ปีการศึกษา 2539

การพัฒนาโปรแกรมสื่อสารด้วยเสียงพูดบนระบบเครือข่าย

โดย

นาย กิจจา	ยวงแก้ว	รหัส 37013281
นาย เกียรติพงษ์	กนกบรรณกร	รหัส 37013282
นาย คทาวุธ	ตรีเทพชาญชัย	รหัส 37013283

อาจารย์ที่ปรึกษา

อาจารย์ ธนา หงษ์สุวรรณ

ปริญญานิพนธ์ปีการศึกษา 2539

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาโปรแกรมสื่อสารด้วยเสียงพูดบนระบบเครือข่าย

ผู้จัดทำ

1.	นาย กิจจา	ยวงแก้ว	รหัส 37013281
2.	นาย เกียรติพงษ์	กนกบรรณกร	รหัส 37013282
3.	นาย คทาวุธ	ตรีเทพชาญชัย	รหัส 37013283

.....อาจารย์ที่ปรึกษา

(อ. ธนา หงษ์สุวรรณ)

การพัฒนาโปรแกรมสื่อสารด้วยเสียงพูดบนระบบเครือข่าย

นาย กิจจา ยวงแก้ว
นาย เกียรติพงษ์ กนกบรรณกร
นาย คทาวุธ ตรีเทพชาญชัย
อ.ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอการพัฒนาโปรแกรมประยุกต์ที่ทำงานบนระบบเครือข่ายที่ใช้โปรโตคอล TCP/IP ซึ่งในปัจจุบันมีผู้ใช้งานอยู่บนระบบนี้อย่างมากมายและ ยังเป็นระบบที่แพร่หลายอย่างยิ่ง โปรแกรมประยุกต์นี้ถูกพัฒนาโดยใช้ ภาษา Visual Basic ทำงานภายใต้ระบบปฏิบัติการ Microsoft Windows 95 รวมทั้งใช้ Winsock ซึ่งเป็น API (Application Program Interface) ในการติดต่อสื่อสารของโปรโตคอล TCP/IP โปรแกรมประยุกต์นี้มีความสามารถที่จะทำการสื่อสารด้วยเสียงพูดระหว่างบุคคล 2 คนผ่านระบบเครือข่าย การพัฒนาโปรแกรมประยุกต์นี้ได้ใช้ภาษา Visual Basic ซึ่งเป็นโปรแกรมเชิงวัตถุ เพื่อเป็นหลักในการพัฒนา และศึกษาในการสร้างโปรแกรมประยุกต์บนระบบเครือข่ายสำหรับผู้ที่สนใจต่อไป

Network Voice Chat application Development

Mr. Kittja Yuangkaew
Mr. Kiatiphong Kanokbanakorn
Mr. Katawut Tritepchanchai

ABSTRACT

This thesis presents about the development of advanced program. That it used on network system . Now a day this system it very popular and there are many people use it. This applied program is developed by using Visual Basic language with Microsoft Windows95 system and Winsock API (Application Programming Interface) to connecting the communication of protocol TCP/IP . This advanced program has the ability in communication between two person with voice by using network system. The evolution of advanced program uses Visual Basic language that is a objective program ot bé the fundamental for everyone who is interested in developing and studying about applied program on network system about the creation of.

สารบัญ

ภาคผนวก

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันนี้ระบบอินเตอร์เน็ตมีบทบาทอย่างมากในการติดต่อสื่อสารจะเห็นได้ว่าโปรแกรมที่ถูกพัฒนาให้สามารถทำงานได้บนระบบอินเตอร์เน็ต มีปริมาณเพิ่มขึ้นอย่างรวดเร็ว เนื่องจากประโยชน์ที่ได้รับมีมากมาย หลายด้าน เช่นด้านการติดต่อสื่อสาร ด้านการเผยแพร่ความรู้ หรือเป็นแหล่งรวบรวมความรู้ในสาขาต่าง ๆ

เดิมที่การติดต่อสื่อสารบนระบบอินเตอร์เน็ต เราใช้ตัวอักษรในการสื่อสารซึ่งมีความยุ่งยากโดยเฉพาะผู้ที่ไม่สามารถพิมพ์สัมผัสได้ จึงได้มีการพัฒนาสร้างโปรแกรมประยุกต์ขึ้นมาโดยใช้พื้นฐานการติดต่อสื่อสารบนระบบอินเตอร์เน็ต โดยโปรแกรมประยุกต์นี้มีความสามารถได้ใช้การติดต่อสื่อสารโดยใช้เสียงซึ่งนับเป็นความสะดวกสบายอีกระดับหนึ่งในการสื่อสาร และด้วยความสามารถของระบบอินเตอร์เน็ต เราสามารถติดต่อสื่อสารจากที่ใด ๆ ก็ได้ที่เชื่อมต่อเข้ากับระบบเครือข่ายอินเตอร์เน็ต

1.2 วัตถุประสงค์

1. สามารถทำการรับส่งข้อมูล (Data/Voice) ในเครือข่ายที่ใช้โปรโตคอล TCP/IP ได้
2. เรียนรู้การบันทึก-เล่น เสียงใน Windows 95 ได้
3. เรียนรู้การบีบอัดและขยายข้อมูล (Compression /Decompression) ของสัญญาณเสียงที่ใช้ใน Windows 95 ได้
4. สามารถเรียนรู้การทำงานของโปรโตคอล TCP/IP และประยุกต์ใช้งานได้

1.3 ขอบเขต

1.โปรแกรมประยุกต์นี้มีความสามารถที่จะทำการสื่อสารด้วยเสียงพูดระหว่างบุคคล 2 คน ซึ่งต่อเครื่องคอมพิวเตอร์ของตนเข้ากับระบบเครือข่าย โดยระบบเครือข่ายนี้ต้องมีการใช้โปรโตคอล TCP/IP และเครื่องคอมพิวเตอร์ทั้งสองต้องมี IP Address ที่แน่นอน

2. สามารถรับเสียงที่ส่งมาจากเครื่องคอมพิวเตอร์ที่เรียกว่าผู้ส่งแสดงออกลำโพงของเครื่องตนได้

3. สามารถที่จะยกเลิกการรับและส่งเสียงได้

4. สามารถที่จะทำการ Compression เสียงได้

1.4 แนวทางในการพัฒนา

โปรแกรมประยุกต์นี้ถูกพัฒนาโดยใช้ภาษา Visual Basic ทำงานภายใต้ระบบปฏิบัติการ Microsoft Windows 95 รวมทั้งได้ใช้ Winsock Controls ซึ่งเป็น Control ในชุดพัฒนา Microsoft Active- X ที่ใช้สำหรับการติดต่อสื่อสารโดยใช้โปรโตคอล TCP/IP โดยชุดพัฒนา Microsoft Active X นี้ทาง Microsoft ได้แจกจ่ายให้กับผู้พัฒนา Application บน อินเตอร์เน็ต ฟรี นอกจากนี้เรายังได้ใช้ API (Application Program Interface) ของ Windows 95 ในส่วนของการติดต่อกับไดร์เวอร์ของการ์ดเสียง เพื่อช่วยในการบันทึกและเล่นกลับเสียง นอกจากนี้ยังได้ใช้ API ในเรื่องของการ Compression / Decompression ของสัญญาณเสียงในรูปแบบต่าง ๆ เช่น GSM610 , PCM , ADPCM ฯลฯ

สรุปแล้วโดยรวมหลังจากที่เราได้เลือกเครื่องไม้เครื่องมือในการทำงานได้เป็นที่เรียบร้อย เราจึงได้วางแผนและขั้นตอนในการพัฒนา ดังนี้

1. ทดลองเขียนโปรแกรมแอพพลิเคชันสำหรับใช้ทดสอบการทำงานติดต่อกับเครือข่ายคอมพิวเตอร์ ซึ่งใช้โปรโตคอล TCP/IP โดยการเรียกใช้ Winsock Control
2. ทำการเขียนแอพพลิเคชันการเข้าควบคุมและสั่งงานการ์ดเสียงเบื้องต้น
3. ทำการเขียนแอพพลิเคชันการเข้าควบคุมและสั่งงานการ์ดเสียงโดยเรียกใช้งาน โดยอาศัยไดร์เวอร์ของตัวการ์ดเสียงเอง
4. ทดลองการเขียนแอพพลิเคชันที่ใช้ความสามารถของ ACM
5. รวมโปรแกรมทั้งหมดเข้าด้วยกัน และทดสอบการทำงาน
6. แก้ไขและดัดแปลงแอพพลิเคชันเป็นครั้งสุดท้าย

บทที่ 2

ความรู้เบื้องต้นเกี่ยวกันระบบเครือข่าย และ การบีบอัดข้อมูลรวมถึงฟังก์ชันต่าง ๆ

2.1 ระบบเครือข่าย และการโปรแกรมบนระบบเครือข่าย

การพัฒนาโปรแกรมประยุกต์บนระบบเครือข่ายจำเป็นต้องมีความรู้พื้นฐานทางด้านระบบเครือข่ายพอสมควรในหัวข้อนี้จะอธิบายกับศัพท์พื้นฐาน และหลักการของระบบเครือข่ายที่จำเป็นสำหรับการทำความเข้าใจในการพัฒนาโปรแกรมประยุกต์บนระบบเครือข่ายดังจะกล่าวต่อไป

2.1.1 ความหมายของระบบเครือข่ายคอมพิวเตอร์ และอินเตอร์เน็ตเวิร์ กกิ้ง(Internetworking)

ระบบเครือข่ายคอมพิวเตอร์ (Computer Network) คือระบบการเชื่อมต่อระหว่างระบบปลายทาง (End-System) ซึ่งระบบปลายทางเป็นระบบที่เป็นอิสระจากกัน (Autonomous) ระบบปลายทางสามารถเป็นได้ ตั้งแต่ไมโครคอมพิวเตอร์ (Microcomputer) ไปจนกระทั่งซุปเปอร์คอมพิวเตอร์ (Supercomputer) ขนาดใหญ่เพื่อจุดมุ่งหมายในการแลกเปลี่ยนข้อมูลและการแบ่งปันทรัพยากรของระบบเช่น ไฟล์ (File) ข้อมูล, เครื่องพิมพ์ (Printer) ,โมเด็ม (Modem) ตลอดจนการให้บริการฐานข้อมูลร่วม (Sharing database) อินเตอร์เน็ตเวิร์คกิ้ง หรืออินเตอร์เน็ต (Internet) คือการเชื่อมต่อของระบบเครือข่าย 2 เครือข่ายขึ้นไป ดังนั้น คอมพิวเตอร์บนระบบเครือข่ายหนึ่งก็สามารถติดต่อกับคอมพิว -เตอร์บนระบบเครือข่ายอื่นๆได้ เช่น เน็ตเวิร์คโทเคนริงค์ (Tokenring network) เชื่อมกับเน็ตเวิร์คอี-เทอร์เน็ท (Ethernet network) โดยมีเกตเวย์ (Gateway) เป็นตัวเชื่อม

2.1.2 รูปแบบของโปรแกรมบนระบบเครือข่าย (Network Programming Models)

จากหัวข้อ 2.1.1 ได้ให้ความหมายของระบบเครือข่าย แสดงถึงวิธีการที่ระบบคอมพิวเตอร์ใดๆ จะทำการเชื่อมต่อกันระบบเครือข่าย แต่ว่าคอมพิวเตอร์ที่เชื่อมต่ออยู่กับระบบเครือข่ายนั้นมีวิธีการอย่างไรในการแลกเปลี่ยน ข้อมูลและแบ่งปันทรัพยากรเหล่านั้นได้ทำให้ต้องมีโปรแกรมประยุกต์ ซึ่งสามารถที่จะจัดการในสิ่งที่กล่าวมาแล้ว อย่างเหมาะสม ซึ่งในหัวข้อนี้จะกล่าวถึงรูปแบบของโปรแกรมบนระบบเครือข่าย 2 รูปแบบคือ ไคลเอนต์เซิร์ฟเวอร์ (Client/Server Computing) และการประมวลผลแบบกระจาย (Distributed Computing)

2.1.2.1 การประมวลผลแบบไคลเอนต์เซิร์ฟเวอร์ (Client/Server Computing)

การประมวลผลแบบไคลเอนต์เซิร์ฟเวอร์นี้การประมวลผลของโปรแกรมประยุ กต์จะแบ่งออกเป็น 2 ส่วนคือ ส่วนฟรอนต์เอนด์ (front-end) ที่ทำงานบนไคลเอนต์ ส่วนนี้จะทำหน้าที่แสดงผลที่ได้จากการประมวลผล และรับข้อมูลจากผู้ใช้ อีกส่วนหนึ่งคือแบ็คเอนด์ (back-end) ทำงานบนเซิร์ฟเวอร์ มีหน้าที่ในการเก็บรวบรวม และจัดการข้อมูลจากฟรอนต์เอนด์ในรูปแบบการประมวลผลไคลเอนต์ - เซิร์ฟเวอร์นี้ เครื่องเซิร์ฟเวอร์นี้มักจะเป็นเครื่องที่มีความสามารถสูงกว่าเครื่องไคลเอนต์ โดยปกติเครื่องเซิร์ฟเวอร์มักจะเป็นเครื่องเมนเฟรม หรือมินิคอมพิวเตอร์และเครื่องไคลเอนต์มักจะเป็นเครื่องคอมพิวเตอร์ส่วนบุคคล ซึ่งการติดต่อกันระหว่างส่วนฟรอนต์เอนด์และแบ็คเอนด์ ทำโดยผ่านระบบเครือข่ายซึ่งในทางปฏิบัติแล้ว ในส่วนของแบ็คเอนด์ที่อยู่บนเซิร์ฟเวอร์ จะเป็นฝ่ายให้บริการแก่งานของไคลเอนต์หลายงานในเวลาเดียวกัน

2.1.2.2 การประมวลผลแบบกระจาย (Distributed Computing)

การประมวลผลของโปรแกรมประยุกต์มีรูปแบบการประมวล 2 แบบ คือ พรีคอลเล็คชั่น (precollection) และการประมวลผลแบบขนาน (parallel processing) ดังนี้

1. พรีคอลเล็คชั่นเป็นลักษณะการทำงานที่ข้อมูลที่ต้องการในการจัดเก็บ และส่งต่อไปทั่วระบบเครือข่ายอยู่ตลอดเวลาอย่างสม่ำเสมอ การทำงานในลักษณะนี้จะเหมาะสมกับงานบางงาน เช่น ต้องการเก็บสถานะของคอมพิวเตอร์ที่อยู่ในระบบเครือข่ายหนึ่งๆ ทุกเครื่อง

2. การประมวลผลแบบขนาน การประมวลผลลักษณะนี้งานใดๆ จะถูกประมวลผลด้วยคอมพิวเตอร์หลายๆ เครื่องโดยเครื่องคอมพิวเตอร์เหล่านั้นสามารถจะติดต่อกันโดยระบบเครือข่าย เช่น การทำการพัฒนาโปรแกรมประยุกต์ขนาดใหญ่ โดยทีมพัฒนาที่มีผู้พัฒนาหลายคน สามารถลดเวลาของการแปล และการรวมโมดูล (module) ต่างๆ เข้าเป็นโปรแกรมเดียวกัน โดยการแบ่งงานการแปลโมดูลเหล่านั้นแก่คอมพิวเตอร์ในระบบเครือข่ายทำการแปลในเวลาเดียวกัน

2.1.3 โมเดล OSI

OSI เป็นคำย่อที่มาจากคำว่า Open System Interconnection โดยที่เป็นมาตรฐานที่ถูกเสนอขึ้นโดย International Standards Organization ซึ่งเป็นองค์กรที่จัดตั้งขึ้นมาเพื่อดูแล และส่งเสริมตลอดจนกำหนดมาตรฐานของการติดต่อสื่อสารของระบบเครือข่ายคอมพิวเตอร์ โดยโมเดล OSI นี้มีลักษณะเป็นสถาปัตยกรรม แบบระบบเปิด (Open System) เพราะมุ่งที่จะให้ระบบคอมพิวเตอร์ในหลายๆ รูปแบบที่แตกต่างกันสามารถเชื่อมต่อกันได้ OSI โมเดลได้แบ่งโปรโตคอล (protocol) ในการสื่อสารออกเป็น 7 เลเยอร์ (layer) ซึ่งโปรโตคอล คือชุดของกฎ หรือข้อตกลงในการติดต่อ ข้อสังเกตโมเดล OSI

เป็นเพียงข้อเสนอแนะ มิใช่ข้อกำหนด และควรรู้ว่ายังไม่มีระบบการเชื่อมต่อใดที่สร้างเหมือนกับโมเดล OSI จริงๆ

7	Application Layer
6.	Presentation Layer
5.	Session Layer
4.	Transport Layer
3.	Network Layer
2.	Datalink Layer
1.	Physical Layer

ภาพที่ 2.2 OSI โมเดลทั้ง 7 เลเยอร์

ในหนึ่งชั้นของเลเยอร์ไม่ได้กำหนดว่าจะต้องมีเพียงหนึ่งโปรโตคอลเท่านั้น ที่อยู่ในระดับเลเยอร์เดียวกัน และในทางตรงข้าม ชุดของโปรโตคอลใดๆ อาจจะมีมากกว่าหนึ่งเลเยอร์ประกอบกับเป็นข้อกำหนดของ ระบบเครือข่ายเรียกว่าชุดโปรโตคอล (Protocol Suite) เช่น ชุดโปรโตคอล TCP/IP (Transmission Control/Internet Protocol) เป็นต้น ประโยชน์ในการแบ่งเป็นเลเยอร์ คือกำหนดการติดต่อระหว่างเลเยอร์ ทำได้โดยไม่ต้องคำนึงถึงการเปลี่ยนในเลเยอร์ใดๆ ที่ติดกัน

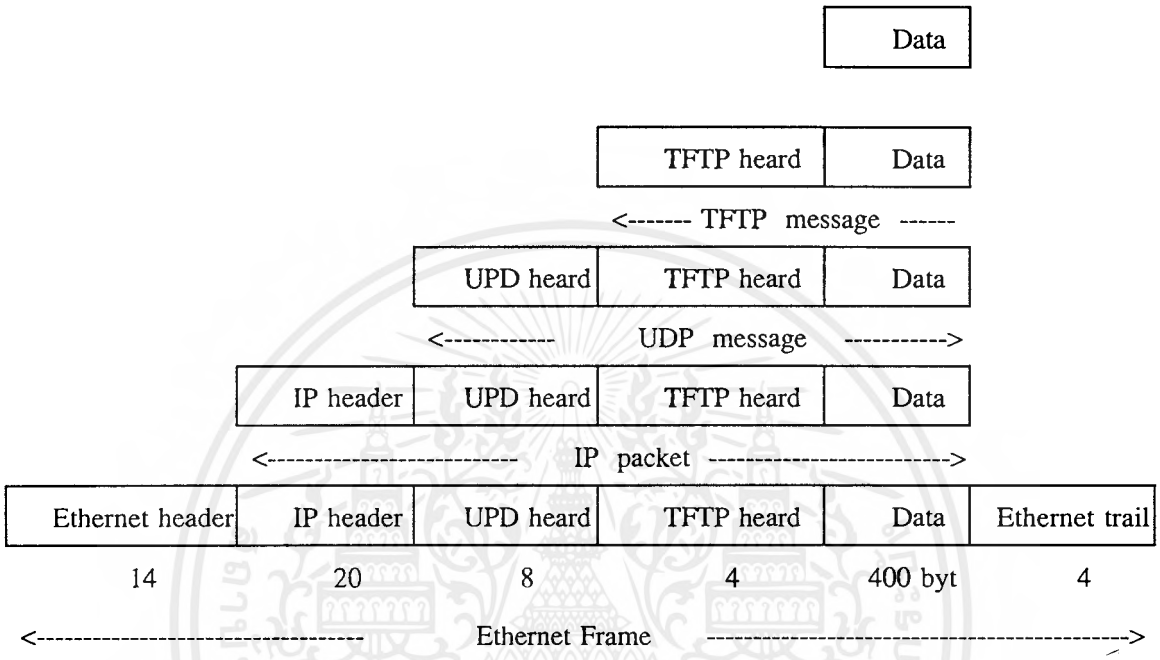
2.1.4 เอ็นแคปซูเลชั่น (Encapsulation)

พิจารณาโปรแกรมประยุกต์ TFTP (File Transfer Protocol) ซึ่งใช้ในโปรโตคอล UDP (User Datagram Protocol) ระหว่างสองระบบซึ่งเชื่อมต่อด้วยอีเทอร์เน็ท ถ้าโปรแกรมไคลเอนต์ TFTPมีข้อมูล 400 byte ต้องการส่งไปที่โปรแกรมเซิร์ฟเวอร์ โปรแกรมไคลเอนต์ TFTP จะเพิ่มข่าวสารควบคุม 4 byte เป็นส่วนหัวของข้อมูลก่อนที่จะผ่านข้อมูลไปสู่เลเยอร์ UDP การเพิ่มของข่าวสารควบคุมไปที่ข้อมูลเรียกว่า เอ็นแคปซูเลชั่น ดังแสดงในภาพ2.3

TFTP heard	Data
4 byte	400 byte

ภาพที่ 2.3 การเอ็นแคปซูเลชั่น

เลเยอร์ UDP จะไม่มีการตีความส่วนหัว TFTP 4 byte งานของเลเยอร์ UDP คือส่งข้อมูล 404 byte ไปสู่เลเยอร์ UDP ของโปรแกรมอีกด้านหนึ่ง จากนั้นเลเยอร์ UDP จะทำการเพิ่มส่วนหัว 8 byte แล้วส่งข้อมูล 432 byte ไปยังเลเยอร์ดาต้าลิ้งค์ ที่เลเยอร์นี้จะมีการเพิ่มส่วนหัวอีก 14 byte และส่วนหางอีก 4 byte ดังภาพ 2.4



2.1.5 ลักษณะของการติดต่อ

แบ่งออกเป็น 2 ชนิด คือ

2.1.5.1 Connection-oriented

คือการติดต่อที่ต้องมีการเชื่อมโปรเซสที่จะทำการติดต่อก่อนที่จะมีการส่งหรือรับข้อมูลซึ่งสามารถใช้คำว่าวงจรเสมือน (Virtual Circuit) เพราะว่าจะทำงานเสมือนมีวงจรต่ออยู่ระหว่าโปรเซส ถึงแม้ว่าข้อมูลนี้อาจจะผ่าน Packet-Switching Network บริการชนิดนี้ส่วนมากจะใช้ในกรณีที่มีข่าวสารต้องการมากกว่าหนึ่งข่าวสาร ดังนั้นสามารถแบ่งขั้นการทำงานออกเป็น

- ขั้นการสร้างการติดต่อ (conecttion establishment)
- ขั้นการส่งผ่านข้อมูล (data transfer)
- ขั้นยกเลิกการติดต่อ (connection termination)

2.1.5.2 Connectionless หรือดาต้าแกรม (Datagram)

คือจะไม่มีขั้นการสร้างการติดต่อ และขั้นการยกเลิกการติดต่อ แต่จะมีขั้นการส่งผ่านข้อมูลอย่างเดียว โดยข้อมูลเรียกว่าดาต้าแกรมจะถูกส่งจากระบบหนึ่งไปสู่ระบบหนึ่งอย่างเป็นอิสระโดยไม่ขึ้นอยู่กับดาต้าแกรมอื่น

2.1.6 แอดเดรส (Address)

การที่ระบบในระบบเครือข่ายสามารถติดต่อกันได้จำเป็นต้องมีแอดเดรสไว้คล้ายกับหมายเลขประจำตัวซึ่งลำดับของแอดเดรสสามารถพิจารณาได้คือ

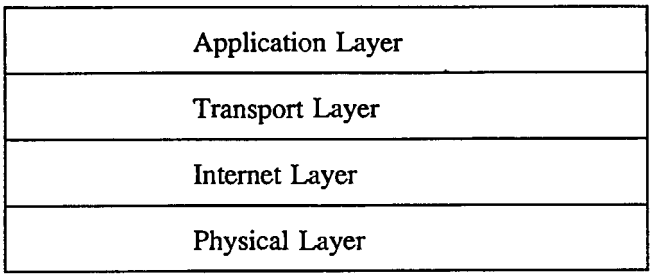
- แต่ละเครือข่ายจะต้องมีแอดเดรสสำหรับเครือข่าย
- คอมพิวเตอร์โฮสต์แต่ละเครื่องในเครือข่ายจะต้องมีแอดเดรส
- แต่ละโปรเซสในโฮสต์จะต้องมีหมายเลขประจำตัว

