A Proxy for Distributed Hash Table based Machine-to-Machine Networks

DAOYUAN LI

KTH Information and
Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

Aalto University
School of Science
Degree Programme in Security and Mobile Computing

Daoyuan Li

A Proxy for Distributed Hash Table based Machine-to-Machine Networks

Master's Thesis
Espoo, June 29, 2011

Supervisors: Professor Gerald Q. Maguire Jr., Royal Institute of Technology (KTH)
Professor Antti Ylä-Jääski, Aalto University
Instructor: Jani Hautakorpi, PhD, Ericsson Research NomadicLab, Finland

Aalto University
School of Science
Degree Programme of Security and Mobile Computing

ABSTRACT OF
MASTER'S THESIS

Author: Daoyuan Li

Title:

A Proxy for Distributed Hash Table based Machine-to-Machine Networks

Date: June 29, 2011 **Pages:** 14 + 74

Professorship: Data Communications Software **Code:** T-110

Supervisors: Professor Gerald Q. Maguire Jr.
Professor Antti Ylä-Jääski

Instructor: Jani Hautakorpi, PhD

Wireless sensor networks (WSNs) have been an increasingly interest for both researchers and entrepreneurs. As WSN technologies gradually matured and more and more use is reported, we find that most of current WSNs are still designed only for specific purposes. For example, one WSN may be used to gather information from a field and the collected data is not shared with other parties.

We propose a distributed hash table (DHT) based machine-to-machine (M2M) system for connecting different WSNs together in order to fully utilize information collected from currently available WSNs. This thesis specifically looks at how to design and implement a proxy for such a system. We discuss why such a proxy can be useful for DHT-based M2M systems, what the proxy should consist of, and what kind of architecture is suitable. We also look into different communication protocols that can be used in these systems and discuss which ones best suit our purposes. The design of the proxy focuses on network management and service discovery of WSNs, and security considerations as well as caching mechanisms in order to improve performance. A prototype is implemented based on our design and evaluated. We find it feasible to implement such a DHT-based M2M system and a proxy in the system can be necessary and useful. Finally, we draw conclusions and discuss what future work remains to be done.

Keywords: M2M, P2P, DHT, Proxy, Gateway, ZigBee, 6LoWPAN, CoAP, Cache

Language: English

Sammanfattning

Trådlösa sensornätverk (WSN) har en allt större intresse för både forskare och företagare. Som WSN teknik successivt mognat och allt fler använder rapporteras, finner vi att de flesta av dagens WSN fortfarande är konstruerade enbart för särskilda ändamål. Till exempel kan en WSN användas för att samla in information från ett fält och de insamlade data inte delas med andra parter.

Vi föreslår en distribuerad hashtabell (DHT) baserad maskin-till-maskin (M2M) system för att koppla olika WSN tillsammans för att fullt ut utnyttja information som samlats in från tillgängliga WSN. Denna avhandling tittar särskilt på hur man kan utforma och genomföra en fullmakt för ett sådant system. Vi diskuterar varför en proxy kan vara användbart för DHT-baserade M2M system, vad proxy bör bestå av, och vilken typ av arkitektur är lämplig.

Vi tittar också på olika kommunikationsprotokoll som kan användas i dessa system och diskuterar vilka som bäst passar våra syften. Utformningen av proxy fokuserar på nätverksadministration och service upptäckten av WSN, och med hänsyn till säkerheten samt cachning mekanismer för att förbättra prestanda. En prototyp genomfördes baserat på vår design och utvärderas. Vi tycker att det är möjligt att genomföra en sådan DHT-baserade M2M system och en proxy i systemet kan vara nödvändig och nyttig. Slutligen drar vi slutsatser och diskutera vad framtida arbete återstår att göra.

Nyckelord: M2M, P2P, DHT, Proxy, Gateway, ZigBee, 6LoWPAN, CoAP, Cache

Acknowledgements

This thesis would not have been possible without the help of several individuals who in one way or another offered their valuable and generous assistance in the preparation and completion of this study.

First and foremost, I want to thank my supervisors, Professor Gerald Q. Maguire Jr. at Royal Institute of Technology and Professor Antti Ylä-Jääski at Aalto University, for their guidance through this thesis project, especially Professor Maguire, who has given me extremely helpful and concrete comments on my thesis drafts.

Secondly, I would like to express my gratitude to people at Ericsson Research NomadicLab. My industrial advisor Dr. Jani Hautakorpi has given me valuable instructions in the design and implementation of the system, as well as in thesis writing. Section manager of LMF/TRM, Jouni Mäenpää warmly welcomed me to the lab and generously offered his help during my stay in the lab. My colleagues have offered many useful suggestions regarding the thesis. I have had very interesting discussions with Jaime Jiménez Bolonio and Nalin Gupta; those discussions have been of great help for my thesis. Rasib Hassan Khan and Gaëtan Charmette have been nice companies, especially during lunch times when we talk and share stories and anecdotes. Their cheerful spirit has kept me motivated and enthusiastic.

Last but not least, I would like to thank my parents, whose kindness, diligence, and positive attitude towards life have deeply influenced me and have been an invaluable influence throughout my life.

Jorvas, Kirkkonummi, Finland
June 22, 2011

Daoyuan Li

Contents

List of Tables

List of Figures

Abbreviations and Acronyms

3G	3rd Generation mobile telecommunications
6LoWPAN	IPv6 over Low-power Wireless Personal Area Networks
ACK	Acknowledgment
AODV	Ad hoc On-demand Distance Vector
APDU	Application layer Protocol Data Unit
API	Application Programming Interface
APS	Application support sub-layer
APSDE	Application support sub-layer data service entity
APSME	Application support sub-layer management service entity
CoAP	Constrained Application Protocol
CRC	Cyclic Redundancy Check
CoRE	Constrained RESTful Environments working group
CRUD	Create, Read, Update and Delete
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DHT	Distributed Hash Table
DDNS	Distributed Domain Name Service
DNS	Domain Name Service
DVI	Digital Visual Interface
EEPROM	Electrically Erasable Programmable Read-Only Memory
ERP	Enterprise Resource Planning
FFD	Full-Function Device
GAF	Geographic Adaptive Fidelity
GEAR	Geographic and Energy-Aware Routing
GPRS	General Packet Radio Service
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol

ICT	Information and Communication Technologies
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPC	Inter-Process Communication
LEACH	Low-energy adaptive clustering hierarchy
LLC	Logical Link Control
LR-WPAN	Low-Rate Wireless Personal Network
LoWPAN	Low-power Wireless Personal Network
M2M	Machine-to-Machine
M2M CE	Machine-to-Machine Communication Enabler
MAC	Medium Access Control
MCN	Monitoring and Control Node
MD5	Message-Digest algorithm 5
MECN	Mminimum Energy Communication Network
MIC	Message Integration Code
OS	Operating System
OSI	Open Systems Interconnection
OTAP	Over The Air Programming
P2P	Peer-to-Peer
PAN	Personal Area Network
PC	Personal Computer
PDU	Protocol Data Unit
PEGASIS	Power-efficient GAthering in Sensor Information Systems
PHY	Physical Layer
QoS	Quality of Service
REST	Representational State Transfer
RFD	Reduced-Function Device
RFID	Radio-Frequency IDentification
RISC	Reduced Instruction Set Computing
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RTT	Round Trip Time
SAR	Sequential Assignment Routing
SCM	Supply Chain Management
SHA	Secure Hash Algorithm
SMECN	Small Minimum Energy Communication Network
SNMP	Simple Network Management Protocol

SPIN	Sensor Protocols for Information via Negotiation
SRAM	Static Random-Access Memory
TEEN	Threshold sensitive Energy Efficient sensor Network protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
URI	Universal Resource Indicator
USB	Universal Serial Bus
WAN	Wide Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
WWAN	Wireless Wide Area Network
ZDO	ZigBee Device Object

Chapter 1

Introduction

1.1 Overview

Wireless sensor networks (WSNs) have been of increasingly interest for both researchers and entrepreneurs. While WSN technologies have gradually matured and more and more uses of WSNs have been reported, still most of current WSNs are designed only for specific purposes. For example, one WSN may be used to gather information from a field, but the collected data is not immediately available to other parties.

As the concept of Internet of Things (IoT) and Machine-to-Machine (M2M) communications have developed, more and more scenarios have been suggested for them. However, current M2M systems may not fit into scenarios where a large number of WSNs and actuator nodes are required to communicate with each other; for example, a smart traffic control scenario where sensors monitor traffic volume and road condition in one place and may need to communicate with actuators several kilometers away. Who makes the decisions to change traffic lights based on the information collected by those sensors? As a result, we propose a distributed hash table (DHT) based M2M system for interconnecting different M2M networks in order to make full use of information collected from currently available WSNs.

In the rest of this chapter we first introduce the problem we are trying to tackle. Next we list the expected contributions of this thesis project. Finally we describe how this thesis is organized.

1.2 Problem Description

Current M2M networks are often used to capture events and translate them into human intelligible information. For example, WSNs are used to gather information within a specific area. These networks usually have a hierarchical or mesh topology. The sensor nodes are organized into clusters; the nodes generally communicate at low bit rate and strive for low power consumption. Low-Rate Wireless Personal Area Network (LR-WPAN) protocols, such as IEEE 802.15.4 (see Section 2.4), are usually used in these scenarios. M2M networks are good for gathering information, but maybe not be sufficient for the case of several M2M networks needing to exchange information with each other, when centralized servers are not available to control the exchange of information.

In this thesis project we will connect M2M networks to a wide area network using a Peer-to-Peer (P2P) overlay. Additionally, the nodes in the system are not only sensors, but could also be actuators (a given node might even support both functions at the same time). The nodes share information collected from environments around them over a wide area network (WAN) in order to make independent decisions and perform operations on their environment. This P2P network consists of proxies complemented by traditional WSNs that forward data to these proxies. The P2P network will be implemented using DHT technology.

For this project, we will design, implement, and evaluate a proxy implementing DHT-based M2M communication. Such a proxy should make it possible to incorporate inexpensive sensors/actuators as part of the DHT network. The proxy nodes are 3G-enabled sensors/actuators. In practice, each proxy is implemented by adding logic (software) to both cheap LR-WPAN sensors and/or 3G-enabled proxy sensors/actuators.

1.3 Contributions

The contributions of this thesis include use of DHT in M2M systems in order to increase scalability, a justification for introducing a proxy in DHT-based M2M systems, a design for such a proxy, a working prototype with minimum functionality, and an evaluation of this prototype.

1.4 Thesis Organization

Chapter 2 discusses related concepts that the user will find useful when reading the remainder of the thesis. We introduce basic terms and technologies, including Internet of Things (Section 2.1), Machine-to-Machine communication (Section 2.2), Wireless Sensor Networks (Section 2.3), Low-Rate Wireless Personal Area Network protocols (Section 2.4), and Distributed Hash Tables (Section 2.5).

In Chapter 3 we describe the design decisions concerning our proxy. Specifically we will consider aspects of wireless personal area network(WPAN)'s node management and service management.We also discuss caching mechanisms used in the proxy and security related issues.

Chapter 4 describes the implementation of the proxy, including specific technologies and techniques we used in the implementation.

Chapter 5 analyzes the prototype proxy implemented in this thesis project and discusses performance considerations that should be taken into account.

We state our conclusions in Chapter 6. We summarize what has been done and our results. The thesis closes with a discussion about what future improvements are needed or might be needed.

Chapter 2

Background

2.1 The Internet of Things

According to "Internet of things in 2020: Roadmap for the future" [1], the Internet of Things (IoT) is defined as "things having identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environmental, and user contexts"; Semantically, IoT means "a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols" [1]. IoT focuses on interconnecting various devices (big or small in size, smart or dumb from the perspective of information processing, mobile or stationary) together to a large area network (typically running on IP).

IoT is a new concept that is becoming more and more popular in the field of (wireless) communications. The basic idea of this paradigm is the pervasive presence of computational resources around us, which are able to interact and cooperate with each other, through unique identification schemes and agreed communication protocols, in order to perform certain common tasks together [2]. The IoT is expected to have a even higher level of device heterogeneity than the current Internet has. Devices such as Radio-Frequency IDentification (RFID) [3] tag readers, sensors, actuators, mobile phones, and so on, are expected to be "things" in the IoT. According to Robin Duke-Woolley: "Within those we look at the individual devices that could be connected and could be of value, and we currently track over 300 different device types being used" [4].

It is foreseeable that IoT will have a strong impact on several aspects of people's daily life. Thanks to the increasing computational power, decreas-

ing size, and increased energy efficiency of the devices, and their interconnectivity and interoperability, the IoT will play a role in numerous scenarios. From the perspective of a private person, the IoT can be used in both work and home environments. For example, users may employ the IoT for health monitoring and for assisted living. Additionally, from the perspective of businesses, IoT will have a high impact on fields such as industrial manufacturing, automation, logistics, process management, and so on.

IoT will be ubiquitous, penetration into our life even more than the Internet and mobile technologies have. The number of devices that will form the IoT could be huge. According to Ericsson, there will be 50 billion connected devices by 2020 [5]. In comparision, today about 5 billion users are connected to mobile networks worldwide. In other words, we will expect a shift of focus from person-to-person communication to pervasive M2M communication.

Some devices in the IoT will be intelligent and will exhibit behaviors according to predefined routines. Furthermore, some devices will have the ability to collaborate with each other in a more intelligent manner, i.e. they may be able to make decisions based upon their environment by themselves or in collaboration with other devices, instead of human users explicitly controlling them. In addition to acting on their own, devices can also gather information and deliver it to users of applications. For example, a device could send an alarm message to its owner when its battery power level is low in the form of a text message sent to the owner's cellular phone.

One of the most challenging issues in IoT is device power consumption, as we expect a large number of battery powered devices to be connected to the IoT. Due to limitations such as tiny physical size, harsh environment, absence of human intervention, and so on, it is important to efficiently utilize available power resources. Another issue is reliability. We do not want to require devices to be extremely reliable, because ensuring the reliability of an individual device may cost a lot. However, the reliability of a group of devices trying to accomplish a common task can be much higher than their individual reliabilities. To achieve this, the devices in the IoT should be adaptive to failures of others and be able to self-configure. Furthermore, since the number of devices in the IoT will be large, individually configuring devices will simply not be feasible in practice.

Despite the excitement of imagining the wonderful vision of the IoT, we need to tackle the two challenges mentioned above before this vision can be realized. Mellor [6] expects that it will take a while before the M2M communications and wireless sensor networks are deployed pervasively. While there are existing M2M technologies, they are still under development and do not

work well with each other. We will explore more about these two challenges in the following two sections.

2.2 Machine-to-Machine Commmunication

Machine-to-Machine (M2M)[1] commmunication generally refers to communication between machines, as opposed to communication between human beings, or communication between humans and machines. Because of the increasing amount of M2M commmunication, the "internet of things" is emerging as a new paradigm.

M2M communications is based on the idea that rather than having a few stand alone machines, interconnecting machines with each other is more useful [7, 8]. M2M systems combine Information and Communication Technologies (ICT) with smart objects, to provide interaction among systems without human intervention [9]. These automated systems can perform a variety of tasks.

M2M is a broad concept and has many application scenarios [10]. M2M "covers an enormous number of potential devices and applications," according to Robin Duke-Woolley at Harbor Research[2]. Harbor Research regularly monitors eight different markets – buildings, energy, industrial, medical, retail, transportation, security/public safety and consumer/professional – so as to track M2M's progress [4].

A typical M2M system has five basic components: users, objects, network, service platform, and enterprise information system [9]. These are described as:

- *Users* – can be individual persons or other objects that make use of the system.
- *Objects* – such as sensors and actuators that can communicate with internal peers and external peers.
- *Network* – the communication network enables objects to communicate either internally with peers or externally with users. This network may be a wired or wireless network.

[1] M2M is sometimes interpreted as Man-to-Machine, Machine-to-Man, Machine-to-Mobile, or Mobile-to-Machine. In this thesis we only refer Machine-to-Machine as M2M.

[2] `http://www.harborresearch.com/`

- *Service platform* – controls data routing as well as administration of the communicating participants. The service platform provides a middleware layer that can optimize data flow among objects and provides services and interfaces to applications.
- *Enterprise information system* – integrates the M2M solution with enterprise applications such as Enterprise Resource Planning (ERP), Supply Chain Management (SCM), and so on.

2.3 Wireless Sensor Networks

Wireless sensor networks (WSNs) have become increasingly popular in recent years, as both the power consumption of sensor nodes and their cost have decreased [11]. WSNs are widely deployed in many application areas [11, 12], such as industrial control and monitoring, home automation, security and military sensing, inventory management and asset tracking, environment sensing, health monitoring, and so on.

2.3.1 WSN Architecture

In a WSN, sensor nodes are scattered over a sensor field. These nodes gather information and transmit this information to a sink node perhaps via other sensor nodes. As shown in Figure 2.1, the sink acts as a gateway, exchanging

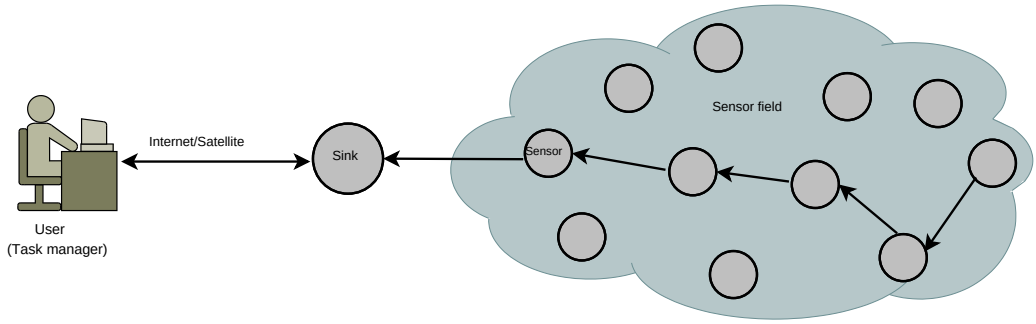

Figure 2.1: A multi-hop wireless sensor network.

data with a task manager node through the Internet or a satellite connection [11]. The user may have some specific requirements for the sensor nodes. The user sends requests to the sink rather than directly to the sensor nodes.

