. However, when critical changes do occur, they can be updated in the Bookkeeping mechanism.

Although the information of the nodes matches a single resource ID in the overlay, it is not stored only in one node. Instead we distribute it among the nodes in the Bookkeeping overlay layer. Each node keeps a fraction of the list, at the end of each part, there is a pointer to the next. The pointer contains the NodeID of the next responsible node, as it will be explained in detail in the security Section for bookkeeping 3.4.6.

3.4.6 Security

In our design we made several assumptions. First, we have not developed how certificates are distributed. In the design, these certificates are assigned by a central server which also assigns Node-IDs. In the practice we use self-signed certificates. Second, we are assuming that the MCN knows all public key certificates of the nodes in the overlay and that all nodes in the overlay know the public key certificate of the MCN.

Considering that, security in this design is done at two different levels: WPAN

security, this involves the security of the communications between the WSN nodes; WWAN security, this involves the security between the DHT nodes and application level security on CoAP and SNMP.

WPAN Security

Security in Zigbee is already managed by the Trust Center. The Coordinator of the WSN takes that role, being the repository for security keys. The TC decides whether to allow or disallow new devices into its network. Ultimately, for our prototype, that decision is delegated from the Proxy Node to the Monitoring and Controlling Node. ZigBee uses three types of keys to manage security: Master, Network and Link key. Their use is explained in detail in the ZigBee specification [67].

Every node must have a shared key that is the same for all the nodes in the network. Link Keys are used between devices that communicate with each other. Master Keys are used as an initial shared secret between two devices when they perform the Key Establishment Procedure (SKKE) to generate Link Keys.

As shown in figure 3.9, there are two auxiliary headers added at the Network and at the Application layer, moreover the application payload is encrypted. Integrity protection adds a Message Integrity Code (MIC), this is a signature bound to the originator of the message that provides integrity protection.

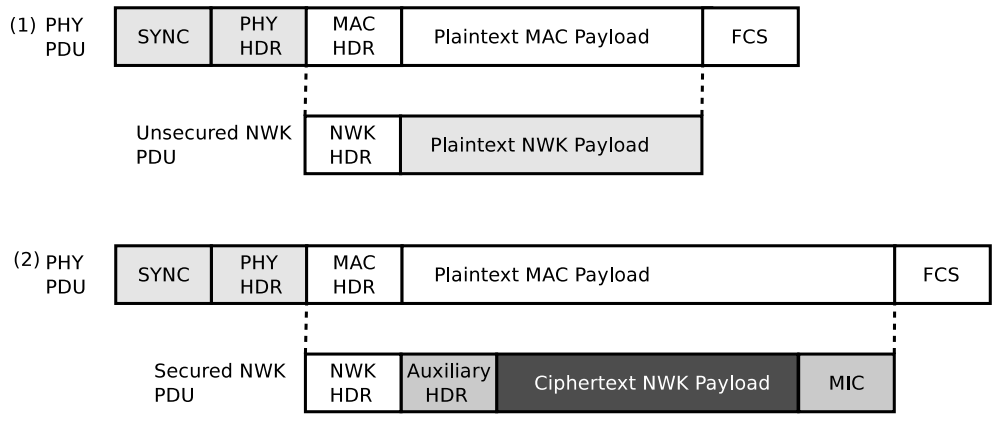

Figure 3.9: Unsecured (1) and secured (2) Zigbee PDU.

WWAN Security

WWAN security is done in three different areas: Overlay messaging, CoAP and SNMP, CoAP-IP mapping and Bookkeeping Security.

Overlay messaging security

Each DHT node shall have a list of certificates of the nodes it has contacted. We used the same certificate as specified in RELOAD, X.509 [19].

The signature shall be computed for each WWAN node's CoAP name, their NodeID in the DHT and the ID of the overlay they are in. In the case of the WPAN nodes, the PN will act as a proxy storing each LNs certificates and operating in their behalf. This is possible since the PN is also the Trust Center of the WPAN, being the repository for security keys in WPAN. Furthermore, like CoAP, we have chosen to send messages using an encrypted transport such as DTLS, therefore the communication is always encrypted.

We use standard public/private key cryptography to verify and generate signatures. When parsing the message we would use the following cryptographic procedure to verify the signature (see Figure 3.10).

The process starts when a node stores data in the overlay or sends a DHT message. We are now assuming our practical example, in which the certificate is self-signed. First the hash over the message will be calculated using SHA-1 [25] and then the hash will be encrypted using the RSA [46] algorithm and the private key of the sender. The certificate is also signed using the sender's private key. Then the certificate and the signature will be included in the message that will be sent.

When the receiver gets the message it will first get the certificate of the sender. The receiver will verify the certificate and obtain the public key of the sender. Then it will parse the signature by first decrypting it. For that it needs the previous public key and the RSA algorithm.

Once the signature is parsed, the original hash of the message will be obtained. Then another hash over the message will be calculated on the receiver side, using again the SHA-1 algorithm and the relevant fields of the message that were already parsed.

To conclude the receiver will compare both hashes and in case they are not equal that will imply a corruption of the data before arrival.

