การให้บริการโทรทัศน์ผ่านอินเตอร์เน็ต
IPTV

โดย
นายเขมวิทย์ บุญรัตน์
นายฉัตรชัย ขวัญฐิตินันท์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมสารสนเทศ
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2546

IPTV

BY

Mr.Khamwith Boonrat

Mr.Chatchai Kwanthitinun

A THESIS SUBBMITTED IN PARTIAL FULLFILMENT OF

THE REQUIREMENT FOR THE DEGREE OF

BACHELOR IN DEPARTMENT ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2003

หัวข้อปริญญานิพนธ์	การให้บริการโทรทัศน์ผ่านอินเตอร์เน็ต			
ชื่อนักศึกษา	นายเขมวิทย์	บุญรัตน์	รหัสนักศึกษา	43010042
	นายฉัตรชัย	ขวัญฐิตินันท์	รหัสนักศึกษา	43010078
อาจารย์ที่ปรึกษา	อาจารย์พนารัตน์	ระวิวรรณ		
	อาจารย์มนต์ชัย	แช่มช้อย		
ระดับการศึกษา	ปริญญาตรีวิศวกรรมศาสตรบัณฑิต			
	สาขาวิศวกรรมสารสนเทศ			
ภาควิชา	วิศวกรรมสารสนเทศ			
ปีการศึกษา	2546			

ปริญญานิพนธ์ฉบับนี้ได้รับความเห็นชอบจากอาจารย์ที่ปรึกษาเป็นที่เรียบร้อยแล้ว

.....

(อาจารย์พนารัตน์ ระวิวรรณ)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

.....

(อาจารย์มนต์ชัย แช่มช้อย)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	การให้บริการโทรทัศน์ผ่านอินเตอร์เน็ต		
ชื่อนักศึกษา	นายเขมวิทย์	บุญรัตน์	รหัสนักศึกษา 43010042
	นายฉัตรชัย	ขวัญฐิตินันท์	รหัสนักศึกษา 43010078
อาจารย์ที่ปรึกษา	อ.มนต์ชัย	แช่มช้อย	
	อ.พนารัตน์	ระวิวรรณ	
ระดับการศึกษา	ปริญญาตรีวิศวกรรมศาสตรบัณฑิต		
	สาขาวิศวกรรมสารสนเทศ		
ภาควิชา	วิศวกรรมสารสนเทศ		
ปีการศึกษา	2546		

บทคัดย่อ

โครงงานนี้เป็นการนำเอาเทคโนโลยีการรับส่งข้อมูลมัลติมีเดียแบบเรียลไทม์ มาสร้างโปรแกรมให้สามารถรับส่งข้อมูลนั้นได้ และสามารถดึงข้อมูลรายการทีวีจากการ์ดทีวีที่ต่อพ่วงกับเซิร์ฟเวอร์มาใช้เป็นข้อมูลที่จะส่งได้ ซึ่งคุณภาพของข้อมูลที่รับได้ จะขึ้นอยู่กับการทำงานของโปรแกรม อุปกรณ์ที่ใช้เป็นเซิร์ฟเวอร์ รวมทั้งจำนวนผู้รับชมในขณะนั้น แต่โดยรวมแล้วโปรแกรมสามารถส่งข้อมูลมัลติมีเดียแบบเรียลไทม์และรองรับผู้รับชมได้หลายคนพร้อมกัน

Thesis Title	IP TV		
Student	Mr.Khamwith	Boonrat	ID.43010042
	Mr.Chatchai	Kwanthitinun	ID.43010078
Advisor	Mr.Monchai	Chamchoy	
	Ms.Panarat	Rawiwan	
Graduate Level	Bachelor Degree of Information Engineering		
Department	Information Engineering		
Academic Year	2003		

Abstract

This project uses the real-time multimedia transmission technology to build the application which is able to transmit the information. Beside, this project even has the ability to retrieve the TV information from TV tuner card which is connected to server and uses the information as the transmitted information. The quality of receive information depends on the performance of the application, the server devices and even the present audiences.

กิตติกรรมประกาศ

ในการทำปริญญานิพนธ์ฉบับนี้ไม่อาจสำเร็จไปได้เลย หากไม่ได้รับความช่วยเหลือจากอาจารย์ที่ปรึกษาปริญญานิพนธ์ รวมทั้งอาจารย์ท่านอื่น ๆ ที่ได้ให้ความรู้ ตั้งแต่พ่อแม่ซึ่งเป็นครูคนแรก ครูทุกๆท่านในชั้นอนุบาล ประถมศึกษา อาจารย์ทุกท่านในชั้นมัธยมศึกษา อาจารย์ทุกท่านในสถาบันเทคโนโลยี พระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง แห่งนี้ ขอขอบคุณ www.google.com ที่ใช้เป็นเครื่องมือในการสืบค้นหาข้อมูลได้เป็นอย่างมาก ขอบคุณสถาบันแห่งนี้ ที่ได้ให้ใช้สถานที่ในการทำ และศึกษาหาความรู้ ขอบคุณทุกคนในครอบครัว เพื่อน ๆ พี่ ๆ น้องๆ ทุกคน ที่ได้ให้ความช่วยเหลือในทุก ๆ ด้าน ขอบคุณทุก ๆ คำต่าและคำชมที่ได้รับ ขอบคุณทุก ๆ สิ่งทุก ๆ อย่างในโลกนี้ที่ได้ประสบพบเจอมาในชีวิตและทำให้เป็นตัวเราอยู่ในขณะนี้ สุดท้ายนี้ขอขอบคุณหัวใจที่ยังเต้นอยู่ที่ทำให้เรามีลมหายใจถึงขณะนี้ จนสามารถทำปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดี

คณะผู้จัดทำ

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญตาราง

บทที่1
บทนำ

1.1 แนวคิดและที่มา

ในปัจจุบันนี้ความต้องการในการติดต่อสื่อสารได้เพิ่มมากขึ้น ซึ่งทำให้ระบบสื่อสารแบบอินเตอร์เน็ต มีความสะดวกสบายมากขึ้น เพราะค่าบริการอินเตอร์เน็ตมีราคาถูกลง มีการกระจายการให้บริการไปในส่วนท้องถิ่นมากขึ้น รวมทั้งสามารถติดต่อสื่อสารได้กว้างไกลทั่วโลก จึงเป็นที่มาของแนวคิดที่ว่า เราอาจจะสามารถนำมาประยุกต์ในการรับชมรายการทีวีต่าง ๆ ได้ แม้จะอยู่ในต่างประเทศที่ห่างไกล ซึ่งอาจเป็นข่าวสารต่าง ๆ ภายในประเทศที่ไม่สามารถใช้โทรทัศน์รับชมได้ในต่างประเทศ ทั้งยังอาจสามารถนำมาประยุกต์ปรับปรุงเพื่อพัฒนาไปสู่การถ่ายทอดสดรายการผ่านระบบอินเตอร์เน็ต และอาจเพิ่มเติมการบันทึกภาพรายการทีวีต่าง ๆ จัดเก็บเป็นไฟล์ โดยที่การพัฒนาการถ่ายทอดรายการทีวีผ่านระบบอินเตอร์เน็ต ก่อให้เกิดประโยชน์ดังนี้

- สามารถรับชมรายการทีวีต่าง ๆ ได้ในพื้นที่ที่ห่างไกล เช่น ในต่างประเทศ
- โปรแกรมที่ใช้ติดต่อรับและส่งรายการทีวีนั้น เป็นโปรแกรมที่พัฒนาขึ้นมาเอง จึงสามารถปรับปรุงเปลี่ยนแปลงให้เหมาะสมกับความสามารถในการใช้งานในรูปแบบต่าง ๆ รวมทั้งการพัฒนาโปรแกรมให้ทันกับเทคโนโลยีในปัจจุบันสามารถทำได้โดยง่าย
- ประหยัดค่าใช้จ่ายในการรับชมรายการทีวี เนื่องจากใช้การติดต่อกับอินเตอร์เน็ต และโปรแกรมที่พัฒนาขึ้นเองทำให้ไม่ต้องเสียค่าลิขสิทธิ์โปรแกรม

1.2 วัตถุประสงค์ของโครงงาน

1. เข้าใจระบบรับส่งข้อมูลแบบสตรีมมิ่ง
2. เข้าใจระบบการรับสิ่งข้อมูลมัลติมีเดีย (Multimedia) แบบเรียลไทม์ (Real-Time)
3. ศึกษาและพัฒนาการรับส่งข้อมูลในการทำงานของ IPTV
4. ศึกษาการจับสัญญาณภาพ (Capture) จากอุปกรณ์ต่อพ่วง และสามารถนำข้อมูลนั้นมาใช้ประโยชน์ในการติดต่อสื่อสารได้
5. วิเคราะห์ระบบเชิงวัตถุ (Object-Oriented) และเขียนโปรแกรมโดยใช้ภาษาจาวา

1.3 ขอบเขตของโครงงาน

1. พัฒนาโปรแกรมโดยใช้ภาษาจาวา
2. การถ่ายทอดรายการทีวีสามารถรองรับการใช้งานได้หลายคนผ่านเครือข่ายอินเตอร์เน็ต โดยในโครงงานนี้จะทำการจำลองเครื่องเซิร์ฟเวอร์ (Server) 1 เครื่องและเครื่องไคลเอ้นท์ (Client) 1 เครื่อง
3. โปรแกรมฝั่งเซิร์ฟเวอร์สามารถส่งข้อมูลมัลติมีเดียเป็นแบบเรียลไทม์ผ่านระบบอินเตอร์เน็ตได้
4. โปรแกรมฝั่งไคลเอ้นท์สามารถรับข้อมูลมัลติมีเดียแบบเรียลไทม์ผ่านระบบอินเตอร์เน็ตได้
5. โปรแกรมฝั่งเซิร์ฟเวอร์สามารถเลือกรับสัญญาณข้อมูลจากการ์ดทีวีเพื่อใช้ในการส่งข้อมูลได้
6. โปรแกรมฝั่งเซิร์ฟเวอร์สามารถเลือกไฟล์ข้อมูลเพื่อใช้ในการส่งข้อมูลได้

1.4 ขั้นตอนการดำเนินโครงงาน

1. ทำการศึกษาระบบรับส่งข้อมูลมัลติมีเดียผ่านเครือข่ายอินเตอร์เน็ต
2. ศึกษาการรับส่งข้อมูลแบบเรียลไทม์
3. ศึกษาการเขียนโปรแกรมด้วยภาษาจาวา
4. รวบรวมข้อมูลแนวทางการทำงานและขอบเขตของโครงงาน
5. ออกแบบการทำงานในส่วนต่าง ๆ ของโปรแกรม
6. ออกแบบอินเตอร์เฟสที่ใช้ติดต่อกับผู้ใช้งาน
7. พัฒนาโปรแกรมและปรับปรุงแก้ไขส่วนต่าง ๆ ให้เหมาะสมกับการใช้งาน
8. ติดตั้งและทดสอบโปรแกรมที่พัฒนาผ่านเครือข่ายอินเตอร์เน็ต
9. สรุปผลการทำงาน ปัญหาที่พบและหาแนวทางการแก้ไข

1.5 ผลที่คาดว่าจะได้รับ

1. เข้าใจระบบการรับส่งโปรโตคอล RTP ผ่านเครือข่ายอินเตอร์เน็ต
2. เข้าใจการออกแบบระบบเชิงวัตถุ และการพัฒนาโปรแกรมด้วยภาษาจาวา
3. เข้าใจการทำงานการดึงข้อมูลจากอุปกรณ์ต่อพ่วงมาใช้ และสามารถส่งผ่านข้อมูลนั้นผ่านระบบเครือข่ายอินเตอร์เน็ตได้
4. สามารถพัฒนาระบบรับส่งไฟล์แบบเรียลไทม์ผ่านเครือข่ายอินเตอร์เน็ตได้

บทที่2
การสื่อสารแบบมัลติมีเดีย

2.1 การสตรีมมิ่งสื่อ (Streaming Media)

ข้อมูลทุกอย่างที่เปลี่ยนแปลงตามเวลาจะสามารถแสดงในรูปแบบสื่อเชิงเวลา (Time-based media) ได้ เช่น ออดิโอคลิป (Audio clips), อนุกรม MIDI (MIDI sequence), มูวี่คลิป (movie clips) และ ภาพเคลื่อนไหว (animation) คือรูปแบบเดียวกับสื่อเชิงเวลา ตัวอย่างของข้อมูลสื่อ (media data) สามารถหาได้จากหลายที่ เช่น โลคอล (local) หรือ ไฟล์เครือข่าย (network files), กล้องถ่ายภาพ และไมโครโฟน เป็นต้น

รูปที่ 2.1 รูปแบบการดำเนินการของมิเดีย (Media processing model)

ลักษณะสำคัญของสื่อเชิงเวลา คือมันต้องการส่งและดำเนินการตามเวลา เมื่อข้อมูลสื่อเริ่มเคลื่อนที่ จะมีกำหนดเวลาเส้นตายที่ข้อมูลจะต้องถึงในทั้งผู้ส่งและผู้รับข้อมูล สำหรับเหตุผลนี้สื่อเชิงเวลาจะถูกนำไปเกี่ยวข้องกับการสร้างทางไหลของสื่ออยู่บ่อย ๆ มันถูกส่งในทางไหลแบบคงที่ (Steady Stream) ซึ่งได้รับและดำเนินการในเวลาเดียวกัน สำหรับตัวอย่างคือ เมื่อภาพเริ่มเล่น ถ้าข้อมูลสื่อไม่สามารถถูกส่งอย่างรวดเร็วพอ มันอาจจะหยุดเล็กน้อยและเสียเวลา (delay) ในการฉายภาพ ในแบบอื่น ถ้าข้อมูลไม่สามารถรับและดำเนินการได้รวดเร็วพอ ภาพอาจจะกระโดดข้ามไปและข้อมูลที่หายไปหรือเฟรม (frame) นั้นก็จะถูกทิ้งเพื่อที่จะรักษาระดับการเล่นที่ถูกต้อง

2.1.1 ประเภทส่วนประกอบของสื่อ (Content Type)

ข้อมูลสื่อจะถูกเก็บอยู่ในรูปประเภทของมัน ตัวอย่างเช่น QuickTime, MPEG และWAV ซึ่งจริง ๆ แล้วประเภทของมันมีความหมายเหมือนกับประเภทไฟล์ (file type) แต่ประเภทส่วนประกอบถูกใช้เพราะข้อมูลสื่อส่วนมากจะได้มาจากแหล่งอื่น ๆ มากกว่าได้จากไฟล์ที่มีอยู่แล้ว

2.1.2 มีเดียสตรีม (Media Streams)

มีเดียสตรีมคือ ข้อมูลสื่อที่ได้จากไฟล์ที่มีอยู่แล้วหรือการจับข้อมูล (Capture) จากกล้องถ่ายภาพหรือได้จากไมโครโฟน มีเดียสตรีมส่วนมากจะถูกบรรจุในช่อง (Channel) ที่รวมข้อมูลหลาย ๆ อย่างซึ่งเรียกว่าแทรค (Track) ตัวอย่างเช่น ไฟล์ประเภท Quicktime อาจจะถูกบรรจุในทั้งแทรคเสียง (audio track) และแทรคภาพ (video track) มีเดียสตรีมที่บรรจุหลากหลายแทรคอาจถูกเรียกว่ามีเดียสตรีมซับซ้อน (multiplexed หรือ complex media stream) และการดีมัลติเพล็กซิ่ง (Demultiplexing) ซึ่งคือการแยกแทรคแต่ละแทรคออกมาจากมีเดียสตรีมซับซ้อน

ชนิดของแทรคจะแสดงชนิดของข้อมูลที่ถูกบรรจุอยู่ เช่นภาพหรือเสียง รูปแบบของแทรคกำหนดว่าข้อมูลสำหรับแทรคนั้นจะถูกสร้างได้อย่างไร

มีเดียสตรีมสามารถแสดงได้โดยโลคอลของมันหรือโปรโตคอล (Protocol) ที่ใช้รับมัน ตัวอย่างเช่น URL อาจจะใช้บอกที่ตั้งของไฟล์ประเภท QuickTime ที่อยู่บนโลคอลหรือระบบห่างไกล (remote system) ถ้าไฟล์คือโลคอล มันจะสามารถถูกเข้าถึงได้โดยผ่านโปรโตคอล FILE ถ้ามันอยู่บนเวปเซิร์ฟเวอร์ (Web Server) ไฟล์จะสามารถถูกเข้าถึงโดยผ่านโปรโตคอล HTTP ได้ โดยจะมี ตัวแสดงที่ตั้งของมีเดีย (media locator) หาทางเพื่อแสดงที่ตั้งของมีเดียสตรีมเมื่อไม่ได้ใช้ URL

มีเดียสตรีมสามารถแบ่งออกเป็นประเภท ๆ ได้ตามวิธีที่ข้อมูลถูกส่งมา

- PULL การถ่ายทอดข้อมูลเริ่มและควบคุมจากฝั่งลูกข่าย (client) ตัวอย่างเช่น Hypertext Transfer Protocol (HTTP) และ FILE
- PUSH เซิร์ฟเวอร์เริ่มการถ่ายทอดข้อมูล และควบคุมการไหลของข้อมูล ตัวอย่างเช่น Real-time Transport Protocol (RTP) คือโปรโตคอล PUSH ที่ใช้สำหรับการสร้างทางไหลของสื่อ ในทำนองเดียวกัน SGI Media Base protocol คือโปรโตคอล PUSH ที่ใช้สำหรับวีดีโอตามต้องการ (video-on-demand (VOD))

2.1.3 รูปแบบสื่อโดยทั่วไป (Common Media Formats)

ตารางข้างล่างแสดงลักษณะบางอย่างของรูปแบบสื่อโดยทั่วไป เมื่อการเลือกรูปแบบ มันสำคัญที่การเข้าถึงลักษณะของรูปแบบ และความคาดหวังของผู้ฟัง ตัวอย่างเช่น ถ้าคุณกำลังส่งข้อมูลสื่อ (media content) ผ่านทางเวป (web) คุณต้องให้ความสนใจกับความต้องการแบนด์วิธ (bandwidth) เป็นพิเศษ

หัวข้อ CPU Requirements แสดงลักษณะที่จำเป็นสำหรับการนำเสนอที่ดีที่สุดของแบบละเอียด หัวข้อ Bandwidth Requirement แสดงลักษณะความเร็วที่จำเป็นในการส่งหรือรับข้อมูลที่รวดเร็วเพียงพอกับการแสดงที่ดีที่สุด

ตาราง 2.1 รูปแบบภาพโดยทั่วไป (Common video formats)

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

ตาราง 2.2 รูปแบบเสียงโดยทั่วไป (Common audio formats)

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

รูปแบบบางอย่างถูกออกแบบมากับแอพพลิเคชั่น (Application) เฉพาะตัวและความต้องการโดยเฉพาะของมันเอง รูปแบบลักษณะสูง (high-quality) และรูปแบบแบนด์วิธสูง (high-bandwidth) ส่วนมากจะใช้ในซีดีรอมหรือแอพพลิเคชั่นเก็บที่ตั้ง (local storage application) โดยทั่วไป H.261 และ H.263 จะถูกใช้สำหรับแอพพลิเคชั่นที่รวมภาพ (video conferencing application) และถูกพัฒนาขึ้นสำหรับภาพซึ่งไม่มีการเคลื่อนไหวมากนัก ในทำนองเดียวกัน G.723 ก็ถูกใช้เป็นตัวอย่าง

2.1.4 การแสดงผล (Media Presentation)

สื่อเชิงเวลาส่วนใหญ่คือข้อมูลเสียงหรือภาพ ซึ่งสามารถแสดงผ่านอุปกรณ์แสดงผล (Output Device) เช่นลำโพงหรือจอภาพ ตัวอย่างอุปกรณ์ส่วนมากมีจุดหมายร่วมกันสำหรับข้อมูลสื่อ เอาท์พุตของมีเดียสตรีม สามารถส่งไปจุดหมายอื่นได้ ตัวอย่างเช่น เซฟ (save) บนไฟล์หรือถ่ายทอดผ่านเครือข่าย จุดหมายที่ส่งข้อมูลสื่อออกไปบางครั้งเรียกว่าดาต้าซิงค์ (Data sink)

2.1.4.1 การควบคุมการแสดงผล (Presentation Controls)

ขณะที่มีเดียสตรีมเริ่มต้นแสดง การควบคุมการแสดงแบบ VCR ถูกใช้โดยเงื่อนไขว่าผู้ใช้จะสามารถควบคุมการเล่นได้ ตัวอย่างเช่น ที่ตัวแสดงภาพ (Movie Player) อาจจะเสนอปุ่มสำหรับหยุด, เริ่ม, เดินหน้าและถอยหลังภาพได้

2.1.4.2 Latency

ในหลาย ๆ กรณี เฉพาะเมื่อการแสดงมีเดียสตรีมที่อยู่บนเครือข่าย การแสดงนั้นจะไม่สามารถเริ่มได้โดยทันที เวลาที่มันใช้ไปก่อนเริ่มแสดงนั้นเรียกว่าการเริ่มแฝง (Start Latency) ผู้ใช้อาจจะรู้เรื่องนี้ว่าเหมือนกับการรอช่วงเวลาที่คลิกปุ่มเริ่มต้นกับเวลาที่เริ่มเล่นจริง ๆ

การแสดงผลแบบมัลติมีเดีย (Multimedia) ส่วนมากจะรวมสื่อเชิงเวลาหลาย ๆ ชนิดในการแสดงในเวลาเดียวกัน ตัวอย่างเช่น การแสดงเสียงดนตรีอาจจะเล่นระหว่างการแสดงภาพสไลด์ หรือตัวอักษรเคลื่อนไหว (Animated Text) อาจจะแสดงในเวลาเดียวกับออดิโอหรือวีดีโอคลิป เมื่อการแสดงมีเดียสตรีมแบบมัลติเปิล (mutiple media stream) แสดงในเวลาเดียวกัน มันเป็นจุดสำคัญที่จะนำไปสู่การเริ่มแฝงของแต่ละสตรีม อีกอย่างหนึ่งการเล่นสตรีมที่แตกต่างอาจจะเริ่ม จริง ๆ ที่เวลาต่างกัน

