國立暨南國際大學資訊工程學系

碩士論文

應用於智慧電網管理之通訊協定 **SIP/SNMP** 效能量測研究

On Network Performance Evaluation toward the Smart Grid: A Case Study of SIP and SNMP

指導教授：吳坤熹博士

研究生：鐘揮雄

中華民國一〇一年七月二十三日

國立暨南國際大學資訊工程學系

碩士論文

應用於智慧電網管理之通訊協定 **SIP/SNMP** 效能量測研究

On Network Performance Evaluation toward the Smart Grid: A Case Study of SIP and SNMP

指導教授：吳坤熹博士

研究生：鐘揮雄

中華民國一〇一年七月二十三日

國立暨南國際大學碩士論文考試審定書

資訊工程 學系（研究所）

研究生 鐘揮雄 所提之論文

應用於智慧電網管理之通訊協定SIP/SNMP效能量測研究

On Network Performance Evaluation toward the Smart Grid: A Case Study of SIP and SNMP

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

[illegible] 委員兼召集人

[illegible] 委員

吳坤熹 委員

中華民國 一〇一 年 七 月 二十三 日

致謝

在我就讀國立暨南國際大學資工系的這段期間內，吳坤熹老師是我最尊敬，也是最感謝的一位老師。 由大三開始的專題指導教授一直到碩士班為止，吳坤熹老師不單單教育我學術研究上的知識，也教育了我該如何用正確的心態去面對生活中的事物。大學時的我，不僅僅對於學習開始找不到方向，也漸漸的開始對學習失去了信心以及興趣。 但是研究所的生活中，我透過學習得到了成就，也找回了對學習的熱誠。 在大學三年級至碩士班畢業的這段時間內，我衷心的感謝吳坤熹老師對我的悉心指導。

感謝實驗室的學長、同學以及學弟妹們，不僅僅在我研究有問題時可以互相討論，一起聚餐時更是可以分享工作經驗以及各種搞笑的事蹟。 感謝意姍學妹擔負起了實驗室便當小精靈的重要角色，讓實驗室的大家常常有好東西可吃！

最後我要感謝我的家人，我的父母提供我一個沒有憂慮的成長環境，也讓我無後顧之憂地完成我的學業。 感謝我的姊姊以及哥哥，在我曾經困難時對我的理解以及支持。

感謝曾經幫助過我的所有人，我衷心的感謝你們。

論文名稱：應用於智慧電網管理之通訊協定 SIP/SNMP 之效能量測研究

校院系：國立暨南國際大學科技學院資訊工程學系　　　　　　頁數：49

畢業時間：中華民國一○一年七月　　　　　　學位別：碩士

研究生：鐘揮雄　　　　　　指導教授：吳坤熹博士

摘要

隨著能量消耗的議題日趨受到重視，智慧電網已開始在日常生活中被應用。然而在現今智慧電網的建設中，卻缺乏通用的管理機制，也致使不同廠商所開發的智慧電網設備無法互相通訊。Session Initiation Protocol（SIP）過去為一個被廣泛使用在網路電話的通訊協定；但在本篇研究中，利用 SIP 控制網路家電的特性，作為智慧電網管理的一套通用機制。另一個被廣泛使用於網路管理的通訊協定 Simple Network Management Protocol（SNMP），也適合作為智能電網管理之協定。本論文將兩協定分別移植至嵌入式系統，並比較其運作效能。初步結論為 SIP 的速度較快，但所佔記憶體空間較大。未來可依不同設備之特性，選用合適之管理協定。

關鍵詞：SIP；SNMP；智慧電網

Title of Thesis: On Network Performance Evaluation toward the Smart Grid: A Case Study of SIP and SNMP

Name of Institute: Department of Computer Science and Information Engineering, College of Science and Technology, National Chi Nan University Pages: 49

Graduation Time: July 2012 Degree Conferred: Master

Student Name: Hui-Hsiung Chung Advisor Name: Dr. Quincy Wu

Abstract

As the energy consumption issue is getting critical, in many countries Smart Grid has become widely adopted in daily life. However, one major obstacle of the development of Smart Grid is that, the products from different vendors cannot communicate with each other easily, due to the absence of a general managing mechanism. Session Initiation Protocol (SIP) has been widely used in Voice over Internet Protocol (VoIP). However, it was also adopted as a protocol to control network appliances. In this thesis, we utilize SIP as a common mechanism to support Smart Grid management. Meanwhile, another candidate is the Simple Network Management Protocol, which is also a popular protocol in managing IP devices. In this work, both protocols were implemented on a real embedded platform, and the performance is further evaluated and compared. The experimental results showed that the SIP protocol stack occupies more memory space, but its response time is shorter. In choosing which management protocol to adopt, device manufacturers can make the technical choices according to whether they need to save the memory space or the response time.

Key words: SIP; SNMP; Smart Grid

Contents

List of Figures

List of Tables

Chapter 1. Introduction

It is believed that in the twenty-first century, there will be over seven billion people on earth. Inevitably, people will face the problem of living with scarce resources. With the finite resources, we need to utilize these resources efficiently. Energy, the resource supporting the modern industry and our everyday lives, is a limited resource that human being must carefully manage and utilize, to ensure the sustainability of our society. In 2005, Smart Grid was brought up as an emerging cyber-physical system [1], whose idea is shown in Figure 1.

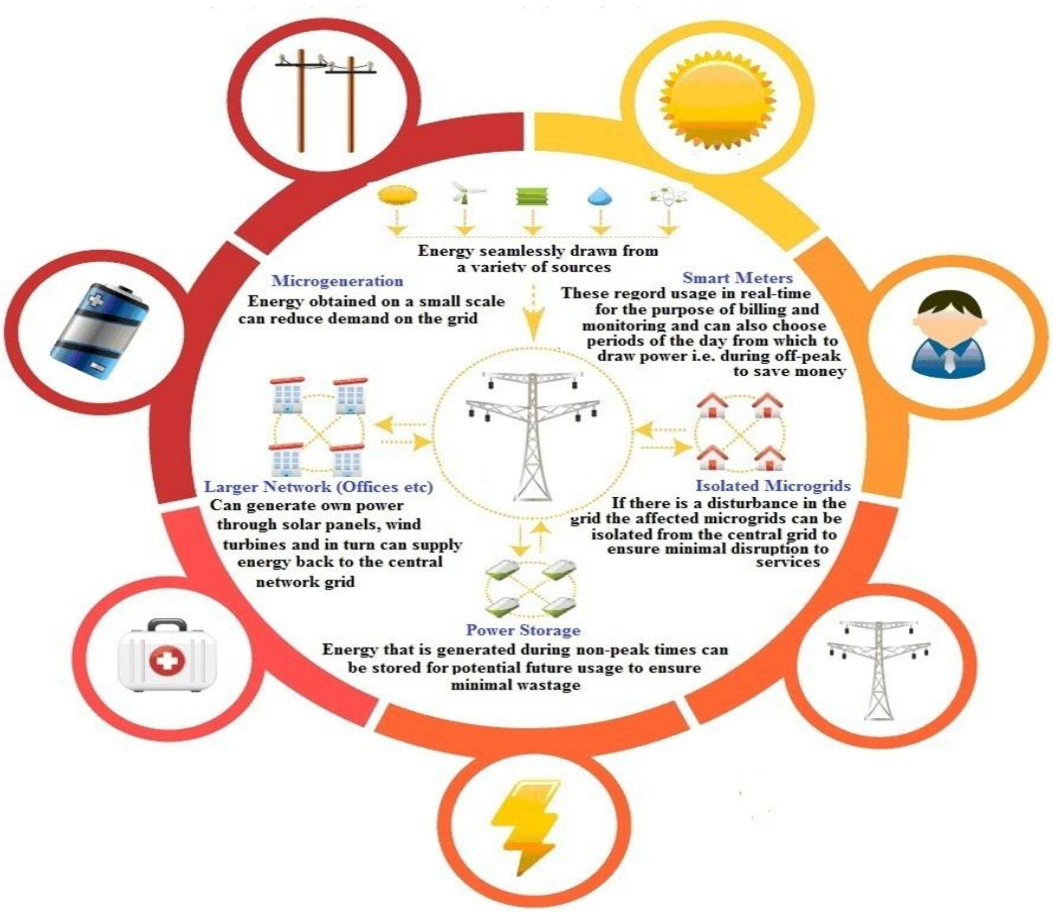

Energy seamlessly drawn from a varietv of sources
Microgeneration
Energy obtained on a small scale can reduce demand on the grid
Smart Meters
These regord usage in real-time for the purpose of billing and monitoring and can also choose periods of the day from which to draw power i.e. during off-peak to save money
Larger Network (Offices etc)
Can generate own power through solar panels, wind turbines and in turn can supply energy back to the central network grid
Isolated Microgrids
If there is a disturbance in the grid the affected microgrids can be isolated from the central grid to ensure minimal disruption to services
Power Storage
Energy that is generated during non-peak times can be stored for potential future usage to ensure minimal wastage

Figure 1 Smart Grid [2]

Smart Grid delivers electricity from suppliers to consumers. It incorporates power infrastructures with two-way communication technologies [3]. Smart Grid combines with power systems, telecommunications, smart energy devices and digital controllers [4]. As shown in Figure 1, Smart Grid integrates different technologies into a universal system. With Smart Grid, people will be able to control electric appliances remotely. Furthermore, the information such as temperature, humidity and power consumption, can be collected by smart devices and delivered to administrators by Smart Grid Networks. Therefore, even when people are far away, they can keep aware of every detail of their houses. Smart Grid not only helps people to live in a smarter style, but also helps people to get more efficiency in electric power utilization. In many countries, there is a policy of Critical Peak Pricing (CPP), which charges higher price for electricity in peak hours. With the assistance of Smart Grid, it is possible to collect the information of power usage by smart devices, figure out the tendency of power consumption in a house, and remind people to turn off unnecessary electricity appliances in peak hours. Some tasks can be re-arranged to non-peak hours; this helps people to save cost and enjoy the lower price in non-peak hours. Naturally, in order to use power more efficiently, the information needs to be delivered in time to support real-time monitoring. To enable such mechanisms, the delay performances of messaging deliveries must satisfy their timing requirements, such as 3ms for relay protections and 16ms for data monitoring in substations [5]. Thus, the communication performance is of critical importance in Smart Grid.

Nowadays, power transmission systems suffer from the fact that intelligence is only applied locally by protection systems and by central control through the supervisory control and data acquisition (SCADA) [6] system, such as distributed network protocol 3.0 (DNP3) [7] and Modbus [8]. The SCADA protocols were originally designed over serial links, so these protocols are not appropriate to be directly applied to the Internet, which has a comparably higher data transmission speed. In scenarios where security may not be a critical issue, it is possible to allow consumers and suppliers communicating with each other through the Internet. According to the above assumption, this thesis designs a Microgrid [9] for demonstration as shown in Figure 2. User agents can use mobile devices to control end devices. End devices send back information to user agents or suppliers for further analysis.

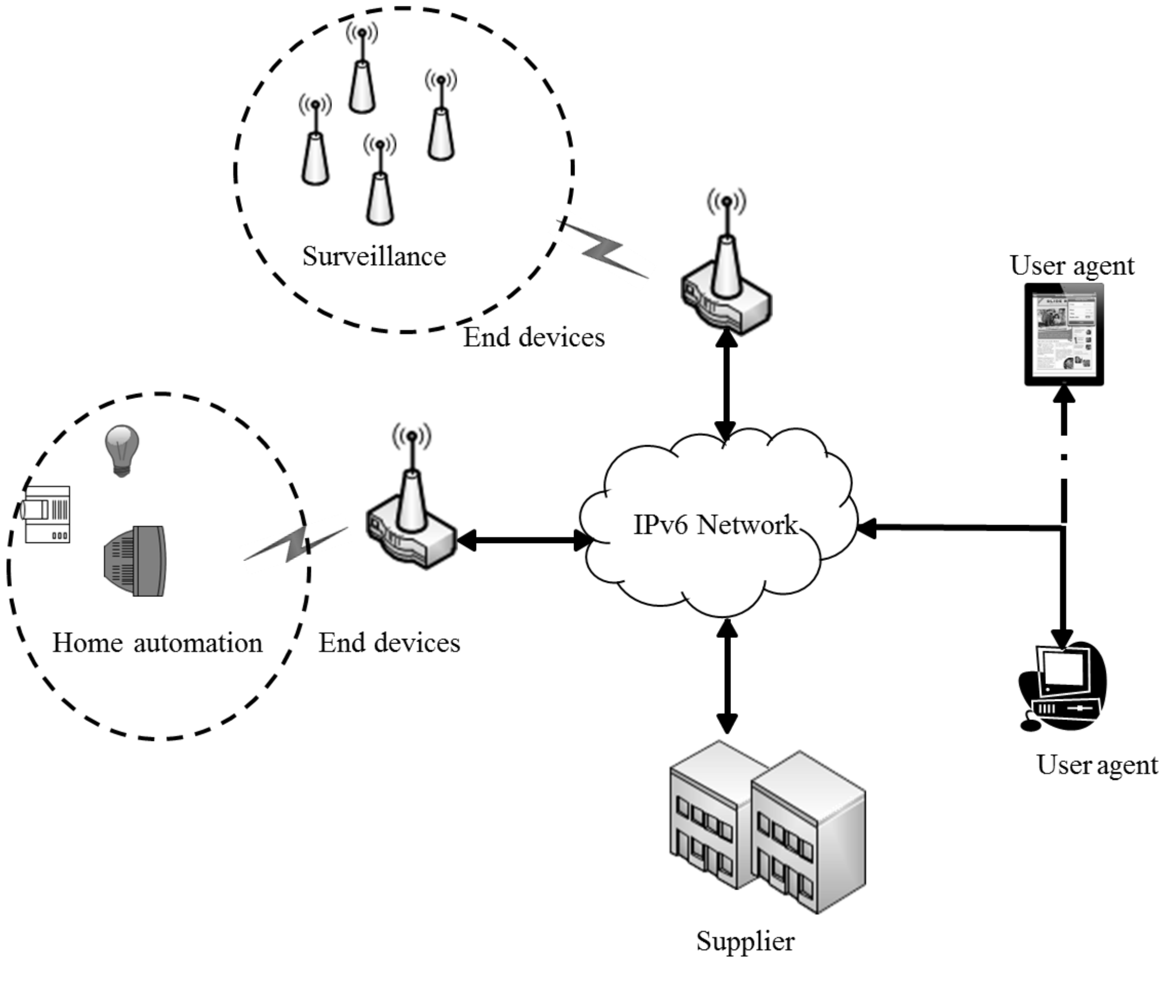


Figure 2 Microgrid

Despite of the prosperity of Smart Grid, there lacks a general management mechanism. Architectures of Smart Grid are often structurally complex and consist of several hierarchies, which consist of a variety of physical communication media used for different functions [10], supported by different protocols as shown in Figure 3. Therefore, most managing interfaces in Smart Grid are proprietary.

