โปรแกรมควบคุมแสดงหน้าจอของเครื่องลูกข่ายในระบบเครือข่ายท้องถิ่น

SCREEN BROADCASTING & CONTROL PROGRAM WITHIN LAN

โดย

นายฉลาด สาตะถา

นายบุญเลิศ ชนะอินทร์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาตรบัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

หัวข้อปริญญานิพนธ์	โปรแกรมควบคุมแสดงหน้าจอของเครื่องลูกข่ายในระบบเครือข่ายท้องถิ่น	
TITLE	**SCREEN BROADCASTING & CONTROL PROGRAM WITHIN LAN**	
โดย	นายฉลาด สาตะถา	43015719
	นายบุญเลิศ ชนะอินทร์	43015729
อาจารย์ผู้ควบคุมปริญญานิพนธ์	อาจารย์มยุรี เลิศเวชกุล	
ภาควิชา	วิศวกรรมสารสนเทศ	
ปีการศึกษา	2544	

ปริญญานิพนธ์ฉบับนี้ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโนยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

(อาจารย์ มยุรี เลิศเวชกุล)
อาจารย์ผู้ควบคุมปริญญานิพนธ์

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อวิทยานิพนธ์	โปรแกรมควบคุมแสดงหน้าจอของเครื่องลูกข่าย ในระบบเครือข่ายท้องถิ่น	
นักศึกษา	นายฉลาด สาตะถา	รหัสประจำตัว 43015719
	นายบุญเลิศ ชนะอินทร์	รหัสประจำตัว 43015729
อาจารย์ผู้ควบคุมวิทยานิพนธ์	อาจารย์ มยุรี เลิศเวชกุล	
ระดับการศึกษา	ปริญญาอุตสาหกรรมศาสตรบัณฑิต	
ภาควิชา	วิศวกรรมสารสนเทศ	
ปีการศึกษา	2544	

บทคัดย่อ

โปรแกรมควบคุมการแสดงหน้าจอของเครื่องลูกข่ายระบบเครือข่ายท้องถิ่น ได้มีแนวคิดมาจากการศึกษาในห้องปฏิบัติการคอมพิวเตอร์ ซึ่งจะเห็นว่าการใช้คอมพิวเตอร์เป็นสื่อในการสอนเป็นสิ่งจำเป็นมากขึ้น เช่น การใช้คอมพิวเตอร์ร่วมกับโปรเจคเตอร์ เป็นต้น แต่จะเห็นว่า ประสิทธิภาพจะลดลงเมื่อมีจำนวนผู้เรียนมาก ดังนั้นจึงได้พัฒนาโปรแกรมประยุกต์ขึ้นมา เพื่อให้เครื่องลูกข่ายสามารถแสดงผลหน้าจอเหมือนเครื่องแม่ข่ายทำให้ลดปัญหาดังกล่าวลงได้

ส่วนการพัฒนาโปรแกรมนี้ได้ใช้ Microsoft Visual Basic โดยใช้ Winsock UDP ในการติดต่อสื่อสารผ่านระบบเครือข่าย ซึ่งโปรแกรมนี้จะทำการจับการแสดงผลหน้าจอของเครื่องแม่ข่ายเป็นระยะๆ แล้วส่งไปยังเครื่องลูกข่ายเพื่อแสดงผลพร้อม ๆ กัน โดยที่ภาพที่ได้จากการจับหน้าจอจะถูกบีบอัดเป็นไฟล์ JPEG เพื่อลดขนาดข้อมูลภาพ ก่อนส่งไปยังเครื่องลูกข่าย

และจากการทดลองใช้งานโปรแกรมสามารถนำการแสดงผลบนหน้าจอของเครื่องแม่ข่ายส่งไปให้เครื่องลูกข่ายเพื่อแสดงผลบนหน้าจอพร้อมๆ กันได้ ซึ่งความเร็วในการแสดงบนหน้าจอภาพของเครื่องลูกข่ายอยู่ในเกณฑ์ที่ใช้ได้ และบางครั้งอาจจะมีความผิดพลาดในการแสดงผลบนหน้าจอของเครื่องลูกข่ายอยู่บ้าง ในการจับหน้าจอที่มีภาพเปลี่ยนแปลงตลอดแต่โอกาสที่จะเกิดขึ้นอย่างนี้ก็ไม่บ่อยมากนัก

ผู้จัดทำหวังว่าการพัฒนาโปรแกรมนี้จะทำให้เกิดประโยชน์ในการสอนและสามารถนำไปประยุกต์หรือพัฒนาให้มีประสิทธิภาพมากขึ้นในอนาคตต่อไปได้

PROJECT TITLE	SCREEN BROADCASTING & CONTROL PROGRAM WITHIN LAN	
STUDENT	Mr. Chalad Satata	No. 43015719
	Mr. Boonlert Chanain	No. 43015729
ADVISOR	Asso.Mayuree Lertwatchakul	
COURSE	Bachelor of Industrial Technology in Electronics	
DEPARTMENT	Information Engineering	
YEAR	2001	

ABSTRACT

The thesis presents designing and implementing a screen broadcasting & control program within LAN. The program was developed to solve the presentation problem. The presentation problem will be occurred when someone need to present or educate many peoples. Even if, we are using projector as a tool to display output of a computer nowadays. But the length of sight and the intensity of light may disturb the audience.

The program was developed by Microsoft Visual Basic. We use Winsock UDP port onto many client-workstations to listen for broadcast messages which are sent by a server-workstation controlled by the instructor. The server-program periodically capture its screen and broadcast them to their client computer using subnet broadcast address. The client workstations can be display screen on monitor as same as the server's screen. But sometimes display are wrong, It not effective for the user.

Be hope to this development program will be useful for learning and it can be develop more effectively in the future.

กิตติกรรมประกาศ

การทำโครงงานปริญญานิพนธ์นี้ สำเร็จได้ด้วยดีเกิดจากการให้คำแนะนำและการให้ข้อมูลของท่านอาจารย์ที่ปรึกษาโครงงานและคณะคณาจารย์ทุกท่านที่ได้อบรบสั่งสอน รวมทั้งได้สนับสนุนเครื่องมือและตำรา, เทคนิคต่าง ๆ รวมถึง Web Site ต่าง ๆ ที่ได้ให้ข้อมูลคำแนะนำและวิธีการที่นำมาประกอบในการพัฒนาโปรแกรมที่ใช้ในการทำปริญญานิพนธ์นี้ รวมทั้งครอบครัวที่ให้การสนับสนุนในการศึกษาที่ผ่านมาจนกระทั่งถึงปัจจุบันนี้ จึงขอกราบขอบพระคุณทุก ๆ ท่านเป็นอย่างสูง

ผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญรูปภาพ

หน้า

สารบัญตาราง

หน้า

บทที่ 1 บทนำ

1.1 ความสำคัญและที่มา

โปรแกรมประยุกต์นี้ได้เกิดแนวความคิดมาจากการเรียนการสอนในห้องคอมพิวเตอร์ ซึ่งจะเห็นว่าการเรียนการสอนที่ใช้คอมพิวเตอร์เป็นสื่อกลางในการสอนนั้นมีความจำเป็นมากขึ้น แต่การใช้สื่อการสอนในสภาพแวดล้อมที่ไม่เหมาะสมก็อาจจะไม่ได้ผลที่ดีมากนัก เช่น การใช้คอมพิวเตอร์ร่วมกับโปรเจคเตอร์ เป็นต้น เมื่อจำนวนผู้เรียนมากขึ้นการแสดงผลของโปรเจคเตอร์ก็อาจจะไม่ชัดเจนเมื่อผู้เรียนอยู่ไกลจากผู้สอน หรือสภาพแวดล้อมมีแสงสว่างมากเกินไปซึ่งอาจจะทำให้ผู้เรียนไม่เข้าใจในสิ่งที่เรียนหรือตามการสอนของผู้สอนไม่ทัน จึงได้คิดจัดทำโปรแกรมประยุกต์ขึ้นมา เพื่อแก้ปัญหาดังกล่าวโดยการทำสำเนาหน้าจอของผู้สอนและกระจายข้อมูลหน้าจอออกไปสู่ระบบเครือข่าย เพื่อนำไปแสดงผลยังเครื่องลูกข่ายที่ผู้เรียนใช้งานอยู่

1.2 วัตถุประสงค์

- ศึกษาการเขียนโปรแกรม Visual Basic
- ศึกษาระบบเครือข่ายคอมพิวเตอร์
- ศึกษาการทำงานของระบบปฏิบัติการวินโดว์
- เพื่อใช้เป็นสื่อช่วยในการสอน ทำให้การสอนมีประสิทธิภาพมากยิ่งขึ้น

1.3 ขอบเขตของโครงงาน

โปรแกรมประยุกต์นี้สามารถจับหน้าจอของเครื่องแม่ข่ายเป็นระยะ ๆ แล้วส่งไปยังเครื่องลูกข่าย โดยข้อมูลภาพที่ส่งไปนั้นจะถูกบีบอัดเป็นไฟล์ JPEG เพื่อลดขนาดไฟล์ภาพก่อนที่จะทำการส่งออกไปในระบบเครือข่าย เพื่อไปแสดงผลบนหน้าจอของเครื่องลูกข่ายให้เหมือนกับที่แสดงผลบนเครื่องแม่ข่าย โดยใช้ Winsock UDP ในการติดต่อระบบเครือข่าย

1.4 ขั้นตอนการดำเนินโครงงาน

โปรแกรมประยุกต์นี้ถูกพัฒนาโดยใช้ภาษา Visual Basic 6.0 ซึ่งเป็นภาษาคอมพิวเตอร์ที่ได้รับความนิยมในการพัฒนาโปรแกรมบนวินโดว์ เนื่องจากสามารถสร้างจอภาพที่ใช้สำหรับติดต่อกับผู้ใช้และง่ายต่อการศึกษา รวมถึงการเขียนโปรแกรมในลักษณะของ Event-driven ซึ่งเป็นการ

เขียนโปรแกรมเพื่อกำหนดขั้นตอนการทำงานให้กับ Control ต่าง ๆ ที่ทำงานตามเหตุการณ์ (Event) ต่างๆ ที่เกิดขึ้น เช่น การเลื่อนเมาส์ หรือการรับข้อมูลจากคีย์บอร์ด ฯลฯ

สำหรับการวางแผนการพัฒนาโปรแกรมประยุกต์ สามารถแบ่งออกได้ดังนี้ คือ

- ศึกษาและทดลองเขียนโปรแกรม Visual Basic 6.0
- ศึกษาและทดลองเขียนโปรแกรมสื่อสารบนระบบเครือข่ายโดยใช้ Winsock UDP
- ศึกษาและทดลองเขียนโปรแกรมการจับหน้าจอ โดยอาศัย Device Context
- ศึกษาและทดลองใช้โปรแกรมบีบอัดข้อมูลภาพ
- รวมโปรแกรมทั้งหมดเข้าด้วยกัน และทดสอบการทำงาน
- แก้ไขและดัดแปลงโปรแกรมประยุกต์เพื่อแก้ไขปัญหาที่พบในช่วงทดสอบการทำงาน

บทที่ 2 ทฤษฎีและหลักการ

2.1 แนะนำ Visual Basic 6

Visual Basic 6.0 เป็นภาษาหนึ่งที่มีความใกล้ชิดกับระบบปฏิบัติการวินโดว์ เพราะมีความสามารถต่างๆ และคำสั่งต่างๆ ส่วนใหญ่จะใช้ร่วมกับความสามารถของวินโดว์ได้โดยตรง ซึ่งในทางปฏิบัติก็หมายความว่า Visual Basic 6.0 ได้ดึงเอาความสามารถของวินโดวส์ มาเป็นส่วนหนึ่งของความสามารถของตัวแปลภาษา เพื่อช่วยให้การสร้างแอพพลิเคชั่นที่สามารถใช้ประโยชน์จากความสามารถของวินโดวส์ได้อย่างเต็มที่

แอพพลิเคชั่นที่สร้างด้วย Visual Basic 6.0 ส่วนใหญ่จะประกอบด้วยองค์ประกอบหลัก 3 ส่วนด้วยกันดังนี้ ฟอร์ม (From),คอนโทรล (ActiveX Custom Control) และออบเจ็กต์ระบบ (System Object)

ฟอร์มและคอนโทรล

เป็นออบเจ็กต์ที่ถูกออกแบบเฉพาะและถูกจัดการโดยตรงจาก Visual Basic 6.0 ซึ่งออบเจ็กต์จะมีความสามารถด้านการปรับแต่งและตอบสนองต่อเหตุการณ์ต่างๆได้อย่างมากมาย ซึ่งเราสามารถแบ่งองค์ประกอบที่เกี่ยวข้องกับออบเจ็กต์ออกได้เป็น 3 ประเภท ดังนี้

คุณสมบัติ (Property) หมายถึง แอตทริบิวส์ของออบเจ็กต์ ที่อนุญาตให้ผู้อ่านแก้ไขหรือปรับแต่งได้ตามที่ต้องการ เช่น คุณสมบัติ Visible หรือ Enabled ของฟอร์มเป็นต้น ซึ่งคุณสมบัติบางรายการจะส่งผลต่อการอ้างถึงออบเจ็กต์นั้นๆ เป็นการเฉพาะก็มี เช่น Name เป็นต้น ซึ่งในกรณีหลังจะไม่มีผลต่อการแสดงผลใดๆ ทั้งสิ้น และบางคุณสมบัติจะไม่สามารถแก้ไขได้ เพราะถูกกำหนดให้เป็นคุณสมบัติที่สามารถอ่านค่าได้อย่างเดียว(read only) เท่านั้น

โพรซิเยอร์เหตุการณ์ (Event) หมายถึง เหตุการณ์ที่ถูกสร้างขึ้นมาตามความสามารถของวินโดวส์ เพื่อตอบสนองต่อการกระทำหรือการเปลี่ยนแปลงคุณสมบัติของออบเจ็กต์ โดยปกติผู้อ่านจะทำการเขียนโค้ดลงใน โพรซิเยอร์เหตุการณ์เพื่อตอบสนองต่อเหตุการณ์ที่เกิดขึ้น

วิธี (Method) หมายถึง การกระทำบนออบเจ็กต์ เช่น วิธี Clear ของคอนโทรล ListBox เป็นต้นซึ่งในความเป็นจริงวิธีก็เป็นเพียงฟังก์ชั่นหนึ่งๆ ของออบเจ็กต์ ที่มีไว้เพื่อสนับสนุนการจัดการกับออบเจ็กต์เท่านั้น

ออบเจ็กต์ระบบ

ออบเจ็กต์ระบบ หมายถึง ออบเจ็กต์ที่สนับสนุนโดยระบบปฏิบัติการวินโดวส์ ซึ่ง Visual Basic 6.0 ได้สนับสนุนการเรียกใช้ออบเจ็กต์ระบบได้โดยตรงอยู่แล้ว เช่น Printer Screen Menu หรือ Clipboard เป็นต้น โดยที่ผู้อ่านสามารถที่จะแก้ไขคุณสมบัติของออบเจ็กต์ระบบเหล่านั้นได้โดยตรงจาก Visual Basic 6.0 [3]

2.2 ความรู้เบื้องต้นเกี่ยวกับระบบเครือข่าย

การพัฒนาโปรแกรมประยุกต์บนระบบเครือข่ายจำเป็นต้องมีความรู้พื้นฐานเกี่ยวกับระบบเครือข่ายคอมพิวเตอร์ มาตรฐานและการทำงานของระบบเครือข่าย ซึ่งจะกล่าวเบื้องต้นดังต่อไปนี้

2.2.1 มาตรฐาน OSI (Open System Interconnection)

OSI เป็นรูปแบบมาตรฐานในการเชื่อมต่อระบบเครือข่ายคอมพิวเตอร์ซึ่งถูกกำหนดขึ้นโดย ISO (International Standards Organization) เพื่อให้คอมพิวเตอร์หรือระบบคอมพิวเตอร์อื่นที่ใช้มาตรฐาน OSI เหมือนกันสามารถติดต่อไปมาหาสู่ระหว่างกันได้

โดยสามารถแบ่งออกเป็น 7 เลเยอร์ ดังนี้

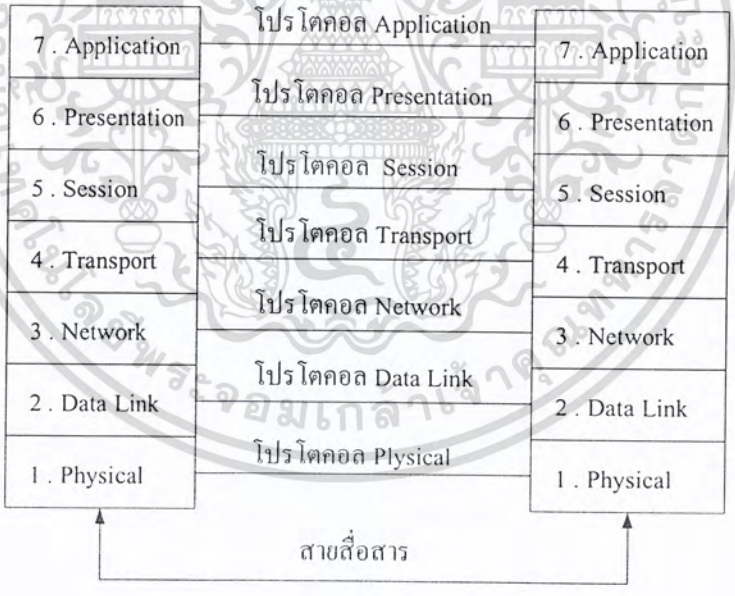
1.) ฟิสิคัลเลเยอร์ (Physical Layer) เป็นเลเยอร์ล่างสุดของการติดต่อสื่อสาร สร้างสัญญาณไฟฟ้าและสัญญาณวิทยุสำหรับการรับและส่งข้อมูลจริง ๆ จากช่องทางการสื่อสาร (สื่อกลาง) ระหว่างคอมพิวเตอร์เครื่องหนึ่งกับคอมพิวเตอร์อื่น ๆ เช่น RS-232 , X.21

2.) ดาต้าลิงค์เลเยอร์ (Data Link Layer) จัดกลุ่มข้อมูลที่ต้องการจะนำส่งแบ่งเป็นเฟรมเพื่อส่งให้กับ ฟิสิคัลเลเยอร์ และตรวจจับเฟรมจาก Bit-stream ที่ได้รับจากฟิสิคัลเลเยอร์ เพื่อส่งให้กับชั้นควบคุมเครือข่าย และนอกเหนือจากนี้ชั้นดาต้าลิงค์เลเยอร์อาจจะทำหน้าที่เป็นเสมือนผู้ตรวจสอบ หรือควบคุมความผิดพลาดในข้อมูลโดยจะแบ่งข้อมูลที่ส่งออก เป็นแพ็กเก็ตหรือเฟรม ถ้าผู้รับได้รับข้อมูลถูกต้องก็จะส่งสัญญาณยืนยันกลับว่าได้รับข้อมูลแล้ว เรียกว่าสัญญาณ ACK (Acknowledge) ให้กับผู้ส่ง แต่ถ้าผู้ส่งไม่ได้รับสัญญาณ ACK หรือได้รับสัญญาณ NAK (Negative

