

Aalto University  
School of Science  
Degree Programme of Computer Science and Engineering

Nalin Gupta

# Management of Decentralized DHT Based M2M Network

Master's Thesis  
Espoo, November 30, 2011

**DRAFT! — December 1, 2011 — DRAFT!**

Supervisor: Professor Antti Ylä-Jääski  
Aalto University School of Science, Finland  
Instructor: Jouni Mäenpää M.Sc. (Tech.)  
Ericsson Research, Finland

Aalto University  
 School of Science  
 Degree Programme of Computer Science and Engineering

ABSTRACT OF  
 MASTER'S THESIS

<b>Author:</b>	Nalin Gupta		
<b>Title:</b>	Management of Decentralized DHT Based M2M Network		
<b>Date:</b>	November 30, 2011	<b>Pages:</b>	9+ 36
<b>Professorship:</b>	Data Communication Software	<b>Code:</b>	T-110
<b>Supervisor:</b>	Professor Antti Ylä-Jääski Aalto University School of Science, Finland		
<b>Instructor:</b>	Jouni Mäenpää M.Sc. (Tech.) Ericsson Research, Finland		
<p>Today Machine-to-Machine (M2M) technology is evolving beyond the traditional telemetry use-cases and becoming key enabler for innovative concepts like Internet-of-Things (IoT), leading to a whole new set of possible applications. Currently there are 13 billion connected devices and it is envisioned that by the year 2020, this number will reach 50 billion [6]. We believe that the key to achieve this massive scalability is using peep-to-peer based decentralized architecture and interoperability using open standards. Architecturally, the Internet is composed of a collection of heterogeneous sub-networks; similarly, we believe that the next generation of M2M networks would also consist of heterogeneous networks joined together using gateways or proxys, giving a hierarchical architecture.</p> <p>Hence, we are conducting research into decentralized hierachical Machine-to-Machine (M2M) networks using IETF and IEEE aligned protocols. The decentralization is achieved using Distributed Hash Table (DHT) based Peer-to-Peer (P2P) algorithms. The project involves studying the latest state-of-the-art technologies like 6LoWPAN, ZigBee, CoAP, DHT, Smart-M3, and creating a prototype testbed platform of embedded Linux and ZigBee devices to carry the experiments and analysis. The focus of this thesis is on how to manage this decentralized hierarchical M2M network using SNMP. The central topics studied are decentralized management, node discovery, resource discovery, node configuration, and gateway functionality.</p>			
<b>Keywords:</b>	M2M, Decentralized, Network Management, SNMP, Proxy, DHT, IoT, Smart-M3, CoAP		
<b>Language:</b>	English		

# Acknowledgements

I would like to express my sincere gratitude to Professor Antti Ylä-Jääski for supervising my thesis and supporting my studies, without which this moment would not have been possible.

I am deeply grateful to my instructors Jouni Mäenpää and Jani Hautakorpi for providing me with the opportunity to work in the prestigious Ericsson Research NomadicLab, and for their guidance, encouragement and kindness throughout the research work.

I would also like to thank my fellow students Jaime Jimenez and Daoyuan Li for their friendship and support during the thesis.

Espoo, November 30, 2011

Nalin Gupta

# Abbreviations and Acronyms

3G	3rd Generation mobile telecommunications
6LoWPAN	IPv6 over Low-power Wireless Personal Area Networks
ACK	Acknowledgment
API	Application Programming Interface
APS	Application support sub-layer
CoAP	Constrained Application Protocol
CoRE	Constrained RESTful Environments working group
DHT	Distributed Hash Table
DDNS	Distributed Domain Name Service
DNS	Domain Name Service
GPS	Global Positioning System
ICT	Information and Communication Technologies
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
IoT	Internet of Things
IPC	Inter-Process Communication
LR-WPAN	Low-Rate Wireless Personal Network
LoWPAN	Low-power Wireless Personal Network
LN	Local Node, less powerful devices without DHT P2P e.g. ZigBee motes
M2M	Machine-to-Machine
M2MCE	Machine-to-Machine Communication Enabler
MCN	Monitoring and Controlling Node
MD5	Message-Digest algorithm 5
MIB	Managed Information Base
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
P2PM2M	Peer-to-Peer based decentralized M2M network
PAN	Personal Area Network

PDU	Protocol Data Unit
PN	Proxy Node for interfacing LNs with WN
QoS	Quality of Service
REST	Representational State Transfer
RFID	Radio-Frequency IDentification
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RTT	Round Trip Time
SIB	Semantic Information Broker
SNMP	Simple Network Management Protocol
UDP	User Datagram Protocol
URI	Universal Resource Indicator
USB	Universal Serial Bus
USM	User-based Security Model
VACM	View-based Access Control Model
WAN	Wide Area Network
WN	Wide Area Node, embedded linux based devices with DHT P2P
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
WWAN	Wireless Wide Area Network
ZDO	ZigBee Device Object

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>Abbreviations &amp; Acronyms</b>	<b>iv</b>
<b>Abbreviations and Acronyms</b>	<b>iv</b>
<b>List of Tables</b>	<b>viii</b>
<b>List of Figures</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Overview . . . . .	1
1.2 Research Objective and Thesis Scope . . . . .	2
1.3 Thesis Organization . . . . .	4
<b>2 Decentralized M2M Network Design</b>	<b>5</b>
2.1 Motivation . . . . .	5
2.2 Design Principles . . . . .	6
2.3 System Architecture . . . . .	8
2.4 Functional Description . . . . .	9
2.4.1 M2M Communication Enabler Abstraction Layer . . . .	10
2.4.2 Monitoring and Controlling Node . . . . .	12
2.4.3 Proxy Node . . . . .	12
2.5 Smart-M3 System . . . . .	13
<b>3 Decentralized M2M Network Management</b>	<b>15</b>
3.1 MCN Functionality . . . . .	15
3.2 MCN Interfaces with M2MCE and PN . . . . .	17
3.3 Security . . . . .	18
3.4 Scenarios . . . . .	18

3.4.1	Name Resolution . . . . .	18
3.4.2	MCN operation . . . . .	20
3.4.3	LN to WN . . . . .	21
3.4.4	WN to LN . . . . .	22
<b>4</b>	<b>Prototype Implementation</b>	<b>24</b>
4.1	Hardware specification . . . . .	24
4.1.1	LN hardware . . . . .	24
4.1.2	WN hardware . . . . .	26
4.1.3	PN hardware . . . . .	26
4.1.4	MCN hardware . . . . .	27
4.2	Software specification . . . . .	27
4.3	Software Implementation . . . . .	30
4.3.1	M2M System Startup . . . . .	31
<b>5</b>	<b>Conclusions and Future Work</b>	<b>33</b>
5.1	Future work . . . . .	33
	<b>Bibliography</b>	<b>35</b>

# List of Tables

2.1	Device Properties . . . . .	10
-----	-----------------------------	----



# List of Figures

1.1	Decentralization of M2M Network . . . . .	2
2.1	Decentralized Hierarchical M2M Network Architecture (P2PM2M)	8
2.2	M2MCE Abstraction Layer . . . . .	11
2.3	Smart-M3 System . . . . .	14
3.1	M2M Protocol Stack . . . . .	16
3.2	Message Sequence for Name resolution . . . . .	19
3.3	MCN sets association between sensor and actuator . . . . .	20
3.4	Information from LN towards WN . . . . .	21
3.5	Information from WN towards LN . . . . .	23
4.1	Local Node . . . . .	25
4.2	Wide Area Node . . . . .	27
4.3	Proxy Node . . . . .	28
4.4	Software Architecture of PN . . . . .	30

# Chapter 1

## Introduction

### 1.1 Overview

Machine-to-Machine (M2M) technology gained roots in early 2000s, powered by the ubiquitous wireless cellular connectivity. Today it is evolving beyond the telemetry applications and becoming cardinal enabler for innovative concepts like Internet-of-Things (IoT), leading to a whole new set of possible applications. Currently there are 13 billion connected devices and it is envisioned that by the year 2020, this number will reach 50 billion [6]. We believe that the key to achieve this level of massive scalability is using Peer-to-Peer (P2P) based decentralized architecture and interoperability using open standards. Architecturally, the Internet is composed of heterogeneous sub-networks, similarly, we believe that the next generation of M2M networks would also consist of collection of heterogeneous subnets joined together using gateways or proxies, giving a hierarchical structure. Hence, we are conducting research into scalable decentralized hierarchical Machine-to-Machine (M2M) networks using IETF and IEEE aligned open standards and protocols. The decentralization is achieved using Distributed Hash Table (DHT) based P2P algorithms. The project involves studying the latest state-of-the-art technologies like 6LoWPAN, ZigBee, CoAP, Smart-M3, and creating a prototype testbed platform of embedded Linux and ZigBee devices to carry the experiments and analysis.

This research project is part of work package for Devices and Interoperability Ecosystem (DIEM) <sup>1</sup> program funded by TEKES <sup>2</sup> - the Finnish Funding Agency for Technology and Innovation. The overall research project also involves close collaboration with the Future Internet program, which is

---

<sup>1</sup><http://www.diem.fi>

<sup>2</sup><http://http://www.tekes.fi>

part of ICT cluster of the Finnish Strategic Centres for Science, Technology and Innovation (ICT SHOK) <sup>3</sup>. Inline with the objectives of these research programs, interoperability and future Internet are key aspects of our research project. This explains the emphasis of this project on CoAP, IP, SNMP and IEEE 802.15.4 technologies.