As Smart Grid becomes an important infrastructure, obviously an efficient and general solution for management is needed. In this thesis, we explore Simple Network Management Protocol (SNMP) [11] and Session Initiation Protocol (SIP) [12] as general management mechanisms in Smart Grid. SNMP is an Internet-standard protocol proposed in 1990 for managing devices on IP networks. Compared with SNMP, SIP is a newbie in network management. However, SIP is foreseen as a clear choice for Smart Grid communication [13] in recent years.

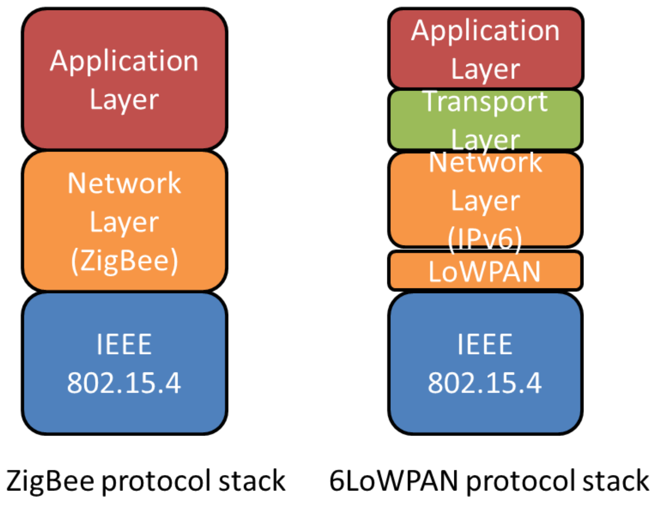


Figure 3 Common protocol stacks in Smart Grid

As more devices are connected to the Internet, the Internet Protocol (IP) address space is facing the crisis of depletion. To mitigate the problem of IP address shortage, Network Address Translation (NAT) [14] was proposed. As many NAT devices deployed to retard the exhaustion of IP addresses, there are a lot of enterprise/home networks applying NAT to access the Internet. Unfortunately, because SNMP has no registration mechanism, NAT becomes an obstacle for SNMP in remote management [15]. As shown in Figure 4, SNMP requests will be blocked by the NAT. When users have devices deployed behind a NAT, it will become unreachable for remote management. On the contrary, SIP provides a registration mechanism. As shown in Figure 5, an end device registers to a SIP proxy server when it is activated. The NAT maps the private IP address and port number (10.21.11.33:5060) of the REGISTER message to a public IP address and port number (163.22.21.83:3456) when it passes through the NAT. Then, for any SIP agent that wants to send a request to this end device, the request is sent to the SIP proxy server and forwarded to 163.22.21.83:3456, which will be translated to 10.21.11.33:5060 by the NAT and routed to the end device successfully. Therefore, with this registration mechanism, a manager can send messages to end devices via the SIP proxy server easily.

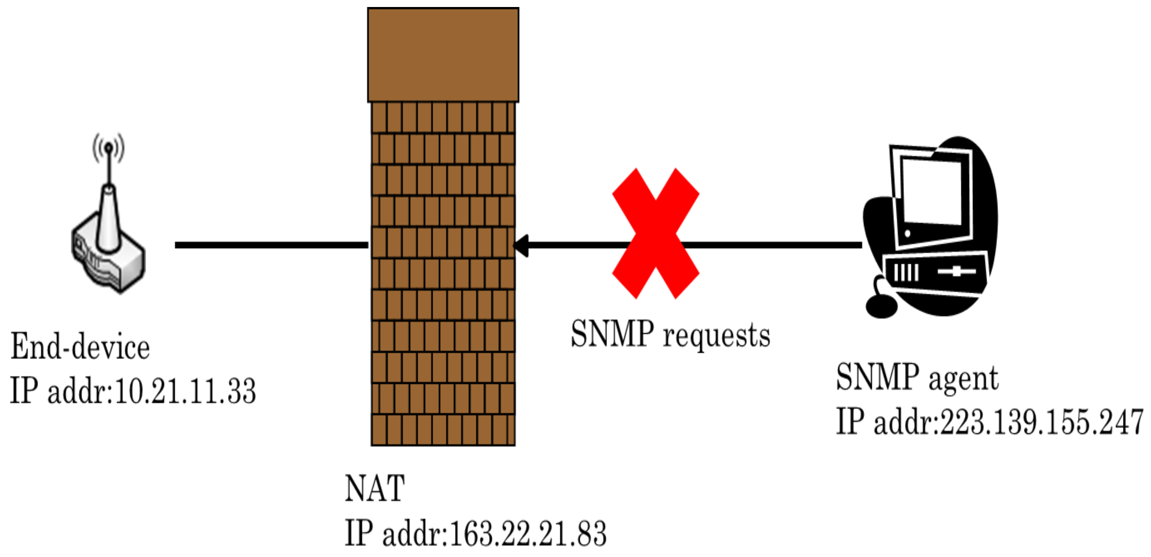


Figure 4 NAT traversal in SNMP

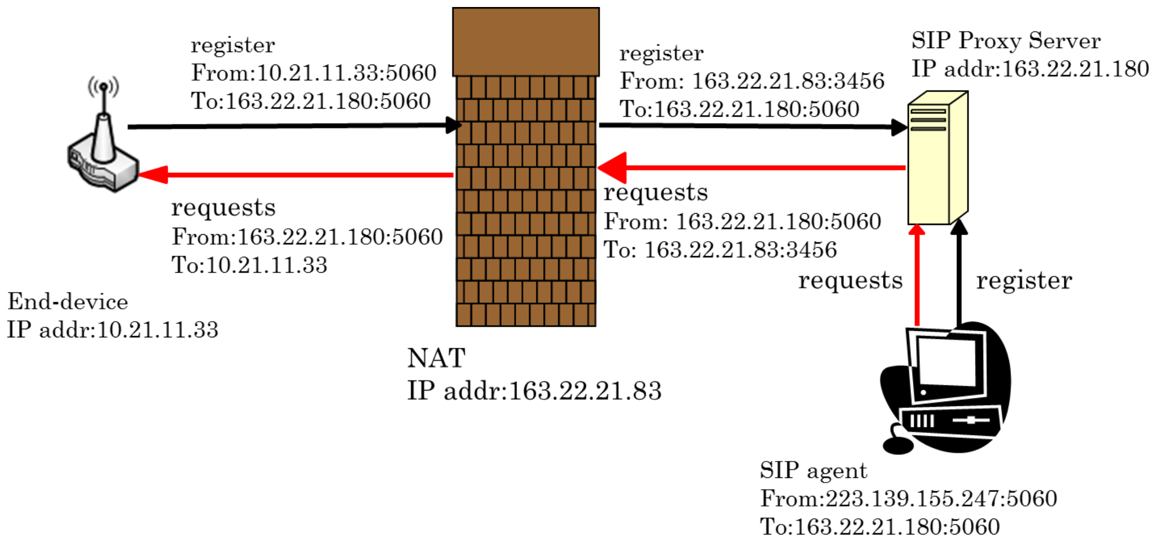


Figure 5 NAT traversal in SIP

Even though SNMP has a long history in traditional network management, Smart Grid management is a new application for both SIP and SNMP. Evaluating advantages and disadvantages of these two management mechanisms in Smart Grid would help to develop better Smart Grid management systems. The remaining of this thesis is organized as follows. Chapter 2 briefly introduced SNMP and SIP protocols. Chapter 3 describes the system architecture and the components in our system. Chapter 4 illustrates the implementation details and performance evaluation. Conclusive remarks and some future works are given in Chapter 5.

Chapter 2. Related Works

2.1 SNMP in Home Automation

Smart Grid allows administrators to turn on/off home appliances remotely, which is part of the scenario in “Home Automation”. In [16], SNMP was used as a management mechanism in home automation. In that paper, the authors deployed several SNMP agents in a house. Then, they set up a SNMP manager, which communicates to SNMP agents by wireless network as shown in Figure 6. Users can control smart devices with SNMP requests and monitor the information that was collected by SNMP agents. The size of an SNMP request is roughly between 100 and 548 bytes. With small packet size, SNMP is an efficient choice for home automation. However, as Figure 6shows, the SNMP manager and smart devices must be in the same Local Area Network (LAN). SNMP messages will be blocked by NAT if the SNMP manager tries to send SNMP messages remotely.

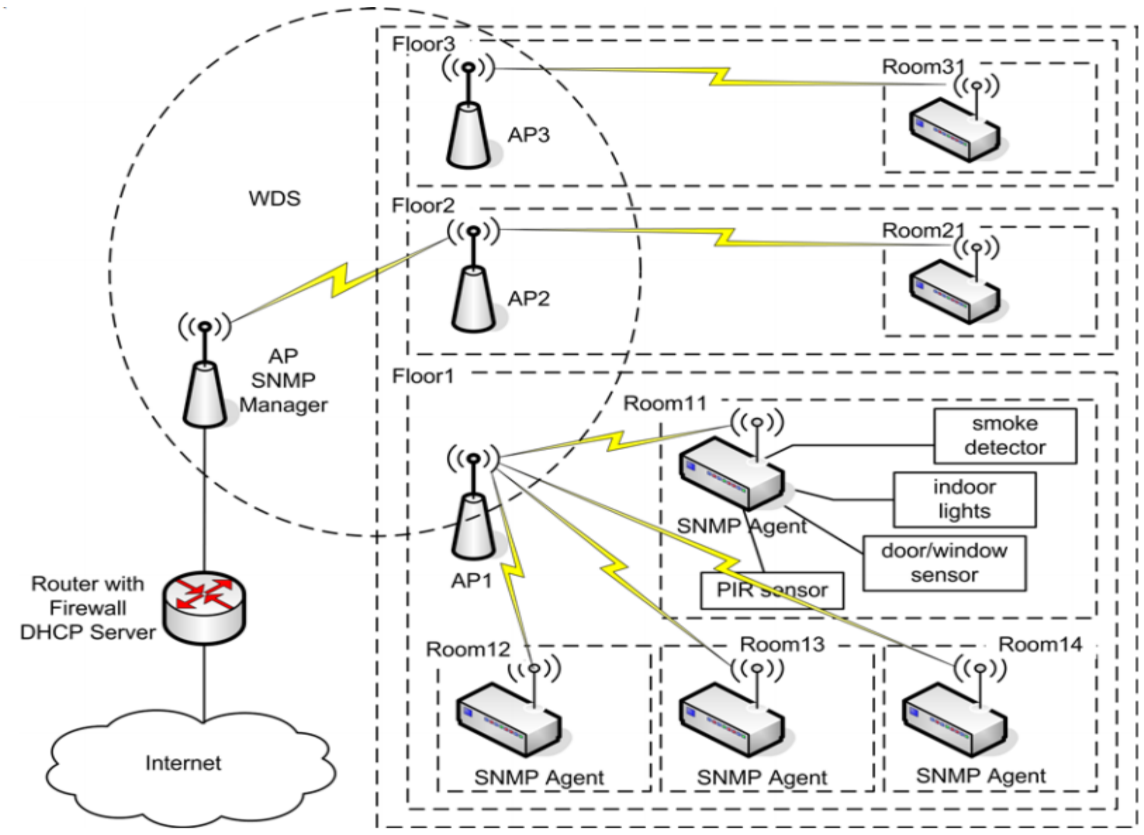


Figure 6 SNMP in Home automation

2.2 SIP in Home Automation

There are some papers talking about using SIP as management mechanisms, such as [17]. In that thesis, it combined SIP and ZigBee [18] in home automation, as shown in Figure 7. The SIP services were only implemented on a ZigBee/Ethernet gateway but not on end devices. When user agents send SIP messages to a ZigBee/Ethernet gateway, the ZigBee/Ethernet gateway translates SIP messages into ZigBee messages and sends them to end devices. In that thesis, they need not worry about the NAT traversal problem because SIP provides registration mechanism.

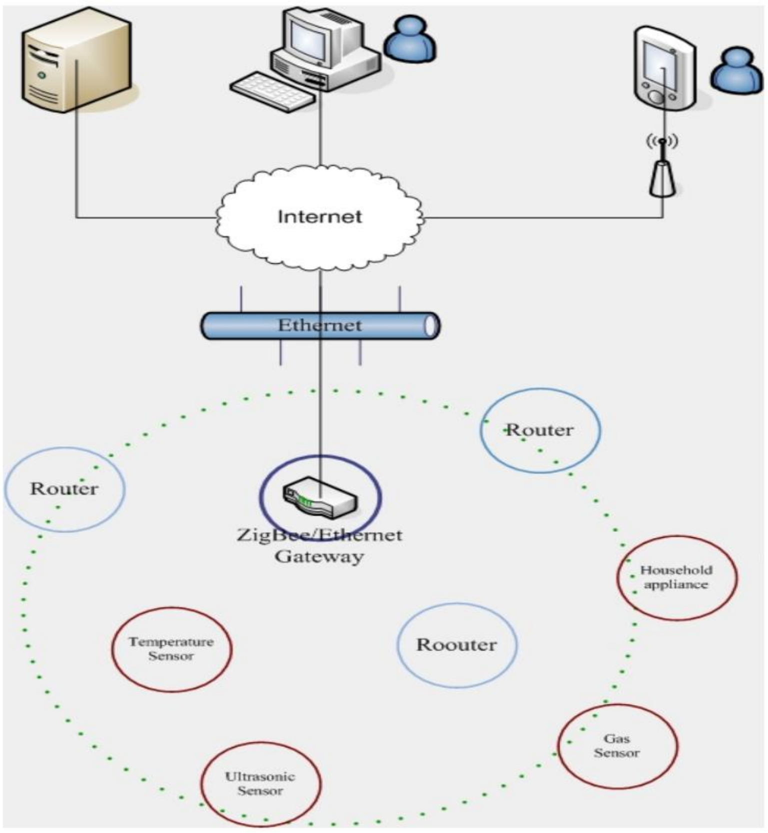


Figure 7 SIP in Home automation

In this approach, they extended SIP headers to describe more details of end devices. However, they did not define their own headers precisely in the thesis. As the result, it is difficult to distinguish each filed of the SIP header. Furthermore, their user interface was written in C# programming language, which is an additional component to be installed.