โดยทั่วไปแอดเดรสของโฮสต์จะประกอบด้วยหมายเลขเครือข่าย (Network ID) และหมายเลขของโฮสต์ (Host ID) ส่วนแอดเดรสของโปรเซสของผู้ใช้จะอยู่ในรูปจำนวนเต็มซึ่งกำหนดโดยโปรโตคอล เช่น โปรโตคอล TCP/IP จะใช้เลขจำนวนเต็มขนาด 32 บิต ในการกำหนดหมายเลขเครือข่ายและหมายเลขของโฮสต์ และทั้ง TCP และ UDP ใช้เลขจำนวนเต็มขนาด 16 บิต เป็นหมายเลขพอร์ตหรือหมายเลขของโปรเซส ชุดของโปรโตคอลส่วนใหญ่จะมีการกำหนดชุดของแอดเดรสสำหรับการบริการที่เป็นที่รู้กัน โดยทั่วกัน เช่นคอมพิวเตอร์ที่โปรโตคอล TCP/IP ส่วนใหญ่จะมี FTP (File Transfer Protocol) ซึ่งไคลเอนต์สามารถติดต่อได้โดยใช้หมายเลขพอร์ตคือ 21

2.2 ความรู้เบื้องต้นเกี่ยวกับโปรโตคอล TCP/IP

2.2.1 เปรียบเทียบระหว่างโปรโตคอล TCP/ IP และ OSI โมเดล

การออกแบบโปรโตคอล TCP/IP นั้นไม่ได้เป็นไปตามรูปแบบของ OSI โมเดล เนื่องด้วยถูกออกแบบโดยองค์กรขนาดใหญ่ซึ่งใช้เวลานานในการออกแบบตลอดจนการรับรองมาตรฐานต่างกับโปรโตคอล TCP/IP ที่ถูกแบบด้วยความต้องการอันเร่งด่วนของรัฐบาลสหรัฐ จึงทำให้การพัฒนาโปรโตคอล TCP/IP มีเงื่อนไขของในด้านความต้องการที่ต่างจาก OSI โมเดล ซึ่งหากเรามองโดยรวมแล้วจะเห็นว่าโปรโตคอล TCP/IP มีการแบ่งเป็นเลเยอร์ที่น้อยกว่า OSI โมเดล คือมี 4 ชั้นเท่านั้น ดังภาพ 2.5 โดยแบ่งเป็น



ภาพที่ 2.5 เลเยอร์ของโปรโตคอล TCP/IP

2.2.1.1 แอพพลิเคชั่นเลเยอร์ (Applicatoin Layer)

ในเลเยอร์นี้ประกอบโปรแกรมประยุกต์ที่ใช้เครือข่าย เช่น โปรแกรมส่งถ่ายข้อมูล (file-transfer protocol) และอาจกล่าวได้ว่าเลเยอร์นี้โปรโตคอล TCP/IP ก็คือ เลเยอร์ในชั้นแอพพลิเคชั่นเลเยอร์ รวมกับชั้นพรีเซนเตชั่นเลเยอร์ (Presentation Layer) ใน OSI โมเดล นั่นเอง และในเลเยอร์ชั้นนี้ของโปรโตคอล TCP/IP จะกลืนอยู่ในตัวโปรแกรมประยุกต์

2.2.1.2 ทรานสปอร์ตเลเยอร์ (Transport Layer)

ในชั้นนี้เป็นชั้นที่ให้การส่งข้อมูลจากจุดปลายถึงจุดปลาย หากเปรียบเทียบกับ OSI โมเดล ก็สามารถเทียบได้กับชั้นเซสชั่นเลเยอร์ (Session Layer) รวมกับทรานสปอร์ตเลเยอร์นั่นเอง โดยโปรโตคอล TCP/IP มีซอกเก็ต (Socket) เป็นจุดปลาย (end-point) ในการสื่อสาร ซึ่งซอกเก็ตนี้ประกอบไปด้วยหมายเลขของคอมพิวเตอร์ และหมายเลขพอร์ต (port) ของเครื่องที่ต้องการส่งข้อมูลไปถึง ในชั้นนี้มีการรับรองการถึงที่หมาย และลำดับของข้อมูลที่ส่งโดยไม่ซ้ำ และความผิดพลาดข้อมูล

2.2.1.3 อินเตอร์เน็ตเลเยอร์ (Internet Layer)

เลเยอร์นี้มีการกำหนดดาต้าแกรม และทำการหาเส้นทางการส่ง หน้าที่ของเลเยอร์นี้เทียบเท่ากับเน็ตเวอร์คเลเยอร์ (Network Layer) และดาต้าลิงค์เลเยอร์ (Datalink Layer) ของ OSI โมเดล

2.2.1.4 ฟิสิคอลเลเยอร์ (Physical Layer)

โปรโตคอล TCP/IP ไม่ได้กำหนดรูปแบบของการเชื่อมต่อในระดับนี้ไว้ใหม่ แต่ได้ใช้มาตรฐานที่มีอยู่เดิมที่กำหนดไว้ก่อน เช่น RS232, อีเทอร์เน็ต (Ethernet) เป็นต้น

2.2.2 รูปแบบการกำหนดแอดเดรสของโปรโตคอล TCP/IP

ในหัวข้อนี้จะได้อธิบายรูปแบบการกำหนดแอดเดรสของโปรโตคอล TCP/IP ซึ่งลักษณะแอดเดรสของโปรโตคอลนี้ค่าของแอดเดรสของเครื่องคอมพิวเตอร์ในระบบเครือข่ายจะไม่ซ้ำกันเลย โดยเลขนี้เรียกว่า IP แอดเดรส (IP Address) เป็นเลข 32 บิต ซึ่งแบ่งเป็นคลาส (Class) ตามหลักในการพิจารณาที่จะได้กล่าวต่อไปนี้

2.2.2.1 การแบ่งเน็ตเวอร์คคลาส (Network Class)

เนื่องจากหมายเลขแอดเดรสของคอมพิวเตอร์เครื่องใดๆ นั้นจะต้องสามารถบอกถึงความแตกต่างระหว่างตัวเครื่องเองตลอดจนเครือข่ายที่คอมพิวเตอร์นั้นเชื่อมต่ออยู่ด้วยหมายเลข IP แอดเดรส จึงแยกออกเป็น 2 ส่วน ได้แก่ ส่วนที่แสดงหมายเลขของเครื่องคอมพิวเตอร์โฮสต์ และส่วนที่เป็นหมายเลขของเครือข่าย

การแบ่งคลาสของแอดเดรสทำได้โดยการพิจารณาจนวนบิตของ 2 ส่วนประกอบข้างต้น ซึ่งมีการแบ่งออกเป็น 5 คลาส แต่มีการใช้เพียง 3 คลาสแรก คือ คลาส A, คลาส B และ คลาส C ส่วนคลาส D และ E ถูกสงวนไว้สำหรับจุดประสงค์พิเศษ

31

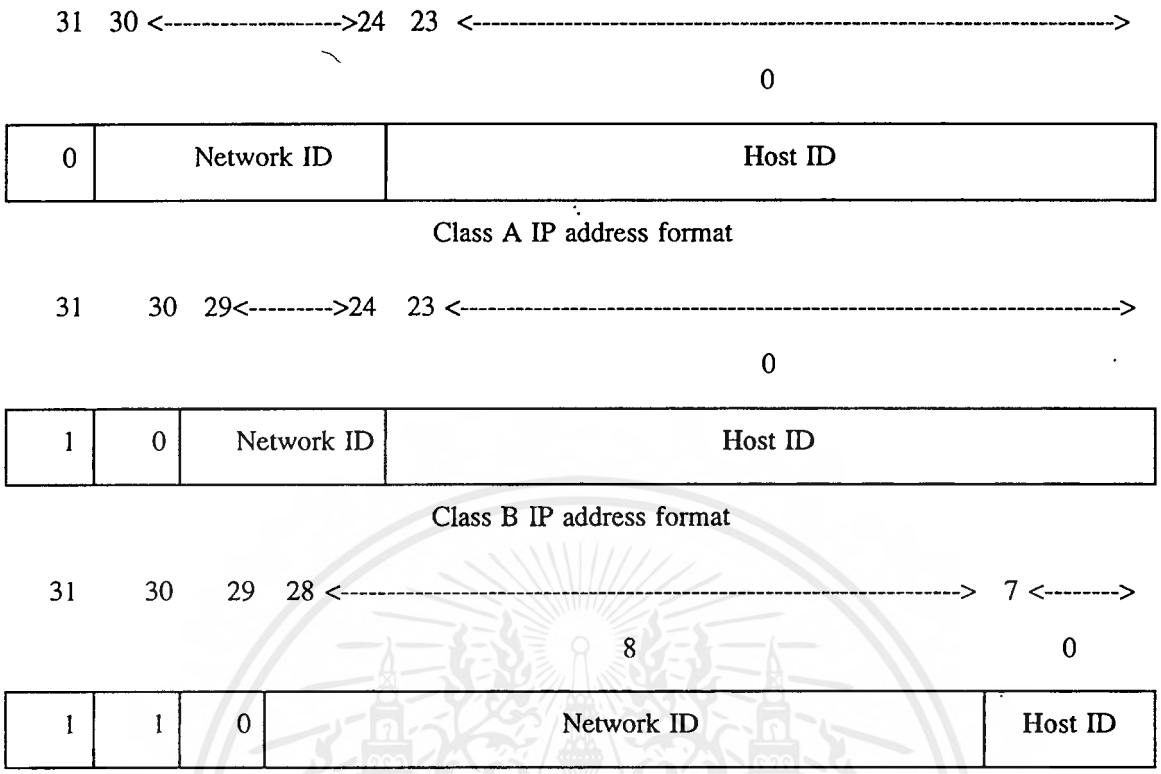
Class ID	Network ID	Host ID

ภาพที่ 2.6 หมายเลข IP แอดเดรส

Network Class	Networks	Hosts per Network
A	126	16,777,214
B	16,382	65,534
C	2,097,150	254

ภาพที่ 2.7 แสดงคลาส,จำนวนเครือข่าย และจำนวนโฮสต์ของแต่ละคลาส

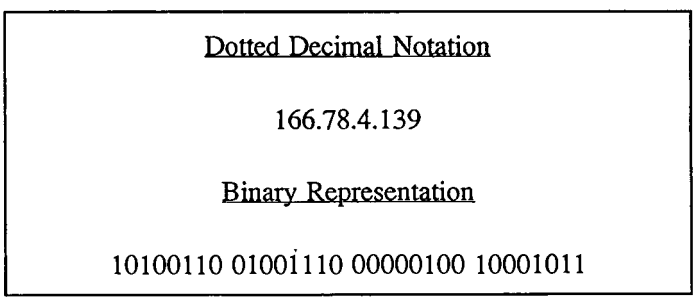
โดยปกติแล้วผู้พัฒนาโปรแกรม ไม่ต้องสนใจความแตกต่างระหว่างคลาสของ IP แอดเดรส



ภาพที่ 2.8 แสดงรูปแบบของ IP แอดเดรสในคลาสต่างๆ

2.2.2.2 การแทนด้วยเลขฐานสิบและจุด (Dotted Decimal Notation)

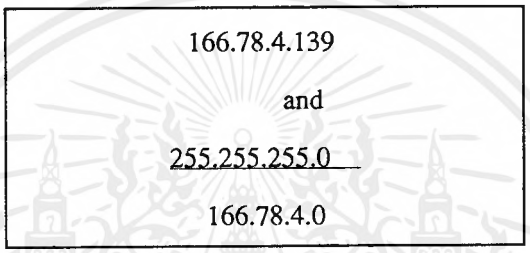
เนื่องจากการแทนหมายเลข IP แอดเดรสเป็นเลขฐาน 2 ซึ่งข้างอ่านไม่สะดวกจึงมีการแทนเลขฐาน 2 เหล่านั้นในรูปเลขฐานสิบและจุด โดยเลขฐานสิบแต่ละตัวจะแทนเลขฐาน 2 จำนวน 8 บิต โดยระหว่างเลขฐานสิบแต่ละตัวจะแทรกด้วยจุด ดังนั้นจะต้องใช้เลขฐานสิบ 4 ตัว ในการแทนเลข 32 บิต ที่เป็น IP แอดเดรส ดังตัวอย่างตามภาพ 2.10 จะสังเกตว่าหมายเลข IP แอดเดรสนี้จัดอยู่ในคลาส B โดยหมายเลขของเครือข่ายคือ 166.78 และหมายเลขประจำเครื่องคือ 4.139



ภาพที่ 2.9 ตัวอย่างหมายเลข IP แอดเดรสทั้งสองแบบ

2.2.2.3 การทำซับเน็ตติ้ง (Subnetting)

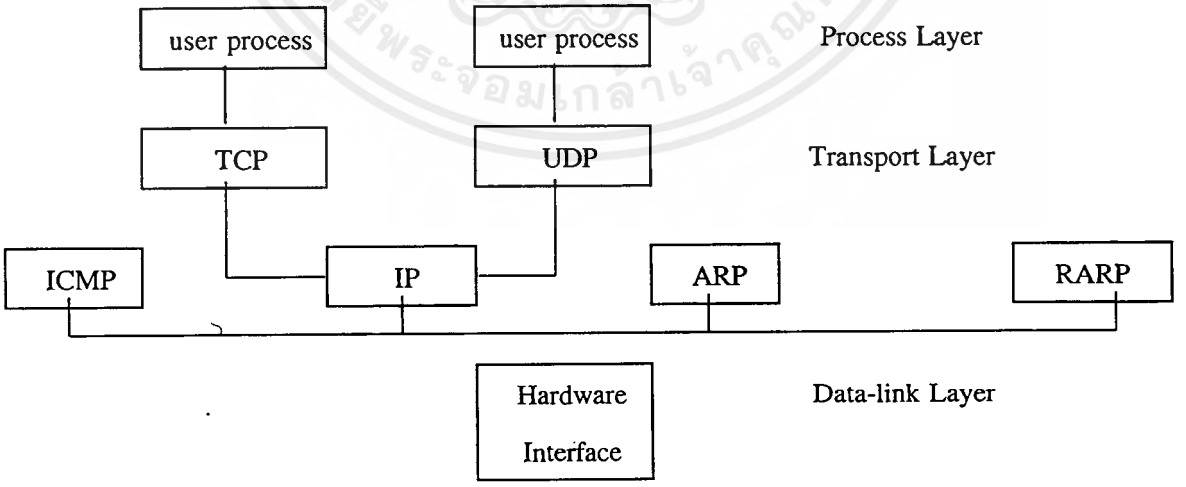
การทำซับเน็ตติ้งเป็นการเปลี่ยนแปลงการใช้หมายเลขของเครื่องโฮสต์และหมายเลขของเครือข่ายในระดับท้องถิ่น โดยในทางตรรกคือ การเลื่อนเส้นแบ่งที่แยกหมายเลขเครื่อง และหมายเลขของเน็ตเวอร์คที่อยู่ในหมายเลข IP แอดเดรส โดยที่ปริมาณของหมายเลขเครื่องโฮสต์ และหมายเลขเครือข่ายจะแปรผกผันกัน เช่น หากมีปริมาณของเน็ตเวอร์คมาก ก็จะทำให้เครื่องใดๆ ที่จะต่อกับระบบเครือ-ข่ายหนึ่ง ๆ จะน้อยลงเป็นต้น ในทางปฏิบัติการทำซับเน็ตติ้งทำโดยการนำซับเน็ตมาร์ค (Subnet mark) คือตัวเลขจำนวน 32 บิต มาทำการ กระทำตรรกและ (AND) กับหมายเลข IP แอดเดรส ดังตัวอย่างโดยกำหนดหมายเลข IP แอดเดรส คือ 166.78.4.139 และซับเน็ตมาร์ค คือ 255.255.255.0 ทำการกระทำ ตรรกและ ดังภาพ



ภาพที่ 2.10 การทำซับเน็ตติ้ง

2.2.3 ชุดโปรโตคอล TCP/IP (TCP/IP Protocol Suite)

ชุดโปรโตคอล TCP/IP นอกจากมีโปรโตคอล TCP และ IP แล้วยังมีโปรโตคอลอย่างอื่นอีก ดังภาพที่ 2.12 แสดงความสัมพันธ์ของชุดโปรโตคอลโดยแบ่งตามเลเยอร์



ภาพที่ 2.11 ความสัมพันธ์ระหว่างชุดโปรโตคอล

2.2.3.1 โปรโตคอล IP (Internet Protocol)

โปรโตคอล IP เป็นโปรโตคอลแบบคอนเนคชั่นเลส (Connectionless Protocol) ซึ่งได้กล่าวถึงลักษณะของโปรโตคอล ชนิดนี้ไปแล้วในหัวข้อ 2.1.5 โดยที่โปรโตคอล IP ไม่รับประกันว่าข้อมูลที่ส่งจะไปถึงปลายทางซึ่งแพคเก็ต (Packet) ของข้อมูลอาจไปถึงในลักษณะที่ผิดลำดับ, ซ้ำกัน หรือไม่ไปถึงเลย โดยความน่าเชื่อถือของการส่งจะถูกควบคุมในโปรโตคอลในเลเยอร์ต่างๆ ไป การหาเส้นทางของข้อมูลจะทำในระดับของโปรโตคอล IP นี้ โดยพิจารณาแต่ละแพคเก็ต แยกออกจากกัน และยังมีหน้าที่ในการจัดเรียงข้อมูลใหม่ที่ปลายทาง อีกด้วย

2.2.3.2 โปรโตคอล ARP (Address Resolution Protocol)

โปรโตคอลนี้ทำหน้าที่จับคู่ระหว่างหมายเลข IP แอดเดรสเข้ากับหมายเลขแอดเดรสทางฮาร์ด-แวร์ โดยโปรโตคอลนี้ทำการส่งข้อความไปทั่วเครือข่ายท้องถิ่น ซึ่งข้อความนี้เป็นลักษณะข้อความที่ตรวจสอบว่ามีคอมพิวเตอร์ที่มีหมายเลข IP แอดเดรสตรงกับที่ต้องการหาหรือไม่ หากคอมพิวเตอร์ที่มีหมายเลข IP แอดเดรสตรงกันนั้นได้รับข้อความนี้ก็จะตอบกลับและเป็นที่หน้าสังเกตว่าโปรโตคอลนี้ทำงานได้กับระบบเครือข่ายท้องถิ่นเท่านั้น เพราะว่าโครงสร้างหมายเลขทางฮาร์ดแวร์ของเครื่องคอมพิวเตอร์จะขึ้นอยู่กับชนิดของเครือข่ายด้วย

2.2.3.3 โปรโตคอล ICMP (Internet Control Message Protocol)

เป็นโปรโตคอลที่จัดการเกี่ยวกับข่าวสารความผิดพลาดและการควบคุมเกตเวย์ และเครื่องคอมพิวเตอร์ในระบบเครือข่าย

2.2.3.4 โปรโตคอล RARP (Reverse Address Resolution Protocol)

เป็นโปรโตคอลที่ทำหน้าที่จับคู่ระหว่างหมายเลขของฮาร์ดแวร์กับหมายเลข IP แอดเดรส หรือทำงานกลับกันกับโปรโตคอล ARP

2.2.3.5 โปรโตคอล UDP (User Datagram Protocol)

โปรโตคอลนี้เป็นโปรโตคอลที่อยู่ในระดับทรานสปอร์ตเลเยอร์ และมีความสำคัญเพราะว่าเป็นโปรโตคอลที่ผู้พัฒนาโปรแกรมสามารถใช้ได้โดยตรงโปรโตคอลนี้เป็นโปรโตคอลแบบคอน-เนคชั่นเลสมีความน่าเชื่อถือต่ำ ไม่รับรองว่าข้อมูลที่ส่งไปจะไปถึงปลายทางหรือไม่ และอาจซ้ำซ้อนหรือผิดลำดับได้ แต่ข้อดีของโปรโตคอลนี้ คือค่าความสิ้นเปลือง (overhead) ที่ต่ำ

2.2.3.6 โปรโตคอล TCP (Transmission Control Protocol)

โปรโตคอลนี้อยู่ในระดับทรานสปอร์ตเลเยอร์ เหมือนกับโปรโตคอล UDP แต่มีลักษณะที่ตรงข้ามกันคือ เป็นโปรโตคอลแบบคอนเนคชั่นออเรียนเต็ด โดยจะมีความน่าเชื่อถือในการรับ ส่งข้อมูล และลำดับของข้อมูลจะมีลำดับเหมือนกับต้นทางและเนื้อข้อมูลไม่ผิดพลาด จึงทำให้เกิดความสิ้นเปลืองในการเชื่อมต่อของการส่งข้อมูลมากกว่าโปรโตคอล UDP

2.2.4 หมายเลขพอร์ต

เนื่องจากในเวลาใดๆ สามารถมีโพรเซสของผู้ใช้สามารถใช้ UDP หรือ TCP ได้พร้อมกัน ดังนั้นจึงต้องมีวิธีแยกแยะว่าข้อมูลเป็นของผู้ใช้คนใด ซึ่งวิธีที่ TCP และ UDP ใช้คือการใช้หมายเลขพอร์ต (Port Number) เมื่อโพรเซสไคลเอนต์ (client process) ต้องการที่จะติดต่อกับเซิร์ฟเวอร์ไคล - เอนต์จะต้องเจาะจง เซิร์ฟเวอร์ที่ต้องการติดต่อ แต่ลำพังรู้แอดเดรสอินเตอร์เน็ต 32 บิตเพียงอย่างเดียวนั้นไม่เพียงพอ เพราะว่าสามารถติดต่อกับโฮสต์ได้เพียงอย่างเดียวแต่ไม่สามารถเจาะจงโพรเซสที่จะทำการติดต่อได้ ดังนั้นเพื่อแก้ปัญหานี้ทั้ง TCP และ UDP ได้มีการกำหนด หมายเลขพอร์ตมาตรฐาน (well-known ports) ซึ่งเป็นที่รู้จักกันเช่น ทุกๆ ระบบ TCP/IP ที่มีเซิร์ฟเวอร์ FTP (File Transfer Protocol) จะมีหมายเลขพอร์ตเป็น 21 เป็นต้น เมื่อ TCP หรือ UDP กำหนดหมายเลขพอร์ตที่ไม่ซ้ำกันให้โพรเซสของผู้ใช้ เราเรียกหมายเลข พอร์ตนี้ว่าหมายเลขพอร์ตชั่วคราว (Ephemeral Port Numbers) เมื่อไคลเอนต์เลิกใช้หมายเลขพอร์ตนี้แล้ว สามารถกำหนดหมายเลขพอร์ตนี้ให้ไคลเอนต์อื่นได้ โพร-เซสที่ได้รับหมายเลขพอร์ต ชั่วคราวนี้จะไม่สนใจว่ามีค่าเท่าใด แต่เป็นหน้าที่ของอีกโพรเซสหนึ่งที่ต่อกันที่ต้องสนใจ เพราะต้องส่งข้อมูลกลับมาที่พอร์ตนี้ ใน TCP และ UDP นั้นหมายเลขพอร์ตตั้งแต่ 1-1023 เป็นพอร์ตที่สงวนไว้สำหรับหมายเลขพอร์ตมาตรฐาน

2.3 Winsock (Window Socket Application Programming Interface)

Winsock คือโปรแกรมส่วนขยายที่ทำหน้าที่ฟังก์ชันด้านเน็ตเวอร์คที่มีการใช้โปรโตคอล TCP/IP โดยสำหรับในระบบของยูนิกซ์ในเริ่มแรกซึ่งยังไม่สนับสนุนโปรโตคอล TCP/IP ดังนั้นในการที่จะทำให้ยูนิกซ์สนับสนุนโปรโตคอล TCP/IP โดยไม่จำเป็นต้องดัดแปลงระบบปฏิบัติการจึงมีการเขียนโปรแกรมส่วนขยายขึ้นมาสำหรับรองรับโปรโตคอล TCP/IP ซึ่งเราจะรู้จักกันดีในชื่อของ ***Berkeley Sockets*** สำหรับบน Microsoft Windows เพื่อที่จะให้สนับสนุนโปรโตคอล TCP/IP จึงได้มีการพัฒนาโปรแกรมส่วนที่จะทำหน้าที่ โปรโตคอล TCP/IP นี้ซึ่งเราจะเรียกว่า Windows Socket นั่นเอง สำหรับในหัวข้อนี้จะได้กล่าวถึง Winsock ที่เป็นไลบรารี (Library) ของฟังก์ชั่นที่ทำหน้าที่เกี่ยวกับการสื่อสาร โดยติดต่อกับซอคเก็ต Winsock ที่มีส่วนขยายมาจาก Berkeley Socket โดยได้เพิ่มส่วนที่เป็นแมสเสจไดรเวน (Message -driven) เพื่อให้ทำงานได้ดีในสภาวะแวดล้อมของวินโดวส์ และ

สามารถจัดการระบบเครือข่ายแบบ TCP/IP ได้ ก่อนที่จะกล่าวถึง Winsock จะได้กล่าวถึง Berkeley Socket เพื่อเป็นพื้นฐานความเข้าใจ Winsock ในอันดับต่อไป

2.3.1 Berkeley Sockets

ซอคเก็ต (Socket) เป็นแอพพลิเคชั่นโปรแกรมอินเตอร์เฟส (Application Program Interface) ในที่นี้จะขอเรียกโดยย่อว่า API โดยที่ API นี้จะเป็นส่วนที่ติดต่อระหว่างทรานสปอร์ตเลเยอร์กับโปรแกรมประยุกต์ ทำให้ผู้พัฒนาโปรแกรมสามารถที่จะเรียกใช้บริการต่างๆ ของทรานสปอร์ตเลเยอร์ได้โดยการใช้ซอคเก็ตนั่นเอง Berkeley Socket เป็นซอคเก็ตที่ใช้กับทรานสปอร์ตเลเยอร์ของโปรโตคอล TCP/IP โดยที่ Berkeley Socket มีความเป็นมาพร้อมๆ กับ อินเตอร์เน็ต

