



การสื่อสารมัลติมีเดียผ่านเครือข่ายอินเทอร์เน็ต
MULTIMEDIA COMMUNICATION ON INTERNET

โดย
นางสาว นิติตา สืบประสงค์
นางสาว วิไลรัตน์ พิธรากร

วัน เดือน ปี..... 18.ค.ค. 2541
เลขทะเบียน..... 039085.....
เลขเรียกหนังสือ..... T.40326 พ. ๒๙๖ ก.

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2540

การสื่อสารมัลติมีเดียผ่านเครือข่ายอินเทอร์เน็ต
MULTIMEDIA COMMUNICATION ON INTERNET

โดย

นางสาว นิลิตา สืบประสงค์ 37014207

นางสาว วิไลรัตน์ พิธรากร 37014410

อาจารย์ที่ปรึกษา

รศ.ดร. สุวิพล ลิทธิชีวภาค

ผศ. เกรียงไกร วงศ์โรจนภรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2540

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การสื่อสารมัลติมีเดียผ่านเครือข่ายอินเทอร์เน็ต

Multimedia Communication On Internet

ผู้จัดทำ

1. นางสาว นิตดา สืบประสงค์ 37014207

2. นางสาว วิไลรัตน์ ทิรรากร 37014410

สุวิมล

อาจารย์ที่ปรึกษา

(รศ.ดร. สุวิมล สิริชีวะภาค)

ผศ.เกรียงไกร วงศ์โรจนภรณ์

อาจารย์ที่ปรึกษา

(ผศ. เกรียงไกร วงศ์โรจนภรณ์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การสื่อสารมัลติมีเดียผ่านเครือข่ายอินเทอร์เน็ต
Multimedia Communication on Internet

โดย นางสาว นิสิตา สืบประสงค์ 37014207

นางสาว วิไลรัตน์ พิธรากร 37014410

อาจารย์ที่ปรึกษา รศ.ดร. สุวิพล ลิทธิชีวะภาค

ผศ. เกรียงไกร วงศ์โรจนภรณ์

บทคัดย่อ

โครงการนี้เป็นการเขียนโปรแกรมประยุกต์เพื่อใช้ติดต่อสื่อสารมัลติมีเดีย โดยทำหน้าที่รับ-ส่งข้อมูลภาพและเสียง รวมทั้งข้อมูลตัวอักษร ข้อมูลออดิโอวิดีโอ และข้อมูลอื่นๆผ่านทางเครือข่ายอินเทอร์เน็ต ซึ่งจะแบ่งการทำงานของโปรแกรมออกเป็น 3 ส่วน คือ โปรแกรมด้านส่งไฟล์ โปรแกรมด้านรับไฟล์ และ โปรแกรมพิมพ์ข้อความโต้ตอบ โดยโปรแกรมต่างๆเหล่านี้ได้ใช้โปรแกรมเดลฟี 2.0 (Delphi 2.0) เป็นเครื่องมือสำหรับพัฒนาโปรแกรม

ABSTRACT

This project presents the program for multimedia communication between two computers on Internet. They are used for sending and receiving text file, audio video file, picture file, sound file and the others. There are three parts sending program, receiving program and exchange string. This project is using Delphi2.0 for program development.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญเรื่อง

เรื่อง	หน้า
บทที่ 1 บทนำ	1
บทที่ 2 ทฤษฎีหรือหลักการ	2
2.1 ระบบเครือข่าย	2
2.1.1 ความหมายของระบบเครือข่ายคอมพิวเตอร์และอินเทอร์เน็ตเวิร์คกิ้ง	2
2.1.2 สถาปัตยกรรมของอินเทอร์เน็ต	2
2.1.3 ข้อกำหนดรูปแบบของเกตเวย์	3
2.2 โมเดลอ้างอิง OSI	3
2.3 TCP/IP	6
2.3.1 TCP/IP คืออะไร	6
2.3.2 ส่วนประกอบของ TCP/IP	6
2.3.2.1 กลุ่มข้อกำหนดรูปแบบการขนส่ง	6
2.3.2.2 กลุ่มข้อกำหนดเกี่ยวกับรูปแบบเส้นทาง	7
2.3.2.3 กลุ่มข้อกำหนดรูปแบบเกี่ยวกับที่อยู่เครือข่าย	7
2.3.2.4 กลุ่มข้อกำหนดรูปแบบเกี่ยวกับเส้นทางการสื่อสารระหว่างเครือข่าย	7
2.3.2.5 กลุ่มข้อกำหนดรูปแบบเกี่ยวกับการบริการผู้ใช้	8
2.3.3 โครงสร้างของชุดโปรโตคอล TCP/IP	8
2.3.4 ข้อแตกต่างระหว่างชุดโปรโตคอล TCP/IP	9
2.3.5 ลักษณะของการติดต่อ	10
2.3.6 ความสัมพันธ์ของ TCP และ UDP กับอินเทอร์เน็ต	10
2.3.7 พอร์ตและซ็อกเก็ต	11
2.3.8 ข้อกำหนดรูปแบบ IP	12
2.3.9 ข้อกำหนดรูปแบบ ICMP	12
2.4 ไอพีแอดเดรส (IP Address)	12
2.5 ซ็อกเก็ต(Socket)	14
2.5.1 เอพีไอ(API) ข้อกำหนดในการติดต่อสื่อสาร	14
2.5.2 ชนิดของซ็อกเก็ต	15
2.5.3 ซ็อกเก็ตแอดเดรส	16
2.5.4 Socket System Call	17
2.5.5 Application Protocol	19
2.5.6 กระบวนการ System Call พื้นฐาน	23
2.6 ระบบมัลติมีเดีย	27

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เรื่อง	หน้า
2.6.1 สื่อที่เป็นองค์ประกอบสำคัญของมัลติมีเดีย	27
2.6.1.1 ภาพ	27
2.6.1.2 เสียง	28
2.6.1.3 การโต้ตอบซึ่งกันและกัน	28
2.6.2 ส่วนประกอบของระบบมัลติมีเดีย	28
2.6.2.1 ฮาร์ดแวร์มัลติมีเดีย	28
2.6.2.2 ซอฟต์แวร์มัลติมีเดีย	29
2.6.3 เทคโนโลยีที่เกี่ยวข้องกับมัลติมีเดีย	30
2.6.4 Microsoft Audio Video Interleave (AVI)	30
2.6.5 Picture File	31
2.6.6 โครงสร้างของ RIFF ไฟล์	35
บทที่ 3 การคำนวณและการสร้าง	40
3.1 โปรแกรมพิมพ์ข้อความโต้ตอบ	42
3.2 โปรแกรมด้านส่งไฟล์	43
3.3 โปรแกรมด้านรับไฟล์	45
บทที่ 4 การทดลองและผลการทดลอง	49
4.1 การทดลองรับ-ส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์บนเครือข่ายอินเทอร์เน็ต	49
4.2 การทดลองรับส่งข้อความโต้ตอบระหว่างเครื่องคอมพิวเตอร์สองเครื่อง	54
บทที่ 5 บทวิจารณ์และบทสรุป	56
ภาคผนวก	
หนังสืออ้างอิง	

สารบัญรูปภาพ

รูป	หน้า
บทที่ 2 รูป 2.1 แสดงเครือข่ายอินเทอร์เน็ต	2
รูป 2.2 แสดงคลาสของ IP แอดเดรส	13
รูป 2.3 แสดงการเปรียบเทียบระหว่างซ็อกเก็ตและ SUN RPC application programming	15
รูป 2.4 แสดงลักษณะทั้งสามแบบของซ็อกเก็ต	16
รูป 2.5 แสดงโครงสร้างของข้อมูลที่ใช้ในการกำหนดซ็อกเก็ตแอดเดรส	17
รูป 2.6 วิธีการแบบ connectionless application protocol โดยใช้ UDP	20
รูป 2.7 วิธีการแบบ connection-oriented application protocol โดยใช้ TCP	22
รูป 2.8 แสดงโครงสร้างของภาพถ่ายในรูปแบบบิตแมพ	32
รูป 2.9 แสดงข้อมูลภาพบิตแมพ 256 สี	34
รูป 2.10 แสดงข้อมูลภาพบิตแมพ 16.7 ล้านสี	34
รูป 2.11 โครงสร้างของ RIFF ไฟล์	35
รูป 2.12 ตัวอย่างของ PCMWAVEFORMAT	38
บทที่ 3 รูป 3.1 บล็อกไดอะแกรมแสดงโครงสร้างของการส่งและการรับข้อมูล ระหว่างด้านส่งและด้านรับของโครงการ	41
รูป 3.2 แสดงการส่งสัญญาณโต้ตอบระหว่างโปรแกรมด้านส่งและด้านรับ	40
รูป 3.3 แผนผังแสดงการทำงานของโปรแกรม Fstring.pas	42
รูป 3.4 แผนผังแสดงการทำงานของโปรแกรม Sender.pas	44
รูป 3.5 แผนผังแสดงการทำงานของโปรแกรม Recv.pas	46-48
บทที่ 4 รูป 4.1 แสดงฟอร์มของโปรเจก Pmedia.dpr	49
รูป 4.2 แสดงฟอร์มของโปรเจก Psender.dpr ขณะทำการติดต่อกับปลายทางเพื่อส่งข้อมูล	50
รูป 4.3 แสดงฟอร์มของโปรเจก Precv.dpr เมื่อมีการติดต่อและพร้อมรับข้อมูล	50
รูป 4.4 แสดงฟอร์มของโปรเจก Psender.dpr เมื่อทำการส่งข้อมูลภาพและเสียง	51
รูป 4.5 แสดงฟอร์มของโปรเจก Precv.dpr เมื่อทำการรับข้อมูลภาพและเสียง	52
รูป 4.6 แสดงกราฟเสียงของไฟล์เสียงที่ส่งจากทางด้านส่ง	53
รูป 4.7 แสดงกราฟเสียงของไฟล์เสียงที่รับได้ทางด้านรับ	53
รูป 4.8 แสดงฟอร์มของโปรเจก Psender.dpr เมื่อทำการส่งไฟล์ข้อมูลแบบ Text	54
รูป 4.9 แสดงฟอร์มของโปรเจก Precv.dpr เมื่อทำการรับไฟล์ข้อมูลแบบ Text	54
รูป 4.10 แสดงผลของฟอร์ม Pfstring.dpr หลังจากส่งข้อมูลแล้ว	55
รูป 4.11 แสดงผลของฟอร์ม Pfstring.dpr หลังจากรับข้อมูลแล้ว	56
รูป 4.12 แสดงผลของฟอร์ม Pfstring.dpr ขณะทำการรับส่งข้อมูล	56

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตาราง	หน้า
บทที่ 2 ตาราง 2.1 OSI โมเดล	4
ตาราง 2.2 โครงสร้างของชุดโปรโตคอล TCP/IP เปรียบเทียบกับ OSI Model	8
ตาราง 2.3 หมายเลขพอร์ตที่ให้บริการต่างๆ	11
ตาราง 2.4 ค่าคงที่ของฟังก์ชันมัลติมีเดีย	36
ตาราง 2.5 โครงสร้างของไฟล์ .wav	37
ตาราง 2.6 กำหนดค่าคงที่	39



บทที่ 1

บทนำ

ปัจจุบันโลกของการติดต่อสื่อสารทางด้านข้อมูลมีความสำคัญอย่างมากทั้งในการแข่งขันทางธุรกิจ การศึกษา และรวมไปถึงการดำเนินชีวิตประจำวัน อินเทอร์เน็ต (Internet) ก็เป็นเทคโนโลยีที่ถูกนำมาใช้กันอย่างแพร่หลายเพื่อรองรับการเจริญเติบโต ทางด้านการสื่อสารข้อมูล ทั้งนี้เนื่องจากระบบอินเทอร์เน็ตเป็นระบบเครือข่ายคอมพิวเตอร์ที่เชื่อมโยงไปทั่วโลกและเป็นเครือข่ายที่มีผู้ใช้งานมากที่สุดในโลกอีกด้วย และโดยเฉพาะอย่างยิ่งในประเทศไทยการใช้งานระบบอินเทอร์เน็ตเริ่มขยายตัวอย่างรวดเร็วมีจำนวนผู้ใช้เพิ่มขึ้นเรื่อยๆ ทั้งภาครัฐและ ภาคเอกชน ทั้งนี้เนื่องจากบุคคลและหน่วยงานทั้งหลายสามารถใช้ประโยชน์จากระบบเครือข่ายนี้ได้อย่างมหาศาล

เราจึงสามารถกล่าวได้ว่า อินเทอร์เน็ตเป็นเทคโนโลยีทางด้านเครือข่ายที่ช่วยอำนวยความสะดวกและสื่อสารได้อย่างรวดเร็ว ทำให้ข้อมูลข่าวสารที่เกิดขึ้นอยู่ทั่วโลกสามารถรับรู้ได้ทั่วถึงกันโดยไม่มีพรมแดนเสมือนย่อโลกให้เล็กลง นับว่าเป็นประโยชน์อย่างมากสำหรับยุคข้อมูลข่าวสารในปัจจุบัน จะเห็นได้ว่าในปัจจุบันนี้การสื่อสารทางอินเทอร์เน็ตได้นำเอาเทคโนโลยีทางมัลติมีเดียมาใช้อำนวยความสะดวกเพื่อสื่อให้เกิดความเข้าใจและง่ายในการรับรู้อย่างยิ่ง คำว่า “มัลติมีเดีย” หรือที่นิยามไว้ในชื่อภาษาไทยว่า ระบบหลายสื่อ โดยคำว่า Media ตรงกับความหมายในภาษาไทยว่า สื่อ ซึ่งก็คือตัวกลางที่ช่วยทำให้เกิดความรู้ความเข้าใจ มักได้แก่ รูปภาพ เสียง ตัวอักษร นอกจากนั้นยังรวมถึงการนำข่าวสารจากที่หนึ่งไปยังอีกที่หนึ่ง โดยสื่อทำหน้าที่เป็นตัวกลางคอยให้บริการ ซึ่งอาจจะส่งไปในลักษณะของตัวอักษร รูปภาพ เสียง โดยใช้กรรมวิธีต่างๆกันไป ในปัจจุบันสื่อได้ถูกนำไปใช้กันอย่างกว้างขวางและมีขอบเขตขยายออกไปเรื่อยๆยังมีการคิดเทคโนโลยีใหม่ๆมากขึ้นเพียงใด รูปแบบของสื่อก็ยิ่งหลากหลายขึ้น อันจะทำให้เกิดความสะดวกและรวดเร็วมากขึ้นกว่าเดิม

คณะผู้จัดทำได้ตระหนักถึงความสำคัญของการสื่อสารและเทคโนโลยีของอินเทอร์เน็ต จึงได้จัดทำโครงการนี้ขึ้นมา ซึ่งโครงการนี้เป็นโปรแกรมที่จัดทำขึ้นเพื่อให้ผู้ใช้เครือข่ายอินเทอร์เน็ตสามารถรับและส่งข้อมูลข่าวสารมัลติมีเดียผ่านเครือข่ายได้ กล่าวคือ สามารถส่งข้อมูลภาพ, เสียง และข้อมูลตัวอักษร รวมทั้งการสนทนาข้อความในรูปตัวอักษรได้ตอบกัน ระหว่างเครื่องคอมพิวเตอร์ได้ โดยใช้โปรแกรมแคลไฟล์ 2.0 (Delphi 2.0) เป็นเครื่องพัฒนาโปรแกรม

บทที่ 2

ทฤษฎีหรือหลักการ

2.1 ระบบเครือข่าย

การพัฒนาโปรแกรมประยุกต์บนระบบเครือข่ายจำเป็นต้องมีความรู้พื้นฐานทางด้านระบบเครือข่ายพอสมควร ในหัวข้อนี้จะอธิบายหลักการของระบบเครือข่ายที่จำเป็นสำหรับการทำความเข้าใจในการพัฒนาโปรแกรมประยุกต์บนระบบเครือข่ายดังจะกล่าวต่อไป

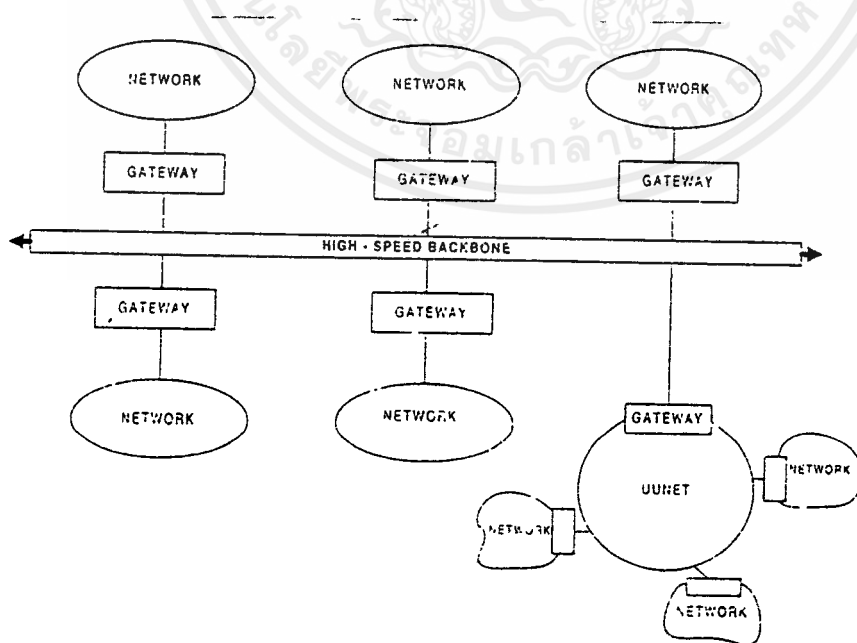
2.1.1 ความหมายของระบบเครือข่ายคอมพิวเตอร์ และอินเทอร์เน็ตเวิร์คกิ้ง(Internet working)

ระบบเครือข่ายคอมพิวเตอร์ (Computer Network) คือ ระบบการเชื่อมต่อระหว่างระบบปลายทาง (End-System) ซึ่งระบบปลายทางเป็นระบบอิสระจากกัน (Autonomous) ระบบปลายทางสามารถเป็นได้ตั้งแต่ไมโครคอมพิวเตอร์ (Microcomputer) ไปจนกระทั่งซูเปอร์คอมพิวเตอร์ (Supercomputer) ขนาดใหญ่ เพื่อจุดมุ่งหมายในการแลกเปลี่ยนข้อมูล และการแบ่งข้อมูลในการแบ่งปันทรัพยากรของระบบ เช่น ไฟล์ข้อมูล, เครื่องพิมพ์(Printer), โมเด็ม (Modem) ตลอดจนการให้บริการฐานข้อมูลร่วม (Sharing database)

อินเทอร์เน็ตเวิร์คกิ้ง หรือ อินเทอร์เน็ต (Internet) คือ การเชื่อมต่อระบบเครือข่าย 2 เครือข่ายขึ้นไป ดังนั้นคอมพิวเตอร์บนระบบเครือข่ายหนึ่งก็สามารถติดต่อกับคอมพิวเตอร์บนระบบเครือข่ายอื่นๆได้

2.1.2 สถาปัตยกรรมของอินเทอร์เน็ต

ถ้ากล่าวถึงอินเทอร์เน็ตว่าเป็นเครือข่ายที่ใหญ่ที่สุดในโลก เครือข่ายที่มีการต่อเสมือนกับใยแมงมุมครอบคลุมโลก อินเทอร์เน็ตเชื่อมโยงโลกเข้าด้วยกันอย่างไร้มิติ (Cyberspace) จากคำกล่าวเหล่านี้ทำให้เราควรรู้ว่าอินเทอร์เน็ตจัดการส่วนประกอบต่างๆอย่างไร จึงทำให้แต่ละส่วนสามารถรับ-ส่งข้อมูลและทำงานสัมพันธ์กันได้อย่างดี เราจึงจำเป็นต้องศึกษาสถาปัตยกรรมของอินเทอร์เน็ตดังต่อไปนี้



รูป 2.1 แสดงเครือข่ายอินเทอร์เน็ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อินเทอร์เน็ตประกอบด้วยสายสื่อสารความเร็วสูงเป็นแบคโบน(Backbone) เป็นสายโทรศัพท์ตามมาตรฐาน TS สามารถวิ่งด้วยความเร็ว 44.736 เมกกะบิตต่อวินาที

เครือข่ายที่ต้องการเชื่อมโยงโดยตรงกับอินเทอร์เน็ตจะต้องกับแบคโบน ด้วยอุปกรณ์ที่เรียกว่า “เกตเวย์” (Gateway) ซึ่งเป็นเครื่องคอมพิวเตอร์ที่ทำหน้าที่ควบคุมการผ่านเข้า-ออกของข่าวสารระหว่างเครือข่ายกับแบคโบน เกตเวย์ทุกตัวสามารถกำหนดการติดต่อกับเกตเวย์ตัวอื่น หรือเครือข่ายอื่นได้ โดยใช้ไอพีแอดเดรส (IP Address) ของเครือข่าย อ้างอิงถึงกัน เช่น อินเทอร์เน็ตประเทศไทยเป็นเกตเวย์ของเครือข่ายในกลุ่มติดต่อกับเกตเวย์ของ UUNET ที่รัฐเวอร์จิเนียสหรัฐอเมริกา

ข่าวสารจากเครือข่ายถูกส่งออกไปผ่านเกตเวย์เข้าสู่อินเทอร์เน็ต โดยที่เกตเวย์เป็นตัวเลือกทิศทางการเดินทางเพื่อไปยังปลายทางที่ต้องการ แต่ตามเส้นทางอาจจะต้องผ่านเกตเวย์อีกหลายตัว เพื่อรับช่วงส่งข่าวสารจนถึงที่หมาย ถึงแม้ว่าจะต้องเดินทางระยะไกลก็ตาม แต่ด้วยสายสื่อสารความเร็วสูงทำให้การส่งข่าวสารทำได้อย่างรวดเร็ว

2.1.3 ข้อกำหนดรูปแบบของเกตเวย์ (Gateway Protocols)

เกตเวย์ต้องมีข้อมูลของเกตเวย์ตัวอื่นและรู้จักเครือข่ายปลายทาง เพื่อนำมาใช้ในการกำหนดเส้นทางที่ข่าวสารสามารถเดินทางไปถึงได้เร็วที่สุด เกตเวย์จะมีการแลกเปลี่ยนข้อมูลระหว่างกันเกี่ยวกับเส้นทาง รายละเอียดสถานะเครือข่าย และคุณสมบัติเครือข่ายย่อยที่ติดต่อเข้าสู่เครือข่ายใหญ่ตามลำดับชั้น จึงต้องมีการกำหนดรูปแบบพิเศษสำหรับเกตเวย์ขึ้นมา

ข้อกำหนดรูปแบบเกตเวย์ แบ่งออกตามการใช้งานได้ดังนี้

1. IGP (Interior Gateway Protocol) ถูกนำใช้กับเกตเวย์ที่อยู่ในเครือข่ายถูก ติดต่อกับเครือข่ายที่ถูกที่อยู่ภายในเครือข่ายแม่เดียวกัน หรือติดต่อกันระหว่างเครือข่ายแม่กับเครือข่ายลูก การเชื่อมโยงเกตประเภทนี้มักจะไม่ค่อยมีการเปลี่ยนแปลง จึงเรียกว่า ระบบอิสระจากกัน (Autonomous) หรือ Selfcomplete

2. EGP (External Gateway Protocol) ในเครือข่ายใหญ่ๆ การติดต่อกับเครือข่ายอื่นจะมีการเปลี่ยนแปลงอยู่เสมอ ตามการเปลี่ยนแปลงของเครือข่ายย่อยที่เชื่อมโยงอยู่เป็นจำนวนมาก จึงมีข้อกำหนดรูปแบบที่ใช้กับการสื่อสารระหว่างเกตเวย์ของเครือข่ายด้วย EGP

3. GGP (Gateway - to - Gateway Protocol) การเดินทางของข่าวสารระยะไกลบนแบคโบน อาจจะต้องผ่านเกตเวย์หลายตัวกว่าจะถึงปลายทาง GGP เป็นข้อกำหนดรูปแบบของการสื่อสารระหว่างเกตเวย์บนแบคโบน เพื่อให้การจราจรบนแบคโบน ไม่ติดขัด ข่าวสารเคลื่อนที่ไปได้รวดเร็ว

2.2 โมเดลอ้างอิง OSI

เพื่อลดปัญหาในความยุ่งยากสับสนในการติดต่อสื่อสารข้อมูล โครงสร้างของการสื่อสารข้อมูลภายในอุปกรณ์คอมพิวเตอร์ส่วนใหญ่จะถูกแบ่งเป็นชั้นๆ โดยแต่ละชั้นมีอิสระไม่ขึ้นต่อกันทำให้การเปลี่ยนแปลงบริการของชั้นใดๆ ไม่ก่อปัญหากับบริการชั้นอื่น การเพิ่มเติมการบริการใหม่ทำได้โดยไม่ต้องเปลี่ยนแปลงโปรแกรมระบบเดิมและสิ่งที่สำคัญก็คือ การทำระดับชั้นนั้นทำให้ตัวโปรแกรมมีขนาดเล็ก สามารถระบุส่วนที่จะต้องปรับปรุงได้แน่นอน ไม่ต้องวิตกกังวลถึงโปรแกรมส่วนอื่น ทำให้การพัฒนาประสิทธิภาพของระบบทำได้ง่ายและดียิ่งขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ISO (International Standardization for Organization) ซึ่งเป็นองค์กรที่จัดตั้งขึ้นมาเพื่อดูแลและส่งเสริมตลอดจนกำหนดมาตรฐานของการติดต่อสื่อสารของระบบเครือข่ายคอมพิวเตอร์ ได้พัฒนาสถาปัตยกรรมข้อกำหนดรูปแบบสำหรับเครือข่าย ซึ่งเป็นลักษณะระบบเปิดที่เรียกว่า ‘Open System Interconnection Model’ (OSI) โดยโมเดล OSI นี้มีลักษณะเป็นสถาปัตยกรรมแบบระบบเปิด (Open System) เพราะมุ่งที่จะให้ระบบคอมพิวเตอร์ในหลายๆรูปแบบที่แตกต่างกันสามารถเชื่อมต่อกันได้ OSI โมเดลได้แบ่งโปรโตคอล (Protocol) ในการสื่อสารออกเป็น 7 เลเยอร์ (Layer) ซึ่งโปรโตคอล คือ ชุดหรือข้อตกลงในการติดต่อ ข้อสังเกตโมเดล OSI เป็นเพียงข้อเสนอแนะ มิใช่ข้อกำหนด และควรรู้อย่างยิ่งที่ไม่มีระบบการเชื่อมต่อที่สร้างเหมือนกับโมเดล OSI จริง

ระดับชั้น	ชื่อระดับชั้น	ระดับหน้าที่
7	Application	Upper Layers
6	Presentation	
5	Session	
4	Transport	Lowery Layers
3	Network	
2	Data Link	
1	Physical	

ตาราง 2.1 OSI โมเดล

ข้อกำหนดรูปแบบแต่ละระดับชั้นของ OSI มีการระบุหน้าที่ต้องปฏิบัติแน่นอน โดยจะต้องเป็นตามกลุ่มการบริการ ดังนี้

1.ระดับชั้น 7 Application Layer เป็นเลเยอร์ชั้นบนสุดของรูปแบบ OSI ซึ่งเป็นชั้นที่ติดต่อกันระหว่างผู้ใช้โดยตรงซึ่งได้แก่ โฮสต์คอมพิวเตอร์ เทอร์มินัลหรือคอมพิวเตอร์ส่วนบุคคล(PC) เป็นต้น แอปพลิเคชันในเลเยอร์นี้ สามารถนำเข้าหรือออกจากระบบเครือข่ายได้โดยไม่ต้องสนใจจะมีขั้นตอนการทำงานอย่างไร เพราะจะมีชั้นพีรีเซนชันเป็นผู้รับผิดชอบอยู่แล้ว ในรูปแบบ OSI นี้ ชั้นแอปพลิเคชันจะการติดต่อกับชั้นพีรีเซนชันโดยตรง

2.ระดับชั้น 6 Presentation Layer คอยรวบรวมข้อความและแปลงรหัสหรือแปลงรูปแบบของข้อมูลที่ได้รับมาจากระดับชั้น 7 เพื่อให้เป็นรูปแบบการสื่อสารเดียวกัน เพื่อช่วยลดปัญหาต่างๆที่อาจจะเกิดขึ้นกับผู้ใช้งานในระบบ

3.ระดับชั้น 5 Session Layer ทำหน้าที่เชื่อมโยงระหว่างผู้ใช้คอมพิวเตอร์เครื่องอื่นๆ โดยผู้ใช้จะใช้คำสั่งหรือข้อความที่กำหนดไว้ป้อนเข้าไปในระบบ ในการสร้างการเชื่อมโยงนี้ผู้ใช้จะต้องกำหนดรหัสตำแหน่งของจุดหมายปลายทางที่ต้องการติดต่อสื่อสารด้วย ในบางครั้งเครือข่ายอาจมีชั้นเซสชันและชั้นทรานสปอร์ตรวมเป็นชั้นเลเยอร์ชั้นเดียวกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. ระดับชั้น 4 Transport Layer บางครั้งเรียกว่าชั้น เครื่องต่อเครื่อง (Host-to-Host) และจากเลเยอร์ชั้นที่ 4 ถึงชั้นที่ 7 นี้เรียกรวมกันว่า เลเยอร์เอนทูเอน (End-to-End) ในเลเยอร์ชั้นทรานสปอร์ต (Transport) นี้จะเป็นการสื่อสารระหว่างต้นทางและปลายทางกันจริงๆ สามารถแบ่งการบริการเป็น 2 กรณีคือ

กรณีที่หนึ่ง ส่งข้อมูลของระดับชั้น 5 โดยนำมาใส่ข้อมูลของปลายทาง เช่น ที่อยู่แล้วส่งออกไป โดยแบ่งข่าวสารยาวๆ ให้เป็นหน่วยที่เล็กลง เรียกว่าแพ็คเกจ (Packet) หรือเฟรม (Frame)

กรณีที่สอง เป็นบริการที่ทำการถอดรหัสข้อมูล ที่รับเข้ามาพิจารณาตัดสินใจว่ามีความสัมพันธ์ เกี่ยวข้องกับข้อมูลที่ได้ส่งออกหรือไม่ อย่างไร

เลเยอร์ชั้น ทรานสปอร์ต จะทำหน้าที่ตรวจสอบว่าข้อมูลที่ส่งมาจากชั้นเลเยอร์ที่ 5 นั้น ไปถึงปลายทางจริงหรือไม่ ดังนั้นการกำหนดตำแหน่งของข้อมูลจึงสำคัญในชั้นนี้

5. ระดับชั้น 3 Network Layer เป็นชั้นที่ออกแบบหรือกำหนดเส้นทางการเดินทางของข้อมูลที่ส่ง-รับในการส่งผ่านข้อมูลระหว่างต้นทางและปลายทาง ซึ่งเป็นที่แน่นอนว่าในการสื่อสารข้อมูลผ่านเครือข่ายการสื่อสารจะต้องมีเส้นทางการรับ-ส่ง ข้อมูลมากกว่า 1 เส้นทาง ดังนั้นในชั้นนี้จะมีหน้าที่เลือกเส้นทางที่ใช้เวลาในการสื่อสาร น้อยที่สุดและระยะทางสั้นที่สุดด้วย

6. ระดับชั้น 2 Data Link Layer เป็นเสมือนผู้ควบคุมการรวมบิตเข้าเป็นตัวอักษรและจัดข้อมูลให้เป็นแพ็คเกจ รวมทั้งตรวจสอบความผิดพลาดในการส่งข้อมูล ถ้าผู้รับได้รับข้อมูลถูกต้องก็จะส่งสัญญาณยืนยันกลับมาว่าได้รับข้อมูลแล้ว เรียกว่า สัญญาณ ACK (Acknowledge) ให้กับผู้ส่ง แต่ถ้าผู้ส่งไม่ได้รับสัญญาณ ACK หรือได้รับสัญญาณ NAK (Negative Acknowledge) กลับมา ผู้ส่งอาจจะทำการส่งข้อมูลไปให้ใหม่ อีกหน้าที่หนึ่งของชั้นนี้ คือ ป้องกันไม่ให้เครื่องส่งทำการส่งข้อมูลเร็วจนเกินขีดความสามารถของเครื่องผู้รับข้อมูลจะรับได้

7. ระดับชั้น 1 Physical Layer เป็นระดับชั้นการติดต่อระหว่างอุปกรณ์กับสื่อกลางของเครือข่ายจริงๆ การส่งข้อมูลระดับชั้นนี้ข้อมูลมีลักษณะเป็นบิต สื่อกลางที่ใช้ อาจจะเป็นสายโทรศัพท์ ไฟเบอร์ออฟติก ไมโครเวฟหรือดาวเทียม ตามระบบที่ใช้ โดยมาตรฐานสำหรับเลเยอร์ชั้นนี้จะกำหนดว่าแต่ละคอนเนคเตอร์ เช่น RS-232-C มีกี่พิน แต่ละพินทำหน้าที่อะไรบ้าง ใช้สัญญาณไฟฟ้าโวลต์ เทคนิคการมัลติเพล็กซ์แบบต่างๆ ก็จะถูกกำหนดอยู่ในชั้นเลเยอร์นี้

ในการสื่อสารกันระหว่างระบบหรือระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ โปรโตคอลแบบหนึ่งก็จะถูกใช้สำหรับการสื่อสารระหว่างชั้น N ของคอมพิวเตอร์เครื่องหนึ่งกับชั้น N เดียวกันของคอมพิวเตอร์อีกเครื่องหนึ่ง ซึ่งในระหว่างการสื่อสารข้อมูลกันจริงๆ นั้น โปรโตคอลชั้น N ของอุปกรณ์ทั้งสองเครื่องจะสื่อสารกันผ่านชั้นล่างซึ่งเป็นกายภาพหรือวัตถุ (สื่อกลางการสื่อสาร) แต่โดยแนวความคิดของการสื่อสารข้อมูลแล้วถือว่าชั้นที่ N ของเครื่องหนึ่งกำลังติดต่อโดยตรงกับชั้น N ของอีกเครื่องหนึ่ง ซึ่งเรียกการสื่อสารแบบนี้ว่า การสื่อสารแบบเสมือนจริง (Virtual Communication)

โปรโตคอลของในเลเยอร์แต่ละชั้นจะแตกต่างกันออกไป แต่อย่างไรก็ตามการที่เครื่องคอมพิวเตอร์หลายๆเครื่องจะติดต่อสื่อสารกันได้ ในแต่ละเลเยอร์ของแต่ละเครื่องจะต้องใช้โปรโตคอลแบบเดียวกัน หรือถ้าใช้โปรโตคอลต่างชนิดกันก็ต้องมีอุปกรณ์ หรือซอฟต์แวร์ที่สามารถแปลงโปรโตคอลที่ต่างกันนั้นให้มีรูปแบบอย่างเดียวกันเพื่อเชื่อมโยงให้คอมพิวเตอร์ทั้ง 2 เครื่องสามารถติดต่อกันได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในหนึ่งชั้นของเลเยอร์ไม่ได้กำหนดว่าจะต้องมีเพียงหนึ่งโปรโตคอลเท่านั้นที่อยู่ระดับเลเยอร์เดียวกัน และในทางตรงข้าม ชุดของโปรโตคอลใดๆ อาจจะมีมากกว่าหนึ่งเลเยอร์ประกอบกันเป็นข้อกำหนดของระบบเครือข่าย เรียกว่า ชุดโปรโตคอล (Protocol Suite) เช่น ชุดโปรโตคอล TCP/IP (Transmission Control Protocol/Internet Protocol) เป็นต้น ประโยชน์ในการแบ่งเป็นเลเยอร์ คือ กำหนดการติดต่อระหว่างเลเยอร์ทำ โดยไม่ต้องคำนึงถึงการเปลี่ยนในเลเยอร์ใดๆ ที่ติดกัน

