โปรแกรมการสื่อสารด้วยเสียงและวิดีโอผ่านทางเครือข่ายอินเทอร์เน็ตด้วยเทคโนโลยีจาวา

Voice and Video Communication over Internet Protocol Program with Java Technology

โดย

นางสาวดลพร หวังเสริมวงศ์

รหัส 45066053

อาจารย์ที่ปรึกษา

ดร. ธนารัตน์ ชลิดาพงศ์

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2546

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	โปรแกรมการสื่อสารด้วยเสียงและวิดีโอผ่านทางเครือข่ายอินเทอร์เน็ตด้วยเทคโนโลยีจาวา
นักศึกษา	นางสาวดลพร หวังเสริมวงศ์
อาจารย์ที่ปรึกษา	ดร. ธนารัตน์ ชลิดาพงศ์
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2546

บทคัดย่อ

โครงงานนี้เป็นการนำเสนอการออกแบบและพัฒนาระบบการสื่อสารด้วยเสียงและวิดีโอผ่านทางเครือข่ายอินเทอร์เน็ต ที่ทำหน้าที่ส่งและรับข้อมูลเสียงและวิดีโอโดยผ่านทางโปรโตคอลอินเทอร์เน็ต และผ่านโปรโตคอล Real-Time Transfer (RTP) ซึ่งทำให้ผู้ใช้งานสามารถสนทนากันแบบเวลาจริงได้

โปรแกรมของระบบการสื่อสารด้วยเสียงและวิดีโอผ่านทางเครือข่ายอินเทอร์เน็ตนั้นพัฒนาในภาษาจาวา โดยใช้ Java Media Framework (JMF) ซึ่งเป็น Application Program Interface (API) ที่ใช้สำหรับส่งข้อมูลเสียงและวิดีโอแบบเวลาจริงได้ รวมทั้งใช้สำหรับดักจับและแสดงผลข้อมูลเสียงและวิดีโอ และจะใช้โปรแกรม JBuilder X เป็นเครื่องมือในการพัฒนาโปรแกรมประยุกต์

Title	Voice and Video Communication over Internet Protocol Program with Java Technology
Student	Ms. Donraporn Wangsermwong
Advisor	Dr. Thanarat Chalidabhongse
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2003

ABSTRACT

This project presents the design and implementation of voice and video communication over Internet Protocol system. The system uses the Internet Protocol to transfer voice and video stream and Real-Time Transfer Protocol for real time communication.

The system's software is developed in Java programming language as it offers Java Media Framework (JMF), Application Programming Interface (API) used for transferring voice and video through RTP stream as well as capturing and presenting media stream.

กิตติกรรมประกาศ

โครงการพัฒนาระบบนี้จะไม่ประสบความสำเร็จได้ ถ้าไม่ได้รับคำปรึกษาในหลายๆเรื่อง และ คำแนะนำจาก อาจารย์ ดร. ธนารัตน์ ชลิดาพงศ์ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงการ อีกทั้ง ขอขอบคุณอาจารย์ ธนารัตน์ ที่ช่วยดูแลการทำโครงการ คอยติดตามผลในการทำงาน และ ที่สำคัญ คือ ได้เปิดโอกาสให้ข้าพเจ้าได้ทำโครงการนี้ โครงงานนี้ทำให้ข้าพเจ้าได้รับความรู้เพิ่มขึ้นอย่างมาก และทำให้ข้าพเจ้ารู้สึกรักคุณแม่ รักอาจารย์ และ รักเพื่อนๆมากขึ้นอย่างมาก

ขอขอบคุณคุณแม่มากที่สุด ที่คอยดูแล เป็นกำลังใจให้ ให้ความสนับสนุน และ ให้โอกาสในการศึกษาเป็นอย่างดีมาตลอด

ขอขอบคุณคุณดวงกมล มหาวีระ คุณกุลชาติ คุณนภดล คุณสุขุม คุณอลงกต คุณพูลศักดิ์ ที่ให้ยืมอุปกรณ์ในการทดสอบโปรแกรม ขอบคุณคุณกุลธิดา คุณเอกภูมิ คุณชัชพรรณ ที่คอยให้ช่วยเหลืออย่างดีมาตลอด และ ขอบคุณคุณดวงกมล อุทยานวิทยา มากสำหรับทุกเรื่องคะ

สุดท้ายขอขอบคุณทุกๆคนที่เป็นกำลังใจให้สามารถทำโครงการนี้ได้สำเร็จลุล่วง รวมถึง ขอบคุณผู้ที่เขียนบทความ หรือ ผู้ที่ถามตอบในกระทู้เกี่ยวกับปัญหาของ JMF ซึ่งทำให้สามารถแก้ไขปัญหาที่เกิดขึ้นในการทำงานได้

ดลพร หวังเสริมวงศ์

ผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญภาพ

หน้า

รูปที่

สารบัญภาพ (ต่อ)

หน้า

รูปที่

สารบัญภาพ (ต่อ)

หน้า

รูปที่

บทที่ 1
บทนำ

1.1. Voice and Video Communication over IP System

ในปัจจุบันการเติบโตของเครือข่ายโทรศัพท์มีอัตราน้อยลง ในขณะที่การเติบโตของเครือข่ายข้อมูลมีอัตราการเติบโตเพิ่มมากขึ้นอย่างรวดเร็ว รูปแบบการให้บริการการสนทนาจึงเริ่มเปลี่ยนแปลงไป จากเดิมที่การสนทนาส่วนใหญ่นั้นจะผ่านทางเครือข่ายโทรศัพท์ PSTN ซึ่งเป็นแบบ Circuit Switching ซึ่งจะจองช่องสัญญาณตลอดเวลาระหว่างการสนทนาแม้ว่าในระหว่างการสนทนานั้นจะมีบางช่วงที่ไม่ได้ส่งข้อมูลใดๆเลยก็ตาม ทำให้ผู้อื่นไม่สามารถใช้งานช่องสัญญาณนั้นได้ แต่การให้บริการในการสนทนาแบบใหม่นั้น จะเป็นการรับส่งข้อมูลผ่านทางเครือข่ายอินเทอร์เน็ตแบบ Packet Switching โดยวงจรจะถูกเชื่อมต่อก็ต่อเมื่อมีการส่งข้อมูลเท่านั้น ช่วงเวลาที่ไม่มีการสนทนาผู้อื่นก็สามารถใช้ช่องสัญญาณได้

รวมทั้งการสนทนาผ่านทางเครือข่ายโทรศัพท์แบบเดิมนั้น ค่าใช้จ่ายในการสนทนานั้นจะคิดตามระยะห่างระหว่างคู่สนทนาทำให้ผู้ใช้งานมีค่าใช้จ่ายในการสื่อสารเป็นจำนวนมาก แต่การสื่อสารผ่านทางเครือข่ายอินเทอร์เน็ต ค่าใช้จ่ายที่ต้องเสียนั้นเป็นเพียงค่าใช้จ่ายในการเชื่อมต่อเข้าไป ที่เครื่องเซิร์ฟเวอร์ของ ISP เท่านั้น ดังนั้นในการติดต่อสื่อสารแบบผ่านทางเครือข่ายอินเทอร์เน็ต แม้ว่าคู่สนทนาจะอยู่ห่างไกลกันมากขึ้น แต่ค่าใช้จ่ายที่ต้องเสียก็ไม่ได้มากขึ้นตามระยะทาง

1.2. ลักษณะของโครงงาน

โครงงานนี้เป็นการพัฒนาโปรแกรมที่ใช้ในการสื่อสารด้วยเสียงและวิดีโอโดยผ่านทางเครือข่ายอินเทอร์เน็ต ซึ่งผู้ใช้สามารถสนทนาโดยเห็นหน้ากันได้แบบเวลาจริง การเขียนโปรแกรมในการส่งเสียงและวิดีโอผ่านทางโปรโตคอลอินเทอร์เน็ตแบบเวลาจริง โดยใช้ภาษา Java ในการพัฒนา และใช้ Java Media Framework API เพื่อให้สามารถเขียนโปรแกรม java ที่สามารถทำงานเกี่ยวกับข้อมูล media ได้

1.3. วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาเทคโนโลยีที่ใช้ในการติดต่อสื่อสารด้วยเสียงและวิดีโอผ่านทางอินเทอร์เน็ต โปรโตคอล
2. เพื่อศึกษาการเขียนโปรแกรม java เพื่อส่งและรับข้อมูลเสียงและวิดีโอผ่านทางอินเทอร์เน็ต โปรโตคอล
3. เพื่อศึกษาการใช้งาน Java Media Framework API
4. เพื่อพัฒนาระบบ video conference over IP เพื่อให้บริการผู้ใช้งานให้สามารถติดต่อสื่อสารกันโดยเห็นหน้าผ่านทางเครือข่ายอินเทอร์เน็ต โดยไม่ต้องเสียค่าใช้จ่ายในการติดต่อสื่อสาร

1.4. ขอบเขตของโครงการ

1. พัฒนาโปรแกรมซึ่งสามารถ
   - รับภาพ และ เสียงจากอุปกรณ์ที่ใช้ในการรับข้อมูล คือ กล้อง web cam และไมโครโฟน และ สามารถแสดงผลมูลที่ได้ออกมา
   - บันทึกภาพ, เสียง และ วิดีโอ ของข้อมูลที่ได้จากอุปกรณ์ที่ใช้ในการรับข้อมูล
   - รับส่งเสียงและวิดีโอผ่านเครือข่ายอินเทอร์เน็ต โดยจะพัฒนาในลักษณะการใช้งานแบบ Point to Point
   - นำข้อมูลที่ได้รับจากคู่สนทนามาบันทึกเป็นไฟล์ และ แสดงผลออกมาได้
2. พัฒนาโปรแกรมที่ติดต่อระหว่างเครือข่าย H.323 ด้วยกัน

1.5 ประโยชน์ที่ได้รับ

ผู้ใช้งาน

1. สามารถติดต่อสื่อสารกันโดยเห็นหน้าผ่านทางเครือข่ายอินเทอร์เน็ต ไม่ว่าผู้ใช้งานจะอยู่ห่างไกลกันเพียงใดก็ตาม
2. ไม่เสียค่าใช้จ่ายในการติดต่อสื่อสาร

ผู้พัฒนา Application

1. ได้ความรู้เรื่องเรื่องการติดต่อสื่อสารด้วยเสียง และ วิดีโอด้วยมาตรฐาน H.323
2. ได้ความรู้ในการเขียนโปรแกรมด้วยภาษา java เพื่อรับส่งวิดีโอและเสียงผ่านอินเทอร์เน็ตโปรโตคอล
3. ได้รับความรู้ในการใช้งาน Java Media Framework API เพื่อทำงานกับข้อมูล media

1.6 ขั้นตอนการดำเนินงาน

1. ศึกษาการติดต่อสื่อสารด้วยเสียง และ วิดีโอด้วยมาตรฐาน H.323
2. ศึกษาการเขียนโปรแกรมด้วยภาษาจาวา โดยใช้โปรแกรม Jbuilder X
3. ศึกษาการใช้งาน Java Media Framework API เพื่อให้สามารถใช้งาน class และ method ได้
4. ออกแบบโครงสร้างของระบบ และ ออกแบบ user interface
5. พัฒนา application
6. ทดสอบการใช้งาน โดยแบ่งการทดสอบ ออกเป็น
   - ทดสอบการรับส่งข้อมูลเสียงและวิดีโอภายใน LAN
   - ทดสอบการรับส่งข้อมูลเสียงและวิดีโอโดยผ่านทางเครือข่ายอินเทอร์เน็ต

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 การสื่อสารด้วยเสียงและวิดีโอผ่านทางเครือข่ายอินเทอร์เน็ต

Video conference คือ การติดต่อสื่อสารระหว่างคน 2 คนขึ้นไป โดยคู่สนทนาคนหนึ่งสามารถมองเห็น และ ได้ยินเสียงของคู่สนทนาอีกคนหนึ่งได้ในเวลาเดียวกัน นอกจากนี้คู่สนทนายังสามารถที่จะใช้ application ร่วมกันได้ด้วย

ระบบ video conference จำเป็นต้องประกอบด้วย อุปกรณ์ที่ใช้ในการรับ และ แสดงภาพ และ เสียง คือ หน้าจอ กล้องวิดีโอ ไมโครโฟน และ ลำโพง รวมถึงต้องมีวิธีในการส่งข่าวสารระหว่างเครื่องด้วย ด้วยเทคโนโลยีที่ก้าวหน้าทำให้มีระบบที่จะบีบอัดวิดีโอซึ่งสามารถส่งข้อมูลวิดีโอผ่านทางเครือข่ายอินเทอร์เน็ต หรือ เครือข่ายโทรศัพท์ในปัจจุบันได้ ซึ่งเป็นการลดต้นทุนของระบบ video conference เป็นอย่างมาก

แนวโน้มของการสื่อสารโทรคมนาคม จะเป็นลักษณะการรวมบริการหลายอย่างไว้ในโครงข่ายเดียว ซึ่งสามารถให้บริการได้ทั้งสัญญาณเสียง ข้อมูล ภาพ และ วิดีโอ ภายใต้โครงข่ายแบบแพ็คเกจ โดยการส่งข้อมูลทั้งสัญญาณภาพ เสียง และ วิดีโอเป็นชุดของข้อมูล บน Internet Protocol ซึ่งมีข้อดี คือ ราคาค่าบริการที่ต่ำกว่า เช่น ค่าบริการโทรศัพท์ทางไกล หรือ โดยเฉพาะอย่างยิ่งกับค่าบริการทางไกลระหว่างประเทศ ซึ่งระบบโทรศัพท์แบบไอพีจะเก็บค่าบริการเท่ากับค่ายริการที่โทรศัพท์ธรรมดาโทรในพื้นที่ที่ต่อเข้ากับ gateway รวมกับค่าบริการที่ต้องจ่ายให้กับผู้ให้บริการอินเทอร์เน็ตเท่านั้น อย่างไรก็ตาม การส่งข้อมูลเสียงและวิดีโอผ่านทางเครือข่ายอินเทอร์เน็ตยังมีข้อบกพร่อง เช่น ความเชื่อถือได้ของการส่งข้อมูลเสียงและวิดีโอนั้นมีจำกัด

2.1.1 การใช้งานโดยทั่วไป (Video Development Initiative. 2002)

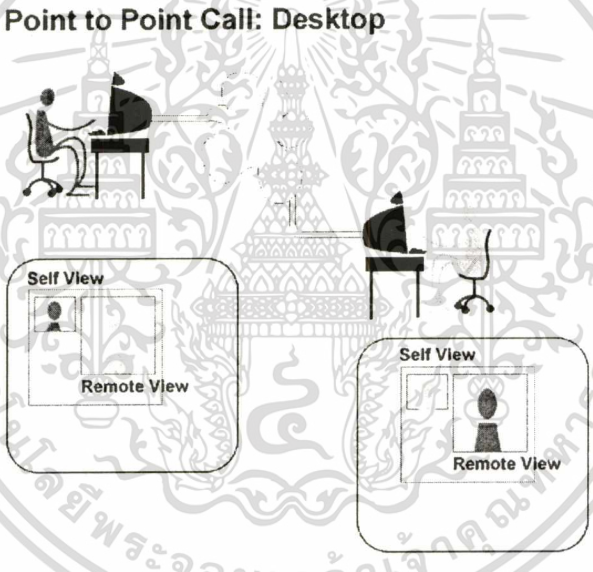
เนื่องจากการใช้งาน video conference นั้น เป็นการติดต่อสื่อสารที่ผู้เข้าร่วมในการประชุมสามารถเห็นหน้า และ ได้ยินเสียงของคู่สนทนาได้ โดยที่ผู้เข้าร่วมการประชุมไม่จำเป็นต้องมา

พบปะกันจริงๆ ซึ่งทำให้สามารถลดค่าใช้จ่ายในการเดินทาง รวมทั้งไม่เสียเวลาในการเดินทาง และทำให้ผู้เข้าร่วมในการประชุมได้รับความสะดวกมากขึ้นด้วย ด้วยสาเหตุนี้ทำให้ video conference เป็นที่นิยมอย่างมาก

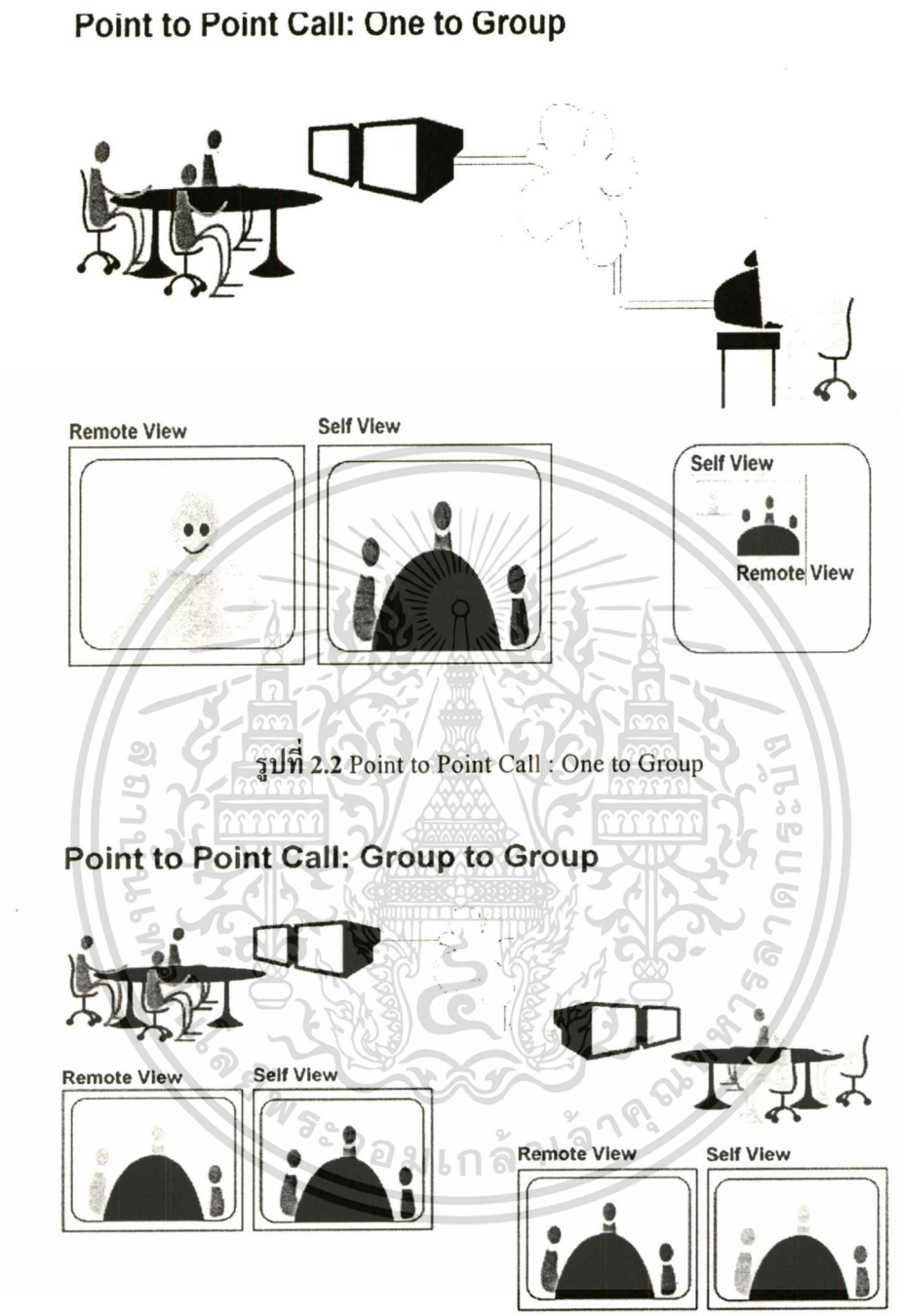
การใช้งาน video conference สำหรับการประชุมนั้นมีได้หลายรูปแบบ ดังนี้

Point to Point

Point to Point เป็นรูปแบบการประชุมที่ง่ายที่สุด คือ เป็นการติดต่อระหว่างเครื่องคอมพิวเตอร์ใน 2 ตำแหน่ง ดังรูปที่ 2.1 คู่สนทนาทั้ง 2 ฝั่งเป็นคนเพียงหนึ่งคน, รูปที่ 2.2 คู่สนทนาฝั่งหนึ่งเป็นคนหนึ่งคน แต่อีกฝั่งเป็นกลุ่มคน และ รูปที่ 2.3 คู่สนทนาทั้ง 2 ฝั่งเป็นกลุ่มคน การติดต่อสื่อสารจะเกิดขึ้นได้ก็ต่อเมื่อ ทำการใส่หมายเลข IP



รูปที่ 2.1 Point to Point call : Desktop

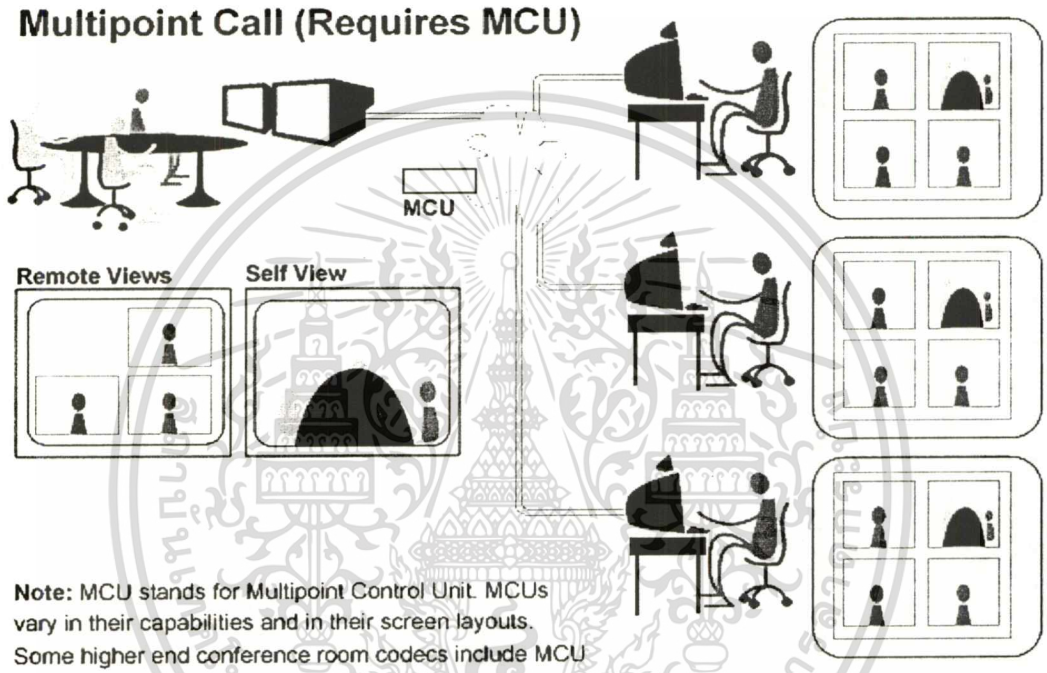


รูปที่ 2.2 Point to Point Call : One to Group

รูปที่ 2.3 Point to Point Call : Group to Group

Point to Multipoint

ในกรณีนี้ ผู้ส่งจำเป็นต้องส่งไปให้ผู้รับที่อยู่ในสถานที่ที่ต่างกันมากกว่า 1 ตำแหน่ง การจะติดต่อสื่อสารแบบนี้ได้นั้นจำเป็นต้องมี Multiple Conference Unit (MCU) เพื่อใช้ในการเชื่อมต่อระบบ video conference ตั้งแต่ 3 ตำแหน่งขึ้นไปเข้าด้วยกัน, จัดการกระจายเสียงและวิดีโอที่มาจากตำแหน่งหนึ่งไปยังตำแหน่งอื่นที่เหลือในกลุ่ม ดังรูปที่ 2.4



รูปที่ 2.4 Multipoint Call

Broadcast

Broadcast เป็นลักษณะของการประชุมในทิศทางเดียวคล้ายกับการกระจายข้อมูลผ่านทางโทรทัศน์ โดยคอมพิวเตอร์ที่ตำแหน่งหนึ่งจะส่งข้อมูลไปยังเครื่องคอมพิวเตอร์ในตำแหน่งที่เหลือทั้งหมดที่เชื่อมต่อกัน ตัวอย่างของ Broadcast อย่างเช่น NASA TV เป็นต้น โดยเราสามารถติดต่อไปยัง NASA TV โดยติดต่อไปที่ IP Address:139.88.27.43 or IP 129.186.112.242

2.1.2 การประยุกต์ใช้

การประยุกต์ใช้การสื่อสารด้วยเสียงและวิดีโอบนเครือข่ายโปรโตคอลอินเทอร์เน็ตนั้นมีหลายประเภท เช่น

- **TeleHealth**

TeleHealth เป็นการใช้ข้อมูลทางอิเลคทรอนิคส์ และ เทคโนโลยีในการสื่อสารในการรองรับเรื่องการดูแลสุขภาพในกรณีที่ระยะทางในการไปสถานพยาบาลมีระยะทางห่างไกล ทำให้คนไข้สามารถที่จะติดต่อกับผู้ที่มีความเชี่ยวชาญด้านสุขภาพได้

- **Telecommuting**

บริษัทต่างๆไม่ว่าจะบริษัทเล็กหรือใหญ่ต่างก็ใช้ Telecommuting กัน โดยอนุญาตให้พนักงานบางส่วนสามารถทำงานที่บ้านได้ ทำให้ค่าใช้จ่ายของบริษัทลดลง เนื่องจากสามารถลดขนาดของบริษัทลงได้ โดยพื้นที่ในบริษัทมีไว้เฉพาะหน่วยงานที่จำเป็นเท่านั้น และการทำงานที่บ้านทำให้พนักงานสามารถรองรับงานที่สำคัญได้อย่างมีประสิทธิภาพยิ่งขึ้น

- **Tele-Education**

Tele-Education ทำให้โอกาสในการเรียนรู้มีเพิ่มมากขึ้นด้วย video conference โดยรวมทั้ง application แบบ remote-teacher/remote-specialist ด้วย การติดต่อสื่อสารกันนั้นจะประกอบไปด้วยระบบ video conference มากกว่า 2 ระบบขึ้นไป นักเรียนสามารถที่จะโต้ตอบกันเอง และสามารถโต้ตอบกับผู้สอนได้ด้วย

- **Judical Application**

ระบบของศาลยุติธรรมนั้นได้ใช้ video conference โดยนำเอาระบบ video conference ไปติดตั้งในคุก และในศาล ส่วนใหญ่จะใช้สำหรับ การขึ้นศาลโดยผ่านทางวิดีโอ คือ นักโทษจะไปยังห้องที่เตรียมไว้สำหรับ video conference ที่อยู่ในคุก และอีกระบบจะอยู่ในศาลโดยจะมีผู้พิพากษา

ผู้ฟ้องร้อง และ ทนาย ซึ่งนักโทษจะเห็นสมาชิกที่อยู่ในศาลทั้งหมด และการตัดสินคดีก็จะดำเนินการเหมือนกับการที่นักโทษมาในศาลจริง หลักการนี้มีประโยชน์มาก เช่น ลดจำนวนของนักโทษที่จะต้องเดินทางจากคุกไปยังศาล ลดความเสี่ยงทางด้านความปลอดภัยในการขนส่งนักโทษ และเป็นการประหยัดค่าใช้จ่าย และ ประหยัดเวลาอีกด้วย

- **Surveillance & Security**

Video conference สามารถใช้เป็นเทคโนโลยีในการสังเกตการณ์ในทิศทางเดียวก็ได้ ในหลายๆมหาวิทยาลัย หรือที่ทำงานนำเอา video conference ไปประยุกต์ใช้กับระบบรักษาความปลอดภัย โดยสามารถติดตั้งอุปกรณ์เพื่อสังเกตการณ์ในบริเวณที่ต้องการเฝ้าระวัง สามารถติดต่อผ่าน LAN และอนุญาตให้ถ่ายวิดีโอได้

2.1.3 ข้อดีของ Video Conference

- ประหยัดค่าใช้จ่ายขององค์กร
- เพิ่มความสะดวกแก่ผู้เข้าร่วมประชุม โดยไม่จะเป็นต้องเสียเวลาในการเดินทางมาประชุม
- สามารถติดต่อสื่อสารโดยเห็นหน้ากัน แม้อยู่ห่างไกลกัน
- เพิ่มประสิทธิภาพของระบบด้านความปลอดภัย
- สามารถโต้ตอบกับผู้เชี่ยวชาญในด้านต่างๆได้ โดยไม่มีปัญหาเรื่องเวลา และ ระยะทาง
- สามารถเรียนรู้ถึงความแตกต่างทางด้านวัฒนธรรม เพราะสามารถติดต่อสื่อสารกับคนที่อยู่ต่างประเภทซึ่งมีวัฒนธรรมแตกต่างกันไป

2.1.4 กลุ่มโปรโตคอลสำหรับสำหรับการส่งเสียงและวิดีโอผ่านทางเครือข่ายอินเทอร์เน็ต

TCP/IP คือ ชุดโปรโตคอลที่ทำหน้าที่ส่งข้อมูล (Data Communication Protocol) โดย โปรโตคอล หมายถึง กฎเกณฑ์ และรูปแบบที่ใช้ในการควบคุมการสื่อสาร โปรโตคอลการสื่อสาร (Communication Protocol) เป็นวิธีกำหนดวิธีการที่อุปกรณ์สื่อสาร 2 ตัวในเครือข่ายจะติดต่อกัน เช่น จะส่งข้อมูลเมื่อไร รับเมื่อไร แก้ไขข้อมูลที่ผิดพลาดได้อย่างไร เป็นต้น โปรโตคอลการสื่อสารจะถูกแบ่งย่อยเป็นลำดับชั้น (layer) ตามหน้าที่การทำงาน

TCP/IP Protocol Stack แบ่งเป็น 5 ชั้น ดังในรูปที่ 2.5

Application Layer	Telnet, FTP, SMTP, e-mail, WWW, SNMP
Transport Layer	TCP, UDP
Network Layer	IP
Data Link Layer	Frame Relay, ATM, SMDS, HDLC, SDLC
Physical Layer	Modem, 56k, T1, E1, RS-232
Medium	

รูปที่ 2.5 TCP/IP Protocol Stack

Internet Protocol ในชั้น network นั้น ทำหน้าที่จัดหาเส้นทางที่จะส่งข้อมูลให้ไปถึงปลายทางที่ถูกต้อง โปรโตคอลอินเทอร์เน็ตนั้นสนับสนุนทั้ง TCP และ UDP ในชั้น transport

TCP และ UDP ทำหน้าที่เดียวกัน คือมีฟังก์ชันที่ส่งข้อมูลเพื่อให้ถึงปลายทางที่ถูกต้องและได้รับอย่างสมบูรณ์ แต่ TCP และ UDP แตกต่างกันตรงที่ TCP ต้องการสถาปนาการเชื่อมต่อก่อนจะส่งข้อมูล เรียกว่า connection-oriented แต่ UDP ไม่ต้องการ เรียกว่า connectionless หรือ datagram ดังนั้นการส่งข้อมูลโดยใช้ UDP จึงใช้เวลาน้อยกว่าแบบ TCP ทำให้ UDP เหมาะกับการส่งข้อมูลเสียงและวิดีโอมากกว่า UDP

สิ่งที่สำคัญสำหรับการส่งเสียงและวิดีโอแบบเวลาจริงนั้น คือ packet ที่ส่งไปนั้นปลายทางจะต้องได้รับด้วยความล่าช้าที่คงที่ ถ้าความล่าช้าในการไปถึงปลายทางไม่คงที่จะทำให้เสียงที่ได้รับขาดตอน และเพื่อลดความต้องการในการประมวลผลสำหรับการส่งเสียงและวิดีโอบนเครือข่ายโปรโตคอลอินเทอร์เน็ต จึงนำ UDP (User Datagram Protocol) มาใช้แทน TCP เพราะว่า UDP มีโอเวอร์เฮดน้อยกว่า TCP ทำให้มันเหมาะสมมากกว่าสำหรับการสื่อสารแบบเวลาจริง