Acknowledge) กลับมา ผู้ส่งก็อาจจะทำการส่งข้อมูลไปให้ใหม่ อีกหน้าที่หนึ่งของเลเยอร์นี้คือป้องกันไม่ให้เครื่องส่งทำการส่งข้อมูลเร็วจนเกิดขีดความสามารถจนเครื่องผู้รับจะรับข้อมูลได้

3.) เน็ตเวิร์คเลเยอร์ (Network Layer) เป็นเลเยอร์ที่กำหนดที่อยู่ (Address) และเส้นทางการเดินทางข้อมูลที่ส่ง-รับในการส่งผ่านข้อมูลระหว่างต้นทางและปลายทาง ซึ่งแน่นอนว่าในการสื่อสารข้อมูลผ่านเครือข่ายการสื่อสารจะต้องมีเส้นทางการรับ-ส่งข้อมูลมากกว่า 1 เส้นทาง ดังนั้นเน็ตเวิร์คเลเยอร์นี้จะมีหน้าที่เลือกเส้นทางที่ใช้เวลาในการสื่อสารน้อยที่สุดและระยะทางสั้นที่สุดด้วย ข่าวสารที่รับมาจากเลเยอร์ที่ 4 จะถูกแบ่งออกเป็นข่าวสารย่อยๆที่เรียกว่า แพ็กเกจ ในเลเยอร์ที่ 3 นี้

4.) ทรานสปอร์ตเลเยอร์ (Transport Layer) บางครั้งเรียกว่า Host-to-Host เลเยอร์หรือเครื่องต่อเครื่องและจากเลเยอร์ที่ 4 ถึงเลเยอร์ที่ 7 นี้รวมกันจะเรียกว่า End-to-End เลเยอร์ในทรานสปอร์ตเลเยอร์นี้เป็นการสื่อสารกันระหว่างเครื่องคอมพิวเตอร์ต้นทางและเครื่องคอมพิวเตอร์ปลายทาง ทรานสปอร์ตเลเยอร์จะทำหน้าที่ตรวจสอบว่าข้อมูลที่ส่งมาจากเซสชั่นเลเยอร์นั้นไปถึงปลายทางจริง ๆ หรือไม่



รูปที่ 2.1 โครงสร้างของสถาปัตยกรรมรูปแบบ OSI

5.) เซสชั่นเลเยอร์ (Session Layer) ทำหน้าที่เชื่อมโยงระหว่างผู้ใช้งานกับคอมพิวเตอร์เครื่องอื่น ๆ โดยผู้ใช้จะใช้คำสั่งหรือข้อความที่กำหนดไว้ป้อนเข้าไปในระบบ ในการสร้างการเชื่อม

โยงนี้ ผู้ใช้จะต้องกำหนดรหัสตำแหน่งของโปรแกรมประยุกต์ปลายทาง ที่ต้องการติดต่อสื่อสารด้วยเซสชั่น เลเยอร์จะส่งข้อมูลทั้งหมดให้กับทรานสปอร์ตเลเยอร์เป็นผู้จัดการต่อไป ในบางเครือข่ายทั้งเซสชั่นเลเยอร์และทรานสปอร์ตเลเยอร์อาจจะเป็นเลเยอร์เดียวกัน

6.) พรีเซนเตชั่นเลเยอร์ (Presentation Layer) ทำหน้าที่เหมือนล่าม กล่าวคือคอยรวบรวมข้อความและแปลงรหัสหรือแปลงรูปแบบของข้อมูลให้เป็นรูปแบบการสื่อสารเดียวกัน เพื่อช่วยลดปัญหาต่างๆ ที่อาจจะเกิดขึ้นกับผู้ใช้งานระบบ

7.) แอพพลิเคชั่นเลเยอร์ (Application Layer) เป็นเลเยอร์บนสุดของรูปแบบ OSI ซึ่งเป็นเลเยอร์ที่ใช้ติดต่อกันระหว่างผู้ใช้โดยตรง ซึ่งได้แก่ โปรแกรมจำลองการเป็นเทอร์มินัล (Telnet) โปรแกรมรับ-ส่งไฟล์ข้อมูล (FTP) โปรแกรมจดหมายอิเล็กทรอนิกส์ (e-mail) เป็นต้น แอพพลิเคชั่นในเลเยอร์นี้สามารถนำเข้าหรือออกจากระบบเครือข่ายได้โดยไม่จำเป็นต้องสนใจว่าจะมีขั้นตอนการทำงานอย่างไร เพราะจะมีพรีเซนเตชั่นเลเยอร์เป็นผู้รับผิดชอบแทนอยู่แล้ว ในรูปแบบ OSI เลเยอร์นั้น Application จะทำการติดต่อกับพรีเซนเตชั่นเลเยอร์โดยตรงเท่านั้น

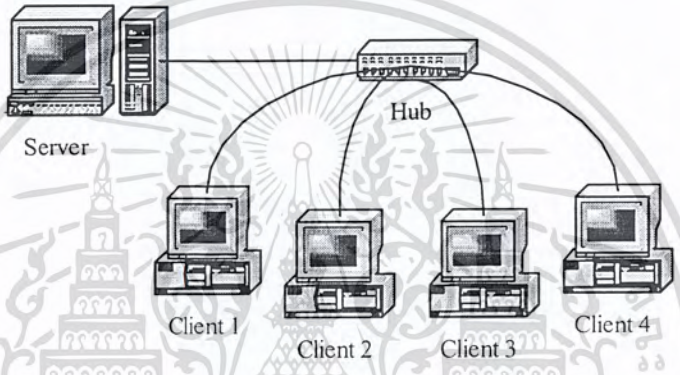
โปรโตคอลของในแต่ละเลเยอร์จะแตกต่างกันออกไป แต่อย่างไรก็ตามการที่เครื่องคอมพิวเตอร์หลาย ๆ เครื่องจะติดต่อสื่อสารกันได้ในแต่ละเลเยอร์ของแต่ละเครื่องจะต้องใช้โปรโตคอลแบบเดียวกันหรือถ้าใช้โปรโตคอลต่างกันก็ต้องมีอุปกรณ์หรือซอฟท์แวร์ที่สามารถแปลงโปรโตคอลที่ต่างกันนั้นให้มีรูปแบบเป็นอย่างเดียวกัน เพื่อเชื่อมโยงให้คอมพิวเตอร์ทั้ง 2 เครื่องให้สามารถติดต่อกันได้ [2]

2.2.2 ประเภทของระบบเครือข่ายแลนซึ่งแบ่งตามลักษณะการทำงาน

ในการแบ่งรูปแบบการเชื่อมต่อระบบเครือข่ายแลนนั้น สามารถแบ่งออกเป็น 2 ประเภทใหญ่ ๆ ได้แก่การเชื่อมต่อแบบ Peer - To - Peer และแบบ Client / Server

1.) แบบ Peer-to-Peer เป็นการเชื่อมต่อเครื่องคอมพิวเตอร์เข้าด้วยกัน โดยเครื่องคอมพิวเตอร์ แต่ละเครื่องจะสามารถแบ่งทรัพยากรต่างๆ ไม่ว่าจะเป็นไฟล์หรือเครื่องพิมพ์ซึ่งกันและกันภายในเครือข่ายได้ เครื่องแต่ละเครื่องจะทำงานในลักษณะที่ทัดเทียมกัน ไม่มีเครื่องใดเครื่องหนึ่งเป็นเครื่องหลักเหมือนแบบ Client / Server แต่ก็ยังคงคุณสมบัติพื้นฐานของระบบเครือข่ายไว้เหมือนเดิม การเชื่อมต่อแบบนี้มักทำในระบบที่มีขนาดเล็กๆ เช่น หน่วยงานขนาดเล็กที่มีเครื่องใช้ไม่เกิน 10 เครื่อง การเชื่อมต่อแบบนี้มีจุดอ่อนในเรื่องของระบบรักษาความปลอดภัย แต่ถ้าเป็นเครือข่ายขนาดเล็กและเป็นงานที่ไม่มีข้อมูลที่เป็นความลับมากนัก เครือข่ายแบบนี้ ก็เป็นรูปแบบที่น่าเลือกนำมาใช้ได้เป็นอย่างดี

2.) แบบ client-server เป็นระบบที่มีเครื่องคอมพิวเตอร์ลูกข่ายทุกเครื่องมีฐานะการทำงานที่เหมือน ๆ กัน เท่าเทียมกันภายในระบบเครือข่าย แต่จะมีเครื่องคอมพิวเตอร์เครื่องหนึ่งที่ทำหน้าที่เป็นเครื่อง Server ที่ทำหน้าที่ให้บริการทรัพยากรต่าง ๆ ให้กับเครื่อง Client หรือเครื่องที่ขอใช้บริการ ซึ่งอาจจะต้องเป็นเครื่องที่มีประสิทธิภาพที่ค่อนข้างสูง ถึงจะทำให้การให้บริการมีประสิทธิภาพตามไปด้วย ข้อดีของระบบเครือข่าย Client - Server เป็นระบบที่มีการรักษาความปลอดภัยสูงกว่า ระบบแบบ Peer To Peer เพราะว่าการจัดการในด้านรักษาความปลอดภัยนั้น จะทำกันบนเครื่อง Server เพียงเครื่องเดียวทำให้ดูแลรักษาง่ายและสะดวก มีการกำหนดสิทธิการเข้าใช้ทรัพยากรต่าง ๆให้กับเครื่องผู้ขอใช้บริการหรือเครื่อง Client



รูปที่ 2.2 รูปแสดงการเชื่อมต่อแบบ Client - Server

2.2.3 โปรโตคอลทีซีพี / ไอพี (TCP/IP)

โปรโตคอล TCP/IP มีการจัดกลไกการทำงานเป็นชั้นหรือเลเยอร์เรียงต่อกันไปเหมือนกับ OSI - Reference Model โดยในแต่ละเลเยอร์นั้นจะมีการทำงานเทียบได้กับ OSI - Reference Model แต่บางเลเยอร์ของโปรโตคอล TCP/IP จะทำงานเทียบกับ OSI-Reference Model หลายเลเยอร์ปนกัน ซึ่งในแต่ละเลเยอร์ของโปรโตคอล TCP/IP จะประกอบไปด้วย

- โพรเซส เลเยอร์ (Process Layer) ได้แก่ Telnet, FTP, DNS
- โฮสต์-ทู-โฮสต์ เลเยอร์ (Host-To-Host Layer) ได้แก่ TCP, UDP
- อินเตอร์เน็ตเวิร์ค เลเยอร์ (Internetwork Layer)ได้แก่ไอพีแอดเดรส,ไอซีเอ็มพี,เออาร์พี ,อาร์เออาร์พี
- เน็ตเวิร์ค อินเตอร์เฟซ เลเยอร์ (Network Interface Layer) ได้แก่ อีเทอร์เน็ต,โทเค็นริ่ง

โดยเมื่อได้เทียบลำดับเลเยอร์กับมาตรฐานของ OSI - Reference Model แล้ว จะเป็นดังรูปที่ 2.3 ซึ่งเราจะเห็นว่าบางเลเยอร์ของ TCP/IP นั้นจะเทียบได้กับมาตรฐาน ISO Model ได้ 2 เลเยอร์

อย่างเช่นเลเยอร์ของโพรเซสเลเยอร์ของโปรโตคอล TCP/IP จะเทียบได้กับ 2 เลเยอร์ คือแอพพลิเคชั่นเลเยอร์กับพรีเซนเตชั่นเลเยอร์ของ OSI - Reference Model รวมกัน เป็นต้น

OSI - Reference Model		โปรโตคอล ทีซีพี/ไอพี
Application Layer		Process Layer
Preentation Layer		
Session Layer		Host- to-Host Layer
Transport Layer		
Network Layer		Internetwork Layer
Data Link Layer		Network Interface Layer
Phisical Layer		

รูปที่ 2.3 TCP/IP เปรียบเทียบกับ OSI Model

ซึ่งในแต่ละเลเยอร์ได้มีการกำหนดหน้าที่การทำงานไว้ดังต่อไปนี้

1. โพรเซสเลเยอร์ จะเป็นแอพพลิเคชั่นโปรโตคอลเชื่อมต่อกับผู้ใช้และให้บริการต่าง ๆ โปรโตคอลหลัก ๆ ที่ทำงานและให้บริการในโพรเซสเลเยอร์ นี้ก็มีอย่างเช่น เอฟทีพี (FTP), เทลเน็ต (Telnet), เอชทีทีพี (HTTP),เอสเอ็มทีพี (SMTP) เป็นต้น จากรูปที่ 2.3 แสดงลำดับเลเยอร์การทำงานของโปรโตคอล TCP/IP เทียบกับมาตรฐานของ OSI - Reference Model นั้น ในเลเยอร์บนสุดที่เรียกว่าโพรเซสเลเยอร์ทำงาน 2 หน้าที่เทียบได้กับแอพพลิเคชั่นเลเยอร์และพรีเซนเตชั่นเลเยอร์

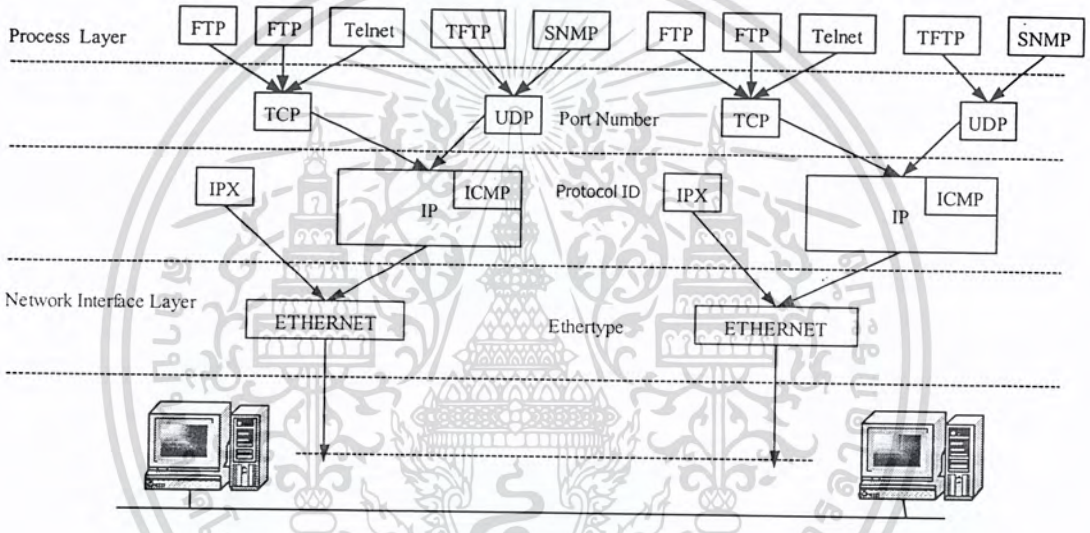
2. โฮสต์-ทู-โฮสต์เลเยอร์ จะเป็น TCP หรือ UDP ที่ทำหน้าที่คล้ายกับเลเยอร์ของเซสชั่นเลเยอร์และทรานสปอร์ตเลเยอร์ของ OSI - Model คือควบคุมการรับส่งข้อมูล จากปลายด้านส่งถึงปลายด้านรับข้อมูล และตัดข้อมูลออกเป็นส่วนย่อย (Segment) ให้เหมาะสมกับเครือข่ายที่ใช้รับส่งข้อมูลรวมทั้งประกอบข้อมูลส่วนย่อย ๆ นี้เข้าด้วยกันเมื่อถึงปลายทาง

3. อินเตอร์เน็ตเวิร์คเลเยอร์ ได้แก่ส่วนของโปรโตคอลไอพีซึ่งทำหน้าที่คล้ายกับเลเยอร์ของเน็ตเวิร์คของ OSI–Model คือเชื่อมต่อคอมพิวเตอร์เข้ากับระบบเครือข่ายที่อยู่เลเยอร์ล่างลงไป และทำหน้าที่เลือกเส้นทางการรับส่งข้อมูลผ่านอุปกรณ์เครือข่ายต่าง ๆ จนไปถึงผู้รับข้อมูล ในเลเยอร์นี้จะจัดการกับกลุ่มข้อมูลในลักษณะที่เรียกว่า IP Packet ในรูปแบบของ TCP/IP ที่เรารู้จักกันนั่นเอง

4.เน็ตเวิร์คอินเตอร์เฟสเลเยอร์ เป็นเลเยอร์ที่ควบคุมฮาร์ดแวร์ การรับส่งข้อมูลผ่านระบบเครือข่าย ซึ่งเทียบได้กับดาต้าลิงค์เลเยอร์กับฟิสิคัลเลเยอร์ของ OSI ในเลเยอร์นี้จะทำหน้าที่เชื่อมต่อกับฮาร์ดแวร์และควบคุมการรับส่งข้อมูลในระบบฮาร์ดแวร์ของเครือข่าย ซึ่งที่ใช้กันอยู่จะเป็นตาม

มาตรฐานของ IEEE เช่น IEEE 802.3 จะเป็นการเชื่อมต่อผ่านแลนแบบอีเทอร์เน็ตแลนหรือ IEEE 802.5 จะเป็นการเชื่อมต่อผ่านแลนแบบโทเค็นริง เป็นต้น

เราจะเห็นได้ว่าที่จริงแล้วโปรโตคอล TCP/IP นั้นแบ่งออกเป็น 2 โปรโตคอลซ้อนกันอยู่ คือ ทีซีพีจะอยู่ในเลเยอร์บนและไอพีจะอยู่ในเลเยอร์ถัดลงมา นั่นคือ TCP/IP ไม่ได้เป็นโปรโตคอลชนิดเดียวกันทั้งหมดและไม่ได้เชื่อมติดเป็นชิ้นเดียวกันทั้งหมดทีซีพีก็มีมาตรฐานของเฟรมที่ใช้รับส่งข้อมูลของมันเองและมีหน้าที่ในการรับส่งข้อมูลแตกต่างไปจากไอพีซึ่งในการรับส่งข้อมูลนั้นเฟรมของทีซีพีที่อยู่ในเลเยอร์บนทั้งหมดจะถูกผนึกอยู่ในส่วนที่เป็นข้อมูลของไอพี เหมือนกับที่แต่ละเลเยอร์ของ OSI - Reference Model ผนึกข้อมูลในเลเยอร์ถัดไปนั่นเอง [2]



รูปที่ 2.4 ภาพแสดงการรับส่งข้อมูลผ่านโปรโตคอล TCP/IP

2.2.4 โปรโตคอลทีซีพี (TCP)

เป็นโปรโตคอลที่มีการรับส่งข้อมูลแบบสตรีม (Stream Oriented Protocol) หมายความว่าการรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่งไป แต่จะแบ่งข้อมูลเป็นส่วนย่อย ๆ ก่อน แล้วจึงจะส่งไปยังปลายทางอย่างต่อเนื่องเป็นลำดับข้อมูล ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไป ก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้ง สำหรับปลายทางก็จะทำหน้าที่จัดเรียงส่วนของข้อมูลดาต้าแกรม (Datagram) ใหม่ให้ต่อเนื่องกันและประกอบกลับเป็นข้อมูลทั้งหมดได้ ซึ่งจะแยกข้อมูลที่ไม่ถูกต้องออก ดังนั้นแอพพลิเคชั่นหรือโปรเซสใดที่อาศัยการส่งผ่านข้อมูลด้วยโปรโตคอลทีซีพีจะต้องใช้หน่วยความจำและขนาดของช่องสัญญาณ (Bandwidth) มากกว่ายูดีพี

