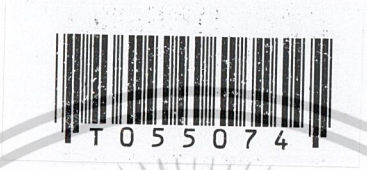
การพัฒนาการติดต่อระหว่างคอมพิวเตอร์กับโทรศัพท์โดย DirectX และ TAPI

Phone Via Net With DirectX and TAPI



นางสาวจารุณี มีเงิน

นายจิรายุ เจริญวิริยะภาพ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

การพัฒนาการติดต่อระหว่างคอมพิวเตอร์กับโทรศัพท์โดย DirectX และ TAPI

Phone Via Net With DirectX and TAPI

โดย

นางสาวจารุณี มีเงิน

นายจิรายุ เจริญวิริยะภาพ

อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ดร.ชุติเมษฎ์ ศรีนิลทา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาการติดต่อระหว่างคอมพิวเตอร์กับโทรศัพท์โดย DirectX และ TAPI Phone Via Net With DirectX and TAPI

คณะผู้จัดทำ นางสาวจารุณี มีเงิน รหัส 43010058

นายจิรายุ เจริญวิริยะภาพ รหัส 43010069

.....อาจารย์ที่ปรึกษา

(ดร.วรวัฒน์ ลิ้มโภคา)

.....อาจารย์ที่ปรึกษา

(ดร.ชุติเมษฎ์ ศรีนิลทา)

การพัฒนาการติดต่อระหว่างคอมพิวเตอร์กับโทรศัพท์โดย DirectX และ TAPI

นางสาวจารุณี มีเงิน 43010058
นายจิรายุ เจริญวิริยะภาพ 43010069
ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ดร. ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

การพัฒนาแอปพลิเคชันในโครงงานนี้เป็นการพัฒนาซอฟแวร์เพื่อศึกษาการเชื่อมต่อและสื่อสารระหว่างเครือข่ายอินเตอร์เน็ตกับเครือข่ายโทรศัพท์ โดยได้เริ่มด้วยการพัฒนาแอปพลิเคชันให้สามารถติดต่อพูดคุยระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ได้ด้วยเสียงและสามารถทำการประชุมหลายสายได้ และในส่วนของการการสื่อสารระหว่างเครือข่ายอินเตอร์เน็ตและเครือข่ายโทรศัพท์ได้ทำการพัฒนาให้คอมพิวเตอร์ติดต่อกับโทรศัพท์ ติดต่อกันในลักษณะของการฝากข้อความ

โดยแอปพลิเคชันที่พัฒนาจะประกอบด้วย 2 ส่วนที่สำคัญคือ ส่วนของคอมพิวเตอร์ที่เป็นปลายทางของเครือข่ายอินเตอร์เน็ต ซึ่งจะมีการเรียกใช้บริการจากเซิร์ฟเวอร์เพื่อติดต่อกับปลายทางโดยรูปแบบที่ใช้ในการติดต่อคือเสียง ซึ่งเป็นข้อมูลแบบเรียลไทม์ และส่วนที่สองคือส่วนของเซิร์ฟเวอร์ที่ทำหน้าที่ในการรับการร้องขอของไคลแอนท์ และสร้างช่องทางการติดต่อ โดยผ่านเครือข่ายอินเตอร์เน็ต ซึ่งการเชื่อมต่อภายในเครือข่ายอินเตอร์เน็ตนั้นสามารถพัฒนาโดยใช้ไดเร็กเพลย์ และในส่วนการสร้างช่องทางการเชื่อมต่อผ่านเครือข่ายโทรศัพท์จะพัฒนาโดยใช้ TAPI

Phone Via Net with DirectX and TAPI

Miss Jarunee Meengern
Mr. Jirayu Charoenviriyaphap
Dr. Voravat Limpoka Advisor
Dr. Chutimet Srinilta Advisor
Academic Year 2003

ABSTRACT

This thesis is learning about connection and communication between telephone network and internet. So we develop the application for connecting between computer and computer and allow user voice chatting through application. And client connect to telephone base on Voice-mail .The application is consist of two important parts, the first is client that request service from server to connect to target with real-time voice and the second is server that service client and create connection to target. The connection between computer and computer is created by DirectPlay and connection between computer and telephone is created by TAPI. Both components are developed by Microsoft.

กิตติกรรมประกาศ

โครงงานและรายงานฉบับนี้สำเร็จลุล่วงไปได้ด้วยดีเนื่องจาก ได้รับความรู้และการประสิทธิ์ประสาทวิชาจากบูรพาจารย์ โดยเฉพาะอย่างยิ่ง อาจารย์วรวัฒน์ ลิ้มโภคา และอาจารย์ชุติเมษฎ์ ศรีนิลทา ที่ได้ให้คำปรึกษา และให้การช่วยเหลือสนับสนุนวิชาต่างๆ เป็นอย่างดีมาตลอดขอขอบคุณทางภาควิชาคอมพิวเตอร์ ที่ได้ให้โอกาสและให้การเอื้อเฟื้อสถานที่และอุปกรณ์เพื่อใช้ในการทดลองต่างๆ

ขอขอบคุณอาจารย์อำนาจ ขาวเน ที่ให้ความอนุเคราะห์ในการใช้โทรศัพท์เพื่อทำการทดลองและซื้อส้มมาฝากและขอขอบคุณอาจารย์ สมศักดิ์ และ อาจารย์ อรัญญา วลัยรัชต์ ที่เข้ามาให้คำปรึกษาและเป็นกำลังใจในการทำการทดลองเมื่อไปทำการทดลอง

ขอขอบคุณพี่ๆ เพื่อนๆ ในห้อง Olala ที่คอยให้กำลังใจและให้ความสนุกสนานทำให้ไม่เครียดกับงานเกินไป และนอกจากความบันเทิงแล้ว ขอขอบคุณเพื่อนๆที่มานั่งอ่านหนังสือและติวกันในช่วงสอบเพราะถ้าไม่ได้ช่วยกันอ่านแล้ว เราคงมีเวลาทำโครงงานน้อยลงไปมาก ไม่เพียงแต่ช่วงสอบเท่านั้น ยังมีงานที่ได้ช่วยกันทำด้วย

และทั้งนี้ทั้งนั้นเหนือสิ่งอื่นใด โครงงานนี้ไม่อาจเสร็จสมบูรณ์ได้หากไม่ได้รับการช่วยเหลือและการให้การสนับสนุน กำลัง ทั้งหลายทั้งปวงจากคุณพ่อ, คุณแม่, ผู้ปกครอง และผู้ที่มีส่วนเกี่ยวข้องในการให้คำปรึกษาที่ดีเสมอมาตลอดระยะเวลาของการทำโครงงานนี้ ทางผู้จัดทำขอขอบคุณทุกท่านอีกครั้ง ทั้งที่เอ่ยนาม และไม่ได้เอ่ยนามไว้ ณ ที่นี้ด้วย

นางสาวจารุณี มีเงิน

นาย จิรายุ เจริญวิริยะภาพ

สารบัญ

สารบัญตาราง

หน้าที่

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

เนื่องจากปัจจุบันอินเตอร์เน็ตแพร่หลายและมีผลต่อการทำงานในปัจจุบันมีการทำงานบนอินเตอร์เน็ตมากขึ้น และการติดต่อกันบนอินเตอร์เน็ตที่เป็นแบบเรลไทม์ (Real-time) นั้นคือการแชท (Chat) นั่นเองที่นิยมในปัจจุบันนั้นคือการแชทแบบพิมพ์ตัวอักษร มีโปรแกรมที่นิยมใช้กันคือ MSN Messenger และ Yahoo Messenger แต่การแชทแบบพิมพ์นั้นสร้างความไม่สะดวกต่อคนที่กำลังทำงานอยู่ โดยที่จะต้องสลับหน้าต่างการทำงานและการแชทไปตลอดเวลา เราจึงเล็งเห็นว่าการแชทแบบใช้เสียงหรือที่เรียกว่า Voice Chat นั้น น่าจะเป็นตัวเลือกที่ดีในการอำนวยความสะดวกในการแชทและทำงานในเวลาเดียวกันและเราได้เห็นถึงความสามารถในการจัดการกับเกมบนเครือข่ายของไดเร็กเอ็ก เราจึงมีความสนใจที่จะนำ ไดเร็กเอ็ก มาพัฒนาในส่วนของการทำวอยซ์แชท และเพื่ออำนวยประโยชน์อีกนั้น การประชุมผ่านทางเครือข่ายก็เป็นสิ่งที่น่าสนใจ โดยเราสามารถประชุมงานผ่านแอปพลิเคชันของเราได้ โดยในขณะคุยนั้นสามารถทำงานอย่างอื่นร่วนกันไปด้วยได้

นอกจากวอยซ์แชทนั้นเราจะมีการนำเอาความสามารถของการติดต่อกับโทรศัพท์เข้ามาเสริมกับแอปพลิเคชันของเราเพื่อที่จะให้ ผู้ใช้โปรแกรมเราสามารถติดต่อกับคนภายนอก โ ดยการติดต่อกับคนที่ใช้โทรศัพท์นั้น เราจะทำการฝากข้อความเสียงไว้โดยสามารถฝากได้ทั้งจากทางไคลแอนท์ที่เป็นคอมพิวเตอร์และบันทึกผ่านทางโทรศัพท์ก็ได้และให้อีกคนที่ใช้โทรศัพท์ได้รับข้อมูลที่เราฝากไว้โดยการติดต่อกับโทรศัพท์นั้นเราใช้ TAPI เข้ามาในการพัฒนา ซึ่งถ้าหากเราทำการฝากข้อความไว้นั้น จะอำนวยความสะดวกในด้านไม่ขาดการติดต่อกับภายนอกแม้ว่าจะทำงานอยู่บนอินเตอร์เน็ตก็ตาม

1.2 วัตถุประสงค์

1. เพื่อศึกษาและพัฒนาการทำติดต่อพูดคุยระหว่างคอมพิวเตอร์แบบไคลแอนท์-เซิร์ฟเวอร์ (Voice Chat Client-Server)
2. เพื่อศึกษาการอินเตอร์เฟซระหว่าง ระบบโทรศัพท์กับ คอมพิวเตอร์
3. เพื่อศึกษาการเขียนโปรแกรมโดยใช้ API ของ Microsoft Window ว่าแตกต่างจากการเขียนโปรแกรมทั่วไปอย่างไร แล้วสามารถพัฒนาโปรแกรมให้มีประสิทธิภาพได้อย่างไร
4. เพื่อศึกษาและพัฒนาระบบฝากข้อความเพื่อถ่ายทอดเสียงไปยังโทรศัพท์
5. เพื่อศึกษาความสามารถของ TAPI และ ไดเร็กเอ็ก ว่าสามารถนำไปพัฒนาใช้ในงานในด้านใด และสามารถพัฒนาแอปพลิเคชันร่วมกันได้อย่างไร

6. เพื่อศึกษาการติดต่อสื่อสารโดยใช้เสียงผ่านทางเครือข่ายอินเตอร์เน็ต และเครือข่ายโทรศัพท์ รวมทั้ง การ Sampling เสียงและการบีบอัดเสียง เพื่อให้สามารถส่งสัญญาณเสียงผ่านทั้งสองระบบนี้ได้
7. เพื่อพัฒนาโปรแกรมที่สามารถทำการติดต่อระหว่างคอมพิวเตอร์และโทรศัพท์เข้าด้วยกัน และให้ผู้ใช้งานสามารถติดต่อสื่อสารทางเสียงได้
8. เพื่อเปรียบเทียบประสิทธิภาพในการติดต่อระหว่างคอมพิวเตอร์ผ่านทางเซิร์ฟเวอร์โดยมีสภาพแวดล้อมต่างกัน

1.3 ขอบเขตของโครงงาน

โครงงานนี้จะพัฒนาแอปพลิเคชันในส่วนของการเชื่อมต่อเครือข่ายอินเตอร์เน็ต ระหว่างคอมพิวเตอร์กับคอมพิวเตอร์จะเป็นส่วนที่อิมพลิเมนท์โดยใช้ ไดเร็กเอ็ก ซึ่งเป็นคอมโพเนนท์ของไมโครซอฟท์โดยในส่วนที่ใช้ในการพัฒนาเพื่อสร้างการติดต่อโดยใช้เสียงผ่านเครือข่ายอินเตอร์เน็ต จะใช้คอมโพเนนท์ส่วน ไดเร็กเพลย์ ในส่วนการสร้างการเชื่อมต่อภายในเครือข่ายและ ไดเร็กเพลย์วอยซ์ ในส่วนของการจัดการรูปแบบของสัญญาณเสียงและควบคุมการส่งสัญญาณเสียงรวมทั้งการเข้ารหัสของเสียงผ่านเครือข่าย ซึ่งส่วนของการสร้างการเชื่อมต่อระหว่างโทรศัพท์ซึ่งเป็นการเชื่อมต่อระหว่างเครือข่ายโทรศัพท์และเครือข่ายอินเตอร์เน็ต ได้มีการนำเอา API อีกคอมโพเนนท์หนึ่งของไมโครซอฟท์มาศึกษาถึงวิธีการสร้างการเชื่อมต่อ โดยใช้ TAPI ซึ่งแม้ TAPI จะมีฟังก์ชันที่ช่วยในการติดต่อสื่อสารผ่านเครือข่ายโทรศัพท์ก็จริงแต่การนำเอามาพัฒนาร่วมกับ ไดเร็กเอ็ก เพื่อให้สามารถติดต่อระหว่างสองเครือข่ายผ่านเสียงนั้นเป็นสิ่งที่ต้องศึกษาและค้นคว้าเพื่อสร้างความสัมพันธ์และนำมาใช้ให้ได้จริงต่อไป

1.4 วิธีการดำเนินงาน

โครงงานนี้เริ่มจากการศึกษาทฤษฎีพื้นฐานต่างๆที่เกี่ยวข้องจึงศึกษาต่อถึงวิธีการส่งสัญญาณเสียงผ่านเครือข่ายทั้งสอง รวมทั้งรูปแบบของ Packet และวิธีการสร้างแพ็กเก็ตเสียงที่ใช้ในการเชื่อมต่อ เช่น CODEC(Compression &Decompression) หรือวิธีการ Sampling เสียงให้ได้เพียงพอต่อการส่งแบบเรียลไทม์โดยไม่ทำให้คุณภาพเสียงลดลงมากนัก รวมถึงโปรโตคอลที่ใช้ร่วมในการพัฒนา อย่างเช่น VOIP (Voice Over Internet Protocol) และ ได้ศึกษาถึง API ต่างๆที่ใช้ในอิมพลิเมนท์อย่างเช่น JMF(Java Multimedia Framework), DirectSound จากนั้นได้ทำการศึกษาเกี่ยวกับการสร้างการเชื่อมต่อระหว่างเครือข่ายด้วย API ต่างๆที่รองรับเช่น WinSock , DirectPlay , JAVA , .Net รวมทั้งการเชื่อมต่อด้วยรูปแบบต่างๆ ไม่ว่าจะเป็นแบบ peer-to-peer หรือ ไคลแอนท์-เซิร์ฟเวอร์

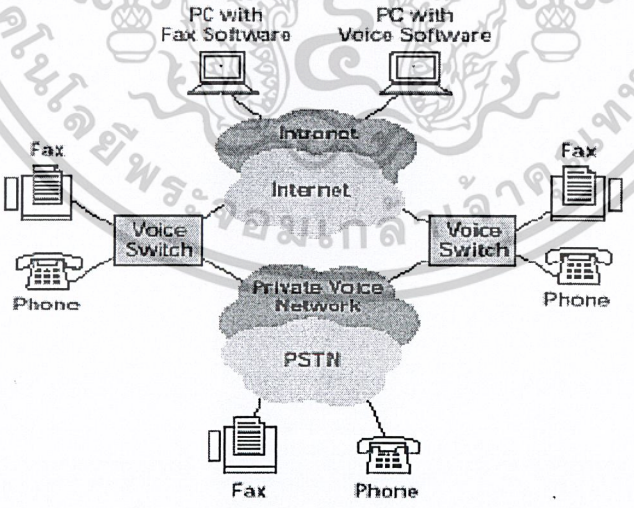
ในส่วนของการติดต่อระหว่างเครือข่ายอินเตอร์เน็ตและเครือข่ายโทรศัพท์ด้วยเทคโนโลยีต่างๆที่ใช้ในปัจจุบัน เช่น VOIP รวมทั้ง API ที่ใช้ในการอิมพลิเมนท์แอปพลิเคชัน เช่น TAPI จากนั้นจะนำเอาเทคโนโลยีทั้งสองส่วนมาพัฒนาร่วมกันเพื่อให้สามารถติดต่อโทรศัพท์พื้นฐานผ่านทางคอมพิวเตอร์ได้ โดยสามารถติดต่อกันโดยเสียงได้ และรับข้อความที่ส่งมาจากแอปพลิเคชันได้

บทที่ 2
ทฤษฎีที่ใช้ประกอบโครงการ

2.1 VOIP

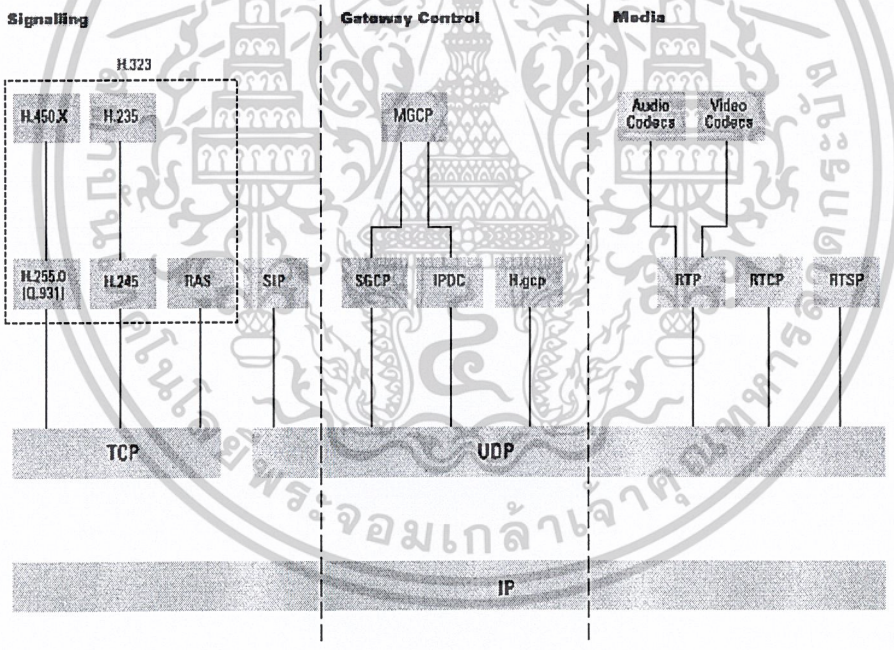
VOIP หรือ Voice over the Internet Protocol คือมาตรฐานที่พัฒนาขึ้นมาสำหรับส่งสัญญาณเสียงบนอินเตอร์เน็ตที่มีพื้นฐานอยู่บน IP (IP based Internet) การส่งเสียงบนเครือข่ายอินเตอร์เน็ต เป็นระบบที่นำสัญญาณข้อมูลเสียงมาบรรจุลงเป็นแพ็กเก็ตไอพีแล้วส่งไปโดยที่เราเตอร์มีวิธีการปรับตัวเพื่อรับสัญญาณแพ็กเก็ตและยังแก้ปัญหาบางอย่างให้ เช่น การบีบอัดสัญญาณเสียงให้มีขนาดเล็กลง การแก้ปัญหาเมื่อมีแพ็กเก็ตสูญหาย หรือได้มาล่าช้า

ระบบ VOIP เป็นระบบที่นำสัญญาณเสียงที่ผ่านการดิจิไตซ์ (Digitize)โดยหนึ่งช่องเสียงเมื่อแปลงเป็นข้อมูลจะมีขนาด 64 Kbps การนำข้อมูลเสียงขนาด 64 Kbps นี้ ต้องนำมาบีบอัดก่อน โดยทั่วไปจะเหลือประมาณ 10 Kbps ต่อช่องสัญญาณเสียง แล้วจึงบรรจุลงในไอพีแพ็กเก็ต เพื่อส่งผ่านทางเครือข่ายอินเตอร์เน็ต การสื่อสารผ่านทางเครือข่ายอินเตอร์เน็ต ต้องมี เราเตอร์ ที่ทำหน้าที่พิเศษเพื่อประกันคุณภาพของช่องสัญญาณเพื่อให้ข้อมูลไปถึงปลายทางหรือกลับมาได้อย่างถูกต้อง และอาจมีการให้สิทธิพิเศษก่อนแพ็กเก็ตไอพีอื่น เพื่อการให้บริการที่ทำให้เสียงมีคุณภาพ นอกจากนี้ VOIP ยังถูกนิยามให้เป็นความสามารถในการใช้โทรศัพท์ (ทำทุกอย่างที่เราสามารถทำได้ในปัจจุบันด้วย PSTN(Public Switched Telephone Network) และส่งสำเนาเอกสารบนเครือข่ายข้อมูลที่มีพื้นฐานอยู่บนเครือข่ายอินเตอร์เน็ตด้วย QoS (Quality of Service) การวัดประสิทธิภาพของระบบสื่อสารซึ่งแสดงถึงคุณภาพในการส่งและความสะดวกในการใช้บริการที่เหมาะสมและราคากับผลประโยชน์ที่คุ้มค่ากว่า



รูปที่ 2-1 Model ของ VOIP

นอกจากนี้ระบบ VOIP ยังรวมการสื่อสารแบบเรียลไทม์(Real-time) และไม่เรียลไทม์(Non-real time) เข้าด้วยกัน การส่งข้อความเสียงและแฟกซ์ใช้วิธีการเดียวกับการเรียกโทรศัพท์ธรรมดา แต่ไม่จำเป็นต้องมี QoS ระดับเดียวกันสำหรับในกรณีที่ต้องการคุณสมบัติแบบ เรียลไทม์ ดังนั้นจึงต้องมีการพัฒนาโปรโตคอลที่สามารถให้คุณสมบัติดังกล่าวในการส่งข้อมูล และที่สำคัญ VOIP ต้องการฟังก์ชันในการสร้างหรือสิ้นสุดการเรียก และหาตำแหน่งที่อยู่ของผู้ใช้ที่ถูกเรียก รวมทั้งฟังก์ชันที่ช่วยให้สามารถให้บริการได้เช่นเดียวกับในระบบโทรศัพท์ ซึ่งในชุดโปรโตคอล TCP/IP (Transport Control Protocol/Internet Protocol) นั้นไม่มีโปรโตคอล ที่ให้ฟังก์ชันดังกล่าว ดังนั้นจึงต้องมีการพัฒนาโปรโตคอลขึ้นใหม่เพื่อรองรับ VOIP ในปัจจุบันโปรโตคอลสำหรับ VOIP มีอยู่ 2 มาตรฐานคือ H.323[2] และ SIP(Session Initial Protocol)[1] โปรโตคอล H.323 เป็นโปรโตคอลที่พัฒนาโดย ITU-T (International Telecommunications Union- Telecommunications section) ส่วน SIP ถูกพัฒนาโดย IETF (Internet Engineering Task Force) โปรโตคอลทั้งสองมีหน้าที่หลักในการสร้างการสิ้นสุด และการเปลี่ยนแปลงรายละเอียดของการเรียกระหว่างผู้ใช้ VOIP รวมทั้งยังสามารถให้
ฟังก์ชันเพิ่มเติมอื่นๆ โปรโตคอลทั้งสองเป็น โปรโตคอลสำหรับ VOIP ซึ่งใช้บริการชุดโปรโตคอล TCP/IP ในชั้นต่ำกว่า และสามารถใช้งานร่วมกับโปรโตคอลอื่น เพื่อให้เกิดการบริการที่มีคุณภาพมากขึ้น



รูปที่ 2-2 VOIP Protocol stack

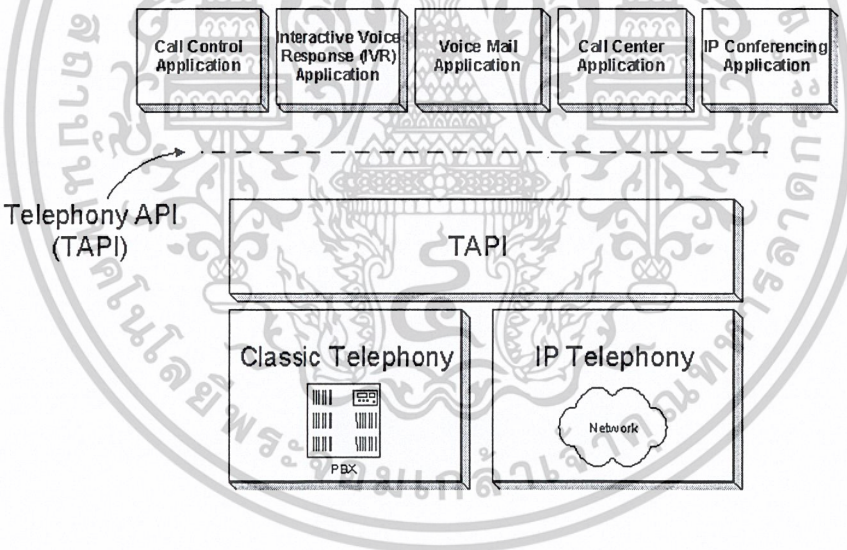
Protocol stack สำหรับ VOIP จะเป็นดังรูปที่ 2-2 จะเห็นว่าทั้งโปรโตคอล H.323 และ SIP เป็นโปรโตคอลในชั้นแอปพลิเคชัน (Application Layer) และใช้บริการของโปรโตคอลในชั้นที่ต่ำกว่า SIP สามารถใช้ได้ทั้ง UDP และ TCP ส่วน H.323 ใช้ TCP เท่านั้น แต่เนื่องจากว่าฟังก์ชันของ H.323 และ SIP

มีขอบเขตจำกัด ดังนั้นจึงได้นำโปรโตคอลอื่นมาช่วยในการทำงาน ซึ่งได้แก่ RTSP (Real-time Streaming Protocol) RSVP (Resource Reservation Protocol) และ RTP/RTCP ซึ่งทำให้ VOIP สามารถให้บริการได้อย่างมีประสิทธิภาพมากขึ้น โปรโตคอลดังกล่าวเป็นโปรโตคอลในชั้นแอปพลิเคชันซึ่งทำงานอยู่บนชุดโปรโตคอล TCP/IP โปรโตคอลเหล่านี้ไม่ได้เป็นโปรโตคอลเฉพาะสำหรับ VOIP แต่สำหรับโปรโตคอล H.323 และ SIP ซึ่งเป็นโปรโตคอลหลักสำหรับ VOIP

2.2 TAPI (Telephony Application Programming Interface)

ในการพัฒนาโปรแกรมสำหรับ IP telephony ในปัจจุบันมีเครื่องมือ (tool) ช่วยในการเขียน เช่น APIต่างๆ ตัวอย่างเช่น JTAPI (Java Telephony API) และ TAPI (Telephony API) รวมทั้งยังได้มีผู้ผลิต Protocol stack ทำให้การสร้างแอปพลิเคชันสำหรับ IP telephony ทำได้สะดวกขึ้น เนื่องจากเครื่องมือเหล่านี้ได้ซ่อนรายละเอียดบางอย่างพร้อมทั้งได้ให้องค์ประกอบร่วมสำหรับแอปพลิเคชันในลักษณะเดียวกัน ซึ่งช่วยลดภาระในการพัฒนาโปรแกรมได้

TAPI (Telephony API) เป็นกลุ่มของฟังก์ชันที่อนุญาตให้เขียนโปรแกรมเกี่ยวกับโทรศัพท์ได้ TAPI สนับสนุนการทำงานทั้งที่เป็นคำพูดและข้อมูล ใช้ได้กับอุปกรณ์ได้หลายชนิด สามารถจัดการกับการติดต่อที่ซับซ้อนได้หลายอย่างเช่น การประชุมหลายสาย



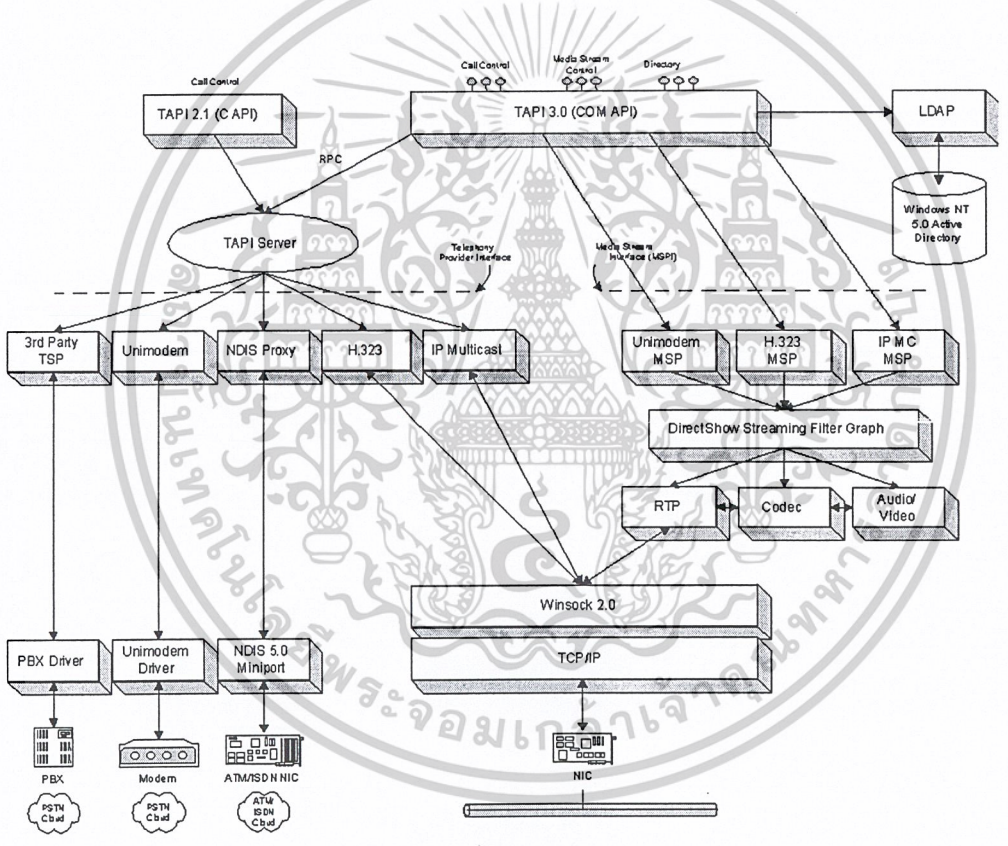
รูปที่ 2-3 โครงสร้างของ TAPI ภายใต้แอปพลิเคชันต่างๆ

TAPI เป็น API ที่ใช้เป็นตัวกลางในการติดต่อระหว่างคอมพิวเตอร์กับระบบโทรศัพท์ ไม่เฉพาะต้องผ่านเครือข่ายโทรศัพท์เท่านั้น แต่ยังสามารถใช้กับเครือข่ายอินเทอร์เนตก็ได้ ซึ่ง TAPI เป็น API บนระบบปฏิบัติการไมโครซอฟท์วินโดว์ (Microsoft Windows) ซึ่งทำให้แอปพลิเคชันสามารถเรียกใช้ฟังก์ชันการทำงานในระบบโทรศัพท์ได้ TAPI ได้ถูกพัฒนาโดยไมโครซอฟท์เพื่อรองรับเทคโนโลยี IP telephony ซึ่งจะช่วยให้การพัฒนาแอปพลิเคชันสำหรับ IP telephony ทำได้สะดวกและรวดเร็วขึ้น TAPI

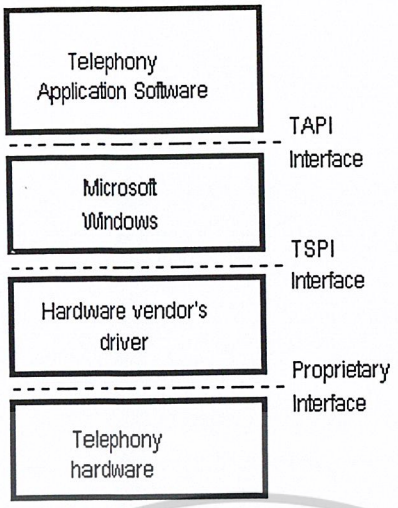
ได้ให้อินเตอร์เฟซของฟังก์ชันการควบคุมการเรียก (call control) สำหรับโปรโตคอลสื่อสารชนิดต่างๆ ทำให้ผู้ใช้สามารถเรียกใช้ได้โดยไม่ต้องสร้างขึ้นมาเอง ในปัจจุบัน TAPI เป็นเวอร์ชัน 3.0 ช่วยทำให้การจัดการเกี่ยวกับเครือข่ายภายในองค์กรทำได้อย่างมีประสิทธิภาพมากขึ้น นอกจากนั้น TAPI 3.0 ยังสนับสนุนในเรื่องของคุณภาพการให้บริการ (QoS) รวมทั้งสนับสนุนมาตรฐาน H.323 ซึ่งพัฒนาโดย ITU

2.2.1 ภาพรวมเทเลโฟนีของไมโครซอฟท์

Telephony Application Programming Interface (TAPI), Service Provider Interface (TSPI) และ Media Service Provider (MSP) ของไมโครซอฟท์นั้นสนับสนุนการพัฒนาแอปพลิเคชันของการติดต่อสื่อสารที่ทำงานบนระบบปฏิบัติการที่สนับสนุน Win32 ซึ่งจากรูปที่ 2-4 จะสังเกตเห็นว่าสถาปัตยกรรมเทเลโฟนีของไมโครซอฟท์นั้นสามารถนำมาใช้กับระบบ PSTN, ISDN และการติดต่อแบบ TCP/IP ทั้งยังสนับสนุนมาตรฐาน H.323 ด้วย



รูปที่ 2-4 ระบบที่เกี่ยวข้องกับโทรศัพท์ของไมโครซอฟท์



รูปที่ 2-5 โครงสร้างของ Microsoft Telephony Programming Model

จากรูปจะเห็นได้ว่าโครงสร้างของ Microsoft Telephony จะประกอบด้วย 2 ส่วนใหญ่ คือ

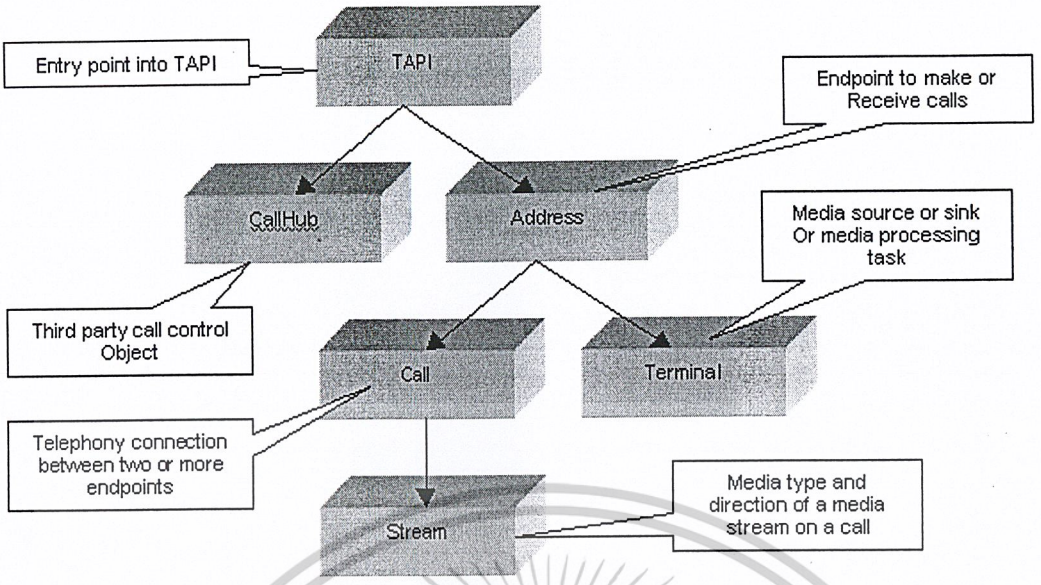
1. ส่วนของแอปพลิเคชันโปรแกรมที่ใช้ในการติดต่อคุณสมบัติด้านโทรศัพท์ของวินโดว์ เรียกว่า TAPI (Telephony Application Programming Interface)
2. ส่วนของวินโดว์ที่ใช้ในการติดต่อกับฮาร์ดแวร์ เรียกว่า TSPI(Telephony Service Provider's Interface)

