การส่งวีดีโอผ่านไอพีโปรโตคอลบนเครือข่ายท้องถิ่น

TRANSMIT VIDEO OVER INTERNET PROTOCOL OVER LAN

โดย

นาย ศราวุธ ไชยชนะ 44010478

นาย ศิระ บริรักษ์วาณิชย์ 44010484

นาย สัมพันธ์ ศรีรัตนประภาส 44010520

อาจารย์ที่ปรึกษา

รศ. ดร. ไกรสิน ส่งวัฒนา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

การส่งวีดีโอผ่านไอพีโปรโตคอลบนเครือข่ายท้องถิ่น

TRANSMIT VIDEO OVER INTERNET PROTOCOL OVER LAN

โดย

นาย ศราวุธ ไชยชนะ

นาย ศิระ บริรักษ์วาณิชย์

นาย สัมพันธ์ ศรีรัตนประภาส

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ภาควิชา
วิศวกรรมโทรคมนาคม

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **การส่งวีดีโอผ่านไอพีโปรโตคอลบนเครือข่ายท้องถิ่น**

Transmit video over internet protocol over LAN

ผู้จัดทำ

1. นาย ศราวุธ ไชยชนะ	**44010478**
2. นาย ศิระ บริรักษ์วาณิชย์	**44010484**
3. นาย สัมพันธ์ ศรีรัตนประภาส	**44010520**

อาจารย์ที่ปรึกษา

(รศ.ดร.ไกรสิน ส่งวัฒนา)

การส่งวีดีโอผ่านไอพีโปรโตคอลบนเครือข่ายท้องถิ่น

Transmit video over internet protocol over LAN

โดย นาย ศราวุธ ไชยชนะ 44010478

นาย ศิระ บริรักษ์วาณิชย์ 44010484

นาย สัมพันธ์ ศรีรัตนประภาส 44010520

อาจารย์ที่ปรึกษา รศ.ดร.ไกรสิน ส่งวัฒนา

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เป็นการนำเสนอ การส่งวีดีโอผ่านทางไอพีโปรโตคอลบนเครือข่ายท้องถิ่น โดยรับอินพุตจากไฟล์วีดีโอ มาตรฐานที่ใช้ในการส่งเป็นแบบ RTP (Real-time Transport Protocol)

ABSTRACT

This thesis propose about transmitting video information across an IP network over a Local Area Network. The video information come from a file on local disk. We use RTP(Real-time Transport Protocol) for transmission.

สารบัญ

หน้า

สารบัญ(ต่อ)

สารบัญรูป

หน้า

สารบัญรูป(ต่อ)

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

ปัจจุบันเครือข่ายคอมพิวเตอร์ถูกนำมาใช้งานอย่างแพร่หลาย โดยเฉพาะเครือข่ายคอมพิวเตอร์ท้องถิ่น (Local Area network) ในการแลกเปลี่ยนข้อมูลเสียงและวีดีโอผ่านระบบเครือข่ายเป็นเรื่องที่น่าสนใจ เนื่องจากในการส่งข้อมูลที่เป็นเสียงและวีดีโอนี้มีสิ่งที่ต้องคำนึงหลายเรื่องด้วยกัน เช่น ต้องคำนึงถึงเวลาที่ใช้ในการส่งผ่านข้อมูลที่จะต้องเป็นลักษณะของเรียลไทม์(Real Time) ข้อมูลที่ไปยังด้านรับจะต้องมีความต่อเนื่อง ดังนั้นจึงได้มีการศึกษารูปแบบการทำงานของระบบ และพัฒนาโปรแกรมการส่งสัญญาณเสียงและวีดีโอผ่านเครือข่ายขึ้นมา

1.2 วัตถุประสงค์ของโครงงาน

1.ศึกษาการทำงานของโปรโตคอลที่ใช้สำหรับข้อมูลแบบเวลาจริง

2.ศึกษาระบบการส่งสัญญาณเสียงและวีดีโอผ่านเครือข่ายคอมพิวเตอร์

3.ศึกษาการเขียนโปรแกรมภาษาจาวา(Java) และ การใช้งานจาวามีเดียเฟรมเวิร์กแอพพลิเคชั่น (Java media framework application)ในส่วนที่เกี่ยวกับสื่อแบบเวลาจริง

4.พัฒนาโปรแกรมที่ใช้สำหรับการส่งสัญญาณเสียงและวีดีโอบนระบบเครือข่ายท้องถิ่น

1.3 วิธีการดำเนินงาน

ในโครงงานนี้เริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่างๆที่เกี่ยวข้อง ซึ่งมีเรื่องหลักที่ต้องศึกษาคือ โครงสร้างและการทำงานของโปรโตคอลที่ใช้สำหรับข้อมูลเวลาจริง , พื้นฐานของแพ็คเกจเสียง, รูปแบบการทำงานของระบบวีโอไอพี (Voice Over IP), การโปรแกรมภาษาจาวา และการใช้งานแอพพลิเคชั่นของโปรแกรม ที่เกี่ยวกับสื่อเวลาจริง แล้วนำความรู้ที่ได้มาทำการพัฒนาเป็นโปรแกรมที่ใช้สำหรับการส่งสัญญาณเสียงและวีดีโอบนระบบเครือข่ายคอมพิวเตอร์ท้องถิ่น

1.4 ขอบเขตของโครงงาน

โปรแกรมสามารถส่ง-รับไฟล์เสียงหรือวีดีโอจากเครื่องคอมพิวเตอร์แบบเวลาจริงผ่านระบบเครือข่ายคอมพิวเตอร์ท้องถิ่น

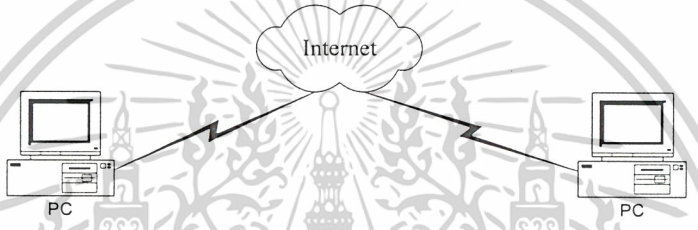
บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 วีโอไอพี (Voice Over IP)

วีโอไอพีเป็นเทคโนโลยีสำหรับการส่งข้อมูลเสียงผ่านอินเตอร์เน็ต ซึ่งจะต่างจากระบบโทรศัพท์เก่าที่ส่งสัญญาณเสียงผ่านชุมสาย วิธีนี้ทำโดยนำข้อมูลเสียงมาทำการแปลงเป็นสัญญาณดิจิตอลแล้วรวมเป็นแพ็กเกจ แล้วจึงส่งแพ็กเกจที่ได้ผ่านอินเตอร์เน็ทไปโดยใช้โปรโตคอลที่กำหนดไว้ สำหรับในโครงงานนี้จะอ้างอิงกับโปรโตคอลอาร์ทีพี ดังจะกล่าวถึงรายละเอียดต่อไป

รูปแบบการใช้ของวีโอไอพีโดยทั่วไปสามารถแบ่งออกเป็น 3 ลักษณะ คือ

1.ระหว่างพีซีกับพีซี

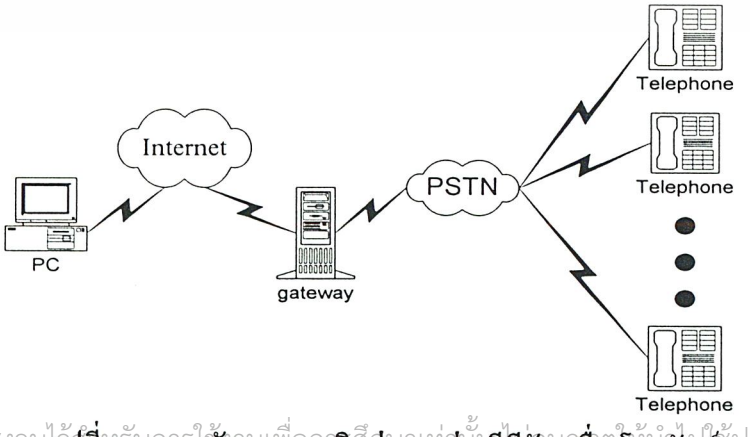


รูปที่ 2-1 แสดงลักษณะการติดต่อระหว่างพีซีกับพีซี

ลักษณะการทำงาน

จากรูปที่ 2-1 จะเป็นการใช้ คอมพิวเตอร์ 2 เครื่องในการ เชื่อมต่อกัน โดยใช้ผ่านทางซอฟต์แวร์หรือใช้งานผ่านทางเว็บในการส่งข้อมูล ระหว่างกัน ข้อมูลที่เป็นเสียงที่เป็นอนาล็อกจะถูกแปลงให้เป็นข้อมูล ดิจิตอล ผ่านทางซาวด์การ์ดแล้วจึงนำข้อมูลที่ได้มาทำการบีบอัด แล้วส่งผ่านโปรโตคอลทีซีที/ไอพี เพื่อส่งผ่านทางอินเตอร์เน็ทไปยังเครื่องคอมพิวเตอร์ปลายทางที่คุยด้วยอยู่ ที่ต้องการจะติดต่อด้วย ดังแสดงดังรูปที่ 1 ตัวอย่างเช่น การใช้ปลั๊กอินในโปรแกรมไอซีซึ่งต้องมีการติดต่อกันโดยทำการล็อกอินเข้าสู่โปรแกรมไอซีคิวทั้ง 2 เครื่องที่จะทำการ ติดต่อกันแล้วจึงจะทำการติดต่อกันได้

2 .ระหว่างพีซีกับเครื่องโทรศัพท์

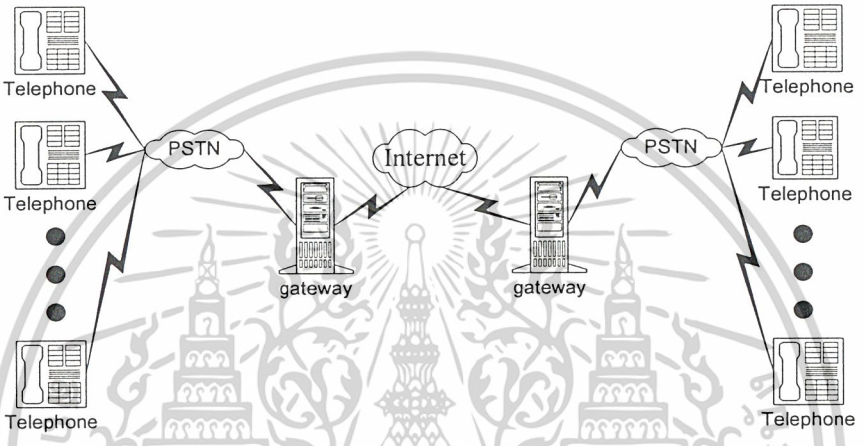


รูปที่ 2-2 แสดงลักษณะการติดต่อระหว่างพีซีกับเครื่องโทรศัพท์

ลักษณะการทำงาน

จากรูปที่ 2-2 เป็นการนำเครื่องคอมพิวเตอร์ทำการติดต่อไปยังระบบโทรศัพท์ที่ใช้อยู่ตามบ้าน (Public Telephone network : PSTN) ในส่วนของคอมพิวเตอร์จะใช้งานผ่านทางซอฟแวร์หรือผ่านทางเว็บที่มีการให้บริการ มีลักษณะการทำงานคล้ายกับแบบพีซีกับพีซีแต่จะมีข้อแตกต่างตรงที่ปลายทางจะเป็นการติดต่อไปยัง ระบบโทรศัพท์ พื้นฐาน ซึ่งต้องมีสิ่งที่เพิ่มขึ้นมาคือเกตเวย์สำหรับการแปลงข้อมูลจากที่ส่งทางอินเตอร์เน็ทไปเป็นสัญญาณในระบบโทรศัพท์ไปยังเครื่องปลายทาง

3.ระหว่างโทรศัพท์กับโทรศัพท์



รูปที่ 2-3 แสดงลักษณะการติดต่อระหว่างเครื่องโทรศัพท์กับเครื่องโทรศัพท์

ลักษณะการทำงาน

จากรูปที่ 2-3 เป็นระบบที่มีการนำเอาระบบโทรศัพท์ระบบเดิมมาใช้ แต่ได้มีการเปลี่ยนแปลงเพิ่มเติมโดยการเพิ่มเกตเวย์ขึ้นที่ชุมสายโทรศัพท์ เพื่อทำการแปลงสัญญาณข้อมูลในระบบโทรศัพท์ไปเป็นข้อมูล เพื่อส่งผ่านทาง อินเตอร์เน็ท และที่ปลายทางก็จะมี เกตเวย์ ทำการแปลงข้อมูลที่ส่งผ่านอินเตอร์เน็ท แปลงกลับไปเป็นสัญญาณโทรศัพท์อีกครั้ง

ประโยชน์ ของวีโอไอพี

1.ประหยัดค่าใช้จ่ายในการติดต่อในระยะทางไกล เช่น การติดต่อระหว่างประเทศ

2 .มีการเรียกใช้แบนด์วิธ น้อยกว่าลดความจำเป็นต้องเช่าคู่สายโทรศัพท์เพิ่มจากเดิมเพื่อใช้ในการโทรศัพท์อีกต่อไป เมื่อนำระบบโทรศัพท์ผ่านอินเตอร์เน็ทมาใช้ ทำให้เราสามารถใช้งานบนระบบเครือข่ายที่มีอยู่แล้วในบริษัทได้

3.มีการให้บริการที่ดีกว่าและมากกว่าการใช้งานแบบเก่าเพราะบริการต่าง ๆ ที่มีอยู่ในตู้สาขาโทรศัพท์(PBX) ที่ใช้ตามบริษัทนั้น ถ้านำระบบโทรศัพท์ผ่านอินเตอร์เน็ทมาใช้ ก็ยังคงให้บริการต่างๆ นั้นได้ปกติอยู่และยังสามารถประยุกต์ใช้งานตามความสามารถของระบบเครือข่ายและระบบอินเตอร์เน็ทที่ต่ออยู่ได้เต็มที่เช่น บริษัทหนึ่งมีเว็บ และแสดงเบอร์โทรศัพท์ติดต่อไปได้ให้ไว้ แต่ว่าขณะนั้นลูกค้าก็ยังไม่อยากจะตัด อินเตอร์เน็ท เออกเพื่อจะโทรติดต่อจึงเป็นการสะดวกแก่ลูกค้าถ้าเราสามารถให้ลูกค้าใช้

โปรแกรมประเภทโทรศัพท์ผ่านอินเตอร์เน็ทเพื่อติดต่อคุยกับพนักงานขายได้โดยตรงผ่านทางอินเตอร์เน็ท เลย ซึ่งมีผลต่อความอยากสั่งซื้อสินค้า

4.เป็นการใช้ประโยชน์จากไอพีได้คุ้มค่าและง่ายเนื่องจากมีการใช้ไอพีเป็นพื้นฐานอยู่ทั่วไปไม่ว่าจะในระบบเน็ตเวิร์กหรือระบบ อินเตอร์เน็ท ในเมื่อมีมันอยู่ทั่วไปแล้ว เราก็ควรจะใช้มันดีกว่าจะต้องแสวงหาทางเลือกอื่นที่อาจจะต้องเสียค่าใช้จ่ายมากกว่าและไม่คุ้มค่า

2.2 พื้นฐานแพ็คเกจเสียง

2.2.1 กระบวนการในการแปลงสัญญาณเสียงเป็นสัญญาณดิจิตอล

เบลได้ทำการค้นพบว่าเสียงที่มนุษย์สามารถได้ยินมีความถี่เสียงอยู่ที่ประมาณ 100 เฮิรตซ์ ถึง 10,000 เฮิรตซ์ โครงข่ายโทรศัพท์แบบอนาล็อก ความถี่เสียงที่สามารถทำการป้อนเข้าสู่ระบบ จะอยู่ในช่วง 300 – 3,300 เฮิรตซ์ แต่ในการทำโครงการนี้เราต้องนำเอาเสียงที่ได้เปลี่ยนมาเป็นข้อมูลดิจิตอลแทน เพื่อที่จะทำการส่งผ่านเครือข่าย การที่จะแปลงเสียงจากอนาล็อกเป็น ดิจิตอลจะมีอยู่ 3 ขั้นตอนซึ่งได้แก่

แซมเปิล (Sample)

ในขั้นตอนนี้จะอาศัยหลักการของคณิตศาสตร์เข้ามาช่วย กระบวนการในการแซมปลิ้ง (Sampling) จะใช้วิธีที่เรียกว่าพัลส์แอมพลิจูดมอดูเลเตอร์ (Pulse Amplitude Modulation, PAM)

พัลส์โค้ดมอดูเลเตอร์นั้นจะเป็นการนำแอมพลิจูดที่มีค่าคงที่ของแซมปลิ้งพัลส์ที่มอดูเลตเป็นโมเดลของอนาล็อกเวฟฟอร์ม ซึ่งในการกำหนดอัตราแซมปลิ้งตามหลักของไนควิส (Nyquist) ได้กล่าวไว้ว่า ต้องกำหนดให้มีอัตราแซมปลิ้งเป็นสองเท่าของความถี่สูงสุดของอนาล็อกอินพุตเวฟฟอร์ม สาเหตุที่เราใช้หลักการของไนควิสในการกำหนดก็เพราะว่าข้อมูลที่ทำการรับนั้นมีจำนวนมากที่ต้องทำการสร้างอินพุต เวฟฟอร์มขึ้นมาใหม่ ถ้ามีการผิดพลาดเกิดขึ้น แซมปลิ้งนั้นก็สามารถที่จะถูกทิ้งได้ สำหรับความถี่เสียงนี้อัตราแซมปลิ้งจะถูกกำหนดไว้ที่ 8,000 เฮิรตซ์ ซึ่งจะมีย่านความถี่เสียงอยู่ที่ 4 กิโลเฮิรตซ์ มีขนาดใหญ่กว่าที่ความต้องการกำหนดไว้ อย่างไรก็ตามความถี่เสียงที่ 4 กิโลเฮิรตซ์ นี้มีประโยชน์หลายอย่างหนึ่งในข้อได้เปรียบนี้คือ ความจริง 4 มาจาก ยกกำลัง 2 ของ 2 และ ตัวประมวลที่เป็นดิจิตอลส่วนมากนั้นจะมีใช้เลขฐาน 2 เป็นพื้นฐานอยู่แล้วดังนั้น 64-Kb/s พีซีเอ็ม เสียงจะสร้าง แซมปลิงทุกๆ 8,000th ของวินาที หรือเท่ากับ 125 μs (ไมโครวินาที)

ควอนไทซ์ (Quantization)

วิธีการทำควอนไทเซชั่นนี้เป็นการนำเอาพัลส์แอมพลิจูดมอดูเลเตอร์มาทำให้เป็นพัลส์โค๊ดมอดูเลเตอร์ในรูปของ 0 หรือ 1 ซึ่งการควอนไทเซชั่นนี้เป็นการนำแซมปลิ้งของ พัลส์แอมพลิจูดมอดูเลเตอร์ที่ผ่านตัวแปลงสัญญาณ อนาล็อกไปเป็นดิจิตอล (A/D) มาทำการเข้ารหัสเพื่อให้ได้รูปแบบที่ต้องการแต่ละแซมเปิลเราจะทำการกำหนด ด้วย 0s หรือ 1s ถ้าพัลส์แอมพลิจูดมอดูเลเตอร์นั้นทำการผลิต 8,000th ของวินาที ดังนั้นบิตที่ใช้แทนพีเอเอ็มแซมเปิลต้องทำการส่งทุก 125 μs หรือ 1/8,000 ของวินาที นี่จะหมายความว่าแต่ละพีเอเอ็มแซมเปิลจะถูกแทนด้วย 16 บิต ดังนั้นการเชื่อมต่อทางดิจิตอลอาจจะมีการรันที่ 125 kb/s (8000x16) จึงต้องมีการจัดการกำจัดบางส่วนออกไป

จะเห็นได้ชัดว่าตั้งแต่ 64 kb/s พีซีเอมทำการรันที่ 64 kb/s และแต่ละพีเอเอ็มแซมเปิลนั้นแทนด้วยจำนวนที่กำลังดีเพราะว่า 8 นั้นมาจาก เพาเวอร์ของ 2 ด้วย ซึ่งจาก 1 ไบต์เท่ากับ 8 บิตดังนั้นเราจึงใช้ดิจิตอลซิกแนลโปรเซสซิ่ง (DSP) ในการดิจิไทซ์เสียงที่มีขนาดโพรเซสเซอร์ 8 บิตซึ่งในการทำงานส่วนต่างๆ จะอยู่ในรูปของ 8 บิต ด้วย

เข้ารหัส (Coding)

การเข้ารหัสนี้เป็นการไลน์โค๊ดหรือว่า 0s หรือ 1s ที่เหมาะสมสำหรับการส่งไปในระยะทางไกลๆ ในการทำการส่งสัญญาณพีเอเอมแซมเปิลในรูปของพีซีเอ็มนั้นจะสามารถทำการส่งได้ในระยะใกล้ๆ เท่านั้นเพราะว่าในการส่งระยะทางไกลๆ จะทำให้ความเสี่ยงในความผิดพลาดมีมากขึ้นจึงต้องมีการเข้ารหัสเพื่อทำการลดข้อผิดพลาดในส่วนนี้

ไลน์โค้ดที่ใช้ในการส่งพีซีเอ็มได้แก่ bipolar alternate mark inversion AMI (bipolar AMI) ได้ถูกนำมาประยุกต์ใช้งานโทรศัพท์ผ่านอินเตอร์เน็ทในระบบ ที-แคเรียร์ ซึ่งเป็นระดับเริ่มแรกของ ที-แคเรียร์มัลติเพล็กไฮลาชี่หรือ ที-วัน ปัจจุบันยังมีการใช้ระบบนี้อยู่ ที-วัน นี้จะไม่ใช้เอเอมไอเรียกว่า binary 8 zero substitution (B8ZS) ซึ่งวิธีนี้เป็นการแก้ปัญหาของวิธีการ เอเอมไอไลน์โค๊ดแบบดั้งเดิม

2.2.2 เซอร์กิตสวิตซ์และแพ็กเกจสวิตซ์

พีเอสทีเอ็นได้พัฒนาเซอร์กิจสวิตซ์เพื่อรองรับความต้องการในการสื่อสารทางเสียงที่ดี แต่การเชื่อมต่อข้อมูลเช่นอินเตอร์เน็ท ได้พัฒนาระบบแพ็กเกจสวิตซ์เพื่อรองรับความต้องการในการสื่อสารข้อมูลที่ดี ทั้งเซอร์กิจสวิตซ์และแพ็กเกจสวิตซ์นั้นมีความแตกต่างในหลายด้านซึ่งสามารถดูได้ดังตารางที่ 2-1

ลักษณะเด่นทางด้าน เน็ตเวิร์ก	Circuit-Switching	Packet-Switching
อัตราการส่งข้อมูล	คงที่และช้า(น้อยกว่า 64 kb/s)	มีความหลากหลาย(ถึง Gb/s)
อัตราการเปลี่ยนแปลงของบิตเรท	ไม่มี	สูง (100/100:1)
ความคงทนต่อการผิดพลาด	สามารถควบคุมได้ด้วยตนเอง	มักจะต้องปลาศจากความผิดพลาด
การส่งข้อมูลซ้ำ	ไม่สามารถทำได้	สามารถทำได้เร็วมากเพียงพอ
การหน่วงของเวลา	มักจะต้องช้าและเสถียร	จะมีความเป็นไปได้สูง
การเชื่อมต่อ	จำเป็นต้องมีการเชื่อมต่อโดยเส้นทางเดินของเวลา	สามารถปราศจากการเชื่อมต่อผ่านเส้นทางเดิมได้

ตารางที่ 2-1 แสดงการเปรียบเทียบการใช้เน็ตเวิร์กของเซอร์กิจสวิตส์และแพ็กเกจสวิตซ์

การสวิตซ์ด้วยเซอร์กิจ

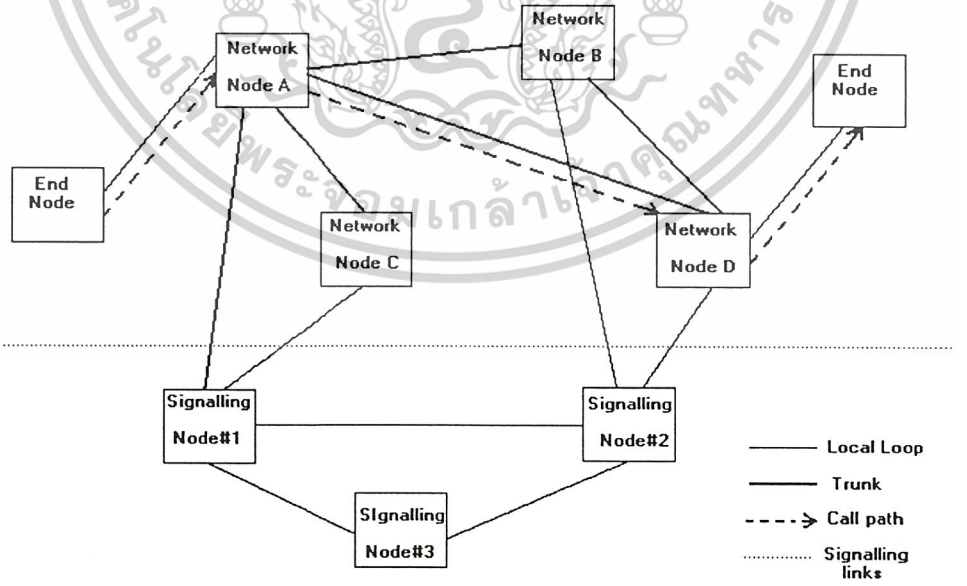
การเชื่อมต่อทางเสียงของพีเอสทีเอ็นที่รู้จักว่าการคอล (call) เมื่อการเชื่อมต่อถูกเซ็ตอัพสำหรับการใช้โมเด็มเพื่อทำการเข้าสู่อินเตอร์เน็ท นี่ก็เป็นวิธีการหนึ่งที่ใช้การคอลของเสียงสู่พีเอสทีเอ็นเสียงของโมเด็มมีความหลากหลายกว่าเสียงของมนุษย์แต่สัญญาณอนาล็อกจะถูกส่งผ่านโครงข่ายกลาง ผ่านสายโทรศัพท์ใหญ่ที่ติดต่อระหว่างเมืองและวนไปเรื่อยเหมือนกับ การคอลของเสียงไปยังเพื่อน ดังนั้นที่ทำการคอลก็เพื่อเป็นการสร้างการเชื่อมต่อและเสียงที่ระดับต่างๆ นั้นจะทำขึ้นเพื่อการคอล เสียงส่วนหนึ่งจะวนอยู่ในที่ตั้งของตัวเอง ส่วนหนึ่งจะถูกส่งผ่านสวิตซ์ของเสียงและส่วนหนึ่งจะผ่านทางสายโทรศัพท์ใหญ่ที่ทำการเชื่อมโยงระหว่างเมือง และอยากถูกส่งไปที่อื่นๆ อีกการเชื่อมต่อนี้จะเป็นชนิดของลาเบล (label) สำหรับชิ้นของเสียงที่ถูกส่งไปตามเส้นทาง ต่างๆ เพื่อเป็นการส่งเสียงไปสู่พีเอสทีเอ็นโดยที่ สวิตซ์จะเป็นตัวที่ทำการสร้างเส้นทางนั้นๆ เส้นทางเป็นแผนที่ของการเชื่อมต่อสู่เน็ตเวิร์กตามความแตกต่างของคุณลักษณะของเซอร์กิจสวิตซ์ เน็ตเวิร์กนั้น จะเป็นลักษณะของคอนเน็คชั่นโอเรียนต์คือเส้นทางจะต้องคงอยู่ตลอดเวลาที่มีการเชื่อมต่อกัน และการเชื่อมต่อนั้นจะเซ็ตอัพโดยการใช้ซิกแนลโปรโตคอล (signaling Protocal) ซึ่งในหลายกรณีบนพีเอสทีเอ็นจะไหลไปบนเน็ตเวิร์กต่างๆ เพื่อใช้สำหรับจุดประสงค์นั้นๆ เซอร์กิจสวิตซ์เน็ตเวิร์กได้แก่พีเอสทีเอ็นจะมีคุณลักษณะร่วม 3 ประการคือ

1 ซิกแนลโปรโตคอลจะใช้ในการเซ็ตอัพการเชื่อมต่อ

2 ทุกบิตจะถูกส่งผ่านเส้นทางเดียวกันทั้งหมด

3 พาธหรือเส้นทางนั้นจะสามารถรองรับทุกย่านความถี่และทุกเวลา

ความสัมพันธ์กันระหว่าง 3 คุณลักษณะนี้ได้แสดงให้ดูได้ดังตัวอย่างดังรูปที่ 2-4



รูปที่ 2-4 แสดงคุณลักษณะของเซอร์กิตสวิตซ์

จากรูปภาพโนดปลาย (End node) ของเน็ตเวิร์กเช่นโทรศัพท์ และ เส้นที่เป็นแสงที่แสดงการทำงานจะแทน โลคอลลูปเน็ตเวิร์กโนดนี้จะเป็นตัวเลือกของโทรศัพท์ และ เส้นที่หนาคือทรั้งค์ (trunk) ที่มีความหนาแน่นของช่องสัญญาณของเสียง เส้นที่มีลูกศรและเป็นเส้นประ จะแทนเส้นทางที่เลือกเมื่อมีการคอลการเชื่อมต่อระหว่างสวิตช์ของเสียงและซิกแนลลิ่งโนด จะกระทำกันเองโดยที่ไม่ต้องมีอะไรมาช่วย ซึ่งแสดงได้จากเส้นประ เมื่อมีการคอลจากซิกแนลลิ่งเน็ทเวิร์กซึ่งจะมีการแยกทางกายภาพจาก เน็ตเวิร์กของเสียง ซึ่งโนดปลายจะไม่มีการต่อตรงกับซิกแนลลิ่งเน็ทเวิร์ก มีเพียงสวิตช์เท่านั้น

ดังนั้น เสียงที่เรียกจะเซ็ตอัพโดยซิกแนลลิ่งโปรโตคอลมันจะทำการเชื่อมต่อเส้นทางสำหรับการเรียกเพื่อเข้าสู่เน็ทเวิร์ก ในขั้นตอนนี้จะมีการเลือกเส้นทางสำหรับการคอลจะไม่สามารถไปสู่ เน็ตเวิร์กโนด C สำหรับเหตุผลตัวอย่างเช่น เป้าหมายไม่สามารถ ถึง โนด C ได้ซิกแนลลิ่งเน็ตเวิร์กนี้สามารถที่จะหาเส้นทางที่คอลผ่าน เน็ตเวิร์กโนด B สู่เน็ตเวิร์ก โนด D แต่ทำไมถึงใช้จาก โนด A สู่โนด B เพราะว่าในการเลือกเส้นทางนั้นตามธรรมชาติจะเลือกเส้นทางที่ใกล้ที่สุดแต่ก็มีกรณีที่ต้องใช้การส่งผ่านเส้นทางโนด B เช่นเส้นทางจากโนด A สู่โนด B ได้ถูกใช้อยู่ มันจะถูกทำการจองเส้นทางไว้ตลอดตามคุณสมบัติของเซอร์กิจสวิตช์ดังนั้นจึงทำการเลือกเส้นทางที่ผ่าน โนด B เป็นทางเลือกที่สอง

ซิกแนลลิ่งโนดจะเป็นส่วนสำคัญของการทำการคอล เช่น จากรูปซิกแนลลิ่งโนดที่ 1 จะต่ออยู่กับโนด A และ C ส่วนซิกแนลลิ่งโนดที่ 2 จะต่ออยู่กับโนด B กับ D ดังนั้นการควบคุมโนดต่างๆจะสามารถทำได้โดยการใช้ซิกแนลลิ่งโนดทั้งสองบนอินเตอร์เน็ทเราท์เตอร์ จะมีหน้าที่ในการแลกเปลี่ยน ข้อมูลต่างๆโดยการใช้เราเตอร์โปรโตคอลซึ่งจะมีการทำงานเหมือนกับซิกแนลลิ่งของวอยซ์ เน็ตเวิร์ก กระนั้นการคอลที่ทำการเซ็ตอัพ หรือ มีการรับส่งจากผู้ที่เราต้องการโทรไปหาจะใช้เส้นทางเดิมในการส่งเสียงทุกบิต ซึ่งในแต่ละทิศทางจะใช้อัตราการส่งข้อมูลนี้จะไม่สามารถเป็นไปได้สำหรับซิกแนลลิ่งเน็ตเวิร์กซึ่งซิกแนลลิ่งแมสเสจไม่สามารถใช้ขนาดอัตราการส่งข้อมูลเดียวกันได้

ดังนั้นคุณลักษณะของซิกแนลลิ่งเน็ตเวิร์กที่ใช้ในการเซ็ตอัพวอยซ์คอลมีดังนี้คือ

1 ซิกแนลลิ่งเน็ตเวิร์กจะเป็นเชื่อมต่อกันแบบคอลเน็คชั่นเลสคือไม่จำเป็นต้องใช้เส้นทางเดิมในการติดต่อเสมอ

2 เมสเสจ ไม่จำเป็นต้องส่งผ่านเส้นทางเดิมเสมอ แต่สามารถที่จะทำการค้นหาเส้นทางใหม่ได้

3 เส้นทางนั้นไม่สามารถทำการกำหนดแบนด์วิธตายตัวได้

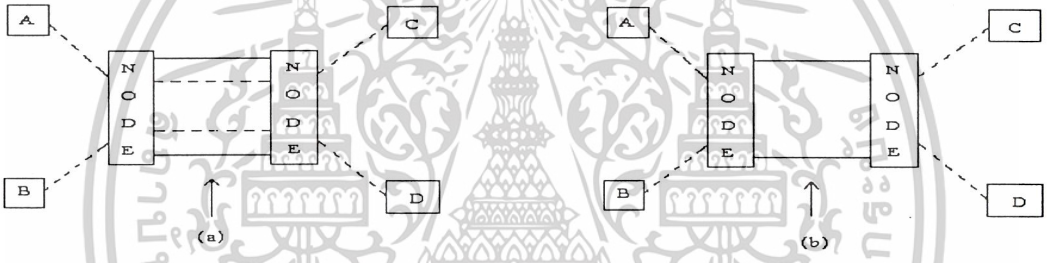
ซิกแนลลิ่งเน็ตเวิร์กที่ควบคุมเสียงในพีเอสทีเอ็น ไม่ใช่ เซอร์กิจสวิตช์แต่เป็นแพ็คเกจสวิตช์

การสวิตช์โดยแพ็คเกจ

เมื่อมีการส่ง ไฟล์ ระหว่างคอมพิวเตอร์สองเครื่องนั้น เส้นทางในการใช้ส่งแบบจุดต่อจุดได้ถูกนำมาใช้ในวิธีนั้นจะทำให้สิ้นเปลืองในการใช้แบนด์วิธมาก แพ็คเกจเน็ตเวิร์กได้ถูกนำมาใช้มันสามารถทำให้ความต้องการของข้อมูลได้เจาะจงมากขึ้นซึ่งเป็นสิ่งที่จำเป็นได้แก่การจัดการอัตราการส่งข้อมูล อัตราการเปลี่ยนแปลงของการส่งข้อมูล ความไม่แตกต่างกันของการหน่วงเวลา และที่สำคัญคือการจัดการกับความผิดพลาดที่เกิดขึ้นได้

แพ็คเกจโดยปกติจะนิยามถึงหน่วยของข้อมูลที่ถูกส่งผ่านจากด้านหนึ่ง แล้วถูกส่งผ่านไปยังส่วนต่างๆ จนกระทั่งถึงจุดหมายปลายทาง ซึ่งนั่นเป็นจุดสำคัญของการใช้ เน็ตเวิร์กในการส่งข้อมูลระหว่างกัน ปกติทั่วไปแพ็คเกจนั้นจะประกอบด้วยเน็ตเวิร์กแอดเดรสซึ่งเป็นส่วนสำคัญที่ใช้ในการค้นหาเส้นทางไปจนกระทั่งถึงปลายทาง

ในการเปลี่ยนแปลงของ แพ็คเกจ ข้อมูลนั้นต้องการวิธีที่ใหม่ในการทำมัลติเพล็กจึงทำให้ไทม์ดิวิชั่นมัลติเพล็ก (time division multiplexing, TDM) นั้นถูกนำมาใช้ในพีเอสทีเอ็นการส่งผ่านข้อมูลไปตามเส้นทาง ของแพ็คเกจ เน็ตเวิร์กนั้นจะใช้วิธีที่เรียกว่า สเตติกทีดีเอ็ม หรือ อะซิงโครนัสทีดีเอ็ม หรือต่อมารู้จักกันในชื่อของ เอทีเอ็ม (ATM) เพราะว่า ข้อมูลนั้นได้ถูกจัดเป็นกลุ่มๆ มันจึงดูไม่เหมาะที่จะนำไปส่งผ่านตามเส้นทางที่ได้กำหนดแบนด์วิธที่ตายตัวแน่นอนอย่างเช่นพีเอสทีเอ็นซึ่งมีการใช้เวลาที่เต็มจำนวนด้วยดังนั้นจึงไม่เหมาะที่จะใช้แบบเซอร์กิจสวิตซ์นี้ แต่เมื่อหันมาใช้แบบแพ็คเกจสวิตซ์แล้วเราสามารถที่จะใช้แบนด์วิธได้เต็มที่ในขณะที่เราสามารถเหลือเวลาในการส่งข้อมูลอื่นได้อีก แนวคิดของแพ็คเกจสวิตซ์นี้จะแสดงได้ดังรูปที่ 2-5



รูปที่ 2-5 (a) ทีดีเอ็ม : 64 kb/s สำหรับ a-c และ b-d

(b) สเตติกทีดีเอ็ม : 128 kb/s สำหรับ a-c และ b-d

สเตติกทีดีเอ็มนั้นได้ถูกสร้างขึ้นมาสำหรับการส่งข้อมูลเป็นกลุ่ม การเชื่อมโยงระหว่างตัวมัลติเพล็กนั้นจะไม่มีช่องสัญญาณที่แน่นอน คอมพิวเตอร์จะไม่ได้กำหนดขนาดของช่องสัญญาณไว้เพียงที่ 64 kb/s ต่อช่องสัญญาณ (นี่เป็นตัวอย่าง) แต่สำหรับทีดีเอ็มสามารถใช้ได้เต็ม 128 kb/s แต่อย่างไรก็ตามทั้ง สเตติกทีดีเอ็มและทีดีเอ็มไม่ได้พยายามที่จะทำงานอย่างแท้จริงที่เวลาเดียวกันเมื่อมีการส่งข้อมูลที่มีขนาดใหญ่มากทั้งสองฝั่งตัวมัลติเพล็กไม่มีทางเลือกจึงจำเป็นต้องนำแพ็กเกจนั้นเก็บลงบัฟเฟอร์ก่อน เมื่อมีการส่งข้อมูลขนาดใหญ่นั้นจบสิ้นลง จึงจะปล่อยแพ็คเกจนั้นออกมา เน็ตเวิร์กซึ่งมีการใช้สเตติกทีดีเอ็มบ่อยๆสามารถจะจัดหาแบนด์วิธได้ตามความต้องการของ โปรแกรมประยุกต์นั้นๆซึ่งเป็นการดีขึ้นเมื่อมีการใช้เทคนิคที่เรียกว่า flexible bandwidth allocation เพราะว่าผู้มีความคิดที่ว่าสามารถใช้แบนด์วิธ ได้มากกว่าที่มีอยู่เช่นเรามีแบนด์วิธใช้งานอยู่ 1.5 Mb/s แต่ถ้าต้อง 2 Mb/s เราสามารถใช้มันได้ ดังนั้นจึงนำบัฟเฟอร์มาช่วยเพื่อให้สามารถทำได้ตามหลักการนี้

บัฟเฟอร์คือพื้นที่ของหน่วยความจำที่สำรองไว้สำหรับ การจัดการเครือข่ายทั่วไป เพราะว่าหน่วยความจำนี้จะจำกัดทรัพยากรในอุปกรณ์เน็ตเวิร์กด้วย คุณภาพของเสียงนั้นจะมีผลมากต่อการเสถียรภาพของการหน่วงเวลา แพ็คเกจเน็ตเวิร์กส่วนมากไม่เหมาะสมสำหรับการส่งข้อมูลเสียงเพราะว่าการส่งข้อมูลเสียงนี้จะไม่สามารถที่จะหน่วงได้เพราะว่าจะทำให้ไม่สามารถสื่อสารได้รู้เรื่อง สาเหตุที่จะทำให้ข้อมูลเสียงสามารถที่จะหน่วงไว้ได้ เช่น เรามีแพ็คเกจอยู่หลายอย่างอยู่ในคิวจนเต็ม ดังนั้นเราจึงไม่สามารถที่จะส่งแพ็คเกจของเสียงออกเลยได้ดังนั้นจึงทำให้เกิดการ หน่วงของข้อมูลขึ้น

ในเซอร์กิจสวิตซ์ไม่มีบัฟเฟอร์จึงทำให้สามารถที่จะส่งข้อมูลได้ตลอดเวลา แต่จะมีความผิดพลาดค่อนข้างสูง ดังนั้นจึงไม่เหมาะที่จะทำการส่งข้อมูลที่ต้องการความผิดพลาดน้อยที่สุด เช่น ข้อมูลทางด้านการเงิน ของ ธนาคารต่างๆในปัจจุบันนี้เราสามารถที่จะนำเอา delay-sensitive packet ที่หัวของ เอาต์พุตคิว ซึ่งแพ็คเกจทั้งหมดจะถูกเซ็ตให้ส่งทันทีที่แพ็คเกจปัจจุบันได้ทำงานเสร็จ นี่เป็นการประกันคุณภาพการบริการของ แพ็คเกจเน็ตเวิร์กอย่างหนึ่ง อย่างไรก็ตามวิธีนี้จะทำให้สามารถลดหน่วยลงได้

2.3 หลักการบีบอัดข้อมูลภาพเคลื่อนไหว

ในงานที่ใช้ทางด้านมัลติมีเดียนั้นส่วนใหญ่แทบจะเป็นไปไม่ได้เลยที่จะทำงานได้โดยปราศจากขบวนการบีบอัดข้อมูลซึ่งปัจจุบันได้มีการพัฒนาด้านสถาปัตยกรรมการประมวลผลภาพนิ่งภาพเคลื่อนไหว โดยการจัดการกับข้อมูลดิจิตอลเพื่อลดขนาดของข้อมูลโดยวิธีการบีบอัดข้อมูลซึ่งแบ่งโดยหลักเป็น 2 วิธี คือ

