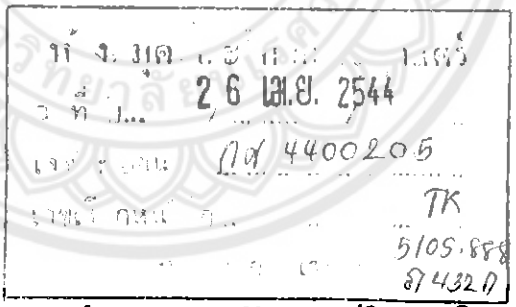
การพัฒนาโปรแกรมสำหรับเครื่องแม่ข่ายเพื่อการถ่ายโอนข้อมูลแบบ FTP

The development of the server programming for FTP data transferring

นายภาณุพงศ์ สอนคม รหัส 40360430

นายโชติ ขำเพชร รหัส 40360620

นางสาวชุลีชา คะชา รหัส 40360687



โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2543

ใบรับรองโครงงานวิจัย

หัวข้อโครงงาน : การพัฒนาโปรแกรมสำหรับเครื่องแม่ข่าย
เพื่อการถ่ายโอนข้อมูลแบบ FTP

Performance : The Development of the Server Programming for FTP Data Transferring

ผู้ดำเนินโครงงาน : นายภาณุพงศ์ สอนคม รหัส 40360430
นายโชติ ขำเพชร รหัส 40360620
นางสาวชุลีชา คะชา รหัส 40360687

อาจารย์ที่ปรึกษา : อาจารย์อธิศ ปทุมวรรณ

อาจารย์ที่ปรึกษาร่วม : อาจารย์แคทรียา อัคสูงเนิน

สาขา : วิศวกรรมคอมพิวเตอร์

ภาควิชา : วิศวกรรมไฟฟ้าและคอมพิวเตอร์

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิจัย

.....ประธานกรรมการ
(อาจารย์อธิศ ปทุมวรรณ)

.....กรรมการ
(อาจารย์วัชร์วีร์ พืชพันธ์)

.....กรรมการ
(อาจารย์ประทีป ตรีรณโอภาส)

หัวข้อโครงงาน : การพัฒนาโปรแกรมสำหรับเครื่องแม่ข่ายเพื่อการถ่ายโอนข้อมูลแบบ FTP
ผู้ดำเนินโครงงาน : นายภาณุพงศ์ สอนคม รหัส 40360430
นายโชติ ขำเพชร รหัส 40360620
นางสาวชุลีชา คะชา รหัส 40360687
อาจารย์ที่ปรึกษา : อาจารย์ อธิศ ปทุมวรรณ
อาจารย์ที่ปรึกษาร่วม : อาจารย์ แคทรียา อัดสูงเนิน
สาขา : วิศวกรรมคอมพิวเตอร์
ภาควิชา : วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา : 2543

.....

บทคัดย่อ

โครงงานการพัฒนาโปรแกรมสำหรับเครื่องแม่ข่ายสำหรับการถ่ายโอนข้อมูลแบบ FTP นี้ได้พัฒนาโปรแกรมตามมาตรฐาน RFC 959 เพื่อให้ได้โปรแกรมตัวอย่างซึ่งใช้งานในระบบปฏิบัติการไมโครซอฟท์วินโดว์ โดยใช้ความรู้เรื่อง TCP/IP , การเขียนโปรแกรมในระบบเน็ตเวิร์ค และการเขียนโปรแกรมบนวินซ็อก โดยใช้โปรแกรม lccwin32 และ visual C++ ในการสร้างและพัฒนาโปรแกรม

ซึ่งผลที่ได้คือโปรแกรม FTP ซึ่งใช้เป็นกรณีศึกษา และเป็นโปรแกรมต้นแบบสำหรับผู้ที่มีความสนใจในด้านนี้ต่อไป

Project Title : The development of the server programming for FTP data transferring

Name : Mr. Panuphong Sornkhom ID. 40360430

Mr. Shoat Kumpet ID. 40360620

MissChuleecha Cachar ID. 40360687

Project Advisor : Mr. Artit Pathumwan

Co-Project Advisor : Miss Cattareeya Adsoongnoen

Field of study : Computer Engineering

Department : Electrical and Computer Engineering

Academic Year : 2000

.....

Abstract

This development of the server programming for FTP data transferring Project is developed the software followed the RFC 959 standard. For the example program which work in Microsoft Windows operating system. By using lccwin32 and visual C++ program with the knowledge of TCP/IP , network programming and Winsock programming for building and development.

Which the result is FTP program which use as case study and example programming for anybody who interested in this area.

กิตติกรรมประกาศ

โครงงานชิ้นนี้ต้องอาศัยความรู้เพิ่มเติมมากมาย และต้องการการชี้แนะอย่างสูง ทางคณะผู้จัดทำจึงขอขอบพระคุณ อาจารย์อธิศ ปทุมวรรณ และ อาจารย์แคทรียา อัดสูงเนิน ที่ให้คำปรึกษาและชี้แนะสิ่งต่างๆมากมาย และขอบคุณผู้ที่คอยให้การสนับสนุนและให้กำลังใจด้วยดีตลอดมา

นายภาณุพงศ์ สอนคม

นายโชติ ขำเพชร

นางสาวชุลีชา คะชา

สารบัญ

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

บทที่ 1
บทนำ

ในบทนี้เป็นการแนะนำส่วนประกอบการทำงานอย่างคร่าวๆของโครงงานนี้ซึ่งประกอบไปด้วยที่มาและความสำคัญของโครงงาน วัตถุประสงค์ในการทำโครงงานนี้ ขอบเขตของโครงงานที่สามารถทำได้ในส่วนนี้ ระยะเวลาการดำเนินงานของแต่ละขั้นตอน ผลที่คาดว่าจะได้รับจากโครงงาน และการใช้งบประมาณในการดำเนินงาน บทนี้มีเพื่อบอกความจำเป็นในการทำโครงงาน ดังที่จะได้กล่าวดังต่อไปนี้

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบันเทคโนโลยีอินเตอร์เน็ตได้เข้ามามีบทบาทในชีวิตประจำวันของเรามากขึ้น การถ่ายโอนแฟ้มข้อมูล (File Transfer) ในระบบเครือข่ายคอมพิวเตอร์ เป็นอีกเทคโนโลยีหนึ่งที่ได้ถูกนำมาใช้กันอย่างแพร่หลายและเป็นที่นิยมอย่างมากในปัจุบัน ซึ่งการถ่ายโอนแฟ้มข้อมูล (File Transfer) ในระบบเครือข่ายคอมพิวเตอร์ TCP/IP โดยใช้การบริการแบบ File Transfer Protocol (FTP) จากเครื่องคอมพิวเตอร์แม่ข่ายเป็นเครื่องมือสำคัญในการโอนถ่ายแฟ้มข้อมูลระหว่างเครื่องคอมพิวเตอร์แม่ข่าย (Server) กับเครื่องคอมพิวเตอร์ลูกข่าย (Client)

ในปัจจุบันโปรแกรม FTP สำหรับเครื่องคอมพิวเตอร์แม่ข่ายยังมิได้มีการพัฒนาอย่างกว้างขวาง ประกอบกับเอกสารและข้อมูลในด้านนี้ ยังมีอยู่น้อย ส่งผลให้มีผู้ที่รู้และเข้าใจในกระบวนการถ่ายโอนแฟ้มข้อมูลในระบบเครือข่ายคอมพิวเตอร์แบบ FTP สำหรับเครื่องคอมพิวเตอร์แม่ข่ายมีเป็นจำนวนน้อย ซึ่งส่งผลกระทบต่อการพัฒนาศักยภาพทางเทคโนโลยีอินเตอร์เน็ต

ทางคณะผู้เสนอโครงงาน จึงมีความประสงค์ที่จะพัฒนาโปรแกรม FTP สำหรับเครื่องคอมพิวเตอร์แม่ข่ายในระบบปฏิบัติการ Windows โดยอ้างอิงตามมาตรฐาน RFC 959 เพื่อใช้เป็นกรณีศึกษา และ เป็นต้นแบบเพื่อการพัฒนาโปรแกรม FTP สำหรับผู้ที่มีความสนใจในด้านนี้ ต่อไป

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อให้สามารถสร้างโปรแกรม FTP Server ได้
2. เพื่อความรู้ความเข้าใจเกี่ยวกับ File Transfer Protocol , Socket และ TCP/IP
3. เพื่อเป็นแนวทางที่จะให้ผู้ที่มีความสนใจเกี่ยวกับระบบการส่งผ่านข้อมูลภายในเครือข่ายแบบ TCP/ IP ได้ศึกษากันต่อไป

1.3 ขอบข่ายของโครงงาน

1. ศึกษาทฤษฎีที่เกี่ยวข้องกับสิ่งต่าง ๆ เหล่านี้ คือ
   - 1.1 File Transfer Protocol อ้างอิงตาม RFC 959
   - 1.2 Socket
   - 1.3 TCP/IP
2. ออกแบบโปรแกรม FTP Server
3. พัฒนาโปรแกรม FTP Server
4. ทำการทดสอบการใช้งาน
5. วิเคราะห์การทดลองและสรุปผลการทดลอง

1.4 ขั้นตอนการดำเนินงาน

กิจกรรม	เดือน – ปี						
	มี.ค.43	เม.ย.43	พ.ค.43	มิ.ย.43	ก.ค.43	ส.ค.43	ก.ย.43
1.ศึกษา	←----------→						
2.ออกแบบ		←----------→					
3.พัฒนา			←----------→				
4.ทดสอบ						←----------→	
5.สรุปผล							←-→

1.5 ผลที่คาดว่าจะได้รับ

1. โปรแกรม FTP Server
2. ความรู้ความเข้าใจเกี่ยวกับ FTP Server , Socket , TCP/IP
3. แนวทางในการพัฒนาโปรแกรมที่เกี่ยวข้องกับระบบเครือข่าย TCP/IP

1.6 <u>งบประมาณที่ใช้</u> นิสิต : คน : 1000 บาท

1. ค่าหนังสือสำหรับการค้นคว้า		2,000	บาท
2. ค่าถ่ายเอกสารหนังสือสำหรับการค้นคว้า		500	บาท
3. ค่ากระดาษ A4 3 รีม		350	บาท
4. ค่าอุปกรณ์เครื่องเขียน		150	บาท
	รวม	3,000	บาท

บทที่ 2
หลักการและทฤษฎี

ในบทที่แล้วได้กล่าวถึงความสำคัญของการทำโครงงานและรูปแบบขั้นตอนการดำเนินงาน แต่ในการที่จะดำเนินโครงการนี้ต่อไปได้จำเป็นที่จะต้องอาศัยความรู้ความเข้าใจพื้นฐานเกี่ยวกับ มาตรฐานสากลในแบบต่างๆ โปรโตคอล TCP/IP และ WINDOWS SOCKET เพื่อการพัฒนาโปรแกรม FTP ดังที่จะได้กล่าวในต่อไปนี้

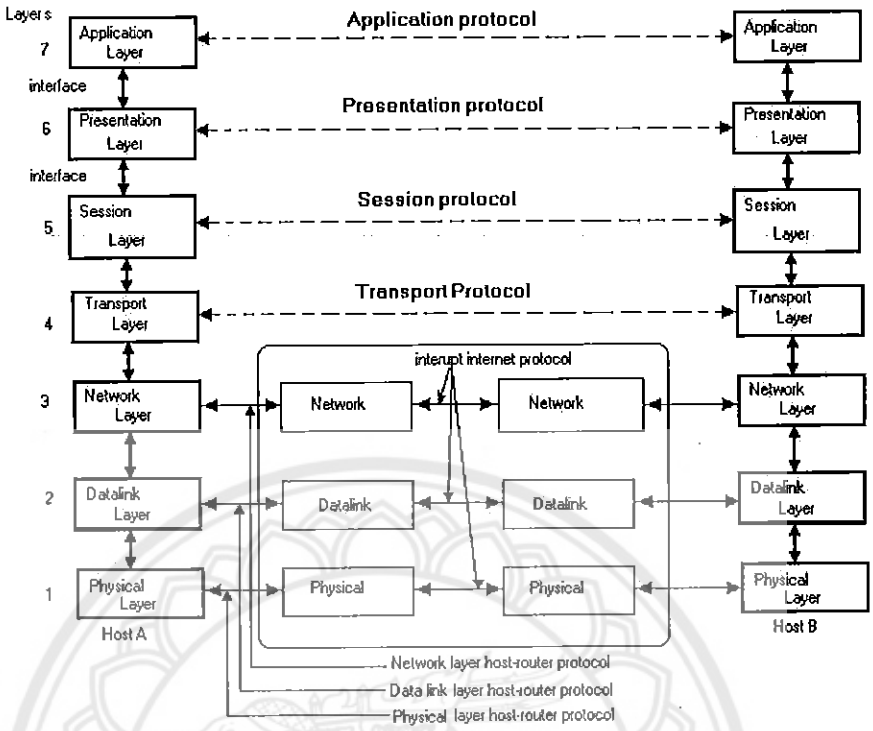
2.1 รูปแบบเครือข่ายระบบมาตรฐานสากล

องค์การ International Standards Organization (ISO) ได้กำหนดรูปแบบโครงสร้างมาตรฐานสากลสำหรับการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ไว้เรียกว่า Open Systems Interconnection (OSI) ซึ่งมีอยู่ทั้งหมด 7 ชั้นสื่อสารดังแสดงในรูป 2.1 ตัวโครงสร้างเองได้เน้นความสำคัญของรูปแบบการติดต่อสื่อสาร ระหว่างระบบเปิด (open systems) กับระบบเปิดจึงสามารถนำไปใช้อ้างอิงได้ในระดับสากลอย่างแท้จริง

แนวความคิดของการกำหนดมาตรฐานเป็นแบบชั้นสื่อสาร (layers) คือ

1. ชั้นสื่อสารแต่ละชั้นถูกกำหนดขึ้นมาตามบทบาทที่แตกต่างกัน
2. แต่ละชั้นสื่อสารจะต้องทำหน้าที่ตามที่ได้รับมอบหมายอย่างดียิ่ง
3. แต่ละฟังก์ชันในชั้นสื่อสารใดๆ จะต้องกำหนดขึ้นมาโดยใช้แนวความคิดในระดับสากลเป็นวัตถุประสงค์หลัก
4. ขอบเขตความรับผิดชอบของแต่ละชั้นสื่อสาร จะต้องกำหนดขึ้นมาเพื่อจำกัดปริมาณการแลกเปลี่ยนข้อมูลและผลกระทบข้างเคียงระหว่างการติดต่อให้มีน้อยที่สุด
5. จำนวนของชั้นสื่อสารจะต้องมีมากพอที่จะแยกฟังก์ชันการทำงานที่แตกต่างกันให้อยู่คนละชั้น แต่จะต้องไม่มีมากเกินความจำเป็น

จากนี้ไปจะเป็นการอธิบายการทำงานของแต่ละชั้นสื่อสารโดยเริ่มจากชั้น physical layer ซึ่งเป็นชั้นล่างสุด เรื่อยไปจนถึงชั้น Application layer ซึ่งเป็นชั้นบนสุด สถาบัน ISO ได้จัดทำรายละเอียดต่างๆของแต่ละชั้นสื่อสารไว้ดังนี้



รูปที่2.1 รูปแบบโครงสร้างโอเอสไอ

(ที่มา:เครือข่ายคอมพิวเตอร์ สัลลยุทธ์ สว่างสวรรณ)

2.1.1 ชั้นสื่อสารกายภาพ (The Physical Layer)

ชั้นกายภาพเป็นระดับชั้นล่างสุดที่เกี่ยวข้องโดยตรงกับอุปกรณ์สื่อสารต่างๆ ทำหน้าที่ในการกำหนดวิธีควบคุมการรับและส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ในระดับบิต ได้แก่ การส่งบิต 0 จะแทนด้วยกระแสไฟฟ้ากี่โวลต์ และบิต 1 จะต้องใช้กี่โวลต์ แต่ละบิตจะใช้ระยะเวลาในการส่งนานเท่าไหร่ การส่งเป็นแบบทางเดียวหรือสองทาง จะเริ่มติดต่ออย่างไร การติดต่อจะสิ้นสุดอย่างไร และสายเคเบิ้ลมีกี่เส้นแต่ละเส้นใช้เพื่ออะไร เป็นต้น จะเห็นได้ว่ากฎระเบียบสำหรับชั้นนี้จะเกี่ยวพันโดยตรงกับการทำงานของอุปกรณ์สัญญาณไฟฟ้า (หรือสัญญาณใดๆ) ขั้นตอนในการใช้อุปกรณ์เหล่านั้น และความสัมพันธ์กับสื่อที่ใช้รับ-ส่งสัญญาณ

2.1.2 ชั้นสื่อสารเชื่อมต่อข้อมูล (Data Link Layer)

หน้าที่หลักของชั้นเชื่อมต่อข้อมูลคือ ทำการรวบรวมข้อมูลจากชั้นกายภาพ ตรวจสอบความถูกต้องของข้อมูล แล้วส่งข้อมูลที่ปราศจากข้อผิดพลาดนี้ให้กับชั้นควบคุมเครือข่ายต่อไป โดยปกติผู้ส่งข้อมูลจะแบ่งข้อมูลที่มีความยาวมากออกเป็นกลุ่มข้อมูลย่อยๆ แต่ละส่วนย่อยเรียกว่า คาต้า

เฟรม (data frame) ซึ่งจะมีขนาดคงที่ประมาณสองหรือสามร้อยไบต์ หรืออย่างมากก็ไม่เกินสองถึงสามพันไบต์ ชุดของดาต้าเฟรมสำหรับข้อมูลที่ต้องการส่งไปให้ผู้รับก็จะถูกส่งไปทีละเฟรมตั้งแต่เฟรมแรกไปจนครบทุกเฟรม ข้างฝ่ายผู้รับจะตอบสนองโดยการส่งดาต้าเฟรมพิเศษ เรียกว่าเฟรมตอบรับ (acknowledge-ment frame) ไปถึงผู้ส่งเพื่อเป็นการบอกให้ทราบว่าได้รับข้อมูลครบแล้ว กระบวนการรับ-ส่งข้อมูลชุดนี้ก็จะเสร็จสิ้นสมบูรณ์

การรับ-ส่งข้อมูลในชั้นกายภาพนั้นจะไม่รับรู้ในเรื่องโครงสร้างข้อมูล คือ จะมองเห็นข้อมูลว่าเป็นบิต 0 หรือบิต 1 กลุ่มหรือชุดหนึ่งที่เรียงตามลำดับ เรียกว่า กระแสบิต (bit stream) จึงเป็นหน้าที่ของโปรแกรมในชั้นเชื่อมต่อข้อมูลจะต้องทำการตรวจสอบความถูกต้อง ซึ่งทำได้โดยการเพิ่มข้อมูลสำหรับการตรวจสอบติดไว้กับข้อมูลทุกเฟรม เช่น การเพิ่มข้อมูลส่วนหัวและส่วนหาง (header and tailler) เข้าไปกับทุกเฟรมซึ่งจะใช้เป็นตัวกำหนดขอบเขตของดาต้าเฟรมด้วย

การส่งข้อมูลผ่านระบบเครือข่ายใดๆก็ตามข้อมูลที่ส่งนั้นมีโอกาสที่จะเสียหายหรือสูญหายไปเลยก็ได้ โปรแกรมในชั้นเชื่อมต่อข้อมูลจะต้องสามารถตรวจสอบความผิดปกตินี้ได้เอง หรืออาจตอบสนองต่อการตรวจพบโดยโปรแกรมในชั้นกายภาพ เมื่อพบความผิดปกตินี้แล้วก็จะต้องมีวิธีการแก้ไข เช่น แจ้งให้ผู้ส่งข้อมูลได้ส่งข้อมูลชุดเดิมกลับมาใหม่ (เรียกเฟรมนี้ว่า duplicate frame) อย่างไรก็ตาม การส่งข้อมูลซ้ำทำให้เกิดปัญหาตามมาในกรณีที่ชุดข้อมูลไม่ได้สูญหายไปไหนเพียงแต่ใช้เวลาเดินทางมากกว่าปกติ ดังนั้นข้อมูลชุดเดียวกันก็จะมาถึงผู้ใช้ทั้งสองเฟรม โปรแกรมในชั้นนี้จึงต้องหาวิธีตรวจสอบและต้องกำจัดเฟรมที่ซ้ำออกไป

ปัญหาอื่นๆ ที่ต้องจัดการให้เรียบร้อย ได้แก่ การรักษาความสมดุลของ การรับ-ส่งข้อมูล เมื่อผู้ส่งพยายามส่งข้อมูลด้วยความเร็วสูงเกินกว่าที่ผู้รับจะทำงานได้ทัน หรือการแก้ปัญหาการช่วงชิงช่องสื่อสารระหว่างผู้รับและผู้ส่งในระบบการสื่อสารสายเดียวแต่สามารถส่งข้อมูลได้ทั้งสองทิศทาง ทั้งนี้เนื่องจากผู้รับนอกจากจะต้องรับข้อมูลแล้ว ยังจะต้องจัดส่งข้อมูลเฟรมตอบรับกลับไปยังผู้ส่งด้วย การแก้ปัญหาเหล่านี้จะได้กล่าวในรายละเอียดต่อไป

2.1.3 ชั้นสื่อสารควบคุมเครือข่าย (Network Layer)

ชั้นควบคุมเครือข่ายมีหน้าที่รับผิดชอบในการควบคุมการติดต่อรับ-ส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ เรียกว่าโหนด (Node) ต่างๆ ในระบบเครือข่ายให้เป็นไปด้วยความเรียบร้อย สิ่งที่สำคัญที่สุดคือการกำหนดเส้นทางเดินของข้อมูลจากโหนดผู้ส่งข้อมูลไปตามโหนดต่างๆ จนถึงโหนดผู้รับข้อมูลในที่สุด โฮสต์บางกลุ่มจะกำหนดเส้นทางเดินข้อมูลโดยศึกษาระบบเครือข่ายแล้วสร้างตารางเส้นทางเดินข้อมูลแบบถาวร โฮสต์บางกลุ่มจะกำหนดเส้นทางเดินข้อมูลในตอนเริ่มต้นของการสื่อสารดังนั้นการสื่อสารในครั้งต่อไป (ติดต่อกับโหนดเดิม) อาจจะเปลี่ยนไปใช้เส้นทางอื่น

ได้ โฮสต์ในกลุ่มที่มีวิธีการซับซ้อนมากจะกำหนดเส้นทางเดินข้อมูลในระดับแพ็กเก็ต กรณีที่มีผู้ส่งข้อมูลพร้อมๆกันหลายจุดจะทำให้เกิดความคับคั่งของข้อมูลคล้ายกับสภาวะการจราจรในชั่วโมงเร่งด่วนซึ่งมีปริมาณรถยนต์มากจนทำให้การจราจรติดขัด โฮสต์ในกลุ่มนี้ก็จะปรับเส้นทางเดินข้อมูลของแต่ละแพ็กเก็ตให้เหมาะสมกับสภาวะของระบบเครือข่ายอยู่ตลอดเวลา

การส่งผ่านข้อมูลในระบบเครือข่ายอาจมีการบันทึก ผู้ส่ง ผู้รับ และปริมาณข้อมูลที่ไหลผ่านโฮสต์หรือเราเตอร์ต่างๆ เพื่อประโยชน์ทางด้านการคิดค่าบริการ ซึ่งจะมีความซับซ้อนมากขึ้นถ้าข้อมูลไหลผ่านระบบเครือข่ายย่อยที่มีการคิดอัตราค่าบริการต่างกัน

เมื่อแพ็กเก็ตเดินทางผ่านเครือข่ายย่อยระบบหนึ่งไปยังอีกระบบหนึ่งอาจเกิดปัญหาความแตกต่างระหว่างกันในด้านต่างๆ อาทิเช่น การใช้กฎการสื่อสารข้อมูลไม่เหมือนกัน ปัญหาที่กล่าวถึงนี้เป็นความรับผิดชอบของโปรแกรมในชั้นควบคุมเครือข่ายที่จะต้องหาทางแก้ไขหรือปรับความแตกต่างระหว่างเครือข่ายต่างๆ ให้สามารถเข้าใจกันได้ แต่อย่างไรก็ตาม การส่งข้อมูลแบบกระจายข่าว (broadcasting) ที่มีใช้ในบางระบบนั้นจะไม่มีปัญหาที่กล่าวถึงอยู่เลยดังนั้นโปรแกรมในชั้นนี้จึงมีหน้าที่การทำงานน้อยมากหรืออาจไม่มีเลยก็ได้

2.1.4 ชั้นจัดการนำส่งข้อมูล (Transport Layer)

โปรแกรมในชั้นนำส่งข้อมูลมีหน้าที่หลักในการรับข้อมูลมาจากชั้นควบคุมหน้าต่างสื่อสาร ซึ่งอาจต้องแบ่งข้อมูลออกเป็นแพ็กเก็ตขนาดย่อม (ในกรณีที่ข้อมูลมีปริมาณมาก) หลายๆ แพ็กเก็ต แล้วจึงส่งข้อมูลทั้งชุดต่อไปให้โปรแกรมในชั้นควบคุมเครือข่าย ทางด้านโปรแกรมชั้นนำส่งข้อมูลของผู้รับก็จะทำหน้าที่ประกอบแพ็กเก็ตชุดนี้ให้กลับมารวมกันเป็นข้อมูลเดิม

ในภาวะปกติ การเชื่อมต่อการสื่อสารจะเป็นการจัดตั้งหน้าต่างสื่อสาร (session) ระหว่างผู้ส่งและผู้รับตามที่เกิดขึ้น ถ้าต้องการเพิ่มประสิทธิภาพก็อาจสร้างโพรเซสของโปรแกรมนำส่งข้อมูลขึ้นมาหลายๆ โพรเซสเพื่อช่วยกันจัดส่งข้อมูลให้เร็วขึ้นแต่ถ้าเน้นในด้านความประหยัดก็อาจทำในทางตรงกันข้ามนั่นคือ การยุบรวมโพรเซสหลายๆโพรเซสให้เหลือจำนวนน้อยลงแล้วจึงจัดการให้โพรเซสที่เหลืออยู่ทำการส่งข้อมูลทั้งหมดโดยการใช้ช่องสื่อสารร่วมกัน

โปรแกรมในชั้นนี้เป็นผู้กำหนดประเภทของการให้บริการต่างๆ รวมไปถึงการอำนวยความสะดวกในการใช้ระบบเครือข่ายซึ่งแบ่งออกได้เป็น 3 ประเภท ประเภทแรกเป็นการให้บริการแบบจุด-ต่อ-จุด โดยเน้นการรับประกันความถูกต้องของข้อมูลเป็นสำคัญ ประเภทที่ 2 เน้นการให้บริการข้อมูล ข้อมูลในระดับแพ็กเก็ตซึ่งแม้ว่าจะไม่รับประกันการสูญหายของข้อมูลแต่ก็ให้ความคล่องตัวสูกว่าแบบแรก เพราะว่าการรับประกันความถูกต้องของข้อมูลสามารถทำในชั้นอื่นได้ และ ประเภท

ที่ 3 เป็นการส่งข้อมูลแบบกระจายข่าวเพื่อประโยชน์ในการส่งข้อมูลชุดเดียวกันไปยังผู้ใช้หลายจุดพร้อมกัน

โปรแกรมในชั้นนำส่งข้อมูลติดต่อถึงกันผ่านช่องสัญญาณเสมือน (virtual channel) ระหว่างผู้ส่งและผู้รับ โดยตรงเรียกว่าเป็นการติดต่อแบบ end-to-end connection ในขณะที่โปรแกรมในสามชั้นแรกนั้นเป็นการติดต่อแบบจุด-ต่อ-จุด ซึ่งผู้รับอาจไม่ใช่ผู้รับข้อมูลที่แท้จริงแต่เป็นเพียงโหนดตัวกลางในการรับแล้วส่งข้อมูลต่อไปตามเส้นทางเดินข้อมูลที่ถูกกำหนดไว้ ดังรายละเอียดที่แสดงไว้ในรูป 2.1

เครื่องโฮสต์ส่วนมากจะใช้ระบบปฏิบัติการที่ให้บริการแบบมัลติโปรแกรมมิ่ง (multiprogramming) คือสามารถสร้างและใช้งานโพรเซสในชั้นการนำส่งข้อมูลได้หลายโพรเซสในขณะเดียวกัน จึงมีความจำเป็นที่จะต้องเพิ่มข้อมูลส่วนหัวเข้าไปกับข้อมูลแต่ละแพ็กเก็ตเพื่อบอกให้ระบบปฏิบัติการของโฮสต์ทราบว่าแพ็กเก็ตที่รับมานั้นเป็นของโพรเซสใด

นอกจากการใช้ช่องสื่อสารร่วมกันแล้วโปรแกรมในชั้นนำส่งข้อมูลจะต้องมีความสามารถในการจัดตั้งหน้าต่างสื่อสารกับโหนดอื่นๆ ในระบบเครือข่ายและจัดการยกเลิกเมื่อการสื่อสารสิ้นสุดลง โปรแกรมในชั้นนี้ยังต้องมีวิธีการกำหนดการตั้งชื่อให้แก่ตนเองและแนะนำให้ผู้อื่นในระบบได้รู้จัก รวมทั้งเรียนรู้การกำหนดที่อยู่ของโหนดอื่นได้ อีกสิ่งหนึ่งที่จำเป็นมากคือการควบคุมการไหลของข้อมูล (flow control) ซึ่งมีทั้งในระดับโฮสต์และระดับเราเตอร์โดยมีวัตถุประสงค์ในการควบคุมการรับและส่งข้อมูลโดยเฉพาะในกรณีที่ผู้ส่งจัดการส่งข้อมูลเร็วเกินกว่าผู้ใช้จะทำงานได้ทัน

2.1.5 ชั้นควบคุมหน้าต่างสื่อสาร (Session Layer)

ชั้นควบคุมหน้าต่างสื่อสารเป็นผู้กำหนดวิธีการควบคุมการเชื่อมต่อระหว่างผู้รับข้อมูลและผู้ส่งข้อมูลตั้งแต่เริ่มต้นการสื่อสารไปจนยุติการสื่อสาร เช่น การติดต่อขอใช้โฮสต์จากเครื่องคอมพิวเตอร์ที่อยู่ไกลออกไป (remote login) หรือการส่งแฟ้มข้อมูลระหว่างคอมพิวเตอร์ผ่านระบบเครือข่ายโดยภาพรวมแล้วการให้บริการในชั้นนี้จะคล้ายกับบริการที่มีให้ในชั้นนำส่งข้อมูล แต่ในชั้นนี้จะให้บริการหลายอย่างที่เป็นประโยชน์มากกว่าสำหรับการประยุกต์ใช้งานบางประเภท

หน้าที่สำคัญอย่างหนึ่งคือ บริหารการแลกเปลี่ยนข่าวสาร (dialogue control) อันได้แก่การกำหนดให้การแลกเปลี่ยนข่าวสารเป็นไปแบบสองทางในเวลาเดียวกัน (full duplex) หรือถ้าเป็นการสื่อสารทางเดียวแต่สลับทิศได้ (half duplex) ก็จะต้องเป็นผู้จัดลำดับให้ทั้งผู้ใช้และผู้ส่งทำการส่งข้อมูลได้คล้ายกับการควบคุมสับหลีกรถไฟ

สำหรับการสื่อสารประเภทที่ต้องใช้ โทเก้น (token) โปรแกรมในชั้นนี้จะเป็นผู้บริหารการใช้โทเก้นเพื่อให้โหนดต่างๆในระบบนั้นผลัดเปลี่ยนการครอบครองโทเก้นอย่างเป็นธรรม หรือตามลำดับความสำคัญ (priority)

หน้าที่อีกประการหนึ่งได้แก่การแก้ปัญหาความล้มเหลวในการส่งข้อมูลขนาดใหญ่มากระหว่างโหนดต่างๆ ในกรณีที่การส่งข้อมูลเกิดล้มเหลวกลางคันโดยไม่มีการแก้ไขใดๆ โหนดทั้งสองก็จะต้องเริ่มต้นใหม่หมด ถ้าเกิดการล้มเหลวขึ้นอีกก็จะต้องเริ่มต้นใหม่อีก วิธีการแก้ไขวิธีหนึ่งคือการแทรกจุดตรวจสอบความถูกต้อง (checkpoint) เข้าไปจำนวนหนึ่ง โดยขึ้นอยู่กับปริมาณข้อมูล ในระหว่างการส่งข้อมูล จุดตรวจสอบทั้งหมดจะต้องถูกแทรกเข้าไปในข้อมูลที่ตำแหน่งเดียวกันของทั้งผู้ส่งและผู้รับซึ่งเรียกว่าการ " synchronization" หากเกิดการล้มเหลวขึ้นโปรแกรมในชั้นนี้ของผู้รับก็จะค้นหาจุดตรวจสอบจุดสุดท้ายก่อนการล้มเหลวเพื่อลบข้อมูลส่วนที่อยู่หลังสุดจุดตรวจสอบนั้นทิ้งไป แล้วแจ้งให้ผู้ส่งเริ่มต้นการส่งข้อมูลใหม่จากจุดตรวจสอบนั้นแทนที่จะต้องเริ่มต้นใหม่ทั้งหมด

2.1.6 ชั้นสื่อสารนำเสนอข้อมูล (Presentation Layer)

โปรแกรมที่ทำงานในระดับชั้นควบคุมต้นๆที่กล่าวมานั้นจะให้ความสนใจในประสิทธิภาพของการรับ-ส่งข้อมูลและมองเห็นว่า ข้อมูลคือ กระแสบิต (bit stream) หรือกระแสไบต์ (byte stream) เท่านั้นโปรแกรมในชั้นนำเสนอข้อมูลจะมองข้อมูลว่าเป็นสิ่งที่มีรูปแบบ (syntax) และความหมาย (semantics) มากกว่ากระแสของบิตหรือไบต์ เช่น ข้อมูลที่เป็นตัวเลขบอกจำนวนเงิน ข้อมูลที่เป็นชื่อ และข้อมูลที่เป็นใบเรียกเก็บเงิน เป็นต้น ความแตกต่างของการให้ความหมายข้อมูลของเครื่องคอมพิวเตอร์ในระบบต่างๆ เป็นปัญหาที่จะต้องได้รับการแก้ไขในระดับส่วนรวมไม่ใช่ให้แต่ละฝ่ายแก้ปัญหาโดยลำพัง การควบคุมรูปแบบและความหมายของข้อมูล การใช้รหัสแทนข้อมูล เช่น รหัส ASCII หรือ Unicode หรือการแทนข้อมูลด้วยระบบ little endian หรือ big endian รวมถึงการเข้ารหัส และการถอดรหัสข้อมูล สิ่งต่างๆที่กล่าวมานี้ล้วนแต่เป็นความรับผิดชอบของโปรแกรมในชั้นนี้

2.1.7 ชั้นสื่อสารการประยุกต์ (Application Layer)

ในปัจจุบัน มีจอภาพเทอร์มินัล (Terminals) อยู่หลายร้อยชนิดทั่วโลก ซึ่งส่วนใหญ่จะไม่สามารถใช้ทดแทนหรือใช้งานร่วมกันได้ การติดต่อระหว่างเครื่องคอมพิวเตอร์ที่อยู่คนละระบบเครือข่ายย่อยจึงไม่สามารถสื่อสารกันได้โดยสมบูรณ์ โปรแกรมในชั้นการประยุกต์จึงเข้ามามีบท

บาทสำคัญสองด้าน คือ การเป็นตัวกลาง หรือส่วนติดต่อระหว่างโปรแกรมประยุกต์ (application programs) กับโปรแกรมใน 6 ชั้นที่เหลือ และการกำหนดแบบมาตรฐานของจอ (terminal type)

การกำหนดแบบมาตรฐานของจอนั้นไม่ได้เป็นการกำหนดวิธีการสร้างจอเทอร์มินัลให้เหมือนกันแต่จะคล้ายกับการสร้างจอเทอร์มินัลเสมือน (virtual terminal) ขึ้นบนจอเทอร์มินัลจริง ทั้งนี้เพื่อทำให้จอเทอร์มินัลทุกชนิดในโลกมีความเข้าใจตรงกัน เช่น ขนาดบริเวณที่ในการแสดงผล การเคลื่อนย้ายตำแหน่งเคอร์เซอร์ (cursor) และการแสดงตำแหน่งตัวอักษร ณ ตำแหน่งต่างๆ บนจอภาพ เป็นต้น จึงทำให้การใช้จอเทอร์มินัลเพื่อการสื่อสารบนระบบเครือข่ายเกิดขึ้นได้แม้ว่าจะใช้จอเทอร์มินัลต่างแบบกันก็ตาม

การสำเนาแฟ้มข้อมูลหรือการคัดลอก (copy) แฟ้มข้อมูลผ่านระบบเครือข่ายก็อาจเกิดปัญหาได้ ตัวอย่าง เช่น ในระบบปฏิบัติการหลายระบบมีวิธีการกำหนดชื่อแฟ้มข้อมูลที่แตกต่างไปจากวิธีที่ใช้ในระบบอื่น จึงจำเป็นจะต้องมีการแก้ไขหรือปรับแต่งส่วนที่ต่างกันให้สามารถเข้ากันได้ ถ้าหากปล่อยให้เป็นหน้าที่ของผู้ใช้โดยตรงแล้วก็จะทำให้เกิดความยุ่งยากมาก ทั้งนี้เพราะผู้ใช้บางส่วนอาจไม่มีความรู้มากพอที่จะแก้ไขได้ นอกจากนี้การส่งจดหมายอิเล็กทรอนิกส์ การใช้เทอร์มินัลสำหรับป้อนข้อมูลจากระยะไกล หรือการดูบัญชีรายชื่อแฟ้มข้อมูล (ในเครื่องคอมพิวเตอร์อื่น) ล้วนเป็นหน้าที่ของโปรแกรมชั้นการประยุกต์ที่จะต้องอำนวยความสะดวกให้แก่ผู้ใช้ทั้งสิ้น

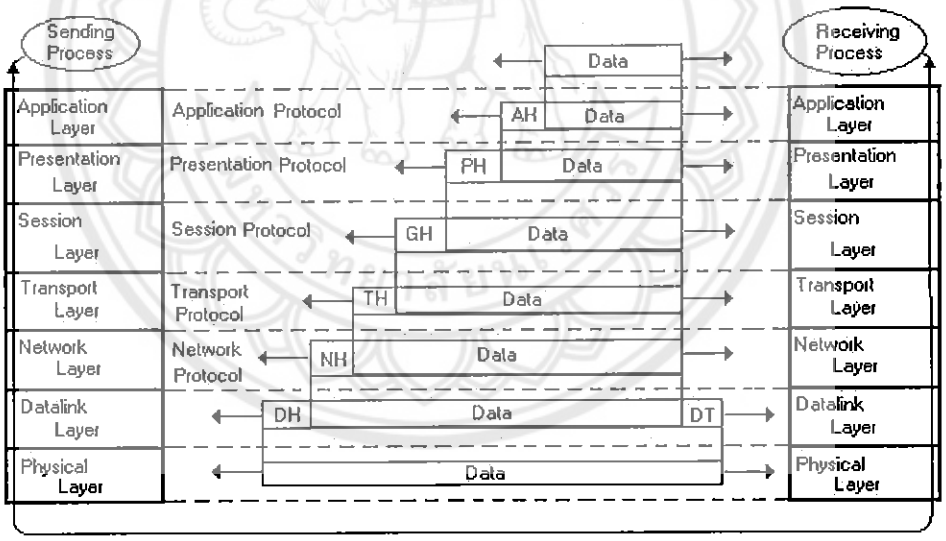
2.2 การส่งข้อมูลในรูปแบบมาตรฐานของ OSI

รูป 2.2 นั้นแสดงให้เห็นขั้นตอนที่ข้อมูลได้ถูกส่งจากผู้ส่งข้อมูลไปให้ผู้รับข้อมูล กระบวนการส่งข้อมูลเริ่มต้นจากโพรเซสที่ต้องการส่งข้อมูลส่งมอบข้อมูลให้กับโปรแกรมชั้นการประยุกต์ ซึ่งจะเติมข้อมูลส่วนหัว (คือส่วนAHในรูป 2.2) ที่ต้องการเข้าไปกับข้อมูลจริงแล้วส่งต่อไปให้โปรแกรมชั้นนำเสนอข้อมูล โปรแกรมในชั้นนี้จะเข้าใจว่าข้อมูลที่ส่งมาทั้งหมดนั้นคือข้อมูลจริง (จะไม่ทราบว่า มีข้อมูลส่วน AH แทรกเข้ามาด้วยแล้ว) จึงเพิ่มเติมข้อมูลในส่วนที่ตัวเองต้องการลงไปกับข้อมูลทั้งหมด (คือส่วน PH ในรูป 2.2) แล้วส่งต่อให้กับโปรแกรมในชั้นกำหนดหน้าต่างสื่อสาร กระบวนการนี้จะทำซ้ำไปเรื่อยๆเมื่อผ่านโปรแกรมในแต่ละชั้นสื่อสารจนถึงชั้นกายภาพซึ่งจะมองเห็นว่า ข้อมูลที่ส่งมาจากชั้นเชื่อมต่อข้อมูลนั้นเป็นเพียง กระแสบิต (bit stream) ที่จะต้องส่งออกไปเท่านั้น

ทางด้านผู้ใช้ข้อมูลจะเริ่มกระบวนการรับข้อมูลโดยชั้นกายภาพรับข้อมูลเข้ามาในลักษณะของกระแสบิตแล้วส่งต่อไปให้โปรแกรมชั้นเชื่อต่อข้อมูล โปรแกรมนี้จะอาศัยข้อมูลส่วนหัว (DH) เพื่อทำความเข้าใจกับลักษณะของข้อมูลที่รับมารวมทั้งการตรวจสอบความถูกต้อง เสร็จแล้วจึงลบข้อมูลส่วนนี้ออกไปและส่งข้อมูลที่เหลือไปให้กับโปรแกรมในชั้นควบคุมเครือข่าย กระบวนการก็

จะเป็นไปอย่างนี้จนในที่สุดข้อมูลก็ถูกส่งาถึงชั้นการประยุกต์ ในชั้นนี้ข้อมูลส่วนหัวที่โปรแกรมในแต่ละชั้น (ของผู้ส่ง) เพิ่มเติมเข้าไปนั้นได้ถูกโปรแกรมในชั้นนั้นๆ (ของผู้ใช้) ดึงออกไปใช้จนเหลือแต่ข้อมูลส่วนหัวของชั้นการประยุกต์ (AH) เท่านั้น โปรแกรมในชั้นนี้ก็จะดึงข้อมูลส่วนสุดท้ายนี้ออกไปใช้งานแล้วจึงส่งข้อมูลที่เหลืออยู่ (คือข้อมูลจริง) ให้กับโพรเซสของผู้รับนำข้อมูลไปใช้งานต่อไป

