

Title	A study on remote management mechanism for M2M system
Author(s)	NGUYEN, Nam Hoang
Citation	
Issue Date	2016-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/13619
Rights	
Description	Supervisor:Yasuo Tan, School of Information Science, Master

A study on remote management mechanism for M2M system

NGUYEN, Nam Hoang

School of Information Science

Japan Advanced Institute of Science and Technology

March, 2016

Master's Thesis

A study on remote management mechanism for M2M system

1310252 NGUYEN, Nam Hoang

Supervisor : Professor Yasuo Tan

Main Examiner : Professor Yasuo Tan

Examiners : Associate Professor Yuto Lim

Professor Yoichi Shinoda

School of Information Science

Japan Advanced Institute of Science and Technology

February 2016

Abstract

Machine to machine (M2M) is a broad label that can be used to describe any technology that enables networked devices to exchange information and perform actions without the manual assistance of humans. The extensive growth of ubiquitous wireless networks worldwide has facilitated a system that allows for M2M on a drastically larger scale, enabling M2M communication at greater speeds using less power and resources thereby opening up a huge arena of potential opportunities spanning diverse industries and applications.

Smart home is a significant part of M2M system. During the last two decades, home network has developed very fast, becomes quite complex due to an increasing of number of devices and the coexistence of different kind of technologies in the same environment. Because of the complexity, topology information is required to help solving problems in the network.

Future home networks will also include sensors and controllers that allow end-users to interact with home systems controlling environment energy, appliances and security. ZigBee is one example of the upcoming technology, it has a low data rate, low power consumption, low cost, wireless networking protocol targeted towards automation and remote control applications.

HTIP (Home network Topology Identify Protocol) can identify the home network topology, and can check the connectivity of the target device and network devices present on the route between AGW and the device. But HTIP can only work on IP-based devices, other Non-IP devices, such as ZigBee are out of scope.

This leads to the combination demand of different standards. My research focuses on an example of combination between HTIP and ZigBee to draw a general view of home network topology, by proposing a new design and implementation of a ZigBee/LLDP Proxy, based on that to combine 2 different protocols: HTIP - ZigBee. From the characteristics of HTIP and ZigBee, we can make these technologies more suitable for humanity.

Acknowledgements

This work is presented in the context of a Master research in the Japan Advanced Institute of Science and Technology. It was carried out from February 2015 to February 2016 in Tan laboratory.

I would like to thank my supervisors Professor Yasuo Tan and Associate Professor Yuto Lim for their assistance and guidance during the twelve months of the master research. And my thanks to Professor Yoichi Shinoda for taking time to attend my defense and review my thesis.

I am thankful to my friends for their help during difficult times. And finally, I would like to thank my family for their continued support and love.

Contents

1	Introduction	1
1.1	Research Background	2
1.1.1	M2M System	2
1.1.2	Smart Homes	4
1.2	Motivation	5
1.3	Thesis Objective and Contributions	6
1.4	Thesis Outline	6
2	Related technology and standards	7
2.1	HTIP (TTC JJ-300.00 and JJ-300.01)	8
2.1.1	LLDP	10
2.1.2	UPnP	11
2.2	ZigBee	12
2.2.1	Overview	12
2.2.2	ZigBee Node Types	13
2.2.3	ZigBee stack architecture	14
3	Proposed Designed Model	16
3.1	System model	17
3.2	ZigBee model	19
3.3	LLDP model	20
3.4	UPnP model	21
3.5	LLDP-ZigBee proxy model	22
3.6	HTIP manager	23

4	Implemetation	24
4.1	ZigBee	25
4.2	LLDP	27
4.3	LLDP - ZigBee Proxy	34
4.3.1	Part 1: ZigBee function	35
4.3.2	Part 2: Proxy function	36
4.3.3	Part 3: LLDP function	39
4.4	UPnP	40
4.5	HTIP manager	41
5	Result and Evaluation	42
5.1	Result of LLDP - ZigBee Proxy	43
5.2	UPnP devices	45
5.3	HTIP manager	48
6	Conclusion and Future research	51

List of Figures

1.1	M2M architecture	2
1.2	SmartHome Environment	4
2.1	HTIP information transmitted	9
2.2	ZigBee Environment	12
2.3	Zigbee Stack architecture	14
3.1	Overview model	17
3.2	Designed model to implement	18
3.3	Designed ZigBee model	19
3.4	Designed LLDP model	20
3.5	Designed UPnP model	21
3.6	Designed LLDP-ZigBee Proxy model	22
3.7	Designed HTIP manager model	23
4.1	ZigBee implementation	25
4.2	LLDPDU frame structure	27
4.3	HTIP TLV structure	27
4.4	Correspondence of TTC Subtype values and the information to be stored .	28
4.5	Correspondence of IDs and the device information to be stored	28
4.6	TLV to store the device category	29
4.7	TLV to store the manufacturer code	30
4.8	TLV to store the model name	32
4.9	TLV to store the model number	33
4.10	Connection between Coordinator - Proxy	35

4.11	ZigBee device information table	35
4.12	ZigBee neighbor table	36
4.13	Mapping descriptions between LLDP and ZigBee	36
4.14	TLV structure to store ZigBee device information	37
4.15	TLV structure to store ZigBee neighbor information	38
4.16	Correspondence of device information and elements in UPnP	40
5.1	Result of the showing LLDP devices	43
5.2	Result of the showing ZigBee devices	44
5.3	UPnP device	45
5.4	UPnP device - TV 1	46
5.5	UPnP device - TV 2	47
5.6	Devices information	48
5.7	Link information	49
5.8	Network Topology	50
5.9	Evaluation	50

Chapter 1

Introduction

1.1 Research Background

1.1.1 M2M System

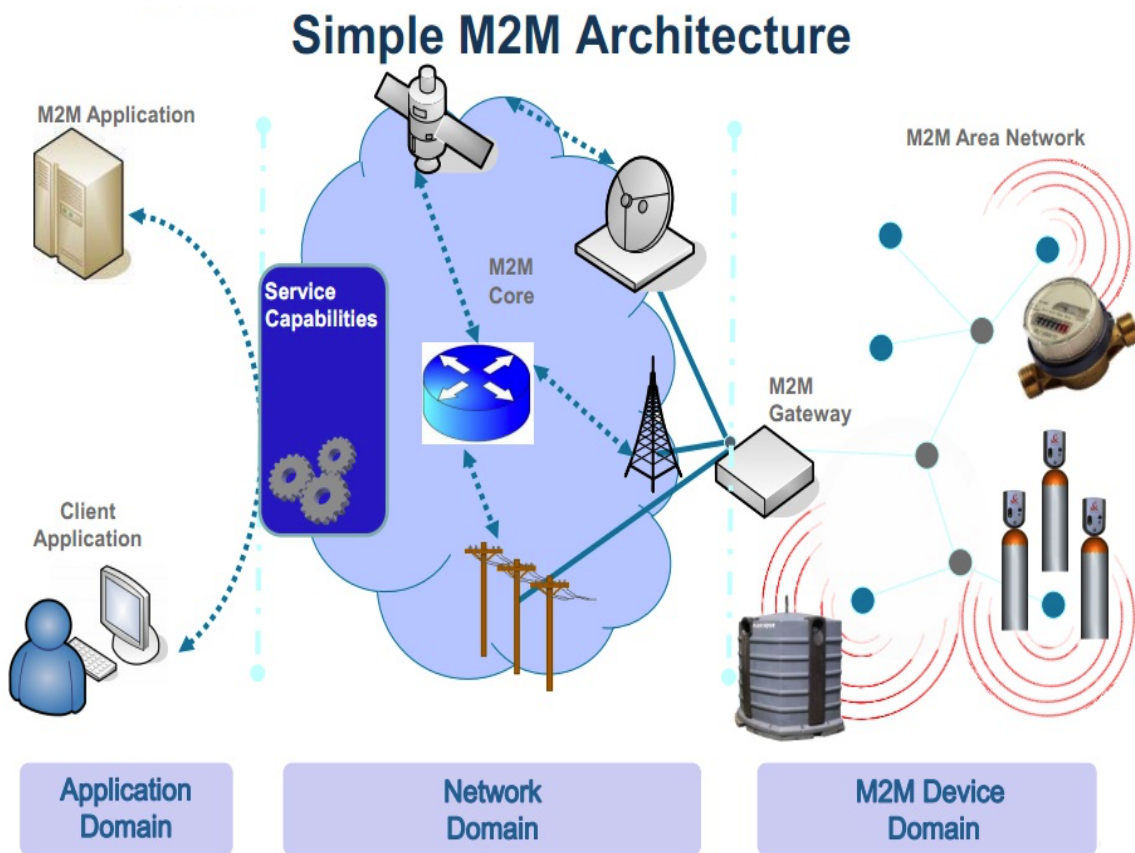


Figure 1.1: M2M architecture

An M2M system includes sensors, RFID, a Wi-Fi or cellular communications link and autonomic computing software programmed to help a networked device interpret data and make decisions. The idea of M2M communications is to enable M2M components to be interconnected, networked, and controllable remotely, with low-cost, scalable, and reliable technologies.

In the image above, we can see 3 main domain of M2M architecture [1], such as:

- M2M Application Domain.
- M2M Network Domain.
- M2M Device Domain.

M2M Applications Domain provides application for M2M technology, such as server applications and end-user applications.

M2M Device Domain contains devices that can connect to M2M Network domain. M2M Device Domain can be called as M2M Area Network. We must refers to market requirements for choose the devices for M2M Device Domain. Diverse technologies can be used to support various applications.

M2M Network Domain provides communication network between M2M Application Domain and M2M Device Domain. M2M Network Domain consist two basic part, such as M2M core and M2M service capabilities. The M2M core network is the central part of the M2M communication network that provides various services to service providers connected via the access network such as GERAN, WiMAX, Satellite, DSL, UTRAN, WLAN or eUTRAN and other mobile network (e.g. 3G, 2G, LTE, 4G). M2M service capabilities are network functions defined to support M2M applications.