1.การบีบอัดข้อมูลแบบไม่สูญเสีย (lossless compression) เป็นการบีบอัดข้อมูลที่เมื่อทำการถอดรหัสแล้วข้อมูลที่สร้างคลับขึ้นมาจะไม่มีการสูญเสียตรงส่วนใดเลย

2.การบีบอัดข้อมูลแบบไม่สูญเสีย (lossy compression) โดยปกติทั่วไปแล้วการจัดการกับข้อมูลภาพหรือวีดีโอนั้น ไม่จำเป็นต้องทำการเข้ารหัสและถอดรหัสแล้วให้ได้ข้อมูลเหมือนเดิมทุกอย่าง การสูญเสียข้อมูลบางส่วนไปโดยได้คุณภาพของสัญญาณเอาท์พุทที่ยอมรับได้

2.3.1 พื้นฐานของการเข้ารหัสข้อมูลที่เป็นภาพเคลื่อนไหว (Video Coding Basics)

หากพิจารณากรณีของภาพนิ่งจะสังเกตได้ว่าข้อมูลที่เป็นภาพนิ่งนั้นมีแนวโน้มที่จะมีส่วนของที่ว่างที่ไม่จำเป็น(Spatial redundancy) อยู่เป็นจำนวนมาก ในขณะที่ภาพเคลื่อนไหวเมื่อเราพิจารณาการเคลื่อนที่ของวัตถุ แล้วจะพบว่าการเคลื่อนที่นั้นทำให้มีการซ้ำกันของบางพื้นที่ในแต่ละเฟรมเกิดขึ้น ที่เรียกว่า ที่ว่างไม่จำเป็นที่เกิดขึ้นชั่วคราว(Temporal redundancy) เป้าหมายสำคัญของการบีบอัดข้อมูลภาพเคลื่อนไหวก็คือ การนำทั้งที่ว่างไม่จำเป็นและที่ว่างไม่จำเป็นที่เกิดขึ้นชั่วคราวมาใช้ประโยชน์เพื่อให้ได้การบีบอัดที่ดีที่สุด ส่วนใหญ่แล้วในการเข้ารหัสภาพเคลื่อนไหว จะมีกระบวนการอยู่ 2 กระบวนการที่จะทำให้ได้การบีบอัดที่ดีที่สุด คือ

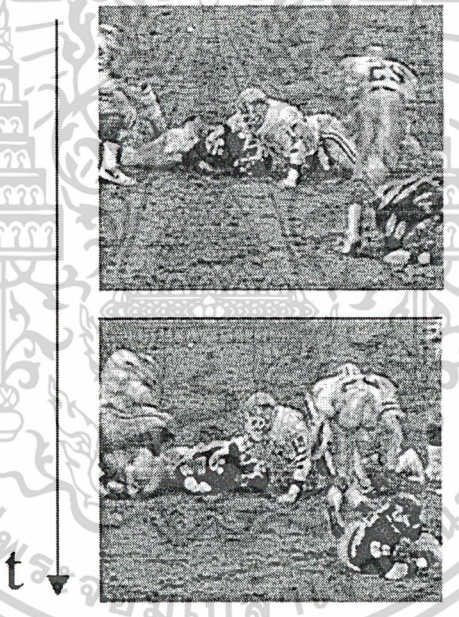
1.กระบวนการสำหรับการลดที่ว่างที่ไม่จำเป็นที่เกิดขึ้นชั่วคราว

2.กระบวนการสำหรับลดที่ว่างที่ไม่จำเป็น

การลดที่ว่างที่ไม่จำเป็นที่เกิดขึ้นชั่วคราว (Reducing Temporal Redundancy)

การประมวลผลภาพในทางอุดมคติที่จะช่วยลดที่ว่างที่ไม่จำเป็นที่เกิดขึ้นชั่วคราวก็คือ การเข้าถึงทุกๆจุดของพิกเซลจากเฟรมหนึ่งไปยังอีกเฟรมหนึ่ง แต่การกระทำเช่นนั้นย่อมทำให้เกิดสัญญาณรบกวน

(noise) เกิดขึ้นด้วยดังนั้นแทนที่เราจะทำทีละพิกเซล เราควรจะแบ่งข้อมูลให้มีขนาดพื้นที่เป็นบล็อกเช่น 16×16ที่เรียกว่ามาโครบล็อก(macroblock)ซึ่งเป็นวิธีการที่เหมาะสมสำหรับการเพิ่มประสิทธิภาพของการลดที่ว่างที่ไม่จำเป็นที่เกิดขึ้นชั่วคราว สมมติว่าเรามีเฟรม2 เฟรมที่ต่อเนื่องกัน โดยให้เป็นเฟรม(t-1) และเฟรม(t) ในขั้นแรกเราจะทำการแยกเฟรม(t) ออกเป็นพื้นที่ขนาด16×16 โดยไม่ซ้อนทับกัน จากนั้นทำการหาว่าในแต่ละบล็อกนั้นสอดคล้องกับพื้นที่จุดสีขนาด 16×16 ของเฟรม(t-1) หรือไม่ ซึ่งผลจากการดำเนินการนี้ กระบวนการลดที่ว่างที่ไม่จำเป็นที่เกิดขึ้นชั่วคราวจะสร้างตัวแทนของเฟรม(t) ซึ่งมีเพียงส่วนที่เป็นความแตกต่างกันของทั้ง2 เฟรมเท่านั้น ถ้าหากเฟรมทั้ง 2 มีความเหมือนกันมาก(High degree of temporal redundancy) เฟรมที่แสดงความแตกต่างก็จะได้ค่าที่ใกล้เคียง 0 เป็นจำนวนมาก แต่ถ้าเฟรมทั้ง 2 ต่างกันอย่างสิ้นเชิงก็จะไม่สามารถหาพื้นที่ที่ซ้ำกันระหว่าง 2 เฟรมนั้นได้ ดังนั้นวิธีการนี้ก็จะไม่ได้ประโยชน์เกิดขึ้น ในระบบการบีบอัดข้อมูลภาพเคลื่อนไหวนั้นวิธีการบีบอัดที่ใช้ในการลดที่ว่างที่ไม่จำเป็นที่เกิดขึ้นชั่วคราวเราเรียกว่า ตัวเข้ารหัสระหว่างเฟรม(Interframe coder)



รูปที่ 2-6 แสดงความสัมพันธ์ของเฟรมภาพที่ต่อเนื่องกัน

การลดที่ว่างที่ไม่จำเป็น (Reducing Spatial redundandancy)

หลังจากขั้นตอนแรกเราจะได้เฟรมที่แสดงถึงความแตกต่างของทั้ง 2 เฟรมที่ติดกันดังนั้นเราจะสามารถหาประโยชน์จากพิกเซลที่เหมือนกันเหล่านั้นได้เพื่อให้ได้ผลของการบีบอัดที่เป็นตัวแทนของเฟรม(t) ซึ่งกระบวนการนี้จะแสดงในขั้นตอนถัดมามาตรฐานการเข้ารหัสจะใช้การเข้ารหัสแบบ DCT เพื่อลดที่ว่างที่ไม่จำเป็น ในระบบการเข้ารหัสภาพเคลื่อนไหวนั้น วิธีการที่ใช้ในการลดที่ว่างที่ไม่จำเป็นเราจะเรียกว่าตัวเข้ารหัสภายในเฟรม(Intraframe coder)การรวมเอาการเข้ารหัสระหว่างเฟรม(Interframe coding) และการเข้ารหัสภายในเฟรม(Intraframe coding)เข้าด้วยกัน เราเรียกว่า วิธีการเข้ารหัสแบบไฮบริด ภายในเฟรมและระหว่างเฟรม (hybrid intraframe/interframe coding method)

กระบวนการชดเชยการเคลื่อนที่ (Motion Compensation)

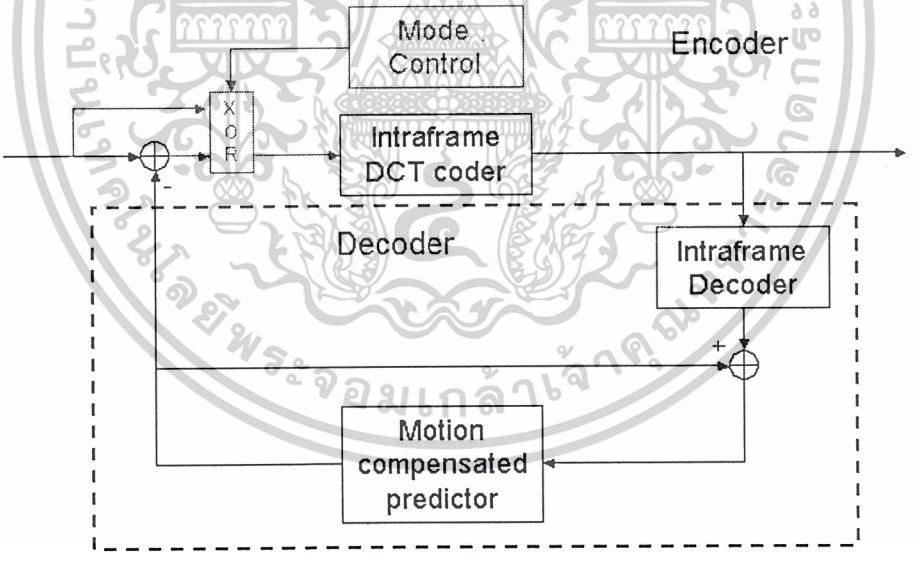
จากกระบวนการที่เปรียบเทียบเฟรม 2 เฟรมที่ติดกันเพื่อหาค่าความแตกต่างของ 2 เฟรมจะอ้างถึงการทำนายที่ต้องใช้การชดเชยการเคลื่อนที่สามารถอธิบายการชดเชยการเคลื่อนที่ได้ว่าเป็นกระบวนการที่ชดเชยการขจัดของวัตถุที่เคลื่อนไหวจากเฟรมหนึ่งไปยังอีกเฟรมหนึ่ง ในทางปฏิบัติการชดเชยการเคลื่อนที่จะถูกดำเนินการด้วยการประมาณค่าการเคลื่อนที่(Motion Estimation) ซึ่งเป็นกระบวนการเพื่อหาพิกเซลที่สอดคล้องกันระหว่างเฟรมที่ต้องการ โดยสามารถเขียนสมการการชดเชยการเคลื่อนที่ได้เป็น

$$e(x,y,t) = I(x,y,t) - I(x-u, y-v, t-1)$$

โดยที่ $I(x,y,t)$ คือ ค่าของพิกเซลที่ตำแหน่ง (x,y) ในเฟรม (t)

$I(x-u, y-v, t-1)$ คือ ค่าของพิกเซลที่สอดคล้องกับ $I(x,y,t)$ ที่ตำแหน่ง $(x-u, y-v)$ ในเฟรม $(t-1)$

จากกระบวนการนี้จะได้เอาท์พุทที่เป็นความสัมพันธ์ของการเคลื่อนที่ของ 1 บล็อคจากเฟรมหนึ่งไปยังอีกเฟรมหนึ่งในรูปเวคเตอร์ (u,v) จากรูปที่ 2-7 แสดงความบล็อกไดอะแกรมของการเข้ารหัสแบบไฮบริด(hybrid coding) สังเกตได้ว่าการชดเชยการเคลื่อนที่จะมีอยู่ทั้งในตัวเข้ารหัสและตัวถอดรหัส แต่การประมาณค่าการเคลื่อนที่นั้นจะมีอยู่ในตัวเข้ารหัสเท่านั้น เมื่อรวมเอาการเข้ารหัสระหว่างเฟรมกับการเข้ารหัสภายในเฟรมเข้าด้วยกันแล้ว จะทำให้อัตราการบีบอัดดีกว่าการทำงานเพียงลำพังกระบวนการเดียว



รูปที่ 2-7 หลักการทำงานของตัวเข้ารหัสและถอดรหัสแบบไฮบริด

2.3.2 การเข้ารหัสโดย DCT (Discreate Cosine Transform)

การเข้ารหัสแบบDCT หรือ Discrete Cosine Tranform เป็นพื้นฐานของมาตรการบีบอัดข้อมูลทั้งภาพ นิ่งและภาพเคลื่อนไหวทั้งหมด พื้นฐานในการแปลงDCT นั้นมีอยู่ด้วยกันหลายประเภทเช่น 1-D DCT, 2-D DCT,3-DDCTซึ่งในแต่ละวิธีนี้ก็สามารถแบ่งออกได้อีกหลายวิธี แต่มาตรฐานของการบีบอัด

ข้อมูลแบบสูญเสีย ของ JPEG และMPEG นั้นจะนิยมใช้แบบ 2-D DCT แต่ในทางทฤษฎีแล้วการแปลง DCT แบบ 3D ประสิทธิ ภาพดีกว่าทั้ง 2 แบบ แต่เนื่องจากว่ามันมีรูปแบบการคำนวณที่สลับซับซ้อนมาก และยุ่งยากต่อการออกแบบทำให้ไม่นิยมใช้กัน ดังนั้นในที่นี้เราจึงขอกล่าวเฉพาะการDCTแบบ 2D เท่านั้น ซึ่งมีรูปแบบคือการแปลงเมตริกซ์ขนาด $N \times N$ ของภาพ และในโดเมนที่ว่าง(Spatial domain) ไปเป็น DCT domain สำหรับมาตรฐานการบีบอัดข้อมูลของภาพนิ่ง จะให้N=8 เหตุผลที่เลือกขนาดของบล็อกภาพเป็น 8×8 พิกเซลก็เพราะว่าไม่ใช้เนื้อที่หน่วย ความจำมากเกินไป และเป็นขนาดที่สายตามนุษย์สามารถสังเกตเห็นความแตกต่างได้

DCT นั้นมีคุณสมบัติเป็น orthogonal transform ดังนั้นเราจึงสามารถเขียนอยู่ในรูปเมตริกซ์ได้ดังนี้

DCT $\quad Y = TXT^{t}$

IDCT $\quad Y = T^{t}XT$

สามารถเขียนสมการแปลงDCT จาก $X^{8\times 8 DCT} \longrightarrow Y$ ได้ดังนี้

$$y_{kl} = \frac{C(k)C(l)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7} x_{ij}\cos\left[\frac{(2i+1)k\pi}{16}\right]\cos\left[\frac{(2i+1)l\pi}{16}\right] \qquad (2.1)$$

โดยที่ $C(k) = 1/2; k = 0$

$= 1; k \neq 0$

หรืออาจเขียนในรูปเมตริกซ์เวคเตอร์ ได้เป็น

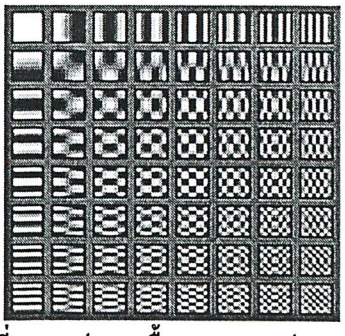
$$Y = TX$$

โดยที่ $X = \{X_{00}, X_{01}, ..., X_{07}, X_{10}, ..., X_{17}, ..., X_{70}, ..., X_{77}\}$

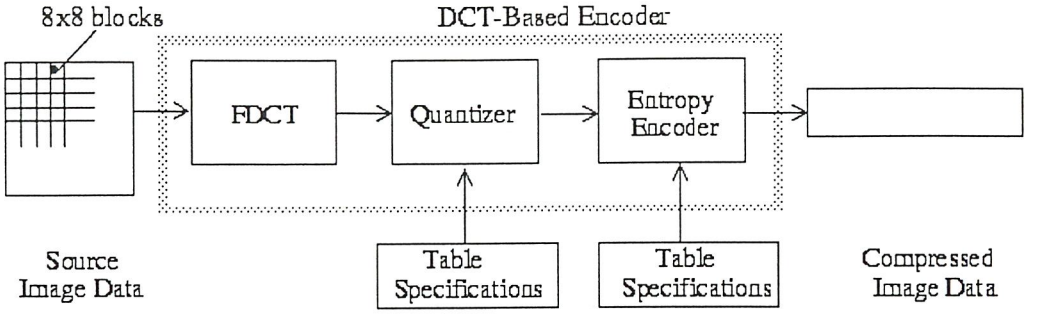
$Y = \{Y_{00}, Y_{01}, ..., Y_{07}, Y_{10}, ..., Y_{17}, ..., Y_{70}, ..., Y_{77}\}$

T เป็นเมตริกซ์ขนาด 64 $\times$ 64 โดยค่าต่างๆ นั้นหาได้จากสมการ DCT ในสมการ (2.1)

การแปลง DCT จะสร้างอินพุทบล็อกใหม่ขึ้นมาเป็นอนุกรมของคลื่น(waveform) ซึ่งแต่ละบล็อกจะมีค่าเฉพาะของ spatial frequency ของตัวมันเอง ซึ่งผลรวมของ 64 คลื่นทั้งหมด นี้จะทำให้ได้ รูปแบบพื้นฐานของ DCT ฟังชั่นดังแสดงในรูปที่ 2-8 และจากรูปแสดงถึงกระบวนการหาสัมประสิทธิ์ของแต่ละคลื่น โดยที่สัม ประสิทธิ์เหล่านี้จะทำให้เราสามารถแปลงข้อมูลกลับมาเป็น เมตริกซ์ X ขนาด 8×8 ได้อีก



รูปที่ 2-8 รูปแบบพื้นฐานการแปลง DCT



รูปที่ 2-9 หลักการทำงานของการแปลงภาพแบบDCT

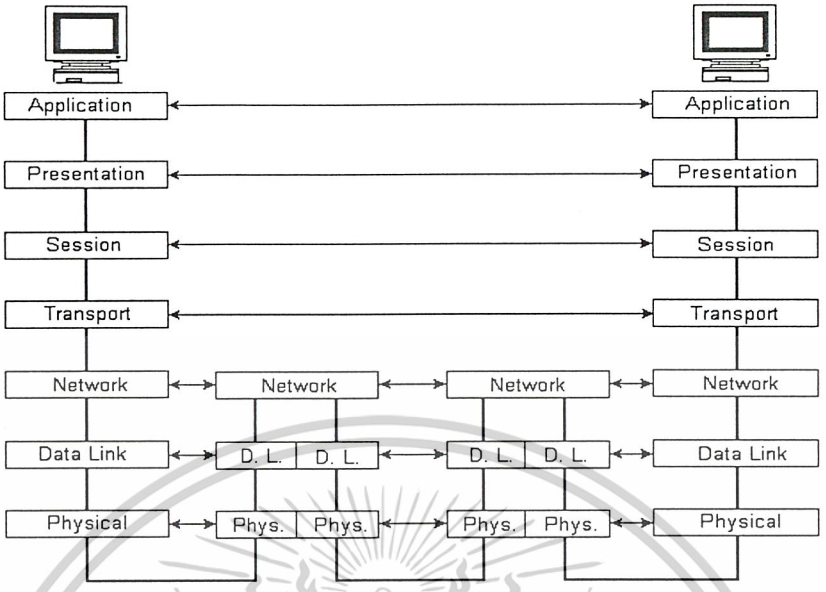
จากรูป 2-9 แสดงให้เห็นถึงขั้นตอนการทำงานโดยทั่วไปของระบบDCTในส่วนของการเข้ารหัส (encoder) DCTจะทำการแปลงบล็อกขนาด 8×8 ในแต่ละบล็อก จากบล็อก X ให้กลายเป็นสัมประสิทธิ์ของบล็อกY จากที่ได้กล่าวมาแล้วว่าค่าสัมประสิทธิ์ในแต่ละอันเหล่านี้จะเป็นตัวถ่วงน้ำหนักที่มีความเกี่ยวข้องกับรูปแบบพื้นฐานของ DCT(DCT basic waveform) ที่มีลักษณะตรงกัน ในระบบการบีบอัดข้อมูลแบบสูญเสีย ตัวถ่วงน้ำหนักบางตัวจะถูกลบออกเพราะฉะนั้นรูปแบบสัญญาณ(waveform) ที่มีค่าตรงกันกับตัวถ่วงน้ำหนักนั้นก็จะไม่ถูกนำมาใช้ในการดีคอมเพรส(Decompress)อีก

จากกระบวนการลบตัวถ่วงน้ำหนักออกจะตรงกันกับขั้นตอนของการควอนไตซ์ ซึ่งเป็นกระบวนการที่ย้อนกลับไม่ได้และเป็นขั้นตอนเดียวที่ทำให้มีการสูญเสียใน DCT coding scheme และหลังจากที่ข้อมูลผ่านการควอนไตซ์แล้วค่า y_{kl} จะถูกบีบอัดข้อมูลแบบไม่สูญเสียโดยใช้การเอนโทรปี (entropy coder)

จากสมการการแปลงDCTในสมการ การคำนวณหาบล็อกขนาด 8×8 นั้นเราต้องใช้ตัวเลขนั้นมาคูณเท่ากับ4096 ครั้ง และในรูปภาพ ภาพหนึ่งนั้นจะประกอบไปด้วยหลายๆบล็อกทำให้ในการคำนวณการแปลงค่าโดย DCTนั้นจะต้องใช้การคำนวณหลายๆครั้งทำให้เสียเวลาในการออกแบบและการคำนวณเป็นอย่างมาก ดังนั้นเราจึงจำเป็นต้องหาวิธีการที่สามารถคำนวณค่าต่างๆโดยใช้การคำนวณให้น้อยที่สุดเพื่อที่จะง่ายต่อการออกแบบและคำนวณเร็วขึ้น วิธีการง่ายๆที่สามารถช่วยได้ก็คือ คำนวณ DCT แบบ 1-D ก่อนแล้วค่อยทำการแปลงจาก 1-D ไปเป็น 2-D อีกครั้งซึ่งช่วยลดการคำนวณจาก 4096 ครั้งไปเป็น 1024 ช่วยลดการคำนวณไปถึง 4 เท่า

2.4 แบบจำลอง OSI (OSI model)

แบบจำลอง OSI (Open System Interconnection model) เป็นรูปแบบของเครือข่ายที่ถูกเสนอขึ้นโดยองค์การ ISO (International Standard Organization) ในปี 1983 ซึ่งมีวัตถุประสงค์ใหญ่เพื่อให้ระบบคอมพิวเตอร์ต่างๆ สามารถเชื่อมต่อเป็นระบบเดียวกันได้ ไม่ว่าเครื่องนี้จะเป็นของผู้ผลิตรายใด แต่ว่ารูปแบบในการเชื่อมต่อกันเป็นเครือข่ายนั้นจะปฏิบัติตามแนวทางเหมือนกันตามแบบจำลอง OSI นอกจากนี้แล้วในแต่ละชั้นของแบบจำลอง OSI ก็ต้องใช้โปรโตคอลแบบเดียวกันด้วย ปัจจุบันจำลอง OSI มีรูปแบบเป็นลำดับขั้นโปรโตคอล (Protocol Hierachies) โดยแบ่งชั้นทั้งหมด 7 ชั้นด้วยกัน



รูปที่ 2-10 แสดงโครงสร้างแบบจำลองโอเอสไอ

ชั้นฟิสิคัล (Physical layer)

เป็นชั้นที่เกี่ยวข้องกับการติดต่อสื่อสารในระดับเบื้องต้น ที่เรียกกันว่าบิทดิบ จุดประสงค์ใหญ่ในการออกแบบหน้าที่ของชั้นคือเมื่อผู้ส่งส่งบิทที่มีค่าเป็น 1 ออกไป ทางด้านผู้รับต้องรับเป็น 1 เช่นเดียวกัน จากความต้องการดังกล่าวทำให้เกิดเป็นข้อกำหนดต่างๆ เช่นการกำหนดแรงดันไฟฟ้าที่มีค่าเป็น 1 และ 0 ช่วงเวลาในการส่งบิท 0 ที่ติดกันนั้นห่างกันได้น้อยแค่ไหน ความเป็นไปได้ในการส่งสัญญาณ 2 ทิศทางโดยมีช่องสัญญาณเดียวกัน การเริ่มต้นการเชื่อมต่อและการยกเลิกการเชื่อมต่อเมื่อการติดต่อสิ้นสุดลงว่าเป็นอย่างไร จะเห็นว่าในชั้นนี้ต้องเกี่ยวกับคุณสมบัติทางกลของไฟฟ้าเป็นส่วนใหญ่

ชั้นดาต้าลิงก์ (Data Link Layer)

หน้าที่หลักของชั้นนี้คือ การจัดส่งข้อมูลผ่านไปยังชั้นฟิสิคคอล รวมทั้งจัดการเกี่ยวกับการตรวจสอบความผิดพลาดและแก้ไขความผิดพลาด ถ้าจะสรุปก็คือเมื่อมองจากชั้นที่อยู่เหนือดาต้าลิงค์ถัดขึ้นไป ซึ่งก็คือชั้นเน็ตเวิร์กนั้น ตัวของชั้นเน็ตเวิร์กจะต้องมองเป็นว่าการส่งผ่านข้อมูลจากตัวเองไปยังชั้นเน็ตเวิร์กของคอมพิวเตอร์ตัวอื่นนั้นเป็นการส่งผ่านข้อมูลที่สมบูรณ์ไม่มีข้อผิดพลาดเลย ซึ่งเป็นหน้าที่ของของชั้นดาต้าลิงค์ที่จะต้องให้บริการแก่ชั้นเน็ตเวิร์ก

เมื่อชั้นดาต้าลิงค์ได้ข้อมูลที่ส่งผ่านจากชั้นเน็ตเวิร์ก ก็จะดำเนินการแบ่งข้อมูล จากนั้นก็จะดำเนินการส่งไปยังชั้นดาต้าลิงค์ของคอมพิวเตอร์เครื่องทางด้านรับ ด้านรับเมื่อได้รับเฟรมข้อมูลแล้วก็จะส่งเฟรมตอบรับ (Acknowledgement Frame) กลับไปยังเครื่องส่ง หากพิจารณาว่าข้อมูลได้ถูกส่งเรียกกันไปโดยไม่สนว่าบิทที่ต่อกันนั้นจะประกอบกันเป็นเฟรมหรือไม่ ดังนั้นจะเป็นหน้าที่ของดาต้าลิงค์ที่จะต้องกำหนดขอบเขตและขนาดของเฟรมข้อมูลเอาเอง ซึ่งชั้นดาต้าลิงค์จะทำโดยการแทรกบิทพิเศษเอาไว้ที่ต้นเฟรมและที่ทายเฟรม

ชั้นเน็ตเวิร์ก (Network Layer)

ชั้นเน็ตเวิร์กมีหน้าที่หลักเกี่ยวกับการควบคุมการทำงานของชุดข้อมูลแพ็ตเกจ ซึ่งต้องกำหนดให้ชุดข้อมูลแพ็กเกจใดจะถูกส่งไปยังเส้นทางใด (Rout) จากด้านส่งไปยังด้านรับ โดยเส้นทางนี้อาจจะถูกกำหนดออกมาแบบตายตัวเลย ระหว่างตัวประมวลผลเชื่อมต่อข่าวสารเรียกว่า IMP (Interface Message Processer) เข้ากับ IMP อีกตัวหนึ่ง หรือการกำหนดแบบไดนามิก(Dynamic) ซึ่งทำให้การติดต่อระหว่าง IMP แต่ละคู่นั้นจะใช้เส้นทางในการติดต่อแตกต่างกัน ทั้งนี้เพื่อลดปัญหาด้านทราฟฟิกของเครือข่าย

ชั้นทรานสปอร์ต (Transport Layer)

หน้าที่หลักของชั้นนี้คือการทำการรับข้อมูลของเซสชั่น จากนั้นทำการแยกข้อมูลออกเป็นหน่วยที่เล็กลง แล้วจึงทำการส่งผ่านหน่วยต่างๆเหล่านี้ไปยังชั้นเน็ตเวิร์กและต้องกระทำการอันเป็นที่แน่ใจว่าหน่วยต่างๆที่จะต้องไปถึงจุดหมายอย่างถูกต้องตามลำดับ ในสภาวะปกตินั้นชั้นทรานสปอร์ตจะทำหน้าที่คอยการสร้างเส้นทางการติดต่อสื่อสารตามที่ชั้นเซสชั่นร้องขอมาเมื่อชั้นเซสชั้นร้องขอมาเมื่อชั้นเซสชั่นต้องการส่งผ่านข้อมูล และทางชั้นทรานสปอร์ตต้องการให้การไหลเวียนของข้อมูลเป็นไปอย่างรวดเร็วก็อาจจะสร้างเส้นทางขึ้นมาหลายเส้นทางแล้วแยกข้อมูลออกเป็นหน่วยย่อย เพื่อที่จะส่งข้อมูลแต่ละหน่วยแยกทางกันออกไป

ชั้นทรานสปอร์ตนั้นจะต้องกำหนดชนิดของบริการให้เซสชั่น ชนิดของการเชื่อมต่อในชั้นนี้เช่น แบบช่องสัญญาณแบบปราศจากความผิดพลาดแบบจุดต่อจุด (Error free point to point channel) ซึ่งจะทำการส่งข้อมูลเรียงตามลำดับที่ถูกส่งออกมาจากเส้นทาง

ชั้นเซสชั่น (Session Layer)

ชั้นนี้จะทำหน้าที่สร้างเซสชั่นระหว่างผู้ให้บริการที่อยู่คนละเครื่องคอมพิวเตอร์กันให้เกิดการติดต่อสื่อสารกันได้ ความหมาย คือ การที่จะยอมให้มีการลำเลียงข้อมูลเป็นลำดับตามมา เช่นในระบบโทรศัพท์ หน้าที่บริการติดต่อผู้เรียกโดยส่งสัญญาณไปยังผู้รับจนผู้รับ รับเข้ามาจนทำให้เกิดการสนทนาเริ่มขึ้นได้ หน้าที่ดังกล่าวจะเป็นของชั้นเซสชั่น แต่หน้าที่ในการส่งสัญญาณแต่ละคำพูด จะเป็นหน้าที่ของชั้นทรานสปอร์ต

บริการอีกอย่างหนึ่งก็คือ การบริหารโทเคน (Token management) โดยโปรโตคอลบางตัวนั้นจะไม่อนุญาติให้เครื่องคอมพิวเตอร์ทั้งสองด้านที่ทำการติดต่อสื่อสารกันอยู่นั้นทำการปฏิบัติการที่เหมือนกันในเวลาเดียวกันได้ นอกจากนี้ยังมีหน้าที่อีกอย่างหนึ่งคือการซิงโครนัสคอมพิวเตอร์สองเครื่องเข้าด้วยกันเพื่อให้การติดต่อเป็นไปอย่างคล้องจองกัน

ชั้นพรีเซนเตชั่น (Presentation Layer)

ชั้นนี้มีหน้าที่จัดการเกี่ยวกับ Syntax และรูปแบบต่างๆ ของข่าวสารที่จะถูกส่งจากเครื่องคอมพิวเตอร์ออกไป เช่นการเข้ารหัสข้อมูลให้อยู่ในรูปที่เข้าใจกันทั้งผู้ส่งและผู้รับ ตัวอย่างก็คือเมื่อชั้นนี้รับข้อความจากชั้นแอปพลิเคชันซึ่งเป็นข้อความแล้ว ก็จะต้องมาทำการเข้ารหัสอักขระแต่ละตัวในข้อความนั้นให้อยู่ในรูปแบบที่ทางผู้รับรับแล้วสามารถแปลงกลับเป็นข้อความที่ถูกต้องได้เหมือนเดิม เช่น การเข้ารหัสข้อมูลแบบแอสกีเป็นต้น

ข้อความนั้นให้อยู่ในรูปแบบที่ทางผู้รับรับแล้วสามารถแปลงกลับเป็นข้อความที่ถูกต้องได้เหมือนเดิม เช่น การเข้ารหัสข้อมูลแบบแอสกีเป็นต้น

ชั้นแอปพลิเคชั่น (Application layer)

ชั้นแอปพลิเคชั่นเป็นชั้นที่มีโปรโตคอลหลากหลายโดยที่ชั้นนี้เป็นส่วนติดต่อกับผู้ใช้โดยตรง ได้แก่โฮสคอมพิวเตอร์ เทอร์มินอลหรือคอมพิวเตอร์ส่วนบุคคล เป็นต้น แอปปลิเคชั่นนี้สามารถนำเข้าหรือออกจากระบบเครือข่ายได้โดยไม่ต้องสนใจว่าจะมีขั้นตอนการทำงานอย่างไรเพราะจะมีชั้นพรีเซนเตชันรับผิดชอบอยู่แล้ว

การส่งข้อมูลในแบบจำลอง OSI

การส่งผ่านข้อมูลโดยใช้เครือข่ายตามแบบจำลองOSI โดยเริ่มจากข้อมูลเดิมที่ส่วนหัว จากนั้นจึงส่งผ่านต่อลงมายังชั้นพรีเซนเตชัน เมื่อได้รับข้อมูลก็จะแปลงให้อยู่ในรูปแบบที่มาตรฐาน เช่นแปลงข้อความที่อยู่ในรูปตัวอักษรให้อยู่ในรูปแอสกี และอาจจะเพิ่มส่วนหัวเข้าไปด้วย แล้วจึงส่งต่อลงมาให้ชั้นเซสชัน นอกจากนี้ตัวของพรีเซนเตชันนั้นประกอบด้วยอะไรบ้าง โดยจะมองข้อมูลที่ส่งออกมาเป็นเนื้อเดียวกันหมดไม่สนใจว่าอันไหนเป็นข้อมูลจริงอันไหนเป็นส่วนเติม ในทำนองเดียวกันกับชั้นบนข้อมูลและถูกส่งผ่านลงมาเรื่อยๆ จนถึงชั้นฟิสิคอล ซึ่งเป็นชั้นที่จะกระทำให้เกิดการส่งข้อมูลที่แท้จริงเพื่อส่งผ่านไปถึงเครื่องคอมพิวเตอร์ตัวรับแล้วข้อมูลก็จะถูกส่งย้อนไปขึ้นข้างบน กระบวนการก็จะกระทำย้อนกลับกับตอนที่ส่งมาจนถึงชั้นแอปพลิเคชั่น

ถึงแม้การส่งข้อมูลจริงๆนั้นจะกระทำในแนวตั้งลงมา แต่ก็สามารถมองได้เสมือนว่าแต่ละชั้นที่อยู่ระดับเดียวกันรับส่งข้อมูลกันได้โดยตรงตามแนวนอน แนวคิดนี้เป็นจุดประสงค์ในการจะทำให้การนำไปใช้ที่ชั้นต่างๆง่ายขึ้น โดยเสมือนว่าติดต่อกับชั้นในระดับเดียวกันของเครื่องคอมพิวเตอร์ตัวอื่น โดยไม่ต้องสนใจเลยว่าแท้จริงแล้วข้อมูลจะถูกส่งผ่านไปอย่างไร

2.5 สแตกโปรโตคอลทีซีพี/ไอพี (TCP/IP)

โปรโตคอล TCP/IP เป็นชื่อเรียกของชุดโปรโตคอลที่สำคัญ มีการใช้งานร่วมกันอย่างแพร่หลายตามการขยายตัวของอินเตอร์เน็ต ความจริงแล้วโปรโตคอล TCP/IP เป็นกลุ่มของโปรโตคอลหลายตัวประกอบกันเป็นชุดให้ใช้งาน โดยมีชื่อคำเต็มว่า (Transmission Control Protocol/Internet Protocol) มีการจัดแบ่งกลไกการทำงานออกเป็นชั้นๆ หรือ เลเยอร์เหมือนกับมาตรฐานโอเอสไอโมเดล และสามารถเทียบเคียงกันได้ซึ่งในแต่ละเลเยอร์ของโปรโตคอลทีซีพี/ไอพี จะประกอบไปด้วย

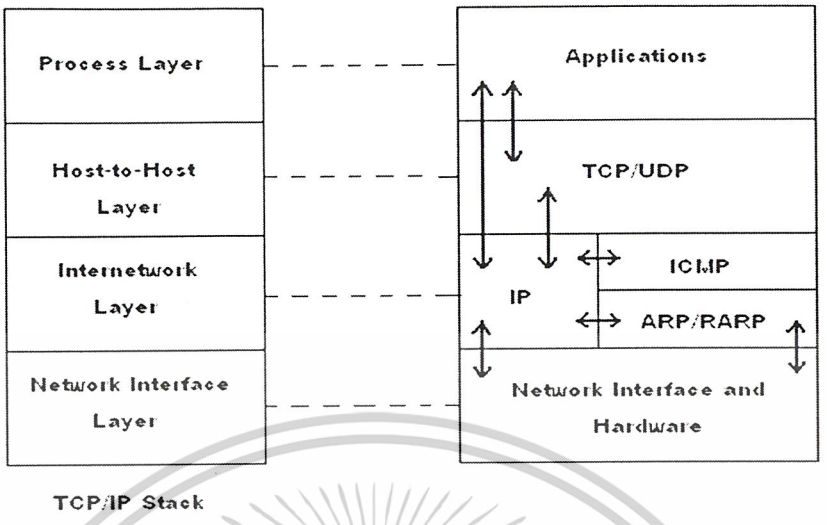
-โพรเซสเลเยอร์ (Process layer)

-โฮสทูโฮสเลเยอร์ (Host-to-Host layer)

- อินเตอร์เน็ตเวิร์กเลเยอร์ (Internetwork layer)

- เน็ตเวิร์ก อินเตอร์เฟสเลเยอร์ (Network Interface layer)

ในแต่ละกลไกลของโปรโตคอลทีซีพี/ไอพี จะมีโปรโตคอลอื่นในชุดร่วมทำงานอยู่ด้วย จึงทำให้เป็นที่มาของชื่อเรียกโปรโตคอลสแตก (Protocol Stack) เนื่องจากมีโปรโตคอลซ้อนทับกันอยู่เพื่อช่วยกันทำงานดังรูปที่2-11



รูปที่ 2-11 แสดงความสัมพันธ์ระหว่างโปรโตคอลต่างๆของ TCP/IP

จากรูปที่ 2-11 จะเห็นได้ว่ามีโปรโตคอลในแต่ละระดับซ้อนกันอยู่หลายตัวด้วยกัน ซึ่งหากเป็น OSI model จะมีข้อบังคับให้แต่ละชั้นติดต่อกันได้เฉพาะชั้นที่ติดต่อกับตนเองเท่านั้น แต่สำหรับ TCP/IP stack จะเห็นว่าบางชั้นสามารถละเลยหรือข้ามไปติดต่อกับชั้นอื่นที่ไม่ติดต่อกับตนได้

2.5.1 โพรเซส เลเยอร์ (Process layer)

จะทำงาน 2 หน้าที่เทียบได้กับ Application layer และ Presentation layer ในชั้นนี้จะรองรับการทำงานของแอพพลิเคชั่นเลเยอร์ต่างๆที่ทำงานเป็นโปรเซส อยู่ในเครื่องเซิร์ฟเวอร์ที่ให้บริการและเครื่องที่ขอใช้บริการ หรือ ไคลเอนต์(client) ซึ่งจะติดต่อกันผ่านโปรโตคอลเฉพาะแอพพลิเคชั่นอีกทีหนึ่ง เช่น เมื่อผู้ใช้งานอินเตอร์เน็ตต้องการโอนถ่ายไฟล์หรือดาวน์โหลด ข้อมูลจากเครื่องเซิร์ฟเวอร์ที่ให้บริการ โดยอาจจะเรียกใช้โปรแกรม ftp client จากนั้นโปรแกรมก็จะเรียกใช้โปรโตคอล FTP(File Transfer Protocol) เพื่อทำการโอนถ่ายไฟล์

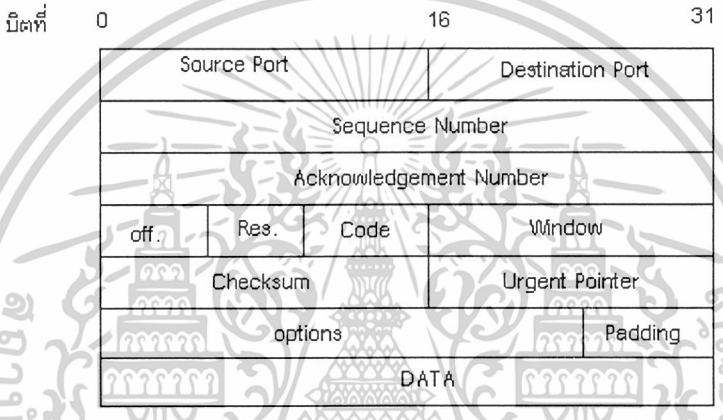
โปรโตคอลหลักๆที่ทำงานในโพรเซสเลเยอร์ ซึ่งมักจะคุ้นเคยกันดีได้แก่ FTP (File Transfer Protocol), Telnet, HTTP (Hyper Text Transfer Protocol) และ SMTP(Simple Mail Transfer Protocol)

2.5.2 โฮสทูโฮสเลเยอร์ (Host-to-Host layer)

การทำงานของเลเยอร์นี้จะมีการสร้างเชื่อมต่อกันระหว่างแอพพลิเคชั่น โดยจุดเชื่อมกันเพื่อรับส่งข้อมูลนี้เรียกว่า พอร์ต (port) หรือ ซ็อกเก็ต (socket) ในแต่ละแอพพลิเคชั่นก็สามารถสร้างการเชื่อมต่อผ่านพอร์ตได้พร้อมกันหลายแอพพลิเคชั่น ทำให้ผู้ใช้บริการสามารถเข้ามาใช้งานได้จำนวนมากในเวลาเดียวกัน ชั้นนี้จะมีโปรโตคอลที่ทำงานแตกต่างกันอยู่ 2 โปรโตคอล คือ โปรโตคอลTCP และโปรโตคอล UDP(User Datagram Protocol) ตัวโปรโตคอลจะมีแอพพลิเคชั่นเฉพาะเพื่อใช้งานแยกกันคือ แอพพลิเคชั่นที่ใช้โปรโตคอล FTP, Telnet, HTTP และ SMTP จะส่งผ่านข้อมูลโดยเรียกใช้ โปรโตคอล TCP ส่วน แอพพลิเคชั่นที่ใช้โปรโตคอล SNMPและDHCP จะส่งผ่านข้อมูลโดยเรียกใช้ โปรโตคอล UDP สำหรับโปรโตคอล DNS สามารถเรียกใช้ได้ทั้ง TCP และ UDP