## 1.2 Research Objective and Thesis Scope

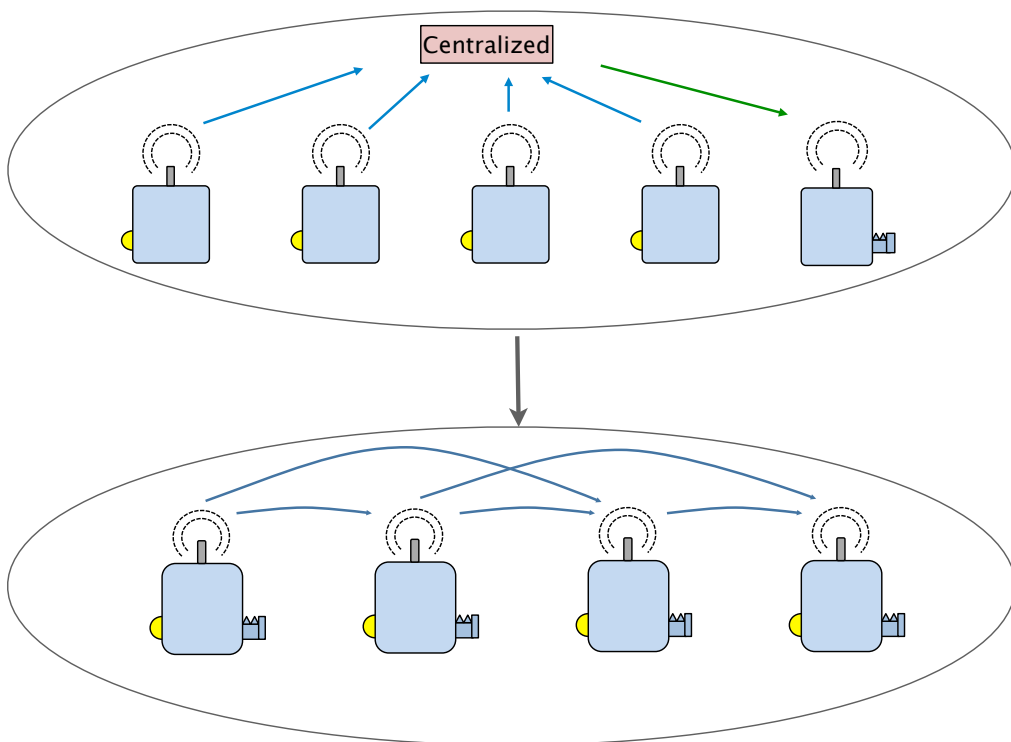


Figure 1.1: Decentralization of M2M Network

The current M2M and WSN networks use centralized approaches. The WSNs use sensors to collect information about its environment and relay it to central location, where the data is processed. M2M networks use embedded devices for remote monitoring and management from central locations. In both the cases, the intelligence and decision making is done using centralized entities.

<sup>3</sup><http://www.futureinternet.fi>

However, the centralized approaches have several challenges which do not lend themselves to the envisioned use-cases for M2M and IoT. Three key issues are:

- scalability to the order of millions of nodes
- distribution of intelligence among nodes
- tolerate failure of parts-of-networks (centralized entities have single point of failure for entire network)

Our aim is to solve the above challenges by decentralizing the M2M networks using DHT-based P2P mechanisms - creating a P2PM2M network (Chapter 2 Section 2.1 describes how P2P addresses these issues). The objective of our research project is to study the challenges in the decentralization of M2M networks. The research project is divided into three focus areas - adapting DHT for the requirements for P2PM2M network, using proxy to connect ZigBee network to the P2PM2M network, and managing the P2PM2M network. The scope of this thesis is the last focus area i.e. how to manage the decentralized M2M network using open standards. The main topics that are studied are decentralized management, node discovery, resource discovery, node configuration, and gateway functionality.

The following specific objectives defines the scope of this thesis:

1. Develop network management software and Managing and Controlling Node (MCN) that provides the required functions for human interface
2. Self-reliant M2M network: the M2M network should be able to operate normally even when MCN is disconnected
3. The software should have gateway or proxy module that allows management of ZigBee based sub-network.
4. The software should allow information interoperability using Smart-M3 platform, enabling other Smart-M3 software to interact with the decentralized M2M system

In tune with the IETF approach of "have running code" <sup>4</sup>, developing the prototype testbed is important part of this project. This testbed platform will then be used for further research on the overall topic of decentralizing M2M networks.

---

<sup>4</sup><http://www.ietf.org/tao.html>

### 1.3 Thesis Organization

Chapter 2 describes the design principles used for prototype, its system architecture and functionality. Next, Chapter 3 covers the details on management of the M2M network and the mechanisms used for decentralized management.

The hardware and software specifications for the prototype implementation are given in Chapter 4. It also covers the software architecture of the prototype and the system startup details. In Chapter 5, the learnings from this project are summed up and useful take-away conclusions are presented. Finally, we mention the research topics that will be carried out as future work, using the testbed developed under the scope of this project.

## Chapter 2

# Decentralized M2M Network Design

In this chapter we describe the motivation for decentralizing M2M networks using P2P, followed by the principles used for designing the M2M network architecture, and then an overview of the architecture, and finally a description of the three most important functional entities in the network.

### 2.1 Motivation

In Section 1.2 we mention that the centralized systems suffer from scalability issues and difficulty in distributing intelligence in network. In this section we elaborate on the need to distribute intelligence in M2M networks, and how P2P effectively solves the scalability issue of M2M networks.

The current technology trend is that the intelligence is moving towards the edge of network, for example in the cellular systems, Base Transceiver Stations (BTS) have evolved into intelligent NodeB. The intelligent devices can interact with other devices to share information, and then use the local information to make decision by itself without requiring any central entities to issue commands. We believe that intelligent devices with Artificial Intelligence (AI) is key for success of concepts like IoT. Consider one of the proposed use-cases for IoT <sup>1</sup>: there is an accident on road, which causes the alarm clock to start ringing 30 minutes earlier to enable the person to reach office on time. Another use-case is handling traffic light signaling and speed limits according to traffic patterns and weather conditions. However, most software can only handle the scenarios for which it was explicitly designed,

---

<sup>1</sup><http://ws4d.e-technik.uni-rostock.de/pipesbox>

and it takes AI to handle new unforeseen scenarios. Since IoT use-cases involve using complex set of information to handle unforeseen scenarios, AI is critical for IoT success. Even though gaming industry have been using AI since a long time, it is Apple "Siri" application [3] which has brought AI to mainstream masses for the first time since more than four decade of Computer Science. P2P technologies enables distribution of intelligence in the network, hence it is well suited for integrating AI with M2M networks. This is an exciting research area, however, AI and its integration with M2M networks is out of scope for this thesis, and we focus on using P2P for M2M communication.

Additionally, peer-to-peer technologies can be used effectively to achieve Internet level of scalability. For example, consider a centralized network consisting of 100000 nodes. Now if another 1000 nodes have to join the network, it would require additional resources from the centralized entities putting additional load, and the system can easily be saturated. In contrast, in a peer-to-peer based system, each new node joining the network pools its own resources towards the network system thereby allowing higher scalability. BitTorrent is a very good example to illustrate the concept. In a BitTorrent system, a machine originally hosting a file needs to transfer the segments of the file only once, and all the other peer machines can get a copy of the file by exchanging the segments among themselves. This way each machine which is a part of the peer-to-peer system shares its resources - computational, storage and network bandwidth - with the whole system allowing even distribution of load, and hence allowing system to scale easily.

## 2.2 Design Principles

In this section we outline the design principles along with a brief motivation for making these design choices.

IEEE 802.15.4 <sup>2</sup> has emerged as the undisputed leader in low power radio technology [14]. It is used in most of the M2M specifications like 6LoWPAN [12], ZigBee, ISA100.11a [1], and WirelessHART [2]. The network formed by IEEE 802.15.4 devices is typically referred to as Wireless Personal Area Network (WPAN). We considered among ZigBee and 6LoWPAN devices for our prototype testbed. The 6LoWPAN devices have native IP support, hence they are a good candidate, however, 6LoWPAN devices were not easily available at the time of writing this thesis and they were more expensive. On the other hand, ZigBee devices are easily available, and moreover

---

<sup>2</sup><http://www.ieee802.org/15/pub/TG4.html>

ZigBee consortium is already aligning themselves to IP <sup>3</sup>, hence we selected ZigBee devices for WPAN.

Internet is a huge success story and two factors have been critical to its mass adoption - interoperability based on open standards and connecting vastly heterogeneous networks by allowing flexibility on the physical layer technologies. We believe that these two factors will also be very important for the success of next generation M2M networks. In specific, IP capability will be crucial since it will enable the embedded devices to easily integrate with the existing Internet infrastructure. To study connecting heterogeneous network, connecting IP enabled M2M network with ZigBee network is one of the important aspects of this research.

The objective of this project is to decentralize M2M networks using P2P technologies. We have chosen Chord [15], which is a DHT based P2P protocol, to form the decentralized distributed system. It is known that currently it is infeasible to execute DHT implementations on low resource embedded devices, hence, the ZigBee nodes do not use P2P technologies. We have chosen Linux based microcontroller boards for hosting Chord protocol.

Another interesting aspect is the range of communication. We are interested in both, local communication between devices (for example, home appliances talking with smart energy metering) and long distance communication (for example, a user checking the status of home appliances from office). The IoT Initiative (IoT-i) program <sup>4</sup> has defined 60 potential use-cases for IoT and in several cases, the devices are spread over large physical distance. Low powered radio technology like ZigBee enable local communication, but for long distance we need wireless cellular technologies. Hence, we have decided to use 3G connectivity for the devices that are connected using P2P. Cellular based M2M devices have the benefit of easier installation and enabling higher mobility and bandwidth <sup>5</sup>.

Additionally, cellular based embedded devices can also nicely work as Gateway devices to connect the WPAN formed by low powered ZigBee devices. The resulting hierarchical structure adds to the scalability of the system and allows the possibilities of interoperating with heterogeneous technologies through Gateway (since the WPAN could be 6LoWPAN or some other technology).

A monitoring and controlling entity is required in the M2M network to provide interface to humans. Since we want the network to be self-reliant, the network must be able to work without requiring continuous presence of this

---

<sup>3</sup><https://docs.zigbee.org/zigbee-docs/dcn/09-5003.pdf>

<sup>4</sup><http://www.iot-i.eu>

<sup>5</sup><http://www.etsi.org/Website/Technologies/M2M.aspx>



entity. Also, keeping in-line with the decentralization goal, the management function should be spread over the devices in the network. Hence, all the M2M devices will host management software.

Summing up the above design principles leads to the system architecture design as shown in Figure 2.1 in next section.