2.3 TCP/IP

อินเทอร์เน็ตนับได้ว่าเป็นเครือข่ายบนเครือข่ายที่เปิดโอกาสให้เครือข่ายคอมพิวเตอร์อื่นเชื่อมโยงเข้ามาใช้งาน หรือเป็นศูนย์กลางเชื่อมโยงเครือข่ายคอมพิวเตอร์อื่นๆ แต่ปัญหาที่เกิดขึ้นในการเชื่อมโยงเครือข่ายคอมพิวเตอร์เข้าด้วยกันก็คือ แต่ละเครือข่ายใช้คอมพิวเตอร์ต่างชนิด ต่างยี่ห้อ และระบบปฏิบัติการที่ต่างกัน มาตรฐานของ TCP/IP จึงถูกใช้เป็นกฎเกณฑ์สำคัญในการแก้ไขปัญหาเหล่านี้ โดยกลายเป็นระบบเปิดที่สมบูรณ์แบบที่มีการเชื่อมโยงคอมพิวเตอร์ได้ตั้งแต่เครื่องพีซี จนถึงเครื่องเมนเฟรม และไม่จำกัดระบบปฏิบัติการที่ใช้ TCP/IP จึงเป็นมาตรฐานที่ทั่วโลกยอมรับ มีอุปกรณ์และซอฟต์แวร์ผลิตออกมาสนับสนุน TCP/IP มากมาย ดังนั้นจึงนับได้ว่า TCP/IP เป็นหัวใจของอินเทอร์เน็ตเลยทีเดียว

2.3.1 TCP/IP คืออะไร

TCP/IP เป็นข้อกำหนดเกี่ยวกับรูปแบบการเชื่อมโยงในเครือข่าย (Networking Protocol) จัดทำเพื่อใช้เป็นกฎเกณฑ์ให้เครื่องคอมพิวเตอร์ใช้งานร่วมกัน ในลักษณะของระบบเปิด (Open System) คือ ไม่ว่าจะเป็นคอมพิวเตอร์ชนิดใดหรือระบบใดก็ตาม จะสามารถติดต่อสื่อสารและแลกเปลี่ยนข้อมูลกันได้

TCP/IP เป็นการกำหนดรูปแบบการสื่อสารระหว่างซอฟต์แวร์ การจัดการโอนย้ายข้อมูล การแสดงสถานะของเครื่องคอมพิวเตอร์ที่อยู่บนเครือข่าย ตลอดจนกฎระเบียบต่างๆ ที่กำหนดให้ทำ เมื่อเกิดความผิดพลาด หรือต้องทำเพื่อป้องกันไม่ให้เกิดความผิดพลาด

TCP/IP เกิดจากการนำข้อกำหนดของรูปแบบต่างๆ กันมาใช้ร่วมกัน TCP และ IP ต่างก็เป็นรูปแบบหนึ่งของชุดข้อกำหนดนี้ (แต่เรียกชุดข้อกำหนดรูปแบบนี้ว่า TCP/IP) ถูกออกแบบมาเพื่อใช้รับส่งหรือโอนย้ายข้อมูลระหว่างคอมพิวเตอร์ที่อยู่บนระบบเครือข่ายเดียวกัน หรือต่างเครือข่ายกันก็ได้ และมีการจัดเตรียมข้อมูลสถานะของเครือข่ายขึ้นได้ภายในตัวข้อกำหนดรูปแบบเอง ในการสร้างซอฟต์แวร์ของระบบเครือข่ายจะใช้ TCP/IP เป็นส่วนสนับสนุนได้ทั้งระบบเครือข่ายเฉพาะบริเวณ (Local Area Network) และเครือข่ายบริเวณกว้าง (Wide Area Network) ไม่ได้ใช้งานเฉพาะกับอินเทอร์เน็ตเท่านั้น

2.3.2 ส่วนประกอบของ TCP/IP

จากที่กล่าวมาแล้วข้างต้นว่า TCP/IP ประกอบไปด้วยชุดข้อกำหนดรูปแบบต่างๆ ซึ่งแบ่งเป็นกลุ่มได้ดังนี้

2.3.2.1 กลุ่มข้อกำหนดรูปแบบการขนส่ง (Transport Protocols) ทำหน้าที่ควบคุมการเคลื่อนย้ายข้อมูลระหว่างคอมพิวเตอร์สองเครื่อง แบ่งย่อยออกได้เป็นสองชนิด คือ

1. TCP (Transmission Control Protocol) เป็นการบริการแบบ Connection Based Service ซึ่งคอมพิวเตอร์ด้านผู้รับและผู้ส่งต้องต่อกันอยู่ตลอดเวลาในระหว่างการสื่อสาร ถ้าเปรียบเทียบกับคล้ายกับระบบโทรศัพท์ที่ต้องติดต่อกันให้ได้ ก่อนจะพูดคุยกันได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. UDP (User Datagram Protocol) เป็นการให้บริการแบบ Connection Service คอมพิวเตอร์ด้านผู้ส่งไม่จำเป็นต้องติดต่อกับด้านผู้รับก่อน เพียงรู้ที่อยู่ของด้านผู้รับแล้วใส่ที่อยู่นั้นไปกับข้อมูลที่ส่งออก ข้อมูลจะเดินทางตามเส้นทางต่างๆเพื่อไปถึงปลายทางตามที่อยู่ คล้ายกับการส่งจดหมายที่ไปรษณีย์ จะส่งให้ตามที่อยู่ที่กำหนดของจดหมาย โดยผู้ส่งและผู้รับไม่ต้องติดต่อกัน

2.3.2.2 กลุ่มข้อกำหนดเกี่ยวกับรูปแบบเส้นทาง (Routing Protocol) ทำหน้าที่พิจารณาเส้นทางที่ดีที่สุดที่ใช้ส่งข้อมูลและถ้ามีข้อมูลเป็นจำนวนมากหรือมีขนาดใหญ่ กลุ่มข้อมูลรูปแบบนี้ก็จะทำการแบ่งย่อยข้อมูลให้มีขนาดเหมาะสม แล้วส่งออกไป เมื่อถึงผู้รับปลายทาง กลุ่มข้อมูลนี้ก็จะทำหน้าที่ตรงข้าม คือ รวบรวมข้อมูลย่อยให้ถูกต้องก่อนการแสดงผล กลุ่มข้อกำหนดรูปแบบกลุ่มนี้ ประกอบด้วย

1. IP (Internet Protocol) เป็นการกำหนดรูปแบบการส่งข้อมูล
2. ICMP (Internet Control Message Protocol) เป็นข้อกำหนดรูปแบบของข้อมูลข่าวสารเกี่ยวกับสถานะของ IP เช่น ข่าวสารความผิดพลาดและผลกระทบต่อเส้นทางเมื่อมีการเปลี่ยนแปลงฮาร์ดแวร์ในเครือข่าย
3. RIP (Routing Information Protocol) ข้อกำหนดรูปแบบหนึ่งที่ใช้สำหรับทำการพิจารณาวิธีการเลือกเส้นทางเพื่อให้ได้เส้นทางที่ดีที่สุด เหมาะสมกับข้อมูลมากที่สุด
4. OSPF (Open Shortest Path First) ข้อกำหนดรูปแบบอีกประเภทหนึ่งที่ใช้ตัดสินใจเลือกเส้นทาง โดยพิจารณาจากเส้นทางที่สั้นที่สุดก่อน

2.3.2.3 กลุ่มข้อกำหนดรูปแบบเกี่ยวกับที่อยู่เครือข่าย (Network Address) ทำหน้าที่พิจารณาที่อยู่ของเครือข่ายและเครื่องคอมพิวเตอร์ไม่ว่าจะเป็นลักษณะตัวเลขหรือชื่อก็ตาม เพื่อความถูกต้องของข้อมูลที่จะไปยังผู้รับปลายทาง โดยที่ไม่ว่าเครือข่ายจะใหญ่โตสักเพียงใดหรือมีเครื่องคอมพิวเตอร์จำนวนมากก็ตาม ที่อยู่จะต้องไม่ซ้ำกัน กลุ่มข้อกำหนดรูปแบบกลุ่มนี้มีดังนี้

1. ARP (Address Resolution Protocol) ข้อกำหนดรูปแบบที่พิจารณาตัวเลขที่อยู่เพื่อไม่ให้เกิดที่อยู่ซ้ำกัน
2. DNS (Domain Name System) ข้อกำหนดรูปแบบที่พิจารณาตัวเลขที่อยู่เมื่อรู้ชื่อของเครือข่ายหรือเครื่องคอมพิวเตอร์ เพราะการใช้งานจริงนั้นใช้เพียงที่อยู่ที่เป็นตัวเลข แต่ระบบชื่อจัดทำขึ้นเพื่อให้สะดวกต่อการใช้งานของผู้ใช้
3. RARP (Reverse Address Resolution Protocol) ข้อกำหนดรูปแบบที่พิจารณาตัวเลขที่อยู่เช่นเดียวกับ ARP แต่จะทำตรงข้ามกับ ARP

2.3.2.4 กลุ่มข้อกำหนดรูปแบบเกี่ยวกับเส้นทางการสื่อสารระหว่างเครือข่าย (Gateway Protocols) และสนับสนุนข้อมูลสถานะเพื่อนำไปใช้เลือกเส้นทางที่เหมาะสม ข้อกำหนดรูปแบบเหล่านี้ประกอบด้วย

1. EGP (Exterior Gateway Protocol) ข้อกำหนดรูปแบบนี้จะทำการถ่ายโอนข้อมูลเส้นทางกันระหว่างเกตเวย์ กับเครือข่ายภายนอกเพื่อทำการสื่อสาร
2. GGP (Gateway- to-Gateway Protocol) เป็นข้อกำหนดรูปแบบที่ทำงานถ่ายโอนข้อมูลเส้นทางกันระหว่างเกตเวย์ กับเกตเวย์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. IGP (Interior Gateway Protocol) ข้อกำหนดรูปแบบที่ถ่ายโอนข้อมูลเส้นทางกันภายในเครือข่ายเดียวกัน

2.3.2.5 กลุ่มข้อกำหนดรูปแบบเกี่ยวกับการบริการผู้ใช้ (User Services) ผู้ใช้สามารถใช้ข้อกำหนดรูปแบบได้โดยตรง ข้อกำหนดรูปแบบนี้ประกอบด้วย

1. BOOTP (BOOT Protocol) เมื่อผู้ใช้เปิดเครื่องคอมพิวเตอร์บนเครือข่ายให้เริ่มทำงาน ข้อกำหนดรูปแบบนี้จะอ่านโปรแกรมควบคุมการทำงานจากคอมพิวเตอร์ให้บริการ (Server Computer) มาให้
2. FTP (File Transfer Protocol) ข้อกำหนดรูปแบบที่ให้บริการถ่ายโอนไฟล์ข้อมูลระหว่างคอมพิวเตอร์ซึ่งอาจจะอยู่บนเครือข่ายเดียวกันหรือต่างเครือข่ายกันก็ได้
3. TELNET เป็นข้อกำหนดรูปแบบที่ให้บริการเกี่ยวกับการควบคุมการติดต่อระยะทางไกล (Remote Login)

กลุ่มข้อกำหนดรูปแบบอื่นที่นอกเหนือจากกลุ่มที่จัดไว้และบริการที่สำคัญๆ จัดทำไว้ให้บนเครือข่ายที่สนใจมีดังนี้

1. NFS (Network File System) เป็นข้อกำหนดรูปแบบที่ทำให้ผู้ใช้คอมพิวเตอร์เครื่องหนึ่งสามารถเข้าไปดูไฟล์ข้อมูลและใช้งานไฟล์ข้อมูลซึ่งอยู่ในคอมพิวเตอร์เครื่องอื่นได้
2. NIS (Network Information Services) เป็นข้อกำหนดรูปแบบที่ให้บริการกับ User Accounts ข้ามเครือข่าย เช่น Logins และ Password
3. RPC (Remote Procedure Call) ข้อกำหนดรูปแบบที่อำนวยความสะดวกให้กับ โปรแกรมประยุกต์ที่ใช้งานกับการควบคุมระยะทางไกล

SMTP (Simple Mail Transfer Protocol) ข้อกำหนดรูปแบบที่ให้บริการถ่ายโอนจดหมายอิเล็กทรอนิกส์ (Electronic Mail) ระหว่างคอมพิวเตอร์

4. SNMP (Simple Network Management Protocol) ข้อกำหนดรูปแบบที่ให้บริการข่าวสารต่างๆ ที่แสดงสถานะของเครือข่ายและอุปกรณ์ที่ต่ออยู่บนเครือข่าย

2.3.3 โครงสร้างของชุดโปรโตคอล TCP/IP

OSI Model	TCP/IP (Internet)
Application	Application
Presentation	
Session	
Transport	Transport
Network	Internet
Data Link	Network Interface
Physical	Physical

เอกสารนี้เป็นตาราง 2.2 โครงสร้างของชุดโปรโตคอล TCP/IP เปรียบเทียบกับ OSI Model ใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. Application Layer ในชั้นนี้ประกอบด้วยโปรแกรมประยุกต์ที่ใช้ในเครือข่าย เช่น โปรแกรมส่งถ่ายข้อมูล (File-Transfer Program) และอาจกล่าวได้ว่าโปรโตคอล TCP/IP ก็คือโปรโตคอลในชั้นแอปพลิเคชันร่วมกับชั้นพีรีเซนเตชันของ OSI โมเดลนั่นเอง

2. Transport Layer ในชั้นนี้เป็นชั้นที่ให้การส่งข้อมูลจากจุดปลายเปรียบเทียบกับชั้นเซสชันรวมกับทรานสปอร์ตเลเยอร์นั่นเอง โดยโปรโตคอล TCP/IP มีซ็อกเก็ต (Socket) เป็นจุดปลาย (End-Point) ในการสื่อสาร ซึ่งซ็อกเก็ตนี้ประกอบไปด้วยหมายเลขของคอมพิวเตอร์ และหมายเลขพอร์ต (Port) ของเครื่องที่ต้องการส่งข้อมูลไปถึง ในชั้นนี้มีการรับรองให้ถึงที่หมาย และลำดับข้อมูลที่ส่งโดยปราศจากข้อมูลซ้ำซ้อน โดยในชั้นนี้มีโปรโตคอลหลัก 2 ตัว คือ TCP และ UDP

3. Internet Layer ในชั้นนี้มีการกำหนดค่าแกรม (Datagram) และทำการหาเส้นทางในการส่ง การทำงานในชั้นนี้จะแบบ Connectionless เนื่องจากไม่มีการเชื่อมต่อระหว่างต้นทางกับปลายทางก่อน ค่าแกรมแต่ละตัวสามารถเลือกเส้นทางไปโดยอิสระ และไม่มีการรับประกันความถูกต้องข้อมูลหรือลำดับการส่ง

4. Network Interface Physical Layer ทำหน้าที่ควบคุมตัวกลางที่ใช้สื่อสารข้อมูลและรูปแบบการเชื่อมต่อในทางกายภาพ ชั้นนี้จะแบ่งข้อมูลออกเป็นส่วนๆ เรียกว่า เฟรม หรือ แพ็คเก็ตและส่งข้อมูลที่ได้ไปยังปลายทางที่เชื่อมต่อกันอยู่บนเครือข่ายเดียวกัน

2.3.4 ข้อแตกต่างระหว่างชุดโปรโตคอล TCP/IP

1. ลำดับการติดต่อสื่อสารของชั้นเลเยอร์ ในรูปแบบ OSI นั้นจะกำหนดลำดับชั้นการสื่อสารที่เป็นลำดับขั้นตอนการติดต่อที่แน่นอน โดยเฉพาะการอินเตอร์เฟสระหว่างชั้นเลเยอร์ ซึ่งทำให้รูปแบบ OSI สามารถเป็นระบบเปิดสำหรับระบบเครือข่ายคอมพิวเตอร์ทั่วไป เพราะไม่ว่าจะมีการเปลี่ยนแปลงโปรโตคอลในเลเยอร์ชั้นใดก็ตามจะไม่มีผลกระทบต่อการสื่อสารเลเยอร์ชั้นถัดไป ในขณะที่ชุดโปรโตคอล TCP/IP จะไม่มีการกำหนดรูปแบบการติดต่อที่ตายตัว เพื่อให้ผู้ออกแบบเครือข่ายมีอิสระสามารถเปลี่ยนแปลงโครงสร้างของเครือข่ายได้ง่ายขึ้น

2. การติดต่อสื่อสารระหว่างเครือข่ายหรือการอินเทอร์เน็ต คือ การติดต่อสื่อสารข้อมูลระหว่างระบบคอมพิวเตอร์ 2 ระบบที่ไม่สามารถติดต่อสื่อสารกันได้โดยผ่านทางเครือข่ายการสื่อสารข้อมูลเพียงเครือข่ายเดียวได้ต้องอาศัยเครือข่ายตั้งแต่ 2 เครือข่ายขึ้นไปในการติดต่อสื่อสารกัน และเครือข่ายเหล่านี้อาจจะมีลักษณะของเครือข่ายที่ต่างกันก็ได้

ความแตกต่างในเรื่องของอินเทอร์เน็ตระหว่างชุดโปรโตคอล TCP/IP กับรูปแบบ OSI ก็คือในชุดโปรโตคอล TCP/IP จะใช้โปรโตคอลสำหรับอินเทอร์เน็ตที่เรียกว่า โปรโตคอล IP (Internet Protocol) ซึ่งในรูปแบบ OSI จะเรียกว่าโปรโตคอลสำหรับการอินเทอร์เน็ตว่า โปรโตคอลเน็ตเวิร์ค

3. การบริการการเชื่อมต่อสื่อสาร (Connection Service) ในชุดโปรโตคอล TCP/IP นั้นจะมีการบริการการเชื่อมต่อสื่อสารระหว่างต้นทางและปลายทาง 2 แบบ คือการบริการแบบ Connectionless และแบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Connection-Oriented ส่วนในรูปแบบ OSI จะให้ความสำคัญเฉพาะกับการบริการแบบ Connection-Oriented เท่านั้น

4. โปรโตคอลควบคุมการจัดการสื่อสาร ในชุดโปรโตคอล TCP/IP จะใช้โปรโตคอล TCP เป็นโปรโตคอลสำหรับควบคุมการสื่อสาร กำหนดตำแหน่งต้นทางและปลายทาง และอื่นๆกับข้อมูล ซึ่งในรูปแบบ OSI นั้นจะแบ่งแยกการควบคุมการสื่อสารออกจากกันโดยใช้โปรโตคอลเซสชันและโปรโตคอลทรานสปอร์ตตามลำดับ

2.3.5 ลักษณะของการติดต่อ

แบ่งออกเป็น 2 ชนิด คือ

Connection-Oriented คือ การติดต่อที่ต้องมีการเชื่อมต่อโปรเซส (Process) ที่จะมีการส่งหรือรับข้อมูล ซึ่งใช้คำว่าวงจรเสมือน (Virtual Circuit) เพราะว่าจะทำงานเสมือนมีวงจรต่ออยู่ระหว่างโปรเซส ถึงแม้ว่าข้อมูลนี้อาจจะผ่าน Packet-Switching Network บริการชนิดนี้ส่วนมากจะใช้ในกรณีที่มีข่าวสารต้องการมากกว่าหนึ่งข่าวสาร ดังนั้นสามารถแบ่งขั้นตอนการทำงานออกเป็น

- ขั้นตอนการสร้างการติดต่อ (Connection Establishment)
- ขั้นตอนการส่งผ่านข้อมูล (Data Transfer)
- ขั้นตอนยกเลิกการติดต่อ (Connection Termination)

Connectionless หรือคำด้าแกรม คือ จะไม่มีขั้นตอนการสร้างการติดต่อ และขั้นตอนยกเลิกการติดต่อ แต่จะมีขั้นตอนการส่งผ่านข้อมูลเพียงอย่างเดียว โดยข้อมูลซึ่งเรียกว่าคำด้าแกรมจะถูกส่งจากระบบหนึ่งไปสู่ระบบหนึ่งอย่างเป็นอิสระโดยไม่ขึ้นอยู่กับคำด้าแกรมอื่น

2.3.6 ความสัมพันธ์ของ TCP และ UDP กับอินเทอร์เน็ต

ตามสถาปัตยกรรมของชุดข้อกำหนดรูปแบบ TCP/IP ระดับชั้น ทรานสปอร์ต จะมีบริการการส่งข้อมูลอยู่ 2 ประเภท คือ TCP และ UDP

ความแตกต่างของ TCP กับ UDP อยู่ที่วิธีการสื่อสารระหว่างเครื่องคอมพิวเตอร์สองเครื่อง TCP ต้องสร้างการติดต่อให้ได้ก่อนแล้วจึงส่งข้อมูลและยกเลิกเส้นทางที่ต่อกันเมื่อจบการส่งข้อมูล ส่วน UDP จะไม่มีการสร้างการติดต่อ แต่จะใช้ ไอพีแอดเรสของคอมพิวเตอร์ปลายทางใส่รวมกับข้อมูลแล้วส่งออกไป ข้อมูลจะเดินทางตามไอพีแอดเรสด้วยเส้นทางที่ไม่แน่นอน ขึ้นอยู่กับความเหมาะสมจนกว่าจะถึงคอมพิวเตอร์ปลายทาง

TCP เป็นวิธีการสื่อสารที่มีความเชื่อมั่นได้มากกว่า UDP ทั้งการส่งและการตอบรับข่าวสาร วิธีของ UDP จะไม่มีการรับประกันว่าข่าวสารจะไปถึงผู้รับ แต่ UDP มีวิธีการที่จะตรวจสอบโดยให้เครื่องคอมพิวเตอร์ปลายทางส่งข่าวสารการตอบรับกลับมาเมื่อได้รับข่าวสารที่ส่งไปให้ ทางด้านผู้ส่งจะตั้งเวลารอว่านานเท่าใดควรมีการตอบกลับรับกลับมา ถ้าเกินเวลา (Time-Out) แล้วยังไม่มีการตอบรับกลับมาก็สันนิษฐานได้ว่า ข่าวสารสูญหาย UDP ก็จะทำการส่งข่าวสารเดิมให้ใหม่ เครือข่ายขนาดใหญ่เช่นอินเทอร์เน็ต ซึ่งมีเครื่องคอมพิวเตอร์บนเครือข่ายเป็นจำนวนมาก การใช้ TCP ที่ต้องสร้างการติดต่อและยกเลิกอยู่บ่อยๆจึงเป็นไปได้ในทางปฏิบัติ ดังนั้นอินเทอร์เน็ต จึงใช้การส่งข่าวสารด้วย UDP แทน

การบริการที่ดำเนินการด้วยชุดข้อกำหนดรูปแบบ TCP/IP ต้องออกแบบการให้บริการเป็น TCP หรือ UDP อย่างใดอย่างหนึ่งเท่านั้น การบริการอย่างเดียวยจะให้บริการทั้งสองประเภทไม่ได้ เช่น การ Logins เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากระยะไกล (Telnet) และการโอนย้ายไฟล์ข้อมูล FTP ใช้การให้บริการแบบ TCP ส่วน TFTP (Trivial File Transfer Protocol) ใช้การให้บริการแบบ UDP

2.3.7 พอร์ตและซ็อกเก็ต (TCP Port and Socket)

การใช้บริการของ TCP ทำได้โดยผ่านทางพอร์ต โดยมีการกำหนดหมายเลขให้กับการบริการต่างๆ การบริการที่ต่างกันจะมีหมายเลขพอร์ตไม่เหมือนกัน

พอร์ตถูกกำหนดตามชนิดของการบริการ เช่น ต้องการสื่อสารกันด้วย Telnet เครื่องคอมพิวเตอร์ที่ต้องการบริการจะส่งความต้องการออกทางพอร์ตหมายเลข 23 ที่จัดไว้สำหรับบริการ Telnet ไปยังเครื่องคอมพิวเตอร์ที่ต้องการติดต่อด้วย ถ้าเครื่องคอมพิวเตอร์ปลายทางยอมรับการติดต่อ การสื่อสาร Telnet ก็จะเริ่มขึ้น ในช่วงเวลานี้ถ้ามีความต้องการการสื่อสารด้วย Telnet จากเครื่องคอมพิวเตอร์อื่นเข้ามา จะไม่สามารถทำการติดต่อได้เนื่องจากพอร์ตหมายเลข 23 สำหรับบริการ Telnet ถูกใช้งานอยู่ TCP พอร์ตเป็นพอร์ตทาง Logical ไม่ใช่พอร์ตที่อยู่ด้านหลังของเครื่องคอมพิวเตอร์ ใช้คอนเน็คเตอร์ (Connector) บน แบคโบน เพื่อให้สามารถสื่อสารบริการหลายประเภทได้พร้อมๆ กัน

ซ็อกเก็ตเป็นหมายเลขกำหนดวงจรซึ่งทางเข้า-ออก ของระดับชั้น TCP ของเครื่องคอมพิวเตอร์บนเครือข่าย หมายเลขซ็อกเก็ตประกอบด้วยเลข ไอพีแอดเดรส กับหมายเลขพอร์ตรวมกัน หมายเลขซ็อกเก็ตใช้กับการสื่อสารเครื่องคอมพิวเตอร์ที่อยู่ต่างเครือข่ายได้ เนื่องจาก ไอพีแอดเดรสของเครือข่ายจะไม่ซ้ำกัน ถึงแม้ว่าหมายเลขพอร์ตจะซ้ำกันได้

เครื่องคอมพิวเตอร์บนเครือข่ายจะสร้างตารางพอร์ตขึ้นมาเก็บรายการของพอร์ตที่มีอยู่เพื่อใช้ตรวจสอบว่าจะใช้บริการพอร์ตที่ต้องการได้หรือไม่ และพอร์ตมีสถานะเป็นอย่างไรการที่พอร์ตที่อยู่ในรายการตารางพอร์ตของแต่ละเครื่อง ร้องขอการสื่อสารซึ่งกันและกัน เรียกว่า 'Binding' ของพอร์ตและวิธีที่ทำให้พอร์ตที่ถูกใช้อยู่สามารถใช้สื่อสารได้อีกจากการขอใช้บริการเดียวกันเรียกว่า 'Multiplexing'

TCP พอร์ต	บริการ
20	FTP Data
21	FTP Control
23	Telnet
25	Simple Mail Transfer Protocol
53	Domain Name Server
69	Trivial File Transfer Protocol
79	Finger

ตาราง 2.3 หมายเลขพอร์ตที่ใช้บริการต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.8 ข้อกำหนดรูปแบบ IP (Internet Protocol)

IP เป็นข้อกำหนดรูปแบบหนึ่งในชุดข้อกำหนดรูปแบบ TCP/IP ลักษณะการให้บริการเป็นแบบ Connectionless เช่นเดียวกับ UDP มีหน้าที่หลักเกี่ยวกับการจัดการ ค้าค้าแกรม ระหว่างเครื่องคอมพิวเตอร์ เช่น ที่อยู่ปลายทาง วิธีการหาเส้นทางที่ดีที่สุดที่จะไปถึงปลายทาง และจัดเตรียมวิธีการแก้ปัญหาที่อาจเกิดขึ้นในกรณีต่างๆ เป็นต้น นอกจากนี้ยังมีหน้าที่แบ่งย่อยค้าค้าแกรมที่มีขนาดใหญ่เกินกว่าที่ IP Message กำหนดไว้ (ประมาณ 64 กิโลไบต์) และรวบรวม ค้าค้าแกรม ย่อยๆ กลับมาเป็น Message ให้เหมือนเดิมที่เครื่องปลายทาง การแบ่งย่อยและรวบรวมค้าค้าแกรม มีอยู่ 4 วิธีคือ

1. Segmentation วิธีแบ่งค้าค้าแกรมขนาดใหญ่ให้เป็นค้าค้าแกรมย่อย
2. Reassemble วิธีรวบรวมค้าค้าแกรมย่อยๆ ให้กลับอยู่ในสภาพเดิม
3. Concatenation วิธีรวมค้าค้าแกรมให้เป็นบล็อก
4. Separation วิธีแบ่งบล็อกให้เป็นค้าค้าแกรม

2.3.9 ข้อกำหนดรูปแบบ ICMP (Internet Control Message Protocol)

ปัญหาหนึ่งที่เกิดขึ้นได้ง่ายกับค้าค้าแกรม ก็คือ ค้าค้าแกรมเดินทางผิดเส้นทาง ทำให้สูญหายหรือบางส่วนข่าวสารเสียหาย เครื่องคอมพิวเตอร์ด้านส่งต้องรู้เงื่อนไขที่จะทำให้เกิดความผิดพลาดต่างๆบนเครือข่าย เพื่อให้ค้าค้าแกรมถูกส่งออกไปอย่างถูกต้อง

ICMP จะรายงานความผิดพลาดในระบบให้กับไอพีพิเศษ ICMP มี Header เหมือนกับข่าวสารไอพีอื่นๆ โดยจะถูกส่งรวมไปกับข่าวสารอื่นสู่เครื่องปลายทาง ซึ่งเครื่องปลายทางจะส่งรายงานของความผิดพลาดกลับมาให้เครื่องด้านส่งได้รู้ว่าเกิดอะไรขึ้นบ้างในการเดินทาง ซึ่งข้อมูลในรายงานจะถูกนำมาใช้แก้ปัญหาต่อไป

2.4 ไอพีแอดเดรส (IP Address)

เครื่องคอมพิวเตอร์ที่ต่อเชื่อมกับอินเทอร์เน็ตหรือเครือข่ายที่ใช้ TCP/IP มีอยู่เป็นจำนวนมาก เครื่องคอมพิวเตอร์แต่ละเครื่องต้องสามารถระบุ หรืออ้างอิงได้โดยไม่เกิดความซ้ำซ้อนกับเครื่องคอมพิวเตอร์อื่น มิฉะนั้นแล้วข่าวสารที่เครือข่ายรับมาจะไม่สามารถส่งไปให้กับเครื่องคอมพิวเตอร์ที่ต้องการข่าวสารนั้นได้ จึงต้องมีการจัดระบบที่ดี เครือข่ายอินเทอร์เน็ตหรือเครือข่ายที่ใช้ TCP/IP ได้ออกแบบการจัดการระบบตรงส่วนนี้ไว้แล้ว

รหัสเลขประจำเครื่อง IP

เครือข่ายอินเทอร์เน็ตใช้รหัสหมายเลขมากำหนดให้แต่ละเครือข่าย และเครื่องคอมพิวเตอร์ภายในเครือข่ายที่เชื่อมโยง เรียกรหัสหมายเลขเหล่านี้ว่าอินเทอร์เน็ตแอดเดรส (Internet Address)

ไอพีแอดเดรส ประกอบด้วยเลขฐานสองจำนวน 32 บิต แบ่งออกเป็น 4 ส่วน แต่ละส่วนมี 8 บิต เมื่อดูเฉพาะแต่ละส่วนเป็นเลขฐานสิบจะได้เลขจำนวน 256 ค่าไม่ซ้ำกัน (0-255) ไอพีแอดเดรสจะนำเอาหมายเลขทั้ง 4 ส่วนมารวมกัน โดยแยกแต่ละส่วนด้วยจุด ดังนั้นหมายเลขทั้งหมดที่เป็นไปได้ โดยค่าไม่ซ้ำกัน คือ 256^4 หรือ 4,294,967,296 จำนวน มีค่าหมายเลขจาก 000.000.000.000 จนถึง 255.255.255.255 หมายเลขเหล่านี้เองที่

อินเทอร์เน็ตใช้กำหนดให้กับเครือข่ายและเครื่องคอมพิวเตอร์ เพื่อใช้อ้างอิงถึง

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไอพีแอดเดรสบางหมายเลขสงวนไว้ใช้ ด้วยจุดหมายกรณีพิเศษ ทำให้ ไอพีแอดเดรสที่ใช้งานทั่วไป ลดลงจากจำนวนที่เป็นไปได้ ความหมายของ ไอพีแอดเดรสจะแบ่งได้เป็น 2 กลุ่มดังนี้

- 1. กลุ่มที่ใช้เป็นรหัสประจำเครือข่าย
- 2. กลุ่มที่ใช้เป็นรหัสประจำเครื่องคอมพิวเตอร์ที่อยู่ภายในเครือข่าย (Host Computers)

ไอพีแอดเดรสในกลุ่มรหัสประจำเครื่องคอมพิวเตอร์สามารถซ้ำกันได้ แต่กลุ่มรหัสประจำเครือข่ายจะ ซ้ำกันไม่ได้ ดังนั้นรหัสเครื่องที่ซ้ำกันจึงไม่มีผลต่อการอ้างอิงถึง

นอกจากนี้เพื่อความเหมาะสมในการกำหนด ไอพีแอดเดรสให้กับผู้ขอ ทางผู้บริหารอินเทอร์เน็ตแบ่ง คลาสของผู้ขอ ไอพีแอดเดรสตามขนาดของเครือข่าย เพื่อให้ทรัพยากรส่วนนี้ถูกใช้อย่างคุ้มค่าที่สุด องค์กร ขนาดใหญ่ก็จะจัดให้อยู่ในอยู่ในคลาสที่สามารถกำหนด ไอพีแอดเดรสให้กับเครื่องคอมพิวเตอร์ในเครือข่ายได้ มาก การแบ่งคลาสจะแบ่งได้

	MSB				LSB	
Class A	0	Network ID (7)			Host ID (24)	
Class B	1	0	Network ID (14)		Host ID (16)	
Class C	1	1	0	Network ID (21)		Host ID (8)
Class D	1	1	1	0	Multicast Address	
Class E	1	1	1	1	0	Reserved

รูป 2.2 แสดงคลาสของ ไอพีแอดเดรส

การกำหนดหมายเลขของเครือข่ายและเครื่องคอมพิวเตอร์ นอกจากจะแบ่ง ไอพีแอดเดรสเป็นคลาส ทั้ง 5 ประเภทแล้ว ยังมีข้อกำหนดปลีกย่อยอีกหลายประการที่ผู้วางระบบต้องรู้ไว้ เพื่อที่จะกำหนดแอดเดรสใช้ งานได้อย่างถูกต้องไม่ผิดพลาด ได้แก่ ไอพีแอดเดรสที่เป็น '0' ทุกบิต จะไม่ใช้งานทั่วไป แต่นำไปใช้กับ อุปกรณ์หาเส้นทาง(Router) เพื่อกำหนด 'Default Route'

- 1. ไอพีแอดเดรสในแต่ละส่วนคือ ส่วนหมายเลขเครือข่ายและส่วนหมายเลขประจำเครื่อง คอมพิวเตอร์จะเป็น '0' หรือ '1' ทุกบิตไม่ได้
 - ถ้าส่วนของหมายเลขประจำเครื่องคอมพิวเตอร์เป็น '0' ทุกบิต หมายถึง หมายเลขเครือข่าย นั้นใช้งานร่วมกับอุปกรณ์หาเส้นทางหรือเพื่อบอกให้เครื่องคอมพิวเตอร์รู้ว่าตัวเองอยู่ในเครือ ข่ายใด
 - ส่วนของหมายเลขประจำเครื่องคอมพิวเตอร์เป็น '1' ทุกบิต หมายถึง หมายเลขเครือข่ายนั้น ใช้สำหรับการกระจายข่าวภายในเครือข่าย (Broadcast Address)

- ถ้า ไอพีแอดเดรสทั้งสองส่วนเป็น '1' ทุกบิต หมายถึง แอดเดรสที่ใช้กระจายข่าวหรืออีกนัยหนึ่งเครื่องคอมพิวเตอร์ที่ส่งแอดเดรสนี้เข้าสู่ระบบ ไม่ทราบว่าตนเองอยู่ในเครือข่ายใด

3. ไอพีแอดเดรสของคลาส A หมายเลข 127.0.0.0 จะสงวนไว้ใช้งานเฉพาะอย่าง เช่น IPC (Inter-Process Communication)

จากรูป 2.2 จะแสดงประเภทของ ไอพีแอดเดรสและข้อกำหนดปลีกย่อยต่างๆ เราสามารถทราบได้ว่าเครือข่ายขององค์กรถูกจัดอยู่ใน Class ไດ โดยดูจากค่าหมายเลขของ 8 บิตแรกซ้ายมือสุด ดังนี้

Class A : ไอพีแอดเดรสอยู่ในช่วง 1 ถึง 126

Class B : ไอพีแอดเดรสอยู่ในช่วง 128 ถึง 191

Class C : ไอพีแอดเดรสอยู่ในช่วง 192 ถึง 223

2.5 ซ็อกเก็ต(Socket)

2.5.1 เอพีไอ(API) ข้อกำหนดในการติดต่อสื่อสาร

TCP/IP เป็น software ในการเชื่อมโยงเครือข่ายชนิดหนึ่งที่สนับสนุนโปรแกรมประยุกต์ในการอินเทอร์เน็ตสำหรับการติดต่อสื่อสารบนอินเทอร์เน็ต ซึ่งจะอธิบายชนิดของ APIs ในที่นี้เพียง 2 ชนิด