2.1.4.3 คุณสมบัติการแสดงผล (Presentation Quality)

คุณสมบัติของการแสดงผลของมีเดียสตรีมต้องอาศัยปัจจัยหลาย ๆ อย่าง เช่น

- ใช้รูปแบบการบีบอัด (compression)
- ความสามารถการทำงานของระบบเล่น (Playback)
- แบนด์วิธที่ใช้ได้

คุณสมบัติที่สูงขึ้นเรื่อย ๆ ไฟล์ขนาดใหญ่ขึ้นและกำลังการดำเนินการ (Processing Power) และแบนด์วิธที่ต้องการมากขึ้น แบนด์วิธจะถูกใช้แสดงผลเหมือนกับตัวเลขของบิท (Bit) ที่ถ่ายทอดในเวลาปัจจุบัน เพื่อที่จะทำการแสดงผลวีดีโอคุณสมบัติสูง (high-quality video) จำนวนของเฟรมในแต่ละช่วงเวลาหรืออัตราของเฟรม (frame rate) ควร

จะสูงเท่าที่จะเป็นไปได้ จริง ๆ แล้ววีดีโอที่อัตราของเฟรม = 30 เฟรมต่อวินาทีจะทำให้แยกไม่ออกว่าเป็นทีวีบอร์ดคาสท์ (TV broadcasts) หรือวีดีโอเทป

2.1.5 การดำเนินการของมีเดีย (Media Processing)

ตัวอย่างข้อมูลในมีเดียสตรีมบิดเบือนก่อนที่มันจะถูกแสดงให้ผู้ใช้ ชุดการทำงานที่เกิดขึ้นก่อนการแสดงผลคือ มีขั้นตอนดังนี้

1. ถ้าสตรีมถูกมัลติเพล็ก (multiplexed) แทรคแต่ละตัวจะถูกแยกออกมาโดยเฉพาะ
2. ถ้าแทรคถูกบีบอัดเฉพาะตัว มันจะถูกถอดรหัส (decode)
3. ถ้าจำเป็น แทรคจะถูกเปลี่ยนสภาพ (convert) ไปรูปแบบที่ต่างกัน
4. Effect filter จะถูกใช้ไปแทรคที่ถอดรหัสแล้ว (ถ้าออกแบบไว้)

แทรคจะถูกส่งไปอุปกรณ์แสดงผลที่เหมาะสม ถ้ามีเดียสตรีมถูกเก็บแทนที่จะส่งไปนำเสนอที่อุปกรณ์แสดงผล แบ่งขั้นตอนการดำเนินการได้ออกเป็นขั้น ๆ ตัวอย่างเช่น ถ้าเราต้องการจับภาพ (capture) ภาพและเสียงจากกล้องถ่ายวีดีโอ, จัดการข้อมูลและเซฟมันไปที่ไฟล์ มีขั้นตอนดังนี้

1. แทรคภาพและเสียงถูกจับภาพ
2. เอฟเฟคฟิลเตอร์ (Effect filter) ถูกประยุกต์มาใช้งานในแถวของแทรค ถ้าออกแบบไว้
3. แทรคแต่ละตัวถูกถอดรหัส เฉพาะตัว
4. แทรคที่บีบอัดแล้วจะถูกมัลติเพล็กไปมีเดียสตรีมแบบเดี่ยว (single media stream)
5. มีเดียสตรีมที่ถูกมัลติเพล็กแล้วถูกเซฟลงไฟล์

2.1.5.1 การดีมัลติเพล็กและมัลติเพล็ก (Demultiplexers and Multiplexers)

ตัวดีมัลติเพล็กจะแยกแทรคแต่ละแทรคของข้อมูลสื่อจากมีเดียสตรีมที่มัลติเพล็กแล้ว ส่วนตัวมัลติเพล็กจะตรงข้ามกัน คือจะนำแทรคที่แยกแล้วมารวมกันเป็นมัลติเพล็กมีเดียสตรีมแบบเดี่ยว (single multiplexed media stream)

2.1.5.2 การเข้ารหัส (Codecs)

ตัวเข้ารหัสจะทำการบีบอัดข้อมูลสื่อและขยายออก เมื่อแทรคถูกเข้ารหัส (encode) มันจะถูกเปลี่ยนสภาพไปเป็นรูปแบบที่ถูกบีบอัดที่พอดีสำหรับการเก็บหรือการถ่ายทอด เมื่อมันถูกถอดรหัส (decode) มันจะถูกเปลี่ยนสภาพเป็นรูปแบบที่ยังไม่บีบอัด ที่พอดีกับการแสดงผล

แต่ละตัวเข้ารหัสมีรูปแบบอินพุตที่แน่นอนตายตัว ซึ่งมันสามารถกำหนดรูปแบบเอาท์พุตที่มันสามารถสร้างได้ ในบางสถานการณ์ ชุดของตัวถอดรหัส อาจจะถูกใช้เพื่อเปลี่ยนสภาพ จากรูปแบบหนึ่ง ไปอีกรูปแบบหนึ่งได้

2.1.5.3 เอฟเฟคฟิลเตอร์ (Effect Filter)

เอฟเฟคฟิลเตอร์ จะทำการประยุกต์ข้อมูลแทรคในบางทาง เพื่อสร้าง เอฟเฟคพิเศษ (Special Effects) อย่างเช่นภาพเบลอหรือเสียงสะท้อน

2.1.5.4 ผลลัพธ์ (Renderer)

ผลลัพธ์คือสิ่งที่ได้ออกมาจากอุปกรณ์การแสดงผล สำหรับเสียง อุปกรณ์การแสดงผลคือ การ์ดเสียงของคอมพิวเตอร์ ที่ส่งเสียงออกทางลำโพง สำหรับภาพ อุปกรณ์การแสดงผลคือหน้าจอคอมพิวเตอร์

2.1.6 การจับสัญญาณข้อมูลมีเดีย (Media Capture)

สื่อเชิงเวลาสามารถจับข้อมูล (Capture) ได้จากแหล่งข้อมูลสด (live source) ตัวอย่างเช่น เสียงสามารถถูกจับข้อมูลได้จากไมโครโฟน หรือการ์ดจับข้อมูลวีดีโอ (video capture card) สามารถใช้เพื่อเอาภาพ มาจากกล้องถ่ายภาพ การจับข้อมูลสามารถเปรียบได้กับเป็นอินพุตของรูปแบบมาตราฐานการทำงานมีเดีย (standard media processing model)
อุปกรณ์การจับข้อมูลอาจจะส่งมีเดียสตรีมแบบมัลติเปิล ตัวอย่างเช่น กล้องถ่ายวีดีโออาจจะส่งได้ทั้งภาพและเสียง สตรีมนี้อาจจะถูกจับข้อมูลและแยกหรือรวมเป็นอันเดียว มัลติเพล็กสตรีมนั้นรวมทั้งแทรคเสียงและแทรคภาพ

2.1.6.1 อุปกรณ์จับสัญญาณข้อมูล (Capture Device)

เพื่อจะจับสัญญาณข้อมูลมีเดียเชิงเวลา เราต้องการอุปกรณ์เพิ่มเติม เช่น เพื่อจะจับสัญญาณข้อมูลเสียงจากแหล่งข้อมูลสด เราต้องการไมโครโฟนและการ์ดเสียงที่เหมาะสม โดยทั่วไปการจับข้อมูลทีวีบอร์ดคาสท์นั้นต้องการตัวปรับสัญญาณทีวีและการ์ดจับข้อมูลวีดีโอที่เหมาะสม ระบบส่วนใหญ่จะเตรียม Query Mechanism เพื่อหาว่าอุปกรณ์จับสัญญาณข้อมูลอันไหนจะใช้ได้

อุปกรณ์จับสัญญาณข้อมูลสามารถกำหนดได้ทั้งแบบ push หรือ pull sources เช่น กล้องเป็น pull source คือผู้ใช้สามารถควบคุมเมื่อจับข้อมูลภาพแล้ว และไมโครโฟนเป็น push source เพราะเป็นแหล่งข้อมูลสดที่จัดเตรียมสตรีมของภาพอย่างต่อเนื่อง

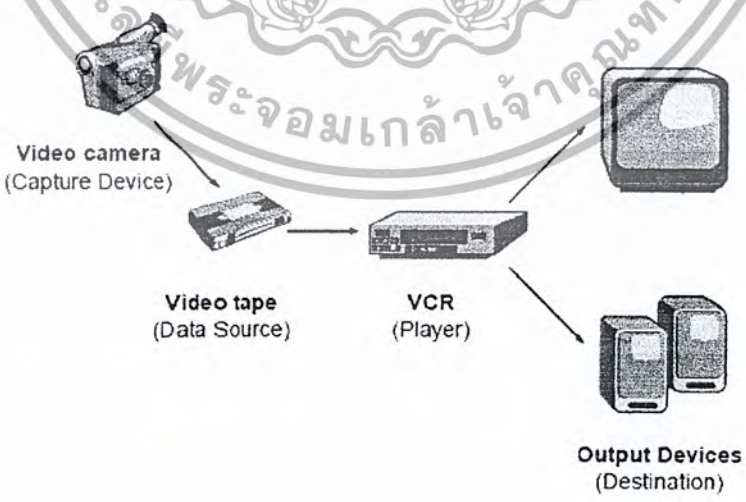
รูปแบบของมีเดียสตรีมที่ถูกจับข้อมูลมานั้น ขึ้นอยู่กับการจัดการ โดยอุปกรณ์จับข้อมูล บางอย่างจะจัดการเล็ก ๆ น้อย ๆ และส่งแถว (raw) โดยข้อมูลที่ยังไม่ได้บีบอัด อุปกรณ์อย่างอื่นอาจจะส่งข้อมูลในรูปแบบที่บีบอัดแล้ว

2.1.6.2 การควบคุมการจับสัญญาณข้อมูล (Capture Control)

ควบคุมให้ผู้ใช้สามารถจัดการโปรเซสการจับข้อมูล ตัวอย่างเช่น พาเนลควบคุมการจับ ข้อมูล (capture control panel) อาจจะให้ผู้ใช้ระบุอัตราข้อมูลและชนิดของการเข้ารหัสสำหรับ สตรีม ที่ถูกจับข้อมูลและเริ่มหรือหยุดโพสเซสการจับข้อมูล

2.2 จาวามีเดียเฟรมเวิร์ก (Java Media Framework : JMF)

การทำงานของ JMF มีหลักการทำงานคล้ายกับการทำงานของระบบวีดีโอ ตัวอย่างเช่น เมื่อเราต้องการดูหนังจากเครื่องเล่นวีดีโอ เราจะต้องใส่ม้วนเทปเข้าไปในเครื่องเล่นวีดีโอ จากนั้นเครื่องเล่นวีดีโอจะทำการอ่านข้อมูลจากเทป แล้วแสดงข้อมูลในรูปของสัญญาณภาพออกทางจอโทรทัศน์และสัญญาณเสียงออกทางลำโพงตามลำดับ JMF ก็ใช้หลักการเช่นเดียวกับระบบวีดีโอทั่วไป เริ่มตั้งแต่การจับภาพและเสียงโดยใช้กล้องวีดีโอ (เทียบได้เป็นอุปกรณ์จับข้อมูล) และนำข้อมูลที่ได้มาเก็บไว้ที่แหล่งข้อมูล (data source) ซึ่งเปรียบเสมือนม้วนวีดีโอเทป หลังจากนั้นแหล่งข้อมูลจะถูกนำไปประมวลผลโดยส่วนแสดงผล (player) ให้สามารถแสดงผลลัพธ์ออกทางอุปกรณ์แสดงผลต่าง ๆ การทำงานของระบบวีดีโอ และการทำงานของ JMF สามารถแสดงได้ดังรูปที่ 2.2



รูปที่ 2.2 เปรียบเทียบการทำงานของกล้องวีดีโอ กับ JMF

ส่วนประกอบของ JMF อธิบายได้ดังนี้

2.2.1 การจับสัญญาณข้อมูล

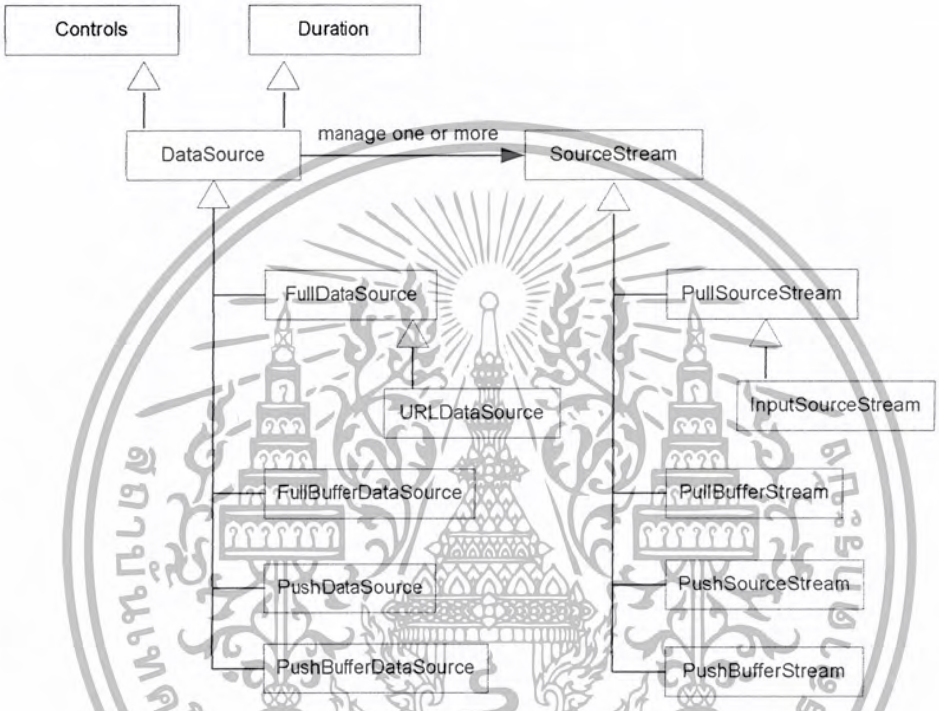
อุปกรณ์การจับข้อมูลแบบมัลติมีเดีย (multimedia capturing device) จะทำหน้าที่เป็นแหล่งข้อมูลประเภทสื่อ สำหรับการทำงานของ JMF และข้อมูลที่ได้จากอุปกรณ์การจับข้อมูลแบบมัลติมีเดีย ใน JMF จะแทนด้วยชื่อแหล่งข้อมูล ตัวอย่างของอุปกรณ์การจับข้อมูลแบบมัลติมีเดีย เช่น การ์ดบันทึกภาพเป็นอุปกรณ์ที่ใช้ในการจับภาพจากกล้องวีดีโอแล้วแปลงมาเป็นข้อมูลในการจับข้อมูลประเภทสื่อจากอุปกรณ์การจับข้อมูลแบบมัลติมีเดียที่ต้องการจะใช้ เราจะต้องดึงเอา MediaLocator ของอุปกรณ์นั้น ๆ ออกมาจาก object CaptureDeviceInfo ซึ่งได้มาจาก classCaptureDeviceManager จากนั้น เราสามารถนำ MediaLocator ที่ได้นี้ ไปสร้างส่วนแสดงผลหรือส่วนดำเนินโปรเซสเซอร์ (Processor) โดยตรง หรือใช้ MediaLocator นี้ สร้างแหล่งข้อมูลซึ่งเราจะใช้เป็นอินพุตให้กับส่วนแสดงผลหรือส่วนดำเนินโปรเซส และเราจะต้องเรียกใช้เมธอด start() ของ ส่วนแสดงผลหรือส่วนดำเนินโปรเซสเพื่อเริ่มต้นทำการจับข้อมูล

2.2.2 แหล่งข้อมูล

JMF API จะให้แหล่งข้อมูลในการจัดส่งมีเดียโดยที่แหล่งข้อมูลนั้น จะทำการที่เรียกว่า เอนแคปซูเลต (encapsulate) ตำแหน่งของมีเดีย (location of media) รวมไปถึงโปรโตคอลและซอฟท์แวร์ที่ใช้ในการส่งซึ่งในการได้มาซึ่งข้อมูลแต่ละครั้งนั้น แหล่งข้อมูลหนึ่ง ๆ จะสามารถส่งให้กับมีเดียได้เพียงตัวใดตัวหนึ่งเท่านั้น ไม่สามารถนำมาส่งให้กับมีเดียตัวอื่นได้

แหล่งข้อมูลนั้นจะถูกแสดงโดย JMF MediaLocator หรือ URL (universal resource locator) ตัวใดตัวหนึ่ง MediaLocator นั้นมีความคล้ายคลึงกันกับ URL และยังสามารถสร้างได้จาก URL ได้ด้วย ซึ่งจะถูกสร้างก็ต่อเมื่อตัวจัดการโปรโตคอล ที่มีลักษณะเช่นเดียวกัน (corresponding protocal handler) ไม่ได้ถูกติดตั้งอยู่ในระบบ กล่าวคือในภาษาจาวา นั้น URL สามารถถูกสร้างได้เพียงตัวเดียว ถ้าตัวจัดการโปรโตคอลที่มีลักษณะเช่นเดียวกันถูกติดตั้งอยู่ในระบบ

แหล่งข้อมูลจะเป็นตัวจัดการกับกลุ่มของ object SourceStream ซึ่งถ้าเป็นข้อมูลมาตรฐาน (standard data source) จะมีการเก็บข้อมูลที่ไว้ใช้ในการส่งเป็นแบบไบต์อาเรย์ (byte array) แต่ถ้าเป็นแบบข้อมูลที่มีการพักเก็บไว้ก่อน (buffer data source) จะใช้ object Buffer เป็นตัวส่งข้อมูล (unit of transfer) ซึ่ง JMF นั้น ได้กำหนดชนิดของออฟเจคไว้ดังรูปที่ 2.3



รูปที่ 2.3 JMF data model

2.2.2.1 แหล่งข้อมูลแบบ PULL และ PUSH (Pull and Push DataSource)

ข้อมูลที่เราได้มานั้น อาจจะได้มาจากแหล่งข้อมูลที่หลากหลาย อาทิเช่น จากข้อมูลในเครื่องที่เรามีอยู่แล้ว, จากเครือข่าย หรือไม่ก็จากการบอร์ดคาสท์ (broadcast) ซึ่งแหล่งของข้อมูลใน JMF สามารถแบ่งออกเป็น 2 ประเภท ตามวิธีการส่งข้อมูล ดังนี้

2.2.2.1.1 แหล่งข้อมูลแบบ PULL (Pull DataSource)

วิธีการเริ่มส่งข้อมูลแบบนี้ไคลเอ้นท์ (client) จะเป็นคนเริ่มสั่งให้มีการส่งข้อมูลและอัตราของการส่ง ลูกข่ายสามารถที่จะควบคุมการส่งข้อมูลต่าง ๆ ได้ โปรโตคอลที่ใช้ในการส่งแบบนี้ก็จะเป็นพวก Hypertext Transfer Protocol (HTTP) และ FILE JMF ได้ทำการกำหนดชนิดของแหล่งข้อมูลแบบ PULL ไว้ 2

แบบคือ PullDataSource และ PullBufferDataSource โดยจะให้ object ของ class Buffer เป็นตัวที่ใช้ส่งข้อมูลข้อแตกต่างของทั้ง 2 แบบนี้ คือการส่งข้อมูล ทาง PullDataSource จะเป็นการส่งแบบต่อเนื่องเรื่อย ๆ แต่ถ้าเป็นแบบ PullBufferDataSource จะส่งข้อมูลมา โดยจะส่งมาแบบทีละบัฟเฟอร์

2.2.2.1.2 แหล่งข้อมูลแบบ PUSH (Push Data-Source)

วิธีการเริ่มส่งข้อมูลแบบนี้เซิร์ฟเวอร์จะเป็นผู้เริ่มสั่งให้มีการส่งข้อมูลและอัตราของการส่ง ซึ่ง PushData-Source นี้จะประกอบไปด้วยมีเดียบอร์ดคาสท์ (broadcast media), มีเดียมัลติคาสท์ (multicast media) และวีดีโอตามต้องการ สำหรับการส่งข้อมูลจำพวกบอร์ดคาสท์นี้ ก็จะมี โปรโตคอลที่เรียกว่า Real Time Transport Protocol (RTP) ซึ่งโปรโตคอลตัวนี้ถูกพัฒนาโดย Internet Engineer Task Force (IETF) ส่วนวีดีโอตามต้องการนั้น จะมีโปรโตคอลที่เรียกว่า มีเดียเบสโปรโตคอล (MediaBase protocol) ซึ่งถูกพัฒนาโดย SGI JMF ได้ทำการกำหนดชนิดของแหล่งของข้อมูลแบบ PUSH ไว้ 2 แบบ คือ PushDataSource และ PushBufferDataSource โดยจะให้ object ของclass Buffer เป็นตัวที่ใช้ส่งข้อมูล

ตัวอย่างที่แสดงให้เห็นถึงความแตกต่างที่ชัดเจนของการส่งข้อมูลทั้ง 2 แบบนี้ คือ ถ้าเรามีไฟล์ประเภท MPEG แหล่งของข้อมูลแบบ PULL จะเป็นการอนุญาตให้ผู้ใช้สามารถที่จะทำการเล่นอีกครั้ง (Replay) หรือทำการเลือกดูข้อมูลส่วนไหนก็ได้ของไฟล์ ในทางตรงกันข้าม ถ้าเป็นแหล่งข้อมูลแบบ PUSH ไฟล์ประเภท MPEG ตัวนี้ จะถูกเก็บไว้ที่เซิร์ฟเวอร์จากนั้นเซิร์ฟเวอร์จะบอร์ดคาสท์ส่งไปให้ผู้ใช้แต่ละคนดู ผู้ใช้ไม่สามารถทำการเล่นอีกครั้งหรือเลือกดูข้อมูลแบบสุ่มได้ ผู้ใช้ต้องดูไปเรื่อยๆ ตามแต่เซิร์ฟเวอร์จะส่งมาให้ เหมือนกับการถ่ายทอดโทรทัศน์ แต่ถ้าเป็นพวกวีดีโอตามต้องการ จะอนุญาตให้ทำการเลือกดูส่วนไหนของไฟล์ได้ แต่การเดินหน้า (forward) และ ถอยหลัง (rewind) จะช้ามาก