## 2.3 System Architecture

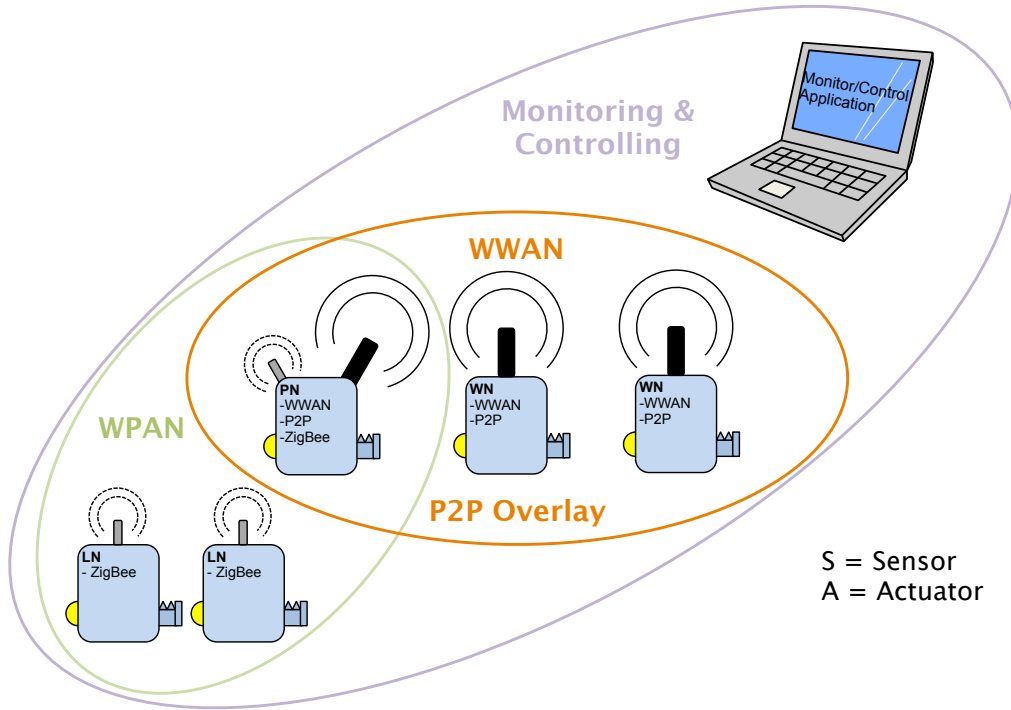


Figure 2.1: Decentralized Hierarchical M2M Network Architecture (P2PM2M)

The system architecture for our prototype is shown in Figure 2.1. Henceforth, we will use the term P2PM2M to refer to this decentralized M2M network. There are 4 distinct types of nodes in P2PM2M and they have been termed individually for easy reference. All the 4 nodes differ in their hardware and functionality.

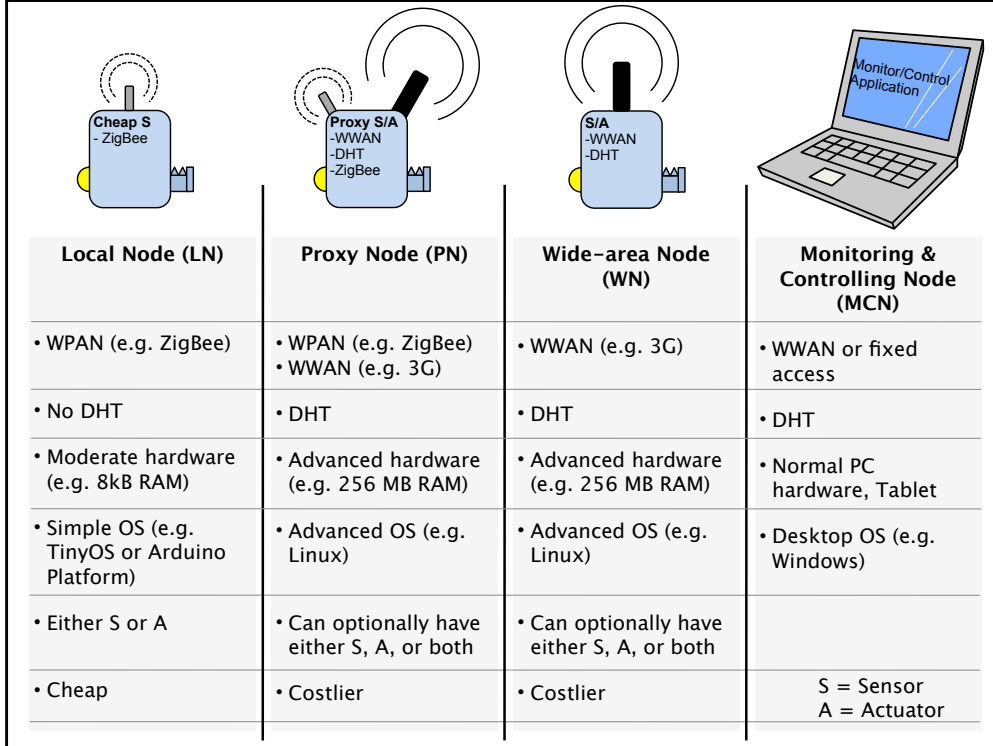
- **Monitoring and Controlling Node (MCN)** : This device provides human interface to the P2PM2M network and it is used to configure the nodes and P2P system. It will not have any sensors or actuators. It can be a laptop, tablet or smart-phone possibly running a GUI. In our prototype, we have used a laptop. As explained later, it uses SNMP protocol for network management.
- **Wide Area Node (WN)** : This device forms part of the P2P overlay using the DHT based implementation. It uses 3G wireless cellular connectivity and since these devices can communicate over large distances, the collection of these devices is called Wireless Wide Area Network (WWAN). It can have both sensors and actuators, and it uses CoAP for M2M communication.
- **Local Node (LN)** : This device is cheaper, but at the same time a highly constrained device with limited processing, storage and short range radio connectivity. It cannot host DHT based P2P implementation, but thanks to the Gateway or Proxy Node functionality, it still participates in the P2P overlay. It can have both sensors and actuators. In our prototype, ZigBee devices are used as LNs.
- **Proxy Node (PN)** : This device is same as WN with the addition that it also performs the Gateway and Proxy functionality for the LNs. It has an attached ZigBee Coordinator device which enables it to handle WPAN management and provide messaging interface to the LNs. It is also responsible for the protocol conversion for CoAP and SNMP messages into ZigBee protocol (and vice-versa).

Since WN can have sensors and actuators, they only differ from ZigBee devices in terms of their ability to host DHT implementation and IP connectivity. So WN represents the next generation M2M embedded devices, and when the advancements in P2P algorithms and hardware capabilities will bridge the gap. Hence without loss of generality or flexibility, this system architecture represents a decentralized M2M networks.

The table 2.1 captures the important properties of all the four types of M2M devices.

## 2.4 Functional Description

There are three functional entities that have been researched and developed in the overall scope of this project to decentralize M2M networks.



Local Node (LN)	Proxy Node (PN)	Wide-area Node (WN)	Monitoring & Controlling Node (MCN)
<ul style="list-style-type: none"> <li>• WPAN (e.g. ZigBee)</li> </ul>	<ul style="list-style-type: none"> <li>• WPAN (e.g. ZigBee)</li> <li>• WWAN (e.g. 3G)</li> </ul>	<ul style="list-style-type: none"> <li>• WWAN (e.g. 3G)</li> </ul>	<ul style="list-style-type: none"> <li>• WWAN or fixed access</li> </ul>
<ul style="list-style-type: none"> <li>• No DHT</li> </ul>	<ul style="list-style-type: none"> <li>• DHT</li> </ul>	<ul style="list-style-type: none"> <li>• DHT</li> </ul>	<ul style="list-style-type: none"> <li>• DHT</li> </ul>
<ul style="list-style-type: none"> <li>• Moderate hardware (e.g. 8kB RAM)</li> </ul>	<ul style="list-style-type: none"> <li>• Advanced hardware (e.g. 256 MB RAM)</li> </ul>	<ul style="list-style-type: none"> <li>• Advanced hardware (e.g. 256 MB RAM)</li> </ul>	<ul style="list-style-type: none"> <li>• Normal PC hardware, Tablet</li> </ul>
<ul style="list-style-type: none"> <li>• Simple OS (e.g. TinyOS or Arduino Platform)</li> </ul>	<ul style="list-style-type: none"> <li>• Advanced OS (e.g. Linux)</li> </ul>	<ul style="list-style-type: none"> <li>• Advanced OS (e.g. Linux)</li> </ul>	<ul style="list-style-type: none"> <li>• Desktop OS (e.g. Windows)</li> </ul>
<ul style="list-style-type: none"> <li>• Either S or A</li> </ul>	<ul style="list-style-type: none"> <li>• Can optionally have either S, A, or both</li> </ul>	<ul style="list-style-type: none"> <li>• Can optionally have either S, A, or both</li> </ul>	
<ul style="list-style-type: none"> <li>• Cheap</li> </ul>	<ul style="list-style-type: none"> <li>• Costlier</li> </ul>	<ul style="list-style-type: none"> <li>• Costlier</li> </ul>	S = Sensor A = Actuator

Table 2.1: Device Properties

### 2.4.1 M2M Communication Enabler Abstraction Layer

We have used OSI layering principles to separate the system functionality into hierarchical layers, where each layer provides well defined APIs for higher layers. Now, DHT algorithm by themselves only provides the ability to store and retrieve a key-value pair. To develop a decentralized M2M system using DHT, we needed to create an abstraction layer - which we termed as M2M Communication Enabler (M2MCE) - on top of DHT, which hides the distributed nature of information storage and provides an API to access information as if the information was stored on the local node itself (hence the APIs are implemented as synchronous function calls and not as asynchronous message passing or callback functions). In addition, M2MCE also provides APIs to enable messaging using the P2P overlay, and accessing the logical predecessor/successor nodes of P2P (Chord based) overlay.