WSNs can be very different from other wireless networks, such as wireless *ad hoc* networks [13]. WSNs have several distinct features as compared with *ad hoc* networks [11]. First of all, in a WSN there are usually a large number of sensors deployed in a field, frequently orders of magnitude more than the number of nodes in an *ad hoc* network. Secondly, unlike *ad hoc* network, WSN nodes are more prone to failure. Thirdly, the topology of WSNs, especially those where the sensor nodes are sparsely deployed [14], is likely to change, as sensor nodes fail. Due to changes in network topology, the sensor nodes have to re-self-organize in order to send data to the sink (gateway). Finally, the energy available to a WSN node is limited, and recharging after deployment is usually very difficult or impossible. As a result of the limitations on node energy power management is a major issue in WSNs.

Due to the above features, WSNs are designed to take these many aspects into account. Generally, WSNs have to be fault tolerant since sensor node failures are common; they should be scalable because the number of nodes in the networks can be very large; and they have to be cost-effective – otherwise there would be little benefit in deploying WSNs; and they must be power-efficient and resource-efficient, due to limitations on the size of the individual sensor nodes and other constraints such as cost and environmental limitations.

2.3.2 WSN Sensor Node Architecture

As shown in Figure 2.2, a sensor is composed of four main units [11]: a sensing

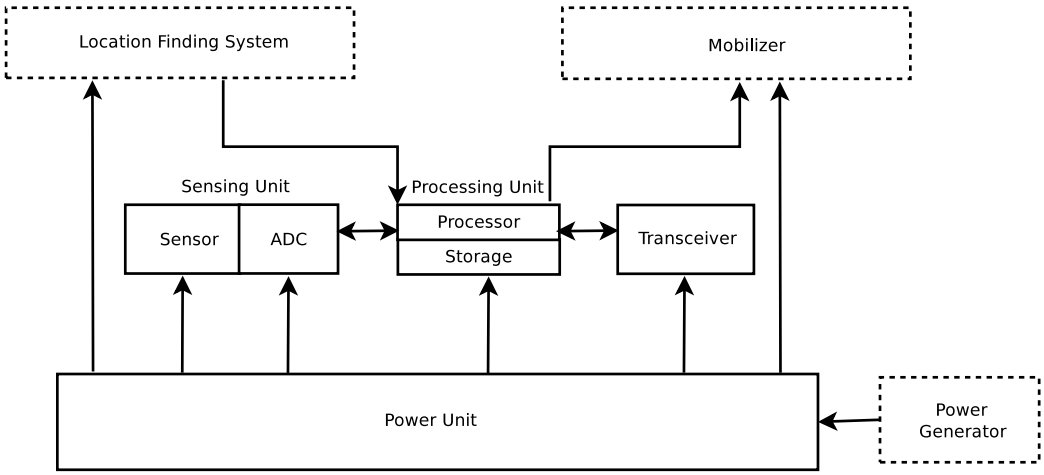

Figure 2.2: The architecture of sensors.

unit for information gathering, a processing unit for processing information, a transceiver for communication with other nodes, and a power unit to supply energy.

Optional modules such as a power generator, location finding system, mobilizer and so on, may be added to this architecture, based on requirements of the actual application. Generally speaking, these modules have to interact with the power unit on a node, i.e., they either provide power to the power unit or consume power provided by the power unit.

2.3.3 Routing in WSNs

Finding the appropriate path to transfer data is very important in WSNs. Both data transfer and the routing protocol should use as little power as possible, since power in sensor nodes is a limited resource. Additionally, the communication processing should not require too much computation or memory, since these are also limited resources, especially if we want to achieve low cost for sensor nodes. As noted previously, the WSN itself should be self-organized. The environment of the sensor field may change, changing both the topology and the set of events that need to be reported. Since generally sensor nodes are left unattended, they must find a valid path from each sensor node to the sink by themselves.

Akkaya and Younis [15] categorize routing protocols in WSNs into the following four categories:

1. Data-centric protocols: They are query-based, thus they depend upon the **name** of the data and the values of these named data. Redundant values of a named data item can be eliminated or aggregated. Mechanisms in this category include flooding and gossiping [16], sensor protocols for information via negotiation (SPIN) [17], directed diffusion [18], etc.

2. Hierarchical protocols: These protocols focus on scalability. In hierarchical routing protocols, network clusters are established and the network is divided into smaller subnetworks, easing management and increasing scalability. Examples of hierarchical routing protocols are low-energy adaptive clustering hierarchy (LEACH) [19], Power-efficient GAthering in Sensor Information Systems (PEGASIS) [20], Threshold sensitive Energy Efficient sensor Network protocol (TEEN) [21], and so on.

3. Location-based protocols: These protocols use knowledge of the physical locations of sensor nodes. They take advantage of location information in an energy efficient way. They may be useful especially when there is no IP-address scheme and nodes are spatially deployed in a region. Protocols falling into this category include minimum energy communication network (MECN) [22], small minimum energy communication network (SMECN) [23], geographic adaptive fidelity (GAF) [24], geographic and energy-aware routing (GEAR) [25], and so on.

4. Network flow and QoS-aware protocols: Some protocols model route setup process as a network flow problem. For example, maximum lifetime energy routing [26] and maximum lifetime data gathering [27] fit into this category. QoS-aware protocols take end-to-end delay requirements into consideration while setting up routes in the network. Sequential assignment routing (SAR) [28] is an example of a QoS-aware protocol.

2.4 Protocols for Wireless Sensor Networks

This section describes WSNs that utilize IEEE 802.15.4 network protocols. We first introduce the standard making community, the IEEE 802.15.4 working group. Next we look at the IEEE 802.15.4 standard, which specifies Low-Rate Wireless Personal Networks (LR-WPANs). After that we look at upper layer protocols that operate over an underlying IEEE 802.15.4 link. We introduce ZigBee™in Section 2.4.3 and 6LoWPAN in Section 2.4.4.

2.4.1 IEEE 802.15 Working Group

The IEEE 802.15 Working Group[3] focuses on the development and standardization of Wireless Personal Area Networks (WPAN)[4], or short distance wireless networks.

This working group is organized into distinct sub-groups. IEEE 802.15 WPAN Task Group 1 (TG1)[5] focuses on standards based on Bluetooth™ technology; while the IEEE 802.15 WPAN Task Group 4 (TG4)[6] is char-

[3] http://www.ieee802.org/15/
[4] http://www.ieee802.org/15/about.html
[5] http://www.ieee802.org/15/pub/TG1.html
[6] http://www.ieee802.org/15/pub/TG4.html

tered to focus on WSNs [29], specifically the LR-WPAN standard for low complexity, low cost, and extremely low-power wireless connectivity.

2.4.2 IEEE Std 802.15.4™

After low data rate technology emerged the IEEE 802.15.4 committee began to work on a low data rate standard. IEEE Std 802.15.4™ is a standard developed and maintained by the IEEE 802.15 WPAN Task Group 4. The latest version of this standard is IEEE Std 802.15.4-2006 [30], which is backward-compatible with IEEE Std 802.15.4-2003 [31]. It specifies the lower layers in the open systems interconnection (OSI) model [32]: the wireless Medium Access Control (MAC) and Physical Layer (PHY) for LR-WPANs, as shown in Figure 2.3. Note that IEEE 802.15.4 operates under the IEEE Logical Link Control (LLC) protocol.

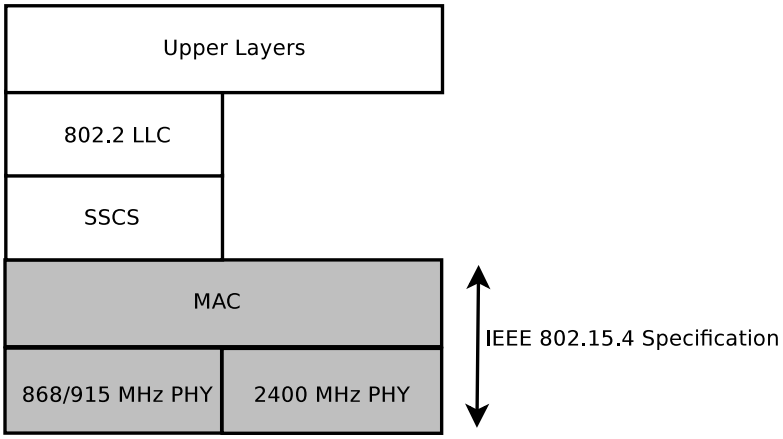

Figure 2.3: IEEE 802.15.4 specification in context.

There may be two different types of devices participating in an IEEE 802.15.4 network: full-function devices (FFDs) and reduced-function devices (RFDs). An FFD may serve as a coordinator in a network or as device; it can communicate with other FFDs and RFDs. In contrast, an RFD may only talk to an FFD. Two or more devices within a personal operating space communicating on the same wireless channel can form a WPAN. IEEE 802.15.4 WPANs support star, tree, cluster tree, and mesh topologies.

IEEE Std 802.15.4-2006 specifies four different data rates for LR-WPANs: 250 kb/s, 100 kb/s, 40 kb/s, and 20 kb/s. Other characteristics of LR-WPANs include star or peer-to-peer topologies; 16-bit short addresses or

64-bit extended address space; carrier sense multiple access with collision avoidance (CSMA/CA) channel access, low power consumption; 16 channels in the 2450 MHz band, 30 channels in the 915 MHz band, and 3 channels in the 868 MHz band; and so on.

The PHY of a device is implemented by a radio transceiver. The PHY layer is responsible for activation and deactivation of the radio transceiver, channel selection, data transmitting and receiving via the physical medium, and so on. The radio operates in unlicensed bands (meaning that the user does not have to have a license to operate the device, but the manufacturer needs to meet the requirements of the regulators), e.g. 868 – 868.6 MHz in Europe, 902 – 928 MHz in North America, and 2400 – 2483.5 MHz world wide [30].

The MAC layer provides addressing and physical channel access for upper layers. Its features include beacon management, channel access, association and disassociation, and so on [30]. It manages all access to the physical radio channel and is responsible for generating network beacons on coordinator devices, synchronizing to network beacons, supporting PAN association and disassociation, employing the CSMA/CA mechanism for channel access, providing link reliability between two peer MAC entities, and so on [30].

On top of the MAC layer is a service-specific convergence sublayer (SSCS), providing the IEEE 802.2 logical link control (LLC) Layer access to the MAC layer. The IEEE 802.2 LLC layer further provides addressing and physical channel access for upper layers.

2.4.3 ZigBee™

ZigBee™ is a low data rate, low power consumption wireless protocol intended for automation and remote control and monitoring [33]. The ZigBee Alliance[7] was established in 2002, in order to "develop standards that ultimately deliver greater freedom and flexibility for a smarter, more sustainable world" [34]. ZigBee™ is developed on top of IEEE Std 802.15.4-2003 [31] and supports only star, tree, and mesh topologies. That is, ZigBee does not support cluster tree network topology [35].

IEEE and the ZigBee Alliance have worked closely during the standardization process. However, these two communities have different foci. The IEEE 802.15.4 Working Group mainly focuses on the physical and data link layer of the protocol stack; while the ZigBee Alliance focuses on specifying the upper layers (from the network layer and above, see Figure 2.4), in order to

[7] `http://www.zigbee.org/`

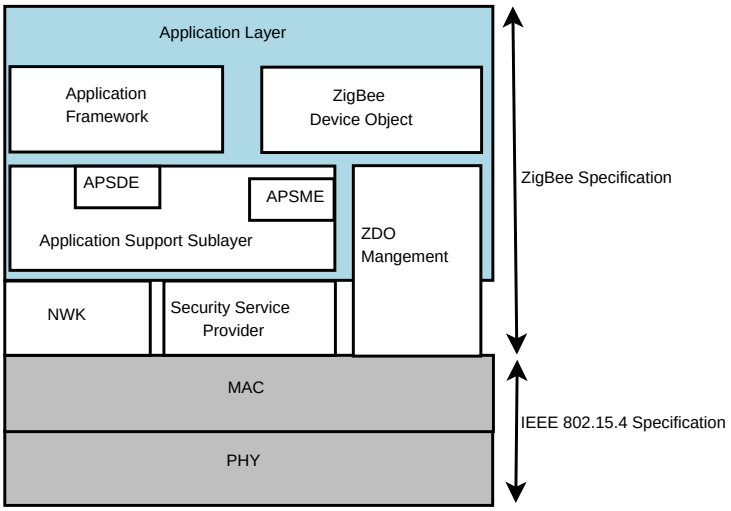

Figure 2.4: ZigBee stack architecture.

provide inter-operable networking, security services, application interfaces, as well as marketing and engineering evolution of the ZigBee^{TM} standard.

The ZigBee^{TM} network layer is designed to facilitate power conservation and to ensure low latency. It provides functionality to control and utilize the MAC layer as well as a service interface to the application support sublayer (APS) above it. The ZigBee network layer is responsible for starting a network, assigning node addresses, configuring new devices, discovering other ZigBee networks, and applying security policies [35].

The APS provides an interface between the network layer and the application layer [36] by providing services that are offered by two entities: the data service entity (APSDE) and the management service entity (APSME). The APSDE enables the transportation of application protocol data units (PDUs) between devices. The services APSDE provides include:

1. Generation of the application level PDUs (APDUs) – adding an appropriate protocol header to APDUs and generating APS PDUs.

2. Binding – creating a unidirectional logical link between a source endpoint-cluster identifier pair and a destination endpoint. The APSDE is able to send messages from one device to another once these two devices are bound.

3. Group address filtering based on endpoint group membership.

4. Reliable transport – providing transaction reliability by employing end-to-end retries.

5. Duplicate packet rejection.

6. Fragmentation – segmentation and reassembly of APDUs longer than the payload of a single network layer packet.

The application layer in ZigBee consists of the application framework and the ZigBee device object (ZDO). The application framework allows each ZigBee node to define up to 240 application endpoints in order to transmit and receive application data. The ZDO provides functions such as service and device discovery, coordinator initialization, security management, application endpoint binding management, and network management.

2.4.4 6LoWPAN

RFC 4919 [37] defines Low-power wireless personal area networks (LoWPANs) as networks comprised of IEEE 802.15.4-2003 [31] devices, which are characterized by short range, low bit rate, small packet size, low power, and low cost. A LoWPAN targets wireless connectivity for applications with limited power and low throughput requirements.

It is beneficial to have IP working over IEEE 802.15.4 links, in that IP networks are pervasive, proven to work, and built on open standards. Furthermore, IPv6 [38] meets LoWPAN requirements in that IPv6 has solutions for network auto-configuration and statelessness, which are desirable for LoWPAN devices, and IPv6 supports a large address space as needed in LoWPANs. In addition, IPv6 supports subsuming IEEE 802.15.4 MAC addresses when desired. Finally, IPv6 provides interconnectivity to other IP networks, e.g., the Internet.

However, there are several challenges when transferring IPv6 packets over IEEE 802.15.4 networks, because of the small frame size limitation and other constraints of LoWPANs. For example, there needs to be a fragmentation and reassembly layer below IP in order to transfer larger packets and there should also be a header compression mechanism in order to reduce overhead.

RFC 4944 [39] defines an adaption layer for enabling IPv6 on top of IEEE 802.15.4 networks. It also defines header compression mechanisms making IPv6 practical on IEEE 802.15.4 networks. However, RFC 4944 does not deal with mesh routing specifications.

2.4.5 CoAP

Constrained Application Protocol (CoAP) [40] is an application layer transfer protocol for resource constrained networks. CoAP is defined by the IETF Constrained RESTful Environments (CoRE) working group. CoAP can be used in M2M applications such as home automation, industrial automation, smart grids, and so on. Because of resource limitations of M2M nodes, CoAP is designed to have small message overhead. Hence, fragmentation is not allowed in CoAP. It realizes a subset of the Representational State Transfer (REST) protocol [41] common with HTTP [42].

It uses a method/response interaction model between different application endpoints, that is, CoAP messages contain either a method or response code, carrying a request or response semantics respectively. A request could either be Confirmable or Non-confirmable. The response to a request is carried in an Acknowledgment when the requested response is immediately available. It is a piggy-backed response. When a response is not immediately available, an empty Acknowledgment is returned first. A new Confirmable message is sent to the client when the response is ready. After receiving the response the client has to return an Acknowledgment.

CoAP also supports built-in resource discovery in order to facilitate M2M applications. This feature is very important for M2M applications since there are no humans in the M2M loop. To achieve this, the endpoints should conform to the CoRE Link Format [43] of discoverable resources. Resource discovery can either be unicast or multicast, which is useful when resources in a limited scope need to be located.

CoAP easily translates to HTTP since it supports a subset of HTTP functionality. It is useful for integration with web services. Sometimes an HTTP to CoAP mapping is necessary, for example, this can be implemented in a CoAP-HTTP proxy.

2.5 Distributed Hash Tables

A hash table or sometimes a hash map is a data structure that maps keys to values using a hash algorithm (see Section 2.5.1). A distributed hash table (DHT) is a hash table constructed and used in a distributed manner. A hash table is easy to deploy in a distributed system since it places few constraints on the keys or data, nor how they are organized. Distributed system's DHTs are maintained by the nodes in a network. These nodes act autonomously,

i.e. nodes join or leave the network without any centralized control [44].

DHT relies on three main components: the key space, the key partitioning algorithm, and the overlay network [44, 45]. The key space is the set of all possible keys. The key splitting algorithm splits the key space into different partitions, which are the responsibility of different nodes. The overlay network connects the participating nodes so that the node storing a specific key and its associated data can be found.

DHTs are widely used in peer-to-peer (P2P) systems for data lookup, since a DHT implements just one function: looking up a key and returning the ID of the node responsible for this key. The major issues when implementing a DHT include the following three [46]:

1. Load balancing among nodes: keys should be evenly assigned to the participating nodes so that every node is responsible for roughly the same number of keys. This assumes that each node has roughly the same local resources; if they have unequal resources, then of course the keys should be assigned proportional to the node's share of the total resource. This can be achieved using consistent hashing, as we will discuss in Section 2.5.2.

