การพัฒนาโปรแกรม รับ-ส่ง ไฟล์แบบบีบอัดอัตโนมัติบนเครือข่ายอินเตอร์เนต

A DEVELOPMENT FILE TRANSFER WITH COMPRESSION PROGRAM ON INTERNET

กังวาน อัศวไชวศิน
ภุชงค์ วิบุญวิริยะวงศ์
วิชาย แซ่ลี้

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

A DEVELOPMENT FILE TRANSFER WITH COMPRESSION PROGRAM ON INTERNET

KUNGWAN ASSAWACHAIWASIN
BHUCHONG VIBOONVIRIYAWONG
VICHAI SAELEE

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCES
FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECGNOLOGY LADKRABANG
ACADEMIC YEAR 1999

หัวข้อปัญหาพิเศษ การพัฒนาโปรแกรมรับ-ส่งไฟล์ แบบบีบอัดอัตโนมัติบนเครือข่ายอินเตอร์เนต
A DEVELOPMENT FILE TRANSFER WITH COMPRESSION PROGRAM ON INTERNET

ชื่อนักศึกษา	นายกังวาน อัศวไชวศิน	39054604
	นายภุชงค์ วิบุญวิริยะวงศ์	39054649
	นายวิชาย แซ่ลี้	39054660
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ	

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2542

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	อาจารย์ไพรบูลย์ พันธรักษ์พงษ์	
กรรมการ	อาจารย์สิริลักษณ์ อนันต์สถิตย์สิน	
กรรมการและอาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ	

(อาจารย์ไพรบูลย์ พันธรักษ์พงษ์)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

หัวข้อปัญหาพิเศษ	การพัฒนาโปรแกรมรับ-ส่ง ไฟล์แบบบีบอัดอัตโนมัติบนเครือข่ายอินเตอร์เนต	
ชื่อนักศึกษา	นายกังวาน อัศวไชวศิน	39054604
	นายภุชงค์ วิบุญวิริยะวงศ์	39054649
	นายวิชาย แซ่ลี้	39054660
ปริญญา	วิทยาศาสตรบัณฑิต	
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
ปีการศึกษา	2542	
อาจารย์ที่ปรึกษา	อาจารย์วิสันต์ ตั้งวงษ์เจริญ	

บทคัดย่อ

ในปัจจุบัน เครือข่ายอินเตอร์เนตเป็นเครือข่ายที่ได้รับความนิยมเป็นอย่างมากในองค์กรภาครัฐและเอกชน ด้วยความสามารถหลายประการ เช่น การเชื่อมต่อระหว่างเครือข่ายที่มีขนาดย่อมหรือแม้แต่เครือข่ายที่มีขนาดใหญ่ ความสามารถในการติดตั้งและมีความเป็นมาตรฐานเดียวกัน

ด้วยสาเหตุที่ได้รับความนิยมเป็นอย่างมาก ปัญหาที่มักจะพบคือ ปัญหาด้านความเร็ว ซึ่งอาจจะเกิดจากปัญหาด้านอุปกรณ์ หรือผู้ให้บริการแต่อีกปัจจัยหนึ่งที่มีความสำคัญเช่นกัน คือขนาดของข้อมูลที่ถูกส่งไปยังผู้รับ พบว่าหากขนาดข้อมูลนั้นถูกบีบอัดให้มีขนาดเล็กลงก่อนที่จะส่งข้อมูลชุดนั้นออกไป ก็จะเป็นการลดเวลาในการส่งได้และยังผลให้ลดปริมาณข้อมูลบนเครือข่ายในระบบเครือข่ายอีกด้วย

ด้วยเหตุผลข้างต้น จึงได้เสนอการทำการพัฒนาโปรแกรมการบีบอัดไฟล์แบบอัตโนมัติเพื่อเข้าจัดการกับปัญหาในเรื่องของขนาดของข้อมูล และอำนวยความสะดวกในการส่งข้อมูลบนเครือข่ายอินเตอร์เนต

ผลที่ทางกลุ่มคาดหวังที่จะได้รับ จากการพัฒนาโปรแกรมการบีบอัดไฟล์แบบอัตโนมัติ คือ สามารถที่จะลดขนาดของข้อมูลให้มีขนาดเล็กลงเพื่อลดเวลาในการส่งข้อมูลบนเครือข่าย อินเตอร์เนต สิ่งนี้จะหมายถึงการไปช่วยลดเวลาในการเชื่อมต่อระหว่างเครือข่ายที่ให้บริการกับผู้ใช้ที่เชื่อมต่อเข้าสู่เครือข่ายอินเตอร์เนต จึงทำให้ปริมาณผู้ใช้บนเครือข่ายมีปริมาณลดลง

Special Project Title	A Development File Transfer with Compression Program on Internet	
Students	Mr. Kungwan Assawachaiwasin	39054602
	Mr. Bhuchong Viboonviriyawong	39054649
	Mr. Vichai Saelee	39054660
Degree	Bachelor's Degree of Science	
Department	Mathematics and Computer Sciences, Faculty of Science	
Programme	Computer Sciences	
Academic Year	1999	
Special Project Advisor	Lecturer Wisan Tangwongcharoen	

ABSTRACT

Nowadays internet network system is very popular in all organization because of many abilities as the connection between system in both large and small sizes,the ability in integration and the same standard in all over the world.

According to all abilities there are some problems as speed which caused by the meterial or service center. Another factor which also important is the size of informations, if the size of that informations is compressed to be smaller before sending it may save more times and reduce traffic in network system.

Because of above reason, we offer the program development which is automatic compression to solve the problems which caused by size of informations and provide more abilities in communication of internet network system

The goal of this program development(automatic compression) is the abilities to reduce size of informations, to save time in term of sending informations on internet network system that means it save time during connection of user and also reduce user in network.

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเรื่อง การพัฒนาโปรแกรม รับ-ส่ง ไฟล์แบบบีบอัดอัตโนมัติบนเครือข่ายอินเตอร์เนต สามารถลุล่วงไปด้วยดี คณะผู้จัดทำต้องขอขอบพระคุณอาจารย์วิสันต์ ตั้งวงษ์เจริญ อาจารย์ผู้รับผิดชอบปัญหาพิเศษฉบับนี้ และ อาจารย์ไพรบูลย์ พันธรักษ์พงษ์ ที่ทั้งสองคอยให้คำปรึกษาในการแก้ไขปัญหาต่างๆ รวมทั้งเป็นผู้ตรวจสอบความถูกต้อง ของปัญหาพิเศษนี้

นอกจากนี้คณะผู้จัดทำต้องขอขอบพระคุณ บิดา มารดา ที่ได้ให้ความกรุณาสนับสนุนทางด้านปัจจัยสี่ และ ทุนทรัพย์ และต้องขอขอบคุณ พี่ปุ้ย จากบริษัท Next step และ พี่จรัญ บัณฑิตรุ่นที่14 ที่ได้ให้คำปรึกษา จนการทำปัญหาพิเศษนี้สำเร็จด้วยดี รวมทั้งเพื่อนๆ และ น้องๆ ที่ให้ความช่วยเหลือในการทำปัญหาพิเศษ ไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2542

สารบัญ

หน้า

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาพิเศษ

ในปัจจุบันการสื่อสารผ่านระบบเครือข่าย Internet เป็นที่ได้รับความนิยมเป็นอย่างมาก ในภาคธุรกิจทั้งภาคเอกชนและ ภาครัฐบาล ด้วยคุณสมบัติต่างๆ ที่เอื้อต่อภาคธุรกิจ ทำให้การจราจรภายในเครือข่ายมีความหนาแน่นมากด้วยเหตุนี้จึงส่งผลให้การทำงานในส่วนของการรับส่งข้อมูลเกิดความล่าช้า ซึ่งการทำงานในส่วนนี้เป็นหน้าที่หลักในระบบเครือข่าย Internet

หากเพียงอาศัยความเร็วระหว่างเครือข่ายที่ทำการเชื่อมต่อกันในการแก้ปัญหาก็เป็นเพียงปัจจัยที่เราไม่สามารถที่จะควบคุมได้เนื่องจาก เป็นการลงทุนของผู้ให้บริการของเครือข่ายนั้น ด้วยเหตุนี้เราจึงหันมามองในปัจจัยที่เป็นส่วนที่ควบคุมหรือเลือกที่จะใช้เพื่อให้เกิดความเร็วในการรับส่งหรือเป็นการลดการจราจรภายในระบบเครือข่ายนั่นคือ"การลดขนาดของข้อมูล" (Compression) ให้มีขนาดที่เล็กที่สุดเท่าที่จะสามารถทำได้ ก็จะสามารถช่วยในการลดความ คับคั่งภายในระบบเครือข่ายได้ในระดับหนึ่งแต่ยังคงต้องยึดถือความถูกต้องของข้อมูลในการรับส่งเป็นสำคัญเพราะ เป็นหัวใจในการรับส่งข้อมูลระหว่างเครือข่าย

1.2 วัตถุประสงค์ของปัญหาพิเศษ

เพื่อศึกษาถึงระบบการทำงานของระบบเครือข่ายโดยอาศัยโปรโตคอล TCP/IP ซึ่งเป็นโปรโตคอล ที่ให้บริการในการรับส่งข้อมูลผ่านเครือข่าย Internet และเพื่อลดความคับคั่งของข้อมูลภายในระบบเครือข่าย โดยการลดขนาดของข้อมูล(Compression) ที่ต้องการสำหรับการรับส่งภายในเครือข่าย

1.3 สมมติฐานของการศึกษา

- สามารถลดความคับคั่งของการจราจรภายในเครือข่าย
- ลดขนาดของข้อมูลให้มีขนาดที่เล็กที่สุด ตามแนวระเบียบวิธีคิด
- มีความถูกต้องของข้อมูลระหว่างการรับส่ง

1.4 ขอบเขตของปัญหาพิเศษ

- พัฒนาโปรแกรมด้วย Borland Delphi version 3.0 และ ภาษา C , C++

- ทำการรับส่งข้อมูลในระบบเครือข่าย Internet โดยผ่านโปรโตคอล TCP/IP และทำงานบน Platform " Window 95/98 "
- การ Compression อ้างอิงถึง Algorithm ของ Huffman และ LZ77 และ LZ78 เป็นหลัก
- ไม่พิจารณาถึงความปลอดภัยอื่นใดนอกจาก Software ที่ได้ผลิตขึ้นนี้
- ไม่พิจารณาถึงการใช้เครื่องคอมพิวเตอร์ ของ Apple และ McIntosh
- พิจารณาข้อมูลที่นำเข้าเฉพาะทางเมาส์ และ คีย์บอร์ด

1.5 ขั้นตอนในการดำเนินงาน

กรกฎาคม
- เก็บรวบรวมเอกสารและข้อมูลต่างๆ ที่เกี่ยว
- ศึกษาทฤษฎีการบีบอัดข้อมูล
- ศึกษาการทำงานของ TCP/IP

สิงหาคม
- ศึกษาการทำงานของระบบงาน Client / Server
- ศึกษา Tool ที่ใช้ในการพัฒน (ภาษา C ,C++ และ Borland Delphi 3.0)

กันยายน
- ศึกษา Tool ที่ใช้ในการพัฒนา (ต่อ)

ตุลาคม
- วิเคราะห์และออกแบบโปรแกรมในส่วนของการส่งข้อมูล

พฤศจิกายน
- ศึกษาลักษณะการทำงานของ Socket windows
- ศึกษาระบบปฏิบัติการของ Windows

ธันวาคม
- วิเคราะห์และออกแบบโปรแกรมในส่วนของการรับข้อมูล
- ศึกษาและทดลองออกแบบโปรแกรมบีบอัดข้อมูล

มกราคม
- พัฒนาทดสอบและปรับปรุงโปรแกรม

กุมภาพันธ์
- พัฒนาทดสอบและปรับปรุงโปรแกรม

มีนาคม
- จัดทำเอกสารประกอบ

1.6 ข้อจำกัดของการศึกษา

- ไม่สามารถทำการ compress file ที่อยู่ใน drive A: ได้
- ไม่สามารถทำการ compress file ที่มีชื่อยาวเกินกว่า 8 ตัวอักษร
- หากทำการcompress file ด้วย Algorithm LZW จะไม่สามารถ decompress file ในกรณีที่ File นั้น มีความแตกต่างของอักขระจำนวนมาก

1.7 ประโยชน์ที่คาดว่าจะได้รับ

- สามารถทำการ compress และ decompress ข้อมูลได้ถูกต้อง
- สามารถทำการส่งข้อมูลผ่าน TCP/IP ได้ถูกต้องครบถ้วน

1.8 เอกสารอ้างอิง

- การเขียนโปรแกรมบน Microsoft Windows
- ระบบปฏิบัติการ Window '98 ชั้นสูง
- การจัดการรับส่งข้อมูลผ่าน TCP/IP
- การบีบอัดข้อมูล (Compression)

1.9 อุปกรณ์ที่ใช้ในการทำปัญหาพิเศษ

1. เครื่องคอมพิวเตอร์จำนวน 2 เครื่อง
   - ขั้นต่ำ Pentium 150 MHz
   - Ram 32 MB
   - HardDisk Drive อย่างต่ำ 1.2 GB
   - ระบบ Multimedia
   - Operating System Platform Window 95/98
2. Modem
3. Ethernet card 2 อัน

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 Compression

Compression คือ การบีบอัดข้อมูลที่ต้องการให้มีขนาดเล็กลง โดยที่ข้อมูลยังคงความถูกต้องเหมือนเดิม เพื่อประโยชน์ในการประหยัดพื้นที่ในการเก็บข้อมูลนั้น และเมื่อข้อมูลมีขนาดเล็กลงแล้ว เมื่อต้องการจะโอนถ่ายข้อมูลก็จะทำได้สะดวกรวดเร็วยิ่งขึ้น

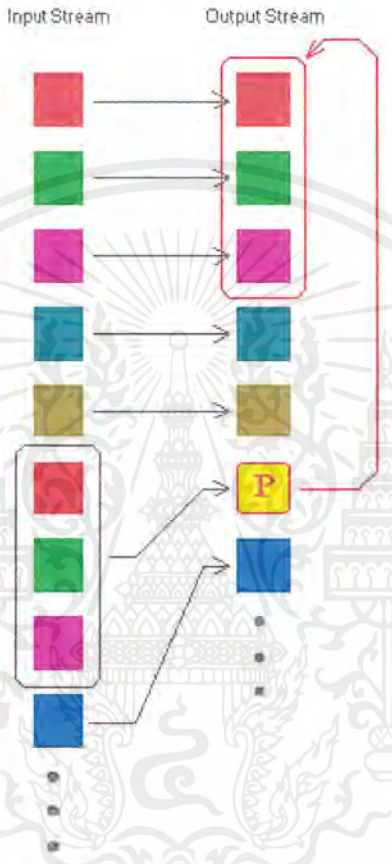
ในปัจจุบัน การ Compression มีการใช้กันอย่างแพร่หลาย ในการทำงานหลายๆด้าน โปรแกรมที่ใช้ในการ Compression ก็มีหลายโปรแกรม เช่น ARJ, LHA, PKZIP (DOS), WINZIP (WINDOWS) โดยแต่ละโปรแกรมก็มีประสิทธิภาพในการบีบอัดไม่เท่ากันนั่นคือ บีบอัดข้อมูลเดียวกันได้ขนาดเล็กไม่เท่ากัน และใช้เวลาทำงานไม่เท่ากัน ทั้งนี้ ขึ้นอยู่กับชนิดข้อมูลที่จะบีบอัดด้วย เช่น ถ้าเป็น text file อาจบีบอัดได้จนเหลือขนาดเพียง 9-15% ของขนาดเดิม แต่ถ้าเป็น file .GIF อาจบีบอัดได้เหลือถึง 90% ของขนาดเดิม นั่นเป็นเพราะ file ชนิด .GIF เป็น file ที่ถือว่า ถูกบีบอัดมาอยู่แล้วเมื่อเวลาทำ เป็นต้น

อย่างไรก็ตาม สิ่งสำคัญที่สุด นั่นคือ algorithm ที่ใช้ในการบีบอัดข้อมูลนั่นเอง algorithm ที่ใช้ในการบีบอัดข้อมูลมีอยู่หลายทฤษฎี เช่น LZ77,LZ78,Shannon-Fano เป็นต้นบางโปรแกรมก็มี algorithm ของตัวเอง แต่ algorithm ที่ใช้กันอย่างแพร่หลายในโปรแกรมส่วนใหญ่และถือเป็นมาตรฐาน คือ Huffman Coding ซึ่งมีการนำไปใช้ในหลายๆระบบงาน อย่างไรก็ตาม algorithm ทุกตัวต่างก็มีจุดมุ่งหมายเดียวกันนั่นคือ ทำให้บีบอัดข้อมูล ได้เหลือน้อยที่สุดเท่าที่จะเป็นไปได้ โดยไม่มีการสูญเสียความถูกต้องของข้อมูล

2.1.1 LZ Coding

LZ เป็น algorithm ในการบีบอัดข้อมูลตระกูลหนึ่ง ซึ่งมีการพัฒนาออกมาเป็นหลาย algorithm โดยได้รับการพัฒนาต่อๆกันมาเรื่อยๆจากตัวแรกคือ LZ77 ซึ่ง algorithm ตระกูลนี้ จัดอยู่ในประเภท "dictionary" การ Compression วิธีนี้ใช้คุณสมบัติของ ความหลากหลายของ ข้อมูลที่ว่าในแต่ละชุด ข้อมูลต่างก็มีข้อมูลที่มีส่วนซ้ำๆกันบ้าง ไม่มากก็น้อย วิธีนี้จึงใช้การเก็บชุด ของ string หรือ characters ซึ่งมีอยู่ซ้ำๆกันในชุดข้อมูลต้นทางที่ต้องการจะ บีบอัดไว้ แล้วนำไปใช้แทนเป็น code เพื่อลดขนาดชุดข้อมูลปลายทาง เมื่อมีการพบ string หรือ character นั้นอีกครั้งในภายหลัง ก็ใช้ code แทนลงไปในชุด ข้อมูลปลายทาง

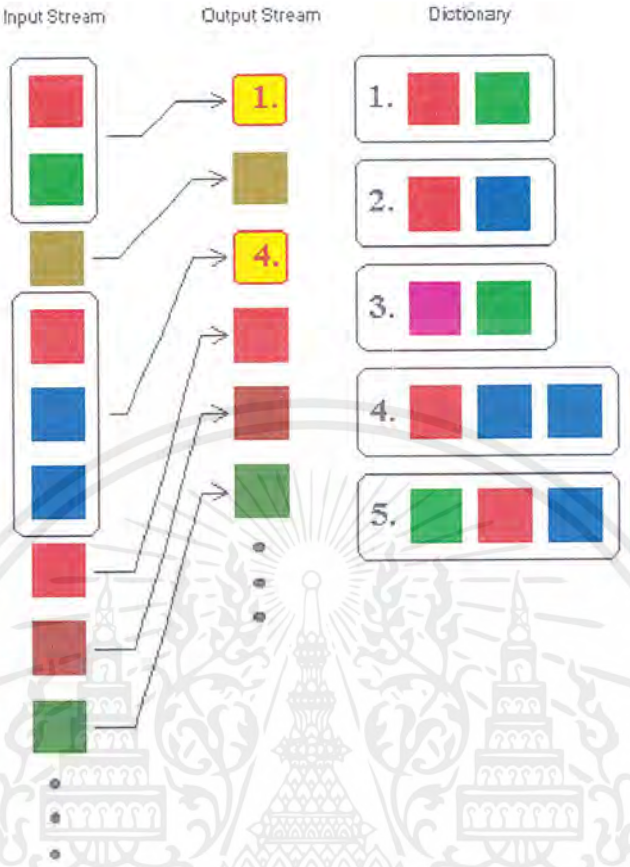
สำหรับ algorithm แบบ dictionary นั้น จะแบ่งได้เป็น 2 กลุ่มใหญ่ๆ ในกลุ่มแรกจะพยายามหา ถ้ากลุ่มของ character ที่กำลังจะบีบอัดนี้ เคยมีอยู่ก่อนหน้านี้มาแล้ว แทนที่ จะใส่ลงในชุด code ที่บีบอัดแล้วไปเป็นกลุ่ม character นั้น ก็จะใช้เป็น pointer ชี้ไปหาส่วน ที่เคยมีมาก่อนแล้วเท่านั้น แสดงได้ดังรูปข้างล่างนี้



รูปที่ 2.1 แสดงวิธีการเข้ารหัสของอัลกอริทึม LZ77

วิธีในกลุ่มแรกนี้ถือว่า เป็นแบบ dictionary โดยนัย ทุก algorithm ในกลุ่มนี้จะพัฒนามาจากตัวเดียวกันคือ LZ77

สำหรับ algorithm ในกลุ่มที่ 2 จะเป็นการสร้างตาราง "dictionary" ของกลุ่ม character ที่มี ปรากฏในชุดข้อมูลต้นทาง เมื่อกลุ่ม character ที่กำลังจะบีบอัด มีอยู่ใน dictionary แล้ว คือ เคยเจอในชุดข้อมูลนี้มาก่อนแล้ว ก็จะใส่ลงในชุด code ที่บีบอัดแล้ว เป็นหมายเลขดัชนีใน dictionary ของกลุ่ม character นั้น ดังที่แสดงได้ในรูปต่อไปนี้



รูปที่ 2.2 แสดงวิธีการเข้ารหัสของอัลกอริทึม LZ78

ต่อไปนี้จะกล่าวถึงทฤษฎีของ LZ พื้นฐาน 2 ตัว ของทั้ง 2 กลุ่ม

2.1.1.1 LZ77

LZ77 เป็น algorithm ตัวแรกในตระกูล LZ ซึ่งเป็นพื้นฐานของ algorithm อื่นๆในกลุ่มแรกนี้ นำไปปรับปรุง ใน LZ77 จะมีส่วนที่สำคัญอยู่ 3 ส่วน นั่นคือ

Code_position หมายถึง ตำแหน่งปัจจุบันที่กำลังพิจารณาในชุดข้อมูลต้นทาง เพื่อนำไปเข้ารหัส

Window หมายถึง ตำแหน่งตั้งแต่ตำแหน่ง Code_position ย้อนกลับไปทางต้นชุดข้อมูลต้นทาง ซึ่งมีความยาว W เป็นชุด character ที่เก็บไว้ใช้เปรียบเทียบเพื่อเป็นต้นแบบสำหรับการเปรียบเทียบ