The M2MCE layer provides the following functions:-

- M2MCE provides APIs which allow nodes to join and leave the P2P

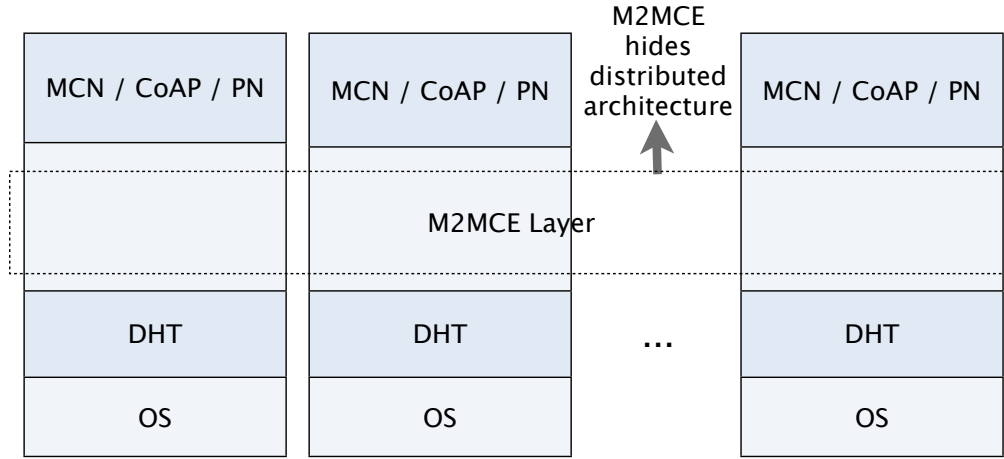


Figure 2.2: M2MCE Abstraction Layer

overlay. These APIs are used by WN and PN to register/unregister to the M2M system. LNs do not directly use these APIs, but PN does it on their behalf.

- M2MCE provides interface for distributed data storage, which internally builds upon the underlying DHT algorithm. Typically, the data is stored over several peers and it is ensured that there is no loss of data even if some peers leave the system. This distributed storage can also be used to store network alarms, traps etc.
- Since all the nodes register to P2PM2M using M2MCE, it can easily maintain a log of all the nodes in the network using the distributed storage. This gives an easy node discovery mechanism.
- M2MCE provides distributed DNS functionality. As such, the M2M system is about machines communicating with machines and hence do not need a human readable name. However, humans need to interact

with the system for provisioning and maintenance, so having human understandable names for devices is required. Also, these names are used in CoAP URIs. Additionally, the notion of *name* helps decouple identity and location for mobile nodes avoiding the well known issue with IPv4.

- M2MCE allows broadcast messaging service in the P2PM2M network, using the Chord overlay. There can be different techniques used to broadcast message, but these methods are transparent for the higher layer (above M2MCE). It is a research agenda for M2MCE to find the most efficient method for broadcasting.

Detailed description of M2MCE is beyond the scope of this thesis, but complete details can be found in [4].

### 2.4.2 Monitoring and Controlling Node

Since MCN is part of the focus area of this thesis, it is covered in-depth in the chapter 3.

### 2.4.3 Proxy Node

The proxy node performs the following functions:-

- The WWAN uses SNMP and CoAP protocols, whereas the WPAN uses ZigBee protocols. Hence, PN does the protocol conversion from SNMP and CoAP to ZigBee. A few messages have direct mapping between the protocols (like the SNMP MIB to trigger detaching a LN is directly mapped to ZigBee ZDO message ZDO\_CLUSTER\_MGMT\_LEAVE\_REQUEST), and the other messages are sent to LNs using Type-Length-Value (TLV) format. PN also interfaces with the CoAP proxy and SNMP proxy as per their respective specifications
- PN is responsible for the ZigBee WPAN management, including node discovery and service discovery. After the PN starts, it enables the ZigBee coordinator node (which is part of PN). Whenever a LN joins (or leaves) the WPAN, it inform the ZigBee coordinator node, which further informs PN. In this way, PN keeps track of all the LNs that are part of the WPAN. ZigBee coordinator node uses a timer based heartbeat mechanism to discover any LN that leave WPAN without informing, for example due to failure. PN uses ZigBee protocol to discover the services hosted by LNs.

- When the LNs join the WPAN, the PN assigns them a name using the well known *dot* notation, for example, if the PN name is PN100, then it can assign LN names like PN100.LN1, PN100.LN2, PN100.LN. Then PN uses M2MCE interface to join the P2P overlay on behalf of each LN. Hence, PN represents LNs in P2P overlay, provides an abstraction to the other peer nodes such that the LNs appear as a full-fledged peer. In a way, we can think of LNs as pseudo-peers. Similarly, when a LN leaves WPAN, PN performs leave operation on DHT on behalf of the departing LN.
- LNs are frontend by PN, and M2MCE layer translates a LN name (like PN100.LN1) into the IP address of the PN (since LN can have only ZigBee address). So for any messages exchanged between WWAN and WPAN, PN does the translation between IP address and ZigBee device address.
- PN caches sensor data. This is especially useful for the sleeping sensor devices, which periodically wake up and send data to the PN. PN then caches this data and uses it to handle any future request. PN can also use P2P overlay to cache the data if required.
- PN can potentially provide firewall functionality to secure WPAN, however this is left as future work.

Detailed description of PN is beyond the scope of this thesis, but complete details can be found in [11].

## 2.5 Smart-M3 System

Smart-M3 system is an information sharing platform for interoperability between multi-vendor, multi-device and multi-part. We integrated our decentralized M2M network with Smart-M3 system to study the issues in using P2P and proxy with Smart-M3.

The Figure 2.3 shows the Smart-M3 system which is used for sharing the information in M2M network, for example the sensor values. All the network nodes have a Smart-M3 *Knowledge Process* (KP) [8] which is responsible for interactions with the Smart-M3 Platform. The LN are represented in the Smart-M3 system by PN, and the KP in PN is responsible for interoperating between Smart-M3's Smart Space Access protocol (SSAP) and ZigBee protocol. Smart-M3 semantic information broker (SIB) is the geographical smart space containing information about all the nodes and resources available [8].

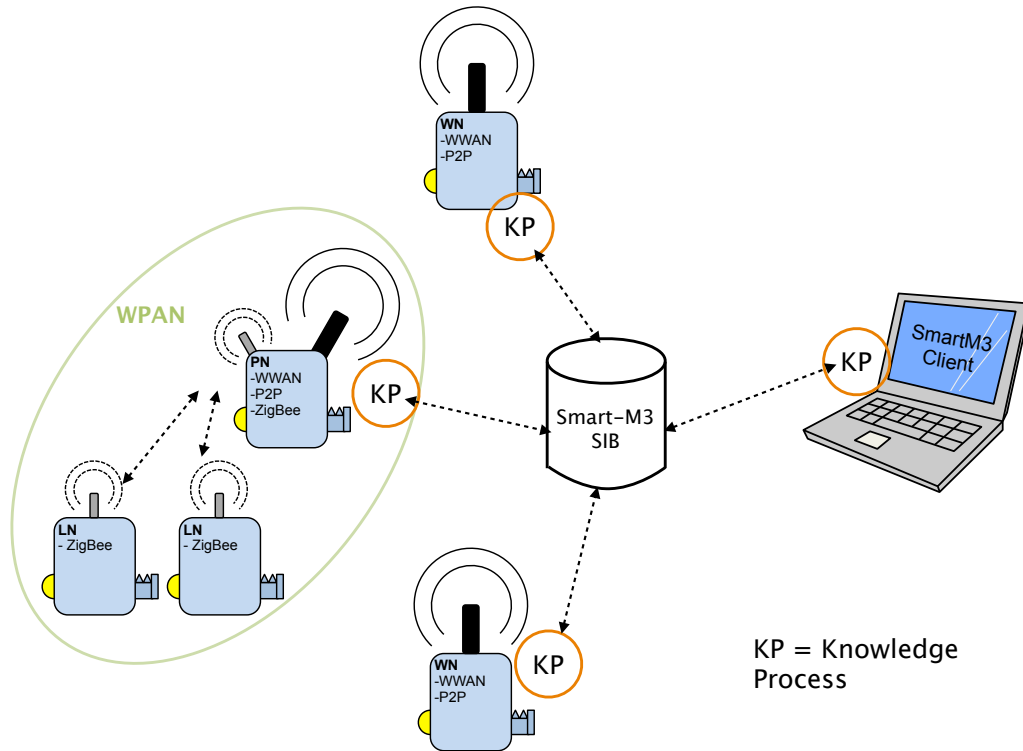


Figure 2.3: Smart-M3 System

The KPs in all M2M devices (WNs and LNs via PN) populate the Smart-M3 SIB with the information about all the nodes and their resources. Then any device with KP can act as Smart-M3 client and interact via Smart-M3 SIB, to display the list of resources and retrieve any resource's value. For example, all ZigBee nodes have temperature sensors and GPS modules, and a Smart-M3 client can interact with both the resources and get the temperature at any particular location. It is worthwhile to note that here no CoAP protocol is used.

## Chapter 3

# Decentralized M2M Network Management

The main tasks of network management can be categorized as follows:

- Provisioning the system, which involves configuring the different resources
- Monitoring the network, services and managing faults
- Administration, consisting of housekeeping activities like node and resource discovery

Accordingly, our research can be categorized into 4 focus areas:

- Decentralization of network management
- Creating self-reliant network
- Identifying the SNMP MIB for P2PM2M
- Proxy functionality for SNMP

### 3.1 MCN Functionality

The MCN performs the following functions:-

- MCN is used for node discovery. SNMP does not provide any mechanism for node discovery, so each system has its own MCN uses M2MCE API to get the list of all the nodes that are part of the P2P overlay. This provides a convenient mechanism for node discovery.