โดยผู้ที่พัฒนาแอปพลิเคชันจะสามารถติดต่อกับเครือข่ายโทรศัพท์โดยใช้ TAPI ซึ่ง TAPI จะซ่อนการทำงานภายในไว้ให้ผู้พัฒนาใช้งานผ่านอินเตอร์เฟสที่สร้างไว้ให้ และ TAPI จะเป็นผู้ทำงานโดยในส่วนการติดต่อกับฮาร์ดแวร์จะเป็นหน้าที่ของ TSPI

2.2.2 ออบเจคต์ในโมดูลของ TAPI 3.0

ในออบเจคต์ของ TAPI 3.0 นั้น แสดงไว้ดังรูปด้านล่าง แอปพลิเคชันจะใช้ ออบเจคต์ของ TAPI เหมือน ต้นทาง ของ TAPI จาก ออบเจคต์ของ TAPI แอปพลิเคชันจะสามารถระบุแอดเดรสของออบเจคต์ ที่ใช้แทนจุดสุดท้ายได้ โดยที่การโทรศัพท์นั้นสามารถทำได้ทั้งการโทรไปเองและการรับสายโทรศัพท์ จากแอดเดรสของออบเจคต์ แอปพลิเคชันจะสามารถระบุว่าให้เทอมินัลนั้นทำการจัดการดับโทรศัพท์ที่แอดเดรสนั้นๆ เทอมินัลเป็นศูนย์กลางในการทำงาน เทอมินัลถูกใช้ในการ ระบุว่าให้สื่อที่ผ่านเข้ามานั้นทำการ จับแล้วส่ง, รับเข้ามาแล้วเล่น หรือ เข้ามาประมวลผล

เมื่อทำการสร้างการโทรออกแล้วหรือเตรียมรับสายนั้น การควบคุมแอปพลิเคชันที่ทำการโทรนั้นจะทำการโทรผ่าน Call Object จาก Call Object แอปพลิเคชันจะทำการระบุ Stream object ที่ใช้แทนสื่อที่มีพื้นฐานอยู่บน ชนิด และทิศทาง ยกตัวอย่างเช่น การคุยกันระหว่างคนสองคนจะมีการสื่อสาร 2 ทาง พร้อมกัน คือการรับและส่งเสียงพร้อมกันแอปพลิเคชันจะใช้ เทอมินัลและStream object ในการตั้งค่าการควบคุมการสื่อสารในการติดต่อ



รูปที่ 2-6 ออบเจคต์โมเดลของ TAPI 3.0

ออบเจคต์หลัก 5 ตัว ในสถาปัตยกรรมของการควบคุมมิเดียและการ Call คือ TAPI, Address, Terminal, Call และ Call Hubในรูปจะแสดงออบเจคต์เหล่านี้และอินเทอร์เฟซต่างๆ ที่สัมพันธ์กัน

- **ออบเจคต์ TAPI**

ออบเจคต์ TAPI แสดงถึงทรัพยากรของเทเลโฟนนี่ทั้งหมดที่คอมพิวเตอร์สามารถเข้าถึงได้ทำให้แอปพลิเคชันสามารถระบุแอดเดรสโลคอลและรีโมตได้ทั้งหมด แอปพลิเคชัน TAPI 3.0 ต้องสร้างอินสแตนท์ของออบเจคต์ TAPI และทำการ Initialize มันด้วย

- **ออบเจคต์ Address**

ออบเจคต์ Address หมายถึงสิ่งที่แสดงว่าสามารถที่จะทำและรับการ Call ได้ ออบเจคต์นี้มีอินเทอร์เฟซและเมธอดที่อนุญาต ให้แอปพลิเคชันกระทำสิ่งต่างๆ ได้เช่น

- สามารถแสดงว่าแอดเดรสที่กำหนดให้สามารถสนับสนุนชนิดของมิเดียที่ต้องการได้หรือไม่
- ระบุการ Call ปัจจุบันที่สัมพันธ์กับแอดเดรส
- สร้างหรือทำการส่งต่อการ Call ได้
- แสดงชื่อของผู้ให้บริการที่เกียวข้องได้
- ถ้ามี MSP อยู่ จะรับพอยน์เตอร์ของอินเทอร์เฟซที่อนุญาตกระทำกับเทอร์มินัลในระดับรายละเอียดได้
- รับและเซ็ตรายละเอียดอื่นๆ เช่น มีข้อความรออยู่หรือไม่

- **ออบเจคต์ Terminal**

ออบเจคต์ Terminal แสดงถึงแหล่งกำเนิด (source) หรือตัวแปลสัญญาณ (Render) เช่น ไมโครโฟนหรือลำโพง เป็นต้น แอปพลิเคชันจะเลือกเทอร์มินัลที่มีขึ้นอยู่กับทิศทางของมิเดียและชนิด

ของเซสชันในการติดต่อสื่อสาร ข้อมูลแต่ละมิเดียที่เกี่ยวข้องก็จะถูกเลือกให้เข้ากับเทอร์มินัลที่เหมาะสม เพื่อที่จะเริ่มทำสตรีมมิ่ง

- **ออบเจ็คต์ Call**

ออบเจคต์ Call แสดงถึงการติดต่อของแอดเดรสระหว่างแอดเดรสโลคอลกับแอดเดรสอื่น ๆ การควบคุมการ Call ทั้งหมดจะถูกทำผ่านออบเจคต์ Call ซึ่ง ITBasicCallCntrol และ ITCallInfo เป็นอินเทอร์เฟซที่ใช้บ่อยที่สุดของออบเจคต์ Call อินเทอร์เฟซเหล่านี้อิมพลิเมนท์การทำงานและการร้องขอต่าง ๆ เช่น ขอพอยท์เตอร์ของอินเทอร์เฟซสำหรับข้อมูลของมิเดีย

- **ออบเจ็คต์ CallHub**

ออบเจคต์ CallHub แสดงถึงลักษณะการมองของเธิร์ดพาร์ตี้ของการ Call จากผู้ผลิตหลายรายอินเทอร์เฟซและเมธอดที่ เกี่ยวข้องด้วยจะรับและเซ็ตข้ อมูลที่เกี่ยวข้องกับศูนย์กลาง (hub) เช่นมันกำลังทำงานอยู่หรือไม่ การใช้ออบเจคต์ CallHub ผู้ใช้ที่ต้องการความปลอดภัยของข้อมูลสามารถพบและควบคุมผู้สนทนาคนอื่น ๆ ในการ Call ได้ ออบเจคต์ CallHub ไม่สามารถถูกสร้างขึ้นโดยตรงได้โดยแอปพลิเคชัน แต่จะถูกสร้างทางอ้อมเมื่อมีการ Call เข้ามาแล้วถูกรับผ่าน TAPI 3.0 ซึ่ง TAPI 3.0 อาศัยผู้ให้บริการTAPI เพื่อจัดหาข้อมูลที่จำเป็นเกี่ยวกับ Call ต่าง ๆ เพื่อที่จะนำมาอิมพลิเมนท์โดยใช้ออบเจคต์ CallHubเนื่องจากไม่ใช่ผู้ให้บริการทั้งหมดที่จะจัดหาข้อมูลเหล่านี้ให้และไม่ใช่ฮาร์ดแวร์ทั้งหมดที่จะสามารถติดตาม CallHub ได้ข้อมูลที่เกี่ยวกับผู้ใช้คนอื่น ๆ ในการติดต่ออาจจะถูกจำกัดหรือไม่มีอยู่เลย

- **ออบเจ็คต์ Stream**

ออบเจคต์ Stream เป็นแอบสแทรกต์ของข้อมูลมิเดียหรือข้อมูลอื่น ๆ ที่เกี่ยวข้องกับเซสชันการ Call อินเทอร์เฟซและเมธอดต่าง ๆ ที่ถูกเปิดใช้ให้กับออบเจคต์สตรีมและสตรีมย่อย (substream) อนุญาตให้แอปพลิเคชันสามารถเข้าถึงการควบคุมในรายละเอียดได้ เช่น การหยุดสตรีมชั่วคราว เพิ่มชนิดของมิเดียให้กับเซสชันในการติดต่อสื่อสารหรือปรับระดับเสียงของผู่ร่วมสนทนา

2.2.3 อีเวนต์ใน TAPI

อีเวนต์เป็นส่วนที่สำคัญการควบคุมการ Call ภายใต้ TAPI 3.0 ซึ่งประกอบด้วย 4 ขั้นตอน โดยในการลงทะเบียนและรองรับการทำงานของอีเวนต์จะต้องทำดังนี้

1. ทำการอิมพลิเมนท์เมธอด ITTAPIEventNotification:Event (TAPI จะ call เมธอดนี้เมื่อมีอีเวนต์เกิดขึ้น) โดยปกติแล้วการอิมพลิเมนท์นี้ ไม่ได้ทำไปมากกว่าการทำ AddRef กับพอยท์เตอร์อินเตอร์เฟซของ IDispatch แล้วจึงส่งให้กับตัวสร้างเมสเซจของแอปพลิเคชัน

2. ลงทะเบียนอินเตอร์เฟซ ITTAPIEventNotification ที่ออกไปโดยใช้ มาตรฐาน COM ของอินเตอร์เฟซ IConnectionPointContainer และ IConnectionPoint แล้วทำการส่งพอยท์เตอร์ของเมธอด ConnectionPointContainer ::Advise ไปยัง ITTAPIEventNotification::Event

3. ทำการเรียกเมธอด ITTAPI::put_EventFilter เพื่อที่จะบอก TAPI ว่าอีเวนต์ไหนบ้างที่แอปพลิเคชันจะรองรับ ตัวกรองอีเวนต์ (event filter) จะประกอบไปด้วยสมาชิก ORed ของตัวระบุ TAPI_EVENT ซึ่งจะต้องทำการเรียกเมธอด ITTAPI::put_EventFilter เพื่อที่จะเซ็ตตัวกรองอีเวนต์และ

รองรับการทำงานของอีเวนต์ ถ้าหากไม่ทำการเรียก ITTAPI::put_EventFilter แล้ว ตัวแอปพลิเคชันจะไม่สามารถรับอีเวนต์ใด ๆ ได้เลย

ตัวอย่างอีเวนต์ของ TAPI

TE_CALLNOTIFICATION

เกิดเซสชันการติดต่อเข้ามาใหม่ในแอดเดรส และ TAPI DLL ได้สร้าง Call object ขึ้นมาใหม่อิเวนต์นี้สามารถเป็นผลของการที่มีสัญญาณเข้า, เกิดการแฮนด์ออฟหรือมีการปาร์คแอดเดรสก็ได้

TE_CALLMEDIA

ตัวกลางที่เกี่ยวข้องกับการโทรศัพท์มีการเปลี่ยนแปลง

TE_CALLSTATE

มีการเปลี่ยนแปลงสถานการณ์โทรศัพท์

2.3 DirectX

เป็นกลุ่มเทคโนโลยีที่ได้รับการออกแบบโดยไมโครซอฟท์ เพื่อทำให้การทำงานบนพื้นฐานของ Microsoft Windows มีรูปแบบการทำงานเหนือความเป็นจริงในการรันและแสดงผลงานเช่นเกมส์ที่มีคุณสมบัติการทำงานแบบอิงเวลาที่เป็นจริง, full-color, ระบบแสดงผลสามมิติ, วีดีโอ, ดนตรีและระบบเสียงรอบทิศทาง ไมโครซอฟท์ได้สร้างไดเรกเอ็กซ์ขึ้นมาใช้กับระบบปฏิบัติการตระกูลวินโดว์โดยตรง เมื่อมีความจำเป็นต้องใช้การทำงานของไดเรกเอ็กซ์ คอมโพแนนท์ของไดเรกเอ็กซ์จะมีความสามารถในการอัพเดทได้เองบนระบบปฏิบัติการโดยอัตโนมัติ เช่นเกมส์เวอร์ชันล่าสุด หรือ งานทางด้านมัลติมิเดียจะเป็นส่วนหนึ่งของการลงโปรแกรม (Installation)

ไดเรกเอ็กซ์ เกิดจากกลุ่มของ APIs (Application Programming Interfaces) ที่ทำให้นักพัฒนาโปรแกรมได้รับการพัฒนาการเข้าถึงลักษณะพิเศษที่มีความก้าวหน้าของฮาร์ดแวร์ เช่น การ์ดจอสามมิติ หรือ การ์ดเสียง APIs นี้จะมีหน้าที่ควบคุมสิ่งที่เรียกว่า "low-level functions" ซึ่งรวมถึงการจัดการหน่วยความจำของกราฟฟิกและการปรับรูปภาพด้วย นั่นคือ สนับสนุนสำหรับอุปกรณ์อินพุต เช่น joysticks, คีย์บอร์ดและเมาส์ รวมทั้งยังควบคุมการทำงานของการนำเสียงมารวมกัน (mixing sound) และการแสดงเสียง(sound output) ซึ่ง low-level functions นี้ ได้ถูกรวมไว้ในคอมโพแนนท์ในการสร้างไดเรกเอ็กซ์ต่อไปนี้

2.3.1 ส่วนประกอบของไดเร็กเอ็กซ์

ไดเร็กเอ็กซ์จะประกอบไปด้วยชุดของคำสั่ง API ซึ่งเตรียมไว้ให้นักพัฒนาสามารถที่จะเข้าถึงการทำงานของฮาร์ดแวร์ที่ติดตั้งอยู่บนเครื่องคอมพิวเตอร์อย่างประสิทธิภาพได้โดยตรง ซึ่งชุดคำสั่ง API เหล่านี้จะทำหน้าที่ควบคุมการทำงานของอุปกรณ์ต่างๆ ซึ่งแบ่งออกเป็นส่วนประกอบต่างๆ ดังต่อไปนี้

- DirectX Graphics

ทำหน้าที่ดูแลและจัดการการแสดงผลทั้งภาพ 2 มิติและ 3 มิติ ช่วยในการให้แสง จัดวางมุมกล้องและทำภาพเคลื่อนไหว ซึ่งจะทำการควบคุมอุปกรณ์แสดงผลและดึงความสามารถของฮาร์ดแวร์ให้เต็มประสิทธิภาพ

- DirectX Audio

ทำหน้าที่ดูแลและจัดการเกี่ยวกับเสียง ไม่ว่าจะเป็นเสียงแบบ 3 มิติหรือเสียงแบบเซอร์ราวด์ นอกจากนั้นยังช่วยในการทำเสียงเอฟเฟ็กต์ต่างๆ อีกด้วย

- DirectX Input

ทำหน้าที่ดูแลและจัดการเกี่ยวกับอุปกรณ์อินพุตที่ใช้ควบคุมต่างๆ เช่น คีย์บอร์ด, เมาส์ หรือจอยสติ๊ก เป็นต้น และยังสามารถทำงานกับอุปกรณ์ที่เป็นแบบป้อนกลับ (Force feedback) ได้อีกด้วย

- DirectX Play

ทำหน้าที่ดูแลและจัดการเกี่ยวกับการติดต่อระหว่างคอมพิวเตอร์ผ่านระบบเครือข่าย ไม่ว่าจะเป็นระบบแลนหรือผ่านโมเด็ม

- DirectX Show

ทำหน้าที่ดูแลและจัดการเกี่ยวกับการแสดงผลภาพวีดีโอ

- DirectX Setup

ทำหน้าที่รวบรวมแอปพลิเคชันต่างๆ ที่พัฒนาขึ้นมาด้วยไดเร็กเอ็กซ์ เพื่อให้สามารถแจกจ่ายไปยังบุคคลอื่นๆ ได้ง่าย

DirectX เหมือนเป็นตัวจัดหาอุปกรณ์ที่ช่วยให้นักพัฒนาโปรแกรมสามารถแสดงผลที่ดีที่สุดจากเครื่องที่ใช้ ซึ่งเห็นได้ชัดว่าจะช่วยจัดหากลไกสำหรับแอปพลิเคชันในการตัดสินความสามารถของฮาร์ดแวร์ของระบบที่ใช้ปัจจุบัน ดังนั้นนักพัฒนาโปรแกรมสามารถ optimal performance

ก่อนที่จะมีการใช้ DirectX นักพัฒนาโปรแกรมสร้างสรรค์งานทางด้านมัลติมีเดียสำหรับเครื่องพีซีจะต้อง customize โปรแกรม ดังนั้นโปรแกรมในสมัยก่อนจึงสามารถทำงานได้บนฮาร์ดแวร์หลายๆ ชนิดและตั้งค่าได้ต่างๆ กันไปบนการทำงานของวินโดว์ DirectX จัดหา Hardware Abstract Layer หรือเรียกสั้นๆว่า HAL เพื่อใช้ไดร์เวอร์ของตัวโปรแกรมในการติดต่อระหว่างเกมส์และฮาร์ดแวร์ของเครื่องคอมพิวเตอร์ ผลที่ได้ก็คือ นักพัฒนาโปรแกรมสามารถใช้องค์ประกอบของแบบไดเรกเอ็กซ์เพียงตัวเดียวในการสร้างสรรค์ผลงานที่สามารถเล่นได้บนหลายๆเครื่องคอมพิวเตอร์ที่มีลักษณะต่างๆกัน

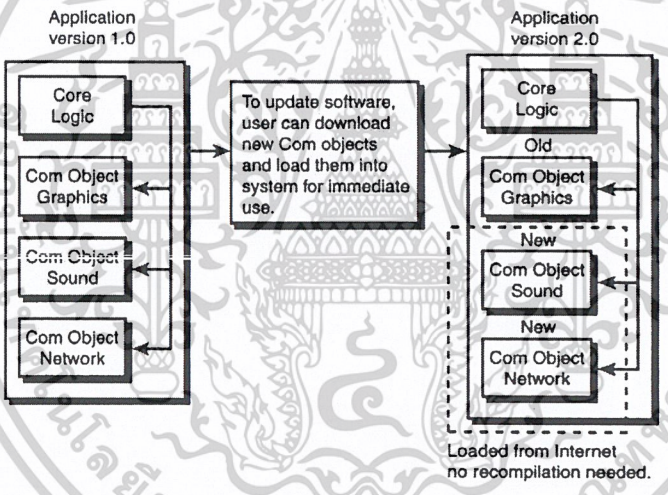
2.3.2 บทบาทของ COM

COM (Component Object Model) ได้กล่าวถึงรายละเอียดเกี่ยวกับวิธีการติดต่อสื่อสารกันระหว่างออบเจคต์ โดยข้ามกันระหว่างโพรเซสและข้ามผ่านเครือข่าย ระบบที่มีการเรียกใช้ COM ใช้งานจะถูกรับรองเป็นออบเจคต์อินสแตนซ์ (Object Instance)

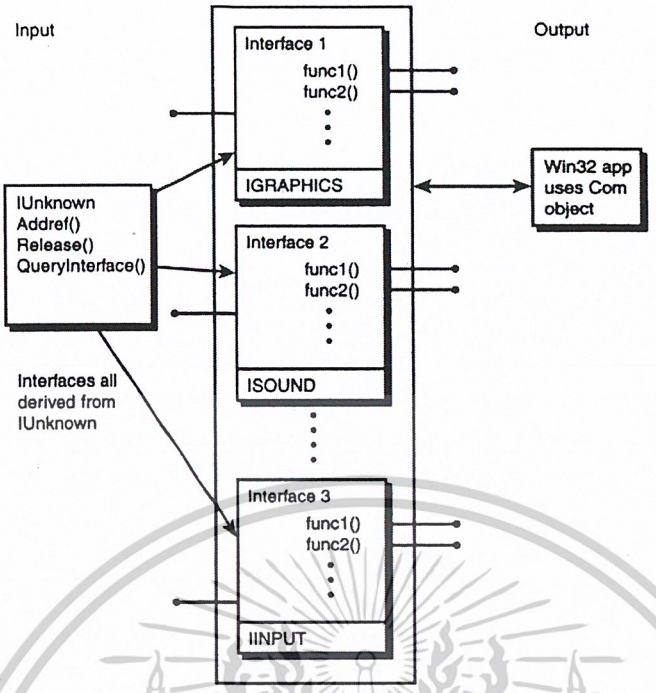
ทุกๆ COM ออบเจคต์จะต้องยึดติดกันจนเป็นโครงสร้างแบบไบนารี ซึ่งมีความคล้ายคลึงกับโครงสร้างของ virtual table ของ C++ ที่ถูกคอมไพล์แล้ว ดังนั้นจะทำให้ทุกภาษาสามารถที่จะสร้าง COM ออบเจคต์ได้ด้วยการจัดโครงสร้างให้เหมือนกับแบบไบนารีหลังจากการคอมไพล์

ทุกๆ COM ออบเจคต์นั้นยังจะต้องถูกสืบทอดมาจาก COM อินเตอร์เฟซมาตรฐาน ซึ่งเรียกว่า IUnknown โดย parent อินเตอร์เฟซจะมีอยู่ด้วยกัน 3 เมธอดคือ AddRef, Delete และ QueryInterface ซึ่ง 2 เมธอดแรก จะจัดการออบเจคต์ด้วยการอ้างอิงถึงจำนวน โดยการนับจำนวนของอินสแตนซ์ตั้งแต่เริ่มต้น ส่วนเมธอดสุดท้ายจะจัดการเกี่ยวกับการสอบถามกับ COM ออบเจคต์ตัวอื่นๆ สำหรับอินเตอร์เฟซที่ต้องการและอินเตอร์เฟซที่มีอยู่ จากนั้นจะคืนค่าพอยท์เตอร์ที่ชี้ไปที่อินเตอร์เฟซนั้น

COM ออบเจคต์ที่ทำการร้องขอนั้น เรียกว่า COM ไคลเอนต์และออบเจคต์ที่ตอบสนองการร้องขอนั้นคือ COM เซิร์ฟเวอร์ เมื่อไคลเอนต์ได้รับการเชื่อมต่อแล้วไคลเอนต์นั้นสามารถที่จะเรียกเมธอดใดก็ได้ใน COM เซิร์ฟเวอร์นั้น โดยใช้การอ้างอิงของอินเตอร์เฟซพอยท์เตอร์ ส่วนการเชื่อมต่ออื่นๆ จะออกไปเพื่อทำให้งานของออบเจคต์ที่เรียกเพื่อเข้ามาใช้งานง่ายขึ้น ในการเชื่อมต่อนี้จะมีเมธอดที่ทำการแสดงรายชื่อของเมธอดที่สนับสนุน เพื่อที่จะช่วยให้เมธอดสามารถเรียกใช้งานได้ ก่อนที่จะรวมเข้าไว้ด้วยกัน ด้วยเทคนิคนี้จะอนุญาตให้สามารถสร้างชนิดของไลบรารีใดๆ และแอปพลิเคชัน เช่น Visual Basic ซึ่งมีข้อดีในการสร้างคุณสมบัติของ COM โปรแกรมที่มีความทนทานต่อความผิดพลาด



รูปที่ 2-7 ภาพโดยรวมของ COM



รูปที่ 2-8 อินเตอร์เฟซของ COM ออบเจคต์

2.3.3 COM และไดเร็กเอ็กซ์

ทุกๆ อินเตอร์เฟซของไดเร็กเอ็กซ์นั้นจะทำการสืบทอดมาจาก IUnknown และมีพื้นฐานในการพัฒนาจาก COM ด้วยเช่นกัน

การใช้ COM ของไมโครซอฟท์ทำให้ได้ชุดคอมโพเน็นต์ของไดเร็กเอ็กซ์ที่มีประสิทธิภาพเพิ่มขึ้น ซึ่งสามารถเข้ากันได้กับไดเร็กเอ็กซ์เวอร์ชั่นก่อนๆ ได้อย่างสมบูรณ์ ด้วยความอิสระไม่ขึ้นกับภาษาที่ใช้เขียนโปรแกรมของ COM ทำให้นักพัฒนาสามารถที่จะตัดสินใจเลือกภาษาที่ใช้เขียนโปรแกรมตามที่เหมาะสมกับนักพัฒนาเองมากที่สุด ซึ่งอาจเป็นภาษาที่ถนัด และยังทำให้ได้ประสิทธิภาพตามที่ต้องการ นอกจากนั้นนักพัฒนายังสามารถที่จะเลือกภาษาอะไรก็ตามที่ COM สนับสนุน และสามารถพัฒนาสภาพแวดล้อมได้ตามที่นักพัฒนาต้องการ ด้วยโปรแกรมมิ่งโมเดลของ COM โปรแกรมเมอร์จะแน่ใจได้ว่าแม้ว่าผู้ใช้จะอัพเกรดหรือเปลี่ยนติดตั้งฮาร์ดแวร์ใหม่ หรือจะเปลี่ยนไดเร็กเอ็กซ์เวอร์ชั่นใหม่ก็ตาม ซอฟต์แวร์ก็ยังคงจะสามารถใช้งานได้ตามปกติ โดยไม่จำเป็นต้องทำการติดตั้งใหม่

ในความเป็นจริงไดรเวอร์ของอุปกรณ์บน Windows ทุกตัวจะรวมกันเป็น WDM (Windows Driver Model) ซึ่งโครงสร้างภายในก็เป็น COM คอมโพเน็นต์ ดังนั้นจึงมีการโต้ตอบที่ดีกว่าระหว่างไดรเวอร์โมเดลกับหน้าที่ของไดรเวอร์ ในสถาปัตยกรรมของไดเร็กเอ็กซ์นั้นทำให้แน่ใจได้ว่าการพัฒนาสำหรับนักพัฒนาเกมส์จะมีความสะดวกมากขึ้น และลดเวลาในการทดสอบลงด้วย

2.4 DirectPlay

Microsoft DirectPlay API เป็นมิเดียที่เป็นอิสระบนเครือข่ายของ API ที่มีการบริการในด้านเครือข่ายที่ transport protocol และ protocol session levels ซึ่งคำว่าเป็นมิเดียที่เป็นอิสระนั้นหมายถึง DirectPlay สามารถรันได้บน TCP/IP networks, IPX networks หรือแม้แต่การต่อตรงจากโมเด็มและ serial cables และการเป็นอิสระทางเครือข่ายนี้จะสามารถถูกพัฒนาต่อไปได้ในอนาคต นั่นคือ สามารถรองรับอุปกรณ์และโปรโตคอลใหม่ๆ เช่น IPV6, multicast หรือ คุณภาพของการบริการด้านโปรโตคอล นั่นหมายถึงว่างานทางด้าน DirectPlay สามารถพัฒนาได้ตลอดเวลา และการพัฒนามาตรฐานทางด้านเน็ตเวิร์กด้วย สรุปแล้ว หน้าที่การทำงานของ DirectPlay ก็คือความสามารถในการทำงานหรือการใช้ DirectPlay บนเครือข่ายนั่นเอง

2.4.1 ไดเร็กเพลย์โปรโตคอลสแต็ค

ไดเร็กเพลย์ได้กำหนดรายละเอียดในการทำงานในส่วนติดต่อกับเครือข่ายไว้ในโปรโตคอลสแต็ค 3 ชั้นได้แก่

- ชั้นของเสียงและเซสชัน ใช้ชั้นบนสุดของสแต็คร่วมกัน เมสเซจทั่วๆไปจะส่งผ่านมาในชั้นของเซสชันและเมสเซจที่กี่ยวข้องกับเสียงจะส่งไปยังชั้นของเสียง
- ชั้นทรานสปอร์ต เป็นชั้นกลางของสแต็ค ทั้งเสียงและเซสชัน จะต้องผ่านชั้นนี้ ชั้นนี้จะทำหน้าที่ในการแตกและรวมเมสเซจ และ ทำการส่งซ้ำในส่วนที่แพ็กเก็ตสูญหายไป
- ชั้นของส่วนที่ให้บริการ เป็นชั้นล่างสุดของสแต็ค ทุกๆเมสเซจที่เกี่ยวข้องกับเครือข่ายจะถูกจัดการด้วย ชั้นนี้ โดยจะใช้ Winsock ในการติดต่อกับชั้นของเครือข่าย

2.4.2 โปรโตคอลของไดเร็คเพลย์ในชั้นทรานสปอร์ต

หัวใจสำคัญของความสามารถทางด้านเครือข่ายของไดเร็กเพลย์นั้น คือ ไดเร็กเพลย์โปรโตคอล ในชั้นทรานสปอร์ตนี้ สามารถใช้ได้กับทุกเมสเซจ โปรโตคอลของไดเร็กเพลย์นั้นจะพยายามทำให้ใช้งานให้ง่ายในการที่จะส่งข้อมูลจากต้นทางไปยังปลายทางโดยจะไม่ต้องสนใจเลยว่าระหว่างทางนั้นจะเกิดอะไรขึ้นบ้าง ซึ่งจะต้องมีการกำหนดรายละเอียดของการทำงานในส่วนนี้โดยไดเร็กเพลย์นั้นได้มีการจัดสรรเมธอด ที่สามารถกำหนดการทำงานไว้ได้หลากหลายแบบ โดยไดเร็กเพลย์จะมีความสามารถในการทำงานในชั้นทราน-สปอร์ตดังนี้

- **เมสเซจที่ต้องการความเชื่อถือได้สูง:**
  เมสเซจที่เชื่อถือได้นั้นจะทำการส่งซ้ำไปเรื่อยๆจนกว่าที่ปลายทางจะได้รับ เราสามารถที่จะกำหนดประเภทของการส่งได้
- **เมสเซจที่มีการกำหนดลำดับ**
  เมสเซจที่มีลำดับนั้นจะถูกส่งตามลำดับจากทางต้นทาง
- **เมสเซจที่มีการแตกและรวม**
  ถ้าหากว่าเมสเซจมีขนาดใหญ่เกินที่ตัวกลางสามารถรับได้ ไดเร็กเพลย์จะทำการแตกเมสเซจออกและรวมเมสเซจให้โดยอัติโนมัติ

- **การควบคุมความหนาแน่นของข้อมูล**

ไดเร็กเพลย์จะทำการจัดการเมสเซจที่จะออกจากแอปพลิเคชันให้สามารถออกไปได้อย่างพอดีลักษณะเด่นอีกประการก็คือสามารถที่จะป้องกันการส่งเมสเซจที่มากเกินกว่าที่จะประมวลผลได้

- **การส่งระดับความสำคัญ**

เพื่อให้มั่นใจได้ว่าเมสเซจที่มีความสำคัญเป็นพิเศษจะสามารถส่งถึงและรับได้เป็นอันดับแรกได้เร็กเพลย์จะอนุญาตให้เราสามารถออกแบบเมสเซจให้มีระดับความสำคัญเป็น สูง กลางหรือต่ำได้ โดยเมสเซจที่มีระดับควาสำคัญสูงนั้นจะถูกส่งออกไปเป็นอันดับแรกของคิว และตามด้วย กลาง และ ต่ำ ตามลำดับ

- **อายุของเมสเซจ**

เพื่อป้องกันความหนาแน่นเกินไปในคิวของเมสเซจที่จะออกไป จึงมีการกำหนดอายุของเมสเซจไว้ ถ้าหากว่าหมดเวลาไปแล้วนั้นก็จะทำการนำเอาเมสเซจนั้นออกไปจากคิวโดยอัตโนมัติ โดยไม่คำนึงถึงว่าจะส่งไปแล้วหรือไม่

2.4.3 การติดต่อสื่อสารของไดเร็กเพลย์ (DirectPlay Network Communication)

ฟังก์ชันหลักของไดเร็กเพลย์ คือการจัดการการสื่อสารระหว่างคอมพิวเตอร์ด้วยเมสเซจที่มีความยืดหยุ่นและประสิทธิภาพสูงที่สามารถรองรับแอปพลิเคชันจากเครือข่าย ฮาร์ดแวร์และซอฟท์แวร์ ถ้าต้องการส่งการเปลี่ยนแปลงสถานะ สามารถทำได้อย่างง่ายโดยการเรียก API ของไดเร็กเพลย์ โดยไม่คำนึงถึงว่าใช้เครือข่ายชนิดใดอยู่ DirectPlay network service provider รองรับการติดต่อบน TCP/IP IPX โมเด็ม และการเชื่อมต่อจาก serial แต่ว่าไดเร็กเพลย์จะไม่สนับสนุนด้านความปลอดภัยของการติดต่อ

- **DirectPlay Addresses**

ในการส่งเมสเซจนั้น ผู้ใช้แต่ละคนจะมีแอดเดรสที่ไม่ซ้ำกัน แอดเดรสนั้นจะสามารถอ้างถึงแอปพลิเคชันที่เราทำงานอยู่บนอะไร(Device address) หรือ คอมพิวเตอร์ที่เราต้องการที่จะติดต่อไปหา (Host address) แอดเดรสของไดเร็กเพลย์นั้นจะอยู่ในรูปแบบของ URL string ซึ่งสตริงจะประกอบไปด้วยโครงสร้าง, ตัวแยกโครงสร้างและข้อมูล โดยมีรูปแบบทั่วไปดังต่อไปนี้

X - directplay: / [data string]

สตริงของข้อมูลนั้นจะบรรจุส่วนต่างๆเอาไว้เพื่อที่จะทำการกำหนดทุกอย่าง ที่จำเป็นในการติดต่อระหว่างผู้ส่งและเป้าหมาย โดยอยู่บนเครือข่ายต่างๆ

ในการใช้งานนั้น URL string จะถูกฝังเข้าไปใน ออบเจคต์ไดเร็กเพลย์แอดเดรส ที่มาจาก เมดธอดของ ไดเร็กเพลย์ API เรามีตัวเลือกที่จะจัดการกับ URL string โดยตรง หรือเรียกใช้ผ่านเมดธอด

- **การติดต่อกับออบเจคต์ของไดเร็กเพลย์:**

ไดเร็กเพลย์นั้นมีจุดสำคัญอยู่ที่เป็นการประกอบกันของ ออบเจคต์ของ COM โดยแต่ละออบเจคต์จะเปิดเผยอินเตอร์เฟตออกมาให้เราเรียกใช้งาน ยกตัวอย่างเช่น การส่งข้อมูลไปยังอีกคนแบบ Peer-to-peer นั้น เราจะต้องส่งเมสเซจโดยการเรียกเมดธอด IDirectPlay8Peer::SendTo โดยที่ไดเร็กเพลย์จะทำการดูแลการส่งเมสเซจไปยังเป้าหมายเอง