2. Forwarding lookups to appropriate nodes: when a node receives a lookup request and does not have the requested content, it should forward the request to a node that is closer to the key so that the request reaches the correct node. For example, when a node i receives a request for key k, which is greater than the node's ID N_i, then this node should forward this request to another node j, such that $N_j > N_i$ and $N_j \leqslant k$.

3. Building routing tables: every node keeps track of some other nodes in order to forward requests to them. This can be done in various ways. We will look at one implementation, Chord, in Section 2.5.3.

2.5.1 Hash Algorithms

Hash functions are algorithms which transform a variable length (binary) string of data into a small fixed length item. Hash functions are lossy compression functions that can be used to generate a fingerprint for a certain input. A widely used hash function is the MD5 [47] algorithm. The MD5 algorithm converts a variable-sized input into a 128-bit message digest of the input. It has been widely used since it was invented, for example the most

common uses are checking the integrity of files, SSL/TLS [48], IPSec [49], pseudo random number generation [50], and so on.

2.5.2 Consistent Hashing

Before introducing consistent hashing, we first describe the phenomenon of *hot spots* in a network. *Hot spots* happen when a single server receives requests from a large number of clients. This may overload the server, causing a denial of service, exponentially increasing load, increase probability of node failure, etc. The hot spot phenomenon is quite common with web services, when the service suddenly becomes very popular there may be more clients simultaneously attempting to access the server than the server was designed to cope with [51].

One way to remedy the hot spot phenomenon is to use cache servers. A cache server sits between the clients and the server. When it receives a request from a client, it looks up the data being requested in its own cache. If the data is in the cache, then the data is returned to the client; otherwise the cache server has to contact other cache servers or the actual server for the data.

One problem with this traditional approach is that the system may not scale, and when one cache server fails or new cache servers are added to the system, the cache servers may need to remap their cached pages [52]. David Karger *et al.* [51] address this problem with random cache trees and consistent hashing. Random cache trees are used to coalesce requests from clients. Consistent hashing is employed to balance the load even with a fluctuating number of cache servers. Consistent hashing tries to split items into sets so that every set has roughly the same number of items; at the same time ensuring that:

1. A change in one set does not cause re-assignment of items to other sets;
2. Moving items from one set to another causes only slightly different arrangements of the mapping of items to sets.

In contrast, a traditional hashing algorithm will cause all items to be remapped when the number of sets changes. Consistent hashing is used to enable easy re-assignment of keys to adjacent nodes when there is a loss or addition of a node. Note that keys and values have to be stored redundantly, otherwise the loss of a node leads to a loss of data. According to [53], consistent hashing ensures with high probability the minimum amount of remapping of keys when the N^{th} node joins or leaves the network; in this case only $O(1/N)$ of

the keys need to be moved to a different node in order to balance the load across the nodes.

A consistent hashing algorithm is easy to implement [54]. First we need a standard hash function, such as cyclic redundancy checks (CRCs) [55], MD5, one of the SHA series of hash functions, e.g. SHA-1 [56], and so on. This function will map strings into numbers in the range $[0, \ldots, M]$, where M is the number of sets. A consistent hashing function can be constituted by dividing the numbers by M so that every hash value falls into the interval $[0, 1]$, thus the values can be mapped to a unit circle. In this way every string (item) is mapped to a single point on the circle. By mapping these hashing cache servers onto the same circle, we assign items to corresponding cache servers, that is, every cache server is responsible for items between itself and the previous cache server.

Consistent hashing is usually used in DHTs to map keys to nodes, thanks to the property that removal or addition of one node changes only the set of keys owned by the nodes with adjacent IDs, while leaving all other nodes unaffected [54].

2.5.3 Chord

Chord [53, 57] is a distributed lookup protocol for efficiently locating a node that stores certain data item in P2P applications. It addresses problems including load balancing, decentralization, scalability, availability, and flexible naming. In a N-node system, each Chord node stores information about $O(\log N)$ other nodes when the system is in steady state. Lookups are resolved via $O(\log N)$ messages to other nodes. Nodes joining and leaving the system will result in no more than $O(\log^2 N)$ messages with high probability.

In Chord, each node has an ID and is mapped to a certain place in a circle [58], as shown in Figure 2.5. The predecessor of a node is the peer in front of it when traversing the circle clockwise. Likewise, the successor of a node is the peer following it. For example, in Figure 2.5 node $N2$'s predecessor and successor are $N0$ and $N3$ respectively. Each node is responsible for data with a key between the predecessor's ID and its own ID. For example, in Figure 2.5 node $N2$ is responsible for data with a key $K1$.

Each Chord node contains a routing table (or finger table) about $O(\log N)$ other nodes in the half of the Chord ring clockwise from the node. The i^{th} entry in node N_n's finger table contains the identity of the first successor that is at least 2^{i-1} away from N_n on the ring. Each node also maintains information about its predecessor. For example, the finger table of node

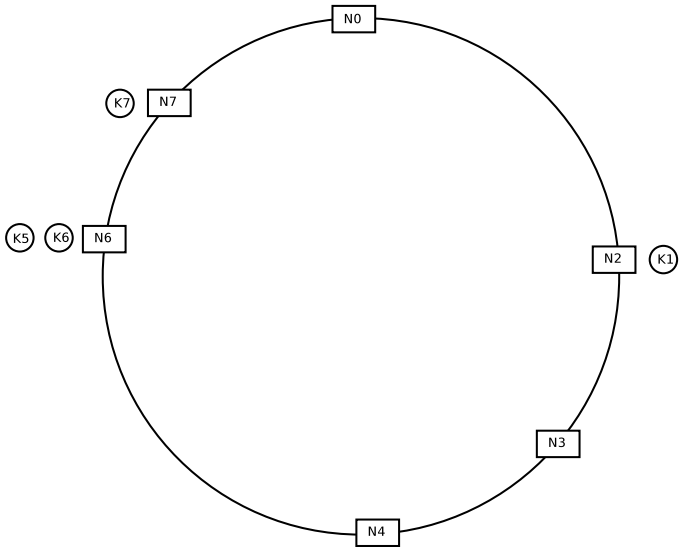

Figure 2.5: Illustration of a Chord ring.

$N2$ in Figure 2.5 looks as Table 2.1. Routing in Chord is accomplished

Table 2.1: An example of Chord finger table.

Start	**Successor**
N2 + 1	N3
N2 + 2	N4
N2 + 4	N6
Predecessor = N0	

by querying the nearest finger of the key being looked up. Each routing hop reduces the distance on the circle/ring to the destination node approximately in half, thus ensuing efficient lookups.

When a node joins, it performs a lookup with its own node ID, treating the result as its successor. It then does lookups to locate its predecessor and updates its finger table. Its predecessor and successor will also update their finger tables subsequently, during their periodic maintenance time slots. When a node plans to leave, it informs its immediate predecessor and successor and transfers its data items to its successor [58].

2.6 Simple Network Management Protocol

The Simple Network Management Protocol (SNMP) [59] aims to provide node management functionality in the Internet. An SNMP management system contains three components [60]:

1. Manager entities that generate commands and listen for responses.
2. Nodes that are managed by the manager. These nodes (agents) are software processes that maintain information about themselves and their environment, as well as respond to commands received from managers. Agents often reside in network equipments such as network hubs, routers, and workstations.
3. A management protocol that conveys management information between the managers and agents.

The SNMPv1 [59] protocol specifies five command, response, or alert PDUs:

1. *GetRequest* – A request sent from a manager to an agent to retrieve the value of a variable stored by the agent.
2. *SetRequest* – A request sent from a manager to an agent to change the value of a variable stored by the agent.
3. *GetNextRequest* – A request sent from a manager to an agent to discover the next available variable and its value as stored by the agent.
4. *Response* – A message sent from an agent to a manager in reply to a fetch command such as *GetRequest*, *SetRequest*, *GetNextRequest*.
5. *Trap* – An unsolicited alert message sent from an agent to a manager when there is a significant event.

SNMPv2 specifies two more PDUs in RFC 1905 [61], namely *GetBulkRequest* which optimizes *GetNextRequest*, and *InformRequest* enabling acknowledged asynchronous notification between managers.

2.7 Summary

In this chapter we have discussed concepts and technologies related to this thesis. We summarize what we have covered as follows:

- The IoT embraces heterogeneous devices into a worldwide network based on standard protocols. IoT will penetrate into our life more than the current technologies have. By 2020 there will be 50 billion connected to the IoT.

- M2M is a similar concept that intersects with IoT. M2M network generally consists of smart objects interacting with each other without human intervention. A typical M2M system has five basic components: users, objects, network, service platform, and enterprise information system.

- WSNs are used for monitoring as well as control purposes, for example, industrial control and monitoring, home automation, security and military sensing, and so on. Nodes in WSNs scatter over a sensor field and the number of sensor node is usually large. These nodes are prone to failure and the network topology may subject to change. Different routing protocols have been proposed to find an appropriate path to transfer data in WSNs. These routing protocols are categorized into four categories: data-centric protocols, hierarchical protocols, location-based protocols, and network flow and QoS-aware protocols.

- IEEE Std 802.15.4 specifies the PHY and MAC layer for LR-WPANs. ZigBee is developed on top of IEEE Std 802.15.4 and it defines the upper layers (from the network layer and above) in the OSI model. ZigBee provides a stack profile standard that allows developers to create their own application profiles. The IEFT specifies 6LoWPAN as an alternative to enable IPv6 on top of IEEE 802.15.4. Since 6LoWPAN is only an adaption layer, CoAP is suggested as the application layer protocol for M2M systems.

- A DHT is a hash table constructed and used in a distributed manner in order to provide an efficient way of data lookup in highly scalable P2P systems. Chord is one of the DHT algorithms that yield good performance in load balancing, decentralization, scalability, availability, and flexible naming.

- SNMP is a protocol that provides node management functionality in the Internet. It is specified by the IETF.

Chapter 3

Design

The goal of this thesis project is to connect resource constrained devices to a larger network, for instance the Internet. In particular, we assume a scenario as shown in Figure 3.1. In this scenario, we assume a number of LR-WPANs are scattered in different geographic locations. The distance from one WPAN to another can be tens of kilometers or more. Thus it is impossible for nodes in one LR-WPAN to communicate with nodes in another WPAN. The WPAN nodes are assumed to be battery powered and have very constrained computational capability.

Under this assumption, one *proxy node* is introduced in each LR-WPAN in order to connect the LR-WPANs together. This proxy node is both wireless wide area network (WWAN) and WPAN enabled. It is responsible for managing WPAN nodes that are in the same WPAN as the proxy, as well as exchanging information with WWAN peers. Note that proxy nodes may also have sensor or actuator modules attached to them. The proxy nodes have fewer resource constraints as compared with WPAN sensors/actuators, and they will mostly be mains powered. However, the power consumption of these nodes are relatively low compared with personal computers, and they have less computational capability.

To ensure scalability, a DHT overlay is used and all proxy nodes as well as other WWAN nodes are connected to this DHT overlay network. DHT is used for node lookup (resembling the functionality of a domain name system (DNS)) and resource locating. A M2M Communication Enabler (M2M CE) provides functionality to translate a resource URI to a responsible proxy node's IP address. However, the functionality of M2M CE is not within the scope of this thesis project.

Furthermore, a monitoring and control node (MCN) is used to gather infor-

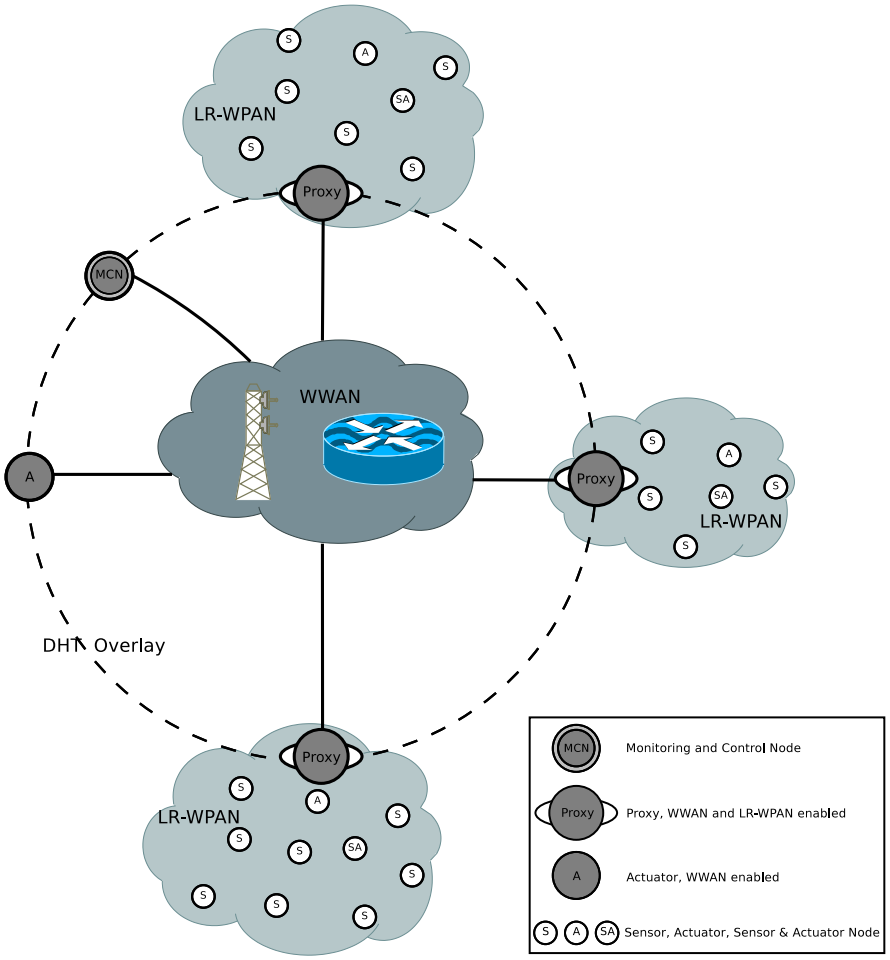

Figure 3.1: Overall architecture.

mation from end WPAN nodes and proxies, as well as to send commands to them in order to configure the system. The MCN may not be online all the time. However, it has to be in the DHT overlay in order to easily access both the WWAN nodes and WPAN nodes. The communication between the Monitoring and Control Node (MCN) and the sensor nodes looks roughly as Figure 3.2.

To simplify the problem, we assume the underlying WWAN link technology is a 3G connection, since it is mature technology and easily accessible. For the LR-WPAN communication we choose ZigBee over 6LoWPAN and other technologies based on IEEE 802.15.4, because ZibBee devices are more popular in the market.

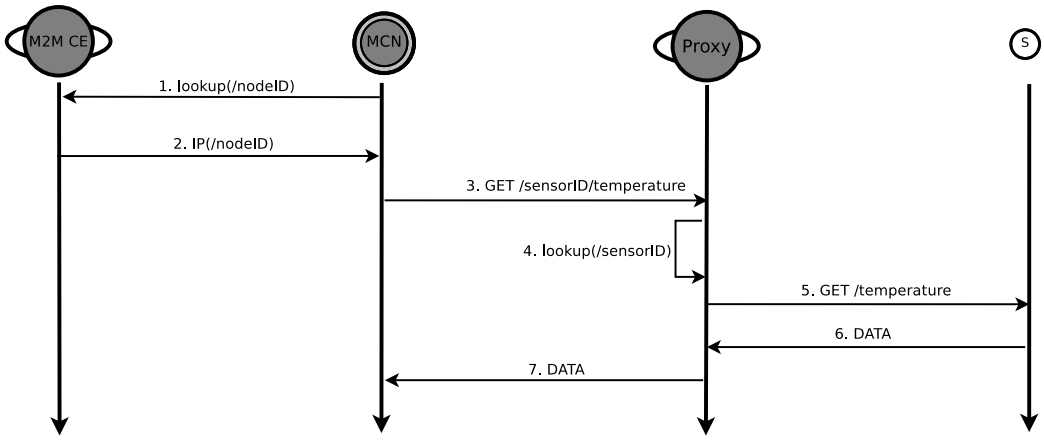

Figure 3.2: Flow of communications between MCN and a single sensor(s) that returns the temperature that it has measured locally.

3.1 Motivation

We believe a proxy is necessary in a P2P-based M2M network for the following reasons:

1. Most nodes in WSNs are so resource-constraint that it is not feasible to run P2P protocols on them. In fact, it may not be feasible to run P2P protocols on even less resource-constraint devices, for instance mobile phones, according to [62], which shows it consumes significant battery and yields long message delays when running REsource LOcation And Discovery (RELOAD) [63] on mobile phones.

2. A proxy enables efficient data aggregation and management. As we discussed in Section 2.3, sensor nodes are generally densely deployed in a WSN, and they typically produce a large amount of data. The same data value may be collected multiple times by a single node or by multiple nodes. Although data aggregation mechanisms may be applied on the sensor nodes themselves, these mechanisms may not produce satisfactory results due to limitations of nodes themselves. Hence it is essential to perform data aggregation at the sink (proxy) node before storing all the gathered information in the P2P overlay, so as to avoid unnecessarily flooding nodes in the overlay (since data will be stored in a distributed manner in the overlay).

3. A proxy can act as a firewall or access control server, thus ensuring

security of the system. Instead of exposing all the sensor nodes to the larger network, i.e. the Internet, the proxy controls which resources in the WSN can be accessed by another specific party. This kind of security control is more convenient than specifying security policies at the individual sensor nodes, due to sensor node limitations.

4. Introducing a proxy into the system simplifies the design of application logic on the WSN nodes. This design paradigm shifts the complexity of sensor nodes to proxies. Since the proxy nodes are supposed to be more capable than the sensor nodes, it is straightforward to keep the sensor nodes simple and shift the complexity to the proxy.

5. A proxy can connect heterogeneous sensor nodes to the system. This is especially important for M2M networks, since nodes in the network may rely on different communication protocols. The proxy can bridge different communication protocols, thus embracing various portions of networks into the M2M system.

