การรับส่งภาพผ่าน Ethernet

IMAGE ON ETHERNET

โดย

นางสาวปนิดา ศรีสุวรรณ

นางสาวอัญชลา ราศรีมินทร์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

การรับส่งภาพผ่าน Ethernet

IMAGE ON ETHERNET

โดย

นางสาวปนิดา ศรีสุวรรณ 35104243

นางสาวอัญชลา ราศรีมินทร์ 35104550

อาจารย์ที่ปรึกษา

ผศ.ดร. สุวิพล สิทธิชีวภาค

อ.นภัทร สระเอี่ยม

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การรับส่งภาพผ่าน Ethernet

Image on Ethernet

ผู้จัดทำ

1. นางสาวปนิดา ศรีสุวรรณ 35104243
2. นางสาวอัญชลา ราศรีมินทร์ 35104550

..... อาจารย์ที่ปรึกษา

(ผศ.ดร. สุวิพล สิทธิชีวภาค)

..... อาจารย์ที่ปรึกษา

(อ. นภัทร สระเอี่ยม)

การรับส่งภาพผ่าน Ethernet

Image on Ethernet

โดย นางสาวปนิดา ศรีสุวรรณ รหัส 35104243
นางสาวอัญชลา ราศรีมินทร์ รหัส 35104550

อาจารย์ที่ปรึกษา ผศ.ดร.สุวิพล สิทธิชีวภาค
อาจารย์นภัทร สระเอี่ยม

บทคัดย่อ

การติดต่อสื่อสารระหว่างไมโครคอมพิวเตอร์ ในระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น เป็นที่นิยมใช้งานกันอย่างแพร่หลายในปัจจุบัน ไม่ว่าจะเป็นการติดต่อสื่อสารระหว่างผู้ใช้ การแลกเปลี่ยนข่าวสารข้อมูล ซึ่งมีขีดความสามารถในการทำงานเพิ่มขึ้นอย่างมาก แต่ก็ยังมีข้อจำกัดที่ว่า การติดต่อสื่อสารดังกล่าวส่วนใหญ่แล้ว มักจะเป็นการติดต่อสื่อสารกันในรูปของข้อมูลที่เป็นตัวอักษร โครงงานนี้จึงมีแนวความคิดที่จะทำให้การติดต่อสื่อสารผ่านระบบเครือข่ายมีความหลากหลายมากยิ่งขึ้น ด้วยการเขียนโปรแกรมเพื่อทำการรับและส่งภาพผ่านระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น แบบ Ethernet โดยอาศัยหลักการเขียนโปรแกรมภายใต้ข้อกำหนดในการเขียนโปรแกรมติดต่อกับระบบจัดการเน็ตแวร์ และการเรียกใช้ฟังก์ชันในบริการของเน็ตแวร์

ABSTRACT

The communication between microcomputers on LAN is very popular and useful. The users can receive and transmit message or information through the high speed network , but almost communicate with text. This project is improved to increase efficiency of network by receiving and transmitting the image on Ethernet LAN. These programes are run under Netware Operating System and programming under Netware API, including the used function so called in Netware Service.

สารบัญ

สารบัญรูปภาพ

ลำดับภาพ **หน้า**

สารบัญผังงาน

ลำดับที่ **หน้า**

บทที่ 4

บทที่ 1
บทนำ

ในปัจจุบันเทคโนโลยี ได้ก้าวหน้าไปอย่างรวดเร็ว ประกอบกับราคาที่ลดลงอย่างรวดเร็วของคอมพิวเตอร์ ทำให้การมีคอมพิวเตอร์ไว้สำหรับใช้งานดูเหมือนจะเป็นเรื่องธรรมดา รวมทั้งมีความสำคัญในการทำงานต่างๆมากกว่าแต่ก่อน ในหลายๆหน่วยงานได้มีการนำเอาระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่นมาใช้ เนื่องด้วยขีดความสามารถที่สูง ความเร็วในการทำงาน ประกอบกับสามารถใช้ทรัพยากรต่างๆร่วมกันได้ ระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่นจึงเป็นระบบที่น่าสนใจมากทีเดียว แต่โดยส่วนใหญ่แล้วการติดต่อสื่อสารในระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่นนี้ จะเป็นการติดต่อสื่อสารในรูปของข้อมูลที่เป็นตัวอักษร โครงงานนี้จึงมีแนวความคิดที่จะทำให้การติดต่อสื่อสารผ่านระบบเตรือข่าย มีความหลากหลายมากยิ่งขึ้น ด้วยการเขียนโปรแกรมเพื่อทำการรับและส่งภาพผ่านระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น

สำหรับในการเขียนโปรแกรมเพื่อทำการรับและส่งภาพผ่านระบบเครือข่ายนั้น จะต้องอาศัยหลักการเขียนโปรแกรมภายใต้ข้อกำหนดในการเขียนโปรแกรมติดต่อกับระบบจัดการเน็ตแวร์ และการเรียกใช้ฟังก์ชันในบริการของเน็ตแวร์ โดยวิธีในการส่งได้เลือกใช้ลักษณะการส่งข้อความแบบกระจาย (Broadcast Message) ซึ่งจะสามารถส่งได้ความยาวสูงสุด 55 ไบต์ต่อ 1 ครั้ง ซึ่งโปรแกรมที่ได้ทำการเขียนขึ้นนี้ สามารถทำการรับส่งภาพได้ใน 3 ลักษณะคือ

1. รับและส่งภาพขาวดำ ในลักษณะ on-line
2. รับและส่งภาพขาวดำ ในลักษณะ off-line
3. รับและส่งภาพสีในลักษณะ off-line

นับเป็นอีกมิติหนึ่งของการสื่อสารผ่านระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น ซึ่งจะเป็นจุดเริ่มต้นสู่การพัฒนาและประยุกต์ใช้ต่อไป

ทั้งนี้จุดประสงค์ของโครงงานนี้ก็เพื่อก่อให้เกิดความหลากหลายขึ้น ในการติดต่อสื่อสารผ่านระบบเครือข่ายคอมพิวเตอร์ แบบท้องถิ่น และนำไปสู่แนวทางในการพัฒนาเพื่อเป็นการเพิ่มขีดความสามารถในการติดต่อสื่อสารผ่านระบบเครือข่ายต่อไป.

บทที่ 2
ระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น

แนวโน้มการใช้งานระบบเครือข่ายคอมพิวเตอร์ในปัจจุบันนั้น ได้มีการนำระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น (Local Area Network) มาใช้กันอย่างแพร่หลาย เนื่องจากมีความเร็วในการทำงานที่สูงขึ้นมาก การส่งข้อมูลทำได้ด้วยความเร็วสูงผิดพลาดน้อย รวมถึงสามารถทำการขยายระบบได้ง่าย อีกทั้งมีค่าใช้จ่ายที่ไม่แพงนัก ระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่นจึงถูกใช้อย่างแพร่หลายดังที่เห็นในปัจจุบัน

2.1 ลักษณะทั่วไปของระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น

ระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่นนี้ เป็นการเชื่อมต่อคอมพิวเตอร์เข้าด้วยกันในระยะใกล้ๆ ไม่เกิน 10 กิโลเมตร ให้กลายเป็นระบบเครือข่ายคอมพิวเตอร์เพื่อที่จะสามารถส่งผ่านข่าวสารข้อมูลระหว่างกันได้ โดยใช้แพ็กเก็ตข่าวสาร

ระบบเครือข่ายแบบท้องถิ่น จะประกอบไปด้วยส่วนต่าง ๆคร่าว ๆ ดังนี้ คือ

สถานีผู้ใช้ (Client) เป็นเครื่องคอมพิวเตอร์ซึ่งใช้งานกันทั่วไป โดยต่อเชื่อมอยู่ในแต่ละโหนดของระบบเครือข่ายคอมพิวเตอร์

ระบบจัดการของสถานีผู้ใช้ เป็นโปรแกรมควบคุมสถานีผู้ใช้ให้สามารถติดต่อกับระบบเครือข่ายคอมพิวเตอร์ได้

การ์ดเชื่อมต่อระบบเครือข่ายคอมพิวเตอร์ เป็นการ์ดติดต่อสำหรับเชื่อมต่อสถานีผู้ใช้เข้ากับระบบเครือข่ายคอมพิวเตอร์

ศูนย์บริการข้อมูล (Server) เป็นเครื่องคอมพิวเตอร์ศูนย์กลางของระบบเครือข่ายคอมพิวเตอร์ ที่มีทรัพยากรของระบบเครือข่ายคอมพิวเตอร์สำหรับให้สถานีผู้ใช้แต่ละเครื่องใช้งานร่วมกัน เช่น เครื่องพิมพ์ เป็นต้น

ระบบจัดการของศูนย์บริการข้อมูล เป็นโปรแกรมควบคุมศูนย์บริการข้อมูลให้สามารถจัดการข้อมูลและทรัพยากรต่างๆ ตามที่สถานีผู้ใช้แต่ละเครื่องร้องขอมาได้

สายเคเบิ้ล สำหรับนำสัญญาณที่ใช้ในการเชื่อมต่อ ซึ่งมีหลายลักษณะที่แตกต่างกันออกไป เช่น สายสัญญาณแบบ UTP (Unshield Twisted Pair) หรือ เส้นใยนำแสง (Optic Fiber)

เครื่องคอมพิวเตอร์แต่ละเครื่องในระบบเครือข่ายสามารถติดต่อสื่อสารกันได้โดยใช้แพ็กเก็ตข่าวสาร ในแพ็กเก็ตจะมีข้อมูลซึ่งบอกตำแหน่งของผู้ส่งแพ็กเก็ตนั้นและผู้รับแพ็กเก็ต ซึ่งจะนำส่วนดังกล่าวมาใช้เพื่อการจัดเส้นทาง การจัดระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่นโดยส่วนใหญ่แล้ว จะต้องมีศูนย์บริการข้อมูลอย่างน้อย 1 เครื่อง อย่างไรก็ตาม การทำงานต่างๆจะกระทำที่เครื่องสถานีผู้ใช้นั้น ไม่ใช่บนศูนย์บริการ

บริการข้อมูล และภายใต้ระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่นนั้นจะมีระบบที่ช่วยให้เครื่องสถานีผู้ใช้แต่ละเครื่องสามารถใช้แฟ้มข้อมูลร่วมกัน การป้องกันเรคคอร์ด การส่งข่าวสารถึงสถานีผู้ใช้เครื่องอื่นๆ

เทคนิคในการส่งข้อมูลในระบบเครือข่ายแบบท้องถิ่น สามารถแบ่งได้ 2 แบบ คือ

1. **ระบบแบบเบสแบนด์** จะมีช่องทางสื่อสารเพียงช่องทางเดียว การส่งข้อมูลในระบบเครือข่ายจะส่งข้อมูลด้วยความเร็งสูง จึงต้องมีเทคนิคในการจัดการข้อมูลเพื่อป้องกันการชนกันของข้อมูล โดยเทคนิคที่ใช้กันอย่างแพร่หลายก็คือ CSMA (Carrier Sense Multiple Access)

2. **ระบบแบบบรอดแบนด์** เป็นการส่งข้อมูลหลายช่องทาง ด้วยความถี่ที่ต่างกันโดยใช้สัญญาณของคลื่นวิทยุในการส่งข้อมูลสัญญาณเดียวหรือหลายสัญญาณ บนสายส่งข้อมูลเส้นเดียวกัน ข้อมูลในระบบจะถูกส่งผ่านช่องทางที่มีความถี่ในการรับและส่งข้อมูลต่างกัน

2.2 ความสามารถของระบบเครือข่ายท้องถิ่น

ระบบเครือข่ายท้องถิ่นมีข้อดีเหนือมินิคอมพิวเตอร์หรือเมนเฟรม กล่าวคือ มีการประมวลผลแบบกระจายระบบงาน (distributed processing) ความรวดเร็วในการติดต่อสื่อสาร การติดต่อระหว่างสถานีผู้ใช้งาน (interconnected station) ใช้โปรแกรมและข้อมูลร่วมกันได้ ใช้ฮาร์ดแวร์ และทรัพยากรที่มีอยู่ร่วมกันได้

การประมวลผลแบบกระจายงาน ในการประมวลผลแบบกระจาย เมื่อสถานีของผู้ใช้บริการขอโปรแกรม และข้อมูลจาก *ศูนย์บริการข้อมูล (file server)* โปรแกรมคำสั่งจะถูกสำเนาจากศูนย์บริการข้อมูลไปยังหน่วยความจำที่สถานีของผู้ใช้ เมื่อเสร็จสิ้นการใช้งานโปรแกรมและไฟล์ข้อมูลแล้ว สถานีผู้ใช้จะส่งข้อมูลกลับไปเก็บยังศูนย์บริการข้อมูลเพื่อใช้งานต่อไป

การประมวลผลแบบกระจายงาน จะช่วยให้สถานีหลายๆ สถานีใช้งานร่วมกันในระบบได้ โดยไม่ไปลดความสามารถในการประมวลผลข้อมูลของแต่ละสถานี ในเครื่องคอมพิวเตอร์แบบเมนเฟรม ความสามารถในการประมวลผล จะถูกแบ่งออกไปในแต่ละสถานี ถ้ายิ่งมีสถานีมากก็จะทำให้ประสิทธิภาพการทำงานลดลง ในการกระจายการประมวลผลข้อมูลจะทำให้เพิ่มความเร็วและประสิทธิภาพของระบบได้

ความรวดเร็วในการติดต่อสื่อสาร ระบบเครือข่ายท้องถิ่นมีการใช้สื่อการส่งข้อมูลของตนเอง (โดยปกติจะเป็นสายส่งข้อมูล) มากกว่าที่จะใช้สื่อสาธารณะเช่น สายโทรศัพท์ ทำให้การติดต่อสื่อสารในระบบเครือข่ายมีความเร็วสูงมากกว่า 1 ล้านบิตต่อวินาที ขึ้นไปในสายส่งข้อมูลความเร็วสูง นอกจากนี้ระบบเครือข่ายท้องถิ่นยังมีช่องทางที่เรียกว่า *ประตูสื่อสาร (gateway)* เชื่อมต่อกับระบบเครือข่ายระดับประเทศ (WAN:Wide Area Network) ในการ ติดต่อสื่อสารอื่นๆ อีกด้วย

การติดต่อระหว่างสถานีผู้ใช้งาน เมื่อแต่ละสถานีเชื่อมต่อเข้าด้วยกันแล้วจะทำให้ผู้ใช้ติดต่อหรือส่งข้อมูลให้แก่กันได้ เช่น การใช้จดหมายอิเล็กทรอนิกส์ (electronic mail) และช่วยให้ผู้ใช้จัดการข้อมูลที่อื่นได้

การใช้โปรแกรมร่วมกัน ในระบบเครือข่ายผู้ใช้สามารถจะใช้ข้อมูลและซอฟต์แวร์ร่วมกันได้ เพื่อลดความซ้ำซ้อนของข้อมูล สื่อบันทึกข้อมูล ตลอดจนการค้นหาตรวจสอบข้อมูลได้รวดเร็วและถูกต้อง การปรับปรุงโปรแกรมที่ใช้งานร่วมกันก็สามารถกระทำ ณ จุดเดียว

การใช้ทรัพยากรร่วมกัน เนื่องจากระบบเครือข่ายท้องถิ่นอนุญาตให้ผู้ใช้ระบบเครือข่ายใช้อุปกรณ์ร่วมกัน (เช่น เครื่องพิมพ์ โมเด็ม ประตูสื่อสาร อุปกรณ์ด้านกราฟิก และอุปกรณ์การเก็บข้อมูล เป็นต้น) ซึ่งจะช่วยลดความซ้ำซ้อนของอุปกรณ์ทำให้ประหยัดค่าใช้จ่าย นอกจากนี้ระบบเครือข่ายท้องถิ่นยังช่วยให้ผู้จัดการระบบสามารถตรวจสอบการใช้อุปกรณ์ต่างๆ เหล่านี้จากผู้ใช้ในระบบได้อีกด้วย

ตัวกลางในการส่งข้อมูล หรือ *สายส่งข้อมูล* มีอยู่หลายแบบ เช่น โคแอกเชียล (coaxial) สายโทรศัพท์ (twisted pair) และเส้นใยนำแสง (fiber optic) เป็นต้น ซึ่งสามารถส่งข้อมูลได้ด้วยความเร็วสูงเป็นล้านบิตต่อวินาที และอุปกรณ์ที่ต่อเชื่อมกับสายส่งข้อมูลเหล่านี้ไม่จำเป็นต้องมีศูนย์สลับสาย (switching center) ซึ่งมีราคาแพง

ระบบเครือข่ายท้องถิ่นได้พัฒนาขึ้นมาหลายแบบ แต่ที่นิยมใช้กันมากเป็นระบบเครือข่ายท้องถิ่นแบบ ISO-OSI (Open System Interconnection) ซึ่งมีคุณสมบัติต่างๆ ดังนี้

1. มีความยืดหยุ่นสูง ง่ายต่อการเปลี่ยนแปลงโครงสร้าง ในการเพิ่มหรือลดอุปกรณ์ต่างๆ ในระบบเครือข่าย
2. เป็นโครงสร้างที่ขยายสาขาพิเศษออกไปได้เมื่อต้องการ
3. ตัวกลางในการส่งข้อมูล และระบบควบคุม สามารถขยายออกไปได้ง่าย
4. เป็นระบบที่มีความเชื่อถือได้สูง ถ้าสายส่งข้อมูลหรือเครื่องใดเสียหายจะไม่ทำให้การส่งหรือรับข้อมูลของอุปกรณ์อื่นเสียหายไปด้วย
5. เป็นระบบที่กระจายการควบคุม (distributed control) เพื่อเพิ่มความเชื่อถือได้ของข้อมูล

2.3 องค์ประกอบของระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น

ระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่นมีองค์ประกอบหลักๆ 3 ส่วนดังนี้

2.3.1 อุปกรณ์ด้านฮาร์ดแวร์

ฮาร์ดแวร์ที่ใช้เชื่อมต่อระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น ส่วนใหญ่จะออกแบบมาเป็นการ์ดหรือแผงวงจรที่ใส่ลงในช่องสล็อต (Slot) ของไมโครคอมพิวเตอร์ โดยลักษณะทางฮาร์ดแวร์นี้สามารถแบ่งออกเป็น 3 ลักษณะ หรือ 3 มาตรฐาน คือ Ethernet,Arcnet และ Token-Ring ซึ่งแต่ละแบบมีรายละเอียดดังจะได้กล่าวต่อไปในหัวข้อ 2.4.3

Network Interface Card (NIC) หรือการ์ด LAN แต่ละรุ่นจะทำมาให้ใช้เฉพาะกับระบบแบบใดแบบหนึ่งใน 3 แบบนี้เท่านั้น จะใช้การ์ดร่วมกันระหว่างระบบเครือข่ายแบบท้องถิ่นต่างระบบไม่ได้เลย

2.3.2 โปรแกรมควบคุมระบบ หรือซอฟท์แวร์

โปรแกรมควบคุมระบบเครือข่ายหรือที่เรียกว่า "Network Operating System" หรือ "NOS" ก็เป็นอีกส่วนหนึ่งที่มีความสำคัญมาก เพราะการที่ระบบจะรู้ว่าข้อมูลที่ใช้งานอยู่ในเครื่องคอมพิวเตอร์ของตนเอง หรืออยู่ที่คอมพิวเตอร์เครื่องอื่น ๆ ในระบบ โปรแกรมควบคุมระบบเครือข่ายจะเป็นผู้จัดการ ดังนั้น โปรแกรมควบคุมระบบที่ดีจะต้องทำให้ผู้ใช้ไม่เห็นความแตกต่างของการใช้งาน ไม่ว่าจะอยู่ในเครื่องคอมพิวเตอร์ของตนเอง หรือทำงานโดยใช้ข้อมูลจากเครื่องคอมพิวเตอร์อื่นในระบบ ซึ่งโปรแกรมควบคุมระบบเครือข่ายแบบท้องถิ่นที่มีใช้ทั่วไปสามารถแบ่งการทำงานออกได้เป็น 2 ประเภท คือ

ประเภทแรก ได้แก่ โปรแกรมควบคุมระบบเครือข่ายที่ทำงานภายใต้ระบบปฏิบัติการอื่น (เช่น DOS, OS/2 เป็นต้น) ตัวอย่างเช่น OS/2 LAN Manager ทำงานภายใต้ระบบปฏิบัติการ OS/2 เป็นต้น

ประเภทที่สอง ได้แก่ โปรแกรมควบคุมระบบเครือข่ายแบบท้องถิ่นที่สร้างระบบปฏิบัติการของตนเองขึ้นมา เช่น NetWare, Vines เป็นต้น

2.3.3 ตัวกลางนำข้อมูล

ตัวกลางนำข้อมูลสำหรับระบบเครือข่ายจะแบ่งตามจุดประสงค์การใช้งาน ซึ่งสามารถแบ่งออกได้เป็น 3 ประเภท คือ

- สายคู่ตีเกลียวทั้งแบบชิลด์และไม่มีชิลด์
- สายโคแอกเซียล
- ใยแก้วนำแสง

2.4 ลักษณะโครงสร้างของระบบเครือข่ายแบบท้องถิ่น

โครงสร้างของระบบจะมีประเด็นที่ควรพิจารณาอยู่หลายประการคือ ลักษณะการต่อสาย หรือ topology, วิธีแบ่งเวลาการใช้สาย หรือ media access control และมาตรฐานทางฮาร์ดแวร์

2.4.1 ลักษณะการต่อสาย (Topology)

ลักษณะการต่อสาย ของเครือข่ายแบบท้องถิ่น ที่เรียกว่า topology ทั่วไปมีอยู่ 3 แบบใหญ่ๆ

1. แบบดาว (Star) คือทุกเครื่องต่อกับอุปกรณ์ที่อยู่ตรงกลางเพียงตัวเดียว เมื่อเครื่องหนึ่งในระบบจะติดต่อกับเครื่องอื่นๆ ก็ต้องผ่านตัวกลางนั้น

2. แบบบัส (Bus) คือใช้สายต่อทุกเครื่องเข้าหาสายใหญ่ที่อยู่ตรงกลางหรือ bus และที่ปลายทั้งสองด้านจะปิดด้วยตัวเทอร์มิเนเตอร์(Terminator)

3. แบบวงแหวน (Ring) คือจะร้อยสายส่งเป็นวงผ่านทุก ๆ เครื่องในระบบจนครบ การส่งข้อมูลจะส่งออกมาในวงแหวน แต่ละเครื่องที่อยู่ระหว่างทางก็จะช่วย ๆ กันส่งข้อมูลที่ผ่านมาต่อเนื่องกันไปจนกว่าจะวนถึงปลายทางที่ต้องการ

เนื่องจากแต่ละเครื่องในระบบเครือข่ายแบบท้องถิ่น ใช้สายสัญญาณชุดเดียวกันในการติดต่อ จึงต้องมีวิธีการที่จะแบ่งเวลาในการใช้สายนี้ให้ทั่วถึง โดยไม่ต้องรอกันนานเกินไป ซึ่งโดยทั่วไปจะมี 2 แบบ คือ

1. CSMA/CD (Carrier Sense Multiple Access / Collision Detection) วิธีนี้ใช้ในกรณีที่ทุกเครื่องต่ออยู่กับสายชุดเดียวกัน คือสำหรับระบบที่เป็นแบบบัสนั่นเอง โดยที่ขณะใดขณะหนึ่งคอมพิวเตอร์แต่ละเครื่องจะคอย"ฟัง" ว่าสายว่างหรือไม่ (carrier sense) ถ้าว่างก็จะเริ่มส่งสัญญาณออกมา ถ้าสายว่างข้อมูลก็จะไปถึงผู้รับได้เลย แต่การเริ่มส่งสัญญาณนี้อาจเกิดขึ้นจากหลาย ๆ สถานีพร้อมกันได้ ผลก็คือสัญญาณที่ได้จะชนกันในสายทำให้ข้อมูลใช้ไม่ได้ ถึงตรงนี้แต่ละเครื่องจะต้องสามารถตรวจจับการชนกันหรือ collision detection ได้ จากนั้นแต่ละเครื่องที่ส่งข้อมูลออกมาชนกันก็จะหยุดส่งและรอ โดยนับถอยหลังในเวลาที่สุ่มขึ้นมาให้แตกต่างกันระหว่างแต่ละเครื่อง เมื่อครบเวลาที่นับของแต่ละเครื่องแล้วก็ค่อยส่งข้อมูลออกมาใหม่ ซึ่งเนื่องจากเวลาที่ใช้รอแตกต่างกันการส่งครั้งใหม่จึงไม่มีโอกาสจะชนกันระหว่างคู่เดิมอีก

