ภาคผนวก

ภาคผนวก ก : การออกแบบและพัฒนาโปรแกรมประยุกต์เพื่อใช้ในการทดสอบ

จากระบบ SIP/MIPv6 ที่ได้ออกแบบขึ้น สามารถแบ่งองค์ประกอบสำคัญออกเป็น 2 ส่วนหลักด้วยกันได้แก่

1. IP Telephony Application
2. MIPv6 Component

โดย IP Telephony Application เป็นโปรแกรมประยุกต์ที่ถูกใช้ในทุกๆโหนดในระบบ SIP/MIPv6 ได้แก่ SIP Server และ SIP Client เพื่อใช้งานและรับผิดชอบการทำงานบนระบบ IP Telephony ในขณะที่ MIPv6 Component จะถูกติดตั้งลงบนทุกๆโหนดเช่นเดียวกัน แต่ทำหน้าที่รับผิดชอบการส่งข้อมูลแต่ละแพ็กเก็ตให้ถึงปลายทางได้อย่างถูกต้องเมื่อมีการเคลื่อนย้ายหรือเปลี่ยน subnet

IP Telephony Application

การพัฒนาโปรแกรมประยุกต์ในส่วนของ IP Telephony Application นี้ได้เลือกใช้ภาษา Java ในการพัฒนาทั้งหมด เนื่องจากภาษา Java มีคุณลักษณะเด่นในการทำงานอิสระต่อระบบปฏิบัติการเป็นผลให้โปรแกรมซึ่งพัฒนาขึ้นสามารถทำงานบนระบบปฏิบัติการ Windows หรือ Linux ได้ และมีความยืดหยุ่นในการประยุกต์ใช้งานต่อไปในอนาคต สำหรับการทำงานของ IP Telephony Application สามารถแบ่งการทำงานออกได้ดังนี้

Control Signal Module

ส่วนการทำงานของ Control Signal Module เป็นส่วนการทำงานในการสร้างสัญญาณควบคุมต่างๆ ซึ่งได้แก่ การร้องขอการติดต่อ, การแลกเปลี่ยนข้อมูลสำคัญต่างๆ เช่น มาตรฐานการเข้ารหัสข้อมูลพหุสื่อ, ความสามารถในการรับส่งข้อมูลพหุสื่อของแต่ละคู่สายการติดต่อ เป็นต้น โดยสัญญาณที่ถูกนำมาพัฒนาใช้ในระบบนี้คือ โพรโตคอล SIP (Session Initiation Protocol) และ SDP (Session Description Protocol) ซึ่งการทำงานของโพรโตคอล SIP จะเป็นสัญญาณหลักในการทำงาน ในขณะที่โพรโตคอล SDP ถูกใช้ร่วมกับโพรโตคอล SIP เพื่อใช้ในการบอกรายละเอียดต่างๆของความสามารถในการรับส่งข้อมูลพหุสื่อ ทั้ง 2 โพรโตคอลนี้ถูกพัฒนาขึ้นในรูปแบบของ

API (Application Programming Interface) เพื่อสะดวกต่อการปรับปรุงและใช้งานต่อไป ซึ่งถูกเรียกว่า cnrsipapi

CNR SIP API

คุณสมบัติและความสามารถของ cnrsipapi ที่พัฒนาขึ้น มีดังนี้

- ใช้ในการร้องขอการติดต่อ, แลกเปลี่ยนความสามารถในการรับส่งข้อมูลพหุสื่อ และยกเลิกการติดต่อ
- สัญญาณทั้งหมดที่ใช้เป็นไปตามมาตรฐานของโพรโตคอล SIP อ้างอิงตาม RFC 3261และโพรโตคอล SDP อ้างอิงตาม RFC 2327 และ RFC 3266
- สนับสนุนการทำงานผ่านทางโพรโตคอล TCP และ UDP
- สนับสนุนการทำงานด้วยไอพีรุ่นที่ 4 และไอพีรุ่นที่ 6

