การเพิ่มประสิทธิภาพของมัลติมีเดียวิดีโอคอนเฟอร์เรนซ์จากโอเพ่นเอ็มซียู

PERFORMANCE ENHANCEMENT OF MULTIPOINT VIDEO CONFERENCE BASED ON OPEN MCU

ชัชวาลย์ ไตรเพิ่ม

CHATCHAWAL TIREPORM

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2548

ISBN 974-15-1943-5

PERFORMANCE ENHANCEMENT OF MULTIPOINT VIDEO CONFERENCE BASED ON OPEN MCU

CHATCHAWAL TIREPORM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2005

ISBN 974-15-1943-5

หัวข้อวิทยานิพนธ์	**การเพิ่มประสิทธิภาพของมัลติมีเดียวิดีโอคอนเฟอร์เรนซ์จากโอเพ่นเอ็มซียู**
นักศึกษา	นายชัชวาลย์ ไตรเพิ่ม
รหัสประจำตัว	46061015
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2548
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.ดร.สุรินทร์ กิตติธรกุล

บทคัดย่อ

งานวิจัยนี้อธิบายการเพิ่มประสิทธิภาพของมัลติมีเดียมัลติพอยน์คอนเฟอร์เรนซ์คอนโทรลยูนิท ตามมาตรฐาน ITU H.323 ผ่านเครือข่ายอินเตอร์เน็ท ด้วยการแปลงโครงสร้างและอัลกอริธึมเพื่อให้รองรับจำนวนผู้เข้าประชุมได้มากขึ้น มัลติพอยท์คอนโทรลยูนิทที่ใช้ในการทดลองเป็นชุดโปรแกรมแบบโอเพ่นซอร์ส ชื่อ OpenMCU ทำงานอยู่บนระบบปฏิบัติการไมโครซอฟต์วินโดวส์และลีนุกซ์ เราสามารถวัดประสิทธิภาพที่เพิ่มขึ้นของ OpenMCU บนระบบปฏิบัติการวินโดวส์โดยเปรียบเทียบการใช้อัลกอริธึมแบบใหม่กับอัลกอริธึมแบบเก่าได้จากความสัมพันธ์ของภาระโหลดเฉลี่ยของซีพียูกับจำนวนผู้ใช้ ซึ่งในกรณีเฉลี่ยอัลกอริธึมแบบใหม่สามารถรองรับจำนวนผู้ใช้ในหนึ่งห้องประชุมได้ประมาณ 185% ของอัลกอริธึมแบบเก่า นอกจากนี้หลังจากการเพิ่มประสิทธิภาพแล้ว เราได้นำโครงสร้างการทำงานของ OpenMCU มาออกแบบและสร้าง SIP MCU เพื่อรองรับการทำมัลติพอยน์คอนเฟอร์เรนซ์ในโปรโตคอลซิพอีกด้วย

Thesis Title	Performance Enhancement of Multipoint Video Conference Based on Open MCU
Student	Mr.Chatchawal Tireporm
Student ID	46061015
Degree	Master of Engineering
Programme	Computer Engineering
Year	2005
Thesis Advisor	Asst.Prof.Dr.Surin Kittitornkun

ABSTRACT

In this thesis, we present a number of enhancement methods for a new high-performance Multimedia Multipoint Control Unit (MCU). The new MCU is based on OpenMCU, the Opensource ITU H.323 MCU project, for audiovisual communication over the internet. Using our new data structures and algorithms, The performance of the OpenMCU can be enhanced further. Performance comparison between the modified OpenMCU and the unmodified OpenMCU shows that in each video conference room, the modified one is able to support the number of users about 185% of the unmodified one. After that, we use the enhanced data structures and algorithms of the OpenMCU for designing and implementing a SIP MCU in order to support multipoint conferencing in SIP protocol.

กิตติกรรมประกาศ

คุณความดีอันใดที่บังเกิดจากวิทยานิพนธ์ฉบับนี้ ขอมอบแด่บิดาและมารดาของผู้วิจัย ผู้ที่คอยห่วงใย เข้าใจ และให้การสนับสนุนในการศึกษามาโดยตลอด

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงลงได้ด้วยดี โดยได้รับความกรุณาจาก ผศ.ดร.สุรินทร์ กิตติธรกุล อาจารย์ที่ปรึกษา ที่ได้ช่วยเหลือในการให้คำแนะนำ ความรู้ทางทฤษฎีต่างๆที่ใช้ และชี้แนะแนวทางในการแก้ปัญหาต่างๆอย่างทุ่มเทรวมทั้งฝึกฝนผู้วิจัยให้มีความสามารถในการทำวิจัยและพัฒนาได้อย่างมีประสิทธิภาพ ผู้วิจัยรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณกรรมการสอบวิทยานิพนธ์ทุกท่านที่ได้กรุณาให้คำแนะนำในทุกๆเรื่อง ทั้งวิธีแก้ปัญหาที่เกิดขึ้นในการทำวิทยานิพนธ์และมุมมองในเชิงวิศวกรรมอื่นๆซึ่งช่วยให้ผู้วิจัยมีวิสัยทัศน์ที่กว้างไกลขึ้น

ขอขอบคุณบัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้การสนับสนุนการทำวิทยานิพนธ์นี้

ขอบคุณพี่ๆเพื่อนๆ และน้องๆนักศึกษาทุกคนในห้องวิจัย Mobile Computing Lab รวมทั้งเพื่อนๆหลายๆคนในอินเตอร์เน็ตที่ช่วยเหลือให้คำแนะนำต่างๆ และให้กำลังใจแก่ผู้วิจัยตลอดมา

ชัชวาลย์ ไตรเพิ่ม

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....I

บทคัดย่อภาษาอังกฤษ.....II

กิตติกรรมประกาศ.....III

สารบัญ.....IV

สารบัญตาราง.....VII

สารบัญรูป.....VIII

บทที่ 1 บทนำ.....1

1.1 ความเป็นมาและความสำคัญของปัญหา.....1

1.2 วัตถุประสงค์ของการศึกษา.....2

1.3 สมมุติฐานของการศึกษา.....2

1.4 ขอบเขตของงานวิจัย.....2

1.5 ขั้นตอนการศึกษา.....3

1.6 ข้อตกลงเบื้องต้น.....3

1.7 ข้อจำกัดของการศึกษา.....3

1.8 รายละเอียดของวิทยานิพนธ์.....4

บทที่ 2 โปรโตคอลสำหรับ VoIP และ OpenMCU.....5

2.1 กล่าวนำ.....5

2.2 โปรโตคอล H.323.....7

2.2.1 สถาปัตยกรรมของ H.323.....8

2.2.2 เอนทิตี้และฟังก์ชันของ H.323.....9

2.3 โปรโตคอล SIP.....16

2.3.1 โปรโตคอลสแต็ก.....17

2.3.2 สถาปัตยกรรมและองค์ประกอบของซิพ.....18

2.3.3 ชื่อและแอดเดรส.....22

2.3.4 Locating Server.....23

2.3.5 Locate User.....23

สารบัญ (ต่อ)

หน้า

2.3.6 ความเชื่อถือได้.....23
2.3.7 ความสามารถในการขยาย.....24
2.3.8 แมสเสจเอสดีพี.....25
2.4 การเปรียบเทียบระหว่าง SIP และ H.323.....26
2.5 OpenMCU.....30
2.5.1 โครงสร้างข้อมูลและการทำงานของ OpenMCU.....30
2.5.2 รูปแบบการบีบอัดที่รองรับ.....34
2.6 งานวิจัยที่เกี่ยวข้อง.....34

บทที่ 3 การเพิ่มประสิทธิภาพของ OpenMCU และการสร้าง SIP MCU.....36
3.1 กล่าวนำ.....36
3.2 การเพิ่มประสิทธิภาพของ OpenMCU.....36
3.2.1 การเพิ่มประสิทธิภาพในส่วนออดิโอ.....36
3.2.2 การเพิ่มประสิทธิภาพในส่วนวิดีโอ.....43
3.2.3 การสร้าง H.323 เทอร์มินัล.....45
3.3 การสร้าง SIP MCU.....45
3.3.1 สถาปัตยกรรมของ SIP MCU.....45
3.3.2 การสร้าง Session ระหว่าง SIP MCU และ SIP UA.....47
3.3.3 การสร้าง SIP Terminal.....49
3.4 สรุปการเพิ่มประสิทธิภาพ.....49

บทที่ 4 การเปรียบเทียบและวิเคราะห์ประสิทธิภาพของ OpenMCU และ SIP MCU.....51
4.1 กล่าวนำ.....51
4.2 วิธีการทดลอง.....51
4.3 ผลการทดลองและบทวิเคราะห์.....54

สารบัญ (ต่อ)

หน้า

4.3.1 การเปรียบเทียบซีพียูโหลด....54
4.3.2 การเปรียบเทียบจำนวนหน่วยความจำ....59
4.3.3 การเปรียบเทียบขนาดของเน็ตเวิร์คทราฟฟิก....61
4.3.4 การเปรียบเทียบคุณภาพของวิดีโอโดยใช้สายตา....63
4.4 การเปรียบเทียบจำนวนผู้ใช้กับ MCU ในท้องตลาด....63
4.5 สรุปผลการเพิ่มประสิทธิภาพ....65

บทที่ 5 บทสรุป....66
5.1 สรุปงานวิจัยที่นำเสนอ....66
5.2 ปัญหาและอุปสรรค....68
5.3 แนวทางการพัฒนาต่อ....69

เอกสารอ้างอิง....70

ภาคผนวก....73
ภาคผนวก ก ไลบรารีต่างๆที่ใช้ในงานวิจัย....74
ภาคผนวก ข การซิงโครไนซ์เธรดบนระบบปฏิบัติการไมโครซอฟต์วินโดส์....76
ภาคผนวก ค เทคโนโลยี SIMD / MMX / SSE / SSE2....82
ภาคผนวก ง ผลงานวิจัยในระหว่างการศึกษาที่ได้รับการตีพิมพ์เผยแพร่....95

ประวัติผู้เขียน....96

สารบัญตาราง

ตารางที่ หน้า

2.1 สรุปการเปรียบเทียบการทำงานระหว่าง H.323 และ SIP.....27

2.2 สรุปการเปรียบระหว่าง H.323 และ SIP.....29

2.3 สรุปการเปรียบเทียบองค์ประกอบระหว่าง H.323 และ SIP.....29

3.1 สรุปการเพิ่มประสิทธิภาพ.....50

4.1 เปรียบเทียบจำนวนผู้ใช้ในกรณีประชุมแบบเสียงอย่างเดียวที่ 128 kbps.....64

4.2 เปรียบเทียบจำนวนผู้ใช้ในกรณีประชุมทั้งภาพและเสียงที่ 128 kbps.....64

ค.1 Data type ที่ถูกใช้ใน Intrinsics.....85

ค.2 ชุดคำสั่งของเทคโนโลยี MMX.....86

ค.3 ชุดคำสั่งของเทคโนโลยี SSE.....88

ค.4 ชุดคำสั่งของเทคโนโลยี SSE2.....91

สารบัญรูป (ต่อ)

รูปที่ หน้า

3.11 SIP Terminal ที่สร้างขึ้น....49

4.1 การทดลอง....51

4.2 โปรแกรม Performance....53

4.3 Task Manager....53

4.4 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนผู้ใช้ในการประชุม ทางเสียงเท่านั้นในหนึ่งห้อง....55

4.5 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนห้องประชุม ในกรณีมีผู้ใช้ 4 คนต่อห้อง ในการประชุมทางเสียงเท่านั้น....56

4.6 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนห้องประชุม ในกรณีมีผู้ใช้ 8 คนต่อห้อง ในการประชุมทางเสียงเท่านั้น....56

4.7 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนผู้ใช้ ในการประชุมทั้งภาพและเสียงในหนึ่งห้อง....57

4.8 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนห้องประชุม ในกรณีมีผู้ใช้ 4 คนต่อห้อง ในการประชุมทั้งภาพและเสียง....58

4.9 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนห้องประชุม ในกรณีมีผู้ใช้ 8 คนต่อห้อง ในการประชุมทั้งภาพและเสียง....59

4.10 กราฟความสัมพันธ์ระหว่างหน่วยความจำกับจำนวนผู้ใช้ใน 1 ห้อง....60

4.11 กราฟความสัมพันธ์ระหว่างขนาดของ Incoming Traffic กับจำนวนผู้ใช้ใน 1 ห้องประชุม....62

4.12 กราฟความสัมพันธ์ระหว่างขนาดของ Incoming Traffic กับจำนวนห้องในกรณีผู้ใช้ 8 คนต่อห้อง....62

4.13 เปรียบเทียบคุณภาพของวิดีโอโดยใช้สายตา (a) H.263 จาก Original OpenMCU และ (b) MPEG-4 จาก Enhanced OpenMCU และ SIP MCU....63

ข.1 Source Code ของ Readers-Writers Object บนระบบ win32....79

ค.1 MMX รีจิสเตอร์ในเทคโนโลยี MMX....82

ค.2 รีจิสเตอร์ในเทคโนโลยี SSE....83

ค.3 รีจิสเตอร์ในเทคโนโลยี SSE2....84

บทที่1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เทคโนโลยีอินเตอร์เน็ตได้เปิดศักราชใหม่ในการติดต่อสื่อสารสำหรับผู้คนทั่วโลกด้วยเครือข่ายที่เปรียบเสมือนใยแมงมุมทำให้การติดต่อสื่อสารของผู้คนเป็นไปด้วยความสะดวกรวดเร็วและเต็มไปด้วยประสิทธิภาพ อีกทั้งวิวัฒนาการในการใช้งานระบบก็ล้ำหน้าไปอย่างมากโดยมุ่งไปสู่การสื่อสารในแบบเวลาจริง (Real-time) ซึ่งเป็นหนทางเชื่อมต่อของเทคโนโลยีที่จะมีขึ้นมาในอนาคต ซึ่ง VoIP [1, 2, 3] คือหนึ่งในเทคโนโลยีที่กำลังมีบทบาทอย่างมากในการสื่อสารในปัจจุบัน โดย VoIP เป็นเทคโนโลยีซึ่งหลอมรวมการสื่อสารสัญญาณเสียงกับการสื่อสารด้วยสัญญาณข้อมูลเข้าไว้ด้วยกัน ทั้งนี้ก็เพื่อให้การส่งผ่านสัญญาณทั้งสองไปบนระบบเครือข่ายด้วยโปรโตคอลที่ใช้กันอย่างแพร่หลายในปัจจุบันอย่าง IP (Internet Telephone) ได้ ตัวอย่างของการใช้งานเทคโนโลยี VoIP ที่คุ้นเคยกันเป็นอย่างดี ก็คือ โปรแกรม MSN Messenger [4], Microsoft NetMeeting [5] ซึ่งสามารถพูดคุยกันผ่านระบบอินเตอร์เน็ตได้ จึงทำให้เราประหยัดค่าใช้จ่ายในการใช้งานโทรศัพท์อย่างมากมาย โดยเฉพาะอย่างยิ่งการโทรศัพท์ทางไกล ถ้าเราเปลี่ยนมาใช้งาน VoIP ก็เพียงเสียค่าใช้จ่ายในด้านการเชื่อมต่อกับเครือข่ายอินเตอร์เน็ตเท่านั้น

ในปัจจุบันโปรโตคอลสำหรับ VoIP มีอยู่ 2 มาตรฐานคือ H.323 [6, 7, 8] และ ซิพ (SIP : Session Initial Protocol) [9, 10, 11] โปรโตคอล H.323 เป็นโปรโตคอลที่พัฒนาโดย ITU-T (International Telecommunications Union-Telecommunications section) [12] ส่วนซิพถูกพัฒนาโดย IETF (Internet Engineering Task Force) [13] โปรโตคอลทั้งสองมีหน้าที่หลักในการสร้างสิ้นสุด และการเปลี่ยนแปลงรายละเอียดของการเรียก โดยโปรโตคอล H.323 และซิพเป็นโปรโตคอลในชั้นแอพพลิเคชัน (application layer) และใช้บริการของโปรโตคอลในชั้นที่ต่ำกว่า

H.323 Multipoint Control Unit (MCU) ทำหน้าที่เป็นมัลติมีเดียเซิร์ฟเวอร์ในการสนับสนุนการประชุมแบบหลายจุด (Multipoint Conference) ระหว่างเทอร์มินัล 3 เทอร์มินัลขึ้นไป โดย MCU ที่มีอยู่ในปัจจุบันเป็นฮาร์ดแวร์เสียส่วนใหญ่ ส่วนที่เป็นซอฟต์แวร์มีน้อยมาก ยิ่งที่เป็นโอเพนซอร์สที่ผู้ทำวิจัยค้นหาเจอมีอยู่เพียงตัวเดียวเท่านั้นคือ OpenMCU ซึ่งเป็นซอฟต์แวร์ตัวหนึ่งในชุดซอฟต์แวร์ OpenH323 [14] ซึ่งเป็นโอเพนซอร์ส

OpenMCU มีการทำงานส่วนใหญ่ที่มีคอมเพล็กซิตีเป็น $O(N^2)$ ซึ่งจะเห็นว่ามีประสิทธิภาพต่ำเมื่อรองรับผู้ใช้จำนวนมากๆ ซึ่งสาเหตุก็น่าจะเนื่องมาจากผู้ออกแบบ OpenMCU มุ่งเน้นในด้านให้โปรแกรมทำงานได้มากกว่าที่จะเน้นในเรื่องประสิทธิภาพของโปรแกรมที่จะรองรับผู้ใช้จำนวนมาก ปัญหาดังกล่าวได้ถูกหยิบยกมาวิเคราะห์แล้วใน [15] ซึ่งได้วิเคราะห์ว่าถ้าจะให้ OpenMCU

สามารถรองรับผู้ใช้ได้จำนวนมากๆจะต้องลดซีพียูโหลด (ซีพียูโหลด) และเน็ตเวิร์คแบนวิดท์ (เน็ตเวิร์คแบนวิดท์) ที่ต้องการโดยตัว OpenMCU ดังนั้นในวิทยานิพนธ์ชิ้นนี้จะมุ่งเน้นในการลดซีพียูโหลด และเน็ตเวิร์คแบนวิดท์ของตัว OpenMCU เพื่อทำให้ OpenMCU สามารถรองรับผู้ใช้ได้มากขึ้น นอกจากนี้ยังพัฒนาโคเดค (Codec) ใหม่คือ MPEG-4 [16, 17] และเทอร์มินัล (Terminal) สำหรับ OpenMCU ที่ได้เพิ่มประสิทธิภาพแล้วอีกด้วย

หลังจากการเพิ่มประสิทธิภาพของ OpenMCU แล้ว เราได้นำโครงสร้างใหม่นี้มาพัฒนา SIP MCU [18, 19, 20, 21] ซึ่งเป็น MCU ตามมาตรฐานซิพ ซึ่งเป็นโปรโตคอลที่เป็นคู่แข่งกับ H.323 และมีแนวโน้มที่จะเติบโตมาแทนที่ในอนาคตอีกด้วย

1.2 วัตถุประสงค์ของการศึกษา

1. ศึกษากลไกการทำงานของ OpenMCU เพื่อค้นหาข้อบกพร่อง
2. นำเสนอกลไกการเพิ่มประสิทธิภาพของ OpenMCU เพื่อให้รองรับผู้ใช้ได้มากขึ้น
3. เปรียบเทียบผลการเพิ่มประสิทธิภาพของ OpenMCU ทั้งด้วยวิธีการทดลอง และด้วยทฤษฎีทางคณิตศาสตร์
4. นำโครงสร้างของ OpenMCU ที่มีประสิทธิภาพแล้วมาสร้าง SIP MCU

1.3 สมมุติฐานของการศึกษา

เพื่อให้บรรลุวัตถุประสงค์ของการศึกษา เราได้นำเสนอหลักการตามทฤษฎีที่เกี่ยวข้องเพื่อทำให้ OpenMCU มีประสิทธิภาพมากขึ้นทั้งในส่วนของการลดซีพียูโหลด และเน็ตเวิร์คแบนวิดท์ โดยหลักการที่นำเสนอก็คือ การลดคอมเพล็กซิตีของการทำงานของ OpenMCU ในส่วนต่างๆโดยใช้เทคนิคต่างๆ เช่น การเปลี่ยนดาตาสตรัคเจอร์ (Data Structures) และ อัลกอริธึม (Algorithms), การใช้เธรดซิงโครไนเซชัน (Thread Synchronization) และการใช้ SIMD (Single Instruction Multiple Data) SSE2 [22] ซึ่งตามทฤษฎีแล้วเมื่อคอมเพล็กซิตีลดลง ซีพียูโหลดก็จะลดลงด้วยส่งผลให้ OpenMCU รองรับผู้ใช้ได้มากขึ้น นอกจากนี้เราได้ใช้กลไกการหยุดการส่งวิดีโอสตรีมมาที่ MCU ที่เทอร์มินัลที่ไม่แสดงผลในช่องวิดีโอและการใช้โคเดค MPEG-4 ซึ่งมีความต้องการแบนวิดท์ในการสื่อสารต่ำเพื่อลดการใช้เน็ตเวิร์คแบนวิดท์ของ OpenMCU อีกด้วย โดยรายละเอียดของการเพิ่มประสิทธิภาพจะถูกกล่าวถึงอย่างละเอียดในบทที่ 3

1.4 ขอบเขตของงานวิจัย

วิทยานิพนธ์ชิ้นนี้ได้ศึกษา และปรับปรุงประสิทธิภาพของ OpenMCU เพื่อให้รองรับผู้ใช้ได้มากขึ้น ด้วยเทคนิคต่างๆเพื่อลดซีพียูโหลด และเน็ตเวิร์คแบนวิดท์ของ OpenMCU เช่น การ

เปลี่ยนคาตาสตรัคเจอร์ และอัลกอริธึม, การใช้เธรดซิงโครไนเซชัน, การใช้ SIMD SSE2 และการใช้โคเดค MPEG-4 เป็นต้น หลังจากได้ทำการเพิ่มประสิทธิภาพของ OpenMCU แล้ว เราได้ทำการวัดประสิทธิภาพของ OpenMCU เปรียบเทียบก่อนและหลังการเพิ่มประสิทธิภาพในแง่ของซีพียูโหลด, เน็ตเวิร์คแบนวิดท์ และจำนวนหน่วยความจำที่ครอบครองโดย OpenMCU นอกจากนี้เรายังได้พัฒนาเทอร์มินัลใหม่สำหรับ OpenMCU เพื่อรองรับโคเดค MPEG-4 และการส่งแมสเสจ (Message) เพื่อหยุดการส่งวิดีโอสตรีมของเทอร์มินัลที่ไม่แสดงผลในช่องวิดีโออีกด้วย

อนึ่ง SIP MCU ที่เราพัฒนาขึ้นมาจากโครงสร้างของ OpenMCU ที่ได้เพิ่มประสิทธิภาพแล้วไม่ได้อยู่ในขอบเขตของวิทยานิพนธ์นี้

1.5 ขั้นตอนการศึกษา

1. ศึกษาซอร์สโค้ดของ OpenMCU เพื่อให้เข้าใจกลไกการทำงานของ OpenMCU เพื่อค้นหาข้อบกพร่องต่างๆ
2. วิเคราะห์ข้อบกพร่องต่างๆของ OpenMCU แล้วนำเสนอกลไกการเพิ่มประสิทธิภาพของ OpenMCU เพื่อให้รองรับผู้ใช้ได้มากขึ้น
3. ทำการเพิ่มประสิทธิภาพของ OpenMCU ตามกลไกที่ได้นำเสนอไว้
4. สร้างเทอร์มินัลใหม่ให้กับ OpenMCU ที่ได้เพิ่มประสิทธิภาพแล้วเพื่อให้สามารถรองรับ MPEG-4 และการส่งแมสเสจจาก OpenMCU
5. เปรียบเทียบผลการเพิ่มประสิทธิภาพของ OpenMCU ทั้งด้วยวิธีการทดลอง และด้วยทฤษฎีทางคณิตศาสตร์
6. นำโครงสร้างของ OpenMCU ที่มีประสิทธิภาพแล้วมาสร้าง SIP MCU ต่อไป

1.6 ข้อตกลงเบื้องต้น

งานวิจัยชิ้นนี้เป็นการเพิ่มประสิทธิภาพบนระบบปฏิบัติการวินโดส์เท่านั้น โดยเครื่องมือพัฒนาที่ใช้คือ Visual C++ เวอร์ชัน 6.0, Service Pack 5, Processor Pack 5 และทำการวัดประสิทธิภาพบนเครื่องคอมพิวเตอร์ CPU Pentium 4 HyperThread ที่คล็อก 2.8 GHz, Ram 512 MByte โดยถือว่า เน็ตเวิร์ค ที่ใช้มีความเร็วสูง มีความคับคั่งและดีเลย์ไทม์ (Delay Time) น้อย

1.7 ข้อจำกัดของการศึกษา

งานวิจัยชิ้นนี้เป็นการเพิ่มประสิทธิภาพบนระบบปฏิบัติการวินโดส์เท่านั้นแม้ว่าตัว OpenMCU จะสามารถรันได้บนระบบอื่นๆแต่ด้วยความรู้อันจำกัดของผู้ทำวิจัยที่มีความรู้ด้านโปรแกรมมิ่งบนวินโดส์เท่านั้นทำให้ไม่สามารถทำการเพิ่มประสิทธิภาพบนแพลทฟอร์ม

(Platform) อื่นๆได้ สำหรับงานวิจัยนี้ทดลองบนเน็ตเวิร์คแลน (LAN) 10/100 Mbits/sec ซึ่งอาจจะถือได้ว่าเน็ตเวิร์คมีประสิทธิภาพมาก ถ้าทดลองบนเน็ตเวิร์คที่มีประสิทธิภาพต่ำ เช่น Dial up modem 56 Kbits/sec อาจมีปัจจัยอื่นๆมาเกี่ยวข้องกับคุณภาพของสัญญาณเสียงและภาพได้เช่น แบนวิดท์ของช่องทางการสื่อสารและดีเลย์ไทม์ในเน็ตเวิร์ค นอกจากนี้การวัดค่า ซีพียูโหลด และเน็ตเวิร์คแบนวิดท์ของ OpenMCU ยังไม่สามารถวัดค่าได้อย่างละเอียดเนื่องจากซีพียูโหลด และเน็ตเวิร์คแบนวิดท์มีความเปลี่ยนแปลงค่อนข้างมากในช่วงระยะเวลาหนึ่งอีกทั้งเครื่องมือที่ใช้วัดประสิทธิภาพยังต้องใช้สายตาในการคาดคะเนมากซึ่งทำให้เราสามารถวัดค่าได้แต่แนวโน้มเท่านั้น

1.8 รายละเอียดของวิทยานิพนธ์

วิทยานิพนธ์ฉบับนี้เป็นการนำเสนอเทคนิคการเพิ่มประสิทธิภาพของ OpenMCU ในแง่ของซีพียูโหลด และเน็ตเวิร์คแบนวิดท์เพื่อให้สามารถรองรับจำนวนผู้ใช้ได้มากขึ้น โดยรายละเอียดต่างๆภายในวิทยานิพนธ์เล่มนี้ได้จัดแบ่งในส่วนเนื้อหาออกเป็น 5 บท ซึ่งแต่ละบทมีหัวข้อและเนื้อหาดังต่อไปนี้

บทที่ 1 "บทนำ" อธิบายถึงวัตถุประสงค์ สมมุติฐานของการศึกษา ขอบเขตและขั้นตอนการศึกษา รวมไปถึงรายละเอียดเนื้อหาโดยสรุปของแต่ละบท

บทที่ 2 "โปรโตคอลสำหรับ VoIP และ OpenMCU" อธิบายการทำงานของโปรโตคอล 2 โปรโตคอลที่ใช้ใน VoIP คือ H.323 และ SIP รวมทั้งเปรียบเทียบข้อดีข้อเสียของทั้ง 2 โปรโตคอล นอกจากนี้ยังได้อธิบายถึงการทำงานของ OpenMCU ที่เราจะทำการเพิ่มประสิทธิภาพและได้กล่าวถึงงานวิจัยต่างๆที่เกี่ยวข้องที่มีคนเคยทำมาแล้วอีกด้วย

บทที่ 3 "การเพิ่มประสิทธิภาพของ OpenMCU และการสร้าง SIP MCU" อธิบายถึงเทคนิคต่างๆที่ถูกนำเสนอเพื่อเพิ่มประสิทธิภาพของ OpenMCU ทั้งในแง่ของการลดซีพียูโหลด, เน็ตเวิร์คแบนวิดท์ และจำนวนหน่วยความจำที่ครอบครองโดย OpenMCU เพื่อให้สามารถรองรับจำนวนผู้ใช้ได้มากขึ้น นอกจากนี้ยังได้กล่าวถึงวิธีการสร้าง SIP MCU จากโครงสร้างของ OpenMCU ที่มีประสิทธิภาพแล้วอีกด้วย

บทที่ 4 "การเปรียบเทียบและวิเคราะห์ผลการเพิ่มประสิทธิภาพของ OpenMCU" กล่าวถึงการเปรียบเทียบประสิทธิภาพของ OpenMCU ทั้งก่อนและหลังการเพิ่มประสิทธิภาพในแง่ของซีพียูโหลด, เน็ตเวิร์คแบนวิดท์ และจำนวนหน่วยความจำที่ครอบครองโดย OpenMCU

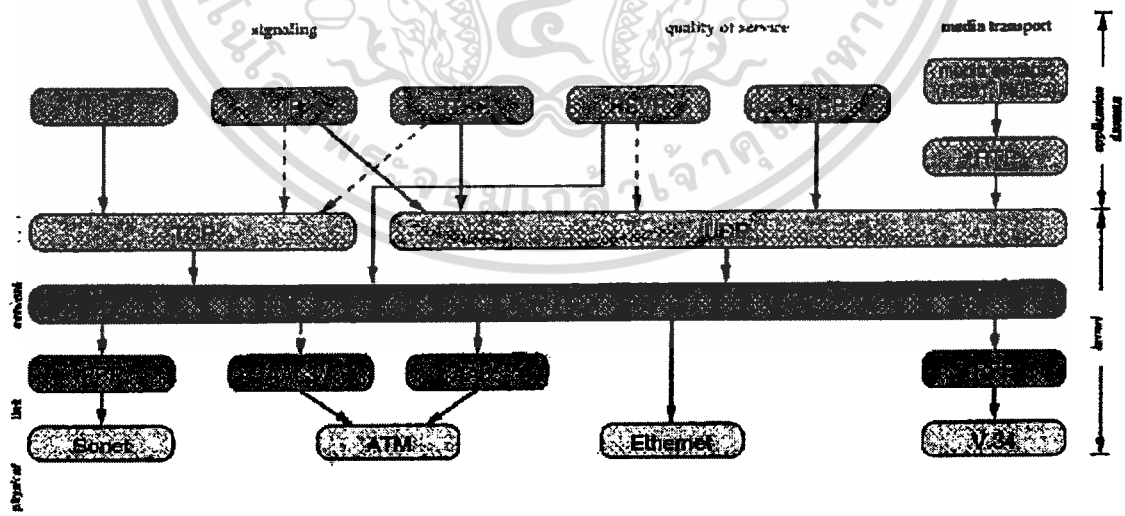
บทที่ 5 "บทสรุป" เป็นการสรุปและวิจารณ์ผลที่ได้จากการวิเคราะห์และการทดลอง พร้อมทั้งปัญหาที่เกิดขึ้น ตลอดจนข้อเสนอแนะสำหรับนำงานวิจัยในวิทยานิพนธ์นี้ไปพัฒนาใช้ประโยชน์ต่อไป

บทที่ 2
โปรโตคอลสำหรับ VoIP และ OpenMCU

2.1 กล่าวนำ

ในปัจจุบันเครือข่ายอินเตอร์เน็ตได้มีจำนวนผู้ใช้เพิ่มขึ้นอย่างรวดเร็วเนื่องจากว่าผู้ใช้สามารถค้นหาข้อมูลที่ต้องการจากแหล่งข้อมูลขนาดใหญ่และสามารถติดต่อสื่อสารกับบุคคลจากที่ต่างๆ ได้อย่างสะดวก นอกจากนี้การพัฒนาการให้บริการในรูปแบบใหม่เพื่อรองรับความต้องการของผู้ใช้ได้เพิ่มขึ้นอย่างรวดเร็ว เนื่องมาจากความยืดหยุ่นของเครือข่ายอินเตอร์เน็ตทำให้สามารถพัฒนาการให้บริการได้ง่ายกว่าเครือข่ายอื่นๆ การให้บริการในรูปแบบหนึ่งที่กำลังได้รับความสนใจจากผู้ให้บริการรายต่างๆ เป็นอย่างมากคือ การใช้งานโทรศัพท์บนเครือข่ายอินเตอร์เน็ตซึ่งใช้โปรโตคอล (protocol) IP (Internet Protocol) โดยเรียกว่า Voice over IP (VoIP) [1, 2, 3] Internet telephony หรือ IP telephony (IPTel) ซึ่งจะรวมไปถึงการส่งวีดีโอและข้อมูลด้วย (หรือเรียกว่า การสื่อสารแบบพหุสื่อ (multimedia communication)) การให้บริการแบบนี้จะทำให้ผู้ใช้สามารถใช้โทรศัพท์บนเครือข่ายอินเตอร์เน็ตแทนเครือข่ายชนิดอื่นที่ใช้อยู่ ซึ่งการใช้งานในลักษณะนี้จะเป็นผลดีต่อผู้ใช้คือ จะช่วยประหยัดค่าโทรศัพท์ทางไกล สามารถใช้งานโทรศัพท์พร้อมกับการใช้งานอย่างอื่นบนอินเตอร์เน็ต มีการเชื่อมต่อในการใช้งานที่ยืดหยุ่นกว่าโทรศัพท์ทั่วไป รวมทั้งยังสามารถปรับแต่งการใช้งานโทรศัพท์ได้หลากหลายกว่า เมื่อพิจารณาในแง่ของผู้ให้บริการหรือผู้ดูแลเครือข่าย การพัฒนาปรับปรุงเครือข่ายหรือการให้บริการสามารถทำได้ง่ายกว่า รวมทั้งการดูแลเครือข่ายทำได้สะดวกกว่า เพราะว่าไม่จำเป็นต้องแยกเครือข่ายสำหรับ ข้อมูล เสียงและวิดีโอ ระบบ VoIP จะทำให้ข้อมูลเสียงและวีดีโอสามารถส่งรวมกันไปในเครือข่ายอินเตอร์เน็ตเพียงเครือข่ายเดียว จากข้อดีเหล่านี้ต่างๆ ทำให้ VoIP เป็นที่สนใจอย่างมาก และอาจจะเป็นไปได้ว่า VoIP จะเข้ามาแทนที่ระบบโทรศัพท์ในปัจจุบันในอนาคตอันใกล้นี้ อย่างไรก็ตามความต้องการระดับพื้นฐานของ VoIP คือคุณภาพของการให้บริการ (QOS) ต้องเทียบเท่าคุณภาพในระบบโทรศัพท์หรือดีกว่าซึ่งเป็นปัญหาสำหรับการใช้งาน VoIP ในทางปฏิบัติ เนื่องจากว่าเครือข่ายอินเตอร์เน็ตเป็นเครือข่ายที่ไม่รับประกันในเรื่องคุณภาพของการให้บริการ ปัญหานี้จึงเป็นอุปสรรคสำคัญในการนำ VoIP มาใช้แทนระบบโทรศัพท์ที่มีอยู่ เครือข่ายอินเตอร์เน็ตได้ให้การบริการในการส่งข้อมูลทั่วไปซึ่งไม่ต้องการคุณสมบัติแบบเวลาจริง (real time) แต่สำหรับในกรณีของ VoIP หรือการสื่อสารแบบพหุสื่อต้องการคุณสมบัติแบบเวลาจริง ดังนั้นจึงต้องมีการพัฒนาโปรโตคอลที่สามารถให้คุณสมบัติดังกล่าวในการส่งข้อมูล และที่สำคัญ VoIP ต้องการฟังก์ชันในการสร้างหรือสิ้นสุดการเรียก และหาตำแหน่งที่อยู่ของผู้ใช้ที่ถูกเรียก รวมทั้งฟังก์ชันที่ช่วยให้สามารถให้บริการได้เช่นเดียวกับในระบบโทรศัพท์ ซึ่งในชุดโปรโตคอล TCP/IP (Transport Control