2.2.2.2 แหล่งข้อมูลแบบพิเศษ (Specialty DataSources)

JMF ได้กำหนด แหล่งข้อมูลแบบพิเศษไว้ 2 อย่างด้วยกันคือ แหล่งข้อมูลที่สามารถโคลนได้ (cloneable data sources) และแหล่งข้อมูลที่รวมกันได้ (merging data sources)

2.2.2.2.1 แหล่งข้อมูลที่สามารถลอกแบบได้

มีไว้สำหรับการสร้างโคลน (clone) ไม่ว่าจะเป็นแหล่งของข้อมูลแบบ PUSH หรือ PULL สำหรับการ สร้างแหล่งข้อมูลที่สามารถลอกแบบได้ เราจะทำการเรียกเมธอด createCloneable แหล่งข้อมูลจากคลาส Manager จากนั้นก็ส่งแหล่งของข้อมูลที่เราจะโคลนไปให้กับ เมธอดนี้ เมื่อ แหล่งข้อมูลที่จะทำการโคลนได้ถูกส่งไปเรียบร้อยแล้ว เราสามารถเรียกใช้งานแหล่งข้อมูลที่ทำการโคลนกับตัวโคลนของมันได้ โดยไม่ต้องเรียกใช้งานโดยตรงกับแหล่งของข้อมูลต้นแบบ แหล่งข้อมูลที่สามารถโคลนได้จะสนับสนุนอินเตอร์เฟส SourceCloneable ซึ่งถูกกำหนดโดย เมธอด createClone เมื่อเราทำการเรียกเมธอดนี้ขึ้นมา เราสามารถกำหนดได้ว่าจะสร้างโคลนขึ้นมาเท่าไร ซึ่งโคลนที่สร้างขึ้นมานี้ เราสามารถควบคุมได้โดยผ่านแหล่งของข้อมูลที่สร้างตัวมันขึ้นมา (เราสามารถใช้เมธอด connect, disconnect, start หรือ stop เพื่อเป็นการควบคุมแหล่งของข้อมูลที่สามารถโคลนได้ได้)

โคลนที่เราสร้างขึ้นมานี้ไม่จำเป็นที่จะต้องมีคุณสมบัติเหมือนกับแหล่งของข้อมูลต้นแบบที่ทำการโคลน กล่าวคือ สมมติว่าเราสร้างโคลนสำหรับอุปกรณ์จับข้อมูลซึ่งมีฟังก์ชั่นเหมือนกับ แหล่งของข้อมูลที่สามารถโคลนได้ทุกประการ ในกรณีที่ไม่มีการเรียกใช้งานจาก แหล่งของข้อมูลที่สามารถโคลนได้ ตัวโคลนของมันก็จะไม่ทำการผลิตข้อมูล แต่ถ้าเรามีการใช้งานทั้ง 2 ตัวคือ แหล่งของข้อมูลที่สามารถโคลนได้และตัวโคลนของมัน ตัวโคลนของมันก็จะทำการผลิตข้อมูลด้วยอัตราที่เทียบเท่ากับต้นแบบของมัน

2.2.2.2.2 แหล่งข้อมูลที่รวมได้ (MergingDataSource)

ไว้สำหรับการรวม SourceStream จากแหล่งของข้อมูล 2 หรือ 3 ตัว เข้าเป็นแหล่งข้อมูล ตัวเดียวโดยเราสามารถควบคุมกลุ่มของแหล่งของข้อมูลนี้ได้เหมือนกับการควบคุมแหล่งข้อมูล ตัวเดียว (โดยใช้เมธอด connect, disconnect, start หรือ stop บนคลาส MergingDataSource เพื่อทำการรวมแหล่งของข้อมูล)

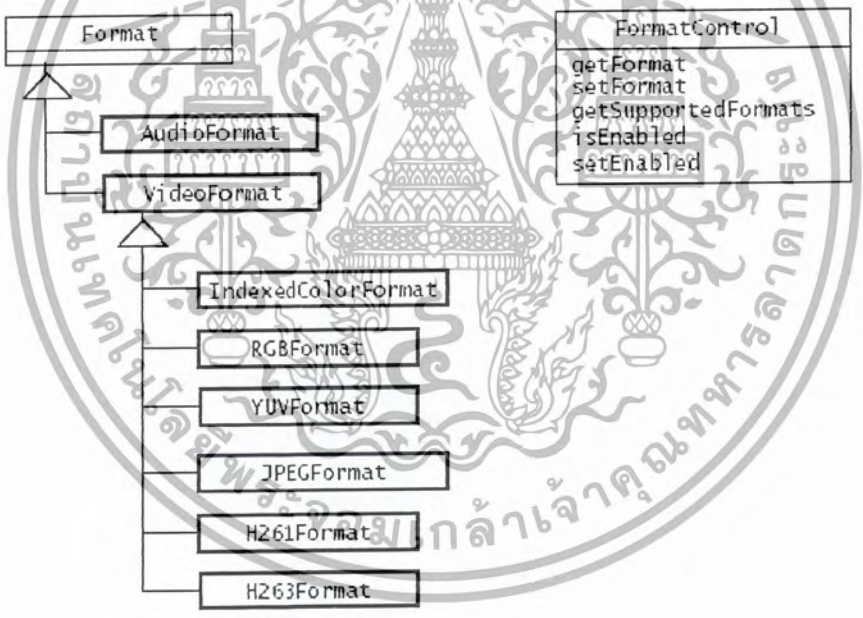
ในการสร้างแหล่งข้อข้อมูลที่รวมกัน เราสามารถเรียกใช้เมธอด createMergingDataSource จาก คลาส Manager จากนั้นก็ส่งอาร์เรย์ที่ประกอบไปด้วยแหล่งของข้อมูลที่ต้องการรวมเข้าด้วยกัน แต่มีข้อแม้อยู่ว่าแหล่งของข้อมูลที่จะทำการรวมกันนั้นจะต้องเป็นชนิดเดียวกัน กล่าวคือ เราไม่สามารถที่จะนำแหล่งของข้อมูลแบบ PULL มารวมกันกับแหล่งของข้อมูลแบบ PUSH ได้ และ

ระยะเวลาของ แหล่งของข้อมูลที่รวมกันเสร็จเรียบร้อยแล้ว จะเท่ากับค่าเวลามากที่สุดของแหล่งของข้อมูลก้อนที่จะทำการรวมกันแต่ละตัวบวกกัน

2.2.2.3 รูปแบบข้อมูล (Data Format)

รูปแบบของการเก็บข้อมูลของสื่อนั้น จะถูกแสดงด้วยออฟเจค (object) หนึ่ง ที่เรียกว่า ฟอร์เมต (Format) ซึ่งจะอธิบายถึงชื่อของการแปลงข้อมูล เป็นรหัสของรูปแบบของการเก็บข้อมูล (format's encoding name) และชนิดของรูปแบบของการเก็บข้อมูลที่ต้องการ

JMF ได้ใช้ออฟเจค Format ในการกำหนดรูปแบบของการเก็บข้อมูลของเสียงและภาพ ดังแสดงไว้ในรูปที่ 2.4



รูปที่ 2.4 รูปแบบมีเดีย JMF (JMF media formats)

แอดทริบิวท์ AudioFormat จะเป็นตัวอธิบายถึงรูปแบบของการเก็บข้อมูลของเสียง ซึ่งประกอบไปด้วย อัตราแซมเปิล (sample rate), บิทต่อแซมเปิล (bits per sample) และจำนวนของช่อง (number of channels) ส่วน VideoFormat นั้น จะเป็นตัวเอนแคปซูเลตเกี่ยวกับข้อมูลที่มีความสัมพันธ์กันกับข้อมูลภาพซึ่ง VideoFormat นี้ได้เป็นตัวอธิบายเกี่ยวกับรูปแบบการเก็บข้อมูลพื้นฐานของวีดีโอดังนี้

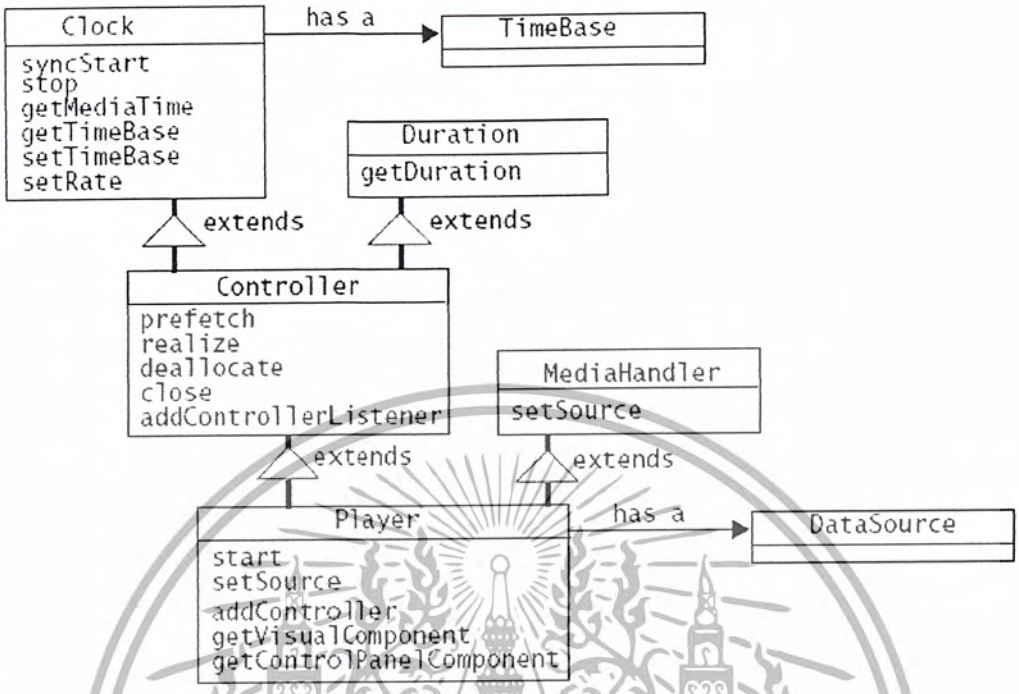
- IndexedColorFormat
- RGBFormat
- YUVFormat
- JPEGFormat
- H261Format
- H263Format

2.2.3 ส่วนแสดงผล (player)

ส่วนแสดงผลเป็นส่วนที่ใช้ในการประมวลผลสตรีมที่อินพุตเข้ามาของข้อมูล โดยจะแสดงผลได้ก็ต่อเมื่อแหล่งข้อมูลนั้นได้จ่ายสตรีมที่อินพุตเข้ามายังส่วนแสดงผล ในส่วนของการแสดงผลนั้น จะขึ้นอยู่กับชนิดของข้อมูลที่ใช้ในการแสดงผล

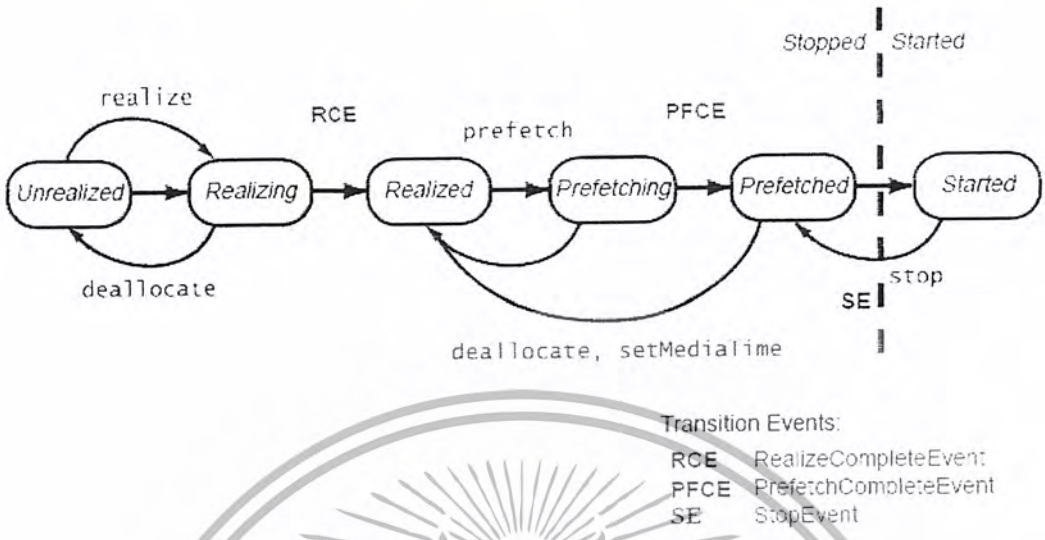
รูปที่ 2.5 โมเดลส่วนแสดงผลของ JMF (JMF player model)

ในส่วนแสดงผลนี้ จะไม่สนใจว่าข้อมูลจะถูกกระทำโปรเซสหรือเรนเดอร์ (render) มาได้อย่างไร มันจะเป็นเพียงแค่ตัวควบคุมมาตรฐานทั่วไป ที่มีอยู่ในทุก ๆ อุปกรณ์ควบคุมข้อมูลต่าง ๆ ตัวอย่างเช่นส่วนแสดงผลจะมีแค่เพียงปุ่มเล่นและหยุดเท่านั้น ไม่มีพวกตัวปรับแต่งสำหรับปรับแต่งข้อมูล เราสามารถแสดงคลาสไดอะแกรม (classdiagram) ของส่วนแสดงผล ได้ดังรูปที่ 2.6



รูปที่ 2.6 คลาสไดอะแกรมของส่วนแสดงผลของ JMF

ส่วนแสดงผลจะมีสถานะที่เป็นไปได้อยู่ 6 แบบด้วยกัน สถานะหลัก 2 สถานะ Stopped และ Started ซึ่งได้ถูกนิยามไว้ในอินเตอร์เฟส Clock แต่เพื่อให้ง่ายต่อการจัดการทรัพยากร (Resource) ตัวควบคุมจึงได้มีการแบ่งสถานะ Stopped ออกเป็น 5 สถานะย่อย คือ Unrealized, Realizing, Realized, Prefetching, และ Prefetched แสดงได้ดังรูปที่ 2.7



รูปที่ 2.7 สถานะต่าง ๆ ของส่วนแสดงผล JMF

ในการทำงานโดยทั่วไปส่วนแสดงผลจะต้องผ่านสถานะต่าง ๆ ตามขั้นตอน ตั้งแต่เริ่มไปจนถึง Started ดังนี้

เมื่อส่วนแสดงผลถูกสร้างขึ้นมาจะอยู่ในสถานะ unrealized ซึ่งในตอนนี้ส่วนแสดงผลจะยังไม่มีข้อมูลเกี่ยวกับสื่อที่ตัวเองจะติดต่อด้วย

เมื่อเมธอด Realize ถูกเรียกส่วนแสดงผล จะเปลี่ยนสถานะตัวเองจาก unrealized เป็น realizing ซึ่งในสถานะนี้ส่วนแสดงผล จะทำการตรวจสอบและพิจารณาความต้องการทรัพยากรของตนเอง ในกระบวนการนี้ส่วนแสดงผลจะขอทรัพยากรที่ตนเองจะร้องขอเพียงแค่ครั้งเดียว ที่ไม่เป็น exclusive-use resource (**exclusive-use resource คือ ทรัพยากรที่มีจำกัด ที่สามารถถูกใช้ได้โดยส่วนแสดงผลเดียวในเวลาหนึ่ง ๆ เท่านั้น ซึ่งทรัพยากรเหล่านี้ จะถูกขอในช่วงสถานะ prefetching**)

เมื่อส่วนแสดงผลผ่านสถานะ realizing แล้ว ก็จะมาอยู่ในสถานะ realized ซึ่งในสถานะนี้ส่วนแสดงผลจะรู้ว่ามันต้องใช้ทรัพยากรอะไรบ้างและรู้ว่าชนิดของข้อมูลที่จะใช้แสดงนั้นคืออะไร เนื่องจากส่วนแสดงผลที่อยู่ในสถานะนี้จะรู้วิธีการแสดงข้อมูลออกมา (render) ดังนั้น มันจึงสามารถให้ส่วนประกอบ (components) สำหรับแสดงผลและควบคุมได้ การเชื่อมต่อกับออฟเจคอื่น ๆ ในระบบ ได้ถูกจัดเตรียมแล้ว แต่ว่าตัวส่วนแสดงผลเอง ยังไม่ได้จับจองทรัพยากรใด ๆ ที่จะป้องกันส่วนแสดงผลจากการเริ่ม

เมื่อเมธอด Prefetch ถูกเรียก ส่วนแสดงผลจะเข้าสู่สถานะ prefetching ซึ่งในสถานะนี้ ส่วนแสดงผลจะเตรียมตัวที่จะนำเสนอข้อมูลออกมา ในกระบวนการนี้ ส่วนแสดงผลจะทำการอ่าน

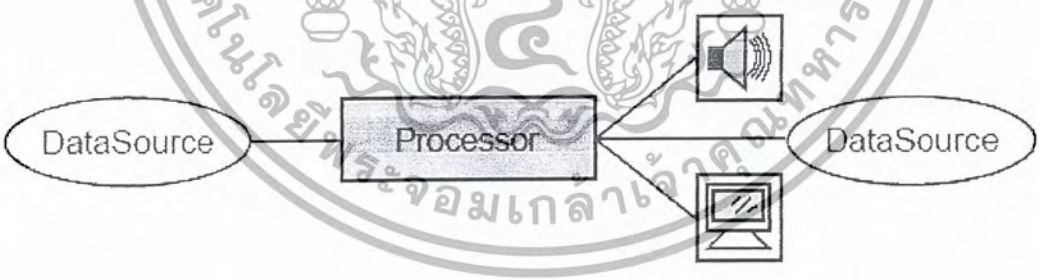
ข้อมูลมาเตรียมไว้ และทำการจับจอง exclusive-use resource และสิ่งต่าง ๆ ที่ต้องการในการนำเสนอ สถานะ prefetching นี้สามารถเกิดขึ้นได้อีก หากมีการเปลี่ยนแปลงใด ๆ ในตัวส่วนแสดงผลที่ทำให้ต้องมีการเตรียมข้อมูลใหม่ เช่น การเปลี่ยนตำแหน่งที่จะอ่านข้อมูล การเปลี่ยนอัตราในการประมวลผลข้อมูล ทำให้ต้องอ่านบัฟเฟอร์มากขึ้น หรือการเปลี่ยนวิธีการประมวลผลข้อมูลเป็นต้น

เมื่อส่วนแสดงผลผ่านสถานะ prefetching แล้ว ก็จะมาอยู่ในสถานะ prefetched ซึ่งในสถานะนี้ส่วนแสดงผลพร้อมแล้วที่จะถูกเริ่ม

การเรียกเมธอด Start จะทำให้ส่วนแสดงผลไปอยู่ในสถานะ started ซึ่งในสถานะนี้เวลาจริงของระบบ และเวลาของข้อมูลจะถูกแมพให้ตรงกัน และนาฬิกาจะเริ่มทำงาน แม้ว่าตัวส่วนแสดงผลเองอาจต้องจะรอให้ถึงเวลาหนึ่ง ก่อนที่จะเริ่มนำเสนอข้อมูลก็ตาม

2.2.4 โปรเซสเซอร์

โปรเซสเซอร์เป็นอีกตัวหนึ่งที่มีไว้สำหรับการแสดงผลสตรีมที่อินพุตของข้อมูลเหมือนกับส่วนแสดงผล แต่โปรเซสเซอร์นี้มีความสามารถมากกว่า กล่าวคือโปรเซสเซอร์สามารถทำการเปลี่ยนแปลง, ปรับปรุงหรือแก้ไขสตรีมที่อินพุตของข้อมูลก่อนการแสดงผลได้ ซึ่งชนิดของข้อมูลที่ส่วนแสดงผลสนับสนุนจะสามารถใช้กับโปรเซสเซอร์ได้

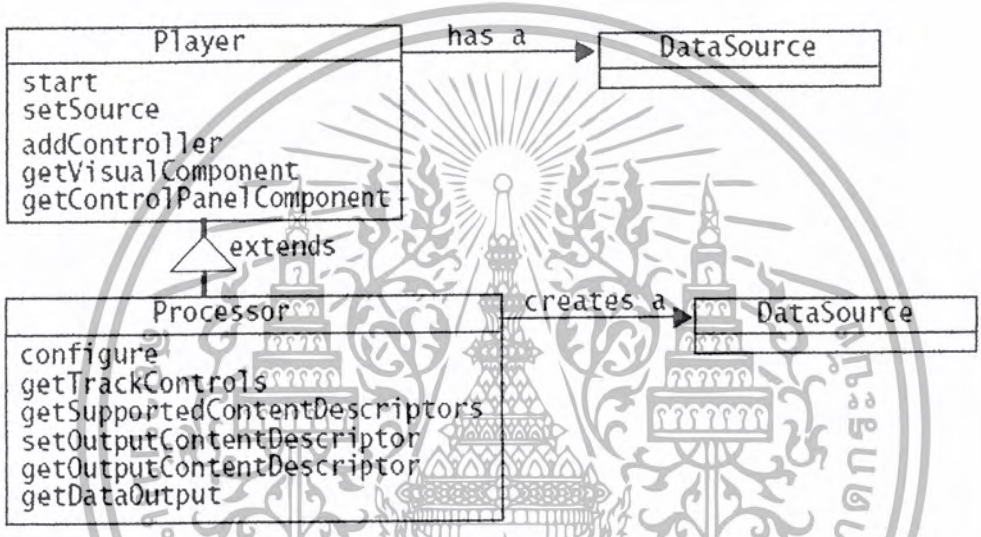


รูปที่ 2.8 โมเดลของโปรเซสเซอร์ใน JMF

สำหรับการแสดงผลข้อมูลนั้น โปรเซสเซอร์สามารถทำได้หลายวิธี คือ สามารถทำการสร้างแหล่งข้อมูลขึ้นมาใหม่จากนั้น จะนำแหล่งของข้อมูลตัวนี้ส่งไปยังส่วนแสดงผลหรือโปรเซสเซอร์ตัวอื่นเพื่อแสดงผล หรือส่งไปยังเป้าหมายอื่น ๆ ในรูปของไฟล์