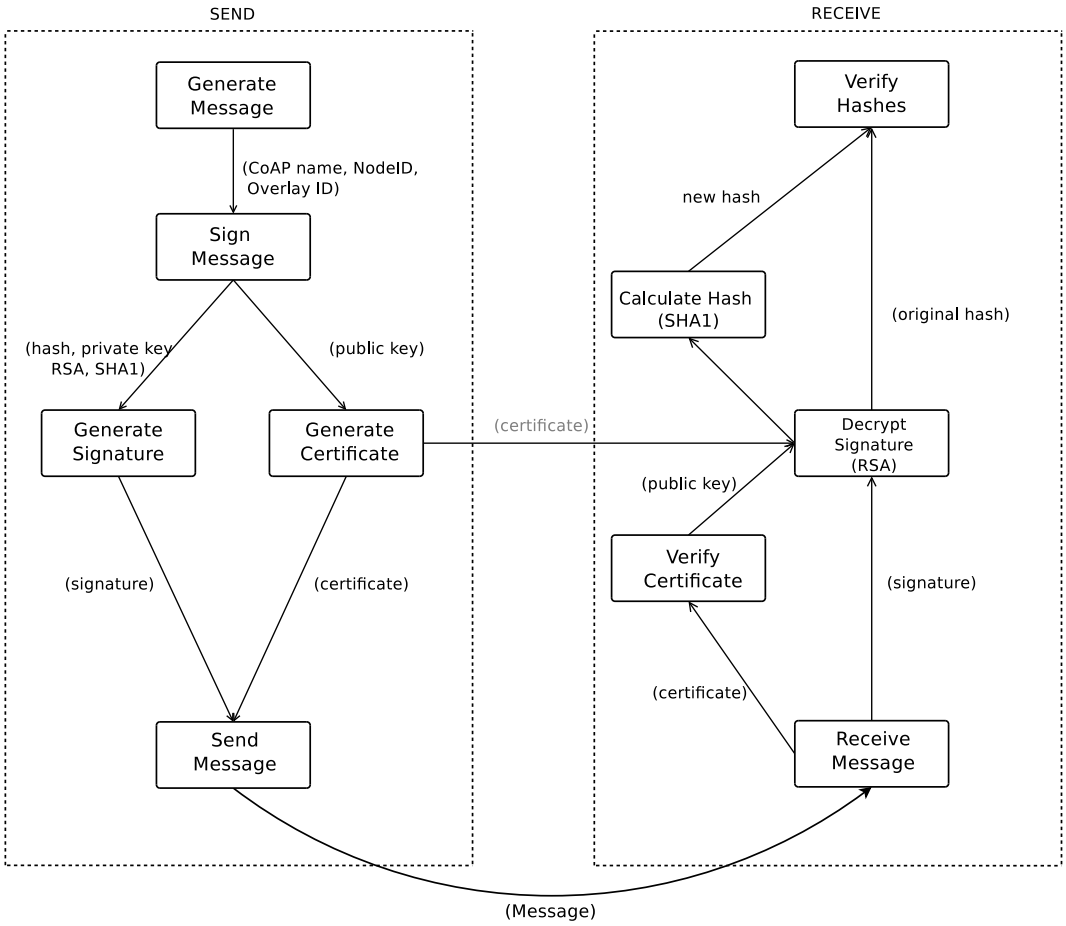

Figure 3.10: Process of generating, sending and parsing a message

CoAP and SNMP security

In CoAP, security is already ensured using Datagram TLS (DTLS) [72] binding or IPSec [34, 24]. Security in SNMP is out of the scope of this thesis, since it is part of the MCN implementation, some relevant RFCs treat the subject extensively [36, 62, 90, 13, 35], and many more.

CoAP-IP mapping Security

CoAP-IP mapping security is achieved by storing single values in the overlay. Since stored objects in our overlay must be signed by the creating peer, security is already provided by the overlay itself. For object level security in order to protect stored data from tampering, by other nodes, each stored value is

digitally signed by the node which created it. When a value is retrieved, the digital signature can be verified to detect tampering.

Bookkeeping Security

Introducing another overlay layer just for bookkeeping purposes permits the MCN to retrieve overlay data with just one get operation. Security is achieved by protecting the entries that conform a Node Information List.

When a Node wants to update or do an initial store of its node information, it will do a store("node list", node_n_information). As shown in image 3.11 the receiving node will belong to that extra overlay we have designed, it will receive a message encrypted with the public key of the MCN and signed with the public key of that particular node and some fields of the payload.

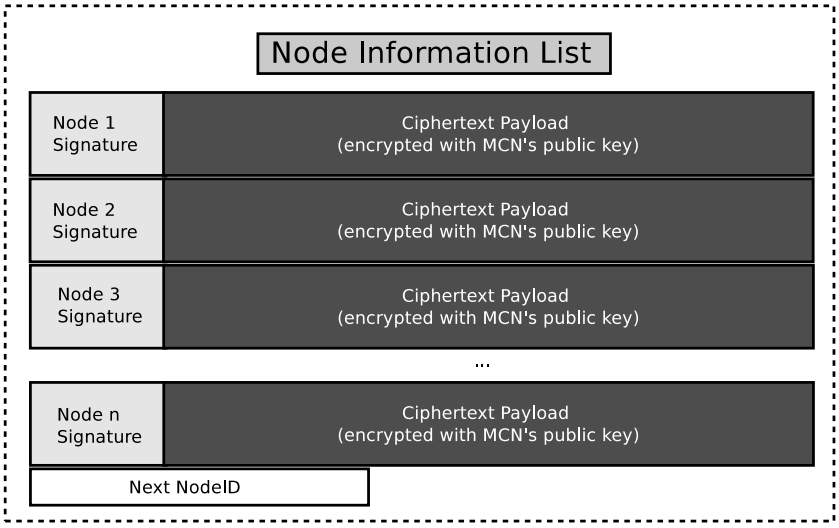

Figure 3.11: Structure of the Node Information List, storing each node's information in one resource.

With this model we guarantee *confidentiality* since only the MCN can read the encrypted payload. By using digital signatures the peer responsible for storing the data can verify the *authenticity* of the message and that this store operation is valid. In addition, it provides *integrity* of the data since the signature is saved along with values of the payload so that the MCN can verify the data.

Availability is increased since the DHT algorithm provides a replication mechanism for the information. Chord, in particular, stores replicas in the predecessor and the successor of the node. Replication also increases *persistence*, if the

responsible peer crashes or if the storing peer leaves the overlay. Replication also guards against Denial Of Service (DoS) attacks.

To improve *load balancing*, since there are many queries for a resource as popular as Node Information List, we propose extra replication at other nodes in the finger table.

3.4.7 Benefits of the Architecture

This architecture has certain advantages. First of all it allows to federate geographically distributed WSN islands. CoAP provides a common namespace for resources in all interconnected WSNs. A sensor in one WSN can access the resources in another WSN since the WSNs are interconnected by the DHT overlay.

The architecture enables decentralized wide-area sensor and actuator architectures. The DHT and the M2MCE provide lookup, storage, bookkeeping and message routing. The overlay maps CoAP URIs to contact information of sensors. Since the DHT is used, it acts as a Distributed Domain Name Service (DDNS), translating CoAP names to addresses. All this without the need to rely on central servers like resource directories or DNS-based service discovery (DNS-SD) [17] that is used by CoAP usages like the one specified in [22] or in home automation [50].

Moreover, this architecture integrates WSNs to the web. Web applications can access the resources in the WSNs federated by the M2MCE using CoAP thanks to the Proxy Nodes. This architecture also enables CoAP clients to access resources on web servers if necessary. In general this bidirectional integration enables new applications such as the Web of Things [47, 85, 39]. The architecture also makes resources in Zigbee based WSNs accessible through CoAP and the Proxy Node. In the long term, technologies like 6LoWPAN will be used instead.

Since the architecture proposes the use of structured P2P technologies, it becomes self-configuring, scalable, robust, and cost-efficient. Scalability is achieved since the addition of a PN or WN, will add resources to the systems. If a high number of devices are added, this should not require investment in new capacity. If we take a traditional client/server system, each node in the network consumes additional resources on the central servers and that leads to the need to add more capacity over time.

The architecture is robust since it does not depend on centralized elements for rendezvous and for relying data. Cost efficiency is achieved since the system has low capital expenditures because there are no central servers or data

centers and, more importantly, low operating expenditures since nodes require low maintenance.

3.5 Use Cases

At this point in time M2M technologies are still blossoming, therefore it is difficult to predict the most useful applications and uses cases. This thesis focuses on dispersed and mobile M2M networks in which cellular solutions fit well. On the other hand fixed and concentrated ones, such as home automation, do not require a priori cellular connectivity.

3.5.1 Dynamic Traffic Signaling

This use case derives from the necessity to solve some some traffic problems. In certain weather conditions (i.e ice, snow, low visibility...) it might be necessary to change the information of the traffic signs at the side of roads that provide information to road users. It mighto be required to give extra information of the set of conditions of the road, or warn about incidents. Traffic volume, which increases and decreases at different times of the day, will also be a factor.

Although not widespread, the technology to provide Variable Speed Limits (VSL) already exists [76] and it has been introduced in Britain, Germany where accidents due to fog were reduced by 80% [76], Austria and Sweden where it is expected to reduce the overall number of fatal crashes by 50% [56]. This technology automatizes speed limit checking, applying truck regulation and warning about dangerous weather and traffic conditions by using changeable traffic signs. The control and monitoring of these signs is done manually at a traffic control center, but human intervention implies lower response in case of immediate risk and higher costs associated with the deployment of this technology.