2. Token-passing วิธีนี้ใช้กับการต่อสายได้หลายๆ แบบ ไม่ว่าจะเป็นแบบบัส แบบดาว หรือแบบวงแหวน โดยในขณะใดขณะหนึ่งจะมีคอมพิวเตอร์เพียงเครื่องเดียวในระบบเครือข่ายแบบท้องถิ่น ที่มีสิทธิ์ในการส่งข้อมูลโดยมีรหัสที่เรียกว่า token เก็บไว้ เมื่อส่งข้อมูลออกไปเสร็จแล้วก็จะส่งรหัส token นี้ออกไปให้เครื่องอื่น ๆ ตามลำดับที่กำหนดไว้ล่วงหน้า เครื่องอื่น ๆ เมื่อได้รับรหัสแล้ว ถ้าเครื่องไหนยังไม่ต้องการส่งข้อมูลก็จะส่งรหัสต่อไปเลย แต่ถ้าต้องการส่งข้อมูลก็ให้ส่งข้อมูลออกมาก่อน โดยมีรหัส token ส่งมาถึง 1 ครั้งภายใน 1 รอบการทำงานหรือ 1 ห้วงเวลาที่กำหนด ทำให้สามารถจำกัดเวลาได้ว่าจะส่งข้อมูลออกไปภายในเวลาไม่เกินกี่มิลลิวินาที

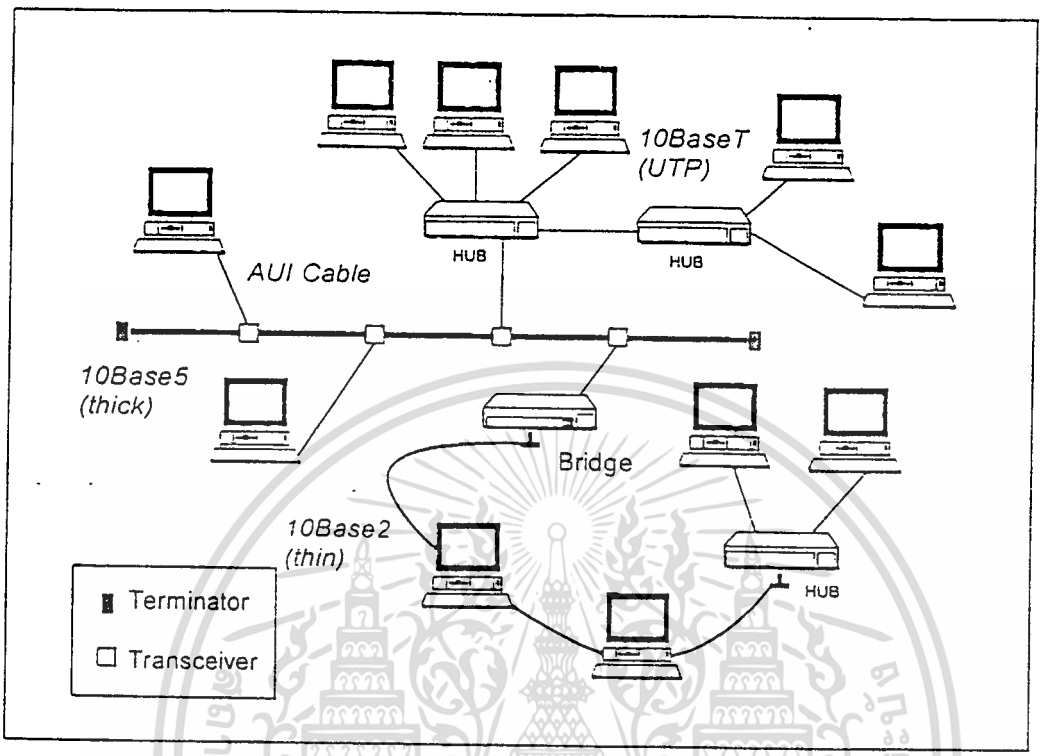
2.4.3 มาตรฐานทางฮาร์ดแวร์ของระบบเครือข่ายแบบท้องถิ่น

ระบบเครือข่ายแบบท้องถิ่นของเครื่องพีซีที่ใช้กันทั่วไปในปัจจุบัน มีลักษณะทางฮาร์ดแวร์แบ่งออกเป็น 3 มาตรฐาน ใหญ่ๆ คือ Ethernet, Arcnet และ Token-Ring แต่ในที่นี้จะอธิบายเพียงรายละเอียดของ Ethernet ดังนี้

1. Ethernet (มาตรฐาน IEEE 802.3) ใช้การต่อสายแบบบัส โดยมีกฎเกณฑ์หรือโปรโตคอลแบบ CSMA/CD (Carrier Sense Multiple Access / Collision Detection) มีความเร็ว 10 ล้านบิตต่อวินาที มีการต่อสายอยู่ 3 แบบ คือ

- Thick (มาตรฐาน *10Base5*) ใช้สายโคแอกเชียลขนาดใหญ่โยงถึงกัน โดยแต่ละจุดจะมีอุปกรณ์รับส่งสัญญาณ (transceiver) เป็นตัวเชื่อม และจากอุปกรณ์รับส่งสัญญาณนี้จะต่อออกไปยังการ์ด LAN โดยจะใช้สายสั้น ๆ ที่เรียกว่า AUI cable (Attachment Unit Interface) อีกทีหนึ่ง
- Thin (มาตรฐาน *10Base2*) ใช้สายโคแอกเชียลขนาดเล็กโยงถึงกัน ใช้วิธีต่อตรงเข้ากับการ์ด LAN โดยไม่ต้องมีอุปกรณ์รับส่งสัญญาณ
- Twisted-pair (มาตรฐาน *10BaseT*) ใช้สายคู่ไขว้ (Unshielded Twisted-Pari หรือ UTP) เหมือนกับสายโทรศัพท์ธรรมดา ต่อจากหลาย ๆ เครื่อง เช่น 8 หรือ 16 เครื่อง เข้าหากล่องรวมสาย (wiring

concentrator หรือ hub) เกิดเป็นสายลักษณะแบบดาวขึ้น จากนั้นจึงต่อเข้ากับสายแบบ thick หรือ thin เพื่อเชื่อมโยงระหว่างแต่ละ hub อีกทีหนึ่ง



รูปที่ 2.1 โครงสร้างของระบบเครือข่ายแบบท้องถิ่น แบบ Ethernet

ระบบเครือข่ายท้องถิ่นแบบ Ethernet เป็นการใช้งานระบบเครือข่ายตามมาตรฐาน IEEE 802.3 ที่มีการส่งข้อมูลแบบเบสแบนด์ความเร็ว 10 ล้านบิตต่อวินาที มีลักษณะการต่อสายแบบบัส และใช้โปรโตคอลแบบ CSMA/CD ข้อมูลที่ถูกส่งในระบบเครือข่าย Ethernet จะเป็นกลุ่มข้อมูลสื่อสารของเฟรม ซึ่งรูปแบบของเฟรมมีตามรูปที่ 2.2

Preamble	Destn. Address	Source Address	Type Field	Data Field	Frame Check Sequence
8 bytes	6 bytes	6 bytes	2 bytes	46-1500 bytes	4 bytes

รูปแบบของเฟรมของ Ethernet

Preamble	Start Delm.	Destn. Address	Source Address	Type Field	Data Field	Pad Byte	Frame Check Sequence
7 bytes	1 byte	2 or 6 bytes	2 or 6 bytes	2 bytes	0-n bytes	0-p bytes	4 bytes

รูปแบบของเฟรมของ IEEE 802.3

รูปที่ 2.2 แสดงรูปแบบกลุ่มข้อมูลสื่อสาร

อุปกรณ์แต่ละตัวในระบบเครือข่าย Ethernet จะมีตำแหน่งของอุปกรณ์ขนาด 48 บิต ซึ่งตำแหน่งของอุปกรณ์นี้สร้างมาจากโรงงานเลย โดย IEEE ในนิวยอร์ค สหรัฐอเมริกาจะควบคุมการกำหนดตำแหน่งเหล่านี้

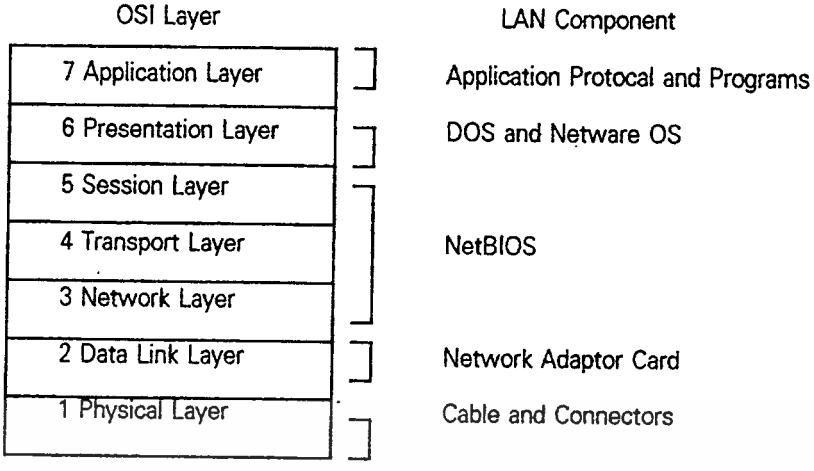
เมื่ออุปกรณ์นี้ได้ส่งกลุ่มข้อมูลสื่อสารไปในระบบเครือข่ายแล้ว จะรออย่างน้อย 9.6 ไมโครวินาทีก่อนจะส่งข้อมูลอีก ซึ่งช่วงเวลานี้เรียกว่าเป็น Inter-Packet-Gab (IPG) ซึ่ง IPG จะช่วยให้อุปกรณ์รับส่งมีเวลาเตรียมพร้อมที่จะรับกลุ่มข้อมูลสื่อสารโดยที่ไม่เกิดปัญหา ถ้าอุปกรณ์พยายามส่งข้อมูลและพบว่าสายไม่ว่างมันจะรอ ถ้าหลังจากที่ส่งกลุ่มข้อมูลสื่อสารไปแล้วเกิดการชนกันขึ้น อุปกรณ์รับส่งจะรอเป็นช่วงเวลาแบบสุ่มก่อนที่เริ่มการส่งกลุ่มข้อมูลสื่อสารที่ชนกันนั้นใหม่ อุปกรณ์รับส่งจะพยายามส่งสูงสุด 16 ครั้ง ถ้ายังส่งไม่ได้จะรายงานว่าเกิดข้อผิดพลาดขึ้น

ความต้องการพื้นฐานของระบบเครือข่าย Ethernet ก็คือเมื่อเกิดการชนกันของข้อมูลขึ้น ทุก ๆ สถานีจะต้องรู้ถึงการชนนั้น ก่อนที่สถานีที่เกี่ยวข้องกับการชนนั้นจะส่งเฟรมขนาดต่ำสุดคือ 572 บิตออกมา ถ้ากฎเกณฑ์เหล่านี้ไม่ดีพอแล้ว อาจทำให้บางสถานีรับข้อมูลได้แต่บางสถานีไม่สามารถรับได้ ข้อมูลจากวงจรโลจิกควบคุม (LLC:Logical Link Control) จะมีความยาว 368 บิต (46 ไบต์) และ 12,000 บิต (1,500 ไบต์) ถ้าข้อมูลมีขนาดสั้นเกินไปจะถูกเติมให้ครบ 368 บิตด้วย 0 ซึ่งจะหมายความว่าถ้าทุกกลุ่มข้อมูลสื่อสารใช้ขนาดต่ำสุดในการส่งแล้วจะสามารถส่งได้ด้วยความเร็ว 7,619,047 บิตต่อวินาที ถ้ากลุ่มข้อมูลสื่อสารใช้ขนาดสูงสุดในการส่งแล้วจะสามารถส่งได้ด้วยความเร็ว 9,523,212 บิตต่อวินาที

2.5 เปรียบเทียบโมเดล OSI กับระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น

โมเดล OSI (Open System Interconnection) เป็นโมเดลที่พัฒนาขึ้นโดย International Standards Organization (ISO) เพื่อเป็นมาตรฐานในการออกแบบ Protocol ในการติดต่อสื่อสารระหว่างคอมพิวเตอร์และเป็นโครงสร้างที่ใช้กันอย่างแพร่หลายสำหรับเปรียบเทียบระบบเครือข่ายทางทฤษฎี

โมเดล OSI แบ่งออกเป็น 7 ชั้น ซึ่งแต่ละชั้นจะประกอบกันเป็นพื้นฐานเพื่อเป็นแนวความคิดในการออกแบบ และการนำระบบเครือข่ายไปใช้งาน โมเดล OSI จะบรรยายกระบวนการติดต่อสื่อสารตามลำดับขั้นของชั้น แต่ละชั้นจะกำหนดการติดต่อกับชั้นที่ติดกัน ซึ่งการติดต่อเหล่านี้มีความยืดหยุ่นพอที่จะให้ผู้ออกแบบระบบใช้ Protocol ได้หลายแบบโดยที่ระบบยังอยู่ในมาตรฐาน



รูปที่ 2.3 แสดงการเปรียบเทียบระหว่างโมเดล OSI กับ
องค์ประกอบในระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น

โดยความสัมพันธ์ขององค์ประกอบในระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น กับโมเดล OSI มีดังนี้

ชั้นที่1 : รวมถึงสื่อที่ใช้ในการส่งข้อมูล ตัวเชื่อมต่อ ตัวขยายสัญญาณ และอุปกรณ์อื่นๆในการรับส่งบิตข้อมูล

ชั้นที่2 : นำไปใช้งานบนการ์ดเชื่อมโยงระบบเครือข่าย กำหนดข้อมูลจากชั้นที่สูงขึ้นไปให้อยู่ในลำดับที่กำหนด คือ Ethernet,Token-Ringหรือ Arcnet และรูปแบบเฟรมอื่นๆจะแตกต่างกันออกไป นอกจากนั้นยังมีการกำหนดตำแหน่งของต้นทางและปลายทางเพื่อระบุผู้ส่งและผู้รับของเฟรม รวมทั้งตรวจสอบความผิดพลาดโดยวิธี CRC (Cyclic Redundancy Check)

ชั้นที่3-5 : นำไปใช้งานใน NetBIOS (Network Basic Input/Output System) NetBIOS จะมีฟังก์ชันสำหรับ Session Layer เพื่อจัดการเชื่อมต่อทางตรรกะระหว่างสถานีผู้ใช้ในระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น ส่วนฟังก์ชันใน Transport Layer และ Network Layer ไม่ต้องใช้ในระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น

NetBIOSอาจจะอยู่ในROMบนการ์ดเชื่อมโยงระบบเครือข่ายหรือจำลองอยู่ที่ซอฟท์แวร์ที่สถานีผู้ใช้และศูนย์บริการข้อมูล

ชั้นที่6 : นำไปใช้งานในระบบปฏิบัติการของระบบเครือข่าย(NOS:Network Operating System) และการจำลองในDOSที่สัมพันธ์กัน

ชั้นที่ 7 : โปรแกรม Protocal และโปรแกรมประยุกต์

บทที่ 3
ระบบจัดการระบบเครือข่ายคอมพิวเตอร์ (Netware)

3.1 ลักษณะโดยทั่วไปของระบบจัดการเครือข่ายแบบ Netware

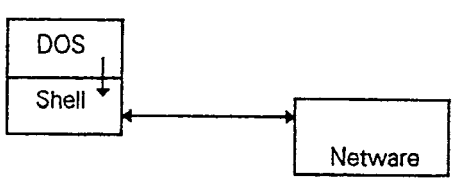
เน็ตแวร์เป็นระบบจัดการระบบเครือข่ายคอมพิวเตอร์บนไมโครคอมพิวเตอร์แบบพีซีที่นิยมใช้กันมากที่สุดในปัจจุบัน ประมาณกันว่ากว่า 90% ของการติดตั้งระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่นในประเทศไทยนั้นใช้เน็ตแวร์เป็นระบบจัดการ ความนิยมนี้เนื่องมาจากเน็ตแวร์ทำงานได้เป็นอย่างดี สามารถทำงานกับฮาร์ดแวร์ได้หลากหลายและยังมีการใช้ระบบรักษาความปลอดภัยที่เหมาะสม นอกจากนี้เน็ตแวร์ยังมีการเตรียมอุปกรณ์และซอฟต์แวร์สำหรับการใช้งานเครือข่ายไว้ให้มากเพียงพอ เ น็ตแวร์สามารถมีเครื่องสถานีผู้ใช้ได้หลายแบบ เช่น ไมโครคอมพิวเตอร์ หรือ แมคอินทอช และสามารถเชื่อมโยงระบบเครือข่ายอื่น ๆ ได้หลายรูปแบบ อีกทั้งยังสามารถติดตั้งได้ในลักษณะการต่อสายหลายๆแบบด้วย

ระบบจัดการระบบเครือข่ายเน็ตแวร์ จะถูกติดตั้งในเครื่องคอมพิวเตอร์เครื่องใดเครื่องหนึ่งในระบบเครือข่าย หลังจากที่ติดตั้งเรียบร้อยแล้ว คอมพิวเตอร์เครื่องนั้นจะถูกเรียกว่า "ศูนย์บริการเครือข่าย" โดยทำหน้าที่บริการแก่เครื่องคอมพิวเตอร์ลูกข่ายที่เรียกว่า "สถานีผู้ใช้" ซึ่งทำงานภายใต้ระบบปฏิบัติการต่างๆ เช่น DOS,OS/2 และจะเข้าไปใช้อุปกรณ์ต่างๆของศูนย์บริการได้ต้องโหลดโปรแกรมเชลล์ (Shell) ซึ่งเป็นโปรแกรมฝังตัวขึ้นมาก่อน

3.2 การทำงานของเน็ตแวร์

ระบบปฏิบัติการเน็ตแวร์ จะสนับสนุนการใช้งานเครื่องไมโครคอมพิวเตอร์ที่ต่างกัน หรือมีระบบปฏิบัติการที่ต่างกัน โดยไม่ต้องมีการแบ่งเนื้อที่ฮาร์ดดิสค์ ไฟล์ทั้งหมดและสถานีผู้ใช้ทุกสถานีสามารถใช้งานไดเรคทอรีร่วมกันได้

เน็ตแวร์จะอยู่ที่ศูนย์บริการข้อมูล และดอสจะอยู่ที่สถานีของผู้ใช้ เน็ตแวร์เชลล์จะอยู่ที่สถานีของผู้ใช้ เพื่อให้ดอสติดต่อกับเน็ตแวร์ได้

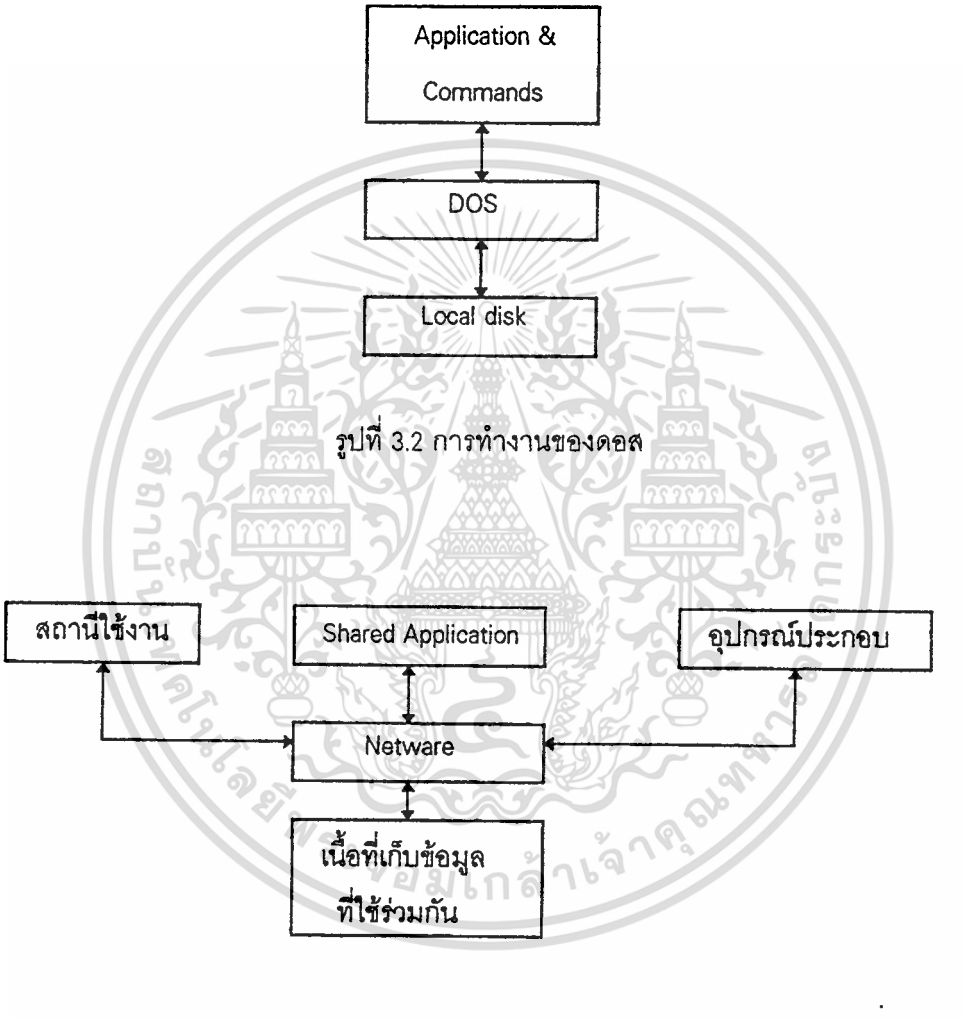


สถานีของผู้ใช้

ศูนย์บริการข้อมูล

รูปที่ 3.1 เชลล์ของเน็ตแวร์ทำหน้าที่ติดต่อระหว่างดอสของสถานีผู้ใช้งานกับศูนย์บริการ

ดอส, เน็ตแวร์ และเชลล์ จะทำงานร่วมกันเพื่อช่วยในการใช้งานข้อมูล และฮาร์ดแวร์ร่วมกัน โดยดอสจะถูกออกแบบให้ประมวลผลโปรแกรมที่สถานีผู้ใช้ในสภาพแวดล้อมแบบผู้ใช้คนเดียว ดังแสดงในรูปที่ 3.2 ในขณะที่เน็ตแวร์ถูกออกแบบมาให้ประมวลผลในสภาพแวดล้อมที่มีผู้ใช้หลายคน ดังแสดงในรูปที่ 3.3 ซึ่งจะช่วยในการใช้งานอุปกรณ์ต่างๆร่วมกัน ควบคุมไดเรคทอรีที่ซับซ้อน และ File Allocation Table (FAT) ที่ศูนย์บริการข้อมูล



รูปที่ 3.2 การทำงานของดอส

รูปที่ 3.3 การทำงานของเน็ตแวร์

3.3 การติดต่อสื่อสารของเน็ตแวร์

ซอฟต์แวร์สำหรับระบบเครือข่ายในเครื่องสถานีผู้ใช้แต่ละเครื่อง จะมีการแบ่งการทำงานเป็นชั้น ชั้นที่อยู่ต่ำที่สุดจะทำหน้าที่ติดต่อกับอุปกรณ์การเชื่อมต่อระบบเครือข่าย ชั้นระดับบนที่สุดจะทำหน้าที่ติดต่อกับโปรแกรมของผู้ใช้งานระบบ และจัดการช่วยเหลือด้านการติดต่อโปรแกรมของผู้ใช้งานระบบกับ

กระบวนการการใช้งานระบบเครือข่าย ในซอฟต์แวร์แต่ละชั้นจะพัฒนาขึ้นโดยมีการกำหนดหน้าที่ในแต่ละชั้นไว้ก่อนล่วงหน้าแล้ว