โปรโตคอลทีซีพี

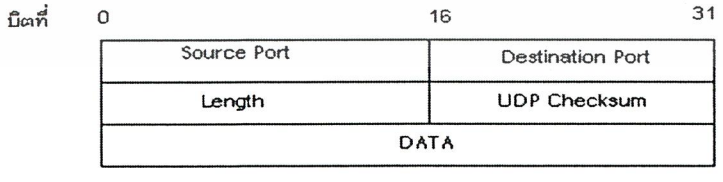
โปรโตคอล TCP(Transmission Control Protocol) เป็นโปรโตคอลที่มีการรับส่งข้อมูลแบบ stream oriented protocol หมายความว่าการรับส่งจะไม่คำนึงถึงข้อมูลที่ส่งไป แต่จะแบ่งข้อมูลเป็นส่วนย่อยๆก่อน แล้วจึงจะส่งไปยังปลายทางต่อเนื่องไปเป็นลำดับข้อมูล ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไป ก็จะส่งส่วนนั้นใหม่อีกครั้ง โดยการติดต่อระหว่างกันจะต้องเป็นแบบ connection-oriented คือต้องมีการสร้างติดต่อกันเป็น sesion ทั้ง 2 ด้านเสียก่อน แล้วจึงจะรับส่งข้อมูลไปได้พร้อมกัน(full duplex) และในระหว่างการรับส่งข้อมูล โปรโตคอลTCP จะเพิ่มขบวนการสอบทานข้อมูลเพื่อให้ข้อมูลมีความถูกต้องไม่ผิดพลาดไปจากเดิม โดยการส่งสัญญาณสอบทานข้อมูล(acknowledgement) ถึงแม้ความเชื่อถือของการส่งผ่านข้อมูลโดยโปรโตคอลนี้จะมีมากแต่ก็ต้องอาศัยทรัพยากรระบบมากในการทำงานเช่นกัน



รูปที่ 2-12 แสดงรูปแบบของ TCP แพ็คเกจ

โปรโตคอลยูดีพี

โปรโตคอลยูดีพี(User Datagram Protocol)ในการรับส่งข้อมูลผ่านโปรโตคอลนี้จะเป็นแบบที่ทั้งสองด้านไม่ต้องอาศัยการสร้างช่องทางการเชื่อมต่อ(connectionless) ไม่มีการสอบทานข้อมูล (acknowledgement) และไม่มีการส่งใหม่เมื่อเกิดความผิดพลาด เมื่อเป็นเช่นนี้แอพพลิเคชั่นใดที่ต้องอาศัยโปรโตคอลยูดีพีจึงต้องสร้างขบวนการตรวจสอบข้อมูลขึ้นมาเอง



รูปที่ 2-13 แสดงรูปแบบ UDP แพ็คเกจ

2.5.3 อินเตอร์เน็ตเวิร์กเลเยอร์ (Internetwork layer)

จะมีโปรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายคือ โปรโตคอล IP (Internet Protocol) นอกจากนี้ยังมีอีก 2 ตัวคือ โปรโตคอล Internet Control Message Protocol (ICMP) และ โปรโตคอล Address Resolution Protocol (ARP)

โปรโตคอลไอพี

ทำหน้าที่ให้บริการส่งผ่านข้อมูลที่มาจาก Host-to-Host layer เพื่อส่งข้ามไปยังเครือข่ายใดๆได้อย่างถูกต้อง แม้ว่าจะมีเครือข่ายเชื่อมต่อกันอยู่หลายเครือข่ายเนื่องจากโปรโตคอลไอพี มีข้อมูลตำแหน่งไอพี ปลายทาง โดยทำงานร่วมกับอุปกรณ์ Router ตัวโปรโตคอลจะทำงานแบบ packet switching คือมีการส่งข้อมูลผ่านสวิทซ์(switch) ไปยังปลายทาง โดยข้อมูลจะเดินทางผ่านสวิทซ์เรื่อยๆไปจนถึงปลายทาง ซึ่งในข้อมูลของโปรโตคอลจะมีหมายเลข IP ปลายทางที่จะส่งข้อมูลไปและเมื่อถึงเครือข่ายปลายทางก็จะมีกลไกแปลงหมายเลข IP ให้เป็นฮาร์ดแวร์ประจำเครื่องที่ถูกต้องอีกทีหนึ่งด้วยโปรโตคอล ARP

โปรโตคอลไอซีเอมพี (ICMP)

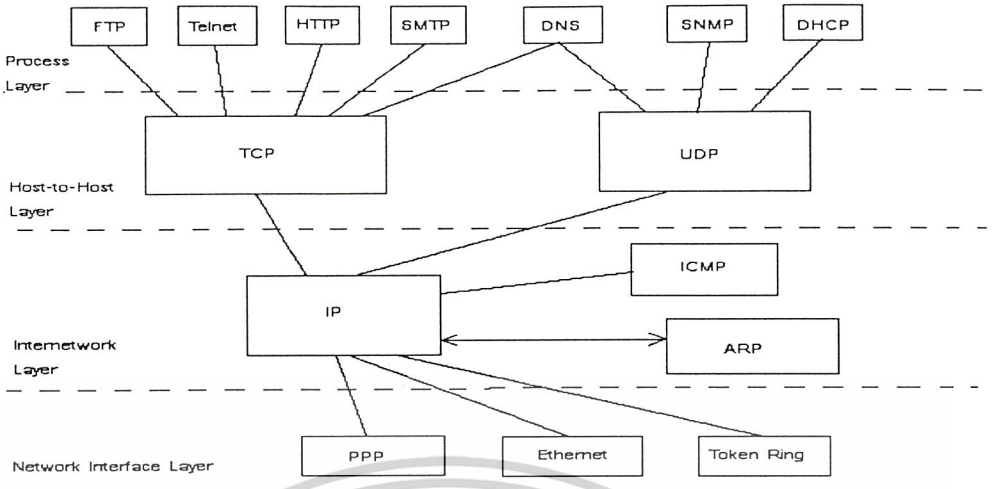
หน้าที่หลักของโปรโตคอล ICMP(Internet Control Message Protocol) คือการแจ้งหรือแสดงข้อความจากระบบเพื่อบอกให้ทราบว่าเกิดอะไรขึ้นกับการส่งผ่านข้อมูลนั้น ซึ่งส่วนใหญ่ก็คือส่งไปไม่ได้หรือปลายทางรับไม่ได้ เป็นต้น นอกจากนี้โปรโตคอลนี้ยังถูกเรียกใช้จาก เครื่องเซิร์ฟเวอร์และRouter อีกด้วย เพื่อแลกเปลี่ยนข้อมูลที่ใช้ควบคุม ส่วนรูปแบบการทำงานจะควบคู่กับโปรโตคอล IP ในระดับเดียวกันและข้อความต่างๆที่แจ้งให้ทราบจะถูกผนึกอยู่ภายในข้อมูลของ IP (IP diagram) อีกทีนึง

โปรโตคอลเออาพี (ARP)

โปรโตคอล ARP(Address Resolution Protocol) ถูกเรียกใช้งานโดยโปรโตคอล IP เพื่อช่วยแปลงหมายเลข IP ไปเป็นหมายเลขฮาร์ดแวร์ปลายทาง ตัวอย่างเช่น เวบเซอร์เวอร์เชื่อมต่ออยู่กับอินเตอร์เน็ต และในการเชื่อมต่อจะต้องอาศัย Network Interface Card (NIC) หรือ LAN card ซึ่งจะมีหมายเลขเฉพาะประจำฮาร์ดแวร์ที่ไม่ซ้ำกับใครเพื่ออ้างอิงการส่งข้อมูลในโครงข่าย แต่เมื่อมาใช้งานโปรโตคอล TCP/IP ก็จะต้องมีการกำหนดหมายเลข ไอพีแอดเดรสประจำตัวเพื่อใช้อ้างอิงกัน และโปรโตคอล ARPจะทำหน้าที่แปลงค่าหมายเลข IP ให้เป็นหมายเลขฮาร์ดแวร์จริงให้ในระดับการทำงานที่ อินเตอร์เน็ตเวิร์กเลเยอร์ซึ่งกลไกการแปลงนี้เรียกว่า address resolution

2.5.4 เน็ตเวิร์ก อินเตอร์เฟส เลเยอร์ (Network Interface layer)

เนื่องจากในด้านกายภาพของเครือข่ายนั้น มีหลายวิธีการและหลายรูปแบบในการเชื่อมต่อระบบให้เป็นเครือข่าย เลเยอร์ชั้นนี้จะทำการแปลงข้อมูลหรือ IP ไดอะแกรมให้อยู่ในรูปแบบที่เหมาะสมเป็นสัญญาณไฟฟ้าส่งไปยังปลายทาง ไม่ว่าจะเป็นการใช้เครือข่ายใยแก้วนำแสง เครือข่าย Ethernet Token Ring หรือ อื่นๆ



รูปที่ 2-14 แสดงโครงสร้างรวมของโปรโตคอลทีซีพี/ไอพีในแต่ละชั้น

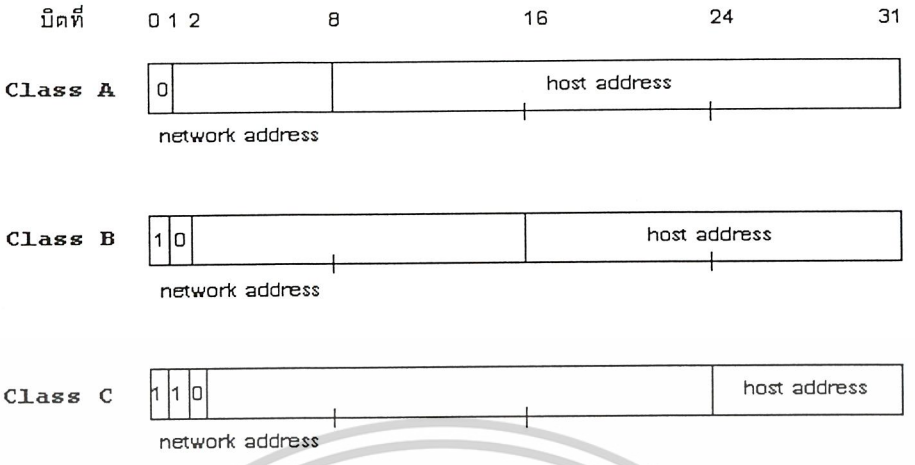
2.6 ไอพีแอดเดรส (IP Address)

ไอพีแอดเดรสถูกกำหนดขึ้นมาให้เป็นหมายเลขอ้างอิงประจำตัวของอุปกรณ์ต่างๆที่เชื่อมต่ออยู่ในเครือข่ายอินเตอร์เน็ตโดยการกำหนดไอพีแอดเดรสให้อุปกรณ์แต่ละเครื่องต้องไม่ซ้ำกัน ไอพีแอดเดรสนี้จะไม่ถูกผูกติดไว้กับตัวฮาร์ดแวร์จึงสามารถกำหนดใหม่หรือแก้ไขเปลี่ยนแปลงได้เมื่อมีการปรับเปลี่ยนฮาร์ดแวร์เนื่องจากเป็นการกำหนดขึ้นโดยซอฟต์แวร์แตกต่างกับหมายเลขแมคแอดเดรสที่เป็นหมายเลขประจำตัวของฮาร์ดแวร์

การทำงานของโปรโตคอลไอพี จำเป็นต้องอาศัยไอพีแอดเดรส เพื่อระบุและอ้างถึงอุปกรณ์ต่างๆที่ต่ออยู่ในเครือข่ายไม่ว่าจะเป็นเว็บเซอร์ฟเวอร์, เมล์เซอร์ฟเวอร์, อุปกรณ์เราเตอร์ หรืออื่นๆ ไอพีแอดเดรสจะเป็นค่าตัวเลขขนาด 32 บิตหรือ 4 ไบต์ ถูกแบ่งออกเป็นส่วนละ8 บิต เป็น 4 ส่วนและคั่นแต่ละส่วนด้วยเครื่องหมายจุด(.) ตัวเลขในแต่ละส่วนจะมีได้ตั้งแต่ 0 ถึง 255 ตัวอย่างของ ไอพีแอดเดรส เช่น 205.144.78.1 เพื่อไม่ให้เกิดปัญหาไอพีแอดเดรส ซ้ำกันจะมีหน่วยงานกลาง InterNIC หรือ Internet Network Information Center ที่ทำหน้าที่กำหนดแจกจ่ายหมายเลขไอพีให้กับองค์กรต่างๆให้เกิดประโยชน์สูงสุด

ไอพีแอดเดรสขนาด 4 ไบต์สามารถแยกได้เป็น 2 ส่วนย่อยคือ ส่วนแรกเป็นหมายเลขของเครือข่าย (network address) และส่วนที่สองเป็นหมายเลขของเครื่องลูกข่าย(host address) ในแต่ละเครือข่ายจะมีความแตกต่างกันระหว่างจำนวนเครือข่ายหลักและลูกข่ายย่อย ได้มีการจัดแบ่งลำดับชั้นของเครือข่ายไว้เป็น 5 ลำดับ คือ A B,C,D และ E

คลาส A :ใช้ 1 ไบต์แรกเป็นหมายเลขเครือข่าย และ 3 ไบต์ที่เหลือเป็นหมายเลขเครื่อง ไบต์แรกมีค่าตั้งแต่ 0-127 จะเป็นลักษณะ net.host.host.host เหมาะสำหรับเครือข่ายขนาดใหญ่มากๆ เพราะสามารถรองรับจำนวนเครื่องได้ในเครือข่าย 16,777,214 เครื่อง ตัวอย่างเช่น ค่าไอพีแอดเดรส คลาส A เป็น 121.7.23.3 หมายถึง เครือข่ายที่ 121 หมายเลขเครื่องคือ 7.23.3



รูปที่ 2-15 แสดงการแบ่งส่วนเครือข่ายและลูกข่ายใน คลาส A,B,C ของไอพีแอดเดรส

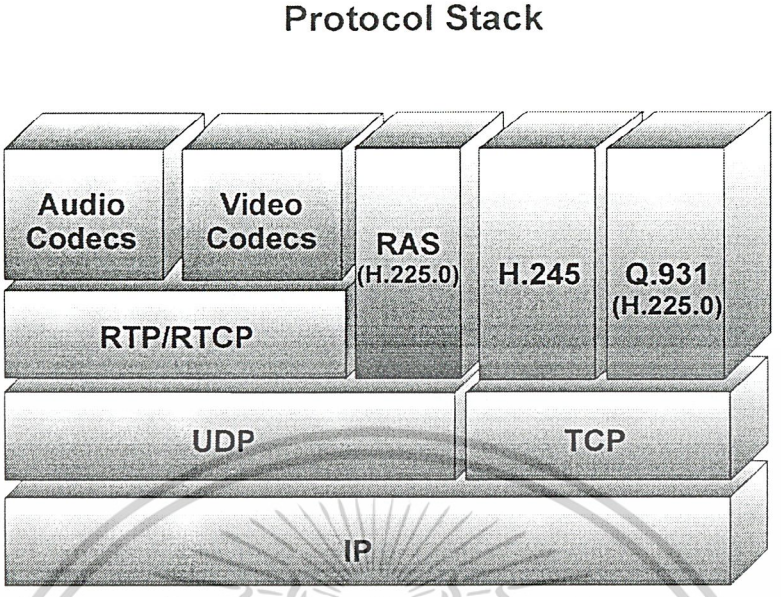
คลาส B : ใช้ 2 ไบต์แรกเป็นหมายเลขเครือข่ายและ 2 ไบต์ที่เหลือเป็นหมายเลขเครื่อง ไบต์แรกจะมีค่าค่าตั้งแต่ 128-191 จะเป็นลักษณะ net.net.host.host เหมาะสำหรับเครือข่ายขนาดใหญ่แต่เล็กกว่าคลาส A รองรับจำนวนเครื่องในเครือข่ายได้ 64,516 เครื่อง ตัวอย่างเช่น ค่าไอพีแอดเดรส คลาส B เป็น 137.103.210.3 หมายถึง เครือข่ายที่ 137.103 หมายเลขเครื่องคือ 210.3

คลาส C : ใช้ 3 ไบต์แรกเป็นหมายเลขเครือข่ายและ 1 ไบต์ที่เหลือเป็นหมายเลขเครื่อง ไบต์แรกจะมีค่าตั้งแต่ 192-223 จะเป็นลักษณะ net.net.net.host เหมาะสำหรับองค์กรขนาดกลางถึงเล็ก รองรับจำนวนเครื่องในเครือข่ายได้ 254 เครื่อง ตัวอย่างเช่น ค่าไอพีแอดเดรส คลาส C เป็น 202.182.255.3 หมายถึงเครือข่ายที่ 202.182.255 หมายเลขเครื่องคือ 3

คลาส D : เป็นการกำหนดไอพีแอดเดรสสำรองไว้สำหรับส่งข้อมูลแบบ multicast ซึ่งจะไม่มีการแจกจ่ายใช้งานทั่วไป

คลาส E : เป็นไอพีแอดเดรสสำหรับกรณีพิเศษในงานพัฒนาและทดสอบไม่มีการกำหนดให้ใช้งานทั่วไป

2.7 โปรโตคอลอาร์ทีพี (RTP)



รูปที่ 2-16 แสดงแสตกของโปรโตคอล

โปรโตคอลอาร์ทีพีเป็นโปรโตคอลที่สร้างขึ้นสำหรับรองรับบริการส่งข้อมูลจากต้นทางไปยังปลายทางด้วยคุณลักษณะของข้อมูลแบบเวลาจริง เช่น การส่งสัญญาณโต้ตอบของเสียงและวีดีโอการบริการนี้จะประกอบไปด้วยการบ่งบอกชนิดของข้อมูล การลำดับหมายเลข กำกับช่วงเวลา ติดตามขบวนการส่งข้อมูลการประยุกต์ใช้งานโดยทั่วไป โปรโตคอลอาร์ทีพีจะทำงานบนโปรโตคอลยูดีพีจะทำให้มันมีการรวบรวมและตรวจสอบความผิดพลาดของข้อมูลโดยตัวโปรโตคอลจะสนับสนุนการทำงานต่างๆ ของชั้นทรานสปอร์ตและเนตเวอร์กเลเยอร์ ซึ่งอาร์ทีพีถูกออกแบบมาเพื่อจุดประสงค์ดังกล่าว เพราะเมื่อใช้โปรโตคอลทีซีพีนั้นมีปัญหาเรื่องเสียโอเวอร์เฮดในการแจ้งข้อผิดพลาดกลับทุกครั้งเมื่อข้อมูลส่งเกิดการผิดพลาด โปรโตคอลอาร์ทีพีอาจจะถูกนำไปใช้กับโครงข่ายหรือโปรโตคอลในระดับล่างที่มีความเหมาะสมได้เช่นกันและยังจะ สนับสนุนการส่งข้อมูลไปยังหลายๆปลายทาง ด้วยการใช้รูปแบบการกระจายตัว

โปรโตคอลอาร์ทีพีเองจะไม่มีกลไกการรับรองในเรื่องเวลาการส่งข้อมูลหรือรับรองในคุณภาพของบริการ(QOS) แต่จะพึ่งพาให้เลเยอร์ชั้นต่ำกว่าทำหน้าที่นี้แทน ไม่รับรองการส่งข้อมูลหรือป้องกันข้อมูลสูญหาย นั่นหมายความว่าเครือข่ายระดับล่างนั้นต้องมีความน่าเชื่อถือและชุดข้อมูลที่ส่งไปจะต้องมีค่าลำดับหมายเลข และการลำดับหมายเลขที่อยู่ในโปรโตคอลนี้จะทำให้ด้านรับสามารถที่จะสร้างชุดข้อมูลของผู้ส่งขึ้นมาใหม่ให้เป็นไปตามความถูกต้องและต่อเนื่อง หมายเลขของลำดับอาจถูกนำมาพิจารณาตำแหน่งลำดับความถูกต้องในชุดข้อมูล

สำหรับโปรโตคอลอาร์ทีพีจุดมุ่งหมายหลักเองจะถูกออกแบบมาให้เหมาะสมกับการใช้ทำงานโต้ตอบทางสื่อมัลติมีเดียแต่ไม่มีการจำกัดหากจะนำไปใช้ในงานเฉพาะอื่น เช่น ใช้จัดเก็บข้อมูลแบบต่อเนื่อง แปลงชุดข้อมูลโต้ตอบ, ประยุกต์ใช้งานในการควบคุมและการวัด

อาร์ทีพีประกอบด้วย 2 ส่วนหลักๆ คือ

-โปรโตคอลอาร์ทีพี สำหรับนำพาข้อมูลที่มีคุณลักษณะของเวลาจริง

-โปรโตคอลอาร์ทีซีพี (RTCP หรือ RTP control protocol) จัดให้มีหน้าที่ตรวจตราคุณภาพในการบริการและนำพาข้อมูลที่เกี่ยวข้องในรูปแบบของการควบคุมชุดข้อมูลที่รวมกันอยู่และลักษณะส่วนท้ายของโปรโตคอลอาทีซีพีเป็นลักษณะการควบคุมที่มีความยืดหยุ่นสูง เช่นหากไม่มีการแจ้งการควบคุมและใช้งานจากสมาชิก มันก็ไม่มีจำเป็นที่จะต้องใช้การควบคุมข้อมูลแบบเต็มรูปแบบโดยจะเปลี่ยนไปใช้การควบ คุมพิเศษเฉพาะบาง ส่วน

อาร์ทีพีแสดงให้เห็นว่ามันเป็นโปรโตคอลรูปแบบใหม่ตามหลักการจัดเฟรมและมีการร่วมกันประมวลผลในชั้นเลเยอร์ที่เสนอขึ้นโดยนายคาร์กและเทนเนนฮอร์สที่ตั้งใจให้มันเป็นรูปแบบที่หลากหลายในการประยุกต์ใช้งานกับข้อมูล ในรูปแบบต่างๆดังต่อไปนี้

1.รูปแบบพื้นฐานในการประชุมทางเสียง

โดยใช้บริการไอพีมัลติแคสกระจายข้อมูลไปยังอินเตอร์เน็ตในรูปแบบการติดต่อสื่อสารทางเสียงโดยใช้หมายเลขแอดเดรสแบบกระจายและพอร์ตเป็นคู่โดยใช้พอร์ทนึงเป็นข้อมูลเสียงและอีกพอร์ทใช้เป็นพอร์ทข้อมูลควบคุม(อาร์ทีซีพี) ตำแหน่งและพอร์ตจะมีการสร้างข้อมูลให้กระจายไปยังผู้ที่มีส่วนร่วม

ข้อมูลเสียงที่ประยุกต์ใช้ในงานการประชุมเสียงจะเป็นการส่งข้อมูลเสียงโดยการสไลด์เวลาของผู้ที่มีส่วนร่วมในการประชุมเป็นช่วงสั้นประมาณ 20 มิลลิวินาทีแต่ละช่วงของข้อมูลเสียงจะดำเนินกระบวนการโดยอาร์ทีพีเฮดเดอร์ แล้วข้อมูลจะถูกบรรจุเข้าไปในยูดีพีแพ็คเกจ อาร์ทีพีเฮดเดอร์ จะมีการบ่งบอกว่าเป็นชนิดข้อมูลอะไร ในการเข้ารหัสเสียง เช่น พีซีเอ็ม(PCM) เอดีพีซีเอ็ม(ADPCM) หรือ แอลพีซี(LPC) ที่ถูกบรรจุในแต่ละแพ็คเกจทำให้ผู้ส่งสามารถเปลี่ยนแปลงช่วงเวลาวิธีการเข้ารหัสสัญญาณในการประชุมให้เหมาะสมเช่นผู้ร่วมประชุมบางท่านอาจจะเชื่อมต่อด้วยลิงค์ที่มีอัตราการส่งผ่านข้อมูลต่ำ

อินเตอร์เน็ตก็เหมือนโครงข่ายแพ็คเกจอื่นๆ ในกรณีที่มีการสูญหายและเปลี่ยนตำแหน่งของแพ็คเกจและ เกิดความล่าช้าซึ่งเป็นตัวแปรที่ขึ้นอยู่กับเวลา ในอาร์ทีพีเฮดเดอร์จะมีการบรรจุข้อมูลของเวลาและลำดับข้อมูลที่ให้สิทธิกับด้านรับให้สามารถสร้างโครงสร้างของเวลาขึ้นมาดังเดิม โดยเหมือนแหล่งข้อมูลตัวอย่างเช่นแหล่งข้อมูลของอาร์ทีพีแพ็คเกจในการประชุม ลำดับของข้อมูลที่ผู้รับสามารถนำมาประเมินว่ามีจำนวนข้อมูลเท่าไรที่มีการสูญหายไป และเมื่อสมาชิกเข้าร่วมหรือออกจากกลุ่มมันจะเป็นประโยชน์ที่จะรับรู้ว่ามีใครเข้าร่วมอยู่บ้างณ เวลานี้ และมีความสามารถรับส่งข้อมูลได้ดีแค่ไหน ในส่วนนี้จะมีการเพิ่มชื่อของผู้ใช้บนอาร์ทีซีพี(พอร์ตควบคุม) การรับข้อมูลจะมีการรายงานให้ทราบสถานะภาพผู้รับฟังและการใช้งานการถอดรหัสการควบคุม ในการเพิ่มส่วนของชื่อผู้ใช้ และบ่งบอกข้อมูลอื่นๆที่อาจจะรวมหัวข้อไปกับแบนด์วิธที่จำกัด และจะมีการส่งอาทีซีพีบาย(BYE) แพ็คเกจเมื่อมีการยกเลิกการประชุม

2.การประชุมด้วยภาพและเสียง

สื่อสารรวมกันทั้งข้อมูลภาพและเสียงที่ใช้ในการประชุม โดยมันจะถูกจัดส่งโดยแยกส่วนอาร์ทีพีและอาทีซีพีแพ็คเกจในแต่ละตัวกลางโดยการใช้คู่ของพอร์ตยูดีพีที่แตกต่างกัน หรืออาจใช้ร่วมกับการระบุตำแหน่งแบบหลายจุดโดยที่จะไม่มีการเชื่อมต่อร่วมกันโดยตรงในระดับของอาร์ทีพีระหว่างส่วนของเสียง

และวีดีโอยกเว้นการประชุมที่มีการกำหนดชื่อของทั้งสองส่วนที่เหมือนกันในอาร์ทีซีพีแพ็คเกจเพื่อให้มีการรวมข้อมูลกัน

3.มิกเซอร์และทรานสเลเตอร์ (Mixer and Translators)

ในกรณีนี้เราจะสมมติว่าทุกส่วนต้องการรับข้อมูลมีเดียในรูปแบบเดียวกัน พิจารณาในกรณีผู้เข้าร่วมประชุมหนึ่งอยู่ในพื้นที่ซึ่งเชื่อมต่อข้อมูลได้ที่ความเร็วต่ำไปยังส่วนหลักที่มีผู้ประชุมอื่นรวมกันอยู่ด้วยโครงข่ายความเร็วสูง มันจะเป็นตัวนำในการบังคับให้ทุกคนใช้แบนด์วิธที่ต่ำลงมา และสร้างการเข้ารหัสเสียงเพื่อรองรับคุณภาพใหม่โดยอาร์ทีพีในส่วนที่เรียกว่ามิกเซอร์อาจจะนำมาวางตรงตำแหน่งที่ใกล้กับพื้นที่แบนด์วิธต่ำโดยมิกเซอร์นี้จะทำการรีซิงโครไนซ์กับการเข้ามาของแพ็คเกจเสียงไปยังการสร้างขึ้นใหม่ของผู้ส่งใน 20 มิลลิวินาที จะทำการรวมสายข้อมูลเสียงไปเป็นสายข้อมูลเดี่ยว การแปลงการเข้ารหัสเสียงไปยังแบนด์วิธที่ต่ำกว่า และทำการส่งกลับผ่านการเชื่อมต่อแบบความเร็วต่ำ โดยแพ็คเกจอาจจะส่งไปยังผู้รับเดี่ยวหรือผู้รับหลายท่านด้วยแอดเดรสที่ต่างกัน

อาร์ทีพีจะมีความสามารถในการบ่งชี้ชนิดข้อมูล,ลำดับเลขข้อมูล,ระบุเวลาและตรวจตราการส่งข้อมูล สามารถลำดับเลขข้อมูลได้ใหม่หากข้อมูลนั้นมาถึงด้านรับโดยไม่เป็นไปลำดับ การมีเลขลำดับทำให้สามารถค้นหาแพ็คเกจข้อมูลที่สูญหายได้ การระบุเวลามีเพื่อใช้ในคุณสมบัติการเล่นมีเดีย ข้อมูลที่มาถึงทางด้านรับจะถูกตรวจสอบอย่างต่อเนื่องโดยโปรโตคอลอาร์ทีซีพี ซึ่งอยู่ในเลเยอร์เดียวกับอาทีพีเพื่อ ให้มีการปรับการเข้ารหัสและแสดงพารามิเตอร์ในการส่งข้อมูล ตัวอย่างเช่นถ้าพบว่ามีแพ็คเกจสูญหายจำนวนมาก มันจะแจ้งให้มีการส่งข้อมูลด้วยอัตราเร็วที่ต่ำลง

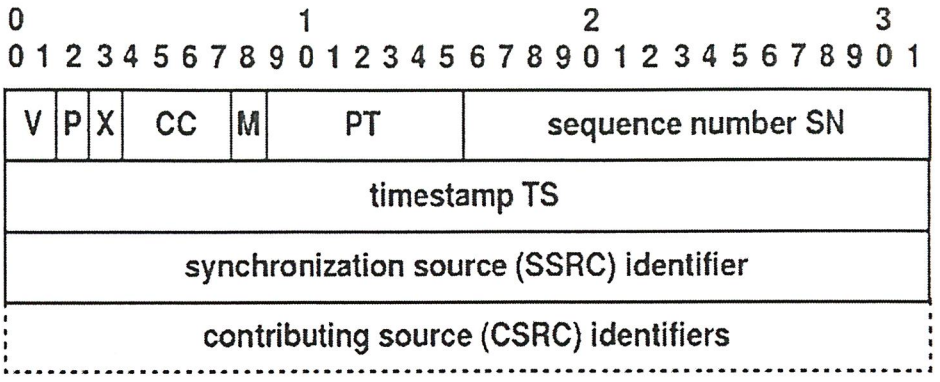
จำนวนเลขที่ระบุอยู่ในโปรโตคอลจะเป็นเลขที่อ้างอิงถึงบิตที่อยู่ในรูปแบบออกเตจโดยเลขลำดับไบต์นี้จะรู้จักกันในชื่อบิ๊กเอนเดียน (big-endian) ทุกๆส่วนของข้อมูลเฮดเดอร์จะถูกจัดเรียงเป็นช่วงความยาวแบบทั่วไป เช่น 16, 32, 64, บิตโดยจัดให้เป็นไบต์จำนวนคู่ ระบบเวลาจะถูกแสดงโดยรูปแบบการประทับเวลาของ เอ็นทีพี (NTP-Network Time Protocol)ซึ่งเกี่ยวเนื่องกันกับเวลายูทีซี(UTC) อัตราการกำหนดเวลาของเอ็นทีพีเต็มที่คือ 64 บิต โดยมีส่วนหลักใน 32 บิตแรก และแยกย่อยใน 32 บิตหลังตามมา

อาร์ทีพีเฮดเดอร์ (RTP header)

รูปแบบส่วนหัวของโปรโตคอลอาร์ทีพีจะถูกกำหนดให้ประกอบไปด้วยชุดข้อมูล โดยที่ 12 ออคเตจ แรกจะเกิดขึ้นในทุกๆอาร์ทีพีแพคเกจ ขณะที่ส่วนของ CSRC จะแสดงขึ้นเฉพาะในบางแพ็คเกจ เมื่อมันทำหน้า ที่เป็นมิกเซอร์ แต่ละส่วนนั้นมีความหมายได้ดังนี้

Version (V):จำนวน 2 บิต ส่วนนี้จะแสดงเวอร์ชันของ โปรโตคอลอาร์ทีพี ซึ่งค่าปัจจุบันเป็นค่าเท่ากับ2 โดยค่า 1 ถูกใช้ให้เป็นเวอร์ชั่นทดลอง

Padding (P):จำนวน 1 บิต ส่วนนี้ถ้าถูกเซตเป็น 1 ชุดข้อมูลจะถูกบรรจุด้วยส่วนเพิ่มเติมที่เป็นออคเตจในตอนท้าย แสดงให้เห็นว่าเมื่อจบเฮดเดอร์แล้วยังไม่ใช่ส่วนของชุดข้อมูลที่จะตามมา ส่วนนี้มีความสำคัญในการทำอัลกอริทึมซึ่งต้องการกำหนดขนาดของบล็อกข้อมูล เพื่อนำพาอาทีพีหลายๆแพ็คเกจในเลเยอร์ที่อยู่ต่ำลงมา



รูปที่ 2-17 แสดงชุดข้อมูลส่วนหัวของโปรโตคอลอาร์ทีพี

Extention (X):จำนวน 1 บิต ส่วนนี้ถ้าถูกเซตเป็น 1 แสดงว่าส่วนหัวของข้อมูลจะยังมีส่วนขยายตามมาอีก

CSRC count (CC):จำนวน 4 บิต ส่วนนี้จะเป็นส่วนที่บ่งบอกจำนวนCSRCที่จะเพิ่มเติมมาในส่วนของเฮดเดอร์ที่ได้กล่าวไว้ก่อนหน้านี้

Mark (M):จำนวน 1 บิต เป็นบิตที่ถูกกำหนดขึ้นเพื่อแสดงให้ทราบช่วงระยะขอบเขตของแพ็คเกจสตรีมแต่ละเฟรม เช่นอาทีพีมาร์คบิตจะถูกเซตเป็น 1 ถ้าแพ็คเกจบรรจุจำนวนบิตที่เหลือต่อจากเฟรมที่อยู่ก่อนหน้า

Payload Type (PT):จำนวน 7 บิต ส่วนนี้จะแสดงชนิดของชุดข้อมูลที่ถูกนำพาโดยอาทีพีแพ็คเกจว่าเป็นรูปแบบใดเช่น MPEG,M-JPEG,PCM

Sequence Number (SN):จำนวน 16 บิต เป็นส่วนแสดงเลขลำดับของชุดข้อมูลอาร์ทีพีที่ได้ทำการส่ง เพื่อให้ฝั่งรับสามารถตรวจเช็คข้อมูลที่สูญหายและเรียงลำดับแพ็คเกจใหม่ ค่าเริ่มต้นของเลขลำดับจะถูกสุ่มขึ้นเพื่อเริ่มต้น

Timestamp (TS): จำนวน 32 บิต ส่วนนี้จะบรรจุข้อมูลเวลาที่แสดงการสุ่มค่าคงที่ของออกเตจแรกในข้อมูลแพ็คเกจอาทีพี และการสุ่มค่าคงที่นี้จะต้องกระทำโดยนาฬิกาที่เป็นลิเนียร์ที่สามารถซิงโครไนซ์และคำนวนการคลาดเคลื่อนของเวลาทางด้านรับได้ โดยค่าเริ่มต้นจะเกิดขึ้นโดยการสุ่ม

Synchronization Source (SSRC): จำนวน 32 บิต ส่วนนี้จะเป็นส่วนซิงโครไนส์แหล่งข้อมูลที่ทำการส่งและรับโดยอาทีพีเซสชั่นแบบการสุ่ม และเป็นส่วนช่วยในการแก้ปัญหาการชนกันของข้อมูล

Contributing Source (CSRC): 0 - 15 ชุด แต่ละชุดมี 32 บิต ส่วนนี้เป็นข้อมูลเพิ่มเติมที่ใช้ในการบ่งบอกตำแหน่งภายในชุดข้อมูลที่มีรูปแบบต่างกัน จำนวนของส่วนนี้จะแสดงไว้ที่CC หน้าที่ของส่วนนี้เช่นบอกตำแหน่งที่แตกต่างกันของสัญญาณเสียงที่เชื่อมต่อกันอยู่ในชุดของข้อมูล

2.8 โปรโตคอลอาร์ทีซีพี (RTP Control Protocol)

อาร์ทีซีพีเป็นโปรโตคอลที่ใช้งานร่วมกับอาร์ทีพีทำงานบนพื้นฐานของแพ็คเกจควบคุมที่ส่งข้อมูลในแบบคาบเวลา ให้กับทุกส่วนที่มีผู้ใช้งานร่วมกัน โดยวิธีการซึ่งคล้ายกับการส่งแพ็คเกจของข้อมูลโปรโตคอลที่อยู่ชั้นต่ำกว่าจะทำการรวมข้อมูลและคอนโทรลแพ็คเกจ เข้าด้วยกัน โดยอาทีซีพีมีหน้าที่การทำงาน 4 ข้อดังนี้

1.หน้าที่หลักคือทำการรายงานผลคุณภาพของการส่งผ่านข้อมูล และ เกี่ยวข้องกับฟังก์ชันการทำงานในส่วนของทรานสปอร์ตโปรโตคอล โดยทำการควบคุมการไหลและบีบอัดข้อมูลของโปรโตคอลในชั้นทรานสปอร์ต อื่นๆ ทำหน้าที่ส่งรายงานผลการรับข้อมูลไปยังผู้ร่วมใช้งานทุกคน ให้สิทธิ์แก่ผู้เฝ้าสังเกตการณ์ประเมินค่าปัญหาที่เกิดขึ้น ด้วยการกระจายตัวที่เหมือนกลไกการมัลติแคสไอพี เป็นผู้ให้บริการที่3 ที่ทำหน้าที่ตรวจตราปัญหาที่เกิดกับเครือข่าย

2.อาร์ทีซีพีจะทำหน้าที่นำพาข้อมูลที่มีการระบุแหล่งของอาร์ทีพี ที่เรียกว่าแคนโนนิคอลเนมหรือซีเนม (canonical name or CNAME) เมื่อการระบุ SSRC อาจเกิดเปลี่ยนแปลงถ้าหากค้นพบข้อขัดข้องหรือ โปรแกรมต้องเริ่มต้นทำงานใหม่ ทางด้านรับต้องการซีเนมที่จะทำการเก็บรักษาข้อมูลในแทร็ก (track)ของผู้ใช้งานแต่ละคน และเพื่อเข้าไปมีส่วนร่วมในชุดข้อมูลรวมที่สัมพันธ์กันกับส่วนของอาร์ทีพี เช่นตัวอย่างการซิงโครไนซ์สัญญาณเสียงและวีดีโอ

3.ประมาณถึงจำนวนผู้เข้าร่วมใช้งานมากที่สุด โดยผู้เข้าร่วมแต่ละคนจะส่งแพ็คเกจควบคุมไปยังคนอื่นๆ โดยแต่ละคนสามารถทราบถึงจำนวนผู้เข้าร่วมใช้งานรวม ซึ่งจำนวนเลขนี้สามารถนำไปคำนวณอัตราแพ็คเกจที่ทำการส่งไปแล้วได้

4.นำพาข้อมูลการควบคุมส่วนย่อย เช่น ตัวอย่างความสามารถในการระบุรายชื่อผู้เข้าร่วมใช้งานที่สามารถแสดงผลบนอินเตอร์เฟสของผู้ใช้ อาทีซีพีเป็นช่องทางที่สะดวกในการส่งข้อมูลไปยังผู้ร่วมใช้งานทุกคน

รูปแบบของแพ็คเกจอาร์ทีซีพี

แพ็คเกจการรายงานอาร์ทีซีพีสามารถเป็นได้ทั้งชนิดผู้ส่งและผู้รับ ด้านผู้รับจะส่งSR(sender reports) ถ้ามันเป็นผู้มีส่วนร่วมในการเข้าประชุมของเซสชั่น มิฉะนั้นแล้วมันก็จะส่งเป็นรายงานการรับในส่วนเพิ่มเติมไปยัง SR และ RRแพ็คเกจ อาทีซีพียังมีชนิดแพ็คเกจอื่นอีกเช่น SDES(Source Description), BYE และ APP(Applicstion defined) โดยค่าที่แสดงชนิดแพ็คเกจจะแสดงไว้ถัดไป

แสดงรายละเอียดของอาทีซีพีแพ็คเกจที่แสดงข้อมูลการควบคุมต่างๆดังนี้

SR:(Sender report) รายงานผู้ส่ง ในเรื่องสถานะภาพการส่งและรับจากผู้เข้าร่วมใช้งานที่ทำหน้าที่ส่งข้อมูล

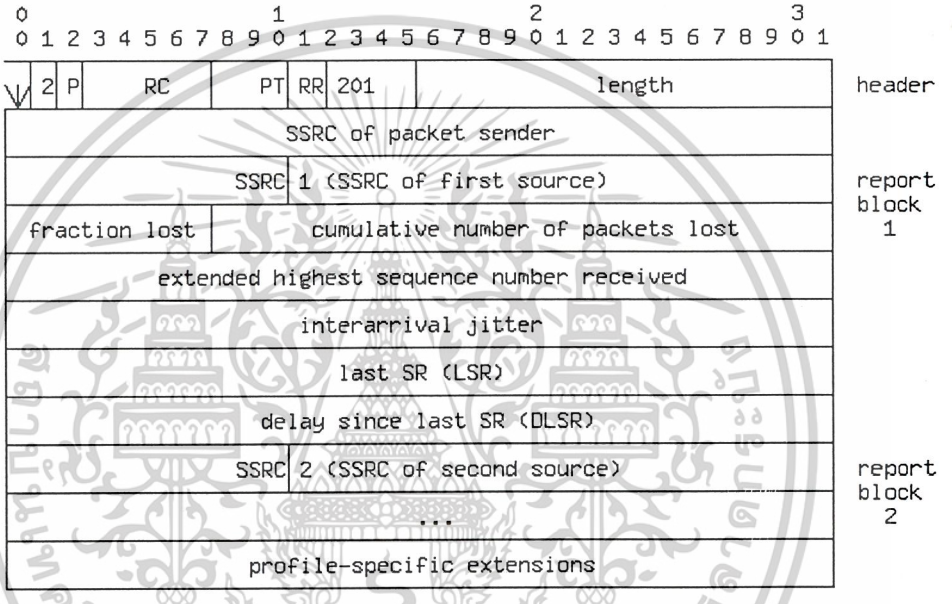
RR:(Receiver report) รายงานผู้รับ ในเรื่องสถานะภาพการรับจากผู้เข้าร่วมใช้งานที่ไม่ได้ทำหน้าที่ส่งข้อมูล

SDES:(Source description items) ส่วนที่บรรยายลักษณะแหล่งจ่าย

BYE:แสดงสถานะการยกเลิกใช้งาน

Packet Type	Playload Type
SR (Sender Report)	200
RR (Receiver Peport)	201
SDES (Source Description)	202
BYE	203
APP (Application defined)	204