Media Module

ส่วนการทำงานของ Media Module เป็นส่วนการทำงานเกี่ยวกับการรับส่งข้อมูลพหุสื่อ ได้แก่ ข้อมูลภาพ และเสียง ผ่านทางโพรโตคอล RTP (Real – Time Transport Protocol) ซึ่งพัฒนาโมดูลนี้ขึ้นมาด้วย JMF API (Java Media Framework API) ซึ่งเป็น API ที่เป็นที่นิยมใช้งานในการรับส่งข้อมูลพหุสื่อด้วยภาษา Java

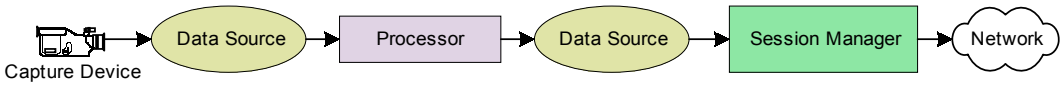
JMF (Java Media Framework)

JMF เป็น API ที่ใช้ในการส่งข้อมูลมัลติพหุสื่อที่เป็นข้อมูลสัญญาณภาพหรือสัญญาณเสียง ผ่านระบบเครือข่ายคอมพิวเตอร์ โดยอาจจะเป็นข้อมูลจากไฟล์ หรือข้อมูลจากอุปกรณ์จับภาพก็ได้ รวมถึงโมดูลต่างๆ ในการทำงานเกี่ยวกับพหุสื่อเช่น การรับข้อมูลเสียง รูปแบบการบีบอัดข้อมูลเสียง โดยสามารถกำหนดรูปแบบในการส่งข้อมูล รวมถึงมีโปรโตคอลที่ใช้ในการส่งข้อมูล ได้แก่ RTP ที่ใช้ในการส่งข้อมูลในลักษณะทันเวลาผ่านระบบเครือข่าย และโปรโตคอล RTCP ที่ใช้ในการควบคุมการรับส่งข้อมูลมัลติพหุสื่อ ผ่านระบบเครือข่าย สำหรับรูปแบบการเข้ารหัสข้อมูลพหุสื่อที่ JMF สนับสนุนและสามารถใช้งานในการรับ-ส่งข้อมูลเสียงได้แก่ PCMU, G.723 และ GSM สำหรับข้อมูลภาพได้แก่ H.263 และ JPEG

การพัฒนา Media Module นั้นประกอบด้วยคลาสหรือโมดูลย่อยที่สำคัญดังนี้

Transmitter Class

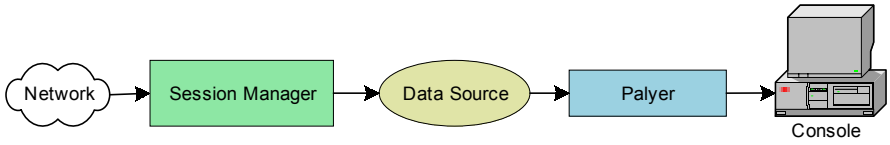
เป็นคลาสที่ทำหน้าที่ในการแปลงสัญญาณภาพจากกล้องและไมโครโฟน เป็นข้อมูลที่เรียกว่า Data Source แล้วทำการประมวลผลเพื่อทำการบีบอัดข้อมูลตามมาตรฐานที่เลือกกลายเป็น Data Source ที่ผ่านการบีบอัดข้อมูลแล้ว หลังจากนั้นจะทำการแบ่งข้อมูลออกเป็น session ต่างๆ แล้วส่งข้อมูลผ่านระบบเครือข่ายต่อไปยังเป้าหมายตามที่อยู่และพอร์ตที่กำหนดซึ่งที่อยู่ที่ทำการส่งอาจเป็นหมายเลขไอพี หรือเป็น Multicast Address ก็ได้ขึ้นอยู่กับการที่ผู้ใช้ทำการระบุ ซึ่งถ้าเป็นการระบุไปยังหมายเลขไอพีทั่วไป จะเป็นการส่งข้อมูล แบบ unicast ไปยังผู้ใช้เป้าหมาย แต่ถ้าทำการระบุหมายเลขไอพี ไปยัง Multicast Address จะเป็นการส่งข้อมูลแบบ multicast ไปยังสมาชิกในกลุ่มไอพี ซึ่งขั้นตอนการทำงานของคลาส Transmitter สามารถอธิบายได้ดังแสดงในรูปที่ 50