แนวความคิดที่เป็นหลักสำคัญของกระบวนการนี้คือ การที่โปรแกรมในแต่ละชั้นนั้นคิดว่าการสื่อสารเป็นไปตามแนวนอน แม้ว่าในความเป็นจริงนั้นจะเป็นไปตามแนวตั้งจะเป็นไปตามแนวตั้งดังที่แสดงในรูป2.2 ทั้งนี้หมายความว่าโปรแกรมในแต่ละชั้นของทางด้านผู้ส่งจะแทรกข้อมูลที่ตนเองต้องการเข้าไปกับข้อมูลโดยมีวัตถุประสงค์ให้โปรแกรมในชั้นเดียวกันของทางผู้รับได้นำข้อมูลนั้นไปใช้ จึงเสมือนกับว่าโปรแกรมในแต่ละชั้นของทางผู้ส่งจะทำงานกับโปรแกรมในชั้นเดียวกันของทางฝ่ายผู้รับ โดยไม่สนใจว่าจะมีโปรแกรมในชั้นอื่นๆ อยู่ด้วย การทำงานของโปรแกรมในแต่ละชั้นจึงเป็นอิสระจากโปรแกรมในชั้นอื่นๆ อย่างแท้จริง และจะปฏิบัติกับข้อมูลที่รับมาราวกับว่าเป็นข้อมูลจริงทั้งหมด (แม้ว่าจะมีข้อมูลของชั้นบนๆ แทรกเข้ามาด้วยก็ตาม)



รูปที่2.2 ตัวอย่างการทำงานของชั้นสื่อสารในรูปแบบโอเอสไอ
(ที่มา:เครือข่ายคอมพิวเตอร์ สัลลยุทธ์ สว่างสวรรณ)

2.3 พิธีการ ทีซีพี/ไอพี (TCP/IP Protocol)

2.3.1 พิธีการ(Protocol)

พิธีการ (Protocol) คือระเบียบวิธีที่กำหนดขึ้นสำหรับการสื่อสารข้อมูล ให้สามารถส่งผ่านข้อมูลไปยังปลายทางได้อย่างถูกต้อง ในปัจจุบันพิธีการในการสื่อสารข้อมูลก็มีอยู่หลายพิธีการนอกเหนือจากทีซีพี/ไอพี (TCP/IP) เช่น พิธีการไอพีเอ็กซ์/เอสพีเอ็กซ์(IPX/SPX) , พิธีการเน็ทไบออส(NetBIOS) และ พิธีการแอปเปิ้ลทอล์ค(Apple Talk) เป็นต้น

2.3.2 พิธีการทีซีพี/ไอพี (TCP/IP)

พิธีการทีซีพี/ไอพี (TCP/IP) เป็นชื่อเรียกของชุดพิธีการที่สำคัญ มีการใช้งานกันอย่างแพร่หลายตามการขยายตัวของอินเตอร์เน็ต/อินทราเน็ต ความจริงแล้วพิธีการทีซีพี/ไอพี(TCP/IP) เป็นกลุ่มพิธีการหลายตัวที่ประกอบกันเป็นชุดให้ใช้งาน โดยมีส่วนประกอบหลัก ๆ 2 ส่วนคือ

TCP ย่อมาจาก Transmission Control Protocol มีหน้าที่ในการตรวจสอบการรับส่งข้อมูลระหว่างคอมพิวเตอร์ผู้รับและผู้ส่ง ให้ได้รับข้อมูลอย่างถูกต้องครบถ้วน หากข้อมูลสูญหายก็จะแจ้งให้ต้นทางส่งข้อมูลมาใหม่

IP ย่อมาจาก Internet Protocol มีหน้าที่เลือกเส้นทางที่ใช้รับส่งข้อมูลผ่านระบบเครือข่าย และตรวจสอบแอดเดรส (Address) ของผู้รับโดยใช้ข้อมูลขนาด 4 ไบต์หรือ 32 บิต เป็นตัวกำหนดแอดเดรส (Address) ของผู้รับเรียกว่าไอพีแอดเดรส (IP Address)

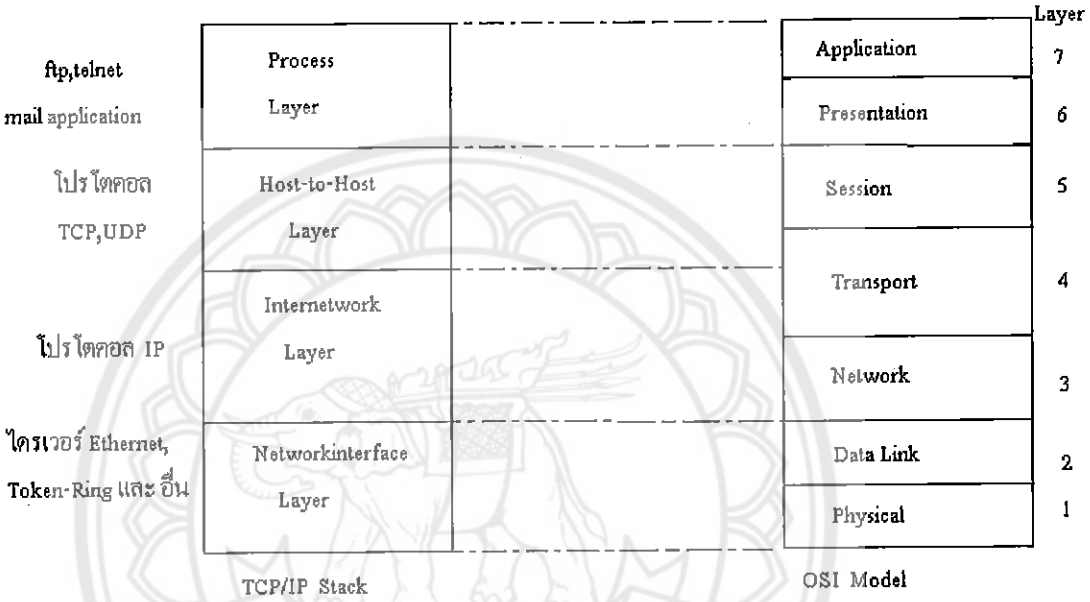
2.3.3 โครงสร้างของพิธีการทีซีพี/ไอพี (TCP/IP)

พิธีการทีซีพี/ไอพี (TCP/IP) มีกลไกการทำงานเป็นชั้นหรือเลเยอร์ (layer) เรียงต่อกัน โดยในแต่ละเลเยอร์ (layer) จะมีการทำงานเทียบได้กับโมเดลของโอเอสไอ (OSI model) มาตรฐาน แต่บางเลเยอร์ (layer) ของ พิธีการทีซีพี/ไอพี (TCP/IP) จะทำงานเทียบกับโอเอสไอ (OSI) หลาย เลเยอร์ ปนกัน ซึ่งในแต่ละ เลเยอร์ ของพิธีการ TCP/IP จะประกอบด้วย

- โปรเซสเลเยอร์ (Process layer)
- โฮสทูโอสเลเยอร์ (Host-to-Host layer)
- อินเตอร์เน็ตเวิร์ก (Internetwork layer)
- เน็ตเวิร์กเลเยอร์ (Network Interface layer)

โดยเมื่อเทียบกับมาตรฐานโอเอสไอ (OSI model) แล้วจะเป็นดังรูปที่ 3.1 ซึ่งเราจะเห็นว่าบางกลไกของพิธีการทีซีพี/ไอพี (TCP/IP) เทียบได้กับมาตรฐานโอเอสไอ (OSI model) สองชั้น

หรือบางกลไกก็จะทำงานคาบเกี่ยวกันระหว่างบางชั้นของ มาตรฐานโอเอสไอ ตัวอย่างเช่น กลไกการทำงานของพิธีการ ทีซีพี/ไอพี ในส่วนเน็ตเวิร์กอินเตอร์เฟซเลเยอร์ (Network Interface layer) เมื่อเทียบกับ มาตรฐานโอเอสไอ จะเทียบได้กับดาต้าลิงค์เลเยอร์ (Data Link layer) และฟิสิคอลเลเยอร์ (Physical layer) 2 ชั้นรวมกัน เป็นต้น ในแต่ละกลไกของพิธีการ ทีซีพี/ไอพี จะมีพิธีการอื่น ๆ ในชุดของ ทีซีพี/ไอพี ร่วมทำงานอยู่ด้วย ซึ่งจะกล่าวโดยละเอียดต่อไป



รูปที่ 2.3 ทีซีพี/ไอพี สแตค เปรียบเทียบกับมาตรฐานโอเอสไอ

(ที่มา : เปิดโลกของ TCP / IP และโปรโตคอลของอินเตอร์เน็ต สุวัฒน์ ปุณณชัยยะ)

◆ โปรเซสเลเยอร์ (Process layer)

จากรูปแสดงลำดับชั้นการทำงานของพิธีการ ทีซีพี/ไอพี เทียบกับมาตรฐาน โอเอสไอ โมเดล นั้น ในชั้นบนสุดเรียกว่าโปรเซสเลเยอร์ (Process layer) ทำงาน 2 หน้าที่เทียบได้กับแอพพลิเคชันเลเยอร์ (Application layer) และพรีเซนต์เตชันเลเยอร์ (Presentation layer) ในชั้นนี้จะรองรับการทำงานของแอพลิเคชันต่าง ๆ ที่ทำงานเป็นโปรเซส อยู่ในเครื่องเซิร์ฟเวอร์ให้บริการและเครื่องที่ขอใช้บริการหรือไคลเอนต์ (client) ซึ่งจะติดต่อกันผ่านพิธีการเฉพาะแอพลิเคชันอีกทีหนึ่ง ตัวอย่างเช่น เมื่อผู้ใช้งานอินเตอร์เน็ตต้องการถ่ายโอนข้อมูลหรือดาวน์โหลด (download) ข้อมูลจากเครื่องเซิร์ฟเวอร์ที่ให้บริการ โดยอาจจะเรียกใช้โปรแกรมเอฟทีพี ไคลเอนต์ (ftp client) ทั่วไป เช่น โปรแกรม WS_ftp ติดต่อกับโปรเซสเอฟทีพี (ftp) ที่กำลังให้บริการอยู่ที่เครื่องเซิร์ฟเวอร์ จากนั้นตัวโปรเซส เอฟทีพี (ftp) ก็จะเรียกให้พิธีการเอฟทีพี (FTP :File Transfer Protocol) เพื่อทำการโอน

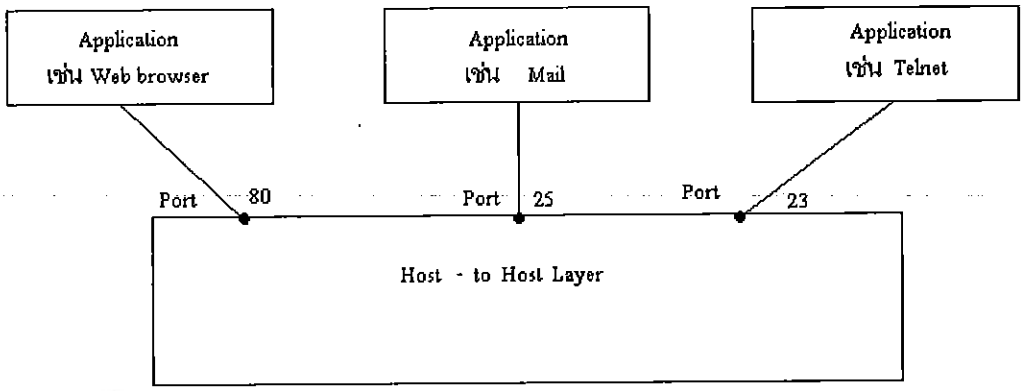
ถ่ายไฟล์นี้ หรือถ้าผู้ใช้ต้องการเรียกใช้งานคอมพิวเตอร์ที่เครื่องที่อยู่ห่างไกลออกไปด้วยการใช้โปรแกรม เทลเน็ต (telnet) ที่เครื่องเซิร์ฟเวอร์ให้บริการ ตัวโปรเซสเทลเน็ต (telnet) ที่ทำงานอยู่ก็จะเรียกใช้พิธีการเทลเน็ต (Telnet) เพื่อติดต่อกัน หรือในกรณีที่มีการเรียกใช้โปรแกรมเว็บเบราเซอร์ (web browser) เช่น เน็ตเคปเนวิเกเตอร์ (Netcape Navigator) เพื่อเรียกดูเว็บเพจในเว็บไซต์ ซีเอ็นเอ็น (CNN) ที่เครื่องซึ่งให้บริการเว็บของซีเอ็นเอ็น (CNN) ก็จะมีโปรเซสเอชทีทีพี (HTTP: HyperText Transfer Protocol) ทำงานอยู่และจะติดต่อกับผู้ใช้ผ่านพิธีการเอชทีทีพี (HTTP) เป็นต้น

การทำงานของแอพลิเคชันต่าง ๆ จะอยู่ที่โปรเซสเลเยอร์ (Process layer) นี้ และมีการติดต่อกันตามแต่ละพิธีการเฉพาะ แล้วแต่แอพลิเคชันที่ใช้งาน จากการที่ โปรเซสเลเยอร์ ของ ทีซีพี/ไอพี รองรับให้พิธีการอื่นทำงานได้หลายโปรเซสและหลายพิธีการได้พร้อมกันนั้น ทำให้ผู้ใช้สามารถเปิดโปรแกรมใช้งานได้หลาย ๆ อย่างพร้อมกัน เช่น เปิดโปรแกรมอินเตอร์เน็ต เอ็กซ์พลอเรอร์ (Internet Explorer) เพื่อเรียกดูเว็บเพจ พร้อมกับใช้งานโปรแกรมเอาต์ลุค เอ็กซ์เพลส (Outlook Express) เพื่อรับส่งอีเมล์ไปพร้อมกันได้โดยไม่ต้องรอให้ทำงานอย่างหนึ่งอย่างใดเสร็จก่อน หรือในปัจจุบันมีการพัฒนาโปรแกรมเว็บเบราเซอร์ (web browser) ให้สามารถเรียกใช้งานพิธีการอื่น ๆ ได้มากขึ้น ทำให้เราสามารถใช้โปรแกรมเว็บเบราเซอร์(web browser) โอนถ่ายไฟล์ข้อมูลที่ใช้พิธีการเอฟทีพี (FTP) ได้โดยไม่ต้องไปหาโปรแกรมอื่นมาใช้

พิธีการหลัก ๆ ทำงานในโปรเซสเลเยอร์ ได้แก่ เอฟทีพี (FTP :File Transfer Protocol), เอทีทีพี(HTTP :Hyper Text Transfer Protocol),เทลเน็ต (Telnet)

◆ โฮสทูโฮสเลเยอร์ (Host-to-Host layer)

การทำงานที่ชั้นของ โฮสทูโฮสเลเยอร์ นี้จะมีบทบาทในการจัดการต่อจาก โปรเซสเลเยอร์ บางครั้งเรามักเรียกชั้นโฮสทูโฮส (Host-to-Host) ว่าเป็นทรานสปอร์ตเลเยอร์ (Transport layer) ซึ่งไม่ใช่ชั้นของทรานสปอร์ตเลเยอร์ (Transport layer) ในมาตรฐานโอเอสไอโมเดล การทำงานของโฮสทูโฮสเลเยอร์ นี้ จะมีการสร้างคอนเนคชัน (connection) หรือการเชื่อมต่อกันระหว่างแอพพลิเคชันกับ โฮสทูโฮสเลเยอร์ โดยจุดที่เชื่อมเพื่อรับส่งข้อมูลนี้เรียกว่า พอร์ท หรือซ็อกเก็ต (socket) (คำว่า พอร์ท ในที่นี้ไม่ได้หมายถึง พอร์ท ทางฮาร์ดแวร์) และในแต่ละแอพพลิเคชันก็จะสร้างการเชื่อมต่อผ่าน พอร์ท ได้พร้อมกันหลายแอพพลิเคชัน ซึ่งในการใช้งาน พอร์ท ของแต่ละแอพพลิเคชันที่อยู่ในชั้น โปรเซสเลเยอร์ จะแตกต่างกันตามหมายเลขที่กำหนดไว้ และแต่ละพิธีการจะมีการใช้งานพอร์ท หมายเลขต่าง ๆ ไม่ซ้ำกัน ดังรูปที่ 2.4

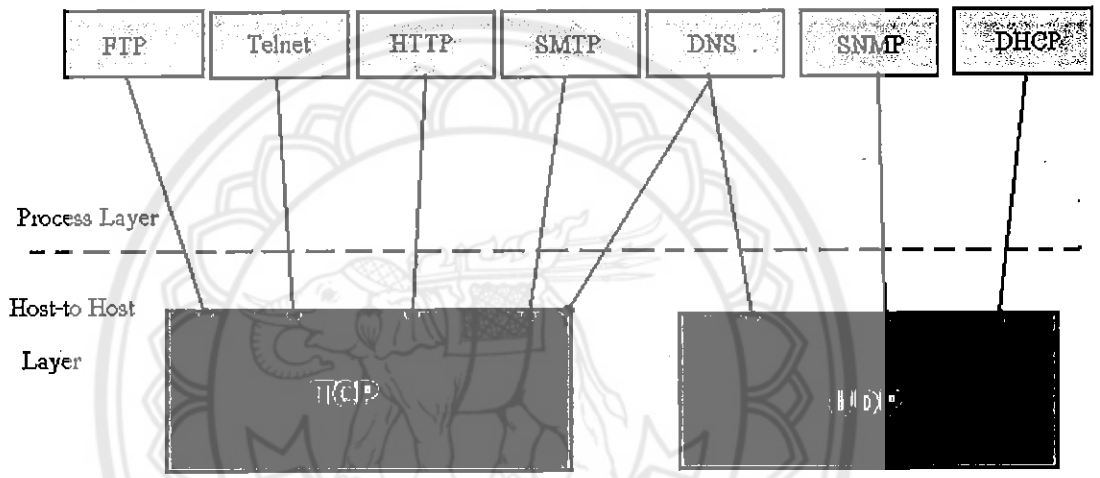


รูปที่ 2.4 แอพพลิเคชันหรือโปรเซสต่าง ๆ สื่อสารกับ โฮสทูโฮสเลเยอร์ ผ่านจุด**เชื่อมต่อหรือ** พอร์ท ส่วนหมายเลขในรูปคือหมายเลข พอร์ท ที่โปรเซสใช้งาน เช่น เว็บหรือโปรเซส**เอชทีทีพี** ใช้งาน พอร์ท 80 ในการส่งผ่านข้อมูล

(ที่มา : เปิดโลกของ TCP / IP และโปรโตคอลของอินเตอร์เน็ต สุวัฒน์ **ปุณณชัยยะ**)

เมื่อแอพพลิเคชันทำงานผ่านพิธีการในชั้น โปรเซสเลเยอร์ จะมีการ**ส่งผ่าน** โฮสทูโฮสเลเยอร์ ที่ชั้นนี้จะมีการเชื่อมต่อผ่าน พอร์ท ที่กำหนดทำให้การรับส่งข้อมูลในแ**ต่ละพิธีการทำได้ถูกต้อง** ถึงแม้ว่าในเครื่องเซิร์ฟเวอร์ที่ให้บริการจะมีการทำงานอยู่หลายโปรเซส**ที่แตกต่างกันก็ตาม หรือมี**ผู้ใช้บริการเข้ามาใช้งานพร้อมกันจำนวนมากและหลายแอพพลิเคชันในเวลาเดียวกัน ในชั้นโฮสทูโฮส (Host-to-Host) หรือทรานสปอร์ตเลเยอร์ (Transport layer) ของ **ทีซีพี/ไอพี นี้ จะมีพิธีการทำงานอยู่** 2 พิธีการที่แตกต่างกัน คือ พิธีการทีซีพี (TCP) และพิธีการ**ยูดีพี** (UDP :User Datagram Protocol) ในการส่งผ่านข้อมูลลงไปที่ชั้นถัด ๆ ไป เราจะเห็นว่าพิธีการทีซีพี (TCP) และยูดีพี (UDP) จะถูกผลึกเข้าไปในพิธีการไอพี (IP) อีกทีหนึ่ง และส่งต่อไปยังเครือข่ายอินเตอร์เน็ตต่อไป

ตัวพิธีการทีซีพี (TCP) และพิธีการยูดีพี (UDP) จะมีแอพพลิเคชันเฉพาะเพื่อใช้งานแยกกัน คือ แอพพลิเคชันที่ใช้พิธีการเอฟทีพี(FTP),เทลเน็ต (Telnet), เอชทีทีพี (HTTP) และ เอชเอ็มทีพี (SMTP) จะมีการส่งผ่านข้อมูลโดยเรียกใช้พิธีการทีซีพี (TCP) ส่วนแอพพลิเคชันที่ใช้พิธีการ เอส เอ็นเอ็มพี (SNMP) และดีเอชซีพี (DHCP) จะส่งผ่านข้อมูลโดยเรียกใช้พิธีการยูดีพี (UDP) และ สำหรับพิธีการดีเอ็นเอส (DNS) นั้น จะสามารถเรียกใช้งานทั้งทีซีพี และ ยูดีพี ดังรูป ซึ่งมีเหตุผลที่มีการเรียกใช้พิธีการทีซีพี และ ยูดีพี แตกต่างกัน ก็เนื่องจากวิธีการทำงานของทั้งสองพิธีการต่างกันนั่นเอง



รูปที่ 2.5 โปรเซสต่าง ๆ ที่เรียกใช้ทรานสปอร์ตเลเยอร์ (Transport layer) เพื่อส่งผ่านข้อมูลโดยอาศัย พอร์ต ซึ่งในแต่ละโปรเซสจะเรียกใช้งาน พอร์ท เฉพาะแตกต่างกัน ยกเว้นดีเอ็นเอส (DNS) ที่สามารถใช้งานได้ทั้ง ทีซีพี และ ยูดีพี

(ที่มา : เปิดโลกของ TCP / IP และโปรโตคอลของอินเตอร์เน็ต สุวัฒน์ ปุณณชัยยะ)

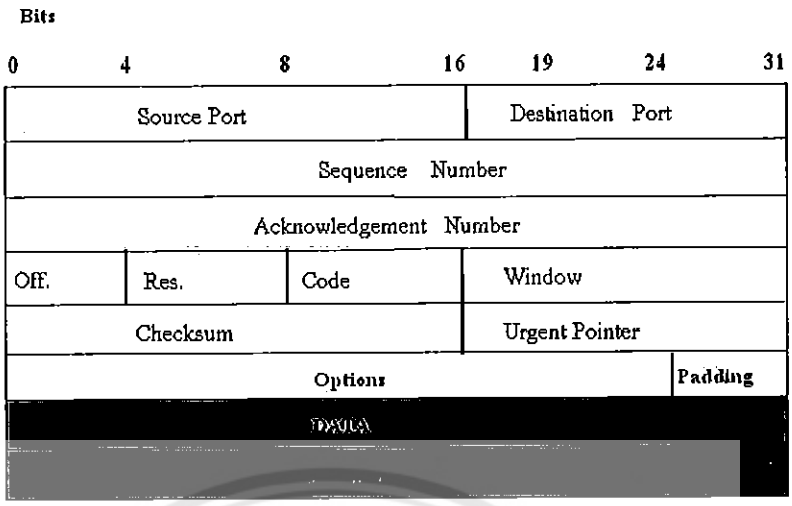
พิธีการทีซีพี (TCP:Transmission Control Protocol) เป็นพิธีการที่มีการรับส่งข้อมูลแบบ stream oriented protocol หมายความว่า การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่งไป แต่ละแบ่งข้อมูลเป็นส่วนย่อย ๆ ก่อน แล้วจึงจะส่งไปยังปลายทางอย่างต่อเนื่องเป็นลำดับข้อมูล ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไป ก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้ง สำหรับปลายทางก็จะทำหน้าที่จัดเรียงส่วนของข้อมูลดาต้าแกรม (datagram) ใหม่ให้ต่อเนื่องกัน และประกอบกลับเป็นข้อมูลทั้งหมดได้ ซึ่งจะแยกข้อมูลที่ไม่ถูกต้องออก ดังนั้นแอพพลิเคชันหรือโปรเซสใดที่อาศัยการ

ส่งผ่านข้อมูลด้วยพิธีการทีซีพี จะต้องใช้หน่วยความจำหรือขนาดของช่องสัญญาณ (bandwidth) มากกว่ายูดีพี

การติดต่อระหว่างกันจะต้องเป็นแบบคอนเนคชันออเรียนเตต (connection-oriented) คือ ต้องมีการสร้างการติดต่อกันเป็นเซสชัน (session) ทั้ง 2 ด้านเสียก่อน แล้วจึงจะรับส่งข้อมูลไปได้พร้อมกัน (full duplex) เหมือนกับการใช้โทรศัพท์ติดต่อกัน เมื่อผู้ติดต่อต้นทางเรียกให้ฝ่ายตรงข้ามรับสายแล้ว จึงเริ่มการสนทนา เช่น พูดคำว่า "สวัสดี" หรือ "ฮัลโหล" กันก่อนเพื่อให้แน่ใจว่าฝ่ายตรงข้ามพร้อมจะติดต่อด้วย จากนั้นจึงเริ่มต้นติดต่อกัน และเมื่อต้องการจะเลิกการติดต่อก็จะมีการพูดคำว่า "สวัสดี" ให้ฝ่ายตรงข้ามทราบว่าจะเลิกการติดต่อและวางสายไป ซึ่งในระหว่างการติดต่อกันนั้น แม้ว่าฝ่ายใดฝ่ายหนึ่งหรือทั้งสองฝ่ายจะเงียบไป คือ ไม่พูดอะไรเป็นเวลานาน ๆ แต่การเชื่อมโยงระหว่างทั้งสองด้านยังคงมีอยู่ไม่ขาดไปจนกว่าฝ่ายใดฝ่ายหนึ่งจะวางสาย เช่นเดียวกับการติดต่อกันด้วยกลไกพิธีการทีซีพี เมื่อแอพพลิเคชันต้องการส่งผ่านข้อมูลจะใช้พิธีการที่เหมาะสมในชั้นโปรเซสเลเยอร์ติดต่อไป และมีการสร้างช่องข้อมูลผ่าน พอร์ท ที่กำหนดเพื่อส่งผ่านข้อมูลไปยังพิธีการ ทีซีพี

ในระหว่างการรับส่งข้อมูลนี้ พิธีการ ทีซีพี จะเพิ่มขบวนการสอบทานข้อมูล เพื่อให้ข้อมูลมีความถูกต้องไม่ผิดพลาดไปจากเดิม โดยการส่งสัญญาณสอบทานข้อมูล (acknowledgement) และส่งข้อมูลให้ใหม่อีกครั้ง ถ้าปลายทางไม่ได้รับหรือเกิดความผิดพลาดขึ้น

ความน่าเชื่อถือของการส่งผ่านข้อมูลโดยพิธีการ ทีซีพี จะมีมากกว่า แต่ก็ต้องอาศัยทรัพยากรของระบบมากกว่าในการทำงานเช่นกัน



รูปที่ 2.6 รูปแบบของทีซีพีแพคเก็ต (TCP packet) จะเห็นว่ามีฟิลด์ Acknowledgement Number และข้อมูลเช็คซัม (Checksum) เพื่อใช้ตรวจสอบการเดินทางของข้อมูล ส่วนเฮดเดอร์มีข้อมูลมากทำให้ต้องอาศัยทรัพยากรของระบบทำงานมาก

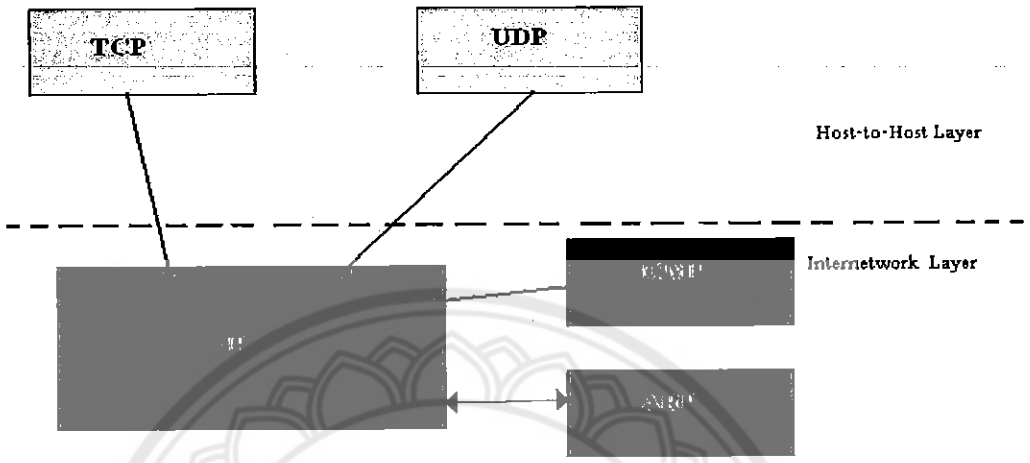
(ที่มา : เปิดโลกของ TCP / IP และโปรโตคอลของอินเตอร์เน็ต สุวัฒน์ ปุณณชัยยะ)

◆ อินเตอร์เน็ตเวิร์กเลเยอร์ (Internetwork Layer)

ในระดับล่างต่อมาในชั้นอินเตอร์เน็ตเวิร์กเลเยอร์ (Internetwork Layer) มีหน้าที่ส่งผ่านข้อมูลในระหว่างเครือข่าย โดยมีพิธีการที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใด ๆ บนอินเตอร์เน็ต คือ พิธีการไอพี (IP :Internet Protocol) นอกจากนี้ในชั้นอินเตอร์เน็ตเวิร์กเลเยอร์ (Internetwork Layer) ยังมีพิธีการทำงานอยู่ด้วยกัน 2 ชนิดคือ พิธีการไอซีเอ็มพี(Internet Control Message Protocol :ICMP) และพิธีการเออาร์พี (Address Resolution Protocol :ARP)

พิธีการไอพี(IP) ทำหน้าที่ให้บริการส่งผ่านข้อมูลที่มาจากโฮสทูโฮสเลเยอร์ เพื่อส่งข้ามไปยังเครือข่ายใด ๆ ได้อย่างถูกต้อง แม้ว่าจะมีเครือข่ายเชื่อมต่อกันอยู่ในอินเตอร์เน็ตเป็นล้าน ๆ เครือข่ายก็ตาม เนื่องจากพิธีการไอพี มีข้อมูลตำแหน่ง ไอพี ปลายทางที่จะส่งไปให้ โดยทำงานร่วมกับอุปกรณ์เราเตอร์ (Router) เพื่อส่งข้อมูลข้ามเครือข่ายออกไปได้ ตัวพิธีการไอพี จะทำงานแบบแพคเก็ตสวิทชิ่ง (packet switching) คือมีการส่งข้อมูลผ่านสวิทช์ (switch) ไปยังปลายทาง โดยข้อมูลจะเดินทางไปยังเครือข่ายต่าง ๆ ผ่านสวิทช์นี้ไปเรื่อย ๆ จนกว่าจะถึงปลายทาง ตัววงจรผ่านหรือสวิทช์ (switch) นี้อาจเป็นเกทเวย์ (Gateway) หรือเราเตอร์ (Router) ในระบบเครือข่ายก็ได้ ซึ่งในข้อมูลของพิธีการไอพี จะมีข้อมูลหมายเลขไอพี ปลายทางที่จะส่งข้อมูลไป และเมื่อถึงเครือข่ายปลายทางแล้ว จะมีกลไกแปลงหมายเลข ไอพี ให้เป็นหมายเลขฮาร์ดแวร์ประจำเครื่องที่ถูกต้องอีกทีหนึ่งด้วย

พิธีการเออาร์พี (ARP) ตามรูปที่ 2.8 ที่จะแสดงการติดต่อกันระหว่างพิธีการในชั้นของโฮสทูโฮสเลเยอร์ และ อินเตอร์เน็ตเวิร์กเลเยอร์

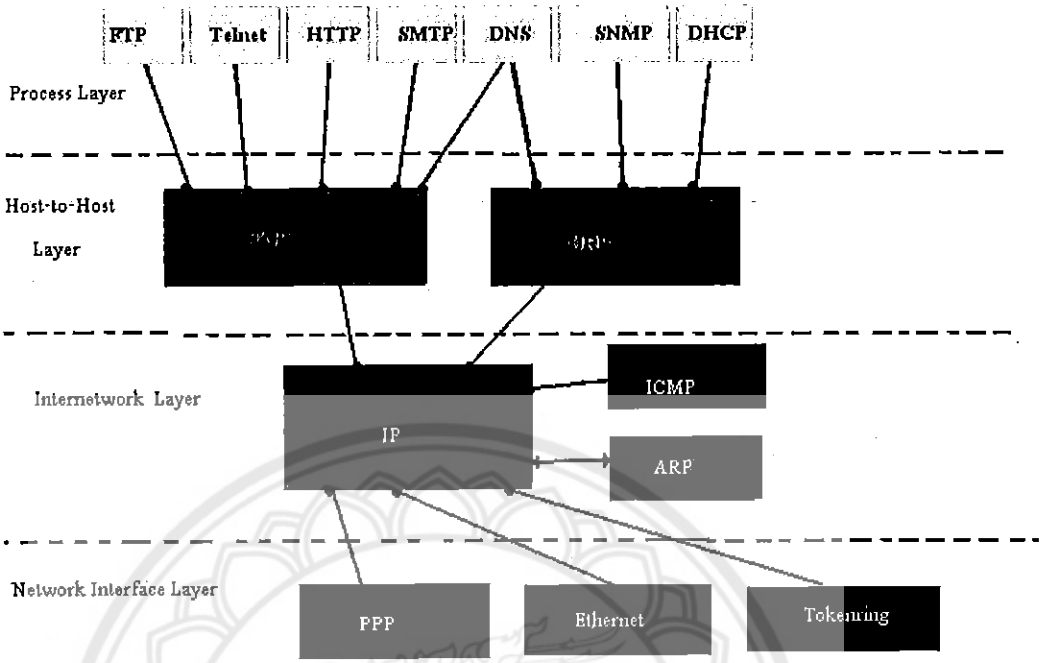


รูปที่ 2.7 พิธีการทีซีพีและยูดีพี อาศัยพิธีการไอพี ที่อยู่ชั้นล่างเพื่อส่งผ่านข้อมูลระหว่างเครือข่าย
(ที่มา : เปิดโลกของ TCP / IP และโปรโตคอลของอินเตอร์เน็ต สุวัฒน์ ปุณณชัยยะ)

◆ เน็ตเวิร์กอินเตอร์เฟซเลเยอร์ (Network Interface Layer)

เนื่องจากในด้านกายภาพของเครือข่ายนั้น มีหลายวิธีการและหลายรูปแบบในการเชื่อมต่อระบบให้เป็นเครือข่าย แต่อย่างไรก็ตามในเครือข่ายอินเตอร์เน็ตนี้ ข้อมูลหรือไอพี ดาต้าแกรม(IP datagram) จะถูกถ่ายทอดและส่งผ่านไปยังปลายทางโดยไม่คำนึงถึงรูปแบบการเชื่อมต่อทางกายภาพ ไม่ว่าจะเป็นการใช้เครือข่ายใยแก้วนำแสงหรือเครือข่ายสาย Unshielded Twist Pair (UTP) เชื่อมต่อเป็นแบบเครือข่ายอีเธอร์เน็ต (Ethernet) ธรรมดาหรือเครือข่ายโทคเก้นริง (Token Ring), เอทีเอ็ม (ATM), ไอเอสดีเอ็น (ISDN) ฯลฯ ก็ตาม

การทำงานระดับล่างสุดต่อจากอินเตอร์เน็ตเวิร์กเลเยอร์ (Internetwork layer) จะเป็นการแปลงข้อมูลไอพีดาต้าแกรม (IP datagram) ให้อยู่ในรูปที่เหมาะสม และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่ายต่อไป ซึ่งในชั้นเน็ตเวิร์กอินเตอร์เฟซเลเยอร์ (Network Interface layer) นี้ เมื่อเทียบกับมาตรฐานโอเอสไอโมเดล แล้วจะเป็นการรวม 2 layer เข้าด้วยกันคือดาต้าลิงค์เลเยอร์ (Data Link layer) และฟิสิคอลเลเยอร์ (Physical layer) กล่าวโดยสรุปคือ การทำงานในชั้นต่าง ๆ ตามโครงสร้างของพิธีการทีซีพี/ไอพี จะมีลักษณะดังรูปที่ 2.9



รูปที่ 2.8 โครงสร้างของพิธีการ ทีซีพี/ไอพี ในแต่ละชั้นหรือเลเยอร์ (layer) จะมี**พิธีการหลักทำหน้าที่ต่าง ๆ** และส่งผ่านข้อมูลไปยังเครือข่ายและออกสู่อินเตอร์เน็ต

(ที่มา : เปิดโลกของ TCP / IP และโปรโตคอลของอินเตอร์เน็ต สุวัฒน์ **ปุณณชัยยะ)**

และในตารางต่อไปนี้ จะแสดงถึงหมายเลขพอร์ตที่ถูกใช้งานโดยทีซีพี

ตารางที่2.1 สรุปหมายเลขบางส่วนของพอร์ตที่ใช้งานโดยทีซีพีและยูดีพี

(**ที่มา** : เปิดโลกของ TCP / IP และโปรโตคอลของอินเตอร์เน็ต สุวัฒน์ ปุณณชัยยะ)

พิธีการที่ใช้งาน	หมายเลขพอร์ต	แบบพิธีการ	รายละเอียด
DNS	53	UDP/TCP	Domain Name System
FTP	21	TCP	FTP ด้าน control
FTP	20	TCP	FTP ด้าน ข้อมูล
HTTP	80	TCP/UDP	HTTP ด้าน server
SMTP	25	TCP	SMTP ด้าน server
Telnet	23	TCP	Teletype Network Protocol

กล่าวโดยสรุปก็คือ พิธีการ TPC/IP ทำงานโดยแบ่งเป็นชั้นเทียบกับ โอเอสไอโมเดล ได้กลไกในการทำงานของพิธีการ ทีซีพี/ไอพี มี 4 ชั้น ซึ่งในขั้นแรก คือ โปรเซสเลเยอร์ ทำหน้าที่ติดต่อกับแอพพลิเคชันและพิธีการที่แอพพลิเคชันนั้น ๆ ใช้งาน และส่งต่อมาให้ชั้น โฮสทูโฮสเลเยอร์ เพื่อติดต่อกันระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องผู้ขอใช้บริการ ในชั้นนี้จะมีการสร้าง เซสชัน (session) หรือการเชื่อมต่อระหว่างระบบขึ้นตามแต่ละพิธีการต้องการ ต่อมาเป็นการผนึกข้อมูลไปเป็นไอพี ดาต้าแกรม (IP datagram) ที่ชั้นอินเตอร์เน็ตเวิร์กเลเยอร์ (Internetwork layer) โดยอาศัยพิธีการ ไอพี เพื่อให้สามารถติดต่อส่งข้อมูลข้ามเครือข่ายไปยังเครือข่ายและเครื่องที่ถูกต้องได้ และสุดท้ายการส่งข้อมูลออกสู่โลกภายนอกต้องอาศัยกลไกชั้นเน็ตเวิร์กอินเตอร์เฟซเลเยอร์ (Network Interface layer) เพื่อแปลงข้อมูลใหม่ เพิ่มข้อมูลที่จำเป็นในการอ้างอิงตำแหน่งและแปลงข้อมูลเป็นสัญญาณไฟฟ้าส่งออกไปยังเครือข่าย และอาจจะออกไปยังเกทเวย์ (Gateway) หรือเราเตอร์ (Router) เพื่อข้ามเครือข่ายออกไปยังเส้นทางที่กำหนดไว้ในอินเตอร์เน็ตต่อไป

เราจะเห็นว่าในแต่ละชั้นของโครงสร้าง ทีซีพี/ไอพี สแตค มีการใช้งานพิธีการต่าง ๆ อยู่หนึ่งพิธีการหรือมากกว่า ในแต่ละพิธีการเหล่านี้ก็จะรับผิดชอบทำหน้าที่ของตน เพื่อส่งผ่านข้อมูลลงไปยังระดับล่าง และออกสู่เครือข่ายอินเตอร์เน็ตในที่สุด

2.3.4 กลไกของพิธีการไอพี

ในการส่งผ่านข้อมูลหรือไอพี ดาต้าแกรม (IP datagram) ไปยังเครือข่ายอินเตอร์เน็ตนั้น พิธีการ ไอพี ทำหน้าที่พิจารณาว่าปลายทางในการส่งไอพี ดาต้าแกรม (IP datagram) นั้นจะเป็นภายในเครือข่ายของตนเองหรือจะต้องส่งข้อมูลข้ามเครือข่ายไปอีก โดยการพิจารณานี้ พิธีการไอพี จะตรวจสอบจากค่าไอพีแอดเดรส ปลายทางว่าส่วนที่เป็นค่าหมายเลขเครือข่าย (network address) จะเหมือนกับค่าหมายเลขเครือข่ายของ ไอพีแอดเดรส ต้นทางหรือไม่ ถ้าค่าตรงกันแสดงว่าการส่งข้อมูลอยู่ภายในเครือข่ายเดียวกัน แต่ถ้าค่าต่างกัน แสดงว่าต้องส่งข้อมูลไปยังปลายทางที่อยู่คนละเครือข่ายกัน

การส่งข้อมูลภายในเครือข่ายเดียวกัน มีกลไกดังนี้

1. พิธีการไอพี จะเรียกใช้บริการพิธีการเออาร์พี (ARP :Address Resolution Protocol) เพื่อแปลงหมายเลข ไอพี ปลายทางให้เป็นค่าหมายเลขฮาร์ดแวร์ เช่น แมคแอดเดรส (MAC address)
2. เมื่อพิธีการไอพี ได้รับค่าหมายเลขฮาร์ดแวร์แล้ว ก็จะส่งข้อมูลนั้นไปยังฮาร์ดแวร์ที่ระบุไว้

การส่งข้อมูลข้ามเครือข่าย มีกลไกดังนี้

2.1 พิธีการไอพี ตรวจสอบพบว่าหมายเลขไอพีแอดเดรส ปลายทางอยู่คนละเครือข่ายกัน โดยพิธีการไอพี จะอ่านค่า ไอพีแอดเดรส ของเราเตอร์ (Router) เพื่อเตรียมส่งข้อมูลไปที่ เราเตอร์ (Router) แทน