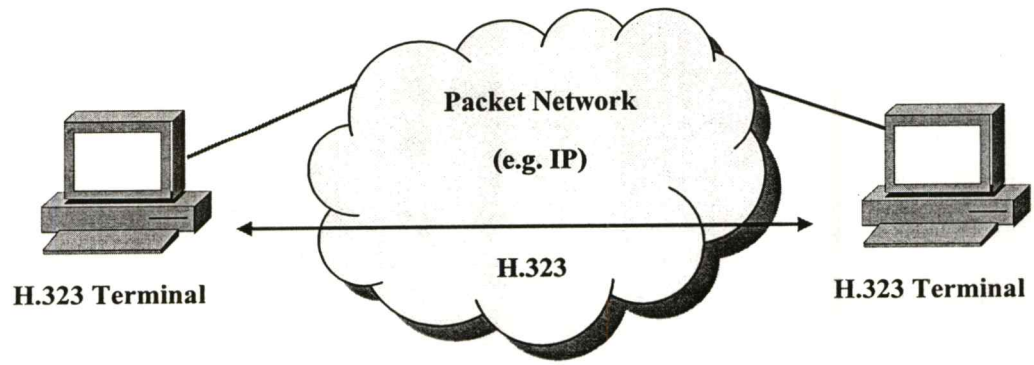
Multimedia Application
RTP, RTCP
UDP/TCP
Network Layer (IPv4, IPv6, IPM)
Data Link Layer
Physical Layer

รูปที่ 2.6 กลุ่มโปรโตคอลสำหรับเสียงและวิดีโอบนโปรโตคอลอินเทอร์เน็ต

กลุ่มของโปรโตคอลสำหรับการส่งเสียงและวิดีโอแบบเวลาจริงบนเครือข่ายโปรโตคอลอินเทอร์เน็ตนั้นเสดงดังรูปที่ 2.6 โดยเหนือ UDP นั้นจะมีโปรโตคอลพิเศษ 2 ตัว คือ RTP (Real Time Transport Protocol) และ RTCP (Real Time Control Protocol) โดย RTP ทำหน้าที่เกี่ยวกับการส่งเสียงและวิดีโอแบบเวลาจริงด้วยความควบคุมของ RTCP และเหนือชั้นของ RTP,RTCP จะเป็นตัวโปรแกรมประยุกต์ที่ทำงานอยู่ ส่วนในชั้น IP นั้น สามารถที่จะเป็น IP version 4 หรือ IP version 6 ซึ่งทั้ง 2 ตัวนี้สามารถที่จะ run บน terminal ที่ต่อเข้ากับอินเทอร์เน็ต หรือ Mobile IP ที่ run บนอุปกรณ์เคลื่อนที่ก็ได้

2.2 มาตรฐาน H.323 (The International Engineering Consortium. 2003)

มาตรฐาน H.323 เป็นเทคโนโลยีที่สำคัญสำหรับการส่งข้อมูล เสียง และวิดีโอแบบเวลาจริงบน packet-based network โดยจะกำหนดอุปกรณ์ โปรโตคอล และวิธีในการติดต่อสื่อสาร



รูปที่ 2.7 H.323 Terminal บน packet network

2.2.1 ส่วนประกอบในมาตรฐาน H.323

ตามมาตรฐาน H.323 ได้แบ่งส่วนประกอบออกเป็น 4 ชนิด ซึ่งถ้านำส่วนประกอบทั้ง 4 ชนิดนี้มาต่อเข้าด้วยกัน จะสามารถให้บริการการสื่อสารมัลติมีเดียแบบ point-to-point และ point-to-multipoint ได้ ส่วนประกอบทั้ง 4 ชนิด ได้แก่

I. Terminals

Terminals ใช้สำหรับการสื่อสารข้อมูลมัลติมีเดียใน 2 ทิศทางแบบเวลาจริง โดย H.323 Terminal อาจเป็นเครื่อง PC หรือ อุปกรณ์ใดๆก็ได้ที่สามารถจะ run H.323 และ โปรแกรมประยุกต์มัลติมีเดีย โดยหน้าที่สำคัญของมันคือ สามารถทำงานร่วมกับ Terminal มัลติมีเดียแบบอื่นๆ ได้

II. Gateways

Gateway ทำหน้าที่เชื่อมต่อระหว่างเครือข่าย H.323 และเครือข่ายประเภทอื่น เช่น gateway ทำการเชื่อมต่อระหว่าง H.323 terminal กับ PSTN (Public Switched Telephone Network) และทำให้เครือข่าย 2 ประเภทที่แตกต่างกันสามารถสื่อสารกันได้ การเชื่อมต่อระหว่าง 2 เครือข่ายที่แตกต่างกันนี้ทำได้โดยแปลงโปรโตคอลที่ใช้สำหรับ call setup และ release และแปลง format ของ media ระหว่างเครือข่ายที่ต่างกัน และส่งผ่านข้อมูลระหว่างเครือข่ายที่เชื่อมต่อกันด้วย gateway แต่ถ้าจะติดต่อสื่อสารระหว่าง 2 เครือข่ายที่เป็นเครือข่าย H.323 เหมือนกัน ก็ไม่จำเป็นที่จะต้องใช้ gateway ในการเชื่อมต่อ 2 เครือข่ายนี้เข้าด้วยกัน

III. Gatekeepers

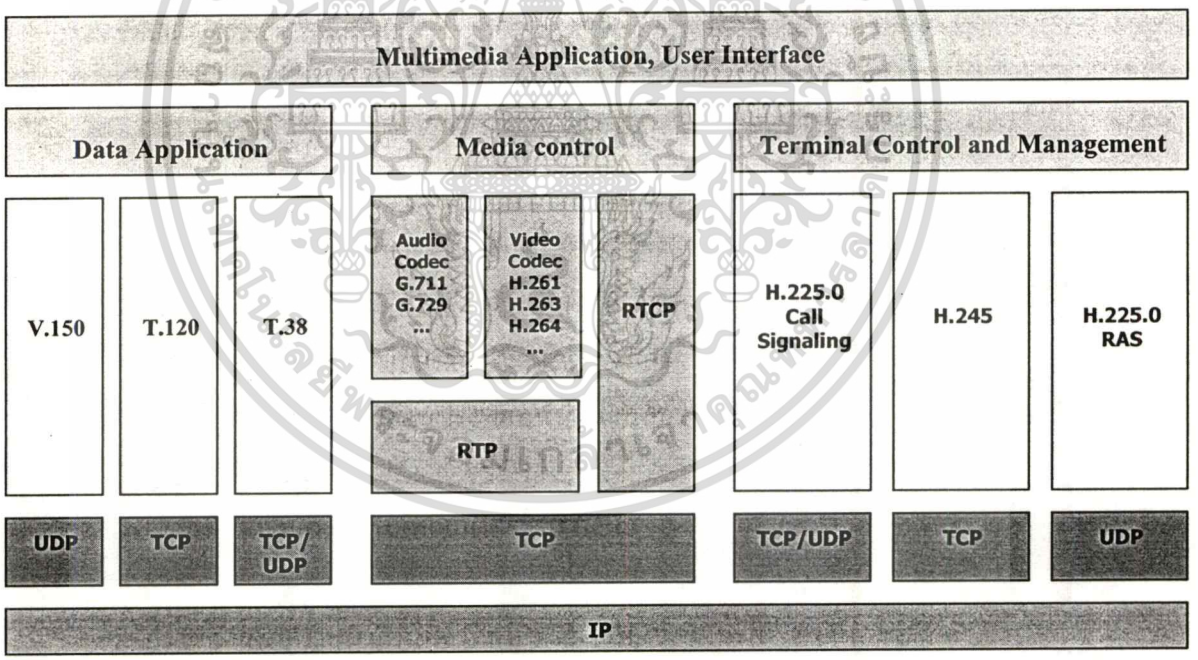
Gatekeeper เป็นอุปกรณ์ที่ไม่จำเป็นต้องมีก็ได้ แต่มันมีบทบาทสำคัญในการให้บริการเกี่ยวกับการจัดการต่างๆ เช่น addressing, authorization, authentication ของ terminals และ gateway, การบริหาร bandwidth, การจัดเก็บค่าบริการ และยังให้บริการในการจัดหาเส้นทางในการโทรศัพท์ด้วย

IV. Multipoint Control Units

MCU มีหน้าที่รองรับการประชุมระหว่าง H.323 Terminals ที่มากกว่า 2 ตัวขึ้นไป ทุกๆ terminals ที่มีส่วนร่วมในการประชุมจะต้องสร้างการติดต่อกับ MCU ซึ่ง MCU จะทำหน้าที่ในการจัดการทรัพยากรในการประชุม, ทำการเจรจาระหว่าง terminal เพื่อที่จะกำหนดวิธีในการเข้ารหัส (CODEC) เสียงและวิดีโอที่ต้องการใช้ และอาจเข้าไปจัดการสายข้อมูล media ด้วย Gateways, Gatekeepers, และ MCUs นั้นตามมาตรฐาน H.323 จะแยกส่วนประกอบทั้ง 3 ส่วนนี้ออกจากกันอย่างชัดเจน แต่ในทางปฏิบัตินั้นส่วนประกอบทั้ง 3 ส่วนนี้อาจจะรวมอยู่ในอุปกรณ์ตัวเดียวกันก็ได้

2.2.2 โปรโตคอลที่ได้รับการกำหนดโดยมาตรฐาน H.323

โปรโตคอลที่ได้รับการกำหนดโดนมาตรฐาน H.323 ตามรูปที่ 2.8 นั้น มีดังต่อไปนี้



รูปที่ 2.8 H.323 Protocol Stack

- **Audio CODECs**

Audio CODEC จะทำหน้าที่เข้ารหัสสัญญาณเสียงที่ได้รับจากไมโครโฟนก่อนที่จะส่งเสียงไปยังปลายทาง ใน H.323 terminal ที่ทำหน้าที่ส่งเสียง และ ทำหน้าที่ในการถอดรหัสสัญญาณเสียงที่ผ่านการเข้ารหัสแล้วที่ได้รับมาจากต้นทาง ใน H.323 terminal ที่เป็นตัวรับสัญญาณเสียง ซึ่งตัว audio CODEC มีหลายตัวด้วยกัน เช่น G.711, G.722, G.729 เป็นต้น

- **Video CODECs**

Video CODEC จะทำหน้าที่เข้ารหัสวิดีโอที่ได้รับมาจากกล้องก่อนที่จะส่งสัญญาณวิดีโอไปยังปลายทาง ใน H.323 terminal ที่ทำหน้าที่ส่งเสียง และ ทำหน้าที่ถอดรหัสสัญญาณวิดีโอที่ผ่านการเข้ารหัสแล้วที่ได้มาจากต้นทาง ใน H.323 terminal ที่เป็นตัวรับสัญญาณ ก่อนที่จะนำวิดีโอมาแสดงผลได้ video CODEC มีหลายตัวด้วยกัน เช่น H.261, H.263 เป็นต้น

- **H.225 registration, admission, and status (RAS)**

Registration, admission, and status (RAS) เป็นโปรโตคอลที่อยู่ระหว่าง endpoint (อาจเป็น terminal หรือ gateway) กับ gatekeeper RAS จะใช้สำหรับดำเนินการเกี่ยวกับการลงทะเบียน, การควบคุมการเข้าใช้, การเปลี่ยนแปลง bandwidth, สถานะ, และกระบวนการในการยกเลิกการติดต่อระหว่าง endpoint กับ gatekeeper โดย RAS message นั้นจะถูกแลกเปลี่ยนผ่านทาง RAS channel ซึ่ง RAS channel นี้จะต้องเปิดขึ้นระหว่าง endpoint กับ gatekeeper ก่อนที่จะสร้าง channel อื่นๆ

- **H.225 call signaling**

H.225 call signaling ใช้ในการสร้างการเชื่อมต่อระหว่าง H.323 endpoint 2 ตัว โดยมันจะทำการแลกเปลี่ยน H.225 protocol message ทาง call signaling channel ซึ่งถูกเปิดระหว่าง H.323 endpoint 2 ตัว หรือ ระหว่าง endpoint กับ gatekeeper

- **H.245 control signaling**

H.245 control signaling ใช้ในการแลกเปลี่ยน control message ระหว่างปลายทางทั้ง 2 โดย control message จะมีข้อมูลเกี่ยวกับ ปริมาณในการแลกเปลี่ยน, flow control message, และ คำสั่งทั่วๆไป

- **Real-time transfer protocol (RTP)**

RTP เป็นตัวที่ทำให้สามารถส่งเสียงและวิดีโอแบบเวลาจริงได้ และถูกออกแบบมาให้ประยุกต์รวมเข้ากับโปรแกรมประยุกต์ต่างๆได้โดยง่าย โดยตัวโปรโตคอลเองจะระบุไว้เฉพาะฟังก์ชั่นหลักๆพื้นฐานเท่านั้นและให้ทางโปรแกรมประยุกต์ไปเพิ่มเติมส่วนที่ต้องการได้ตามความเหมาะสม ดังนั้นจึงไม่มีการสร้าง layer ใหม่ขึ้นมา และเปรียบเสมือนว่า RTP ทำงานอยู่บน application layer ซึ่ง RTP นี้ถูกออกแบบมาให้ใช้คู่กันกับ UDP (User Datagram Protocol) โดย RTP Header ถูกสร้างขึ้นมาให้มีขนาดเล็กโดยจะบรรจุข้อมูลเฉพาะที่จำเป็นในการทำงานแบบเวลาจริงเท่านั้น เมื่อนำ RTP Header มารวมกับ UDP Header ขนาดของ header ก็ยังมีขนาดเล็กอยู่ทำให้การส่งข้อมูลเป็นไปได้อย่างรวดเร็วนอกจากนี้ตัว RTP ยังได้รับการสนับสนุนการทำงานแบบ multicast ด้วย

- **Real-time control protocol (RTCP)**

RTP ถูกสร้างขึ้นมาใช้เป็นข้อตกลงในการสื่อสารแบบเวลาจริง และถูกออกแบบมาให้โปรแกรมประยุกต์ต่างๆนำไปใช้ได้โดยง่าย ตัว RTP ไม่ใช่ตัวควบคุมทรัพยากร และไม่สามารถรับประกันคุณภาพในการให้บริการสำหรับการให้บริการแบบเวลาจริง ดังนั้นในการทำงานต้องอาศัย Real-time Control Protocol (RTCP) มารวมด้วย RTCP เป็นโปรโตคอลที่ออกแบบมาเพื่อใช้งานร่วมกับ RTP โดยจะควบคุมการส่งข้อมูล, จัดการเกี่ยวกับ minimal control และระบุการทำงานต่างๆ

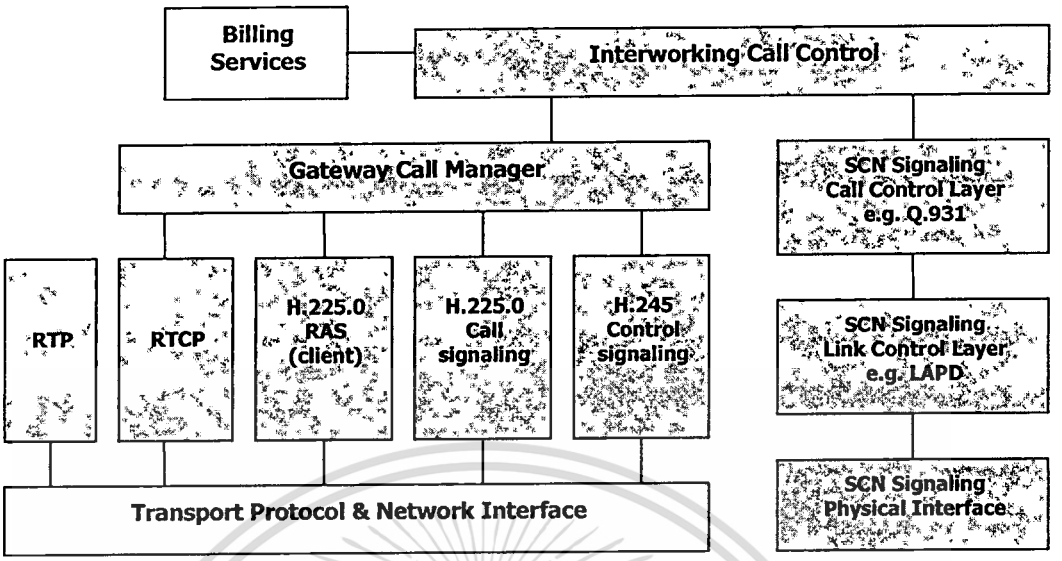
2.2.3 คุณลักษณะของ Gateway และ Gatekeeper

- **คุณลักษณะของ gateway**

Gateway มีหน้าที่แปลงโปรโตคอลที่ใช้สำหรับ call setup และ release และแปลง format ของ media ระหว่างเครือข่ายที่ต่างกัน และส่งผ่านข้อมูลระหว่างเครือข่ายที่เป็น H.323 และไม่เป็น H.323 จากรูปที่ 2.9 โปรแกรมประยุกต์ของ H.323 gateway จะอยู่ใน IP Telephony โดยที่ H.323 gateway จะเชื่อมต่อเครือข่าย IP เข้ากับเครือข่าย SCN (เช่น เครือข่าย ISDN, PSTN)

ทางด้านของ H.323 gateway จะ run H.245 control signaling เพื่อดูแลกับปริมาณในการแลกเปลี่ยน, H.225 call signaling เพื่อดูแลการทำ call setup และ call release, และ H.225 registration, admission, and status (RAS) เพื่อดูแลการลงทะเบียนกับตัว gatekeeper ส่วนทางด้านของ SCN gateway จะ run โปรโตคอลที่ใช้เฉพาะใน SCN (เช่น ISDN และ SS7)

Terminal จะติดต่อสื่อสารกับ gateway โดยใช้โปรโตคอล H.245 control signaling และ H.225 call signaling ซึ่ง gateway จะทำหน้าที่แปลงโปรโตคอลเหล่านี้ไปเป็นโปรโตคอลที่คล้ายกันในเครือข่ายที่ไม่เป็น H.323 gateway เป็นตัวทำการ call set up และ clearing ทั้งเครือข่ายทางด้านที่เป็น H.323 และเครือข่ายด้านที่ไม่เป็น H.323 ด้วย และ เป็นตัวแปลง format ของเสียง, วิดีโอ และข้อมูลด้วย แต่การแปลง format ของเสียงและวิดีโอนั้นไม่จำเป็นต้องทำ ถ้า terminal ทั้ง 2 ด้านมีโหมดในการติดต่อสื่อสารชนิดเดียวกัน เช่น gateway กับ H.320 terminal ในเครือข่าย ISDN ต่างก็ใช้การเข้ารหัส G.711 สำหรับการเข้ารหัสเสียง และใช้ H.261 สำหรับการเข้ารหัสวิดีโอเหมือนกัน ดังนั้นจึงไม่จำเป็นต้องมีการแปลง format ของเสียงและวิดีโอ gateway ยังมีลักษณะของทั้ง H.323 terminal บนเครือข่าย H.323 และ terminal แบบอื่นๆบนเครือข่ายที่ไม่เป็น H.323 ด้วย



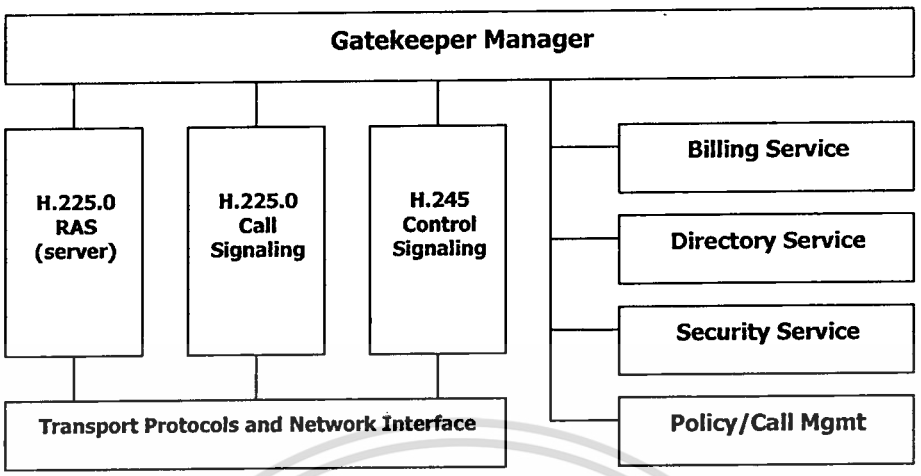
รูปที่ 2.9 Gateway Protocol Stack

Gatekeeper ต้องตรวจดูว่า endpoint ที่เข้ามาลงทะเบียนกับ gatekeeper นั้นเป็น gateway หรือ terminal gateway สามารถรองรับการร้องขอ (call) ได้หลายการร้องขอในเวลาเดียวกัน gateway นั้นเป็นส่วนประกอบที่มาตรฐาน H.323 ได้กำหนดมา แต่ในทางปฏิบัติจริงนั้น มันอาจจะเป็นส่วนหนึ่งของ gatekeeper หรือ MCU ก็ได้

- **คุณลักษณะของ gatekeeper**

Gatekeeper เป็นตัวให้บริการเกี่ยวกับ call-control สำหรับ H.323 endpoint เช่น ให้บริการในการแปลง address และจัดการ bandwidth ตามที่ได้กำหนดไว้ใน RAS โดยในเครือข่าย H.323 นั้นไม่จำเป็นต้องมี gatekeeper ก็ได้

ตามมาตรฐาน H.323 ได้มีการกำหนดการให้บริการของ gatekeeper 2 แบบด้วยกัน คือ การให้บริการหลักที่ gatekeeper จำเป็นต้องมี และการให้บริการที่เป็นทางเลือกคือ gatekeeper ไม่จำเป็นต้องให้บริการก็ได้ การให้บริการที่เป็นทางเลือก เช่น การให้บริการค้นหาเส้นทางให้กับ call-signaling ส่วนการให้บริการหลักที่ gatekeeper มีให้ เช่น การให้บริการตามโปรโตคอล RAS การแปลง address การควบคุมการเข้าใช้งาน การควบคุม bandwidth และการจัดการโซน ตามในรูปที่ 2.10



รูปที่ 2.10 ส่วนประกอบต่างๆของ Gatekeeper

ในเครือข่าย H.323 ที่ไม่มี gatekeeper ก็อาจจะไม่มีความสามารถต่างๆเหล่านี้ แต่เครือข่ายที่มี IP-telephony gateway ควรจะมีตัว gatekeeper เพื่อทำหน้าที่แปลง E.164 telephone address ไปเป็น transport address gatekeeper นั้นเป็นส่วนประกอบที่มาตรฐาน H.323 ได้กำหนดมา แต่ในทางปฏิบัติจริงนั้น มันอาจจะเป็นส่วนหนึ่งของ gateway หรือ MCU ก็ได้

การให้บริการหลักของ gatekeeper มีดังนี้

1. **การแปลง address (Address Translation)**

Call ที่เกิดขึ้นภายในเครือข่าย H.323 อาจใช้ alias เป็นตัวบอกที่อยู่ terminal ปลายทาง ส่วน call ที่เกิดขึ้นภายนอกเครือข่าย H.323 และใช้ gateway ในการเชื่อมต่อเครือข่ายอาจใช้เบอร์โทรศัพท์เป็นตัวบอกที่อยู่ของ terminal ปลายทาง gatekeeper จะแปลงเบอร์โทรศัพท์ (E.164) หรือ alias ของ terminal ปลายทางไปเป็น network address เช่น 204.252.32.456 สำหรับเครือข่ายโปรโตคอลอินเทอร์เน็ตก่อนถึงจะสามารถติดต่อกันได้ เนื่องจากว่าในเครือข่าย H.323 นั้นจะติดต่อระหว่าง endpoint ได้ด้วย network address

2. การควบคุมการเข้าใช้งาน (Admission Control)

Gatekeeper สามารถควบคุมการเข้าใช้งานของ endpoint ในเครือข่าย H.323 ได้ โดย gatekeeper จะใช้ RAS message, การขอเข้าใช้งาน (Admission Request-ARQ), การยืนยัน (Confirm-ACF), และการปฏิเสธ (Reject-ARJ) ในการควบคุม

3. การควบคุม bandwidth (Bandwidth Control)

Gatekeeper สามารถควบคุม bandwidth ได้โดยใช้ RAS messages, การขอ bandwidth (Bandwidth Request-BRQ), การยืนยัน (Confirm-BCF) และการปฏิเสธ (Reject-BRJ) Gatekeeper สามารถควบคุม bandwidth โดยถ้า network manager ได้ระบุระดับ threshold สำหรับจำนวนการติดต่อที่เข้ามาพร้อมกันในเครือข่าย H.323 เอาไว้ ดังนั้น gatekeeper สามารถที่จะปฏิเสธที่จะให้การเชื่อมต่ออีกถ้าการเชื่อมต่อในขณะนั้นถึงระดับ threshold ที่ตั้งเอาไว้แล้ว

4. การจัดการโซน (Zone Management)

Gatekeeper ให้บริการต่างๆสำหรับ terminal, gateway, และ MCU ที่ตั้งอยู่ในโซนที่มันควบคุมเท่านั้น

2.2.4 H.225 Registration, Admission, and Status

H.225 RAS จะใช้ส่งระหว่าง H.323 endpoint (terminal และ gateway) กับ gatekeeper ซึ่งมีหน้าที่ต่างๆดังนี้

- **Gatekeeper Discovery (GRQ)**

กระบวนการในการค้นหา gatekeeper (gatekeeper discovery) เป็นกระบวนการที่ H.323 endpoint ตัดสินใจว่า gatekeeper ตัวไหนที่ endpoint นั้นจะต้องไปลงทะเบียนด้วย การค้นหา gatekeeper นั้นสามารถทำได้ทั้งแบบ static และ dynamic การค้นหาแบบ static นั้น endpoint จะรู้ transport address ของตัว gatekeeper ก่อนล่วงหน้าแล้ว ส่วนการค้นหาแบบ dynamic นั้น endpoint

จะส่ง multicasts GRQ message ไปยัง multicast address ที่ใช้ในการค้นหาของ gatekeeper ทุกตัว โดยถามว่า "ใครเป็น gatekeeper ของฉัน" gatekeeper ตัวหนึ่งหรือหลายตัวก็จะส่ง GCF message ตอบกลับมาว่า "ฉันสามารถเป็น gatekeeper ของคุณได้"

- **Endpoint Registration and location**

Endpoint ทุกตัวจำเป็นต้องลงทะเบียนกับ gatekeeper กระบวนการในการลงทะเบียนนั้นจะทำให้ gatekeeper สามารถที่จะ update ข้อมูล alias address และ transport address ใน database ของมันให้ตรงกับ alias address และ transport address ของ endpoint ซึ่ง gatekeeper จะใช้ alias address และ transport address ในการหาเส้นทางของการ call

- **Other Control**

Gatekeeper ใช้ RAS message เพื่อที่จะทำการจัดการ และควบคุมเรื่องอื่นๆอีก เช่น การควบคุมการเข้าใช้งาน และการจัดการ bandwidth

2.2.5 H.225 Call Signaling และ H.245 Control Signaling

- **H.225 Call Signaling**

H.225 call signaling ใช้สำหรับการ set up การเชื่อมต่อระหว่าง H.323 endpoint (terminal และ gateway) call signaling เป็น H.252 protocol message ที่แลกเปลี่ยนกันบน call-signaling channels ที่น่าเชื่อถือ เช่น H.225 protocol message จะส่งบน TCP ในเครือข่าย H.323

ในกรณีที่ในเครือข่าย H.323 ไม่มีตัว gatekeeper H.225 message จะถูกแลกเปลี่ยนระหว่าง endpoint แต่ถ้าในเครือข่ายมีตัว gatekeeper ด้วย H.225 message จะถูกแลกเปลี่ยนทั้งระหว่าง endpoint โดยตรง เรียกว่า "direct call signaling" หรือ ระหว่าง endpoint ภายหลังจากที่มีการหาเส้นทางโดยผ่าน gatekeeper แล้ว เรียกว่า "gatekeeper-routed call signaling" ส่วนจะเลือกใช้วิธีไหนนั้น gatekeeper จะเป็นตัวตัดสินใจระหว่างที่มีการแลกเปลี่ยน RAS-admission message

Gatekeeper-Routed Call Signaling

Admission message จะถูกส่งระหว่าง endpoint และ gatekeeper บน RAS channel gatekeeper จะรับ call-signaling message บน call-signaling channel จาก endpoint ตัวที่หนึ่ง และหาเส้นทางส่ง call-signaling เหล่านั้นไปยัง endpoint ตัวที่สองบน call-signaling channel ของ endpoint ตัวที่สอง

Direct Call Signaling

ระหว่างยืนยันการเข้าใช้งาน gatekeeper จะเป็นตัวระบุ endpoint ที่สามารถแลกเปลี่ยน call-signaling message กันได้โดยตรง โดย endpoint จะแลกเปลี่ยน call-signaling บน call-signaling channel

- **H.245 Control signaling**

H.245 control signaling เป็นการแลกเปลี่ยน H.245 message ระหว่าง H.323 endpoint ที่ติดต่อสื่อสารกัน H.245 control message จะส่งโดยผ่านทาง H.245 control channel โดย H.245 control channel เป็น logical channel และเปิดอยู่ตลอดเวลาไม่เหมือนกับ media channel และ H.245 control message จะประกอบด้วย message ที่เพื่อแลกเปลี่ยนประสิทธิภาพ (Capabilities Exchange) ของ terminal และใช้เพื่อเปิดและปิด logical channel

Capabilities Exchange

การแลกเปลี่ยนประสิทธิภาพ (Capability Exchange) เป็นกระบวนการที่ใช้ message ที่แลกเปลี่ยนกันของ terminal ที่กำลังติดต่อสื่อสารกันอยู่นั้นเป็นตัววัดประสิทธิภาพในการรับและส่งข้อมูลไปยัง endpoint อีกตัว โดย ประสิทธิภาพในการส่ง (Transmit capabilities) จะเป็นตัวอธิบายถึงความสามารถของ terminal ในการส่ง media streams และ ประสิทธิภาพในการรับ (Receive capabilities) จะเป็นตัวอธิบายถึงความสามารถของ terminal ในการรับและประมวลผล media stream ที่เข้ามา

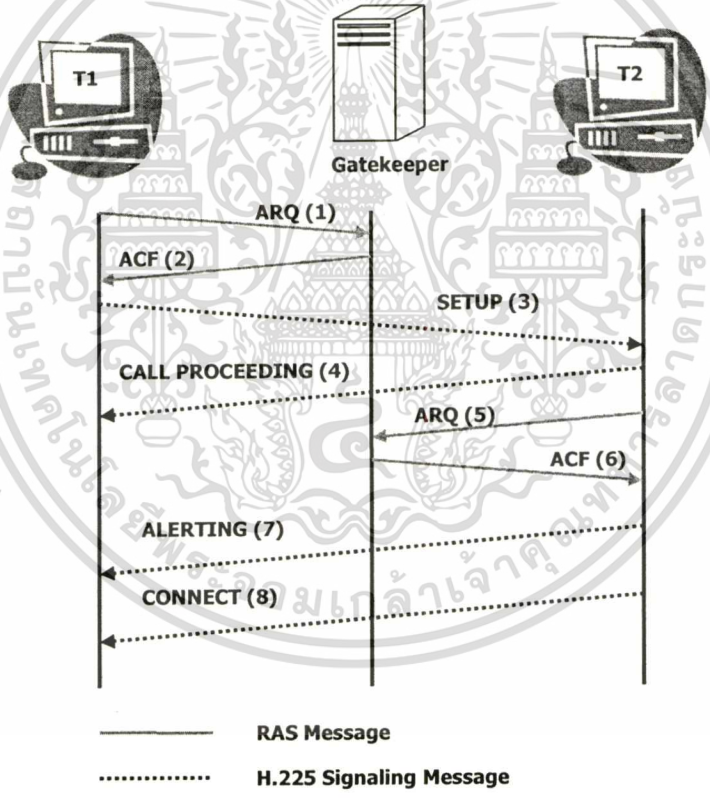
Logical Channel Signaling

Logical channel ใช้สำหรับส่งข้อมูลจาก endpoint ตัวหนึ่ง ไปยัง endpoint อีกตัวหนึ่ง (ในกรณีของการประชุมแบบ point-to-point) หรือ ไปยัง endpoint หลายตัว (ในกรณีของการประชุม