การติดต่อระหว่างกันจะต้องเป็นแบบ Connection-oriented คือต้องมีการสร้างการติดต่อกันเป็น session ทั้ง 2 ด้านเสียก่อน แล้วจึงจะรับส่งข้อมูลไปได้พร้อมกัน (Full duplex) เหมือนกับการใช้โทรศัพท์ติดต่อกัน เมื่อผู้ติดต่อต้นทางเรียกให้ฝ่ายตรงข้ามรับสายแล้ว จึงเริ่มการสนทนา เช่น พูดคำว่า "สวัสดี" หรือ "ฮัลโหล" กันก่อนเพื่อให้แน่ใจว่าฝ่ายตรงข้ามพร้อมจะติดต่อด้วย จากนั้นจึงเริ่มต้นติดต่อกัน และเมื่อต้องการจะเลิกการติดต่อก็จะมีการพูดคำว่า "สวัสดี" ให้ฝ่ายตรงข้ามทราบว่าจะเลิกการติดต่อและวางสายไป ซึ่งในระหว่างการติดต่อกันนั้น แม้ว่าฝ่ายใดฝ่ายหนึ่งหรือทั้งสองฝ่ายจะเงียบไป คือไม่พูดอะไรเป็นเวลานาน ๆ แต่การเชื่อมโยงระหว่างทั้งสองด้านยังคงมีอยู่ไม่ขาดไปจนกว่าฝ่ายใดฝ่ายหนึ่งจะวางสาย เช่นเดียวกับการติดต่อกันด้วยกลไกโปรโตคอลทีซีพี เมื่อแอพพลิเคชั่นต้องการส่งผ่านข้อมูลจะใช้โปรโตคอลที่เหมาะสมในโพรเซสเลเยอร์ ติดต่อไปและมีการสร้างช่องส่งข้อมูลผ่านพอร์ตที่กำหนดเพื่อส่งผ่านข้อมูลไปยังโปรโตคอลทีซีพี

ในระหว่างการรับส่งข้อมูลนี้ โปรโตคอลทีซีพีจะเพิ่มขบวนการสอบทานข้อมูลเพื่อให้ข้อมูลมีความถูกต้องไม่ผิดพลาดไปจากเดิม โดยการส่งสัญญาณสอบทานข้อมูล (Acknowledgement) และส่งข้อมูลให้ใหม่อีกครั้ง ถ้าปลายทางไม่ได้รับหรือเกิดความผิดพลาดขึ้น

ความน่าเชื่อถือของการส่งผ่านข้อมูลโดยโปรโตคอลทีซีพีจะมีมากกว่า แต่ก็ต้องอาศัยทรัพยากรของระบบมากกว่าในการทำงานเช่นกัน [1]

2.2.5 โปรโตคอลยูดีพี (UDP)

การรับส่งข้อมูลผ่านโปรโตคอลยูดีพีจะเป็นแบบที่ทั้งสองด้านไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกัน (Connectionless) ระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องที่ขอใช้บริการโดยไม่ต้องแจ้งให้ฝ่ายรับข้อมูลเตรียมรับข้อมูลเหมือนโปรโตคอลทีซีพีและไม่มีการตรวจสอบความถูกต้องครบถ้วนในการรับส่งข้อมูลนั้น ๆ ด้วย เนื่องจากโปรโตคอลยูดีพี ไม่มีสัญญาณสอบทานข้อมูล ในการส่งข้อมูลแต่ละครั้งและไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความผิดพลาดของการส่งข้อมูล เมื่อเป็นเช่นนี้แอพพลิเคชั่นหรือโพรเซสใดที่ต้องอาศัยโปรโตคอลยูดีพี ในการส่งผ่านข้อมูลก็อาจจะต้องสร้างขบวนการตรวจสอบข้อมูลขึ้นมาเอง [1]

2.2.6 ไอพี แอดเดรส (IP Address)

หมายเลขไอพีแอดเดรส นั้นจะประกอบไปด้วยตัวเลขฐาน 2 จำนวน 4 ไบต์หรือ 32 บิต โดยจะแยกออกเป็น 2 ส่วน คือหมายเลขเครือข่ายและหมายเลขเครื่อง แต่เนื่องจากว่าตัวเลขไบนารี 32 หลัก เป็นตัวเลขที่จดจำได้ยากแต่เครื่องคอมพิวเตอร์ก็ใช้เลขเหล่านี้ได้อย่างถูกต้อง แต่เมื่อกำหนดเลข 4 ไบต์ แต่ละไบต์ มีขนาด 0-255 เมื่อดูแล้วจะทำให้จำได้ง่ายขึ้น เนื่องจากขนาดของ

เครือข่ายมีขนาดแตกต่างกันมาก ดังนั้นจึงมีการกำหนดการแบ่งคลาสของเครือข่ายออกเป็น 3 คลาสคือ คลาส A คลาส B คลาส C

- คลาส A

กำหนดตัวเลขไอพีเน็ตเวิร์คเพียงไบต์แรกไบต์เดียว ที่เหลืออีกสามไบต์จึงเป็นรหัสประจำเครื่องที่อยู่ในเครือข่าย หมายเลข ไอพีแอดเดรส ตำแหน่งบิตแรก ของไอพีเน็ตเวิร์คจะต้องเป็น '0' บิตที่เหลืออีก 7 บิตจึงเป็นหมายเลขไอพีเน็ตเวิร์คทำให้คลาส A มีจำนวนเครือข่ายได้ทั้งหมด 127 เครือข่าย ส่วน 3 ไบต์ที่เหลือ มีจำนวน 24 บิต ทำให้มีโนด (Node)ได้ 16,777,216 โนด โดยมีช่วงไอพีแอดเดรส ที่ใช้ได้อยู่ที่ 1.0.0.0 ถึง 127.255.255.255

- คลาส B

กำหนดตัวเลขไอพีเน็ตเวิร์คที่ 2 ไบต์แรก จึงเป็นรหัสประจำเครื่องที่อยู่ในเครือข่ายหมายเลขไอพีแอดเดรสตำแหน่ง 2 บิต แรกของไอพีเน็ตเวิร์คจะต้องเป็น 10 บิตที่เหลืออีก 14 บิต จึงเป็นหมายเลขไอพีเน็ตเวิร์ค ทำให้คลาส B มีจำนวนเครือข่ายได้ทั้งหมด 16,384 เครือข่าย ส่วน 2 ไบต์ ที่เหลือ มีจำนวน 16 บิต ทำให้มีโนดได้ 65,534 Nodeโดยมีช่วงไอพีแอดเดรสที่ใช้ได้อยู่ที่ 128.0.0.0 ถึง191.255.255.255

- คลาส C

กำหนดตัวเลขไอพีเน็ตเวิร์คที่ 3 ไบต์แรก ที่เหลืออีก 1 ไบต์ จึงเป็นรหัสประจำเครื่อง ที่อยู่ในเครือข่ายหมายเลขไอพีแอดเดรสตำแหน่ง 3 บิตแรก ของไอพีเน็ตเวิร์คจะต้องเป็น 110 บิตที่เหลืออีก 21 บิต จึงเป็นหมายเลขไอพีเน็ตเวิร์คทำให้คลาส C มีจำนวนเครือข่าย ได้ทั้งหมด 2,097,152 เครือข่าย ส่วน 1 ไบต์ ที่เหลือมีจำนวน 8 บิต ทำให้มี Node ได้ 254 Node โดยมีช่วงไอพีแอดเดรสที่ใช้ได้อยู่ที่ 192.0.0.0 ถึง 223.255.255.255

- คลาส D

ใช้สำหรับกระจายข้อมูลข่าวสารแบบหลายจุด ตำแหน่ง 4 บิตแรก ของไอพีเน็ตเวิร์คจะต้องเป็น 1110 มีช่วงไอพีแอดเดรสที่ใช้ได้อยู่ที่ 224.0.0.0 ถึง 239.255.255.255

- คลาส E

มีไว้ใช้สำรองในอนาคต ตำแหน่ง 5 บิตแรก ของไอพีเน็ตเวิร์คจะต้องเป็น 11110 มีช่วงไอพีแอดเดรสที่ใช้ได้อยู่ที่ 240.0.0.0 ถึง 247.255.255.255

การใช้ไอพีแอดเดรสกำหนดเลขเครือข่ายและโฮสต์มีข้อกำหนดปลีกย่อยที่ผู้วางระบบเครือข่ายต้องเข้าใจเพื่อเลือกใช้ได้อย่างถูกต้อง ข้อกำหนดที่กล่าวถึงนี้คือแอดเดรสที่ไม่อนุญาตให้ใช้งานและแอดเดรสที่สงวนไว้ใช้งานในความหมายเฉพาะอย่างตามรูปแบบ โดยมีรายละเอียดดังนี้

1.) แอดเดรสกำหนดเครือข่าย

แอดเดรสที่มีเลขโฮสต์เป็น "0" ทั้งหมด ใช้กำหนดตัวเครือข่ายจึงไม่สามารถนำมาใช้เป็นไอพีแอดเดรสสำหรับอินเทอร์เฟสได้ เช่น 5.0.0.0 เป็นเครือข่ายหนึ่งในคลาส A 158.108.0.0 เป็นเครือข่ายในคลาส B และ 202.105.100.0 เป็นเครือข่ายในคลาส C

2.) แอดเดรสบรอดคาสต์ (Broadcast Address)

แอดเดรสที่มีเลขโฮสต์ทุกบิตเป็น "1" ใช้เป็นแอดเดรสบรอดคาสต์โดยตรงภายในเครือข่าย(net directed broadcast) เช่น เครือข่าย 158.108.0.0/16 มีแอดเดรสบรอดคาสต์เท่ากับ 158.108.255.255 และเครือข่าย 202.105.100.0/24 มีแอดเดรสบรอดคาสต์เท่ากับ 202.105.100.255

3.) แอดเดรสซับเน็ตที่สงวนไว้ เมื่อมีการแบ่งซับเน็ตเลขซับเน็ตที่มีค่า '0' ทุกบิต (เรียกเลขซับเน็ตนี้ว่า All-0S ซับเน็ต) และ 1 ทุกบิต (เรียกเลขซับเน็ตนี้ว่า All-1S ซับเน็ต) จะสงวนไว้ไม่ให้นำมาใช้กำหนดแอดเดรส

3.) แอดเดรสกำหนดโฮสต์

แอดเดรส 0.0.0.0 หมายถึงตัวโฮสต์เอง แอดเดรสนี้สงวนไว้ใช้สำหรับโฮสต์ในการเริ่มต้นระบบด้วยโปรโตคอลยูดีพีและไม่อนุญาติให้ใช้เป็นไอพีแอดเดรสปลายทาง นอกจากนี้แอดเดรส 0.0.0.0 ยังมีความหมายเฉพาะในการเลือกเส้นทางโฮสต์หรือเราเตอร์ใช้แอดเดรสนี้เป็น ดีฟอลต์เราท์(Default Route) หรือเส้นทางโดยปริยาย

4.) แอดเดรสลูปแบ็ค (Loop Back)

แอดเดรส 127.X.X.X ใช้เป็นแอดเดรสเพื่อตรวจสอบการทำงานของตนเองเชื่อมเข้าสู่อินเตอร์เฟสลูปแบ็ค ค่า X มีค่าเท่าใดก็ได้ แต่โดยปกติแล้วจะนิยมใช้แอดเดรส 127.0.0.1 [1]

2.2.7 พอร์ต (Port)

เนื่องจากในเวลาใดๆ สามารถมีโพรเซสของผู้ใช้สามารถใช้ยูดีพีหรือทีซีพีได้พร้อมกันดังนั้นจึงต้องมีวิธีแยกแยะว่าข้อมูลเป็นของผู้ใช้คนใด ซึ่งวิธีที่ทีซีพีและยูดีพีใช้คือการใช้หมายเลขพอร์ต (Port Number) เมื่อโพรเซสไคลเอ็น (Client Process) ต้องการที่จะติดต่อกับเซิร์ฟเวอร์ไคลเอ็นจะต้องเจาะจงเซิร์ฟเวอร์ที่ต้องการติดต่อ แต่ลำพังรู้แอดเดรสอินเตอร์เน็ต 32 บิต เพียงอย่างเดียวนั้นไม่พอ เพราะว่าสามารถติดต่อกับโฮสต์ได้เพียงอย่างเดียวแต่ไม่สามารถเจาะจงโพรเซสที่จะทำการติดต่อได้ ดังนั้นเพื่อแก้ปัญหานี้ทั้งทีซีพีและยูดีพีได้มีการกำหนด หมายเลขพอร์ตมาตรฐาน (Well-know Port) ซึ่งเป็นที่รู้จักกัน เช่น ทุกๆระบบ TCP/IP ที่มีเซิร์ฟเวอร์ FTP (File Transfer Protocol) จะมีหมายเลขพอร์ตเป็น 21 เป็นต้น เมื่อทีซีพีหรือยูดีพี กำหนดหมายเลขพอร์ตที่ไม่ซ้ำกันให้โพรเซสของผู้ใช้ เราเรียกหมายเลขพอร์ตนี้ว่าหมายเลขพอร์ตชั่วคราว (Ephemeral Port

Numbers) เมื่อไคลเอ็นเลิกใช้หมายเลขพอร์ตนี้แล้ว TCP/UDP จะสามารถกำหนดหมายเลขพอร์ตนี้ให้ไคลเอ็นอื่นได้ โพรเซสที่ได้รับหมายเลขพอร์ตชั่วคราวนี้จะไม่สนใจว่ามีค่าเท่าใดแต่เป็นหน้าที่ของอีกโพรเซสหนึ่งที่ต่อกันที่ต้องสนใจ เพราะต้องส่งข้อมูลกลับมาที่พอร์ตนี้ การทำงานของทีซีพีและยูดีพีนั้นหมายเลขพอร์ตตั้งแต่1-1023 เป็นหมายเลขพอร์ตที่สงวนไว้สำหรับโปรแกรมประยุกต์มาตรฐาน

2.3 Winsock (Window Socket Application Programming Interface)

Winsock เป็นอินเตอร์เฟสของซ็อคเก็ต (Socket) แบบมาตรฐานของระบบปฏิบัติการ Window ที่ใช้สำหรับโปรแกรมประยุกต์ของเครือข่ายที่มีการใช้โปรโตคอล TCP/IP ซึ่งมีรากฐานเดียวกันกับซ็อคเก็ตของยูนิกส์ตระกูล BSN (Berkeley Software Distribution) จากมหาลัยแคลิฟอร์เนีย วิทยาเขต Berkeley สำหรับโปรโตคอลที่ใช้ใน Winsock จะมี 2 โปรโตคอลด้วยกันคือ Winsock TCP (Transmission Control Protocol) และ Winsock UDP (User Datagram Protocol) โดยการทำงานแบบ Winsock TCP จะคล้ายกับการติดต่อทางโทรศัพท์คือจะต้องมีการเชื่อมต่อกันก่อนจึงจะติดต่อกันได้ และการส่งข้อมูลจะมีการตรวจสอบด้วยว่าผู้รับได้รับข้อมูลครบถูกต้องหรือไม่ถ้าไม่ถูกต้องหรือข้อมูลที่ส่งไปขาดหาย ผู้ส่งก็จะทำการส่งข้อมูลนั้นไปใหม่จนกว่าผู้รับจะได้รับข้อมูลที่ถูกต้อง ส่วนการทำงานแบบ Winsock UDP จะเป็นแบบ Connectionless คือจะไม่มีการเชื่อมต่อกันและไม่มีการตรวจสอบข้อมูลด้วย ว่าผู้รับได้รับข้อมูลที่ส่งไปไหม ซึ่งก็เหมือนกับสถานีส่งคลื่นวิทยุหรือการส่งจดหมายธรรมดานั่นเอง

2.3.1. โพรซิเยอร์เหตุการณ์ในวินซ็อค (Winsock Procedure)

วินซ็อคมีโพรซิเยอร์สำหรับกระทำติดต่อสื่อสารทั้งทางฝ่าย Server และ Client ซึ่งสามารถแบ่งออกได้ดังนี้ คือ

- Close เป็นเหตุการณ์เมื่อมีหยุดหรือยกเลิกการติดต่อสื่อสารของฝ่าย Server หรือ Client
- Connect เป็นเหตุการณ์ที่ฝ่าย Client มีการส่งสัญญาณติดต่อกับมายัง Sever ส่งผลให้โพรซิเยอร์นี้ของฝ่าย Server ทำงานขึ้นมา
- ConnectionRequest เป็นเหตุการณ์เมื่อฝ่าย Client ส่งสัญญาณติดต่อกลับมายัง Server โพรซิเยอร์ส่วนนี้ก็จะทำงานพร้อมกับค่า RequestID ซึ่งเป็นแฮนเดิลของการเชื่อมต่อที่ส่งมาจากระบบ โดยจะให้ฝ่าย Server รับรู้ว่าใช้ ID จากคอนโทรลตัวใด เพื่อจะได้สื่อสารถึงกันในการเชื่อมต่อสามารถทำได้เพียงการเชื่อมต่อเดียวในเวลาหนึ่งเท่านั้น

- DataArrival เหตุการณ์นี้เกิดขึ้นเมื่อมีการส่งข้อมูลระหว่าง Server และ Client โพรซิเยอร์นี้ก็จะทำงานขึ้นมา พร้อมกับค่าจำนวนไบต์ที่รับเข้ามา

- Error เป็นเหตุการณ์ที่เกิดขึ้นเมื่อเกิดความผิดพลาดระหว่างการติดต่อสื่อสารระหว่าง Server และ Client โดยจะส่งค่าหมายเลขที่เกิดความผิดพลาด พร้อมทั้งรายละเอียดของการผิดพลาดในเหตุการณ์นั้นๆ

- SendProgress จะเกิดขึ้นในขณะที่มีการส่งข้อมูลอยู่ ซึ่งจะบอกขนาดไบต์ที่กำลังทำการส่งอยู่และขนาดไบต์ที่เหลืออยู่

- SendComplete เหตุการณ์เมื่อมีการส่งข้อมูลออกไปยังฝ่ายตรงข้ามเสร็จเรียบร้อยแล้ว

2.3.2 คุณสมบัติและวิธีของวินซ็อค (Winsock Properties & Method)

- Accept (requestID) คือการตกลงกันระหว่าง Server และ Client ในการเลือกหมายเลข ID Control ให้ตรงกันเพื่อสามารถสื่อสารได้ถูกต้อง

- Close เป็นการส่งสัญญาณยกเลิกการติดต่อระหว่างกัน จะเป็นฝ่าย Server หรือ Client ก็ได้ ซึ่งจะทำให้โพรซิเยอร์โคลสในฝ่ายตรงข้ามทำงาน

- Connect เป็นการส่งสัญญาณว่าตอนนี้ทำการติดต่อได้เรียบร้อยแล้ว ซึ่งจะส่งผลให้ Procedure ฝ่ายตรงข้ามทำงาน

- Getdata เป็นการรับข้อมูลเมื่อฝ่ายตรงข้ามส่งมาโดยประโยคคำสั่งนี้จะอยู่ในส่วนของ Procedure DataArrival เนื่องจากเป็นเหตุการณ์ที่การกระทำขณะเมื่อฝ่ายตรงข้ามส่ง ข้อมูลเข้ามา

- Listen การกระทำที่จะคอยตรวจสอบสัญญาณการเชื่อมต่อที่ส่งไปว่าฝ่ายตรงข้ามตอบรับการร้องขอการติดต่อ