1.1.2 Smart Homes

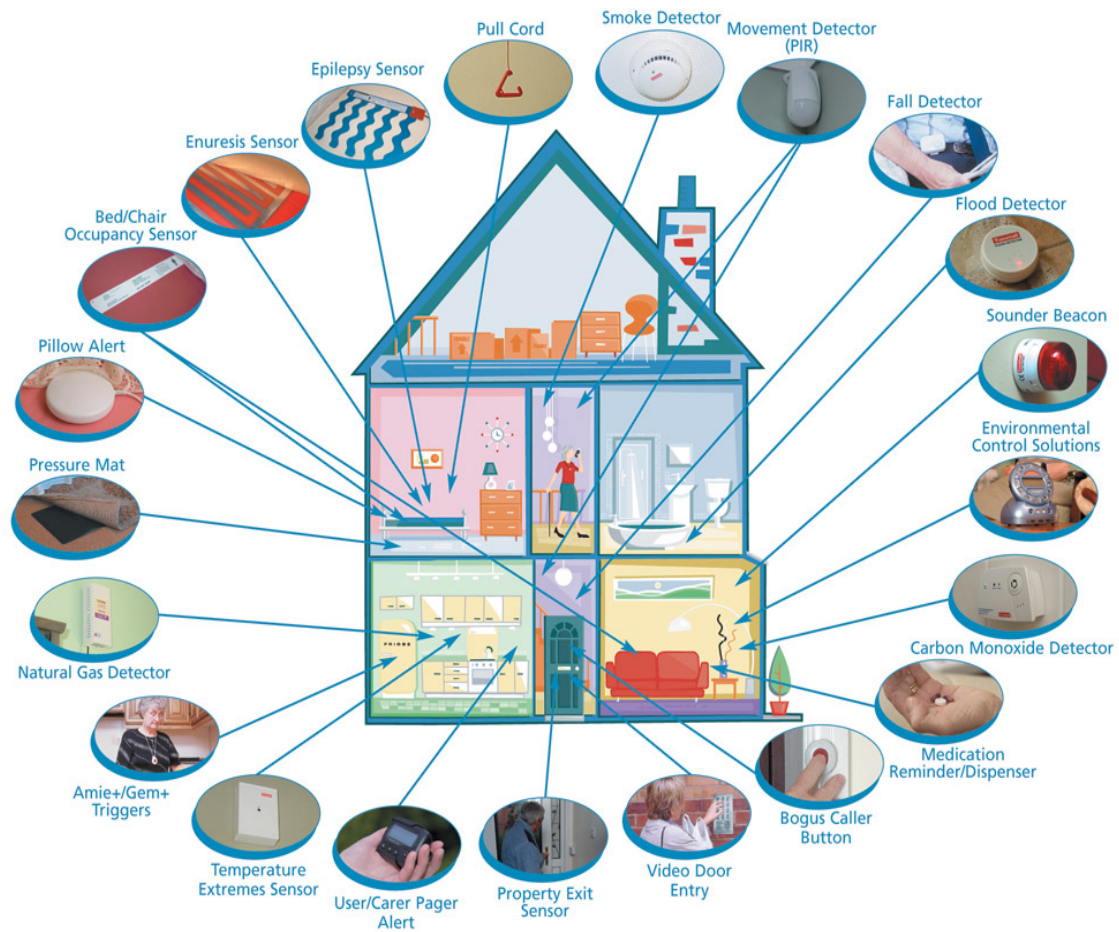


Figure 1.2: SmartHome Environment

Smart home is a significant part of M2M system. It is a living environment that incorporates the appropriate technology, called Smart Home technology, to meet the resident goals of comfort living, life safety, security and efficiency. Smart Home technology started for more than a decade to introduce the concept of networking devices and equipment in the house. It is a home automation system that allows for controlling over a home environment, media systems, home security, and integrates with an easily accessible user interface through home network.

The devices and systems in Smart Home environment can communicate with each other and can be controlled automatically in order to interact with the household members and improve the quality of their daily life.

1.2 Motivation

During the last two decades, home network has developed very fast, becomes quite complex due to an increasing of number of devices and the coexistence of different kind of technologies in the same environment. Because of the complexity, topology information is required to help solving problems in the network. Topology information includes:

- A list of active devices and their capabilities.
- A map of connections between devices.
- Technologies used in the home network on a per-connection basis.

HTIP (Home network Topology Identify Protocol) can identify the home network topology, and can check the connectivity of the target device and network devices present on the route between a public-Access Gateway and the devices.

ZigBee technology is a low data rate, low power consumption, low cost, wireless networking protocol targeted towards automation and remote control applications. ZigBee technology builds on IEEE standard 802.15.4 which defines the physical and MAC layers. Above this, ZigBee defines the application and security layer specifications enabling interoperability between products from different manufacturers.

But HTIP can only work on IP-based devices, other Non-IP devices such as ZigBee devices are out of scope. This leads to the combination demand of different standards,

and my research focuses on an example of combination between HTIP and ZigBee to draw a general view of home network topology.

1.3 Thesis Objective and Contributions

The purpose of this paper is to design and to implement ZigBee - LLDP proxy, based on that to combine 2 different protocols: HTIP - ZigBee. We want to use the characteristics of HTIP and ZigBee, to make these technologies more suitable for humanity.

The contribution of this research is divided into 2 folds:

- Design a system model, which contains HTIP and ZigBee in smart house to test;
- Propose a solution to solve the difference of 2 protocols, that is creating a LLDP - ZigBee proxy, then implement and evaluate this solution.

1.4 Thesis Outline

- Chapter 1: explains the research background, motivation, objective and contributions.
- Chapter 2: gives some related work, shows the related standard and technology, give basic knowledge.
- Chapter 3: propose the designed model to test.
- Chapter 4: implement the proposed model.
- Chapter 5: evaluate the results.
- Chapter 6: conclusion and idea for future research.

Chapter 2

Related technology and standards

2.1 HTIP (TTC JJ-300.00 and JJ-300.01)

Home-network topology identifying protocol (HTIP) used to identify the topology of a home network. This HTIP only applies to the broadcast domain in the link layer. Manager, the software which identifies the home-network topology, can be located on any terminal in the home network. [2] [3]

HTIP includes:

- IP Terminal: A hardware terminal that has an IP address and terminates IP packets.
- HTIP-IP Terminal: An IP Terminal in which L3 Agent is installed. One or more L3 Agents can reside on one HTIP-IP Terminal.
- Ethernet Bridge: A hardware device that has two or more ports and a function to transfer frames and packets received at one port to the other port by using a MAC forwarding table. Some types of Ethernet Bridge connect different transmission media to each other. Ethernet Bridges exclude the devices (e.g., repeater hubs) that do not use a MAC forwarding table even when they have a frame and packet transfer function.
- HTIP-Ethernet Bridge: An Ethernet Bridge in which L2 Agent is installed. Only one L2 Agent can reside on one HTIP-Ethernet Bridge. L3 Agent can also reside on an HTIP-Ethernet Bridge.
- Manager: A software program that collects device and link information from individual Agents and identifies the topology of home network. Manager also checks its connectivity with individual Agents. Manager can operate on either an IP Terminal or an Ethernet Bridge.
- L3 Agent: A software program that resides on an HTIP-IP Terminal and transmits device information to Manager.
- L2 Agent: A software program that resides on an HTIP-Ethernet Bridge and transmits device and link information to Manager.

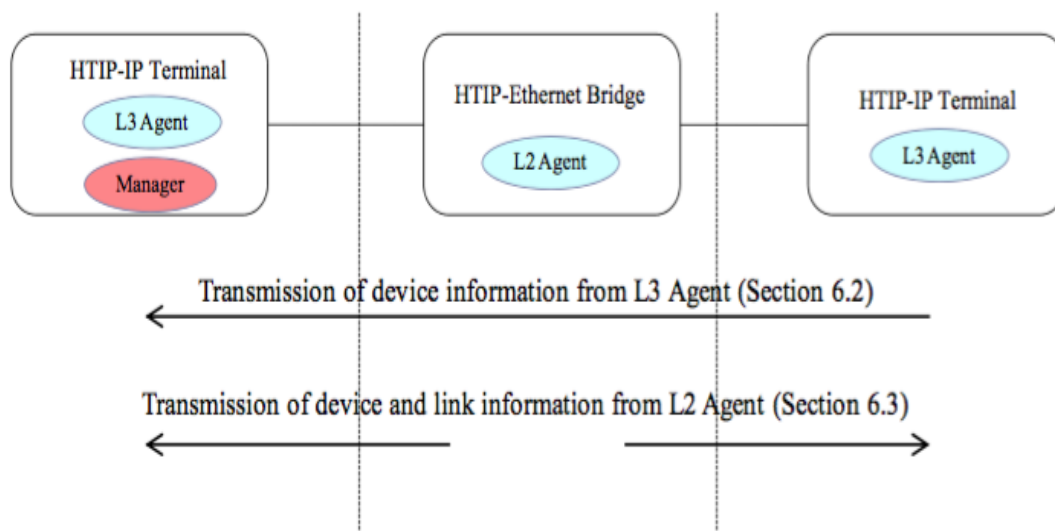


Figure 2.1: HTIP information transmitted

Each HTIP-IP Terminal which incorporates the controlled device functions defined by the UPnP Device Architecture [3] transmits device information to Manager. Each HTIP-Ethernet Bridge which incorporates the LLDP Agent functions (transmit-only mode) defined by IEEE 802.1AB [4] transmits both device information and link information (a MAC forwarding table) in the home network. Manager collects the device information and link information from HTIP-IP Terminals and HTIP-Ethernet Bridges in the home network and identifies the home network topology on the basis of the collected information.

2.1.1 LLDP

2.1.1.1 Overview

L2 Agent must transmit device and link information to Manager by using LLDP [5]. There is also another possible method to use LLDP only for transmission of the address of an HTIP-Ethernet Bridge on which L2 Agent resides and use SNMP for transmission of device and link information. The possible method might be defined in the next or later version of HTIP. This version of HTIP uses only LLDP to transmit device and link information.

The Link Layer Discovery Protocol (LLDP) is an industry-standard, vendor-neutral method to allow networked devices to advertise capabilities, identity, and other information onto a LAN. The Layer 2 protocol, detailed in IEEE 802.1AB-2005, replaces several proprietary protocols implemented by individual vendors for their equipment.

2.1.1.2 How LLDP works

LLDP compliant network devices regularly exchange LLDP advertisements with their neighbors and store it in their internal database (MIB). A Network Management Software (NMS) can use SNMP to access this information to build an inventory of the network devices connected on the network, and for other applications. LLDP advertisements can be sent to received from devices that are directly connected with each other through a physical link. [6]

2.1.1.3 LLDP on HTIP

LLDP information is sent by devices from each of their interfaces at a fixed interval, in the form of an Ethernet frame. Each frame contains one LLDP Data Unit (LLDPDU). Each LLDPDU is a sequence of type-length-value (TLV) structures. The detailed information of LLDPDU frame will be explained more in section 4. [13]

2.1.2 UPnP

2.1.2.1 Overview

The UPnP architecture defines peer-to-peer network connectivity of intelligent appliances, devices, and control points. It is designed to bring easy-to-use, flexible, standards-based connectivity to ad-hoc, managed, or unmanaged networks, whether these networks are in the home, small businesses, or attached directly to the Internet. The UPnP architecture is a distributed, open networking architecture that uses existing TCP/IP and Web technologies to enable seamless proximity networking, in addition to control and data transfer among networked devices.

UPnP is an IP-based protocol suite based on preliminary versions of Web Services protocols such as XML and Simple Object Access Protocol (SOAP). With UPnP, a device can dynamically join a network, obtain an IP address, convey its capability, and discover the presence and capabilities of other devices on the network. [4]

2.1.2.2 How UPnP works

6 working steps : [4]

- Device Addressing: Determining whether to use Auto-IP, Choosing an address, Testing the address, Periodic checking for dynamic address availability, Device naming and DNS interaction, Name to IP address resolution.
- Discovery: Advertisement, Search.
- Description: Device description, Service description, Retrieving a description.
- Control: Protocols, Action, Query for variable.
- Eventing: Subscription, Event messages.
- Presentation.

2.1.2.3 UPnP on HTIP

On HTIP, L3 Agent in a home network transmits device information by using UPnP controlled device functions [13] to Manager residing on a device in the home network. The

UPnP controlled device functions defined by UDA must be implemented on L3 Agent.

This specification defines some new elements to be added in the Basic Device Information part in the DDD described in UDA and a policy for describing values in existing elements to transmit device information to Manager. These new elements will be explained more in section 5.

2.2 ZigBee

2.2.1 Overview



Figure 2.2: ZigBee Environment

ZigBee is created by an organization named ZigBee Alliance. The goal of the ZigBee Alliance is to provide the consumer with ultimate flexibility, mobility, and ease of use by building wireless intelligence and capabilities into everyday devices.

ZigBee is a standard for low-power Wireless Personal Area Networks (WPANs), which is to say wireless networks with a short range, typically 10-100 meters. ZigBee is commonly used for wireless control and monitoring applications such as wireless sensor networks (WSNs), industrial plant monitoring, building control, hospitals, smart metering and home automation. There are actually public profiles defined in the ZigBee specification for many of these use cases.