2.3 XMPP in Home Automation

There are some protocols that have also been proposed in home automation, such as Extensible Messaging and Presence Protocol (XMPP). In [19] XMPP and Open Service Gateway Initiative (OSGi) technology is used to define a solution of service delivery for home automation between remote service providers and local devices. In that paper, they proposed OSGi for the detection function, while XMPP was applied to build a mechanism of message notification. The flow of fault detection of that paper includes four parts: (1) the state capturing component, (2) the knowledge base which stores each kind of refrigerator state, (3) the inference engine which infers error causes, (4) the user interface shows users the inferring results. The flow of fault detection of digital refrigerator was shown in Figure 8. However, because of the decentralized management, any two XMPP servers can communicate with each other. Therefore, an XMPP user may receive messages from unauthorized users.

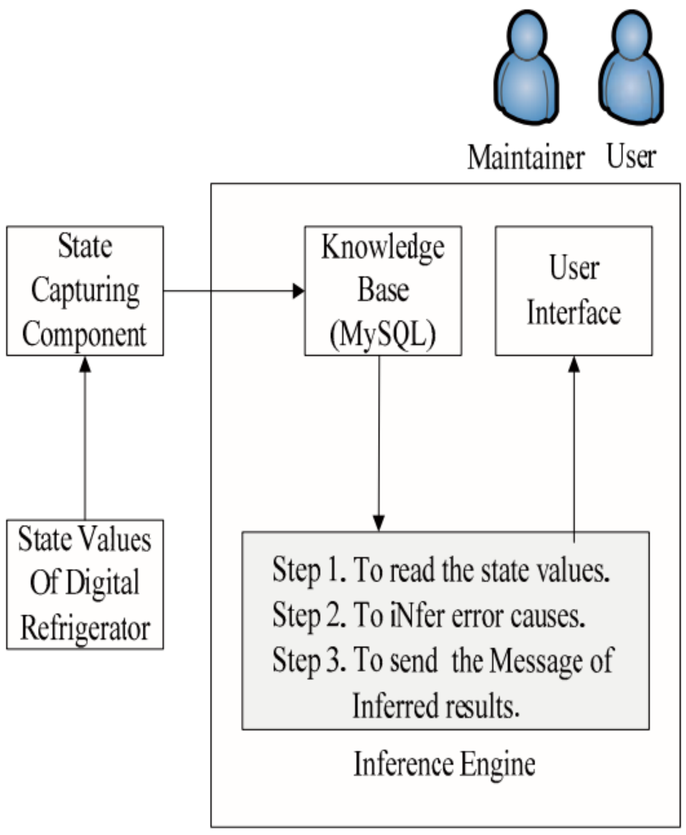

Maintainer
User
State Capturing Component
Knowledge Base (MySQL)
User Interface
State Values Of Digital Refrigerator
Step 1. To read the state values.
Step 2. To iNfer error causes.
Step 3. To send the Message of Inferred results.
Inference Engine

Figure 8 The flow of fault detection of digital refrigerator

For the three different protocols (SNMP, SIP, XMPP), we used these three protocols as key words and combine them with two additional keywords ("Smart Grid" and "home automation") to search on the IEEE Xplorer database. The results are shown in Figure 9. As we can see, SIP becomes a popular issue in Smart Grid management.

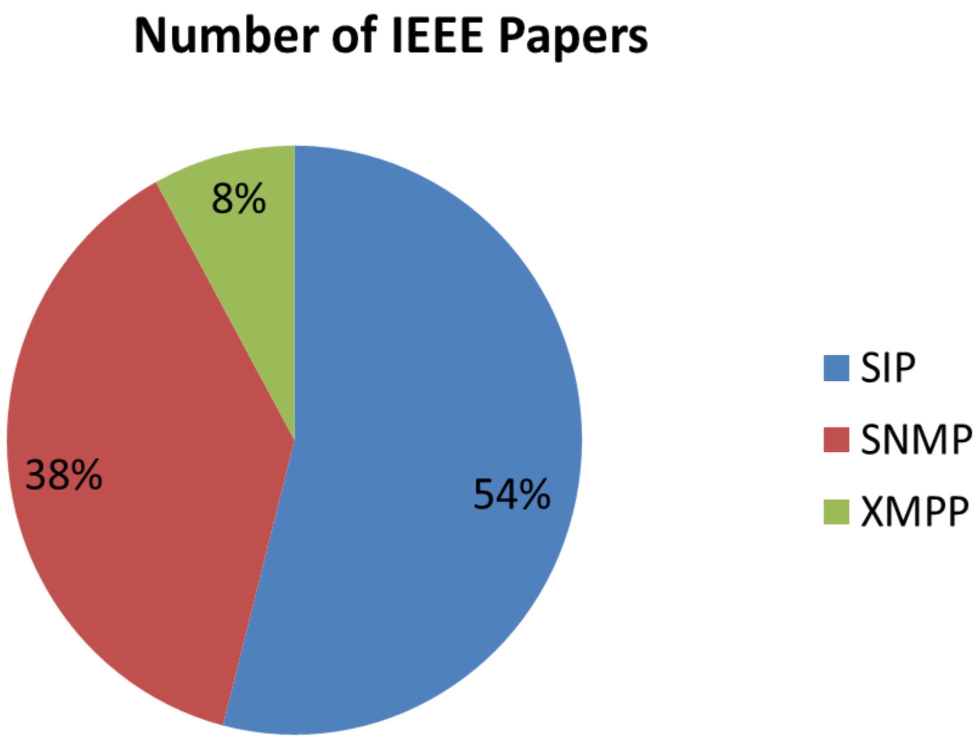


Figure 9 Number of IEEE papers

Chapter 3. System Architecture

In this chapter, we shall introduce our system architecture as shown in Figure 10.

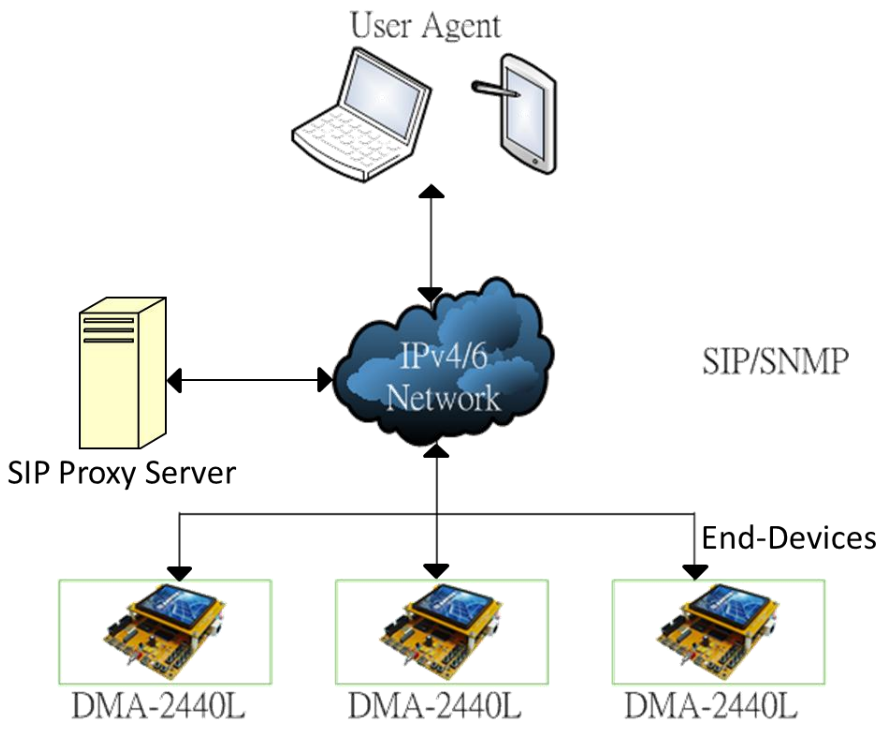


Figure 10 System Architecture

3.1 .Components

Our system consists of three components: (1) user agents, (2) SIP proxy server, and (3) end devices. SNMP has no registration mechanism, so unlike SIP, we need not specifically allocate a machine as a SNMP server. User agents in Figure 10 are designated as managers in our system. User agents send control messages to end devices and end devices send back information (such as system uptime or system descriptions) to user agents. In our scenario, user agents can be desktops, laptops, smart phones, or any device with Internet connection. User agents can easily control end devices remotely. And user agents do not need to install special software to control end devices. In our system architecture, we use CSIPSimple [20] as a SIP softphone running on the Android 2.3.7 operating system, to send SIP messages to end devices. For SNMP, we use SNMP Management Service [21] to send SNMP messages to control end devices. Furthermore, these software tools can be easily found and, the best thing is, they are free!

A SIP proxy server is indispensable in our system architecture. User agents send SIP messages via the SIP proxy server. Furthermore, SIP provides registration mechanism, every node registers to SIP proxy server in our scenario. It helps us to achieve centralized management. The SIP proxy server not only helps us to manage user agents and end devices but also helps us to solve the NAT traversal problem. We install OpenSER as our SIP proxy server running on the FreeBSD 8.2 operating system. OpenSER is a SIP proxy server, call router and user agent registration server used in VoIP and instant messaging applications. Now, the OpenSER has been renamed to OpenSIPS [22].

We use DMA-2440Ls as end devices in our system. We shall introduce them with more details in chapter four.

3.2 Protocols

. Open System Interconnection (OSI) reference model was defined by International Standard Organization (ISO). In order to interconnect different networks, OSI model defines several layers of protocols, as shown in Figure 11. OSI model is a precise concept, but it is difficult to write a program sometimes. Therefore, Defense Advanced Research Project Agency (DARPA) proposed the TCP/IP model. TCP/IP model has a similar idea to OSI model, but it simplified OSI seven-layer model to four layers.

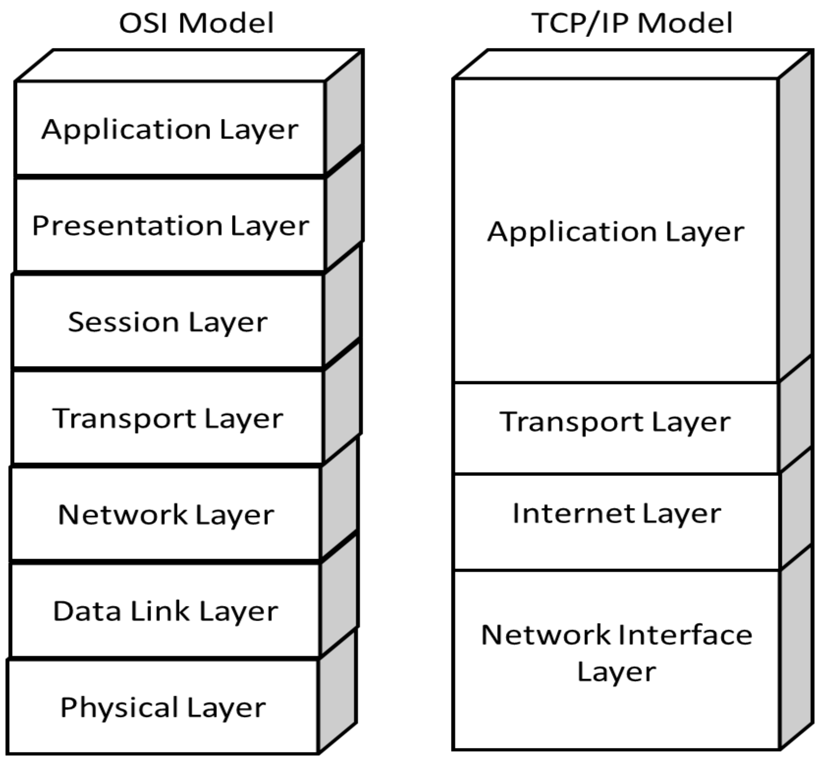


Figure 11   Protocol models

Our research focuses on Internet-layer and application-layer. TCP/IP model provides simple idea to describe the Internet. Therefore, we use TCP/IP as the reference model in our reasearch. In the Internet-layer, we provide Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6) to access the Internet. Even IPv6 has been brought up over a decade, IPv4 is still popular all over the world. Therefore, our system provide dual stack protocols in every component. More details of IPv4 and IPv6 will be given in the following subsections. In the application layer, we utilize SNMP and SIP protocols. Evaluating the differences of these two protocols is the main goal in this thesis.

3.2.1 Internet Protocol version 4

The Internet is a global system of interconnected computer networks that use the standard Internet protocol suite to serve billions of users all over the world. The components of a basic Internet network are shown in Figure 12. Every device connecting to the Internet must possess an IP address. The Internet consists of a great deal of these devices such as personal computers (PC), laptops and printers.