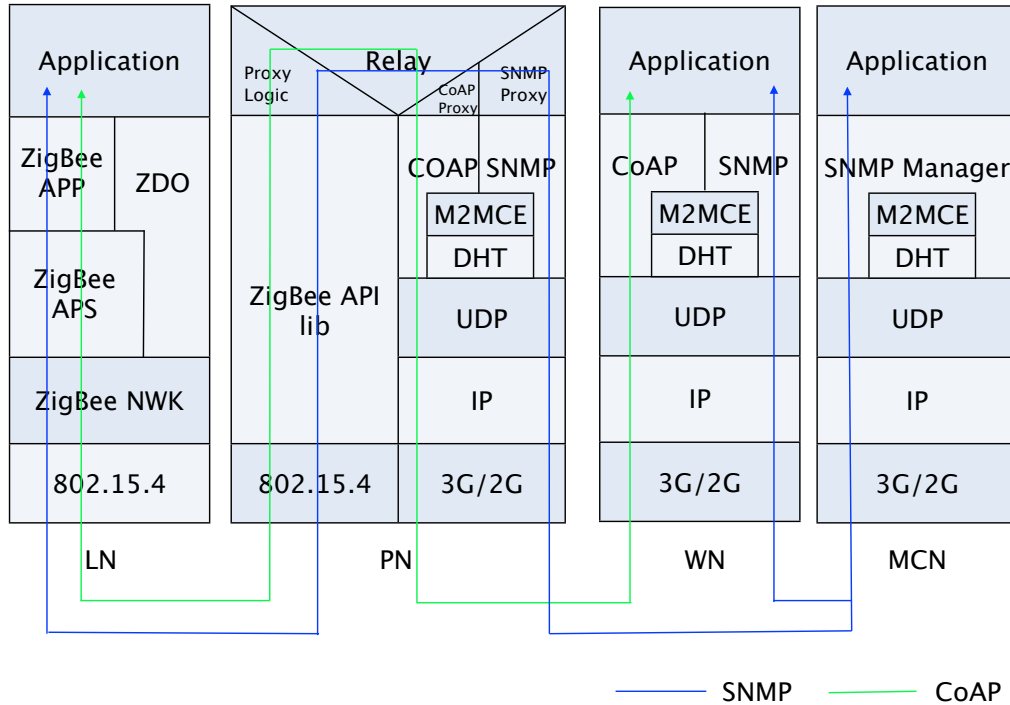


Figure 3.1: M2M Protocol Stack

- MCN is used to manage the WWAN like detaching nodes from P2P overlay, assigning the type of node WN/PN, setting the node name, setting the association between sensor and association. An important point here is that only the sensor-actuator association is controlled by MCN using SNMP, the flow of data between them is using CoAP and autonomously done by the nodes.
- MCN is used for discovery and provisioning of resources on the nodes. The resources can be dynamically added or removed. This database of resources is stored in MIB and it is also used by CoAP to advertise the resources to other nodes using the .well.known URL [ref to coap spec]. The resources in MIB are considered master database, which keeps the resource list is synchronized across CoAP and SNMP protocols.
- MCN manages the WPAN using the proxy features on PN, like detaching the LNs from the WPAN.
- The M2M system does not require MCN to be present in the system

for its functioning and the system is *self reliant*. It means that the monitoring of the system is distributed over the nodes itself and they use DHT to store information which is later retrieved by MCN.

- MCN is used for setting system parameters on nodes like enabling or disabling the Smart-M3 feature, controlling the logging level etc. MCN uses SNMP to manage the nodes, so it can use the standard MIBs to configure system parameters. Since the nodes use IP, IP-MIB and IF-MIB MIBs are particularly useful.
- MCN can have a GUI interface for laptops and smart-phones, which allows full functionality of MCN accessible like node discovery, resource discovery, visual indications for alarms, and a library of graphs and charts for displaying data from nodes. The idea is to have a generic interface for M2M interface which should be portable for other M2M and IoT networks. A good example of work in this direction is Pachube <sup>1</sup>, which allows storing of data feeds, controlling it and displaying data. The challenge here is to have a generic interface, massive scalability, and enable real time operation.
- MCN is responsible for monitoring the nodes in the network for problems, performance issues etc. Since the role of monitoring is decentralized and distributed among the network nodes, MCN uses the APIs provided by M2MCE to retrieve the monitoring status and alarms. Event driven: polling approach would have implied that MCN will periodically check with M2MCE for any alarms, which is wasteful. Hence, an event driven approach has been chosen where M2MCE checks for the presence of MCN and informs it for any network alarms.

## 3.2 MCN Interfaces with M2MCE and PN

Interfaces with M2MCE:

- get IP From Node Name (DDNS functionality)
- send multicast message
- get node information like P2P joining time
- get traps information

---

<sup>1</sup><https://pachube.com>

- get the predecessor and successor nodes (for monitoring purposes)
- get event information when a predecessor and successor nodes leave DHT, so that the monitoring node can start monitoring new neighboring peer

Interfaces with PN:

- send and receive messages in WPAN
- protocol conversion for SNMP messages into ZigBee messages
- WPAN management
- protocol conversion for CoAP messages into ZigBee messages
- protocol conversion for Smart-M3 messages into ZigBee messages

### 3.3 Security

PN is responsible for the security of WPAN. ZigBee provides mechanisms for security of the application data and the network. The PN uses ZigBee coordinator node (which is part of PN itself) as the trust center for managing and distributing keys. LNs use the shared keys for cryptographic algorithms.

M2MCE is responsible for security of P2P overlay and stored data. MCN is responsible for SNMP security and securing access. SNMPv3 User-based Security Model (USM) can be used for authenticating users and SNMPv3 View-based Access Control Model (VACM) can be used for user-id based access restriction to WPAN.

### 3.4 Scenarios

A few scenarios are explained here which describe the functioning of the prototype testbed. We show the message sequence charts, followed by its explanation.

#### 3.4.1 Name Resolution

As mentioned in section 2.4.1, M2MCE provides a distributed name resolution service to other entities like MCN and WN. The M2MCE API for this is a synchronous function call which takes node name as input and returns corresponding IP address.

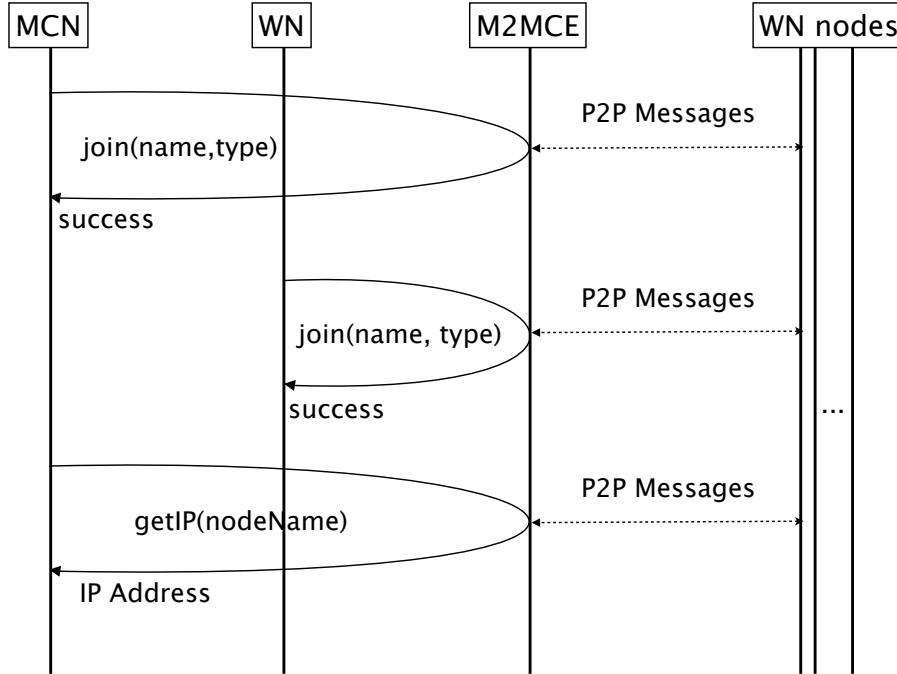


Figure 3.2: Message Sequence for Name resolution

These bullet points are in reference to the message sequence chart in figure.

1. First the nodes, like MCN and WN, invoke M2MCE `join()` API to register themselves and become part of the P2P overlay network. M2MCE layer, in turn, uses P2P operations to distribute the information over the P2P overlay network. These P2P operations and distributed storage is hidden from higher layers.
2. Later when required, any node (MCN here) invokes the M2MCE `getIP()` API to get the the IP address for a given node name. M2MCE layer again uses P2P operations to retrieve the information from distributed storage.
3. Note that when `getIP()` is invoked to resolve IP address for a LN, the IP address returned is in-fact that of the PN representing the LN. However, the routing of message to LN is transparent to the sending

node as the proxy feature on PN is responsible for routing the message to the correct target LN. This proxy feature uses the *uri-host* part of the CoAP [7] and Proxy Forwarder [10] feature of the SNMP.

Similar interaction happens for providing node discovery and other functionality which store/retrieve data into/from the distributed DHT-based database, with M2MCE providing a higher level abstraction API which hides all the complexities of distributed system. This message sequence is basic and part of all the next scenarios.

### 3.4.2 MCN operation

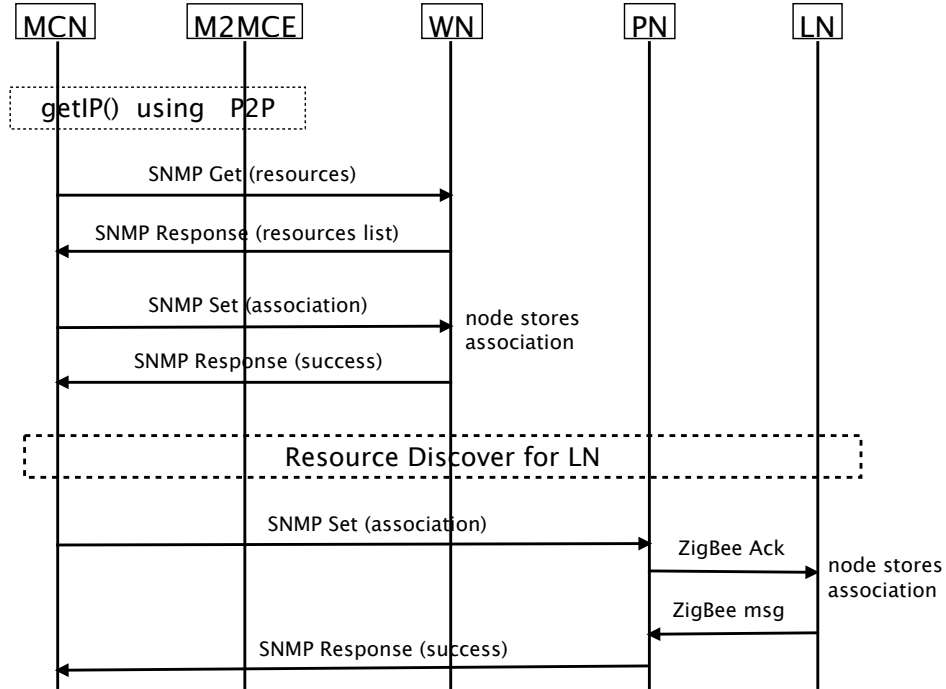


Figure 3.3: MCN sets association between sensor and actuator

One of the useful functionality of MCN is setting the association between sensors and actuators of any arbitrary nodes. The node containing sensor uses this association to inform the node containing actuator when a pre-determined event occurs. For example, a sensor monitoring fire can inform actuator controlling sprinklers when it detects a fire outbreak.