Although some of the above functionality may be realized by nodes other than a proxy; for example, a dedicated security server can be used to handle resource access control, we feel these mechanisms may again add complexity to the whole system. As a result,we try to keep the sensor node logic simple and move all the complexity to proxy nodes, without complicating the M2M system too much.

3.2 Objective

The goal of this thesis project is to make WPAN nodes addressable from the WWAN. A proxy node bridges the communication between a WWAN node and a WPAN node. Thus, a WWAN node requests WPAN resources through the proxy node, and *vice versa*. In our case the proxy should have the following capabilities:

1. WPAN node discovery: The proxy should be able to discover new WPAN devices when they are connected to the WPAN and announce them to the DHT overlay if needed.

2. WPAN service discovery: The proxy should be able to discover the services that each WPAN device provides. For instance, the proxy should be aware of all services each sensor/actuator node supports.

3. Retrieve information from WPAN nodes and WWAN nodes. For example, the proxy should be able to get information from and post data to sensors and actuators both in the WPAN or WWAN.

4. Process information gathered from WPAN nodes or WWAN nodes and store this information to the DHT overlay (if required).

3.3 Principles

When designing the proxy, we have kept several principles in mind. These principles have clearly affected the design and implementation of the proxy. These principles are:

1. Power efficiency: The proxy should take power consumption into consideration when communicating with WPAN nodes, since the WPAN nodes may have very limited energy on-board.

2. Interoperability: The proxy should enable heterogeneous devices to communicate with each other. We assume that the WPAN nodes may have different hardware architectures, operate over different underlying communication protocols, or suit different application purposes. The proxy should provide transparent access to the WPAN nodes from the WWAN nodes.

3. Scalability: The proxy should to be able to function well when the number of nodes in its WPAN increases. The proxy should not cause low responsiveness, long response times, service unavailability, etc., due to the large number of nodes in the WPAN.

4. Security: The proxy should protect both the confidentiality and integrity of the data collected by a WPAN node. It should also be able to detect potential security attacks and report unusual events to a MCN.

5. Reliability: The proxy should reply with reliable data upon request from WWAN nodes. Furthermore, it should be aware of the unavailability of WPAN nodes and act accordingly in order to produce satisfying results for the WWAN nodes.

3.4 Architecture

Based on the above discussions, we present a system architecture as shown in Figure 3.3. In this architecture, the proxy provides the MCN with management and data access to the WPAN nodes. The proxy intelligence is a daemon that listens to incoming requests, distinguishes different types of requests, and calls the corresponding APIs. The management API provides interfaces for managing WPANs, WPAN security, and WPAN services. The data API provides interfaces for collecting data from the WPAN and delivering commands to the WPAN.

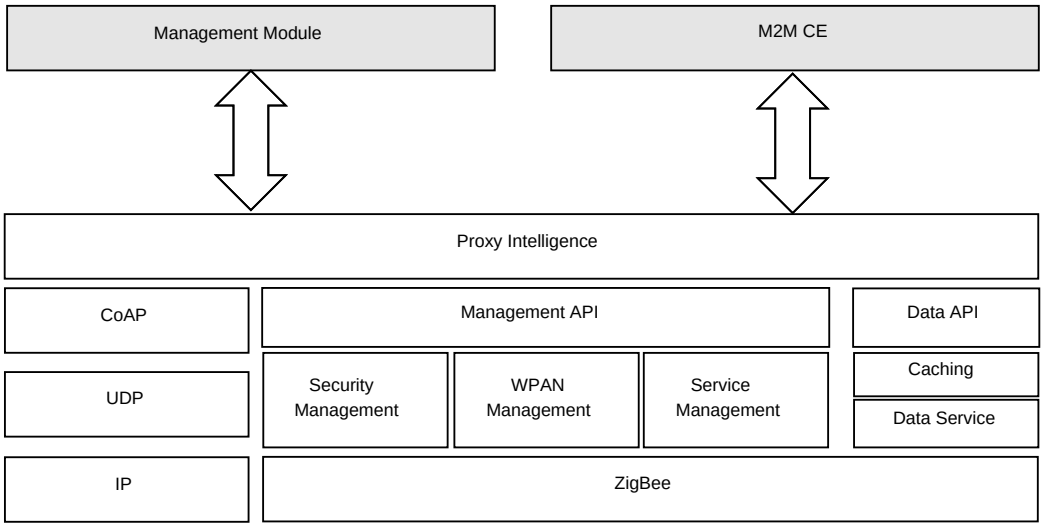

Figure 3.3: Proposed proxy architecture.

This proxy bridges the communication between a WWAN and a WPAN. On the WWAN (IP network) side, the proxy communicates with the management module and M2M CE using protocols based on IP, such as SNMP and CoAP; on the WPAN (ZigBee network) side, the proxy communicate with WPAN nodes using WPAN protocols, for example, ZigBee and CoAP (if 6LoWPAN is the used in the WPAN). The protocol stack on a proxy node is shown in Figure 3.4.

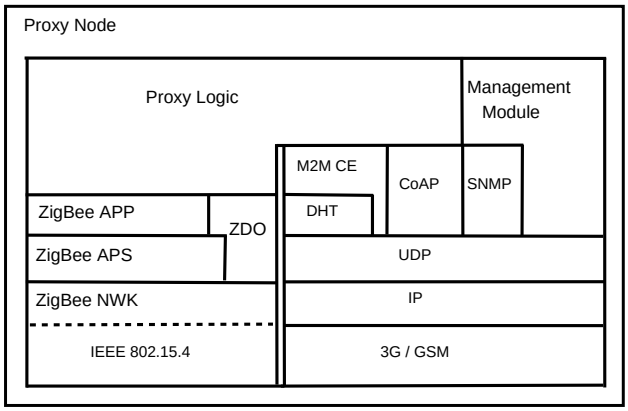

Figure 3.4: Protocol stack of the proxy node.

3.5 Details

The unified modeling language (UML) [64] use case diagram of the proxy is shown in Figure 3.5. We will discuss the functionality in detail in the following sections.

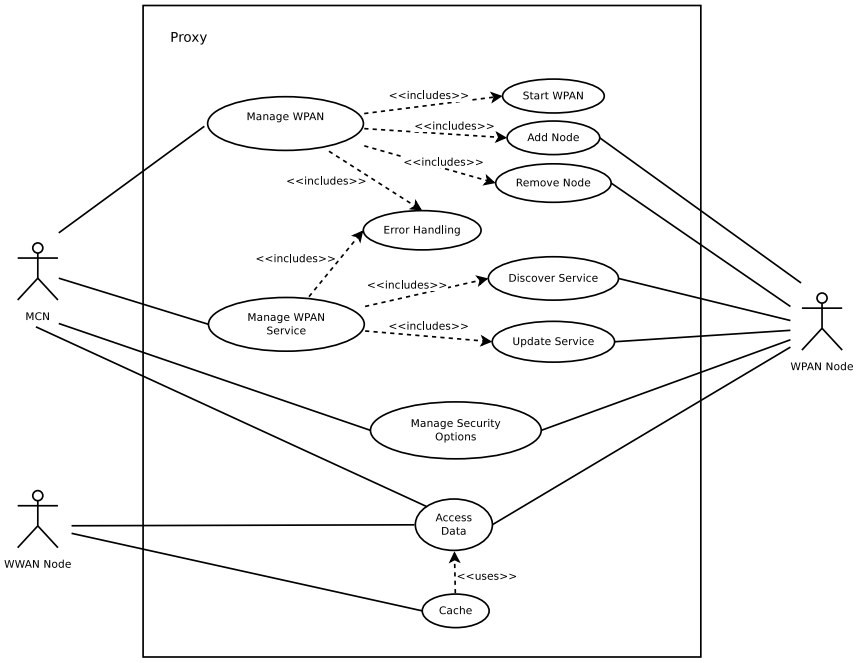

Figure 3.5: The UML use case diagram of the proxy.

3.5.1 Proxy Joining and Leaving

The proxy itself has to be a part of the DHT overlay in order to make WPAN nodes accessible from the DHT. As a result, the proxy should join the DHT first *before* announcing WPAN nodes to the DHT or revoking them from the overlay.

To join the overlay, the proxy should have a CoAP URI, which can be pre-configured or configured using SNMP. When the proxy joins the DHT, it negotiates with an M2M CE instance, which may either reside in the same node or on a different node in the network. After it becomes a part of the overlay, the proxy scans all the resources that are supposed to be visible in the WWAN and announces them to the overlay, again via negotiations with an M2M CE instance.

To leave a DHT overlay, the proxy first revokes all the resources it has announced to the overlay. Then it leaves the overlay itself. All operations on the DHT overlay by the proxy are done through an M2M CE instance.

3.5.2 WPAN Management

The proxy is responsible for configuring the WPAN. It should allow or disallow new WPAN devices joining the network. When a node leaves the network or silently leaves – due to failure because of battery exhaustion or hardware failure, the proxy should detect that a node has left the network within a suitable time period ranging from a few seconds to possibly hours, depending on the actual application scenario.

Another important aspect of WPAN management is ensuring security. The proxy should enable secure data transmission among WPAN nodes. This security should ensure both data integrity and authenticity of the data. We will discuss more about system security in Section 3.5.5.

The proxy acts as the coordinator in the WPAN. It is responsible for the initial configuration of each WPAN network. It selects an appropriate ZigBee PAN ID and operating frequency (radio channel), then configures security options including encryption options and encryption keys. Details of the proxy's operation will be described in more detail below.

3.5.2.1 WPAN Start Up

To start up a ZigBee network, the proxy first performs a channel scan creating a list of potential channels, while removing channels with excessive energy levels (i.e., these channels are already busy with other traffic) [65].

Next the proxy tries to select a PAN ID. To ensure that PAN ID is not already in use the proxy performs a PAN scan by sending beacon broadcasts on each potential channel. After it receives responses from nearby coordinators and routers that have already joined a network, it either randomly selects a available PAN ID or chooses a PAN ID specified by the upper layers.

Next the proxy will configure the network based upon any specified security policies. To perform this configuration, the proxy sets whether security is enabled in the network, and if so generates the network security key and the trust center link key. Follow this, the proxy configures the relevant encryption options, such as whether to use a trust center and how to send the security key when each node joins the network.

The reason we co-locate the ZigBee PAN coordinator and the proxy is to minimize energy consumption. If the coordinator is placed in another node, then transmitting messages would consume more energy. All of these requirements on the proxy and coordinator implies that the node supporting the proxy and coordinate should be mains powered.

3.5.2.2 Node Joining

In ZigBee networks routers and end devices can join an existing network. In order to join a network, the node has to configure itself with the PAN ID of the network that it wants to join, then perform a channel scan with a bitmask containing the proxy's operating channel. Additionally, this potential new node must have a security policy conforming to the proxy's settings.

A node can only join a network if it has the correct PAN ID, pre-configured security keys, and the proxy permits this node to join the network. We demonstrate the process of a node joining a network in Figure 3.6. After the proxy starts up, initially there are no other nodes in the network. In order for the proxy to permit a node to join its network, the node's parameters must be configured to conform to the proxy network settings. Initially we will assume that the user manually configures the sensor node, then sends a "add node request" from the MCN to the proxy. The proxy should first verify the request, then enable nodes to join the network. At this point the node is able to join the network. Once this node has joined the network the

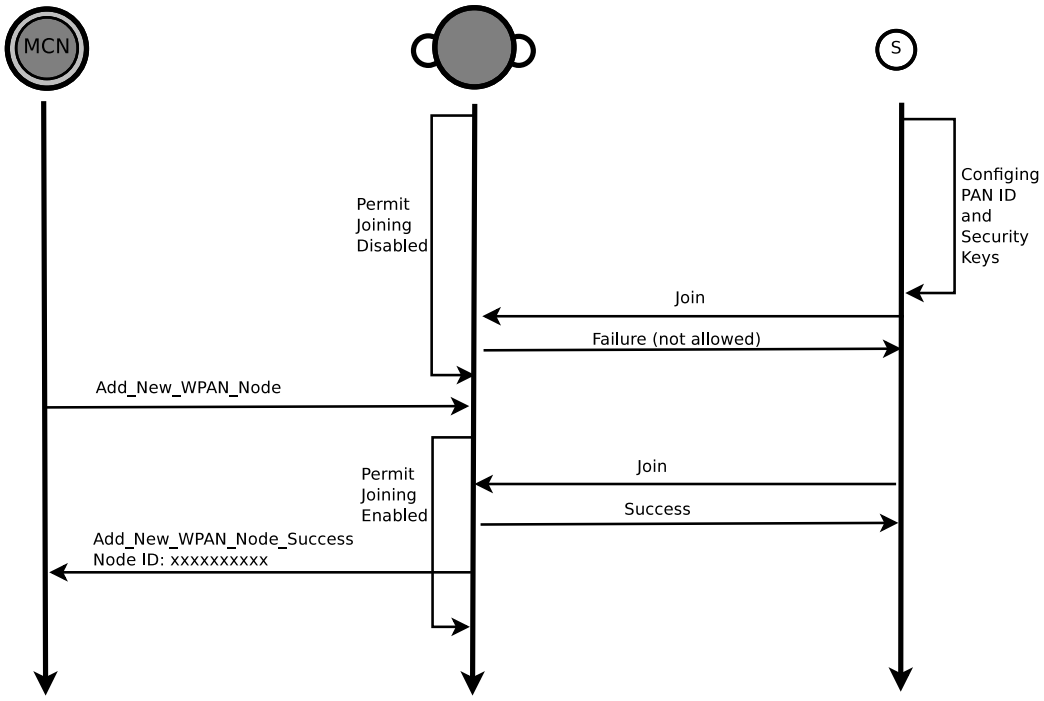

Figure 3.6: Node joining a ZigBee network.

proxy will send an acknowledgment to the MCN.

After a node has joined the network, the proxy will keep a record of the newly added node – including what services it can provide. In order to do that, a node service discovery is performed, this will be discussed in Section 3.5.3.1.

3.5.2.3 Node Leaving

Some nodes may need to leave the WAPN. Generally speaking, there could be two ways that a node leaves a network: explicit leaving and implicit leaving. Explicit leaving means that a node notifies the network that it intends to leave, while implicit leaving refers to the situation where the network is not explicitly informed when a node leaves.

If a node leaves the network explicitly, then it sends a "leaving network" message to the proxy. This will cause the proxy to remove the node from its list of "available nodes". The M2M CE should also be notified about the change in the network topology. Moreover, the MCN should be notified of this change.

Implicit leaving is a bit more complicated. Two schemes may be used to find out whether a node is alive or not, depending on how aggressive the proxy wishes to be.

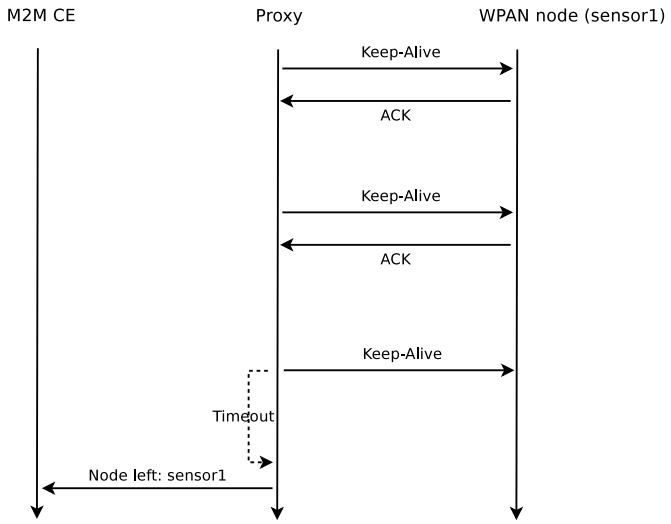

Figure 3.7: The "eagle" scheme in which the proxy polls WPAN nodes.

The first scheme will be named the "eagle" scheme, in this scheme the keep-alive messages are used. The keep-alive messages can either be sent by the proxy (see Figure 3.7) or by the WPAN nodes (see Figure 3.8).

In the first case the proxy continually asks the nodes if they are alive or not. If the proxy does not receive an acknowledgement from a WPAN node during a timeout period, then the proxy considers the WPAN node to be unavailable. The keep-alive message interval can be configured by the management module.

In the second case the WPAN nodes actively send keep-alive messages to the proxy without proxy polling. Similarly, when the proxy times out waiting for a keep-alive message from a WPAN node, it treats that node as dead and informs the DHT overlay via an M2M CE. Since the WPAN nodes control the intervals to send keep-alive messages, the proxy should be able to configure this parameter by negotiating with the corresponding WPAN node. The solution for this may depend on the actual WPAN node, for example, the communication protocol and firmware of the node. This may be possible for nodes that support over the air programming (OTAP) [66].

Unfortunately, both cases in the "eagle" scheme consumes energy of both the target node and other nodes on the path to this node. However, this scheme

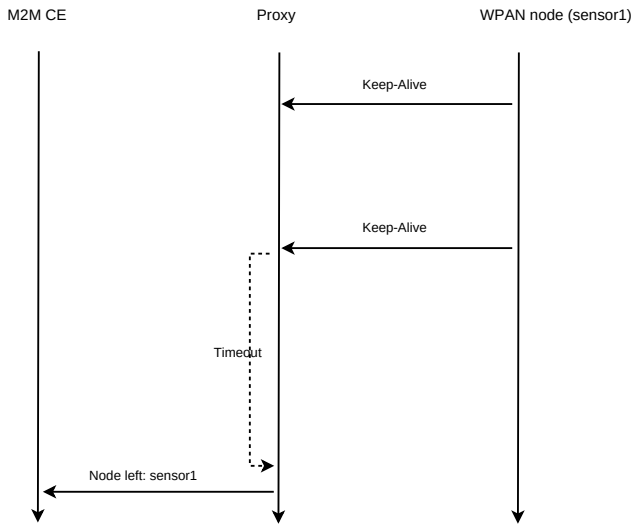

Figure 3.8: The "eagle" scheme in which WPAN nodes actively send keep-alive messages to the proxy.

may be necessary for WPAN nodes that are critical for the system.