Again, applying a distributed network of sensors seems a good approach for this problem. As an example, if the road becomes icy, when a sensor detects the change in the conditions of the road it will inform the gateway of this change. In turn, the gateway will reduce the speed limit displayed on the sign and show a message warning of that change. It will also propagate it to the neighboring nodes informing them of this change, thus the other nodes will also adapt their signals reducing as well the speed limit or operating accordingly to the new conditions. An example in figure 3.12 shows a sensor (1) that detects ice on the road and alerts the gateway (2) that modifies the speed limit. The gateway in turn alerts the other two nodes of the DHT (3) and (4) that also

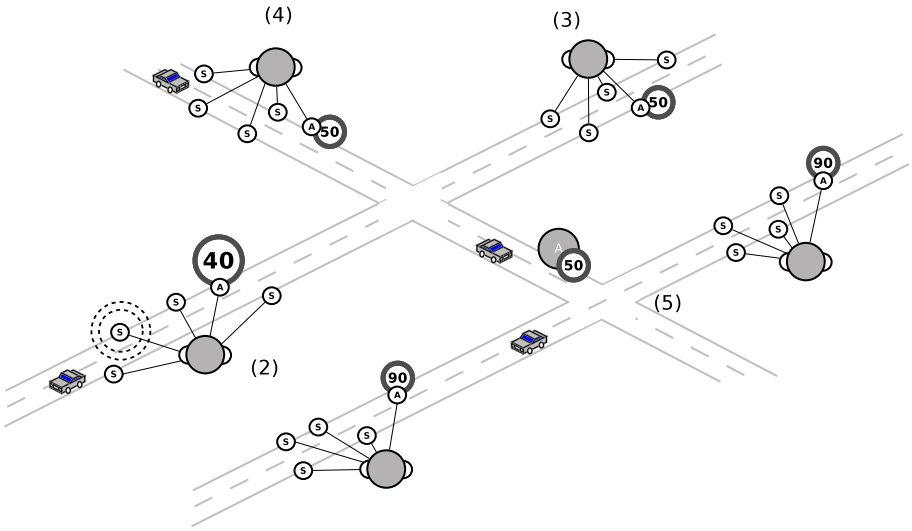

Figure 3.12: Dynamic traffic monitoring use case.

modify the speed limit. Traffic in the parallel road (5) is no modified since there is no need for it.

3.5.2 Water System Automation

Most modern cities have a water system that takes care of the process of taking raw (untreated) water, purify it, pressurize it and distribute it through a pipe network to the consumers and usage points. It also takes care of connecting waste water to the sewer system.

According to ABB [8], worldwide there are two types of countries when it comes to water management. Those with a relatively low water loss (below 15%) like Finland, Switzerland or New Zealand and those with high water loss (more than 40%) like India, Turkey or Bulgaria that represent the majority of countries. Most of that water loss does not arise from bursts in the main pipes, which have high flow rates, but rather on the service pipes due to lack of detection and timely reaction.

Considering again the set of hydrological and hydraulic components that constitute the water supply we are going to focus on the pipe network. These pipes usually copy the traditional city block grid pattern [66]. This structure is very useful to distribute water efficiently and also provides loops used to re-route the water in case of leakage or lack of pressure in some part of the network. When a leak occurs the responsible valve needs to be manually closed. Even if

state of the art technologies involving magnetic sensors that detect differences in water pressure or electronically activated valves that can close a pipe in case of a leak are used, they still rely on a centralized (and expensive) control center.

Applying a distributed network of sensors seems an excellent solution for this problem. For instance, if a leak happens in one pipe of the network, when a sensor detects the difference in pressure in the pipe it can alert the gateway of this change which in turn would activate the responsible valve to close the pipe. Then the gateway would share this alert with other nodes of the network in order to efficiently and automatically reroute water to that location if possible and to equilibrate the pressure.

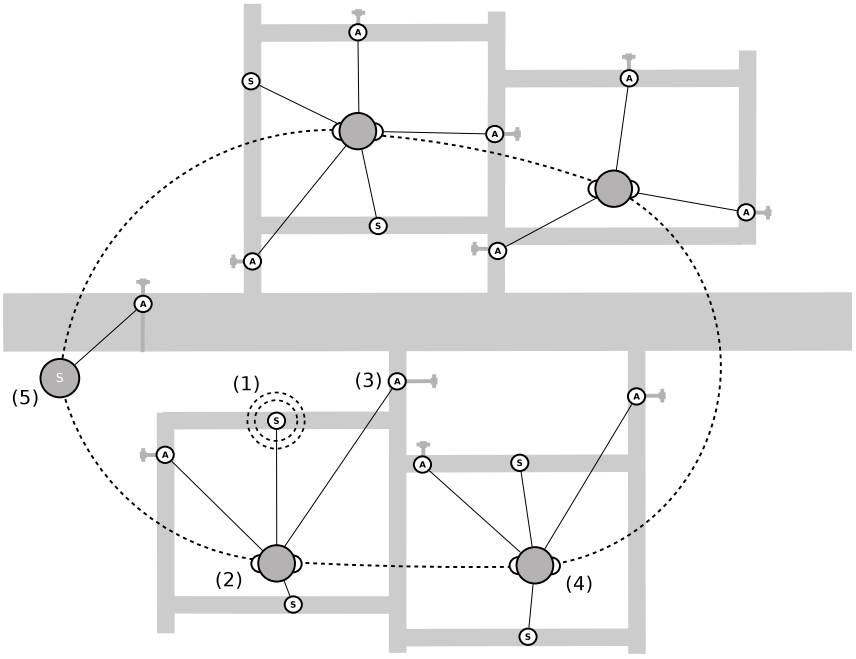

Figure 3.13: Water system automation use case.

Another feature would be the use of chemical sensors that continuously test the water for poisons and other chemical components that would be nocive for the population. Upon detection, the gateways could immediately close the water supply without the need of human intervention that would cause a fatal delay and communicate with the other nodes of the network to reroute clean water if possible. In figure 3.13 we see an example in which a sensor (1) detects a leak in a service pipe and alerts the gateway (2). The gateway in turn closes the access valve of the damaged pipe (3), alerts node (4) to reroute the water flow to the destination and node (5) to increase the water pressure of the main

pipe to compensate for the previous leak.

Chapter 4

Implementation

For this thesis, we have implemented a prototype of a wide area node and the M2MCE layer. The major functionality consists on the bookkeeping mechanism, an interface towards other layers and the MCN and the use of the DHT overlay for DNS functionality translating from CoAP to IP address. We introduce what hardware and software has been used and some examples that have been implemented.

4.1 Hardware and Software

This section with deal with the hardware used for our experiments: the LN, the WN and the PN.

4.1.1 Local Node (LN)

From all the sensors that were evaluated none had IPv6 nor 6LoWPAN at the moment this thesis was written. Some of the LN alternatives can be seen in the following table:

Name	Microcontroller	RAM and Flash	OS and other
Iris Mote	ATmega 1281	8K and 128K	TinyOS
Mica Mote	ATmega 103	128K and 512K	TinyOS
Mica Mote	ATmega 103	128K and 512K	TinyOS
Mulle	Renesas M16C	31K and 384K	Contiki and TinyOS
IMote 2.0	ARM 11-400	32 MB and 32 MB	Linux and TinyOS
Waspmote	ATmega 1281	8K and 128K	Different modules

The Libelium[1] Waspmote[2] was chosen over the other two main competitors, Mulle [3] and Irismote [4], since it provided all components (see Figure 4.1) in a development kit, therefore implementation was more straightforward than in the other cases.

The Waspmote includes a Zigbee transceiver, a Digi4[5] XBeeTM ZB5. It also had different sensor boards for different applications such as: gases, transport, smart cities, agriculture and event management.