Socket System Calls

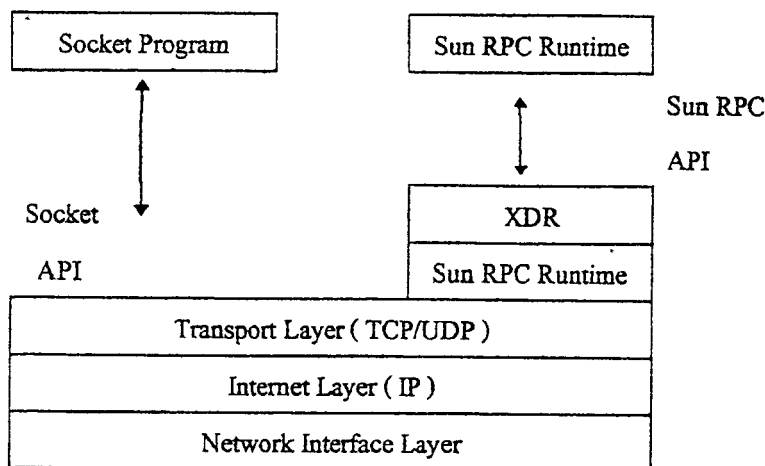
ซ็อกเก็ต เข้าถึง ข้อกำหนดในการติดต่อสื่อสารของโครงข่าย ซึ่งโดยทั่วไปจะเกี่ยวข้องกับการเปลี่ยนแปลง BSD ของ UNIX ซึ่งเป็นอุปกรณ์อันดับแรกของ ซ็อกเก็ต API การอินเทอร์เน็ตของซ็อกเก็ต ได้จัด API สำหรับการติดต่อสื่อสารของโครงข่าย ซึ่งเป็น API ที่ใกล้เคียงกับการทำงานของ I/O มากๆ

การอินเทอร์เน็ตของซ็อกเก็ตเป็นเครื่องมือที่ใช้งานอย่างกว้างขวางสำหรับระบบปฏิบัติการ UNIX และระบบปฏิบัติการอื่นๆ โปรแกรมที่เข้าถึงซ็อกเก็ตสามารถที่จะใช้งานเป็นเครื่องมือสำหรับการติดต่อสื่อสารบนโครงข่ายอินเทอร์เน็ต โดยอยู่ในสภาพแวดล้อม ที่แตกต่างกันได้และติดต่อสื่อสารได้กับโฮสต์ทุกชนิด ซ็อกเก็ต API โดยทั่วไปมักใช้สำหรับโครงข่ายที่ใช้งานบน TCP/IP และก็สามารถใช้งานได้ดีกับโครงข่ายแบบอื่นๆ

Remote Procedure Call Facilities

Remote Procedure Call (RPC) เป็นชนิดหนึ่งของ API ซึ่งยอมให้ผู้เขียนโปรแกรมสำหรับโครงข่าย ใช้ procedure call ที่มีโครงสร้างอย่างง่ายๆ เพื่อที่จะควบคุมกันระหว่างโพรซีเยอร์ของโฮสต์ที่มีการทำงานแตกต่างกันบนอินเทอร์เน็ต สิ่งอำนวยความสะดวกของ RPC ที่ใช้กันอย่างกว้างขวางบนสถานะ TCP/IP ได้แก่

- SUN RPC
- Hewlett - Packard RPC



รูป 2.3 แสดงการเปรียบเทียบระหว่าง ซ็อกเก็ต และ SUN RPC application programming

ซ็อกเก็ตอินเตอร์เฟซเป็นการเริ่มพัฒนาสำหรับหลายๆ BSD ของระบบปฏิบัติการ UNIX โดยการใช้ซ็อกเก็ตอินเตอร์เฟซ จะมี 2 โปรแกรมประยุกต์คือ

- 1.การทำงานที่ Local System
- 2.การทำงานที่ Remote System

ซึ่งสามารถติดต่อกับโฮสต์อื่นๆได้ด้วยวิธีมาตรฐาน ซ็อกเก็ตอินเตอร์เฟซที่ใช้กันอย่างกว้างขวางคือ API ที่ใช้งานบน TCP/IP

ซ็อกเก็ตAPI เป็นอุปกรณ์ที่ใช้สำหรับการเชื่อมต่อระหว่างผู้ขอใช้บริการและผู้ให้บริการระหว่าง 2 ส่วนประกอบ ประยุกต์ที่ทำงานบนระบบของคอมพิวเตอร์ที่ต่างกัน

การติดต่อระหว่าง ผู้ขอใช้บริการและผู้ให้บริการ แต่ละครั้ง จะเรียกฟังก์ชัน ขึ้นมาจัดการการเชื่อมต่อระหว่าง ผู้ขอใช้บริการและผู้ให้บริการ นั่นๆ การเชื่อมต่อระหว่าง 2 ส่วนประกอบ(component) ประยุกต์นี้เป็นพื้นฐานบนการใช้โครงสร้างข้อมูลที่ถูกเรียกว่า ซ็อกเก็ต เพื่อเป็นการบริการการติดต่อสื่อสารโดยใช้ TCP/IP

ส่วนประกอบของผู้ขอใช้บริการและผู้ให้บริการ (Client-Server component) จะเรียก ฟังก์ชันที่ใช้ส่งและรับข้อมูลบน อินเทอร์เน็ต ซึ่งคล้ายกับวิธีการการเรียกฟังก์ชันที่ใช้กับ ไฟล์ I/O ทั่วไป

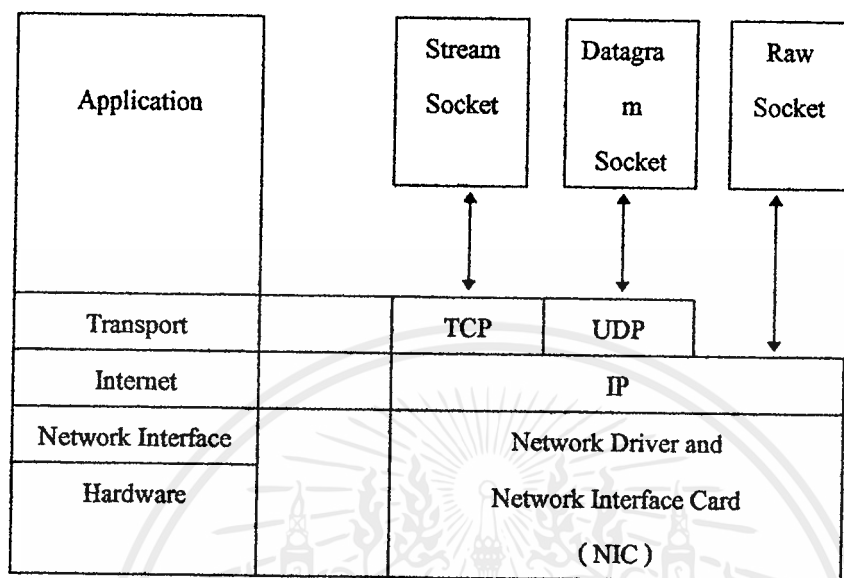
ซ็อกเก็ตAPI สามารถจะใช้งานได้ไม่ว่าโปรโตคอลอื่นๆที่นอกเหนือจาก TCP/IP แต่ในที่นี้จะอธิบายเฉพาะโปรโตคอล TCP/IP เท่านั้น

2.5.2 ชนิดของซ็อกเก็ต

ซ็อกเก็ตที่ใช้ในการติดต่อสื่อสารบน TCP/IP จะมีอยู่สามแบบด้วยกันดังนี้

1. Stream Socket คือ ซ็อกเก็ตที่ใช้โปรโตคอล TCP ชั้น ทรานสปอร์ต ในการติดต่อ ซ็อกเก็ตชนิดนี้จะใช้การติดต่อแบบ Connection Oriented ซึ่งการส่งรับข้อมูลแบบนี้สามารถเชื่อถือได้
 2. Datagram Socket คือ ซ็อกเก็ตที่ใช้โปรโตคอล UDP ชั้น ทรานสปอร์ต ในการติดต่อ Datagram Socket จะเชื่อถือไม่ค่อยได้ รูปแบบของข้อมูลที่ส่งจะถูกแบ่งออกเป็นส่วนๆ โดยแต่ละอันจะถูกส่งไปปลายทางโดยอิสระจากกัน
- เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. Raw Socket คือ ซ็อกเก็ตที่อยู่ภายใต้ชั้น IP และ ICMP ซ็อกเก็ตชนิดนี้จะใช้เฉพาะการส่งที่มีจุดมุ่งหมายพิเศษ



TCP/IP Architectural Layers

TCP/IP Components and Protocols

รูป 2.4 แสดงลักษณะทั้งสามแบบของซ็อกเก็ต

2.5.3 Socket Address

ระบบของ ซ็อกเก็ต โดยทั่วไปจะอ้างอิงโครงสร้างข้อมูลแบบพอยเตอร์ (pointer) ที่ประกอบไปด้วยข้อมูลต่างๆ ซึ่งก็คือซ็อกเก็ตแอดเดรสนั่นเอง

ซ็อกเก็ตแอดเดรสจะใช้เฉพาะการสื่อสารแบบ one-end

สำหรับ โพรโตคอล TCP/IP จะมีโครงสร้างของ ซ็อกเก็ตแอดเดรสดังนี้

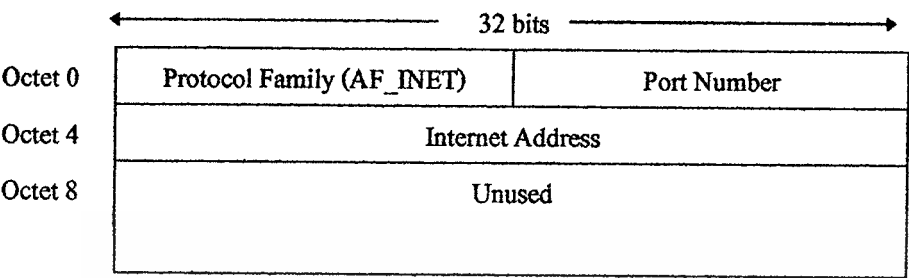
1. Family เป็นเลขจำนวนเต็ม 16 บิต ที่ชี้เฉพาะ โพรโตคอล เช่น โพรโตคอล TCP/IP คือ AF_INET , UNIX คือ AF_UNIX

2. พอร์ต เป็นเลขจำนวนเต็ม 16 บิต ที่ระบุหมายเลขพอร์ต สำหรับใช้ในระบบการต่างๆ สำหรับ ผู้ให้บริการ หมายเลขพอร์ตที่ใช้จะถูกกำหนดโดยตัวผู้ให้บริการเอง ซึ่งผู้ขอใช้บริการ ที่ต้องการติดต่อกับ ผู้ให้บริการ โดยใช้โปรแกรมนั้นจะต้องรู้หมายเลขพอร์ตที่ใช้ติดต่อจึงจะสามารถติดต่อกันได้

3. Address เป็นเลขจำนวนเต็ม 32 บิต ซึ่งภายในคือ ไอพีแอดเดรสที่อยู่ในรูปแบบของเลขฐานสอง ในการทำงานเพื่อติดต่อกัน ผู้ขอใช้บริการ ต้องรู้ไอพีแอดเดรส ของผู้ให้บริการ จึงจะสามารถทำการติดต่อกันได้ แต่ละการติดต่อของ ผู้ขอใช้บริการ -ผู้ให้บริการ ต้องมีการกำหนด ซ็อกเก็ตแอดเดรส ซึ่งซ็อกเก็ตแอดเดรสก็หมายถึงรวมถึงซ็อกเก็ตแอดเดรสของ ผู้ให้บริการ และ ซ็อกเก็ตแอดเดรสของ ผู้ขอใช้บริการ กระบวนการติดต่อของ ผู้ขอใช้บริการและผู้ให้บริการ ต้องมีการชี้เฉพาะไปที่ โพรโตคอลชั้น ทรานสปอร์ต เพื่อใช้ในการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ติดต่อสื่อสารรูป 2.5 จะแสดงความหมายของชนิดข้อมูลและ โครงสร้างของข้อมูลที่จะใช้ในการกำหนด
ซ็อกเก็ตแอดเดรส



รูป 2.5 แสดงโครงสร้างของข้อมูลที่ใช้ในการกำหนดซ็อกเก็ตแอดเดรส

2.5.4 Socket System Call

ซ็อกเก็ตAPI ให้ Socket System Call เป็น โปรแกรมประยุกต์ที่ทำหน้าที่เรียกบริการการร้องขอที่จะทำ
การติดต่อสื่อสาร Socket System Call พื้นฐานที่ใช้กันทั่วไปอธิบายได้ดังนี้

Socket System Calls

- 1. **accept.** ถูกใช้ใน connection-oriented application protocol ที่ใช้ TCP โดยใช้ในขบวนการผู้ให้บริการ ซึ่ง ผู้ให้บริการ จะรอการร้องขอเพื่อติดต่อจาก ผู้ขอใช้บริการ
- 2. **bind.** ถูกใช้ร่วมกับ Endpoint identifier (internet address และ หมายเลขพอร์ต) ซึ่งเลขจำนวนเต็ม เป็นตัวอธิบายในการเปิดซ็อกเก็ต
- 3. **close.** ใช้ในการปิดซ็อกเก็ต โดยในรายละเอียดจะคล้ายกับวิธีปิดไฟล์
- 4. **connect.** ถูกใช้ใน connection-oriented application protocol ที่ใช้ TCP โดยขบวนการ ผู้ขอใช้ บริการ จะสร้างการติดต่อแบบ TCP กับ ผู้ให้บริการ
- 5. **gethostbyaddr.** จะใช้เมื่อต้องการข้อมูลเกี่ยวกับ โฮส ซึ่งรู้อินเทอร์เน็ตแอดเดรสแล้ว
- 6. **gethostbyname.** จะใช้เพื่อให้ได้ อินเทอร์เน็ตแอดเดรส ของ remote host ซึ่งเรารู้ host name แล้ว
- 7. **gethostid.** ถูกใช้เพื่อให้ได้ อินเทอร์เน็ตแอดเดรส ของ local host
- 8. **gethostname.** ถูกใช้เพื่อให้ได้ชื่อของ local host
- 9. **getpeername.** ถูกใช้เพื่อให้ได้ Endpoint address ซึ่งเกี่ยวข้องกับขบวนการด้านรีโมท
- 10. **getsockname.** ถูกใช้เพื่อให้ได้ชื่อที่ซ็อกเก็ตให้มา
- 11. **getprotobyname.** ถูกใช้เพื่อให้ได้ค่าเลขจำนวนเต็มซึ่งเป็นค่าของ โปรโตคอล ที่จะบอกชื่อของ โปรโตคอล
- 12. **getservbyname.** ถูกใช้เพื่อให้ได้ค่าเลขจำนวนเต็มซึ่งเป็นค่าที่ให้ชื่อของการบริการของโปรแกรม ประยุกต์บนชั้น แอปพลิเคชัน ของ โมเดล TCP/IP ที่รู้จักกันดี

13. **getsockopt.** ถูกใช้เพื่อให้ได้รายชื่อพารามิเตอร์ของซ็อกเก็ต
14. **listen.** ใช้ในการติดต่อแบบ connection-oriented application protocol ที่ใช้ TCP โดยเป็นขบวนการของ ผู้ให้บริการ ที่จะแสดงให้เห็นว่าผู้ให้บริการจะยอมรับการติดต่อจาก ผู้ขอใช้บริการ เมื่อมีการร้องขอเพื่อที่จะสร้างการติดต่อแบบ TCP
15. **read.** ถูกใช้เพื่อที่จะรับข้อมูลมาจากซ็อกเก็ต
16. **readv.** เป็นอีกตัวเลือกหนึ่งของคำสั่ง read เพื่อที่จะรับข้อมูลใส่ไว้ในบัพเฟอร์ที่ไม่ติดกัน
17. **recv.** เป็นอีกตัวเลือกหนึ่งของ read เพื่อที่จะรับข้อมูลที่กำลังเข้ามาจากซ็อกเก็ตและใช้แฟล็กมา เป็นข้อกำหนดเพิ่มเข้าไปด้วย
18. **recvfrom.** ใช้กับ connectionless application protocol ที่ใช้แบบ UDP เพื่อที่จะรับ user datagram จาก datagram socket เพื่อที่จะให้ได้ที่อยู่ซ็อกเก็ตของผู้ส่ง
19. **recvmsg.** ใช้กับ connectionless application protocol ที่ใช้แบบ UDP เพื่อที่จะรับ user datagram จาก datagram socket โดยในบัพเฟอร์จะประกอบด้วยส่วนหัวของ user datagram ซึ่งก็คือ ส่วนหนึ่งของข้อมูลของ user datagram
20. **select.** ถูกใช้เพื่อให้ขบวนการรอคอย(wait) สมบูรณ์
21. **send.** ใช้เป็นอีกทางเลือกของ write เพื่อที่จะส่งข้อมูลบนซ็อกเก็ตและใช้เงื่อนไขของแฟล็กเพิ่มมา ด้วย
22. **sendmsg.** ใช้กับ connectionless application protocol ที่ใช้แบบ UDP เพื่อที่จะส่ง user datagram ไปบน datagram socket โดยในบัพเฟอร์จะประกอบด้วยส่วนหัวของ user datagram ซึ่งก็คือ ส่วนหนึ่งของข้อมูลของ user datagram
23. **sendto.** ใช้กับ connectionless application protocol ที่ใช้แบบ UDP เพื่อที่จะส่งข้อมูลบน datagram socket และระบุ socket address ของ remote process, sendto จะถูกเรียกมาเพื่อให้สัมพันธ์กับการสร้างการติดต่อในเวลาที่จะส่งข้อมูล
24. **setsockopt.** ใช้เพื่อกำหนดค่าสำหรับพารามิเตอร์ต่างๆของซ็อกเก็ต
25. **shutdown.** ใช้กับ connection-oriented application protocol ที่ใช้แบบ TCP โดย shutdown จะทำการยกเลิกการติดต่อที่ TCP สร้างขึ้น
26. **socket.** ใช้เพื่อเริ่มต้นโครงสร้างทางข้อมูลของซ็อกเก็ต ทำการแยกแยะ โปรโตคอลที่ใช้ งาน สำหรับการติดต่อสื่อสารในชั้นทรานสปอร์ต และได้จำนวนเต็มที่สามารถบอกรายละเอียดได้ โดยหลังจากที่ระบบการเรียกของซ็อกเก็ตสามารถใช้ตัวเลขนี้อ้างอิงถึงซ็อกเก็ตได้
27. **write.** ใช้เพื่อส่งข้อมูลบนซ็อกเก็ต
28. **writv.** ใช้เพื่อเป็นอีกทางเลือกหนึ่งของ write เพื่อที่จะส่งข้อมูลจากบัพเฟอร์ที่ไม่ติดกัน



2.5.5 Application Protocol

สามารถมีซ็อกเก็ต API ได้ทั้ง 2 แบบ คือทั้ง Datagram Socket และ Stream Socket ที่จะใช้ในการออกแบบโปรแกรม โดยมีโปรโตคอลการติดต่อแบ่งเป็น 2 ประเภทดังนี้

Connectionless Application Protocol

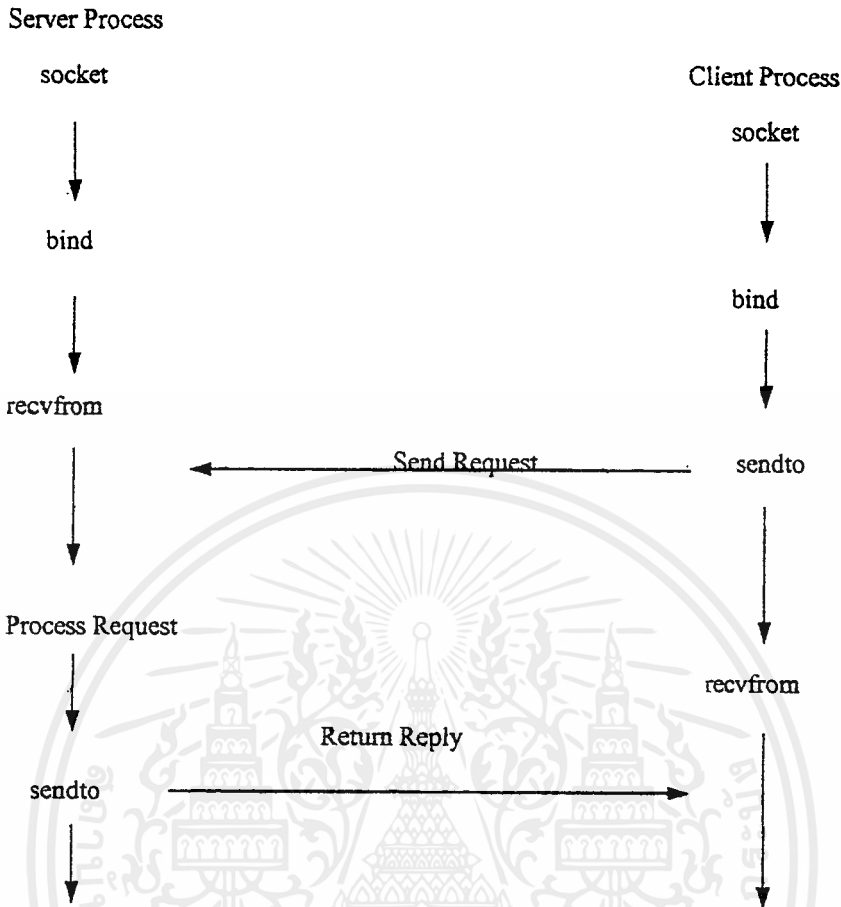
เป็น โปรแกรมประยุกต์ ผู้ขอใช้บริการและผู้ให้บริการ ที่ใช้ Connectionless Application Protocol จะใช้กับ Datagram Socket เพื่อจะติดต่อสื่อสารบนการติดต่อ UDP ดังแสดงในรูป 2.6 แสดงให้เห็นถึงลำดับของ Socket System Call ที่ใช้แบบ Connectionless Application Protocol การทำแบบนี้จะ ไม่มีการตรวจสอบความผิดพลาดและการส่งซ้ำ (Retransmission) เนื่องจากเป็นการบริการแบบ UDP

พิจารณากระบวนการของ ผู้ให้บริการ จะใช้ Socket , Bind System Calls และรับข้อมูลโดยใช้ Recvfrom ซึ่ง System Calls ทั้ง 5 กระบวนการนี้ต้องการ การเชื่อมต่อที่สมบูรณ์ระหว่าง กระบวนการ ผู้ขอใช้บริการ และ กระบวนการผู้ให้บริการ สำหรับ Connectionless Application Protocol นี้การเชื่อมต่อจะยังไม่สมบูรณ์จนกระทั่งถึงเวลาที่ กระบวนการผู้ขอใช้บริการ ได้ส่งข้อมูลไปยังกระบวนการผู้ให้บริการแล้วจริงๆ

พิจารณากระบวนการของผู้ขอใช้บริการจะใช้ Socket , Bind , Sendto System Calls เพื่อให้การเชื่อมต่อระหว่างกระบวนการผู้ขอใช้บริการและ กระบวนการผู้ให้บริการนั้นสมบูรณ์ ซึ่งการเชื่อมต่อจะสมบูรณ์เมื่อเวลาที่ผู้ขอใช้บริการ ส่งค่าคำแถมอันดับแรกไปถึงผู้ให้บริการแล้ว

Server System Calls

1. ผู้ให้บริการจะสั่งให้ socket system call เข้าถึงข้อมูลของ โปรโตคอลใน 5-tuple ที่สัมพันธ์กัน ซึ่งในที่สุดแล้วก็จะถูกสร้างเพื่อสนับสนุนการติดต่อสื่อสารระหว่างขบวนการผู้ขอใช้บริการและขบวนการผู้ให้บริการ ใน connectionless application จะเห็นว่า socket system call เจาะจงที่จะใช้ UDP
2. ผู้ให้บริการจะสั่งให้ bind system call (ระบุที่อยู่ซ็อกเก็ตของผู้ให้บริการ) ทำแทนในส่วนที่เหลือของข้อมูลข่าวสารที่กำลังเชื่อมเซม server's half- association , โดย half- association ของผู้ให้บริการบรรจุไปด้วย โปรโตคอลที่มีลักษณะเฉพาะตัว ในตอนนี้ อินเทอร์เน็ตแอดเดรส และหมายเลขพอร์ตของผู้ให้บริการกำลังทำงานอยู่
3. ผู้ให้บริการจะสั่งให้ทำ recvfrom system call เพื่อให้ผู้ให้บริการรับข้อมูลจาก ผู้ขอใช้บริการ ที่กำลังเข้ามา, recvfrom system call เป็นสาเหตุให้ผู้ให้บริการ ขบวนการรอคอย จนกระทั่งข้อมูลจาก ผู้ขอใช้บริการ มาถึง ซึ่งมันก็คือความต้องการที่จะขอใช้บริการจากทางผู้ให้บริการ
4. เมื่อ user datagram จาก ผู้ขอใช้บริการ มาถึง, ผู้ให้บริการจะทำขบวนการ request process ซึ่งจะเห็นว่าส่วนหัวข้อมูลของ user datagram ที่ส่งมาจะประกอบไปด้วย อินเทอร์เน็ตแอดเดรส และหมายเลขพอร์ตของ ผู้ขอใช้บริการ ข่าวดังกล่าวนี้จะจัดหาหนทางไปผู้ให้บริการจนกระทั่ง 2 ชิ้นสุดท้ายของข้อมูลถูกร่องขอเพื่อให้ 5- tuple สมบูรณ์ ซึ่งจะสัมพันธ์ระหว่างด้าน ผู้ขอใช้บริการ และผู้ให้บริการ
5. ผู้ให้บริการตอบกลับไปยัง ผู้ขอใช้บริการ 1 ครั้งหรือมากกว่า โดยใช้คำสั่ง sendto system call ซึ่งจะทำการส่ง user datagram กลับไปหา ผู้ขอใช้บริการ , sendto system call จะใช้ socket address ของ ผู้ขอใช้บริการเป็นตัวอ้างอิงเพื่อที่จะตกลงตอบรับการติดต่อ



รูป 2.6 วิธีการแบบ connectionless application protocol โดยใช้ UDP

Client system calls

1. ผู้ขอใช้บริการ จะสั่งให้ socket system call เข้าถึงข้อมูลของ โปรโตคอลสำหรับ half - association หนึ่ง ใน connectionless application protocol จะเห็นว่า socket system call จะใช้ UDP
2. ผู้ขอใช้บริการ จะสั่งให้ bind system call ระบุที่อยู่ซ็อกเก็ตของผู้ขอใช้บริการแล้วซึ่งเต็มไปด้วยส่วนที่เหลือของข้อมูลข่าวสารที่มีลักษณะ half - association ของ ผู้ขอใช้บริการ , ขบวนการผู้ขอใช้บริการเป็นชนิดที่ร้องขอ networking software ที่จะเป็นแหล่งของอินเทอร์เน็ตแอดเดรสของผู้ขอใช้บริการและจัดหาหมายเลขพอร์ตที่ไม่ได้ใช้เอามาใช้ชั่วคราว
3. ผู้ขอใช้บริการจะสั่งให้ sendto system call ทำการส่ง user datagram ไปยังผู้ให้บริการ ทางด้านผู้ขอใช้บริการ จำเป็นต้องรู้ที่อยู่ซ็อกเก็ตของผู้ให้บริการและที่อยู่ซ็อกเก็ตนี้จะต้องถูกอ้างอิงใน sendto system call
4. ถ้าผู้ขอใช้บริการคาดหวังว่าจะได้รับข้อมูลจากผู้ให้บริการ มันจะต้องสั่งให้เรียก recvfrom ซึ่งเป็นสาเหตุให้ผู้ขอใช้บริการคอยจนกระทั่งข้อมูลจากผู้ให้บริการมาถึง
5. หลังจากนั้นขบวนการติดต่อกันระหว่างผู้ขอใช้บริการและผู้ให้บริการจะสิ้นสุดลง เป็นอันสิ้นสุดขบวนการ

Connection-Oriented Application Protocol

เป็นโปรแกรมประยุกต์ผู้ขอใช้บริการและผู้ให้บริการที่ใช้ Connection-Oriented Application Protocol จะใช้กับ Stream Socket เพื่อจะติดต่อสื่อสารบนการติดต่อบน TCP ดังรูป 2.7

การส่งข้อมูลโดยใช้ Connection-Oriented อันดับแรกผู้ให้บริการจะเตรียมตัวให้พร้อมด้วยตัวของมันเอง และรอการร้องขอการติดต่อจากผู้ขอใช้บริการ ส่วนผู้ขอใช้บริการจะร้องขอการติดต่อและส่งข้อมูลไปยังผู้ให้บริการภายใต้การทำงานของ TCP จะมีการตรวจสอบความผิดพลาด มีการส่งข้อมูลซ้ำใหม่และมีการควบคุมการไหลของข้อมูล

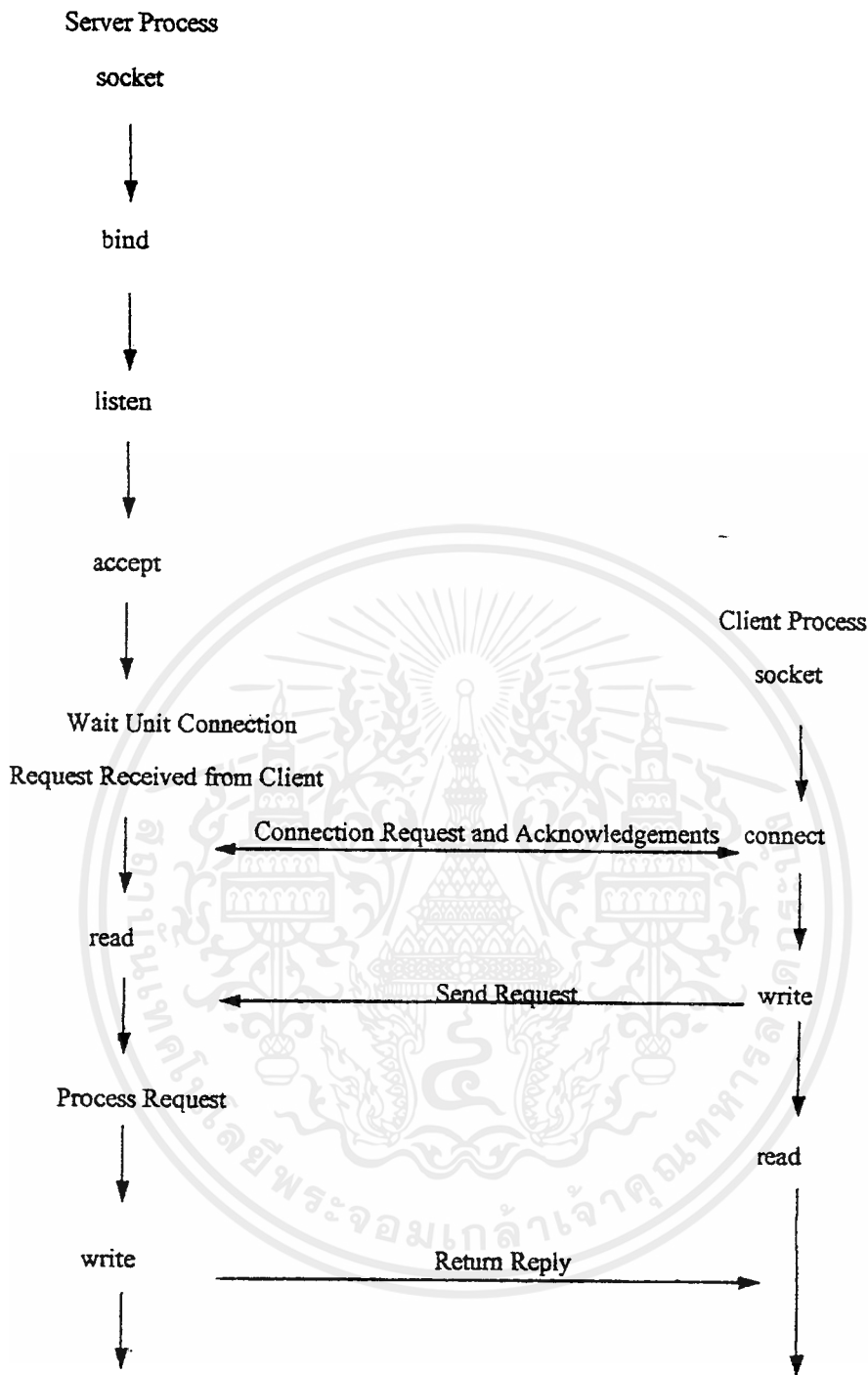
พิจารณากระบวนการของผู้ให้บริการจะใช้ Socket , Bind System Calls ต่อมาหากมีการติดต่อจะใช้ Accept System Call เพื่อยอมรับการติดต่อนั้น วิธี Connection-Oriented Application Protocol จะสมบูรณ์เมื่อสร้างการเชื่อมต่อสำเร็จ

พิจารณากระบวนการของผู้ขอใช้บริการจะใช้ Socket , Connect System Calls จะเห็นว่ามี Three way Handshake ซึ่งจะเริ่มเมื่อผู้ขอใช้บริการใช้ Connect System Call ตามด้วยการแลกเปลี่ยนข้อมูลระหว่างผู้ขอใช้บริการและผู้ให้บริการ ดังนั้นการติดต่อนี้จะสมบูรณ์เมื่อเวลาที่ผู้ขอใช้บริการใช้ Connect System Call

Server System Calls

1. ผู้ให้บริการจะสั่งให้ socket system call เข้าถึงข้อมูลของโปรโตคอลใน 5-tuple ที่สัมพันธ์กัน ซึ่งในที่สุดแล้วก็จะถูกสร้างเพื่อสนับสนุนการติดต่อสื่อสารระหว่างขบวนการผู้ขอใช้บริการและขบวนการผู้ให้บริการใน connection-oriented application จะเห็นว่า socket system call เจาะจงที่จะใช้ TCP
2. ผู้ให้บริการจะสั่งให้ bind system call (ระบบที่อยู่ซ็อกเก็ตของผู้ให้บริการ) ทำแทนในส่วนที่เหลือของข้อมูลข่าวสารที่กำลังเชื่อม server's half- association , โดย half- association ของผู้ให้บริการบรรจุไปด้วยโปรโตคอลที่มีลักษณะเฉพาะตัว ในตอนนี้อินเทอร์เน็ตแอดเดรสและหมายเลขพอร์ตของผู้ให้บริการกำลังทำงานอยู่
3. ผู้ให้บริการจะสั่งให้ทำ listen system call ซึ่งแสดงว่าผู้ให้บริการกำลังรอการร้องขอจากผู้ขอใช้บริการเพื่อสร้างการติดต่อ TCP
4. ผู้ให้บริการจะสั่งให้ทำ accept system call เมื่อ มีการร้องขอการดำเนินการเชื่อมต่อ TCP ที่ได้รับจาก client accept system call จะอ้างอิง ซ็อกเก็ตแอดเดรส ที่เกิดจากซ็อกเก็ตแอดเดรสของกระบวนการผู้ขอใช้บริการ เมื่อมีการร้องขอสำหรับการเชื่อมต่อที่ได้รับจริงๆ
5. ผู้ให้บริการจะสั่งให้ทำ read system call หรือ recv system call เพื่ออนุญาตให้ผู้ให้บริการรับข้อมูลจากผู้ขอใช้บริการบนการเชื่อมต่อ TCP
6. ผู้ให้บริการสามารถสั่งให้ทำ write system call หรือ send system call เพื่อที่จะอนุญาตให้ผู้ให้บริการส่งข้อมูลไปยังผู้ขอใช้บริการได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 2.7 วิธีการแบบ connection-oriented application protocol โดยใช้ TCP

Client system calls

1. ผู้ขอใช้บริการจะสั่งให้ socket system call เข้าถึงข้อมูลของโปรโตคอลสำหรับ half-association หนึ่งใน connection-oriented application protocol จะเห็นว่า socket system call จะใช้ TCP
2. ผู้ขอใช้บริการจะสั่งให้ connect system call ที่อ้างอิงชื่อเกิดแอดเดรสของผู้ขอใช้บริการที่จะสร้างการเชื่อมต่อ TCP กับ ผู้ให้บริการ , connect system call เกิดได้จากการมีข้อมูลโฮสต์กลับ และอันดับ 4 ระหว่างกระบวนการผู้ขอใช้บริการและกระบวนการผู้ให้บริการเพื่อที่จะสร้างกระบวนการ three way handshake ขึ้นอย่างสมบูรณ์เพื่อที่จะต้องการสร้างการติดต่อ ข้อมูลที่แลกเปลี่ยนระหว่างกระบวนการ handshake รวมเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทั้งข้อมูลที่ต้องการเพื่อที่จะสร้างการเชื่อมต่อที่สมบูรณ์ระหว่างผู้ขอใช้บริการและผู้ให้บริการ ขบวนการผู้ขอใช้บริการต้องรู้ชื่อเกิดแอดเดรสของกระบวนการผู้ให้บริการ ในการออกคำสั่ง connect system call และเป็นชนิดที่ร้องขอ networking software ที่จะเป็นแหล่งของ อินเทอร์เน็ตแอดเดรส ของผู้ขอใช้บริการ และจัดหาหมายเลขพอร์ตที่ไม่ได้ใช้เอามาใช้ชั่วคราว