1. Typically the first thing to do before sending message to any node, is to resolve the node name into its IP address.
2. Next, MCN uses SNMP protocol to discover the resources (sensors and actuators) on a WN node. Then using this list of resources, MCN can associate a sensor with an actuator on another node (the resource discovery for node containing actuator is not shown in the chart).
3. The procedure is similar for LNs, except in step(1) above the IP address returned by M2MCE is that of PN, and the SNMP message is dynamically converted into ZigBee protocol by PN.

### 3.4.3 LN to WN

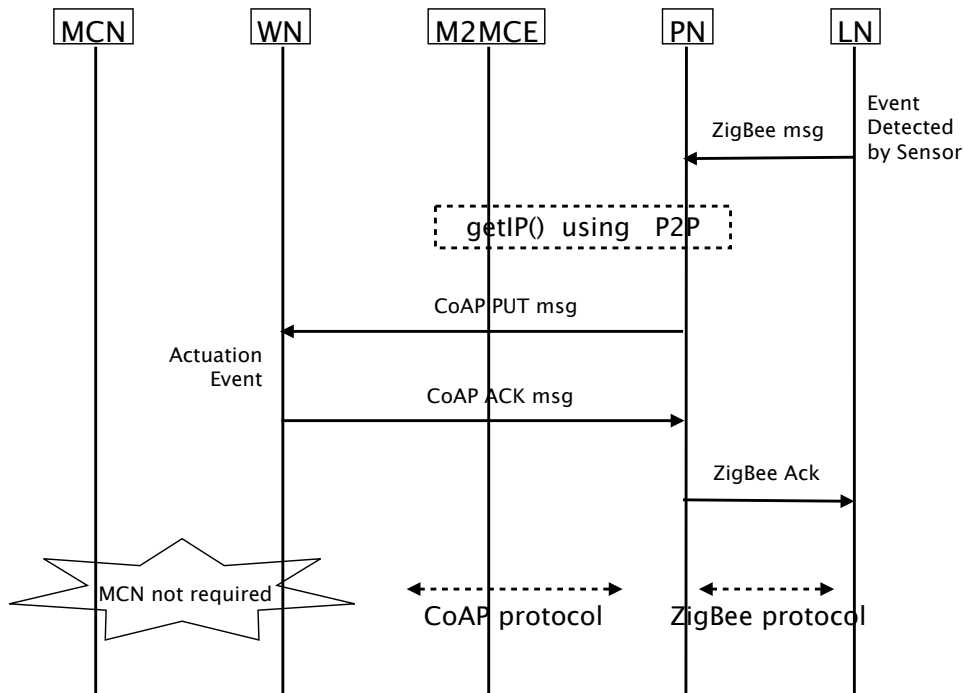


Figure 3.4: Information from LN towards WN

The figure 3.4 shows information sent from LN (in WPAN) to WN (in WWAN), via PN, using ZigBee and CoAP protocols.

1. When an event is detected at LN, it checks its MIB if any actuator association is configured with the particular sensor detecting the event. If an actuator is found, LN sends the event information in a ZigBee message to the target node using the ZigBee coordinator node (which is part of PN).
2. On receiving the ZigBee message, PN converts the ZigBee message into CoAP message and uses CoAP library to send it to the target node. CoAP library uses the M2MCE API to resolve the target node name into IP address, and then uses UDP protocol to send the message.
3. The target node receives the CoAP message and triggers the actuator using the information in the CoAP message. It then acknowledges the message by sending CoAP message to the PN.
4. On receiving the CoAP Acknowledgement, PN converts it to ZigBee Acknowledgement and sends it to the LN, completing the transaction.
5. It is worthwhile to note that MCN is not required in any step of this scenario, making the network self-reliant.

#### 3.4.4 WN to LN

The figure 3.5 shows information sent in the opposite direction of figure 3.4 i.e. from WN (in WWAN) to LN (in WPAN), via PN, using ZigBee and CoAP protocols.

1. When an event is detected at WN, it checks its MIB if any actuator association is configured with the particular sensor detecting the event. If an actuator is found, WN uses CoAP library to send the event information in a CoAP message to the target node.
2. Since the LN is represented by the PN, the CoAP message is received by the CoAP server on the PN. The CoAP server checks HOST\_URI field to identify the target LN for which the message is intended. The PN then converts the CoAP message into ZigBee message and uses zigbee-api library to send the message to LN.
3. The target LN finally receives the ZigBee message and triggers the actuator using the information in the message. It then acknowledges the message by sending ZigBee Ack message to the PN.

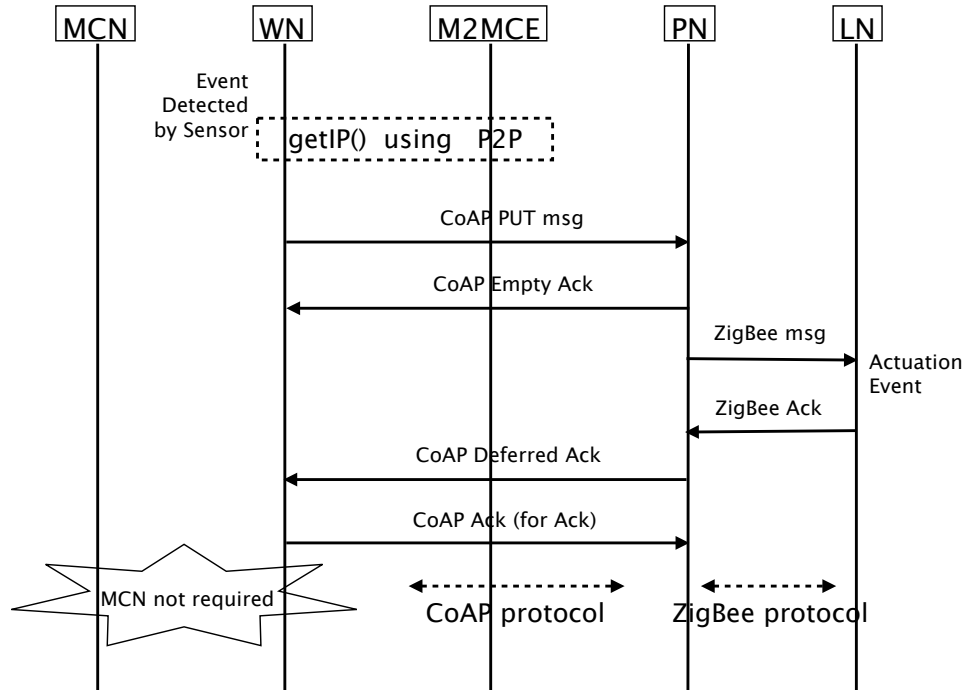


Figure 3.5: Information from WN towards LN

4. On receiving the ZigBee Acknowledgement, PN converts it to CoAP Acknowledgement and sends it to the source WN, completing the transaction. The PN uses CoAP Token to match the Requests and Acknowledgements.
5. Like the previous scenario, it is worthwhile to note that MCN is not required in any step of this scenario, making the network self-reliant.



## Chapter 4

# Prototype Implementation

This chapter describes the hardware and software used for the prototype testbed and the system startup details. We also give the rationale for choosing these specific hardware and software. In general, we have preferred to use open source software wherever possible.

### 4.1 Hardware specification

The prototype testbed uses the hardware described below. In order to duplicate our prototype testbed, we recommend using hardware with same specifications. In order to allow flexibility, we have used Java, Linux, ZigBee and Ardurino platforms which work on a wide variety of hardware, and hence minimal modifications should be required to run our software on different hardware.

#### 4.1.1 LN hardware

We have used ZigBee devices as LNs. Figure 4.1 shows the LN hardware. They are constituted of two part - a microcontroller board and a ZigBee RF module.