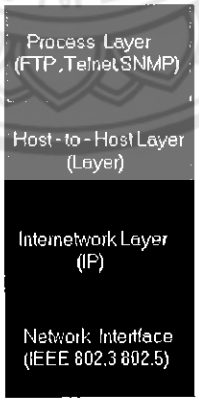
2.2 พิธีการไอพี จะเรียกใช้บริการพิธีการเออาร์พี (ARP) เพื่อแปลงค่าไอพีแอดเดรส ของเราเตอร์ (Router) ให้เป็นค่าหมายเลขฮาร์ดแวร์

2.3 พิธีการไอพี ส่งข้อมูลไอพี ดาต้าแกรม (IP datagram)ไปยังเราเตอร์ (Router) ที่กำหนดไว้ จากนั้นเราเตอร์ (Router)ส่งข้อมูลข้ามเครือข่ายไปตามขั้นตอนต่อไป

3. โอเอสไอ กับ ทีซีพี/ไอพี

เนื่องจากโอเอสไอ (OSI) เกิดขึ้นมาหลังจากที่ทีซีพี/ไอพี (TCP/IP หรือ Transmission Control Protocol / Internet Protocol ได้มีการใช้งานกันอย่างแพร่หลายไปแล้ว โดย ทีซีพี/ไอพี ใช้ในเครือข่ายอาร์พาเน็ต (ARPANET) เป็นเครือข่ายแรก ซึ่งต่อมาได้ขยายการเชื่อมต่อไปทั่วโลกเป็นเครือข่ายอินเตอร์เน็ต ทำให้มาตรฐาน ทีซีพี/ไอพี เป็นที่ยอมรับกันอย่างกว้างขวางและการที่ ทีซีพี/ไอพี เป็นพิธีการชนิดที่ใช้ได้ฟรีไม่ต้องจ่ายค่าลิขสิทธิ์ การใช้งาน ทีซีพี/ไอพี ก็ยิ่งมีจำนวนผู้ใช้เพิ่มมากขึ้นไปอีกจนถือเป็นมาตรฐานที่มีผู้ใช้รับส่งข้อมูลมากที่สุดในปัจจุบัน

เมื่อ ทีซีพี/ไอพี เป็นมาตรฐานที่เกิดขึ้นก่อนโอเอสไอ โดยทีซีพี/ไอพี จะมีการแบ่งจำนวนขั้นตอนที่ใช้รับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์สองระบบออกเป็น 4 ชั้นเท่านั้น หรือเรียกว่าเป็น ทีซีพี/ไอพี สแตก โดยมีชื่อเรียกแตกต่างกันดังนี้ (ดูรูปที่ 2.9ประกอบ)

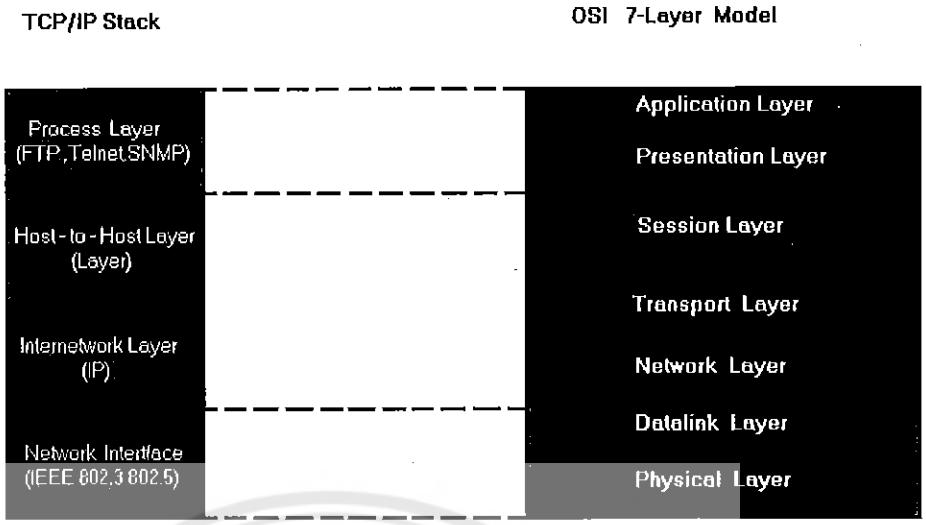


รูปที่ 2.9 โครงสร้างของโปรโตคอล TCP /IP

(ที่มา : เปิดโลกของ TCP / IP และโปรโตคอลของอินเตอร์เน็ต สุวัฒน์ ปุณณชัยยะ)

4.โครงสร้างพิธีการ ทีซีพี/ไอพี

- ชั้นบนเรียกว่าโปรเซสเลเยอร์ จะเป็นแอพพลิเคชันโปรโตคอล (Application Protocol) ที่ทำหน้าที่เชื่อมต่อกับผู้ใช้และให้บริการต่าง ๆ เช่น FTP,Telnet,SNMP ฯลฯ
- ชั้นถัดมาเรียกว่าโฮสทูโฮสเลเยอร์ จะเป็นทีซีพีหรือยูดีพี ที่ทำหน้าที่คล้ายกับชั้นที่ 4 ของ โอเอสไอโมเดล คือควบคุมการรับส่งข้อมูลจากปลายด้านส่งถึงปลายด้านรับข้อมูล และตัดข้อมูลออกเป็นส่วนย่อยให้เหมาะกับเครือข่ายที่ใช้รับส่งข้อมูล รวมทั้งประกอบข้อมูลส่วนย่อย ๆ นี้เข้าด้วยกันเมื่อถึงปลายทาง
- ชั้นถัดลงมาคือ อินเตอร์เน็ตเวิร์กเลเยอร์ (Internetwork Layer) ได้แก่ส่วนของพิธีการไอพี ซึ่งทำหน้าที่คล้ายกับชั้นที่ 3 ของโอเอสไอโมเดล(OSI 7-Layer Model) คือเชื่อมต่อคอมพิวเตอร์เข้ากับระบบเครือข่ายที่อยู่ชั้นล่างลงไป และทำหน้าที่เลือกเส้นทางการรับส่งข้อมูลผ่านอุปกรณ์เครือข่ายต่าง ๆ จนไปถึงผู้รับข้อมูล ในชั้นนี้จะจัดการกับกลุ่มข้อมูลในลักษณะที่เรียกว่าเฟรม (Frame) ในรูปแบบของ ทีซีพี/ไอพี ที่เรารู้จักกันนั่นเอง
- ส่วนชั้นสุดท้ายที่อยู่ล่างสุด เรียกว่าเน็ตเวิร์กอินเตอร์เฟซ (Network Interface) คือชั้นที่ควบคุมฮาร์ดแวร์การรับส่งข้อมูลผ่านเครือข่าย ซึ่งเทียบได้กับชั้นที่ 1 และ 2 ของโอเอสไอโมเดล (OSI 7-Layer Model) ในชั้นนี้จะทำหน้าที่เชื่อมต่อกับฮาร์ดแวร์ และควบคุมการรับส่งข้อมูลในระดับฮาร์ดแวร์ของเครือข่าย ซึ่งที่ใช้กันอยู่จะเป็นตามมาตรฐานของไอทริปเปิลอี (IEEE) เช่น IEEE 802.3 จะเป็นการเชื่อมต่อผ่านแลน (LAN) แบบอีเธอร์เน็ต (Ethernet LAN) หรือ IEEE 802.5 จะเป็นการเชื่อมต่อผ่านแลน (LAN) แบบโทคเก้นริง (Token Ring) เป็นต้น



รูปที่ 2.10 TCP / IP โปรโตคอลเมื่อเทียบกับ OSI 7-Layer Model
(ที่มา : เปิดโลกของ TCP / IP และโปรโตคอลของอินเตอร์เน็ต สุวัฒน์ ปุณณชัยยะ)

2.3.5 พิธีการทีซีพี/ไอพี เมื่อเทียบกับโอเอสไอโมเดล

ถึงแม้ว่าทีซีพี/ไอพี จะไม่ได้มีการแบ่งชั้นของการสื่อสารข้อมูลตรงตามโอเอสไอโมเดล และไม่ได้เป็นมาตรฐานเดียวกัน แต่โอเอสไอ ก็ออกแบบมาให้เปิดกว้างและเข้ากันได้ดีกับทีซีพี/ไอพี โดยทีซีพี จะเทียบได้กับประมาณชั้นที่ 4 ของโอเอสไอ และไอพี จะเทียบได้กับประมาณชั้นที่ 3 ของโอเอสไอ แม้ว่าจะไม่ลงตัวตรงกันพอดีนัก แต่ก็สามารถเชื่อมต่อทำงานด้วยกันได้ ทำให้มาตรฐานของโอเอสไอ สามารถนำ ทีซีพี/ไอพี มาใช้งานร่วมกันได้เป็นอย่างดี เมื่อเรากลับไปมองมาตรฐานของโอเอสไอโมเดล (OSI 7-Layer Model) ที่เปิดกว้างให้เราเลือกใช้มาตรฐานต่าง ๆ ของแต่ละชั้นมาใช้งานร่วมกันแล้ว จะพบว่าการกำหนดมาตรฐานของโอเอสไอ ได้บรรลุวัตถุประสงค์เป็นอันมาก คือเราสามารถเลือกใช้อุปกรณ์ฮาร์ดแวร์เครือข่าย และโปรแกรมควบคุมในชั้นที่ 1 และ 2 จากบริษัทใดก็ได้มาเชื่อมต่อเข้าด้วยกัน แล้วนำ ทีซีพี/ไอพี ซึ่งมีการใช้งานกันอย่างแพร่หลายมาใช้ในชั้นที่ 4 และ 3 ตามลำดับ ส่วนชั้นที่ 5 ถึงชั้นที่ 7 จะเป็นแอพพลิเคชันที่ต้องการ

2.4 ไอพีแอดเดรส (IP Address)

แนวความคิดหลักของระบบเครือข่ายคอมพิวเตอร์ก็คือ การเชื่อมโยงอุปกรณ์เข้าด้วยกัน ไม่ว่าจะเป็นเครื่องเซิร์ฟเวอร์ที่ให้บริการ (หรือบางที่เรียกว่า host) และอุปกรณ์ในเครือข่ายอื่น ๆ เช่น เราเตอร์ (Router), เครื่องพิมพ์ เพื่อให้สามารถแชร์การใช้อุปกรณ์ร่วมกันได้ หรือสามารถส่งผ่านข้อมูลไปมาระหว่างกันได้ถูกต้อง เมื่อมีการเชื่อมต่อกันแล้วก็จำเป็นต้องมีการกำหนดหรือระบุเลขหมายของอุปกรณ์ทุกชิ้นทุกชนิดในเครือข่าย เพื่อให้อ้างอิงได้โดยไม่ซ้ำกัน เพราะถ้าซ้ำกันแล้วการรับส่งข้อมูลอาจจะไม่ถึงมือผู้รับปลายทางได้อย่างถูกต้อง เลขหมายดังกล่าวจะเรียกว่า แอดเดรส (address)หรือเลขหมายประจำตัวที่มีข้อกำหนดเป็นมาตรฐาน ซึ่งในการใช้งานพิธีการทีซีพี/ไอพี ที่เชื่อมโยงเครือข่ายอินเตอร์เน็ตนี้ เลขหมายที่ใช้อ้างอิงถึงกันจะใช้เป็นตัวเลขที่เรียกว่ าไอพีแอดเดรส (IP Address :Internet Protocol address) หรือ "หมายเลข ไอพี"

หมายเลข ไอพีแอดเดรส ถูกกำหนดขึ้นมาให้เป็นหมายเลขอ้างอิงประจำตัวของอุปกรณ์ต่าง ๆ ที่เชื่อมต่ออยู่ในเครือข่ายอินเตอร์เน็ต โดยการกำหนดหมายเลขแอดเดรส(address) ให้แต่ละเครื่องหรือแต่ละอุปกรณ์นี้จะต้องไม่ซ้ำกัน ซึ่งหมายเลข ไอพีแอดเดรส นี้จะไม่ถูกผูกติดกับตัวฮาร์ดแวร์แต่อย่างใด จึงสามารถกำหนดใหม่หรือแก้ไขเปลี่ยนแปลงได้เมื่อมีการเปลี่ยนตัวฮาร์ดแวร์ ทั้งนี้เนื่องจากเป็นการกำหนดด้วยซอฟต์แวร์ แตกต่างกับหมายเลขแมคแอดเดรส (MAC address :Media Access Control address) ซึ่งเป็นหมายเลขประจำตัวของอุปกรณ์ที่ต่ออยู่ในเครือข่าย ค่าแมคแอดเดรส (MAC address) จะถูกกำหนดจากบริษัทผู้ผลิตอุปกรณ์ตั้งแต่เริ่มผลิต เช่น อุปกรณ์เน็ตเวิร์อินเตอร์เฟซการ์ด (Network Interface Card) (NIC) จะมีค่าแมคแอดเดรส (MAC address) ประจำตัวที่ไม่ซ้ำกันและไม่สามารถแก้ไขได้ ค่าแมคแอดเดรส (MAC address) เป็นการระบุค่าอ้างอิงของอุปกรณ์ฮาร์ดแวร์ในระดับล่างสุด (Physical Layer) ของกลไกการรับส่งข้อมูลในเครือข่าย ถ้าจะใช้หมายเลขแมคแอดเดรส (MAC address) สำหรับระบุอ้างอิงกันในเครือข่ายแล้วจะเกิดปัญหามาก เมื่อมีการเปลี่ยนหรือย้ายเครื่องต้องทำการกำหนดระบบเครือข่ายใหม่ (configuration) นอกจากนี้ยังจดจำได้ยากกว่า ตัวอย่างของหมายเลขแมคแอดเดรส (MAC address) คือ 08:0a:0e:12:b5:05 การที่ ไอพีแอดเดรส ถูกใช้อ้างอิงในการติดต่อกันด้วยพิธีการ ทีซีพี/ไอพี เพราะการใช้หมายเลขไอพีแอดเดรส จะยืดหยุ่นและคล่องตัวกว่า

การทำงานของพิธีการไอพี จำเป็นต้องอาศัยหมายเลข ไอพีแอดเดรส นี้เพื่อระบุและอ้างถึงอุปกรณ์ต่าง ๆ ที่ต่ออยู่ในเครือข่ายไม่ว่าจะเป็นเว็บเซิร์ฟเวอร์, เมล์เซิร์ฟเวอร์,อุปกรณ์เราเตอร์ (Router) ฯลฯ หมายเลข ไอพีแอดเดรส จะเป็นค่าตัวเลขขนาด 32 บิต ถูกแบ่งออกเป็นส่วนละ 8 บิต รวมเป็น 4 ส่วนและคั่นแต่ละส่วนด้วยเครื่องหมายจุด ดังนั้นค่าตัวเลขในแต่ละส่วนจะมีได้ตั้งแต่ 0 ถึง 255 ตัวอย่างของ ไอพีแอดเดรส เช่น 205.144.78.1 หรือ 10.0.0.1 เป็นต้น นอกจากนี้

ไอพีแอดเดรส บางหมายเลขหรือบางช่วงจะมีการใช้งานในลักษณะความหมายและหน้าที่พิเศษออกไปในการทำงานของพิธีการ ทีซีพี/ไอพี เช่น หมายเลข ไอพี ที่ 127.0.0.0 เป็นหมายเลข ไอพี ที่ใช้ทำหน้าที่เป็นลูปแบคแอดเดรส (loop back address) คือใช้กำหนดค่าลูปแบค (loop back) หรือแอดเดรสย้อนกลับให้กับอุปกรณ์นั้น

ลูปแบคแอดเดรส (Loop back address) เป็นแอดเดรสที่กำหนดขึ้นเพื่อใช้ในงานที่ต้องการให้โปรเซส (process) หนึ่งติดต่อกับโปรเซส (process) อื่น ๆ ในเครื่องเดียวกันผ่าน ไอพีแอดเดรส หรืออาจจะกล่าวโดยง่ายว่าโปรเซส(Process) ในเครื่องจะติดต่อกันระหว่างโปรเซสเองได้โดยกลไกลูปแบคอินเตอร์เฟซ (Loop back interface) ซึ่งมีการกำหนดหมายเลข ไอพีแอดเดรส พิเศษให้เป็นลูปแบคแอดเดรส (Loop back address) ดังนี้

127.0.0.0	ลูปแบค (Loop back) สำหรับเครือข่ายคลาสเอ (class A)
191.255.0.0	ลูบแบค (Loop back) สำหรับเครือข่ายคลาสบี (class B)
223.255.255.0	ลูปแบค (Loop back) สำหรับเครือข่ายคลาสซี (class C)

ดังนั้นถ้าเราพบว่า ไอพีแอดเดรส เป็น 127.0.0.2 หรือ 191.255.0.1 หรือ 223.255.255.111 ให้คิดว่าเป็นการกำหนดค่าลูปแบค (Loop back) ในเครือข่ายคลาส เอ,บี ,ซี (class A,B,C) ตามลำดับ

ค่า ไอพีแอดเดรส ที่กำหนดเป็นลูปแบค (Loop Back) นี้จะถูกผนึกหรือ bind (กำหนดการอ้างอิงกับฮาร์ดแวร์) เข้ากับลูปแบคเน็ตเวิร์กอินเตอร์เฟซ (Loop back network interface) ของอุปกรณ์นั้น ๆ เช่น ค่า ไอพีแอดเดรส ที่ 127.0.0.1สำหรับในระบบวินโดว์เอ็นที (Windows NT) จะถือเป็นค่าที่ใช้เป็นลูปแบค (loop back) ปกติที่ระบบรู้จักและกำหนดไว้ให้ใช้งาน ดังนั้นเมื่อมีการส่งผ่านข้อมูลไปที่หมายเลขไอพี 127.0.0.1 จะไม่มีการส่งข้อมูลออกไปที่เครือข่าย แต่จะย้อนกลับมาที่ ไอพี ต้นทางโดยกลไกการลูปแบค(loop back) ของ ลูปแบคไดรเวอร์ (loop back driver) นั่นเอง

ค่าของ ไอพีแอดเดรส จะถูกกำหนดออกเป็น 2 ความหมาย คือ ค่าของหมายเลขอุปกรณ์ในเครือข่าย (host address) และค่าของหมายเลขเครือข่าย (network address) ตัวอย่างเช่น มีเครื่องเว็บเซิร์ฟเวอร์เชื่อมต่ออยู่ในเครือข่าย 2 เครื่อง โดยแต่ละเครื่องมี ไอพีแอดเดรส ประจำตัวคือ 205.144.78.2 และ 205.144.78.3 ตามลำดับ เครื่องทั้งสองมีค่าของเลขหมายเครือข่ายเหมือนกัน คือ 205.144.78แสดงว่าเครื่องทั้งสองต่อเชื่อมอยู่ในเครือข่ายเดียวกัน บนสายสัญญาณที่เชื่อมโยงเส้นเดียวกันแต่มีหมายเลขประจำตัวเครื่องที่แตกต่างกันคือ 2 และ 3 ตามลำดับ

เพื่อไม่ให้การกำหนดแจกจ่ายค่า ไอพีแอดเดรส ซ้ำซ้อนกัน จึงมีหน่วยงานกลางทำหน้าที่กำหนด ไอพีแอดเดรส และแจกจ่ายเลขหมายให้แต่ละองค์กรได้ใช้งาน คือ หน่วยงาน InterNIC หรือ Internet Network Information Center เป็นผู้ดูแลฐานข้อมูลการแจกจ่าย ไอพีแอดเดรส และ

การใช้งานให้เกิดประโยชน์สูงสุด บริษัทหรือองค์กรที่ต้องการ ไอพีแอดเดรส ในการกำหนดใช้งานเพื่อเชื่อมระบบเครือข่ายของตนเข้าสู่เครือข่ายอินเตอร์เน็ต จะต้องสมัครเป็นสมาชิกองค์กร InterNIC เสียก่อน โดยจะมีการเก็บค่าบำรุงเป็นรายปีและแบ่งระดับของสมาชิกเป็น องค์กรขนาดใหญ่ องค์กรขนาดกลางและองค์กรขนาดเล็ก ตามลำดับ ค่าบำรุงสมาชิกก็จะจ่ายไม่เท่ากัน เช่น ค่าบำรุงสมาชิกขององค์กรขนาดกลางอยู่ที่ปีละ 5,000 $ หรือประมาณเกือบสองแสนบาทต่อปี เมื่อบริษัทหรือองค์กรได้เป็นสมาชิกแล้ว ก็สามารถแจ้งความจำนงขอ ไอพีแอดเดรส มาใช้งานได้ต่อไป สำหรับในประเทศไทย บริษัทใดที่ต้องการเชื่อมต่อเครือข่ายของตนเข้าสู่อินเตอร์เน็ต จะต้องติดต่อกับบริษัทผู้ให้บริการอินเตอร์เน็ต (Internet Service Provider หรือ ISP) แทน และสามารถขอ ไอพีแอดเดรส ที่ ISP ได้ทันที เพราะทาง ISP ได้ขอ ไอพีแอดเดรส เอาไว้สำหรับแจกจ่ายให้กับลูกค้าของตนแล้ว

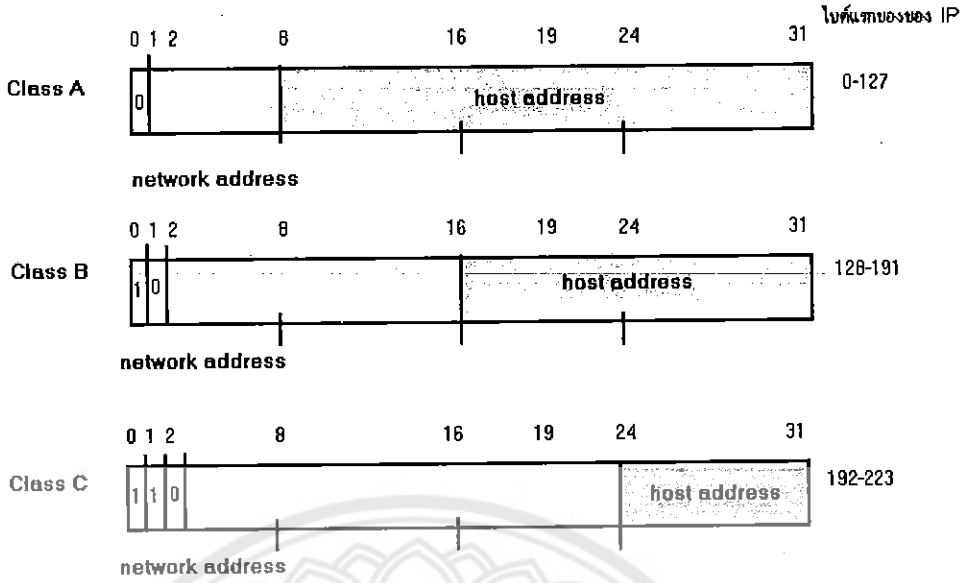
2.5 การจัดลำดับชั้นของเครือข่าย (Network Class)

การกำหนดค่า ไอพีแอดเดรส ไม่สามารถกำหนดขึ้นได้ตามใจชอบ แต่มีระเบียบวิธีแบ่งและการกำหนดที่ชัดเจนเป็นมาตรฐาน ในหมายเลข ไอพีแอดเดรส จะถูกแบ่งเป็น 4 ส่วนคั่นด้วยเครื่องหมายจุด ซึ่งสามารถแยกเป็น 2 ส่วนย่อยคือ ส่วนแรกเป็นหมายเลขของเครือข่าย (network address) และส่วนที่สองเป็นหมายเลขของเครื่องลูกข่าย เรียกว่าเน็ตเวิร์กคลาส (network class) ซึ่งมีการกำหนดให้มีเน็ตเวิร์กคลาส (network class) นี้ก็เพื่อให้สามารถแจกจ่าย ไอพีแอดเดรส ให้กับเครือข่ายต่าง ๆ ได้อย่างเหมาะสม เพราะในแต่ละเครือข่ายมักจะแตกต่างกัน บางเครือข่ายก็มีจำนวนเครื่องลูกข่ายมาก บางเครือข่ายมีน้อยแต่มีเครือข่ายย่อย ๆ ในเครือข่ายหลักจำนวนมาก ฉะนั้นถ้าไม่มีการจัดลำดับของเครือข่ายให้ดี หมายเลข ไอพีแอดเดรส ก็จะถูกใช้งานอย่างสิ้นเปลืองและใช้งานไม่ได้เต็มจำนวนที่มี ลำดับชั้นของเครือข่ายแบ่งได้เป็น 5 ลำดับคือ class A,B,C,D และ E

ในแต่ละเน็ตเวิร์กคลาส (network class) หมายเลข ไอพี ทั้ง 32 บิตจะถูกกำหนดเป็นหมายเลขของเครือข่ายและหมายเลขของเครื่องลูกข่าย โดยมีเงื่อนไขดังนี้คือ

- คลาสเอ (class A) ไอพีแอดเดรส บิตแรกของไบต์แรกสุดจะเป็น 0 เสมอ
- คลาสบี (class B) ไอพีแอดเดรส 2 บิตแรกของไบต์แรกสุดจะเป็น 1 และ 0 เสมอ
- คลาสซี (class C) ไอพีแอดเดรส 3 บิตแรกของไบต์แรกสุดจะเป็น 1,1 และ 0 เสมอ

การแบ่งหมายเลขเครือข่ายและหมายเลขเครื่องลูกข่ายจะเป็นดังรูปที่ 2.11



รูปที่ 2.11 การแบ่งส่วนของหมายเลขเครือข่ายและหมายเลขเครื่องลูกข่าย ในแต่ละลำดับชั้นของเครือข่ายใน class A , B และ C

(ที่มา : เปิดโลกของ TCP / IP และโปรโตคอลของอินเตอร์เน็ต สุวัฒน์ ปุณณชัยยะ)

การแบ่งส่วนของหมายเลขเครือข่ายและหมายเลขเครื่องลูกข่าย ในแต่ละลำดับชั้นของเครือข่ายใน class A,B และ C

ดังนั้นแต่ละ network class จะมีเครื่องลูกข่าย (host address) ที่เป็นสมาชิกในเครือข่ายของตนได้ตามตาราง

ตารางที่2.2 แสดงลักษณะและจำนวนเครื่องลูกข่าย

(ที่มา : เปิดโลกของ TCP / IP และโปรโตคอลของอินเตอร์เน็ต สุวัฒน์ ปุณณชัยยะ)

Class	จำนวนเครื่องลูกข่ายที่มีได้	ไอพีแอดเดรส เริ่มต้น
A	$2^{24} = 16,777,214$	0-127
B	$2^{16} = 65,536$	128-191
C	$2^{8} = 256$	192-223
D	-	224-239
E	-	240-255

Class A เป็นหมายเลข ไอพีแอดเดรส ที่เริ่มตั้งแต่ 0-127 จะกำหนดให้กับเครือข่ายที่มีขนาดใหญ่ เพราะ 1 เครือข่ายสามารถมีเครื่องลูกข่ายได้กว่า 16 ล้านเครื่อง หมายเลข ไอพีแอดเดรส จะเป็นลักษณะ net.host.host.host ตัวอย่างเช่น ค่า ไอพีแอดเดรส ของ class A เป็น 121.7.23.3 หมายถึงเครือข่ายที่ 121 หมายเลขเครื่องคือ 7.23.3

- Class B เป็น ไอพีแอดเดรส ที่เริ่มตั้งแต่ 128-191 จะกำหนดให้กับเครือข่ายที่มีขนาดใหญ่เช่นกันแต่เล็กกว่า class A ในแต่ละเครือข่ายของ class B สามารถมีเครือลูกข่ายได้ 2^{16} – จำนวนแอดเดรสที่ใช้ควบคุมระบบเครือข่าย = 64,516 เครื่อง หมายเลข ไอพีแอดเดรส จะเป็นลักษณะ net.net.host.host ตัวอย่างเช่น ค่า ไอพีแอดเดรส ของ class B เป็น 137.103.210.2 หมายถึง เครือข่ายเลขที่ 137.103 หมายเลขเครื่องคือ 210.2
- Class C เป็น ไอพีแอดเดรส ที่เริ่มตั้งแต่ 192-223 จะกำหนดให้กับเครือข่ายที่เป็นองค์กรทั่วไป ซึ่งส่วนใหญ่จะเป็นองค์กรขนาดกลางถึงเล็ก ในแต่ละเครือข่ายมีเครื่องลูกข่ายไม่เกิน 2^{8} – จำนวนแอดเดรสที่ใช้ควบคุมเครือข่าย = 254 เครื่อง หมายเลข ไอพีแอดเดรส ของ class C เป็น net.net.net.host ตัวอย่างเช่น ค่า ไอพีแอดเดรส ของ class C เป็น 202.182.255.3 หมายถึง เครือข่ายเลขที่ 202.192.255 หมายเลขเครื่องคือ 3
- Class D เป็นการกำหนดหมายเลข ไอพีแอดเดรส สำรองไว้สำหรับส่งข้อมูลแบบ multicast ซึ่งไม่มีการแจกจ่ายให้ใช้งานทั่วไป
- Class E เป็นหมายเลข ไอพีแอดเดรส พิเศษที่ใช้สำหรับงานทดสอบและพัฒนา ไม่มีการกำหนดให้ใช้งานทั่วไป

เราอาจจำหมายเลข ไอพีแอดเดรส ได้ง่าย ๆ คือ ในเครือข่าย class A จะมีหมายเลข ไอพี ตัวแรกอยู่ระหว่าง 0-127 ดังนั้นค่าที่ตามมาจะเป็นหมายเลขของเครื่อง ส่วนเครือข่าย class B หมายเลข ไอพี หน้าสุดจะเป็น 128-191 ดังนั้นค่า ไอพี สองตัวแรกจะเป็นหมายเลขเครือข่ายและค่า ไอพี สองตัวหลังจะเป็นหมายเลขเครื่องสุดท้าย เครือข่ายใน class C จะมีหมายเลข ไอพี หน้าสุดเป็น 192-223 ดังนั้นค่า ไอพี ตัวหลังสุดจะเป็นหมายเลขเครื่อง

2.6 ส่วนประกอบของ File Transfer Protocol (FTP)

FTP มีส่วนประกอบพื้นฐาน 4 ส่วน คือ

1. Control Connection
2. Data Connection
3. FTP Command
4. FTP Reply

2.6.1 Control Connection

เครื่อง เอฟทีพี ไคลเอนต์ จะทำการติดต่อแบบ TCP กับเครื่อง เอฟทีพี เซิร์ฟเวอร์ ทางพอร์ต 21 การติดต่อขั้นต้นระหว่างเครื่อง เซิร์ฟเวอร์ กับเครื่อง ไคลเอนต์ แบบนี้เรียกว่า Control Connection เครื่อง ไคลเอนต์ใช้การติดต่อนี้เพื่อส่งคำสั่งแบบ FTP หรือ FTP Command ไปให้กับเครื่อง เซิร์ฟเวอร์ และเพื่อรับการตอบรับหรือ รีพลาย จาก เซิร์ฟเวอร์ ทั้งนี้ รีพลาย หนึ่งๆอาจแตกออกเป็นหลายบรรทัด แต่มันก็จะอยู่ในรูปแบบ ASCII

2.6.2 Data Connection

มี คอมมานด์ เพียง 3 ตัวเท่านั้น ที่จะทำให้เกิด การติดต่อระหว่างเครื่อง ไคลเอนต์ กับเครื่อง เซิร์ฟเวอร์ ผ่านทาง พอร์ต 20 หรือ Data port นั่นคือ STOR (ส่งไฟล์), RETR (รับไฟล์), LIST (รับไฟล์และแสดงออกในรูปไดเรคทอรี) โดยทั่วไป เครื่องเอฟทีพีไคลเอนต์ ใช้คำสั่ง PORT เพื่อส่ง Local Socket Name (เช่น IP Address และ port number) ไปยังเครื่อง เอฟทีพี เซิร์ฟเวอร์ และเครื่อง เซิร์ฟเวอร์ จะสามารถติดต่อกลับผ่านทาง Address และ พอร์ต ที่ได้รับข้อมูลมา เครื่อง ไคลเอนต์ จะยอมรับ Data Connection จากทาง เซิร์ฟเวอร์ และ การส่งผ่านข้อมูลจะเริ่มทำไปเรื่อยๆจนกว่ากระบวนการจะเสร็จสิ้น

2.6.3 FTP Command

- ABOR ยกเลิกคำสั่งที่ค้างอยู่
- CWD[directory] เปลี่ยนไดเรคทอรีปัจจุบัน
- DELE[file name] ลบไฟล์ที่กำหนดไว้
- LIST[[path][file set] รับรายชื่อไฟล์ และ ไดเรคทอรี
- PASS[password] ส่ง password ของ user
- PORT[socket name] ไคลเอนต์ บอก เซิร์ฟเวอร์ ว่า address ไหนที่จะใช้ต่อเพื่อส่ง

	ข้อมูล
♦ PWD	print working directory แสดงไดเรคทอรีที่ทำงานอยู่ในปัจจุบัน
♦ QUIT	Logout จาก เอฟทีพี เซิร์ฟเวอร์
♦ RETR[file name]	ดึงไฟล์ที่ต้องการ
♦ STOR[file name]	ส่งไฟล์ที่ระบุไว้ให้กับเซิร์ฟเวอร์
♦ TYPE[data type]	เปลี่ยน data type
♦ USER[user name]	ส่ง user name เพื่อทำการ Login

นี่เป็นคำสั่งที่ใช้บ่อยๆ

คำสั่ง FTP สามารถแบ่งเป็น 3 กลุ่มดังนี้

2.6.3.1 ACCESS CONTROL COMMAND ได้แก่

- ♦ USER = USER NAME ใช้ในการแสดงตัวของ user เพื่อให้ เซิร์ฟเวอร์ ใช้ในการ access สู่ ระบบข้อมูล
- ♦ PASS = PASSWORD ใช้ในการยืนยันตัว user
- ♦ ACCT = ACCOUNT ใช้ในการแสดงตัว user account
- ♦ CWD = CHANGE WORKING DIRECTORY ใช้เปลี่ยนไดเรคทอรีที่ทำงานหรือเปลี่ยนชุดของข้อมูลสำหรับใช้เก็บไฟล์
- ♦ CDUP = CHANGE TO PARENT DIRECTORY คล้ายคำสั่ง CWD แต่ได้รวมตัวโปรแกรมสำหรับรับส่ง directory tree ระหว่างระบบปฏิบัติการไว้ในคำสั่งด้วย
- ♦ SMNT = STRUCTURE MOUNT ให้ user สร้างโครงสร้างข้อมูลของระบบไฟล์ที่แตกต่างโดยไม่ต้องเปลี่ยนการ login หรือไม่ต้องใช้ accounting information
- ♦ REIN = REINITIALIZE ยกเลิก user ยกเว้นในกรณีที่จะให้ต้องการส่งข้อมูลใดๆที่ทำอยู่ให้เสร็จก่อน
- ♦ QUIT = LOGOUT ยกเลิก user ไม่สนว่าการส่งข้อมูลที่มีอยู่จะเสร็จหรือไม่

2.6.3.2 TRANSFER PARAMETER COMMANDS ได้แก่

- PORT = DATA PORT กำหนด พอร์ตให้ โฮสต์ ในการทำ data connection
- PASV = PASSIVE ให้ server DTP ทำการ listen บน data port (ซึ่ง พอร์ตนั้น ไม่ใช่ port default) และรอการเชื่อมต่อ
- TYPE = REPRESENTATION TYPE เจาะจงชนิดของการนำเสนอข้อมูลให้เป็นตามที่ได้บรรยายไว้ในส่วนขอมูลที่นำเสนอและจัดเก็บโดย

  A = ASCII , I = IMAGE

  E = EBCDIC , L = ตามขนาดของข้อมูล

- STRU = FILE STRUCTURE กำหนดชนิดของโครงสร้างที่จะนำเสนอและจัดเก็บโดย

  F = ไฟล์

  R = เรคคอร์ด

  P = หน้า

  ปกติจะเป็นแบบไฟล์

- MODE = TRANSFER MODE กำหนดรูปแบบการส่งไฟล์

  S = เป็นสาย (stream)

  B = บล็อก (block)

  C = บีบอัด (compressed)

  ปกติจะเป็นแบบเป็นสาย

2.6.3.3 FTP SERVICE COMMANDS ได้แก่

- RETR = RETRIEVE คำสั่งนี้ทำให้ เซิร์ฟเวอร์ ถ่ายโอน copy ของไฟล์ ระบุลงใน pathname ให้กับ เซิร์ฟเวอร์ หรือ user DTPในตอนท้ายของ data connection
- STOR = STORE คำสั่งนี้ทำให้ server DTP ยอมรับการถ่ายโอนข้อมูลโดยผ่านทาง data connection และเก็บข้อมูลเป็นไฟล์ไว้ใน server site (ถ้าไฟล์ที่ตรง pathname เดียวกันได้คงอยู่ใน server site นั้นๆแล้ว) อาจจะเกิดการทับของข้อมูลที่ถูกส่งมาใหม่
- STOU = STORE UNIQUE คำสั่งนี้คล้าย STOR แต่ต่างกันที่ไฟล์นี้จะถูกสร้างจากไดเรคทอรีที่มีชื่อเดียวกันกับชื่อไฟล์นี้

- APPE = APPEND (with create) คำสั่งนี้ทำให้ server DTP ยอมรับการถ่ายโอนข้อมูลผ่านทาง data connection และเก็บข้อมูลเป็นไฟล์ไว้ที่ server site ข้อมูลที่ได้มาอาจจะต่อจากข้อมูลเดิมในไฟล์
- ALLO = ALLOCATE คำสั่งนี้มีไว้สำหรับการจองหน่วยความจำไว้สำหรับไฟล์ที่กำลังจะถูกถ่ายโอนเข้ามาใหม่ ข้อความจะเป็นเลขจำนวนเต็มฐานสิบเพื่อบอกจำนวนไบต์ที่ต้องเตรียมไว้
- REST = RESTART เป็นข้อความที่ให้ server marker ทำการ restart ขณะการถ่ายโอนไฟล์ใดๆ คำสั่งนี้ไม่ได้ทำให้เกิดการถ่ายโอนไฟล์แต่เป็นการข้ามไฟล์นั้นๆให้ไปที่จุด check point ของข้อมูลนั้นก่อน
- RNFR = RENAME FROM คำสั่งนี้จะระบุ pathname เก่าที่เราจะทำการเปลี่ยนชื่อและจะต้องตามด้วยคำสั่ง rename to ทันที
- RNTO = RENAME TO คำสั่งนี้จะระบุ pathname ใหม่สำหรับไฟล์ที่เราต้องการโดยจะต้องนำหน้าด้วยคำสั่ง rename from ก่อน สองคำสั่งนี้จะต้องใช้คู่กันเสมอเมื่อมีการ rename
- ABOR = ABORT ให้ เซิร์ฟเวอร์ ยกเลิกคำสั่งก่อนหน้านี้ หรือยกเลิกการถ่ายโอนไฟล์ในขณะนั้น เซิร์ฟเวอร์จะมี 2 สถานะคือ คำสั่งได้ถูกกระทำเรียบร้อยแล้ว และคำสั่งอยู่ในระหว่างการทำงาน
- DELE = DELETE ทำให้ไฟล์ใน pathname ที่ระบุถูกลบจาก server site
- RMD = REMOVE DIRECTORY คำสั่งนี้จะลบทั้งไดเรคทอรีที่ระบุไว้ หรือเป็น ไดเรคทอรีลูก (subdirectory) ที่ทำงานอยู่ในปัจจุบัน
- MKD = MAKE DIRECTORY คำสั่งนี้จะสร้างไดเรคทอรีที่ระบุไว้ หรือสร้างไดเรคทอรีลูก (subdirectory) ที่ทำงานอยู่ในปัจจุบัน
- PWD = PRINT WORKING DIRECTORY คำสั่งนี้ให้บอกชื่อไดเรคทอรีที่ทำงานอยู่ในปัจจุบันจากทาง รีพลาย
- LIST = LIST คำสั่งนี้ให้แสดงชื่อไดเรคทอรีจาก เซิร์ฟเวอร์ มาที่ passive DTP ถ้า pathname ระบุไดเรคทอรี หรือกลุ่มอื่นของไฟล์ เซิร์ฟเวอร์ จะส่งชื่อของไฟล์ในไดเรคทอรีที่ต้องการมาให้ ถ้า pathname ระบุไฟล์นั้นๆแล้ว เซิร์ฟเวอร์ จะส่งรายละเอียดเกี่ยวกับไฟล์นั้นมาให้

- NLST = NAME LIST คำสั่งนี้จะแสดงชื่อไดเรคทอรีจาก เซิร์ฟเวอร์ มาที่ server site pathname จะระบุไดเรคทอรี เซิร์ฟเวอร์ จะส่งข้อความเป็นสาย (stream)ซึ่งเป็นชื่อของไฟล์อย่างเดียว
- SITE = SITE PARAMETERS คำสั่งจะถูกใช้โดย เซิร์ฟเวอร์ ในการเตรียมบริการในการถ่ายโอนไฟล์ที่จำเป็นแต่ไม่มีคำสั่งมาตรฐานที่รองรับได้ภายในโปรโตคอล
- SYST = SYSTEM คำสั่งนี้ให้ค้นหาชนิดของระบบปฏิบัติการของ เซิร์ฟเวอร์ คำตอบจะอยู่ในบรรทัดแรกของการรีพลาย
- STAT = STATUS คำสั่งนี้ทำให้เป็นลักษณะการตอบกลับของ control connection ในรูปของรีพลาย คำสั่งอาจจะส่งระหว่างการถ่ายโอนไฟล์
- HELP = HELP คำสั่งนี้จะให้ เซิร์ฟเวอร์ช่วยส่งคำอธิบาย ในสถานะของคำสั่งที่ต้องการรายละเอียด ผ่าน control connection ให้กับผู้ใช้
- NOOP = NOOP คำสั่งนี้ไม่มีผลใดๆ เพียงแต่ให้ เซิร์ฟเวอร์ ส่ง OK reply กลับมาเท่านั้น

2.6.4 FTP REPLIES

เอฟทีพี เซิร์ฟเวอร์ จะมีการตอบสนองต่อผู้ใช้ เรียกว่า FTP replies เพื่อให้ผู้ใช้แน่ใจว่าเซิร์ฟเวอร์ ได้รับคำสั่งแล้ว และจะรู้การทำงานของ เซิร์ฟเวอร์ ว่ามีปัญหาหรือไม่ แต่ละคำสั่งจะมีการรีพลายอย่างน้อย 1 ครั้ง

FTP reply จะประกอบด้วย ตัวเลข 3 ตัว คั่นด้วย space bar 1 ครั้งและตามด้วย ข้อความอธิบาย และจะปิดท้ายด้วย end-of-line code (ในระบบ TELNET) เช่น

120 Service ready in nnn minutes

แต่ละตำแหน่งของตัวเลขจะมีความสำคัญ คือ หลักแรก หมายความถึงสถานะของคำสั่งในตอนนี้ว่า success(S) , wait for reply(W) หรือ fail(F) ซึ่งแต่ละตัวมีความหมายดังนี้คือ