ในการสื่อสารชั้นล่างสุดนั้น เครื่องคอมพิวเตอร์ในระบบเครือข่ายนั้นจะทำการติดต่อกับเครื่องอื่น ๆ ในระบบ และกับศูนย์บริการข้อมูล โดยการใช้แพ็กเก็ตข่าวสาร หรือมักเรียกกันว่า เฟรม ขั้นตอนการทำงานทั้งหมดของระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่นนั้น จะมีการรับส่งโดยใช้อุปกรณ์การเชื่อมต่อระบบเครือข่าย และการเรียกการใช้งานผ่านซอฟต์แวร์สำหรับอุปกรณ์นั้น

ซอฟต์แวร์สำหรับระบบเครือข่ายนั้น ทำการส่งเฟรมเพื่อการทำงานในหลาย ๆ หน้าที่ดังนี้

- การเริ่มต้นเปิดช่องทางการสื่อสาร
- การส่งข้อมูล เช่น เรคคอร์ดจากแฟ้มข้อมูล ไปยังเครื่องคอมพิวเตอร์
- การตอบรับการรับรู้มาถึงของข้อมูล
- การกระจายข่าวสารไปยังเครื่องอื่น ๆ
- การปิดช่องทางการสื่อสาร

การพัฒนาระบบเครือข่ายขึ้นมานั้น มีการกำหนดรูปแบบของเฟรมหลายรูปแบบแตกต่างกันไป แต่รูปแบบที่แสดงไว้นั้นเป็นข้อมูลที่มีการใช้เป็นประจำ ประกอบไปด้วย

- ตำแหน่งที่อยู่ระบบเครือข่ายของผู้ที่ส่งเฟรมนั้น
- ตำแหน่งที่อยู่ระบบเครือข่ายของผู้รับเฟรมนั้น
- ข้อมูลแสดงประเภทของข้อมูลภายในเฟรม
- ข้อมูลหรือข่าวสาร
- ข้อมูลตรวจสอบความถูกต้อง เช่น การตรวจสอบผลรวมหรือการตรวจสอบแบบ CRC

Sender ID	Desti ID	Frame Type	Data/Message	CRC

รูปที่ 3.4 แสดงรูปแบบพื้นฐานของเฟรมที่ใช้ในระบบเน็ตแวร์

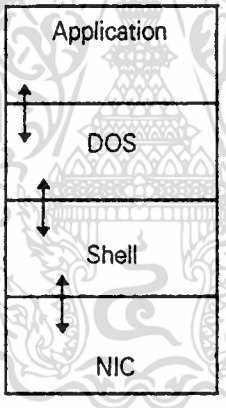
การรับส่งเฟรมจะกระทำโดยซอฟท์แวร์ระบบเครือข่าย ขั้นตอนการทำงานของเครื่องคอมพิวเตอร์จะเป็นตัวเรียกการทำงานนั้นเอง เมื่อมีการใช้แฟ้มข้อมูลที่อยู่ภายในศูนย์บริการข้อมูล หรือโดยการทำงานตามโปรโตคอล เช่น จาก NetBIOS หรือ IPX ที่จะส่งข่าวสารถึงเครื่องคอมพิวเตอร์เครื่องอื่นภายในระบบเครือข่าย

3.4 การเขียนโปรแกรมบนระบบเครือข่าย

3.4.1 เน็ตแวร์เชลล์

เน็ตแวร์เชลล์ เป็นโปรแกรมที่จะต้องถูกโหลดขึ้นมาในหน่วยความจำก่อน จึงจะสามารถเริ่มใช้งานในระบบเครือข่ายได้ ในเน็ตแวร์ 2.1 ขึ้นไป เชลล์จะอยู่ในรูปของ 2 โปรแกรมประกอบกันอยู่คือ IPX และ Net3 โดย IPX ใช้ติดต่อระหว่างศูนย์บริการข้อมูล และสถานีผู้ใช้ อื่น ๆ ส่วน Net3 เป็นส่วนอินเตอร์เฟสระหว่าง ส่วนของระบบเครือข่ายกับดอส

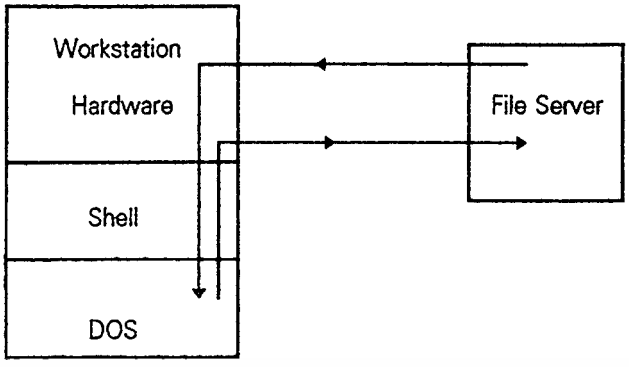
ประโยชน์ของเชลล์ก็คือ ทำหน้าที่เชื่อมการติดต่อระหว่างดอสกับระบบเครือข่ายเมื่อเชลล์ถูกโหลดขึ้นมา ซอฟต์แวร์สำหรับการเชื่อมต่อก็จะถูกสร้างขึ้น และศูนย์บริการข้อมูลที่ใกล้ที่สุดก็จะถูกนำมาให้บริการแก่สถานีผู้ใช้นั้น



รูปที่ 3.5 แสดงความสัมพันธ์ระหว่างเน็ตแวร์เชลล์กับดอส

Mapping เป็นวิธีการอ้างถึง (access) ไดเรกทอรีย่อยของศูนย์บริการข้อมูลจากไดรฟ์ปกติ โดยจะต้องกล่าวถึง Path ของไดเรคทอรีย่อยที่ต้องการอ้างถึงให้ครบ

ดังนั้น เชลล์จึงเป็นเหมือนตัวติดต่อระหว่าง สถานีผู้ใช้กับศูนย์บริการข้อมูล เช่นถ้าผู้ใช้ต้องการเปิดไฟล์ของระบบเครือข่ายใช้งาน เมื่อเชลล์รู้ว่าผู้ใช้ต้องการเปิดไฟล์ก็จะทำการส่งแพ็กเก็ตไปยังศูนย์บริการข้อมูลเพื่อขอเปิดไฟล์ที่ต้องการ หลังจากนั้นศูนย์บริการข้อมูลก็จะส่งแพ็กเก็ตกลับมายังเชลล์ เพื่อแจ้งว่าการขอเปิดไฟล์นั้นสำเร็จหรือไม่



รูปที่ 3.6 แสดงการร้องขอและการตอบสนองระหว่างสถานีผู้ใช้กับระบบเครือข่าย

3.4.2 ลักษณะการจัดเก็บข้อมูลในหน่วยความจำของเน็ตแวร์

เน็ตแวร์มีรูปแบบการจัดเก็บข้อมูลในหน่วยความจำแตกต่างไปจากการจัดเก็บข้อมูลในสถาปัตยกรรมของไมโครโปรเซสเซอร์ตระกูลของอินเทล 80x86 ซึ่งใช้รูปแบบของการจัดเก็บข้อมูลในหน่วยความจำเป็นแบบ High-low Format คือจะเก็บจากไบต์นัยสำคัญต่ำ (LSB) ไว้ก่อนไบต์นัยสำคัญสูง (MSB) เช่น ข้อมูลค่า 0x1234 จะถูกจัดเก็บเป็น 0x3412 หรือค่า 0x12345678 จะถูกจัดเก็บเป็น 0x87564312 ส่วนตัวเน็ตแวร์จะเก็บค่าข้อมูลเหล่านี้ไว้ในทางกลับกัน การเขียนโปรแกรมจึงต้องมีการสลับค่าไบต์สูง-ต่ำอยู่บ่อย ๆ

3.4.3 ภาษาที่ใช้ในการเขียนโปรแกรม

โปรแกรมประยุกต์สำหรับเน็ตแวร์ สามารถพัฒนาขึ้นมาได้จากภาษาใดก็ได้ ที่สามารถเชื่อมต่อกับรูทีนภายนอก สามารถอ้างถึงรีจิสเตอร์ภายใน และสามารถเรียกใช้ฟังก์ชันอินเทอร์รัพต์หมายเลข 21h ได้

สำหรับโครงงานฉบับนี้ เลือกใช้ภาษา C เนื่องจากเป็นภาษาที่มีความยืดหยุ่น มีความสามารถสูง และเป็นที่นิยมอย่างกว้างขวาง นอกจากนี้โครงสร้างของภาษายังเอื้ออำนวยต่อการใช้งานแพ็กเก็ตร้องขอและตอบรับ ซึ่งใช้ติดต่อสื่อสารกับศูนย์บริการข้อมูลอีกด้วย

3.4.4 แพ็กเก็ตร้องขอและแพ็กเก็ตตอบรับ (Request and Reply Packet)

การเรียกใช้ฟังก์ชันการทำงานของเน็ตแวร์ โดยทั่วไปจะใช้รีจิสเตอร์เพียงไม่กี่ตัว แต่ทุกฟังก์ชันจะต้องใช้แพ็กเก็ตร้องขอและแพ็กเก็ตตอบรับในการร้องขอและรับทราบข้อมูลจากระบบ แพ็กเก็ตในที่นี้เทียบเท่ากับข้อมูลโครงสร้างในภาษาซี ซึ่งก็คือกลุ่มของฟิลด์ข้อมูลที่ต้องส่งไปร้องขอการทำงานของระบบเครือข่ายที่ศูนย์บริการข้อมูล ซึ่งจะต้องมีการเซ็ตค่าข้อมูลให้สมบูรณ์ก่อนที่จะทำการส่งไปยังศูนย์บริการข้อมูลเพื่อทำกระบวนการร้องขอ และต้องกำหนดขนาดของแพ็กเก็ตร้องขอไว้ที่ตอนต้นของแพ็กเก็ตด้วยเสมอ โดยเวิร์ดแรกของแพ็กเก็ตต้องเก็บความยาวของแพ็กเก็ตเป็นไบต์ไว้ หากไม่ได้กำหนดไว้ก่อน การร้องขอจะผิดพลาด ส่วนแพ็กเก็ตตอบรับจะเป็นข้อมูลที่ระบบเครือข่ายส่งกลับมาให้ ซึ่งต้องมีการกำหนดความยาวของแพ็กเก็ตไว้เช่นเดียวกัน

โดยทั่วไปแล้วในการเรียกใช้ฟังก์ชันของระบบจะใช้คู่รีจิสเตอร์ DS:SI ในการอ้างแอดเดรสของแพ็กเก็ตร้องขอ และใช้คู่รีจิสเตอร์ ES:DI ในการอ้างแอดเดรสของแพ็กเก็ตตอบรับ แต่บางฟังก์ชันก็อาจจะใช้แพ็กเก็ตแบบเดียวก็ได้

3.4.5 การสื่อสารภายในเครือข่ายโดยใช้บริการของเน็ตแวร์ แบบ IPX

ปัจจุบันซอฟต์แวร์ที่ใช้เป็นตัวจัดการระบบเครือข่ายแบบท้องถิ่น จะเป็นของบริษัท Novel NetWare แต่เนื่องจากโปรแกรมที่ใช้งานโดยทั่วไปจะถูกใช้งานไปในลักษณะการใช้ฐานข้อมูลร่วมกัน (share data) ซึ่งถูกเขียนขึ้นจากโปรแกรมสำเร็จรูป เช่น FOXPRO/LAN หรือ CLIPPER แต่การเขียนโปรแกรมในลักษณะการรับ หรือส่งข้อมูลระหว่างสถานีผู้ใช้ในระบบเครือข่ายแบบท้องถิ่นยังเป็นสิ่งที่ยาก และไม่แพร่หลายนัก เนื่องจากขาดแคลนหนังสือคู่มือประกอบ

ในส่วนนี้จะเป็นการแนะนำวิธีการเขียนโปรแกรมรับหรือส่งข้อมูลระหว่าง 2 สถานีผู้ใช้ใดๆ ในระบบเครือข่ายท้องถิ่นวงเดียวกันของบริษัท Novell NetWare version 3.1 ขึ้นไป การให้บริการของเน็ตแวร์ในการสื่อสารมีหลายวิธี ดังนี้

- โดยใช้ IPX
- โดยใช้ SPX
- โดยใช้ NetBIOS

ในที่นี้จะอธิบายวิธีการสื่อสารโดยใช้ IPX เนื่องจากเป็นวิธีที่มีประสิทธิภาพดีที่สุดในทั้ง 3 วิธี

โปรโตคอล IPX

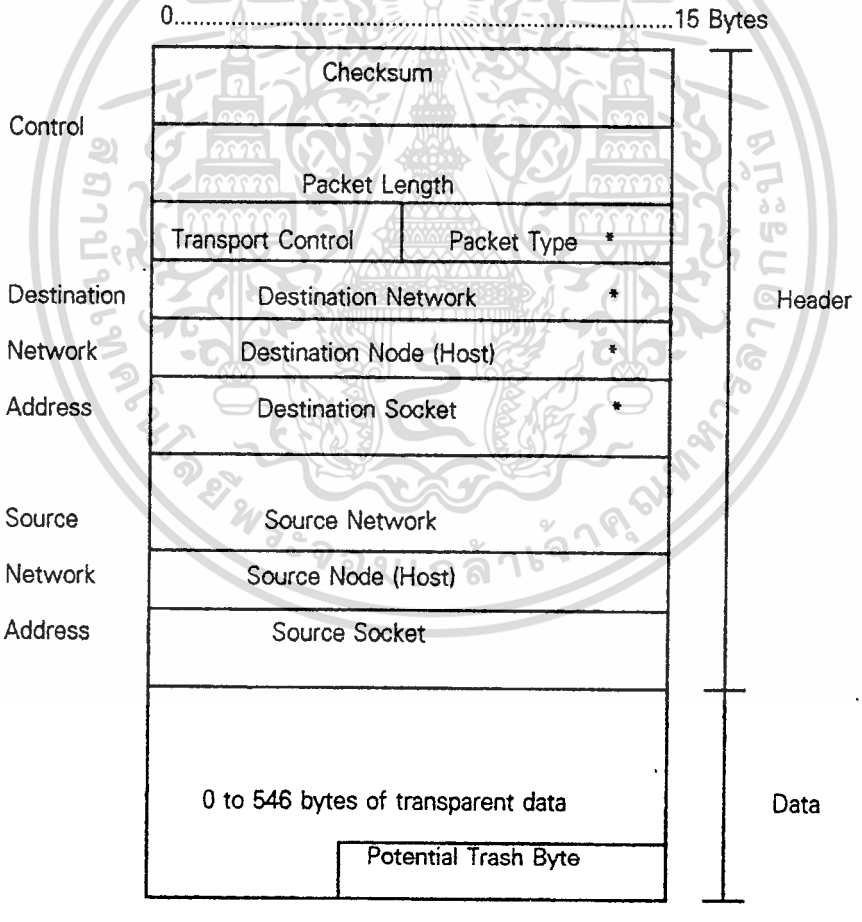
โปรโตคอล IPX (Internetwork Packet Exchange) ซึ่ง Novell ได้ปรับปรุงมาจากโปรโตคอล XNS (Xerox Network System) ของบริษัท Xerox โดยพัฒนาขึ้นมาเพื่อใช้เป็นมาตรฐานของเน็ตแวร์โดยเฉพาะ ดังนั้นในการที่ระบบเน็ตเวิร์กอื่นจะมาติดต่อกับเน็ตแวร์ได้ ก็โดยแปลงโปรโตคอลของตนเข้ามาในแบบของ IPX หรือแปลงโปรโตคอล IPX เป็นโปรโตคอลอื่น

IPX เป็นโปรโตคอลที่จัดอยู่ใน NetWork Layer ตามมาตรฐานของ OSI ซึ่งจะให้บริการแบบดาต้าแกรม (Datagram) หมายถึงข้อมูลที่จะถูกส่งจากต้นทางผ่านเครือข่ายไปยังปลายทางได้อย่างถูกต้อง แต่จะไม่มีการยืนยันได้ว่าปลายทางได้รับข้อมูลหรือไม่

ทุกครั้งที่มีการรับหรือส่งข้อมูล จะมีการควบคุมโดย ECB (Event Control Block) เช่น ก่อนที่โปรแกรมจะสามารถส่งข้อมูลในรูปของแพ็กเก็ตไปได้ โปรแกรมจะต้องมีการเตรียม ECB ก่อน จากนั้นโปรแกรมจึงนำแอดเดรสของ ECB ไปไว้ในคู่ของเซ็กเมนต์รีจิสเตอร์ที่เหมาะสม แล้วจึงเรียกใช้ IPX เพื่อส่งข้อมูล

ESR (Event Service Routine) เป็นฟังก์ชันที่ถูกกำหนดขึ้นในโปรแกรม โดยฟังก์ชันนี้ จะถูกเรียกใช้โดย ECB หลังจากที่มีการรับส่งข้อมูลเกิดขึ้น

โครงสร้างของ IPX จะประกอบด้วยส่วนของเฮดเดอร์ 30 ไบต์ และตามด้วย 0-546 ไบต์ข้อมูล (ในการส่งข้อมูลแบบ IPX จะส่งแต่ละครั้งได้มากที่สุด 546 ไบต์) โดยโครงสร้างของ IPX แสดงได้ดังรูปที่ 3.7



รูปที่ 3.7 แสดงเฟรมของ IPX

โปรแกรมจะต้องให้ค่าตัวแปรในช่องที่มีเครื่องหมาย * เมื่อจะส่งข้อมูล ส่วนตัวแปรในช่องที่เหลือจะถูกจัดการโดยตัว IPX เอง

รายละเอียดของเฮดเดอร์

- Checksum จะเก็บค่าผลลัพธ์ของเฮดเดอร์แพ็กเก็ตทั้ง 30 ไบต์
- Packet Length จะเก็บค่าความยาวของข้อมูลทั้งหมด (มีค่าตั้งแต่ 0-576 ไบต์)
- Transport Control จะมีค่าเป็น 0 ก่อนที่จะส่งข้อมูล
- Packet Type จะบอกชนิดของการให้บริการ ซึ่งชนิดของการให้บริการมีดังนี้

Type	ความหมาย
0	Unknow Packet Type
1	Routing Information Packet
2	Echo Packet
3	Error Packet
4	Packet Exchange Packet
5	Sequenced Packet Protocol Packet

เมื่อใช้บริการของ IPX ค่า Packet Type ควรมีค่าเป็น 4

- Destination Network และ Source Network จะเก็บค่าแอดเดรสของเครือข่ายปลายทางและต้นทาง ตามลำดับ
- Destination Node และ Source Node จะเก็บค่า LAN Board ของปลายทางและต้นทาง ตามลำดับ
- Destination Socket และ Source Socket จะเก็บค่าแอดเดรสของซอกเก็ตปลายทางและต้นทาง ตามลำดับ

โครงสร้างของ ECB มีรูปแบบดังนี้

Offset	Content	Type
0	Link Address	byte(4)
4	ESR Address	byte(4) r,s
8	In Use Flag	byte
9	Completion Code	byte r,s
10	Socket Number	word
12	IPX Workspace	byte(4)
16	Driver Workspace	byte(12) s
28	Immediate Address	byte(6) r,s

Offset	Content	Type
34	Flagment Count	word r,s
36	Flagment Address1	byte(4) r,s
40	Flagment size1	word r,s
42	Flagment Address2	byte(4) r,s
46	Flagment size2	word r,s

โปรแกรมจะต้องให้ค่าตัวแปรในช่องที่มีเครื่องหมาย s เมื่อจะส่งข้อมูล และในช่องที่มีเครื่องหมาย r เมื่อจะรับข้อมูล

- ESR จะเก็บค่าแอดเดรสของฟังก์ชัน ESR ที่ถูกกำหนดไว้ลงในโปรแกรม
- In Use Flag จะเก็บค่าสถานะปัจจุบัน
- Completion Code จะเก็บค่ารหัสซึ่งใช้บอกให้ทราบว่าทำการรับข้อมูลเรียบร้อยแล้ว
- Immediate Address จะเก็บค่าแอดเดรสของโหนด โดยค่านี้จะได้มาจากฟังก์ชัน GetLocalTarget
- Flagment Count จะเก็บจำนวนของบัฟเฟอร์ในแพ็กเก็ต ค่าของ Flagment Count จะขึ้นอยู่กับจำนวน Flagment Address และ Flagment size
- Flagment Address จะเก็บค่าแอดเดรสของแต่ละบัฟเฟอร์
- Flagment size จะเก็บขนาดของแต่ละบัฟเฟอร์

3.5 ข้อกำหนดการเขียนโปรแกรมติดต่อกับระบบจัดการของเน็ตแวร์

เน็ตแวร์ APIs (Application Programming Interfaces) หรือรู้จักกันในชื่อ Netware Core Protocol (NCP) เป็นรายละเอียดการเขียนโปรแกรมติดต่อกับตัวระบบจัดการเน็ตแวร์ ซึ่งมีฟังก์ชันให้เรียกใช้มากกว่า 200 ฟังก์ชัน และทั้งหมดมีการเรียกใช้อินเทอร์รัพต์ 21h ซึ่งเป็นอินเทอร์รัพต์ของดอส โดย API นี้จะจัดการในเรื่องไฟล์, การพิมพ์, การติดต่อสื่อสาร, การจัดการบนระบบเครือข่าย ที่เรียกใช้ได้โดยสถานีผู้ใช้ใดๆที่ต่ออยู่กับระบบเครือข่าย

โปรแกรมที่เขียนกับ API ช่วยในการบริการบางอย่างของระบบเครือข่าย แต่จะประมวลผลได้เฉพาะบนระบบเครือข่ายที่ใช้เน็ตแวร์เท่านั้น ส่วนโปรแกรมในระบบเครือข่ายที่เขียนโดยใช้ดอส จะประมวลผลได้บนระบบเครือข่ายท้องถิ่นของเครื่องไมโครคอมพิวเตอร์ได้แทบทุกระบบ แต่ฟังก์ชันในการใช้และควบคุมระบบเครือข่ายมีจำกัด และการทำงานในระบบเครือข่าย บางครั้งต้องการการทำงานมากกว่าที่ฟังก์ชันของดอสมีให้

สภาพแวดล้อมของระบบเครือข่ายส่วนใหญ่จะสนับสนุน API ตามลำดับขั้นของชั้นของระบบเครือข่าย ซึ่งชั้นดังกล่าวจะสัมพันธ์กับชั้นทั้ง 7 ของโมเดล OSI ในระดับชั้นล่าง จะเป็นการบริการที่ค่อนข้างธรรม

ธรรมดา ในขณะที่ระดับสูงจะให้การบริการที่มีประสิทธิภาพมาก ซึ่งจะอยู่ล้อมรอบโปรแกรมประยุกต์จากรูทีนในระดับต่ำ

API แต่ละตัวที่ติดตั้งจะใช้หน่วยความจำล่างจำนวนหนึ่งเพื่อเป็นที่อยู่ของมัน ทำให้หน่วยความจำต้องสูญเสียไปจากหน่วยความจำที่มีอยู่ เช่น เชลล์ของเน็ตแวร์ที่สถานีผู้ใช้ จะใช้หน่วยความจำประมาณ 60 กิโลไบต์

API ประกอบไปด้วยส่วนบริการต่าง ๆ 7 ส่วน คือ

1. ส่วนบริการฐานข้อมูลเครือข่าย (Bindery Services)
2. ส่วนบริการด้านศูนย์บริการข้อมูล (File Server Environment Services)
3. ส่วนบริการด้านการเชื่อมต่อ (Connection Services)
4. ส่วนบริการด้านสถานีผู้ใช้ (Workstation Services)
5. ส่วนบริการด้านข้อความ (Message Services)
6. ส่วนบริการด้านไดเรคทอรี (Directory Services)
7. ส่วนบริการด้านการพิมพ์ (Print Services)

ในที่นี้จะกล่าวถึงส่วนบริการที่สำคัญที่ใช้ในโครงงานนี้ คือ

3.5.1 ส่วนบริการฐานข้อมูลเครือข่าย

Bindery เป็นโปรแกรมแบบฝังตัวที่อยู่ในแต่ละศูนย์บริการข้อมูล โดยทำหน้าที่เป็นฐานข้อมูลของระบบเครือข่าย ซึ่งประกอบไปด้วยรายการของออบเจคท์ (Objects) การเรียกใช้บริการของ Bindery เพื่อค้นหารายละเอียดของออบเจคท์ที่กำหนดบนเครือข่าย