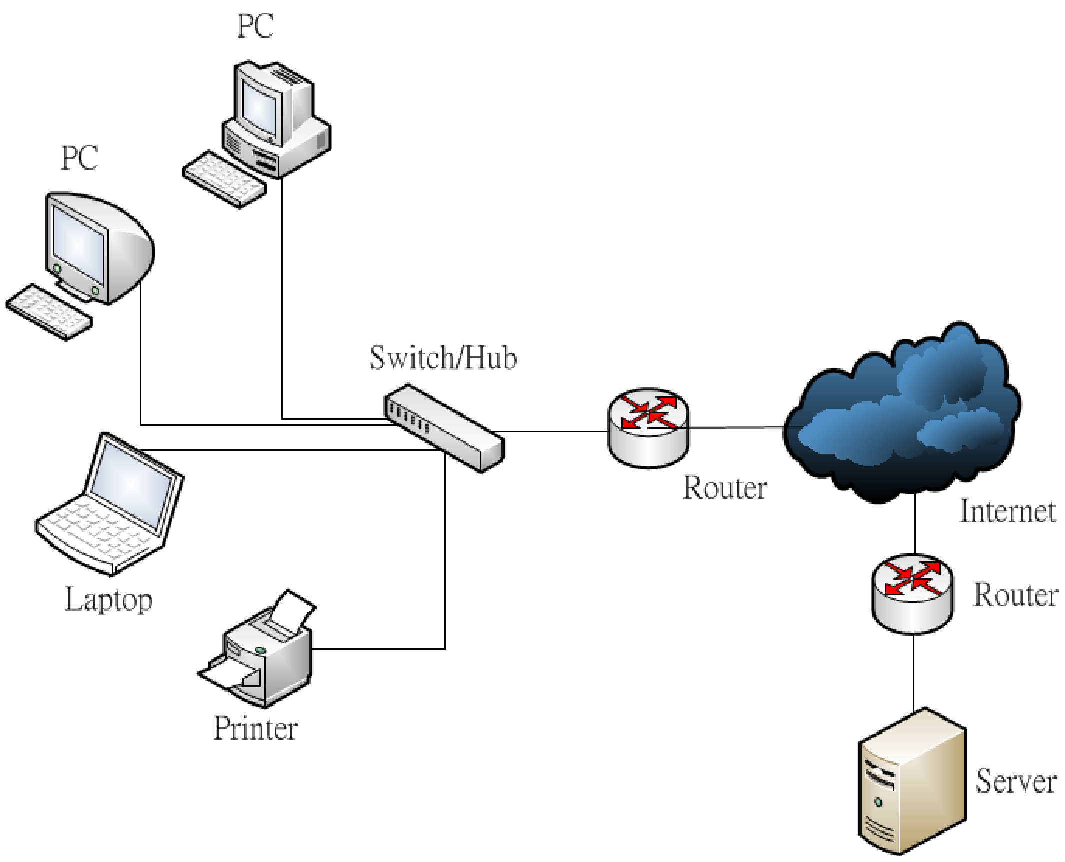
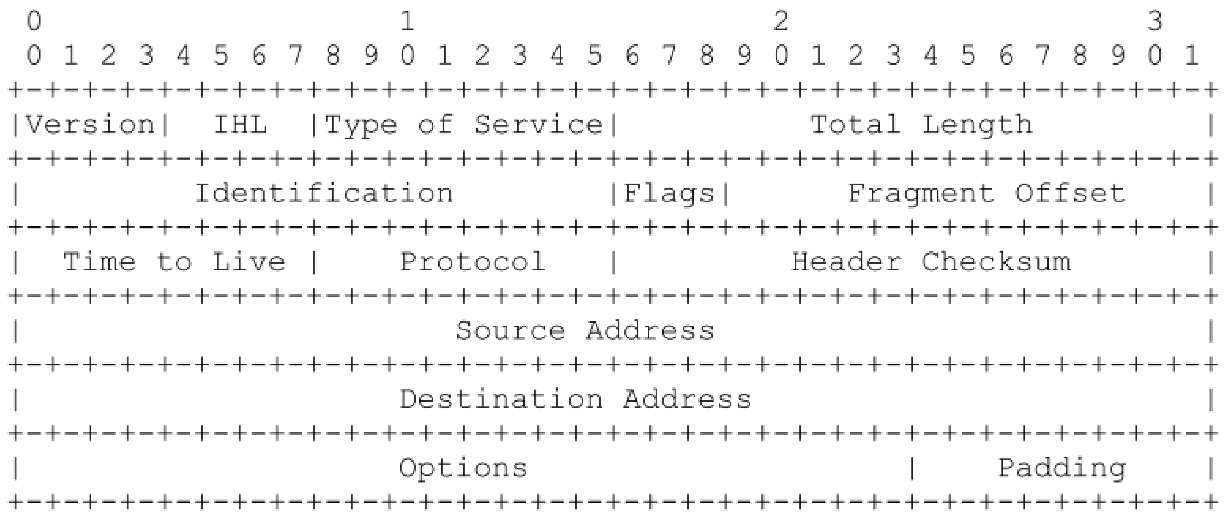


Figure 12 A basic network which is part of the Internet

In order to connect each node of Internet, many Internet-layer protocols were proposed. Nowadays, the most popular Internet-layer protocol is IPv4 [23]. The header of

IPv4 is shown in Figure 13. Each field in the IP packet contains different information to instruct the network how to handle the packet. For instance, the Time to Live (TTL) field indicates the lifetime of a packet, where the range of TTL is between 0 to 255. The value of TTL decreases when a packet passes through a router. When TTL reaches 0, then the packet must be dropped. The Protocol field identifies the higher-layer protocol carried in the packet. Some possible values are listed in Table 1 .



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13 IPv4 header

Table 1 Protocol values of IPv4 header [24]

Value (Hexadecimal)	Value (Decimal)	Protocol
00	0	Reserved
01	1	ICMP
02	2	IGMP
03	3	GGP
04	4	IP-in-IP Encapsulation
06	6	TCP
08	8	EGP
11	17	UDP

IPv4 addresses are fixed length of four octets (32 bits). An address begins with a network number, followed by a local address. As the length of IPv4 address is 32 bits, this limits the address space to 2^{32}, that is, less than 4.3 billion. In February 2011, it is formally announced that IPv4 address has been exhausted [25] because of the rapid growth of the Internet. Therefore, a new version of Internet Protocol was proposed.

3.2.2 Internet Protocol version 6

IPv6 [26] is the next generation standard of Internet Protocol. Since IPv4 address exhausted in February 2011, people are to use IPv6. IPv6 specifies a new header format, which is intentionally designed to ease the packet header processing by routers. The IPv6 header format is shown in Figure 14.Because the headers of IPv4 packets and IPv6 packets are significantly different, these two Internet Protocols are not interoperable. However IPv6 is a conservative extension of IPv4, most transport-layer or application-layer protocols need little or even no change to operate over IPv6. The main advantage of IPv6 over IPv4 is its larger address space. The length of IPv6 address is 128 bits, which means that theoretically IPv6 can have 2^{128} addresses. Compared to IPv4, IPv6 provides abundant IP addresses.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14 IPv6 header

According to Figure 14, the header size has a fixed length of 40 octets. The Options header field in IPv4 is now implemented as additional extension headers after the IPv6 header, which limits the size only by the size of an entire packet. The extension header mechanism makes IPv6 protocol extensible, and allow future services like security and quality of service (QoS) to be easily introduced. Furthermore, IPv6 has simplified its header format. Although an IPv6 address is four times more than an IPv4 address, the header size of IPv6 is only twice of an IPv4 header.

IPv6 addresses are written in eight groups of four hexadecimal digits; these eight groups are separated by colons, such as 2620:0:1cfe:face:b00c::3. The first four groups of IPv6 address denote the network prefix, while the rest groups denote the host identifier. The host identifier is self-generated according to the Media Access Control (MAC) address of the network interface card (NIC), which is called the Extended Unique Identifier (EUI)-64 as shown in Figure 15.

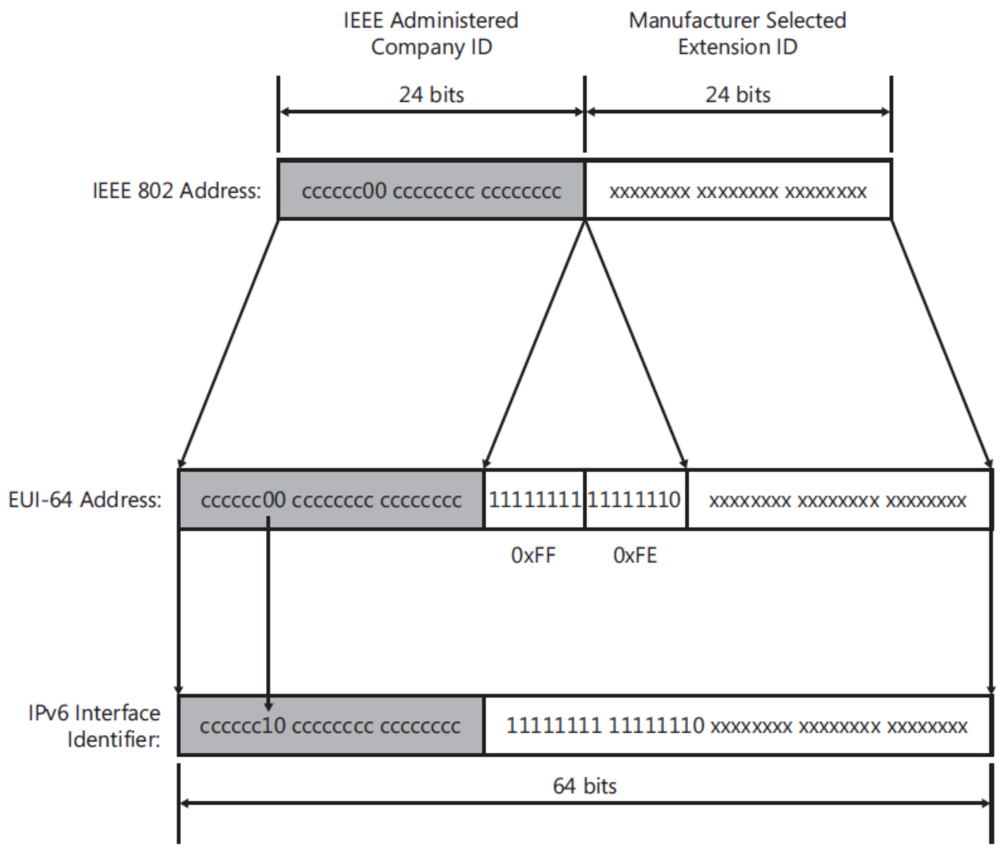

IEEE Administered Company ID
Manufacturer Selected Extension ID
24 bits
24 bits
IEEE 802 Address:
cccccc00 cccccccc cccccccc
xxxxxxxx xxxxxxxx xxxxxxxx
EUI-64 Address:
cccccc00 cccccccc cccccccc
11111111
11111110
xxxxxxxx xxxxxxxx xxxxxxxx
0xFF
0xFE
IPv6 Interface Identifier:
cccccc10 cccccccc cccccccc
11111111 11111110 xxxxxxxx xxxxxxxx xxxxxxxx
64 bits

Figure 15 EUI-64 generation

IPv6 supports auto-configuration addressing, with either stateful and stateless approaches. The IPv6 stateful address auto-configuration is assigned by Dynamic Host Configuration Protocol version 6 (DHCPv6) [27]; the StateLess Address Auto Configuration (SLAAC) is specified in RFC 4862 [28], in which a network device acquires a 64-bit prefix from its local router and combines that with its own 64-bit host identifier (EUI-64 as mentioned above). After acquiring an IPv6 address, the network device needs to perform Duplicate Address Detection (DAD) [29] to make sure its IPv6 address is unique in Internet.

3.2.3 Simple Network Management Protocol

SNMP is an application-layer protocol that offers network management services in IP networks. Devices that typically support SNMP include routers, switches, workstations, servers, printers, etc. SNMP provides a consistent mechanism to manage these devices. SNMP is based on a client/server model in which the client issues requests to the server and the server processes requests and responds to the client. In SNMP terminologies, an “agent” denotes a client, and a manager denotes a server. The essence of SNMP is that information can be collected by simple operating procedures. The SNMP protocol consists of some basic commands such as:

1. Read: The manager retrieves the values of objects from the agent.
2. Write: The manager sets the values of objects at the agent.
3. Trap: Agent notifies the manager about the significant events.

To enable SNMP services, there must be a manager, an agent, the protocol and the Management Information Base (MIB). A manager is an application of network management to collect and analyze the information which is sent by agent. An agent is a process of an end device which is used for receiving commands and sending information to manager. The protocol refers to SNMP which uses User Datagram Protocol (UDP) to delivery segments between a manager and an agent. SNMP provides five message formats to exchange information between a manager and an agent as shown in Figure 16. MIB is a set of managing information. In general, the architecture of SNMP is quite simple. However, different network devices provide different functions. Therefore, we need different MIB modules to provide different functions of different devices.

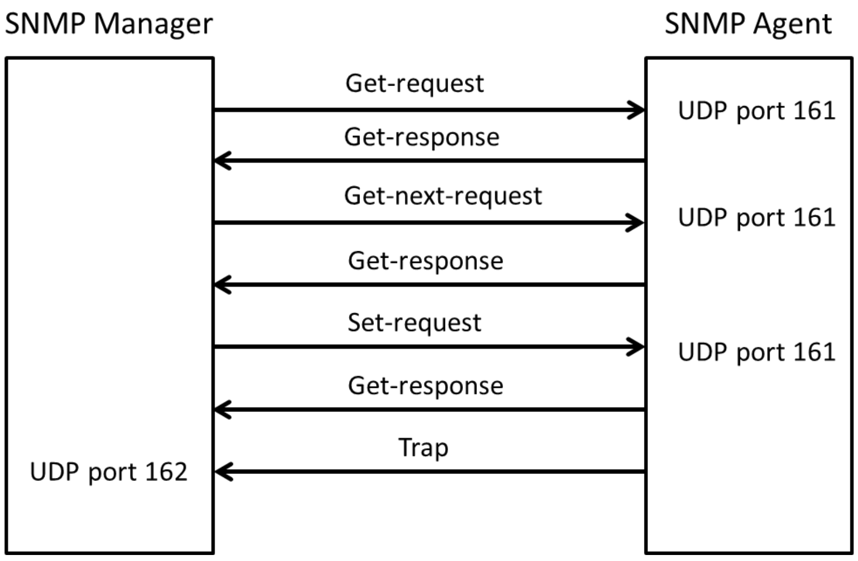


Figure 16 SNMP messages exchanging

With SNMP, a network administrator can control an agent from a manager. However, the administrator needs to know what information is supported of the agent, then, he can analyze it. Therefore, managing information of end device needs to translate into object in which is called Managed Object (MO). MIBs consist of a lot of MO. Except for standard MIBs, there are many manufacturers provide additional MIBs to have an advantage.