3. ผู้ขอใช้บริการจะสั่งให้ write system call หรือ send system call เพื่อที่จะส่งข้อมูลไปยังผู้ให้บริการ บนการเชื่อมต่อ TCP ที่ถูกสร้างขึ้น
4. ผู้ขอใช้บริการสามารถสั่งให้ read system call หรือ recv system call เพื่อที่จะรับข้อมูลที่เข้ามาจากผู้ให้บริการ
5. หลังจากนั้นขบวนการติดต่อกันระหว่างผู้ขอใช้บริการและผู้ให้บริการจะสิ้นสุดลง เป็นอันสิ้นสุดขบวนการ

2.5.6 กระบวนการ System Call พื้นฐาน

System Call ในหัวข้อนี้ใช้บน TCP/IP และจะให้ค่าผลลัพธ์เป็น integer ซึ่งค่านี้จะบอกได้ถึงความผิดพลาดที่เกิดขึ้นในการทำงานของชื่อเกิดนั้น

Socket System Call

ใช้สำหรับสร้างชื่อเกิดและสร้างการติดต่อระหว่าง 2 กระบวนการติดต่อ โดยมีการกำหนด รูปแบบดังนี้

Socket (family, type, protocol)

- family : แสดง Family ของโปรโตคอลที่ใช้ในการติดต่อ เช่น โปรโตคอล TCP/IP คือ

AF_INET

- type : แสดงชนิดของชื่อเกิดที่ใช้ เช่น Stream Socket คือ SOCK_STREAM

- protocol : ซึ่งเฉพาะ โปรโตคอลที่ใช้ในการติดต่อ ทั่วไปแล้วจะกำหนดให้เท่ากับศูนย์

ตัวอย่าง

การสร้าง Stream Socket สำหรับ Connection - Oriented Application Protocol โดยใช้ TCP

```
stream_sockfd = socket ( AF_INET, SOCK_STREAM, 0 );
```

การสร้าง Datagram Socket สำหรับ Connectionless Application Protocol โดยใช้ UDP

```
dgram_sockfd = socket ( AF_INET, SOCK_DGRAM, 0 );
```

Bind System Call

จะสร้างแอดเดรสและพอร์ตสำหรับการติดต่อ โดยที่ ผู้ให้บริการ จะบันทึก อินเทอร์เน็ตแอดเดรสและหมายเลขพอร์ตด้วยซอฟต์แวร์ของการติดต่อสื่อสาร TCP/IP ส่วนกระบวนการ ของผู้ขอใช้บริการ ที่มีการติดต่อแบบ Connectionless Application Protocol ทั่วไปจะใช้ Bind เพื่อบันทึกอินเทอร์เน็ตแอดเดรสและสร้างหมายเลขพอร์ตชั่วคราวที่ ผู้ให้บริการ สามารถใช้ส่งข้อมูลกลับมาที่กระบวนการ ผู้ขอใช้บริการ

ผู้ขอใช้บริการ สามารถร้องขอซอฟต์แวร์ของการติดต่อ TCP/IP เพื่อจะให้อินเทอร์เน็ตแอดเดรสของผู้ขอใช้บริการเองและสร้างหมายเลขพอร์ตชั่วคราวที่ยังไม่ได้มีการใช้ระหว่างการติดต่อ แต่ในการติดต่อแบบ

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Connection-Oriented Application Protocol กระบวนการ ผู้ขอใช้บริการ ไม่จำเป็นต้องใช้ Bind ซึ่งถ้า ผู้ขอใช้บริการ ไม่เรียกใช้ Bind

Connect System Call จะถูกเรียกใช้โดยอัตโนมัติเพื่อให้กระบวนการ Binding สมบูรณ์ รูปแบบแสดงได้ดังนี้

```
bind ( socket , local_address , address_lenght )
```

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- local address : แสดง local แอดเดรสที่ bind สร้างขึ้น
- address length : เป็นเลขจำนวนเต็มแสดงความยาวของแอดเดรสในหน่วยไบต์

ตัวอย่าง

```
bind ( sockfd,( struct sockaddr * ) &own_addr,sizeof ( own_addr ) );
```

Connect System Call

หลังจากมีการสร้างซ็อกเก็ตแอดเดรสแล้ว ซ็อกเก็ตเป้าหมายจะทำการติดต่อโดยใช้ Connect System Call ซึ่งทั่วๆไปแล้ว จะออกโดย กระบวนการ ผู้ขอใช้บริการ ในการติดต่อแบบ Connection-Oriented Application Protocol ซึ่งจะมี กระบวนการ Three way Handshake รวมทั้งแลกเปลี่ยนอัตโนมัติทั้งข้อมูลที่เป็นและเชื่อมต่ออย่างสมบูรณ์ระหว่างกระบวนการทั้งสอง แสดงรูปแบบดังนี้

```
connect ( socket , destination_address , address_length )
```

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- destination address : แสดงแอดเดรสปลายทางที่ต้องการติดต่อ
- address length : เป็นเลขจำนวนเต็มแสดงความยาวของแอดเดรสปลายทางในหน่วยไบต์

ตัวอย่าง

```
connect ( sockfd,( struct sockaddr * ) &serv_addr,sizeof ( serv_addr ));
```

Listen System Call

Listen System Call ออกโดย กระบวนการ ผู้ให้บริการในการติดต่อแบบ Connection-Oriented Application Protocol โดยใช้ โปรโตคอล TCP โดยจะใช้ Listen System Call เพื่อบอกแก่ซอฟต์แวร์ในการติดต่อแบบ TCP/IP ซึ่งกระบวนการผู้ให้บริการจะยอมรับ (Accept) การร้องขอจากกระบวนการผู้ขอใช้บริการ สำหรับการสร้างการติดต่อบน TCP รูปแบบแสดงดังนี้

```
listen ( socket , queue_length )
```

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- queue length : เป็นตัวเลขที่ชี้ให้เห็นว่า มีการร้องขอการติดต่อเพื่อรอคอยกระบวนการตอบ

รับจำนวนเท่าใด ตัวแปรนี้มีค่ามากที่สุดเท่ากับ 5

ตัวอย่าง

```
listen ( sockfd,5 );
```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Accept System Call

ส่วนใหญ่แล้ว Accept System Call จะถูกออกโดยกระบวนการทางผู้ให้บริการ ใน Connection-Oriented Application Protocol เป็นผลให้ กระบวนการทางผู้ให้บริการต้องรอการร้องขอที่จะเข้ามาจากผู้ขอใช้บริการ รูปแบบแสดงดังนี้

```
accept ( socket , client_address , address_length )
```

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- client address : แอดเดรสของ ผู้ขอใช้บริการ ที่ต้องการทำการติดต่อ
- address length : เป็นเลขจำนวนเต็มแสดงความยาวของแอดเดรสของ ผู้ขอใช้บริการใน

หน่วยไบต์

ผลที่ได้จะสร้าง โครงสร้างข้อมูลซ็อกเก็ตใหม่ และคืนค่าตัวแปรของซ็อกเก็ตที่สามารถใช้หลังจากอ้างอิงโครงสร้างข้อมูลของซ็อกเก็ตใหม่

ตัวอย่าง

```
newsockfd = accept ( sockfd, ( struct sockaddr *) &cli_addr,&clilen );
```

Write and Send System Call

Write and Send System Call ใช้สำหรับส่งข้อมูลไปยังอีกฝ่ายหนึ่ง Write และ Send คล้ายกันแต่ Send System Call จะประกอบด้วยตัวแปรที่มากกว่า มีรูปแบบดังนี้

```
write ( socket , bubuffer , length )
```

```
send ( socket , buffer_address , length , flags )
```

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- buffer address : เป็นบัพเฟอร์ที่ใช้เก็บข้อมูลในการส่ง
- length : จำนวน ไบต์ที่กำหนดเพื่อใช้ในการส่ง
- flags : เป็นตัวที่ใช้กำหนดเจาะจงทางเลือกพิเศษสำหรับการส่ง

ผลที่ได้จาก System Call นี้คือ จำนวนไบต์ที่ส่งได้

ตัวอย่าง

```
nwritten = write ( sockfd,buffer,noctets );
```

```
nwritten = send ( sockfd,buffer,noctets,sendflag );
```

Sendto System Call

Sendto System Call คล้ายกับ Send และ Write System Call แต่จะ ใช้กับการติดต่อแบบ

Connectionless Application Protocol บน UDP Sendto System Call จะสร้างการเชื่อมต่อระหว่าง 2 กระบวนการ ซึ่งการติดต่อนี้จะสมบูรณ์เมื่อข้อมูลได้ถูกส่งออกไปแล้ว มีรูปแบบดังนี้

```
sendto ( socket , buffer_address , length , flags , destination , address_length )
```

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- buffer address : เป็นบัฟเฟอร์ที่ใช้เก็บข้อมูลในการส่ง
- length : จำนวน ไบต์ที่กำหนดเพื่อใช้ในการส่ง
- flags : เป็นตัวที่ใช้กำหนดเจาะจงทางเลือกพิเศษสำหรับการส่ง
- destination address : แสดงแอดเดรสปลายทางที่ต้องการติดต่อ
- address length : เป็นเลขจำนวนเต็มแสดงความยาวของแอดเดรสปลายทางในหน่วยไบต์

ผลที่ได้จาก System Call นี้คือ จำนวนไบต์ที่ส่งได้จริง

ตัวอย่าง

```
nwritten = sendto ( sockfd,buffer,noctets,sendflag,(struct ,sockaddr *)
                    &serv_addr,sizeof(serv_addr));
```

Read and Recv System Call

Read and Recv System Call ใช้สำหรับการยอมรับ (Accept) ข้อมูลที่เข้ามาจากอีกฝ่ายหนึ่ง โดยที่ Recv System Call ทำหน้าที่คล้ายกับ Read System Call แต่มีตัวแปรที่มากกว่ารูปแบบแสดงดังนี้

- ```
read (socket , buefffer , length)
recv (socket , buffer_address , length , flags)
```
- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
  - buffer address : เป็นบัฟเฟอร์ที่ใช้เก็บข้อมูลในการรับ
  - length : จำนวน ไบต์ที่กำหนดเพื่อใช้ในการรับ
  - flags : เป็นตัวที่ใช้กำหนดเจาะจงทางเลือกพิเศษสำหรับการรับ

ผลที่ได้จาก System Call นี้คือ จำนวนไบต์ที่รับได้จริง

ตัวอย่าง

```
nread = read (sockfd,buffer,bufflength);
nread = recv (sockfd,buffer,beflength,recvflag);
```

### Recvform System Call

Recvform System Call คล้ายกับ Recv และ Read System Call แต่จะใช้กับการติดต่อแบบ Connectionless Application Protocol ที่ใช้ UDP Recvform System Call จะสร้างการเชื่อมต่อระหว่าง 2 กระบวนการซึ่งการติดต่อนี้จะสมบูรณ์เมื่อ ได้รับข้อมูลเข้ามาเรียบร้อยแล้ว มีรูปแบบดังนี้

```
recvform (socket , buffer_address , length ,flags ,source_address , address_length)
```

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- buffer address : เป็นบัฟเฟอร์ที่ใช้เก็บข้อมูลในการรับ
- length : จำนวน ไบต์ที่กำหนดเพื่อใช้ในการรับ
- flags : เป็นตัวที่ใช้กำหนดเจาะจงทางเลือกพิเศษสำหรับการรับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษานี้เท่านั้น เมื่ออนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- source address : แสดงแอดเดรสที่ต้องการส่งข้อมูล

- address length : เป็นเลขจำนวนเต็มแสดงความยาวของแอดเดรสในหน่วยไบต์

ผลที่ได้จาก System Call นี้คือ จำนวนไบต์ที่รับได้จริง

ตัวอย่าง

```
nwritten = recvfrom (sockfd,buffer,bufflength,recvflag,(struct ,sockaddr *)
 &serv_addr, &length);
```

### Close System Call

การเรียกใช้ Close System Call นั้นจะทำหลังจากมีเชื่อมต่อที่สมบูรณ์แล้ว การเรียกใช้ System Call นี้จะเกิดผลดีในการจัดเตรียมเนื้อที่ในหน่วยความจำ ในการติดต่อแบบ Connection Oriented Application Protocol ที่ใช้ TCP จะเรียก Close System Call หลังจากการติดต่อสมบูรณ์แล้วโดยที่ มันใจว่าข้อมูลต่างๆที่ต้องการส่งนั้นได้ถูกส่งออกไปเรียบร้อยแล้วก่อนที่การติดต่อจะสิ้นสุดลง มีรูปแบบดังนี้

```
close (socket)
```

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call

ตัวอย่าง

```
close (sockfd);
```

## 2.6 ระบบมัลติมีเดีย

การให้คำจำกัดความของคำว่า มัลติมีเดีย ในขณะนี้ยังไม่มีที่สิ้นสุด ทั้งนี้เนื่องมาจากการพัฒนาเทคโนโลยีต่าง ๆ ยังไม่มีที่สิ้นสุด จึงขอยกนิยามอย่างกว้าง ๆ ว่า คือ การรวบรวมสื่อต่าง ๆ ที่มีองค์ประกอบสำคัญใหญ่ ๆ 3 อย่าง คือ ภาพ เสียง การโต้ตอบซึ่งกันและกัน

### 2.6.1 สื่อที่เป็นองค์ประกอบสำคัญของมัลติมีเดีย

#### 2.6.1.1 ภาพ

เป็นสื่อที่มนุษย์เรานิยมใช้มาตั้งแต่อดีตและนิยมใช้มาจนถึงปัจจุบัน ความจริงแล้วสื่อที่เป็นเสียงนั้น เป็นสื่อแรกที่มนุษย์รู้จักและใช้ แต่เสียงนั้นมีข้อจำกัดอยู่ที่ระยะทางและภาษา การสื่อสารภายในเผ่าเดียวกันก็ไม่ใช่เป็นปัญหามาก แต่หากจะต่างเผ่าออกไป การติดต่อสื่อสารระหว่างกันจะเริ่มยุ่งยาก หากสังเกตร่องรอยที่ถูกบันทึกไว้โดยมนุษย์ในยุคแรก ๆ จะเป็นภาพวาดมา

ภาพในระบบมัลติมีเดียแบ่งออกเป็นประเภทใหญ่ ๆ ได้ดังนี้ คือ

1. ตัวอักษร (Text)

2. รูปภาพ (Image)

และสามารถแบ่งออกตามลักษณะของภาพเป็นประเภทใหญ่ ๆ ได้อีกคือ ภาพนิ่ง และภาพเคลื่อนไหว

ถ้าหากพิจารณาถึงที่มาของภาพแล้ว สามารถจำแนกออกเป็น 2 ลักษณะใหญ่ ๆ คือ

1. ภาพจากการจำลองแบบ (Imitated Image) ได้แก่ ภาพที่ถ่ายแบบมาจากของจริง โดยอาศัยเครื่องมือ

ช่วยในการจำลองแบบ เช่น กล้องถ่ายภาพนิ่ง กล้องวิดีโอ เครื่องอ่านภาพ (Scanner) เป็นต้น

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ภาพจากการสร้าง (Creative Image) เป็นภาพที่สร้างขึ้นจากจินตนาการของมนุษย์ เช่น ภาพวาด ภาพการ์ตูน เป็นต้น

เหตุที่เราไม่แยกตัวอักษรกับภาพออกจากกันก็เพราะถ้ามองในส่วนของการใช้งานคอมพิวเตอร์ในปัจจุบัน ตัวอักษรจัดเป็นภาพจากการสร้าง โดยที่การสร้างแบบตัวอักษร (Font) จะมีลักษณะการสร้างในเชิงรูปภาพ

### 2.6.1.2 เสียง

เสียงนับได้ว่าเป็นสื่อแรกที่มนุษย์ใช้ และเสียงหรือภาษาพูดก็เป็นสื่อที่มีความสำคัญไม่น้อยไปกว่าภาษาเขียน การผลิตภาษาเชิงภาพจะมีขั้นตอนยุ่งยากกว่าภาษาเสียง ซึ่งอาศัยหูเป็นสื่อกลางเพียงอย่างเดียวก็ใช้งานได้แล้ว อย่างไรก็ตาม เสียงก็มีข้อจำกัดที่ลดลงไปมากกว่าในอดีต เนื่องจากการพัฒนาสื่อทางเสียงได้รับการพัฒนาไปเร็วกว่าสื่อทางภาพ และที่สำคัญการพัฒนาสื่อทางเสียง จะไม่ร้ายแรงเท่ากับความผิดพลาดในเชิงภาพ ทั้งนี้ เพราะความผิดพลาดทางเสียงจะมีผลแค่เพียง ทำให้ความชัดเจนของเสียงลดลงหรือเพี้ยนไปจากเดิม เช่น หูมกเกินไป แห่ลมกเกินไป เป็นต้น แต่ถ้าเป็นความผิดพลาดในลักษณะของสื่อทางภาพแล้ว อาจถึงขั้นทำให้ใช้ไม่ได้เลยทีเดียว ระบบมัลติมีเดียจึงเป็นการประสานกันระหว่างสื่อทางภาพกับสื่อทางเสียงนั่นเอง

### 2.6.1.3 การโต้ตอบซึ่งกันและกัน

คำว่าโต้ตอบกันได้ (Interactive) หมายถึง แสดงผลการทำงานว่าถูกต้องหรือไม่ ใช้ได้หรือไม่ได้ในขณะนั้น ระบบมัลติมีเดียต้องมีการโต้ตอบกันได้จึงจะสมบูรณ์ จากลักษณะนี้เองที่ทำให้ขอบเขตของมัลติมีเดียกว้างขวางออกไป จำแนกได้ดังนี้

1. การตอบโต้กันได้ ในลักษณะตัวเลือก มีการจำลองสถานการณ์ (Simulation) จากปัญหาหนึ่งไว้หลายรูปแบบ โดยพิจารณาจากเงื่อนไขที่แตกต่างกัน หากสามารถจำลองสถานการณ์ได้มาก และซับซ้อนครอบคลุมได้มากเท่าไร จะทำให้ระบบมัลติมีเดียที่นั้นสมบูรณ์มากขึ้นเท่านั้น

2. การตอบโต้กันได้ ในลักษณะการติดต่อสื่อสาร ใช้สมรรถนะของการคมนาคมที่ทันสมัย เข้าช่วยผลลัพธ์ที่ได้จะปรับเปลี่ยนตามเหตุการณ์ที่เปลี่ยนไป เช่น เดิมถ้าต้องการทราบข้อมูลเพียงว่า มีโรงแรมอะไรบ้างเท่านั้น แต่ปัจจุบันเมื่อเราเลือกโรงแรมใดแล้ว เราจะทราบถึงขนาดว่าโรงแรมนั้นมีห้องพักกี่ห้องและมีห้องว่างหรือไม่ด้วย เป็นต้น

ขอบเขตของระบบมัลติมีเดีย ได้รับการพัฒนากว้างขวางมากขึ้นทำให้ข้อจำกัดในการใช้งานมีน้อยลง เดิมคิดกันแต่เพียงว่าระบบมัลติมีเดียจะเหมาะสำหรับงานนำเสนอเท่านั้นแต่ความจริงเราสามารถนำมัลติมีเดียมาประยุกต์ใช้งานได้มากมาย ได้แก่ ประยุกต์ใช้ในการศึกษา ด้านธุรกิจ ด้านการโฆษณา ด้านการให้ข้อมูลตามสถานที่ต่าง ๆ เป็นต้น

## 2.6.2 ส่วนประกอบของระบบมัลติมีเดีย

สิ่งที่จำเป็นสำหรับระบบมัลติมีเดีย แยกออกเป็น 2 ส่วน คือ

2.6.2.1 ฮาร์ดแวร์มัลติมีเดีย องค์ประกอบที่สำคัญของระบบมัลติมีเดียคือ ภาพ เสียง และการตอบโต้กันได้ ส่วนที่เป็นฮาร์ดแวร์จึงต้องจำแนกออกไปตามองค์ประกอบดังกล่าว โดยมีคอมพิวเตอร์เป็นส่วนสำคัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คอมพิวเตอร์ที่ใช้งานในระบบมัลติมีเดียต้องแยกจากกัน ระหว่างการสร้างงานจำเป็นต้องใช้คอมพิวเตอร์ที่มีสมรรถนะสูง ถ้าเป็นระดับเวิร์คสเตชันจะดีมาก ซึ่งจะเป็นตัวเชื่อมโยงไปหาฮาร์ดแวร์อื่น ๆ

1. ฮาร์ดแวร์สำหรับงานด้านภาพ อุปกรณ์ที่สำคัญและจำเป็นนอกจากคอมพิวเตอร์ ได้แก่ เครื่องอ่านภาพ (Scanner) ใช้สำหรับงานสำเนาภาพจากต้นฉบับที่เป็นภาพนิ่ง มีหลายแบบ หลายระดับ มีลักษณะเป็นภาพสองมิติ กล้องถ่ายภาพเชิงตัวเลข (Digital Camera) เป็นกล้องถ่ายภาพธรรมดาที่มีการเปลี่ยนส่วนรับภาพที่เป็นฟิล์มเป็นตัวเปลี่ยนสัญญาณภาพมาเป็นสัญญาณเชิงตัวเลขหรือเรียกว่า CCD (Charge Couple Device) ใช้ทำสำเนาภาพนิ่งได้ทั้ง 2 มิติ และ 3 มิติ กล้องถ่ายภาพวิดีโอ (Video Camera) ใช้งานถ่ายภาพเคลื่อนไหว และใช้สัญญาณเชิงตัวเลขและใช้ CCD เช่นเดียวกับกล้องถ่ายภาพเชิงตัวเลข การ์ดแปลงสัญญาณอนาล็อกเป็นสัญญาณเชิงตัวเลข (Digital Card) เนื่องจากเครื่องเล่นวีดีโอและกล้องถ่ายภาพวิดีโอ ส่วนใหญ่ให้สัญญาณแบบอนาล็อก การจะนำสัญญาณจากอุปกรณ์ดังกล่าวไปใช้งานกับคอมพิวเตอร์ จึงจำเป็นต้องแปลงสัญญาณจากอนาล็อกให้เป็นสัญญาณเชิงตัวเลข

2. ฮาร์ดแวร์สำหรับงานด้านเสียง ในส่วนของระบบเสียงนั้น เดิมเป็นแบบอนาล็อก แต่ในปัจจุบันได้มีการประยุกต์และพัฒนาระบบเสียงให้ใช้สัญญาณเชิงตัวเลขได้ ฮาร์ดแวร์ที่ใช้สำหรับงานด้านเสียงจึงแทบจะประยุกต์ใช้กับระบบมัลติมีเดียได้ทันที ทั้งโดยตรงและโดยอ้อม อุปกรณ์มีดังนี้ เครื่องเล่นซีดี (CD-ROM) ถ้าไม่ต้องการบันทึกเสียงลงในคอมพิวเตอร์ก็สามารถจะแยกใช้ระบบเสียงจากภายนอกได้ โดยอาศัยการควบคุมการทำงาน วิธีที่สะดวกที่สุดจะเป็นการใช้เครื่องเล่นซีดี-รอม ที่ในปัจจุบันเล่นได้ทั้งระบบเสียงและระบบอ่านข้อมูลคอมพิวเตอร์ เพียงแต่ต้องบันทึกลงแผ่นซีดีมาก่อนจากนั้นจึงมาเปิดใช้งาน โดยอาศัยคำสั่งจากตัวเครื่องคอมพิวเตอร์เป็นตัวควบคุมอีกทีหนึ่ง เสียงสำเร็จรูป (Clip Sound) เป็นระบบเสียงสำเร็จรูปในลักษณะไฟล์สัญญาณเชิงตัวเลขสามารถใช้งานได้ทันที ส่วนใหญ่จะเป็นเสียงเพลงหรือเอฟเฟ็กต์ต่าง ๆ

#### 2.6.2.2 ซอฟต์แวร์มัลติมีเดีย แยกออกเป็น 2 ตอน คือ ตอนสร้างกับตอนใช้งาน

1. ซอฟต์แวร์สร้างงานระบบมัลติมีเดีย แยกออกเป็นสองส่วนคือ ซอฟต์แวร์สำหรับสร้างภาพและเสียง กับซอฟต์แวร์จัดระบบ

- ซอฟต์แวร์สำหรับสร้างภาพและเสียง มักมีการใช้งานและเป็นพื้นฐานของผู้ใช้คอมพิวเตอร์อยู่แล้ว มีการตกแต่ง ตัดต่อ ดัดแปลง สร้างใหม่ ซึ่งแยกกันทำเป็นขั้นเป็นตอนได้

- ซอฟต์แวร์จัดระบบมัลติมีเดีย เป็นซอฟต์แวร์ที่ควบคุมงานส่วนต่าง ๆ มาจัดลำดับ เพื่อให้มีการตอบโต้กันได้ เช่น บอกให้รู้ว่าถึงตอนนี้จะมีการแสดงต่อไปจะมีเสียง หรือถ้าคุณตอนนี้มีภาพนั้นภาพนี้ออกมา หรือมีเสียงนั้นเสียงนี้ เป็นต้น ซอฟต์แวร์ประเภทนี้ยังไม่เป็นที่คุ้นเคยมากนัก ทำให้ยังมีน้อย ราคาจึงสูงและที่สำคัญยังต้องใช้กับคอมพิวเตอร์ที่มีสมรรถนะสูงด้วย

2. ซอฟต์แวร์ใช้งานระบบมัลติมีเดีย ระบบมัลติมีเดียนี้เกิดจากการสร้างขึ้นมาโดยผ่านทางระบบคอมพิวเตอร์ ทำให้ไม่เป็นการยากนักหากจะใช้งาน แต่มีข้อจำกัดอยู่ที่ความง่ายในการใช้งาน

### 2.6.3 เทคโนโลยีที่เกี่ยวข้องกับมัลติมีเดีย

เนื่องจากระบบคอมพิวเตอร์มัลติมีเดีย เป็นการรวบรวมเทคโนโลยีหลายอย่างเข้าด้วยกัน เพื่อให้เกิดความสมบูรณ์ในการทำงาน เทคโนโลยีเหล่านั้น ได้แก่

1. การพัฒนาเทคโนโลยีในการบันทึกข้อมูลการทำงานของมัลติมีเดียประกอบไปด้วยภาพและเสียง
2. การพัฒนาด้านระบบคอมพิวเตอร์เครือข่าย สิ่งที่ระบบคอมพิวเตอร์มัลติมีเดียเข้าไปมีบทบาทร่วมกับระบบคอมพิวเตอร์เครือข่าย เช่น การติดต่อสื่อสารด้วยระบบ Electronics Mail ซึ่งเดิมเป็นการติดต่อที่เป็นลักษณะ Text Base เท่านั้น เป็นการนำสองเทคโนโลยีมารวมช่วยกัน ทำให้การติดต่อสื่อสารในระบบเครือข่ายคอมพิวเตอร์ทำได้ทั้งที่เป็นภาพและเสียง
3. การพัฒนาเทคนิคการย่อขนาดข้อมูล การย่อข้อมูลที่มีประสิทธิภาพจะเป็นปัจจัยสำคัญอย่างหนึ่งในการทำงานของระบบคอมพิวเตอร์มัลติมีเดีย
4. การพัฒนาไมโครคอมพิวเตอร์ การทำงานของคอมพิวเตอร์มัลติมีเดียเป็นการทำงานที่เกี่ยวข้องกับข้อมูลในปริมาณมหาศาล กระบวนการย่อและขยายขนาดข้อมูลจะต้องเกิดอย่างรวดเร็วและมากพอที่จะทำให้การติดต่อส่งข้อมูลระหว่างหน่วยความจำและอุปกรณ์ต่าง ๆ ไม่เกิดการหยุดชะงัก เพราะถ้าเกิดเหตุการณ์เช่นนี้จะทำให้การแสดงผลทั้งภาพและเสียงอาจเพี้ยนไปจากของจริงได้
5. การพัฒนาของจอภาพ
6. การพัฒนาอุปกรณ์ป้อนข้อมูล
7. การพัฒนาซอฟต์แวร์ ส่วนหนึ่งที่ทำให้โลกคอมพิวเตอร์มัลติมีเดียเป็นจริงก็คือ การพัฒนาซอฟต์แวร์ให้มีประสิทธิภาพสูงและมีการใช้งานได้ง่ายขึ้น การพิจารณาเลือกซอฟต์แวร์เพื่อนำมาทำงานด้านมัลติมีเดียอาจพิจารณาได้จาก
  - ความง่ายในการใช้งาน
  - ความสามารถในการนำเสนองาน
  - ความสามารถในการติดต่อกับผู้ใช้
  - ความสามารถในการใช้ฟังก์ชันในการคำนวณ
  - ความสามารถในการใช้งานร่วมกับ โปรแกรมอื่น ๆ
  - มี Library สนับสนุนการทำงาน
  - ความสามารถในการทำเอกสารประกอบ โปรแกรม
  - ความสามารถในการส่งแอปพลิเคชันที่เสร็จแล้วให้ผู้ใช้

### \* 2.6.4 Microsoft Audio Vidro Interleave (AVI)

เป็น Format ของไฟล์ที่ใช้ในการ Capture, แก้ไข, PlayBack ลำดับของภาพและเสียง โดยทั่วไปไฟล์ AVI จะเก็บ stream ของข้อมูลหลากหลายรูปแบบ แต่ส่วนใหญ่จะเก็บข้อมูลของภาพและเสียงเป็นหลัก ไฟล์ AVI พื้นฐานจะเก็บเฉพาะลำดับของรูปภาพ จนถึงไฟล์ AVI แบบพิเศษที่สามารถเพิ่มเติม Control Track หรือ MIDI Track ลงไปได้ Control Track จะทำให้สามารถควบคุมอุปกรณ์ภายนอก เช่น เครื่องบันทึก Video Disc ได้ และ MIDI Track ทำให้สามารถเล่นเสียง MIDI เป็น Background ได้ แต่ต้องใช้โปรแกรมเฉพาะจึงจะเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สามารถใช้ความสามารถเหล่านี้ได้ โปรแกรมประยุกต์ทั่วไปไม่สามารถใช้ความสามารถเหล่านี้ได้ จึงแสดงผลเป็นภาพและเสียงได้เท่านั้น

## ✕ 2.6.5 Picture File

Format ของ Image File ในปัจจุบันมีอยู่มากมายตามลักษณะการใช้งาน เราสามารถแบ่งลักษณะของ Image File เป็น 2 ประเภท คือ

1.Bitmap File เป็นไฟล์ที่เก็บรายละเอียดของภาพในลักษณะของ Pixel จะมีการกำหนดไว้ อย่างแน่นอนว่า ที่ตำแหน่งใดของภาพจะมีลักษณะอย่างไร เมื่อเราย่อหรือขยายภาพของไฟล์ชนิด Bitmap นี้ เรา จะสูญเสียความคมชัดของภาพไป แต่ในการเก็บภาพเพื่อใช้ในการงานมัลติมีเดีย เรามักใช้ Image File ประเภทนี้ เพราะสามารถเรียกแสดงได้รวดเร็ว ในปัจจุบันได้นำเทคโนโลยีการบีบข้อมูลมาใช้กันอย่างแพร่หลาย และได้ นำมาใช้ในการเก็บ Still Image ด้วย เช่น RLE File, JPEG File, GIF File ในแต่ละวิธีจะมีอัลกอริทึมในการ บีบข้อมูลที่แตกต่างกันออกไป

2.Vector File เป็นไฟล์ที่สามารถย่อและขยายภาพได้โดยมีการสูญเสียรายละเอียดน้อยมาก แต่ เรามักใช้กับภาพที่ไม่มีรายละเอียดมากหรือลักษณะของภาพที่มีการย่อและขยายมาก เช่น Logo และในงาน เขียนแบบต่าง ๆ

### BMP

เป็น File Format มาตรฐานของ Windows โปรแกรมส่วนใหญ่ของ Windows ที่สามารถทำงานกับ Still Image ได้ จะสามารถทำงานกับไฟล์ใน Format นี้ได้ และ Visual Basic ก็สามารถใช้ไฟล์ใน format นี้ได้ โดยตรง อาทิเช่น Form .Picture Control , Image Control ไฟล์ใน Format นี้สามารถแสดงสีได้ถึง 16.7 ล้านสี (True Color 24 Bits) แต่ข้อด้อยของไฟล์ใน Format นี้คือ ไม่มีการใช้เทคโนโลยีการบีบข้อมูลเลยทำให้ไฟล์มี ขนาดค่อนข้างใหญ่มากเมื่อเปรียบเทียบกับไฟล์ใน Format อื่น

### รูปแบบข้อมูลภาพแบบ BMP

รูปแบบที่มีใช้อยู่โดยทั่วไปมีรูปแบบดังนี้คือ GIF IFF/LBM WPG BMP PIC TGA PCX และ TIFF สำหรับรูปแบบใหญ่ๆ ที่นิยมใช้กันมากคือ PCX GIF BMP และ TIFF แต่ รูปแบบหนึ่งที่มีความนิยมมากที่สุด คือ BMP หรือบิตแมพ สาเหตุก็เพราะว่าข้อมูลแบบบิตแมพนั้น ไม่มีการบีบข้อมูล (compression) จึงมีความ รวดเร็วและสะดวกในการอ่านข้อมูลอย่างมากและเป็นรูปแบบข้อมูลกลางที่ใช้ในการแลกเปลี่ยนระหว่างกัน ของโปรแกรมประยุกต์ ต่างๆ ใช้ในการส่งข้อมูลไปให้อุปกรณ์ต่างๆ เช่น อุปกรณ์การแสดงผล เป็นต้นแต่ในรูป แบบบิตแมพนี้ต้องเปลืองเนื้อที่ฮาร์ดดิสในการจัดเก็บข้อมูลมากกว่ารูปแบบอื่นที่ใช้การเข้ารหัสบีบข้อมูล

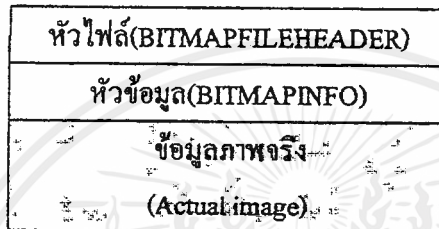
### รูปแบบข้อมูลแบบ BMP

รูปแบบสำหรับใช้กับ โปรแกรมระบายสีในวินโดวเวอร์ชัน (version) 3.0 เป็นต้นมา โดยรูปแบบข้อมูล แบบนี้สามารถเป็นข้อมูลสีได้ตั้งแต่ 1-24 บิต มีส่วนหัว (header) ในการบอกรายละเอียดต่างๆ ของภาพ โดย

กำหนดในลักษณะโครงสร้าง (structure) ในภาษาระดับสูง ข้อมูลภาพในรูปแบบบิตแมพประกอบด้วย 3 ส่วน คือ

- หัวไฟล์(BITMAPFILEHEADER)
- หัวข้อมูล(BITMAPINFO) ซึ่งรวมทั้งข้อมูลต่างๆ และพาเลตของสี(color palette)
- ข้อมูลภาพจริง

รูปที่ 2.8 แสดงส่วนประกอบกันของโครงสร้างที่ใหญ่ที่สุดของภาพถ่ายในรูปแบบบิตแมพ ซึ่งข้อมูลพาเลตสีและข้อมูลภาพจะเปลี่ยนโครงสร้างไปตามรูปแบบของจำนวนสีของภาพและวิธีการถอดรหัสที่ใช้ในการบีบอัดข้อมูลภาพ



รูป 2.8 แสดง โครงสร้างของภาพถ่ายในรูปแบบบิตแมพ

#### หัวไฟล์(BITMAPFILEHEADER)

หัวของไฟล์ (BITMAPFILEHEADER) จะประกอบด้วยข้อมูลเกี่ยวกับชนิดของ รูปแบบข้อมูลภาพ ขนาดของภาพและ โครงร่างดังนี้