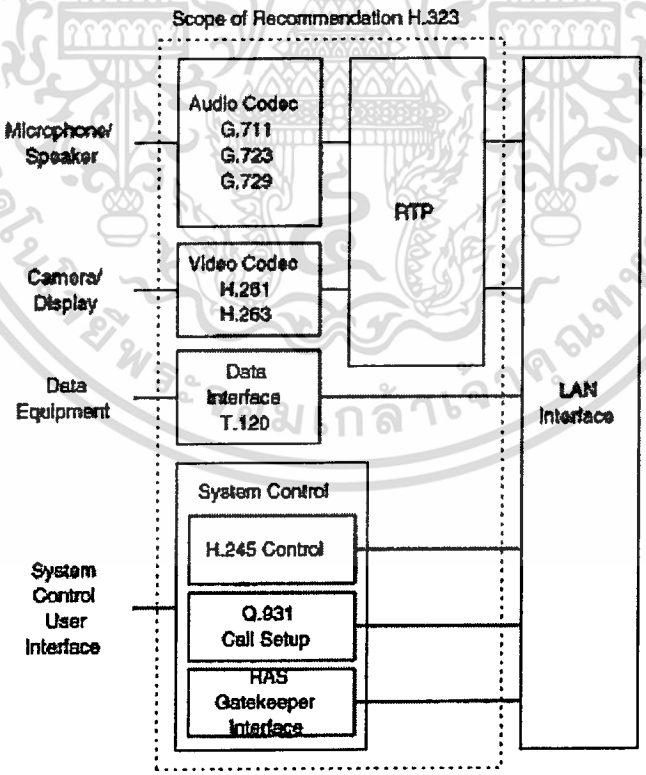
Protocol/Internet Protocol) นั้นไม่มีโปรโตคอล ที่ให้ฟังก์ชันดังกล่าว ดังนั้นจึงต้องมีการพัฒนาโปรโตคอลขึ้นใหม่เพื่อรองรับ VoIP ในปัจจุบันโปรโตคอลสำหรับ VoIP มีอยู่ 2 มาตรฐานคือ H.323 [6, 7, 8] และซิพ (SIP : Session Initial Protocol) [9, 10, 11] โปรโตคอล H.323 เป็นโปรโตคอลที่พัฒนาโดย ITU-T (International Telecommunications Union-Telecommunications section) [12] ส่วน SIP ถูกพัฒนาโดย IETF (Internet Engineering Task Force) [13] โปรโตคอลทั้งสองมีหน้าที่หลักในการสร้าง สิ้นสุด และการเปลี่ยนแปลงรายละเอียดของการเรียก ระหว่างผู้ใช้ VoIP รวมทั้งยังสามารถให้ฟังก์ชันเพิ่มเติมอื่นๆ โปรโตคอลทั้งสองเป็นโปรโตคอลสำหรับ VoIP ซึ่งใช้บริการชุดโปรโตคอล TCP/IP ในชั้นต่ำกว่า และสามารถใช้งานร่วมกับโปรโตคอลอื่น เพื่อให้เกิดการบริการที่มีคุณภาพมากขึ้นโปรโตคอลสแต็ก (Protocol stack) สำหรับ VoIP จะเป็นดังรูปที่ 2.1 จะเห็นว่าทั้งโปรโตคอล H.323 และซิพเป็นโปรโตคอลในชั้นแอพพลิเคชัน (application layer) และใช้บริการของโปรโตคอลในชั้นที่ต่ำกว่า ซิพสามารถใช้ได้ทั้ง UDP และ TCP ส่วน H.323 ใช้ TCP เท่านั้น แต่เนื่องจากว่าฟังก์ชันของ H.323 และซิพมีขอบเขตจำกัด ดังนั้นจึงได้นำโปรโตคอลอื่นมาช่วยในการทำงาน ซึ่งได้แก่ RTSP (Real-time Streaming Protocol) [23] RSVP (Resource Reservation Protocol) [24] และ RTP/RTCP [25] ซึ่งทำให้ VoIP สามารถให้บริการได้อย่างมีประสิทธิภาพมากขึ้นโปรโตคอลดังกล่าวเป็นโปรโตคอลในชั้นแอพพลิเคชันซึ่งทำงานอยู่บนชุดโปรโตคอล TCP/IP โปรโตคอลเหล่านี้ไม่ได้เป็นโปรโตคอลเฉพาะสำหรับ VoIP ดังนั้นจะกล่าวถึงอย่างคร่าวๆ ในส่วนนี้ แต่สำหรับโปรโตคอล H.323 และซิพซึ่งเป็นโปรโตคอลหลักสำหรับ VoIP จะกล่าวถึงรายละเอียดในหัวข้อถัดไป



รูปที่ 2.1 แสดง IP telephony protocol stack

2.2 โปรโตคอล H.323 (H.323 Protocol)

H.323 เป็นมาตรฐานหรือโปรโตคอลสำหรับการสื่อสารแบบพหุสื่อ (multimedia communication) แบบเวลาจริง บนเครือข่าย IP โปรโตคอล H.323 ได้ให้รายละเอียดสำหรับขั้นตอนในการสร้างการเรียก (call setup) เอนทิตี้ภายในเครือข่าย H.323 และการทำงานร่วมกันระหว่างเอนทิตี้ภายในเครือข่าย H.323 ถูกพัฒนาโดย ITU-T โดยเป็นส่วนหนึ่งของมาตรฐาน H.32x ที่เป็นมาตรฐานสำหรับการประชุมแบบพหุสื่อ (multimedia conference) บนเครือข่ายต่างๆ เช่น H.320 สำหรับเครือข่าย ISDN (Integrated Service Digital Networks) H.324 สำหรับเครือข่าย PSTN (Public Switching Telephone Networks) H.323 จะครอบคลุมโปรโตคอลอื่นไว้ คือ H.225.0 สำหรับ call signaling และการจัดรูปแบบแพ็กเกตมีเดีย (media packet format) H.245 สำหรับการแลกเปลี่ยนข้อมูลความสามารถเกี่ยวกับมีเดีย (media capability exchange) และการควบคุมช่องสัญญาณมีเดีย (media channel control) H.450.x เป็นขั้นตอนสำหรับสร้างการบริการเพิ่มเติม (supplementary service) และ H.235 เป็นมาตรฐานเกี่ยวกับความปลอดภัย เป็นต้น รวมทั้งยังได้อ้างอิงถึงมาตรฐานในการเข้ารหัสสำหรับสัญญาณเสียง เช่น G.711 G723.1 G.729 และสัญญาณวีดีโอเช่น H.261 และ H.263



รูปที่ 2.2 แสดง H.323 terminal architecture

ขอบเขตของ H.323 ดังแสดงในรูปที่ 2.2 จะจำกัดอยู่ที่มาตรฐานในการบีบอัดข้อมูล (compression) รูปแบบแพ็กเกตมีเดีย (media packet format) การส่งสัญญาณ (signalling) และ การควบคุมการรับส่งข้อมูล (flow control) ซึ่งจะมีรายละเอียดดังนี้

2.2.1 สถาปัตยกรรมของ H.323 (H.323 Architecture)

โปรโตคอล H.323 ครอบคลุมและอ้างอิงถึงโปรโตคอลอื่นๆ เช่น H.225.0/Q.931[26] H.245 [27] และH.225.0/RAS [26] เพื่อช่วยในการทำงานของโปรโตคอล H.323 โดยสถาปัตยกรรมของโปรโตคอล H.323 สำหรับเอนด์พอยน์ (endpoint) หรือเทอร์มินัล (terminal) จะเป็นดังรูปที่ 2.2

- การเข้ารหัส/ถอดรหัสสัญญาณวิดีโอ (video codec) ทำหน้าที่ในการเข้ารหัสสัญญาณวีดีโอสำหรับการส่ง และถอดรหัสสัญญาณวีดีโอที่ได้รับซึ่งจะถูกนำไปแสดงผลต่อไป มาตรฐานการเข้ารหัส/ถอดรหัสที่เอนด์พอยน์จำเป็นต้องเข้า/ถอดรหัสได้คือ H.261 ที่ระดับความละเอียด QCIF (Quarter Common Intermediate Format) ส่วน H.263 ซึ่งให้คุณภาพที่ดีกว่าเป็นตัวเลือกที่อาจจะรองรับหรือไม่ก็ได้ รายละเอียดเกี่ยวกับการเข้า/ถอดรหัสสัญญาณวีดีโอจะตกลงกันระหว่างเอนด์พอยน์ในช่วงของการแลกเปลี่ยนความสามารถ (capability exchange) โดยการใช้โปรโตคอล H.245 มาตรฐานการเข้ารหัส/ถอดรหัสที่จะใช้จะต้องรองรับโดยทุกๆเอนด์พอยน์ที่เข้าร่วมในการสื่อสาร
- การเข้ารหัสสัญญาณเสียง (audio codec) ทำหน้าที่ในการเข้ารหัสเสียงจากไมโครโฟน หรือแหล่งกำเนิดอื่นสำหรับการส่ง และถอดรหัสสัญญาณที่ถูกเข้ารหัสที่รับได้ซึ่งจะถูกส่งต่อไปยังลำโพง มาตรฐานที่เอนด์พอยน์จำเป็นต้องรองรับ คือ G.711 สำหรับ G.722, G.728, G.729, MPEG-1 และ G.723.1 เป็นตัวเลือกที่อาจจะรองรับหรือไม่ก็ได้ มาตรฐานการเข้ารหัส/ถอดรหัสที่ใช้จะต้องรองรับโดยทุกเอนด์พอยน์ซึ่งจะทำการตกลงกันโดยใช้โปรโตคอล H.245
- ช่องสัญญาณส่งข้อมูล (data channel) มาตรฐานที่ใช้ คือ T.120 สำหรับการสร้างการประชุมข้อมูล (data conferencing) รายละเอียดหรือพารามิเตอร์อาจจะตกลงกันโดยใช้โปรโตคอล H.245
- RTP (real time transport protocol) ทั้งสัญญาณเสียงและวีดีโอจะถูกส่งโดยบรรจุในแพ็กเกต RTP ซึ่งเป็นโปรโตคอลที่ใช้สำหรับการส่งข้อมูลแบบเวลาจริง บนเครือข่าย IP โดย RTP จะทำงานร่วมกับ RTCP ซึ่งทำหน้าที่ในการควบคุมการส่งข้อมูลโดย RTP ดังที่ได้กล่าวมาแล้ว

System Control Unit เป็นส่วนที่ทำหน้าที่เกี่ยวกับ การส่งสัญญาณ และ flow control ประกอบด้วยส่วนต่างๆ ดังนี้

- H.245 เป็นโปรโตคอลควบคุมมีเดีย (media control protocol) ทำหน้าที่ในการแลกเปลี่ยนความสามารถ (capability exchange) ตกลงรายละเอียดของช่องสัญญาณ (channel negotiation) เปลี่ยนโหมดของมีเดีย (switching of media mode) และการสร้างช่องสัญญาณทางตรรก (logical channel) สำหรับการส่งเสียงหรือวิดีโอ โปรโตคอลนี้จะใช้ TCP ในการส่งแมสเสจโดยใช้ช่องสัญญาณในการส่งของตัวเอง
- H.225.0/Q.931 เป็นโปรโตคอล H.225.0 ในส่วนที่ทำหน้าที่สร้างการเชื่อมต่อ (connection establishment) ซึ่งถูกดัดแปลงมาจาก โปรโตคอล Q.931 โดยโปรโตคอลนี้จะใช้ TCP ในการส่งแมสเสจผ่านช่องสัญญาณของตัวเอง
- H.225.0 RAS เป็นโปรโตคอล H.225.0 ที่ทำหน้าที่ในการควบคุมการยอมรับ (admission control) การลงทะเบียน (registration) และการรายงานสถานะ โปรโตคอลนี้จะใช้ระหว่างเอนด์พอยน์และเกทคีปเปอร์ (gatekeeper) เพื่อใช้สำหรับการควบคุมดูแลโดยเกทคีปเปอร์ โดยแมสเสจในโปรโตคอลจะใช้ UDP
- H.225.0 layer โปรโตคอล H.225.0 เป็นโปรโตคอลสำหรับ call-signaling ซึ่งมีหน้าที่ดังที่กล่าวมาข้างต้น นอกจากนั้นยังทำหน้าที่ในการแปลงมีเดีย (วิดีโอ เสียง และข้อมูล) และข้อมูลสำหรับการควบคุม (control data) ที่จะถูกส่งให้อยู่ในรูปแพ็กเกตที่เหมาะสมเพื่อส่งต่อให้กับ network interface และทำหน้าที่รับข้อมูลทั้งหมด (media data และ control data) จาก network interface เพื่อส่งให้ส่วนอื่นต่อไป

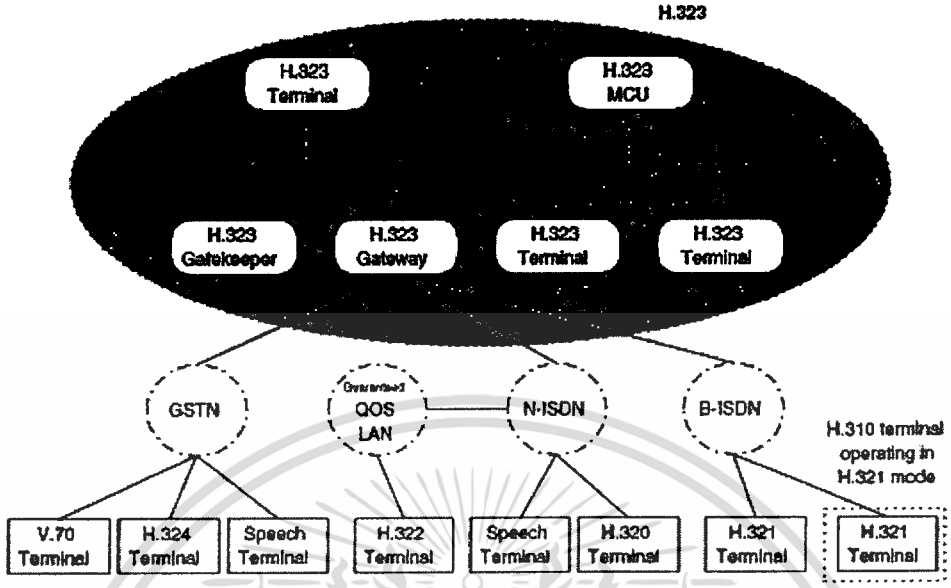
2.2.2 เอนทิตี้และฟังก์ชันของ H.323 (H.323 entities & functions)

ในมาตรฐาน H.323 ได้อธิบายถึงเอนทิตี้ที่เป็นองค์ประกอบเครือข่าย H.323ซึ่งได้แก่ เทอร์มินัล MCU (Multipoint Control Unit) เกทเวย์ และเกทคีปเปอร์ การเชื่อมต่อระหว่างเอนทิตี้ภายในเครือข่าย H.323 กับเครือข่ายอื่นดังแสดงในรูปที่ 2.3

รายละเอียดของแต่ละเอนทิตี้มีดังนี้

1) H.323 เทอร์มินัล (H.323 terminal)

เทอร์มินัลดังแสดงในรูปที่ 2.2 เป็นเอนด์พอยน์ของเครือข่ายซึ่งอาจจะเป็นคอมพิวเตอร์หรือชุดอุปกรณ์ที่สามารถใช้งานโปรโตคอล H.323ได้ เทอร์มินัลต้องสนับสนุนการสื่อสารโดยใช้เสียง ส่วนสัญญาณวีดีโอและข้อมูลเป็นตัวเลือก ซึ่งฟังก์ชันหลักของเทอร์มินัลมีดังนี้



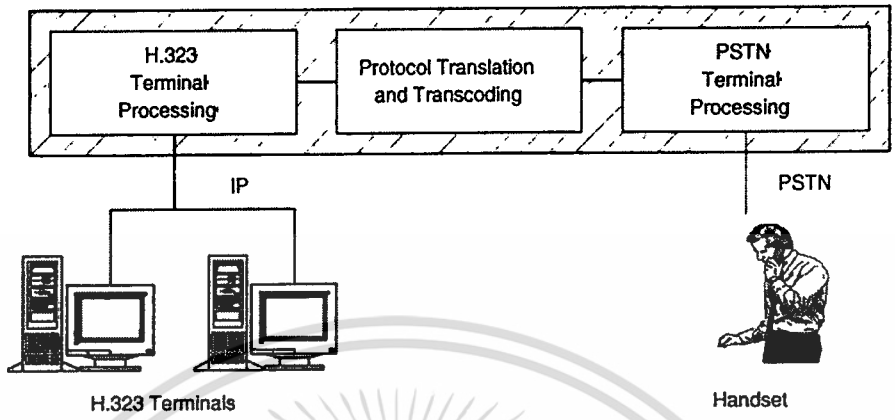
รูปที่ 2.3 เครือข่าย H.323 เอนทิตี้และฟังก์ชัน

- ทำหน้าที่ในการติดต่อกับผู้ใช้ โดยรับคำสั่งและแสดงผลให้กับผู้ใช้
- จัดการในการส่ง call signaling ให้กับ voice gateway
- ส่งหมายเลขโทรศัพท์ (มาตรฐาน E.164) และหมายเลข IP ของผู้ใช้ ให้กับเกทคีปเปอร์ ซึ่งเป็นหมายเลขที่ใช้อ้างอิงถึงในการเชื่อมต่อ ในการส่งหมายเลขดังกล่าวจะบรรจุอยู่ในแมสเสจ ARQ ของโปรโตคอล H.225.0/ RAS ซึ่งอาจจะมีหมายเลข alias address ส่งไปพร้อมกัน
- ทำการแปลงแพ็กเกตที่ได้รับจากเครือข่าย โดยผ่านกระบวนการของโปรโตคอลในชั้นต่างๆ ตามลำดับ (IP->UDP->RTP) เป็นเฟรมเสียง แล้วทำการถอดรหัส G.xxx ให้อยู่ในรูปของ PCM (Pulse Code Modulation) สตรีมเพื่อทำการส่งให้กับซาวน์การ์ดเพื่อแสดงผลต่อไป
- ทำการเข้ารหัส G.xxx ให้กับ PCM สตรีมจากซาวน์การ์ดและทำการรวมเป็นแพ็กเกต แล้วแปลงแพ็กเกตเป็นแพ็กเกตที่ส่งในเครือข่ายโดยผ่านกระบวนการของโปรโตคอลในชั้นต่างๆ (RTP->UDP->IP) แล้วจึงทำการส่งผ่านเครือข่ายในรูปของแพ็กเกต

2) H.323 เกทเวย์ (H.323 Gateway)

เกทเวย์เป็นเอนทิตี้ที่ทำหน้าที่ในการเชื่อมต่อระหว่างเครือข่าย H.323 กับเครือข่ายอื่นซึ่งอาจจะไม่จำเป็นต้องมีในกรณีที่ไม่มีการเชื่อมต่อกับเครือข่ายชนิดอื่นๆ การเชื่อมต่อเครือข่าย H.323

กับเครือข่ายอื่นโดยใช้เกทเวย์จะมีลักษณะดังรูปที่ 2.4 เกทเวย์ทำหน้าที่เสมือนเป็นเอนด์พอยน์ของเครือข่ายหนึ่งในการเชื่อมต่อระหว่างเครือข่าย โดยจะทำหน้าที่ในการเชื่อมต่อระหว่างเครือข่ายดังนี้



รูปที่ 2.4 H.323/PSTN Gateway

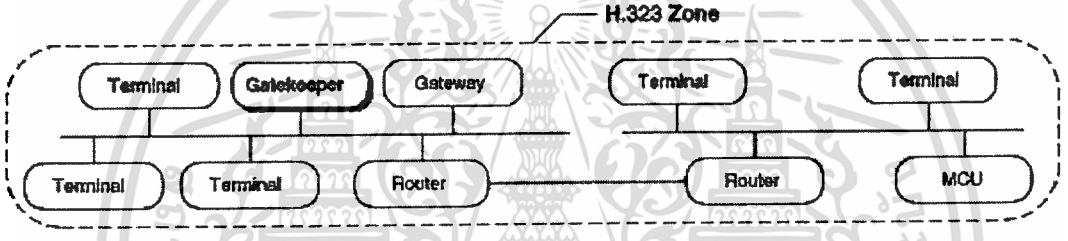
- สร้างการเชื่อมต่อกับเทอร์มินัล PSTN ในระบบแอนาลอก
- สร้างการเชื่อมต่อกับเทอร์มินัลที่รองรับมาตรฐาน H.320 บนเครือข่าย switched circuit ที่เป็น ISDN
- สร้างการเชื่อมต่อเทอร์มินัลที่รองรับมาตรฐาน H.324 บนเครือข่าย PSTN

เนื่องจากเกทเวย์สามารถให้การเชื่อมต่อระหว่าง H.323 กับเครือข่ายอื่น ดังนั้นฟังก์ชันของเกทเวย์ จึงเป็นฟังก์ชันในการแปลงข้อมูลระหว่าง 2 เครือข่ายคือ

- รับและประมวลผลการเรียกที่มาจากเทอร์มินัลในเครือข่ายอื่นไปยังเทอร์มินัล H.323 เกทเวย์จะต้องทำการแปลง การส่งสัญญาณและcontrol ต่างๆ จากเครือข่ายอื่นมาเป็นของ H.323 เช่น จากขั้นตอนในการสร้างการสื่อสารจาก H.242 เป็น H.245 รวมทั้งทำหน้าที่สร้างและสิ้นสุดการเรียก หรืออาจจะมองได้ว่าเกทเวย์จะทำหน้าที่แทนเทอร์มินัลในเครือข่ายอื่นโดยเสมือนกับเป็นเทอร์มินัลในเครือข่าย H.323
- รับและประมวลผลการเรียกจากเทอร์มินัล H.323 ไปยังเครือข่ายอื่น เกทเวย์จะต้องทำการแปลงการส่งสัญญาณ และ control ต่างๆ ตาม H.323 ให้เป็นมาตรฐานในเครือข่ายอื่น เช่น แปลงจากโปรโตคอล H.245 เป็น H.242 รวมทั้งสร้างและสิ้นสุดการเรียก หรืออาจจะมองว่าเกทเวย์จะทำหน้าที่แทนเทอร์มินัลในเครือข่าย H.323 เสมือนกับเป็นเทอร์มินัลในเครือข่ายอื่น
- ทำหน้าที่ในการดูแลกระบวนการการเรียก (call) และการส่งสัญญาณ (signaling) ว่ามีความผิดพลาดเกิดขึ้นหรือไม่ ซึ่งการทำงานจะอยู่ภายใต้การควบคุมของเกทคีปเปอร์

- ทำการเข้ารหัส G.xxx ให้กับสัญญาณ PCM จากเครือข่ายอื่น แล้วทำการรวมสัญญาณที่ถูกเข้ารหัสให้เป็นแพ็กเกตเพื่อส่งไปในเครือข่าย IP โดยผ่านกระบวนการแปลงของโปรโตคอลในชั้นต่างๆ เพื่อให้อยู่ในรูปแพ็กเกตที่สามารถส่งไปในเครือข่าย ดังในหัวข้อที่ 2
- ทำการแปลงแพ็กเกตจากเครือข่าย IP กลับแพ็กเกตเสียง แล้วทำการถอดรหัส G.xxx และแปลงสัญญาณ PCM เพื่อส่งให้กับเครือข่ายภายนอก

สำหรับฟังก์ชันเพิ่มเติมของเกทเวย์ไม่ได้มีการกำหนดไว้แน่นอนขึ้นอยู่กับผู้ออกแบบ เช่น จำนวนของเทอร์มินัลที่รองรับจำนวนการเชื่อมต่อกับเครือข่าย switched circuit และจำนวนการประชุมที่สนับสนุน เป็นต้น เมื่อเกทเวย์มีการพัฒนามากขึ้นจะทำให้เครือข่าย H.323 สามารถเชื่อมต่อกันกับเครือข่ายชนิดอื่นๆ ได้มากขึ้น



รูปที่ 2.5 H.323 Zone

3) H.323 เกทคีปเปอร์ (Gatekeeper)

เกทคีปเปอร์ทำหน้าที่ในการดูแลและให้บริการกับเอนทิตี้อื่นภายในโซนดังรูปที่ 2.5 โดยโซนจะประกอบไปด้วยเกทคีปเปอร์ 1 ตัวและเอนทิตี้อื่นๆ ทั้งหมดที่ลงทะเบียนกับเกทคีปเปอร์ ถึงแม้ว่าเกทคีปเปอร์เป็นเอนทิตี้ที่ไม่จำเป็นต้องมีในเครือข่าย H.323 แต่เกทคีปเปอร์ก็เป็นเอนทิตี้ที่สำคัญมาก ด้วยเหตุผลต่างๆ ดังนี้

- เครือข่ายขนาดใหญ่สามารถแบ่งเป็นหลายโซน ซึ่งแต่ละโซนจะอยู่ภายในการดูแลของเกทคีปเปอร์เพื่อความสะดวกในการดูแลรักษาเครือข่าย
- เกทคีปเปอร์สามารถให้ความปลอดภัยในการเข้าถึงเครือข่ายได้โดยการให้บริการ authentication สำหรับแต่ละการเรียก หรือแต่ละเอนทิตี้
- เกทคีปเปอร์เป็นศูนย์กลางในการ authentication authorization และ admission (เรียกรวมกันว่า AAA) ของโซน
- เกทคีปเปอร์สามารถจัดการควบคุมแบนด์วิดท์(bandwidth management) เช่นการจำกัดจำนวนของการเชื่อมต่อ

เพื่อเป็นการรักษาแบนวิดท์สำหรับการใช้งานอย่างอื่น เช่น อีเมล์ การโอนย้ายไฟล์ เกทคีปเปอร์ไม่สามารถเป็นเอนด์พอยน์ของการเชื่อมต่อได้ เมื่อมีเกทคีปเปอร์ทุกเอนทิตี้จะต้องทำการลงทะเบียนกับเกทคีปเปอร์ ดังนั้นเกทคีปเปอร์จึงทำหน้าที่เป็นศูนย์กลางของการเรียกทั้งหมดภายในโซนและอาจจะให้บริการเพิ่มเติมกับโซนได้ สำหรับฟังก์ชันหลักที่จำเป็นของเกทคีปเปอร์ตามมาตรฐาน H.323 มี 4 ฟังก์ชันดังนี้

- การแปลงแอดเดรส (Address translation) เกทคีปเปอร์จะทำหน้าที่ในการแปลง alias address ให้เป็น transport address เอนทิตี้จะทำการส่ง alias พร้อมกับการลงทะเบียนโดยใช้แมสเสจ RRQ ซึ่งอาจจะสามารถปรับเปลี่ยนในภายหลังได้
- การควบคุมการยอมรับ (Admission control) เมื่อเอนทิตี้ภายในโซนต้องการสร้างการเรียกจะต้องทำการขออนุญาตไปยังเกทคีปเปอร์ โดยใช้แมสเสจ ARQ เกทคีปเปอร์อาจจะอนุญาตหรือไม่ก็ได้โดยจะทำการตรวจสอบจากเงื่อนไขต่างๆ เช่น แบนด์วิดท์ แหล่งกำเนิดการเรียก (call) และ authentication เป็นต้น
- การควบคุมแบนด์วิดท์ (Bandwidth control) เกทคีปเปอร์สามารถรองรับการควบคุมแบนด์วิดท์ได้ เอนทิตี้จะทำการร้องขอแบนด์วิดท์ที่ต้องการโดยใช้แมสเสจ BRQ และเกทคีปเปอร์จะทำการตรวจสอบค่าแบนด์วิดท์ที่ร้องขอกับเงื่อนไขที่กำหนดไว้สำหรับการจัดการแบนด์วิดท์ (bandwidth management) แล้วจึงจะอนุญาตหรือไม่อนุญาตด้วยการส่งแมสเสจ BCF หรือ BRJ ตามลำดับ
- การจัดการโซน (Zone management and Directory service) โซนจะประกอบด้วยเทอร์มินัล เกทเวย์และ MCU ทั้งหมดที่ลงทะเบียนกับ เกทคีปเปอร์ 1 ตัว เกทคีปเปอร์ทำหน้าที่ในการดูแลและจัดการให้กับทุกเอนทิตี้ที่อยู่ในภายในโซน โดยการใช้ฟังก์ชันข้างต้นและการให้บริการอื่นๆ รวมทั้งการให้บริการ directory service ของโซน

นอกจากฟังก์ชันหลักดังกล่าวแล้ว เกทคีปเปอร์อาจจะให้ฟังก์ชันเพิ่มเติมอื่นๆ มีดังนี้

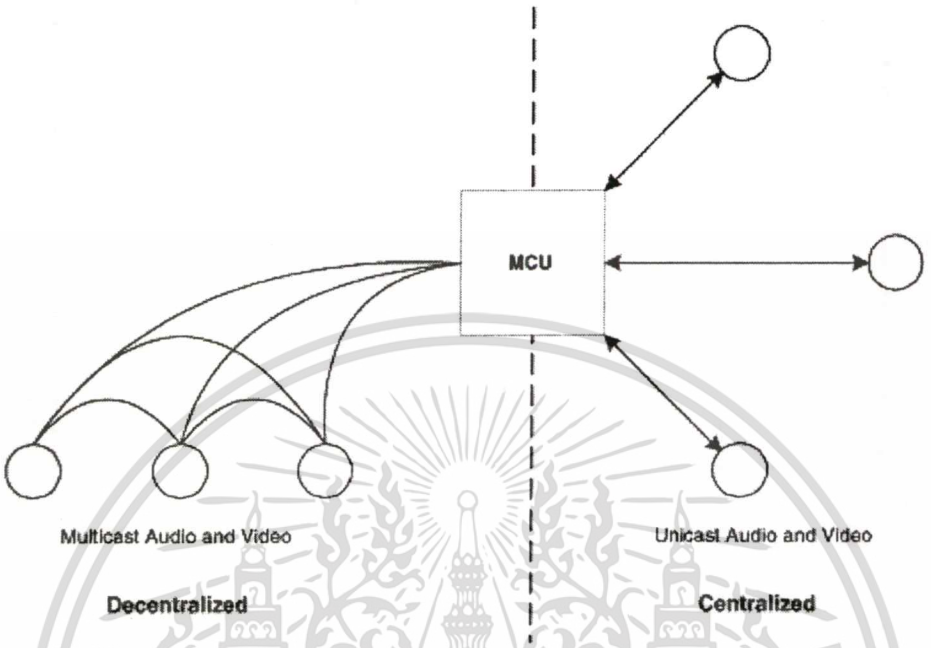
- การควบคุมการส่งสัญญาณ (call control signaling) เกทคีปเปอร์อาจจะช่วยในการประมวลผลแมสเสจ Q.931 ที่ส่งระหว่าง เทอร์มินัลได้ในระหว่างการสร้างการเรียก
- การตรวจสอบการเรียก (call authorization) เกทคีปเปอร์อาจจะปฏิเสธการสร้างการเรียกจากเทอร์มินัลด้วยเหตุผลบางอย่าง เช่น จำกัดการเข้าถึงจากเทอร์มินัลหรือเกทเวย์บางตัว จำกัดการเข้าถึงในบางช่วงเวลา เป็นต้น ซึ่งเงื่อนไขในการตรวจสอบจะอยู่นอกเหนือขอบเขตของ H.323
- การจัดการแบนด์วิดท์ เกทคีปเปอร์จะปฏิเสธการเรียกจากเทอร์มินัลในกรณีที่มีแบนด์วิดท์ไม่เพียงพอ รวมถึงในกรณีที่มีการร้องขอการเพิ่มแบนด์วิดท์ สำหรับเงื่อนไขจะอยู่นอกเหนือขอบเขตของ H.323

- การจัดการการเรียก (call management) เกทคีปเปอร์อาจจะทำการเก็บรักษารายการการเรียกที่เกิดขึ้นเพื่อใช้ในการระบุเทอร์มินัลที่ถูกเรียกว่าว่างหรือไม่ หรือเพื่อให้ข้อมูลกับฟังก์ชันในการจัดการแบนด์วิดท์
- การตรวจสอบผู้ใช้ (authenticating users) สามารถจำกัดการเข้าถึงของผู้ใช้ได้ตามเงื่อนไขที่กำหนดไว้
- การจัดการบริการ (managing services) เกทคีปเปอร์จะทำหน้าที่ในการจัดการให้บริการต่างๆ แก่ผู้ใช้
- การจัดการฐานข้อมูลของสมาชิก (managing subscriber databases) เกทคีปเปอร์ทำหน้าที่ดูแลและจัดการเกี่ยวกับฐานข้อมูลของสมาชิกที่ได้ลงทะเบียนไว้กับเกทคีปเปอร์
- การหาตำแหน่งของสมาชิก (locating subscribers) เกทคีปเปอร์ทำหน้าที่ในการหาตำแหน่งของสมาชิกได้โดยการค้นหาจากข้อมูลของสมาชิกที่ได้จากการลงทะเบียน
- การรวบรวมข้อมูลสำหรับการเก็บค่าบริการ (collecting charging information) เกทคีปเปอร์จะทำการเก็บข้อมูลที่จำเป็นสำหรับการคิดค่าบริการของการเรียก โดยที่การเรียก (call) ต้องถูกจัด เส้นทางผ่านเกทคีปเปอร์
- การควบคุมเกทเวย์ (managing gateway) เกทคีปเปอร์จะควบคุมการทำงานของเกทเวย์ เช่น ควบคุมการสร้างการเรียก ของเกทเวย์ระหว่างเครือข่าย
- การช่วยในการสร้างการเรียก (assisting in call setup) เมื่อการเรียกถูกจัดเส้นทางผ่านเกทคีปเปอร์ เกทคีปเปอร์จะช่วยในการสร้างการเรียก เช่นอาจจะทำการจัดเส้นทางให้กับการเรียก ไปยังเกทเวย์ที่เหมาะสม การติดต่อสื่อสารระหว่างเอนทิตี้กับเกทคีปเปอร์จะใช้โปรโตคอล H.225.0 /RAS ส่วนแมสเสจ call signaling (H.225.0/ Q.931) และ media control (H.245) อาจจะผ่านเกทคีปเปอร์หรือไม่ก็ได้ขึ้นอยู่กับการลงทะเบียนของเอนทิตี้และเงื่อนไขของเกทคีปเปอร์สำหรับ เกทคีปเปอร์อาจจะถูกรวมอยู่ในเกทเวย์และ MCU ได้ โดยที่ต้องแยกทางตรรก (logical) จากเอนด์พอยน์

4) Multipoint Control Unit (MCU)

MCU ทำหน้าที่ในการสนับสนุนการประชุมแบบหลายจุด (multipoint conference) ระหว่างเทอร์มินัล 3 เทอร์มินัลขึ้นไป MCU เป็นเอนทิตี้ที่จะมีหรือไม่ก็ได้ MCU ประกอบด้วย multipoint controller (MC) และ multipoint processor (MP) ในการประชุมจะต้องมี MC ส่วน MP อาจจะมีหรือไม่มีก็ได้ หรืออาจจะมีมากกว่าหนึ่งก็ได้ MC เป็นส่วนที่ทำหน้าที่ในจัดการเกี่ยวกับการส่งสัญญาณในการควบคุมมีเดีย (media control signaling) ให้กับแต่ละเทอร์มินัล โดยที่ทุกเทอร์มินัลต้องมีช่องสัญญาณ H.245 เชื่อมต่อกับ MC แบบจุดถึงจุด (point-to-point) ส่วน MP จะทำ

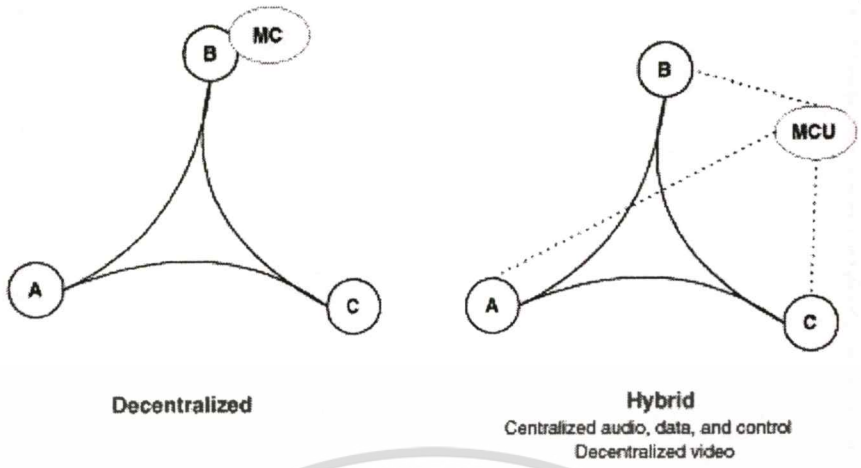
หน้าที่ในการจัดการกับมีเดียสตรีมโดยทำหน้าที่ในการผสม (mixing) สวิตซ์ (switching) และประมวลผลมีเดียที่ใช้การประชุมภายใต้การ ควบคุมของ MC



รูปที่ 2.6 Centralized และ Decentralized MCU

การประชุมแบบหลายจุด (multipoint conference) คือการสื่อสารที่มีผู้เข้าร่วมมากกว่า 2 ซึ่งจำเป็นต้องมี MC อยู่เป็นอย่างน้อย สำหรับแบบจำลองที่ใช้มี 3 แบบ

- Centralize Model ดังรูปที่ 2.6 ในแบบจำลองนี้จำเป็นต้องมี MCU อยู่ ทุกๆเทอร์มินัลที่เข้าร่วมในการประชุมต้องมีช่องสัญญาณ H.245 เชื่อมต่อแบบจุดถึงจุด (point-to-point) กับ MCU ซึ่ง MC จะหน้าที่ควบคุมการประชุมโดยใช้ฟังก์ชันของ H.245 ส่วน MP จะทำหน้าที่รับมีเดียสตรีมจากทุกเทอร์มินัลทำการรวมสัญญาณเสียง เลือกสัญญาณวีดีโอที่ตรงกัน และประมวลผล แล้วทำการส่งกลับไปให้กลับเทอร์มินัลอื่นๆ ทุกเทอร์มินัล
- Decentralized Model ดังรูปที่ 2.6 และ 2.7 ในแบบจำลองนี้ เทอร์มินัลจะมัลติคาสท์สัญญาณเสียงและวีดีโอให้กับเทอร์มินัลอื่นๆ โดยไม่ผ่าน MCU แต่การควบคุมยังคงถูกควบคุมโดย MC ผ่านทางช่องสัญญาณ H.245 ที่เชื่อมต่อกับเทอร์มินัลแบบจุดถึงจุด (point-to-point) เทอร์มินัลที่ได้รับสัญญาณจะทำหน้าที่ในการประมวลผลสัญญาณเอง โดยอาจจะใช้ฟังก์ชัน MP ของแต่ละเทอร์มินัลช่วยทำหน้าที่ในการประมวลผลมีเดียสตรีม