ไดเร็กเพลย์จะทำการติดต่อกับแอปพลิเคชันของเราผ่าน Callback function ฟังก์ชันเหล่านี้จะมีลักษณะคล้ายกับ Windows procedure แอปพลิเคชันของเราจะทำการอิมพลิเมนท์ callback function นี้และส่งค่าพอยท์เตอร์ผ่านฟังก์ชันนั้นไปยังไดเร็กเพลย์ และเมื่อไดเร็กเพลย์ต้องการจะติดต่อกลับมายังแอปพลิเคชันของเรานั้นก็จะติดต่อมาทาง Callback Function โดยจะเรียก Callback Function และทำการส่งค่าเข้ามา 2 ค่าคือ

- หมายลขเมสเซจ ที่จะเป็นตัวระบุชนิดของเมสเซจ
- พอยท์เตอร์ที่ชี้ไปยัง Block ของข้อมูล ที่เก็บโครงสร้าง และข้อมูลที่เรา

ยกตัวอย่างเช่นเมื่อเมสเซจส่งมาแบบตัวอย่างนานบน ส่งถึงเป้าหมายแล้ว callback function ของแอปพลิเคชันเป้าหมาย จะรับเมสเซจด้วย หมายเลขเมสเซจที่ชื่อว่า DPNMSGID_RECEIVE เพื่อบอกว่าเมสเซจนี้มาจากผู้ใช้คน

เพราะว่าการจัดการส่งเมสเซจของไดเร็กเพลย์นั้นเป็นแบบมัลติเทรด จึงใช้ Callback Function ในการอิมพลิเมนท์

2.4.4 DirectPlay Lobby Support

ลอบบี้เป็นแอปพลิเคชันที่มีจุดประสงค์หลักเพื่อช่วยให้ผู้ใช้จัดการกับเกมส์ที่มีผู้เล่นหลายคนได้ โดยลอบบี้จะใช้สำหรับคอมพิวเตอร์ที่ติดต่อกับเซิร์ฟเวอร์ที่อยู่ระยะไกล(Remote Server) และผู้ใช้สามารถเข้าถึงลอบบี้ผ่านทางอินเตอร์เน็ต โดยการสร้างเซสชันที่เซิร์ฟเวอร์เอง หรือเข้าร่วมเซสชันที่ผู้ใช้อื่นสร้างไว้ก็ได้ ซึ่งลอบบี้จะเข้ามาจัดการสำหรับเกมส์ที่เล่นกันเป็นกลุ่ม

โดยปกติในการทำให้เกมส์ติดต่อกับลอบบี้ได้นั้นจะประกอบไปด้วยส่วนประกอบ 3 ส่วนดังนี้

- lobby server
- lobby client
- lobbyable game

Microsoft® DirectPlay® ไม่ได้ระบุลงไปว่าเราจะอิมพลิเมนท์แอปพลิเคชันลอบบี้เซิร์ฟเวอร์อย่างไร แต่ว่าไดเร็กเพลย์กลับไปสนับสนุนลอบบี้ไคลแอนท์ ลอบบี้ไคลแอนท์เป็นแอปพลิเคชันที่ถูกอิมพลิเมนท์โดยผู้ขาย ลอบบี้เซิร์ฟเวอร์ และถูกลงไว้ในระบบของผู้ใช้แต่ละคน และจะทำตัวเป็นตัวเชื่อมระหว่างผู้ใช้กับลอบบี้ ในขณะที่เราจัดการกับการติดต่อสื่อสารโดยตรงนั้น เราจะต้องรู้ถึงรายละเอียดในการอิมพลิเมนท์ของทุกๆลอบบี้ที่จะเกิดขึ้นในเกมส์ของเราด้วย

แอปพลิเคชันของลอบบี้ไคลแอนท์นั้นจะจัดการเกี่ยวกับรายละเอียดในการติดต่อกับ ลอบบี้เซิร์ฟเวอร์ โดยใช้โปรโตคอลที่เหมาะสม ลอบบี้ไคลแอนท์ทำการติดต่อกับผู้ใช้และแอปพลิเคชันเกมส์

ผ่านทางอินเตอร์เฟซของไดเร็กเพลย์ ไดเร็กเพลย์จะทำการส่งเมสเซจไปยังแอปพลิเคชัน และแอปพลิเคชันก็สามารถส่งค่าไปยังลอบบี้ผ่านทาง อินเตอร์เฟซของไดเร็กเพลย์ได้เช่นเดียวกัน

- **DirectPlay Lobby Architecture**

ขั้นตอนของการบริหารและการจัดการเซสชันของเกมส์ที่มีผู้เล่นหลายคนนี้ใช้ไดเร็กเพลย์ในการจัดการ โดยมีส่วนประกอบ 5 ส่วนที่สำคัญ คือ

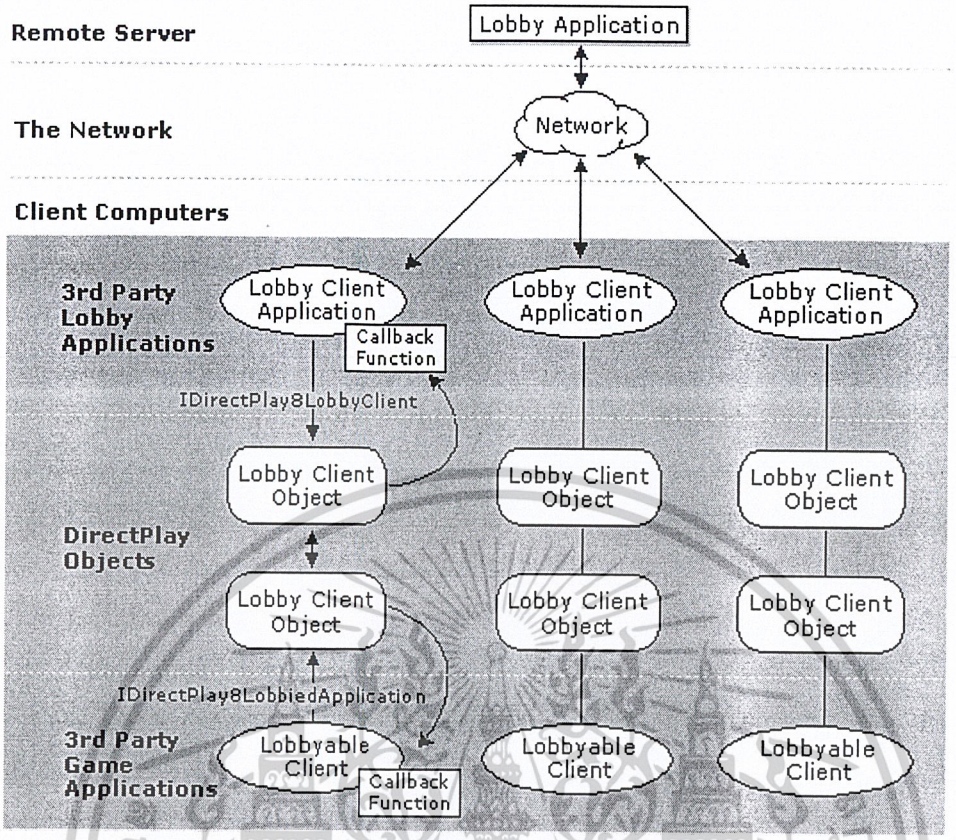
ซึ่งผู้เล่นจะต้องทำการติดตั้ง ส่วนประกอบอื่นๆลงในคอมพิวเตอร์อีก 4 ส่วน ดังจะกล่าวในรายระเอียดต่อไป

- *Lobby Sever* เป็นแอปพลิเคชันประเภท Third Party โดยจะตั้งอยู่บนเซิร์ฟเวอร์ระยะไกล(remote server) จะทำการติดต่อโดยผ่านทาง อินเตอร์เน็ต

อีก 4 ส่วน จะถูกติดตั้งบนคอมพิวเตอร์ของผู้เล่น ดังนี้

- *Lobby client* เป็นแอปพลิเคชันประเภท third Party เช่นกัน ที่ทำการสื่อสารกับ lobby sever โดย ผู้เล่นเกมส์จะทำการสื่อสารกับ Lobby server โดยใช้ DirectPlay lobby client object
- *Lobbyable game application* เป็นแอปพลิเคชันที่อนุญาตให้เล่นเกมส์นี้ ก็ เป็นแบบ third party โดย ใช้ DrrectPlay Lobby ร่วมกับ lobby client และ lobby sever
- *DirectPlay lobby client object*
- *DirectPlay lobbied application object*

ทั้ง 2 ออปเจ็คของไดเร็กเพลย์จะเชื่อมต่อระหว่างเกมส์กับ Lobby client โดยการติดต่อระหว่างผู้เล่นแต่ละคนนั้นจะทำการติดต่อกันผ่าน ช่องสื่อสารส่วนตัวจากกราฟข้างล่างแสดงให้เห็นโครงสร้างการติดต่อและแสดงการติดต่อกันระหว่างแต่ละผู้เล่นแต่ละคน



รูปที่ 2-9 โครงสร้างของลอบบี้ในไดเร็กเพลย์

- **Lobby Server**

Lobby Server เป็นแอปพลิเคชัน ที่ใช้ในการอนุญาตให้ผู้เล่นแต่ละคนสามารถเข้าถึงเกมส์ได้ส่วนใหญ่ จะถูกติดตั้งอยู่บนคอมพิวเตอร์ ที่สามารถเข้าถึงได้โดยอินเตอร์เน็ตระยะไกล นอกจากนี้ Lobby Server ยังมีฟังก์ชันการทำงานอื่นๆเช่น ห้อง สนทนา กระดานข่าว กระดานการซื้อขาย การจัดการเกมส์ที่มีผู้เล่นหลายคน Lobby server จะมีการ handles เมสเซจหลายส่วนด้วยกัน ดังนี้

- การจัดการ network address ของผู้เล่นและเกมส์
- จัดการเซสชันโดยการเข้าไปดูแลความสัมพันธ์ในเกมส์ของแต่ละผู้ใช้ที่เข้ามาเล่นเกมส์
- สามารถอนุญาต ในผู้เล่นรายใหม่ สามารถ เข้ามาเล่นเกมส์เพิ่มได้อย่างต่อเนื่อง
- ทำการติดต่อคอมพิวเตอร์ทุกตัวให้ทำงานเข้ากับ network เดียวกัน
- สามารถจดจำสถานะของผู้เล่นได้ว่าผู้เล่นคนใดเปลี่ยนฉาก หรือว่าหยุดเล่น หรือว่าเปลี่ยน เครื่องเล่น

รายละเอียดของโปรแกรมประยุกต์ lobby sever นั้นจะไม่ขึ้นอยู่กับ ประเภทของเซิร์ฟเวอร์ โดย Microsoft® DirectPlay ไม่ได้กำหนดวิธีที่ตายตัวในการอิมพลิเมนท์ Lobby server แต่ sever

ควรจะมีส่วนที่ทำการสนับสนุน การสื่อสารระหว่างคอมพิวเตอร์ อย่างไรก็ตามคอมพิวเตอร์ของผู้ใช้จะต้องสามารถรองรับ lobby Client และ DirectPlay ได้

- **Lobby Clients**

Lobby Client จะเป็นแอปพลิเคชันที่สนับสนุนการทำงานของ ของ lobby server โดยจะถูกติดตั้งบนเครื่องของผู้เล่นแต่ละเครื่อง โดย lobby Client จะประกอบด้วยส่วนที่ ทำหน้าที่ ติดต่อระหว่างผู้เล่น และเกมส์ และ การติดตั้งจะต้องดาวน์โหลดโปรแกรมจาก lobby server โดย เมื่อทำการดาวน์โหลดเสร็จ ก็จะเป็นการ ติดตั้ง lobby Client โดยอัตโนมัติ

ขั้นตอน การติดตั้ง โดยทั่วไปเป็นดังนี้

1. ผู้เล่นเข้าไปที่ เว็บไซต์ แล้วทำการลงทะเบียน
2. ในระหว่าง การลงทะเบียน Lobby Client จะถูกดาวน์โหลดมาสู่ เครื่องของผู้เล่นโดยอัตโนมัติ
3. ผู้เล่นจะสามารถเลือกเกมส์ที่จะเล่นและ เลือก Join ไปที่เกมส์ๆนั้น
4. เริ่มการทำงานของ Lobby client ที่คอมพิวเตอร์ของผู้ใช้ โดยทั่วไปจะทำการชี้ไปที่ตำแหน่ง URL ที่ Lobby client ดำเนินการอยู่ และ Lobby Client จะทำการ เริ่มเกมส์ และจะทำการจดจำฉาก และเวลาการเริ่มเล่น
5. ถ้า เกมส์ ยังไม่หมดเวลา lobby client จะทำการติดต่อกับเซิร์ฟเวอร์ตลอด เพื่อที่ จะบันทึกว่าผู้เล่น เล่นอยู่ฉากไหน เข้าออก เมื่อไร หรือเปลี่ยนเครื่องเมื่อไร

การติดต่อสื่อสารระหว่าง Lobby Client

การติดต่อสื่อสารระหว่าง Lobby Client ส่วนใหญ่จะเป็นแบบทางอ้อม หากแอปพลิเคชันมีบางอย่างที่มีผลต่อ Lobby Client ก็จะส่งเมสเซจไปให้ไคลแอนท์แทน ซึ่งจะมี 2 methods ในการที่จะใช้ส่งเมสเซจให้ไคลแอนท์ คือ

IDirectPlay8LobbiedApplication::UpdateStatus

IDirectPlay8LobbiedApplication::Send

จะต้องทำการติดต่อกับ lobby Client เมื่อเกิดการเปลี่ยนแปลงดังต่อไปนี้

- เมื่อทำการติดต่อ ระหว่าง session
- เมื่อหยุดทำการติดต่อ ระหว่าง session
- เมื่อหลุดการติดต่อ ระหว่าง session
- เมื่อสิ้นสุดการติดต่อ ระหว่างsession
- เมื่อทำการเปลี่ยน host session

- **Lobbyable Applications**

Lobbyable Application ได้ถูกออกแบบให้ทำงานร่วมกันไคลแอนท์ ซึ่งจะมีประโยชน์อย่างมากต่อไคลแอนท์ดังนี้

- Lobby client จะได้รับการอัพเดตโดย อัตโนมัติ ทันทีที่ สถานะของเกมส์เปลี่ยนไป
- Lobby client สามารถใช้ มาตรฐาน API ในการ ติดต่อ สื่อสาร กับ โปรแกรมเกมส์
- โปรแกรมเกมส์ สามารถใช้ มาตรฐาน API ในการ ติดต่อ สื่อสาร กับ Lobby client

การจัดการของ Lobbyable Application

สิ่งแรกสุดที่ Lobbyable Applecaion จะต้องจัดการคือการสร้าง Lobbied application object และจะต้องทำการอิมพลิเมนท์ฟังก์ชันของการ handle เมสเซจที่รับจากไคลแอนท์ว่าจะจัดการอะไร ซึ่งมีส่วนสำคัญที่ต้องจัดการตามลำดับดังนี้

- สร้าง Lobbied application object
- Initialize ออบเจคต์ที่สร้างขึ้นมา
- ถ้าสามารถ initial ได้ (รีเทิร์นค่าที่ถูกต้อง) แอปพลิเคชันก็จะถูกควบคุมโดย Lobbyable Application
- ในการ initial จะมีการรีเทิร์นค่าต่างๆของผู้ใช้(user context) ซึ่งจะต้องประกอบด้วยข้อมูลของการเล่นเกมส์ที่มาจาก Lobby Client
- เมสเซจที่ใช้ในการติดต่อจะรับผ่านการ handler เมสเซจของ Lobby application ซึ่งเมสเซจจะประกอบด้วยข้อมูลที่หลายหลาย อาธิเช่น ไอดีของผู้ส่ง หรือข้อมูลอย่างอื่นที่ไคลแอนท์ต้องการส่ง

เมื่อทำการจัดการกับแอปพลิเคชันของลอบบี้เสร็จสมบูรณ์ ไดเร็กเพลย์จะมีการส่งเมสเซจเพื่ออัปเดทสถานะของ Lobby Client โดยอัตโนมัติ และเราสามารถใช้ Lobby application ในการติดต่อกับไคลแอนท์ได้เองโดยการส่งเมสเซจให้กับ Lobby Client

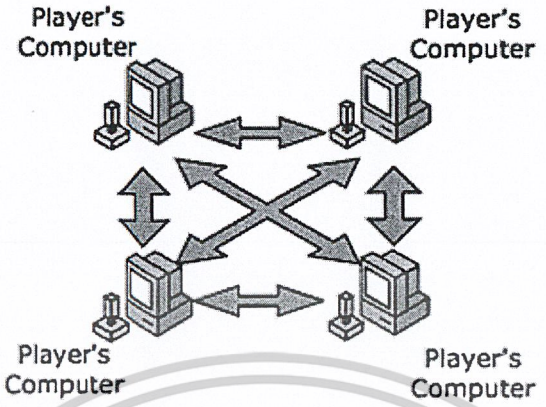
ข้อควรระวังในการ handler เมสเซจคือ การได้รับเมสเซจที่จาก Lobby Client ก่อนที่จะมี initial หรือได้รับเมสเซจในช่วงที่มีการเปลี่ยนแปลงการติดต่อ หรือ หยุดการติดต่อ

2.4.5 โครงสร้างการติดต่อของไดเร็กเพลย์

- **Peer-to-peerTopology**

ในรูปแบบ Peer-to-peerนั้นไคลแอนท์ทุกตัวที่อยู่ในเซสชัน จำทำการเชื่อมต่อกันโดยตรงเครื่องต่อเครื่อง โดยไม่ต้องการเซิร์ฟเวอร์ที่อยู่ตรงกลางเลย ในรูปแบบนี้ไคลแอนท์ตัว

หนึ่งจะทำหน้าที่ในการดูแลการติดต่อ โดยไดเร็กเพลย์ นั้นรองรับการย้ายโฮสต์ นั่นก็คือถ้าหากโฮสต์ใดล่มไปก็จะให้ไคลแอนท์ที่อยู่ในการติดต่อนั้นเป็นโฮสต์



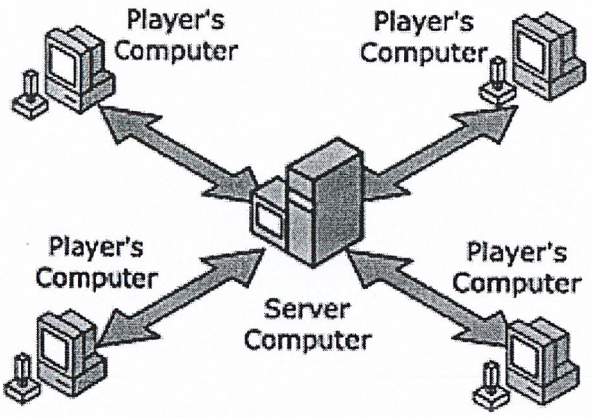
รูปที่ 2-10 โครงสร้างการติดต่อแบบ Peer-to-peer ของ DirectPlay

ถ้าหากว่ามีผู้เล่นที่มีการเปลี่ยนแปลงจะต้องส่งเมสเซจไปบอกกับทุกๆเครื่อง เมื่อมีเซสชันเกิดและรันขึ้นมา เมสเซจที่เกิดขึ้นมาจะถูกส่งไปแบบ คนต่อคน ถ้าหากว่า Lobby เซิร์ฟเวอร์มีการเรียกขึ้นมาจะมีการจับเมสเซจเฉพาะการเปลี่ยนแปลงที่เกิดขึ้นเท่านั้น

โครงสร้างแบบ Peer-to-peer นั้นมีข้อได้เปรียบอยู่ตรงที่มีความง่ายในการใช้งาน สิ่งที่ต้องการนั้นมีเพียงกลุ่มของไคลแอนท์ที่ใช้งาน และการสร้างเซสชันเท่านั้น แต่ข้อด้อยของแบบ peer-to-peer นั้นคือไม่เหมาะกับงานที่มีขนาดใหญ่ๆแทนเครือข่าย รูปแบบนี้เหมาะสำหรับจำนวนคนน้อยๆ ที่ไม่เกิน 64 คนแต่ที่ใช้กันอยู่ปัจจุบันจะอยู่ประมาณ 20-30 คน

- **Client-Server**

ในแบบนี้ไคลแอนท์จะไม่รู้ถึงการมีอยู่ของไคลแอนท์ตัวอื่นๆ ในเซสชัน และมีความสามารถเพียงแค่ส่งแพ็กเก็ตไปหาเซิร์ฟเวอร์ได้นั้น ถ้าไคลแอนท์ต้องการจะส่งข้อมูลไปยังไคลแอนท์ตัวอื่น จะต้องเริ่มด้วยการส่งแพ็กเก็ตไปยังเซิร์ฟเวอร์และเซิร์ฟเวอร์จะทำการส่งผ่านไปให้ โดยที่เซิร์ฟเวอร์สามารถจัดการได้พร้อมๆกันหลายไคลแอนท์ รูปแบบนี้เหมาะกับผู้ใช้ที่มีจำนวนมากๆ

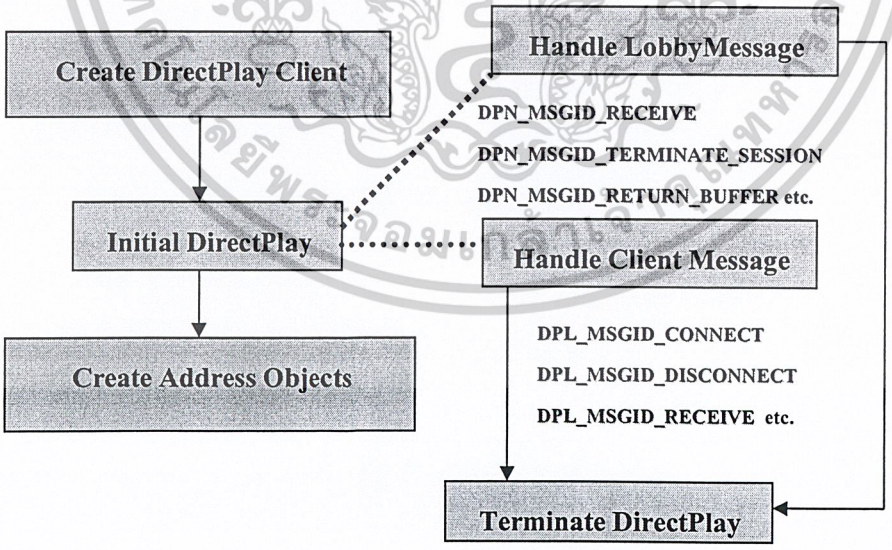


รูปที่ 2-11 โครงสร้างการติดต่อแบบ Client-Server ของ DirectPlay

ข้อดีของโครงสร้างแบบไคลแอนท์-เซิร์ฟเวอร์

- ประสิทธิภาพสูงเพราะว่า การที่มีผู้ใช้เพิ่มขึ้นมา 1 คน นั้นมีปริมาณเมสเซจเพิ่มมากขึ้นเป็นแนวเส้นตรง ทำให้การวางโครงสร้างแบบนี้มีความจำเป็นมากเมื่อมีผู้ใช้งานมาขึ้น
- เราจะไม่ต้องถูกจำกัดประสิทธิภาพของเครื่องโดยการต้องรับภาระในการจัดการติดต่อเอาไว้
- ถ้าหากเราจะมีการแก้ไขแอปพลิเคชันของเรานั้นจะไม่ต้องแก้ไขที่ไคลแอนท์จำนวนมากแต่เราจะแก้ที่เซิร์ฟเวอร์ทีเดียวไปเลย

2.4.6 การทำงานของของ DirectPlay Client



รูปที่ 2-12 **การทำงานของของ DirectPlay Client**

- **Handling Client Messages**

เมสเซจที่อยู่ในส่วนของไคลแอนท์-เซิร์ฟเวอร์จะมีลักษณะสำคัญดังต่อไปนี้

– ไคลเอนท์จะไม่มีการรับเมสเซจที่มีข้อมูลเกี่ยวกับผู้ใช้รายอื่นๆ หรือนอกกลุ่ม เพราะว่าไดเร็กเพลย์จะไม่อนุญาตให้ไคลแอนท์รู้ข้อมูล หรือ ติดต่อกันเอง

– ไดเร็กเพลย์จะไม่มีการส่งเมสเซจที่เป็นการย้ายโฮสต์ระหว่างการสื่อสารเพราะ เซิร์ฟเวอร์จะต้องเป็นโฮสต์ และไคลแอนท์จะไม่สามารถมีโฮสต์เป็นไคลเอนท์

เมสเซจที่สำคัญของไคลแอนท์ตามลำดับการทำงานมีดังนี้

1. Client Startup Messages
2. Client Messaging During Normal Game Play
3. Client Session Termination Messages

Client Startup Messages

เมื่อเราเลือกและกดเพื่อเข้าสู่การพูดคุยนั้น จะต้องมีการเลือกเซิร์ฟเวอร์ก่อน จากนั้นจะทำการติดต่อไปยังเซิร์ฟเวอร์และคอยรับเมสเซจที่เกิดจากการเริ่มต้นการติดต่อในการระบุและติดต่อไปยังเซิร์ฟเวอร์นั้นเราจะต้องทำการรอรับเมสเซจดังต่อไปนี้

DPN_MSGID_ENUM_HOSTS_RESPONSE

เพื่อระบุว่ามีเซิร์ฟเวอร์ใดที่ให้บริการอยู่บ้าง โดยการ *IDirectPlay8Client::EnumHosts* จากนั้นจะรองรับเมสเซจ *DPN_MSGID_ENUM_HOSTS_RESPONSE* จากทุกๆเซิร์ฟเวอร์ที่ตอบรับกลับมา

DPN_MSGID_CONNECT_COMPLETE

หลังจากเราทำการเลือกเซิร์ฟเวอร์และเรียก *IDirectPlay8Client::Co*nnect เพื่อพยายามเข้าไปสู่เซสชันของการติดต่อ เราจะต้องทำการรอรับเมสเซจ *DPN_MSGID_CONNECT_COMPLETE* ที่กลับมาพร้อมกับการตอบสนองของเซิร์ฟเวอร์

Client Messaging During Normal Game Play

เมสเซจที่เราจะได้รับเมื่อโปรแกรมดำเนินไปอย่างปกติ

DPN_MSGID_SEND_COMPLETE

เราจะส่งเมสเซจนี้ไปให้เซิร์ฟเวอร์โดยการเรียก *IDirectPlay8Client::Send* เหมือนกับแบบ peer-to-peerและเราจะได้รับเมสเซจ *DPN_MSGID_SEND_COMPLETE* เพื่อเป็นการบอกว่าเมสเซจของเรานั้นได้ถูกส่งเข้าไปแล้ว

DPN_MSGID_RECEIVE

เมื่อเซิร์ฟเวอร์ส่งข้อมูลมาให้เราจะเข้ามาอยู่ในเมสเซจ *DPN_MSGID_RECEIVE*

DPN_MSGID_APPLICATION_DESC, DPN_MSGID_SERVER_INFO

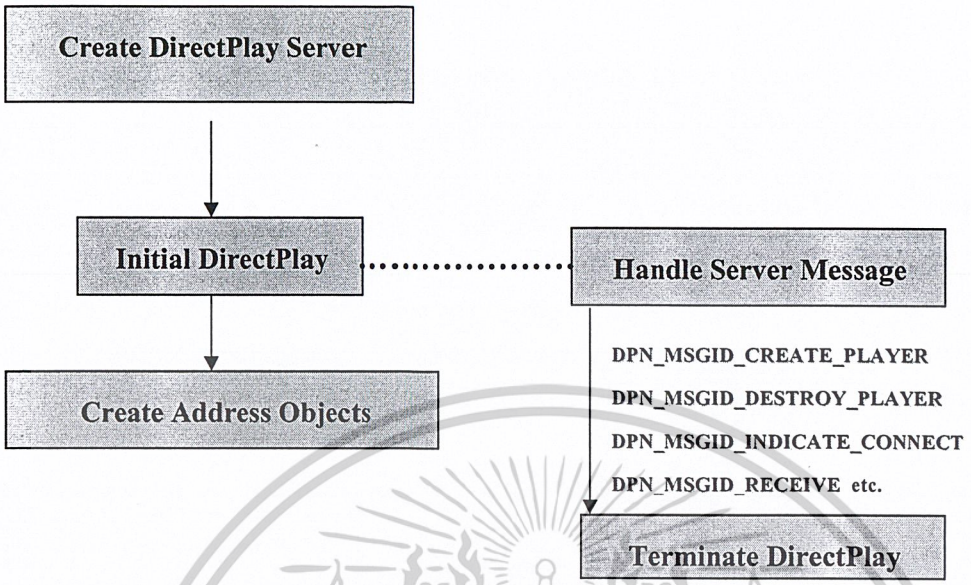
โครงสร้างของข้อมูลนั้นจะเกี่ยวข้องกับแอปพลิเคชัน และเซิร์ฟเวอร์ถ้าหากข้อมูลมีการเปลี่ยนแปลงระหว่างในเซสชันเราจะได้รับเมสเซจในการโต้ตอบกลับมา การที่จะร้องขอการเปลี่ยนแปลงแอปพลิเคชันและสถานะของเซิร์ฟเวอร์ทำได้โดยการเรียก *IDirectPlay8Client::GetApplicationDesc, IDirectPlay8Client::GetServerInfo.*

Client Session Termination Messages

เซสชันของไคลแอนท์-เซิร์ฟเวอร์จะจบลงอย่างปกติเมื่อเซิร์ฟเวอร์เรียก *IDirectPlay8Server::Close* เซสชันจะสามารถจบลงได้ก็ต่อเมื่อเซิร์ฟเวอร์ตัดการติดต่อไปแล้ว เมื่อเซสชันจบลง แต่ละไคลเอนท์จะได้รับเมสเซจดังต่อไปนี้

DPN_MSGID_TERMINATE_SESSION

2.4.7 การทำงานของของ DirectPlay Server



รูปที่ 2-13 การทำงานของของ DirectPlay Server

- **Handling Server Messages**

เมสเซจที่สำคัญของเซิร์ฟเวอร์ตามลำดับการทำงานมีดังนี้

1. Server Startup Messages
2. Server Messaging During Normal Game Play
3. Server Session Termination Messages

Server Startup Messages

เมื่อเราทำการให้เซิร์ฟเวอร์เริ่มทำงานนั้นเราจำเป็นที่จะต้องเตรียมรองรับเมสเซจดังต่อไปนี้

DPN_MSGID_ENUM_HOSTS_QUERY

ถ้าเราเป็นโฮสต์แบบบรอดคาสต์นั้น เราจะต้องจัดการเมสเซจ *DPN_MSGID_ENUM_HOSTS_QUERY* จากผู้เล่นที่พยายามหาโฮสต์ที่เหมาะสม และถึงแม้เราจะไม่ใช้เซิร์ฟเวอร์แบบบรอดคาสต์เราก็จะต้องเตรียมรอรับและจัดการกับเมสเซจนี้

DPN_MSGID_INDICATE_CONNECT

เราจะได้รับเมสเซจนี้ก็ต่อเมื่อผู้ใช้พยายามที่จะเข้ามาสู่เซสชันการติดต่อเราจะคืนค่า DPN_OK เพื่อรับการติดต่อและอนุญาตให้ผู้ใช้เข้ามาสู่เซสชันของเราและคืนค่าอะไรก็ได้ในการปฏิเสธการติดต่อ

DPN_MSGID_CREATE_PLAYER

เราจะได้รับเมสเซจนี้เมื่อเรายอมรับการติดต่อของผู้ใช้

DPN_MSGID_INDICATED_CONNECT_ABORTED

เราจะได้รับเมสเซจนี้เมื่อผู้ใช้ยกเลิกการติดต่อและเราส่งเมสเซจ *DPN_MSGID_INDICATE_CONNECT* แต่ก่อนที่เราจะส่งเมสเซจ *DPN_MSGID_CREATE_ PLAYER.*

Server Messaging During Normal Game Play

DPN_MSGID_CREATE_PLAYER และ *DPN_MSGID_DESTROY_PLAYER*

เราได้รับ *DPN_MSGID_CREATE_PLAYER* เมื่อผู้ใช้เข้ามาภายในแอปพลิเคชันและเมสเซจ *DPN_MSGID_DESTROY_PLAYER* เมื่อผู้ใช้นั้นออกจากแอปพลิเคชันไป

DPN_MSGID_CREATE_GROUP และ *DPN_MSGID_DESTROY_GROUP*

กลุ่มจะเห็นโดยเซิร์ฟเวอร์เท่านั้นเราจะได้รับเมสเซจ *DPN_MSGID_CREATE_GROUP* เมื่อเราสร้างกลุ่มและเมสเซจ *DPN_MSGID_DESTROY_ GROUP* เมื่อราลบกลุ่ม

DPN_MSGID_ADD_PLAYER_TO_GROUP และ *DPN_MSGID_REMOVE_PLAYER_ FROM_ GROUP*

เราจะได้รับเมสเซจ *DPN_MSGID_ADD_PLAYER_TO_GROUP* ทุกครั้งที่มีการเพิ่มผู้ใช้เข้ามาในกลุ่มและเมสเซจ *DPN_MSGID_REMOVE_PLAYER_FROM_GROUP* ทุกครั้งที่มีผู้ใช้ออกจากกลุ่มและเมื่อเราลบกลุ่มเราจะได้รับเมสเซจ *DPN_MSGID_REMOVE_ PLAYER_FROM_GROUP* สำหรับผู้ใช้ในกลุ่มที่เหลือทุกคน

DPN_MSGID_SEND_COMPLETE

เมื่อเราส่งเมสเซจไปหาไคลเอนท์โดยการเรียก *IDirectPlay8Server::SendTo* เราจะได้รับ เมสเซจ *DPN_MSGID_SEND_CO*MPLETE เพื่อบอกว่าเมสเซจของเราถูกส่งไปเรียบร้อยแล้ว

DPN_MSGID_RECEIVE

เมื่อไคลเอนท์ส่งข้อมูลมาหาเราจะถูกจัดการโดยเมสเซจ *DPN_MSGID_RECEIVE* *DPN_MSGID_APPLICATION_DESC,* *DPN_MSGID_CLIENT_INFO,* DPN_MSGID_ GROUP_INFO โครงสร้างข้อมูลจะเกี่ยวข้องกับแอปพลิเคชัน เซิร์ฟเวอร์และกลุ่มของเรา ถ้า

DPN_MSGID_APPLICATION_DESC, *DPN_MSGID_CLIENT_INFO*, DPN_MSGID_GROUP_INFO โครงสร้างข้อมูลจะเกี่ยวข้องกับแอปพลิเคชัน เซิร์ฟเวอร์และกลุ่มของเรา ถ้าข้อมูลนี้มีการเปลี่ยนแปลงระหว่างเซสชันการติดต่อ เราจะได้รับเมสเซจตอบสนอง การที่เราจะเรียกดูการเปลี่ยนแปลงสถานะนั้นจะต้องทำการเรียก *IDirectPlay8Client::GetApplicationDesc, IDirectPlay8Server::GetClientInfo* และ*IDirectPlay8Server::GetGroupInfo*

Server Session Termination Messages

หลังจากทำการปิดเซสชันแล้วเราจะได้รับเมสเซจดังต่อไปนี้

DPN_MSGID_DESTROY_PLAYER

เราจะได้รับเมสเซจ DPN_MSGID_DESTROY_PLAYERสำหรับผู้ใช้ที่อยู่ในเซสชันทั้งหมด

DPN_MSGID_DESTROY_GROUP และ*DPN_MSGID_REMOVE_PLAYER_FROM_GROUP*

เราจะได้รับเมสเซจ DPN_MSGID_REMOVE_PLAYER_FROM_GROUP สำหรับทุกๆผู้ใช้ภายในกลุ่ม และเมสเซจ DPN_MSGID_DESTROY_GROUP สำหรับทุกๆกลุ่ม