For our case we only used the default Waspmote with no sensor board attached, the only sensors on it where the temperature and the accelerometer. Its characteristics are a low-power Atmel3 8-bit RISC-based microcontroller with a frequency of 8MHz, the RAM memory is of 8KB and the flash memory is 128KB. It also has 4KB of EEPROM and other communication modules such as Bluetooth and GPRS. Its consumption is of 9mA when connected, 62uA on sleep mode and as low as 0,7uA in hibernation.

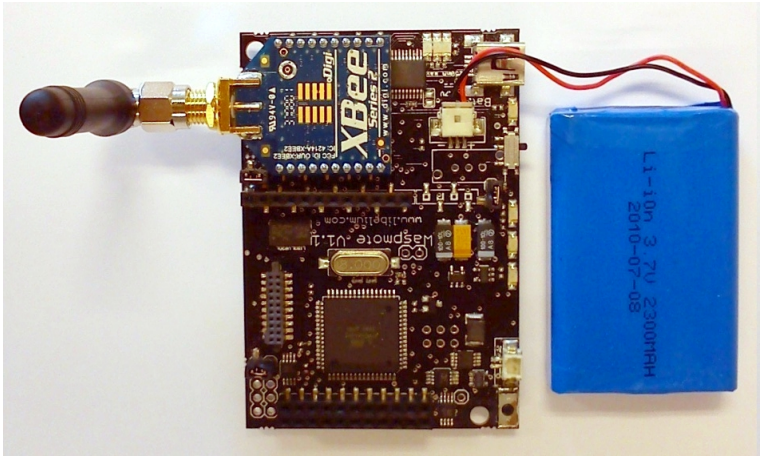

Figure 4.1: A Libelium Waspmote

Libelium Waspmote's Atmel [6] board uses the open-source electronics prototyping platform called Arduino[7]. Arduino IDE provides a cross-platform application development environment that allows compiling and uploading programs to the board directly. Libelium also provides APIs to access different

[1]http://www.libelium.com
[2]http://www.libelium.com/products/waspmote
[3]http://www.csee.ltu.se/ jench/mulle.html
[4]http://www.xbow.com
[5]http://www.digi.com
[6]http://www.atmel.com
[7]http://www.arduino.cc/

components to handle the sensors on the board [82, 83] and examples [8].

4.1.2 Wide-Area Node (WN) and Proxy Node (PN)

The Wide-Area Node, has been built on a Gumstix Overo® Earth™ [9] computer-on module[10]. The module has a ARM Cortex™ A8 CPU[11], the processor is a Texas Instruments[12] OMAP 3503[13] Applications Processor running at 600 MHz. Another of its main properties is its small size, the dimensions are 17mm x 58mm x 4.2mm and it only weights 4.3 grams. All this provides a really small computer powerfull enough to run a Linux system. The Linux distribution is in a 4GB micro SD card that contains the kernel image and the root filesystem. There is a community built around the development on Gumstix[14].

We use a custom distribution based on the pre-built images provided by Gumstix[15]. We chose the Linux Angstrom Distribution 2.6.34 release 2010.7-test-20101020[16]. We use opkg[17] package managment tool to fetch meta-packages. All the implementation is done in Java therefore we needed a Virtual Machine to run the applications, the two options were JamVM[18] and CacaoVM[19] the later was already included in the opkg repository.

As shown in figure 4.2.(1) Overo Earth is powered via an expansion board connected to dual 70-pin AVX connectors. For our tests we had two different boards: Pinto-TH™ 4.2.(2) and Tobi™ 4.2.(3).

The Tobi expansion board[20] provides 0/100baseT Ethernet (I), SB Host (II), DVI-D signals on an HDMI connector (III) and also features audio stereo in and out (IV), USB OTG (V), +5V power port (VI) and USB client (VII). To connect to the Overo Earth we connected a mini-B to standard A USB cable from USB console port to a USB port on our test laptop and then opened a kermit[21] console session.

[8] http://www.libelium.com/development/waspmote#examples
[9] http://www.gumstix.com/store/product_info.php?products_id=211
[10] http://www.gumstix.com
[11] http://www.arm.com/products/processors/cortex-a/cortex-a8.php
[12] http://www.ti.com/
[13] http://focus.ti.com/docs/prod/folders/print/omap3503.html
[14] http://www.gumstix.org
[15] http://cumulus.gumstix.org/images/angstrom/factory/
[16] http://cumulus.gumstix.org/images/angstrom/misc/daily/
[17] http://wiki.openmoko.org/wiki/Opkg
[18] http://jamvm.sourceforge.net
[19] http://www.cacaovm.org
[20] http://www.gumstix.com/store/product_info.php?products_id=230
[21] http://www.columbia.edu/kermit

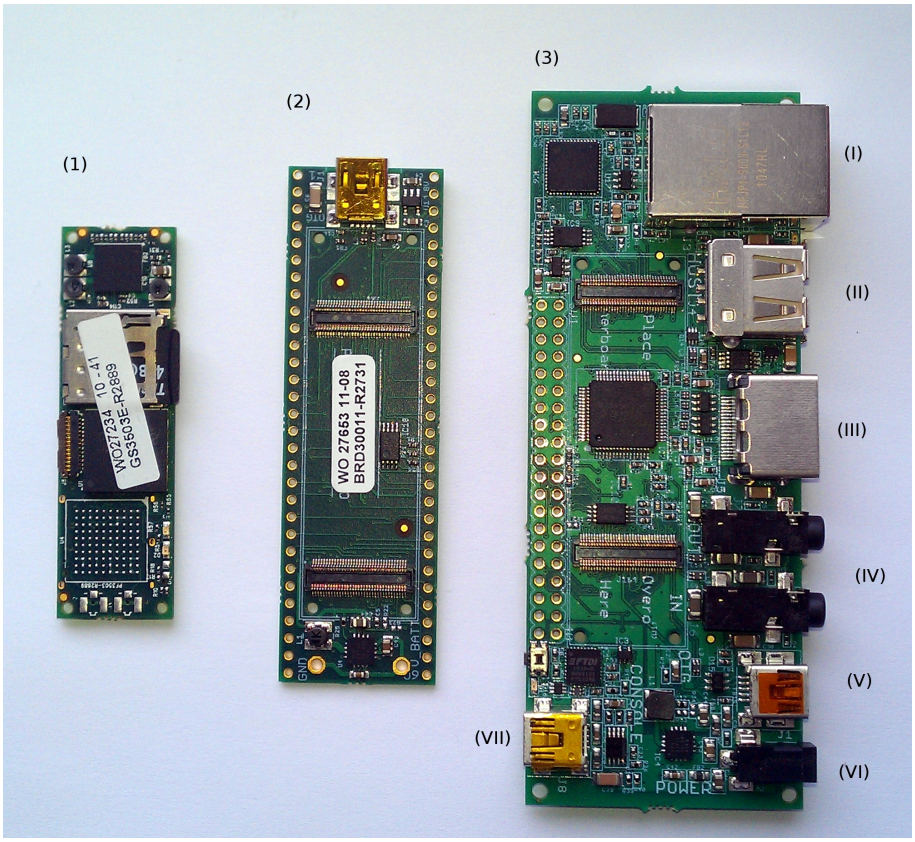

Figure 4.2: Overo Earth COM, Pinto-TH and Tobi extension boards

The Pinto-TH expansion board[22] features a USB mini-AB port with OTG signals (not a console port) and 5V power pins. As shown in the figure 4.3, this board is the one we used for our final prototype, since it is much smaller and lighter. Its dimensions are tiny, 76.2mm x 23mm and only weights 6.6 grams. It is powered by a 6Ah Polymer Lithium Ion Battery and charged with a Polymer Lithium Ion battery charger[23] that permits powering the board while charging the batteries.