- LocalHostName คำสั่งนี้จะส่งชื่อของ Computer name ของเครื่องนั้นๆ

- LocalIP คำสั่งนี้จะทำการส่งหมายเลข IP Address

- LocalPort คำสั่งที่จะส่งค่าของหมายเลขในการติดต่อ TCP/IP ของเครื่องนั้นๆ

- RemoteHost กำหนดหรือคืนค่าชื่อ Computer name ของเครื่องที่จะทำการติดต่อ

- RemoteHostIP กำหนดหมายเลข IP Address ของเครื่องที่จะทำการติดต่อ

- RemoteHostPort กำหนดหมายเลข Port ที่จะใช้ในการติดต่อระหว่างกัน

- SocketHandle จะคืนค่าของช่องทางที่ใช้ในการติดต่อระหว่างกัน

- State จะคืนค่าของสถานะของ Socket ขณะที่ใช้ติดต่อระหว่างอยู่ โดยอาจจะใช้ตรวจสถานะ โดยค่าคงที่เหล่านี้เช่น sckClosed (มีค่า=0) Socket ปิดการใช้งาน, sckOpen (มีค่า= 1) Socket เปิดใช้งาน หรือ sckError(มีค่า = 9) Socket มีความผิดพลาดเกิดขึ้น เป็นต้น [4]

2.4 GDI และ Device Context

วินโดวส์เป็นระบบปฏิบัติการที่มีส่วนติดต่อกับผู้ใช้ในแบบกราฟิกส์ นั่นหมายความว่าตัววินโดวส์เองก็ต้องมีฟังก์ชั่นที่ใช้สำหรับการวาดภาพกราฟิกส์เหล่านี้ GDI (Graphics Device Interface) คือกลุ่มของฟังก์ชั่นและโครงสร้างข้อมูลที่มีไว้ให้วินโดวส์และโปรแกรมต่าง ๆ ใช้ในการแสดงผลกราฟิกส์ไม่ว่าจะเป็นทางจอภาพ, เครื่องพิมพ์หรือดีไวซ์อื่น ๆ ฟังก์ชั่น GDI จะมีมาพร้อมกับวินโดวส์ทุกรุ่น และเนื่องจาก Windows NT และ Windows 95 เป็นต้นมา ต่างก็เป็นวินโดวส์ในแบบ 32 บิต ดังนั้นตัว GDI ในวินโดวส์เหล่านี้จึงถือว่าเป็น GDI32 ซึ่งเป็นการบ่งบอกว่าเป็นฟังก์ชั่นกราฟิกส์สำหรับวินโดวส์ 32 บิต เราสามารถใช้ฟังก์ชั่น GDI ในการวาดเส้นตรง, เส้นโค้ง, รูปปิด, ข้อความ, รูปบิตแมป ฯลฯ

โปรแกรมสามารถติดต่อกับดีไวซ์แต่ละตัวได้โดยการสร้างดีไวซ์คอนเท็กซ์ (DC - Device Context) ซึ่งดีไวซ์คอนเท็กซ์เป็นโครงสร้างข้อมูลที่ GDI ใช้จัดการกับรายละเอียดของดีไวซ์นั้นๆ เช่น รูปแบบในการดำเนินการ, สิ่งที่ถูกเลือกในขณะนั้น โดยโปรแกรมต่าง ๆ สามารถสร้างดีไวซ์คอนเท็กซ์ได้จากฟังก์ชั่นที่ใช้จัดการด้านนี้ ซึ่ง GDI จะส่งค่าแฮนเดิลของดีไวซ์แต่ละตัวกลับมา และค่าแฮนเดิลนี้เองที่ใช้ในการแยกแยะดีไวซ์คอนเท็กซ์แต่ละตัว (ค่าแฮนเดิลเป็นตัวเลขชนิด Long ที่กำหนดขึ้นโดยวินโดวส์ และไม่ได้นำไปใช้คำนวณ)

เมื่อแอพพลิเคชั่นที่รันบนวินโดว์(หรือแม้แต่วินโดว์เอง)จะวาดข้อความ หรือวาดรูปภาพเพื่อแสดงผลไม่ว่าจะเป็นจอภาพหรือเครื่องพิมพ์ โดยปกติแล้ววินโดว์ไม่ได้วาดโดยตรงกับฮาร์แวร์เหมือนกับโปรแกรมดอสทำ ทั้งนี้เพราะถูกห้ามไว้ด้วยระบบของวินโดว์

แอพพลิเคชั่นจะใช้ดีไวซ์คอนเท็กซ์หรือเรียกย่อๆ ว่า ดีซี (DC) ดีไวซ์คอนเท็กซ์เป็นโครงสร้างที่กำหนดเซตของออบเจ็กต์กราฟิก และแอตทริบิวต์ (Attributes) ต่างๆ รวมทั้งโหมดกราฟิกที่มีผลต่อเอาท์พุท ดังนั้น ดีซี จึงถูกนำมาใช้อย่างกว้างขวางในวินโดว์ ซึ่งก็คือ กราฟิก ดีไวซ์อินเตอร์เฟส (Graphic Device Interface) ซึ่งเรียกย่อๆ ว่า GDI โดย GDI นั้นจะมีฟังก์ชั่น วินโดว์เอพีไอจำนวนมากให้ใช้งาน ในการใช้งานจะผ่านค่าพารามิเตอร์ต่างๆ จาก GDI ซึ่ง GDI จะนำไปเรียกใช้ดีไวซ์ ไดร์วเวอร์ (Device Driver) ที่เหมาะสมกับอุปกรณ์ชนิดต่างๆ เพื่อให้แสดงได้กราฟิกได้อย่างถูกต้อง

ดีไวซ์คอนเท็กซ์ไม่ถูกจำกัดเพียงแต่อุปกรณ์ที่มีอยู่จริงจับต้องได้ ซึ่งเรียกฟิสิกส์คอลดีไวซ์ (physical device) เท่านั้น แต่ดีไวซ์คอนเท็กซ์สามารถอ้างถึงอุปกรณ์ของโลจิกคอลดีไวซ์ เช่น เมต้าไฟล์(metafile) ซึ่งเป็นกลุ่มของโครงสร้างที่เก็บรูปภาพในฟอร์แมตที่อิสระต่อดีไวซ์ อีกตัวอย่างหนึ่งคือ บิตแมพ(bitmap)

ฟังก์ชั่น GDI32 ถึงแม้จะไม่มีการทำงานที่รวดเร็วเหมือนกับ DirectX หรือ OpenGL รวมทั้งไม่มีเอฟเฟ็กต์ให้ใช้มากนัก และยังไม่มีส่วนจัดการกับกราฟิกส์ 3 มิติโดยตรง แต่ก็มีความง่ายในการใช้งาน และยังทำงานบนเครื่องผู้ใช้อย่างไม่มีปัญหาและไม่ต้องติดตั้งอะไรเพิ่มเติม ซึ่งจริงๆ แล้ว GDI32 ยังมีฟังก์ชั่นที่น่าสนใจอีกมาก โดยที่สามารถนำมาใช้งานได้อย่างหลากหลายไม่เฉพาะกับเกมเท่านั้น ยังรวมไปถึงโปรแกรมต่างๆ ด้วย ซึ่งสามารถศึกษาเพิ่มเติมได้จาก MSDN Library

2.4.1 การสร้างและใช้งานดีไวซ์คอนเท็กซ์

วินโดวส์ได้เตรียม DC มาตรฐานอยู่ภายใน(build-in) หน่วยความจำที่เรียกว่า pool เพื่อให้สามารถใช้ในการวาดกราฟิกต่างๆ ลงในพื้นที่แสดงผลของหน้าต่างหรือดีไวซ์ต่างๆได้ โดยถ้าต้องการวาดกราฟิกลงในพื้นที่ของหน้าต่าง หรือดีไวซ์ก็ต้องอ้างอิงถึง DC ตัวใดตัวหนึ่งจาก pool และกำหนดค่าแอตทริบิวส์ตามที่ต้องการ เช่น ขนาดหรือสีของเส้นเป็นต้นให้กับ DC จากนั้นจึงเรียกใช้ฟังก์ชั่นวินโดวส์ API ด้านกราฟิกเพื่อวาดกราฟิกและเมื่อไม่ต้องการใช้ DC อีกต่อไปก็ต้องทำการยกเลิกการถือครอง DC ดังกล่าว เพื่อให้ DC ที่ถูกยกเลิกการถือครองได้ถูกส่งกลับไปยัง pool และเป็น DC ที่ว่างสำหรับถูกเรียกใช้งานโดยแอพพลิเคชั่นตัวอื่นๆ ต่อไปและนอกจากนี้ถ้าหากไม่พอใจกับ DC มาตรฐานที่วินโดวส์ได้เตรียมไว้ให้ ผู้อ่านก็สามารถที่จะสร้าง DC ขึ้นมาใหม่ให้คอมแพททิเบิลกับ DC มาตรฐานเพื่อนำมาใช้ส่วนตัวสำหรับแต่ละแอพพลิเคชั่นหนึ่งๆได้เช่นกัน

ออบเจ็กต์ DC ไม่ว่าจะเป็น DC มาตรฐานหรือที่ถูกสร้างขึ้นมาโดยฟังก์ชั่นวินโดวส์ CreateDC ก็ตาม จะต้องมีหมายเลข handle ประจำตัวเสมอ เพื่อให้สามารถใช้กำหนดการเรียกใช้ DC โดยฟังก์ชั่นวินโดวส์ API ได้อย่างถูกต้อง สำหรับตัวของออบเจ็กต์ DC ก็เป็นข้อมูลโครงสร้างภายในของวินโดวส์ที่มีขนาดประมาณ 800 ไบต์ โดยที่วินโดวส์ได้เตรียมวิธีการในการสร้างและเข้าถึง DC โดยขึ้นกับลักษณะของ DC ดังนี้

Private Device Context

เมื่อหน้าต่างมีคลาสเป็น CS_OWNDC CS_PARENTDC หรือ CS_CLASSDC หน้าต่างเหล่านี้จะหมายถึง Private Device Context โดยกรณีของคลาส CS_OWNDC แต่ละหน้าต่างก็จะมี DC เป็นของตัวเอง ซึ่งฟอร์มและคอนโทรล Picturebox ของ Visual Basic 6.0 ก็จัดอยู่ในคลาสของ CS_OWNDC เช่นกัน เราสามารถหาหมายเลขของ handle ของฟอร์มและคอนโทรล Picturebox ได้จากคุณสมบัติ hDC สำหรับข้อดีของคลาส CS_OWNDC จะเป็นของออบเจ็กต์ที่สร้างคลาสขึ้นมาเป็นการเฉพาะ และวินโดวส์จะไม่ทำการแก้ไข DC ไม่ว่ากรณีใดๆ ก็ตาม ยกเว้นออบเจ็กต์ที่สร้างคลาสจะเป็นผู้ดำเนินการเอง สำหรับคลาส CS_PARENTDC ก็จะกำหนดให้หน้าต่างมี DC ที่เป็นเจ้าของโดยหน้าต่างแม่(parent window) ของตัวเองและสุดท้ายคลาส CS_CLASSDC ก็จะกำหนด

ให้หน้าต่างมี DC เพียงตัวเดียวที่ถูกแบ่งใช้งานโดยหน้าต่างทั้งหมดที่อยู่ในคลาสเดียวกัน สำหรับการอ่านหมายเลข handle ของ Private Device Context สามารถกระทำได้โดยการใช้ฟังก์ชั่นวินโดวส์ GetDC หรือ GetDCEx

Cached Device Context

Cached Device Context สามารถเรียกอีกชื่อหนึ่งว่า Build-in Device Context ซึ่งวินโดวส์ได้เตรียม Cached Device Context เอาไว้ภายในและสามารถเรียกใช้ได้โดยแอพพลิเคชั่นทั้งหมดที่กำลังทำงานในปัจจุบัน และเนื่องจาก Cached Device Context เป็น DC มาตรฐานของวินโดวส์ ดังนั้นจึงไม่สามารถทำลายได้ และที่สำคัญการใช้งาน DC นั้นจำเป็นต้องมีการอ้างอิงถึงเสียก่อนโดยฟังก์ชั่นวินโดวส์ GetDC หรือ GetDCEx ดังนั้นก่อนที่จะจบการทำงานของโปรแกรม จึงต้องเขียนโค้ดเพื่อยกเลิกการอ้างอิงด้วยฟังก์ชั่นวินโดวส์ ReleaseDC ทุกครั้ง เพราะถ้าหากโปรแกรมจบการทำงานในขณะที่ยังถือครอง DC มาตรฐานเอาไว้ จะทำให้ทรัพยากรของระบบหมดไป วินโดวส์อาจจะหยุดการทำงานได้ทันที สำหรับการอ่านหมายเลข handle ของ Cached Device Context สามารถกระทำได้โดยฟังก์ชั่น GetDC หรือ GetDCEx เช่นเดียวกับ Private Device Context

Created Device Context

Created Device Context เป็น DC ที่ถูกสร้างขึ้นมาใหม่โดยโค้ดของโปรแกรม ซึ่งสามารถสร้าง DC สำหรับแอพพลิเคชั่นๆได้โดยใช้ฟังก์ชั่นวินโดวส์ CreateDC หรือ CreatCompatibleDC โดยในทางทฤษฎีเราสามารถสร้าง DC ประเภทนี้ได้มากเท่าที่ต้องการ ไม่มีข้อจำกัด แต่ในทางปฏิบัติจำนวนของ Created Device Context จะถูกจำกัดโดยทรัพยากรของระบบ สำหรับหมายเลข handle ของ Created Device Context จะถูกรายงานโดยฟังก์ชั่นวินโดวส์ CreatDC หรือ CreatCompatibleDC เมื่อการสร้าง Created Device Context ประสบผลสำเร็จ

เราสามารถกำหนดค่าแอดทริบิวส์ของ DC ได้โดยการใช้ฟังก์ชั่น API ซึ่งในการเขียนจริงจะต้องมีการเลือกแอดทริบิวส์ให้กับ DC ก่อนที่จะใช้งานเสมอซึ่งที่ต้องเลือกให้กับ DC สำหรับการวาดกราฟิกมีดังนี้

Drawing object

การเลือก Drawing object ให้กับ DC เช่น การเลือก Pen ให้กับ DC เป็นต้น สามารถกระทำได้โดยการใช้ฟังก์ชั่นวินโดวส์ SelectObject ซึ่งฟังก์ชั่นจะส่งกลับหมายเลข handle ของออบเจ็กต์ที่อยู่ใน DC ก่อนที่จะถูกแทนที่ และที่สำคัญก่อนที่จะยกเลิก DC จะต้องแทนที่ออบเจ็กต์ที่เลือกให้กับ DC ด้วยออบเจ็กต์ต้นฉบับ(origin object) โดยใช้หมายเลข handle ที่ได้จากฟังก์ชั่น SelectObject ทุกครั้ง

Drawing attribute and Mapping mode

แอดทริบิวส์สำหรับการวาดภาพ เช่น drawing mode หรือ intercharacter spacing สามารถกำหนดโหมดของการทำ mapping และระบบของโคออร์ดิเนต สำหรับการวาดภาพเอาไว้อีกด้วย สำหรับค่าของแอดทริบิวส์ในกลุ่มนี้ไม่จำเป็นต้องคืนค่ากลับให้ DC

Color palette

สำหรับ Visual Basic 6.0 สนับสนุนการใช้สีแบบ 24 บิต(true color) สำหรับใช้ในการวาดกราฟิก ซึ่งมีความคอมแพททิเบิลกับสีที่ได้จากฟังก์ชั่นวินโดวส์ API ต่างๆที่เกี่ยวข้องกับการกำหนดสีของคอนโทรลหรือฟอร์มได้ โดยอาศัยไฟล์ .dib ที่มากับ Visual Basic หรือโปรแกรมประเภทที่สามารถสร้างถาดสีได้

DC ที่ได้จากการสร้างและการอ้างอิง จะมีวิธีการเขียนโค้ดที่ต่างกันเพื่อไม่ให้สับสนจะอธิบายพอสังเขป เกี่ยวกับฟังก์ชั่น API ที่ใช้ในการสร้างและการอ้างอิงไปยัง DC ชนิดต่างๆดังนี้

การสร้าง DC

DC ที่จะสร้างขึ้นมาใหม่นั้นจะต้องเป็น DC ประเภท Created Device Context ซึ่งการสร้าง DC ขึ้นมาใหม่โดยฟังก์ชั่น CreatDC หรือ CreatCompatibleDC ก็หมายถึง การสร้างข้อมูลโครงสร้าง DC ขึ้นมาในหน่วยความจำ สำหรับการใช้งานเฉพาะของออบเจ็กต์ที่สร้าง DC ขึ้นมา และ Created Device Context ที่ถูกสร้างขึ้นมาจะไม่ถูกทำลาย ดังนั้นเมื่อสิ้นสุดการใช้งานจะต้องใช้ฟังก์ชั่น DestroyDC เพื่อทำลาย Created Device Context ที่ถูกสร้างขึ้นมาเสมอ ซึ่งทรัพยากรของระบบก็จะสามารถถูกนำไปใช้โดยแอพพลิเคชั่นอื่นๆต่อไป

การอ้างอิงถึง DC

DC ที่จะสามารถถูกอ้างอิงเพื่อถือครองจะต้องเป็น DC ประเภท Private Device Context หรือ Cached Device Context ซึ่งวินโดวส์ได้เตรียม DC ดังกล่าวเอาไว้ใน pool โดยอัตโนมัติซึ่งการอ้างอิงไปยัง DC ใน pool สามารถกระทำได้โดยการใช้ฟังก์ชั่น GetDC หรือ GetDCEx ดังนั้น เมื่อสิ้นสุดการใช้งานหรือก่อนแอพพลิเคชั่นจะสิ้นสุดการทำงานเราไม่สามารถทำลาย DC ดังกล่าวได้ เพราะเป็นของระบบ แต่ในทางกลับกันจะต้องเขียนโค้ดเพื่อยกเลิกการอ้าง DC ด้วยฟังก์ชั่น ReleaseDC ทุกครั้งเพื่อให้ DC ถูกคืนกลับไปยัง pool เพราะถ้าโปรแกรมจบการทำงานในขณะที่ยังถือครอง DC เอาไว้จะทำให้ทรัพยากรระบบหมดไปแล้ววินโดวส์จะหยุดการทำงานได้ทันที [5]

2.5 การบีบอัดหรือลดขนาดข้อมูล

หลักการของการบีบอัดข้อมูลคือ การกำจัดข้อมูลที่ซ้ำๆกันออกไปหรือ การเก็บข้อมูลบางอย่างที่เกะกะ ให้มีรูปแบบที่กระชับและสั้นลง ซึ่งจะส่งผลให้ขนาดของไฟล์นั้นๆ มีขนาดที่เล็กลงกว่าเดิม และเมื่อเราต้องการใช้งานข้อมูลนั้นๆ กระบวนการบีบอัดก็จะทำคลายหรือย้อนหรือคำนวณย้อนกลับออกมาเป็นไฟล์ ที่มีลักษณะเหมือนกับต้นฉบับได้ ข้อมูลที่นำมาบีบอัดจะเป็นอะไรก็ได้ ตั้งแต่ ไฟล์ตัวหนังสือ text ธรรมดา แต่ที่นิยมกันมากก็คือไฟล์ประเภทรูปภาพ