2.5 DirectPlayVoice

Microsoft® DirectPlay® Voice เป็น API ของการติดต่อโดยใช้เสียง ที่ใช้ไดเร็กเพลย์เพื่อจัดการในส่วนของการจัดการครือข่าย และการส่งผ่านเครือข่าย โดยไดเร็กเพลย์วอยส์จะไม่ทำการนำเอาส่วนประกอบต่างๆจากไดเร็กเพลย์มาใช้เพราะฉะนั้นก่อนที่เจะเรียกใช้ไดเร็กเพลย์วอยส์ได้เราจะต้องทำการสร้างการติดต่อที่เกิดจากไดเร็กเพลย์ขึ้นมาก่อน

ไดเร็กเพลย์วอยส์ได้ทำการรวมเอา ไดเร็กเพลย์ซาวน์มาใช้ในส่วนของการ เล่นและอัดเสียงและดึงเอาการทำงานของไดเร็กเพลย์ซาวน์ออดิโอมาทั้งหมด รวมถึงความสามารถในการเล่นเสียงแบบพิเศษต่างๆไม่ว่าจะเป็น การใช้เอฟเฟ็กหรือการเล่นเสียงสามมิติ

2.5.1 การติดต่อของ DirectPlay Voice

ในปัจจุบันนั้นแนวของเกมส์ส่วนใหญ่จะเป็นแนวของการเล่นเป็นทีม โดยที่มีการคุยกันของผู้เล่นเป็นส่วนประกอบของเกมส์ สมัยก่อนจะเป็นการคุยกันแบบพิมพ์ตัวหนังสือ เมื่อผู้เล่นพิมพ์แล้วก็จะส่งไปหาคนในทีม แต่ว่าแบบพิมพ์นั้นเหมาะกับการเล่นแบบเป็นตาๆ คือไม่ใช่แบบเวลาจริง เพราะถ้าเล่นแบบอิงเวลาจริงแล้วนั้นการพิมพ์จะช้าเกินไป

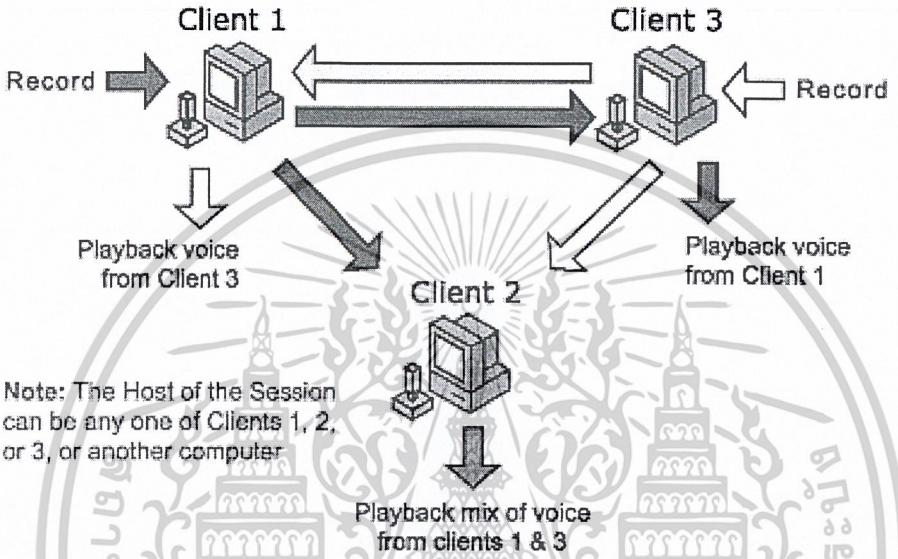
โดย Microsoft® Windows® platform ได้จัดเตรียมเครื่องมือเกี่ยวกับการทำงานในส่วนของการส่งเสียงแบบเวลาจริงแล้ว เหลือแต่เพียงว่าผู้พัฒนาจะเลือกใช้การบีบอัดเสียงแบบใด ที่จะสามารถจัดการได้ในส่วนของเครือข่ายที่มีช่องสัญญาณแคบ

โดย Microsoft® Windows® platform ได้จัดเตรียมเครื่องมือเกี่ยวกับการทำงานในส่วนของการส่งเสียงแบบเวลาจริงแล้ว เหลือแต่เพียงว่าผู้พัฒนาจะเลือกใช้การบีบอัดเสียงแบบใด ที่จะสามารถจัดการได้ในส่วนของเครือข่ายที่มีช่องสัญญาณแคบ

ไดเร็กเพลย์วอยซ์มีการติดต่อ 2 แบบ นั่นก็คือแบบ Peer-to-peerและ Client/Server ดังรูป

1. Peer-to-Peer

เป็นการสร้างการเชื่อมต่อระหว่างคอมพิวเตอร์โดยตรง ดังรูป

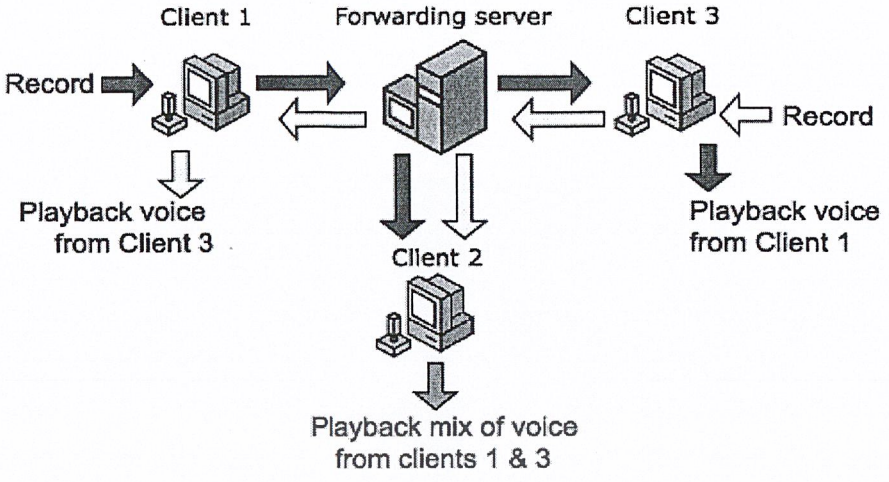


รูปที่ 2-14 การใช้DirectPlay ในการติดต่อแบบ Peer-to-Peer

2. Client/Server

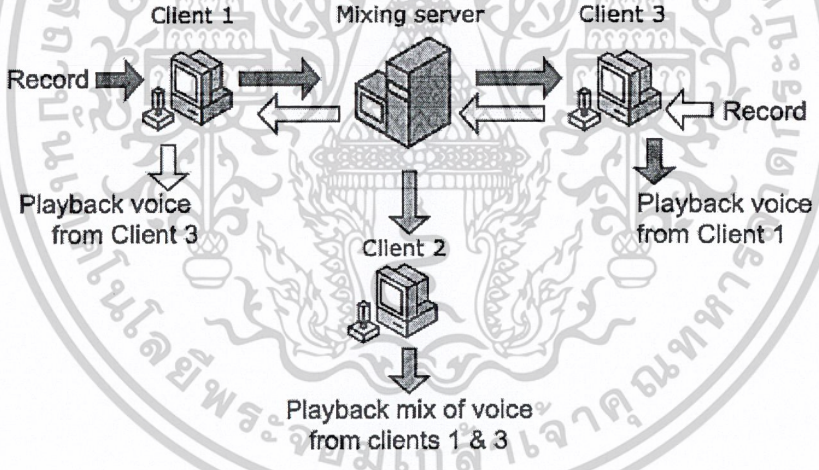
การติดต่อที่มีเซิร์ฟเวอร์เป็นตัวกลางซึ่งเซิร์ฟเวอร์จะมีอยู่ 2 ลักษณะ คือ

Forwarding Server คือ เมื่อได้รับเสียงจากไคลแอนท์แล้วจะส่งต่อไปยัง ไคลแอนท์อื่นๆเลย เมื่อใช้การทำงานของเซิร์ฟเวอร์แบบนี้จะทำให้หน้าที่ของการผสมเสียงและเล่นเสียงที่ได้รับ การเลือกตั้งเซิร์ฟเวอร์แบบนี้จะใช้Bandwidth สูงแต่เซิร์ฟเวอร์จะใช้ทรัพยากรของหน่วยประมวลผลกลางไม่มาก เพราะไม่ต้องทำการ Compress/Depress เสียง



รูปที่ 2-15 การใช้DirectPlay ในการติดต่อแบบ Forwarding Server

Mixing Server คือ เมื่อได้รับเสียงจากไคลแอนท์แล้วจะทำการผสมเสียง แล้วส่งไปให้กับไคลแอนท์ จึงทำให้มีการใช้หน่วยประมวลกลางสูง ส่วนเรื่อง Bandwidth ก็ต่ำกว่าแบบ Forwarding Server



รูปที่ 2-16 การใช้DirectPlay ในการติดต่อแบบ Mixing Server

2.5.2 Compression/Decompression Algorithms (CODECs)

วิธีการที่ใช้ในการบีบอัด และ คลายเสียงนั้น จัดหามาโดยไดเร็กเพลย์จะมีไว้เพื่อทำให้สามารถใช้แบนด์วิดท์ที่น้อยได้ Codec ทั้งหมดจะทำงานบน 8 kHz, 16 bit mono-format based data อย่างไรก็ตาม เราไม่สามารถที่จะใช้ Codec ที่เขียนเอง หรือ codec ที่นอกเหนือจากที่กำหนดไว้กับไดเร็กเพลย์วอยซ์ได้

CODECs จะไม่สามารถที่จะเปลี่ยนแปลงได้ในขณะที่กำลังใช้งานเซสชันอยู่ ผู้ใช้ที่ติดต่อกันจะต้องใช้ Codec เดียวกัน ถ้าหากว่าต้องทำการติดต่อกับ Codec ที่ต่างกันจะทำการตัดการติดต่อแล้วสร้างการเชื่อมต่อใหม่แล้วใช้ Codec ตัวเดียวกัน

การติดต่อผ่านเครือข่ายแบนด์วิดท์ถือเป็นเรื่องที่สำคัญ เพราะว่าเราไม่เพียงแต่จะส่งเสียงเท่านั้น แต่การติดต่อกันนั้นจะมีการส่งเมสเซจข้อมูล ซึ่งถ้าหากว่าเราไม่เตรียมไว้แต่ข้างต้น จะทำให้เกิดความล่าช้าในการส่งข้อมูลและการทำ Codec นั้นนอกเหนือจากการที่เราจะต้องคิดว่าจะต้องผ่านเครือข่ายนั้น การทำ Codec ก็จะกินประสิทธิภาพของ CPU ไปอีกด้วย

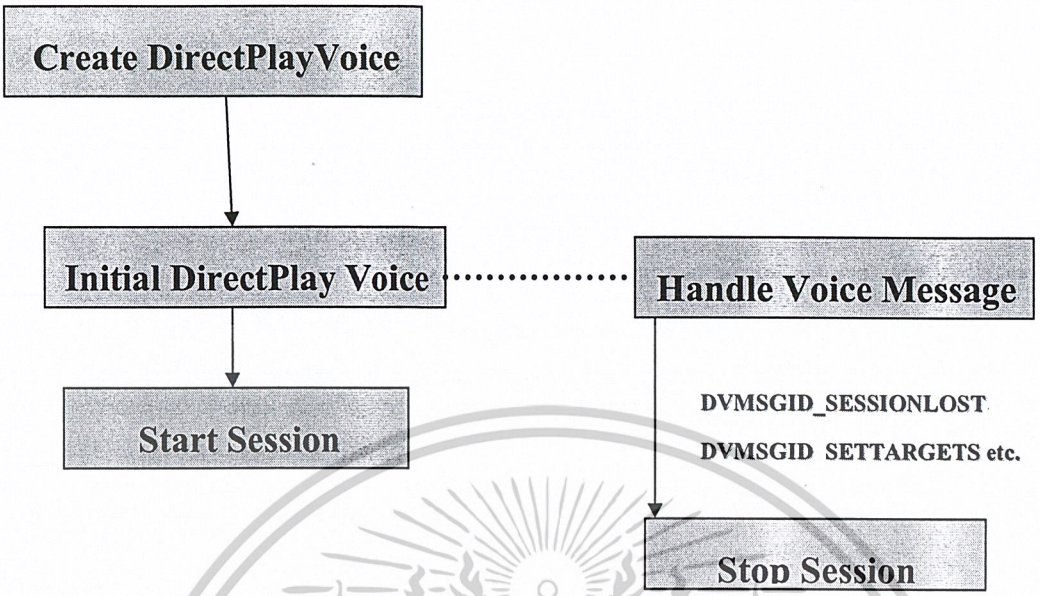
เป็นสิ่งสำคัญที่ต้องจำไว้ว่า ถ้าหากแบนด์วิดท์น้อย ก็จะทำให้คุณภาพของเสียงน้อยตามไปด้วย ตารางด้านล่างจะเป็น Codec ที่ไดเร็กเพลย์ยอมรับ แบนด์วิดท์จะอยู่ในหน่วยกิโลบิทต่อวินาทีและ GUID ก็เป็นการที่จะเลือก codec นั้นมาใช้

Codec	Bandwidth	GUID
Voxware VR12	Variable (1.2 Kbps, avg.)	DPVCTGUID_VR12
Voxware SC03	3.2 Kbps	DPVCTGUID_SC03
Voxware SC06	6.4 Kbps	DPVCTGUID_SC06
TrueSpeech	8 Kbps	DPVCTGUID_TRUESPEECH
Microsoft® GSM	13 Kbps	DPVCTGUID_GSM
Microsoft® ADPCM	32 Kbps	DPVCTGUID_ADPCM
Microsoft® PCM	64 Kbps	DPVCTGUID_NONE

ตารางที่ 2-1 ตารางแสดง codec bandwidth และ GUID ที่ให้เรียกใช้งาน

Codec 3 อันแรกนั้นจัดไว้ในการบีบอัดระดับสูง ที่เกือบจะเท่ากับปริมาณที่ทรัพยากรต้องการ บนคอมพิวเตอร์ 500 MHz Pentium III codec ใช้ประมาณ 1.5 % ของความสามารถของ CPU VR12 codec เสียงจะเล็กๆ และเหมือนกันหุ่นยนต์ SC03 และ SC06 จะใกล้เคียงกับความเป็นจริง PCM codec จะได้เสียงคุณภาพสูง และเหมือนกับไม่ได้ถูกบีบอัดเลย

2.5.3 การทำงานของของ DirectPlay Voice



รูปที่ 2-17 การทำงานของของ DirectPlay Voice

- **Handling Voice Client Messages**

ในส่วนนี้เราจะอธิบายในส่วนของการจัดการกับเมสเซจสำหรับไคลเอนท์ของ Microsoft® DirectPlay® Voice session. ซึ่งจะมีเมสเสจที่สำคัญตามลำดับการทำงานดังนี้

1. General Voice Messaging Considerations
2. Startup Messages
3. Messaging During Normal Game Play
4. Session Termination Messages

General Voice Messaging Considerations

เมสเซจหลายอันที่ถูกใช้โดย ไดเร็กเพลย์วอยซ์มีความเหมือนกันกับที่ใช้ใน ส่วนสำคัญของ DirectPlay API อย่างไรก็ตามไม่ว่าจะคล้ายกันอย่างไรก็ยังมีที่แตกต่างบ้าง ตัวอย่างเช่น เมื่อเราติดต่อเข้าไปในส่วนสำคัญของเซสชันของเสียงนั้น ไดเรกเพลย์จะคืนค่าเป็นเมสเซจที่สมบูรณ์ด้วยผลของการติดต่อ เราอาจจะได้รับเมสเซจ *DPN_MSGID_CONNECT_COMPLETE* ภายใต้การเปลี่ยนแปลงเมื่อคุณติดต่อเข้ามาแบบซิงโครนัส แต่แบบการติดต่อเข้ามาของ ไดเร็กเพลย์วอยซ์นั้นเราจะได้เมสเซจ *DVMSGID_CONNECTRESULT* ถ้าเราติดต่อแบบซิงโครนัส แต่เราจะได้เมสเซจแบบสมบูรณ์ถ้าเราติดต่อแบบ อซิงโครนัส

ข้อสังเกตอีกอย่างของการจัดการกับเมสเซจของส่วนหลักกับส่วนเสียงนั้นคือ ส่วนจัดการกับเมสเซจหลักนั้นจะต้องรับเมสเซจของส่วนหลักทุกเมสเซจแต่ถ้าเป็นส่วนจัดการเสียงนั้น เราจะสามารถเลือกที่จะรับเมสเซจได้โดยเรียกใช้คำสั่ง *IDirectPlayVoiceClient::SetNotifyMask*. เราจะได้รับเฉพาะเมสเซจที่เราเลือกเท่านั้นผ่านทางพารามิเตอร์ที่ชื่อว่า *pdwMessageMask* โดยที่รายการเมสเซจที่เราจะรับนั้นจะต้องมีอย่างน้อยเมสเซจหนึ่งชนิดที่ต้องเลือกไว้ เราจะไม่สามารถใช้คำสั่ง *IDirectPlayVoiceClient::SetNotifyMask* ในการที่จะเลือกว่าไม่รับเมสเซจใดๆเลย

Startup Messages

ไดเร็กเพลย์วอยซ์เป็นส่วนสริมที่เพิ่มขึ้นมาของ DirectPlay session.มันจะทำการทำให้การติดต่อโดยใช้เสียงระหว่างเซสชันนั้นสามารถใช้งานได้ ก่อนที่เราจะเป็นไคลเอนท์ในส่วนของการติดต่อด้วยเสียงนั้นเราจะต้องสร้างออบเจคต์ของไดเรกเพลย์ขึ้นมาและทำการติดต่อด้วยไดเรกเพลย์ก่อนไม่ว่าจะเป็นแบบ Peer-to-peer Sessions หรือ Client/Server Sessions

ทุกๆเซสชันของเสียงนั้นจะต้องมีโฮสต์ถ้าเป็นเซสชันแบบ Peer-to-peerนั้นหนึ่งในสมาชิกจะต้องตั้งตัวเองเป็นโฮสต์ของเสียง โดยที่คนที่จะเป็นโฮสต์ของเสียงนั้นจะต้องเป็นคนละตัวกับคนที่เป็นส่วนหลักของไดเรกเพลย์ แต่ถ้าหากว่าเป็นเซสชันแบบไคลแอนท์-เซิร์ฟเวอร์ เซิร์ฟเวอร์จะต้องเป็นทั้งส่วนหลักและส่วนโฮสต์ของเสียงในตัวเดียวกัน *IDirectPlayVoiceServer::StartSession* เซสชันหนึ่งถูกสร้างขึ้นมาและเริ่มต้นทำงานไคลแอนท์สามารถติดต่อเข้ามาได้

DVMSGID_CONNECTRESULT

เมื่อเซสชันถูกตั้งค่า ไคลแอนท์จะต้องเรียก *IDirectPlayVoiceClient::Connect* เพื่อทำการติดต่อไปยังเซสชันส่วนของเสียงถ้าเราเรียกเมธอดแบบอซิงโครนัสและได้รับค่าที่คืนมาเป็น DV_PENDING เราจะได้รับเมจเซจ *DVMSGID_CONNECTRESULT* เมสเซจนี้สามารถจะมาถึงก่อนหรือหลังจากเมธอด *IDirectPlayVoiceClient::Connect* คืนค่า *DV_PENDING* กลับมา ถ้าเราเรียกแบบซิงโครนัส เราจะได้รับเมสเซจ *DVMSGID_CONNECTRESULT*

DVMSGID_CREATEVOICEPLAYER

ไคลแอนท์ส่วนเสียงจะรับเมสเซจนี้ได้ก็ต่อเมื่อ ถ้าหากว่าเป็นแบบ peer-to-peer หลังจากที่ทำการติดต่อและได้รับผล *DVMSGID_CONNECTRESULT* เราจะได้รับเมสเซจ *DVMSGID_CREATEVOICEPLAYER* สำหรับตัวเราและสมาชิกทุกคนของเซสชันเสียง *dwFlags* ที่เป็นสมาชิกของโครงสร้างนี้ประกอบด้วย แฟลก 2 ค่าที่อธิบายผู้ใช้ถ้าผู้ใช้เป็น Local *DVPLAYERCAPS_LOCAL* flag จะถูกเซ็ต ถ้าเป็นแบบ half-duplex *DVPLAYERCAPS_HALFDUPLEX* flag จะถูกเซ็ต

Common Voice Messages

เมสเซจต่อไปนี้จะสามารถรับได้โดยไคลเอนท์ทั้งสองแบบ

DVMSGID_GAINFOCUS และ DVMSGID_LOSTFOCUS

เมสเซจจะทำให้เราสามารถที่จะจับเสียง เมื่อเราสามารถจับเสียงได้เราจะได้รับเมสเซจ *DVMSGID_GAINFOCUS* เมื่อเราไม่สามารถจับสัญญาณเสียงได้จะได้รับเมสเซจ *DVMSGID_LOSTFOCUS* ทั้งสองเมสเซจนี้เป็นตัวบ่งชี้พื้นฐานและไม่เกี่ยวข้องกับโครงสร้าง

การที่เราสามารถจับสัญญาณเสียงได้หรือไม่นั้น ไม่จำเป็นว่าจะต้องได้ตอนเริ่มต้นหรือจบการส่ง การระบุในขั้นตอนการเริ่มและส่งนั้นจะจัดกรโดยเมสเซจต่อไปนี้

DVMSGID_RECORDSTART และ DVMSGID_RECORDSTOP

เมสเซจนี้จะบอกว่ามีการสื่อสารกับผู้ใช้ที่เป็น Local เราจะได้รับเมสเซจ *DVMSGID_RECORDSTART* เมื่อการส่งเริ่มต้นขึ้นและเมสเซจ *DVMSGID_RECORDSTOP* เมื่อการส่งจบลง โครงสร้างของเมสเซจนี้จะมีข้อมูลเกี่ยวกับผู้ที่ใช้เสียงอยู่ เมสเซจ *DVMSGID_GAINFOCUS* จะตามมาด้วยเมสเซจ *DVMSGID_RECORDSTART* ถ้าเราพลาดในส่วนของการจับเสียงขณะที่ผู้ใช้ยังพูดอยู่การส่งจะหยุดลงและเราจะได้รับเมสเสจ *DVMSGID_LOSTFOCUS* แล้วตามด้วยเมสเซจ *DVMSGID_RECORDSTOP* ถ้าเราจับสัญญาณเสียงได้อีกการส่งก็จะเริ่มต้นอีกครั้ง

เราจะได้รับเมสเซจ *DVMSGID_RECORDSTOP* เมื่อการส่งหยุดลงแต่ถ้าเราหยุดการส่งโดยการตัดการเชื่อมต่อขณะที่มีการส่งอยู่จะไม่แน่ใจว่าจะได้รับเมสเซจ *DVMSGID_RECORDSTOP*

DVMSGID_INPUTLEVEL และ DVMSGID_OUTPUTLEVEL

เมสเซจ *DVMSGID_INPUTLEVEL* จะเป็นตัวกำหนดระดับของเสียงที่ผ่านมาทางไมโครโฟน ส่วนเมสเซจ *DVMSGID_OUTPUTLEVEL* จะกำหนดระดับเสียงที่จะออกมาจากทางลำโพง เมสเซจนี้จะส่งมาในช่วงเวลาหลังจากที่เราเริ่มเมสเซจ *DVMSGID_RECORDSTART* และจะหยุดเมื่อได้รับเมสเซจ *DVMSGID_RECORDSTOP* เราสามารถกำหนดช่วงเวลาที่จะส่งเมสเซจได้โดยกำหนดที่เมสเซจ *dwNotifyPeriod*

Session Termination Messages

DVMSGID_DISCONNECTRESULT

ทำการตัดการติดต่อจากเซสชันการพูดคุยโดนเรียก*IDirectPlayVoiceClient::Disconnect* ถ้าเราเรียกแบบอซิงโครนัสจะได้ค่าที่กลับมาเป็น *DV_PENDING* เราจะได้รับเมสเซจ

DVMSGID_DISCONNECTRESULT เพื่อรายงานผลที่ได้ โดยที่เมสเซจนี้อาจจะถึงก่อนหรือหลังเมสเซจ *IDirectPlayVoiceClient::Disconnect*

DVMSGID_SESSIONLOST

ถ้าหากเซสชันถูกหยุดลงโดยข้อผิดพลาดที่ไม่ได้เตรียมการไว้ก่อนไคลเอนท์จะได้รับเมสเซจ *DVMSGID_SESSIONLOST* และเหตุผลของความผิดพลาดนั้นจะเป็นสมาชิกของ *hrResult*

DVMSGID_DELETEVOICEPLAYER

สำหรับ peer-to-peer นั้นเราจะได้รับเมสเซจ *DVMSGID_DELETEVOICEPLAYER* สำหรับทุกๆผู้ใช้งานที่อยู่ในเซสชันที่เราตัดการติดต่อไป ไม่ว่าจะเป็นการที่เราตัดการเชื่อมต่อเองหรือ เกิดความผิดพลาดก็ตามทำให้เราแน่ใจว่าทุกครั้งที่มีการสร้างผู้ใช้ โดยเมสเซจ *DVMSGID_CREATEVOICEPLAYER* จะต้องจบลงด้วยเมสเซจ *DVMSGID_DELETE VOICEPLAYER* เสมอ เพราะเมสเซจนี้จะมาถึงก่อนเมสเซจ *DVMSGID_DELETE VOICEPLAYER*

- **Handling Voice Host Messages**

ทุกๆเซสชันของ Microsoft® DirectPlay® Voice จะต้องมีโฮสต์ที่รับผิดชอบการเริ่มต้นการจัดการและการจบเซสชัน โฮสต์ของเสียงนั้นจะต้องสร้างออบเจคต์ของเซิร์ฟเวอร์เสียง (CLSID_DirectPlayVoiceServer) และจัดการเซสชันของเสียงผ่านทางอินเตอร์เฟสของออบเจคต์นั้น เหมือนกับทางฝั่งไคลเอนท์จะต้องสร้างโฮสต์เสียงขึ้นมาเพื่อรอรับและจัดการกับเมสเซจเสียงที่เข้ามา โดยจะมีเมสเซจที่เกี่ยวข้องดังนี้ *DVMSGID_CREATEVOICEPLAYER* , *DVMSGID_DELETEVOICEPLAYER และ DVMSGID_SESSIONLOST* จะไม่เหมือนกับเมสเซจส่วนหลักของไดเรกเพลย์ เพราะจะขึ้นอยู่กับประเภทของเซสชัน เมสเซจเหล่านี้จะสามารถรับได้โดยส่วนจัดการกับเมสเซจเสียงทางฝั่งไคลเอนท์

1. Messaging During Normal Game Play
2. Session Termination Messages

Messaging During Normal Game Play

โฮสต์ของ client/server voice นั้นจะรับเมสเซจเหล่านี้เมื่อเซสชันของเสียงเริ่มต้นขึ้น

DVMSGID_CREATEVOICEPLAYER

ได้รับเมื่อมีผู้เล่นเข้ามาในแอพพลิเคชัน

DVMSGID_DELETEVOICEPLAYER

ได้รับเมื่อมีผู้เล่นออกไปจากแอพพลิเคชัน

Session Termination Messages

โฮสต์แบบไคลแอนท์-เซิร์ฟเวอร์จะได้รับเมสเซจดังต่อไปนี้เมื่อจะทำการจบเซสชันของเสียงลง

DVMSGID_SESSIONLOST

ถ้าหากเซสชันถูกหยุดลงโดยข้อผิดพลาดที่ไม่ได้เตรียมการไว้ก่อนไคลเอนท์จะได้รับเมสเซจ *DVMSGID_SESSIONLOST* และเหตุผลของความผิดพลาดนั้นจะเป็นสมาชิกของ *hrResult*

DVMSGID_DELETEVOICEPLAYER

สำหรับ peer-to-peer นั้นเราจะได้รับเมสเซจ *DVMSGID_DELETEVOICEPLAYER* สำหรับทุกๆ ผู้ใช้งานที่อยู่ในเซสชันที่เราตัดการติดต่อไป ไม่ว่าจะเป็นการที่เราตัดการเชื่อมต่อเอง หรือ เกิดความผิดพลาดก็ตามทำให้เราแน่ใจว่าทุกครั้งที่มีการสร้างผู้ใช้ โดยเมสเซจ *DVMSGID_CREATEVOICEPLAYER* จะต้องจบลงด้วยเมสเซจ*DVMSGID_DELETE VOICEPLAYER* เสมอเราเพราะเมสเซจนี้จะมาถึงก่อนเมสเซจ *DVMSGID_DELETE VOICEPLAYER*

2.6 การอิมพลิเมนท์ Callback Function ในไดเร็กเพลย์และไดเร็กพลย์วอยซ์

ไดเร็กเพลย์และไดเร็กเพลย์วอยซ์จำเป็นที่จะต้องทำการ อิมพลิเมนท์และลงทะเบียน callback function ก่อนที่เราจะสามารถจับเมสเซจ ของไดเร็กเพลย์ได้ ไดเร็กเพลย์ เป็นมัลติเธรด และจะส่งเมสเซจได้ครั้งละหลายเมสเซจพร้อมกัน ดังนั้นเพื่อความถูกต้องและสมบูรณ์ของการเข้าถึงข้อมูลนั้นเราจำเป็นที่จะต้องทำการอิมพลิเมนท์ในเมธอดของการทำซิงโครนัส ของมัลติเธรด ที่เรารู้จักกันในชื่อ *re-entrant* หรือ *threadsafe*

2.6.1 โครงสร้างของ Callback Function

โครงสร้างของ Callback Function นั้นจะเป็นไปตาม มาตรฐาน Microsoft Windows Win32 API

```
HRESULT WINAPI Callback(
 PVOID pvUserContext,
 DWORD dwMessageType,
 PVOID pMessage );
```

pvUserContext เป็นค่าที่เราใส่เมื่อเรา register callback function ด้วยไดเร็กเพลย์ ถ้าเราส่งค่านี้ไปให้ไดเร็กเพลย์ เมื่อเราลงทะเบียนแล้ว ค่าจะส่งกลับมาเมื่อไดเร็กเพลย์เรียกฟังก์ชันของเรา

dwMessageType เป็นหมายเลขหนึ่งที่เราส่งเข้าไปให้ไดเร็กเพลย์

pMessage จะบรรจุเมสเซจที่เราส่งไปให้ไดเร็กเพลย์

2.6.2 Registering Callback

เครือข่ายของ DirectPlay callback functions นั้นเป็นชนิด *PFNDPNMESSAGEHANDLER* ขึ้นอยู่กับชนิดของเซสชันของเครือข่าย เราจะ ลงทะเบียนแอดเดรสของ callback function ด้วย *IDirectPlay8Peer::Initialize, IDirectPlay8Client::Initialize* หรือ *IDirectPlay8Server::Initialize*

ถ้าเราลงทะเบียน DirectPlay voice callback function เราจะ ลงทะเบียนแอดเดรสของ callback ด้วย *IDirectPlayVoiceClient::Initialize* หรือ *IDirectPlayVoiceServer::Initialize* ขึ้นอยู่กับชนิดของเซสชันของ ไดเร็กเพลย์วอยซ์ที่เราต้องการสร้างขึ้นมา

<u>*ตัวอย่าง*</u>

```
HRESULT WINAPI Callback(PVOID, DWORD, PVOID);
IDirectPlay8Peer* pdp8Peer;
// Get the server interface
hr = CoCreateInstance( CLSID_DirectPlay8Peer, ...)
...
// Register the callback
hr = pdp8Peer->Initialize(NULL, Callback, 0);
```

การอิมพลิเมนท์ callback ของเครือข่ายไดเร็กเพลย์โดยใช้ Critical Section Objects

ไดเร็กเพลย์และไดเร็กเพลย์วอยซ์จำเป็นที่จะต้องทำการ อิมพลิเมนท์และลงทะเบียน callback function ก่อนที่เราจะสามารถจับเมสเซจ ของไดเร็กเพลย์ได้ ไดเร็กเพลย์เป็นมัลติเธรด และจะส่งเมสเซจได้ครั้งละหลายเมสเซจพร้อมกัน ดังนั้นเพื่อความถูกต้องและสมบูรณ์ของการเข้าถึงข้อมูลนั้นเราจำเป็นที่จะต้องทำการอิมพลิเมนท์ในเมธอดของการทำซิงโครนัสของมัลติเธรด ที่เรารู้จักกันในชื่อ *re-entrant* และ *threadsafe*

มีการทำซิงโครนัสของมัลติเธรด อยู่ 3 วิธีด้วยกัน คือ

1. Mutex Objects
2. Semaphore Objects
3. Critical Section Objects

ตัวอย่างของไดเร็กเพลย์วอยซ์ที่มาพร้อมกับ DirectX 8.1 SDK ใช้แบบ Critical Section Objects และเราจะมาอธิบายว่าเราจะนำวิธีการนี้มาใช้ได้อย่างไร ถ้าเราต้องการนำ Mutex หรือ Semaphore มาใช้

ตัวอย่าง

```
CRITICAL_SECTION g_csPlayerContext;
InitializeCriticalSection(&g_csPlayerContext);
Next, implement the DirectPlay message callback handler.
HRESULT WINAPI DirectPlayMessageHandler( PVOID pvUserContext,
                                         DWORD dwMessageId,
                                         PVOID pMsgBuffer )
{
  switch( dwMessageId )
  {
    case DPN_MSGID_CREATE_PLAYER:
    {
      EnterCriticalSection( &g_csPlayerContext );
      //callback is now locked
      //perform operation on player data
      LeaveCriticalSection( &g_csPlayerContext );
    }
  }
}
```

สุดท้ายนี้ ระหว่างที่เราต้องการออกจากแอปพลิเคชันนั้น เราต้องแน่ใจว่าเราเรียก DeleteCriticalSection ขึ้นมาเพื่อทำการจัดการคืนค่าหน่วยความจำแล้วเรียบร้อย

ตัวอย่าง

```
DeleteCriticalSection( &g_csPlayerContext );
```

บทที่ 3
การวิเคราะห์และออกแบบ

3.1 ความต้องการของระบบ (Requirement)

3.1.1 หลักการ

เป็นแอปพลิเคชันที่ใช้ในการติดต่อพูดคุยระหว่างยูสเซอร์โดยยูสเซอร์สามารถติดต่อผ่านเครือข่ายอินเตอร์เน็ตหรือเครือข่ายโทรศัพท์ก็ได้ โปรแกรมจะประกอบด้วย 2 ส่วน คือ

- *ส่วนของเซิร์ฟเวอร์* จะคอยรับการติดต่อจากไคลแอนท์ ว่าต้องการจะพูดคุยกับใครผ่านทางเซิร์ฟเวอร์โดยที่การคุยจะเป็นการคุยแบบเรลไทม์ (Real-time) โดยสามารถที่จะคุยกันแบบคนต่อคนหรือ จะคุยแบบการประชุมครั้งละหลายๆคนก็ได้ และสามารถที่จะให้บริการเป็น Voice Mail Server ที่สามารถบันทึกข้อความเสียงจากไคลเอ็นท์ที่เป็นคอมพิวเตอร์และโทรศัพท์ และให้โทรศัพท์เข้ามาฟังข้อความที่ฝากไว้โดยมีการกด User ID และ Password
- *ส่วนของไคลแอนท์* เป็นส่วนของผู้ใช้คอมพิวเตอร์ที่จะติดต่อกับคอมพิวเตอร์หรือโทรศัพท์พื้นฐาน ถ้าหากว่ายูสเซอร์ต้องการที่จะพูดคุยกันแบบคอมพิวเตอร์กับคอมพิวเตอร์จะต้องทำการติดต่อไปยังเซิร์ฟเวอร์แล้วเซิร์ฟเวอร์จะส่งรายชื่อและสถานะของบุคคลที่เชื่อมต่ออยู่กับเซิร์ฟเวอร์ในขณะนั้น มาให้เลือกว่าเราจะคุยกับ ไคลเอ็นท์จะเป็นส่วนที่ ติดต่อกับโทรศัพท์ ไม่ว่าจะเป็นการรับ หรือ ส่ง ข้อมูล