Look_ahead_buffer หมายถึง ตั้งแต่ตำแหน่ง Code_position จนกระทั่งถึงสิ้นสุดชุดข้อมูลต้นทาง เป็นส่วนที่เรากำลังจะพิจารณานำมาเข้ารหัส

ในการทำงานจะเป็นการหาWindowที่ยาวที่สุดที่มี string ตรงกับใน Look_ahead_buffer ตั้งแต่ตำแหน่ง Code_position และCode ที่จะใส่ลง Output นั้น จะแทนเป็น pointer ที่ชี้ไปสู่ตำแหน่งของ string ที่ตรงกัน ว่าตรง Window ที่ตรงนั้นต้องย้อนขึ้นไป อีกกี่ตัวอักษร แล้วตามด้วย character ตัวแรกที่ไม่ตรงกับใน Window จาก Look_ahead_buffer

การเข้ารหัส

1. ตั้งตำแหน่ง Code_position ที่จุดเริ่มของชุดข้อมูลต้นทาง
2. หาส่วนที่ตรงกันให้ยาวที่สุดใน Window กับใน Look_ahead_buffer
3. ผลลัพธ์ที่ได้ที่จะเป็น Code จะออกมาเป็นคู่ (P,C)
   - P จะใช้ชี้แทนตำแหน่งที่ตรงกันของ Window
   - C เป็น character ตัวแรกใน Look_ahead_buffer ที่ไม่ตรงกัน
4. ถ้า Look_ahead_buffer ไม่ว่าง (empty) อยู่ ให้เลื่อน Code_position และ Window ไปข้างหน้า L+1 ตัวอักษร โดย L คือ ความยาวของ string ส่วนที่ตรง และทำ ซ้ำขั้นที่ 2

ผลลัพธ์ที่จะเป็นใส่ในชุดข้อมูลปลายทางที่บีบอัดแล้ว จะใส่ในรูป (B,L)C โดยที่

- (B,L) คือ P ที่เป็น pointer ชี้ไปส่วนที่ตรง จะเป็นเหมือนคำสั่งเวลา decode ว่าให้ย้อนกลับไปเป็นจำนวน B ตัวอักษรใน Window และ copy L ตัวอักษรใส่ในชุดข้อมูลปลายทาง
- C คือ ตัวอักษรที่เป็นตัวแรกที่ต่อจาก string ส่วนที่ตรง

ตัวอย่าง

ตำแหน่ง	1	2	3	4	5	6	7	8	9
ตัวอักษร	A	A	B	C	B	B	A	B	C

ขั้นที่	ตำแหน่ง	ส่วนที่ตรง	อักษรต่อไป	output
1	1	-	A	(0,0)A
2	2	A	B	(1,1)B
3	4	-	C	(0,0)C
4	5	B	B	(2,1)B
5	7	AB	C	(5,2)C

การถอดรหัส

สำหรับ Window จะใช้วิธีเดิมกับที่เข้ารหัส แต่ละ step จะอ่านเป็นคู่ (P,C) จากชุด ข้อมูลต้นทางที่บีบอัดมา แล้วผลที่ได้ จะเป็น string ที่เป็นลำดับจาก Window ในส่วนที่ P ชี้แล้ว จึงตามด้วย C

จากการทดสอบ ratio จะดีเมื่อชุดข้อมูลที่จะบีบอัด มีข้อมูลหลากหลายรูปแบบ แต่จะ ใช้เวลาค่อนข้างมากในการเข้ารหัส เพราะต้องมีการเปรียบเทียบมากระหว่าง Window กับ Look_ahead_buffer ส่วนอีกด้านในการถอดรหัสจะง่ายและเร็ว จะใช้ Memory น้อยทั้งการเข้าและถอดรหัส ส่วนที่อยู่ใน Memory จะมีแค่ Window ซึ่งปกติใช้แค่ 4-64 K

2.1.1.2 LZ78

เป็น algorithm ที่พัฒนาอีกรูปแบบนอกจาก LZ77 จะมีคำต่างๆที่นิยามไว้ดังนี้

Charstream ลำดับของข้อมูลที่จะบีบอัด

Character ข้อมูลพื้นฐานใน Charstream

Prefix ลำดับ Character ซึ่งนำหน้า Character หนึ่ง

String Prefix ที่ต่ออยู่กับคู่ Character ของมัน

Code word ข้อมูลพื้นฐานใน Codestream ที่ใช้แทน String จาก Dictionary

Codestream ลำดับ Code ที่เข้ารหัสแล้ว รวมทั้ง Character ด้วย

Dictionary ตาราง String ทุก String จะมี Code word ที่ใช้แทนเรียงตามลำดับ

Current Prefix Prefix ปัจจุบันที่จะมาทำในการเข้ารหัส แทนด้วย P

Current Character Character ที่จะตัดสินในการเข้ารหัส ปกติจะเป็น Characterที่นำหน้าโดย Prefix ปัจจุบัน

Current Code Word Code word ที่กำลังทำในการถอดรหัส แทนด้วย W และ String ซึ่งแทนด้วย W

การเข้ารหัส

เริ่มด้วยใน Dictionary จะว่างเปล่าก่อน เราจะมาดูกันตอนมี String บ้างแล้ว เมื่อเริ่มวิเคราะห์ Prefix ใหม่ ใน Charstream เริ่มต้นที่ Empty Prefix ถ้า Prefix + Character (P+C) ตรงกับใน Dictionary Prefix จะถูกรวมกับ C การรวมจะทำจนกระทั่ง เราได้ String ที่ไม่ตรงกับใน Dictionary แล้ว ถึงจุดนั้น เราจะ output ออกไป 2 อย่างใน Codestream คือ Code word ที่แทน Prefix P และ Character C แล้วเราจะเพิ่มทั้ง String P+C ใน Dictionary และเริ่มทำPrefix ต่อไปใน Charstream ถ้าใน Dictrionary ไม่มีอะไรเลย(ในตอนเริ่มทำ) เราจะได้ output ออกเป็นCode word พิเศษ ซึ่งแทน Empty ใน Dictionary และ Character นี้จะใส่ลง Dictionary ด้วย

output จาก algorithm จะเป็นลำดับ Code word ที่เป็นคู่ (W,C) แต่ละครั้งคู่ที่เป็น output String จาก Dictionary จะถูกแปลงเป็น W ที่ C และ string ที่เป็นผลลัพธ์จะใส่ลง Dictionary

Algorithm

1. Dictrionary และ P จะ Empty
2. C = Character ใหม่ใน Charstream
3. P+C มีใน Dictionary หรือไม่
   - ถ้ามี P = P+C
   - ถ้าไม่มี
      1. output เป็น 2 ตัวใส่ Codestream คือ Code word ที่แทน P และ C ที่เป็น Character จาก Charstream
      2. เพิ่ม String P+C ใน Dictionary
      3. P = Empty
4. มี Character ใน Charstream อีกไหม
   - ถ้ามี ย้อมกลับไปทำขั้นที่ 2
   - ถ้าไม่มี
      1. ถ้า P ไม่ Empty ใส่ Code word แทน P ลงใน Codestream
      2. จบการทำงาน

การถอดรหัส

เริ่มที่ Dictionary Empty แต่ละ Step คู่ Code word กับ Character(W,C) จะถูกอ่านจาก Codestream Code word จะแทน String ปัจจุบันที่แทนใน Dictionary String W และ C จะเป็น output ให้ Charstream และ String W+C รวมกันจะใส่ใน Dictionaryหลังการถอดรหัส แล้ว Dictionary ที่ได้จะเหมือนหลังการเข้ารหัส

Algorithm

1. เริ่มทำ Dictionary Empty
2. W = Code word ตัวต่อไปใน Codestream
3. C = Character ที่ตามหลัง W
4. Output ออกไปเป็น String W และ Character C
5. เพิ่ม String W+C ลงใน Dictionary
6. มี Code word ใน Codestream อีกหรือเปล่า

- ถ้ามี ทำขั้นที่ 2 ใหม่
- ถ้าไม่ จบการทำงาน

ตัวอย่าง

ตำแหน่ง	1	2	3	4	5	6	7	8	9
Character	A	B	B	C	B	C	A	B	A

ขั้นที่	ตำแหน่ง	Dictionary	output(W,C)
1	1	A	(0,A)
2	2	B	(0,B)
3	3	BC	(2,C)
4	5	BCA	(3,A)
5	8	BA	(2,A)

จากการทดสอบ LZ78 จะลดจำนวนการเปรียบเทียบในแต่ละขั้นการเข้ารหัส ratio ที่ได้ใกล้เคียง LZ77 และใช้แพร่หลายต่อใน LZW

สำหรับ LZW นั้น จะมีข้อแตกต่างกับ LZ78 ดังนี้

- จะมีแค่ Code word เท่านั้นที่ออกเป็น output นั่นหมายความว่า ใน Dictionary จะไม่เป็น empty แต่จะมี character ที่มีอยู่ได้ในชุดข้อมูล นั้นทั้งหมด อยู่เป็นตัวเดี่ยวๆใน dictionary
- Prefix ที่พิจารณาจะเป็น character เดียว ดังนั้นตอนเริ่มทำที่จะหาใน dictionary จะเป็นการเปรียบเทียบกับ 2 character
- Character ที่จะกลายเป็น Prefix ใหม่ ในการเข้ารหัสครั้งต่อไป คือเป็น character ตัวสุดท้ายของ string ครั้งที่แล้ว นั่นคือ C
- พิจารณา Prefix ทีละ character โดยถือว่า เป็นอักษรสุดท้ายจาก string ตัวหน้า character นั้น

2.1.2 การเข้ารหัสพื้นฐาน ด้วยวิธี Huffman

Huffman Coding จะใช้ code สั้นๆในการแทนที่รหัส ASCII ของ character แต่ละตัวใน file โดยมีหลักการสำคัญอยู่ที่ว่า character ใดที่มีใน file นั้นมาก จะใช้จำนวน bit ของ code ที่ใช้แทนน้อย แต่ถ้า characterใดมีน้อย ก็จะใช้จำนวน bit ของ code ที่ใช้แทนนั้นมากขึ้น เพราะวิธี Huffman ยึดถือว่า การลดขนาดลงนั้น ขึ้นอยู่กับโอกาสที่จะมี character นั้นเกิดขึ้นอยู่ด้วย ซึ่งเป็น

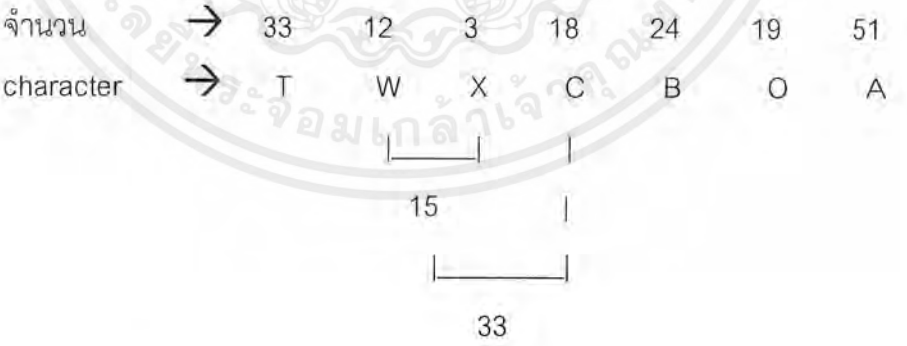
การลดการใช้ code เดิมของ character ที่เป็น ASCII ซึ่งใช้ 8 bit เสมอ แต่ถ้าแทนด้วย code ตามวิธีของ Huffman แล้ว ก็จะใช้จำนวน bit น้อยลง เพราะอาจใช้แค่ 2-3 bit เท่านั้น

ในการเข้ารหัส จะทำตามขั้นตอนดังนี้

- หาว่าข้อมูลทั้งหมดนั้น มี character ที่ไม่ซ้ำกันอยู่ทั้งหมดกี่ตัว
- หาจำนวนที่แต่ละ character มีอยู่ในข้อมูล ว่ามี character ละกี่ตัว
- เลือก 2 character ที่มีอยู่น้อยที่สุดออกมาก่อน แล้วเอาจำนวนที่มีในข้อมูลของแต่ละ character มารวมกัน แล้วถือเป็นค่าของ node ใหม่ เช่น

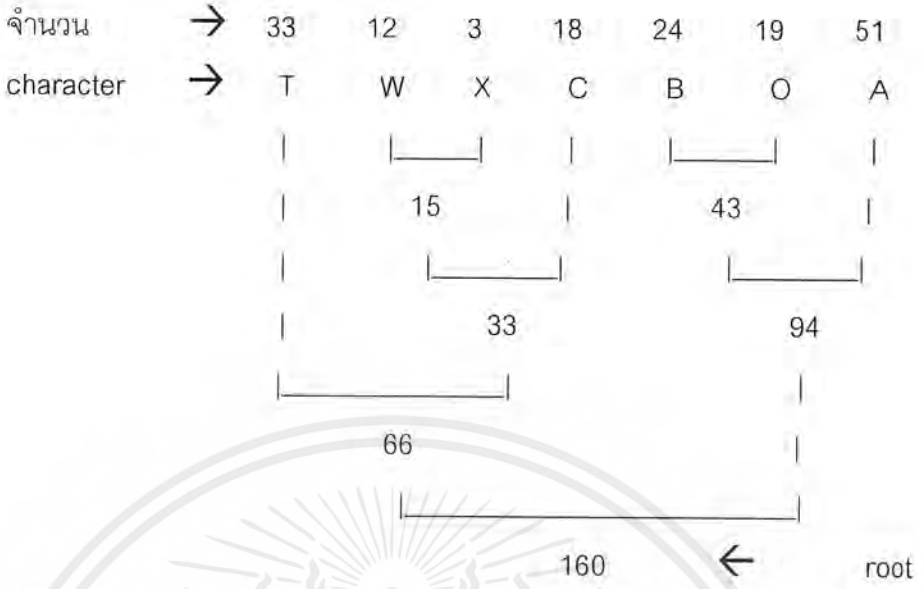
ถ้าในข้อมูลทั้งหมด มี T 33 ตัว, W 12 ตัว, X 3 ตัว, C 18 ตัว, B 24 ตัว, O 19 ตัว, A 51 ตัว ก็จะเลือกเอา W และ X จะได้ผลดังนี้

- นำ node ใหม่ที่ได้ไปเปรียบเทียบ กับจำนวน character ของตัวอื่นๆที่เหลือ โดยใน character ที่เหลือ หาตัวที่มีจำนวนน้อยที่สุด แต่มากกว่า ค่าของ node ใหม่ ในที่นี้คือ C



จากนั้นทำตามขั้นตอนนี้ไปเรื่อยๆ

- หากไม่สามารถหา character ที่มีจำนวนมากกว่าจำนวนของ node ใหม่แล้ว ก็เลือกอีกคู่ของ character ที่มีจำนวนน้อยที่สุดออกมา ในรูปคือ เราทำมาถึง 66 แล้ว แต่ตัวที่เหลือไม่มีมากกว่าแล้ว ก็เริ่มจับคู่ใหม่ นั่นคือ เริ่มที่ B และ O



- เมื่อทำจนครบข้อมูลทุกตัวแล้ว เราก็จะได้ tree ที่เอาไว้ตีค่า code ของแต่ละ character โดยเวลาทำ code ก็จะไล่ไปตาม tree จาก root ไปจนถึง leaf แต่ละตัวที่เป็น character โดยการตีค่าจะคิดจาก การที่ต้องไล่ไปตามกิ่ง tree ถ้าต้องไปทางซ้าย จะได้ค่า 0 ถ้าต้องไปทางขวา จะได้ค่า 1

เช่น
T : 00
W : 0100
X : 0101
C : 011
B : 100
O : 101
A : 11

ด้วยวิธีนี้เราสามารถประหยัดเนื้อที่ในการเก็บ Character ไปได้ จากเดิม มีทั้งหมด 160 character เราต้องใช้ทั้งหมด 8*160 = 1290 bits แต่หลังการทำ compression แล้ว เราใช้แค่ 411 bits เท่านั้น เราสามารถบีบอัดข้อมูลได้เหลือเพียง 32% ของขนาดเดิม แต่วิธีในการขยาย ข้อมูลที่ถูกบีบอัดมานั้น วิธีเดียวที่จะรู้ว่า รหัสนี้เป็น character ใดก็คือ เราต้องส่งตัว tree ไปกับ ข้อมูลด้วย ไม่เช่นนั้น ขยายออกมาไม่ได้ข้อมูลเดิม

สำหรับเวลาจะขยายข้อมูลที่ถูกบีบอัดนั้น ก็จะรับ code ที่เป็นข้อมูลที่ถูกบีบอัดมา จากนั้นก็จะไล่ไปทีละ bit ว่า เป็น 0 หรือ 1 เมื่ออ่านมา 1 bit ก็จะไล่ไปตาม tree ไปด้วย เมื่อไล่ไปจน

ถึงส่วน leaf นั่นคือ code ส่วนที่ไล่มาเป็นของ character หนึ่ง ก็จะส่งออกไปให้ output (อาจจะเป็นหน้าจอ, file) ว่าเป็น character ที่อยู่ที่ leaf ที่เจอ จากนั้นก็จะย้อนไปหาที่ root เพื่อเริ่ม ไล่ตาม code ที่เหลือต่อไป เมื่อไล่ไปเรื่อยๆ ก็จะได้ character ออกมาทีละตัว จนครบ

ตัวอย่าง

หากเราต้องการที่จะบีบอัดข้อความต่อไปนี้

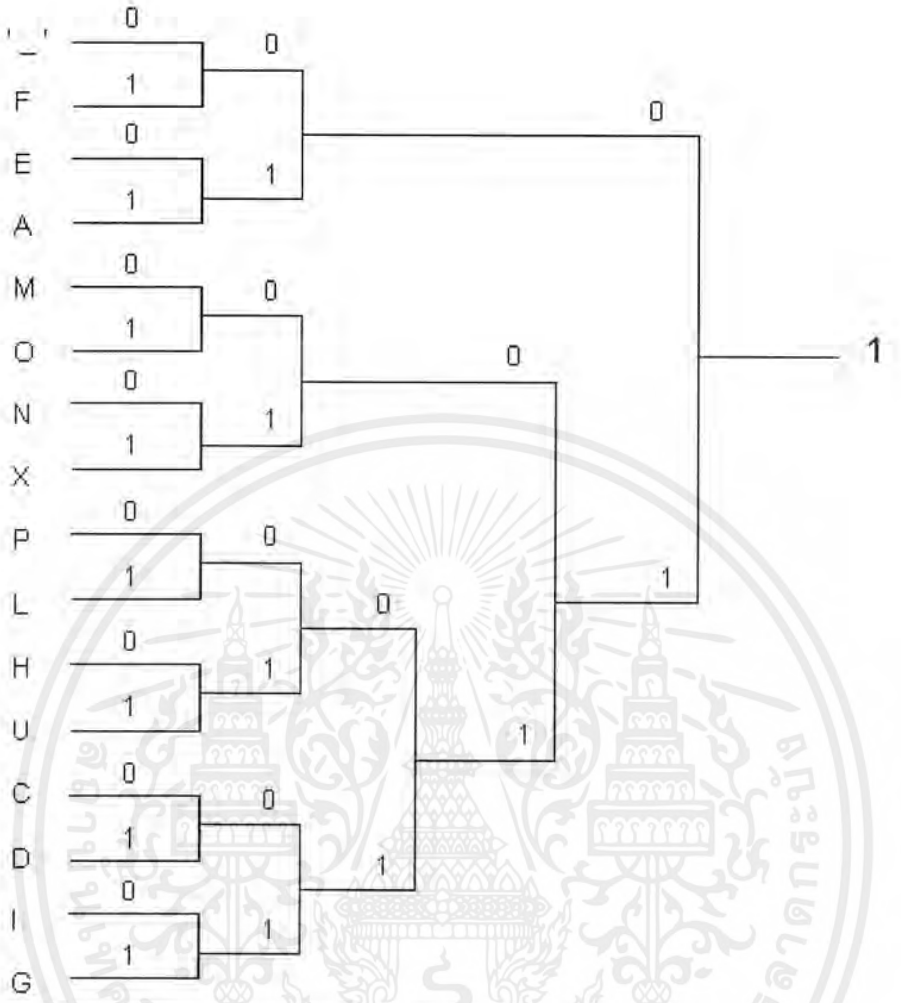
"EXAMPLE OF HUFFMAN CODE"

ขั้นแรก ให้ดูที่ความน่าจะเป็นของ character แต่ละตัว จะได้ออกมาดังตาราง ในต่อไปนี้

ตารางที่ 2.1 แสดงสัญลักษณ์และความน่าจะเป็น

Symbol	Probability
E	2/25
X	1/25
A	2/25
M	2/25
P	1/25
L	1/25
O	2/25
F	2/25
H	1/25
U	1/25
C	1/25
D	1/25
I	1/25
N	2/25
G	1/25
Space	3/25

เราจะได้ tree ในหน้าต่อไป



รูปที่ 2.3 แสดงรูปแบบ tree ที่ได้จากตัวอย่างตารางที่ 2.1

จะเห็นได้ว่าcharacter ที่มีโอกาสจะมีอยู่หรืออยู่ในชุดข้อมูลนั้นสูง เช่น E,F,A จะมี code ที่แทนออกมาได้ โดยใช้เพียงแค่ 3 bit เท่านั้น แต่ character ตัวอื่นๆนั้นโอกาสที่มีอยู่ น้อยกว่าจึงต้องใช้ code มากกว่า

Huffman Coding ถูกนำไปใช้ในโปรแกรม compression ทั่วไป, เครื่อง FAX, หรือในการทำ file ชนิด JPEG Huffman Coding ที่กล่าวมานั้นเป็นแค่พื้นฐาน ใน algorithm ของการ compression บางโปรแกรมมีการนำ Huffman Coding ไปเพิ่มเติมกับ algorithm อื่นๆอีก เพื่อให้ได้ประสิทธิภาพที่ดีขึ้น