There are several vendors providing microcontroller boards that support ZigBee RF modules. Among them, we mainly considered 3 boards: Waspnote<sup>TM</sup> (from company called Libelium <sup>1</sup>), Mulle<sup>TM</sup> (from company Eistec <sup>2</sup>) and IRIS<sup>TM</sup> Mote (from company Crossbow <sup>3</sup>), and finally we se-

---

<sup>1</sup><http://www.libelium.com>

<sup>2</sup><http://www.eistec.se>

<sup>3</sup><http://www.xbow.com>

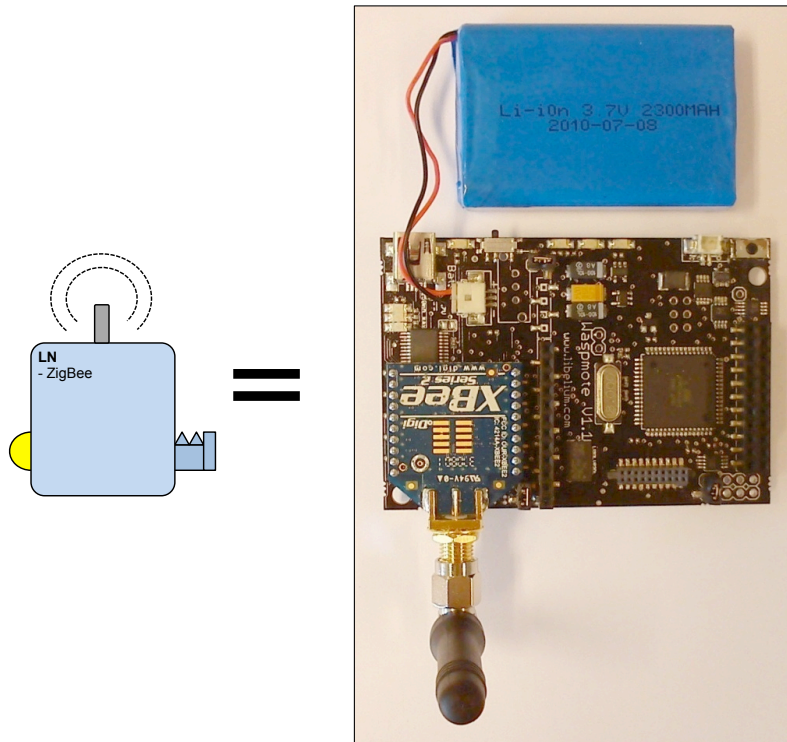


Figure 4.1: Local Node

lected Waspote<sup>4</sup> board because of the following reasons:-

- It comes with good development kit
- It uses Open Source APIs
- It has excellent documentation
- It comes with 3 ranges 2.4GHz - 7km, 900MHz - 24km, and 868MHz - 40km
- It consumes very little power : 0.7uA in Hibernate mode, 62uA in Deep Sleep mode and 9mA when ON.
- It has the possibility to add more than 50 sensors, giving a lot of flexibility

<sup>4</sup><http://www.libelium.com/products/waspote>

Waspote™ board has Atmel's <sup>5</sup> ATmega1281 microcontroller operating at 8MHz frequency with 8KB SRAM, 4KB EEPROM, and 128KB FLASH. It uses Arduino<sup>6</sup> platform, which comes with a IDE to develop, build and upload software onto the device. In addition to ZigBee, it supports Bluetooth and GPRS radio technologies.

To provide the ZigBee connectivity, we have used XBee™ ZB <sup>7</sup> ZigBee™ RF Modules from Digi International Inc <sup>8</sup> which are interoperable with other vendor's ZigBee RF modules. Additionally, these RF modules are compatible with Waspote microcontroller boards.

### 4.1.2 WN hardware

Figure 4.2 shows the WN hardware. We use single-board-computer (SBC) called Overo Earth™ <sup>9</sup> from the company Gumstix Inc <sup>10</sup>. Overo Earth are Linux based SBC which are available in a wide range of configurations (choices in DSP, Graphics acceleration, operating temperature ranges etc), and it is compatible with numerous expansion boards which can add capabilities like GPS, touch screen LCD, HDMI etc to the SBC. It uses OMAP™ 3503 Applications Processor from Texas Instruments <sup>11</sup> (which is based on ARM Cortex™ A8 CPU design). In addition, there is a development community around different products from Gumstix Inc providing a lot of open source software and technical discussion forums. We have used MicroSD card (on Overo Earth) to store the Linux image, which is very easy to clone, simplifying the procedure of adding extra nodes in the testbeds. All these factors make Overo Earth suitable for rapid prototyping.

The wireless connectivity is provided by using 3G UMTS dongle which is connected to the SBC using USB.

### 4.1.3 PN hardware

Figure 4.3 shows the PN hardware. PN uses the same hardware as WN, with the addition of a Waspote gateway device which is attached to the GumStix using USB. This gateway device, together with the zigbee-api library <sup>12</sup>,

---

<sup>5</sup><http://www.atmel.com>

<sup>6</sup><http://www.arduino.cc>

<sup>7</sup><http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module>

<sup>8</sup><http://www.digi.com>

<sup>9</sup>[http://www.gumstix.com/store/product\\_info.php?products\\_id=211](http://www.gumstix.com/store/product_info.php?products_id=211)

<sup>10</sup><http://www.gumstix.com>

<sup>11</sup><http://www.ti.com>

<sup>12</sup><http://code.google.com/p/xbee-api>

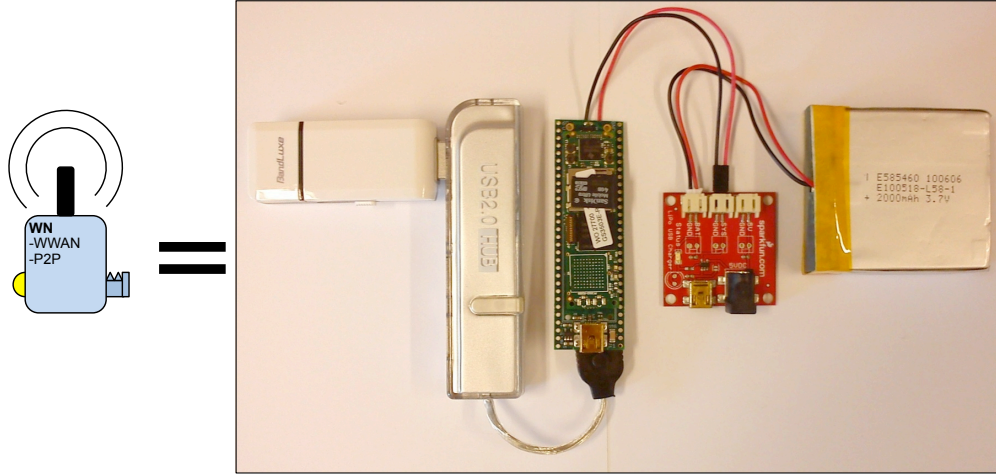


Figure 4.2: Wide Area Node

provides the interface to the WPAN. This interface is used by the CoAP and SNMP software modules to provide the Proxy functionality.

#### 4.1.4 MCN hardware

Dell Latitude D630 laptop is used as MCN. In addition, this laptop is also used as a Smart-M3 client. Another laptop (Dell Latitude D630) is used to run 3 entities - the bootstrap node for the P2P network, a set of simulated WNs, and the Smart-M3 SIB.

## 4.2 Software specification

We have used the following software, mostly open source, for the prototype implementation:

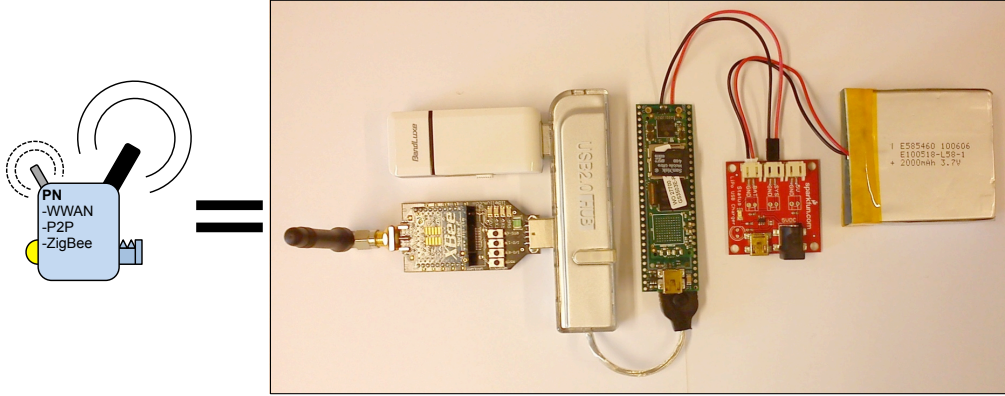


Figure 4.3: Proxy Node

- **SNMP4J** : SNMP4J <sup>13</sup> is an open source enterprise grade implementation of state-of-the-art SNMP using Java 2SE 1.6. It is used on WN and PN for implementing both SNMP-Agent and SNMP-Manager functionality, and on the MCN for SNMP-Manager functionality. The versions used are 1.11.3 and 1.4.3 for the SNMP-Manager and SNMP-Agent respectively.
- **JCoAP** : JCoAP <sup>14</sup> is an open source Java based implementation of CoAP protocol. We have used version 0.1, which is based on the IETF drafts draft-ietf-core-coap-05, draft-ietf-core-block-03, draft-ietf-core-link-format-04, and draft-ietf-core-observe-02. We had also considered jCoAP<sup>15</sup> (notice the small "j"), however, on evaluation we found that JCoAP was more feature- complete than jCoAP, hence we used JCoAP. One important feature that JCoAP lacked was the proxy functionality

<sup>13</sup><http://www.snmp4j.org>

<sup>14</sup><https://github.com/dapaulid/JCoAP>

<sup>15</sup><http://code.google.com/p/jcoap>

(as defined in section-5.3 of CoAP specification [7]), so it was added as part of this thesis.

- **zigbee-api library** : Libelium does not provide any library to interface with the ZigBee gateway device, hence we had to use a third party library. Zigbee-api library is an open source Java implementation for interfacing with Radio Frequency devices complying with XBee/XBee-Pro series 1 (802.15.4) and series 2 (ZNet 2.5 and ZB/ZigBee Pro). It requires RXTX <sup>16</sup> java library to communicate with ZigBee coordinator node using USB port.
- **P2P network based on DHT** : We have used Ericsson research's implementation of P2P network which is based on DHT. It uses Chord protocol to logically organize the peers into a circle. Chord protocol is the basis for RELOAD protocol [9], and it enables storage and messaging using the overlay.
- **Arduino** : Libelium Waspote development uses Arduino development environment. We used Waspote IDE version 2, which in turn is based on Arduino IDE. There are code examples for handling various sensors in Waspote which are very helpful in development.
- **Linux** : Linux embedded operating system is used on GumStix (used as WNs in the prototype). The flavour used is linux-gnueabi, which is ARM EABI Debian based port of Linux for the ARM architecture (named armel). `opkg` <sup>17</sup> based package management is used.
- **CACAO JVM** : CACAO <sup>18</sup> Java Virtual Machine (JVM) is developed by Vienna University of Technology with support for ARM processors. We chose CACAO over JamVM <sup>19</sup> because it supported the pre-existing DHT based P2P implementation out-of-the-box.
- **Smart-M3 Platform** : We have used Smart-M3 platform, which is created as part of DIEM project, to provide information level interoperability using Smart-M3 Knowledge Processes [8]. The version used is 0.9.5 [5].
- **Smart-M3 Java library** : Smart-M3 platform provides interface in C. Language bindings have been created for Java and Python to enable

---

<sup>16</sup><http://users.frii.com/jarvi/rxtx>

<sup>17</sup><http://code.google.com/p/opkg>

<sup>18</sup><http://www.cacaovm.org>

<sup>19</sup><http://jamvm.sourceforge.net>

software development in these languages. We have used the Java binding created by University of Bologna and Finnish Technical Research Centre VTT [16], version 5.5. The java binding internally uses JSON library.