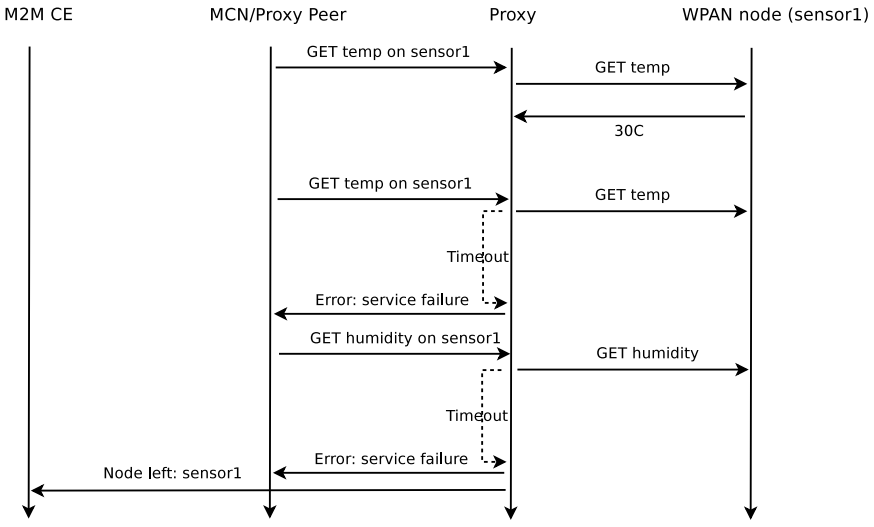

Figure 3.9: The "ostrich" scheme.

The second scheme will be named the "ostrich" scheme (see Figure 3.9), in this scheme the proxy does not send keep-alive messages at all. Instead, the proxy queries a node only when data transmission is needed. If there are

several consecutive missing acknowledgments, then the proxy considers this node to be dead and notifies the upper layers.

In order for the WPAN service to be more reliable, the proxy should provide mechanisms to warn the MCN about potential failures that are going to happen in a period of time. For instance, when the proxy detects a WPAN node has low battery level, it should report this to the MCN.

3.5.2.4 Naming, Addressing, and Routing

We use concepts of names, addresses, and routes conforming to those defined in [67]. A name denotes an entity and is usually human-readable. A route is a list of names representing the path from one to another. An address is intended for machine processing and route generation.

Every entity has a human-readable CoAP name in our system. This name is used whenever human users are involved in the loop, for instance, when a user configures which node a sensor should report to when certain thresholds are met. More specifically, CoAP names are used mainly by the MCN, via which human users communicate with the M2M system.

The DHT overlay acts as a distributed domain name service (DDNS) system, translating CoAP names to addresses. Since our DHT overlay relies on IP, the CoAP names are always mapped to an IP address, that is, names of proxy nodes and WWAN nodes map to IP addresses (of themselves), and names of WPAN nodes map to the IP addresses of their *proxies*. Each proxy is responsible for translating names of WPAN nodes to their corresponding addresses, in our case, their 16-bit WPAN addresses or 64-bit link addresses.

Routing in IP networks is not a major concern of this thesis, as we utilize the current network infrastructure; otherwise routing in WPANs depends on the use and functionality of the WPAN. For instance, many-to-one routing [65] is suitable for scenarios where many nodes send data to a single central collector. For use cases where a node sends data to many other nodes, source routing can result in low packet delays and good network performance. Generally speaking, mesh routing mechanisms based on *ad hoc* on-demand distance vector (AODV) [68] can also be appropriate for mesh networks.

3.5.3 WPAN Service Management

Not only is the proxy responsible for managing the WPAN nodes, but it is also responsible for managing the services provided by these nodes. The proxy discovers what services are offered by each WPAN node. For example, the proxy may need to be aware of whether a WPAN node provides a temperature reporting service or a light switch service.

The proxy may need to update the list of services provided by a WPAN or WPAN node. For instance, when a service on a node is no longer available or the node becomes unavailable, then the proxy should be aware of this change; furthermore, when a new service is enabled, the proxy should be informed of the change. We will discuss this in detail in the following sections.

3.5.3.1 Service Discovery

Service discovery is usually performed by the proxy immediately after a new node joins the WPAN. However, the proxy may also perform service discovery upon request from a MCN (there may even be more than one MCN in the system).

When service discovery is needed, the proxy sends a "service discovery" request to a WPAN node. After getting a response from the node, it parses the response and creates entries in a service table. The table should contain information about the WPAN node's identity, service type, supported access methods, and availability. The table might also contain additional information about the service, for instance, description of the service. An example of the entries that might be in such a service table are shown in Table 3.1.

Table 3.1: An example of service table, where the proxy manages one sensor and one actuator.

Node ID	**Service**	**Methods**	**Last Available Time**
sensor1	Temperature	GET	13:55:21 Mar 29, 2011
	Humidity	GET	13:25:29 Mar 29, 2011
	Wind Speed	GET	13:56:41 Mar 29, 2011
actuator2	Light Switch	GET, PUT	06:36:41 Mar 29, 2011

Service discovery in WPAN can be done in various ways [69]. In our case we perform service discoveries by sending queries to WPAN nodes. These queries can be broadcast when the WPAN is initially formed. After the network

becomes more stable, unicast queries are sent to specific nodes instead of the whole network.

We are able to use ZigBee ZDO functionality. For example, ZDO provides commands to request node descriptors from a ZigBee device, active endpoints in a node and simple descriptor of a specific endpoint. Similarly, for WPANs running on 6LoWPAN, CoAP with CoRE link format [43] provides mechanisms to perform resource discovery. For general purpose WPANs, SensorML [70] provides an XML encoding for describing any processes and resources specifically for sensor networks.

The proxy is not required to inform the M2M CE or the MCN of service table changes, since this table could constantly change. Constantly updating the overlay of table changes may cause flooding in the overlay and may unnecessarily consume energy at the proxy as well (although the proxy is mains powered, energy efficiency should always be a concern). On the other hand, when certain critical services go down, it is necessary for the proxy to inform upper layers. We will discuss this in the following section.

3.5.3.2 Service Updates

The proxy does not have to inform the overlay about all changes that occur in the service table. However, if critical changes occur in the service table, then the M2M CE or the MCN should be notified. Therefore, in the service table we keep track of all the critical services. Once a critical service is detected to be unavailable, then a notification will be immediately sent to the M2M CE.

When a service is required to be actively turned off, then the proxy will mark the status of this service as "Not Available". This can be done by sending a request from the MCN to the proxy, requesting that specific services be turned off. Similarly, the proxy may enable services, thus an MCN can turn on (or off) specific services. Upon receiving an indication of unavailable services for a request, an error message will be returned to the sender of this request.

3.5.4 Caching

Caching mechanisms will improve system performance when used correctly. We employ cache mechanisms in two separate ways in our system, that is, caching is used both for WPAN node data and for proxy peers. We discuss both of them in detail in this section.

3.5.4.1 Caching for WPAN Nodes

The proxy nodes cache data received from the WPAN. Cached data is sent back when the requested data resides in the cache. Caching for WPAN nodes in the proxy is need for several reasons:

1. To save energy: Energy is a major issue for LR-WPANs, thus we should avoid frequently operations on these nodes. Additionally, it is frequently unnecessary to get real-time data from the LR-WPAN nodes. For instance, the temperature as measured by one sensor node may not change significantly in a short time, e.g. in a few seconds. For applications that do not need real-time data, the proxy can reply to requests with cached data, thus avoiding waking up WPAN nodes constantly, while still providing the requested data via the proxy.

2. To ensure reliability: Nodes in a LR-WPAN may be vulnerable due to their constrained on-board resources and changing environment. As a result, they may be temporally out of service at times. Employing cache mechanisms will offer more reliable data service, at a cost of potentially obsolete data.

3. To improve response time: Getting data from the WPAN nodes will require more time than to get data from the proxy.

While caching is useful with applications not so critical with getting data in real-time, for certain applications caching may be inappropriate. As a result, the proxy should be configured explicitly with which data can be cached and which should not. This can be configured using SNMP.

3.5.4.2 Caching in DHT

To further increase reliability caching for peers (proxy nodes and WWAN nodes) is needed. Since all nodes in a WPAN communicate with nodes outside the WPAN via a proxy, this proxy represents a single point of failure. Once a proxy fails, all the nodes in a WPAN would be unable to send or receive data, hence there are two alternatives: a standby (or secondary) proxy and peer caching.

A peer caching mechanism enables a proxy in the DHT network to cache a copy of data for another node. When one proxy fails, the DHT overlay still has a relatively fresh copy of the data provided by the dead proxy, thus it may respond to requests on behalf of this failed proxy. This mechanism is

realized by requesting the M2M CE to store data in the overlay. Note that the overlay itself does not actually store any data, instead it keeps track of where specific data has been stored.

3.5.5 Security

We discussed in Section 3.1 using a proxy simplifies the security management of WPAN nodes. This design thus emphasizes implementing a secure gateway that authenticates the source of incoming service requests and ensures secure data communication in WPANs.

3.5.5.1 Secure Communication between WWAN Peers

The security of communication between WWAN peers or proxy nodes is not a complicated issue, since CoAP and SNMP are used for different purposes. Generally we utilize CoAP for data communication and SNMP for management tasks. Security is ensured in CoAP either with Datagram TLS (DTLS) [71] binding or IPSec [49]. Security in SNMP is also well specified in many IETF RFCs, for example, RFC 1909 [72], RFC 1910 [73], RFC 3414 [74], RFC 3816 [75], etc.

3.5.5.2 WPAN Security

Our WPANs consist of ZigBee networks in this thesis project. ZigBee ensures both infrastructure security and application data security [76]. Infrastructure security in ZigBee includes network access control, packet routing integrity, and prevention of unauthorized packet transport; and application data security ensures message integrity, authentication, freshness, and privacy.

When a node joins a ZigBee network, it has to possess a shared network key. After it joins the network any two nodes communicating with each other share a link key. A trust center is used for key management and distribution. The trust center in our case is the ZigBee coordinator, which is a part of the proxy node. Figure 3.10 shows authentication and encryption performed on a ZigBee packet when both network layer security and APS layer security are applied.

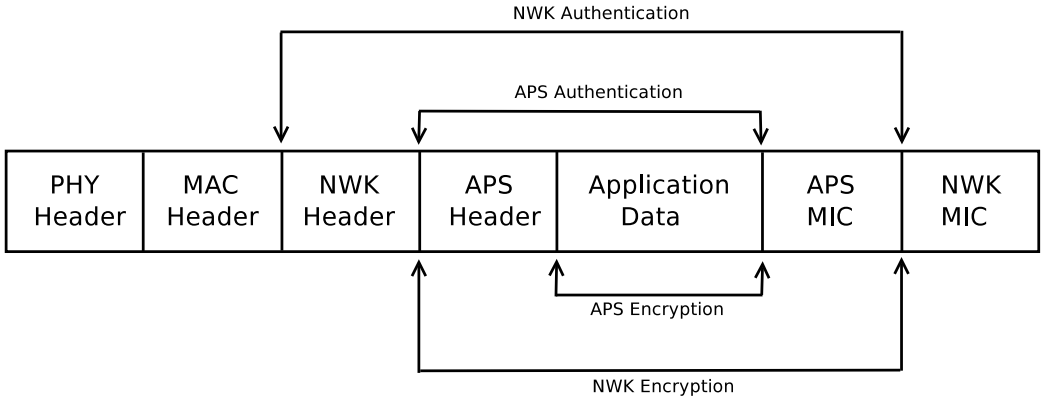

Figure 3.10: Authentication and encryption in ZigBee packets.

3.6 Summary

This chapter has covered the design of our proxy in a DHT-based M2M system. We first discussed why a proxy in such systems is needed: WPAN nodes can not run P2P protocols; a proxy is good for data aggregation and security; it simplifies WPAN node logic; and a proxy connects heterogeneous devices to the system. Then we proposed a proxy architecture based on principles including power efficiency, interoperability, scalability, security, and reliability. Then we looked at proxy joining and leaving, WPAN management, and service management in detail. We introduced caching mechanisms in our design in order to achieve power efficiency, reliability, and better response time. Finally we have discussed security considerations regarding communication between the proxy and WWAN peers as well as communication between WPAN nodes.

Chapter 4

Implementation

We have implemented a prototype proxy for this thesis project. The major functionality of this prototype includes WPAN node management, WPAN resource announcement to the DHT overlay, and translation between IP address and ZigBee address. We introduce what hardware and software are used and how the prototype is implemented. More details can be found in Appendix A.

4.1 Hardware and Software

We want to build a prototype with low cost. As a result, inexpensive products and free/open source software are used in this prototype implementation. We discuss what hardware and software resources have been used in this section.

4.1.1 WPAN Nodes

We choose the Libelium[1] Waspmote[2] as our WPAN node (see Figure 4.1). On board it has a low-power Atmel[3] 8-bit RISC-based microcontroller with a frequency of 8MHz, combined with 128KB of flash memory, 8KB of SRAM, and 4KB of EEPROM [77]. It has low power consumption, supports ZigBee and other communication modules such as Bluetooth and GPRS. More importantly, it has a sensor board to which many sensors can be adapted. For example, this sensor board supports sensors to detect gases, temperature,

[1] `http://www.libelium.com`

[2] `http://www.libelium.com/products/waspmote`

[3] `http://www.atmel.com/`

liquid level, weight, pressure, humidity, and so on. For our application we use a Digi[4] XBee™ ZB[5] transceiver as our ZigBee communication module.

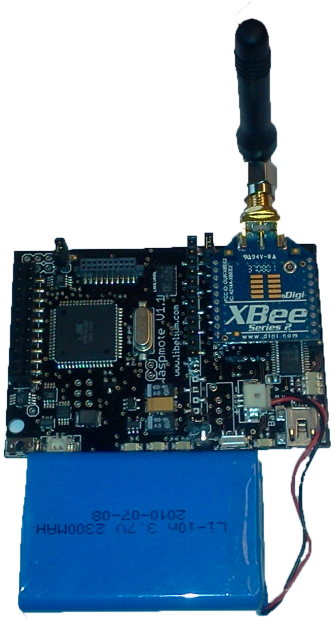

Figure 4.1: A Libelium Waspmote used in our implementation.

The Libelium Waspmotes are based on an open source project named Arduino[6]. Arduino is a platform designed for sensing electronics. It consists of two components: an Arduino board and an Arduino integrated development environment (IDE). The board of Waspmotes was designed by Libelium, while the Waspmote IDE is based on Arduino IDE. On top of Arduino IDE Libelium provides a set of APIs for application developers. The APIs are also open source and include functions to handle sensor utilities and ZigBee communication [78, 79].

[4] http://www.digi.com/

[5] http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module.jsp

[6] http://www.arduino.cc/

4.1.2 Proxy and Wide Area Nodes

The proxy nodes are implemented with Gumstix[7] Overo™ Earth computer-on-module[8] developing board. It ships with an ARM Cortex™ A8 CPU (OMAP3503 applications processor[9] from Texas Instruments[10]), with a clock rate of 600 MHz. It has 256MB of RAM and 256MB of flash memory. In addition, Overo Earth supports other extension boards. For example, Gumstix Tobi[11] can be mounted to Overo Earth and it provides support of digital visual interface (DVI) display, Ethernet, USB console, stereo audio, and so on. The OS on Overo Earth is a Linux distribution using opkg[12] package management system. We use cacao[13] Java virtual machine in order to run Java applications. The 3G data communication is provided by a 3G USB

Figure 4.2: The hardware of the proxy prototype, where a Libelium Waspmote Gateway and a 3G dongle are connected to a Gumstix Overo Earth through a USB hub.

[7] http://www.gumstix.com/
[8] http://www.gumstix.com/store/product_info.php?products_id=211
[9] http://focus.ti.com/docs/prod/folders/print/omap3503.html
[10] http://www.ti.com/
[11] http://www.gumstix.com/store/product_info.php?products_id=230
[12] http://wiki.openmoko.org/wiki/Opkg
[13] http://www.cacaovm.org/

dongle.

In order for the proxy node to communicate with the WPAN, a ZigBee communication module is needed. We implement this with a Libelium Waspmote Gateway. It is an XBee ZB transceiver mounted on a USB dongle. Libelium does not provide any APIs to operate the XBee module of the Waspmote Gateway. For example, when receiving data from the XBee module, the data frame is read directly from the USB serial port; likewise, we need to construct data frames by ourselves when sending data to the XBee module. To ease the XBee communication process and to focus on our proxy prototype, we use an open source API named *xbee-api* [80]. Although this API is not complete for all of our purposes, it provides functionality to send and receive data, and ease the management of the XBee coordinator. The xbee-api uses RXTX[14] to enable data communication via the USB serial port.

SNMP4J [81] is used for node management purposes. An SNMP4J agent runs on every proxy node and wide area node. We choose JCoAP [82] as the CoAP stack implementation for our prototype because it is under active development and supports all the features specified by CoAP RFC drafts at the time of writing this thesis.

4.1.3 Prototype Architecture

We implemented a prototype DHT-based M2M system, the architecture of which is shown in Figure 4.3. Since there will be many processes running on the proxy or wide area nodes, inter-process communication (IPC) is needed. In this prototype we use Java to implement most of the logic module, as a result, we choose Java remote method invocation (RMI) [83] as the IPC mechanism. The protocol stack used for the implementation is shown in Figure 4.4.