1st digit	Meaning
1 yz	Positive Preliminary Reply คือ คำสั่งนี้ สามารถทำได้ แต่ยังไม่เสร็จในขณะนี้ (เพราะเพิ่งเริ่มต้น)
2 yz	Positive Completion Reply คือ คำสั่งนี้ได้ทำเสร็จสมบูรณ์ และกำลังรอ คำสั่งใหม่
3 yz	Positive Intermediate Reply คือคำสั่งนี้สามารถทำได้แต่ต้องขอข้อมูลเพิ่มเติมจากผู้ใช้จึงจะทำต่อได้
4 yz	Transient Negative Completion Reply คือ คำสั่งนี้ไม่สามารถทำได้เพราะความผิดพลาดในระบบชั่วขณะหนึ่ง
5 yz	Permanent Negative Reply คือ คำสั่งนี้ไม่สามารถทำได้เพราะความผิดพลาดของข้อมูลต้องทำการแก้ไขก่อนที่จะลองใหม่อีกครั้ง

ตัวเลขหลักถัดมาจะบอกถึงสถานะของตัวเลขแรก เป็นการขยายความว่าผิดพลาดหรือสำเร็จเพราะอะไร หรือต้องการข้อมูลส่วนใดเพิ่ม ตัวเลขดังกล่าวมีดังนี้

2nd	Meaning
x 0 z	Syntax คือ syntax error หรือ illegal command
x 1 z	Information คือ ต้องการข้อมูลเพิ่ม
x 2 z	Connections คือ ต้องการให้ตรวจสอบสถานะของ data connection หรือ control connection
x 3 z	Authentication and accounting คือ ต้องการข้อมูลสำหรับการ login เช่น password หรือ account ของผู้ใช้
x 4 z	ไม่ได้ใช้
x 5 z	File System ให้ดูสถานะของ เซิร์ฟเวอร์ ว่ายัง connect อยู่รึเปล่า

ตัวเลขหลักสุดท้าย เป็นการระบุรายละเอียดลงไปให้ละเอียดมากขึ้น เช่น

331 ตรวจสอบ password

332 ตรวจสอบการ login

ขอยกตัวอย่างคร่าวๆของการติดต่อระหว่าง ไคลเอนต์และ เซิร์ฟเวอร์

ในตอนแรกเมื่อผู้ใช้ (client) ต้องการติดต่อ FTP เพื่อขอการถ่ายโอนไฟล์จากผู้ให้บริการ (server) จะมีการ connect 2 ครั้ง คือ ครั้งแรกเพื่อขอเข้าระบบ และครั้งที่สองเพื่อขอใช้งานภายในระบบ ครั้งแรกจะใช้คำสั่งนี้

ftp < file name> เช่น

ftp arthur.cs.purdue.edu

connect to arthur.cs.purdue.edu

จากนั้น เซิร์ฟเวอร์ จะ รีพลาย กลับมาว่า ติดต่อได้แล้ว ขอชื่อ และpassword เพื่อทำการ login อย่างถูกต้อง และสามารถเข้าไปใช้ประโยชน์ใน เซิร์ฟเวอร์ ได้

220 arthur.cs.purdue.edu FTP server (DYNIX v3.0.12) ready

Name(arthur:usera): anonymous

331 Guest login OK , send ident as password.

Password:guest

230 Guest login OK , access restriction apply

เมื่อ ไคลเอนต์ สามารถ access เข้าไปได้แล้ว ต่อไปจะเป็นการขอถ่ายโอนข้อมูลที่ชื่อ tcpbook.tar เข้าเครื่อง ไคลเอนต์ และเปลี่ยนชื่อเป็น bookfile

ftp> get pub / comer / tcpbook.tar bookfile

200 PORT command okay.

150 Opening data connection for /bin/ls(128.10.2.1,2363)(7897088 bytes)

226 Transfer complete.

8272793 bytes received in 98.04 seconds (82 kbytes / s)

เมื่อมีการถ่ายโอนเรียบร้อยแล้ว ก็ปิดการทำงาน คือ

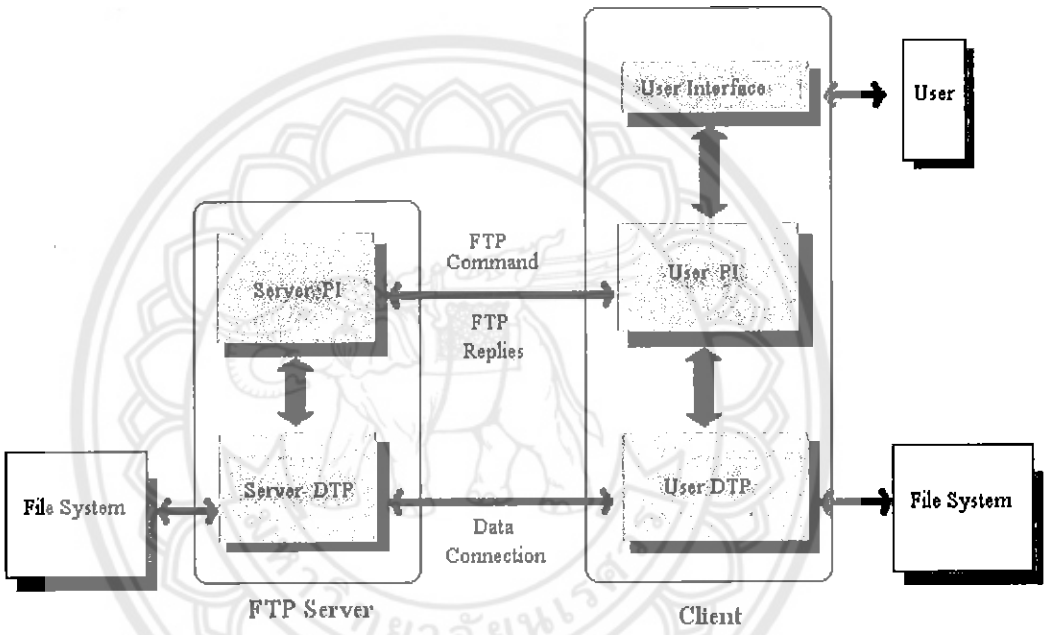
ftp> close

221 Goodbye

ftp> quit

เป็นการออกจาก access อย่างสมบูรณ์

2.6.5 FTP Model



รูปที่2.12 โมเดลการทำงานของเอฟทีพีเซิร์ฟเวอร์

เริ่มจาก User ใช้เครื่อง ไคลเอนต์โดยผ่าน User interface ของเครื่องไคลเอนต์ User interface จะเป็นตัวสื่อสาร กับ เซิร์ฟเวอร์ โดยผ่านทาง User PI (protocol interpreter) โดย User PI จะทำหน้าที่แปลงคำสั่ง FTP ของ User เป็นคำสั่ง FTP มาตรฐาน แล้ว User PI จะสร้าง Control connection ขึ้น (พอร์ต 21) เพื่อส่งคำสั่ง FTP มาตรฐานสู่ Server PI ว่าต้องการอะไร จะขอสร้าง Data connection ที่พอร์ตไหน (ปรกติจะเป็น พอร์ต 20 ถ้าไม่มีการระบุลงไป)แล้ว Server PI จะตอบ รีพลาย กลับมาเป็นเลข 3 หลัก เพื่อบอกกับ User ว่าจะตกลงจะทำตามคำสั่งหรือไม่ สถานะตอนนี้เป็นอย่างไร

จากนั้น Server PI ก็จะส่งคำสั่งนั้นไปที่ Server DTP (Server Data Transfer Process) Server DTP ก็จะจัดการกับไฟล์ตามคำสั่ง FTP ที่รับมา ถ้าเป็นการส่งข้อมูลให้ User Server จะรี

พลายตอบ เซิร์ฟเวอร์ ผ่านทาง Control connection และ Server PI จะบอกให้ User DTP ทำการ "Listen" ที่ พอร์ต ที่ระบุไว้ ให้เป็น Data connection จากนั้น Server DTP ก็จะทำการเปิด พอร์ต เพื่อสร้าง Data connect จากนั้นก็อ่านข้อมูลจากระบบไฟล์ แล้วส่งผ่าน Data connection โดย พอร์ตนั้นเป็นพอร์ตที่ User ระบุไว้แต่แรก (ถ้าไม่ระบุก็จะใช้ port default คือ พอร์ต 20) User DTP ก็จะจัดเก็บข้อมูลเข้าระบบข้อมูลของตัวเอง

เมื่อ Server DTP จัดการส่งข้อมูลตามที่ Server PI สั่งมาเสร็จแล้ว ก็จะปิด Data connection แล้ว Server DTP ก็จะรายงานไปที่ Server PI จากนั้น Server PI ก็จะรีพลายไปที่ User PI ว่าส่งข้อมูลเสร็จแล้ว สถานะเป็นอย่างไร แต่ Control connection จะไม่ถูกปิด จนกว่าจะมีคำสั่ง Quit มาจาก User PI

2.6.6 สรุปการติดต่อระหว่าง ไคลเอนต์ และ เซิร์ฟเวอร์

ในตอนแรกเครื่อง ไคลเอนต์ จะส่งคำสั่งเข้าไปเพื่อขอทำการติดต่อ เมื่อเครื่องเซิร์ฟเวอร์ ได้รับก็จะทำการถามชื่อ และpassword เพื่อความปลอดภัยของระบบ จากนั้นเมื่อเข้าระบบได้ แล้วก็จะสามารถใช้คำสั่งต่างๆ (FTP Commands) ได้ เช่น จะขอให้มีการส่งผ่านข้อมูลแบบไฟล์ ไปยังเครื่องไคลเอนต์ เครื่องไคลเอนต์ ก็จะส่ง FTP Command (ในที่นี้ใช้คำสั่ง get) ไปยังเครื่อง เซิร์ฟเวอร์ เครื่อง เซิร์ฟเวอร์ ก็จะส่ง รีพลาย กลับมา (ในขั้นตอนที่กล่าวมาทั้งหมดนี้ จะมีการส่ง ผ่านทาง พอร์ต 21 ที่เป็น control port เท่านั้น) หลังจากนั้นก็จะมีการเปิด data port เพื่อส่งผ่าน ข้อมูล เมื่อส่งเรียบร้อยแล้ว data port ก็จะปิด และจะมีการส่ง รีพลาย จาก เซิร์ฟเวอร์ ไปยัง ไคลเอนต์ว่า การส่งผ่านเรียบร้อยแล้ว และก็สามารถเริ่มการทำงานอื่นๆต่อได้ หรือจะเลิกการ ทำงาน (โดยใช้คำสั่ง close และ quit) ก็ได้

2.7 วินโดวส์ซ็อกเก็ต (Windows Sockets)

2.7.1 ประวัติของวินโดวส์ซ็อกเก็ต

ในระบบปฏิบัติการยูนิกซ์(Unix) มีความสามารถอย่างหนึ่งที่เรียกว่า ซ็อกเก็ต ซึ่งก็คือการที่โปรแกรมต่าง ๆ สามารถสื่อสารข้อมูลระหว่างกันได้ โดยไม่จำเป็นว่าจะต้องทำงานอยู่บนเครื่องเดียวกัน ซึ่งได้กลายเป็นมาตรฐานของการสื่อสารข้อมูลทางเครือข่ายคอมพิวเตอร์ไปโดยปริยาย

เมื่อระบบปฏิบัติการ Windows ได้รับความนิยมสูงขึ้นในเวลาต่อมา จึงได้มีการพัฒนาความสามารถในแบบซ็อกเก็ต ซึ่งเรียกว่า วินโดวส์ซ็อกเก็ต หรือ วินซ็อก โดยการพัฒนาวินซ็อกเกิดจากความร่วมมือระหว่างหลายบริษัทด้วยกัน และ วินซ็อกได้ถูกทำให้เป็นมาตรฐานเปิด (Open Standard) ซึ่งทางทีมงานได้ปล่อยวินซ็อก 1.0 ออกมาในเดือน มิถุนายน ค.ศ. 1992 และ อีก 6 เดือนถัดมา คือ มกราคม ค.ศ. 1993 ก็ได้ปล่อย วินซ็อก 1.1 ออกมา และได้กลายเป็นมาตรฐานที่ใช้ใน วินโดวส์ 95

ในมุมมองของนักพัฒนาโปรแกรมบนระบบเครือข่าย วินซ็อกคือ ชุดคำสั่งมาตรฐานให้เรียกใช้ ซึ่งเรียกชุดคำสั่งมาตรฐานนี้ว่า เอพีไอ (API : Application Program Interface) ดังนั้น วินซ็อก จึงมีอีกชื่อเรียกว่า วินซ็อกเอพีไอ (Winsock API)

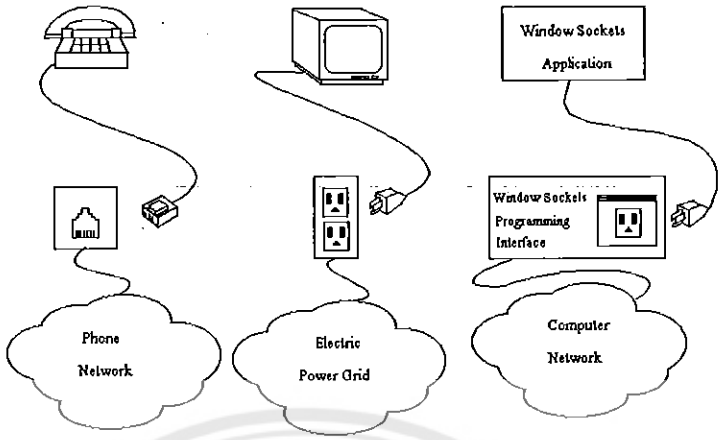
ในระบบปฏิบัติการ วินโดวส์ 95/98 จะมี ไลบรารีไฟล์สำหรับเรียกใช้ วินซ็อกเอพีไอ อยู่แล้ว โดยไฟล์นี้มีชื่อว่า wsock32.dll ซึ่งจะถูกเก็บอยู่ในโฟลเดอร์ \windows\system

2.7.2 หลักการของวินซ็อก

◆ วินซ็อกคืออะไร

วินซ็อก คือ ระบบเปิดที่เชื่อมต่อการเขียนโปรแกรมระบบเครือข่ายภายใต้ระบบปฏิบัติการวินโดวส์ คำว่าระบบเปิดหมายถึง เป็นระบบที่ผู้พัฒนาโปรแกรมสามารถจะนำวินซ็อคมาใช้ และดัดแปลงโดยที่ไม่ต้องจ่ายเงินค่าลิขสิทธิ์

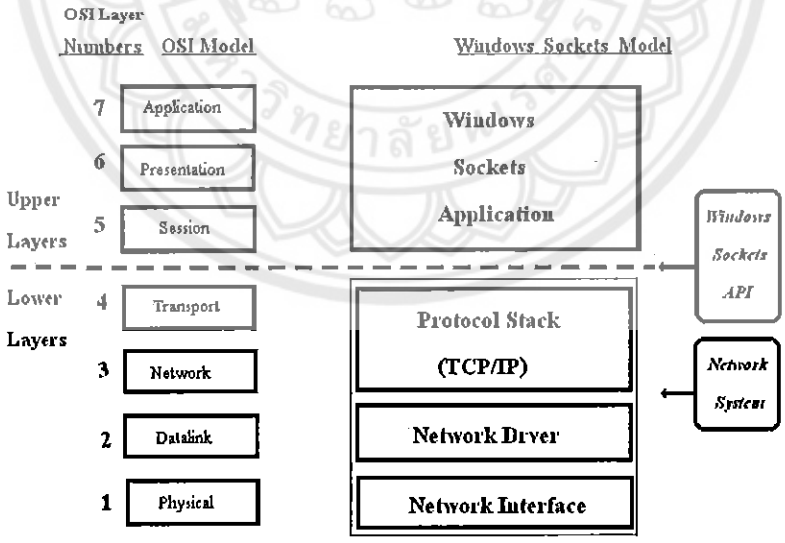
วินซ็อก หรือ วินโดวส์ซ็อกเก็ตเอพีไอ (Windows Socket API:WSA) รวบรวมฟังชั่น และโครงสร้างข้อมูล ที่จำเป็นต่อการพัฒนาโปรแกรมในระบบเครือข่ายให้ผู้พัฒนาได้นำไปใช้ภายใต้มาตรฐานเดียวกัน ถ้าจะให้เปรียบเทียบแล้ว วินซ็อก เปรียบเสมือน ปลั๊กเสียบสำหรับการเขียนโปรแกรม (programming plug) ที่จะเชื่อมต่อระหว่างโปรแกรม กับระบบเครือข่ายเข้าด้วยกัน เช่นเดียวกันกับ ปลั๊กไฟ ที่เชื่อม ไฟฟ้า กับ เครื่องใช้ไฟฟ้าเข้าด้วยกัน หรือ เปรียบเหมือนกับ แจ็คโทรศัพท์ ที่เชื่อมต่อระหว่างเครื่องโทรศัพท์เข้ากับสายสัญญาณโทรศัพท์



รูปที่ 2.13 แสดงหลักการของวินซ็อก

(ที่มา : Windows Socket Network Programming ,Bob Quinn and Dave Shute)

- เปรียบเทียบโอเอสไอโมเดล(OSI Model) กับวินซ็อก



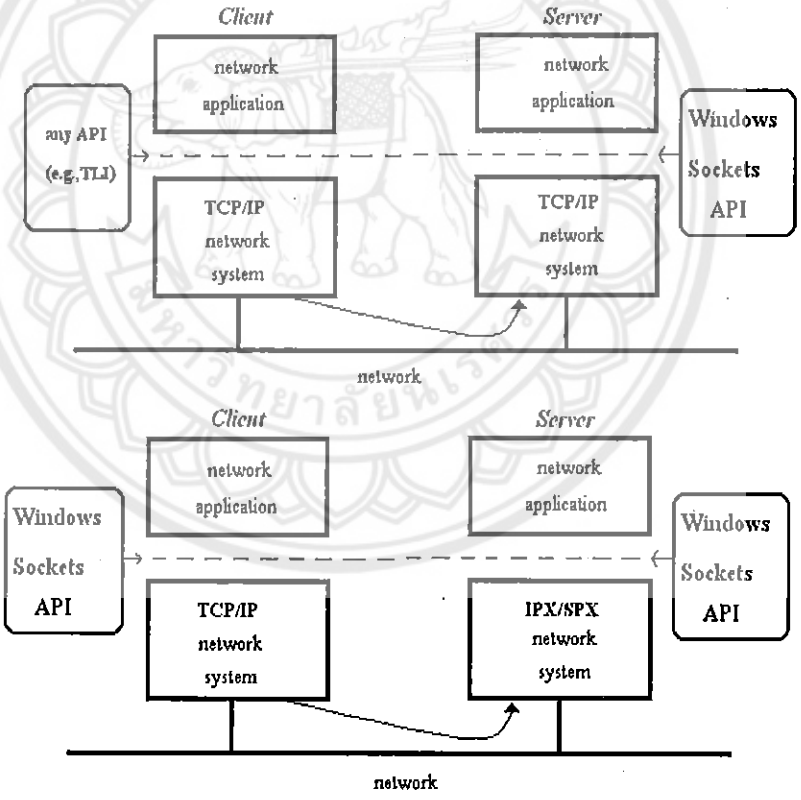
รูปที่ 2.14 เปรียบเทียบระหว่างโอเอสไอโมเดลกับวินซ็อก

(ที่มา : Windows Socket Network Programming ,Bob Quinn and Dave Shute)

จากรูป 2.14 เราแบ่ง โอเอสไอโมเดล ออกเป็น 2 ส่วนใหญ่ ๆ คือ ชั้นบน(Upper Layers) กับ ชั้นล่าง (Lower Layers) โดย วินซ็อกเอพีไอ คือ ตัวที่เชื่อมอยู่ระหว่างชั้นบนกับชั้นล่าง ในมุมมองของ วินซ็อก ชั้นบนคือโปรแกรมที่เรียกใช้งานวินซ็อก ส่วนชั้นล่างคือระบบเครือข่าย

◆ โปรโตคอลและเอพีไอ(Protocols and APIs)

วินซ็อคเอพีไอเป็นอิสระต่อโปรโตคอล วินซ็อคเอพีไอ สามารถติดต่อกับ โปรโตคอลได้หลากหลายชนิด แต่การที่โปรแกรมจะสื่อสารกันได้นั้นจะต้องอยู่บนโปรโตคอลเดียวกัน ดังนั้นหากอยู่บนโปรโตคอลเดียวกันแล้ว ถึงแม้โปรแกรมจะพัฒนามาจากเอพีไอ คนละชุดกันก็สามารถที่จะสื่อสารกันได้ ถึงแม้จะไม่ 100 เปอร์เซ็นต์ก็ตาม แต่ถ้าโปรแกรมนั้นอยู่คนละโปรโตคอลกัน ก็จะไม่มีทางสื่อสารกันได้เลย ถึงแม้จะพัฒนาโปรแกรมมาจากเอพีไอเดียวกันก็ตาม ดังรูปที่ 2.15 ข้างล่างนี้



รูปที่ 2.15 การติดต่อของวินซ็อกเอพีไอกับโปรโตคอลต่างๆ

(ที่มา : Windows Socket Network Programming ,Bob Quinn and Dave Shute)

2.7.3 ทีซีพี และ ยูดีพี (TCP and UDP)

รูปแบบการสื่อสารข้อมูล ของโปรโตคอลทีซีพี/ไอพี(TCP/IP) มี 2 แบบ คือ ทีซีพี (Transmission Control Protocol) และ ยูดีพี (User Datagram Protocol)

ทีซีพี คือการสื่อสารข้อมูลแบบที่ต้องมีการเชื่อมต่อก่อน (Connection Oriented) คือ จะมีการเชื่อมต่อกันในครั้งแรก แล้วระหว่างการรับหรือส่งข้อมูล จะต้องมีการส่งสัญญาณโต้ตอบกันตลอด เช่น ถ้า A ส่งข้อมูลไปหา B ทาง A ก็จะรอจนกว่า B จะตอบกลับมาว่าได้รับข้อมูลอย่างถูกต้อง แล้ว A จึงจะส่งข้อมูลชุดต่อไป

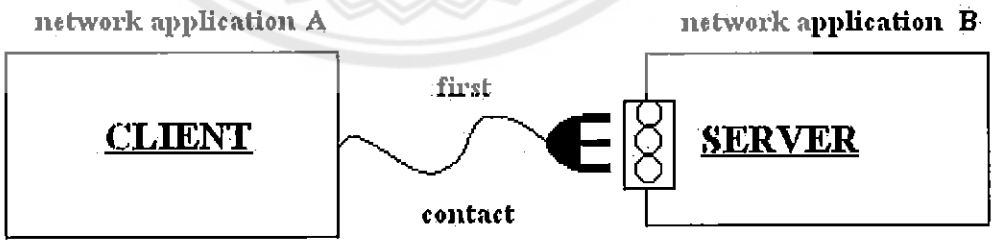
ยูดีพี (User Datagram Protocol) คือ การสื่อสารข้อมูลแบบที่ไม่ต้องมีการเชื่อมต่อ มีลักษณะเหมือนกับการประกาศ ไม่ต้องสนใจว่าผู้รับจะได้รับหรือไม่ ส่งไปเรื่อย ๆ ไม่มีการรอสัญญาณตอบกลับ

ดังนั้นในการรับส่งข้อมูลนั้น แบบ ทีซีพี จะมีความถูกต้องของข้อมูล มากกว่า แบบยูดีพี แต่จะช้าและยุ่งยากกว่า

2.7.4 กลไกการทำงานของโปรแกรมในระบบเครือข่าย

- แบบจำลอง ไคลเอนต์-เซิร์ฟเวอร์ (Client – Server Model)

ทุก ๆ โปรแกรมในระบบเครือข่าย จะต้องมีปลายทางของการสื่อสาร ซึ่งก็คือ ไคลเอนต์และเซิร์ฟเวอร์ โดยในการสื่อสารไคลเอนต์จะส่งแพ็กเก็ตแรก และ เซิร์ฟเวอร์จะรับไปเพื่อสร้างการเชื่อมต่อ



รูปที่ 2.16 แบบจำลอง ไคลเอนต์ – เซิร์ฟเวอร์

(ที่มา : Windows Socket Network Programming ,Bob Quinn and Dave Shute)

ในการเชื่อมต่อระหว่างไคลเอนต์กับเซิร์ฟเวอร์นั้น ไคลเอนต์จะต้องรู้ตำแหน่งและรู้จักซ็อกเก็ตของเซิร์ฟเวอร์ และเซิร์ฟเวอร์จะต้องตั้งชื่อซ็อกเก็ตของตัวเองเพื่อให้ไคลเอนต์สามารถใช้อ้างอิงได้ ชื่อของซ็อกเก็ต จะสอดคล้องกับ ไอพีแอดเดรส (IP Address) และ หมายเลขพอร์ต (Port Number)

เมื่อไคลเอนต์ทำการเชื่อมต่อกับเซิร์ฟเวอร์ได้สำเร็จ ซ็อกเก็ตของทั้งสองจะถูกรวมกันอยู่ในรูปแบบของการเชื่อมต่อ ซึ่งประกอบด้วยองค์ประกอบ 5 สิ่งด้วยกันคือ

1. โปรโตคอล (ต้องเป็นโปรโตคอลเดียวกันทั้งไคลเอนต์และเซิร์ฟเวอร์)
2. ไอพีแอดเดรสของไคลเอนต์
3. หมายเลขพอร์ตของไคลเอนต์
4. ไอพีแอดเดรสของเซิร์ฟเวอร์
5. หมายเลขพอร์ตของเซิร์ฟเวอร์

◆ ขั้นตอนการทำงานของโปรแกรมในระบบเครือข่าย

โปรแกรมในระบบเครือข่ายทั้งหมดไม่ว่าจะเป็นไคลเอนต์หรือเซิร์ฟเวอร์ จะมีขั้นตอนการทำงานในการสื่อสารข้อมูล 5 ขั้นตอนดังนี้

1. เปิดซ็อกเก็ต(Open a socket)

ทั้งไคลเอนต์และเซิร์ฟเวอร์ต้องการซ็อกเก็ตในการสื่อสารข้อมูลในระบบเครือข่าย การเปิดซ็อกเก็ตทำได้โดยใช้ฟังก์ชัน socket() โดยมีรูปแบบการใช้ฟังก์ชันดังนี้

```
SOCKET socket(int af,   /* protocol suite */
              int type,         /* protocol type */
              int protocol);    /*  protocol name */
```

af : "address family"

type : ชนิดของซ็อกเก็ต

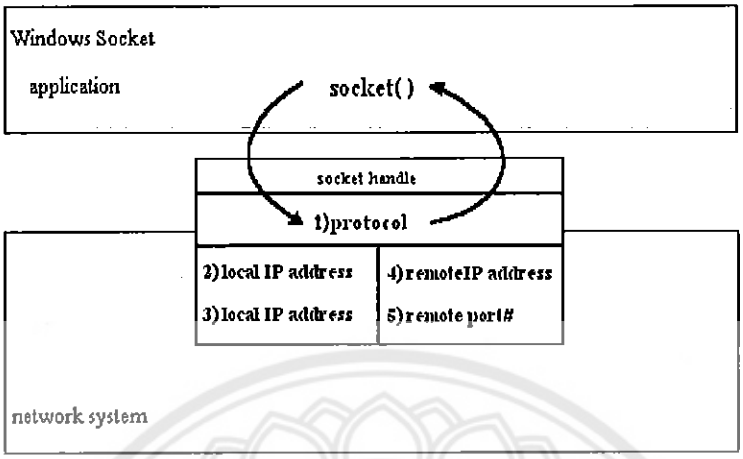
protocol : ชื่อโปรโตคอล

ตัวอย่างเช่น SOCKET socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);

AF_INET เป็นแอดเดรสแฟมิลี่

SOCK_STREAM เป็นซ็อกเก็ตชนิด TCP

IPPROTO_TCP เป็นโปรโตคอลชนิดทีซีพี/ไอพี



รูปที่ 2.17 ขั้นตอนการทำงานของโปรแกรมในระบบเครือข่าย

(ที่มา : Windows Socket Network Programming ,Bob Quinn and Dave Shute)

ฟังก์ชัน socket() จะคืนค่า socket descriptor เมื่อทำงานสำเร็จ และจะคืนค่าเป็น INVALID_SOCKET เมื่อทำงานผิดพลาด

2.ให้ชื่อซ็อกเก็ต(Name the socket)

ไคลเอนต์ต้องสามารถรู้ตำแหน่งและรู้จักซ็อกเก็ตของเซิร์ฟเวอร์ และ เซิร์ฟเวอร์ต้องให้ชื่อกับซ็อกเก็ตเพื่อให้ไคลเอนต์สามารถใช้อ้างอิงได้ ซึ่งชื่อของซ็อกเก็ตจะประกอบด้วย 3 สิ่งคือ โปรโตคอล หมายเลขพอร์ต และ แอดเดรส

ในการให้ชื่อซ็อกเก็ต เซิร์ฟเวอร์จะทำการประกาศโครงสร้างข้อมูลซ็อกเก็ตแอดเดรส (Socket address structure) และเรียกฟังก์ชัน bind() โดยโครงสร้างซ็อกเก็ตและฟังก์ชัน bind() จะทำการให้ค่าที่อยู่และคุณสมบัติต่าง ๆ กับซ็อกเก็ต

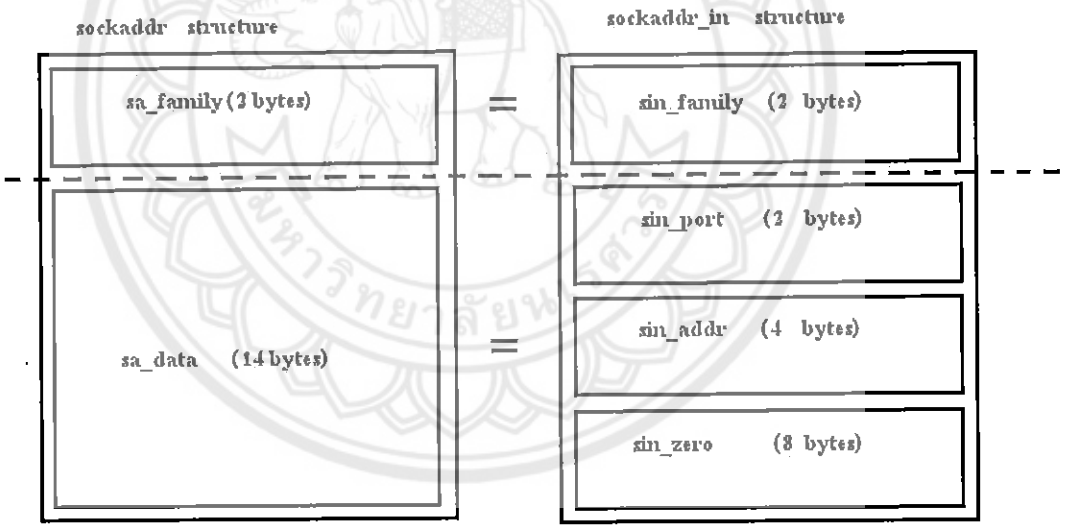
โครงสร้างข้อมูลของซ็อกเก็ต

sockaddr Structure

```
struct sockaddr{
        u_short         sa_family;      /* address family */
        char            sa_data;        /* undefine */
};
```

sockaddr structure เป็นรูปทั่วไปของโครงสร้างข้อมูลซ็อกเก็ตซึ่งในการใช้งานจริงจะไม่อยู่ในรูปแบบนี้ สำหรับการใช้งานกับโปรโตคอลทีซีพี/ไอพี เราจะใช้โครงสร้างข้อมูล sockaddr_in โดยรายละเอียดของโครงสร้างนี้คือ

```
struct sockaddr_in{
        short           sin_family;             /* address family */
        u_short         sin_port;               /*port number*/
        struct          in_addr sin_addr;       /*IP address (32 bit) */
        char            sin_zero[8];            /*<unused filler>*/
```



รูปที่ 2.18 หลักการให้ชื่อซ็อกเก็ต

(ที่มา : Windows Socket Network Programming ,Bob Quinn and Dave Shute)

ฟังก์ชัน bind()

```
int    bind(SOCKET s,              /* an unbound socket */
            struct sockaddr addr,  /* local port and IP address*/
```

int namelen); /* addr structure length */

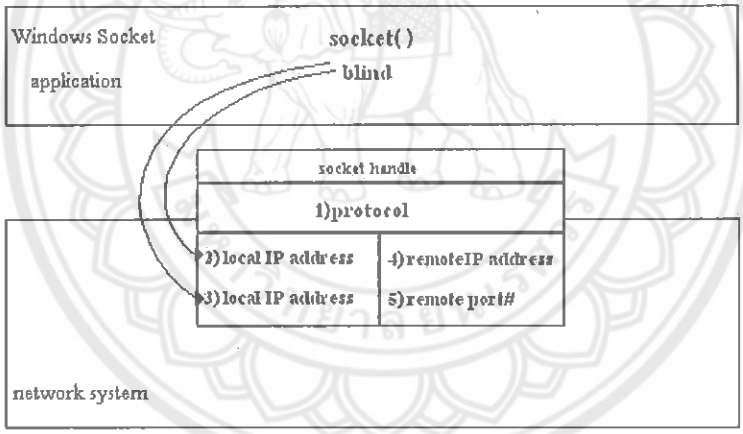
s: socket handle

addr: พ้อยเตอร์(pointer) ที่ชี้ โครงสร้างข้อมูลซ็อกเก็ตแอดเดรส

namelen: ความยาวของโครงสร้างข้อมูลซ็อกเก็ตแอดเดรสที่พ้อยเตอร์ addr เป็นตัวชี้อยู่

ตัวอย่าง

```
SOCKADDR_IN saServer;
SaServer.sin_family = AF_INET;
SaServer.sin_addr.s_addr=INADDR_ANY; //Let WinSock supply address
SaServer.sin_port=htons(nPort); //Use port from command line
int nRet;
    nRet=bind(listenSocket,(LPSOCKADDR)&saServer,sizeof(struct sockaddr));
```



รูปที่ 2.19 หลักการใช้ฟังก์ชัน bind()

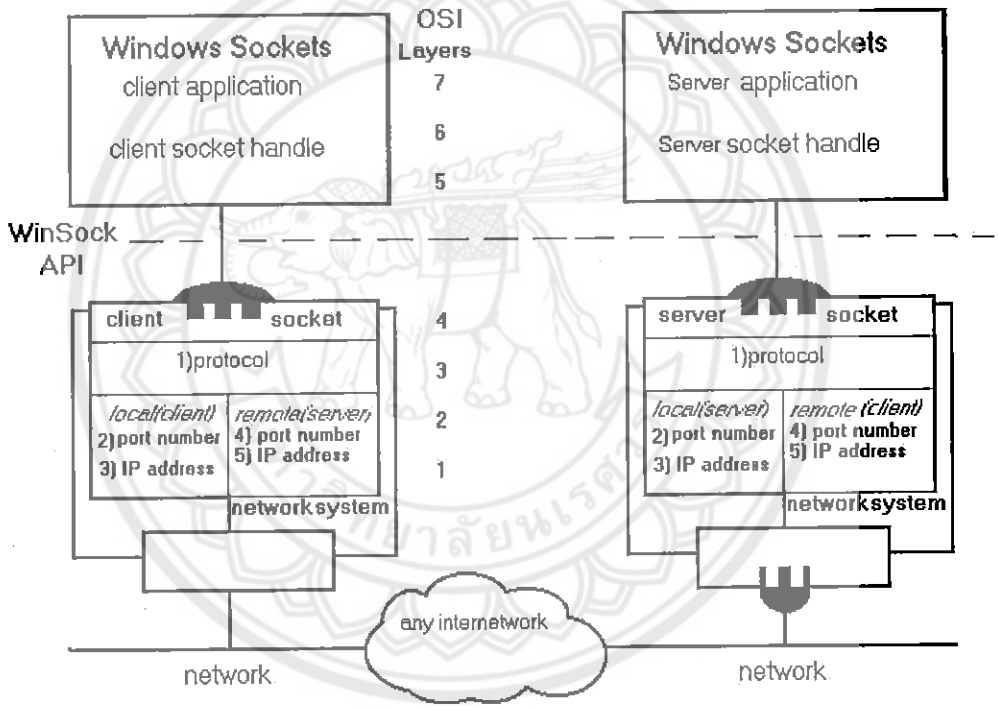
(ที่มา : Windows Socket Network Programming ,Bob Quinn and Dave Shute)

ฟังก์ชัน bind() จะคืนค่าศูนย์เมื่อทำงานสำเร็จและจะคืนค่า SOCKET_ERROR เมื่อทำงานผิดพลาด สำหรับฟังก์ชัน bind() นี้จำเป็นต้องเรียกใช้สำหรับโปรแกรมที่เป็นเซิร์ฟเวอร์ แต่สำหรับโปรแกรมที่เป็นไคลเอนต์จะเรียกใช้หรือไม่ก็ได้(ไม่จำเป็น)

3.สื่อสารกับซ็อกเก็ตอื่น (Associate with another socket)

การสื่อสารระหว่าง 2 ซ็อกเก็ต มีขั้นตอนดังนี้คือ

- เซิร์ฟเวอร์เตรียมการสำหรับการสื่อสาร
- ไคลเอนต์เริ่มการสื่อสาร
- เซิร์ฟเวอร์ตอบสนองการสื่อสาร



รูปที่ 2.20 แสดงหลักการของวินซ็อก

(ที่มา : Windows Socket Network Programming ,Bob Quinn and Dave Shute)

หลังจากการสื่อสารสำเร็จ ทั้งไคลเอนต์และเซิร์ฟเวอร์จะรู้จักซ็อกเก็ตของอีกฝ่าย

การเตรียมการสื่อสารของเซิร์ฟเวอร์

สำหรับสตรีมเซิร์ฟเวอร์ (stream server) ซึ่งทำงานในแบบ ทีซีพี คือ ต้องมีการเชื่อมต่อก่อนจึงจะส่งข้อมูลได้ ทางฝั่งเซิร์ฟเวอร์จะรอรับการเชื่อมต่อด้วยฟังก์ชัน listen()

```
int     listen(SOCKET s,                /* a named,unconnected socket */
               int backlog);            /* pending connect queue length */
```

ตัวอย่าง

```
nRet=listen(listenSocket, //Bound socket
            SOMAXCONN); //Number of connection request queue
```

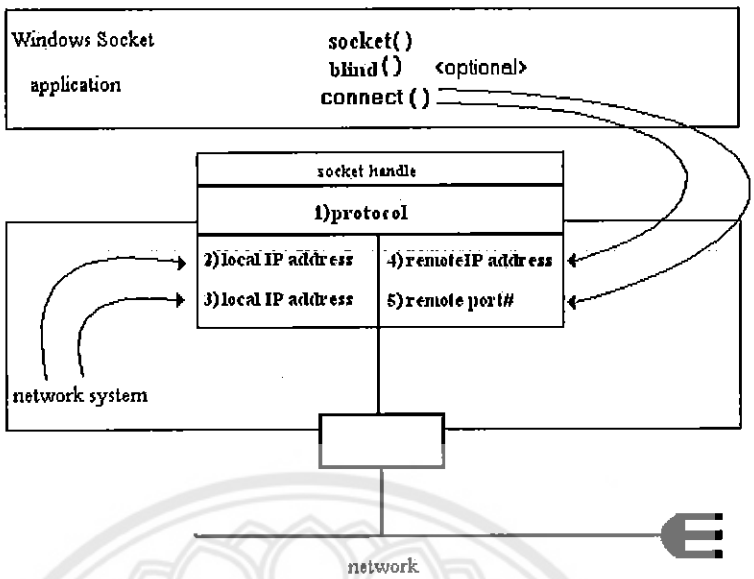
ฟังก์ชัน listen() จะคืนค่าศูนย์เมื่อทำงานสำเร็จและจะคืนค่า SOCKET_ERROR หากทำงานผิดพลาด

การเริ่มการสื่อสารของไคลเอนต์ สำหรับการเชื่อมต่อแบบทีซีพี ทางฝั่งไคลเอนต์จะต้องเรียกฟังก์ชัน connect() เพื่อที่จะเริ่มการเชื่อมต่อกับเซิร์ฟเวอร์

```
int connect(SOCKET s,           /* an unconnected socket */
            struct sockaddr *addr,  /*remote port and IP address*/
            int namelen);           /* address structure length */
```

ตัวอย่าง nRet=connect(theSocket,(LPSOCKADDR)&saServer,sizeof(struct sockaddr));

ก่อนที่จะเรียกฟังก์ชัน connect() เราจะต้องกำหนดค่าให้กับโครงสร้างซ็อกเก็ตแอดเดรสซะก่อน โดยจะต้องกำหนดค่า sin_port และ sin_addr ของ ซ็อกเก็ตทางฝั่งเซิร์ฟเวอร์ (ซ็อกเก็ตของฝั่งตรงข้ามเราจะเรียกว่า remote socket) ฟังก์ชัน connect() จะทำการระบุที่อยู่ (address) และพอร์ตของซ็อกเก็ตทางฝั่งไคลเอนต์ให้เอง ดังนั้นฟังก์ชัน bind() จึงไม่จำเป็นสำหรับโปรแกรมทางฝั่งไคลเอนต์ ฟังก์ชัน connect() จะคืนค่าศูนย์เมื่อทำงานสำเร็จและจะคืนค่า SOCKET_ERROR เมื่อทำงานผิดพลาด



รูปที่ 2.21 รูปแบบการเชื่อมต่อในระบบเครือข่าย

(ที่มา : Windows Socket Network Programming ,Bob Quinn and Dave Shute)

เซิร์ฟเวอร์ตอบรับการเชื่อมต่อ

สำหรับการเชื่อมต่อแบบทีซีพี เซิร์ฟเวอร์จะทำการตอบสนองต่อฟังก์ชัน connect() ของไคลเอนต์ ด้วยฟังก์ชัน accept() ซึ่งจะคอยตรวจสอบการ connect() ของไคลเอนต์