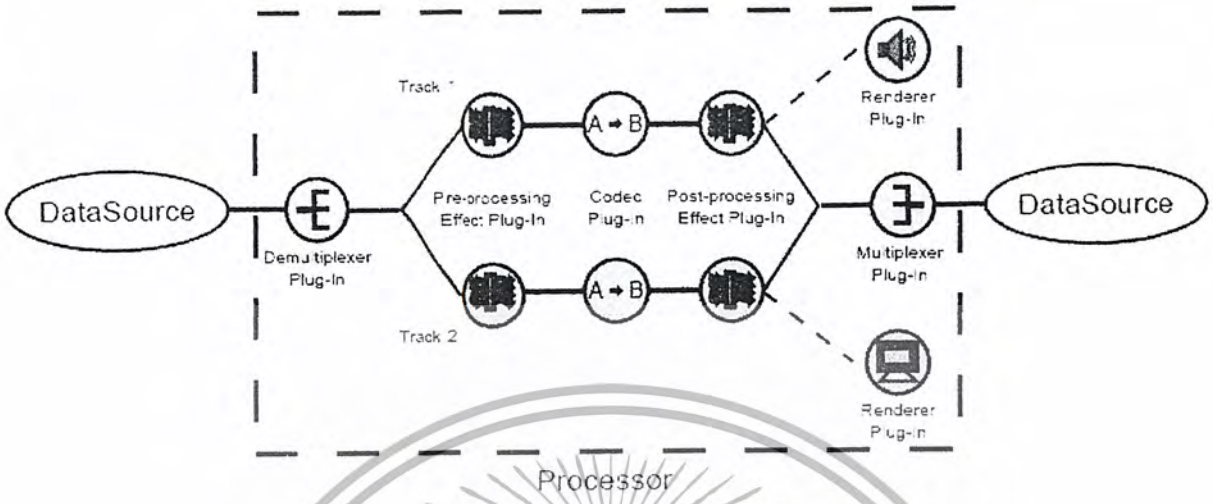
2.2.4.1 กระบวนการทำงานของโปรเซสเซอร์

โปรเซสเซอร์คือส่วนแสดงผลที่นำเอาแหล่งของข้อมูลมาทำการปรับปรุงเปลี่ยนแปลงข้อมูล เพื่อสร้างเป็นแหล่งข้อมูลอีกตัวขึ้นมา และสามารถส่งข้อมูลไปให้อุปกรณ์แสดงผลอย่างอื่นได้ ถ้าข้อมูลที่ส่งออกไปนั้นถูกส่งไปยังแหล่งข้อมูลซึ่งจะสามารถส่งข้อมูลที่อยู่ในตัวมันเองไปให้ ส่วนแสดงผลหรือโปรเซสเซอร์อื่น ๆ หรือไม่ก็ส่งไปให้คาต้าซิงค์ได้อีกด้วย



รูปที่ 2.9 คลาสไดอะแกรมของโปรเซสเซอรของ JMF

ในขณะที่กระบวนการทำงานของส่วนแสดงผลถูกกำหนดโดยนักพัฒนาโปรแกรม โดยที่โปรเซสเซอร์ได้มีการอนุญาตให้ นักพัฒนาโปรแกรมสามารถกำหนดชนิดของกระบวนการทำงานที่เกี่ยวกับข้อมูลได้เช่นกัน ตัวอย่างเช่น เอฟเฟค การผสมหรือการทำงานแบบเรียลไทม์ ซึ่งเราจะแสดงกระบวนการทำงานของโปรเซสเซอร์ถูกแบ่งออกเป็นขั้นตอนดังรูปที่ 2.10



รูปที่ 2.10 ภาพโดยรวมสถานะของโปรเซสเซอร์ของ JMF

การดีมัลติเพล็กคือกระบวนการที่กระจายข้อมูลออกเป็นแทรคย่อย ๆ กล่าวคือถ้าข้อมูลที่เข้ามาประกอบไปด้วยหลายแทรครวมกัน เราสามารถทำการกระจายข้อมูลให้ออกมาเป็นแต่ละแทรคได้ ยกตัวอย่างเช่น ไฟล์มัลติมีเดีย แบบ QuickTime เราสามารถทำการดีมัลติเพล็กออกเป็นแทรคเสียงและ ภาพได้ ข้อมูลที่เข้ามายังโปรเซสเซอร์จะสามารถทำการดีมัลติเพล็กได้ก็ต่อเมื่อ เป็นข้อมูลแบบดีมัลติเพล็กคาต้า (multiplex data) เท่านั้น

พรี-โปรเซสซิ่ง (Pre – Processing) คือกระบวนการที่ตอบสนองต่ออัลกอริธึม effect ของแต่ละแทรคหลังจากการดีมัลติเพล็ก

ทรานสโคดดิ้ง (Transcoding) คือกระบวนการที่ทำการแปลงสัญญาณของแต่ละแทรคจากรูปแบบหนึ่งให้ไปเป็นอีกรูปแบบหนึ่ง ถ้าทำการแปลงสัญญาณจากข้อมูลชนิดที่มีการบีบอัดไปเป็นข้อมูลชนิดที่ไม่มีการบีบอัดจะเรียกว่า “การถอดรหัส” และในทางกลับกัน ถ้าทำการแปลงสัญญาณจากข้อมูลที่ไม่มีการบีบอัด ไปเป็นข้อมูลที่มีการบีบอัดจะเรียกว่า “ การเข้ารหัส ”

โพส-โปรเซสซิ่ง (Post – Processing) คือเป็นกระบวนการที่เกี่ยวกับการตอบสนองต่อ อัลกอริธึ่มเอฟเฟคของแทรคที่ถูกถอดรหัส

การมัลติเพล็ก (Multiplexing) คือกระบวนการรวมแทรคแต่ละแทรคที่จะออกจาก โปรเซสเซอร์ให้เป็นข้อมูลเพียงชุดเดียว เช่น เราสามารถทำการรวมแทรคของภาพและเสียงให้เป็นข้อมูลชนิด MPEG-1 เพียงข้อมูลเดียวได้ ซึ่งเราสามารถกำหนดชนิดของข้อมูลที่จะออกจากโปรเซสเซอร์ได้จากเมธอดที่ชื่อว่า setOutputContentDescriptor()

การเรนเดอร์ (Randering) คือกระบวนการในการส่วนของการแสดงข้อมูลไปยังอุปกรณ์แสดงผลในกระบวนการทำงานของแต่ละขั้นตอนจะถูกจัดการโดยส่วนประกอบของ JMF ที่มีชื่อว่า " JMFplug-in " ซึ่งถ้าโปรเซสเซอร์ของเราสนับสนุน TrackControls เราสามารถที่จะเลือก ปลั๊กอิน (plug-in) ที่เราต้องการใช้ในแต่ละแทรคได้ ซึ่งมีปลั๊กอินให้เลือกทั้งหมด 5 ชนิด ดังนี้

ตัวดีมัลติเพล็ก– เป็นการกระจายข้อมูลที่เป็นมีเดียสตรีม ตัวอย่างเช่น WAV, MPEG หรือQuickTime ที่สนับสนุนการมัลติเพล็กออกเป็นหลายแทรค

เอฟเฟค– เป็นการเพิ่มลักษณะพิเศษลงไปในแทรคของข้อมูล

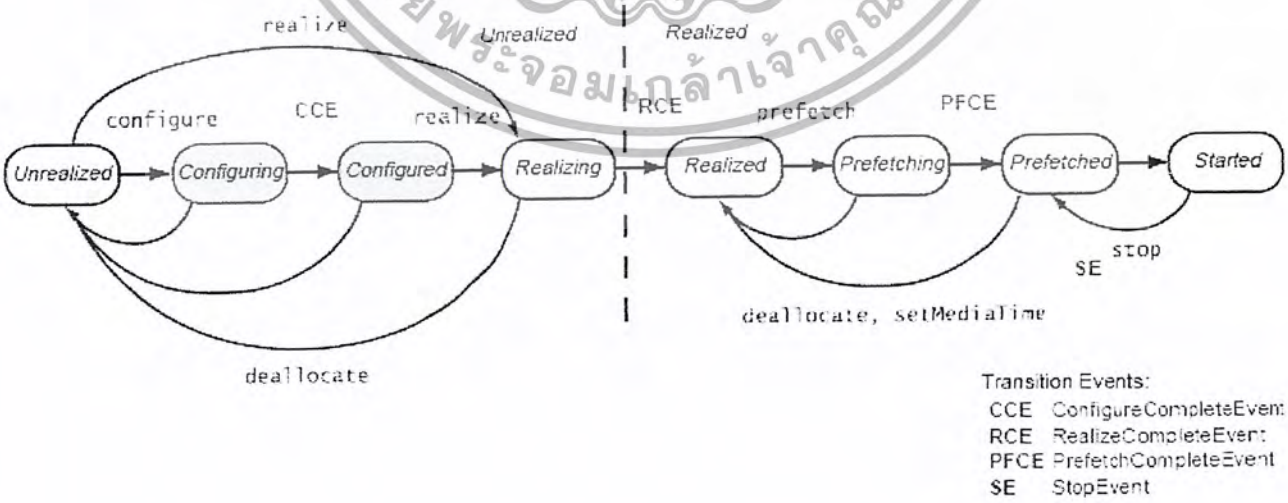
การเข้ารหัส – เป็นการทำการเข้ารหัสข้อมูล หรือการถอดรหัสข้อมูล

ตัวมัลติเพล็ก (Multiplexer) – เป็นการผสมแทรคหลาย ๆ แทรค ให้เป็นข้อมูลเพียง ชุดเดียว แล้วส่งข้อมูลนี้ไปยังแหล่งของข้อมูล

ตัวเรนเดอร์ (Renderer) – เป็นการแสดงผลข้อมูลในแต่ละแทรค ไปยังอุปกรณ์แสดงภาพและเสียง

2.2.4.2 สถานะของโปรเซสเซอร์

โปรเซสเซอร์จะมีสถานะเพิ่มขึ้นจากส่วนแสดงผลด้วยกัน 2 สถานะคือ Configuring และ Configured โดยที่ทั้ง 2 สถานะนี้จะเกิดขึ้นก่อนที่โปรเซสเซอร์จะเข้าสู่สถานะ Realizing



รูปที่ 2.11 สถานะของโปรเซสเซอร์ของ JMF (JMF Processor Stages)

- โปรเซสเซอร์จะเข้าสู่สถานะ Configuring หลังจากที่มีการเรียกใช้เมธอด configure เมื่อโปรเซสเซอร์อยู่ในสถานะนี้ จะมีการติดต่อกับแหล่งข้อมูลโดยการดีมัลติเพล็กข้อมูลที่เข้ามา และแจ้งให้แหล่งข้อมูลทราบถึงชนิดของข้อมูลนี้
- โปรเซสเซอร์จะเข้าสู่สถานะ Configured เมื่อกำลังติดต่อกับแหล่งข้อมูลและได้ทราบถึงชนิดของข้อมูลแล้ว เมื่อโปรเซสเซอร์อยู่ในสถานะ Configured จะได้คลาสที่ชื่อ ConfigureCompleteEvent ออกมา
- เมื่อมีการเรียกใช้งานเมธอด realize Processor จะเปลี่ยนไปเป็นสถานะ Realized ซึ่งเราจะถือว่าโปรเซสเซอร์ของเราได้มีการทำงานเสร็จสมบูรณ์

ในขณะที่การทำงานของโปรเซสเซอร์อยู่ในสถานะ Configured เราจะสามารถเรียกใช้งาน methodgetTrackControls เพื่อเอาค่าใน object TrackControl สำหรับแต่ละแทรคของข้อมูลมา ซึ่งเราสามารถระบุการทำงานของโปรเซสเซอร์ได้จาก object TrackControl นี้

ในการเรียกใช้ method realize โดยตรงโปรเซสเซอร์จะเปลี่ยนจากสถานะจาก Unrealized ไปเป็นสถานะ Realized โดยอัตโนมัติ แต่เราจะไม่สามารถใช้ option TrackControl ดูรายละเอียดของข้อมูลแต่ละแทรคได้ และเราไม่สามารถเรียกใช้เมธอด TrackControl เมื่อโปรเซสเซอร์อยู่ในสถานะ Realized ได้

2.2.4.3 การควบคุมโปรเซสเซอร์

เราสามารถเรียกใช้เมธอด getTrackControls ในการดึงเอาข้อมูลในแต่ละแทรค โดยผ่าน classTrackControl เพื่อควบคุมการทำงานของโปรเซสเซอร์ที่กระทำต่อแทรคแต่ละแทรคเราสามารถเรียกใช้คลาส PlugInManager เพื่อทำการตรวจสอบว่าเราได้มีการติดตั้งปลั๊กอินอะไร

การควบคุมการทรานสโคดในแต่ละแทรคโดยตัวเข้ารหัสแต่ละตัว สามารถเรียกใช้ตัวช่วยจากเมธอด getControls ของ TrackControl ได้ ซึ่งเมธอดนี้จะให้ค่า codec control ของแทรคเช่นBitRateControl และ QualityControl

ในการกำหนดชนิดของข้อมูลเอาท์พุตให้เป็นไปตามที่เราต้องการ เราสามารถเรียกใช้ methodsetFormat ในการระบุชนิดของรูปแบบที่เราต้องการ จากนั้นโปรเซสเซอร์จะเลือกตัว เข้ารหัสและตัวเรนเดอร์ที่เหมาะสมให้ อีกวิธีหนึ่งที่เราสามารถทำได้คือ ทำการระบุชนิดของข้อมูลเมื่อมีการสร้างโปรเซสเซอร์ขึ้นมาโดยใช้คลาส ProcessorModel

ซึ่งในคลาส ProcessorModel นี้จะทำการกำหนด อินพุตและเอาท์พุตที่ต้องการสำหรับโปรเซสเซอร์ที่เราสร้างขึ้นมา

2.2.4.4 ข้อมูลเอาท์พุต (Data Output)

เมธอด getDataOutput จะทำการคืนค่าผลลัพธ์ที่ได้จากโปรเซสเซอร์ในรูปของแหล่งข้อมูลซึ่งแหล่งข้อมูลนี้อาจอยู่ในรูปของ PushDataSource, PushBufferDataSource, PullDataSource และPullBufferDataSource และสามารถเป็นข้อมูลที่ป้อนให้กับส่วนแสดงผลหรือโปรเซสเซอร์ตัวอื่น หรือเป็นข้อมูลที่ส่งไปยังดาต้าซิงค์ได้ ซึ่งโปรเซสเซอร์ที่ไม่มีการส่งค่าผลลัพธ์ให้แหล่งข้อมูลจะถูกเปรียบเสมือนว่าเป็นส่วนแสดงผล

2.2.5 เมเนเจอร์ (Managers)

ใน JMF API มีการทำงานของ object ต่าง ๆ หลายส่วนประกอบกัน เช่น การจับสัญญาณข้อมูล, การประมวลผล (process) และการนำเสนอสื่อเชิงเวลาซึ่งแต่ละส่วนนี้จะมีความสัมพันธ์ซึ่งกันและกัน เราจำเป็นที่จะต้องมีออฟเจ็คที่ใช้เป็นตัวกลาง ในการทำให้ออฟเจ็ค ต่าง ๆ สามารถทำงานร่วมกันได้ ซึ่ง JMF ก็มี intermediary object ไว้สำหรับให้เรียกใช้อยู่แล้ว ซึ่งเราเรียก intermediary object ว่าเมเนเจอร์

โดย JMF มีเมเนเจอร์ที่เกี่ยวข้องทั้งหมด 4 เมเนเจอร์ คือ

1. Manager ใช้จัดการเกี่ยวกับการสร้างออฟเจค ของคลาส Players, Processors, DataSources และ DataSinks
2. Package Manager ใช้ในการเก็บรายการ (registry) ของแพ็คเกตทั้งหมดที่เกี่ยวกับ JMF เช่น คลาส Players, Processors, DataSources และ DataSinks ที่เราสามารถเขียนขึ้นมาเอง
3. CaptureDeviceManager ใช้ในการเก็บรายการของอุปกรณ์จับข้อมูลที่ใช้ได้ต่าง ๆ
4. PlugInManager ใช้ในการเก็บรายการของปลั๊กอินต่าง ๆ ที่มีใน JMF ซึ่งใน JMF มีปลั๊กอินด้วยกันทั้งสิ้น 5 ปลั๊กอินคือ Multiplexers, Demultiplexers, Codecs, Effects และRenderers

ในการพัฒนาโปรแกรมโดยใช้ JMF เราจำเป็นที่จะต้องใช้เมธอดของคลาส Manager เพื่อทำการสร้าง Players, Processors, DataSources และ DataSinks ตัวอย่างเช่น ในการจับภาพของกล้องวีดีโอนั้น จำเป็นที่จะต้องใช้คลาส CaptureDeviceManager เพื่อทำการค้นหาอุปกรณ์จับข้อมูล จากนั้นจึงทำการดึงข้อมูลจาก อุปกรณ์จับข้อมูลตัวนั้นมาเก็บไว้ และถ้าต้องการนำข้อมูลที่จับข้อมูล

มาได้มาทำการประมวลผลต่อ เราสามารถเรียกใช้คลาส PlugInManager เพื่อทำการดึงรายการของปลั๊กอินทั้งหมดที่ได้ลงทะเบียนไว้แล้ว

เราสามารถทำการเพิ่มรายการของปลั๊กอินใหม่ ๆ ได้โดยการใช้คลาส PlugInManager ซึ่งจะทำให้ปลั๊กอินที่เราเพิ่มเข้ามาใหม่นี้ ใช้ได้กับโปรเซสเซอร์ที่อยู่ในแอพพลิเคชั่นของเราได้ และถ้าเราสร้างส่วนแสดงผล, โปรเซสเซอร์, DataSources และดาต้าซิงค์ขึ้นมาใหม่ เราจะต้องทำการเพิ่มรายการ package prefix ให้กับคลาส PackageManager ด้วยเมธอดในคลาสของมันเอง

2.3 การทำงานกับมีเดียสตรีมแบบเรียลไทม์ (Real-Time Media Streams)

เพื่อที่จะส่งหรือรับสื่อแบบบอร์ดคาสท์ หรือ วีดีโอคอนเฟอเรนซ์บนอินเตอร์เน็ตหรืออินทราเน็ต จะต้องรับและถ่ายทอดมิเดียสตีมในระบบแบบเรียลไทม์ได้ หัวข้อนี้จะกล่าวถึงหลักการของการส่งมิเดียสตีมและบอกรายละเอียดของโปรโตคอลที่ใช้สำหรับรับและส่งมิเดียสตีมผ่านระบบเครือข่าย

2.3.1 การสตรีมมิ่งมีเดีย

เมื่อรายละเอียดสื่อถูกส่งแบบสตรีมไปเครื่องลูกข่ายในระบบเรียลไทม์ ลูกข่ายสามารถเริ่มเล่นได้เลยโดยไม่ต้องรอให้ดาวน์โหลดเสร็จก่อน แต่ในความจริง การสตรีมที่ไม่มีช่วงรอระหว่างการดาวน์โหลดนั้นไม่มีทางเป็นไปได้ การสตรีมมิเดียใช้เทคโนโลยีทั้งสองอันคือ การส่งคอนเทนบนเครือข่ายแบบเรียลไทม์และมิเดียคอนเทนแบบเรียลไทม์ที่ถูกส่ง

การสตรีมมิเดียคือทุกที่ที่คุณเห็นบนเวปที่เป็นวิทยุสด, ทีวีบอร์ดคาสท์, เวปคาสท์คอนเสิร์ตและขณะนี้มันเป็นไปได้ที่จะควบคุมการรวมภาพและเสียงบนอินเตอร์เน็ตด้วยการส่งโดยตรงที่มีประสิทธิภาพ (คอนเทนมิเดียที่ติดต่อกันผ่านเครือข่าย) การสตรีมมีเดียกำลังเปลี่ยนทางที่ผู้คนใช้ติดต่อสื่อสารและเข้าถึงข้อมูล

2.3.2 โปรโตคอลสำหรับการสตรีมมิ่งมีเดีย

การถ่ายทอดข้อมูลมิเดียผ่านเครือข่ายในแบบเรียลไทม์นั้นต้องการระบบเครือข่ายสูง มันง่ายกว่าที่จะชดเชยข้อมูลที่สูญเสียไปมากกว่าชดเชยสำหรับเวลามาก ๆ ในการรับส่งข้อมูล ซึ่งยากมากจากการเข้าใกล้ข้อมูลแบบสตาติก เช่นไฟล์ ที่ซึ่งสิ่งสำคัญที่สุดคือข้อมูลทั้งหมดจะต้องมาถึงจุดหมายของมัน เนื่องจากโปรโตคอลที่ใช้สำหรับข้อมูลแบบสตาติกนั้นไม่มีสำหรับการสตรีมมีเดีย

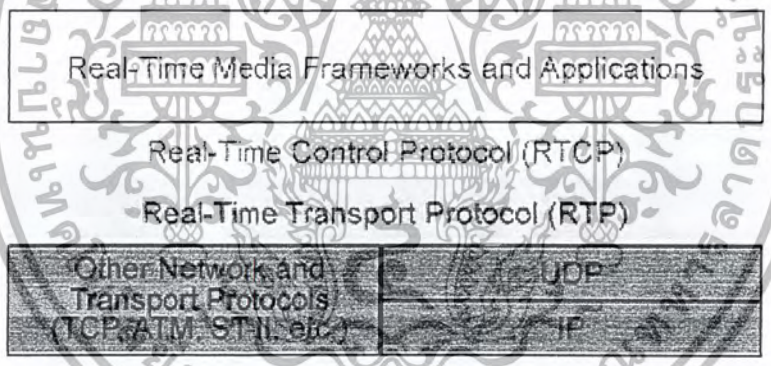
โปรโตคอล HTTP และ FTP เป็นพื้นฐานของ TCP ซึ่งคือ Transport-layer Protocol ที่ออกแบบมาสำหรับการติดต่อสื่อสารข้อมูลที่เชื่อถือได้บนแบนด์วิธที่ต่ำและระบบที่มีอัตราการ