4.2 Proxy Start Up

When the proxy starts up, it first loads a proxy configuration from a file specified by the user. The configuration file contains information about the M2M CE, the XBee coordinator and the proxy. It specifies which name and URI to look for a M2M CE instance with Java RMI. It also defines which port the XBee coordinator uses to connect to the Gumstix and what baud

[14] `http://users.frii.com/jarvi/rxtx/`

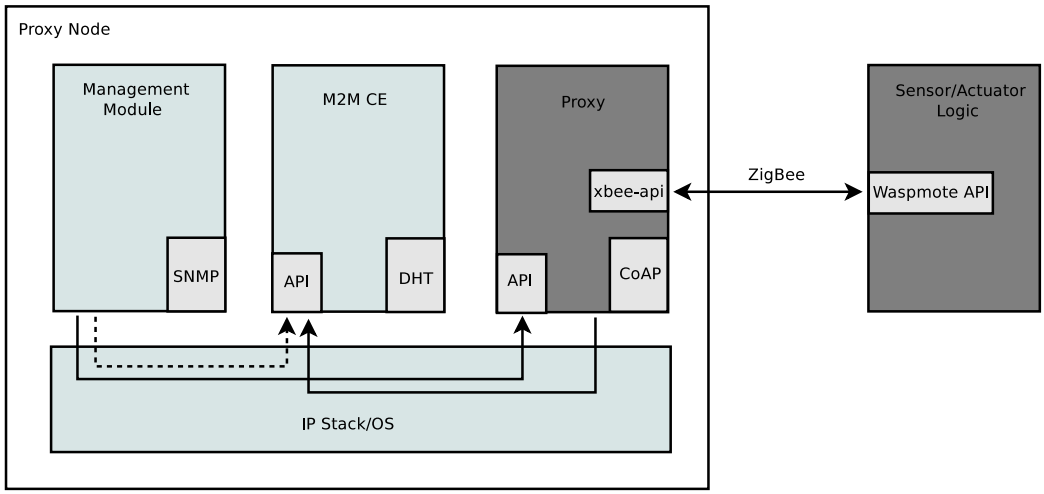

Figure 4.3: A prototype of the DHT-based M2M system, where a proxy is used to make WPAN nodes globally addressable and accessible from a 3G WWAN.

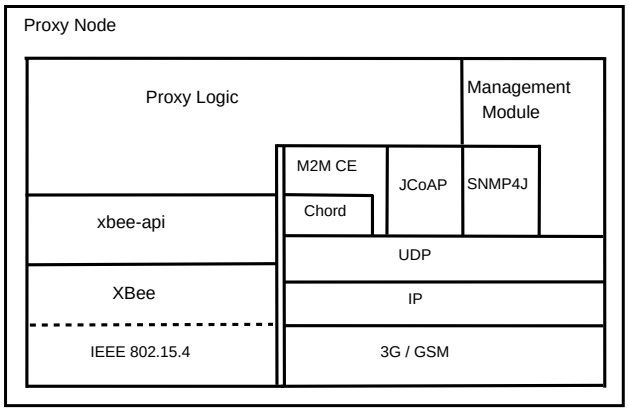

Figure 4.4: A prototype used for our implementation.

rate should be used for serial communication. Finally the configuration file specifies the CoAP name of the proxy and Java RMI information so that the proxy instance can be called from another application process.

The proxy then gets a handle on the M2M CE instance with Java RMI and joins the DHT overlay with its name specified in the configuration file. After it becomes a part of the overlay it is ready to announce WPAN nodes to the overlay. In addition, a CoAP server starts listening on a certain port to accept incoming requests from other proxy peers or WWAN nodes.

4.3 WPAN Node Joining and Leaving

Initially when the proxy starts up, it allows XBee nodes with correct security options to join the WPAN. Once an XBee node joins, it is announced to the overlay by the proxy with a predefined CoAP name. This name can be later configured using SNMP. However, it is not implemented in this implementation.

An XBee coordinator checks its network configuration (PAN ID, operating channel, stack profile, and security options) after a reset or power cycle. If any parameter is invalid then the coordinator tries to start up another network, unless the network settings have been written to non-volatile memory. An XBee end device runs in a similar way as the coordinator, except that it tries to join another network instead of starting another network when the parameters are not valid after a reset or power cycle.

When an XBee coordinator starts up a network and end devices join the network, they will send ZDO "device announcement" messages to the coordinator. After the XBee coordinator is reset or power cycled, then the end devices do not send a "device announcement" to it when they are started. Instead, when many-to-one routing is used in the XBee network they send the coordinator a "route record indicator" messages, which carries a distance-vector like information indicating all the nodes the message have passed by. When a "route record indicator" message is received by the coordinator, we consider an XBee node has (re)joined the network.

4.4 Application Logic

4.4.1 CoAP Message Parsing

When the proxy receives a message it checks the destination of the message. More specifically, a "URI-Host" option defines the destination. If the final destination is the proxy, it digests the message itself. On the other hand, if the destination is a WPAN node, the proxy then translates the CoAP name into ZigBee 64-bit address and relays the message to the corresponding XBee node.

4.4.2 Waspmote Application Packet Fragmentation and Reassembling

Libelium introduces an application header to deal with fragmentation and reassembly. This header is shown as in Figure 4.5.

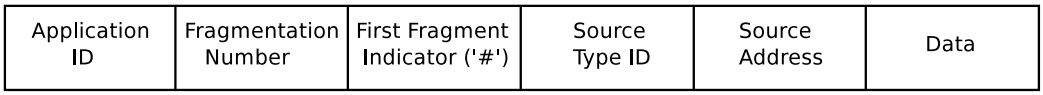

Figure 4.5: Waspmote application header.

In this header an application is used in the receiver differentiates different packets. When the data to be transferred is longer than the maximum payload it has to be fragmented. A fragment number and first fragment indicator fulfill this purpose. The original sender fills the source type (64-bit MAC, 16-bit network address, or maximum 20-byte node identifier string) and source address. Finally, the data field carries the actual payload.

On the proxy side xbee-api does not provide any APIs to operate on Waspmote application packets. As a result we implemented those functions to send and receive Waspmote application packets. These functions deal specifically with fragmentation and assembly of Waspmote application packets.

Chapter 5

Discussions

We have stated in Section 3.3 that principles including power efficiency, interoperability, scalability, security, and reliability have affected the proxy design. Some design choices are results endeavor of conforming to one or more principles. For instance, the introduction of caching mechanism in the system tries to improve both power efficiency and reliability (and potentially scalability as well). However, the focus of this thesis project has been system architecture, design, and prototype implementation. We have confirmed the basic functionality of the prototype and carried out some preliminary performance measurements on this prototype.

In the remaining of this chapter we discuss the functionality and performance evaluation we have done, what are the aspects that may affect system performance, and our thoughts on power sources.

5.1 Functionality Evaluation

We stated in Section 3 that the goal of this thesis project is to make resource constrained WPAN devices accessible from a larger network. In our design, sensors/actuators are able to join a WPAN managed by a proxy, which announces them to a DHT-overlay that runs on top of IP. The proxy is able to discover services available in a sensor/actuator and announce these services to the DHT-overlay as well.

Simple applications have been developed to test the functionality of the proxy. For example, we have two sample scenarios shown in Figure 5.1 and Figure 5.2, in which nodes from one WPAN are able to communicate transparently with nodes in a WWAN. In Figure 5.1 we have verified a scenario

where a Waspmote detects motion and sends a message to a Gumstix node connected to an IP network. Specifically, when the Waspmote is moved or shook, it sends a message "Motion detected" to a Gumstix node (the destination is hard-coded at the moment). This message is handled by the Waspmote's proxy node. The proxy first gets the IP address of the Gumstix node, then it sends a CoAP message to the IP address.

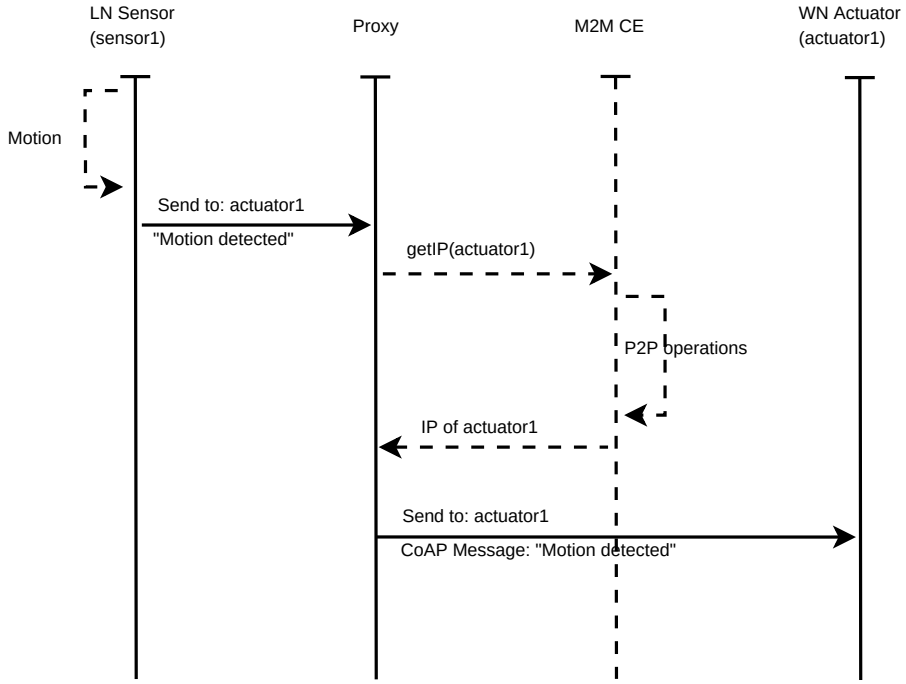

Figure 5.1: A sample scenario implemented in our prototype, where a local WPAN sensor sends message to a WWAN actuator.

Figure 5.2 describes another scenario where the data flow is from a Gumstix to a Waspmote in a WPAN. Users can specify which WPAN node the Gumstix should send a message to, using a command line interface (this interface will be updated with a button in our following work). This Gumstix then looks up the IP of the proxy responsible for the WPAN node and sends a CoAP message to the proxy, specifying which WPAN node is the final destination. Upon receiving the message, the proxy looks up the ZigBee address of the Waspmote and sends a ZigBee message to the corresponding Waspmote. The Waspmote blinks its LED lights if the message is "Blink".

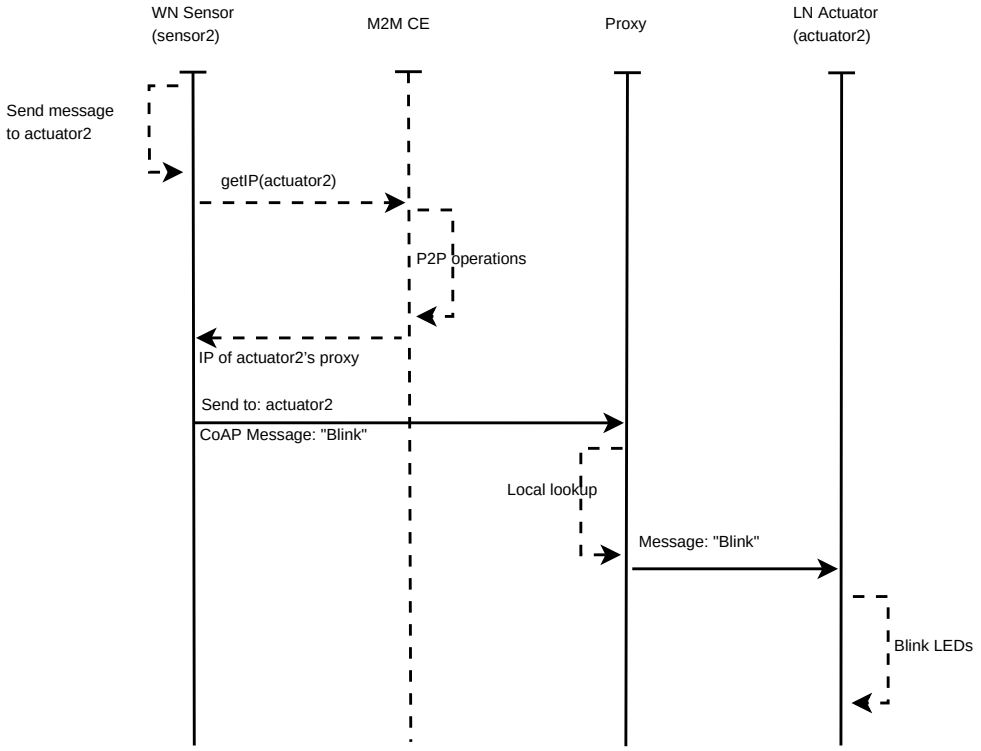

Figure 5.2: A sample scenario implemented in our prototype, where a WWAN sensor sends message to a local WPAN actuator.

5.2 Performance Measurements

We did some preliminary measurements investigating the performance of our prototype. More specifically, we measured how the number of nodes in the DHT overlay affects the time it takes for one node to look up the IP address of another node, and how the number of nodes in the DHT and CoAP message size affect the round trip time (RTT) between a WWAN node and a WPAN node. We will present our result in the remaining of this section.

5.2.1 Node Lookup Time

We set up an environment to investigate the relationship between the number of nodes in the DHT and the time it takes for one node A to get the IP address of another node B by looking up the node B's name in the DHT. In this environment we have two Gumstix nodes running Chord DHT. We

have a personal computer (PC) in the same network as those two Gumstix nodes. The connection was Ethernet instead of 3G, since our operator has blocked most of the ports we needed to run DHT algorithms (when we are doing the evaluation). On this PC a number of virtual nodes are started. For the first measurement there are 8 virtual DHT nodes running on the PC, and 18 virtual nodes in the second measurement, 28 virtual nodes in the third measurement, and so forth. We look up the IP address of Gumstix A from Gumstix node B and note down the time this process takes. Finally we plotted a figure showing how the number of DHT nodes affects node lookup time, as shown in Figure 5.3.

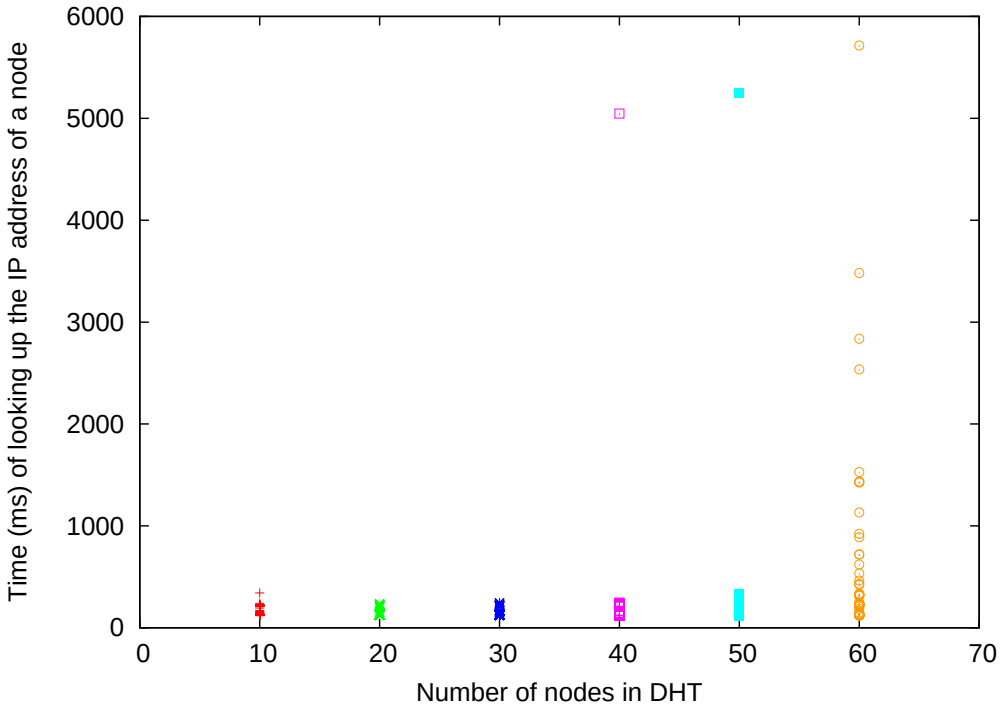

Figure 5.3: The number of nodes in the DHT does not significantly affect node lookup time, but he stability of node looking up decreases as the number of DHT nodes increases.

We can see from Figure 5.3 that the both the average and medium lookup time is around 200 milliseconds when there are no more than 30 nodes in the DHT. Almost all lookups were successful within 300 milliseconds. However, as the number of DHT nodes increases to 40, the average lookup time remains low, while the medium lookup time increases significantly because of one peek value in one case where the lookup takes more than 5000 milliseconds. This

abnormality might have been caused by busy process scheduling on the PC on which the virtual DHT nodes run. Similarly, when the number of DHT nodes increases to 50, another peek value is found. When the number of DHT nodes increases to 60, the average lookup time increases not so much, but the medium lookup time increases much and the variations are large compared with the testing case where there are no more than 30 DHT nodes. We believe these phenomena are caused by the depletion of resources (CPU slots and memory) on the PC running virtual DHT nodes. In fact, when we had 58 virtual nodes running on a single PC, it was too busy running the virtual nodes to response to external user input, for example, keyboard interruptions. From this observation we feel that the node lookup time is low enough in most cases, when the number of DHT nodes is small. Further measurements need to be carried out with the actual network factors involving into the test cases, for example, by running a number of virtual DHT nodes on different PCs placed in a network.

5.2.2 RTT between WWAN and WPAN Nodes

Next we measure the RTT between a WWAN node and a WPAN node, using the same environment as we did in the previous sub-section. This time we still simulate virtual DHT nodes on a PC, and try to investigate how the number of DHT nodes and CoAP message size affect the RTT from a Gumstix node to a Waspmote. We generate random character strings with size from 1 to 180 bytes and note down the time the messages travel from the Gumstix to the Waspmote and then get back to the original Gumstix node. We have plotted the collected data in Figure 5.4.

From Figure 5.4 we see that RTT is not significantly affected by the number of nodes in the DHT, as the curves show similar behaviors whether there are 10 or 60 DHT nodes. It should not be a surprise since the measured RTT does not include node IP lookup time. Furthermore, the DHT nodes are mostly virtual nodes and therefore the number of nodes does not significantly influence traffic volume in the network. Note that some messages larger than 150 bytes yield exceptionally long RTT. It happens with scenarios where 10, 20, or 30 DHT nodes participate in the network. This happens because of some CoAP message timeout and retransmissions (see the following discussions).