รูปที่ 50 ขั้นตอนการทำงานของคลาส Transmitter

Receiver Class

เป็นคลาสที่ทำหน้าที่ในการรับข้อมูลมัลติพหุสื่อสตรีมที่ส่งมาจากผู้ส่งโดยตรงหรือจาก Multicast Address โดยในขั้นแรกจะทำการรับข้อมูลที่ส่งมาจากระบบเครือข่าย ซึ่ง RTP Session Manager จะทำการแยกข้อมูลแต่ละสตรีม ที่ส่งเข้ามาจากระบบเครือข่าย หลังจากนั้นจะทำการประมวลผลเพื่อทำการสร้างตัวเล่นที่เหมาะสมกับข้อมูลมัลติพหุสื่อที่รับเข้ามาแล้วทำการแสดงเป็นข้อมูลสัญญาณภาพออกทางหน้าจอ หรือเป็นข้อมูลสัญญาณเสียงออกแสดงทางลำโพงก็ได้ ซึ่งจากการที่ RTP Manager สามารถทำการแยกข้อมูลของแต่ละสตรีมได้ ทำให้เมื่อมีการส่งพหุสื่อสตรีมมาจาก Multicast Address จะสามารถทำการแยกข้อมูลแต่ละสตรีมตามจำนวนผู้ใช้ที่ทำการส่งข้อมูลพหุสื่อสตรีมมายัง Multicast Address นี้ทำให้สามารถทำการแสดงข้อมูลสัญญาณภาพและสัญญาณเสียงของแต่ละผู้ใช้ที่ทำการส่งข้อมูลพหุสื่อสตรีมได้ ดังแสดงในรูปที่ 51



รูปที่ 51 ขั้นตอนการทำงานของคลาส Receiver

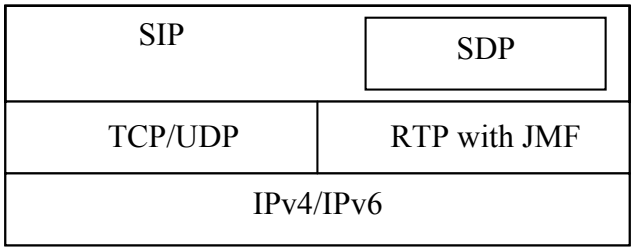
ซึ่งเราสามารถทำการควบคุมการทำงานของ Player ได้ โดยใช้ตัว Controller ต่างๆ ซึ่ง JMF ทำไว้ให้ ได้แก่

- Visual Component ทำหน้าที่ควบคุมเกี่ยวการแสดงผลของข้อมูลภาพ
- Gain Control ทำหน้าที่ควบคุมเกี่ยวกับการแสดงผลของข้อมูลเสียง
- Control Component ทำการควบคุมการเล่นของข้อมูลพหุสื่อซึ่งเมื่อทำการแสดงผลข้อมูลพหุสื่อนั้นๆ จะมีตัว Clock ขึ้นมาควบคุมการเล่น

JMF supporting IPv6

เนื่องจากเดิมที JMF สามารถทำงานด้วยมาตรฐาน IPv4 เพียงเท่านั้น แต่เนื่องจากปัจจุบันบริษัท Sun Microsystems ได้พัฒนาภาษา Java ให้มีการทำงานสนับสนุน IPv6 ใน JDK (Java Development Kit) version 1.4 ดังนั้นจึงทำการแก้ไขการทำงานในส่วนของการรับและส่งข้อมูลพหุสื่อ โดยการเปลี่ยนแปลงแก้ไขการทำงานของ Socket ต่างๆให้สนับสนุน IPv6 โดยใช้ Class Inet6Address ในการทำงานเกี่ยวกับ IPv6

Framework ของ IP Telephony Application สามารถสรุปได้ดังนี้



รูปที่ 52 แสดง IP Telephony Application Framework

จากรูปที่ 52 แสดงภาพโครงสร้างของ IP Telephony Application ที่ประกอบด้วยการทำงานของโพรโตคอลในแต่ละ layer แตกต่างกัน เริ่มจากระดับชั้น Application ประกอบด้วยการทำงานของโพรโตคอล SIP ซึ่งมีการทำงานร่วมกับโพรโตคอล SDP เพื่อใช้ในการส่งสัญญาณควบคุม (Control Signaling) ต่อมาในระดับชั้นTransport ในส่วนของการส่งสัญญาณควบคุมสามารถทำงานเหนือโพรโตคอล TCP และ UDP แต่สำหรับการส่งข้อมูลพหุสื่อด้วย JMF นั้นจะใช้โพรโตคอล RTP ซึ่งมีการทำงานร่วมกับโพรโตคอล UDP ในระดับชั้นต่อมาสามารถมีการทำงานด้วยไอพีรุ่นที่ 4 และไอพีรุ่นที่ 6