ZigBee operates in the Industrial, Scientific and Medical (ISM) radio bands and the exact frequency will depend where you are in the world. It can use the 868 MHz band in much of Europe, 915 MHz in the USA and 2.4 GHz in many other locations. The 2.4 GHz band is very common as many of the available chipsets use it. The speeds available depend on which band you are using, but the maximum is 250 Kbps. This is slower than other popular wireless technologies such as WiFi but is also cheaper and lower cost.

2.2.2 ZigBee Node Types

There are three node types that a device can act as within a ZigBee network. Whatever node type a device is acting as it can also be doing some useful work such as acting as a sensor. Node types are only relevant to the topology of the ZigBee network and how devices help to route messages. The available node types are described below: [10]

- **Coordinator:** Every ZigBee network must have a single Coordinator. This node is the first node to start up and initialises the rest of the network, selecting the frequency to use, the PAN ID of the network, and allowing other nodes to join the network. It acts as the parent to nodes that connect to the network through it (its children). The Coordinator also often runs other services such as routing and certain security services, although many of these services are options and some can be run on separate dedicated nodes.
- **Router:** Routers are not required in all ZigBee topologies but are still commonly found. They are responsible for relaying messages to other nodes. Nodes can also join the

network via a Router, with the Router becoming their parent node; this can include one Router being the parent of another Router.

- End Device: An End Device is a simple node that sends and receives messages but performs no other special function in the network. Other nodes cannot join the network through them. End Devices are the only nodes that can sleep according to the ZigBee specification with the parent node (a Router or Coordinator) buffering messages until it wakes up again.

2.2.3 ZigBee stack architecture

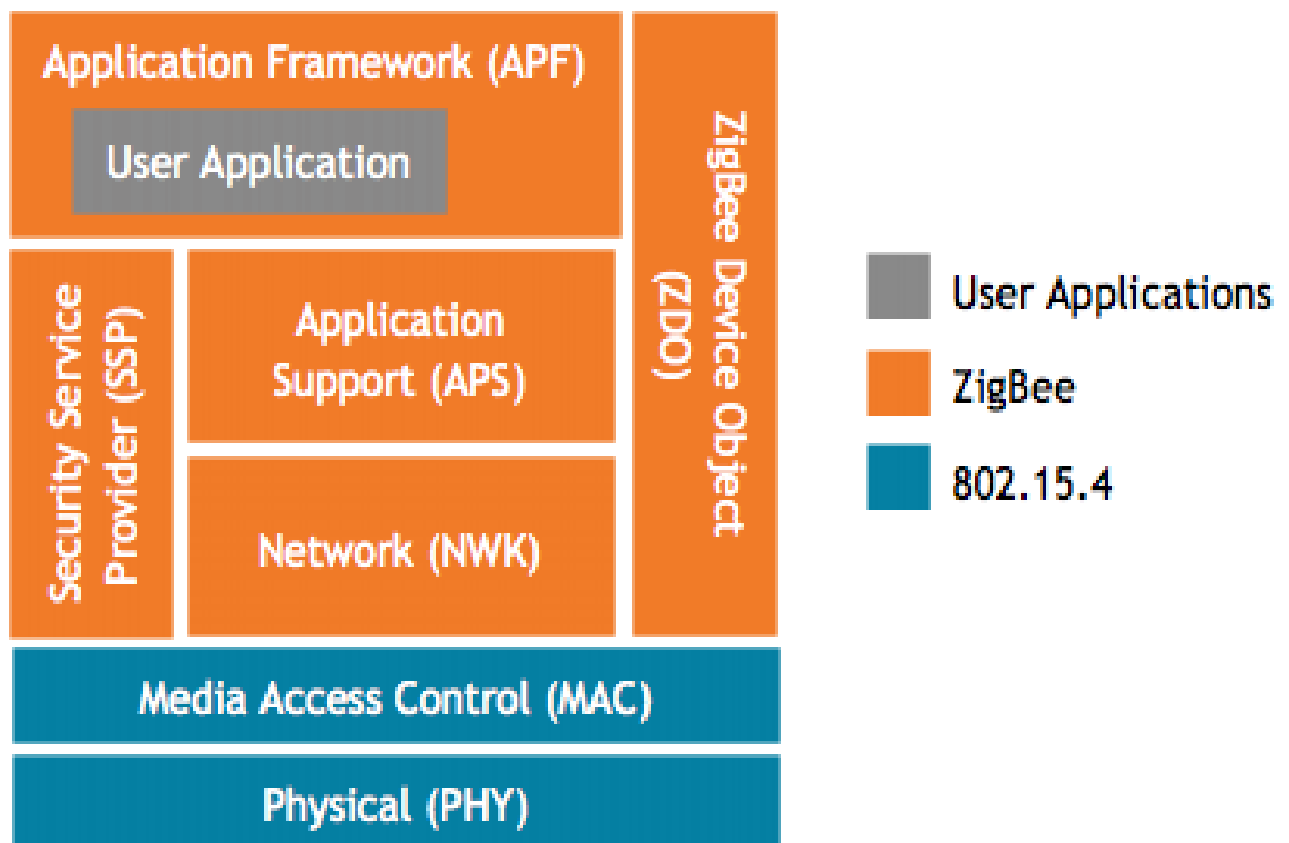


Figure 2.3: Zigbee Stack architecture

- PHY: Defined by 802.15.4 the PHY layer is responsible for the modulation, demodulation and physical transmission of packets over the air and handles various things needed for robust radio transmission in noisy, interference prone environments.
- MAC: Also defined by 802.15.4 and similar to MAC layers in other protocols, ZigBee does not actually use all of its features. This layer performs functions such as CSMA/CA to avoid collisions when transmitting frames and defines a frame format with things like MAC addresses etc. The MAC layer also defines network topologies which ZigBee builds upon and enhances at higher levels of the stack.
- NWK: A ZigBee layer that builds on 802.15.4 and is one of the more complex ZigBee layers. It provides the ability to discover and join networks and expands on the topologies defined by 802.15.4 at the MAC layer to allow mesh networking, a popular feature of ZigBee. The NWK layer also determines routes through the ZigBee network and supports ZigBee addresses which are different to the MAC addresses present at the MAC layer.
- APS: A ZigBee layer implementing features needed by ZigBee applications and acting as an interface to the NWK layer. It performs some filtering of duplicate packets from the NWK layer and maintains a binding table of nodes in the network.
- SSP: Provides ZigBee security services to the NWK and APS layers including key establishment and transport, device management and frame protection.
- ZDO: Responsible for the overall management of the ZigBee device. The ZDO initialises the APS and NWK layer, allows device discovery, manages binding requests and defines the device mode (coordinator, router or end device).
- APF: The APF is an execution environment for ZigBee user applications and facilitates sending and receiving of data by those applications. It also provides an Endpoint for each application, with Endpoint 0 being reserved for the ZDO and Endpoint 255 for a broadcast address. Applications themselves implement the function of the ZigBee device (eg. a sensor). [10]

Chapter 3

Proposed Designed Model

3.1 System model

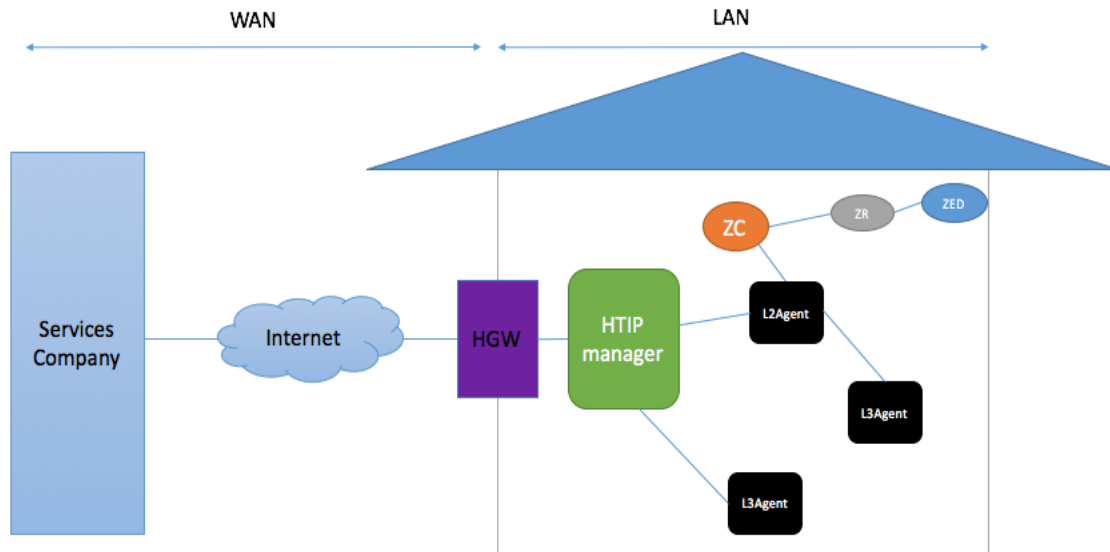


Figure 3.1: Overview model

As the figure 3.1, the system model has 2 parts: on the right is the WAN part, this part is out of scope, on the left is the LAN part, in this paper, I will focus only in this part.

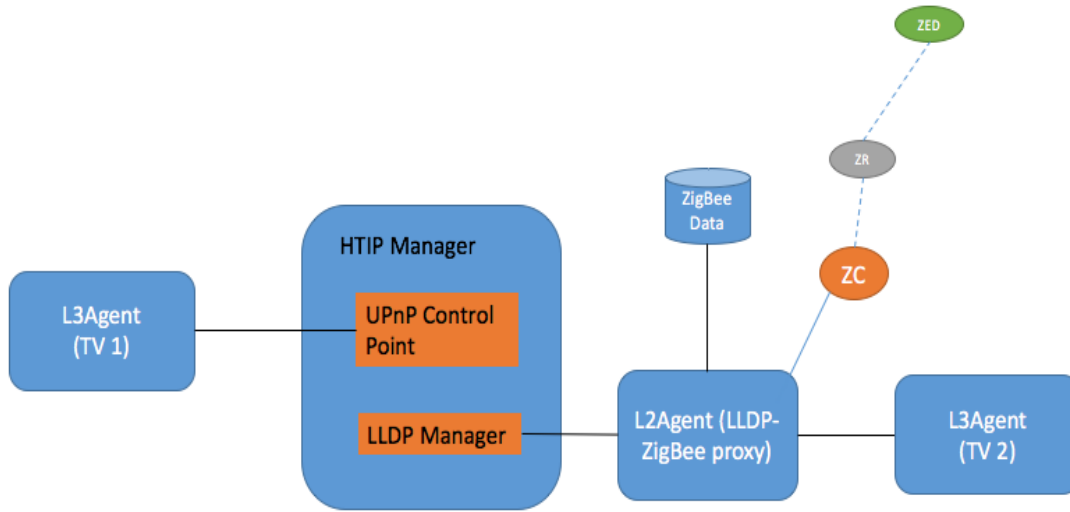


Figure 3.2: Designed model to implement

The designed system model is composed by 2 parts:

Part 1: HTIP model includes LLDP model (L2 Agent) and UPnP Model (L3 Agent). LLDP model has LLDP Agent and LLDP Manager. UPnP model has virtual UPnP TV 1, virtual UPnP TV 2 and UPnP control point.

Part 2: ZigBee system model and LLDP - ZigBee Proxy model.