การบีบอัดข้อมูลแบ่งออกเป็น 2 ลักษณะ คือ แบบที่ไม่มีการสูญเสียรายละเอียด (Lossless) กับแบบที่มีการสูญเสียรายละเอียด (Lossy)

1.) Lossless compression การบีบอัดข้อมูลแบบนี้ ถ้านำภาพต้นฉบับมาบีบอัด และเมื่อคลายกลับออกมาแล้ว ภาพจะไม่มีการสูญเสียรายละเอียดต่างๆ ไปแม้แต่น้อย แต่ในทางปฏิบัติจะเห็นได้ว่า การบีบอัดข้อมูลแบบนี้สามารถย่อข้อมูลลงไปได้น้อยมากอย่างเก่งก็ไม่เกินครึ่งหนึ่งของไฟล์ต้นฉบับ แต่โดยส่วนใหญ่จะย่อลงมาได้แค่ 10% เท่านั้นเอง รูปแบบนี้ใช้กันในไฟล์นามสกุล LZW (Lempel-Ziv-Welch) หรือไฟล์ GIF และ TIFF

2.) Lossy compression แม้เราสามารถเลือกการบีบอัดข้อมูลแบบไม่มีการสูญเสียรายละเอียดได้ก็จริง แต่ในทางปฏิบัติบางครั้งเราต้องการไฟล์ที่มีขนาดเล็กมากกว่า เพราะรายละเอียดบางอย่างในภาพเรายอมลดลงไปได้บ้างโดยไม่รู้สึกถึงความแตกต่างอะไร เนื่องจากสายตาของมนุษย์จะไม่สามารถแยกความแตกต่างกันได้มากนัก โดยเราสามารถจะเลือกระดับได้ว่าจะให้บีบข้อมูลขนาดไหนในระดับใด เช่น ไฟล์ JPEG (Joint Photographic Experts Group) สามารถย่อข้อมูลภาพได้ตั้งแต่ 10:1 จนถึง 40:1 แต่เราจะไม่สามารถนำวิธีการนี้แบบนี้ไปใช้ลดขนาดข้อมูลประเภทโปรแกรมคอมพิวเตอร์หรือไฟล์ข้อมูลทั่วไปได้ เนื่องจากข้อมูลเหล่านี้หากถอดรหัสกลับมาแล้วไม่เหมือนเดิม 100% ก็จะใช้งานไม่ได้เลย

โดยปกติขนาดของไฟล์จะแปรผันตามคุณภาพของภาพนั้นๆ เพราะภาพที่มีคุณภาพดีคือ มี Resolution สูง นั่นหมายถึงจำนวน Pixel ที่เพิ่มตามไปด้วย ทำให้การจัดเก็บเป็นไฟล์ข้อมูลมีขนาดใหญ่ตามไปด้วยเช่นกัน แม้ภาพที่มีความละเอียดต่ำสุดที่ 640 x 480 pixel ก็ยังมี pixel ทั้งหมด 307,200 pixels และถ้าเก็บที่ color-depth 24-bit (=3 bytes เพราะ 1 byte มี 8 bits) ก็จะทำให้ภาพนี้ มีขนาด 1MB กว่าๆ แล้วถ้าเป็นภาพขนาด 1024 x 768 จะยิ่งมีขนาดใหญ่ ดังนั้นเพื่อให้ได้ภาพที่มีคุณภาพ ในขณะที่ขนาดของไฟล์ไม่ใหญ่จนเกินไปจะอยู่ที่การเลือกรูปแบบหรือ format ของไฟล์ให้ถูกต้องเป็นสำคัญประกอบกับการเลือกใช้เทคโนโลยีการบีบอัดข้อมูล (compressed file)

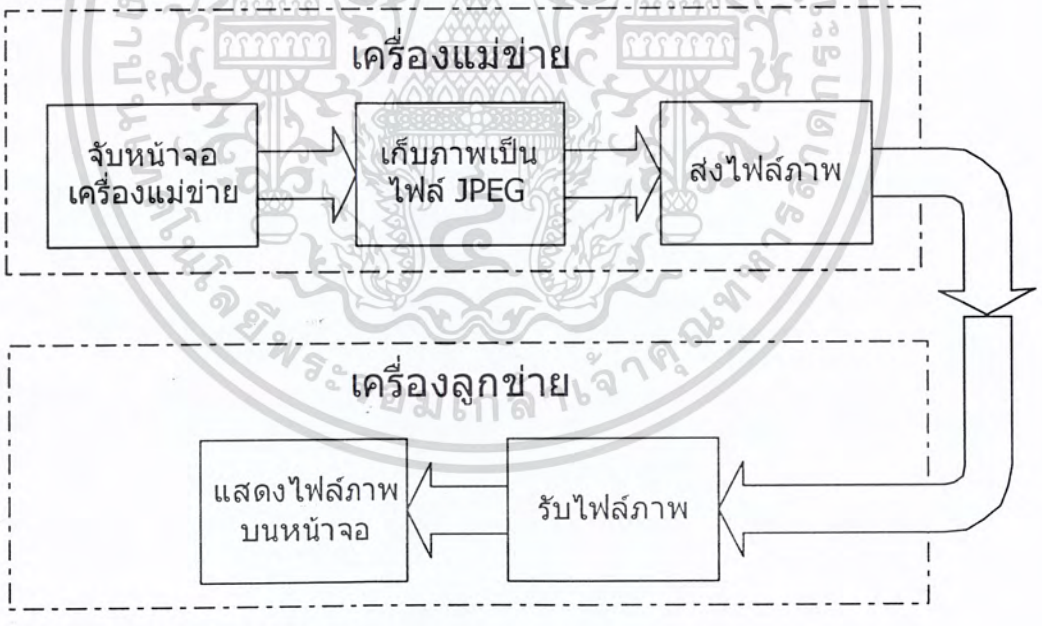
ปัจจุบันรูปแบบไฟล์สำหรับเก็บรูปภาพที่ได้รับการยอมรับว่าเล็กที่สุด ในขณะที่ให้ความยืดหยุ่นในการใช้งานมากที่สุดที่นิยมใช้กันจะเป็นไฟล์ในรูปแบบ JPEG หรือ JPG ย่อมาจากคำว่า

Joint Photographic Experts Group เป็นรูปแบบไฟล์ที่ผสมผสานเทคโนโลยีการบีบอัดข้อมูลโดยที่เราสามารถเลือกได้ว่าจะบีบหรือลดขนาดข้อมูลได้มากน้อยแค่ไหนและถ้าลดขนาดมากเท่าไร คุณภาพของภาพก็จะลดลงมากเท่านั้น ซึ่งก็ขึ้นอยู่กับวัตถุประสงค์ที่เราจะนำภาพไปใช้งานว่าเราจะเน้นที่คุณภาพของรูปหรือขนาดของไฟล์ เช่น ถ้าหากต้องการนำไปใช้บน Web หรือ ส่งทาง E-mail ขนาดของไฟล์ นับว่าเป็นเรื่องสำคัญมากแต่ถ้าต้องการภาพเพื่อนำไปขยายให้ใหญ่ขึ้นมากๆ ก็ควรเลือกเน้นที่คุณภาพเป็นสำคัญ

บทที่ 3 หลักการออกแบบ

3.1 หลักในการออกแบบ

หลักการทำงานของโปรแกรมควบคุมแสดงผลหน้าจอของเครื่องลูกข่ายในระบบเครือข่ายท้องถิ่น เป็นการนำการแสดงผลบนหน้าจอของเครื่องแม่ข่ายหรือหน้าจอเครื่องคอมพิวเตอร์ของผู้สอนส่งไปให้เครื่องลูกข่ายหรือเครื่องคอมพิวเตอร์ของผู้เรียน ซึ่งมีจำนวนหลาย ๆ เครื่อง เพื่อให้มีการแสดงผลบนหน้าจอเหมือนกับเครื่องแม่ข่ายหรือเครื่องคอมพิวเตอร์ของผู้สอน ซึ่งจากหลักการทำงานนี้จะเห็นว่าจะต้องมีโปรแกรม 2 โปรแกรม คือโปรแกรมที่ทำหน้าเป็นเครื่องแม่ข่ายและโปรแกรมที่ทำหน้าที่เป็นเครื่องลูกข่าย โดยโปรแกรมที่ทำหน้าที่บนเครื่องแม่ข่ายจะมีหน้าที่ในการจับหน้าจอของเครื่องแม่ข่ายแล้วส่งไปยังเครื่องลูกข่าย แต่เนื่องจากขนาดของไฟล์ภาพที่ทำการจับหน้าจอนั้นมีขนาดใหญ่ ซึ่งต้องมีการลดขนาดข้อมูลภาพโดยทำการบีบอัดไฟล์ภาพที่จับได้นั้นเป็นไฟล์ JPEG และเครื่องลูกข่ายจะมีหน้าที่รับข้อมูลภาพที่เครื่องแม่ข่ายส่งมาแล้วแสดงบนหน้าจอของเครื่องลูกข่าย ซึ่งสามารถแสดงบล็อกไดอะแกรมการทำงานของโปรแกรมได้ ดังรูปที่ 3.1



รูปที่ 3.1 แสดงบล็อกไดอะแกรม การทำงานของโปรแกรม

จากบล็อกไดอะแกรมในการออกแบบโปรแกรมจะแบ่งออกเป็น 3 ส่วนที่สำคัญคือ

1. การจับหน้าจอของเครื่องแม่ข่าย
2. การลดขนาดไฟล์ภาพหรือการบีบอัดภาพเป็นไฟล์ JPEG
3. การส่ง-รับ ข้อมูลภาพผ่านระบบเครือข่าย

3.1.1 การจับหน้าจอของเครื่องแม่ข่าย

ในการจับหน้าจอของเครื่องแม่ข่ายนั้นจะมีส่วนของดีไวซ์คอนเท็กซ์หรือ DC เข้ามาเกี่ยวข้องเพราะในการวาดรูปหรือแสดงผลใด ๆ ก็ตามภายใต้ระบบปฏิบัติการวินโดวส์ จะต้องถูกกระทำผ่านทางดีไวซ์คอนเท็กซ์ ซึ่งเป็นออบเจ็กต์ตัวหนึ่งของวินโดวส์ที่ช่วยในการจัดการในด้านการจัดการกราฟิก และในระบบปฏิบัติการวินโดว์เองก็จะมีการมองแต่ละส่วนไม่ว่าจะเป็นไฟล์ .dll .vxd .exe หรือ .ocx เป็นเพียงออบเจ็กต์หนึ่งๆของวินโดว์ในการโหลดแต่ละออบเจ็กต์ วินโดว์จะกำหนดหมายเลข ID ให้กับแต่ละออบเจ็กต์โดยที่หมายเลข ID นี้จะไม่ซ้ำกันเลย สำหรับทุกๆหน้าต่างที่ถูกโหลดโดยวินโดว์ก็จะมีหมายเลข handle กำกับให้กับทุกๆหน้าต่าง ซึ่งหมายเลข handle นี่เองที่ใช้สำหรับกำหนดหรืออ้างถึงชนิดของออบเจ็กต์ของวินโดว์ และในการกระทำเกี่ยวกับดีไวซ์คอนเท็กซ์เราจะใช้ฟังก์ชั่น API (Application Programming Imterface) ซึ่ง API นี้เป็นที่เก็บฟังก์ชั่นการทำงานต่างๆ ของวินโดว์ที่แอพพลิเคชั่นต่างๆที่ทำงานอยู่บนวินโดว์สามารถเรียกไปใช้งานเพื่อทำงานอย่างใดอย่างหนึ่งได้ เช่น การวาดรูป การบันทึกข้อมูลลงในไฟล์ การเตรียมพื้นที่ในหน่วยความจำ ฯลฯ เป็นต้น ภายใน API จะประกอบด้วยฟังก์ชั่นและโพซิเยอร์ต่างๆ ซึ่งถูกจัดเก็บในไฟล์นามสกุล DLL ในที่นี้ขอยกตัวอย่างง่ายๆ ในการจับหน้าจอแล้วนำมาแสดงบนฟอร์ม ดังนี้

1. ประกาศฟังก์ชั่น API ที่ต้องการใช้งาน
2. หาขนาดของหน้าจอแสดงผลทั้งหมด
3. ใช้ฟังก์ชั่น GetDesktopWindow เพื่อหาค่า handle ของวินโดว์เดสก์ทอปหรือหน้าจอแสดงผลทั้งหมดของวินโดว์
4. ใช้ฟังก์ชั่น GetWindowDC เพื่อหาค่า handle ของดีไวซ์คอนเท็กซ์ของวินโดว์
5. ใช้ฟังก์ชั่น BitBlt เพื่อคัดลอกรูปภาพที่เก็บอยู่ในดีไวซ์คอนเท็กซ์ของเดสก์ทอปไปเก็บไว้ในดีไวซ์คอนเท็กซ์ของฟอร์ม
6. ใช้ฟังก์ชั่น ReleaseDC เพื่อส่งคืนดีไวซ์คอนเท็กซ์กลับสู่ระบบวินโดว์

ซึ่งแสดงการเขียนโปรแกรมดังโค๊ดข้างล่างนี้

```
' การประกาศฟังก์ชั่น API ที่ใช้งาน
  Private Declare Function GetDesktopWindow Lib "user32" () As Long
  Private Declare Function GetWindowDC Lib "user32" _
    (ByVal hwnd As Long) As Long
  Private Declare Function BitBlt Lib "gdi32" (ByVal hDCDest As Long, _
    ByVal XDest As Long, ByVal YDest As Long, ByVal nWidth As Long, _
    ByVal nHeight As Long, ByVal hDCSrc As Long, ByVal xSrc As Long, _
    ByVal ySrc As Long, ByVal dwRop As Long) As Long
  Private Declare Function ReleaseDC Lib "user32" _
    (ByVal hwnd As Long, ByVal hdc As Long) As Long
'   ฟังก์ชั่นการจับหน้าจอ
  Privat Function CaptureScreen() As Picture
    Dim hWndScr As Long
    Dim hDCSrc As Long
    Dim WidthSrc As Long
    Dim HeightSrc As Long

    'Get Size DesktopWindow
    WidthSrc = Screen.Width \ Screen.TwipsPerPixelX
    HeightSrc = Screen.Height \ Screen.TwipsPerPixelY
    'Get device context for entire window
    hWndSrc = GetDesktopWindow
    'Get device context for entire window
    hDCSrc = GetWindowDC(hWndSrc)
    'Copy device context to Form1
     r = BitBlt(Form1.hdc, 0, 0, WidthSrc, HeightSrc, hDCSrc, _
       LeftSrc, TopSrc, vbSrcCopy)
    ' Release the device context resources back to the system
    r = ReleaseDC(hWndSrc, hDCSrc) ()
  End Function
```

ดังนั้นเมื่อเราเรียกฟังก์ชั่นนี้ก็จะทำการจับหน้าจอของวินโดว์แล้วไปแสดงผลบนฟอร์ม1 นี่ก็เป็นโปรแกรมตัวอย่างการใช้งานเบื้องต้นของดีไวซ์คอนเท็กซ์ในการจับการแสดงผลหน้าจอของวินโดว์

3.1.2 การลดขนาดไฟล์ภาพหรือเก็บภาพเป็นไฟล์ JPEG

ในส่วนของการลดขนาดของไฟล์หรือการบีบอัดไฟล์ภาพเป็น JPEG นั้น ผู้เขียนไม่ได้เขียนโปรแกรมขึ้นเอง ซึ่งในส่วนนี้ได้นำซอร์สโค๊ดที่ได้จากการดาวน์โหลดมาจากเว็ป www.vbAccelerator.com มาทำการศึกษาและดัดแปลงให้สามารถใช้งานร่วมกับโปรแกรมได้

3.1.3 การส่ง-รับ ข้อมูลภาพผ่านระบบเครือข่าย

ในที่นี้เราได้ใช้ Winsock Control ซึ่งเป็น Components หนึ่งของ Visual Basic 6.0 ในการติดต่อส่งและรับข้อมูลผ่านเครือข่ายและเลือกใช้โปรโตคอล UDP เพราะโปรโตคอลนี้สนับสนุนการบรอดคาสต์แอดเดรสและขนาดเฮดเดอร์น้อยกว่าโปรโตคอล TCP แต่ก็มีข้อเสียตรงที่ไม่มีการตรวจสอบข้อมูลว่าข้อมูลที่ส่งออกไปถึงผู้รับไหม สำหรับการติดต่อสามารถทำได้ดังนี้

- บนเครื่องแม่ข่าย

ก่อนอื่นต้องกำหนดช่องที่จะใช้ในการสื่อสารหรือหมายเลขพอร์ตที่จะฟังการติดต่อของเครื่องลูกข่ายจากนั้นก็ทำการไบน์(Bind) หรือการเริ่มการติดต่อ ดังแสดงในโค๊ดข้างล่างนี้

```
Winsock.LocalPort = 3000
Winsock.Bind
```

ในที่นี้กำหนดให้เครื่องแม่ข่ายรอการติดต่อจากเครื่องลูกข่ายที่หมายเลขพอร์ต 3000 เมื่อเครื่องลูกข่ายทำการติดต่อมายังหมายเลขพอร์ตนี้ ก็จะสามารถทำการส่งหรือรับข้อมูลได้ สำหรับการส่งข้อมูลจะเมธอด SendData เมื่อต้องการส่งข้อมูล ดังแสดงในโค๊ดข้างล่างนี้

```
Winsock.SendData DataToSend
```

ในที่นี้ DataToSend คือข้อมูลที่ต้องการจะส่ง และสำหรับการรับข้อมูลเมื่อมีการส่งข้อมูลมาจากเครื่องลูกข่ายหรือเครื่องอื่นที่ทำการติดต่ออยู่จะทำให้เกิดเหตุการณ์ DataArrival ขึ้น เราสามารถรับข้อมูลที่ส่งมาได้จากโพซิเยอร์นี้ โดยใช้เมธอด GetData ดังแสดงในโค๊ดข้างล่างนี้

```
Winsock.GetData DataToRecv
```

ในที่นี้ DataToRecv คือบัฟเฟอร์ที่ทำการรับข้อมูล และเมื่อจะเลิกการติดต่อเครือข่ายก็โดยการใช้เมธอด Close ดังแสดงในโค๊ดข้างล่างนี้

```
Winsock.Close
```

- บนเครื่องลูกข่าย

การที่เครื่องลูกข่ายจะสามารถติดต่อกับเครื่องแม่ข่ายได้เครื่องลูกข่ายจะต้องรู้หมายเลขไอพีแอดเดรสและหมายเลขพอร์ตของเครื่องแม่ข่ายที่จะทำการติดต่อ ข้อสำคัญหมายเลขพอร์ตที่จะติดต่อต้องตรงกับหมายเลขพอร์ตที่เครื่องแม่ข่ายรอฟังการติดต่ออยู่ มิฉะนั้นก็จะทำให้ไม่สามารถทำการติดต่อได้ ซึ่งการติดต่อเครื่องแม่ข่ายสามารถทำได้ ดังโค๊ดข้างล่างนี้

```
Winsock.RemoteHost "169.246.47.1"
Winsock.RemotePort 3000
Winsock.Bind
```