รูปที่ 2.7 Decentralized และ Hybrid MCU

- Hybrid Model ดังรูปที่ 2.7 แมสเสจ H.245 รวมทั้งสัญญาณเสียงหรือวีดีโอจะถูกส่งและประมวลผลผ่านMCUโดยใช้การเชื่อมต่อแบบจุดถึงจุด (point-to-point) ส่วนสัญญาณที่เหลือจะถูกส่งโดยเทอร์มินัลแบบมัลติคาสท์ให้กับเทอร์มินัลอื่นๆ

2.3 โปรโตคอลซิพ (SIP: Session Initial Protocol)

เซสชั่นอินนิชิเอชั่นโปรโตคอลหรือซิพ (SIP) เป็นโปรโตคอลที่ใช้งานสำหรับไอพีเทเลโฟนนี่ (IP Telephony) ซึ่งกำหนดโดยไออีทีเอฟ (IETF : Internet Engineering Task Force) ซิพเป็นโปรโตคอลในชั้นแอพพลิเคชั่นซึ่งทำหน้าที่ในการสร้าง สิ้นสุด และเปลี่ยนแปลงแก้ไขมัลติมีเดียเซสชั่น (multimedia session) เช่น โปรแกรมรับส่งสารด่วน (Instant Messaging), การประชุมทางไกล (Multimedia Conferencing) และวอยซ์โอเวอร์ไอพี (Voice over IP) เป็นต้น ซิพเป็นโปรโตคอลแบบไคล์เอนท์-เซิร์ฟเวอร์ (client – server) โดยใช้การส่งข้อมูลในรูปของตัวอักษร (text based) และซิพสามารถใช้งานหรือทำงานร่วมกับโปรโตคอลอื่นๆ เพื่อเพิ่มประสิทธิภาพให้ดีขึ้น เช่น ทีซีพี (TCP : Transmission Control Protocol), ทีแอลเอส (TLS : Transmission Layer Security), ยูดีพี (UDP : User Datagram Protocol), ไอพี (IP : Internet Protocol), ดีเอ็นเอส (DNS : Domain Name System) เป็นต้น แต่ฟังก์ชั่นและการทำงานของซิพไม่ขึ้นอยู่กับโปรโตคอลเหล่านี้ สำหรับฟังก์ชันที่ SIP สนับสนุนมีดังนี้

- User location การกำหนดเอนด์พอยน์ที่ใช้ในเซสชันการสื่อสาร
- User capabilities การกำหนดมีเดียและพารามิเตอร์ของมีเดียที่ใช้ในการสื่อสาร
- User availability การกำหนดความต้องการของผู้ถูกเรียกว่าต้องการเข้าร่วมในเซสชันหรือไม่

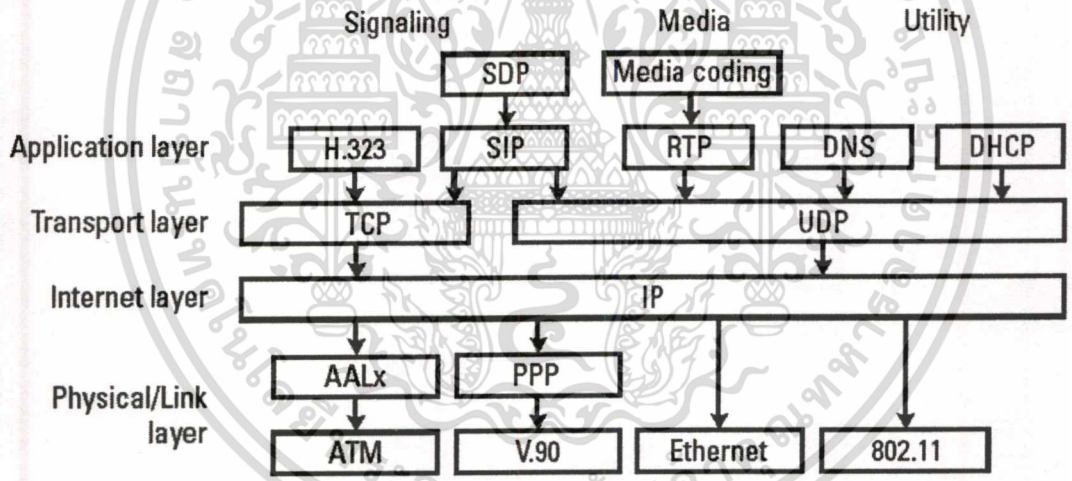
- Call setup การสร้าง การเรียก และกำหนดพารามิเตอร์ของการเรียก
- Call handling การจัดการกับ การเรียก รวมทั้งการโอนย้าย การเรียก และการสิ้นสุดการเรียก

2.3.1 โปรโตคอลสแต็ก (Protocol Stack)

SIP โปรโตคอลสแต็กแสดงดังรูปที่ 2.8 โดยมีรายละเอียดดังนี้

1) ชั้นสื่อสารกายภาพ/ชั้นสื่อสารเชื่อมต่อ (Physical Layer / Link Layer)

เป็นชั้นที่อยู่ล่างสุด ซึ่งมีหลายวิธีการและหลายรูปแบบในการเชื่อมต่อ เช่น อีเธอร์เน็ต (Ethernet) ผ่านเครือข่ายเฉพาะพื้นที่ (LAN : Local Area Network), สายโทรศัพท์ (V.90 หรือ 56K โมเด็ม) ผ่านโปรโตคอลจุดต่อจุด (PPP : Point-to-Point Protocol), ดีเอสแอล (DSL : Digital subscriber line) ผ่าน (ATM : Asynchronous Transport Mode), หรือผ่านเครือข่ายไร้สาย 802.11(Wireless LAN) โดยในชั้นนี้จะเป็นการกำหนดคุณสมบัติทางกายภาพของฮาร์ดแวร์ที่ใช้เชื่อมต่อ



รูปที่ 2.8 SIP Protocol Stack

2) ชั้นควบคุมเครือข่ายบนระบบอินเตอร์เน็ต (Internet Layer)

เป็นชั้นที่ทำหน้าที่เลือกเส้นทางการรับส่งข้อมูลผ่านเครือข่ายจนไปถึงผู้รับข้อมูลโดยใช้โปรโตคอลไอพี (IP Protocol) เมื่อมีการเชื่อมต่อกันแล้วก็จำเป็นต้องมีการกำหนดหรือระบุหมายเลขของอุปกรณ์ทุกชนิดในเครือข่าย เพื่อให้อ้างอิงได้โดยไม่ซ้ำกัน หมายเลขที่ใช้อ้างถึงกันจะใช้เป็นตัวเลขที่เรียกว่าไอพีแอดเดรส (IP Address) เช่น 207.134.3.5 เป็นต้น

3) ชั้นสื่อสารนำส่งข้อมูล (Transport Layer)

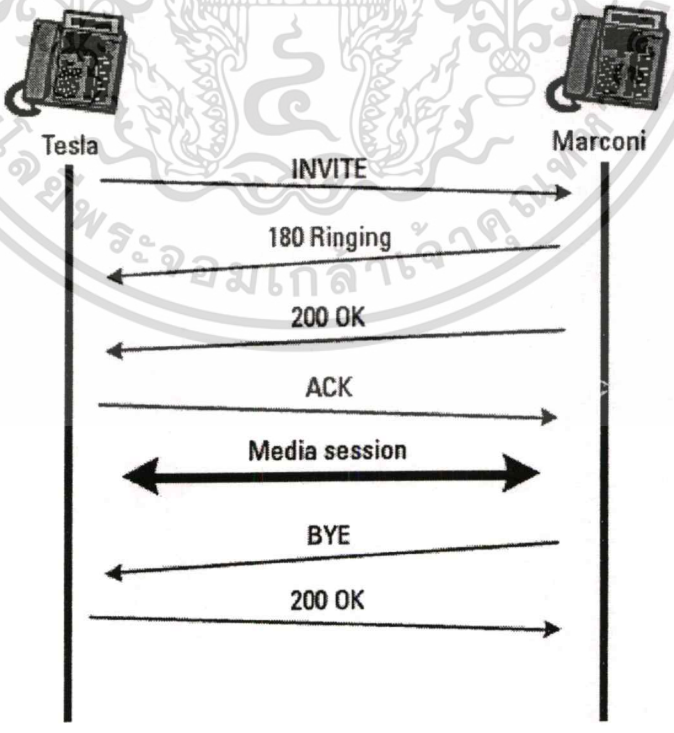
เป็นชั้นที่จะมีการสร้างการเชื่อมต่อกันระหว่างชั้นแอพพลิเคชั่นกับชั้นสื่อสารนำส่งข้อมูล โดยจุดเชื่อมกันเพื่อรับส่งข้อมูลนี้เรียกว่า พอร์ต (port) หรือซ๊อกเก็ต (socket) เมื่อแอพพลิเคชั่นทำงานผ่านโปรโตคอลในชั้นแอพพลิเคชั่น จะมีการส่งผ่านข้อมูลไปยังชั้นสื่อสารนำส่งข้อมูล ที่ในชั้นนี้จะมีการเชื่อมต่อผ่านพอร์ตที่กำหนด เช่น เอชทีทีพี (HTTP) ใช้พอร์ตหมายเลข 80 ส่วนซิพใช้พอร์ตหมายเลข 5060 หรือ 5061 ในชั้นนี้จะมีโปรโตคอลทำงานอยู่ 2 โปรโตคอลที่แตกต่างกันคือทีซีพีและยูดีพี

4) ชั้นสื่อสารโปรแกรมประยุกต์ (Application Layer)

เป็นชั้นที่รองรับการทำงานของแอพพลิเคชั่นต่างๆ และมีการติดต่อกันตามแต่ละโปรโตคอลเฉพาะแล้วแต่แอพพลิเคชั่นที่ใช้งาน เช่น เซสชั่นอินนิชิเอชั่นโปรโตคอล, เรียลไทม์ทรานสปอร์ต (RTP : Real-time Transport Protocol), เอช323โปรโตคอล (H.323) และ เซสชั่นเดสคริปชั่นโปรโตคอล (SDP : Session Description Protocol) ซึ่งอยู่เหนือซิพในโปรโตคอลสแต็ก (Protocol Stack) เพราะ SDP จะอยู่ในเมสเสจบอดี้ (SIP message body)

2.3.2 สถาปัตยกรรมและองค์ประกอบของซิพ (SIP Architecture & Components)

ซิพ เป็นโปรโตคอลแบบ ไคลเอนท์ – เซิร์ฟเวอร์ (Client – Server) โดย ไคลเอนท์จะส่งข้อความร้องขอ (request message) ไปให้กับเซิร์ฟเวอร์เพื่อทำการประมวลผลแล้วจึงส่งข้อความตอบกลับ (response message) กลับมายังไคลเอนท์ดังรูปที่ 2.9



รูปที่ 2.9 ตัวอย่างการสร้าง SIP Session

จากรูปที่ 2.9 เป็นตัวอย่างการสร้าง SIP Session โดยเริ่มจากฝ่ายผู้เรียก (Tesla) ส่ง INVITE message ไปที่ฝ่ายถูกเรียก (Marconi) ซึ่ง INVITE message นี้ประกอบด้วยเนื้อหาของรูปแบบของ session หรือ call ตัวอย่างเช่น อาจจะเป็นเพียง voice (audio) session แบบง่ายๆ, multimedia session เช่น video conference หรือ อาจจะเป็น game session ก็ได้

โดยที่ INVITE message อาจจะเป็นดังนี้

```
1   INVITE sip:marconi@radio.org SIP/2.0
2   Via: SIP/2.0/UDP lab.high-voltage.org:5060;branch=z9hG4bKfw19b
3   Max-Forwards: 70
4   To: G. Marconi <sip:Marconi@radio.org>
5   From: Nikola Tesla <sip:n.tesla@high-voltage.org>;tag=76341
6   Call-ID: 123456789@lab.high-voltage.org
7   CSeq: 1 INVITE
8   Subject: About That Power Outage...
9   Contact: <sip:n.tesla@lab.high-voltage.org>
10  Content-Type: application/sdp
11  Content-Length: 158

12  v=0
13  o=Tesla 2890844526 2890844526 IN IP4 lab.high-voltage.org
14  s=Phone Call
15  c=IN IP4 100.101.102.103
16  t=0 0
17  m=audio 49170 RTP/AVP 0
18  a=rtpmap:0 PCMU/8000
```

ส่วนบนของ INVITE message ตั้งแต่บรรทัดที่ 1-11 เป็นส่วนที่บอกว่าผู้เรียกเป็นใคร ผู้รับเป็นใคร และ IP Address อะไร ส่วนบรรทัดที่ 12-18 เป็นส่วนของ SDP บอกว่าผู้เรียกต้องการเชื่อมต่อด้วยเนื้อหาของมีเดียอะไรบ้าง ที่พอร์ตไหนบ้าง

หลังจากนั้นเมื่อ Marconi ได้รับ INVITE message แล้วก็จะส่ง SIP response message คือ 180 Ringing มีความหมายเหมือนสัญญาณกริ่งโทรศัพท์ดังในโทรศัพท์ปกติ

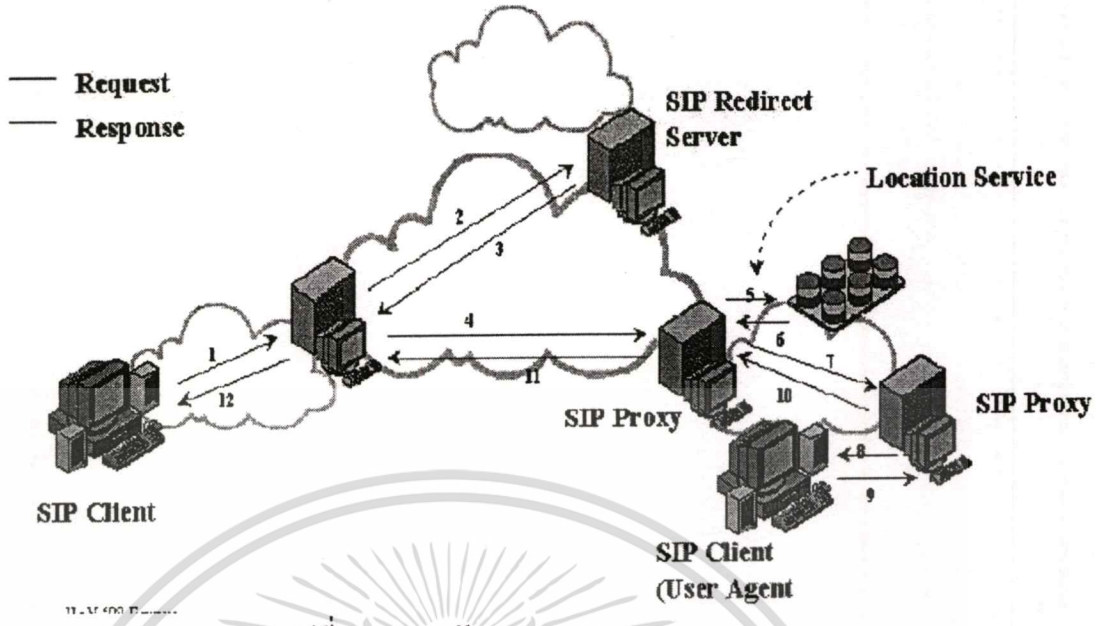
หลังจากนั้น Marconi ตรวจสอบว่ามีเดียใน INVITE message กับของที่ตนมีตรงกันหรือไม่ แล้วจะตอบกลับเพื่อยืนยันการสร้าง Session ด้วย 200 OK response ซึ่งมีเนื้อหาดังนี้

```
1   SIP/2.0 200 OK
2   Via: SIP/2.0/UDP lab.high-voltage.org:5060;branch=z9hG4bKfw19b
3   ;received=100.101.102.103
4   To: G. Mareoni <sip:marconi@radio.org>;tag=a53e42
5   From: Nikola Tesla <sip:n.tesla@high-voltage.org>;tag=76341
6   Call-ID: 123456789@lab.high-voltage.org
7   CSeq: 1 INVITE
8   Contact: <sip:marconi@tower.radio.org>
9   Content-Type: application/sdp
10  Content-Length: 155

11  v=0
12  o=Marconi 2890844528 2890844528 IN IP4 tower.radio.org
13  s=Phone Call
14  c=IN IP4 200.201.202.203
15  t=0 0
16  m=audio 60000 RTP/AVP 0
17  a=rtpmap:0 PCMU/8000
```

ส่วนบนของ 200 OK response ตั้งแต่บรรทัดที่ 1-10 เป็นส่วนที่บอกว่าผู้เรียกเป็นใคร ผู้รับเป็นใคร และ IP Address อะไร ส่วนบรรทัดที่ 11-17 เป็นส่วนของ SDP บอกว่าผู้ถูกเรียกยืนยันการสร้าง session ด้วยเนื้อหาของมีเดียอะไรบ้าง ที่พอร์ตไหนบ้าง

หลังจากนั้นเมื่อผู้เรียกได้รับ 200 OK response แล้วก็จะตอบกลับด้วย ACK การแลกเปลี่ยนมีเดียจึงเริ่มขึ้น เมื่อฝ่ายใดฝ่ายหนึ่งต้องการยุติการแลกเปลี่ยนมีเดียจะส่ง BYE message ไปให้อีกฝ่ายหนึ่ง อีกฝ่ายจะตอบยืนยัน BYE message ด้วย 200 OK response SIP session นี้เป็นอันยุติ



รูปที่ 2.10 สถาปัตยกรรม SIP

ในการส่งแมสเสจร้องขอ แมสเสจอาจจะถูกส่งผ่านเซิร์ฟเวอร์หลายตัวจนกระทั่งถึงเซิร์ฟเวอร์ที่สามารถตอบสนองคำร้องขอของไคลเอนท์ได้ ในระบบ SIP จะมีองค์ประกอบที่ทำหน้าที่ของไคลเอนท์และเซิร์ฟเวอร์ องค์ประกอบเหล่านี้จะทำการติดต่อสื่อสารกันโดยใช้แมสเสจ SIP ซึ่งมีสถาปัตยกรรมดังรูปที่ 2.10

ในซิพจะแบ่งองค์ประกอบเป็น 2 ชนิดหลักคือ ยูสเซอร์เอเจนท์ (User Agent) และ เน็ตเวิร์กเซิร์ฟเวอร์ (Network Server) ดังรายละเอียดต่อไปนี้

1) ยูสเซอร์เอเจนท์

เป็นส่วนที่ทำหน้าที่แทนผู้ใช้ในการติดต่อสื่อสาร (Endpoint) และเนื่องจากผู้ใช้สามารถทำได้ทั้งการร้องขอและการตอบกลับ ดังนั้นยูสเซอร์เอเจนท์ควรที่จะสามารถทำหน้าที่เป็นได้ทั้งไคลเอนท์และเซิร์ฟเวอร์ในกรณีที่ผู้ใช้ทำการร้องขอ ผู้ใช้จะทำหน้าที่เป็นไคลเอนท์เพื่อทำการร้องขอไปยังผู้ถูกเรียกซึ่งจะทำหน้าที่เป็นเซิร์ฟเวอร์ในการตอบสนองการร้องขอ โดยทั่วไปยูสเซอร์เอเจนท์จึงประกอบด้วยส่วนที่ทำหน้าที่เป็นไคลเอนท์และเซิร์ฟเวอร์ดังนี้

1.1) ยูสเซอร์เอเจนท์ไคลเอนท์ (UAC : User agent client) จะทำหน้าที่ในการเริ่มการเรียกโดยการส่งข้อความร้องขอไปยังผู้ถูกเรียกโดยผ่านทางเน็ตเวิร์กเซิร์ฟเวอร์

1.2) ยูสเซอร์เอเจนท์เซิร์ฟเวอร์ (UAS : User agent server) จะทำหน้าที่ในการรับคำร้องขอและตอบสนองต่อคำร้องขอโดยจะรอการตอบสนองจากผู้ใช้ ซึ่งการตอบสนองอาจจะเป็นการยอมรับหรือปฏิเสธการเรียก ในกรณีที่ผู้ใช้มีการใช้งานเทอร์มินัลหลายตัว ผู้ใช้ยังอาจจะกำหนดให้ UAS ทำการปรับทิศทางใหม่ (redirect) ไปยังที่ UAS อื่นที่ผู้ใช้ใช้งานอยู่จริง

2) เน็ตเวิร์กเซิร์ฟเวอร์

เป็นเซิร์ฟเวอร์ภายในเครือข่ายซึ่งจะทำหน้าที่ในการจัดการกับข้อความที่ได้รับโดยอาจจะได้รับจากยูสเซอร์เอเจนท์หรือเน็ตเวิร์กเซิร์ฟเวอร์อื่นๆ การจัดการกับข้อความจะขึ้นกับชนิดของเซิร์ฟเวอร์ ซึ่งมี 2 ชนิดคือ

2.1) พร็อกซี่เซิร์ฟเวอร์ (Proxy server) จะทำการกำหนดเอนทิตี้ที่จะรับข้อมูลต่อไป โดยเอนทิตี้ที่จะรับข้อมูลอาจจะเป็น UAS หรือเน็ตเวิร์กเซิร์ฟเวอร์ก็ได้ จากนั้นเซิร์ฟเวอร์จะเป็นผู้ทำการร้องขอไปยังเอนทิตี้นั้น พร้อมกับส่งข้อมูลตอบสนองให้กับผู้ที่ร้องขอมา เพื่อระบุว่ากำลังรอการตอบสนองจากผู้ถูกเรียก เมื่อเซิร์ฟเวอร์ได้รับการตอบสนองจากผู้ถูกเรียกเซิร์ฟเวอร์จึงจะส่งข้อความตอบกลับไปให้กับผู้ที่ร้องขอมา เซิร์ฟเวอร์ชนิดนี้จะทำหน้าที่เป็นทั้งไคลเอนท์และเซิร์ฟเวอร์ในกรณีที่ส่งข้อความร้องขอ จะเป็น Client ส่วนในกรณีที่ส่ง message request จะเป็น Server

2.2) รีไดเรคท์เซิร์ฟเวอร์ (Redirect server) เมื่อเซิร์ฟเวอร์ได้รับข้อความร้องขอแล้วจะกำหนดเอนทิตี้ที่จะรับข้อมูลต่อไป จากนั้นเซิร์ฟเวอร์จะส่งแอดเดรสของเอนทิตี้นั้นไปให้กับผู้ที่ร้องขอมา เมื่อผู้ที่ร้องขอมาได้รับแอดเดรสไปแล้วจึงจะทำการส่งคำร้องไปยังเอนทิตี้นั้นด้วยตนเอง เนื่องจากว่าผู้ใช้อาจจะมีการเปลี่ยนเทอร์มินัลที่ใช้งานได้ เน็ตเวิร์กเซิร์ฟเวอร์จะต้องสามารถกำหนดเอนทิตี้ที่รับข้อมูลเพื่อให้ สามารถส่งข้อความไปให้กับผู้ถูกเรียกได้เมื่อมีการเปลี่ยนเทอร์มินัล เน็ตเวิร์กเซิร์ฟเวอร์จะทำการติดต่อกับโลเคชั่นเซิร์ฟเวอร์ (location server) เพื่อกำหนดเอนทิตี้ต่อไปที่จะรับข้อมูล โลเคชั่นเซิร์ฟเวอร์จะทำหน้าที่ในการหาตำแหน่งปัจจุบันของผู้ถูกเรียกโดยการกำหนดเอนทิตี้ที่จะรับข้อความต่อไปแล้วส่งแอดเดรสของเอนทิตี้นี้ให้กับเน็ตเวิร์กเซิร์ฟเวอร์ ข้อมูลของโลเคชั่นเซิร์ฟเวอร์จะได้รับจากรีจิสตราร์เซิร์ฟเวอร์ (registrar server) ซึ่งทำหน้าที่ในการรับข้อมูลเกี่ยวกับตำแหน่งของผู้ใช้ แล้วส่งข้อมูลนี้ไปโลเคชั่นเซิร์ฟเวอร์ ในการให้ข้อมูลของผู้ใช้กับรีจิสตราร์เซิร์ฟเวอร์จะทำได้โดยใช้ข้อความ REGISTER เพื่อบอกตำแหน่งที่อยู่ของผู้ใช้ โดยทั่วไปแล้วรีจิสตราร์เซิร์ฟเวอร์ จะถูกรวมเข้ากับเน็ตเวิร์กเซิร์ฟเวอร์

2.3.3 ชื่อและแอดเดรส (Addressing & Naming)

ในระบบ ซิพ การส่งข้อความระหว่างเอนทิตี้จะต้องระบุ ซิพยูอาร์แอล (SIP URL) เพื่อใช้อ้างอิงถึงผู้ใช้ ซิพยูอาร์แอลจะประกอบด้วย ซิพแอดเดรส รูปแบบของแอดเดรสจะอยู่ในรูปของ name@domain โดยอาจจะเป็น user@domain, user@address, phone-number@gateway และ user@host แอดเดรสนี้จะถูกใช้อ้างอิงถึงผู้ใช้ ทั้งผู้เรียกและผู้ถูกเรียกในการส่งข้อความตัวอย่างของซิพยูอาร์แอล เช่นsip://j.doe@example.com โดยที่ ยูอาร์แอล (URL) นี้จะอยู่ในส่วนของ header ในการส่งข้อความไปยังซิพยูอาร์แอลที่ระบุไว้จะต้องมีการแปลง ซิพแอดเดรสให้อยู่ในของ user@host โดยอาจจะผ่านการแปลงมากกว่าหนึ่งครั้งจนกระทั่งได้ตำแหน่งที่อยู่ของผู้ใช้ ในการ

แปลงแอดเดรสอาจจะใช้ดีเอ็นเอส (DNS : Domain Name Service) หรือ แอลแด็ป (LDAP : Lightweight Directory Access Protocol)

2.3.4 Locating Server

ในการส่งแมสเสจจะใช้ SIP URL อ้างอิงถึงในการส่ง โดยจะต้องมีการแปลงส่วนdomain ของซิพแอดเดรสไปเป็นหมายเลข IP ซึ่งเป็น แอดเดรสของ SIP server ที่สามารถค้นหาตำแหน่งของผู้ใช้ต่อไปได้ การแปลงซิพแอดเดรสอาจจะทำโดย UAC หรือ UAC จะส่งแมสเสจให้กับเซิร์ฟเวอร์ที่กำหนดซึ่งเซิร์ฟเวอร์จะเป็นผู้ที่ทำหน้าที่ในการแปลงซิพแอดเดรสแทน ในการแปลงซิพแอดเดรสนี้สามารถใช้ DNS เข้ามาช่วยได้

2.3.5 Locate User

จากข้างต้นเมื่อได้ตำแหน่งของเซิร์ฟเวอร์ที่สามารถส่งข้อมูลให้กับผู้ถูกเรียกและต่อไปจะเป็นการหาตำแหน่งของผู้ถูกเรียก เมื่อ SIP server ได้รับแมสเสจร้องขอแล้ว เซิร์ฟเวอร์จะต้องทำการค้นหาผู้ใช้ที่อ้างอิงถึงใน SIP แอดเดรส โดยการร้องขอข้อมูลไปยัง location server ซึ่งจะตอบกลับด้วยรายการตำแหน่งที่เป็นไปได้ของผู้ถูกเรียก เมื่อ SIP server ได้ข้อมูลเกี่ยวกับตำแหน่งของผู้ถูกเรียกแล้ว ถ้าเป็น proxy server จะทำส่งแมสเสจร้องขอต่อไปยังตำแหน่งต่างๆ ตามรายการที่ได้รับจาก location server ไว้ โดยอาจจะส่งแบบ sequential หรือ parallel ส่วนถ้าเป็น redirect server จะส่งรายการตำแหน่งของผู้ถูกเรียกไปให้ผู้เรียกโดยใช้เฮดเดอร์ contact เพื่อให้ผู้เรียกส่งแมสเสจร้องขอไปเอง สำหรับตำแหน่งของผู้ใช้จะต้องทำการลงทะเบียนกับ registrar โดยใช้เฮดเดอร์ REGISTER รวมทั้งยังอาจจะอัพโหลด script ของผู้ใช้เองเพื่อเก็บไว้ที่เซิร์ฟเวอร์สำหรับจัดการกับการเรียก ตามความต้องการของผู้ใช้

2.3.6 ความเชื่อถือได้ (Reliability)

ในระบบซิพจะมีกลไกเรื่องความเชื่อถือได้ (reliability) ไม่ว่าจะใช้โปรโตคอล UDP หรือ TCP โดยการใช้เมธอด Ack ไคลเอนท์จะส่งแมสเสจร้องขอใหม่ตามช่วงเวลาที่กำหนดจนกระทั่งได้รับแมสเสจตอบจากเซิร์ฟเวอร์ ทางด้านเซิร์ฟเวอร์ก็จะส่งแมสเสจตอบจนกระทั่งได้รับแมสเสจ Ack จากไคลเอนท์จึงทำให้การร้องขอที่สมบูรณ์ต้อใช้การแลกเปลี่ยนแมสเสจ 3 แมสเสจ เซิร์ฟเวอร์อาจจะตอบสนองต่อ Ack โดยการส่งแมสเสจตอบสุดท้ายไปให้กับไคลเอนท์ซึ่งอาจจะไม่จำเป็นต้องมีก็ได้ สำหรับการส่งมีเดียสตรีมเซิร์ฟเวอร์จะยอมให้มีการส่งเมื่อได้รับ Ack จากไคลเอนท์เท่านั้น ด้วยกลไกนี้จึงทำให้เกิดความเชื่อถือได้ในการแลกเปลี่ยนแมสเสจโดยไม่จำเป็นต้องอาศัยกลไกของโปรโตคอลในชั้นต่ำกว่า เช่น TCP

2.3.7 ความสามารถในการขยาย (Protocol extension)

SIP สามารถรองรับคุณลักษณะใหม่ที่เพิ่มเติมขึ้นสำหรับ เมธอด เฮดเดอร์ และ status code ได้ดังนี้

- เมธอด เซิร์ฟเวอร์จะส่งแมสเสจแสดงความผิดพลาด (error message) กลับมาให้ไคลเอนท์ถ้าเมธอดที่ร้องขอมาเซิร์ฟเวอร์ไม่เข้าใจ และจะบอกเมธอดที่เซิร์ฟเวอร์เข้าใจโดยใช้เฮดเดอร์ Public และ Allow ไคลเอนท์อาจจะส่งแมสเสจร้องขอเพื่อขอทราบเมธอดที่เซิร์ฟเวอร์สนับสนุนโดยใช้ตัวเลือกที่เฮดเดอร์ (header option)
- เฮดเดอร์ เมื่อเอนทิตี้ได้รับเฮดเดอร์ที่ไม่เข้าใจ ก็จะละทิ้งเฮดเดอร์นั้น ในกรณีที่ไคลเอนท์จำเป็นต้องการใช้เฮดเดอร์บางเฮดเดอร์ ไคลเอนท์จะส่งแมสเสจเพื่อร้องขอเฮดเดอร์ที่จำเป็นต้องใช้ไปโดยระบุในเฮดเดอร์ Require หากมีเฮดเดอร์ที่เซิร์ฟเวอร์ไม่สามารถให้การสนับสนุนได้เซิร์ฟเวอร์จะตอบปฏิเสธกลับมา
- status code ได้แบ่งเป็นคลาสต่างๆ เช่นเดียวกับ response code ของโปรโตคอล HTTP ซึ่งไคลเอนท์ต้องเข้าใจในความหมายในแต่ละคลาสเพื่อที่จะได้ทราบผลของการร้องขอว่าสำเร็จหรือไม่ สำหรับ status code ในแมสเสจตอบจะมีข้อความต่อหลังซึ่งจะเป็นความหมายของ code ซึ่งสามารถอ่านเข้าใจได้ โดยถ้าไคลเอนท์ไม่เข้าใจในรายละเอียดของ code ทั้งหมด ไคลเอนท์จะตีความหมายเป็น X00 เมื่อ X เป็นตัวเลขตัวแรกของ status code และนอกจากนั้นอาจจะนำ PEP (protocol extension protocol) มาปรับปรุงใช้งานกับ SIP ได้

ในกรณีมีการส่งแมสเสจผ่านหลายเซิร์ฟเวอร์ จะใช้เฮดเดอร์ Via เพื่อระบุเซิร์ฟเวอร์ที่เป็นทางผ่านของแมสเสจทั้งหมด สำหรับใช้ในการส่งแมสเสจตอบสนองกลับไปให้ผู้เรียกในระหว่างการส่งแมสเสจร้องขอ

และแมสเสจตอบสนองจะมีการตกลงเกี่ยวกับพารามิเตอร์ของเซสชันด้วย ซึ่งรายละเอียดจะอยู่ในส่วนของ message body เช่นในกรณีของการสื่อสารโดยใช้เสียง พารามิเตอร์จะเป็น IP แอดเดรส พอร์ตสำหรับ RTP และการเข้า/ถอดรหัสเสียง หลังการสร้างการเรียกเสร็จสมบูรณ์ ช่องสัญญาณสำหรับ RTP จะถูกสร้างขึ้นทำให้ทั้งสองฝ่ายสามารถสื่อสารกันได้ รวมทั้งยังอาจจะเชิญผู้อื่นมาเข้าร่วมในเซสชันนี้ได้ ในกรณีที่ต้องการเปลี่ยนพารามิเตอร์ของเซสชัน สามารถทำได้โดยส่งแมสเสจร้องขอใหม่อีกครั้งโดยใช้เมธอด Invite ซึ่งมี call-id เดิมไปยังผู้ร่วมเซสชันพร้อมทั้งค่าพารามิเตอร์ของเซสชันใหม่ที่ต้องการใช้ รายละเอียดในส่วนนี้จะอยู่ในส่วนของ message body ซึ่งโดยทั่วไปจะใช้โปรโตคอล SDP ในการอธิบายความหมาย

2.3.8 แมสเสจเอสดีพี (SDP Message)

เอสดีพีเป็นโปรโตคอลที่บรรยายลักษณะของการเชื่อมต่อ อธิบายเกี่ยวกับลักษณะของเซสชั่นที่ต้องการเปิด ซึ่งจำเป็นสำหรับซิพ เพราะซิพไม่ได้กำหนดชนิดของสื่อที่จะส่ง ดังนั้นผู้ส่งจะต้องบอกลักษณะของสื่อว่าเป็นชนิดไหน เช่น เสียง, วีดีโอ หรือข้อมูล โดยใช้การส่งข้อมูลในรูปของตัวอักษร (text based) เช่นเดียวกับซิพ

การใช้เอสดีพีในซิพนั้น จะใช้เมื่อต้องการสร้างการเชื่อมต่อ โดยส่งไปพร้อมกับอินไวท์แมสเสจ (INVITE message) หรือ แอคแมสเสจ (ACK message) ซึ่งเอสดีพีจะระบุชนิดของสื่อต่างๆ ที่ยอมรับได้ และจะต้องยอมรับได้ทั้งสองฝ่ายจึงจะสามารถสร้างการเชื่อมต่อได้ นอกจากนี้ยังระบุหมายเลขพอร์ตและ ไอพีแอดเดรสที่จะใช้ในการสร้างการเชื่อมต่อกันอีกด้วย

อธิบายแต่ละฟิลด์ของเอสดีพีได้ดังต่อไปนี้

1) Protocol Version (v)

ใน v จะประกอบด้วยตัวเลขของเวอร์ชั่นและจะต้องมีค่าเป็น v=0 เพราะว่าเวอร์ชั่นปัจจุบันของ เอสดีพีคือ 0

2) Origin (o)

ใน o จะประกอบด้วยข้อมูลเกี่ยวกับผู้สร้างเซสชั่นและเซสชั่นไอดีที่ไม่ซ้ำ

o : note 3300619822 3300619823 IN IP4 161.246.5.202

o : [username] [session-id] [version] [network-type] [address-type] [address]

username คือ ชื่อผู้ใช้หรือโฮสต์หรือ – ถ้าไม่มี

session-id คือ ใช้ตัวเลขสุ่มที่ต้องที่ต้องไม่ซ้ำ

version คือ ตัวเลขที่เพิ่มขึ้นเมื่อมีการเปลี่ยนแปลง session

network-type เป็น IN เสมอสำหรับอินเตอร์เน็ต

address-type ใช้ IP4 สำหรับ IPv4 หรือ IP6 สำหรับ IPv6

address ใช้เป็นไอพีแอดเดรสหรือชื่อของโฮสต์ก็ได้

3) Session Name and Information (s)

ใน s จะประกอบด้วยชื่อสำหรับเซสชั่นซึ่งแสดงข้อมูลเกี่ยวกับเซสชั่นนั้นๆ เช่น SipSession Phone call เป็นต้น

4) Connection Data (c)

ใน c จะประกอบด้วยข้อมูลเกี่ยวกับการติดต่อสื่อสาร

c : IN IP4 161.246.5.202

c : [network-type] [address-type] [connection-address]

network-type เป็น IN เสมอสำหรับอินเตอร์เน็ต

address-type ใช้ IP4 สำหรับ IPv4 หรือ IP6 สำหรับ IPv6

connection-address คือ ไอพีแอดเดรสที่ต้องการติดต่อ

5) Time, Repeat Times, and Time Zones (t)

ใน t จะประกอบด้วยเวลาเริ่มต้นและ เวลาหยุดของเซสชั่น

t=start-time stop-time

เวลาจะถูกระบุโดยใช้ NTP(Network Time Protocol) timestamp ถ้าเวลาหยุดเป็นศูนย์ หมายความว่าเป็นเซสชั่นที่ทำไปเรื่อยๆ ไม่มีกำหนด ถ้าทั้งเวลาเริ่มต้นและ เวลาหยุดเป็นศูนย์ หมายความว่าเป็นเซสชั่นถาวร

6) Media Announcements (m)