ออบเจคท์ประกอบไปด้วยชื่อของออบเจคท์ และแบบของออบเจคท์อันได้แก่ ผู้ใช้ กลุ่มของผู้ใช้ ไดเรคทอรี แถวรองานพิมพ์ ศูนย์บริการที่ต่ออยู่และรายชื่ออื่นๆบนระบบเครือข่าย แต่ละออบเจคท์จะมีคุณสมบัติประจำตัวที่เรียกว่า พรอพเพอร์ตี้ (property) เช่นข้อมูลที่เกี่ยวข้องกับออบเจคท์แบบผู้ใช้ (user) จะได้แก่ กลุ่มที่ผู้ใช้ใช้อยู่ ข้อมูลของการล็อกอินและรหัสผ่าน ซึ่งก็จะมีพรอพเพอร์ตี้เป็น GROUP_I'M_IN จะเก็บรายชื่อของกลุ่มที่ผู้ใช้อยู่ และ PASSWORD จะเก็บรหัสผ่านของผู้ใช้ที่ผ่านการเข้ารหัสมาแล้ว

ระดับการเข้าถึง	ผู้ที่ได้รับอนุญาต	ความหมาย
0	ทุกคน	ทุกออบเจคท์สามารถเข้าถึงได้ไม่ว่าจะมีการล็อกอินเข้าสู่ศูนย์บริการข้อมูลหรือไม่ก็ตาม
1	Logged	อนุญาตให้เฉพาะลูกข่าย(client)ที่ล็อกอินเข้าสู่ศูนย์บริการข้อมูลเท่านั้น
2	Object	อนุญาตให้เฉพาะลูกข่ายที่ล็อกอินเข้าศูนย์บริการข้อมูลด้วยชื่อของออบเจคท์, ชนิด และรหัสผ่านที่กำหนดไว้เท่านั้น
3	Supervisor	อนุญาตให้เฉพาะ supervisor หรือออบเจคท์ที่เทียบเท่า เท่านั้น(ออบเจคท์จะมีฐานะเทียบเท่ากับ supervisor ได้ ก็ต่อเมื่อมีคุณสมบัติ SECURITY_EQUALS บรรจุอยู่ในหมายเลขของออบเจคท์ของ supervisor)
4	Netware	เพียงเน็ตแวร์เท่านั้นที่สามารถเข้าถึงออบเจคท์หรือพรอบเพอร์ตี้ในความปลอดภัยระดับนี้

รูปที่ 3.8 แสดงระดับความปลอดภัยในการเข้าถึงบินเดอรี

3.5.1.1 Bindery Objects

ออบเจคท์เป็นองค์ประกอบพื้นฐานของบินเดอรี่ ซึ่งถูกกำหนดขึ้นโดยชื่อและแบบของออบเจคท์ เช่นแบ่งเป็น ผู้ใช้ , กลุ่มผู้ใช้ , ศูนย์บริการข้อมูล หรืออะไรก็ตามที่ผู้พัฒนาต้องการ ดังนั้นเราสามารถใช้ 2 ออบเจคท์ที่ชื่อ SERVER-1 เหมือนกันโดยออบเจคท์หนึ่งสามารถใช้เป็นศูนย์บริการข้อมูล อีกออบเจคท์หนึ่งใช้เป็น ศูนย์บริการการพิมพ์ก็ได้

- Object Security

ความปลอดภัยของบินเดอรี่มีผลต่อความปลอดภัยของระบบเครือข่าย ดังนั้นจึงมีการจัดระดับชั้นความปลอดภัยขึ้นในบินเดอรีให้สำหรับแต่ละออบเจคท์ โดยกำหนดอยู่ในรูปสิทธิต่างๆในการอ้างถึงบินเดอรี่ ดังแสดงในรูปที่ 3.8

การตั้งค่าเกี่ยวกับความปลอดภัยสำหรับแต่ละออบเจคท์และพรอพเพอร์ตี้จะมี 2 มีลักษณะคือ อนุญาตหรือปฏิเสธในสิทธิการ อ่านข้อมูล กับควบคุมการบันทึกค่าสิทธิ (เพิ่มหรือลดค่าของพรอพเพอร์ตี้ต่างๆ) การตั้งค่าความปลอดภัยจะใช้เพียงไบต์เดียว (ใช้ 4 บิตด้านสูงเก็บค่าสิทธิ, ส่วนด้านต่ำใช้กำหนดสิทธิการอ่าน) โดยค่าปกติของเน็ตแวร์คือ ชั้น Logged สำหรับการอ่านข้อมูลและชั้น Supervisor สำหรับการบันทึกค่าสิทธิ

นอกจากเรื่องของความปลอดภัย (security field) ยังมีอีกหลายฟิลด์ที่เกี่ยวข้องกับแต่ละออบเจคท์ เช่น flag,identification,object types เป็นต้น

- Bindery Object Types

จะมีการกำหนดค่าขนาด 2 ไบต์ขึ้นมาเพื่อแสดงจุดประสงค์ของแต่ละออบเจคท์ในบินเดอรี่ โดยค่านี้จะถูกเซตให้เป็น 0 เมื่อไม่ทราบชนิดของออบเจคท์นั้น ชนิดของออบเจคท์ที่ถูกกำหนดขึ้นนี้จะเป็นสิ่งที่แสดงถึงจุดประสงค์ของออบเจคท์นั้น เช่น ผู้พัฒนาสร้างออบเจคท์ชนิด 9 ชื่อ Archive Server ขึ้นเพื่อเป็นออบเจคท์ที่มีการปฏิบัติงานที่สอดคล้องกับโปรโตคอลของโนเวลล์ที่ชื่อ Archive Server

- The Bindery Object Flag

ออบเจคท์แฟลกจะมีขนาด 1 ไบต์ ใช้แสดงให้เห็นว่าออบเจคท์นั้นเป็นแบบ static (ให้ค่าเป็น-0) หรือ dynamic (ให้ค่าเป็น 1) โดยออบเจคท์แบบสแตติกจะคงอยู่ต่อไปจนกระทั่งถูกลบออกอย่างตั้งใจ ส่วนแบบไดนามิกนั้นจะถูกลบออกโดยอัตโนมัติเมื่อยกเลิกการทำงานของศูนย์บริการข้อมูล user เป็นตัวอย่างของออบเจคท์แบบสแตติก ส่วน Advertising file server ถือเป็นตัวอย่างของออบเจคท์แบบไดนามิก

- The Bindery Security Flag

เป็นแฟลกที่มีขนาด 1 ไบต์ ใช้กำหนดสิทธิให้ผู้ใช้ว่าอยู่ในระดับชั้นใดในระดับชั้นความปลอดภัย

3.5.1.2 Properties of Bindery Objects

พรอพเพอร์ตี้เป็นตัวกำหนดความสัมพันธ์ระหว่าง value กับออบเจคท์ว่า value ควรจะสัมพันธ์กับออบเจคท์ใด โดยพรอพเพอร์ตี้เหล่านี้จะสอดคล้องกับลักษณะพิเศษของออบเจคท์หรือไม่ก็ได้ พรอพเพอร์ตี้ของเน็ตแวร์ที่ถูกกำหนดไว้ได้แก่ GROUP_I'M_IN,IDENTIFICATION และSECURITY_EQUALS และเช่นเดียวกับออบเจคท์ พรอพเพอร์ตี้ก็จะถูกแสดงให้เห็นอยู่ในหลายๆฟิลด์ เช่น ชื่อ, แฟลก, security เป็นต้น

- The Bindery Property Flag

มีลักษณะเหมือนกับออบเจคท์แฟลก คือเป็นแฟลกขนาด 1 ไบต์ ใช้ระบุว่าพรอพเพอร์ตี้นั้นเป็นแบบไดนามิกหรือสแตติกเช่นเดียวกัน โดยเมื่อออบเจคท์แบบไดนามิกถูกลบไป ค่าพรอพเพอร์ตี้และ value ก็จะถูกลบไปด้วย แต่จะต่างกับออบเจคท์แฟลกตรงที่ บิตสูงสุดจะถือเป็นแฟลกที่ใช้ระบุว่า พรอพเพอร์ตี้นี้เป็นแบบ item (0) หรือเป็นแบบ set (1) โดย set เป็นรายชื่อของหมายเลขของออบเจคท์ (Object ID) ขนาด 4 ไบต์ ส่วน item เป็นส่วนฟิลด์ข้อมูลพิเศษซึ่งบรรจุข้อมูลของโครงสร้างใดๆไว้ เช่นรหัสผ่าน

- The Bindery Security Flag

มีลักษณะเหมือนกับออบเจคท์คือ เป็นแฟลกที่มีขนาด 1 ไบต์ ใช้กำหนดสิทธิในการอ่าน-บันทึกค่าพรอพเพอร์ตี้

โดยต่อไปนี้จะเป็นตัวอย่างของพรอพเพอร์ตี้ที่โนเวลล์กำหนดขึ้น

- ACCOUNT_SERVERS เป็นชุดของพรอพเพอร์ตี้ที่เก็บรายชื่อของทุกศูนย์บริการข้อมูลที่มีอำนาจต่อผู้ใช้ในการให้บริการ สิทธิในการเข้าถึงพรอพเพอร์ตี้นี้คือ Logged read , Supervisor write ออบเจคท์ของพรอพเพอร์ตี้นี้สามารถใช้เป็นศูนย์บริการข้อมูล, ศูนย์บริการการพิมพ์, ศูนย์บริการฐานข้อมูล, ผู้ใช้ หรือออบเจคท์แบบอื่นๆ

- ACCOUNT_LOGOUT เป็นพรอพเพอร์ตี้แบบสแตติกของศูนย์บริการข้อมูล ใช้เป็นคุณสมบัติที่บอกให้รู้ว่าแอคเคานท์ใดที่ทำการล็อกเอาท์ออกจากศูนย์บริการข้อมูลนั้นแล้ว สิทธิในการเข้าถึงปกติจะเป็น Supervisor ทั้ง read และ write

- GROUP_MEMBERS เป็นเซตของพรอพเพอร์ตี้แบบสแตติก บรรจุชื่อของสมาชิกของกลุ่มผู้ใช้ สิทธิการเข้าถึงปกติจะอยู่ที่ Logged read , Supervisor write

- GROUP_I'M_IN มีลักษณะตรงข้ามกับ GROUP_MEMBERS โดยเป็นเซตของพรอพเพอร์ตี้แบบสแตติกที่บรรจุรายชื่อของกลุ่มที่ผู้ใช้เป็นเจ้าของอยู่ สิทธิการเข้าถึงปกติจะเป็น Logged read , Supervisor write

- IDENTIFICATION เป็นพรอพเพอร์ตี้แบบสแตติกที่บรรจุชื่อเต็มของออบเจคท์ ไม่ว่าจะเป็นผู้ใช้หรือกลุ่มผู้ใช้ตามที่กำหนด มีความยาวตั้งแต่ 0-128 ไบต์

- LOGIN_CONTROL แสดงคุณสมบัติที่เป็นชนิดของผู้ใช้ โดยมี SynCon. เป็นตัวเก็บข้อมูลต่างๆที่เกี่ยวข้องด้วย เช่นความยาวของรหัสผ่านที่ต้องการ, วันหมดอายุของรหัสผ่าน เป็นต้น

3.5.1.3 Value

Value คือ ค่าของข้อมูลจริงที่ติดต่อกับโปรแกรม ค่า value นี้จะเป็นแบบ item หรือ set ก็ขึ้นอยู่กับค่า Bindery Property Flag โดยค่าข้อมูลนี้สามารถมีขนาดใหญ่กว่า 128 ไบต์ของแพ็กเก็ตตอบรับได้ ซึ่งแพ็กเก็ตก็จะทำการเชื่อมต่อหลายๆแพ็กเก็ตเข้าด้วยกันเอง (เป็น Multiple Packet)

3.5.1.4 Bindery Files

เน็ตแวร์ 286 จะมีบินเดอรี่ที่ประกอบไปด้วย 2 ไฟล์หลักคือ NET$BIND.SYS เก็บค่าของออบเจคท์และพรอพเพอร์ตี้ กับ NET$VAL.SYS เก็บค่าของ value ส่วนเน็ตแวร์ 386 จะมี 3 บินเดอรี่ไฟล์คือ NET$OBJ.SYS, NET$PROP.SYS, และ NET$VAL.SYS โดยไฟล์เหล่านี้จะถูกกำหนดไว้เป็นไฟล์แบบ Hidden& System และถูกเก็บอยู่ในไดเรคทอรี่ SYS:SYSTEM ของแต่ละศูนย์บริการข้อมูล โดยมีศูนย์บริการทำหน้าที่ปิดและเปิด

3.5.1.5 Calls in the Bindery API

ข้อมูลจะแจ้งให้ทราบว่า ฟังก์ชันใดใช้ได้ในเน็ตแวร์เวอร์ชันใดบ้าง มีประโยชน์อย่างไรบ้างแค่เพียงคร่าวๆดังต่อไปนี้

- OpenBindery เป็นฟังก์ชันที่มีใช้ในแอดวานซ์เน็ตแวร์ 1.0, 1.02, 2.0, 2.1 และในเน็ตแวร์ 386 เวอร์ชันต่างๆ โดยจะทำหน้าที่เป็นบินเดอรี่ ซึ่งจะต้องถูกเรียกใช้หลังจากมีการปิดการใช้งานบินเดอรี่โดยใช้ฟังก์ชัน CloseBindery แล้วเพราะบินเดอรี่อื่นจะไม่สามารถทำงานได้ ถ้ามีบินเดอรี่ที่ยังไม่ถูกเปิด

- CloseBindery เป็นฟังก์ชันที่มีใช้ในแอดวานซ์เน็ตแวร์ 1.0, 1.02, 2.0, 2.1 และเน็ตแวร์ 386 เวอร์ชันต่างๆ ทำหน้าที่ปิดบินเดอรี่ โดยมีเพียง supervisor หรือผู้มีฐานะเทียบเท่าเท่านั้น ที่ สามารถปิดบินเดอรี่ได้

- CreatBinderyObject มีใช้ในเน็ตแวร์เวอร์ชันต่างๆเหมือนฟังก์ชันที่กล่าวมา ทำหน้าที่สร้างออบเจคท์ขึ้นในบินเดอรี่ โดยต้องให้โปรแกรมเมอร์เป็นผู้ระบุชื่อของออบเจคท์, ชนิด, ระดับความปลอดภัย และค่าในแฟลกต่างๆของออบเจคท์

- ScanBinderyObject ทำหน้าที่สแกนบินเดอรี่ ส่งกลับค่าในรูปของออบเจคท์และข้อมูลต่างๆที่ออบเจคท์ในบินเดอรี่ (ได้แก่ ID, ชื่อ, ชนิด และค่าในแฟลกต่างๆ)

3.5.2 การบริการด้านข้อความ (Message Services)

การบริการด้านข้อความ เป็นบริการระดับสูง สำหรับการเชื่อมต่อในระบบเครือข่ายซึ่งถูกออกแบบมาสำหรับการส่งข้อความจากคอมพิวเตอร์เครื่องหนึ่ง ไปยังคอมพิวเตอร์อีกเครื่องหนึ่งในระบบเครือข่ายโดยลักษณะของข้อความนี้ จะแบ่งออกเป็น 2 ลักษณะ คือ

1. **ข้อความที่มีลักษณะการส่งแบบกระจาย** (Broadcast Messages) เช่น การส่งข้อความโดยใช้คำสั่ง "SEND" ในระบบเครือข่ายแบบท้องถิ่นที่ใช้เน็ตแวร์

2. **ข้อความที่มีลักษณะการส่งแบบท่อสัญญาณ** (Pipe Messages) ซึ่งจะใช้สำหรับการติดต่อสื่อสารที่มีจำนวนข้อมูลมากๆ โดยจะส่งข้อความเป็นแถวยาวผ่านท่อสัญญาณได้จำนวน 6 ข้อความ

ในการนำมาใช้งานต้องคำนึงถึงว่า การบริการด้านข้อความเป็นเครื่องมือระดับสูงในการติดต่อสื่อสารระหว่างคอมพิวเตอร์ในระบบเครือข่าย รวมไปถึงศูนย์บริการข้อมูล ดังนั้นจึงไม่เหมาะกับการใช้งานกับระบบเครือข่ายคอมพิวเตอร์แบบที่ไม่มีศูนย์บริการข้อมูล โดยการบริการด้านข้อความนี้จะสามารถเรียกใช้ได้ในระบบจัดการเครือข่ายแบบเน็ตแวร์ ที่เวอร์ชันสูงกว่าเน็ตแวร์ 386 เนื่องจากการติดต่อสื่อสารระหว่างคอมพิวเตอร์ในระบบเครือข่าย โ ดยใช้การบริการด้านข้อความนี้ จะต้องกระทำโดยใช้โปรโตคอล IPX หรือ SPX

3.5.2.1 ข้อความที่มีลักษณะการส่งแบบกระจาย

ข้อความที่มีลักษณะการส่งแบบกระจายนี้ เป็นที่รู้จักกันดีโดยการใช้คำสั่ง "SEND" ในระบบเครือข่ายแบบท้องถิ่นที่ใช้เน็ตแวร์ ซึ่งสามารถส่งข้อความได้ความยาวถึง 55 ไบต์ โดยจะแสดงข้อความในบรรทัดที่ 25 ของหน้าจอคอมพิวเตอร์ที่เราติดต่อด้วย และข้อความจะปรากฏพร้อมกับเสียงบี๊บและจะรอจนกว่าผู้ใช้จะกดปุ่ม Ctrl-Enter ข้อความก็จะหายไป โดยกรรมวิธีในการส่งข้อความเช่นนี้สามารถนำไปประยุกต์ใช้ในการส่งข้อความไปยังผู้ใช้อื่นๆในระบบเครือข่ายหรือศูนย์บริการข้อมูลได้

- โหมดของข้อความที่มีลักษณะการส่งแบบกระจาย (Broadcast Message Mode)

สถานีผู้ใช้แต่ละสถานีจะมีส่วนรองรับข้อมูล (Buffer) จำนวน 55 ไบต์ สำหรับข้อความที่มีลักษณะการส่งแบบกระจาย และเนื่องจากมันมีส่วนรองรับข้อมูลเพียงชุดเดียว ถ้ามีข้อความเข้ามามากกว่า 1 ชุด ข้อความชุดที่มาถึงสถานีผู้ใช้ก่อนจะปรากฏเพียงข้อความเดียวจนกว่าจะมีการลบข้อความชุดนี้ ข้อความชุดต่อๆมาจึงจะปรากฏ โดยศูนย์บริการข้อมูลก็มีส่วนรองรับข้อมูลด้วยเช่นกัน ดังนั้นจึงจะมีข้อความ 2 ชุดที่อยู่บนเส้นทางการส่งข้อมูล คือจะอยู่ที่สถานีผู้ใช้ 1 ข้อความและอยู่ที่ศูนย์บริการข้อมูลอีก 1 ข้อความ สำหรับการส่งข้อความที่มีข้อความจำนวนมากๆจึงจำเป็นต้องใช้การส่งข้อความที่มีลักษณะการส่งแบบท่อสัญญาณ

ข้อความที่มีลักษณะการส่งแบบกระจายนี้ สามารถส่งได้โดยสถานีผู้ใช้ใดๆหรือศูนย์บริการข้อมูลไปยังสถานีผู้ใช้อื่นๆหรือศูนย์บริการข้อมูลได้ โ ดยศูนย์บริการข้อมูลจะอยู่ในรูปของสถานีผู้ใช้สถานีหนึ่งเมื่อมีข้อความส่งเข้ามา ทั้งนี้ขึ้นอยู่กับการเลือกโหมดของสถานีผู้ใช้และมีเพียงข้อความจากศูนย์บริการข้อมูลของมันเองเท่านั้นที่จะสามารถทับข้อมูลเก่าที่อยู่ในส่วนรองรับข้อมูลของศูนย์บริการข้อมูลนั้นๆได้

โหมดในการส่งข้อความที่มีลักษณะการส่งแบบกระจายแบ่งออกเป็น 4 โหมด ซึ่งขึ้นอยู่กับการเลือกของผู้ใช้ โดยการเรียกใช้ฟังก์ชัน "Get Broadcast Mode" และ "Set Broadcast Mode" เพื่อที่จะเป็นการกำหนดว่าศูนย์บริการข้อมูลจะจัดการกับข้อความที่ถูกส่งมาอย่างไร

โหมด 0 : โหมดนี้จะถูกเลือกเมื่อผู้ใช้ไม่ได้เลือกโหมดอื่น โดยศูนย์บริการจะรับข้อความทั้งของศูนย์บริการและของผู้ใช้ และเชลล์จะทำการแสดงข้อความแต่ละข้อความบนจอภาพโดยอัตโนมัติ

โหมด 1 : โหมดนี้ศูนย์บริการจะรับเฉพาะข้อความของศูนย์บริการ ส่วนข้อความของผู้ใช้จะไม่สนใจ คือข้อความของสถานีผู้ใช้จะไม่ทำการแสดงที่สถานีนี้ และเซลล์จะทำการแสดงข้อความของศูนย์บริการบนจอภาพโดยอัตโนมัติ

โหมด 2 : โหมดนี้ศูนย์บริการจะรับเพียงข้อความจากศูนย์บริการเท่านั้น แต่เซลล์จะไม่ทำแสดงข้อความของศูนย์บริการโดยอัตโนมัติ แต่จะทำการแสดงข้อความนั้นๆ เมื่อมีการเรียกใช้ฟังก์ชัน "Get Broadcast Message"

โหมด 3 : โหมดนี้ศูนย์บริการจะรับทั้งข้อความจากศูนย์บริการและสถานีผู้ใช้ แต่จะไม่ทำการแสดงข้อความทั้ง 2 ประเภทนั้นโดยอัตโนมัติ จะทำการแสดงข้อความขึ้นมาก็ต่อเมื่อมีการเรียกใช้ฟังก์ชัน "Get Broadcast Message"

- การใช้การส่งข้อความที่มีลักษณะการส่งแบบกระจาย (Using Broadcast Message)

การส่งข้อความที่มีลักษณะการส่งแบบกระจายมีลักษณะในการเรียกใช้งาน 3 แบบคือ

1. "Send Broadcast Message" จะทำการส่งข้อความที่มีลักษณะการส่งแบบกระจายผ่านศูนย์บริการข้อมูลไปยังสถานีผู้ใช้ปลายทางใดๆ

2. "Get Broadcast Message" จะทำการรับข้อความที่มีลักษณะการส่งแบบกระจาย

3. "Broadcast to Console" จะทำการส่งข้อความไปยังศูนย์บริการข้อมูลของระบบเครือข่าย

3.5.2.2 ข้อความที่มีลักษณะการส่งแบบท่อสัญญาณ

การส่งข้อความที่มีลักษณะการส่งแบบท่อสัญญาณนั้น จะมีฟังก์ชันการทำงานมากกว่าการส่งข้อความที่มีลักษณะการส่งแบบกระจาย โดยสามารถส่งข้อความได้ 6 ลำดับของข้อความ 126 ไบต์ ในการใช้การส่งข้อความที่มีลักษณะการส่งแบบท่อสัญญาณ จะต้องเริ่มต้นด้วยการเรียกใช้ฟังก์ชัน "Open Message Pipe" เพื่อทำการเปิดช่องทางการสื่อสารของสถานีที่จะทำการติดต่อด้วย และเมื่อช่องทางการสื่อสารนี้ได้ถูกเปิดแล้ว การส่งข้อความจะสามารถทำการส่งได้โดยการเรียกใช้ฟังก์ชัน "Send Personal Message" และในการรับข้อความจะทำได้โดยการเรียกใช้ฟังก์ชัน "Get Personal Message" ซึ่งจะทำการดึงข้อความมาจากแถวข้อมูลของสถานีผู้ใช้ ช่องทางการสื่อสารนี้จะถูกปิดลงเมื่อมีการเรียกใช้ฟังก์ชัน "Close Message Pipe" เมื่อการสื่อสารสิ้นสุด ถ้าช่องทางการสื่อสารไม่ถูกปิดลง ข้อความจะมาทับถมกันอยู่ที่แถวของข้อมูล ทำให้สถานีที่ทำการส่งไม่สามารถแยกแยะได้ว่า ข้อความนั้นๆได้ทำการแสดงไปแล้วหรือยังไม่ได้แสดง