Decompressionคือ การขยายข้อมูลที่ถูกบีบอัดมาแล้ว ให้กลายเป็นข้อมูลที่นำไป ใช้งานได้จริง ซึ่งการ decompression ก็ขึ้นอยู่กับว่า compression มาด้วย algorithm ใดหรือโปรแกรมใด เพราะต่างมีวิธีการเข้ารหัสหรือให้ code ต่างกัน หรืออาจเพิ่มส่วนประกอบ ที่ซับซ้อนไม่เหมือน

กันเข้าไปในการให้ code ด้วย โปรแกรมในการ Compression ต่างๆจะมี ส่วนโปรแกรมสำหรับทำ Decompression ไว้อยู่แล้ว

2.2 Transmission Control Protocol / Internet Protocol (TCP / IP)

TCP/IPเป็นเทคโนโลยีในการสื่อสารข้อมูลที่กำเนิดขึ้นจากความต้องการที่จะพัฒนาระบบการสื่อสารของ กระทรวงกลาโหม ประเทศ สหรัฐอเมริกา ซึ่งผู้ที่เป็นคนออกแบบ Core Protocol ชนิดนี้คือ Vinton G. Cerf and Robwert E. Kahn ซึ่งได้ใช้เวลาในการพัฒนาจาก ARPANET เป็นเวลา 3 ปี ลักษณะที่เป็นจุดเด่นของ TCP/IP คือ สามารถที่จะทำให้account ที่เข้ามาในระบบ network นั้น คงอยู่ได้นานและ ยังถูกออกแบบมาให้เป็นอิสระจากการทำงานของ Host Hardware หรือ Operating System

TCP/IP สามารถที่รวมNetworks ที่เป็น Academic Network และCommercial Network เข้าไว้ด้วยกันได้และ ได้ทำการเชื่อมต่อกับ ARPANET ซึ่งเป็นเครือข่ายที่มีอยู่เดิม จึงทำ ให้เกิดเป็น Super network ซึ่งถูกเรียกว่า Internet

2.2.1 TCP / IP Suite Service

TCP/ IP มีจุดเด่นอีกด้านและเป็นเหตุผลที่ทำให้ โปรโตคอล นี้เป็นที่นิยม คือ สามารถที่จะเชื่อมต่อระหว่าง Local Network ที่แตกต่างกันเข้าไว้เป็นเสมือน Network เดียวกันได้

2.2.2 Application – to – Application Communication

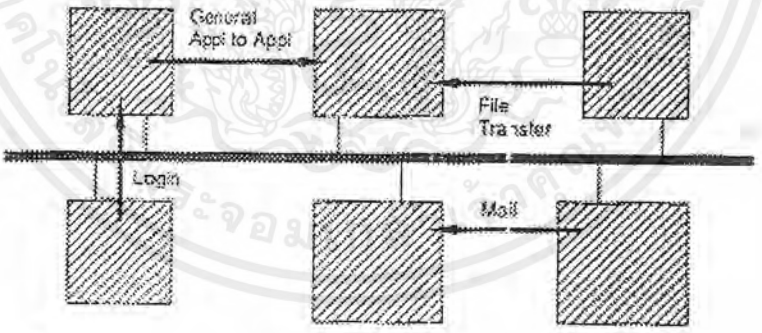
ในการทำงานระหว่าง Application ที่ มีหน้าที่ในการเชื่อมต่อกันระหว่าง network นั้น ได้จัดให้มีลักษณะในการ communicate ได้ 2 รูปแบบ คือ

- Connection – oriented communication เป็นรูปแบบมาตรฐานในการติดต่อสื่อสาร ซึ่งมักจะใช้ในการส่งข้อมูลแบบ เครื่อง กับเครื่อง (peer to peer) เช่น การส่ง file มีการทำงานหลัก คือ
    - Setup Connection เป็นการระบุต้นทาง และ ปลายทางของข้อมูลที่ต้องการสื่อสาร
    - Send ฝ่ายส่งทำการส่งข้อมูลที่ใช้ในการสื่อสารไปยังฝ่ายรับข้อมูล
    - Receive ฝ่ายรับข้อมูลต้องทำการรับและส่งสัญญาณ ในการรับและเมื่อรับเสร็จเรียบร้อย
    - Disconnection ทำการยกเลิกการติดต่อที่ถูก set up ขึ้นมาทุกครั้งเมื่อการสื่อสารเป็นที่เรียบร้อย

- Connectionless communication เป็นรูปแบบอีกอย่างหนึ่งที่เป็นมาตรฐานในการติดต่อสื่อสารซึ่งมักจะใช้ในลักษณะค้นหาข้อมูลบนแหล่งข้อมูลเครือข่าย ซึ่งมีการทำงานหลัก คือ
  - Create เป็นการสร้างเส้นทางขึ้นมาจากต้นทางไปสู่ปลายทาง
  - Transmit เป็นการส่งข้อมูลที่ต้องการค้นหาเข้าไปยังแหล่งข้อมูลเพื่อทำการค้นหา
  - Receive เป็นการรับข้อมูลหรือ message ที่ส่งมาจาก แหล่งข้อมูล

TCP/IP ได้ถูกสร้างมาและได้มีการจัดเตรียมให้มีการให้มีการบริการที่เป็นพื้นฐานของการติดต่อระหว่าง Local Network ต่างที่ใช้ TCP/ IP เป็น โปรโตคอล คือ

- File Transfer เป็นความสามารถที่นำเอา สิ่งที่เราสนใจใน file จากระบบหนึ่งไปใช้งาน หรือ เก็บเป็นข้อมูลในอีกระบบหนึ่งได้ โดยส่งผ่าน TCP /IP
- Terminal Access เป็นการอนุญาติให้ Single Terminal (Telnet) สามารถเชื่อมต่อกับ HOST ตัวใดตัวหนึ่งที่ติดตั้งอยู่ในระบบเครือข่าย
- Electronic Mail หรือ e-mail เป็นการสื่อสารทางจดหมายซึ่งถูกกำหนด รูปแบบมาตรฐานของจดหมายไว้ และ อาศัยกลไกของ SMTP ในการส่งโดยตรงหรือ ส่งต่อไปยังผู้รับอื่น ๆ ได้



รูปที่ 2.4 Basic services on a network

2.2.3 TCP / IP Architecture

TCP/IP เป็น โพรโตคอล ที่ถูกสร้างเพื่อติดต่อสื่อสารระหว่างอุปกรณ์หลาย ๆ ชนิด ที่แตกต่างกัน โดยสามารถทำงานสื่อสารข้อมูลผ่านตัวกลางต่างๆ ที่อยู่ในระบบเครือข่าย และจัดการให้ Network ทุก Network ถูกจัดให้เป็นเพียง Network เดียวคือ Internet เพื่อที่จะสามารถใช้บริการพื้นฐานที่ TCP/IP ได้จัดเตรียมไว้ให้

ลักษณะการส่งผ่านข้อมูลใน TCP/IP

- ข้อมูลที่จะถูกส่งนั้นจะถูกแยกออกเป็น ส่วน ๆ
- ข้อมูลที่ถูกแยก จะถูกเคลื่อนที่แต่ละส่วนไปยังปลายทาง โดยเป็นอิสระต่อกัน
- การทำงานนี้จะถูกควบคุมด้วย Layer ที่มีหน้าที่ในการจัดการ
- ที่ต้นทางและ ปลายทาง จะมี function ของ อีก Layer หนึ่ง ทำการตรวจสอบและ รับประกัน ข้อมูลที่ถูกส่งมาให้มี ความถูกต้อง และ ครบถ้วน

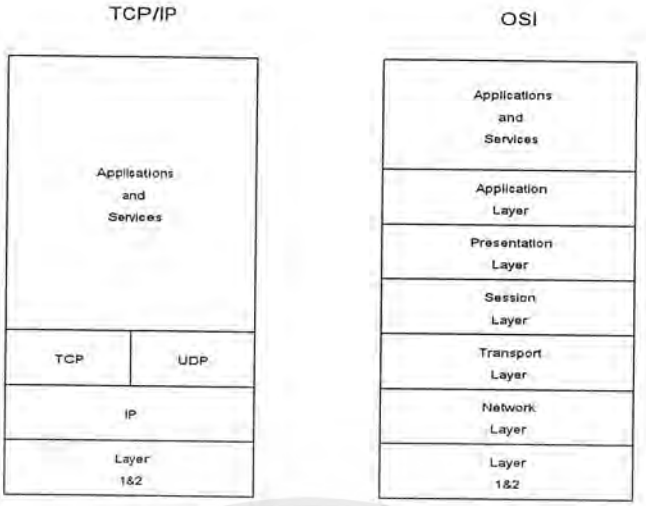
2.2.4 Layering (7 layers)

ในการส่งข้อมูลผ่าน Layer ต่างๆ นั้นแต่ละ Layer ก็ได้มีการจัดนำเอาสิ่งที่จำเป็น สิ่งที่มีความหมายในการบ่งชี้ถึงเส้นทางความถูกต้องของข้อมูล ในการส่ง จัดใส่เข้าไปพร้อมกับ ตัวข้อมูล ซึ่งได้แก่

- Format the data
- Package the data
- Determine the path that the data will follow
- Regulate the rate of data transfer according to the bandwidth and the capacity of the receiver to absorb data
- Transmit the data on a physical medium.
- Assemble incoming data so that it is in sequence and there are no missing pieces.
- Check incoming data for duplicated pieces
- Notify the sender of how much data has been received safely
- Deliver data to the right application
- Handle error or problem events

ในการทำงานของแต่ละ Layer นั้น ได้ถูกมอบหมายให้ตามความต้องการของ Academic and defense communities ซึ่งในส่วนของ IPจะทำหน้าที่ในการเป็นตัวเชื่อมต่อระหว่าง network ที่ต่างกันโดยอาศัยหมายเลข address ส่วนของ TCP นั้นทำหน้าที่รับผิดชอบในการส่งถ่ายข้อมูลให้มีความถูกต้องตามแบบต้นฉบับเดิม

Layer ของ TCP/ IP ได้ถูกจัดทำขึ้นตามแบบ ของ OSI layering (Open System Interconnection layering)เนื่องจาก OSI layering เป็นมาตรฐานในการจัดทำ layer ในส่วนของการติดต่อสื่อสารข้อมูล



รูปที่ 2.5 TCP/ IP and OSI Layers.

OSI layering นั้นสามารถแบ่งได้เป็น 2 ส่วนได้แก่

1. Lower layering ประกอบด้วย Physical layering , Datalink layering , Network layering จะทำงานในระดับ hareware

2. Upper layering ประกอบด้วย Transport layering , Session layering , Presentation layering , Application layering จะทำงานในระดับ software

Lower layer เป็น layer ที่ทำการเชื่อมต่อระหว่าง driver ของ device ซึ่งจะทำการจัดการแบ่งข้อมูลที่ต้องการจะส่งนั้นออก เป็น " frames " or " packets " และทำการส่งข้อมูล นั้นไปยัง Local system ส่งไปยัง ปลายทาง

2.2.4.1 Physical layer (layer 1)

การทำงานในระดับนี้ จะเป็นการทำงานเกี่ยวกับการรับส่งข้อมูลผ่านตัวกลางต่างๆ ในลักษณะ bit ต่อ bit ต่อเป็นสายของข้อมูล

2.2.4.2 Datalink layer (layer 2)

การทำงานในระดับนี้ จะเป็นการกำหนด หรือควบคุมการเชื่อมระหว่างอุปกรณ์สื่อสาร ให้เป็นไปตามข้อกำหนดมาตราฐาน(Protocol) และ รูปแบบการเชื่อมต่อ(Topology)

2.2.4.3 IP and OSI Network layers (Layer 3)

The Internet Protocol (IP) ทำหน้าที่ในการเคลื่อนย้ายข้อมูลระหว่าง host ซึ่งอาจจะเคลื่อนที่ผ่านเพียง 1 Network หรือ อาจจะมากกว่านั้น และ ข้อมูลที่ถูกเคลื่อนที่นั้นจะถูกเรียกว่า " datagrams"

IP นั้นจะเชื่อมต่อในลักษณะของ Connectionless เพราะว่าไม่มีการรับประกันว่า datagrams นั้นจะถูกจัดส่งถึงเป็นที่เรียบร้อย และจะไม่มีการเคลื่อนย้ายจาก Application ไปยัง Application ในขั้นของ IP นี้จะตรงกับ layer 3 ของ โครงสร้าง OSI layer แต่ในโครงสร้าง OSI นั้น ใน Layer นี้ จะมีการให้บริการทั้ง Connectionless และ Connection oriented ด้วย ตัวอย่างเช่น X.25

2.2.4.4 TCP and OSI Transport Layer (Layer 4)

The Transmission Control Protocol (TCP) ในส่วนนี้จะมีการรับประกันของข้อมูลที่จะถูกเคลื่อนที่ให้มีความน่าเชื่อถือได้ เพราะจะต้องคอยให้บริการกับ Application

TCP จะทำการส่งสิ่งที่เป็นข้อมูลจาก TCP ว่า " segment " โดยจะส่งผ่านไปยัง IP ซึ่ง IP จะทำการส่งไปยัง ปลายทาง และ ในทางกลับกัน TCP ก็จะรับ segments จาก IPและทำการตรวจสอบว่าเป็น Segment ของ Application ไหน แล้วจึงทำการผ่านข้อมูล (segment) ตัวนั้นไปยัง Application

ในส่วนของ Transport Layer ของ OSI นั้น ก็จัดให้มีการใช้รูปแบบ Connection – oriented เป็นการรับประกันการส่งผ่านข้อมูลว่าจะไปถึงและถูกต้องแน่นอน ส่วนของ Layer อื่นๆ เช่น Session layer ทำหน้าที่ในการ setup และ terminate ให้ระหว่าง application –to-application

User Datagram Protocol (Layer 4) ในกรณีที่ Application นั้นต้องการส่ง message ไปยัง Application ที่อยู่บนเครือข่ายเดียวกัน จึงก่อให้เกิด การร้องขอ The User Datagram Protocol (UDP) ขึ้น เมื่อ UDP ถูกแยกเป็นหน่วยเพื่อเตรียมตัวส่งนั้นจะถูกเรียกว่า " User Datagrams" ซึ่ง User Datagrams จะถูกส่งต่อไปยังIPซึ่งเป็นการเชื่อมต่อแบบ Connectionless เหมาะสำหรับการค้นหา database

2.2.4.5 Session layer (layer 5)

การทำงานในระดับนี้ เป็นการตรวจสอบความถูกต้องของข้อมูลที่ทำการรับส่ง

2.2.4.6 Presentation layer (layer 6)

การทำงานในระดับนี้ เป็นการจัดเรียงข้อมูลให้ถูกต้อง ทำการเข้ารหัสข้อมูล ถอดรหัส ข้อมูล(encode)

2.2.4.7 Application Service (layer 7)

ในชั้นนี้นั้น TCP/IP จะประกอบไปด้วย กลุ่มของบริการที่เป็นมาตรฐานของ TCP /IP คือ File Transfer Protocol(FTP) , Simple Mail Transfer Protocol(SMTP) , Telnet Terminal

access and Domain Name System (DNS) ซึ่งถ้าเปรียบเทียบกับ OSI แล้วจัดว่าในชั้นนี้ยังคงต้องพัฒนาในรายละเอียดที่ซับซ้อน

2.2.5 ลักษณะของเครือข่ายที่ใช้ในระดับองค์กร

- เครือข่ายเฉพาะบริเวณ(LAN:Local Area Network)

เครือข่ายขนาดเล็กที่เป็นของผู้ใช้กลุ่มเล็กๆ กลุ่มหนึ่งหรือใช้ภายในบริเวณ มีลักษณะเฉพาะที่แตกต่างจากระบบอื่นๆดังนี้

1. มีขนาดเล็ก ใช้ในระยะทางไม่เกิน 2-3 km.
2. เทคโนโลยีที่ใช้ในการรับส่งข้อมูล คือจะต้องมีความเร็วสูง , ง่ายต่อการใช้งาน และเป็นเทคโนโลยีที่เป็นแบบเดียวกันทั้งระบบ
3. รูปแบบการจัดโครงสร้างของระบบ(Topology) จะต้องใช้ Topology แบบเดียวกันทั้งระบบ (ซึ่งถ้ามากกว่า 1 จะทำให้เกิดการขัดแย้งกันในการใช้งาน)
4. ไม่มีการถ่ายโอนข้อมูลข้าม Network จะถ่ายโอนข้อมูลเฉพาะในวง Lan ของตนเอง Lan จะทำงานในส่วนของ Physical Layer กับ Datalink Layer และใช้งาน ได้ปกติ แต่ถ้าใช้งานได้ในส่วนของ Network Layer (ส่งข้อมูลไปให้กับ Network วงอื่นๆ ได้) แสดงว่าเป็น MAN

- เครือข่ายในเขตเมือง(MAN: Metropolitan Area Network)

เป็นการเชื่อมต่อส่งข้อมูลภายในเครือข่าย และสามารถส่งข้ามเครือข่ายกันได้ ซึ่งมีลักษณะเฉพาะดังนี้

1. มีขนาดใหญ่กว่า LAN
2. สามารถมีหลายเทคโนโลยีได้
3. มีหลาย Topology ได้
4. มีการถ่ายโอนข้อมูลระหว่างเครือข่าย หรือ LAN สองวง

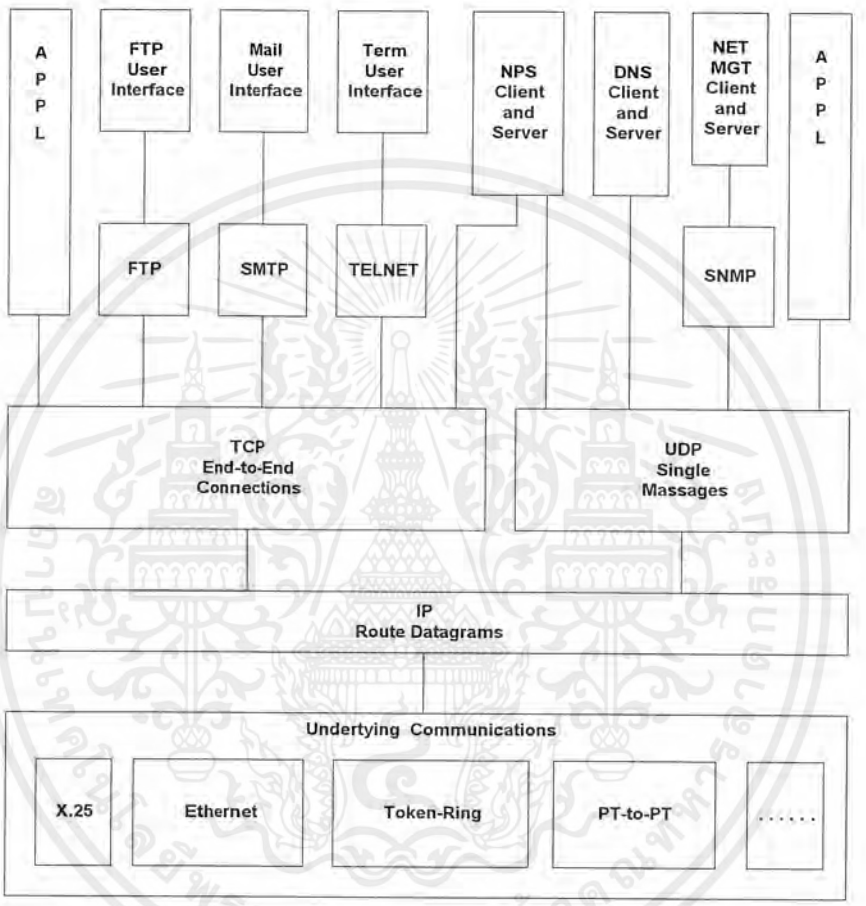
- เครือข่ายวงกว้าง(WAN:Wide Area Network)

เป็นเครือข่ายขนาดใหญ่ ครอบคลุมทั่วโลก มีลักษณะเฉพาะดังนี้

1. มีขนาดใหญ่
2. เทคโนโลยีมีความเร็วต่ำ(เพราะว่าใช้ในเครือข่ายที่มีขนาดใหญ่ ระยะทางไกลมาก) ใช้ในการเชื่อมต่อแบบ point to point ระหว่าง 2 เครือข่ายเข้าด้วยกัน
3. Topology เป็นแบบ point to point
4. มีการถ่ายโอนข้อมูลข้ามเครือข่ายได้

2.2.6 PROTOCOL OVERVIEW

ในการใช้งาน Basic Service ของ TCP/IP นั้นบ่อยครั้งที่เกิดเหตุการณ์ที่ข้อมูลที่ส่ง นั้นไม่เป็นมาตรฐานจึงมีหลาย ๆ vendor นั้นได้เสนอให้ใช้ " The Berkeley Software Distribution UNIX end-user interfaces" ซึ่งจะช่วยในการจัดชุดข้อมูลให้สามารถติดต่อกับ IP ได้สะดวกและเป็นไปในมาตรฐานเดียวกัน