2.3.2 เปรียบเทียบ Berkeley Socket และ Winsock

ซอคเก็ตทั้งสองมีความแตกต่างกัน โดยแสดงเป็นข้อๆ ได้ดังนี้

1.Winsock รุ่น 1.1 สนับสนุนการทำงานกับกลุ่มโดเมน (Domain) TCP/IP เท่านั้น ส่วนซอคเก็ตของยูนิกซ์ (UNIX) นั้นสามารถสนับสนุนการทำงานกับกลุ่มโดเมนของโดเมนยูนิกซ์และกลุ่มโดเมน Xerox XNS ด้วย

2. ค่าที่ส่งกลับจากฟังก์ชั่น (Return Value) ใน Berkeley Socket กับ Winsock จะต่างกัน เช่น ถ้าเรียกให้ฟังก์ชั่น socket 0 จะส่งค่ากลับเป็น -1 หากการทำงานผิดพลาดในการทำงานของ ระบบปฏิบัติการยูนิกซ์ แต่ถ้าหากเป็น Winsock ค่าที่ส่งกลับในกรณีนี้คือ INVALID_SOCKET

3. ชื่อของฟังก์ชั่นที่ทำหน้าที่เดียวกันอาจแตกต่างกันเช่น ฟังก์ชั่น close 0 เป็นฟังก์ชั่นในการปิด การติดต่อของซอคเก็ตในระบบปฏิบัติการยูนิกซ์ แต่ถ้าหากเป็นชื่อใน Winsock จะใช้ชื่อ closesocket 0

4. ในระบบการปฏิบัติการยูนิกซ์การจัดการกับซอคเก็ตจะเหมือนกับแฟ้มข้อมูล (File) ที่อยู่บนดิสก์ แต่ใน Winsock การจัดการกับซอคเก็ตจะต่างกันออกไป

5. Winsock มีฟังก์ชั่นที่เพิ่มเติมจาก Berkeley Socket เพื่อจะสนับสนุนการทำงานแบบแมสเสจไดรเวนของสถาปัตยกรรมของระบบปฏิบัติการวินโดส์ซึ่งจะได้กล่าวในหัวข้อต่อไป

2.3.3 ส่วนเพิ่มเติมของ Winsock จาก Berkeley Socket

ส่วนที่เพิ่มขึ้นมาหลายส่วนของ Winsock เนื่องมาจากเหตุที่ระบบปฏิบัติการวินโดวส์นั้นมีสถาปัตยกรรมที่เรียกว่าแมสเสจไดรเวน และเพื่อจะสนับสนุนการทำงานแบบนอนพรีเอมพ์ตีฟ (Nonpreemptive) ของระบบปฏิบัติการวินโดวส์อีกด้วย ในส่วนต่อไปจะได้กล่าวถึงสถาปัตยกรรมแมสเสจไดรเวนของระบบปฏิบัติการวินโดวส์ก่อน เพื่อเป็นพื้นฐานความเข้าใจในการอธิบายต่อไป

2.3.3.1 สถาปัตยกรรมแมสเสจไดรเวนของวินโดวส์

ในที่นี้จะกล่าวโดยสรุปเกี่ยวกับสถาปัตยกรรมแมสเสจไดรเวน ดังนั้นทุกๆ โปรแกรมที่ทำงานภายใต้ระบบปฏิบัติ การวินโดวส์นี้จะต้องมีส่วนประกอบหลักสำคัญ 2 ส่วน ได้แก่ ลูป (loop) ที่ทำหน้าที่รับแมสเสจ และโปรแกรมย่อยอื่นๆที่เป็นส่วนปฏิบัติงานจริงของโปรแกรม โดยในส่วน ที่เป็นลูปคอยรับแมสเสจนั้นจะรับแมสเสจจากแมสเสจคิว (Message Queue) เมื่อโปรแกรมได้รับแมสเสจแล้วจะทำการปฏิบัติงานตามโปรแกรมย่อยที่สัมพันธ์กับแมสเสจที่ได้รับนั้น ซึ่งการทำงานแบบนี้จึงเป็นที่มาของแมสเสจไดรเวน หรือ อีเวนต์ไดรเวน (Event-driven) เพราะไม่มีส่วนของโปรแกรมใดๆ ทำงานได้จนกว่าจะมีเหตุการณ์ หรือแมสเสจที่เกี่ยวข้องเกิดขึ้น โดยทั่วไปตัวอย่างของแมสเสจคือ การกดคีย์บอร์ด หรือ การเลื่อนเมาส์ และมาจากภายในตัวระบบปฏิบัติการเอง ส่วนโปรแกรมย่อยที่เรียกใช้ในการปฏิบัติการจริงของโปรแกรมที่จะปฏิบัติตาม แมสเสจที่ได้รับนั้นจะอยู่ในรูปคลาสของโปรแกรม โดยมีที่มา 2 แบบ คือมาจากการโปรแกรมของผู้เขียนโปรแกรมนั้น และบางส่วนจะเป็นคลาสที่ถูกกำหนดไว้ล่วงหน้าโดยระบบปฏิบัติการวินโดวส์

เมื่อทำงานตามโปรแกรมย่อยแล้ว โปรแกรมจะเข้าสู่การทำงานในลูปรอรับแมสเสจอีกครั้งหนึ่งเพื่อคอยแมสเสจที่จะเข้ามาในแมสเสจคิวต่อไป แต่ในระหว่างที่โปรแกรมกำลังทำงานอยู่ในโปรแกรมย่อยระบบปฏิบัติการวินโดวส์จะไม่สามารถควบคุมการทำงานของโปรแกรมนั้นๆ ได้ เรียกลักษณะงานแบบนี้ว่านอนพรีเอมพ์ตีฟ (Nonpreemptive)

2.3.3.2 ฟังก์ชั่นการทำงานแบบอะซิงโครนัสของ Winsock

ในการออกแบบ Winsock นั้นครั้งแรกออกแบบมาเพื่อทำงานกับสภาวะแบบนอนพรีเอมพ์ตีฟของระบบปฏิบัติ การวินโดวส์ ด้วยเหตุนี้จึงต้องเพิ่มฟังก์ชั่นการทำงานจาก Berkeley Socket คือ **การทำงานแบบบล็อกกิ้ง (Blocking) และนอนบล็อกกิ้ง(Nonblocking)**

ฟังก์ชั่นในการทำงานของ Berkeley Sockets หลายฟังก์ชั่นเป็นฟังก์ชั่นที่ไม่ทราบเวลาในการส่งค่ากลับที่แน่นอน เรียกลักษณะการทำงานแบบนี้ว่าการทำงานแบบบล็อก ซึ่งในสภาวะการทำงานของระบบปฏิบัติการยูนิกซ์การทำงานของฟังก์ชั่นแบบนี้จะไม่เป็นปัญหาที่ร้ายแรง เพราะระบบปฏิบัติการจะทำการจัดการในส่วนที่จะพรีเอพต์ (Preempt) โปรแกรมที่บล็อก และให้โปรแกรมอื่นทำงานแทน แต่ในทางกลับกันบนระบบปฏิบัติการวินโดวส์ ระบบปฏิบัติการไม่สามารถจะทำ การพรีเอมพต์โปรแกรมได้ จึงต้องรอให้โปรแกรมนั้นคืนการทำงานให้แก่ระบบปฏิบัติการเท่า นั้น Winsock จึงจำเป็นต้องฟังก์ชั่นการทำงานแบบอะซิงโครนัส เพื่อที่จะแก้ปัญหาดังกล่าว เพราะว่าการทำงานบนระบบเครือข่ายไม่สามารถกำหนดเวลาได้แน่นอน ฟังก์ชั่นที่ทำงานแบบ อะซิงโครนัสคือ ฟังก์ชั่นที่เมื่อเรียกทำงานแล้วจะส่งค่ากลับทันที แต่อาจจะยังไม่มีการทำงานเกิด ขึ้นในทันที และเมื่อทำงานเสร็จแล้วจะ

มีแมสเสจส่งกลับมาหาโปรแกรมที่เรียกใช้ฟังก์ชั่นนั้น เพื่อบอกถึงการทำงานที่เสร็จสิ้น และฟังก์ชั่นที่เป็นฟังก์ชั่นแบบอะซิงโครนัสจะมีคำนำหน้าว่า WSAAsync นำหน้าชื่อฟังก์ชั่น ซึ่งจะเห็นได้ว่าการทำงานแบบอะซิงโครนัสถูกออกแบบมาเพื่อให้เข้ากับการทำงานในระบบปฏิบัติการวินโดวส์เป็นอย่างดี

2.4 การใช้งาน Winsock Control

Winsock จะมีคำสั่งที่ยอมให้ผู้ขียน Program ควบคุมการติดต่อ เครื่องคอมพิวเตอร์ที่อยู่ที่อื่น และควบคุมการแลกเปลี่ยนข้อมูลระหว่างคอมพิวเตอร์ 2 เครื่อง Internet ActiveX Control Pack จะประกอบด้วย คำสั่งของ Winsock 2 แบบ คือ Winsock TCP (Transmission Control Protocol) และ Winsock UDP (User Datagram Protocol) ลักษณะการติดต่อ 2 แบบนี้ สามารถที่จะใช้ในการสร้างโปรแกรมที่ใช้ในการติดต่อระหว่างเครื่องลูกข่ายกับเครื่องแม่ข่าย

ข้อแตกต่างระหว่างโปรโตคอล 2 แบบนี้

- รูปแบบของ Winsock TCP เป็นแบบการติดต่อขั้นพื้นฐาน กล่าวคือ จะคล้ายกับการติดต่อทางโทรศัพท์คือผู้ใช้จะต้องทำการตรวจสอบก่อนที่จะเริ่มการติดต่อ

- รูปแบบของ Winsock UDP จะเป็นแบบ Connectionless คือเป็นแบบที่ไม่มีการตรวจสอบ มีลักษณะการทำงานคล้ายกับสถานีส่งคลื่นวิทยุ คือ เครื่องคอมพิวแตอร์จะส่งข้อมูลแบบกระจายโดยไม่ต้องทราบว่าข้อมูลที่ส่งถึงผู้รับหรือไม่

ทั้ง Winsock TCP และ Winsock UDP นั้น มีลักษณะการส่งถ่ายข้อมูลทั้งสองทิศทาง

2.4.1 Winsock TCP Control

ลักษณะการใช้งาน:

- ใช้สร้างโปรแกรมประยุกต์ทางด้านผู้ส่ง โดยมีการเก็บข้อมูลของผู้ที่จะส่งไว้ก่อนที่จะส่งข้อมูลให้ฝ่ายผู้รับ

- สร้างโปรแกรมประยุกต์ทางด้านผู้รับที่มี ฟังก์ชั่นในการรวบรวมข้อมูลของผู้ส่งแต่ลงเครื่อง

ลักษณะการใช้งานที่ใช้ Winsock TCP ในการส่งข้อมูลไปให้ Computer ที่จะติดต่อ:

ในการใช้งาน Winsock TCP เพื่อติดต่อกับ Computer 2 เครื่องแบบ Real-time ระบบเครือข่ายนั้น เครื่อง Computer ที่จะเป็นผู้ให้บริการ (Server) จะเป็นผู้รับข้อมูล , กำหนดช่องการสื่อสารให้ Computer ที่จะขอใช้บริการ (Client) เมื่อ Winsock TCP บนเครื่อง Server ได้รับคำร้องขอใช้บริการ มันจะสร้าง Socket ใหม่ และทำการสร้างการติดต่อ เมื่อสามารถสร้างการติดต่อได้แล้ว Server และ Client จะสามารถส่งข้อมูลซึ่งกันและกันได้ และเมื่อการสื่อสารเสร็จสิ้น Client จะทำการขอยกเลิกการติดต่อ Winsock ที่ทำงานบนเครื่อง Server เมื่อนั้น Server จะทำการลบ Socket ที่ได้ทำการสร้างไว้

2.5 การเรียกใช้งาน Function ในการทำงานเกี่ยวกับเสียง ด้วยฟังก์ชั่น API

ฟังก์ชั่น API ด้านมัลติมีเดียในกลุ่มของ Command-Message และ Command-string เป็นฟังก์ชั่นที่เหมาะกับโปรแกรมประเภท VB/Win เป็นอย่างมาก กลุ่มฟังก์ชั่นระดับล่าง (Low Level Waveform Audio Function) ซึ่งเป็นกลุ่มฟังก์ชั่น API ด้านมัลติมีเดียที่ได้รับการออกแบบสำหรับภาษาซีโดยเฉพาะ และประกอบด้วยฟังก์ชั่นที่สามารถใช้จัดการกับไฟล์ .WAV เช่น การอัดเสียง (Recording) การทำเทคนิคเสียงพิเศษ (Sound effect) การควบคุมระดับโทนเสียง (Pitch Control) เป็นต้น

2.5.1 รายละเอียดคำนำหน้าฟังก์ชั่น

aux	ฟังก์ชั่นด้าน Auxiliary Audio
joy	ฟังก์ชั่นด้าน Joystick
mci	ฟังก์ชั่นด้าน Media Control Interface
midi	ฟังก์ชั่นด้าน Low-Level MIDI Audio
mmio	ฟังก์ชั่นด้านการจัดการ I/O ของไฟล์มัลติมีเดีย
snd	ฟังก์ชั่นด้าน High-Level Sound
time	ฟังก์ชั่นด้าน Timer Event
wave	ฟังก์ชั่นด้าน Low-Level Waveform Audio

ฟังก์ชั่นที่นำหน้าด้วยคำว่า wave ที่เป็นฟังก์ชั่นระดับล่างที่จัดการกับไฟล์.wav ซึ่งสามารถจัดกลุ่มได้ดังนี้

2.5.2 กลุ่มของฟังก์ชั่น

ไมโครซอฟต์ได้ทำการแบ่งกลุ่มของฟังก์ชั่นระดับล่างด้านเสียงออกตามลักษณะหน้าที่การให้บริการ การแยกกลุ่มของฟังก์ชั่นด้านมัลติมีเดียซึ่งเป็นรูปแบบเฉพาะของไมโครซอฟต์โดยไมโครซอฟต์ได้ใช้คำนำหน้า (prefix) นำหน้าชื่อของฟังก์ชั่นเพื่อกำหนดความสัมพันธ์ของแต่ละฟังก์ชั่นดังต่อไปนี้

2.5.2.1 กลุ่มฟังก์ชั่นด้าน Querying

ก่อนจะทำการเล่นหรืออัดเสียงด้วยไฟล์.wav จะต้องทำการสำรวจความสามารถของฮาร์ดแวร์ในเครื่องคอมพิวเตอร์เสียก่อน ทั้งนี้เพื่อให้สามารถจัดการกับไฟล์ .wav ได้อย่างถูกต้อง ซึ่งประกอบด้วยฟังก์ชั่นดังต่อไปนี้

waveInGetNumDevs รายงานจำนวนอุปกรณ์รับสัญญาณเสียง(waveform) ที่ติดตั้งในระบบ
waveInGetDevCaps รายงานความสามารถของแต่ละอุปกรณ์รับสัญญาณเสียง
waveOutGetNumDevs รายงานจำนวนอุปกรณ์ส่งสัญญาณเสียง(waveform) ที่ติดตั้งในระบบ
waveOutGetDevCaps รายงานความสามารถของแต่ละอุปกรณ์ส่งสัญญาณเสียง

2.5.2.2 กลุ่มฟังก์ชั่นด้านการเปิดและปิดอุปกรณ์

ก่อนการเล่นกลับหรือการอัดเสียงในรูปแบบ .wav ทุกครั้ง จะต้องทำการเปิดอุปกรณ์ที่เกี่ยวข้องเสียก่อน เพื่อเป็นการอ้างสิทธิ์การใช้งานสำหรับแอพพลิเคชั่นนั้นๆ และควรจะปิดอุปกรณ์เมื่อไม่ต้องการใช้งานอีกต่อไป เพื่อให้แอพพลิเคชั่นอื่น ๆ สามารถด้างสิทธิ์การใช้งานได้ต่อไป ซึ่งประกอบด้วยฟังก์ชั่นดังต่อไปนี้

waveInOpen เปิดอุปกรณ์รับสัญญาณเสียงสำหรับการอัดเสียง

waveInClose ปิดอุปกรณ์รับสัญญาณเสียงที่กำหนด

waveOutOpen เปิดอุปกรณ์ส่งสัญญาณเสียงสำหรับการเล่นกลับ

waveOutClose ปิดอุปกรณ์ส่งสัญญาณเสียงที่กำหนด

2.5.2.3 กลุ่มของฟังก์ชั่นด้านการอ่านค่า ID ของอุปกรณ์เสียงสัญญาณเสียง

เนื่องจากการจัดการออบเจ็คใดๆ ของวินโดวส์ มันจะใช้รหัส ID ของแต่ละออบเจ็คในการควบคุมการทำงานหรือสั่งการ ซึ่งอุปกรณ์ด้านมัลติมีเดียก็เช่นกัน ก่อนที่วินโดวส์จะมีการติดต่อกับอุปกรณ์เหล่านี้ก็จะมีการกำหนดหมายเลข ID (device handle) ให้เสียก่อน ซึ่งสามารถอ่านหมายเลข ID โดยใช้ฟังก์ชั่นต่อไปนี้

waveInGetID อ่านหมายเลข ID ของอุปกรณ์รับสัญญาณเสียง

waveOutGetID อ่านหมายเลข ID ของอุปกรณ์ส่งสัญญาณเสียง

2.5.2.4 กลุ่มฟังก์ชั่นด้านการเล่นข้อมูลในรูปแบบสัญญาณเสียง

ฟังก์ชั่นในกลุ่มนี้ได้เตรียมไว้สำหรับการจัดเตรียมกลุ่มของข้อมูล(block) เพื่อใช้ส่งต่อให้กับอุปกรณ์ส่ง สัญญาณเสียงต่อไปนี้ ทั้งนี้เนื่องจากการจัดเก็บข้อมูลในรูปแบบสัญญาณเสียงลงในไฟล์นั้น ยังไม่เหมาะสำหรับการจัดส่งให้กับอุปกรณ์ส่งสัญญาณเสียงที่เดียว เพราะข้อมูลในไฟล์ .wav นอกจากจะมีการจัดเก็บข้อมูลของสัญญาณเสียงแล้ว ยังมีการจัดเก็บต่าต่างๆเอาไว้ในส่วนของ header เพื่อใช้เป็นข้อมูลสำหรับการจัดการกับบัฟเฟอร์หรือข้อมูลในไฟล์ต่อไป โดยหัวข้อเกี่ยวกับไฟล์ในรูปแบบของ RIFF ซึ่งประกอบด้วยฟังก์ชั่นดังต่อไปนี้

waveOutPrepareHeader	เตรียมความพร้อมของดีไวซ์ไดรเวอร์สำหรับการส่งสัญญาณเสียง และเป็นการบอกดีไวซ์ไดรเวอร์ให้ทราบว่ากลุ่มของข้อมูลที่จะส่งมาเป็นข้อมูลสำหรับการเล่นกลับ
waveOutPrepareHeader	เขียนกลุ่มของข้อมูลให้กับอุปกรณ์ส่งสัญญาณเสียง
waveOutUnprepareHeader	บอกดีไวซ์ไดรเวอร์ส่งสัญญาณเสียงสามารถยกเลิกหรือลบทิ้ง การเตรียมการกับกลุ่มของข้อมูลที่ถูกส่งมาเป็นข้อมูลสำหรับการเล่นกลับ

2.5.2.5 กลุ่มของฟังก์ชั่นด้านการอัดสัญญาณเสียง

นอกจากจะสามารถเล่นกลับไฟล์ .wav อัดสัญญาณเสียงที่ได้รับจากอุปกรณ์รับสัญญาณเสียงลงในไฟล์ .wav ด้วยฟังก์ชั่นในกลุ่มนี้ ซึ่งเสียงที่รับเข้ามาจะถูกจัดเก็บเอาไว้ในบัฟเฟอร์ซึ่งเป็นพื้นที่ใน

หน่วยความจำซึ่งเตรียมเอาไว้สำหรับการขนถ่ายข้อมูลในระหว่างการอัดสัญญาณเสียง ซึ่งประกอบด้วยฟังก์ชั่นดังต่อไปนี้

-waveInAddBuffer ส่งบัฟเฟอร์ให้กับอุปกรณ์สำหรับการส่งสัญญาณเสียง ซึ่งบัฟเฟอร์นี้จะถูกใช้จัดเก็บข้อมูลในรูปแบบสัญญาณเสียงและจะถูกส่งกลับมายังแอพพลิเคชั่นต้นทางต่อไป

-waveInPrepareHeader บอกให้อุปกรณ์รับสัญญาณเสียงทราบว่า กลุ่มของข้อมูลที่จะส่งมาเป็นข้อมูลสำหรับการอัดเสียง

-waveInUnprepareHeader บอกดีไวซ์ไดรเวอร์สำหรับการรับสัญญาณเสียงสามารถยกเลิกหรือลบทิ้งการเตรียมการกับกลุ่มของข้อมูลที่ถูกส่งมาเป็นข้อมูลสำหรับการอัดเสียง

2.5.2.6 กลุ่มฟังก์ชั่นด้านการอ่านตำแหน่งปัจจุบันของอุปกรณ์สัญญาณเสียง

ในขณะที่ทำการเล่นกลับหรืออัดเสียงด้วยไฟล์ .wav จะสามารถอ่านตำแหน่งการจัดการข้อมูลในปัจจุบันจากอุปกรณ์ได้โดยใช้ฟังก์ชั่นในกลุ่มนี้ ดังต่อไปนี้

-waveInGetPosition อ่านตำแหน่งการอัดเสียงปัจจุบันจากอุปกรณ์รับสัญญาณเสียง

-waveOutGetPosition อ่านตำแหน่งการอัดเสียงปัจจุบันจากอุปกรณ์ส่งสัญญาณเสียง

2.5.2.7 กลุ่มฟังก์ชั่นด้านการควบคุมการเล่นกลับ

ฟังก์ชั่นในกลุ่มนี้จะทำหน้าที่ควบคุมวิธีการเล่นกลับทั้งหมดของอุปกรณ์ส่งสัญญาณเสียง เช่น Pause,Stop หรือ Play เป็นต้น ซึ่งการเล่นกลับนั้นจะเกิดขึ้นทันทีที่มีการเขียนโค้ดเพื่อทำการส่งกลุ่มของข้อมูลไปยังอุปกรณ์ส่งสัญญาณเสียง ซึ่งประกอบด้วยฟังก์ชั่นดังต่อไปนี้

-waveOutBreakLoop หลุดลูปของอุปกรณ์ส่งสัญญาณเสียง

-waveOutPause หยุดการเล่นกลับชั่วคราวของอุปกรณ์ส่งสัญญาณเสียง

-waveOutRestart เริ่มการเล่นกลับต่อจากตำแหน่งที่หยุดชั่วคราวของอุปกรณ์ส่งสัญญาณเสียง

-waveOutReset หยุดการเล่นกลับของอุปกรณ์ส่งสัญญาณเสียง

2.5.2.8 กลุ่มฟังก์ชั่นด้านการควบคุมการอัดสัญญาณเสียง

ฟังก์ชั่นในกลุ่มนี้ทำหน้าที่ควบคุมวิธีการอัดเสียงลงในไฟล์ .wav ซึ่งจะมีความแตกต่างจากฟังก์ชั่นด้านการอัดสัญญาณเสียง ซึ่งประกอบด้วยฟังก์ชั่นต่อไปนี้

-waveInStart เริ่มต้นการอัดเสียงของอุปกรณ์รับสัญญาณเสียง

-waveInStop หยุดการอัดเสียงชั่วคราวของอุปกรณ์รับสัญญาณเสียง

-waveInReset หยุดการอัดเสียงของอุปกรณ์ส่งสัญญาณเสียง

2.5.2.9 กลุ่มฟังก์ชั่นด้านการเปลี่ยนเสียงสูง-ต่ำและอัตราความเร็วด้านการเล่นกลับ

สามารถใช้ฟังก์ชั่นในกลุ่มนี้สำหรับการตบแต่งสัญญาณเสียงให้มีความแตกต่างจากข้อมูลต้นฉบับในไฟล์ .wav ได้ โดยการปรับระดับเสียงสูง-ต่ำ(Pitch) หรือการปรับอัตราความเร็วด้านการเล่นกลับซึ่งในแอพพลิเคชั่นด้านมัลติมีเดียชั้นนำก็ล้วนแต่มีคำสั่งสำหรับงานด้านนี้รวมอยู่ด้วย ซึ่งประกอบด้วยฟังก์ชั่นดังต่อไปนี้