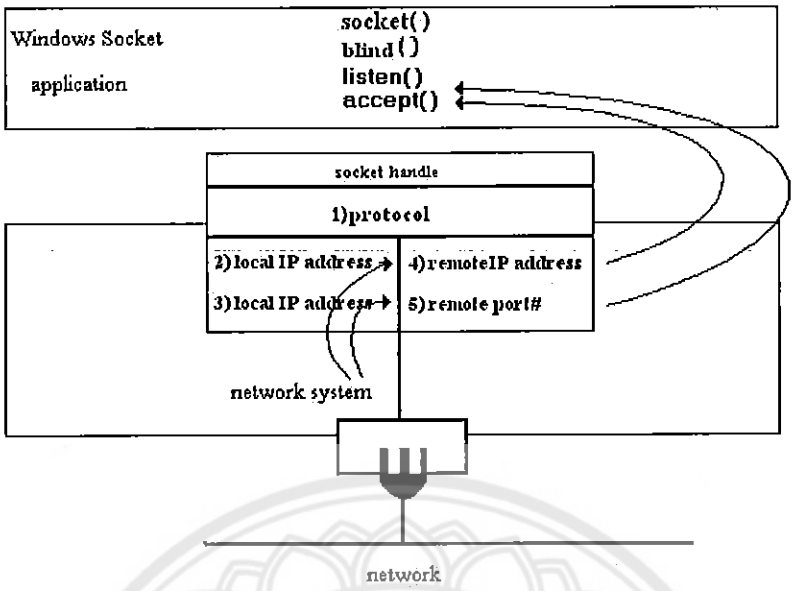
ฟังก์ชัน accept() คืนค่าเป็นซ็อกเก็ตใหม่หลังจากที่ได้รับการร้องขอการติดต่อจาก ซ็อกเก็ตที่รอการเชื่อมต่อ (listening socket) รูปแบบของฟังก์ชัน accept() คือ

```
SOCKET accept(SOCKET s,      /* a listening socket */
              struct sockaddr *addr,    /* name of incoming socket */
              int *addrlen);            /*  length of sockaddr */
```

ตัวอย่าง

```
SOCKET remoteSocket;
remoteSocket = accept(listenSocket,      //Listening socket
                      NULL,              //Optional client address
                      NULL);
```

ฟังก์ชัน accept() จะคืนค่าเป็น INVALID_SOCKET ถ้าทำงานผิดพลาด



รูปที่ 2.22 การส่งและรับข้อมูลระหว่างเครือข่าย

(ที่มา : Windows Socket Network Programming ,Bob Quinn and Dave Shute)

4. ส่งและรับข้อมูลระหว่างซ็อกเก็ต (Send and receive between sockets)

การส่งข้อมูลของซ็อกเก็ตที่ได้ทำการเชื่อมต่อ (connect) เรียบร้อยแล้ว เราจะใช้ฟังก์ชัน send() นั่นหมายความว่าก่อนที่จะใช้ฟังก์ชัน send() ได้ จะต้องมีการเรียกฟังก์ชันconnect() สำเร็จก่อน รูปแบบของฟังก์ชัน send() ดังนี้คือ

```
int     send(SOCKET s,              /* associated socket */
             const char *buf,       /* buffer with outgoing data */
             int len,               /* bytes to send */
             int flags);            /* option flags */
```

s: socket handle

buf : ตัวชี้ที่ชี้ไปยังข้อมูลที่จะส่งออกไป

len : ความยาวของข้อมูลที่จะส่ง (หน่วยเป็น ไบต์)

flags : สัญญาณที่จะส่ง

การรับข้อมูล

เราจะใช้ฟังก์ชัน recv() ในการรับข้อมูล โดยมีรูปแบบการใช้งานดังนี้

```
int     recv(SOCKET s,          /* associated socket */
```

```
    char FAR *buf,          /* buffer with outgoing data */
    int len,                /* bytes to send */
    int flags);             /* option flags */
```

ทั้งฟังก์ชัน send() และ recv() เมื่อทำงานผิดพลาด จะคืนค่า SOCKET_ERROR และจะคืนค่าเป็นตัวเลขจำนวนไบต์ที่ส่งหรือรับข้อมูล เมื่อทำงานสำเร็จ

5. ปิดซ็อกเก็ต

ทุกครั้งที่มีการเปิดซ็อกเก็ต เราจะต้องปิดซ็อกเก็ตเมื่อทำงานเสร็จแล้วเสมอ เพื่อไม่ให้เป็นการจองทรัพยากรของเครื่องไปเปล่า ๆ สำหรับการปิดซ็อกเก็ตมีรูปแบบการใช้งานดังนี้

```
int closesocket(SOCKET s);      /* a valid socket */
```

2.7.5 กลไกการทำงานของไคลเอนต์-เซิร์ฟเวอร์

ในการพัฒนาโปรแกรมที่ทำงานในระบบเครือข่ายโดยการใช้วินซ็อก เราจะสามารถสร้างโปรแกรมที่เป็นไคลเอนต์ และ โปรแกรมที่เป็นเซิร์ฟเวอร์ ขึ้นมาคู่กัน โดยที่เราเพียงปรับปรุงโปรแกรมเล็กน้อย เพราะขั้นตอนการทำงานของไคลเอนต์และเซิร์ฟเวอร์มีลักษณะการเรียกใช้งานฟังก์ชันของวินซ็อกเอพีไอที่คล้าย ๆ กัน จะต่างกันเพียงบางขั้นตอนเท่านั้น

นอกจากฟังก์ชันในกลุ่มที่ใช้ในการเชื่อมต่อกันระหว่างเซิร์ฟเวอร์แล้ว วินซ็อกยังมีฟังก์ชันอื่น ๆ ที่ใช้เพื่อเตรียมการเชื่อมต่อ หรือประโยชน์อื่น ๆ อีก เช่น ฟังก์ชันที่ใช้ในการรับค่าแอดเดรสของเครื่อง ฟังก์ชันที่ใช้ในการกำหนดรูปแบบการทำงานของซ็อกเก็ต และอื่น ๆ อีกมาก

Client	Server
Socket()	socket()
Initialize sockaddr_in structure	Initialize sockaddr_in structure
With server(remote) socket name	With server(local) socket name
	bind()
	listen()
Connect() ----------→	
	accept()
< association created , either side can send or receive>	
Send() ----------→	recv()
Recv() ←----------	send()
Closesocket()	Closesocket() (connected socket)
	Closesocket() (listening socket)

รูปที่ 2.23 สรุปกลไกการทำงานของไคลเอนต์ - เซิร์ฟเวอร์

(ที่มา : Windows Socket Network Programming ,Bob Quinn and Dave Shute)

จากรูป เป็นรูปแบบการเชื่อมต่อและรับส่งข้อมูล ระหว่างไคลเอนต์ และ เซิร์ฟเวอร์ ที่ทำงานแบบ ทีซีพี

บทที่ 3
วิธีการดำเนินงาน

ในบทที่ 2 ได้กล่าวถึงความรู้พื้นฐานที่จำเป็นสำหรับการพัฒนา FTP ต่อไปจะขอกล่าวถึงวิธีการดำเนินงานในโครงงานพัฒนาโปรแกรมสำหรับเครื่องแม่ข่ายเพื่อการถ่ายโอนข้อมูลแบบ FTP ดังนี้

3.1 การดำเนินงานโครงงาน

การดำเนินงานพัฒนาโปรแกรมสำหรับเครื่องแม่ข่ายเพื่อการถ่ายโอนข้อมูลแบบเอฟทีพีนั้นมีขั้นตอนการดำเนินงานดังนี้

- ศึกษาทฤษฎีที่เกี่ยวข้องและกำหนดขอบเขตในการศึกษา

โดยทฤษฎีที่เกี่ยวข้องนั้นประกอบด้วยเรื่องต่าง ๆ ดังนี้

◆ ทีซีพี/ไอพี (TCP/IP) และ โอเอสไอโมเดล (OSI Model) เนื่องจาก เอฟทีพี(FTP) เป็นบริการ(Service) หนึ่งของโปรโตคอลทีซีพี/ไอพี และ โอเอสไอโมเดลเป็นรูปแบบมาตรฐานของโปรแกรมที่ทำงานในระบบเครือข่าย เราจึงต้องศึกษาและทำความเข้าใจกับกลไกการทำงานของทีซีพี/ไอพีและโอเอสไอโมเดล เพื่อให้เข้าใจถึงกลไกการสื่อสารข้อมูลในระบบเครือข่าย

◆ RFC959(Request For Comments 959) เป็นมาตรฐานของระบบเอฟทีพี ที่ทั่วโลกยอมรับ ดังนั้นการพัฒนาโปรแกรมถ่ายโอนข้อมูลแบบเอฟทีพี จึงใช้ RFC959 เป็นขอบเขตในการศึกษา เพื่อให้โปรแกรมที่พัฒนามีความสามารถตามที่มาตรฐานกำหนดไว้ ซึ่งมาตรฐานที่กำหนดไว้นั้นมีดังนี้

Type – ASCII Non-print

Mode – Stream

Commands – USER,QUIT,PORT,TYPE,MODE,STRU

For the default value RETR,STOR,NOOP

The default values for transfer parameters are

Type – ASCII Non – print

Mode – Stream

STRU – File

หมายความว่า กำหนดให้รูปแบบการรับส่งข้อมูลตามปกติ ข้อมูลจะเป็นรหัส ASCII และรูปแบบการส่งข้อมูลเป็นแบบ Stream (TCP) คือ ไคลเอนต์ต้องติดต่อกับเซิร์ฟเวอร์ก่อนจึงจะส่งข้อมูลได้ และโครงสร้างข้อมูลจะอยู่ในลักษณะแฟ้มข้อมูล (File) นอกจากนี้จะต้องสามารถรับคำสั่งพื้นฐานได้ คือ USER,QUIT,PORT,TYPE,MODE,STRU และ คำสั่งสำหรับร้องขอไฟล์ หรือ ส่งไฟล์ และสั่งให้อยู่เฉย ๆ คือ RETR,STOR,NOOP

♦ วินซ็อกเอพีไอ(Winsock API) เป็นมาตรฐานการสื่อสารข้อมูลของโปรแกรมที่ทำงานบนระบบปฏิบัติการวินโดวส์ และถือเป็นเครื่องมือสำคัญในการพัฒนาโปรแกรมบนระบบเครือข่าย

- ออกแบบโปรแกรม

ออกแบบว่าจะให้โปรแกรมทำงานบนระบบปฏิบัติการรุ่นใด ต้องใช้เครื่องมืออะไรบ้าง และโปรแกรมมีลักษณะการทำงานอย่างไร ซึ่งรายละเอียดของการออกแบบ จะกล่าวถึงในหัวข้อที่ 3.2

- พัฒนาโปรแกรม

แบ่งโปรแกรมออกเป็นส่วน ๆ แล้ว พัฒนาทีละส่วนเพื่อมารวมกัน ในการพัฒนาโปรแกรม ผู้พัฒนาจะต้องมีความเข้าใจเกี่ยวกับการทำงานของระบบปฏิบัติการ เครื่องมือที่ใช้ และเข้าใจในหลักการทำงานของโปรแกรมที่เราจะพัฒนา

- ทดสอบและแก้ไข

หลักจากที่พัฒนาเสร็จแล้ว นำมาทดสอบการใช้งาน และ แก้ไขจุดที่ผิดพลาด หรือ เพิ่มเติมความสามารถ และ ปรับแต่งให้มีประสิทธิภาพมากขึ้นและใช้งานได้ง่ายขึ้น

- วิเคราะห์และสรุปผล

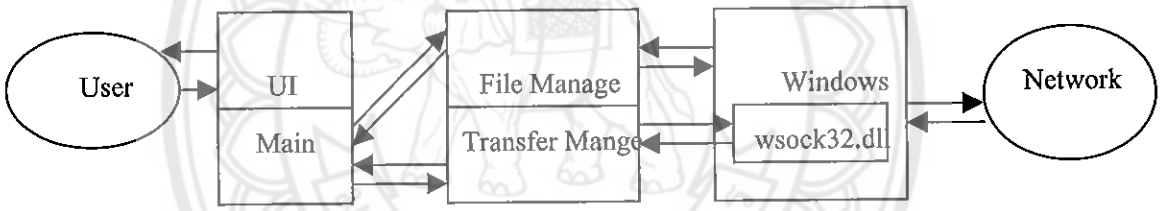
เมื่อปรับปรุงแก้ไขจนถึงจุดที่คิดว่าใช้ได้แล้ว วิเคราะห์ถึงปัญหาที่เกิดขึ้น และสรุปผลและแนวทางในการพัฒนาลำดับที่สูงขึ้นต่อไป

3.2 การออกแบบ

โปรแกรมจะทำงานในระบบปฏิบัติการวินโดวส์ 95/98 หรือ วินโดวส์ NT พัฒนาโดยใช้ภาษา C/C++ (คณะผู้จัดทำโครงงานใช้ Microsoft Visual C/C++ 6.0) และตัวโปรแกรมจะต้องมีความสามารถของเอฟทีพี อ้างอิงตามมาตรฐาน RFC959 คือ

- รูปแบบการส่งข้อมูลเป็นแบบสตรีม
- ชนิดของข้อมูลเป็นแบบ ASCII
- โครงสร้างข้อมูลเป็นแบบไฟล์ (File)
- คำสั่งพื้นฐานที่สามารถใช้ได้ คือ USER , QUIT , PORT , TYPE, MODE, STRU RETR , STOR,NOOP
- การตอบสนอง (FTP Reply) จะเป็นเลข 3 หลัก โดยมีความหมายตามที่ระบุใน RFC959

องค์ประกอบของโปรแกรม



รูปที่ 3.1 ส่วนประกอบของโปรแกรมที่ออกแบบ

User Interface(UI) คือ ส่วนที่จะติดต่อกับผู้ใช้ มีหน้าที่รับข้อมูลและคำสั่งจากผู้ใช้ และแสดงผลการทำงานให้ผู้ใช้ได้รับทราบ

File Management คือ ส่วนที่ดูแลและจัดการเกี่ยวกับไฟล์

Main คือ ส่วน ส่วนที่ดูแลเรื่องการสั่งงานส่วนอื่น ๆ และ ตอบสนอง เพื่อให้การทำงานเป็นไปตามกลไกของเอฟทีพี

Transfer Management คือ ส่วนที่ดูแลเรื่องการรับส่ง ข้อมูล โดยการใช้งานวินซ็อกเอพีไอผ่านทางไลบรารีไฟล์ wsock32.dll

3.3 ขั้นตอนและรายละเอียดของการ Reply

จะเป็นการแสดงรูปแบบการ Reply ของแต่ละคำสั่ง ซึ่งมีดังต่อไปนี้

- Connection Establishment

- 120 Service ready in mmm minutes. (บอกว่ามีการเริ่มต้นการติดต่อและจะแล้วเสร็จในกี่วินาที)

- 220 host name is ready. (บอกว่าการเตรียมการติดต่อกับ host name(ชื่อของเครื่องแม่ข่าย) ได้เสร็จเรียบร้อยแล้ว)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการเตรียมพร้อมต่อการติดต่อได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

- Login

USER

- 230 Welcome to host name. User name logged in. (ขอต้อนรับสู่ FTP ของ host name นั้นๆ ผู้ใช้ได้ทำการล็อกอินเรียบร้อยแล้ว ในกรณีที่ไม่ต้องใช้รหัสผ่าน)

- 530 Not logged in. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะความผิดพลาดของระบบ)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการเตรียมพร้อมต่อการติดต่อได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
ทั้ง 3 Reply นี้ (500,501,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 331 User name okay, need password. (ชื่อผู้ใช้งานได้รับการยอมรับให้เข้าสู่ระบบ แต่ยังต้องการรหัสผ่านก่อนการใช้งานจริงอยู่)

- 332 Need account for login. (ต้องมีการใช้ account ของผู้ใช้ด้วย จึงจะสามารถเข้าใช้งานได้)

ทั้ง 2 Reply นี้ (331,332) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

PASS

- 230 Welcome to host name. User name logged in. (ขอต้อนรับสู่ FTP ของ host name นั้นๆ ผู้ใช้ได้ทำการล็อกอินเรียบร้อยแล้ว ในกรณีที่ต้องใช้รหัสผ่าน)
- 202 Command not implemented, superfluous at this site. (เป็นกรณีที่ไม่จำเป็นต้องใช้รหัสผ่านในการเข้าทำการล็อกอินในระบบนี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะความผิดพลาดของระบบ)
- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 503 Bad sequence of command. (คำสั่งนี้ทำไม่ได้เพราะมีการเรียงคำสั่งผิดพลาด)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการเตรียมพร้อมต่อการติดต่อได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

ทั้ง 4 Reply นี้ (500,501,503,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 332 Need account for login. (ต้องมีการใช้ account ของผู้ใช้ด้วย จึงจะสามารถเข้าใช้งานได้)

ACCT

- 230 Welcome to host name. User name logged in. (ขอต้อนรับสู่ FTP ของ host name นั้นๆ ผู้ใช้ได้ทำการล็อกอินเรียบร้อยแล้ว ในกรณีที่ต้องใช้รหัสผ่าน)
- 202 Command not implemented, superfluous at this site. (เป็นกรณีที่ไม่จำเป็นต้องใช้รหัสผ่านในการเข้าทำการล็อกอินในระบบนี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะความผิดพลาดของระบบ)
- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 503 Bad sequence of command. (คำสั่งนี้ทำไม่ได้เพราะมีการเรียงคำสั่งผิดพลาด)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการเตรียมพร้อมต่อการติดต่อได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

ทั้ง 4 Reply นี้ (500,501,503,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

CWD

- 250 Requested file action okay, completed. (สามารถทำงานภายในไดเร็คทอรี่ที่ต้องการได้ โดยไม่จำเป็นต้องล็อกอินใหม่)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการเตรียมพร้อมต่อการติดต่อได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

- 530 Not logged in. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะความผิดพลาดของระบบ)

- 550 Request action not taken. File unavailable.(e.g.,file not found , no access)(ไม่สามารถเข้าไปยังไดเร็คทอรีที่ต้องการได้ เนื่องจากไม่สามารถเข้าถึงไฟล์นั้นๆได้(เพราะไม่มีไฟล์ หรือ ไฟล์ถูกป้องกันการเข้าใช้งาน))

ทั้ง 6 Reply นี้ (500,501,502,503,421,550) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

CDUP

- 200 PORT Command successful. (สามารถใช้งานคำสั่งนี้ได้)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการเตรียมพร้อมต่อการติดต่อได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

- 530 Not logged in. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะความผิดพลาดของระบบ)

- 550 Request action not taken. File unavailable.(e.g.,file not found , no access)(ไม่สามารถเข้าไปยังไดเร็คทอรีที่ต้องการได้ เนื่องจากไม่สามารถเข้าถึงไฟล์นั้นๆได้(เพราะไม่มีไฟล์ หรือ ไฟล์ถูกป้องกันการเข้าใช้งาน))

ทั้ง 6 Reply นี้ (500,501,502,503,421,550) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

SMNT

- 202 Command not implemented, superfluous at this site. (เป็นกรณีที่ไม่จำเป็นต้องใช้รหัสผ่านในการเข้าทำการล็อกอินในระบบนี้)

- 250 Requested file action okay, completed. (สามารถทำงานภายในไดเร็คทอรี่ที่ต้องการได้ โดยไม่จำเป็นต้องล็อกอินใหม่)

ทั้ง 2 Reply นี้ (202,250) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการเตรียมพร้อมต่อการติดต่อได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

- 530 Not logged in. (ในกรณีที่ไม่สามารถล็อกอินได้ เพราะความผิดพลาดของระบบ)

- 550 Request action not taken. File unavailable.(e.g.,file not found , no access)(ไม่สามารถเข้าไปยังไดเร็คทอรีที่ต้องการได้ เนื่องจากไม่สามารถเข้าถึงไฟล์นั้นๆได้(เพราะไม่มีไฟล์ หรือ ไฟล์ถูกป้องกันการเข้าใช้งาน))

ทั้ง 6 Reply นี้ (500,501,502,503,421,550) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- Logout

REIN

- 120 Service ready in mmm minutes. (บอกว่ามีการเริ่มต้นการเลิกการติดต่อและจะแล้วเสร็จในกี่วินาที)

- 220 host name is ready. (บอกว่าการเตรียมการเลิกการติดต่อกับ host name(ชื่อของเครื่องแม่ข่าย) ได้เสร็จเรียบร้อยแล้ว)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการเตรียมพร้อมต่อการเลิกการติดต่อได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถล็อกเอาท์ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)

ทั้ง 2 Reply นี้ (500,502) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

QUIT

- 221 Service closing control connection. Good bye. (ทำการปิดการติดต่อทาง control connection และออกจากการใช้งาน)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถล็อกเอาท์ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

-Transfer parameters

PORT

- 200 PORT Command successful. (สามารถใช้งานคำสั่งนี้ได้)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 4 Reply นี้ (500,501,421,530) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

PASV

- 227 Entering Passive Mode (h1, h2, h3, h4, p1, p2). (เข้าสู่การใช้งานคำสั่งโหมดนี้ โดยจะคอย "Listen" ทาง Data port)
- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,502,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

MODE

- 200 PORT Command successful. (สามารถใช้งานคำสั่งนี้ได้)
- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 504 Command not implemented for that parameter. (คำสั่งไม่สามารถใช้ได้กับค่าการใช้งานแบบนี้ได้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,504,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

TYPE

- 200 PORT Command successful. (สามารถใช้งานคำสั่งนี้ได้)
- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 504 Command not implemented for that parameter. (คำสั่งไม่สามารถใช้ได้กับค่าการใช้งานแบบนี้ได้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,504,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

STRU

- 200 PORT Command successful. (สามารถใช้งานคำสั่งนี้ได้)
- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 504 Command not implemented for that parameter. (คำสั่งไม่สามารถใช้ได้กับค่าการใช้งานแบบนี้ได้)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,504,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- File action commands

ALLO

- 200 PORT Command successful. (สามารถใช้งานคำสั่งนี้ได้)

- 202 Command not implemented, superfluous at this site. (เป็นกรณีที่ไม่จำเป็นต้องใช้รหัสผ่านในการเข้าทำการใช้งานในระบบนี้)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 504 Command not implemented for that parameter. (คำสั่งไม่สามารถใช้ได้กับค่าการใช้งานแบบนี้ได้)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,504,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

REST

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,502,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 350 Requested file action pending further information. (คำสั่งนี้ยังต้องการข้อมูลเพิ่มเติมอยู่)

STOR

- 125 Data connection already open; transfer starting. (Data connection ได้ถูกเปิดเรียบร้อยแล้ว และกำลังเริ่มต้นการถ่ายโอนข้อมูล)
- 150 Opening ASCII mode data connection for current file or current directory. (รายงานว่าได้เปิด data connection เพื่อทำการถ่ายโอนไฟล์ข้อมูลที่มีชื่ออยู่ใน Reply นั้นๆ)

ทั้ง 2 Reply นี้ (125,150) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

(- 110 Restart marker reply. In point that appropriate e.g. MARK yyy = mmm, where yyy is User-process data stream marker,and mmm is server's equivalent marker. (เป็นการกำหนดจุดเพื่อการถ่ายโอนข้อมูลที่เป็นไปอย่างต่อเนื่องในกรณีที่เกิดความล้มเหลวระหว่างการถ่ายโอน ณ เวลาใดๆ))

- 226 Closing data connection. Request file transfer successful. (e.g. file transfer or file abort) (ปิด data connection การถ่ายโอนไฟล์เสร็จสิ้น(อาจเป็นเพราะการถ่ายโอนเสร็จเรียบร้อยหรือเกิดการยกเลิกกลางคันก็ได้))
- 250 Requested file action okay,completed. (ไฟล์ที่ต้องการได้ทำการถ่ายโอนเสร็จสิ้นสมบูรณ์แล้ว)

ทั้ง 2 Reply นี้ (226,250) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 425 Can't open data connection. (ไม่สามารถทำการเปิด Data connection ได้)
- 426 Connection closed, transfer aborted.(เกิดการปิด Control connection ก่อนที่จะถ่ายโอนข้อมูลเสร็จสิ้น การถ่ายโอนข้อมูลจึงถูกยกเลิก)
- 451 Request action aborted: local error in processing. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากเกิดความขัดข้องภายในของเครื่องแม่ข่ายระหว่างการประมวลผล)
- 551 Request action aborted: page type unknown. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากไม่ทราบประเภทของ Page ว่าเป็น 0,1,2 หรือ 3)
- 552 Request action aborted: exceed storage allocation. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากมีเนื้อที่ในการรองรับไฟล์ที่ต้องการไม่พอ

ทั้ง 5 Reply นี้ (425,426,451,551,552) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 532 Need account for storing file. (ต้องการข้อมูล Account เพื่อทำการบันทึกไฟล์)
- 450 Requested file action not taken. File unavailable. (e.g. File busy) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ อาจจะเนื่องมาจากไฟล์นั้นกำลังถูกใช้งานอยู่)
- 452 Requested file action not taken. Insufficient storage space in system. (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เพราะเนื้อที่สำหรับบันทึกภายในระบบมีไม่พอ)
- 553 Requested file action not taken. File name not allowed. (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เพราะไฟล์นี้ไม่ได้รับการอนุญาตให้ถ่ายโอนข้อมูลได้)

ทั้ง 4 Reply นี้ (532,450,452,553) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 4 Reply นี้ (500,501,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

STOU

- 125 Data connection already open; transfer starting. (Data connection ได้ถูกเปิดเรียบร้อยแล้ว และกำลังเริ่มต้นการถ่ายโอนข้อมูล)

- 150 Opening ASCII mode data connection for current file or current directory. (รายงานว่าได้เปิด data connection เพื่อทำการถ่ายโอนไฟล์ข้อมูลที่มีชื่ออยู่ใน Reply นั้นๆ)

ทั้ง 2 Reply นี้ (125,150) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

(- 110 Restart marker reply. In point that appropriate e.g. MARK yyy = mmm, where yyy is User-process data stream marker,and mmm is server's equivalent marker. (เป็นการกำหนดจุดเพื่อการถ่ายโอนข้อมูลที่เป็นไปอย่างต่อเนื่องในกรณีที่เกิดความล้มเหลวระหว่างการถ่ายโอน ณ เวลาใดๆ))

- 226 Closing data connection. Request file transfer successful. (e.g. file transfer or file abort) (ปิด data connection การถ่ายโอนไฟล์เสร็จสิ้น(อาจเป็นเพราะการถ่ายโอนเสร็จเรียบร้อยหรือเกิดการยกเลิกกลางคันก็ได้))

- 250 Requested file action okay,completed. (ไฟล์ที่ต้องการได้ทำการถ่ายโอนเสร็จสิ้นสมบูรณ์แล้ว)

ทั้ง 2 Reply นี้ (226,250) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 425 Can't open data connection. (ไม่สามารถทำการเปิด Data connection ได้)

- 426 Connection closed, transfer aborted.(เกิดการปิด Control connection ก่อนที่จะถ่ายโอนข้อมูลเสร็จสิ้น การถ่ายโอนข้อมูลจึงถูกยกเลิก)

- 451 Request action aborted: local error in processing. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากเกิดความขัดข้องภายในของเครื่องแม่ข่ายระหว่างการประมวลผล)

- 551 Request action aborted: page type unknown. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากไม่ทราบประเภทของ Page ว่าเป็น 0,1,2 หรือ 3

- 552 Request action aborted: exceed storage allocation. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากมีเนื้อที่ในการรองรับไฟล์ที่ต้องการไม่พอ

ทั้ง 5 Reply นี้ (425,426,451,551,552) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 532 Need account for storing file. (ต้องการข้อมูล Account เพื่อทำการบันทึกไฟล์)

- 450 Requested file action not taken. File unavailable. (e.g. File busy) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ อาจจะเนื่องมาจากไฟล์นั้นกำลังถูกใช้งานอยู่)

- 452 Requested file action not taken. Insufficient storage space in system. (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เพราะเนื้อที่สำหรับบันทึกภายในระบบมีไม่พอ)

- 553 Requested file action not taken. File name not allowed. (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เพราะไฟล์นี้ไม่ได้รับการอนุญาตให้ถ่ายโอนข้อมูลได้)

ทั้ง 4 Reply นี้ (532,450,452,553) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 4 Reply นี้ (500,501,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

RETR

- 125 Data connection already open; transfer starting. (Data connection ได้ถูกเปิดเรียบร้อยแล้ว และกำลังเริ่มต้นการถ่ายโอนข้อมูล)

- 150 Opening ASCII mode data connection for current file or current directory. (รายงานว่าได้เปิด data connection เพื่อทำการถ่ายโอนไฟล์ข้อมูลที่มีชื่ออยู่ใน Reply นั้นๆ)

ทั้ง 2 Reply นี้ (125,150) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

(- 110 Restart marker reply. In point that appropriate e.g. MARK yyy = mmm, where yyy is User-process data stream marker,and mmm is server's equivalent marker. (เป็นการกำหนดจุดเพื่อการถ่ายโอนข้อมูลที่เป็นไปอย่างต่อเนื่องในกรณีที่เกิดความล้มเหลวระหว่างการถ่ายโอน ณ เวลาใดๆ))

- 226 Closing data connection. Request file transfer successful. (e.g. file transfer or file abort) (ปิด data connection การถ่ายโอนไฟล์เสร็จสิ้น(อาจเป็นเพราะการถ่ายโอนเสร็จเรียบร้อยหรือเกิดการยกเลิกกลางคันก็ได้))
- 250 Requested file action okay,completed. (ไฟล์ที่ต้องการได้ทำการถ่ายโอนเสร็จสิ้นสมบูรณ์แล้ว)

ทั้ง 2 Reply นี้ (226,250) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 425 Can't open data connection. (ไม่สามารถทำการเปิด Data connection ได้)
- 426 Connection closed, transfer aborted.(เกิดการปิด Control connection ก่อนที่จะถ่ายโอนข้อมูลเสร็จสิ้น การถ่ายโอนข้อมูลจึงถูกยกเลิก)
- 451 Request action aborted: local error in processing. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากเกิดความขัดข้องภายในของเครื่องแม่ข่ายระหว่างการประมวลผล)

ทั้ง 3 Reply นี้ (425,426,451) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 450 Requested file action not taken. File unavailable. (e.g. File busy) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ อาจจะเนื่องมาจากไฟล์นั้นกำลังถูกใช้งานอยู่)
- 550 Requested action not taken. File unavailable.(e.g. file not found, noaccess) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เนื่องมาจากหาไฟล์นั้นไม่พบหรือ มีการกำหนดสิทธิ์ไม่ให้สามารถเข้าถึงได้)

ทั้ง 2 Reply นี้ (450,550) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 4 Reply นี้ (500,501,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

LIST

- 125 Data connection already open; transfer starting. (Data connection ได้ถูกเปิดเรียบร้อยแล้ว และกำลังเริ่มต้นการถ่ายโอนข้อมูล)

- 150 Opening ASCII mode data connection for current file or current directory. (รายงานว่าได้เปิด data connection เพื่อทำการถ่ายโอนไฟล์ข้อมูลที่มีชื่ออยู่ใน Reply นั้นๆ)

ทั้ง 2 Reply นี้ (125,150) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 226 Closing data connection. Request file transfer successful. (e.g. file transfer or file abort) (ปิด data connection การถ่ายโอนไฟล์เสร็จสิ้น(อาจเป็นเพราะการถ่ายโอนเสร็จเรียบร้อยหรือเกิดการยกเลิกกลางคันก็ได้))

- 250 Requested file action okay,completed. (ไฟล์ที่ต้องการได้ทำการถ่ายโอนเสร็จสิ้นสมบูรณ์แล้ว)

ทั้ง 2 Reply นี้ (226,250) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 425 Can't open data connection. (ไม่สามารถทำการเปิด Data connection ได้)

- 426 Connection closed, transfer aborted.(เกิดการปิด Control connection ก่อนที่จะถ่ายโอนข้อมูลเสร็จสิ้น การถ่ายโอนข้อมูลจึงถูกยกเลิก)

- 451 Request action aborted: local error in processing. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากเกิดความขัดข้องภายในของเครื่องแม่ข่ายระหว่างการประมวลผล)

ทั้ง 3 Reply นี้ (425,426,451) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 450 Requested file action not taken. File unavailable. (e.g. File busy) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ อาจจะเนื่องมาจากไฟล์นั้นกำลังถูกใช้งานอยู่)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,502,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

NLIST

- 125 Data connection already open; transfer starting. (Data connection ได้ถูกเปิดเรียบร้อยแล้ว และกำลังเริ่มต้นการถ่ายโอนข้อมูล)
- 150 Opening ASCII mode data connection for current file or current directory. (รายงานว่าได้เปิด data connection เพื่อทำการถ่ายโอนไฟล์ข้อมูลที่มีชื่ออยู่ใน Reply นั้นๆ)

ทั้ง 2 Reply นี้ (125,150) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 226 Closing data connection. Request file transfer successful. (e.g. file transfer or file abort) (ปิด data connection การถ่ายโอนไฟล์เสร็จสิ้น(อาจเป็นเพราะการถ่ายโอนเสร็จเรียบร้อยหรือเกิดการยกเลิกกลางคันก็ได้))
- 250 Requested file action okay,completed. (ไฟล์ที่ต้องการได้ทำการถ่ายโอนเสร็จสิ้นสมบูรณ์แล้ว)

ทั้ง 2 Reply นี้ (226,250) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 425 Can't open data connection. (ไม่สามารถทำการเปิด Data connection ได้)
- 426 Connection closed, transfer aborted.(เกิดการปิด Control connection ก่อนที่จะถ่ายโอนข้อมูลเสร็จสิ้น การถ่ายโอนข้อมูลจึงถูกยกเลิก)
- 451 Request action aborted: local error in processing. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากเกิดความขัดข้องภายในของเครื่องแม่ข่ายระหว่างการประมวลผล)

ทั้ง 3 Reply นี้ (425,426,451) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 450 Requested file action not taken. File unavailable. (e.g. File busy) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ อาจจะเนื่องมาจากไฟล์นั้นกำลังถูกใช้งานอยู่)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,502,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

APPE

- 125 Data connection already open; transfer starting. (Data connection ได้ถูกเปิดเรียบร้อยแล้ว และกำลังเริ่มต้นการถ่ายโอนข้อมูล)
- 150 Opening ASCII mode data connection for current file or current directory. (รายงานว่าได้เปิด data connection เพื่อทำการถ่ายโอนไฟล์ข้อมูลที่มีชื่ออยู่ใน Reply นั้นๆ)

ทั้ง 2 Reply นี้ (125,150) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

(- 110 Restart marker reply. In point that appropriate e.g. MARK yyy = mmm, where yyy is User-process data stream marker,and mmm is server's equivalent marker. (เป็นการกำหนดจุดเพื่อการถ่ายโอนข้อมูลที่เป็นไปอย่างต่อเนื่องในกรณีที่เกิดความล้มเหลวระหว่างการถ่ายโอน ณ เวลาใดๆ))

- 226 Closing data connection. Request file transfer successful. (e.g. file transfer or file abort) (ปิด data connection การถ่ายโอนไฟล์เสร็จสิ้น(อาจเป็นเพราะการถ่ายโอนเสร็จเรียบร้อยหรือเกิดการยกเลิกกลางคันก็ได้))

- 250 Requested file action okay,completed. (ไฟล์ที่ต้องการได้ทำการถ่ายโอนเสร็จสิ้นสมบูรณ์แล้ว)
ทั้ง 2 Reply นี้ (226,250) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 425 Can't open data connection. (ไม่สามารถทำการเปิด Data connection ได้)
- 426 Connection closed, transfer aborted.(เกิดการปิด Control connection ก่อนที่จะถ่ายโอนข้อมูลเสร็จสิ้น การถ่ายโอนข้อมูลจึงถูกยกเลิก)
- 451 Request action aborted: local error in processing. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากเกิดความขัดข้องภายในของเครื่องแม่ข่ายระหว่างการประมวลผล)
- 551 Request action aborted: page type unknown. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากไม่ทราบประเภทของ Page ว่าเป็น 0,1,2 หรือ 3
- 552 Request action aborted: exceed storage allocation. (การใช้งานคำสั่งนี้ถูกยกเลิกเนื่องจากมีเนื้อที่ในการรองรับไฟล์ที่ต้องการไม่พอ
ทั้ง 5 Reply นี้ (425,426,451,551,552) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 532 Need account for storing file. (ต้องการข้อมูล Account เพื่อทำการบันทึกไฟล์)
- 450 Requested file action not taken. File unavailable. (e.g. File busy) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ อาจจะเนื่องมาจากไฟล์นั้นกำลังถูกใช้งานอยู่)
- 550 Requested action not taken. File unavailable.(e.g. file not found, noaccess) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เนื่องมาจากหาไฟล์นั้นไม่พบหรือ มีการกำหนดสิทธิ์ไม่ให้สามารถเข้าถึงได้)
- 452 Requested file action not taken. Insufficient storage space in system. (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เพราะเนื้อที่สำหรับบันทึกภายในระบบมีไม่พอ)
- 553 Requested file action not taken. File name not allowed. (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เพราะไฟล์นี้ไม่ได้รับการอนุญาตให้ถ่ายโอนข้อมูลได้)
ทั้ง 5 Reply นี้ (532,450,550,452,553) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,502,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

RNFR

- 450 Requested file action not taken. File unavailable. (e.g. File busy) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ อาจจะเนื่องมาจากไฟล์นั้นกำลังถูกใช้งานอยู่)

- 550 Requested action not taken. File unavailable.(e.g. file not found, noaccess) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เนื่องมาจากหาไฟล์นั้นไม่พบหรือ มีการกำหนดสิทธิ์ไม่ให้สามารถเข้าถึงได้)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,502,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 350 Requested file action pending further information. (คำสั่งนี้ยังต้องการข้อมูลเพิ่มเติมอยู่)

RNTO

- 250 Requested file action okay,completed. (ไฟล์ที่ต้องการได้ทำการถ่ายโอนเสร็จสิ้นสมบูรณ์แล้ว)
- 532 Need account for storing file. (ต้องการข้อมูล Account เพื่อทำการบันทึกไฟล์)
- 553 Requested file action not taken. File name not allowed. (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เพราะไฟล์นี้ไม่ได้รับการอนุญาตให้ถ่ายโอนข้อมูลได้)

ทั้ง 2 Reply นี้ (532,553) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)
- 503 Bad sequence of commands. (เกิดการเรียงคำสั่งไม่ถูกต้องตามลำดับ)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 6 Reply นี้ (500,501,502,503,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

DELE

- 250 Requested file action okay,completed. (ไฟล์ที่ต้องการได้ทำการถ่ายโอนเสร็จสิ้นสมบูรณ์แล้ว)
- 450 Requested file action not taken. File unavailable. (e.g. File busy) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ อาจจะเนื่องมาจากไฟล์นั้นกำลังถูกใช้งานอยู่)
- 550 Requested action not taken. File unavailable.(e.g. file not found, noaccess) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เนื่องมาจากหาไฟล์นั้นไม่พบหรือ มีการกำหนดสิทธิ์ไม่ให้สามารถเข้าถึงได้)

ทั้ง 2 Reply นี้ (450,550) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,502,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

RMD

- 250 Requested file action okay,completed. (ไฟล์ที่ต้องการได้ทำการถ่ายโอนเสร็จสิ้นสมบูรณ์แล้ว)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)
- 550 Requested action not taken. File unavailable.(e.g. file not found, noaccess) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เนื่องมาจากหาไฟล์นั้นไม่พบหรือ มีการกำหนดสิทธิ์ไม่ให้สามารถเข้าถึงได้)

ทั้ง 6 Reply นี้ (500,501,502,530,421,550) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

MKD

- 257 "PATHNAME"created. (ชื่อ Pathname จะถูกสร้างขึ้นมา)
- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)
- 550 Requested action not taken. File unavailable.(e.g. file not found, noaccess) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เนื่องมาจากหาไฟล์นั้นไม่พบหรือ มีการกำหนดสิทธิ์ไม่ให้สามารถเข้าถึงได้)

ทั้ง 6 Reply นี้ (500,501,502,530,421,550) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

PWD

- 257 "PATHNAME"created. (ชื่อ Pathname จะถูกสร้างขึ้นมา)
- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

- 550 Requested action not taken. File unavailable.(e.g. file not found, noaccess) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ เนื่องมาจากหาไฟล์นั้นไม่พบหรือ มีการกำหนดสิทธิ์ไม่ให้สามารถเข้าถึงได้)

ทั้ง 5 Reply นี้ (500,501,502,421,550) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

ABOR

- 225 Data connection open; no transfer in progress. (data connection เปิดอยู่ แต่ไม่มีการถ่ายโอนข้อมูลระหว่างนั้น)

- 226 Closing data connection. Requested file action successful. (ปิด data connection เพราะถูกใช้คำสั่งยกเลิก)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

ทั้ง 4 Reply นี้ (500,501,502,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- Information commands

SYST

- 215 NAME system type. (ชื่อของระบบปฏิบัติการที่ใช้อยู่ในขณะนั้น)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)

- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

ทั้ง 4 Reply นี้ (500,501,502,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

STAT

- 211 System status or system help reply. (สถานะของระบบในขณะนั้นหรือข้อมูลช่วยเหลือผู้ใช้เกี่ยวกับระบบ)
- 212 Directory status. (สถานะของไดเร็คทอรีที่ต้องการทราบ)
- 213 File status. (สถานะของไฟล์ข้อมูลที่ต้องการทราบ)

ทั้ง 3 Reply นี้ (211,212,213) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- 450 Requested file action not taken. File unavailable. (e.g. File busy) (ไม่สามารถทำการถ่ายโอนไฟล์ที่ขอได้ อาจจะเนื่องมาจากไฟล์นั้นกำลังถูกใช้งานอยู่)
- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง 5 Reply นี้ (500,501,502,530,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

HELP

- 211 System status or system help reply. (สถานะของระบบในขณะนั้นหรือข้อมูลช่วยเหลือผู้ใช้เกี่ยวกับระบบ)
- 214 Help message. (ข้อความที่มีไว้อธิบายคำสั่งที่ต้องการทราบ มีไว้สำหรับผู้ใช้โดยเฉพาะ)

- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 502 Command not implemented. (คำสั่งนี้ไม่มีการใช้ในระบบนี้)
- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

ทั้ง 4 Reply นี้ (500,501,502,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

- Miscellaneous commands

SITE

- 200 PORT Command successful. (สามารถใช้งานคำสั่งนี้ได้)
- 202 Command not implemented, superfluous at this site. (เป็นกรณีที่ไม่จำเป็นต้องใช้รหัสผ่านในการเข้าทำการล็อกอินในระบบนี้)
- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)
- 501 Syntax error in parameters or arguments. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์รูปแบบการใช้งานผิดพลาด)
- 530 Not logged in. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะความผิดพลาดของระบบ)

ทั้ง Reply นี้ (500,501,530) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