Internet Engineering Task Force (IETF) not only defined the standard of SNMP, but also defined common MOs of network devices for unified information. The definitions of MOs were written in RFC1213 [30]. An object assigns an unambiguous identification is required. This is achieved by registration. The registration tree is managed in a completely decentralized way and it is impossible to be exhaustive. The registration tree shows as Figure 17. The registration is defined and managed following ITU-T X.660 & 670 Recommendation series.

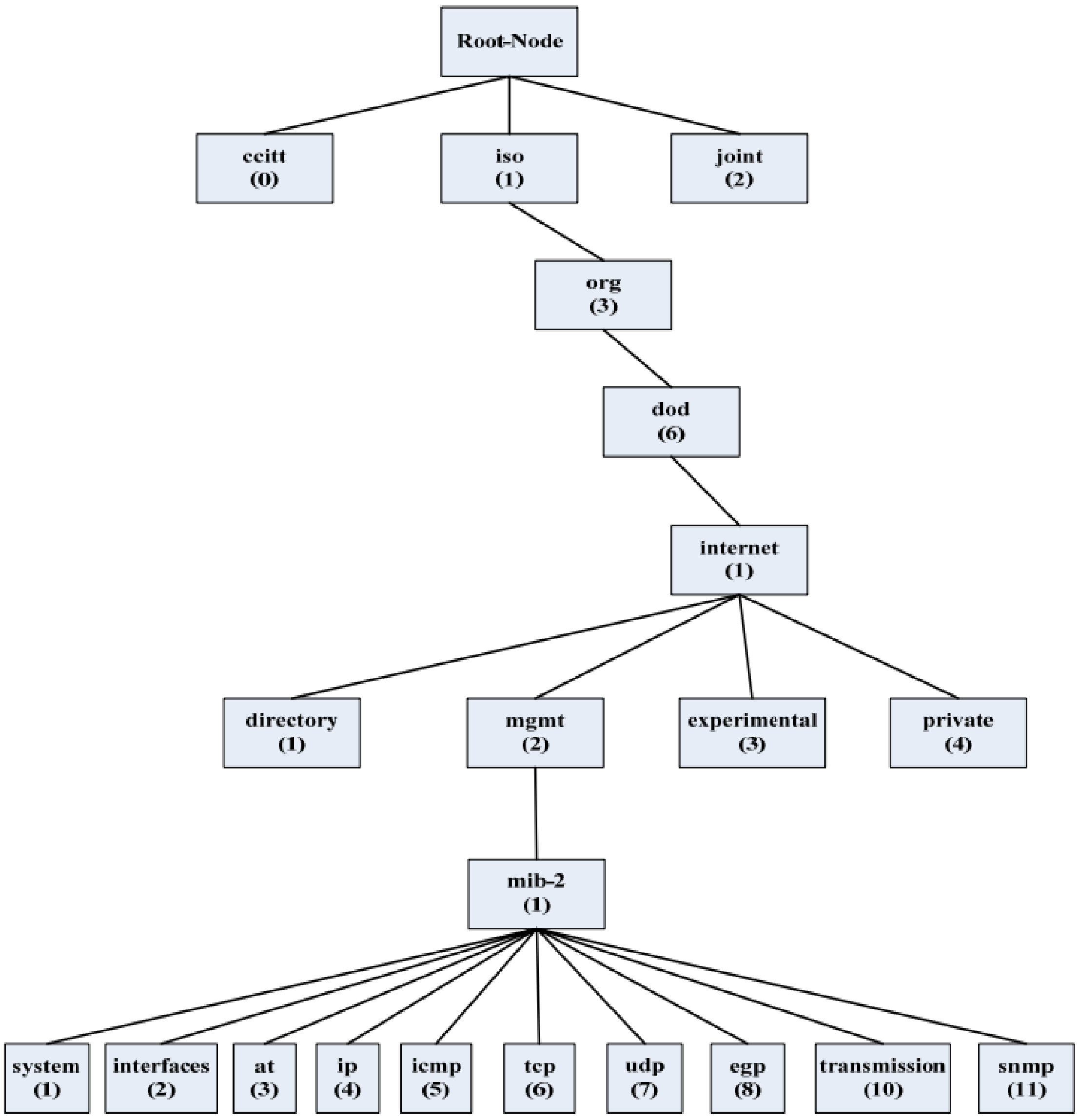


Figure 17 The registration tree [31]

3.2.4 Session Initiation Protocol

SIP is a plain text signaling protocol in application-layer that is used to establish, modify and terminate multimedia sessions such as Internet telephony calls. SIP uses a Uniform Resource Identifier (URI) as an addressing format. For example, a SIP URI sip:Memphis@example.com is easy to memorize because it looks like an email address. SIP defines two types of messages: request and response. Some SIP request messages are shown in Table 2 **,** while some SIP response messages are shown in Table 3 .

Table 2 SIP request messages

Request Methods	**Description**	**Defined in**
INVITE	Establishes a session	RFC 3261
ACK	Confirms an INVITE request	RFC 3261
BYE	Ends a session	RFC 3261
CANCEL	Cancels establishing of a session	RFC 3261
OPTIONS	Queries the capabilities of SIP phones	RFC 3261
REGISTER	Communicates user location (host name or IP address)	RFC 3261
PRACK	Provisional acknowledgement	RFC 3262
NOTIFY	Notify the subscriber of a new event	RFC 3265
SUBSCRIBE	Subscribes for an event of notification	RFC 3265
UPDATE	Modifies the state of a session without changing the state of the dialog	RFC 3311
MESSAGE	Transports instant	RFC 3428

	messages using SIP	
PUBLISH	Publishes an event to the server	RFC 3903
INFO	Sends mid-session information that does not modify the session state	RFC 6086

Table 3 SIP response messages

Status code	**Description**
1xx	Informational responses
2xx	Success responses
3xx	Redirection responses
4xx	Request failures
5xx	Server errors
6xx	Global failures

3.3 Services

In our system architecture, we provide two different protocols of services. The first one is using SNMP in our Microgrid as shown in Figure 18. We use mobile devices as SNMP managers, which send SNMP requests to query the system uptime of end devices. End devices send back responses to user agents. Therefore, user agents can control end devices and collect information of end devices remotely.

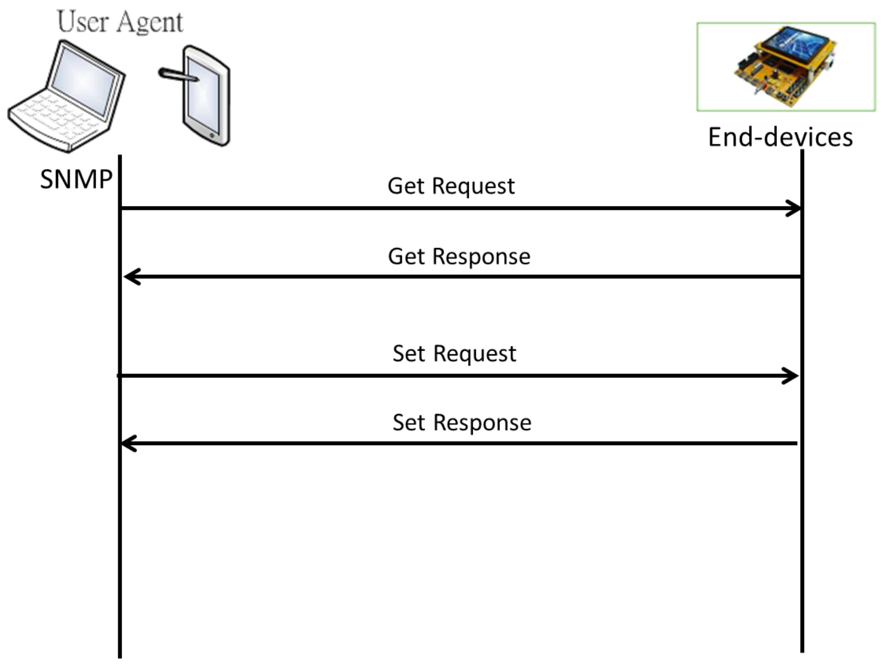


Figure 18 SNMP in Microgrid

The other service in our system architecture is using SIP in Microgrid as shown in Figure 19. User agents first register to SIP proxy server, so do end devices. Then, user agents are able to send SIP messages to end devices via the SIP proxy server. Afterwards, end devices send back response to user agents via the SIP proxy server. Therefore, user agents can control end devices and collect information of end devices remotely. One significant difference between SNMP and SIP is that SIP does not require every end device to possess a public IP addresses for remote management.

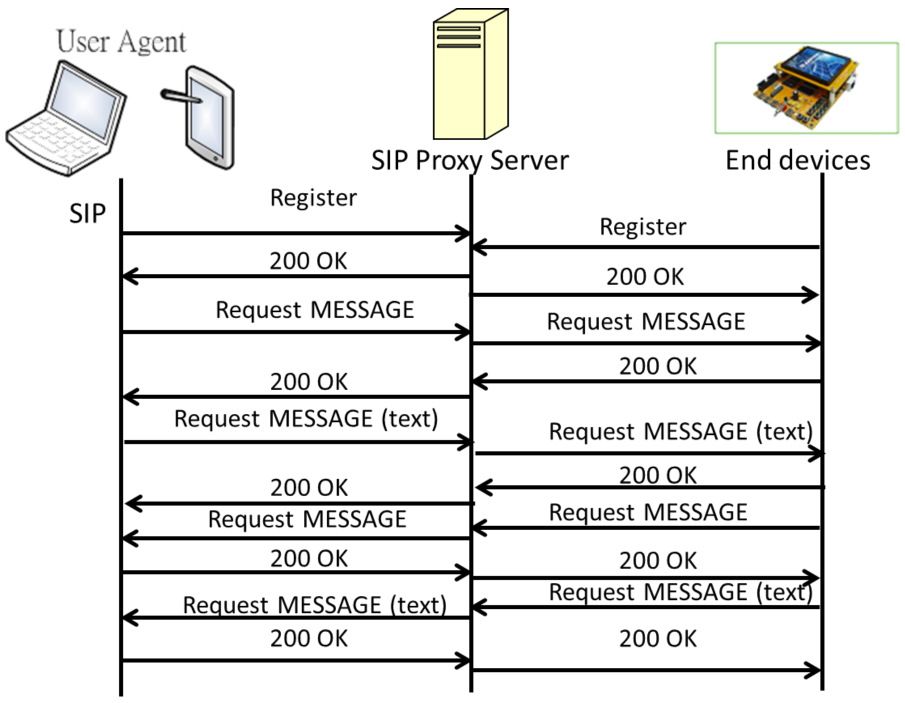


Figure 19 SIP in Microgrid

Figure 20 shows an example of SIP request message in Smart Grid management. The management data was encoded in XML format. XML tags are used to categorize information of smart devices. For example, tag `<type>` denotes the query types of messages. The query types including six different types:

0: loginIBC, to login an in-building controller (IBC).

1: queryPower, to query power usage information of a smart device.

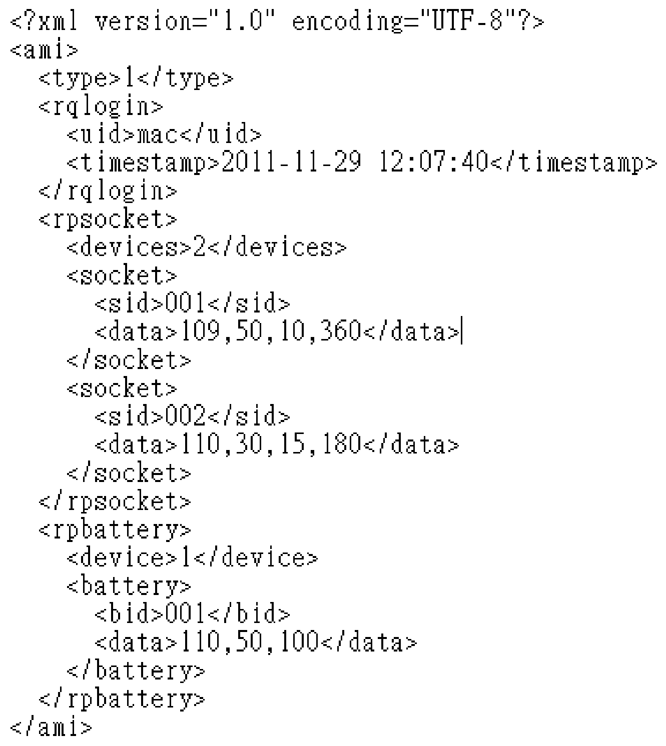
2: warntoChange, to warn the administrator that the status of this smart device has changed.

3: retrPower, to retransmit information of power usage.

4: ack, to acknowledge the instruction sent from the administrator.

5: socketStatus, to represent status of power socket.