ตารางที่ 2-2 ระบุค่าของชนิดแพ็คเกจอาทีพีในเพลย์โหลดไทป์



รูปที่ 2-18 แสดงรูปแบบแพ็คเกจการรายงานโดยด้านรับในอาร์ทีซีพี

ยกตัวอย่างจะมีส่วนที่เพิ่มเติมมาจากอาร์ทีซีพีเช่น

Reception report count (RC) :ส่วนนี้ 5 บิต จะแสดงจำนวนของบล็อกรายงานด้านรับที่บรรจุมาในแพ็คเกจอาร์ทีพี

Fraction Lost:ส่วนนี้ 7 บิต แสดงแพ็คเกจอาร์ทีพีที่สูญหายไปก่อนที่แพ็คเกจ SR หรือ RR ถูกส่ง

Cumulative number of packets lost:ส่วนนี้ 24 บิต แสดงจำนวนเลขรวมของข้อมูลอาร์ทีพีแพ็คเกจทั้งหมดที่สูญหายจากการเริ่มส่งในเซสชั่น

Extended highest sequence number received:ส่วนนี้ 32 บิต 16 บิตนัยสำคัญน้อยจะบรรจุหมายเลขลำดับที่ได้ทำการรับมาในข้อมูลอาร์ทีพีแพ็คเกจและ16 บิตที่มีนัยสำคัญมากกว่าเป็นส่วนขยายที่แสดงหมายเลขลำดับที่มีการโต้ตอบกลับไป

Inter-arrival jitter:ส่วนนี้ 32 บิตจะแสดงการประมาณค่าความแตกต่างของเวลาในแพ็คเกจอาร์ทีพีที่มาถึงซึ่งวัดโดยไทม์แสตมป์ยูนิต

Last SR timestamp:ส่วนนี้ 32 บิต จะบรรจุ 32 บิตกลางของ 64 บิตที่เอ็นทีพีไทม์แสตมป์ของ SR แพ็คเกจที่ถูกรับมาล่าสุด

2.9 การใช้จาวามีเดียเฟรมเวิร์ก

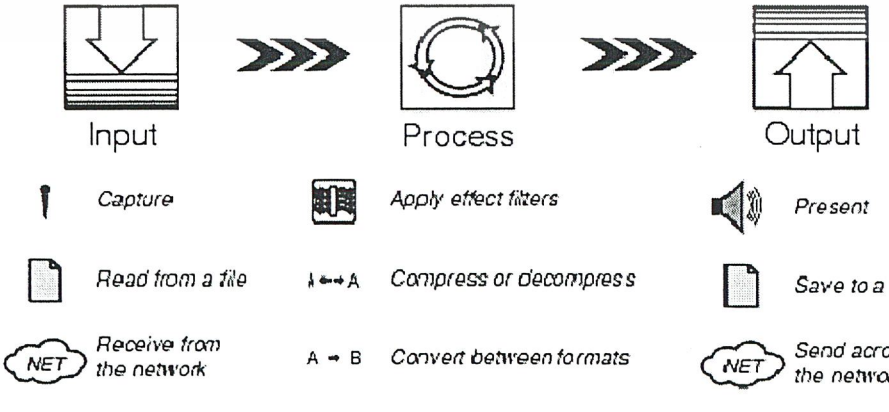
จาวามีเดียเฟรมเวิร์ก(เจเอ็มเอฟ)(Java Media Framework) ถูกออกแบบมาเพื่อใช้ในการติดต่อโดยประยุกต์ใช้โปรแกรมสำหรับสื่อข้อมูลที่ขึ้นอยู่กับเวลา(time-based media)ในงานประยุกต์ของจาวา(Java applications) และแอพเพรต(applets) ในส่วนนี้จะอธิบายการใช้เจเอ็มเอฟจัดการกับสื่อข้อมูลที่ขึ้นอยู่กับเวลา การทำงานประยุกต์ของมัน ชนิดของสื่อข้อมูลที่สนับสนุน อธิบายเกี่ยวกับกระบวนการการจัดการของเจเอ็มเอฟ โดยสามารถนำความรู้จากในส่วนนี้ไปพัฒนางานประยุกต์ต่างๆ ที่เกี่ยวกับสื่อที่ขึ้นอยู่กับเวลา เช่น การส่งเสียงผ่านอินเตอร์เน็ต,การส่งวีดีโอทางอินเตอร์เน็ต,การประชุมผ่านอินเตอร์เน็ต,ตัวแสดงผลไฟล์วีดีโอ,ตัวแสดงผลไฟล์เสียง และอื่นๆ อีกมากมาย เจเอ็มเอฟ 1.0 เอพีไอ นั้นออกแบบมาให้โปรแกรมเมอร์พัฒนาการโปรแกรมภาษาจาวาที่นำเสนอสื่อที่ขึ้นอยู่กับเวลา เจเอ็มเอฟ 2.0 เอพีไอ เพิ่มหน้าที่การใช้งานของเฟรมเวิร์กให้สามารถรองรับ จับและเก็บข้อมูลสื่อได้ สามารถควบคุมประเภทของกระบวนการและเล่นย้อนกลับได้ นอกจากนี้แล้ว เจเอ็มเอฟ 2.0 ยังเพิ่มความสามารถที่ช่วยให้ผู้พัฒนาโปรแกรมสามารถนำไปใช้และเพิ่มหน้าที่การใช้งานเจเอ็มเอฟได้อย่างง่ายดาย เจเอ็มเอฟ 2.0 สนับสนุนการจับสื่อและการใส่ที่อยู่ซึ่งจำเป็นสำหรับการพัฒนาโปรแกรมประยุกต์ที่ใช้ในการควบคุมกระบวนการส่งสื่อ สรุปแล้วเจเอ็มเอฟ 2.0 ออกแบบมาเพื่อ

- ง่ายต่อการโปรแกรม
- สนับสนุนการจับข้อมูลสื่อ
- สนับสนุนการพัฒนาการส่งสื่อและงานประยุกต์เกี่ยวกับการประชุมผ่านอินเตอร์เน็ตในจาวา

2.9.1 การจัดการกับสื่อข้อมูลที่ขึ้นอยู่กับเวลา

ข้อมูลที่ขึ้นอยู่กับเวลายกตัวอย่างเช่น ไฟล์ของเสียงเล็ก ๆ , ไฟล์ของภาพยนตร์เล็ก ๆ และภาพเคลื่อนไหว ซึ่งข้อมูลเหล่านี้สามารถนำมาจากแหล่งหลาย ๆ แหล่ง เช่น ไฟล์ที่นำมาจากเน็ตเวิร์ก หรือเครือข่ายท้องถิ่น , กล้อง , ไมโครโฟน และ การถ่ายทอดสด

ในหัวข้อนี้จะอธิบายถึงลักษณะโดยสำคัญของสื่อที่ขึ้นอยู่กับเวลาและบรรยายการใช้สื่อที่ขึ้นอยู่กับเวลาในรูปแบบของโมเดลการวิเคราะห์ข้อมูลพื้นฐาน



รูปที่ 2-19 แบบจำลองกระบวนการของสื่อ

2.9.1.1 สายของสื่อ (Media Stream)

ลักษณะที่สำคัญของสื่อข้อมูลที่ขึ้นอยู่กับเวลาคือ การส่งและการวิเคราะห์ที่มีเวลาเข้ามาเกี่ยวข้อง โดยเริ่มต้นจากการไหลของข้อมูลที่จำกัดเวลา ทั้งในรูปของการรับและการแสดงข้อมูล เนื่องด้วยเหตุผลนี้ สื่อที่ขึ้นอยู่กับเวลาจึงมักเรียกว่า สายของสื่อ มันถูกส่งในสายที่คงที่ที่ต้องรับและดำเนินการภายในช่วงเวลาที่เฉพาะเจาะจง เพื่อให้ได้ผลที่ยอมรับได้ ยกตัวอย่างเช่น เมื่อหนังกำลังเล่นถ้าหากข้อมูลสื่อไม่สามารถส่งได้เร็วเพียงพออาจจะมีการหยุดและช้าในการเล่น ในทางกลับกันถ้าข้อมูลไม่สามารถรับและดำเนินการได้เร็วเพียงพอหนังอาจจะมีการข้ามหรือข้อมูลอาจจะสูญเสียในการพยายามที่จะรักษาความเร็วในการแสดงผล

สายของสื่ออาจนำมาจากในไฟล์ในดิสไดร์ฟ หรือเอามาโดยผ่านเน็ตเวิร์ก หรือจับมาจากกล้องหรือไมโครโฟน สายของข้อมูลมักจะแบ่งออกเป็นหลายช่องสัญญาณของข้อมูลหรือเรียกว่า แทร็ก ยกตัวอย่างเช่น ไฟล์ ๆ หนึ่งอาจจะประกอบไปด้วยแทร็กของออดิโอและแทร็กของวีดีโอ สายของสื่อสามารถประกอบไปด้วยแทร็กหลาย ๆ แทร็ก ซึ่งเรียกว่ามัลติเพล็กซ์ หรือ คอมเพล็กซ์มีเดียสตรีม ดีมัลติเพล็กซ์ เป็นกระบวนวิธีการในการแยกแทร็กแต่ละอันจากคอมเพล็กซ์มีเดียสตรีม

ประเภทของแทร็กจะถูกระบุโดยชนิดของข้อมูลที่บรรจุอยู่ เช่น ออดิโอ หรือ วีดีโอ รูปแบบของแทร็กจะให้ความหมายว่า ข้อมูลสำหรับแทร็กถูกประกอบไปด้วยอะไร

สายของสื่อสามารถทำการระบุโดยที่ตั้งและโปรโตคอลที่ใช้ในการเข้าถึงมัน ยกตัวอย่างเช่น ยูอาแอล (URL) ยูอาแอลอาจถูกใช้เพื่อบรรยายถึงที่ตั้งของไฟล์ที่อยู่บนระบบท้องถิ่นหรือที่ไกลออกไป ถ้าไฟล์อยู่ในท้องถิ่นมันสามารถถูกเข้าถึงโดย ไฟล์โปรโตคอล (FILE protocol) ในทางกลับกันถ้าเกิดมันอยู่ในเวปเซอร์ เวอร์ (Web server) ไฟล์สามารถถูกเข้าถึงโดยเอสทีทีพี โปรโตคอล (HTTP protocol) และมีเดียโลเคเตอร์ (media locator) ใช้สำหรับระบุถึงที่ตั้งสายของสื่อเมื่อยูอาแอลไม่สามารถใช้ได้ สายของสื่อสามารถแบ่งกลุ่มได้โดยขึ้นอยู่กับว่าข้อมูลนั้นถูกส่งอย่างไร

- ดึง (Pull) : การส่งข้อมูลถูกเริ่มต้นและควบคุมโดยฝั่งของไคล์เอนต์ยกตัวอย่างเช่นไฮเปอร์เท็กซ์ ทรานสเฟอร์ โปรโตคอล (Hypertext Transfer Protocol) และ ไฟล์เป็น พูลโปรโตคอล (Pull protocol)
- ผลัก (Push) : ฝั่งของเซอร์เวอร์ เริ่มต้นส่งข้อมูลและควบคุมการไหลของข้อมูล ยกตัวอย่างเช่น เรียลไทม์ทรานสปอร์ตโปรโตคอล อาร์ทีพี (Real-time Transport Protocol : RTP) เป็นพุชโปรโตคอล (Push protocol) ใช้สำหรับสายสื่อคล้ายคลึงกับ เอสจีไอมีเดียเบสโปรโตคอล (SGI MediaBase protocol) เป็นพุชโปรโตคอลใช้สำหรับความต้องการของวีดีโอ (video-on-demand) (VOD)

2.9.1.2 รูปแบบของสื่อ (Media Format)

ตารางต่อไปนี้จะระบุถึงคุณลักษณะบางประการของรูปแบบของสื่อโดยทั่ว ๆ ไป เมื่อเลือกรูปแบบมันเป็นสิ่งจำเป็นที่จะต้องใส่คุณลักษณะของรูปแบบ สิ่งแวดล้อมเป้าหมาย ยกตัวอย่างเช่น ถ้าส่งสื่อผ่านเวป ต้องให้ความสนใจเกี่ยวกับแบนวิดท์

ในคอลัมน์ของความต้องการของซีพียู ซึ่งจำเป็นสำหรับกระบวนการในการนำเสนอรูปแบบที่ต้องการ ในคอลัมน์ความต้องการของแบนวิดท์ซึ่งจำเป็นสำหรับการส่งข้อมูลให้เร็วตามความต้องการ

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

รูปตารางที่ 2-3 เปรียบเทียบรูปแบบของวีดีโอในการใช้ทรัพยากร

แต่ละรูปแบบถูกออกแบบมาเพื่องานเฉพาะแต่ละอย่าง รูปแบบที่คุณภาพสูงและแบนวิดท์สูงโดยทั่วไปแล้วถูกออกแบบมาสำหรับงานประยุกต์ของซีดีรอมและที่จัดเก็บท้องถิ่น H.261 และ H.263 ถูกออกแบบมาเพื่องานประยุกต์การประชุมผ่านวีดีโอ (video conferencing) ซึ่งไม่มีการเคลื่อนไหวมากนัก ในทำนองเดียวกัน G.723 ใช้สำหรับเสียงพูดที่มีอัตราบิตเรตต่ำ สำหรับงานประยุกต์ทางด้านโทรศัพท์

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

รูปตาราง 2-4 เปรียบเทียบรูปแบบของออดิโอในการใช้ทรัพยากร

แต่ละรูปแบบถูกออกแบบมาเพื่องานเฉพาะแต่ละอย่าง รูปแบบที่คุณภาพสูงและแบนวิดท์สูง โดยทั่วไปแล้วถูกออกแบบมาสำหรับงานประยุกต์ของซีดีรอมและที่จัดเก็บท้องถิ่น H.261 และ H.263 ถูกออกแบบมาเพื่องานประยุกต์การประชุมผ่านวีดีโอ (video conferencing) ซึ่งไม่มีการเคลื่อนไหวมากนัก ในทำนองเดียวกัน G.723 ใช้สำหรับเสียงพูดที่มีอัตราบิตเรตต่ำ สำหรับงานประยุกต์ทางด้านโทรศัพท์

2.9.1.3 การนำเสนอสื่อ (Media Presentation)

สื่อที่ขึ้นอยู่กับเวลาส่วนใหญ่คือข้อมูลออดิโอและวีดีโอถูกแสดงผลโดยอุปกรณ์เช่นลำโพงและหน้าจอ อุปกรณ์เหล่านี้เป็นจุดปลายทางโดยทั่วไป (destination) สำหรับเอาท์พุตของข้อมูลสื่อ ยกตัวอย่าง เช่น เก็บไว้ในไฟล์หรือส่งผ่านข้ามเน็ตเวิร์ก จุดหมายปลายทางสำหรับเอาท์พุตของข้อมูลสื่อบางทีถูกเรียกว่า ดาต้าซิงค์ (data sink)

การควบคุมการนำเสนอ

ในขณะที่สายของสื่อถูกนำเสนอการควบคุมการนำเสนอในรูปแบบวีซีอาร์ (VCR - style) ใช้เพื่อควบคุมให้ผู้ใช้สามารถเล่นกลับ ยกตัวอย่างเช่น คอนโทรลพาเนล (control parnel) สำหรับเครื่องเล่นหนังมักจะมีปุ่มกดสำหรับหยุด เริ่มต้น กรอโดยเร็ว และ ย้อนกลับ

ลาเท็นซี (Latency)

ในหลาย ๆ กรณี โดยเฉพาะเมื่อกำลังนำเสนอข้อมูลของสายสื่อโดยผ่านเน็ตเวิร์ก การนำเสนอสายสื่อไม่สามารถทำได้โดยทันทีทันใด เวลาก่อนนำเสนอจะถูกเรียกว่าสตาร์ทลาเท็นซี (start latency) ผู้ใช้อาจจะเคยมีประสบการณ์เกี่ยวกับการล่าช้าเมื่อกดปุ่มเริ่มต้นและจะใช้เวลาชั่วครู่หนึ่งก่อนจะมีการเล่นเกิดขึ้น

การนำเสนอสื่อผสม (Multimedia) มักจะรวมเอาสื่อที่ขึ้นอยู่กับเวลาหลาย ๆ ประเภทมานำเสนอในรูปแบบของการซิงโครไนซ์ (synchronized presentation) ยกตัวอย่างเช่น ดนตรีที่เป็นแบล็กกราวด์อาจจะเล่นระหว่างรูปภาพที่นำมาสับเปลี่ยน (slide-show) หรือ ภาพเคลื่อนไหวของข้อความอาจจะถูกซิงโครไนซ์กับคลิปของออดิโอหรือวีดีโอ เมื่อการนำเสนอของสายของสื่อผสมถูกซิงโครไนซ์กันมันจำเป็นที่จะต้องมีการกำหนดสตาร์ทลาเท็นซีของแต่ละสาย มิฉะนั้นแล้วการเล่นของแต่ละสายอาจจะเริ่มต้นที่เวลาที่แตกต่างกัน

คุณภาพของการนำเสนอของสายสื่อขึ้นอยู่กับปัจจัยหลาย ๆ ประการ ดังนี้

- การบีบอัดที่ใช้
- ความสามารถของกระบวนการการเล่นของระบบ
- แบนวิดท์ที่หาได้ (สำหรับสายของสื่อที่ต้องส่งผ่านเนตเวิร์ก)

ยิ่งต้องการคุณภาพสูงขนาดของไฟล์ที่ใหญ่และความสามารถในการโปรเซส (process) ที่สูงและแบนวิดท์ก็เป็นที่ต้องการด้วย แบนวิดท์มักจะถูกนำเสนอโดยปริมาณของบิตที่ส่งในช่วงเวลาหนึ่งหรือเรียกว่าบิตเรต (bit rate)

คุณภาพของการนำเสนอวีดีโอที่สูงปริมาณของเฟรมต่อหนึ่งหน่วยเวลา (frame rate)จะต้องสูงด้วยเช่นเดียวกัน โดยปกติแล้วหนังจะใช้อัตรา 30 เฟรมต่อหนึ่งวินาที ซึ่งไม่แตกต่างกับการกระจายเสียงของทีวีหรือวีดีโอเทป

2.9.1.4 การประมวลผลสื่อ (Media Processing)

สายของสื่อจะถูกจัดการก่อนที่จะนำเสนอต่อผู้ใช้ โดยทั่วไปแล้วจะมีกระบวนการในการโปรเซสก่อนที่จะนำเสนอ

1. ถ้าสายของสื่อถูกมัลติเพล็กแต่ละแทร็กจะถูกแยกออกมา
2. ถ้าแต่ละแทร็กถูกบีบอัดจะถูกคลายออกมา
3. ถ้าจำเป็นแทร็กจะถูกเปลี่ยนไปเป็นอีกรูปแบบ (format)
4. เอฟเฟ็กท์ ฟิลเตอร์ (effect filter) จะถูกนำมาใช้สำหรับแทร็กที่ถอดรหัส (ถ้าต้องการ)

ต่อจากนั้นแทร็กจะถูกส่งไปยังอุปกรณ์เอาท์พุตที่ถูกต้อง ถ้าสายของมีเดียถูกเก็บไว้แทนที่จะเอาไปแสดงผลยังอุปกรณ์เอาท์พุต ขั้นตอนการโปรเซสจะแตกต่างกันเล็กน้อย ยกตัวอย่างเช่น ถ้าคุณต้องการจับเสียงและวีดีโอจากกล้องวีดีโอ , โปรเซสข้อมูล , บันทึกลงในไฟล์

1. แทร็กของออดิโอและวีดีโอถูกจับ
2. เอฟเฟ็กฟิลเตอร์จะถูกนำมาใช้กับแทร็กดิบ (ถ้าต้องการ)
3. แต่ละแทร็กจะถูกเข้ารหัส
4. แทร็กจะถูกบีบอัดและถูกมัลติเพล็กเป็นสายสื่อเดียว
5. สายของสายสื่อที่ถูกมัลติเพล็กจะถูกบันทึกลงไปในไฟล์

ดีมัลติเพล็กซ์เซอร์และมัลติเพล็กซ์เซอร์ (Demultiplexers & Multiplexers)

กระบวนการในการดีมัลติเพล็กซ์เซอร์จะคายสายของข้อมูลแต่ละแทร็กจากสายของข้อมูลที่ถูกมัลติเพล็กซ์ กระบวนการในการมัลติเพล็กซ์เซอร์จะทำหน้าที่ที่ตรงกันข้ามกันคือการนำเอาสายของมีเดียแต่ละแทร็กมารวมกันเป็นสายของสื่อที่ถูกมัลติเพล็กซ์

โคเด็กซ์ (Codecs)

กระบวนการในการโคเด็คซ์ คือ การบีบอัด (compression) และการคลายการบีบอัด (decompression) เมื่อแทร็กถูกเข้ารหัสแล้วสายของสื่อจะถูกเปลี่ยนไปเป็นรูปแบบ(Format)ที่เหมาะสมสำหรับการเก็บหรือการส่ง เมื่อมันถูกถอดรหัสมันจะถูกเปลี่ยนไปเป็นรูปแบบที่ไม่ผ่านการบีบอัด (non-compressed) ซึ่งเหมาะสำหรับการนำเสนอ

แต่ละโคเด็คซ์จะมีรูปแบบของอินพุตที่แน่นอนที่สามารถจัดการและรูปแบบของเอาท์พุตที่แน่นอนที่สามารถสร้างขึ้น ในบางสถานการณ์ ชุดของโคเด็คซ์อาจใช้สำหรับเปลี่ยนจากรูปแบบหนึ่งไปยังอีกรูปแบบหนึ่ง

เรนเดอร์เลอร์ (Renderers)

เรนเดอร์เลอร์เป็นอุปกรณ์ที่ใช้ในการนำเสนอ สำหรับออดิโอโดยทั่วไปแล้วอุปกรณ์ที่ใช้นำเสนอคือการ์ดเสียงที่จะส่งเสียงไปยังลำโพง สำหรับวีดีโอโดยทั่วไปแล้วอุปกรณ์ที่ใช้นำเสนอคือหน้าจอคอมพิวเตอร์

คอมโพสิทิง (Compositing)

อุปกรณ์พิเศษที่สนับสนุนการรวมของสื่อที่ขึ้นอยู่กับเวลาเป็นกระบวนการรวมแทร็กหลายๆแทร็กเข้าด้วยกันเข้าไปยังตัวกลางที่ใช้นำเสนอ ยกตัวอย่างเช่น ข้อความที่ที่อยู่บนภาพวีดีโอที่ถูกนำเสนอเป็นหนึ่งในรูปแบบการรวมกัน การรวมกันสามารถทำได้ทั้งฮาร์ดแวร์และซอฟท์แวร์ อุปกรณ์ที่ทำหน้าที่รวมกันเรียกว่าเรนเดอร์เลอร์ที่สามารถรับแทร็กหลายๆแทร็กของข้อมูล

2.9.1.5 การจับสื่อ (Media Capture)

สื่อที่ขึ้นอยู่กับเวลาสามารถถูกจับขณะที่กำลังโพรเสสและกำลังเล่นอยู่ ยกตัวอย่างเช่น ออดิโดสามารถถูกจับโดยไมโครโฟนหรือการ์ดวีดีโดสามารถใช้เพื่อจับวีดีโอจากกล้อง การจับสามารถคิดได้ว่าเป็นอินพุตของแบบจำลองกระบวนการโพรเสสสื่อ

อุปกรณ์ที่ใช้สำหรับจับ

เพื่อที่จะจับสื่อที่ขึ้นอยู่กับเวลาจำเป็นที่จะต้องมีฮาร์ดแวร์ ยกตัวอย่างเช่น การจับเสียงจากแหล่งกำเนิดเสียง จำเป็นจะต้องมีไมโครโฟน และการ์ดเสียงที่ถูกต้อง โดยคล้ายคลึงกัน การจับสัญญาณทีวีที่กำลังออกอากาศอยู่ ก็จำเป็นที่จะต้องมี ทีวีจูนเนอร์ และการ์ดวีดีโอ โดยระบบส่วนใหญ่จะมีกลไกในการตรวจสอบว่ามีอุปกรณ์อะไรอยู่บ้าง

อุปกรณ์จับสามารถแสดงคุณสมบัติเป็นได้ทั้งแหล่งพูลหรือพุช(push or pull sources) ยกตัวอย่างเช่น กล้องวีดีโอเป็นแหล่งพูล ผู้ใช้เป็นผู้ควบคุมเมื่อกำลังจับภาพ ไมโครโฟนเป็นอุปกรณ์พุชแหล่งกำเนิดเสียงให้สายของเสียงอย่างต่อเนื่อง

รูปแบบของการจับสายของสื่อขึ้นอยู่กับกระบวนการโพรเสสของอุปกรณ์ที่ใช้จับ บางอุปกรณ์มีกระบวนการที่น้อยและส่งข้อมูลดิบออกไป บางอุปกรณ์จะส่งข้อมูลที่อยู่ในรูปแบบบีบอัดแล้ว

การควบคุมการจับ

บางครั้งการควบคุมอาจอนุญาตให้ผู้ใช้จัดการกระบวนการจับ ยกตัวอย่างเช่น มีแถบแสดงการควบคุมการจับที่อนุญาตให้อยู่ใช้กำหนดบิตเรทและวิธีในการเข้ารหัสและการเริ่มต้นและหยุดในกระบวนการจับ

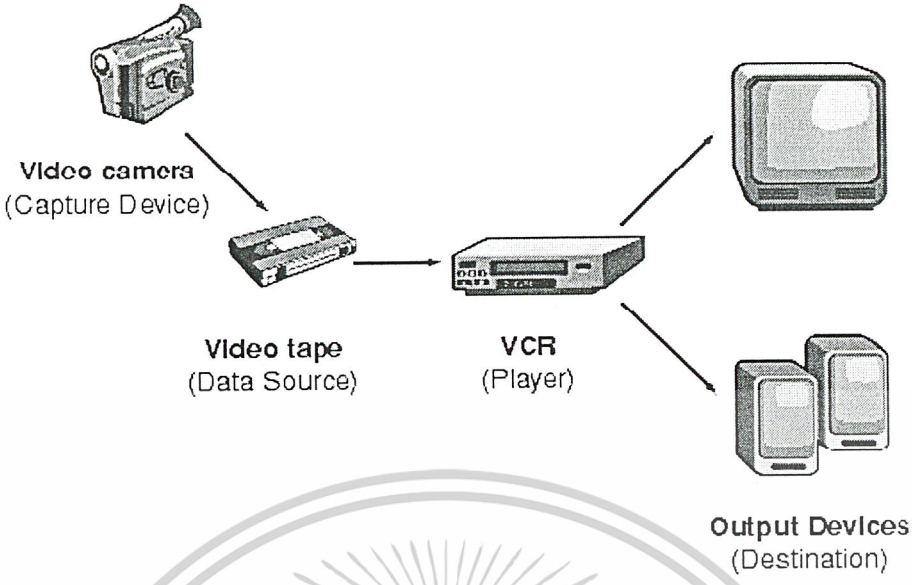
2.9.2 การทำงานของเจเอ็มเอฟ

จาวามีเดียเฟรมเวิร์ก(เจเอ็มเอฟ) มีสถาปัตยกรรมที่รวมวิธีหลายๆอย่างในการจัดการกับสิ่งรับมากระบวนวิธีในการโพรเสสและส่งสื่อที่ขึ้นอยู่กับเวลา เจเอ็มเอฟสนับสนุนประเภทของสื่อหลายๆประเภท เช่น AIFF,AU,AVI,GSM,MIDI,MPEG,QuickTime,RMF และ WAV

เจเอ็มเอฟยังรักษาคุณสมบัติของจาวาที่ว่า เขียนครั้งเดียวเอาไปรันที่ไหนก็ได้ เพื่อเป็นประโยชน์กับนักพัฒนาโปรแกรมที่ต้องการพัฒนาจาวาโปรแกรมเกี่ยวกับเสียงและวีดีโอ

สถาปัตยกรรมในระดับสูง(High-Level Architecture)

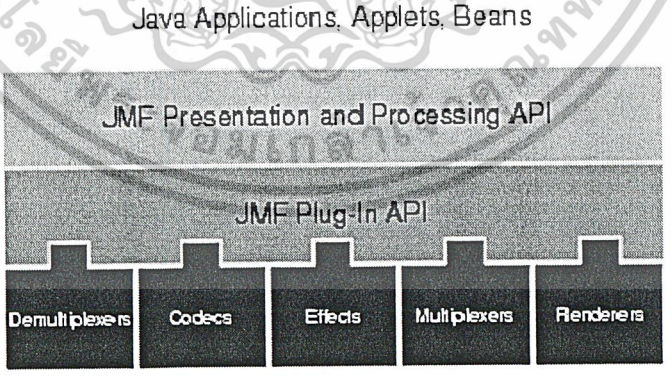
อุปกรณ์เช่นเครื่องเล่นวีซีอาร์มีรูปแบบการทำงานสำหรับบันทึก โพรเสส และนำเสนอสื่อที่ขึ้นอยู่กับเวลา เมื่อคุณต้องการเปิดหนังดูโดยใช้เครื่องเล่นวีซีอาร์ คุณนำสายของสื่อไปยังเครื่องเล่นวีซีอาร์โดยการใส่วีดีโอเทปเข้าไป เมื่อเครื่องเล่นวีซีอาร์อ่านและจะแปลงข้อมูลบนเทปและส่งสัญญาณที่ถูกต้องไปยังโทรทัศน์และลำโพง.



รูปที่ 2-20 รูปการบันทึก , กระบวนการและการนำเสนอสื่อที่ขึ้นอยู่กับเวลา

เจเอ็มเอฟก็ใช้รูปแบบพื้นฐานที่เหมือนกัน ดาต้าซอร์ส (data source) ห่อหุ้มสายของสื่อคล้าย ๆ กับวีดีโอเทปและเพลเยอร์ (player) ควบคุมกลไกคล้าย ๆ กับเครื่องเล่นวีซีอาร์ การจับออดิโอและวีดีโอกับเจเอ็มเอฟต้องการอุปกรณ์อินพุตและเอาท์พุตที่เหมาะสมเช่น ไมโครโฟน กล้องวีดีโอ และหน้าจอ

ดาต้าซอร์สและเพลเยอร์เป็นการรวมเอาส่วนของเจเอ็มเอฟระดับสูงสำหรับการจัดการเกี่ยวกับการจับ การนำเสนอ และโพรเสสสื่อที่ขึ้นอยู่กับเวลา

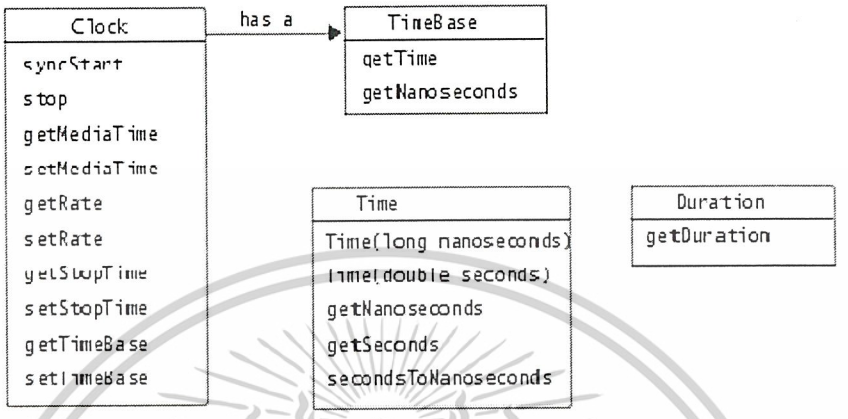


รูปที่ 2-21 สถาปัตยกรรมเจเอ็มเอฟระดับสูง

2.9.2.1 แบบจำลองทางเวลา

เจเอ็มเอฟให้ความเที่ยงตรงทางเวลาอยู่ในระดับนาโนวินาที ส่วนที่เกี่ยวข้องกับเวลามักถูกนำเสนอโดยออบเจ็ค Time บางคลาสยังสนับสนุนการระบุเวลาในหน่วยนาโนวินาที

คลาสซึ่งสนับสนุนแบบจำลองทางเวลาของเจเอ็มเอฟอิมพลีเมนต์มาจากอินเตอร์เฟส Clock เพื่อรักษาเวลาของแทร็คสำหรับสายของสื่อที่เฉพาะเจาะจง อินเตอร์เฟส Clock นิยามพื้นฐานของเวลาและการจัดการการซิงโครไนซ์ที่จำเป็นสำหรับการควบคุมการนำเสนอข้อมูลสื่อ



รูปที่ 2-22 แบบจำลองทางเวลาของเจเอ็มเอฟ

อินเตอร์เฟส Clock ใช้อินเตอร์เฟส TimeBase เพื่อรักษาเวลาของแทร็คในขณะที่สายของสื่อถูกนำเสนอ อินเตอร์เฟส TimeBase ให้เวลาที่คงที่คล้าย ๆ กับคริสตอลออสซิเลเตอร์ในนาฬิกา ข้อมูลเพียงอย่างเดียวที่อินเตอร์เฟส TimeBase ให้คือเวลาในปัจจุบันของมันซึ่งเรียกว่า เวลาTimeBase เวลา TimeBase นั้นไม่สามารถหยุดหรือตั้งค่าใหม่ได้ เวลาTimeBase ขึ้นอยู่กับนาฬิกาของระบบ

ออบเจ็คClock ของเวลาสื่อแสดงถึงตำแหน่งปัจจุบันภายในสายสื่อ จุดเริ่มต้นของสายคือเวลาของสื่อเท่ากับศูนย์ จุดสิ้นสุดของสายคือเวลาสื่อมากที่สุดของสาย duration ของสายสื่อคือเวลาตั้งแต่เริ่มต้นจนถึงสิ้นสุด หรือคือความยาวของเวลาที่ใช้ในการนำเสนอสายสื่อ

เพื่อที่จะรักษาแทร็คของเวลาสื่อปัจจุบันอินเตอร์เฟส Clock ใช้สิ่งต่อไปนี้

- The time-base start-time คือ เวลาที่ TimeBase รายงานว่าการนำเสนอเริ่มต้น
- The media start-time คือ ตำแหน่งในสายสื่อซึ่งการนำเสนอเริ่มต้น
- The playback rate บอกว่า Clock วิ่งเร็วแค่ไหนเมื่อเทียบกับ TimeBase rate เป็นสเกลซึ่งนำมาใช้กับ TimeBase ยกตัวอย่างเช่น rate 1.0 หมายถึง อัตราการเล่นที่ปกติสำหรับสายของสื่อในขณะที่ rate 2.0 หมายถึงการนำเสนอที่เร็วกว่า 2 เท่าของอัตราปกติ rateติดลบหมายถึง Clock กำลังวิ่งในทิศทางตรงข้ามกับ TimeBase ของมัน ยกตัวอย่างเช่น rate ติดลบใช้สำหรับใช้กรอกลับสายของสื่อ

เมื่อการนำเสนอเริ่มต้นเวลาของสื่อจะถูกแม็ปกับเวลาtime-base และประโยชน์ของเวลาtime-base ใช้สำหรับวัดระยะของเวลา ระหว่างการนำเสนอเวลาสื่อปัจจุบันสามารถคำนวณได้จากสูตร

MediaTime = MediaStartTime + Rate (TimeBaseTime – TimeBaseStartTime)

เมื่อการนำเสนอหยุด เวลาสื่อหยุดแต่เวลา time-base ยังคงเดินต่อไปเพื่อใช้ประโยชน์ ถ้าการนำเสนอเริ่มต้นใหม่เวลาของสื่อจะแมปอีกครั้งกับเวลา time-base ปัจจุบัน

2.9.2.2 การจัดการ (Managers)

เจเอ็มเอฟ เอพีไอ ประกอบไปด้วยอินเตอร์เฟสที่สำคัญ ๆ ซึ่งนิยามพฤติกรรมและการติดต่อระหว่างออบเจ็คซึ่งใช้ในการจับโพรเสสและนำเสนอสื่อที่ขึ้นอยู่กับเวลา การอิมพลีเมนต์อินเตอร์เฟสเหล่านี้ดำเนินการภายในโครงสร้างของเฟรมเวิร์กโดยใช้ออบเจ็คที่เรียกว่า managers

เจเอ็มเอฟใช้ managers สื่อย่างดังต่อไปนี้

- Manager จัดการโครงสร้างของ Players , Processors ,Datasources และDatasinks
- PackageManager จัดการเกี่ยวกับการลงทะเบียนของ แพคเก็จที่บรรจุในคลาสของเจเอ็มเอฟ เช่น Players, Processors, Datasources และ Datasinks
- CaptureDeviceManager จัดการเกี่ยวกับการลงทะเบียนของอุปกรณ์จับที่มีอยู่
- PlugInManager จัดการเกี่ยวกับการลงทะเบียนของปลั๊กอินของเจเอ็มเอฟ ส่วนประกอบการโพรเสส เช่น มัลติเพล็กเซอร์ ดีมัลติเพล็กเซอร์ โคเด็ค เอฟเฟ็ก และ เรนเดอร์เรอร์

การที่จะเขียนโปรแกรมโดยใช้เจเอ็มเอฟ จำเป็นที่จะต้องใช้คลาส Manager สร้างเมทธ็อดเพื่อสร้าง Players , Processoes Datasources และ Datasinks สำหรับงานประยุกต์ ถ้าจับข้อมูลสื่อจากอุปกรณ์อินพุตจะต้องใช้คลาส CaptureDeviceManager เพื่อที่จะหาว่ามีอุปกรณ์อะไรอยู่มั่งและเข้าถึงข้อมูลเกี่ยวกับมัน และถ้าต้องการที่จะควบคุมว่าจะใช้โพรเสสอะไรในการแสดงข้อมูลคุณจะต้องค้นหาโดยใช้คลาส PlugInManager เพื่อที่จะให้แน่ใจว่าปลั๊กอินนั้นได้ลงทะเบียนหรือยัง

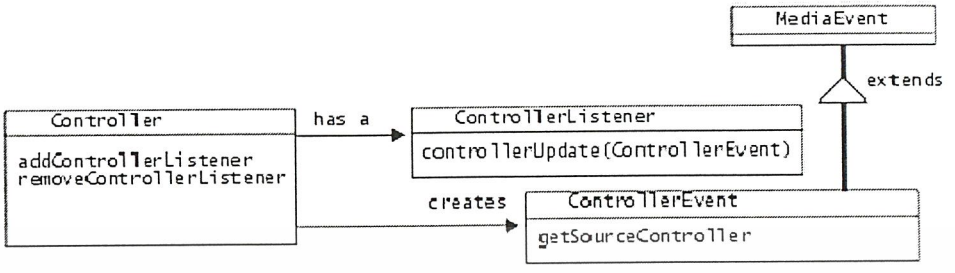
เมื่อต้องการที่จะสืบทอดหน้าที่ของเจเอ็มเอฟโดยการเพิ่มเติมปลั๊กอินใหม่สามารถที่จะลงทะเบียนกับ PlugInManager เพื่อให้มันสามารถหาได้กับ Processors ที่สนับสนุนปลั๊กอินเอพีไอเพื่อที่จะใช้ Player , Processor , Datasource , Datasink กับ เจเอ็มเอฟ คุณสามารถลงทะเบียนแพคเก็จที่ต่างออกไปกับ PackageManager

2.9.2.3 แบบจำลองเหตุการณ์ (Event Model)

เจเอ็มเอฟใช้เหตุการณ์ณ์ที่มีโครงสร้างรายงานกลไกที่ใช้ในการโปรแกรมโดยเจเอ็มเอฟเพื่อแจ้งให้ทราบว่าสภาวะปัจจุบันของระบบสื่อ (media system) และทำให้การโปรแกรมกับเจเอ็มเอฟนั้นรองรับต่อเหตุการณ์ผิดพลาดที่เกิดขึ้นเช่น ไม่มีข้อมูลและแหล่งที่อยู่ไม่มีอยู่ เมื่อใดก็ตามที่ออบเจ็คของเจเอ็มเอฟต้องการที่จะรายงานสภาวะปัจจุบันมันจะโพสไปยัง MediaEvent MediaEvent เป็นสับคลาสที่เฉพาะเจาะจงกับแต่ละประเภทของเหตุการณ์ ออบเจ็คเหล่านี้จะสร้างรูปแบบของจาวาบีน สำหรับเหตุการณ์

สำหรับแต่ละประเภทของออบเจ็คของเจเอ็มเอฟที่สามารถโพสไปยัง MediaEvents เจเอ็มเอฟได้นิยามอินเตอร์เฟส listener ที่รับผิดชอบทางด้านรับ เพื่อที่จะรับการแจ้งมาเมื่อ MediaEvents ถูกส่งต่อ สามารถอิมพลีเมนต์ อินเตอร์เฟส listener ที่ถูกต้องและลงทะเบียนคลาสของ listener กับออบเจ็คที่โพสเหตุการณ์โดยเรียกว่า เมทธ็อด addlistener

ออบเจ็ค Controller (เช่น Player และ Processors) และออบเจ็คที่ใช้ในการควบคุม เช่น GainControl จะโพสเหตุการณ์ของสื่อ (media events)



รูปที่ 2-23 แบบจำลองเหตุการณ์ของเจเอ็มเอฟ

2.9.2.4 ดาต้าซอร์ส (Data source)

ดาต้าซอร์สจะห่อหุ้มสายของมีเดียคล้าย ๆ กับซีดีเพลงในเจเอ็มเอฟ ออบเจ็ค DataSource นำเสนอสื่อออดิโอ สื่อวีดีโอ หรือรวมกันทั้งสองอย่าง DataSource สามารถเป็นไฟล์หรือสายของข้อมูลที่มาจากอินเตอร์เนต สิ่งที่ดีสำหรับคลาสนี้คือคุณสามารถระบุที่ตั้งหรือโปรโตคอล DataSource จะห่อหุ้มทั้งที่ตั้งของสื่อและโปรโตคอลและซอฟแวร์ที่ใช้ในการส่งสื่อ เมื่อ DataSource ถูกสร้างขึ้นมาแล้วจะถูกส่งนำไปให้ Player เพื่อนำมาแสดงผล ซึ่ง Player จะไม่สนใจว่า DataSource มีต้นกำเนิดมาจากที่ใด

ข้อมูลสื่อสามารถมาจากหลาย ๆ ที่ เช่น ไฟล์ที่มาจาเน็ตเวิร์กหรือเครือข่ายท้องถิ่น หรือการวัดค่าจากอินเตอร์เนต

2.9.2.5 เพลเยอร์ (Player)