##### BITMAPFILEHEADER ประกอบด้วย

- WORD      bfType;
- DWORD     bfSize;
- WORD      bfReserved1;
- WORD      bfReserved2;
- DWORD     bfOffBits

โดย bfType เป็นชนิดของรูปแบบข้อมูลภาพในที่นี้สำหรับบิตแมพคือ "BM"

bfSize เป็นขนาดของไฟล์ทั้งหมด

bfReserved1,2 ไม่ได้ใช้งานปกติกำหนดให้มีค่าเท่ากับศูนย์

bfOffBits เป็นขนาดของหัวไฟล์ทั้งหมด ใช้เพื่อข้าม ไปจุดเริ่มต้นของข้อมูลภาพจริง

#### หัวข้อมูล(BITMAPINFO)

หัวข้อมูลจะเป็นตัวกำหนดขนาดต่างๆ (dimensions)และข้อมูลสี(color information)ดังนี้

##### BITMAPINFO ประกอบด้วย

- BITMAPINFOHEADER สำหรับ bmiheader; เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-RGBQUAD

bmiColors[1];

โดยที่ bmiHeader เป็นข้อมูลเกี่ยวกับขนาดต่างๆ จำนวนสีและรูปแบบของสี  
bmiColors เป็นอาร์เรย์ข้อมูลสีอาร์จีบีที่เป็นตัวกำหนดสีในการแสดงผลของภาพ  
ที่เป็นรายละเอียดต่างๆ จะอยู่ใน BITMAPINFOHEADER ดังนี้

#### STRUCTURE BITMAPINFOHEADER ประกอบด้วย

|        |                  |
|--------|------------------|
| -DWORD | biSize;          |
| -DWORD | biWidth;         |
| -DWORD | biHeight;        |
| -WORD  | biPlanes;        |
| -WORD  | biBitCount;      |
| -DWORD | biCompression;   |
| -DWORD | biSizeImage;     |
| -DWORD | biXPelsPerMeter; |
| -DWORD | biYPelsPerMeter; |
| -DWORD | biClrUsed;       |
| -DWORD | biClrImportant;  |

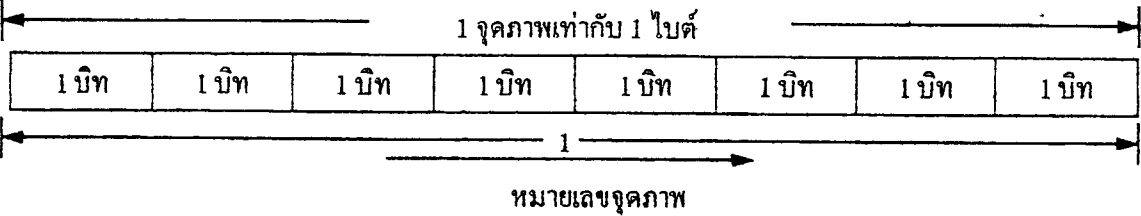
โดยที่ biSize เป็นขนาดของส่วนหัวข้อมูล (BITMAPINFOHEADER) มีหน่วยเป็น ไบต์  
biWidth เป็นความกว้างของภาพ (มีหน่วยเป็น จุดภาพ/เส้น)  
biHeight เป็นความสูงของภาพ (มีหน่วยเป็น จุดภาพ/เส้น)  
biPlanes เป็นจำนวนหน้าของสี (color plane) สำหรับอุปกรณ์เป้าหมาย (target device)  
ปกติกำหนดให้เป็น 1 เสมอ  
biBitCount จำนวนบิตต่อจุดภาพ(1,4,8,24)  
biCompression ชนิดของการบีบอัด(compression)  
biSizeImage เป็นขนาดของภาพ(มีหน่วยเป็น ไบต์)  
biXPelsPerMeter ความละเอียดสำหรับอุปกรณ์เป้าหมายในแนวนอนต่อหนึ่งเมตร  
biYPelsPerMeter ความละเอียดสำหรับอุปกรณ์เป้าหมายในแนวตั้งต่อหนึ่งเมตร  
biClrUsed จำนวนดัชนีสี (color index) ในตารางสี (color table) มีค่าได้ 3 กรณี  
biClrImportant จำนวนความสำคัญของดัชนีสี (color index) ในการแสดงผล ถ้าเป็นศูนย์  
ทุกสีจะมีความสำคัญเท่ากันหมด

#### **ข้อมูลภาพจริง (Actual Image Information)**

ส่วนของข้อมูลบิตแมพจริงจะมีการเก็บที่แตกต่างกันตามจำนวนบิตข้อมูลภาพ (biBitCount) และ  
ลักษณะการบีบอัด biCompression แต่โดยทั่วไปแล้วข้อมูลในรูปแบบบิตแมพปกติที่นิยมในปัจจุบันในการส่ง

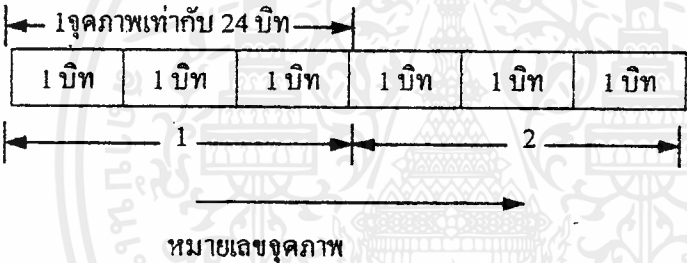
ข้ามระหว่างโปรแกรมประยุกต์ต่าง ๆ นั้น จะไม่ใช้การบีบอัดข้อมูล ดังนั้นจึงแบ่งรูปแบบข้อมูลในลักษณะที่นิยมใช้กันในปัจจุบันได้ดังนี้

**กรณีที่ 1** ถ้าเป็น 8 บิตต่อภาพ หมายความว่าดัชนีสี bmiColors ที่เป็นแอร์เรย์ จำนวน 256 แอร์เรย์ คือ มีสี 256 สี โดยข้อมูลแต่ละ 8 บิตจะแทนหนึ่งจุดภาพ ดังรูป



รูป 2.9 แสดงข้อมูลภาพบิตแมพ 256 สี

**กรณีที่ 2** ถ้าเป็น 24 บิตต่อจุดภาพ หมายความว่า จะสามารถแสดงสีได้พร้อมๆ กันเท่ากับ  $2^{24}$  คือ 16.7 ล้านสีคือสามารถผสมสีได้โดยตรง ไม่ต้องมีการใช้ดัชนีสี ดังนั้นในกรณีนี้ bmiColors เท่ากับ NULL ข้อมูลแต่ละ 3 ไบต์ที่เรียงกันจะแสดง relative intensity ของสีฟ้า สีเขียว และสีแดง (BGR) ตามลำดับ โดยข้อมูลแต่ละ 24 บิตจะแทน 1 จุดภาพดังรูป



รูป 2.10 แสดงข้อมูลภาพบิตแมพ 16.7 ล้านสี

**GIF**

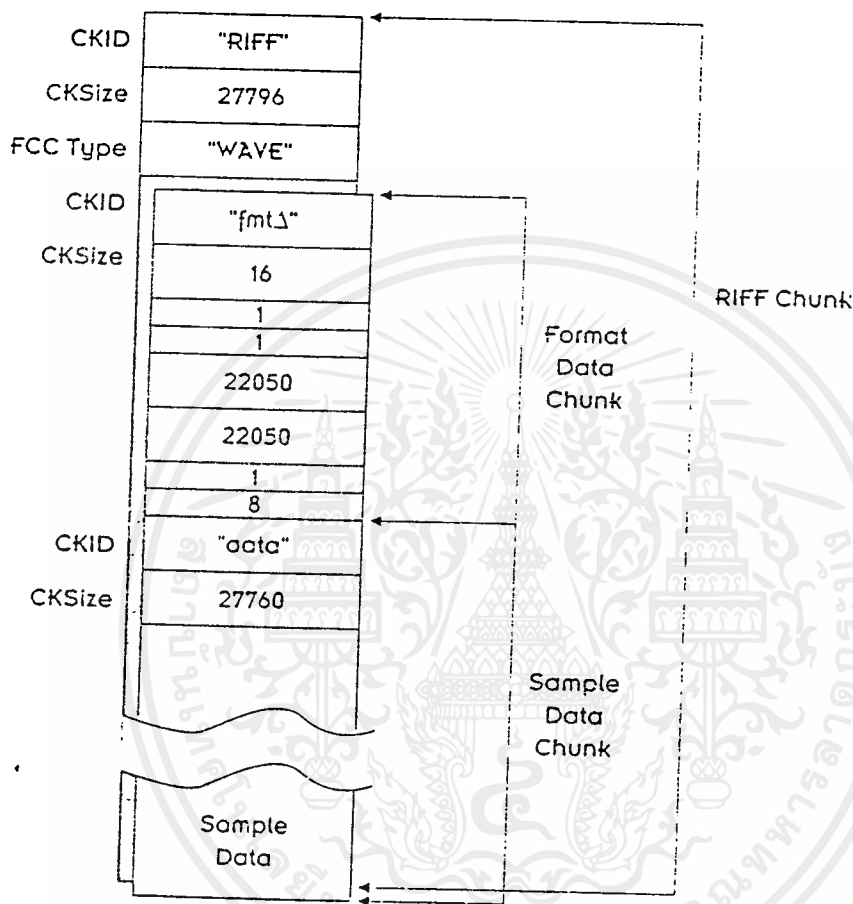
เป็น File Format ของ CompuServe ซึ่งเป็น Format ของไฟล์ที่ใช้ในการสื่อสารข้อมูล (Down Load, Up Load) ไฟล์ใน Format นี้จะสามารถเรียกใช้ได้โดยโปรแกรมตกแต่งภาพ (Image Retouching) โดยทั่วไปสามารถเรียกใช้ได้ใน Visual Basic โดยผ่านทาง Third Party Control ที่ชื่อว่า GIFBOX.BOX ซึ่งมีข้อดีกว่า Image Control และ Picture Control ไฟล์ใน Format นี้สามารถแสดงจำนวนสีได้ 256 สี (8 Bits) มีการใช้เทคโนโลยีการบีบข้อมูลเข้ามาช่วย เพื่อให้ขนาดของไฟล์ลดลง เพื่อประโยชน์ในการส่งผ่านข้อมูล คุณภาพของภาพหลังจากบีบข้อมูลแล้ว ไม่มีการสูญเสียรายละเอียดของภาพแต่อย่างใด

**JPEG**

เป็น Format ของ Still Image ที่กำลังเป็นที่นิยมมากในการเก็บข้อมูล เพราะว่าสามารถบีบข้อมูลได้มาก มีจำนวนสี 16.7 ล้านสี สามารถกำหนด Factor ของการบีบขนาดภาพได้ และใน Visual Basic สามารถใช้ Third Party Control QTPIC.VBX ในการเรียกภาพขึ้นมาแสดง แต่ข้อเสียของ File Format นี้ คือมีการสูญเสียรายละเอียดของภาพตามอัตราการบีบข้อมูลและการเรียกใช้ใน Visual Basic ยังมีปัญหาอยู่มาก เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6.6 โครงสร้างของ RIFF ไฟล์

ไฟล์ RIFF ( ย่อมาจาก Resource Interchange File Format ) เป็นไฟล์ที่ได้รับการพัฒนาสำหรับการจัดเก็บข้อมูลทางด้านมัลติมีเดีย ( Multimedia ) โดยเฉพาะ โดยข้อมูลที่เก็บลงในรูปแบบของไฟล์ RIFF จะจัดเก็บเป็นบล็อกๆ โดยแต่ละบล็อกเราเรียกว่า Chunk สำหรับเก็บไฟล์ที่มาระบบปฏิบัติการวินโดวส์ ดังรูป 2.11 เป็น ตัวอย่างโครงสร้างของ RIFF ไฟล์



รูป 2.11 โครงสร้างของ RIFF ไฟล์

ในระดับสูงสุดของ Chunk โดยตัวมันเอง จะระบุชื่อซึ่งเป็นตัวระบุของ Chunk Tds ถ้าจุดสูงสุดของไฟล์เสียงจะพบว่า 4 ไบท์แรกจะบรรจุตัวอักษร R,I,F และ F อยู่ การอ่านและการเขียน RIFF ไฟล์ จะใช้มาตรฐานโครงสร้างข้อมูลซึ่งเรียกว่า MMCKINFO ( Multimedia Chunk Information ) ซึ่งจะมีโครงสร้างดังนี้

ข้อมูลโครงสร้าง MMCKINFO

เป็นข้อมูลโครงสร้างสำหรับจัดเก็บข้อมูลของบล็อกข้อมูล RIFF โดยแต่ละฟิลด์มีรายละเอียดดังนี้

ckid

เป็นข้อมูลสตริงขนาด 4 ไบท์ ซึ่งฟิลด์ชนิดนี้ได้ถูกประกาศให้เป็นข้อมูลชนิด FOURCC ซึ่งข้อมูลโครงสร้างนี้ก็จะประกอบไปด้วยสมาชิกเพียงฟิลด์เดียวคือ

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ การแก้ไขเนื้อหาโดยไม่ได้รับอนุญาตให้โทษตามกฎหมาย ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Type FOURCC  
FourBytes As String \*4  
End Type

cksize

ข้อมูลชนิดเลขจำนวนเต็ม long บอกขนาดของฟิลด์ข้อมูลของกลุ่ม โดยที่ค่าของ cksize จะเท่ากับขนาดของฟิลด์ข้อมูล และ ไม่รวม 4 ไบต์สำหรับฟิลด์ cksize

fccType

ข้อมูลชนิดสตริงขนาด 4 ไบต์เช่นเดียวกับฟิลด์ ckid ซึ่งจะจัดเก็บสตริงซึ่งบอกถึงประเภทของรูปแบบ ( form ) สำหรับบล็อก RIFF และบอกถึงประเภทของ List ของบล็อก LIST

dwFlags

ข้อมูลชนิดตัวเลขจำนวนเต็ม long ซึ่งเราจะใช้กำหนดเงื่อนไขการทำงานของฟังก์ชันไฟล์มัลติมีเดีย I/O ซึ่งสามารถกำหนดได้ตามค่าคงที่ดัง ตารางที่ 2.4

| ค่าคงที่       | ค่าตัวเลข | รายละเอียด                                                 |
|----------------|-----------|------------------------------------------------------------|
| MMIO_READ      | &H0       | กำหนดให้เปิดไฟล์สำหรับการอ่านเท่านั้น                      |
| MMIO_WRITE     | &H1       | กำหนดให้เปิดไฟล์สำหรับการเขียนเท่านั้น                     |
| MMIO_READWRITE | &H2       | กำหนดให้เปิดไฟล์สำหรับการทั้งการอ่านและการเขียน            |
| MMIO_COMPAT    | &H0       | กำหนดให้เปิดไฟล์ในโหมดของการคอมแพททิเบิล                   |
| MMIO_EXCLUSIVE | &H10      | กำหนดให้เปิดไฟล์ในโหมด Exclusive เพื่อใช้ในงานส่วนตัว      |
| MMIO_DENYWRITE | &H20      | ให้เปิดไฟล์โดยไม่ให้มีการเขียนทับจากแอปพลิเคชันอื่น        |
| MMIO_DENYRED   | &H30      | กำหนดให้เปิดไฟล์โดยไม่มีการอ่านจากแอปพลิเคชัน              |
| MMIO_DENYNONE  | &H40      | กำหนดให้เปิดไฟล์โดยสามารถอ่านหรือเขียนทับได้จากแอปพลิเคชัน |

ตาราง 2.4 ค่าคงที่ของฟังก์ชันมัลติมีเดีย

แต่สำหรับส่วนถัดไปของ RIFF ที่เรียกว่า subchunk หรือ format chunk นั้น จะมีความแตกต่างจากบล็อกอื่นๆ อยู่บ้าง ดังนั้นเราจะอ่านข้อมูลในบล็อกนี้โดยใช้ข้อมูลโครงสร้าง PCMWAVEFORMAT ดังมีรายละเอียดดังนี้

โครงสร้างของไฟล์เสียง ( WAVE FILE )

การที่จะเข้าใจว่าไฟล์เสียงมีหลักการทำงานอย่างไรนั้น เราจะพิจารณาจากตัวอย่างซึ่งทำการวิเคราะห์ไฟล์ wave

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในระดับต่ำที่สุดของไฟล์เสียงจะบรรจุด้วยกลุ่ม ( chunk ) 3 กลุ่ม โดยที่กลุ่ม ของ RIFF จะมีข้อมูลระบุขนาดมากที่สุด ซึ่งในความเป็นจริง กลุ่มของไฟล์เสียงทั้งหมดก็คือกลุ่ม RIFF นั่นเอง ซึ่งการตรวจสอบขนาด ( คำสั่ง cksize ) จะพบได้หลังจากปรากฏ 'RIFF' อย่างทันที ถัดไปคือตรวจสอบค่าอุปกรณ์ ( คำสั่ง ckID ) จะระบุค่าของไฟล์หารด้วย 8 ไบต์ ที่ต้องการจะบันทึกลงใน RIFF ไฟล์ ในลำดับถัดไป จะเรียกว่ากลุ่มย่อย ( subchunk ) ซึ่งจะระบุอยู่ในกลุ่ม RIFF ส่วนแรกของกลุ่มนี้คือ 'fmt' จะระบุข้อมูลที่สำคัญของโครงสร้างไฟล์เสียงแบบ พีซีเอ็ม ( PCMWAVEFORMAT ) ซึ่งอยู่ในรูปของข้อมูลแบบ ดิจิตอล ( digital ) ในกลุ่มย่อยที่ 2 จะบรรจุข้อมูล ( data ) เป็นส่วนที่ใหญ่ส่วนหนึ่งของไฟล์ ตอนจบของข้อมูล เป็นข้อตกลงว่าเป็นการสิ้นสุดของ RIFF ขนาดของ RIFF เท่ากับผลรวมของจำนวนไบต์ที่มีอยู่ของ 'fmt' และ 'data' ตัวอย่างของไฟล์ .Wav มีในตารางที่ 2.5

RIFF ไฟล์อาจจะบรรจุไปด้วยชนิดของกลุ่มอื่น เรียกว่า LIST ซึ่งรายการที่ระบุลงไปนี้จะจะเป็นข้อมูลที่บอกถึง ลิขสิทธิ์ หรือ ข้อมูลส่วนตัวของผู้ผลิต

| Position |     | Size in Bytes   | Contents | Comments                                             |
|----------|-----|-----------------|----------|------------------------------------------------------|
| Hex      | Dec |                 |          |                                                      |
| 0000     | 0   | 4               | "RIFF"   | Each byte contains one character. ckId               |
| 0004     | 4   | 4               | 27796    | Equals the file size minus eight bytes. ckSize       |
| 0008     | 8   | 4               | ""WAVE"  | iccType                                              |
| 000C     | 12  | 4               | "fmt "   | Next ckID: notice the blank, must be four characters |
| 0010     | 16  | 4               | 16       | The WAVE format chunk is 16 bytes, ckSize            |
| 0014     | 20  | 2               | 1        | 1 indicates a PCM Wave format. wFormatTag            |
| 0016     | 22  | 2               | 1        | Number of channels, nChannels                        |
| 0018     | 24  | 4               | 22050    | Sampling rate, nSamplesPerSec                        |
| 001C     | 28  | 4               | 22050    | nAvgBytesPerSec                                      |
| 0020     | 32  | 2               | 1        | Effectively bytes per sample. nBlockAlign            |
| 0022     | 34  | 2               | 3        | wBitsPerSample                                       |
| 0024     | 36  | 1               | "data"   | Next ckID: this chunk contains the wave data itself  |
| 0026     | 40  | 4               | 27760    | Next ckID: size of wave data                         |
| 002C     | 44  | Depends On Data |          | The digitized audio data                             |

ตาราง 2.5 โครงสร้างของไฟล์.wav

### ข้อมูลโครงสร้าง PCMWAVEFORMAT

โดยที่ข้อมูลต่าง ๆ ที่จัดเก็บในบล็อกนี้จะเป็นข้อมูลที่ถูกใช้ในการคำนวณและบอกคุณลักษณะของข้อมูลของสัญญาณเสียง ในรูปแบบ PCM สำหรับข้อมูลแบบโครงสร้าง PCMWAVEFORMAT จะประกอบด้วยสมาชิกเพียง 2 ฟیلด์ เท่านั้นคือ

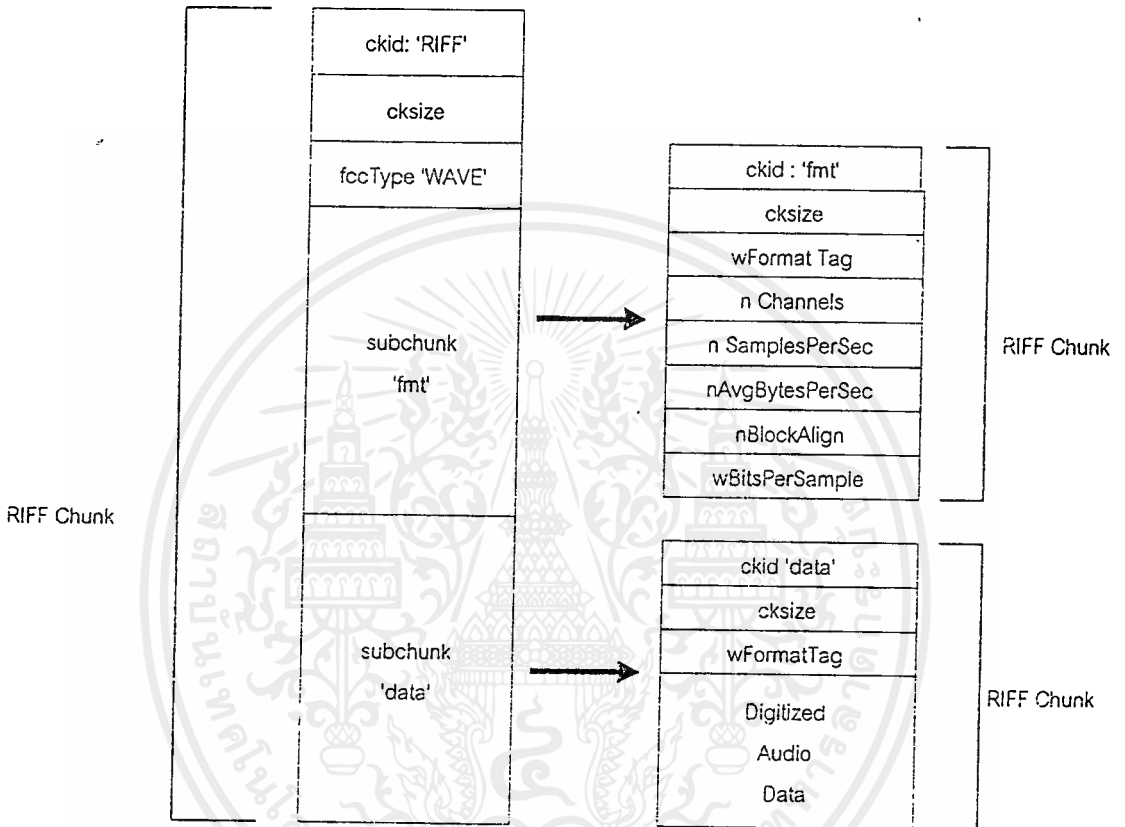
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

# Type PCMWAVEFORMAT

wf As WAVEFORMAT

wBitsPerSample As Integer

End Type



รูป 2.12 ตัวอย่างของ PCMWAVEFORMAT

เป็นข้อมูล โครงสร้าง WAVEFORMAT ซึ่งแต่ละตัวสมาชิกมีรายละเอียดดังนี้

## Type WAVEFORMAT

|                 |    |         |                                               |
|-----------------|----|---------|-----------------------------------------------|
| wFormatTag      | As | Integer | 'Format Type                                  |
| nChannels       | As | Integer | 'Number of chennels(i.e. mono , stereo , etc) |
| nSamplesPerSec  | As | Long    | 'Sample rate                                  |
| nAvgBytesPerSec | As | Long    | 'For buffer estimation                        |
| nBlockAlign     | As | Integer | 'Block size of data                           |

End Type

ข้อมูลชนิดจำนวนเต็ม กำหนดชนิดรูปแบบ ซึ่งในปัจจุบัน ได้มีการกำหนดในรูปแบบเดียวเท่านั้น คือ

| ค่าคงที่        | ค่าตัวเลข | รายละเอียด         |
|-----------------|-----------|--------------------|
| WAVE_FORMAT_PCM | 1         | ข้อมูลในรูปแบบ PCM |

ตาราง 2.6 กำหนดค่าคงที่

nChannels

ข้อมูลชนิดจำนวนเต็ม กำหนดจำนวนแชนแนลของข้อมูล โดยที่ข้อมูลโมโน (mono) จะใช้เพียง 1 แชนแนล ในขณะที่ข้อมูลสเตอริโอ (stereo) จะใช้ 2 แชนแนล

nSamplePerSec

ข้อมูลชนิดเลขจำนวนเต็ม long กำหนดอัตราส่วนการสุ่มข้อมูล ในหน่วยของอัตราการสุ่มข้อมูล ใน 1 วินาที

nAvgBytesPerSec

ข้อมูลชนิดเลขจำนวนเต็ม long กำหนดขนาดของข้อมูลเฉลี่ยในหน่วยของ ไบต์ สำหรับการขนถ่ายในช่วงเวลา 1 วินาที

nBlockAlign

ข้อมูลชนิดเลขจำนวนเต็ม กำหนดค่าขอบเขตการจัดเรียง (alignment) บล็อกในหน่วยของไบต์ โดยที่ข้อมูลในรูปแบบ PCM จะเรียงข้อมูลตามจำนวนไบต์ที่ใช้ในการสุ่มแต่ละครั้ง เช่น ถ้าหากข้อมูลเป็นแบบสเตอริโอ nBlockAlign ก็จะมีค่าเท่ากับ 4 ไบต์ เป็นต้น

wBitsPerSample

ข้อมูลชนิดเลขจำนวนเต็ม กำหนดจำนวนบิตที่ใช้ในการสุ่มข้อมูล 1 ครั้ง เช่น 8 หมายถึง ใช้ 8 บิตในการสุ่มข้อมูลแต่ละครั้ง เป็นต้น

ส่วนในบล็อก subchunk 'data' หรือที่เรียกว่า data chunk นั้น ก็เป็นบล็อกที่จัดเก็บข้อมูลของสัญญาณเสียงในรูปแบบข้อมูลดิจิทัล ดังนั้นในการเล่นกลับไฟล์เสียง เราจะต้องทำการอ่านข้อมูลทั้งหมดในบล็อกนี้ เพื่อส่งให้กับอุปกรณ์ส่งสัญญาณเสียงต่อไป

### บทที่ 3

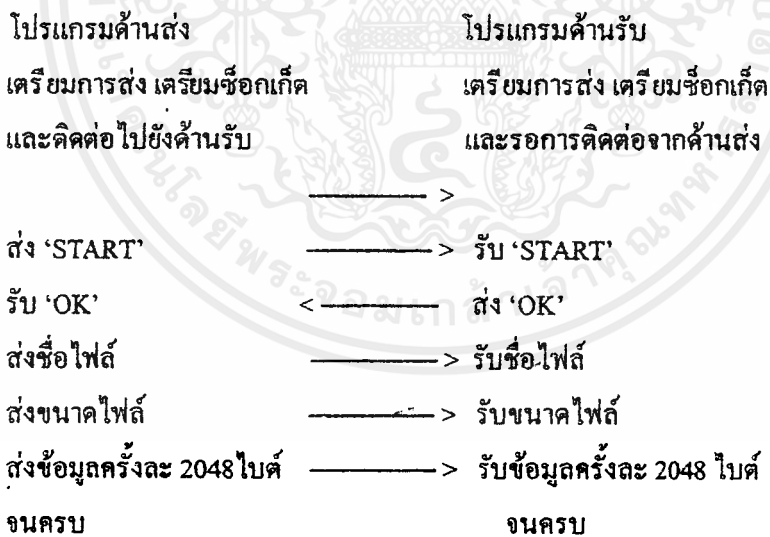
#### การคำนวณและการสร้าง

โปรแกรมที่สร้างขึ้นในโครงงานนี้สามารถแสดงบล็อกไดอะแกรมการรับส่งข้อมูลได้ดังรูป 3.1 สร้างขึ้นโดยใช้เดลฟี 2.0 (Delphi2.0) เป็นเครื่องมือในการสร้าง ซึ่งแต่ละโปรแกรมที่สร้างขึ้นโดยเดลฟีจะเรียกว่าโปรเจก ในโครงงานนี้ได้สร้างโปรเจกชื่อ 'MEDIA.DPR' ซึ่งเป็นโปรเจกหลักสำหรับเรียกใช้โปรเจกย่อยๆ ซึ่งแต่ละโปรเจกจะมีหน้าที่ต่างกัน โดยในบทนี้จะอธิบายลำดับขั้นตอนการทำงานของโปรเจกย่อยๆ ที่สำคัญ และเกี่ยวข้องกับโครงงานนี้ทั้งหมด พร้อมทั้งแสดงผังงานของแต่ละโปรเจกย่อย (Flowchart) เพื่ออธิบายการทำงานและแสดงโปรแกรมไว้ในภาคผนวก

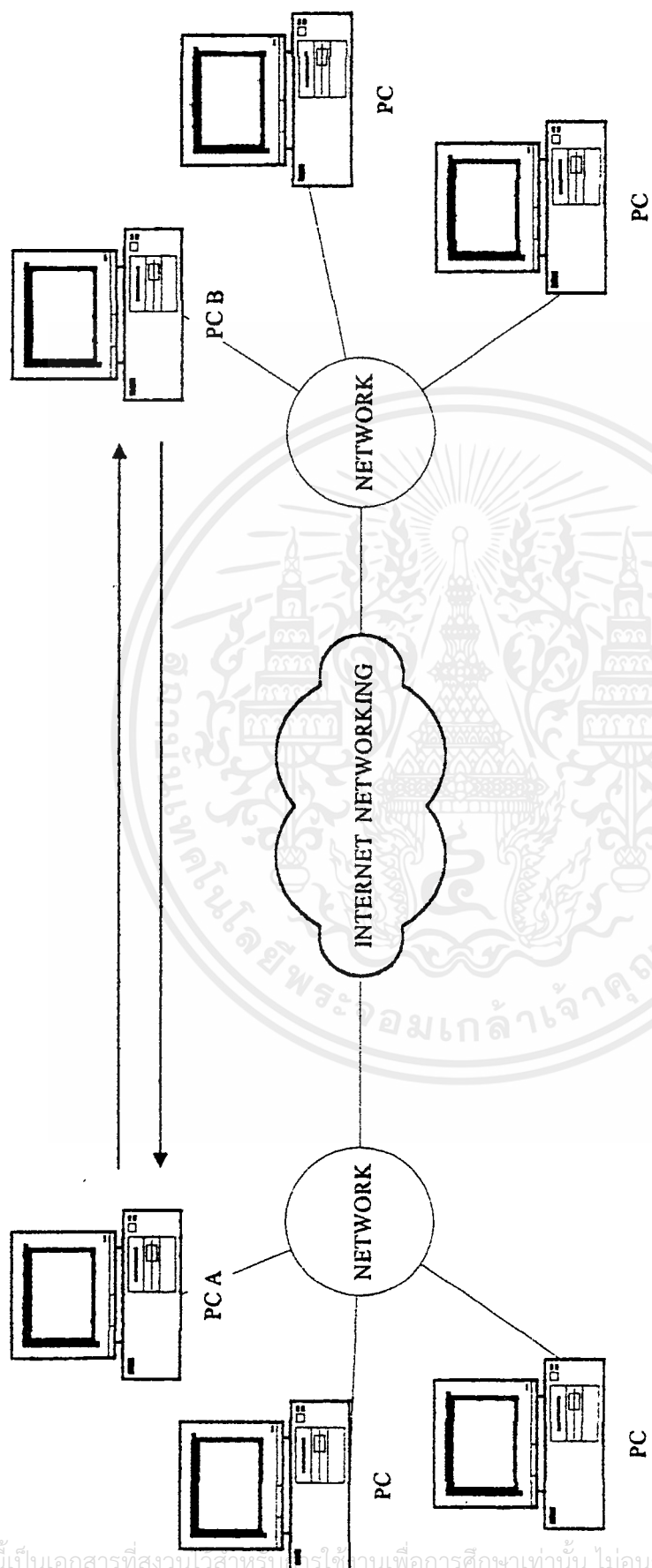
สำหรับโปรเจก 'MEDIA.DPR' (โปรแกรม 'MEDIA.PAS') เป็นโปรเจกหลักสำหรับรวมเอาโปรเจกย่อย ซึ่งมีหน้าที่การทำงานต่าง ๆ กันเข้าด้วยกันดังนี้

1. โปรเจก 'FSTRING' ทำหน้าที่รับส่งข้อมูลชนิดสตริงระหว่างด้านส่งและด้านรับ
2. โปรเจก 'SENDER' ทำหน้าที่ส่งไฟล์ข้อมูลชนิดต่างๆ จากด้านส่งไปยังด้านรับ
3. โปรเจก 'RECV' ทำหน้าที่รับไฟล์ข้อมูลชนิดต่างๆ ที่ส่งมาจากด้านส่ง

เนื่องจากการทำงานของโปรแกรมเกี่ยวข้องกับการรับส่งไฟล์ผ่านเครือข่ายอินเทอร์เน็ตจะต้องมีการโต้ตอบกันระหว่างทางด้านส่งและด้านรับ ดังนั้นจะอธิบายขั้นตอนการทำงานของโปรแกรมด้านส่งและโปรแกรมด้านรับควบคู่กันไป



รูป 3.2 แสดงการส่งสัญญาณโต้ตอบระหว่างโปรแกรมด้านส่งและด้านรับ



รูป 3.1 บทที่ 3.1 แสดงโปรแกรมแสดงโครงสร้างการรับข้อมูลระหว่างด้านรับของโครงการงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ในงานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

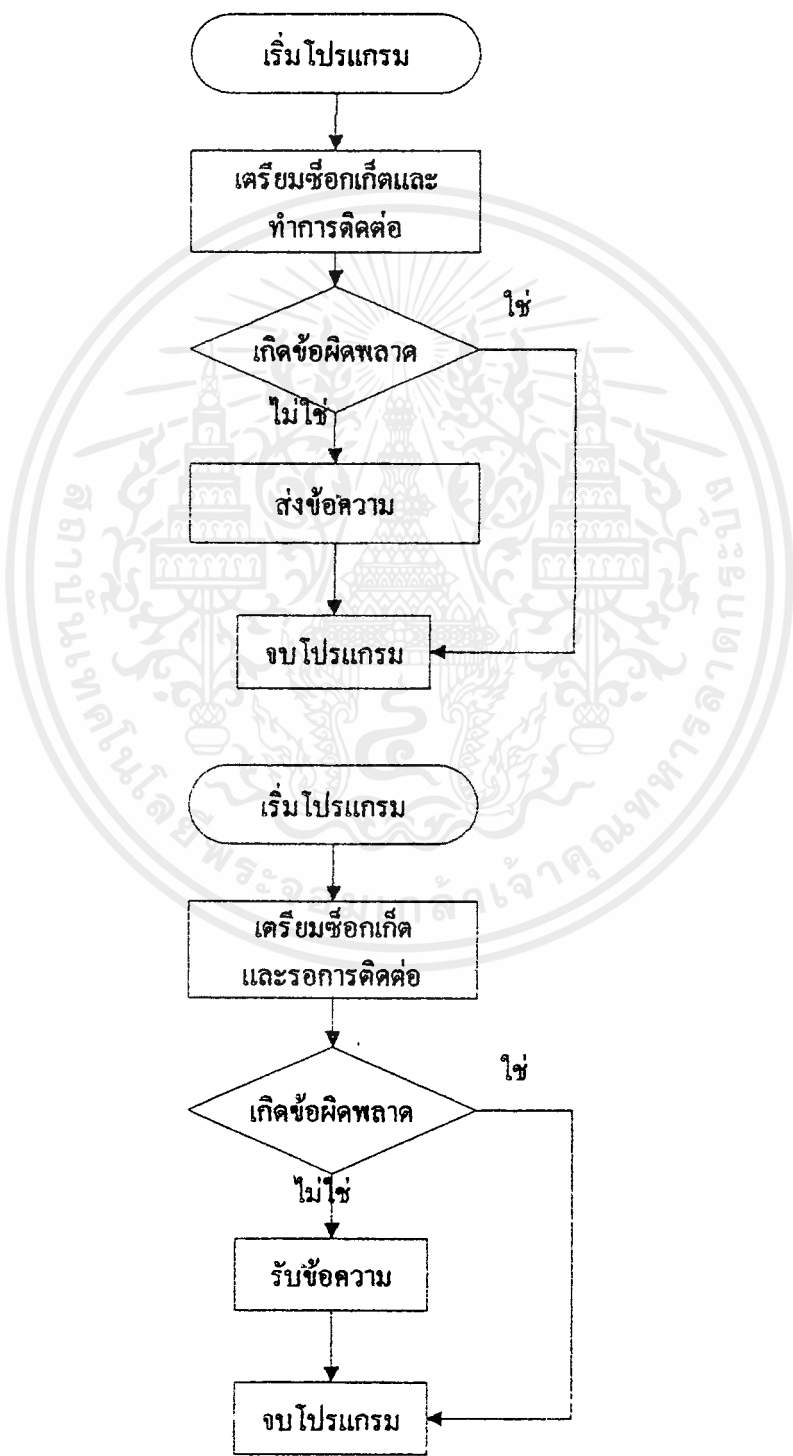
หลักการทำงานของโปรเจคต่างๆ โดยแยกตามลักษณะการทำงาน อธิบายได้ดังนี้