-waveOutGetPitch อ่านค่าสเกลของระดับเสียงสำหรับอุปกรณ์ส่งสัญญาณเสียง

-waveOutGetPlaybackRate อ่านสเกลของระดับเสียงของการเล่นกลับสำหรับอุปกรณ์ส่งสัญญาณเสียง

-waveOutSetPitch กำหนดค่าสเกลของระดับเสียงสำหรับอุปกรณ์ส่งสัญญาณเสียง

-waveOutSetPlaybackRate กำหนดค่าสเกลของระดับเสียงของการเล่นกลับสำหรับอุปกรณ์ส่งสัญญาณเสียง

2.5.2.10 กลุ่มฟังก์ชั่นด้านการเปลี่ยนความดังของเสียง

ฟังก์ชั่นในกลุ่มนี้จะทำหน้าที่อ่านและกำหนดระดับความดังของสัญญาณเสียง (volume) สำหรับการเล่นกลับเท่านั้น ซึ่งประกอบด้วยฟังก์ชั่นดังต่อไปนี้

-waveOutGetVolume อ่านระดับความดังของสัญญาณเสียงปัจจุบัน จากอุปกรณ์ส่งสัญญาณเสียง

-waveOutSetVolume กำหนดระดับความดังของสัญญาณเสียงปัจจุบัน ของอุปกรณ์ส่งสัญญาณเสียง

2.5.2.11 กลุ่มฟังก์ชั่นด้านการส่งข้อความ (Message) ให้กับไดรเวอร์

วินโดวส์เป็นระบบปฏิบัติการที่ทำงานในรูปแบบของข้อความ (Message based) ซึ่งข้อความที่ส่งไปมาระหว่างวินโดว์ต่างๆกับระบบปฏิบัติการนั้นไม่ใช่ข้อความตามที่อ่านกัน แต่เป็นค่าตังเลขที่บอกถึงความหมายของข้อความที่ส่งให้แก่กัน ซึ่งไดรเวอร์ด้านมัลติมีเดียก็เช่นกัน ดังนั้นในแอพพลิเคชั่นจึงสามารถสื่อสารกับไดรเวอร์เหล่านี้ได้โดยอาศัยข้อความนี้ ซึ่งประกอบด้วยฟังก์ชั่นดังต่อไปนี้

- waveInMessage ส่งข้อความให้ดีไวซ์ไดรเวอร์รับสัญญาณเสียง

- waveOutMessage ส่งข้อความให้ดีไวซ์ไดรเวอร์ส่งสัญญาณเสียง

2.5.2.12 กลุ่มฟังก์ชั่นด้านการจัดการข้อผิดพลาด

ฟังก์ชั่นด้านมัลติมีเดียก็เหมือนกับฟังก์ชั่นอื่นๆ ก็คือ สามารถเกิดข้อผิดพลาดได้ถ้าหากข้อมูลหรืออุปกรณ์ที่เกี่ยวข้องมีการทำงานที่ไม่ถูกต้อง ดังนั้นเมื่อฟังก์ชั่นในระดับล่างตามที่กล่าวมาแล้วทั้งหมดในข้างต้นคืนกลับรหัสค่าเลขบอกความผิดำลาด ก็จะสามารถใช้ฟังก์ชั่นในกลุ่มนี้ในการอ่านรายละเอียดข้อผิดพลาดที่แกิดขึ้นได้ซึ่งประกอบด้วยฟังก์ชั่นดังต่อไปนี้

-waveInGetErrorText อ่านรายละเอียดของข้อผิดพลาดที่เกิดขึ้นกับการรับสัญญาณเสียง

-WaveOutGetErrorText อ่านรายละเอียดของข้อผิดพลาดที่เกิดขึ้นกับการส่งสัญญาณเสียง

ฟังก์ชั่นตามที่ได้กล่าวมาทั้งหมด ล้วนแต่เป็นฟังก์ชั่นระดับล่าง สำหรับการจัดการกับการรับส่งข้อมูลในรูปแบบสัญญานเสียง ซึ่งความสามารถโดยส่วนใหญ่ของฟังก์ชั่นในกลุ่มนี้จะไม่มีในฟังก์ชั่นระดับสูง ดังนั้นถ้าต้องการสร้างแอพพลิเคชันด้านมัลติมีเดียที่จัดการเกี่ยวกับสัญญานเสียง จำเป็นต้องเรียนรู้วิธีการใช้งานฟังก์ชั่นเหล่านี้ แต่เนื่องจากฟังก์ชั่นเหล่านี้ได้รับการออกแบบที่เน้นเฉพาะภาษาซีเท่านั้น ดังนั้นในการใช้งานจริงในบางครั้งอาจจะต้องอาศัยเทคนิคพิเศษในการแปลงข้อมูลเพื่อให้เกิดความคอมแพตทิเบิลกับ VB/Win ซึ่งจะกล่าวในหัวข้อถัดไป

2.5.3 การใช้ฟังก์ชั่นและข้อมูลโครงสร้าง

ในที่นี้จะขออธิบายเฉพาะฟังก์ชั่นในกลุ่มฟังก์ชั่นด้าน Querying เท่านั้น แต่ก่อนที่จะอธิบายถึงฟังก์ชั่นเหล่านี้ จะขออธิบายถึงข้อมูลโครงสร้างที่เกี่ยวข้องเสียก่อน ดังต่อไปนี้

2.5.3.1 ข้อมูลแบบโครงสร้าง WAVEINCAPS

เป็นโครงสร้างสำหรับฟังก์ชั่น waveInGetDevCaps ซึ่งประกอบด้วยสมาชิกที่บอกถึงคุณสมบัติหรือความสามารถของอุปกรณ์รับสัญญาณเสียงเช่น รหัสประจำตัวของผู้ผลิต รูปแบบสัญญาณเสียงหรือจำนวนแชลแนล เป็นต้น โดยที่แต่ละสมาชิกมีรายละเอียดดังต่อไปนี้

```
Type WAVEINCAPS
        wMid            As Integer                  'manufacturer ID
        wPid            As Integer                  'product ID
        vDriverVersion  As Integer                  'version of the driver
        szPname         As String *MAXPNAMELEN      'product name (NULL Terminated string)
        dwFormats       As Long                     'formats supported
        wChannels       As Integer                  'number of channels supported
End Type
```

wMid และ wDeviceID

ข้อมูลชนิดเลขจำนวนเต็ม บอกถึงรหัสประจำตัวของผู้ผลิตและรหัสประจำตัวผลิตภัณฑ์ตามลำดับ ซึ่งโดยปกติผู้ผลิตการ์ดเสียงต่างๆ จะทำการอัพเดตรหัสของตนลงในซอร์ฟแวร์ โดยการแจ้งการลงทะเบียนไปยังหน่วยงาน Multimedia Systems Group ของไมโครซอฟต์ สำหรับข้อมูลในตารางที่ 1 เป็นตัวอย่างของรหัสประจำตัวที่ได้รับการลงทะเบียนและอัพเดตลงในซอฟต์แวร์แล้ว

รหัสประจำตัวผู้ผลิต (Manufacturer ID)

ค่าคงที่	ตัวเลข	รายละเอียด
MM_MICROSOFT	1	ไดรเวอร์ที่ถูกพัฒนาโดยบริษัทไมโครซอฟต์

รหัสประจำตัวผลิตภัณฑ์ (Product ID)

ค่าคงที่	ตัวเลข	รายละเอียด
MM_MIDI_MAPPER	1	ไมโครซอฟต์ MIDI MAPPER
MM_SNDBLST_MIDIOUT	3	พอร์ตส่ง MIDI ของ Sound Blaster
MM_SNDBLST_MIDIN	4	พอร์ตรับ MIDI ของ Sound Blast
MM_SNDBLST_SYNTH	5	ซินธีไซเซอร์ภายในของ Sound Blaster
MM_SNDBLST_WAVEOUT	6	พอร์ตส่ง waveform ของ Sound Blaster
MM_SNDVLST_WAVEIN	7	พอร์ตรับ waveform ของ Sound Blaster
MM_ADLIB	9	ซินธีไซเซอร์ที่คอมแพททิเบิลของ Ad Lib
MM_MPU401_MIDIOUT	10	พอร์ตส่ง MIDI ของ MPU401
MM_MPU401_MIDIIN	11	พอร์ตรับ MIDI ของ MPU401
MM_PC_JOYSTICK	12	การ์ดควบคุมเกมของบริษัท IBM

ตารางที่ 1

vDriver Version

ข้อมูลชนิดเลขจำนวนเต็มกำหนดหมายเลขเวอร์ชันของดีไวซ์ไดรเวอร์สำหรับอุปกรณ์รับสัญญาณเสียง ซึ่งมีการจัดจัดเก็บในรูปแบบเลขฐานสิบหก 2 ไบต์ โดยที่ไบต์ลำดับสูงจะเป็นหมายเลข

หลัก (major number) และไบต์ลำดับล่างจะเป็นหมายเลขรอง(minor number) เช่น ค่า 200H จะหมายถึงหมายเลขเวอร์ชัน 2.00 เป็นต้น

szPname

ข้อมูลชนิดสตริงมีความยาวเท่ากับ 32 ไบต์(MAXPNAMELEN) สำหรับจัดเก็บรายละเอียดของผู้ผลิตหรือผลิตภัณฑ์ก็ได้ โดยที่สตริงที่ได้จะเป็น ASCIIZ ซึ่งหมายถึงเป็นสตริงงที่สิ้นสุดด้วยตัวอักษร ASCII 0

dwFormats

ข้อมูลชนิดเลขจำนวนเต็ม long integer ที่บอกถึงรูปแบบของการจัดเก็บสัญญาณเสียงมาตรฐานที่อุปกรณ์รับ สัญญาณเสียงสนันสนุน ซึ่งโดยปกติอุปกรณ์หนึ่งๆจะสนับสนุนมากกว่า 1 รูปแบบ โดยค่าที่จัดเก็บลงในสมาชิกนี้ จึงจัดเก็บในรูปแบบของค่าแฟลก โดยใช้ตัวปฏิบัติการ Or ดังนั้นในการตรวจสอบค่าของ dwFormats จึงต้องใช้ตัว ปฏิบัติการ And ในการทดสอบแต่ละแฟลก ซึ่งรูปแบบมาตรฐานที่เป็นไปได้มีดังตารางที่ 2

ค่าคงที่	ตัวเลข	รายละเอียด
WAVE_INVALIDFORMAT	&HO	รูปแบบไม่เป็นจริง (โดยส่วนใหญ่หมายถึงการเกิดข้อผิดพลาด)
WAVE_FORMAT_1M08	&H1	รูปแบบ 11.025 kHz,Mono, 8-bit
WAVE_FORMAT_1S08	&H2	รูปแบบ 11.025 kHz,Stereo, 8-bit
WAVE_FORMAT_1M16	&H4	รูปแบบ 11.025 kHz, Mono, 16-bit
WAVE_FORMAT_1S16	&H8	รูปแบบ 11.025 kHz, Stereo, 16-bit
WAVE_FORMAT_2M08	&H10	รูปแบบ 22.05 kHz,Mono, 8-bit
WAVE_FORMAT_2S08	&H20	รูปแบบ 22.05 kHz,stereo,8-bit
WAVE_FORMAT_2M16	&H40	รูปแบบ 22.05 kHz,Mono, 16-bit

WAVE_FORMAT_2S16	&H80	รูปแบบ 22.05 kHz,Stereo, 16-bit
WAVE_FORMAT_4M08	&H100	รูปแบบ 44.1 kHz,Mono, 8-bit
WAVE_FORMAT_4S08	&H200	รูปแบบ 44.1 kHz,Stereo, 8-bit
WAVE_FORMAT_4M16	&H400	รูปแบบ 44.1 kHz,Mono, 16-bit
WAVE_FORMAT_4S16	&H800	รูปแบบ 44.1 kHz,Stereo, 16-bit

ตารางที่ 2

ข้อมูลชนิดเลขจำนวนเต็ม ที่บอกถึงจำนวนแชลแนลที่อุปกรณ์รับสัญญานเสียงสนับสนุนซึ่งมีค่าตั้งแต่ 1 ถึง 2 เท่านั้น โดยที่ 1 หมายถึงโมโน และ 2 หมายถึง สเตอริโอ

2.5.3.2 ข้อมูลแบบโครงสร้าง WAVEOUTCAPS

เป็นโครงสร้างสำหรับฟังก์ชั่น wavwOutGetDevCaps ซึ่งประกอบด้วยสมาชิกที่บอกถึงคุณภาพหรือความสามารถของอุปกรณ์ส่งสัญญาณเสียง เช่น รหัสประจำตัวของผู้ผลิต รูปแบบสัญญาณเสียง หรือ จำนวนแชลแนลเป็นต้น โดยที่มีสมาชิกที่เหมือนกับสมาชิกของโครงสร้าง WAVEINCAPS แต่จะแตกต่างกันก็เฉพาะสมาชิก dwSupport เท่านั้นที่ไม่มีในโครงสร้าง WAVEINCAPS ดังนั้นผู้อ่านจึงสามารถอ่านรายละเอียดของสมาชิกที่เหมือนกันได้จากโครงสร้างWAVEINCAPS

waveInGetNumDevs	
Declaration	Declare Functio waveInGetNumDevs Lib "MMSYSTEM" ( ) As Integer
Description	ทำหน้าที่รายงานจำนวนอุปกรณ์รับสัญญาณเสียง(waveform) ที่ติดตั้งในระบบ
Parameters	ไม่มี
Example	Dim wInNumDevs% wInNumDevs% = waveInGetNumDevs ( ) Print "Number of Input Device : "; wInNumDevs%

Comment	ค่าที่ส่งกลับโดยฟังก์ชั่นนี้จะถูกนำไปใช้ในการกำหนดอุปกรณ์รับสัญญาณเสียง สำหรับฟังก์ชั่น waveInGetDevCaps ต่อไป

```
Type WAVEOUTCAPS
    wMid            As Integer              ' manufacturer ID
    wPid            As Integer              ' product ID
    vDriverVersion  As Integer              ' version of the driver
    szPname         As String * MAXPNAMELEN ' product name
                                              (NULL terminated string)
    dwFormats       As Long                 ' formats supported
    wChannels       As Integer              ' number of sources supported
    dwSupport       As Long                 ' functionality supported by driver
Fnd Type
```

dwSupport

ข้อมูลชนิดเลขจำนวนเต็ม long ที่บอกถึงฟังก์ชั่นที่แต่ละอุปกรณ์ส่งสัญญาณเสียงสนับสนุน ดังมีรายละเอียดดังตารางที่ 3

ตารางที่ 3

ค่าคงที่	ตัวเลข	รายละเอียด
WAVECAPS_PITCH	&H1	สนับสนุนการควบคุมระดับโทนเสียง (pitch)

WAVECAPS_PLAYBACKRATE	&H2	สนับสนุนการควบคุมอัตราความเร็วในการเล่นกลับ
WAVECAPS_VOLUME	&H4	สนับสนุนการควบคุมระดับความดังของเสียง (volume)
WAVECAPS_LRVOLUME	&H8	สนับสนุนการแยกการควบคุมระดับความดังด้านซ้ายขวา
WAVECAPS_SYNC	&H10	แสดงถึงการ synchronous ของไดรเวอร์

waveInGetDevCaps

Declaration Declare Function waveInGetDevCaps Lib "MMSYSTEM" (ByVal udeviceid As _Integer, lpCaps As WAVEINCAPS, ByVal uSize As Integer) As Integer

Description ทำหน้าที่รายงานความสามารถและคุณสมบัติของแต่ละอุปกรณ์รับสัญญาณเสียงตามที่กำหนด

Parameters

udeviceid	ตัวแปรชนิดเลขจำนวนเต็ม กำหนดหมายเลข ID ของอุปกรณ์รับสัญญาณเสียงที่ต้องการอ่านข้อมูลรายละเอียด โดยที่หมายเลข ID จะได้จากฟังก์ชั่น waveInGetNumDevs และหมายเลข ID ที่กำหนดให้กับตัวแปรนี้จะสามารถมีค่าได้ตั้งแต่ 0 ถึง 1 เท่านั้น
lpCaps	ตัวแปรชนิดข้อมูลโครงสร้าง WAVEINCAPS
uSize	ตัวแปรชนิดเลขจำนวนเต็ม กำหนดของขนาดข้อมูลโครงสร้าง WAVEINCAPS ที่ส่งให้กับฟังก์ชั่นนี้ ดังนั้น ในการเขียนโค้ดจึงอ่านจำนวนไบต์ของข้อมูลโครงสร้าง WAVEINCAPS โดยใช้ฟังก์ชั่น Len

Example

```
Dim WavIn As WAVEINCAPS
Dim wInNumDevs%
```

```
Dim wInDevsFlag%

wInNumDevs% = waveInGetNumDevs ( )

Print "Waveform Audio Device Information : "

Print "Number of Input Device : "; wInNumDevs%

wInDevsFlag% = waveInGetDevCaps ( (wInNumDevs%-1), WavIn, Len
(WavIn) )

Print "Waveform Audio Input Device Information :"

Print "ManuFacTurer ID : "; WavIn.wMid

Print "Product ID : "; WavIn.wPid

Print "Driver Version : "; sPhraseVersion(WavIn.vDriverVersion)

Print "Product Name : "; sZtrim(WavIn.szPname)

Print "Formats Supported :";sFormatDesc(WavIn.dwFormats)

Print "Number Of Channels :";WavIn.wChannels
```

Comment ค่าที่ส่งกลับโดยฟังก์ชั่นนี้จะบอกถึงความสำเร็จของฟังก์ชั่นดังนี้

0 หมายถึง ฟังก์ชั่นทำงานสำเร็จ

<> 0 หมายถึง ฟังก์ชั่นทำงานล้มเหลว โดยที่รหัสข้อผิดพลาดที่สามารถเป็นไปได้มีดังนี้

MMSYSERR_BADDEVICEID หมายถึง รหัส ID ของอุปกรณ์ไม่ถูกต้อง

MMSYSERR_NODRIVER หมายถึง ไม่มีการติดตั้งไดรเวอร์

waveOutGetNumDevs

Declaration	Declare Function waveOutGetNumDevs Lib "MMSYSTEM" ( ) As Integer
Description	ทำหน้าที่รายงานจำนวนอุปกรณ์ส่งสัญญาณเสียง (waveform) ที่ติดตั้งในระบบ
Parameters	ไม่มี
Example	Dim wOutNumDevs% wOutNumDevs% = waveOutGetNumDevs ( ) Print "Number of Output Device :"; wOutNumDevs%
Comment	ค่าที่ส่งกลับโดยฟังก์ชั่นนี้จะถูกนำไปใช้ในการกำหนดอุปกรณ์ส่งสัญญาณเสียงสำหรับฟังก์ชั่น waveOutGetDevCaps ต่อไป

waveOutGetDevCaps

Declaration	Declare Function waveOutGetDevCaps Lib "MMSYSTEM" (ByVal udeviceid As _Integer, lpCaps As WAVEOUTCAPS,ByVal usize As Integer) As Integer	
Description	ทำหน้าที่รายงานความสามารถและคุณสมบัติของแต่ละอุปกรณ์ส่งสัญญาณเสียงตามที่กำหนด	
Parameters	udeviceid	ตัวแปรชนิดเลขจำนวนเต็ม กำหนดหมายเลข ID ของอุปกรณ์ส่งเสียงที่ต้องการอ่านข้อมูลรายละเอียดโดยที่หมายเลข ID จะได้จากฟังก์ชั่น waveOutGetNumDevs และหมายเลข ID ที่กำหนดให้กับตัวแปรนี้จะสามารถมีค่าได้ตั้งแต่ 0 ถึง 1 เท่านั้น
	lpCaps	ตัวแปรชนิดข้อมูลโครงสร้าง WAVEOUTCAPS

	uSize	ตัวแปรชนิดเลขจำนวนเต็ม กำหนดขนาดของข้อมูลโครงสร้าง WAVEOUTCAPS ที่ส่งให้กับฟังก์ชั่นนี้ ดังนั้นในการเขียนโค้ดจึงอ่านจำนวนไบต์ของข้อมูลโครงสร้าง WAVEOUTCAPS โดยใช้ฟังก์ชั่น Len

Example

```
Dim WavOut As WAVEOUTCAPS
Dim wOutNumDevs%
Dim wOutDevsFlag%
wOutNumDevs% = waveOutGetNumDevs ( )
Print "Waveform Audio Device Information :"
Print "Number of Output Device :" wOutNumDevs%
wOutDevsFlag% = waveOutGetNumDevCaps ( (wOutNumDevs%-1), WavOut, Len
(WavOut) )
Print "Waveform Audio Output Device Information :"
Print "Manufacturer ID : "; WavOut.wMid
Print "Product ID : "; WavOut.wpid
Print "Driver Version : ";sPhraseVersion(WavOut.vDriverVersion)
Print "Product Name : "; sZtrim(WavOut.szPname)
Print "Formats Supported : "; sFormatDesc(WavOut.dwFormats)
Print "Number Of Channels : "; WavOut.wChannels
Print "Functionality Supported : "; sFuncSupport (WavOut.dwSupport)
```

Comment	ค่าที่ส่งกลับโดยฟังก์ชั่นนี้จะบอกถึงความสำเร็จของฟังก์ชั่น ดังนี้
	0 หมายถึง ฟังก์ชั่นทำงานสำเร็จ
	<> 0 หมายถึง ฟังก์ชั่นทำงานล้มเหลว โดยที่รหัสข้อผิดพลาดที่สามารถเป็นไปได้มีดังนี้
	MMSYSERR_BADDEVICEID หมายถึง รหัส ID ของอุปกรณ์ไม่ถูกต้อง
	MMSYSERR_NODRIVER หมายถึง ไม่มีการติดตั้งไดรเวอร์

2.6 การควบคุมการบีบอัดข้อมูลโดยการใช้ Audio Compression Manager

สิ่งนี้เป็นการอธิบาย service function ที่มีอยู่บนACM (Audio Compression Manager)และบรรยายถึงการ Programming ที่นำ service ต่างๆไปใช้ว่าเราสามารถใช้ service ต่างๆได้อย่างไร

Audio Compression Managerให้บริการด้านใดบ้าง

ตัวจัดการบีบอัดสัญญาณเสียงจะมีบริการที่จะรองรับการทำงานต่อไปนี้

- ทำการบีบอัดสัญญาณเสียงและคลายสัญญาณเสียงที่ถูกบีบอัดให้อยู่ในรูปเดิม
- สามารถเลือกรูปแบบและวิธีการในการบีบอัดสัญญาณเสียงได้
- สามารถเลือกรูปแบบของการFilter ได้
- สามารถเปลี่ยนรูปแบบของการบีบอัดสัญญาณจากแบบหนึ่งไปเป็นอีกแบบหนึ่งได้
- ทำการ Filter สัญญาณได้

การทำการ Mapping Waveform Audio Devices

Microsoft Win 32 Application Programming Interface (API) จะมีกลุ่มของ function มาตรฐานสำหรับ audio devices โดย function เหล่านั้นจะสั่งการให้อุปกรณ์เหล่านั้น (audio

device)โดย call ผ่าน device driver ระบบจะใช้ Module ที่เรียกว่า mapper เพื่อบริหารการทำงานของ device ที่ได้มีการติดตั้งไว้แล้ว ตัว mapper จะใช้ช่องทางพิเศษใน driver เพื่อเป็นสื่อกลางในการสั่งการให้อุปกรณ์ เหล่านั้นทำงาน ตัว Mapper จะเป็นตัวตอบสนองต่อ Application ที่ขอใช้บริการ device ที่มีอยู่บนระบบระบบจะมี mapper ต่างๆที่เป็นมาตรฐานเช่น Waveform- Audio , MIDI และ AUX device

ACM เป็นส่วนขยายระบบ Multimedia และจะถูกติดตั้งเหมือน Mapper นั่นหมายความว่า ACM จะใช้ช่องการติดต่อของ Mapper เพื่อติดต่อไปยัง Waveform - audio device ต่างๆ ACM จะทำการ deccode หรือ encode ข้อมูล Audio ต่างๆ ก่อนที่จะส่งไปให้หรือรับจาก Device driver ของ Waveform - Audio device

ข้อแตกต่างระหว่าง ACM และ Mapper มาตรฐาน คือ ACM สามารถค้นหา Waveform - audio deviceที่ให้บริการเฉพาะรูปแบบบางแบบหรือหาอุปกรณ์ Waveform - audio device มีการทำงานหลายๆ หน้าที่และ ACM จะเป็นตัว compress และ decompress ข้อมูลเพื่อให้อยู่ในรูปแบบเฉพาะที่ audio device นั้นๆ support

เมื่อ application ร้องขอบริการต่อระบบให้เปิดอุปกรณ์ Waveform- audio(Input / Output) รูปแบบของการใช้บริการนั้นๆ จะอยู่ใน format เฉพาะของ devices นั้นๆ เมื่อคำสั่งนั้นถูกส่งมาที่ Mapper และ mapper จะทำการหาอุปกรณ์ที่ support ต่อรูแบบนั้น ๆ ACM จะหาอุปกรณ์ (Waveform audio device) ที่ให้บริการต่อข้อมูลรูปแบบนั้น ถ้า ACM หาอุปกรณ์ ที่จะ support ข้อมูลที่มีรูปแบบนั้นๆไม่พบ มันจะทำการ compress และ decompress ในขณะเดียวกันก็จะหาอุปกรณ์ที่มี support รูปแบบเหล่านั้นด้วย หลังจาก ACM หา device ที่ supportรูปแบบข้อมูล ทีมันทำการแปลงได้แล้ว มันสามารถสั่งให้อุปกรณ์เหล่านั้นทำการ play หรือ record ตามรูปแบบคำสั่งที่ได้ รับมาได้ ถึงแม้ว่า device นั้นๆจะไม่ support รูปแบบเหล่านั้นโดยตรงก็ตาม นอกจาก ACM จะทำการเปลี่ยน format ของข้อมูลแล้วมันยังให้บริการทำ compression ,Decompression , filter ,format selection และ filter selection.