Tag `<devices>` denotes the information of how many smart devices will be sent. Tag `<sid>` denotes the identities of smart devices. The most attractive information may be the one contained in tag `<data>`, which shows the voltage, current, electric power and dielectric phase angle of smart devices. These SIP request messages, together with the information contained in the XML tages, allows an administrator to manage smart devices with SIP flexibly.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<ami>
  <type>1</type>
  <rqlogin>
    <uid>mac</uid>
    <timestamp>2011-11-29 12:07:40</timestamp>
  </rqlogin>
  <rpsocket>
    <devices>2</devices>
    <socket>
      <sid>001</sid>
      <data>109,50,10,360</data>
    </socket>
    <socket>
      <sid>002</sid>
      <data>110,30,15,180</data>
    </socket>
  </rpsocket>
  <rpbattery>
    <device>1</device>
    <battery>
      <bid>001</bid>
      <data>110,50,100</data>
    </battery>
  </rpbattery>
</ami>
```

Figure 20 Example of Smart Grid management in SIP

Chapter 4. Implementation and Performance Evaluation

In this thesis, we use a DMA-2440L embedded Linux development platform as our end device. However, the default configuration of DMA-2440L does not IPv6, SNMP and SIP. Therefore, the first step of our implementation is to include modules of these protocols. In our scenario, we have two different protocols to provide services. Therefore, we have two different protocol stacks. The protocol stacks of SNMP and SIP are shown in Figure 21 and Figure 22. All the tools in this thesis which can be found in the following hyper link: http://ms11.voip.edu.tw/~memphis/Download. Also, you can find them from Internet.

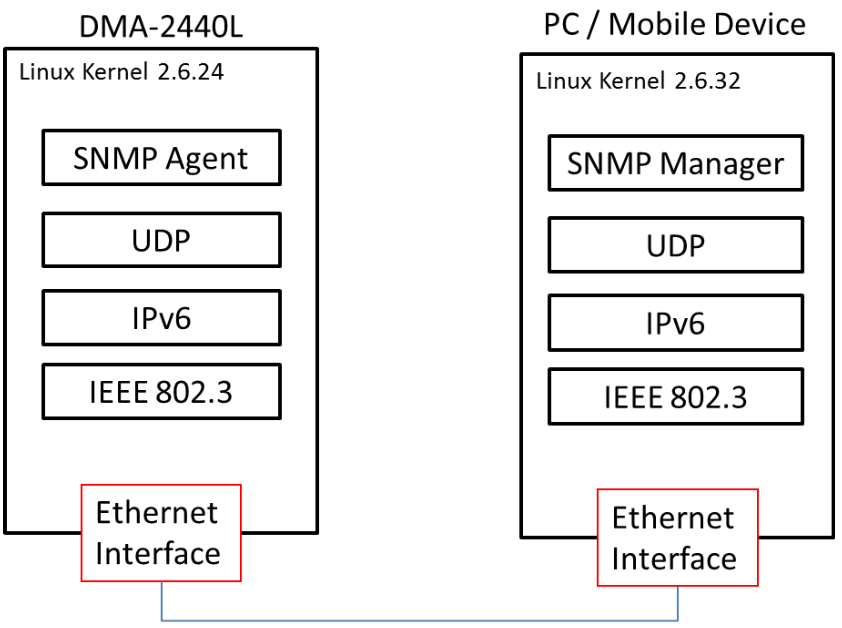


Figure 21 Protocol stack of SNMP management mechanism

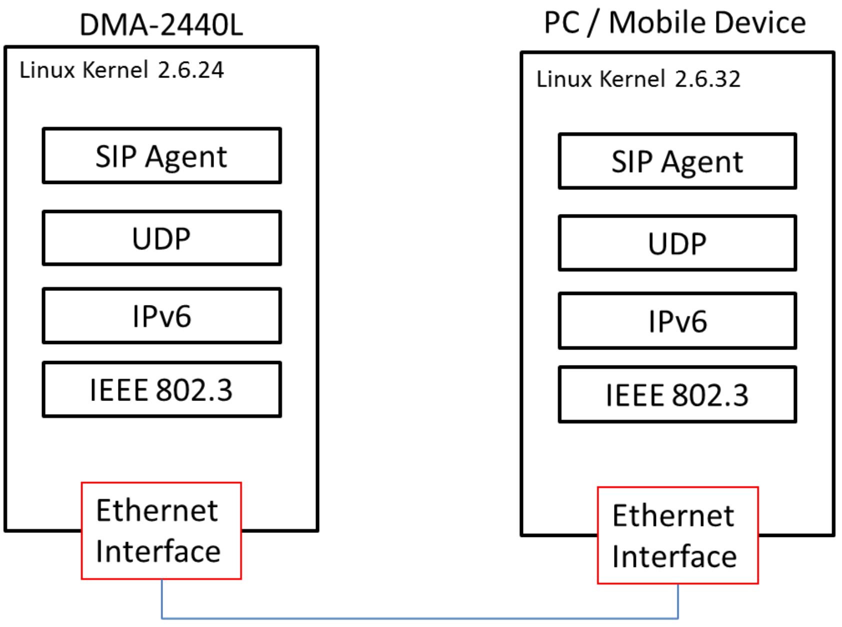
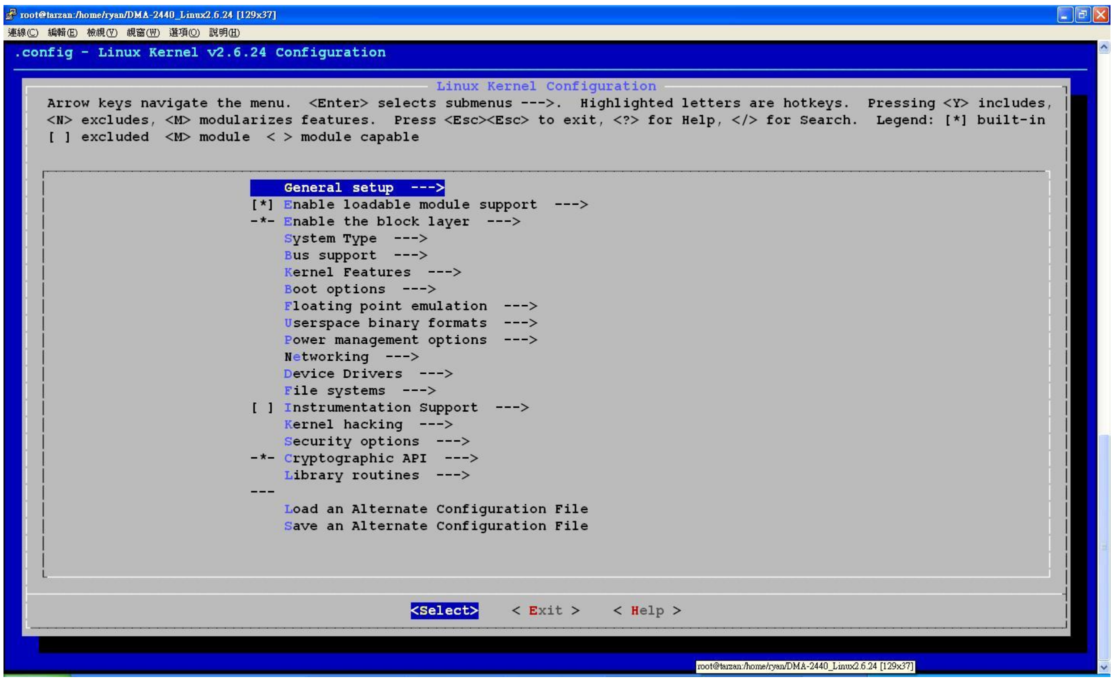


Figure 22 Protocol stack of SIP management mechanism

4.1 Implementation

4.1.1. Kernel Porting

DMA-2440L uses S3C2440 processor of ARM9 series which is made by Samsung. DMA-2440L provides a stable clock rate at 400MHz. Furthermore, with the 64MB memory supports, DMA-2440L provides a good performance and a low cost for users who develop applications on embedded systems. DMA-2440L provides various drivers to support different interfaces in Linux, so it is convenient to develop an experimental system with this platform. However, to enable the full services in Figure 21 of DMA-2440L, lots of re-configuration work must be done. DMA-2440L is equipped with Linux kernel 2.6.24. To enable IPv6, we need to re-compile the kernel. Linux kernel porting includes kernel configuring, kernel compiling and kernel loading. To configure a kernel, we type the command: `#make menuconfig` in a terminal. This displays a menu of kernel configuration as shown in Figure 23. Because DMA-2440L limits its kernel size to be less than 2MB, unnecessary features must be removed, such as multicasting, tunneling, RARP, etc. After kernel configuration is finished, we type the following command: `#make zImage` to compile the kernel with the cross-compiler `arm-linux-gcc-4.0.3`. This will generate two files, including Image and zImage. An Image file is an uncompressed kernel. A zImage is a compressed image file, which provides smaller size of kernel. Because zImage has smaller size than Image, so it is appropriate to embedded systems. DMA-2440L can use Trivial File Transfer Protocol (TFTP) to download the zImage and write it to DMA-2440. So far, the kernel porting is finished.



```
root@tarzan:/home/ryan/DMA-2440_Linux2.6.24 [129x37]
連線(C)  編輯(E)  檢視(V)  視窗(W)  選項(O)  說明(H)
.config - Linux Kernel v2.6.24 Configuration
                          Linux Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y> includes,
  <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in
  [ ] excluded  <M> module  < > module capable

                        General setup  --->
                    [*] Enable loadable module support  --->
                    -*- Enable the block layer  --->
                        System Type  --->
                        Bus support  --->
                        Kernel Features  --->
                        Boot options  --->
                        Floating point emulation  --->
                        Userspace binary formats  --->
                        Power management options  --->
                        Networking  --->
                        Device Drivers  --->
                        File systems  --->
                    [ ] Instrumentation Support  --->
                        Kernel hacking  --->
                        Security options  --->
                    -*- Cryptographic API  --->
                        Library routines  --->
                    ---
                        Load an Alternate Configuration File
                        Save an Alternate Configuration File

                        <Select>    < Exit >    < Help >
```

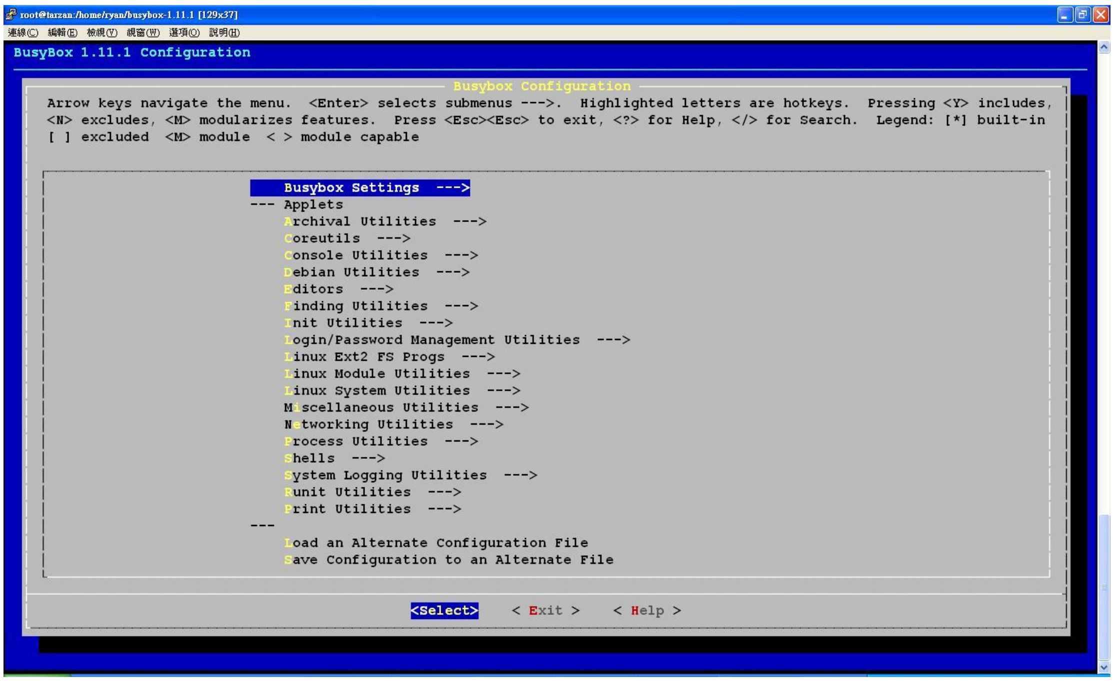
Figure 23 Kernel configuring

4.1.2 File System

After the kernel is ready, the platform still needs a file system which consists of structures necessary for storing and managing data. These structures typically include an operating system boot record, directories, and files. The main functions of a file system include tracking allocated and free space, maintaining directories and file names, tracking where each file is physically stored on the disk. There are some common file systems used for embedded Linux, such as RoomFS, Second Extended File System (EXT2), RAMDISK, Compressed ROM File System (CRAMFS), etc. CRAMFS is a compressed file system which does not need to decompress all the contents of image into memory. It allows us to utilize maximum number of memory. Therefore, we choose CRAMFS as the file system of our platform.

In this thesis, we use BusyBox 1.11.1 [32] to create a CRAMFS file system. BusyBox is a development kit which combines tiny versions of many common UNIX utilities into a single small executable. BusyBox provides a fairly complete POSIX environment for any small or embedded system. File system configuring is similar to kernel configuring. Typing the command: #`make menuconfig` to configure the settings of file system. As Figure 24 shows, we select the options to include the utilities such as `which` and `gzip` that are necessary in our experiment. Then, type #make to run the cross-compiler. This will generate /etc, /usr, profile, fstab, etc. These files are located at the _install/ directory. After creating a file system, typing #`mkdir to` create another directory whose name is rootfs/. Copy all the contents from _install/ directory to rootfs/ directory.

Net-SNMP [33] and Sofia-SIP [34] are chosen as the protocol stacks to support SNMP and SIP, so we need to download the source code of these libraries use a cross-compiler to generate the libraries for the ARM processor in DMA-2440L. Copy the above libraries to rootfs/ and type #`./mkcramfs`, then, the file system is created. And its file name is root_dma.cramfs. Use TFTP to download the file system and burn it into DMA-2440L. After rebooting the DMA-2440L, our system porting is ready and functioning as shown in Figure 25.



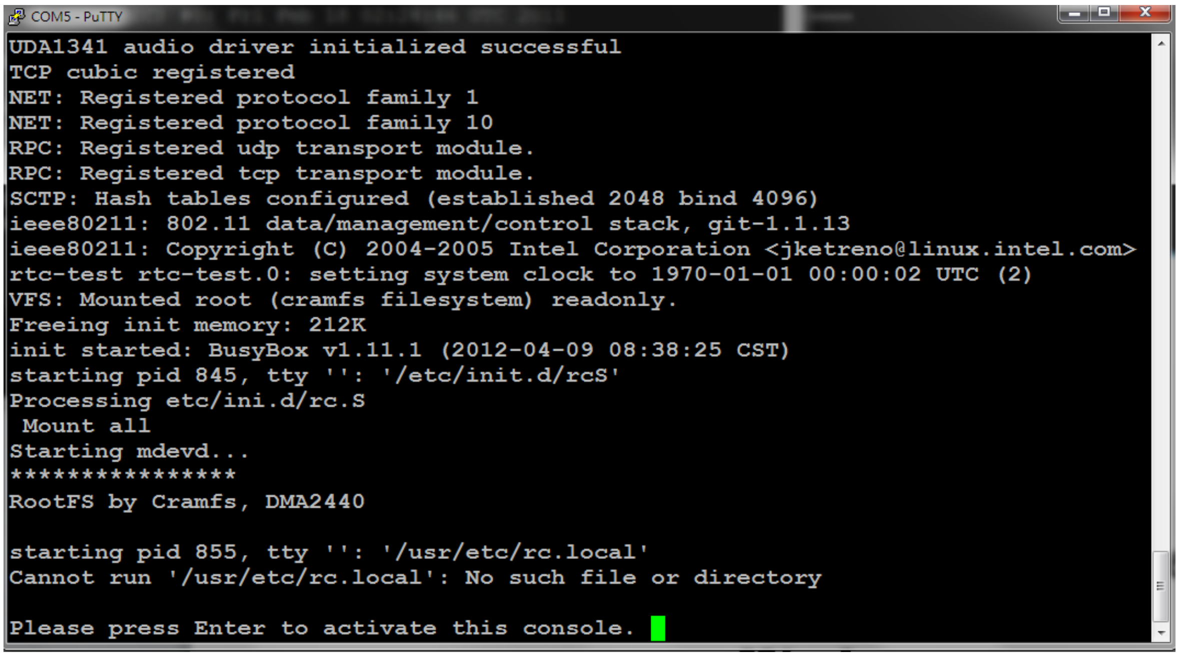
```
root@tarzan:/home/ryan/busybox-1.11.1 [129x37]
連線(C)  編輯(E)  檢視(V)  視窗(W)  選項(O)  說明(H)
BusyBox 1.11.1 Configuration