3.2 ZigBee model

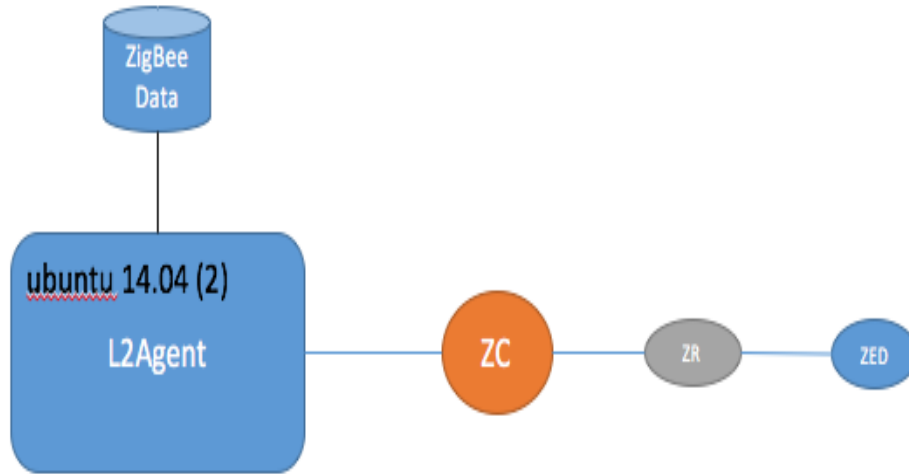


Figure 3.3: Designed ZigBee model

Using Jennic 5139 Kit.

ZigBee system contains 1 ZC, 1 ZR and 1 ZED.

ZC communicates with virtual Computer(ubuntu 14.04 (2)) through RS232 cable serial port .

ZigBee information will be saved into SQLite3 database.

3.3 LLDP model

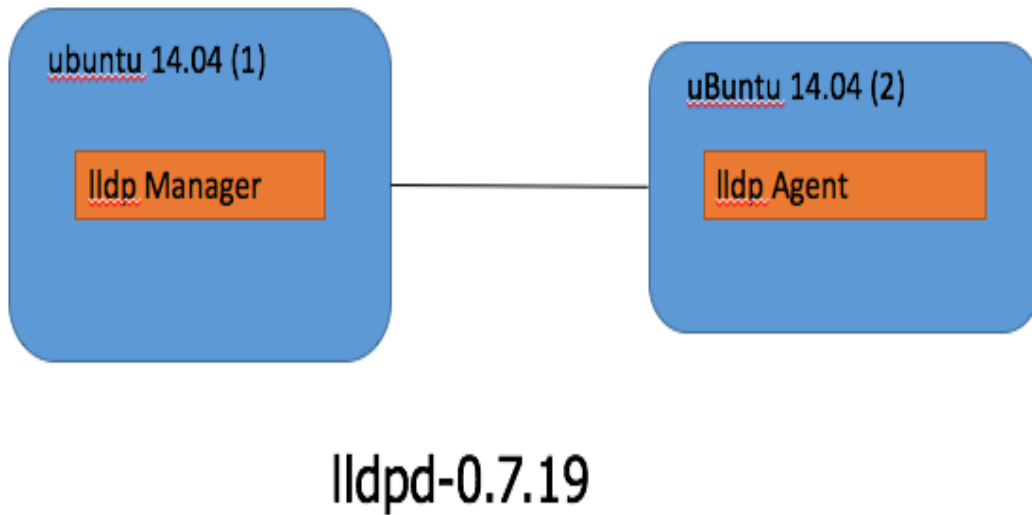


Figure 3.4: Designed LLDP model

LLDP Agent and LLDP Manager are implemented in 2 ubuntu 14.04 virtual Computer. LLDP Agent is in ubuntu 14.04 (1) and LLDP Manager is in ubuntu 14.04 (2).

3.4 UPnP model

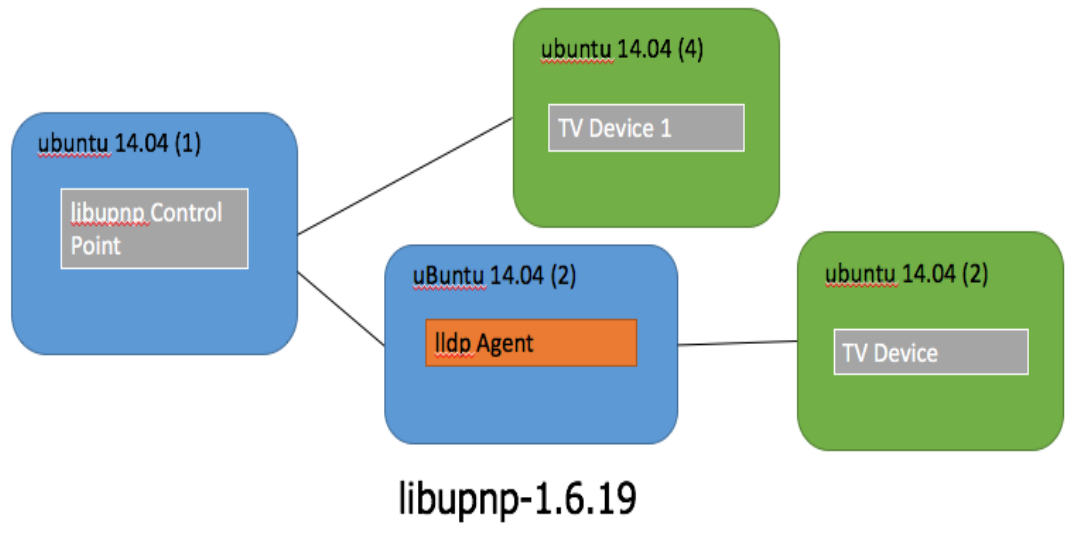


Figure 3.5: Designed UPnP model

UPnP is implemented as 2 virtual TV devices:

- TV device 1 in ubuntu 14.04 (4), connect directly with Control point in ubuntu 14.04 (1)
- TV device 2 in ubuntu 14.04 (3) , connect undirectly with Control point through L2 Agent - ubuntu 14.04 (2).

3.5 LLDP-ZigBee proxy model

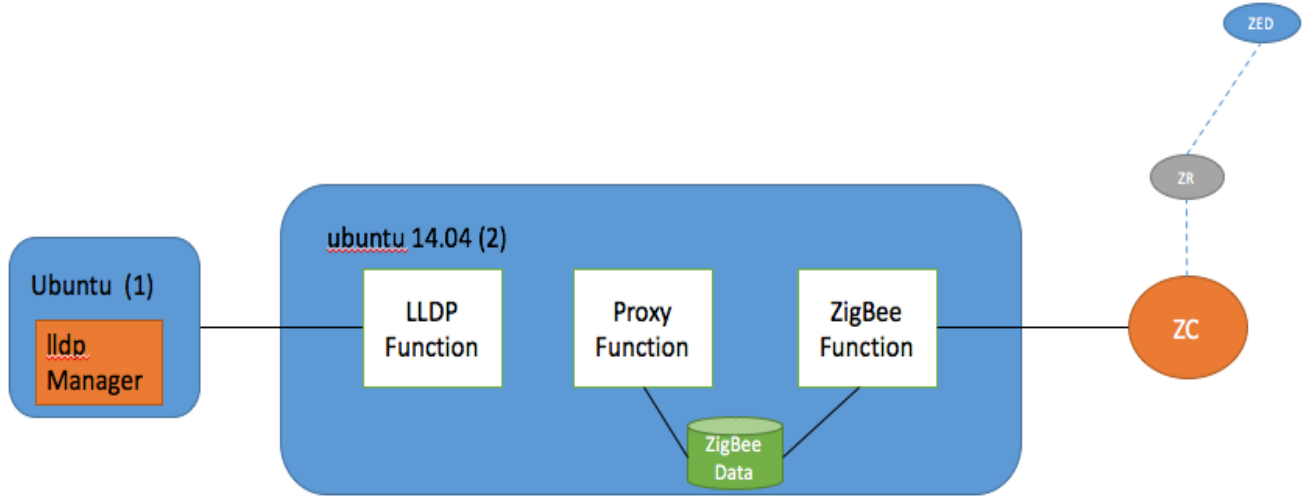


Figure 3.6: Designed LLDP-ZigBee Proxy model

The LLDP - ZigBee Proxy is implemented in ubuntu 14.04 (2). It includes 3 parts:

- ZigBee function: connect with ZigBee network and get data, push into database.
- Proxy function: transfer ZigBee data from database into LLDP frame.
- LLDP function : connect and transmit data to the LLDP Manager

3.6 HTIP manager

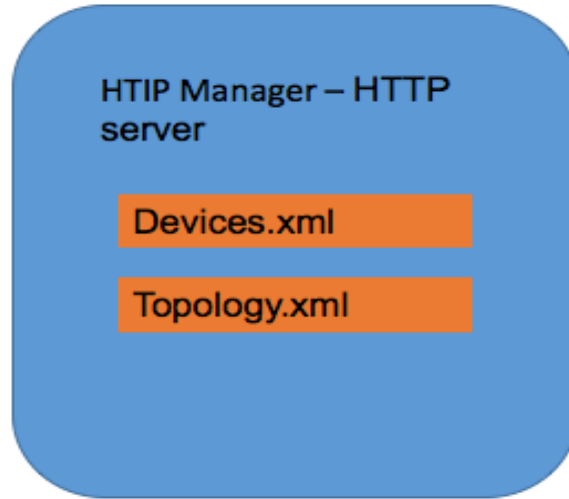


Figure 3.7: Designed HTIP manager model

The HTIP manager is a HTTP server, it collects information from L2 Agent and L3 Agent, then create 2 XML files: Devices.xml and Topology.xml to draw the home network topology.

Chapter 4

Implemetation

4.1 ZigBee

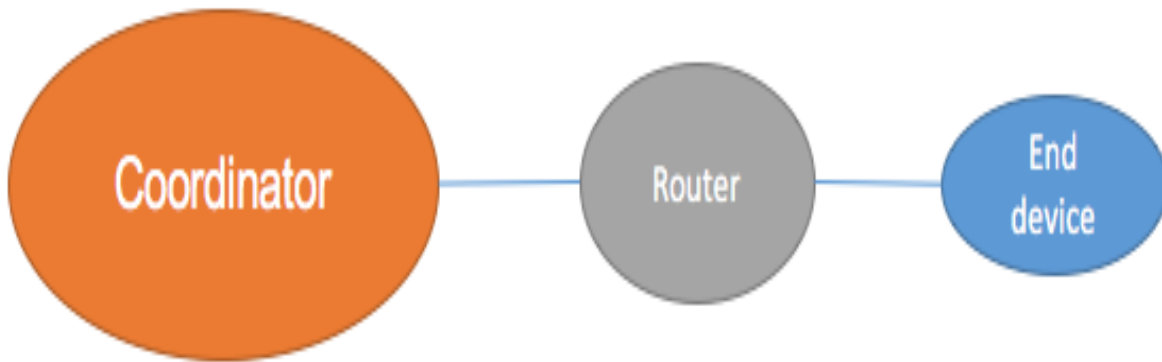


Figure 4.1: ZigBee implementation

Base on Jennic application development environment (Codeblock IDE), I use Jennic 5139 Kit to create implementation of a multi-hop mesh network used for the remote monitoring of ZigBee device information and neighbor information.

The ZigBee network contains 1 Coordinator, 1 Router and 1 End device.

ZigBee device transmit data to the nearest parent.

ZigBee Router read data from the on-board sensors and transmit this data, via the radio, to the network coordinator. They route messages received from other devices to the network coordinator, allowing the creation of a multi-hop mesh network.