ใน m ประกอบด้วยข้อมูลเกี่ยวกับชนิดของสื่อ

m : application 49152 TCP Warship

m : [media] [port] [transport] [format-list]

media ใช้ เสียง, วีดีโอ, แอพพลิเคชั่น, ข้อมูล, การโทรศัพท์, หรือการควบคุม

port คือ หมายเลขพอร์ต

transport คือ โปรโตคอลสื่อสารเช่น อาร์ทีพี, ทีซีพี หรือยูดีพี

format-list คือ ข้อมูลอื่นๆ เกี่ยวกับสื่อ

7) Attributes (a)

ใน a ประกอบด้วยแอททริบิวท์ (attributes) ของสื่อ เช่น

a=rtpmap:0 PCMU/8000

a=rtpmap:6 DVI4/16000

a=rtpmap:8 PCMA/8000

rtpmap คือ RTP/AVP list เช่น a=rtpmap:0 PCMU/8000 มี Media encoding เป็น PCM μ Law และ Sampling rate 8000 Hz

2.4 การเปรียบเทียบระหว่าง SIP และ H.323 (SIP and H.323 comparisons)

ในปัจจุบันมีโปรโตคอลสำหรับ IP telephony คือ H.323 ซึ่งพัฒนาโดย ITU และซิพซึ่งพัฒนาโดย IETF ดังที่ได้กล่าวมาแล้วโปรโตคอล H.323 ได้ถูกพัฒนาขึ้นก่อนจึงทำให้มีการใช้งานโปรโตคอลนี้มากกว่าซิพซึ่งถูกพัฒนาขึ้นในภายหลัง แต่อย่างไรก็ตามโปรโตคอลทั้งสองก็มีทั้งข้อดีและข้อเสีย ในที่นี้จะพิจารณาในแง่ของความซับซ้อน (complexity) ความสามารถในการขยายขนาดของเครือข่าย (scalibility) ความสามารถในการเพิ่มเติมคุณลักษณะของโพรโตคอล (extensibility) ฟังก์ชันและการบริการ (functionality and services) คุณภาพการให้บริการ (QoS) และการทำงานร่วมกัน (interoperability)

การเปรียบเทียบระหว่าง H.323 และซิพในแง่ต่างๆ สามารถสรุปได้ใน [28] จากการเปรียบเทียบในข้างต้นจะเห็นว่าซิพสามารถให้บริการได้คล้ายกับ H.323 แต่มีความซับซ้อนน้อยกว่า ความสามารถในการเพิ่มเติมคุณลักษณะของโปรโตคอล (extensibility) มากกว่า และสามารถรองรับขนาดของเครือข่ายได้ใหญ่กว่า (scalibility) เมื่อพิจารณาถึงการพัฒนาของทั้งสองโปรโตคอล การพัฒนาจะเป็นในลักษณะที่ เรียนรู้ซึ่งกันและกัน เช่น H.323 เวอร์ชัน 3 จะมีความใกล้เคียงกับซิพจึงอาจจะเป็นไปได้ว่าโปรโตคอลทั้งสองอาจจะมีประสิทธิภาพในการทำงานใกล้เคียงกันแต่อย่างไรก็ตาม H.323 เป็นมาตรฐานที่เกิดขึ้นก่อน ดังนั้นในปัจจุบันจึงมีการใช้งาน H.323 มากกว่า ในขณะที่ซิพเป็น โปรโตคอลใหม่จึงยังมีการใช้งานน้อยกว่า สำหรับ H.323 มีข้อได้เปรียบคือ ITU-T ซึ่งเป็นผู้พัฒนา H.323 เป็นผู้กำหนดมาตรฐานต่างๆ ในระดับล่าง ลงไปถึงในชั้นกายภาพ (physical layer) ในขณะที่ IETF ซึ่งเป็นผู้พัฒนาซิพจะเกี่ยวข้องเฉพาะในชั้นเครือข่าย (network layer) ขึ้นไป H.323 จึงอาจจะมีความเข้ากันได้หรือประสิทธิภาพในการทำงานร่วมกับเครือข่ายได้ดีกว่าซิพ ส่วนข้อเสียของ H.323 คือ H.323 ค่อนข้างจะอ้างอิงไปทางแบบจำลอง circuit-switch จึงทำให้มีราคาสูงและยุ่งยากในการใช้งานจริง ในขณะที่ซิพสามารถใช้งานได้ง่ายและมีราคาถูกกว่า ตารางที่ 2.2 สรุปการเปรียบเทียบข้อจำกัดระหว่าง H.323 และซิพ จากการพิจารณาข้างต้นจะเห็นว่าโปรโตคอลทั้งสองต่างก็มีทั้งข้อดีและข้อเสียจึงไม่อาจบอกได้ชัดว่าโปรโตคอลใดจะเป็นโปรโตคอลที่เหมาะสมสำหรับ IP telephony มากที่สุด และนอกจากนั้นก็ยังมีแนวโน้มว่าอาจจะใช้โปรโตคอลทั้งสองทำงานร่วมกัน ตารางที่ 2.3 [29] แสดงองค์ประกอบเปรียบเทียบระหว่าง H.323 และซิพจะเห็นได้ว่ามีข้อแตกต่างกันหลายประการ

ตารางที่ 2.1 สรุปการเปรียบเทียบการทำงานระหว่าง H.323 และ SIP

	H.323 v1	H.323 v2	H.323 v3	SIP
FUNCTIONALITY				
CALL CONTROL SERVICES:				
Call Holding	No	Yes	Yes	Yes
Call Transfer	No	Yes	Yes	Yes
Call Forwarding	No	Yes	Yes	Yes
Call Waiting	No	Yes	Yes	Yes
ADVANCED FEATURES:				
Third Party Control	No	No	No	Yes

ตารางที่ 2.1 (ต่อ)

	H.323 v1	H.323 v2	H.323 v3	SIP
Conference	Yes	Yes	Yes	Yes
Click-for-Dial	Yes	Yes	Yes	Yes
Capability Exchange	Yes&Better	Yes&Better	Yes&Better	Yes
QUALITY OF SERVICE				
Call Setup Delay	6~7 RT	3~4 RT	2~3 RT	2~3 RT
RELIABILITY:				
Packet Loss Recovery	ThroughTCP	ThroughTCP	ThroughTCP	ThroughTCP
Fault Detection	Yes	Yes	Yes	Yes
Fault Telerance	N/A	N/A	Better	Good
MANAGEABILITY				
Admission Control	Yes	Yes	Yes	No
Policy Control	Yes	Yes	Yes	No
ResourceReservation	No	No	No	No
SCALABILITY				
Complexity	More	More	More	Less
Server Processing	Stateful	Stateful	Stateful or Stateless	Stateful or Stateless
Inter-Server Communication	No	No	Yes	Yes
FLEXIBILITY				
Transport Protocol Neutrality	TCP	TCP	TCP/UDP	TCP/UDP
Extensibility of Functionality	Vender Specified			Yes, IANA
Ease of Customization	Harder	Harder	Harder	Easier
INTEROPERABILITY				
Version Compatibility	N/A	Yes	Yes	Unknown
SCN Signaling Interoperability	Better	Better	Better	Worse
EASE OF IMPLEMENTATION				
Protocol Encoding	Binary	Binary	Binary	Text

ตารางที่ 2.2 สรุปการเปรียบเทียบระหว่าง H.323 และ SIP

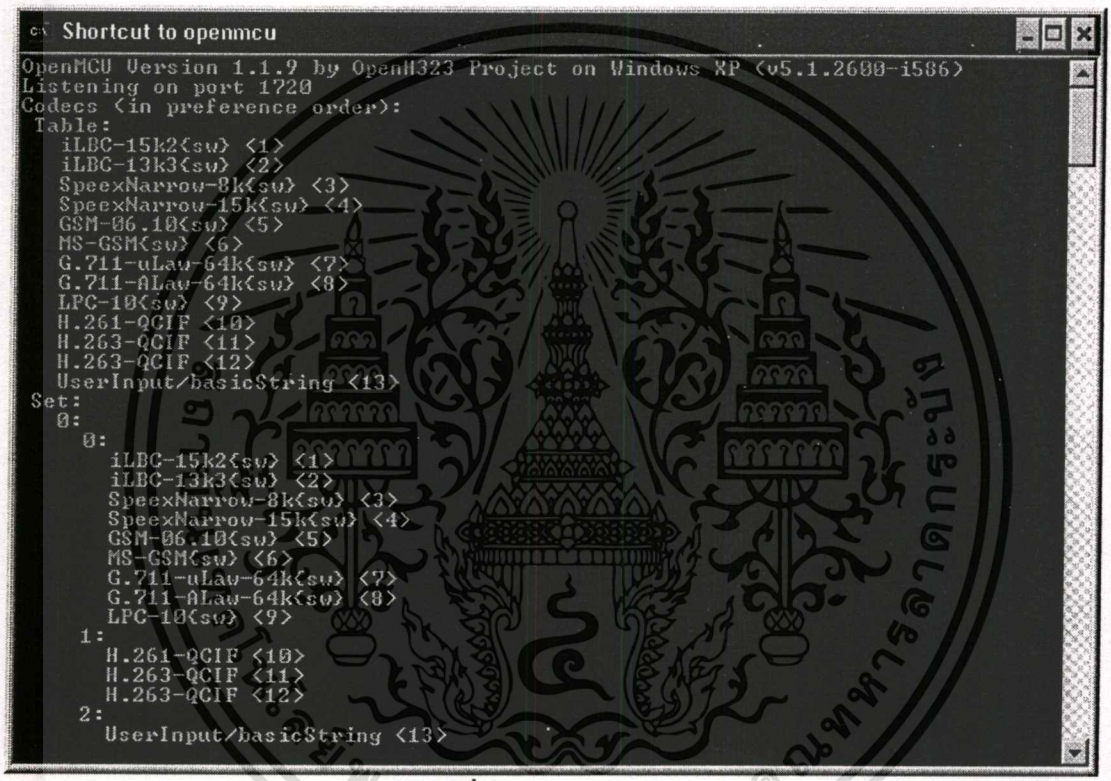
Capability	Protocol	
	H.323	SIP
Cost	High	Low
Complexity	High	Low
Maturity	Good	Poor
Scope of Definition	Full	Limited
Interoperability	Good	Some
SS7 Compatibility	Poor	Poor
Similar to ISDN	Yes	No

ตารางที่ 2.3 สรุปการเปรียบเทียบองค์ประกอบระหว่างระหว่าง H.323 และ SIP

Requirement	The H.323 Way	The SIP Way
Call signaling	H.255/Q.931	SIP INVITE Request/Response, ACK, BYE
Media Negotiation	H.245 Capability exchange or Q.931 Fast Start Method	SDP embedded in SIP INVITE Req/Rsp, ACK
Registration, Location and admission service	H.255 RAS Protocol to gatekeeper	SIP REG request, to a variety of location services. Proxies locate things in processing an INVITE.
Media Transmission	RTP	RTP
Centralized Call Control	Gatekeeper-routed call signaling	Use of SIP proxies

2.5 OpenMCU

OpenMCU ดังรูปที่ 2.11 เป็นโอเพ่นซอร์ส H.323 MCU ซึ่งใช้ในงานวิจัยนี้ โดยทำงานบนไลบรารีหลัก 2 ตัวคือ PwLib.dll และ OpenH323.dll โดย PwLib.dll จะบรรจุคลาสต่างๆสำหรับการเขียนโปรแกรมข้ามแพลตฟอร์ม ส่วน OpenH323.dll จะบรรจุคลาสต่างๆที่เกี่ยวกับ H.323 ทั้งหมดไม่ว่าจะเป็นการติดต่อ, การยกเลิกการติดต่อ, codec ต่างๆทั้งออดิโอและวิดีโอ โดย version ที่ใช้ในการวิจัยนี้คือ OpenH323 เวอร์ชัน 1.13.5-1, pwlib เวอร์ชัน 1.6.6-1 และ OpenMCU เวอร์ชัน 1.13.4 รายละเอียดของ OpenH323 และ PwLib จะกล่าวถึงอย่างละเอียดในภาคผนวก ก.



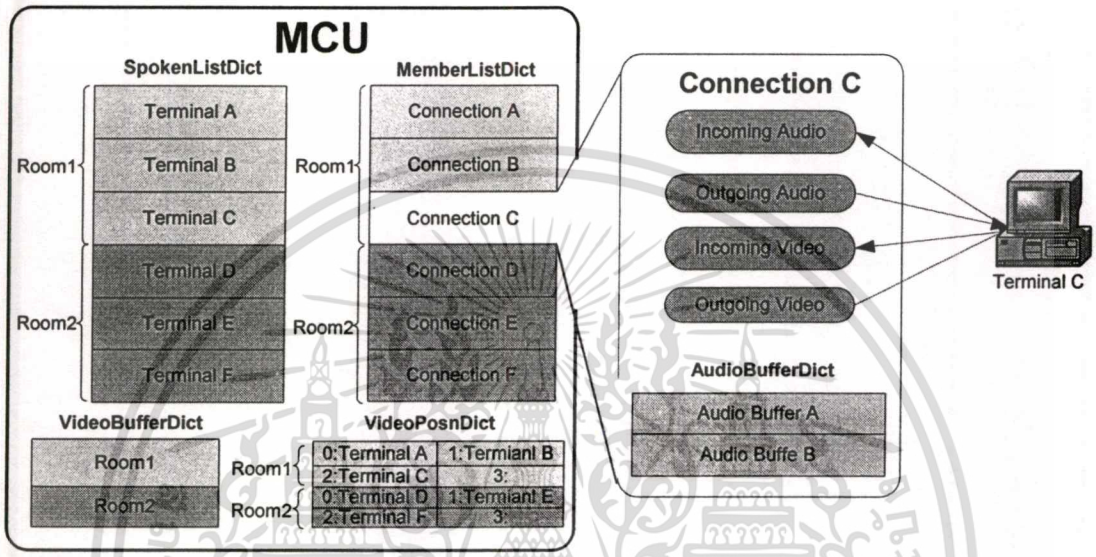
รูปที่ 2.11 โปรแกรม OpenMCU

2.5.1 โครงสร้างข้อมูลและการทำงานของ OpenMCU

OpenMCU มีส่วนประกอบหลักดังรูปที่ 2.12 โดยประกอบด้วยส่วนประกอบ 4 ส่วนคือ SpokenListDict, MemberListDict, VideoBufferDict และ VideoPosnDict เราจะเห็นว่าส่วนประกอบทั้งสี่มีคำว่า Dict ตามหลัง คำว่า Dict นี้มีความหมายว่าส่วนประกอบทั้ง 4 มีลักษณะคล้าย Dictionary คือเป็นที่เก็บออบเจ็กต์โดยใช้คีย์ซึ่งเป็นสตริงในการค้นหา

MemberListDict เป็น Dict ที่ใช้เก็บ MemberList โดยใช้คีย์เป็นชื่อของห้อง MemberList นี้เป็นลิสต์ที่เก็บออบเจ็กต์คอนเน็คชัน โดยแต่ละคอนเน็คชันจะทำหน้าที่ติดต่อกับเทอร์มินัล ออบเจ็กต์คอนเน็คชันประกอบด้วย Incoming Audio มีหน้าที่รับออดิโอสตรีมที่เข้ามา, Outgoing Audio มีหน้าที่ส่งออดิโอสตรีมไปสู่เทอร์มินัล, Incoming Video มีหน้าที่รับวิดีโอสตรีมที่เข้ามาและ

Outgoing Video ทำหน้าที่ส่งวิดิโอสตรีมไปสู่เทอร์มินัล ภายในออบเจ็กต์คอนเน็คชันนี้ยังประกอบไปด้วย AudioBufferDict ซึ่งเป็น Dict ที่ใช้เก็บออดิโอบัฟเฟอร์ของทุกคอนเน็คชันยกเว้นของตัวมันเอง (ป้องกัน Echo) เพื่อใช้ประโยชน์ในการผสมออดิโอสตรีมที่เข้ามาแล้วส่งไปให้เทอร์มินัลต่างๆ โดยเราสามารถแบ่งการทำงานของ OpenMCU ออกเป็น 2 ส่วนคือ ส่วนออดิโอ และส่วนวิดีโอ ดังนี้

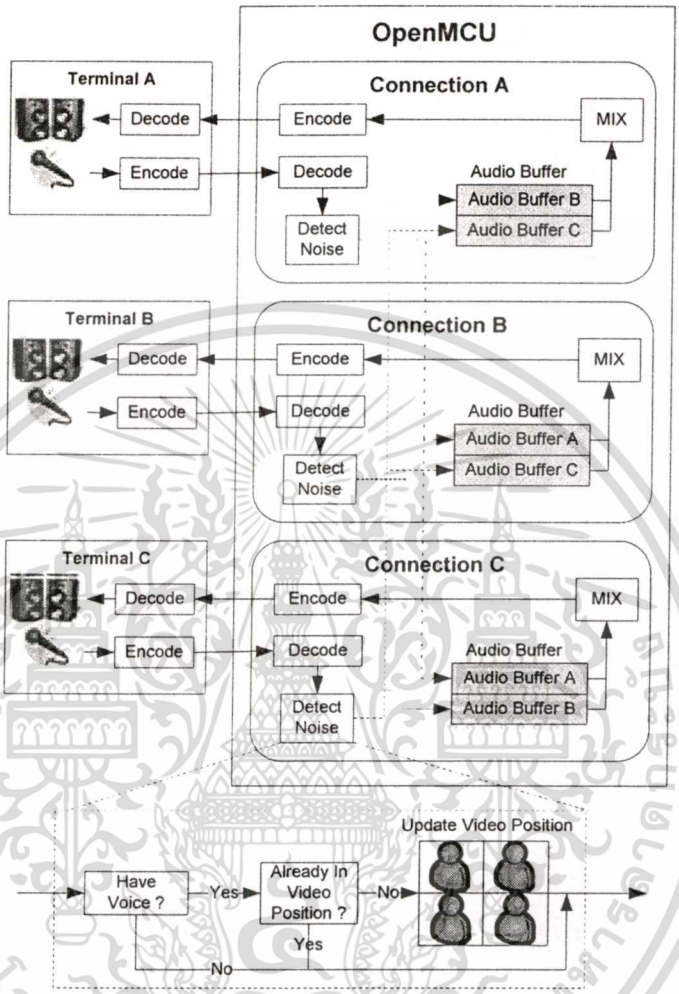


รูปที่ 2.12 ส่วนประกอบหลักของ OpenMCU

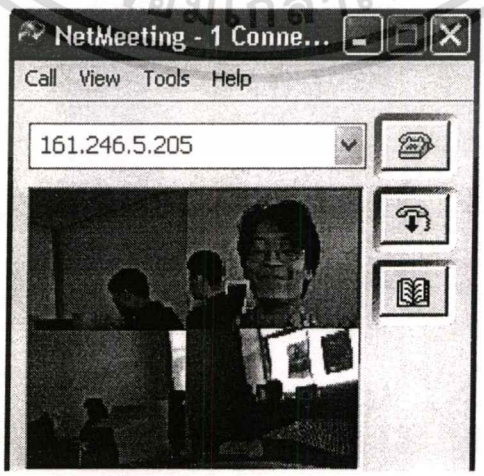
1) ส่วนออดิโอ

การทำงานของส่วนออดิโอสามารถแสดงได้ดังรูปที่ 2.13 โดยเทอร์มินัลต่างๆเชื่อมต่อกับ OpenMCU ผ่านทางคอนเน็คชันออบเจ็ก โดยเทอร์มินัลต่างๆจะบีบอัดออดิโอสตรีมที่ได้มาแล้วส่งต่อให้ OpenMCU หลังจากที่คอนเน็คชันออบเจ็กใดๆ ได้รับออดิโอสตรีมจากเทอร์มินัลแล้วก็จะทำการดีโค้ดแล้วใส่ในบัฟเฟอร์หลังจากนั้นจะนำบัฟเฟอร์นี้ไปยังฟังก์ชัน DetectNoise โดยฟังก์ชัน DetectNoise จะหาว่าผู้ใช้ที่เทอร์มินัลนี้พูดหรือไม่ ถ้าพูดก็จะทำการปรับปรุง SpokenList (หาได้จาก SpokenListDict โดยใช้คีย์เป็นชื่อห้อง) โดยใส่ชื่อของเทอร์มินัลนี้ที่ท้ายลิสต์ หลังจากนั้นก็จะทำการตรวจสอบว่าเทอร์มินัลนี้อยู่ใน 4 อันดับแรกของ VideoPosnList (หาได้จาก VideoPosnDict โดยใช้คีย์เป็นชื่อห้อง) หรือไม่ ถ้าไม่อยู่ก็จะแทนที่เทอร์มินัลนี้กับเทอร์มินัลที่พูดในอันดับแรกสุด (โดยดูจากเทอร์มินัลที่อยู่ในอันดับ 5 จากท้ายของ SpokenList) ใน VideoPosnList โดย VideoPosnList นี้ใช้ประโยชน์ในการผสมวิดีโอสตรีมจากผู้ใช้ที่พูดล่าสุดเป็น 4 ช่องดังรูปที่ 2.14 หลังจากนั้น OpenMCU จะสำเนาออดิโอบัฟเฟอร์นี้ไปใส่ในออดิโอบัฟเฟอร์นี้ของทุกคอนเน็คชัน

หลังจากนั้นเมื่อถึงเวลาที่กำหนด (เวลา Playout Time) ก็จะนำทุกออดิโอบัฟเฟอร์ภายใน AudioBufferDict ในแต่ละคอนเน็คชันมาผสมกัน จากนั้นทำการบีบอัดและส่งกลับไปให้เทอร์มินัล



รูปที่ 2.13 Block Diagram การทำงานของ OpenMCU ในส่วนออดิโอ



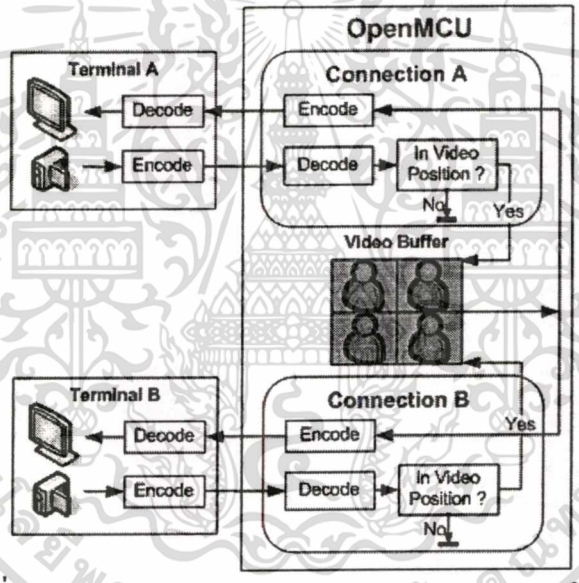
รูปที่ 2.14 การผสมภาพเมื่อเชื่อม Microsoft Netmeeting กับ OpenMCU

จะเห็นว่าทุกคอนเน็คชันมีออดิโอบัฟเฟอร์ N-1 บัฟเฟอร์ เมื่อนับทุกคอนเน็คชันแล้วก็จะมีออดิโอบัฟเฟอร์เป็น N(N-1) ดังนั้น สเปซ คอมเพล็กซิตี (Space Complexity) ของโปรแกรมจึงมีค่าเท่ากับ

$$O_{space} = O(N \times (N-1)) = O(N^2) \tag{2.1}$$

โดยที่ N คือจำนวนคอนเน็คชันหรือผู้ใช้ การที่จำนวนออดิโอบัฟเฟอร์เป็น $O(N^2)$ ทำให้จำนวนการสำเนาและการผสมออดิโอมี Run-Time คอมเพล็กซิตีเท่ากับ $O(N^2)$ ด้วย ซึ่งจะเห็นว่าเมื่อผู้ใช้มีจำนวนมากๆจะทำให้ OpenMCU ทำงานหนักมากส่งผลให้มีซีพียูโหลดที่สูง

2) การทำงานของส่วนวิดีโอ



รูปที่ 2.15 Block Diagram การทำงานของ OpenMCU ในส่วนวิดีโอ

การทำงานของส่วนวิดีโอแสดงดังรูปที่ 2.15 เมื่อ OpenMCU รับวิดีโอสตรีมมาจากเทอร์มินัลแล้วจากนั้นดีโคด แล้วตรวจสอบว่าอยู่ใน 4 อันดับแรกของ VideoPosnList หรือไม่ ถ้าใช่ก็ทำการ Down Sampling ข้อมูลวิดีโอแล้วสำเนาลงในวิดีโอบัฟเฟอร์ในช่องที่เหมาะสม จากนั้นเมื่อถึงเวลาที่เหมาะสม (เวลา Playout Time) ก็จะนำข้อมูลวิดีโอจากบัฟเฟอร์ทั้ง 4 ช่องมาบีบอัดจากนั้นส่งกลับไปให้กับเทอร์มินัลต่างๆ ดังแสดงในรูปที่ 2.14 จากการทำงานของส่วนนี้จะเห็นว่าไม่ว่าจะมีกี่คอนเน็คชัน คอมเพล็กซิตีของการสำเนาวิดีโอลงบัฟเฟอร์ก็จะเป็น O(1) เท่านั้นซึ่งดีอยู่แล้ว แต่ที่เวลาใดๆ OpenMCU จะแสดงผลเฉพาะข้อมูลวิดีโอของ 4 คนที่พูดล่าสุดเท่านั้นส่วนคนอื่นๆที่อยู่

นอกเหนือ 4 คนนี้ OpenMCU ก็ยังคงจะต้องดีโคดแล้วทิ้งไป ซึ่งส่งผลเสียต่อซีพียูโหลด และอินคัมมิ่งแบนด์วิดท์ (Incoming Bandwidth) ของระบบเป็นอย่างสูง

จากการทำงานของ OpenMCU ที่กล่าวมาทั้งในส่วนออดิโอและวิดีโอจะเห็นว่ามีการทำงานที่ไม่ดีนักในแง่ของบิ๊กโอ (Big-Oh) คือเป็น $O(N^2)$ ซะเป็นส่วนใหญ่ ดังนั้นในบทต่อไปจะแนะการเพิ่มประสิทธิภาพในแง่ของการลดบิ๊กโอของ OpenMCU

2.5.2 รูปแบบการบีบอัดที่รองรับ

OpenMCU รองรับการบีบอัดข้อมูลออดิโอ, วิดีโอและตัวอักษรต่างๆดังนี้

1) การบีบอัดออดิโอ

- iLBC-15k2
- iLBC-13k3
- SpeexNarrow-8k
- SpeexNarrow-15k
- GSM-06.10
- MS-GSM
- G.711-uLaw-64k
- G.711-ALaw-64k
- LPC-10

2) การบีบอัดวิดีโอ

- H.261-QCIF
- H.263-QCIF

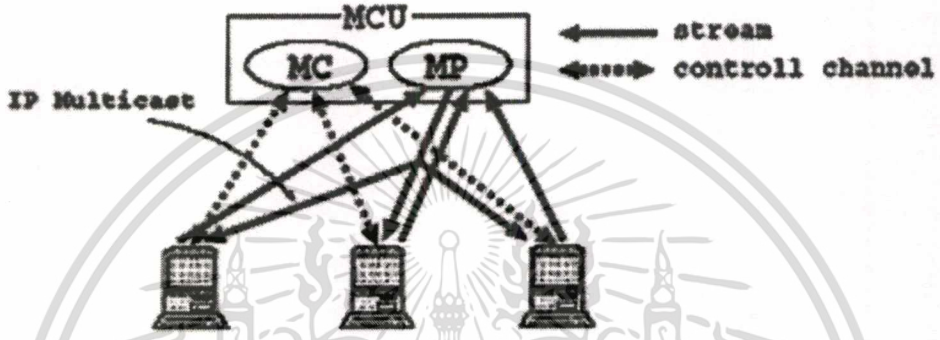
3) ตัวอักษร

- UserInput/basicString

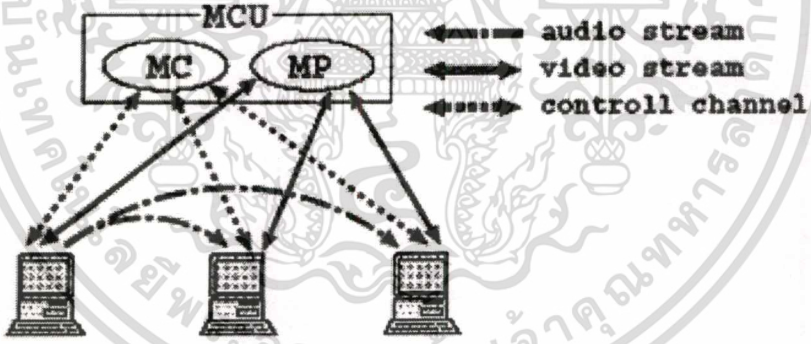
2.6 งานวิจัยที่เกี่ยวข้อง

[30] ได้อธิบายถึงความแตกต่างและข้อดีข้อเสียของ Centralized MCU และ Distributed MCU และได้เสนอ Distributed Multimedia Multiparty Teleconferencing (MMT) ที่พวกเขาได้สร้างขึ้น [31] ได้เสนอเทคนิคการผสมภาพวิดีโอใน Coded-Domain เป็น 6 ช่อง [32] ได้เสนอ Playout อัลกอริธึมเพื่อลดระยะเวลาในการทำ Jitter Buffer ของ OpenMCU [15] ได้วัดคอขวดของ OpenMCU และพบว่าที่จำนวนผู้ใช้มากๆ ซีพียูโหลดและเน็ทเวิร์คแบนวิดท์ของ OpenMCU มีผลต่อประสิทธิภาพของ OpenMCU และได้เสนอเทคนิคสำหรับแก้ปัญหาเหล่านี้ ดังนี้

- โดยการใช้ Multi-Cast แทน Uni-Cast ในการส่งวิดีโอและออดิโอสตรีมไปสู่เทอร์มินัล เพื่อที่จะลดซีพียูโหลดและเน็ทเวิร์คแบนวิดท์ของ OpenMCU เมื่อรองรับจำนวนผู้ใช้จำนวนมากๆ
- โดยการใช้เทอร์มินัลสื่อสารข้อมูลออดิโอกันเองโดยไม่ผ่าน MCU เพื่อที่จะลดซีพียูโหลดของ OpenMCU ในการประมวลผลสัญญาณออดิโอจากเทอร์มินัลและยังเป็นการลดเน็ทเวิร์คแบนวิดท์ที่เกิดจากออดิโอสตรีมที่ตัว OpenMCU อีกด้วย



รูปที่ 2.16 การใช้ Multi-Cast แทน Uni-Cast ในการส่งมัลติมีเดียสตรีมของ MCU



รูปที่ 2.17 การใช้เทอร์มินัลสื่อสารข้อมูลออดิโอโดยไม่ผ่าน MCU

บทที่ 3
การเพิ่มประสิทธิภาพของ OpenMCU และการสร้าง SIP MCU

3.1 กล่าวนำ

จากบทที่แล้วเราได้เรียนรู้เกี่ยวกับโปรโตคอลพื้นฐานของ VoIP ทั้ง 2 โปรโตคอล ได้แก่ H.323 และซิพ (SIP) อีกทั้งยังได้เรียนรู้เกี่ยวกับสถาปัตยกรรมพื้นฐานและการทำงานทั้งในส่วนออดิโอและวิดีโอของ OpenMCU ซึ่งเป็นซอฟต์แวร์ที่ใช้ในงานวิจัย ซึ่งจากการวิเคราะห์การทำงานของ OpenMCU พบว่าในส่วนออดิโอมีสเปซคอมเพล็กซิตี (Space Complexity) ของจำนวนออดิโอบัฟเฟอร์เป็น $O(N^2)$ ทำให้จำนวนการสำเนาและการผสมข้อมูลออดิโอมี Run-time คอมเพล็กซิตีเป็น $O(N^2)$ ด้วย ซึ่งจะเห็นว่า OpenMCU จะต้องทำงานหนักมากเมื่อมีผู้ใช้จำนวนมาก ในส่วนวิดีโอก็ยังมีงานที่ไม่จำเป็นต้องทำคือการดีโค้ดข้อมูลวิดีโอที่ไม่แสดงผลซึ่งจะเปลืองซีพียูโหลดและเน็ทเวิร์คแบนด์วิดท์ขาเข้า (Incoming Network Bandwidth) อย่างมากเนื่องจากวิดีโอเป็นข้อมูลขนาดใหญ่และใช้ความสามารถในการดีโค้ด (decode) ข้อมูลที่สูง

ในบทนี้เราจะแก้ไขปัญหาดังกล่าวข้างต้นโดยทำการเพิ่มประสิทธิภาพของ OpenMCU ให้มีประสิทธิภาพเพิ่มขึ้น โดยการลดบิ๊กโอ (Big-Oh) ซึ่งส่งผลให้ซีพียูโหลด และแบนด์วิดท์ลดลงตามไปด้วย ดังที่จะทดลองให้เห็นในบทถัดไป หลังจากที่ได้เพิ่มประสิทธิภาพของ OpenMCU แล้วเราจะนำโครงสร้างนี้มาพัฒนาต่อเป็น SIP MCU ซึ่งเป็น MCU ตามมาตรฐานโปรโตคอลซิพ ซึ่งเป็นโปรโตคอลที่ใหม่กว่าโปรโตคอล H.323 นอกจากนี้ยังได้วัดประสิทธิภาพของ OpenMCU ก่อนและหลังการเพิ่มประสิทธิภาพเปรียบเทียบกัน ดังที่จะได้ทดลองให้เห็นในบทถัดไป

3.2 การเพิ่มประสิทธิภาพของ OpenMCU

การเพิ่มประสิทธิภาพของ OpenMCU จะทำบนระบบปฏิบัติการไมโครซอฟต์วินโดส์ โดยแบ่งออกเป็น 2 ส่วนคือ ส่วนออดิโอและส่วนวิดีโอ ดังนี้

3.2.1 การเพิ่มประสิทธิภาพในส่วนออดิโอ

Block Diagram การทำงานของ OpenMCU ภายหลังการเพิ่มประสิทธิภาพในส่วนออดิโอแสดงดังรูปที่ 3.1 โดยมีรายละเอียดของการเพิ่มประสิทธิภาพดังนี้

1) การแก้ไขโครงสร้างข้อมูลของออดิโอบัฟเฟอร์

จากรูปที่ 3.1 เป็นโครงสร้างข้อมูลการเก็บข้อมูลออดิโอแบบใหม่ เราได้เพิ่มออดิโอบัฟเฟอร์เข้าไป 1 บัฟเฟอร์ในแต่ละคอนเน็คชัน โดยทุกครั้งที่มีออดิโอสตรีมเข้ามาก็จะทำการสำเนา

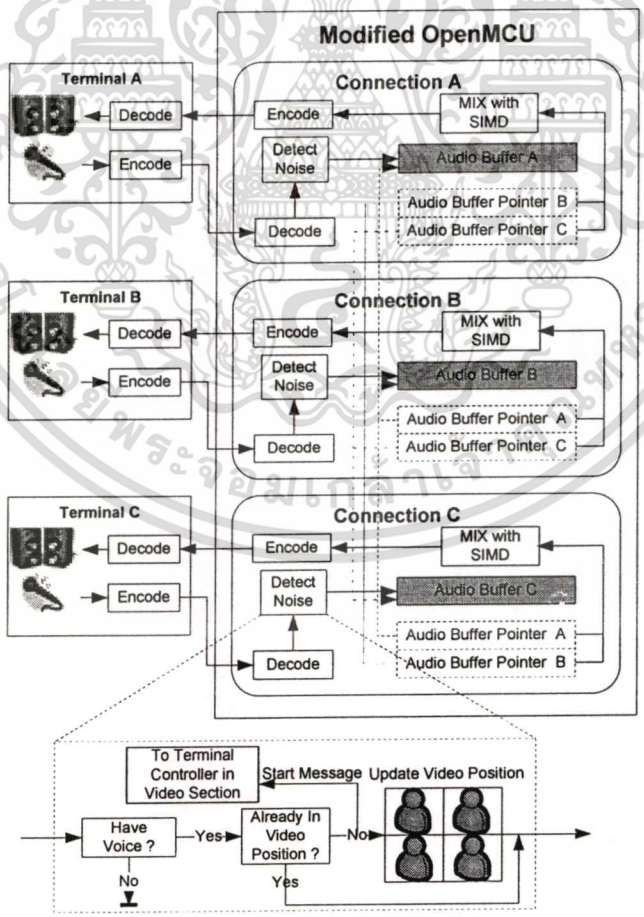
ข้อมูลออดิโอลงบัฟเฟอร์นี้ แทนที่จะสำเนาข้อมูลลงออดิโอบัฟเฟอร์ใน AudioBufferDict ของทุกๆ คอนเน็คชัน

เราได้เพิ่ม AudioBufferPointerDict เข้าไปในออบเจ็กต์คอนเน็คชันด้วย โดย Dict นี้จะเก็บออบเจ็กต์ออดิโอบัฟเฟอร์พอยน์เตอร์ซึ่งชี้ไปที่ออดิโอบัฟเฟอร์ที่อยู่ในแต่ละคอนเน็คชัน เพื่อใช้ประโยชน์ในการผสมออดิโอ โดยเมื่อออดิโอสตรีมเข้ามาก็จะทำการปรับปรุงขนาดของออดิโอบัฟเฟอร์ที่เข้ามาใหม่ในออบเจ็กต์ออดิโอบัฟเฟอร์พอยน์เตอร์ของทุกคอนเน็คชัน ดังนั้นขนาดของหน่วยความจำที่ลดลงสามารถคำนวณได้จากสมการ

$$mem = (N^2 - 2N) * BufferSize \tag{3.1}$$

โดย mem คือจำนวนหน่วยความจำที่ลดลง, N คือจำนวนผู้ใช้ และ BufferSize คือขนาดของออดิโอบัฟเฟอร์