MIPv6 Component

แบ่งออกเป็น 2 โหนดหลัก ได้แก่ Home Agent ซึ่งทำหน้าที่เป็น router โดยรับผิดชอบการรับส่งข้อมูลให้ไปถึงโหนดปลายทางได้ถูกต้อง และ MIPv6 Client Component ซึ่งได้แก่ Mobile Node และรวมถึง client ในระบบ โดยทั้ง 2 โหนดนี้จำเป็นต้องมีการติดตั้งหน่วยการทำงานของ MIPv6 ทั้งสิ้น แต่มีการกำหนดค่า (Configuration) แตกต่างกัน ซึ่งในที่นี้จะทำการติดตั้ง MIPL (Mobile IPv6 for Linux) บนทุกๆโหนดในระบบ SIP/MIPv6

MIPL

เป็นการพัฒนาหน่วยการทำงานเพื่อให้สนับสนุนการทำ Mobile IP บนไอพีรุ่นที่ 6 อ้างอิงตาม Internet Draft (draft-ietfmobileip-ipv6-15.txt) สำหรับ MIPL ถูกพัฒนาเริ่มขึ้นในโครงการ HUT Software โดย Sami Kivisaari, Niklas Kämpe, Juha Mynttinen, Toni Nykänen, Henrik Petander และ Antti Tuominen และถูกพัฒนาอย่างต่อเนื่องโดย HUT Telecommunications and Multimedia Lab ซึ่งมี Henrik Petander และ Antti Tuominen เป็นผู้ดูแลในขณะนี้ ซึ่งมีการทำงานอยู่บนระบบปฏิบัติการ Linux โดยสนับสนุนการทำงานบน Kernel รุ่น 2.4.20 ขึ้นไป สำหรับ MIPL รุ่นใหม่ล่าสุดคือ mipv6-0.9.5.1-v2.4.20

ดังนั้นในการพัฒนาระบบ SIP/MIPv6 ขึ้นจึงเลือกติดตั้งระบบปฏิบัติการ Linux Redhat 9.0 ซึ่งมี Kernel รุ่น 2.4.20 และเป็นรุ่นใหม่ล่าสุดของระบบปฏิบัติการ Linux จึงสามารถนำมาติดตั้ง MIPL ได้ทันทีโดยไม่ต้อง compile kernel ใหม่ โดยเมื่อติดตั้ง MIPL ในแต่ละโหนดแล้วจะมีการกำหนดค่าให้แต่ละโหนดแตกต่างกัน ดังต่อไปนี้

Home Agent Configuration

สำหรับ Home Agent จำเป็นต้องมีการติดตั้ง radvd (Router Advertisement Daemon) เพื่อให้ทำหน้าที่เป็น IPv6 router ซึ่งจะทำหน้าที่ส่งสัญญาณ Router Advertisement ซึ่งอ้างอิงตาม RFC 2461 ไปยัง local Ethernet LAN ใน subnet ที่รับผิดชอบหรือในกรณีที่ได้รับสัญญาณ Router Solicitation จาก client เมื่อเข้ามาอยู่ในระบบนั้น โดยมีการกำหนดค่าดังต่อไปนี้ ซึ่งเป็นตัวอย่างการกำหนดค่า prefix ที่ต้องการใช้ภายในระบบนี้

```
interface eth0
{
AdvSendAdvert on;
MinRtrAdvInterval 3;
MaxRtrAdvInterval 10;
AdvHomeAgentFlag on;

        prefix 3ffe:b80:1e99:1::/64
        {AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
        };

};
```