ในที่นี้สมมติว่าเครื่องแม่ข่ายมีหมายเลขไอพีแดรสเป็น "169.246.47.1" และหมายเลขพอร์ตที่เครื่องแม่ข่ายรอการติดต่ออยู่ที่พอร์ต 3000 หลังจากกำหนดดังนี้แล้วก็สามารถที่จะส่งหรือรับข้อมูลได้ ซึ่งมีการใช้งานเหมือนกับเครื่องแม่ข่ายที่ได้กล่าวมาแล้วนั้น แต่ถ้าต้องการส่งข้อมูลในลักษณะบรอดคาสต์แอดเดรสก็ทำได้โดยให้ RemoteHost เป็นหมายเลขบรอดคาสต์แอดเดรสนั้นได้เลย โดยการกำหนดให้เลขโฮสต์เป็น 1 ทั้งหมด นอกจากนี้ยังสามารถตรวจสถานะของการติดต่อได้ โดยตรวจสอบ State ของการติดต่อ ซึ่งจะคืนค่าสถานะของการติดต่อขณะนั้น เช่น sckClosed ยกเลิกการติดต่อ, sckOpen สามารถทำการติดต่อได้, sckError เกิดความผิดพลาดในการติดต่อ เป็นต้น

ข้อจำกัดของ Winsock UDP

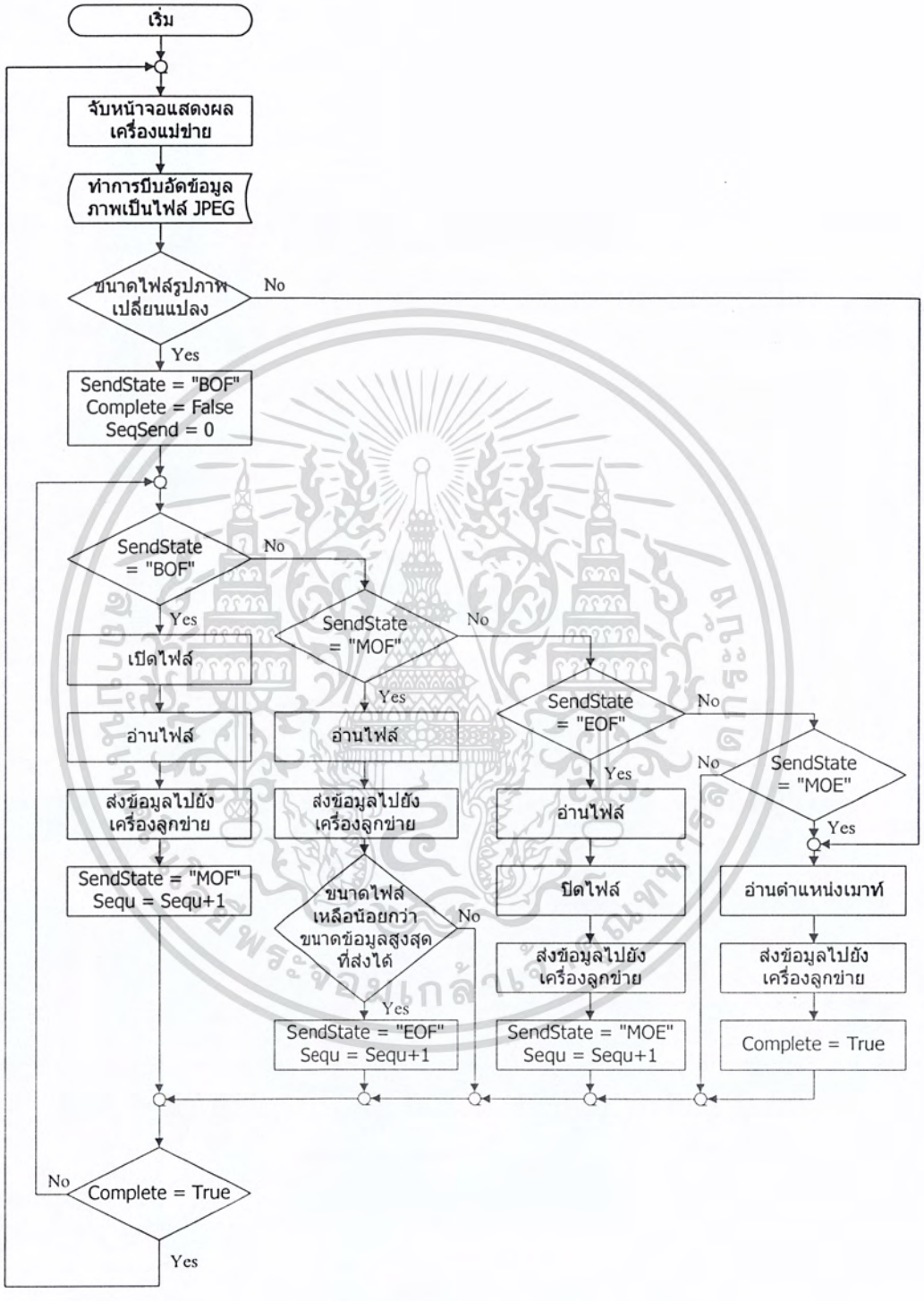
ส่วนข้อจำกัดของ Winsock จากการที่ได้ศึกษาหรือทดลองคือไม่อนุญาตส่งข้อมูลในแต่ละครั้งหรือแต่ละแพ็กเก็ตเกิน 8 KBytes ได้และการส่งข้อมูลในแต่ละครั้งไม่สามารถตรวจสอบได้ว่าการส่งข้อมูลนั้นเสร็จเรียบร้อยหรือยัง ควรจะส่งข้อมูลครั้งต่อไปเมื่อไร ซึ่งจากข้อจำกัดนี้จึงต้อง

เขียนโปรแกรมในการตรวจสอบว่าเครื่องแม่ข่ายส่งข้อมูลออกไปเสร็จเรียบร้อยหรือยังโดย การกำหนดหน้าที่ให้เครื่องลูกข่าย 1 เครื่องทำหน้าที่ในการส่งเฟรมข้อมูลที่ได้รับกลับมายังเครื่องแม่ข่าย การรับข้อมูลที่เครื่องแม่ข่ายส่งออกไปนั้นเข้ามาอีกครั้งหนึ่งเพื่อตรวจสอบขนาดของข้อมูลที่เครื่องแม่ข่ายส่งออกไปว่าตรงกับขนาดของข้อมูลที่เข้าเข้ามาหรือไม่ถ้าข้อมูลมีขนาดเท่ากันก็แสดงว่าเครื่องแม่ข่ายสามารถส่งข้อมูลเสร็จเรียบร้อยแล้ว และจะมีการหน่วงเวลาอีกเล็กน้อยไว้เพื่อให้แน่ใจว่าข้อมูลที่ส่งไปนั้นเสร็จเรียบร้อยแล้วจริงๆ โดยโปรแกรมนั้นไม่มีส่วนตรวจสอบความถูกต้องของข้อมูลแต่อย่างใด

3.2 การเขียนโปรแกรม

สำหรับการเขียนโปรแกรมนั้นเพื่อง่ายต่อการติดตั้งและการใช้งาน ผู้เขียนจึงได้เขียนโปรแกรมทั้งเครื่องแม่ข่ายและเครื่องลูกข่ายไว้ในโปรแกรมเดียวกันซึ่งสามารถเลือกโปรแกรมได้ว่าสามารถที่จะทำงานเป็นเครื่องแม่ข่ายหรือเครื่องลูกข่าย นอกจากนี้ยังสามารถปรับแต่งค่าต่างๆที่ใช้ในโปรแกรมเพื่อประสิทธิและความเร็วในการทำงานของโปรแกรม และโฟลว์ชาร์ตหลักในการทำงานของโปรแกรมสามารถแสดงได้ดังรูปที่ 3.2 และรูปที่ 3.3

3.2.1 เครื่องแม่ข่าย (Server)



รูปที่ 3.2 โฟลว์ชาร์ตแสดงการทำงานเมื่อมีการส่งข้อมูลภาพไปยังเครื่องลูกข่าย 1 ภาพ

จากรูปโฟลว์ชาร์ตแสดงการทำงานของเครื่องแม่ข่ายสามารถอธิบายหลักการทำงานดังนี้ เริ่มแรกเมื่อเริ่มให้โปรแกรมทำงานก็จะทำการจับหน้าจอของเครื่องแม่ข่าย จากนั้นทำการลดขนาดข้อมูลภาพ โดยการบีบอัดภาพที่ได้จากการจับหน้าจอเป็นไฟล์ JPEG และเมื่อบีบอัดภาพเป็นไฟล์ JPEG แล้วก็จะมาเช็คขนาดข้อมูลของภาพที่เปลี่ยนแปลงว่าเปลี่ยนแปลงไปเท่าไหร่ ในการดูขนาดไฟล์ภาพว่าเปลี่ยนแปลงไปเท่าไหร่นี้ก็เพื่อจะดูว่าภาพบนหน้าจอของเครื่องแม่ข่ายมีการเปลี่ยนแปลงไหม เมื่อภาพมีการเปลี่ยนแปลงขนาดของไฟล์ภาพก็ย่อมเปลี่ยนแปลงไปด้วย ดังนั้นถ้าภาพบนจอภาพของเครื่องแม่ข่ายไม่มีการเปลี่ยนแปลงจึงไม่มีความจำเป็นที่จะส่งภาพเดิมนั้นไปอีก ในการเขียนโปรแกรมจึงได้กำหนดขนาดของไฟล์ภาพว่าถ้าไฟล์ภาพเปลี่ยนแปลงไปน้อยกว่าขนาดที่ตั้งไว้ก็จะทำการส่งไปเฉพาะตำแหน่งเมาท์ที่เคลื่อนที่เท่านั้น ซึ่งจะเป็นผลทำให้โปรแกรมทำงานได้เร็วขึ้น แต่ถ้าไฟล์ภาพมีการเปลี่ยนแปลงมากกว่าที่ตั้งไว้ก็จะทำการส่งไฟล์ภาพนั้นไปให้เครื่องลูกข่ายใหม่ และในการส่งไฟล์ภาพนั้นเนื่องจากว่า Winsock UDP สามารถส่งข้อมูลในแต่ละแพ็กเก็ตได้ไม่เกิน 8 KBytes ดังนั้นจึงต้องมาการแบ่งไฟล์ภาพออกเป็นส่วนๆหรือเป็นแพ็กเก็ต ซึ่งในการส่งข้อมูลไปยังเครื่องลูกข่ายจะต้องมีหัวไฟล์บอกเครื่องลูกข่ายด้วยว่าส่งอะไรไปให้เครื่องลูกข่ายซึ่งสามารถแสดงหัวไฟล์ที่ส่งไปยังเครื่องลูกข่ายได้ ดังนี้

ถ้าไฟล์ภาพเป็นส่วนของต้นไฟล์ จะมีการจัดแพ็กเก็ต ดังนี้

S@	ลำดับเฟรม	ลำดับแพ็กเก็ต	BOF	ขนาดไฟล์ภาพที่ส่ง	ข้อมูลภาพ

ถ้าไฟล์ภาพเป็นส่วนของกลางไฟล์ จะมีการการส่งแพ็กเก็ต ดังนี้

S@	ลำดับเฟรม	ลำดับแพ็กเก็ต	MOF	ข้อมูลภาพ

ถ้าไฟล์ภาพเป็นส่วนของท้ายไฟล์ จะมีการส่งแพ็กเก็ต ดังนี้

S@	ลำดับเฟรม	ลำดับแพ็กเก็ต	EOF	ข้อมูลภาพ

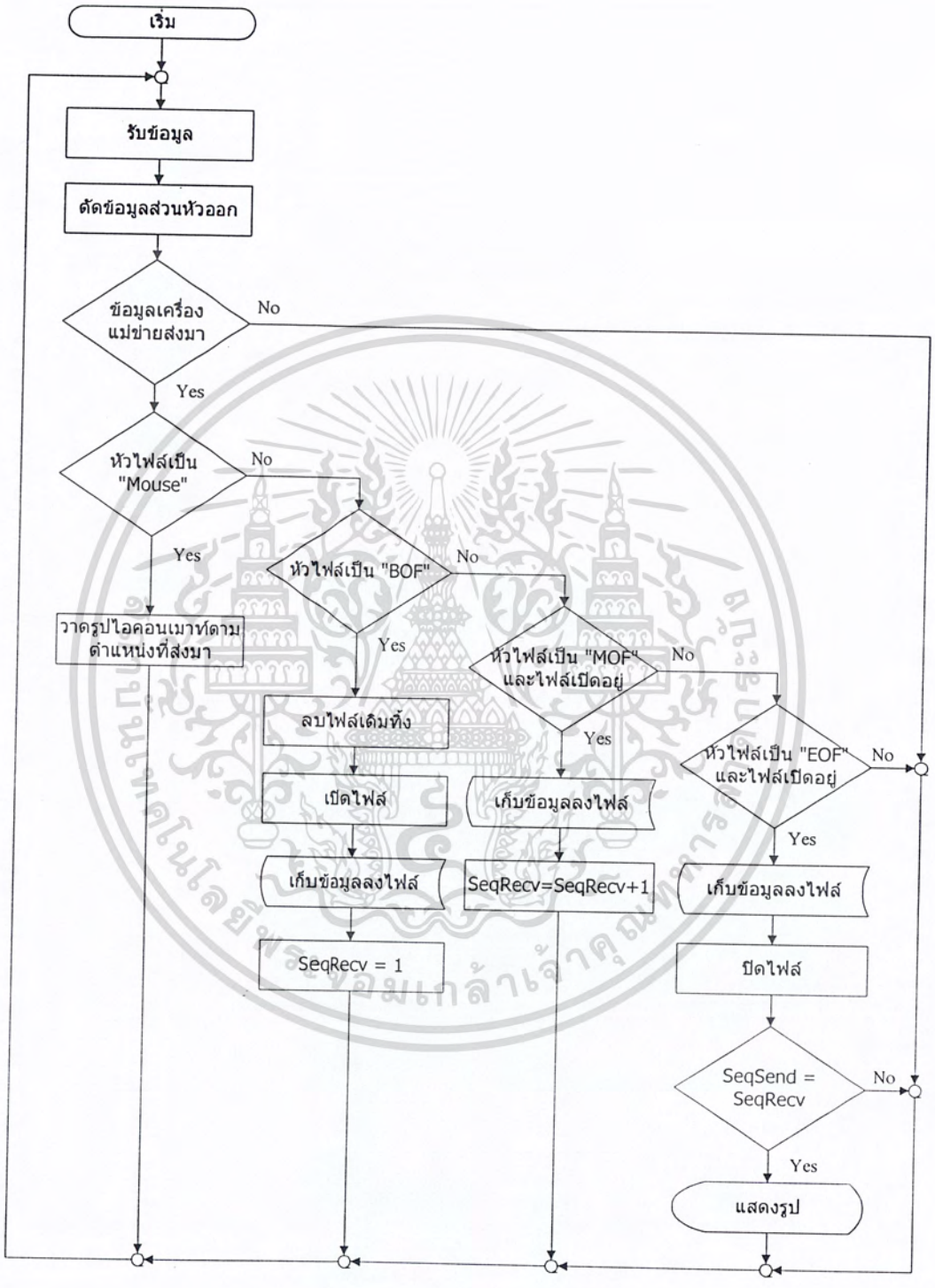
ถ้าเป็นการส่งตำแน่งเมาท์ จะมีการส่งแพ็กเก็ต ดังนี้

S@	Mouse	ตำแหน่ง X	ตำแหน่ง Y	[U] Lock

โดยที่

S@	เป็นการบอกว่าแพ็กเก็ตนี้ส่งมาจากเครื่องแม่ข่าย
ลำดับเฟรม	ลำดับไฟล์รูปภาพที่ส่ง
ลำดับแพ็กเก็ต	ลำดับข้อมูลไฟล์ที่ทำการแบ่งออกเป็นส่วน ๆ
BOF	เป็นการบอกว่าเป็นต้นไฟล์
MOF	เป็นการบอกว่าเป็นช่วงกลางของไฟล์
EOF	เป็นการบอกว่าเป็นท้ายไฟล์
Mouse	เป็นการบอกว่าข้อมูลนี้เป็นตำแหน่งเมาท์
[U] Lock	เป็นการบอกว่าจะให้เครื่องลูกข่ายล็อกหน้าจอหรือไม่

3.2.2 เครื่องลูกข่าย (Client)



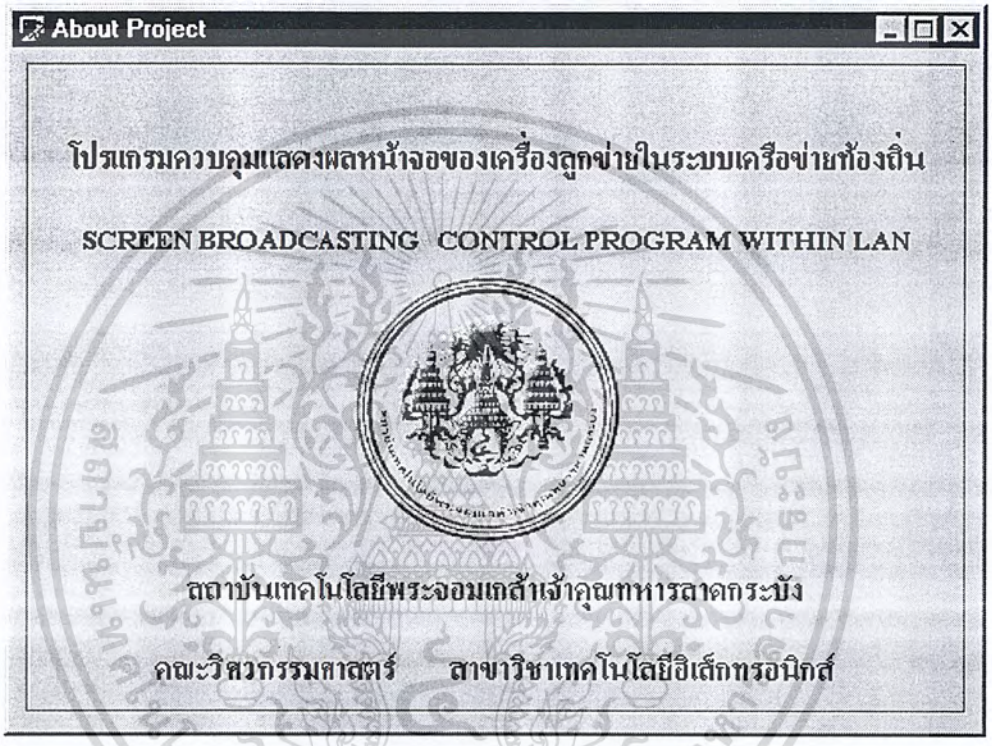
รูปที่ 3.3 โฟลว์ชาร์ตแสดงการทำงานของเครื่องลูกข่ายเมื่อมีการส่งภาพมา 1 ภาพ