จากโครงสร้างข้อมูลใหม่นี้พบว่ามีข้อดีคือลดจำนวนออดิโอบัฟเฟอร์จาก $O(N^2)$ เป็น $O(N)$ และลดจำนวนการสำเนาออดิโอบัฟเฟอร์ไปยังคอนเน็คชันต่างๆจาก $O(N^2)$ เป็น $O(N)$



รูปที่ 3.1 Block Diagram การทำงานของ OpenMCU ในส่วนออดิโอภายหลังการแก้ไข

2) การตรวจสอบความดังของออดิโอก่อนสำเนาลงบัฟเฟอร์

OpenMCU มีฟังก์ชันตรวจสอบความดังของข้อมูลออดิโออยู่แล้วคือฟังก์ชัน DetectNoise แต่ไม่ได้ใช้ประโยชน์อื่นจากฟังก์ชันนี้นอกจากใช้ปรับปรุงคนที่พูดล่าสุดที่จะแสดงผลในส่วนวิดีโอเท่านั้น เราจะนำฟังก์ชันนี้มาใช้ประโยชน์โดยตรวจสอบจากฟังก์ชัน DetectNoise ว่าผู้ใช้คนใดไม่ได้พูดจะไม่ทำการสำเนาข้อมูลออดิโอของผู้ใช้ท่านนี้ลงบัฟเฟอร์เลย ส่งผลให้เราไม่ต้องนำออดิโอบัฟเฟอร์ของผู้ใช้ท่านนี้ไปผสมเนื่องจากไม่มีข้อมูลที่จะทำการผสม ทำให้จำนวนการสำเนาและการผสมเมื่อนับทุกคอนเน็คชันในกรณี Best Case (ไม่มีผู้ใช้ท่านใดพูด) จะเป็น O(1) ทั้งคู่, กรณีพูด 1 คน (Average Case) จำนวนการสำเนาจะเป็น O(1) และการผสมจะเป็น O(N) และกรณีพูด k คนจำนวนการสำเนาจะเป็น O(k) และการผสมจะเป็น O(kN) เมื่อ N คือจำนวนผู้ใช้

3) การแก้ไขอัลกอริธึมการผสมออดิโอ

จากหัวข้อด้านบนเราได้แก้ไขบิ๊กโอของจำนวนออดิโอบัฟเฟอร์และจำนวนการสำเนาข้อมูลออดิโอแล้ว แต่บิ๊กโอของจำนวนการผสมออดิโอบัฟเฟอร์ยังคงเป็น $O(N^2)$ อยู่ดังนั้นเราจะแก้ไขอัลกอริธึมการผสมออดิโอในหัวข้อนี้

สมการการผสมข้อมูลออดิโอที่ได้จากการประชุมแสดงได้ดังสมการที่ 3.2

$$y(n) = \begin{cases} x1(n), & |x1(n)| > |x2(n)|, n = 1,2,3,...,N \\ x2(n), & otherwise, n = 1,2,3,...,N \end{cases} \quad (3.2)$$

โดยที่ n คือตำแหน่งของข้อมูล, y(n) คือข้อมูลออดิโอที่ผสมแล้ว, x1(n) คือข้อมูลออดิโอในบัฟเฟอร์แรก และ x2(n) คือข้อมูลออดิโอในบัฟเฟอร์ที่สอง การทำงานของอัลกอริธึมเดิมจะเป็นการเปรียบเทียบค่าสัมบูรณ์ของข้อมูล PCM ขนาด 16 bit ในทุกออดิโอบัฟเฟอร์ที่อยู่ใน AudioBufferDict ของคอนเน็คชันโดยกระทำทีละ 2 ออดิโอบัฟเฟอร์แล้วนำผลจากทั้ง 2 นี้ไปผสมกับบัฟเฟอร์ที่ 3, 4, 5 ... อัลกอริธึมการผสมออดิโอบัฟเฟอร์แบบเก่าแสดงดังรูปที่ 3.2 (อัลกอริธึมแบบใหม่ใช้ออดิโอบัฟเฟอร์ที่พอยน์เตอร์ชี้ไปแทนแบบเก่า) ภายในคอนเน็คชันหนึ่ง ถ้าข้อมูลมีขนาดเท่ากับ N คอมเพล็กซิตีก็จะเป็น O(N) จะเห็นว่าเราไม่มีวิธีที่จะเปลี่ยนคอมเพล็กซิตีของอัลกอริธึมนี้ได้เนื่องจากต้องใช้ข้อมูลทั้งหมดในการคำนวณ แต่เราสามารถประมวลผลข้อมูลแบบขนาน (SIMD: Single Instruction Multiple Data) กับข้อมูลนี้ด้วย SSE2 [22] ของ Pentium 4 โดยทำการประมวลผลแบบขนานกับข้อมูลขนาด 16 bit 8 ชุด จึงทำให้เกิด Speedup สูงสุด 8 เท่า อัลกอริธึมการผสมออดิโอบัฟเฟอร์แบบใหม่แสดงดังรูปที่ 3.3 รายละเอียดเกี่ยวกับ SSE 2 จะกล่าวถึงอย่างละเอียดในภาคผนวก ค.

```
Void Mix(char * dst, char * src, int count)
{
        #define mix_abs(x) ((x) >= 0 ? (x) : -(x))
        int i;
        for (i = 0; i < count; i += 2)
        {
                //Load src and dst
                int srcVal = *(short *)src;
                int dstVal = *(short *)dst;
                int newVal = dstVal;
                //compare absolute value of both
                if (mix_abs(newVal) > mix_abs(srcVal))
                        dstVal = newVal;
                else
                        dstVal = srcVal;
                //Store compared value
                *(short *)dst = (short)dstVal;
                //Go to next data
                dst += 2;
                src += 2;
}
```

รูปที่ 3.2 อัลกอริธึมการผสมออดิโอแบบเก่า

```
void Mix(char * dst, const char * src, int count)
{
        count = count >> 4;
        __m128i * dst128 = (__m128i *) dst;
        __m128i * src128 = (__m128i *) src;
        __m128i m0, m1, m2, m3, m4;
        while (count--)
```

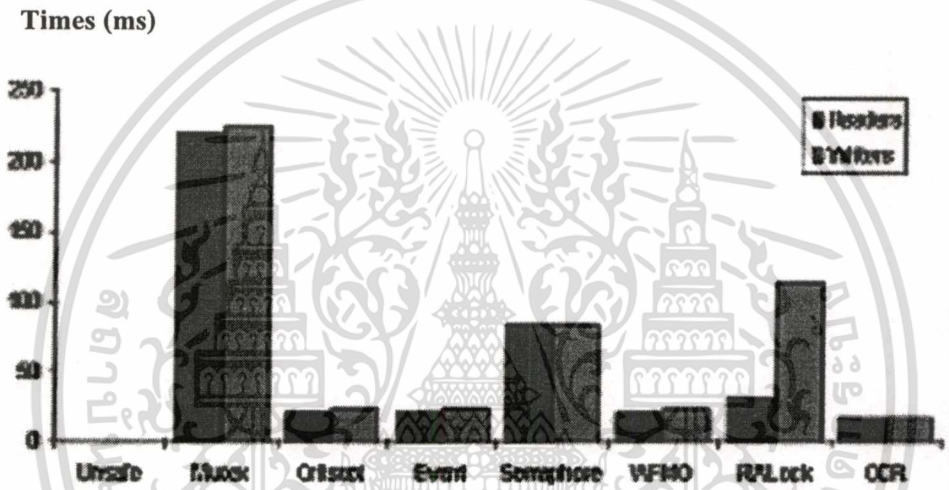
```
{
    //Load dst audio data
    m0 = _mm_loadu_si128 (dst128);
    //Absolute dst
    m2 = _mm_srai_epi16 (m0, 15);
    m1 = _mm_xor_si128 (m0, m2);
    m1 = _mm_subs_epi16 (m1, m2);
    //Load src audio data
    m2 = _mm_loadu_si128 (src128);
    //Absolute src
    m3 = _mm_srai_epi16 (m2, 15);
    m4 = _mm_xor_si128 (m2, m3);
    m4 = _mm_subs_epi16 (m4, m3);
    //Compare dst and src
    m1 = _mm_cmpgt_epi16 (m1, m4);
    m0 = _mm_and_si128 (m0, m1);
    m1 = _mm_andnot_si128 (m1, m2);
    m0 = _mm_or_si128 (m0, m1);
    //Store result in dst
    _mm_storeu_si128 (dst128, m0);
    //Go to next data
    dst128 ++;
    src128 ++;
  }
}
```

รูปที่ 3.3 อัลกอริธึมการผสมออดิโอแบบใหม่โดยใช้ SSE2

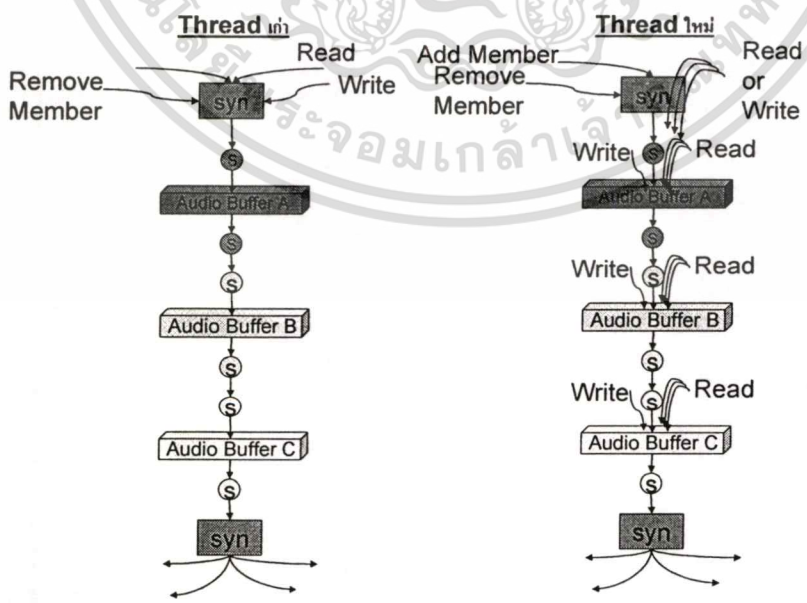
4) การแก้ไขเธรดซิงโครไนซ์เซชันออบเจ็กต์ (Thread Synchronization Object)

จากรูปที่ 3.4 [33] เป็นข้อมูลจากไมโครซอฟต์แสดงประสิทธิภาพของเธรดซิงโครไนซ์เซชันออบเจ็กต์ต่างๆเมื่อสร้างเป็น รีดเดอร์ไรท์เตอร์(Readers-Writers) เธรดซิงโครไนซ์เซชันออบเจ็กต์ บนระบบปฏิบัติการไมโครซอฟต์วินโดส์ จากกราฟจะพบว่า Mutex เมื่ออิมพลีเมนท์เป็น

รีดเดอร์ไรท์เตอร์เธรดซิงโครไนซ์เซชันออบเจ็กต์แล้ว ใช้เวลาของซีพียูในการล็อคและปลดล็อคแต่ละทีมากกว่าออบเจ็กต์ Critical Section และ Events เกือบ 10 เท่าทีเดียว ซึ่ง OpenMCU มีการเรียกใช้ Mutex นี้ในการทำเธรดซิงโครไนซ์เซชัน ซึ่งข้อดีของ Mutex ก็คือสามารถซิงโครไนซ์ได้ระหว่างโพรเซสส่วน Critical Section มีการทำงานที่คล้ายกันแต่สามารถซิงโครไนซ์ระหว่างเธรดที่อยู่ในโพรเซสเดียวกันได้เท่านั้นไม่สามารถซิงโครไนซ์ข้ามโพรเซสได้ แต่ OpenMCU ไม่ต้องการการซิงโครไนซ์ระหว่างโพรเซสทำให้การใช้ Mutex ดูจะกินโอเวอร์เฮดมากไปสักหน่อย ดังนั้นเราจึงทำการเปลี่ยนออบเจ็กต์ซิงโครไนซ์เซชันใหม่โดยเปลี่ยนจาก Mutex มาเป็น Critical Section และ Events รายละเอียดเกี่ยวกับการใช้ออบเจ็กต์ซิงโครไนซ์เซชันจะกล่าวถึงอย่างละเอียดในภาคผนวก ข.



รูปที่ 3.4 กราฟความสัมพันธ์ระหว่างเวลาในการล็อคกับออบเจ็กต์เธรดซิงโครไนซ์เซชันต่างๆ



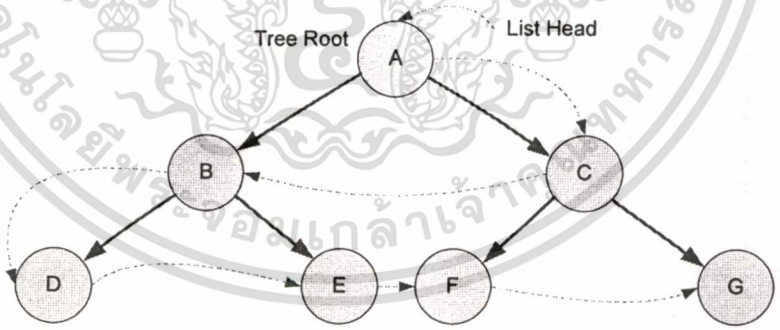
รูปที่ 3.5 Thread Synchronization ที่ Audio Buffer

5) การใช้รีดเดอร์ไรท์เตอร์เธรดซิงโครไนซ์เซชัน (Readers-Writers Thread Synchronization) ที่ออดิโอบัฟเฟอร์

เนื่องจากอัลกอริธึมแบบเก่าจะปล่อยเทรดให้ทำการเขียนหรืออ่านออดิโอบัฟเฟอร์ในเวลาใดๆเพียงเทรดเดียวเท่านั้นแม้ว่าจะไม่ได้ทำการอ่านหรือเขียนออดิโอบัฟเฟอร์เดียวกัน ซึ่งในระบบปฏิบัติการวินโดส์ เราสามารถให้เธรดต่างๆทำงานพร้อมๆกันได้เพื่อประหยัดเวลาในการประมวลผล เราสามารถแก้ไขได้โดยปล่อยให้มีการเขียนออดิโอบัฟเฟอร์ที่ไม่ใช่บัฟเฟอร์เดียวกันพร้อมกันและอ่านออดิโอบัฟเฟอร์ได้อย่างอิสระแม้จะเป็นออดิโอบัฟเฟอร์เดียวกันโดยใช้รีดเดอร์ไรท์เตอร์เธรดซิงโครไนซ์เซชัน ดังรูปที่ 3.5 ส่วนรายละเอียดการสร้างออบเจ็กต์รีดเดอร์ไรท์เตอร์เธรดซิงโครไนซ์เซชัน จะกล่าวถึงอย่างละเอียดในภาคผนวก ข.

6) การใช้บาลานซ์ไบนารีเซิร์ชทรี (Balanced Binary Search Tree) และลิ้งค์ลิสต์ (Linked List) แทนดิคท์ (Dict)

จากการดีบักโปรแกรมพบว่าเกิดข้อผิดพลาดขึ้นหลังจากการใช้รีดเดอร์ไรท์เตอร์เธรดซิงโครไนซ์เซชัน โดยจากการตรวจสอบพบว่าข้อผิดพลาดที่เกิดขึ้นมาจากการที่ดิคท์ซึ่งเป็นโครงสร้างที่ใช้ในการเก็บข้อมูลไม่ใช่ Thread Safe คือไม่สามารถเข้าถึงได้จากหลายเธรด และยังมีคุณสมบัติบางประการที่ไม่เหมาะสมเช่นเมื่อต้องการออบเจ็กต์ทั้งหมดในดิคท์ จะต้องทำการค้นหาทีละออบเจ็กต์จนกว่าจะหมด ทำให้คอมเพล็กซิตีเป็น O(N) เมื่อ N เป็นจำนวนออบเจ็กต์ เราจึงแก้ปัญหานี้โดยออกแบบการเก็บข้อมูลใน OpenMCU ใหม่โดยใช้บาลานซ์ไบนารีเซิร์ชทรี โดยที่โหนด (Node) ทั้งหมดในทรีต่อกันเป็นลิ้งค์ลิสต์ ดังรูปที่ 3.6



รูปที่ 3.6 Balanced Binary Search Tree และ Linked List

สำหรับการค้นหาในโครงสร้างการเก็บข้อมูลใหม่นี้ในกรณี Worse Case จะเป็น Log_2N และเมื่อต้องการออบเจ็กต์ทั้งหมดเราก็จะดึงเอาส่วนหัวของลิ้งค์ลิสต์มาได้เลย เนื่องจากแต่ละโหนดเชื่อมต่อกันเป็นลิ้งค์ลิสต์อยู่แล้ว ไม่ต้องค้นหาทีละออบเจ็กต์ทำให้คอมเพล็กซิตีลดลงจาก O(N) เป็น O(1) โดยที่ N เป็นจำนวนออบเจ็กต์

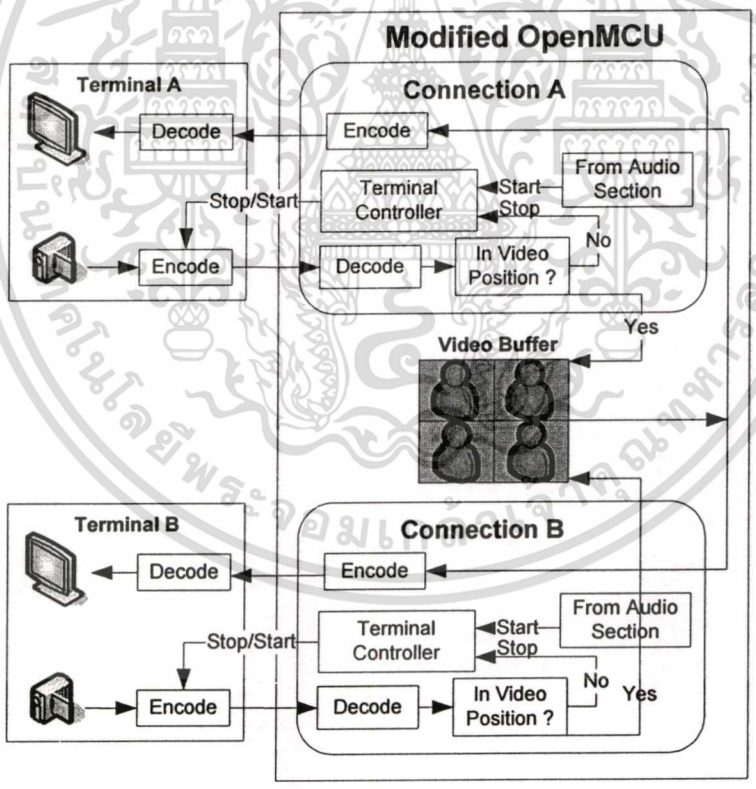
3.2.2 การเพิ่มประสิทธิภาพในส่วนวิดีโอ

Block Diagram การทำงานของ OpenMCU ภายหลังการเพิ่มประสิทธิภาพในส่วนวิดีโอ แสดงดังรูปที่ 3.7 โดยมีรายละเอียดการเพิ่มประสิทธิภาพดังนี้

1) การใช้การส่งแมสเสจเพื่อควบคุมเทอร์มินัล

จากรูปที่ 3.7 OpenMCU ที่ปรับปรุงแล้วนี้จะมีฟังก์ชันที่ทำหน้าที่พิเศษคือ Terminal Controller โดยมีหน้าที่ส่งแมสเสจไปให้เทอร์มินัล โดยถ้าเทอร์มินัลที่กำลังส่งวิดีโอสตรีมไม่ได้อยู่ใน 4 ตำแหน่งที่จะแสดงผล Terminal Controller ก็จะส่งแมสเสจ "STOP" ไปให้เทอร์มินัลนั้น เพื่อให้เทอร์มินัลนั้นหยุดการส่งวิดีโอสตรีมส่งผลให้ซีพียูโหลดที่ MCU ลดลงเนื่องจากบิ๊กโอของการดีโค้ดข้อมูลวิดีโอลดลงจาก O(N) เป็น O(1) เมื่อ N เป็นจำนวนเทอร์มินัลที่เชื่อมต่อกับ MCU นอกจากนี้ยังทำให้เน็ทเวิร์คทราฟฟิกขาเข้า (Incoming Network Traffic) ลดลงด้วยเนื่องจากไม่มีการส่งวิดีโอสตรีมจากเทอร์มินัลมา MCU นอกเหนือจากเทอร์มินัลที่อยู่ใน 4 ตำแหน่งที่จะแสดงผล

เมื่อมีเทอร์มินัลใดๆเข้ามาในตำแหน่ง 4 ตำแหน่งที่จะแสดงผลมันจะส่งแมสเสจ "Start" ไปให้เทอร์มินัลนั้นส่งผลให้เทอร์มินัลนั้นเริ่มการส่งวิดีโอสตรีมมาที่ MCU



รูปที่ 3.7 Block Diagram การทำงานของ OpenMCU ในส่วนออดิโอภายหลังการแก้ไข

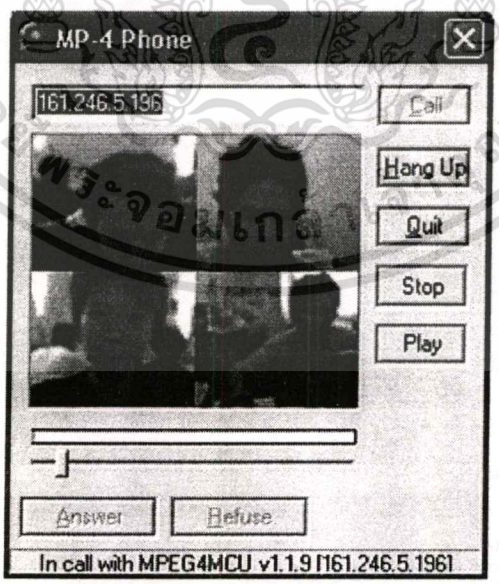
2) การใช้รีดเดอร์ไรท์เตอร์เธรดซิงโครไนซ์เซชัน (Readers-Writers Thread Synchronization) ที่วิดีโอบัฟเฟอร์

ในทำนองเดียวกันกับในส่วนออดิโอ เราได้เพิ่มสภาวะการทำงานพร้อมกันของเธรดต่างๆ ในการเข้าถึงวิดีโอบัฟเฟอร์ โดยใช้รีดเดอร์ไรท์เตอร์เธรดซิงโครไนซ์เซชัน โดยแต่ละเธรดสามารถเขียนลงวิดีโอบัฟเฟอร์เดียวกันได้ทีละเธรด และอ่านวิดีโอบัฟเฟอร์เดียวกันได้ทีละหลายเธรด

3) การรองรับวิดีโอโคเดคตามมาตรฐาน MPEG-4

MPEG-4 [16, 17] เป็นมาตรฐานการบีบอัดข้อมูลวิดีโอที่พัฒนาขึ้นมาเพื่อใช้ในการส่งข้อมูลวิดีโอผ่านโครงข่ายอินเตอร์เน็ต และเครือข่ายไร้สาย (wireless network) ต่างๆ โดยมีจุดเด่นในด้านการบีบอัดข้อมูลที่สามารถลดขนาดไฟล์ และบิทเรท (bit rate) ลงได้ต่ำมากโดยที่คุณภาพของข้อมูลยังอยู่ในเกณฑ์ที่ดี นอกจากนี้ MPEG-4 ยังมีคุณสมบัติที่ทนทานต่อการเกิดความผิดพลาดในช่องสัญญาณ (Error-resilience) และสามารถกำหนดคุณภาพของการบีบอัดข้อมูล (กำหนด profile และ scalable parameter ต่างๆ) ได้หลากหลายกว่า MPEG-2 เทคนิคการบีบอัดข้อมูลที่มีเพิ่มขึ้นจากที่ใช้ในมาตรฐาน MPEG-2, H.261 และ H.263 ได้แก่ความสามารถในการบีบอัดข้อมูลโดยแยกเป็นชิ้นส่วนวัตถุได้ (Object-based coding)

ใน OpenMCU ที่พัฒนาขึ้นใหม่นี้จะรองรับมาตรฐาน MPEG-4 นอกเหนือจาก H.261 และ H.263 ที่รองรับโดย OpenMCU ตัวเก่า โดยโคเดค MPEG-4 ที่ใช้ในงานวิจัยนี้มาจากไลบรารี libavcodec ของ FFmpeg [34] โดย FFmpeg ที่ใช้คือเวอร์ชัน 0.4.8 รายละเอียดเกี่ยวกับ FFmpeg จะถูกกล่าวถึงอย่างละเอียดในภาคผนวก ก.



รูปที่ 3.8 H.323 Terminal ซึ่งมี MPEG-4 Codec ที่สร้างขึ้น

3.2.3 การสร้าง H.323 เทอร์มินัล

เราได้พัฒนา H.323 เทอร์มินัลขึ้นบนระบบปฏิบัติการวินโดส์ เพื่อรองรับคุณสมบัติใหม่ที่มีอยู่ใน OpenMCU หลังจากการเพิ่มประสิทธิภาพ ดังรูปที่ 3.8 ซึ่งเทอร์มินัลนี้สามารถสื่อสารได้ทั้งวิดีโอและออดิโอซึ่งเป็นตรงตามมาตรฐานของโปรโตคอล H.323 ทุกอย่าง โดย H.323 เทอร์มินัลที่พัฒนาขึ้นมีคุณสมบัติดังนี้

- สามารถรองรับการส่งแมสเสจจาก Terminal Controller ของตัว OpenMCU
- รองรับมาตรฐาน MPEG-4
- สามารถเชื่อมต่อกันเองแบบจุดต่อจุด (Point-to-Point) โดยไม่ผ่าน MCU

3.3 การสร้าง SIP MCU

หลังจากที่เราได้ทำการเพิ่มประสิทธิภาพของ OpenMCU แล้ว เราจะนำสถาปัตยกรรมและการทำงานของ OpenMCU มาสร้าง SIP MCU [18, 19, 20, 21] ซึ่งเป็น MCU ตามมาตรฐานโปรโตคอลซิพ โดยสร้างจากไลบรารี oSIP [35] เวอร์ชัน 2.2.1-pre3 และ eXosip [36] เวอร์ชัน 0.9.0-pre1 ซึ่งรายละเอียดของไลบรารีเหล่านี้จะกล่าวถึงอย่างละเอียดในภาคผนวก ก.

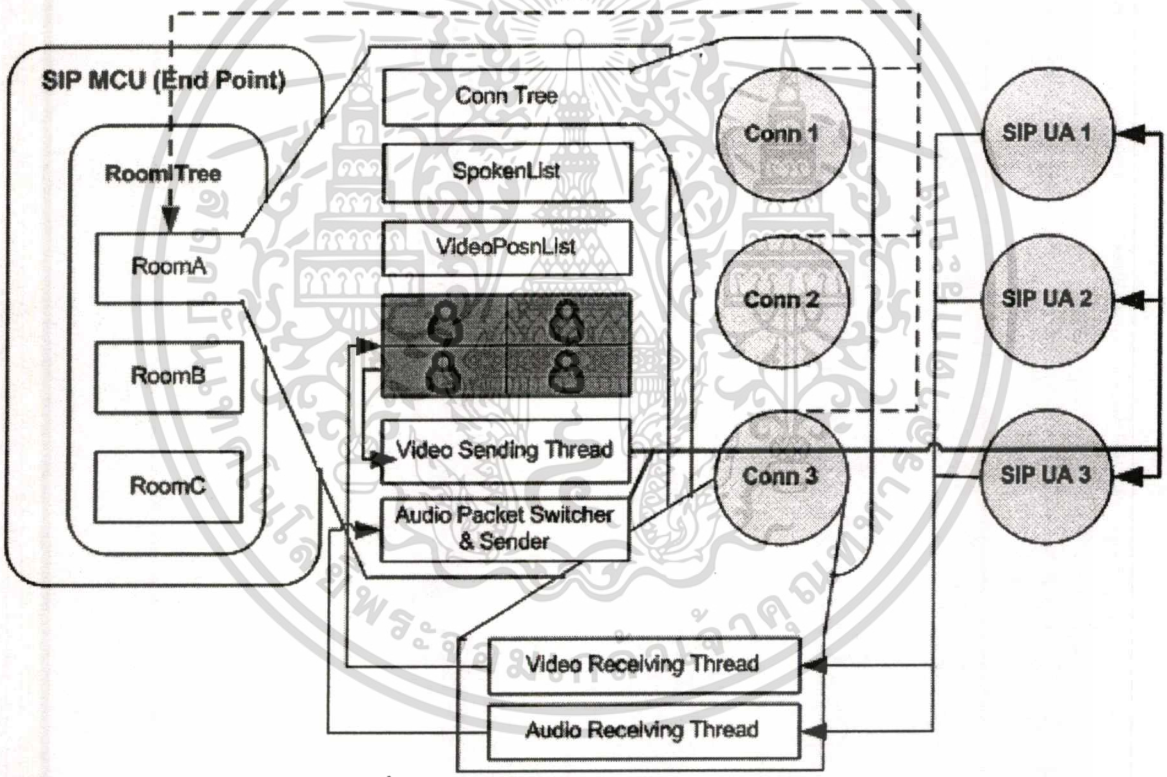
3.3.1 สถาปัตยกรรมของ SIP MCU

สถาปัตยกรรมของ SIP MCU ดังรูปที่ 3.9 มีความแตกต่างจากสถาปัตยกรรมของ OpenMCU มาก โดยมีออบเจ็กต์เอนด์พอยน์ (Endpoint Object) ทำหน้าที่เป็น SIP UAS ของ SIP MCU โดยออบเจ็กต์เอนด์พอยน์จะประกอบด้วย Room Tree ซึ่งเป็นทรีที่ใช้เก็บ Room Object แตกต่างกับ OpenMCU ซึ่งเก็บห้องเป็นสตริง สาเหตุของการเก็บห้องเป็นออบเจ็กต์ก็เนื่องมาจากต้องการที่จะแยกการจัดการมีเดียในแต่ละห้องออกจากกันเพื่อเป็นการลดการทำเธรดซิงโครไนซ์นั่นเอง ภายใน Room Object ประกอบด้วย Conn Tree ซึ่งทำหน้าที่เก็บคอนเน็คชันต่างๆซึ่งจะสร้างขึ้นทุกครั้งที่มี SIP Terminal (SIP UA) เชื่อมต่อเข้ามาใหม่ โดยออบเจ็กต์คอนเน็คชันนี้ทำหน้าที่ต่างกับของ OpenMCU คือจะทำหน้าที่เก็บข้อมูลต่างๆเกี่ยวกับแต่ละคอนเน็คชันเท่านั้น ในส่วนSpokenList, VideoPosnList และ Video Buffer จะเหมือนกับของ OpenMCU ต่างกันแต่เพียงว่าไม่ได้เก็บ List ใน Dict แต่เก็บในห้องแทนเพื่อลดการทำเธรดซิงโครไนซ์เมื่อต้องการเข้าถึง List ในแต่ละห้องดังที่ได้กล่าวถึงก่อนหน้านี้ ภายใน Room Object จะมีเธรดอยู่เพียง 3 เธรดแตกต่างจาก OpenMCU ที่แต่ละคอนเน็คชันจะมี 3 เธรดซึ่งเป็นการลด CPU Load เนื่องจากจำนวนเธรดมากทำให้ CPU Load เพิ่มขึ้นตามไปด้วย โดยเธรดเหล่านี้ คือ

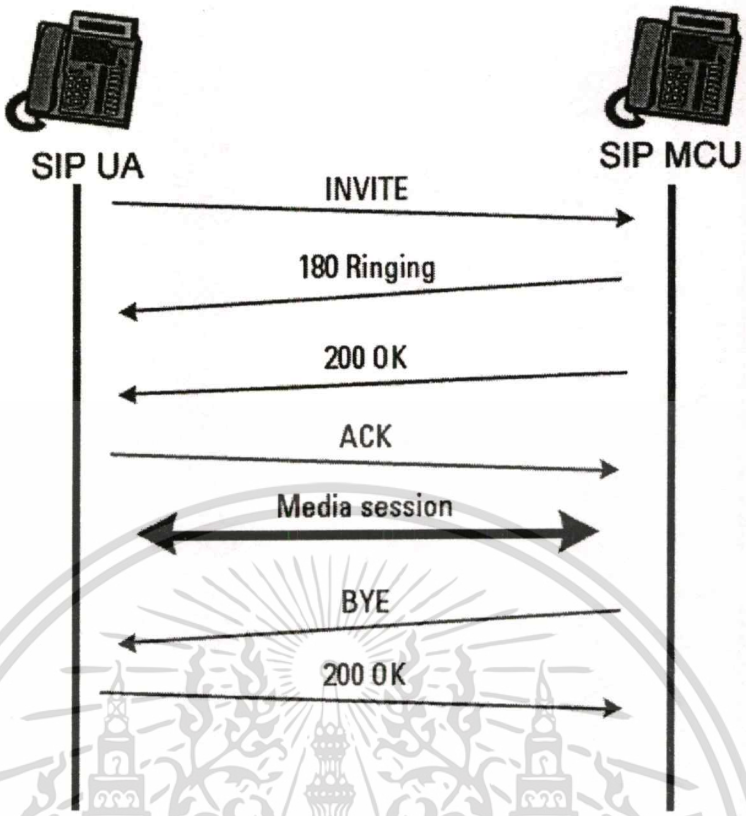
- Video Receiving Thread ถูกสร้างขึ้นสำหรับการรับวิดีโอจาก SIP UA ต่างๆแล้วนำไปเก็บยัง Video Buffer โดยการรับจะใช้ Socket Server คือเปิดพอร์ต (Port) เดียวแต่สามารถรับ

ข้อมูลจาก SIP UA ได้ทั้งหมด เนื่องจากมี Terminal Controller ดังนั้นเราจะรับข้อมูลวิดิโออย่างมากที่สุดจากผู้ใช้เพียง 4 คนเท่านั้น ดังนั้น 1 เธรดและ Socket 1 พอร์ตก็เพียงพอแล้ว

- Video Sending Thread จะนำข้อมูลจาก Video Buffer ส่งไปให้ SIP UA ทุกตัว โดยการเอนโคดวิดีโอจะทำเพียงครั้งเดียวเท่านั้นและส่งไปให้แต่ละ SIP UA ซึ่งต่างจาก OpenMCU ที่จะต้องทำการเอนโคด 1 ครั้งต่อ 1 คอนเน็คชัน
- Audio Receiving Thread จะนำข้อมูลออดิโอที่ได้รับจาก SIP UA แต่ละตัวมาทำการเลือกว่าช่วงนี้ใครได้ครองช่องสัญญาณ ก็จะส่งเสียงของคนนั้นไปที่ทุกๆ SIP UA ส่วนเสียงของคนอื่นจะทิ้งไป โดยจะไม่ใช้การผสมเสียง ซึ่งอยู่บนสมมุติฐานที่ว่าที่เวลาใดๆมีผู้พูด 1 คน ทำให้ไม่ต้องมีการดีโคดและเอนโคดออดิโอเลย ซึ่งอยู่บนข้อจำกัดที่ว่า SIP UA ต่างๆอยู่บนเน็ตเวิร์คที่ดีและมีออดิโอโคเดคตัวเดียวกัน



รูปที่ 3.9 สถาปัตยกรรมของ SIP MCU



รูปที่ 3.10 การสร้าง SIP Session ระหว่าง SIP UA และ SIP MCU

3.3.2 การสร้าง Session ระหว่าง SIP MCU และ SIP UA

การสร้าง Session ระหว่าง SIP UA และ SIP MCU เริ่มจาก SIP UA ส่งข้อความร้องขอ (request message) ไปให้กับ SIP MCU เพื่อทำการประมวลผลแล้วจึงส่งข้อความตอบกลับ (response message) กลับมายัง SIP UA ดังรูปที่ 3.10

จากรูปที่ 3.10 เป็นตัวอย่างการสร้าง SIP Session โดยเริ่มจาก SIP UA ส่ง INVITE Message ไปที่ SIP MCU ซึ่ง INVITE Message นี้ประกอบด้วยเนื้อหาของ SDP ซึ่งใช้อธิบายมีเดียที่จะใช้แลกเปลี่ยนกันดังนี้

```
1   v=0
2   o=- 0 0 IN IP4 127.0.0.1
3   s=-
4   c=IN IP4 127.0.0.1
5   b=AS 10
6   t=0 0
7   m=audio 1000 UDP 97
```

```
8   a=rtpmap 97 speex
9   m=video 1001 UDP 98
10  a=rtpmap 98 MP4V/10
```

บรรทัดที่ 2, 3, 6 ไม่ได้ใช้ในโปรโตคอลซิพ แต่มีไว้เพื่อให้เข้ากันได้กับมาตรฐาน SDP บรรทัดที่ 5 บอกแบนด์วิดท์ของตัว SIP UA ในที่นี้คือ 10 kbits/sec บรรทัดที่ 7 บอกว่ามีเดียที่จะแลกเปลี่ยนคือออดิโอที่พอร์ต 1000 (โดยตัว SIP UA จะสุ่ม UDP พอร์ตมา 1 พอร์ตสำหรับรับส่งข้อมูลออดิโอ) UDP บอกว่าใช้ UDP ในการรับส่ง สาเหตุที่ต้องใช้ UDP เนื่องจาก SIP MCU ตัวนี้มีแผนที่จะให้รองรับ Mobile Phone ในอนาคต ซึ่ง Mobile Phone ไม่มี RTP Stack โดยหมายเลขของโปรไฟล์ของการบีบอัดออดิโอคือ 97 ในที่นี้คือ speex [37] ซึ่งจริงๆแล้วเป็น RTP โปรไฟล์เหตุที่ใช้ RTP โปรไฟล์เนื่องจาก UDP ไม่มีโปรไฟล์จึงไม่รู้จะใช้อะไรในการตรวจสอบมีเดียที่จะใช้ทำการแลกเปลี่ยนจึงใช้ RTP โปรไฟล์แทน บรรทัดที่ 9 บอกว่าเป็นวิดีโอที่พอร์ตที่มากกว่าออดิโออยู่หนึ่งและมีเดียที่ใช้เป็น MPEG-4 Visual