ผิดพลาดสูง เมื่อแพ็คเกตสูญหายหรือถูกรบกวน มันจะต้องส่งส่วนหัว (header) ของการรับประกันการส่งข้อมูลที่เชื่อถือได้ใหม่ จะทำให้ระดับการส่งทุกอย่างช้าลงไป

จากเหตุการณ์นี้เอง โปรโตคอลอื่น ๆ ที่ต่ำกว่า TCP จะถูกนำไปเป็นแบบอย่างใช้สำหรับมิเดียสตรีม อันหนึ่งในนั้นคือใช้ UDP โดยทั่ว ๆ ไปแล้ว UDP คือโปรโตคอลที่ไม่น่าเชื่อถือ มันไม่รับประกันว่าแต่ละแพ็คเกตจะไปถึงจุดหมายของมัน ไม่มีการรับประกันว่าทุกแพ็คเกตจะมาถึงในคำสั่งที่พวกมันถูกส่งมา ผู้รับต้องสามารถชดเชยข้อมูลที่เสียไป (แพ็คเกตที่เหมือนกันทุกประการ) และแพ็คเกตที่มาถึงโดยไม่ได้สั่ง

2.3.3 Real-Time Transport Protocol (RTP)

RTP จัดเตรียมจุดสูงสุดถึงต่ำสุดของระบบการบริการแบบตรงของการถ่ายทอดของข้อมูลแบบเรียลไทม์ RTP คือเครือข่ายและเป็นโปรโตคอลการขนส่งที่เป็นอิสระ ทำให้มีคนใช้งานมากกว่า UDP



รูปที่ 2.12 สถาปัตยกรรมของ RTP

RTP สามารถใช้บนทั้งการบริการเครือข่ายแบบยูนิคาสท์ (unicast network service) และแบบมัลติคาสท์ multicast network services โดยที่การบริการเครือข่ายแบบยูนิคาสท์ สำเนาที่เหมือนกันของข้อมูลจะถูกส่งจากแหล่งไปแต่ละจุดหมาย ในบริการเครือข่ายแบบมัลติคาสท์ ข้อมูลจะส่งถูกส่งจากแหล่งจากแหล่งครั้งเดียวเท่านั้น และระบบจะทำการตัดสินใจอย่างน่าเชื่อถือสำหรับการส่งข้อมูลไปยังหลาย ๆ พื้นที่ การมัลติคาสท์มีประสิทธิภาพมากกว่าสำหรับแอพพลิเคชั่นมัลติคาสท์ (application multicasting) มาก ๆ เช่น วีดีโอ มาตราฐานอินเตอร์เน็ตโปรโตคอล (IP) รองรับการมัลติคาสท์

2.3.4 บริการของ RTP (RTP Service)

RTP จะให้คุณตรวจสอบ ชนิดของข้อมูลที่กำลังส่ง ให้ตัดสินใจว่าจะสั่งให้แพ็คเกตข้อมูลควรจะถูกนำเสนอในอะไร และสตีมมิเดียในเวลาเดียวกันจากแหล่งที่ต่างกัน

แพ็คเกตข้อมูล RTP ยังคงไม่มีการรับประกันว่าจะถึงภายในคำสั่งที่พวกมันถูกส่งไป มันไม่รับประกันว่าจะถึงทั้งหมด มันจะถึงผู้รับเพื่อเรียงลำดับ แพ็คเกตของผู้ส่งใหม่ และตรวจสอบแพ็คเกตที่หายไปโดยการใช้ข้อมูลที่ถูกจัดอยู่ในส่วนหัว ขณะที่ RTP ไม่ได้จัดเตรียมเทคนิคอื่น ๆ ที่จะรองรับการส่งที่ตรงเวลา หรือจัดเตรียมคุณสมบัติอื่น ๆ ของการรับประกันการบริการ มันถูกขยายโดยโปรโตคอลควบคุม (RTCP) ซึ่งให้คุณดูแลตรวจตราคุณสมบัติของการแบ่งข้อมูลออกเป็นส่วน ๆ RTCP มักจะจัดเตรียมการควบคุมและเทคนิคการแยกข้อมูลสำหรับการขนส่ง RTP

2.3.5 สถาปัตยกรรมของ RTP

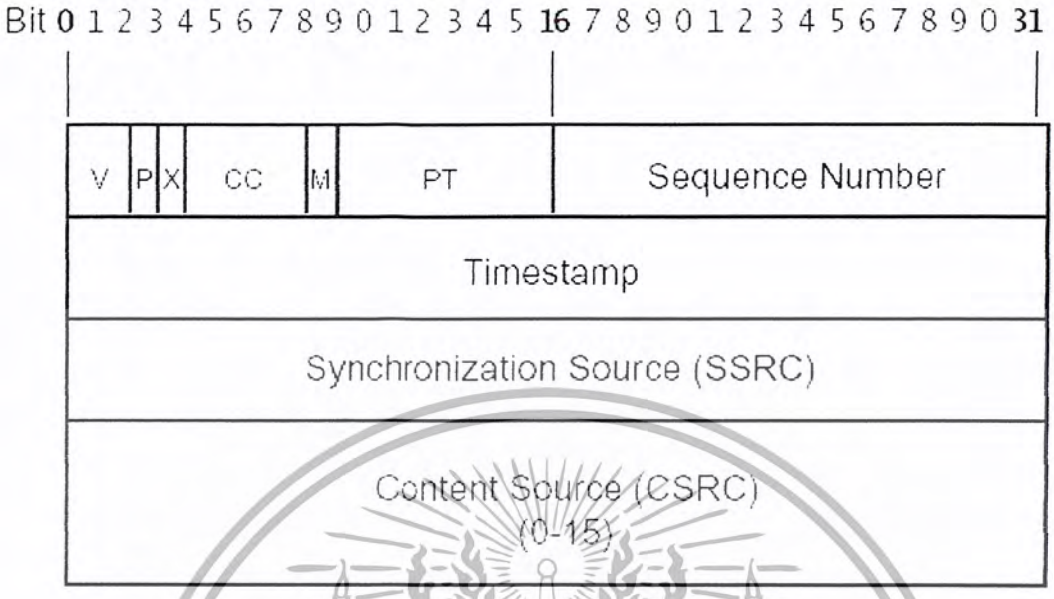
เซสชั่น (session) ของ RTP เป็นการรวมกันระหว่างเซตของการติดต่อแอปพริเคชั่นกับเซสชั่น RTP ถูกตรวจสอบโดยที่อยู่เครือข่าย (network address) และคู่ของพอร์ท โดยที่พอร์ทอันหนึ่งจะถูกใช้สำหรับข้อมูลมีเดียและอันอื่นจะใช้สำหรับข้อมูลควบคุม (RTCP)

ผู้มีส่วนร่วมคือ เครื่องจักรเดี่ยว, โฮสหรือ คนใช้ร่วมในเซสชั่น การใช้ร่วมในเซสชั่นสามารถประกอบด้วยการรับข้อมูลโดยไม่ตอบโต้ (ผู้รับ) การส่งข้อมูล (ผู้ส่ง) หรือทั้งคู่

แต่ละชนิดของมีเดียถูกติดต่อในเซสชั่นที่ต่างกัน ตัวอย่างคือ ถ้าทั้งภาพและเสียงถูกใช้ในคอนเฟอเรน (conference) อันหนึ่ง เซสชั่นอันหนึ่งจะถูกใช้ไปติดต่อข้อมูลเสียงและ เซสชั่นที่เหมือนกัน จะถูกใช้ไปติดต่อข้อมูลภาพ

ผู้มีส่วนร่วมนี้จะเลือกได้ว่า ชนิดของข้อมูลมีเดียในรูปแบบ อันไหนที่พวกมันต้องการจะรับ ตัวอย่างเช่น บางคนที่มีการติดต่อแบบแบนด์วิธต่ำอาจจะต้องได้รับเพียงแพ็คเกตเสียงของข้อมูล

ข้อมูลมิเดียสำหรับเซสชั่นจะถูกส่งเป็นชุดของแพ็คเกตแต่ละอัน ซึ่งชุดข้อมูลนั้นมาจากแหล่งเดียวกันดังเป็น RTP Stream ซึ่งแต่ละ แพ็คเกตข้อมูล RTP ในสตรีมคอนเทนแบ่งเป็น 2 ส่วน คือส่วนหัวกับส่วนข้อมูล



รูปที่ 2.13 รูปแบบส่วนหัวของแพ็คเกต RTP

ส่วนหัวของแพ็คเกตข้อมูล RTP ประกอบด้วย

The RTP version number (V) มี2บิท เวอร์ชั่นปัจจุบันมีค่าเป็น 2

Padding (P) มี 1 บิท ถ้าถูกเซต คือจะมี ในตอนท้ายของแพ็คเกตซึ่งยังไม่ใช่ส่วนของข้อมูลจะมีส่วนของการแพ็ดดิ้งติดมาด้วย ทุก ๆไบต์สุดท้ายของแพ็คเกตจะบอกจำนวนไบต์ของการแพ็ดดิ้ง ซึ่งจะถูกใช้ในอัลกอริธึมการเข้ารหัสบางอัลกอริธึม

Extension (X) มี 1 บิท ถ้าบิทนี้ถูกเซตจะทำให้ส่วนหัวมีส่วนหัวเพิ่มเติม (extension header) ตามมาด้วย การที่มีบิทนี้ทำให้สามารถเพิ่มข้อมูลของส่วนหัวของ RTP ได้

CSRC Count (CC) 4 บิท ตัวเลขของ CSRC Count จะเป็นตัวแสดงจำนวนที่มาของส่วนหัว

Marker (M) มี 1 บิท ใช้เมื่อจำเป็นต้องให้เห็นเป็นพิเศษ

Payload Type (PT) 8 บิท เป็นรายละเอียดในตารางข้อมูลมีเดีย ซึ่งบอกแสดงชนิดข้อมูลว่าถูกเข้ารหัสมาในรูปแบบใด

Sequence Number 16 บิท หมายเลขของแพ็คเกตที่แสดงตำแหน่งของแพ็คเกตนี้ในชุดของแพ็คเกต โดยหมายเลขแพ็คเกตนี้จะเพิ่มขึ้นทีละ 1 สำหรับการส่งแต่ละครั้ง

Timestamp 32 บิท ผลของค่าคงที่การแซมปลิ้น (Sampling) ในเพย์โหลด (payload) แพ็คเกตต่างกันสามารถมีไทม์สแตมป์ (Timestamp) เดียวกันได้ ถ้าถูกสร้างในเวลาเดียวกัน (เช่น ถ้ามันคือส่วนทั้งหมดของเฟรมวีดีโอเดียวกัน)

SSRC 32 บิท จะบอกหมายเลขของแหล่งกำเนิด ถ้ามาจากแหล่งเดียวจะเป็นหมายเลขของแหล่งนั้น ถ้ามาจากหลายแหล่ง จะเป็นหมายเลขของแหล่งใหม่ที่เกิดจากการรวมสัญญาณ

CSRC แต่ละอันมี 32 บิท ใช้แสดงแหล่งข้อมูล ในกรณีที่มาจากหลายแหล่งข้อมูล โดยจะมีได้มากสุด 16 แหล่ง

2.3.6 แพ็คเกตควบคุม (Control Packet)

ข้อมูลมิเดียที่เพิ่มเข้าไปสำหรับ เซสชั่นแพ็คเกตข้อมูลควบคุม (RTCP) จะถูกส่งไปส่วนต่าง ๆ ของเซสชั่น ซึ่งประกอบด้วยข้อมูลเกี่ยวกับ QoS สำหรับเซสชั่นส่วนประกอบ (session participants) ข้อมูลเกี่ยวกับแหล่งที่มาของมิเดียถูกส่งบนพอร์ทข้อมูล (data port)

RTCP Packets มีหลายชนิดด้วยกัน

- Sender Report
- Recevier Report
- Source Description
- Bye
- Application-specific

แพ็คเก็ต RTPC สามารถเก็บได้และถูกส่งเป็น แพ็คเกตที่ประกอบด้วย 2 แพ็คเกตคือ แพ็คเกตรายงาน (report packet) และแพ็คเกตบอกแหล่งที่มา (source description packet) โดยส่วนประกอบทั้งหมดใน เซสชั่นจะส่ง RTCP แพ็คเกต ส่วนประกอบที่ปัจจุบันจะส่งแพ็คเกตข้อมูลในหัวข้อ sender report (SR) ซึ่งจะประกอบไปด้วยจำนวนรวมของแพ็คเกตและไบต์ที่ถูกส่งดีเหมือนกับข้อมูลที่ใช้ไปส่ง synchronize media stream จากเซสชั่นที่ต่างกัน Receiver Report (RR) ประกอบด้วยข้อมูลเกี่ยวกับจำนวนแพ็คเกตที่หายไป จำนวนที่มากที่สุดที่ได้รับ และ timestamp ซึ่งสามารถใช้หาช่วงเวลาระหว่างผู้ส่งและผู้รับได้

RTCPแพ็คเกตที่ผสมมาจะต้องได้รับการรวม source description (SDES) ด้วยซึ่งจะมี canonical name (CNAME) ที่สามารถแสดงแหล่งข้อมูลที่เพิ่มเข้ามาจะต้องรวมในลักษณะของแหล่งเช่น ชื่อของ DataSource, ที่อยู่อิเล็กทรอนิคส์ (email address), ชื่อแอพพลิเคชั่นหรือข้อความที่บอกสถานะปัจจุบันของแหล่งข้อมูล

2.4 การส่งผ่านข้อมูลแบบ RTP มิเดียสตรีม (Transmitting RTP Media Streams)

ในการส่งข้อมูลแบบ RTP Stream จะต้องสร้าง RTP encoded DataSource และทั้ง SessionManager หรือดาต้าซิงค์เพื่อมาควบคุมการส่งข้อมูล

อินพุตที่เข้าไปในโปรเซสเซอร์จะสามารถเป็นได้ทั้งข้อมูลที่เก็บไว้อยู่แล้วหรือข้อมูลที่จับภาพมา สำหรับข้อมูลที่เก็บไว้อยู่แล้วนั้น เราสามารถ ใช้ MediaLocator เพื่อตรวจสอบไฟล์ในขณะสร้างโปรเซสเซอร์ ได้ แต่สำหรับข้อมูลที่จับสัญญาณภาพมานั้น เราจะใช้แหล่งข้อมูลการจับภาพ (capture DataSource) เป็นอินพุทเข้าไปในโปรเซสเซอร์

มี 2 ทางที่จะส่งข้อมูล RTP Stream ได้คือ

1. ใช้ MediaLocator ซึ่งมีพารามิเตอร์ของ RTP เซสชั่น เพื่อสร้าง RTP DataSink โดยการเรียก Manager.createDataSink
2. ใช้เซสชั่น Manager เพื่อสร้างช่องทางการส่งสำหรับแสดงและควบคุมการส่งข้อมูล

.

ถ้าใช้ MediaLocator เพื่อสร้าง RTPDataSink เราจะสามารถส่งได้แค่สตรีมแรกใน DataSoure เท่านั้น ถ้าอยากจะส่งแบบหลาย ๆ สตรีมในเซสชั่นเดียวหรือต้องการเซสชั่นอื่น ๆ เราจำเป็นต้องใช้ SessionManager โดยตรง

สิ่งที่ต้องทำในการส่งข้อมูล RTP Stream คือ

1. สร้างโปรเซสเซอร์ด้วยแหล่งข้อมูลที่ทำหน้าที่เป็นข้อมูลที่ต้องการจะส่ง
2. ปรับแต่งโปรเซสเซอร์ให้มีเอาท์พุตเป็นข้อมูล RTP ที่เข้ารหัสแล้ว
3. รับเอาท์พุตจากโปรเซสเซอร์เป็น DataSource

2.4.1 การปรับแต่งโปรเซสเซอร์

เพื่อที่จะปรับแต่งให้โปรเซสเซอร์สามารถเข้ารหัสข้อมูลได้ จำเป็นต้องกำหนดรูปแบบของ RTP-specific สำหรับแต่ละแทรคและระบุรายละเอียดของเอาท์พุตที่ต้องการ รูปแบบแทรคถูกเซตโดยการรับ TrackControl สำหรับแต่ละแทรคและการเรียก setFormat เพื่อระบุรูปแบบของ RTP-specific โดยที่ RTP-specific อันนึงจะถูกเลือกโดยการเซตรูปแบบ encoding string ไปเป็น RTP-specific string ดังเช่น " AudioFormat.GSM_RTP " โปรเซสเซอร์จะพยายามโหลดปลั๊กอินซึ่งรองรับรูปแบบนี้ ถ้าไม่สามารถหาปลั๊กอินที่รองรับได้ รูปแบบ RTP นั้นจะไม่สามารถรองรับได้และ UnSupportedFormatException จะถูกส่งผ่านไป

รูปแบบเอาท์พุตถูกเซตโดยเมธอด setOutputContentDescriptor ถ้าไม่จำเป็นต้องใช้ การมัลติเพล็กซึ่งแบบพิเศษ Output content descriptor จะสามารถเซตเป็น ContentDescriptor.RAW ได้

ซึ่งสตรีมของเสียงและภาพจะไม่ถูกแทรก ถ้าแทรคของโปรเซสเซอร์ประกอบด้วยมิเดียที่ต่างชนิดกัน แต่ละมีเดียสตรีมจะถูกส่งแยกเป็นเซสชั่นของRTP

การดึงข้อมูลเอาท์พุตของโปรเซสเซอร์

ครั้งหนึ่งรูปแบบของแทรคของโปรเซสเซอร์ถูกเซตและโปรเซสเซอร์ถูกรีไรซ์ DataSource ที่เอาท์พุตจากโปรเซสเซอร์สามารถดึงได้ เราจะดึงข้อมูลเอาท์พุตของโปรเซสเซอร์มาเป็นแหล่งข้อมูลโดยการเรียกใช้ get DataOutput โดยที่แหล่งข้อมูลที่ถูกดึงกลับมาสามารถเป็นได้ทั้ง PushBufferDataSource หรือ PullBufferDataSource ขึ้นอยู่กับsourceของข้อมูล โดยเอาท์พุตของ DataSource ถูกติดต่อ SessionManager ที่ใช้เมธอด CreateSendStream เซสชั่นของเมเนเจอร์ต้องถูกเตรียมการเริ่มต้นก่อนที่เราจะสร้างช่องทางการส่ง ถ้าแหล่งข้อมูลประกอบด้วย SourceStream หลากหลายแบบ แต่ละ SourceStream นั้นจะถูกส่งเป็น separate RTP stream ถ้าไม่ใช่เซสชั่นเดียวกันก็เซสชั่นที่แตกต่างกัน ถ้าแหล่งข้อมูลบรรจุด้วยทั้งสตรีมภาพและเสียง separate RTP session จะต้องถูกสร้างมาสำหรับภาพและเสียง เราสามารถลอกแบบแหล่งข้อมูลและส่งตัวที่ลอกแบบมานั้นออกไปเป็น RTP Stream ที่ต่างกันในเซสชั่นเดียวกัน หรือเซสชั่นที่ต่างกันก็ได้

2.4.2 การควบคุมแพคเกตดีเลย์ (Packet Delay)

แพคเกตดีเลย์ (ที่รู้จักกันในชื่อ Packetization interval) คือเวลาใช้ไปในการส่งแต่ละแพคเกตผ่านระบบเครือข่าย แพคเกตดีเลย์หมายถึงช่วงดีเลย์น้อยที่สุดตั้งแต่เริ่มถึงสิ้นสุด ซึ่งจะยาวกว่าการส่งแพตเกต RTP Media Stream 147 ภาครับควรจะตอบรับแพคเกตระหว่าง 0 ถึง 200 มิลลิเซคคั่นสำหรับข้อมูลเสียง (ถ้าเป็นเฟรมเสียงที่ถูกเข้ารหัสมา ภาครับควรจะตอบรับภายใน 200 มิลลิเซคคั่นหารด้วยระยะทาง) ทำให้ส่งผลต่อขนาดของบัฟเฟอร์ของภาครับ แต่ละการเข้ารหัสแพคเกตจะมีมาตราฐานการดีเลย์ที่เหมาะสมของมัน

ถ้าตัวเข้ารหัสอนุญาตให้มีการปรับแต่งระยะเวลาได้ มันจะต้องเรียกใช้ PacketSizeControl ซึ่งระยะเวลาแพคเกตดีเลย์จะถูกเปลี่ยนแปลงได้ หรือเซตได้โดยผ่านเมธอด setPacketSize

สำหรับสตรีมภาพนั้น เฟรมภาพแต่ละอันจะถูกส่งไปในแพคเกต RTP แบบมัลติเปิล (multiple RTP packet) ขนาดของแต่ละแพคเกตนั้นจะถูกจำกัดโดย Maximum Transmission Unit (MTU) ของเครือข่ายที่ต่ำกว่า พารามิเตอร์นี้จะถูกกำหนดการใช้เมธอด setPacketSize ใน PacketSizeControl ของตัวเข้ารหัส

2.4.3 การส่งข้อมูล RTP ด้วยดาต้าซิงค์

ทางที่ง่ายที่สุดที่จะส่งข้อมูลแบบ RTP คือสร้าง ดาต้าซิงค์ของ RTP โดยใช้เมธอด Manager.createDataSink เราผ่านเข้าไปในเอาพุตแหล่งข้อมูลจากโปรเซสเซอร์และ MediaLocator ที่บอกเซสชั่น RTP ที่แหล่งข้อมูลถูกสตรีมไป (MediaLocator จะบอกแอดเดรสและพอร์ทของ RTP session) เพื่อที่จะควบคุมการส่งข้อมูล เราจะเรียกใช้การเริ่มและหยุดบนดาต้าซิงค์ PRST Stream ในแหล่งข้อมูลเท่านั้นที่ถูกส่งไป