3.1.2 ขอบเขตของการแอปพลิเคชัน

1. เครื่องคอมพิวเตอร์ของยูสเซอร์จะต้องสนับสนุน DirectX 8.0
2. เซิร์ฟเวอร์ยังไม่มีการเก็บรายละเอียดของยูสเซอร์แต่ละคนมาไว้เป็นข้อมูลในการพิสูจน์ตนเก็บเพียงแต่ User ID และ Password ลงในไฟล์เท่านั้น
3. ไม่สามารถที่จะกำหนดห้ามบุคคลใดบุคคลหนึ่งเข้ามาคุยกับเราได้
4. เซิร์ฟเวอร์ไม่สามารถทำงานเป็นตัวกลางในการเชื่อมต่อระหว่างไคลเอ็นท์กับโทรศัพท์ได้ แบบเรลไทม์ แต่สามารถทำเป็น Voice Mail Server ได้

3.2 การวิเคราะห์และการออกแบบระบบ

การพัฒนาแอปพลิเคชันนี้เป็นแอปพลิเคชันที่พัฒนาบนระบบปฏิบัติการวินโดว์(Win32) อีกทั้งยังใช้ API บนระบบปฏิบัติการวินโดว์ซึ่งเป็นไปตามหลักการในการพัฒนาแบบ Object-Oriented จึงมีการวิเคราะห์ ออกแบบ และเขียนโปรแกรมตามหลักการของออบเจ็คต์ โดยใช้ UML (Unified Modeling Language) ซึ่งเป็นการวิเคราะห์และออกแบบระบบโดยการสร้างโมเดลขึ้นมาเพื่อจำลองออบเจ็คต์ต่างๆที่มีอยู่ในระบบ และแสดงความสัมพันธ์ระหว่างออบเจ็คต์เหล่านั้น

ในการวิเคราะห์และออกแบบแอปพลิเคชันโดยใช้ UML ที่ประกอบด้วย 3 ไดอะแกรมดังนี้

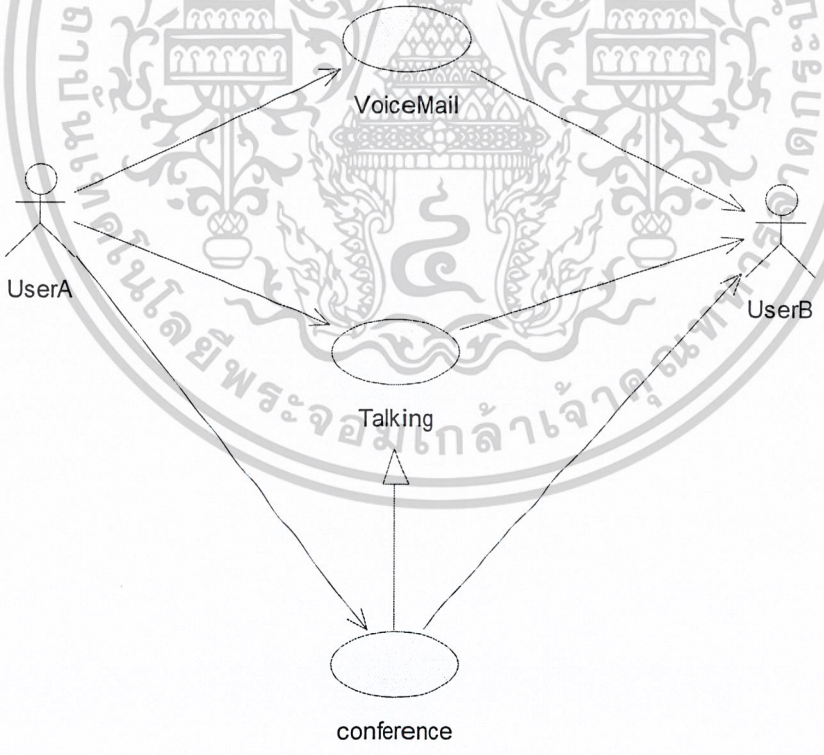
- Use Case Diagram
- Sequence Diagram
- State Diagram

ในการแสดงรายละเอียดการทำงานของแอปพลิเคชันในแต่ละไดอะแกรมนั้นจะแสดงเฉพาะส่วนที่สำคัญเท่านั้น ดังแสดงต่อไปนี้

3.2.1 Use Case Diagram

เป็นไดอะแกรมที่แสดงฟังก์ชันการทำงานที่สำคัญของแอปพลิเคชันที่ยูสเซอร์สามารถเข้ามาใช้บริการได้โดยจะมี ฟังก์ชันที่สำคัญอยู่ 3 ฟังก์ชันคือ

- **VoiceMail** เป็นฟังก์ชันที่ให้บริการยูสเซอร์ฝากข้อความและให้ผู้ใช้โทรศัพท์โทรเข้ามาฟังได้
- **Talking** เป็นฟังก์ชันที่ให้บริการในการติดต่อพูดคุยกับยูสเซอร์คนอื่นที่ใช้บริการบนเครือข่ายอินเตอร์เน็ตเหมือนกัน โดยจะพูดคุยกันแบบ 1:1
- **Conference** เป็นฟังก์ชันที่ให้บริการในการติดต่อพูดคุยกับยูสเซอร์คนอื่นที่ใช้บริการบนเครือข่ายอินเตอร์เน็ตเหมือนกัน โดยจะพูดคุยกับทุกๆคนหรือคุยกันแบบ conference



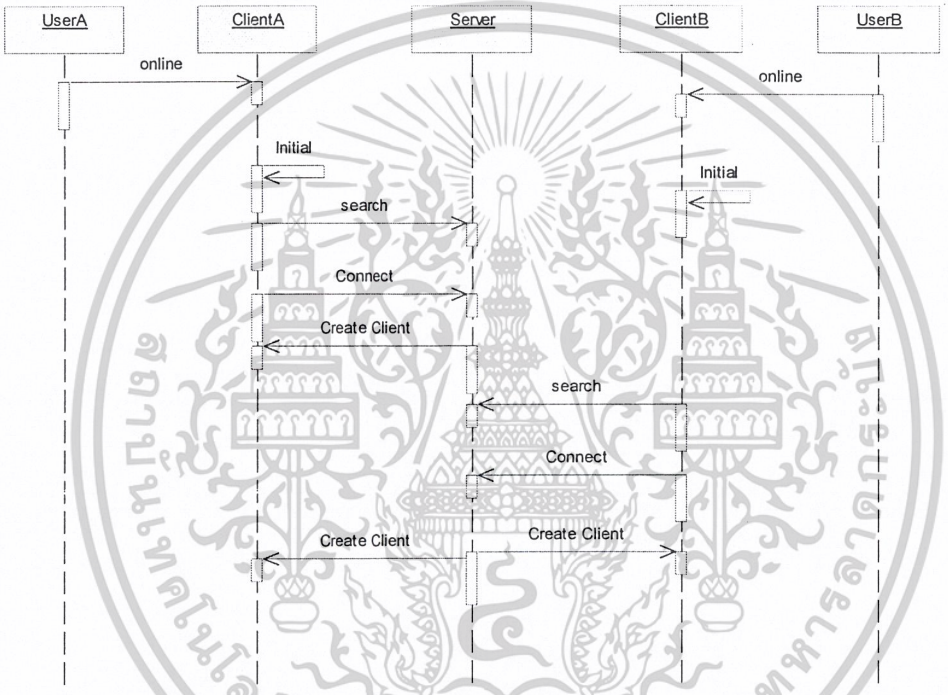
รูปที่ 3-1 ยูสเคสไดอะแกรมของระบบ

3.2.2 Sequence Diagram

เป็นไดอะแกรมที่แสดงลำดับการทำงานภายใน Use Case Diagram อย่างมีลำดับขั้นตอน และการทำงานร่วมกันของอ็อบเจ็กต์ต่างๆ

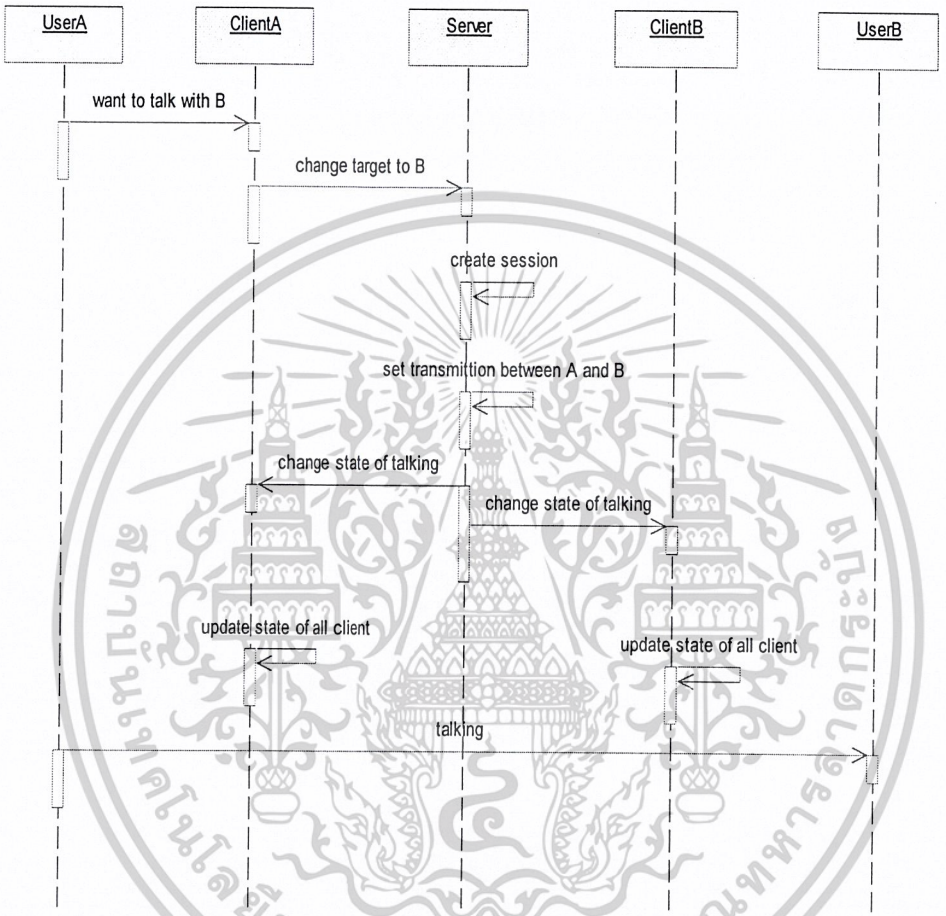
โดยจะแสดงเฉพาะซีเควนไดอะแกรม ที่สำคัญดังต่อไปนี้

1. **Register Client :** เป็นซีเควนไดอะแกรมที่แสดงขั้นตอนการทำงานของการที่ไคลแอนท์ร้องขอการติดต่อกับเซิร์ฟเวอร์ เพื่อให้เซิร์ฟเวอร์สร้างไคลแอนท์ขึ้นมาและทำการส่งข้อมูลเหล่านี้ต่อไปยังไคลแอนท์ทุกๆ ไคลแอนท์



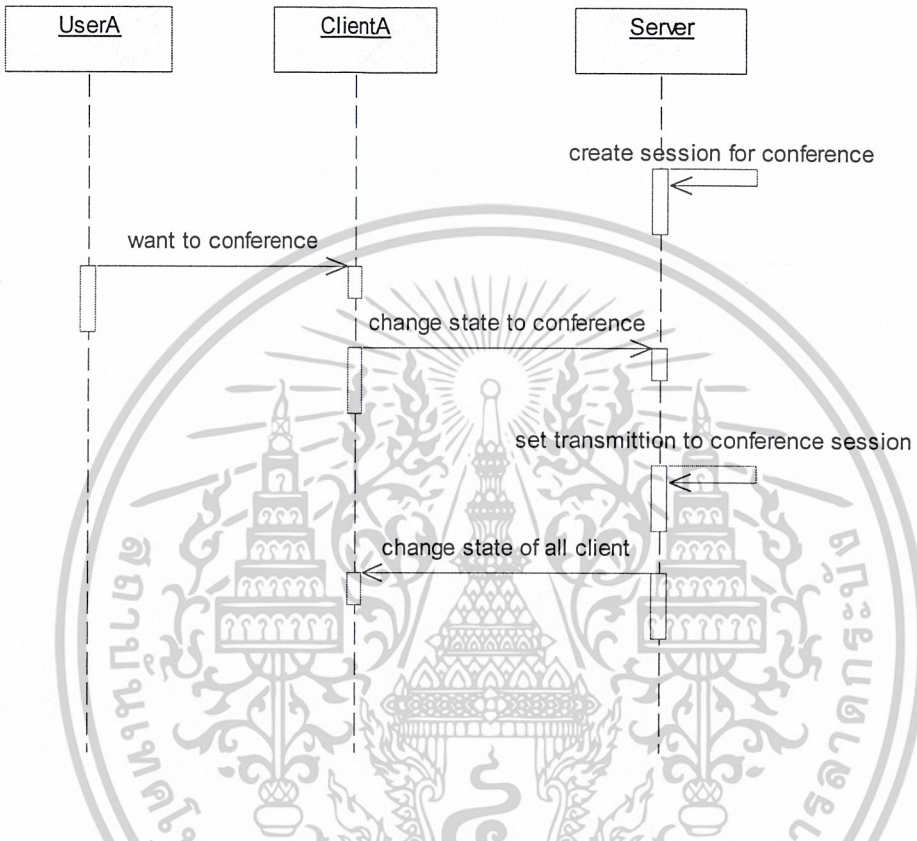
รูปที่ 3-2 ซีเควนไดอะแกรมของไคลแอนท์ที่ร้องขอการติดต่อจากเซิร์ฟเวอร์

2. Peer-to-peer : เป็นซีเควนไดอะแกรมที่แสดงขั้นตอนของการติดต่อเซิร์ฟเวอร์เพื่อร้องขอบริการการพูดคุยกันแบบ peer-to-peer ซึ่งไคลแอนท์ต้องส่งแมสเซจไปให้กับเซิร์ฟเวอร์เพื่อให้เซิร์ฟเวอร์สร้างห้องพูดคุยกันระหว่างไคลแอนท์รวมทั้ง set transmission ให้คู่ของไคลแอนท์คุยกันได้ก่อนจะส่งแมสเซจให้กับไคลแอนท์ทุกคนเพื่ออัปเดทสถานะของตัวเอง



รูปที่ 3-3 ซีเควนไดอะแกรมของการใช้บริการพูดคุยกันแบบ peer-to-peer

3. **Conference :** เป็นซีเควนไดอะแกรมที่แสดงขั้นตอนของการติดต่อเซิร์ฟเวอร์เพื่อร้องขอบริการการพูดคุยกันแบบ conference ซึ่งไคลแอนท์ต้องส่งแมสเซจไปให้กับเซิร์ฟเวอร์เพื่อให้เซิร์ฟเวอร์ set transmission ให้กับไคลแอนท์คุยกันได้ก่อนจะส่งแมสเซจให้กับไคลแอนท์ทุกคนเพื่ออัปเดทสถานะของตัวเอง

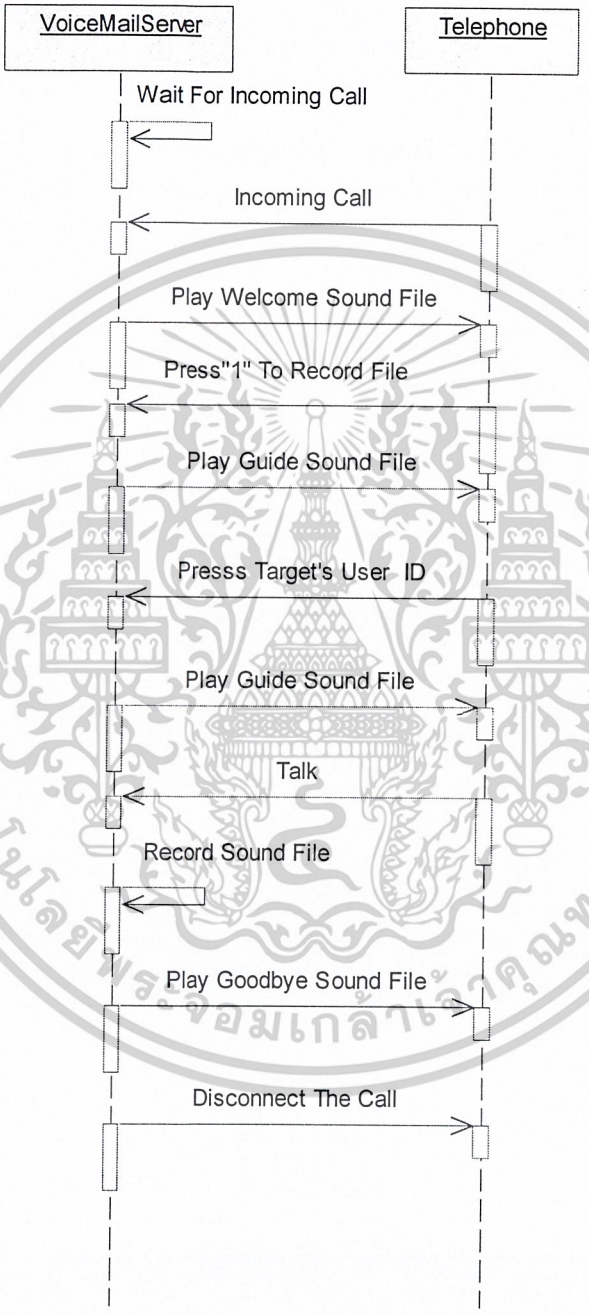


รูปที่ 3-4 ซีเควนไดอะแกรมของการใช้บริการพูดคุยกันแบบ Conference

4. Save Voicemail (Client): เป็นซีเควนไดอะแกรมในส่วนของการที่ไคลเอ็นท์ที่จะทำการฝากข้อความเสียงไปไปยังเซิร์ฟเวอร์ โดยเมื่อมีการร้องขอฝากข้อความจากยูสเซอร์ ไคลแอนท์จะส่งเมสเสสให้เซิร์ฟเวอร์เพื่อสร้างไฟล์เสียงรอไว้ หลังจากนั้นไคลแอนท์จะส่งข้อมูลที่อยู่ในบัฟเฟอร์ที่บันทึกไว้ไปยังเซิร์ฟเวอร์ให้เซิร์ฟเวอร์บันทึกลงไฟล์ที่เตรียมไว้

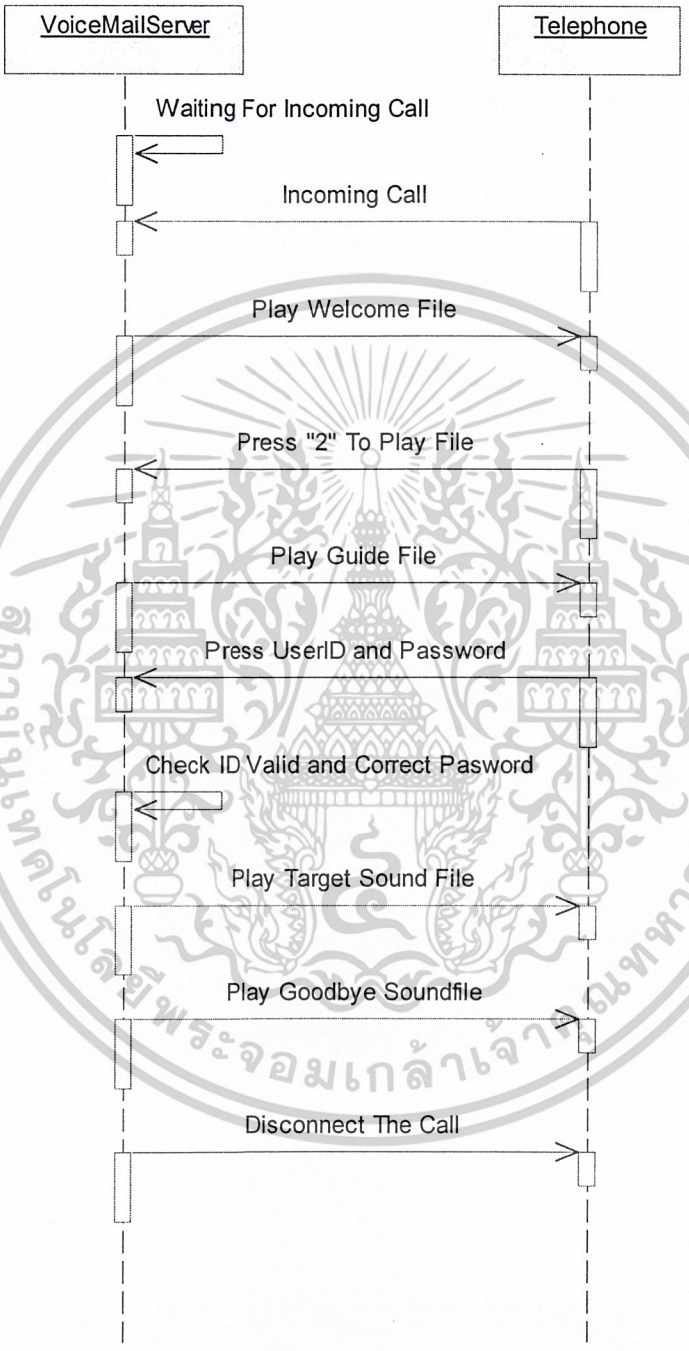
รูปที่ 3-5ซีเควนไดอะแกรมของการฝากข้อความในไคลแอนท์

5. Phone Record: เมื่อมีสายเข้ามาทางเซิร์ฟเวอร์ เซิร์ฟเวอร์จะทำการสอบถามและรับความต้องการจากโทรศัพท์โดยการตรวจสอบสถานะของการกดปุ่ม จากนั้นจะทำการบันทึกข้อความเสียงจากทางโทรศัพท์และทำการจัดเก็บไว้ในโฟลเดอร์ที่เป็นหมายเลขของผู้รับ



รูปที่ 3-6 ซีเควนไดอะแกรมของการฝากข้อความผ่านโทรศัพท์

6. Phone Play: เมื่อเซิร์ฟเวอร์ตรวจสอบว่ามีสายเข้ามาและมีความต้องการจะฟังข้อความที่ฝากไว้จะทำการตรวจสอบหมายเลขยูสเซอร์และ รหัสผ่าน ถ้าหากตรงก็จะทำการเล่นข้อความเสียงนั้น แต่ถ้าไม่จะทำการตัดการเชื่อมต่อทันที



รูปที่ 3-7 ซีเควนไดอะแกรมของการฟังข้อความผ่านโทรศัพท์

3.2.3 State Diagram

เป็นไดอะแกรมที่อธิบายพฤติกรรมของซอฟต์แวร์และการเปลี่ยนแปลงสถานะเมื่อมีเหตุการณ์ต่างๆ เกิดขึ้น ซึ่งในแอปพลิเคชันนี้มีซอฟต์แวร์อยู่ 2 ส่วนที่สามารถแสดงเสตสไดอะแกรมดังต่อไปนี้ คือ

- **เซิร์ฟเวอร์**

ส่วนของเซิร์ฟเวอร์จะมี 15 state ดังนี้

1. **Initial Server state :** เป็นส่วนที่จะทำการตั้งค่าเริ่มต้นต่างๆ ให้กับเซิร์ฟเวอร์ทั้งการสร้างออบเจคต์ของไดเร็กเพลย์และไดเร็กเพลย์วอยซ์ รวมทั้งการกำหนดว่าจะให้เซิร์ฟเวอร์เป็นแบบ Mixing Server หรือแบบ Forwarding Server หรือการกำหนด Codec ของเสียง เป็นต้น

2. **Server Listen state :** เมื่อเซิร์ฟเวอร์เริ่มทำงาน (start server) จะมีการรอรับแมสเซจของไคลแอนท์ โดยจะมี Call Back function ให้การจัดการ(Handle message) กับแมสเซจเหล่านี้

3. **Create Client state :** เมื่อมีไคลแอนท์ติดต่อเข้ามายังเซิร์ฟเวอร์ ก็จะเกิดแมสเซจในการสร้างไคลแอนท์ขึ้น ทำให้เซิร์ฟเวอร์เข้าสู่สถานะนี้เพื่อทำการเก็บข้อมูลของไคลแอนท์และส่งแมสเซจที่มีข้อมูลของไคลแอนท์ใหม่ไปยังทุกคนที่ online อยู่แล้วจึงกลับสู่สถานะ Listen Server อีกครั้ง

4. **Create Group state :** ในขณะที่เซิร์ฟเวอร์อยู่ในสถานะ Listen Server หากให้รับแมสเซจของไคลแอนท์ว่าต้องการพูดคุยกับใครคนหนึ่งที่ online อยู่เช่นกัน (peer-to-peer) เซิร์ฟเวอร์ก็จะเข้าสู่สถานะ Create Group เพื่อสร้างห้องคุยกันระหว่าง 2 คน

5. **Add Client to Group state :** เป็นสถานะที่เพิ่มไคลแอนท์เข้าไปในห้องสนทนา โดยไคลแอนท์ที่จะเข้าสู่สถานะนี้มี 2 อย่างคือ

   - ต้องการ conference ซึ่งการพูดคุยกันแบบ conference เซิร์ฟเวอร์จะมีการการสร้างห้องคุยกันตั้งแต่แรกเมื่อไคลแอนท์ต้องการ conference ก็จะเข้าสู่สถานะนี้
   - ต้องการพูดคุย แบบ peer-to-peer เป็นสถานะที่ต่อจากสถานะ Create Group เพื่อเพิ่มไคลแอนท์ทั้งสองสู่ห้องสนทนาที่สร้างขึ้นเพื่อคุยกัน 2 คน

6. **Set Transmission state :** เป็นสถานะที่เซิร์ฟเวอร์จะทำการกำหนดทิศทางในการให้ไคลแอนท์ซึ่งจะมี 3 กรณี คือ

   - กำหนดทิศทางให้ไคลแอนท์ทั้ง 2 ที่อยู่ในห้องคุยเดียวกันพูดคุยกันได้โดยการ set transmission ไปยังห้องสนทนาที่สร้างขึ้นจากสถานะ Create Group
   - กำหนดทิศทางให้ไคลแอนท์พูดคุยกันแบบ conference แบบ conference โดยการ set transmission ไปยังห้องสนทนาแบบ conference ซึ่งได้สร้างขึ้นตั้งแต่เริ่มต้นแล้ว

- กำหนดทิศทางให้ไคลแอนท์หยุดการพูดคุยทั้งแบบ conference หรือแบบ peer-to-peer แล้วจึงส่งแมสเซจให้กับไคลแอนท์ทั้งหมดเพื่อเปลี่ยนแปลงสถานะของไคลแอนท์ทั้งหมดที่เก็บไว้ก่อนจะเข้าสู่สถานะ Listen Server อีกครั้ง

7. Remove Client from Group state : เป็นสถานะที่ทำการลบไคลแอนท์ออกจากห้องสนทนาโดยไคลแอนท์จะเข้าสู่สถานะนี้เมื่อต้องการออกจากการ conference เพราะเซิร์ฟเวอร์จะลบไคลแอนท์ออกจากห้องสนทนานี้โดยไม่ลบห้องทิ้งเพราะห้องสนทนาแบบ conference จะสร้างไว้อย่างถาวร

8. Delete Group state : เป็นสถานะที่เซิร์ฟเวอร์จะทำการลบห้องสนทนาของคู่ไคลแอนท์ใดๆ ซึ่งจะทำหลังจากที่เซิร์ฟเวอร์หยุดการพูดคุยระหว่างไคลแอนท์จากสถานะ set transmission แล้ว

9. Create VoiceMail state: เป็นสถานะที่เซิร์ฟเวอร์จะทำการสร้างไฟล์เพื่อเตรียมสำหรับการบันทึกข้อความเสียงของไคลแอนท์ที่ต้องการบริการรับฝากข้อความเสียง

10. Save Voice state : เป็นสถานะที่เมื่อยูสเซอร์เริ่มการบันทึกข้อความเสียงไคลแอนท์ก็จะส่งเมสเสสพร้องทั้งข้อมูลที่เป็นเสียงมายังเซิร์ฟเวอร์เพื่อทำการบันทึกลงไฟล์ที่ได้สร้างไว้แล้ว

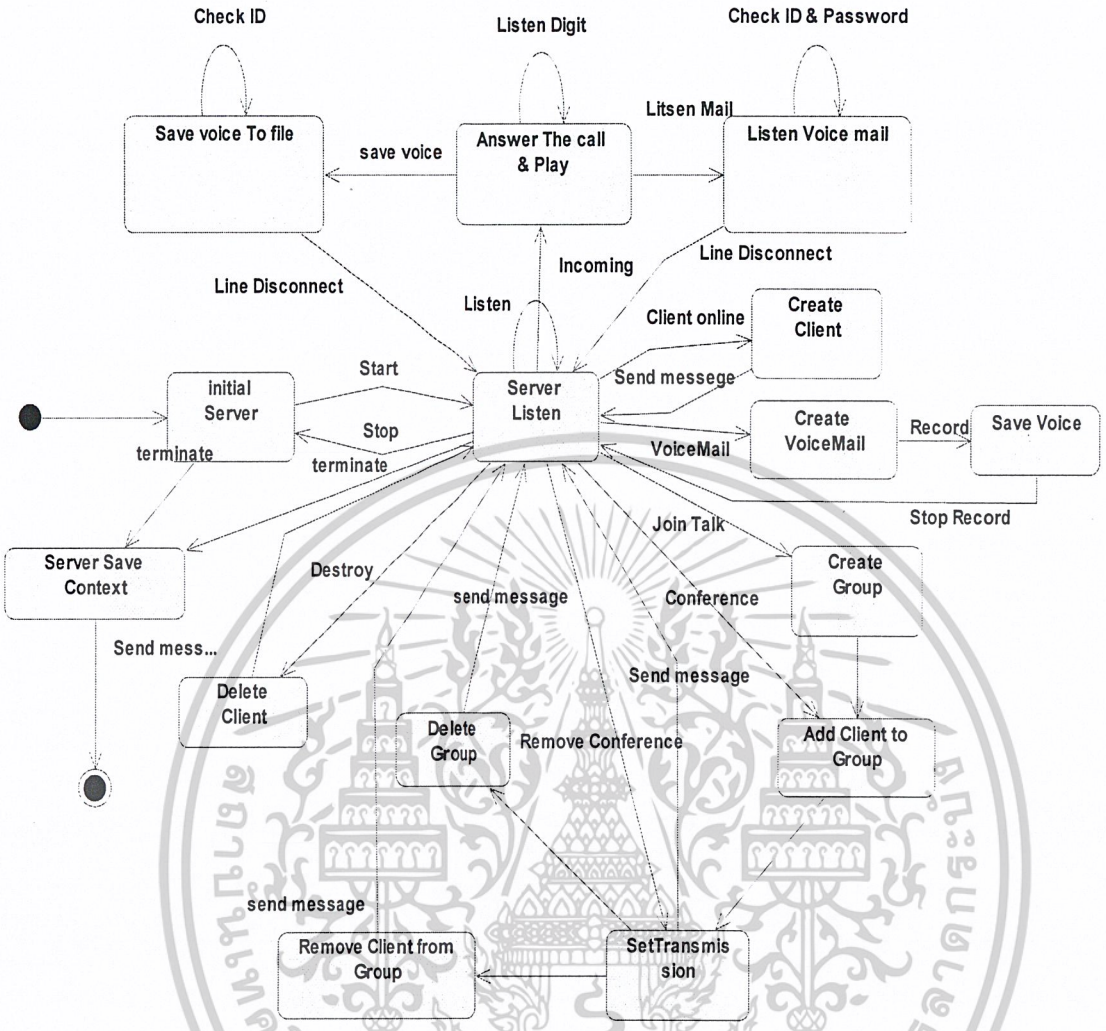
11. Save Context of Server state : เป็นสถานะสุดท้ายก่อนที่จะปิดเซิร์ฟเวอร์ โดยเซิร์ฟเวอร์ที่อยู่ในสถานะนี้จะทำการเคลียร์หน่วยความจำทั้งหมดที่เซิร์ฟเวอร์ครอบครองไว้

12. Answer The call & Play state : ทำการตรวจสอบความต้องการของยูสเซอร์ว่าต้องการจะฝากข้อความหรือว่า ฟังข้อความของตนเอง

13. Listen Voice mail state: จะทำการตรวจสอบยูสเซอร์และรหัส ว่าถูกต้องหรือไม่ ถ้าหากว่าถูกต้องจะทำการเล่นข้อความเสียงที่ฝากไว้ให้ฟัง

14. Delete Client state : เมื่อเซิร์ฟเวอร์ได้รับเมสเซจว่าไคลแอนท์ออกจากระบบไปก็จะเข้าสู่สถานะนี้เพื่อลบไคลแอนท์ออกจากระบบและส่งเมสเซจให้กับไคลแอนท์อื่นๆในระบบเพื่ออัปเดทสถานะ

15. Save Voice to File state : ทำการตรวจสอบว่าเป้าหมายที่ต้องการจะฝากข้อความนั้นมีอยู่หรือไม่ ถ้าหากว่ามีก็จะทำการบันทึกข้อความเสียงแต่ถ้าหากว่าไม่มีจะจบการเชื่อมต่อโดยทันที



รูปที่ 3-8 เสต็สไดอะแกรมของเซิร์ฟเวอร์

- ไคลแอนท์

ส่วนของไคลแอนท์ะมี 7 state ดังนี้

1. **Initial Client state :** เป็นสถานะที่จะทำการตั้งค่าเริ่มต้นต่างๆ ให้กับไคลแอนท์ทั้งการสร้างออบเจคต์ของไดเร็กเพลย์และไดเร็กเพลย์วอยซ์

2. **Client Listen state :** เป็นสถานะที่ไคลแอนท์พยายามตรวจหาเซิร์ฟเวอร์และร้องขอการติดต่อจากเซิร์ฟเวอร์