Although the Tobi board provided Ethernet connectivity we used a Bandluxe™ C170 3G USB Modem[24] for data communication, it connected to the board with a USB Hub. Since there is no console port in the Pinto-TH, we connected to it with ssh, for that we implemented a script that provided the IP address of the board.

[22]http://www.gumstix.com/store/product_info.php?products_id=239
[23]https://www.sparkfun.com/products/726
[24]http://www.bandrich.com/Data-Card_C170.html

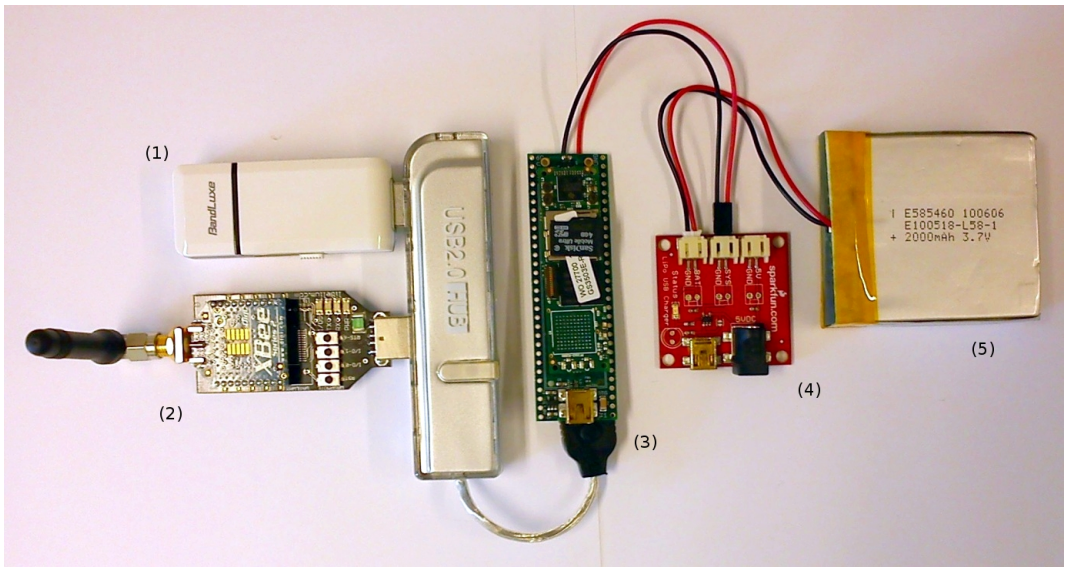

Figure 4.3: A Proxy Node with 3G USB Modem (1), XBee ZB transceiver [2], Pinto-TH board with Overo Earth module (3), LiPoly Charger (4) and batteries (5)

The Proxy Node had the same hardware as the Wide-Area Node with the exception that it included a transceiver to communicate with the WPAN. Our transceiver was a Waspmote Gateway included in the Libelium Developers Kit. This XBee ZB transceiver is mounted on a USB dongle.

For node management we used a Java version of SNMP, SNMP4J[25]. For CoAP we used the CoAP implementation JCoAP[26]since it is the most active implementation at the moment and supports the basic setups we aim at in our prototype implementation.

Cryptographic operations were implemented using BouncyCastle's [27] lightweight cryptographic Application Programming Interface (API). Peers in the overlay used self-signed X.509 certificates. RSA is used as the public key algorithm. The RSA key length is 1024 bits. Secure Hash Algorithm (SHA-1) with RSA encryption is used as the signature algorithm. Bookkeeping Security was not implemented.

Messages in our overlay are exchanged over UDP. Use of a secure transport protocol such as Transport Layer Security (TLS) or Datagram Transport Layer Security (DTLS) was not possible due to the following reasons. First, our Java

[25]http://www.snmp4j.org/
[26]https://github.com/dapaulid/JCoAP
[27]http://www.bouncycastle.org/

does not include a DTLS implementation. Second, although we could have used a client-side TLS implementation, our nodes are too constrained to act as TLS servers. This prevented the use of TLS since a node participating in a DHT overlay needs to act as both a TLS client and server.

4.2 Prototype Architecture

Our prototype was implemented following the design laid in chapter 3. The architectures of the WN and PN prototype are similar, we will focus in the PN since it is more elaborated and has more elements. The WN can be understood as a subset of the PN with a different Application Logic.

As shown in Figure 4.4 there were three main parts in the architecture: Management Module, the M2MCE and the Proxy itself. The three parts were implemented separately in Java. To integrate them we used Java's Remote Method Invocation (Java RMI) that performs the object-oriented equivalent of remote procedure calls (RPC) for Inter Process Communication (IPC).

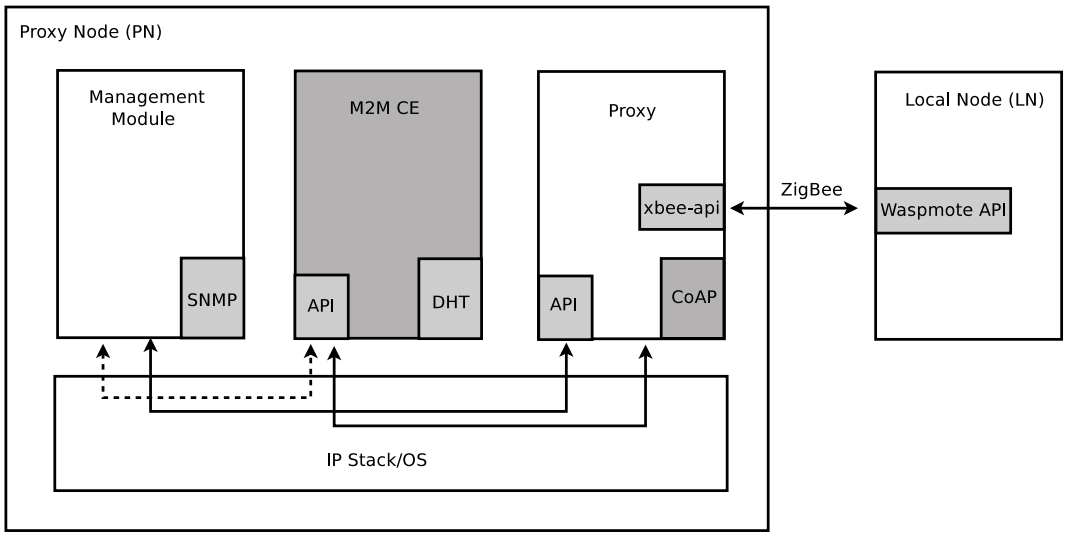

Figure 4.4: Prototype Scenario

When the WN starts up, it loads a configuration file with certain M2MCE parameters specific for that particular node, such as: Java RMI information, interface, IP and port of the bootstrap peer, CoAP name, Chord configuration delays, number of successors, predecessors, fingers and so on. In the case of the PN, the configuration file also contains information about the XBee

coordinator and the proxy itself, such as baudrate for serial communication and XBee port to be used.