Audio Compression Manager ทำงานอย่างไร

ACM จะใช้ช่องสื่อสารของเดิมที่มีอยู่เพื่อให้สามารถทำงานแทนที่ mapping แบบเดิม ACM จะเป็นตัวทำหน้าที่ติดต่อกับ device ต่างๆ เมื่อเราเรียกใช้งาน ACM แล้ว ACM เองจะมีความสามารถที่จะทำงานกับข้อมูล audioได้หลายๆอย่างในเวลาเดียวกันเป็นการทำ compression และdecompression ในเวลาเดียวกัน

ACM จะบริหารหารทำงานของdriver ต่างๆ ดังนี้

- การทำ compression และ decompression

- เปลี่ยนรูปแบบของข้อมูลให้ driver เข้าใจ

- ทำ filter

การ compress และ decompress จะทำการปรับเปลี่ยนรูปแบบจากรูปแบบหนึ่งไปเป็นอีกรูปแบบหนึ่ง ตัวอย่างเช่น ทำการเปลี่ยน PCM เป็น ADPCM การเปลี่ยน format จะเปลี่ยนเฉพาะ format แต่ไม่เปลี่ยนแปลงชนิดของข้อมูล เช่น ทำการเปลี่ยน 44 - khz , 16 Bit data เป็น 44 - khz 8 Bit data filter จะไม่เปลี่ยนรูปแบบของ data ทั้งหมด แต่จะเปลี่ยนเป็นบางส่วนเท่านั้น เช่น การ filter สามารถรวมข้อมูลของ data และแยกข้อมูลออกจากกันได้

สำหรับการส่งสัญญาณ audio ออกนั้น ACM จะผ่านข้อมูลไปยัง converter เพื่อทำการconvert ทันทีที่ข้อมูลมาถึง converter จะทำการ decompress ข้อมูลแล้วส่งข้อมูลที่ทำการ Decompress แล้วไปไว้ใน ACM ใน shadow buffer แล้วตัว ACM จะทำการส่งข้อมูลที่อยู่ใน shadow buffer ให้ถึง audio driver อีกทีหนึ่ง ACM จะทำการกำหนด shadow Buffer เมื่อมันได้รับคำสั่งให้เตรียมการสำหรับการรับสัญญาน audio ACM จะส่ง shadow Buffer ที่ว่างเปล่าให้ driver และมันจะรอจน driver แจ้งว่าได้ใส่ข้อมูลลงใน shadow Buffer แล้ว ACM จึงจะส่ง buffer ไปใน driver ส่วน compress เพื่อ compress ข้อมูล หลังจากทำการ compress ข้อมูลแล้ว driver จะผ่านข้อมูลไปให้ application

ฟังก์ชั่นต่าง ๆ ของ Audio Compression Manager Function และโครงสร้าง

function ต่างๆ ของ ACM ถูกจัดรวบรวมเป็นหมวดหมู่ซึ่งทำให้ง่ายต่อการอ้างอิง โดยมีการแบ่งออกเป็น 2 ส่วน คือ ลักษณะการใช้งานและหน้าที่ของ function นั้นๆ function ที่อยู่ในกลุ่มของ filter และ format จะมีลักษณะที่เหมือนกันเกือบจะทุก function ที่ทำงานบน filter จะมี function ที่ทำงานเทียบเท่ากับการทำงานบน formats function ที่อยู่ในกลุ่มของ format บาง function จะใช้ Waveform audio format tag ในขณะที่บาง function จะต้องการรูปแบบเดิม และ function ในกลุ่มของ filter ก็เช่นเดียวกัน

Functions ที่สามารถถูกเรียกใช้โดย System.

ระบบจะมีลักษณะของ function 3 แบบแบ่งตามลักษณะการเรียกใช้งาน Call Back Function คือ function ที่ application นั้นๆทำการขอใช้บริการ

- Hook คือ procedure ที่ช่วย applicationในการทำงานในรูปแบบเฉพาะของผู้ใช้บริการ

- Driver คือ procedure ใน application นั้นๆซึ่งจะมีการ codec convert และ filter

Callback Function มีรายชื่อดังต่อไปนี้

- ACM Driver Enum Callback

- ACM Filter Enum Callback

- ACM Filter Tag Enum Callback

- ACM Format Enum Callback

- ACM Format Tag Enum Callback

- ACM Stream Convert Callback

รายชื่อ function ทั้งหมดใน ACM จะใช้ Callback function เช่นเมื่อเรียก function ACM จะทำการเรียกใช้งานผ่าน Callback function บาง function ไม่สามารถเรียกโดยตรงจาก Callback function เพราะมันจะถูกเรียกใช้โดย function ที่มีอยู่อีกทีหนึ่ง เช่น

- ACM Driver Add

- ACM Driver Priority

- ACM Driver Remove

นอกจากนี้ระบบจะเรียก function เหล่านี้เพื่อช่วยในการบริการ Application ตามความต้องการของผู้ใช้

- ACM Filter Choose Hook Proc

- ACM Format Choose Hook Proc

และ function ต่อไปจะถูกกำหนดเป็นPrototype โดยยอมให้Applicationใช้ Codec , Convert หรือ Filterตามที่ผู้ใช้ต้องการโดย Prototype นี้ จะส่งค่าผ่านACM Driver Add.

- ACM Driver Proc

2.7 ความรู้เบื้องต้นเกี่ยวกับการแปลงสัญญานเสียง

2.7.1 การส่งสัญญาณเสียงผ่านระบบเครือข่าย

ในการส่งสัญญาณเสียงไปบนระบบเครือข่ายนั้นเราจำเป็นอย่างยิ่งที่จะต้องทำการเปลี่ยนรูปแบบของสัญญาณจากสัญญาณ Analog ไปเป็นสัญญาณ Digital เพราะการส่งสัญญาณ Analog ผ่านแบบเครือข่ายนั้นมีข้อจำกัดต่าง ๆ มากมายยกตัวอย่างเช่น

- สัญญาณ Analog ที่จะถูกส่งออกไปจะต้องมีการขยายระดับของสัญญาณ โดยการขยายสัญญาณ สัญญาณที่ถูกขยายควรจะมีขนาดที่สมมูลกับสัญญาณต้นแบบแต่บางครั้งอาจเกิดความผิดพลาดจากการขยายสัญญาณ ซึ่งเกิดจากอุปกรณ์ที่ทำหน้าที่ขยายสัญญาณนั่นเอง

- ปัญหาเกี่ยวกับ Noise บนของสัญญาณ สัญญาณไฟฟ้าที่เกิดขึ้นในสาย หรือเคเบิลจะถูกสร้างโดยการไหลของอิเลคตรอนอย่างเป็นระเบียบ แต่ในขณะเดียวกัน Noise ที่เกิดจากความร้อนก็จะถูกสร้างขึ้นในสาย เช่นเดียวกันถึงแม้ว่าจะไม่ใช่สาย Cable ในการส่งสัญญาณเราก็อาจจะประสบปัญหาของ Noise (สัญญาณรบกวนได้) ซึ่งได้แก่สัญญาณแม่เหล็กโลก รังสีจากดวงดาวและดวงอาทิตย์

- สัญญาณที่ถูกส่งไปยังตัวกลางจะมีการอ่อนกำลังของสัญญาณซึ่งเกิดจากการสูญเสียและลดทอนระหว่างที่สัญญาณผ่านอยู่ในตัวกลาง ความแรงของสัญญาณที่ถูกลดทอนอาจก่อปัญหาให้กับทางด้านผู้รับ ในการตีความหมายของสัญญาณเราสามารถลดปัญหาของสัญญาณนี้ได้โดยเพิ่มขนาดสายนำของสัญญาณแต่อย่างไรก็ตามก็ไม่สามารถทอนให้หมดไปได้

ระบบสัญญาณแบบ Digital สามารถขจัดปัญหาต่างๆ ได้ โดยการแทนค่าของข้อมูลที่จะถูกส่งออกไปยังตัวกลางด้วยค่าแบบ Binary โดยสัญญาณแบบ Analog จะถูกแปลงไปเป็นกลุ่มของตัวเลขแบบ Digital ที่เรียงต่อกันและนำเลข Digital เหล่านี้ส่งผ่านทางระบบสื่อสารและเป็นที่แน่-นอนว่าสัญญาณแบบ Digital ก็ยังมีของผิดพลาดเช่นเดียวกับของผิดพลาดของสัญญาณแบบAnalog นั่นคือมีการถูกลดทอนของสัญญาณ และมีสัญญาณรบกวนแต่อย่างไรก็ตาม สัญญาณแบบ Digital (ค่าของเลข Binary) ที่ถูกสร้างขึ้นแทนสัญญาณแบบ Analog โดยการแบ่งระดับของแรงดันไฟฟ้าจากการที่สัญญาณแบบ Digital มีค่าของระดับเพียง 2 ค่านั้น ทำให้มันง่ายและสะดวกต่อการจัดการ

ขนาดและระดับของมันถ้าสัญญาณแบบ Digital ถูกทำการสุ่มในระดับความเร็วที่ยอมรับได้และระดับของแรงดันที่เหมาะสมจะทำให้สัญญาณแบบ Digital ที่ถูกส่งไปให้ผู้รับจะมีความน่าเชื่อถือและลดความผิดพลาดจะเห็นปัญหาจาก Noise จะมีน้อยกว่าสัญญาณ Analog

2.7.2 หลักการย่อขนาดข้อมูล (ทำไมต้องมีการย่อขนาดของข้อมูล)

โดยปกติโปรแกรมที่ใช้เสียงในการสื่อสารบนระบบอินเตอร์เน็ต (Audio Conferencing program) จะมีการทำงานโดยเปลี่ยนสัญญานเสียงพูดจาก Analog เป็น Digital โดยวิธีการ Digitizing แล้วทำการส่งข้อมูลที่ได้ไปบนอินเตอร์เน็ตจะเห็นว่าเป็นการทำงานง่ายๆ แต่จริง ๆ แล้วเราไม่สามารถส่งข้อมูลที่ได้รับจากการ Digitizing ไปได้ในทันที ต้องมีการย่อขนาดของข้อมูลก่อน เนื่องจากข้อจำกัดของ bandwidth ของช่องส่งข้อมูล เช่นถ้าเราจะทำการสื่อสารด้วยเสียงผ่าน Modem ความเร็ว 28.8 KBPS เราสามารถส่งและรับข้อมูลที่ไม่มีการย่อขนาดได้เพียง 3600 bytes/s เท่านั้น แต่คุณภาพของเสียงที่เราสามารถยอมรับได้ต้องการความเร็วในการส่งข้อมูลที่ 8000 byte/s มีวิธีการแก้ปัญหา 2 ทาง คือ

1. ขยาย bandwidth
2. ลดขนาดของข้อมูลก่อนทำการส่ง

2.7.3 ทฤษฎีและวิธีการในการย่อขนาดของข้อมูล

The Nyquist Criterion

การเปลี่ยนรูปแบบสัญญาณ Analog ไปยังสัญญาณ Digital จะมีพื้นฐานอยู่บนหลักการของ Nygquist คือจะต้องทำการสุ่มของสัญญาณอย่างน้อย 2 ครั้งของความถี่ที่สูงที่สุดของสัญญาณความถี่นั้นๆ จึงจะสามารถทำการสร้างรูปสัญญาณ รูปสัญญาณนั้น ๆ ให้กลับคืนเหมือนสัญญาณต้นฉบับได้

สัญญาณ F(t) สามารถที่จะปลูกสร้างกลับจากตัวอย่างที่ถูกเก็บไว้ได้ โดยใช้ Low Pass Filter ตาม ทฤษฎีแล้วค่าข้อมูลที่ได้จากการสุ่มตัวอย่างคือค่าของแอมพลิจูด ตามเวลาที่กำหนดไว้และสัญญาณที่ต่อเนื่องกันจะถูกสร้างใหม่โดยใช้ วงจร Low Pass Filter ในการใช้งานจริงกับเสียงพูด จำนวนของการสุ่มตัวอย่างนั้นไม่เพียงพอต่อคุณภาพที่ได้รับ แต่เราก็สามารถที่จะปรับเปลี่ยนได้ ในปี ค.ศ 1933 Harry Nyquist ได้กำหนดจำนวนต่ำสุดของการสุ่มตัวอย่างคือ

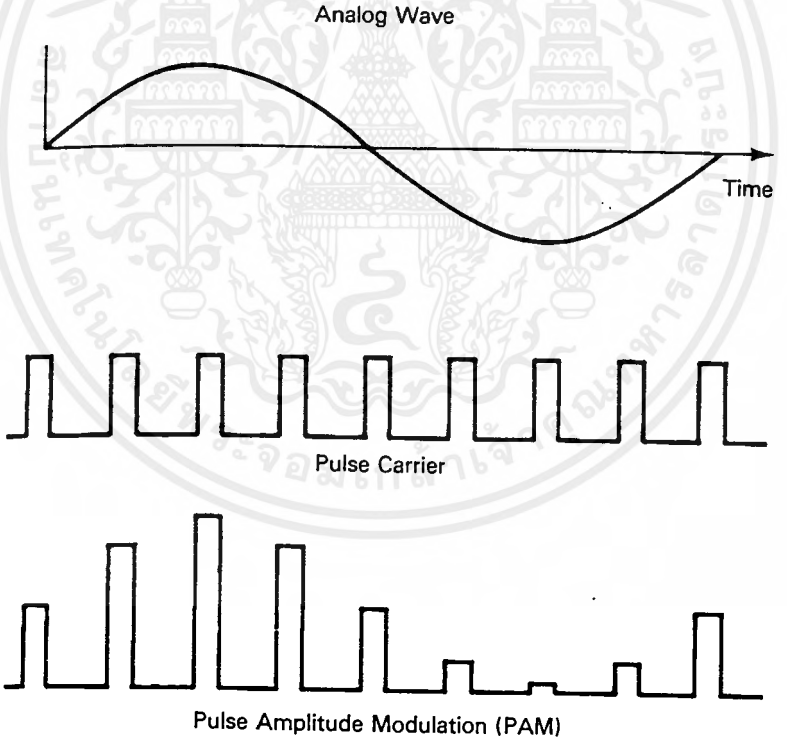
$$fs > 2\ bw$$

โดยที่ fs = ความถี่ของการสุ่มตัวอย่าง bw = ความถี่ที่สูงที่สุดของสัญญาณ input ตามทฤษฎีของ Nyquist ซึ่งแสดงถึงว่าการ Sampling จะถูกพิจารณาด้วย Pulse ที่ต่อเนื่อง และมีความถี่ที่เท่า ๆ กัน รวมถึงขนาดของ Pulse ด้วยแหล่งสัญญาณที่จะนำมาทำการ Sampling จะผ่านระบบที่เรียกว่า Band Limit Filter เพื่อจุดประสงค์ 2 ประการคือ เพื่อจำกัดความถี่ที่เกินจากความสามารถที่จะ

Sampling ได้ และเพื่อนำความถี่ที่มีค่ามากกว่า f2/2 ออกไปเพื่อป้องกันไม่ให้ความถี่ของสัญญาณรบกวนปะปนเข้ามา ค่าของจำนวนการ Sampling ที่ใช้ในงานอุตสาหกรรมคือประมาณ 8,000 ครั้งต่อวินาที ตามพื้นฐานของ Nyquist Sampling Theory ที่ว่าระดับการ Sampling ที่ยอมรับได้ของช่องสัญญาณที่มีความถี่ 4,000 Hz (ค่าของสัญญาณที่สูงกว่า 4,000 Hz จะถูกนำออกไป) การ Sampling 8,000 ครั้งต่อวินาทีนั้น มีความน่าเชื่อถือและประสิทธิภาพในการใช้งานในระบบสายโทรศัพท์สาธารณะอย่างเพียงพอ

Pulse Amplitude Modulation (PAM)

ค่าของการ Sampling จะถูกเก็บด้วยความถี่ที่กำหนดไว้นั่นคือที่ความถี่ 8,000 ครั้งต่อวินาที ขบวนการของการเก็บค่านี้จะกระทำโดยการเปลี่ยนค่าของ Pulse Carrier ด้วยค่าของ Analog Signal Pulse Carrier คือค่า Pulse ที่ใช้ในการ Sampling และสามารถที่จะเปลี่ยนแปลงไปตามขนาดของสัญญาณ Analog เรียกวิธีนี้ว่า Pulse Amplitude Modulation (PAM)



ด้วยวิธีการ PAM ค่า Amplitude ของ Pulse Carrier จะเปลี่ยนแปลงไปตามค่าของสัญญาณ Analog ที่เข้ามาดังในรูป ค่าของ Pulse จะถูกกำหนดด้วยส่วนที่เกี่ยวข้องคือค่าของเวลาและตำแหน่ง PAM จะถูกแบ่งโดยวิธีการ Modulation เพราะค่าที่ได้ออกมาของสัญญาณ Analog จะถูกรวมระหว่าง amplitude กับสัญญาณ Sampling Pulse code Modulation (PCM) สัญญาณที่

ได้จากขั้นตอน PAM ยังคงอยู่ในรูปของสัญญาณ Analog เพราะว่ามันยังมี Amplitude หลายค่า และขั้นตอนต่อไป คุณลักษณะนี้จะถูกเปลี่ยนโดยการแทนค่าของสัญญาณ ด้วยจำนวนเลขซึ่งจะแทนค่าของ Amplitude อันนั้น ค่าของเลขจำนวนจะถูกทำให้อยู่ในรูปของจำนวน binary ซึ่งมีค่าสองค่าคือ 1 และ 0 เทคนิคนี้เราเรียกว่า Pulse code Modulation (PCM) วิธีการเฉพาะที่จะใช้ในการกำหนดค่าของแต่ละครั้งจากการ Sampling จะถูกเรียกว่า Quantizing และแต่ละครั้งของ PAM จะมีการประมาณค่าโดยใช้เลข Binary จำนวนเป็น Bit เพื่อแทนค่า

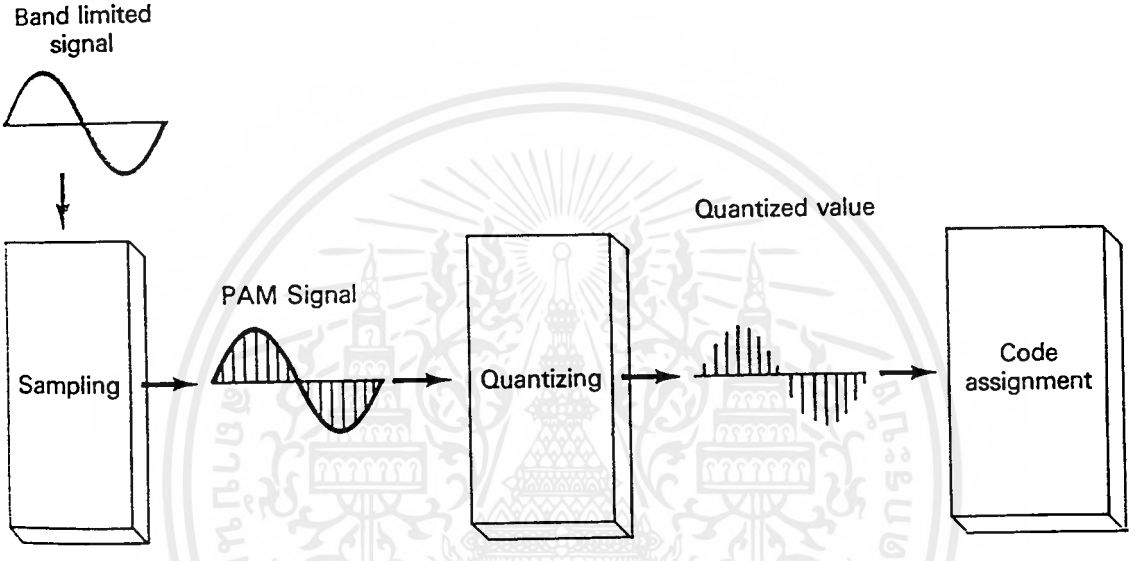
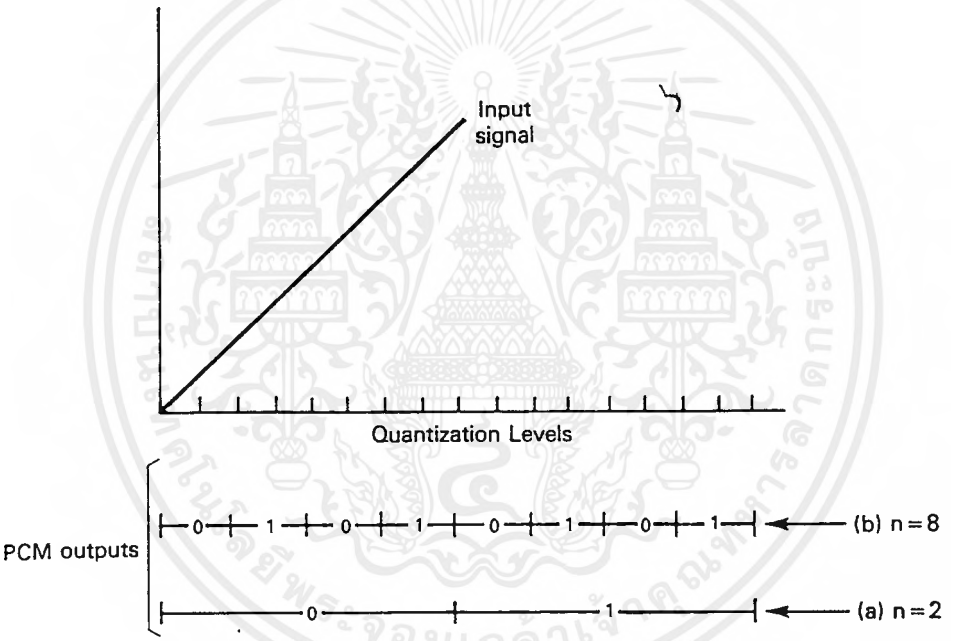


FIGURE 3-18. Pulse Code Modulation

การทำ Quantizing แบบดั้งเดิมจะนำมาซึ่งข้อผิดพลาดเพราะค่าของการแทนขนาดของการ Sampling แต่ละค่ามีความแตกต่างกัน แต่อย่างไรก็ตามการ Quantizing 1 ครั้งสัญญาณจะถูกหน่วงให้ช้าลงในทุก ๆ ช่วงโดยปราศจากการลดทอนของสัญญาณโดยการควบคุมจำนวนขั้นของการ Quantizing

จำนวนของค่า n นั้นก็ค่อนข้างจะสำคัญ (ในรูป A กำหนดให้ความสัมพันธ์แบบคงที่ระหว่างสัญญาณภาพ PAM กับ PCM) ค่าของ n จะถูกระบบแบ่งให้มีขนาดครึ่งหนึ่งของAmplitude ของ PAM หรือในอีกทางหนึ่งคือค่าของจำนวน Bit จะถูกแบ่งโดยขอบเขตของ 2n เช่น ในรูป b การใช้ค่า 8bit ในการแทนจะมีความเที่ยงตรงน่าเชื่อถือมากกว่า

ระบบ PCM แบบเก่าจะใช้จำนวน bit เท่ากับ 7bit แต่ในระบบใหม่จะใช้ค่า n เท่ากับ 8 ถ้าการ Quantize กำหนดจำนวนของขั้น ออกเป็น 128 ขั้น เราจะใช้ค่า n เท่ากับ 7bit ในการแทนค่าแต่ละครั้งของการ Sampling (2^7 เท่ากับ 128) ถ้าเราใช้จำนวนขั้นของการทำ Quantize เท่ากับ 256 ระดับเราต้องใช้ถึง 8bit ในการแทนค่าที่ได้มา (2^8 เท่ากับ 256) นอกจากนี้ถ้าเราจะใช้การ Quantize ที่ 128 ระดับ เราต้องการส่งข้อมูลเท่ากับ 56,000 bit/วินาที (8,000 x 7 เท่ากับ 56,000) และถ้าต้องการให้มีจำนวนของการ Quantize ที่ 256 ระดับ เราต้องส่งข้อมูลมีปริมาณถึง 64,000 bit ต่อวินาที (8,000 x 8 เท่ากับ 64,000) จำนวน bit ของข้อมูลที่มากขนาดนี้ต้องการช่องสัญญาณที่ส่งข้อมูลได้ดี แต่ในระบบของ Modem เราต้องทำการลดจำนวนของข้อมูลจึงจะสามารถส่งของมูลได้ เนื่องจาก Modem มีความเร็วในการส่งข้อมูลไม่มากพอ