แบบ point-to-multipoint) H.245 จะเตรียม message เพื่อใช้สำหรับเปิดหรือปิด unidirectional logical channel (logical channel เป็น channel ที่สามารถส่งข้อมูลได้ในทิศทางเดียวเท่านั้น)

2.2.6 กระบวนการในการติดต่อ

กระบวนการในการติดต่อเป็นการอธิบายขั้นตอนในการสร้าง H.323 call, การสร้างการติดต่อสื่อสารข้อมูล media, และ การจบการ call (release call) ตัวอย่างด้านล่างจะเป็นเครือข่ายที่ประกอบไปด้วย H.323 terminal (T1 และ T2) ที่เชื่อมต่ออยู่กับ gatekeeper และสมมติว่าใช้ Direct call signaling และ media stream นั้นใช้ RTP encapsulation การเริ่มต้น H.323 call ทำดังรูปที่ 2.11



รูปที่ 2.11 H.323 Call Establishment

กระบวนการในการสร้าง H.323 Call มีดังนี้

1. T1 ส่ง RAS ARQ message บน RAS channel ไปยัง gatekeeper เพื่อลงทะเบียน (registration) T1 ร้องขอการใช้งาน direct call signaling

2. Gatekeeper ยืนยันการเข้าใช้งานของ T1 โดยการส่ง ACF ให้กับ T1 และ gatekeeper จะบอกว่า T1 สามารถใช้ direct call signaling ได้โดยบอกมาใน ACF

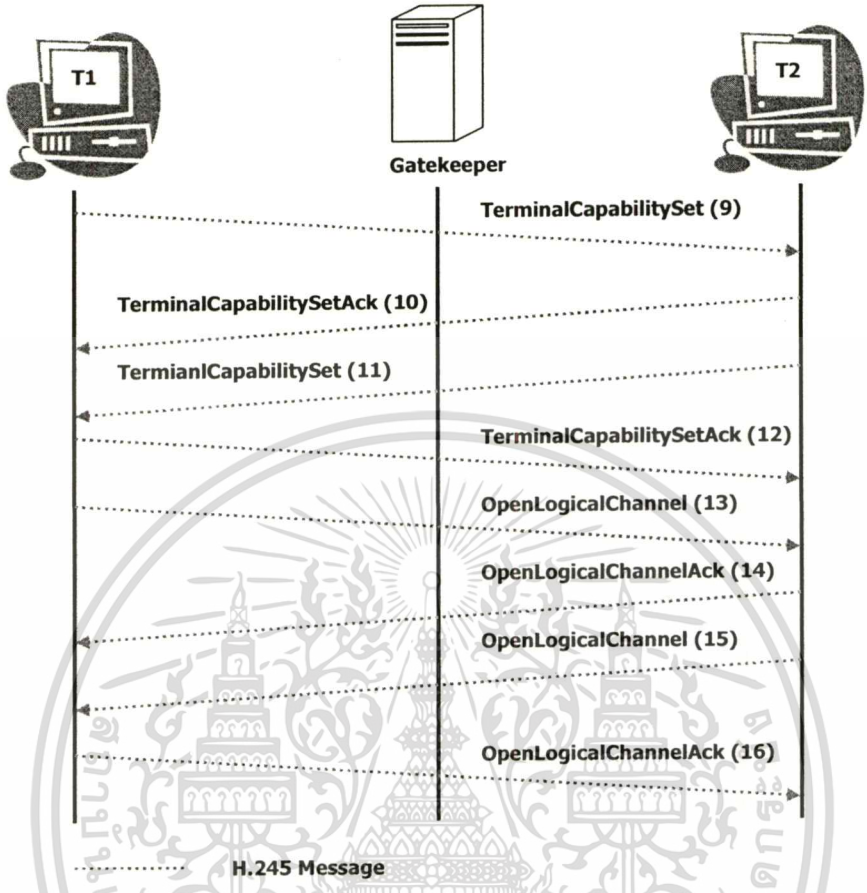
3. T1 ส่ง H.225 call signaling setup message ไปยัง T2 เพื่อทำการร้องขอการติดต่อ

4. T2 ตอบรับการร้องขอนั้นโดยส่ง H.225 call proceeding message ไปยัง T1
5. ในขณะนี้ T2 จำเป็นต้อง register กับตัว gatekeeper โดยส่ง RAS ARQ message ไปยัง gatekeeper บน RAS channel

6. Gatekeeper ยืนยันการลงทะเบียน (registration) โดยการส่ง RAS ACF message ไปยัง T2

7. T2 แจ้งเตือน T1 เกี่ยวกับการสร้างการติดต่อโดยการส่ง H.225 alerting message

8. จากนั้น T2 จะยืนยันการสร้างการติดต่อโดยการส่ง H.225 connect message ไปยัง T1 และ call ก็จะถูกสร้างขึ้น



รูปที่ 2.12 H.323 Control Signaling Flow

9. H.245 control channel ถูกสร้างขึ้นระหว่าง T1 และ T2 จากนั้น T1 จะส่ง H.245 TerminalCapabilitySet message ไปยัง T2 เพื่อที่จะแลกเปลี่ยนประสิทธิภาพของมัน

10. T2 จะตอบกลับไปหา T1 ว่า T2 ได้รู้ประสิทธิภาพของ T1 แล้วโดยการส่ง H.245 TerminalcapabilitySetAck message

11. T2 จะแลกเปลี่ยนประสิทธิภาพของมันกับ T1 โดยส่ง H.245 TerminalCapabilitySet message ไปให้ T1

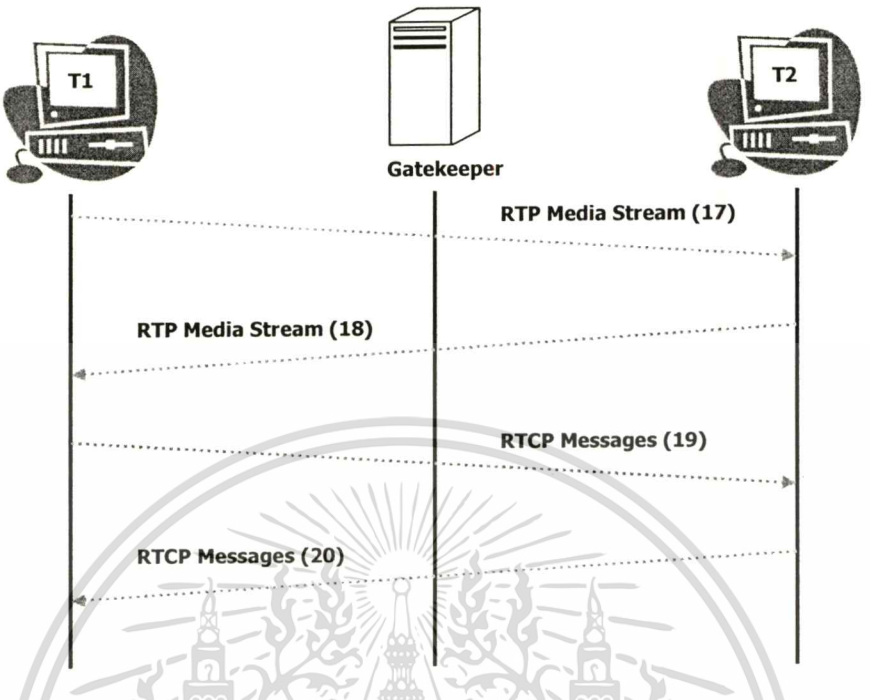
12. T1 จะตอบกลับไปหา T2 ว่า T1 ได้รู้ประสิทธิภาพของ T2 แล้วโดยการส่ง H.245 TerminalcapabilitySetAck message

13. T1 จะเปิด media channel กับ T2 โดยการส่ง H.245 openLogicalChannel message ซึ่ง transport address ของ RTCP channel จะรวมอยู่ใน message นี้ด้วย

14. T2 จะตอบกลับไปหา T1 ว่า T2 ได้รับรู้การสร้าง unidirectional logical channel จาก T1 ไปยัง T2 โดยการส่ง H.245 openLogicalChannelAck message ซึ่งใน message ที่ T2 ส่งกลับไปนั้นจะมี RTP transport address ที่ T2 กำหนดเอาไว้ให้ T1 ใช้สำหรับส่ง RTP media stream และมี RTCP address ที่ T2 ได้รับจาก T1 ในขั้นตอนก่อนหน้านี้

15. จากนั้น T2 จะเปิด media channel กับ T1 โดยการส่ง H.245 openLogicalChannel message และ transport address ของ RTCP channel ก็จะรวมอยู่ใน message ด้วย

16. T1 จะตอบกลับไปหา T2 ว่า T1 ได้รับรู้การสร้าง unidirectional logical channel จาก T2 ไปยัง T1 โดยการส่ง H.245 openLogicalChannelAck message ซึ่งใน message ที่ T2 ส่งกลับไปนั้นจะมี RTP transport address ที่ T1 กำหนดเอาไว้ให้ T2 ใช้สำหรับส่ง RTP media stream และมี RTCP address ที่ T1 ได้รับจาก T2 ในขั้นตอนก่อนหน้านี้ เห็นได้ว่าในขณะนี้การติดต่อสื่อสาร media stream แบบ 2 ทิศทาง (bidirectional media stream communication) ได้ถูกสร้างขึ้นเรียบร้อยแล้ว



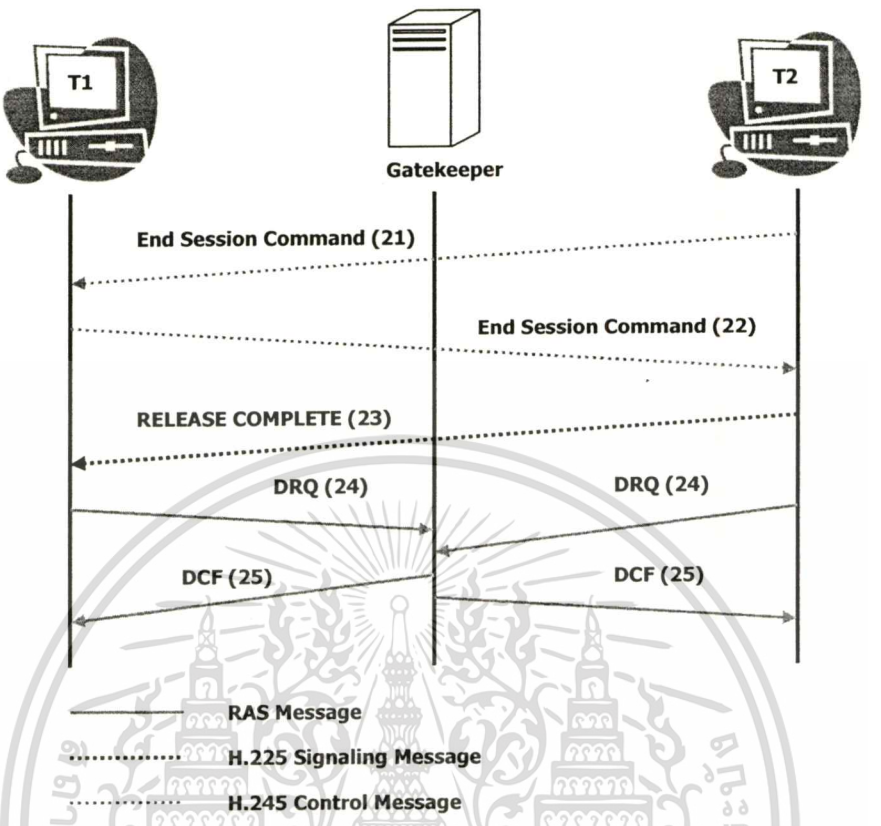
รูปที่ 2.13 H.323 Media Stream and Media Control Flow

17. T1 ส่ง RTP media stream ไปยัง T2

18. T2 ส่ง RTP media stream ไปยัง T1

19. T1 ส่ง RTCP message ไปยัง T2

20. T2 ส่ง RTCP messages ไปยัง T1



รูปที่ 2.14 H.323 Call Release

21. T2 เริ่มต้น call release โดยการส่ง H.245 EndSessionCommand message ไปยัง T1

22. T1 release call ที่ endpoint และยืนยันการ release โดยการส่ง H.245 EndSessionCommand message ไปยัง T2

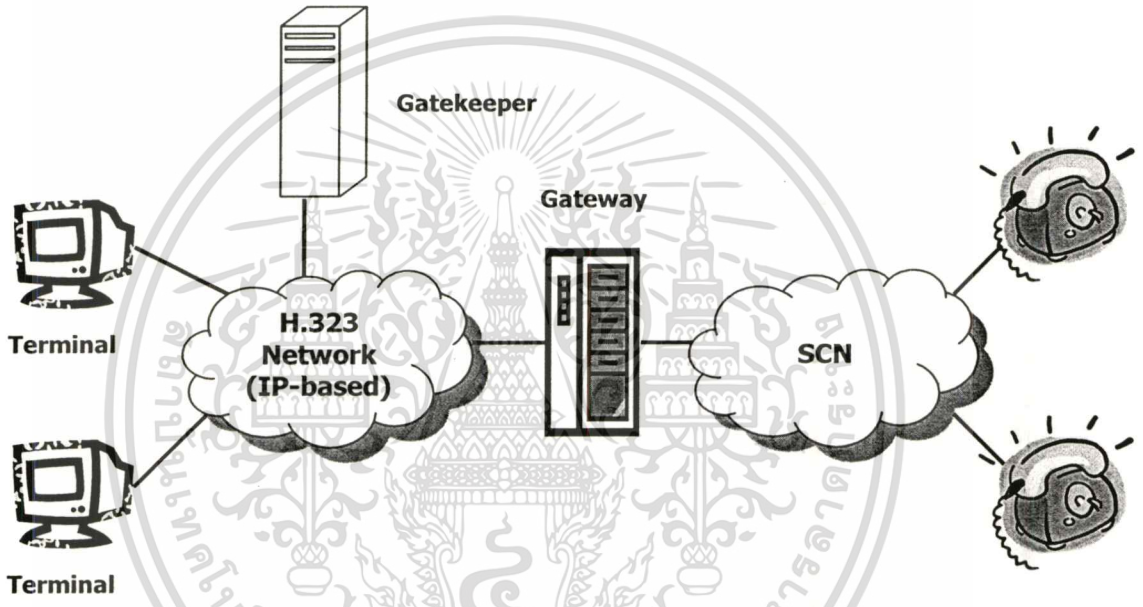
23. T2 ทำการ release call เสร็จสมบูรณ์โดยการส่ง H.225 release complete message ไปยัง T1

24. T1 และ T2 ยกเลิกเชื่อมต่อกับ gatekeeper โดยส่ง RAS DRQ message ไปยัง gatekeeper

25. Gatekeeper ยกเลิกการเชื่อมต่อกับ T1 และ T2 และยืนยันโดยส่ง DCF message ไปยัง T1 และ T2

2.2.7 การเชื่อมต่อเครือข่าย H.323 กับเครือข่าย multimedia อื่นๆ

เครือข่าย H.323 สามารถที่จะเชื่อมต่อกับเครือข่ายอื่นๆได้ โดยเครือข่ายที่นิยมเชื่อมต่อกับเครือข่าย H.323 มากที่สุด คือ IP Telephony โดย H.323 เป็นเครือข่าย IP และเครือข่ายที่มาเชื่อมต่อกับเครือข่าย H.323 คือเครือข่าย SCN ตามในรูปที่ 2.15 โดย SCN จะรวมทั้งเครือข่าย PSTN และเครือข่าย ISDN



รูปที่ 2.15 H.323 Internetworking with SCN

2.3 Java Media Framework (JMF) (Sun Microsystem. 1999)

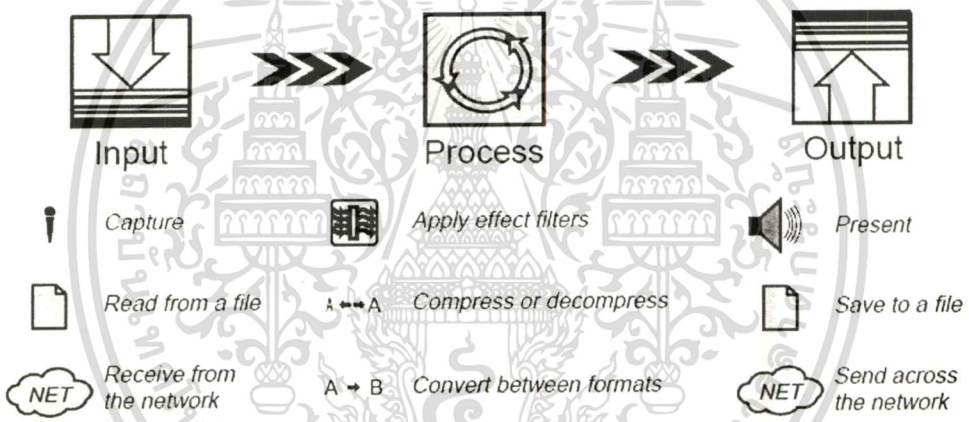
Sun Microsystems ได้จัดทำ Application Programming Interface (API) ที่ใช้สำหรับการส่งข้อมูล media ที่ขึ้นอยู่กับเวลา คือ ข้อมูลของ media จะเปลี่ยนไปตามเวลาที่เปลี่ยนแปลงไป เช่น ข้อมูลเสียง, วิดีโอ, MIDI, และ animation โดยในปัจจุบันเวอร์ชั่นของ JMF คือ เวอร์ชั่น 2.1.1e JMF2.1.1e ทำให้นักพัฒนาสามารถที่จะทำงานเกี่ยวกับ multimedia ได้หลายอย่าง เช่น

- สามารถเล่นไฟล์ multimedia ใน java applet หรือ java application ด้วย format ของไฟล์ที่มันรองรับ เช่น AVI, MIDI, MPEG, Quicktime, และ WAV

- สามารถเล่น media stream จากอินเทอร์เน็ตได้
- Capture เสียงและวิดีโอจากไมโครโฟนและกล้องวิดีโอ และนำมาเก็บใน format ที่รองรับ
- ประมวลผลข้อมูล media ที่ขึ้นกับเวลา และเปลี่ยนแปลง format ของ ชนิดของ content
- ส่งข้อมูลเสียง และวิดีโอแบบเวลาจริงบนอินเทอร์เน็ต
- กระจายเสียงสด หรือ รายการโทรทัศน์

2.3.1 การทำงานกับข้อมูล media ที่ขึ้นกับเวลา (Time-Based Media)

เราสามารถที่จะอธิบายการใช้งานของ media ที่ขึ้นกับเวลาในเทอมของรูปแบบการประมวลผลข้อมูล ได้ดังรูปที่ 2.16



รูปที่ 2.16 Media Processing Model

คุณลักษณะของข้อมูล media ที่ขึ้นอยู่กับเวลา มีดังนี้ คือ

1. Streaming Media

ลักษณะที่สำคัญของ media ที่ขึ้นกับเวลา คือ มันจำเป็นที่จะต้องประมวลผลและส่งให้ทันเวลา ด้วยเหตุนี้ข้อมูล media ที่ขึ้นอยู่กับเวลา ก็จะหมายถึง Streaming media คือมันต้องส่ง, รับ และประมวลผลในเวลาที่เฉพาะเจาะจงเพื่อที่จะแสดงผลลัพธ์ที่ยอมรับได้

Content type

Content type คือ format ของข้อมูล media ที่เก็บไว้ เช่น QuickTime, MPEG และ WAV

Media Stream

Media stream เป็น ข้อมูล media ที่ได้รับมาจากทางเครือข่าย, หรือได้มาจากการ capture จากกล้อง หรือ ไมโครโฟน โดย media stream จะประกอบไปด้วย channel ของข้อมูลหลายช่อง เรียกว่า "track" เช่น ไฟล์ QuickTime จะประกอบไปด้วย track ของเสียง (audio track) และ track ของวิดีโอ (video track) media stream ที่ประกอบไปด้วย track หลาย track นั้น บ่อยครั้งมักจะเรียกว่า "multiplex media stream" หรือ "complex media stream" ดังนั้น การ demultiplexing ก็คือ กระบวนการที่ทำการแยก track แต่ละ track ของ complex media stream ออกมาจากกัน

ชนิดของ track จะบอกถึงชนิดของข้อมูลที่ track นั้นบรรจุอยู่ เช่น เสียง หรือ วิดีโอ ส่วน format ของ track จะบอกถึงโครงสร้างของข้อมูลใน track นั้น

Media stream จะถูกระบุได้จากตำแหน่งของมัน และ โปรโตคอลที่ใช้ในการเข้าถึง media stream เช่น URL จะถูกใช้ระบุตำแหน่งของไฟล์ QuickTime ถ้า ไฟล์นั้นอยู่บนเครื่องที่ทำงานอยู่จะสามารถเข้าถึงได้โดยใช้ โปรโตคอล FILE แต่ถ้าไฟล์นั้นอยู่บน web server ก็จะสามารถเข้าถึงได้โดยผ่านโปรโตคอล HTTP ถ้าไม่สามารถใช้ URL เราสามารถระบุตำแหน่งของ media stream ได้โดยใช้ "media locator"

Media stream สามารถแบ่งได้เป็น 2 ประเภทตามวิธีการที่ข้อมูลถูกส่งมา คือ

- Pull คือ การส่งข้อมูลนั้นจะเริ่มขึ้น และ จะถูกควบคุมโดย client เช่น โปรโตคอล HTTP และ FILE จะเป็นแบบ pull
- Push คือ การส่งข้อมูลนั้นจะเริ่มขึ้น และ จะถูกควบคุมโดย server เช่น โปรโตคอล RTP เป็นแบบ push

Common Media Formats

ตารางต่างๆต่อไปนี้จะแสดงคุณลักษณะของ format media ซึ่งการเลือก format นั้นต้องพิจารณาสิ่งแวดล้อมของปลายทาง และพิจารณาความคาดหวังของผู้ฟังด้วย เช่น ถ้าต้องการส่ง media content ผ่านทาง web ก็จะต้องให้ความสนใจกับเรื่อง bandwidth เป็นพิเศษ

CPU Requirement แสดงถึง power ที่จะเป็นในการประมวลผล Bandwidth Requirement แสดงถึง ความเร็วในการส่งที่จำเป็นเพื่อให้รับและส่งข้อมูลได้เร็วพอที่จะแสดงผลได้อย่างน่าพอใจ บาง format ของ media ถูกออกแบบมาเพื่อ application ที่เฉพาะเจาะจง เช่น H.261 และ H.263 จะใช้สำหรับ Video conference application ส่วน G.723 จะใช้สำหรับ telephony application

ตารางที่ 2.1 Common video format

Format	Content Type	Quality	CPU Requirements	Bandwidth Requirements
Cinepak	AVI QuickTime	Medium	Low	High
MPEG-1	MPEG	High	High	High
H.261	AVI RTP QuickTime	Low	Medium	Medium
H.263	AVI RTP	Medium	Medium	Low
JPEG	AVI RTP	High	High	High
Format	**Content Type**	**Quality**	**CPU Requirements**	**Bandwidth Requirements**
Indeo	QuickTime AVI	Medium	Medium	Medium

ตารางที่ 2.2 Common audio formats

Format	Content Type	Quality	CPU Requirements	Bandwidth Requirements
PCM	AVI QuickTime WAV	High	Low	High
Mu-Law	AVI QuickTime WAV RTP	Low	Low	High
ADPCM (DVI, IMA4)	AVI QuickTime WAV RTP	Medium	Medium	Medium
MPEG-1	MPEG	High	High	High
MPEG Layer3	MPEG	High	High	Medium
GSM	WAV RTP	Low	Low	Low
G.723.1	WAV RTP	Medium	Medium	Low

2. Media Presentation

ข้อมูล media ที่ขึ้นอยู่กับเวลา คือ เสียง และ วิดีโอส่วนใหญ่จะแสดงผลผ่านทางลำโพง และ จอภาพ ซึ่งอุปกรณ์เหล่านี้ก็เป็น "destination" สำหรับ output ของข้อมูล media destination ของ output ของข้อมูล media อาจเป็นการบันทึกอยู่ในรูปของไฟล์ หรือ การส่งผ่านเครือข่ายก็ได้ โดยบางทีจะเรียก destination ของ output ของข้อมูล media ว่า "data sink"

Presentation controls

ในขณะที่กำลังแสดงผลของ media stream นั้น VCR-style presentation controls จะให้ user สามารถควบคุมการแสดงผลได้ เช่น มีปุ่มสำหรับหยุด, เริ่ม, forward และ rewind ภาพยนตร์ได้

Latency

ในการแสดงผล media stream มักจะแสดงผลผ่านทางเครือข่าย ดังนั้นการแสดงผล media stream นั้นไม่สามารถที่จะเริ่มได้ทันที ซึ่งเวลาที่ใช้ในการรอก่อนที่การแสดงผลจะเริ่มขึ้นนั้น เรียกว่า "start latency"

การแสดงผล multimedia นั้นส่วนใหญ่มักจะประกอบไปด้วย media ที่ขึ้นกับเวลาหลายชนิดมา synchronized กันเพื่อที่จะแสดงผล เนื่องจากการแสดงผลต้องนำ media stream หลายชนิดมา synchronize กัน ดังนั้นเราจึงต้องพิจารณาถึง start latency ของแต่ละ stream ด้วย

Presentation Quality

คุณภาพของการแสดงผล media stream นั้นขึ้นอยู่กับหลายๆปัจจัย คือ

- ชนิดของ compression ที่ใช้
- ประสิทธิภาพในการประมวลผลของระบบที่ใช้แสดงภาพ
- Bandwidth ของช่องสัญญาณในกรณีที่ media stream ต้องส่งผ่านทางเครือข่าย โดย bandwidth จะแสดงถึงจำนวน bit ที่สามารถส่งไปได้ในหนึ่งหน่วยเวลา เรียกว่า "bit rate"

ถ้าจำนวนของ frame วิดีโอที่แสดงผลได้ในหนึ่งหน่วยเวลา (frame rate) มีค่าสูงมากเท่าไหร่ คุณภาพของการแสดงผลวิดีโอก็จะยิ่งดีขึ้นมากเท่านั้น โดยปกติ frame rate ที่ยอมรับได้จะอยู่ที่ 30 frames ต่อวินาที

3. Media Processing

ข้อมูลใน media stream นั้นก่อนที่จะมาแสดงผลให้ user ได้นั้นต้องมีการเปลี่ยนแปลงให้เหมาะสมก่อน โดยปกติแล้วขั้นตอนในการประมวลผลที่เกิดขึ้นก่อนการแสดงผล มีดังนี้

- ถ้า stream เป็น multiplex ต้องทำการแยกแต่ละ track ออกจากกันก่อน
- ถ้าแต่ละ track ถูก compress มา ก็ต้องมาทำการ decode ก่อน
- ถ้าจำเป็นอาจต้องแปลง track ให้อยู่ใน format ที่ต่างจาก format เริ่มต้น
- อาจต้องมีการใช้ effect filter กับ track ที่ทำการ decode เรียบร้อยแล้ว

จากนั้น track จะถูกส่งไปยังอุปกรณ์ output ที่เหมาะสม ถ้า media ถูกเก็บเอาไว้แทนที่จะ render ไปยังอุปกรณ์ output ขั้นตอนในการประมวลผลจะแตกต่างกันเล็กน้อย เช่น ถ้าต้องการที่จะ capture เสียงและวิดีโอจากกล่องวิดีโอ, นำมาประมวลผล, และบันทึกเป็นไฟล์ จะมีขั้นตอนดังนี้

- Track ของ เสียง และ วิดีโอ จะถูก capture
- อาจมีการใช้ effect filter กับ track ถ้าต้องการ
- Encode ข้อมูลของแต่ละ track
- Track ที่ compress แล้วจะนำมา multiplex รวมกันเป็น media stream อันเดียว
- Media stream ที่ multiplex เรียบร้อยแล้วจะถูกบันทึกเป็นไฟล์

Demultiplexers and Multiplexers

Demultiplexer จะแยก track ของ media data จาก multiplexed media stream ออกจากกัน ส่วน multiplexer จะทำการรวม track และ track เข้าด้วยกันให้กลายเป็น multiplexed media stream อันเดียว

Codecs

Codec จะทำการ compress และ decompress ข้อมูล media เมื่อ track ถูก encode มันจะถูกแปลงให้อยู่ใน format ที่ compress แล้วและเหมาะสมที่จะเก็บ หรือ ส่งไปยังปลายทาง และ เมื่อ

track ถูก decode มันจะถูกแปลงให้อยู่ใน format ที่ไม่ได้ compress และเหมาะสมที่จะนำไปแสดงผล

แต่ละ codec จะมี format ของ input และ format ของ output แตกต่างกันไปตามแต่ละชนิดของ codec ในบางกรณีอาจใช้ codec หลายตัวในการแปลงข้อมูลจาก format หนึ่งไปเป็นอีก format

Effect Filters

Effect filter จะทำการเปลี่ยนแปลง track ของข้อมูล โดยอาจทำการเพิ่ม effect พิเศษเข้าไป เช่น ทำให้ภาพเบลอ หรือ ทำเสียง echo

Effect filter สามารถแบ่งได้เป็นแบบ pre-processing effect หรือ แบบ post-processing effect ขึ้นอยู่กับว่าต้องการนำ effect ไปใช้ก่อนหรือหลังการทำ codec ในแต่ละ track แต่โดยทั่วไปแล้ว effect filter จะนำไปใช้กับข้อมูลที่ยังไม่ได้ทำการ compress

Renderers

Renderer เป็นอุปกรณ์นามธรรมที่ใช้ในการแสดงผล ถ้าในกรณีของเสียงนั้น อุปกรณ์ที่จะใช้ในการแสดงผลก็จะเป็นการ์ดเสียงซึ่งเป็น hardware ของเครื่องคอมพิวเตอร์ที่จะ output เสียงนั้นไปยังลำโพง ส่วนในกรณีของวิดีโอนั้น อุปกรณ์ที่ใช้ในการแสดงผลก็จะเป็นจอคอมพิวเตอร์นั้นเอง

4. Media Capture

Media ที่ขึ้นอยู่กับเวลานั้นจะถูก capture ได้จาก source ที่มีการดำเนินอยู่ เช่น เสียงสามารถที่จะ capture ได้จากไมโครโฟน หรือ วิดีโอ ก็สามารถที่จะ capture ได้จากกล้อง การ capture ข้อมูลนั้นเปรียบได้กับ "input" phase ของรูปแบบการประมวลผล media

อุปกรณ์ที่ทำหน้าที่ในการ capture อาจจะส่ง media stream หลายสาย เช่น กล้องวิดีโอก็อาจจะส่งทั้งเสียงและวิดีโอ ซึ่ง stream เหล่านี้อาจถูก capture และ เปลี่ยนแปลงให้เหมาะสมโดย

แยกทำแต่ละ stream หรือ จะรวมกันเป็น multiplexed stream สายเดียวที่ประกอบไปด้วยทั้ง track ของเสียง และ track ของวิดีโอ

Capture Devices

การ capture ข้อมูล media ที่ขึ้นอยู่กับเวลานั้น จำเป็นจะต้องใช้ hardware ที่เฉพาะเจาะจง เช่น การ capture เสียงก็จะใช้ไมโครโฟนและการ์ดเสียง การ capture การกระจายโทรทัศน์ก็ต้องใช้ TV Tuner และ video capture card ระบบส่วนใหญ่จะมีบริการค้นหาอุปกรณ์ที่ใช้ในการ capture ให้

อุปกรณ์ที่ใช้ในการ capture นั้นสามารถเป็นลักษณะของ push หรือ pull source ก็ได้ เช่น กล้องถ่ายภาพนิ่งเป็น pull source เพราะ user เป็นคนควบคุมว่าเมื่อไหร่ที่จะต้องการให้มัน capture ภาพ ส่วนไมโครโฟนจะเป็นลักษณะของ push source เพราะว่า source มีการดำเนินไปอย่างต่อเนื่องและจะได้เป็น stream ของเสียง