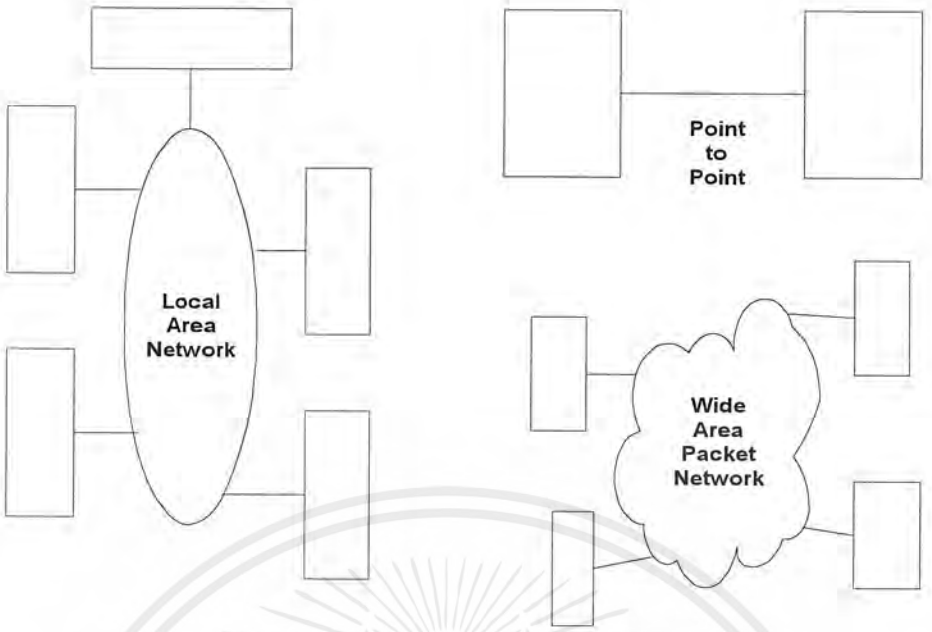


รูปที่ 2.6 Protocol architecture Overview

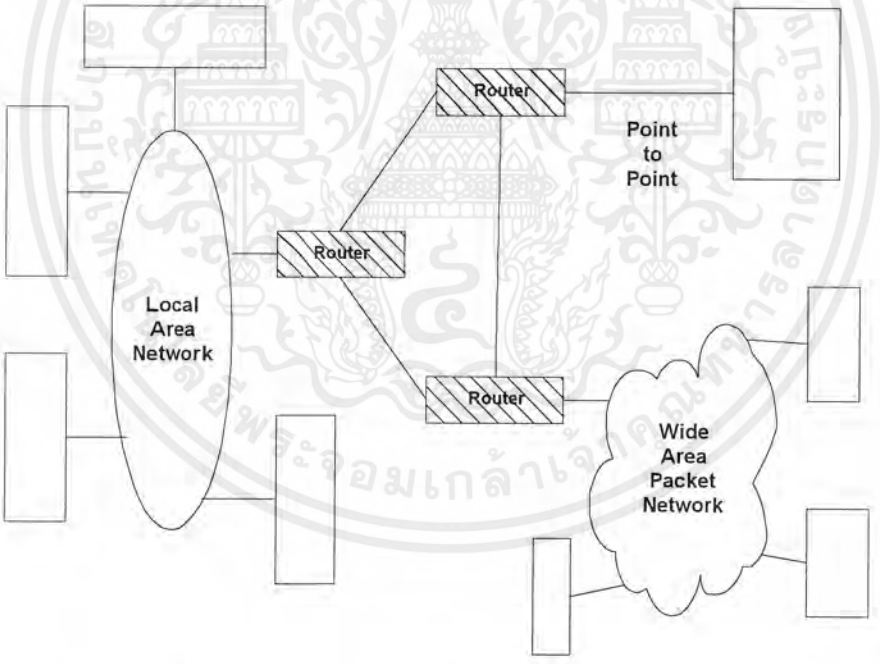
2.2.7 TOPOLOGIES

TCP/IP ไม่ได้ใช้ได้แต่ Internet หรือ complex network TCP/IP ยังสามารถที่จะใช้งานกับ Standalone LANs and WANs ซึ่งในรูปของ 2.7 จะแสดงให้เห็นถึงโครงสร้างต่างๆ ของ แต่ละ topology ที่ไม่มีการเชื่อมต่อกันระหว่าง topology ซึ่งในรูปของ 2.8 จะแสดงการเชื่อมต่อของ แต่ละ topology ด้วย IP routers ซึ่ง IP router จะคอยตรวจสอบสถานะภายในระบบที่ตัวมันเชื่อมต่ออยู่ และคอยที่จะหาเส้นทางที่เหมาะสมให้แก่ Datagram

แต่เดิมจะใช้อุปกรณ์ " gateway " ในการเชื่อมต่อระหว่างแต่ละ topologies กับ เครือข่าย Internet มากกว่าการที่จะนำ IP router มาใช้



รูปที่ 2.7 TCP /IP on standalone LANs and WANs



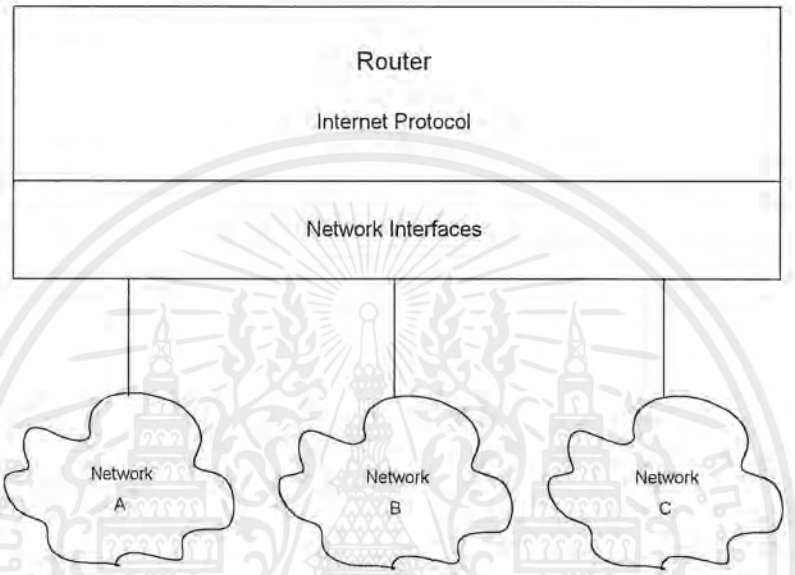
รูปที่ 2.8 Creating an internet with IP routers.

2.2.8 IP ARCHITECTURE

The Internet Protocol software จะถูกสั่งให้ทำงานบนเครื่อง IP hosts และ ใน IP router ซึ่งมองจากรูป 2.9 แล้วจะเห็นว่า มี IP router เชื่อมต่ออยู่กับ Network 3 Networks ซึ่งแน่นอนว่า

การเชื่อมต่อผ่าน router นั้นจะต้อง เชื่อมต่ออย่างน้อย 2 Networks และ จากรูป 2.9 ไม่มีส่วนของ TCP เนื่องจากไม่มีการเชื่อมต่อระหว่าง Application ที่ router

ปัจจุบันนี้ router รุ่นใหม่มักมีประสิทธิภาพสูงขึ้นสามารถเชื่อมต่อกับ Hardware Slots ได้หลายแบบ ทำให้สามารถที่จะรองรับ Configures ได้หลายแบบ อาทิเช่น Ethernet , Token Ring , point-to-point synchronous , fiber optic....



รูปที่ 2.9 IP router protocol architecture

2.2.9 IP Actions

ถ้าหากจุดหมายปลายทางสำหรับ datagram ไม่ได้อยู่บน Network เดียวกันกับ host ที่ทำการส่ง datagram ตัว IP จะทำการส่ง datagram ไปยัง Local router หลังจากนั้น ถ้า router ไม่ได้เชื่อมต่อกับ network ที่เป็นจุดหมาย ตัว router ที่เป็น Local router ก็จะทำการส่ง datagram ไปยัง router ตัวอื่นจนกว่าจะถึงตัว router ที่เชื่อมต่อกับเครื่อง host ปลายทาง

การตัดสินใจของ router ว่าจะส่ง datagram ต่อหรือไม่นั้นก็ขึ้นอยู่กับว่าตัว router นั้นสามารถที่จะหาชื่อ ของ host หรือ entry ใดก็ได้ของ host ที่เป็นข้อมูลเก็บอยู่ใน router table หากมีการ match แล้วตรงกันก็จะหมายถึงว่าได้มาถึง Host ปลายทางแล้ว

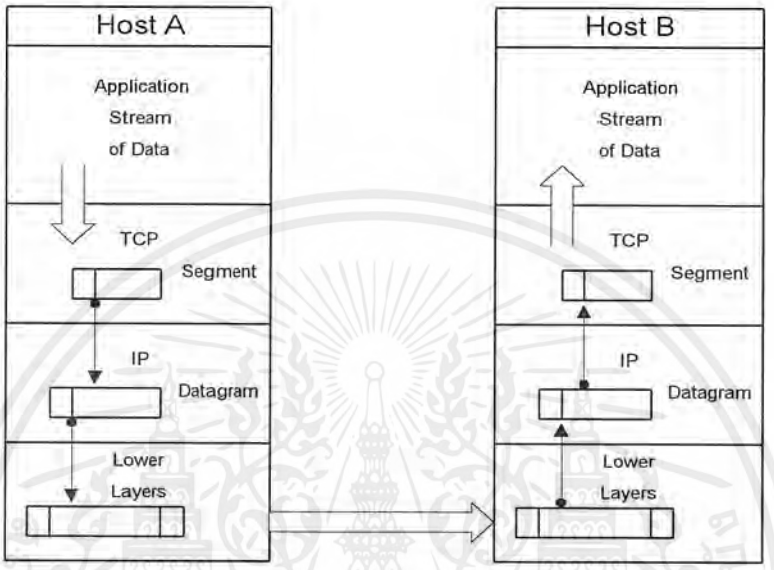
2.2.10 TCP ARCHITECTURE

TCP จะถูกสร้างขึ้น ภายใน Host ที่ต้องการจะส่งข้อมูล จุดสำคัญอยู่ตรงที่ว่าต้องมีการรับประกันว่าข้อมูลที่ส่งไปหรือรับมานั้นมีความถูกต้อง ซึ่งใน TCP ก็จะประกอบไปด้วย

- ความแน่นอน (Accurate)
- ความสมบูรณ์ของข้อมูล (Complete)

- ไม่มีการซ้ำซ้อน สองฉบับหรือมากกว่านั้น (Free of duplicates)

การส่งข้อมูลที่ยังไม่ได้แยก(stream of bytes) ของ TCP นั้น ตัว stream นั้นจะถูก แยกออกเป็นส่วน ๆ และแต่ละส่วนจะถูกเพิ่ม header เข้าไป ซึ่งจะถูกเรียกว่า " Segment " หลังจากนั้นก็จะถูกจัดส่ง ไปยังส่วนของ IP และ ก็จะถูก IP นั้น ส่งเป็น datagram ดังรูป 2.10



รูปที่ 2.10 Packaging a stream of application data

การรับข้อมูลมานั้นสิ่งสำคัญคือความถูกต้องของข้อมูล ซึ่งสามารถตรวจสอบได้โดยการส่ง สัญญาณ Acknowledge (ACK) กลับไปยังเครื่อง host ต้นทาง หากเกินเวลาที่กำหนด(time out) ทาง Host ต้นทางยังไม่ได้รับ สัญญาณ ACK ก็จะทำการส่งข้อมูลชุดเดิมออกมาใหม่อีกครั้ง หลังจากนั้นแล้วตัว host ที่เป็นปลายทางนั้นจะต้อง จัดเรียง Incoming Segment ให้อยู่ในลำดับที่ถูกต้องด้วยและห้ามขาด segment ใดๆ เลย

2.3 Window Sockets

Window Socket คือ ส่วนติดต่อกับ network เสมือนเป็นเส้นทางผ่านระหว่าง microsoft Window network และช่วยให้ application สามารถติดต่อกันผ่าน network ได้ ซึ่งมี network Application มากมายที่สนับสนุน Window Sockets ภายใต้ Protocol ต่างๆ เช่น Transmission Control Protocol /Internet Protocol (TCP/IP) ,Xerox Network System(XNS) , Digital Equipment Corporation's DECNet protocol ,Internet Packet Exchange/Sequence Packet Exchange (IPX/SPX) ,ฯลฯ แต่โดยพื้นฐานแล้ว จะต้องสนับสนุน TCP / IP หรือ Protocol ใดๆ ที่สามารถทำงานร่วมกับ Window Socketได้ โดยการ Implement ไว้ใน DLL(Dynamic Link

Library) ซึ่ง Sockets จะช่วยให้ programmer ไม่จำเป็นต้องทราบว่า network ทำงานอย่างไร เพราะ socket ได้จัดการส่วนนั้นให้หมดแล้ว

Microsoft Foundation Class Library (MFC) สนันสนุนการโปรแกรมกับ window sockets APIโดยมี 2 class ให้ใช้ หนึ่งในนั้น คือ Csocket มันคอยจัดการเกี่ยวกับการเขียนโปรแกรมการสื่อสารข้อมูลบน network

2.3.1 นิยามของ Socket

Socket คือ ปลายทางของการสื่อสารข้อมูล Socket ต้องมี type และต้องรวมอยู่ใน Process ที่กำลังทำงานอยู่ โดยทั่วไป Socket จะสื่อสารข้อมูลกับ Socket ตัวอื่นๆ ที่กำลัง ทำงานอยู่ใน Communication Domain เดียวกันโดยใช้ IP ช่วยสามารถแบ่งได้เป็น 2 ชนิด คือ

2.3.1.1 Socket แบบ Stream

ทำงานกับข้อมูลที่จะส่งไปโดยไม่กำหนดขอบเขตของเรคอร์ด หรือไม่เป็นเรคอร์ด (ข้อมูลที่ส่งเรียกว่า a stream of byte) และ stream จะรับประกันข้อมูลถูกส่งถึงปลายทางอย่าง ถูกต้อง โดยข้อมูลไม่สลับกันและไม่มีการส่งซ้ำซ้อน

2.3.1.2 Socket แบบ Datagram

สนับสนุนการส่งข้อมูลเป็นเรคอร์ดแต่ไม่รับประกันว่าได้ส่งไปถึงและข้อมูลอาจจะมีการสลับกันหรืออาจมีการส่งข้อมูลซ้ำกันได้

หมายเหตุ ในบางเน็ตเวิร์ค เช่น XNS.Stream อาจสนับสนุนการส่งข้อมูลเป็นเรคอร์ค (Stream of record)

2.3.2 The Socket Data Type

Socket ของ MFC จะซ่อนตัว handle ของ Window Sockets ไว้ในแฮนเดิลที่ชื่อว่า Socket เหมือนกับที่ใช้กับ HWND ซึ่งรายละเอียดของ Socket จะมีอยู่ในข้อมูลของ WIN32 SDK ทั่วไป

การใช้ Sockets

Sockets มักถูกใช้ในการสื่อสารข้อมูล 3 ลักษณะดังนี้

- Client/Server models
- Peer-to-Peer scenarios (เช่น Chat Application ต่างๆ)
- การทำ Remote Procedure Call(RPC) โดยใช้แอพพลิเคชันที่รับข้อมูลแปลเมสเสจเป็นการเรียกใช้ฟังก์ชันต่างๆ

2.3.3 Stream Sockets

Stream sockets ใช้ในการส่งข้อมูลในลักษณะที่ไม่เป็นเรคอร์ดซึ่ง Stream of byte นี้สามารถส่งได้ทั้งสองทาง คือการส่งและรับใน socket เดียวกัน Stream นี้สามารถเชื่อถือได้ว่าจะถูกส่ง อย่างถูกต้อง ไม่สลับกันและไม่ซ้ำกัน และส่งไปถึงอย่างแน่นอน โดย Stream ยังสามารถจัดการ ส่งและรับข้อมูลขนาดใหญ่ได้ดีอีกด้วยเพราะโปรโตคอลในเน็ตเวิร์คเลเยอร์อาจแตกข้อมูลให้อยู่ ในขนาดที่สมควรซึ่ง class Csocket จะจัดการแตกและรวมให้เอง

การสื่อสารข้อมูลแบบ Stream นั้นเป็นการสื่อสารข้อมูลแบบ Connection-Oriented Stream นั้นจะต้องมีการเชื่อมต่อกันอย่างชัดเจน ถ้า Socket A ต้องการเชื่อมต่อกับ Socket B และ Socket A จะต้องมีการร้องขอการเชื่อมต่อ ไปยัง Socket B แล้ว Socket B จะต้องยอมรับหรือปฏิเสธการร้องขอนั้น เหมือนกันการโทรศัพท์ : ซึ่งจะต้องมีผู้เรียกแล้วผู้รับจะต้องรับสายจึงจะคุยกันได้ และการสนทนานั้นจะไม่มีเสียงซ้ำกัน ถ้าผู้พูดไม่ได้พูดซ้ำและลำดับของเสียงที่พูดจะ ตรงกับลำดับของเสียงที่ผู้ฟังได้ยินและที่สำคัญที่สุด คือเสียงที่ผู้พูดพูดออกไปนั้นจะถูกส่งไปถึง ผู้รับแน่นอนด้วยเหตุนี้จึงทำให้ Steam socket เป็นที่นิยมเรียกมากว่า Datagram socket

2.3.4 การใช้ Socket กับ Archive

ในส่วนนี้จะอธิบายเกี่ยวกับลักษณะการเขียนโปรแกรมด้วย Class CAsynSocket, Class Csocket ได้ใช้ระบบการ Serialization ของ MFC ในการส่งข้อมูลระหว่าง Object Csocket ผ่าน Class Carchive

2.3.5 Sequence of Operations

ในส่วนนี้จะกล่าวถึงลำดับขั้นตอนการสร้างและสื่อสารข้อมูลระหว่าง socket ทั้งด้าน server และ client

ตารางที่ 2.2 แสดงลำดับขั้นตอนการทำงานของการสื่อสารระหว่าง socket ทั้ง 2 ฝั่ง

	Server	Client
สร้าง socket	Csocket sockServer; SockServer.Create(nPort); //*nPort เป็น UNIT ที่ใช้กำหนดเลข port ที่จะมา connect ด้วย	Csocket sockClient; SockClient.Create();
หา connection		SockClient.Connect(streAddress,nPort); //*strAddress คือ String ที่มีค่า IP Address อยู่

ตารางที่ 2.2 แสดงลำดับขั้นตอนการทำงานของการสื่อสารระหว่าง socket ทั้ง 2 ฝั่ง(ต่อ)

	Server	Client
		*nPort เป็น UNIT ที่ได้กำหนดไว้ตอนที่ server ได้ Create ขึ้น
สร้าง socket ว่างๆ และ Accept connection	CsockReceive; SockServer.Accept(sockReceive);	

ตารางที่ 2.3 แสดงการทำงานในขั้นเตรียมการส่งไฟล์

สร้าง file object	CsocketFile file(&sockServer);
สร้าง archive	Carchive arIn(&file,Carchive::load); หรือ Carchive arOut(&file,Carchive::store); Carchive arIn(&file,Carchive::load); หรือ Carchive arOut(&file,Carchive::store);
ใช้ archive ในการส่งหรือรับข้อมูล	ArIn<<dwValue; หรือ arOut>>dwValue;

2.3.6 Sockets ทำงานกับ Archive อย่างไร

จากตัวอย่าง PacketSerialize เป็นการส่ง object ที่เหมือนกับการ Serialize ใน MFC ทั่วไปจุดแตกต่างกันก็คือข้อมูลที่ได้ถูกเก็บไว้ใน object Cfile หรือ disk ทั่วไปแต่เป็น object CSocketfile ที่เชื่อมกับ socket อีกที

Class CSocketFile สืบทอดคุณสมบัติจาก class CFile แต่ไม่สนับสนุนความสามารถบางอย่างของ CFile (เช่น Seek, GetLenght, SetLenght, ฯลฯ) การเก็บข้อมูลลง CSocketFile จะต้องเก็บเป็น sequence แต่เนื่องจากที่ CSocketFile สืบทอดคุณสมบัติมาจาก CFile มันจึงต้องรับคุณสมบัติ CFile มาทั้งหมด, เพื่อป้องกันปัญหาเหล่านี้ function ของ CFileที่ CSocketFile ไม่สนับสนุนจะถูก override แล้ว throw exception ที่ชื่อว่า CnotSupportedException กลับมาแทน CArchive, CSocketFile, Socket

การทำงานของมันช่วยให้ไม่ต้องจัดการรายละเอียดของ socket ด้วยตัวเองเพียงสร้าง socket, file และ archive ขึ้นมาแล้วจึงส่ง/รับข้อมูลผ่าน archive

โดยปกติ object Csocket เป็น two-state object คือมีสองสถานะ คือ มีสถานะ คือ สถานะปกติ (asynchronous) และ synchronous เมื่อ socket อยู่ในสถานะ asynchronous มันสามารถรับสัญญาณ acynchronous จาก framework แต่ในขณะที่มันกำลังส่งหรือรับข้อมูล มันจะเปลี่ยนสถานะเป็น synchronous นั่นหมายความว่า socket ไม่สามารถรับสัญญาณ asynchronousได้อีกจนกว่าการทำงานของสถานะ synchronous จะเสร็จ

```
CmySocket::OnReceive()
{
        //...
        ar >> str;
        //...
}
```

ถ้า CSocket ไม่ได้สร้างเป็นแบบ two-state object มันอาจสามารถรับสัญญาณอื่นได้อีกสำหรับการทำงานของเหตุการณ์เดียวกับที่มันกำลังทำอยู่ จากตัวอย่างมันอาจรับสัญญาณให้ทำ OnReceive ในขณะที่มันกำลังทำ OnReceive อยู่เช่นการรับค่า str จาก archive นี้อาจทำให้เกิดการ recursion ได้ดังนั้น CSocket จึงได้ป้องกันเหตุการณ์เช่นนี้โดยใช้วิธีสลับ state

2.3.7 การใช้ Class CAsyncSocket

class CAyncSocket ได้ซ่อนการทำงานของ Windows Sockets API ไว้(ซึ่งได้ทำไว้ในvery low level) ทำให้โปรแกรมเมอร์ที่มีความรู้ด้านการเชื่อมต่อเครือข่ายสามารถจัดเก็บ event ในเน็ตเวิร์คได้สะดวกยิ่งขึ้นโดยไม่จำเป็นต้องทราบรายละเอียดของ API แต่อย่างใด

2.3.7.1 วิธีใช้ CAsyncSocket

- สร้าง object CAsyncSocket ซึ่งมีสองส่วนคือ

```
CAsyncSocket sock;
Sock.Create();//Use the default parameters
```

หรือ

```
CAsyncSocket* pSocket = new CAsyncSocket;
Int nPort = 27;
```

PSocket->Create(nPort,SOCK_DGRAM);

- ถ้า socket นั้นเป็น serverให้มีการListenด้วยการเรียกฟังก์ชัน CAsyncSocket::Listen แล้วเมื่อมีการร้องขอเชื่อมต่อเข้ามาให้เรียกฟังก์ชัน CAsyncSocket::Accept เพื่อยอมรับการ ร้องขอ หรือ ถ้า socket นั้นเป็น client ใช้ฟังก์ชัน CAsyncSocket::Connect เพื่อไปทำการเชื่อม ต่อกับ server socket ที่ได้ listen รอไว้อยู่แล้ว

- จัดการสื่อสารข้อมูลด้วยฟังก์ชันของ CAsyncSocket ที่ซ่อนการทำงานของ Windows Socket API ไว้

- ทำลาย object CAsyncSocket

2.3.8 การจัดการกับ CAsyncSocket

เมื่อ object ของ CasyncSocket ได้ถูกสร้างขึ้น object นั้นจะจัดการกับ handle Socket ให้ซึ่งโปรแกรมเมอร์สามารถจัดการกับลักษณะต่างๆของ Socket เช่น