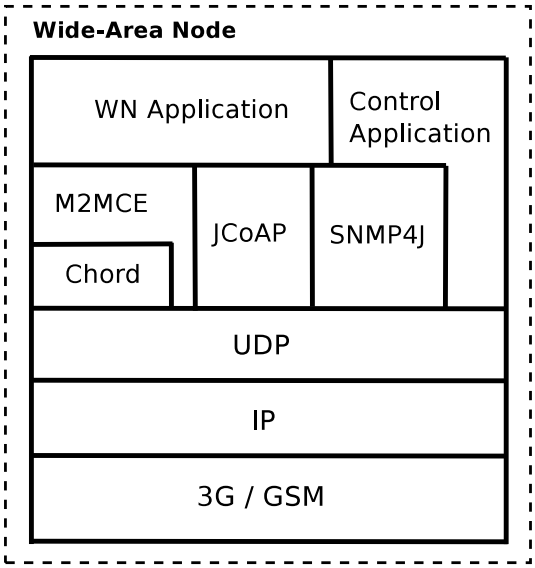

Figure 4.5: Wide Area Node as implemented in our scenario

The application logic in the WN (See figure 4.5) will call the M2MCE to join the Overlay by contacting the Bootstrap Peer. Once it is part of the Overlay it will store the bookkeeping and addressing information in it. The CoAP name specified in the configuration file will be used for addressing the node. At this point the WN will also store the information of the sensors or actuators attached to itself. In the case of the PN it will announce WPAN nodes to the overlay. Each node instance will also launch a CoAP server listening on port 5683 for requests from other WNs or PNs.

Once the node is in the overlay, it can be addressed by using the CoAP Name Registration Service and querying for the coap name of the node. The CoAP name will be the host field of the CoAP uri. For instance, *"coap:" "//" host [":" port] path-abempty ["?" query]*. That overlay query will return the IP address of the Gumstix device.

If the Gumstix PN has some Libelium Waspmotes connected, it can offer resource discovery by starting a CoAP session and sending to it the GET message *"coap://example.net/.well-known/core"* as defined in [65]. That query will return the set of Waspmotes, their availability and the sensor information they offer.

Another possibility is to send a CoAP message Request to an actuator, either in the WN, PN or LN indicating that a *"Button pressed"*. Upon receiving

this message the Actuator will perform a predefined action, in our case it is a blinking LED.

Towards the MCN, the Bookkeeping mechanism will show in the terminal the list of nodes available in the overlay in the moment of the query. So far the information is very simple, if we lookup for *"list_nodes"* it will return the node name, IP, joining time, leaving time, number of sensors and whether it is a database node. In the case of the sensor it will return the name, the type, join and leave timestamps and latest stored value. A sample result would be:

```
Gumstix1 | 192.168.0.1 | 1311608824 | 0 | 3 | F
        Waspmote1 | temperature | 131167613 | 0 | 35
        Waspmote2 | humidity | 1311608121 | 1311923675 | 15
        Waspmote3 | temperature | 1311608222 | 0 | 35
Gumstix2 | 192.168.0.2 | 1311609071 | 0 | 2 | T
        Waspmote4 | pressure | 1311608222 | 0 | 1002
        Waspmote5 | temperature | 1311608222 | 0 | 22
Gumstix3 | 192.168.0.3 | 1311609091 | 1311609394 | 0 | F
Gumstix4 | 192.168.0.4 | 1311609071 | 0 | 0 | T
```

Chapter 5

Conclusions and Future Work

> *"The machine does not isolate man from the great problems of nature but plunges him more deeply into them."*
>
> Antoine de Saint-Exupery

This work was a part of the Devices and Interoperability EcosysteM (DIEM) program [1], sponsored by the Finnish Funding Agency for Technology and Innovation (TEKES) [2] and done in collaboration with the ICT Future Internet SHOK program [3]

The project is a joint work to study how to move the intelligence of an M2M system from a centralized server to the network edge. Our solution consists on a DHT based M2M System. The project was divided in three different thesis topics that related to the different devices in the network: one focuses on the Proxy Node and Sensor Network; another on the Monitoring and Controlling Node and system management; and finally this thesis focuses on the Wide-Area Nodes and the DHT to allow the Distributed Intelligence.

This chapter summarizes the main achievements in this thesis. We present the main conclusions and suggest some directions for future work.

5.1 Conclusions

In this thesis we propose a new architecture for wide area sensor and actuator networking. The architecture binds together WSN and P2P protocols to pro-

[1] http://www.diem.fi/programme
[2] http://www.tekes.fi/en/
[3] www.futureinternet.fi

vide a set of distributed WSNs. For this we have designed and implemented the set of nodes that will be responsible for such communication, as well as studied different use cases for them. Advantages of this architecture are decentralization (except for joining procedure), scalability, self-organization and robustness.

The work progressed in several steps, starting with studying the state of the art in machine to machine communications, sensor and peer-to-peer technologies. An important decision was to focus on the different protocols for our prototype, namely Chord, CoAP and Zigbee.

Since sensor networks tend to be application specific, a set of scenarios was envisaged for our prototype. Development and implementation of the different parts of the network was done following this use cases while at the same time, trying to make the design as versatile as possible. This was successfully achieved by relying on the application layer to provide most of the final functionality while leaving network, link and physical layers generic.

Each of the nodes we have designed was necessary. A PN was key since current sensor nodes (LNs) are too constrained to run our DHT and do not implement an IP stack. Moreover, a PN provides additional caching and security in the network. In prevision for future improvement of sensor capabilities, thus becoming less constrained, the WN was implemented, adding the possibility of attaching sensors and actuators to it. A MCN was required in order to monitor and access the resources in the WSN federated by the DHT and to create associations between nodes.

CoAP was chosen to provide a common namespace for resources in the network. SNMP was chosen to access resources in the network and to provide MCN functionality.

Different caching mechanisms have been proposed. First at the PN level, to reduce the number of queries to the LNs, then at the DHT level for load balancing and persistence. Caching at the DHT also improves system reliability and prevents a single node from over-quering. Security has also been addressed both in the WN and the M2MCE design. Moreover, a bookkeeping mechanism has been proposed to improve system reliability and enable functionality provided by the MCN.

Based on such design the nodes have been implemented using Gumstix and Waspmote sensor devices as hardware. Different Open Source solutions have been used and others have been implemented. We have successfully tested our implementation in different scenarios and confirmed its basic functionality.

5.2 Future Work

We have written a paper that we expect to publish in the following months. In *"Using RELOAD and CoAP for Wide Area Sensor and Actuator Networks"* [58], we evaluate the performance of our scheme. In terms of amount of traffic and volume of inter-device communication (i.e. number of CoAP observation relationships and frequency of CoAP notifications) a client/server performs better than the P2P approach. In this paper we show that the general advantages of P2P (scalability, robustness and so on) become visible as the number of observation relationships grow. Thus P2P is recommended in large scale deployments rather than small ones where a client/server approach is more efficient. In this paper we also propose a CoAP usage for RELOAD similar to the one used in our design.

One of the goals of this project is to make devices as product-like as possible. For that we have started testing under real conditions and we have started designing cases for the different devices. Not only the casing but the devices themselves are also a matter of improvement. So far we have reduced their dimensions to some centimeters, but we believe that in the near future we might find even smaller wireless sensors in the market. Their cost is currently relatively high, we expect that it will decrease when these smart devices are manufactured and deployed in large numbers. If we want this technology to spread out and become really ubiquitous those two factors, size and cost, are paramount.

We believe that the future for smart objects in terms of interoperability and standardization is IP. Our implementation lacks transparency towards the WSN and relies on the PN for most of the tasks. The future lies not in Zigbee but in using IP in the sensor devices directly.

Challenges in our model are centered in following current standards and making our devices as interoperable as possible. To do so, we will try to build sensor devices based on 6LoWPAN, IPv6 and CoAP. These standards, we believe, are the key to success to the Internet of Things, as the technology is produced by many different parties.