Format ของ media stream ที่ได้จากการ capture นั้นจะขึ้นกับการประมวลผลที่อุปกรณ์ capture ดำเนินการ อุปกรณ์บางตัวจะประมวลผลข้อมูลเพียงเล็กน้อยและส่งข้อมูลดิบที่ไม่ได้ compress ไปเลย แต่อุปกรณ์บางตัวก็จะส่งข้อมูลใน format ที่ compress เรียบร้อยแล้ว

Capture Controls

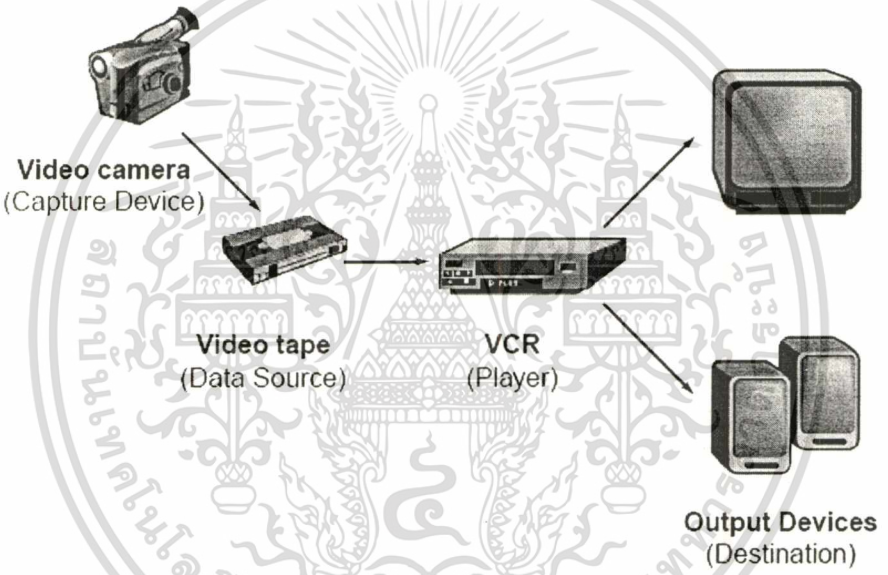
การควบคุมในบางครั้งจะให้ user สามารถที่จะจัดการกระบวนการในการ capture เช่น capture control panel จะให้ user สามารถที่จะระบุ data rate และชนิดของการ encoding สำหรับ stream ที่ capture มาและให้ user สามารถที่จะเริ่ม และ หยุดกระบวนการในการ capture ได้

2.3.2 สถาปัตยกรรมของ JMF

Java Media Framework (JMF) ได้จัดเตรียมสถาปัตยกรรมและ messaging protocol ที่นำมารวมให้สอดคล้องกันสำหรับจัดการเรื่องการรับข้อมูล, การประมวลผลข้อมูล, และการส่งข้อมูล media ที่ขึ้นกับเวลา JMF ถูกออกแบบมาให้รองรับชนิดของ content ของ media ที่เป็นมาตรฐานส่วนใหญ่ เช่น AVI, GSM, MIDI, MPEG, QuickTime, RMF, และ WAV

การใช้ JMF ทำให้เราสามารถสร้าง applet และ application ที่แสดงผล capture จัดการ และเก็บข้อมูล media ที่ขึ้นกับเวลาได้

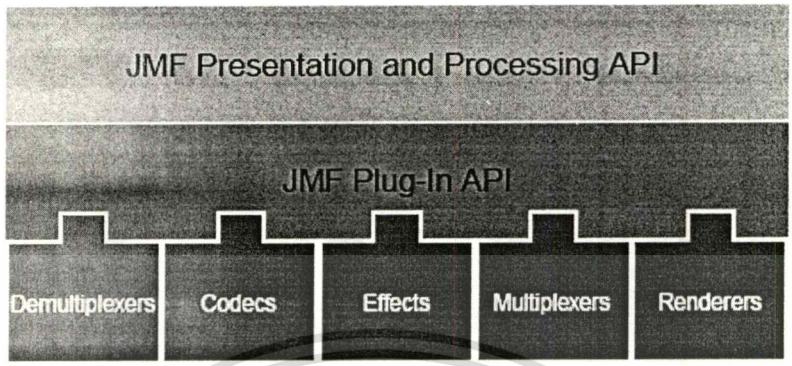
อุปกรณ์ VCR เป็นอุปกรณ์ที่มีรูปแบบการทำงานสำหรับบันทึกข้อมูล ประมวลผล และแสดงผลข้อมูล media ที่ขึ้นกับเวลา เมื่อเราดูภาพยนตร์โดยใช้ VCR แสดงว่าเราได้ทำการเตรียม media stream ให้กับ VCR โดยการใส่วิดีโอไปให้มัน VCR ทำการอ่านและแปลข้อมูลในเทปและส่งสัญญาณที่เหมาะสมไปยังเครื่องโทรทัศน์และลำโพง



รูปที่ 2.17 การบันทึกข้อมูล, ประมวลผลข้อมูล, และการแสดงผลข้อมูล media ที่ขึ้นกับเวลา

JMF ก็มีรูปแบบการทำงานเช่นเดียวกัน คือ "data source" ที่ห่อหุ้ม media stream เอาไว้ก็เปรียบเหมือนวิดีโอเทป และ "player" ทำหน้าที่ประมวลผลและควบคุมการทำงาน ก็เปรียบเหมือน VCR ส่วนที่แสดงผล หรือ capture เสียง และ วิดีโอด้วย JMF นั้น JMF ต้องการอุปกรณ์ input และ output ที่เหมาะสม เช่น ไมโครโฟน กล้อง ลำโพง และ จอภาพ

Data source และ player รวมกันอยู่ในส่วนของ API ของ JMF ในระดับสูง สำหรับจัดการการ capture ข้อมูล การแสดงผล และการประมวลผลข้อมูล media ที่ขึ้นกับเวลา



รูปที่ 2.18 สถาปัตยกรรม JMF ในระดับสูง

Manager

JMF API ประกอบด้วย interface ที่เป็นตัวกำหนดพฤติกรรมและการตอบโต้ของ object ที่ใช้ในการ capture, ประมวลผล, และ แสดงผลข้อมูล media ที่ขึ้นอยู่กับเวลา "Managers" ซึ่งเป็นวัตถุที่ใช้ในการเชื่อมตรงกลางนั้น ได้รวมการ implement interface ที่สำคัญที่ทำให้สามารถเรียกใช้ได้โดยไม่เหมือนว่ามีคลาสอยู่ เราสามารถจินตนาการว่า managers เป็นเหมือน object ที่สามารถ interface ได้กับ object 2 ตัวที่แตกต่างกัน เช่น *Manager* ทำให้เราสามารถสร้าง *Player* จาก *DataSource* ได้

JMF ใช้ managers 4 ตัวด้วยกัน คือ

- *Manager* : เราเรียกใช้ *Manager* เพื่อที่จะสร้าง *Players, Processors, DataSources,* and *DataSinks* เช่น ถ้าต้องการที่จะ render *DataSource* เราสามารถใช้ *Manager* สร้าง *Player* เพื่อใช้ render *DataSource* ได้

- *PackageManager* : manager ตัวนี้จะทำหน้าที่ในการดูแล registry ของ packages ที่ประกอบไปด้วย JMF classes เช่น *Players, Processors, DataSources,* and *DataSinks*

- *CaptureDeviceManager* : manager ตัวนี้จะดูแลเกี่ยวกับ registry ของอุปกรณ์ที่ใช้ในการ capture

- *PlugInManager* : manager ตัวนี้ดูแลเกี่ยวกับ registry ของ component ที่ใช้ในการประมวลผลที่เป็น JMF plug-in เช่น *Multiplexers, Demultiplexers, Codecs, Effects,* และ *Renderer*

ถ้าต้องการเขียนโปรแกรมที่ขึ้นอยู่กับ JMF จะต้องใช้ Manager *create* method ในการสร้าง *Players, Processors, DataSource,* และ *DataSinks* สำหรับ application ถ้าเรา capture ข้อมูล media จากอุปกรณ์ input เราจำเป็นที่จะต้องใช้ *CaptureDeviceManager* เพื่อที่จะหาว่ามีอุปกรณ์ตัวไหนสามารถทำงานและเข้าถึงข้อมูลที่ต้องการได้บ้าง ถ้าเราสนใจในการควบคุมว่ากระบวนการประมวลผลอะไรที่กำลังทำกับข้อมูล เราสามารถค้นหา *PlugInManager* เพื่อที่จะกำหนดว่า plug-in ตัวไหนที่ได้ลงทะเบียนเรียบร้อยแล้ว

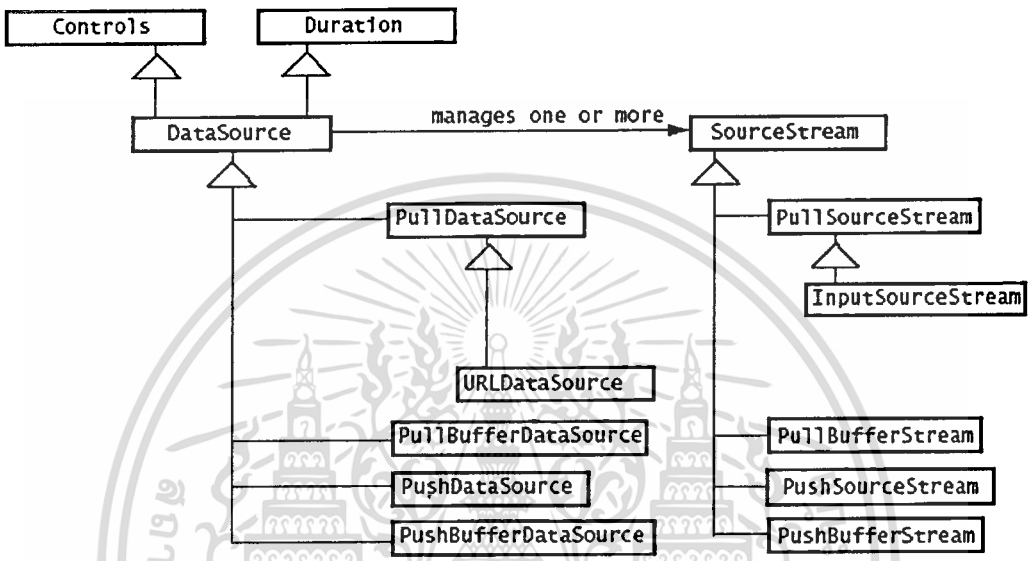
Data Model

JMF media players ใช้ *DataSources* เพื่อที่จะจัดการเกี่ยวกับการส่ง content ของ media โดย *DataSources* object อาจจะเป็น media เสียง, media วิดีโอ, หรือ ทั้ง 2 อย่างประกอบกัน *DataSources* จะห่อหุ้มทั้งตำแหน่งของ media และ โปรโตคอลและ software ที่จะต้องใช้ในการส่ง media เมื่อทำการสร้าง *DataSources* แล้วนำไปป้อนให้ *Player* เพื่อทำการ render *Player* จะไม่สนใจเลยว่า DataSources เริ่มต้นมาจากที่ไหน หรือ รูปแบบดั้งเดิมของมันคืออะไร

DataSources อาจเป็นไฟล์ในเครื่อง หรือ อาจเป็น stream ที่ได้รับมาจากอินเทอร์เน็ตก็ได้ *DataSources* จะถูกระบุตำแหน่งได้โดยทั้ง JMF *MediaLocator* หรือ โดย URL ซึ่ง JMF ได้กำหนดชนิดของ DataSource objects ตามลักษณะการส่งข้อมูลเอาไว้ดังนี้ คือ

- Pull Data-Source : client ทำหน้าที่เริ่มการส่งข้อมูล และ ควบคุมการไหลของข้อมูล โปรโตคอลที่ใช้สำหรับข้อมูลประเภทนี้ คือ Hypertext Transfer Protocol (HTTP) และ FILE

- Push Data-Source : server ทำหน้าที่เริ่มการส่งข้อมูล และ ควบคุมการไหลของข้อมูล Push data-sources คือพวก broadcast media, multicast media, และ video-on-demand (VOD) สำหรับ broadcast data จะใช้โปรโตคอล Real-time Transport Protocol (RTP) ส่วน VOD จะใช้โปรโตคอล MediaBase



รูปที่ 2.19 JMF Data Model

Capture Device

อุปกรณ์ที่ใช้ในการ capture คือ ฮาร์ดแวร์ที่ใช้ในการ capture ข้อมูลมา เช่น ไมโครโฟน กล้องถ่ายภาพ หรือ กล้องวิดีโอ ข้อมูล media ที่ capture ได้จะถูกส่งไปยัง Player เพื่อทำการ render และ ประมวลผลเพื่อที่จะแปลงให้อยู่ในรูปของ format อื่น หรือ เก็บเอาไว้ใช้ในอนาคต

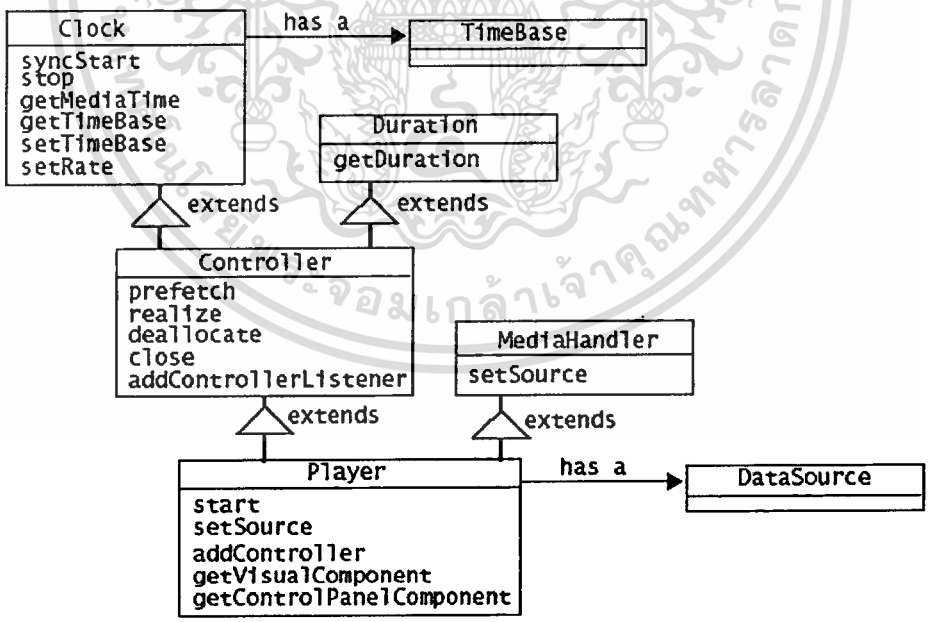
อุปกรณ์ที่ใช้ในการ capture ก็จะเป็นเหมือน DataSources เช่น อุปกรณ์ที่ส่งข้อมูลให้ตรงตามกำหนดเวลา ก็จะเป็น *PushDataSource* ชนิดของ DataSource ที่ใช้กับอุปกรณ์ที่ capture ข้อมูล คือ *PushDataSource*, *PushBufferDataSource*, *PulldataSource*, หรือ *PullBufferDataSource*

Players

Player ประมวลผล input stream ของข้อมูล media และ render มันที่เวลาที่เหมาะสม *DataSource* ใช้เพื่อส่ง input media-stream ไปยัง *Player* ส่วนการ render ที่ปลายทางนั้นจะขึ้นอยู่กับชนิดของ media ที่ต้องการแสดง

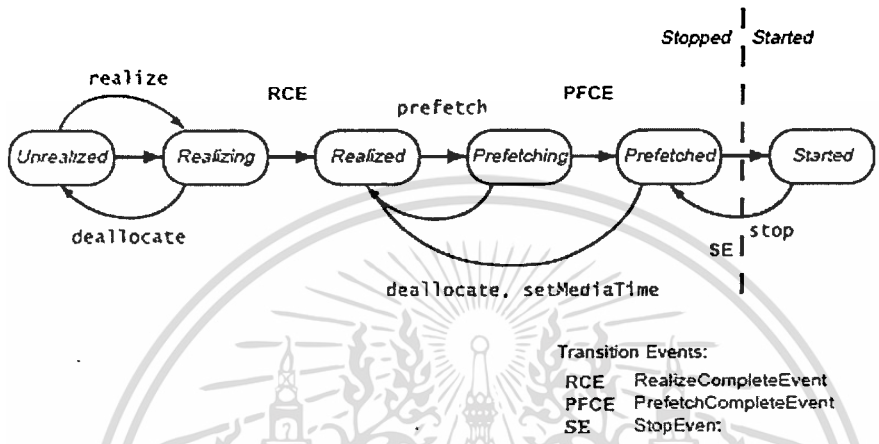
รูปที่ 2.20 JMF Player Model

Player รองรับ user control ที่เป็นมาตรฐาน และ ยืดหยุ่นให้กับข้อจำกัดบางอย่างที่ถูกบังคับโดย *Clock* และ *Controller*



รูปที่ 2.21 JMF Players

Player แบ่งได้ออกเป็น 6 states ด้วยกัน โดย Clock interface จะกำหนด 2 states หลัก คือ **Stopped** และ **Started** แต่เพื่อความสะดวกในการจัดการทรัพยากร *Controller* จึงแตก state **Stopped** ออกมาเป็น 5 states ด้วยกัน คือ **Unrealized, Realizing, Realized, Prefetching,** และ **Prefetched**



รูปที่ 2.22 Player states

ในการทำงานปกติ *Player* จะค่อยเปลี่ยนแปลง state ไปทีละ state จะถึง state **Started**

- *Player* จะอยู่ใน **Unrealized** state เมื่อถูกสร้างขึ้น แต่มันยังไม่รู้เกี่ยวกับ media ของมันเลย

- เมื่อ method **realize()** ของ *Player* ถูกเรียก *Player* จะเปลี่ยน state จาก **Unrealized** state ไปยัง **Realizing** state ใน **Realizing** state *Player* จะอยู่ในกระบวนการของการกำหนดความต้องการทรัพยากรของมัน ในระหว่าง Realization นั้น *Player* จะได้รับทรัพยากรตามที่มันต้องการเพียงครั้งเดียวเท่านั้น ซึ่งทรัพยากรที่มันต้องการนั้นจะรวมถึงทรัพยากรที่ใช้ในการ render ด้วย *Player* ที่อยู่ใน **Realizing** state นั้นจะ download สิ่งที่จำเป็นผ่านทางเครือข่าย

- เมื่อ *Player* ทำการ **Realizing** เสร็จเรียบร้อยแล้ว มันจะย้ายไปอยู่ใน **Realized** state โดย **Realized** Player จะรู้ว่ามันต้องการทรัพยากรอะไร และรู้ข้อมูลเกี่ยวกับชนิดของ media ที่

มันต้องแสดงผล ใน state นี้ *Player* สามารถที่จะเตรียม visual components และ control เพราะว่า **Realized** *Player* รู้ว่าต้อง render ข้อมูลของมันอย่างไร และ *Player* ก็สามารถที่จะติดต่อกับ object อื่นๆในระบบได้แล้ว

- เมื่อ method **prefetch()** ถูกเรียก *Player* จะย้ายจาก **Realized** state ไปเป็น **Prefetching** state ที่ state นี้ *Player* จะเตรียมที่จะแสดงผล media ในระหว่าง phase นี้ *Player* จะโหลดข้อมูล media ของมันไว้ล่วงหน้า, ได้รับทรัพยากรที่เอาไว้ใช้แต่เพียงผู้เดียว, และทำสิ่งอื่นๆที่มันต้องทำเพื่อที่จะแสดงผล media state **Prefetching** อาจจะถูกเรียกอีกครั้งถ้า media ของ *Player* object ที่มันแสดงผลเปลี่ยนตำแหน่ง หรือ ถ้าการเปลี่ยนแปลง rate ของ *Player* object ต้องการ buffer เพิ่มเติม หรือ มีการประมวลผลตัวอื่นเกิดขึ้น

- เมื่อ *Player* ทำขั้นตอน **Prefetching** เสร็จสิ้น มันก้อจะเข้าไปสู่ **Prefetched** state ที่ state นี้ *Player* พร้อมที่จะเริ่มต้น (start)

- เมื่อเราเรียก method **start()** *Player* จะเข้าสู่ **Started** state ที่ state นี้ *Player* พร้อมที่จะแสดงข้อมูล media ของมันแล้ว

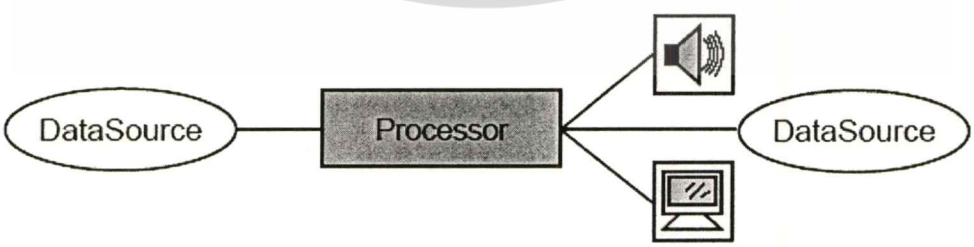
ทุก method ใน *Player* นั้น ไม่สามารถที่จะเรียกได้ในทุก state เพื่อป้องกัน race condition ตารางที่ 2.3 จะระบุข้อบังคับในการเรียกใช้ method ใน state ต่างๆ ถ้าเราเรียก method ใน state ที่ห้ามเรียก *Player* จะส่ง error มา

ตารางที่ 2.3 Method restrictions for players

Method	Unrealized Player	Realized Player	Prefetched Player	Started Player
addController	NotRealizedError	legal	legal	ClockStartedError
deallocate	legal	legal	legal	ClockStartedError
getControlPanelComponent	NotRealizedError	legal	legal	legal
getGainControl	NotRealizedError	legal	legal	legal
getStartLatency	NotRealizedError	legal	legal	legal
getTimeBase	NotRealizedError	legal	legal	legal
getVisualComponent	NotRealizedError	legal	legal	legal
mapToTimeBase	ClockStoppedException	ClockStoppedException	ClockStoppedException	legal
removeController	NotRealizedError	legal	legal	ClockStartedError
setMediaTime	NotRealizedError	legal	legal	legal
setRate	NotRealizedError	legal	legal	legal
setStopTime	NotRealizedError	legal	legal	StopTimeSetError if previously set
setTimeBase	NotRealizedError	legal	legal	ClockStartedError
syncStart	NotPrefetchedError	NotPrefetchedError	legal	ClockStartedError

Processors

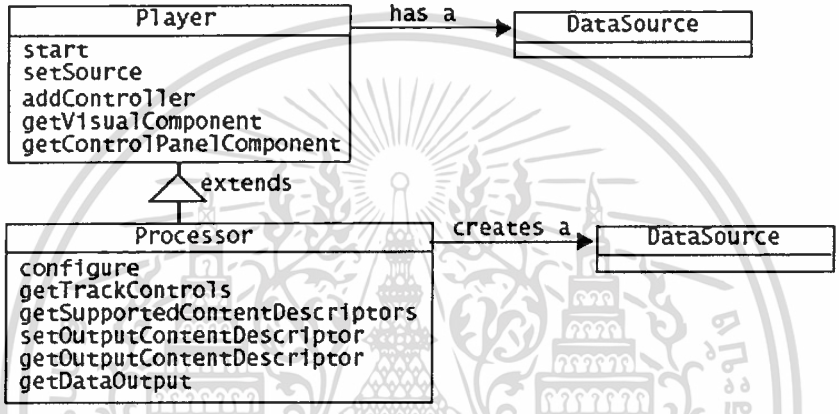
Processors เป็น *Player* ชนิดหนึ่ง สามารถที่จะใช้แสดงผลข้อมูล media ได้เช่นกัน โดย *Processor* interface extend มาจาก *Player* สิ่งที่ *Processor* ต่างจาก *Player* ก็คือ *Processor* สามารถควบคุมการประมวลผลที่ทำกับ input media stream



รูปที่ 2.23 JMF processor model

นอกจากการ render ข้อมูล media เพื่อไปยังอุปกรณ์แสดงผลแล้ว *Processor* ยังสามารถที่จะแสดงผล media โดยผ่านทาง *DataSource* ดังนั้นมันจึงสามารถแสดงผลโดย *Player* หรือ *Processor* อีกตัวได้ นอกเหนือจากการถูกจัดการโดย *Processor* อีกตัว หรือ สามารถที่จะแปลงให้อยู่ในรูปแบบ format อื่นๆได้

Processor คือ *Player* ที่มี input เป็น *DataSource*, ทำการประมวลผลข้อมูล media ตามที่ user กำหนด, และ output ก็คือข้อมูล media ที่ประมวลผลเรียบร้อยแล้ว

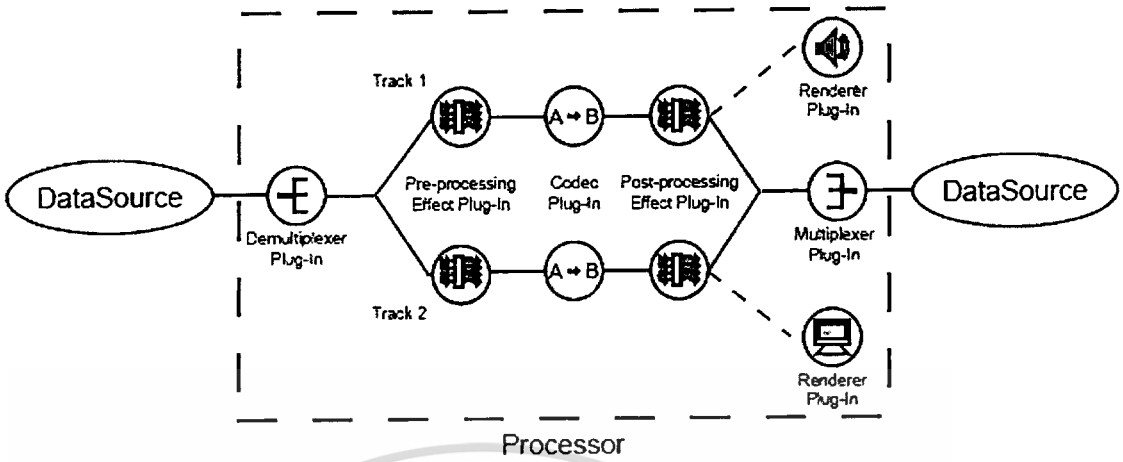


รูปที่ 2.24 JMF Processors

Processor สามารถส่งข้อมูลที่เป็น output ไปยังอุปกรณ์ที่ใช้ในการแสดงผล หรือ ไปยัง *DataSource* ถ้าข้อมูลถูกส่งไปยัง *DataSource* *DataSource* จะสามารถใช้เป็น input ให้กับ *Player* หรือ *Processor* อีกตัวได้ หรือ จะใช้เป็น input ไปยัง *DataSink* ก็ได้

การประมวลที่ทำโดย Player จะถูกกำหนดไว้ก่อนแล้ว แต่สำหรับ Processor นั้นจะอนุญาตให้นักพัฒนา application สามารถกำหนดชนิดของการประมวลผลที่ต้องการมาประยุกต์ใช้กับข้อมูล media ได้

การประมวลผลข้อมูล media จะถูกแบ่งออกเป็นหลายขั้นตอนด้วยกัน คือ



รูปที่ 2.25 Processor stages

- Demultiplexing คือ กระบวนการของการวิเคราะห์ input stream ถ้า input stream ประกอบไปด้วย track หลาย track มันจะถูกแยกออกมาและ output แยกออกจากกันไป เช่น ไฟล์ QuickTime อาจจะถูก demultiplexed เพื่อแยกออกเป็น track ของเสียง และ track ของวิดีโอ การ demultiplexing นี้จะทำโดยอัตโนมัติเมื่อไหร่ก็ตามที่ input stream นั้นเป็นแบบ multiplexed data

- Pre-processing คือ กระบวนการในการประยุกต์ใช้ algorithm ที่เพิ่ม effect ไปที่ track ที่แยกออกมาจาก input stream

- Transcoding คือ กระบวนการในการแปลงแต่ละ track ของข้อมูล media จาก input format ไปเป็นอีก format โดยเมื่อ data stream ถูกแปลงจากประเภท compressed เป็นเป็นประเภท uncompressed จะเรียกว่าการ decoding แต่ในทางตรงกันข้าม ถ้า data stream ถูกแปลงจากประเภท uncompressed ไปเป็นแบบ compressed จะเรียกว่าการ encoding

- Post-Processing คือ กระบวนการในการประยุกต์ใช้ algorithm ที่เพิ่ม effect ไปที่ track ที่ถูก decoded เรียบร้อยแล้ว

- Multiplexing คือ กระบวนการในการซ้อนทับ track ของ media ที่ถูก transcoded แล้วให้รวมมาอยู่ในรูปของ output stream อันเดียวเท่านั้น เช่น track ของเสียง และ track ของวิดีโอ ที่ถูกแยกออกจากกันแล้วนั้นจะถูก multiplexed รวมกันไปเป็น MPEG-1 data stream อันเดียว เราสามารถที่จะระบุ data type ของ output stream ได้โดยใช้ method **setOutputContentDescriptor()** ของ *Processor*

- Rendering คือ กระบวนการในการแสดงผล media ให้กับ user

การประมวลผลในแต่ละขั้นตอนนั้นจะถูกทำโดย component ที่ใช้ในการประมวลผลแตกต่างกัน โดย component ที่ใช้ในการประมวลผลเหล่านี้ คือ JMF plug-ins ถ้า Processor รองรับ TrackControls เราจะสามารถที่จะเลือก plug-ins ที่เราต้องการให้มันใช้ในการประมวลผลใน track ใดโดยเฉพาะได้ JMF plug-ins มีอยู่ 5 ชนิดด้วยกัน คือ

- Demultiplexer ทำหน้าที่วิเคราะห์ media stream เช่น WAV, MPEG หรือ QuickTime ถ้า stream นั้นเป็นแบบ multiplexed มันจะทำการแยกแต่ละ track ออกจากกัน

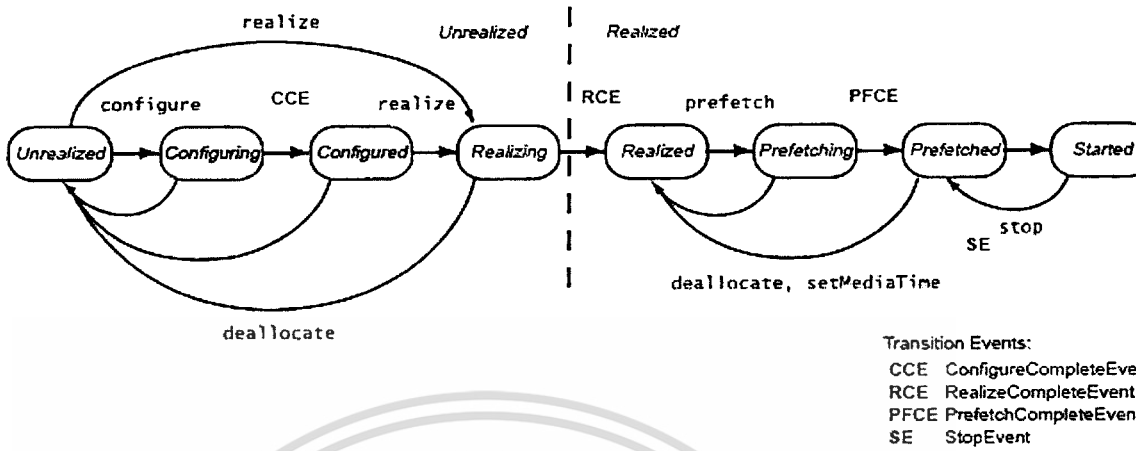
- Effect ทำหน้าที่ประมวลผล effect พิเศษให้แต่ละ track ของข้อมูล media

- Codec ทำหน้าที่ encoding และ decoding ข้อมูล

- Multiplexer ทำหน้าที่รวม track หลายๆ track ของข้อมูล input ไปเป็น output stream อันเดียวที่เกิดจากการเอาแต่ละ track มาซ้อนทับกัน และส่ง stream ที่ได้ไปเป็น output *DataSource*

- Renderer ทำหน้าที่ประมวลผลข้อมูล media และส่งไปยังปลายทาง เช่น จอภาพ หรือ ลำโพง