รูปที่ 53 แสดงตัวอย่างการกำหนดค่าให้แก่ radvd

จากรูปที่ 53 แสดงการกำหนดค่าให้แก่ radvd ซึ่งการแก้ไขค่าภายใน /etc/radvd.conf หลังจากกำหนดค่าดังกล่าวแล้ว จำเป็นต้องทำการกำหนดค่าให้แก่ MIPL เพื่อให้ทำหน้าที่เป็น Home Agent โดยการกำหนดค่าให้ MIPL นั้นสามารถเลือกการทำงานได้อย่างใดอย่างหนึ่งระหว่าง Home Agent กับ Mobile Node หรือ MIPv6 Client เท่านั้น ซึ่งสำหรับการกำหนดค่าเป็น Home Agent ทำได้โดยการกำหนดค่า FUNCTIONALITY=ha ซึ่งอยู่ใน /etc/sysconfig/network-mip6.conf

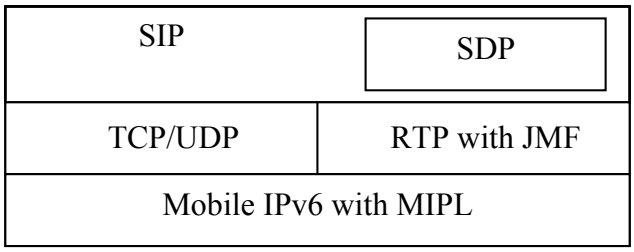
MIPv6 Client Configuration

สำหรับการกำหนดค่านั้นสามารถทำได้เพียงกำหนดค่าใน /etc/sysconfig/network-mip6.conf ที่ FUNCTIONALITY=mn และทำการกำหนดค่าเกี่ยวกับ Home Agent ที่รับผิดชอบด้วยพารามิเตอร์ HOMEADDRESS และ HOMEAGENT ซึ่งจากตัวอย่างในรูปที่ 53 ค่าที่กำหนดเป็นไปดังนี้

HOMEADDRESS=3ffe:b80:1e99:2::2/64

HOMEAGENT=3ffe:b80:1e99:2::1/64

SIP/MIPv6 Framework



รูปที่ 54 แสดง Framework ของทุกโหนด ในระบบ SIP/MIPv6

จากรูปที่ 54 แสดงภาพรวมของแต่ละโหนด ซึ่งมีการทำงานร่วมกันระหว่างมาตรฐานต่างๆกัน ในระบบ SIP/MIPv6 ดังนั้นในระบบ SIP/MIPv6 ที่พัฒนาขึ้นจะประกอบด้วยโหนดต่างๆ ดังต่อไปนี้

SS/HA – SIP Server/Home Agent มีการทำงานของ SIP Server Application ร่วมกับการทำงานของ MIPL เมื่อกำหนดการทำงานเป็น Home Agent

สำหรับส่วนของ SIP Server Application มีการทำงานเป็นไปดังนี้

- มีโหมดการทำงาน 3 โหมดด้วยกัน ได้แก่ Registrar Server, Proxy Server และ Redirect Server
- รองรับการทำงานด้วยโพรโตคอล TCP และ UDP
- สนับสนุนการเชื่อมต่อด้วยไอพีรุ่นที่ 4 และไอพีรุ่นที่ 6

Mobile Node/ Correspondent Node มีการทำงานของ SIP Client Application พัฒนาด้วย cnrsipapi ซึ่งทำงานร่วมกับการทำงานของ MIPL เมื่อกำหนดการทำงานเป็น Mobile Node และมีความสามารถในการส่งข้อมูลภาพและเสียงด้วย JMF

ภาคผนวก ข : วิธีการวัดค่าเวลาหน่วง

ในหัวข้อนี้จะอธิบายที่มาของค่าเวลาหน่วงซึ่งมีการนำเสนอในบทที่ 5 ซึ่งแบ่งออกเป็นหัวข้อย่อยตามค่าเวลาหน่วงทั้งหมดที่มีการวัด ดังต่อไปนี้

วิธีการวัดค่า Movement Detection Delay (D_{MD})