จากรูปที่ 3.3 โฟลว์ชาร์ตการทำงานของเครื่องลูกข่ายเมื่อเครื่องลูกข่ายได้รับข้อมูลมาก็จะเช็คก่อนว่าใช่ข้อมูลที่เครื่องแม่ข่ายส่งมาไหมถ้าไม่ใช่ก็จะไม่รับข้อมูลนั้น จากนั้นก็จะตัดส่วนหัวไฟล์ทั้งหมดออก แล้วก็เช็คหัวไฟล์นั้น เช่น ถ้าหัวไฟล์เป็น Mouse ก็จะแสดงตำแหน่งเมาส์ เป็นต้น เมื่อรับไฟล์ภาพได้มาครบหมดแล้วก็จะทำการเช็คอีกว่าไฟล์ภาพที่รับเข้ามานั้นครบตามจำนวนที่เครื่องแม่ข่ายส่งมาหรือไม่ โดยเช็คขนาดไฟล์ภาพที่ส่งมากับขนาดไฟล์ที่รับได้ว่าเท่ากันหรือไม่ และจำนวนแพ็กเก็ตที่รับได้ว่าเท่ากับแพ็กเก็ตที่เครื่องแม่ข่ายส่งมาหรือไม่ ถ้าไม่ครบก็จะไม่แสดงไฟล์รูปภาพนั้นออกมา ซึ่งแสดงว่าไฟล์ภาพนั้นไม่สามารถใช้งานได้

บทที่ 4 การทดลองและผลการทดลอง

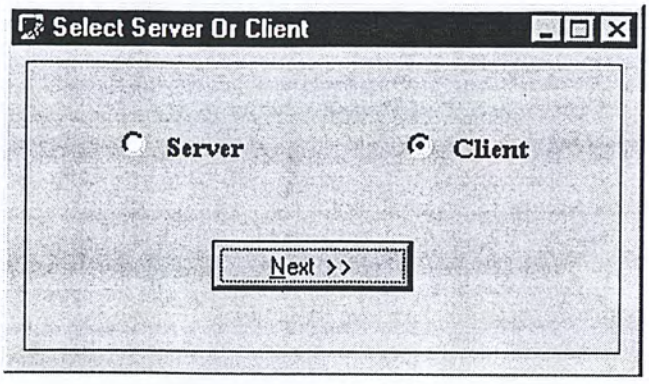
4.1 การใช้งานโปรแกรม

การทำงานโปรแกรมควบคุมแสดงผลหน้าจอของเครื่องลูกข่ายในระบบเครือข่ายท้องถิ่น เมื่อโปรแกรมเริ่มทำงานก็จะปรากฏฟอร์ม ดังรูปที่ 4.1



รูปที่ 4.1 แสดงฟอร์มเกี่ยวกับโปรแกรม

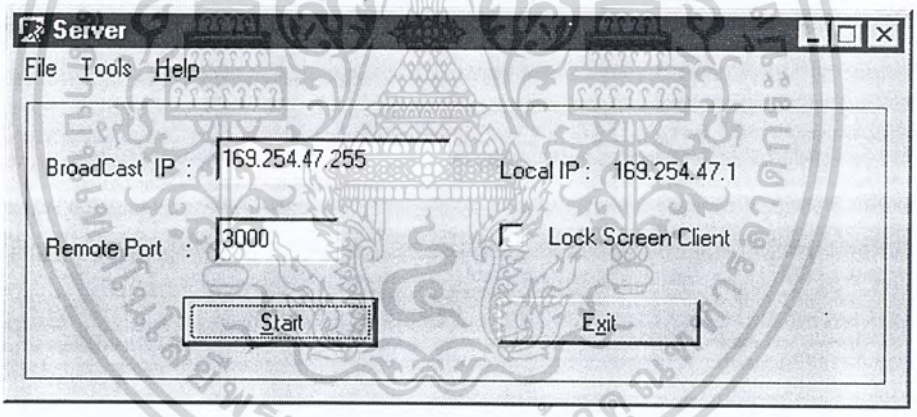
หลังจากนั้นสักพักก็จะปรากฏฟอร์มตามรูปที่ 4.2 เพื่อเลือกการทำงานของโปรแกรมให้เป็นเครื่องแม่ข่าย (Server) หรือเครื่องลูกข่าย (Client) ซึ่งในการทำงานของเครื่องแม่ข่ายและเครื่องลูกข่ายจะมีการทำงานไม่เหมือนกัน



รูปที่ 4.2 แสดงฟอร์มการเลือกให้เป็นเครื่องแม่ข่าย (Server) หรือเครื่องลูกข่าย (Client)

4.1.1 เครื่องแม่ข่าย (Server)

จากฟอร์มรูปที่ 4.2 เมื่อเลือกให้โปรแกรมทำงานเป็นเครื่องแม่ข่ายก็จะปรากฏฟอร์มหลักของเครื่องแม่ข่ายดังรูปที่ 4.3



รูปที่ 4.3 แสดงฟอร์มหลักของเครื่องแม่ข่าย

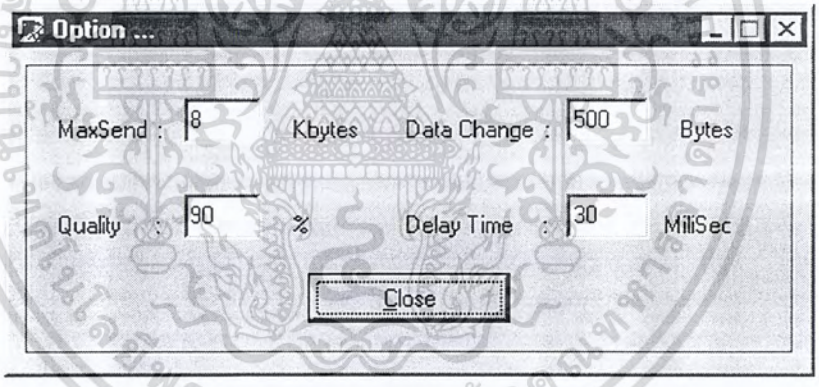
จากรูปที่ 4.3 อธิบายส่วนต่างของฟอร์มหลักของเครื่องแม่ข่ายได้ดังนี้

- BroadCast IP เป็นหมายเลขไอพี บรอดคาสต์ในเครือข่าย
- Remote Port เป็นหมายเลขพอร์ตของเครื่องลูกข่ายที่จะทำการติดต่อ ซึ่งหมายเลขพอร์ตนี้จะต้องตรงกับหมายเลข Receive Port ของเครื่องลูกข่าย (รูปที่ 4.8)
- Local IP เป็นหมายเลขไอพีของเครื่องแม่ข่ายที่โปรแกรมทำงานอยู่
- Lock Screen Client เป็นการกำหนดให้เครื่องลูกข่ายอยู่ในสภาวะล็อกหน้าจอ เมื่อเครื่องลูกข่ายมีการกดปุ่ม Full Screen แล้ว (รูปที่ 4.8)

เมื่อจะเริ่มต้นให้โปรแกรมทำการจับหน้าจอแล้วส่งไปยังเครื่องลูกข่ายก็ทำการกดปุ่ม Start โปรแกรมก็จะเริ่มทำการจับหน้าจอแล้วส่งไปให้เครื่องลูกข่ายเป็นระยะ ๆ และเมื่อจะยกเลิกการจับหน้าจอก็ทำได้โดยการกดปุ่ม Stop และสามารถออกโปรแกรมโดยการกดปุ่ม Exit

สำหรับเมนูฟอร์มหลักของเครื่องแม่ข่ายมีดังนี้

- File
  - --> Exit ออกจากโปรแกรม
- Tools
  - --> Option แสดงฟอร์มปรับแต่งโปรแกรมของเครื่องแม่ข่าย รูปที่ 4.4
  - --> Server Status แสดงฟอร์มการทำงานของโปรแกรมของเครื่องแม่ข่าย รูปที่ 4.5
  - --> Change Screen แสดงฟอร์มการเปลี่ยนโหมดการแสดงผลของจอภาพ รูปที่ 4.6
- Help
  - --> Help Topics แสดงฟอร์มอธิบายเกี่ยวกับการใช้งานโปรแกรม รูปที่ 4.7
  - --> About Program แสดงฟอร์มเกี่ยวกับโปรแกรม รูปที่ 4.1



รูปที่ 4.4 แสดงฟอร์มการปรับแต่งโปรแกรมเพิ่มเติม

จากรูปที่ 4.4 ค่าต่าง ๆ ที่เราสามารถปรับแต่งเพื่อให้การทำงานของโปรแกรมเร็วขึ้น มีดังนี้

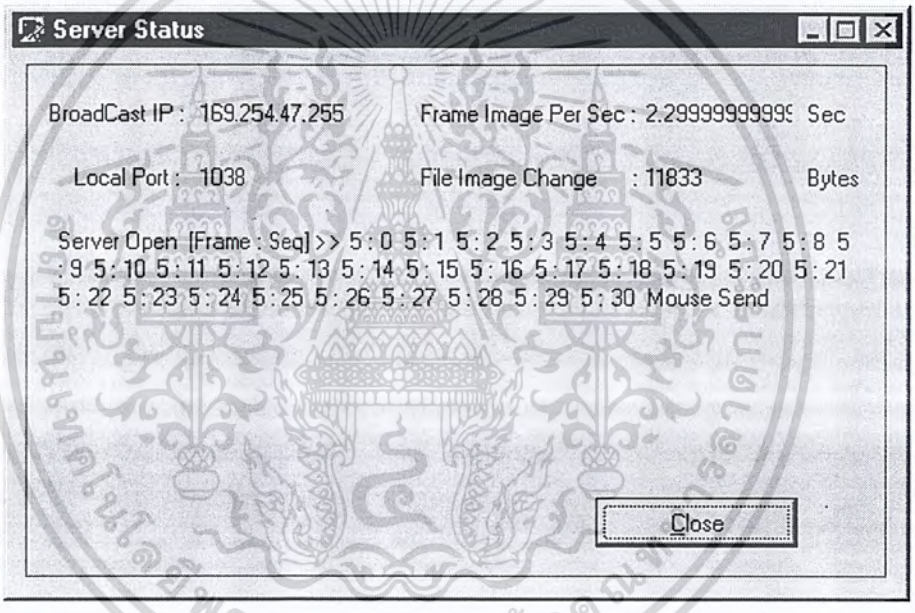
MaxSend เป็นขนาดของแพ็กเก็ตในการส่งข้อมูล ซึ่งแพ็กเก็ตของ Winsock มีขนาดสูงสุด 8 Kbytes การปรับค่านี้จะทำเมื่อการส่งข้อมูลแพ็กเก็ตขนาดใหญ่มีปัญหา เครื่องลูกข่ายไม่สามารถรับข้อมูลได้หรือไม่สามารถส่งข้อมูลไปยังเครื่องลูกข่ายได้

Data Change เป็นการตรวจสอบว่าภาพบนหน้าจอมีการเปลี่ยนมากน้อยเพียงใดโดยดูจากขนาดของไฟล์ภาพที่ทำการบีบอัดแล้ว ถ้าขนาดของไฟล์ภาพใหม่กับขนาดของไฟล์ภาพเดิมมีการเปลี่ยนแปลงน้อยกว่า Data Change ที่ตั้งไว้ การส่งข้อมูลก็จะส่งไปเฉพาะตำแหน่งเมาส์เท่านั้น ซึ่งจะทำให้

การแสดงภาพที่เครื่องลูกข่ายสามารถทำได้เร็วขึ้น การดูว่าขนาดไฟล์เปลี่ยนแปลงไปเท่าใดก็สามารถดูได้จาก Server Status ดังรูปที่ 4.5

Quality เป็นการกำหนดระดับคุณภาพของภาพที่จะทำการส่งไปยังเครื่องลูกข่าย โดยถ้ากำหนดระดับคุณภาพของภาพที่จะทำการส่งไปน้อยลงก็จะทำให้การทำงานของโปรแกรมเร็วขึ้นสามารถส่งได้เร็วขึ้น แต่ภาพก็จะมีความคมชัดน้อยลงซึ่งไม่ควรกำหนดค่าน้อยกว่า 60 %

Delay Time เป็นการกำหนดเวลาในการส่งแพ็กเก็ตแต่ละแพ็กเก็ตซึ่งขนาดแพ็กเก็ตจะมีขนาดเท่ากับ MaxSend ถ้ากำหนดเวลาน้อยลงก็จะทำให้สามารถส่งข้อมูลได้เร็วขึ้น แต่ถ้ากำหนดน้อยเกินไปก็จะทำให้การส่งข้อมูลเกิดความผิดพลาดหรือข้อมูลสูญหายเป็นต้น ในที่นี้กำหนดไว้ที่ 30 MiliSec สำหรับเครื่องคอมพิวเตอร์ความเร็วตั้งแต่ 333 MHz ขึ้นไป



รูปที่ 4.5 แสดงฟอร์มสภาวะการทำงานของเครื่องแม่ข่าย

จากรูปที่ 4.5 เป็นการแสดงสภาวะการทำงานของเครื่องแม่ข่าย ซึ่งแสดงอธิบายได้ดังนี้

BroadCast IP เป็นหมายเลขไอพีบรอดคาสต์ที่ทำการส่ง

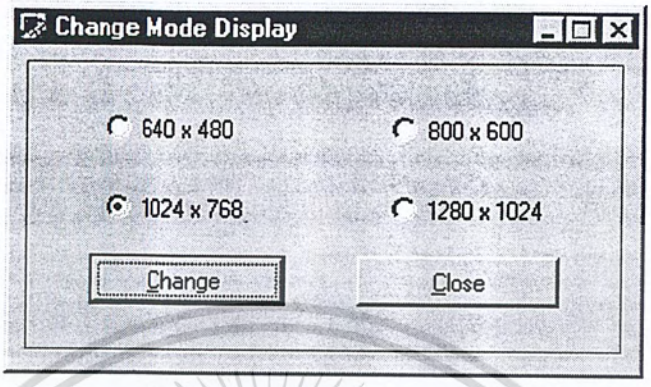
Local Port เป็นหมายเลยพอร์ตของเครื่องแม่ข่ายที่ใช้งานอยู่

Frame Image Per Sec เป็นเวลาที่ใช้ในการส่งข้อมูลแต่ละภาพ

File Image Change เป็นขนาดไฟล์ภาพที่เปลี่ยนแปลงจากไฟล์ภาพเดิม

ส่วนด้านล่างเป็นการบอกสถานะของ Winsock ว่าพร้อมที่จะส่งข้อมูลไหมถ้าแสดงเป็น Server Open แสดงว่าสามารถทำการส่งข้อมูลได้ และการส่งข้อมูลของเครื่องแม่ข่ายซึ่งจะเป็น

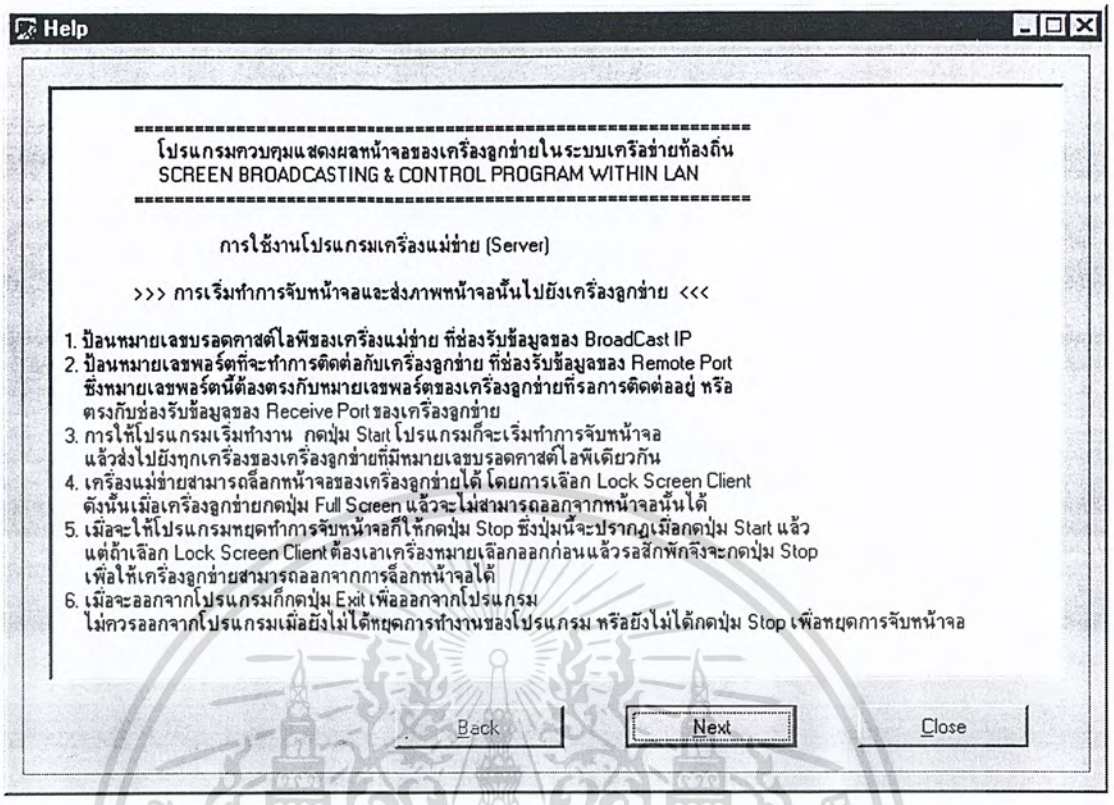
ลักษณะของ Frame : Seq ซึ่งจะบอกถึงลำดับเฟรมและลำดับแพ็กเก็ตที่ทำการส่งอยู่ ส่วน Mouse Send เป็นการแสดงการส่งเมาส์



รูปที่ 4.6 แสดงฟอร์มการเปลี่ยนโหมดการแสดงผลของหน้าจอ

จากรูปที่ 4.6 เป็นฟอร์มการเปลี่ยนโหมดการแสดงผลของหน้าจอ ซึ่งในที่นี้ได้กำหนดไว้ 4 โหมดด้วยกัน สำหรับโหมดการแสดงผลทั้งเครื่องแม่ข่ายและเครื่องลูกข่ายต้องมีโหมดการแสดงผลเหมือนกัน มิฉะนั้นจะทำให้การแสดงภาพที่เครื่องลูกข่ายคลาดเคลื่อนได้ และการเลือกโหมดการแสดงผลขนาดใหญ่จะทำให้โปรแกรมทำงานได้ช้าลงตามขนาดของภาพบนหน้าจอ

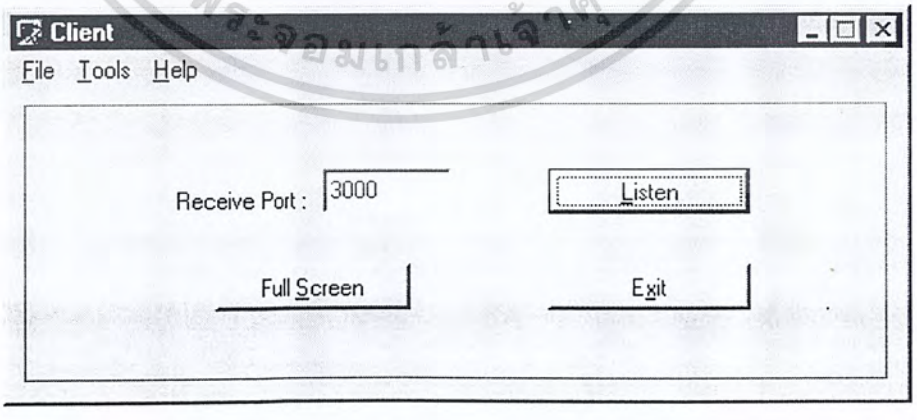
สำหรับฟอร์มการช่วยเหลือนั้นจะเป็นการอธิบายการใช้งานโปรแกรมของเครื่องแม่ข่าย ซึ่งแสดงดังรูปที่ 4.7