นอกเหนือจาก state ของ *Player* 6 state ที่ได้กล่าวไปแล้วนั้น *Processor* ยังมี state ที่เพิ่มเติมเข้าไปอีก 2 state โดย 2 state นี้จะเกิดขึ้นก่อนที่ *Processor* จะเข้าไปใน **Realizing** state แต่อยู่หลัง **Unrealized** state



รูปที่ 2.26 Processor states

- *Processor* จะเข้าไป **Configuring** state เมื่อ method **configure()** ถูกเรียก ในขณะที่ *Processor* อยู่ใน **Configuring** state นั้น Processor จะติดต่อไปยัง *DataSource*, จะทำการ demultiplex input stream, และจะหาข้อมูลเกี่ยวกับ format ของข้อมูล input

- *Processor* จะย้ายไปที่ **Configured** state เมื่อมันถูกเชื่อมต่อกับ *DataSource* เรียบร้อยแล้ว และ ได้รู้ format ของข้อมูลแล้ว เมื่อ *Processor* มาถึง **Configured** state แล้ว **ConfigureCompleteEvent** จะถูกประกาศไว้

- เมื่อ method **Realized()** ถูกเรียกขึ้น *Processor* จะย้ายไปยัง **Realized** state เมื่อ *Processor* อยู่ใน **Realized** state แสดงว่ามันถูกสร้างอย่างแท้จริงแล้ว

ทุก method ใน *Process* นั้น ไม่สามารถที่จะเรียกได้ในทุก state เพื่อป้องกัน race condition ตารางที่ 2.4 จะระบุข้อบังคับในการเรียกใช้ method ใน state ต่างๆ ถ้าเราเรียก method ใน state ที่ห้ามเรียก *Processor* จะส่ง error มา

ตารางที่ 2.4 Method restrictions for processors

Method	Unrealized Processor	Configuring Processor	Configured Processor	Realized Processor
addController	NotRealizedError	NotRealizedError	NotRealizedError	legal
deallocate	legal	legal	legal	legal
getControlPanelComponent	NotRealizedError	NotRealizedError	NotRealizedError	legal
getControls	legal	legal	legal	legal
getDataOutput	NotRealizedError	NotRealizedError	NotRealizedError	legal
getGainControl	NotRealizedError	NotRealizedError	NotRealizedError	legal
getOutputContentDescriptor	NotConfiguredError	NotConfiguredError	legal	legal
getStartLatency	NotRealizedError	NotRealizedError	NotRealizedError	legal
getSupportedContent-Descriptors	legal	legal	legal	legal
getTimeBase	NotRealizedError	NotRealizedError	NotRealizedError	legal
getTrackControls	NotConfiguredError	NotConfiguredError	legal	FormatChange-Exception
getVisualComponent	NotRealizedError	NotRealizedError	NotRealizedError	legal
mapToTimeBase	ClockStoppedException	ClockStoppedException	ClockStoppedException	ClockStopped-Exception
realize	legal	legal	legal	legal
removeController	NotRealizedError	NotRealizedError	NotRealizedError	legal
setOutputContentDescriptor	NotConfiguredError	NotConfiguredError	legal	FormatChange-Exception
setMediaTime	NotRealizedError	NotRealizedError	NotRealizedError	legal
setRate	NotRealizedError	NotRealizedError	NotRealizedError	legal
setStopTime	NotRealizedError	NotRealizedError	NotRealizedError	legal
setTimeBase	NotRealizedError	NotRealizedError	NotRealizedError	legal
syncStart	NotPrefetchedError	NotPrefetchedError	NotPrefetchedError	NotPrefetchedError

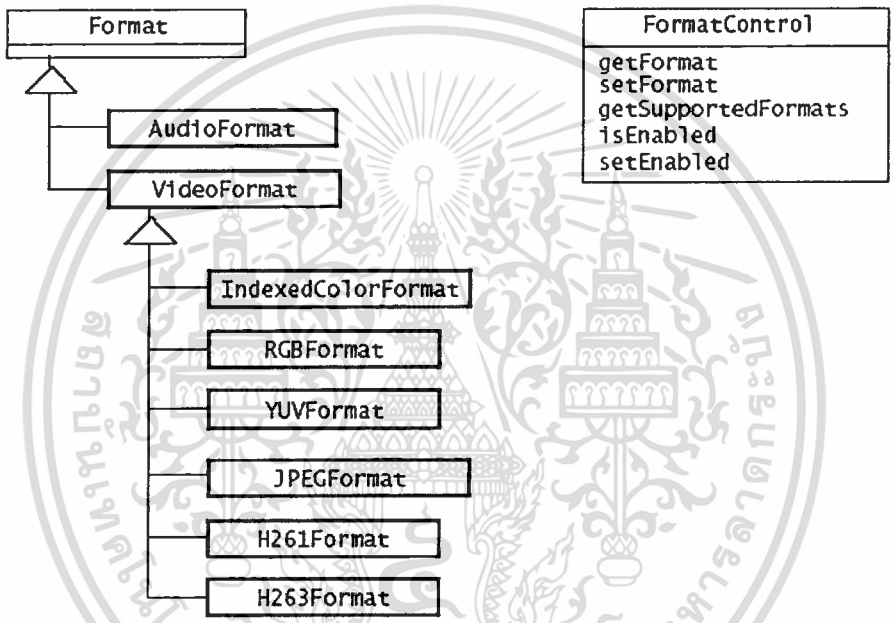
DataSink

DataSink เป็น interfaceพื้นฐานสำหรับ objects ที่ใช้สำหรับอ่าน content ของ media ที่ส่งมาจาก *DataSource* และ render media ไปยังปลายทาง ตัวอย่างของ *DataSink* ให้ลองพิจารณา file-writer object ที่ทำหน้าที่เก็บ media ให้อยู่ในรูปของไฟล์

Format

Format เป็นตัวที่แสดงถึง format ของ media ด้วยตัวของ *Format* เองนั้นจะไม่มี parameter ในการ encoding ที่เฉพาะเจาะจง หรือ มีข่าวสารเกี่ยวกับ global-timing *Format* จะบอกชื่อของ format ที่ใช้ในการ encoding และชนิดของข้อมูลที่ format นั้นต้องการ

JMF extends *Format* เพื่อที่จะกำหนด format ที่เฉพาะเจาะจงของเสียง และวิดีโอ



รูปที่ 2.27 JMF Media Format

Subclass ของ *Format* จะประกอบด้วย *AudioFormat* และ *VideoFormat* และใน *VideoFormat* ก็จะประกอบด้วย 6 subclass คือ

- IndexedColorFormat
- RGBFormat
- YUVFormat
- JPEGFormat

- H261Format
- H263Format

2.3.3 การสร้าง Player

สิ่งหนึ่งที่สำคัญในการเขียนโปรแกรม JMF คือการสร้าง *Player* โดยการเรียก method **createPlayer()** ของ *Manager* ซึ่ง *Manager* จะใช้ URL ของมีเดียนั้น หรือ *MediaLocator* ที่เราสามารถระบุเพื่อที่จะสร้าง *Player* ที่เหมาะสมได้ จากนั้นเมื่อมี *Player* แล้วเราต้องได้รับ visual components ของ *Player* object เราสามารถที่จะเพิ่ม visual component เหล่านี้ลงไปใน application หรือ applet ก็ได้

ในการแสดง visual component ของ *Player* object จำเป็นต้องทำดังนี้

1. สร้าง visual component โดยการเรียก method getVisualComponent()
2. เพิ่ม visual component ไปที่ application หรือ applet

Method หลาย method ของ *Player* นั้นจะสามารถเรียกได้เมื่อ *Player* อยู่ใน **realized** state เท่านั้น ดังนั้นเพื่อเป็นการยืนยันว่า *Player* อยู่ใน state นี้แล้ว เราสามารถใช้ method **createRealizedPlayer()** ของ *Manager* เพื่อที่จะสร้าง *Player* ซึ่ง method นี้จะอำนวยความสะดวกโดยจะทำการสร้าง *Player* และ realize *Player* ให้ในขั้นตอนเดียวเลย

Method **start()** นั้นจะเรียกได้หลังจากที่ *Player* ได้ถูกสร้างขึ้นเรียบร้อยแล้วแต่ต้องก่อนที่ *Player* จะอยู่ใน **prefetched** state โดยที่ method **start()** นั้นจะพยายามย้าย *Player* ไปยัง **started** state ไม่ว่าในตอนนั้น *Player* จะอยู่ในสถานะใดอยู่ก็ตาม ตัวอย่างเช่น เราสามารถที่จะเรียก method **start()** ภายหลังจากที่สร้าง Player ขึ้นได้เลย โดย method **start()** นั้นจะไปเรียก method อื่นที่จำเป็นเพื่อที่จะให้ *Player* เข้าสู่ **started** state เอง

2.3.4 Capture ข้อมูลมีเดีย

การ capture มีเดียเป็นอีกงานที่สำคัญในการเขียนโปรแกรม JMF คุณสามารถที่จะ capture ข้อมูล media โดยใช้อุปกรณ์ที่ใช้ในการ capture เช่น ไมโครโฟน หรือ กล้องวิดีโอ สิ่งที่ต้องทำเพื่อที่จะ capture ข้อมูล media มีดังต่อไปนี้ คือ

1. ระบุตำแหน่งของอุปกรณ์ที่ใช้ในการ capture ที่เราต้องการใช้โดยการสอบถามผ่านทาง *CaptureDeviceManager*
2. สร้าง *CaptureDeviceInfo* object สำหรับอุปกรณ์นั้น
3. เรียกใช้ *MediaLocator* จาก *CaptureDeviceInfo* object และใช้มันในการสร้าง *DataSource*
4. สร้าง *Player* หรือ *Processor* เพื่อใช้ *DataSource*
5. เริ่มใช้งาน *Player* หรือ *Processor* เพื่อที่เริ่มทำการ capture

เราใช้ *CaptureDeviceManager* เพื่อที่จะเข้าไปใช้งานอุปกรณ์ที่ใช้ในการ capture ที่มีอยู่ในระบบ โดย manager จะทำหน้าที่เหมือนกับหน่วยงานกลางที่มีข้อมูลของอุปกรณ์ที่ใช้ในการ capture ทั้งหมดที่สามารถใช้งานได้ เราสามารถเรียกดูรายชื่อของอุปกรณ์ที่สามารถใช้งานได้ทั้งหมดโดยเรียก method **getDevicelist()** ตัวที่ใช้เรียกแทนอุปกรณ์ที่ใช้ในการ capture คือ *CaptureDeviceInfo* object และเรายังสามารถเรียกใช้ method **getDevice()** ของ *CaptureDeviceManager* เพื่อที่จะเรียกใช้ *CaptureDeviceInfo* สำหรับอุปกรณ์ที่ใช้ในการ capture ที่เฉพาะเจาะจง

ในการใช้อุปกรณ์ที่ใช้ในการ capture ข้อมูลมีเดีย จำเป็นที่จะต้องเรียกใช้ *MediaLocator* ของอุปกรณ์ จาก *CaptureDeviceInfo* object และสามารถที่จะใช้ *MediaLocator* ในการสร้าง *Player* หรือ *Processor* โดยตรงได้ด้วย หรือใช้ *MediaLocator* เพื่อที่จะสร้าง *DataSource* ซึ่งสามารถที่จะใช้เป็น input ไปยัง *Player* หรือ *Processor* ก็ได้ และใช้ method **start()** ของ *Player* หรือ *Processor* เพื่อที่จะเริ่มต้นกระบวนการ capture

2.3.5 การทำงานกับ media stream แบบเวลาจริง

โปรโตคอลสำหรับ Streaming Media

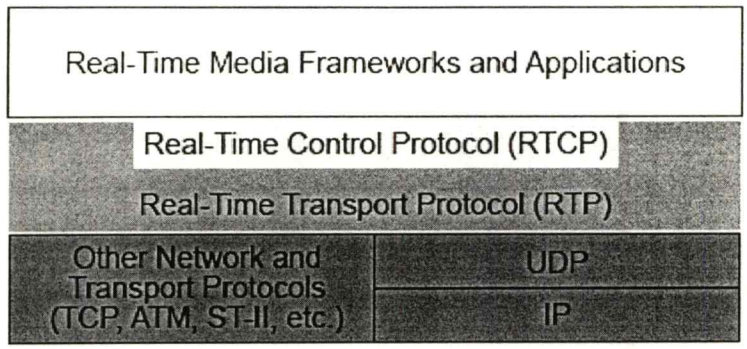
โปรโตคอล HTTP และ FTP จะขึ้นกับ Transmission Control Protocol (TCP) ซึ่ง TCP เป็นโปรโตคอลในชั้น transport ที่ออกแบบมาสำหรับการสื่อสารข้อมูลให้มีความน่าเชื่อถือ เมื่อมี packet สูญหาย หรือ ผิดพลาด จะต้องทำการส่งใหม่อีกครั้งหนึ่ง ทำให้จำนวนโอเวอร์เฮดในการติดต่อสื่อสารที่มีการรับประกันการส่งข้อมูลที่มีความน่าเชื่อถือนั้นมีจำนวนมาก และทำให้อัตราการส่งข้อมูลทั้งหมดช้าลงด้วย

เนื่องจากเหตุผลนี้ โปรโตคอลที่เหมาะสมในการส่ง media stream มากกว่า TCP คือ User Datagram Protocol (UDP) UDP เป็นโปรโตคอลที่ไม่น่าเชื่อถือ มันจะไม่รับประกันว่า packet ที่ส่งไปจะไปถึงยังปลายทาง และ ไม่รับประกันว่า packet ที่ไปถึงปลายทางจะเรียงลำดับอย่างถูกต้องตามที่ส่งไป ทำให้ทางฝั่งผู้รับนั้นต้องสามารถที่จะชดเชยกับข้อมูลที่สูญหาย, packet ที่ซ้ำ, และ packet ที่ไปถึงปลายทางโดยไม่เป็นไปตามลำดับที่ถูกต้อง

UDP เป็นโปรโตคอลที่อยู่ในชั้น transport เช่นเดียวกับ TCP ส่วน Real-Time Transport Protocol (RTP) นั้นกำหนดมาเป็น มาตรฐานของอินเทอร์เน็ตสำหรับการส่งข้อมูลแบบเวลาจริง เช่น ข้อมูลเสียง และ ข้อมูลวิดีโอ

Real-Time Transport Protocol

RTP ให้บริการส่งข้อมูลแบบเวลาจริง โดยส่วนใหญ่ RTP จะใช้งานบน UDP RTP สามารถใช้งานได้ทั้งแบบ unicast และ multicast ถ้าการใช้งานแบบ unicast นั้น ต้นทางจะส่งสำเนาของข้อมูลไปยังปลายทางแต่ละตัวโดยแยกจากกัน แต่ถ้าการใช้งานแบบ multicast นั้น ต้นทางจะส่งข้อมูลมาเพียงชุดเดียวเท่านั้น และ เครือข่ายจะมีหน้าที่ในการส่งข้อมูลนั้นไปยังตำแหน่งต่างๆเอง Multicasting มีประสิทธิภาพมากกว่าใน multimedia application หลายประเภท เช่น video conference



รูปที่ 2.28 สถาปัตยกรรมของ RTP

การให้บริการของ RTP

RTP ทำให้สามารถที่จะระบุชนิดของข้อมูลที่ถูกส่ง, ตัดสินใจว่าควรให้ลำดับของ packet ของข้อมูลอันไหนแสดงผล และ ทำให้ media stream ที่มาจากแต่ละแหล่งสอดคล้องกัน

Packet ของข้อมูล RTP นั้นไม่รับประกันว่าจะมาถึงปลายทางในลำดับที่ถูกต้องตามที่ฝั่งส่งส่งมา และ ไม่ได้รับประกันด้วยว่า packet ทุก packet จะมาถึงปลายทาง ทางฝั่งรับต้องทำการจัดลำดับของ packet ที่มาถึงเอง และ ตรวจจับ packet ที่สูญหายได้เองโดยใช้ข้อมูลที่มีให้ใน packet header

ในขณะที่ RTP นั้นไม่มีกลไกที่จะทำให้มั่นใจได้ว่าการส่งข้อมูลนั้นจะส่งได้ทันเวลา หรือไม่มีการรับประกันคุณภาพในการให้บริการใดๆเลย ดังนั้นจึงมีการเพิ่มโปรโตคอลที่ใช้ในการควบคุมเข้าไป คือ Real-Time Control Protocol (RTCP) ทำให้สามารถวัดคุณภาพของการกระจายข้อมูลได้

สถาปัตยกรรม RTP (นิรมล เริงวิทย์. 2543)

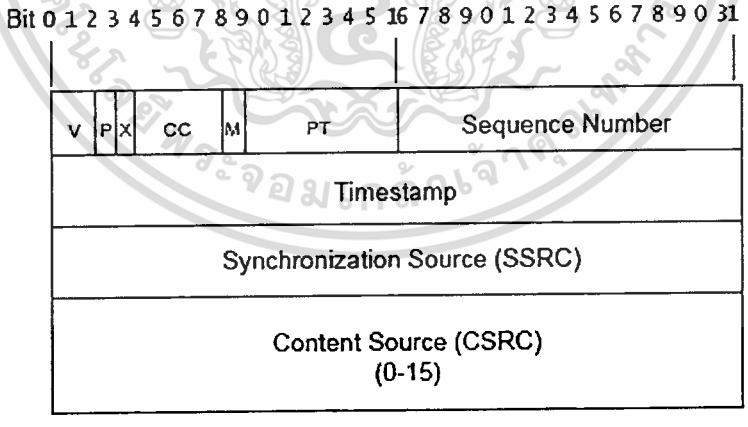
RTP session เป็นตัวเชื่อมระหว่างกลุ่มของ application ที่ติดต่อสื่อสารกันด้วย RTP โดย session นั้นจะถูกระบุด้วย network address คู่กับ หมายเลขพอร์ต พอร์ตหนึ่งสามารถถูกใช้สำหรับข้อมูลมีเดีย และ อีกพอร์ตหนึ่งสามารถถูกใช้สำหรับข้อมูลในการควบคุม (RTCP)

Participant คือ อุปกรณ์, host หรือ user เพียงตัวเดียวที่มาติดต่อร่วมกันใน session การติดต่อร่วมกันใน session นั้นจะประกอบไปด้วยผู้ที่ทำหน้าที่รับข้อมูล (ผู้รับ) และ ผู้ที่ทำหน้าที่ส่งข้อมูล (ผู้ส่ง)

ชนิดของมีเดียแต่ละชนิดจะถูกส่งใน session ที่แตกต่างกัน ตัวอย่าง เช่น ถ้าทั้งเสียง และวิดีโอ ถูกใช้ในการประชุม จะมี session หนึ่งเอาไว้สำหรับการส่งข้อมูลเสียง และอีก session หนึ่งเอาไว้สำหรับการส่งข้อมูลวิดีโอ การส่งข้อมูลในลักษณะนี้ทำให้ฝั่งผู้รับสามารถที่จะเลือกว่าชนิดของมีเดียอันไหนที่ต้องการจะรับ เพราะฝั่งผู้รับบางคนนั้นอาจมีการเชื่อมต่อกับเครือข่ายที่มี bandwidth ต่ำอาจจะต้องการรับเพียงข้อมูลเสียงเท่านั้นก็จะสามารถทำได้

1. **Data Packets**

ข้อมูลมีเดียสำหรับ session จะถูกส่งในลักษณะของข้อมูลที่ต่อเนื่องกัน packet ของข้อมูลที่ต้อเนื่องกันนั้นจะถูกสร้างมาจากต้นทางในลักษณะของ RTP stream แต่ละ packet ข้อมูล RTP ใน stream นั้นประกอบไปด้วย 2 ส่วน คือ ส่วนเฮดเดอร์ที่เป็นโครงสร้าง และ ข้อมูลจริงๆ (payload ของ packet)



รูปที่ 2.29 รูปแบบของข้อมูล RTP และเฮดเดอร์ของ packet

เฮดเดอร์ของ packet ข้อมูล RTP นั้น ประกอบด้วย

- หมายเลขเวอร์ชัน RTP : 2 bits
  เป็นตัวระบุเวอร์ชันของ RTP ซึ่งปัจจุบันมีค่าเป็น 2

- Padding : 1 bit
  ไว้ระบุว่าข้อมูลชุดนี้มี padding หรือไม่ ถ้า field นี้มีค่าเป็น 1 แสดงว่ามี padding ซึ่งค่า padding นี้จะถูกบรรจุมาหลังส่วนข้อมูลจริง (payload) โดยจะมีไบต์หลังสุดของ padding เป็นตัวระบุว่ามี padding มาด้วยกี่ไบต์ padding นี้ต้องใช้ในการเข้ารหัสข้อมูลบางประเภท ที่ระบุขนาดบล็อกของข้อมูลมาเพื่อบรรจุข้อมูลทีละหลายๆ RTP packet

- Extension (X) : 1 bit
  ถ้าบิต extension นี้เป็น 1 หมายความว่าเฮดเดอร์จะถูกกำหนดตายตัวโดยเฮดเดอร์ extension แบบพิเศษซึ่งถูกระบุโดยเฉพาะ

- CRSC count (CC) : 4 bits
  ค่าใน field นี้จะถูกระบุ ถ้ามี field CRSC โดยใช้ระบุจำนวนของ CSRC

- Marker (M) : 1 bit
  ค่าในบิตนี้มีไว้เพื่อ mark packet ที่ต้องการได้ ซึ่งจะถูกระบุค่าโดย profile โดยที่ค่าที่ Marker bit นี้อาจจะมีค่า หรือ ไม่มีก็ได้ แล้วแต่ profile

- Payload Type (PT) : 7 bits
  บรรจุชนิดของการ encode

- Sequence Number : 16 bits
  เลขลำดับประจำ RTP packet โดยจะเพิ่มค่าขึ้นครั้งละ 1 ต่อ การส่ง RTP packet 1 ครั้ง มีไว้เพื่อตรวจสอบการซ้ำ, สูญหาย, และเรียงลำดับ packet ที่ส่งเข้ามาได้

- Timestamp : 32 bits

  ระบุค่าที่ได้จากการสุ่มไบต์แรกของ packet ข้อมูล RTP

- Synchronization Source Identifier (SSRC) :32 bits

  ใช้ในการระบุ synchronization source โดยค่า SSRC นี้จะได้มาจากการสุ่มตัวอย่าง แต่มีเงื่อนไขว่า ใน RTP session หนึ่งๆ ต้องมีค่า SSRC ที่ไม่ซ้ำกัน

- CSRC list : 32 bits

  ใช้ระบุ Contributing source สำหรับ packet นั้นๆ โดย Contributing packet จะถูกสร้างโดย mixer

2. Control Packets

นอกเหนือจากข้อมูลมีเดียสำหรับ session แล้ว packet ข้อมูลควบคุม (RTCP) ยังถูกส่งไปเช่นกัน โดยส่งไปหาผู้ที่เข้าร่วมติดต่อทุกคนใน session RTCP packet นั้นจะประกอบไปด้วยข้อมูลเกี่ยวกับต้นทางที่ส่งข้อมูลมีเดียมาบนพอร์ตข้อมูล, และมีข้อมูลทางสถิติที่เกี่ยวข้องกับข้อมูลที่ถูกส่งมา

Packet ของ RTCP มีหลายชนิดด้วยกัน คือ

- Sender Report (SR)

  เป็นรายงานเกี่ยวกับสถิติการส่งข้อมูล และ การได้รับข้อมูลของผู้ส่ง

- Receiver Report (RR)

  เป็นรายงานที่ฝ่ายผู้รับส่งมาเพื่อแสดงสถิติการได้รับข้อมูลของตน

- Source Description (SDES)

  SDES packet ประกอบด้วย SDES header และ chunk ที่อธิบายต้นกำเนิดหลายๆ chunk แต่ละ chunk จะบรรจุตัวกำหนด SSRC/CSRC และ ชุดของ SDES item ซึ่ง SDES item เอง

ประกอบด้วย type code (8 bits), ความยาว field (8 bits), และ SSRC/CSRC chunk จำนวนมาก บรรจุอยู่ใน SDES packet

- BYE
  ระบุกับปลายทางว่าตนต้องการปิดการส่งข้อมูลแล้ว

- Application-specific
  ใช้ระบุฟังก์ชั่นพิเศษเฉพาะงาน

RTP Applications

RTP applications ส่วนใหญ่จะแบ่งออกเป็น application ที่ต้องการรับข้อมูลจากเครือข่าย (RTP Clients) และ application ที่ต้องการส่งข้อมูลผ่านทางเครือข่าย (RTP Server) บาง application ก็ต้องการทำหน้าที่ทั้ง 2 อย่าง เช่น conferencing application ต้องการ capture และส่งข้อมูล และในเวลาเดียวกันมันก็ต้องการที่จะรับข้อมูลจากเครือข่ายด้วยเช่นกัน

การรับ media stream จากเครือข่าย

Application หลายชนิดจำเป็นต้องรับ RTP stream ได้ เช่น

- Conferencing application ต้องการรับ media stream จาก RTP session และ render media stream นั้นที่ส่วนแสดงผล

- เครื่องตอบรับโทรศัพท์จำเป็นต้องรับ media stream จาก RTP session และเก็บ media stream นั้นเอาไว้ในรูปของไฟล์

- Application ที่ทำหน้าที่บันทึกการสนทนา หรือ การประชุม จำเป็นต้องเก็บ media stream จาก RTP session และเก็บ media stream ไว้ในรูปของไฟล์

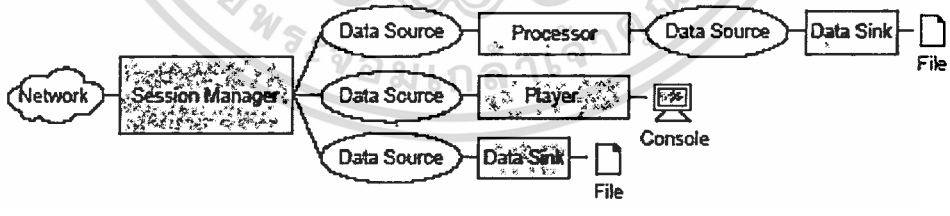
การส่ง media stream ผ่านเครือข่าย

Application ที่ทำหน้าที่เป็น RTP server นั้น ต้องการที่จะส่ง media stream ที่ capture มา หรือ media stream ที่เก็บไว้ ผ่านทางเครือข่าย เช่น ใน conferencing application media stream จะถูก capture มาจากกล้องวิดีโอ และ ส่งออกไปที่ RTP session หนึ่ง session หรือ มากกว่านั้น media stream อาจต้องถูกเข้ารหัสให้อยู่ในรูปแบบมีเดียที่แตกต่างกันไป และส่งออกไปยัง RTP session จำนวนหนึ่งเพื่อทำให้ฝั่งผู้รับที่แตกต่างกันสามารถที่จะประชุมกันได้ การประชุมแบบ multiparty นั้นสามารถ implement ได้โดยไม่ต้องใช้ IP multicast แต่ใช้ unicast RTP session หลายๆ session แทน

2.3.6 JMF RTP API

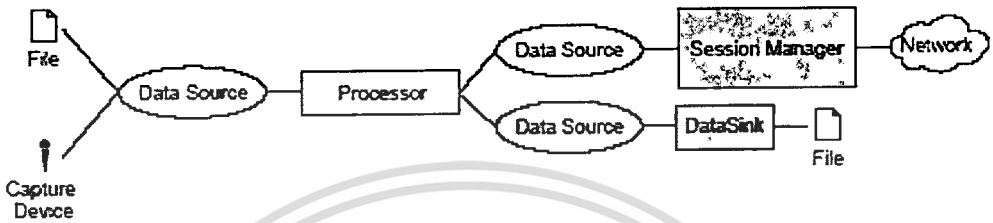
JMF ทำให้สามารถแสดงผล และ ส่ง RTP stream ผ่าน API ที่ถูกกำหนดใน javax.media.rtp, javax.media.rtp.event, และ javax.media.rtp.rtcp JMF ยังสามารถขยายให้สามารถรองรับรูปแบบ RTP ที่เฉพาะเจาะจงเพิ่มเติม โดยผ่านมาตรฐาน JMF plug-in

เราสามารถจะแสดง RTP stream ที่เข้ามา, เซฟ RTP stream ลงไฟล์, หรือ ทั้งสองอย่าง ตัวอย่าง เช่น RTP API สามารถใช้เพื่อ implement telephony application ที่สามารถตอบรับโทรศัพท์ และ สามารถบันทึกข้อความได้เช่นกัน



รูปที่ 2.30 การรับ RTP stream

RTP API สามารถใช้ในการส่ง media stream ที่ได้จากการ capture หรือ ที่เก็บเอาไว้ผ่านเครือข่ายได้ โดย RTP stream ที่ส่งออกไปนั้นอาจจะได้มาจากไฟล์ หรือ ได้มาจากอุปกรณ์ที่ใช้ในการ capture และ RTP stream ที่ส่งออกไปนั้นก็สามารถนำไปแสดงที่ปลายทาง, เก็บไว้ในไฟล์ หรือ ทั้งสองอย่าง

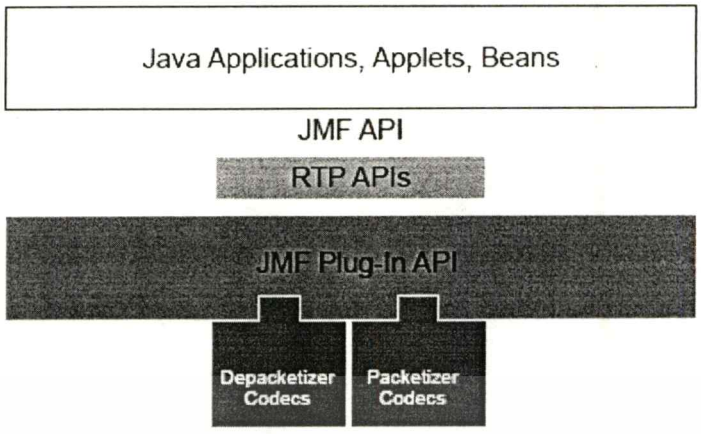


รูปที่ 2.31 การส่ง RTP stream

ตัวอย่าง เช่น เราสามารถ implement video conference application ได้โดยการ capture ข้อมูลเสียง และ วิดีโอ จากนั้นทำการส่งข้อมูลนั้นผ่านทางเครือข่ายโดยใช้ RTP session สำหรับแต่ละชนิดของ media แยกกันไป ซึ่งเราก็สามารถที่จะบันทึกการประชุมนั้น หรือ ใช้ข้อมูลเสียงที่มีการบันทึกเอาไว้ก่อนเป็น เสียงเพลงในขณะรอสาย ใน conference application ก็ได้

สถาปัตยกรรม RTP

JMF RTP API ถูกออกแบบมาให้ทำงานเกี่ยวกับการ capture, แสดงผล, และ ประมวลผล ของ JMF เราสามารถส่ง media stream ที่ได้มาจากการ capture ของอุปกรณ์ที่ใช้ในการ capture โดยใช้ *DataSource* หรือ media stream ที่เก็บเอาไว้ในรูปของไฟล์โดยใช้ *DataSink* JMF สามารถรองรับรูปแบบ RTP และ payload ที่นอกเหนือจากมาตรฐานที่กำหนดไว้โดยใช้ JMF plug-In API



รูปที่ 2.32 สถาปัตยกรรม JMF RTP ระดับสูง

ในการทำงานที่เกี่ยวข้องกับ RTP session นั้นจะมี manager ที่คอยดูแล คือ *SessionManager* ซึ่งมันจะทำหน้าที่เก็บ track ของ ผู้ที่ร่วมติดต่อใน session และ stream ที่ถูกส่งไปด้วย นอกจากนั้นยังมีหน้าที่จัดการเกี่ยวกับ RTCP control channel และ รองรับ RTCP สำหรับทั้งทางฝั่งผู้ส่ง และ ฝั่งผู้รับ

2.3.7 การส่ง RTP Media Stream

ในการส่ง RTP stream นั้น จำเป็นต้องใช้ *Processor* เพื่อที่จะผลิต RTP-encoded *DataSource* และ สร้าง *SessionManager* หรือ *DataSink* เพื่อที่จะควบคุมการส่งข้อมูล