We have two test cases in which the RTT between two WWAN nodes is measured. Specifically, we noted down the time it takes for CoAP messages to travel from one Gumstix to another Gumstix proxy node in the same network and then get back to the original Gumstix. This is done by letting the

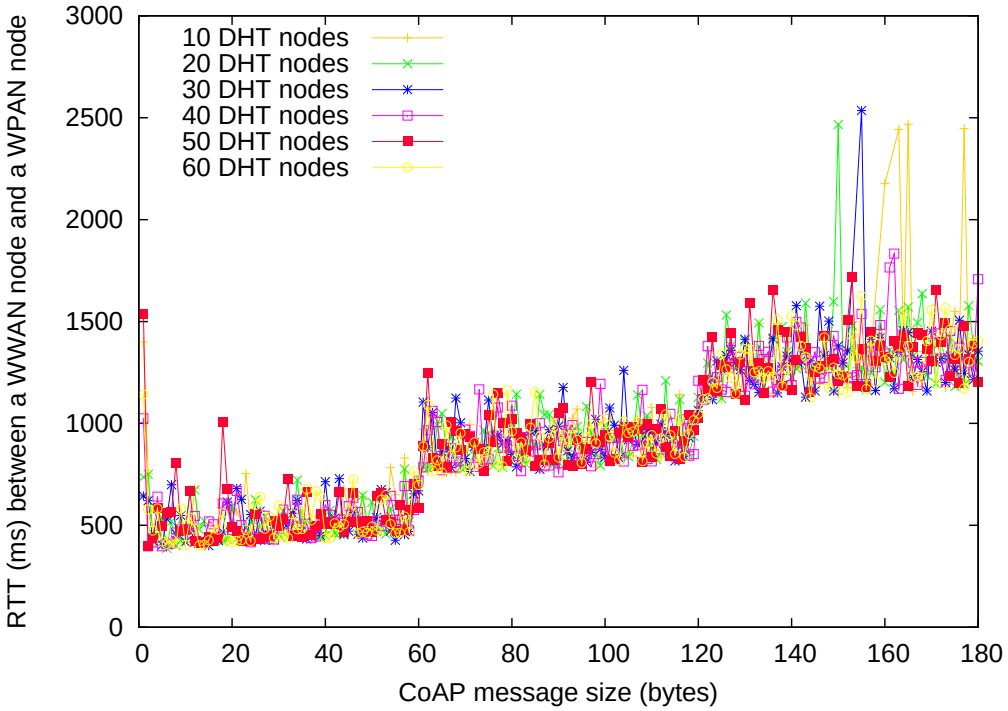

Figure 5.4: RTT is affected more significantly by the size of CoAP messages rather than the number of DHT nodes, when the majority of DHT nodes are virtual nodes residing on a single PC.

Gumstix proxy to reply with the same CoAP message it receives if messages are larger than 180 bytes, without sending the message to a WPAN node. That is, CoAP messages no more than 180 bytes are delivered to WPAN nodes (results have been plotted in Figure 5.4) but larger messages are simply returned to the sender. The measured data is plotted in Figure 5.5. We notice from Figure 5.5 that one CoAP message yields a RTT of almost 650 milliseconds. However, if we leave this abnormality out, the medium RTT for both test cases is around 350 milliseconds and the average RTT is around 300 milliseconds. Another phenomenon we can see from Figure 5.5 is that CoAP message size does not seem to have a significant impact on the RTT (when the CoAP messages are between 180 and 300 bytes). Although we do not have any measured data on how it goes with CoAP messages less than 180 bytes, we believe it is safe to assume that the medium RTT between two WWAN nodes will be around 350 milliseconds and the average RTT will be around 300 milliseconds, assuming no CoAP messages time out or need to be retransmitted.

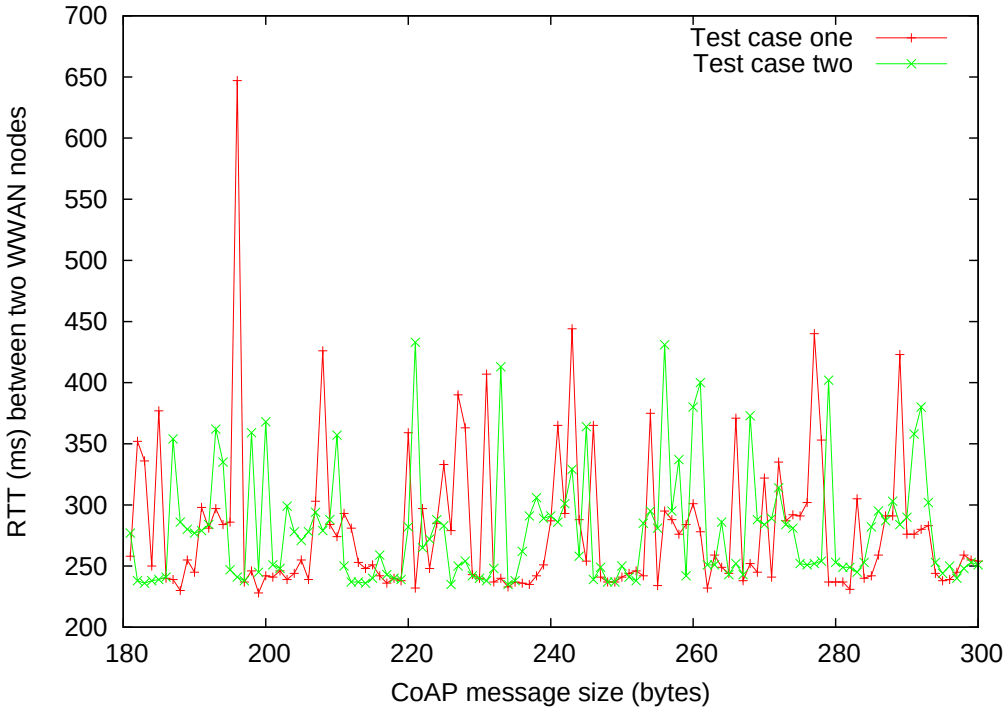

Figure 5.5: RTT between two WWAN nodes is not significantly influenced by CoAP message size, when CoAP messages are between 180 and 300 bytes.

Since those curves in Figure 5.4 are so similar with each other, now we will just analyze one curve (see Figure 5.6), which shows the relationship between the size of CoAP messages and RTT, when there are 50 nodes in the DHT. The RTT is quite large when the Gumstix is sending one-byte CoAP messages. This is probably caused by the fact that neither the Gumstix sender or the receiving proxy node has fully loaded the relevant applications in their cache memory. After that the RTT does not change drastically until the CoAP message size reaches 62 bytes, where the RTT almost doubled. This is because the proxy has to fragment any messages larger than 61 bytes. The maximum Waspmotes message payload that may fit into one packet is 61 bytes in our prototype. Similarly, when the CoAP message size reaches 123 bytes, the RTT again goes onto another higher stage, since the maximum payload for the second (and following) fragments is 62 (rather than 61) bytes. We can also see from the curve that messages divided into the same number of fragments have similar RTT.

From the above discussions we may conclude the size of CoAP messages does not play an important role in affecting the RTT between two WWAN nodes,

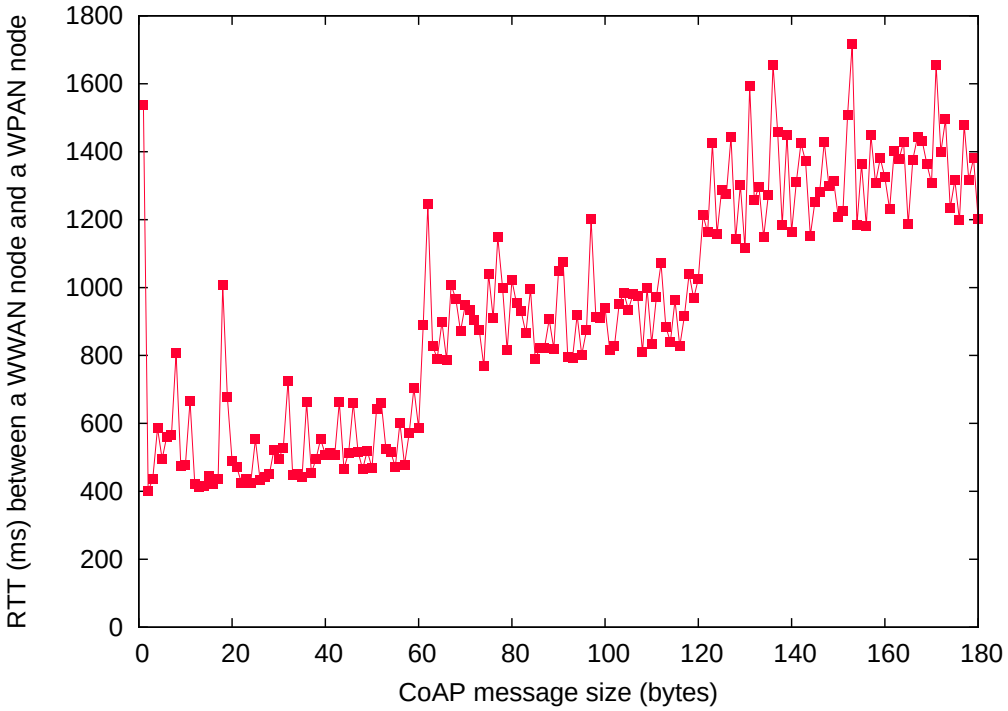

Figure 5.6: RTT is affected by the size of CoAP messages when the system has 50 DHT nodes (48 of which are virtual nodes residing on a single PC).

especially for small messages. However, the message size indeed influence the RTT between a WWAN node and a WPAN node. To be exact, RTT increases as the number of fragments the proxy have to divide the message into increases. The RTT is acceptable in most cases given the hardware and software we use. Nevertheless, the system performance may be further improved by disabling unnecessary logging and debugging functions. Due to the fact that the virtual DHT nodes do not significantly influence network traffic, we can not claim that the number of DHT nodes does not significantly affect the RTT between a WWAN node and a WPAN node. We still need to investigate this with more measurements in scenarios with an extremely large number of DHT nodes and practical network environment.

5.3 Performance Discussions

Although we have just done some preliminary performance evaluation of the prototype, it is still worthwhile discussing what factors may affect our

system's performance. In this section we specifically discuss in two aspects: proxy throughput and reliability.

5.3.1 Proxy Throughput

DHT-based P2P systems are often thought to be scalable. However, the use of DHT in our system does not immediately imply that our system is scalable. The DHT provides scalability in the perspective of connecting a large number of WWAN nodes to the system. In this section we discuss what can be the bottlenecks that may hinder scalability of our design. The proxy in our system are responsible of WPAN management and act as a gateway for WPAN nodes. All requests from or to a WPAN are processed by the proxy. As a result, how fast the proxy can process these requests affects the overall performance of the M2M system.

We expect an increase of requests to the proxy as the number of WPAN nodes relying on that proxy grows. Similarly, when the number of WWAN nodes grows larger and the communication between WWAN nodes and WPAN nodes become more frequent, the proxy should also expect a grow of requests. Ideally, the proxy should be able to process as many requests as possible and produce low delays. Otherwise the performance of the M2M system is downgraded when proxy throughput can not match the increase of requests.

In order to increase proxy throughput, we may need to use more advanced hardware for proxies and switch to technologies providing more incoming and outgoing bandwidth (other than 3G). However, this will obviously increase the cost of proxies and eventually the cost of the system. An alternative is to introduce more proxies when more WPAN nodes need to join the system. This alternative attacks the problem by slightly shifting the scalability issue of proxies to the DHT.

5.3.2 Proxy Reliability

Since a proxy reflects a single point of failure in our design, caching mechanisms are introduced so tackle this problem. Caching partly remedies this problem at the cost of potentially obsolete data. That is, caching in DHT (see Section 3.5.4.2) may be useful for other nodes to get a data copy collected by a WPAN node even if the proxy fails. However, this does not help when WPAN management operations are need on a WPAN when its proxy has failed.

One way to remedy the above problem is introducing redundancy on proxies. Unfortunately in our case this is not feasible since the proxy acts as the coordinator node, while only one coordinator is allowed for each ZigBee network. Another option is to ensure fast recovery of the proxy. Fast recovery implies fast detection of proxy failure. Fast failure detection can be implemented using underlying DHT algorithms. For instance, Chord ensure robustness in face of node failures; and this mechanism can be used for node failure detection. Once node failure is detected, alarm messages can be sent to (predefined or dynamically configured) administrators. Then it should be easier to diagnose what has failed and bring up the node again.

5.4 Power Source

In our design the proxies are supposed to be mains powered. It may be desirable to have battery powered proxies in some scenarios, for example where no electric sockets are available. In such cases renewable energy sources may come to help. For example, we can use solar panel batteries to provide power the proxies.

The WPAN sensors are expected to last. Generally speaking, these sensor are in sleeping mode most of the time in order to save energy. For sensors that may consume much energy, we can consider using wireless charging technologies [84], which is a quickly emerging and promising technology.

Chapter 6

Conclusions and Future Work

This study was a part of the Devices and Interoperability EcosysteM (DIEM) program[1] sponsored by the Finnish Funding Agency for Technology and Innovation (TEKES)[2] as well as other industrial and research parties. The study at NomadicLab of Ericsson Finland focuses on finding a solution to move the intelligence of an M2M system from a centralized server to the network edge. We proposed a DHT-based M2M system to fulfill this purpose. This project has been further divided into three master's thesis topics: one topic focuses on system management applications; one focuses on distributed intelligence; and finally this thesis study focuses on the network edge, i.e., the proxy and end WPAN nodes. We close this thesis with what we have done in this thesis project, our conclusions, and what future work is needed.

6.1 Summary

We have done a study on DHT-based M2M networks in this thesis project. More specifically, we focused on the network edge and introduced a proxy in such networks. We discussed why proxies are necessary for these systems and why those efforts are essential. We have kept in mind principles including power efficiency, interoperability, scalability, security, and reliability during this study. Based on these principles, we proposed our design of the proxy, implemented a prototype, and finally analyzed our design and implementation.

A proxy is necessary since WPAN nodes can not run P2P algorithms due to

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

resource constraints. In addition, proxies are good for data management and security. They also simplify application logic on WSN nodes and connect heterogeneous WPAN nodes to the M2M system. Although some of the above functionality may be realized by nodes other than a proxy, they are likely to add complexity to the whole system. As a result, a proxy can be a good tradeoff to keep the WPAN node logic simple and at the same time avoid complicating the M2M system too much.

The design of our proxy mainly focuses on WPAN node management and service management. The proxy acts as both the coordinator and data collector of the WPAN. It is responsible for configuring the WPAN and allows or disallows new WPAN devices joining the network. In our design node joining is done by applying security options as an access control mechanism, i.e., only nodes with the same shared security keys as the proxy may join the WPAN. After one node has joined a WPAN, it may voluntarily or involuntarily leave the network. When a node voluntarily leaves, it may either inform the proxy or not. The proxy provides mechanisms to detect whether a node has left the WPAN. The proxy is also responsible for discovering available services one WPAN provides. This is dependent on the underlying WPAN protocols, for example, in CoAP resource discovery can be used and in ZigBee ZDO is our choice.

Caching mechanisms have been introduced in order to provide better performance. One proxy can cache data both for WPAN nodes and in DHT. Caching for WPAN nodes saves the efforts of querying the WPAN all the time; caching in DHT improves system reliability. Security has also been a concern in the proxy design. The secure communication between a proxy and its WWAN peers takes advantage of security mechanisms specified either by CoAP or SNMP; and the secure communication between the WPAN nodes is ensure by WPAN protocols (for example ZigBee security specifications).

Based on our design, a prototype proxy was implemented using Gumstix boards and Waspmote sensor nodes. The implementation took advantage of inexpensive hardware and free/open source software in order to keep the cost low. We have confirmed the basic functionality of the prototype and made some discussions regarding the design and the prototype. The focus of this thesis project has been system architecture, design, and prototype implementation, while not so much performance evaluation has been conducted. However, a few preliminary performance measurements have been carried out. We believe that it should be feasible to use our prototype in practical scenarios.

6.2 Future Work

Since no performance evaluation has been done in this thesis work, this should be carried out in the following study. Specifically, the evaluation should focus on power consumption of WPAN nodes, delays the proxy introduces, proxy throughput, comparison of proxy with and without caching mechanisms, and system scalability. Although ZigBee devices are designed to last several months or even years with batteries, different applications may have different battery lifetime. Further investigations are needed in order to provide concrete evidence on which application scenarios are appropriate. The proxy's performance should be further measured from the perspective of delay and throughput, since these factors affect system's performance as well as scalability.

Our design mainly uses ZigBee as the WPAN protocol. In the future more WPAN protocols should be included in the system, for example, 6LoWPAN, Z-Wave, Bluetooth, and so on. When we think about IoT or M2M, devices in either system are heterogeneous. Various protocols have been developed for different purposes. However, devices running on top of these protocols can be integrated into one system by making the proxy support them. One proxy node may not support different protocols at the same time since it may add complexity to the proxy logic. However, proxies supporting different underlying WPAN protocols will easily embrace different devices and application scenarios into one system, making the system truly intelligent and more ubiquitous.

References

[1] 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. Version 1.1, http://www.iot-visitthefuture.eu/fileadmin/documents/researchforeurope/270808_IoT_in_2020_Workshop_Report_V1-1.pdf.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, pp. 2787–2805, October 2010.

[3] S. E. Sarma, S. A. Weis, and D. W. Engels, "RFID Systems and Security and Privacy Implications," in *In Workshop on Cryptographic Hardware and Embedded Systems*, pp. 454–470, Springer, 2002.

[4] J. Conti, "The Internet of Things," *Communications Engineer*, vol. 4, pp. 20 –25, December – January 2006.

[5] A. Walter-Krisch, "Heading towards 50 billion connections." Ericsson OSS/BSS: http://www.ericsson.com/campaign/opportunitysupportsystems/newsfeed/posts/15/. Accessed February 14, 2011.

[6] B. Mellor, "A world of connections," *The Economist*, April 2007.

[7] G. Lawton, "Machine-to-machine technology gears up for growth," *Computer*, vol. 37, pp. 12 – 15, September 2004.