3.1 โปรแกรมพิมพ์ข้อความโต้ตอบ

ได้แก่ โปรเจค PFSTRING.DPR โปรแกรมชื่อ FSTRING.PAS

ทำหน้าที่รับส่งข้อความระหว่างเครื่องคอมพิวเตอร์สองเครื่อง โดยผู้ต้องการติดต่อจะทำการกำหนดหมายเลข ไอพีแอดเดรส เมื่อติดต่อได้แล้วจึงส่งข้อความไปที่เครื่องที่รอการติดต่อ

ส่วนเครื่องที่รอการติดต่อ จะทำการรอเพื่อรับข้อความที่ถูกส่งมา แผนภาพการทำงานแสดงดังรูป 3.3



### 3.2 โปรแกรมค้นส่งไฟล์

คือ โปรแกรม PSENDER.DPR โปรแกรมชื่อ SENDER.PAS

ทำหน้าที่ส่งไฟล์ข้อมูลต่างๆ เช่น ภาพ (\*.bmp, \*.jpg) เสียง (\*.wav) ออดิโอวิดีโอ (\*.avi) และไฟล์ข้อมูลอื่นๆ โดยมี พารามิเตอร์ที่ผู้ใช้ต้องป้อนเข้าไปได้แก่

1. หมายเลข IP ของเครื่องด้านรับปลายทาง
2. หมายเลขพอร์ตที่ใช้ในการติดต่อสื่อสาร
3. ชื่อไฟล์ที่ต้องการจะส่งโดยผู้ใช้อาจจะป้อนโดยตรง หรือเลือกชื่อไฟล์โดยใช้ปุ่ม Browse ก็ได้

หลังจากที่ผู้ใช้ป้อนพารามิเตอร์ต่างๆ เรียบร้อยแล้วก็เริ่มการส่งโดยกดปุ่ม Connect โปรแกรมจะเรียกโปรซีเจอร์ InitSend และ โปรซีเจอร์ BeginSendfile เพื่อทำการกำหนดพารามิเตอร์เบื้องต้นที่จำเป็น ได้แก่ โหมดการทำงานของซ็อกเก็ตเป็นแบบ non-blocking ซึ่งกำหนดเวลาสำหรับรอการติดต่อเป็น 30 วินาที จากนั้นโปรแกรมจะทำการติดต่อไปยังหมายเลข IP ที่ผู้ใช้ป้อนเข้าไป

เมื่อสามารถติดต่อกับเครื่องปลายทางได้ในเวลาที่กำหนด โปรแกรมจะเรียกใช้โปรซีเจอร์ Handshake เพื่อตรวจสอบกับเครื่องด้านรับปลายทาง ว่ารอรับการติดต่อ กับเครื่องด้านส่ง อยู่หรือไม่ โดยมีขั้นตอนการตรวจสอบดังนี้ คือ เครื่องด้านส่งจะส่งข้อความ 'START' แล้วรอรับข้อความ 'OK' ถ้าไม่มีข้อผิดพลาดใดๆเกิดขึ้น โปรแกรมส่งทางเครื่องด้านส่งจะสามารถเริ่มส่งข้อมูลได้ โดยแบ่งเป็น 3 กรณีคือ

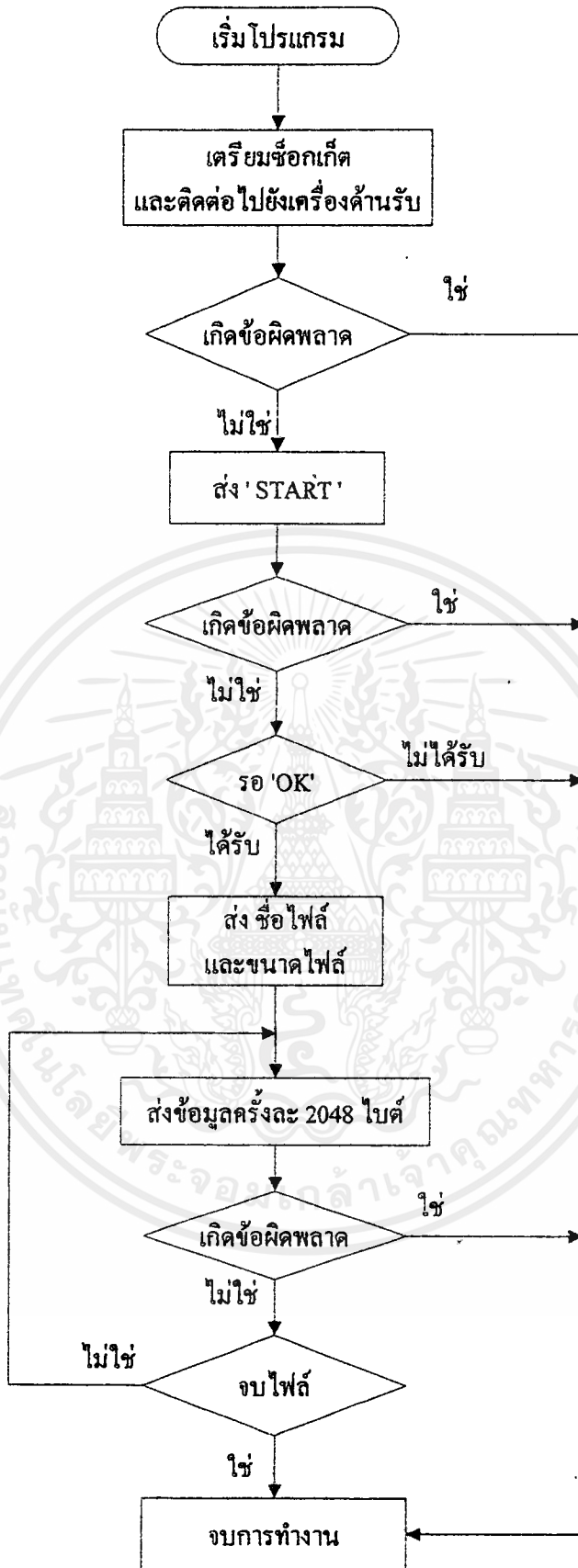
1. เลือกที่จะส่งไฟล์ข้อมูลเพียงไฟล์เดียว เช่น ไฟล์ภาพ (\*.bmp, \*.jpg) ไฟล์ออดิโอวิดีโอ (\*.avi) ไฟล์ข้อมูลแบบตัวอักษร (\*.txt)
2. เลือกที่จะส่งไฟล์ข้อมูลเสียงเพียงไฟล์เดียว
3. เลือกที่จะส่งข้อมูลทั้งภาพและเสียง คือ ไฟล์ภาพ (\*.bmp, \*.jpg) ไฟล์เสียง (\*.wav)

พิจารณากรณีที่ 1 การเลือกที่จะส่งไฟล์ข้อมูลเพียงไฟล์เดียว โปรแกรมจะเรียกโปรซีเจอร์ Sendonefile ในการส่ง เริ่มทำงานโดยการส่งชื่อไฟล์ ตามด้วยขนาดไฟล์ จากนั้นจะเป็นส่วนของข้อมูลจริงซึ่งจะส่งทีละ 2048 ไบต์จนจบไฟล์

พิจารณากรณีที่ 2 การเลือกส่งไฟล์ข้อมูลเสียงเพียงอย่างเดียวโดยสามารถเลือกไฟล์ข้อมูลเสียงที่จะส่งโดย Browse เฉพาะไฟล์เสียง หรือ ทำการอัดไฟล์เสียงโดยโปรแกรมจะไปเรียกโปรซีเจอร์ Recclick เพื่อทำการอัดไฟล์เสียงเพื่อที่จะส่ง จากนั้นใช้ปุ่ม Stop เพื่อเป็นการหยุดแล้วส่งข้อมูลทันทีโปรแกรมจะไปเรียกโปรซีเจอร์ Sendonefile โดยการส่งชื่อไฟล์ ขนาดไฟล์ จากนั้นทำการส่งข้อมูลเสียงทีละ 2048 ไบต์จนจบไฟล์

พิจารณากรณีที่ 3 การเลือกส่งข้อมูลทั้งภาพและเสียง โปรแกรมจะเรียกโปรซีเจอร์ Sendtwofile ในการส่ง เริ่มทำงานโดยการส่งชื่อไฟล์เสียง, ขนาดของไฟล์เสียง, ชื่อไฟล์ภาพ, ขนาดของไฟล์ภาพ จากนั้นจะเป็นส่วนของข้อมูลจริงซึ่งจะส่งทีละ 2048 ไบต์จนจบไฟล์

ทั้ง 3 กรณีนี้ผู้ส่งสามารถที่จะเลือกทำการส่งไฟล์ได้อีก โดยกดปุ่ม Browse เพื่อเลือกไฟล์ที่ต้องการจะส่งและทำการติดต่อใหม่ โดยป้อนพารามิเตอร์ต่างๆ แผนภาพการทำงานแสดงดังรูป 3.4



รูป 3.4 แผนภาพแสดงการทำงานของโปรแกรม Sender.pas

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

### 3.3 โปรแกรมด้านรับไฟล์

คือ โปรแกรม RECV.DPR โปรแกรม RECV.PAS

ผู้รับไฟล์สามารถกำหนดไครเทรทอรีสำหรับเก็บไฟล์ได้ โดยในที่นี้กำหนดให้เป็น 'C:\Multimedia' และกำหนดหมายเลขพอร์ตให้ตรงกับทางด้านส่ง

โปรแกรมจะเริ่มด้วยเรียกโปรซีเยอร์ Initreceive และ Beginlisten เพื่อกำหนดพารามิเตอร์ที่จำเป็น เช่น โหมดของการทำงานของซ็อกเก็ตเป็น non-blocking โดยไม่กำหนดช่วงเวลาในการคอย (Time out = 0) หมายถึงให้ซ็อกเก็ตรอการติดต่อจากผู้เรียกได้ไม่จำกัดเวลา จากนั้นเรียกใช้โปรซีเยอร์ Receive เพื่อรอการติดต่อจากผู้ใช้ เมื่อมีการติดต่อแล้วจะเรียกโปรซีเยอร์ Handshake เพื่อตรวจสอบว่าที่เครื่องด้านส่งใช้โปรแกรมเดียวกันหรือไม่ โดยรอรับข้อความ 'START' แล้วส่งข้อความ 'OK' กลับไปที่เครื่องด้านส่ง หลังจากนั้นจะเรียกโปรซีเยอร์ Getfile เพื่อรับข้อมูลโดยแบ่งเป็น 3 กรณีคือ

1. ไฟล์ที่ส่งมามีเพียงไฟล์เดียว
2. ไฟล์ที่ส่งมาเป็นไฟล์เสียง (สามารถ รับและแสดงผลข้อมูลทีละ 2048 ไบต์)
3. ไฟล์ที่ส่งมาเป็นไฟล์ข้อมูล 2 ไฟล์ คือ ข้อมูลภาพ และข้อมูลเสียง

พิจารณากรณีที่ 1 โปรแกรมจะเรียก โปรซีเยอร์ Getonefile เพื่อรับ ชื่อไฟล์ และขนาดของไฟล์ โดยโปรซีเยอร์ GetOneFile นี้จะทำการสร้างไฟล์ใหม่ ชื่อ FileTest0x.นามสกุลเดิม แล้วรับข้อมูลมาบันทึกทีละ 2048 ไบต์ จนสิ้นสุดการรับไฟล์

เมื่อรับไฟล์เรียบร้อยแล้ว โปรซีเยอร์ GetOneFile จะตรวจสอบว่า ข้อมูลที่ได้รับเป็น ไฟล์ข้อมูลชนิดใด เพื่อทำการแสดงผลแบ่งเป็น

- ไฟล์ ออดิโอวิดีโอ ' \*.avi ' โปรแกรมจะเรียก โปรซีเยอร์ Play เพื่อทำการเล่นไฟล์ข้อมูลนี้
- ไฟล์ข้อมูลภาพ ' \*.bmp หรือ \*.jpg ' โปรแกรมจะเรียกโปรซีเยอร์ Showpicture เพื่อทำการแสดงภาพบนหน้าจอ
- ไฟล์ข้อมูลแบบตัวอักษร ' \*.txt ' โปรแกรมจะเรียก โปรซีเยอร์ Showclick เพื่อแสดงผล

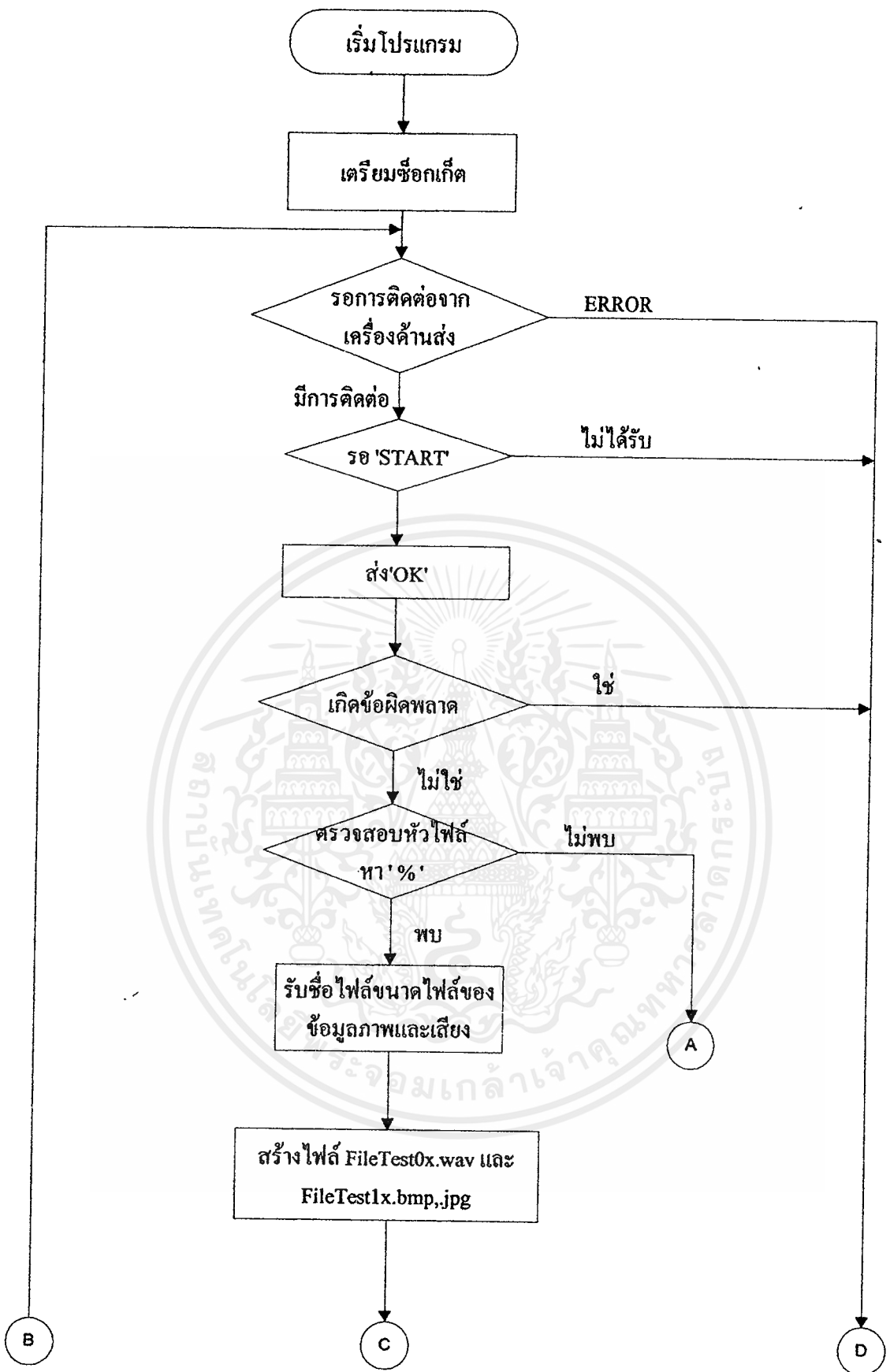
หากเป็นไฟล์ข้อมูลอื่นๆ ที่นอกเหนือจากนี้จะไม่แสดงผล

พิจารณากรณีที่ 2 โปรแกรมจะเรียกโปรซีเยอร์ GetoneSound เพื่อรับชื่อไฟล์ และขนาดของไฟล์ โดยโปรซีเยอร์ GetOneSound นี้จะรับข้อมูลมาบันทึกทีละ 2048 ไบต์และทำการสร้างไฟล์ใหม่ชื่อ 'c:\temp1.wav' และไฟล์ 'c:\temp2.wav' เพื่อผลัดกันรับข้อมูล และผลัดกันแสดงผล พร้อมทั้งสร้างไฟล์ชื่อ File0x.ไฟล์เพื่อบันทึกข้อมูลของไฟล์ที่รับมาได้ทั้งหมด ( ไฟล์ข้อมูลเสียงที่เหมือนทางด้านส่งทุกประการ )

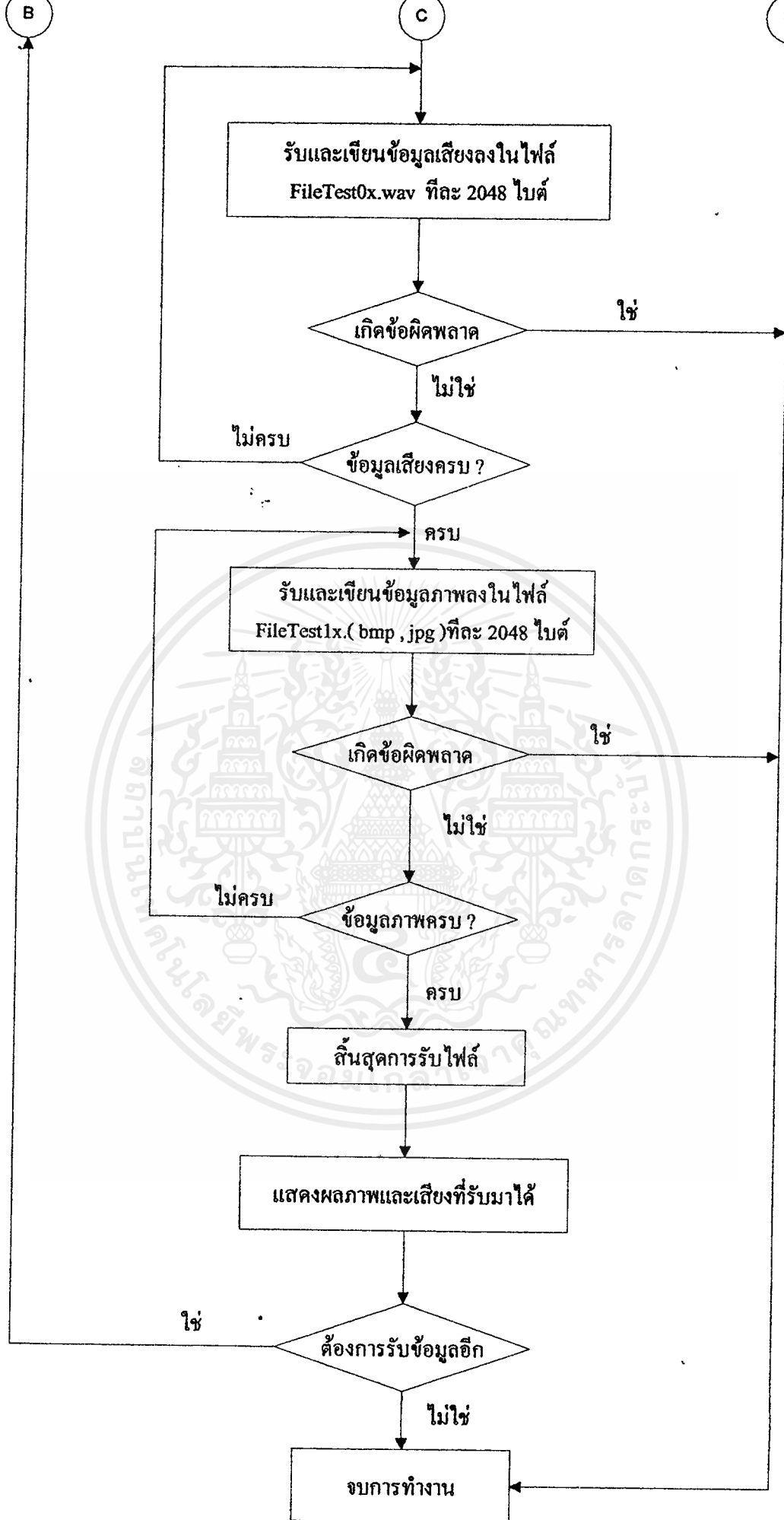
พิจารณากรณีที่ 3 โปรแกรมจะเรียกโปรซีเยอร์ GetTwoFile เพื่อรับ ชื่อไฟล์ และขนาดของไฟล์ โดยโปรซีเยอร์ GetTwoFile นี้จะทำการสร้างไฟล์ใหม่เพื่อรับข้อมูลเสียงชื่อ FileTest0x.wav และสร้างไฟล์ FileTest1x.( .bmp, .jpg ) เพื่อรับข้อมูลภาพ ต่อมาจะทำการรับข้อมูลมาบันทึกทีละ 2048 ไบต์ จนสิ้นสุดการรับไฟล์เมื่อรับไฟล์เรียบร้อยแล้ว โปรแกรมจะทำการแสดงผลข้อมูลภาพและเสียงพร้อมกัน

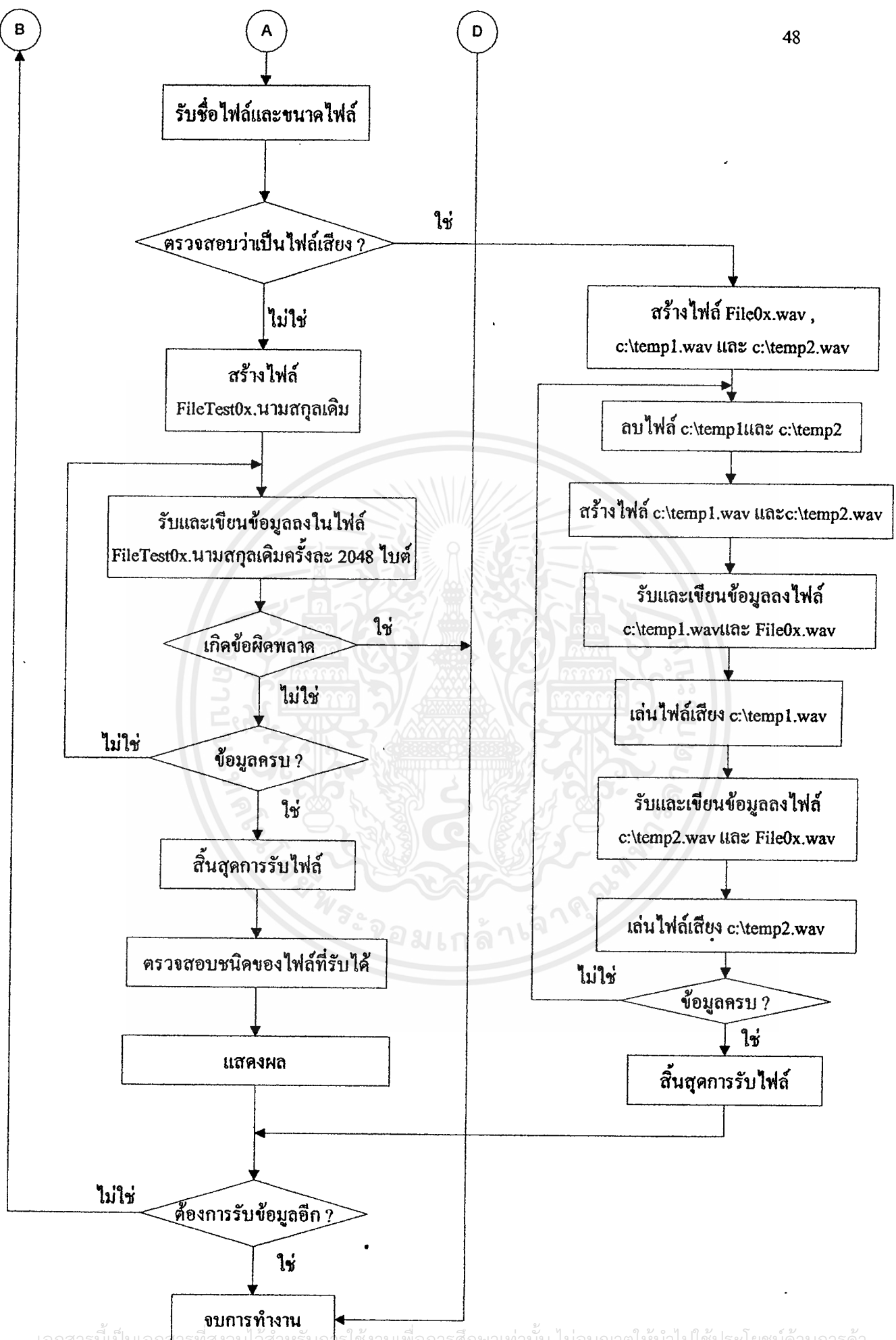
หลังจากทำการรับไฟล์เรียบร้อยแล้วจะมีการถามว่าจะทำการรอรับไฟล์ข้อมูลอีกหรือไม่ ถ้ารอรับก็จะเริ่มทำการรอรับขนาดไฟล์ ชื่อไฟล์และข้อมูลอีกครั้ง ถ้าไม่รอรับเป็นอันสิ้นสุดการทำงาน แผนภาพการทำงานของโปรแกรมแสดงดังรูป 3.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.5 A แผนผังแสดงการทำงานของโปรแกรม RECV.pas



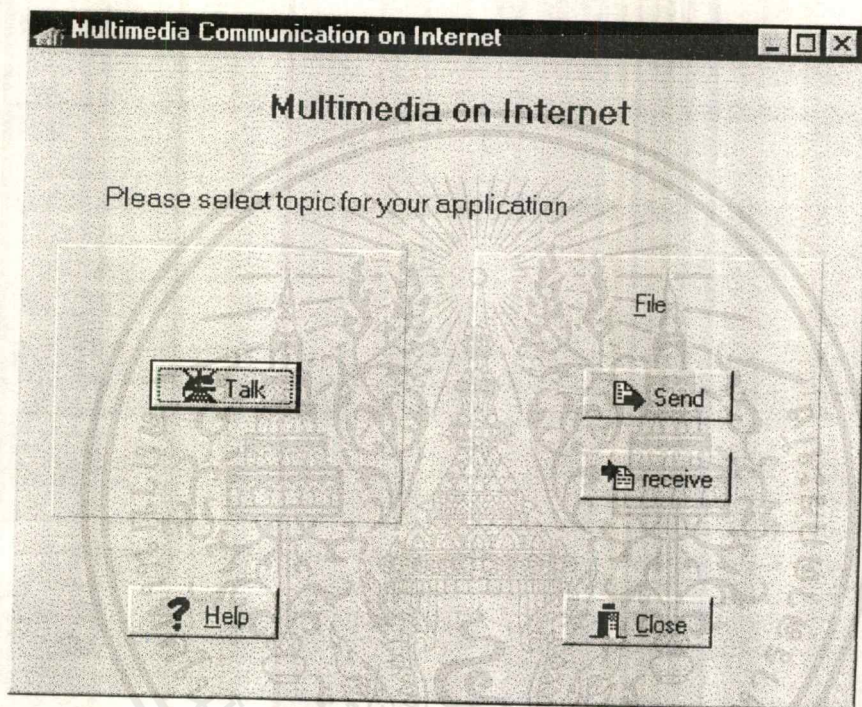


## บทที่ 4

### การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึงการทดลองรับ-ส่งข้อมูลเป็นขั้นตอน โดยแบ่งเป็นการรับส่งข้อมูลเป็นไฟล์ และการรับส่งข้อมูลเป็นสตรีม ขั้นตอนต่างๆอธิบายได้ดังนี้

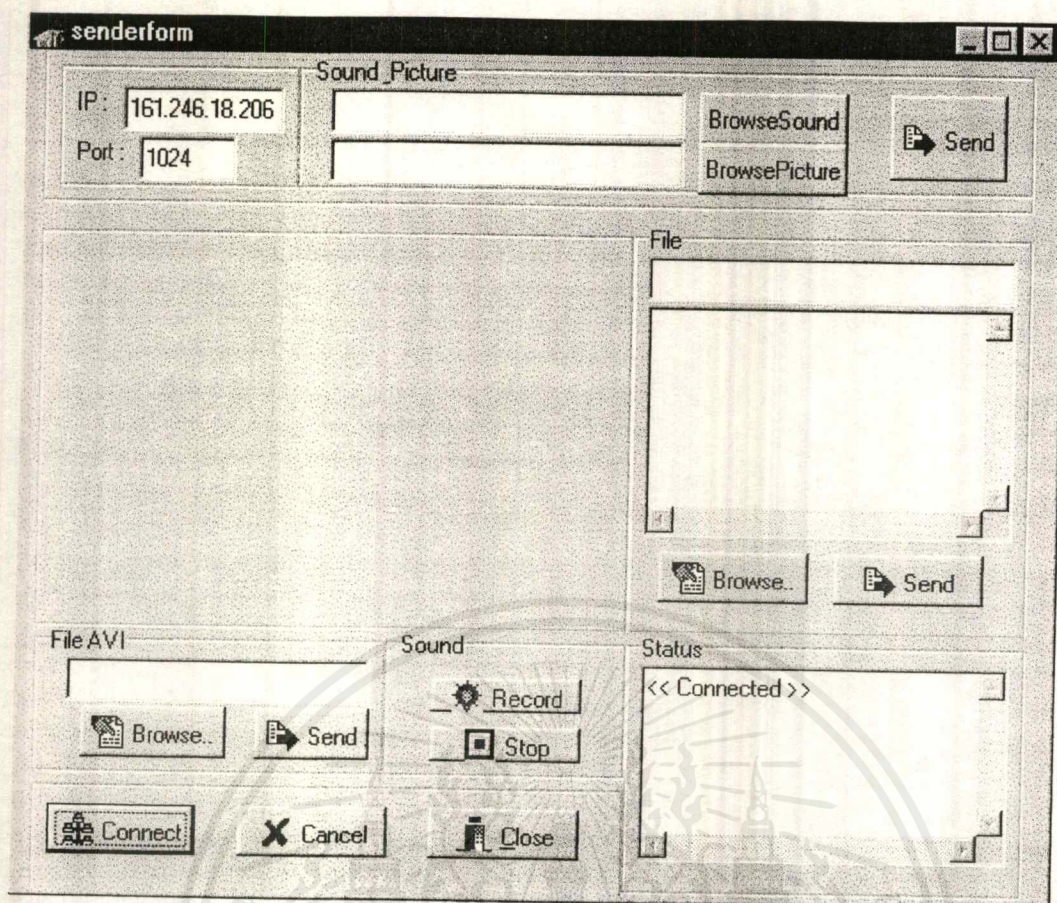
เมื่อทำการรันโปรแกรม Pmedia.exe จะปรากฏฟอร์มหลัก( Multimedia Communication on Internet ) ดังแสดงในรูป 4.1



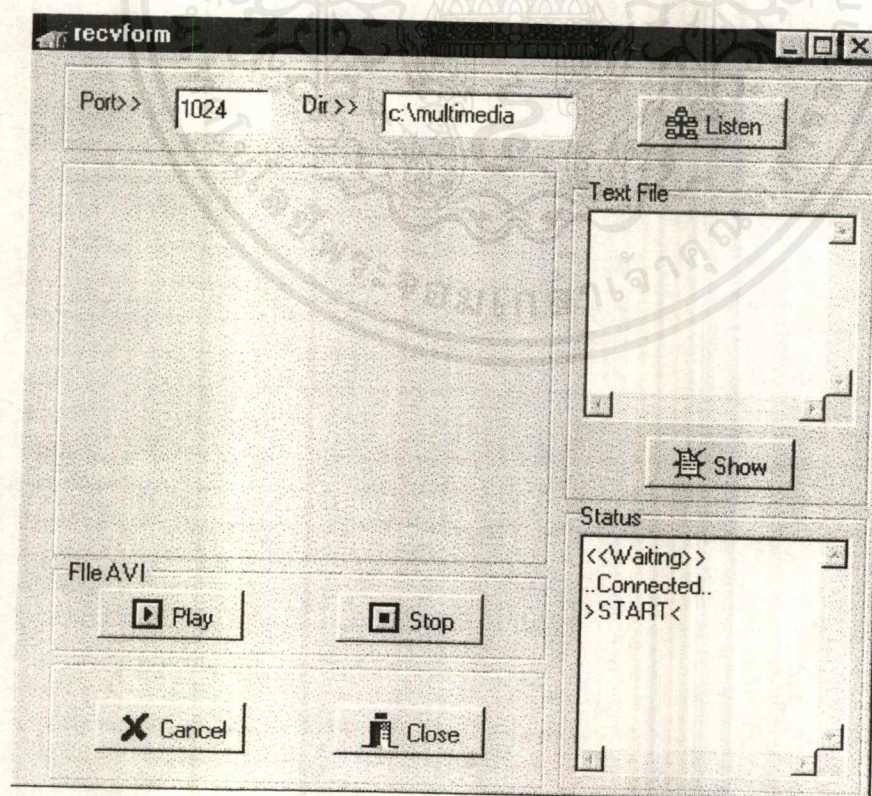
รูป 4.1 แสดงฟอร์มของโปรเจค Pmedia.dpr

#### 4.1 การทดลองรับ-ส่ง ข้อมูลระหว่างเครื่องคอมพิวเตอร์บนเครือข่ายอินเทอร์เน็ต

ทางเครื่องด้านส่งจะกดปุ่ม Sender เพื่อเรียกโปรเจค Psender.dpr จะปรากฏฟอร์มSenderform ผู้ใช้ต้องกำหนดหมายเลขไอพีแอดเดรสของเครื่องที่ต้องการติดต่อ ในการทดลองนี้ใช้เครื่องที่มีหมายเลขไอพีแอดเดรส เป็น 161.246.18.206 เป็นเครื่องที่ต้องการติดต่อ และใส่หมายเลขพอร์ตที่ต้องการใช้ติดต่อสื่อสาร จากนั้นจะกดปุ่ม Connect เพื่อทำการเชื่อมต่อกับเครื่องปลายทางที่กำลังรอการติดต่อ ดังแสดงในรูป 4.2



รูป 4.2 แสดงฟอร์มของโปรเจก Psender.dpr ขณะทำการติดต่อกับปลายทางเพื่อส่งข้อมูล



รูป 4.3 แสดงฟอร์มโปรเจก Precv .dpr เมื่อมีการติดต่อและพร้อมรับข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