The ZigBee Coordinator is controller board from the evaluation kit. The software running on this board simply receives messages from the Router and sends the data to a Proxy via the UART. Data is transmitted at 9600-8-N-1 and, when using HyperTerminal, is displayed as follows:

- Sensor 1:

Device information:

Profile ID = 0x291

Manufacture code = 0x4151

Device name = Jennic 5139

Device ID = 0x13

Network address = 0x0001

IEEE address = 0x00158d0000069907

Neighbor information:

Neighbor 1's information:

IEEE address = 00158d0000069580

Relationship = 0x00

IEEE address = 0x00158d0000069a46

Relationship = 0x01

with

- Profile ID: is the 16-bit identifier of the ZigBee application profile supported by the endpoint. This must be an application profile identifier issued by the ZigBee Alliance.
- Manufacturer Code: contains 16 bits (bits 0-15) indicating the manufacturer code for the node, where this code is allocated to the manufacturer by the ZigBee Alliance.
- Device ID: is the 16-bit identifier of the ZigBee device description supported by the endpoint. This must be a device description identifier issued by the ZigBee Alliance.
- Network address: This 16-bit address identifies the node in the network and is local to that network (thus, two nodes in separate networks may have the same network address). It is sometimes called the short address.
- IEEE address: This is a 64-bit address, allocated by the IEEE, which uniquely identifies the device - no two devices in the world can have the same IEEE address. It is often referred to as the MAC address and, in a ZigBee network, is sometimes called the extended address.

- Relationship: Relationship between this node and nearest nodes. Values 0x00: Parent; 0x01: Child; 0x02: Sibling; 0x03: None. [11]

4.2 LLDP

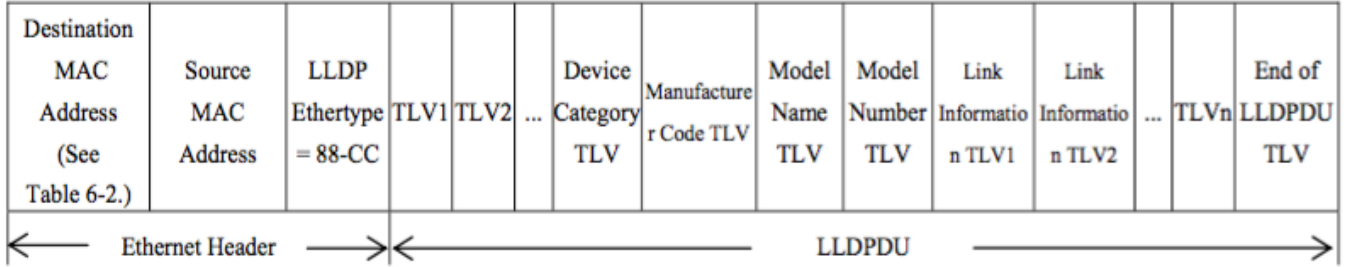


Figure 4.2: LLDPDU frame structure

Figure 4.2 draws the LLDPDU frame. Each LLDPDU is a sequence of type-length-value (TLV) structures as figure 4.3.

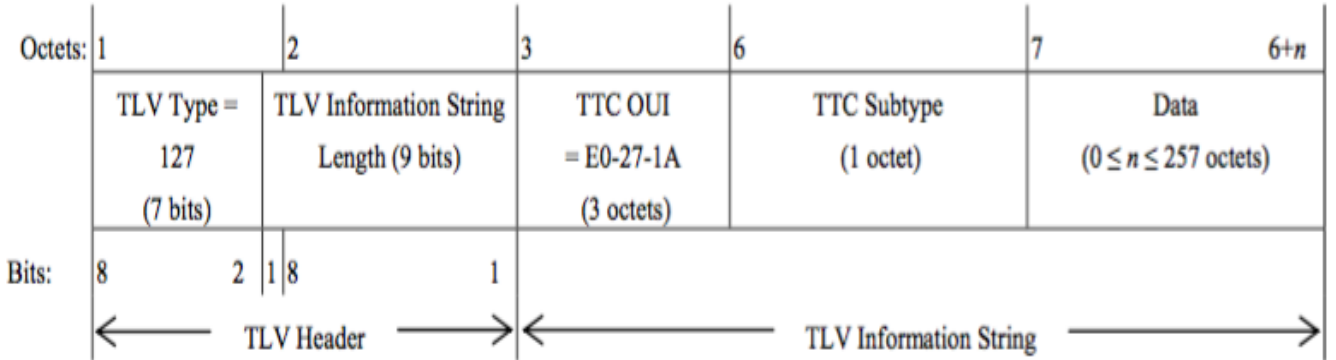


Figure 4.3: HTIP TLV structure

L2 Agent uses the vendor-specific extension fields (TLV Type = 127) defined by IEEE 802.1AB. These TLVs contain the TTC OUI code "E0-27-1A" and the information defined by TTC. The length of strings in a TLV must be represented in units of octet. For details of the notation method, see IEEE 802.1AB [7] .

Using opensource software lldpd-0.7.19 [14] and related documents [12], the HTIP frame is built as follow:

LLDPDU transmitted from LLDP Agent must always include not only the TLVs containing device information but also the link information.

TTC Subtype	Data	Required, recommended, or optional	Reference section
1	Device information	Required	Section 6.3.2
2	Link information	Required	Section 6.3.3
3	MAC address list	Optional	Section 6.3.4
0 or 4-255	Reserved		

Figure 4.4: Correspondence of TTC Subtype values and the information to be stored

Figure 4.4 lists the values of the TTC Subtype. When the TTC Subtype is "1", the TLV indicates device information. When the TTC Subtype is "2", the TLV indicates link information. When the TTC Subtype is "3", the TLV indicates Mac address list.

Device information ID	Device information	Required, recommended, or optional
1	Device category	Required
2	Manufacturer code	Recommended
3	Model name	Recommended
4	Model number	Required
255	Vendor-specific extension field	
0 or 5-254	Reserved	

Figure 4.5: Correspondence of IDs and the device information to be stored

When TTC Subtype is "1", IDs are assigned to individual items of device information,

and the data corresponding to each ID is stored by using an TLV format. Figure 4.5 lists the correspondence of device information IDs and the device information to be store: device category , manufacturer code , model name , model number.

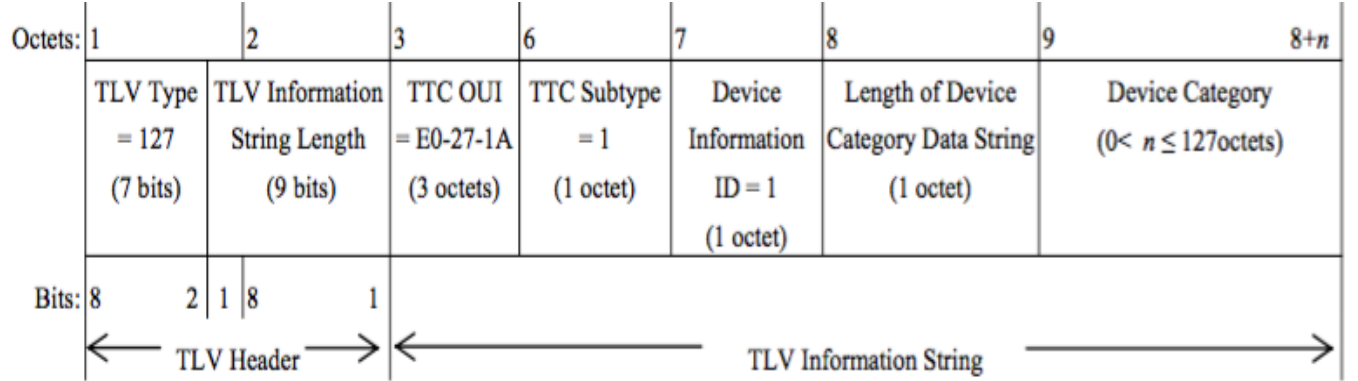


Figure 4.6: TLV to store the device category

Figure 4.6 describes the TLV structure to store device category, is encoded as follow:

```

char *htip_device_category = "Proxy_category\0";
    if (!(
        POKE_START_LLDP_TLV(LLDP_TLV_ORG) &&
        POKE_BYTES(htip, sizeof(htip)) && /*TTC OUT */
        POKE_UINT8(LLDP_TLV_HTIP_DEVICE) && /*TTC Subtype
        =1 */
        POKE_UINT8(LLDP_TLV_HTIP_DEVICE_CATEGORY) && /*
        device into ID = 1 */
        POKE_UINT8(strlen(htip_device_category)) &&
        POKE_BYTES(htip_device_category, strlen(
            htip_device_category)) &&
        POKE_END_LLDP_TLV))
        goto toobig;

```

and decoded as follow:

```

case LLDP_TLV_HTTP_DEVICE_CATEGORY:
    http_device_info_length = PEEK_UINT8;
    if ((http_device_info = (char *)calloc(1,
        http_device_info_length)) == NULL) {
        log_warn("http", "unable_to_allocate_space_for_
            device_info");
        goto malformed;
    } else {
        PEEK_BYTES(http_device_info,
            http_device_info_length);
        chassis->c_http_device_category =
            http_device_info;
    }
    break;

```

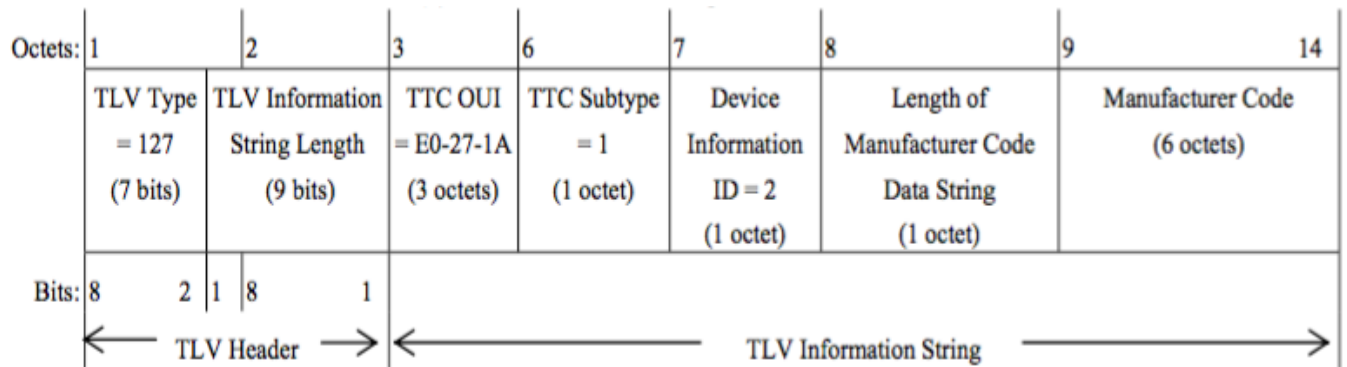


Figure 4.7: TLV to store the manufacturer code

Figure 4.7 describes the TLV structure to store manufacture code, is implemented as follow:

```

char *http_device_maker= "Proxy_manufacture_code\0";
    if (!(
        POKE_START_LLDP_TLV(LLDP_TLV_ORG) &&

```

```

POKEBYTES(htip, sizeof(htip)) && /*TTC OUT */
POKE_UINT8(LLDP_TLV_HTIP_DEVICE) && /*TTC Subtype
    =1 */
POKE_UINT8(LLDP_TLV_HTIP_DEVICE_CATEGORY) && /*
    device intoID = 2 */
POKE_UINT8(strlen(htip_device_maker)) &&
POKEBYTES(htip_device_maker, strlen(
    htip_device_maker)) &&
POKE_END_LLDP_TLV))
    goto toobig;

```

and decoded as follow:

```

case LLDP_TLV_HTIP_DEVICE_MAKER:
    htip_device_info_length = PEEK_UINT8;
    if ((htip_device_info = (char *)calloc(1,
        htip_device_info_length)) == NULL) {
        log_warn("htip", "unable_to_allocate_space_for_
            device_info");
        goto malformed;
    } else {
        PEEKBYTES(htip_device_info,
            htip_device_info_length);
        chassis->c_htip_device_maker = htip_device_info;
    }
    break;