- Blocking Mode
- Byte Ordering
- Converting String

ซึ่งจะกล่าวรายละเอียดในหัวข้อต่อไป

2.3.9 การสืบทอดคุณสมบัติจาก Class Socket ต่าง ๆ

การสืบทอดความสามารถของ class ใหม่จาก class CASyncSocket หรือ Csocket นั้น ทำให้โปรแกรมเมอร์สามารถเพิ่มคุณสมบัติของมันขึ้น และใน class เหล่านั้นจะมี virtual ฟังก์ชัน ที่สามารถ override ได้ นั่นคือ OnReceive,OnSend,OnAccept,OnConnect และ OnClose ซึ่งเป็นลักษณะของ callback ฟังก์ชันเมื่อเกิด notification ของ socket event นั้นๆ และยังมี virtual function(advance overridable)ของ Csocket ซึ่ง MFC จะเรียกฟังก์ชันเมื่อ socket กำลังดึงเมสเสจมาจากวินโดว์

2.3.10 Socket Notification

ในส่วนนี้จะอธิบายเกี่ยวกับฟังก์ชันที่ framework เรียกกลับมาที่ socket เมื่อเกิด เหตุการณ์ต่างๆดังนี้

- OnRerceive จะ notify เมื่อมีข้อมูลเข้ามาใน buffer จากการที่ socket อีกด้านที่ส่ง ข้อมูลนั้นมาและสามารถรับข้อมูลนั้นได้โดยการเรียกฟังก์ชัน Receive
- OnSend จะ notify เมื่อ socket กำลังจะเริ่มส่งข้อมูล
- OnAccept จะ notify เมื่อมี socket อื่นมาเชื่อมต่อกับ socket ที่ได้ Listen ไว้และสามารถเรียกฟังก์ชัน Accept เพื่อยอมรับการร้องขอการเชื่อมต่อนั้นได้

- OnConnect จะ notify เมื่อ socket นี้ได้เชื่อมต่อกับ socket ที่ Listen รอเสร็จแล้ว
- OnClose จะ notify เมื่อการเชื่อมต่อได้ถูกปิดลง

2.3.11 Byte Ordering

ในเครื่องคอมพิวเตอร์ที่มีสถาปัตยกรรมต่างกัน อาจมีการจัดเรียงลำดับ byte ที่ต่างกัน เช่น ในตระกูล Intel จะสลับ byte สูงกับต่ำ เรียกว่า "little-Endian" แต่ในตระกูล McIntosh ไม่สลับ เรียกว่า "big-Endian" สำหรับ CAsyncSocket นั้น โปรแกรมเมอร์จะต้องจัดการ byte ordering เอง แต่สำหรับ CSocket ได้จัดการส่วนนี้ไว้ให้แล้ว

2.3.12 Port และ Address ของ Socket

2.3.12.1 Port

Port เป็นเลขมาตรฐานที่ใช้กำหนดค่าให้กับ socket แต่ละตัวโดยจะไม่มีเลขที่ซ้ำกันเกิดขึ้นพร้อมกัน เลขport จะถูกเชื่อมต่อกับ application ที่สนับสนุน Window Sockets ดังนั้น Socket ของ application จะมีค่า port ไม่ซ้ำกัน ทำให้สามารถมี application ที่ทำงานกับ Socket ได้หลายตัวพร้อมกัน

2.3.12.2 Socket Address

Socket แต่ละตัวจะติดต่อกับ Internet Protocol (IP) Address ใน Network โดยทั่วไป address จะเปรียบเหมือนชื่อเครื่อง เช่น www.kmitl.ac.th หรือ "161.246.10.21"

2.4 การจัดการกับวินโดว์

2.4.1 เมสเสจ

เมสเสจเป็น input หรือข้อมูลเข้าเพียงทางเดียวของ application วินโดว์เป็นตัวแทนต่อทุกๆเหตุการณ์ทั้งหมดที่ต้องการตอบสนอง เมสเสจเป็นตัวแปรโครงสร้างชนิดหนึ่งที่ประกอบด้วยส่วนค่าอ้างอิงของเมสเสจนั้น กับส่วนพารามิเตอร์ของเมสเสจ โดยที่พารามิเตอร์จะขึ้นกับเมสเสจชนิดนั้นๆ

2.4.1.1 การสร้างและการประมวลผลเมสเสจ

วินโดว์จะมีการสร้างเมสเสจขึ้นทุกครั้งที่มี input เข้ามายังระบบ ไม่ว่าจะเป็นการเลื่อนเมาส์ กดคีย์ หรืออื่นๆ เพื่อบอกแก่ application หรือตัววินโดว์ถึงเหตุการณ์ ที่เกิดขึ้น โดยวินโดว์จะนำเมสเสจที่สร้างขึ้นเหล่านี้ใส่เข้าไปในคิวของระบบ จากนั้นก็ส่งผ่านไปยังคิวของ application ที่เหมาะสมต่อไป ลักษณะคิวจะเป็นแบบ เข้าก่อนออกก่อน การเข้าคือ วินโดว์ดึงเมสเสจจากคิวระบบมาใส่ให้แก่คิวของ application ส่วนการออกคือ application เป็นตัวดึงออกไปเองโดย เรียก

ฟังก์ชัน GetMessage จากนั้นก็แจกแจง และส่งไปยังฟังก์ชันประจำวินโดว์ต่างๆด้วยฟังก์ชัน DispatchMessage

มีเมสเสจบางตัวที่วินโดว์ต้องการส่งไปยังฟังก์ชันประจำวินโดว์โดยตรงแทนที่จะวางลงในคิวอาจเรียกเมสเสจนี้ว่าเป็นเมสเสจคิว เมสเสจคิวนี้จะเป็นเมสเสจที่มีผลกับวินโดว์นั้นๆ โดยเฉพาะวินโดว์และ application จะส่งเมสเสจชนิดนี้ด้วยฟังก์ชัน SendMessage ซึ่งจะเป็นการส่งเมสเสจไปยังฟังก์ชันวินโดว์โดยตรง อีกทั้งฟังก์ชันประจำวินโดว์นั้นยังสามารถส่งค่าการประมวลผลเมสเสจกลับมาได้อีกด้วย

ตัวอย่างเมสเสจคิวก็คือ การใช้ฟังก์ชัน CreateWindow นอกจากเป็นการสร้างวินโดว์แล้วยังเป็นการบังคับให้วินโดว์ส่งเมสเสจ WM_CREATE ซึ่งเป็นเมสเสจลัดคิวไปยังฟังก์ชันประจำวินโดว์ แล้วรอจนกว่าฟังก์ชันนั้นจะจัดการกับเมสเสจเสร็จ เมสเสจ WM_CREATE จะไม่ผ่าน application เลย

แต่ก็ไม่ใช่ว่ามีเพียงวินโดว์เท่านั่นที่สามารถสร้างเมสเสจขึ้นมาได้ ตัว application เอง ก็สามารถสร้างเมสเสจแล้วส่งไปยังคิวของตัวเอง และคิวของ application อื่นๆได้เช่นกัน

application จะใช้ฟังก์ชัน GetMessage ในลูปฟังก์ชัน WinMain เพื่อดึงเมสเสจออกจากคิวของตัว application เอง ฟังก์ชัน GetMessage จะทำงานโดยเริ่มจากดูว่าในคิวมีเมสเสจ หรือไม่ หากมีก็จะดึงเมสเสจที่อยู่แรกสุดไป แต่หากไม่มีเมสเสจอยู่ในคิว ก็จะเกิดการรอเมสเสจ ขึ้นและปล่อยการควบคุมกลับคืนไปให้วินโดว์ เพื่อแบ่งการควบคุมให้ application อื่นสามารถทำงานได้ต่อไป

เมื่อฟังก์ชัน WinMain ของ application ได้รับเมสเสจจากคิวแล้ว ก็ต้องใช้ฟังก์ชัน DispatchMessage เพื่อแจกแจงเมสเสจ และการส่งเมสเสจนั้นไปให้ฟังก์ชันประจำวินโดว์ต่างๆ การทำเช่นนั้น จะเป็นการถ่ายการควบคุมให้กับฟังก์ชันประจำวินโดว์ที่จะประมวลผลเมสเสจนั้น หลังจากที่ทำงานเกี่ยวข้องกับเมสเสจเรียบร้อยแล้ว ก็ถ่ายการควบคุมกลับไปให้ฟังก์ชันหลักของ application เพื่อดึงเมสเสจต่อไปในคิวมาประมวลผลต่อไป

ข้อควรระวังคือ ฟังก์ชันประจำวินโดว์ไม่ควรคาดว่าเมสเสจใดจะเข้ามาก่อน เพราะวินโดว์สามารถส่งเมสเสจมาในลำดับอย่างไรก็ได้

ส่วนการรับข้อมูลจากคีย์บอร์ดนั้น วินโดว์จะรับเข้ามาในลักษณะของรหัสคีย์ที่กด ดังนั้นจึงต้องมีการใช้ฟังก์ชัน TranslateMessage เพื่อแปลงรหัสของคีย์เหล่านั้นให้เป็นรหัสของตัวอักษร แล้วส่งเมสเสจตัวเองเข้าไปในคิว พร้อมทั้งรหัสตัวอักษรที่ผู้ใช้กดเข้ามาเป็นพารามิเตอร์ของ เมสเสจนั้น

2.4.1.2 การแปลและตีความเมสเสจ

โดยทั่วไปแล้วจะมีการใช้ฟังก์ชัน TranslateMessage นี้กับทุกๆเมสเสจที่เข้ามาโดยที่หากไม่ใช่เมสเสจที่เกี่ยวกับคีย์บอร์ดก็จะไม่มีผลอันใด

ตัวอย่างต่อไปนี้ก็จะเป็นการแสดงถึงหน้าตาเมสเสจลูปที่อยู่ในฟังก์ชัน WinMain ของ application เมสเสจลูปจะดึงเมสเสจออกจากคิวและแจกจ่ายดังนี้

```
int PASCAL WinMain( hInt,hPrevInst,lpCmdLine,ShowCmd )
HINSTANCE hInst;
HINSTANCE hPrevInst;
LPSTR lpCmdLine;
Int ShowCmd;
{
    MSG msg;
    .
    .
    .
    While( GetMessage( &msg,NULL,0,0 )){
      TranslateMessage( &msg );
      DispatchMessage( &msg );
    }
    return msg.wParam;
}
```

ส่วน application ที่มีการใช้คีย์ทันใจ(accelerator key)ของเมนู ก็ต้องมีการสร้างตารางคีย์ทันใจไว้ใช้ในไฟล์รีซอร์ส จากนั้นก็ต้องมีการโหลดตารางนี้เข้ามาโดยใช้ฟังก์ชัน LoadAccelerator โดยจะทำให้ลักษณะของลูปในการรับและส่งเมสเสจดังนี้

```
While( GetMessage((LPMSG)&msg,(HWND)NULL,0,0 )))
{
    if( TranslateAccelerator( hWindow,hAccel,((LPMSG)&msg) == 0 )
    {
```

```
            TranslateMessage((LPMSG)&msg));
            DispatchMessage((LPMSG)&msg));
        }
    }
```

ที่ต้องมีการตรวจสอบว่าค่าที่ได้กลับมาจากการเรียกใช้ฟังก์ชัน TranslateAccelerator นั้นก็เพราะว่าจากเมสเสจนั้น เป็นการกดคีย์ทันใจ ฟังก์ชัน TranslateAccelerator ก็จะทำการแปลงและส่งเมสเสจที่แปลงแล้วนั้น ไปยังฟังก์ชันประจำวินโดว์เอง ดังนั้นจึงไม่ต้องมาเรียกฟังก์ชัน TranslateMessage และฟังก์ชัน DispatchMessage อีก

2.4.1.3 การตรวจสอบเมสเสจ

เราใช้ฟังก์ชัน peekMessageเพื่อมองหาและตรวจสอบเมสเสจบางเมสเสจในระหว่างการทำงานที่กินเวลา โดยไม่ต้องการมีผลกระทบกระเทือนต่อการเป็นอยู่ของเมสเสจ เมื่อเรียกใช้ฟังก์ชั่นนี้ ถ้ามีเมสเสจในคิวฟังก์ชันนี้จะให้ค่าที่ไม่เป็น 0 กลับมา ทำให้สามารถประมวลเมสเสจได้โดยไม่ต้องกลับไปที่ลูปหลักของฟังก์ชัน WinMain ประโยชน์ของฟังก์ชันนี้คือ เมื่อมีการทำงานบางอย่างที่ต้องกินเวลามาก (ใช้เวลานานกว่าจะคืนการควบคุมไปให้ฟังก์ชันหลัก) เช่น การอ่านเขียนดิสก์ ก็สามารถใช้ฟังก์ชันนี้ในการตรวจสอบว่า ขณะนี้ผู้ใช้มีการกดคีย์เพื่อยกเลิกการทำงานหรือไม่

2.4.1.4 การส่งและส่งผ่านเมสเสจ

ฟังก์ชัน PostMessage และ SendMessageมีหน้าที่ในการส่งเมสเสจไปยังฟังก์ชันประจำวินโดวส์ของตัวเองหรือฟังก์ชันประจำวินโดวส์ของแอพพลิเคชันอื่น นอกจากนี้ยังมีฟังก์ชัน PostAppMessage ที่ทำงานแบบเดียวกับฟังก์ชัน PostMessage แต่เป็นการส่งผ่านโดยอาศัย Handle โมดูลของแอพพลิเคชันแทนฟังก์ชัน PostMessageจะส่งเมสเสจผ่านไปทางคิวของ แอพพลิเคชัน โดยที่ผู้ส่งจะยังไม่เสียการควบคุม และผลของเมสเสจที่ส่งไปนั้นก็จะยังไม่เกิดขึ้น จนกว่าแอพพลิเคชันนั้นๆ จะดึงเมสเสจที่ส่งไปนั้นขึ้นมา ส่วน SendMessage จะส่งเมสเสจลัดคิว ไปยังฟังก์ชันประจำวินโดวส์ที่ต้องการทันที โดยที่ไม่ต้องผ่านคิวของแอพพลิเคชัน ซึ่งการส่ง เมสเสจไปแบบนี้ผู้ที่ส่งจะสูญเสียการควบคุมไปให้ฟังก์ชันประจำวินโดว์ที่ส่งเมสเสจให้และเมื่อ ฟังก์ชันทำงานเสร็จแล้วก็จะคืนการควบคุมกลับมา ส่วนค่าที่ได้กลับมาของการเรียกใช้ SendMessage นั้นคือค่าที่ฟังก์ชันประจำวินโดวส์นั้นส่งค่ากลับมา แต่ถ้าเป็นค่าที่ได้กลับมาจาก ฟังก์ชัน PostMessage เป็นเพียงว่าสามารถส่งเมสเสจนั้นไปได้สำเร็จหรือไม่

2.4.1.5 ฟังก์ชันที่ใช้กับเมสเสจ

ฟังก์ชันในกลุ่มนี้มีหน้าที่ในการอ่านเมสเสจขึ้นจากคิว และประมวลผลเมสเสจในวินโดว์ ฟังก์ชันที่ใช้กับเมสเสจก็จะมีตารางดังนี้

ตารางที่ 2.4 แสดงฟังก์ชันที่ทำงานกับเมสเสจของวินโดว์

ฟังก์ชัน	หน้าที่
CallWindowProc	ส่งผ่านข้อมูลของเมสเสจไปยังฟังก์ชัน
DispatchMessage	แจกแจงเมสเสจและส่งเมสเสจไปยังฟังก์ชันประจำวินโดว์ใดๆ
GetMessage	รับเมสเสจในช่วงของเมสเสจที่กำหนดไว้
GetMessagePost	ให้ค่าตำแหน่งของเมาส์ในขณะที่มีการรับเมสเสจล่าสุด
GetMessageTime	ให้ค่าเวลาในขณะที่มีการรับเมสเสจล่าสุด
PeekMessage	ตรวจสอบว่ามีเมสเสจในคิวหรือไม่โดยไม่ได้ดึงออกมา
PostAppMessage	ส่งเมสเสจไปยังแอพพลิเคชัน
PostMessage	ส่งเมสเสจไปยังคิวของของแอพพลิเคชัน
PostQuitMessage	ส่ง WM_QUIT ไปยังแอพพลิเคชัน
ReplyMessage	ตอบรับเมสเสจ
SendMessage	ส่งเมสเสจไปยังวินโดว์ต่างๆ
TranslateAccelerator	ประมวลผลคีย์ทันใจของเมนูให้เป็นเมสเสจและส่งเมสเสจให้แก่วินโดว์
TranslateMessage	แปลงค่ารหัสของคีย์ที่กดให้เป็นรหัสตัวอักษร
WaitMessage	ส่งการควบคุมไปให้แอพพลิเคชันอื่น
WinMain	เป็นจุดที่เริ่มเข้าไปทำงานของวินโดว์แอพพลิเคชัน

2.4.2 การสร้างและการจัดการวินโดว์

ส่วนนี้จะเป็นเรื่องของการสร้าง ยกเลิก เปลี่ยนแปลง หรือดึงข้อมูลจากวินโดว์เป็นเรื่องที่เรียกได้ว่าเป็นหัวใจของการสร้าง Application บน Windows เลยทีเดียว

2.4.2.1 Class of Window

Class ของ Window หมายถึง ลักษณะเฉพาะที่เป็นตัวกำหนดว่า Window นั้นจะมี รูปร่างหน้าตาเป็นอย่างไร มีความสามารถและมีพฤติกรรมเป็นอย่างไร ก่อนที่ Application จะ สร้าง Window ขึ้นมาให้ผู้ใช้เห็นบนหน้าจอได้ต้องมีการสร้าง " Class" ของ Window เสียก่อน

การสร้างก็เพียงผ่านพารามิเตอร์ที่เหมาะสมไปให้ฟังก์ชัน Register Class โดยทีสามารถสร้างClass ของ Window ได้มากเท่าที่ต้องการ จากนั้นก็สร้าง Window โดยบอกว่า Window นั้นให้มีลักษณะเป็นไปตาม Class ใด ซึ่ง Class หนึ่งๆ นั้น สามารถใช้โดยที่ Window ก็ได้และ Class ของ Window ที่สร้างไว้ก็จะคงจนกว่าจะจบ Applicaition

แม้ว่าข้อมูลที่ต้องใสตอนสร้าง Class นั้นจะมีมากมาย แต่จะไม่กี่ตัวที่จำเป็นจริงๆ สำหรับการสร้าง Class เช่น Class name ฟังก์ชันที่ใช้ในการรับ Message มาประมวล และ handle ที่เป็นค่าคงที่ของ Applicatiion นั้นๆส่วนข้อมูลตัวอื่นๆ ก็จะเป็นพวกขนาดตำแหน่ง ลักษณะCursor และอื่นๆ

Class ของ Window สามารถแบ่งได้ตามลักษณะการสร้างและการคงอยู่ 3 ประเภทดังนี้

- Class ของระบบ (System Global Class)

Class ชนิดนี้ windows จะสร้างขึ้นเมื่อเริ่มทำงาน ทุก Application สามารถเรียกใช้ได้ทันที Class ชนิดนี้ไม่สามารถถูกสร้างหรือยกเลิกโดย Application ได้ ตัวอย่างของ Class เหล่านี้ก็เช่น Control Edit หรือ กรอบรายชื่อ

- Class ส่วนรวมของ Application (Application Global Class)

Application หรือ Library สามารถสร้าง Class ที่ถูกเรียกใช้โดย Application อื่นๆ ได้โดยกำหนด CS_GLOBAL CLASS ในตอนสร้าง Class โดยที่ Class ชนิดนี้จะหายไปก็ต่อเมื่อ Application หรือ Library นั้นเลิกทำงาน

- Class ของ Application (Application Local Class)

เป็น Class ที่พบเห็นกันมากที่สุดคือ Application ใดสร้างก็สามารถเรียกใช้งานได้ใน Application นั้นเท่านั้น

เมื่อมีการสร้าง window โดยระบุ Class ที่ต้องการ Window จะไปหาใน Class ของ Application ก่อน ถ้าไม่พบก็จะไปหาใน Class ที่ใช้ได้ทุก Application และถ้าไม่เจออีกก็จะไปหาใน Class ของระบบ การที่ window มีลำดับการหา Class อย่างนี้ทำให้สามารถสร้าง Class ขึ้นมาทับ Class เดิม โดยที่ไม่กระทบถึง Application ที่ใช้ Class เดิมนั้นอยู่

2.4.2.2 ส่วนประกอบของ Class ของ window

ส่วนประกอบต่างๆ Class ของ window จะเป็นตัวที่บอกลักษณะของ window นั้นซึ่ง ข้อมูลเหล่านี้จะถูกเก็บไว้ในโครงสร้างแบบ WNDCLASS จากนั้นก็จะส่งผ่านเป็น พารามิเตอร์ ให้กับฟังก์ชัน Register Class ซึ่งข้อมูลเหล่านี้สามารถเรียกดูภายหลังได้โดยฟังก์ชัน
get class info ส่วนประกอบเหล่านี้มีดังตารางต่อไปนี้

ตารางที่ 2.5 แสดงส่วนประกอบต่างๆใน Class ของวินโดว์

ชื่อ	จุดประสงค์
- ชื่อ class	เป็นชื่อของ class ซึ่งแตกต่างกันไปในแต่ละclass
- address funtion ประจำ window	เป็นการกำหนด address ของ function ที่จะใช้ในการรับ message ที่ส่งมายัง function ของ application
- handle instant	เป็นตัวบอกถึง application ที่ขึ้นทะเบียน class นั้น
- cursor of class	กำหนดรูปร่างของcursorที่จะปรากฏใน window ของclass
- Icon of class	กำหนด icon เมื่อ window ถูกลดขนาด
- Blackground of class	กำหนดสีและรูปแบบของพื้นที่ใช้งาน(Client area)เมื่อแสดง window เป็นครั้งแรก
- Menu of class	กำหนดmenu ของ window ที่ไม่มีการกำหนดmenuโดยเฉพาะ
- สไตล์ ของ class	กำหนดว่าจะทำอย่างไรเมื่อมีการย้าย window เปลี่ยนขนาด window และ อื่นๆ
- พื้นที่ class พิเศษ	กำหนดขนาดของหน่วยจำ(byte) ที่window ควรจองไว้ต่อจากโครงสร้างของ class
- พื้นที่ window พิเศษ	กำหนดขนาดของหน่วยจำที่window ควรจองไว้ต่อจากโครงสร้างของ window ที่application สร้างขึ้น