การส่งข้อความที่มีลักษณะการส่งแบบท่อสัญญาณ แต่ละสถานีจะสามารถเก็บข้อความได้ถึง 6 ลำดับของข้อความ 126 ไบต์ โดยข้อความที่อยู่ในแถวของข้อมูลนี้จะทำการส่งแบบ FIFO คือข้อความที่ถูกส่งเข้ามาก่อนก็จะถูกส่งออกไปก่อนตามลำดับ ซึ่งจะจัดการโดยศูนย์บริการข้อมูล สำหรับสถานะของช่องทางการสื่อสารจะสามารถทราบได้จากการเรียกใช้ฟังก์ชัน "Check Pipe Status"

- **การใช้การส่งข้อความที่มีลักษณะการส่งแบบท่อสัญญาณ** (Using Pipe Message)

การส่งข้อความที่มีลักษณะการส่งแบบท่อสัญญาณ มีลักษณะในการเรียกใช้งาน 5 แบบคือ

1. Open Message Pipe จะทำการเปิดช่องทางการสื่อสารของสถานีที่จะทำการติดต่อด้วย
2. Close Message Pipe จะทำการปิดช่องทางการสื่อสาร
3. Send Personal Message จะใช้ในการส่งข้อความ
4. Get Personal Message จะใช้ในการรับข้อความ
5. Check Pipe Status จะทำการตรวจสอบสถานะของช่องทางการสื่อสาร

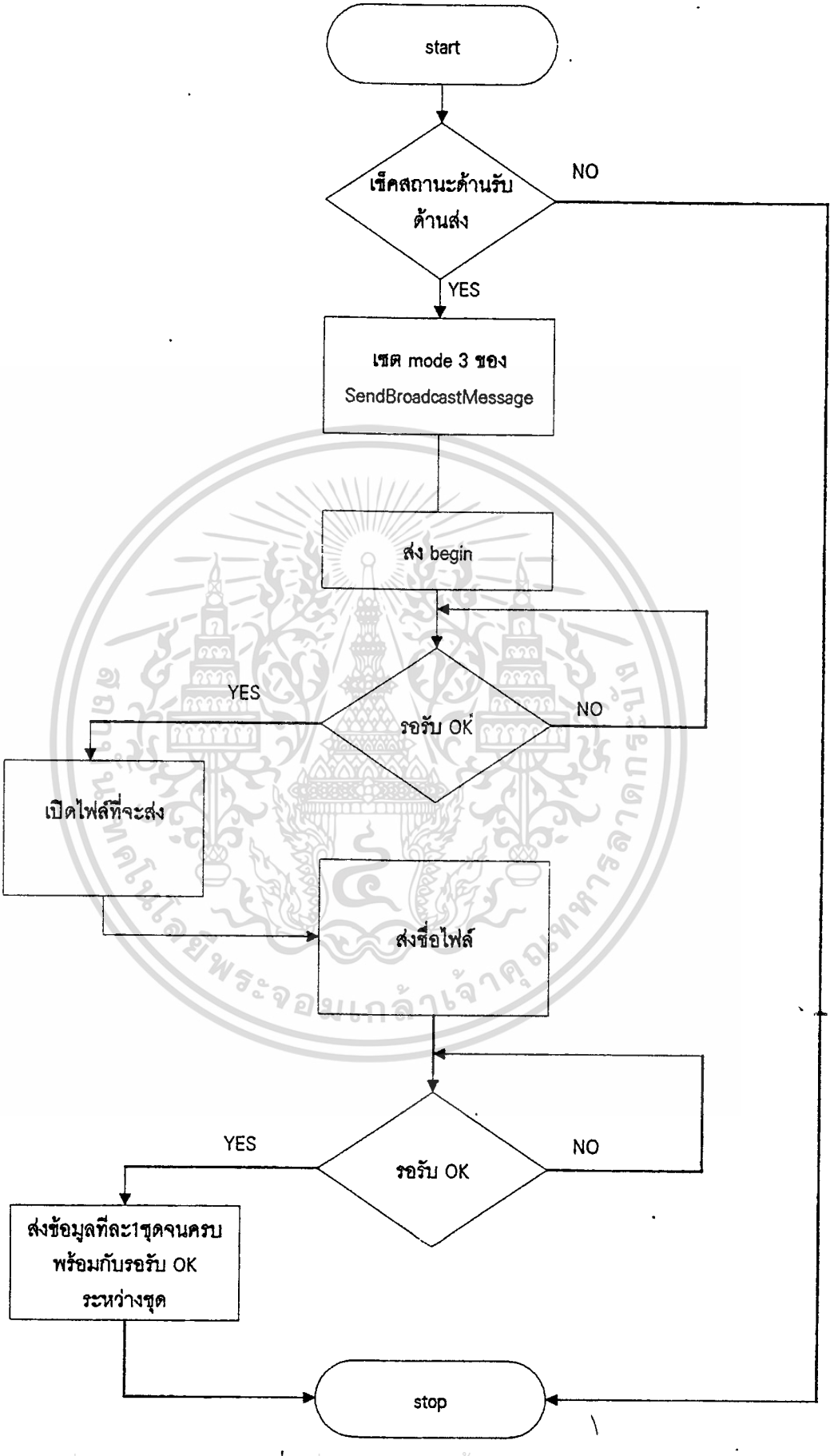
บทที่ 4
แนวความคิดในการทดลอง

การติดต่อสื่อสารในระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น โดยปกติแล้วมักจะติดต่อสื่อสารกันในรูปของข้อมูลที่เป็นตัวอักษร เพื่อให้เกิดความหลากหลายในการติดต่อสื่อสารในระบบเครือข่าย โครงงานนี้จึงมีแนวความคิดที่จะทำการรับและส่งภาพผ่านระบบเครือข่าย โดยอาศัยหลักการเขียนโปรแกรมภายใต้ข้อกำหนดในการเขียนโปรแกรมติดต่อกับระบบจัดการเน็ตแวร์และการเรียกใช้ฟังก์ชันในบริการของเน็ตแวร์

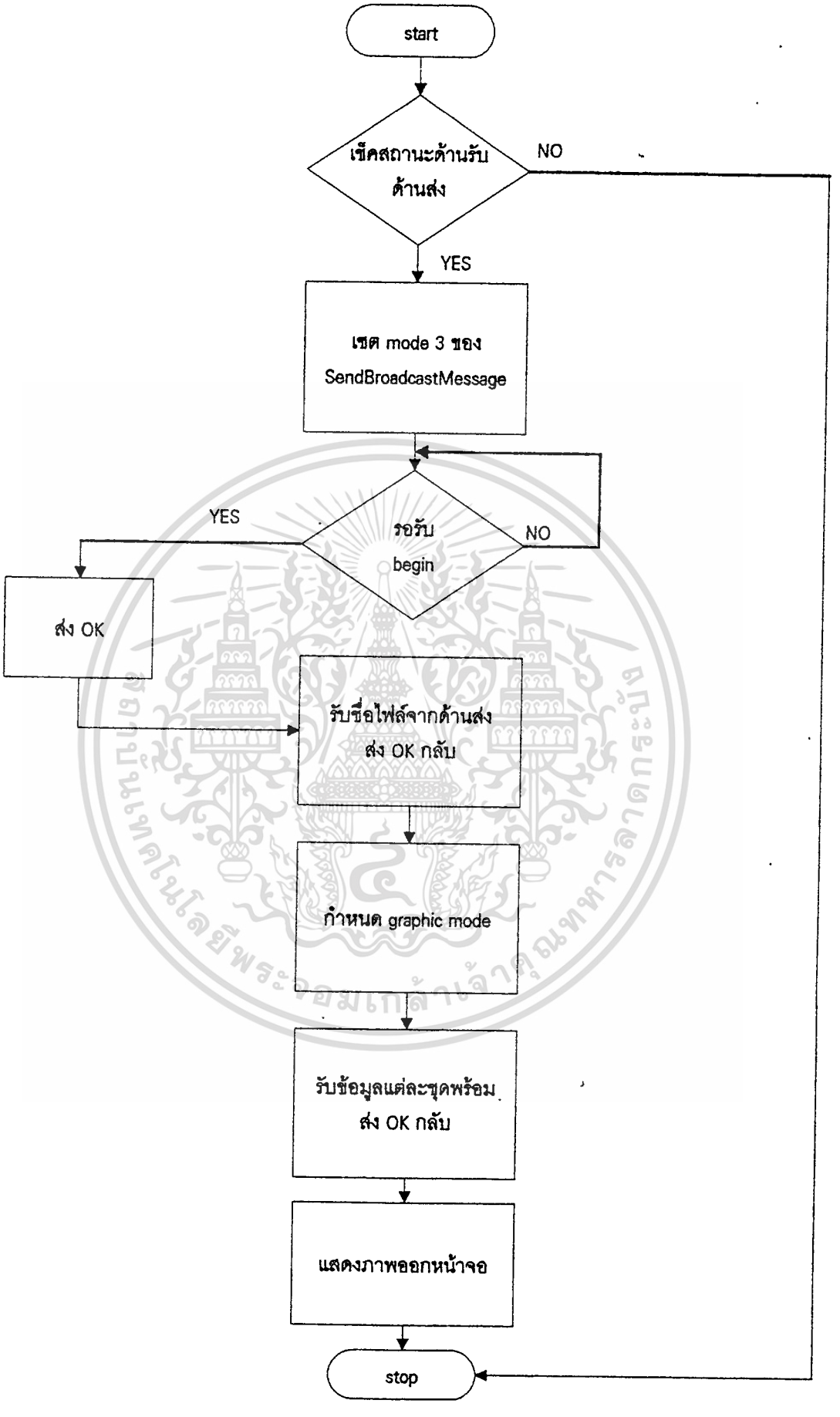
โดยลักษณะของภาพที่จะทำการรับส่งนั้น แบ่งออกเป็น 3 ลักษณะคือ

1. ภาพขาวดำที่มีลักษณะการส่งแบบ on-line
2. ภาพขาวดำที่มีลักษณะการส่งแบบ off-line
3. ภาพสีรูปแบบต่างๆที่มีลักษณะการส่งแบบ off-line

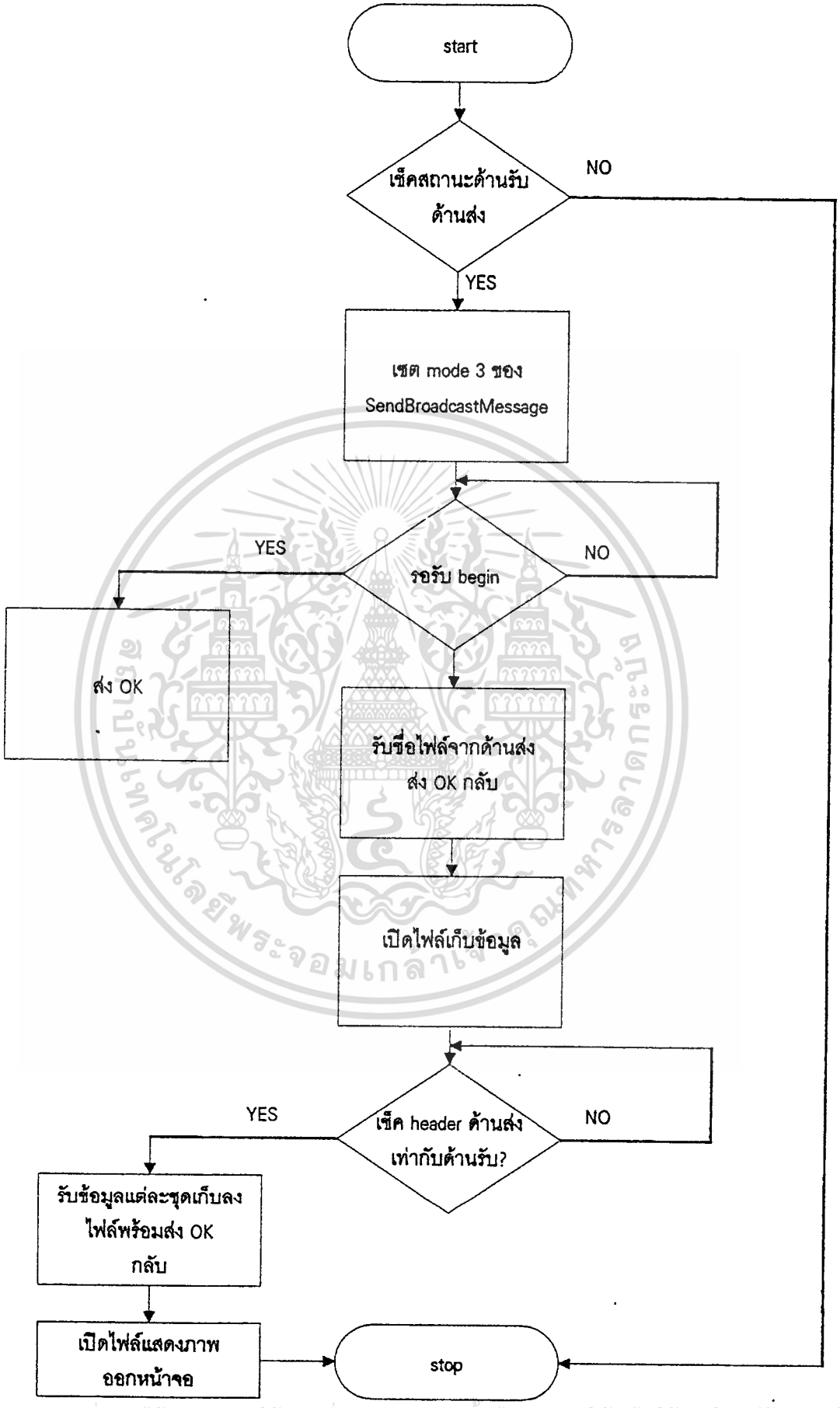
ซึ่งขั้นตอนในการทำงานของโปรแกรมรับและส่งภาพต่างๆแสดงได้ดังโฟล์วชาร์ต



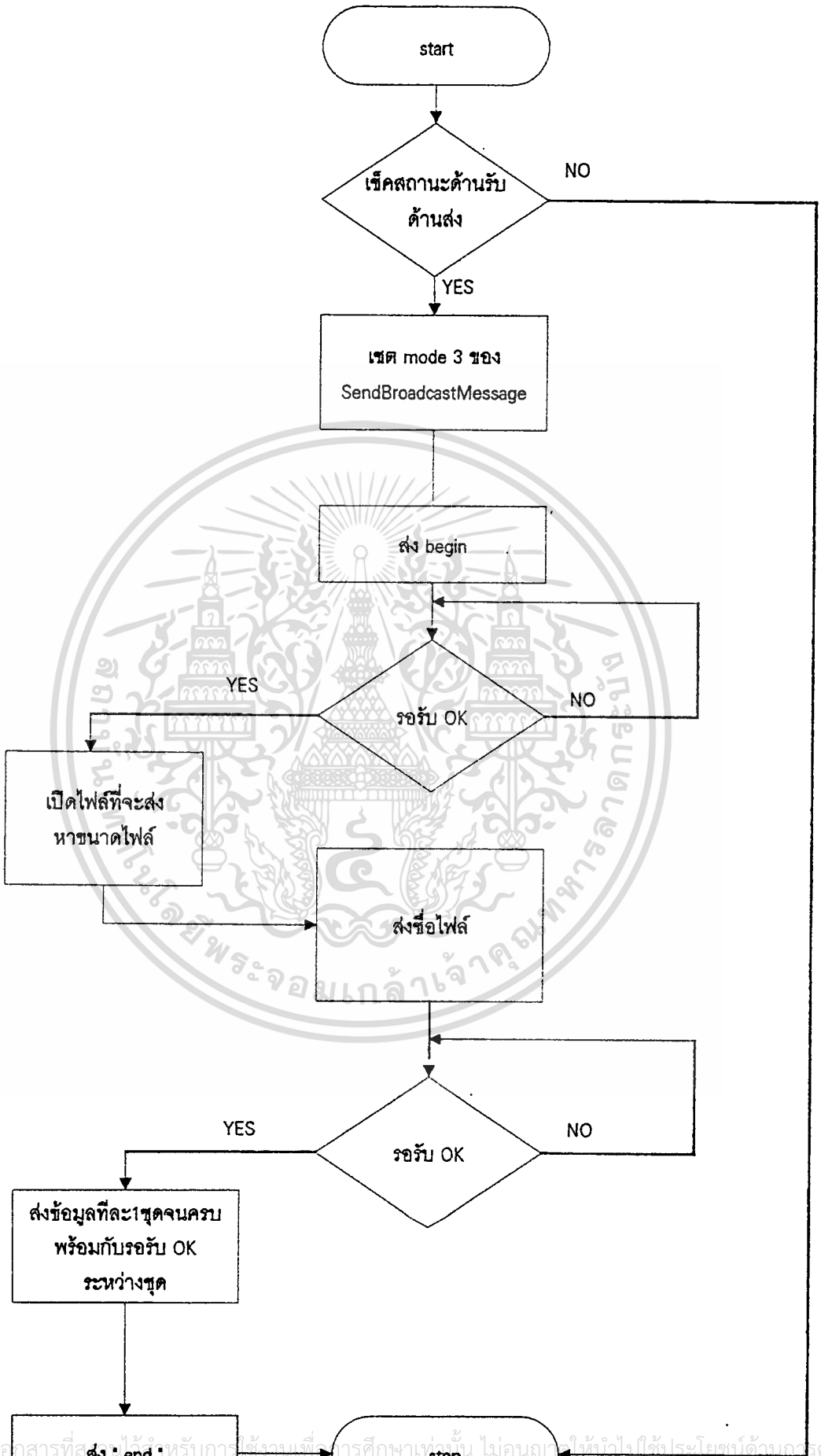
โฟล์วชาร์ตที่ 4.1 โปรแกรมส่งภาพขาวดำนามสกุล GS



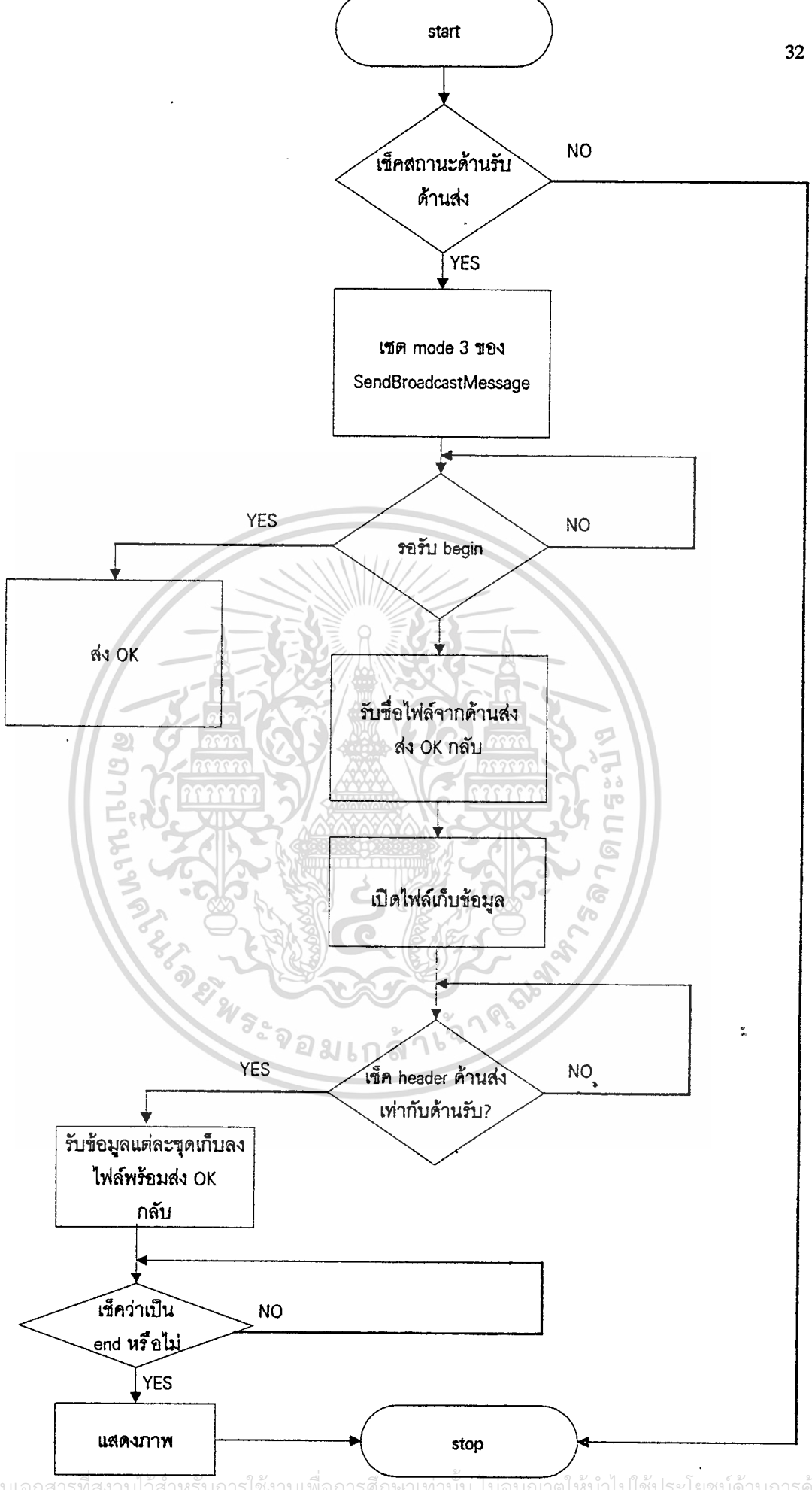
โฟล์วชาร์ตที่ 4.2 โปรแกรมรับภาพขาวดำนามสกุล GS .....(แบบ on-line)



โฟล์วชาร์ตที่ 4.3 โปรแกรมรับภาพขาวดำนามสกุล GS .....(แบบ off-line)



ฟลัวชาร์ตที่ 4.4 โปรแกรมส่งภาพสี



โฟล์วชาร์ตที่ 4.5 โปรแกรมรับภาพสี.....(แบบ off-line)

บทที่ 5
การทดลองและผลการทดลอง

การทดลอง

การทดลองสำหรับโครงงานนี้ คือ การรับและส่งภาพผ่านระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น โดยการทดลองนี้ จะแบ่งออกเป็น 4 หัวข้อ คือ

1. การส่งและรับภาพขาวดำในลักษณะ on-line ซึ่งขั้นตอนการทำงานของโปรแกรม แสดงได้ดังโฟล์วชาร์ตที่ 4.1 และ 4.2 ตามลำดับ

2. การส่งและรับภาพขาวดำในลักษณะ off-line ซึ่งขั้นตอนการทำงานของโปรแกรม แสดงได้ดังโฟล์วชาร์ตที่ 4.1 และ 4.3 ตามลำดับ

3. การส่งและรับภาพสีที่มีรูปแบบต่างๆ ในลักษณะ off-line ซึ่งขั้นตอนการทำงานของโปรแกรมแสดงได้ดังโฟล์วชาร์ตที่ 4.4 และ 4.5 ตามลำดับ

4. ทำการเปรียบเทียบไฟล์รูปภาพ ระหว่างด้านรับและด้านส่งไบต์ต่อไบต์ เพื่อตรวจสอบดูว่ามีความผิดพลาดของข้อมูล อันเนื่องมาจากการส่งผ่านระบบเครือข่ายหรือไม่

นอกจากนั้นยังได้ทำการทดลองการส่งและรับภาพ ในระบบเครือข่ายที่มีการใช้งานจริง (ปกติจะทำการทดลองผ่านระบบเครือข่ายที่สร้างขึ้นเอง) เป็นจำนวน 20 ครั้ง ดังตารางที่ 5.1

ครั้งที่	ผลการส่งและรับภาพ
1	ถูกต้อง
2	ถูกต้อง
3	ถูกต้อง
4	ถูกต้อง
5	ถูกต้อง
6	ถูกต้อง
7	ถูกต้อง
8	ถูกต้อง
9	ถูกต้อง
10	ถูกต้อง
11	ถูกต้อง
12	ถูกต้อง