```

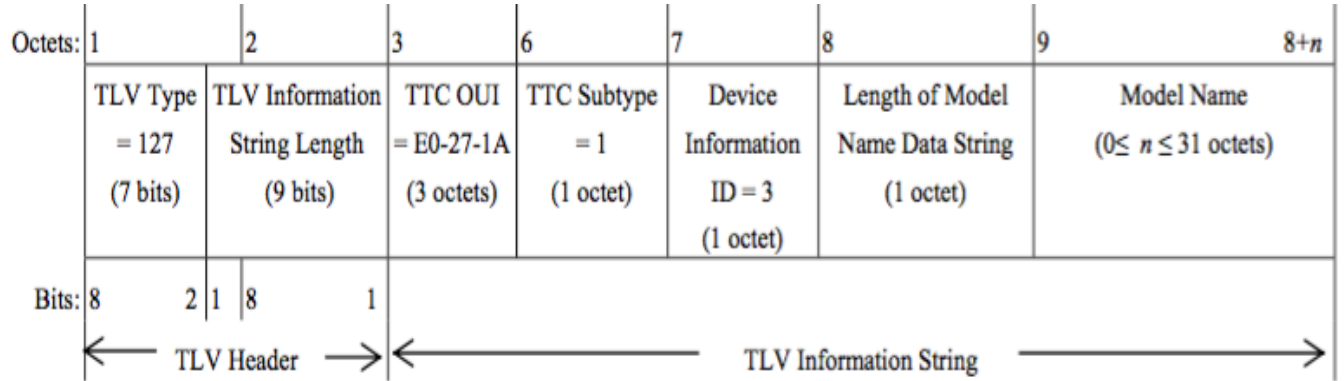



Figure 4.8: TLV to store the model name

Figure 4.8 describes the TLV structure to store model name, is implemented as follow:

```

char *htip_device_name = "Proxy_name\0";
    if (!(
        POKE_START_LLDP_TLV(LLDP_TLV_ORG) &&
        POKE_BYTES(htip, sizeof(htip)) && /*TTC OUT */
        POKE_UINT8(LLDP_TLV_HTIP_DEVICE) && /*TTC Subtype
        =3 */
        POKE_UINT8(LLDP_TLV_HTIP_DEVICE.CATEGORY) && /*
        device into ID = 1 */
        POKE_UINT8(strlen(htip_device_name)) &&
        POKE_BYTES(htip_device_name, strlen(
            htip_device_name)) &&
        POKE_END_LLDP_TLV))
        goto toobig;

```

and decoded as follow:

```

case LLDP_TLV_HTIP_DEVICE_NAME:
    htip_device_info_length = PEEK_UINT8;
    if ((htip_device_info = (char *)calloc(1,
        htip_device_info_length)) == NULL) {

```

```

        log_warn("htip", "unable to allocate space for
                device_info");
        goto malformed;
    } else {
        PEEK_BYTES(htip_device_info,
                htip_device_info_length);
        chassis->htip_device_name = htip_device_info;
    }
    break;

```

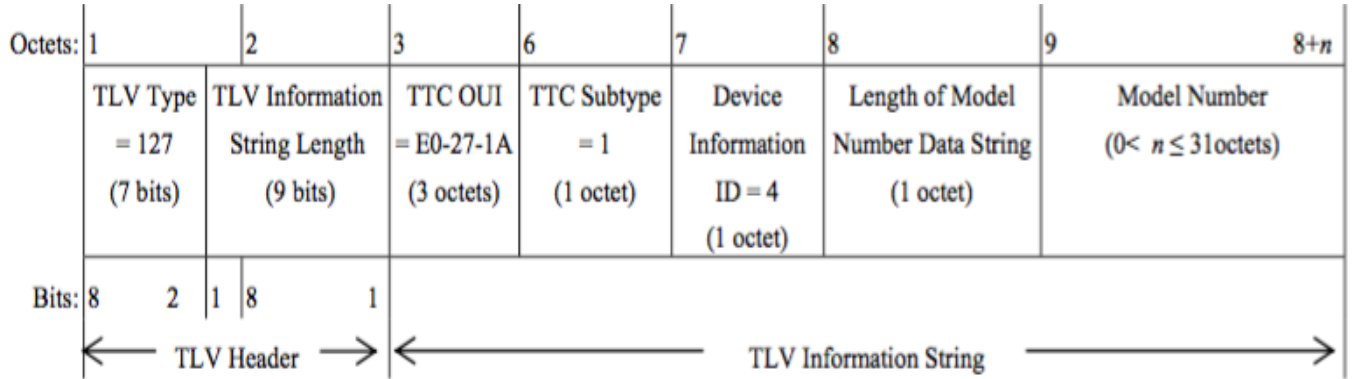


Figure 4.9: TLV to store the model number

Figure 4.9 describes the TLV structure to store device number, is implemented as follow:

```

char *htip_device_number = "Proxy_number\0";
if (!(
    POKE_START_LLDP_TLV(LLDP_TLV_ORG) &&
    POKE_BYTES(htip, sizeof(htip)) && /*TTC OUT */
    POKE_UINT8(LLDP_TLV_HTIP_DEVICE) && /*TTC Subtype
    =1 */
    POKE_UINT8(LLDP_TLV_HTIP_DEVICE.CATEGORY) && /*
    device into ID = 4 */
    POKE_UINT8(strlen(htip_device_number)) &&

```

```

        POKEBYTES(htip_device_number , strlen(
            htip_device_number) &&
        POKE_END_LLDP_TLV))
        goto toobig;

```

and decoded as follow:

```

case LLDP_TLV_HTTP_DEVICE_NUMBER:
    htip_device_info_length = PEEK_UINT8;
    if ((htip_device_info = (char *)calloc(1,
        htip_device_info_length)) == NULL) {
        log_warn("http", "unable_to_allocate_space_for_
            device_info");
        goto malformed;
    } else {
        PEEK_BYTES(htip_device_info ,
            htip_device_info_length);
        chassis->htip_device_number = htip_device_info;
    }
    break;

```

4.3 LLDP - ZigBee Proxy

In LLDP - ZigBee Proxy, there are 3 parts with 3 functions as described in section 3.5.

4.3.1 Part 1: ZigBee function

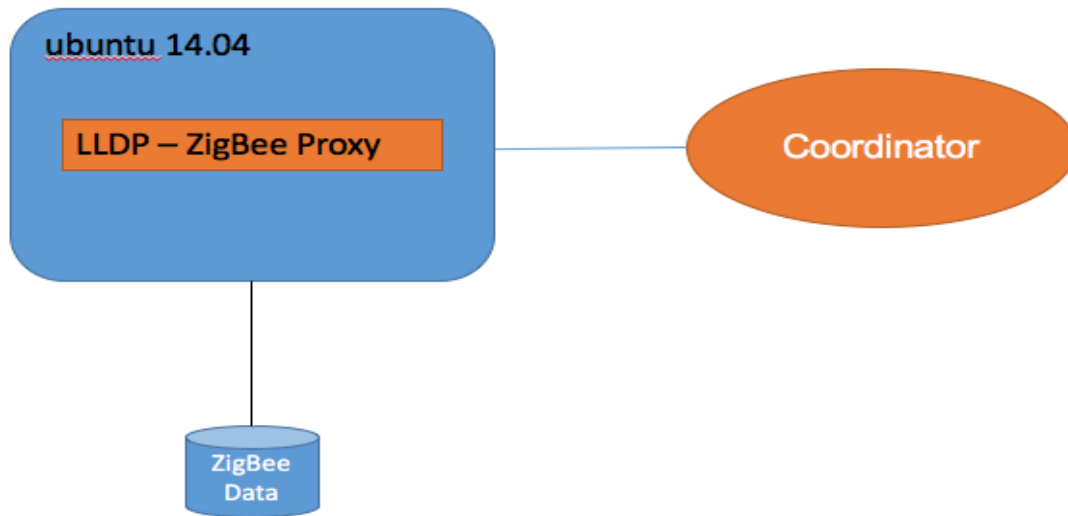


Figure 4.10: Connection between Coordinator - Proxy

The ZigBee Coordinator sends data to Proxy via UART, Proxy gets data and save to SQL database using SQLite3.

SQL database include 2 tables: ZigBee device information table and ZigBee neighbor table:

SID	Profile ID	Manufacture code	Device ID	Model Name	Network address	IEEE address
1	ProfileID1	Mcode1	DeviceID1	Jennic5139	Addr1	MAC1
2	ProfileID2	Mcode2	DeviceID2	Jennic5139	Addr2	MAC2
3	ProfileID3	Mcode3	DeviceID2	Jennic5139	Addr3	MAC3

Figure 4.11: ZigBee device information table

ID	Node MAC	Neighbor MAC	Relationship
1	nMAC1	bMAC1	Re1
2	nMAC2	bMAC2	Re2

Figure 4.12: ZigBee neighbor table

4.3.2 Part 2: Proxy function

This is the mapping descriptions of LLDP - ZigBee.

Type	LLDP Field name	ZigBee field name
Device Information	Device Type	Profile ID
	Manufacture code	Manufacturer Code
	Model Name	Jennic5139
	Model Number	Device ID
	MAC address	Network address + IEEE Address
Link Information	Interface	-
	Port	-
	MAC table	Neighbor table

Figure 4.13: Mapping descriptions between LLDP and ZigBee

Interoperating ZigBee with LLDP requires a device description mapping between two standards. In this respect, two standards are compared. Figure 4.13 shows a device description mapping table for making proxy LLDP and ZigBee. From this, we need to create a structure to transmit ZigBee data got from database into LLDP frame.

From that, I created 2 TLVs containing: ZigBee device information which has TTC Subtype = 4, ID = 1 and ZigBee neighbor information TTC Subtype = 4, ID = 2.

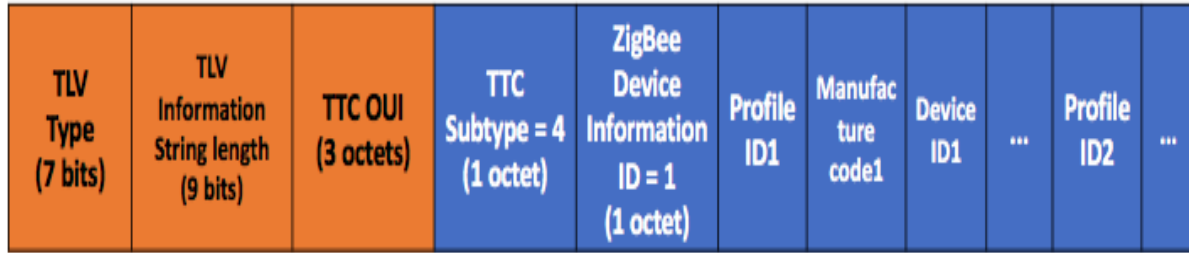


Figure 4.14: TLV structure to store ZigBee device information

Figure 4.14 describes the TLV structure to store ZigBee device information, is encoded as follow:

```

char buffer[512];
    DZDATA dzig_data;
    dzig_data = get_zigBee_device_data();
    sql2buffer(dzig_data,buffer);

    if (!(
        POKE_START_LLDP_TLV(LLDP_TLV_ORG) &&
        POKE_BYTES(htip, sizeof(htip)) && /*TTC OUT */
        POKE_UINT8(LLDP_TLV_HTIP_DEVICE) && /*TTC Subtype
            = 4 */
        POKE_UINT8(LLDP_TLV_HTIP_DEVICE_ZIGBEE) && /*
            device into ID = 5 */
        POKE_UINT8(strlen(buffer)) &&
        POKE_BYTES(buffer, strlen(buffer)) &&
        POKE_END_LLDP_TLV))
        goto toobig;