2.4.2.3 สไตล์ ของ class

สไตล์ ของ class เป็นการกำหนดลักษณะของ window เพิ่มเติม โดยสามารถกำหนดได้ครั้งละมากกว่าหนึ่งสไตล์โดยใช้ โอเปอร์เรเตอร์ OR สไตล์ของ class มีดังตารางต่อไปนี้

ตารางที่ 2.6 แสดงสไตล์ของ Class ที่กำหนดลักษณะของวินโดว์

สไตล์	คุณสมบัติ
CS_BYTEALIGNCLIENT	จัดวางพื้นที่ใช้งานตาม byte (แนวนอน)
CS_BYTEALIGNWINDOW	จัดวางwindow ตาม byte(แนวนอน)
CS_CLASSDC	จอง contex display เพียงหนึ่งสำหรับทุก window
CS_DBCLKS	ส่งmessage double click ไปยังfunction ประจำ window
CS_GLOBALCLASS	กำหนดว่า class นั้นสามารถถูกใช้ได้กับทุก application
CS_HREDRAW	กำหนดว่าจะต้องมีการวาด window ใหม่เมื่อมีการย้ายหรือเปลี่ยนแปลงขนาดที่มีผลต่อความกว้างของ window

ตารางที่ 2.6 แสดงสไตล์ของ Class ที่กำหนดลักษณะของวินโดว์(ต่อ)

CS_NOCLOSE	ไม่ให้ใช้คำสั่ง close ในmenu ของระบบ
CS_OWNDC	จอง contex display for window ที่ใช้ class นั้น
CS_PARENTDC	ให้ class ของ window นั้นใช้ context display ของ window parent ของ window นั้นๆ
CS_SAVBITS	กำหนดให้มีการเก็บภาพส่วนของ window ที่ถูกบัง เพื่อใช้ในการวาดใหม่เมื่อ window ถูกย้าย การทำเช่นนี้จะไม่มีการส่ง WM_PAINT ไปยังwindow ถ้าหน่วยจำที่เก็บรูปแบบไม่ถูกทิ้ง
CS_VREDRAW	กำหนดว่าจะต้องมีการวาด window ใหม่เมื่อมีการย้ายหรือเปลี่ยนแปลงขนาดที่มีผลต่อความสูงของ window

2.4.2.4 Function ประจำ Windows

Function ประจำ Windows เป็น function ที่ทำหน้าที่ประมวล Message ของ Window Message จะถูกส่งมาโดย window เมื่อมี input (เช่น mouse , keyboard ,timer) เข้ามาหรือไม่ Window ก็ต้องการให้ Window ทำงานบางอย่าง เช่นวาดพื้นที่ใช้งานใหม่ นอกจากมีหน้าที่ในการรับ Input , Message ต่างๆแล้ว function ประจำ window ยังมีหน้าที่ในการรับข้อมูลเมื่อระบบถูกเปลี่ยนแปลงโดย application อื่นๆ เช่น มีการแก้ไข file win.ini หรือการทำตามคำร้องขอบางอย่างเช่น การเปลี่ยนแปลง menu ก่อนการแสดงให้ผู้ใช้เห็น หรือการรับทราบว่าขณะนี้ตัวเองได้เป็น Window Active แล้ว รวมทั้งข้อมูลอย่างอื่นๆ อีกมากมาย

แม้ว่า Message ส่วนใหญ่จะถูกส่งมาจาก Window แต่ Window ด้วยกันเองก็สามารถส่ง Message ถึงกันได้ (หรือจากตัวเองก็ได้) ทั้งนี้เพื่อจะได้ทราบความเป็นไปของ Window อื่นๆ ตามปกติแล้วการรับ Message ของ function ประจำ Window จะเป็นไปเรื่อยๆ จนกว่า Window นั้นจะถูกทำลาย

เนื่องจากการทำงานของ application นั้นขึ้นกับ Message ที่แต่ละ Window ได้รับ ดังนั้นตัว function ประจำ window จึงเป็นส่วนที่บังคับการดำเนินไปของ Window นั้นๆ เช่น ใน application ที่มีการเปิด file เมื่อผู้ใช้เลือกคำสั่ง open function ประจำ Window ก็จะเป็นตัวสั่งการว่าต่อไปให้ทำการเปิด file ตามที่ผู้ใช้ต้องการ หรือผู้ใช้มีการเลื่อน scroll bar function ประจำ Window ก็ต้องเลื่อนข้อมูลที่อยู่นอก window ขึ้นมาให้ผู้ใช้ดูตามต้องการ

ส่วน Message ที่เข้ามายัง function ประจำ Window แต่ไม่ได้ใช้ประโยชน์อะไรนั้น ต้องมี

การส่งต่อไปให้ function DefWindowProc เป็นตัวประมวลผล Message เหล่านั้น โดยเฉพาะอย่างยิ่ง Message ที่เกิดจากนอกพื้นที่ใช้งาน Window ซึ่งเป็นงานของ Window เช่น การ click ที่ title bar หรือการลดขนาด Window ดังนั้นใน switch case ของทุก function ประจำ Window ควรมี DefWindowProc นี้อยู่เสมอ

2.4.2.5 Message ของ Window

message เป็นกลุ่มของข้อมูลอย่างหนึ่งที่ถูกส่งไปยัง function ประจำ window ต่างๆ ที่ทำงานใน window ขณะนั้น ประกอบด้วย 4 ส่วน คือ handle ของ window ที่ต้องการส่งไป ตัว message id parameter ขนาด 16 บิต และสุดท้ายคือ parameter ขนาด 32 บิต สำหรับ parameter 2 ตัวหลังนั้นจะเป็นข้อมูลของอะไรก็จะขึ้นกับชนิดของmessageนั้น ซึ่งบาง message จะไม่ใช้ parameter ครบทั้งสองตัว

ลักษณะของ function ประจำ window ที่มี message เป็น argument เป็นดังนี้

```
LONG  FAR  PASCAL Wndproc( hWnd,wMsg,wParam,lparam)
HWND       hWnd;
WORD       wMsg;
WORD       wParam;
DWORD      lparam;
```

HWnd เป็นพารามิเตอร์ที่เก็บแฮนเดิลของวินโดว์ที่รับเมสเสจนี้ wMsg เป็นตัวเมสเสจ ที่ส่งมา wParam เป็นพารามิเตอร์เพิ่มเติมของเมสเสจที่ส่งมา โดยข้อมูล 16 บิต และ lparam เป็นพารามิเตอร์เพิ่มเติมของเมสเสจที่ส่งมา โดยเป็นข้อมูล 32 บิต ตัวอย่างการใช้พารามิเตอร์ สองตัวหลังเช่น ถ้ามีการเลื่อนเมาส์ก็จะเกิดเมสเสจ WM_MOUSEMOVE และ wParam ก็จะมี ข้อมูลของปุ่มที่ถูกกด ส่วน lparam ก็จะเป็นตำแหน่งของเมาส์ที่ถูกเลื่อนมา

2.4.2.6 งานตอบสนองปกติ

ฟังก์ชัน DefWindowProc เป็นฟังก์ชันที่ใช้ในการประมวลผลเมสเสจที่ผ่านเข้ามายังฟังก์ชันประจำวินโดว์แต่ไม่ได้ใช้งาน ซึ่งวินโดว์ได้กำหนดการตอบสนองตามปกติที่ฟังก์ชัน DefWiondowProc ในตัววินโดว์

2.4.2.7 สไตล์ของวินโดว์

เมื่อใช้ฟังก์ชัน Create Window ในการสร้างวินโดว์ใหม่สามารถกำหนดสไตล์หรือลักษณะเฉพาะของวินโดว์นั้นได้ ซึ่งสไตล์เหล่านี้สามารถนำมาผสมกันในวินโดว์เดียวกันได้

สไตล์ของวินโดว์มีดังนี้

- วินโดว์แบบซ้อนทับกัน

วินโดว์แบบนี้จะใช้เป็นวินโดว์หลักของแต่ละ application คือจะอยู่ในระดับบนเสมอ ไม่เป็นวินโดว์ child ของวินโดว์อื่น ส่วนประกอบของวินโดว์แบบนี้สามารถมีได้ครบทุกอย่าง ไม่ว่าจะเป็นเมนูของระบบ กล่องลดขนาด กล่องขยายขนาด scroll bar หากมีการกำหนดว่าให้ มี และถ้าวินโดว์นี้เป็นวินโดว์หลัก ขอแนะนำให้มีเมนูของระบบและกล่องขยาย/ลดขนาดเสมอ

การกำหนดให้มีวินโดว์สไตล์นี้ให้ใช้ WS_OVERLAPPED มีกรอบและ caption หรือ WS_OVERLAPPEDWINDOW มีกรอบหนา caption เมนูระบบ กล่องลด/ขยายขนาด ในฟังก์ชัน Create Window

- วินโดว์แบบมีเจ้าของ

วินโดว์แบบนี้ ต้องมีวินโดว์ที่เป็นวินโดว์ parent และวินโดว์ parent นั้นต้องเป็นแบบซ้อนทับ (WS_OVERLAPPED) ด้วยการที่เป็นวินโดว์แบบนี้ทำให้เกิดข้อบังคับเกิดขึ้นดังนี้

- วินโดว์แบบมีเจ้าของนี้ จะต้องปรากฏอยู่บนวินโดว์ parent ของมันเสมอ และถ้าพยายามย้ายวินโดว์ parent ขึ้นมาข้างบนก็จะมีการเปลี่ยนตำแหน่งของวินโดว์แบบนี้เพื่อให้แน่ใจว่ายังอยู่บนวินโดว์ parent
- เมื่อยกเลิกวินโดว์ parent วินโดว์แบบมีเจ้าของนี้ ก็จะถูกยกเลิกไปด้วย
- เมื่อวินโดว์ parent ถูกลดขนาด วินโดว์แบบมีเจ้าของ ก็จะหายไป

การสร้างวินโดว์ให้มีสไตล์แบบนี้ ทำได้โดยกำหนดแฮนเดิลของวินโดว์ parent ไปที่ hWndParent ในพารามิเตอร์ของฟังก์ชัน create window ส่วนสไตล์ที่กำหนดให้ ก็เป็น WS_OVERLAPPED กรอบข้อความเป็นวินโดว์แบบนี้ โดยอัตโนมัติอยู่แล้ว

- วินโดว์แบบ pop-up

วินโดว์แบบ pop-up เป็นชนิดหนึ่งของวินโดว์แบบซ้อนทับ มีคุณสมบัติดังกล่าวเหมือนกันทั้งหมด เพียงแต่วินโดว์แบบ pop-up จะสามารถเลือกได้ว่า ให้มี caption หรือไม่ ถ้าเป็น แบบซ้อนทับต้องมีเสมอ

- วินโดว์ child (child window)

วินโดว์แบบนี้จะถูกจำกัดบริเวณอยู่ภายในพื้นที่ใช้งานของวินโดว์parentเท่านั้น ส่วนมากจะใช้ในการแบ่งพื้นที่ใช้งานของวินโดว์ parent เป็นส่วนย่อยๆ การสร้างวินโดว์ชนิดนี้ ให้ใช้ WS_CHILD ในฟังก์ชัน create window ส่วนการแสดง หรือไม่แสดง วินโดว์ก็ใช้ฟังก์ชัน show window

ทุกๆ window child ต้องมี parent เสมอ วินโดว์ที่เป็น parent ของมันสามารถเป็นวินโดว์ซ้อนทับ วินโดว์แบบ pop-up หรือว่าเป็น วินโดว์ child เองก็ได้ เมื่อวินโดว์ parent มีวินโดว์ child

อยู่ในพื้นที่ใช้งานของมัน ก็หมายความว่า พื้นที่ใช้งานส่วนนั้น ก็จะไม่เป็นของวินโดว์ parent อีกต่อไป เมสเสจที่เกิดจาก พื้นที่ในงานตรงนั้น ก็จะถูกส่งไปยังฟังก์ชันวินโดว์ child นั้น วินโดว์ child ที่เข้ามาอยู่ในพื้นที่ใช้งานของวินโดว์ parent หนึ่งๆนั้น ไม่จำเป็นต้องมี class เดียวกันทำให้สามารถรับผิดชอบงานต่างๆกัน ในพื้นที่ใช้งานเดียวกันได้

ตัววินโดว์ child เอง นั้นนอกจากพื้นที่ใช้งานแล้ว จะไม่มีส่วนประกอบอื่นโดยอัตโนมัติ จำเป็นต้องมีการกำหนดลงไปว่า ต้องการอะไรบ้าง โดยปกติแล้ว วินโดว์childนี้ จะต้องมีค่าอ้างอิงประจำตัว โดยกำหนดผ่านทาง create window เพื่อบอกกับ วินโดว์ parent ว่า เมสเสจที่ส่งมานี้มาจาก วินโดว์ child ใด

การกำหนดตำแหน่งของวินโดว์ child จะอ้างอิงเทียบกับ มุมบนซ้ายของพื้นที่ใช้งานของวินโดว์ parent หากบางส่วนของวินโดว์ child ถูกเลื่อนออกไป นอกพื้นที่ใช้งานส่วนนั้นก็จะไม่มีการแสดง ส่วนเมสเสจนั้นก็สามารถส่งตรงไปยังวินโดว์ child ได้โดยตรงถ้าต้องการ เว้นเสียแต่ว่ามีการใช้ enable window ในกรณีนี้ วินโดว์จะส่งเมสเสจุไปยังวินโดว์ parent ของ วินโดว์ child นั้น เพื่อเปิดโอกาสให้วินโดว์ parent มีการตรวจสอบเมสเสจ และอาจมี enable window child ขึ้นมาอีกครั้ง

การกระทำส่วนมากที่เกิดขึ้นกับวินโดว์ parent ก็ยังมีผลกับวินโดว์ child ด้วย ดังรายการในตารางต่อไปนี้

ตารางที่ 2.7 แสดงผลกระทบของวินโดว์ parent ที่มีต่อวินโดว์ child

วินโดว์ parent	วินโดว์ child
ถูกแสดง	แสดงขึ้นมาตามวินโดว์ parent ด้วย
ถูกซ่อน	ถูกซ่อนตามวินโดว์ parent
ถูกยกเลิก	ถูกยกเลิกตามวินโดว์ parent
ถูกเคลื่อนย้าย	ถูกเคลื่อนย้ายโดยอิงกับตำแหน่งใหม่
ถูกขยายขนาด	มีการวาดส่วนที่ก่อนวินโดว์ parent ขยายหรือเพิ่มขนาด

2.4.2.8 วงจรชีวิตของวินโดว์

หน้าที่หลักของวินโดว์ คือ รับ input และการแสดง output ดังนั้นวงจรชีวิตของวินโดว์จึงเริ่มขึ้นเมื่อ application ต้องการรับ Input และ Output และจะสิ้นสุดเมื่อไม่ต้องการหรือจบ application

โดยปกติแล้ววินโดว์ก็จะคงอยู่จนกว่าจะจบ application แต่วินโดว์ที่ใช้งานเฉพาะ บางอย่างก็จะมีชีวิตสั้นๆ เช่น วินโดว์ของกรอกข้อความ

วินโดว์เริ่มขึ้นจากขั้นตอนการสร้าง โดยการส่ง class ของวินโดว์ ที่ทำการขึ้นทะเบียนไว้แล้วให้กับฟังก์ชัน create window ดังนั้นวินโดว์ก็จะไปเตรียมข้อมูลเกี่ยวกับวินโดว์ใหม่นั้นแล้ว ให้ค่าตัวเลขกลับมาสู่ application สำหรับใช้ในการอ้างถึงวินโดว์นี้ภายหลัง เรียกตัวเลขนี้ว่า handle ของวินโดว์

เมสเสจแรกที่ถูกส่งไปยังฟังก์ชันประจำวินโดว์ คือ WM_CREATE ซึ่งเป็นเมสเสจที่บอกว่าให้ทำการเตรียมตัวบางอย่างก่อนการแสดงวินโดว์ เช่น การจองหน่วยความจำ หรือเปิด file ข้อมูลเตรียมไว้ ข้อมูลที่ส่งมาด้วยก็จะอยู่ใน lParam เป็น pointer ไปยังโครงสร้าง create structer ซึ่งเก็บข้อมูลเกี่ยวกับวินโดว์นั้นอยู่ เมสเสจ WM_CREATE และ WM_NCREATE นี้ ถูกส่งให้วินโดว์โดยไม่ผ่านคิวของ application ซึ่งก็หมายความว่า จะทำงานก่อน loop หลัก ของ application

การที่จะเริ่มใช้งาน window ได้นั้น จะต้องแสดง window นั้นขึ้นมาเสียก่อน หาก window ไม่ได้ถูกสร้างด้วยสไตล์ WS_VISIBLE แล้ว ก็ต้องใช้ function show window ในการแสดงแต่สำหรับ window หลักของ application แล้ว ไม่ควรกำหนดให้เป็นสไตล์ WS_VISIBLE ควรใช้ show window แทน โดยใช้ตัวแปร mCmdShow เป็นพารามิเตอร์ของ function เพื่อบอกว่าควรแสดงหรือไม่

เมื่อ window สิ้นสุดการทำงานหรือจบ application แล้ว ก็ต้องยกเลิกหรือทำลายwindow นั้น ด้วย function Destroy Window โดย function นั้นจะหลบwindow นั้นออกจากหน้าจอ และส่ง message WM_DESTROY และ WM_NCDESTROY ไปยัง function ประจำ window (อีกทางหนึ่งที่ function ของ window จะได้รับ message นี้ คือ มี message WN_CLOSE ส่งไปยัง function DefWindowproc)

สำหรับ window ที่เป็น window หลักของ application ควรจะถูกทำลายเป็น window สุดท้าย และควรมีการบอกให้จบ application ด้วย โดยเมื่อ window หลักได้รับWM_DESTROY แล้วก็ให้เรียกฟังก์ชัน PostQuitMessage เพื่อส่ง WM_QUIT ไปยังคิวของ application และเมื่อเมนูลูปอ่านเมสเสจขึ้นมา ก็จะเป็นการจบ application

2.4.2.9 ฟังก์ชันสำหรับสร้างวินโดว์

ฟังก์ชันในกลุ่มนี้ มีหน้าที่เกี่ยวกับการสร้าง ทำลาย เปลี่ยนแปลง หรือดึงข้อมูลจากวินโดว์ เรื่องราวของฟังก์ชันกลุ่มนี้ แสดงดังตารางต่อไปนี้

ตารางที่ 2.8 แสดงฟังก์ชันที่มีหน้าที่สำหรับการสร้างวินโดว์

ฟังก์ชัน	หน้าที่
Adjust Window Rect	คำนวณขนาดของวินโดว์ให้เหมาะกับพื้นที่ใช้งาน
Adjust Window Rect Ex	คำนวณขนาดวินโดว์แบบ extended ให้เหมาะกับพื้นที่ใช้งาน
Create Window	สร้างวินโดว์ชนิด child,pop-up,ทับกัน
Create Window Ex	สร้างวินโดว์ชนิด child,pop-up ทับกัน แบบ extended
Def Dig Proc	กำหนดวิธีการที่จะจัดการกับเมสเสจของกรอบข้อความที่ application ไม่ได้ประมวลผล
Def Window Proc	กำหนดวิธีการที่จะจัดการกับเมสเสจ ของฟังก์ชันของวินโดว์ที่ application ไม่ได้ประมวลผล
Destroy Window	ยกเลิกวินโดว์
Get Class Info	ดึงข้อมูลเกี่ยวกับ class ที่ต้องการ
Get Class Door	ดึงข้อมูลของ class ของวินโดว์จากโครงสร้าง WMDCLASS
Get Class Name	ดึงข้อมูลชื่อของ class
Get Class Word	ดึงข้อมูลของ class ของวินโดว์ จากโครงสร้าง WMDCLASS
Get Last Active Pop-Up	ดูว่าวินโดว์ใด active หลังสุด
Get Window Long	ดึงข้อมูลเกี่ยวกับวินโดว์
Get Windto Word	ดึงข้อมูลเกี่ยวกับวินโดว์
Register Class	register class ใหม่
Set Class Long	เปลี่ยนแปลงข้อมูลโครงสร้าง ในแบบ WMDCLASS
Set Class Word	เปลี่ยนแปลงข้อมูลโครงสร้าง ในแบบ WMDCLASS
Set Window Long	เปลี่ยนลักษณะของวินโดว์
Set Window Word	เปลี่ยนลักษณะของวินโดว์
Unregister Class	ยกเลิก class ของวินโดว์จากตาราง class

บทที่ 3
การออกแบบโปรแกรม

3.1 Functionการทำงานหลักของ TwSocket

TwSocket เป็น class ที่ถูกออกแบบเพื่อเอื้อประโยชน์ในการเรียกใช้งานพื้นฐานในการควบคุม socket มี feature ในการเคลื่อนย้ายข้อมูล ทั้งข้อมูลที่เป็นชนิด binary และ ASCII code ซึ่งจะแบ่งลักษณะการทำงานเป็นสองฝ่าย คือ ฝ่ายที่ทำการส่งข้อมูล และฝ่ายที่ทำการรับข้อมูล

3.1.1 ฝ่ายที่ทำการส่งข้อมูล

Close : TwSocket จะทำการจัดการปิด connection ต่างๆ ที่ทำการ connect อยู่ ณ เวลานั้น เพื่อเตรียมพร้อมสำหรับการสร้างเส้นทางสำหรับการเชื่อมโยงสำหรับเครื่องปลายทางที่ได้ถูกกำหนดเอาไว้

Addr : TwSocket จะทำการอ่านค่า address ของเครื่องปลายทาง และเก็บเอาไว้ใน property addr ซึ่งเป็นคุณสมบัติพื้นฐานในการเชื่อมต่อผ่าน TCP/IP

Port : TwSocket จะทำการอ่านค่า port ที่ใช้ในการส่งออกไปของเครื่องผู้ส่ง และจัดเก็บเอาไว้ใน property port ซึ่งเป็นคุณสมบัติพื้นฐานในการเชื่อมต่อผ่าน TCP/IP