การส่งข้อมูลแบบRTP โดยใช้เซสชั่น Manager

โพสเซสพื้นฐานสำหรับการส่งคือ

1. สร้างโพรเซสเซอร์ JMF และกำหนดรูปแบบของแต่ละแทรคที่ RTP-specific format
2. ดึงข้อมูลเอาท์พุตแหล่งข้อมูลจากโพรเซสเซอร์
3. เรียกใช้ creatSendStream บนเซสชั่นเมเนเจอร์ที่สร้างและเตรียมการมาก่อนหน้า ผ่านไปในแหล่งข้อมูลและรายละเอียดสตรีม เซสชั่นเมเนเจอร์จะสร้าง SendStream สำหรับระบุ SourceStream
4. เริ่มเซสชั่นเมเนเจอร์โดยเรียก SessionManager startSession
5. ควบคุมการส่งข้อมูลโดยผ่านเมธอด SendStream โดยมี SendStreamListener สามารถเป็นรีจิสเตอร์(Register) ที่คอยฟังเหตุการณ์บน SendStream

2.4.4 การสร้าง SendStream

ก่อนที่เซสชั่นเมเนเจอร์จะส่งข้อมูลได้นั้น มันจำเป็นต้องรู้ว่าจะได้ข้อมูลจากที่ไหน เมื่อเราสร้างSendStreamอันใหม่ เราจะดึง SessionManager และแหล่งข้อมูลจากอันที่ได้รับข้อมูล ตั้งแต่แหล่งข้อมูลสามารถรวมสตรีมหลายอันได้ เราจำเป็นต้องบอกรายละเอียดของสตรีมนั้นเพื่อส่งมาที่เซสชั่นนี้ ซึ่งการทำเช่นนี้เราสามารถสร้างหลายสตรีมการส่งได้โดยการผ่านแหล่งข้อมูลที่ต่างกันไป creatSendStream หรือโดยการระบุรายละเอียดสตรีมต่างกัน

เซสชั่นเมเนเจอร์เรียกใช้ฟอร์เมตของ SourceStream เพื่อเลือก ถ้ามันมีเพลย์โหลดชนิดที่ใช้สำหรับรูปแบบนี้ ถ้ารูปแบบของข้อมูลไม่ใช่แบบ RTP หรือชนิดของเพลย์โหลดไม่สามารถลงในแบบ RTP ได้ซึ่งจะมี UnSupportedFormatException ผ่านเข้าไปกับข้อความด้วย เพลย์โหลดแบบไดนามิค (Dynamic Playload) สามารถเชื่อมโยงกับ RTP format ที่ใช้เมธอด SessionManager addFormat

2.4.5 การใช้แหล่งข้อมูลที่สามารถลอกแบบได้ (Cloneable DataSource)

เนื่องจากการส่งเป็นแบบมัลติเปิล RTP เซสชั่น (multiple RTP session) นั้น สตรีมที่ใช้ในการจะต้องเป็นแบบมัลติเปิลฟอร์เมต (multiple format) หรือเป็นแบบที่เข้ารหัสสตรีมให้เป็นแบบมัลติเปิลฟอร์เมตก่อนแล้วจึงส่งข้อมูล ถ้าหากเราต้องการส่งสตรีมที่ถูกเข้ารหัสเป็นแบบซิงเกิลฟอร์เมต (single format) เราจำเป็นต้องลอกแบบแหล่งข้อมูลที่เอาท์พุตออกมาจากโพรเซสเซอร์จากอันที่ข้อมูลถูกจับสัญญาณมา ซึ่งทำได้โดยการสร้างแหล่งข้อมูลที่สามารถลอกแบบได้ผ่าน Manager และการเรียกใช้ getClone บนแหล่งข้อมูลนั้น โพรเซสเซอร์อันใหม่นี้จะสามารถสร้างการส่งผ่านข้อมูล RTP Media Stream 151 จากแต่ละแหล่งข้อมูลที่ถูกออกแบบมาได้ แทรคของข้อมูลนั้นจะถูกเข้ารหัสในรูปแบบที่ต้องการและสตรีมอาจถูกส่งออกไปบนเซสชั่นของ RTP

2.4.6 การควบคุมสตรีมที่ถูกส่งไป (Controlling a Send Stream)

เราจะใช้เมธอด RTPStream start และ stop เพื่อควบคุม SendStream การเริ่มทำให้ SendStream เริ่มต้นส่งข้อมูลบนเครือข่ายและการหยุดจะเป็นตัวที่ชี้ทางให้หยุดส่งข้อมูล ในการเริ่มต้นการส่งข้อมูล SendStream แต่ละตัวจะต้องถูกสั่งให้เริ่มทำงาน ซึ่งคำสั่งเริ่มหรือหยุดทำงานนั้นจะเหมือน ๆ กันในแหล่งข้อมูลของตัวมันเอง แต่ถ้าแหล่งข้อมูลเริ่มทำงานเองในขณะที่ใน SendStream เป็นคำสั่งหยุด ข้อมูลนั้น ๆ จะถูกทิ้ง (PushBufferDataSource) หรือไม่ถูกดึงออกมา (PullBufferDataSource) โดยเซสชั่นแมเนเจอร์ (session manager) ในเวลาระหว่างนี้จะไม่มีข้อมูลใด ๆ ถูกส่งออกไปบนเครือข่ายเลย

2.5 การสร้าง RTPManager

แมเนเจอร์ของ RTP แบ่งออกได้เป็น 3 แบบ ซึ่งมีวิธีการสร้างอธิบายได้ดังนี้

2.5.1 แมเนเจอร์แบบที่มีเซสชั่นแบบยูนิคาสท์ (Unicast Session) เปรียบเสมือนแมเนเจอร์ส่งผ่านเราต์เตอร์และ ติดต่อไปยังไคลเอ้นท์ที่ต้องการ

ตัวอย่าง

```
Import java.net.*;
Import javax.media.rtp.*;
//สร้าง RTP Manager
RTPManager rtpManager = RTPManager.newInstance();
```

```
//สร้างโลคอลเอนพอยต์ระบุแอดเดรส ของเมเนเจอร์ที่สร้าง
SessionAddress localAddress = new SessionAddress();

//ทำการ initialize RTPManager
rtpManager.initilize(localAddress);

//ใช้ ReceiveStreamListener ถ้าต้องการรับข้อมูล และทำแอปพริเคชั่นอื่น ๆ
...
//ระบุแอดเดรสที่ต้องการส่งไปของเซสชั่นแบบยูนิคาสท์นี้
//โดยที่สตริงของแอดเดรสและพอร์ทที่เห็นสามารถเปลี่ยนได้ตามแต่ว่าจะส่งไปที่ไหน
InetAddress ipAddress = InetAddress.getByName("168.1.2.3");
SessionAddress remoteAddress = new SessionAddress(ipAddress,3000);
//เริ่มต้นเปิดการติดต่อ
rtpManager.addTarget(remoteAddress);

//สร้าง SendStream สำหรับเอาท์พุตที่ออกมาจากโปรเซสเซอร์
DataSource dataOutput = createDataSource();
SendStream sendStream = rtpSession.createSendStream(dataOutput,1);

//เริ่มSendStream
sendStream.start();

//ส่งข้อมูลและทำแอปพริเคชั่นอื่น ๆ ตามที่ระบุ
...
//ปิดการติดต่อ
rtpManager.removeTarget(remoteAddress,"client disconnected.");
//เรียก dispose ที่จุดสิ้นสุดของวงจร RTPManagerนี้
rtpManager.dispose();
```

2.5.2 เมเนเจอร์แบบที่มีเซสชั่นแบบมัลติยูนิคาสท์ (Multi-Unicast Session) คล้ายกันกับแบบยูนิคาสท์ เหมือนกันในตอนแรกแต่จะแตกต่างกันตรงที่หลังจากการสร้าง sendStream และเริ่มต้นมันแล้วเราจะสามารถเพิ่มแอดเดรสที่จะส่งไปได้อีกตามแต่ที่เราจะใช้ addTarget เพิ่มเข้าไป โดยการใช้

```
addTarget(remoteAddress2);
addTarget(remoteAddress3);
```

2.5.3 เมเนเจอร์แบบที่มีเซสชั่นแบบมัลติคาสท์ (Multicast Session) การสร้างเซสชั่นแบบนี้จะคล้ายกันกับตัวอย่างแบบยูนิคาสท์ แต่แทนที่การระบุตำแหน่งและบังคับไปสู่จุดหมายด้วยแอดเดรสของมัลติคาสท์เซสชั่น ซึ่งจำเป็นต้องถูกสร้างและส่งผ่านไปยังการ initialize และเรียกใช้ addTarget นอกนั้นจะเหมือนกับตัวอย่างแบบยูนิคาสท์ ดังเช่น

```
//...
//สร้างมัลติคาสท์แอดเดรสสำหรับ 224.1.1.0 และพอร์ท 3000/3001
IPAddress = InetAddress.getByName("224.1.1.0");
SessionAddress multiAddress = new SessionAddress(ipAddress,3000);

//ทำการ initialize RTPManager
rtpManager.initialize (multiAddress);
//เพิ่มจุดหมายในการส่ง
rtpManager.addTarget (multiAddress);

//...
```

บทที่ 3
การออกแบบ

3.1 ข้อมูลเบื้องต้น

จากการศึกษาลักษณะการทำงานและการใช้งานของไอพีทีวี เราสามารถแบ่งการทำงานออกได้เป็น 2 ฝั่ง คือ ฝั่งเซิร์ฟเวอร์ ที่จะต้องสามารถเลือกข้อมูลที่จะส่ง และฝั่งไคลเอ้นท์ ที่จะต้องสามารถเลือกไอพีและพอร์ทเพื่อเลือกรับชมรายการได้

3.1.1 เซิร์ฟเวอร์

ออกแบบให้มีรูปแบบการใช้งานดังนี้

- หน้าจอแสดงการทำงานในขณะนั้น
- แถบ RTP Transfer ใช้ในการเลือกระบุไอพีและพอร์ทที่จะบอร์ดคาสท์ไป
- ช่องระบุไอพีที่ต้องการจะบอร์ดคาสท์
- ช่องระบุพอร์ทที่ต้องการจะบอร์ดคาสท์
- แถบ Add Media ใช้ในการเลือกไฟล์ที่มีอยู่ในเครื่อง
- ปุ่ม Browse ใช้ค้นหาข้อมูลที่ใช้ในการหาไฟล์เป็นข้อมูลในการส่ง
- ปุ่ม add ใช้เลือกไฟล์ที่จะทำการบอร์ดคาสท์
- ปุ่ม OK ใช้ในการยืนยันไฟล์ที่เลือก, ไอพีและพอร์ทที่จะทำการบอร์ดคาสท์
- แถบ Device ใช้ในการเลือกส่งรายการที่ดึงข้อมูลมาจากการ์ดทีวี
- ปุ่ม Detect ใช้ในการเลือกดึงข้อมูลจากการ์ดทีวี
- ปุ่ม Start ใช้ในการเริ่มการส่งข้อมูลที่ได้จากการ์ดทีวี
- ปุ่ม Stop ใช้ในการหยุดการส่งข้อมูลที่ได้จากการ์ดทีวี
- ปุ่ม Send ใช้ในการเริ่มส่งข้อมูลที่ได้จากไฟล์
- ปุ่ม Stop ใช้ในการหยุดการส่งข้อมูลที่ได้จากไฟล์
- ตารางแสดงรายการทีวีที่ได้ทำการส่งในขณะนั้น
- ปุ่ม Update Program สำหรับอัพเดตตารางรายการทีวี

3.1.2 ไคลเอ้นท์

ออกแบบให้มีรูปแบบการใช้งานดังนี้

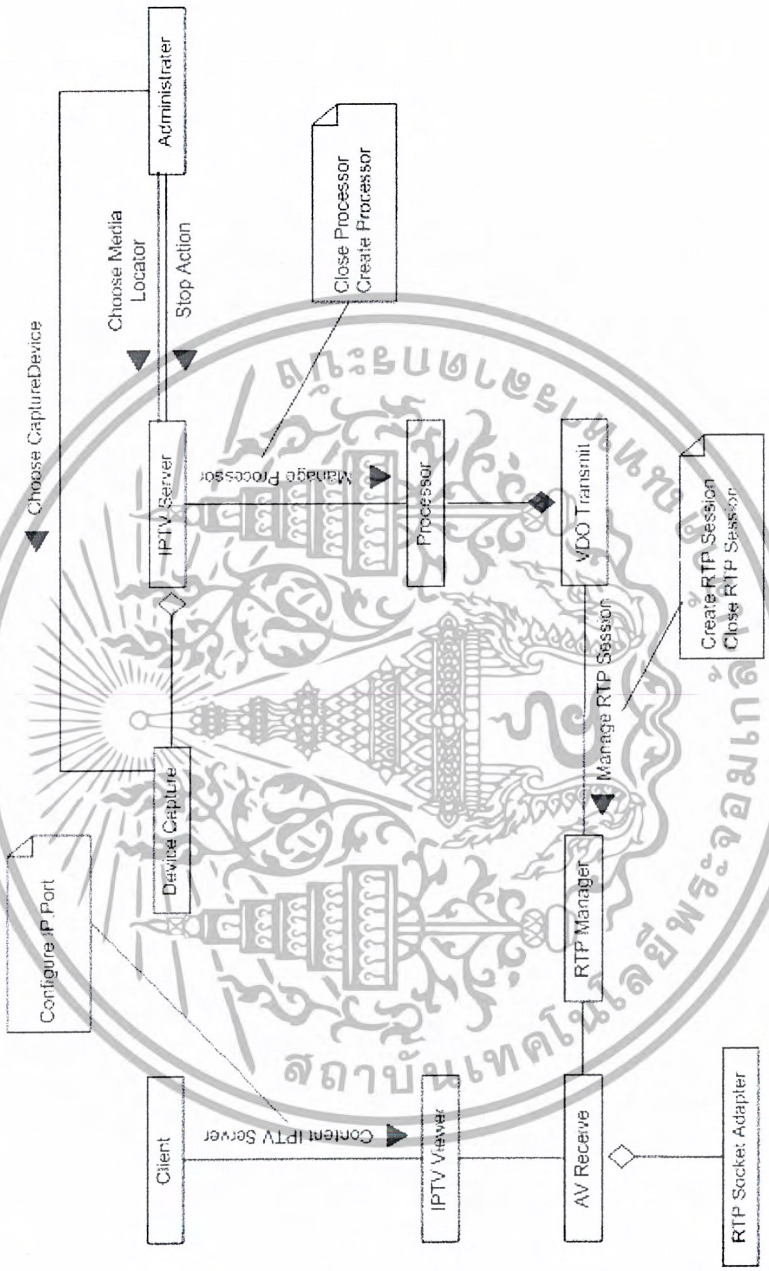
- แถบระบุไอพีที่ใช้ติดต่อเพื่อรับข้อมูล
- แถบระบุพอร์ทที่ใช้ติดต่อเพื่อรับข้อมูล
- ปุ่ม Connect ใช้ในการเริ่มการรับข้อมูล
- ตารางรายการทีวี
- ปุ่ม Update Program ใช้อัพเดตรายการทีวีจากเซิร์ฟเวอร์
- หน้าจอแสดงผลข้อมูลที่ได้รับมา

3.2 การออกแบบระบบ โดยอาศัยการวิเคราะห์เชิงวัตถุ

แนวความคิดของการเขียนโปรแกรมแบบ OOP (Object Oriented Programming) จะประกอบด้วยข้อกำหนด 3 ประการคือ

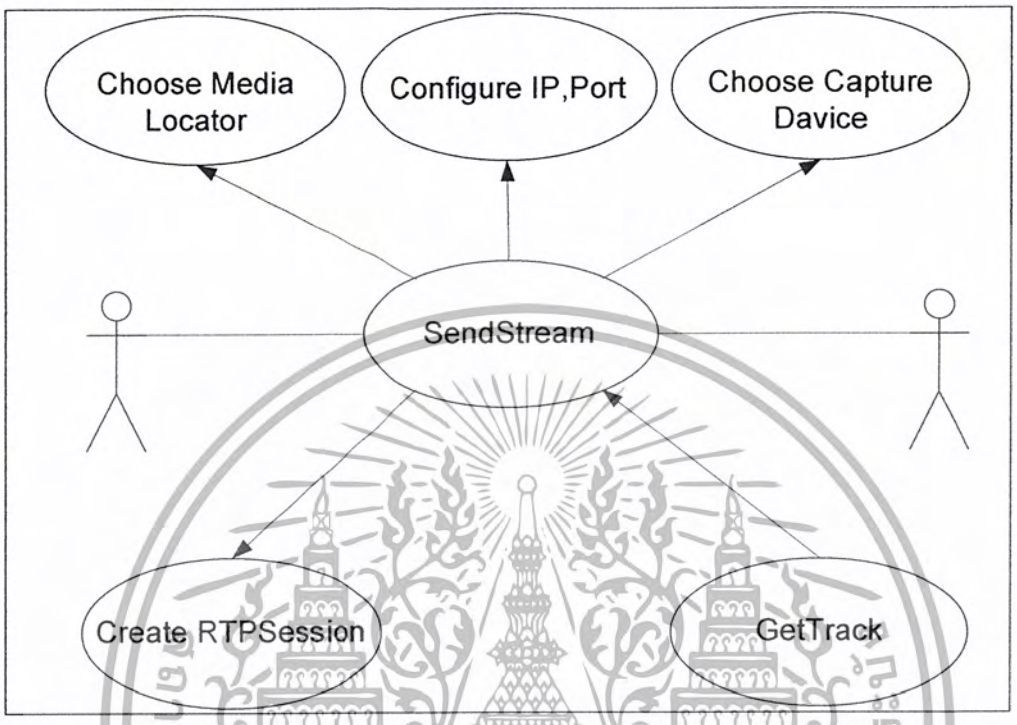
- Encapsulation ระบุว่าคลาสใด ๆ ก็ตามที่สร้างขึ้นมา ผู้ใช้งานไม่จำเป็นต้องรู้ว่าภายในคลาสมีกลไกอย่างไรบ้าง
- Inheritance คือการสืบทอดคลาส คลาสลูกจะมีความสามารถทุกอย่างเหมือนคลาสแม่ ซึ่งทำสำเนาตัวเองได้
- Polymorphism คลาสที่ถูกสำเนาขึ้นใหม่ สามารถแก้ไข เปลี่ยนแปลงความสามารถของคลาสนั้นได้ด้วย

3.2.1 ไคลเอ้นท์อินเทอร์วิว

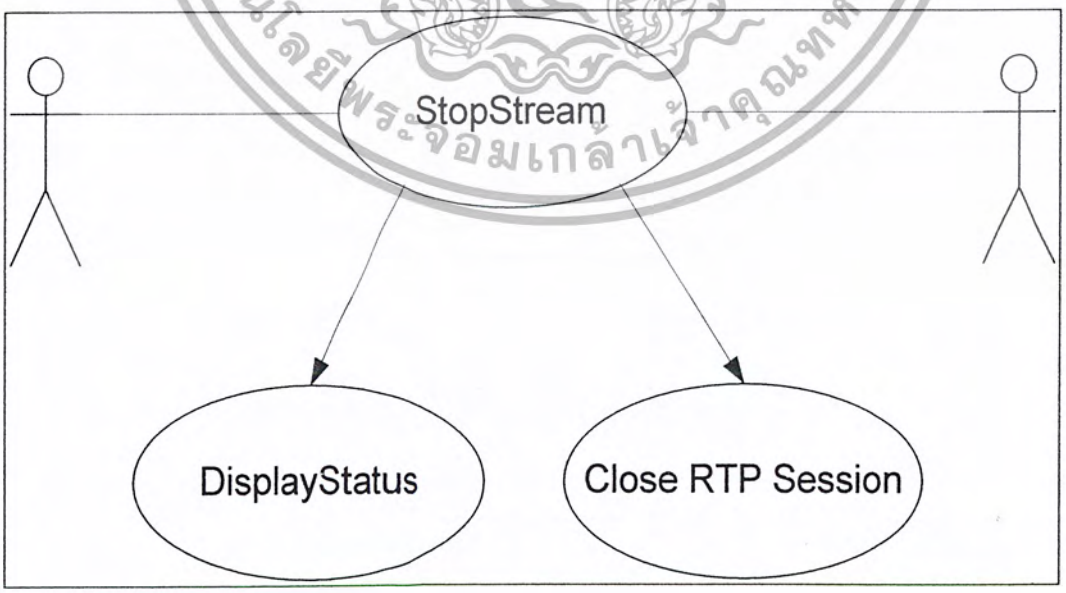


รูปที่ 3.1 ไคลเอ้นท์อินเทอร์วิว

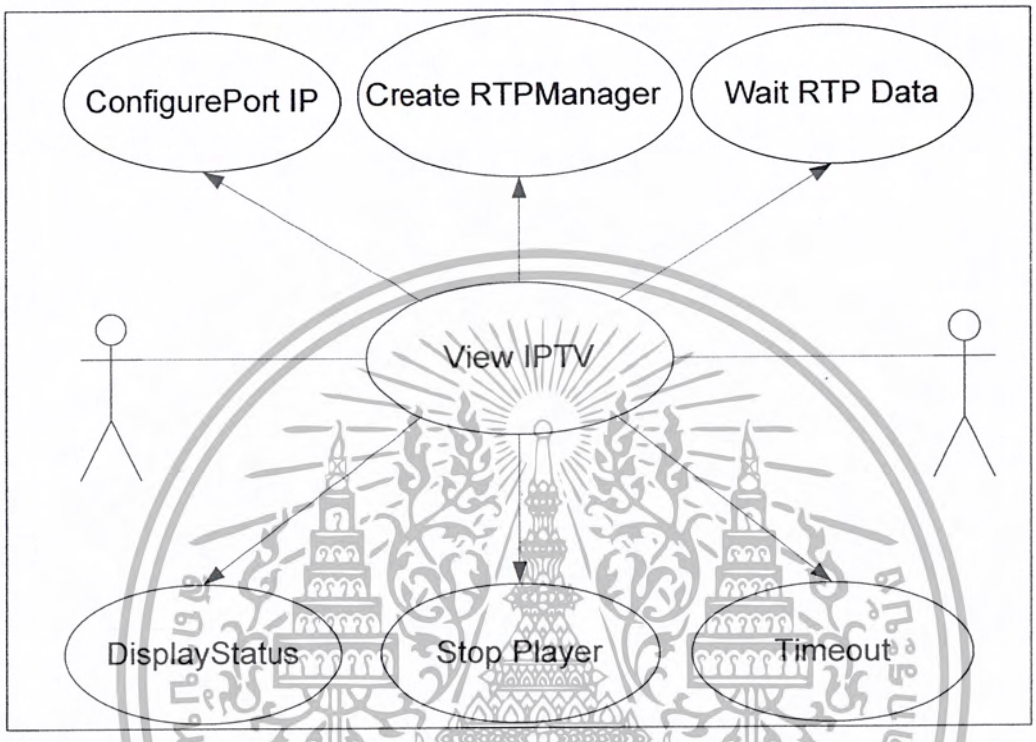
3.2.2 ยูสเคสไดอะแกรม



รูปที่ 3.2 ยูสเคสไดอะแกรม (1)