```

and decoded as follow:

```

case LLDP_TLV_HTIP_DEVICE_ZIGBEE:
    htip_device_info_length = PEEK_UINT8;

```

```

if ((http_device_info = (char *)calloc(1,
    http_device_info_length)) == NULL) {
    log_warn("http", "unable_to_allocate_space_for_
        device_info");
    goto malformed;
} else {
    PEEK_BYTES(http_device_info,
        http_device_info_length);
    chassis->buffer = http_device_info;
}
break;

```

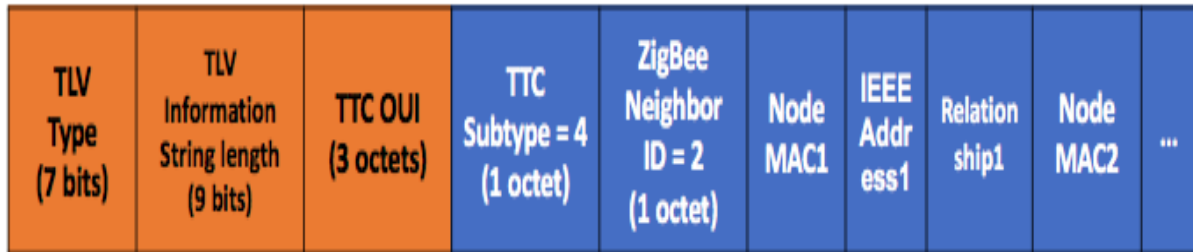


Figure 4.15: TLV structure to store ZigBee neighbor information

Figure 4.15 describes the TLV structure to store ZigBee neighbor information, is encoded as follow:

```

char buffer[512];
NZDATA nzig_data;
nzig_data = get_zigBee_neighbor_data();
sql2buffer(zig_data, buffer);

if (!(
    POKE_START_LLDP_TLV(LLDP_TLV_ORG) &&
    POKE_BYTES(http, sizeof(http)) && /*TTC OUT */

```

```

POKE_UINT8(LLDP_TLV_HTTP_DEVICE) && /*TTC Subtype
    = 4 */
POKE_UINT8(LLDP_TLV_HTTP_DEVICE_ZIGBEE) && /*
    device intoID = 5 */
POKE_UINT8(strlen(buffer)) &&
POKE_BYTES(buffer, strlen(buffer)) &&
POKE_END_LLDP_TLV))
    goto toobig;

```

and decoded as follow:

```

case LLDP_TLV_HTTP_NEIGHBOR_ZIGBEE:
    http_device_info_length = PEEK_UINT8;
    if ((http_device_info = (char *)calloc(1,
        http_device_info_length)) == NULL) {
        log_warn("http", "unable to allocate space for
            device_info");
        goto malformed;
    } else {
        PEEK_BYTES(http_device_info,
            http_device_info_length);
        chassis->buffer = http_device_info;
    }
    break;

```

4.3.3 Part 3: LLDP function

It will transmit the new TLVs frame to the LLDP manager as section 4.2.

4.4 UPnP

Using the open source libupnp-1.6.19 [15] .

Add some new elements in the Basic Device Information part in the DDD described in UDA and a policy for describing values in existing elements to transmit device information to Manager.

Regarding the device category, The List of Device Category must be referenced, and only the value of the category of the relevant device must be described in the "http:X-DeviceCategory" element. Multiple categories can be listed in the element.

Regarding the manufacturer code only the relevant company ID registered with the IEEE must be described in the "http:X-ManufacturerOUI" element². If the manufacturer of the installed device has not obtained a registered company ID, no value may be described in the "http:X-ManufacturerOUI" element. The "http:X-ManufacturerOUI" element must always be prepared even without a value described, and the manufacturer name must be described in the "http:X-ManufacturerOUI" element. Neither "http:X-DeviceCategory" nor "http:X-ManufacturerOUI" element is allowed to appear multiple times in the same DDD. [2]

Element to contain device information	Device information	Maximum length of string containable in element (octets)
http:X_DeviceCategory	(a) Device category	127
http:X_ManufacturerOUI	(b) Manufacturer code	6
modelName	(c) Model name	31
modelNumber	(d) Model number	31

Figure 4.16: Correspondence of device information and elements in UPnP

Figure 4.16 describes the new elements are added to DDD, by the code in below:

```
<http:X_DeviceCategory xmlns:http="http://www.ttc.or.jp/  
Home-network_WG/JJ-300.00">AV_TV</http:
```

```

X_DeviceCategory>
<http:X_ManufacturerOUI xmlns:http="http://ieee.ttc.or.jp/
Home-network_WG/JJ-300.00">00172E</http:X_ManufacturerOUI
>

```

4.5 HTIP manager

The HTIP manager is built as a HTTP server, it collects information from LLDP and UPnP protocols then create a XML file to draw the home network topology on the website.

Chapter 5

Result and Evaluation

5.1 Result of LLDP - ZigBee Proxy

When using command "lldpdcli show neighbors", it show the information of L2 Agent using LLDP protocol :

```
-<lldp label="LLDP neighbors">
- <interface label="Interface" name="eth0" via="LLDP" rid="1" age="0 day, 00:11:21">
  -<chassis label="Chassis">
    <id label="ChassisID" type="mac">00:1c:42:34:12:fb</id>
    <name label="SysName">ubuntu</name>
  -<descr label="SysDescr">
    Ubuntu 14.04.3 LTS Linux 3.19.0-25-generic #26~14.04.1-Ubuntu SMP Dec 24 21:18:00 UTC 2015 i686
  </descr>
  <http_category label="HttpDeviceCategory">proxy_category</http_category>
  <http_maker label="HttpDeviceMaker">proxy_manufacture_code</http_maker>
  <http_name label="HttpDeviceName">proxy_name</http_name>
  <http_number label="HttpDeviceNumber">proxy_number</http_number>
  <mgmt-ip label="MgmtIP">192.168.11.4</mgmt-ip>
  <capability label="Capability" type="Bridge" enabled="off"/>
  <capability label="Capability" type="Router" enabled="off"/>
  <capability label="Capability" type="Wlan" enabled="off"/>
  <capability label="Capability" type="Station" enabled="on"/>
</chassis>
- <port label="Port">
  <id label="PortID" type="mac">00:1c:42:64:9a:53</id>
  <descr label="PortDescr">eth0</descr>
</port>
- <port label="Port">
  <id label="PortID" type="mac">00:1c:42:61:bf:fb</id>
  <descr label="PortDescr">eth1</descr>
</port>
</interface>
</lldp>
```

Figure 5.1: Result of the showing LLDP devices

When using new command "lldpdcli show zigbee", it will show the information of ZigBee devices under LLDP protocol:

```
- <lldp label="ZigBee sensors">
- <http_zigbee_device label="HttpDeviceZigBee">
- <device_data>
  <zigbee_profileID label="ProfileID">0x291</zigbee_profileID>
  <zigbee_mcode label="Manufacture code">0x4149</zigbee_mcode>
  <zigbee_name label="Model name">Jennic 5139</zigbee_name>
  <zigbee_number label="Device number">0x0006</zigbee_number>
  <zigbee_address label="Network address">0x0000</zigbee_address>
  <zigbee_MAC label="MAC">0x00158d0000069580</zigbee_MAC>
</device_data>
- <device_data>
  <zigbee_profileID label="ProfileID">0x291</zigbee_profileID>
  <zigbee_mcode label="Manufacture code">0x4151</zigbee_mcode>
  <zigbee_name label="Model name">Jennic 5139</zigbee_name>
  <zigbee_number label="Device number">0x0000</zigbee_number>
  <zigbee_address label="Network address">0x0001</zigbee_address>
  <zigbee_MAC label="MAC">0x00158d0000069907</zigbee_MAC>
</device_data>
- <device_data>
  <zigbee_profileID label="ProfileID">0x291</zigbee_profileID>
  <zigbee_mcode label="Manufacture code">0x4154</zigbee_mcode>
  <zigbee_name label="Model name">Jennic 5139</zigbee_name>
  <zigbee_number label="Device number">0x0010</zigbee_number>
  <zigbee_address label="Network address">0x0002</zigbee_address>
  <zigbee_MAC label="MAC">0x00158d0000069a46</zigbee_MAC>
</device_data>
</http_zigbee_device>
- <http_neighbor_zigbee label="HttpNeighborZigBee">
- <neighbor_data>
  <neighbor_ID label="SID">1</neighbor_ID>
  <neighbor_MAC label="Neighbor MAC">0x00158d0000069580</neighbor_MAC>
  <neighbor_rel label="Neighbor relationship">0x00</neighbor_rel>
  <neighbor_ID label="SID">2</neighbor_ID>
  <neighbor_MAC label="Neighbor MAC">0x00158d0000069a46</neighbor_MAC>
  <neighbor_rel label="Neighbor relationship">0x01</neighbor_rel>
</neighbor_data>
- <neighbor_data>
  <neighbor_ID label="SID">1</neighbor_ID>
  <neighbor_MAC label="Neighbor MAC">0x00158d0000069907</neighbor_MAC>
  <neighbor_rel label="Neighbor relationship">0x00</neighbor_rel>
</neighbor_data>
</http_neighbor_zigbee>
</lldp>
```

Figure 5.2: Result of the showing ZigBee devices

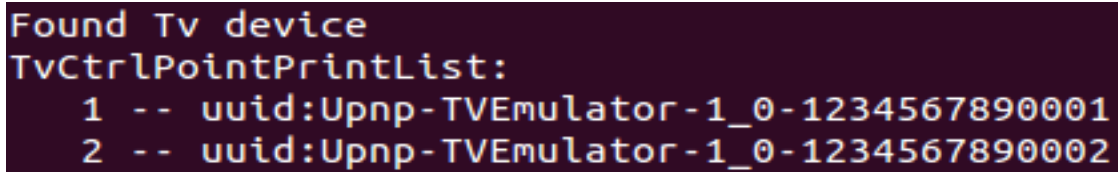
As the result shows, the proxy can detect and transmit information of ZigBee system.

The manager can use these data to draw a topology of ZigBee system through LLDP protocol.

ZigBee network includes 3 sensors: 1 Coordinator with Network address = 0x00, 1 Router with Network address = 0x01 and 1 Device with Network address = 0x02.