NOOP

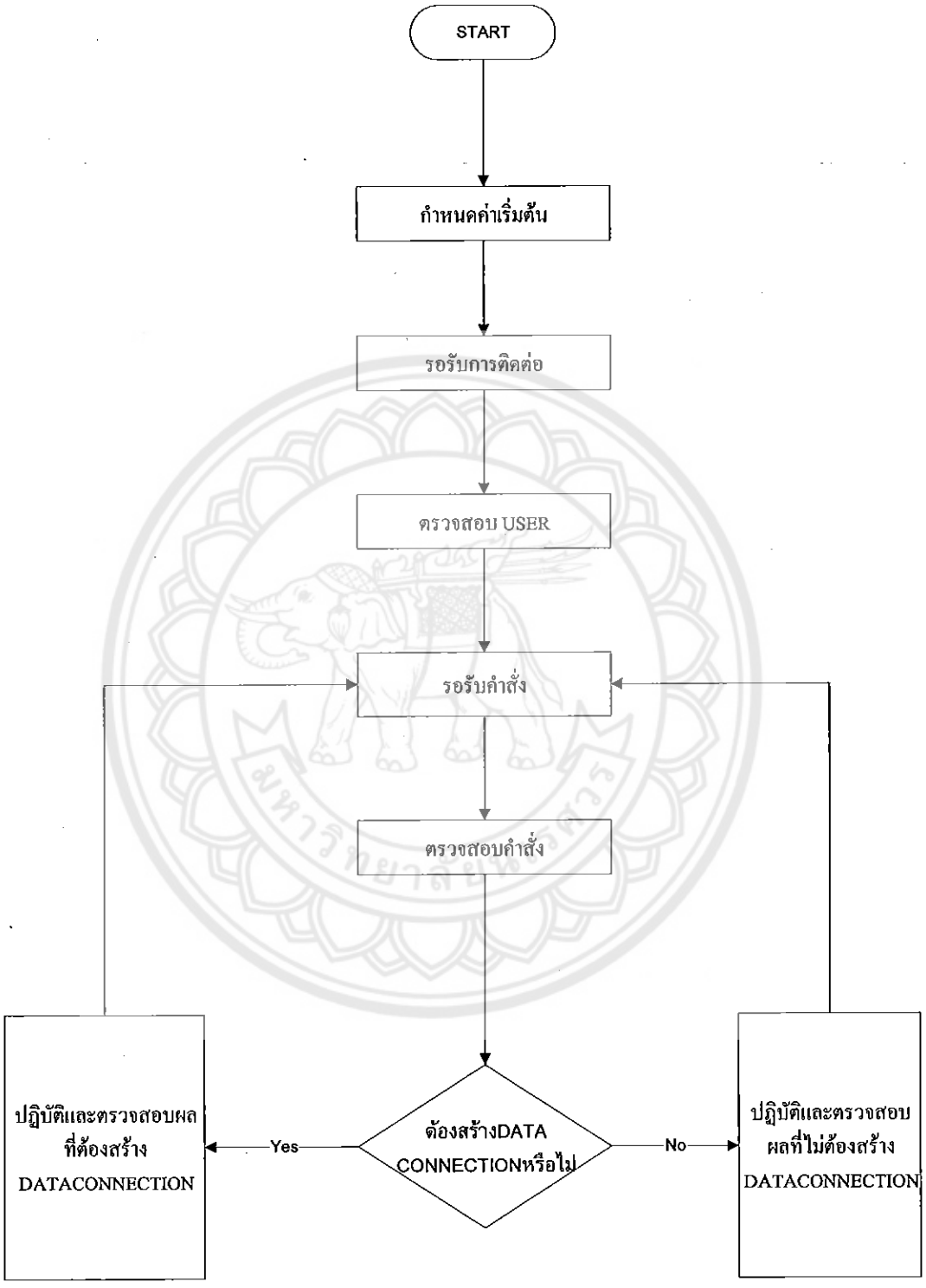
- 200 PORT Command successful. (สามารถใช้งานคำสั่งนี้ได้)
- 500 Syntax error, command unrecognized. This may include error such as command line too long. (ในกรณีที่ไม่สามารถใช้งานคำสั่งนี้ได้ เพราะพิมพ์คำสั่งผิดพลาด อาจเป็นเพราะคำสั่งมีความยาวเกินไป เช่น มี<SP>เข้ามาต่อจากคำสั่งก่อนที่จะมี end-of-line code)

- 421 Service not available, closing control connection. (ในกรณีที่ไม่สามารถทำการใช้งานคำสั่งนี้ได้ เครื่องแม่ข่ายจะส่ง Reply นี้)

ทั้ง 2 Reply นี้ (500,421) จะส่งเพียงตัวใดตัวหนึ่งเท่านั้น

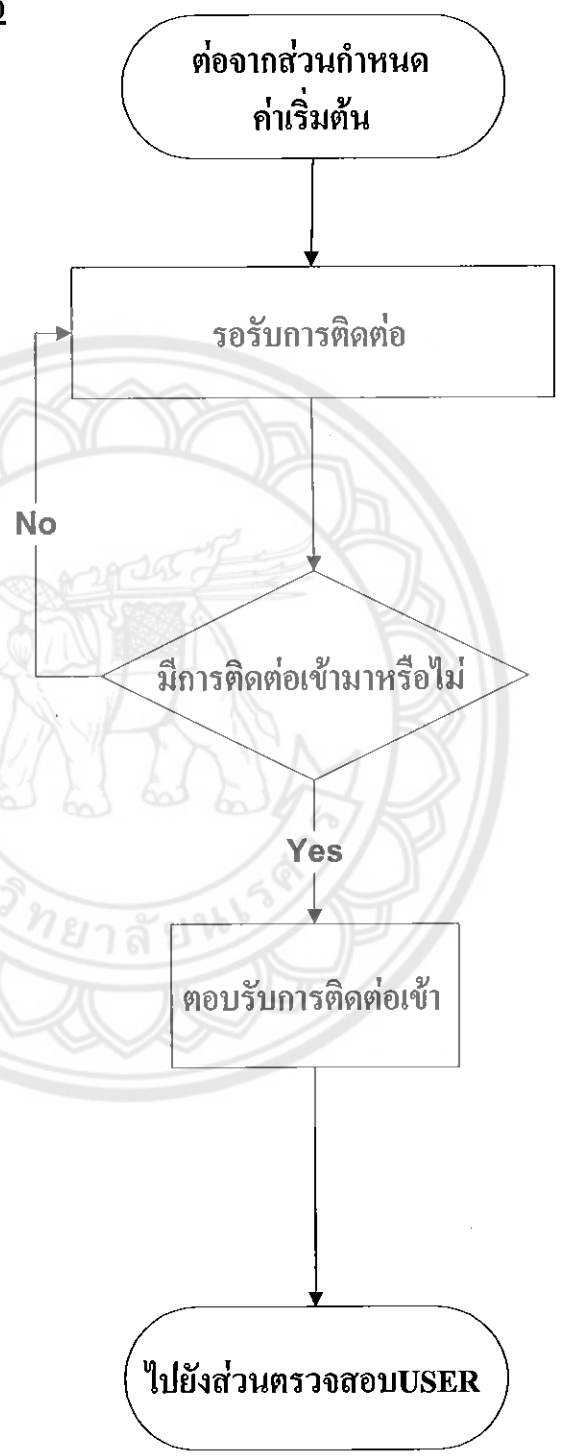
และจากนี้ไปจะเป็น Flow chart หรือรูปแบบการทำงานของโปรแกรมถ่ายโอนข้อมูลทั้งหมด ซึ่งจะเป็นการแสดงภาพโปรแกรมโดยคร่าวๆ และจะมีการแสดงส่วนรายละเอียดส่วนที่สำคัญดังนี้

3.2.2 ผังการทำงานโดยรวมของโปรแกรม

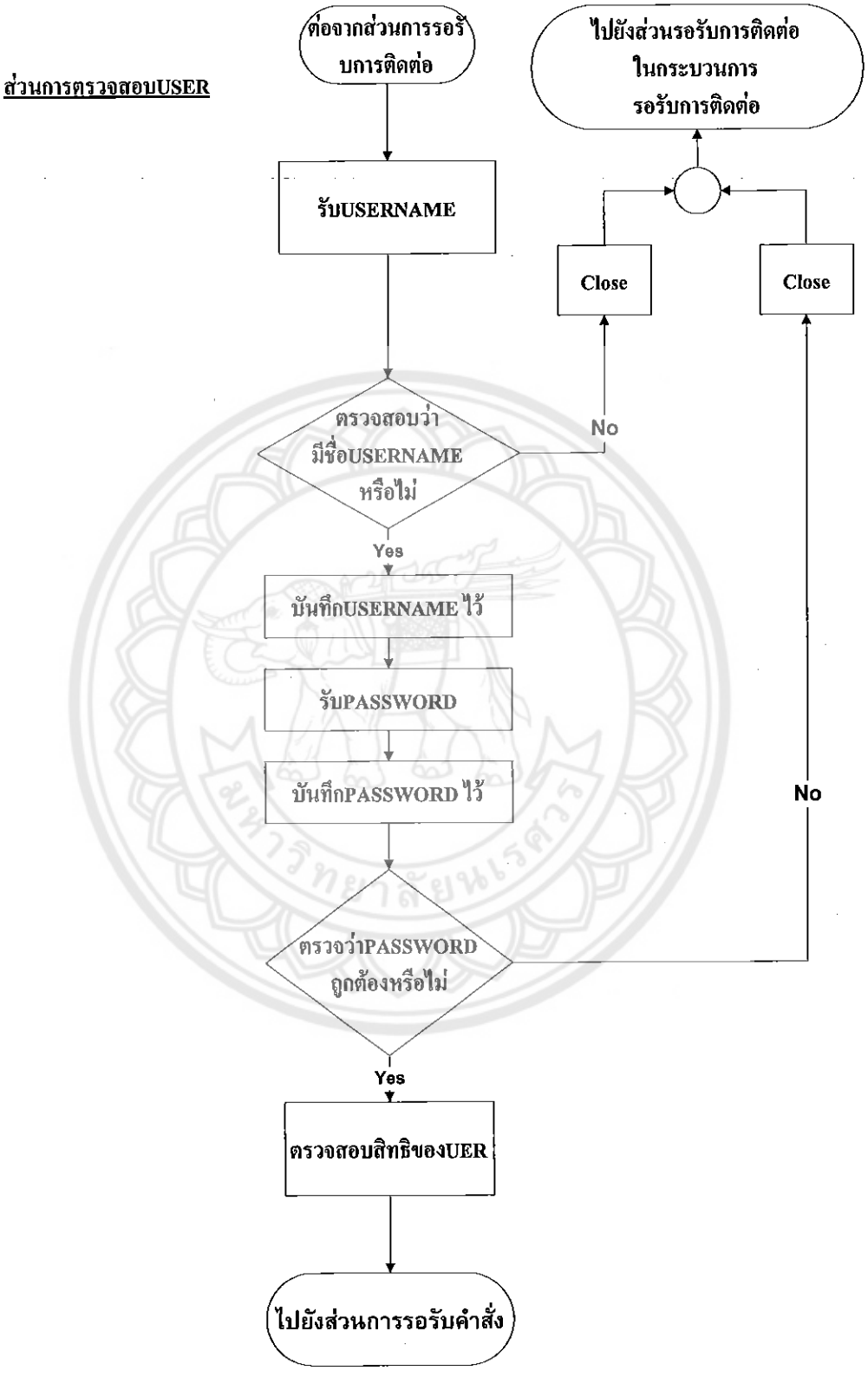


รูปที่3.2 รูปผังการทำงานโดยรวมของโปรแกรม

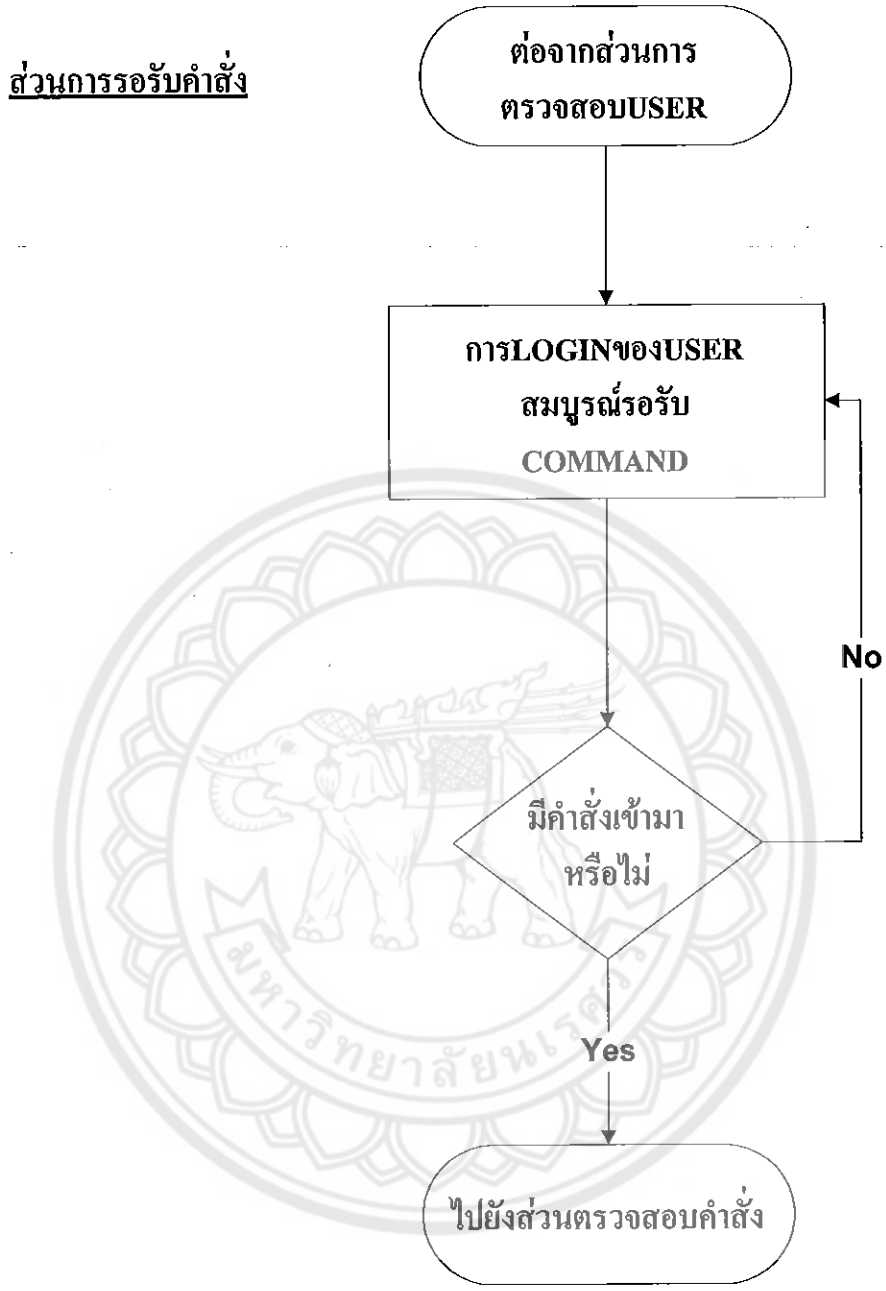
ส่วนการรอรับการติดต่อ



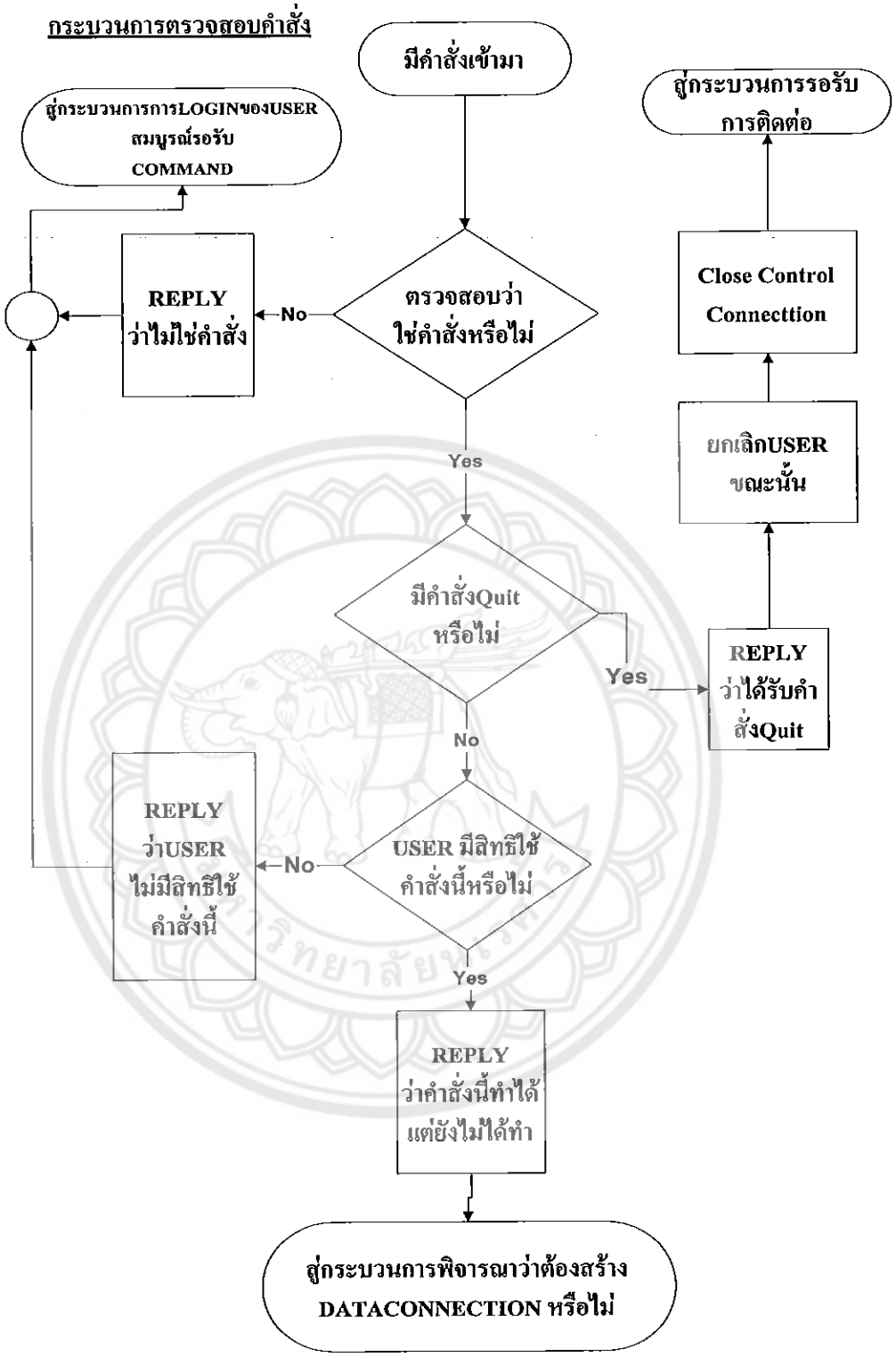
รูปที่ 3.3 รูปผังการทำงานของส่วนการรอรับการติดต่อ



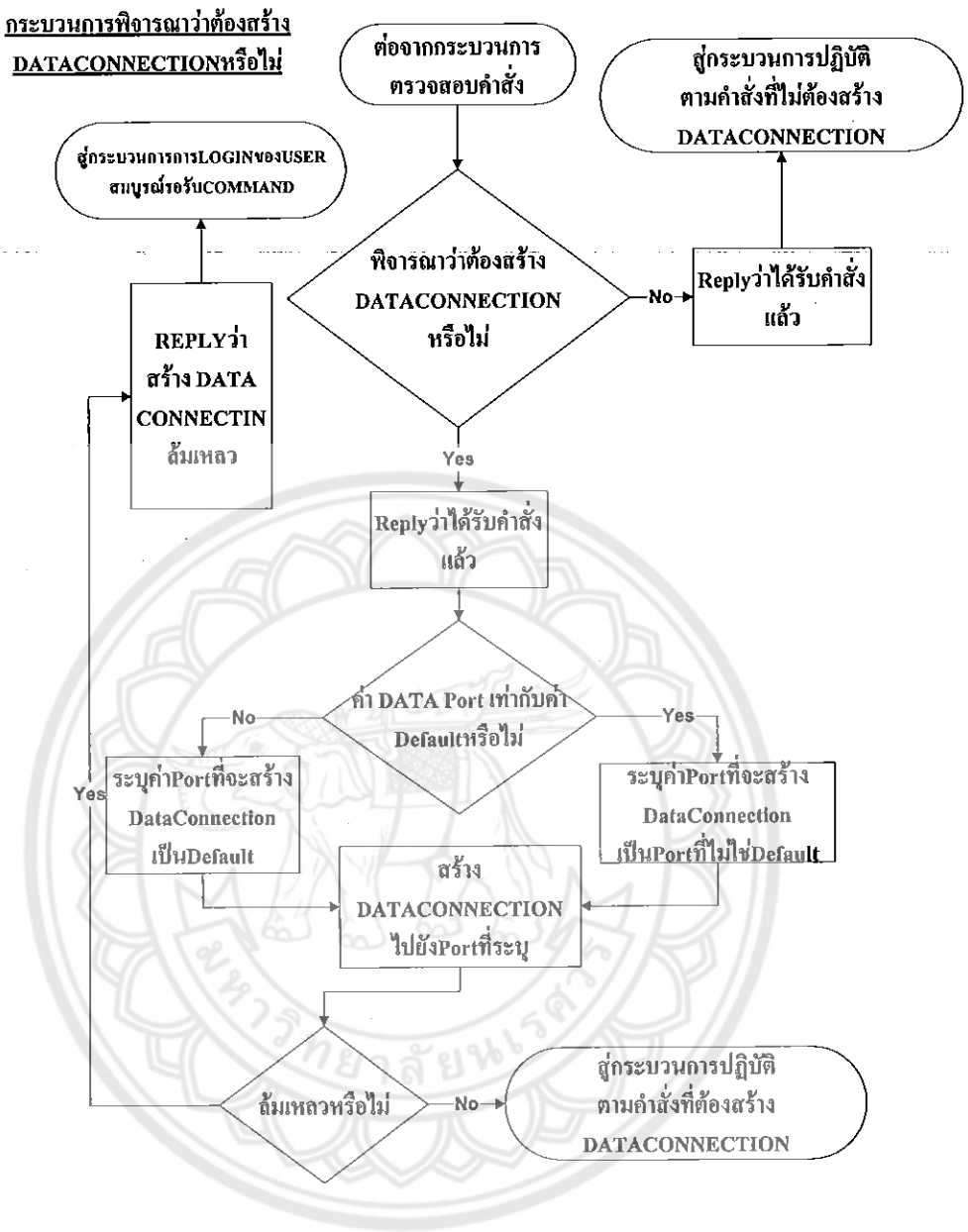
รูปที่ 3.4 รูปผังการทำงานของส่วนการตรวจสอบUSER



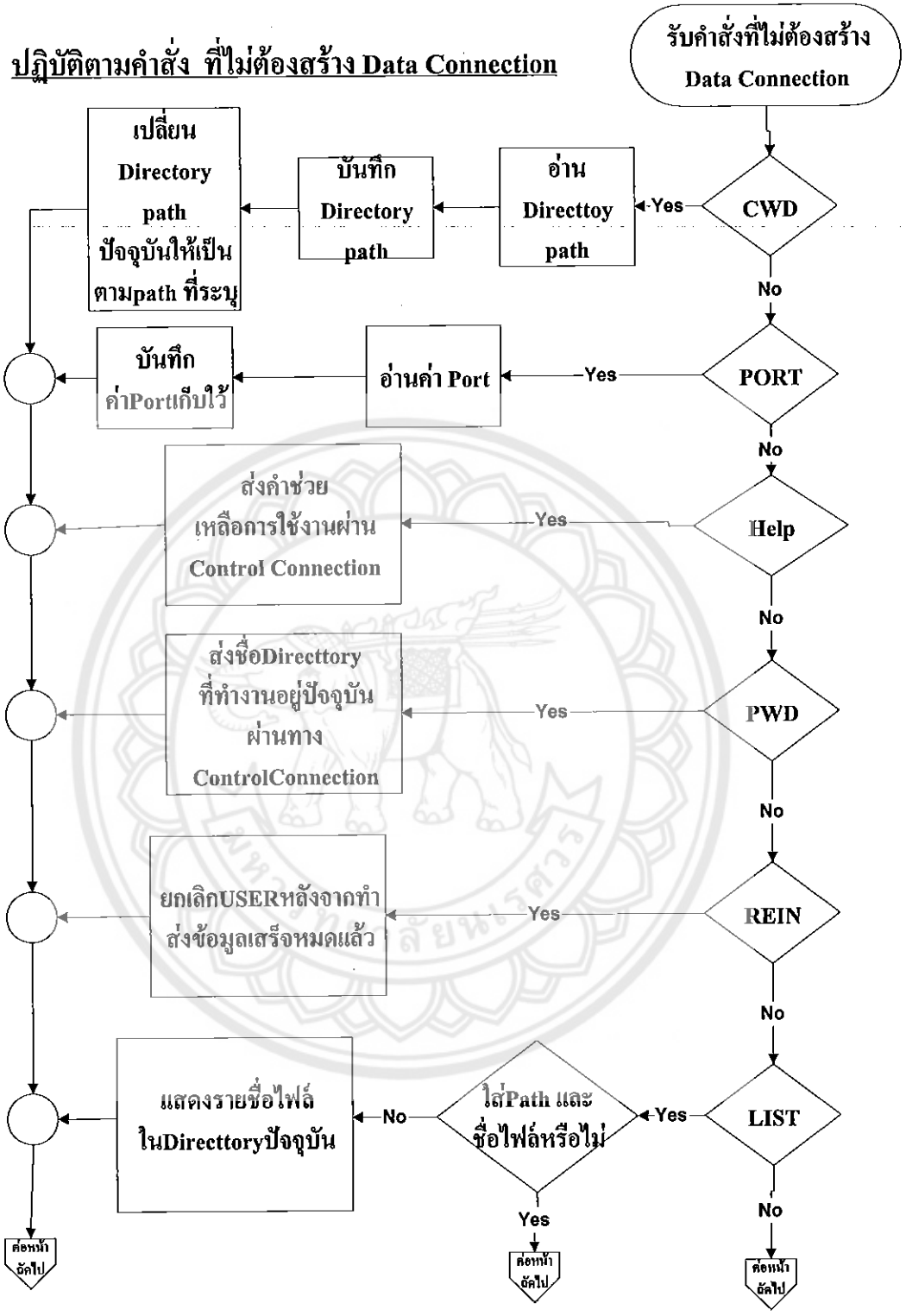
รูปที่ 3.5 รูปผังการทำงานของส่วนการรอรับคำสั่ง



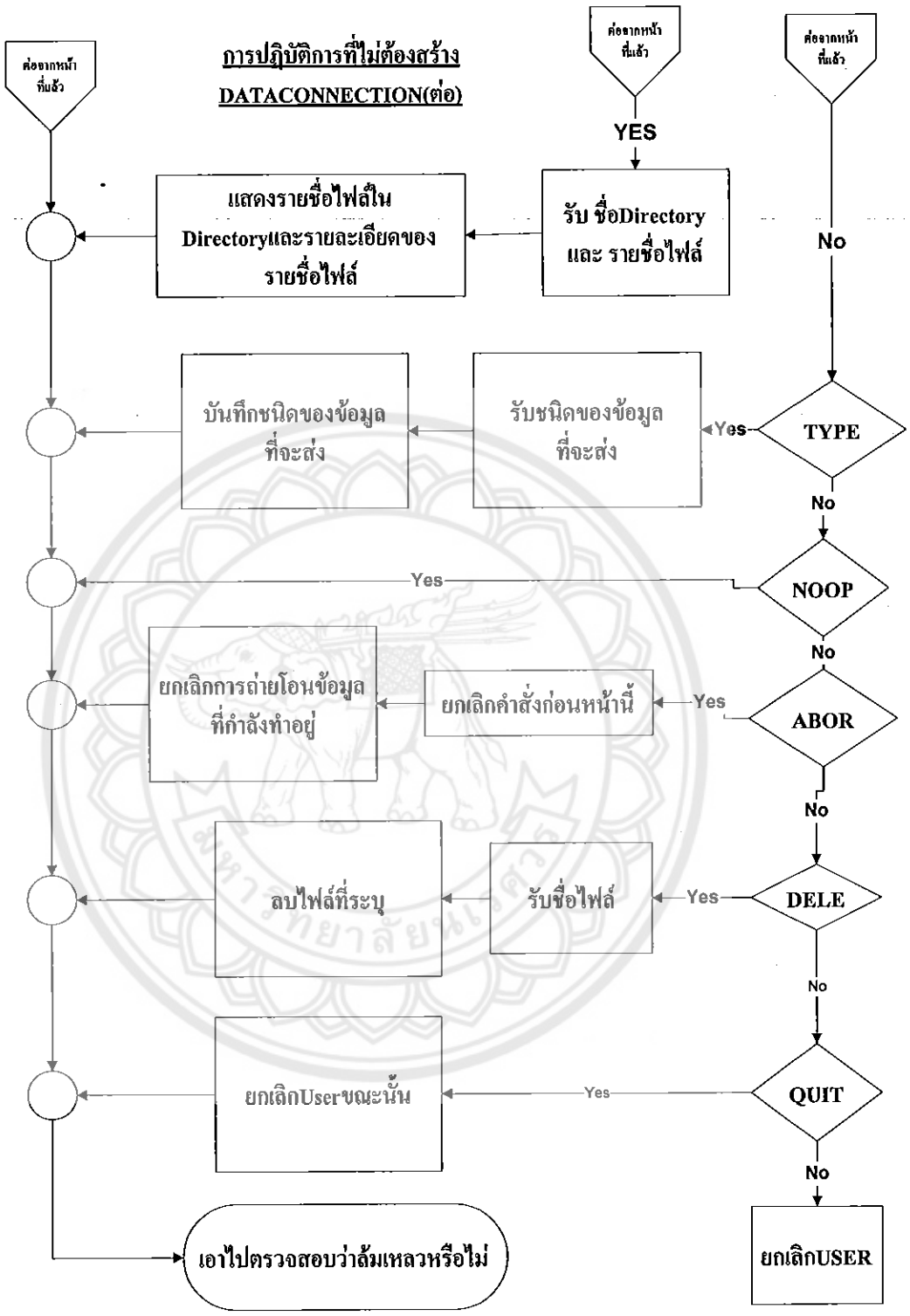
รูปที่ 3.6 รูปผังการทำงานของส่วนการตรวจสอบคำสั่ง



รูปที่ 3.7 รูปผังการทำงานของส่วนการพิจารณาว่าต้องสร้างDATACONNECTIONหรือไม่

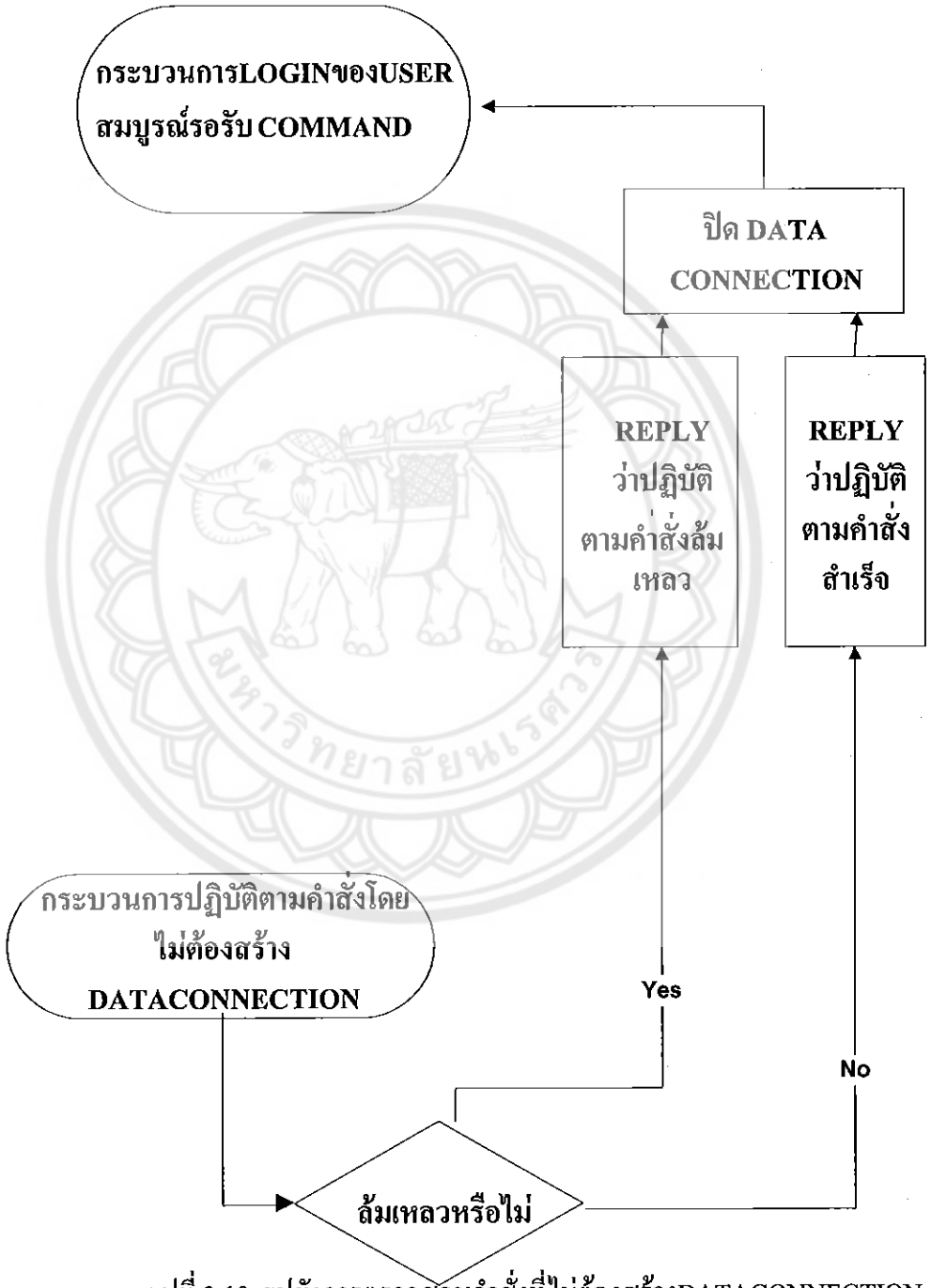


รูปที่ 3.8 รูปผังการปฏิบัติตามคำสั่งที่ไม่ต้องสร้างDATACONNECTION

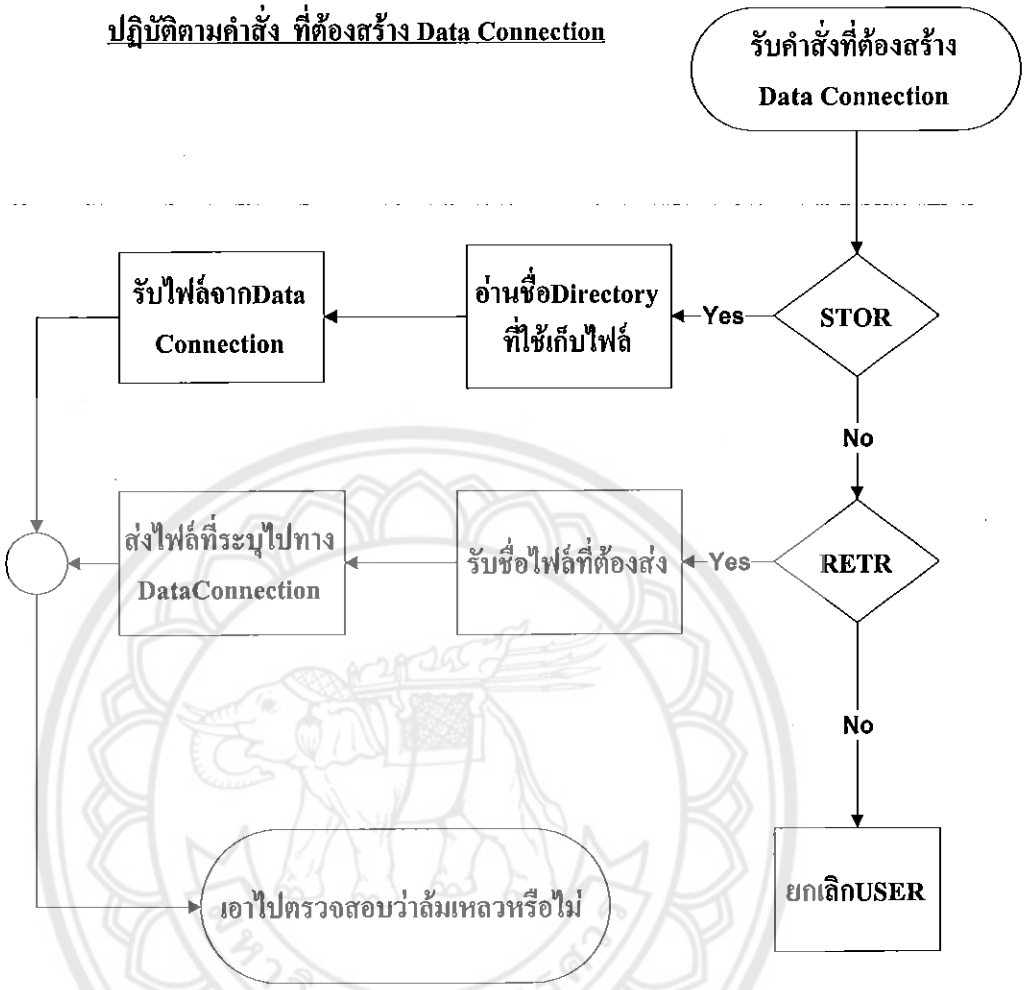


รูปที่ 3.9 รูปผังการปฏิบัติตามคำสั่งที่ไม่ต้องสร้างDATACONNECTION(ต่อ)

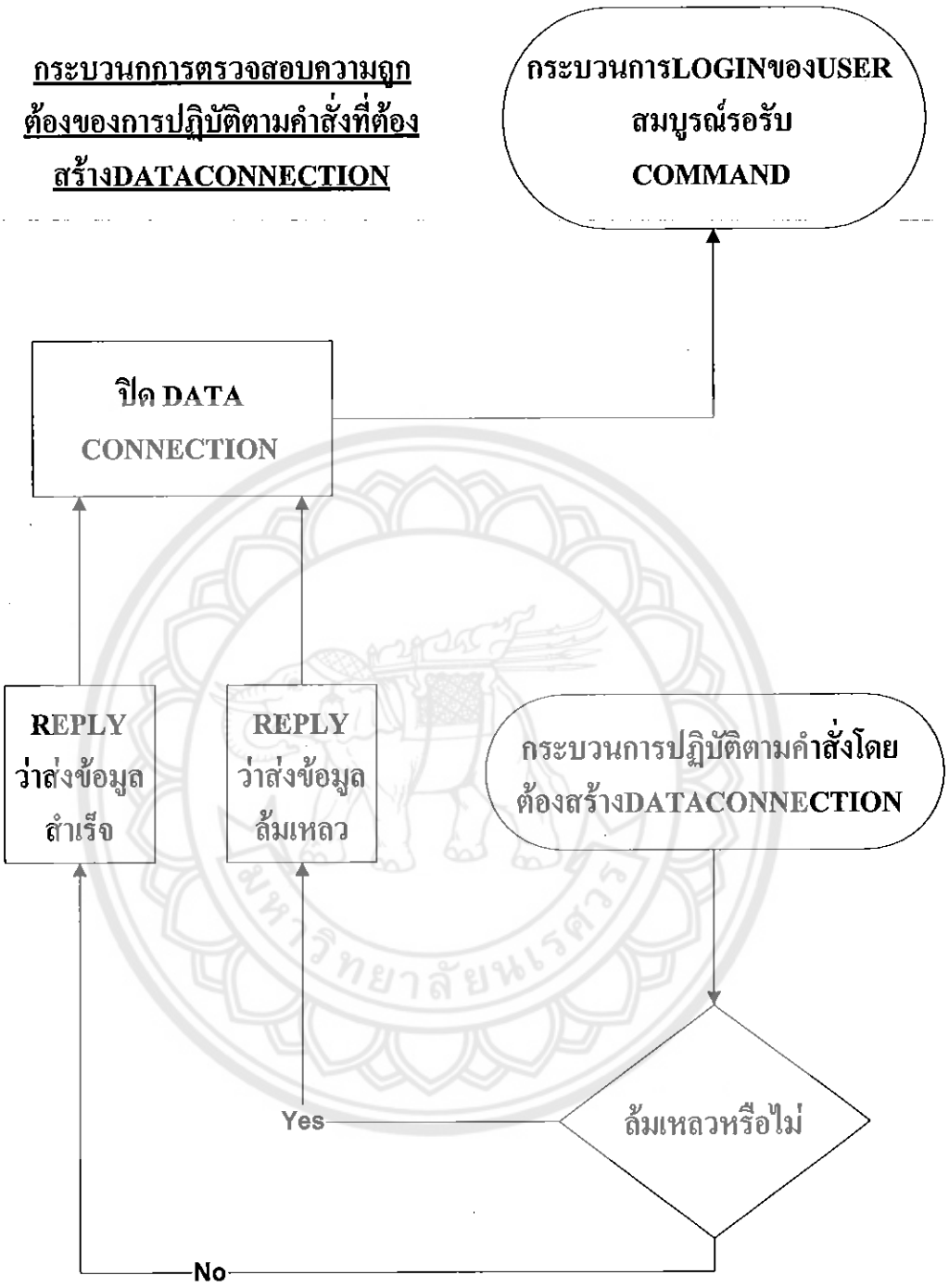
กระบวนการตรวจสอบความถูกต้องของการปฏิบัติตาม
คำสั่งที่ไม่ต้องสร้างDATACONNECTION



รูปที่ 3.10 รูปผังการตรวจสอบคำสั่งที่ไม่ต้องสร้างDATACONNECTION



รูปที่ 3.11 รูปผังการปฏิบัติตามคำสั่งที่ต้องสร้างDATACONNECTION



รูปที่ 3.12 รูปผังการตรวจสอบคำสั่งที่ต้องสร้างDATACONNECTION

ทั้ง 4 ส่วนนี้ เป็นองค์ประกอบของโปรแกรมที่เราจะพัฒนาขึ้น โดยเป็นการออกแบบในลักษณะภาพรวมทั้งหมดก่อน ขั้นตอนต่อไปคือ การกำหนดข้อตกลงเพื่อให้ทั้ง 4 ส่วนที่ถูกสร้างขึ้นมาแยกกันนั้นสามารถทำงานร่วมกันได้ โดยผู้ใช้ไม่ได้รู้สึกว่าโปรแกรมถูกแยกออกเป็นส่วน ๆ