สำหรับการตรวจจับเวลา D_{MD} นั้นจำต้องติดต่อไปยังอุปกรณ์เชื่อมต่อระบบเครือข่าย ในที่นี้คือ Wireless Lan Card ซึ่งบนระบบปฏิบัติการ Linux มีอินเตอร์เฟตให้ผู้ใช้สามารถเข้าถึงเพื่อดึงข้อมูลเกี่ยวกับอุปกรณ์นั้นๆได้ โดยปกติ Wireless Lan Card จะสามารถเชื่อมต่อไปยัง Access Point ได้เพียงตัวเดียวเท่านั้น ดังนั้นในการจับเวลา D_{MD} จึงทำการเขียนโปรแกรมติดต่อกับอินเตอร์เฟตดังกล่าวโดยเริ่มนับตั้งแต่ขาดหายการเชื่อมต่อจาก Access Point ตัวปัจจุบันจนกระทั่งสามารถเชื่อมต่อไปยัง Access Point ตัวใหม่ได้ ซึ่งเมื่อดึงข้อมูลขึ้นมาจากอินเตอร์เฟตในขณะที่ Wireless Lan Card ทำการค้นหาและเลือก Access Point อยู่นั้นจะได้ผลดังรูปที่ 55 และเมื่อ Wireless Lan Card ทำการเชื่อมต่อกับ Access Point ตัวใหม่ได้สำเร็จ จะดึงค่าข้อมูลได้ดังรูปที่ 56

รูปที่ 55 แสดงค่าข้อมูลจาก Wireless Lan Card ขณะทำการค้นหา Accees Point ใหม่

วิธีการวัดค่า Router Advertisement Delay (D_{RA})

ก่อนการวัดค่า D_{MD} ได้ทำการรันโปรแกรม Ethereal เพื่อรอรับสัญญาณ Router Advertisement จาก Router ในระบบเครือข่ายใหม่ ในขั้นตอนการวัด D_{MD} เมื่อสามารถดึงค่าข้อมูลได้จาก Wireless Lan Card แล้วจึงทำการบันทึกค่าเวลาปัจจุบันทันที จากนั้นโปรแกรม Ethereal จะสามารถจับสัญญาณ Router Advertisement ได้ซึ่งจะแสดงเวลาปัจจุบันที่ได้รับสัญญาณนี้ ดังรูปที่ 57

รูปที่ 57 แสดงการจับสัญญาณ Router Advertisement ด้วยโปรแกรม Ethereal

ด้วยค่าเวลาที่ได้จากโปรแกรม Ethereal ลบด้วยค่าเวลาที่ได้จากการสิ้นสุดค่าจับเวลาของ D_{MD} ทำให้ได้ค่าเวลาหน่วง D_{RA} ซึ่งบ่งบอกถึงเวลาในการรอรับสัญญาณ Router Advertisement

วิธีการวัดค่า Binding Update Delay สำหรับโปรโตคอล SIP ($D_{Binding}$)

การวัดค่าในส่วนนี้ต่อเนื่องจากการวัดค่า D_{RA} โดยจับเวลาตั้งแต่ Mobile Node ทำการส่งสัญญาณ re-INVITE ไปยังคู่สายการติดต่อรอจนกระทั่งได้รับสัญญาณ 200 OK ตอบกลับมา โดยค่าเวลาและสัญญาณสามารถตรวจจับได้ด้วยโปรแกรม Ethereal และทำการกำหนดให้โปรแกรม Ethereal

แสดงค่าเวลาแต่ละแพ็กเกตนับต่อจากเวลาของแพ็กเกตก่อนหน้า เป็นผลให้ค่าเวลาซึ่งแสดงในลำดับที่ได้รับแพ็กเกตของสัญญาณ 200 OK เป็นค่าเวลาหน่วงของการทำ Binding Update ด้วยโปรโตคอล SIP

รูปที่ 58 แสดงการจับสัญญาณ re-INVITE ด้วยโปรแกรม Ethereal

วิธีการวัดค่า Processing Delay สำหรับการทำ Hand Over ด้วยโปรโตคอล SIP ($D_{Processing}$)

การวัดค่าในส่วนนี้เริ่มแรกวัดจากโปรแกรม Ethereal โดยทำการจับสัญญาณต่อเนื่องหลังจากการทำ Binding Update ด้วยโปรโตคอล SIP เสร็จสิ้น ซึ่งพบว่าค่าเวลาหน่วงมีค่าประมาณ 100 ms ดังรูปที่ 59