Bibliography

[1] *P2PSIP IETF Working Group*, `http://datatracker.ietf.org/wg/p2psip/charter`, Referenced on 22.05.2011.

[2] *A scalable content-addressable network*, vol. 31, New York, NY, USA, ACM, August 2001.

[3] *Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service*, 2001.

[4] 802.11 2003, *IEEE Standard for Information technology. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2003.

[5] 802.15.4 2003, *IEEE Standard for Information technology. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*, October 2003.

[6] 802.15.1 2005, *IEEE Standard for Information technology. Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)*, 2005.

[7] 802.15.4 2006, *IEEE Standard for Information technology. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)*, 2006.

[8] ABB, *Solutions for the water cycle Leakage management Water Leakage*, 2009.

[9] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, *Wireless sensor networks: a survey*, Computer networks **38** (2002), no. 4, 393–422.

[10] Cesare Alippi, Romolo Camplani, Cristian Galperti, and Manuel Roveri, *A Robust, Adaptive, Solar-Powered WSN Framework for Aquatic Environmental Monitoring*, IEEE Sensors Journal **11** (2011), no. 1, 45–55.

[11] Luigi Atzori, Antonio Iera, and Giacomo Morabito, *The Internet of Things: A survey*, Computer Networks **54** (2010), 2787–2805.

[12] S. Baset and H. Schulzrinne, *Peer-to-peer protocol (p2pp)*, March 2007, Work in progress.

[13] U. Blumenthal and B. Wijnen, *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*, RFC 3414 (Standard), December 2002, Updated by RFC 5590.

[14] D. Bryan, P.Matthews, E. Shim, D. Willis, and S. Dawkins, *Concepts and terminology for peer to peer sip*, `http://tools.ietf.org/html/draft-ietf-p2psip-concepts-03`, October 2010, Work in progress.

[15] David A. Bryan, Bruce B. Lowekamp, and Cullen Jennings, *Sosimple: A serverless, standards-based, p2p sip communication system*, Advanced Architectures and Algorithms for Internet Delivery and Applications, 2005. AAA-IDEA 2005. First International Workshop on (2005).

[16] J.D. Case, M. Fedor, M.L. Schoffstall, and J. Davin, *Simple Network Management Protocol (SNMP)*, RFC 1157 (Historic), May 1990.

[17] S. Cheshire and M. Krochmal, *DNS-Based Service Discovery*, Internet-Draft draft-cheshire-dnsext-dns-sd-10, Internet Engineering Task Force, February 2011, Work in progress.

[18] W. Colitti, K. Steenhaut, and N. De Caro, *Integrating wireless sensor networks with the web*, (2011).

[19] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 5280 (Proposed Standard), May 2008.

[20] E. Cooper, A. Johnston, and P. Matthews, *A distributed transport function in p2psip using hip for multi-hop overlay routing*, June 2007, Work in progress.

[21] W.C. Craig, *Zigbee:âĂIJwireless control that simply worksâĂİ*, ZigBee Alliance http://www. zigbee. org/resources/documents/2004_ZigBee_CDC-P810_Craig_Paper. pdf (2004).

[22] P. Van der Stok and K. Lynn, *CoAP Utilization for Building Control*, Internet-Draft draft-vanderstok-core-bc-04, Internet Engineering Task Force, July 2011, Work in progress.

[23] T. Dierks and E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246 (Proposed Standard), August 2008, Updated by RFCs 5746, 5878, 6176.

[24] D. Eastlake 3rd, *Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)*, RFC 4305 (Proposed Standard), December 2005, Obsoleted by RFC 4835.

[25] D. Eastlake 3rd and P. Jones, *US Secure Hash Algorithm 1 (SHA1)*, RFC 3174 (Informational), September 2001, Updated by RFCs 4634, 6234.

[26] L. Eggert, *Congestion Control for the Constrained Application Protocol (CoAP)*, Internet-Draft draft-eggert-core-congestion-control-01, Internet Engineering Task Force, January 2011, Work in progress.

[27] Bob Emmerson, *M2M : the Internet of 50 billion devices*, 2005.

[28] S.C. Ergen, *ZigBee/IEEE 802.15. 4 Summary*, 2004.

[29] Ericsson, *More than 50 billion connected devices, taking connected devices to mass market and profitability*, `http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf`. Accessed April 29, 2011.

[30] Information Society European Commission and Networked Enterprise & RFID Unit (D4) Media, *Internet of things: an early reality of the future internet*, May 2009.

[31] Zhang Fan, Li Wenfeng, Jens Eliasson, Laurynas Riliskis, and H. Makitaavola, *TinyMulle: A Low-Power Platform for Demanding WSN Applications*, Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on, IEEE, 2010, pp. 1–5.

[32] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616 (Draft Standard), June 1999, Updated by RFCs 2817, 5785, 6266.

[33] R.T. Fielding, *Architectural styles and the design of network-based software architectures*, Ph.D. thesis, Citeseer, 2000.

[34] S. Frankel and S. Krishnan, *IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap*, RFC 6071 (Informational), February 2011.

[35] J. Galvin and K. McCloghrie, *Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)*, RFC 1446 (Historic), April 1993.

[36] J. Galvin, K. McCloghrie, and J. Davin, *SNMP Security Protocols*, RFC 1352 (Historic), July 1992.

[37] S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suciu, *What can databases do for peer-to-peer*, WebDB Workshop on Databases and the Web, Citeseer, 2001.

[38] Saikat Guha, Neil Daswani, and Ravi Jain, *An Experimental Study of the Skype Peer-to-Peer VoIP System*, IPTPS'06: The 5th International Workshop on Peer-to-Peer Systems, Microsoft Research.

[39] D. Guinard and V. Trifa, *Towards the web of things: Web mashups for embedded devices*, Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain, Citeseer, 2009.

[40] K. Hartke and Z. Shelby, *Observing Resources in CoAP*, Internet-Draft draft-ietf-core-observe-02, Internet Engineering Task Force, March 2011, Work in progress.

[41] INFSO D.4 Networked Enterprise & RFID INFSO G.2 Micro & Nanosystems in cooperation with the Working Group RFID of the ETP EPOSS, *Internet of things in 2020, roadmap for the future*, May 2008.

[42] C. Jennings, B. Lowekamp, E. EricRescorla, S. Baset, and H. Schulzrinne, *A SIP Usage for RELOAD*, Internet-Draft draft-ietf-p2psip-sip-06, Internet Engineering Task Force, July 2011, Work in progress.

[43] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, *Resource location and discovery (reload) base protocol*, `http://tools.ietf.org/html/draft-ietf-p2psip-base-17`, July 2011, Work in progress.

[44] C. Jennings, J. Rosenberg, and E. Rescorla, *Address settlement by peer to peer*, July 2007, Work in progress.

[45] XingFeng. Jiang, HeWen. Zheng, C. Macian, and V. Pascual, *Service extensible p2p peer protocol*, February 2008, Work in progress.

[46] J. Jonsson and B. Kaliski, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*, RFC 3447 (Informational), February 2003.

[47] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, et al., *People, places, things: Web presence for the real world*, Mobile Networks and Applications **7** (2002), no. 5, 365–376.

[48] A. Knauf, G. Hege, T. Schmidt, and M. Waehlisch, *A Usage for Shared Resources in RELOAD (ShaRe)*, Internet-Draft draft-knauf-p2psip-share-01, Internet Engineering Task Force, July 2011, Work in progress.

[49] G Kortuem, F Kawsar, V Sundramoorthy, and Fitton, *Smart objects as building blocks for the internet of things.*, IEEE Internet Computing (2010).