5.2 UPnP devices

There are 2 UPnP Devices, TV1 and TV2:



```
Found Tv device
TvCtrlPointPrintList:
  1 -- uuid:Upnp-TVEulator-1_0-1234567890001
  2 -- uuid:Upnp-TVEulator-1_0-1234567890002
```

Figure 5.3: UPnP device

Information of TV1:

```
TvCtrlPointPrintDevice:
TvDevice -- 1
|
+- UDN          = uuid:Upnp-TVEulator-1_0-1234567890001
+- DescDocURL   = http://192.168.11.12:49152/tvdevicedesc.xml
+- FriendlyName = UPnP Television Emulator
+- PresURL      = http://192.168.11.12:49152/tvdevicepres.html
+- Adver. Timeout = 100
+- HTIP DeviceCategory = AV_TV
+- HTIP Manufacturer OUI = 00172E
|
+- Tv Control Service
|   +- ServiceId      = urn:upnp-org:serviceId:tvcontrol1
|   +- ServiceType    = urn:schemas-upnp-org:service:tvcontrol:1
|   +- EventURL       = http://192.168.11.12:49152/upnp/event/tvcontrol1
|   +- ControlURL     = http://192.168.11.12:49152/upnp/control/tvcontrol1
|   +- SID            = uuid:9f64da70-1dd1-11b2-a702-9ca8c291d4ff
|   +- ServiceStateTable
|       +- Power      = 1
|       +- Channel    = 1
|       +- Volume     = 5
|
+- Tv Picture Service
|   +- ServiceId      = urn:upnp-org:serviceId:tvpicture1
|   +- ServiceType    = urn:schemas-upnp-org:service:tvpicture:1
|   +- EventURL       = http://192.168.11.12:49152/upnp/event/tvpicture1
|   +- ControlURL     = http://192.168.11.12:49152/upnp/control/tvpicture1
|   +- SID            = uuid:9f650018-1dd1-11b2-a702-9ca8c291d4ff
|   +- ServiceStateTable
|       +- Color      = 5
|       +- Tint       = 5
|       +- Contrast   = 5
|       +- Brightness = 5
```

Figure 5.4: UPnP device - TV 1

Information of TV2:

```
TvCtrlPointPrintDevice:
  TvDevice -- 2
  |
  +- UDN          = uuid:Upnp-TVEulator-1_0-1234567890002
  +- DescDocURL   = http://192.168.11.11:49152/tvdevicedesc.xml
  +- FriendlyName = UPnP Television Emulator
  +- PresURL      = http://192.168.11.11:49152/tvdevicepres.html
  +- Adver. TimeOut = 100
  +- HTIP DeviceCategory = AV_TV
  +- HTIP Manufacturer OUI = 00172E
  |
  +- Tv Control Service
  |   +- ServiceId      = urn:upnp-org:serviceId:tvcontrol1
  |   +- ServiceType    = urn:schemas-upnp-org:service:tvcontrol:1
  |   +- EventURL       = http://192.168.11.11:49152/upnp/event/tvcontrol1
  |   +- ControlURL     = http://192.168.11.11:49152/upnp/control/tvcontrol1
  |   +- SID            = uuid:73e729d6-1dd2-11b2-9ed3-b851bfab7c82
  |   +- ServiceStateTable
  |       +- Power      = 1
  |       +- Channel    = 1
  |       +- Volume     = 5
  |
  +- Tv Picture Service
  |   +- ServiceId      = urn:upnp-org:serviceId:tvpicture1
  |   +- ServiceType    = urn:schemas-upnp-org:service:tvpicture:1
  |   +- EventURL       = http://192.168.11.11:49152/upnp/event/tvpicture1
  |   +- ControlURL     = http://192.168.11.11:49152/upnp/control/tvpicture1
  |   +- SID            = uuid:73e75ff0-1dd2-11b2-9ed3-b851bfab7c82
  |   +- ServiceStateTable
  |       +- Color      = 5
  |       +- Tint       = 5
  |       +- Contrast   = 5
  |       +- Brightness = 5
```

Figure 5.5: UPnP device - TV 2

5.3 HTIP manager

The HTIP manager is a HTTP server, it collects information from LLDP and UPnP protocol, then create 2 XML files: Devices.xml and Topology.xml to draw the home network topology.

```
-<DeviceList>
-  <Device>
    <device_category>proxy_category</device_category>
    <device_manufacture>proxy_manufacture_code</device_manufacture>
    <model_name>proxy_name</model_name>
    <model_number>proxy_number</model_number>
  -<Network>
    <IP>192.168.11.4</IP>
    <MAC>00:1c:42:34:12:fb</MAC>
  </Network>
  -<ZigBee>
    -<device_data>
      <zigbee_profileID>0x291</zigbee_profileID>
      <zigbee_mcode>0x4149</zigbee_mcode>
      <zigbee_name>Jennic 5139</zigbee_name>
      <zigbee_number>0x0006</zigbee_number>
      <zigbee_address>0x0000</zigbee_address>
      <zigbee_MAC>00:15:8d:00:00:06:95:80</zigbee_MAC>
    </device_data>
    -<device_data>
      <zigbee_profileID>0x291</zigbee_profileID>
      <zigbee_mcode>0x4151</zigbee_mcode>
      <zigbee_name>Jennic 5139</zigbee_name>
      <zigbee_number>0x0000</zigbee_number>
      <zigbee_address>0x0001</zigbee_address>
      <zigbee_MAC>00:15:8d:00:00:06:99:07</zigbee_MAC>
    </device_data>
    -<device_data>
      <zigbee_profileID>0x291</zigbee_profileID>
      <zigbee_mcode>0x4154</zigbee_mcode>
      <zigbee_name>Jennic 5139</zigbee_name>
      <zigbee_number>0x0010</zigbee_number>
      <zigbee_address>0x0002</zigbee_address>
      <zigbee_MAC>00:15:8d:00:00:06:9a:46</zigbee_MAC>
    </device_data>
  </ZigBee>
</Device>
-  <Device>
    <device_category>HTIP_TV_emulator</device_category>
    <device_manufacture>HTIP_TV_Manufacture</device_manufacture>
    <model_name>TVEmulator1</model_name>
    <model_number>1.0</model_number>
  -<Network>
    <IP>192.168.11.12</IP>
    <MAC>00:1c:42:89:74:4f</MAC>
  </Network>
</Device>
-  <Device>
    <device_category>HTIP_TV_emulator</device_category>
    <device_manufacture>HTIP_TV_Manufacture</device_manufacture>
    <model_name>TVEmulator2</model_name>
    <model_number>1.0</model_number>
  -<Network>
    <IP>192.168.11.11</IP>
    <MAC>00:1c:42:61:bf:fb</MAC>
  </Network>
</Device>
</DeviceList>
```

Figure 5.6: Devices information

Device list has information of all devices in the network.

```

- <Topology>
- <port label="Port">
  <interface label="Interface">ethernetCsmacd</interface>
  <descr label="PortDescr">eth0</descr>
  <id label="PortID" type="mac">00:1c:42:34:12:fb</id>
- <port>
  <interface label="Interface">ethernetCsmacd</interface>
  <descr label="PortDescr">eth0</descr>
  <id label="PortID" type="mac">00:1c:42:64:9a:53</id>
</port>
- <port>
  <interface label="Interface">ethernetCsmacd</interface>
  <descr label="PortDescr">eth1</descr>
  <id label="PortID" type="mac">00:1c:42:61:bf:fb</id>
</port>
- <port>
  <interface label="Interface">serialPort</interface>
  <descr label="PortDescr">ttyS0</descr>
- <zigbee label="Neighbors">
  - <node>
    <node_MAC>00:15:8d:00:00:06:99:07</node_MAC>
    <neighbor_MAC>00:15:8d:00:00:06:95:80</neighbor_MAC>
    <neighbor_rel>Parent</neighbor_rel>
    <neighbor_MAC>00:15:8d:00:00:06:9a:46</neighbor_MAC>
    <neighbor_rel>Child</neighbor_rel>
  </node>
  - <node>
    <node_MAC>00:15:8d:00:00:06:9a:46</node_MAC>
    <neighbor_MAC>00:15:8d:00:00:06:99:07</neighbor_MAC>
    <neighbor_rel>Parent</neighbor_rel>
  </node>
</zigbee>
</port>
- <port>
  <interface label="Interface">ethernetCsmacd</interface>
  <descr label="PortDescr">eth1</descr>
  <id label="PortID" type="mac">00:1c:42:89:74:4f</id>
</port>
</Topology>

```

Figure 5.7: Link information

Topology has information of all port connected to the manager.

From these result, the HTIP manager can draw a general view of the home network topology.

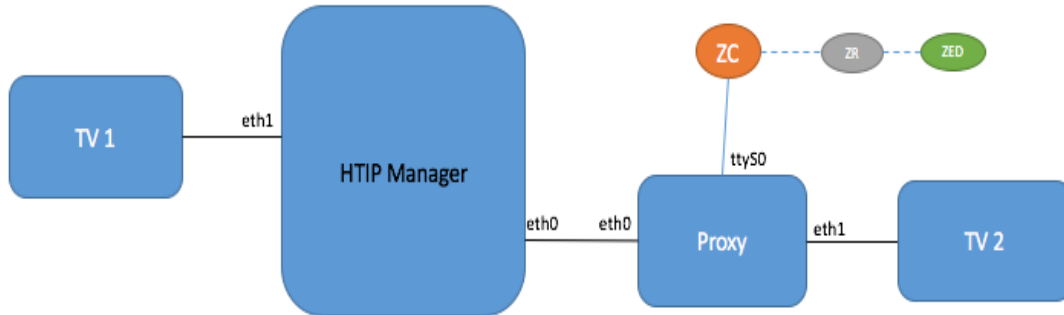


Figure 5.8: Network Topology

With the evaluation of time and size calculating on Virtual system:

Case	HTIP LLDP frame (bytes)	LLDP-ZigBee frame (bytes)	Execution time (s)	Accuracy rate
1 ZC + 1 ZR + 1 ZED	175	321	0.000356	100%
1 ZC + 1 ZR	175	283	0.000312	100%

Figure 5.9: Evaluation

Chapter 6

Conclusion and Future research

The HTIP technology are an important enabler for seamless networking and data sharing in digital homes. And ZigBee is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power. With the increasing popularity of sensors, demands for combining HTIP and ZigBee are required.

In this paper, to combine 2 different standards, HTIP and ZigBee, a LLDP-ZigBee Proxy is proposed in a designed home network. To represent ZigBee information as LLDP information, the proposed Proxy with the new proposed frame structure employs ZigBee data, mapping the differences to transfer it into LLDP frame, then ZigBee information will be sent under LLDP information. So that, the HTIP manager can identify ZigBee devices through LLDP protocol, and draw a general home network topology. From the result of implementation, it is shown that the proposed solution works correctly.

The designed network is on a small scale, with only some devices to test the proposed solution. In the future, with the very fast increasing popularity of sensors, the home network will include a lots of devices with many different protocols. Base on this proposed Proxy, we can apply it to combine other protocols, to make these technologies more suitable for humanity.

This paper focuses only in the home network, the outer side, from the Internet to Home gateway is out of scope. The other management protocol such as TR-069[8] , OMA-LW [9] , ... can be applied to manage home network from far away.

Bibliography

- [1] ESTI, “Machine-to-Machine communications (M2M): Functional architecture”, *European Telecommunications Standards Institute 2013*.
- [2] TTC JJ-300.00, “HTIP: Home network Topology Identifying Protocol”, *Aug. 2010*
- [3] TTC JJ-300.01, “The List of Device Category”, *Nov. 2010*
- [4] ISO/IEC 29341-1, “Information technology UPnP Device Architecture Part 1: UPnP Device Architecture Version 1.0”,
- [5] ISO/IEC 29341-2, “Information technology UPnP Device Architecture Part 2: Basic Device Control Protocol Basic Device”, *Edition 1.0, December 2008*.
- [6] IEEE Computer Society, “802.1AB-2009: Local and Metropolitan Area Networks Station and Media Access Control Connectivity Discovery” (LLDP), *Sep. 2009*
- [7] Internet Assigned Numbers Authority (IANA), “IANAifType-MIB DEFINITIONS” ver. 200905060000Z, *May 2009*
- [8] Broadband Forum, “TR-069, CPE WAN Management Protocol”, ver.1.1, *2007*
- [9] Open Mobile Alliance, “OMA-TS-LightweightM2M”, ver 1.0 *2007*
- [10] ZigBee Alliance, “ZigBee Document 053474r06”, Version 1.0 *December 14th, 2004*
- [11] NXP, “ZigBee Stack User Guide”, *June, 2014*
- [12] Miyamoto Takahiro, “A remote setting mechanism for network enabled devices in the home environment” *March, 2015*

- [13] Yoshiyuki Mihara, Takefumi Yamazaki, Akama Takehiro, “Designing HTIP: Home Network Topology Identifying Protocol”, 2011 IEEE International Conference p1 - 6
June 2011
- [14] <https://vincentbernat.github.io/lldpd/>
- [15] <http://pupnp.sourceforge.net/>