Connect : TwSocket จะทำการเชื่อมต่อไปยังเครื่องปลายทาง ซึ่งอ้างอิงจากหมายเลข address ใน property ชื่อ addr โดยผ่าน port ซึ่งอ้างอิงจาก property ชื่อ port เป็นเสมือนการสร้างเส้นทางจำลอง เตรียมเพื่อจะส่งข้อมูลออกไป

Sendstr : TwSocket จะอ่านค่า string ใน buffer ซึ่งจะถูก set ให้ทำการ เก็บข้อมูลของ file ที่ต้องการจะส่ง มาเก็บไว้ใน buffer แล้วทำการส่งออกไปให้แก่เครื่องปลายทางที่ได้ทำการเชื่อมต่อไว้แล้ว(connect)

3.1.2 ฝ่ายที่ทำการรับข้อมูล

Close : TwSocket จะทำการจัดการปิด connection ต่างๆ ที่ทำการ connect อยู่ ณ เวลานั้น เพื่อเตรียมพร้อมสำหรับการสร้างเส้นทางสำหรับการเชื่อมโยงสำหรับเครื่องปลายทางที่ได้ถูกกำหนดเอาไว้

Addr : ถูก set เป็นค่า 0.0.0.0 เพื่อพร้อมสำหรับการเชื่อมต่อจากทุกๆ IP (address)

Port : TwSocket จะทำการอ่านค่า port ที่ใช้ในการส่งออกไปของเครื่องผู้ส่ง และจัดเก็บเอาไว้ใน property port ซึ่งเป็นคุณสมบัติพื้นฐานในการเชื่อมต่อผ่าน TCP/IP

Proto : TwSocket ทำการ set ชื่อของ protocol เอาไว้ เพื่อทำการจัดการกับรูปแบบ (format) ตามมาตรฐานของแต่ละ protocol ซึ่งในที่นี้ Proto จะถูก set เป็น TCP

Listen : TwSocket จะทำการเปิด port และเตรียมพร้อม socket สำหรับการเข้ามาเชื่อมต่อจากเครื่องที่ต้องการจะติดต่อด้วย โดย port จะถูกอ้างอิงจาก property ของ TwSocket

Receive : TwSocket จะให้ข้อมูลที่ถูกผ่านเข้า socket และนำไปเก็บไว้ใน buffer ขนาด 2048 ไบต์ และทำการ read จาก buffer เข้ามาเก็บไว้เป็น file

นอกจากนี้ยังมีฟังก์ชั่นที่สามารถใช้ร่วมกันได้ ทั้งฝ่ายส่งข้อมูลและฝ่ายรับข้อมูลคือ

Disconnect : TwSocket จะเรียกฟังก์ชั่น Close มาทำงานเพื่อ ทำการปิดทุกๆ connection ที่มีอยู่ขณะนั้นโดยทันที

3.2 หน้าที่หลักของการทำงานของโปรแกรม

3.2.1 การเชื่อมต่อไปยังเครื่องปลายทาง

เป็นการร้องขอเพื่อทำการเชื่อมต่อไปยังเครื่องปลายทาง โดยที่ทางเครื่องปลายทางจะมีการเฝ้าฟังอยู่ตลอด(listening) ซึ่งเมื่อทำการเชื่อมต่อกับเครื่องปลายทางแล้ว ก็จะมีการส่ง echo กลับไปยังเครื่องที่ทำการร้องขอมา เพื่อเป็นการแจ้งให้ทราบ

3.2.2 การบีบอัด file (compression)

เป็นการทำงานโดยการรับ argument มาจากหน้าจอ แล้วทำการส่ง argument ซึ่งเป็นชื่อ file ที่ต้องการบีบอัด ไปยังโปรแกรม compress ซึ่งจะทำงานอยู่ด้านหลังของส่วน interface หลังจากนั้น ก็จะทำการบีบอัดข้อมูลตาม algorithm ที่ได้เลือกใช้จากทางหน้าจอ interface และทำการเก็บข้อมูลที่บีบอัดแล้ว เป็นชื่อ file default ซึ่งถูก set ค่าใช้ใหม่ทุกครั้ง ที่ทำการ compress

3.2.3 การส่งข้อมูล (sending)

เป็นการใส่ข้อมูลที่ถูกบีบอัดแล้วลงไปใน buffer แล้วหลังจากนั้นก็จะทำการอ่านข้อมูลใน buffer จัดเป็น block โดยที่ 1 block เท่ากับขนาด buffer แล้วทำการส่งข้อมูลนั้นผ่าน socket ไปทีละ block

3.2.4 การรับข้อมูล (receive)

ข้อมูลจะถูกส่งเข้ามาใน buffer ของเครื่องปลายทาง แล้วทำการอ่านจาก buffer เข้าไป ทีละครั้งจนครบเป็น file ที่ถูกส่งมา

3.2.5 การขยายข้อมูล (decompress)

เป็นการทำงานโดยรับ argument ที่เป็น destination ผ่านหน้าจอ interface แล้วทำการอ่าน file ที่ถูกบีบอัดผ่าน location ที่ถูก set เป็น default ภายในโปรแกรม หลังจากนั้นก็จะถูก

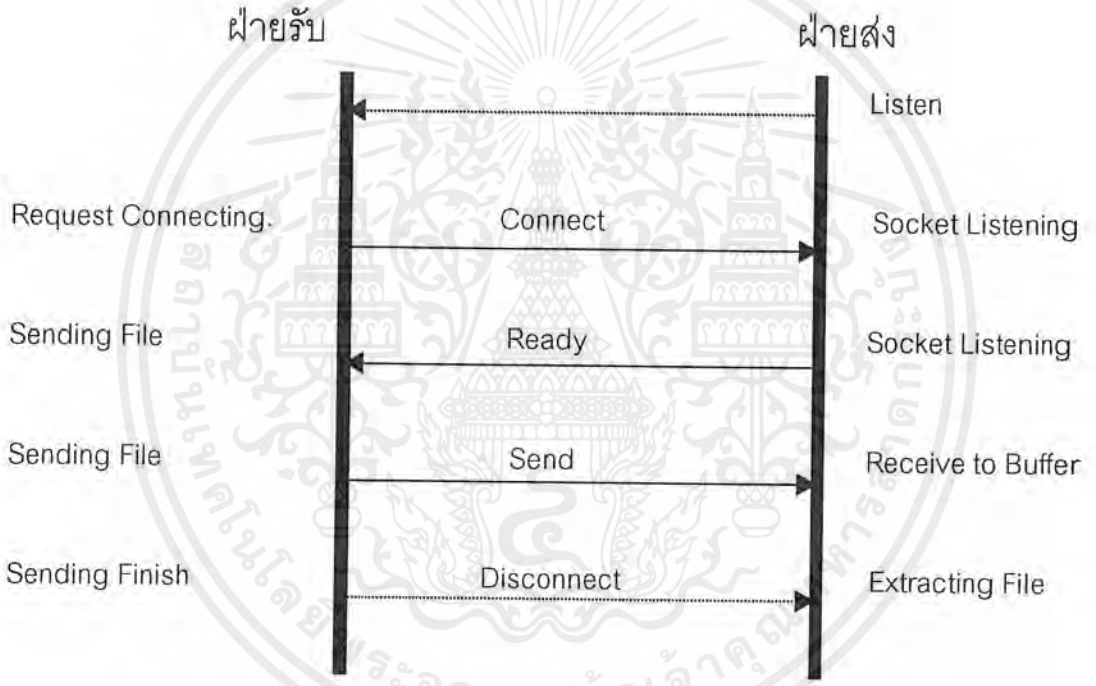
ขยาย(decompress)ตาม algorithm เมื่อทำการ decompress แล้ว file ที่ถูกขยายออกมาจะถูกจัดเก็บตาม argument ที่ถูกส่งมาในตอนแรก

3.2.6 การยกเลิกการเชื่อมต่อ (disconnect)

เป็นการยกเลิกการเชื่อมต่อที่มีอยู่

3.3 ลักษณะการทำงานระหว่างเครื่อง 2 เครื่อง

เครื่องคอมพิวเตอร์ 1 เครื่องนั้นสามารถเป็นได้ทั้งฝ่ายรับและฝ่ายส่งขึ้นกับการใช้งานของผู้ใช้ว่าต้องการจะเลือกใช้แบบใด ดังนั้นเราสามารถแสดง diagram ได้ดังนี้



รูปที่ 3.1 Diagram แสดงการติดต่อกันระหว่างทำการส่งและรับ

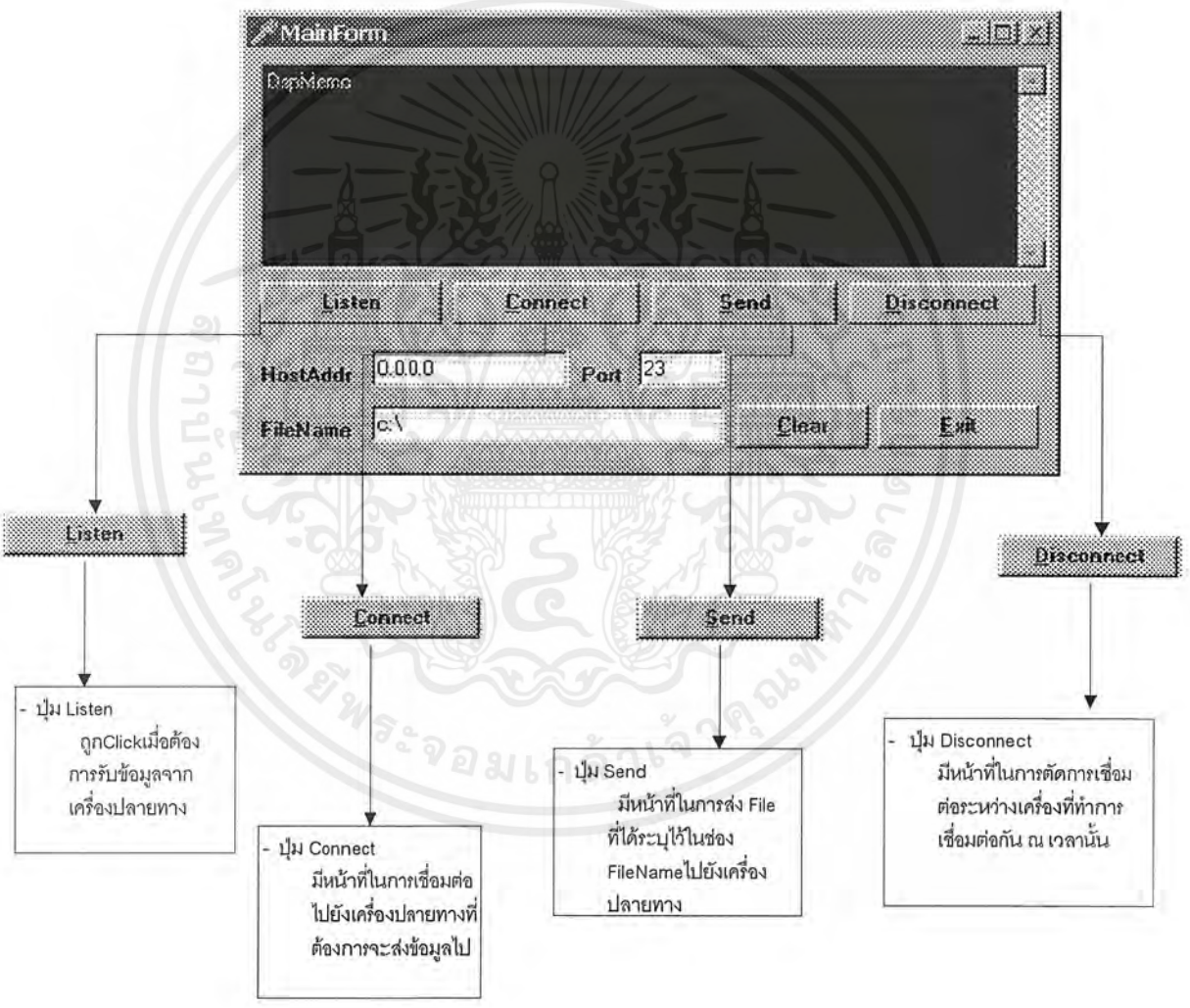
สามารถอธิบาย diagram ได้ดังนี้

- เครื่องฝ่ายรับทำการเฝ้ารอฟังการเชื่อมต่อ(listen)จากเครื่องฝ่ายส่ง
- เครื่องฝ่ายส่งเมื่อทำการ connect มาแล้วจะเกิด file ชื่อ " Outfile.bin " ที่ฝ่ายรับ
- เครื่องฝ่ายรับพร้อมที่จะรับข้อมูล
- เครื่องฝ่ายส่งจะทำการส่งข้อมูลไปยังเครื่องที่ทำการรับโดยค้นหา File ตามชื่อที่ได้ระบุไว้แล้ว ทำการบีบอัด File ข้อมูลพร้อมทั้งเปลี่ยนชื่อเป็น " Compress.Huf"
- เครื่องฝ่ายรับจะทำการรับข้อมูลมาเก็บไว้ใน File ที่ชื่อ " Outfile.bin " ซึ่งจะอยู่ใน buffer

- เครื่องฝ่ายส่งทำการ Disconnect มา ดังนั้นไฟล์ Outfile.bin ทางฝ่ายรับจะถูกขยายออก (Extracting)และจะถูกจัดเก็บลงใน Directory “Compress/Zip”

บทที่ 4
การพัฒนาโปรแกรม

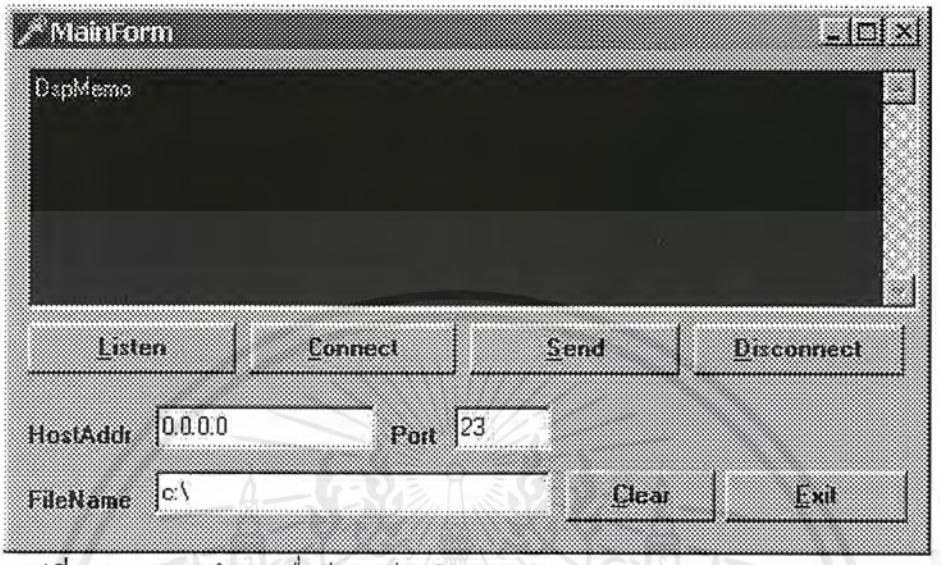
เนื่องด้วยโปรแกรม สามารถทำงานได้ทั้งเป็นฝ่ายรับและฝ่ายส่ง ดังนั้นจึงจะอธิบายพร้อมกันทั้ง 2 ฝ่ายพร้อมกัน diagram ข้างล่างนี้เป็นการแสดงถึง module ที่สำคัญใน program



รูปที่ 4.1 แสดงหน้าที่ของส่วนประกอบในหน้าจอโปรแกรม

4.1 การเรียกไฟล์

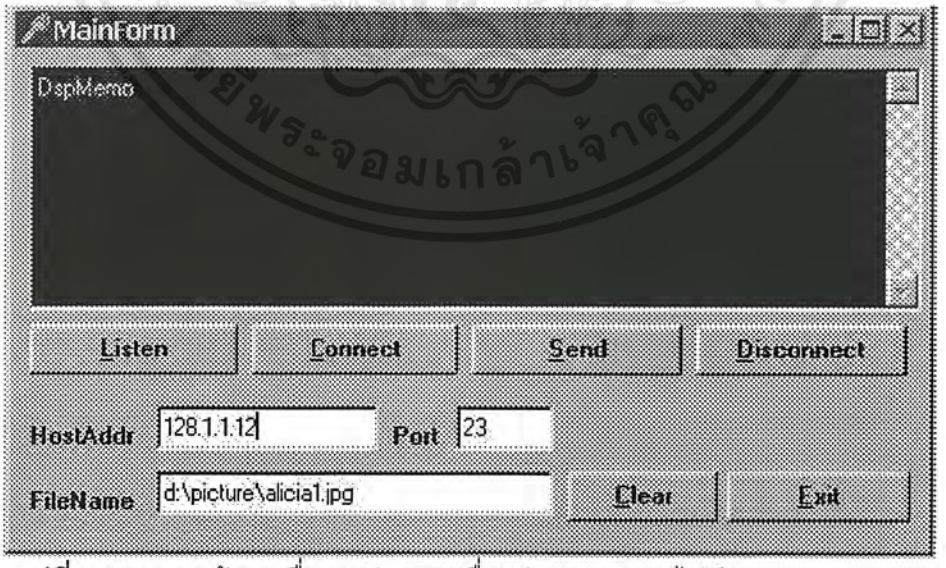
เมื่อมีการเรียกไฟล์ จะแสดงหน้าจอโปรแกรมดังรูป



รูปที่ 4.2 แสดงหน้าจอเมื่อมีการเรียกโปรแกรม

4.2 การระบุค่าของเครื่องปลายทางและไฟล์ข้อมูลที่จะส่ง

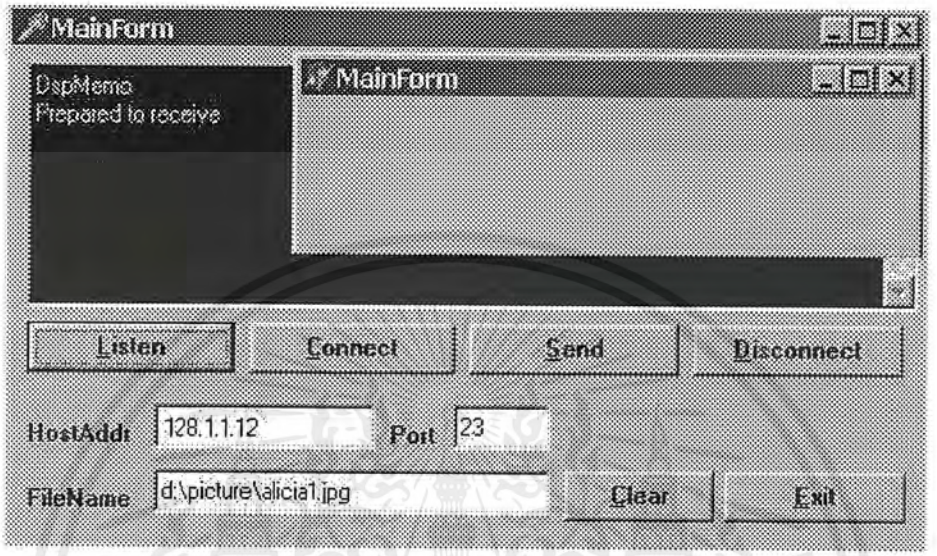
ทำการระบุค่าของเครื่องปลายทาง(Host Address)และไฟล์ข้อมูลที่จะส่ง



รูปที่ 4.3 แสดงหน้าจอเมื่อระบุค่าของเครื่องปลายทางและไฟล์

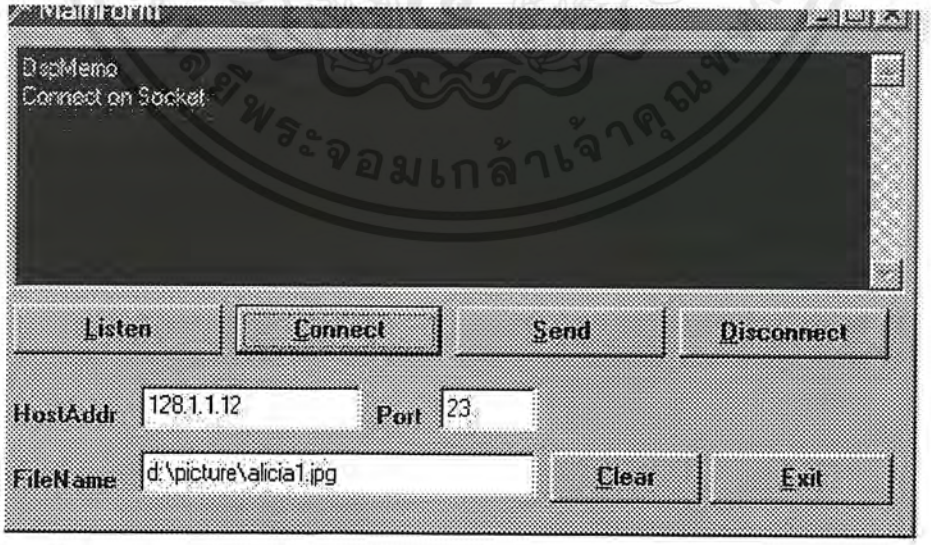
4.3 ฝ่ายรับทำการเฝ้าฟัง(listen)

เมื่อฝ่ายรับทำการเฝ้าฟัง(listen) จะปรากฏ form ขึ้นมา 1 form เป็นตัวทำหน้าที่คอยเฝ้าฟัง