[50] Matthias Kovatsch, Markus Weiss, and Dominique Guinard, *Embedding internet technology for home automation*, Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2010) (Bilbao, Spain), September 2010.

[51] B. Krishnamachari and C.S. Raghavendra, *Performance evaluation of the IEEE 802.15.4 MAC for low-rate low-power wireless networks*, IEEE International Conference on Performance, Computing, and Communications, 2004 (2004), 701–706.

[52] A.D. Kshemkalyani and M. Singhal, *Distributed computing: principles, algorithms, and systems*, Cambridge Univ Pr, 2008.

[53] N. Kushalnagar, G. Montenegro, and C. Schumacher, *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*, RFC 4919 (Informational), August 2007.

[54] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, and M. Karir, *Atlas internet observatory 2009 annual report*, Arbor Networks, the University of Michigan and Merit Network, Tech. Rep (2009).

[55] D. Levi, P. Meyer, and B. Stewart, *Simple Network Management Protocol (SNMP) Applications*, RFC 3413 (Standard), December 2002.

[56] Gunnar Lind, *Variable speed limits at intersections in Sweden*, European Commission, Directorate General Energy and Transport (2007), 0–12.

[57] J. Maaenpa and J.J. Bolonio, *Performance of resource location and discovery (reload) on mobile phones*, Wireless Communications and Networking Conference (WCNC), 2010 IEEE, april 2010, pp. 1 –6.

[58] J. Maaenpa, J.J. Bolonio, and S. Loreto, *Using reload and coap for wide area sensor and actuator networking*, 2011, (Accepted for publication).

[59] J. Maenpaa and G. Camarillo, *A study on maintenance operations in a chordbased peer-to-peer session initiation protocol overlay network*, May 2009.

[60] J. Maenpaa, G. Camarillo, and J. Hautakorpi, *A self-tuning distributed hash table (dht) for resource location and discovery (reload)*, February 2009, Work in progress.

[61] E. Marocco and E. Ivov, *Extensible peer protocol (xpp)*, June 2007, Work in progress.

[62] K. McCloghrie, *An Administrative Infrastructure for SNMPv2*, RFC 1909 (Historic), February 1996.

[63] Steve Meloan, *Toward a global internet of things*, `http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/`. Accessed April 29, 2011.

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

[65] M. Nottingham and E. Hammer-Lahav, *Defining Well-Known Uniform Resource Identifiers (URIs)*, RFC 5785 (Proposed Standard), April 2010.

[66] City of Sequim, *Water system map*, `http://www.ci.sequim.wa.us/planning/Maps/Water%20System%20Map.pdf`. Accessed February 5, 2011.

[67] ZigBee Standards Organization, *Zigbee specification*, January 2008.

[68] Y. Peng, W. Wang, Z. Hao, and Y. Meng, *An SNMP Usage for RELOAD*, Internet-Draft draft-peng-p2psip-snmp-02, Internet Engineering Task Force, July 2011, Work in progress.

[69] Y. Peng, W. Wang, J. JinPeng, L. Le, Z. Hao, and Y. Meng, *Network Management Scenarios for RELOAD*, Internet-Draft draft-peng-p2psip-network-management-scenarios-02, Internet Engineering Task Force, March 2011, Work in progress.

[70] M. Petit-Huguenin, *Configuration of Access Control Policy in REsource LOcation AndDiscovery (RELOAD) Base Protocol*, Internet-Draft draft-petithuguenin-p2psip-access-control-03, Internet Engineering Task Force, July 2011, Work in progress.

[71] Jan S. Rellermeyer, Michael Duller, Ken Gilmer, Damianos Maragkos, Dimitrios Papageorgiou, and Gustavo Alonso, *The software fabric for the internet of things.*, IOT (Christian Floerkemeier, Marc Langheinrich, Elgar Fleisch, Friedemann Mattern, and Sanjay E. Sarma, eds.), Springer, 2008, pp. 87–104.

[72] E. Rescorla and N. Modadugu, *Datagram Transport Layer Security*, RFC 4347 (Proposed Standard), April 2006, Updated by RFC 5746.

[73] J. Rosenberg, *Interactive connectivity establishment (ice): A protocol for network address translator (nat) traversal for offer answer protocols*, October 2007, Work in progress.

[74] J. Rosenberg, R. Mahy, and P. Matthews, *Traversal using relays around nat (turn): Relay extensions to session traversal utilities for nat (stun)*, November 2008, Work in progress.

[75] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *SIP: Session Initiation Protocol*, RFC 3261 (Proposed Standard), June 2002, Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141.

[76] Peter Schick, *Einfluss von Streckenbeeinflussungsanlagen auf die Kapazitat von Autobahnabschnitten sowie die Stabilitat des Verkehrsflusses*, Inst. fur Strass en und Verkehrswesen, Univ. Stuttgart, 2003.

[77] Detlef Schoder, Kai Fischbach, and Christian Schmitt, *Core concepts in peer-to-peer networking*, Peer-to-peer computing: the evolution of a disruptive technology **296** (2005), 1–27.

[78] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, *Constrained Application Protocol (CoAP)*, Internet-Draft draft-ietf-core-coap-07, Internet Engineering Task Force, July 2011, Work in progress.

[79] Zach Shelby and Carsten Bormann, *6lowpan: The wireless embedded internet*, Wiley Publishing, 2010.

[80] X. Shen and Z. Li, *Implementing database management system in p2p networks*, 2008 International Seminar on Future Information Technology and Management Engineering, IEEE, 2008, pp. 528–532.

[81] Sven Siorpaes, Gregor Broll, Massimo Paolucci, Enrico Rukzio, John Hamard, Matthias Wagner, Albrecht Schmidt, and Docomo Eurolabs, *Mobile interaction with the internet of things*, In Adjunct Proc. of Pervasive 2006 Late Breaking Results, 2006.

[82] Libelium Comunicaciones Distribuidas S.L., *Waspmote datasheet*, `http://www.libelium.com/documentation/waspmote/waspmote-datasheet_eng.pdf`. Accessed April 22, 2011.

[83] ———, *Waspmote technical guide*, `http://www.libelium.com/documentation/waspmote/waspmote-technical_guide_eng.pdf`. Accessed April 22, 2011.

[84] W. Stallings, *Snmpv3: A security enhancement for snmp*, Communications Surveys Tutorials, IEEE **1** (1998), no. 1, 2 –17.

[85] V. Stirbu, *Towards a restful plug and play experience in the web of things*, Semantic computing, 2008 IEEE international conference on, IEEE, 2008, pp. 512–517.

[86] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications*, ACM SIGCOMM Computer Communication Review **31** (2001), no. 4, 149–160.

[87] Daozong Sun, Sheng Jiang, Weixing Wang, and Jingchi Tang, *WSN Design and Implementation in a Tea Plantation for Drought Monitoring*, 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (2010), 156–159.

[88] G. Tolle, *A UDP/IP Adaptation of the ZigBee Application Protocol*, draft-tolle-cap-00 (Informational), october 2008.

[89] Jean-Philippe Vasseur and Adam Dunkels, *Interconnecting smart objects with IP - the next Internet*, Morgan Kaufmann, 2010.

[90] G. Waters, *User-based Security Model for SNMPv2*, RFC 1910 (Historic), February 1996.

[91] Evan Welbourne, Leilani Battle, Garret Cole, Kayla Gould, Kyle Rector, Samuel Raymer, Magdalena Balazinska, and Gaetano Borriello, *Building the internet of things using rfid*, IEEE Internet Computing **13** (2009).