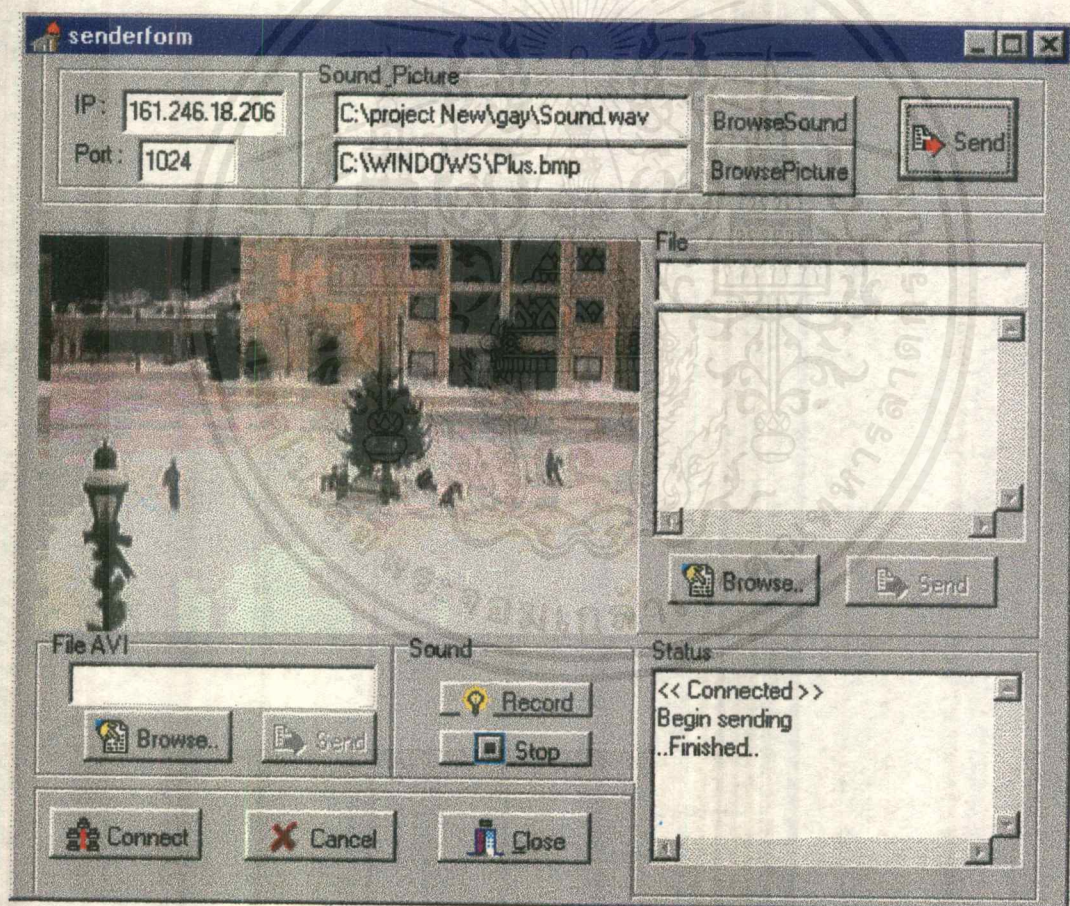
ทางเครื่องด้านรับจะต้องมีการเปิดโปรแกรมเพื่อรอการติดต่อ โดยจากหน้าฟอร์มหลักผู้รับจะกดปุ่ม Receive โปรแกรมจะไปเรียกโปรเจก Precv.dpr ในการทดลองนี้ไฟล์ที่รับได้จะนำมาเก็บไว้ที่ ไดรฟ์ที่ C:\Multimedia ผู้รับ ต้องเลือกปุ่ม Listen เพื่อรอการติดต่อจากเครื่องด้านส่ง

เมื่อเครื่องด้านส่งทำการติดต่อมา เครื่องด้านรับจะแสดงสถานะว่ามีการติดต่อ และเมื่อทั้งสองด้านพร้อมแล้ว เครื่องด้านรับจะแสดงสถานะว่าพร้อมที่จะรับข้อมูล แสดงดังรูป 4.3

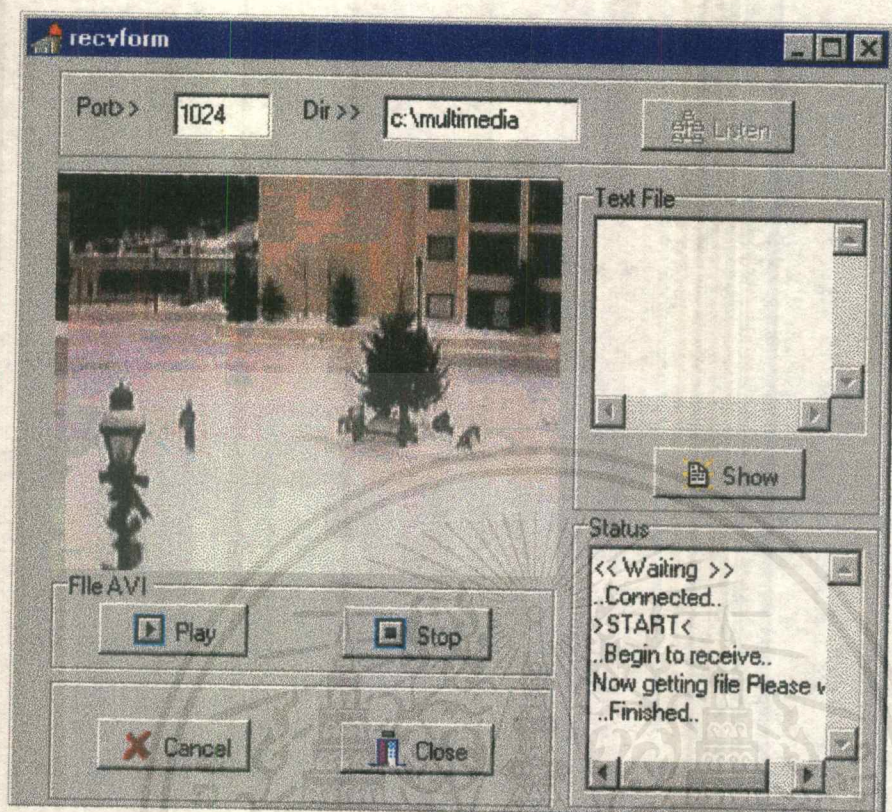
จากนั้นจะทำการเลือกชื่อไฟล์ที่ต้องการส่งโดยกดปุ่ม Browse เพื่อเลือกไฟล์ที่ต้องการจะส่งหรือกำหนดชื่อไฟล์โดยตรงก็ได้ โดยสามารถที่จะ Browse ส่งไฟล์เสียงและไฟล์ภาพไปพร้อมกัน ซึ่งในการทดลองนี้ทำการส่งไฟล์ชื่อ Sound.wav และ Plus.bmp แสดงผลในรูป 4.4

หากไฟล์ที่ส่งมาเป็นไฟล์เสียง โปรแกรมจะทำการเล่นไฟล์เสียง 1 ครั้ง โดยสามารถเล่นอีกครั้งได้โดยกดปุ่ม Play

และรูป 4.5 แสดงผลของเครื่องทางด้านรับ เมื่อรับไฟล์เรียบร้อยแล้ว

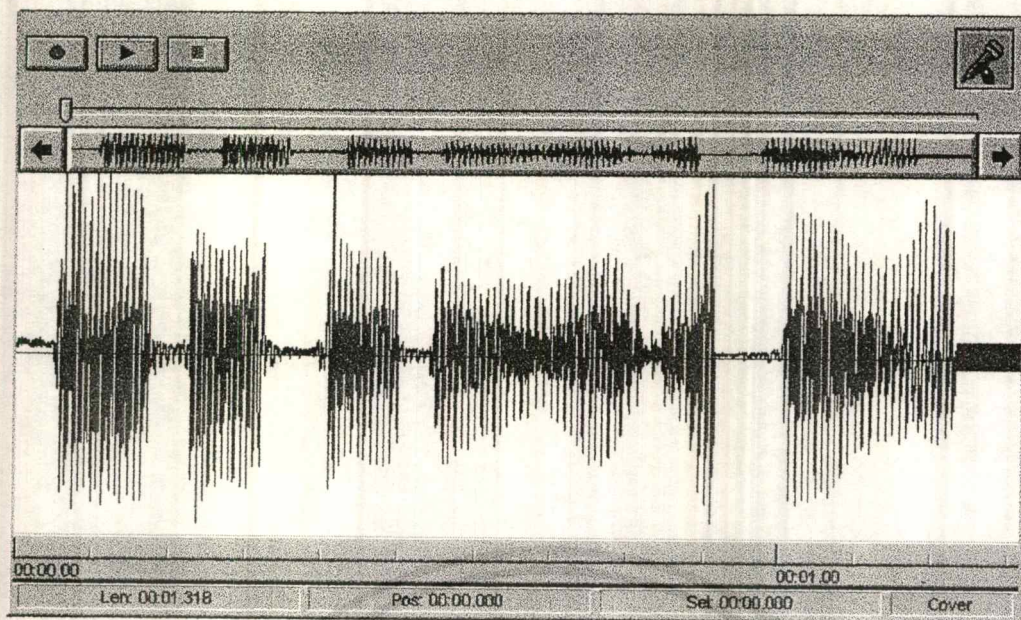


รูป 4.4 แสดงหน้าฟอร์มของโปรเจก Psender.dpr เมื่อทำการส่งข้อมูลภาพและเสียงเรียบร้อยแล้ว

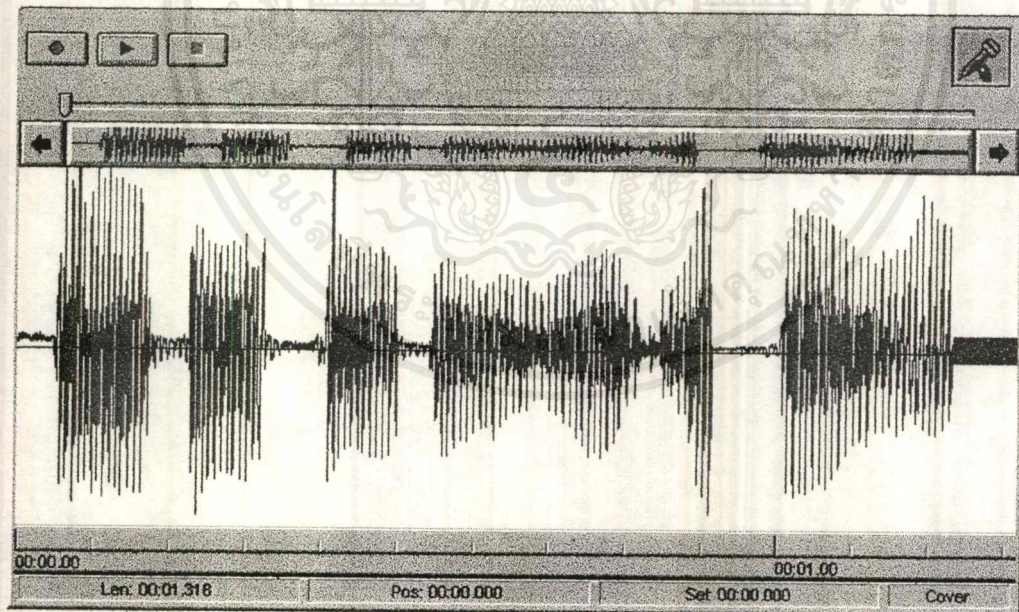


รูป 4.5 แสดงฟอร์มโปรเจก Precv.dpr เมื่อทำการรับข้อมูลภาพและเสียง พร้อมทั้งแสดงข้อมูลที่รับมาได้

จากการทดลองส่งไฟล์ภาพและไฟล์เสียง สามารถแสดงผลการเปรียบเทียบระหว่างไฟล์เสียงด้านส่งกับไฟล์เสียงด้านรับได้ดังรูป 4.6 และ 4.7



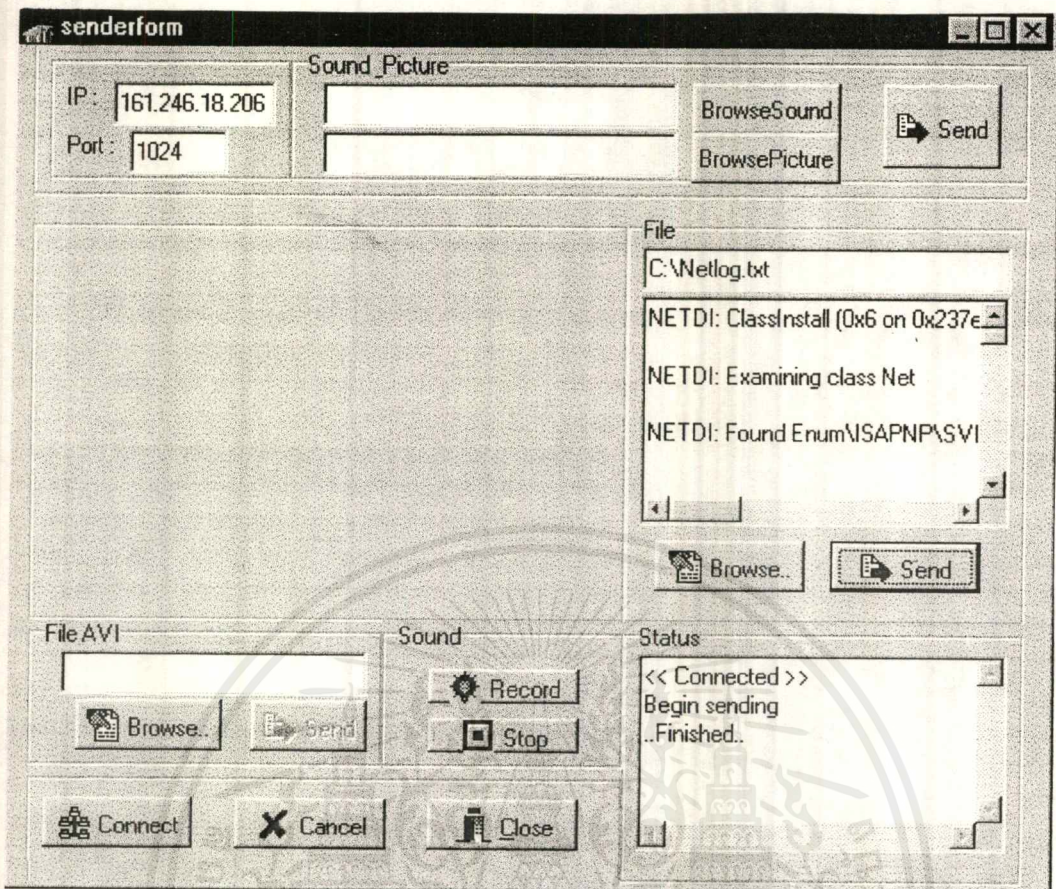
รูป 4.6 แสดงกราฟเสียงของไฟล์เสียงที่ส่งจากทางด้านส่ง



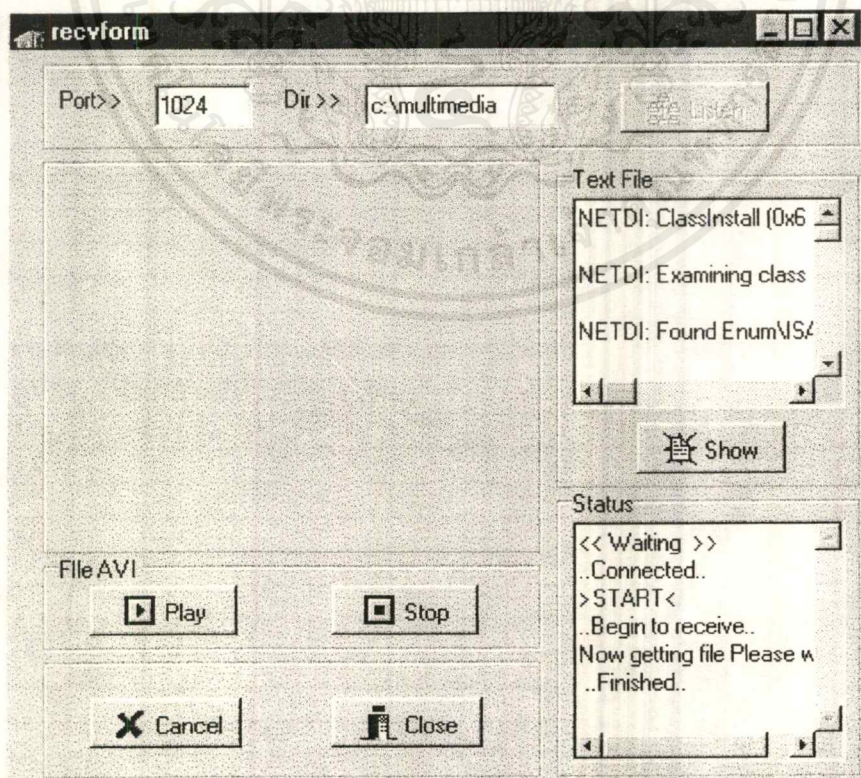
รูป 4.7 แสดงกราฟเสียงของไฟล์เสียงที่รับได้ทางด้านรับ

จากนั้นทำการทดลองเลือกส่งไฟล์แบบตัวอักษร โดยการทดลองนี้ทำการส่งไฟล์ชื่อ Netlog.txt ดังรูป 4.8 แสดงผลทางด้านเครื่องส่ง และรูป 4.9 แสดงผลทางด้านเครื่องรับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.8 แสดงหน้าฟอร์มของโปรเจก Psender.dpr เมื่อทำการส่งไฟล์ข้อมูลแบบ Text เรียบร้อยแล้ว

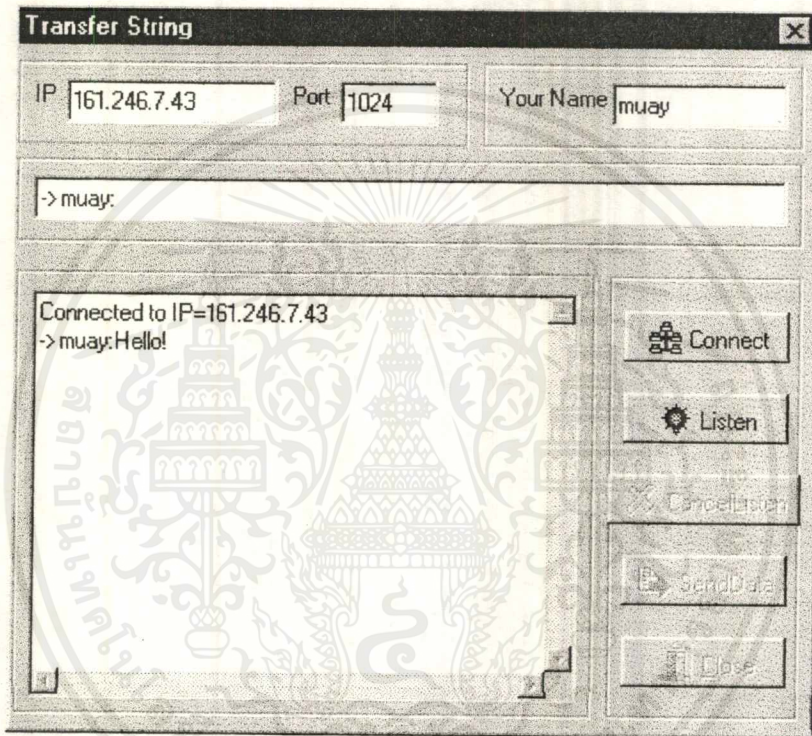


รูป 4.9 แสดงฟอร์ม Precv.dpr เมื่อทำการรับไฟล์ข้อมูลแบบ Text เรียบร้อยแล้วพร้อมทั้งแสดงข้อมูลเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

#### 4.2 การทดลองรับส่งข้อความโต้ตอบระหว่างเครื่องคอมพิวเตอร์สองเครื่อง

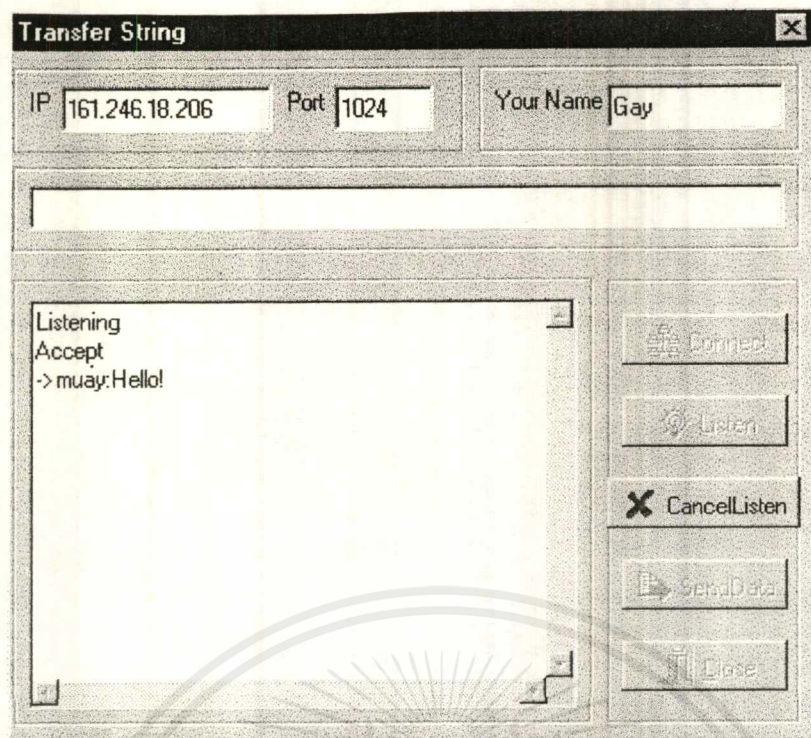
จากฟอร์มหลักในรูป 4.1 เลือกปุ่ม String โปรแกรมจะไปเรียกโปรเจค Pstring.dpr จะปรากฏฟอร์ม Transfer String

ทางด้าน ผู้รอการติดต่อจะต้องรอการติดต่อ โดยการกดปุ่ม Listen ซึ่งสามารถยกเลิกการรอคอยได้โดยการกดปุ่ม Cancel Listen ส่วนผู้ที่ต้องการติดต่อจะทำการกำหนดค่า ไอพีแอดเดรส แล้วจึงกดปุ่ม Connect เพื่อทำการติดต่อ จากนั้นจึงทำการพิมพ์ข้อความที่ต้องการติดต่อแล้วทำการกดปุ่ม Send data เพื่อทำการส่งข้อความ ซึ่งแสดงการส่งข้อความแล้ว ดังรูป 4.10



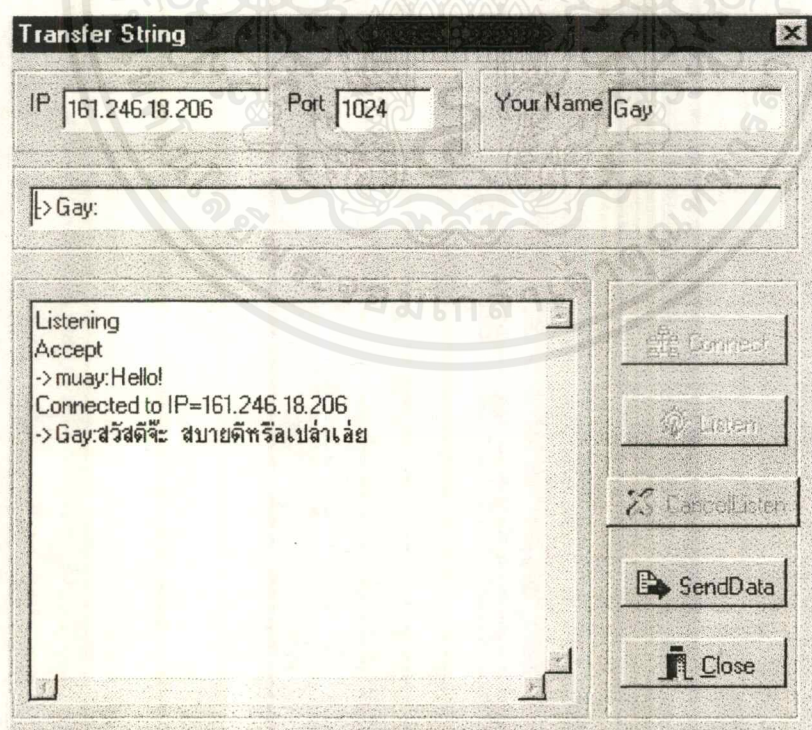
รูป 4.10 แสดงผลของฟอร์ม Pstring.dpr หลังจากส่งข้อมูลแล้ว

และจะแสดงสถานะทางด้านผู้รอการติดต่อขณะรับข้อมูลเรียบร้อยแล้ว ดังรูป 4.11



รูป 4.11 แสดงผลของฟอร์ม Pfstring.dpr หลังจากรับข้อมูลแล้ว

หลังจากนั้นเครื่องที่รอการติดต่อจะสามารถเป็นผู้ส่งข้อความได้ในทำนองเดียวกัน แสดงดังรูป 4.12



รูป 4.12 แสดงผลของฟอร์ม Pfstring.dpr ขณะทำการรับ-ส่ง ข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## บทที่ 5

### บทวิจารณ์และบทสรุป

#### สรุปผลการทดลอง

จากการทดลองสามารถรับส่งข้อความโต้ตอบกันได้ และ ทำการรับส่งข้อมูลภาพและเสียง โดยเฉพาะข้อมูลเสียงสามารถแสดงผลได้ทันทีที่รับข้อมูลได้ รวมถึงสามารถส่งข้อมูลต่างๆ เช่น ข้อมูลแบบอักษร ข้อมูลออกดีไอวีดีไอ และแสดงผลของข้อมูลที่ได้รับมาได้อย่างถูกต้อง

#### ปัญหาที่พบจากการทดลอง

1. ในการรับส่งไฟล์ ที่เครื่องด้านรับต้องมีการเปิดโปรแกรม เพื่อรอการติดต่อก่อน เครื่องด้านส่งจึงจะทำการติดต่อมาได้
2. ในขณะที่ด้านรับรอคอยการติดต่อจะไม่สามารถใช้โปรแกรมอื่นได้

#### แนวทางในการพัฒนา

1. ปรับปรุงโปรแกรมเพื่อแก้ไขปัญหาที่เกิดขึ้น
2. พัฒนาโปรแกรมให้สามารถติดต่อสื่อสารกันได้อย่างสะดวกและรวดเร็วยิ่งขึ้น



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
unit Sender;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, Buttons, Sockets, Winsock, MPlayer, ExtCtrls, Jpgimg, Gauges,
ComCtrls;
```

```
type
```

```
Tsenderform = class(TForm)
```

```
Sockets1: TSockets;
```

```
GroupBox1: TGroupBox;
```

```
GroupBox2: TGroupBox;
```

```
Label1: TLabel;
```

```
EditIP: TEdit;
```

```
port: TLabel;
```

```
Editport: TEdit;
```

```
GroupBox3: TGroupBox;
```

```
EditSound: TEdit;
```

```
Editpicture: TEdit;
```

```
BrowseSButton: TButton;
```

```
BrowsePButton: TButton;
```

```
GroupBox6: TGroupBox;
```

```
TExtfile: TGroupBox;
```

```
Memo1: TMemo;
```

```
GroupBox8: TGroupBox;
```

```
EditAVI: TEdit;
```

```
SendAVI: TBitBtn;
```

```
BrowseAVI: TBitBtn;
```

```
EditText: TEdit;
```

```
SendText: TBitBtn;
```

```
BrowseText: TBitBtn;
```

```
GroupBox4: TGroupBox;
```

```
JPEGImage1: TJPEGImage;
```

```
CloseBitBtn: TBitBtn;
```

```
CancelBitBtn: TBitBtn;
```

```
GroupBox5: TGroupBox;
```

```
Memo2: TMemo;
```

```
GroupBox7: TGroupBox;
```

```
MediaPlayer1: TMediaPlayer;
```

```
Rec: TBitBtn;
```

```
stop: TBitBtn;
```

```
connect: TBitBtn;
```

```
SendSP: TBitBtn;
```

```
OpenDialog1: TOpenDialog;
```

```
procedure SendSPClick(Sender: TObject);
```

```
procedure FormCreate(Sender: TObject);
```

```
procedure Sockets1ErrorOccurred(Sender: TObject; Socket,
```

```
Error: Integer; Msg: string);
```

```
procedure Sockets1SessionClosed(Sender: TObject; Socket: Integer);
```

```
procedure CancelBitBtnClick(Sender: TObject);
```

```
procedure BrowseSButtonClick(Sender: TObject);
```

```
procedure BrowsePButtonClick(Sender: TObject);
```

```
procedure CloseBitBtnClick(Sender: TObject);
```

```
procedure SendAVIClick(Sender: TObject);
```

```
procedure BrowseAVIClick(Sender: TObject);
```

```
procedure StopAVIClick(Sender: TObject);
```

```
procedure ConnectClick(Sender: TObject);
```

```
procedure BrowseTextClick(Sender: TObject);
```

```
procedure SendTextClick(Sender: TObject);
```

```
procedure RecClick(Sender: TObject);
```

```
procedure stopClick(Sender: TObject);
```

```
private
```

```
{ Private declarations }
```

```
procedure InitSend;
```

```
procedure SendOnefile;
```

```
procedure SendTwofile;
```

```
procedure DoSend;
```

```
procedure CloseSocket;
```

```
procedure Handshake;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

function SocketsError:boolean;
public
{ Public declarations }
TimedOut: Boolean;
ErrorReturn :Integer;
end;

var
senderform: Tsenderform;

implementation

{$R *.DFM}

procedure Tsenderform.SendSPClick(Sender: TObject);
begin
if(EditSound.text <> '') and (EditPicture.text <> '') then SendTwoFile ;
if(EditSound.text <> '') and (EditPicture.text = '') then SendOneFile;
if(EditSound.text = '') and (EditPicture.text <> '') then SendOneFile;
if(EditSound.text = '') and (EditPicture.text = '') then
 messageDlg('No selected file '+#13+ ' Please Select
again',mtinformation,[mbOK],0);
Sockets1.Getpeerport(sockets1.socketNumber);
Sockets1.GetpeerIPAddr(sockets1.socketNumber);
InitSend;
end;

procedure Tsenderform.sendTwofile ;
var
f1,f2:file;
numreadSound,numreadPicture: integer;
fsizeSound,fsizePicture,bytewrite1,bytewrite2,total:longint;
filetosendS,filetosendP:string;
szbuf1:array[0..2047] of char;
szbuf2:array[0..2047] of char;
begin
Memo2.lines.add('Begin sending');
filetosendS := EditSound.text;
filetosendP := EditPicture.text;
assignfile (f1,filetosendS);
assignfile (f2,filetosendP);
{$I-}
reset(f1,1);
{$I+}
if IOresult<> 0 then
begin
Memo2.lines.add('..Can not open fileSound..');
closesocket;
exit;
end;
{$I-}
reset(f2,1);
{$I+}
if IOresult<> 0 then
begin
Memo2.lines.add('..Can not open filepicture..');
closesocket;
exit;
end;

fsizeSound:= filesize(f1);
fsizePicture:= filesize(f2);
bytewrite1:= 0;
bytewrite2:= 0;
total := fsizeSound + fsizePicture;
sockets1.text := '%('+filetosendS+')'+ '<' +inttostr(fsizeSound)+'>'
+ '('+filetosendP+')'+ '<<' +inttostr(fsizePicture)+'>>';
repeat
blockread(f1,szbuf1, sizeof(szbuf1),numreadSound);
sockets1.send(sockets1.socketnumber,szbuf1,numreadSound);
bytewrite1 := bytewrite1+numreadSound;
until (bytewrite1 >= fsizeSound) or socketerror;

```

เอกสารนี้เป็นเอกสารสงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

if socketerror then
begin
Memo2.lines.add('Error while sending Sound');
closesocket;
end;
closefile(f1);
repeat
blockread(f2,szbuf2, sizeof(szbuf2),numreadPicture);
sockets1.ssend(sockets1.socketnumber,szbuf2,numreadPicture);
bytewrite2 := bytewrite2+numreadPicture;
until (bytewrite2 >= fsizePicture) or socketerror;
if socketerror then
begin
Memo2.lines.add('Error while sending PICTURE');
closesocket;
end;
closefile(f2);
Memo2.lines.add('..Finished..');

end;
procedure Tsenderform.FormCreate(Sender: TObject);
var head : array[0..60] of char;
 nwrite: integer;
 f : file ;
begin
head := 'RIFF'+#$33+$0+$00+$00+'WAVEfmt '+
#$12+$00+$00+$00+$01+$00+$01+$00+
#$22+$56+$00+$00+$22+$56+$00+$00+
#01#00+$08+$00+$38+$00+$66+$61+ #$63+$74+$04+$00+$00+$00+$00+$00+
#00+$00+$64+$61+$74+$61+$00+$00+ #00+$00 ;
assignfile(f,'c:\muay.wav');
{$I-} rewrite(f,1);{$I+}
blockwrite(f,head,sizeof(head),nwrite);
closefile(f);
EditIP.Text:= '161.246.18.206';
SendSP.Enabled:= True ;
CancelBitBtn.Enabled:= True;
CloseBitBtn.Enabled:= True;
MediaPlayer1.filename := 'Muay.wav';
Memo1.lines.clear;
Memo2.lines.clear;
MediaPlayer1.open;
end;

procedure Tsenderform.InitSend;
begin
BrowseSButton.enabled := True;
BrowsePButton.enabled := True;
BrowseText.enabled := True;
BrowseAVI.enabled := True;
TimedOut := False;
ErrorReturn := 0;
end;

procedure Tsenderform.Sockets1ErrorOccurred(Sender: TObject; Socket,
Error: Integer; Msg: string);
begin
ErrorReturn := Error;
if Error =WSAETimedout then
begin
TimedOut:= True;
ErrorReturn :=0;
end;
Messagedlg('@'+inttostr(error)+' ':'+msg, mterror,[mbok],0);
if Sockets1.SocketNumber<>INVALID_SOCKET then
begin
Memo2.lines.add('<< Close socket>> ');
CloseSocket;
end;
end;

```

function Tsenderform.SocketSErrror: Boolean;

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

begin
 Result := false;
 if Timedout then
 begin
 Result := true;
 Errorreturn := 0;
 Messagedlg('Time out error' ,mterror,[mbok],0);
 end;
 if (Errorreturn <> 0) then
 begin
 Result := true;
 Errorreturn := 0;
 Messagedlg('socket error' ,mterror,[mbok],0);
 end;
end;

procedure Tsenderform.Closesocket;
begin
 if Sockets1.Socketnumber <> INVALID_SOCKET then
 begin
 Sockets1.SClose;
 BrowseSButton.enabled := True;
 BrowsePButton.enabled := True;
 BrowseText.enabled := True;
 BrowseAVI.enabled := True;
 end;
end;

procedure Tsenderform.Sockets1SessionClosed(Sender: TObject; Socket: Integer);
begin
 Memo2.lines.add('<< socket Closed>> ');
 CloseSocket;
end;

procedure Tsenderform.DoSend;
begin
 Memo2.lines.add('<< Connected >>');
 SendSP.enabled:=True;
 CancelBitBtn.enabled:=true;
 BrowseSButton.enabled:=true;
 BrowsePButton.enabled:=true;
 CloseBitBtn.enabled:= true;
 Handshake;
 if SocketsError then
 begin
 Memo2.lines.add('>>socket error<<');
 closesocket;
 exit;
 end;
end;

end;

procedure Tsenderform.HandShake;
var
 str: string;
begin
 sockets1.text:= 'START';
 if SocketsError then
 begin
 Memo2.lines.add('..ERROR..');
 exit;
 end;
 str:= sockets1.text;
 if SocketsError then
 begin
 Memo2.lines.add('..error..') ;
 exit;
 end;
 if pos('OK',str) =0 then
 begin

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Memo2.lines.add('..Invalid HEADER..');
ErrorReturn:= -1;
exit;
end;
end;

```

```

procedure Tsenderform.CancelBitBtnClick(Sender: TObject);
begin
if sockets1.socketnumber<>INVALID_SOCKET then
sockets1.sclose;
Memo2.lines.add('Sender stop');
InitSend;
close;
end;

```

```

procedure Tsenderform.BrowseSButtonClick(Sender: TObject);
begin
BrowseAVI.enabled := False;
BrowsePButton.enabled := True;
BrowseText.enabled := False;
SendAVI.enabled := False;
SendText.enabled := False;
SendSP.enabled := true;
if opendialog1.execute then
EditSound.text:= opendialog1.filename
else messagedlg('No selected file '+#13+ ' browseSound again',
mtinformation,[mbOK],0);
end;

```

```

procedure Tsenderform.BrowsePButtonClick(Sender: TObject);
var extention : string;
post : integer;
begin
SendAVI.enabled := False;
SendText.enabled := False;
SendSp.enabled := True;
BrowseText.enabled := False;
BrowseAVI.enabled := False;
With OpenFileDialog1 do
if Execute then
begin
EditPicture.text:= opendialog1.filename;
post:=pos('.',Editpicture.text);
Extention := copy(Editpicture.text,post,4);
if Extention = '.jpg' then
JPEGImage1.filename := Opendialog1.filename;
if Extention = '.bmp' then
JPEGImage1.picture.LoadFromFile(Opendialog1.filename);
end
else messagedlg('No selected file '+#13+ ' BrowsePicture again',
mtinformation,[mbOK],0);
end;

```

```

procedure Tsenderform.CloseBitBtnClick(Sender: TObject);
begin
Close;
end;

```

```

procedure Tsenderform.SendAVIClick(Sender: TObject);
begin
Sendonefile;
Sockets1.Getpeerport(sockets1.socketNumber);
Sockets1.GetpeerIPAddr(sockets1.socketNumber);
InitSend;
end;

```

procedure Tsenderform.BrowseAVIClick(Sender: TObject);

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

begin
BrowseSButton.enabled := False;
BrowsePButton.enabled := False;
BrowseText.enabled := False;
SendSP.enabled := False;
SendText.enabled := False;
SendAVI.enabled := True;
if Opendialog1.Execute then
begin
EditAVI.text := Opendialog1.filename;
MediaPlayer1.filename:=EditAVI.Text;
MediaPlayer1.Open;
MediaPlayer1.Play;
end;
end;
procedure Tsenderform.StopAVIClick(Sender: TObject);
begin
MediaPlayer1.Stop;
end;

procedure Tsenderform.ConnectClick(Sender: TObject);
begin
InitSend;
Sockets1.IPAddr := EditIP.Text;
Sockets1.Port :=EditPort.Text;
BrowseSButton.Enabled:= True;
BrowsePButton.Enabled:= True;
BrowseAVI.Enabled:= True;
BrowseText.Enabled:= True;
Sockets1.Nonblocking := False;
Sockets1.Timeout := 30;
Sockets1.SConnect ;
If SocketsError then
begin
Memo2.lines.add('..Can not find server..');
CloseSocket;
Exit;
end;
DoSend;
end;

procedure Tsenderform.BrowseTextClick(Sender: TObject);
var post: integer;
Extension : string;
begin
BrowseSButton.Enabled:= False;
BrowsePButton.Enabled:= False;
BrowseAVI.Enabled:= False;
SendSP.enabled := False;
SendAVI.enabled := False;
SendText.enabled := true;
If Opendialog1.Execute then
begin
EditText.text := Opendialog1.filename;
post:=pos('.',EditText.text);
Extension := copy(EditText.text,post,4);
if Extension = '.txt' then
Memo1.Lines.LoadFromFile(OpenDialog1.filename);
end;
end;

procedure Tsenderform.SendTextClick(Sender: TObject);
begin
Sendonefile;
Sockets1.Getpeerport(sockets1.socketNumber);
Sockets1.GetpeerIPAddr(sockets1.socketNumber);
InitSend;
end;

procedure Tsenderform.SendOnefile;
var
f1:file;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

numread: integer;
fsize,bytewrite:longint;
filetosend:string;
szbuf:array[0..2047] of char;
begin
Memo2.lines.add('Begin sending');
filetosend:= Opendialog1.filename;
assignfile (f1,filetosend);
{$I-}
reset(f1,1);
{$I+}
if IOresult<> 0 then
begin
Memo2.lines.add('..Can not open file..');
closesocket;
exit;
end;
fsize:= filesize(f1);
bytewrite:= 0;
sockets1.text :=(''+filetosend+'')+'<'+inttostr(fsize)+'>';
repeat
blockread(f1,szbuf, sizeof(szbuf),numread);
sockets1.ssend(sockets1.socketnumber,szbuf,numread);
bytewrite := bytewrite+numread;
until (bytewrite >= fsize) or socketerror;
if socketerror then
begin
Memo2.lines.add('Error while sending ');
closesocket;
end;
closefile(f1);
Memo2.lines.add('..Finished..')
end;

procedure Tsenderform.RecClick(Sender: TObject);
begin
with mediaplayer1 do
begin
filename := 'c:\muay.wav';
open;
position := 0;
startrecording;
end;
end;

procedure Tsenderform.stopClick(Sender: TObject);
begin
with mediaplayer1 do
begin
filename := 'c:\sound.wav';
stop;
save;
close;
end;
end;
end.