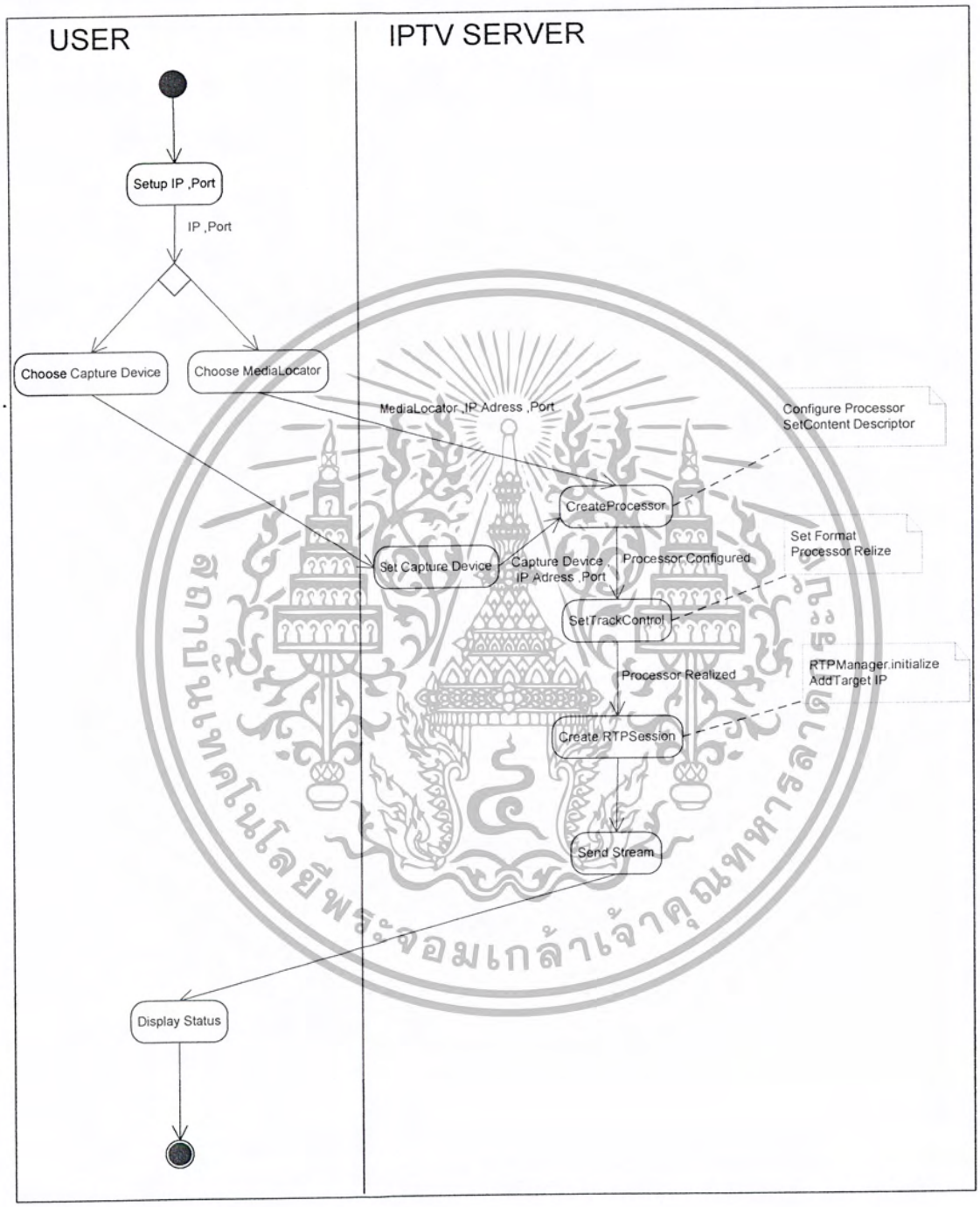


รูปที่ 3.3 ยูสเคสไดอะแกรม (2)

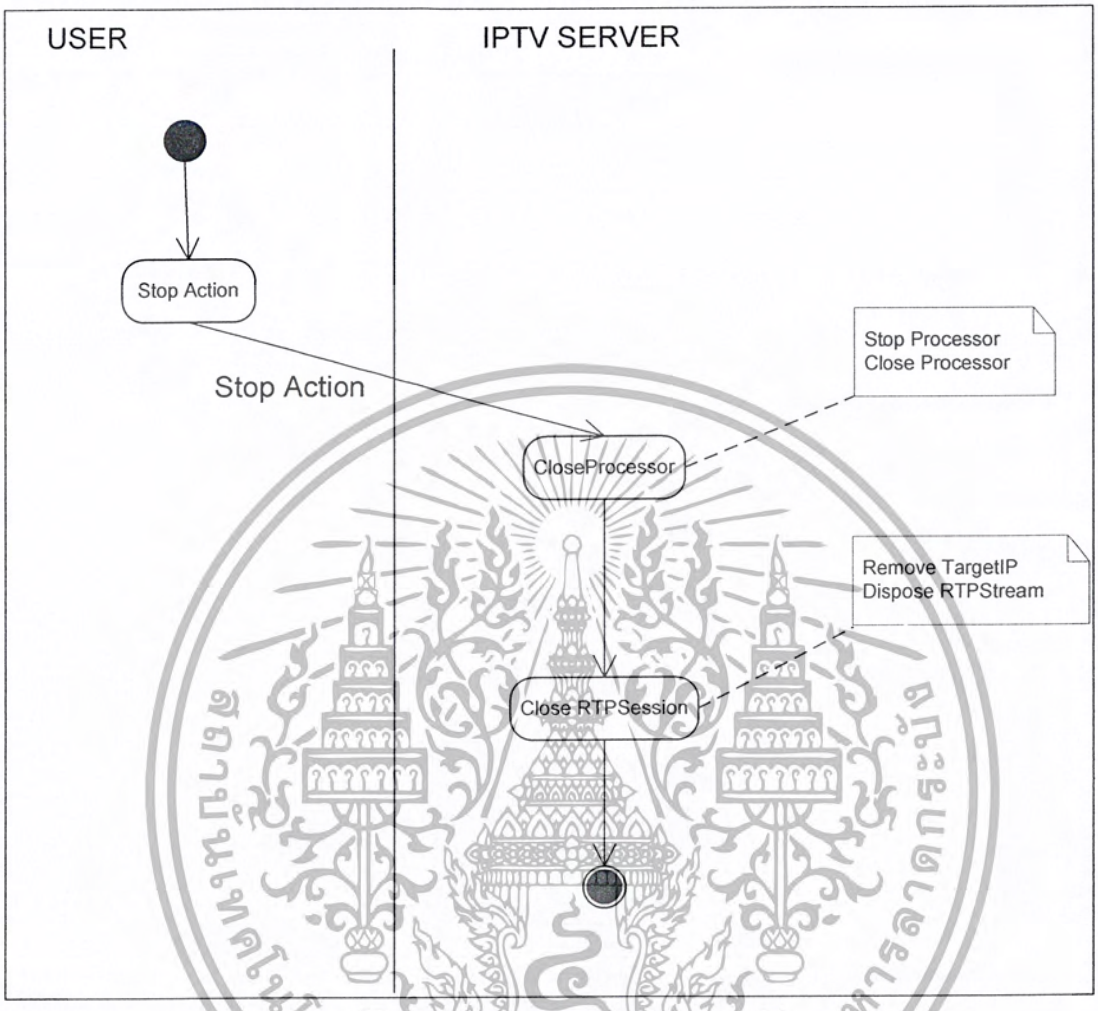


รูปที่ 3.4 ยูสเคส ไดอะแกรม (3)

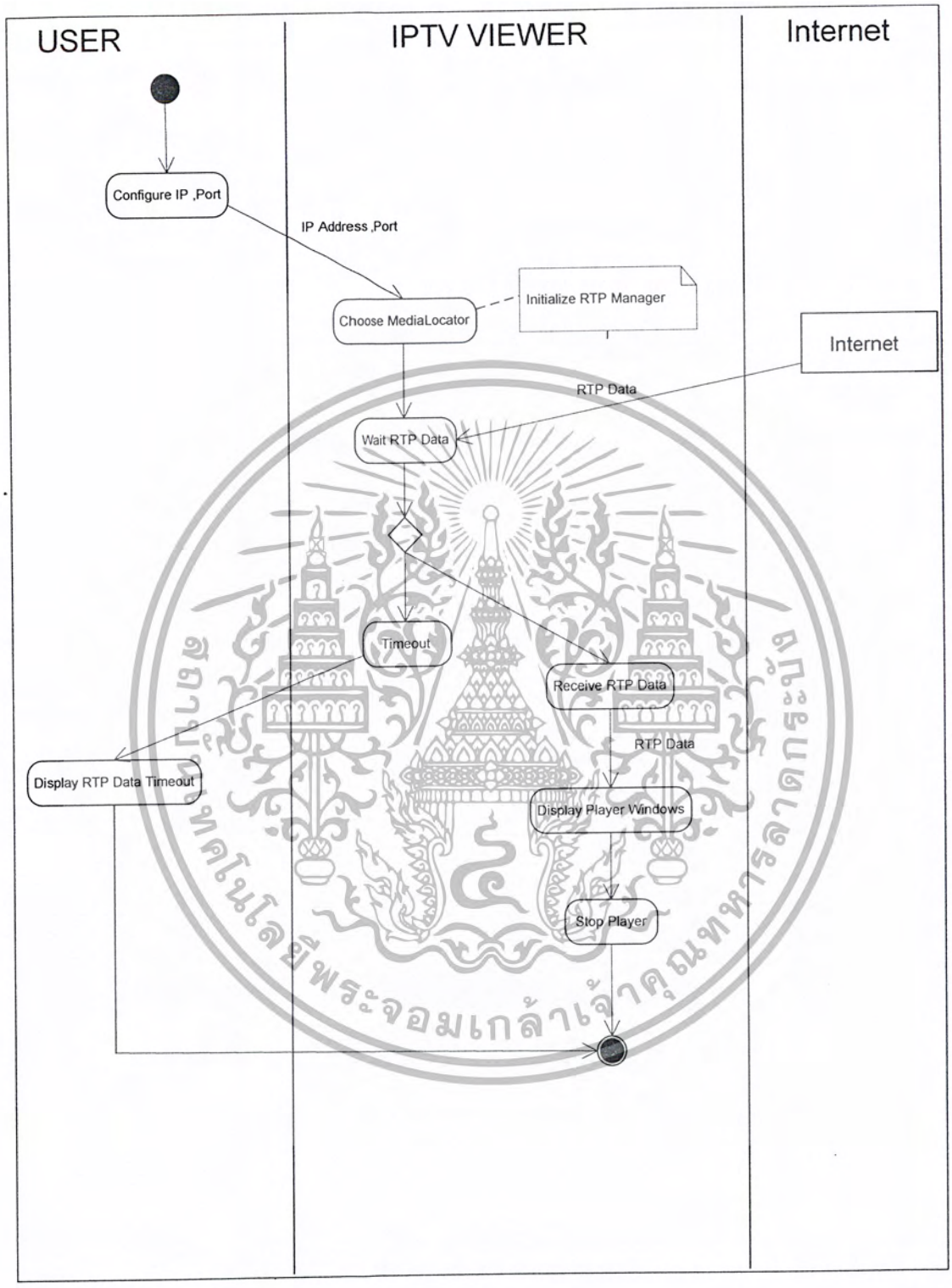
3.2.3 แอคทิวิตี้ไดอะแกรม



รูปที่ 3.5 แอคทิวิตี้ไดอะแกรม SendStream



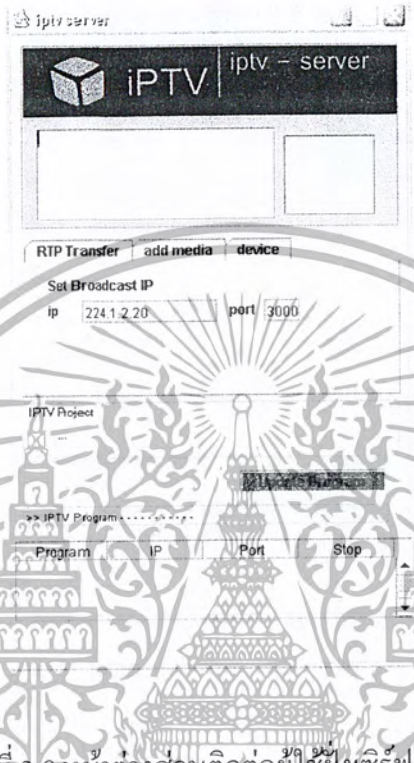
รูปที่ 3.6 แอคทิวตี้ไดอะแกรม StopStream



รูปที่ 3.7 แอคทิวิตี้ไดอะแกรม Viwe IPTV

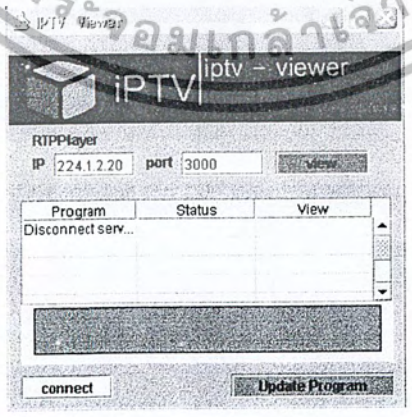
3.2.5 การออกแบบส่วนติดต่อผู้ใช้ (User Interface)

3.2.5.1 ส่วนติดต่อผู้ใช้ฝั่งเซิร์ฟเวอร์



รูปที่ 3.8 หน้าต่างส่วนติดต่อผู้ใช้ฝั่งเซิร์ฟเวอร์

3.2.5.2 ส่วนติดต่อผู้ใช้ฝั่งไคลเอ้นท์



รูปที่ 3.9 หน้าต่างส่วนติดต่อผู้ใช้ฝั่งไคลเอ้นท์

บทที่4

ผลการทดลอง

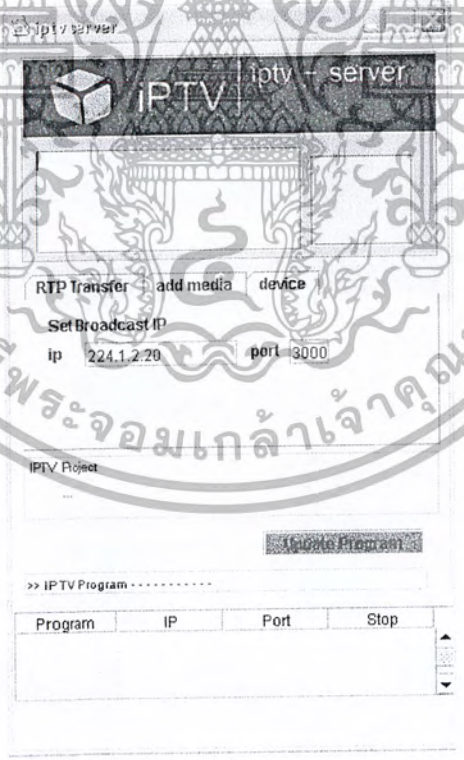
การใช้งานโปรแกรมประยุกต์ในโครงงานนี้ สามารถแบ่งผู้ใช้งานออกได้เป็น 2 ฝั่ง ซึ่งจะมีลักษณะการใช้งานและเครื่องมือที่ใช้งานแตกต่างกันออกไป คือ

1. การใช้งานของโปรแกรมฝั่งเซิร์ฟเวอร์
2. การใช้งานของโปรแกรมฝั่งไคลเอนท์

ดังนั้นการทดลองและผลลัพธ์ที่ได้จะแบ่งเป็น 2 ส่วนตามการใช้งาน การทดสอบได้ทำในระบบปฏิบัติการบนวินโดว์เอ็กซ์พี

4.1 การใช้งานของโปรแกรมฝั่งเซิร์ฟเวอร์

เมื่อรันโปรแกรม จะปรากฏหน้าต่างการทำงานของโปรแกรมดังรูปที่ 4.1



รูปที่ 4.1 หน้าต่างหลักของโปรแกรมฝั่งเซิร์ฟเวอร์

ในรูปที่ 4.1 ส่วนบนสุดจะมีหน้าต่างแสดงการทำงานของโปรแกรมในขณะนั้น ดังในรูปตัวอย่างนี้เป็นขณะที่โปรแกรมกำลังส่งผ่านข้อมูลมีเดีย จากไฟล์ D:\temp\Ladie.mpg ไปที่แอดเดรส 224.1.2.20 พอร์ท 3000

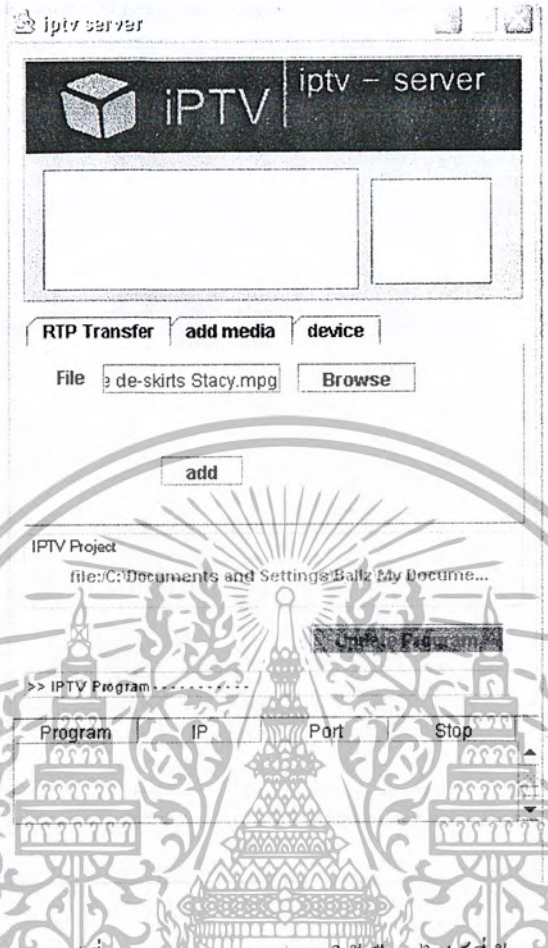
ส่วนกลางจะมีแถบให้เลือก RTP Transfer จะเป็นการเลือกเซตไอพีและพอร์ทที่ต้องการจะส่ง แถบ add media จะเป็นแถบสำหรับเลือกรายการที่ต้องการจะส่งไป โดยจะสามารถเลือกได้จากไฟล์ที่มีอยู่แล้ว

และส่วนล่างจะเป็นส่วนของปุ่มควบคุมการทำงาน มีปุ่มเริ่มและหยุดการทำงาน

4.1.1 การบอร์ดคาสท์รายการทีวีโดยเลือกจากไฟล์ที่มีอยู่แล้ว

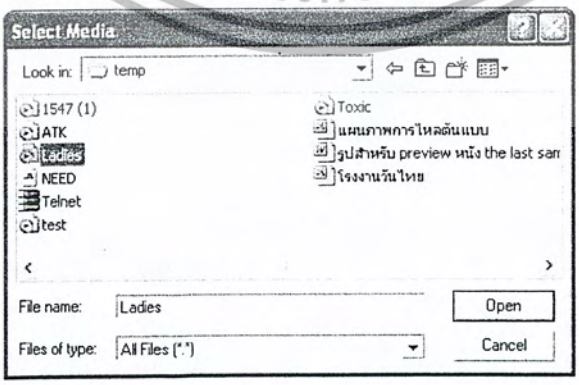
. เป็นการเลือกรายการจากโฟลเดอร์ที่มีอยู่แล้ว โดยการป้อนข้อมูลที่อยู่ของไฟล์ หรืออาจจะใช้การค้นหา ซึ่งทำได้โดย

- เลือกแถบ RTP Transfer ในส่วนกลางของหน้าหลัก แล้วระบุไอพีและพอร์ทที่ต้องการจะบอร์ดคาสท์ไป
- เลือกแถบ add Media ในส่วนกลางของหน้าจอหลัก ดังรูปที่ 4.2



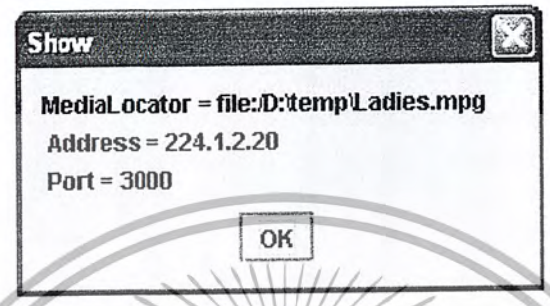
รูปที่ 4.2 แถบ add media ใช้เลือกไฟล์ที่ต้องการจะส่ง

- กดปุ่ม Browse เพื่อเลือกหาไฟล์ที่ต้องการจะส่ง โดยเลือกได้เฉพาะไฟล์ที่เป็นมัลติมีเดีย หากเลือกไฟล์ชนิดอื่นอาจเกิดการผิดพลาด



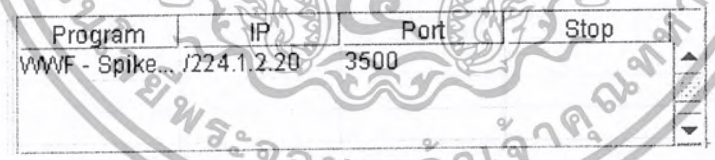
รูปที่ 4.3 การหาไฟล์ที่ต้องการจะส่ง เมื่อกดปุ่ม Browse