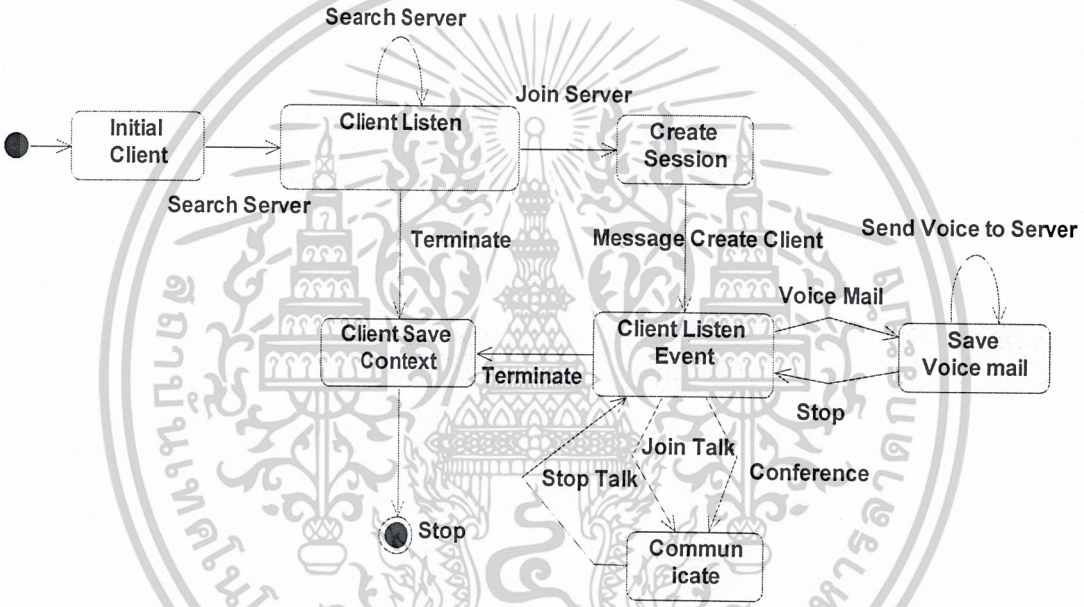
3. **Create Session state :** เมื่อเซิร์ฟเวอร์ตอบรับการร้องขอจากไคลแอนท์แล้ว เซิร์ฟเวอร์จะส่งแมสเซจมาให้ไคลแอนท์ ไคลแอนท์จึงเข้าสู่สถานะนี้โดยไคลแอนท์จะทำการเก็บข้อมูลของไคลแอนท์คนอื่นๆ ทุกคนที่ online อยู่ซึ่งเซิร์ฟเวอร์จะเป็นผู้ส่งข้อมูลทั้งหมดมาให้

4. **Client Listen Event state :** เป็นสถานะที่ไคลแอนท์รอยูสเซอร์ว่าต้องการบริการแบบใดทั้งแบบ conference หรือ แบบ peer-to-peer หรือแม้กระทั่งการฝากข้อความเสียง

5. **Communicate state :** หากยูสเซอร์มีการคลิ๊กปุ่มเพื่อขอใช้บริการ ทั้ง conference หรือ peer-to-peer ก็จะเข้าสู่สถานะนี้เพื่อส่งความต้องการไปยังเซิร์ฟเวอร์เพื่อให้เซิร์ฟเวอร์สร้างการเชื่อมต่อให้ และหากยูสเซอร์ต้องการหยุดการพูดคุยก็จะกลับสู่สถานะ Client Listen Event ตามเดิม

6. **Save Voice mail state:** เป็นสถานะที่มีการรับเสียงจากทางไคลเอ็นท์จากนั้นทำการบันทึกลงเป็นไฟล์นามสกุล wav และมีคุณสมบัติตามที่ต้องการ

7. **Client Save Context state :** เป็นสถานะสุดท้ายก่อนที่จะออกจากแอปพลิเคชัน โดยไคลแอนท์อยู่ในสถานะนี้จะทำการเคลียร์หน่วยความจำทั้งหมดที่เซิร์ฟเวอร์ครอบครองไว้



รูปที่ 3-9 เสต็ตไดอะแกรมของไคลแอนท์

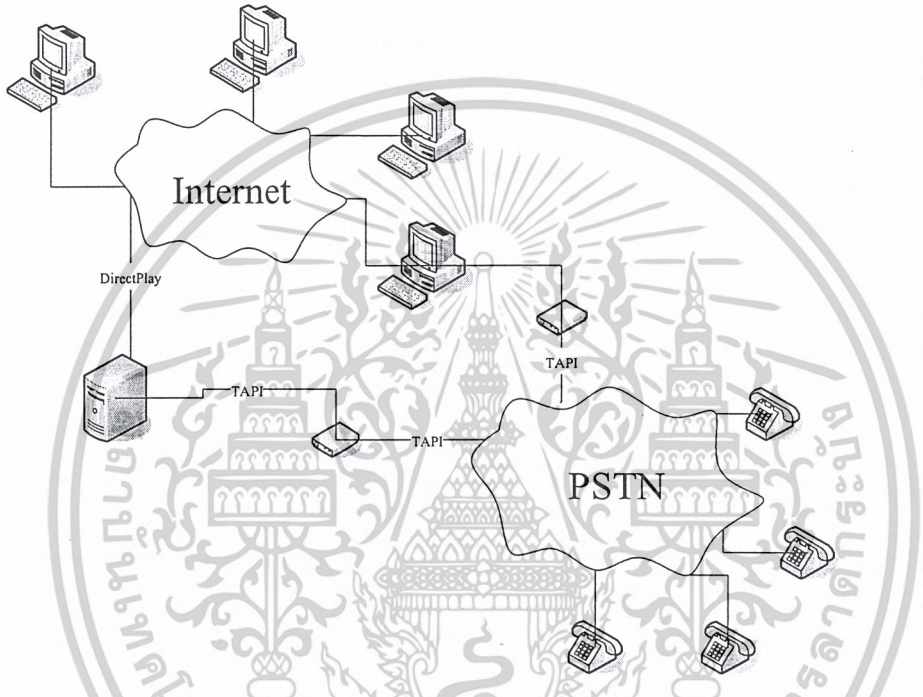
3.3 โครงสร้างของซอฟแวร์

ซอฟแวร์ที่พัฒนานั้นจะแบ่งออกเป็น 2 ส่วนหลักดังที่ได้กล่าวในส่วนของการออกแบบซึ่งจะแสดงความสัมพันธ์คร่าวๆดังแสดงในรูปที่ *3-10* ซึ่งจากไดอะแกรมจะมีส่วนสำคัญของโปรแกรมอยู่ 2 ส่วน คือ

1. *ส่วนของเซิร์ฟเวอร์* ซึ่งจะเป็นส่วนที่รับ Request ของ ไคลแอนท์ เพื่อติดต่อกับคอมพิวเตอร์ โดยในส่วนการติดต่อกับไคลแอนท์ จะใช้ ไดเร็กเพลย์ ในการสร้างการติดต่อ และส่วนของการจัดการกับสตรีมเสียงนั้นจะใช้ไดเร็กเพลย์วอยซ์ ในการจัดการกับเสียง โดยการที่เราจะเรียกใช้การทำงานของไดเร็ก

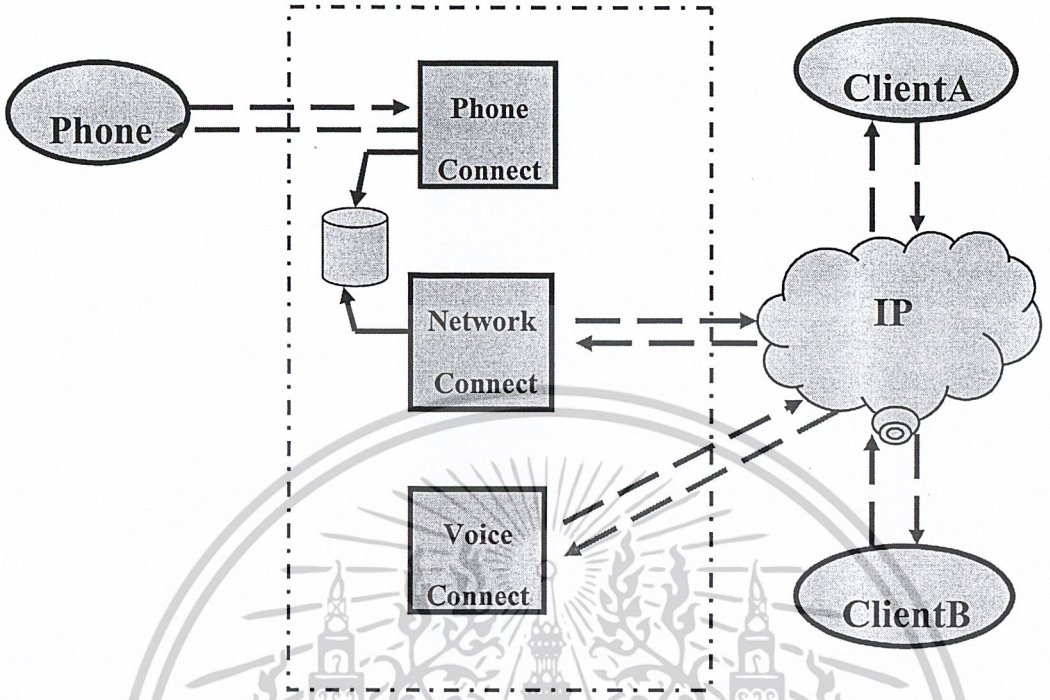
เพลย์วอยซ์ ได้นั้นเราจะต้องสร้างการเชื่อมต่อด้วยไดเร็กเพลย์เสียก่อน และจะทำการดูแลเรื่องของ Voicemail Server ที่ให้บริการเกี่ยวกับโทรศัพท์

2. *ส่วนของไคลแอนท์* ที่ทำงานบนคอมพิวเตอร์นั้นจะเป็นการสร้างการเชื่อมต่อกับเซิร์ฟเวอร์ผ่านเครือข่ายด้วยไดเร็กเพลย์ หลังจากนั้น ถ้าเราต้องการพูดกับบุคคลใดเราสามารถส่งการร้องขอไปยังเซิร์ฟเวอร์ และเซิร์ฟเวอร์จะสร้างช่องทางติดต่อได้แล้วจึงทำการสื่อสารกันระหว่างคอมพิวเตอร์กับคอมพิวเตอร์โดยใช้ไดเร็กเพลย์วอยซ์ในการจัดการกับสตรีมเสียง และสามารถฝากข้อความเสียงไปยังเซิร์ฟเวอร์ได้ด้วย



รูปที่ 3- 10 ไดอะแกรมการทำงานโดยรวมของระบบ

3.3.1 โครงสร้างซอฟร์แวร์ในส่วนของเซิร์ฟเวอร์



รูปที่ 3-11 ไดอะแกรมการทำงานส่วนของเซิร์ฟเวอร์

การพัฒนาแอปพลิเคชันในส่วนของเซิร์ฟเวอร์นั้น ประกอบด้วย 3 ส่วนหลักด้วยกัน คือ

1. Network Connect

เป็นส่วนที่ทำหน้าที่สำคัญในการควบคุมการสื่อสารระหว่างเครือข่ายอินเตอร์เน็ตไม่ว่าจะเป็นการสร้างการเชื่อมต่อกับไคลแอนท์ การควบคุมการสร้างเซสชันของการสนทนา หรือการหยุดการเชื่อมต่อ ซึ่งส่วนนี้ได้ออกแบบให้มีคุณสมบัติดังนี้

- ใช้ไดเร็กเพลย์ในการพัฒนา
- ใช้การติดต่อในชั้น Transport Layer คือ TCP และ UDP
- สามารถจัดการดูแลการเชื่อมต่อเกี่ยวกับเครือข่ายทุกๆแอปพลิเคชันได้
- รองรับไคลแอนท์ได้หลายคน
- สามารถสร้างห้องสนทนาส่วนตัวของแต่ละคู่สนทนา(peer-to-peer)
- สามารถสร้างห้องสนทนาแบบประชุม(conference)ได้
- สามารถจัดการข้อมูลแบบเรลไทม์ได้เพราะการติดต่อเป็นแบบใช้เสียง ดีเลย์ เป็นส่วนสำคัญมาก
- มีการทำงานหลายอย่างในเวลาเดียวกันจึงออกแบบเป็น มัลติเทรด
- สามารถตรวจสอบสถานะของแต่ละไคลเอ็นท์และส่งไปให้ทุกๆไคลเอ็นท์ได้

2. Voice Connect

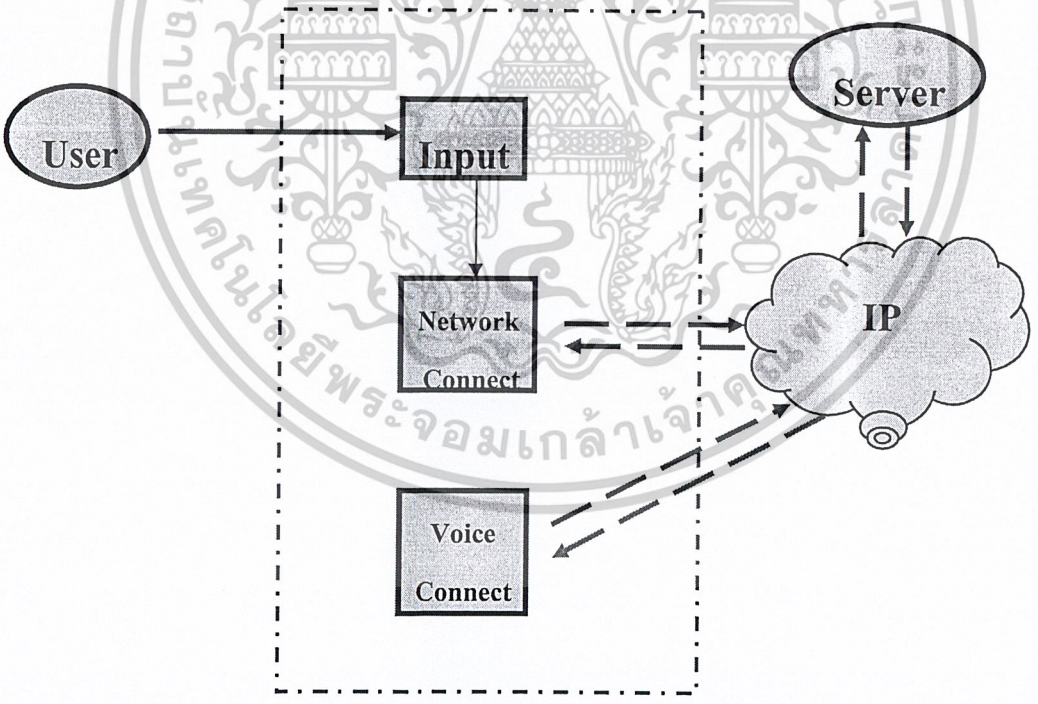
- เป็นส่วนที่ทำหน้าที่ในการดูแล ควบคุมเกี่ยวกับสตรีมเสียงทั้งหมดในการรับ-ส่งผ่านเครือข่าย มีคุณสมบัติดังต่อไปนี้
- ใช้ไดเร็กเพลย์ในการพัฒนา
- สามารถจัดการเสียงให้คุยกันเป็นคู่ๆ(peer-to-peer)ได้
- สามารถจัดการเสียงให้เป็นการประชุม(conference)ได้
- สามารถตรวจสอบสถานะการพูด ขณะนั้นๆว่ากำลังพูด หรือว่าเงียบไป
- สามารถรองรับการทำงานหลายๆ สตรีมพร้อมกันได้

3. Phone Connect

ทำหน้าที่ในการติดต่อและให้บริการกับโทรศัพท์ โดยจะมีการทำงานที่สำคัญดังนี้ คือ

- สามารถรับสายโทรศัพท์ได้
- สามารถตรวจสัญญาณการกดปุ่มบนโทรศัพท์ได้
- สามารถบันทึกเสียงจากโทรศัพท์ได้
- สามารถเล่นเสียงจากไฟล์ไปยังโทรศัพท์ได้

3.3.2 โครงสร้างซอฟร์แวร์ในส่วนของไคลแอนท์



รูปที่ 3-12 ไดอะแกรมการทำงานส่วนของไคลแอนท์

การพัฒนาแอปพลิเคชันในส่วนของไคลเอ็นท์นั้น ประกอบด้วย 2 ส่วนหลักด้วยกัน คือ

1. Network Connect:

เป็นส่วนที่สร้างคำสั่งในการทำการติดต่อกับปลายทางผ่านเซิร์ฟเวอร์ในเครือข่าย IP ได้ออกแบบให้มีคุณสมบัติดังนี้

- ใช้ ไดเร็กเพลย์ ในการพัฒนา
- ในส่วนของการสร้างการติดต่อนั้น ยูสเซอร์จะระบุมาที่เซิร์ฟเวอร์ว่าต้องการติดต่อกับปลายทางแบบใด ระหว่างติดต่อกันแบบเป็นคู่ หรือว่าแบบประชุมหลายคนพร้อมกัน
- ใช้การติดต่อในชั้น Transport Layer คือ TCP และ UDP
- หลังจากทำการเชื่อมต่อกับเซิร์ฟเวอร์แล้ว ยูสเซอร์แต่ละคนจะมองเห็นสถานะของกันและกัน ดังต่อไปนี้

  Online: ยูสเซอร์กำลังใช้แอปพลิเคชันอยู่ แต่ไม่ได้พูดคุยกับใครในขณะนี้

  Busy: ยูสเซอร์กำลังพูดคุยกับอีกฝ่ายอยู่

  Conference: ยูสเซอร์กำลังคุยกันแบบประชุมหลายคนอยู่

- สามารถเข้าไปร่วมคุยกับคนที่กำลังประชุมกันอยู่ได้โดยไม่ต้องทำการสร้างห้องคุยใหม่

2. Voice Connect:

- ดูแลในส่วนของการจัดการกับสตรีมเสียงเพื่อเตรียมส่งไปให้ยังเซิร์ฟเวอร์
- ในระหว่างการพูดคุยนั้นหน้าต่างที่ทำการพูดคุยจะแสดงสถานะของยูสเซอร์ทั้งสองฝ่าย ดังนี้

  Talking: กำลังพูดอยู่

  Silent : ไม่ได้พูดอยู่

- ดูแลการบันทึกข้อความเสียงของไคลแอนท์ไว้ที่ไว้ที่เซิร์ฟเวอร์

บทที่ 4

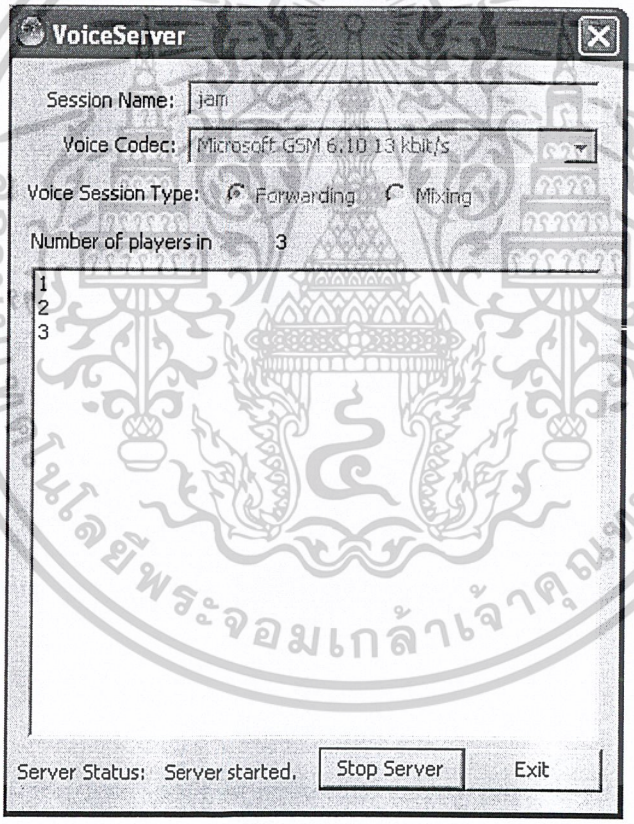
การทดสอบและผลการทดลอง

4.1 ตัวอย่างการใช้งานแอปพลิเคชัน

เมื่อทำการพัฒนาแอปพลิเคชันแล้วจะได้โปรแกรม 2 ส่วนด้วยกัน คือ ส่วนของเซิร์ฟเวอร์ และอีกส่วนหนึ่งจะเป็นส่วนของไคลแอนท์ที่ให้ยูสเซอร์ใช้งานโดยจะแยกการแสดงรายละเอียดของแต่ละส่วนดังนี้

4.1.1 ตัวอย่างการใช้งานเซิร์ฟเวอร์

จะเป็นส่วนที่เปิดบริการตลอดเวลาเพื่อให้บริการแก่ยูสเซอร์ที่จะเข้ามาใช้งานโดยจะมีการเซตค่าเริ่มต้นต่างๆ สำหรับเซิร์ฟเวอร์ไม่ว่าจะเป็น ระบบการทำงานของเซิร์ฟเวอร์ หรือการเซต Codec แบบต่างๆเพื่อการใช้งานที่เหมาะสม

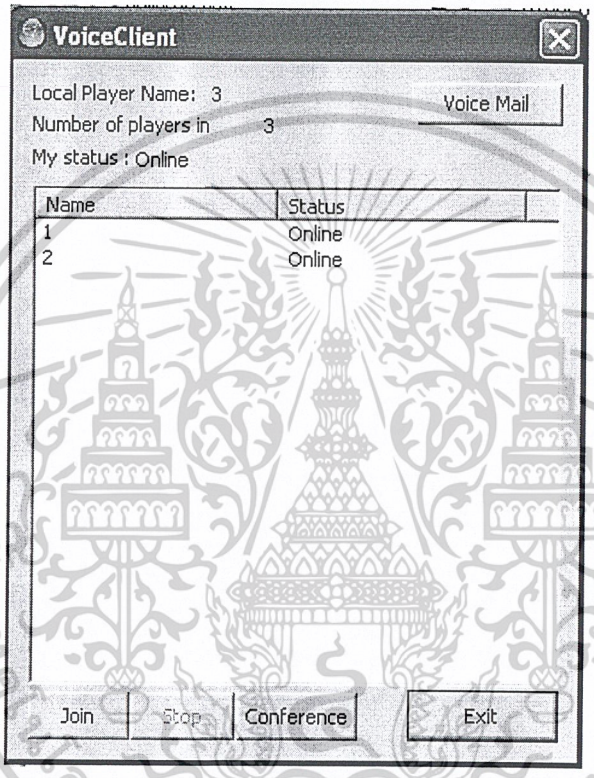


รูปที่ 4-1 หน้าต่างแสดงการทำงานของเซิร์ฟเวอร์

จากรูปแสดงการเริ่มทำงานของเซิร์ฟเวอร์หลังจากสตาร์ทเซิร์ฟเวอร์ซึ่งได้มีการกำหนดค่าเริ่มต้นของเซิร์ฟเวอร์ไม่ว่าจะเป็นประเภทของเซิร์ฟเวอร์หรือประเภทของการ Codec โดยหากมีไคลแอนท์ติดต่อเข้ามายังเซิร์ฟเวอร์ก็จะแสดงว่ามียูสเซอร์คนใดบ้างดังรูป

4.1.2 ตัวอย่างการใช้งานไคลแอนท์

จะเป็นส่วนที่ให้บริการแก่ยูสเซอร์ซึ่งเมื่อยูสเซอร์ทำการใช้งานแอปพลิเคชัน จะเริ่มด้วยการหาเซิร์ฟเวอร์ และติดต่อเข้ามายังเซิร์ฟเวอร์ โดยสถานะตอนแรกสุดจะเป็นสถานะ Online ดังรูป

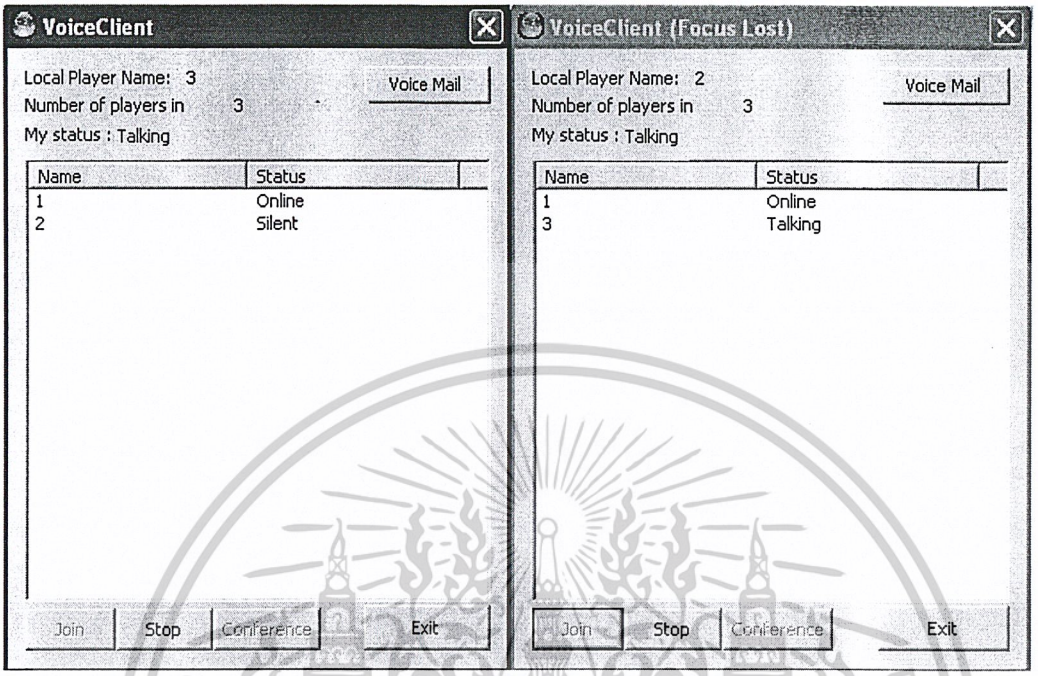


รูปที่ 4-2 หน้าต่างการทำงานของไคลแอนท์เมื่อติดต่อมายังเซิร์ฟเวอร์ได้แล้ว

โดยแอปพลิเคชันในส่วนไคลแอนท์นี้จะมีการให้บริการ 3 ส่วนด้วยกันคือ

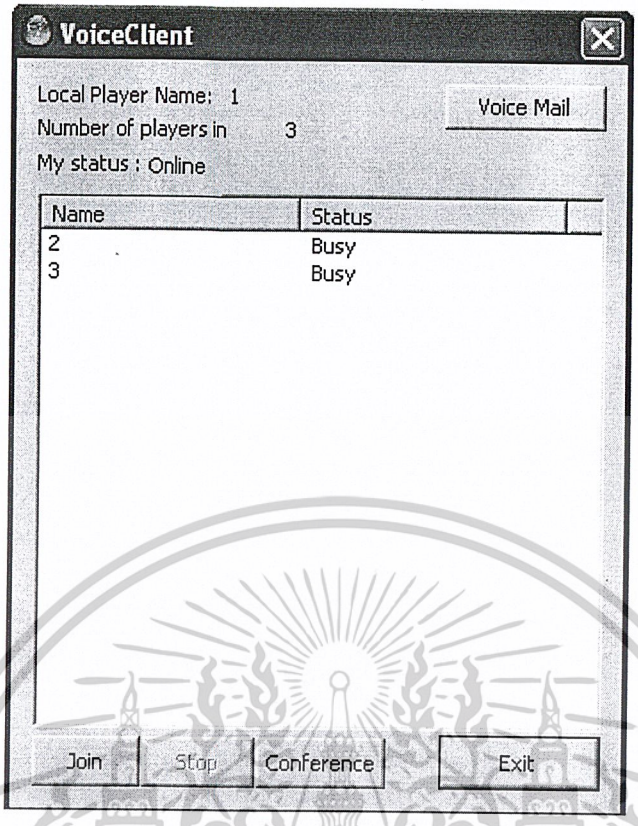
- **talking :** เป็นส่วนที่ให้บริการในการพูดคุยแบบ peer-to-peer กับยูสเซอร์คนอื่นที่แสดงสถานะของตัวเองอยู่ หาว่าเป็นสถานะ online เหมือนกันก็จะสามารถติดต่อพูดคุยได้โดยการคลิ๊กที่ปุ่ม *Join* ซึ่งจะส่งผลทำให้การแสดงผลของตัวเองเปลี่ยนไปคือ สถานะของตัวเอง(My status)เปลี่ยนเป็น talking และ สถานการณ์คุยของคนที่พูดคุยอยู่ (status) ก็จะเปลี่ยนแปลงตามระดับเสียงที่พูด หมายความว่าหากอีกด้านหนึ่งพูดอยู่สถานะก็จะเป็น talking และหากอีกด้านไม่ได้พูดสถานะก็จะเปลี่ยนไปเป็น Silent นอกจากนั้นสถานะของปุ่มกดก็จะเปลี่ยนไปโดย

เปลี่ยนสถานะของปุ่ม *Join* ให้ไม่สามารถคลิ๊กได้เพื่อไม่ให้มีการแทรกซ้อนของการคุยกันได้ ดังรูป



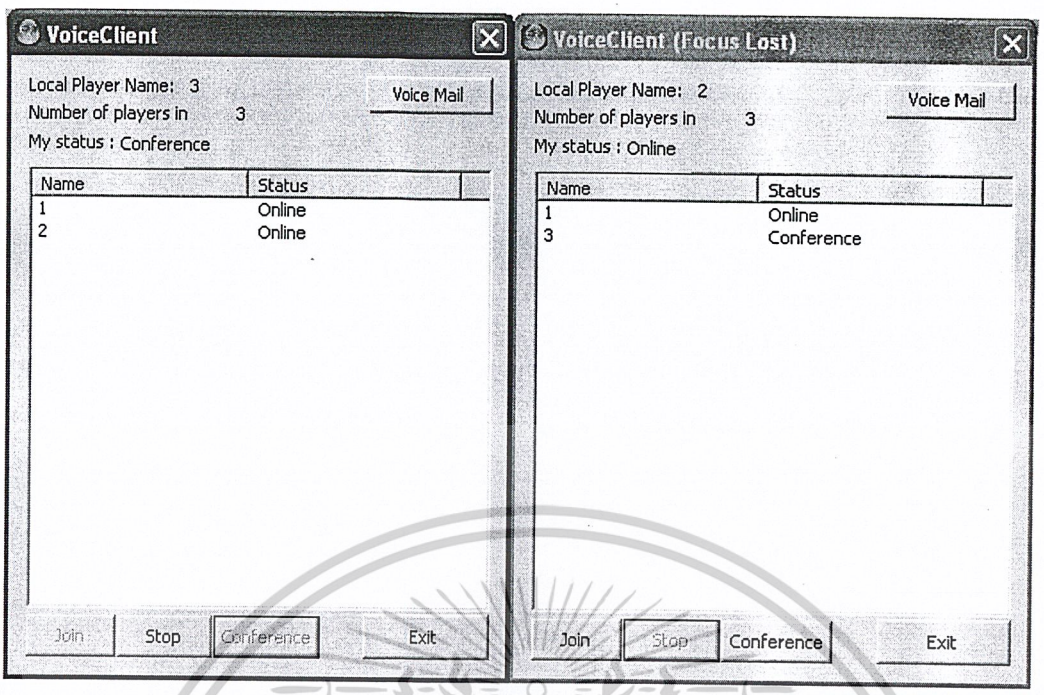
รูปที่ 4-3 หน้าต่างของไคลแอนท์เมื่อมีการเลือกพูดคุยแบบ Peer-to-peer

นอกจากนั้นการแสดงผลของไคลแอนท์คนอื่นๆ ก็จะเปลี่ยนไปโดยจะเห็นว่าสถานะของคนนั้นๆ เป็น Busy หากว่ามีการคุยกัน และยูสเซอร์นี้ก็จะไม่สามารถ คลิ๊กปุ่ม *Join* เพื่อพูดคุยกับคนอื่นๆ ที่มีสถานะ นี้ได้



รูปที่ 4-4หน้าต่างของไคลแอนท์ในการแสดงสถานะของไคลแอนท์คนอื่นที่พูดคุยอยู่

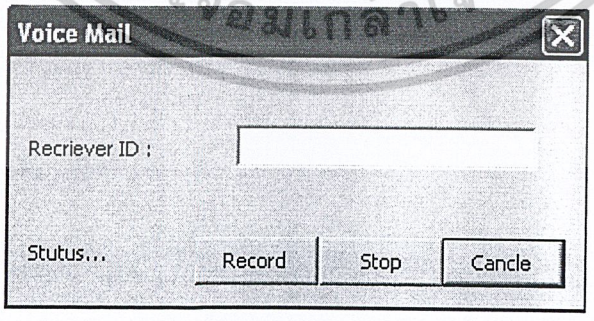
- **Conference :** เป็นส่วนที่ให้บริการในการพูดคุยแบบ conference กับยูสเซอร์คนอื่นโดยการคลิ๊กที่ปุ่ม *Conference* ซึ่งจะส่งผลทำให้การแสดงผลของตัวเองเปลี่ยนไปคือ สถานะของตัวเอง(My status) เปลี่ยนไปเป็น conference และสถานะการคุยของคนที่พูดคุยอยู่(status) นอกจากนั้นสถานะของปุ่มกดก็จะเปลี่ยนไปโดยเปลี่ยนสถานะปุ่ม *Join* และปุ่ม *Conference* ให้ไม่สามารถคลิ๊กได้ เพื่อไม่ให้มีการแทรกซ้อนของการคุยกันได้ ดังรูป



รูปที่ 4-5หน้าต่างของไคลแอนท์เมื่อมีการเลือกพูดคุยแบบ Conference

นอกจากนั้นการแสดงผลของไคลแอนท์คนอื่นๆ ก็จะเปลี่ยนไปโดยจะเห็นว่าสถานะของคนนั้นๆ เป็น conference แต่จะไม่มีผลใดๆ กับยูสเซอร์นี้ คือ สามารถที่จะ conference หรือพูดคุยกับคนอื่นๆที่มีสถานะ online ได้ตามปกติ

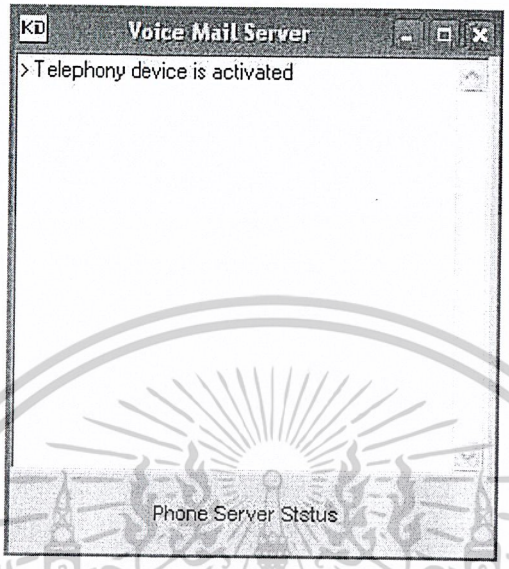
- **Voice Mail :** เป็นส่วนที่ให้ไคลแอนท์ใช้ในการฝากข้อความเสียงไปยังโทรศัพท์บ้าน โดยหากต้องการฝากข้อความเสียงก็คลิ๊กปุ่ม *VoiceMail* ก็จะแสดงหน้าต่างดังรูปที่ *4-6* ซึ่งหน้าต่างนี้ใช้สำหรับให้ยูสเซอร์ฝากข้อความ โดยมีการป้อนหมายเลขของผู้รับเพื่อส่งไปยังกล่องข้อความเสียงของผู้รับที่ต้องการ



รูปที่ 4-6 หน้าต่างของการรับฝากข้อความจากไคลแอนท์

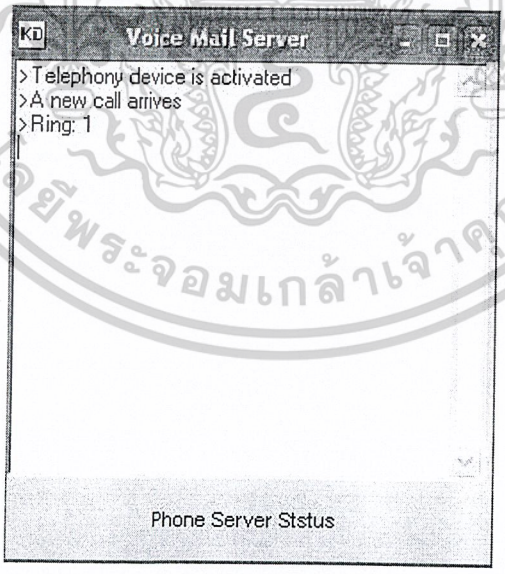
4.1.3 ตัวอย่างการใช้งานรับฝากข้อความผ่านทางโทรศัพท์