Player จะเอาอินพุตของออดิโอหรือวีดีโอแล้วส่งไปยังลำโพงหรือหน้าจอเช่นเดียวกับเครื่องเล่นซีดีที่อ่านข้อมูลจากซีดีและส่งเสียงดนตรีไปยังลำโพง Playerสามารถทำงานโดยอัตโนมัติ เพราะว่า Player มีการเตรียมตัวมันเองและ DataSourceของมัน ก่อนที่จะเริ่มเล่นสื่อ เพื่อที่จะเข้าใจถึงสิ่งนี้ลองใส่ซีดีเข้าไปในเครื่องเล่นซีดีของคุณและเล่นเพลงที่สี่และดูถึงผลที่ตามมาเมื่อเครื่องเล่นซีดียังไม่ได้เล่นเพลงอย่างแรกที่มันทำคือเริ่มหาแทร็คของเพลงที่สี่ที่จะเริ่มต้นและทำบางสิ่งอื่นเพื่อเป็นการเตรียมตัว หลังจากประมาณครึ่งวินาที (ขึ้นอยู่กับเครื่องเล่นซีดี) จะเริ่มได้ยินเสียงดนตรีเช่นเดียวกัน JMF Player ก็ต้องเตรียมตัวก่อนที่จะได้ยินเสียงออดิโอหรือได้ดูวีดีโอ ในวิธีการโดยทั่วไป Player จะมีลำดับของสภาวะไปจนถึงสภาวะสุดท้าย

สภาวะของ player ในเจเอ็มเอฟ นิยามไว้ 6 สภาวะดังนี้

- Unrealized : ในสภาวะนี้ออบเจ็คของเพลเยอร์เพิ่งจะเริ่มต้นซึ่งไม่รู้สภาวะแวดล้อมของตัวเอง ไม่รู้อะไรเลยเกี่ยวกับสื่อของมัน

- Realizing : Player เปลี่ยนจากสภาวะ unrealized เป็นสภาวะ realizing เมื่อคุณเรียกเมทธ็อด realize ( ) ของ Player ในสภาวะ realizing Player จะอยู่ในกระบวนการเลือกความต้องการของแหล่งของมัน
- Realized : ซึ่งจะเปลี่ยนมาจากสภาวะ realizing ในสภาวะนี้ Player จะรู้ว่าแหล่งอะไรที่มันต้องการและมีข้อมูลเกี่ยวกับประเภทของสื่อที่จะแสดง มันยังแสดงส่วนประกอบและการควบคุมที่มองเห็นได้อีกด้วยและมีการเชื่อมต่อไปยังออบเจ็คในระบบ
- Prefetching : เมื่อเมทธ็อด prefetch() ถูกเรียก Player จะเข้าสู่สภาวะ prefetching สภาวะนี้ Player จะเตรียมตัวที่จะแสดงสื่อ Player จะโหลดข้อมูลสื่อและสิ่งอื่นที่จำเป็นในการเล่นข้อมูลสื่อ
- Prefetched : สภาวะนี้จะเริ่มต้นเมื่อสิ้นสุดกระบวนการ Prefetching มันพร้อมที่จะเล่น
- Started : สภาวะนี้จะเริ่มต้นเมื่อคุณเรียกเมทธ็อด start() Player ตอนนี้พร้อมที่จะแสดงผลข้อมูลสื่อแล้ว

2.9.2.6 Processor

Processor เป็นอีกประเภทหนึ่งของ Player ในเจเอ็มเอฟเอพีไอ อินเตอร์เฟส Processor สืบทอดมาจาก Player ดังนั้น Processoe จึงสนับสนุนการควบคุมการนำเสนอเช่นเดียวกับ Player แต่สิ่งที่ไม่เหมือนกับ Player คือ Processoe ควบคุมสิ่งที่กระบวนการถูกแสดงบนสายของสื่อ

นอกจากนี้แล้วในการส่งแหล่งของข้อมูล Processor ยังคงแสดงเอาท์พุตของข้อมูลสื่อ โดยผ่านทาง DataSource ดังนั้นมันสามารถถูกนำเสนอโดย Player หรือ Processor

นอกจากสภาวะ 6 สภาวะของ Player Processor ยังรวมสภาวะอีกสองสภาวะ ซึ่งเกิดขึ้นก่อน Processor เข้าสู่ สภาวะ realizing แต่หลังจากสภาวะ unrealized

Configuring: Processor เข้าสู่สภาวะ configuring

2.9.2.7 ดาตาซิงค์(DataSink)

DataSink เป็นอินเตอร์เฟสพื้นฐานสำหรับออปเจ็คที่ใช้ในการอ่านสื่อที่ส่งมาจาก DataSource และส่งสื่อไปยังปลายทาง

2.9.2.8 ฟอร์แมท(Format)

ออปเจ็คฟอร์แมทจะแสดงฟอร์แมทที่ถูกต้อง แก่ สื่อ ฟอร์แมท โดยตัวของมันเองแล้วจะไม่มีพารามิเตอร์เกี่ยวกับการเข้ารหัสโดยเฉพาะ หรือ ข้อมูลเกี่ยวกับเวลา มันจะบ่งบอกถึงชื่อของการเข้ารหัสและประเภทของของข้อมูลที่ฟอร์แมทต้องการ สับคลาสของฟอร์แมทจะรวม AudioFormat และ VideoFormat และ VideoFormat ประกอบด้วยสับคลาส 6 คลาส

- H261Format
- H263Format
- IndexedColorFormat
- JPEGFormat

- RGBFormat
- YUVFormat

2.9.2.9 เมเนเจอร์ (Manager)

ออปเจค manager เป็นออปเจ็คที่ใช้ในการเชื่อมต่อ อิมพิเมนต์อินเตอร์เฟสหลายๆอินเตอร์เฟสที่ใช้กับคลาสที่เกิดขึ้น ไม่มีอยู่ในโลกของความเป็นจริงในระบบสเตอริโอ แต่คุณสามารถจินตนาการ manager คล้ายกับเป็นสิ่งเชื่อมต่อวัตถุสองสิ่งที่แตกต่างกัน ยกตัวอย่างเช่น สร้าง Player จาก Datasource เจเอ็มเอฟ นำเสนอ managers 4 ประเภท

- Manager: ใช้ Manager สร้าง Players, Processors, DataSources และ DataSinks ยกตัวอย่างเช่น ถ้าต้องการที่จะส่ง DataSource สามารถใช้ Manager สำหรับการสร้าง Player สำหรับ DataSource
- PackageManager: manager นี้เป็นตัวเก็บรักษาค่าที่ลงทะเบียนของแพ็คเก็จที่บรรจุไว้ในคลาสของเจเอ็มเอฟ เช่น Players, Processors, DataSources และ DataSinks
- CaptureDeviceManager: manager นี้เก็บรักษาค่าที่ลงทะเบียนไว้ของอุปกรณ์ที่ตรวจพบ
- PlugInManager: manger ตัวนี้เก็บค่าที่ลงทะเบียนของอุปกรณ์ปลั๊กอินของเจเอ็มเอฟที่มีอยู่

การสร้าง Player

การโปรแกรมมัลติมีเดียโดยใช้เจเอ็มเอฟ งานหนึ่งที่สำคัญที่สุดคือการสร้าง Player โดยการเรียกเม็ธทอด createPlayer() Manager ใช้ URL ของสื่อหรือ MediaLocator ที่ระบุไว้ในการสร้าง Player ที่เหมาะสม เมื่อทำการสร้าง Player แล้ว สามารถสร้างส่วนประกอบที่มองเห็นได้ของ Player โดยใช้หน้าต่างหรือใช้แอพเพ็ต

เพื่อที่จะแสดงวัตถุที่มองเห็นได้ของออปเจ็ค Player จะต้อง

1. เรียกส่วนประกอบที่มองเห็นได้โดยเรียกใช้เม็ธทอด getVisualcomponet()
2. ใส่ส่วนประกอบที่มองเห็นได้โดยเรียกหน้าต่างของแอพริเคชันหรือแอบเพ็ต

Player สามารถที่รวมพาเนลที่ใช้ควบคุมกับปุ่มที่มีปุ่ม เริ่มต้น หยุดชั่วคราว และหยุด สายของสื่อเช่นเดียวกับการควบคุมเสียง เหมือนกับปุ่มที่ใช้บนเครื่องเล่นซีดี

เม็ธทอดของ Player สามารถเรียกได้ตั้งแต่ Player อยู่ในสภาวะ realized เพื่อจะให้แน่นอนว่ามันอยู่ในสภาวะนี้ โดยสามารถเรียกใช้เม็ธทอดของ Manager ที่เรียกว่า createRealizedPlayer() สร้าง Player เม็ธทอดนี้เป็นวิธีที่สะดวกที่จะสร้าง Player ภายในขั้นตอนเดียว เมื่อถูกเรียก มันจะปิดกั้นจน Player อยู่ในสภาวะ realized ต่อไปจากนี้ เม็ธทอด start() สามารถที่จะถูกเรียกหลังจาก Player ถูกสร้างแต่ก่อนที่มันจะไปถึงสภาวะ prefetched เม็ธทอด start() จะพยายามที่เปลี่ยนแปลง Player ไปยังสภาวะ started จากสภาวะที่เป็นอยู่ ยกตัวอย่างเช่น สามารถเรียก start() ทันทีหลังจากที่ Player ถูกสร้างขึ้น เม็ธทอด start() จะเรียกเม็ธทอดที่จำเป็นทั้งหมดเพื่อที่จะนำ Player ไปสู่สภาวะ started

ข้อมูลสื่อที่ถูกจับ

การจับสื่อเป็นงานสำคัญอีกอย่างในการโปรแกรมด้วยเจเอ็มเอฟ สามารถทำการจับสื่อโดยใช้อุปกรณ์จับสื่อเช่น microphone หรือ กล้องวีดีโอ มันสามารถโพรเสสและส่งต่อ หรือเก็บไว้ในฟอร์แมทสื่อ เพื่อที่จะจับข้อมูลสื่อจำเป็นที่จะต้อง

1. ติดตั้งอุปกรณ์ตรวจจับที่ต้องการจะใช้โดยการค้นหาใช้ CaptureDeviceManager
2. รับออปเจ็ค CaptureDeviceInfo สำหรับอุปกรณ์
3. รับ MediaLocator จากออปเจ็ค CaptureDeviceInfo และใช้มันเพื่อสร้าง DataSource
4. สร้าง Player หรือ Processor โดยใช้ DataSource
5. เริ่มต้น Player หรือ Processor เพื่อที่จะเริ่มต้นกระบวนการจับ

การโพรเสสมัลติมีเดียที่เป็นแบบเวลาจริง(Realtime multimedia processing)

สามารถใช้เจเอ็มเอฟในการส่งหรือรับการบอร์ดแคส เช่น วิทยุที่กระจายเสียหรือการออกอากาศของทีวี หรือ ประชุมผ่านเครื่อข่ายอินเตอร์เน็ตหรืออินทราเน็ต

คุณลักษณะของสื่อที่เป็นแบบเวลาจริงจะแตกต่างกับการเข้าถึงข้อมูลแบบคงที่ โปรโตคอลที่ใช้แบบเวลาจริงไม่ประกันว่า แพ็คเก็จทุกแพ็คเก็จจะมาถึงโดยปลอดภัย สิ่งที่สำคัญมากที่สุดคือ ทำอย่างไรจึงจะไม่เกิดการสูญเสียโดยรวมและประกันว่าไม่มีความล่าช้าในการรับ เมื่อเล่นข้อมูลสื่อโดยปราศจากการโหลดข้อมูลทั้งหมดมา การส่งผ่านอินเตอร์เน็ตในแบบเวลาจริงต้องการแบนด์วิทที่มากเพื่อที่ฝ่ายรับจะสามารถเล่นข้อมูลสื่อได้อย่างต่อเนื่องโดยใช้โปรโตคอลของมันโดยเฉพาะเพื่อส่งแพ็คเก็จของสายของสื่อแบบเวลาจริง องค์กร The Internet Engineering Task force (IETF) ได้กำหนดโปรโตคอลที่เรียกว่า Real-Time Transport Protocol (RTP) ซึ่งเป็นโปรโตคอลที่เหมาะสมสำหรับงานประยุกต์ที่ต้องการส่งข้อมูลแบบเวลาจริงเช่น audio,video ผ่านการบริการแบบมัลติคลาส (multicast) หรือแบบ unicast โปรโตรคอล อาร์ทีพีมักจะใช้บนโปรโตคอล User Datagram Protocol (UDP) ไม่มีการรับประกันว่าข้อมูลที่ส่งแบบอาร์ทีพีจะมาถึงตามลำดับที่มันถูกส่งมา ในความเป็นจริงแล้ว ไม่มีการรับประกันว่ามันจะมาถึงทั้งหมด มันขึ้นอยู่กับฝั่งรับว่าจะรวมแพ็คเก็จถูกส่งมาตามลำดับหรือไม่ และตรวจสอบโดยใช้ข้อมูลที่ถูกส่งมาในส่วนหัวของแพ็คเก็จ และมีการใช้โปรโตคอล Real-Time Transport Control Protocol (RTCP) ที่มีการใส่ที่อยู่ของต้นทาง และประกันคุณภาพของการบริการแบบเวลาจริง งานประยุกต์ที่ใช้อาร์ทีพีจะสามารถแบ่งกลุ่มได้เป็นอาร์ทีพีเซอร์เวอร์ (RTP servers) (งานประยุกต์ที่จำเป็นที่ใช้ในการส่งผ่านเน็ตเวิร์ก) อย่างไรก็ตาม บางแอพริเคชัน เช่น การประชุมทางไกล จะสร้างเซ็ดชันอาร์ทีพีขึ้นมาจับและส่งข้อมูล เช่นเดียวกันกับทางรับ

เจเอ็มเอพเอพีไอที่ถูกนิยามในแพ็คเก็จก็คือ javax.media.rtp,javax.media.rtp.event, และ javax.media.rtp.rtcp สำหรับการเล่นสายของ RTP ในการเล่นสายและการส่งของเจเอ็มเอฟ อาร์ทีพี เอพีไอ สามารถทำงานได้กับอุปกรณ์จับของ เจเอ็มเอฟ, players, processors และ ความสามารถในการโพรเสส นอกจาก managers สื่อย่างที่ได้อธิบายไป ยังมี manager อีกอย่างที่ใช้ในการจัดการกับอาร์ทีพีเซจชัน คือ SessionManager มันยังคงเก็บแทร็กของเซ็จชันที่เข้าร่วม และสายที่ถูกส่ง เช่นเดียวกับการจัดการควบคุมโดยใช้ อาร์ทีซีพี และสนับสนุนอาร์ทีซีพีสำหรับทั้ง ภาคส่งและ ภาครับ

2.10 การส่งสายของสื่อแบบอาร์ทีพี

ในการที่จะส่งสายของสื่อแบบอาร์ทีพี นั้น ต้องใช้ Processor สร้าง DataSource ซึ่งเข้ารหัสแบบอาร์ทีพี รวมทั้งสร้าง SessionManager หรือ DataSink เพื่อที่จะควบคุมการส่งอินพุตที่ป้อนให้ Processor สามารถเป็นได้ทั้งข้อมูลที่ถูกเก็บหรือจับมาจากอุปกรณ์ตรวจจับ สำหรับข้อมูลที่ถูกเก็บไว้ ต้องใช้ Media Locator เพื่อที่จะระบุไฟล์เมื่อสร้าง Processor สำหรับข้อมูลที่จับนั้น DataSource ตรวจจับนั้นใช้เป็นอินพุตกับ Processor

มีสองวิธีที่ใช้ในการส่งสายของสื่อแบบอาร์ทีพี

1.ใช้ Medialocator ที่มีพารามิเตอร์ของเซ็จชั่นแบบ เพื่อที่สร้าง DataSink แบบอาร์ทีพีโดยเรียก Manager.createDataSink

2.ใช้ Session manager เพื่อที่จะสร้างสายของสื่อที่ใช้ส่ง สำหรับcontent และ การควบคุมการส่ง ถ้า เราใช้ MediaLocator เพื่อที่จะสร้าง DataSink แบบอาร์ทีพี เราจำเป็นที่จะต้องส่งสายของสื่ออันแรกใน DataSource ถ้าเราต้องการที่จะส่งสายของสื่อแบบอาร์ทีพีหลายๆสาย หรือต้องการที่จะชี้เฉพาะเซ็จชั่น เรา จำ เป็นต้องใช้ SessionManager โดยตรง

โดยไม่คำนึงว่าเราจะเลือกส่งสายของสื่อแบบอาร์ทีพีอย่างไร เราจำเป็นจะต้อง

1. สร้าง Processor กับ DataSource ที่แสดงข้อมูลที่เราต้องการ
2. กำหนดค่าให้กับ Processor เพื่อให้ได้ข้อมูลที่เข้ารหัสแบบอาร์ทีพีได้เอาท์พุตจาก Processor ในรูปแบบ DataSource

การกำหนดค่าให้ Processor

ในการที่จะกำหนดค่าให้ Processor เพื่อที่จะผลิตข้อมูลที่เข้ารหัสแบบอาร์ทีพีนั้น เราต้องตั้งค่ารูปแบบที่เฉพาะเจาะจงเป็น อาร์ทีพีสำหรับแต่ละ track และระบุ content ของเอาท์พุตที่เราต้องการรูปแบบแทร็กถูกตั้งค่าไว้โดยรับเอา TrackControl สำหรับแต่ละแทร็กและเรียก setFormat เพื่อที่จะระบุรูปแบบที่เฉพาะแบบอาร์ทีพี รูปแบบที่เฉพาะแบบอาร์ทีพี นั้นจะถูกเลือกโดยตั้งค่าสตริงของ format ที่ต้องการที่จะระบุเช่น “AudioFormat.GSM_RTP” สายของสื่อแบบออดิโอและวีดีโอไม่ควรที่จะแทรกกัน ถ้าแทร็กของ Processor เป็นประเภทของสื่อที่ต่างกัน แต่ละสายของสื่อจะถูกส่งในแต่ละเซ็จชั่นของอาทีพี

การกู้เอาเอาท์พุตของ Processor

ครั้งแรกที่รูปแบบแทร็กของ Processor ถูกตั้งค่าและ Processor ถูกทำให้อยู่ในสภาวะRealized DataSource ของ Processor สามารถที่จะถูกกู้ เรากูเอาเอาท์พุตของ Processor เป็น DataSource โดยการเรียก getDataOutput DataSource ที่ถูกคืนค่ากลับมาสามารถเป็นได้ทั้ง PushBufferDataSource หรือ PullBufferDataSource ขึ้นอยู่กับแหล่งของข้อมูล DataSource ที่เป็นเอาท์พุตถูกต่อกับ Session Manager ใช้เม็ธท็อด createSentStream session manager ต้องถูกเตรียมพร้อม (initialized) ก่อนที่เราจะสามารถสร้างสายของสื่อที่ใช้ส่งถ้า DataSource บรรจุ SourceStream หลายๆ อย่าง แต่ละ SourceStream จะถูกส่งเป็นสายของสื่อแบบแยกจากกัน ไม่ว่าจะเป็น session เดียวกันหรือต่างเซ็จชันกัน ถ้า DataSource บรรจุทั้งสายของสื่อแบบ วีดีโอและออดิโอ ซึ่งการแยกเซ็จชั่นของอาร์ทีพีจะต้องถูกสร้างสำหรับออดิโอและวีดีโอ

เรายังสามารถสร้าง DataSorce ที่คัดแบบและส่งออกเป็นสายของสื่อแบบอาร์ทีพี ที่ต่างกัน ในเซ็จชั่นที่เหมือนหรือต่างกัน

การควบคุมการเลื่อนทางเวลาของแพ็คเก็จ

แต่ละแพ็คเก็จที่เลื่อนทางเวลา ถูกรู้จักในนามของช่วงของการแพ็คเก็จ (packetization interval) เป็นเวลาที่ถูกแสดงโดยแพ็จเก็จแบบอาร์ทีพีที่ถูกส่งข้ามเน็ตเวิร์กช่วงของการแพ็คเก็จระบุเป็นการเลื่อนของเวลาจากปลายสุดถึงปลายสุด แพ็คเก็จยิ่งมากควรที่จะลดข้อมูลที่เป็นส่วนหัว แต่จะทำให้การเลื่อนของเวลาสูงขึ้นและทำให้แพ็คเก็จสูญหาย ตัวรับควรที่จะยอมรับแพ็คเก็จที่อยู่ระหว่าง 0 และ 200 ms ของข้อมูลแบบออดิโอ(สำหรับการเข้ารหัสออดิโอแบบ frame ตัวรับควรที่จะยอมรับแพ็คเก็จที่เวลา 200 ms ถูกหารด้วยช่วงของเฟรมขึ้นไป) การจำกัดนี้อนุญาตให้ขนาดของบัฟเฟอร์ที่เหมาะสมโดยตัวรับ แต่ละการโคเด็ก(codec) มีการแตกเป็นแพ็จเก็จที่เหมาะสมสำหรับการเข้ารหัสสำหรับการส่งสตรีมของวีดีโอ เฟรมของวีดีโอแรกถูกส่งในแพ็คเก็จของอาร์ทีพี หลายๆ แพ็คเก็จ ขนาดของแต่ละแพ็คเก็จถูกจำกัดโดยหน่วยของการส่งสูงสุดของเน็ตเวิร์ก พารามิเตอร์นี้ตั้งโดยการใช้เม็ธท็อด setPacket ของ PacketSizeControl

การส่งข้อมูลอาร์ทีพีโดยใช้ DataSink

วิธีที่ง่ายที่สุดในการส่งข้อมูลแบบอาร์ทีพี คือสร้าง DataSink แบบอาร์ทีพี ขึ้นมาโดยใช้เม็ธท็อด Manager.createDataSink เราสามารถส่งเอาท์พุตของ DataSource จาก Processor และ MediaLocator ที่บรรจุข้อมูลของเซ็จชั่นของ RTP ซึ่ง DataSource จะถูกสตรีม (MediaLocator บรรจุข้อมูลแอดเดรสและพอร์ตของเซ็จชั่นอาร์ทีพี)เพื่อที่จะควบคุมการส่ง เราต้องเรียกเม็ธท็อด Start และ Stop บน DataSink สายของสื่อแรกใน DataSource จะถูกส่งไป
ในตัวอย่างที่ 1 ข้อมูลออดิโอจะถูกจับและส่งไปโดยใช้ DataSink

ตัวอย่างที่ 1 : การส่งข้อมูลแบบอาร์ทีพี โดยใช้ DataSink

```
// First find a capture device that will capture linear audio
      // data at 8bit 8Khz

      AudioFormat format= new AudioFormat(AudioFormat.LINEAR, 8000, 8, 1);

      Vector devices= CaptureDeviceManager.getDeviceList( format);

      CaptureDeviceInfo di= null;

      if (devices.size() > 0) {
          di = (CaptureDeviceInfo) devices.elementAt( 0);
      }
      else {
         // exit if we could not find the relevant capturedevice.
         System.exit(-1);
      }

      // Create a processor for this capturedevice & exit if we
      // cannot create it
      try {
            Processor p = Manager.createProcessor(di.getLocator());
```

```
    } catch (IOException e) {
        System.exit(-1);
    } catch (NoProcessorException e) {
        System.exit(-1);
    }

    // configure the processor
    processor.configure();

    // block until it has been configured

    processor.setContentDescriptor(
        new ContentDescriptor( ContentDescriptor.RAW));

    TrackControl track[] = processor.getTrackControls();

    boolean encodingOk = false;

    // Go through the tracks and try to program one of them to
    // output gsm data.

    for (int i = 0; i < track.length; i++) {
        if (!encodingOk && track[i] instanceof FormatControl) {

            if (((FormatControl)track[i]).
                setFormat( new AudioFormat(AudioFormat.GSM_RTP,
                                           8000,
                                           8,
                                           1)) == null) {

                track[i].setEnabled(false);
            }
            else {
                encodingOk = true;
            }
        } else {
            // we could not set this track to gsm, so disable it
            track[i].setEnabled(false);
        }
    }

    // At this point, we have determined where we can send out
    // gsm data or not.
    // realize the processor
    if (encodingOk) {
        processor.realize();
        // block until realized.
        // get the output datasource of the processor and exit
        // if we fail
        DataSource ds = null;

        try {
            ds = processor.getDataOutput();
        } catch (NotRealizedError e) {
            System.exit(-1);
        }
```

```
        // hand this datasource to manager for creating an RTP
        // datasink our RTP datasimnk will multicast the audio
        try {
            String url= "rtp://224.144.251.104:49150/audio/1";

            MediaLocator m = new MediaLocator(url);

            DataSink d = Manager.createDataSink(ds, m);

            d.open();
            d.start();
        } catch (Exception e) {
            System.exit(-1);
        }
    }
```

การส่งข้อมูลแบบอาร์ทีพี ด้วย Session Manager

กระบวนการพื้นฐานของการส่งข้อมูลแบบอาร์ทีพี ด้วย Session manger นั้นทำได้ดังนี้

1. สร้าง Processor และตั้งค่ารูปแบบของแต่ละแทร็กเป็นแบบอาร์ทีพี
2. กู้เอา DataSource จาก Processor
3. เรียก createSendStream ในการสร้างครั้งก่อน และเตรียมพร้อม Session Manager โดยส่ง DataSource และดัชนีของสตรีม session manager สร้าง SendStream สำหรับระบุ SourceStream
4. เริ่ม session manager โดยการเรียก SessionManager startSession
5. ควบคุมการส่ง โดยผ่านทางเม็ธท็อด SendStream SendStreamListener สามารถลงทะเบียนเพื่อฟังเหตุการณ์ณีที่เกิดขึ้นบน SendStream

การสร้าง Send Stream

ก่อนที่ session manager สามารถส่งข้อมูล จำเป็นที่จะต้องรู้ว่าที่ไหนที่จะรับข้อมูลที่จะส่ง เมื่อเราสร้าง SendStream ขึ้นมาใหม่ เราจะต้องระบุ DataSource ให้ SessionManager ที่ร้องขอข้อมูล เพราะว่า DataSource สามารถที่จะบรรจุสตรีมหลายๆสตรีมเราจำเป็นจะต้องระบุดัชนีของสตรีมเพื่อที่จะส่งในเซ็จชั่นเราสามารถสร้างสตรีมหลายๆสตรีมโดยการส่งDataSource ที่แตกต่างกันแก่เม็ธท็อด createSendStream โดยระบุดัชนีของสตรีมที่แตกต่างกัน

การใช้แหล่งของข้อมูลที่สามารถถอดแบบมาได้

การใช้ RTP หลายๆอย่างเกี่ยงข้องกับการส่งสตรีมผ่านทางเซ็จชั่นของอาร์ทีพี หลายๆ อย่าง หรือการเข้ารหัสสตรีมไปในหลายๆ รูปแบบและส่งมันไปในหลายๆเซ็จชั่นของอาร์ทีพีเมื่อสตรีมถูกเข้ารหัสในรูปแบบเดียวถูกส่งไปในหลายๆเซ็จชั่นของอาร์ทีพี แล้ว เราจำเป็นจะต้องคัดแบบเอาท์พุตของ DataSource จาก Processor จากข้อมูลที่ถูกตรวจจับ การสร้าง DataSource ที่คัดแบบได้ทำโดยผ่านทาง Manager และเรียก getClone บน DataSource ที่คัดแบบ Processor ใหม่ที่สามารถสร้างจาก DataSource ที่ถูกคัดแบบ แทร็กของมันถูกเข้ารหัสในรูปแบบที่ออกแบบและสตรีมถูกส่งออกผ่านทางเซ็จชั่นของอาร์ทีพี

การควบคุมสายของสื่อที่ส่ง

เราใช้เม็ธท็อด start และ stop ของ RTPStream เพื่อควบคุม SendStream การใช้เม็ธท็อด Start จะเป็นการเริ่มต้นการเคลื่อนย้ายของข้อมูลผ่านเน็จเวิร์ก และการใช้เม็ธท็อดstopเป็นการระบุว่าให้หยุดการส่ง ข้อมูลการเริ่มต้น หรือหยุดการส่งสายของสื่อจะรับผิดชอบการกระทำบน DataSource ของมัน อย่างไรก็ตาม ถ้า DataSource ถูกเริ่มโดยอิสระในขณะที่ SendStream ถูกหยุด ข้อมูลจะถูกหยุด หรือไม่มีการดึงโดย session manager ระหว่างนี้จะไม่มีข้อมูลถูกส่งผ่านเน็ตเวิร์ก

การส่งออดิโอที่ถูกตรวจจับในหนึ่งเซ็จชั่น

ตัวอย่างที่ 2 การจับข้อมูลออดิโอแบบโมโนและส่งมันไปในเซ็จชั่นแบบอาร์ทีพี

ตัวอย่างที่ 2 :การส่งออดิโอที่ถูกตรวจจับบนเซ็จชั่นเดียว

```
// First, we'll need a DataSource that captures live audio:

        AudioFormat format = new AudioFormat(AudioFormat.ULAW, 8000, 8, 1);

        Vector devices= CaptureDeviceManager.getDeviceList( format);

        CaptureDeviceInfo di= null;
        if (devices.size() > 0) {
            di = (CaptureDeviceInfo) devices.elementAt( 0);
        }

        Else {
            // exit if we could not find the relevant capture device.
            System.exit(-1);
        }
        // Create a processor for this capture device & exit if we
        // cannot create it
        try {
            Processor p = Manager.createProcessor(di.getLocator());
        } catch (IOException e) {
            System.exit(-1);
        } catch (NoProcessorException e) {
            System.exit(-1);
        }

        // at this point, we have succesfully created the processor.
        // Realize it and block until it is configured.

        processor.configure();

        // block until it has been configured

        processor.setContentDescriptor(
            new ContentDescriptor( ContentDescriptor.RAW));

        TrackControl track[] = processor.getTrackControls();

        boolean encodingOk = false;
```

```
        // Go through the tracks and try to program one of them to
        // output ULAW_RTP data.
        for (int i = 0; i < track.length; i++) {
            if (!encodingOk && track[i] instanceof FormatControl) {

                if (((FormatControl)track[i]).
                    setFormat( new AudioFormat(AudioFormat.ULAW_RTP, 8000,  8,  1))==
null) {

                    track[i].setEnabled(false);
                }
                else {
                    encodingOk = true;
                }
            }
            else {
                // we could not set this track to gsm, so disable it
                track[i].setEnabled(false);
            }
        }
```

```
        // Realize it and block until it is realized.
        processor.realize();

        // block until realized.
        // get the output datasource of the processor  and exit
        // if we fail

        DataSource ds = null;

        try {
            ds = processor.getDataOutput();
        } catch (NotRealizedError e){
            System.exit(-1);
        }

        // Create a SessionManager and hand over the
        // datasource for SendStream creation.

        SessionManager rtpsm = new com.sun.media.rtp.RTPSessionMgr();

        // The session manager then needs to be initialized and started:
        // rtpsm.initSession(...);
        // rtpsm.startSession(...);

        try {
            rtpsm.createSendStream(ds, 0);
        } catch (IOException e)  {
            e.printStackTrace();
        } catch( UnsupportedFormatException e) {
            e.printStackTrace();
        }
```

การส่งออดิโอที่ถูกตรวจจับในเซ็จชั่นหลายเซ็จชั่น

ตัวอย่างที่ 3 และตัวอย่างที่ 4 ทั้งการเข้ารหัสออดิโอและการส่งไปในหลายเซ็จชั่น ในตัวอย่างที่ ข้อมูลจะถูกเข้ารหัสในแบบ GSM ในตัวอย่างที่ ข้อมูลจะถูกเข้ารหัสในรูปแบบที่แตกต่าง

ตัวอย่างที่ 3 การส่งข้อมูลแบบ RTP ในหลายเซ็จชั่น

```
// First find a capture device that will capture linear audio
        // data at 8bit 8Khz

        AudioFormat format= new AudioFormat(AudioFormat.LINEAR, 8000, 8, 1);

        Vector devices= CaptureDeviceManager.getDeviceList( format);

        CaptureDeviceInfo di= null;

        if (devices.size() > 0) {
            di = (CaptureDeviceInfo) devices.elementAt( 0);
        }
        else {
           // exit if we could not find the relevant capturedevice.
           System.exit(-1);
        }

        // Now create a processor for this capturedevice & exit if we
        // cannot create it
        try {
           Processor p = Manager.createProcessor(di.getLocator());
        } catch (IOException e) {
           System.exit(-1);
        } catch (NoProcessorException e) {
           System.exit(-1);
        }

        // configure the processor
        processor.configure();

        // block until it has been configured

        processor.setContentDescriptor(
           new ContentDescriptor( ContentDescriptor.RAW));

        TrackControl track[] = processor.getTrackControls();

        boolean encodingOk = false;

        // Go through the tracks and try to program one of them to
        // output gsm data.

        for (int i = 0; i < track.length; i++) {
           if (!encodingOk && track[i] instanceof FormatControl) {

              if (((FormatControl)track[i]).
                 setFormat( new AudioFormat(AudioFormat.GSM_RTP, 8000, 8, 1)) ==
null) {
```

```
            track[i].setEnabled(false);
        }
        else {
            encodingOk = true;
        }
    }

    else {
        // we could not set this track to gsm, so disable it
        track[i].setEnabled(false);
    }
}

// At this point, we have determined where we can send out
// gsm data or not.
// realize the processor

if (encodingOk) {
    processor.realize();

    // block until realized.

    // get the output datasource of the processor  and exit
    // if we fail

    DataSource origDataSource = null;

    try {
        origDataSource = processor.getDataOutput();
    } catch (NotRealizedError e) {
        System.exit(-1);
    }

    // We want to send the stream of this datasource over two
    // RTP sessions.

    // So we need to clone the output datasource of the
    // processor and hand the clone over to the second
    // SessionManager

    DataSource cloneableDataSource = null;
    DataSource clonedDataSource = null;

    cloneableDataSource
      = Manager.createCloneableDataSource(origDataSource);

    clonedDataSource
      = ((SourceCloneable)cloneableDataSource).createClone();

    // Now create the first SessionManager and hand over the
    // first datasource for SendStream creation.

    SessionManager rtpsm1
      = new com.sun.media.rtp.RTPSessionMgr();

    // The session manager then needs to be
```

```
// initialized and started:
// rtpsm1.initSession(...);
// rtpsm1.startSession(...);

try {
   rtpsm1.createSendStream(cloneableDataSource, // Datasource 1
                           0);

} catch (IOException e) {
   e.printStackTrace();
} catch( UnsupportedFormatException e) {
   e.printStackTrace();
}

try {
   cloneableDataSource.connect();
   cloneableDataSource.start();
} catch (IOException e) {
   e.printStackTrace();
}

// create the second RTPSessionMgr and hand over the
// cloned datasource
if (clonedDataSource != null) {
   SessionManager rtpsm2
    = new com.sun.media.rtp.RTPSessionMgr();

   // rtpsm2.initSession(...);
   // rtpsm2.startSession(...);

   try {
      rtpsm2.createSendStream(clonedDataSource,0);
   } catch (IOException e) {
      e.printStackTrace();
   } catch( UnsupportedFormatException e) {
      e.printStackTrace();
   }
}
}
else {
   // we failed to set the encoding to gsm. So deallocate
   // and close the processor before we leave.

   processor.deallocate();
   processor.close();
}
```

และในตัวอย่างที่ 4 การเข้ารหัสออดิโอที่ถูกตรวจจับในรูปแบบที่หลากหลาย และจะส่งไปในเซ็จชั่นแบบอาร์ทีพี หลายๆเซ็จชั่น

DataSource อินพุตจะถูกถอดแบบและทำการ processor ตัวที่สองถูกสร้างจากการถอดแบบแทร็กใน Processor สองตัวถูกตั้งค่าเป็น GSM และ dvi และ DataSource เอาท์พุตถูกส่งไปใน session manager แบบอาร์ทีพี ที่แตกต่างกัน ถ้าจำนวนแทร็กมากกว่า 1 ตัวอย่างนี้พยายามตั้งค่าการเข้ารหัสของ

แทร็กหนึ่งเป็น gsm และอีกอัน หนึ่งเป็น dvi DataSource ที่เหมือนกันจะถูกจัดให้เป็น session managers แบบ RTP ที่แยกกันซึ่งดัชนีของสตรีมแรกจะถูกตั้งเป็น 0 และดัชนีของสตรีมที่สองจะถูกตั้งค่าเป็น 1

ตัวอย่างที่ 4 การเข้ารหัสและส่งข้อมูลในรูปแบบหลายรูปแบบ

```
// Find a capture device that will capture linear 8bit 8Khz
        // audio

        AudioFormat format = new AudioFormat(AudioFormat.LINEAR, 8000, 8, 1);

        Vector devices= CaptureDeviceManager.getDeviceList( format);

        CaptureDeviceInfo di= null;

        if (devices.size() > 0) {
            di = (CaptureDeviceInfo) devices.elementAt( 0);
        }
        else {
           // exit if we could not find the relevant capture device.
           System.exit(-1);
        }

        // Since we have located a capturedevice, create a data
        // source for it.

        DataSource origDataSource= null;

        try {
           origDataSource = Manager.createDataSource(di.getLocator());
        } catch (IOException e) {
           System.exit(-1);
        } catch (NoDataSourceException e) {
          System.exit(-1);
        }

        SourceStream streams[] = ((PushDataSource)origDataSource)
                                .getStreams();

        DataSource cloneableDataSource = null;
        DataSource clonedDataSource = null;

        if (streams.length == 1) {
           cloneableDataSource
             = Manager.createCloneableDataSource(origDataSource);
           clonedDataSource
             = ((SourceCloneable)cloneableDataSource).createClone();
        }
        else {
           // DataSource has more than 1 stream and we should try to
           // set the encodings of these streams to dvi and gsm
        }

        // at this point, we have a cloneable data source and its clone,
        // Create one processor from each of these datasources.
```

```
Processor p1 = null;

try {
    p1 = Manager.createProcessor(cloneableDataSource);
} catch (IOException e) {
    System.exit(-1);
} catch (NoProcessorException e) {
    System.exit(-1);
}

p1.configure();

// block until configured.

TrackControl track[] = p1.getTrackControls();
boolean encodingOk = false;

// Go through the tracks and try to program one of them
// to output gsm data
for (int i = 0; i < track.length; i++) {
    if (!encodingOk && track[i] instanceof FormatControl) {
        if (((FormatControl)track[i]).
            setFormat( new AudioFormat(AudioFormat.GSM_RTP, 8000, 8, 1)) == null)
{

            track[i].setEnabled(false);
        }
        else {
            encodingOk = true;
        }
    }
    else {
        track[i].setEnabled(false);
    }
}

if (encodingOk) {
    processor.realize();
    // block until realized.
    // ...
    // get the output datasource of the processor
    DataSource ds = null;

    try {
        ds = processor.getDataOutput();
    } catch (NotRealizedError e) {
        System.exit(-1);
    }

    // Now create the first SessionManager and hand over the
    // first datasource for SendStream creation .

    SessionManager rtpsm1
      = new com.sun.media.rtp.RTPSessionMgr();

    // rtpsm1.initSession(...);
```

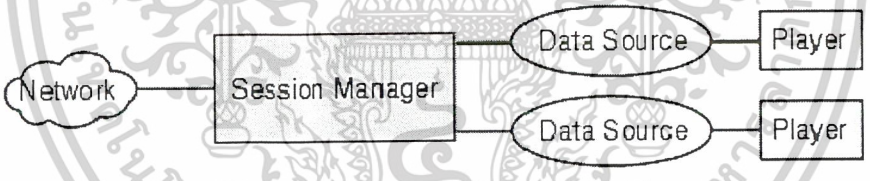
```
        // rtpsm1.startSession(...);

        try {
            rtpsm1.createSendStream(ds, // first datasource
                                    0); // first sourcestream of
                                        // first datasource
        } catch (IOException e) {
            e.printStackTrace();
        } catch( UnsupportedFormatException e) {
            e.printStackTrace();
        }
    }

    // Now repeat the above with the cloned data source and
    // set the encoding to dvi. i.e create a processor with
    // inputdatasource clonedDataSource
    // and set encoding of one of its tracks to dvi.
    // create SessionManager giving it the output datasource of
    // this processor.
```

2.11 การรับและแสดงผลสายของสื่อแบบอาร์ทีพี

อินเทอร์เฟส Playersและ Processors สนับสนุนการแสดงผล การจับ และเปลี่ยนกลไกของข้อมูลเป็นสายของสื่อแบบ



รูปที่ 2-24 แสดงการไหลของข้อมูลด้านรับแบบอาร์ทีพี

Player ที่แยกกันนั้นใช้สำหรับแต่ละสายของสื่อที่รับมาโดย session manager จากนั้นสามารถสร้าง Player สำหรับแต่ละสายของอาร์ทีพี โดยผ่านทางใช้กลไกของเม็ธท็อด createPlayer ของ Manager นอกจากนั้นยังสามารถสร้าง Player โดย

ใช้ MediaLocator ที่มีพารามิเตอร์ของอาร์ทีพี session และสร้าง Player โดยการเรียก Manager.createPlayer(MediaLocator)

สร้าง Player สำหรับ ReceiveStream โดยการกู้ DataSource จากสายของสื่อ และส่งต่อไปยัง Manager.creatPlayer(DataSource)

ถ้าใช้ MediaLocator สร้าง Player สามารถที่จะแสดงผลสายของสื่อแบบอาร์ทีพี ครั้งแรกที่ถูกตรวจ ถ้าต้อง การที่จะเล่น (blay back) สายของสื่อแบบอาร์ทีพี หลายๆ ตัวในหนึ่งกระบวนวิธี จำเป็นที่จะต้องใช้ SessionManager โดยตรงและสร้าง Player สำหรับแต่ละสายของสื่อที่ถูกรับ

การสร้าง Player สำหรับจัดการกับกลุ่มของอาร์ทีพี

เมื่อใช้ MediaLocator สร้าง Player สำหรับกลุ่มของอาร์ทีพี Manager จะสร้าง Player สำหรับสายของสื่อแรกที่ถูกตรวจพบในกลุ่ม Player นั้นจะแจ้งเหตุการณ์นี้ ซึ่งเราสามารถที่จะรับเอาคอมโพเนนต์ในการควบคุมจากมันได้

ข้อสังเกต: เพราะว่า Player สำหรับสายของสื่อแบบ RTP ไม่สามารถที่จะเข้าสู่สภาวะ realizing จนถึงเวลาที่ข้อมูลได้ถูกตรวจพบในกลุ่ม เราไม่ควรพยายามที่จะใช้ Manager.createRealizePlayer เพื่อที่จะสร้าง Player สำหรับ สายของสื่อแบบ RTP ไม่มี Player ที่จะถูกส่งค่ากลับจนกว่าข้อมูลจะมาถึงหรือไม่ก็ข้อมูลถูกตรวจพบ การพยายามที่จะทำให้ Player อยู่ในสภาวะ Realized จะถูกป้องกันโดยปริยาย