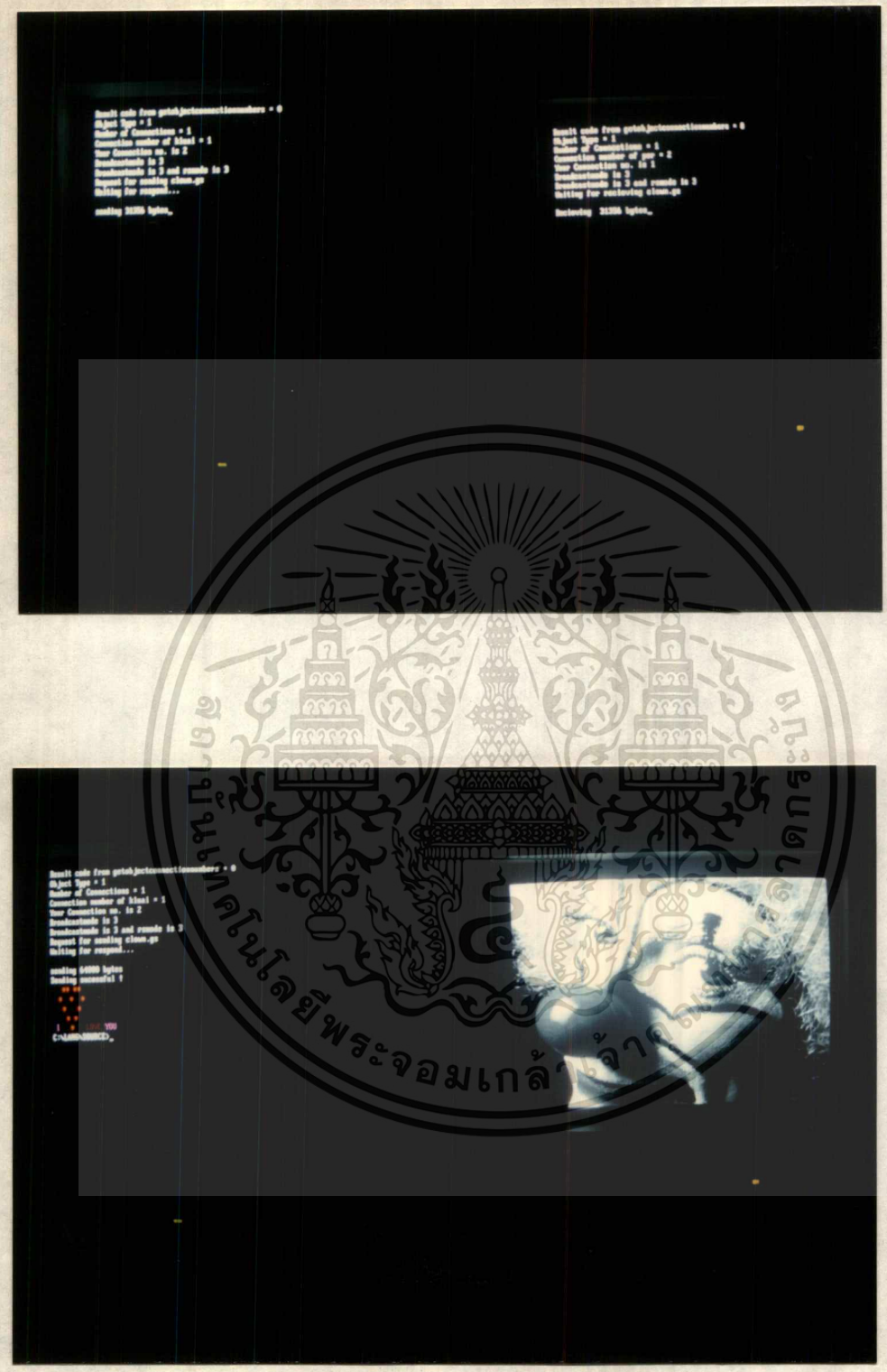
ครั้งที่	ผลการส่งและรับภาพ
13	ถูกต้อง
14	ถูกต้อง
15	ถูกต้อง
16	ถูกต้อง
17	ถูกต้อง
18	ถูกต้อง
19	ถูกต้อง
20	ถูกต้อง

ตารางที่ 5.1 แสดงผลการส่งและรับภาพในระบบเครือข่ายที่มีการใช้งานจริง

ผลการทดลอง

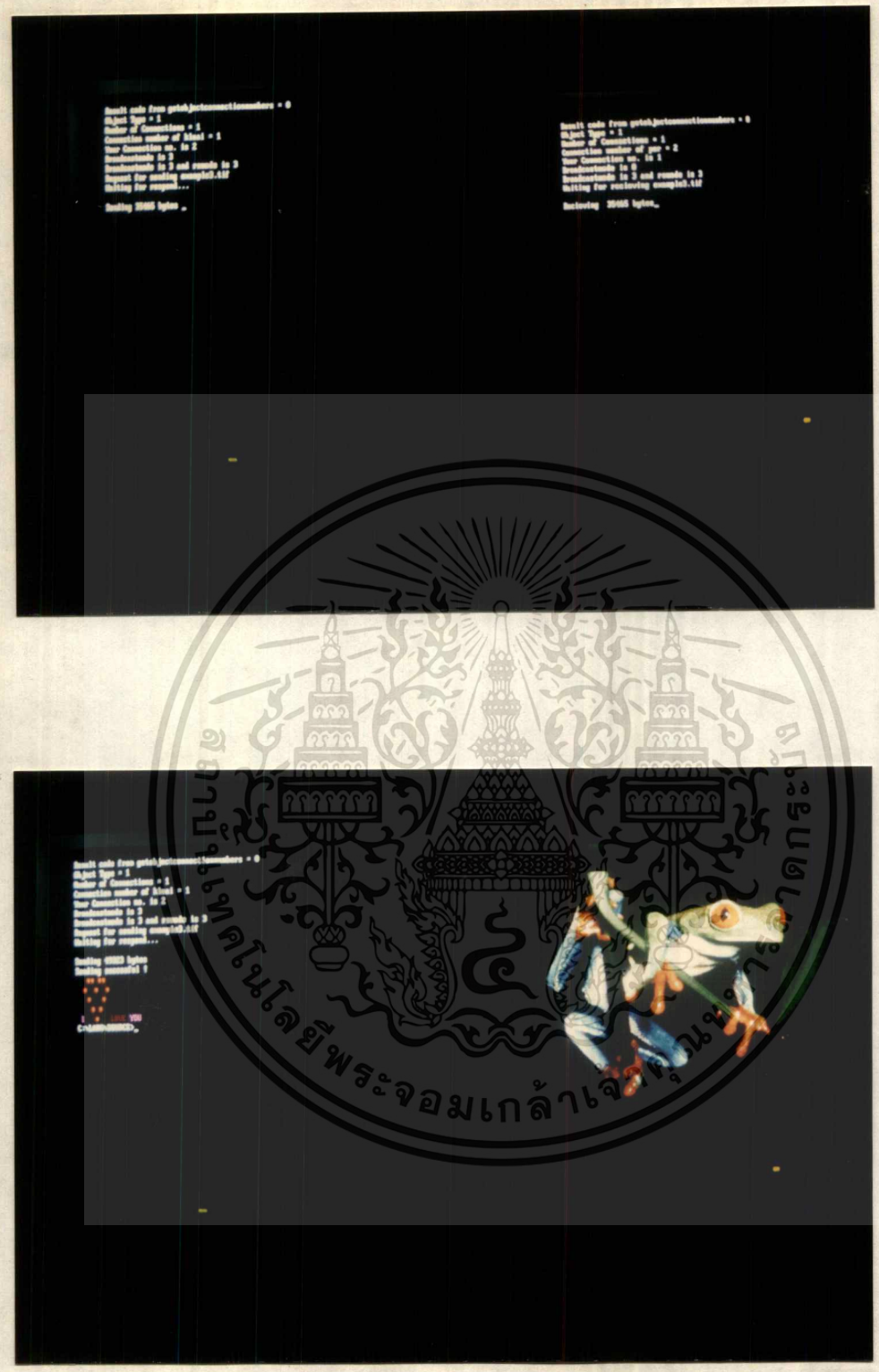
จากการทดลองรันโปรแกรมต่างๆข้างต้น ผลที่ได้ คือ สามารถส่งภาพระหว่างคอมพิวเตอร์ 2 เครื่อง ได้อย่างถูกต้อง ซึ่งผลการทดลองที่ได้แสดงดังรูปที่ 5.1-5.5 สำหรับผลการเปรียบเทียบไฟล์รูปภาพระหว่างด้านรับและด้านส่ง แสดงได้ดังรูปที่ 5.6-5.9

และจากตารางที่ 5.1 จะเห็นได้ว่าในการส่งและรับภาพทั้ง 20 ครั้ง ไม่เกิดความผิดพลาดขึ้นเลย ซึ่งนับว่าเป็นที่น่าพอใจมาก

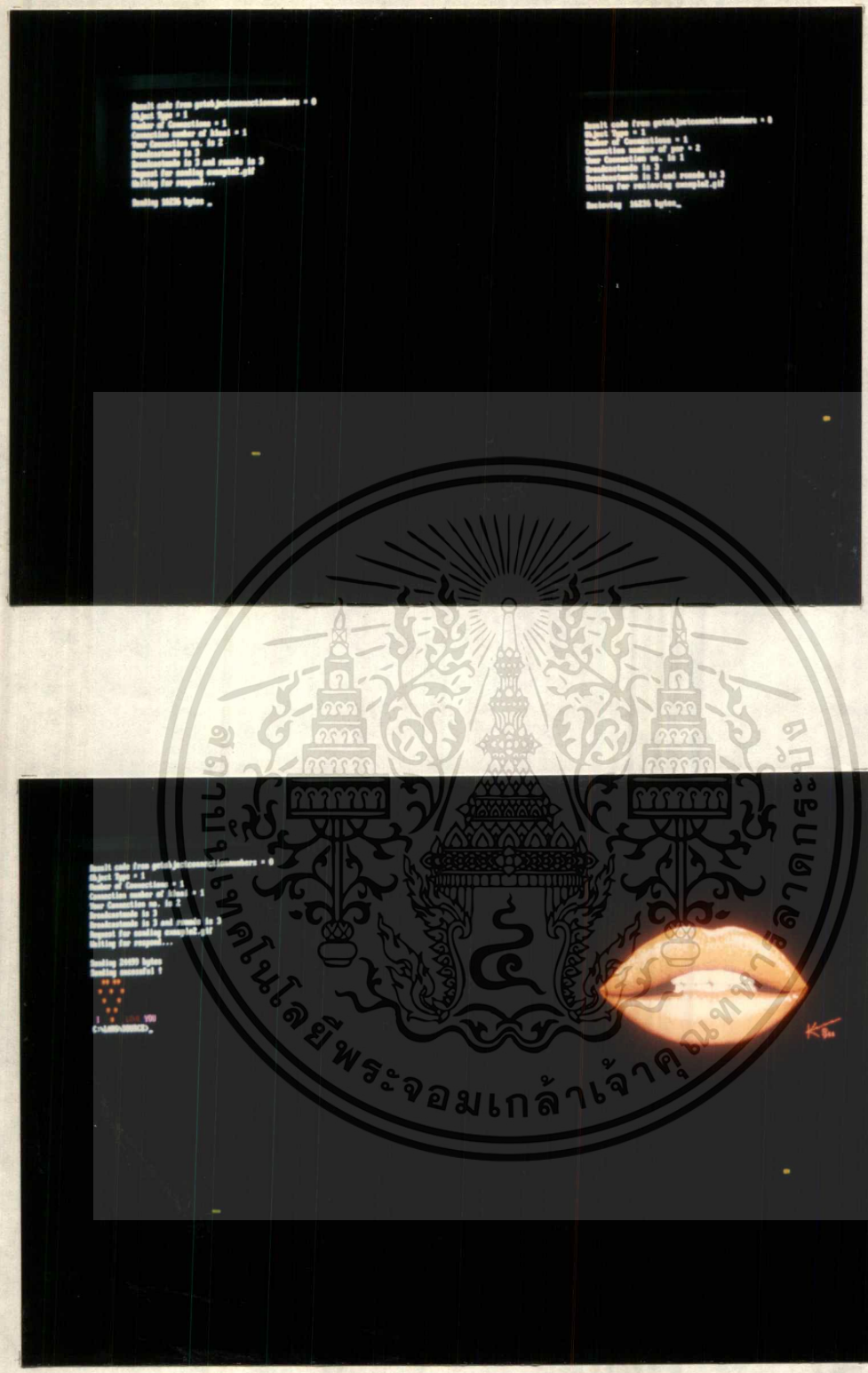


รูปที่ 5.2 ผลการทดลองจากการรันโปรแกรมรับและส่งภาพขาวดำ นามสกุล GS แบบ off-line

รูปที่ 5.3 ผลการทดลองจากการรันโปรแกรมรับและส่งภาพสี นามสกุล PCX แบบ off-line



รูปที่ 5.4 ผลการทดลองจากการรันโปรแกรมรับและส่งภาพสี นามสกุล TIF แบบ off-line



รูปที่ 5.5 ผลการทดลองจากการรันโปรแกรมรับและส่งภาพสี นามสกุล GIF แบบ off-line

Byte No.	Transmitter(Hex)	Receiver(Hex)	Result(Dec)
1	51	51	NO ERROR
2	58	58	NO ERROR
3	53	53	NO ERROR
4	53	53	NO ERROR
5	50	50	NO ERROR
6	50	50	NO ERROR
7	53	53	NO ERROR
8	51	51	NO ERROR
9	50	50	NO ERROR
10	50	50	NO ERROR
11	53	53	NO ERROR
12	51	51	NO ERROR
13	50	50	NO ERROR
14	4B	4B	NO ERROR
15	4B	4B	NO ERROR
16	50	50	NO ERROR
17	51	51	NO ERROR
18	50	50	NO ERROR

Press anykey to continue, q = quick analyze, ESC = exit

Byte No.	Transmitter(Hex)	Receiver(Hex)	Result(Dec)
19	51	51	NO ERROR
20	4C	4C	NO ERROR
21	4B	4B	NO ERROR
22	49	49	NO ERROR
23	49	49	NO ERROR
24	49	49	NO ERROR
25	49	49	NO ERROR
26	49	49	NO ERROR
27	4B	4B	NO ERROR
28	51	51	NO ERROR
29	51	51	NO ERROR
30	51	51	NO ERROR
31	51	51	NO ERROR
32	51	51	NO ERROR
33	51	51	NO ERROR
34	51	51	NO ERROR
35	51	51	NO ERROR
36	51	51	NO ERROR

Press anykey to continue, q = quick analyze, ESC = exit

รูปที่ 5.6 แสดงผลการรันโปรแกรมเพื่อเปรียบเทียบไฟล์รูปภาพระหว่างด้านส่งและด้านรับ แบบไบต์ต่อไบต์ ของภาพ lisaw.gs

```
Compare errors between lisaw1.gs and lisaw2.gs
Transmitter filename is    lisaw1.gs
Receiver filename is    lisaw2.gs
Transmitter filesize  =      64000 bytes
Receiver filesize     =      64000 bytes
Number of byte error  =          0 bytes
Transmitter file isn't different from the receiver file.
```

รูปที่ 5.7 แสดงผลลัพธ์ของการรันโปรแกรมเพื่อตรวจสอบความผิดพลาดจากการส่งและรับภาพ lisaw.gs ในระบบเครือข่าย

Byte No.	Transmitter(Hex)	Receiver(Hex)	Result(Dec)
1	47	47	NO ERROR
2	49	49	NO ERROR
3	46	46	NO ERROR
4	38	38	NO ERROR
5	39	39	NO ERROR
6	61	61	NO ERROR
7	80	80	NO ERROR
8	2	2	NO ERROR
9	90	90	NO ERROR
10	1	1	NO ERROR
11	B3	B3	NO ERROR
12	0	0	NO ERROR
13	0	0	NO ERROR
14	0	0	NO ERROR
15	0	0	NO ERROR
16	0	0	NO ERROR
17	EE	EE	NO ERROR
18	BB	BB	NO ERROR

Press anykey to continue, q = quick analyze, ESC = exit

Byte No.	Transmitter(Hex)	Receiver(Hex)	Result(Dec)
19	BB	BB	NO ERROR
20	CC	CC	NO ERROR
21	99	99	NO ERROR
22	99	99	NO ERROR
23	99	99	NO ERROR
24	66	66	NO ERROR
25	66	66	NO ERROR
26	FF	FF	NO ERROR
27	AA	AA	NO ERROR
28	AA	AA	NO ERROR
29	FF	FF	NO ERROR
30	AA	AA	NO ERROR
31	AA	AA	NO ERROR
32	FF	FF	NO ERROR
33	66	66	NO ERROR
34	66	66	NO ERROR
35	CC	CC	NO ERROR
36	0	0	NO ERROR

Press anykey to continue, q = quick analyze, ESC = exit

รูปที่ 5.8 แสดงผลการรันโปรแกรมเพื่อเปรียบเทียบไฟล์รูปภาพระหว่างด้านส่งและด้านรับ แบบไบต์ต่อไบต์ ของภาพ example.gif .

```
Compare errors between example1.gif and example2.gif
Transmitter filename is example1.gif
Receiver filename is example2.gif
Transmitter filesize  =      24499 bytes
Receiver filesize     =      24499 bytes
Number of byte error  =          0 bytes
Transmitter file isn't different from the receiver file.
```

รูปที่ 5.9 แสดงผลลัพธ์ของการรันโปรแกรมเพื่อตรวจสอบความผิดพลาดจากการส่งและรับภาพ example.gif ในระบบเครือข่าย

บทที่ 6
สรุปผลการทดลอง ปัญหา แนวทางในการพัฒนา

สรุปผลการทดลอง

จากการทดลองการใช้งานโปรแกรมรับและส่งภาพผ่านระบบเครือข่ายคอมพิวเตอร์แบบท้องถิ่น สามารถส่งและรับภาพระหว่างคอมพิวเตอร์ 2 เครื่อง ได้อย่างถูกต้อง คือ ภาพที่ด้านรับจะเหมือนกับด้านส่งทุกประการ ซึ่งสามารถทำการตรวจสอบความผิดพลาดได้ด้วยโปรแกรมเปรียบเทียบไฟล์รูปภาพระหว่างด้านรับและด้านส่ง และสามารถทำการส่งและรับได้ทั้งภาพขาวดำและภาพสี แต่ความเร็วในการส่งยังนับว่าไม่เร็วนัก เนื่องจากข้อจำกัดของกรรมวิธีการส่งที่ใช้

ปัญหาที่พบจากการทดลอง

1. การรับส่งภาพนี้ยังมีข้อจำกัด คือสามารถทำการรับส่งภาพได้เพียงครั้งละ 2 เครื่องเท่านั้น
2. ความเร็วในการส่งยังไม่ค่อยสูงเป็นที่น่าพอใจ เนื่องจากข้อจำกัดของกรรมวิธีในการส่งที่ใช้

ข้อจำกัดของการส่งข้อความแบบกระจาย (Broadcast Message) คือ

- ความยาวในการส่งแต่ละครั้งสูงสุดเพียง 55 ไบต์
- เป็นการส่งแบบสตริง จึงก่อให้เกิดปัญหาเมื่อเจอ 0 (binary) ในข้อมูลภาพ โดยในโครงงานนี้แก้ปัญหาโดย เมื่อเจอ 0 จะทำการส่ง 0 ไปทีละไบต์ จึงมีผลกระทบต่อความเร็วในการรับส่งภาพส่วนหนึ่งด้วย
- ส่วนรองรับข้อมูลที่มีจำกัด (buffer) คือ มีเพียง 2 ตัว ที่สถานีผู้ใช้และศูนย์บริการข้อมูล และตราบใดที่ข้อมูลในส่วนรองรับข้อมูลยังไม่ได้ถูกส่งออกไป ข้อมูลอื่นๆที่ถูกส่งเข้ามา ก็จะไม่สามารถส่งออกไปได้

3. การซิงโครไนส์ระหว่างด้านส่งและด้านรับ ทำได้ค่อนข้างยาก การติดต่อระหว่างผู้ใช้ต้องใช้ศูนย์บริการข้อมูลเป็นตัวสื่อกลาง มิใช่การติดต่อกันโดยตรง

แนวทางในการพัฒนา

1. ปรับปรุงโปรแกรมเพื่อให้การรับส่งสามารถกระทำได้มากกว่า 2 เครื่องขึ้นไป

2. พัฒนากรรมวิธีการส่งอื่นที่อาจมีประสิทธิภาพสูงกว่า เช่น ลักษณะการส่งข้อความแบบท่อ (Pipe Message) เป็นต้น

3. ปรับปรุงโปรแกรมเพื่อให้สามารถรับส่งภาพเคลื่อนไหวได้

ภาคผนวก

```
/*************************************************************************/
*                                                                       *
*        Send Image (Gray Scale) By Using Broadcast Messages            *
*                                                                       *
/*************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <dir.h>
#include <io.h>

/***************************************/
/* Bindery object type classifications */
/***************************************/c

#define   ESC           27

#define UNKNOWN         0x0000 /* unknown object */
#define USER            0x0001 /* User type object */

/****************/
/* All typedefs */
/****************/

typedef unsigned long    dword;      /* four bytes */
typedef unsigned int     word;       /* two bytes */
typedef unsigned char    byte;       /* single byte */

typedef struct {
   unsigned char hi_byte;       /* NetWare int, */
   unsigned char lo_byte;       /* hibyte first (inverse native PC) */
   } nw_int;

typedef struct {
   unsigned char hihi_byte;    /* NetWare long */
   unsigned char hilo_byte;    /* hibyte first (inverse native PC) */
   unsigned char lohi_byte;
   unsigned char lolo_byte;
   } nw_long;


/*************************************/
/* All MESSAGE function declarations */
/*************************************/

int GetBroadcastMessage(char *message);
int GetBroadcastMode(void);
int SendBroadcastMessage( byte connectionCount,byte *connections,
                    char *message , byte *resultCodes);
int SetBroadcastMode(int messageMode);


/*************************************/
/* All CONNECT function declarations */
/*************************************/

int GetConnectionInformation(word connectNo,char *objectName,
     word *objectType,long *objectID,byte *loginTime);
int GetConnectionNumber(void);
int GetObjectConnectionNumbers(word ObjType,char *ObjName,
     word *ConnectionCount,char *Connections);
```

```
int             result,bmode,rsmode,rgetbm,rsendbm;
int             respond,c,i,j;
byte            resultCodes;
word            ObjType;
byte            ConnectionCount;
byte            Connections[100];
unsigned char   length;
unsigned char   rmessage[55];
unsigned char   smessage[55];
unsigned long   n;
unsigned char   number;
long            fsize;

union REGS rin;
union REGS rout;

FILE *file;
char far *video;

void  main(int argc,char *argv[])
     {

      clrscr();

      if( argc != 3 )
       {
         printf("Usage : %s username <inputfile>\n",argv[0]);
         exit(1);
       }

       ObjType = USER;

      /* get detail about another user */

      result = GetObjectConnectionNumbers(ObjType,argv[1],
             (word *)&ConnectionCount,(byte *)&Connections);

     /*  int GetObjectConnectionNumbers(word ObjType,char *ObjName,
       word *ConnectionCount,char *Connections);  */


      printf("Result code from getobjectconnectionnumbers = %
             X\n",result);
      printf("Object Type = %d\n",ObjType);
      printf("Number of Connections = %d\n",ConnectionCount);
      printf("Connection number of %s = %d\n",argv[1],Connections[0]);

      result =  GetConnectionNumber();
      printf("Your Connection no. is %d\n",result);

      bmode =  GetBroadcastMode();
      printf("Broadcastmode is %d\n",bmode);



      /* Set broadcastmode to mode 3 */

      rsmode = SetBroadcastMode(3);
      bmode =  GetBroadcastMode();

      printf("Broadcastmode is %d and rsmode is %d \n",bmode,rsmode);

      printf("Request for sending %s\n",argv[2]);
```

```
/* sending acknowledge to the receiver */

rsendbm = SendBroadcastMessage((byte)ConnectionCount,
             (byte *)&Connections,"begin",&resultCodes);

printf("Waiting for respond...\n");

/* Waiting for the first responding from the reciever */

respond = 1;

while(respond != 0)
 {
 result =  GetBroadcastMessage(rmessage);
 respond = strcmp(rmessage,"OK");

 if(bioskey(1))
  {
    printf("Quit...");
    exit(1);
  }
 }


   /* open file for sending */

 if( (file = fopen(argv[2],"rb")) == NULL )
 {
  printf("Error openning %s",argv[2]);
  exit(1);
 }


 /* copy filename into smessage for sending */

length = strlen(argv[2]);
smessage[0] = 1;
smessage[1] = length;

for(i=0;i<length;i++)
 smessage[i+2] = argv[2][i];

rsendbm = SendBroadcastMessage((byte)ConnectionCount,
             (byte *)&Connections,smessage,&resultCodes);


 /* find size of image file */

fsize = filelength(fileno(file));


respond = 1;



/* check respond from the receiver */

while(respond != 0)
 {

 result =  GetBroadcastMessage(rmessage);
 respond = strcmp(rmessage,"OK");
```

```
 if(bioskey(1))
  {
    printf("Quit...");
    exit(1);
  }

 }

i = 0; n = 0;
number = 1;
length = 0;


/* Loop for getting imagefile to send */

while( n < fsize )
{
smessage[0] = (unsigned char)number;
c = fgetc(file);
length++;
n++;

if( c != 0 && i <= 50 && n < fsize)
{
 smessage[i+2] = (unsigned char)c;
 i++;
}

else
 {


  /*  Sending the number of data vary on length of message */

  smessage[1] = length;
  smessage[i+2] = (unsigned char)c;

  rsendbm = SendBroadcastMessage((byte)ConnectionCount,
            (byte *)&Connections,smessage,&resultCodes);

  i = 0;
  respond = 1;


  /* waiting respond from the receiver after send smessage */

  while(respond != 0)
  {
    result =  GetBroadcastMessage(rmessage);
    respond = strcmp(rmessage,"OK");


   if(bioskey(1))
    {
     printf("Quit...");
     exit(1);
    }
  }

  length = 0;
    number++;

  if ( number == 255 ) number = 1;
  gotoxy(1,11);
```

```
        /* show the bytes number of imagefile while sending */

        printf("sending %5u bytes",n);
      }
    }

    printf("\nSending sucessful !");
    printf("\007");

}
```