1. เริ่มต้นการใช้โปรแกรมจะมีหน้าต่างแสดงสถานะของเซิร์ฟเวอร์ขึ้นมา



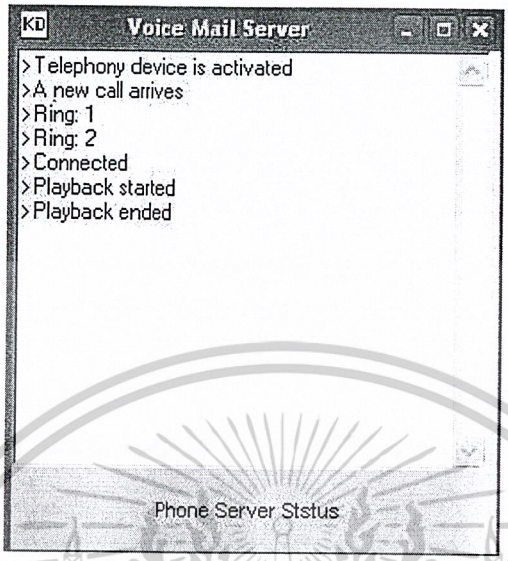
รูปที่4-7 หน้าต่างของเซิร์ฟเวอร์เมื่อเริ่มต้นการใช้งาน

2. จากนั้นเซิร์ฟเวอร์จะทำการฟังสัญญาณจากสายโทรศัพท์เมื่อมีสายเข้าจะเกิดผลดังนี้



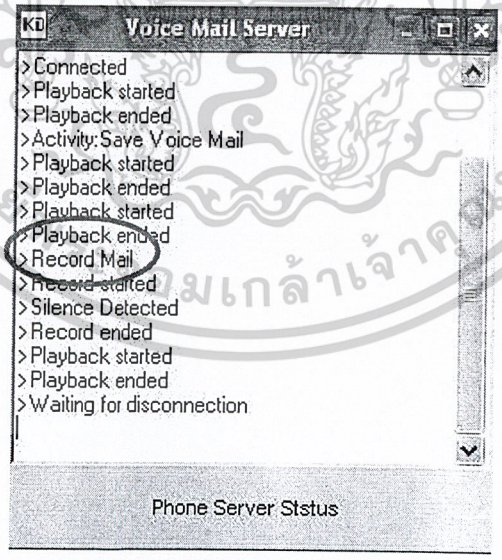
รูปที่ 4-8 หน้าต่างของเซิร์ฟเวอร์เมื่อมีสายเรียกเข้า

3. เซิร์ฟเวอร์จะทำการเล่นไฟล์เสียงที่จะบอกให้ยูสเซอร์เลือกการทำงานว่าจะฝากข้อความหรือฟังข้อความ



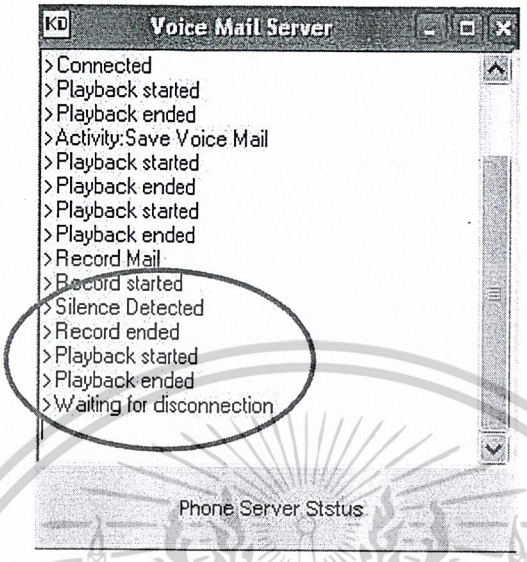
รูปที่ 4-9 หน้าต่างของเซิร์ฟเวอร์เมื่อมีการเล่นไฟล์เสียงต้อนรับ

4. ถ้าหากยูสเซอร์ต้องการที่จะฝากข้อความก็จะทำการกด 1 จากนั้นก็จะมีเสียงจากระบบให้กด USER ID แล้วตามด้วย *



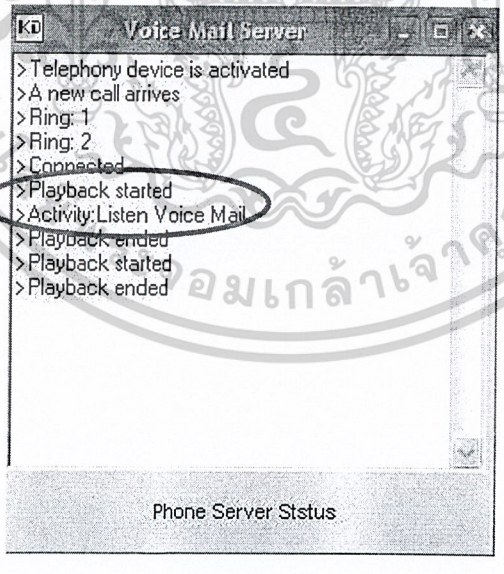
รูปที่ 4-10 หน้าต่างของเซิร์ฟเวอร์เมื่อเริ่มต้นการใช้งานการฝากข้อความ

5. จากนั้นระบบจะทำการบันทึกข้อความเสียงจนกระทั่งจับสัญญาณได้ว่ายูสเซอร์หยุดพูดจะทำการบันทึกและตัดการเชื่อมต่อ



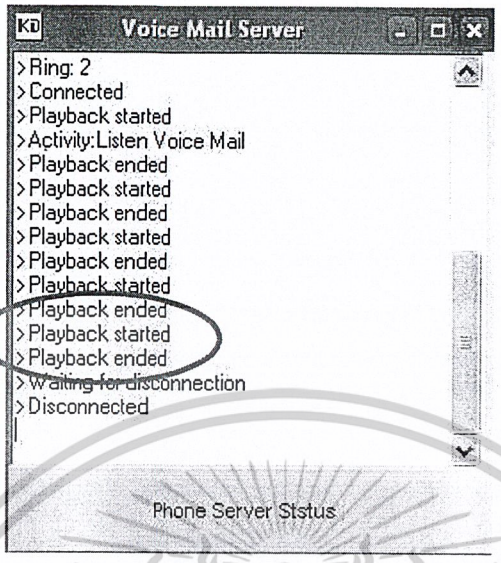
รูปที่ 4-11 หน้าต่างของเซิร์ฟเวอร์เมื่อยูสเซอร์หยุดพูดและตัดการเชื่อมต่อ

6. ถ้าหากว่ายูสเซอร์ต้องการจะฟังข้อความเสียงของตนจะกด 2 เพื่อบอกความต้องการ จากนั้นจะได้ยินคำแนะนำถัดไป



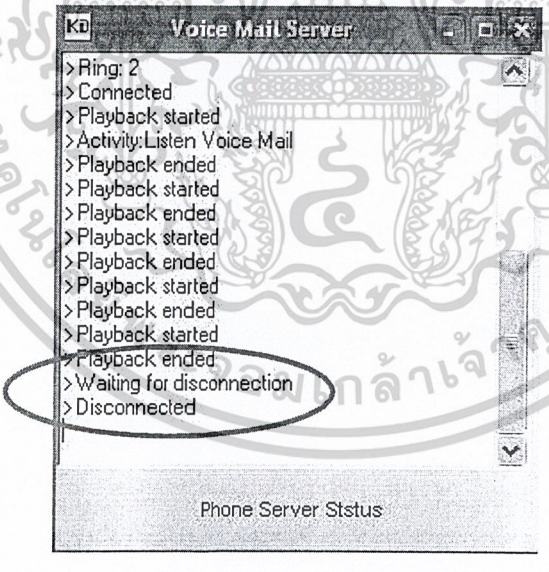
รูปที่ 4-12 หน้าต่างของเซิร์ฟเวอร์เมื่อเริ่มต้นการใช้งานการฟังข้อความเสียง

7. กด User ID และ Password จากนั้นรอฟังข้อความ



รูปที่ 4-13 หน้าต่างของเซิร์ฟเวอร์เมื่อมีการเล่นไฟล์ข้อความเสียง

8. เมื่อข้อความจบเซิร์ฟเวอร์จะทำการตัดการติดต่อทันที



รูปที่ 4-14 หน้าต่างของเซิร์ฟเวอร์ในการตัดการเชื่อมต่อ

9. และหากว่ามีการกด User ID หรือ Password ผิดจะตัดการติดต่อทันที

4.2 การทดสอบประสิทธิภาพของแอปพลิเคชัน

เป็นการทดสอบความสามารถในการทำงานของแอปพลิเคชัน โดยจะพิจารณาจาก แบนด์วิดธ์ (Bandwidth) ที่ใช้สำหรับแอปพลิเคชัน ซึ่งการทดสอบจะใช้โปรแกรม Ultra Network Snifferในตรวจสอบแพ็กเก็จของแอปพลิเคชัน ซึ่งโปรแกรมนี้สามารถที่จะกรองเอาเฉพาะแพ็กเก็จที่เกี่ยวข้องกับแอปพลิเคชันเท่านั้น และสามารถหาผลรวมของปริมาณข้อมูลที่สื่อสารกันออกมาแสดงผลในรูปแบบของกราฟ และแผนภูมิซึ่งเข้าใจง่าย

เพื่อเป็นการควบคุมผลกระทบจากแพ็กเก็จอื่นๆในเครือข่าย การทดสอบจึงใช้สายครอสกันระหว่างคอมพิวเตอร์เพื่อทดสอบในส่วนที่เกี่ยวข้องกับเครือข่ายอินเตอร์เน็ต

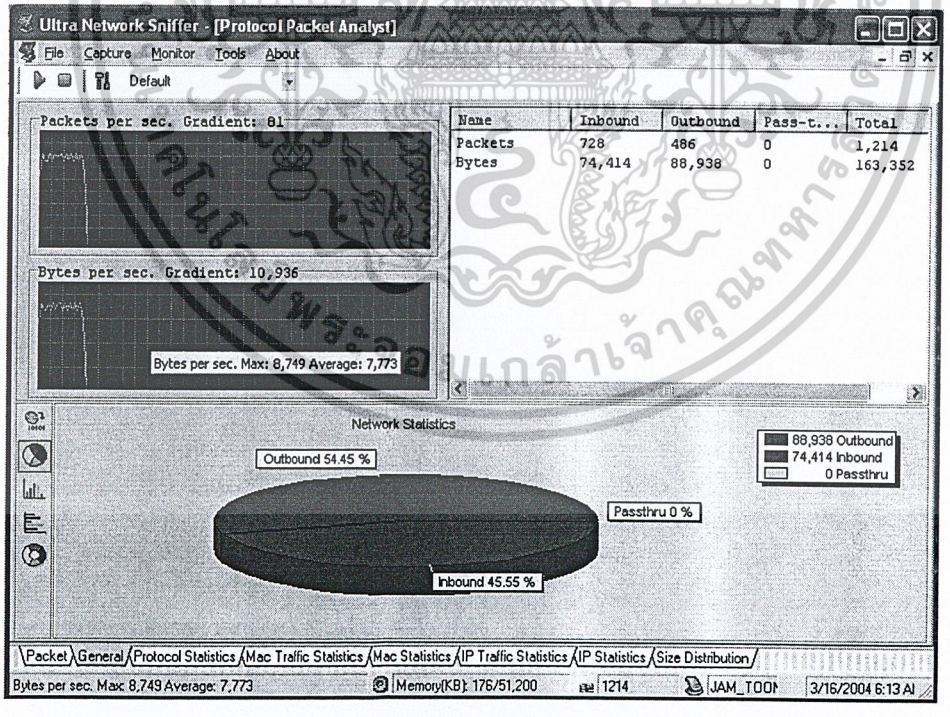
4.2.1 การทดสอบการทำงานของเซิร์ฟเวอร์

สามารถแยกการทดสอบเป็นส่วนๆ ดังต่อไปนี้

- **การทดสอบการทำงานของเซิร์ฟเวอร์ต่างๆ**

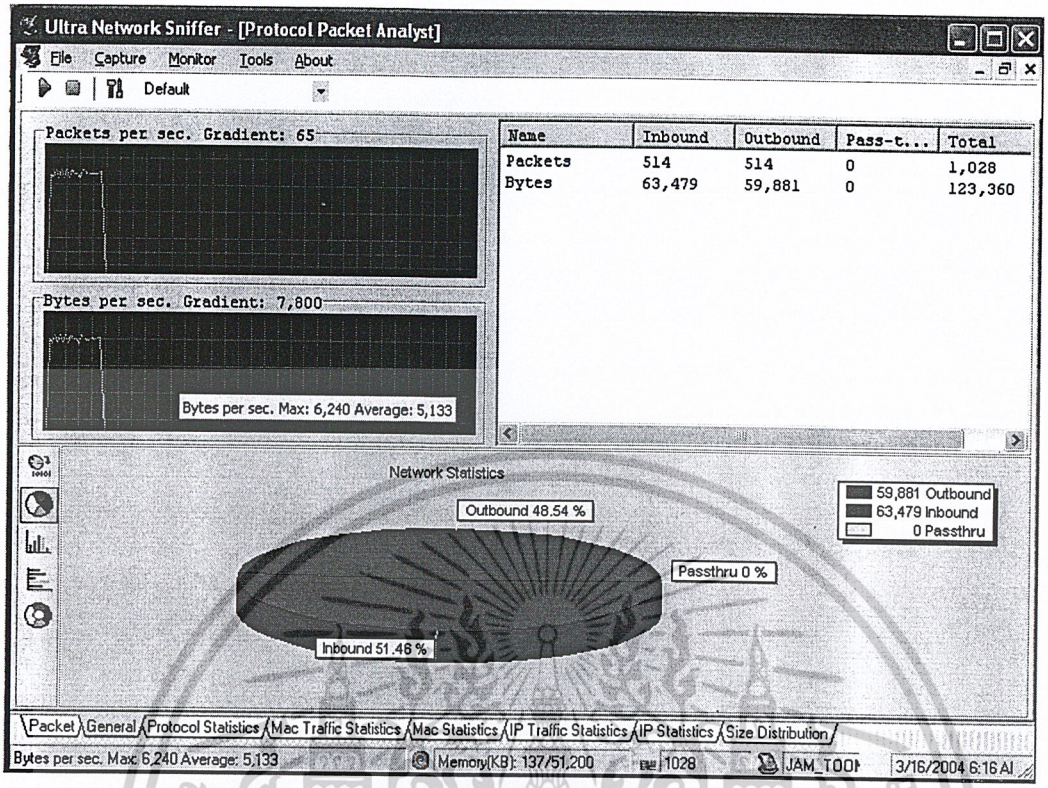
เซิร์ฟเวอร์สามารถเซตการทำงานได้ 2 แบบคือ แบบ Forwarding และแบบ Mixing โดยการวัดผลทั้งสองแบบได้กำหนดให้อยู่ในปัจจัยเดียวกันคือ ใช้ Codec แบบ Microsoft GSM 6.10 13 kbps และใช้ไคลแอนท์ที่พูดคุยกันเพียงคู่เดียวเท่านั้น ซึ่งจะมีลักษณะที่แตกต่างกันดังนี้

- **Forwarding Server :**

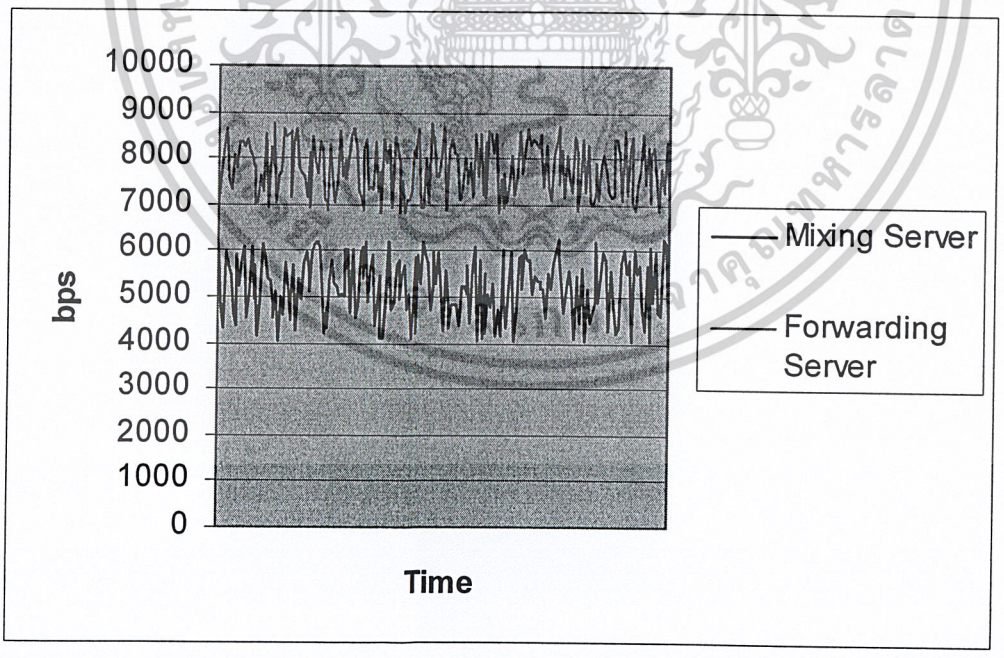


รูปที่ 4-15 ผลการตรวจสอบปริมาณการรับส่งข้อมูลของ Forwarding Server

- **Mixing Server :**



รูปที่ 4-16 ผลการตรวจสอบปริมาณการรับส่งข้อมูลของ Mixing Server



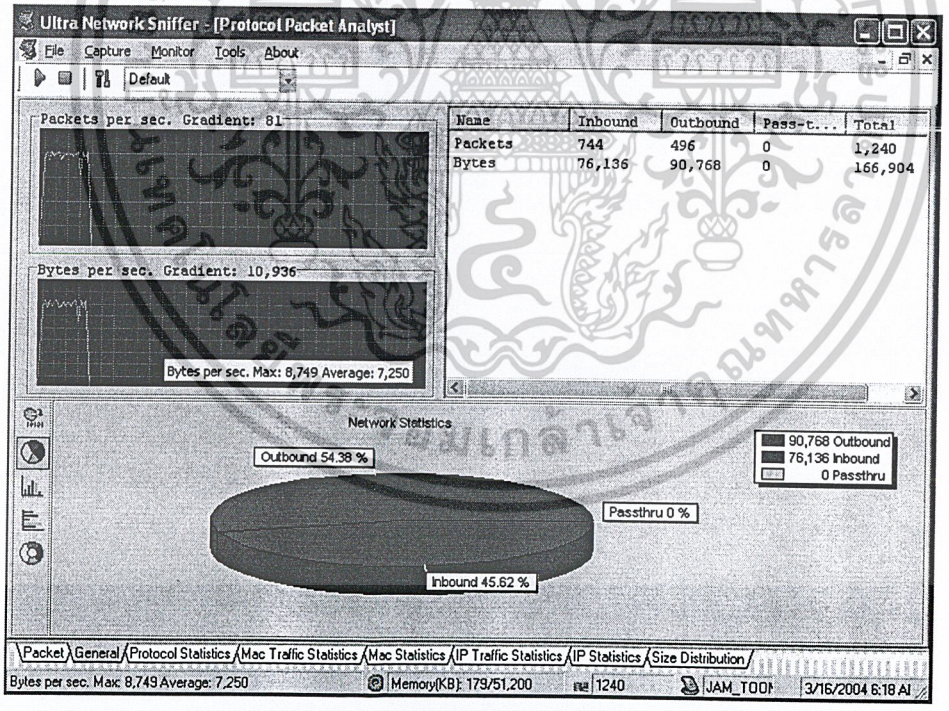
รูปที่ 4-17 กราฟเปรียบเทียบการใช้ Bandwidth ระหว่าง Mixing และ Forwarding Server

จากผลการทดสอบจะสังเกตได้ว่าแบบ Forwarding Server ใช้แบนด์วิดธ์ 7773 bytes/sec ส่วน Mixing Server ใช้เพียง 5133 bytes/sec ทั้งนี้เพราะแบบ Mixing Server ได้มีการรวมเอาสัญญาณเสียงจาก Client ทั้งหมดก่อนส่งออกไป ทำให้ใช้แบนด์วิดธ์น้อยกว่า แต่ก็มีผลคือทำให้เครื่องเซิร์ฟเวอร์ใช้ซีพียูจำนวนมากในโปรเซสนี้ อีกทั้งการใช้เซิร์ฟเวอร์แบบ Mixing Server จะไม่ทราบสถานะของคู่สนทนา เพราะไม่สามารถแยกเสียงที่เข้ามาได้ว่าเป็นของไคลแอนท์คนใด

- **การทดสอบการทำงานของเซิร์ฟเวอร์ในการรองรับปริมาณของไคลแอนท์**

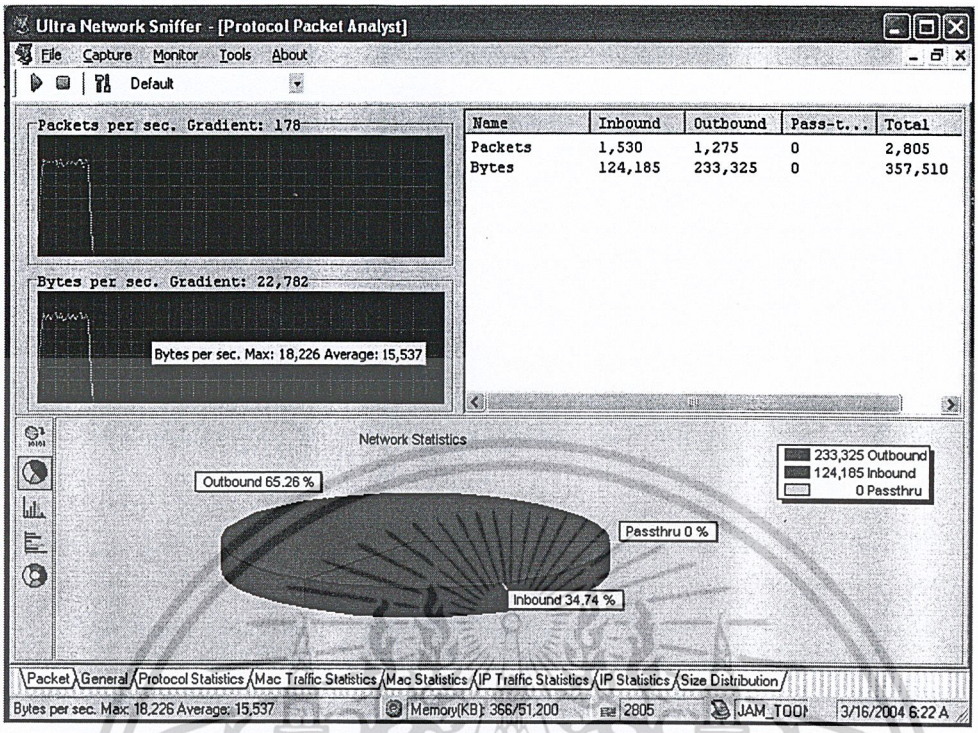
เป็นการทดสอบว่าเซิร์ฟเวอร์สามารถรองรับไคลแอนท์ได้ปริมาณมากน้อยเพียงใด โดยดูจากแบนด์วิดธ์ที่ใช้ และยังฟังคุณภาพของเสียง รวมทั้งดีเลย์ที่เกิดขึ้นด้วย ซึ่งได้มีการกำหนดปัจจัยภายนอกให้คงที่คือ กำหนดให้ใช้ Codec แบบ Microsoft GSM 6.10 13 kbps และใช้เซิร์ฟเวอร์แบบ Forwarding Server โดยทุกๆไคลแอนท์จะคุยกันแบบ conference เพื่อให้สามารถสังเกต *maximum bandwidth* ที่เกิดจากไคลแอนท์ทั้งหมด

- **ไคลแอนท์ 2 คน :**



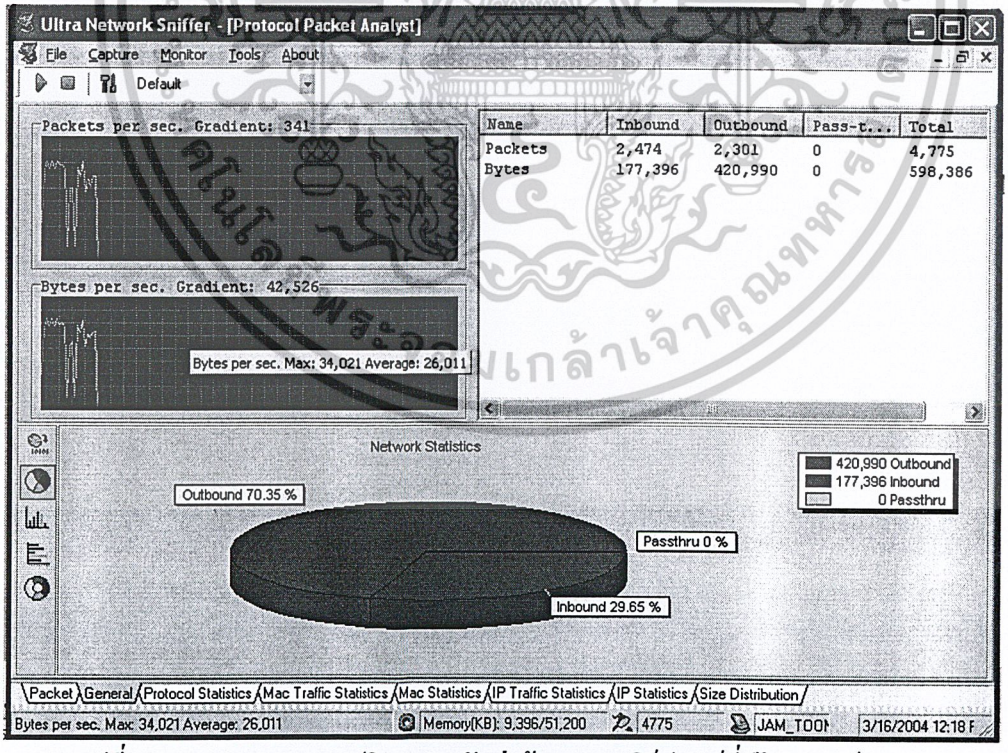
รูปที่ 4-18 ผลการตรวจสอบปริมาณการรับส่งข้อมูลของเซิร์ฟเวอร์ที่มีไคลแอนท์เพียง 2 คน

- ไคลแอนท์ 5 คน :



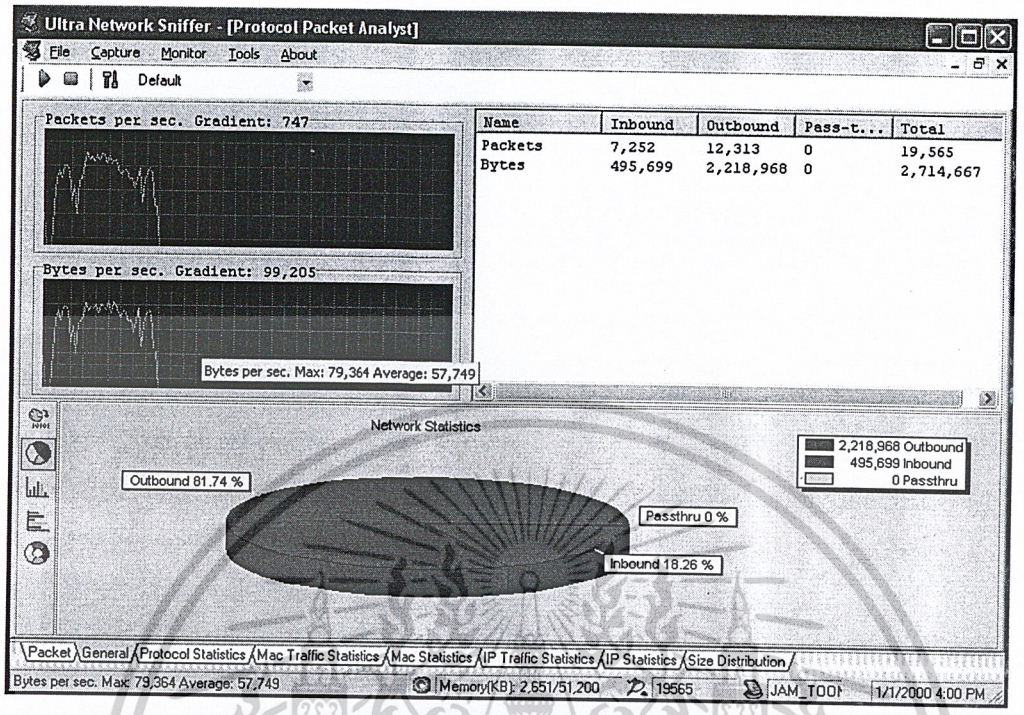
รูปที่ 4-19 ผลการตรวจสอบปริมาณการรับส่งข้อมูลของเซิร์ฟเวอร์ที่มีไคลแอนท์ 5 คน

- ไคลแอนท์ 10 คน :

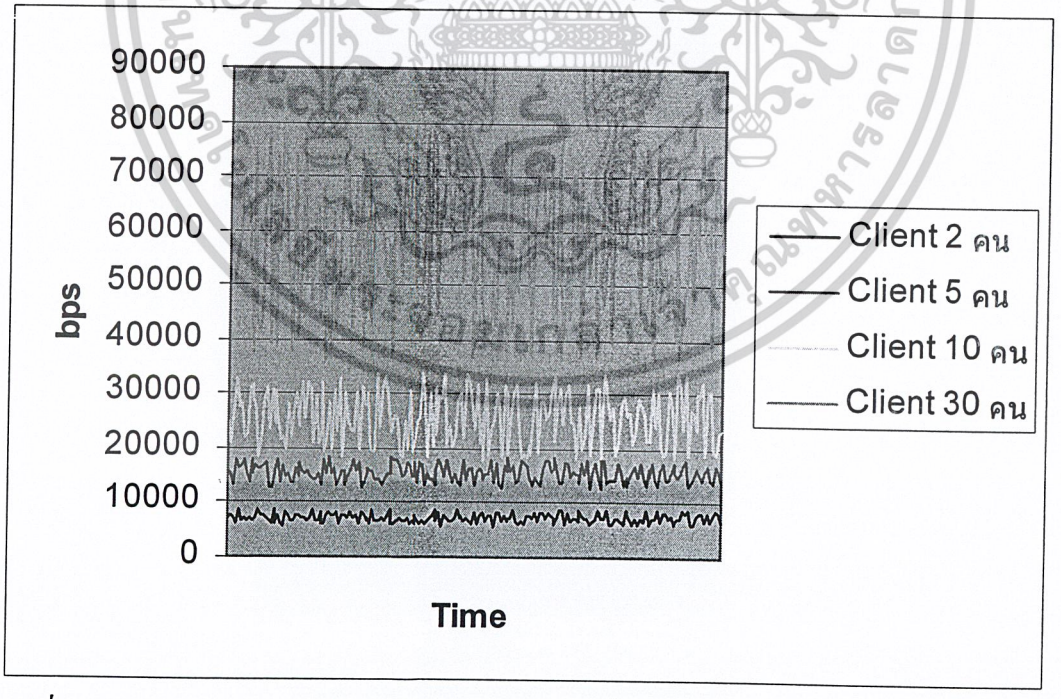


รูปที่ 4-20 ผลการตรวจสอบปริมาณการรับส่งข้อมูลของเซิร์ฟเวอร์ที่มีไคลแอนท์ 10 คน

- ไคลแอนท์ 30 คน :



รูปที่ 4-21 ผลการตรวจสอบปริมาณการรับส่งข้อมูลของเซิร์ฟเวอร์ที่มีไคลแอนท์ 30 คน



รูปที่ 4-22 กราฟเปรียบเทียบการใช้แบนด์วิดธ์ของเซิร์ฟเวอร์เมื่อมีไคลแอนท์จำนวน 2,5,10 และ 30 คน

จากกราฟแสดงการเปรียบเทียบการใช้แบนด์วิดธ์ของเซิร์ฟเวอร์ จะสังเกตได้ว่าเมื่อมีไคลแอนท์เพิ่มขึ้นแบนด์วิดธ์ก็จะเพิ่มขึ้นมาก และช่วงของการใช้แบนด์วิดธ์ก็กว้างขึ้นมาก และจะเพิ่มขึ้นอย่างรวดเร็วขึ้น ในส่วนของคุณภาพเสียงที่ได้ยินนั้นการประชุมกันไปเกิน 10 คนแม้จะมีดีเลย์เกิดขึ้นแต่ก็พอยอมรับได้ แต่หากจำนวนไคลแอนท์มีถึง 30 คน เสียงจะก้องและฟังไม่รู้เรื่อง

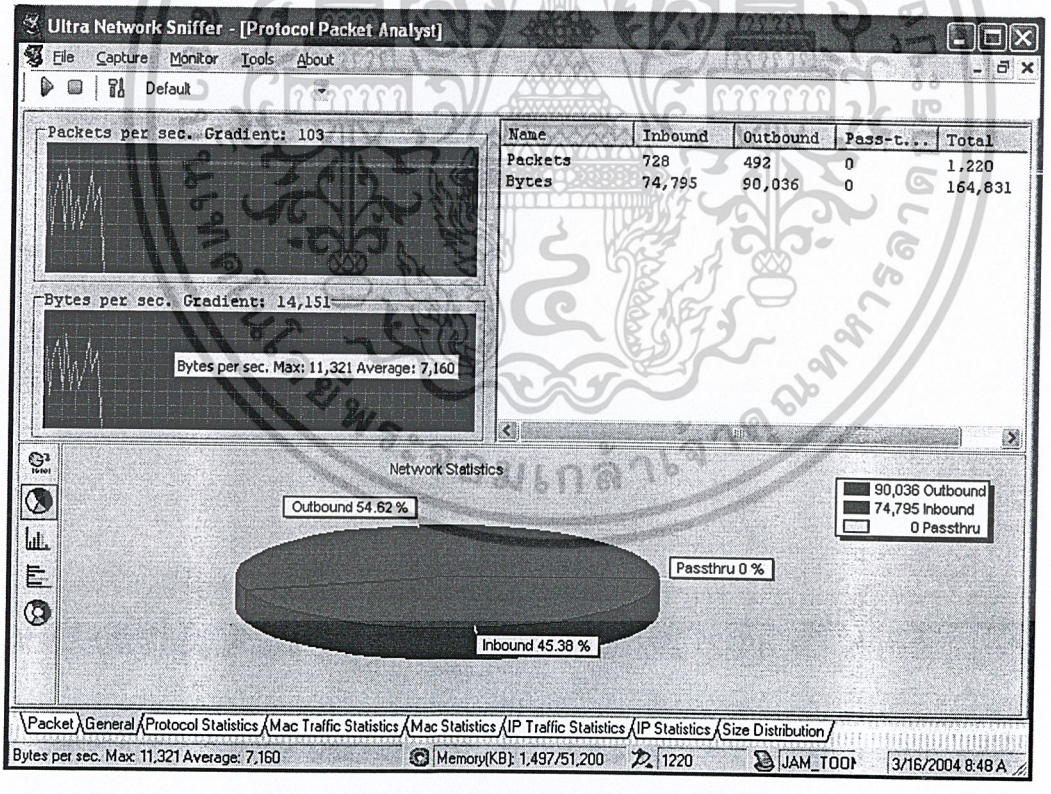
4.2.2 การทดสอบการทำงานของไคลแอนท์

จะเป็นการทดสอบการทำงานเฉพาะที่ไคลแอนท์เดียว โดยสามารถแยกการทดสอบเป็นส่วนๆ ดังต่อไปนี้