Input ของ *Processor* นั้นสามารถเป็นได้ทั้งข้อมูลที่เก็บบันทึกไว้ หรือ เป็นข้อมูลที่ได้จากการ capture ในขณะนั้นเลย สำหรับข้อมูลที่เก็บบันทึกเอาไว้ก่อนนั้น เราสามารถที่จะใช้ *MediaLocator* ในการระบุไฟล์ตอนที่เราสร้าง *Processor* ส่วนข้อมูลที่ได้จากการ capture ในขณะนั้น *DataSource* จะถูกใช้เป็น input ของ *Processor*

วิธีที่ใช้ในการส่ง RTP stream มีอยู่ 2 วิธี คือ

- ใช้ *MediaLocator* ที่มี parameter ของ RTP session เพื่อที่จะสร้าง RTP *DataSink* โดยการเรียกใช้ *Manager.CreateDataSink*

- ใช้ session manager เพื่อสร้าง stream ที่จะส่งสำหรับข้อมูลนั้น และ ควบคุมการส่ง stream หลังจากที่ดึง ouput *DataSource* คืนมาจาก *Processor* แล้วให้เรียก method **createSendStream()** และ **startSession()** ของ *SessionManager*

ถ้าใช้ *MediaLocator* ในการสร้าง RTP *DataSink* เราจะทำได้แค่เพียงส่ง stream แรกใน *DataSource* แต่ถ้าเราต้องการส่ง RTP stream หลายๆ stream ในหนึ่ง session หรือ ต้องการที่จะดูแลสถิติของ session เราจะเป็นต้องใช้ *SessionManager* โดยตรง

ถ้าไม่คำนึงถึงวิธีในการเลือกเพื่อที่จะส่ง RTP stream สิ่งที่เราจำเป็นต้องทำ คือ

1. สร้าง *Processor* ของ *DataSource* ที่เป็นตัวแทนข้อมูลที่เราต้องการส่ง
2. เตรียมพร้อม *Processor* เพื่อที่จะ output ข้อมูล RTP-encoded
3. รับเอา output จาก *Processor* ในรูปของ *DataSource*

2.3.8 การรับ และ แสดงผล RTP Media Streams

JMF Player และ Processor เป็นตัวใช้ในการแสดงผล, capture, และ เป็นเครื่องมือที่ใช้ในการแปลงข้อมูลสำหรับ RTP streams

รูปที่ 2.33 Data flow ของการรับ RTP

Player คนละตัวถูกใช้สำหรับแต่ละ stream ที่ได้รับมาจาก session manager เราสามารถสร้าง Player สำหรับ RTP stream ผ่านทาง method createPlayer() ของ Manager เราสามารถทำได้ทั้ง 2 ทาง คือ

- ใช้ *MediaLocator* ที่มี parameter ของ RTP session และ สร้าง *Player* โดยเรียก method **createPlayer(MediaLocator)** ของ *Manager*

- สร้าง *Player* สำหรับ *ReceiveStream* โดยเฉพาะ โดยการดึง *DataSource* จาก stream และส่งไปให้ method **createPlayer(DataSource)** ของ *Manager*

ถ้าเรารู้ตัว *MediaLocator* เราจะใช้ทางเลือกแรก แต่ถ้าเรามี data source เราก็ควรใช้ทางเลือกที่สอง ซึ่งทั้งสองวิธีนั้น *DataSource* จะมาจาก *SessionManager* ที่รับข้อมูล media แบบเวลาจริงมาจากเครือข่าย

ถ้าเราใช้ *MediaLocator* เพื่อสร้าง *Player* เราจะสามารถแสดง RTP stream ได้แค่เพียง stream แรกเท่านั้นที่ถูกตรวจจับได้ใน session แต่ถ้าเราต้องการเล่น RTP stream หลาย stream ใน session เราจำเป็นต้องใช้ *SessionManager* โดยตรง และ สร้าง *Player* สำหรับแต่ละ *RecieveStream*

บทที่ 3

การวิเคราะห์และการออกแบบโปรแกรม

3.1 Voice and Video Communication over IP System

ระบบการสื่อสารด้วยเสียงและวิดีโอผ่านทางเครือข่ายอินเทอร์เน็ตนั้น เป็นระบบที่ทำให้ผู้ใช้งานสามารถติดต่อสื่อสารกันไม่ว่าจะห่างไกลเพียงใดก็ตาม ด้วยค่าใช้จ่ายที่ประหยัด และยังสามารถมองเห็นคู่สนทนาได้ด้วย โดยจะสื่อสารผ่านทางเครือข่ายอินเทอร์เน็ต

3.2 ขั้นตอนการดำเนินการพัฒนาระบบ

ประกอบไปด้วยขั้นตอนการดำเนินงานดังต่อไปนี้ คือ

3.2.1 ความต้องการของระบบ

3.2.1.1 ความต้องการในระดับสูง

- ระบบต้องสามารถส่งข้อมูลโดยใช้โปรโตคอลอินเทอร์เน็ต
- ระบบต้องสามารถใช้งานได้กับการให้บริการอินเทอร์เน็ตในปัจจุบัน
- ระบบไม่ควรขึ้นกับระบบปฏิบัติการ
- ระบบไม่ควรขึ้นกับ component ที่เป็นกรรมสิทธิ์ของ third party
- ระบบควรใช้ฮาร์ดแวร์ และ ซอฟท์แวร์ที่เป็นมาตรฐาน
- ระบบควรนำไปใช้ได้กับทั้ง LAN และ WAN
- ระบบไม่ควรให้ผู้ใช้งาน configure มาก

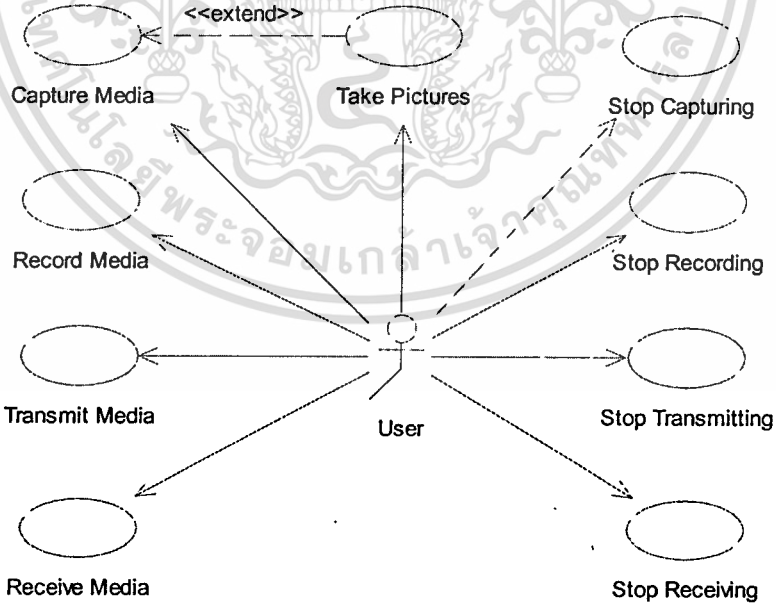
3.2.1.2 ความต้องการในการปฏิบัติงานของระบบ

- ระบบสามารถรับภาพ และ เสียงจากอุปกรณ์ที่ใช้ในการรับข้อมูล คือ กล้อง web cam และไมโครโฟน และ สามารถแสดงผลมูลที่ได้ออกมา
- ระบบสามารถบันทึกภาพ, เสียง และ วิดีโอของข้อมูลที่ได้จากอุปกรณ์ที่ใช้ในการรับข้อมูล
- ระบบสามารถรับส่งเสียงและวิดีโอผ่านเครือข่ายอินเทอร์เน็ต โดยจะพัฒนาในลักษณะการใช้งานแบบ Point to Point
- ระบบสามารถนำข้อมูลที่ได้รับจากคู่สนทนามาบันทึกเป็นไฟล์ และ แสดงผลออกมาได้
- ผู้ใช้งานต้องรู้หมายเลข IP ของผู้ร่วมสนทนาเพื่อให้สามารถติดต่อสื่อสารกันได้ .

3.2.2 การออกแบบ

Diagram ต่างๆของ UML ที่นำมาใช้ในการออกแบบ คือ

3.2.2.1 Use Case Diagram Voice and Video Communication over IP System



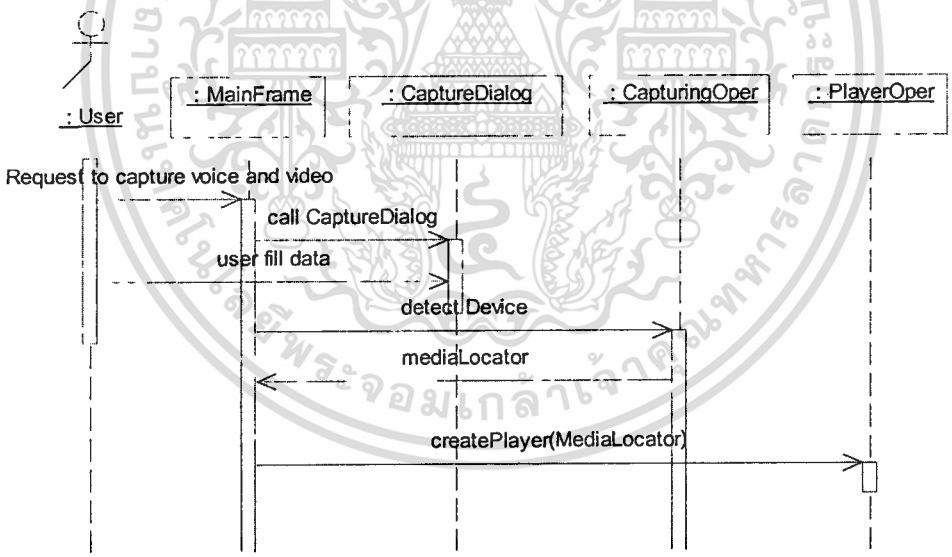
รูปที่ 3.1 Use Case Diagram Voice and Video Communication over IP System

ระบบประกอบด้วยยูสเคสต่างๆ ดังนี้

1. Capture Media ทำหน้าที่ capture มีเดีย
2. Take Pictures ทำหน้าที่ถ่ายภาพจากมีเดีย
3. Record Media ทำหน้าที่ในการบันทึกมีเดีย
4. Transmit Media ทำหน้าที่ในการส่งมีเดียไปยังปลายทาง
5. Receive Media ทำหน้าที่ในการรับมีเดียที่ส่งมาจากต้นทาง
6. Stop Capturing ทำหน้าที่ในการหยุดการ capture มีเดีย
7. Stop Recording ทำหน้าที่ในการหยุดการบันทึกมีเดีย
8. Stop Transmitting ทำหน้าที่ในการหยุดการส่งมีเดียไปยังปลายทาง
9. Stop Receiving ทำหน้าที่ในการหยุดการรับมีเดียจากต้นทาง

3.2.2.2 Sequence Diagram Voice and Video over IP System

1. Capture Media

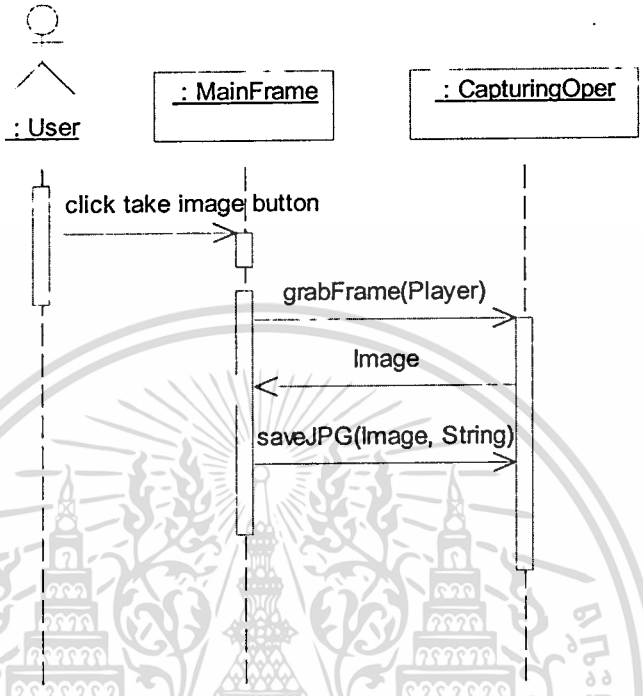


รูปที่ 3.2 Sequence Diagram ของยูสเคส Capture Media

ตารางที่ 3.1 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Capture Media

Use Case	Capture Media
วัตถุประสงค์	ใช้เมื่อผู้ใช้งานต้องการ capture ข้อมูลมีเดียจากกล้อง หรือ ไมโครโฟน และ แสดงผลการตรวจจับผ่านทางจอภาพ หรือ ลำโพง
เงื่อนไขเมื่อเริ่มต้น	-
เมื่อทำงานเสร็จ	ผู้ใช้งานสามารถ capture ข้อมูลมีเดียจากกล้อง หรือ ไมโครโฟน และแสดงผลการตรวจจับผ่านทางจอภาพ หรือ ลำโพงได้
เมื่อทำงานไม่เสร็จ	ผู้ใช้งานไม่สามารถ capture ข้อมูลมีเดียจากกล้อง หรือ ไมโครโฟน และแสดงผลการตรวจจับผ่านทางจอภาพ หรือ ลำโพงได้
Actor ที่เกี่ยวข้อง	User
สิ่งกระตุ้นการทำงาน	User เรียกเมธอดที่ใช้ในการ capture ข้อมูลมีเดีย
Input	สัญญาณวิดีโอ และ สัญญาณเสียง
Output	การแสดงผลข้อมูลมีเดียที่ capture ได้
รายละเอียด	1. ผู้ใช้งานทำการเรียกใช้งานฟังก์ชั่นในการ capture media ในคลาส MainFrame 2. คลาส MainFrame เรียก CaptureDialog เพื่อให้ผู้ใช้งานกรอกข้อมูล 3. เมื่อผู้ใช้งานกรอกข้อมูลเรียบร้อยแล้ว คลาส MainFrame จะเรียกเมธอดในคลาส CapturingOper เพื่อทำการตรวจจับอุปกรณ์ที่รับข้อมูลมีเดียประเภทนั้น และ ส่งค่า MediaLocator กลับคืนมาให้ยังคลาส MainFrame 4. คลาส MainFrame ไปเรียกเมธอดในคลาส PlayerOper เพื่อทำการสร้างตัวแสดงผล โดยส่งข้อมูล MediaLocator ไปเป็น input parameter ของเมธอดนั้น

2. Take Pictures



รูปที่ 3.3 Sequence Diagram ของยูสเคส Take Pictures

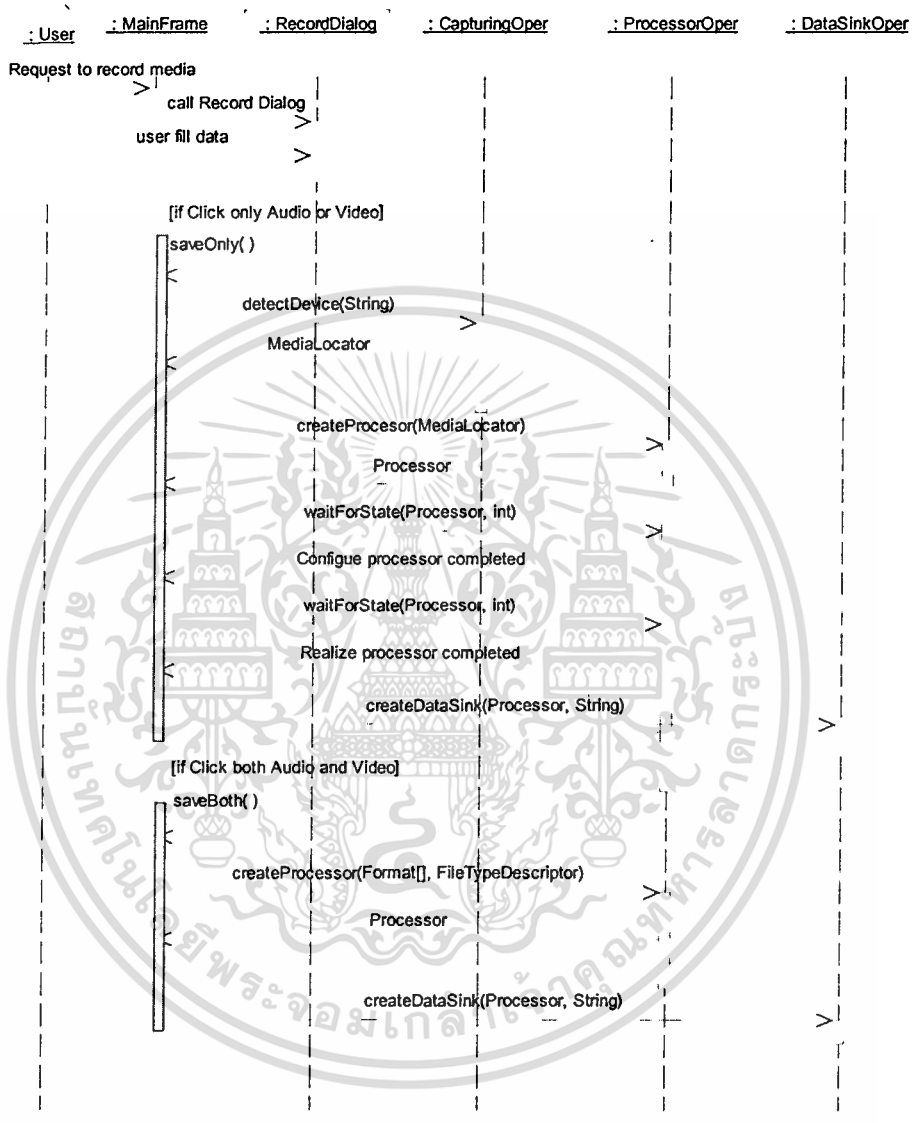
ตารางที่ 3.2 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Take Pictures

Use Case	Takes Pictures
วัตถุประสงค์	ใช้เมื่อผู้ใช้งานต้องการถ่ายภาพนิ่งจากข้อมูลวิดีโอที่แสดงผลอยู่
เงื่อนไขเมื่อเริ่มต้น	มีการแสดงผลข้อมูลวิดีโออยู่ก่อนแล้ว
เมื่อทำงานเสร็จ	ผู้ใช้งานได้รับไฟล์รูปภาพที่เกิดจากการถ่ายภาพ
เมื่อทำงานไม่เสร็จ	ไม่สามารถบันทึกรูปภาพที่ถ่ายไว้เป็นไฟล์ได้
Actor ที่เกี่ยวข้อง	User
สิ่งกระตุ้นการทำงาน	User เรียกเมธอดที่ใช้ในการถ่ายภาพนิ่ง
Input	ข้อมูลวิดีโอที่กำลังแสดงผลอยู่
Output	ไฟล์รูปภาพที่ได้จากข้อมูลวิดีโอที่แสดงผลอยู่

ตารางที่ 3.2 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Take Pictures (ต่อ)

รายละเอียด	1. ในขณะที่ข้อมูลวิดีโอกำลังแสดงผลอยู่นั้น ถ้าผู้ใช้งานต้องการบันทึกภาพนิ่งในขณะนั้น 2. คลาส MainFrame ไปเรียกเมธอดในคลาส CapturingOper ให้ทำการเก็บเฟรมในเวลานั้นไว้ โดยส่งข้อมูล player ที่กำลังใช้ในการแสดงผลข้อมูลวิดีโอ ไปเป็น input parameter ของเมธอดนั้น ซึ่งเมธอดนั้นจะให้ข้อมูลที่เป็นภาพนิ่งกลับคืนมา 3. จากนั้นต้องนำภาพนิ่งที่ได้รับมานั้น ไปทำการบันทึกลงไฟล์ โดยเรียกเมธอด saveJPG ในคลาส CapturingOper และส่งข้อมูลรูปภาพที่ได้จากขั้นตอนก่อนหน้าไปเป็น input parameter ให้กับเมธอดนี้ เพื่อทำการบันทึกภาพที่ได้รับมาเป็น JPEG ไฟล์

3. Record Media



รูปที่ 3.4 Sequence Diagram ของยูสเคส Record Media

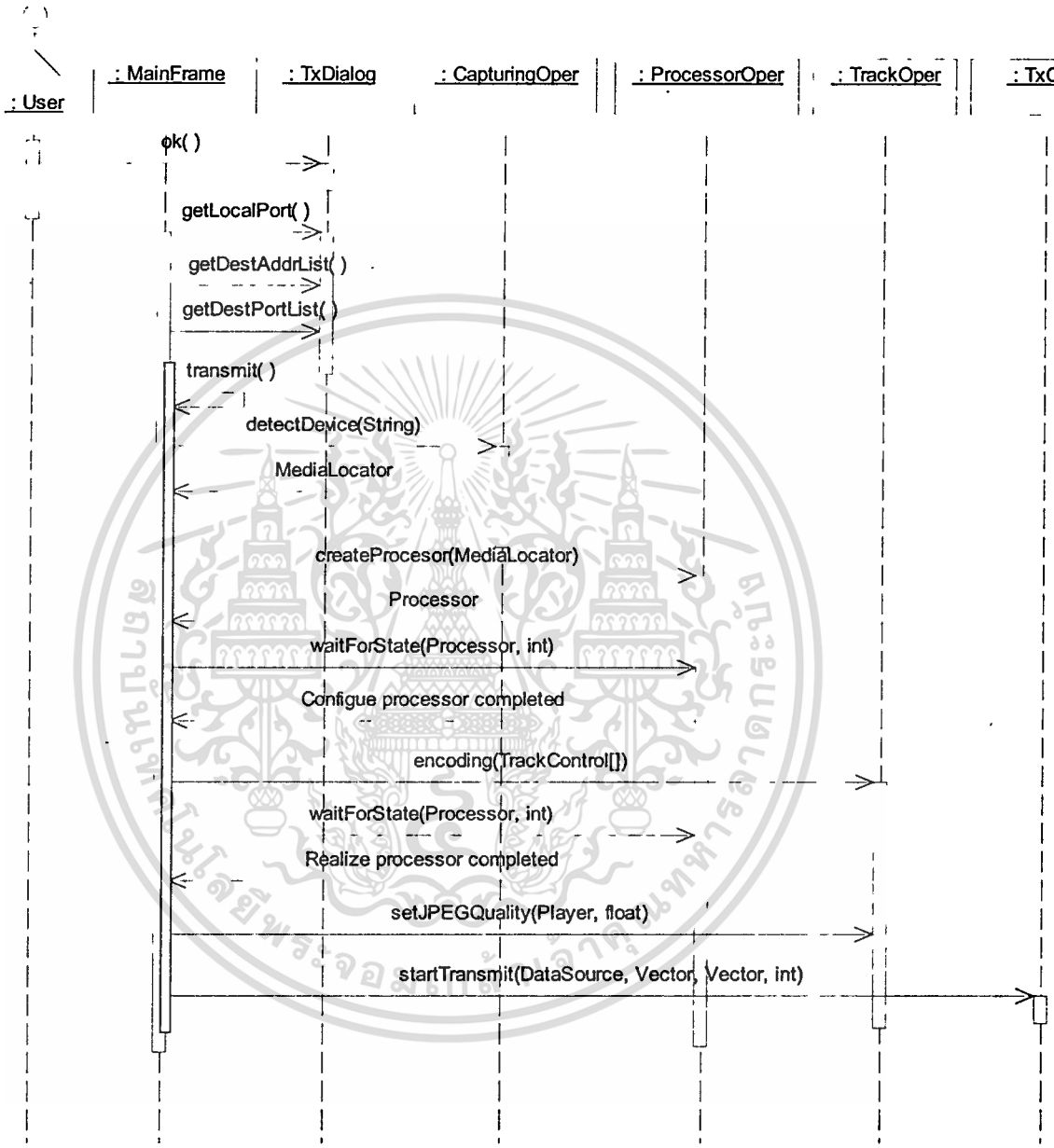
ตารางที่ 3.3 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Record Media

Use Case	Record Media
วัตถุประสงค์	ใช้เมื่อผู้ใช้งานต้องการบันทึกข้อมูลมีเดียที่จับได้จากอุปกรณ์รับสัญญาณ
เงื่อนไขเมื่อเริ่มต้น	-
เมื่อทำงานเสร็จ	ผู้ใช้งานสามารถบันทึกข้อมูลมีเดียที่ได้จากอุปกรณ์รับสัญญาณ
เมื่อทำงานไม่เสร็จ	ผู้ใช้งานไม่สามารถบันทึกข้อมูลมีเดียที่ได้จากอุปกรณ์รับสัญญาณ
Actor ที่เกี่ยวข้อง	User
สิ่งกระตุ้นการทำงาน	User เรียก method ที่ใช้ในการส่งบันทึกข้อมูลมีเดีย
Input	ข้อมูลเสียง หรือ วิดีโอที่อุปกรณ์รับสัญญาณตรวจจับได้
Output	ไฟล์เสียง หรือ วิดีโอที่ได้จากข้อมูลที่รับมากจากอุปกรณ์รับสัญญาณ
รายละเอียด	1. ผู้ใช้งานเลือกฟังก์ชั่นที่ใช้ในการบันทึกข้อมูลมีเดียในคลาส MainFrame 2. คลาส MainFrame เรียก RecordDialog เพื่อให้ผู้ใช้งานใส่ข้อมูลที่จำเป็น 3. คลาส MainFrame ตรวจสอบข้อมูลว่าผู้ใช้งานต้องการบันทึกข้อมูลประเภทใด และต้องการบันทึกลงเป็นไฟล์ชนิดใด 4. ถ้าผู้ใช้งานต้องการบันทึกข้อมูลวิดีโอ หรือ เสียง อย่างใดอย่างหนึ่ง MainFrame จะทำการเรียกเมธอด saveOnly() โดยเมธอดนี้ไปเรียกเมธอด detectDevice ในคลาส CapturingOper เพื่อทำการตรวจหาอุปกรณ์ที่ใช้ในการรับสัญญาณข้อมูล ซึ่งเมธอดจะส่งข้อมูล MediaLocator คืนกลับมาให้ 5. จากนั้นนำข้อมูล MediaLocator ที่ได้รับไปเป็น input parameter ของเมธอด createProcessor ในคลาส ProcessorOper เพื่อทำการสร้าง processor เมธอดนี้จะส่งค่า processor ที่ทำการสร้างคืนกลับมา 6. เปลี่ยนสถานะของ processor ที่ได้รับมาให้อยู่ในสถานะ configured โดยเปลี่ยนผ่านทางเมธอด waitForState 7. เมื่อ processor อยู่ในสถานะ configured เรียบร้อยแล้ว ให้เปลี่ยนสถานะของ processor ไปเป็น realized ผ่านทางเมธอด waitForState 8. จากนั้นทำการสร้าง dataSink ที่ใช้ในการบันทึกข้อมูลลงไฟล์ โดยเรียกใช้เมธอด createDataSink ในคลาส DataSinkOper โดยส่งค่า

ตารางที่ 3.3 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Record Media (ต่อ)

	processor ที่อยู่ในสถานะ realized และชื่อไฟล์ที่ต้องการบันทึกไป เป็น input parameter ของเมธอดนั้น 9. ถ้าผู้ใช้งานต้องการที่จะบันทึกทั้งข้อมูลเสียงและวิดีโอลงในไฟล์เดียวกัน MainFrame จะไปเรียกเมธอด saveBoth() แทน โดยเมธอดนี้ไปเรียกเมธอด createProcessor ในคลาส ProcessorOper โดยส่งข้อมูล format ของเสียง และวิดีโอ และชนิดของไฟล์ที่ต้องการบันทึกไปเป็น input parameter ของเมธอดนั้น โดยเมธอดนั้นจะส่งค่า processor ที่อยู่ในสถานะ realized กลับคืนมา 10. จากนั้นทำการสร้าง dataSink ที่ใช้ในการบันทึกข้อมูลลงไฟล์ โดยเรียกใช้เมธอด createDataSink ในคลาส DataSinkOper โดยส่งค่า processor ที่อยู่ในสถานะ realized และชื่อไฟล์ที่ต้องการบันทึกไป เป็น input parameter ของเมธอดนั้น

4. Transmit Media



รูปที่ 3.5 Sequence Diagram ของยูสเคส Transmit Media

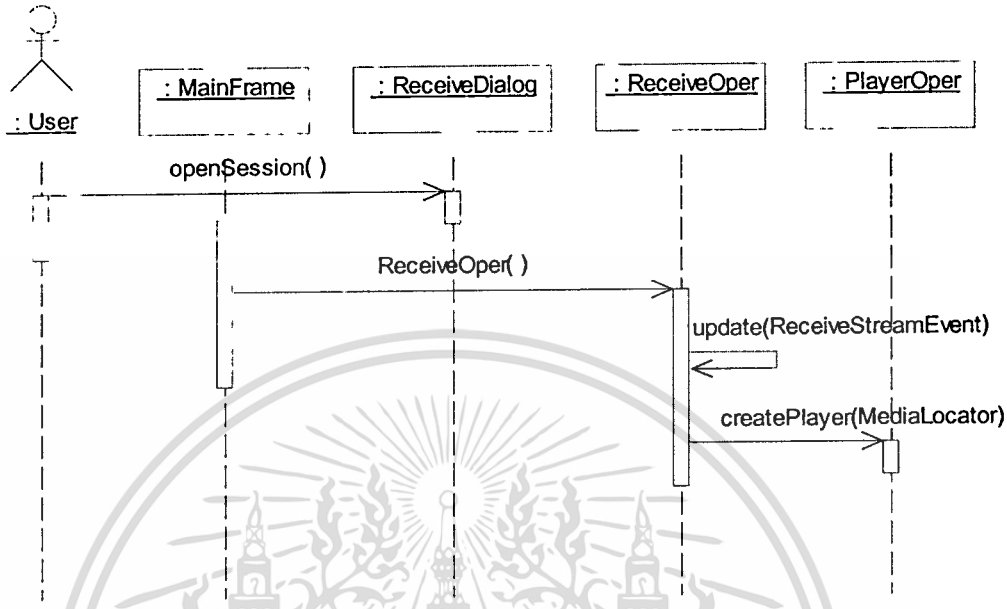
ตารางที่ 3.4 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Transmit Media

Use Case	Transmit Media
วัตถุประสงค์	ใช้เมื่อผู้ใช้งานต้นทางต้องการส่งข้อมูลมีเดียแบบเวลาจริงไปยังปลายทาง

ตารางที่ 3.4 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Transmit Media (ต่อ)

เงื่อนไขเมื่อเริ่มต้น	-
เมื่อทำงานเสร็จ	ต้นทางสามารถส่งข้อมูลมีเดียแบบเวลาจริงไปให้ปลายทางได้
เมื่อทำงานไม่เสร็จ	ต้นทางไม่สามารถส่งข้อมูลมีเดียแบบเวลาจริงไปให้ปลายทางได้
Actor ที่เกี่ยวข้อง	User
สิ่งกระตุ้นการทำงาน	User เรียก method ที่ใช้ในการส่งข้อมูลมีเดียแบบเวลาจริง
Input	Port ของเครื่องต้นทาง และ IP address และ Port ของปลายทาง
Output	ข้อมูลมีเดียที่ส่งไปยังปลายทาง
รายละเอียด	1. ผู้ใช้งานทำการเลือกประเภทของข้อมูลที่ต้องการส่งไปยังปลายทาง และ กำหนดพอร์ตของเครื่องผู้ใช้งาน และ ใส่ IP Address และพอร์ตของคู่สนทนา 2. เมื่อผู้ใช้งานตกลงที่จะส่งข้อมูล คลาส MainFrame จะไปนำประเภทของข้อมูลมีเดียที่ผู้ใช้งานต้องการส่ง หมายเลขพอร์ตของผู้ใช้งาน และ หมายเลข IP และ พอร์ตของคู่สนทนา ที่คลาส TxDialog จากนั้นจะไปเรียกเมธอด transmit 3. ในเมธอด transmit MainFrame จะไปเรียกเมธอด detectDevice ในคลาส CapturingOper และเมธอดนี้จะส่งค่า MediaLocator คืนกลับมาให้ ซึ่งค่า MediaLocator นี้จะนำไปเป็น input parameter ของเมธอด createProcessor ในคลาส ProcessorOper เพื่อทำการสร้าง processor 4. เมื่อได้รับ processor มาต้องเปลี่ยนสถานะของ processor ให้เป็นสถานะ configure โดยผ่านทางเมธอด waitForState 5. เมื่อ processor อยู่ในสถานะ configure แล้ว จะทำการเข้ารหัส track ข้อมูลที่ได้มาจาก processor โดยเรียกใช้เมธอด encoding ในคลาส TrackOper 6. เมื่อทำการเข้ารหัส track ข้อมูลเรียบร้อยแล้ว จะเปลี่ยนสถานะของ processor ไปเป็นสถานะ realized 7. ทำการ set คุณภาพของข้อมูลที่จะทำการส่งไป โดยเรียกใช้เมธอด setJPEGQuality ในคลาส TrackOper จากนั้นจึงทำการส่งข้อมูลแบบเวลาจริงโดยผ่านเมธอด startTransmit ในคลาส TxOper