Player สามารถส่งออกการควบคุม RTP แบบระบุชนิดซึ่งก็คือ RTPControl

ตัวอย่างที่ 5 การสร้าง Player สำหรับ RTP กลุ่มหนึ่ง

```
String url= "rtp://224.144.251.104:49150/audio/1";

        MediaLocator mrl= new MediaLocator(url);

        if (mrl == null) {
            System.err.println("Can't build MRL for RTP");
            return false;
        }

        // Create a player for this rtp session
        try {
            player = Manager.createPlayer(mrl);
        } catch (NoPlayerException e) {
            System.err.println("Error:" + e);
            return false;
        } catch (MalformedURLException e) {
            System.err.println("Error:" + e);
            return false;
        } catch (IOException e) {
            System.err.println("Error:" + e);
            return false;
        }

        if (player != null) {
            if (this.player == null) {
                this.player = player;
                player.addControllerListener(this);
                player.realize();
            }
        }
```

การฟังการเปลี่ยนแปลงรูปแบบ

เมื่อ Player แจ้งว่ามีการเปลี่ยนแปลงรูปแบบของข้อมูล (FormatChangeEvent) ถ้ามีการบอกว่าข้อ มูลที่รับอยู่เปลี่ยนแปลงเกิดขึ้น Players จะสร้างพร้อมกับ MediaLocator จะถูกเปลี่ยนแปลงโดยอัตโนมัติ ในกรณีโดยทั่วไป กระบวนการนี้จะเกี่ยวข้องกับการสร้าง Player ตัวใหม่เพื่อจัดการกับรูปแบบใหม่ งานประยุกต์ที่แสดงผลของสายของสื่อแบบอาร์ทีพีจำเป็นจะต้องฟัง FormatChangedEvents เพื่อที่จะสามารถตอบสนองต่อถ้ามี Player ตัวใหม่เกิดขึ้น

เมื่อ FormatChangeEvent ถูกแจ้ง จะมีการตรวจสอบว่าการควบคุมอ็อปเจ็คของ Player และคอมโพเน็นท์ที่มองเห็นเปลี่ยนหรือไม่ ถ้ามี Player ตัวใหม่จะถูกสร้างและจำเป็นที่จะต้องเอาคอมโพเนนต์ของ player ตัวเก่าออก และรับเอาคอมโพเนนท์ของอ็อปเจ็คตัวใหม่

ตัวอย่างที่ 6 การฟังการเปลี่ยนแปลงรูปแบบ

```
public synchronized void controllerUpdate(ControllerEvent ce) {
        if (ce instanceof FormatChangeEvent) {
            Dimension vSize = new Dimension(320,0);
            Component oldVisualComp = visualComp;

            if ((visualComp = player.getVisualComponent()) != null) {
              if (oldVisualComp != visualComp) {
                 if (oldVisualComp != null) {
                    oldVisualComp.remove(zoomMenu);
                 }

                 framePanel.remove(oldVisualComp);

                 vSize = visualComp.getPreferredSize();
                 vSize.width = (int)(vSize.width * defaultScale);
                 vSize.height = (int)(vSize.height * defaultScale);

                 framePanel.add(visualComp);

                 visualComp.setBounds(0,
                                      0,
                                      vSize.width,
                                      vSize.height);
                 addPopupMenu(visualComp);
              }
            }

            Component oldComp = controlComp;

            controlComp = player.getControlPanelComponent();

            if (controlComp != null)
            {
               if (oldComp != controlComp)
               {
                  framePanel.remove(oldComp);
                  framePanel.add(controlComp);
```

```
                if (controlComp != null) {
                    int prefHeight = controlComp
                                .getPreferredSize()
                                .height;

                    controlComp.setBounds(0,
                                vSize.height,
                                vSize.width,
                                prefHeight);
                }
            }
        }
    }
}
```

การสร้าง Player แบบ RTP สำหรับแต่ละสายของสื่อที่เข้ามาใหม่

เพื่อที่จะเล่นสายของสื่อที่รับเข้ามาทั้งหมดนั้น เราจำเป็นจะต้องสร้าง Player ที่แยกเฉพาะสำหรับแต่ละสายของสื่อ เมื่อสายของสื่อได้ถูกสร้างขึ้นมา manager ของเซ็จชั่นจะแจ้ง NewReceiveStreamEvent โดยทั่วไปแล้ว เราจำเป็นที่จะต้องลงทะเบียนเป็น ReceiveStreamListener และสร้าง Player สำหรับแต่ละ ReceiveStream เพื่อจะสร้าง Player เราจำเป็นจะต้องกู้เอา DataSource จาก ReceiveStream และส่งต่อไปยัง Manager.createPlayer

วิธีการที่จะสร้าง Player สำหรับแต่ละสายของสื่อที่ถูกรับมาในหนึ่งเซ็จชั่น

1. จัดตั้งเซ็จชั่น RTP
    a. ให้ทำการสร้าง SessionManager ขึ้นมายกตัวอย่างเช่น สร้างอินสแตนท์ของ com.sun.media.rtp.RTPSessionMgr.(RTPSessionMgr เป็นอิมพลีเมนต์ของ SessionManager)
    b. เรียก RTPSessionMgr addReceiveStreamListener เพื่อที่จะลงทะเบียนกับ listener
    c. เตรียม RTP session โดยการเรียก RTPSessionMgr initSession
    d. เริ่มต้น RTP session โดยเรียก RTPsessionMgr startSession

ตัวอย่างที่ 7 การจัดตั้งอาร์ทีพี session

```
public SessionManager createManager(String address,
                                    int port,
                                    int ttl,
                                    boolean listener,
                                    boolean sendlistener)
    {
        mgr = (SessionManager)new com.sun.media.rtp.RTPSessionMgr();

        if (mgr == null) return null;
        mgr.addFormat(new AudioFormat(AudioFormat.DVI_RTP, 44100, 4, 1, 18);
```

```
if (listener) mgr.addReceiveStreamListener(this);
if (sendlistener) new RTPSendStreamWindow(mgr);

// ask session mgr to generate the local participant's CNAME
String cname = mgr.generateCNAME();
String username = null;

try {
   username = System.getProperty("user.name");
} catch (SecurityException e){
   username = "jmf-user";
}

// create our local Session Address
SessionAddress localaddr = new SessionAddress();

try{
   InetAddress destaddr = InetAddress.getByName(address);

   SessionAddress sessaddr = new SessionAddress(destaddr,
                                 port,
                                 destaddr,
                                 port + 1);
   SourceDescription[] userdesclist= new SourceDescription[]
   {
      new SourceDescription(SourceDescription
                  .SOURCE_DESC_EMAIL,
                  "jmf-user@sun.com", 1, false),


      new SourceDescription(SourceDescription
                  .SOURCE_DESC_CNAME, cname, 1, false),

      new SourceDescription(SourceDescription
                  .SOURCE_DESC_TOOL,
                  "JMF RTP Player v2.0", 1, false) };

   mgr.initSession(localaddr, userdesclist, 0.05, 0.25);

   mgr.startSession(sessaddr,ttl,null);
 } catch (Exception e) {
   System.err.println(e.getMessage());
   return null;
 }

 return mgr;
}
```

2. ในเมธ็อดอัพเดต ReceiveStreamListener คอยเฝ้าดู NewReceiveStreamEvent ที่มีการแจ้งว่ามีสายข้อมูลถูกตรวจพบ

3. เมื่อ NewReceiveStreamEvent ได้ถูกตรวจพบแล้วนั้น ให้ทำการกู้เอา ReceiveStream จาก NewReceiveStreamEvent โดยเรียก getReceiveStream
4. ทำการกู้เอา RTP DataSource จาก ReceiveStream โดยการเรียก getDataSource ซึ่งเป็นแบบ PushBufferDataSource กับ Format ที่เฉพาะเจาะจงแบบ RTP ยกตัวอย่างเช่น ถ้าเข้ารหัสสำหรับ DVI จะเป็น DVI_RTP
5. ส่ง DataSource ไปให้ Manager.creatPlayer เพื่อที่จะสร้าง Player สำหรับ Player เพื่อจะถูกสร้างให้สำเสร็จ ปลั๊กอินที่จำเป็นสำหรับการเข้ารหัสและการคลายแพ็คเก็จของข้อมูลแบบ RTP ต้องมีอยู่

ตัวอย่างที่ 8 การฟัง New Receive Stream Events

```
public void update( ReceiveStreamEvent event)
   {
      Player newplayer = null;
      RTPPlayerWindow playerWindow = null;

      // find the sourceRTPSM for this event
      SessionManager source = (SessionManager)event.getSource();

      // create a new player if a new recvstream is detected
      if (event instanceof NewReceiveStreamEvent)
      {
         String cname = "Java Media Player";
         ReceiveStream stream = null;

         try
         {
            // get a handle over the ReceiveStream
            stream =((NewReceiveStreamEvent)event)
                  .getReceiveStream();

            Participant part = stream.getParticipant();

            if (part != null) cname = part.getCNAME();

            // get a handle over the ReceiveStream datasource
            DataSource dsource = stream.getDataSource();

            // create a player by passing datasource to the
            // Media Manager
            newplayer = Manager.createPlayer(dsource);
            System.out.println("created player " + newplayer);
         } catch (Exception e) {
            System.err.println("NewReceiveStreamEvent exception "
                        + e.getMessage());
            return;
         }

         if (newplayer == null) return;
```

```
            playerlist.addElement(newplayer);
            newplayer.addControllerListener(this);

            // send this player to player GUI
            playerWindow = new RTPPlayerWindow( newplayer, cname);
        }
    }
```

การจัดการการเปลี่ยนแปลง Payload อาร์ทีพี

ถ้ามีการเปลี่ยนแปลง Payload ของสายของสื่อในเซ็จชั่นของอาร์ทีพี ReceiveStream จะแจ้ง RemotePayloadChangeEvent โดยทั่วไปแล้ว เมื่อ Payload เปลี่ยนแปลง Player ที่เกิดขึ้นนั้นจะไม่สามารถจัดการกับ Format ที่เปลี่ยนแปลงได้และ JMF จะโยนข้อผิดพลาด ถ้าเราพยายามแสดงผลของ payload ใหม่ และเพื่อที่จะหลีกเลี่ยงเหตุการณ์ดังกล่าวแล้ว ReceiveStreamListener จำเป็นที่จะต้องเฝ้าระวังสำหรับ RemotePayloadChangeEvents และเมื่อ RemotePayloadChangeEvent ถูกลบออกแล้ว เราจำเป็นจะต้อง

1. ปิด player ที่เกิดขึ้น
2. ลบ listeners ทั้งหมดจำ player ที่ถูกลบออก
3. สร้าง Player ตัวใหม่กับ DataSource แบบ RTP ที่เหมือนกัน
4. สร้าง component ที่มองเห็นสำหรับ Player ตัวใหม่
5. ใส่ listeners ที่จำเป็นแก่ Player ตัวใหม่

ตัวอย่างที่ 9 : การจัดการกับ payload ที่เปลี่ยนแปลง

```
public void update(ReceiveStreamEvent event) {
        if (event instanceof RemotePayloadChangeEvent) {
            // payload has changed. we need to close the old player
            // and create a new player

            if (newplayer != null) {
                // stop player and wait for stop event
                newplayer.stop();

                // block until StopEvent received...

                // remove controllerlistener
                newplayer.removeControllerListener(listener);

                // remove any visual and control components
                // attached to this application
                // close the player and wait for close event
                newplayer.close();

                // block until ControllerClosedEvent received...

                try {
                    // when the player was closed, its datasource was
                    // disconnected. Now we must reconnect the data-
```

```
            // source before a player can be created for it.

            // This is the same datasource received from
            // NewReceiveStreamEvent and used to create the
            // initial rtp player

            rtpsource.connect();
            newplayer = Manager.createPlayer(rtpsource);

            if (newplayer == null) {
               System.err.println("Could not create player");
               return;
            }

            newplayer.addControllerListener(listener);
            newplayer.realize();

            // when the new player is realized, retrieve its
            // visual and control components
         } catch (Exception e) {
            System.err.println("could not create player");
         }
      }
   }
}
```

การควบคุมบัฟเฟอร์ ของสายของสื่อแบบอาร์ทีพีที่เข้ามา

เราสามารถที่จะควบคุมบัพเฟอร์ที่รับมาแบบอาร์ทีพีโดยผ่านทาง BufferControl ที่ส่งออกโดย SessionManager ซึ่งการควบคุมนี้จะอนุญาตให้เราตั้งพารามิเตอร์ขึ้นมาสองตัวความยาวของบัฟเฟอร์และเทร็สโฮลด์(threshold)

ความยาวของบัฟเฟอร์คือขนาดของบัฟเฟอร์ที่ถูกดูแลโดยตัวรับ เทร็สโฮลด์คือปริมาณที่น้อยที่สุด ของข้อมูลที่จะถูกแบ่ง โดยการควบคุมก่อนที่ข้อมูลแบบผลักจะออกไป หรืออนุญาตให้ข้อมูลถูกดึงเข้ามาข้อมูลพร้อมสำหรับอ็อปเจ็คนี้เมื่อเทร็สโฮลด์ที่น้อยที่สุดนี้มาถึง ถ้าปริมาณบัฟเฟอร์ของข้อมูลเหลือต่ำกว่าเทร็สโฮสด์ ข้อมูลจะถูกเก็บสะสมไว้จนกว่าจะถึงระดับเทร็สโอลด์

ค่าขนาดของบัฟเฟอร์และเทร็สโฮลด์จะถูกระบุเป็นมิลลิวินาที จำนวนบัฟเฟอร์ของแพ็คเก็จแบบออดิโดหรือเฟรมของวิดีโอจะขึ้นอยู่กับฟอร์แมทของสายของสื่อที่เข้ามา สายของสื่อที่รับเข้ามาแต่ละอย่างจะรักษาค่าที่กำหนดให้และค่าสูงสุดของมันทั้ง ความยาวของบัฟเฟอร์และเทร็สโฮลด์

เพื่อที่จะรับ BufferControl จากเซ็สชั่น เราต้องเรียก getControl บน SessionManager เราสามารถกู้คอมโพเน็นท์ของกราฟฟิคจาก BufferControl โดยการเรียก getControlcomponent

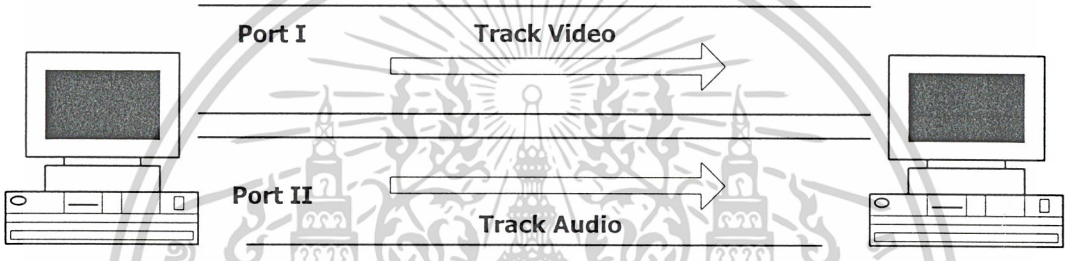
บทที่ 3

การออกแบบและการสร้าง

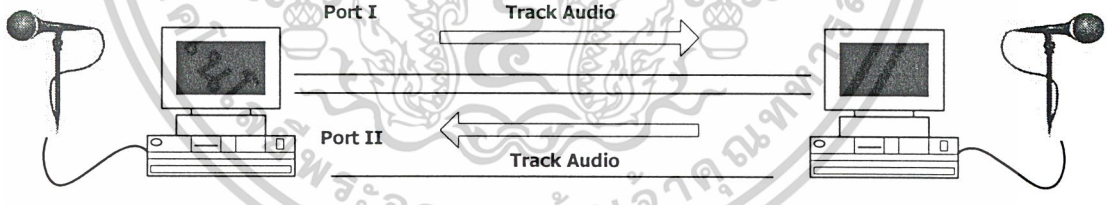
3.1 โครงสร้างโดยรวมของโปรแกรม

โปรแกรมที่ผู้จัดทำออกแบบนั้นสามารถทำได้ดังต่อไปนี้

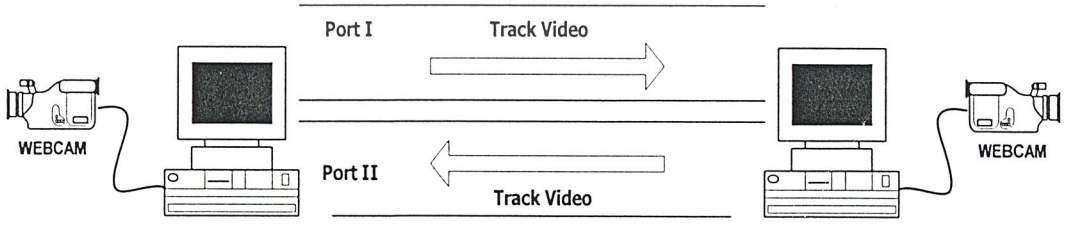
1. ส่งและรับไฟล์วีดีโอแบบเวลาจริง(Streaming video file) ผ่านระบบเครือข่ายท้องถิ่น
2. จับเสียงจากไมค์โครโฟนแล้วส่งและรับแบบเวลาจริงผ่านระบบเครือข่ายท้องถิ่น
3. จับวีดีโอจากเว็ปแคมแล้วส่งและรับแบบเวลาจริงผ่านระบบเครือข่ายท้องถิ่น
4. จับวีดีโอจากเว็ปแคมและเสียงจากไมโครโฟนแล้วส่งและรับแบบเวลาจริงผ่านระบบเครือข่ายท้องถิ่น



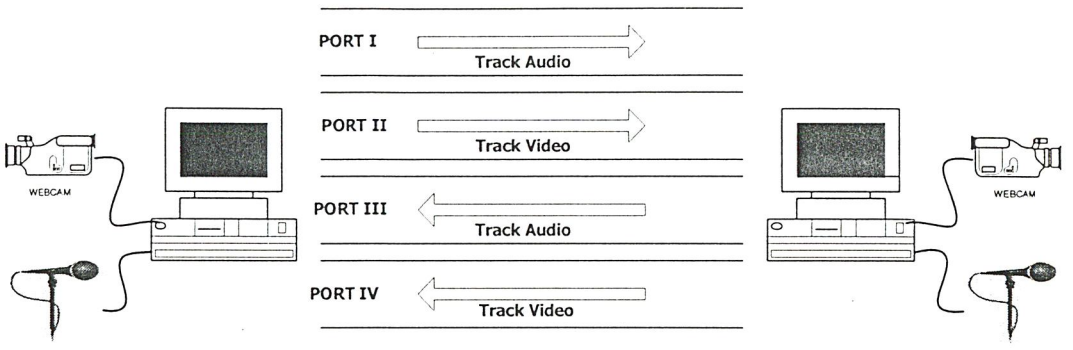
รูปที่ 3-1 แสดงการทำงานของโปรแกรมที่ออกแบบตามข้อ 1.



รูปที่ 3-2 แสดงการทำงานของโปรแกรมที่ออกแบบตามข้อ 2.



รูปที่ 3-3 แสดงการทำงานของโปรแกรมที่ออกแบบตามข้อ 3.

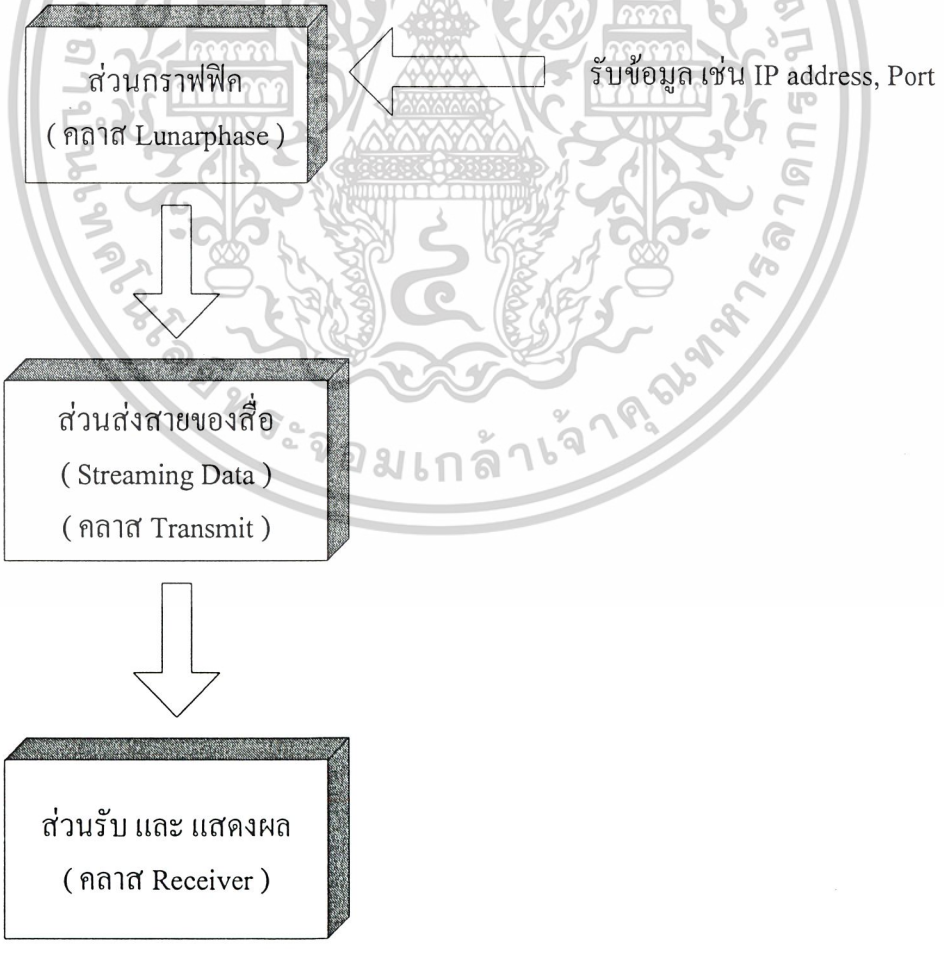


รูปที่ 3-4 แสดงการทำงานของโปรแกรมที่ออกแบบตามข้อ 4

โปรแกรมที่ได้ออกแบบแบ่งออกเป็นส่วนสำคัญ 3 ส่วน ดังนี้

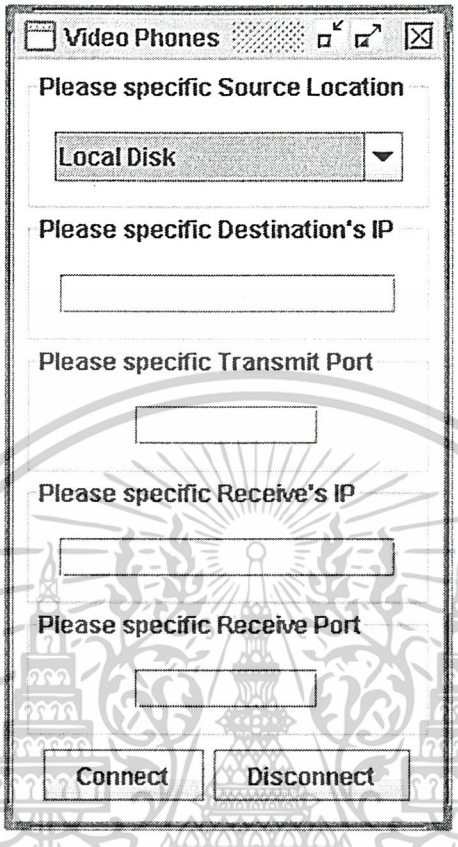
1. ส่วนส่งสายของสื่อ(Transmit.java)
2. ส่วนที่ใช้ในการรับและแสดงผล(Receive.java)
3. ส่วนกราฟฟิค(LunarPhase.java)

ซึ่งแต่ละส่วนที่ได้กล่าวมาข้างต้นนั้นได้เขียนแยกออกเป็นคลาสใหญ่ 3 คลาส ทั้งนี้เพื่อง่ายต่อการโปรแกรมและการพัฒนาโปรแกรม ตามรูปที่ แสดงความสัมพันธ์ระหว่างแต่ละส่วน



รูปที่ 3-5 แสดงความสัมพันธ์ระหว่างส่วนต่างๆของโปรแกรม

3.1.1. ส่วนกราฟฟิกซึ่งใช้ติดต่อกับผู้ใช้(คลาส LunarPhases)



รูปที่ 3-6 แสดงกราฟฟิกที่ติดต่อกับผู้ใช้

ตามรูปจะเห็นได้ว่าภายในเฟรมประกอบไปด้วยพาเนล (Panel) 7 พาเนล แบ่งออกเป็นพาเนลหลัก 1 พาเนล และพาเนลย่อย 6 พาเนล พาเนลหลักไว้สำหรับใส่พาเนลย่อยทั้งหก ต่อไปนี้เป็นตัวอย่างการสร้างพาเนล

```
//Create the phase selection and display panels.
selectPanel = new JPanel();                                          --------1
displayPanel = new JPanel();                                         --------2

//Add various widgets to the sub panels.                             --------3
addWidgets();                                                        --------4

//Create the main panel to contain the two sub panels.               --------5
mainPanel = new JPanel();                                            --------6
mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.PAGE_AXIS));  --------7
mainPanel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));       --------8

//Add the select and display panels to the main panel.               --------9
mainPanel.add(selectPanel);                                          --------10
mainPanel.add(displayPanel);                                         --------11
```

บรรทัดที่ 1 กับ บรรทัดที่ 2 เป็นการสร้างพาเนลย่อย บรรทัดที่ 5 เป็นการสร้างพาเนลหลัก บรรทัดที่ 7-8 เป็นการจัดชั้น (layout) และขอบเขต(Border) ระหว่างพาเนลหลักและย่อย บรรทัด 10-11 เป็นการใส่พาเนลย่อยในพาเนลหลัก

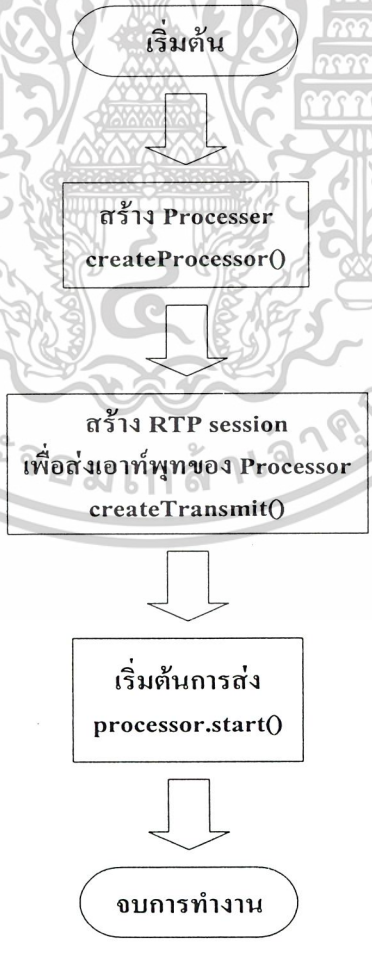
เม็ธท็อดที่สำคัญในคลาส LunarPhase นี้คือ actionPerformed เป็นเม็ธท็อดที่รับเหตุการณ์ของผู้ใช้ตลอดเวลา ยกตัวอย่างเช่น ผู้ใช้กดปุ่ม connect โปรแกรมจะทำงานในเม็ธท็อดนี้ ดังนั้นผู้จัดทำจึงได้ออกแบบให้มีการติดต่อกับคลาส Transmit ในเม็ธท็อดนี้

```
public void actionPerformed(ActionEvent event) {
        ........
        ........
        Transmit Tran = new Transmit();
        Tran.transmit();
}
```

ผู้จัดทำจะไม่อธิบายเกี่ยวกับกราฟฟิคมากนักเพราะสามารถหาอ่านได้จากหนังสือ java โดยทั่วไป

3.1.2.ส่วนส่งสายของสื่อ (คลาส Transmit)

การทำงานของส่วนนี้แสดงได้ตามโฟว์ชาร์ตดังรูปที่ 3-7

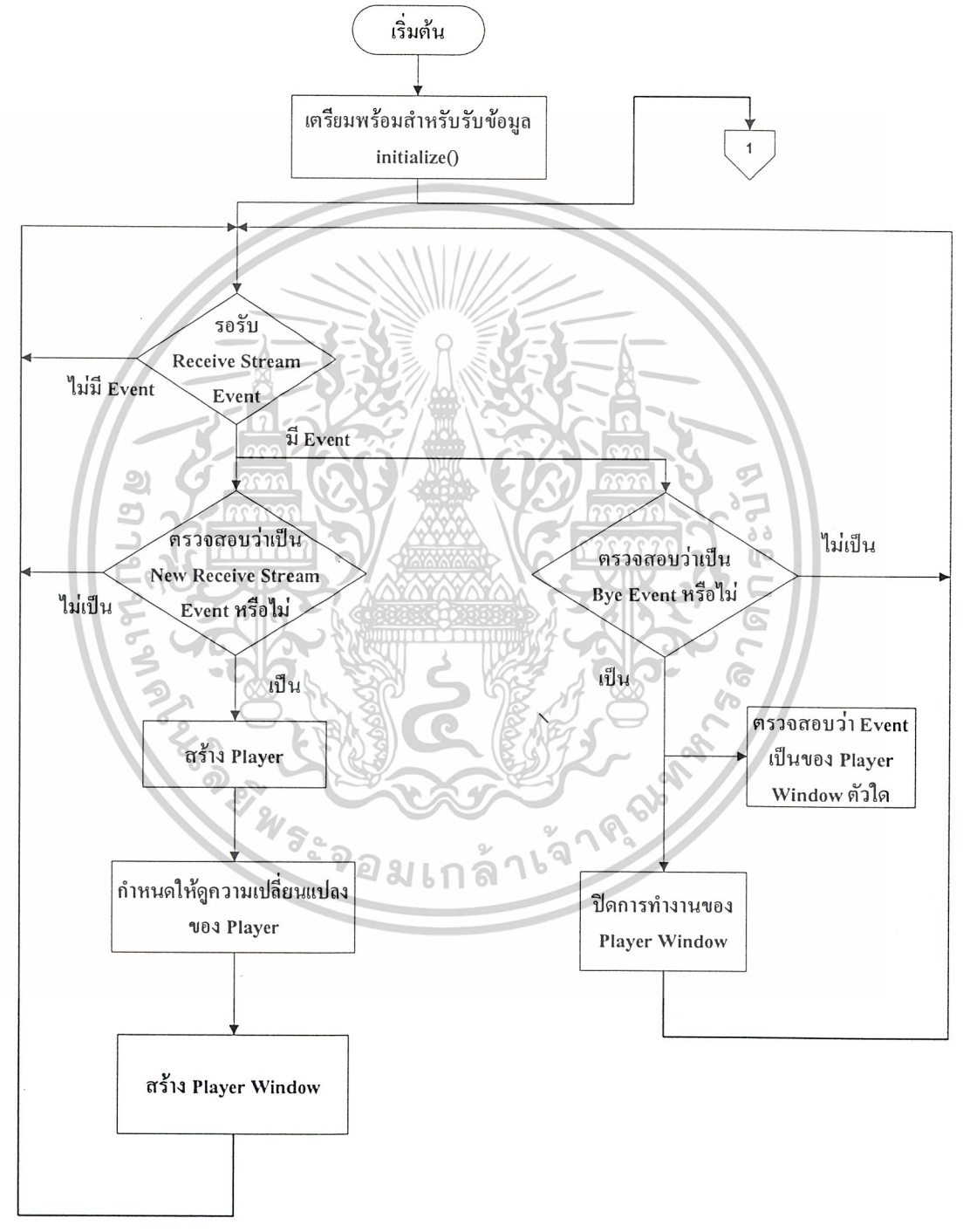


รูปที่ 3-7 โฟว์ชาร์ตแสดงการทำงานของคลาส Transmit

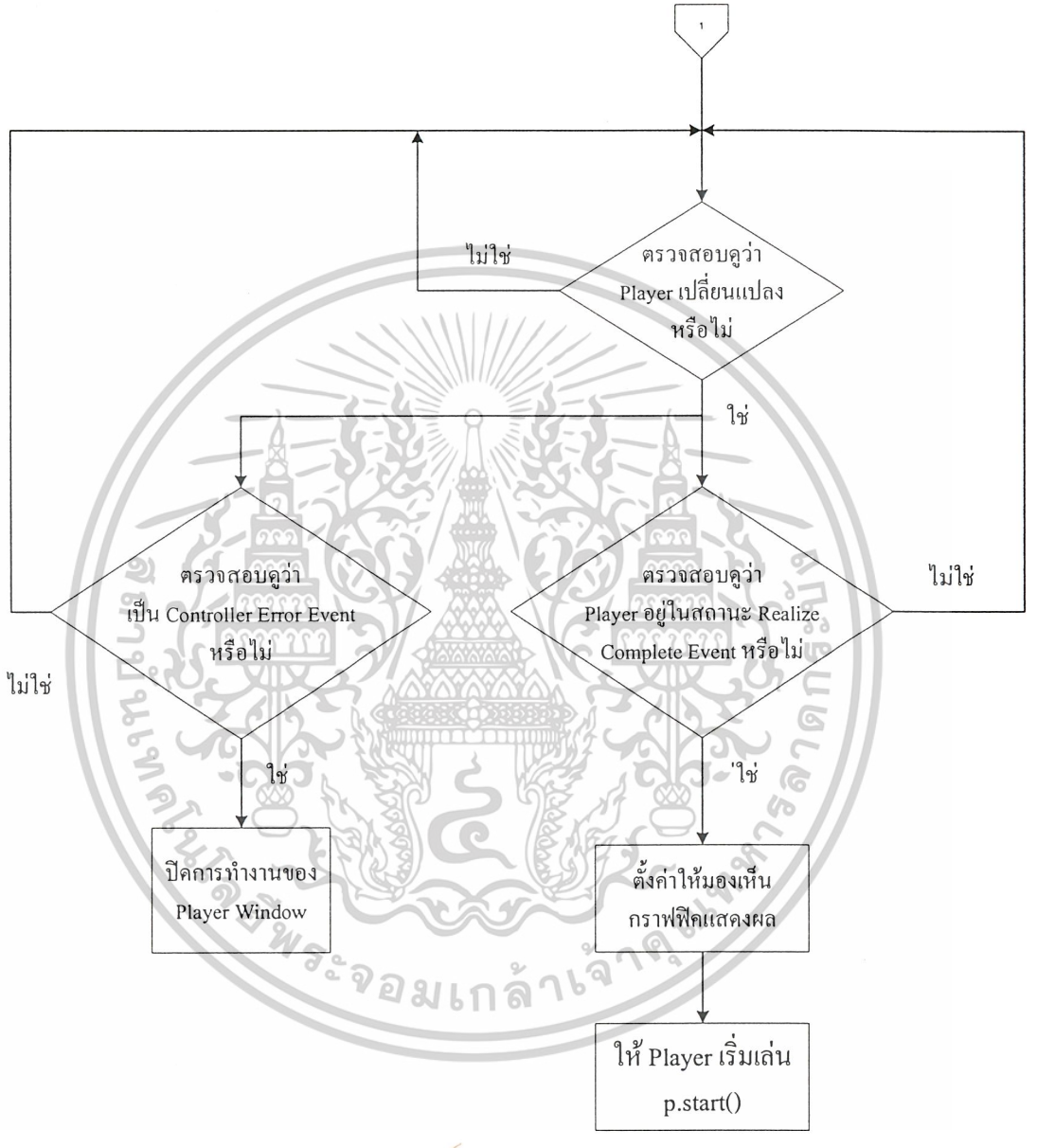
การสร้าง Processor และ RTP session สามารถหาดูได้จากส่วนเนื้อหา

3.1.3. ส่วนที่ใช้ในการรับและแสดงผล(คลาส Receive)

แสดงการทำงานของคลาสนี้ตามรูปที่ 3-8



รูปที่ 3-8ก โฟว์ชาร์ตแสดงการทำงานของคลาส Receive



รูปที่ 3-8ข โฟว์ชาร์ตแสดงการทำงานของคลาส Receive

3.2 วิธีการทำงานของโปรแกรม

การทำงานของโปรแกรมในส่วนการส่งและการรับ แบ่งได้เป็นสองส่วนคือ การทำงานทางฝั่งไคล์เอนท์(Client) และการทำงานทางฝั่งเซิร์ปเวอร์(Server) ซึ่งจะอธิบายการทำงานหลัก ๆ ของทั้งฝั่งไคล์เอนต์และฝั่งเซริปเวอร์ก่อนที่จะอธิบายรายละเอียด

3.2.1 หลักการทำงานโดยหลักของฝั่งไคล์เอนท์และเซิร์ปเวอร์

1. เริ่มการทำงานของไคล์เอนท์เทร็ด(Client thread) โดยฝั่งไคล์เอนท์เชื่อมต่อกับเซิร์ปเวอร์บนพอร์ตที่กำหนดไว้คงที่ตลอด ซึ่งฝั่งเซิร์ปเวอร์จะคอยเฝ้าระวัง(monitor) กิจกรรม(activity)ที่เกิดขึ้นกับพอร์ตที่กำหนดไว้ตลอดเวลา
2. ฝั่งเซริปเวอร์กำหนดค่าพอร์ตใหม่ที่หาได้(available)ให้ไคล์เอนท์เพื่อที่จะเชื่อมต่อกัน
3. ฝั่งไคล์เอนท์ก็จะปล่อยพอร์ตเริ่มต้นที่ใช้ติดต่อกับเซิร์ปเวอร์ในข้อ 1. และปิดการเชื่อมต่อพอร์ตนั้น
4. ฝั่งไคล์เอนท์สร้างการเชื่อมต่อใหม่กับพอร์ตที่ฝั่งเซิร์ปเวอร์ให้มา
5. ฝั่งไคล์เอนท์และฝั่งเซิร์ปเวอร์เข้าสู่ชุดของการส่งสัญญาณแฮนท์เช็คกิ่ง(handshaking) ซึ่งจะเป็นการรักษาสภาวะของเซิร์ปเวอร์และไคล์เอนท์
6. ฝั่งเซิร์ปเวอร์ส่งข้อมูลทั้งหมดให้ไคล์เอนท์และส่ง bye ให้ไคล์เอนท์
7. การเชื่อมต่อถูกปิดและพอร์ตก็จะถูกปล่อย

การทำงานที่นำเสนอมาข้างต้นจะแตกต่างกับโปรแกรมในภาคผนวกบางส่วนคือโปรแกรมในภาคผนวกฝั่งไคล์เอนท์และฝั่งเซิร์ปเวอร์จะต้องระบุพอร์ตที่ใช้ในการเชื่อมต่อให้ตรงกันโดยการป้อนข้อมูลโดยผู้ใช้ ซึ่งผู้จัดทำมองเห็นว่าโปรแกรมไม่ยืดหยุ่นต่อการใช้งาน ผู้จัดทำจึงนำเสนอการทำงานที่ยืดหยุ่นกว่าโดยให้โปรแกรมเริ่มต้นระบุพอร์ตที่ใช้ในการติดต่อระหว่างเซิร์ปเวอร์กับไคล์เอนท์ หลังจากเชื่อมต่อกันแล้วฝั่งเซิร์ปเวอร์จะจัดหาพอร์ต(assign) ในการติดต่อใหม่ แล้วปิดการเชื่อมต่อพอร์ตเดิม ที่ทำเช่นนี้เพื่อให้ไคล์เอนท์หลาย ๆ ไคล์เอนท์สามารถติดต่อกับเซิร์ปเวอร์ได้ โดยเริ่มต้นเชื่อมต่อกับพอร์ตที่ระบุไว้คงที่ ซึ่งสามารถนำมาประยุกต์ในงานเช่น การประชุมผ่านอินเตอร์เน็ต

คำว่าเทร็ดที่ผู้จัดทำใช้นั้น ในทางโปรแกรมเทร็ดคือการแบ่งการทำงานของโปรแกรมออกเป็นโปรแกรมย่อยๆ ทำงานพร้อมกัน กล่าวคือหากโปรแกรมสร้างเทร็ดสองเทร็ด ซีพียูจะสลับทำงานระหว่างเทร็ดสองเทร็ดซึ่งเปรียบเสมือนมีซีพียูสองตัวทำงานให้กับเทร็ดแต่ละเทร็ด

3.2.2 การทำงานโดยละเอียดฝั่งไคล์เอนท์

1. กำหนดพอร์ตเริ่มต้นที่ใช้ติดต่อระหว่างฝั่งไคล์เอนท์และเซิร์ปเวอร์(กำหนดโดยตัวโปรแกรม)
2. รับไอพีแอดเดรสของฝั่งเซิร์ปเวอร์จากผู้ใช้
3. สร้างเทร็ดขึ้นมาเพื่อติดต่อกับฝั่งเซิร์ปเวอร์
4. สร้างซ็อคเก็ต(Socket) ขึ้นมาเพื่อเชื่อมต่อกับฝั่งเซิร์ปเวอร์โดยต้องระบุไอพีแอดเดรสของฝั่งเซิร์ปเวอร์และพอร์ตเริ่มต้น แนะนำให้ใช้คลาส Socket ในจาวา
5. รอรับพอร์ตที่ฝั่งเซิร์ปเวอร์จัดให้ผ่านทางซ็อคเก็ต
6. สร้างซ็อคเก็ตขึ้นมาใหม่โดยระบุไอพีแอดเดรสของฝั่งเซิร์ปเวอร์และพอร์ตที่ฝั่งเซิร์ปเวอร์จัดให้

7. ปิดการเชื่อมต่อเดิม ปล่อยพอร์ตเริ่มต้นให้เป็นอิสระ

/* เข้าสู่ชุดของสัญญาณ handshaking โดยอ่านและเขียนจากซ็อคเก็ต เมื่อรับการตอบจากฝั่งเซิร์ปเวอร์ ไคล์เอนท์จะเปลี่ยนจากสภาวะหนึ่งไปอีกสภาวะหนึ่ง สภาวะของไคล์เอนท์มีอยู่ 4 สภาวะคือ "ESTABLISH_CONNECTION","INITIALIZE","END_OF_TRANSMISSION","CLOSE_CONNECTION" */

8. ตั้งค่าสภาวะของไคล์เอนท์ให้อยู่ในสภาวะ "ESTABLISH_CONNECTION"
9. วนลูปเช็คสภาวะของไคล์เอนท์

ถ้า สภาวะของไคล์เอนท์อยู่ในสภาวะ "ESTABLISH_CONNECTION"