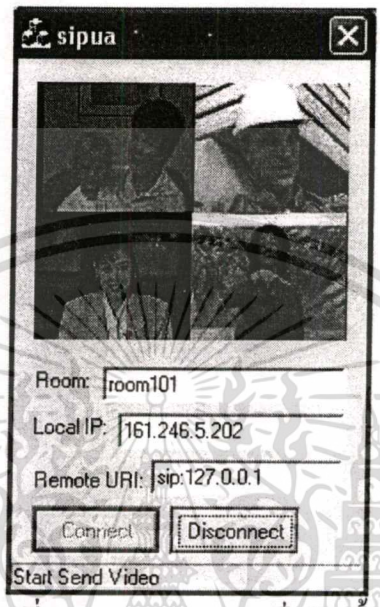
หลังจากนั้นเมื่อ SIP MCU ได้รับ INVITE Message แล้วก็จะส่ง SIP Response Message คือ 180 Ringing มีความหมายเหมือนสัญญาณกริ่งโทรศัพท์ดังในโทรศัพท์ปกติ

หลังจากนั้น SIP MCU จะตรวจสอบว่ามีเดียใน INVITE Message กับของที่ตนมีตรงกันหรือไม่แล้วตอบกลับเพื่อยืนยันการสร้าง Session ด้วย 200 OK response ซึ่งมีเนื้อหาของ SDP ดังนี้

```
1.  v=0
2.  o=- 0 0 IN IP4 127.0.0.1
3.  s=-
4.  c=IN IP4 127.0.0.1
5.  b=AS 10
6.  t=0 0
7.  m=audio 2000 UDP 97
8.  a=rtpmap 97 speex
9.  m=video 2001 UDP 98
10. a=rtpmap 98 MP4V/10
```

โดยที่ SIP MCU จะยืนยันรูปแบบของมีเดียที่ตรงกันที่พร้อมจะทำการแลกเปลี่ยน โดย SIP MCU จะสุ่มพอร์ต UDP มา 2 พอร์ตเพื่อทำการแลกเปลี่ยน

หลังจากนั้นเมื่อ SIP UA ได้รับ 200 OK Response แล้วก็จะตอบกลับด้วย ACK การแลกเปลี่ยนมีเดียจึงเริ่มขึ้น และเมื่อ SIP UA ต้องการยุติการแลกเปลี่ยนมีเดียก็จะส่ง BYE Message ไปให้ SIP MCU ซึ่ง SIP MCU จะตอบยืนยัน BYE Message ด้วย 200 OK Response SIP Session นี้เป็นอันยุติ



รูปที่ 3.11 SIP Terminal ที่สร้างขึ้น

3.3.3 การสร้าง SIP Terminal

เราได้พัฒนา SIP Terminal ขึ้นบนระบบปฏิบัติการวินโดส์ เพื่อรองรับ SIP MCU ที่ได้สร้างขึ้น ดังรูปที่ 3.11 โดย SIP Terminal ที่พัฒนาขึ้นมีคุณสมบัติดังนี้

- สามารถรองรับการส่งแมสเสจจาก Terminal Controller ของ SIP MCU
- รองรับมาตรฐาน MPEG-4 (FFMPEG 4.8 [34])
- ใช้ออดิโอโคเดค Speex [37]

3.4 สรุปการเพิ่มประสิทธิภาพ

สรุปการเพิ่มประสิทธิภาพเป็นดังตารางที่ 1 โดยที่ N = จำนวนคอนเน็คชัน (Terminal) และ Big-Oh ที่ได้เป็น Big-Oh ในหนึ่งห้อง และใน Average Case

ตารางที่ 3.1 สรุปการเพิ่มประสิทธิภาพ

Algorithms	Original OpenMCU	Enhanced OpenMCU	SIP MCU
จำนวนออดิโอบัฟเฟอร์	$O(N^2)$	O(N)	O(1)
จำนวนการสำเนาข้อมูลออดิโอลงบัฟเฟอร์	$O(N^2)$	O(1)	O(1)
จำนวนการผสม Audio	$O(N^2)$	O(N/8)	O(1)
จำนวนการ Encode Video	O(N)	O(N)	O(1)
จำนวนการ Decode Video	O(N)	O(1)	O(1)
จำนวนการ Encode Audio	O(N)	O(N)	O(1)
จำนวนการ Decode Audio	O(1)	O(1)	O(1)
จำนวนการค้นหาออบเจ็กต์ทั้งหมดในดิคท์	O(N)	O(1)	O(1)
จำนวนการค้นหา SpokenList	O(1)	O(1)	O(1)
จำนวนการค้นหา VideoPosnList	O(N)	O(1)	O(1)
จำนวนการค้นหา Video Buffer	O(N)	O(N)	O(1)
Room	String	String	Object
Thread Synchronization Object	Mutex	Critical Section & Event	Critical Section & Event
จำนวน Thread	O(N)	O(N)	O(1)
Thread Access	One Thread Access	Readers-Writers Thread Access	Readers-Writers Thread Access

บทที่ 4

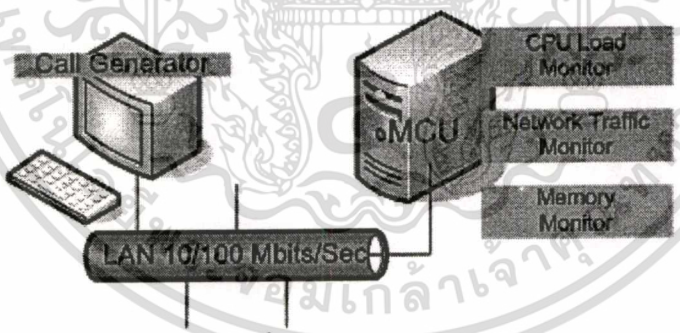
การเปรียบเทียบและวิเคราะห์ประสิทธิภาพของ OpenMCU และ SIP MCU

4.1 กล่าวนำ

ในบทนี้จะเป็นการเปรียบเทียบและวิเคราะห์ผลการเพิ่มประสิทธิภาพของ OpenMCU และ SIP MCU ในแง่ของซีพียูโหลด (CPU Load) และเน็ทเวิร์คแบนด์วิดท์ (Network Bandwidth) ต่อจำนวนผู้ใช้ ซึ่งได้ถูกวิเคราะห์ใน [15] แล้วว่าเป็นคอขวดของ OpenMCU ซึ่งส่งผลให้ OpenMCU รองรับผู้ใช้ได้น้อย นอกจากนี้เรายังวัดผลของจำนวนหน่วยความจำที่ใช้ซึ่งส่งผลต่อขนาดของการอ่าน I/O (Input and Output) ของซีพียูอีกด้วย

4.2 วิธีการทดลอง

ในการทดลองนี้ เราทำการทดลองแสดงได้ดังรูปที่ 4.1 ซึ่งการทดลองนี้จัดทำขึ้นโดยใช้วิธียิงคอนเน็คชันจาก Call Generator แล้ววัดซีพียูโหลด, หน่วยความจำ และเน็ตเวิร์คทราฟฟิกที่จำนวนผู้ใช้ต่างๆเปรียบเทียบ OpenMCU ก่อนและหลังการเพิ่มประสิทธิภาพ และ SIP MCU



รูปที่ 4.1 การทดลอง

โดยสิ่งแวดล้อมที่ใช้ในการทดลองเป็นดังนี้

- OpenMCU และ SIP MCU รันบนคอมพิวเตอร์ CPU Pentium 4 HyperThread ที่คล็อก 2.8 GHz, Ram 512 MBytes
- ระบบปฏิบัติการ Windows XP Service Pack 2
- Lan 10/100 Mbits/sec

- Original OpenMCU
  - Video Codec = H.263 QCIF (FFMPEG 4.7 with patch for netmeeting compatible) ที่ Frame rate=10 frames/sec, Bit rate 50 kbits/sec, Max key frame interval = 10 frames
  - Audio Codec = G.711-uLaw ที่ 64 kbits/sec
- Enhanced OpenMCU
  - Video Codec = MPEG-4 QCIF (FFMPEG 4.8) ที่ Frame rate=10 frames/sec, Bit rate 50 kbits/sec, Max key frame interval = 10 frames Audio Codec = G.711-uLaw ที่ 64 kbits/sec
- SIP MCU
  - Video Codec = MPEG-4 QCIF (FFMPEG 4.8)) ที่ Frame rate=10 frames/sec, Bit rate 50 kbits/sec, Max key frame interval = 10 frames
  - Audio Codec = G.711-uLaw ที่ 64 kbits/sec
- ในการเปรียบเทียบจำนวนผู้ใช้และจำนวนห้องให้วัดซีพียูโหลดที่ 60 % เนื่องจากที่ซีพียูโหลดมากกว่า 60 % คุณภาพเสียงจะไม่ดี ซึ่งเป็นสภาพแวดล้อมที่ไม่ดีนักในการทำการประชุม

ในการทดลอง เราจำเป็นจะต้องมีเครื่องมือ 4 อย่าง คือ H.323 Call Generator, SIP Call Generator, เครื่องมือวัดซีพียูโหลดและเน็ตเวิร์คทราฟฟิก (Network Traffic), และเครื่องมือวัดจำนวนหน่วยความจำ โดยรายละเอียดของเครื่องมือทั้ง 4 เป็นดังนี้

1) H.323 Call Generator

เราได้สร้าง H.323 call generator ขึ้นโดยดัดแปลงจาก Callgen323 [38] เวอร์ชัน 1.4 ซึ่งปกติแล้วสามารถสื่อสารออดิโอได้เพียงอย่างเดียวและมีห้องเดียว ให้สามารถสื่อสารวิดีโอและให้มีหลายห้องได้ Call Generator นี้จะทำหน้าที่เป็นเทอร์มินัลจำลองให้กับ OpenMCU

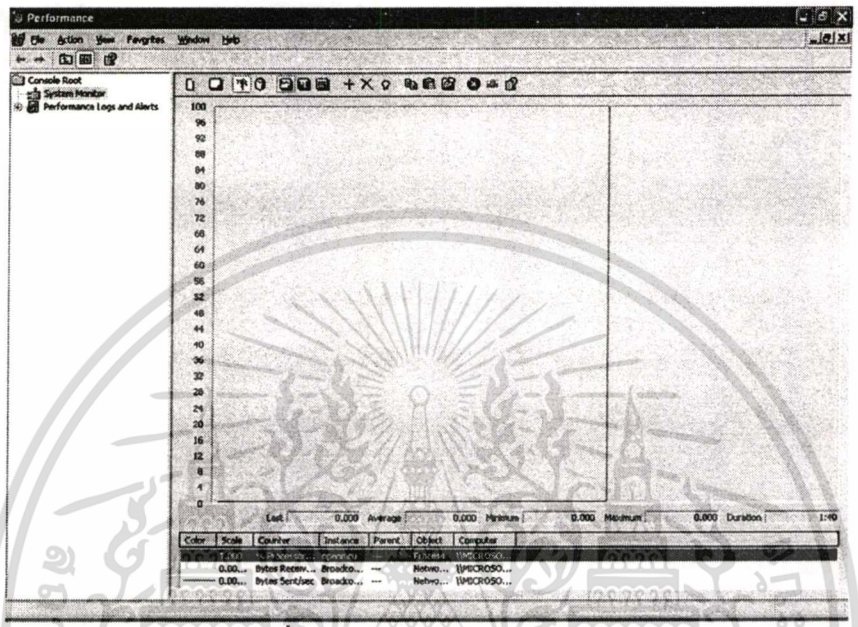
2) SIP Call Generator

สร้าง SIP call generator จาก oSIP [35] เวอร์ชัน 2.2.1-pre3 และ eXosip [36] เวอร์ชัน 0.9.0-pre1 โดยสามารถสื่อสารได้ทั้งภาพและเสียงและมีหลายห้องได้

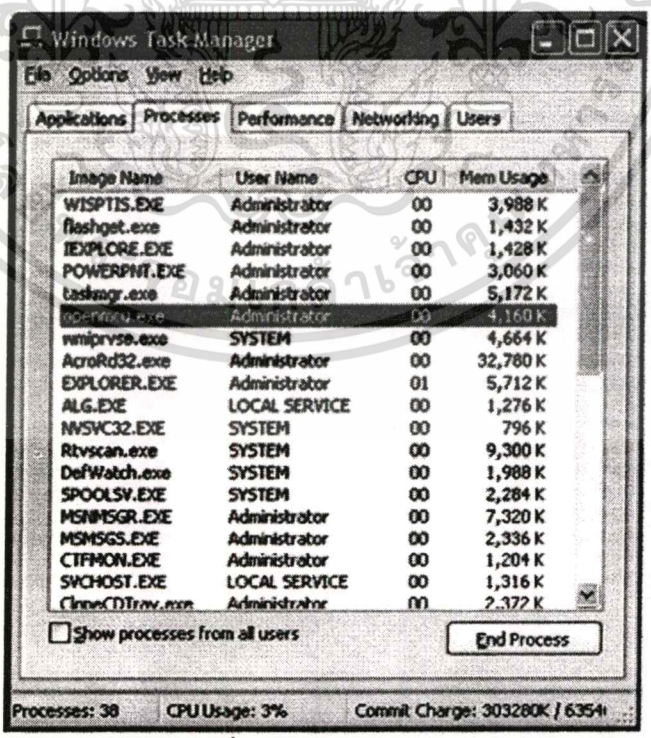
3) เครื่องมือวัดซีพียูโหลด (CPU Load) และเน็ตเวิร์คทราฟฟิก (Network Traffic)

ซีพียูโหลดและเน็ตเวิร์คทราฟฟิกของโปรแกรมใดๆ ไม่สามารถวัดค่าออกมาเป็นตัวเลขคงที่ค่าใดค่าหนึ่งได้ เนื่องจากในโปรแกรมสามารถรันอยู่ได้ในหลายสถานะ และสถานะนั้นๆ อาจจะมีซีพียูโหลดและเน็ตเวิร์คทราฟฟิกที่ไม่เท่ากันได้ ดังนั้นซีพียูโหลดและเน็ตเวิร์คทราฟฟิกจะต้องเป็นค่าเฉลี่ยในช่วงระยะเวลาหนึ่ง ซึ่งเราสามารถวัดได้โดยใช้เครื่องมือที่ชื่อว่า Performance

ดังรูปที่ 4.2 ซึ่งเป็นเครื่องมือที่มีอยู่แล้วในระบบปฏิบัติการวินโดส์ XP เนื่องจากไม่สามารถหาเครื่องมือวัดเน็ตเวิร์คทราฟฟิกสำหรับโปรแกรมใดโปรแกรมหนึ่งโดยเฉพาะ ดังนั้นการทดลองนี้จึงจำเป็นจะต้องวัดเน็ตเวิร์คทราฟฟิกโดยรวมของคอมพิวเตอร์ซึ่งในขณะทดลองจะต้องไม่ให้มีโปรแกรมที่ไม่เกี่ยวข้องใช้เน็ตเวิร์ค ซึ่งอาจจะทำให้การวัดคลาดเคลื่อนได้



รูปที่ 4.2 โปรแกรม Performance



รูปที่ 4.3 Task Manager

4) เครื่องมือวัดหน่วยความจำ

เครื่องมือวัดหน่วยความจำในการทดลองของเรานี้ใช้ Task Manager ดังรูปที่ 4.3 ซึ่งมีอยู่ในระบบปฏิบัติการวินโดส์ โดยผลที่วัดได้เป็น Physical Memory ที่ครอบครองโดยโปรแกรมนั้น

4.3 ผลการทดลองและบทวิเคราะห์

ในส่วนนี้เป็นการทดลองและวิเคราะห์ผลการทดลองที่ได้จากการยิงคอนเน็คชันจาก Call generator แล้ววัดค่าซีพียูโหลด, หน่วยความจำที่ใช้, เน็ตเวิร์คทราฟฟิกขาเข้า (Incoming Traffic) ที่จำนวนผู้ใช้และจำนวนห้องต่างๆของ OpenMCU ก่อนและหลังการเพิ่มประสิทธิภาพ และ SIP MCU เปรียบเทียบกัน รวมทั้งยังมีการเปรียบเทียบคุณภาพของวิดีโอโคเดคต่างๆ อีกด้วย ดังนี้

4.3.1 การเปรียบเทียบซีพียูโหลด

การทดลองนี้เป็นการวัดซีพียูโหลดของ OpenMCU ก่อนเพิ่มและหลังเพิ่มประสิทธิภาพ และ SIP MCU ที่จำนวนผู้ใช้และจำนวนห้องต่างๆเปรียบเทียบกัน

สมมุติฐานของการทดลอง

จากการเพิ่มประสิทธิภาพของ OpenMCU ในแง่ของคอมเพล็กซิตีดังแสดงในตารางที่ 3.1, การปรับปรุงเธรดซิงโครไนซ์เซชัน, การใช้ SSE2, และการส่งแมสเสจเพื่อควบคุมเทอร์มินัล จะทำให้การทำงานของโปรแกรมเร็วขึ้นส่งผลให้โปรแกรมใช้เวลาในการครอบครองซีพียูน้อยลง ทำให้ซีพียูโหลดเฉลี่ยของโปรแกรมต่ำลง ส่งผลให้ OpenMCU รองรับจำนวนผู้ใช้และจำนวนห้องได้มากขึ้น นอกจากนี้จากโครงสร้างใหม่ที่นำมาใช้พัฒนา SIP MCU ที่มีคอมเพล็กซิตีที่ดีกว่าก็น่าที่จะทำให้มีซีพียูโหลดน้อยลงกว่า OpenMCU ก่อนและหลังการเพิ่มประสิทธิภาพอีกด้วย

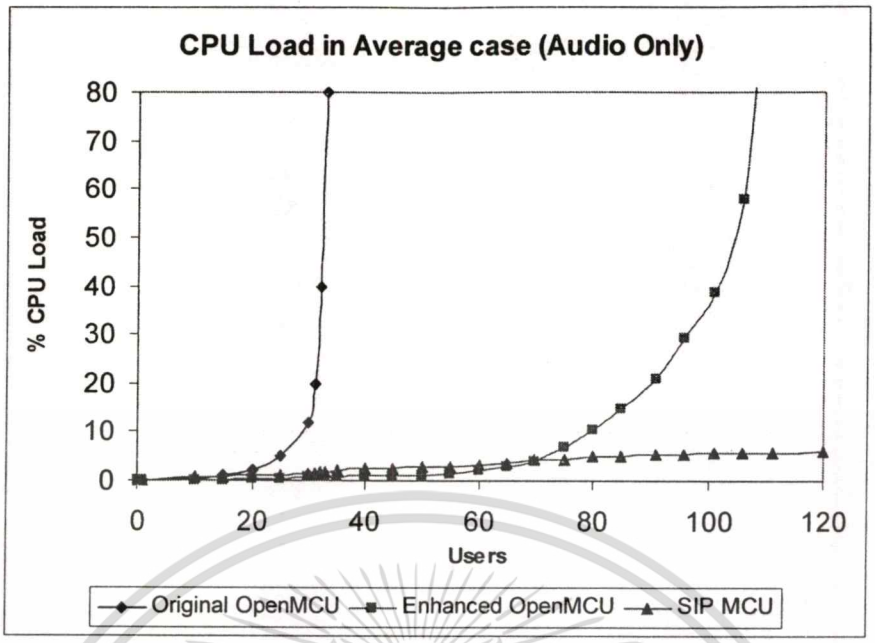
ปัจจัยการทดลอง

ในการทดลองนี้มีการกำหนดปัจจัยการทดลองเพิ่มเติมจากระบบทดสอบพื้นฐาน ดังนี้

1. ในการวัดซีพียูโหลดจะใช้ซีพียูโหลดเฉลี่ยเนื่องจากไม่สามารถวัดค่าที่แน่นอนได้
2. ในการวัดจะวัดในกรณีเฉลี่ย (Average Case) คือ มีผู้ใช้ที่พูดเพียง 1 คนในขณะใดขณะหนึ่งเท่านั้น ซึ่งเป็นกรณีที่เกิดขึ้นจริง ส่วนที่กรณีอื่นที่ผู้ใช้พูดมากกว่า 1 คนจะเป็นสภาวะที่ไม่น่าจะเกิดขึ้นบ่อยนักในการประชุมเนื่องจากจะฟังไม่รู้เรื่อง

ผลการทดลอง

1) การประชุมที่มีการสื่อสารทางเสียงเท่านั้นโดยเปรียบเทียบความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนผู้ใช้ในหนึ่งห้อง โดยเปรียบเทียบระหว่าง OpenMCU ก่อนและหลังการเพิ่มประสิทธิภาพ และ SIP MCU ได้ผลดังกราฟในรูปที่ 4.4

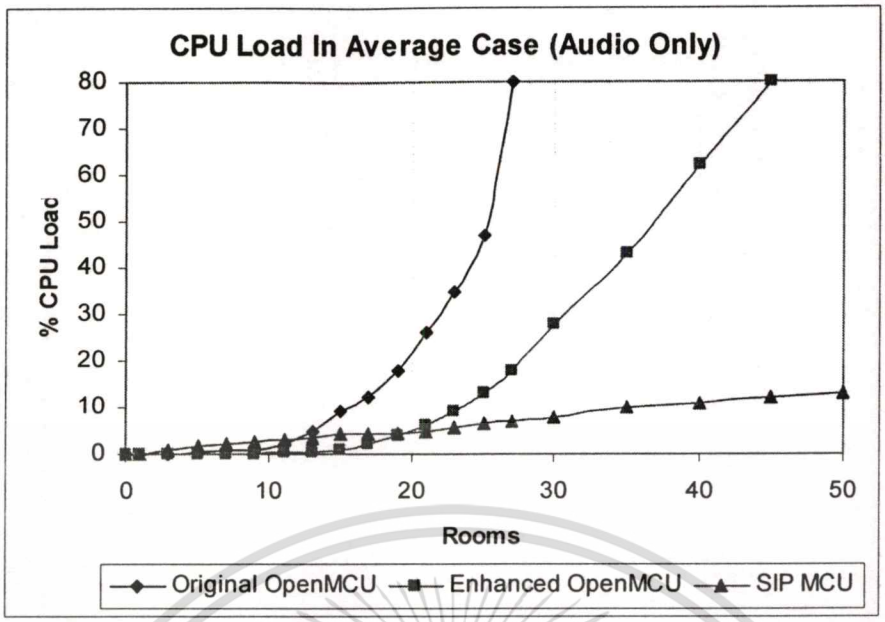


รูปที่ 4.4 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนผู้ใช้ในการประชุมทางเสียงเท่านั้นในหนึ่งห้อง

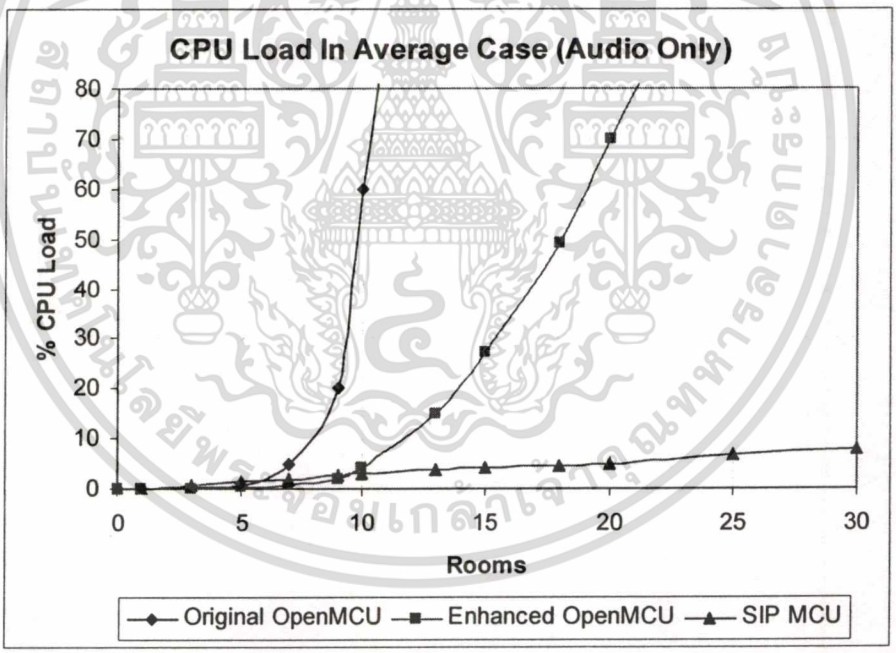
<u>วิเคราะห์ผลการทดลอง</u>

จากรูปที่ 4.4 ในหนึ่งห้องประชุมที่มีการประชุมทางเสียงเท่านั้น OpenMCU หลังการเพิ่มประสิทธิภาพสามารถรองรับจำนวนผู้ใช้ได้เพิ่มขึ้นเป็น 330% ของจำนวนผู้ใช้ที่รองรับได้ใน OpenMCU ตัวก่อนการเพิ่มประสิทธิภาพ เนื่องมาจาก OpenMCU หลังการเพิ่มประสิทธิภาพมีคอมเพล็กซิตีที่น้อยกว่าทำให้ใช้เวลาในการทำงานน้อยลง ส่งผลให้ใช้เวลาในการครอบครอง CPU น้อยลง ส่วน SIP MCU ซีพียูโหลดจะเพิ่มขึ้นช้ามากทำให้รองรับจำนวนผู้ใช้ได้เป็นจำนวนมาก เนื่องมาจากที่หนึ่งห้องจะทำการเปิดเธรดเพียง 3 เธรดเท่านั้นเมื่อมีคอนเน็คชันใหม่มาจะไม่เธรดเพิ่ม ต่างจาก OpenMCU ที่จะเปิด 3 เธรดต่อคอนเน็คชัน อีกทั้ง SIP MCU ยังไม่ต้องทำการเอนโคด, ดีโคด และผสมเสียง เพียงแต่จะกรองแพ็คเกจ (Packet) เสียง และส่งแพ็คเกจที่ถูกต้องไปยังเทอร์มินัลต่างๆเท่านั้น ส่วนสาเหตุที่ทำให้ซีพียูโหลดของ SIP MCU ไม่คงที่เนื่องมาจากว่าจะต้องทำการส่งข้อมูลเสียงให้กับเทอร์มินัลทั้งหมดซึ่งมีคอมเพล็กซิตีเป็น O(N) ทำให้ซีพียูโหลดเพิ่มขึ้นเป็นเชิงเส้นแต่ค่อนข้างเพิ่มขึ้นช้า ซึ่งจากการทดลองนี้สอดคล้องกับสมมุติฐานที่ตั้งไว้ทุกประการ

2) การประชุมที่มีการสื่อสารทางเสียงเท่านั้นโดยเปรียบเทียบความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนห้องประชุมของ OpenMCU ก่อนและหลังการเพิ่มประสิทธิภาพ และ SIP MCU ได้ผลดังกราฟในรูปที่ 4.5 และ 4.6



รูปที่ 4.5 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนห้องประชุม ในกรณีมีผู้ใช้ 4 คนต่อห้อง ในการประชุมทางเสียงเท่านั้น

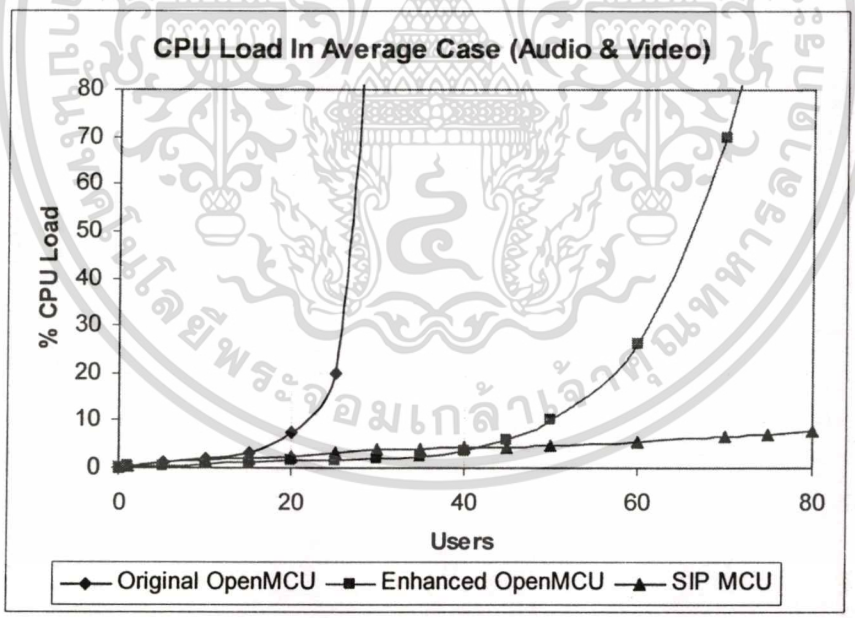


รูปที่ 4.6 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนห้องประชุม ในกรณีมีผู้ใช้ 8 คนต่อห้อง ในการประชุมทางเสียงเท่านั้น

วิเคราะห์ผลการทดลอง

จากรูปที่ 4.5 และ 4.6 ในการประชุมทางเสียงอย่างเดียว OpenMCU หลังการเพิ่มประสิทธิภาพสามารถรองรับจำนวนห้องได้มากขึ้นเป็น 150 % ของตัวเดิมสำหรับกรณีที่มีผู้ใช้ห้องละ 4 คน และ 180 % ในกรณีที่มีผู้ใช้ห้องละ 8 คน ทั้งนี้เนื่องมาจาก OpenMCU หลังการเพิ่มประสิทธิภาพมีคอมเพล็กซิตีที่น้อยกว่าทำให้ใช้เวลาในการทำงานน้อยลง ส่งผลให้ใช้เวลาในการครอบครอง CPU น้อยลง ส่วน SIP MCU จะเห็นว่าซีพียูโหลดเพิ่มขึ้นช้ามากทำให้สามารถรองรับผู้ใช้ได้เป็นจำนวนมาก สาเหตุเนื่องมาจากมีโครงสร้างและคอมเพล็กซิตีที่ดีกว่า คือจะมีคอมเพล็กซิตีของจำนวนเธรด และจำนวนซ็อคเกต (Socket) เพิ่มขึ้นตามจำนวนห้องเป็น O(N) แตกต่างจาก OpenMCU ที่จะเพิ่มขึ้นตามจำนวนคอนเน็คชัน ซึ่งจากการทดลองนี้สอดคล้องกับสมมุติฐานที่ตั้งไว้ทุกประการ

3) การประชุมทั้งภาพและเสียงเปรียบเทียบความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนผู้ใช้ของ OpenMCU ก่อนและหลังการเพิ่มประสิทธิภาพ และ SIP MCU โดยใช้ไฟล์วิดีโอมาตรฐาน Mother&Daughter ขนาด QCIF เป็นข้อมูลวิดีโอของ Call generator ได้ผลดังกราฟในรูปที่ 4.7

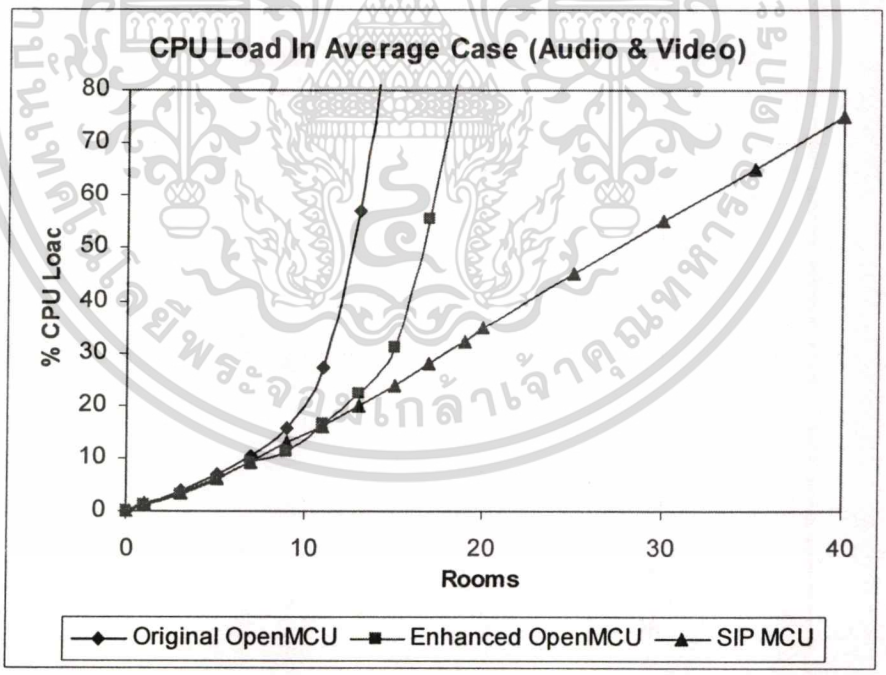


รูปที่ 4.7 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนผู้ใช้ ในการประชุมทั้งภาพและเสียงในหนึ่งห้อง

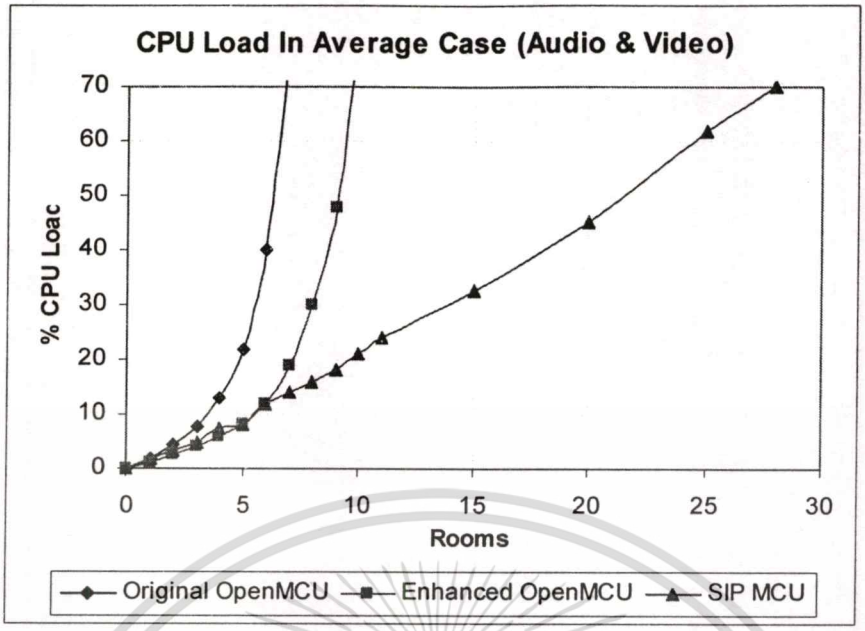
วิเคราะห์ผลการทดลอง

จากรูปที่ 4.7 ในหนึ่งห้องประชุมที่มีการประชุมทั้งภาพและเสียง OpenMCU หลังการเพิ่มประสิทธิภาพสามารถรองรับจำนวนผู้ใช้ได้เพิ่มขึ้นเป็น 185 % ของจำนวนผู้ใช้ที่รองรับได้ใน OpenMCU ก่อนการเพิ่มประสิทธิภาพ ทั้งนี้เนื่องมาจาก OpenMCU หลังการเพิ่มประสิทธิภาพมีคอมเพล็กซิตีที่น้อยกว่า ดังที่ได้กล่าวไปแล้ว อีกทั้งในส่วนของ Terminal Controller ยังช่วยหยุดการส่งวิดีโอของเทอร์มินัลที่ไม่ถูกแสดงผลอีกด้วย ทำให้คอมเพล็กซิตีของการดีโคดวิดีโอลดลง ส่วน SIP MCU สามารถรองรับจำนวนผู้ใช้ได้มากกว่าเนื่องมาจากมีการเอนโคดวิดีโอเพียงครั้งเดียวเท่านั้น แทนที่จำนวนการเอนโคดจะเพิ่มขึ้นตามจำนวนคอนเน็คชันเหมือน OpenMCU อีกทั้งยังมีจำนวนเธรดคงที่ที่ 3 เธรดต่างจาก OpenMCU ที่จำนวนเธรดจะเพิ่มขึ้นตามจำนวนคอนเน็คชัน ซึ่งจากการทดลองนี้สอดคล้องกับสมมุติฐานที่ตั้งไว้ทุกประการ

4) การประชุมทั้งภาพและเสียง เปรียบเทียบความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนห้องประชุมของ OpenMCU ก่อนและหลังการเพิ่มประสิทธิภาพ และ SIP MCU โดยใช้ไฟล์วิดีโอมาตรฐาน Mother&Daughter, Foreman, Akiyo และ Silent ขนาด QCIF เป็นข้อมูลวิดีโอของ Call generator ได้ผลดังกราฟในรูปที่ 4.8 และ 4.9



รูปที่ 4.8 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนห้องประชุม ในกรณีมีผู้ใช้ 4 คนต่อห้อง ในการประชุมทั้งภาพและเสียง



รูปที่ 4.9 กราฟความสัมพันธ์ระหว่างซีพียูโหลดเฉลี่ยกับจำนวนห้องประชุม ในกรณีมีผู้ใช้ 8 คนต่อห้อง ในการประชุมทั้งภาพและเสียง