[8] S. Gupta and A. Hirdesh, "Overview of M2M." Ankit Hirdesh Papers website: http://hriday.ankit.googlepages.com/M2M_overview_paper.pdf. Accessed February 16, 2011.

[9] I. Brezeanu and G. Gorghiu, "Machine to machine applications in enterprise a new step to intelligent internet control," in *Proceedings of the 1st*

Workshop on Energy, Transport and Environment Control Applications (ETECA'09), (Targoviste, Romania), pp. 30–45, May 2009.

[10] G. Privat, "From Smart Devices to Ambient Communication." Workshop "From RFID to the Internet of Things". Brussels, Belgium, March 2006.

[11] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, pp. 102–114, August 2002.

[12] J. Edgar H. Callaway, *Wireless Sensor Networks: Architectures and Protocols*. AUERBACH, August 2003.

[13] E. M. Royer and C.-K. Toh, "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks," *IEEE Personal Communications*, vol. 6, pp. 46–55, 1999.

[14] L. Song and D. Hatzinakos, "Architecture of Wireless Sensor Networks With Mobile Sinks: Sparsely Deployed Sensors," *IEEE Transactions on Vehicular Technology*, vol. 56, pp. 1826–1836, 2007.

[15] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, pp. 325–349, 2005.

[16] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman, "A survey of gossiping and broadcasting in communication networks," *Networks*, vol. 18, no. 4, pp. 319–349, 1988.

[17] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, (New York, NY, USA), pp. 174–185, ACM, 1999.

[18] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, (New York, NY, USA), pp. 56–67, ACM, 2000.

[19] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, p. 10 pp. vol.2, January 2000.

[20] S. Lindsey and C. Raghavendra, "Pegasis: Power-efficient gathering in sensor information systems," in *Aerospace Conference Proceedings, 2002. IEEE*, vol. 3, pp. 1125 – 1130, 2002.

[21] A. Manjeshwar and D. Agrawal, "Teen: a routing protocol for enhanced efficiency in wireless sensor networks," in *Parallel and Distributed Processing Symposium., Proceedings 15th International*, pp. 2009 – 2015, April 2001.

[22] V. Rodoplu and T. Meng, "Minimum energy mobile wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 17, pp. 1333 – 1344, August 1999.

[23] L. Li and J. Halpern, "Minimum-energy mobile wireless networks revisited," in *Communications, 2001. ICC 2001. IEEE International Conference on*, vol. 1, pp. 278 –283, June 2001.

[24] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, (New York, NY, USA), pp. 70–84, ACM, 2001.

[25] Y. Yu, R. Govindan, and D. Estrin, "Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks," tech. rep., UCLA Computer Science Department, 2001.

[26] J.-H. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 12, pp. 609–619, August 2004.

[27] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Maximum lifetime data gathering and aggregation in wireless sensor networks," *Networks The Proceedings of the Joint International Conference on Wireless LANs and Home Networks ICWLHN 2002 and Networking ICN 2002*, pp. 685–696, 2002.

[28] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications*, vol. 7, pp. 16–27, 2000.

[29] J. A. Gutiérrez, E. H. Callaway, and R. L. Barrett, *Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4*. IEEE Standards, USA: Standards Information Network - IEEE Press, 1. ed., 2003.

[30] LAN/MAN Standards Committee of the IEEE Computer Society, *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).* New York, USA: IEEE, IEEE Std 802.15.4-2006 ed., 2006.

[31] LAN/MAN Standards Committee of the IEEE Computer Society, *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).* New York, USA: IEEE, IEEE Std 802.15.4-2003 ed., 2003.

[32] H. Zimmermann, "Innovations in Internetworking," ch. OSI reference model – The ISO model of architecture for open systems interconnection, pp. 2–9, Norwood, MA, USA: Artech House, Inc., 1988.

[33] S. C. Ergen, "ZigBee/IEEE 802.15.4 Summary," September 2004. `http://pages.cs.wisc.edu/~suman/courses/838/papers/zigbee.pdf`. Accessed March 12, 2011.

[34] ZigBee Alliance, "The Alliance," `http://www.zigbee.org/About/AboutAlliance/TheAlliance.aspx`. Accessed April 4, 2011.

[35] A. Elahi and A. Gschwender, *ZigBee Wireless Sensor and Control Network*. Prentice Hall Communications Engineering and Emerging Technologies Series, Prentice Hall, 2009.

[36] ZigBee Standards Organization, *ZigBee Specification, Document 053474r17.* California, USA: `http://www.zigbee.org/Specifications.aspx`, January 2008.

[37] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," RFC 4919, Internet Engineering Task Force, August 2007.

[38] S. Deering and R. Hinden, "Internet protocol, version 6 (IPv6) specification," RFC 2460, Internet Engineering Task Force, December 1998.

[39] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944, Internet Engineering Task Force, September 2007.

[40] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained Application Protocol (CoAP), work in progress," draft-ietf-core-coap-05, Internet Engineering Task Force, 2011.

[41] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, Irvine, California, 2000.

[42] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – HTTP/1.1," RFC 2616, Internet Engineering Task Force, June 1999.

[43] Z. Shelby, "CoRE Link Format, work in progress," draft-ietf-core-link-format-03, Internet Engineering Task Force, 2011.

[44] G. S. Manku, *Dipsea: a modular distributed hash table*. PhD thesis, Stanford University, August 2004.

[45] M. Naor and U. Wieder, "Novel Architectures for P2P Applications: the Continuous-Discrete Approach," *ACM TRANSACTIONS ON ALGORITHMS*, vol. 3, no. 3, pp. 50–59, 2007.

[46] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking up data in P2P systems," *Communications of the ACM*, vol. 46, pp. 43–48, February 2003.

[47] R. Rivest, "The MD5 Message-Digest algorithm," RFC 1321, Internet Engineering Task Force, April 1992.

[48] R. Oppliger, *SSL and TLS: Theory and Practice*. Norwood, MA, USA: Artech House, Inc., 2009.

[49] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, Internet Engineering Task Force, December 2009.

[50] P. Gutmann and P. Gutmann, "Software generation of practically strong random numbers," in *In Proceedings of the 8th USENIX Security Symposium*, pp. 243–257, 1998.

[51] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, (New York, NY, USA), pp. 654–663, ACM, 1997.

[52] T. White, "Consistent hashing," `http://www.lexemetech.com/2007/11/consistent-hashing.html`. Accessed February 21, 2011.

[53] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, (New York, NY, USA), pp. 149–160, ACM, 2001.

[54] D. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi, "Web caching with consistent hashing," *Computer Networks*, vol. 31, no. 11-16, pp. 1203 – 1213, 1999.

[55] W. Peterson and D. Brown, "Cyclic Codes for Error Detection," *Proceedings of the Institute of Radio Engineers*, vol. 49, pp. 228 –235, January 1961.

[56] D. Eastlake and P. Jones, "US secure hash algorithm 1 (SHA1)," RFC 3174, Internet Engineering Task Force, September 2001.

[57] F. Dabek, E. Brunskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan, "Building peer-to-peer systems with chord, a distributed lookup service," in *Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, pp. 71–76, 2001.

[58] J. Hautakorpi and G. Camarillo, "Evaluation of DHTs from the viewpoint of interpersonal communications," in *Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*, MUM '07, (New York, NY, USA), pp. 74–83, ACM, 2007.

[59] J. Case, M. Fedor, M. Schoffstall, and C. Davin, "Simple network management protocol (SNMP)," RFC 1157, Internet Engineering Task Force, May 1990.

[60] D. Harrington, R. Presuhn, and B. Wijnen, "An architecture for describing simple network management protocol (SNMP) management frameworks," RFC 3411, Internet Engineering Task Force, December 2002.

[61] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, "Protocol operations for version 2 of the simple network management protocol (SNMPv2)," RFC 1905, Internet Engineering Task Force, Januaray 1996.

[62] J. Mäenpää and J. J. Bolonio, "Performance of resource location and discovery (reload) on mobile phones," in *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pp. 1 – 6, April 2010.

[63] C. Jennings, B. B. Lowekamp, E. Rescorla, S. A. Baset, and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol, work in progress," draft-ietf-p2psip-base-15, Internet Engineering Task Force, May 2011.

[64] S. W. Ambler, *The Object Primer : Agile Model-Driven Development with UML 2.0*. Cambridge University Press, 2004.

[65] Digi International, "XBee® /XBee-PRO® ZB RF Modules," November 2010. `http://ftp1.digi.com/support/documentation/90000976_G.pdf`. Accessed April 26, 2011.

[66] Libelium Comunicaciones Distribuidas S.L., "Over the Air Programming with 802.15.4 and ZigBee: Laying the groundwork," May 2011. `http://www.libelium.com/documentation/waspmote/over_the_air_programming.pdf`. Accessed June 16, 2011.

[67] B. M. Hauzeur, "A model for naming, addressing and routing," *ACM Transactions on Information Systems (TOIS)*, vol. 4, pp. 293–311, December 1986.

[68] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand distance vector (AODV) routing," RFC 3561, Internet Engineering Task Force, July 2003.

[69] C. Frank, V. Handziski, and H. Karl, "Service discovery in wireless sensor networks," tech. rep., Technical University Berlin, 2004.

[70] M. Botts and A. Robin, "SensorML." `http://www.ogcnetwork.net/SensorML`. Accessed June 19, 2011.

[71] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security," RFC 4347, Internet Engineering Task Force, April 2006.

[72] K. McCloghrie, "An administrative infrastructure for SNMPv2," RFC 1909, Internet Engineering Task Force, 1996.

[73] G. Waters, "User-based security model for SNMPv2," RFC 1910, Internet Engineering Task Force, 1996.

[74] U. Blumenthal and B. Wijnen, "User-based security model (USM) for version 3 of the simple network management protocol (SNMPv3)," RFC 3414, Internet Engineering Task Force, 2002.

[75] U. Blumenthal, F. Maino, and K. McCloghrie, "The advanced encryption standard (AES) cipher algorithm in the SNMP user-based security model," RFC 3826, Internet Engineering Task Force, June 2004.

[76] J. S. Reddy, "ZigBee Security," 2004. `http://www.zigbee.org/imwp/idms/popups/pop_download.asp?contentID=9436`. Accessed June 12, 2011.

[77] Libelium Comunicaciones Distribuidas S.L., "Waspmote Datasheet," March 2011. `http://www.libelium.com/documentation/waspmote/waspmote-datasheet_eng.pdf`. Accessed April 26, 2011.

[78] Libelium Comunicaciones Distribuidas S.L., "Waspmote Technical Guide," October 2010. `http://www.libelium.com/documentation/waspmote/waspmote-technical_guide_eng.pdf`. Accessed April 26, 2011.

[79] Libelium Comunicaciones Distribuidas S.L., "Waspmote ZigBee Networking Guide," July 2010. `http://www.libelium.com/documentation/waspmote/waspmote-zigbee-networking_guide.pdf`. Accessed April 26, 2011.

[80] Google Code Project Home, "xbee-api: A Java API for Digi XBee/XBee-Pro OEM RF Modules." `http://code.google.com/p/xbee-api/`. Accessed June 18, 2011.

[81] SNMP4J.org, "The SNMP API for Java." `http://www.snmp4j.org/`. Accessed June 18, 2011.

[82] D. Pauli, "Java imlementation of CoAP." `https://github.com/dapaulid/JCoAP`. Accessed June 18, 2011.

[83] Oracle, "Remote Method Invocation Home." `http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html`. Accessed June 13, 2011.

[84] D. W. Harrist, "Wireless battery charging system using radio frequency energy harvesting," Master's thesis, University of Pittsburgh, USA, 2004.

[85] Trouble shooting - Rxtx. `http://rxtx.qbang.org/wiki/index.php/Trouble_shooting#How_does_rxtx_detect_ports.3F__Can_I_override_it.3F`. Accessed June 18, 2011.

[86] Libelium Comunicaciones Distribuidas S.L., "Waspmote Accelerometer Programming Guide," July 2010. `http://www.libelium.com/documentation/waspmote/waspmote-accelerometer-programming_guide.pdf`. Accessed April 26, 2011.

Appendix A

Implementation Issues

A.1 RXTX Port Scan

RXTX tries scanning all devices in /dev for potential ports by default. This process takes a long time (around two minutes) on Gumstix. One may restrict the scanning process in order to speed up the proxy logic [85]. In Java this can be done by specifying environment parameters "gnu.io.rxtx.SerialPorts" and "gnu.io.rxtx.ParallelPorts". For example, in our case we start up the proxy using the following command:

```
java -cp . \
     -Djava.security.policy=client.policy \
     -Dgnu.io.rxtx.SerialPorts=/dev/ttyUSB3 \
     -jar proxy.jar proxy.config
```

A.2 Bug in Waspmote API

There is a bug related to security key setting in Waspmote API version 0.15 and previous versions. Only the first eight bytes of the key will be used when setting the network key or link key. The problem lies in function "gen_data()" in lines 4355-4403 of "WaspXBeeCore.cpp" in Waspmote API version 0.15, as listed below.

```
1 /*
2  Function: Generates the API frame to send to the XBee module
3  Parameters:
4   data : The string that contains part of the API frame
```

```
5    param : The param to set
6   Returns: Nothing
7   Values: Stores in 'command' variable the API frame to send to
        the XBee module
8 */
9 void WaspXBeeCore::gen_data(const char* data, uint8_t* param)
10 {
11   uint8_t inc=0;
12   uint8_t inc2=0;
13
14   clearCommand();
15   it=0;
16   while(data[it] != '\0') {
17     inc++;
18     it++;
19   }
20   inc/=2;
21
22   while(inc2<inc){
23     command[inc2]=Utils.converter(data[2*inc2],data[2*inc2+1]);
24     inc2++;
25   }
26
27   if(inc>11)
28   {
29     for(it=0;it<8;it++)
30     {
31       command[inc-9+it]=param[it];
32     }
33   }
34   else if(inc==11)
35   {
36     for(it=0;it<3;it++)
37     {
38       command[inc-4+it]=param[it];
39     }
40   }
41   else if(inc==10)
42   {
43     for(it=0;it<2;it++)
44     {
45       command[inc-3+it]=param[it];
46     }
47   }
48   else command[inc-2]=param[0];
49 }
```

However, this bug has been fixed since Waspmote API version 0.16[1]. The fixed function "gen_data()" is listed as follows.

```
1  /*
2   Function: Generates the API frame to send to the XBee module
3   Parameters:
4    data : The string that contains part of the API frame
5    param : The param to set
6   Returns: Nothing
7   Values: Stores in 'command' variable the API frame to send to
        the XBee module
8  */
9  void WaspXBeeCore::gen_data(const char* data, uint8_t* param)
10 {
11      uint8_t inc=0;
12      uint8_t inc2=0;
13
14      clearCommand();
15      it=0;
16      while(data[it] != '\0') {
17           inc++;
18           it++;
19      }
20      inc/=2;
21
22      while(inc2<inc){
23           command[inc2]=Utils.converter(data[2*inc2],data[2*inc2
                 +1]);
24           inc2++;
25      }
26      // The following lines have been updated.
27      if(inc==24)
28      {
29           for(it=0;it<16;it++)
30           {
31                command[inc-17+it]=param[it];
32           }
33      }
34      else if(inc==16)
35      {
36           for(it=0;it<8;it++)
37           {
38                command[inc-9+it]=param[it];
39           }
40      }
41      else if(inc==11)
42      {
```

[1] http://www.libelium.com/downloads/Waspmote_changelog.txt

```
43            for(it=0;it<3;it++)
44            {
45                command[inc-4+it]=param[it];
46            }
47        }
48        else if(inc==10)
49        {
50            for(it=0;it<2;it++)
51            {
52                command[inc-3+it]=param[it];
53            }
54        }
55        else command[inc-2]=param[0];
56    }
```

A.3 Tweaks in Waspmote API

A.3.1 Direction Change Interruption Thresholds

We changed some thresholds in Waspmote API in order to detect more sensitive motion on the Waspmotes, as no API are provided to change them in the application [86]. Lines (357-378) in "WaspAcc.h" have been modified, as shown below.

```
1  #define DD_THSI_L_val          0x00
2
3  /*!  \def DD_THSI_H_val
4       \brief Direction Change Internal Threshold MSB
5
6       Direction Change Internal Threshold MSB.
7   */
8  #define DD_THSI_H_val          0x01
9
10 /*!  \def DD_THSE_L_val
11      \brief Direction Change External Threshold LSB
12
13      Direction Change External Threshold LSB.
14  */
15 #define DD_THSE_L_val          0x00
16
17 /*!  \def DD_THSE_H_val
18      \brief Direction Change External Threshold MSB
19
20      Direction Change External Threshold MSB.
```

```
21  */
22 #define DD_THSE_H_val          0x02
```

The original values are:

```
1  #define DD_THSI_L_val          0x00
2
3  /*! \def DD_THSI_H_val
4      \brief Direction Change Internal Threshold MSB
5
6      Direction Change Internal Threshold MSB.
7   */
8  #define DD_THSI_H_val          0x30
9
10 /*! \def DD_THSE_L_val
11     \brief Direction Change External Threshold LSB
12
13     Direction Change External Threshold LSB.
14  */
15 #define DD_THSE_L_val          0x00
16
17 /*! \def DD_THSE_H_val
18     \brief Direction Change External Threshold MSB
19
20     Direction Change External Threshold MSB.
21  */
22 #define DD_THSE_H_val          0x3C
```

A.3.2 Maximum Data Length

We changed the length of the maximum data that can be transferred by the Waspmotes. The original value is 100 bytes, we change it to 300 bytes. This is done by changing the value in line 56 of "WaspXBeeConstants.h" in Waspmote API version 0.18.

```
1 #define MAX_DATA      300
```

A.4 Waspmote Logic

We simulate two different type of WPAN nodes in our prototype: a motion sensor and an LED actuator. A motion sensor detects its own movement. When it is moved or shook, it sends a message to a destination node (hard-coded at the moment). An LED actuator listens to incoming ZigBee messages and once it receives a specific command (hard-coded and will be configurable in following work), it blinks its LED lights on board.

TRITA-ICT-EX-2011:146

www.kth.se