It is worth noting that all the software is developed using Java, other than the C on Wasmotes (since Wasmotes do not support Java). The basic idea behind using Java is to allow easy porting to new (different) hardware, and given that M2M field is rapidly advancing, we expect to continually upgrade the testbed hardware.

### 4.3 Software Implementation

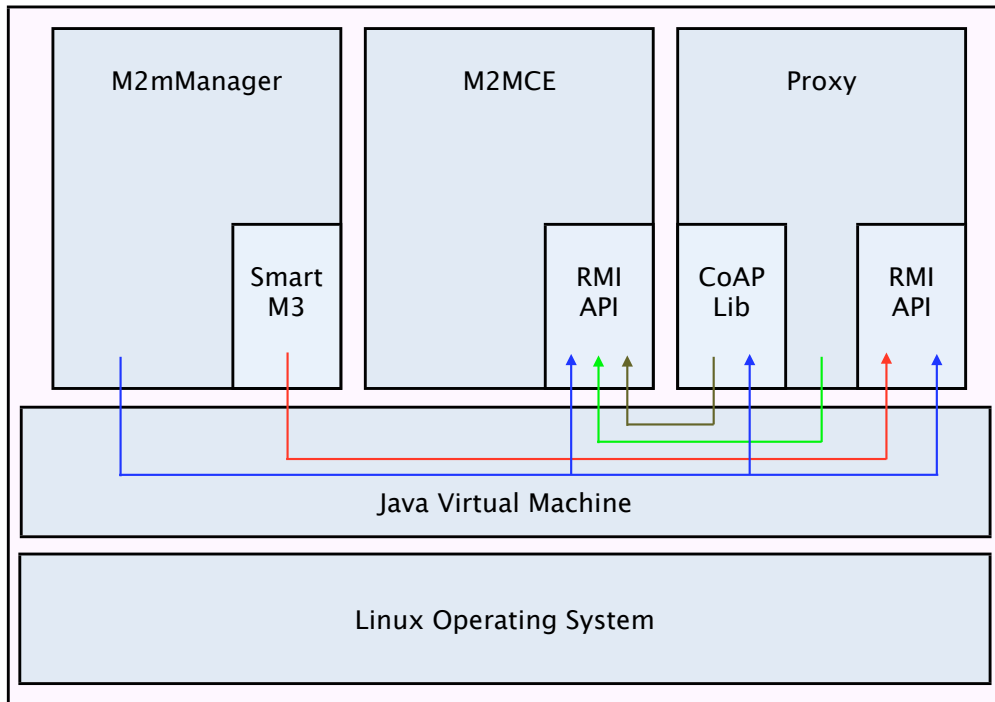


Figure 4.4: Software Architecture of PN

The high level software architecture of PN is shown in figure 4.4. The design of WN and MCN is subset of PN, so it is not explicitly discussed here. The software is split into 3 Java jar for modularity - m2mManager.jar,

proxy.jar and m2mce.jar - and they use Java Remote Method Invocation (RMI)<sup>20</sup> for communicating with each other. These 3 jar files are present in all nodes which allows the flexibility to change a PN into WN (and vice-versa) and to use any node as MCN. There are 6 significant modules contained in these 3 jar files:

1. M2MCE
2. Proxy
3. SNMP
4. Smart-M3 Client
5. CoAP client-server library
6. MCN command-line application

### 4.3.1 M2M System Startup

First the P2P bootstrap-peer device is started, followed by the rest of the devices in the network. Each node begins by launching m2mManager module, and then onwards the m2mManager module is responsible for further launching the other modules and configure the local node. The m2mManager module uses SNMP MIB information for this purpose.

M2mManager checks if there are any MIB values specified in the (optional) configuration file and loads these values, if any, into the MIB. Next, m2mManager checks the node-type (PN/WN/MCN), the node name, and the feature flag for Smart-M3, and accordingly does the following:

- If a node name is given in configuration file, m2mManager uses this name and the node-type to invoke the join() API of M2MCE module. M2MCE then contacts the bootstrap-peer and joins the P2P overlay. If no node name is specified in the configuration file, M2MCE module automatically assigns a name. This mechanism is used since in a network with large number of nodes, it would not be possible to individually assign names to all nodes; on the other hand, this offers capability to assign/modify node names via MCN.
- If node-type is PN, m2mManager launches the Proxy module; if the node-type is MCN, it instead launches the command-line module; otherwise, it does not launch any extra module.

---

<sup>20</sup><http://java.sun.com/j2se/1.5/pdf/rmi-spec-1.5.0.pdf>



- If the feature flag for Smart-M3 is enabled, m2mManager launches the Smart-M3 module. Then Smart-M3 module retrieves all the resources present in the node and publishes them in Smart-M3 SIB. If the node is a PN, Smart-M3 is responsible for discovering resources on the LNs and publishing them in Smart-M3 SIB. Additionally, Smart-M3 module subscribes, with the Smart-M3 SIB, for any requests for the registered resources.
- Lastly, m2mManager launches the CoAP module which, in turn, starts listening for any CoAP requests.

At the end of above steps, the system is up and functional. Now, the MCN commandline interface can be used for operations like node discovery, resource discovery, setting sensor-actuator associations, controlling log level etc. Similarly, Smart-M3 client can be used to retrieve information from the M2M devices.

## Chapter 5

# Conclusions and Future Work

This research project is part of work package for Devices and Interoperability Ecosystem (DIEM) <sup>1</sup> program funded by TEKES <sup>2</sup> - the Finnish Funding Agency for Technology and Innovation. The overall research project also involves close collaboration with the Future Internet program, which is part of ICT cluster of the Finnish Strategic Centres for Science, Technology and Innovation (ICT SHOK) <sup>3</sup>.

The project started with the study of possible use-cases for next generation M2M networks, followed by evaluation of several embedded hardware and software which could be used in our prototype testbed. Finally, the design and implementation for the prototype was carried out. To the best of our knowledge, this is the first research project which applies P2P principles to M2M.

### 5.1 Future work

A GUI development is currently ongoing as part of another student's Master's thesis. This will have generic M2M functionalities like displaying device's location on a geographical map, nodes discovery, resource discovery, using graphs and charts to display sensor / actuator data, setting parameters on devices etc. Pachube <sup>4</sup> is good example in this direction. The challenge here is to have a generic interface covering maximum envisioned M2M functionality (so that it is portable to other M2M/IoT systems), massive scalability, and enable real time operation.

---

<sup>1</sup><http://www.diem.fi>

<sup>2</sup><http://www.tekes.fi>

<sup>3</sup><http://www.futureinternet.fi>

<sup>4</sup><https://pachube.com>

This prototype will be used for further research, most notably in MAMOTH project which deals with large scale M2M network. Also, we are exploring the possibility of using the testbed for IoT-I<sup>5</sup> program.

---

<sup>5</sup><http://www.iot-i.eu>

# Bibliography

- [1] ISA100.11a Draft Standard release 1. Tech. rep., ISA100.11a Working Group, December 21, 2007.
- [2] WirelessHART Communication Standard. HART 7.0 Specifications, September 7, 2007.
- [3] APPLE. Ask Siri to help you get things done. <http://www.apple.com/iphone/features/siri.html>. Accessed Oct 1, 2011.
- [4] BOLONIO, J. A. J. Adapting a DHT (Distributed Hash Table) to a Self-Reliant M2M (Machineto- Machine) Network. Master's thesis, Aalto School of Science and Technology, Finland, 2011.
- [5] DIEM. Smart-M3 platform. <http://smart-m3.sourceforge.net>. Accessed Sept 1, 2011.
- [6] ERICSSON. More than 50 billion connected devices, taking connected devices to mass market and profitability. <http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf>. Accessed April 1, 2011.
- [7] FRANK, B., SHELBY, Z., HARTKE, K., AND BORMANN, C. Constrained Application Protocol (CoAP). Tech. Rep. draft-ietf-core-coap-08.txt, IETF Secretariat, Fremont, CA, USA, July 2011.
- [8] HONKOLA, J., LAINE, H., BROWN, R., AND TYRKKO, O. Smart-m3 information sharing platform. In *The 1st Int'l Workshop on Semantic Interoperability for Smart Spaces (SISS 2010)* (June 2010).
- [9] JENNINGS, C., LOWEKAMP, B., RESCORLA, E., BASET, S., AND SCHULZRINNE, H. Resource location and discovery (reload) base protocol. <http://tools.ietf.org/html/draft-ietf-p2psip-base-17>, July 2011. Work in progress.
- [10] LEVI, D., MEYER, P., AND STEWART, B. SNMPv3 Applications. RFC 2273 (Proposed Standard), Jan. 1998. Obsoleted by RFC 2573.

- [11] LI, D. A Proxy for Distributed Hash Table based Machine-to-Machine Networks. Master's thesis, Aalto School of Science and Technology, Finland, 2011.
- [12] MONTENEGRO, G., KUSHALNAGAR, N., HUI, J., AND CULLER, D. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard), Sept. 2007.
- [13] PRESUHN, R. Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). RFC 3416 (Standard), Dec. 2002.
- [14] SHELBY, Z., AND BORMANN, C. *6LoWPAN: The Wireless Embedded Internet*. John Wiley & Sons, Inc, 2010.
- [15] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference* (San Diego, California, August 2001).
- [16] UNIVERSITY OF BOLOGNA, AND VTT. Java-KPI interface for Smart-M3. <http://smartm3-javakpi.sourceforge.net>.