9.11 เชื่อมต่อกับฝั่งเซิร์ปเวอร์

9.12 รอ message "CONNECTION ESTABLISHED" จากฝั่งเซิร์ปเวอร์

9.13 ส่ง message "INITIALIZE" ไปยังฝั่งเซิร์ปเวอร์

9.14 เปลี่ยนสภาวะของไคล์เอนท์เป็น "INITIALIZE"

ถ้าสภาวะของไคล์เอนท์อยู่ในสภาวะ "INITIALIZE"

9.21 รอข้อมูลจากฝั่งเซิร์ปเวอร์

9.22 ถ้าข้อมูลไม่ใช่ "EOT" ให้นำไปแสดงผล

9.23 ถ้าข้อมูลเป็น "EOT" นั่นหมายถึงข้อมูลได้ถูกส่งหมดแล้ว

9.24 เปลี่ยนสภาวะเป็น "END_OF_TRANSMISSION"

ถ้าสภาวะของไคล์เอนท์อยู่ในสภาวะ "END_OF_TRANSMISSION"

9.31 รอให้เซิร์ปเวอร์ส่งจำนวนข้อมูล

9.32 การส่งสิ้นสุดลง

9.33 เปรียบเทียบกับจำนวนข้อมูลที่รับมา

9.34 ถ้าเหมือนกันให้ส่ง "OK" ถ้าไม่ส่ง "NOK"

9.35 เปลี่ยนสภาวะเป็น "CLOSE_CONNECTION"

ถ้าสภาวะของไคล์เอนท์อยู่ในสภาวะ "CLOSE_CONNECTION"

9.41 ส่ง "BYE" message ไปยังเซิร์ปเวอร์

9.42 ปิดการเชื่อมต่อ

9.43 ปิดซ็อกเก็ต

3.2.3 การทำงานโดยละเอียดฝั่งเซิร์ปเวอร์

1 สร้างเทร็ดขึ้นมาคอยเฝ้าระวังบนพอร์ตเริ่มต้นว่ามีการร้องขอจากไคล์เอนท์หรือไม่

2 สร้างซ็อคเก็ตขึ้นมาโดยระบุพอร์ตเริ่มต้น

3 รอรับการร้องขอบนพอร์ตที่เริ่มต้นโดยซ็อคเก็ตที่สร้างขึ้น

4 สร้างซ็อคเก็ตฝั่งเซิร์ปเวอร์ขึ้นมาใหม่โดยระบุพอร์ตที่สุ่มมา

5 ส่งพอร์ตที่สุ่มมาไปยังไคล์เอนท์

6 รอการเชื่อมต่อจากไคล์เอนท์

7 กำหนดสภาวะเซิร์ปเวอร์ให้อยู่ในสภาวะ “ESTABLISS_CONNECTION”

8. วนลูปเพื่อเช็คสภาวะของเซิร์ป และทำการส่งข้อมูลระหว่างเซิร์ปเวอร์และไคล์เอนท์ สภาวะ ของเซิร์ปเวอร์มี อยู่ 4 สภาวะคือ “ESTABLISH_CONNECTION”,”INITIALIZE”,”END_OF_TRANSMISSION”, ”CLOSE_CONNECTION”

ถ้าสภาวะของเซิร์ปเวอร์เป็น “ESTABLISH_CONNECTION”

8.12 ส่ง “ESTABLISH_CONNECTION” message ไปยังไคล์เอนท์

8.13 เปลี่ยนสภาวะเป็น “INITIALIZE”

ถ้าสภาวะของเซิร์ปเวอร์เป็น “INITIALIZE”

8.21 รอให้ไคล์เอนท์ส่ง “INITIALIZE” message เพื่อให้แน่ใจว่าไคล์เอนท์ได้รับ “ESTABLISH_CONNECTION”

8.22 หลังจากได้รับ message “INITIALIZE” จากไคล์เอนท์แล้ว ให้เริ่มเคลื่อนย้ายข้อมูลทั้งหมด

8.23 ส่ง “EOT” message ไปยังไคล์เอนท์

8.24 ส่งจำนวนของข้อมูลที่ส่งไปให้กับไคล์เอนท์

8.25 เปลี่ยนสภาวะเป็น “END_OF_TRANSMISSION”

ถ้าสภาวะของเซิร์ปเวอร์เป็น “END_OF_TRANSMISSION”

8.31 ส่ง “BYE” message ไปยังไคล์เอนท์

8.32 เปลี่ยนสภาวะเป็น “CLOSE_CONNECTION”

ถ้าสภาวะของเซิร์ปเวอร์เป็น “CLOSE_CONNECTION”

8.41 ปิดการเชื่อมต่อ

8.42 ปิดซ็อคเก็ต

3.2.4 การส่งและรับอาร์ทีซีพีแพ็คเก็ต

การประกอบ RTCP แพ็คเก็ต ซึ่งโปรแกรมจะตัดสินใจชนิดของ RTCP แพ็คเก็จที่จำเป็นต้องสร้างและส่งมันออกไป ถ้าเป็นผู้ส่ง RTP แพ็คเก็จ จะส่ง Sender Report(SR) ออกไป นอกนั้นจะส่ง Receiver Report(RR) ออกไป

SDES แพ็คเก็ตจะถูกต่อกับแพ็คเก็ต SR และ RR ถ้ามีการร้องขอ BYE โดยโปรแกรม BYE แพ็คเก็ตจะถูกส่งออกไป

การส่ง RTCP แพ็คเก็ตโดยสร้างเทร็ดที่แยกต่างหาก เทร็ดที่จะหลับ(sleep) เป็นเวลาชั่วขณะหนึ่งซึ่งจะคำนวณโดยอาศัยพารามิเตอร์ของ RTCP และการตอบรับ เมื่อเทร็ดออกจากสภาวะหลับ มันจะตัดสินใจว่าชนิดของ RTCP แพ็คเก็ตที่จำเป็นต้องส่ง และสร้าง RTCP แพ็คเก็จที่เหมาะสมและส่งมันออกไป

การทำงานของโปรแกรมนี้เป็นดังต่อไปนี้

1. กำหนดพอร์ตสำหรับส่ง RTCP แพ็คเก็ต
2. กำหนดแอดเดรสสำหรับส่ง RTCP แพ็คเก็ต
3. สร้างตัวนับจำนวนแพ็คเก็ต
4. สุ่มตัวเลขขึ้นมาหนึ่งตัว
5. ให้โปรแกรมหยุดทำงานจนกว่าจะมีอินเตอร์รัพ(interupted)

การรับ RTCP แพ็คเก็ต

1. สร้างเทร็ดขึ้นมาเพื่อรับ RTCP แพ็คเก็ต
2. เทร็ดสร้างซ็อคเก็ตขึ้นมา
3. เมื่อ RTCP แพ็คเก็ตถูกรับ ผู้รับต้องตรวจความถูกต้องของ RTCP แพ็คเก็ตแรกตามกฎต่อไปนี้

   Version ต้องเป็น 2

   Payload Type ของ RTCP แพ็คเก็ตต้องเป็น SR และ RR

   Padding ต้องเป็น 0 สำหรับแพ็คเก็ตแรก

4. เมื่อแพ็คเก็ตถูกตรวจสอบแล้ว มันจะถูกใช้กับประเภทของแพ็คเก็ต ยกตัวอย่างเช่น (Sender Report,Receive Port,BYE,Sdes);

การสร้างและส่ง RTP แพ็คเก็ต และการรับ RTP แพ็คเก็ต โปรแกรมจะสร้างเทร็ดขึ้นมาสองเทร็ดสำหรับรับและส่ง RTP แพ็คเก็ต

1 กำหนดแอดเดรสสำหรับ RTP แพ็คเก็ต

2 กำหนดพอร์ตสำหรับ RTP แพ็คเก็ต

3 กำหนดความยาวส่วนหัวของ RTP แพ็คเก็ตเท่ากับ 12

4 กำหนด PayloadType = 0

5 กำหนดไบต์แรกของส่วนหัวเป็น 0x80= 10000000

```
// +-+-+-+-+-+-+-+-+
// |V=2|P|X|  CC   |
// +-+-+-+-+-+-+-+-+
//  1 0 0 0 0 0 0 0 = 0x80
```

สองบิตแรกเป็นเวอร์ชั่น เวอร์ชั่น = 2, Padding 1 บิต = 0, extension (X): 1 บิต = 0, CSRC count (CC): 4 บิต = 0

6 กำหนด sequence number ขึ้นมา(16 บิต)

7 กำหนด timestamp ขึ้นมา(32 บิต)

การส่งแพ็คเก็จโดยรับพารามิเตอร์เป็นอาร์เรย์ของไบต์

8 ใส่ version,Padding,extension,csrc ใน RTP แพ็คเก็ต

9 ใส่ timestamp จำนวน 4 ไบต์ ใน RTP แพ็คเก็ต

10 ใส่ sequence number จำนวน 2 ไบต์ ใน RTP แพ็คเก็ต

11 ใส่ SSRC จำนวน 4 ไบต์ ใน RTP แพ็คเก็ต

12 ใส่ข้อมูลหลังส่วนหัว

13 ทุกครั้งที่สร้างหนึ่งแพ็คเก็ตให้บวก sequence number ขึ้น 1

14 สร้าง Datagram Packet

15 ส่งไปโดยซ็อคเก็ตที่สร้างขึ้น

16 ปรับปรุงสถานะตัวเองให้เป็น Active Sender

17 กำหนดค่าเวลาที่ส่งแพ็คเก็จสุดท้ายให้เท่ากับเวลาปัจจุบัน

18 นับจำนวนแพ็คเก็จขึ้นอีกหนึ่ง

การรับ RTP แพ็คเก็ต

สร้างซ็อคเก็ตขึ้นมารับแพ็คเก็ต

1. สร้าง Datagram แพ็คเก็ตโดยกำหนดบัพเฟอร์ให้เหมาะสม
2. รับข้อมูลมาจาก Datagram แพ็คเก็ตที่สร้างขึ้น
3. กู้เอา Payload Type,Sequence Number,TimeStamp,SSRC,
4. กู้เอาข้อมูล ซึ่งก็คือส่วนที่อยู่หลังจากส่วนหัวที่กำหนดไว้ 12 ไบต์
5. สามารถระบุแหล่งที่ส่งแพ็คเก็ตมาจากข้อมูล SSRC
6. ใส่ค่าเวลาที่แพ็คเก็ตสุดท้ายเข้ามาเท่ากับเวลาปัจจุบัน
7 เปลี่ยนแปลง sequence number
8 เพิ่มจำนวนที่นับของ RTP แพ็คเก็ตขึ้น 1
9 ถ้ารับแพ็คเก็ตหมดแล้ว ให้ปิดซ็อคเก็ต

บทที่ 4

การทดลองและผลการทดลอง

โปรแกรมที่ผู้จัดทำออกแบบนั้นสามารถทำได้ดังต่อไปนี้

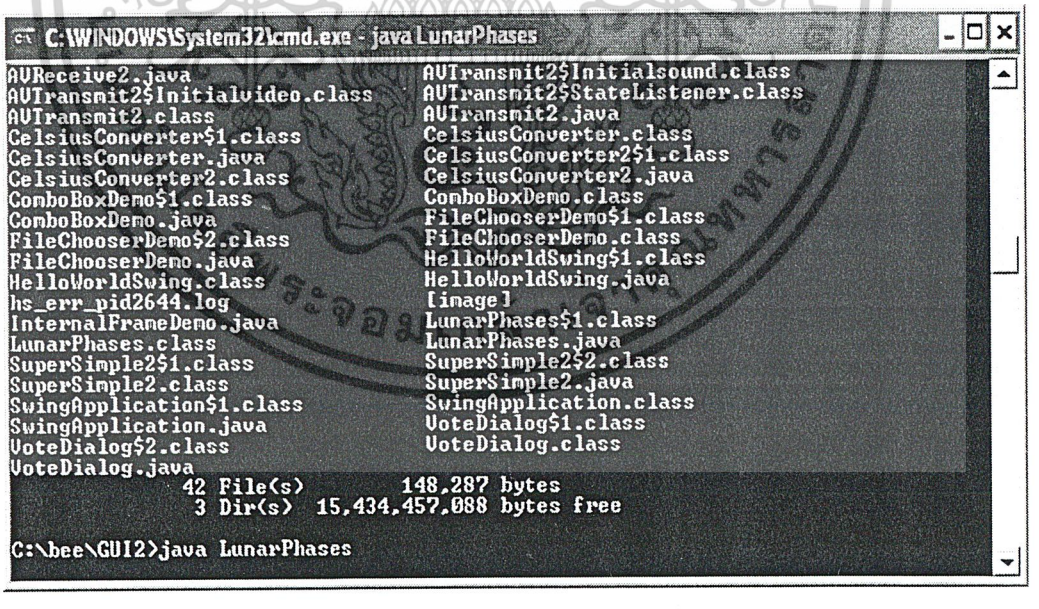
1. ส่งและรับไฟล์วีดีโอแบบเวลาจริง(Streaming video file) ผ่านระบบเครือข่ายท้องถิ่น
2. จับเสียงจากไมค์โครโฟนแล้วส่งและรับแบบเวลาจริงผ่านระบบเครือข่ายท้องถิ่น
3. จับวีดีโอจากเว็ปแคมแล้วส่งและรับแบบเวลาจริงผ่านระบบเครือข่ายท้องถิ่น
4. จับวีดีโอจากเว็ปแคมและเสียงจากไมโครโฟนแล้วส่งและรับแบบเวลาจริงผ่านระบบเครือข่ายท้องถิ่น

ในการทดลองจะทดสอบเฉพาะข้อ 1 และข้อ 4 เท่านั้น เพราะข้อ 2 และ 3 รวมอยู่ในข้อ 4 แล้ว

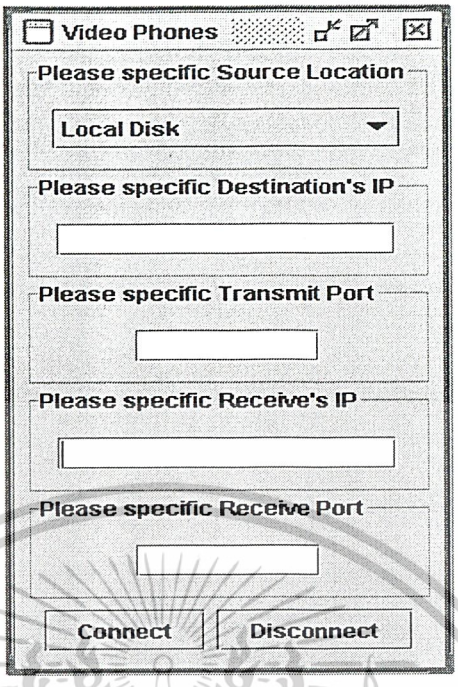
การทดลองที่ 1 การส่งไฟล์วีดีโอแบบเวลาจริง (Streaming video file) จากคอมพิวเตอร์เครื่องหนึ่งไปยังอีกเครื่องหนึ่ง

ขั้นตอนการทดลอง

1. รันโปรแกรม LunarPhases ที่ได้จากการคอมไพล์โปรแกรม LunarPhases.java โดยพิมพ์คำสั่งที่คอมมานด์พร็อมท์ดังนี้ “java LunarPhases” ตามรูป ซึ่งจะปรากฏรูปกราฟฟิคของตัวโปรแกรมตามรูปที่ 4-1

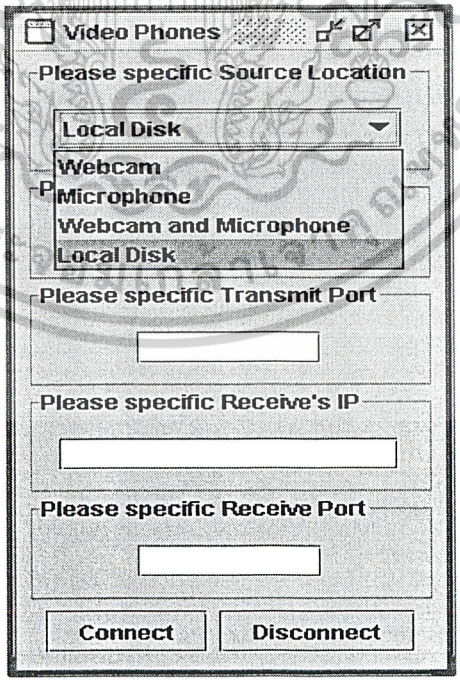


รูปที่ 4-1ก การรันโปรแกรม LunarPhases

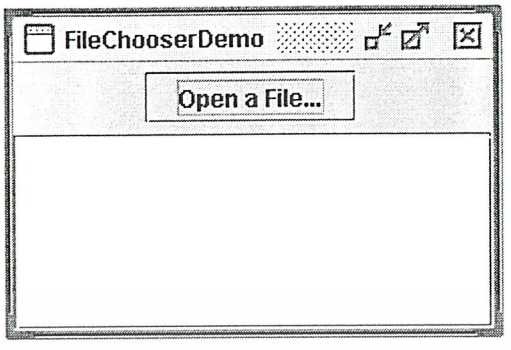


รูปที่ 4-1ข การรันโปรแกรม LunarPhases

2. ทำการเลือกที่ช่อง "Please specific Source Location" ให้เลือก "Local Disk" ซึ่งจะปรากฏกราฟฟิคให้เลือกไฟล์ ตามรูป 4-2

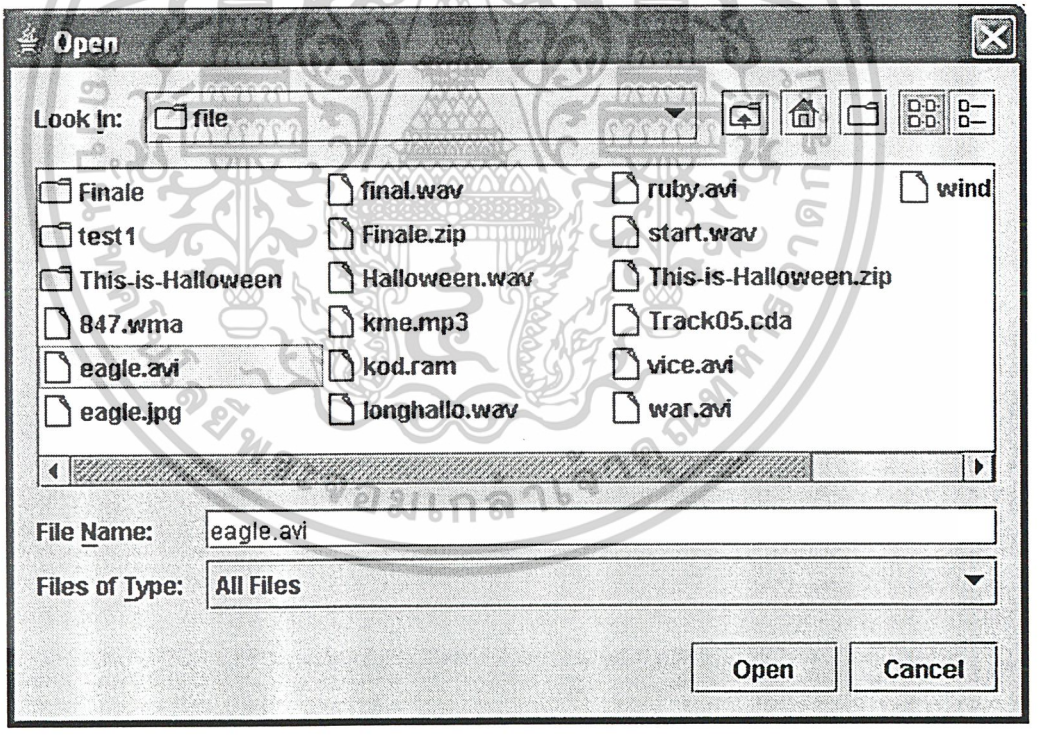


รูปที่ 4-2 รูปแสดงการเลือก Source Location



รูปที่ 4-3 หน้าต่างที่ได้จากการเลือก Source Location

3. ทำการเลือกไฟล์วีดีโอตามรูป 4-4



รูปที่ 4-4 แสดงการเลือกไฟล์

4. ทำการระบุไอพีแอดเดรสและพอร์ตของปลายทางที่ต้องการส่งตามรูปที่ 4-5 และกดปุ่ม connect

รูปที่ 4-5 แสดงการ connect

5.ทางคอมพิวเตอร์ด้านรับให้รันโปรแกรม LunarPhases และระบุไอพีแอดเดรสของต้นทางและรับพอร์ตเดียวกันกับที่ใช้ส่ง และกดปุ่ม connect ตามรูป 4-6

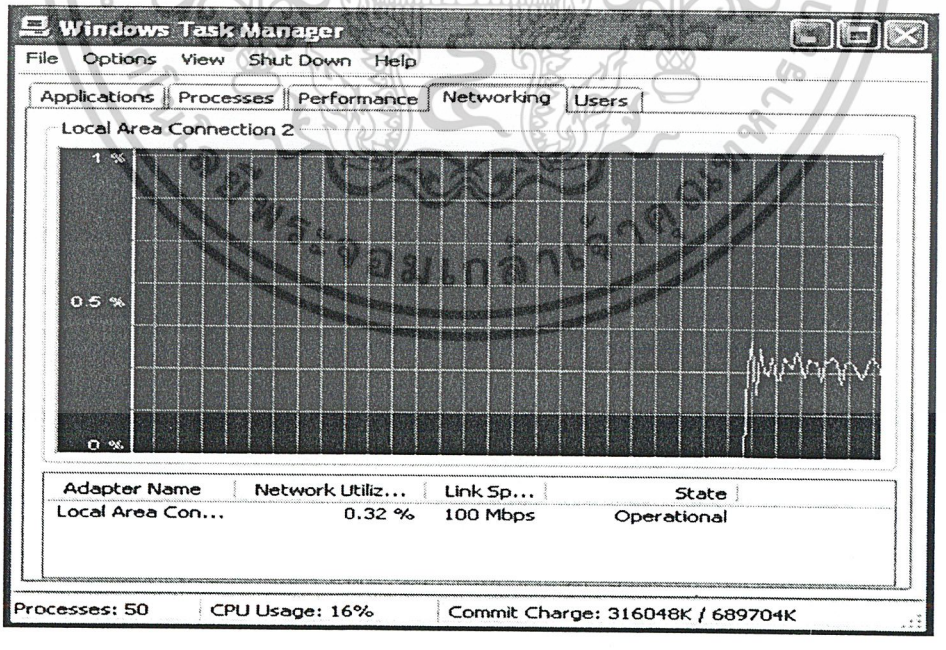
รูปที่ 4-6 รันโปรแกรมทางด้านรับ

ผลการทดลอง

สามารถส่งสตรีมข้อมูลวีดีโอซึ่งเป็นไฟล์ชนิด avi ได้ ซึ่งทางฝั่งรับจะเล่นไฟล์วีดีโอที่ส่งมาแบบเวลาจริงและถ้าลองหยุดโปรแกรมส่ง ทางฝั่งรับจะหยุดการเล่นวีดีโอ

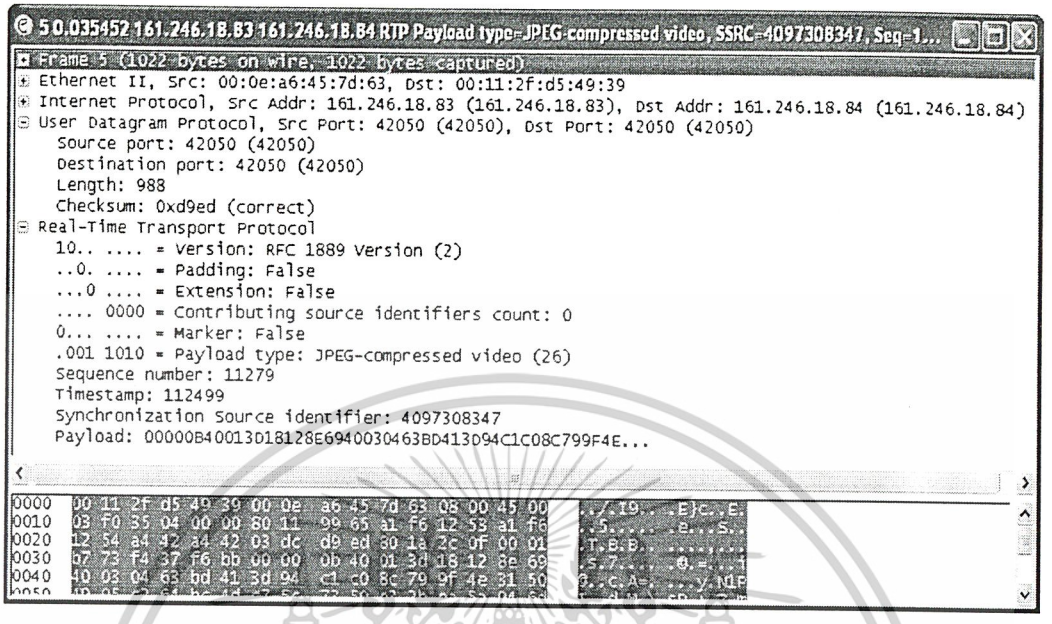
รูปที่ 4-7 ภาพแสดงผลการเล่นไฟล์วีดีโอทางฝั่งรับ

และทำการวัดแบนด์วิดที่ได้จากการใช้โปรแกรม Window Task Manager มีค่าประมาณ 300 kbps



รูปที่ 4-8 ภาพที่ได้จากการวัดแบนด์วิดโดยใช้โปรแกรม Windows Task Manager ทางฝั่งส่ง

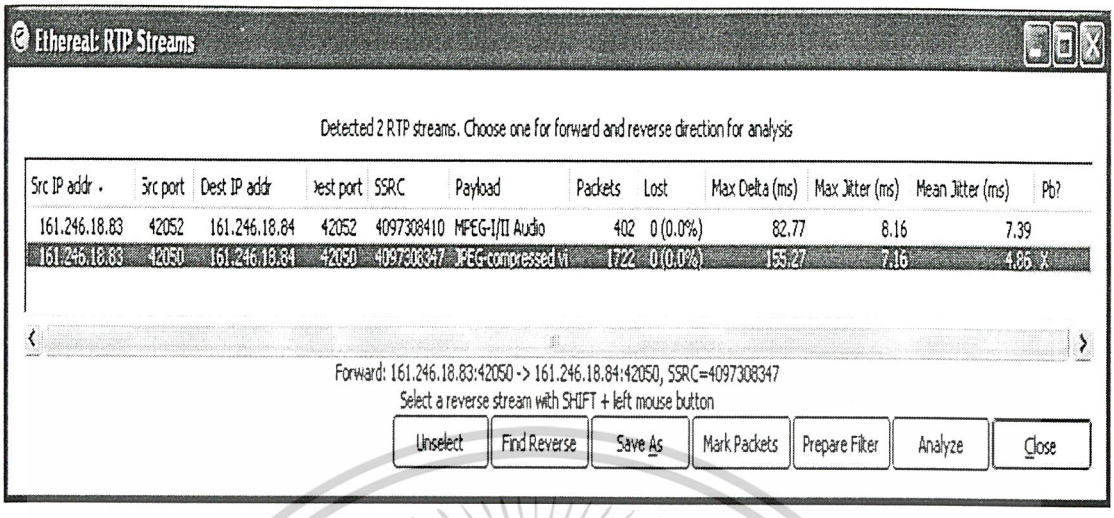
แล้วทำการจับแพ็คเกจและวัดผลโดยโปรแกรม Ethereal ได้ผลต่างๆดังนี้



รูปที่ 4-9 ภาพที่แสดงข้อมูลแพคเกจอาทีพีในเครือข่ายที่จับโดย Ethereal

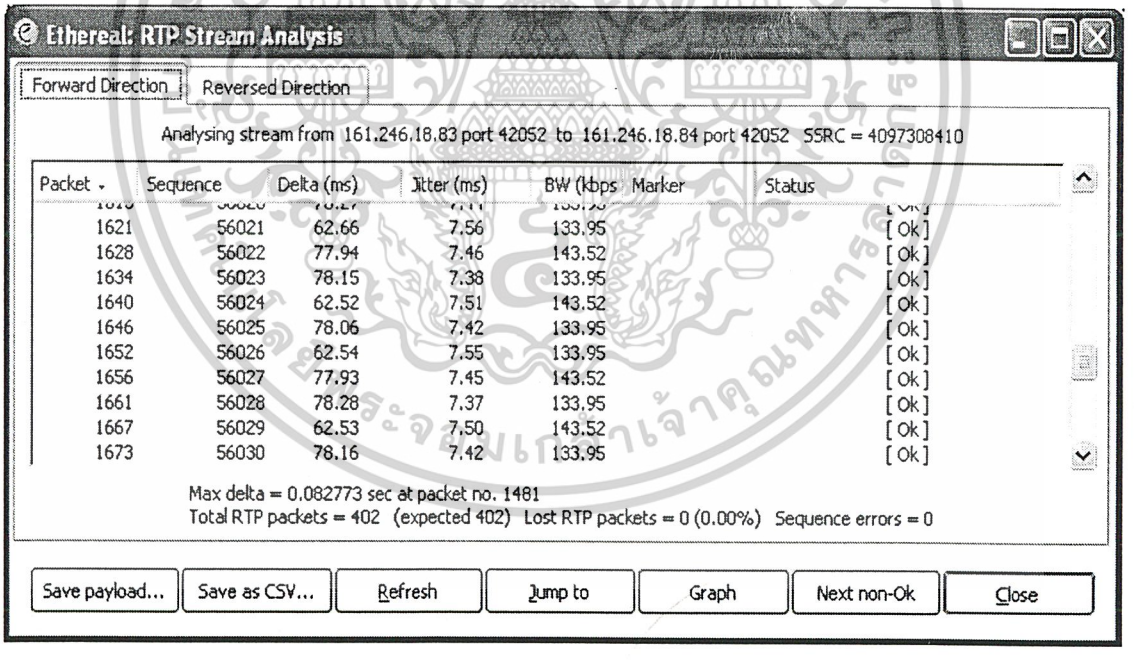
รูปที่ 4-10 แสดงผลค่าเฉลี่ยของแพ็คเกจและจำนวนบิตต่ออินาที

ได้ค่า Average packets/sec = 73.950, Average Mbit/sec = 0.539



รูปที่ 4-11 แสดงผลแพ็คเกจลอสและดีเลย์ไทม์ในสตรีมอาทีพี

ได้ค่า Packets loss = 0% , Max Delta time = 155.27 ms

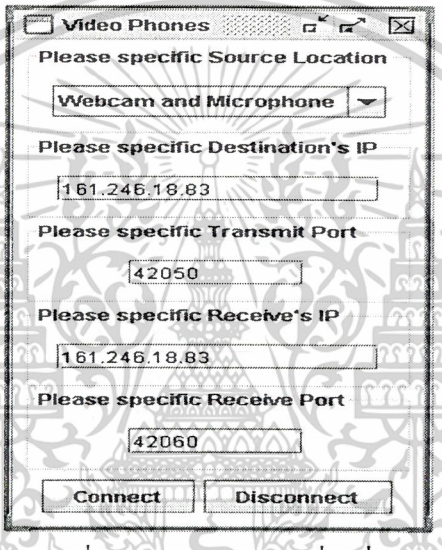


รูปที่ 4-12 แสดงกผลการวิเคราะห์ค่าต่างๆในแต่ละแพคเกจของอาทีพี

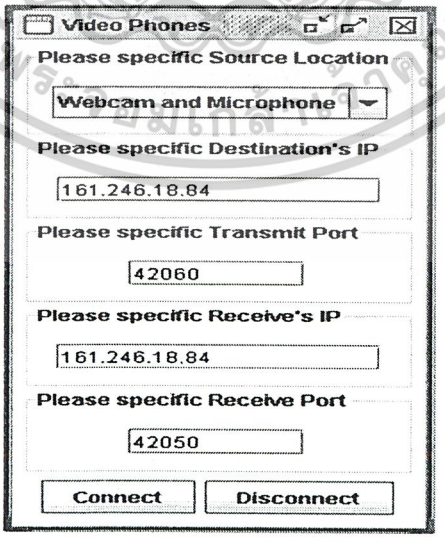
การทดลองที่ 2 จับวีดีโอจากเว็ปแคมและเสียงจากไมโครโฟนแล้วส่งและรับแบบเวลาจริงผ่านระบบเครือข่ายท้องถิ่น

ขั้นตอนการทดลอง

1. ที่คอมพิวเตอร์เครื่องที่ 1. ให้รันโปรแกรม LunarPhases แล้วทำการเลือกที่ช่อง "Please specific Source Location" ให้เลือก "Webcam and Microphone" และระบุไอพีแอดเดรส และพอร์ทตามรูป ที่ 4-9
2. ที่คอมพิวเตอร์เครื่องที่ 2. ให้รันโปรแกรม LunarPhases แล้วทำการเลือกที่ช่อง "Please specific Source Location" ให้เลือก "Webcam and Microphone" และระบุไอพีแอดเดรสและพอร์ทตามรูป ที่ 4-10



รูปที่ 4-13 คอมพิวเตอร์เครื่องที่ 1



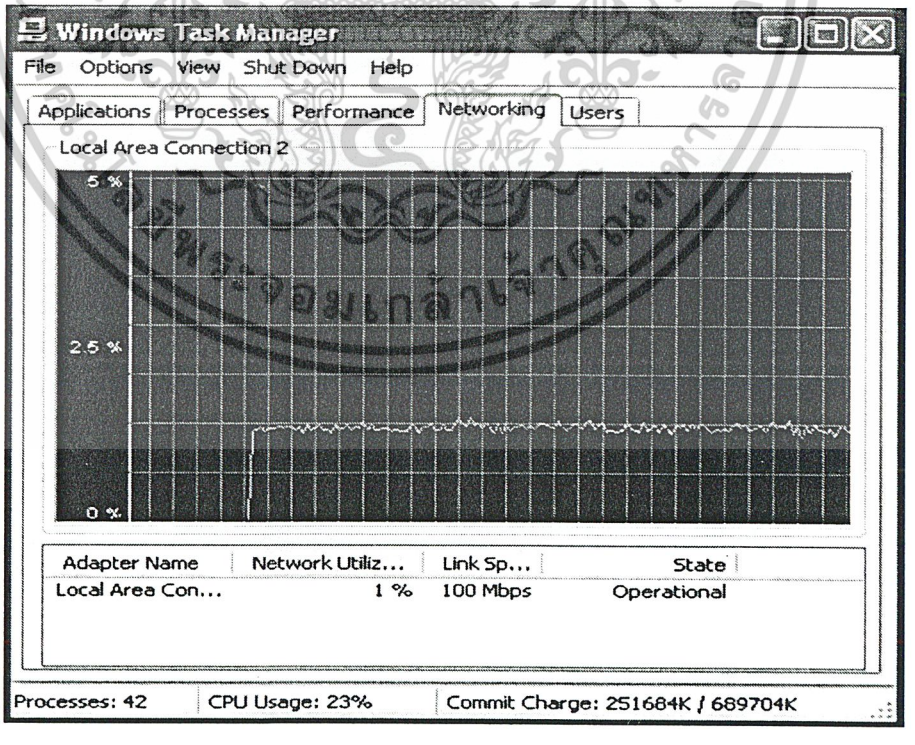
รูปที่ 4-14 คอมพิวเตอร์เครื่องที่ 2

ผลการทดลอง

สามารถจับวีดีโอจากเว็ปแคมและเสียงจากไมโครโฟนและส่งแบบเวลาจริงไปยังคอมพิวเตอร์อีกเครื่องในเครือข่ายท้องถิ่นได้

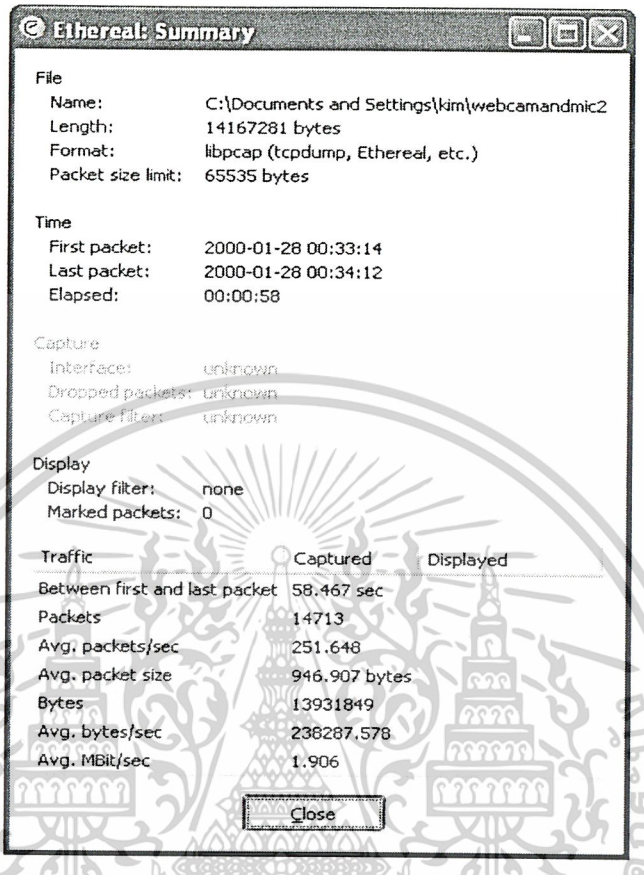
รูปที่ 4-15 รูปที่ได้จากคอมพิวเตอร์เครื่องที่ 2

และทำการวัดแบนด์วิดที่ได้จากการใช้โปรแกรม Window Task Manager มีค่าประมาณ 1Mbit/sec



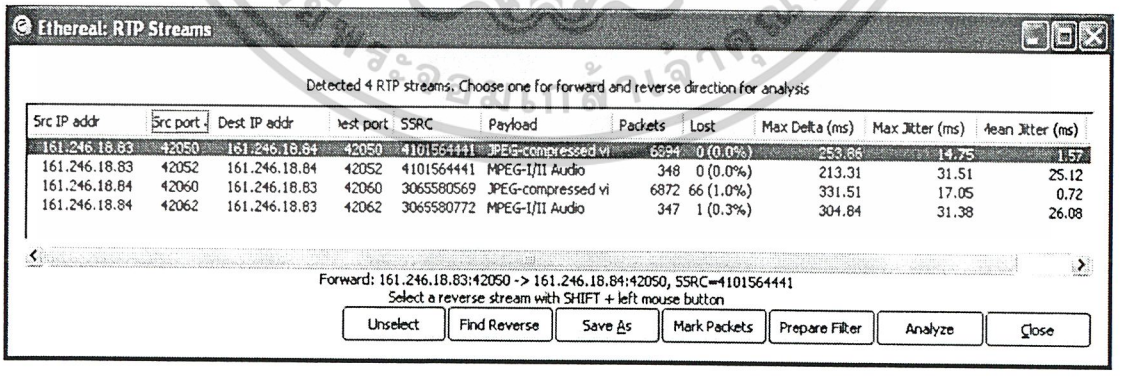
รูปที่ 4-16 ภาพที่ได้จากการวัดแบนด์วิดโดยใช้โปรแกรม Windows Task Manager

แล้วทำการจับแพ็คเกจและวัดผลโดยโปรแกรม Ethereal ได้ผลต่างๆดังนี้



รูปที่ 4-17 แสดงผลค่าเฉลี่ยของแพ็คเกจและจำนวนบิตต่อวินาที

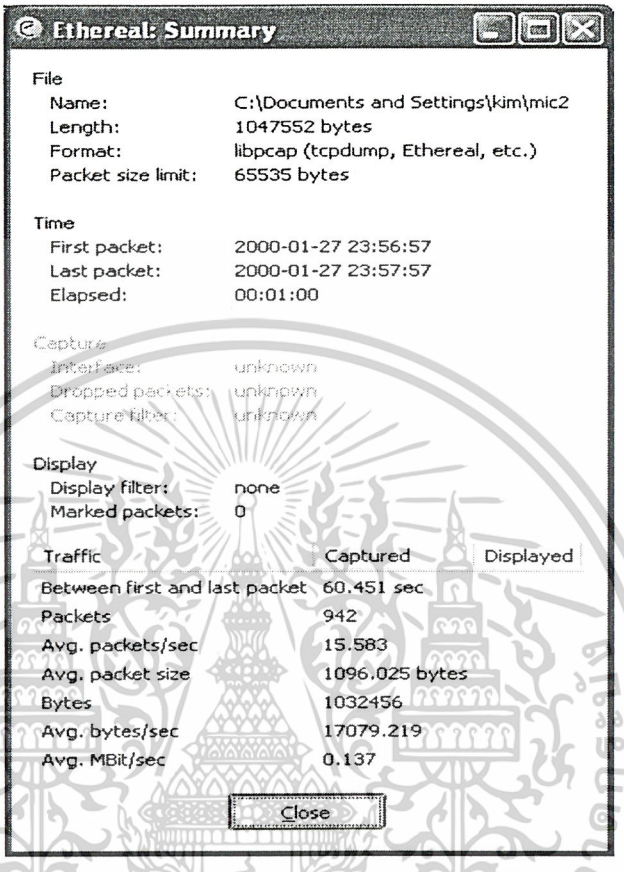
ได้ค่า Average packets/sec = 251.648, Average Mbit/sec = 1.906



รูปที่ 4-18 แสดงผลแพ็คเกจลอสและดีเลย์ไทม์ในสตรีมอาทีพี

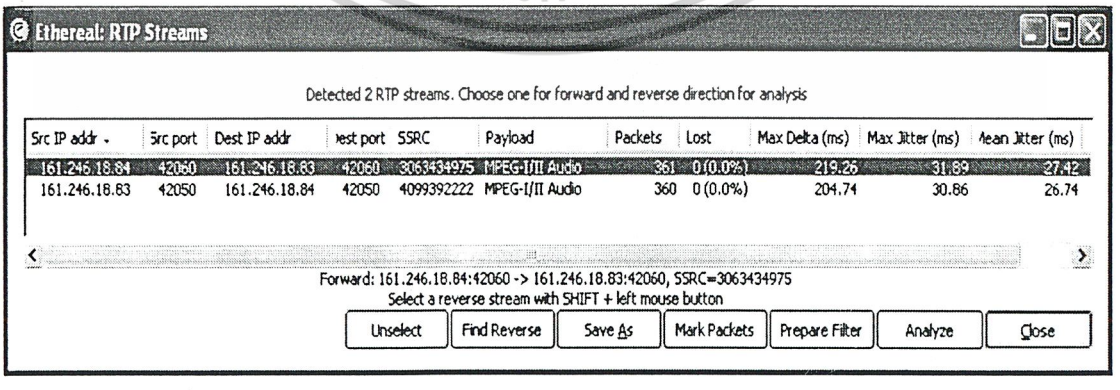
ได้ค่า Packets loss = 1.3 % , Max Delta time = 331.51 ms