****

```
/*************************************************************************/
*                                                                       *
*            Receive Image (Gray Scale) : On Line                       *
*                                                                       *
/*************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <dir.h>
#include <io.h>
#include <string.h>

/*************************************/
/* Bindery object type classifications */
/*************************************/

#define   ESC           27

#define UNKNOWN         0x0000 /* unknown object */
#define USER            0x0001 /* User type object */

/****************/
/* All typedefs */
/****************/

typedef unsigned long    dword;      /* four bytes */
typedef unsigned int     word;       /* two bytes */
typedef unsigned char    byte;       /* single byte */

typedef struct {
   unsigned char hi_byte;       /* NetWare int, */
   unsigned char lo_byte;       /* hibyte first (inverse native PC) */
   } nw_int;

typedef struct {
   unsigned char hihi_byte;    /* NetWare long */
   unsigned char hilo_byte;    /* hibyte first (inverse native PC) */
   unsigned char lohi_byte;
   unsigned char lolo_byte;
   } nw_long;


/************************************/
/* All MESSAGE function declarations */
/************************************/

int GetBroadcastMessage(char *message);
int GetBroadcastMode(void);
int SendBroadcastMessage( byte connectionCount,byte *connections,
                    char *message , byte *resultCodes);
int SetBroadcastMode(int messageMode);

/************************************/
/* All CONNECT function declarations */
/************************************/

int GetConnectionNumber(void);
int GetObjectConnectionNumbers(word ObjType,char *ObjName,
     word *ConnectionCount,char *Connections);
```

```
int             result,bmode,rsmode,rgetbm,rsendbm;
int             chkheader,respond,c,i,j,row,col;
byte            resultCodes;
word            ObjType;
byte            ConnectionCount;
byte            Connections[100];
unsigned char   length;
unsigned char   rmessage[55];
unsigned char   smessage[55];
char            filename[20];
unsigned long   n;
unsigned char   number,num;


union REGS rin;
union REGS rout;

FILE        *imagefile;
char far    *video;


void  main(int argc,char *argv[])
     {

      clrscr();

      if( argc != 2 )
       {
         printf("Usage : %s <username>",argv[0]);
         exit(1);
       }

       ObjType = USER;

      /* get detail about another user */

      result = GetObjectConnectionNumbers(ObjType,argv[1],
             (word *)&ConnectionCount,(byte *)&Connections);

     /*  int GetObjectConnectionNumbers(word ObjType,char *ObjName,
       word *ConnectionCount,char *Connections);  */


      printf("Result code from getobjectconnectionnumbers = %
              X\n",result);
      printf("Object Type = %d\n",ObjType);
      printf("Number of Connections = %d\n",ConnectionCount);
      printf("Connection number of %s = %d\n",argv[1],Connections[0]);

      /* get detail about your connection */

      result =  GetConnectionNumber();
      printf("Your Connection no. is %d\n",result);
      bmode =  GetBroadcastMode();



      printf("Broadcastmode is %d\n",bmode);

      /* set broadcastmode to mode 3 */

      rsmode = SetBroadcastMode(3);
      bmode =  GetBroadcastMode();
```

```
   printf("Broadcastmode is %d and rsmode is %d \n",bmode,rsmode);

   printf("Waiting for recieving ");

   respond = 1;

   /* Waiting for first respond from the transmitter */

   while(respond != 0)
   {
    result = GetBroadcastMessage(rmessage);
    respond = strcmp(rmessage,"begin");

    /* Sending OK to the transmitter */

   if(respond == 0)
    {
   result = SendBroadcastMessage((byte)ConnectionCount,
             (byte *)&Connections,"OK",&resultCodes);
    }

   if(bioskey(1))
    {
    printf("\nquit...");
    exit(1);
    }

 }

 num = 1;
 number = 10;

 chkheader = 1;

 /* receive image filename from the transmitter */

 while( chkheader != 0 )
   {
     result = GetBroadcastMessage(rmessage);
     number = (unsigned char)rmessage[0];

     if(number != num)
       {
        chkheader = 1;
        }
       else chkheader = 0;

      if( bioskey(1) )
        {
          printf("Quit...");

          exit(1);
        }
   }

   length = rmessage[1];

   for(i=0;i<length;i++)
    rmessage[i] = rmessage[i+2];

   rmessage[length] = 0;
   strcpy(filename,rmessage);
   printf("%s\n",filename);
```

```
    result = SendBroadcastMessage((byte)ConnectionCount,
              (byte *)&Connections,"OK",&resultCodes);

    n = 0; number = 10;
    num = 1;
    length = 0;

    /* set the display to mode 13h for displaying graphic */

    rin.h.ah = 0;
    rin.h.al = 0x13;
    int86( 0x10, &rin, &rout );
    rin.h.ah = 0x10;
    rin.h.al = 0x10;
    for ( i = 0 ; i < 64 ; i++ ) {
     rin.h.dh = (unsigned char) i;
     rin.h.ch = (unsigned char) i;
     rin.h.cl = (unsigned char) i;
     rin.x.bx = i;
     int86( 0x10, &rin, &rout );

   }

  rin.h.ah = 0x10;
  rin.h.al = 0x1b;
  rin.x.cx = 256;
  rin.x.bx = 0;
  int86( 0x10, &rin, &rout );

  video = (char far *) 0xA0000000L;
  rin.h.ah = 0x0c;
  rin.h.bh = 0;

  while( n < 64000 )
   {
     chkheader = 1;

     /* loop for receive each message from the transmitter */

     while( chkheader != 0 )
       {
         result = GetBroadcastMessage(rmessage);

         number = (unsigned char)rmessage[0];


         /* check header between sending and receiving */

         if(number != num)
          {
          chkheader = 1;
          }
          else chkheader = 0;

         if( bioskey(1) )
          {
          printf("Quit...");
          exit(1);
         }

       }
```

```
        /* saving data to output file */

        length = (unsigned char)rmessage[1];

        for(j=2;j<length+2;j++)
         {
          c = rmessage[j];

          /* Displaying image */

          *video++ = (char) ( c >> 2 );
          n++;
         }

        num++ ;
        if (num == 255) num = 1;

        /* sending acknowledge to transmitter */

        result = SendBroadcastMessage((byte)ConnectionCount,
                (byte *)&Connections,"OK",&resultCodes);

    }

  printf("\007");
  getch();
  rin.h.ah = 0;
  rin.h.al = 3;
  int86( 0x10, &rin, &rout );

}
```

****

```
/*************************************************************************/
*                                                                       *
*                    Receive Image (Gray Scale)                         *
*                                                                       *
/*************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <dir.h>
#include <io.h>
#include <string.h>

/***************************************/
/* Bindery object type classifications */
/***************************************/

#define   ESC           27

#define UNKNOWN         0x0000 /* unknown object */
#define USER            0x0001 /* User type object */

/*****************/
/* All typedefs */
/*****************/

typedef unsigned long    dword;      /* four bytes */
typedef unsigned int     word;       /* two bytes */
typedef unsigned char    byte;       /* single byte */

typedef struct {
   unsigned char hi_byte;        /* NetWare int, */
   unsigned char lo_byte;        /* hibyte first (inverse native PC) */
   } nw_int;

typedef struct {
   unsigned char hihi_byte;    /* NetWare long */
   unsigned char hilo_byte;    /* hibyte first (inverse native PC) */
   unsigned char lohi_byte;
   unsigned char lolo_byte;
   } nw_long;


/*************************************/
/* All MESSAGE function declarations */
/*************************************/

int GetBroadcastMessage(char *message);
int GetBroadcastMode(void);
int SendBroadcastMessage( byte connectionCount,byte *connections,
                    char *message , byte *resultCodes);
int SetBroadcastMode(int messageMode);

/*************************************/
/* All CONNECT function declarations */
/*************************************/

int GetConnectionNumber(void);
int GetObjectConnectionNumbers(word ObjType,char *ObjName,
     word *ConnectionCount,char *Connections);
```

```
int             result,bmode,rsmode,rgetbm,rsendbm;
int             chkheader,respond,c,i,j,row,col;
byte            resultCodes;
word            ObjType;
byte            ConnectionCount;
byte            Connections[100];
unsigned char   length;
unsigned char   rmessage[55];
unsigned char   smessage[55];
char            filename[20];
unsigned long   n;
unsigned char   number,num;


union REGS rin;
union REGS rout;

FILE         *imagefile;
char far     *video;


void  main(int argc,char *argv[])
     {

      clrscr();

      if( argc != 2 )
       {
         printf("Usage : %s <username>",argv[0]);
         exit(1);
      }

       ObjType = USER;

       /* get detail about another user */

      result = GetObjectConnectionNumbers(ObjType,argv[1],
             (word *)&ConnectionCount,(byte *)&Connections);

     /*  int GetObjectConnectionNumbers(word ObjType,char *ObjName,
             word *ConnectionCount,char *Connections);  */


      printf("Result code from getobjectconnectionnumbers = %
              X\n",result);
      printf("Object Type = %d\n",ObjType);
      printf("Number of Connections =.%d\n",ConnectionCount);
      printf("Connection number of %s = %d\n",argv[1],Connections[0]);

      /* get detail about your connection */

      result =  GetConnectionNumber();
      printf("Your Connection no. is %d\n",result);

      bmode =  GetBroadcastMode();
      printf("Broadcastmode is %d\n",bmode);

      /* Set broadcastmode to mode 3 */

      rsmode = SetBroadcastMode(3);
      bmode =  GetBroadcastMode();
      printf("Broadcastmode is %d and rsmode is %d \n",bmode,rsmode);
```

```
int             result,bmode,rsmode,rgetbm,rsendbm;
int             chkheader,respond,c,i,j,row,col;
byte            resultCodes;
word            ObjType;
byte            ConnectionCount;
byte            Connections[100];
unsigned char   length;
unsigned char   rmessage[55];
unsigned char   smessage[55];
char            filename[20];
unsigned long   n;
unsigned char   number,num;


union REGS rin;
union REGS rout;

FILE        *imagefile;
char far    *video;


void  main(int argc,char *argv[])
     {

      clrscr();

      if( argc != 2 )
       {
         printf("Usage : %s <username>",argv[0]);
         exit(1);
      }

       ObjType = USER;

       /* get detail about another user */

      result = GetObjectConnectionNumbers(ObjType,argv[1],
             (word *)&ConnectionCount,(byte *)&Connections);

     /*  int GetObjectConnectionNumbers(word ObjType,char *ObjName,
             word *ConnectionCount,char *Connections);  */


      printf("Result code from getobjectconnectionnumbers = %
              X\n",result);
      printf("Object Type = %d\n",ObjType);
      printf("Number of Connections = %d\n",ConnectionCount);
      printf("Connection number of %s = %d\n",argv[1],Connections[0]);

      /* get detail about your connection */

      result =  GetConnectionNumber();
      printf("Your Connection no. is %d\n",result);

      bmode =  GetBroadcastMode();
      printf("Broadcastmode is %d\n",bmode);

      /* Set broadcastmode to mode 3 */

      rsmode = SetBroadcastMode(3);
      bmode =  GetBroadcastMode();
      printf("Broadcastmode is %d and rsmode is %d \n",bmode,rsmode);
```

```
/* Waiting for the first responding from the transmitter */

printf("Waiting for recieving ");

respond = 1;
while(respond != 0)
 {
 result = GetBroadcastMessage(rmessage);
 respond = strcmp(rmessage,"begin");

 /* Sending OK to the transmitter */

 if(respond == 0)
  {
   result = SendBroadcastMessage((byte)ConnectionCount,
          (byte *)&Connections,"OK",&resultCodes);
  }

 if(bioskey(1))
  {
   printf("\nquit...");
   exit(1);
   }

}

num = 1;

number = 10;

chkheader = 1;

/* receive image filename from the transmitter */

while( chkheader != 0 )
 {
   result = GetBroadcastMessage(rmessage);

   number = (unsigned char)rmessage[0];

   if(number != num)
    {
     chkheader = 1;
     }

    else chkheader = 0;

   if( bioskey(1) )

     {
       printf("Quit...");
       exit(1);
     }
   }

length = rmessage[1];

for(i=0;i<length;i++)
rmessage[i] = rmessage[i+2];

rmessage[length] = 0;
strcpy(filename,rmessage);
printf("%s\n",filename);
```

```
result = SendBroadcastMessage((byte)ConnectionCount,
          (byte *)&Connections,"OK",&resultCodes);

/* open file for saving */

if( (imagefile = fopen(filename,"wb")) == NULL )
 {
  printf("Error openning %s for saving...",filename);
  exit(1);
 }

 n = 0; number = 10;
 num = 1;
 length = 0;

while( n < 64000 )
 {
   chkheader = 1;
   while( chkheader != 0 )
    {
      result = GetBroadcastMessage(rmessage);
      number = (unsigned char)rmessage[0];
      if(number != num)
       {
       chkheader = 1;
       }
       else chkheader = 0;

      if( bioskey(1) )
       {
       fclose(imagefile);
       printf("Quit...");
       exit(1);
      }

    }


   /* find length of message */

   length = (unsigned char)rmessage[1];



   /* saving data vary on length of message */

   for(j=2;j<length+2;j++)
    {
      c = rmessage[j];
      fputc(c,imagefile);
      n++;
     }

   num++ ;
   if (num == 255) num = 1;


   /* sending acknowledge to the transmitter */

   result = SendBroadcastMessage((byte)ConnectionCount,
          (byte *)&Connections,"OK",&resultCodes);
```

```
   /* Show the number of data was received from the transmitter */

         gotoxy(1,10);
         printf("Recieving %6lu bytes",n);

       }

     fclose(imagefile);
     printf("\nReceiving sucessful !");
     printf("\007");

     /* Displaying image */

     view();

}

view()
{

    printf("\nViewing...");
    delay(2000);

    /* Open file for reading for displaying */

    if( (imagefile = fopen(filename,"rb")) == NULL )
     {
       printf("Error openning %s for displaying",filename);
       exit(1);
      }


    /*  Set the display to mode 13h for displaying image */

    rin.h.ah = 0;
    rin.h.al = 0x13;
    int86( 0x10, &rin, &rout );
    rin.h.ah = 0x10;
    rin.h.al = 0x10;


    for ( i = 0 ; i < 64 ; i++ ) {
      rin.h.dh = (unsigned char) i;
      rin.h.ch = (unsigned char) i;
      rin.h.cl = (unsigned char) i;
      rin.x.bx = i;
      int86( 0x10, &rin, &rout );

    }

    rin.h.ah = 0x10;
    rin.h.al = 0x1b;
    rin.x.cx = 256;
    rin.x.bx = 0;
    int86( 0x10, &rin, &rout );

    video = (char far *) 0xA0000000L;
    rin.h.ah = 0x0c;
    rin.h.bh = 0;

    row = 0;
    col = 0;
```

```
    for ( row = 0 ; row < 200 ; row++ )
       for ( col = 0 ; col < 320 ; col++ )
          *video++ = (char) ( getc( imagefile) >> 2 );

    getch();
    rin.h.ah = 0;
    rin.h.al = 3;
    int86( 0x10, &rin, &rout );

}
```