<u>วิเคราะห์ผลการทดลอง</u>

จากรูปที่ 4.8 และ 4.9 ในการประชุมทั้งภาพและเสียง OpenMCU หลังการเพิ่มประสิทธิภาพสามารถรองรับจำนวนห้องได้มากขึ้นเป็น 140 % ของตัวเดิมสำหรับทั้งกรณีที่มีผู้ใช้ห้องละ 4 คน และ 8 คน ทั้งนี้เนื่องมาจาก OpenMCU หลังการเพิ่มประสิทธิภาพมีคอมเพล็กซิตีที่น้อยกว่าทำให้ใช้เวลาในการทำงานน้อยลง ส่งผลให้ใช้เวลาในการครอบครอง CPU น้อยลง อีกทั้งในส่วนของ Terminal Controller ยังช่วยหยุดการส่งวิดีโอของเทอร์มินัลที่ไม่ถูกแสดงผลอีกด้วย ทำให้คอมเพล็กซิตีของการดีโคดวิดีโอลดลง ทำให้ในกรณีของ 8 คนต่อห้องมีซีพียูโหลดใกล้เคียงกับ 4 คนต่อห้อง ส่วน SIP MCU จะเห็นว่าซีพียูโหลดเพิ่มขึ้นช้ามากทำให้สามารถรองรับผู้ใช้ได้เป็นจำนวนมาก คือ สามารถรองรับจำนวนห้องได้มากขึ้นเป็น 240 % ของ OpenMCU ตัวเดิมสำหรับกรณีที่มีผู้ใช้ห้องละ 4 คน และเป็น 350 % ในกรณีห้องละ 8 คน สาเหตุเนื่องมาจากมีโครงสร้างและคอมเพล็กซิตีที่ดีกว่า คือจะมีคอมเพล็กซิตีของจำนวนเธรด , จำนวนซ็อคเกต และจำนวนการเอนโคดวิดีโอ เพิ่มขึ้นตามจำนวนห้องแตกต่างจาก OpenMCU ที่จะเพิ่มขึ้นตามจำนวนคอนเน็คชัน ซึ่งจากการทดลองนี้สอดคล้องกับสมมุติฐานที่ตั้งไว้ทุกประการ

4.3.2 การเปรียบเทียบจำนวนหน่วยความจำ

การทดลองนี้เป็นการวัดจำนวนหน่วยความจำที่ครอบครองโดย OpenMCU ก่อนเพิ่มและหลังเพิ่มประสิทธิภาพ ที่จำนวนผู้ใช้ต่างๆเปรียบเทียบกัน

สมมติฐานของการทดลอง

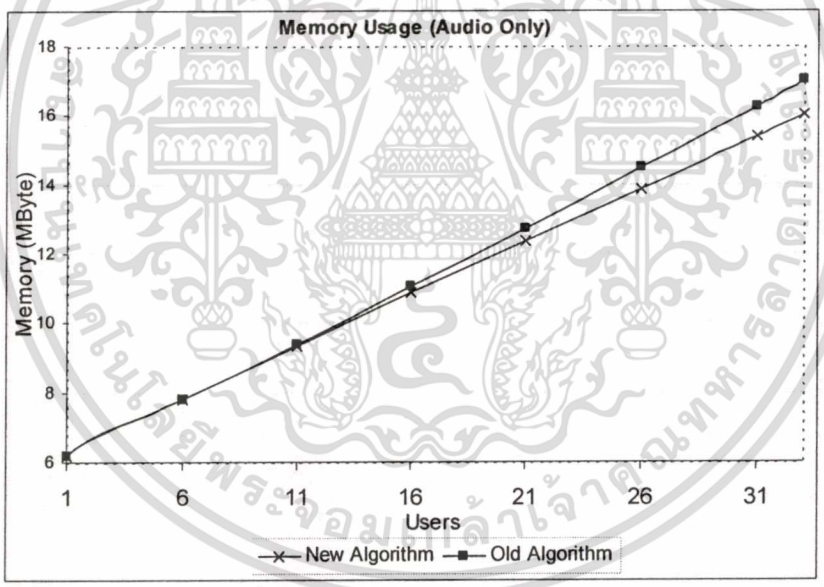
จากการเพิ่มประสิทธิภาพของ OpenMCU ในแง่ของสเปซคอมเพล็กซิตีของจำนวนออดิโอบัฟเฟอร์จาก $O(N^2)$ เป็น $O(N)$ ทำให้ OpenMCU ครอบครองหน่วยความจำน้อยลงดังสมการที่ 3.1

$$mem = (N^2 - 2N) * BufferSize$$

ซึ่งทำให้ OpenMCU อ่าน I/O น้อยลงซึ่งถือว่าเป็นกิจกรรมที่ทำงานได้ช้ามาก ซึ่งส่งผลให้ประสิทธิภาพของ OpenMCU ดีขึ้น

ผลการทดลอง

การประชุมที่มีการสื่อสารทั้งภาพและเสียง โดยเปรียบเทียบความสัมพันธ์ระหว่างหน่วยความจำกับจำนวนผู้ใช้ใน 1 ห้องเปรียบเทียบระหว่าง OpenMCU ก่อนและหลังการเพิ่มประสิทธิภาพ ได้ผลดังกราฟในรูปที่ 4.10



รูปที่ 4.10 กราฟความสัมพันธ์ระหว่างหน่วยความจำกับจำนวนผู้ใช้ใน 1 ห้อง

วิเคราะห์ผลการทดลอง

จากการทดลองจะเห็นว่าที่จำนวนผู้ใช้ 33 คน ปริมาณหน่วยความจำที่ลดลงจากสมการที่ 3.1 มีขนาดเท่ากับ $(33^2–2*33)$*BufferSize โดยขนาดของ BufferSize เท่ากับ 960 Bytes ดังนั้นหน่วยความจำที่ลดลงมีขนาดเท่ากับ 982,080 Bytes หรือประมาณ 1 MBytes ตรงกับขนาดหน่วยความจำที่ลดลงของเส้นกราฟที่จำนวนผู้ใช้ 33 คน ซึ่งส่งผลต่อประสิทธิภาพของ OpenMCU โดยหน่วยความจำที่ลดลงจะทำให้ OpenMCU อ่านเขียนหน่วยความจำน้อยลงซึ่งทำให้ประสิทธิภาพดีขึ้น เนื่องจากการอ่านเขียน I/O เป็นขั้นตอนที่ช้ามาก สอดคล้องกับสมุติฐานที่ตั้งไว้

ส่วนสาเหตุที่เส้นกราฟของ OpenMCU ตัวก่อนการเพิ่มประสิทธิภาพมีการเพิ่มขึ้นแบบเชิงเส้นแทนที่จะเป็นพาราโบลาตามการเพิ่มขึ้นของออดิโอบัฟเฟอร์ เนื่องจากจำนวนหน่วยความจำที่วัดได้ได้รับผลกระทบจากความต้องการหน่วยความจำในส่วนอื่นๆที่ใหญ่กว่าออดิโอบัฟเฟอร์มากและมีการเพิ่มขึ้นในเชิงเส้นของ OpenMCU เช่น จำนวนออบเจ็กต์คอนเน็คชัน และจำนวนวิดีโอบัฟเฟอร์

4.3.3 การเปรียบเทียบขนาดของเน็ตเวิร์คทราฟฟิก

การทดลองนี้เป็นการวัดเน็ตเวิร์คทราฟฟิกของ OpenMCU ก่อนเพิ่มและหลังเพิ่มประสิทธิภาพและ SIP MCU ที่จำนวนผู้ใช้และจำนวนห้องต่างๆเปรียบเทียบกัน

สมมุติฐานของการทดลอง

จากการเพิ่มประสิทธิภาพของ OpenMCU ในแง่ของเน็ตเวิร์คแบนด์วิดท์ที่ใช้ในการสื่อสารโดยวิธีการส่งแมสเสจเพื่อควบคุมเทอร์มินัล ทำให้ OpenMCU และ SIP MCU ต้องการแบนด์วิดท์น้อยลง โดยในด้านทราฟฟิกขาเข้า (Incoming Traffic) ภายหลังจากที่มีผู้ใช้มากกว่า 4 คนไปแล้วแบนด์วิดท์จะค่อนข้างคงที่ซึ่งเกิดจากการส่งแมสเสจไปหยุดการส่งวิดีโอสตรีมของผู้ใช้ที่ไม่ได้ถูกแสดงผล

ปัจจัยการทดลอง

ในการทดลองนี้มีการกำหนดปัจจัยการทดลองเพิ่มเติมจากระบบทดสอบพื้นฐาน ดังนี้

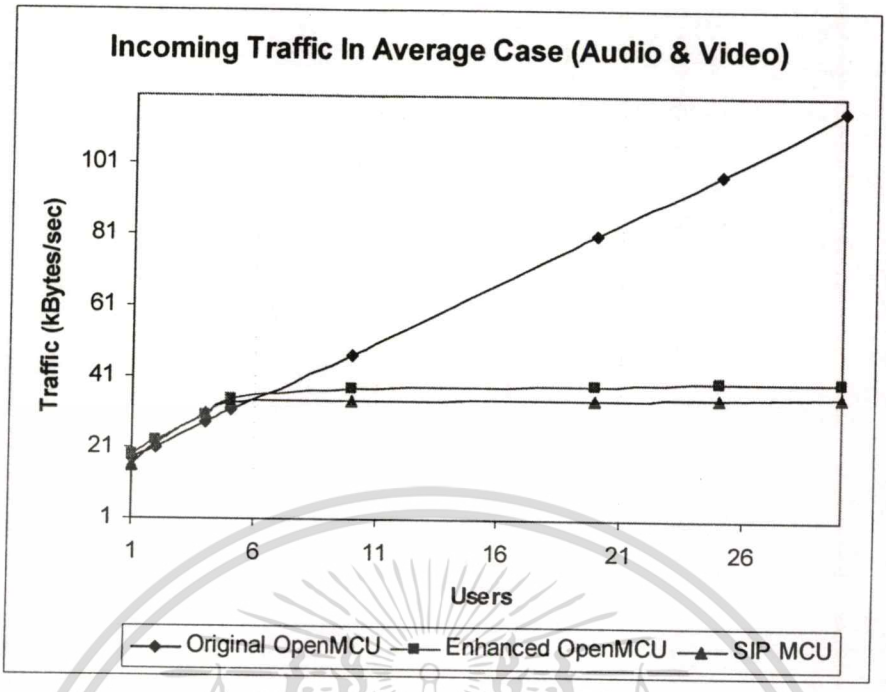
1. การทดลองจะทำที่กรณีเฉลี่ยคือมีคนพูดคนเดียวเท่านั้น
2. ทราฟฟิกที่ได้เป็นทราฟฟิกโดยเฉลี่ย เนื่องจากทราฟฟิกมีการเปลี่ยนแปลงไม่คงที่

ผลการทดลอง

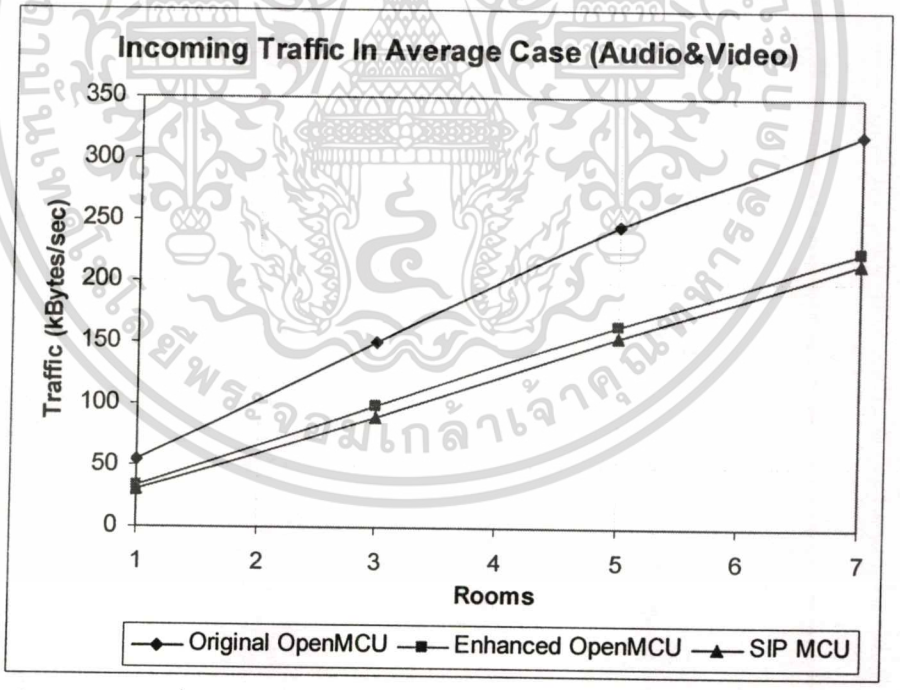
1. การประชุมที่มีการสื่อสารทั้งภาพและเสียง โดยเปรียบเทียบความสัมพันธ์ระหว่างขนาดของแบนด์วิดท์ที่ต้องการกับจำนวนผู้ใช้ใน 1 ห้องประชุมเปรียบเทียบระหว่าง OpenMCU ก่อนและหลังการเพิ่มประสิทธิภาพ และ SIP MCU ได้ผลดังกราฟในรูปที่ 4.11 และ 4.12

วิเคราะห์ผลการทดลอง

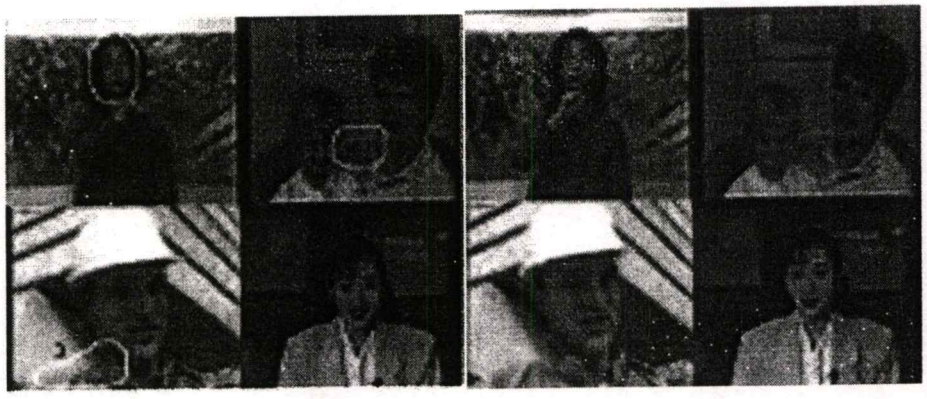
จากรูปที่ 4.11 OpenMCU ภายหลังการเพิ่มประสิทธิภาพ และ SIP MCU ต้องการแบนด์วิดท์ในด้านขาเข้าน้อยลง และค่อนข้างจะคงที่ที่ผู้ใช้มากกว่า 4 คนเนื่องจาก MPEG-4 เทอร์มินัลที่พัฒนาขึ้นมีส่วนช่วยในการหยุดการส่งวิดีโอเมื่อได้รับแมสเสจ “STOP” จาก OpenMCU และ SIP MCU ซึ่งเป็นเหตุผลทำให้ทราฟฟิกขาเข้าในรูป 4.12 เมื่อมีจำนวนห้องมากขึ้นลดลงตามไปด้วย ซึ่งจากการทดลองนี้สอดคล้องกับสมมุติฐานที่ตั้งไว้ทุกประการ



รูปที่ 4.11 กราฟความสัมพันธ์ระหว่างขนาดของ Incoming Traffic กับจำนวนผู้ใช้ใน 1 ห้องประชุม



รูปที่ 4.12 กราฟความสัมพันธ์ระหว่างขนาดของ Incoming Traffic กับจำนวนห้องในกรณีผู้ใช้ 8 คนต่อห้อง



(a) (b)

รูปที่ 4.13 เปรียบเทียบคุณภาพของวิดีโอโดยใช้สายตา (a) H.263 จาก Original OpenMCU และ (b) MPEG-4 จาก Enhanced OpenMCU และ SIP MCU

4.3.4 การเปรียบเทียบคุณภาพของวิดีโอโดยใช้สายตา

จากรูปที่ 4.13 เป็นเปรียบเทียบภาพวิดีโอจากการทดลองโดยใช้สายตา โดยภาพวิดีโอมาจากไฟล์วิดีโอมาตรฐานขนาด QCIF โดยผลของการเปรียบเทียบเป็นดังนี้

- ที่ใบหน้ารูปบนซ้ายของ (a) จะเห็นว่าภาพวิดีโอ H.263 ไม่ค่อยคมชัด สู้รูป (b) ซึ่งเป็น MPEG-4 ไม่ได้
- ที่รูปบนขวาของ (a) ที่บริเวณไหล่จะเห็นว่า H.263 จะเบลอๆแต่ MPEG-4 จะคมชัดกว่า
- ที่รูปล่างซ้ายของ (a) จะเห็นได้อย่างชัดเจนบริเวณไหล่เนื่องจากเป็นรูปที่เคลื่อนไหวมาก พบว่าคุณภาพของ H.263 สู้ MPEG-4 ไม่ได้เช่นกัน
- รูปล่างขวาของ (a) และ (b) เป็นรูปที่เคลื่อนไหวน้อยทำให้ไม่เห็นความแตกต่างมากนัก

4.4 การเปรียบเทียบจำนวนผู้ใช้กับ MCU ในท้องตลาด

จากการเปรียบเทียบในตารางที่ 4.1 และ 4.2 พบว่า Enhanced OpenMCU และ SIP MCU จะสามารถรองรับจำนวนผู้ใช้ได้มากกว่า MCU ที่มีในท้องตลาดมาก แต่ในแง่ของความเป็นจริงแล้วจะพบว่าสินค้า MCU ในท้องตลาดเหล่านี้มีลูกเล่นที่มากกว่า Enhanced OpenMCU และ SIP MCU มาก

ตารางที่ 4.1 เปรียบเทียบจำนวนผู้ใช้ในกรณีประชุมแบบเสียงอย่างเดียวที่ 128 kbps

Voice Only 1 ห้องที่ 128 kbps	
Product	**Ports**
Origianl OpenMCU	33
Enhanced OpenMCU	105
SIP MCU	> 200
Radvision MCU 15	26
Radvision MCU 30	45
Radvision MCU 60	90
Codian MCU-4205	24
Codian MCU-4210	40
Codian MCU-4220	80
Cisco IP/VC 3511 MCU	26

ตารางที่ 4.2 เปรียบเทียบจำนวนผู้ใช้ในกรณีประชุมทั้งภาพและเสียงที่ 128 kbps

Voice & Video 1 ห้องที่ 128 kbps	
Product	**Ports**
Origianl OpenMCU	28
Enhanced OpenMCU	68
SIP MCU	>180
Radvision MCU 15	15
Radvision MCU 30	30
Radvision MCU 60	60
Codian MCU-4205	12
Codian MCU-4210	20
Codian MCU-4220	40
Cisco IP/VC 3511 MCU	15

4.5 สรุปผลการเพิ่มประสิทธิภาพ

จากการเพิ่มประสิทธิภาพของ OpenMCU ในแง่ของคอมเพล็กซิตี, การปรับปรุงเธรดซิงโครไนซ์เซชัน, การใช้ SSE2, การส่งแมสเสจเพื่อควบคุมเทอร์มินัลและการใช้โคเดค MPEG-4 ซึ่งทำให้การทำงานของโปรแกรมเร็วขึ้น ส่งผลให้โปรแกรมครอบครองซีพียูน้อยลงทำให้ซีพียูโหลดเฉลี่ยของโปรแกรมต่ำลง ทำให้ OpenMCU รองรับจำนวนผู้ใช้และจำนวนห้องได้มากขึ้น, ใช้แบนด์วิดท์ในการสื่อสารน้อยลง และครอบครองหน่วยความจำน้อยลงซึ่งทำให้ OpenMCU อ่าน I/O น้อยลงซึ่งถือว่าเป็นกิจกรรมที่ทำงานได้ช้ามาก ส่งผลให้ประสิทธิภาพของ OpenMCU ดีขึ้นมาก เมื่อนำภาพวิดีโอของ Original OpenMCU ที่เป็น H.263 และ Enhanced OpenMCU ที่ Bit rate เท่ากันมาเปรียบเทียบกัน พบว่า Enhanced OpenMCU ให้คุณภาพของวิดีโอที่ดีกว่าอีกด้วย และเมื่อนำ Original OpenMCU และ Enhanced OpenMCU มาเปรียบเทียบกับ SIP MCU พบว่า SIP MCU มีซีพียูโหลดและทราฟฟิกขาเข้าที่ต่ำกว่าเนื่องมาจากโครงสร้างและคอมเพล็กซิตีที่ดีกว่า ซึ่งทำให้รองรับจำนวนผู้ใช้ได้มากขึ้น สอดคล้องกับสมมุติฐานที่ตั้งไว้ทุกประการ

บทที่ 5
บทสรุป

5.1 สรุปงานวิจัยที่นำเสนอ

วิทยานิพนธ์นี้ได้ทำการเพิ่มประสิทธิภาพของ H.323 OpenMCU ซึ่งเป็นโอเพ่นซอร์ส H.323 MCU ที่มีประสิทธิภาพที่ไม่ดีนัก บนระบบปฏิบัติการวินโดส์ให้รองรับจำนวนผู้ใช้ได้มากขึ้น โดยได้เสนอวิธีการเพิ่มประสิทธิภาพมากมายซึ่งนำมาแก้ไขคอขวดของ OpenMCU ดังต่อไปนี้

<u>การเพิ่มประสิทธิภาพของ OpenMCU ในส่วนออดิโอ</u>

- การแก้ไขโครงสร้างข้อมูลของออดิโอบัฟเฟอร์ โดยใช้พอยน์เตอร์แทนการสำเนาออดิโอไปยังแต่ละคอนเน็คชัน ส่งผลให้คอมเพล็กซิตีลดลงจาก $O(N^2)$ เป็น $O(N)$
- การตรวจสอบความดังของออดิโอก่อนสำเนาลงบัฟเฟอร์ ซึ่งส่งผลให้ในกรณีเฉลี่ยเมื่อนับทุกคอนเน็คชันแล้ว จำนวนการสำเนาจะลดลงจาก $O(N^2)$ เป็น $O(1)$ และการผสมเสียงจะลดลงจาก $O(N^2)$ เป็น $O(N)$
- การแก้ไขอัลกอริธึมการผสมออดิโอเป็น SIMD SSE2 ทำให้ทำการผสมออดิโอเร็วขึ้น 8 เท่า

<u>การเพิ่มประสิทธิภาพในส่วนวิดีโอ</u>

- การใช้การส่งแมสเสจเพื่อควบคุมเทอร์มินัล ทำให้เทอร์มินัลที่ไม่แสดงผลไม่ต้องส่งวิดีโอสตรีมมาที่ MCU ส่งผลให้ลดทราฟฟิกขาเข้า (Incoming Traffic) ไปได้มาก
- การใช้โคเดควิดีโอตามมาตรฐาน MPEG-4 แทนโคเดคเดิมเพื่อลดแบนด์วิดท์ ของ MCU

<u>การเพิ่มประสิทธิภาพทั้งในส่วนออดิโอและวิดีโอ</u>

- การแก้ไขออบเจ็กต์เธรดซิงโครไนซ์เซชันจาก Mutex เป็น Critical Section และ Events เพื่อลดโอเวอร์เฮด (overhead) ของระบบ
- การใช้รีดเดอร์ไรท์เตอร์เธรดซิงโครไนซ์เซชัน (Readers-Writers Thread Synchronization) ในการเข้าถึงออดิโอบัฟเฟอร์ และวิดีโอบัฟเฟอร์เพื่อให้สามารถทำงานต่างๆพร้อมกันได้หลายเธรดทำให้โปรแกรมทำงานเร็วขึ้น
- การใช้บาลานซ์ไบนารี่เซิร์ชทรี (Balanced Binary Search Tree) และลิงค์ลิสท์ (Linked List) แทนดิคท์ (Dict) เพื่อลดข้อผิดพลาดจากการใช้การเข้าถึงบัฟเฟอร์แบบหลายเธรดและทำให้คอมเพล็กซิตีของการดึง ออบเจ็กต์ทั้งหมดในดิคท์ลดลงจาก $O(N)$ เป็น $O(1)$

หลังจากได้ทำการเพิ่มประสิทธิภาพแล้ว เราได้นำโครงสร้างที่พัฒนาแล้วนี้มาสร้าง SIP MCU ซึ่งมีโครงสร้างที่ดีกว่า OpenMCU คือมีคอมเพล็กซิตีส่วนใหญ่เพิ่มขึ้นตามจำนวนห้องแทนที่จะเพิ่มขึ้นตามจำนวนคอนเน็คชันเหมือนกับ OpenMCU

นอกจากที่กล่าวไปแล้ว เรายังได้ทำการพัฒนา MPEG-4 Terminal ขึ้นบนระบบปฏิบัติการวินโดส์ เพื่อรองรับ OpenMCU ที่ได้ทำการเพิ่มประสิทธิภาพแล้ว และ SIP MCU ซึ่งมีคุณลักษณะสำคัญดังนี้

- รองรับมาตรฐาน MPEG-4
- รองรับการส่งแมสเสจจาก OpenMCU
- รองรับการเชื่อมต่อแบบจุดต่อจุด (Point-to-Point) โดยปราศจาก MCU (H.323 เทอร์มินัลเท่านั้น)

ภายหลังการเพิ่มประสิทธิภาพของ OpenMCU เราได้ทดลองวัดประสิทธิภาพของ OpenMCU บนระบบปฏิบัติการวินโดส์เปรียบเทียบก่อนและหลังการเพิ่มประสิทธิภาพและ SIP MCU ได้ผลดังนี้

- ในหนึ่งห้องประชุมที่มีการประชุมทางเสียงเท่านั้น OpenMCU หลังการเพิ่มประสิทธิภาพสามารถรองรับจำนวนผู้ใช้ได้เพิ่มขึ้นเป็น 330% ของจำนวนผู้ใช้ที่รองรับได้ใน OpenMCU ตัวก่อนการเพิ่มประสิทธิภาพ ส่วน SIP MCU สามารถรองรับจำนวนผู้ใช้ได้มากกว่า OpenMCU มาก
- ในการประชุมทางเสียงอย่างเดียว OpenMCU หลังการเพิ่มประสิทธิภาพสามารถรองรับจำนวนห้องได้มากขึ้นเป็น 150 % ของตัวเดิมสำหรับกรณีผู้ใช้ห้องละ 4 คน และ 180 % ในกรณีผู้ใช้ห้องละ 8 คน ส่วน SIP MCU สามารถรองรับจำนวนห้องได้มากกว่า OpenMCU มาก
- ในหนึ่งห้องประชุมที่มีการประชุมทั้งภาพและเสียง OpenMCU หลังการเพิ่มประสิทธิภาพสามารถรองรับจำนวนผู้ใช้ได้เพิ่มขึ้นเป็น 185% ของจำนวนผู้ใช้ที่รองรับได้ใน OpenMCU ตัวก่อนการเพิ่มประสิทธิภาพ ส่วน SIP MCU สามารถรองรับจำนวนผู้ใช้ได้มากกว่า OpenMCU มาก
- ในการประชุมทั้งภาพและเสียง OpenMCU หลังการเพิ่มประสิทธิภาพสามารถรองรับจำนวนห้องได้มากขึ้นเป็น 140 % ของตัวเดิม สำหรับทั้งกรณีผู้ใช้ห้องละ 4 คน และ 8 คน
- OpenMCU ครอบครองหน่วยความจำน้อยลงซึ่งทำให้ต้องอ่าน I/O น้อยลงซึ่งถือว่าเป็นกิจกรรมที่ทำงานได้ช้ามากส่งผลให้ประสิทธิภาพของ OpenMCU ดีขึ้น

- OpenMCU หลังจากการเพิ่มประสิทธิภาพและ SIP MCU ต้องการแบนด์วิดท์ขาเข้าน้อยลง โดยในการประชุมห้องเดียว ภายหลังจากที่มีผู้ใช้มากกว่า 4 คนไปแล้วแบนด์วิดท์จะค่อนข้างคงที่

ซึ่งจากผลการทดลองนี้แสดงให้เห็นว่า การเพิ่มประสิทธิภาพของเราให้ผลดีเป็นอย่างมาก ทั้งในแง่ของซีพียูโหลด, เน็ตเวิร์คแบนด์วิดท์ และหน่วยความจำที่ครอบครองโดย OpenMCU ที่ลดลง ซึ่งส่งผลให้ OpenMCU รองรับจำนวนผู้ใช้ได้มากขึ้น โดยหลังการเพิ่มประสิทธิภาพของ OpenMCU แล้วเราได้นำโครงสร้างของ OpenMCU ที่มีประสิทธิภาพแล้วนี้มาสร้าง SIP MCU ซึ่งได้แก้ไขข้อบกพร่องต่างๆของ OpenMCU ให้ดีขึ้น ซึ่งจากการทดลองก็ให้ผลที่ดีกว่า OpenMCU ตัวก่อนและหลังการเพิ่มประสิทธิภาพ

5.2 ปัญหาและอุปสรรค

ในปัจจุบัน H.323 MCU ส่วนใหญ่เป็นฮาร์ดแวร์ และที่เป็นซอฟต์แวร์ก็ไม่สามารถดาวน์โหลดมารันเปรียบเทียบประสิทธิภาพกับ OpenMCU ที่ได้เพิ่มประสิทธิภาพแล้วได้ เราจึงเปรียบเทียบให้ดูกับ OpenMCU ก่อนการเพิ่มประสิทธิภาพแล้วเท่านั้น นอกจากนี้การเพิ่มประสิทธิภาพยังอ้างอิงระบบปฏิบัติการวินโดส์ซึ่งจริงๆแล้ว OpenMCU สามารถรันได้ในระบบปฏิบัติการตระกูล UNIX ได้อีกด้วย แต่ด้วยความสามารถและความรู้อันจำกัดของผู้ทำการวิจัยที่มีความรู้ทางโปรแกรมมิ่งส่วนใหญ่บนระบบปฏิบัติการวินโดส์เท่านั้น เราจึงไม่สามารถเพิ่มประสิทธิภาพของ OpenMCU บน UNIX ได้

สำหรับงานวิจัยนี้ทดลองบนเน็ตเวิร์คแลน 10/100 Mbits/sec ซึ่งอาจจะถือได้ว่าเน็ตเวิร์คมีประสิทธิภาพมาก แต่จากการทดลองที่เน็ตเวิร์คต่ำๆเช่น Dial-up modem 56 kbits/sec ภาพที่ได้มีความไม่ต่อเนื่องอันเนื่องมาจาก Delay Jitter ในเน็ตเวิร์คมีมากทำให้ภาพมีความไม่ต่อเนื่อง อีกทั้งเสียงยังขาดๆหายๆอีกด้วย สำหรับประสิทธิภาพที่วัดได้นั้นเป็นเพียงแค่แนวโน้มเท่านั้นไม่สามารถวัดค่าได้อย่างละเอียดเนื่องจากซีพียูโหลด และเน็ตเวิร์คแบนวิดท์มีความเปลี่ยนแปลงค่อนข้างมากในช่วงระยะเวลาหนึ่ง อีกทั้งเครื่องมือที่ใช้วัดประสิทธิภาพยังต้องใช้สายตาในการคาดคะเนมาก นอกจากนี้เมื่อเวลาผ่านไปซีพียูโหลดของ OpenMCU ค่อนข้างจะลดลงเรื่อยๆ เนื่องจากข้อมูลที่ต้องการโดย OpenMCU ได้ถูกโหลดเข้ามาในแคช (Cache) เรียบร้อยแล้ว ทำให้ซีพียูทำงานน้อยลง เราจึงต้องวัดค่าในช่วงเวลาใดเวลาหนึ่งเท่านั้น ซึ่งอาจจะมีความคลาดเคลื่อนไปบ้าง แต่ผลของการเพิ่มประสิทธิภาพก็ยังยอมรับได้เนื่องจากประสิทธิภาพของ OpenMCU ก่อนและหลังการเพิ่มประสิทธิภาพแตกต่างกันชัดเจนมากทำให้แสดงผลออกมาทางเส้นกราฟที่เด่นชัด

5.3 แนวทางการพัฒนาต่อ

ในอนาคตจะทำการพัฒนา SIP MCU ให้สมบูรณ์ยิ่งขึ้น อีกทั้งยังต้องพัฒนา Jitter Buffer ให้การสื่อสารทั้งภาพและเสียงราบเรียบสมบูรณ์ยิ่งขึ้น นอกจากรองรับ SIP Terminal ที่รันบนระบบ PC แล้วต่อไป SIP MCU นี้ยังจะต้องรองรับ SIP Termianl บนระบบอื่นๆเช่น บน Mobile Phone อีกด้วย

หลังจากที่พัฒนา SIP MCU เสร็จสิ้นแล้ว เราก็จะรวม SIP MCU และ H.323 OpenMCU ที่ได้เพิ่มประสิทธิภาพแล้วเข้าด้วยกันเป็น Universal MCU เพื่อให้รองรับได้ทั้ง H.323 Terminal และ SIP Terminal อีกด้วย

เอกสารอ้างอิง

[1] Goode B., "Voice over Internet protocol (VoIP)", Proceedings of the IEEE, Volume 90, Issue 9, Sept. 2002. pp. 1495 – 1517

[2] Chong H.M. and Matthews H.S., "Comparative analysis of traditional telephone and voice-over-Internet protocol (VoIP) systems", Electronics and the Environment 2004, Conference Record. 2004, IEEE International Symposium, 10-13 May 2004. pp. 106 – 111

[3] Puthiprasert S., Kamolphiwong S. and Wuttisittikulkij L., "Internet Telephony Protocols", NECTEC Technical Journal, Vol. III, No. 10, 2001. pp.69-84.

[4] "MSN Messenger" [Online]. Available : http://messenger.msn.com/

[5] "Microsoft NetMeeting" [Online]. Available : http://www.microsoft.com/windows/netmeeting/

[6] International Telecommunication Union, "Packet based multimedia communication system", recommendation H.323, Telecommunication Standardization Sector of ITU, Feb. 1998

[7] Thom G.A., "H.323: the multimedia communications standard for local area networks", Communications Magazine, IEEE Volume 34, Issue 12, Dec. 1996. pp. 52 - 56

[8] Liu H. and Mouchtaris P., "Voice over IP signaling: H.323 and beyond", Communications Magazine, IEEE Volume 38, Issue 10, Oct. 2000.pp 142 – 148

[9] Handley M., Schulzrinne H., Schooler E., and Rosenberg J., "SIP : session initial protocol", RFC 2543, Internet Engineering Task Force, Mar 1999

[10] Tam K.K. and Goh H.L., "Session Initiation Protocol", Industrial Technology, IEEE ICIT '02, 2002 IEEE International Conference on Volume 2, 11-14 Dec. 2002. pp. 1310 - 1314

[11] Johnston A.B. SIP: Understanding the Session Initiation Protocol. Second edn. Artech House, 2004

[12] "The ITU Telecommunication Standardization Sector (ITU-T)" [Online]. Available : http://www.itu.int/ITU-T/

[13] "IETF Home Page" [Online]. Available : http://www.ietf.org/

[14] "The OpenH323 Project" [Online]. Available : http://www3.openh323.org/

[15] NAGAO T. and OKAMURA K., "High Performance MCU for Scalable Remote Conference System", DICOMO2002, July 2002.

[16] Sikora T., "The MPEG-4 video standard verification model", Circuits and Systems for Video Technology, IEEE Transactions on Volume 7, Issue 1, Feb. 1997. pp. 19 – 31

[17] Tung Y.S. ,Wu J.L., and Ho C.C., "Architecture design of an MPEG-4 system", Consumer Electronics, ICCE. 2000 Digest of Technical Papers. International Conference, 13-15 June 2000. pp. 122 - 123

[18] Koskelainen P., Schulzrinne H., and Wu X., "A SIP-based Conference Control Framework", NOSSDAV'02, Miami Beach, Florida, USA, May 12-14, 2002

[19] Singh K., Nair G., and Schulzrinne H., " Centralized conferencing using SIP," in Proceedings of the 2nd IP-Telephony Workshop (IPTel'2001), New YorkCity, USA, April 2001

[20] Miladinovic I. and Stadler J., "Multiparty Conference Signalling usingthe Session Initiation Protocol (SIP)", In Proceedings of the INC 2002, Plymouth, England, July 2002. pp. 191–198

[21] Rosenberg J. and Schulzrinne H., "Models for multi party conferencing in SIP" , Internet Draft, Internet Engineering Task Force, Nov. 2000. Work in progress.