- เมื่อเลือกไฟล์ได้แล้ว กดปุ่ม Open จะปรากฏหน้าต่างแสดงผลว่าเลือกไฟล์ใดและต้องการส่งไปที่ไอพีไหน พอร์ทอะไร ดังรูป4.4 เป็นตัวอย่างหน้าต่างที่ขึ้นมา เมื่อเลือกไฟล์ที่ c:\temp\Ladies.mpg และต้องการบอร์ดคาสท์ไปที่ ไอพี 224.1.2.20 พอร์ท 3000



รูปที่ 4.4 หน้าต่างยืนยันไฟล์และแอดเดรสที่ต้องการบอร์ดคาสท์

- เมื่อกดปุ่ม OK เพื่อยืนยันไฟล์และแอดเดรสที่ต้องการบอร์ดคาสท์ไปแล้ว เลือกกดปุ่ม Send ที่อยู่ตรงส่วนล่างของหน้าต่างหลัก เพื่อเริ่มทำการบอร์ดคาสท์ โดยตารางรายการด้านล่างจะทำการเพิ่มชื่อรายการเองดังรูปที่ 4.5 แต่เมื่อต้องการจะหยุด ให้ใช้ปุ่ม Stop เป็นการหยุดการบอร์ดคาสท์ โดยจะมีหน้าต่างส่วนบนของหน้าต่างหลักของโปรแกรม แสดงการทำงานของโปรแกรมในขณะนั้น

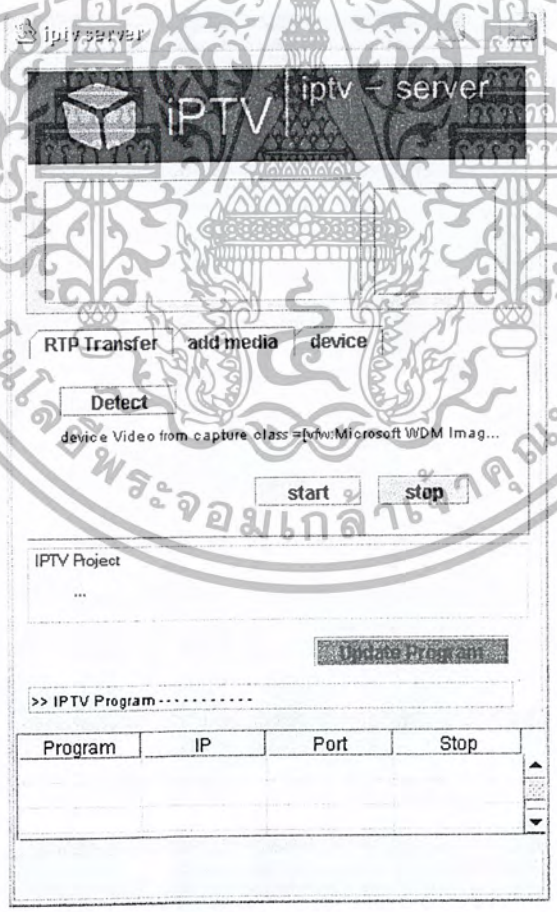


รูปที่ 4.5 หน้าต่างรายการฝั่งเซิร์ฟเวอร์

4.1.2 การบอร์ดคาสท์รายการทีวีที่ดึงข้อมูลมาจากการ์ดทีวี

เป็นการเลือกรายการทีวีที่ได้รับมาจากการ์ดทีวี ซึ่งเป็นรายการที่สามารถรับชมได้โดยโทรทัศน์ในขณะนั้นจริง และนำข้อมูลนั้นมาบอร์ดคาสท์ผ่านเครือข่ายอินเตอร์เน็ต เป็นแบบเรียลไทม์ ทำได้โดย

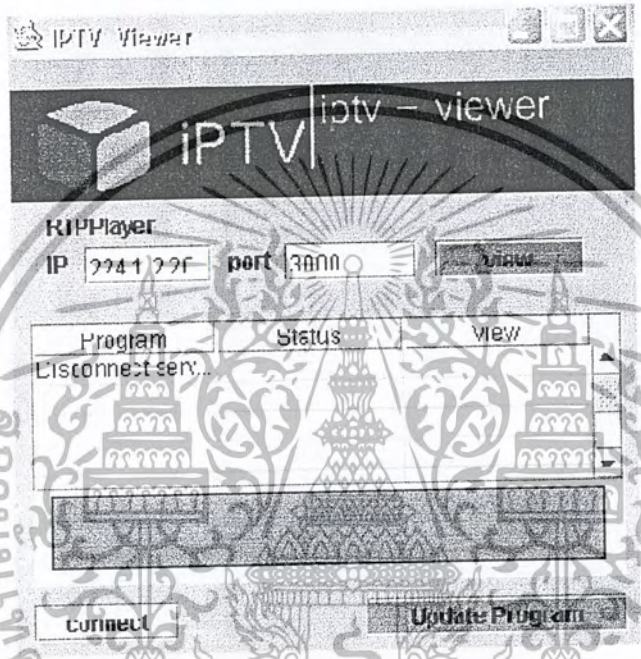
- เลือกแถบ RTP Transfer ในส่วนกลางของหน้าหลัก แล้วระบุไอพีและพอร์ทที่ต้องการจะบอร์ดคาสท์ไป
- เลือกแถบ Device ที่อยู่ในส่วนกลางของหน้าหลักของโปรแกรม ดังรูปที่ 4.6
- กดปุ่ม Detect ที่อยู่ในแถบ Device เพื่อเป็นการเลือกดึงข้อมูลจากการ์ดทีวี
- กดปุ่ม Start ในแถบ เพื่อเป็นการเริ่มการบอร์ดคาสท์แบบดึงข้อมูลมาจากการ์ดทีวี และปุ่ม Stop ในแถบ ใช้ในการหยุดการบอร์ดคาสท์แบบดึงข้อมูลจากการ์ดทีวี



รูปที่ 4.6 แถบเลือกการบอร์ดคาสท์แบบดึงข้อมูลจากการ์ดทีวี

4.2 การใช้งานของโปรแกรมฝั่งไคลเอ้นท์

การใช้งานโปรแกรมในส่วนนี้จะเป็นโปรแกรมที่ใช้เพื่อรับข้อมูลมัลติมีเดียหรือรายการทีวีที่มีการบอร์ดคาสท์มาจากโปรแกรมฝั่งไคลเอ้นท์ โดยใช้การรับข้อมูลแบบเรียลไทม์ คือจะมีการแสดงผลในขณะที่การรับข้อมูลยังไม่เสร็จสิ้นได้



รูปที่ 4.7 หน้าต่างโปรแกรมฝั่งไคลเอ้นท์

โดยที่โปรแกรมฝั่งไคลเอ้นท์ จะสามารถระบุไอพีและพอร์ท ที่โปรแกรมฝั่งเซิร์ฟเวอร์ได้ทำการบอร์ดคาสท์มาเพื่อรับข้อมูลมัลติมีเดีย เมื่อกดปุ่ม connect จะเป็นการเริ่มรับข้อมูลรายชื่อรายการทีวีที่สามารถเลือกรับชมได้ในขณะนั้น เมื่อเลือกรายการได้แล้วให้ระบุไอพีและพอร์ทของรายการนั้น แล้วกดปุ่ม view จะมีหน้าต่างแสดงผลข้อมูลมัลติมีเดียที่ได้รับมาปรากฏขึ้น ดังรูป 4.8

รูปที่ 4.8 หน้าต่างแสดงผลของโปรแกรมฝั่งไคลเอ้นท์

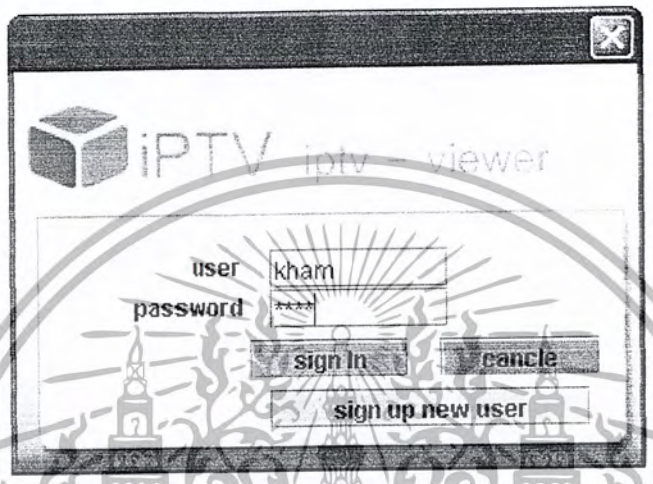
ในการรับข้อมูลมัลติมีเดียที่บอร์ดคาสท์มาจากเซิร์ฟเวอร์ ทั้งการรับข้อมูลที่เซิร์ฟเวอร์ส่งเป็นไฟล์ข้อมูลที่มีอยู่แล้วมา หรือการรับข้อมูลที่เซิร์ฟเวอร์บอร์ดคาสท์มาจากการ์ดทีวี มีรูปแบบการรับและวิธีใช้โปรแกรมฝั่งไคลเอ้นท์วิธีเดียวกัน ซึ่งคุณภาพที่ได้จะต่างกัน คือ การรับชมรายการทีวีที่บอร์ดคาสท์มาจากการ์ดทีวีของเซิร์ฟเวอร์ จะมีคุณภาพด้อยกว่า รายการที่รับมาจากไฟล์ของเซิร์ฟเวอร์ เนื่องจาก ต้องนำข้อมูลมาจากการ์ดทีวีซึ่งการ์ดทีวีรับสัญญาณทีวีจากเสาอากาศอีกที อาจเกิด loss สูญเสีย หรือมีคลื่นรบกวนได้ง่าย ผิดกับรายการที่รับมาจากไฟล์ซึ่งเป็นข้อมูลที่มาจากเซิร์ฟเวอร์โดยตรง จึงเกิดการสูญเสียได้น้อยกว่าการรับมาจากการ์ดทีวีมาก

รูปที่ 4.9 ข้อมูลที่รับได้ที่ไคลเอ้นท์จากการบอร์ดคาสท์จากการ์ดทีวี

รูปที่ 4.10 ข้อมูลที่รับได้ที่ไคลเอ้นท์จากการบอร์ดคาสท์จากไฟล์ในเครื่องเซิร์ฟเวอร์

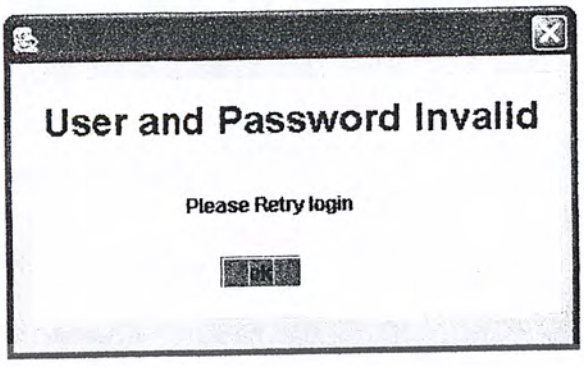
4.3 การล็อคอินเข้าสู่ระบบ

ก่อนการใช้งานโปรแกรมฝั่งไคลเอ้นท์ จะต้องทำการล็อคอินเพื่อยืนยันตนเองก่อน ให้เซิร์ฟเวอร์รับรู้ว่าขณะนั้นมีผู้ใช้งานกี่คน และเป็นผู้ใช้งานคนใดบ้าง โดยที่จะมีหน้าต่างล็อคอินแสดงดังรูปที่ 4.11



รูปที่ 4.11 หน้าต่างหน้าล็อคอิน

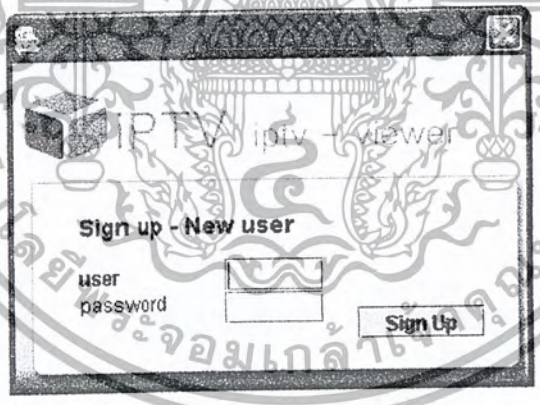
เมื่อเข้าสู่หน้าต่างล็อคอินแล้ว ให้กรอกชื่อผู้ใช้งาน และพาสเวิร์ดลงในช่องที่ระบุไว้ จากนั้นกดปุ่ม sign in เพื่อติดต่อไปยังเซิร์ฟเวอร์ แล้วเซิร์ฟเวอร์จะทำการตรวจสอบรหัสชื่อผู้ใช้และพาสเวิร์ด ซึ่งหากถูกต้องครบถ้วน โปรแกรมจะเข้าสู่หน้าต่างหลักของฝั่งไคลเอ้นท์ และจะสามารถใช้งานได้ แต่หากรหัสชื่อผู้ใช้หรือพาสเวิร์ดไม่ถูกต้อง จะปรากฏหน้าต่างเตือนให้ตรวจสอบรหัสชื่อผู้ใช้และพาสเวิร์ดให้ถูกต้องอีกครั้งดังรูปที่ 4.12 และหากต้องการจะออกจากหน้าต่างล็อคอิน ให้กดปุ่ม cancle



รูปที่ 4.12 หน้าต่างเตือนให้ตรวจสอบรหัสไอดีลพาสเวิร์ดอีกครั้ง

4.4 การสมัครเป็นผู้ใช้งานใหม่

หากต้องการสมัครเป็นผู้ใช้งานใหม่ ต้องเข้าสู่ระบบโดยเปิดโปรแกรมฝั่งไคลเอ้นท์ จะปรากฏหน้าต่างหน้าล็อคอิน แล้วกดปุ่ม sing up new user ซึ่งจะปรากฏหน้าต่างหน้าสมัครเป็นผู้ใช้งานใหม่ดังรูปที่ 4.13



รูปที่ 4.13 หน้าต่างหน้าสมัครเป็นผู้ใช้งานใหม่

จากนั้นให้ทำการกรอกข้อมูลของผู้ใช้งานใหม่ลงในช่องต่างๆให้ครบถ้วน และเมื่อทำการกรอกข้อมูลผู้ใช้งานใหม่เรียบร้อยแล้ว กดปุ่ม Sign Up จะปรากฏหน้าต่างยืนยันการสมัคร ให้กดปุ่ม OK เพื่อยืนยันการสมัคร แล้วเซิร์ฟเวอร์จะทำการอัพเดตข้อมูลลงในฐานข้อมูล จึงจะสามารถใช้ชื่อผู้ใช้และพาสเวิร์ดที่ทำการสมัครใหม่ได้

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

การรับชมรายการทีวีผ่านเครือข่ายอินเตอร์เน็ตนี้ มีฟังก์ชั่นการทำงานที่ติดต่อกับผู้ใช้ และสามารถทำงานรับชมรายการได้เป็นอย่างดี และผู้ใช้สามารถใช้งานได้สะดวก การทำงานในส่วนของเซิร์ฟเวอร์ สามารถรับข้อมูลจากการ์ดทีวีและนำข้อมูลนั้น ๆ มาส่งผ่านเครือข่ายอินเตอร์เน็ตแบบเรียลไทม์ได้ และยังสามารถเลือกไฟล์ที่มีอยู่แล้วให้สามารถส่งผ่านข้อมูลนั้นแทนการดึงข้อมูลมาจากการ์ดทีวีได้ ส่วนการทำงานของส่วนไคลเอ้นท์นั้น สามารถรับข้อมูลมัลติมีเดียที่ส่งจากเซิร์ฟเวอร์และแสดงผลออกได้ในเวลาเดียวกัน โดยไม่ต้องรอการรับข้อมูลให้เสร็จสิ้นก่อน โดยที่ระบบการทำงานนี้สามารถรองรับผู้ใช้งานได้หลายคน หลังจากการทำโครงงานชิ้นนี้แล้ว ทำให้มีความเข้าใจหลักการของการทำงานโปรโตคอล RTP การบอร์ดคาสท์แบบมัลติคาสท์ และการรับส่งข้อมลมัลติมีเดียแบบเรียลไทม์ได้เป็นอย่างดี อีกทั้งยังเข้าใจหลักการออกแบบเชิงวัตถุ ซึ่งสามารถนำมาใช้ประโยชน์ในการศึกษาและพัฒนาระบบต่าง ๆ ต่อไป

5.2 ปัญหาที่เกิดขึ้นระหว่างการทดลอง

- การทำงานด้วย JMF เป็นภาษาที่ค่อนข้างใหม่สำหรับตอนนี้ ทำให้การหาหนังสือเพื่อมาศึกษาวิธีใช้งานและนำมาใช้งานนั้นยาก ส่วนใหญ่ที่หามาได้จะเป็นภาษาอังกฤษ ซึ่งทำให้บางครั้ง การตีความจากภาษาอังกฤษเป็นภาษาไทยลำบาก เข้าใจยาก ต้องใช้การลองทำดูทีละขั้นตอน ทำให้ใช้เวลาในการศึกษาค่อนข้างเยอะ
- จำนวนเครื่องที่ใช้ในการทดลองโครงงานนี้ เนื่องจากต้องมีการส่งผ่านระบบอินเตอร์เน็ต ต้องใช้มากกว่า 1 เครื่องขึ้นไป ซึ่งต้องยืมเครื่องของกลุ่มอื่น ๆ เพื่อใช้ในการทดลอง
- เนื่องจากใช้เวลาในการศึกษาและออกแบบมากเกินไป ทำให้การพัฒนาโปรแกรมมีเวลาน้อยลง ซึ่งการที่จะทำงานให้ตรงตามที่ออกแบบไว้ตอนแรกทำได้ยาก บางครั้งต้องมีการตัดตอนบางส่วน ที่คิดว่าไม่จำเป็นออกไปบ้าง เพื่อให้การพัฒนาทันเวลาที่กำหนด โดยที่ยังคงส่วนการทำงานหลัก ๆ ของโปรแกรมไว้อย่างเข้มงวด และเพิ่มเติมส่วนที่ตัดออกไปให้มากที่สุดเท่าที่จะทำได้ในภายหลัง
- มีการทำงานแยกออกเป็นส่วน ๆ ซึ่งในบางครั้งตัวแปรที่ใช้ในการเขียนโปรแกรมใช้ชื่อไม่เหมือนกัน ทำให้ทำงานร่วมกันไม่ได้และการแก้ไขข้อผิดพลาดทำได้ลำบาก และบางครั้งการทำงานของโปรแกรมแต่ละส่วนที่พัฒนามา ใช้รูปแบบของไฟล์ไม่เหมือนกัน

- การออกแบบอินเตอร์เฟสให้ใช้งานง่าย สวยงาม และรองรับการทำงานของโปรแกรมที่เหมาะสมนั้น ทำได้ยาก เนื่องจากโปรแกรมแต่ละส่วนที่เขียนขึ้นมานั้น มีการทำงานเฉพาะตัวของมันเอง เช่น การส่งข้อมูลแบบเลือกไฟล์ในการส่ง และการส่งข้อมูลแบบดึงข้อมูลมาจากการ์ดทีวีนั้น ต่างก็มีปุ่มเริ่มและหยุดการทำงานของมันเอง เมื่อนำปุ่มทั้งสองมารวมกันในปุ่มเดียว อาจเกิดข้อผิดพลาดต่าง ๆ ขึ้นมา ส่งผลกระทบต่อการออกแบบหน้าจออินเตอร์เฟส
- โปรแกรมที่พัฒนาขึ้นเมื่อมีผู้ใช้หลาย ๆ คนเปิดรับชมรายการทีวีในเวลาเดียวกัน จะทำให้เกิดการแบ่งแบนด์วิธกันใช้ ส่งผลให้เกิดการรับชมภาพกระตุก ๆ ในบางระยะ
- ตัวการ์ดทีวีที่ใช้ในการรับสัญญาณทีวี รับสัญญาณโดยการใช้เสาอากาศ ซึ่งทำให้คุณภาพของข้อมูลที่ได้ขึ้นอยู่กับความสามารถของเสาอากาศของการ์ดทีวี ซึ่งคุณภาพที่ได้ยังเป็นคุณภาพที่ไม่ดีนัก

5.3 แนวทางการพัฒนาโครงงาน

- พัฒนาโปรแกรมฝั่งเซิร์ฟเวอร์ให้สามารถบังคับการ์ดทีวี ให้สามารถค้นหาคลื่นสัญญาณทีวีเองได้อย่างมีประสิทธิภาพ
- พัฒนาโปรแกรมฝั่งไคลเอ้นท์ให้สามารถขยายภาพการรับชมตามต้องการได้ รวมทั้งอาจจะเพิ่มให้สามารถบันทึกรายการทีวีเป็นไฟล์ได้ตามต้องการ
- เลือกใช้อุปกรณ์ต่าง ๆ ที่ใช้ ให้มีประสิทธิภาพดียิ่งขึ้น ตามเทคโนโลยีและโอกาสที่มีในขณะนั้น เนื่องจากในอนาคตอาจจะมีอุปกรณ์ที่มีคุณภาพดียิ่งขึ้น เช่น การ์ดทีวี หากเลือกใช้การ์ดที่มีประสิทธิภาพการทำงานดีมาก ๆ ก็จะส่งผลให้ คุณภาพของข้อมูลที่ได้ ดียิ่งขึ้น หรือตัวคอมพิวเตอร์ฝั่งเซิร์ฟเวอร์ หากใช้คอมพิวเตอร์ที่มีสเปคเครื่องดีมาก ๆ ก็จะสามารถลดปัญหาการแบ่งแบนด์วิธเมื่อมีผู้ใช้หลายคน ได้ส่วนหนึ่งด้วย

บรรณานุกรม

1. Dr. Harvey M. Deitel and Paul J. Deite, Java How to Program FIFTH EDITION, Deitel & Associates, Inc., 2003
2. JMF 2.0 FCS, JAVA Media Framework API Guide, Sun Microsystems, Inc., November 19, 1999
3. ดร. วีระศักดิ์ ซึงถาวร, JAVA PROGRAMMING Volume I, บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), พ.ศ. 2543
4. ดร. วีระศักดิ์ ซึงถาวร, JAVA PROGRAMMING Volume II, บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), พ.ศ. 2543