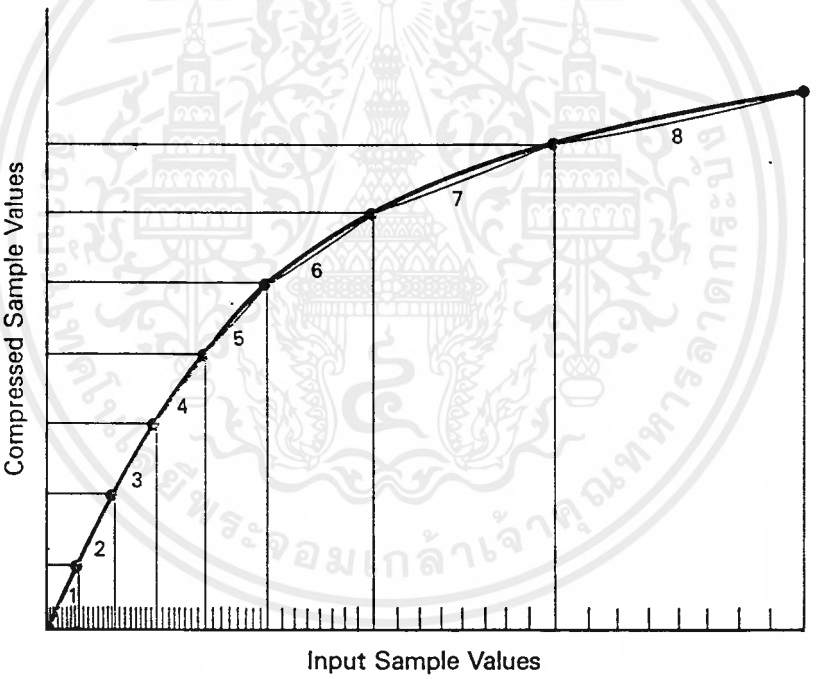
Sampling และ Quantizing Error

การส่งสัญญาณแบบ Digital ใช่ว่าจะขจัดปัญหาได้ สัญญาณ Digital สามารถถูกทอนสัญญาณได้จากหลาย ๆ ทาง คือความถี่ในการ Sampling ปัญหานี้สามารถแก้ไขได้โดย เพิ่มความถี่ของการ Sampling แต่แน่นอนมันต้องการอุปกรณ์ที่มีราคาแพง เพื่อรองรับกับจำนวนข้อมูลที่เพิ่มขึ้นและไม่มีเทคนิคใดที่สามารถขจัดปัญหาและถูกลดทอนสัญญาณให้หมดไปได้ ส่วนปัญหาที่ 2 คือ ปัญหาที่เกิดจาก Quantizing Error คือขบวนการที่การ Quantizing ไม่สามารถแทนค่าที่ได้มาจาก PAM ได้อย่างแม่นยำ มันเป็นเพียงค่าประมาณเท่านั้น ผลที่ตามมาคือสัญญาณที่จะถูกสร้าง

กลับไม่เหมือนกับสัญญาณต้นฉบับ ปัญหานี้เราสามารถแก้ไขได้โดยการเพิ่มจำนวน Step ในการทำ Quantizing แต่การเพิ่มจำนวน Step นั้น ก็นำมาซึ่งปริมาณข้อมูลที่เพิ่มขึ้น

Companding

เทคนิคที่สามารถลดขนาด Amplitude ลงเพื่อลดจำนวนของระดับการ Quantizing หรือในอีกทางหนึ่งคือแอมพลิจูดที่มีขนาดเล็ก จะถูกขยายเป็นการเพิ่มจำนวนของระดับการ Quantize และลดจำนวนของระดับการ Quantize โดยรวมลง หลังจากที่สัญญาณถูกตีความ มันจะกลับไปอยู่ในรูปแบบเดิม การผสมผสานระหว่างการบีบอัด (Compression) และการขยายช่วง (Expanding) จึงเรียกวิธีนี้ว่า Companding การ Quantize สัญญาณที่มีความสูงมากจะถูกกำหนดให้มีระดับการ Quantize ต่ำ(จำนวนขั้นของการทำ Quantize น้อย) แต่การ quantize สัญญาณที่มีความสูงน้อยจะถูกกำหนดให้มีการ Quantize มาก(จำนวนขั้นของการทำ Quantize มาก) ตามความเหมาะสม



1: Slope = 1
2: Slope = ½
3: Slope = ¼
4: Slope = ⅛
5: Slope = 1/16
6: Slope = 1/32
7: Slope = 1/64
8: Slope = 1/128

A-law และ Mu-law

ขบวนการเข้ารหัสจะนำวิธีการ companding มาใช้เพื่อสร้างค่าที่มีความใกล้เคียงกันสำหรับสัญญาณเสียงระดับจะถูกกำหนดความสำคัญตามกฎของ Mu-law (ใช้ในอเมริกาเหนือและญี่ปุ่น) และ A-law ใช้ใน (ยุโรป) กฎของ A-law และ Mu-law ค่อนข้างที่จะเหมือนกันแตกต่างที่ A-law จะให้ความสำคัญแบบต่อเนื่องในขณะ Amplitude ที่เล็ก ๆขนาดช่องที่เล็กที่สุดคือ 2/4096 สำหรับ A-law และ 2/8159 สำหรับ Mu-law

การ Companding แบบไม่ต่อเนื่องโดยปรกติแล้วจะถูกสร้างอย่างเป็นขั้นตอนสำหรับ Mu-law ค่า Mu=255 จะถูกใช้ และค่าของ Companding Value จะถูกแปลงให้อยู่ในรูปของเลขจำนวน (โดยปกติ bit ที่ 8 แสดงถึงค่าว่าเป็นบวกหรือลบ)

Quantization Code (Q)		*Segment(s)*							
		000	*001*	*010*	*011*	*100*	*101*	*110*	*111*
0000	0	0	31	95	223	479	991	2015	4063
0001	1	1	35	103	239	511	1055	2143	4319
0010	2	3	39	111	255	543	1119	2271	4575
0011	3	5	43	119	271	575	1183	2399	4831
0100	4	7	47	127	287	607	1247	2527	5087
0101	5	9	51	135	303	639	1311	2655	5343
0110	6	11	55	143	319	671	1375	2783	5599
0111	7	13	59	151	335	703	1439	2911	5855
1000	8	15	63	159	351	735	1503	3039	6111
1001	9	17	67	167	367	767	1567	3167	6367
1010	10	19	71	175	383	799	1631	3295	6623
1011	11	21	75	183	399	831	1695	3423	6879
1100	12	23	79	191	415	863	1759	3551	7135
1101	13	25	83	199	431	895	1823	3679	7391
1110	14	27	87	207	447	927	1887	3807	7647
1111	15	29	91	215	463	959	1951	3935	7903
		31	95	223	479	991	2015	4063	8159

ตารางของ Mu=255 PCM 1 Bit จะใช้ในการบอกขั้วซึ่ง 0 = positive และ 1 = negative อีก 3 Bit จะบอกว่าอยู่ที่ส่วนใด อีก 4 Bit ที่เหลือบอก Step ของ Quantize

Linear Predictive Coding (LPC)

เป็นระบบที่ทำการบีบอัดข้อมูลโดยใช้การทำนายค่าต่อเนื่องเพื่อที่จะลดจำนวนข้อมูลซึ่งวิธีการนี้ประสบผลสำเร็จในการลดขนาดของข้อมูลได้อย่างมาก เมื่อเทียบกับวิธีการบีบอัดข้อมูลวิธีอื่น ๆ แต่วิธีการบีบอัดข้อมูลวิธีนี้ก็ต้องการการประมวลผลข้อมูลอย่างมาก เหมือนอย่างเช่นวิธีการของ GSM LPC ต้องการการคำนวณอย่างมากเพื่อที่จะให้ได้ Out Put ออกมา โดยใช้การคำนวณแบบ Floating Point ถ้าเครื่องคอมพิวเตอร์ของคุณไม่มี Math Coprocessor เป็นที่แน่นอนว่า เกือบจะไม่สามารถใช้วิธีการบีบอัดข้อมูลแบบ PLC ในการทำงานแบบ Real Time ได้วิธีการบีบอัดข้อมูลแบบ LPC มีความไวต่อสัญญาณรบกวนที่มีความถี่สูง และการตัดสัญญาณที่มีระดับของสัญญาณ Input สูง ๆ ถ้าคุณได้ยินเสียงแตกซ่าบ่อย ๆ คุณต้องลดกำลังขยายของไมโครโฟนลงหรือพูดให้ห่างจากไมโครโฟนมากขึ้น หลีกเลี่ยงการพูดไปยังไมโครโฟนโดยตรง และถ้าเรานำวิธี LPC ไปใช้กับเสียงที่มีระดับความดังมาก ๆ อาจจะไม่สามารถใช้วิธีการนี้ได้ เพราะสัญญาณที่มีความถี่สูง ๆ จะขาดหายไป ถ้าเปรียบเทียบวิธีการบีบอัดข้อมูลแบบ GSM เป็น Cellular Phone LPC ก็เหมือนกับ Short Wave Radio มันจะทำงานได้ไม่สม่ำเสมอ คุณต้องใช้ความระมัดระวังในการใช้เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด ซึ่งในการทำงานจริงมักจะมี Noise และถูกลดทอนสัญญาณบ้าง แต่มันก็เหมือนกับ Short Wave คือมันจะยังคงทำให้คุณสามารถส่งเสียงได้ แม้ว่าจะไม่มีวิธีการอันใดทำได้ ในเครือข่ายของคุณที่ทำงานช้า วิธีนี้ก็เป็นวิธีที่น่าลองเอาไปใช้

LPC-10

เป็นการบีบอัดข้อมูลที่แตกต่างจาก LPC มันถูกกำหนดมาตรฐานโดยสำนักงานควบคุมคุณภาพของสหรัฐอเมริกา เลขที่ 1015/NATOSTANAG-4198 ถูกตีพิมพ์เผยแพร่ วิธีการบีบอัดข้อมูลแบบ LPC-10 นี้ จะใช้ในการเข้ารหัสสัญญาณเสียงแบบ Real Time โดยสามารถลดขนาดของข้อมูลให้เหลือ 2400 Bits/วินาที วิธีการบีบอัดข้อมูลแบบ LPC-10 สามารถบีบอัดข้อมูลเสียงได้มาก เหลือข้อมูลที่จะถูกส่งไปยังเครือข่ายเพียง 346 Byte/วินาที จะเห็นได้ว่าอัตราส่วนในการบีบอัดข้อมูลมีมากถึง 26 ต่อ 1 ความเที่ยงตรงของเสียงเมื่อผ่านขบวนการบีบอัดโดยวิธี LPC-10 แล้วจะมีคุณภาพด้อยกว่าวิธีการของ GSM แต่มันก็เพียงพอสำหรับการติดต่อสื่อสารโดยใช้เสียง การบีบอัดข้อมูลแบบ LPC-10 เราควรหลีกเลี่ยงสัญญาณเสียงที่สูงเกินไปและดังเกินไป และควรกำจัดสัญญาณเสียงฮัมและ Noise ที่เข้ามาขัดขวางขบวนการบีบอัดข้อมูลให้ออกไปเสียก่อน ข้อเสียของการบีบอัดข้อมูลแบบ LPC-10 นี้คือ มันต้องการการคำนวณอย่างมหาศาล และการคำนวณส่วนใหญ่ยังใช้ Floating Point อีกด้วย เพราะฉะนั้นเป็นที่แน่นอนว่ามันต้องใช้ Math Professor

CELP

เพื่อที่จะได้มาซึ่งวิธีการบีบอัดข้อมูลที่มีลักษณะการใช้งานกับเสียงพูดโดยเฉพาะ สมมุติฐานหลักซึ่งรู้จักกันดีทางด้านการสร้างสัญญาณเสียงพูด ได้ถูกนำมาใช้โดยสัญญาณเสียงจะถูกส่งผ่านตัวกรองเพื่อทำการผลิตเสียงขึ้นมา วิธีการที่ง่ายที่สุดรู้จักกันในชื่อ LPC ที่ทุก ๆ กลุ่มของเสียงจะถูกคำนวณและกรองเพื่อสร้างเสียงตามขนาดและความถี่ที่ได้ตกลงกันไว้ ตัวแปรสัญญาณสำหรับเสียงพูดที่ตีความชุดของข้อมูลเสียงพูดปรกติ จะถูกส่งผ่านเข้าไปยังชุดสร้างรูปสัญญาณ และมีการเพิ่มขนาดเพื่อให้เป็นเสียงพูด ระบบนี้จะเป็นระบบที่มีประสิทธิภาพและนิยมใช้กันจะมีปริมาณของเสียงพูดอยู่ที่ระดับ 1200-2400 Bps ด้วยการคาดคะเนเสียงแบบใช้ Vector สามารถลดจำนวนของข้อมูลลงได้เหลือ 300-600 Bps แต่มันก็มีข้อเสียคือ ขาดความเป็นธรรมชาติ กลุ่มของวิธีการ CELP นี้จะทดแทนคุณสมบัติในเรื่องการขาดหายของคุณภาพสัญญาณเสียงตามวิธีการของ LPC

G.728 Low Delay CELP Codec

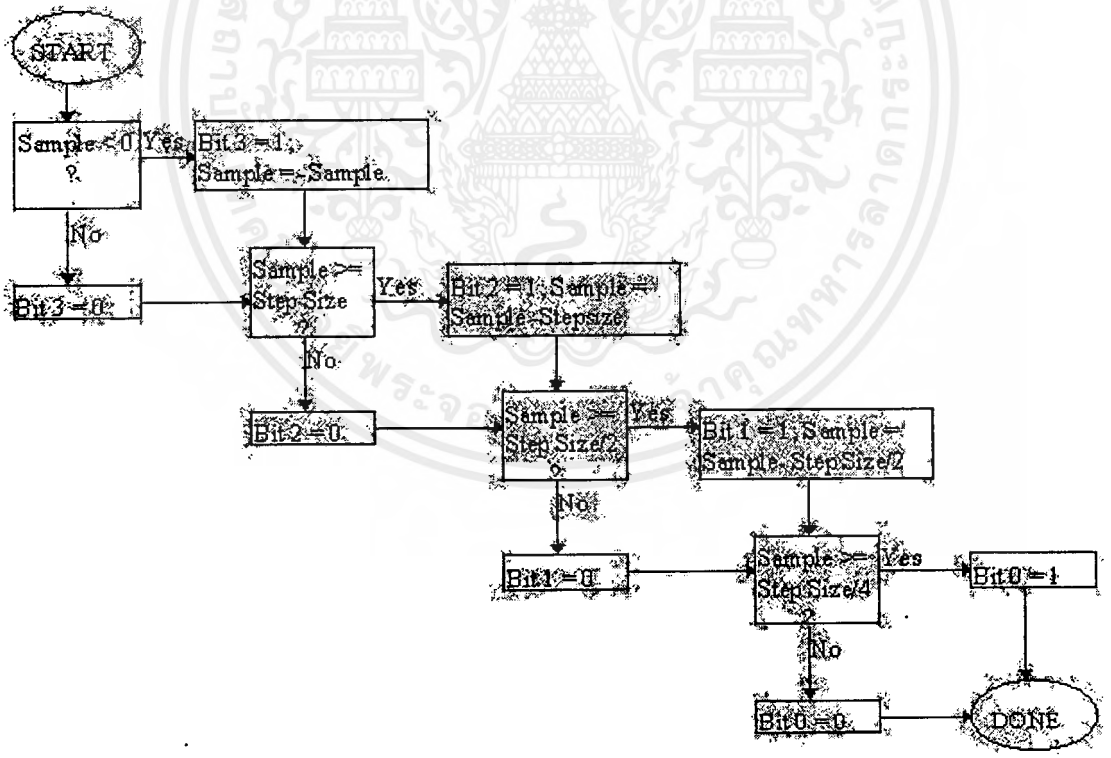
ที่จำนวนของข้อมูลประมาณ 16 KBITS/วินาที คุณภาพของสัญญาณจะตกลงอย่างมาก ดังนั้นอัตราของการเข้าและถอดรหัส โดยเฉพาะอย่างยิ่ง ตามวิธีการของ CELP ที่ถูกดัดแปลงมามีแนวโน้มที่จะนำมาใช้ได้ แต่อย่างไรก็ตามเนื่องจากการตัดสินใจล่วงหน้าของส่วน short term ที่ใช้ในการเข้าและถอดรหัสมักเกิดความล่าช้า ความล่าช้าในการเข้าและถอดรหัสนี้จะถูกกำหนดโดยเวลาเมื่อสัญญาณเสียงพูดมาถึง ส่วนของมัน (ชุด Codec) ชุด Codec จะตอบสนองต่อตัวอย่างสัญญาณเสียงที่ได้รับ โดยสร้าง Out Put ออกมา ปริมาณของข้อมูลที่เกิดจากการเข้ารหัสจะถูกส่งไปยังตัวถอดรหัส (Decoder) โดยปรกติการเข้าและถอดรหัสเสียงพูด จะใช้เวลาอยู่ในช่วง 50-100 มิลลิวินาที และความล่าช้าที่เกิดขึ้นจะเป็นปัญหาสำคัญ ดังนั้นในปี ค.ศ.1988 CCITT ได้กำหนดมาตรฐานที่ปริมาณของข้อมูลเท่ากับ 16 Kbits /วินาที ออกมาเป็นมาตรฐาน ความต้องการส่วนใหญ่ของการเข้าและถอดรหัสข้อมูลควรจะมีคุณภาพเทียบได้กับ G.721 32 Kbits /วินาที ADPCM และควรมีความล่าช้าน้อยกว่า 5 มิลลิวินาที และทางอุดมคติควรน้อยกว่า 2 มิลลิวินาที ความต้องการของ CCITT นี้ ได้ถูกบรรจุอยู่ในวิธีการเข้ารหัสแบบ CELP ซึ่งวิธีการนี้ได้ถูกพัฒนาต่อโดย AT & T Bell Lab และถูกตั้งเป็นมาตรฐานในปี ค.ศ.1992 ได้ชื่อว่า G.728

วิธีการเข้าและถอดรหัสวิธีนี้จะใช้การคำนวณค่าที่เคยมีอยู่ เพื่อสร้างส่วนของ Short Term Filter ที่มีประสิทธิภาพ นั่นหมายความว่า ต้องใช้ Buffer ที่สามารถเก็บข้อมูลได้มากกว่า 20 มิลลิวินาที หรือมากกว่านั้น สัญญาณเสียงที่เข้ามาจะถูกคำนวณค่าการ Filter ที่เหมาะสมจากสัญญาณเสียงที่ถูกสร้างไปแล้ว นั่นหมายความว่า การเข้าและถอดรหัสโดยวิธีนี้สามารถใช้ความยาวของ Frame ข้อมูลได้สั้นกว่าวิธีการ CELP ที่มีอยู่เดิมได้ และ G.728 จะใช้ความยาวของ Frame ที่มีค่าเพียง 5 ตัว และมีความล่าช้าเพียง 2 มิลลิวินาที ที่ปริมาณความต้องการสูง ๆ การคาดคะเนแบบสั้น

จะถูกนำมาใช้ ฉะนั้น เราจึงไม่มีความจำเป็นที่จะต้องใช้การคาดคะเนแบบ Long Term ดังนั้น ข้อมูลจำนวน 10 Bits จะถูกนำมาใช้สำหรับค่า 5 ตัวที่ 16 KBPS โดยใช้แทนวิธีการของ Fixed Code Book โดยใน 10 Bits นี้ 7 Bits จะถูกนำมาใช้เพื่อแทนข้อมูลแบบ Fixed Code Book และอีก 3 Bits จะถูกใช้แทนขนาด และการใช้ค่าที่เคยเก็บไว้ เพื่อช่วยในการเทียบระดับของ Quantize และที่ขบวนการถอดรหัสจะใช้ค่า Filter แบบเดิม เพื่อเพิ่มคุณภาพของสัญญาณเสียงและทั้งหมดจะนำสู่ปริมาณข้อมูลเข้าและถอดรหัสที่ 16 BPS และมีความล่าช้าน้อยกว่า 2 มิลิวินาที นอกจากนี้คุณภาพของเสียงพูดจะเท่ากับ G.721

IMA ADPCM COMPRESSION

การบีบอัดข้อมูลโดยวิธี IMA ADPCM นี้ จะทำการคำนวณความแตกต่างระหว่างค่าของการสุ่มปัจจุบัน X[N] กลับค่าการทำนายของการสุ่มครั้งก่อน Xt[N-1] และจะใช้ขนาดความแตกต่างนี้ในการคำนวณระดับของการ Quantize โดยใช้วิธีการของ IMA ALGORITHM แต่ละครั้งของการ Quantize ระดับของ Quantize จะถูกแทนด้วยจำนวนเลข 4 Bits และ Bits ที่ 4 จะเป็น Bit เครื่องหมาย



จาก Diagram แสดงขบวนการ Quantization มันสามารถแสดงให้เห็นถึงผลลัพธ์ของความแตกต่าง และระยะห่างของ X [X] และ Xt[N-1] ซึ่งคือระยะห่างของแต่ละ Step ในปัจจุบันนั้นเอง

ระยะห่างที่กล่าวถึงนี้จะถูกแทนด้วยค่า Bit ที่ 3 ขบวนการจัดการแบ่งขนาด Step ออกเป็นครึ่งหนึ่ง และทำการเปรียบเทียบอีกครั้งกับระดับความแตกต่างจริงที่พบ ถ้าระดับความแตกต่างไม่เท่ากัน Bit ที่ 2 จะถูกกำหนดเป็น 1 ถ้าค่าของการ Sampling เท่ากับค่าของ Step Size ตัวเข้ารหัสจะทำการแบ่ง Step Size ออกเป็นครึ่งหนึ่ง แล้วทำการเปรียบเทียบอีก และถ้ามันแตกต่างกันอีก Bit ที่ 1 จะถูกกำหนดให้เป็น 1 ดังนั้น 3 Bits นี้จะแทนระดับการ Quantize ที่แสดงดังรูปล่าง

ตารางที่ 1 Table Lookup for IMA ADPCM Quantizer Adaption

Three Bits Quantized Magnitude	**Index Adjustment**
000	-1
001	-1
010	-1
011	-1
100	2
101	4
110	6
111	8

ในตารางบน ค่าของ Index จะถูกปรับโดยการให้ค่าไปเก็บไว้ใน Index แต่ละตัว และช่วงเลขจะอยู่ในช่วง 0-88 ซึ่งจะถูกใช้งานดังตารางที่ 2 ถ้าผลลัพธ์ที่ได้จากตารางที่ 2 คือ Step Size อันใหม่ ค่าของระดับ Quantization Xt[n] จะถูกเก็บไว้โดยการเก็บค่า 4 Bits ดังนั้น 1 ตัวอักษร (8 Bits) สามารถแทนค่าระดับได้ 2 ค่า

ตัวถอดรหัส ADPCM จะทำการสร้างสัญญาณเสียง Xt[n] โดยการใส่ค่าของการถอดรหัสครั้งก่อน Xt[n-1] เพื่อที่จะได้รับผลลัพธ์ของขั้วและขนาด C[n] และระดับของ Quantizer

Xt[n] = Xt[n-1]+ Step-Size [n]x C[n]

ค่าของ Step Size [n] คือค่าประมาณของผลลัพธ์ของค่าครั้งก่อน (Step Size [n-1] และ Function ของกาคำนวณหาขนาด F(C[n-1])

Step-size[n]=step-size[n-1] X F(C[n-1]

ตารางที่ 2

Index	Step Size	Index	Step Size	Index	Step Size	Index	Step Size
0	7	22	60	44	494	66	4,026
1	8	23	66	45	544	67	4,428
2	9	24	73	46	598	68	4,871
3	10	25	80	47	658	69	5,358
4	11	26	88	48	724	70	5,894
5	12	27	97	49	796	71	6,484
6	13	28	107	50	876	72	7,132
7	14	29	118	51	963	73	7,845
8	16	30	130	52	1,060	74	8,630
9	17	31	143	53	1,166	75	9,493
10	19	32	157	54	1,282	76	10,442
11	21	33	173	55	1,411	77	11,487
12	23	34	190	56	1,552	78	12,635
13	25	35	209	57	1,707	79	13,8999
14	28	36	230	58	1,878	80	15,289
15	31	37	253	59	2,066	81	16,818
16	34	38	279	60	2,272	82	18,500
17	37	39	307	61	2,499	83	20,350
18	41	40	337	62	2,749	84	22,358
19	45	41	371	63	3,024	85	34,633
20	50	42	408	64	3,327	86	27,086
21	55	43	449	65	3,660	87	29,794
						88	32,767