[22] Intel Corp, Intel Pentium 4 and Intel Xeon Processor Optimization Manual , Order number 248966-05

[23] Schulzrinne H., Rao A., and Lanphier R., "Real time streaming protocol (RTSP)", RFC 2326, Internet Engineering Task Force, Apr. 1998

[24] Braden R. and Zhang L. ,"Resource Reservation Protocol (RSVP)-version1 functional specification", RFC 2205, Internet Engineering Task Force, Sep 1997

[25] Schulzrinne H., Casner S., Frederick R., and Jacobson V., "RTP: a transport protocol for real-time application", RFC 1889, Internet Engineering Task Force, Jan 1996

[26] International Telecommunication Union, "Media stream packetization and synchronization on non-guaranteed quality of service LANs", recommendation H.225.0, Telecommunication Standardization Sector of ITU, Nov. 1996

[27] International Telecommunication Union, "Control protocol for multimedia communication", recommendation H.245, Telecommunication Standardization Sector of ITU, Nov. 1998

[28] Nortel Networks., Inc. A Comparison of H.323v4 and SIP. Tdoc S2-000505. 3GPP2, January 2000

[29] A. Percy. "IP Telephony Inter-Gateway Protocols," Brooktrout Technology, Inc. [Online]. Available : www.brooktrout.com

[30] Willebeek-LeMair M.H. and Shae, Z-Y, "Centralized versus distributed schemes for videoconferencing", Distributed Computing Systems, Proceedings of the Fifth IEEE Computer Society Workshop on Future Trends of , 28-30 Aug. 1995

[31] Loui A., Sun M.T., and Chen T.C., "Coded-domain multipoint video bridging", Circuits and Systems, ISCAS97, Proceedings of 1997 IEEE International Symposium on , Volume: 2 , 9-12 June 1997

[32] Narbutt M. and Murphy L., "Adaptive Anti-jitter Mechanism for Multi-Party Conferencing in a H.323 Multi-Point Control Unit", IEI/IEE Postgraduate Symposium Telecommunications Systems Research, Dublin, Nov. 2001

[33] Ruediger R. Asche. "Compound Win32 Synchronization Objects". Microsoft Developer Network Technology Group. July 21, 1994, [Online]. Available : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndllpro/html/msdn_locktest.asp

[34] "FFmpeg Project" [Online]. Available : http://ffmpeg.sourceforge.net

[35] "oSIP Project" [Online]. Available : http://www.gnu.org/software/osip/osip.html

[36] "eXosip Project" [Online]. Available : http://savannah.nongnu.org/projects/exosip/

[37] "Speex Project" [Online]. Available : http://www.speex.org/

[38] "Callgen323 Project" [Online]. Available :http://callgen.sourceforge.net

[39] Triperm C. and Kittitornkun S., "High Performance Video Conference Multipoint Control Unit with MPEG-4 Codec", ECTI-Con 2005, Pataya, Thailand, May 12-13, 2005. pp. 554-557

[40] Triperm C. and Kittitornkun S., "High Performance MPEG-4 Multpoint Conference Unit", IASTED International Conference on Networks and Communication Systems, Krabi, Thailand, 18-20 April, 2005. pp. 451-456

ภาคผนวก

ภาคผนวก ก.
ไลบรารีต่างๆที่ใช้ในงานวิจัย

ในงานวิจัยนี้ได้ใช้ไลบรารีต่างๆที่เป็นโอเพ่นซอร์สจำนวนมาก ซึ่งในที่นี้จะกล่าวถึงไลบรารีหลักๆที่สำคัญเท่านั้น ดังนี้

ก.1 OpenH323 ไลบรารี

OpenH323 [14] เป็นโปรเจคที่พัฒนาขึ้นสำหรับโปรโตคอลสแต็ก H.323 ซึ่ง OpenH323 นี้เป็น โอเพ่นซอร์สที่ใช้ในการพัฒนาโปรแกรมส่วนตัวหรือโปรแกรมเพื่อการค้ามากมาย โปรเจคนี้เริ่มต้นขึ้นในเดือนกันยายน 1998 โดย Equivalence Pty Ltd ซึ่งเป็นบริษัทส่วนตัวที่ตั้งขึ้นในประเทศออสเตรเลีย โดย OpenH323 มีหน้าที่ในการสร้างการเชื่อมต่อ และสิ้นสุดการสื่อสารของ VoIP ภายใน OpenH323 มีโคเดคต่างๆมากมายสำหรับการสื่อสารทั้งภาพและเสียงโดยโคเดคต่างๆเป็น Plug-in ซึ่งเป็นโอเพ่นซอร์สที่มีอยู่ในอินเตอร์เน็ต ไลบรารี OpenH323 ทำงานอยู่บน PWLib อีกทีหนึ่ง

ก.2 PWLib ไลบรารี

PWLib [14] เป็นคลาสที่ใหญ่พอสมควรซึ่งถูกสร้างขึ้นมาหลายปีแล้ว โดยเป็นโปรแกรมซึ่งทำให้แอพพลิเคชันที่เขียนอยู่บนมันสามารถรันได้ทั้งระบบปฏิบัติการไมโครซอฟต์วินโดส์และระบบ Unix X-Windows

ก.3 FFmpeg ไลบรารี

FFmpeg [34] เป็นโปรเจคโอเพ่นซอร์สที่พัฒนาขึ้นสำหรับการบันทึก เปลี่ยนแปลงและแก้ไขวิดีโอและออดิโอสตรีม ภายใน FFmpeg ประกอบด้วย libavcodec ซึ่งเป็นโคเดคชั้นนำทางด้านออดิโอและวิดีโอ FFmpeg ถูกพัฒนาบนระบบปฏิบัติการ Linux แต่มันยังสามารถถูก Compile ได้บนระบบปฏิบัติการอื่นๆรวมทั้งวินโดส์อีกด้วย

ก.4 oSIP ไลบรารี

GNU oSIP [35] เป็น Implementation ของ SIP โปรโตคอล ซึ่งเป็นโอเพ่นซอร์สที่ถูกเขียนขึ้นโดยใช้ภาษา C มาตรฐานเท่านั้น ดังนั้น oSIP จึงไม่เชื่อมโยงกับระบบจึงสามารถพอร์ตไปรันบนระบบปฏิบัติการไหนก็ได้ oSIP นั้นมีคุณสมบัติที่ปลอดภัยในการใช้เธรดดังนั้นจึงเหมาะที่จะนำไปใช้ในโปรแกรมแบบหลายเธรด oSIP มีขนาดเล็กดังนั้นจึงเหมาะที่จะถูกนำไปสร้างเป็น IP soft-phone

และ Embedded SIP software oSIP ไม่ได้ถูกจำกัดอยู่แต่เพียง Endpoint agents เท่านั้นมันยังถูกนำไปสร้างเป็น SIP Proxy ได้อีกด้วย oSIP ไม่ได้พยายามที่จะให้ API ชั้นสูงสำหรับควบคุม SIP Session แต่มันได้ให้ API สำหรับ SIP message parser และการจัดการ SIP transactions เท่านั้น จึงทำให้การพัฒนา SIP Endpoint จะต้องใช้ความรู้ที่ค่อนข้างสูงทีเดียว

ก.5 eXosip ไลบรารี

eXtended osip [36] ไลบรารีหรือเรียกสั้นๆว่า eXosip เป็นโปรเจคโอเพ่นซอร์สที่พัฒนาบน oSIP ไลบรารี ซึ่งมันได้ซ่อนความซับซ้อนของการสร้าง SIP Session สำหรับ Multimedia ของ oSIP เอาไว้ มันจึงเหมาะที่จะใช้สร้าง SIP Endpoints หรือ SIP Conference Server นอกจากนี้มันยังอาจจะมีประโยชน์สำหรับบางแอพพริเคชันที่ต้องการจะสร้าง sessions เช่น Multiplayer games อีกด้วย

ภาคผนวก ข.

การซิงโครไนซ์เธรดบนระบบปฏิบัติการไมโครซอฟต์วินโดส์

ข.1 เธรด

เธรดเป็นหน่วยพื้นฐานของการจัดสรรการใช้ประโยชน์ของซีพียูซึ่งประกอบด้วย

- หมายเลขเธรด (thread ID) เป็นหมายเลขเธรดในโพรเซส
- ตัวนับ เพื่อติดตามให้ทราบคำสั่งต่อไปที่จะเอ็กซิคิวต์
- ชุดของรีจิสเตอร์ เพื่อ เก็บค่าตัวแปรที่ทำงานอยู่
- สแต็ก(stack) เพื่อเก็บประวัติการเอ็กซิคิวต์

เธรดดั้งเดิม (ที่เรียกว่า heavyweigth) ที่มีการควบคุมเพียง 1 เธรดแสดงว่าทำงานได้ 1 งาน แต่เธรดมีหลายเธรด (อาจเรียกว่า multi-thread) จะทำงานได้หลายงานในเวลาเดียวกัน

ข.1.1 ข้อได้เปรียบของ multi-threaded

1. การตอบสนอง : ระบบ multi-threaded ที่ตอบโต้ แอพพลิเคชันจะยินยอมให้โปรแกรมยังคงดำเนินต่อไป

2. การแชร์รีซอร์ส : ข้อได้เปรียบของการโค้ดจะทำให้แอพพลิเคชันสามารถมีกิจกรรมของเธรดได้หลายๆกิจกรรมภายในแอ็ดเดรสเดียวกัน

3. ความประหยัด : เนื่องจากเธรดแชร์รีซอร์สของเธรดที่มันอาศัยอยู่แล้ว ทำให้เกิดการประหยัดในการสร้างเธรด

4. การเอื้อประโยชน์ของสถาปัตยกรรมมัลติเธรด : ข้อได้เปรียบของ multi-threaded ช่วยเสริมสถาปัตยกรรมมัลติเธรดให้สูงขึ้น

ข.1.2 User และ Kernel Thread

- User Thread จะได้รับการสนับสนุนจาก Kernel และอยู่ในไลบรารีของเธรดในระดับของผู้ใช้
- Kernel Thread ที่ได้รับการสนับสนุนโดยตรงจากระบบปฏิบัติการ Kernel จะสร้าง จัดเวลา และจัดการเธรด ภายในพื้นที่เอง

ข.1.3 โมเดลของ Multi-threading

- โมเดล Many-to-One เป็นโมเดลที่ใช้ kernel thread 1 หน่วยกับ User thread หลายหน่วย
- โมเดล One-to-One เป็นโมเดลที่แต่ละ User thread จะจับคู่กับ Kernel thread ใน 1 ลักษณะ
- โมเดล Many-to-Many เป็นโมเดลที่อาจมีจำนวน User thread มากกว่าหรือเท่ากับจำนวน Kernel thread ก็ได้

ข.1.4 การยกเลิกเธรด

เธรดที่ถูกยกเลิกอาจจะเรียกว่า target thread ซึ่งการยกเลิกอาจมี 2 วิธี

- การยกเลิกแบบ Asynchronous เป็นการยกเลิกที่เธรดอื่นสั่งให้ target thread หยุดทันที
- การยกเลิกแบบ Deferred เป็นการยกเลิกโดย Target thread จะใช้เวลาตรวจสอบว่าตนเองจะถูกยกเลิกหรือไม่ ถ้าต้องยกเลิกก็จะยกเลิกด้วยตัวเอง

ข.2 การซิงโครไนซ์เธรด (Thread Synchronization)

การซิงโครไนซ์เธรด [32] หมายถึงการทำงานของเธรด 2 เธรดที่ต้องมีการเกี่ยวข้องกัน อาจเป็นเพราะการใช้รีซอร์สร่วมกัน หรืออาจจะเป็นการรอการเอ็กซิคิวต์เธรดหลังจากที่เธรดอื่นๆเอ็กซิคิวต์ไปแล้วเป็นต้นทำให้ต้องมีการรอจังหวะที่เหมาะสมเพื่อให้การทำงานนั้นถูกต้อง

ข.2.1 ซิงโครไนซ์ออบเจ็กในระบบ win32

- Critical Sections

Critical Section คือส่วนหนึ่งของโค้ดที่สามารถเข้าถึงทรัพยากรที่ใช้ร่วมกันได้ โดยทรัพยากรนี้อาจจะเป็นตำแหน่งของหน่วยความจำ, Data Structure หรือทรัพยากรใดๆซึ่งสามารถเข้าถึงได้เพียงเธรดเดียวที่เวลาใดเวลาหนึ่ง โดยที่มีเพียงเธรดเดียวเท่านั้นที่สามารถอยู่ภายใน Critical Section ที่เวลาใดๆ เธรดอื่นๆจะถูกบล็อกจากการเข้าสู่ Critical Section โดยเธรดเหล่านี้จะต้องคอยให้เธรดที่อยู่ใน Critical Section ออกไปเสียก่อน Critical Section ถูกใช้ในการซิงโครไนซ์เธรดที่อยู่ในโพรเซสเดียวกันเท่านั้นไม่สามารถซิงโครไนซ์ข้ามโพรเซสได้ Critical Section ไม่ใช่ kernel objects ดังนั้นมันจึงเป็นเหตุผลที่ใช้ในการซิงโครไนซ์เธรดที่อยู่ในโพรเซสเดียวกัน

- Mutex

Mutex เป็น kernel object ที่ยอมให้เธรดใดๆในระบบครอบครองทรัพยากรแบบ Mutually exclusive โดยมีเพียง 1 เธรดเท่านั้นที่เวลาใดๆสามารถครอบครอง Mutex object Mutex สามารถถูกใช้ข้ามโพรเซสไม่เหมือนกับ Critical Section แต่ข้อเสียของการใช้ Mutex ก็คือมันใช้เวลาในการล็อคและปลดล็อคนานกว่า Critical Section ถึง 100 เท่า

- Semaphores

Semaphores เป็น kernel object ที่มีจำนวนเป็นตัวเลขของเธรดที่ยอมให้เข้าถึงทรัพยากรร่วมกัน โดยพิจารณา Semaphores เป็นตัวแปร Integer ซึ่งมี Operation สองอย่างถูกทำบนมัน ก็คือ wait และ signal Semaphores สามารถถูกใช้ในการทำ mutual exclusion ระหว่างโพรเซสได้ Semaphores ส่วนใหญ่ถูกใช้เพื่อแก้ปัญหา producer/consumer problems ซึ่งการอ่านและเขียนสามารถถูกทำที่เวลาเดียวกัน

- Events

Event object เป็น kernel object ที่อยู่ในสถานะ nonsignaled จนกว่าเงื่อนไขหนึ่งจะบรรลุ หรือกล่าวได้อีกอย่างหนึ่งว่าเธรดใดๆจะไม่สามารถผ่าน event object ที่อยู่ในสถานะ nonsignaled ได้จนกว่าจะมีเธรดอื่นๆทำการ Signal event object นี้ โปรแกรมเมอร์มีหน้าที่ในการเซต event object ให้อยู่ในสถานะ signaled หรือ nonsignaled ไม่เหมือนกับ mutex หรือ semaphore ที่ระบบปฏิบัติการมีหน้าที่สั่งให้อยู่ในสถานะ signaled หรือ nonsignaled แนวคิดของ event object มีความคล้ายคลึงกับตัวแปรแบบเงื่อนไขซึ่งทำให้โปรแกรมเมอร์สามารถที่จะมีความยืดหยุ่นอย่างมากในการสร้าง Synchronization Object

ข.2.2 ปัญหาการซิงโครไนซ์เธรด

ปัญหาที่เกิดขึ้นกับระบบปฏิบัติการเป็นปัญหาที่น่าสนใจและถกเถียงกันอย่างกว้างขวางตลอดจนมีการวิเคราะห์โดยการใช้วิธีการด้านซิงโครไนซ์ ในที่นี้จะนำมากล่าว 3 ปัญหาดังนี้

- The Dining Philosophers Problem

ปัญหาแรกนี้ไดจก์สตรา ได้เสนอไว้โดยสมมุติว่ามีนักปราชญ์ 5 ท่านนั่งรอบโต๊ะกลมเพื่อทานอาหารที่วางตรงหน้า ในการทานอาหารนักปราชญ์แต่ละท่านต้องใช้ตะเกียบทั้ง 2 ข้างที่วางอยู่ซ้ายขวามือของนักปราชญ์ท่านนั้น เมื่อทานเสร็จแล้วในรอบนั้นให้วางตะเกียบลงเพื่อเปิดโอกาสให้นักปราชญ์ท่านอื่นได้ทานอาหารบ้าง จะเกิด Deadlock เมื่อปราชญ์ทั้ง 5 หยิบตะเกียบคนละ 1 ข้างพร้อมกันสมมุติว่าทุกท่านหยิบตะเกียบข้างซ้ายพร้อมกัน ทุกท่านจะต้องรอตะเกียบข้างขวาว่างจึงทานอาหารได้และถ้ามีการสลับกันทานอาหาร โดยนักปราชญ์ที่โชคร้ายถูกนักปราชญ์ซ้ายขวามาสลับกันทานอาหารทำให้นักปราชญ์ท่านกลางไม่มีโอกาสทานอาหารเรียกปรากฏการณ์เช่นนี้ว่า "Starvation"

- The Sleeping Barber Problem

ปัญหาที่ 3 เป็นปัญหาที่เกิดขึ้นกับร้านตัดผมที่มีช่างตัดผมหนึ่งคนเก้าอี้ 1 ตัวและเก้าอี้ที่นั่งรอคิวอีกจำนวนหนึ่ง ถ้าไม่มีลูกค้า ช่างตัดผมจะนอนรอคิว เมื่อลูกค้าเข้ามาภายในร้านจะปลุกช่างตัดผม ถ้ามีลูกค้าเข้ามาเพิ่มก็จะนั่งรอ ปัญหาคือการเขียนโปรแกรมย่อยหรือโปรซีเดอร์ต่างๆที่เรียกเข้ามา การแก้ปัญหาจะใช้ 3 Semaphore ดังนี้

- Customers ที่นับจำนวนลูกค้า (ไม่รวมลูกค้าบนเก้าอี้ตัดผม)
- Barbers ที่นับจำนวนช่างตัดผม (0 หรือ 1) ที่ไม่ได้ตัดผม(นอนรอลูกค้า)
- Mutex ที่ใช้สำหรับ Mutual Exclusion

- The Readers-Writers Problem

Readers-Writers Problem เป็นปัญหาเกี่ยวกับระบบคอมพิวเตอร์ที่มีกลุ่มข้อมูลที่ใช้งานร่วมกัน ซึ่งมีเธรด 2 ประเภท คือ เธรดที่อ่านข้อมูลร่วมกัน และเธรดที่บันทึกหรือเขียนข้อมูล โดย Readers-Writers Problem เป็นปัญหาส่วนใหญ่ที่พบในงานวิจัยนี้ ดังนั้นเราจึงสร้างออบเจ็กต์ขึ้นมาบนระบบ win32 สำหรับใช้แก้ปัญหานี้ ดัง Source Code ในรูป ข.1 โดยเมื่อต้องการจะอ่านก็ให้ใช้คำสั่ง Object.LockReader() ถ้าต้องการหยุดการอ่าน ก็ใช้คำสั่ง Object.UnlockReader() โดยการอ่านนี้จะอ่านพร้อมกันได้หลายเธรด ส่วนการเขียนนั้นให้ใช้คำสั่ง Object.LockWriter() และเมื่อจะหยุดการเขียนก็ใช้คำสั่ง Object.UnlockWriter() โดยสำหรับการเขียนนี้จะเขียนได้พร้อมกันเพียงเธรดเดียวเท่านั้น

```
class ReadWriteSynObj
{
public:
        ReadWriteSynObj();
        ~ReadWriteSynObj();
        void LockReader();
        void UnlockReader();
        void LockWriter();
        void UnlockWriter();
protected:
        CRITICAL_SECTION cri_writer;
        CRITICAL_SECTION cri_reader;
        int num_readers;
        HANDLE event_rw;
```

```
};
void ReadWriteSynObj::LockReader()
{
        // We make sure that nobody is writing
        ::EnterCriticalSection(&cri_writer);
        // We need a critical section to update the number of readers
        ::EnterCriticalSection(&cri_reader);
                num_readers++;
                if ( num_readers == 1 )
                {
                        ResetEvent(event_rw);
                        //Lock
                }

        ::LeaveCriticalSection(&cri_reader);
        ::LeaveCriticalSection(&cri_writer);
}
void ReadWriteSynObj::UnlockReader()
{
        // We need a critical section to update the number of readers
        ::EnterCriticalSection(&cri_reader);
        {
                num_readers--;
                if ( !num_readers)
                {
                        // We indicate that there are no more readers
                        SetEvent(event_rw);
                        //Unlock
                }
        }
        ::LeaveCriticalSection(&cri_reader);
}
```

```
void ReadWriteSynObj::LockWriter()
{
        //Only one writer at a time
        ::EnterCriticalSection(&cri_writer);
        //Wait for all readers to leave
        WaitForSingleObject(event_rw,INFINITE);
}
void ReadWriteSynObj::UnlockWriter()
{
        // Let other readers and writers in
        ::LeaveCriticalSection(&cri_writer);
}
ReadWriteSynObj::ReadWriteSynObj()
:num_readers(0)
{
        ::InitializeCriticalSection(&cri_writer);
        ::InitializeCriticalSection(&cri_reader);
        event_rw = ::CreateEvent(NULL, true, true, NULL);
}
ReadWriteSynObj::~ReadWriteSynObj()
{
        ::CloseHandle(event_rw);
        ::DeleteCriticalSection(&cri_reader);
        ::DeleteCriticalSection(&cri_writer);
}
```

รูปที่ ข.1 Source Code ของ Readers-Writers Object บนระบบ win32

ภาคผนวก ค.

เทคโนโลยี SIMD / MMX / SSE / SSE2

ในปัจจุบันโปรเซสเซอร์รุ่นใหม่ๆได้ออกมาพร้อมกับส่วนเพิ่มเติมของชุดคำสั่งของพวกมัน ส่วนเพิ่มเติมเหล่านี้โดยทั่วไปคือ ชุดคำสั่ง SIMD ซึ่ง SIMD นี้ย่อมาจาก Single Instruction Multiple Data ซึ่งหมายความว่าหนึ่งคำสั่ง เช่น add จะทำงานบนข้อมูลจำนวนหนึ่งแบบขนาน ยกตัวอย่างเช่น คำสั่ง add ของ SIMD 1 คำสั่งจะ add ข้อมูล 16 bit จำนวน 8 ชุดแบบขนาน ซึ่งแน่นอนว่าสิ่งนี้จะเพิ่มความเร็วของการทำงานได้อย่างมหาศาลซึ่งก็คือเหตุผลว่าทำไมชุดคำสั่งเหล่านี้จึงถูกสร้างขึ้น

วิธีที่จะใช้ชุดคำสั่งเหล่านี้อย่างโดยตรงที่สุดคือใช้ชุดคำสั่งภาษา inline assembly ในซอร์สโค้ดของเรา อย่างไรก็ตามวิธีนี้ก็เป็นวิธีการที่ไม่ยืดหยุ่นและใช้เวลามากเกินไป อินเทล [22] จึงได้จัดเตรียมวิธีการที่ง่ายกว่าโดยใช้ชุดคำสั่ง API เพิ่มเติมเรียกว่า Intrinsics โดย Intrinsics นี้เป็นภาษา C แบบพิเศษรูปแบบหนึ่ง

ค.1 ประโยชน์ของการใช้ intrinsics

ประโยชน์ของการใช้ intrinsics ก็คือเราสามารถที่จะใช้คุณลักษณะพิเศษที่ไม่มีในภาษาอื่นๆดังนี้

ค.1.1 รีจิสเตอร์ใหม่จำนวนหนึ่ง รีจิสเตอร์เหล่านี้สามารถรองรับข้อมูลได้สูงถึง 128 bit ในการประมวลผลแบบ SIMD โดยรีจิสเตอร์เหล่านี้จะแตกต่างกันไปในแต่ละโปรเซสเซอร์

Data type	MMX-Register							
1x 64 bit (raw) integer	Bits 63 .. 0							
2x 32 bit integers	32 bit word 1				16 bit word 0			
4x 16 bit integers	16 bit word 3		16 bit word 2		16 bit word 1		16 bit word 0	
8x 8 bit integers	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0

รูปที่ ค.1 MMX รีจิสเตอร์ในเทคโนโลยี MMX

<u>รีจิสเตอร์ในเทคโนโลยี MMX</u>

Intrinsics ในเทคโนโลยี MMX ได้จัดเตรียมรีจิสเตอร์ใหม่จำนวน 8 ตัว(MM0 ถึง MM7) ซึ่งแต่ละอันมีความยาว 64 bits (0 to 63) ซึ่งรีจิสเตอร์เหล่านี้สามารถเก็บข้อมูลต่างๆได้ดังรูปที่ ค.1

<u>รีจิสเตอร์ในเทคโนโลยี Streaming SIMD Extensions (SSE) และ Streaming SIMD Extensions 2 (SSE2)</u>

Intrinsics ในเทคโนโลยี SSE และ SSE2 ได้จัดเตรียมรีจิสเตอร์ใหม่จำนวน 8 ตัว (XMM0 ถึง XMM7) ซึ่งแต่ละอันมีความยาว 128 bits (0 to 127) ดังรูปที่ ค.2 และ ค.3

รีจิสเตอร์ข้อมูลแบบใหม่เหล่านี้ทำให้เราสามารถประมวลผลข้อมูลอย่างขนาน เนื่องจากว่าแต่ละรีจิสเตอร์สามารถบรรจุข้อมูลได้มากกว่า 1 ชุด ซึ่งทำให้โปรเซสเซอร์สามารถประมวลผลข้อมูลได้มากกว่า 1 ชุดพร้อมกัน ความสามารถในการประมวลผลนี้เรียกว่า SIMD เพื่อที่จะสามารถประมวลผลแบบ SIMD ใน C/C++ คอมไพเลอร์ได้ data types แบบใหม่จะต้องถูกนิยามสำหรับรีจิสเตอร์แบบใหม่เหล่านี้

Data type	**XMM-Register**			
1x 128 bit raw integer	Bits 127 .. 0			
4x 32 bit floating point values	32 bit floating point value 3	32 bit floating point value 2	32 bit floating point value 1	32 bit floating point value 0

รูปที่ ค.2 รีจิสเตอร์ในเทคโนโลยี SSE

Data type	**XMMx-Register**															
1x 128 bit raw integer	Bits 127 .. 0															
2x 64 bit integer	64 bit word 1								64 bit word 0							
4x 32 bit integer	32 bit word 3				32 bit word 2				32 bit word 1				32 bit word 0			
8x 16 bit integer	16 bit word 7		16 bit word 6		16 bit word 5		16 bit word 4		16 bit word 3		16 bit word 2		16 bit word 1		16 bit word 0	
16x 8 bit integer	Byte 15	Byte 14	Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
2x 64 bit floating point values	64 bit floating point value 1								64 bit floating point value 0							
4x 32 bit floating point values	32 bit floating point value 3				32 bit floating point value 2				32 bit floating point value 1				32 bit floating point value 0			

รูปที่ ค.3 รีจิสเตอร์ในเทคโนโลยี SSE2

ค.1.2 Data Types แบบใหม่ ซึ่งสามารถที่จะรองรับข้อมูลได้สูงสุดถึง 16 ชุดในหนึ่งรีจิสเตอร์

data type แบบใหม่เหล่านี้แทนรีจิสเตอร์แบบใหม่ซึ่งถูกใช้เป็น operands ในฟังก์ชันใน Intrinsics ซึ่ง data type เหล่านี้แสดงในตาราง ค.1

ตาราง ค.1 Data type ที่ถูกใช้ใน Intrinsics

New data type	MMX technology	Streaming SIMD Extensions	Streaming SIMD Extensions 2 instructions
__m64	Yes	Yes	Yes
__m128	Not available	Yes	Yes
__m128d	Not available	Not available	Yes
__m128i	Not available	Not available	Yes

- __m64 data type ถูกใช้เพื่อแทนเนื้อหาในรีจิสเตอร์แบบ MMX ซึ่งเป็นรีจิสเตอร์ที่ถูกใช้ในเทคโนโลยี MMX Intrinsics โดย __m64 data type นี้สามารถที่จะเก็บค่าขนาด 8 bits จำนวน 8 ชุด, ค่าขนาด 16 bits จำนวน 4 ชุด, ค่าขนาด 32 bits จำนวน 2 ชุด หรือ ค่าขนาด 64 bits จำนวน 1 ชุด
- __m128 data types ถูกใช้เพื่อแทนเนื้อหาในรีจิสเตอร์แบบ SSE และ SSE2 ซึ่งเป็นรีจิสเตอร์ขนาด 128 bit

ค.2 ชุดคำสั่งของแต่ละเทคโนโลยี

ค.2.1 ชุดคำสั่งของเทคโนโลยี MMX

ในปี ค.ศ. 1997 อินเทลได้แนะนำเทคโนโลยี เพนเทียมโปรเซสเซอร์ซึ่งมีเทคโนโลยี MMX หลังจากนั้นเทคโนโลยี เพนเทียม 2, 3 และ 4 จึงได้พัฒนาขึ้น นอกจากนี้โปรเซสเซอร์อื่นๆเช่น AMD Athlon และ Duron ได้เพิ่มชุดคำสั่งเพิ่มเติมเหล่านี้เข้าไปด้วย โดยชุดคำสั่ง MMX สำหรับ Integer แสดงในตารางที่ ค.2

ตารางที่ ค.2 ชุดคำสั่งของเทคโนโลยี MMX

Instruction class	Instruction	Operation
Data movement	movq	64 bit move from/to mmx register
	movd	32 bit move from/to mmx register
Data format conversion / movement	packsswb/dw	Parallel pack signed words into bytes / dwords into words with signed saturation
	packuswb/dw	Parallel pack unsigned words into bytes / dwords into words with unsigned saturation
	punpckhbw/wd/dq	Parallel unpack high 32 bits: bytes to words / words to dwords / dwords to qwords
	punpcklbw/wd/dq	Parallel unpack low 32 bits: bytes to words / words to dwords / dwords to qwords
Arithmetical operations	paddb/w/d	Parallel add for bytes / words / dwords
	psubb/w/d	Parallel subtraction for bytes / words / dwords
	paddsb/w	Parallel saturated add for signed bytes / words
	paddusb/w	Parallel saturated add for unsigned bytes / words
	psubsb/w	Parallel saturated subtraction for signed bytes / words
	psubusb/w	Parallel saturated subtraction for unsigned bytes / words
	pmaddwd	Parallel multiply signed words and add the results
	pmullhw	Parallel multiply signed words and store high 16 bits of results
	pmulllw	Parallel multiply signed words and store low 16 bits of results
	pcmpeqb/w/d	Parallel compare signed bytes / words / dwords for equality
	pcmpgtb/w/d	Parallel compare signed bytes / words / dwords for "greater than"

ตารางที่ ค.2 (ต่อ)

Instruction class	Instruction	Operation
Logical operations	pand	Parallel and operation on all 64 bits
	pandn	Parallel and not operation on all 64 bits
	por	Parallel or operation on all 64 bits
	pxor	Parallel xor operation on all 64 bits
Shift operations	psllw/d/q	Parallel shift logical left words / dwords / qwords
	psraw/d	Parallel shift right signed words / dwords
	psrlw/d/q	Parallel shift right unsigned words / dwords / qwords
Enable FPU instructions	emms	Resets the FPU register states to enable FPU instructions

ค.2.2 ชุดคำสั่งของเทคโนโลยี SSE

เพนเทียม 3 โปรเซสเซอร์ได้ถูกแนะนำในปี ค.ศ. 1999 ซึ่งเป็นโปรเซสเซอร์แรกที่มาพร้อมกับชุดคำสั่งเพิ่มเติมของเทคโนโลยี SSE แต่อย่างไรก็ตามในขณะนั้น AMD ได้เพิ่มชุดคำสั่งที่คล้ายกันเรียกว่า 3DNow! เข้าไปใน AMD K6 CPU ของพวกเขาด้วย

ชุดคำสั่ง SSE ได้พัฒนาขึ้นมาจาก MMX 3 ข้อด้วยกันคือ

- เพิ่มเติมชุดคำสั่งของ MMX เช่น min/max
- มีคำสั่ง Prefetch และ write-through เพื่อเพิ่มความสามารถในการเคลื่อนย้ายข้อมูลจากหรือไปสู่ L2/L3 caches และ main memory
- มี XMM รีจิสเตอร์แบบใหม่ขนาด 128 bits และมีชุดคำสั่งสำหรับ 32 bit floating point (single precision)

ชุดคำสั่งสำหรับเทคโนโลยี SSE ได้รวมชุดคำสั่งสำหรับเทคโนโลยี MMX เข้าไปด้วย แต่อย่างไรก็ตามในตารางที่ ค.3 แสดงเฉพาะชุดคำสั่งที่เพิ่มเข้ามาในเทคโนโลยี SSE เท่านั้น

ตารางที่ ค.3 (ต่อ)

Instruction class	Instruction	Operation
XMM arithmetical operations	rcpss	Approximate reciprocal single precision floating point value in lower 32 bits
	sqrtps	Parallel square root single precision floating point values
	sqrtss	Square root single precision floating point value in lower 32 bits
	rsqrtps	Parallel approximate reciprocal square root single precision floating point values
	rsqrtss	Approximate reciprocal square root single precision floating point value in lower 32 bits
	minps	Parallel minimum of single precision floating point values
	minss	Minimum of precision floating point value in lower 32 bits
	maxps	Parallel maximum of single precision floating point values
	maxss	Maximum of precision floating point value in lower 32 bits
	cmpps	Parallel ompare single precision floating point values and return masks as result
	cmpss	Compare single precision floating point value in lower 32 bits and return mask as result
	comiss	Compare single precision floating point value in lower 32 bits and return result in flag register
	ucomiss	Compare unordered single precision floating point value in lower 32 bits and return result in flag register
XMM Logical operations	andps	Parallel and operation on all 128 bits
	andnps	Parallel and not operation on all 128 bits
	orps	Parallel or operation on all 128 bits
	xorps	Parallel xor operation on all 128 bits

ตารางที่ ค.4 (ต่อ)

Instruction class	Instruction	Operation
Data format conversion / movement	pmovmskpd	Extract upper bits of all qwords and store in a general register
	pshuflw	Shuffle words in lower 64 bits (XMM only)
	pshufhw	Shuffle words in high 64 bits (XMM only)
	pshufpd	Shuffle dwords (XMM only)
	unpckhpd	Parallel unpack and interleave high qwords (XMM only)
	unpcklpd	Parallel unpack and interleave low qwords (XMM only)
	cvtpi2pd	Parallel convert integer dwords to double precision floating point values (XMM only)
	punpckhqdq	Parallel unpack high qwords
	punpcklqdq	Parallel unpack lower qwords
	cvtpi2sd	Convert integer dword to double precision floating point value (XMM only)
	cvtpd2si	Parallel convert double precision floating point values to integer qwords (XMM only)
	cvtsd2si	Convert double precision floating point value to integer qword (XMM only)
	cvtps2pd	Parallel convert single precision floating point values to double precision floating point values
	cvtpd2ps	Parallel convert double precision floating point values to single precision floating point values
	cvtss2sd	Convert single precision floating point value to double precision floating point value
	cvtsd2ss	Convert double precision floating point value to single precision floating point value
	cvtps2dq	Parallel convert single precision floating point values to signed dword integers

ตารางที่ ค.4 (ต่อ)

Instruction class	Instruction	Operation
Data format conversion / movement	cvtdq2ps	Parallel convert signed dword integers to single precision floating point values
	movq2dq	Move MMX register value to XMM register
	movdq2q	Move lower 64 bit of XMM register to MMX register
Integer arithmetical operations	paddq	Parallel add for 64 bit integers
	psubq	Parallel subtraction for 64 bit integers
	pmuludq	Parallel multiply 32 bit unsigned integers and return 64 bit result
XMM arithmetical operations	addpd	Parallel add double precision floating point values
	addsd	Add double precision floating point value in lower 64 bits
	subpd	Parallel subtract double precision floating point values
	subsd	Subtract double precision floating point value in lower 64 bits
	mulpd	Parallel multiply double precision floating point values
	mulsd	Multiply double precision floating point value in lower 64 bits
	divpd	Parallel divide double precision floating point values
	divsd	Divide double precision floating point value in lower 64 bits
	rcppd	Parallel approximate reciprocal double precision floating point values
	rcpsd	Approximate reciprocal double precision floating point value in lower 64 bits
	sqrtpd	Parallel square root double precision floating point values
	sqrtsd	Square root double precision floating point value in lower 64 bits
	rsqrtpd	Parallel approximate reciprocal square root double precision floating point values

ตารางที่ ค.4 (ต่อ)

Instruction class	Instruction	Operation
XMM arithmetical operations	rsqrtsd	Approximate reciprocal square root double precision floating point value in lower 64 bits
	minpd	Parallel minimum of double precision floating point values
	minsd	Minimum of precision floating point value in lower 64 bits
	maxpd	Parallel maximum of double precision floating point values
	maxsd	Maximum of precision floating point value in lower 64 bits
	cmpps	Compare double precision floating point values and return masks as result
	cmpsd	Compare double precision floating point value in lower 64 bits and return mask as result
	comisd	Compare double precision floating point value in lower 64 bits and return result in flag register
	ucomisd	Compare unordered double precision floating point value in lower 64 bits and return result in flag register
XMM logical operations	andpd	Parallel and operation on all 128 bits
	andnpd	Parallel and not operation on all 128 bits
	orpd	Parallel or operation on all 128 bits
	xorpd	Parallel xor operation on all 128 bits
XMM shift operations	pslldq	Shift all 128 bits left
	psrldq	Shift all 128 bits rights, filling in zeros

ค.3 MMX, SSE และ SSE2 บน Visual C++ 6

Visual C++ 6 สามารถใช้ MMX, SSE และ SSE2 Intrinsics ได้ โดยจะต้องลง Service Pack 5 และ Processor Pack 5 เท่านั้น

ภาคผนวก ง.
ผลงานวิจัยในระหว่างการศึกษาที่ได้รับการตีพิมพ์เผยแพร่

[1] Triperm C. and Kittitornkun S., "High Performance Video Conference Multipoint Control Unit with MPEG-4 Codec", *ECTI-Con 2005*, Pattaya, Thailand , May 12-13, 2005. pp. 554-557

[2] Triperm C. and Kittitornkun S., "High Performance MPEG-4 Multpoint Conference Unit", *IASTED International Conference on Networks and Communication Systems*, Krabi, Thailand, 18-20 April, 2005. pp. 451-456

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายชัชวาลย์ ไตรเพิ่ม
วันเดือนปีเกิด	24 ธันวาคม พ.ศ. 2521 ที่จังหวัดปทุมธานี
ที่อยู่	5 หมู่ 6 ต.คลองสี่ อ.คลองหลวง จ.ปทุมธานี 12120
	โทร. 01-742-9077
ประวัติการศึกษา	สำเร็จการศึกษาระดับปริญญาตรี หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า จากคณะวิศวกรรมศาสตร์ มหาวิทยาลัยธรรมศาสตร์ ปีการศึกษา 2544