```

```
unit recv;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
Sockets, StdCtrls, Buttons, winsock, MPlayer, ExtCtrls, JpgImg, Gauges,
ComCtrls, MMsystem;
```

```
type
```

```
Trecvform = class(TForm)
```

```
Sockets1: TSockets;
```

```
GroupBox1: TGroupBox;
```

```
GroupBox2: TGroupBox;
```

```
GroupBox3: TGroupBox;
```

```
GroupBox4: TGroupBox;
```

```
GroupBox5: TGroupBox;
```

```
Dir: TLabel;
```

```
port: TLabel;
```

```
EditPort: TEdit;
```

```
EditFilename: TEdit;
```

```
listen: TBitBtn;
```

```
PlayButton: TBitBtn;
```

```
StopButton: TBitBtn;
```

```
CancelBitBtn: TBitBtn;
```

```
CloseBitBtn: TBitBtn;
```

```
Show: TBitBtn;
```

```
JPEGImage1: TJPEGImage;
```

```
Memo1: TMemo;
```

```
GroupBox6: TGroupBox;
```

```
Memo2: TMemo;
```

```
MediaPlayer1: TMediaPlayer;
```

```
procedure Sockets1SessionClosed(Sender: TObject; Socket: Integer);
```

```
procedure Sockets1ErrorOccurred(Sender: TObject; Socket,
```

```
Error: Integer; Msg: string);
```

```
procedure CancelBitBtnClick(Sender: TObject);
```

```
procedure FormCreate(Sender: TObject);
```

```
procedure PlayButtonClick(Sender: TObject);
```

```
procedure StopButtonClick(Sender: TObject);
```

```
procedure ShowClick(Sender: TObject);
```

```
procedure listenClick(Sender: TObject);
```

```
private
```

```
{ Private declarations }
```

```
procedure Initreceive;
```

```
procedure Beginlistening;
```

```
procedure Play;
```

```
procedure ShowPicture;
```

```
procedure Receive;
```

```
procedure Closesocket;
```

```
procedure Getfile;
```

```
procedure GetTwoFile;
```

```
procedure GetOneFile;
```

```
procedure GetOneSound;
```

```
procedure Handshake;
```

```
procedure findheadOnefile;
```

```
procedure findheadTwofile;
```

```
procedure findheadriff;
```

```
procedure findheadData;
```

```
function GetNameSound:string;
```

```
function GetSizeSound:longint;
```

```
function GetNamePicture:string;
```

```
function GetSizePicture:longint;
```

```
function Socketerror :boolean;
```

```
public
```

```
{ Public declarations }
```

```
Timedout :boolean;
```

```
errorreturn :integer;
```

```
end;
```

```
type
```

```
Vszbufs= array[0..2047] of char;
```

เอกสารนี้เป็นเอกสารลิขสิทธิ์ส่วนราชการสำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

var
 recvform: Trecvform;
 Count,num,numheaddata,numheadOne,numstartTwo: integer;
 header,Extension,ExtensionP :string;
 f,f1,f2,s,p : file;
 SzbufS,SzbufSOne,SzbufSTwo,headRiff,headdata,headOne,headTwo:vszbufs ;

implementation

{$R *.DFM}

procedure Trecvform.Initreceive;
begin
 Errorreturn := 0;
 Timedout := false;
end;

procedure Trecvform.Beginlistening;
begin
 Sockets1.IPAddr :='' ;
 Sockets1.Port := Editport.text;
 Sockets1.Nonblocking := False;
 Sockets1.Timeout := 0;
 Memo2.lines.add('<< Waiting >> ');
 CancelBitbtn.Enabled:= true;
 Sockets1.sListen;
 if Socketerror then
 begin
 Memo2.lines.add('..Can not connecting..');
 Sockets1.sCancellisten;
 Closesocket;
 Exit;
 end;
 Count:=0;
 while true do
 begin
 receive;
 if messagedlg ('Waiting next File',mtconfirmation,[mbytes,mbno],0)=
 mrno then
 begin
 Listen.enabled := False;
 break;
 end;
 inc(Count);
 Memo2.lines.add('..Wait next file..');
 end;
end;

procedure Trecvform.Sockets1SessionClosed(Sender: TObject; Socket: Integer);
begin
 Memo2.lines.add('..Socket Closed..');
 Closesocket;
end;

procedure Trecvform.Sockets1ErrorOccurred(Sender: TObject; Socket,
 Error: Integer; Msg: string);
begin
 Errorreturn := Error;
 if Error =WSAETimedout then
 begin
 Timedout := true;
 Errorreturn :=0;
 end;
 Messagedlg('@'+inttostr(error)+' '+msg, mterror,[mbok],0);
 if Sockets1.Socketnumber<> INVALID_SOCKET then
 begin
 Memo2.lines.add('<< Close Socket>>');
 Closesocket;
 end;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

end;

function Trecvform.SocketsError : boolean;
begin
 Result := false;
 if Timedout then
 begin
 Result := true;
 Errorreturn := 0;
 Messagedlg('Time out error',mtError,[mbok],0);
 end;
 if (Errorreturn <> 0) then
 begin
 Result := true;
 Errorreturn := 0;
 Messagedlg(' Socket Error',mtError,[mbok],0);
 end;
 end;

procedure Trecvform.Closesocket;
begin
 if Sockets1.Socketnumber<> INVALID_SOCKET then
 Sockets1.Sclose;
 end;

procedure Trecvform.CancelBitBtnClick(Sender: TObject);
begin
 Sockets1.SCancelListen;
 if Sockets1.Socketnumber<> INVALID_SOCKET then
 Sockets1.Sclose;
 Memo2.lines.add('..Stop..');
 Initreceive;
 Close;
end;

procedure Trecvform.Receive;
begin
 Sockets1.sAccept;
 Memo2.lines.add('..Connected..');
 Handshake;
 if Socketerror then
 begin
 Closesocket;
 Exit;
 end;
 Getfile;
 sockets1.GetPeerPort(sockets1.SocketNumber);
 sockets1.GetPeerIPAddr(sockets1.SocketNumber);
 Memo2.lines.add(' ..Finished..');
 end;

procedure Trecvform.findheadTwofile;
var
 Int,numheadtwo,num2:integer;
begin
 num2:=0;
 numstartTwo := 0;
 while SzbufsTwo[num2] <> '$' do num2 := num2+1;
 numstartTwo := num2+1;
 numheadTwo :=0;
 For Int := numstartTwo to 2047 do
 begin
 headTwo[numheadTwo] := szbufsTwo[Int];
 numheadTwo := numheadTwo + 1;
 end;
end;

procedure Trecvform.GetTwofile;
var
 NumreadS,NumreadP,il,postS,postP,n,np: integer;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

SizeSound,SizePicture,byteLeftS,ByteLeftP: longint;
NameSound,NamePicture:string;
buffS,buffP: char;

begin
Memo2.lines.add('..Begin to receive 2 file..');
NameSound:= GetNameSound;
SizeSound:= GetSizeSound;
NamePicture:= GetNamePicture;
SizePicture:= GetSizePicture;
postS:=pos('.',NameSound);
Extension := copy(NameSound,postS,4);
postp:=pos('.',Namepicture);
ExtensionP := copy(Namepicture,postp,4);
Assignfile(s,EditFilename.text+'\FileTest0'+inttostr(count)+Extension);
{$I-} Rewrite(s,1); {$I+}
if IOResult<> 0 then
begin
Memo2.lines.add('..Error IO sound..');
Exit;
end;
Assignfile(p,Editfilename.text+'\FileTest1'+inttostr(count)+ExtensionP);
{$I-} Rewrite(p,1); {$I+}
if IOResult<> 0 then
begin
Memo2.lines.add('..Error IO picture..');
Exit;
end;
ByteleftS :=SizeSound;
ByteLeftP := SizePicture;
n := 0;
while (byteleftS >0) and (not socketerror) do
begin
numreadS := sizeof(SzbufSTwo);
numreadS:= sockets1.sreceive(sockets1.socketnumber,SzbufSTwo,numreadS);
if not Socketerror then
begin
n := n+1;
if n = 1 then
begin
findheadTwofile;
blockwrite (s,headTwo,2048-numstartTwo);

end
else { for n =1}
begin
blockwrite (s,SzbufSTwo,numreadS);
end ;
byteleftS :=byteleftS - numreadS;
end
else { for if not socketError}
begin
Memo2.lines.add('..Error while recv..');
sockets1.sclose;
exit;
end;
end; { end of receive while sound}
Closefile(s);
Memo2.lines.add('Now getting file Please wait!');
while (byteleftP >0) and (not socketerror) do
begin
numreadP := sizeof(SzbufSTwo);
numreadP:= sockets1.sreceive(sockets1.socketnumber,SzbufSTwo,numreadP);
if not Socketerror then
begin
blockwrite (p,SzbufSTwo,numreadP);
byteleftP :=byteleftP - numreadP;
end
else { else for not socketsError}
begin
Memo2.lines.add('..Error while recv picture..');
sockets1.sclose;
exit;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

 end;
 end; { end of receive while picture}
closefile(p);
Closesocket;
with JPEGImage1 do
begin
 memol.lines.add(' jpect ');
 if ExtensionP = '.bmp' then
 JPEGImage1.Picture.LoadFromFile
 (EditFilename.text+'\FileTest1'+inttostr(count)+'.bmp');
 if ExtensionP = '.jpg' then
 JPEGImage1.Filename:=
 EditFilename.text+'\FileTest1'+inttostr(count)+'.jpg';
 end; { end with jpegimage1}
end;
play;
end;{ end procedure}

```

```

procedure Trecvform.findheadRiff;
var
 Int,numstartR,numheadRiff:integer;
begin
 num:=0;
 numstartR := 0;
 while Szbufs[num] <> '>' do num := num+1;
 numstartR := num+1;
 numheadRiff := 0;
 For Int := numstartR to num +59 do
 begin
 headRiff[numheadRiff] := szbufs[Int];
 numheadRiff := numheadRiff +1;
 end;
 end;
end;

```

```

procedure Trecvform.findheadData;
var
 Int,numstartdata :integer;
begin
 numstartdata := 0;
 numstartdata := num + 59; { position of Real data};
 numheaddata :=0;
 For Int := numstartdata to 2047 do
 begin
 headData[numheaddata] := szbufs[Int];
 numheaddata := numheaddata + 1;
 end;
 end;
end;

```

```

procedure Trecvform.GetOneSound;
var
 numreadS,i,postS,n,real,seq,nwrite,endnumread,newnum: integer;
 SizeFile,byteleftS: longint;
 NameFile:string;
 buffS: char;
 headf : array[0..57]of char;
 newheaddata : array[0..2047]of char;
 Label waitfile1,waitfile2,waitfile3;

begin
 headf := 'RIFF'+#$33+#$08+#00+#00+'WAVEfmt ' +
 #$12+#$00+#$00+#$00+#$01+#$00+#$01+#$00+
 #$22+#$56+#00+#00+#$22+#$56+#00+#00+
 #01+#00+#08+#00+#$38+#00+#$66+#$61+ #$63+#$74+#$04+#00+#00+#00+#00+#$08+
 #00+#00+#$64+#$61+#$74+#$61+#00+#08+ #00+#00;
 memo2.lines.add('..Begin to receive2..');
 NameFile:= GetNameSound;
 memo2.lines.add('NAME DETECT'+namefile);
 SizeFile:= GetSizeSound;
 memo2.lines.add('SIZE DECT'+inttostr(sizefile));
 ByteleftS :=SizeFile;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

postS:=pos('.',NameFile);
Extension := copy(NameFile,postS,4);
memo2.lines.add('BEGIN ASSIGN FILE');
Assignfile(f,EditFilename.text+'\File0'+inttostr(count)+Extension);
{$I-} Rewrite(f,1); {$I+}
if IOResult<> 0 then
begin
Memo2.lines.add('..Error IO HA HA!..');
Exit;
end;
n:=0;
memo2.lines.add('BEGIN LOOP WHILE');

{start while for receive and decreat file size}
while (byteleftS >0) and (not socketerror) do
begin {start begin while}
numreadS := sizeof(szbufS);
numreadS:= sockets1.sreceive(sockets1.socketnumber,szbufS,numreadS);
if not Socketerror then
begin {start not socketError}
n:=n+1;
if n=1 then
begin {start if n =1}
deletefile('c:\templ.wav');
assignFile(f1,'c:\templ.wav');
{$I-} Rewrite(f1,1); {$I+}
if IOResult<> 0 then
begin
Memo2.lines.add('..Error IO HA1 HA1!..');
Exit;
end;
blockwrite(f1,Headf,sizeof(Headf),Nwrite);
findheadRiff;
blockwrite (f,headRiff,58);
findheaddata;
i:=0;
newnum := num-1;
for i := 0 to newnum do
NEWheaddata[i] := #00;
blockwrite (f1,headdata,2048-num);
blockwrite (f1,NEWheaddata,num);
blockwrite (f,headdata,numheaddata);
closeFile(f1);
waitfile3:if sndPlaySound('c:\templ.wav',SND_SYNC) then
Memo1.lines.Add('Playing') else goto waitfile3;
end{end N=1}
else { n <> 1}
begin { n<> 1}
seq := n mod 2 ;
if seq = 1 then
begin
deletefile('c:\templ.wav');
assignFile(f1,'c:\templ.wav');
{$I-}rewrite(f1,1);{$I+}
if IOResult<> 0 then
begin
Memo2.lines.add('..Error IO HA11 HA1!1..');
Exit;
end;
blockwrite(f1,Headf,sizeof(Headf),Nwrite);
if numreadS < 2048 then
begin
endnumread := numreads;
while endnumread < 2048 do
begin
szbufs[endnumread +1]:= #00;
endnumread := endnumread +1 ;
end;{end of while numreads < 2048 }
blockwrite (f1,szbufS,2048);
blockwrite (f,szbufS,numreads);
closefile(f1);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

end{numread < 2048}
else {numreadS = 2048}
begin
 blockwrite (f1,szbufs,numreadS);
 blockwrite (f,szbufs,numreads);
 closeFile(f1) ;
 end; {numreads =2048 }
waitfile1;if sndPlaySound('c:\temp1.wav',SND_SYNC) then
Memo1.lines.Add('Playing1') else goto waitfile1;
end {seq = 1}
else
begin {seq = 0}
deletefile('c:\temp2.wav');
assignFile(f2,'c:\temp2.wav');
{$I-}rewrite(f2,1); {$I+}
if IOResult<> 0 then
begin
Memo2.lines.add('..Error IO HA2 HA!2..');
Exit;
end;
blockwrite(f2,Headf,sizeof(Headf),Nwrite);
if numreadS < 2048 then
begin
while numreadS < 2048 do
begin
szbufs[numreadS +1]:= #00;
numreadS := numreadS +1 ;
end;{end of while numreads < 2048 }
blockwrite (f2,szbufS,2048);
blockwrite (f,szbufS,numreads);
closefile(f2);
end {numread < 2048}
else {numreadS = 2048}
begin
blockwrite (f2,szbufS,numreads);
blockwrite (f,szbufs,numreads);
closeFile(f2);
end; { numreads = 2048}
waitfile2;if sndPlaySound('c:\temp2.wav',SND_SYNC) then
Memo1.lines.Add('Playing2') else goto waitfile2;
end;{end seq = 0}
end; {end <> 1}
byteleftS :=byteleftS - numreadS;
end { socket noterror}
else
begin { start socket error}
Memo2.lines.add('..Error while recv..');
sockets1.sclose;
exit;
end; {end socket error}
end; { end of receive while file}
Closefile(f);
Closesocket;
end;{ end procedure}

```

```

procedure Trecvform.findheadOnefile;
var
 Int,numstart,num1 :integer;
begin
 num1:=0;
 numstart := 0;
 while SzbufsOne[num1] <> '>' do num1 := num1+1;
 numstart := num1+1;
 numheadOne :=0;
 For Int := numstart to 2047 do
 begin
 headOne[numheadOne] := SzbufSOne[Int];
 numheadOne := numheadOne + 1;
 end;
 end;
end;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

procedure Trecvform.GetOnefile;
var
 numreadS,il,postS,n: integer;
 SizeFile,byteleftS: longint;
 NameFile:string;
 buffS: char;
begin
 memo2.lines.add('..Begin to receive..');
 NameFile:= GetNameSound;
 SizeFile:= GetSizeSound;
 postS:=pos('.',NameFile);
 Extension := copy(NameFile,postS,4);
 Assignfile(s,EditFilename.text+'\FileTest0'+inttostr(count)+Extension);
 {$I-} Rewrite(s,1); {$I+}
 if IOResult<> 0 then
 begin
 Memo2.lines.add('..Error IO sound..');
 Exit;
 end;
 n:= 0;
 ByteleftS :=SizeFile;
 while (byteleftS >0) and (not socketerror) do
 begin
 numreadS := sizeof(SzbufSOne);
 numreadS:= sockets1.sreceive(sockets1.socketnumber,SzbufSOne,numreadS);
 if not Socketerror then
 begin
 n := n+1;
 if n = 1 then
 begin
 findheadOnefile;
 blockwrite(s,headOne,numheadOne);
 end
 else
 begin
 blockwrite (s,SzbufSOne,numreadS);
 end ;
 byteleftS :=byteleftS - numreadS ;
 end
 else
 begin
 Memo2.lines.add('..Error while recv..');
 sockets1.sclose;
 exit;
 end;
 end; { end of receive while file}
 Closefile(s);
 Closesocket;
 if (Extension='.avi') then Play ;
 if (Extension='.bmp') or (Extension = '.jpg')then ShowPicture;
 if (Extension='.txt') then Showclick(self);
end;{ end procedure}

```

```

procedure Trecvform.Getfile;
var
 post,PosEx: integer;
 buf, NameExten:string;
begin
 buf := '';
 buf := sockets1.peek;
 post:= pos('%',buf);
 if post <> 0 then
 begin
 GetTwoFile ;
 end
 else
 begin
 NameExten := GetNameSound;
 PosEx := pos('.',NameExten);
 Extension := copy(NameExten,PosEx,4);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

if Extension = '.wav' then
 GetOneSound
else
 GetOneFile;
end;
end;

procedure Trecvform.Play;
begin
MediaPlayer1.filename:=EditFilename.text+'\FileTest0'+inttostr(count)+Extension;
MediaPlayer1.open;
MediaPlayer1.play;
end;

procedure Trecvform.ShowPicture;
begin
 with JPEGImage1 do
 begin
 if Extension = '.bmp' then
 JPEGImage1.Picture.LoadFromFile(EditFilename.text+'\FileTest1'+inttostr(count)+'.'+
 Extension+'.bmp');
 if Extension='.jpg' then
 JPEGImage1.Picture.LoadFromFile(EditFilename.text+'\FileTest1'+inttostr(count)+'.'+
 Extension+'.jpg');
 end; { end with jpegimage1 }
 end;

Function Trecvform.GetNameSound: string ;
var
 buf:string;
 szbuf: array[0..255] of char absolute buf;
 idx,idy,len:integer;
begin
 buf := '';
 buf := sockets1.peek;
 if socketerror then exit;
 idx := pos('(',buf);
 if idx<= 0 then exit;
 idy := pos(')',buf);
 len :=idy-idx;
 if len <= 0 then Exit;
 Result := copy(buf,(idx+1),(len-1));
end;

Function Trecvform.GetNamePicture: string ;
var
 buf:string;
 szbuf: array[0..255] of char absolute buf;
 idx,idy,len:integer;
begin
 buf := '';
 buf := sockets1.peek;
 if socketerror then exit;
 idx := pos('(',buf);
 if idx<= 0 then exit;
 idy := pos(')',buf);
 len :=idy-idx;
 if len <= 0 then Exit;
 Result := copy(buf,(idx+2),(len-2));
end;

function Trecvform.GetSizeSound:longint;
var
 buf:string;
 szbuf:array[0..255] of char absolute buf;
 ch: char;
 idx,idy,len:integer;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

begin
 buf:='';
 buf:= sockets1.peek;
 idx :=pos('<',buf);
 if idx<= 0 then exit;
 idy :=pos('>',buf);
 len := idy-idx;
 if len <= 0 then exit;
 Result := strtoint(copy(buf,(idx+1),(len-1)));
end;

```

```

function Trecvform.GetSizePicture:longint;
var
 buf:string;
 szbuf:array[0..255] of char absolute buf;
 ch: char;
 idx,idy,len:integer;
begin
 buf:='';
 buf:= sockets1.peek;
 idx :=pos('<<',buf);
 if idx<= 0 then
 exit;
 idy :=pos('$',buf);
 len := idy-idx;
 if len <= 0 then exit;
 Result := strtoint(copy(buf,(idx+2),(len-4)));
end;

```

```

procedure Trecvform.Handshake;
var str: string;
begin
 str:= sockets1.text;
 Memo2.lines.add('>'+str+'<');
 if pos('START',str)=0 then
 begin
 Memo2.lines.add('...Invalid header...');
 errorreturn :=-1;
 exit;
 end;
 sockets1.text:= 'OK';
end;

```

```

procedure Trecvform.FormCreate(Sender: TObject);
begin
 memo1.lines.clear;
 memo2.lines.clear;
 EditFilename.text := 'c:\multimedia';
end;

```

```

procedure Trecvform.PlayButtonClick(Sender: TObject);
begin
 MediaPlayer1.filename :=EditFilename.text+'\FileTest0'+inttostr(count)+
 Extension;
 MediaPlayer1.open;
 MediaPlayer1.play;
end;

```

```

procedure Trecvform.StopButtonClick(Sender: TObject);
begin
 MediaPlayer1.Stop;
end;

```

```

procedure Trecvform.ShowClick(Sender: TObject);
begin
 Memo1.lines.LoadFromFile(EditFilename.text+'\FileTest0'+inttostr(count)+
 extension);
end;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
procedure Trecvform.listenClick(Sender: TObject);
begin
 Initreceive;
 Beginlistening;
end;
end.
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

unit fstring;

interface

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
 Forms, Dialogs, StdCtrls, WinSock, Sockets, Buttons;

type
 Tstringform = class(TForm)
 Sockets1: TSockets;
 GroupBox1: TGroupBox;
 IPAddr: TEdit;
 Port: TEdit;
 Label1: TLabel;
 Label2: TLabel;
 GroupBox2: TGroupBox;
 Edit1: TEdit;
 GroupBox3: TGroupBox;
 yourname: TEdit;
 Label3: TLabel;
 GroupBox4: TGroupBox;
 GroupBox5: TGroupBox;
 Memo1: TMemo;
 Connect: TBitBtn;
 Listen: TBitBtn;
 CancelListen: TBitBtn;
 SendData: TBitBtn;
 Close: TBitBtn;
 procedure ConnectClick(Sender: TObject);
 procedure FormCreate(Sender: TObject);
 procedure SendDataClick(Sender: TObject);
 procedure ListenClick(Sender: TObject);
 procedure CloseClick(Sender: TObject);
 procedure CancelListenClick(Sender: TObject);
 procedure Sockets1SessionConnected(Sender: TObject; Socket: integer);
 procedure Sockets1SessionAvailable(Sender: TObject; Socket: integer);
 procedure Sockets1DataAvailable(Sender: TObject; Socket: integer);
 procedure Sockets1SessionClosed(Sender: TObject; Socket: integer);
 procedure Edit1KeyPress(Sender: TObject; var Key: Char);
 procedure Edit1Click(Sender: TObject);
 procedure Edit1Enter(Sender: TObject);

 { Private declarations }
 public
 { Public declarations }
 end;

var
 stringform: Tstringform;
 clientserver: integer;

implementation

{$R *.DFM}

procedure Tstringform.ConnectClick(Sender: TObject);
begin
 Edit1.Enabled := true;
 Listen.Enabled := False;
 Sockets1.IPAddr := IPAddr.Text;
 Sockets1.Port := Port.Text;
 Sockets1.SConnect;
 clientserver := 1;
 Connect.Enabled := False;
 senddata.Enabled := true;
end;

procedure Tstringform.FormCreate(Sender: TObject);
begin
 memo1.lines.clear;
 Port.Text := Sockets1.Port;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

IPAddr.Text := Sockets1.IPAddr;
Connect.Enabled := True;
SendData.Enabled := False;
Listen.Enabled := True;
Close.Enabled := False;
CancelListen.Enabled := False;
clientserver := 0;
end;

procedure Tstringform.SendDataClick(Sender: TObject);
begin
 Sockets1.Text := Edit1.Text;
 Edit1.Text := '';
end;

procedure Tstringform.ListenClick(Sender: TObject);
begin
 Edit1.Text := '';
 Edit1.Enabled := False;
 Connect.Enabled := False;
 SendData.Enabled := False;
 Sockets1.Port := Port.Text;
 Sockets1.SListen;
 Listen.Enabled := False;
 Close.Enabled := False;
 CancelListen.Enabled := True;
 clientserver := 2;
 Memo1.Lines.Add('Listening');
end;

procedure Tstringform.CloseClick(Sender: TObject);
begin
 Sockets1.SClose;
 if clientserver <> 2 then {not receiver?}
 begin
 Connect.Enabled := True;
 SendData.Enabled := False;
 Listen.Enabled := True;
 Close.Enabled := False;
 CancelListen.Enabled := False;
 clientserver := 0;
 end;
end;

procedure Tstringform.CancelListenClick(Sender: TObject);
begin
 If Sockets1.SocketNumber <> INVALID_SOCKET Then
 Sockets1.SClose;
 If Sockets1.MasterSocket <> INVALID_SOCKET Then
 Sockets1.SCancelListen;
 Connect.Enabled := True;
 SendData.Enabled := False;
 Listen.Enabled := True;
 Close.Enabled := False;
 CancelListen.Enabled := False;
end;

procedure Tstringform.Sockets1SessionConnected(Sender: TObject; Socket: integer);
begin
 Connect.Enabled := False;
 SendData.Enabled := True;
 Listen.Enabled := False;
 Close.Enabled := True;
 CancelListen.Enabled := False;
 Memo1.Lines.Add('Connected to IP='+IPAddr.Text);
end;

procedure Tstringform.Sockets1SessionAvailable(Sender: TObject; Socket: integer);
var
 new_client: TSocket;
begin
 new_client := Sockets1.SAccept;
 Memo1.Lines.Add('Accept');

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Listen.Enabled := False;
CancelListen.Enabled := True;
end;

procedure Tstringform.Sockets1DataAvailable(Sender: TObject; Socket: integer);
var
 Buf: string;
begin
 Sockets1.SocketNumber := Socket;
 buf := Sockets1.Text;
 Memo1.Lines.Add(buf);
 If clientserver = 2 Then
 begin
 Sockets1.Text := buf;
 end;
end;

procedure Tstringform.Sockets1SessionClosed(Sender: TObject; Socket: integer);
begin
 Memo1.Lines.Add('Lost connection');
 CloseClick(Self);
end;

procedure Tstringform.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
 Case key of
 #13: SenddataClick(Self);
 end;
end;

procedure Tstringform.Edit1Click(Sender: TObject);
begin
 Edit1.text:='->'+yourname.text+':';
end;

procedure Tstringform.Edit1Enter(Sender: TObject);
begin
 Edit1.text:='->'+yourname.text+':';
end;

end.

```

## ภาคผนวก ข

### แสดงความหมายของคำสั่งเกี่ยวกับคอมโพเนนต์ Socket

|                                              |                                                                                                                                                                                         |
|----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Sockets.IPAddr                               | กำหนดหมายเลข IP ของที่จะทำการติดต่อด้วย                                                                                                                                                 |
| Sockets.Nonblocking                          | กำหนดโหมดของการทำงานของ Socket ว่าเป็น non blocking mode หรือ blocking mode                                                                                                             |
| Sockets.OnErrorOccurred                      | ส่งแมสเสจแจ้ง โปรแกรมเมื่อเกิดข้อผิดพลาดของSocket                                                                                                                                       |
| Sockets.OnSessionClosed                      | ส่งแมสเสจแจ้ง โปรแกรมเมื่อ Socket ของอีกฝ่ายหนึ่งที่ติดต่อกันอยู่ขณะนั้นถูกปิด                                                                                                          |
| Sockets.Peek                                 | สั่งให้ทำการเรียกดูข้อมูลใน Buffer โดยไม่ต้องข้อมูลที่เก็บไว้ใน Buffer ออกมา ข้อมูลสูงสุด 255 ตัวอักษร                                                                                  |
| Sockets.Port                                 | กำหนดหมายเลขของพอร์ต ของ Socket ที่ใช้ในการติดต่อกับ โปรแกรมประยุกต์ที่ต้องการ                                                                                                          |
| Sockets.SAccept                              | จะใช้คู่กับคำสั่ง SListen โดยเมื่อมีผู้ขอติดต่อ หลังจากคำสั่ง SListen แล้วฝ่ายที่สั่ง Slisten จะสั่ง SAccept จะสร้างการติดต่อให้เกิดขึ้น                                                |
| Sockets.SCancelListen                        | สั่งให้ยกเลิกการรอคอย                                                                                                                                                                   |
| Sockets.SClose                               | สั่งให้ปิด Socket ที่ได้ทำการเปิดอยู่แล้ว                                                                                                                                               |
| Sockets.SConnect                             | ใช้ร้องขอการติดต่อกับฝ่ายที่คอยด้วยคำสั่ง SListen                                                                                                                                       |
| Sockets.SListen                              | สั่งให้ทำการคอยการติดต่อจากอีกฝ่ายหนึ่ง ซึ่งจะเปรียบเสมือน เซิร์ฟเวอร์ โดยจะทำการคอยเป็นเวลาที่ได้กำหนดไว้ใน Timeout แต่ถ้า Timeout = 0 จะทำการรอคอยต่อไปอย่างไม่มีกำหนด                |
| Sockets.SocketNumber                         | แสดงหมายเลข Socket ที่ถูกกำหนดเมื่อมีการติดต่อเกิดขึ้น                                                                                                                                  |
| Sockets.SReceive(SocketNumber,Buffer,Lenght) | สั่งให้ทำการอ่านข้อมูลจากซ็อกเก็ต SocketNumber มาเก็บไว้ยัง Buffer เป็นจำนวน Length ไบต์ คำสั่งนี้จะทำการคืนค่าจำนวน ที่สามารถอ่านได้กลับให้อ่านข้อมูลจาก ซ็อกเก็ตให้แ่ก่ตัวแปรตัวหนึ่ง |
| Sockets.Ssend(SocketNumber,Data,Numwrite)    | สั่งให้ทำการอ่านข้อมูล Data ไปยังซ็อกเก็ตหมายเลข SocketNumber เป็นจำนวน numwrite ไบต์                                                                                                   |
| Sockets.Text                                 | ให้ส่งข้อมูลตัวแปรสตริง ซึ่งจะส่งได้สูงสุด 255ตัวอักษร                                                                                                                                  |
| Sockets.Timeout                              | กำหนดเวลาที่ใช้ในการรอรับข้อมูลจากซ็อกเก็ต ถ้าซ็อกเก็ตไม่ส่งข้อมูลมาภายในเวลาที่กำหนดนี้จะส่งสัญญาณเพื่อแสดงข้อผิดพลาด                                                                  |
| Sockets.Getpeerport                          | ให้ทำการคืนค่าพอร์ตที่ใช้                                                                                                                                                               |
| Sockets.GetpeerIPAddr                        | ให้ทำการคืนค่าไอพีแอดเดรสที่ใช้                                                                                                                                                         |

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## หนังสืออ้างอิง

1. Gary Connell,Troy Strain, *Delphi Nut & Bolts for experienced programmers*, Berkeley,CA:McGraw-Hill,c1995.
2. Scott Jarol, Dan Haygood: Chris D.Coppola, *Delphi 2 Multimedia:adventure set*, Scottsdale, AZ,Coriolis Group Books, c1996.
3. Murphy, Eamon, *TCP/IP tutorial and technical overview*, upper Saddle River, NJ:Prentice Hall PTR,c1995.
4. Tim Parker, *Teach Yourself TCP/IP in 14 Days*, second Edition:Indianapolis, Indiana :SAMS PUBLISHING,c1996.
5. James Martin with Joe Leben, *TCP/IP NETWORKING Architecture* , Administration, and Programming,Englewood Cliffs,NJ : PTR Prentice Hall, c1994.
6. กิตติ ภัคดีวัฒนะกุล, Netscape ( ALL - IN - ONE ), บริษัท เคทีพี คอมพ์ แอนด์ คอนซิลท์ จำกัด, กรุงเทพฯ, 2540