3.3 การจัดสร้างชิ้นงาน

โปรแกรมทำงานในระบบปฏิบัติการวินโดวส์ และพัฒนาขึ้นตามมาตรฐาน RFC959 โดยใช้ TCP/IPโปรโตคอล และวินโดว์ซอกเก็ตเป็นกลไกสำคัญของโปรแกรม และใช้ภาษา C/C++ เป็นภาษาในการสร้างและพัฒนา เพื่อให้สามารถทำงานแบบมัลติทาสก์(Multitask) ได้ และสามารถทำงานได้อย่างมีประสิทธิภาพ โดยการพัฒนาโปรแกรมนี้ จะแบ่งโปรแกรมออกเป็นส่วนๆ เพื่อให้ง่ายต่อการสร้างและปรับปรุงแก้ไข จากนั้นนำแต่ละส่วนมารวมกันและปรับปรุงแก้ไขอีกครั้ง

3.4 ขั้นตอนการทดลอง

- ทดลองสร้างโปรแกรมทั่วไปแบบง่าย ๆ ในวินโดวส์ เพื่อให้เข้าใจถึงการพัฒนาโปรแกรมในวินโดวส์
- ทดลองพัฒนาโปรแกรมรับ-ส่งข้อมูลทั้งในระหว่างเครื่องเดียวกัน และ ในระบบเครือข่าย โดยใช้วินซ็อกเอพีไอ
- ทดลองเขียนโปรแกรมจัดการกับระบบไฟล์
- ทดลองเขียนโปรแกรมที่ทำงานตามกลไกของเอฟทีพี
- นำสิ่งที่ทำลองและโปรแกรมส่วนต่าง ๆ มารวมกัน และ ปรับปรุงแก้ไข

บทที่ 4
การทดสอบโปรแกรมและวิเคราะห์ผล

ระหว่างการทำโครงการ ในช่วงการพัฒนาโปรแกรม ทางกลุ่มผู้จัดทำโครงการได้ ทำการทดสอบโปรแกรมไปพร้อม ๆ กับการพัฒนา เพราะการพัฒนาโปรแกรมในระบบเครือข่ายโดยใช้วินโดวส์ซ็อกเก็ตนั้น เป็นเรื่องที่ซับซ้อน อีกทั้งกลุ่มผู้จัดทำโครงการยังขาดประสบการณ์ในการพัฒนาโปรแกรมลักษณะนี้ จึงได้ทำการแบ่งโปรแกรมออกเป็นส่วนย่อย ๆ และพัฒนาทีละส่วนดังที่ได้เสนอไปแล้วในบทที่ 3

สำหรับในบทนี้ ทางกลุ่มผู้จัดทำได้ทำการทดลองพัฒนาโปรแกรมในส่วนที่สำคัญที่สุด คือ ส่วนการติดต่อผ่านทางระบบเครือข่าย โดยใช้วินโดวส์ซ็อกเก็ต ตามขั้นตอนดังต่อไปนี้

4.1 ออกแบบการทดลอง

เราได้ออกแบบการทดลองขึ้นมา 5 การทดลอง ดังนี้

- การทดลองที่ 1 ทดสอบการเชื่อมต่อระหว่างเซิร์ฟเวอร์ กับ ไคลเอนท์ ภายในเครื่องเดียวกัน (loop back address) เป็นการทดลองระหว่างการพัฒนาโปรแกรม เพื่อให้เข้าใจกลไกการทำงานของไคลเอนท์-เซิร์ฟเวอร์ และ วินโดวส์ซ็อกเก็ต
- การทดลองที่ 2 ทดสอบการเชื่อมต่อระหว่างเซิร์ฟเวอร์ กับ ไคลเอนท์ ต่างเครื่องกัน เพื่อทดสอบว่าสามารถเชื่อมต่อผ่านระบบเครือข่ายได้จริง
- การทดลองที่ 3 ทดสอบการเชื่อมต่อโดยใช้ FTP client ที่ทำงานในระบบปฏิบัติการ DOS เพื่อทดสอบการรับส่งข้อมูล (FTP command และ FTP reply)
- การทดลองที่ 4 ทดสอบการเชื่อมต่อโดยใช้ FTP client ที่ทำงานในระบบปฏิบัติการ Windows เพื่อทดสอบการรับส่งข้อมูล (FTP command และ FTP reply)
- การทดลองที่ 5 ทดสอบการถ่ายโอนข้อมูล เพื่อทดสอบว่าสามารถรับส่งไฟล์ได้

4.2 ดำเนินการทดลอง

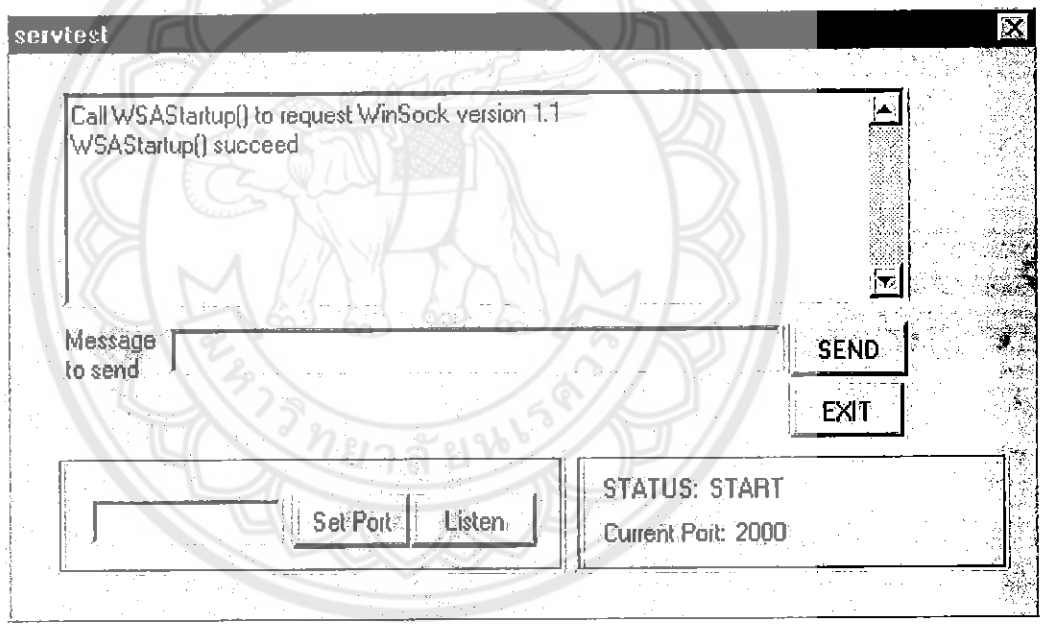
การทดลองที่ 1

ทดสอบการเชื่อมต่อระหว่างเซิร์ฟเวอร์ กับ ไคลเอนต์ ภายในเครื่องเดียวกัน (loop back address)

ในการทดลองนี้ เราได้ทำการพัฒนาโปรแกรม server ขึ้นมาเอง เพื่อทดสอบการเชื่อมต่อกันในแบบง่าย ๆ ส่วน client นั้น จะใช้โปรแกรม telnet ซึ่งมีในระบบปฏิบัติการ Windows อยู่แล้ว โดยทั้ง server และ client จะทำงานอยู่ในเครื่องเดียวกัน และ มี IP address เดียวกัน คือ 127.0.0.1 (สำหรับเครื่องที่ไม่ได้อยู่ในระบบเครือข่าย)

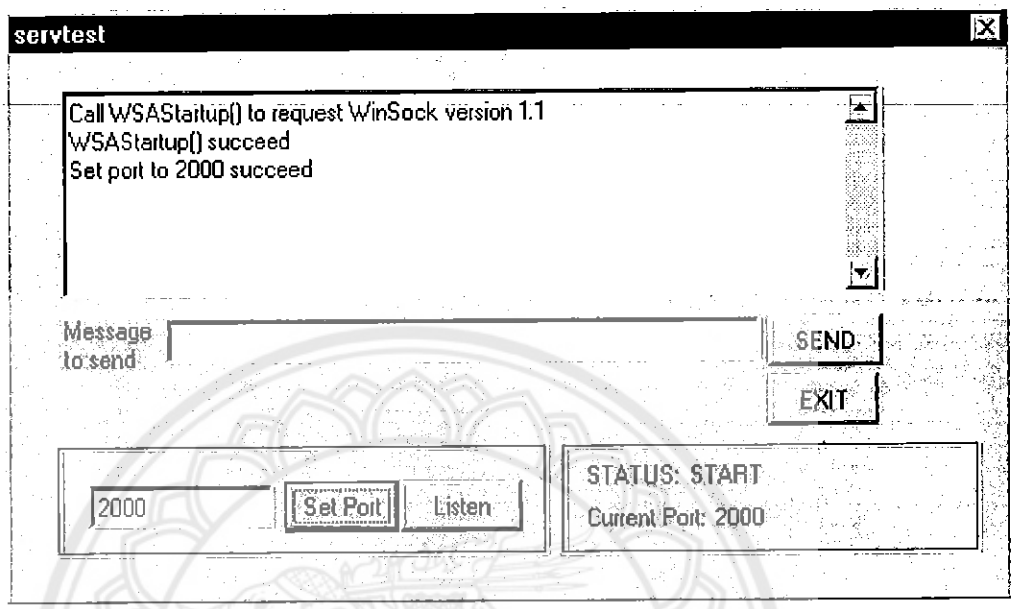
ซึ่งมีขั้นตอนการทดลองดังนี้

- เรียกโปรแกรม server ให้ทำงาน



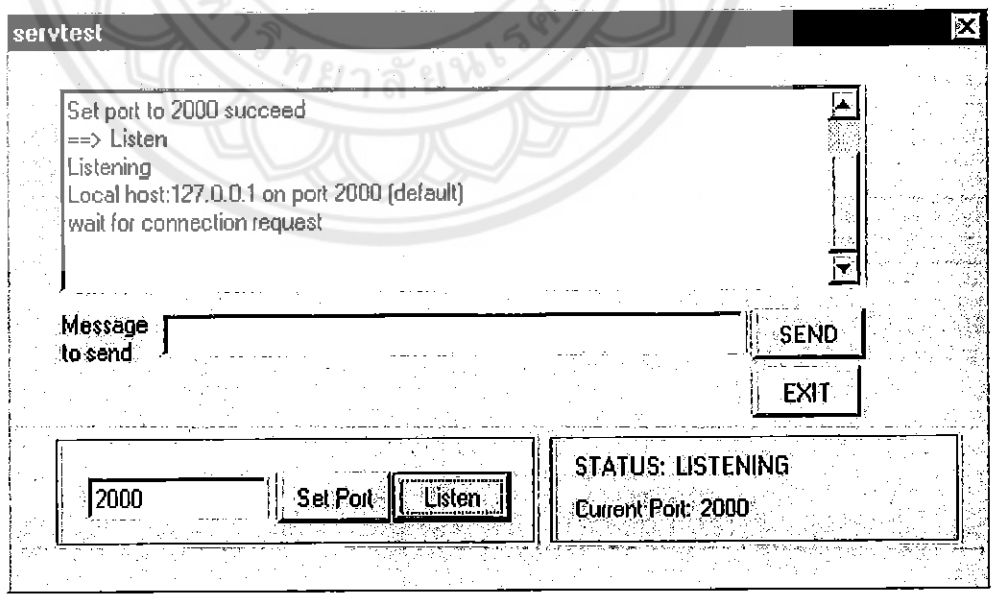
รูปที่ 4.1 โปรแกรม server (servtest)

- ระบุ port ที่จะให้ server ทำการ listen (รอรับการติดต่อ) ซึ่งในที่นี้คือ 2000



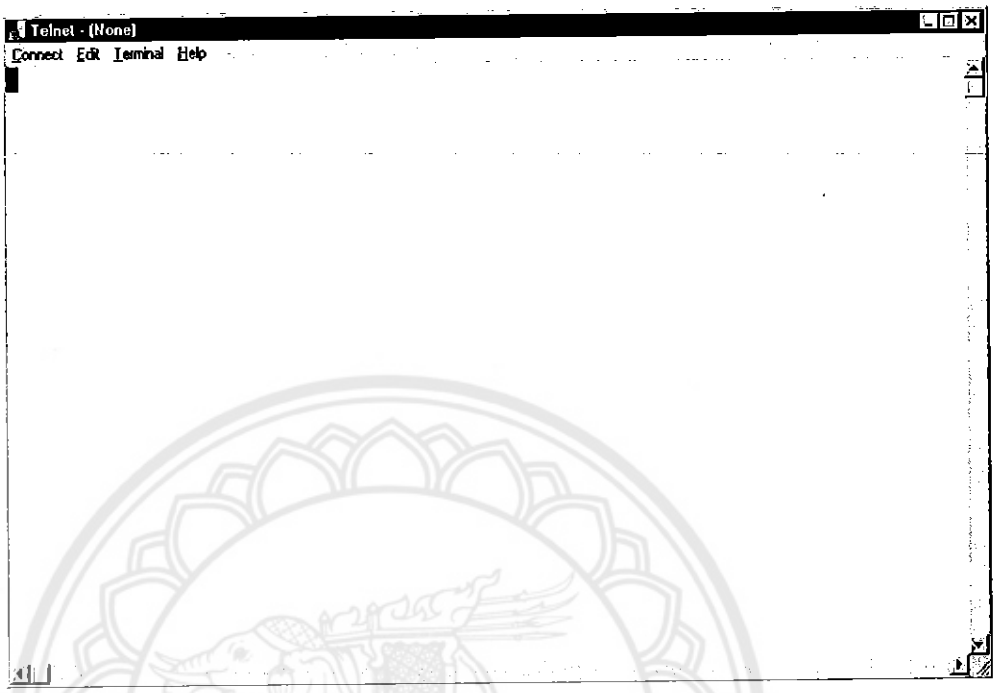
รูปที่ 4.2 ทำการ set port ให้กับโปรแกรม server

- สั่งให้ server ทำการ listen



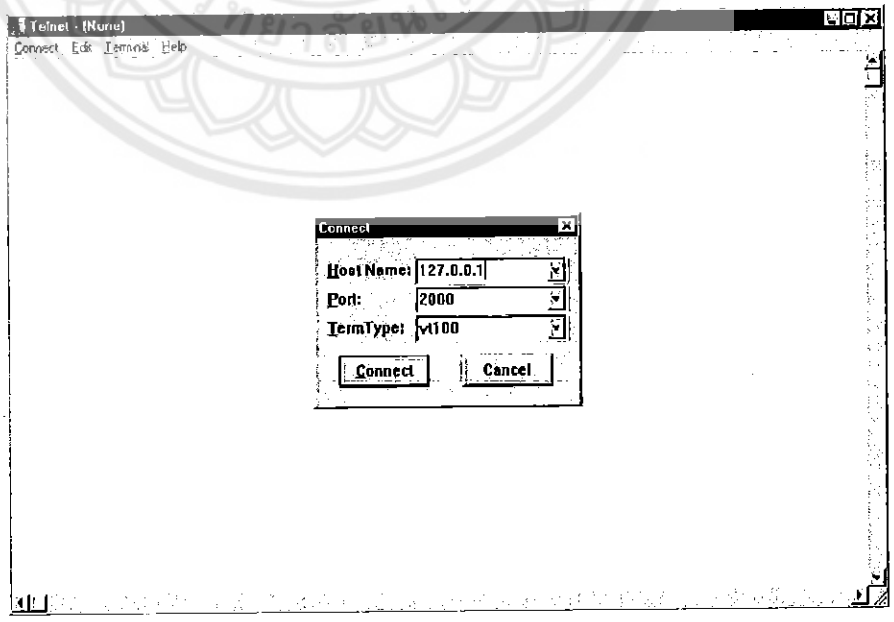
รูปที่ 4.3 server ทำการ listen

- เรียกโปรแกรม telnetให้ทำงาน (ไปที่ เมนู run แล้วพิมพ์คำว่า telnet แล้ว enter)



รูปที่ 4.4 โปรแกรม telnet

- ระบุ port ที่จะให้ telnet ติดต่อไปยัง server
- สั่งให้ telnet ติดต่อไปยัง server ที่ address 127.0.0.1 ใน port ที่ได้กำหนดไว้



รูปที่ 4.5 ระบุ address และ port ที่ใช้ติดต่อไปยัง server

- สังเกตและบันทึกผลการทดลอง

การทดลองที่ 2

ในการทดลองนี้ เรายังคงใช้โปรแกรมเดิมทั้ง server และ client โดยทั้ง server และ client จะทำงานอยู่ต่างเครื่องกัน (IP address ต่างกัน) เราทดลองโดยใช้เครื่องที่อยู่ภายในเครือข่ายเดียวกัน (อยู่ในวง LAN เดียวกัน) เพื่อให้สะดวกแก่การบันทึกผล ซึ่งมีขั้นตอนการทดลองดังนี้

- เรียกโปรแกรม server ให้ทำงาน
- ระบุ port ที่จะรอรับการติดต่อ
- สั่งให้ server ทำการ listen
- เรียกโปรแกรม client (telnet) ให้ทำงาน จากอีกเครื่องหนึ่ง
- ระบุ port ที่จะให้ client ติดต่อไปยัง server
- สั่งให้ client ติดต่อไปยัง address ของ server
- สังเกตและบันทึกผลการทดลอง

หมายเหตุ เนื่องจากรูปในการทดลองที่ 2 มีความคล้ายคลึงกับการทดลองที่ 1 จึงไม่ได้นำมาแสดงในการทดลองที่ 2 อีก

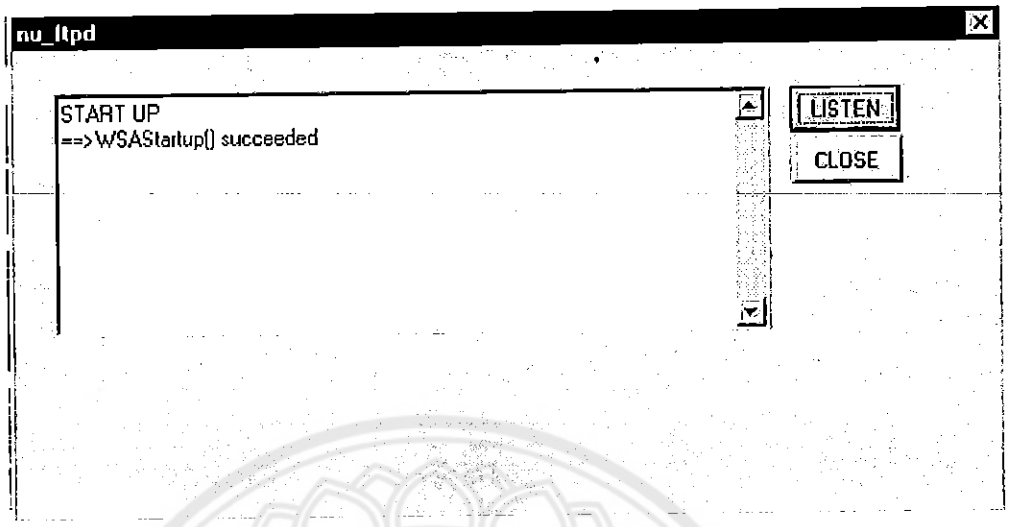
การทดลองที่ 3

ทดสอบการเชื่อมต่อโดยใช้ FTP client ที่ทำงานในระบบปฏิบัติการ DOS

เพื่อทดสอบการรับส่งข้อมูล (FTP command และ FTP reply)

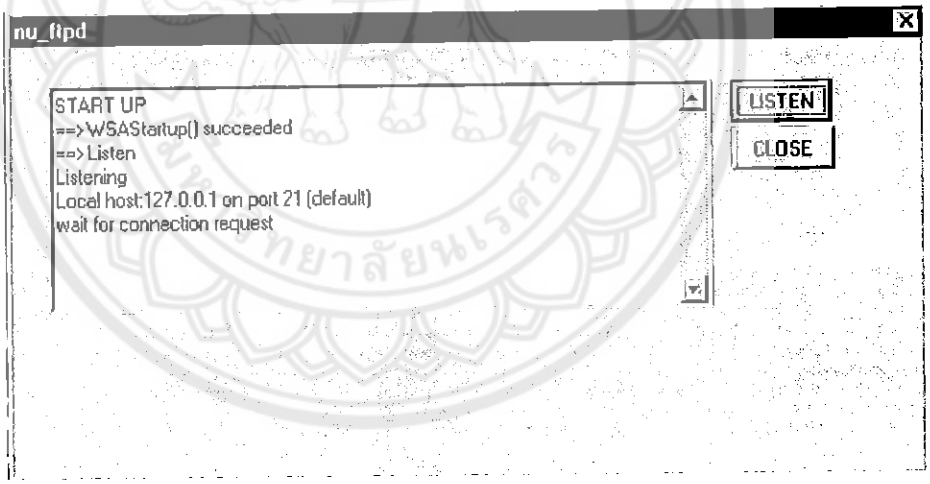
ในการทดลองนี้ เราได้ทำการพัฒนาโปรแกรม FTP server โดยใช้โครงร่างบางส่วนจากโปรแกรม server ในการทดลองที่ผ่านมา แต่ port ที่ใช้ในการติดต่อนั้น จะต้องเป็น port FTP นั่นคือ port 21 สำหรับ client ที่ใช้ นั้น เรา ใช้โปรแกรม FTP client ที่ทำงานภายใต้ระบบปฏิบัติการ DOS ซึ่งมีมาพร้อมกับ Windows 95/98 ทุกเครื่องอยู่แล้ว (สามารถเรียกใช้ได้โดย ไปที่ เมนู คำสั่ง Run พิมพ์คำว่า ftp แล้ว enter)

- เรียกโปรแกรม FTP server ให้ทำงาน



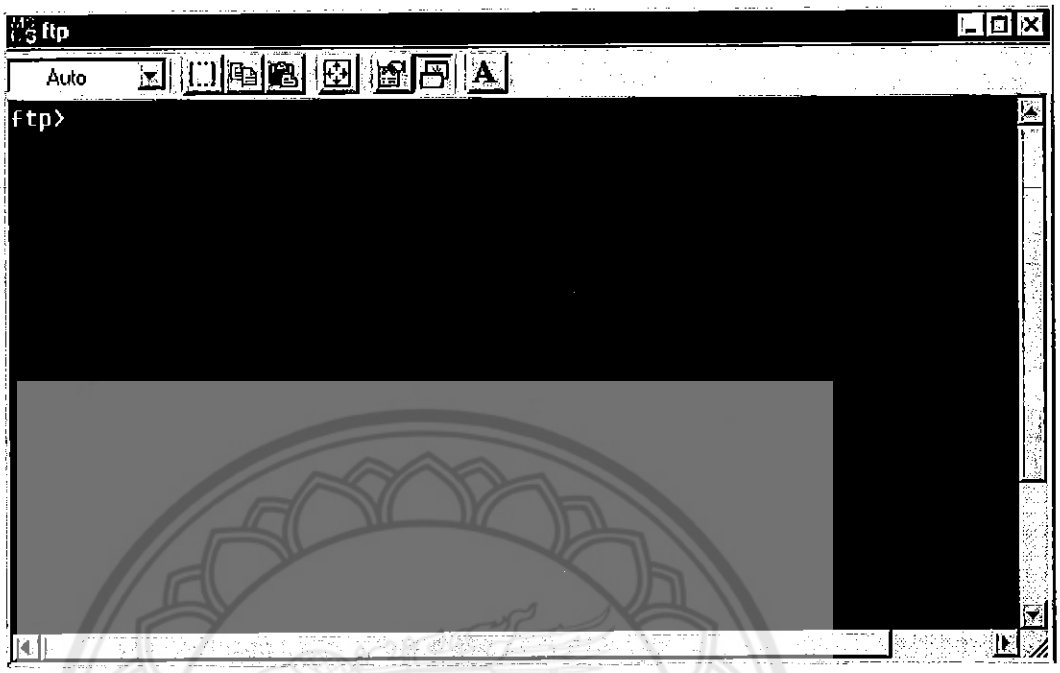
รูปที่ 4.6 โปรแกรม FTP server (nu_ftpd)

- สั่งให้ server ทำการ listen



รูปที่ 4.7โปรแกรม FTP server ทำการ listen

- เรียกโปรแกรม FTP client ให้ทำงาน



รูปที่4.8 โปรแกรม FTP client

- สั่งให้ client ติดต่อไปยัง address ของ server (สามารถเรียกชื่อ host name ของ เครื่องที่เป็น server ได้ ในที่นี้คือ default)

รูปที่4.9 โปรแกรม FTP client ทำการ connect ไปยัง server

- ทดลองส่งคำสั่ง (FTP command) ไปยัง server ในที่นี้จะทดลองส่งคำสั่ง USER ,PASS,PWD,CWD,STOR และ RETR
- สังเกตและบันทึกผลการทดลอง

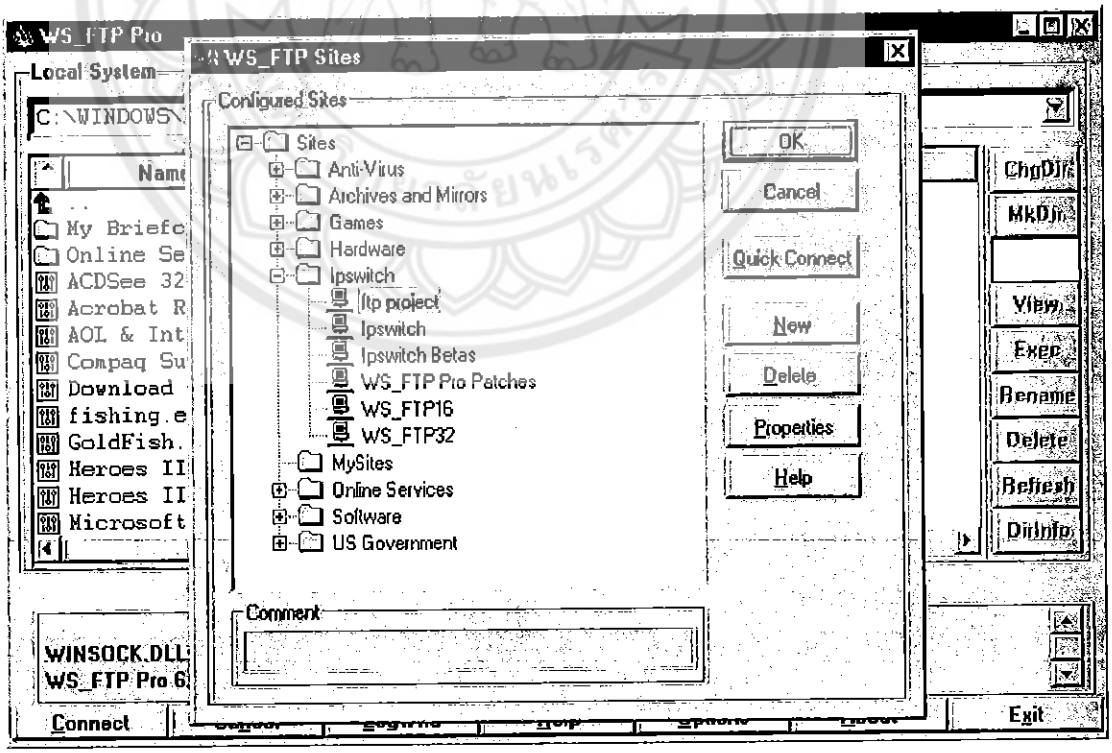
การทดลองที่ 4

ทดสอบการเชื่อมต่อโดยใช้ FTP client ที่ทำงานในระบบปฏิบัติการ Windows

เพื่อทดสอบการรับส่งข้อมูล (FTP command และ FTP reply)

ในการทดลองนี้เราใช้โปรแกรม FTP server โปรแกรมเดิม แต่เปลี่ยนโปรแกรม FTP client มาใช้โปรแกรม WS_FTP version 6.0 แทน เหตุผลที่เลือกใช้โปรแกรมนี้ เนื่องจากเป็น FTP client ที่รองรับมาตรฐานได้ดีและสามารถแสดงผลการกระทำต่าง ๆ ได้ละเอียด เหมาะแก่การสังเกตผลการทดลอง โดยมีขั้นตอนการทดลองดังนี้

- เรียกโปรแกรม FTP server ให้ทำงาน
- สั่งให้ server ทำการ listen (port ที่ทำการ listen จะเป็น port 21 อ้างอิงตาม RFC 959)
- เรียกโปรแกรม FTP client ให้ทำงาน



รูปที่4.10 การเรียกโปรแกรมClientมาใช้

- สั่งให้ FTP client ติดต่อไปยัง IP address ของ FTP server
- ทดลองส่งคำสั่ง (FTP command) ไปยัง server ในที่นี้จะทดลองส่งคำสั่ง USER ,PASS,PWD,CWD,STOR และ RETR
- สังเกตและบันทึกผลการทดลอง

การทดลองที่ 5

ทดสอบการรับส่งไฟล์

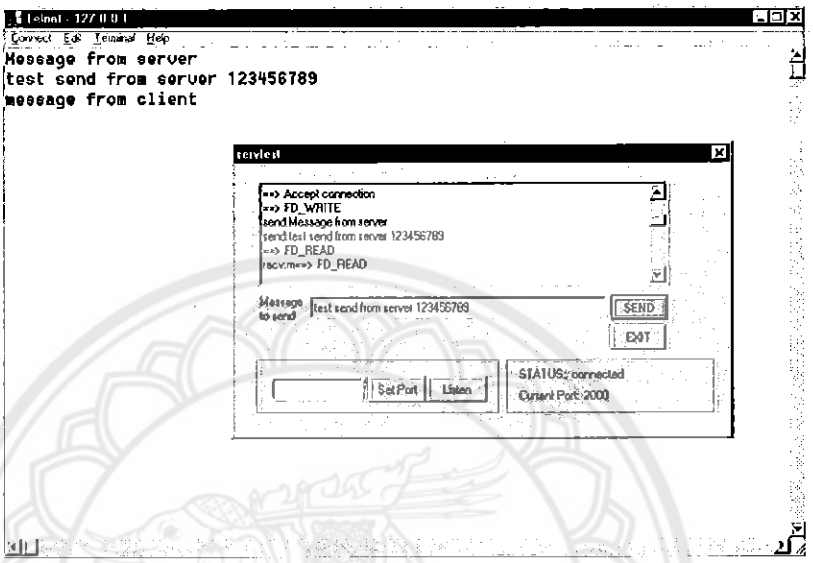
จากการทดลองที่ 4 ทำให้เราได้ทราบว่า โปรแกรม FTP server ของเราสามารถรองรับการเชื่อมต่อจาก FTP client มาตรฐานได้ ในการทดลองนี้เราจะทำการทดสอบการตอบสนองต่อ FTP command หลาย ๆ รูปแบบ รวมทั้งคำสั่งที่ต้องมีการเปิด Data connection คือ LIST , RETR และ STOR เพื่อทำการถ่ายโอนแฟ้มข้อมูล ซึ่งขั้นตอนการทดลองมีดังนี้

- เรียกโปรแกรม FTP server ให้ทำงาน
- สั่งให้ server ทำการ listen ที่ port 21
- เรียกโปรแกรม FTP client ให้ทำงาน
- สั่งให้ FTP client ติดต่อไปยัง IP address ของ FTP server
- ให้ client ส่ง FTP command ต่างๆ เพื่อดูผลการตอบสนอง
- ส่ง FTP command LIST เพื่อแสดงผล directory ใน server
- ส่ง FTP command RETR เพื่อขอ download ไฟล์ที่ต้องการจาก server
- ส่ง FTP command STOR เพื่อขอ upload ไฟล์ไปเก็บไว้ใน server
- สังเกตและบันทึกผลการทดลอง

4.3 ผลการทดลอง

การทดลองที่ 1

ทดลองการเชื่อมต่อระหว่าง server กับ client ภายในเครื่องเดียวกัน (loop back address) ทำการทดสอบโดยใช้เหตุการณ์ต่างๆกัน แล้วสังเกตผลที่เกิดขึ้น



รูปที่ 4.11 เมื่อ server และ client เชื่อมต่อกันเรียบร้อยแล้ว จะสามารถส่งข้อมูลถึงกันได้

ตารางที่ 4.1 ผลการทดลองการเชื่อมต่อระหว่าง server กับ client ภายในเครื่องเดียวกัน (loop back address)

ลำดับ	เหตุการณ์	ผลการทดลอง
1	ให้ server ทำการ listen ที่ 127.0.0.1 port 2000 แล้วให้ client ทำการ connect ที่ 127.0.0.1 port 2000	เกิด event FD_ACCEPT ขึ้นที่ server ทำให้สามารถรับส่งข้อมูลระหว่างกันได้(เกิด event FD_READและ FD_WRITE)
2	ให้ client ทำการ connect ที่127.0.0.1 port 2000 โดย server ไม่ทำการ listen	client ไม่สามารถ connect ได้สำเร็จ
3	ให้ client ทำการ connect ที่ 127.0.0.1 port 3000 ในขณะที่ server listen ที่ 127.0.0.1 port 2000	client ไม่สามารถ connect ได้สำเร็จ
4	ให้client connect ที่ port เดียวกับ server (2000) แต่ใส่ address ที่ไม่ใช่ 127.0.0.1	client ไม่สามารถ connect ได้สำเร็จ

การทดลองที่ 2

การทดลองเชื่อมต่อระหว่าง server กับ client ต่างเครื่องกัน

ในการทดลองนี้ เราใช้ระบบเครือข่ายแบบ LAN ซึ่งเครื่อง server จะมี IP address เป็น 192.168.0.2 และ client จะมี IP address เป็น 192.168.0.3

ทำการทดสอบโดยใช้เหตุการณ์ต่างๆกัน แล้วสังเกตผลที่เกิดขึ้น

ตารางที่ 4.2 การทดลองเชื่อมต่อระหว่าง server กับ client ต่างเครื่องกัน

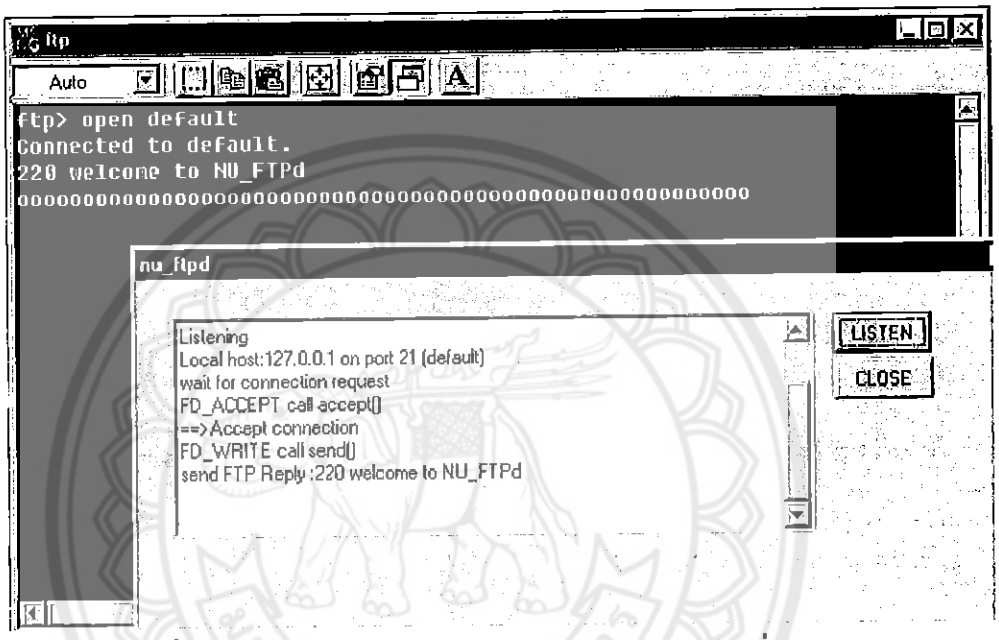
ลำดับ	เหตุการณ์	ผลการทดลอง
1	ให้ server ทำการ listen ที่ 192.168.0.2 port 2000 แล้วให้ client ทำการ connect ที่ 192.168.0.2 port 2000	เกิด event FD_ACCEPT ขึ้นที่ server ทำให้สามารถรับส่งข้อมูลระหว่างกันได้(เกิด event FD_READและ FD_WRITE)
2	ให้ client ทำการ connect ที่192.168.0.2 port 2000 โดย server ไม่ทำการ listen	client ไม่สามารถ connect ได้สำเร็จ
3	ให้ client ทำการ connect ที่ 192.168.0.2 port 3000 ในขณะที่ server listen ที่ 192.168.0.2 port 2000	client ไม่สามารถ connect ได้สำเร็จ
4	ให้client connect ที่ port เดียวกับ server (2000) แต่ใส่ addressที่ไม่ใช่192.168.0.2	client ไม่สามารถ connect ได้สำเร็จ

การทดลองที่ 3

ทดลองการเชื่อมต่อโดยใช้ FTP client ที่ทำงานในระบบปฏิบัติการ DOS

ทำการทดสอบโดยใช้เหตุการณ์ต่างๆกัน แล้วสังเกตผลที่เกิดขึ้น

ในการทดลองนี้เราใช้ระบบเครือข่ายแบบ LAN ซึ่งเครื่อง server จะมี IP address เป็น 192.168.0.2 และ client จะมี IP address เป็น 192.168.0.3 โดยใช้ port ของ FTP คือ port 21



รูปที่4.12 ใช้โปรแกรม FTP client ใน DOS connect ไปที่ FTP server

ตารางที่ 4.3 ทดลองการเชื่อมต่อโดยใช้ FTP client ที่ทำงานในระบบปฏิบัติการ DOS

ลำดับ	เหตุการณ์	ผลการทดลอง
1	ให้ server ทำการ listen ที่ 192.168.0.2แล้วให้ client ทำการ connect ที่ 192.168.0.2	เกิด event FD_ACCEPT ขึ้นที่ server แล้วทำให้ server ส่ง FTP reply ว่า " 220 welcome\r\n" ผลลัพธ์คือ ที่เครื่อง client แสดงผลของการ reply ไม่ถูกต้องคือ แสดงเป็น "220 welcome 000000000000000000" และค้างอยู่อย่างนั้น
2	หลังจากเหตุการณ์ในลำดับที่ 1 แล้ว สั่งให้ server ยกเลิกการ listen (ปิดโปรแกรม server)	ที่เครื่อง client จะแสดงข้อความ "Connection closed by remote host"
3	ให้ client ทำการ connect ที่ 192.168.0.2 โดยที่ server ไม่ได้ทำการ listen	Client ไม่สามารถ connect ได้สำเร็จ โดยขึ้นข้อความ "ftp>:connect:10061 แล้วกลับมาอยู่ที่ ftp>(ftp prompt เหมือนเดิม)

การทดลองที่ 4

ทดลองการเชื่อมต่อโดยใช้ FTP client ที่ทำงานในระบบปฏิบัติการ Windows 95/98

ทำการทดสอบโดยใช้เหตุการณ์ต่างๆกัน แล้วสังเกตผลที่เกิดขึ้น

ในการทดลองนี้เราใช้ระบบเครือข่ายแบบ LAN ซึ่งเครื่อง server จะมี IP address เป็น 192.168.0.2 และ client จะมี IP address เป็น 192.168.0.3 โดยใช้ port ของ FTP คือ port 21

ตารางที่ 4.4 ผลการทดลองการเชื่อมต่อโดยใช้ FTP Client ที่ทำงานในระบบปฏิบัติการ Windows 95/98

ลำดับ	เหตุการณ์	ผลการทดลอง
1	ให้ server ทำการ listen ที่ 192.168.0.2 แล้วให้ client ทำการ connect ที่ 192.168.0.2	เกิด event FD_ACCEPT ขึ้นที่ server แล้วให้ server ส่ง FTP reply ว่า "220 welcome\r\n" ผลลัพธ์คือ ที่เครื่อง client แสดงผลของการ reply ได้ถูกต้อง
2	หลังจากเหตุการณ์ในลำดับที่ 1 แล้ว client จะทำการส่ง FTP command (USER anonymous)	ที่ server เกิด event FD_READ ให้ server รับ FTP command มาวิเคราะห์คำสั่ง และ argument ที่ได้รับมา ซึ่ง server ทำการ แสดง ผลสิ่งที่รับมาได้ถูกต้อง จากนั้นทำการส่ง FTP Reply กลับไปตามสถานะที่เป็นอยู่ขณะนั้น
3	Client ส่ง FTP command ที่ server ไม่รู้จัก	Server ส่ง reply กลับไปว่า error
4	Client ปิดการเชื่อมต่อ	เกิด event FD_CLOSE ที่ server

การทดลองที่ 5

ทดลองการรับส่งไฟล์

ทำการทดลองโดยใช้เหตุการณ์ต่างๆกัน แล้วสังเกตผลที่เกิดขึ้น

ในการทดลองนี้เราใช้ระบบเครือข่ายแบบ LAN ซึ่งเครื่อง server จะมี IP address เป็น 192.168.0.2 และ client จะมี IP address เป็น 192.168.0.3 โดยใช้ port ของ FTP คือ port 21 ทดลองส่งคำสั่งต่าง ๆ ไปตามลำดับดังนี้

ตารางที่ 4.5 ผลการทดลองการรับส่งไฟล์ทำการทดสอบโดยใช้เหตุการณ์ต่าง ๆ กัน แล้วสังเกตผลที่เกิดขึ้น

ลำดับ	Client	Server
1	-	listen ที่ 192.168.0.2
2	connect ไปที่ 192.168.0.2	เกิด event FD_ACCEPT และ FD_WRITE ยอมรับการเชื่อมต่อ และส่ง FTP reply กลับไปว่า 220 welcome to NU_FTPd
3	ส่ง FTP command USER anonymous	เกิด event FD_READ รับ FTP command และ argument มา พิจารณาแล้ว ส่ง FTP reply กลับไปว่า 331 USER anonymous login password required
4	ส่ง FTP command PASS wsftp60@	เกิด event FD_READ รับ command มาพิจารณาแล้วส่ง reply กลับไปว่า 230 login success
5	ส่ง FTP command PWD	เกิด event FD_READ รับ command มาพิจารณาแล้วส่ง reply กลับไปว่า 257 A:\NU_FTPd\lcc is working directory (ผลที่ได้นั้นแล้วแต่ว่า FTP server กำลังทำงานอยู่ที่ directory ไหน)
6	ส่ง FTP command PORT192,168,0,3,4,3	เกิด event FD_READ รับ command มาพิจารณา แล้วทำการเปิด data connection ที่ address ซึ่ง client กำหนดมาโดยคำสั่ง PORT เกิด event FD_CONNECT และ FD_WRITE ใน data connection แล้วส่ง reply กลับไปว่า 200 PORT command successfull
7	ส่ง FTP command LIST	เกิด event FD_READ รับ command มาพิจารณาแล้ว ไม่สามารถปฏิบัติตามได้สำเร็จจึงส่ง reply ว่า 500 under construction
8	ได้รับแจ้งว่าเกิด error	
9	ปิดการเชื่อมต่อ	เกิด event FD_CLOSE