5. Receive Media



รูปที่ 3.6 Sequence Diagram ของยูสเคส Receive Media

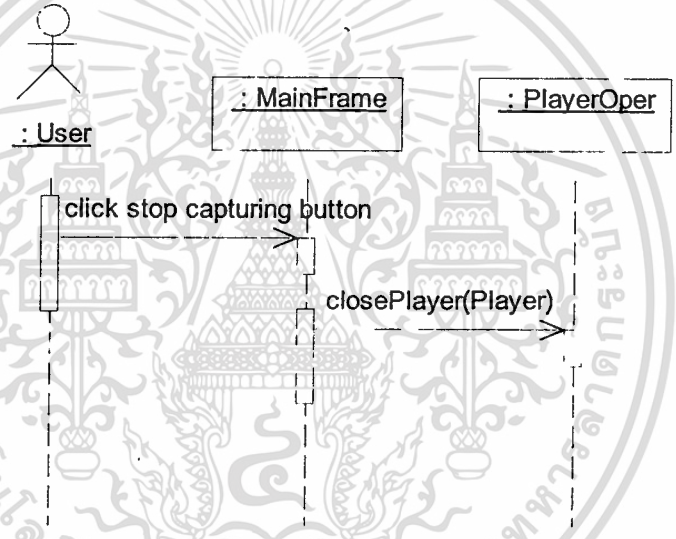
ตารางที่ 3.5 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Receive Media

Use Case	Receive Media
วัตถุประสงค์	ใช้เมื่อผู้ใช้งานต้นทางต้องการส่งข้อมูลมีเดียแบบเวลาจริงไปยังปลายทาง
เงื่อนไขเมื่อเริ่มต้น	เมื่อมีข้อมูลมีเดียส่งมายังเครื่องของผู้ใช้งานและเครื่องผู้ใช้งานมีการเปิด session เพื่อรอรับข้อมูลเอาไว้
เมื่อทำงานเสร็จ	ผู้ใช้งานสามารถรับข้อมูลมีเดียแบบเวลาจริงที่ส่งมาจากต้นทางได้
เมื่อทำงานไม่เสร็จ	ผู้ใช้งานไม่สามารถรับข้อมูลมีเดียแบบเวลาจริงที่ส่งมาจากต้นทางได้
Actor ที่เกี่ยวข้อง	-
สิ่งกระตุ้นการทำงาน	ข้อมูลมีเดียที่ส่งมายังเครื่องของผู้ใช้งาน
Input	IP address และ Port ของปลายทาง
Output	ข้อมูลมีเดียที่แสดงผลผ่านทางหน้าจอ หรือ ลำโพง
รายละเอียด	1. ผู้ใช้งานทำการเปิด session เพื่อเป็นช่องทางที่ใช้รับข้อมมูลมีเดียที่จะส่งเข้ามา

ตารางที่ 3.5 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Receive Media (ต่อ)

	2. MainFrame ไปสร้าง object ของคลาส ReceiveOper เพื่อให้ ReceiveOper เรียกเมธอด update เพื่อคอยดูว่ามี media stream เข้ามาทาง session ที่เปิดรอไว้หรือไม่ ถ้ามี media stream เข้ามา จะไปสร้างตัวแสดงผล โดยใช้เมธอด createPlayer ในคลาส PlayerOper

6. Stop Capturing



รูปที่ 3.7 Sequence Diagram ของยูสเคส Stop Capturing

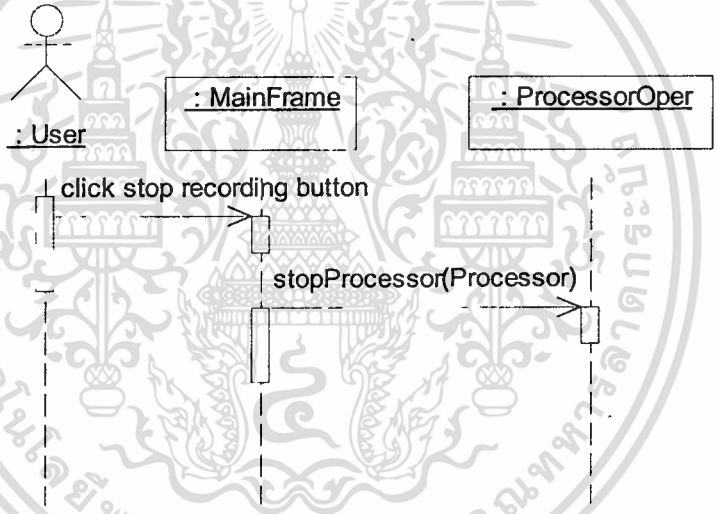
ตารางที่ 3.6 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Stop Capturing

Use Case	Stop Capturing
วัตถุประสงค์	ใช้เมื่อผู้ใช้งานต้องการที่จะหยุดการแสดงผลสัญญาณเสียง หรือ วิดีโอ
เงื่อนไขเมื่อเริ่มต้น	เมื่อมีการแสดงผลข้อมูลเสียง หรือ วิดีโออยู่ก่อนแล้ว
เมื่อทำงานเสร็จ	ผู้ใช้งานสามารถหยุดการแสดงผลสัญญาณเสียง หรือ วิดีโอ
เมื่อทำงานไม่เสร็จ	ผู้ใช้งานไม่สามารถหยุดการแสดงผลสัญญาณเสียง หรือ วิดีโอ
Actor ที่เกี่ยวข้อง	User
สิ่งกระตุ้นการทำงาน	User เรียก method ที่ใช้ในการหยุดการแสดงผลสัญญาณเสียง หรือ วิดีโอ

ตารางที่ 3.6 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Stop Capturing (ต่อ)

Input	-
Output	-
รายละเอียด	1. ผู้ใช้งานกดปุ่ม Stop capturing เมื่อต้องการหยุดการแสดงผลข้อมูล 2. คลาส MainFrame จะไปเรียกเมธอด closePlayer ในคลาส PlayerOper เพื่อทำการปิด และ ทำลายตัวแสดงผล โดยจะส่ง player ที่ต้องการปิด และ ทำลาย ไปเป็น input parameter ของเมธอดนั้นด้วย

7. Stop Recording



รูปที่ 3.8 Sequence Diagram ของยูสเคส Stop Recording

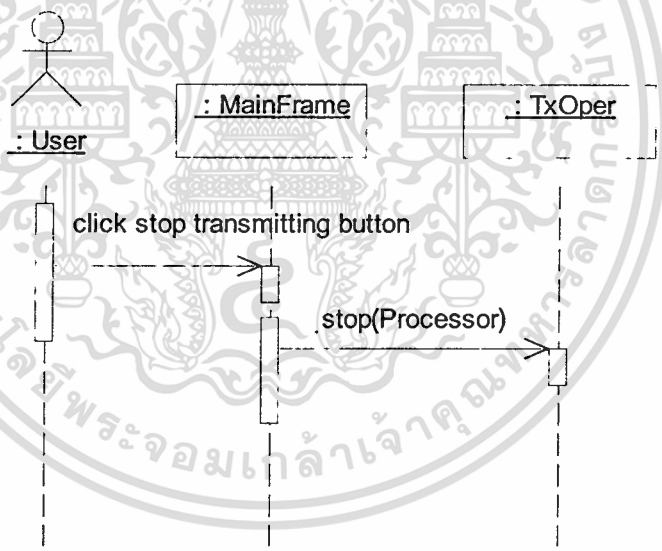
ตารางที่ 3.7 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Stop Recording

Use Case	Stop Recording
วัตถุประสงค์	ใช้เมื่อผู้ใช้งานต้องการต้องการหยุดการบันทึกข้อมูลมีเดียลงไฟล์
เงื่อนไขเมื่อเริ่มต้น	มีการบันทึกข้อมูลมีเดียลงไฟล์อยู่
เมื่อทำงานเสร็จ	สามารถหยุดการบันทึกข้อมูลมีเดียลงไฟล์
เมื่อทำงานไม่เสร็จ	ไม่สามารถหยุดการบันทึกข้อมูลมีเดียลงไฟล์
Actor ที่เกี่ยวข้อง	User

ตารางที่ 3.7 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Stop Recording (ต่อ)

สิ่งกระตุ้นการทำงาน	User เรียก method ที่ใช้ในการหยุดการบันทึกข้อมูล
Input	-
Output	ไฟล์ที่ได้จากการบันทึกข้อมูลมีเดีย
รายละเอียด	1. ผู้ใช้งานกดปุ่ม Stop Recording เมื่อต้องการหยุดการแสดงผลข้อมูล 2. คลาส MainFrame จะไปเรียกเมธอด stopProcessor ในคลาส ProcessorOper เพื่อทำการปิด และ ทำลาย processor โดยจะส่ง processor ที่ต้องการปิด และ ทำลายไปเป็น input parameter ของเมธอดนั้นด้วย

8. Stop Transmitting



รูปที่ 3.9 Sequence Diagram ของยูสเคส Stop Transmitting

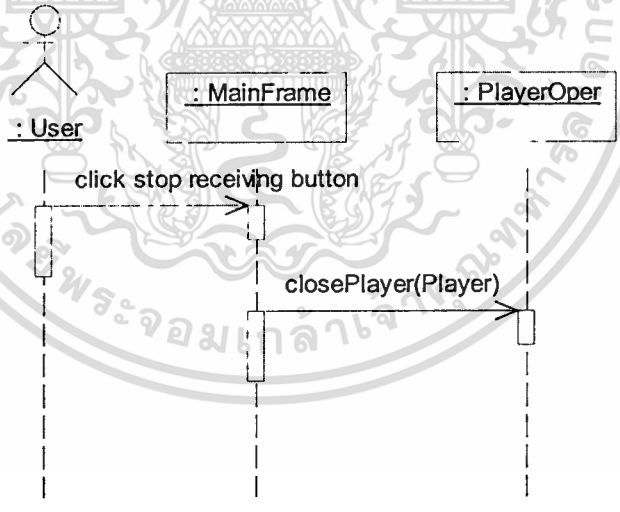
ตารางที่ 3.8 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Stop Transmitting

Use Case	Stop Transmitting
วัตถุประสงค์	ใช้เมื่อผู้ใช้งานต้องการหยุดการส่งข้อมูลมีเดียแบบเวลาจริงไปยังปลายทาง
เงื่อนไขเมื่อเริ่มต้น	มีการส่งข้อมูลมีเดียแบบเวลาจริงไปยังปลายทางอยู่

ตารางที่ 3.8 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Stop Transmitting (ต่อ)

เมื่อทำงานเสร็จ	ผู้ใช้งานสามารถหยุดการส่งข้อมูลมีเดียแบบเวลาจริงไปยังปลายทางได้
เมื่อทำงานไม่เสร็จ	ผู้ใช้งานไม่สามารถหยุดการส่งข้อมูลมีเดียแบบเวลาจริงไปยังปลายทาง
Actor ที่เกี่ยวข้อง	User
สิ่งกระตุ้นการทำงาน	User เรียก method ที่ใช้ในการหยุดส่งข้อมูลมีเดียแบบเวลาจริง
Input	-
Output	-
รายละเอียด	1. ผู้ใช้งานกดปุ่ม Stop Transmiting เมื่อต้องการหยุดการส่งข้อมูลมีเดียแบบเวลาจริงไปยังปลายทาง 2. คลาส MainFrame จะไปเรียกเมธอด stop ในคลาส TxOper เพื่อทำการปิด และ ทำลาย processor โดยจะส่ง processor ที่ต้องการปิด และทำลายไปเป็น input parameter ของเมธอดนั้นด้วย

9. Stop Receiving

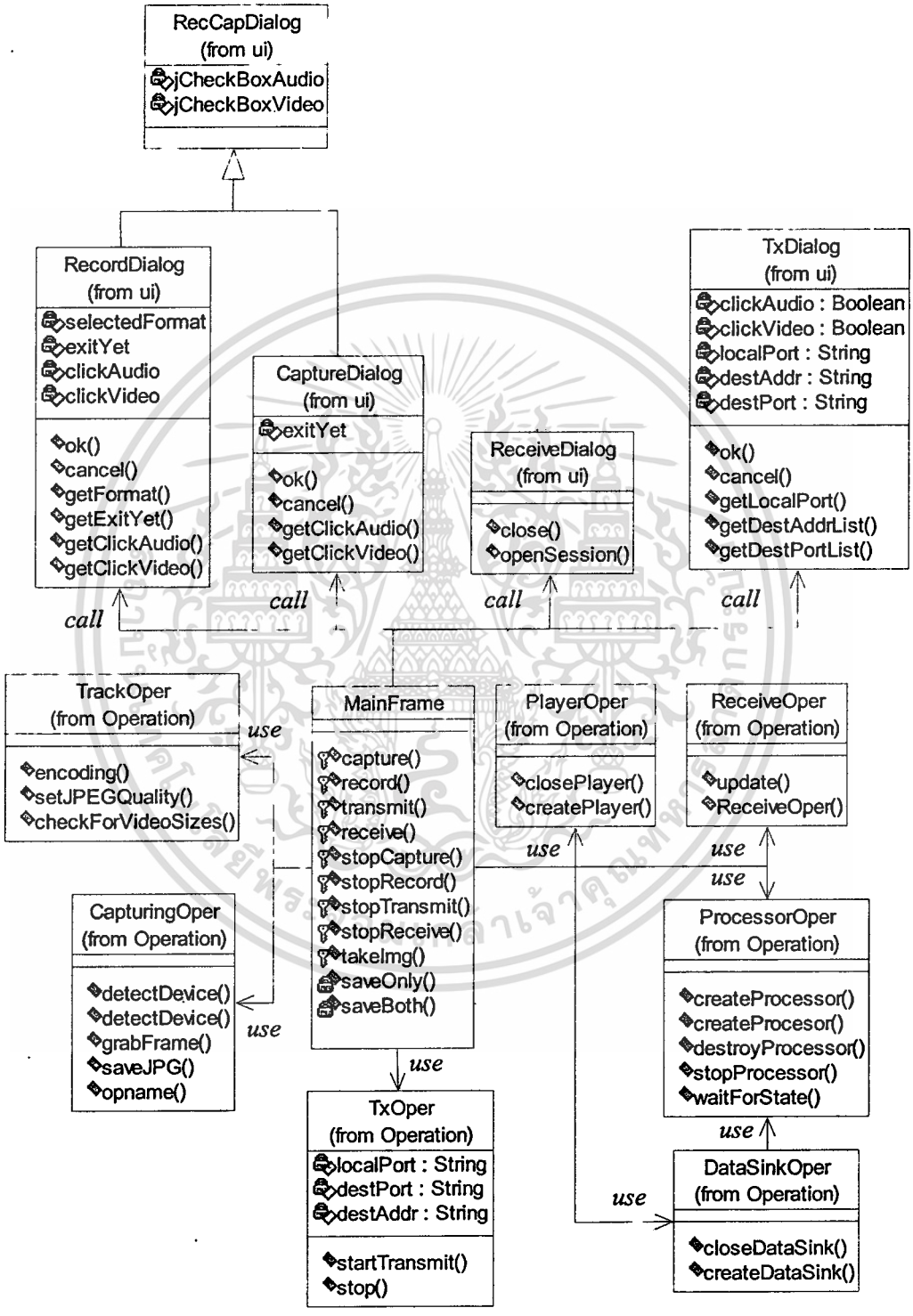


รูปที่ 3.10 Sequence Diagram ของยูสเคส Stop Receiving

ตารางที่ 3.9 รายละเอียดประกอบ Sequence Diagram ของยูสเคส Stop Receiving

Use Case	Stop Receiving
วัตถุประสงค์	ใช้เมื่อผู้ใช้งานปลายทางต้องการหยุดรับข้อมูลมีเดียแบบเวลาจริงจากต้นทาง
เงื่อนไขเมื่อเริ่มต้น	ผู้ใช้งานปลายทางกำลังรับข้อมูลจากผู้ใช้งานต้นทางอยู่
เมื่อทำงานเสร็จ	ผู้ใช้งานปลายทางสามารถหยุดรับข้อมูลมีเดียแบบเวลาจริงจากต้นทาง
เมื่อทำงานไม่เสร็จ	ผู้ใช้งานปลายทางไม่สามารถหยุดรับข้อมูลมีเดียแบบเวลาจริงจากต้นทาง
Actor ที่เกี่ยวข้อง	User
สิ่งกระตุ้นการทำงาน	User เรียก method ที่ใช้ในการหยุดรับข้อมูลมีเดียแบบเวลาจริงจากผู้ใช้งานต้นทาง
Input	-
Output	-
รายละเอียด	1. ผู้ใช้งานกดปุ่ม Stop Receiving เมื่อต้องการหยุดการรับข้อมูลมีเดียแบบเวลาจริงจากต้นทาง 2. คลาส MainFrame จะไปเรียกเมธอด closePlayer ในคลาส PlayerOper เพื่อทำการปิด player โดยจะส่ง player ที่ต้องการปิดไปเป็น input parameter ของเมธอดนั้นด้วย

3.2.2.3 Class Diagram Voice and Video Communication over IP System



รูปที่ 3.11 Class Diagram ของ Voice and Video Communication over IP System

ในระบบประกอบด้วยคลาสทั้งหมด 13 คลาส ซึ่งแยกออกเป็น 2 แพคเกจ คือ

1. ในแพคเกจ ui ประกอบด้วยคลาส 5 คลาส คือ

- RecCapDialog เป็นคลาสที่ใช้เป็นคลาส parent ของคลาส CaptureDialog และ ReceiveDialog
- CaptureDialog เป็นคลาสที่ใช้ในการติดต่อกับผู้ใช้งานเกี่ยวกับการ capture
- ReceiveDialog เป็นคลาสที่ใช้ในการติดต่อกับผู้ใช้งานเกี่ยวกับการรับข้อมูล
- RecordDialog เป็นคลาสที่ใช้ในการติดต่อกับผู้ใช้งานเกี่ยวกับการบันทึกข้อมูล
- TxDialog เป็นคลาสที่ใช้ในการติดต่อกับผู้ใช้งานเกี่ยวกับการส่งข้อมูล

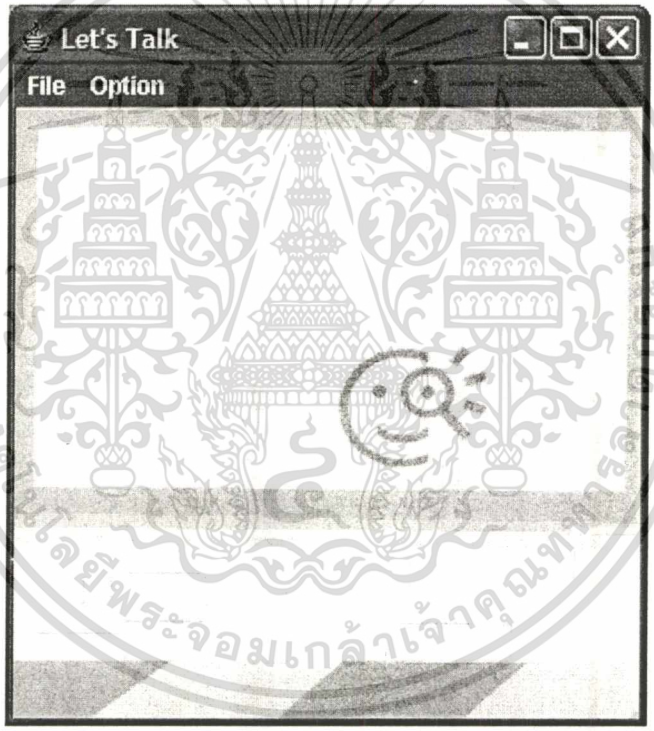
2. ในแพคเกจ operation ประกอบด้วยคลาส 7 คลาส คือ

- CapturingOper เป็นคลาสที่รวมการทำงานเกี่ยวกับการ capture
- DataSinkOper เป็นคลาสที่รวมการทำงานเกี่ยวกับ DataSink
- PlayerOper เป็นคลาสที่รวมการทำงานเกี่ยวกับ Player
- ProcessorOper เป็นคลาสที่รวมการทำงานเกี่ยวกับ Processor
- TrackOper เป็นคลาสที่รวมการทำงานเกี่ยวกับ track ข้อมูล
- TxOper เป็นคลาสที่รวมการทำงานเกี่ยวกับการส่งข้อมูล

และ คลาสอีกหนึ่งคลาสที่ไม่ได้รวมอยู่ในแพคเกจใดเลย คือ คลาส MainFrame ซึ่งเป็นคลาสที่ทำหน้าที่เป็นตัวหลักในการติดต่อกับคลาสอื่นๆ เพื่อให้ทำงานได้ตามที่ระบบต้องการ

บทที่ 4
การใช้งานโปรแกรมประยุกต์

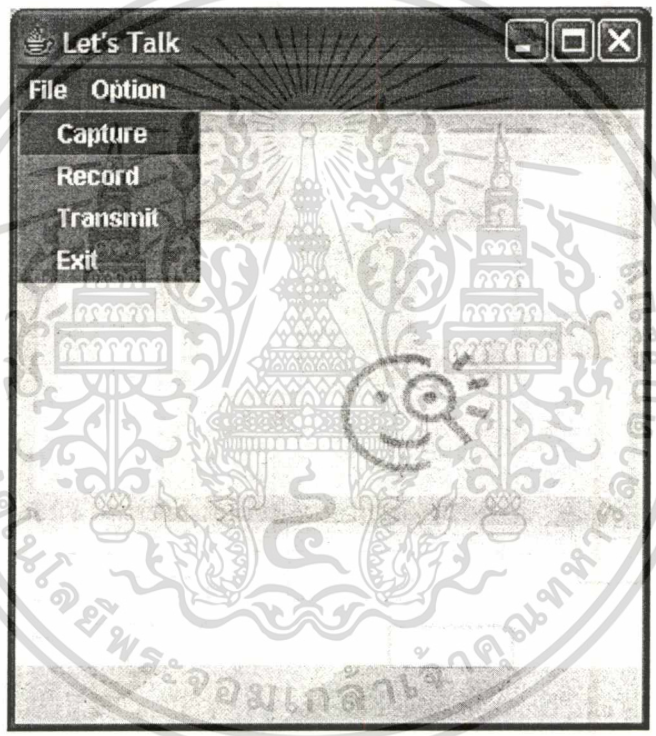
เมื่อเริ่มเข้าสู่การใช้งานโปรแกรมผู้ใช้งานจะเห็นหน้าจอการใช้งานดังรูปที่ 4.1 โดยโปรแกรมนี้ จะเป็นทั้ง server และ client ในตัวเดียวกัน ดังนั้นทั้งต้นทางและปลายทางจึงใช้โปรแกรมนี้ได้เหมือนกัน



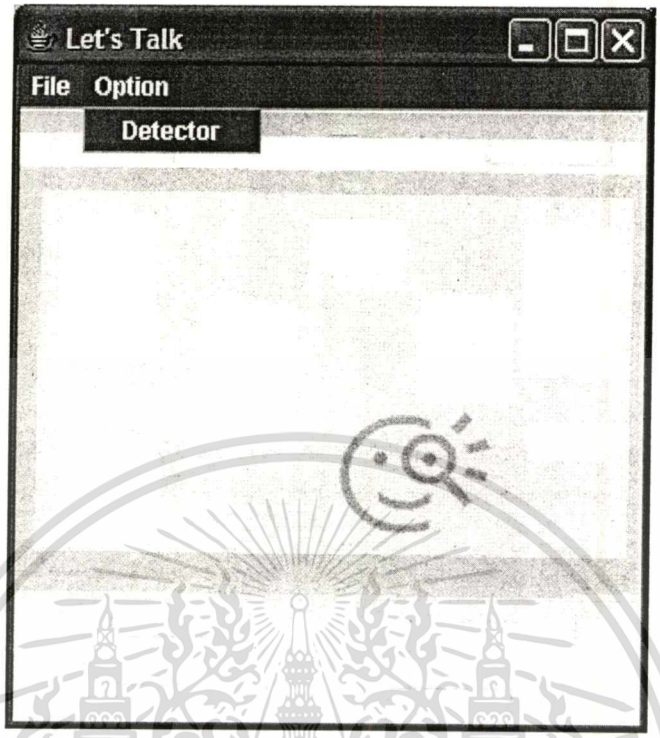
รูปที่ 4.1 แสดงหน้าจอแรกของโปรแกรมประยุกต์

ฟังก์ชั่นหลักของการทำงานของโปรแกรม มี 3 ฟังก์ชั่น ตามที่แสดงในรูปที่ 4.2 และฟังก์ชั่นเสริมของโปรแกรมมี 1 ฟังก์ชั่น คือ ฟังก์ชั่น Detector ตามแสดงในรูปที่ 4.3 ดังนี้

- ฟังก์ชั่น Capture ใช้ในการ capture ข้อมูลเสียง หรือ วิดีโอ
- ฟังก์ชั่น Record ใช้ในการบันทึกข้อมูลเสียง หรือ วิดีโอ
- ฟังก์ชั่น Transmit ใช้ในการส่งข้อมูลเสียง หรือ วิดีโอ ไปยังคู่สนทนาปลายทาง
- ฟังก์ชั่น Detector ใช้ในการตรวจจับสิ่งผิดปกติเพื่อประโยชน์ด้านการรักษาความปลอดภัย



รูปที่ 4.2 แสดงหน้าจอที่แสดงฟังก์ชั่นการทำงานหลักของโปรแกรม



รูปที่ 4.3 แสดงหน้าจอที่แสดงฟังก์ชั่นการทำงานเสริมของโปรแกรม

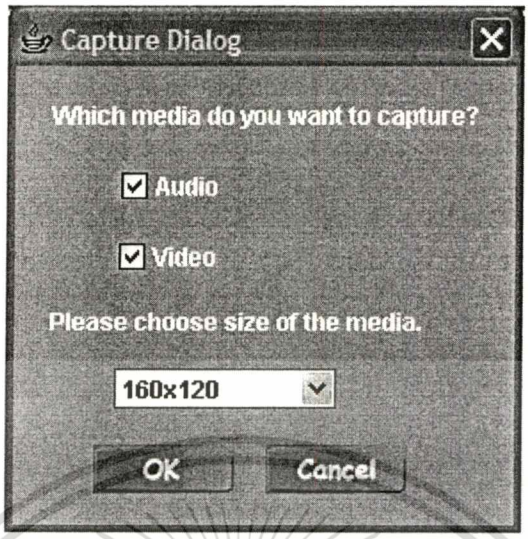
ในแต่ละฟังก์ชั่นของโปรแกรมมีการทำงานดังต่อไปนี้ คือ

4.1 ฟังก์ชั่น Capture

ฟังก์ชั่น Capture นี้จะมีหน้าที่ในการ capture ข้อมูลเสียง หรือ วิดีโอ จากอุปกรณ์รับสัญญาณ เช่น ไมโครโฟน หรือ กล้อง web cam จากนั้นนำผลข้อมูลที่ capture ได้ไปแสดงผลผ่านทางลำโพง หรือ จอภาพ

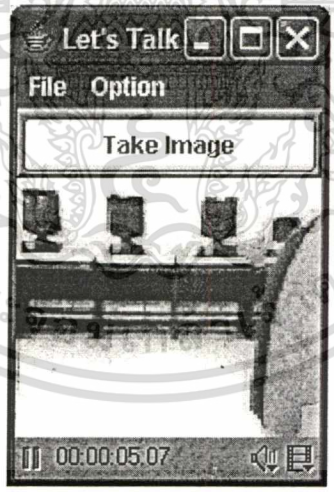
ขั้นตอนการทำงานเมื่อเลือก Menu Capture จากหน้าจอหลัก มีดังต่อไปนี้ คือ

- เมื่อเลือก Menu Capture จากหน้าจอหลัก Capture Dialog ดังรูปที่ 4.4 จะแสดงขึ้นมา เพื่อให้ผู้ใช้งานเลือกว่าผู้ใช้งานต้องการ capture ข้อมูลเสียง หรือ ข้อมูลวิดีโอ หรือ ทั้ง 2 อย่างพร้อมกัน ถ้าต้องการ captureวิดีโอด้วย จะให้เลือกขนาดของวิดีโอที่ต้องการจะ capture ด้วย



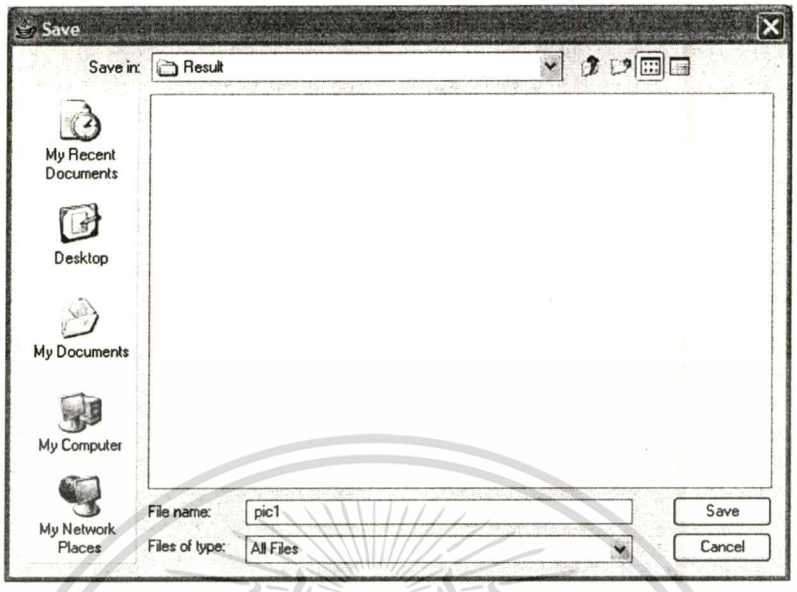
รูปที่ 4.4 แสดงหน้าจอ Capture Dialog

- เมื่อผู้ใช้งานกด OK ในหน้าจอ Capture Dialog หน้าจอแสดงผลจะปรากฏขึ้น ดังรูปที่ 4.5

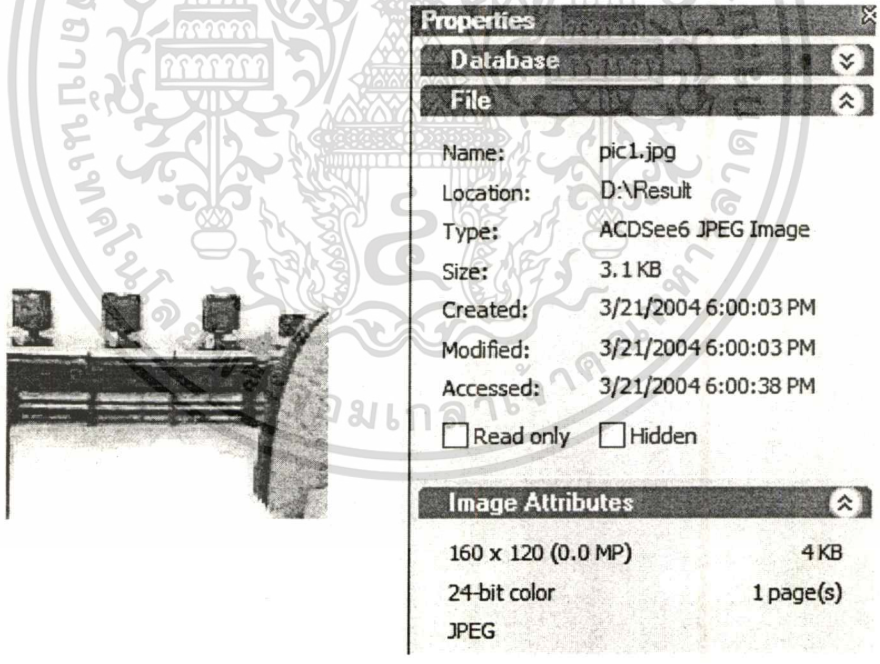


รูปที่ 4.5 หน้าจอแสดงผลการ capture ข้อมูล

- ในหน้าจอแสดงผลนั้น มีปุ่ม Take Image โดยปุ่ม Take Image ใช้เมื่อผู้ใช้งานต้องการที่จะบันทึกรูปภาพจากข้อมูลวิดีโอที่ได้จากการ capture ในขณะนั้น โดยระบบจะทำการบันทึกเฟรมวิดีโอที่แสดงผลในขณะที่ผู้ใช้งานกดปุ่มโดยจะให้ผู้ใช้งานใส่ชื่อไฟล์และเลือกเส้นทางที่ใช้เก็บไฟล์นั้น ดังในรูปที่ 4.6 โดยภาพจะมีนามสกุล .JPG ดังรูปที่ 4.7