รูปที่ 4.7 แสดงฟอร์มช่วยเหลือการใช้โปรแกรมของเครื่องแม่ข่าย

4.1.2 เครื่องลูกข่าย (Client)

จากฟอร์มรูปที่ 4.2 เมื่อเลือกให้โปรแกรมทำงานเป็นเครื่องลูกข่ายก็จะปรากฏฟอร์มหลักของเครื่องลูกข่ายดังรูปที่ 4.8

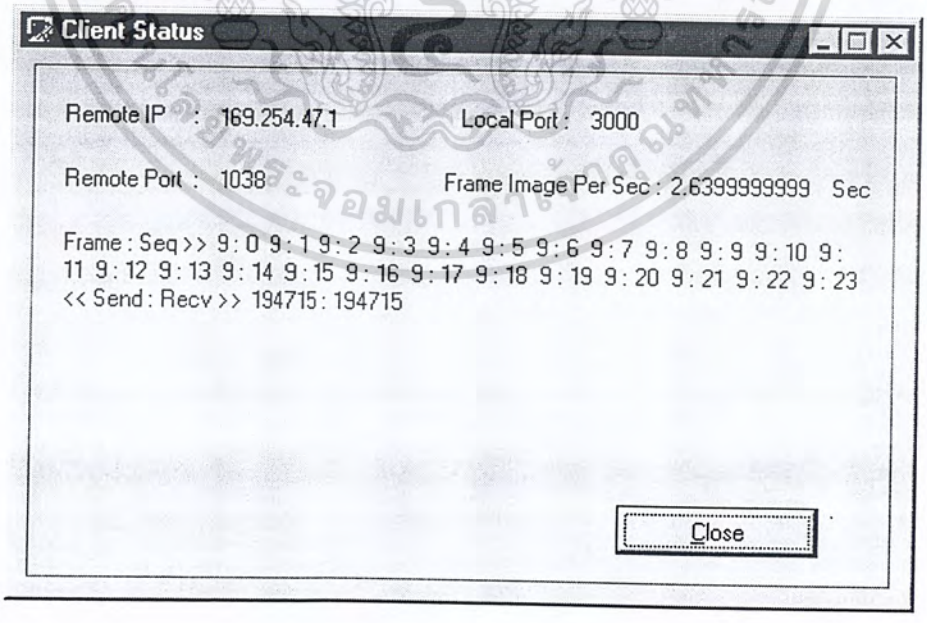


รูปที่ 4.8 แสดงฟอร์มหลักของเครื่องลูกข่าย

จากรูปที่ 4.8 อธิบายส่วนต่างของฟอร์มหลักของเครื่องลูกข่ายได้ดังนี้ Receive Port เป็นหมายเลยพอร์ตที่รอการติดต่อของเครื่องแม่ข่าย ซึ่งหมายเลขพอร์ตนี้จะต้องตรงกับหมายเลขพอร์ตที่เครื่องแม่ข่ายติดต่อมา เมื่อจะให้เริ่มโปรแกรมทำงานก็โดยการกดปุ่ม Listen แล้วกดปุ่ม Full Screen เพื่อแสดงรูปที่เครื่องแม่ข่ายส่งมา เมื่อกดปุ่ม Full Screen แล้วสามารถออกได้โดยการกดคีย์บอร์ด "ESC" หรือคลิกเมาท์ที่ปุ่มขวาก็ได้โดยจะมีเมนูขึ้นมาให้เลือกว่าจะพักหน้าจอหรือกลับไปยังหน้าจอหลักของเครื่องลูกข่าย แต่ถ้าเครื่องแม่ข่ายทำการล็อกหน้าจอก็จะไม่สามารถออกจากหน้าจอนี้ได้ และสามารถทำการยกเลิกการติดต่อกับเครื่องแม่ข่ายโดยกดปุ่ม Close Listen และออกจากโปรแกรมโดยกดปุ่ม Exit

สำหรับเมนูฟอร์มหลักของเครื่องลูกข่ายมีดังนี้

- File
  - --> Exit ออกจากโปรแกรม
- Tools
  - --> Client Status แสดงฟอร์มการทำงานของโปรแกรมของเครื่องลูกข่าย รูปที่ 4.9
  - --> Change Screen แสดงฟอร์มการเปลี่ยนโหมดการแสดงผลของจอภาพ รูปที่ 4.6
- Help
  - --> Help Topics แสดงฟอร์มอธิบายเกี่ยวกับการใช้งานโปรแกรม รูปที่ 4.10
  - --> About Program แสดงฟอร์มเกี่ยวกับโปรแกรม รูปที่ 4.1



รูปที่ 4.9 แสดงสภาวะการทำงานของเครื่องลูกข่าย

จากรูปที่ 4.9 เป็นการแสดงสภาวะการทำงานของเครื่องลูกข่าย ซึ่งแสดงอธิบายได้ดังนี้

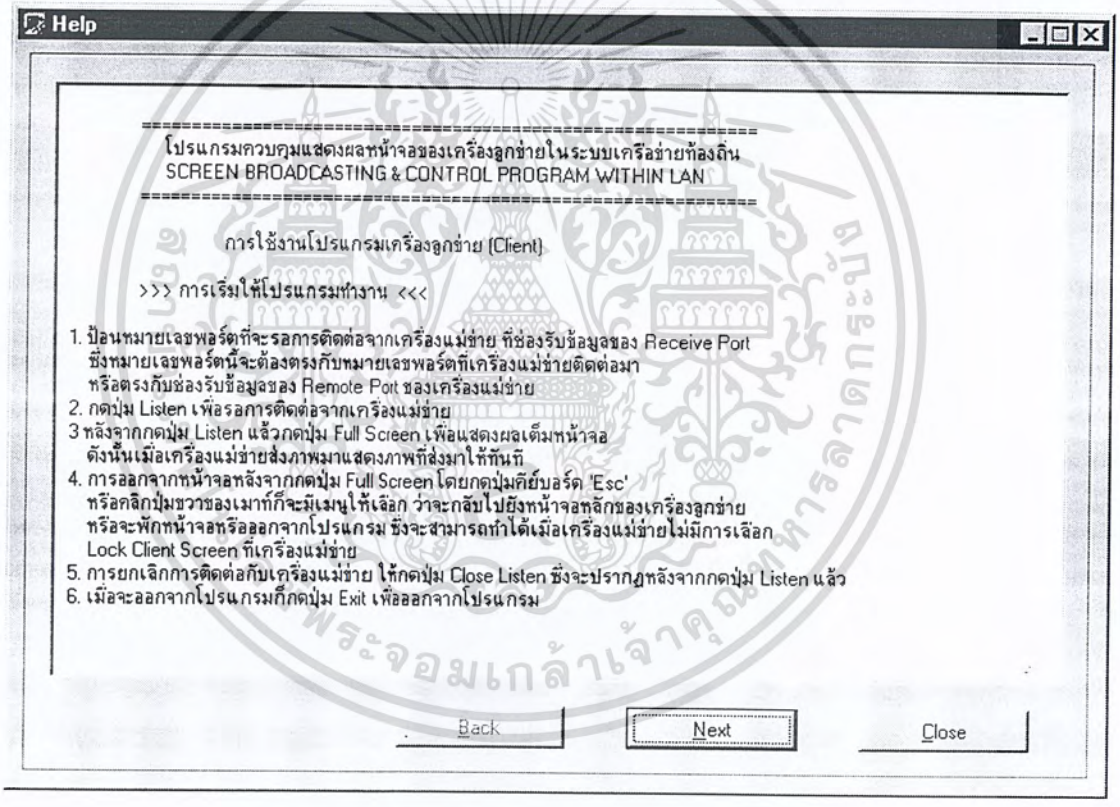
Remote IP เป็นหมายเลขไอพีแอดเดรสของเครื่องแม่ข่ายที่ทำการติดต่อ

Remote Port เป็นหมายเลขพอร์ตของเครื่องแม่ข่ายที่ทำการติดต่อ

Local Port เป็นหมายเลยพอร์ตที่ใช้งานของเครื่องลูกข่าย

Frame Image Per Sec เป็นเวลาที่ใช้ในการแสดงภาพแต่ละครั้งต่อวินาที

ส่วนด้านล่างเป็นบอกว่าลำดับการรับข้อมูลเฟรมและแพ็กเก็ตที่เครื่องแม่ข่ายส่งมา และบอกขนาดข้อมูลที่เครื่องแม่ข่ายส่งมากับขนาดที่เครื่องลูกข่ายรับได้ สำหรับฟอร์มการช่วยเหลือนั้นจะเป็นการอธิบายการใช้งานโปรแกรมของเครื่องลูกข่าย ซึ่งแสดงดังรูปที่ 4.10



รูปที่ 4.10 แสดงฟอร์มช่วยเหลือการใช้โปรแกรมของเครื่องลูกข่าย

4.2 ผลการทดลอง

วันที่ทำการทดลอง 02 เม.ย. 2545

เครื่อง A --> AMD 333 MHz , RAM 256 MB , VGA 8 MB , DISPLAY 1024*768*24Bit

เครื่อง B --> Pentium-MMX 200 MHz, RAM 32 MB , VGA 4 MB , DISPLAY 1024*768*24Bit

เครื่อง C --> Pentium lll 733 MHz , RAM 256 MB, VGA 32 MB , DISPLAY 1024*768*16 Bit

เครื่อง D --> Athlon 700 MHz , RAM 128 MB , VGA 32 MB , DISPLAY 1024*768*16Bit

กำหนดค่า Option ของเครื่องแม่ข่าย ดังนี้

MaxSend = 8 KBytes Data Change = 0 Bytes

Quality = 90 Delay Time = 30 miliSec

จำนวนหน้าจอของเครื่องแม่ข่ายที่ส่งไปยังเครื่องลูกข่ายแต่ละครั้ง 100 หน้าจอ

1. ส่งหน้าจอของเครื่องแม่ข่ายขณะยังไม่มีการทำงานของโปรแกรมอื่น

เครื่องแม่ข่าย	เครื่องลูกข่าย	จำนวน		เวลาที่ใช้ (นาที)	ขนาดไฟล์ (Kbytes)
		รับได้	รับได้แต่แสดงภาพเพี้ยน,เสีย		
A	B	50	0	3.48	230 - 280
		51	0	3.51	230 - 280
B	A	97	0	4.58	230 - 280
		99	0	5.09	230 - 280
A	B	51	2	2.46	135 - 155
		52	0	2.45	135 - 155
B	A	98	0	3.08	135 - 155
		99	0	2.53	135 - 155
C	A	99	1	2.51	150 -180
		98	0	2.49	150 -180
A	C	100	0	2.48	150 -180
		100	0	2.51	150 -180

ตารางที่ 4.1 ผลการทดลองส่งหน้าจอเครื่องแม่ข่ายขณะยังไม่มีการทำงานของโปรแกรมอื่น

เครื่องแม่ข่าย	เครื่องลูกข่าย	จำนวน		เวลาที่ใช้ (นาที)	ขนาดไฟล์ (Kbytes)
		รับได้	รับได้แต่แสดงภาพเพี้ยน,เสีย		
D	C	100	0	2.34	180 - 210
		100	0	2.35	180 - 210
C	D	100	0	2.45	180 - 210
		100	0	2.44	180 - 210

ตารางที่ 4.1 (ต่อ) ผลการทดลองส่งหน้าจอเครื่องแม่ข่ายขณะยังไม่มีการทำงานของโปรแกรมอื่น

2 . ส่งหน้าจอของเครื่องแม่ข่ายขณะใช้งานโปรแกรม Microsoft Word

เครื่องแม่ข่าย	เครื่องลูกข่าย	จำนวน		เวลาที่ใช้ (นาที)	ขนาดไฟล์ (Kbytes)
		รับได้	รับได้แต่แสดงภาพเพี้ยน,เสีย		
A	B	64	0	4.16	130 -170
		57	0	4.20	130 -170
B	A	96	0	5.44	130 -170
		98	0	5.37	130 -170
D	C	100	0	3.11	180 - 210
		100	1	3.05	180 - 210
C	D	100	0	3.23	180 - 210
		99	0	3.09	180 - 210

ตารางที่ 4.2 ผลการทดลองส่งหน้าจอเครื่องแม่ข่ายขณะใช้งานโปรแกรม Microsoft Word

3 . ส่งหน้าจอของเครื่องแม่ข่ายขณะใช้งานโปรแกรม ACDSee โดยกำหนดให้มีการโหลดรูปทุก 2 วินาที

เครื่องแม่ข่าย	เครื่องลูกข่าย	จำนวน		เวลาที่ใช้ (นาที)	ขนาดไฟล์ (Kbytes)
		รับได้	รับได้แต่แสดงภาพเพี้ยน,เสีย		
A	B	58	1	3.56	200 – 350
		53	0	3.29	200 – 350
B	A	97	0	5.33	200 – 350
		97	0	5.37	200 – 350
D	C	100	0	2.39	180 - 250
		100	1	2.35	180 - 250
C	D	100	0	2.50	180 - 250
		100	0	2.48	180 - 250

ตารางที่ 4.3 ผลการทดลองส่งหน้าจอเครื่องแม่ข่ายขณะใช้งานโปรแกรม ACDSee โดยกำหนดให้ มีการโหลดรูปทุก 2 วินาที

จากการทดลองเมื่อให้โปรแกรมทำการจับหน้าจอของเครื่องแม่ข่ายแล้วส่งไปยังเครื่องลูกข่ายทุก ๆ ครั้งที่จับได้จะใช้เวลาประมาณ 3-4 วินาทีต่อหนึ่งหน้าจอ และในบางครั้งหน้าจอของเครื่องลูกข่ายอาจจะแสดงภาพที่เพี้ยนหรือเสียใช้งานไม่ได้ ซึ่งความผิดพลาดนี้เกิดขึ้นขณะทำการบีบอัดภาพที่จับบนหน้าจอเครื่องแม่ข่ายเป็นไฟล์ JPEG การเกิดความผิดพลาดนี้จะเกิดขึ้นน้อยมาก และจะเป็นเฉพาะบางภาพเท่านั้น ซึ่งสามารถแก้ไขได้โดยการปรับเปลี่ยนค่า Quality เมื่อทำการเปลี่ยนแล้วภาพที่เพี้ยนหรือเสียใช้งานไม่ได้ก็จะเป็นปกติ

บทที่ 5 สรุปและวิจารณ์ผลการทดลอง

5.1 บทสรุป

โครงงานที่จัดทำขึ้นมานี้จะทำการจับหน้าจอการแสดงผลของเครื่องแม่ข่ายเป็นระยะๆ แล้วส่งไปให้เครื่องลูกข่ายเพื่อให้เครื่องลูกข่ายแสดงผลหน้าจอเหมือนกับเครื่องแม่ข่าย ซึ่งสามารถที่จะแสดงผลบนหน้าจอเครื่องลูกข่ายได้ประมาณ 3-4 วินาทีต่อภาพ (โหมด 1024*768*24 Bit) ในกรณีที่ภาพบนเครื่องแม่ข่ายมีการเปลี่ยนแปลงตลอดและจะเร็วขึ้นเมื่อภาพที่จับนั้นไม่แตกต่างกันมากนัก เช่น มีสีซ้ำกันมากๆ ซึ่งผลการทำงานของโปรแกรมก็อยู่ในเกณฑ์ที่ใช้ได้ แต่บางครั้งในการแสดงผลบนหน้าจอของเครื่องลูกข่ายอาจจะมีความผิดพลาดอยู่บ้าง ในการจับหน้าจอภาพที่มีการเปลี่ยนแปลงตลอด แต่ก็ไม่มีผลอะไรมากนักเพราะเมื่อส่งภาพใหม่ไปก็จะแสดงได้ตามปกติและโอกาสที่จะเกิดขึ้นเช่นนี้ก็น้อยมาก

5.2 ปัญหาและอุปสรรค

สำหรับในการทำโครงงานนี้เนื่องจากผู้เขียนยังไม่ชำนาญในการเขียนโปรแกรมมากนัก ดังนั้นในการทำโครงงานนี้ส่วนใหญ่ผู้เขียนจึงไม่ได้ศึกษาถึงรายละเอียดมากนัก ซึ่งจะศึกษาในเฉพาะส่วนที่ต้องการนำมาใช้งานเท่านั้น ดังนั้นโปรแกรมบางส่วนของโครงงานนี้ผู้เขียนก็ใช้การประยุกต์จากโปรแกรมอื่นเช่น การบีบอัดภาพเป็นไฟล์ JPEG ซึ่งได้จากการค้นคว้ามาผสมผสานกันและพัฒนาเพิ่มขึ้นเรื่อย ๆ จึงอาจจะทำให้โปรแกรมนี้ยังไม่สมบูรณ์เท่าที่ควร แต่ก็ใช้ได้อยู่ในเกณฑ์ที่น่าพอใจ

5.3 แนวทางการพัฒนาต่อ

- เนื่องจากข้อจำกัดของ Winsock ซึ่งสามารถส่งข้อมูลได้สูงสุด 8 KBytes และใช้โปรโตคอล UDP ในการติดต่อระบบเครือข่าย ซึ่งสามารถพัฒนาโดยการใช้โปรโตคอลอื่น ที่สามารถทำงานได้ดีกว่านี้ได้ เพื่อให้โปรแกรมสามารถทำงานได้เร็วขึ้น

- ในส่วนของโปรแกรมการส่ง-รับไฟล์ภาพ โดยใช้ Winsock UDP ผู้เขียนไม่ได้การทำตรวจสอบแต่ละแพ็กเก็ตว่าถ้าเครื่องลูกข่ายไม่สามารถที่จะรับแพ็กเก็ตนั้นที่เครื่องแม่ข่ายส่งมาควรจะให้เครื่องแม่ข่ายทำการส่งแพ็กเก็ตเดิมนั้นมาให้อีกครั้งหรือไม่ (Re-Tranmition) หรือถ้ารับแพ็กเก็ตที่เครื่องแม่ข่ายส่งมาไม่ครบควรจะทำการแก้ไขอย่างไร ซึ่งในส่วนนี้สามารถที่จะนำไปศึกษาและพัฒนาต่อไปได้

เอกสารอ้างอิง

[1] สถาปัตยกรรมและโปรโตรคอลทีซีพี/ไอพี ; สุรศักดิ์ สงวนพงษ์,โครงการตำราวิชาการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ พิมพ์ครั้งที่ 1 พ.ศ.2543

[2] เปิดโลกของ TCP/IP และโปรโตคอลของอินเตอร์เน็ต ; สุวัฒน์ ปุณณชัยยะ,ต้น ตัณฑ์สุทธิวงศ์ และ สุพจน์ ปุณณชัยยะ พิมพ์ครั้งที่ 1 พ.ศ.2543

[3] Visual Basic 6 ฉบับโปรแกรมเมอร์ ; กิตติ ภักดีวัฒนะกุล และ จำลอง ครูอุตสาหะ พิมพ์ครั้งที่ 9 พ.ศ.2543

[4] MSDN – Microsoft Developer Network ; Visual Studio 6.0 Library,Microsoft

[5] การเขียนโปรแกรมด้วย Visual Basic 6.0 ระดับสูง ; สุทธิศักดิ์ พงศ์ธนาพานิช , โรงพิมพ์จุฬาลงกรณ์มหาวิทยาลัย พ.ศ. 2542

http://www.vbaccelerator.com

http://www.thaiio.com

http://www.planet-source.com