กรณีที่เลือกส่งเฉพาะสัญญาณสียงที่ทำการจับจากไมค์โดยโต้ตอบกันระหว่างเครื่องคอมพิวเตอร์ทั้ง 2 จะได้ผลดังนี้



รูปที่ 4-19 แสดงผลค่าเฉลี่ยของแพ็คเกจและจำนวนบิตต่อวินาที

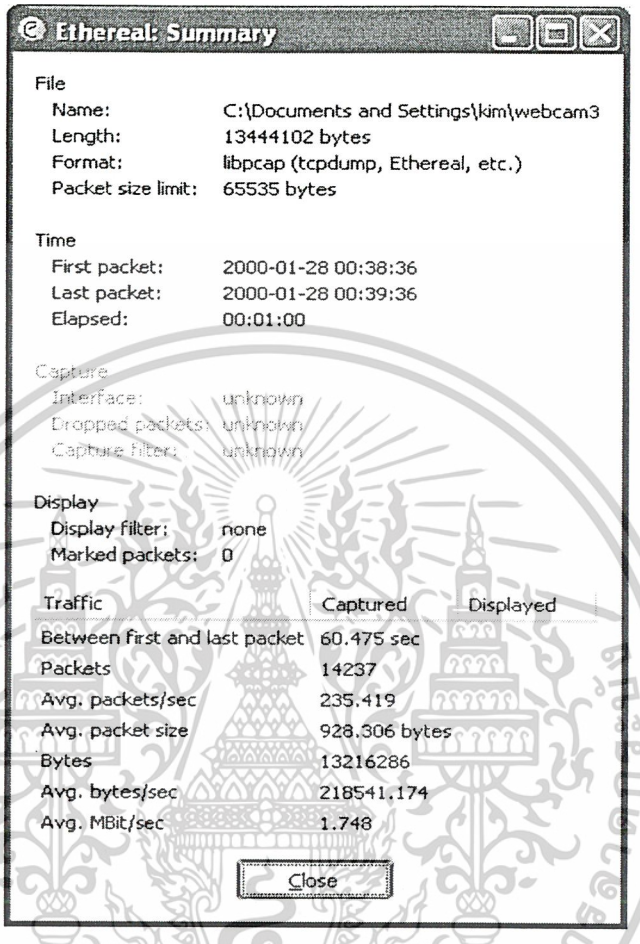
ได้ค่า Average packets/sec = 15.583 , Average Mbit/sec = 0.137



รูปที่ 4-20 แสดงผลแพ็คเกจลอสและดีเลย์ไทม์ในสตรีมอาทีพี

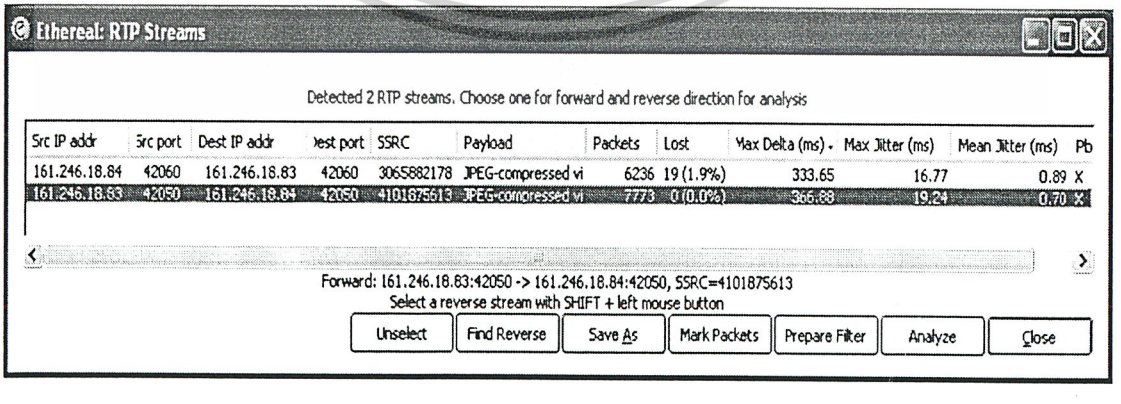
ได้ค่า Packets loss = 0% , Max Delta time = 219.26 ms

กรณีที่เลือกส่งเฉพาะสัญญาณวีดีโอที่ทำการจับจากเวบแคมโดยโต้ตอบกันระหว่างเครื่องคอมพิวเตอร์ทั้ง 2 จะได้ผลดังนี้



รูปที่ 4-21 แสดงผลค่าเฉลี่ยของแพ็คเกจและจำนวนบิตต่อวินาที

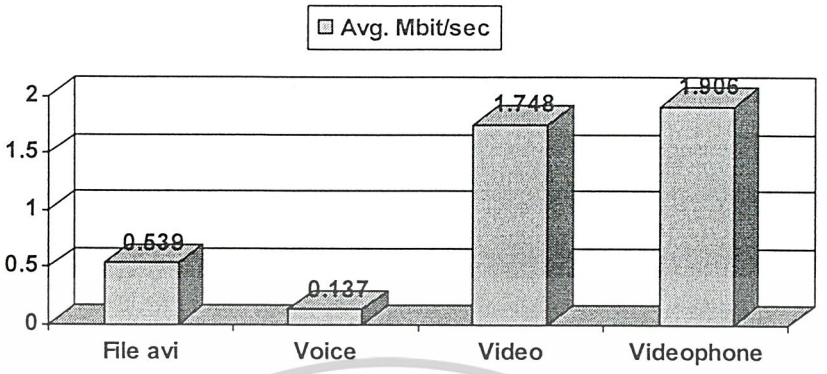
ได้ค่า Average packets/sec = 235.419 , Average Mbit/sec = 1.748



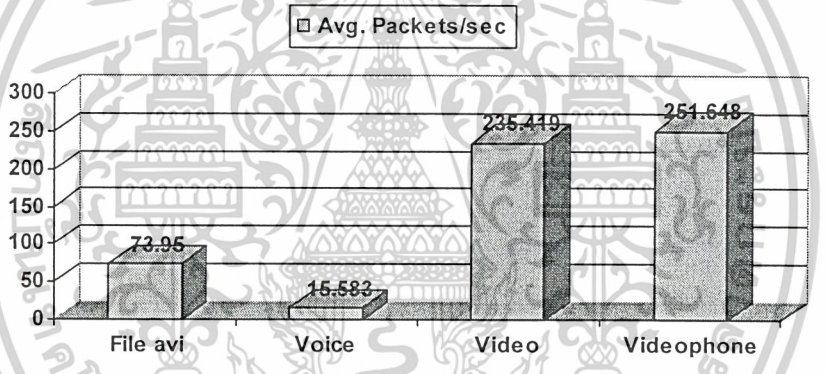
รูปที่ 4-22 แสดงผลแพ็คเกจลอสและดีเลย์ไทม์ในสตรีมอาทีพี

ได้ค่า Packets loss =1.9% , Max Delta time = 366.88 ms

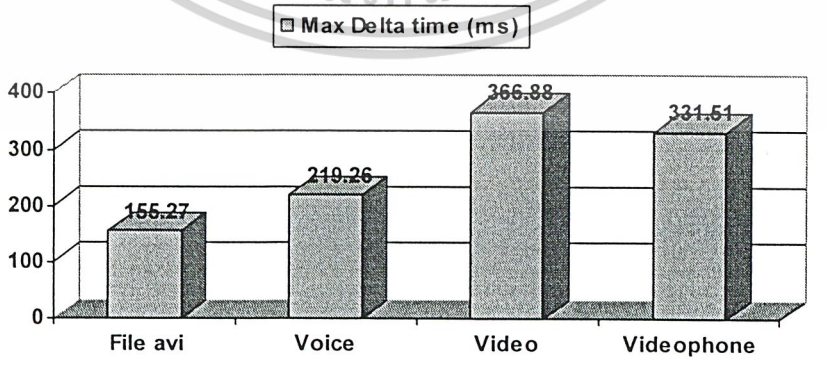
เปรียบเทียบผลการส่งข้อมูลในรูปแบบต่างๆตามการทดลองได้ดังนี้



รูปที่ 4-23 แสดงภาพแผนภูมิเปรียบเทียบค่า Average Mbit/sec ของการทดลอง



รูปที่ 4-24 แสดงภาพแผนภูมิเปรียบเทียบค่า Average Packets/sec ของการทดลอง



รูปที่ 4-25 แสดงภาพแผนภูมิการเปรียบเทียบค่า Max Delta time (ms)

บทที่ 5
บทสรุปและวิจารณ์

ปริญญานิพนธ์นี้เป็นการนำเสนอการส่งวีดีโอผ่านทางไอพีโปรโตคอลบนเครือข่ายท้องถิ่น โดยรับอินพุตจากไฟล์วีดีโอ มาตรฐานที่ใช้ในการส่งเป็นแบบ RTP (Real-time Transport Protocol)

ผู้จัดทำได้สร้างซอฟท์แวร์ที่สามารถรับอินพุตจากไฟล์วีดีโอ และส่งแบบ RTP ซึ่งผลที่ได้เป็นที่น่าพอใจ ผู้จัดทำยังได้พัฒนาต่อโดยสามารถรับอินพุตจากทั้งไมโครโฟนและเว็ปแคม(webcam) ซึ่งผู้ใช้สองคนสามารถคุยกันในเครือข่ายท้องถิ่นโดยเห็นภาพด้วย (video phone) ซึ่งถ้าหากพัฒนาต่อก็จะเป็นการประชุมผ่านอินเตอร์เน็ต (video conferencing) ซึ่งผู้จัดทำเห็นว่าได้เลยขอบเขตของปริญญานิพนธ์นี้ไปมากแล้ว จึงเหลือความท้าทายไว้ให้นักพัฒนารุ่นต่อไป

อนึ่งแบนด์วิดของการส่งไฟล์ภาพและเสียงของโปรแกรมนี้ประมาณ 300 Kbps ซึ่งหากมองเทียบกับแบนด์วิดของสายโทรศัพท์คือ 56 kbps และ 2 Mbps สำหรับ ADSL แล้ว ถือว่ามากเกินไป แต่ผู้จัดทำมองว่าแบนด์วิดขนาดนี้ไม่มากเกินไปเลยสำหรับเทคโนโลยีในอนาคต

การพัฒนาต่อที่น่าสนใจในความเห็นของผู้จัดทำคือ การนำโปรแกรมนี้ไปใช้ในคอมพิวเตอร์แบบฝังตัว (embedded pc) หรือในอุปกรณ์ประเภทพีดีเอ ซึ่งมีขนาดเล็ก ติดต่อเครือข่ายโดยระบบเครือข่ายท้องถิ่นแบบไร้สาย (wireless Lan) ผู้จัดทำมองว่าในอนาคต แทบทุกห้างสรรพสินค้าหรือสถานที่ต่างๆ จะมีการติดตั้งระบบเครือข่ายท้องถิ่นแบบไร้สาย ซึ่งหากสามารถพัฒนาต่อได้ อุปกรณ์ตัวนี้ก็เปรียบเสมือนมือถือติดตัวซึ่งได้เปรียบต่อผู้ใช้ตรงค่าบริการที่ต่ำซึ่งก็คือค่าบริการอินเตอร์เน็ต

ภาคผนวก

โปรแกรม LunarPhases.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.URL;
public class LunarPhases implements ActionListener {
    final static int NUM_IMAGES = 8;
    final static int START_INDEX = 3;
      public static String
Address,ReceiveAdd,Receiveport,Transmitport;
      public static int CHOICE;
    ImageIcon[] images = new ImageIcon[NUM_IMAGES];
    JPanel mainPanel, selectPanel, displayPanel, ReceivePortPanel,
           TransmitPortPanel,ReceiveAddPanel,ConnectPanel;
    JComboBox phaseChoices = null;
    JTextField DestIPText,ReceiveAddText,TransPortText,RePortText;
    JButton ConnectButton,DisconnectButton;
    public LunarPhases() {
        //Create the phase selection and display panels.
        selectPanel = new JPanel();
        displayPanel = new JPanel();
        ReceivePortPanel = new JPanel();
        TransmitPortPanel = new JPanel();
            ConnectPanel = new JPanel();
            ReceiveAddPanel = new JPanel();
        //Add various widgets to the sub panels.
        addWidgets();
        //Create the main panel to contain the two sub panels.
        mainPanel = new JPanel();
        mainPanel.setLayout(new BoxLayout(mainPanel,
BoxLayout.PAGE_AXIS));
        mainPanel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        //Add the select and display panels to the main panel.
        mainPanel.add(selectPanel);
        mainPanel.add(displayPanel);
        mainPanel.add(TransmitPortPanel);
        mainPanel.add(ReceiveAddPanel);
        mainPanel.add(ReceivePortPanel);

        mainPanel.add(ConnectPanel);
    }
    /*
     * Get the images and set up the widgets.
     */
    private void addWidgets() {
        DestIPText = new JTextField(15);
        TransPortText = new JTextField(8);
        RePortText = new JTextField(8);
        ReceiveAddText = new JTextField(15);
        String[] phases = { "Webcam", "Microphone", "Webcam and
Microphone",
                            "Local Disk"};
        phaseChoices = new JComboBox(phases);

        phaseChoices.setSelectedIndex(START_INDEX);
        DestIPText.setText("");
        TransPortText.setText("");

        ConnectButton = new JButton("Connect");
        ConnectButton.setActionCommand("Connect");
        ConnectButton.addActionListener(this);
        DisconnectButton = new JButton("Disconnect");
```

```
        ConnectButton.setActionCommand("Disconnect");
            DisconnectButton.addActionListener(this);
        //Add a border around the select panel.
        selectPanel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Please specific Source
Location"),
            BorderFactory.createEmptyBorder(5,5,5,5)));

        //Add a border around the display panel.
        displayPanel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Please specific
Destination's IP"),
            BorderFactory.createEmptyBorder(5,5,5,5)));
        displayPanel.setSize(new Dimension(200, 40));


TransmitPortPanel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Please specific
Transmit Port"),
            BorderFactory.createEmptyBorder(5,5,5,5)));

        ReceiveAddPanel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Please specific
Receive's IP"),
            BorderFactory.createEmptyBorder(5,5,5,5)));


ReceivePortPanel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Please specific Receive
Port"),
            BorderFactory.createEmptyBorder(5,5,5,5)));
        displayPanel.add(DestIPText);
        selectPanel.add(phaseChoices);
        TransmitPortPanel.add(TransPortText);
        ReceivePortPanel.add(RePortText);
        ReceiveAddPanel.add(ReceiveAddText);

        ConnectPanel.add(ConnectButton);
        ConnectPanel.add(DisconnectButton);
        //Listen to events from the combo box.
        phaseChoices.addActionListener(this);
    }

    public void actionPerformed(ActionEvent event) {
        Object source = event.getSource();
        if (source instanceof JComboBox){
            System.err.println("JcomboBox");
            JComboBox cb = (JComboBox)event.getSource();
            String select = (String)cb.getSelectedItem();
            if (select == "Local Disk"){
                  CHOICE = 1;
                  FileChooserDemo Chooser;
                  Chooser = new FileChooserDemo();
                  Chooser.active();
                  System.err.println(Chooser.mediaFile);
            }else if (select == "Webcam"){
                  CHOICE = 2;
            }else if (select == "Microphone"){
                  CHOICE = 3;
            }else if (select == "Webcam and Microphone"){
                  CHOICE = 4;
            }
        }
        if (source instanceof JButton){
            System.err.println("JButton111111");
```

```
            System.err.println(FileChooserDemo.mediaFile);
            String command = event.getActionCommand();
                  System.err.println(" 60 seconds...");
                  Address = DestIPText.getText();
                  System.err.println(Address);
                  Receiveport = RePortText.getText();
                  Transmitport = TransPortText.getText();
                  ReceiveAdd = ReceiveAddText.getText();
                  System.err.println(Transmitport);
                  Transmit Tran = new Transmit();
                  Tran.transmit();
                  System.err.println(" I'm here");
        }
    }
    /**
     * Create the GUI and show it.  For thread safety,
     * this method should be invoked from the
     * event-dispatching thread.
     */
    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);
        //Create a new instance of LunarPhases.
        LunarPhases phases = new LunarPhases();
        //Create and set up the window.
        JFrame lunarPhasesFrame = new JFrame("Video Phones");
        lunarPhasesFrame.setSize(new Dimension(1200, 40));

lunarPhasesFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        lunarPhasesFrame.setContentPane(phases.mainPanel);
        //Display the window.
        lunarPhasesFrame.pack();
        lunarPhasesFrame.setVisible(true);
    }
    public static void main(String[] args) {
        //Schedule a job for the event-dispatching thread:
        //creating and showing this application's GUI.
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

โปรแกรม FileChooserDemo

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.filechooser.*;
public class FileChooserDemo extends JPanel
                             implements ActionListener {
    static private final String newline = "\n";
    JButton openButton, saveButton;
    JTextArea log;
    JFileChooser fc;
    public static String  mediaFile;
    public static JFrame frame = new JFrame("FileChooserDemo");
    public FileChooserDemo() {
        super(new BorderLayout());
  //Create the log first, because the action listeners
  //need to refer to it.
        log = new JTextArea(5,20);
        log.setMargin(new Insets(5,5,5,5));
        log.setEditable(false);
        JScrollPane logScrollPane = new JScrollPane(log);
  //Create a file chooser
        fc = new JFileChooser();
        openButton = new JButton("Open a File...");
        openButton.addActionListener(this);
        JPanel buttonPanel = new JPanel(); //use FlowLayout
        buttonPanel.add(openButton);
  //Add the buttons and the log to this panel.
        add(buttonPanel, BorderLayout.PAGE_START);
        add(logScrollPane, BorderLayout.CENTER);
    }
    public void actionPerformed(ActionEvent e) {
  //Handle open button action.
        if (e.getSource() == openButton) {
            int returnVal = fc.showOpenDialog(FileChooserDemo.this);
            if (returnVal == JFileChooser.APPROVE_OPTION) {
                File file = fc.getSelectedFile();
                 mediaFile = file.getPath();
    //This is where a real application would open the file.
                log.append("Opening: " + file.getName() + "." +
newline);

    //disappear after open
                frame.setVisible(false);
    //disappear after open
            } else {
                log.append("Open command cancelled by user." +
newline);
            }
            log.setCaretPosition(log.getDocument().getLength());
  //Handle save button action.
        }
    }
 /** Returns an ImageIcon, or null if the path was invalid. */
    protected static ImageIcon createImageIcon(String path) {
        java.net.URL imgURL = FileChooserDemo.class.getResource(path);
        if (imgURL != null) {
            return new ImageIcon(imgURL);
        } else {
            System.err.println("Couldn't find file: " + path);
```

```
            return null;
        }
    }

 /**
  *Create the GUI and show it.  For thread safety,
  *this method should be invoked from the
  *event-dispatching thread.
  */
    private static void createAndShowGUI() {
  //Make sure we have nice window decorations.
  //JFrame.setDefaultLookAndFeelDecorated(true);
  //JDialog.setDefaultLookAndFeelDecorated(true);
  //Create and set up the window.
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  //Create and set up the content pane.
        JComponent newContentPane = new FileChooserDemo();
        newContentPane.setOpaque(true); //content panes must be
opaque
        frame.setContentPane(newContentPane);
  //Display the window.
        frame.pack();
        frame.setVisible(true);
    }
     public static void active(){
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

โปรแกรม Transmit.java

```
import java.awt.*;
import java.io.*;
import java.net.InetAddress;
import javax.media.*;
import javax.media.protocol.*;
import javax.media.protocol.DataSource;
import javax.media.format.*;
import javax.media.control.TrackControl;
import javax.media.control.QualityControl;
import javax.media.rtp.*;
import javax.media.rtp.rtcp.*;
import com.sun.media.rtp.*;
import java.util.Vector;
public class Transmit extends Thread {

    // Input MediaLocator
    // Can be a file or http or capture source
    private MediaLocator locator;
    private String ipAddress;
    private int portBase;
      public static int NUMport;
    private Processor mediaprocessor = null;
    private RTPManager rtpMgrs[];
    private DataSource dataOutput = null;
    private static final Format[] FORMATS = null;
    private static final ContentDescriptor CONTENT_DESCRIPTOR =
        new ContentDescriptor(ContentDescriptor.RAW_RTP);

      public Transmit(){
      }
      public  Transmit(
                  String ipAddress,
                  int port,
                  Format format,
                  CaptureDeviceInfo di) {

      this.locator =  di.getLocator();
      System.err.println(di.getLocator());
      this.ipAddress = ipAddress;
      this.portBase = port;
    }

    public Transmit(MediaLocator locator,
                  String ipAddress,
                  String pb,
                  Format format) {

      this.locator = locator;
      this.ipAddress = ipAddress;
      Integer integer = Integer.valueOf(pb);
      if (integer != null)
          this.portBase = integer.intValue();
    }
    public synchronized String startt() {
      String result;
      // Create a processor for the specified media locator
      // and program it to output JPEG/RTP
      result = createProcessor();
      if (result != null)
          return result;
      // Create an RTP session to transmit the output of the
      // processor to the specified IP address and port no.
```

```
        result = createTransmitter();
        if (result != null) {
            mediaprocessor.close();
            mediaprocessor = null;
            return result;
        }
        // Start the transmission
        mediaprocessor.start();

        return null;
    }
    /**
     * Stops the transmission if already started
     */
    public void stopp() {
        synchronized (this) {
            if (mediaprocessor != null) {
                mediaprocessor.stop();
                mediaprocessor.close();
                mediaprocessor = null;
                for (int i = 0; i < rtpMgrs.length; i++) {
                    rtpMgrs[i].removeTargets( "Session ended.");
                    rtpMgrs[i].dispose();
                }
            }
        }
    }
    private String createProcessor() {
        if (locator == null)
            return "Locator is null";

        DataSource ds;
        DataSource clone;
        try {
            ds = javax.media.Manager.createDataSource(locator);
        } catch (Exception e) {
            return "Couldn't create DataSource";
        }

        try {
                mediaprocessor = Manager.createRealizedProcessor(
                new ProcessorModel(ds, FORMATS, CONTENT_DESCRIPTOR));
    } catch (NoProcessorException npe) {
            return "Couldn't create processor";
        } catch (IOException ioe) {
            return "IOException creating processor";
        } catch (CannotRealizeException e){
                return "CannotRealizeException";
        }


        // Get the tracks from the processor
        TrackControl [] tracks = mediaprocessor.getTrackControls();

        // Do we have atleast one track?
        if (tracks == null || tracks.length < 1)
            return "Couldn't find tracks in processor";

        // Set the output content descriptor to RAW_RTP
        // This will limit the supported formats reported from
        // Track.getSupportedFormats to only valid RTP formats.
        ContentDescriptor cd = new
ContentDescriptor(ContentDescriptor.RAW_RTP);
        mediaprocessor.setContentDescriptor(cd);
        Format supported[];
```

```
        Format chosen;
        boolean atLeastOneTrack = false;
        // Program the tracks.
        for (int i = 0; i < tracks.length; i++) {
            Format format = tracks[i].getFormat();
            if (tracks[i].isEnabled()) {
              supported = tracks[i].getSupportedFormats();
              if (supported.length > 0) {
                  if (supported[1] instanceof VideoFormat) {
                    // For video formats, we should double check the
                    // sizes since not all formats work in all sizes.
                    chosen = supported[1];
                  } else
                    chosen = supported[6];
                  tracks[i].setFormat(chosen);
                  System.err.println("Track " + i + " is set to
transmit as:");
                  System.err.println("  " + chosen);
                  atLeastOneTrack = true;
              } else
                  tracks[i].setEnabled(false);
            } else
              tracks[i].setEnabled(false);
        }

        if (!atLeastOneTrack)
            return "Couldn't set any of the tracks to a valid RTP
format";

        // Get the output data source of the processor
        dataOutput = mediaprocessor.getDataOutput();

        return null;
      }
      /**
       * Use the RTPManager API to create sessions for each media
       * track of the processor.
       */
      private String createTransmitter() {

        // Cheated.  Should have checked the type.
        PushBufferDataSource pbds = (PushBufferDataSource)dataOutput;
        PushBufferStream pbss[] = pbds.getStreams();

        rtpMgrs = new RTPManager[pbss.length];

        SessionAddress localAddr, destAddr;
        InetAddress ipAddr;
        SendStream sendStream;
        int port;
        SourceDescription srcDesList[];


        for (int i = 0; i < pbss.length; i++) {
            try {

              rtpMgrs[i] = RTPManager.newInstance();
              System.err.println( "pbss.length"+pbss.length);

              port = portBase + 2*i;
              ipAddr = InetAddress.getByName(ipAddress);

              localAddr = new
SessionAddress( InetAddress.getLocalHost(),
                                                    port);
```

```
            destAddr = new SessionAddress( ipAddr, port);

            rtpMgrs[i].initialize( localAddr);

            rtpMgrs[i].addTarget( destAddr);

            System.err.println( "Created RTP session: " + ipAddress +
" " + port);

            sendStream = rtpMgrs[i].createSendStream(dataOutput, i);

            sendStream.start();
          } catch (Exception  e) {
            return e.getMessage();
          }
        }
        return null;
      }
      /**************************************************************
       * Convenience methods to handle processor's state changes.

**************************************************************/

      private Integer stateLock = new Integer(0);
      private boolean failed = false;

      Integer getStateLock() {
        return stateLock;
      }

      void setFailed() {
        failed = true;
      }

      /**************************************************************
       * Inner Classes

**************************************************************/
      static class  Initialsound extends Thread{
        private String ipAddress;
        private Format format;
        private int port;
        public Initialsound(String ipAddress,
                                    int pb,
                                   Format format
                                  ){
        this.ipAddress = ipAddress;
        this.format = format;
        this.port = pb;

        }
        public void run(){
              CaptureDeviceInfo di = null;
              Vector deviceList =
CaptureDeviceManager.getDeviceList(new
                          AudioFormat(null));
                    if (deviceList.size() > 0)
                    di = (CaptureDeviceInfo)deviceList.firstElement();
                    else
             // Exit if we can't find a device that does linear,
            // 44100Hz, 16 bit,
             // stereo audio.
                    System.exit(-1);
              Transmit at = new  Transmit(
```

```
                                        ipAddress, port, format,di);

            String result = at.startt();
                  if (result != null) {
                  System.err.println("Error : " + result);
                  System.exit(0);
                  }
      }
    }
   static class Initialvideo extends Thread{
      private String ipAddress;
      private Format format;
      private int port;
      public Initialvideo(String ipAddress,
                                             int pb,
                                             Format format
                                        ){
            this.ipAddress = ipAddress;
            this.format = format;
            this.port = pb;

      }

      public void run(){
            CaptureDeviceInfo di = null;
            Vector deviceList =
CaptureDeviceManager.getDeviceList(new
                        VideoFormat(null));
                  if (deviceList.size() > 0)
                  di = (CaptureDeviceInfo)deviceList.firstElement();
                  else
             // Exit if we can't find a device that does linear,
            // 44100Hz, 16 bit,
             // stereo audio.
                  System.exit(-1);
            Transmit at = new  Transmit(
                                    ipAddress, port, format,di);
                  String result = at.startt();
                  if (result != null) {
                  System.err.println("Error : " + result);
                  System.exit(0);
                  }
      }
    }

    class StateListener implements ControllerListener {

      public void controllerUpdate(ControllerEvent ce) {

          // If there was an error during configure or
          // realize, the processor will be closed
          if (ce instanceof ControllerClosedEvent)
            setFailed();

          // All controller events, send a notification
          // to the waiting thread in waitForState method.
          if (ce instanceof ControllerEvent) {
            synchronized (getStateLock()) {
                getStateLock().notifyAll();
            }
          }
      }
    }
```

```
    public void run(){
      transmit();

    }
    public void transmit(){
      Format fmt = null;
      int i = 0;
      int j = 0;

      if(LunarPhases.CHOICE == 1){
            String b;
            b = "file:\\";
            String a;
            a = b.concat(FileChooserDemo.mediaFile);
            System.err.println(a);
            System.err.println(LunarPhases.Address);
            System.err.println(LunarPhases.Transmitport);
            Transmit at = new Transmit(new MediaLocator(a),
                                       LunarPhases.Address,
LunarPhases.Transmitport, fmt);
      // Start the transmission
      String result = at.startt();
      }
       if((LunarPhases.CHOICE == 2)|| (LunarPhases.CHOICE == 4)){
            j=j+1;
            Integer integer =
Integer.valueOf(LunarPhases.Transmitport);
       NUMport = integer.intValue();
           new Initialvideo(LunarPhases.Address,NUMport,fmt).start();
       }
      i =0;
      if((LunarPhases.CHOICE == 3)|| (LunarPhases.CHOICE == 4)){
            if (j==0){
                  Integer integer =
Integer.valueOf(LunarPhases.Transmitport);
            NUMport = integer.intValue();
                  Initialsound audio = new
Initialsound(LunarPhases.Address,NUMport,fmt);
                  audio.start();
            }else{
                  Initialsound audio = new
Initialsound(LunarPhases.Address,NUMport+2,fmt);
                  audio.start();
            }
      }
      // Receive
      Receive avReceive = new Receive();
      if (!avReceive.initialize()) {
          System.err.println("Failed to initialize the sessions.");
          System.exit(-1);
      }
      try {
          while (!avReceive.isDone())
            Thread.sleep(1000);
      } catch (Exception e) {}

      System.err.println("Start transmission for 60 seconds...");
      try {
          Thread.currentThread().sleep(60000);
      } catch (InterruptedException ie) {
      }
      System.err.println("...transmission ended.");
    }
}
```

โปรแกรม Receive.java

```
import java.io.*;
import java.awt.*;
import java.net.*;
import java.awt.event.*;
import java.util.Vector;
import javax.media.*;
import javax.media.rtp.*;
import javax.media.rtp.event.*;
import javax.media.rtp.rtcp.*;
import javax.media.protocol.*;
import javax.media.protocol.DataSource;
import javax.media.format.AudioFormat;
import javax.media.format.VideoFormat;
import javax.media.Format;
import javax.media.format.FormatChangeEvent;
import javax.media.control.BufferControl;
class Receive extends Thread implements ReceiveStreamListener,
SessionListener,
      ControllerListener
{
    String sessions[] = null;
    RTPManager mgrs[] = null;
    Vector playerWindows = null;

    boolean dataReceived = false;
    Object dataSync = new Object();
    int Recport;


    public Receive() {
      this.sessions = sessions;
    }

    protected boolean initialize() {

        try {
          InetAddress ipAddr;
          SessionAddress localAddr = new SessionAddress();
          SessionAddress destAddr;

          mgrs = new RTPManager[2];
          playerWindows = new Vector();

          SessionLabel session;
          System.err.println("already receive " );

          // change LunarPhases.Receiveport to int
            if (LunarPhases.Receiveport != null) {
            try {
                Integer integers =
Integer.valueOf(LunarPhases.Receiveport);
                if (integers != null)
                   Recport = integers.intValue();
            } catch (Throwable t) {
                throw new IllegalArgumentException();
            }
          } else
            throw new IllegalArgumentException();
            // change LunarPhases.Receiveport to int

          // Open the RTP sessions.
          for (int i = 0; i < /*sessions.length*/2; i++) {
```

```
            if(i==1){
                  Recport = Recport+2;
            }

            System.err.println("  - Open RTP session for: addr: " +
LunarPhases.ReceiveAdd + " port: " + /*session.port*/Recport + " ttl:
" /*+ session.ttl*/);

            mgrs[i] = (RTPManager) RTPManager.newInstance();
            mgrs[i].addSessionListener(this);
            mgrs[i].addReceiveStreamListener(this);

            ipAddr = InetAddress.getByName(LunarPhases.ReceiveAdd);



            if( ipAddr.isMulticastAddress()) {
                // local and remote address pairs are identical:
                localAddr= new SessionAddress( ipAddr,
                                               Recport);
                destAddr = new SessionAddress( ipAddr,
                                               Recport);
            } else {
                localAddr= new
SessionAddress( InetAddress.getLocalHost(),
                                                Recport);
                    destAddr = new SessionAddress( ipAddr,Recport);
            }

            mgrs[i].initialize( localAddr);

            // You can try out some other buffer size to see
            // if you can get better smoothness.
            BufferControl bc =
(BufferControl)mgrs[i].getControl("javax.media.control.BufferControl"
);
            if (bc != null)
                bc.setBufferLength(350);

            mgrs[i].addTarget(destAddr);
          }

        } catch (Exception e){
            System.err.println("Cannot create the RTP Session: " +
e.getMessage());
            return false;
        }

      // Wait for data to arrive before moving on.

      long then = System.currentTimeMillis();
      long waitingPeriod = 30000;  // wait for a maximum of 30 secs.

      try{
          synchronized (dataSync) {
            while (!dataReceived &&
                  System.currentTimeMillis() - then < waitingPeriod)
{
                if (!dataReceived)
                  System.err.println("  - Waiting for RTP data to
arrive...");
                dataSync.wait(1000);
            }
          }
```

```
    } catch (Exception e) {}

    if (!dataReceived) {
        System.err.println("No RTP data was received.");
        close();
        return false;
    }

      return true;
  }


  public boolean isDone() {
    return playerWindows.size() == 0;
  }


  /**
   * Close the players and the session managers.
   */

  protected void close() {

    for (int i = 0; i < playerWindows.size(); i++) {
        try {
          ((PlayerWindow)playerWindows.elementAt(i)).close();
        } catch (Exception e) {}
    }

    playerWindows.removeAllElements();

    // close the RTP session.
    for (int i = 0; i < mgrs.length; i++) {
        if (mgrs[i] != null) {
            mgrs[i].removeTargets( "Closing session from
Receive");
            mgrs[i].dispose();
          mgrs[i] = null;
        }
    }
  }


  PlayerWindow find(Player p) {
    for (int i = 0; i < playerWindows.size(); i++) {
        PlayerWindow pw = (PlayerWindow)playerWindows.elementAt(i);
        if (pw.player == p)
          return pw;
    }
    return null;
  }


  PlayerWindow find(ReceiveStream strm) {
    for (int i = 0; i < playerWindows.size(); i++) {
        PlayerWindow pw = (PlayerWindow)playerWindows.elementAt(i);
        if (pw.stream == strm)
          return pw;
    }
    return null;
  }


  /**
   * SessionListener.
```

```
     */

    public synchronized void update(SessionEvent evt) {
      if (evt instanceof NewParticipantEvent) {
         Participant p = ((NewParticipantEvent)evt).getParticipant();
          System.err.println("  - A new participant had just joined:
" + p.getCNAME());
      }
    }

    /**
     * ReceiveStreamListener
     */

    public synchronized void update( ReceiveStreamEvent evt) {

      RTPManager mgr = (RTPManager)evt.getSource();
      Participant participant = evt.getParticipant(); // could be
null.
      ReceiveStream stream = evt.getReceiveStream();  // could be
null.

      if (evt instanceof RemotePayloadChangeEvent) {

          System.err.println("  - Received an RTP
PayloadChangeEvent.");
          System.err.println("Sorry, cannot handle payload change.");
          System.exit(0);

      }

      else if (evt instanceof NewReceiveStreamEvent) {

          try {
            stream = ((NewReceiveStreamEvent)evt).getReceiveStream();
            DataSource ds = stream.getDataSource();

            // Find out the formats.
            RTPControl ctl =
(RTPControl)ds.getControl("javax.media.rtp.RTPControl");
            if (ctl != null){
                System.err.println("  - Recevied new RTP stream: " +
ctl.getFormat());
            } else
                System.err.println("  - Recevied new RTP stream");

            if (participant == null)
                System.err.println("      The sender of this stream
had yet to be identified.");
            else {
                System.err.println("      The stream comes from: " +
participant.getCNAME());
            }

            // create a player by passing datasource to the Media
Manager
            Player p = javax.media.Manager.createPlayer(ds);
            if (p == null)
                return;

            p.addControllerListener(this);
            p.realize();
            PlayerWindow pw = new PlayerWindow(p, stream);
            playerWindows.addElement(pw);
```

```
            // Notify intialize() that a new stream had arrived.
            synchronized (dataSync) {
                dataReceived = true;
                dataSync.notifyAll();
            }

        } catch (Exception e) {
            System.err.println("NewReceiveStreamEvent exception " +
e.getMessage());
            return;
        }

    }

    else if (evt instanceof StreamMappedEvent) {

        if (stream != null && stream.getDataSource() != null) {
            DataSource ds = stream.getDataSource();
            // Find out the formats.
            RTPControl ctl =
(RTPControl)ds.getControl("javax.media.rtp.RTPControl");
            System.err.println("  - The previously unidentified
stream ");
            if (ctl != null)
                System.err.println("      " + ctl.getFormat());
            System.err.println("      had now been identified as sent
by: " + participant.getCNAME());
        }
    }

    else if (evt instanceof ByeEvent) {

        System.err.println("  - Got \"bye\" from: " +
participant.getCNAME());
        PlayerWindow pw = find(stream);
        if (pw != null) {
            pw.close();
            playerWindows.removeElement(pw);
        }
    }

}


/**
 * ControllerListener for the Players.
 */

public synchronized void controllerUpdate(ControllerEvent ce) {

    Player p = (Player)ce.getSourceController();

    if (p == null)
        return;

    // Get this when the internal players are realized.
    if (ce instanceof RealizeCompleteEvent) {
        PlayerWindow pw = find(p);
        if (pw == null) {
            // Some strange happened.
            System.err.println("Internal error!");
            System.exit(-1);
        }
        pw.initialize();
        pw.setVisible(true);
```

```
        p.start();
    }

    if (ce instanceof ControllerErrorEvent) {
        p.removeControllerListener(this);
        PlayerWindow pw = find(p);
        if (pw != null) {
          pw.close();
          playerWindows.removeElement(pw);
        }
        System.err.println("AVReceive2 internal error: " + ce);
    }

}
/**
 * A utility class to parse the session addresses.
 */

class SessionLabel {

  public String addr = null;
  public int port;
  public int ttl = 1;

  SessionLabel(String session) throws IllegalArgumentException {

      int off;
      String portStr = null, ttlStr = null;

      if (session != null && session.length() > 0) {
        while (session.length() > 1 && session.charAt(0) == '/')
          session = session.substring(1);

        // Now see if there's a addr specified.
        off = session.indexOf('/');
        if (off == -1) {
          if (!session.equals(""))
            addr = session;
        } else {
          addr = session.substring(0, off);
          session = session.substring(off + 1);
          // Now see if there's a port specified
          off = session.indexOf('/');
          if (off == -1) {
            if (!session.equals(""))
              portStr = session;
          } else {
            portStr = session.substring(0, off);
            session = session.substring(off + 1);
            // Now see if there's a ttl specified
            off = session.indexOf('/');
            if (off == -1) {
              if (!session.equals(""))
                ttlStr = session;
            } else {
              ttlStr = session.substring(0, off);
            }
          }
        }
      }

      if (addr == null)
        throw new IllegalArgumentException();
      if (portStr != null) {
```

```
            try {
                Integer integer = Integer.valueOf(portStr);
                if (integer != null)
                  port = integer.intValue();
            } catch (Throwable t) {
                throw new IllegalArgumentException();
            }
          } else
            throw new IllegalArgumentException();

          if (ttlStr != null) {
            try {
                Integer integer = Integer.valueOf(ttlStr);
                if (integer != null)
                  ttl = integer.intValue();
            } catch (Throwable t) {
                throw new IllegalArgumentException();
            }
          }
        }
    }


    /**
     * GUI classes for the Player.
     */

    class PlayerWindow extends Frame {

        Player player;
        ReceiveStream stream;

        PlayerWindow(Player p, ReceiveStream strm) {
            player = p;
            stream = strm;
        }

        public void initialize() {
            add(new PlayerPanel(player));
        }

        public void close() {
            player.close();
            setVisible(false);
            dispose();
        }

        public void addNotify() {
            super.addNotify();
            pack();
        }
    }


    /**
     * GUI classes for the Player.
     */

    class PlayerPanel extends Panel {

        Component vc, cc;

        PlayerPanel(Player p) {
            setLayout(new BorderLayout());
            if ((vc = p.getVisualComponent()) != null)
```

```
            add("Center", vc);
        if ((cc = p.getControlPanelComponent()) != null)
            add("South", cc);
    }

    public Dimension getPreferredSize() {
        int w = 0, h = 0;
        if (vc != null) {
            Dimension size = vc.getPreferredSize();
            w = size.width;
            h = size.height;
        }
        if (cc != null) {
            Dimension size = cc.getPreferredSize();
            if (w == 0)
                w = size.width;
            h += size.height;
        }
        if (w < 160)
            w = 160;
        return new Dimension(w, h);
    }
  }

}
```

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยดีอันเนื่องมาจาก รศ.ดร. ไกรสิน ส่งวัฒนา อาจารย์ที่ปรึกษาที่คอยดูแลเอาใจใส่ในทุกๆด้านตั้งแต่เริ่มโครงการ หากไม่ได้รับความอนุเคราะห์ที่ดีเช่นนี้ ปริญญานิพนธ์ฉบับนี้คงไม่สามารถสำเร็จได้สักส่วนหนึ่ง คณะผู้จัดทำจึงขอขอบพระคุณเป็นอย่างยิ่ง

เอกสารอ้างอิง

1. STAMPER , DAVID A., กฤตศิลป์ บุรัมยากร แปล,ระบบเครือข่าย LAN LOCAL AREA NETWORKS ,เพียร์สัน เอ็ดดูเคชั่น และ ซีเอ็ดยูเคชั่น. 532 หน้า, 2546
2. ANDREW S. TANENBAUM, สัลยุทธ์ สว่างวรรณ แปล, ซีเอ็ดยูเคชั่น, 504 หน้า, 2546
3. ด.ร.วีรศักดิ์ ซึงถาวร, Java Programming Volume I, ซีเอ็ดยูเคชั่น ,2543
4. สุวัฒน์ – สุพจน์ ปุณณชัยยะ, ตัน ตัณฑ์สุทธิวงศ์, เปิดโลก TCP/IP และโปรโตคอลของอินเตอร์เน็ต 2545
5. อรพิน ประวัติบริสุทธิ์, คู่มือการเขียนโปรแกรมด้วยภาษา JAVA, โปรวิสชั่น , 332 หน้า , 2547
6. JMF 1.0 Programmer guide, http://java.sun.com/products/java-media/jmf/reference/docs
7. RFC 1889- RTP document: A Transport Protocol for Real-Time Applications ,www.facs.org /rfcs /rfc1889.html