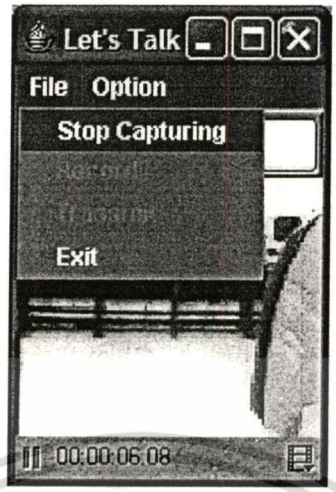


รูปที่ 4.6 หน้าต่างแสดง dialog ที่ให้เลือกเส้นทางที่จะใช้เก็บไฟล์ที่ได้จากการ capture



รูปที่ 4.7 หน้าต่างแสดงไฟล์รูปภาพที่ถูกบันทึก

- เมื่อต้องการที่จะหยุดการ capture ข้อมูลมีเดีย ทำได้โดยการเลือกเมนู Stop Capturing ดังรูปที่ 4.8 ระบบก็จะทำการหยุดการ capture ข้อมูล



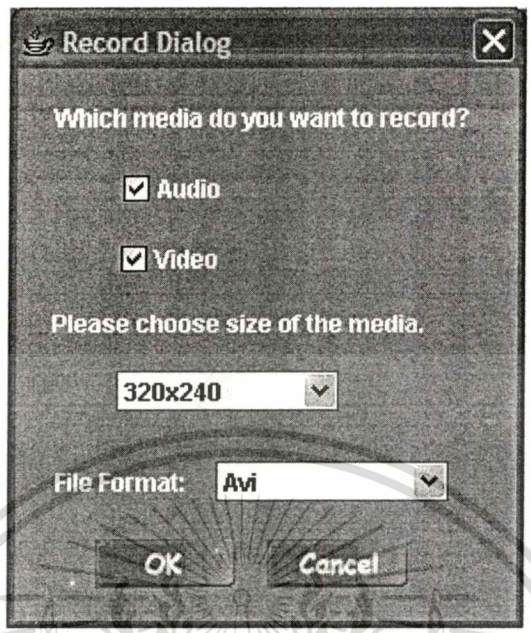
รูปที่ 4.8 หน้าจอแสดงฟังก์ชั่นที่ใช้ในการหยุดการ capture

4.2 ฟังก์ชั่น Record

ฟังก์ชั่น Record นี้จะมีหน้าที่ในการบันทึกข้อมูลเสียง หรือ วิดีโอ จากอุปกรณ์รับสัญญาณ เช่น ไมโครโฟน หรือ กล้อง web cam แปลงเป็นไฟล์ และ เก็บลงในเครื่อง

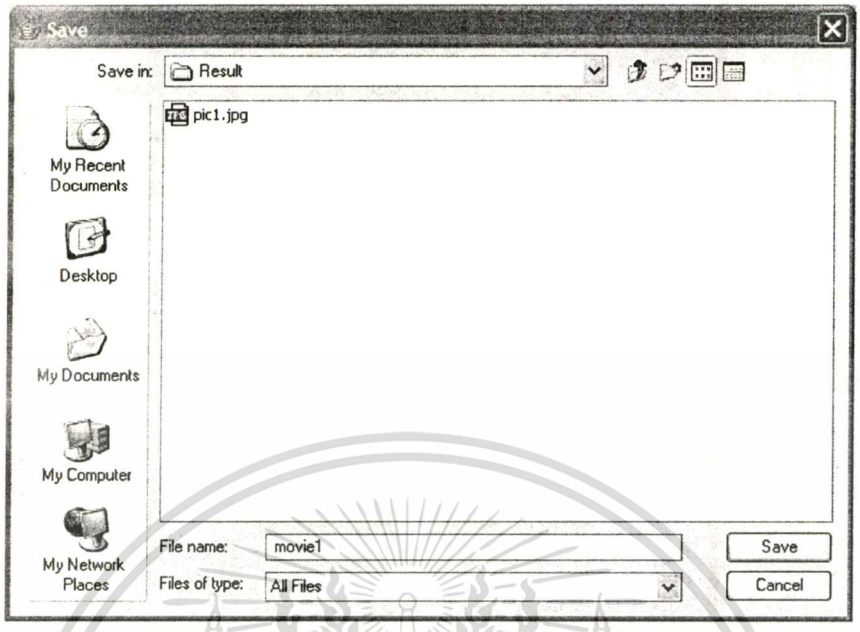
ขั้นตอนการทำงานเมื่อเลือก Menu Record จากหน้าจอหลัก มีดังต่อไปนี้ คือ

- เมื่อเลือก Menu Record จากหน้าจอหลัก Record Dialog ดังรูปที่ 4.9 จะแสดงขึ้นมา เพื่อให้ผู้ใช้งานเลือกว่าผู้ใช้งานต้องการบันทึกข้อมูลเสียง หรือ ข้อมูลวิดีโอ หรือ ทั้ง 2 อย่างพร้อมกัน ให้เลือกขนาดที่ต้องการบันทึก และเลือก file format ที่ต้องการบันทึก

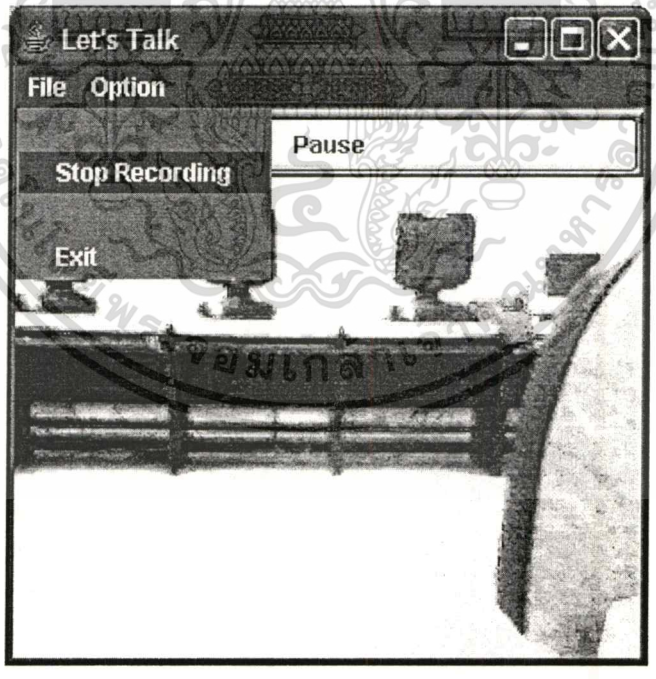


รูปที่ 4.9 แสดงหน้าจอ Record Dialog

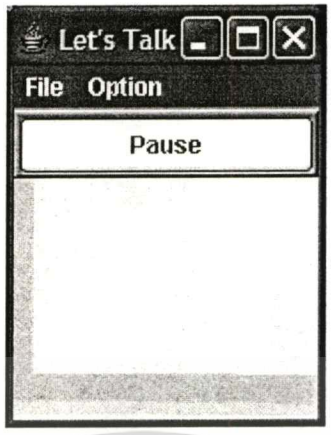
- เมื่อผู้ใช้งานกด OK ในหน้าจอ Record Dialog จะแสดง dialog เพื่อให้ผู้ใช้งานใส่ชื่อไฟล์และเส้นทางในการเก็บไฟล์ดังรูปที่ 4.10 จากนั้นระบบจะทำการบันทึกข้อมูลไปเรื่อยๆ จนกว่าผู้ใช้งานจะเลือกเมนู Stop Recording ในรูปที่ 4.11 ระบบจึงหยุดการบันทึกข้อมูล โดยในระหว่างการบันทึกอยู่นั้นผู้ใช้งานสามารถที่จะทำการหยุดบันทึกข้อมูลชั่วคราว ดังรูปที่ 4.12 และทำการ resume เพื่อทำการบันทึกต่อ ดังรูปที่ 4.13 และผลที่ได้จากการบันทึกแสดงในรูปที่ 4.14



รูปที่ 4.10 แสดงหน้าจอที่ทำการใส่ชื่อไฟล์และเส้นทางในการเก็บไฟล์

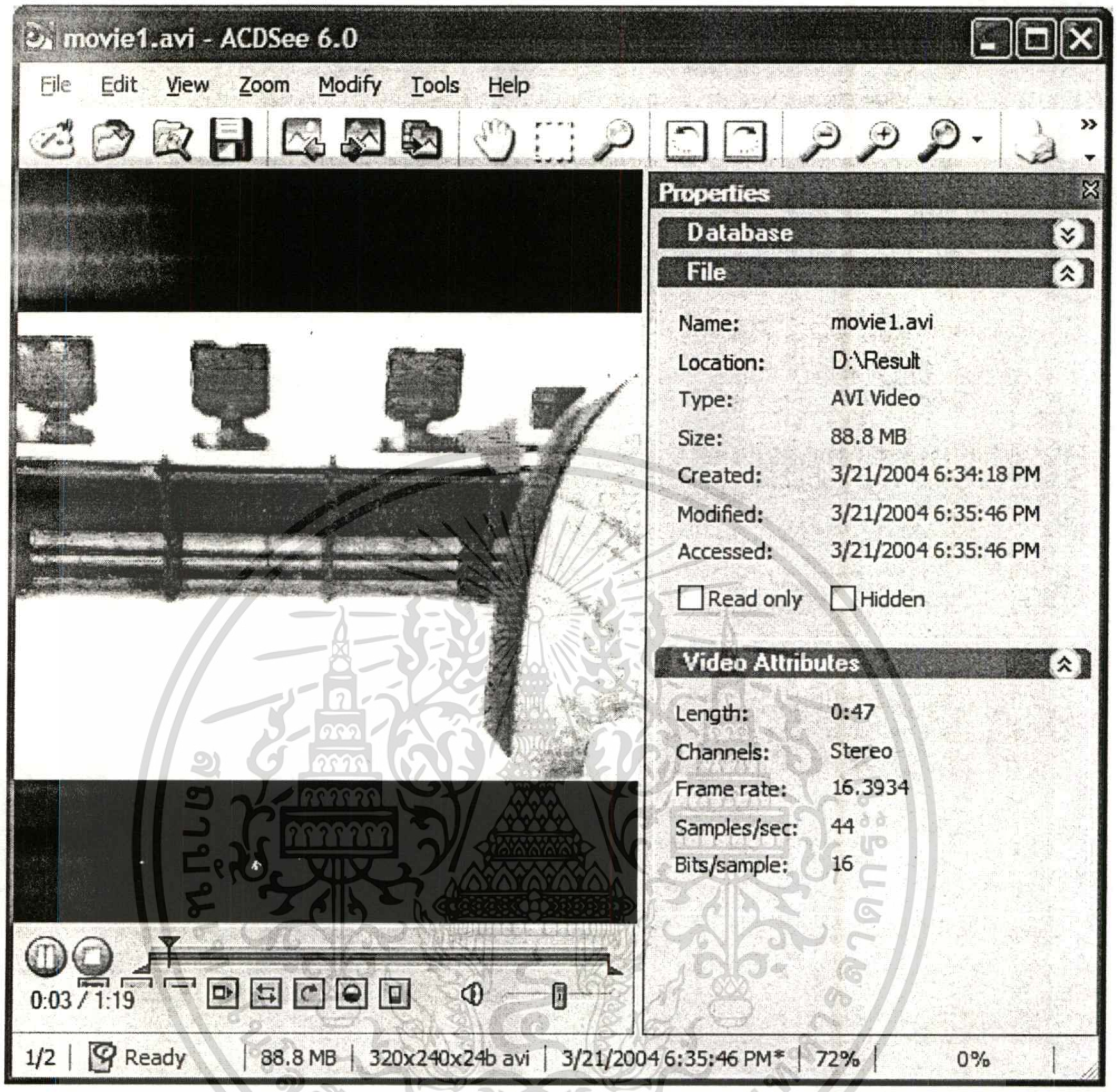


รูปที่ 4.11 แสดงหน้าจอที่มีเมนู Stop Recording



รูปที่ 4.12 แสดงหน้าจอที่มีปุ่ม Pause

รูปที่ 4.13 แสดงหน้าจอที่มีปุ่ม Resume



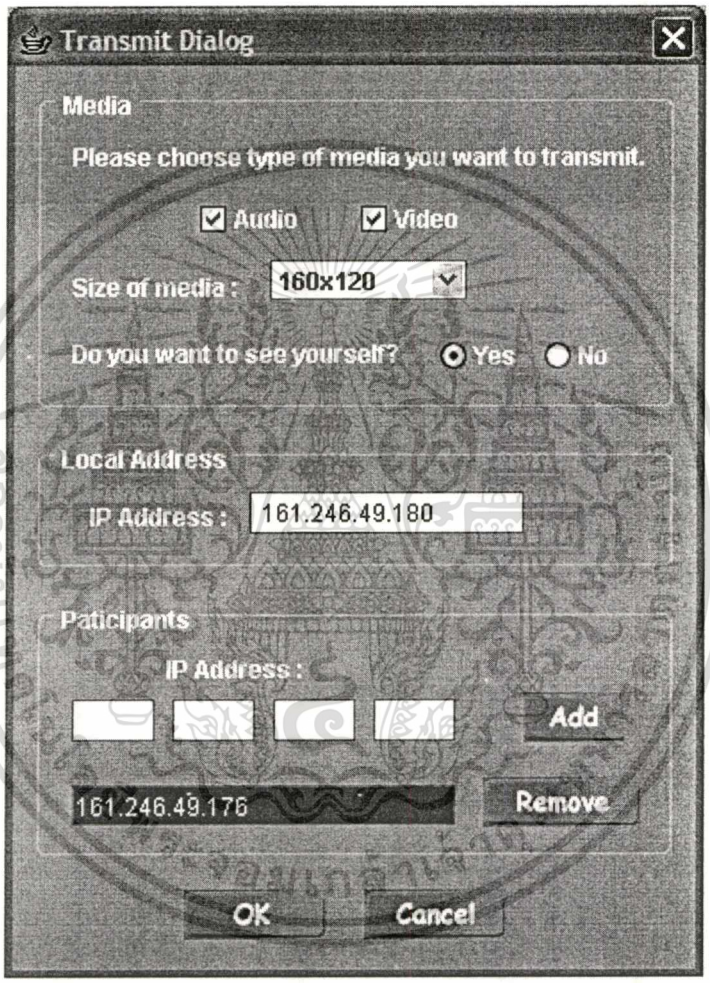
รูปที่ 4.14 หน้าต่างแสดงไฟล์เสียง หรือ วิดีโอที่ถูกบันทึก

4.3 ฟังก์ชั่น Transmit

ฟังก์ชั่น Transmit มีหน้าที่ในการส่งข้อมูลเสียง หรือ วิดีโอ จากอุปกรณ์รับสัญญาณ เช่น ไมโครโฟน หรือ กล้อง web cam ไปยังคู่สนทนาปลายทาง

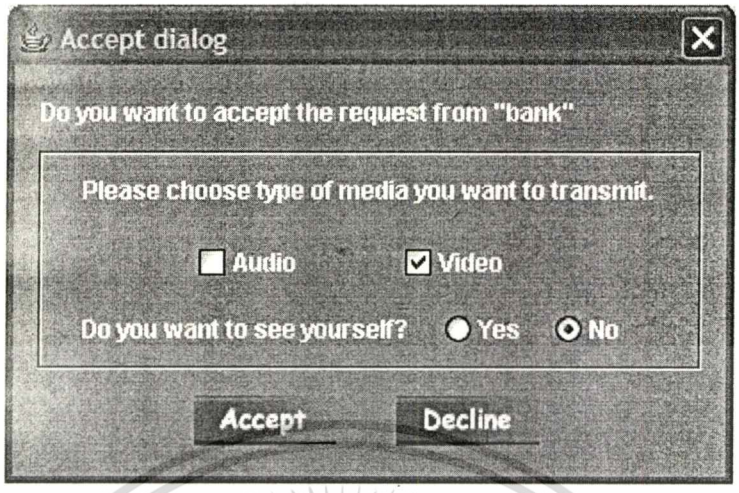
ขั้นตอนการทำงานเมื่อเลือก Menu Transmit จากหน้าจอหลัก มีดังต่อไปนี้ คือ

- เมื่อเลือก Menu Transmit จากหน้าจอหลัก Transmit Dialog ดังรูปที่ 4.15 จะแสดงขึ้นมา เพื่อให้ผู้ใช้งานกรอกข้อมูลต่างๆที่จำเป็นในการส่งข้อมูลไปยังปลายทาง เช่น ให้ผู้ใช้งานเลือกว่าต้องการส่งข้อมูลเสียง หรือ วิดีโอ หรือ ทั้งสองอย่างไปให้คู่สนทนาปลายทาง และให้ใส่หมายเลข IP Addressของคู่สนทนาปลายทาง



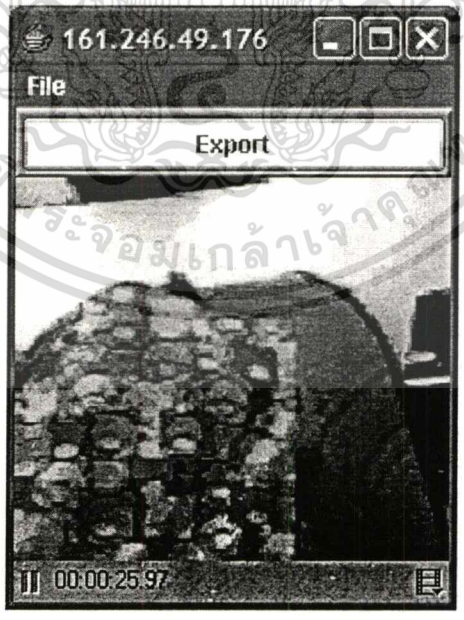
รูปที่ 4.15 แสดงหน้าจอ Transmit Dialog

- เมื่อกดปุ่ม OK เพื่อตกลงที่จะส่งข้อมูลที่ต้องการไปยังปลายทาง ทางปลายทางจะมี dialog เตือนขึ้นมาว่ามีผู้ที่ต้องการจะติดต่อด้วย และสอบถามว่าปลายทางต้องการที่จะรับ หรือ ส่งข้อมูลกลับหรือไม่ ดังรูปที่ 4.16



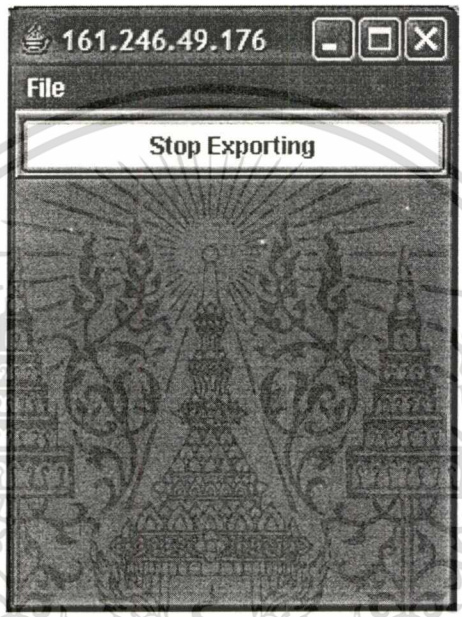
รูปที่ 4.16 แสดงหน้าจอที่เตือนทางด้านผู้รับว่าต้องการติดต่อกับทางฝั่งส่งหรือไม่

- ถ้าปลายทางต้องการที่จะสื่อสารด้วยก็ทำการใส่รายละเอียดของข้อมูลมีเดียที่ต้องการส่งไปยังฝั่งต้นทางที่เป็นผู้เริ่มต้นการสื่อสาร และตอบตกลง เมื่อตกลงแล้วทั้ง 2 ด้านจะได้รับข้อมูลมีเดียของอีกฝั่งที่ส่งมา โดยจะปรากฏในหน้าจอใหม่ ดังรูปที่ 4.17

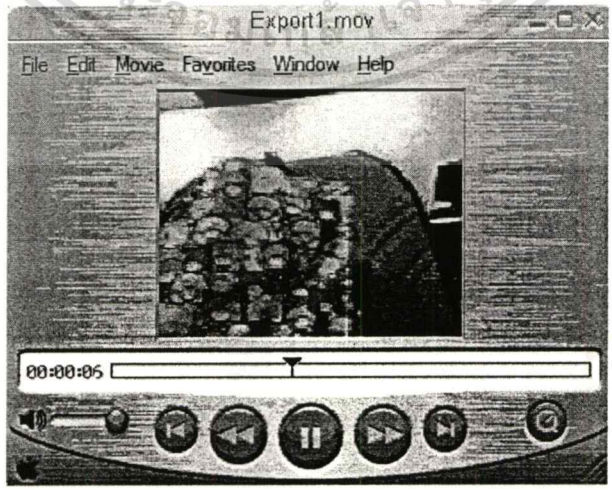


รูปที่ 4.17 แสดงหน้าจอแสดงข้อมูลมีเดียที่ได้รับมาจากคู่สนทนา

- ผู้ใช้งานสามารถที่จะ export ข้อมูลมีเดียที่ได้รับจากคู่สนทนาโดยทำการเก็บไว้ในรูปไฟล์แทนได้ โดยกดปุ่ม export เมื่อกดปุ่ม export จะมี dialog ให้ใส่ชื่อและเส้นทางที่ต้องการเก็บไฟล์นั้น และหน้าจอแสดงผลจะเปลี่ยนไปแสดงดังรูปที่ 4.18 และเมื่อต้องการหยุดการ export ข้อมูลมีเดียที่ได้รับมาจากคู่สนทนาก็ทำได้โดยกดปุ่ม Stop Exporting และไฟล์ที่บันทึกไว้นั้นสามารถนำมาแสดงได้ดังรูปที่ 4.19

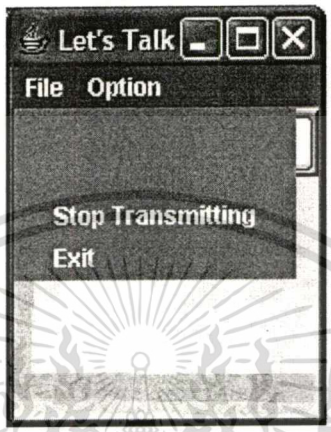


รูปที่ 4.18 แสดงหน้าจอขณะที่ทำการ export ข้อมูลมีเดียที่ได้รับจากคู่สนทนา



รูปที่ 4.19 แสดงไฟล์ที่ได้จากการ export ข้อมูลมีเดียที่ได้รับจากคู่สนทนา

- ขณะที่ทำการส่งข้อมูลไปยังปลายทางนั้น ผู้ใช้งานสามารถทำการ pause และ resume การส่งข้อมูลได้ และเมื่อผู้ใช้งานต้องการหยุดส่งข้อมูลไปยังปลายทางก็สามารถทำโดยเลือกเมนู Stop Transmitting โดยหน้าจอในขณะทำการส่งข้อมูลแสดงได้ดังรูปที่ 4.20



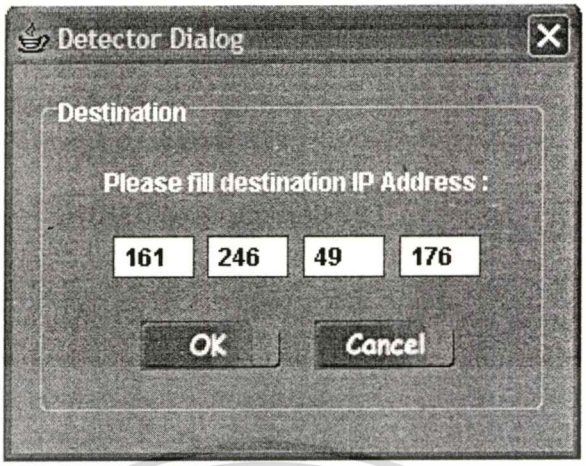
รูปที่ 4.20 แสดงหน้าจอขณะทำการส่งข้อมูล

4.4 ฟังก์ชั่น Detector

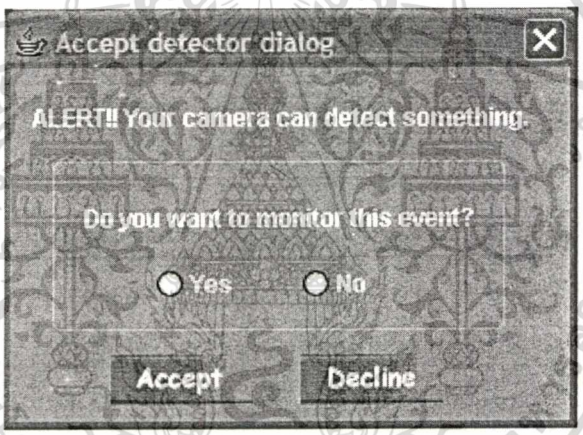
ฟังก์ชั่น Detector มีประโยชน์เกี่ยวกับการรักษาความปลอดภัย โดยจะดักจับความเคลื่อนไหว และถ้าตรวจจับพบความเคลื่อนไหวจะส่งข้อมูลมีเดียในขณะนั้นไปยังปลายทาง

ขั้นตอนการทำงานเมื่อเลือก Menu Detector จากหน้าจอหลัก มีดังต่อไปนี้ คือ

- เมื่อทำการเลือกเมนู Detector จะมีหน้าจอให้ผู้ใช้งานกรอก IP Address ที่ต้องการให้ส่งข้อมูลมีเดียไปในกรณีที่สามารถตรวจจับพบความเคลื่อนไหว แสดงได้ดังรูปที่ 4.21
- เมื่อระบบสามารถตรวจจับพบความเคลื่อนไหว ปลายทางจะได้รับหน้าจอแจ้งว่าตรวจพบความเคลื่อนไหว และถามว่าปลายทางต้องการที่จะรับข้อมูลมีเดียหรือไม่ ดังรูปที่ 4.22
- ถ้าปลายทางตอบตกลงที่จะรับข้อมูลมีเดีย ต้นทางก็จะทำการส่งข้อมูลไปยังปลายทาง และปลายทางก็จะทำการรับข้อมูลนั้น



รูปที่ 4.21 แสดงหน้าจอที่ให้ผู้ใช้งานกรอก IP Address ปลายทาง



รูปที่ 4.22 แสดงหน้าจอที่ปลายทางได้รับแจ้งในกรณีตรวจพบความเคลื่อนไหว

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

โครงงานนี้แบ่งการทำงานออกเป็น 2 ส่วนหลัก คือ ส่วนการศึกษาทฤษฎีที่เกี่ยวข้อง และการพัฒนาโปรแกรมประยุกต์ โดยในส่วนของการศึกษานั้นเน้นไปในการศึกษาการติดต่อสื่อสารด้วยเสียง และ วิดีโอตามมาตรฐาน H.323 และศึกษาการใช้งาน Java Media Framework API

ส่วนการพัฒนาโปรแกรมนั้น ได้พัฒนาตามหลักการของ object-oriented โดยเริ่มจากการการหาความต้องการของระบบที่จะพัฒนา จากนั้นจึงมาออกแบบระบบโดยใช้ UML มาช่วยในการออกแบบ และใช้ไดอะแกรมของ UML แสดงการทำงานของระบบ โดยจะมี Use Case Diagram ที่เป็นแบบจำลองที่มาใช้เพื่ออธิบายกิจกรรมของระบบ, Class Diagram ที่แสดงถึงองค์ประกอบของระบบที่ทำการพัฒนา และ Sequence Diagram ที่ใช้ในการอธิบายความสัมพันธ์ของ object ต่างๆในระบบ และ ในส่วนของการเขียนโปรแกรมนั้นเขียนโดยใช้ภาษา Java โดยใช้ JMF API เข้ามาช่วยจัดการการทำงานที่เกี่ยวกับข้อมูลมีเดีย และ การรับส่งข้อมูลมีเดียแบบเวลาจริง

5.2 ข้อเสนอแนะในการพัฒนาโปรแกรม

- เนื่องจากโปรแกรมที่พัฒนานั้นได้ทำการพัฒนาในส่วนที่เป็นการรับและส่งข้อมูลมีเดียผ่านทางเครือข่าย แต่ตามมาตรฐาน H.323 นั้น นอกจากในส่วนของการรับส่งข้อมูลมีเดียแบบเวลาจริงแล้ว ยังมีในส่วนของการสถาปนาการติดต่อสื่อสารก่อน และ ส่วนของการปลดปล่อยการติดต่อสื่อสารด้วย ซึ่งสามารถนำไปพัฒนาต่อให้ตรงตามมาตรฐาน H.323

- โปรแกรมที่ได้พัฒนานั้นยังเป็นการติดต่อระหว่าง PC-TO-PC ซึ่งสามารถพัฒนาโดยให้ซอฟท์แวร์ทำงานบนระบบฝังตัว (embedded system) โดยทำหน้าที่เป็นตัวที่ติดต่อกับโทรศัพท์ ให้สามารถติดต่อสื่อสารกันระหว่างคอมพิวเตอร์กับโทรศัพท์ได้

- เนื่องจากโปรแกรมที่พัฒนานั้นได้พัฒนาตามหลักของ object-oriented นักพัฒนาที่ต้องการพัฒนาโปรแกรมที่เกี่ยวข้องกับข้อมูลมีเดียสามารถนำเอาคลาสไป reuse ได้ โดยสามารถเรียกใช้เมธอดที่จำเป็นในคลาสนั้นได้เลย เป็นการประหยัดเวลาในการเขียนคลาสใหม่เองทั้งหมด

บรรณานุกรม

นิรมล เริงวิทย์. 2543. **"การส่งสัญญาณเสียงบนโปรโตคอลอินเทอร์เน็ต"** โครงงานวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า วิศวกรรมศาสตร์, มหาวิทยาลัยเกษตรศาสตร์.

Derek *et.al.* 2003. **Voice Communication system over Internet Protocol.** [Online] Available: http://www.ece.ubc.ca/~elec474/reports/summer03/Team%204.pdf.

Ernest Price, David and James Spence, Alexander. 2002. **An Introduction to H.323 Video Conference.** JNT Association.

Sun Microsystem. 1999. **Java Media Framework API Guide.** [Online] Available: https://jsecom16.sun.com/ECom/EComActionServlet;jsessionid=jsecom16.sun.com-185e0%3A3fe7d547%3Adc6682174c995fcc.

The International Engineering Consortium. **H.323.** [Online]. Available: http://www.iec.org/online/tutorials/h323/.

Video Development Initiative. 2002. **Videoconferencing Cookbook.** [Online] Available: http://www.videnet.gatech.edu/cookbook/.

ประวัติผู้เขียน

ชื่อ	นางสาวดลพร หวังเสริมวงศ์
วันเกิด	27 ตุลาคม พ.ศ. 2524
ประวัติการศึกษา	ระดับอุดมศึกษา วิศวกรรมไฟฟ้า มหาวิทยาลัยเกษตรศาสตร์
ที่อยู่ปัจจุบัน	1078 ถ.ประชาชื่น บางซื่อ กทม. 10800
E-mail	kidkute@hotmail.com