                         Busybox Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y> includes,
  <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in
  [ ] excluded  <M> module  < > module capable

                        Busybox Settings  --->
                    --- Applets
                        Archival Utilities  --->
                        Coreutils  --->
                        Console Utilities  --->
                        Debian Utilities  --->
                        Editors  --->
                        Finding Utilities  --->
                        Init Utilities  --->
                        Login/Password Management Utilities  --->
                        Linux Ext2 FS Progs  --->
                        Linux Module Utilities  --->
                        Linux System Utilities  --->
                        Miscellaneous Utilities  --->
                        Networking Utilities  --->
                        Process Utilities  --->
                        Shells  --->
                        System Logging Utilities  --->
                        Runit Utilities  --->
                        Print Utilities  --->
                    ---
                        Load an Alternate Configuration File
                        Save Configuration to an Alternate File

                      <Select>    < Exit >    < Help >
```

Figure 24 BusyBox configuring



```
UDA1341 audio driver initialized successful
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 10
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
SCTP: Hash tables configured (established 2048 bind 4096)
ieee80211: 802.11 data/management/control stack, git-1.1.13
ieee80211: Copyright (C) 2004-2005 Intel Corporation <jketreno@linux.intel.com>
rtc-test rtc-test.0: setting system clock to 1970-01-01 00:00:02 UTC (2)
VFS: Mounted root (cramfs filesystem) readonly.
Freeing init memory: 212K
init started: BusyBox v1.11.1 (2012-04-09 08:38:25 CST)
starting pid 845, tty '': '/etc/init.d/rcS'
Processing etc/ini.d/rc.S
 Mount all
Starting mdevd...
****************
RootFS by Cramfs, DMA2440

starting pid 855, tty '': '/usr/etc/rc.local'
Cannot run '/usr/etc/rc.local': No such file or directory

Please press Enter to activate this console.
```

Figure 25 Embedded system start up

4.2 Performance Evaluation

4.2.1 Performance Analysis

Smart Grid is a new application for both SNMP and SIP, so evaluating their advantages and disadvantages on Smart Grid is certainly needed. In our scenario, both SNMP and SIP can do the same tasks. As an example, we chose responding MAC address as the event to evaluate the performances of SNMP and SIP with Wireshark [35]. In order to get a fair result of experiments, the experimental conditions of SNMP and SIP are the same. The end devices are DMA-2440L, using Ethernet to transmit frames. We assume that there is no packet loss in this small network. According to Figure 18, one request and one response are required to complete an event in SNMP. Meanwhile, according to Figure 19, four requests and four responses are required to complete an event in SIP. Suppose $T_{req.}$ denotes the latency of a request, $T_{res.}$ denotes the latency of a response,

$T_{PSNMP.}$ denotes the processing time of SNMP, and $T_{PSIP.}$ denotes the processing time of SIP. The execution time of SNMP and SIP are shown in Table 4 . The transmission time for SNMP and SIP are approximately the same, so we assume that in this table, $T_{req.} = T_{res.}$ for both SNMP and SIP. What we would like to further investigate is the difference of the protocol stack processing time.

Table 4 Execution time analysis for each protocol

Limited Time	**SNMP**	**SIP**
Execution Time	$T_{req.} + T_{res.} + T_{PSNMP.}$	$4T_{req.} + 4T_{res.} + T_{PSIP.}$

4.2.2 Experiment Analysis

Wireshark is a free and open-source packet analyzer. We use Wireshark to help us analyzing the information of each packet. We evaluate the performances of SNMP and SIP under the following two constraints, respectively:

1. *Limited Time*: Evaluate how many messages of SNMP/SIP can be sent in a limited time.
2. *Total Message*: Measure how long it takes for SNMP/SIP messages to be delivered.

According to the information captured by Wireshark, an SNMP message size is between 46 to 97 bytes. An SIP message size is between 706 to 758 bytes. The results of experiments are shown in Table 5 and Table 6 .

Table 5 Experiment result of limited time

Limited Time	**SNMP**	**SIP**
5 Seconds	67.3 Messages	232.3 Messages

15 Seconds	193.7 Messages	704.7 Messages
25 Seconds	319 Messages	1171 Messages
Average	12.9 Messages/Second	46.8 Messages/Second

Table 6 Experimental result of Total messages

Total Messages	**SNMP**	**SIP**
10 Messages	0.66 Seconds	0.33 Seconds
50 Messages	3.71 Seconds	1.17 Seconds
100 Messages	7.77 Seconds	2.22 Seconds
Average	0.08 Seconds/ Message	0.023 Seconds/Messages

We can find that the performance of SIP is almost four times more than SNMP. As Table 4 shows, we can say that the library of SIP that we chose is more efficient than the one for SNMP, so the total execution time gets shorter. According to the experiments, we can draw the following figures as shown in Figure 26 and Figure 27.

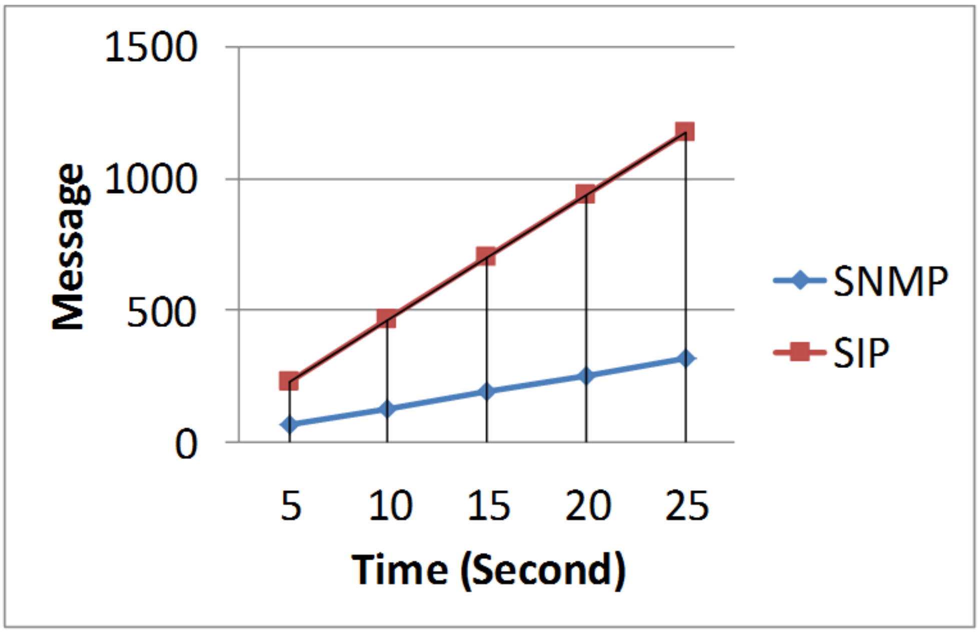


Figure 26 Messages sent in Limited time

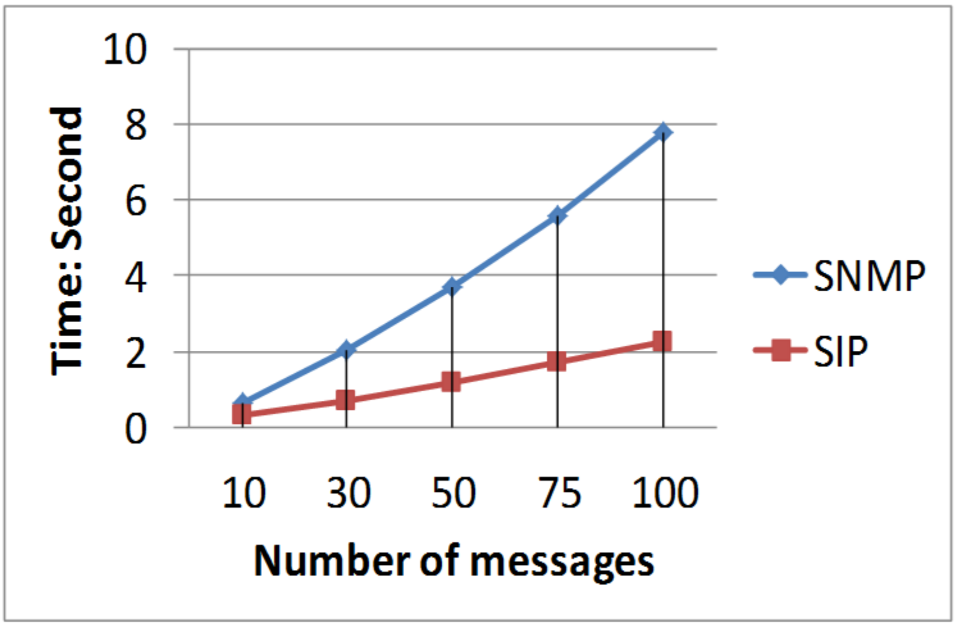


Figure 27 Transmission time required for messages

As the figures indicate, the time is proportional to the number of messages. Furthermore, SIP gets better performance than SNMP. However, as shown in Table 7 , the disadvantage of SIP is that it requires more memory space, which is a scarce resource on an embedded system.

Table 7 Footprints of SNMP and SIP

Footprint	SNMP	SIP
Standby	3,045 KB	6,699 KB
Running	3,045 KB	10,962 KB

According to the results from the above experiments, we can summarize the advantages and disadvantages of SNMP and SIP. We observed the experiments and we have the following preliminary conclusions: the small packet size of SNMP is its main advantage. However, longer processing time, hard understanding OID and NAT traversal problem are disadvantages of SNMP. SIP provides the following advantages: shorter processing time, plain text content and no NAT traversal problem. Furthermore, the American Standard Code for Information Interchange (ASCII) format of SIP packets makes SIP has more scalability. The most critical issue of SIP is its large packet size. Large packet size makes SIP have much overhead, decreasing the efficiency of network.

Chapter 5. Conclusion and Future Work

Without a general management mechanism, the development of Smart Grid will be slow down, even if the demand is strong. In this thesis, we provided two feasible management mechanisms in Smart Grid. To verify their performance, we established a Microgrid to provide experimental studies of SNMP and SIP. According to statistics in Chapter 4, we found that SIP has a better performance than SNMP. However, SIP consumes more memory resources. The small packet size and lower resources cost make SNMP to be appropriate to embedded systems, if memory size is a strict limitation.

In this thesis, we provide a case study of SIP and SNMP. According to the experiment results, we have following conclusions:

1. With lower resource cost and small packet size, SNMP is appropriate to Microgrids. Without IPv6 supports, remote management is difficult.
2. SIP is a clear choice for Smart Grid communications.

However, SIP delivery messages in plain text. Therefore, encryption is certainly needed. Furthermore, the default transport-layer protocol of SIP is the unreliable UDP, which is one issue for SIP in Smart Grid communications. Heavy overhead is another issue. In the future, reliable transport-layer protocol and header compression will be important research topics of SIP if it has to be successfully adopted for Smart Grid communications.

References

[1] S. Massoud Amin, Bruce F. Wollenberg, “Toward a Smart Grid – Power delivery for the 21st century”, IEEE Power & Energy Magazine, Vol. 3, No. 5, PP. 34-41, September/October 2005.

[2] David Amstell, “Smart Grid leads German revolution”, October 2009, [http://www.ngpowereu.com/news/smart-grid-revolution/]

[3] Office of the National Coordinator for Smart Grid Interoperability, “NIST framework and roadmap for smart grid interoperability standards, release 1.0,” NIST Special Publication 1108, pp. 1–145, Jan. 2010.

[4] Wei-Lun Wang, Quincy Wu, “Relay Placement Problem in Smart Grid Deployment”, *International Symposium on Leveraging Applications of Formal Methods (ISoLA 2010)*, Crete, Greece.

[5] Xiang Lu, Zhuo Lu, Wenye Wang, Jianfeng Ma, “On Network Performance Evaluation toward the Smart Grid: A Case Study of DNP3 over TCP/IP”, Global Communication Conference (GLOBECOM 2011), Houston, USA.

[6] Office of the Manager National Communications System, “Supervisory Control and Data Acquisition (SCADA) Systems”, National Communications System, October 2004.

[7] K. Curtis, “DNP3 protocol primer,” in DNP User Group, 2005.

[8] A. West, “Securing DNP3 and Modbus with AGA12-2J,” in2008 IEEE Power and Energy Society General Meeting (PES ’08), 2008.

[9] Microgrid, [http://galvinpower.org/microgrids]

[10] Tariq Samad, Brian Frank, "Leveraging the Web: A Universal Framework for Building Automation", American Control Conference (ACC 2007), New York City, USA.

[11] J. Case, M. Fedor, M. Schoffstall, J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, May 1990.

[12] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[13] J. DiAdamo, *SIP: The Clear Choice for Smart Grid Communications,* July 2009, [http://www.smartgridnews.com/artman/publish/commentary/SIP_The_Clear_Choice_for_Smart_Grid_Communications-604.html]

[14] K. Egevang, P. Francis, "The IP Network Address Translator", May, 1994.

[15] Choongul Park, Kitae Jeong, Sungil Kim, Youngseok Lee, "NAT Issues In the Remote Management of Home Network Devices", IEEE Network Magazine, Vol.22, Issue: 5, PP.48-55, September-October 2008.

[16] Demeter Robert, Sakany Istvan, "SNMP Protocol Based Home Automation System", Romanian educational and research network (RoEduNet) International Conference – Networking and Research (RoEduNet 2011), Iasi, Romania.

[17] J. S. Li, "Design and Application of ZigBee/Ethernet Gateway in Remote Control Systems", *Master Thesis*, *National Yunlin University of Science & Technology*, January, 2007.

[18] ZigBee Alliance, "ZigBee Specifications", ZigBee Document 053474r17, November 2009.

[19] Mong-Fong Horng, Mao-Hsiung Hung, Yi-Ting Chen, Jeng-Shyang Pan, Wen Huang, "A new approach based on XMPP and OSGi technology to home automation on Web",

Computer Information Systems and Industrial Management Applications (CISIM 2010), Kraków, Poland

[20] CSIPSimple, [http://code.google.com/p/csipsimple/]

[21] Y, Matsumoto, SNMP Management Service, [https://play.google.com/store/apps/details?id=jp.ymatsumoto.management&feature=search_result#?t=W251bGwsMSwxLDEsImpwLnltYXRzdW1vdG8ubWFuYWdlbWVudCJd]

[22] OpenSIPS, [http://www.opensips.org/]

[23] J. Postel, "Internet Protocol," RFC 791, September 1981.

[24] J. Reynolds, J. Postel, "Assigned Numbers", RFC 1700, October 1994.

[25] APNIC (Asia-Pacific Network Information Center), "IPv4 Exaustion details", [http://www.apnic.net/community/ipv4-exhaustion/ipv4-exhaustion-details]

[26] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC2460, December 1998.

[27] R. Droms, Ed. J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) ," RFC3315, July 2003.

[28] S. Thomson, T. Narten, T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC4862, September 2007.

[29] N. Moore, "Optimistic Duplicate Address Detection (DAD) for IPv6," RFC4429, April 2006.

[30] K. McCloghrie, M. Rose, "Management Information Base for Network Management of TCP/IP based internets: MIB-II", RFC1213, March 1991.

[31] Wen-Long Yang, "A study of SNMP-based Detection of ARP Attacks", Master Thesis, National Chi Nan University, August 2007.

[32] BusyBox: The Swiss Army Knife of Embedded Linux, [http://www.busybox.net/]

[33] Net-SNMP, [http://net-snmp.sourceforge.net/]

[34] Sofia-SIP, [http://sofia-sip.sourceforge.net/]

[35] Wireshark, the world's foremost network protocol analyzer. [http://www.wireshark.org/]

Appendix: Procedures in Preparing the Experimental Environment

A.1 Kernel configuration SOP of DMA-2440

1. Get the kernel source of LINUX 2.6.24 from http://ms11.voip.edu.tw/~memphis/DMA2440L_SOP/dma-linux-2.6.24_rel_1.0_2440L_090603.tar.bz2, then type the following commands