- **การทดสอบการทำงานของไคลแอนท์ตามฟังก์ชันการใช้งาน**

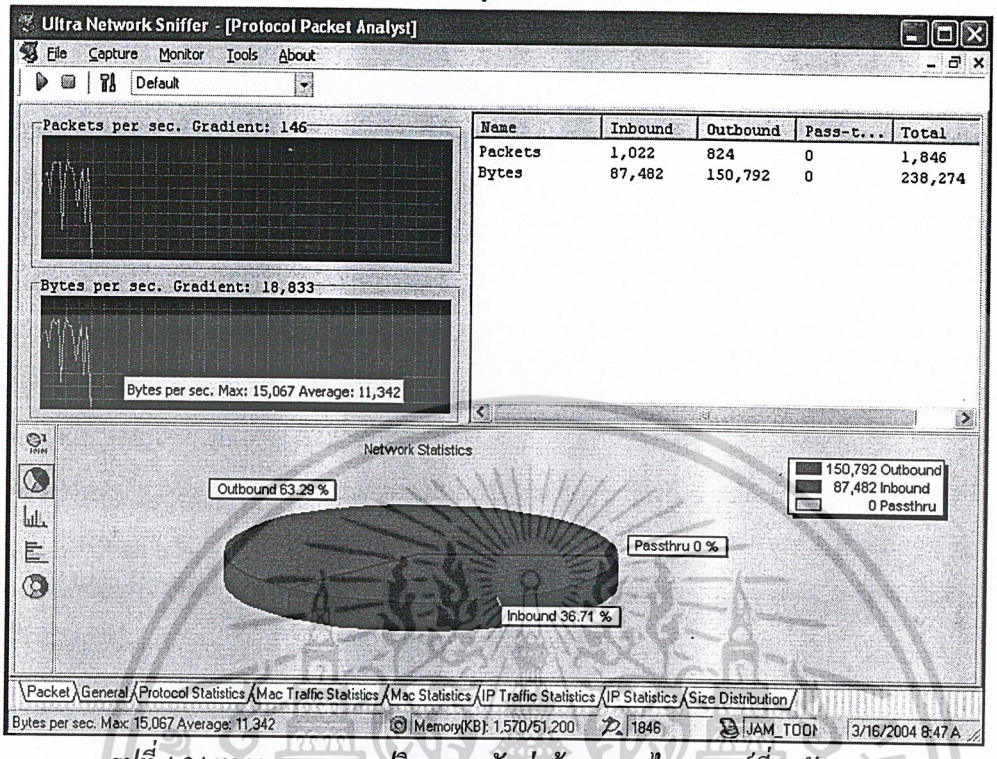
จะทำการทดสอบเพื่อเปรียบเทียบการทำงานของไคลแอนท์แบบต่างโดยได้กำหนดปัจจัยในการควบคุมคือ กำหนดให้ใช้ Codec แบบ Microsoft GSM 6.10 13 kbpsและใช้เซิร์ฟเวอร์แบบ Forwarding Server และ กำหนดให้มีไคลแอนท์ในระบบ 4 ไคลแอนท์

— **Talking :** ในการทดสอบได้มีการกำหนดให้ไคลแอนท์คุยกันเป็นคู่ทั้ง 2 คู่ซึ่งได้ผลดังนี้

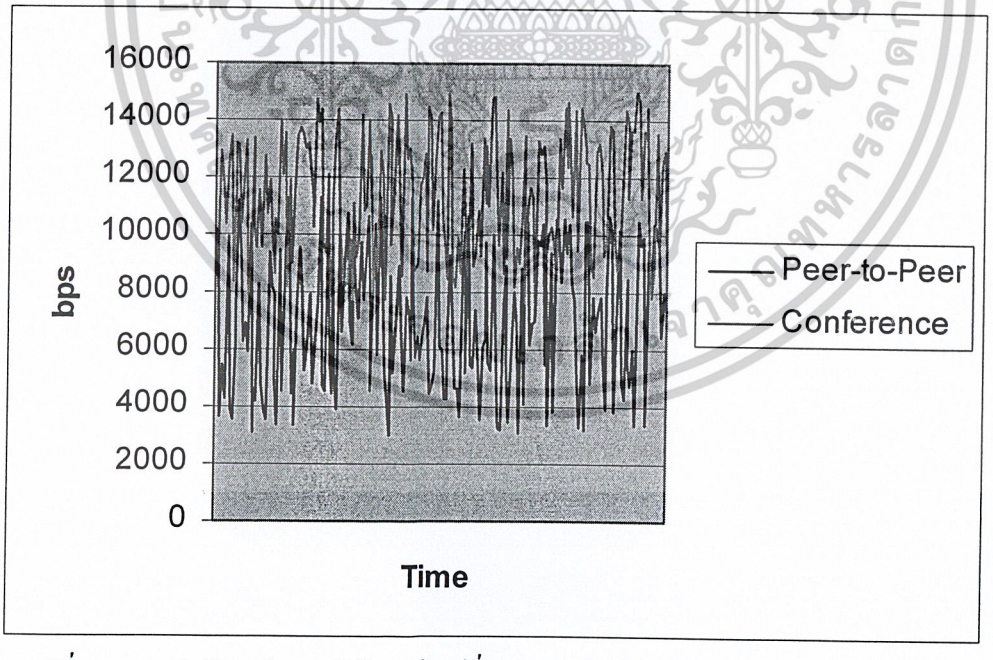


รูปที่ 4-23 ผลการตรวจสอบปริมาณการรับส่งข้อมูลของไคลแอนท์ที่คุยกันแบบ Peer-to-peer

– **Conference** : แบบนี้ในการทดสอบจะมีการกำหนดให้ไคลแอนท์ทั้ง 4 คนทำการ conference กัน และกำหนดให้ conference กันทีละคู่ซึ่งได้ผลใกล้เคียงกันดังนี้



รูปที่ 4-24 ผลการตรวจสอบปริมาณการรับส่งข้อมูลของไคลแอนท์ที่คุยกันแบบ Conference



รูปที่ 4-25 กราฟเปรียบเทียบการใช้แบนด์วิดธ์ที่ไคลแอนท์ระหว่างการคุยแบบ Peer-to-Peer และ Conference

จากกราฟเปรียบเทียบการใช้แบนด์วิดธ์ระหว่างการพูดคุยกันแบบ peer-to-peer และ conference แบบ conference จะใช้แบนด์วิดธ์สูงกว่า โดยถ้าสังเกตจากจำนวนแพ็กเก็จแสดงในกราฟของโปรแกรมที่ใช้ทดสอบจะเห็นว่าแบบ peer-to-peer จะมีแพ็กเก็จเข้าและออก จำนวนเท่าๆ กัน แต่ในส่วนแบบ conference นั้นแพ็กเก็จที่ถูกส่งออกไปจะมากกว่าทั้งนี้เพราะว่าแบบ conference ไคลแอนท์ต้องส่งข้อมูลตัวเองให้ทุกๆ ไคลแอนท์ที่ conference อยู่แต่ถ้าเป็นแบบ peer-to-peer จะส่งข้อมูลให้เฉพาะคู่สนทนาเท่านั้นแบนด์วิดธ์ขาเข้าและออกจึงเท่าๆกัน

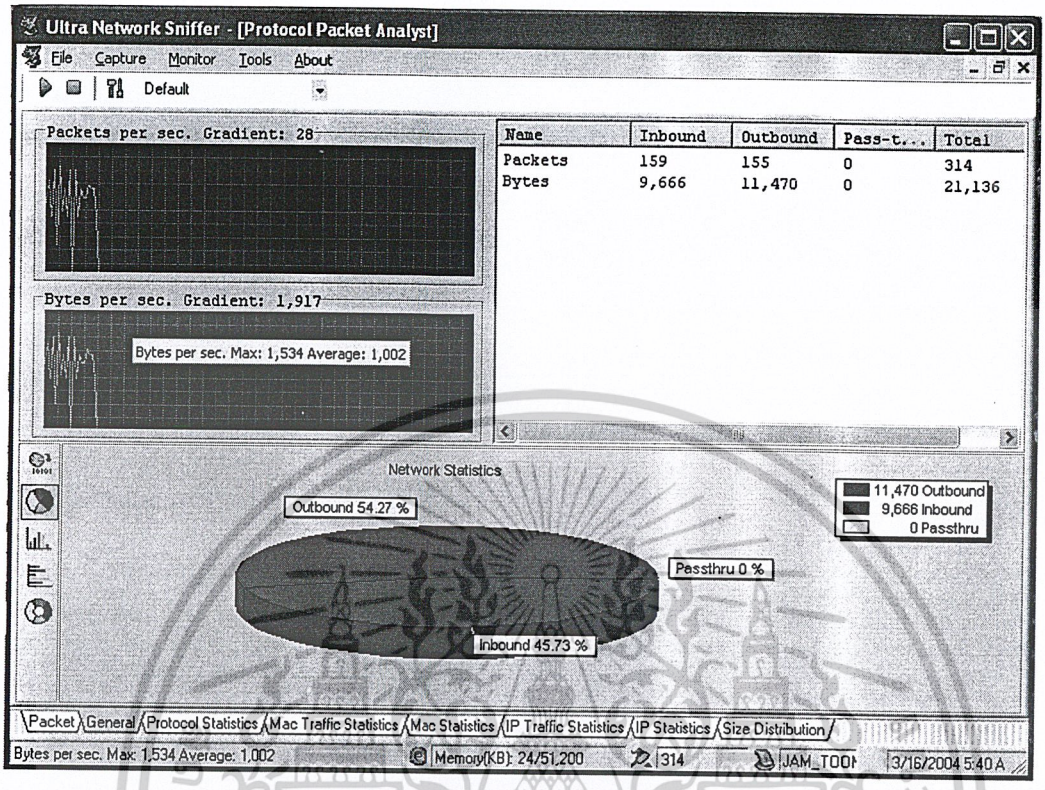
- **การทดสอบการทำงานของไคลแอนท์ในการใช้ Codec แบบต่างๆ**

แอปพลิเคชันสนับสนุนการ Codec เสียงแบบต่างๆ ซึ่งแต่ละแบบมีลักษณะที่แตกต่างกัน และยังมีประสิทธิภาพต่างกัน ซึ่งได้ทำการทดลอง Codec เสียงทั้งหมดที่มีใน API ของไดเร็กเพลย์ เพื่อเปรียบเทียบคุณภาพว่าแบบใดเหมาะสมกับแอปพลิเคชันที่สุด โดยได้กำหนดปัจจัยในการควบคุมคือ กำหนดให้ใช้เซิร์ฟเวอร์แบบ Forwarding Server และ กำหนดให้มีไคลแอนท์ในระบบ 2 ไคลแอนท์คุยกันแบบ peer-to-peer

เมื่อทำการวัดค่าต่างๆ จะได้ผลออกมาดังต่อไปนี้ จะแสดงผลเฉพาะการทดสอบโดยใช้โปรแกรม Ultra Network Sniffer เพียง Codec เดียว ส่วนอันอื่นๆ จะอธิบายในเชิงคุณภาพเสียงและดีเลย์แทน

- Voxware SC03 3.2 Kbps
- Voxware SC06 6.4 Kbps
- TrueSpeech 8 Kbps
- Microsoft® GSM 13 Kbps
- Microsoft® ADPCM 32 Kbps
- Microsoft® PCM 64 Kbps

- Voxware VR12 1.2 Kbps



รูปที่ 4-26 ผลการตรวจสอบปริมาณการรับส่งข้อมูลของเซิร์ฟเวอร์ที่ใช้

Codec แบบ Voxware VR121.2 Kps

จะเห็นได้ว่า Codec แต่ละตัวจะมีอัตราการสุ่มข้อมูลไม่เท่ากันแม้จากการทดลองแบนด์วิดธ์ที่วัดได้อาจไม่เท่ากับที่มารตฐานที่กำหนดมา โดยใน Codec แบบ Voxware นั้นแม้จะใช้แบนด์วิดธ์ต่ำ แต่คุณภาพเสียงค่อนข้างต่ำเช่นกัน อย่างใน Voxware VR12 นั้นเสียงที่ฟังกระท่อนกระแท่นมาก และยิ่งเลือก Codec ที่มีการสุ่มที่สูงขึ้น คุณภาพเสียงก็จะมากขึ้น แต่ก็จะเริ่มมีดีเลย์ อย่างใน Microsoft® PCM 64 Kbps นั้นเกิดดีเลย์สูงมากจนฟังไม่เป็นประโยค

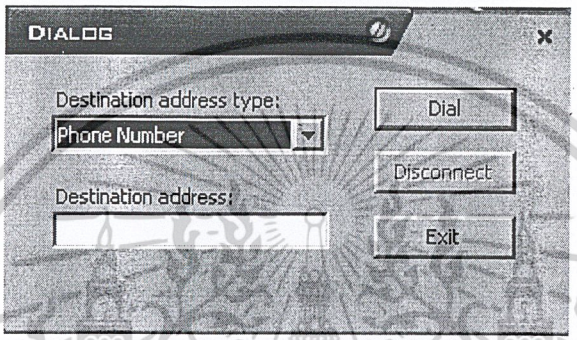
เพราะฉะนั้นการสุ่มคุณภาพสุ่มคุณภาพเสียงก็จะมีผลต่อคุณภาพเสียงและดีเลย์ ยิ่งสุ่มมากคุณภาพเสียงก็ยิ่งมาก แต่ เมื่อสุ่มมากปริมาณข้อมูลก็จะละเอียด และมีปริมาณเยอะ ทำให้กินแบนด์วิดธ์มากขึ้นเรื่อยๆ ถ้ายิ่งใช้แบนด์วิดธ์มากก็จะทำให้เกิดความล่าช้ามากขึ้นด้วย เพราะฉะนั้นเราควรเลือกใช้ Codecให้เหมาะกับแบนด์วิดธ์ของเรา ยกตัวอย่างเช่น ถ้าหากว่าเราอยู่บนเครือข่ายอีเธอร์เน็ต เราก็สามารถที่จะเลือกใช้ Codec ที่มีอัตราการสุ่มของข้อมูลสูงๆ เพื่อให้ได้เสียงที่มีคุณภาพดี แต่ถ้าเราต่ออินเตอร์เน็ตจากที่บ้านมา เราก็สมควรใช้ Codec ที่มีอัตราการสุ่มต่ำๆ เพื่อให้เกิดความล่าช้าน้อยๆ

4.2.3 การทดสอบการทำงานของไคลเอนท์ในการเรียกโทรศัพท์พื้นฐาน

การทดสอบการทำงานพื้นฐานของไคลแอนท์นั้นจะทดสอบทีละขั้นตอนตามฟังก์ชันการใช้งานของโทรศัพท์โดยเราจะทำการทดสอบทีละส่วนดังต่อไปนี้

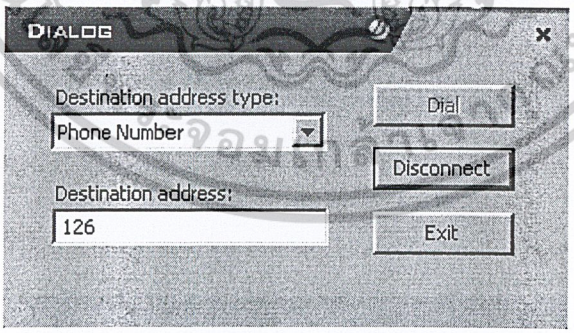
- **Dial:** เมื่อมีการคลิ๊กปุ่ม Dial จะเป็นส่วนของการโทรศัพท์จากแอปพลิเคชันไปยังโทรศัพท์บ้าน

1. เมื่อเรียกใช้ฟังก์ชันการทำงานของการโทรศัพท์ขึ้นมาจะมีหน้าต่าง ด้านต่างการติดต่อปรากฏขึ้นมา



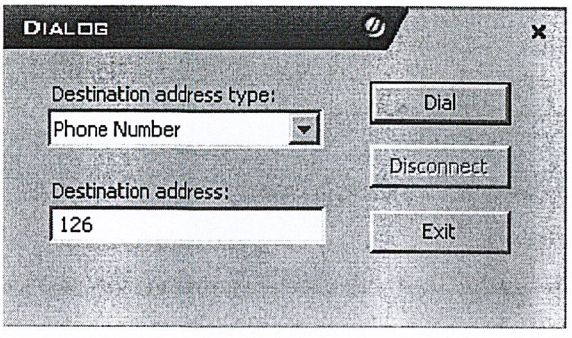
รูปที่ 4-27 หน้าต่างการโทรศัพท์ไปยังโทรศัพท์

2. ใส่หมายเลขโทรศัพท์ที่ต้องการติดต่อลงไป จากนั้นคลิ๊กที่ปุ่ม *Dial* หลังจากคลิ๊กปุ่มแล้ว ปุ่ม *Dial* จะหายไป แล้วแอปพลิเคชันจะทำการโทรศัพท์ไปยังหมายเลขดังกล่าว (ตัวอย่างจะเป็นหมายเลข 126)



รูปที่ 4-28 หน้างต่างแสดงการโทรศัพท์ไปยังหมายเลขที่ต้องการ

3. เมื่อสนทนาจบแล้วต้องการที่จะวางสายให้คลิ๊กปุ่ม *Disconnect* จากนั้น แอปพลิเคชันจะทำการวางสายและกลับมายังหน้าต่างการติดต่อ เหมือนเดิม



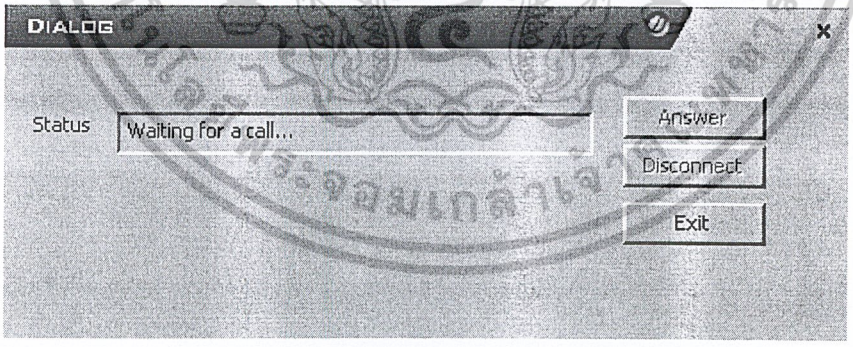
รูปที่ 4-29 หน้าต่างการติดต่อหลังจากวางสายโทรศัพท์แล้ว

4. ถ้าหากว่าต้องการจะกลับไปใช้ในส่วนของการคุยกันระหว่าง คอมพิวเตอร์ ให้คลิ๊กปุ่ม *Exit* แล้วหน้าต่างการติดต่อโทรศัพท์จะหายไป

5. ถ้าหากระหว่างการคุยกันนั้นคลิ๊กปุ่ม *Exit* การสนทนานั้นจะยุติลงและสายโทรศัพท์จะตัดการเชื่อมต่อทันที

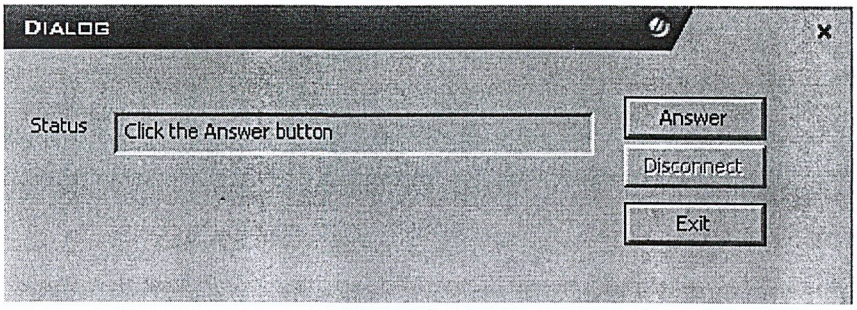
- **Incoming:** เป็นส่วนของการให้แอปพลิเคชันรอรับสัญญาณโทรศัพท์ โดยมีขั้นตอนดังนี้

1. เมื่อเรียกใช้การทำงานในส่วนของการรับโทรศัพท์นั้นจะมีหน้างต่างดังรูปด้านล่างปรากฏขึ้นมา และที่หน้าจอสถานะจะมีสถานะของแอปพลิเคชันปรากฏอยู่ ขณะที่รอสายนั้นจะมีสถานะเป็น *"Waiting for call..."*



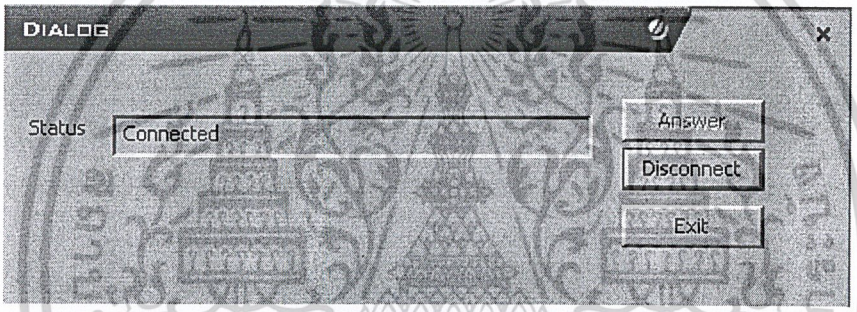
รูปที่ 4-30 หน้าต่างการรอรับสายโทรศัพท์

2. เมื่อมีสายเรียกเข้ามาสถานะจะเปลี่ยนไปเป็น *"Click the Answer Button"* และปุ่ม *Answer* จะปรากฏขึ้นมา ถ้าเราต้องการับสายให้เราคลิ๊กปุ่ม *Answer*



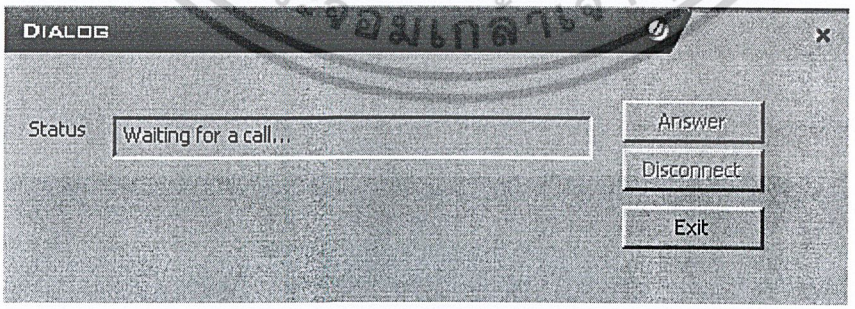
รูปที่ 4-31 หน้าต่างการรอคำสั่งรับสายโทรศัพท์

3. หลังจากนั้นสถานะจะเปลี่ยนไปเป็น *Connected* และปุ่ม *Answer* จะหายไป ในขณะนี้จะสามารถพูดคุยกันได้



รูปที่ 4-32 หน้าต่างการสนทนา

4. เมื่อต้องการวางสายให้คลิ๊กปุ่ม *Disconnect* แล้วสายโทรศัพท์จะวางลง และหน้าต่างจะกลับมาอยู่ในสถานะรอสายโทรศัพท์เรียกเข้า

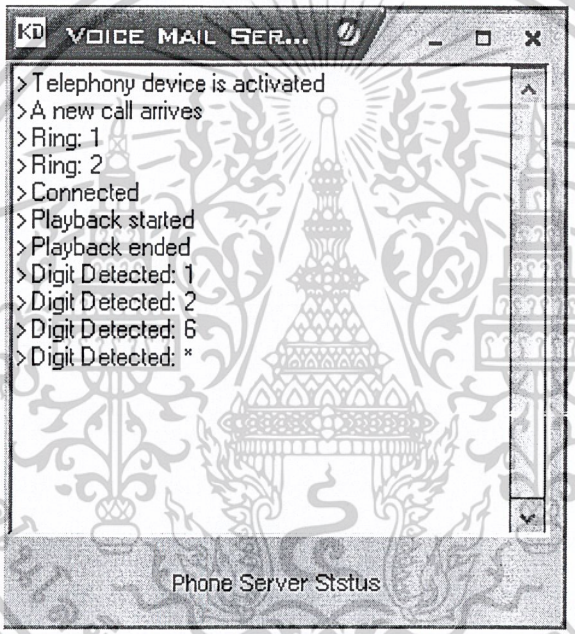


รูปที่ 4-33 หน้าต่างการรอรับสายโทรศัพท์

5. ถ้าหากว่าต้องการหยุดการติดต่อกับโทรศัพท์และกลับไปทำงานในส่วนของการคุยกันระหว่างคอมพิวเตอร์ให้คลิ๊กปุ่ม *Exit*
6. ถ้าหากระหว่างการสนทนามีการคลิ๊กปุ่ม *Exit* แล้วการสนทนาจะถูกตัดลงโดยอัติโนมัติ

- **Answer :** เมื่อมีสัญญาณเรียกเข้าสามารถดีเทค สัญญาณได้ดังต่อไปนี้
  - สัญญาณมีสายเรียกเข้า
  - สัญญาณว่าเสียงที่เข้ามาหยุดไป
  - สัญญาณการกดปุ่มโทรศัพท์

และสามารถเล่นและบันทึกข้อความเสียงผ่านทางโทรศัพท์ได้



รูปที่ 4-34 หน้าต่างของแอปพลิเคชันในส่วนของการติดต่อกับโทรศัพท์

อธิบายการทำงานของส่วนติดต่อกับโทรศัพท์

1. เริ่มต้นด้วยการรันเซิร์ฟเวอร์ไว้
2. ทำการรอรับสายเรียกเข้าจากโทรศัพท์
3. เมื่อมีสายโทรศัพท์เข้ามาเซิร์ฟเวอร์จะทำการเล่นไฟล์เสียงต้อนรับและสอบถามว่ายูสเซอร์ต้องการใช้งานแบบใด ระหว่าง ตรวจสอบข้อความเสียง หรือ ฝากข้อความเสียง

4. ถ้าหากยูสเซอร์ต้องการจะฝากข้อความเสียงให้กดรหัสของปลายทาง จากนั้นพูดข้อความไว้เซิร์ฟเวอร์จะทำการบันทึกและทำการตัดการเชื่อมต่อ
5. ถ้าต้องการตรวจสอบข้อความเสียงให้กดหมายเลขยูสเซอร์และรหัสผ่าน ก็จะสามารถเข้าฟังข้อความเสียงได้
6. ถ้าหากมีการกดหมายเลขยูสเซอร์ หรือ รหัสผิดจะตัดการเชื่อมต่อทันที

ผลการทดสอบในส่วนที่เกี่ยวข้องกับโทรศัพท์

1. สามารถเลือกโมเด็มได้โดยอัตโนมัติ
2. สามารถตรวจจับสัญญาณว่ามีสายเรียกเข้าได้
3. สามารถทำการตรวจจับสัญญาณการกดปุ่มได้ว่ากดปุ่มอะไรเข้ามา
4. สามารถทำการเล่นไฟล์เสียงที่เก็บไว้ผ่านไปยังสายโทรศัพท์ได้
5. สามารถบันทึกเสียงจากโทรศัพท์ได้
6. คุณภาพของเสียงที่ได้มีความชัดเจนปานกลาง มีสัญญาณรบกวนเล็กน้อยเนื่องจากคุณภาพของสายโทรศัพท์
7. สามารถรับสายได้เพียงทีละคู่สายเท่านั้น
8. สามารถนำไฟล์เสียงที่บันทึกจากไคลแอนท์ที่เป็นคอมพิวเตอร์เข้ามาเล่นทางโทรศัพท์ได้
9. สามารถที่จะตัดสายโทรศัพท์ได้เมื่อมีการกดหมายเลขยูสเซอร์หรือรหัสผิดพลาด

บทที่ 5
สรุปและวิจารณ์

5.1 การสรุปและวิจารณ์

จากแอปพลิเคชันที่ได้ทำการออกแบบและพัฒนาขึ้นมานั้น สามารถสรุปผลการพัฒนาได้ออกเป็นสองส่วนคือ

1. **ส่วนที่ใช้ในการติดต่อพูดคุยในเครือข่ายอินเตอร์เน็ต**
   - สามารถให้ผู้ใช้ทำการพูดคุยกันโดยใช้เสียงผ่านทางเซิร์ฟเวอร์
   - สามารถคุยกันได้ทั้งแบบ คนต่อคน หรือแบบ การประชุม
   - เกิดดีเลย์ในการพูดคุยแต่ก็สามารถได้ยินเสียงได้อย่างชัดเจน
   - แอปพลิเคชันสามารถแสดงผลสถานะของการพูดคุยของไคลแอนท์ได้อย่างถูกต้องและครบถ้วน
   - มีฟังก์ชันในให้สามารถกำหนดการทำงานของเซิร์ฟเวอร์ให้เป็นแบบ Forwarding Server หรือ Mixing Server
   - มีฟังก์ชันให้สามารถกำหนด Codec ของเซิร์ฟเวอร์ได้หลากหลายเพื่อให้เลือกใช้ตามความเหมาะสม

2. **ส่วนที่ใช้ในการเชื่อมต่อกับโทรศัพท์** ได้พัฒนาให้เป็นแบบการรับฝากข้อความจากไคลแอนท์ที่เป็นคอมพิวเตอร์หรือโทรศัพท์ โดยจะมีหมายเลขผู้ใช้ในการระบุว่าจะฝากข้อความไปหาใครและสามารถฝากข้อความผ่านทางโทรศัพท์และทางคอมพิวเตอร์ โดยการโทรเข้ามาฟังข้อความนั้น สามารถโทรเข้ามาผ่านทางโทรศัพท์และมีการยืนยันตนเองด้วยกันกด User ID และ Password เพื่อฟังข้อความที่ฝากถึงตนเอง

5.2 ปัญหาที่เกิดขึ้น

1. ระบบที่พัฒนานั้นได้พัฒนาเป็นระบบขนาดเล็กที่รองรับไคลแอนท์ได้จำนวนไม่มากนักเพราะส่วนนี้ใช้ไดเร็กเพลย์และ ไดเร็กเพลย์วอยซ์ ในการพัฒนาซึ่งมี Overhead สูงโดยที่ไม่สามารถเข้าไปจัดการให้ดีขึ้น (optimize) ได้เอง

2. ปัญหาในการสร้างการเชื่อมต่อระหว่างไดเร็กเพลย์และ TAPI หรือการส่งสตรีมเสียงระหว่างคอมพิวเตอร์ไปยังโทรศัพท์ เพราะไม่สามารถเข้าถึงข้อมูลระดับสตรีมซึ่งไดเร็กเพลย์วอยซ์ ห่อหุ้ม (encapsulate) ได้ เพราะฉะนั้น ในการพัฒนาแอปพลิเคชันในการส่งสตรีมระหว่างกันจึงทำผ่านไฟล์

3. ในส่วนของ TAPI นั้นมีปัญหาอยู่ในส่วนของการติดต่อระหว่างคอมพิวเตอร์กับโทรศัพท์ คือ ในระหว่างการสร้างการเชื่อมต่อก่อนที่จะติดต่อกันสำเร็จนั้นจะต้องมีการหาอุปกรณ์มารองรับสตรีมที่ส่งมาทางสายโทรศัพท์ ซึ่งเมื่อก่อนการเชื่อมต่อระหว่างโทรศัพท์กับคอมพิวเตอร์ ต้องมีการกำหนดทิศทางการติดต่อ แต่ว่าหลังจากกำหนดทิศทางลงไปในฟังก์ชันแล้ว ไม่ได้ผล จึงทำให้การคุยกันระหว่าง

คอมพิวเตอร์กับโทรศัพท์ไม่สมบูรณ์นัก คือคุยได้แบบทางเดียวเท่านั้น เพราะฉะนั้นการพัฒนาแอปพลิเคชันจึงใช้การฝากข้อความจากคอมพิวเตอร์ไปยังโทรศัพท์ และจากโทรศัพท์ฝากให้กับโทรศัพท์ ซึ่งสนับสนุนความสามารถของ TAPI ในการพัฒนาการเชื่อมต่อกับโทรศัพท์

4. มีนักพัฒนาโปรแกรมจำนวนน้อยที่เข้ามาทดลองทำงานกับ TAPI จึงมีตัวอย่างและความรู้ที่จะอ้างอิงไปถึงน้อยมาก (มีแต่ใน msdn) เท่านั้น

5.3 แนวทางในการปรับปรุงและพัฒนาต่อไป

ภายในปริญญานิพนธ์เล่มนี้ได้อธิบายโปรแกรมในส่วนสำคัญๆไว้อย่างครบถ้วน ง่ายต่อการศึกษาและทำความเข้าใจเช่น ฟังก์ชันการทำงานของไดเร็กเพลย์และ TAPI ในอนาคตที่มีอินเตอร์เน็ตความเร็วสูงเข้ามาใช้อย่างแพร่หลายนั้นการพูดคุยกันระหว่างคอมพิวเตอร์จะได้รับความนิยมมากขึ้น ทำให้สามารถนำแอปพลิเคชันนี้ไปพัฒนาเพิ่มในส่วนเสริมอื่นๆได้ เช่นมีการสมัครสามาชิกก่อนเพื่อให้มีการเก็บข้อมูลเอาไว้แล้วนำมาทำเป็น Profile ของแต่ละบุคคล หรือการจัดตั้งกลุ่มเฉพาะ หรือ ป้องกันผู้ที่เราไม่ต้องการให้เข้ามาคุยได้ และแอปพลิเคชันของเราพัฒนามาบนรูปแบบของไคลแอนท์ซึ่งเป็นที่นิยมมากในปัจจุบัน การจะเพิ่มเติมหรือแก้ไข ถ้าหากทำบนเซิร์ฟเวอร์จะช่วยลดเวลาในการไปแก้ไขไคลเอ็นท์

และในส่วนของโทรศัพท์นั้นถ้าหากว่าสามารถหาทางให้สตรีมสามารถส่งระหว่าง TAPI และไดเร็กเพลย์ได้ ก็จะสามารถพัฒนาแอปพลิเคชันแบบไคลแอนท์-เซิร์ฟเวอร์อย่างสมบูรณ์ เพื่อให้เซิร์ฟเวอร์เป็นผู้ควบคุมเซสชันของการติดต่อและสื่อสารระหว่างปลายทางทั้งหมดไม่ว่าจะเป็นระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ หรือ ระหว่างคอมพิวเตอร์กับโทรศัพท์ หรือแม้กระทั้งโทรศัพท์กับโทรศัพท์

บรรณานุกรม

[1] Mason McCaskey : "*Special Effects Game Programming with DirectX*", Premier Press

[2] Todd Barron (2001): " *Multiplayer Game Programming*" , Prama Publishing, 2001

[3] Davidson, Jonathan : "*Voice over IP fundamentals*" , Cisco Press , 2000

[4] Goralski, Walter J : "*IP telephony*", McGraw-Hill. 2000

[5] Microsoft Corporation : "*MSDN Library*"

[6] Walter J.Goralski and Matthew C.Kolon : "*IP Telephony*", McGraw – Hill

[7] Daniel Collins : "*Carrier Grade Voice over IP*" , McGraw – Hill

[8] http://msdn.microsoft.com , 2003