****

```
/*************************************************************************/
*                                                                       *
*        Send Color Image By Using Broadcast Messages                   *
*                                                                       *
/*************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <dir.h>
#include <io.h>

/*************************************/
/* Bindery object type classifications */
/*************************************/

#define   ESC           27

#define UNKNOWN         0x0000 /* unknown object */
#define USER            0x0001 /* User type object */

/****************/
/* All typedefs */
/****************/

typedef unsigned long    dword;     /* four bytes */
typedef unsigned int     word;      /* two bytes */
typedef unsigned char    byte;      /* single byte */

typedef struct {
   unsigned char hi_byte;       /* NetWare int, */
   unsigned char lo_byte;       /* hibyte first (inverse native PC) */
   } nw_int;

typedef struct {
   unsigned char hihi_byte;    /* NetWare long */
   unsigned char hilo_byte;    /* hibyte first (inverse native PC) */
   unsigned char lohi_byte;
   unsigned char lolo_byte;
   } nw_long;


/************************************/
/* All MESSAGE function declarations */
/************************************/

int GetBroadcastMessage(char *message);
int GetBroadcastMode(void);
int SendBroadcastMessage( byte connectionCount,byte *connections,
                    char *message , byte *resultCodes);
int SetBroadcastMode(int messageMode);

/************************************/
/* All CONNECT function declarations */
/************************************/

int GetConnectionInformation(word connectNo,char *objectName,
     word *objectType,long *objectID,byte *loginTime);
int GetConnectionNumber(void);
int GetObjectConnectionNumbers(word ObjType,char *ObjName,
     word *ConnectionCount,char *Connections);
```

```
long            fsize;
int             result,bmode,rsmode,rgetbm,rsendbm;
int             respond,c,i,j;
byte            resultCodes;
word            ObjType;
byte            ConnectionCount;
byte            Connections[100];
unsigned char   length;
unsigned char   rmessage[50];
unsigned char   smessage[50];
unsigned long   n;
unsigned char   number;
char            *temp;
int             dec,sign;

union REGS rin;
union REGS rout;
FILE *file;
char far *video;

void  main(int argc,char *argv[])
     {

      clrscr();

      if( argc != 3 )
       {
         printf("Usage : %s username <inputfile>\n",argv[0]);
         exit(1);
      }

       ObjType = USER;

      /* get detail about another user */

      result = GetObjectConnectionNumbers(ObjType,argv[1],
             (word *)&ConnectionCount,(byte *)&Connections);

     /*  int GetObjectConnectionNumbers(word ObjType,char *ObjName,
             word *ConnectionCount,char *Connections);  */


      printf("Result code from getobjectconnectionnumbers = %
              X\n",result);
      printf("Object Type = %d\n",ObjType);
      printf("Number of Connections = %d\n",ConnectionCount);
      printf("Connection number of %s = %d\n",argv[1],Connections[0]);


      result =  GetConnectionNumber();
      printf("Your Connection no. is %d\n",result);


      bmode =  GetBroadcastMode();
      printf("Broadcastmode is %d\n",bmode);

      /* set broadcastmode to mode 3 */

      rsmode = SetBroadcastMode(3);
      bmode =  GetBroadcastMode();
      printf("Broadcastmode is %d and rsmode is %d \n",bmode,rsmode);

      printf("Request for sending %s\n",argv[2]);
```

```
/* sending acknowledge to the receiver */

rsendbm = SendBroadcastMessage((byte)ConnectionCount,
            (byte *)&Connections,"begin",&resultCodes);

printf("Waiting for respond...\n");

/* Waiting for the first responding from the reciever */

respond = 1;

while(respond != 0)
 {
 result =  GetBroadcastMessage(rmessage);
 respond = strcmp(rmessage,"OK");

 if(bioskey(1))
  {
    printf("Quit...");
    exit(1);
  }
 }

/* open input file for sending */

if( (file = fopen(argv[2],"rb")) == NULL )
 {
 printf("Error openning %s",argv[2]);
 exit(1);
 }

/* copy filename into smessage */

length = strlen(argv[2]);
smessage[0] = 1;
smessage[1] = length;

for(i=0;i<length;i++)
 smessage[i+2] = argv[2][i];

/* send filename to the receiver */

rsendbm = SendBroadcastMessage((byte)ConnectionCount,
            (byte *)&Connections,smessage,&resultCodes);

/* find size of imagefile */

fsize = filelength(fileno(file));

respond = 1;

/* wait respond from the receiver */

while(respond != 0)
 {
 result =  GetBroadcastMessage(rmessage);
 respond = strcmp(rmessage,"OK");

 if(bioskey(1))
  {
    printf("Quit...");
    exit(1);
  }
 }
```

```
    i = 0; n = 0;
    number = 1;
    length = 0;

    /* Loop for getting imagefile to send */

    while(!feof(file) && n < fsize)
    {

    smessage[0] = (unsigned char)number;
    c = fgetc(file);
    length++;
    n++;

    if( c != 0 && i <= 39 && n < fsize )
    {
     smessage[i+2] = (unsigned char)c;
     i++;
    }
    else
     {

      /* Sending the number of data vary on length of smessage */

      smessage[1] = length;
      smessage[i+2] = (unsigned char)c;

      rsendbm = SendBroadcastMessage((byte)ConnectionCount,
                (byte *)&Connections,smessage,&resultCodes);

      i = 0;

      respond = 1;


/* waiting respond from the receiver after sending each smessage */

      while(respond != 0 && n != fsize )
      {
        result =  GetBroadcastMessage(rmessage);
        respond = strcmp(rmessage,"OK");

       if(bioskey(1))
        {
         printf("Quit...");
         exit(1);
        }
      }

      length = 0;
        number++;

      if ( number == 255 ) number = 1;
      gotoxy(1,11);

      /* show the bytes number of imagefile while sending */

      printf("Sending %5lu bytes ",n);

    }
  }
```

```
   /* copy end into smessage for sending */

   strcpy(temp,"end");
   smessage[0] = number;
   smessage[1] = 3;

   for(i=0;i<3;i++)
    smessage[i+2] = temp[i];
    smessage[5] = 0;

   /* sending "end" to tell end of imagefile to the receiver */

   rsendbm = SendBroadcastMessage((byte)ConnectionCount,
                  (byte *)&Connections,smessage,&resultCodes);
   printf("\nSending sucessful !");
   printf("\007");

}
```

```
/*************************************************************************/
*                                                                       *
*                Receive Color Image (pcx format)                       *
*                                                                       *
/*************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <dir.h>
#include <io.h>
#include <string.h>

#include "vesa.c"

/*************************************/
/* Bindery object type classifications */
/*************************************/

#define   ESC           27

#define UNKNOWN         0x0000 /* unknown object */
#define USER            0x0001 /* User type object */

/****************/
/* All typedefs */
/****************/

typedef unsigned long    dword;      /* four bytes */
typedef unsigned int     word;       /* two bytes */
typedef unsigned char    byte;       /* single byte */

typedef struct {
   unsigned char hi_byte;       /* NetWare int, */
   unsigned char lo_byte;       /* hibyte first (inverse native PC) */
   } nw_int;

typedef struct {
   unsigned char hihi_byte;    /* NetWare long */
   unsigned char hilo_byte;    /* hibyte first (inverse native PC) */
   unsigned char lohi_byte;
   unsigned char lolo_byte;
   } nw_long;


/************************************/
/* All MESSAGE function declarations */
/************************************/

int GetBroadcastMessage(char *message);
int GetBroadcastMode(void);
int SendBroadcastMessage( byte connectionCount,byte *connections,
                    char *message , byte *resultCodes);
int SetBroadcastMode(int messageMode);

/************************************/
/* All CONNECT function declarations */
/************************************/

int GetConnectionNumber(void);
int GetObjectConnectionNumbers(word ObjType,char *ObjName,
     word *ConnectionCount,char *Connections);
```

```
int             result,bmode,rsmode,rgetbm,rsendbm;
int             chkheader,respond,c,i,j,row,col,length;
byte            resultCodes;
word            ObjType;
byte            ConnectionCount;
byte            Connections[100];
unsigned char   rmessage[50];
unsigned char   smessage[50];
unsigned long   count;
unsigned char   number,num;

union REGS rin;
union REGS rout;

FILE *file;
char far *video;


unsigned char  picture[3][3000];
int            _24bit;
char           imagename[20];
char           party[20];
int  picx,picy;

FILE          *imagefile;
unsigned int  width,depth;
unsigned int  bytes;
int           n,c,ij;
char          _palette[768];
int           x,xoff,yoff;
int           mode = 0x101;
int           end;
char          temp[5];
unsigned char      filename[20];

int      receive();
void     view();

typedef       struct {
      char manufacturer;
      char version;
      char encoding;
      char bits_per_pixel;
      int  xmin,ymin;
      int  xmax,ymax;
      int  hres;
      int  vres;
      char __palette[48];
      char reserved;
      char color_planes;
            int  bytes_per_line;
            int  palette_type;
            char filler[58];
      } PCXHEAD;
PCXHEAD   header;



int chartohex(char ch);

void  main(int argc,char *argv[])
     {
      clrscr();
```

```
 if( argc != 2 )
  {
    printf("Usage : %s <username>",argv[0]);
    exit(1);
 }

  ObjType = USER;

 /* get detail about another user */

 result = GetObjectConnectionNumbers(ObjType,argv[1],
        (word *)&ConnectionCount,(byte *)&Connections);

/*  int GetObjectConnectionNumbers(word ObjType,char *ObjName,
        word *ConnectionCount,char *Connections);  */


 printf("Result code from getobjectconnectionnumbers = %
         X\n",result);
 printf("Object Type = %d\n",ObjType);
 printf("Number of Connections = %d\n",ConnectionCount);
 printf("Connection number of %s = %d\n",argv[1],Connections[0]);


 /* get detail about your connection */

 result =  GetConnectionNumber();
 printf("Your Connection no. is %d\n",result);

 bmode =  GetBroadcastMode();
 printf("Broadcastmode is %d\n",bmode);

 /* set broadcastmode to mode 3 */

 rsmode = SetBroadcastMode(3);
 bmode =  GetBroadcastMode();
 printf("Broadcastmode is %d and rsmode is %d \n",bmode,rsmode);

 printf("Waiting for recieving ");
 strcpy(rmessage,"wait");

 /* waiting for first respond from the transmitter */

respond = 1;

while(respond != 0)
 {
 result = GetBroadcastMessage(rmessage);
 respond = strcmp(rmessage,"begin");



 /* sending OK to the transmitter */

 if(respond == 0)
  {

   result = SendBroadcastMessage((byte)ConnectionCount,
          (byte *)&Connections,"OK",&resultCodes);
  }

 if(bioskey(1))
  {
```

```
    printf("\nquit...");
    exit(1);
    }

}

num = 1;
number = 10;

chkheader = 1;

/* receive image filename from the transmitter */

while( chkheader != 0 )
    {
     result = GetBroadcastMessage(rmessage);
     number = (unsigned char)rmessage[0];

     if(number != num)
     {
      chkheader = 1;
     }
     else  chkheader = 0;

     if( bioskey(1) )
       {
         printf("Quit...");
         exit(1);
       }
    }

    length = rmessage[1];

    for(i=0;i<length;i++)
    rmessage[i] = rmessage[i+2];
    rmessage[length] = 0;
    strcpy(filename,rmessage);
    printf("%s\n",filename);

    result = SendBroadcastMessage((byte)ConnectionCount,
             (byte *)&Connections,"OK",&resultCodes);

    /* open file for saving */

    if( (imagefile = fopen(filename,"wb")) == NULL )
   {
     printf("Error openning %s for saving...",filename);
     exit(1);
    }

n = 0; number = 10;
num = 1;
length = 0;
end = 0;

/* Loop for receive each message from the transmitter */

while( end != 1 )
   {
     chkheader = 1;
     while( chkheader != 0 )
       {
         result = GetBroadcastMessage(rmessage);
         number = (unsigned char)rmessage[0];
```

```
        /* check header between sending and receiving */

        if(number != num)
        {
        chkheader = 1;
        }
        else chkheader = 0;

        for(i=0;i<3;i++)
       {
        temp[i] = rmessage[i+2];
       }

       if( bioskey(1) )
        {
        fclose(imagefile);
        printf("Quit...");
        exit(1);
       }
     }

     /* check end of imagefile */

      if( strcmp(temp,"end") == 0 )
        {
        end = 1;
        }
        else
        end = 0;

   /* saving data to output file */

   if( end != 1 )
   {

   length = (unsigned char)rmessage[1];


   for(j=2;j<length+2;j++)
    {
     c = rmessage[j];
     fputc(c,imagefile);
     count++;
    }

   num++ ;
   if (num == 255) num = 1;

   /* sending acknowledge to transmitter */

   result = SendBroadcastMessage((byte)ConnectionCount,
          (byte *)&Connections,"OK",&resultCodes);

   gotoxy(1,10);

   /* show the bytes number of imagefile while receiving */

   printf("Recieving %6lu bytes",count);
  }

}
```

```
   fclose(imagefile);
   printf("\nReceiving sucessful !");
   printf("\007");

   view();

  } .

  /* Displaying  PCX format image to the monitor */

void  view()
{

    clrscr();

    /* open VESA mode */

    if(!opengraph(mode))
      {
        printf("Error while opening VESA mode\n");
        exit(1);
      }

    if(getbitperpix() < 8)
      {
        closegraph();
        printf("This program runs on 256 colors or upper\n");
        exit(1);
      }

  if((imagefile = fopen(filename,"rb"))!=NULL)
  {

    if(fread((char *)&header,1,sizeof(PCXHEAD),imagefile)
       == sizeof(PCXHEAD))

    {
      if(header.bits_per_pixel == 8 && header.color_planes == 3)
      _24bit = 1; else
        {
          if(!fseek(imagefile,-769L,SEEK_END))
          {
            if(fgetc(imagefile) == 0x0c &&
                  fread(_palette,1,768,imagefile) == 768)
             {
               _24bit = 0;

               for(x= 0;x<768;x+=3)
                 setdac((int)(x/3),(unsigned char)_palette[x]>>2,
                                   (unsigned char)_palette[x+1]>>2,
                                   (unsigned char)_palette[x+2]>>2);
               fseek(imagefile,128L,SEEK_SET);
             }
            else
             { picx = picy = depth = i = 0; }
          }
        }


      bytes = header.bytes_per_line;
      width = header.xmax-header.xmin+1;
      depth = header.ymax-header.ymin+1;
```

```
if((_24bit)^(getbitperpix()!=8))
{
  closegraph();
  printf("Please use 8bit PCX file with 256 color \n");
  printf("  ..or 24bit PCX file with true color \n");
  exit(1);
}

picx = width;
picy = depth;

if(picx%2) picx++;

for (i=0;i<depth;i++)
{
  if(_24bit)
    {
      for (ij=2;ij>=0;ij--)
      {
        n = 0;

        do{
          c = fgetc(imagefile) &0xff;

          if((c & 0xc0) == 0xc0)
           {
            j = c & 0x3f;
            c = fgetc(imagefile);

            while(j--)
            {
              picture[ij][n] = (unsigned char)c;
              n++;
            }
           }
          else
           {
            picture[ij][n] = (unsigned char)c;
            n++;
           }
        }while(n<bytes);
      }
    }
 else
    {
      n = 0;
      do{
      c = fgetc(imagefile) &0xff;

      if((c & 0xc0) == 0xc0)
        {
          j = c & 0x3f;
          c = fgetc(imagefile);
          while(j--)
            {
            picture[0][n] = (unsigned char)c;
            n++;
            }
        }
      else
        {
          picture[0][n] = (unsigned char)c;
          n++;
        }
```

```
          }while(n<bytes);
        }

      xoff = (getxres()-picx)>>1;
      yoff = (getyres()-picy)>>1;
      if (xoff<0)xoff = 0;
      if (yoff<0)yoff = 0;

      for(x = 0;x<(picx-1);x++)
       {
        if(_24bit)
          {
           directpixel(x+xoff,i+yoff,(unsigned int)picture[2][x],
                                     (unsigned int)picture[1][x],
                                     (unsigned int)picture[0][x]);
          }
        else
          {
           putpixel(x+xoff,i+yoff,(unsigned int)picture[0][x]);
        }
     }
     }
   }else

   {
     closegraph();
     printf("Error reading file %s",filename);
     exit(1);
   }

 fclose(imagefile);

 }else
 {
   closegraph();
   printf("Error opening file %s",filename);
   exit(1);
 }

 printf("\007");
 getch();
 closegraph();

}
```

****

```
/*************************************************************************/
*                                                                       *
*                        Receive Color Image                            *
*                                                                       *
/*************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <dir.h>
#include <io.h>
#include <string.h>

#include "vesa.c"

/*************************************/
/* Bindery object type classifications */
/*************************************/

#define   ESC           27

#define UNKNOWN         0x0000 /* unknown object */
#define USER            0x0001 /* User type object */

/****************/
/* All typedefs */
/****************/

typedef unsigned long    dword;      /* four bytes */
typedef unsigned int     word;       /* two bytes */
typedef unsigned char    byte;       /* single byte */

typedef struct {
   unsigned char hi_byte;        /* NetWare int, */
   unsigned char lo_byte;        /* hibyte first (inverse native PC) */
   } nw_int;

typedef struct {
   unsigned char hihi_byte;    /* NetWare long */
   unsigned char hilo_byte;    /* hibyte first (inverse native PC) */
   unsigned char lohi_byte;
   unsigned char lolo_byte;
   } nw_long;


/************************************/
/* All MESSAGE function declarations */
/************************************/

int GetBroadcastMessage(char *message);
int GetBroadcastMode(void);
int SendBroadcastMessage( byte connectionCount,byte *connections,
                    char *message , byte *resultCodes);
int SetBroadcastMode(int messageMode);

/************************************/
/* All CONNECT function declarations */
/************************************/

int GetConnectionNumber(void);
int GetObjectConnectionNumbers(word ObjType,char *ObjName,
    word *ConnectionCount,char *Connections);
```

```
int             result,bmode,rsmode,rgetbm,rsendbm;
int             chkheader,respond,c,i,j,row,col,length;
byte            resultCodes;
word            ObjType;
byte            ConnectionCount;
byte            Connections[100];
unsigned char   rmessage[50];
unsigned char   smessage[50];
unsigned long   count;
unsigned char   number,num;

union REGS rin;
union REGS rout;

FILE *file;
char far *video;

char                imagename[20];
char                party[20];
FILE                *imagefile;
unsigned int        bytes;
int                 n,c,ij;
int                 end;
char                temp[5];
char                *extension;
unsigned char       filename[20];

int     receive();
void    viewpcx();
void    viewgif();
void    viewbmp();
void    viewtif();
void    viewwpg();
void    viewimg();
void    vpcx();


int chartohex(char ch);

void  main(int argc,char *argv[])
     {

      clrscr();

      if( argc != 2 )
       {
         printf("Usage : %s <username>",argv[0]);
         exit(1);
       }

       ObjType = USER;



      /* get detail about another user */

      result = GetObjectConnectionNumbers(ObjType,argv[1],
             (word *)&ConnectionCount,(byte *)&Connections);

     /*  int GetObjectConnectionNumbers(word ObjType,char *ObjName,
       word *ConnectionCount,char *Connections);  */
```

```
  printf("Result code from getobjectconnectionnumbers = %
         X\n",result);
  printf("Object Type = %d\n",ObjType);
  printf("Number of Connections = %d\n",ConnectionCount);
  printf("Connection number of %s = %d\n",argv[1],Connections[0]);

  /* get detail about your connection */

  result =  GetConnectionNumber();
  printf("Your Connection no. is %d\n",result);
  bmode =  GetBroadcastMode();
  printf("Broadcastmode is %d\n",bmode);

  /* set broadcast mode to mode 3 */

  rsmode = SetBroadcastMode(3);
  bmode =  GetBroadcastMode();
  printf("Broadcastmode is %d and rsmode is %d \n",bmode,rsmode);

  printf("Waiting for recieving ");
  strcpy(rmessage,"wait");

  /* waiting for first respond from the transmitter */

 respond = 1;

 while(respond != 0)
  {
  result = GetBroadcastMessage(rmessage);
  respond = strcmp(rmessage,"begin");

  /* sending OK to the transmitter */

  if(respond == 0)
   {
  result = SendBroadcastMessage((byte)ConnectionCount,
           .(byte *)&Connections,"OK",&resultCodes);
   }

  if(bioskey(1))
   {
   printf("\nquit...");
   exit(1);
   }

}


num = 1;
number = 10;

chkheader = 1;

/* receive image filename from the transmitter */

while( chkheader != 0 )
   {
    result = GetBroadcastMessage(rmessage);
    number = (unsigned char)rmessage[0];

    if(number != num)
    {
     chkheader = 1;
    }
```

```
    else  chkheader = 0;

    if( bioskey(1) )
      {
        fclose(imagefile);
        printf("Quit...");
        exit(1);
      }
   }

   length = rmessage[1];

   for(i=0;i<length;i++)
   rmessage[i] = rmessage[i+2];
   rmessage[length] = 0;
   strcpy(filename,rmessage);
   printf("%s\n",filename);

   result = SendBroadcastMessage((byte)ConnectionCount,
            (byte *)&Connections,"OK",&resultCodes);

   if( (imagefile = fopen(filename,"wb")) == NULL )
  {
    printf("Error openning %s for saving...",filename);
    exit(1);
   }

n = 0; number = 10;
num = 1;
length = 0;
end = 0;

while( end != 1 )
  {
    chkheader = 1;

    while( chkheader != 0 )
      {
         result = GetBroadcastMessage(rmessage);
         number = (unsigned char)rmessage[0];


         /* check header between sending and receiving */

         if(number != num)
         {
         chkheader = 1;
         }
         else chkheader = 0;

         for(i=0;i<3;i++)
        {
         temp[i] = rmessage[i+2];
        }

        if( bioskey(1) )
         {
         fclose(imagefile);
         printf("Quit...");
         exit(1);
        }
      }
```

```
        /* check end of imagefile */

          if( strcmp(temp,"end") == 0 )
            {
            end = 1;
            }
            else
            end = 0;


        if( end != 1 )
        {

        length = (unsigned char)rmessage[1];

        /* saving data vary on length of message */

        for(j=2;j<length+2;j++)
         {
          c = rmessage[j];
          fputc(c,imagefile);
          count++;
         }

        num++ ;

        if (num == 255) num = 1;

        /* sending acknowledge to transmitter */

        result = SendBroadcastMessage((byte)ConnectionCount,
               (byte *)&Connections,"OK",&resultCodes);

        gotoxy(1,10);


    /* show the number of data was received from the transmitter */

          printf("Recieving %6lu bytes",count);
       }
      }

fclose(imagefile);
printf("\nReceiving sucessful !");
printf("\007");


/* open file for displaying */

 strlwr(filename);

 extension = strstr(filename,".");
 extension++;

 if( strcmp(extension,"pcx") == 0 )
   {
     viewpcx();
   }

 if( strcmp(extension,"gif") == 0 )
   {
     viewgif();
   }
```

```
    if( strcmp(extension,"bmp") == 0 )
     {
       viewbmp();
     }

     if( strcmp(extension,"tif") == 0 )
     {
       viewtif();
     }

   if( strcmp(extension,"wpg") == 0 )
     {
       viewwpg();
     }

    if( strcmp(extension,"img") == 0 )
     {
       viewimg();
     }

 }

void  viewgif()
 {
    char    com[50];

    clrscr();
    strcpy(com,"vgif ");


    strcat(com,filename);
    strcat(com," ");
    strcat(com,"-v13");
    system(com);

 }

void  viewbmp()
 {
    char    com[50];

    clrscr();
    strcpy(com,"vbmp ");
    strcat(com,filename);
    strcat(com," ");
    strcat(com,"-v13");
    system(com);

 }

void  viewtif()
 {
    char    com[50];

    clrscr();
    strcpy(com,"vtif ");
    strcat(com,filename);
    strcat(com," ");
    strcat(com,"-v13");
    system(com);

 }
```

```
void  viewwpg()
 {
    char    com[50];

    clrscr();
    strcpy(com,"vwpg ");
    strcat(com,filename);
    strcat(com," ");
    strcat(com,"-v13");
    system(com);

 }

void  viewimg()
 {
    char    com[50];

    clrscr();
    strcpy(com,"vimg ");
    strcat(com,filename);
    strcat(com," ");
    strcat(com,"-v13");
    system(com);

 }

void  viewpcx()
 {
    char    com[50];

    clrscr();
    strcpy(com,"vpcx ");
    strcat(com,filename);
    strcat(com," ");
    strcat(com,"-v13");
    system(com);

 }
```

****

```
/*************************************************************************/
*                                                                       *
*       Compare Error Between Transmitter File and Receiver File        *
*                                                                       *
/*************************************************************************/

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <io.h>

FILE    *file1, *file2;
FILE    *output;
int     c1,c2,diff;
int     x,y,i,ch;
char    tranname[20],recname[20];
long    count,number;
long    size1,size2;


main(int argc,char *argv[])
 {

   clrscr();

   if( argc != 3 )
    {
      printf("Usage : %s <transmitter file> <receiver file>",argv[0]);
      exit(1);
    }


      /* Open transmitter file */

   if( (file1 = fopen(argv[1],"rb")) == NULL )
    {
      printf("Can't open %s",argv[1]);
      exit(1);
    }


      /* Open receiver file */

   if( (file2 = fopen(argv[2],"rb")) == NULL )
    {
      printf("Can't open %s",argv[2]);
      exit(1);
    }

    strcpy(tranname,argv[1]);
    strcpy(recname,argv[2]);


      /* Set window to show image datas */

    Monitor1();
    window(2,4,78,23);
    number = 1; i = 1;
    count = 0;
    ch = 0;


      /* Loop get data until end of file */

    while( (c1=fgetc(file1)) != EOF && ch != 'q' )
```

```
{
  c2 = fgetc(file2);
  diff = c1 - c2;

  textcolor(GREEN);
  gotoxy(2,i);

  cprintf("%9lu",number);
  cprintf("%20X",c1);
  cprintf("%20X",c2);


  /* find different of data */

  if( diff == 0 )
   {
  textcolor(RED);
  cprintf("%20s","NO ERROR");
   }
  else
   {
  count++;
  cprintf("%20d",diff);
   }
  number++; i++;


  /* show datas each page */

  if(i % 19 == 0)
   {
  gotoxy(10,20);
  textcolor(LIGHTBLUE);

  cprintf("Press anykey to continue, q = quick analyze");
  cprintf(", ESC = exit");

  ch = getch();
  ch = tolower(ch);
  if( ch == 0 )
    ch = getch();


  /* press 'q' to quick analize */

  if( ch == 'q' )
   {
     Quick();
   }


  /* press ESC to quit program */

  if( ch == 27 )
   {
    window(1,1,80,25);
    textbackground(BLACK);
    textcolor(WHITE);
          clrscr();
    exit(1);
   }
  clrscr();
  i = 1;
  }
}
```

```
      /* show end of files */

    if( ch != 'q' ) getch();
    window(1,1,80,25);
    Monitor2();


      /* show all detail to analize */

    Conclusion();

    fclose(file1);
    fclose(file2);

}


      /* Function for writing window */

Monitor1()
 {

   int i,j;

   clrscr();
   textcolor(LIGHTCYAN);

   cputs("Ú");              /* Character 218 */
   for(i=2;i<=79;i++)
   {
      gotoxy(i,0);
      cputs("Ä");           /* Character 196 */
   }

   cputs("¿");              /* Character 191 */
   for(j=2;j<=24;j++)
    {
     gotoxy(1,j);
     cputs("³");           /* Character 179 */
     gotoxy(80,j);
       cputs("³");
    }

   gotoxy(1,24);
   cputs("À");              /* Character 192 */
   for(i=2;i<=79;i++)
   {
    gotoxy(i,24);
      cputs("Ä");           /* Character 196 */
   }

   gotoxy(80,24);
   cputs("Ù");

   gotoxy(2,2);
   textcolor(YELLOW);

   cprintf("%10s","Byte No.");
   cprintf("%20s","Transmitter(Hex)");
   cprintf("%20s","Receiver(Hex)");
   cprintf("%20s","Result(Dec)");
```

```
      /* write the second line */

   textcolor(LIGHTCYAN);
   for(i=2;i<=79;i++)
   {
    gotoxy(i,3);
      cputs("Ä");          /* Character 196 */
   }

 }


      /* set monitor */

Monitor2()
 {

   int i,j;
   clrscr();
   textcolor(LIGHTCYAN);

   cputs("Ú");           /* Character 218 */
   for(i=2;i<=79;i++)
   {
      gotoxy(i,0);
      cputs("Ä");          /* Character 196 */
   }

   cputs("¿");           /* Character 191 */
   for(j=2;j<=24;j++)
    {
     gotoxy(1,j);
     cputs("³");          /* Character 179 */
     gotoxy(80,j);
       cputs("³");
    }

   gotoxy(1,24);
   cputs("À");           /* Character 192 */
   for(i=2;i<=79;i++)
   {
    gotoxy(i,24);
      cputs("Ä");          /* Character 196 */
   }

   gotoxy(80,24);
   cputs("Ù");

 }


      /* calculate error */

Quick()
 {

    while( (c1=fgetc(file1)) != EOF )
    {
      c2 = fgetc(file2);
      diff = c1 - c2;
      if( diff != 0 )  count++;
    }

 }
```

```
      /* show  all detail that analize */

Conclusion()
{
    size1 = filelength(fileno(file1));
    size2 = filelength(fileno(file2));

    x = 15; y = 6;
    textcolor(YELLOW);

    gotoxy(x,y);
    cprintf("Compare errors between %s and %s",tranname,recname);

    gotoxy(x,y+2);
    cprintf("Transmitter filename is%13s",tranname);

    gotoxy(x,y+4);
    cprintf("Receiver filename is%13s",recname);

    gotoxy(x,y+6);
    cprintf("Transmitter filesize  = %10lu bytes",size1);

    gotoxy(x,y+8);
    cprintf("Receiver filesize     = %10lu bytes",size2);

    gotoxy(x,y+10);
    cprintf("Number of byte error  = %10lu bytes",count);

    gotoxy(x,y+12);
    if( count == 0 )
     {
      cprintf("Transmitter file isn't different from the receiver
               file.");
     }

    else
     {
      cprintf("Transmitter file is different from Receiver file.");
     }

    getch();
    textcolor(WHITE);
    textbackground(BLACK);
    clrscr();
}
```

****

กิตติกรรมประกาศ

โครงงานนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากการให้คำปรึกษา ข้อแนะนำ และความช่วยเหลือจาก ผศ.ดร.สุวิพล สิทธิชีวภาค , อาจารย์นภัทร สระเอี่ยม , นายอดิรุตน์ เยี่ยมเจริญ 4A และนายวุฒิเจตน์ อร่ามดิลกรัตน์ 4D

ทางผู้จัดทำจึงขอขอบพระคุณไว้ ณ ที่นี้ด้วย

ผู้จัดทำ

เอกสารอ้างอิง

[1.] ธันวา ศรีประมง , "การเขียนโปรแกรมภาษาซีทางวิศวกรรม", : มหาวิทยาลัยเทคโนโลยีมหานคร 1994.

[2.] อัครเสน สมุทรผ่อง , "ระบบเครือข่ายคอมพิวเตอร์ LAN และการใช้งาน Novell Netware" , : สำนักพิมพ์ บริษัทซีเอ็ดยูเคชันจำกัด 2535.

[3.] ต้น ตันฑ์สุทธิวงศ์ , "คู่มือใช้งาน Netware ฉบับสมบูรณ์ เวอร์ชัน 3.12 และ 4.x", : สำนักพิมพ์ บริษัทโปรวิชันจำกัด 2537.

[4.] Charles G. Rose , "Programmer's Guide to Netware", McGraw-Hill,Inc.,1990.

[5.] Barry Nance, "Network Programming in C", Que corporation.