```
#tar -zxvf linux-2.6.24.tar.gz
#cd dma-linux-2.6.24
```

2. Modify Makefile

```
#vi Makefile
```

Modify the following lines

```
ARCH    ?=arm
CROSS_COMPILE    ?=/usr/local/arm/4.0.3/bin/arm-linux-
```

3. Get the cross compiler 4.0.3 from http://ms11.voip.edu.tw/~memphis/Download/cross_4.0.3.tar.bz2 and unzip it beneath /usr/local/arm

Please note that to cross-compile the kernel, version 4.0.3 is used for compiling kernel, while version 3.4.1 is used for file system. Version 3.4.1 can be downloaded from the URL specified in the next section.

4. Before using the cross-compiler, we need to export the path of cross-compiler. Type the following command:

```
#export PATH=/usr/local/arm/4.0.3/bin/:$PATH
```

5.Before compiling the kernel, make sure that your machine(CentOS) has already installed ncurses-devel, if not, type following command to install it

```
#yum install ncurses-devel
```

6.Type the command

```
#make menuconfig
```

7.Choose the module that you need, such as the following:

```
[*] Enable loadable module support --->
[*] Module unloading
[*] Automatic kernel module loading
System Type ---->
[*] S3C2410 DMA support
[*] Support ARM920T processor
S3C2410 Machines --->
[*] SMDK2410/A9M2410
S3C2440 Machines --->
[*] SMDK2440
[*] SMDK2440 with S3C2440 CPU module
File systems-
[*]Network File Systems----
<*>NFS file system support
[*]Provide NFSv3 client support
[*]Provide client support for the NFSv3 ACL protocol extension
[*] Root file system on NFS
Networking --->
[*] Networking support
Networking options --->
[*] TCP/IP networking
[*] IP: multicasting
[*] IP: kernel level autoconfiguration
[*] IP: DHCP support
[*] IP: BOOTP support
<*> IP: IPsec transport mode
<*> IP: IPsec tunnel mode
<*> IP: IPsec BEET mode
<*> INET: socket monitoring interface
The IPv6 protocol --->
[*] IPv6: Privacy Extensions support
[*] IPv6: Router Preference (RFC 4191) support
```

8.After selecting the module you need, you can make an image of kernel

```
#make zImage
```

After this step, you can find two files zImage and Image. This implies that the kernel file is ready. The file zImage is the compressed kernel file that we need.

A.2 File system configuration SOP of DMA-2440

1.Install ncurses-devel, if this has not been done.

```
#yum install ncurses-devel
```

2.Download busybox-1.11 from http://www.busybox.net/downloads/busybox-1.11.1.tar.bz2 and Cross Compiler 4.0.3 from http://ms11.voip.edu.tw/~memphis/Download/cross-4.0.3.tar.bz2, then install on your LINUX machine

3.Unzip BusyBox and Cross Compiler

```
#tar -jxvf busybox-1.11.1.tar.bz2
#cd busybox-1.11.1
```

4.`#vi Makefile`

Modify the following lines:

```
CROSS_COMPILE ?=/usr/local/arm/4.0.3/bin/arm-linux-
ARCH ?=arm
#cp cross-4.0.3.tar.bz2 /usr/local/arm
#tar -jxvf /usr/local/arm/cross-4.0.3.tar.bz2
#export PATH=/usr/local/arm/4.0.3/bin/
```

5.Configure the file system

```
#make menuconfig
```

6. Choose the module that you need, such as follows:

```
General configuration--->
Build options--->
[*] Build Busybox as a static binary (no shared libs)
[ ] Force NOMMU build
[*] Build with large file support (for accessing files > 2GB)
(/usr/local/arm/3.4.1/bin/arm-linux-) cross-compiler prefix
Debugging options--->
[*] Don't use /usr
Applets links(as soft-links) --->
(./_install) Busybox installation prefix
Installation options--->
Busybox library tuning--->
   [ ] Support version 2.2.x to 2.4.x Linux kernels
Miscellaneous utilities--->
```

```
    [ ] Inotifyd
[ ]taskset
```

Save the settings and exit the configuration.

7. Because there are some compatibility issues of different versions of cross-compilers, to prevent the compilation from being failed, we must add a statement "`#define ARPHRD_INFINIBAND 32`" in the following files :

busybox-1.11.1/networking/interface.c

busybox-1.11.1/networking/libiproute/ll_types.c

8.make busybox

```
#make && make install
```

9.We can found busybox tools beneath /_install, then we will create a file system

10. Go to the parent directory of busybox and create file system

```
#cd ..
#mkdir rootfs
#cd rootfs/
#mkdir bin dev etc lib proc sbin sys usr mnt tmp var
#cp -rfd ../busybox-1.11.1/_install/* .
#cp -rf ../busybox-1.11.1/examples/bootfloopy/etc/* ./etc
```

Edit following file:

```
#vi etc/profile
#etc/profile: system-wide .profile file for the Bourne shells
echo
echo -n "Processing /etc/profile…"
echo "Set Search library path"
LD_LIBRARY_PATH=/lib:/usr/lib
export LD_LIBRARY_PATH

#Set user path
echo "Set user path"
PATH=/bin:/sbin:/usr/bin:/usr/sbin
export PATH

#Set PS1
echo "Set PS1"
export PS1="[$USER@dma2440L ]\\$"
echo "Done"
```

```
echo
```

Save the file and exit. The following script is used for designating environment variables.

11.Edit inittab file

```
#vi etc/inittab
#This is run first except when booting
::sysinit:/etc/init.d/rcS
#Start an “askfirst” shell on the console
#::askfirst:-/bin/bash
::askfirst:/bin/sh
#Stuff to do when restarting the init process
::restart:/sbin/init
::askfirst:/bin/sh
#::once:/sbin/raja.sh
#::respawn:/sbin/iom
::once:/usr/etc/rc.local
#Stuff to do before rebooting
::ctrlaltdel:/sbin/reboot
::shutdown:/bin/umount -a –r
```

Save and exit

12.Edit etc/fstab

```
#vi etc/fstab
none           /proc   proc    defaults    0   0
none           /tmp    ramfs   defaults    0   0
mdev           /dev    ramfs   defaults    0   0
sysfs          /sys    sysfs   defaults    0   0
```

13.Edit etc/init.d/rcS

```
#vi etc/init.d/rcS
#! /bin/sh
echo "Processing etc/ini.d/rc.S"
echo " Mount all"

/bin/mount -a



/bin/mknod /dev/console c 5 1
/bin/mknod /dev/null c 1 3
```

```
/bin/mknod /dev/ttySAC0 c 204 64
/bin/mknod /dev/ttySAC1 c 204 65
/bin/mknod /dev/ttySAC2 c 204 66

echo "Starting mdevd..."
/bin/echo /sbin/mdev > proc/sys/kernel/hotplug
mdev -s
#/sbin/udevd --daemon
#/sbin/udevadm trigger

ln -s /dev/ts0 /dev/ts

echo "****************"
echo "RootFS by Cramfs, DMA2440"
echo
```

Save and exit

14. Create mdev.conf beneath etc directory

```
#touch mdev.conf
```

15. Login as super user to create console and null nodes beneath dev

```
#cd ../dev/
#mknod –m 600 console c 5 1
#mknod –m 666 null c 1 3
```

17.The file system configuration is ready. Before make CRAMFS, make sure that your machine has installed zlib-devel. If not, install it. Then type following command to create CRAMFS file

```
#./mkcramfs rootfs/ root_dma.cramfs
```

A.3 Cross-compile Net-SNMP for the ARM architecture

1. Download the source file of Net-SNMP version 5.6.1.1 from http://sourceforge.net/projects/net-snmp/files/latest/download?source=directory

2. Unzip it

```
#tar –xvf net-snmp-5.6.1.1.tar.gz
```

3. Create the Makefile of net-snmp-5.6.1.1

```
#cd net-snmp-5.6.1.1
# ./configure CC=/usr/local/arm/4.0.3/bin/arm-linux-gcc
--prefix=/home/memphis/snmp --build=i386-linux
--host=arm-linux --enable-mini-agent --with-endianness=little
--disable-manuals --disable-ucd-snmp-compatibility
--enable-as-needed --disable-embedded-perl
--without-perl-modules --disable-snmptrapd-subagent
--disable-applications --disable-scripts
--enable-ipv6 --with-mib-modules="mibII/ipv6"
--with-tranports="UDPIPv6 TCPIPv6"
# make
# make install
```

4. Copy the libraries of net-snmp to the embedded file system

```
#cp /usr/local/lib/libnetsmp* rootfs/lib/
```

5. Net-SNMP porting is completed.

A.4 Cross-compile Sofia-SIP for the ARM architecture

1. Download the source file of Sofia-SIP-1.12.11 from http://sourceforge.net/projects/sofia-sip/files/sofia-sip/1.12.11/sofia-sip-1.12.11.tar.gz/download

2. Unzip it

```
#tar –xvf sofia-sip-1.12.11.tar.gz
```

3.Create the Makefile of sofia-sip

```
#./configure --host=arm-linux
CC=/usr/local/arm/4.0.3/bin/arm-linux-gcc
```

4. Copy the libraries of sofia-sip to the embedded file system

```
#cp /usr/local/lib/libsofia-sip-ua* rootfs/lib/
```

5. Sofia-SIP porting is completed.