4.4 วิเคราะห์ผลการทดลอง

การทดลองที่ 1

จากผลการทดลองพบว่าวินโดวส์ซ็อกเก็ตสามารถรับส่งข้อมูลระหว่างโปรแกรมได้

การทดลองที่ 2

จากผลการทดลองที่ 1 และ ผลการทดลองนี้ ทำให้สรุปได้ว่า วินโดวส์ซ็อกเก็ตสามารถเชื่อมต่อเพื่อรับส่งข้อมูลระหว่างโปรแกรมได้ ไม่ว่าโปรแกรมทั้งคู่จะมี IP address เหมือนกัน หรือต่างกัน เพียงแต่ในการ connect จะต้องทำการระบุ IP address และ port ให้ถูกต้อง และ ถ้า server ไม่ทำการ listen แล้ว client จะไม่สามารถทำการ connect ได้สำเร็จ

การทดลองที่ 3

จากผลการทดลองพบว่า โปรแกรม FTP ที่เราพัฒนาขึ้นไม่สามารถรองรับการเชื่อมต่อจาก FTP client ที่ทำงานในระบบปฏิบัติการ DOS ได้ จากการวิเคราะห์คาดว่า สาเหตุอาจอยู่ที่รูปแบบของอักขระปิดท้าย FTP reply นั้นต่างกัน ทำให้ FTP client ไม่เข้าใจ FTP reply ที่server ของเราส่งไป

การทดลองที่ 4

จากผลการทดลองพบว่า โปรแกรม FTP client (ในที่นี้คือ WS_FTP version 6.0) สามารถรับและแสดงผล FTP reply ที่server ส่งไปให้ได้ถูกต้อง และ server สามารถรับ FTP command พร้อมทั้ง argument จาก FTP client ได้ถูกต้อง

การทดลองที่ 5

จากผลการทดลองพบว่า โปรแกรม FTP server สามารถรับและตอบสนองต่าง ๆ ได้เป็นบางคำสั่งได้แก่ USER , PASS , SYST , PWD, CWD , PORT ส่วนคำสั่งอื่น ๆ นั้น ยังไม่สามารถตอบสนองได้ เนื่องจากเราพบปัญหาในการตอบสนองต่อคำสั่ง LIST เพราะไม่สามารถส่งข้อมูลภายใน directory ที่ทำงานอยู่ไปยัง server ได้ เนื่องจาก ไม่มีข้อมูลเพียงพอในการพัฒนาโปรแกรมในส่วนนี้ ดังนั้นจึงเกิดปัญหาในทางปฏิบัติที่ผู้ใช้ทางฝั่ง client จะไม่สามารถรู้ได้ว่ามีข้อมูลอะไรอยู่ใน directory ของ server บ้าง เพราะไม่สามารถใช้คำสั่ง LIST ได้ แต่จากการวิเคราะห์เมื่อ server ได้รับ คำสั่ง PORT แล้ว server สามารถทำการ connect เพื่อเปิด Data connection ได้ แสดง

ว่า เราสามารถรับ-ส่ง ข้อมูลได้ แต่เนื่องจากปัญหากับคำสั่ง LIST และ เวลาอันจำกัด เราจึงไม่สามารถพัฒนาโปรแกรมในส่วนนี้ได้สมบูรณ์

บทที่ 5
บทสรุป

หลังจากได้ทำการทดสอบและพัฒนาโปรแกรม FTP server แล้ว เราได้ข้อสรุปต่าง ๆ ดังต่อไปนี้

5.1 สรุปผลการทดลอง

ความสามารถและข้อจำกัดของโปรแกรม FTP server ที่ ได้พัฒนาขึ้น

- สามารถรับการเชื่อมต่อจาก FTP client ที่ทำงานใน windows 95/98 ได้ แต่มีปัญหากับการเชื่อมต่อจาก client ที่ทำงานใน DOS
- สามารถรับ-ส่ง FTP command และ FTP reply ผ่านทาง control connection (port 21) ได้ โดยสามารถวิเคราะห์ command และ ตอบสนองตามสถานะได้ แต่ยังไม่สามารถปฏิบัติตามคำสั่งบางคำสั่งได้ เช่น LIST ,STOR,RETR
- สามารถเปิด data connection ได้ แต่ยังไม่สามารถถ่ายโอนแฟ้มข้อมูลได้ เนื่องจากปัญหาดังที่กล่าวไว้ในหัวข้อ 4.4 การทดลองที่ 5

5.2 ปัญหาที่เกิดขึ้น

- ยังไม่สามารถรับการเชื่อมต่อจากโปรแกรม FTP client ที่ทำงานในระบบปฏิบัติการ DOS ได้
- ยังไม่รองรับ FTP command ครบทุกคำสั่ง รวมทั้งยังไม่สามารถพัฒนาการถ่ายโอนข้อมูลแบบ binary
- ยังไม่รองรับ feature ที่อยู่นอกเหนือ RFC 959 เช่น การ resume หรือ การแยก connection ออกเป็นหลาย ๆ ส่วน
- การจัดการระบบ USER รวมทั้งสิทธิในการเข้าถึง และ Directory ยังไม่สมบูรณ์
- ยังไม่มีระบบความปลอดภัยที่ดี

5.3 แนวทางในการแก้ไขปัญหา

- ศึกษาเพิ่มเติมเกี่ยวกับรูปแบบของ command ใน telnet (เพราะ FTP ใช้รูปแบบของ command แบบเดียวกับ telnet) เพื่อให้สามารถรองรับการเชื่อมต่อจาก client ได้หลาย ๆ รูปแบบ
- เพิ่มเติมคำสั่งที่ยังไม่ได้พัฒนา
- ศึกษาเพิ่มเติมเกี่ยวกับ feature ใหม่ ๆ ที่อยู่นอกเหนือมาตรฐาน RFC 959 และนำมาพัฒนาเพิ่มเติม
- ศึกษาเพิ่มเติมและพัฒนาเกี่ยวกับ ระบบจัดการ USER และ Directory รวมทั้งเรื่องของการเข้ารหัสและความปลอดภัย

5.4 ข้อเสนอแนะเพิ่มเติม

- สามารถนำความรู้เกี่ยวกับ Windows socket ไปประยุกต์ใช้ในการเขียนโปรแกรมระบบเครือข่ายอื่น ๆ เช่น การรับส่งข้อความ เกมส์ที่เล่นผ่านระบบเครือข่าย และอื่น ๆ ได้
- นำโปรแกรม FTP server ที่ได้ ไปพัฒนาเพิ่มเติมให้กลายเป็นโปรแกรมที่รองรับมาตรฐานได้มากขึ้น และ สามารถนำไปใช้เป็นโปรแกรมต้นแบบเพื่อการศึกษาได้

5.5 ข้อดีและข้อเสียในการทดลอง

ข้อดี

- ทำให้เกิดความรู้ความเข้าใจเกี่ยวกับระบบเครือข่ายและการพัฒนาโปรแกรมในระบบเครือข่าย
- ทำให้เกิดตัวอย่างในการศึกษาเพื่อนำไปพัฒนาระบบที่ดีกว่าเดิมต่อไป
- พัฒนาจากภาษาซี ซึ่งต้องเขียนทุกอย่างขึ้นเอง ทำให้สามารถมองเห็นกลไกการทำงานภายในได้ จึงเกิดความเข้าใจและแนวคิดในการพัฒนาโปรแกรมในระบบเครือข่ายได้ลึกซึ้งกว่าการใช้อุปกรณ์สำเร็จรูป

ข้อเสีย

- เป็นระบบที่มีอยู่แล้ว (ไม่ใช่สิ่งใหม่)
- เป็นการพัฒนาเพื่อการศึกษา ดังนั้นจึงพัฒนาโดยใช้ภาษาซี ซึ่งต้องเขียนทุกอย่างเองเพื่อให้เกิดความเข้าใจ แต่ก็เป็นการช้าและยุ่งยากซับซ้อน ในการพัฒนางานจริง ๆ

เนื่องจากในปัจจุบันมีอุปกรณ์สำเร็จรูป(component) ให้ใช้อยู่มากมาย ซึ่งจะพัฒนาได้ง่ายและรวดเร็วกว่านี้ แต่อาจจะไม่เข้าใจกลไกการทำงานภายในอย่างแท้จริง

5.6 แนวทางในการพัฒนาต่อไป

- ปรับปรุงให้เป็นระบบที่มีประสิทธิภาพมากยิ่งขึ้น เช่น การจัดการกับ directory จัดการกับ USER รวมทั้งสิทธิในการใช้งาน และ ระบบความปลอดภัย
- นำไปประยุกต์เป็นระบบที่ต่างออกไปจากเดิม โดยใช้พื้นฐานในการติดต่อและรับส่งข้อมูลระหว่างกัน ซึ่งจะทำให้ได้โปรแกรมต่าง ๆ หลากหลาย

บรรณานุกรม

1. นิรุธ อำนวยศิลป์ คู่มือการเขียนโปรแกรมMicrosoft Visual C++ Version6.0 ฉบับเพื่อการใช้งานจริง พิมพ์ครั้งที่ 2 , กรุงเทพฯ : บริษัท ส. เอเซียเพรส(1989)จำกัด : 1998
2. สัจจะ จรัสรุ่งรวีวร Internet Programming ด้วย Visual Basic 6.0 และ ASP พิมพ์ครั้งที่ 1 นนทบุรี:
   สำนักพิมพ์ อินโฟเพรส : 2542
3. สุวัฒน์ ปุณณชัยยะ , ต้น ตัณฑ์สุทธิวงศ์ , สุพจน์ ปุณณชัยยะ เปิดโลกของ TCP/IP และโปรโตคอลของอินเตอร์เน็ต พิมพ์ครั้งที่ 1 กรุงเทพฯ : บริษัท โปรวิชั่น จำกัด : 2543
4. Andrew S. Tanenbaum , สัลยุทธ์ สว่างวรรณ แปล Computer Networks : เครือข่ายคอมพิวเตอร์ พิมพ์ครั้งที่ 1 กรุงเทพฯ : บริษัท เพียร์สัน เอ็ดดูเคชั่น อินโดไชน่า จำกัด : 2543
5. BOB QUIN , DAVE SHUTE Windows Sockets Network Programming 1st EDITION , ADDI-SON-WESLEY PUBLISHING COMPANY, INC AMERICA : 1995
6. DOUGLAS E. COMER , DAVID L. STEVENS Internetworking With TCP/IP Vol III : Client-Server Programming And Application BSD Socket Version Fourth Printing, PRENTICE-HALL OF INDIA PRIVATE LIMITED , NEW DELHI : 1998

ภาคผนวก

ตัวโปรแกรม

(Sourcecode)

```
/*@@ Wedit generated application. Written Thu Sep 21 09:18:01 2000
 @@header: c:\programming\nu_ftpd\nu_ftpdres.h
 @@resources: c:\programming\nu_ftpd\nu_ftpd.rc
 Do not edit outside the indicated areas */
/*<------------------------------------------------------------------------>*/
/*<------------------------------------------------------------------------>*/
#include <windows.h>
#include <windowsx.h>
#include <commctrl.h>
#include <string.h>
#include "nu_ftpdres.h"

#include <io.h>
#include <direct.h>
#include <winsock.h>
#include <time.h>
#include <stdio.h>
//#include <wininet.h>

#define Display(s) SendDlgItemMessage(hwndDlg,IDT_DISPLAY, \
                                      EM_REPLACESEL,0,(LPARAM)
((LPSTR)s))
```

```
#define WSA_ASYNC WM_USER+1
#define MAXHOSTNAME 256
#define MAXFILENAME 256

/* define state of application */
#define START 1
#define LISTEN 2
#define LOGIN 3
#define CONNECTED 4
#define PROCESS 5
#define INITDATACONN 6
#define DATACONNECTED 7

/* --- The following code comes from C:\lcc\lib\wizard\dlgbased.tpl. */
/*<-----------------------------------------------------------------------*/

/*
Template for a dialog based application. The main procedure for this
template is the DialogFunc below. Modify it to suit your needs.
*/

/* Global variable */
HINSTANCE hInst;  //instance handle
HWND hWinMain;   // main window(dialog) handle
unsigned int iLstnPort;    //listen port
unsigned int iDataPort; //Data port
int STATE;
SOCKET hCtrlSock=INVALID_SOCKET;        // FTP control socket
SOCKET hLstnSock=INVALID_SOCKET;        //listening control socket
SOCKET hDataSock=INVALID_SOCKET;        //FTP data socket
```

```
SOCKADDR_IN stCLclName;   /* control socket name (local client) */
SOCKADDR_IN stCRmtName;  /* control soket  name (remote server) */
SOCKADDR_IN stDLclName;  /* data socket name (local client) */
SOCKADDR_IN stDRmtName;  /* data socket name (remote server) */

char szRmtHost[MAXHOSTNAME];
char szHostName[MAXHOSTNAME];   //ตัวแปรเก็บค่า HostName
char szBuf[MAXHOSTNAME]; // buffer
char szFtpCmd[80];   //buffer สำหรับรับ FTP Command
char szFtpReply[80]; // buffer สำหรับ FTP Reply
char szUser[10];//ตัวแปรเก็บ ชื่อ user ที่ login เข้ามา
char szPath[MAXHOSTNAME]; //ตัวแปรเก็บ path ที่ทำงาน
char szTempFile[256];
/* FTP Command structure */
struct Cmd{
        char szCmd[5];  //FTP Command
        char szArg[80]; //FTP Command argument
};
typedef struct Cmd Cmd;
Cmd FtpCmd;


/* prototype for the dialog box function. */
static BOOL CALLBACK DialogFunc(HWND hwndDlg, UINT msg, WPARAM wParam,
LPARAM lParam);
/*  Function Prototype */
void SolveDataPort();
BOOL InitLstnSock(int iLstnPort,PSOCKADDR_IN pstSockName,HWND hWnd,u_int
nAsyncMsg);
LONG GetHostID();
```

```
SOCKET InitDataConn(PSOCKADDR_IN pstName,HWND hwndDlg,u_int nAsyncMsg);
SOCKET AcceptCtrlConn(SOCKET hLstnSock,PSOCKADDR_IN pstName);
int RecvFtpCmd(SOCKET hCtrlSock,LPSTR szFTPCmd,int nLen);
int SendFtpReply(SOCKET hCtrlSock,LPSTR szFtpReply,int nLen);
u_long GetAddr(LPSTR szHost);
void CloseConn(SOCKET hSock);
void ChkState();
void login();
BOOL ChkPass();
void CmdProcess();
void Transfer();
BOOL GetLclDir(LPSTR szTempFile);
/*
Win main just registers a class of the same type that the dialog class, and
then calls DialogBox. Then it exits. The return value is the return value of
the dialog procedure.
*/

int APIENTRY WinMain(HINSTANCE hinst, HINSTANCE hinstPrev, LPSTR lpCmdLine, int
nCmdShow)
{
        int nRet;  //return code
        WSADATA wsaData; // WinSock DLL data structure
        WORD wVersionRequired=MAKEWORD(1,1);  //request winsock version 1.1
        MSG msg;
        hInst=hinst;

        WNDCLASS wc;

        memset(&wc,0,sizeof(wc));
```

```
wc.lpfnWndProc = DefDlgProc;
wc.cbWndExtra = DLGWINDOWEXTRA;
wc.hInstance = hinst;
wc.hCursor = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground = (HBRUSH) (COLOR_WINDOW + 1);
wc.lpszClassName = "bleftpd";
RegisterClass(&wc);

// Initialize the WinSock DLL
nRet=WSAStartup(wVersionRequired,&wsaData);
        if(nRet!=0)
        {
                MessageBox(NULL,"Error on WSAStaratup()","Test",MB_OK);
                return 1;
        }

        //Confirm that the version requested is avalible
        if(wsaData.wVersion!=wVersionRequired)
                {
                        //Version needed is not avalible
                        MessageBox(NULL,"Wrong WinSock version","error",MB_OK);
                        WSACleanup();
                        return 1;
                }
        else
                {
                STATE=START;
                DialogBox(hInst, MAKEINTRESOURCE(IDD_MAINDIALOG), NULL,
(DLGPROC) DialogFunc);
```

```
                    //release Winsock DLL
                    WSACleanup();
                        if(nRet==SOCKET_ERROR)

        MessageBox(NULL,"WSACleanup()","ERROR",MB_OK);
                    }
        return msg.wParam;
}

/*
You should add your initialization code here. This function will be called
when the dialog box receives the WM_INITDIALOG message.
*/
static int InitializeApp(HWND hDlg,WPARAM wParam, LPARAM lParam)
{
        strcpy(szPath,"C:\\public");
        chdir(szPath);
        strcpy(szTempFile,"list.tmp");
        return 1;
}

/*
This is the main function for the dialog. It handles all messages. Do what your
application needs to do here.
*/
static BOOL CALLBACK DialogFunc(HWND hwndDlg, UINT msg, WPARAM wParam,
LPARAM lParam)
{
        WORD WSAEvent;
        hWinMain=hwndDlg;
```

```
        switch (msg) {
        /* This message means the dialog is started but not yet visible.
          Do All initializations here
      */
        case WSA_ASYNC:
              /* control socket async notification message handlers */
              WSAEvent=WSAGETSELECTEVENT(lParam);
                    switch(WSAEvent)
                    {
                          case FD_READ:
                                Display("==>FD_READ call recv()\r\n");
                                RecvFtpCmd(hCtrlSock,szFtpCmd,sizeof(szFtpCmd));
                                      wsprintf(szBuf,"recv command:%s",szFtpCmd);
                                Display(szBuf);

                                      wsprintf(szBuf,"cmd=%s  arg=%s
\r\n",FtpCmd.szCmd,FtpCmd.szArg);
                                Display(szBuf);
                                            ChkState();
                                SendFtpReply(hCtrlSock,szFtpReply,sizeof(szFtpReply));
                                      wsprintf(szBuf,"send FTP Reply
:%s\r\n",szFtpReply);
                                          Display(szBuf);

                                break;
                          case FD_ACCEPT:
                                Display("FD_ACCEPT call accept()\r\n");
                                if(hLstnSock!=INVALID_SOCKET)
                                      {
```

```
                                        /* accept the incoming control connection
request */
                                        hCtrlSock=AcceptCtrlConn
(hLstnSock,&stCRmtName);
                                        if(hCtrlSock==INVALID_SOCKET)
                                                {
                                                        Display("ERROR!!
hCtrlSock=INVALID_SOCKET\r\n");
                                                }
                                        else
                                                {
                                                        Display("==>Accept
connection\r\n");
                                                        STATE=LOGIN;
                                                }
                                }
                        else
                                Display("ERROR!!
hLstnSock=INVALID_SOCKET\r\n");

                        break;
                case FD_WRITE:
                        Display("FD_WRITE call send()\r\n");
                        ChkState();
                        SendFtpReply(hCtrlSock,szFtpReply,sizeof(szFtpReply));
                                wsprintf(szBuf,"send FTP Reply
:%s\r\n",szFtpReply);
                                Display(szBuf);
                        break;
                case FD_CLOSE:
```

```
                    Display("FD_CLOSE\r\n");
                    CloseConn(hCtrlSock);
                    hCtrlSock=INVALID_SOCKET;
                    if(hCtrlSock==INVALID_SOCKET)
                        {
                            Display("Close connection\r\n");
                        }
                    wsprintf(szFtpReply,"");//clear reply
                    wsprintf(FtpCmd.szCmd,"");//clear command
                    break;
                default:
                    break;
            } //end switch(WSAEvent)
        break;
case WSA_ASYNC+1:
        /* data connection */
        WSAEvent=WSAGETSELECTEVENT(lParam);
        switch(WSAEvent)
        {
        case FD_CONNECT:
            Display("Data connection:FD_CONNECT\r\n");
            break;
        case FD_READ:
            Display("Data connection:FD_READ\r\n");
            wsprintf(szFtpCmd,"200 Port command successful\r\n");
            break;
        case FD_WRITE:
            STATE=DATACONNECTED;
            Display("Data connection:FD_WRITE\r\n");
            break;
```

```
                case FD_CLOSE:
                        Display("Data connection:FD_CLOSE\r\n");
                        break;
                default:
                                break;
                        }
                break;
        case WM_INITDIALOG:
                Display("START UP\r\n");
                Display("==>WSAStartup() succeeded \r\n");
                InitializeApp(hwndDlg,wParam,lParam);
                return TRUE;
        /* By default, IDOK means close this dialog returning 1, IDCANCEL means
      close this dialog returning zero
    */
        case WM_COMMAND:
                switch (LOWORD(wParam)) {
                        case IDB_LISTEN:
                                Display("==>Listen\r\n");
                                iLstnPort=21;

    hLstnSock=InitLstnSock(iLstnPort,&stCLclName,hwndDlg,WSA_ASYNC);
                                if(hLstnSock!=INVALID_SOCKET)
                                {
                                /* get our local info for display */
                                        STATE=LISTEN;//wait for connection request;

                                        stCLclName.sin_addr.s_addr=GetHostID();
                                        gethostname(szHostName,MAXHOSTNAME);
```

```
                    wsprintf(szBuf,"Local host:%s on port %d
(%s)\r\n",inet_ntoa(stCLclName.sin_addr),
                                htons(stCLclName.sin_port),
                                szHostName[0]?(LPSTR)szHostName:(LPSTR)"
<unknow>");
                    Display(szBuf);
                    Display("wait for connection request\r\n");
                    //MessageBox(NULL,szBuf,"Test",MB_OK);
                }
                else
                {
                    Display("Error!!INVALID_SOCKET\r\n");
                    return 1;
                }

                break;
            case IDB_CLOSE:
                CloseConn(hCtrlSock);
                Display("Close connection");
                    STATE=START;
                //EndDialog(hwndDlg,0);
                return 1;


            }
            break;
        /* By default, WM_CLOSE is equivalent to CANCEL */
        case WM_CLOSE:
```

```
                //MessageBox(hwndDlg,"Are you sure","exit",MB_YESNO);
                        EndDialog(hwndDlg,0);
                return TRUE;

        }
        return FALSE;
}

void ChkState()
{
        switch(STATE)
                {
                case START:
                        wsprintf(szFtpReply,"");
                        strcpy(szPath,"c:\\public");
                        break;
                case LISTEN:
                        break;
                case LOGIN:
                        login();
                        break;
                case CONNECTED:
                        CmdProcess();
                        break;
                case PROCESS:
                        break;
                case INITDATACONN:

                        break;
```

```
                case DATACONNECTED:
                        Transfer();
                        break;
                default:
                        break;
                }

}


BOOL ChkPass()
{
        return TRUE;
}
void login()
{

        if(strcmpi(FtpCmd.szCmd,"")==0)
                wsprintf(szFtpReply,"220 welcome to NU_FTPd\r\n");
        else if(strcmpi(FtpCmd.szCmd,"USER")==0)
                {
                strcpy(szUser,FtpCmd.szArg);
                wsprintf(szFtpReply,"331 USER %s login password required\r\n",szUser);
                STATE=LOGIN;
                }
        else if(strcmpi(FtpCmd.szCmd,"PASS")==0)
                {
                        if(ChkPass())
                                {
                                wsprintf(szFtpReply,"230 login success\r\n");
```

```
                    STATE=CONNECTED;
                    }
                else
                    {
                    wsprintf(szFtpReply,"530 sorry not allowed\r\n");
                    CloseConn(hCtrlSock);
                    STATE=START;
                    }
            }
        else
            {
                wsprintf(szFtpReply,"500 internal error login fail\r\n");
            }

}

void CmdProcess()
{
    STATE=CONNECTED;
    if(strcmpi(FtpCmd.szCmd,"PWD")==0)
        {
        MessageBox(NULL,"receive PWD","PWD",MB_OK);
        getcwd(szPath,sizeof(szPath));
        wsprintf(szFtpReply,"257 %s is working directory\r\n",szPath);
        }
    else if(strcmpi(FtpCmd.szCmd,"PORT")==0)
        {
        MessageBox(NULL,"receve PORT","PORT",MB_OK);
        SolveDataPort();
        wsprintf(szBuf,"Remote host:%s:%d",szRmtHost,iDataPort);
```

```
                MessageBox(NULL,szBuf,"",MB_OK);
                stDRmtName.sin_addr.s_addr=GetAddr(szRmtHost);
                        if(stDRmtName.sin_addr.s_addr==INADDR_ANY)
                        {
                                MessageBox(NULL,"can 't
connect","Sorry",MB_OK);
                                wsprintf(szFtpReply,"500 Port fail\r\n");
                        }
                        else
                                {
hDataSock=InitDataConn(&stDRmtName,hWinMain,WSA_ASYNC+1);
                                }
                                if(hDataSock!=INVALID_SOCKET)
                                {
                                wsprintf(szFtpReply,"200 PORT command
successfull\r\n");
                                STATE=DATACONNECTED;
                                }
                                else
                                {
                                wsprintf(szFtpReply,"500 Port fail\r\n");
                                CloseConn(hCtrlSock);
                                }

                }
        else if(strcmpi(FtpCmd.szCmd,"SYST")==0)
                {
                        MessageBox(NULL,"receve SYST","SYST",MB_OK);
                        wsprintf(szFtpReply,"215 Windows_NT version 4.0\r\n");
```

```
        }
    else if(strcmpi(FtpCmd.szCmd,"CWD")==0)
        {
            MessageBox(NULL,"receve CWD","CWD",MB_OK);
            strcpy(szBuf,FtpCmd.szArg);

                int nRet;
                nRet=chdir(FtpCmd.szArg);
                if(nRet==-1)
                    wsprintf(szFtpReply,"550 Request action not taken. File
unavailable.\r\n");
                else
                    wsprintf(szFtpReply,"250 Requested file action okay,
completed. \r\n");
        }
    else
        MessageBox(NULL,"command not implement",FtpCmd.szCmd,MB_OK);

}

void Transfer()
{
    if(strcmpi(FtpCmd.szCmd,"LIST")==0)
        {
        MessageBox(NULL,"receve LIST","LIST",MB_OK);
            BOOL bGetcwd;
            if(bGetcwd=GetLclDir(szTempFile))
                {
                    //wsprintf(szBuf,"notepad %s",szTempFile);
                    //MessageBox(NULL,szBuf,"dir list",MB_OK);
```

```
                    wsprintf(szFtpReply,"150 open ASCII mode dataconnection for
/bin/ls \r\n");
                    wsprintf(szBuf,"test.txt \n");
                        send(hDataSock,szBuf,sizeof(szBuf),0);
                    wsprintf(szFtpReply,"226 Transfer complete\r\n");
                    }
                else
                    wsprintf(szFtpReply,"500 under construction\r\n");
            }
        else if(strcmpi(FtpCmd.szCmd,"RETE")==0)
            {
            MessageBox(NULL,"receve RETE","RETE",MB_OK);
            wsprintf(szFtpReply,"500 under construction \r\n");
            }
        else if(strcmpi(FtpCmd.szCmd,"STOR")==0)
            {
            MessageBox(NULL,"receve STOR","STOR",MB_OK);
            wsprintf(szFtpReply,"500 under construction \r\n");
            }
        else
            {
                wsprintf(szFtpReply,"500 unknown command on transfer status\r\n");
            }
        STATE=CONNECTED;
        CloseConn(hDataSock);

}
/* Function : InitDataConn() */
SOCKET InitDataConn(PSOCKADDR_IN pstName,HWND hwndDlg,u_int nAsyncMsg)
{
```

```
int nRet;
        hDataSock=socket(AF_INET,SOCK_STREAM,0);
        if(hDataSock==INVALID_SOCKET)
                {
                        MessageBox(NULL,"socket()","ERROR",MB_OK);
                }
        else
                {
                        /* request async notification for most events */
                        nRet=WSAAsyncSelect(hDataSock,hwndDlg,nAsyncMsg,
                                (FD_CONNECT|FD_READ|FD_WRITE|FD_CLOSE));
                                if(nRet==SOCKET_ERROR)
                                        {
                                                MessageBox(NULL,"WSAAsyncSelect
()","ERROR",MB_OK);
                                                closesocket(hDataSock);
                                                hDataSock=INVALID_SOCKET;
                                        }
                                else
                                        {
                                                pstName->sin_family=PF_INET;
                                                pstName->sin_port=htons(iDataPort);
                                                nRet=connect(hDataSock,
(LPSOCKADDR)pstName,sizeof(struct sockaddr));
                                                if(nRet==SOCKET_ERROR)
                                                        {
                                                                int
WSAErr=WSAGetLastError();
                                                                if
(WSAErr!=WSAEWOULDBLOCK)
```

```
void SolveDataPort()
{
int i,count;
                    i=0;
                    count=0;
                    MessageBox(NULL,FtpCmd.szArg,"host",MB_OK);
               while(count!=4)
                    {
                         szRmtHost[i]=FtpCmd.szArg[i];
                              if(FtpCmd.szArg[i]==',')
                                   {
                                   count++;
                                   szRmtHost[i]='.';
                                   }
                              else if(FtpCmd.szArg[i]=='\0')
                                   {
                                   MessageBox(NULL,"break","",MB_OK);

          MessageBox(NULL,FtpCmd.szArg,"",MB_OK);
                                   break;
                                   }
                              i++;
                    }
                    szRmtHost[--i]='\0';
                    /* จบการเปลี่ยนค่า argument ip */
                    char szDataPortH[4];int nPortH;
                    char szDataPortL[4];int nPortL;
                    char szDataPort[5];char szHexPort[5];
                    int j=0;
```

```
while(FtpCmd.szArg[++i]!=',')
        {
                szDataPortH[j++]=FtpCmd.szArg[i];
        }
        szDataPortH[j]='\0';
                nPortH=atoi(szDataPortH);
                //nPortH=(nPortH&0xFF);
        j=0;
while(FtpCmd.szArg[i++]!='\0')
        {
                szDataPortL[j++]=FtpCmd.szArg[i];
        }
        szDataPortL[j]='\0';
                nPortL=atoi(szDataPortL);
                //nPortL=(nPortL&0xFF);
        wsprintf(szBuf,"nPortH=%X nPortL=%X",nPortH,nPortL);
        MessageBox(NULL,szBuf,"Port",MB_OK);

        if(nPortL>=0&&nPortL<=15)
                wsprintf(szHexPort,"%X0%X",nPortH,nPortL);
        else
                wsprintf(szHexPort,"%X%X",nPortH,nPortL);
        MessageBox(NULL,szHexPort,"Port",MB_OK);
        char ch;
        int x;int nLen;
        iDataPort=0;int nHex;
        i=0;
        while(szHexPort[i]!='\0')i++;
                nLen=i;
                wsprintf(szBuf,"nLen=%d",nLen);
```

```
                                                                    {

            MessageBox(NULL,"connect()","Error",MB_OK);

            closesocket(hDataSock);

            hDataSock=INVALID_SOCKET;
                                                                    }
                                                                }
                                                            }
                                                        }
            return  hDataSock;
}

/*..... Function : AcceptCtrlConn() .....*/
/* ตอบรับ connection request จาก client */


/* Function: InitLstnSock() */
/* เตรียมการ listen  */

BOOL InitLstnSock(int iLstnPort,PSOCKADDR_IN pstSockName,HWND hwndDlg,u_int
nAsyncMsg)
{
            int nRet;
            int nLen=sizeof(struct sockaddr);

            hLstnSock=socket(AF_INET,SOCK_STREAM,0);
                        if(hLstnSock==INVALID_SOCKET)
                                    {
```

```
                    MessageBox(NULL,"error on socket()","Test",MB_OK);
                    return INVALID_SOCKET;
                }
            else
                {
                    nRet=WSAAsyncSelect(hLstnSock,hwndDlg,nAsyncMsg,
                    (FD_ACCEPT|FD_READ|FD_WRITE|FD_CLOSE));
                    if(nRet==SOCKET_ERROR)
                        {
                            MessageBox(NULL,"error on WSAAsyncSelect
()","ERROR!!",MB_OK);
                            return INVALID_SOCKET;
                        }
                    else
                        {
                            pstSockName->sin_family=PF_INET;
                            pstSockName->sin_port=htons(iLstnPort);

                            nRet=bind(hLstnSock,(LPSOCKADDR)
pstSockName,nLen);
                            if(nRet==SOCKET_ERROR)
                                {
                                    MessageBox(NULL,"error on
bind()","ERROR!!",MB_OK);
                                    return INVALID_SOCKET;
                                }
                            else
                                {
                                    nRet=listen(hLstnSock,5);
                                    if(nRet==SOCKET_ERROR)
```

```
                                                    {
                                                        MessageBox
(NULL,"error on listen()","ERROR!!",MB_OK);
                                                        return
INVALID_SOCKET;
                                                    }
                                                }
                                            }
                                        if(nRet==SOCKET_ERROR)
                                            {
                                                closesocket(hLstnSock);
                                                hLstnSock=INVALID_SOCKET;
                                            }
                                        }

                                    Display("Listening\r\n");
                                    //MessageBox(NULL,"Listening","Test",MB_OK);
                                    return (hLstnSock);

} /* end InitLstnSock() */

SOCKET AcceptCtrlConn(SOCKET hLstnSock,PSOCKADDR_IN pstName)
{
        SOCKET hNewSock;
        int nLen=sizeof(struct sockaddr);

        hNewSock=accept(hLstnSock,(LPSOCKADDR)pstName,(LPINT)&nLen);
            if(hNewSock==INVALID_SOCKET)
                {
                    int WSAErr=WSAGetLastError();
```

```
nRet=recv(hCtrlSock,(LPSTR)szFtpCmd,nLen,0);
if(nRet==SOCKET_ERROR)
        {
                int WSAErr=WSAGetLastError();
                if(WSAErr!=WSAEWOULDBLOCK)
                        {
                                MessageBox(NULL,"RecvFtpCmd
()","Error",MB_OK);
                        }
        }
i=0;

/* ตัดเฉพาะ FTP Command จาก buffer ที่รับ Command มา (szFtpCmd) */
while((ch=szFtpCmd[i])!=' '&&(ch=szFtpCmd[i])!='\n')
        {
                FtpCmd.szCmd[i]=ch;
                i++;
        }
                if(ch=='\n')
                        {
                                i--;
                                strcpy(FtpCmd.szArg,"no argument");
                        }
                FtpCmd.szCmd[i]='\0';

/* ตัดเฉพาะ Argument */
        i++; // เลื่อนไป 1 ช่อง
        int j=0;
        while((ch=szFtpCmd[i])!='\n')
```

```
                {
                        FtpCmd.szArg[j++]=ch;
                        i++;
                }
                FtpCmd.szArg[--j]='\0';
        }

        return nRet;
}

/* Fucntion : SendFtpReply()  */
int SendFtpReply(SOCKET hCtrlSock,LPSTR szFtpReply,int nLen)
{
        int nRet=0;
                if(hCtrlSock!=INVALID_SOCKET)
                        {
                                nRet=send(hCtrlSock,(LPSTR)szFtpReply,nLen,0);
                                if(nRet==SOCKET_ERROR)
                                        {
                                                int WSAErr=WSAGetLastError();
                                                if(WSAErr!=WSAEWOULDBLOCK)
                                                        {

        MessageBox(NULL,"SendFtpReply","Error",MB_OK);
                                                        }
                                        }
                        }
        return nRet;
}
```

```
                MessageBox(NULL,szBuf,"port",MB_OK);
        j=nLen;
        for(i=0;i<nLen;i++)
                {
                        j--;
                        szDataPort[j]=szHexPort[i];
                }
                szDataPort[nLen]='\0';
        wsprintf(szBuf,"szDataPort=%s",szDataPort);
        MessageBox(NULL,szBuf,"",MB_OK);
        for(j=nLen-1;j>=0;j--)
                {
                        ch=szDataPort[j];
                        if(ch>='0'&&ch<='9')
                                x=ch-'0'; //convert char to int
                        else if(ch>='A'&&ch<='F')
                                x=ch-'A'+10;  //convert char hex
to int
                        wsprintf(szBuf,"szDataPort
[%d]=%d",j,x);
                        MessageBox(NULL,szBuf,"",MB_OK);
                        switch(j)
                        {
                                case 0:nHex=1;break;
                                case 1:nHex=16;break;
                                case 2:nHex=256;break;
                                case 3:nHex=4096;break;
                        }
                        iDataPort=x*nHex+iDataPort;
                }
```

```
                                        wsprintf(szBuf,"Port=%d",iDataPort);
                                        MessageBox(NULL,szBuf,"Port",MB_OK);
        }

    /* end SolveDataPort() */


        /* Fucntion : GetHostID()  */
        /* return ค่า IP address */

        LONG GetHostID()
        {
                char szLclHost[MAXHOSTNAME];
                LPHOSTENT lpstHostent;
                SOCKADDR_IN stLclAddr;
                int nAddrSize=sizeof(SOCKADDR);
                int nRet;

                /* init local address (to zero) */
                stLclAddr.sin_addr.s_addr=INADDR_ANY;

                /* get the local host name */
                nRet=gethostname(szLclHost,MAXHOSTNAME);
                if(nRet!=SOCKET_ERROR)
                        {
                                /* resolve host name for local address */
                                lpstHostent=gethostbyname((LPSTR)szLclHost);
                                if(lpstHostent)
                                        stLclAddr.sin_addr.s_addr=*((u_long FAR*)(lpstHostent->
        h_addr));
```

```
                }

        return (stLclAddr.sin_addr.s_addr);
} /* end GetHostID() */


void CloseConn(SOCKET hSock)
{
        closesocket(hSock);
}

u_long GetAddr(LPSTR szHost)
{
        LPHOSTENT lpstHost;
        u_long lAddr=INADDR_ANY;
        /* check that we have a string */
        if(*szHost)
                {
                        lAddr=inet_addr(szHost);
                        if((lAddr==INADDR_NONE)&&(strcmp(szHost,"255.255.255.255")))
                                {
                                        lpstHost=gethostbyname(szHost);
                                        if(lpstHost)
                                                {
                                                        lAddr=*((u_long FAR*)(lpstHost->
h_addr));
                                                }
                                                else
                                                        {
```

```
                                                    lAddr=INADDR_ANY;
                                                }
                                            }
                                        }
                                        return lAddr;
}

BOOL GetLclDir(LPSTR szTempFile)
{
        //struct _finddata_t stFile;
        return FALSE;
}
```

ประวัติผู้เขียนโครงงาน

นายภาณุพงศ์ สอนคม
เกิดวันที่ 22 กุมภาพันธ์ พ.ศ. 2522
ภูมิลำเนา 83/224 ถ.ประชาอุทิศ อ.เมือง จ.พิษณุโลก 65000
โทร 055-215219 E:mail - bouble@thaimail.com

ประวัติการศึกษา

- ปีการศึกษา 2533 จบการศึกษาชั้นประถมศึกษาปีที่ 6 จากโรงเรียนอนุบาลพิษณุโลก
- ปีการศึกษา 2536 จบการศึกษาชั้นมัธยมศึกษาตอนต้นจากโรงเรียนพิษณุโลก-พิทยาคม
- ปีการศึกษา 2539 จบการศึกษาชั้นมัธยมศึกษาตอนปลาย จากโรงเรียนพิษณุโลกพิทยาคม
- ปีการศึกษา 2539 ดำรงตำแหน่งประธานรุ่น ม.6/2539 โรงเรียนพิษณุโลกพิทยาคม
- ปีการศึกษา 2540 ศึกษาต่อชั้นอุดมศึกษา ณ มหาวิทยาลัยนเรศวร
- ปีการศึกษา 2540 ดำรงตำแหน่งคณะกรรมการสโมสรนิสิต คณะวิศวกรรมศาสตร์
- ปีการศึกษา 2541 ดำรงตำแหน่งสมาชิกสภานิสิตมหาวิทยาลัยนเรศวร
- ปีการศึกษา 2541 ได้รับรางวัลนิสิตดีเด่นด้านกิจกรรมเสริมหลักสูตร
- ปีการศึกษา 2541 ประธานโครงการเตรียมความพร้อมทางวิชาการ ครั้งที่ 2
- ปีการศึกษา 2542 ดำรงตำแหน่งสมาชิกสภานิสิตมหาวิทยาลัยนเรศวร และบรรณาธิการหนังสือพิมพ์เสียงนิสิต
- ปีการศึกษา 2542 ได้รับรางวัลนิสิตดีเด่นด้านผลการเรียน ประจำปีการศึกษา 2540
- ปีการศึกษา 2542 ประธานโครงการเตรียมความพร้อมทางวิชาการ ครั้งที่ 3
- ปีการศึกษา 2540–2542 ดำรงตำแหน่งรองประธานชมรมวิชาการฯ
- ปีการศึกษา 2543 เป็นวิทยากรพิเศษในรายวิชาปฏิบัติงานเทคโนโลยีการศึกษา 2 (การออกแบบและผลิตสื่อด้วยคอมพิวเตอร์) ให้กับนักศึกษาวิชาเอกเทคโนฯ ค.บ./41 สถาบันราชภัฏพิบูลสงคราม พิษณุโลก

โชติ ขำเพชร
เกิดวันที่ 3มิถุนายน พ.ศ.2521
ภูมิลำเนา 99/19 ถ.พญาเสือ อ.เมือง จ.พิษณุโลก 65000
โทร 055-213214 E:mail - Superior_s@usa.net

ประวัติการศึกษา

- ปีการศึกษา 2533 จบการศึกษาชั้นประถมศึกษาปีที่ 6 จากโรงเรียนวัดดอนไก่เตี้ย จังหวัดเพชรบุรี
- ปีการศึกษา 2536 จบการศึกษาชั้นมัธยมศึกษาตอนต้น จากโรงเรียนพิษณุโลกพิทยาคม
- ปีการศึกษา 2539 จบการศึกษาชั้นมัธยมศึกษาตอนปลายจากโรงเรียนพิษณุโลกพิทยาคม
- ปีการศึกษา 2540 ศึกษาต่อชั้นอุดมศึกษา ณ มหาวิทยาลัยนเรศวร
- ปีการศึกษา 2540 ดำรงตำแหน่งคณะกรรมการสโมสรนิสิต คณะวิศวกรรมศาสตร์

ชุลีชา คะชา
เกิดวันที่ 18กุมภาพันธ์ พ.ศ.2522
ภูมิลำเนา 345/1 ถ.พิษณุโลกหล่มสัก อ.เมือง จ.พิษณุโลก 65000
โทร 055-223754 E:mail - Khawtu@hotmail.com

ประวัติการศึกษา

- ปีการศึกษา 2533 จบการศึกษาชั้นประถมศึกษาปีที่ 6 จากโรงเรียนอนุบาลพิษณุโลก
- ปีการศึกษา 2536 จบการศึกษาชั้นมัธยมศึกษาตอนต้นจากโรงเรียนเฉลิมขวัญสตรี
- ปีการศึกษา 2538 จบการศึกษาชั้นมัธยมศึกษาตอนปลาย จากโรงเรียนเฉลิมขวัญสตรี
- ปีการศึกษา 2540 ศึกษาต่อชั้นอุดมศึกษา ณ มหาวิทยาลัยนเรศวร