รูปที่ 4.4 แสดงหน้าจอเมื่อฝ่ายรับทำการเฝ้าฟัง

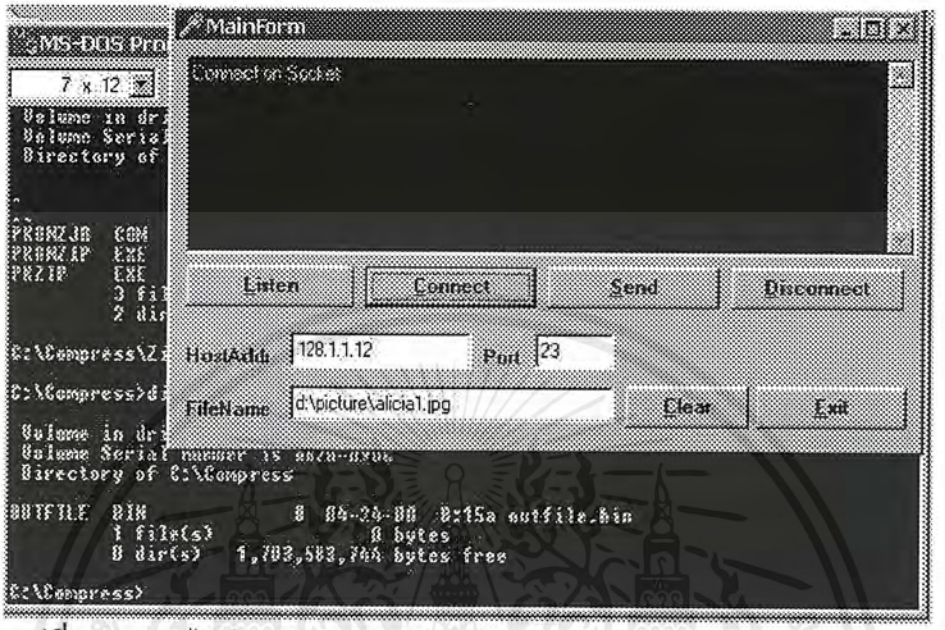
ฝ่ายส่งทำการเชื่อมต่อไปยังเครื่องปลายทางตามที่ระบุใน Host Address โดยกดปุ่ม connect



รูปที่ 4.5 แสดงหน้าจอเมื่อฝ่ายส่งทำการเชื่อมต่อ

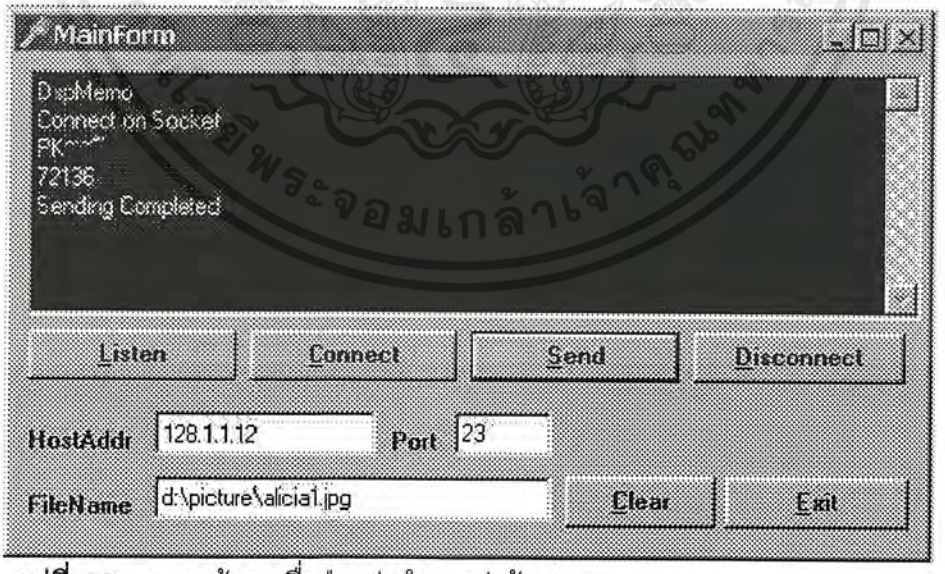
4.4 การ Connect

เมื่อเกิดการ Connect ขึ้น ฝ่ายรับจะเก็บข้อมูลในไฟล์ชื่อ"Outfile.bin" ซึ่งจะมีขนาด 0 byte



รูปที่ 4.6 แสดงไฟล์ outfile.bin ขนาด 0 byte

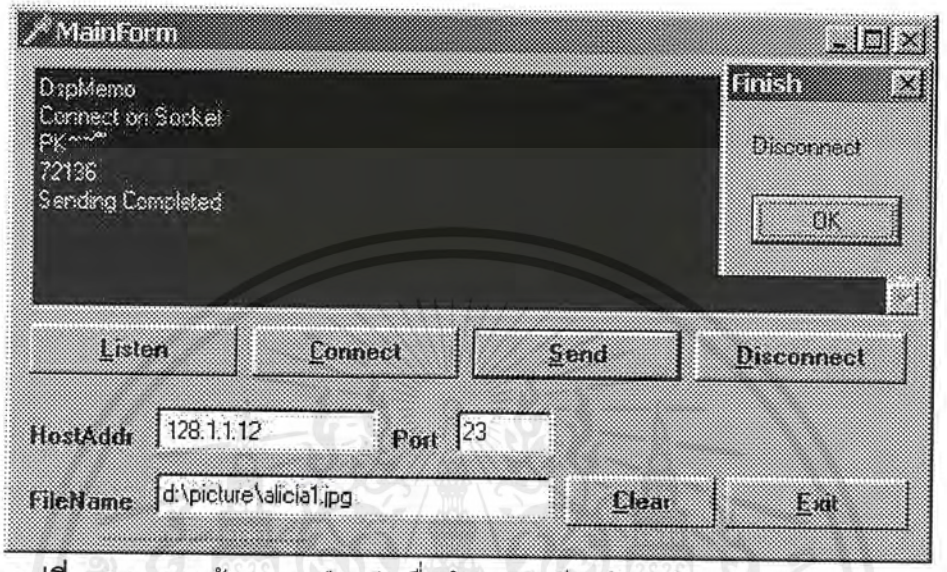
ฝ่ายส่งทำการส่งข้อมูล โดยการกดปุ่ม Send ข้อมูลจะถูกส่งไปยังปลายทางที่ได้ระบุไว้



รูปที่ 4.7 แสดงหน้าจอเมื่อฝ่ายส่งทำการส่งข้อมูล

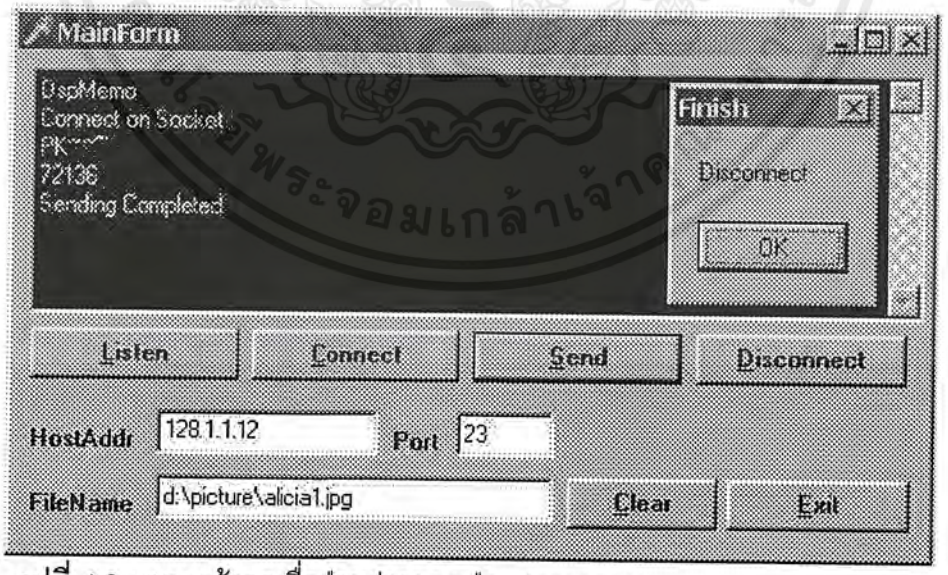
4.5 ฝ่ายรับทำการรับข้อมูล

ฝ่ายรับทำการรับข้อมูล แต่ยังไม่ทำการจัดเก็บลงใน Hard Disk จนกระทั่ง Disconnect จากฝ่ายส่งโดยการกดปุ่ม Check



รูปที่ 4.8 แสดงหน้าจอของฝ่ายส่งเมื่อทำการส่งเรียบร้อยแล้ว

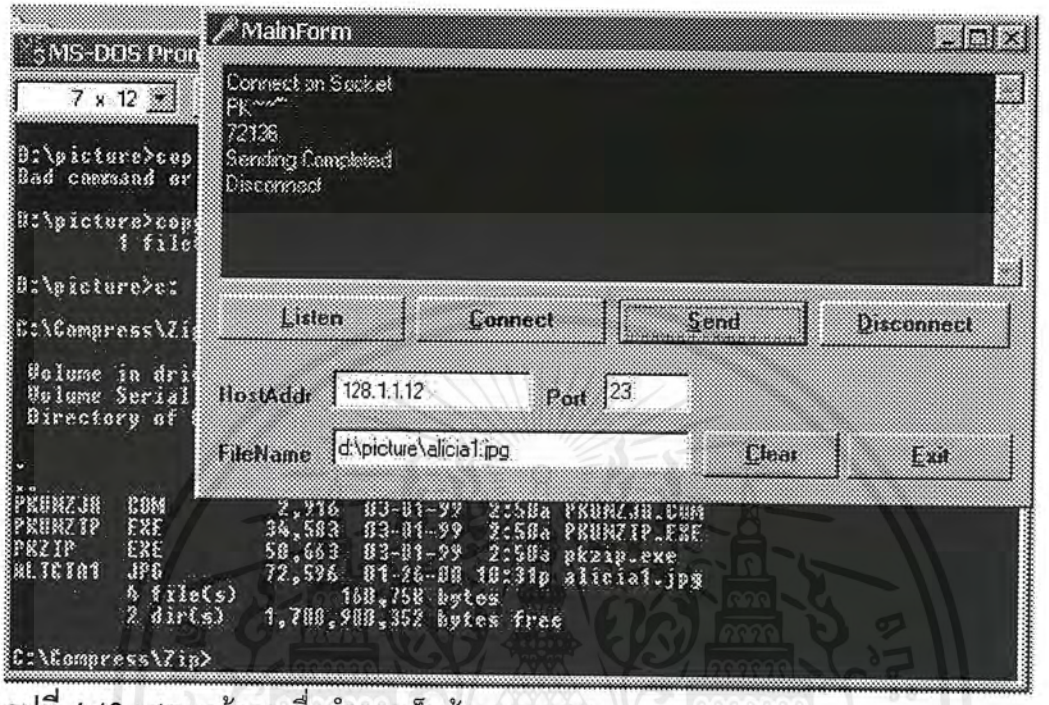
ฝ่ายส่งทำการตัดการเชื่อมต่อ โดยการกดปุ่ม Disconnect



รูปที่ 4.9 แสดงหน้าจอเมื่อฝ่ายส่งจะกดปุ่ม disconnect

4.6 การจัดเก็บข้อมูลลงใน Hard Disk

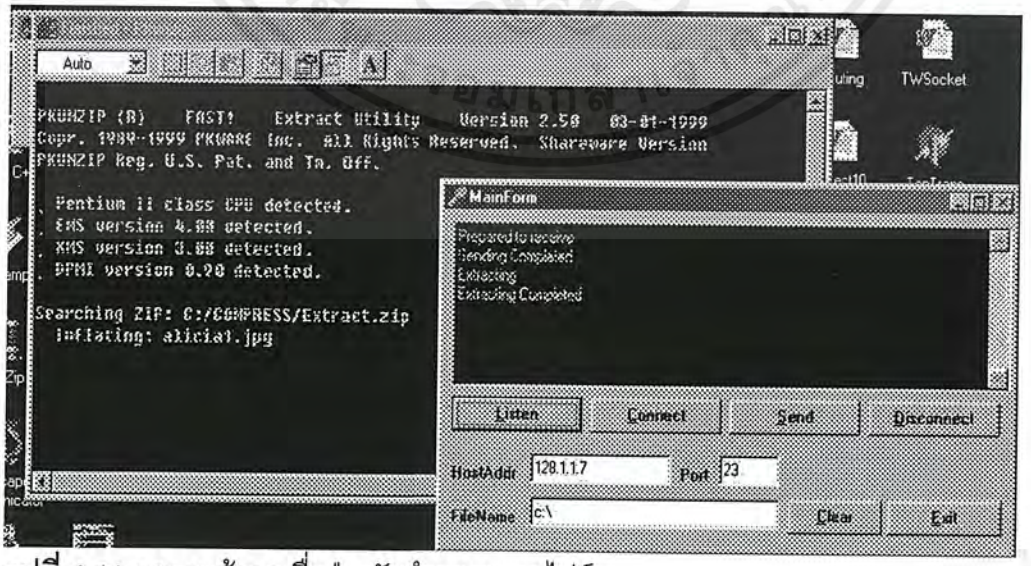
มีการจัดเก็บข้อมูลลงใน Hard Disk แล้ว โดยกดปุ่ม Disconnect



รูปที่ 4.10 แสดงหน้าจอเมื่อทำการเก็บข้อมูลลง disk

4.7 ฝ่ายรับทำการขยายไฟล์ออกมาโดยอัตโนมัติ

เมื่อได้รับไฟล์ครบเรียบร้อยแล้ว ฝ่ายรับก็จะทำการขยายไฟล์ออกมาโดยอัตโนมัติ



รูปที่ 4.11 แสดงหน้าจอเมื่อฝ่ายรับทำการขยายไฟล์

บทที่ 5
บทสรุป และเสนอแนะ

5.1 บทสรุป

จากความมุ่งหวังที่จะพัฒนาโปรแกรมการบีบอัดไฟล์แบบอัตโนมัติเพื่อเข้าจัดการกับปัญหาขนาดของข้อมูลและอำนวยความสะดวกในการส่งข้อมูลบนเครือข่าย Internet ได้ประกอบด้วยส่วนสำคัญ คือ การออกแบบและพัฒนาโปรแกรมการบีบอัดและขยายไฟล์ (Compression and Decompression) และ การออกแบบและพัฒนาโปรแกรมการรับ-ส่งข้อมูลระหว่าง IP ADDRESS โดยอ้างอิงโปรโตคอล TCP/IP

ในการพัฒนาโปรแกรมนั้นสามารถสรุปผลได้ดังนี้

5.1.1 การบีบอัด

- ถูกพัฒนาตามอัลกอริทึ่มของ Huffman และ LZ ด้วยภาษา C++
- สามารถลดขนาดของข้อมูลลงได้ตามอัตราส่วนตามแต่ชนิด ของไฟล์นั้นๆ ว่าเป็นไฟล์ชนิดใด และ ตามแต่ อัลกอริทึมที่ใช้ ซึ่ง หากใช้วิธี Huffman จะได้ผลดีกว่าวิธี LZ
- การใช้อัลกอริทึม LZ กับไฟล์ประเภท กราฟฟิก , ภาพ ,เสียง จะบีบอัดได้ไม่มาก จะเหลือไฟล์ขนาดใหญ่ และในบางกรณี อาจได้ไฟล์หลังการบีบอัดที่ใหญ่กว่าไฟล์เดิมที่ไม่ได้รับการบีบอัดเสียอีก
- สามารถบีบอัดข้อมูลโดยอัตโนมัติ ก่อนที่จะส่งออกไปจากเครื่องที่เป็นฝ่ายส่งได้ แต่ต้องให้ผู้ใช้ที่เครื่องที่เป็นฝ่ายส่ง ทำการกดปุ่มตัดการติดต่อ เพื่อปล่อยการควบคุมไฟล์ที่ส่งนั้น เพื่อให้เครื่องที่เป็นฝ่ายรับ สามารถขยายข้อมูลได้และให้เครื่องที่เป็นฝ่ายรับ คอยตรวจสอบขนาดของไฟล์บีบอัดที่ถูกส่งมาเอง
- เวลาที่ใช้ในการส่งไฟล์ที่บีบอัดแล้ว จะใช้เวลาน้อยกว่าการส่งไฟล์เดิมที่ไม่ได้รับการบีบอัด แต่ก่อนที่จะส่ง จะต้องเสียเวลาในกระบวนการบีบอัด ซึ่งเวลาที่ใช้ในการบีบอัดนั้นมากหรือน้อย จะขึ้นอยู่กับขนาดของไฟล์เดิมว่าใหญ่หรือเล็ก ดังนั้น ถ้ารวมเวลาทั้งการบีบอัดและการส่งแล้ว อาจจะทำให้ใช้เวลามากกว่าการส่งไฟล์นั้นแบบธรรมดา โดยไม่ต้องทำการบีบอัดก่อน
- ตารางในหน้าต่อไปนี้ แสดงให้เห็นการเปรียบเทียบของการบีบอัดไฟล์ข้อมูลว่ามีประสิทธิภาพต่างกันอย่างไร ซึ่งจะมีประสิทธิภาพดังที่ได้กล่าวมาแล้ว

ตารางที่ 5.1 แสดงการเปรียบเทียบประสิทธิภาพการบีบอัดข้อมูลด้วยวิธีต่างๆ

File Type	Original Size	Compress Algorithm (file remain)		
		Pkzip (%)	Huff (%)	LZ (%)
.DOC	69.5 K	***19 (13.3 K)	**54 (38 K)	*79 (54.8 K)
.TXT	32 K	***14.6 (132 K)	**39 (12.5 K)	*66 (21.2 K)
.EXE	92 K	***53.5 (49.3 K)	**71.6 (65.9 K)	*87 (80.3 K)
.COM	91.6 K	***45.3 (41.5 K)	**71.7 (65.7 K)	*93.3 (85.5 K)
.BMP	64 K	***19.8 (12.7 K)	**34 (21.9 K)	*73 (47.2 K)
.GIF	101 K	**97.8 (98.8 K)	***97.6 (98.6 K)	*143 (145 K)
.JPG	135 K	***97 (132 K)	**100 (135 K)	*143 (194 K)
.VSD	38 K	***76 (29.1 K)	**87 (33.3 K)	*123 (46.8 K)
.XLS	29 K	***29 (25.8 K)	**63.6 (55.1 K)	*83 (72 K)
.MP3	3.17 M	***97 (3.09 M)	**99 (3.14 M)	*144 (4.57 M)
.VCD	64 K	***10 (195 bytes)	**12.5 (8.01 K)	*97 (62.2 K)

***Num ประสิทธิภาพดีที่สุด **Num ประสิทธิภาพดี

*Num ประสิทธิภาพไม่ดี

จาก ตาราง การวัดประสิทธิภาพ อ้างอิงจากการเปรียบเทียบเพียง 3 Algorithm เท่านั้น

5.1.2 การเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์

- ถูกพัฒนาด้วย Borland Delphi 3.0 Client/Server suit
- เป็นโปรแกรมที่สามารถทำงานได้ทั้งการรับและการส่งข้อมูล
- ความเร็วในการส่งข้อมูลขึ้นอยู่กับขนาดไฟล์ที่จะทำการส่ง และ ขนาดของบัฟเฟอร์
- ในโปรแกรมจะระบุขนาดบัฟเฟอร์ เท่ากับ 2048 bytes เมื่อเปรียบเทียบกับขนาด 256 bytes แล้วสามารถทำงานได้เร็วกว่า
- สามารถส่งข้อมูลผ่านระหว่างเครื่องสองเครื่องที่เชื่อมต่อกัน โดยการกำหนดค่าตำแหน่ง (IP ADDRESS) ของเครื่องให้กับโปรแกรมได้

5.2 ข้อเสนอแนะ

ในการทำปัญหาพิเศษในหัวข้อการพัฒนาโปรแกรมรับ-ส่งไฟล์แบบบีบอัดอัตโนมัติบนเครือข่าย Internet ได้มีการศึกษาการทำงานในส่วนของการบีบอัดไฟล์และหลักการรับ-ส่งข้อมูลผ่านเครือข่าย ที่นำมาใช้ในปัญหาพิเศษนี้ เนื่องจากเวลาในการทำงานจำกัด จึงไม่สามารถที่จะศึกษาได้อย่างละเอียด ส่วนของงานที่ควรเพิ่มเติมเพื่อให้ระบบสมบูรณ์ ได้แก่

5.2.1 การบีบอัดข้อมูล

- การบีบอัดและการขยายข้อมูล ควรมีประสิทธิภาพดีขึ้น สามารถรองรับไฟล์ได้หลายชนิด และอัตราส่วนในการบีบอัดควรสูงกว่านี้ รวมถึงใช้เวลาในการบีบอัดที่น้อยกว่านี้ด้วย
- เพื่อให้ประสิทธิภาพของการส่งโดยรวมที่ดีกว่า ควรนำอัลกอริทึม Huffman และ LZ ไปปรับปรุงเพื่อให้ได้ประสิทธิภาพทั้งด้านเวลาและขนาดที่ดีกว่านี้ หรือใช้อัลกอริทึมอื่นที่มีประสิทธิภาพดีกว่านี้
- ทดสอบโดยการนำเอาไฟล์ที่ถูกบีบอัดด้วยวิธีต่างๆมาบีบอัดอีกครั้ง

5.2.2 การเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์

- ทดลองหาขนาดของบัฟเฟอร์ ที่เหมาะสม สามารถทำงานได้เร็วและส่งข้อมูลผ่านอุปกรณ์เครือข่ายได้หลายชนิด เช่น Router, Switch, Gateway
- Multithreadผู้ที่ทำการรับข้อมูลควรที่จะรับข้อมูลได้จากหลายๆผู้ใช้ที่ทำการเชื่อมต่อผ่านเครือข่ายในขณะนั้น
- กำหนดส่วนที่แสดงถึง การตอบสนองต่างๆกับผู้ใช้ ให้มีความชัดเจนมากขึ้น
- รับไฟล์และเส้นทางที่เก็บไฟล์ โดยให้ชื่อยาวกว่าแปดตัวอักษรได้

บรรณานุกรม

จิรพัฒน์ จันทร์เจิดศักดิ์ และ วีระ นพนิราพาธ. 2521. **การเขียนโปรแกรมบน Microsoft Windows.** กรุงเทพฯ : ซีเอ็ดยูแคชั่น.

David Petorson, M. n.d. **TCP/IP Networking series advisor,**A Guide to the IBM Enviroment : McGraw-Hill International Edition.

Mark Nelson and Jea-Loup Gailly. 1996. **The Data compression Book.** 2nd edition New York : M&T Books

Govalski Walter. 1995. **TCP/IP Application and Protocol.** Charleston South Carolina : Computer Technology Research

Xavier Pacheco and Steve Teixerira. 1995. **Delphi Developer's Guide :** SAMS Publishing

Data Compression Reference Center **"Compression Algorithm"** [Online] , Available : http://www.rasip.fer.hr/resource/compress/index.html