True Speech

True Speech 6.3/5.3 หรืออีกชื่อหนึ่งคือ True Speech G.723 เป็นสมาชิกใหม่ของกลุ่ม True Speech ซึ่งเป็นวิธีการบีบอัดข้อมูลเสียง ถูกพัฒนาโดย บริษัท DSP GROUP INC. มีอัตราบีบอัดข้อมูลอยู่ในช่วง 20 ต่อ 1 และ 24 ต่อ 1 (6.3 KBPS และ 5.3 KBPS ตามลำดับ) True Speech

6.3/5.3 หรือ G.723 และได้เป็นมาตรฐานของ International telecom communication Union (ITU) ที่ได้กำหนดไว้ G.723 ได้ถูกนำมาใช้เป็นมาตรฐานของการสื่อสารด้วยภาพและเสียงผ่านเครือข่ายโทรศัพท์สาธารณะ และเป็นส่วนหนึ่งของ ITU H.324 ซึ่งเป็นมาตรฐานในการสื่อสารสัญญาณเสียงและสัญญาณภาพที่กำลังเป็นมาตรฐานใหม่ในระบบ INTERNET

Name	Data Rate	Compression Factor
True Speech 8.5	8.5 kBps	15:1
True Speech 6.3	6.3 kBps	20:1
True Speech 5.3	5.3 kBps	24:1
True Speech 4.8	4.8 kBps	27:1

คุณภาพของสัญญาณเสียงถูกทดสอบโดยการทดสอบที่เรียกว่า MEAN OPINION SCORE (MOS) ซึ่งใช้ค่าตัวเลขบอกถึงคุณภาพในระดับ Scale ที่สูงที่สุด แสดงถึงคุณภาพของเสียงที่ใช้ในสายโทรศัพท์ธรรมดา จากการทดสอบ True Speech G.723 ได้รับค่าถึง 3.98 ดังนั้น Ture Speech G.723 จึงสามารถสื่อสารด้วยเสียงผ่านระบบ Internet ได้ดี

GSM Compression

วิธีการบีบอัดข้อมูลแบบ GSM ได้ถูกพัฒนาขึ้นที่วิทยาลัยเทคนิคของเบอร์ลินในปี ค.ศ. 1992 วิธีการบีบอัดแบบ GSM นี้เป็นวิธีการหนึ่งซึ่งมีความซับซ้อนในการบีบอัดข้อมูลเป็นอย่างมาก ซึ่งผลที่ได้รับคืออัตราการบีบอัดข้อมูลมีมากถึง 10 ต่อ 1 การทำงานของ GSM Compression มีพื้นฐานแนวความคิดมาจาก Global System for Mobile Telecommunication Protocol ซึ่ง Protocol นี้เป็นที่นิยมอย่างมากในการสื่อสารของระบบ Digital Cellular phone

Input ของ GSM ประกอบด้วย Frame ข้อมูลขนาด 160 ตัวแบบมีเครื่องหมายซึ่งได้มาจากวิธีการ PCM แบบที่ได้ข้อมูล 13 Bits ต่อการ Sampling 1 ครั้งซึ่งความถี่ในการ Sampling คือ 8000 ครั้งต่อ 1 วินาที GSM จะมีวิธีการแปล Frame ข้อมูลของ PCM ไปเป็น GSM และสามารถทำการแปลงกลับได้ ตัวเข้ารหัสจะทำการบีบข้อมูลขนาด 160 ตัว แต่ละตัวมีขนาด 16 Bits ให้เป็น Frame ข้อมูลขนาด 260 Bits ของ GSM จะเห็นได้ว่าในเวลา 1 วินาที เราสามารถลดขนาดจำนวนของข้อมูลลงเหลือเพียง 1625 Bytes เท่านั้น ดังนั้น ถ้าเราทำการบีบอัดข้อมูลแบบ GSM ลงในหน่วยความจำขนาด 1 MB. เราจะสามารถเก็บข้อมูลได้นานถึง 10 นาทีทีเดียว

ข้อมูล 16 Bit/การ Sample 1 ครั้งต้องการ

(264 Bits X 8000 Samples/Sec.)/160 Sample = 13.2 Kbits/second

บทที่ 3

การออกแบบ

3.1 หลักในการออกแบบ

จุดมุ่งหมายของโปรแกรม NetWork Voice Chat ก็เพื่อที่จะให้ผู้ใช้งานคอมพิวเตอร์ที่มีการติดตั้งการ์ดเสียงและต่อคอมพิวเตอร์เข้ากับระบบเครือข่ายสามารถที่จะสื่อสารด้วยเสียงโดยผ่านทางระบบเน็ตเวิร์กที่มีอยู่ ซึ่งจะสามารถใช้งานระบบเน็ตเวิร์กที่มีอยู่ให้เกิดประสิทธิภาพ และเป็นที่แน่นอนในการสื่อสารผ่านระบบเน็ตเวิร์กจำเป็นต้องมีการใช้โปรโตคอลซึ่งเปรียบเสมือนกับภาษาที่ใช้ในการพูดคุยระหว่างเครื่องคอมพิวเตอร์ที่ต่ออยู่ในระบบเน็ตเวิร์ก ซึ่งก็มีอยู่ใช้งานกันหลายโปรโตคอล อาทิเช่น IPX/SPX, NetBeui ,TCP/IP ฯลฯ และเพื่อให้การใช้งานโปรแกรม NetWork Voice Chat สามารถที่จะใช้ได้อย่างกว้างขวางและทั่วไปเราจึงทำการออกแบบโปรแกรมโดยเลือกใช้โปร-โตคอล TCP/IP (Transmission Control Protocol/Internet Protocol)

การออกแบบตัวโปรแกรมจะแบ่งออกเป็น 3 ส่วนสำคัญ ๆ คือ

1. การรับและส่งข้อมูลผ่านระบบเน็ตเวิร์ก
2. การจัดการเกี่ยวกับเสียง
3. การจัดการเกี่ยวกับ Compression ของเสียง

ซึ่งทั้ง 3 ส่วนหลักจะกล่าวต่อไปในรายละเอียดด้านหลัง ในการออกแบบของทั้ง 3 ส่วนหลักจะพยายามเลือกใช้รูปแบบการทำงานที่ง่ายสำหรับการพัฒนาเพื่อที่จะสามารถทำการปรับปรุงและแก้ไขได้อย่างรวดเร็ว ดังนั้นเราจึงทำการพัฒนาโปรแกรมขึ้นมาโดยอาศัยหลักการของ Object ซึ่งหลักการของ Object จะทำให้เราสามารถที่จะแยกโปรแกรมแต่ละส่วนเป็นอิสระออกจากกัน แต่แต่ละ Object ก็ยังคงมีความสัมพันธ์กัน

หลังจากที่เราได้คิดออกแบบหลักการทำงานของโปรแกรมแล้วเราก็หันมาคิดว่า NetWork Voice Chat ว่าจะต้องมีความสามารถอย่างไรบ้าง (ความสามารถพื้นฐาน) ซึ่งเราได้วางความสามารถของโปรแกรม NetWork Voice Chat ไว้ดังนี้

1. สามารถทำการส่งเสียงพูดจากเครื่องหนึ่งที่เรียกว่าผู้ส่งไปแสดงเสียงออกเครื่องคอมพิวเตอร์อีกเครื่อง หนึ่งที่เรียกว่าผู้รับ

2. สามารถรับเสียงที่ส่งมาจากเครื่องคอมพิวเตอร์ที่เรียกว่าผู้ส่งแสดงออกลำโพงของเครื่องตนได้

3. สามารถที่จะยกเลิกการรับและส่งเสียงได้

4. สามารถที่จะทำการ Compression เสียงได้

การทำงานของโครงงาน

ภาพแสดงขั้นตอนของการทำงานในส่วนของการรับเสียงและเข้ารหัสเสียงก่อนส่งออกไป

3.2 ขั้นตอนในการพัฒนา

ในการพัฒนาเราได้เลือกใช้ Visual Basic 4.0 เป็น Tools ในการพัฒนา ซึ่งจากการศึกษาและใช้งานพบว่าเราสามารถที่จะเขียนโปรแกรมในลักษณะของ object โดยใช้ Visual Basic 4.0 ได้ และขั้นตอนในการเขียนค่อนข้างจะสามารถเข้าใจและนำไปใช้งานรวมทั้งการแก้ไขได้อย่างสะดวกและรวดเร็ว

3.2.1 การพัฒนาในส่วนของการรับและส่งข้อมูลผ่านระบบเน็ตเวิร์ก (NetWork Connection)

ดังที่ได้กล่าวข้างต้นแล้วว่าเราได้เลือกใช้โปรโตคอล TCP/IP เป็นโปรโตคอลในการใช้รับและส่งข้อมูลในระบบเน็ตเวิร์ก ดังนั้นเราจึงจำเป็นจะต้องเขียนโค้ดซึ่งอยู่ในรูปแบบของ Visual Basic 4.0 ขึ้นมาเพื่อใช้เป็นตัวจัดการรับและส่งข้อมูลโดยในการใช้ Visual Basic 4.0 เขียนโปรแกรมเพื่อรับและส่งข้อมูลโดยการใช้โปรโตคอล TCP/IP นั่นเราสามารถที่จะทำได้ 2 วิธี คือ

1.การใช้ไฟล์ Winsock.bas ซึ่งข้างในจะประกาศคำสั่งในการเรียกใช้ฟังก์ชั่นใน Winsock.dll โดยการประกาศนี้จะประกาศออกมาให้อยู่ในรูปของตัวแปรที่ Visual Basic 4.0 สนับสนุน และในการเรียกใช้ก็จะกระทำเหมือนกับการ Call Function DLL จาก API ของ Windows โดยทั่ว ๆ ไป

2.การใช้ Control Winsock ซึ่งเป็น control ในชุดของ Active X Internet Control Pack ซึ่งทาง Microsoft ได้แจกมาให้ผู้พัฒนาโปรแกรมที่เกี่ยวกับ Internet ด้วย Visual Basic 4.0 ได้ใช้กันอย่างง่าย โดยวิธีนี้จะช่วยให้ผู้พัฒนาแอพพลิเคชันที่เกี่ยวกับการเชื่อมต่อเข้าสู่เครือข่าย Internet ไม่ต้องเรียนรู้ถึงการ Control การใช้ Windows Socket อย่างลึกซึ้ง ซึ่งจะช่วยประหยัดเวลาเป็นอย่างมาก

3.2.2 การพัฒนาในส่วนของการจัดการเรื่องเสียงบน Windows

ในการจัดการเรื่องเสียงบน Windows โดยการติดต่อกับการ์ดเสียงโดยตรงนั้นทำได้ยากมาก อีกทั้งยังเป็นการเสี่ยงต่อการ Crash ของระบบอีกด้วย ดังนั้นในการจัดการเรื่องเกี่ยวกับเสียงเราจึงจะต้องกระทำผ่านไดร์เวอร์ของการ์ดเสียงนั้น ๆ ซึ่งตัวโปรแกรม Windows เองก็ออกแบบมาให้ผู้พัฒนากระทำผ่านตัวไดร์เวอร์ของ Windows แต่ในความเป็นจริงแล้วเราสามารถที่จะจัดการระบบเสียงได้หลายวิธี ดังนี้

1 การเรียกใช้โดยผ่าน MCI (Multimedia Command String) ซึ่งวิธีนี้เป็นวิธีที่ง่ายที่สุด แต่ว่าวิธีนี้จะไม่สามารถเข้าจัดการระบบเสียงของระบบได้ลึกและมีประสิทธิภาพเท่าที่ควร

2.การเรียก Function ระดับต่ำจากตัวระบบของ Windows เอง ซึ่งวิธีการนี้จะสามารถเข้าถึงฟังก์ชันพื้นฐานในระดับลึกที่จะเข้าไปจัดการได้อย่างสูงสุดและเต็มประสิทธิภาพตามที่ไดร์เวอร์ที่ติดตั้งอยู่จะสามารถทำให้ได้

3.2.3 การพัฒนาในส่วนของการจัดการเรื่อง Compression เสียงบน Windows

ในส่วนนี้เราได้ใช้ประโยชน์ที่ได้จากไดร์เวอร์ของการ์ดเสียง และการเรียกใช้ API ซึ่งเป็น Function ระดับต่ำจากตัวระบบของ Windows โดยการเรียกใช้ตัวจัดการเกี่ยวกับการ Compression / Decompresion ที่ชื่อว่า Audio Compression Manager (ACM) ซึ่ง ACM นี้จะกระทำงานโดยสัมพันธ์กับไดร์เวอร์ของการ์ดเสียงที่ติดตั้งอยู่ในระบบตัว Audio Compression Manager จะทำการ Compression / Decompresion จากรูปแบบหนึ่งเป็นเป็นอีกรูปแบบหนึ่ง เช่นจาก GSM610 ไปเป็น PCM หรือกลับกับ

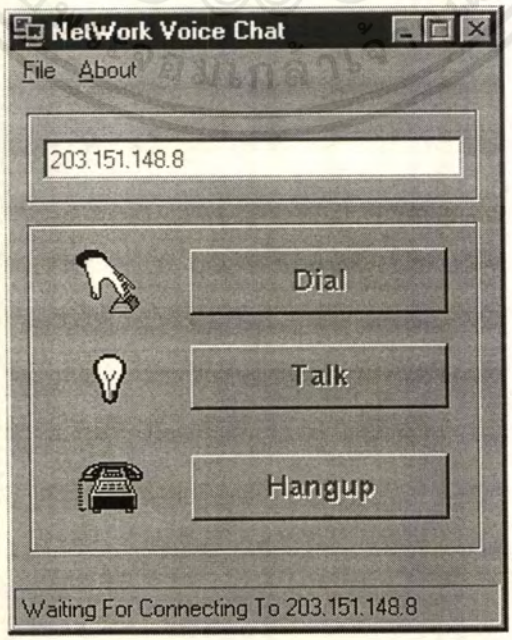
3.3 อุปกรณ์และระบบที่ต้องการ (Requirement)

1. เครื่องคอมพิวเตอร์สูงกว่า 486DX2-66
   - หน่วยความจำอย่างต่ำ 8 MB
2. การ์ดเสียง ที่ Compatible Sound Balster
3. การ์ดเน็ตเวิร์ก ที่สนับสนุน TCP/IP Protocol
4. ลำโพงและไมค์
5. Windows 95

3.4 โครงสร้างโดยรวมของโปรแกรม

โปรแกรม NetWork Voice Chat เป็นโปรแกรมประเภท Voice Chat ซึ่งก็หมายถึงใช้เสียงในการพูดคุย การทำงานของโปรแกรมจะมีลักษณะการทำงานเหมือนกับการใช้วิทยุมือถือ (VR) ดังกล่าวไปแล้ว นั่นหมายถึงว่าเป็นการสื่อสารแบบ Half-Duplex ตัวโปรแกรมจะใช้โปรโต-คอล TCP/IP ในการนำข้อมูลไปถึงยังผู้รับ ซึ่งการใช้งานโปรโตคอล TCP/IP นั้นกำหนดไว้ว่าเครื่องคอมพิวเตอร์ที่โปรโตคอลนี้จะมี IP Adress ที่แน่นอนเป็นของตนเองและไม่ซ้ำกับใครในเครือข่ายเน็ตเวิร์ก ซึ่งเปรียบเสมือนเบอร์โทรศัพท์นั่นเอง โปรแกรมจะทำการบันทึกเสียงแล้วทำการ Compression ตามที่กำหนดจากนั้นก็ส่งเข้าไปแพ็กข้อมูลให้อยู่ในรูปของ TCP/IP แล้วจึงทำการส่งออกไป และในทางกลับกันก็จะนำ Packages ของ TCP/IP ที่ส่งมาถึงตนทำการแกะข้อมูลที่เป็นเสียงออกมาแล้วทำการส่งออกไปเล่นเสียงให้ออกสู่ลำโพง

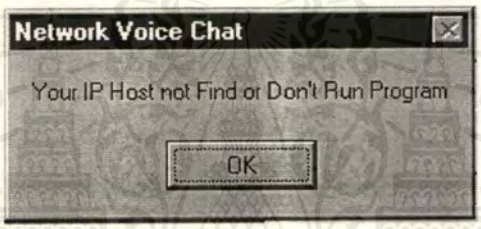
3.5 การออกแบบส่วนติดต่อกับผู้ใช้ (User-Interface) และการใช้งานโปรแกรม



หน้าตาการติดต่อกับผู้ใช้ของโปรแกรม NetWork Voice Chat จะมีหน้าตาดังรูปด้านบน โดยหลัก ๆ แล้วจะมีปุ่มอยู่ 3 ปุ่มคือ 1. Dial 2.Talk 3.Hangup และมีช่องสำหรับให้ผู้ใช้ใส่หมายเลข IP Address ของเครื่องคอมพิวเตอร์ปลายทาง นอกจากนี้ข้างล่างจะมี Status Bar สำหรับแสดงสถานะการทำงานปัจจุบันของโปรแกรม และในส่วนเมนูของโปรแกรมจะประกอบด้วย 2 เมนูหลักคือ File และ About

สำหรับรายละเอียดการทำงานของปุ่มต่าง ๆ มีดังนี้

1. ปุ่ม Dial จะเป็นการเริ่มทำ Connection ขึ้นระหว่างเครื่องคอมพิวเตอร์ขึ้นโดยจะนำเอาหมายเลข IP Address ที่ผู้ใช้ใส่ในช่องนำไปเริ่มทำ Connection โดยเมื่อทำสำเร็จปุ่ม Dial ก็จะถูก Disable การทำงานไปและปุ่ม Talk และ Hangup ก็จะ Active ขึ้นมาให้ผู้ใช้สามารถใช้งานได้ แต่ถ้าหมายเลข IP Address นั้นไม่มีหรือไม่ได้เปิดโปรแกรมรออยู่จะแสดงหน้าจอผิดพลาดในการติดต่อดังรูป



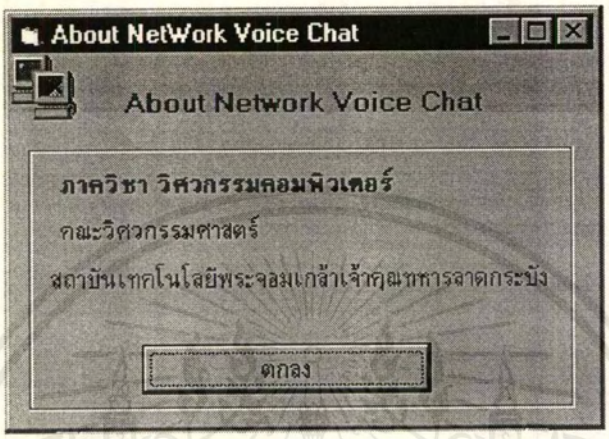
2. ปุ่ม Talk ปุ่มนี้จะทำงานได้เมื่อการทำ Connection สำเร็จ โดยการทำงานจะทำหน้านี้เมื่อผู้ใช้กดปุ่มนี้ค้างไว้ในเวลาที่ต้องการจะส่งเสียงไปยังอีกฝ่ายหนึ่ง ซึ่งมันจะเริ่มอัดเสียงแล้วส่งออกไป และขณะที่มันทำงานอัดและส่งเสียงไปมันจะเปลี่ยนชื่อปุ่มเป็น Talking เมื่อผู้ใช้ไม่ได้กดปุ่มค้างไว้มันจะอยู่ในสภาวะ Listen รอรับเสียงจากอีกฝ่ายหนึ่ง และเมื่อมันรับเสียงเข้ามาแล้วทำการเล่นกลับเสียงมันจะเปลี่ยนชื่อปุ่มเป็น Playing แทน
3. ปุ่ม Hangup จะทำหน้าที่ยกเลิกการทำ Connection หรือการวางหูนั่นเอง ซึ่งเป็นการสิ้นสุดการสนทนา
4. ช่องใส่หมายเลข IP Adress จะเป็นช่องสำหรับใส่หมายเลข IP Adress ของเครื่องคอมพิวเตอร์ที่ต้องการจะติดต่อด้วย โดยผู้ใช้จะต้องใส่เป็นเลขจำนวนเต็มตามรูปแบบของ IP Adress ดังนี้ XXX.XXX.XXX.XXX
5. Status Bar ซึ่งจะอยู่ด้านล่างของโปรแกรมซึ่งจะแสดงสถานะการทำงานของโปรแกรมดังนี้

-Ready to connection หมายถึงโปรแกรมพร้อมจะทำงานคือพร้อมรับสัญญาณการเรียกเข้าและพร้อมจะเรียกไปยังคอมพิวเตอร์เครื่องอื่น

- Waiting For Connection to XXX.XXX.XXX.XXX หมายถึงรอการตอบรับจากเครื่องคอมพิวเตอร์อีกฝ่ายที่เราต้องการติดต่อด้วย

- Connection to XXX.XXX.XXX.XXX หมายถึงกำลังติดต่อกับเครื่องคอมพิวเตอร์อีกฝ่ายที่เราต้องการติดต่อด้วยอยู่และพร้อมทำงานแล้ว

6. สำหรับเมนูของโปรแกรมซึ่งมี 2 เมนูหลักคือ File และ About โดยในเมนู File จะมีเมนูย่อย Exit สำหรับออกจากโปรแกรม NetWork Voice Chat ส่วนในเมนู About จะแสดงรายละเอียดของโปรแกรมเล็กน้อยดังรูป



บทที่ 4

การทดลองและการสรุปผล

4.1 การทดลองพัฒนาในส่วนของการรับและส่งข้อมูลผ่านระบบเน็ตเวิร์ก (NetWork Connection)

ในที่นี้เราได้สร้างโปรแกรมสำหรับการทดลองรับและส่งข้อมูลผ่านเครือข่าย โดยใช้โปรโตคอล TCP/IP ทั้งนี้เราได้เขียนขึ้นโดยใช้โปรแกรม Visual Basic 4.0 พร้อมได้เรียกใช้ Winsock Control

Winsock TCP Protocol

ขั้นตอนการทำงาน

บนเครื่อง Server Computer

1. กำหนดช่องที่จะใช้ในการสื่อสาร
2. รอรับการขอใช้บริการ
3. เมื่อได้รับคำขอใช้บริการจะตอบสนองต่อความต้องการนั้น ๆ
4. ทำการสื่อสารข้อมูล

บนเครื่อง Client Computer

1. กำหนดหมายเลขเครื่องปลายทางที่จะขอทำการติดต่อ
2. กำหนดช่องสื่อสารเครื่องปลายทางที่จะขอทำการติดต่อ
3. ทำการสื่อสารข้อมูล
4. แจ้งผู้ใช้งานว่าทำการติดต่อสำเร็จ
5. ทำการแลกเปลี่ยนข้อมูล

6. ขอปิดช่องทางการสื่อสาร

การเริ่มต้นใช้งาน Winsock TCP ทำโดยสร้าง Socket แรกเริ่มโดยกำหนดตำแหน่ง 0 ให้โดยใน Visual Basic ต้องมีการประกาศตัวแปลดังนี้

Public gSockInstance As Integer

Visual Basic จะมี Object ที่จะถูกใช้งานดังนี้

บนเครื่อง Server Computer

- Winsock TCP control named "SktTCPServer"
- Form named " frmServer"
- CommandButton control named " cmdSendData"
- TextBox control named "txtSend"

บนเครื่อง Client Computer

- WinSock TCP control named "skt TCP Client"
- CommandButton control named "cmdConnect"
- CommandButton control named "cmdCloseConnection"
- TextBox control named "txtRecieved"
- Label control named "lblStatus"

Server : กำหนดคุณสมบัติของPORT ต่างๆ

WinSock TCP ของ Server จะทำการกำหนดคุณสมบัติต่างๆให้ Port ก่อนเป็นอันดับแรกแม้ว่าเราจะสามารถใช้Portใดๆก็ได้ แต่หมายเลข Port บางหมายเลขถูกสงวนไว้สำหรับใช้งานเฉพาะเป็น HTML browser จะใช้ Port หมายเลข 80 Code ข้างใต้จะใช้หมายเลข 1007 สำหรับการติดต่อที่ไม่มี Protocalไหนใช้มัน

sktTCPsvr. Local Port = 1007

Server : ทำการรอการขอเรียกใช้บริการ

จะมี Port ที่ใช้ในการรอการเรียกใช้บริการว่า TCP จะต้องรอการร้องขอใช้บริการจากตัว client Computer ดังแแสดงที่ Code ข้างล่างนี้

```
Private Sub frmServer_ Load( )

        sktTCPsvr. Local Port = 1007 '  Set the Local port.

        sktTCPsvr. Listen '  Use the Listen method.

End Sub
```

Client :ต้องการติดต่อกับ Server

เริ่มการติดต่อต้องมีการทำการติดต่อก่อนเพื่อที่จะทำการติดต่อได้สำเร็จ Client ต้องใช้ Connect Method โดยจะต้องมีการใช้งาน RemoteHost และ RemotePort RemoteHost จะทำการระบุว่าใครต้องการทำการติดต่อด้วย โดยกำหนดเป็น IP address และ RemotePort ค่ากำหนดว่าจะทำการติดต่อกับ Remote ที่ Port ใด

```
Private Sub cmdConnect_ Click ( )

With  sktTCPClient

        RemoteHost  =  " 123.123.101.201"

        RemotePort  =        1007

        Connect

End With
```

```
End Sub
```

หรืออาจใช้รูปแบบนี้ก็ได้

```
Private Sub cmdConnect _ Click ( )

        sktClient. Connect "123.123.101.201" , 1007

End Sub
```

Server : การใช้ Accept Method เพื่อตอบรับการติดต่อ

เมื่อ Server ได้รับการขอการติดต่อจาก Client Computer จะใช้ Accept Method ในการตอบสนองเพื่อบอก Client ว่าการติดต่อสำเร็จ เมื่อ Server ได้รับ Request โดยมีการขอทำ Connection Server จะทำการสร้าง Socket ใหม่แล้วให้ Socketที่สร้างใหม่ทำการตอบสนองแทน

```
Private Sub sktServer _ ConnectionRequest_

( Index As Integer, ByVal requestID AS Long )

        ' Increment the global variable.

        gSockInstance = gSockInstance + 1

        Load sktServer (gSockInstance )

        sktServer (gSockInstance ). Accept request ID

End Sub
```

Client : ทำการแจ้ง User ว่า ทำการติดต่อสำเร็จแล้วดัง Code ข้างล่างจะเป็นการบอก User ว่าการติดต่อสำเร็จ

```
Private Sub sktClient Connect ( )

        If sktClient. State = sckConnected Then
```

```
            ' Presuming a Statusbar exists, with one panel.
            lblStatus. Caption = " Connection Successful !"
            End If
        End Sub
```

Server : ส่งข้อมูลให้Client โดยใช้ SendData Method

เมื่อทำการติดต่อสำเร็จเราสามารถส่งข้อมูลไปให้ Computer อีกฝั่งได้โดยใช้ SendData Method ตาม Code ข้างล่าง

```
        Private Sub cmdSendData _ Click( )
            sktClient . SendData " this is how we begin."
        End Sub
```

Client : ใช้วิธีการ Get Data เพื่อรับข้อมูลบนเครื่อง Client เมื่อมีการส่งข้อมูลมาถึง จะใช้ Get Data Method เพื่อทำการรับข้อมูล

```
        Private Sub sktServer DataArrival
        ( Index As Integer , byVal bytesTotal As Long )
            Dim vtData ' Declare a variant to hold the data.
            sktServer ( Index ). Get Data vtData , vbString
            txtReceived . Text = vt Data ' Display the data .
        End Sub
```

Client : ปิดการติดต่อโดยใช้ Close Method หลังจากทำการส่งข้อมูลเสร็จสิ้น Client จะทำการยกเลิกการติดต่อโดยใช้ Close Method

```
Private Sub cmdCloseConnection _ Click ( )

        sktTCPClient . Close

End Sub
```

และมีการปิดการบริการจากบน Server ด้วยโดยการยกเลิกการควบคุมและลดค่าตัวแปรรวมลง

```
Private Sub sktTCPServer _ Close ( Index As Integer )

        sktTCPServer ( Index ) . Close

        Unload sktTCPServer ( Index ) ' Unload the instance .

        gSockInstance = gSockInstance - 1 ' Decrement the ' variable.

End Sub
```

WinSock UPD Control

WinSock UPD จะมีการทำงานเหมือน WinSock TCP ถึงอย่างไรก็ตามมันเป็น Protocol ที่ไม่มีการตรวจสอบการติดต่อเหมือน WinSock TCP การทำการส่งข้อมูลจาก Computer เครื่องหนึ่งไปยัง Computer อีกเครื่องหนึ่ง

บนเครื่องผู้ส่ง

* กำหนด RemoteHost และ RemotePort

* ส่งข้อมูลโดยใช้ Send Data Method

บนเครื่องผู้รับ

* กำหนด Port ที่จะรับข้อมูล

* รับข้อมูลโดยใช้ Get Data Method

การติดต่อ :

Visual Basic จะใช้ object ดังต่อไปนี้

บนเครื่องผู้ส่ง

* Form named " frmSend "

* UPD control named " udpSender "

* CommandButton control named " cmdSendData "

* TextBox control named " txtSend "

บนเครื่องผู้รับ

* Form named " frmReceiver "

* UDP control named " udpReceiver "

* TextBox control named " txtReceived"

Sender : กำหนดหมายเลขของเครื่อง Computer ที่ต้องการสื่อสารด้วยรวมถึง Port ด้วย

```
private Sub frmSend _ Load ( )

        udpSender . RemoteHost  =  " 123.123.101.201 "

        udpSender . RemotePort   =  1007

End Sub
```

Receiver : เครื่อง Computer เป็นผู้รับทำการกำหนด Port ที่จะทำการตอบสนองถึงผู้ส่ง

```
Private Sub frmReceiver _ Load ( )

        udpReceiver . LocalPort = 1007
```

```
End Sub
```

Sender : ส่งข้อความให้ผู้รับโดยใช้ SendData Method

```
Private Sub cmdSendData _ Click ( )

    udpSender . SendData " Calling all cars ....."

End Sub
```

Receiver : ใช้ Get Data Method เพื่อทำการรับ Data ที่ผู้ส่งมาให้ โดยข้อความที่ได้รับจะถูกเขียนลงใน TextBox

```
Private Sub udpREceiver _ DataArrival_

( Byval bytesTotal As Long )

    Dim vtData ' Declare a variant to hold the data .

    udpreceiver . Get Data vtData , vbString

    txtReceived . Text = vtData ' Display the message .

End Sub
```

ผลการทดลอง

จากการทดลองเขียนโปรแกรมติดต่อผ่านเครือข่ายทั้งสองโปรแกรมโดยใช้โปรโตคอล TCP/IP เราสามารถที่จะทำการรับและส่งข้อมูลได้จริง

4.2 การทดลองพัฒนาในส่วนของการติดต่อกับการ์ดเสียง

```
' Project Name  : INFO_LOW.MAK
' Program Name  : Get Infomation by Low-Level Audio API Function.
' File name     : INFO_LOW.FRM
```

ประกาศตัวแปรและฟังก์ชันใน section ของ Decleation

```
Option Explicit
Dim NL As String * 2

Private Function sFormatDesc(wavFormat As Long) As String
    Dim p$
    If (WAVE_INVALIDFORMAT And wavFormat) Then
        p$ = "   Invalid Format"
    Else
        If (WAVE_FORMAT_1M08 And wavFormat) Then
            p$ = p$ & "   11.025 kHz, Mono, 8-bit" & NL
        End If
        If (WAVE_FORMAT_1S08 And wavFormat) Then
            p$ = p$ & "   11.025 kHz, Stereo, 8-bit" & NL
        End If
        If (WAVE_FORMAT_1M16 And wavFormat) Then
            p$ = p$ & "   11.025 kHz, Mono, 16-bit" & NL
        End If
        If (WAVE_FORMAT_1S16 And wavFormat) Then
            p$ = p$ & "   11.025 kHz, Stereo, 16-bit" & NL
        End If
        If (WAVE_FORMAT_2M08 And wavFormat) Then
            p$ = p$ & "   22.05 kHz, Mono, 8-bit" & NL
        End If
        If (WAVE_FORMAT_2S08 And wavFormat) Then
            p$ = p$ & "   22.05 kHz, Stereo, 8-bit" & NL
        End If
        If (WAVE_FORMAT_2M16 And wavFormat) Then
            p$ = p$ & "   22.05 kHz, Mono, 16-bit" & NL
        End If
        If (WAVE_FORMAT_2S16 And wavFormat) Then
            p$ = p$ & "   22.05 kHz, Stereo, 16-bit" & NL
        End If
        If (WAVE_FORMAT_4M08 And wavFormat) Then
```

```
        p$ = p$ & "   44.1 kHz, Mono, 8-bit" & NL
      End If
      If (WAVE_FORMAT_4S08 And wavFormat) Then
        p$ = p$ & "   44.1 kHz, Stereo, 8-bit" & NL
      End If
      If (WAVE_FORMAT_4M16 And wavFormat) Then
        p$ = p$ & "   44.1 kHz, Mono, 16-bit" & NL
      End If
      If (WAVE_FORMAT_4S16 And wavFormat) Then
        p$ = p$ & "   44.1 kHz, Stereo, 16-bit" & NL
      End If
    End If
    sFormatDesc = p$
End Function

Private Function sFuncSupport(wavFunc As Long) As String
    Dim p$

    If (WAVECAPS_PITCH And wavFunc) Then
      p$ = "   Supports pitch ontrol." & NL
    End If
    If (WAVECAPS_PLAYBACKRATE And wavFunc) Then
      p$ = p$ & "   Supports playback rate control." & NL
    End If
    If (WAVECAPS_VOLUME And wavFunc) Then
      p$ = p$ & "   Supports volume control." & NL
    End If
    If (WAVECAPS_LRVOLUME And wavFunc) Then
      p$ = p$ & "   Separate left-right volume control." & NL
    End If
    If (WAVECAPS_SYNC And wavFunc) Then
      p$ = p$ & "   Synchronous Driver." & NL
    End If
    sFuncSupport = p$
End Function
```

```
Private Function sPhraseVersion(wavVersionNum As Integer) As String
    Dim p$

    p$ = Format$(Hex$(wavVersionNum), "0000")
    sPhraseVersion = Val("&H" & Left$(p$, 2)) & "." & Val("&H" & Right$(p$, 2))
End Function

Private Function sZtrim(msg$) As String
    Dim i%, p$

    On Error Resume Next

    p$ = msg$
    i% = InStr(p$, Chr$(0))
    If i% Then p$ = Left$(p$, i% - 1)
    sZtrim = RTrim$(p$)
End Function
```

ประกาศตัวแปรและฟังชั่นใน FORM

```
Option Explicit

Private Sub Form_Activate()
    Dim WavIn As WAVEINCAPS
    Dim WavOut As WAVEOUTCAPS
    Dim p$, wInNumDevs%, wOutNumDevs%
    Dim wInDevsFlag%, wOutDevsFlag%

    NL = Chr$(13) & Chr$(10)
    wInNumDevs% = waveInGetNumDevs()
    wOutNumDevs% = waveOutGetNumDevs()
    p$ = "Waveform Audio Device Information :" & NL
    p$ = p$ & "Number of Input Device : " & wInNumDevs% & NL
    p$ = p$ & "Number of Output Device : " & wOutNumDevs% & NL & NL

    wInDevsFlag% = waveInGetDevCaps((wInNumDevs% - 1), WavIn, Len(WavIn))
    p$ = p$ & "Waveform Audio Input Device Information :" & NL
```

```
    p$ = p$ & "Manufacturer ID : " & WavIn.wMid & NL
    p$ = p$ & "Product ID : " & WavIn.wPid & NL
    p$ = p$ & "Driver Version : " & sPhraseVersion(WavIn.vDriverVersion) & NL
    p$ = p$ & "Product Name : " & sZtrim(WavIn.szPname) & NL
    p$ = p$ & "Formats Supported : " & NL & sFormatDesc(WavIn.dwFormats)
    p$ = p$ & "Number Of Channels : " & WavIn.wChannels & NL & NL

    wOutDevsFlag% = waveOutGetDevCaps((wOutNumDevs% - 1), WavOut, Len(WavOut))
    p$ = p$ & "Waveform Audio Output Device Information :" & NL
    p$ = p$ & "Manufacturer ID : " & WavOut.wMid & NL
    p$ = p$ & "Product ID : " & WavOut.wPid & NL
    p$ = p$ & "Driver Version : " & sPhraseVersion(WavOut.vDriverVersion) & NL
    p$ = p$ & "Product Name : " & sZtrim(WavOut.szPname) & NL
    p$ = p$ & "Formats Supported : " & NL & sFormatDesc(WavOut.dwFormats)
    p$ = p$ & "Number Of Channels : " & WavOut.wChannels & NL
    p$ = p$ & "Functionality Supported : " & NL & sFuncSupport(WavOut.dwSupport)
    Text1.Text = p$
End Sub


Private Sub Form_Load()
    ' Show form in the middle of screen.
    Move (Screen.Width - ScaleWidth) \ 2, (Screen.Height - ScaleHeight) \ 2
End Sub

Private Sub Form_Unload(Cancel As Integer)
    End
End Sub
```

ประกาศตัวแปรและฟังก์ชันใน Text

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
    KeyAscii = 0
End Sub
```

ประกาศตัวแปรและฟังก์ชันใน Module info_low.bas

```
Option Explicit
'
' Project Name  : INFO_LOW.MAK
' Program Name  : Get Infomation by Low-Level Audio API Function.
' File name     : INFO_LOW.FRM
'         }
'

Global Const MAXPNAMELEN = 32                    ' max product name length (including NULL)

' Defines for dwFormat field of WAVEINCAPS and WAVEOUTCAPS.
Global Const WAVE_INVALIDFORMAT = &H0            ' invalid format
Global Const WAVE_FORMAT_1M08 = &H1              ' 11.025 kHz, Mono,   8-bit
Global Const WAVE_FORMAT_1S08 = &H2              ' 11.025 kHz, Stereo, 8-bit
Global Const WAVE_FORMAT_1M16 = &H4              ' 11.025 kHz, Mono,   16-bit
Global Const WAVE_FORMAT_1S16 = &H8              ' 11.025 kHz, Stereo, 16-bit
Global Const WAVE_FORMAT_2M08 = &H10             ' 22.05  kHz, Mono,   8-bit
Global Const WAVE_FORMAT_2S08 = &H20             ' 22.05  kHz, Stereo, 8-bit
Global Const WAVE_FORMAT_2M16 = &H40             ' 22.05  kHz, Mono,   16-bit
Global Const WAVE_FORMAT_2S16 = &H80             ' 22.05  kHz, Stereo, 16-bit
Global Const WAVE_FORMAT_4M08 = &H100            ' 44.1   kHz, Mono,   8-bit
Global Const WAVE_FORMAT_4S08 = &H200            ' 44.1   kHz, Stereo, 8-bit
Global Const WAVE_FORMAT_4M16 = &H400            ' 44.1   kHz, Mono,   16-bit
Global Const WAVE_FORMAT_4S16 = &H800            ' 44.1   kHz, Stereo, 16-bit

' Flags for dwSupport field of WAVEOUTCAPS.
Global Const WAVECAPS_PITCH = &H1                ' supports pitch control
Global Const WAVECAPS_PLAYBACKRATE = &H2         ' supports playback rate control
Global Const WAVECAPS_VOLUME = &H4               ' supports volume control
Global Const WAVECAPS_LRVOLUME = &H8             ' separate left-right volume control
Global Const WAVECAPS_SYNC = &H10

' Waveform input device capabilities structure.
Type WAVEINCAPS
   wMid          As Integer                      ' manufacturer ID
```

```
    wPid            As Integer                      ' product ID
    vDriverVersion  As Integer                      ' version of the driver
    szPname         As String * MAXPNAMELEN         ' product name (NULL terminated string)
    dwFormats       As Long                         ' formats supported
    wChannels       As Integer                      ' number of channels supported
End Type

' Waveform output device capabilities structure.
Type WAVEOUTCAPS
    wMid            As Integer                      ' manufacturer ID
    wPid            As Integer                      ' product ID
    vDriverVersion  As Integer                      ' version of the driver
    szPname         As String * MAXPNAMELEN         ' product name (NULL terminated string)
    dwFormats       As Long                         ' formats supported
    wChannels       As Integer                      ' number of sources supported
    dwSupport       As Long                         ' functionality supported by driver
End Type

Declare Function waveInGetDevCaps Lib "MMSYSTEM" (ByVal udeviceid As Integer, lpCaps As
                                        WAVEINCAPS, ByVal uSize As Integer) As Integer
Declare Function waveInGetNumDevs Lib "MMSYSTEM" () As Integer
Declare Function waveOutGetDevCaps Lib "MMSYSTEM" (ByVal udeviceid As Integer, lpCaps As
                                        WAVEOUTCAPS, ByVal uSize As Integer) As Integer
Declare Function waveOutGetNumDevs Lib "MMSYSTEM" () As Integer
```

ผลการทดลอง

จากการโปรแกรมด้านบนเป็นโปรแกรมที่จะศึกษาถึงวิธีเข้าควบคุมการทำงานของการ์ดเสียงโดยการเรียกใช้ API ที่เกี่ยวกับ Mulimedia ของ Windows ซึ่งจะเป็นแนวทางในการเขียนโปรแกรมสำหรับใช้ในโปรเจคต่อไป

บทที่ 5
สรุปและวิจารณ์

จากการพัฒนาโปรแกรมสื่อสารด้วยเสียงพูดบนระบบเครือข่ายนี้ ทำให้ได้ข้อสรุปที่ว่าเราสามารถที่จะทำการรับและส่งข้อมูลเสียงผ่านระบบเครือข่ายได้จริง ซึ่งประสิทธิภาพของการรับและส่งข้อมูลเสียงที่ได้จากการทดลองใช้งานในระบบเครือข่ายต่าง ๆ พบว่าถ้าเป็นการรับและส่งในเครือข่ายเดียวกันประสิทธิภาพของสัญญาณเสียงที่ได้นับว่าพอจะรับฟังได้เป็นที่น่าพอใจมากทีเดียว ถึงแม้จะมีการหน่วงของสัญญาณเสียงบ้าง แต่ถ้าเป็นการรับและส่งคนละเครือข่ายคุณภาพของสัญญาณเสียงที่ได้ค่อนข้างจะแย่ ซึ่งถ้าพิจารณาแล้วก็ต้องยอมรับกันว่าคงจะต้องเป็นเช่นนั้น เพราะเหตุผลต่าง เช่น

1. Traffic ของข้อมูลที่อยู่นอกเครือข่ายมีค่อนข้างสูงทำให้เกิดการหน่วงของสัญญาณที่สูงตามสภาวะของ Traffic ขณะนั้นด้วย
2. พื้นฐานการทำงานของโปรโตคอล TCP/IP ที่ต้องรับ Package ของข้อมูลทุก ๆ Package
3. ช่วงเวลาในการบันทึกและบีบข้อมูลก่อนส่งออกไปของตัวโปรแกรมเอง
4. วิธีการบีบย่อและขยายข้อมูล

โดยปัญหาต่าง ๆ ข้างต้นเราสามารถที่จะแก้ไขได้ด้วยตนเองได้ในบางจุดเช่นการพัฒนาวิธีการบีบย่อและขยายข้อมูลให้สามารถที่จะบีบย่อได้มากขึ้นในเวลาอันรวดเร็วโดยไม่สูญเสียข้อมูลหรือสูญเสียในระดับที่ยอมรับได้ หรือหันไปใช้โปรโตคอล TCP/IP แบบ UDP ซึ่งจะช่วยลดการหน่วงของสัญญาณลงได้ แต่ก็จะมีการขาดหายของข้อมูลในเครือข่ายที่มี Traffic ที่สูง หรือการพัฒนาไปใช้โปรโตคอลใหม่ ๆ ที่คิดค้นขึ้นมาเองสำหรับใช้รับและส่งข้อมูลที่เป็นเสียงโดยเฉพาะ

ก้าวต่อไป

โปรแกรมสื่อสารด้วยเสียงพูดบนระบบเครื อข่ายยังมีจุดที่จะสามารถพัฒนาต่อไปให้มีประ-สิทธิภาพและคุณสมบัติใหม่ ๆ เพิ่มเติมได้ เช่น

1. หันไปใช้โปรโตคอล UDP แทน TCP เพื่อช่วยลดการหน่วงของเสียงดังกล่าวข้างต้น
2. พัฒนาวิธีการบีบย่อและขยายข้อมูลให้สามารถที่จะบีบย่อได้มากขึ้นในเวลาอันรวดเร็ว ซึ่งเป็นวิธีการบีบย่อและขยายสำหรับสัญญาณเสียงเท่านั้น
3. เพิ่ม Feature ให้สามารถเลือกผู้ที่จะคุยได้
4. เลือกคุณภาพของการรับและส่งได้ตามสภาวะของ Traffic ในขณะนั้น

5. เพิ่มความสามารถให้สามารถฝากข้อความที่เป็นเสียงได้ในขณะไม่เปิดโปรแกรม

6.อาจจะให้สามารถคุยกับโปรแกรมของผู้ผลิตรายอื่น ๆ ได้ เช่น InterNet Phone ของ Vocal Tech เป็นต้น

สุดท้ายนี้หวังว่าโปรแกรมสื่อสารด้วยเสียงพูดบนระบบเครือข่ายจะเป็นแนวทางในการที่จะช่วยผู้พัฒนาคนอื่น ๆ สร้างแอพพลิเคชันสำหรับใช้งานบนระบบเครือข่ายได้ ถึงแม้จะไม่เป็นแอพพลิเคชันที่เกี่ยวกับการรับและส่งข้อมูลเสียงก็ตาม

ภาคผนวก

กิตติกรรมประกาศ

โปรเจคชิ้นนี้สำเร็จลงได้ก็เพราะความช่วยเหลือของหลาย ๆ ฝ่ายที่ได้ให้แนวคิดแนวทางในการออกแบบตลอดจนแนวทางในการเขียนโปรแกรมและเทคนิคต่าง ๆ ซึ่งก็ต้องขอกราบขอบพระคุณบุคคลดังต่อไป

อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษาที่คอยช่วยแนะนำในแนวคิดกว้าง ๆ

ระบบเครือข่ายอินเตอร์เน็ต ที่คอยเป็นแหล่งในการหาข้อมูลและการแก้ปัญหาต่าง ๆ

www.microsoft.com Web Site ที่เป็นแหล่งศึกษาและศึกษาถึงแนวทางในการเขียนโปรแกรม

ผู้แต่งตำรา Microsoft Visual BASIC ต่าง ๆ ที่ใช้เป็นแหล่งศึกษาถึงแนวทางในการใช้งานโปรแกรม Visual Basic

และท้ายสุด ขอขอบคุณเพื่อน ๆ ทุกคนที่คอยถามถึงสารทุกข์สุขดิบและความเป็นไปของโปรเจค และอีกหลาย ๆ คนที่ไม่ได้กล่าวถึงในที่นี้

หนังสืออ้างอิง

Visual C++ 4 Unleashed แต่งโดย Viktor Toth
สำนักพิมพ์ SAMS Publishing

Windows 95 Multimedia & ODBC API Bible Book3 แต่งโดย Richard J.Simon ,Tony David ,John Eaton ,Murray Goertz
สำนักพิมพ์ WAITE Group Press

Visual Basic 4 HOW-TO แต่งโดย THOMAS
สำนักพิมพ์ WAITE Group Press

Building OLE Applications With Visual Basic 4 แต่งโดย Forrest Houlette,Steven Ellid White
สำนักพิมพ์ QUE

Network Programming With WINDOWS SOCKETS แต่งโดย PAT BONNER
สำนักพิมพ์ PTR PH

Building Internet Application with Visual Basic แต่งโดย Michael Marchuk
สำนักพิมพ์ QUE

WEB Site หนังสืออ้างอิง

http://www.microsoft.com	เอื้อเฟื้อข้อมูลเกียวกับ ACM,Sound Card Function
http://www.bdti.com/faq/23.htm	About ADPCM
http://lands.let.kun.nl/links.en.html	Dept. of LANGUAGE AND SPEECH Links
http://www.ee.cityu.edu.hk/~cfchan/demo.html	Dr. C.F. Chan, Voice Coder Demonstrations
http://xfactor.wpi.edu/Works/MQP/ephone/main.html	Ethernet Phone Tittle Page
http://itre.ncsu.edu/gsm/	GSM Streaming Audio for the Web
http://ima.org/cparch/audio/index.html	IMA ADPCM
http://www.wildstar.net/irc/clients/windows/winsock/ftp/	Index of -irc-clients-windows-winsock-ftp
http://www.cam.org/~noelbou/gsm_wine.html	Internet Audio Publisher and DirectAudio Player
http://rpcp.mit.edu/~itel/	Internet Telephony Interoperability Consortium
http://ds.internic.net/ds/dspg0intdoc.html	InterNIC Internet Documentation (RFC's, FYI's.
http://www.sockets.com/ms_icmp.htm	Microsoft's ICMP API
http://christie.prognet.com/	Progressive Networks, The Home of RealAudio
http://www.microsoft.com/MEDIADEV/AUDIO/MSPEECH1.HTM	Speech API Microsoft
http://www.wpi.edu/~murti/mqp/chapter2.html	Speech Compression Under Ethernet Phone
http://rpcp.mit.edu/~itel/dsp.html	Speech Compression-DSP
http://eeandhui.cityu.edu.hk/spch_sw.htm	Speech Processing Source Code
http://tips.iworld.com/	The Internet Product Site Home Page.
http://www.apexsc.com/vb/library.html	The Library at Carl & Gary's
http://www.kingsoft.com/qaid/vbwn0011.htm	vbwn0011.htm at www
http://www.northcoast.com/savetz/voice-faq.html	voice-faq.html at www
http://whatis.com/nfindex.htm	whatis.com Home Page
http://www.illuminet.net/~jsmith/home.html	Wild Web of Visual Basic!
http://www.learningtree.com/us/cbt/cbt301m1.htm	Windows Programming With Visual C$$ and MFC
http://www.mcp.com/que/	QUE Publishing Home Page