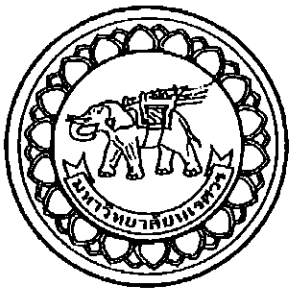


การพัฒนาโปรแกรมสำหรับเครื่องแม่ข่าย
เพื่อการรับส่งจดหมายอิเล็กทรอนิกส์ 2

THE DEVELOPMENT OF SERVER PROGRAMING
FOR ELECTRONIC MAIL II

นายอารักษ์	คเชนทร์	รหัส 43360692
นายเจน	โพธิ์ลังกา	รหัส 43360700
นายรณเชษฐ	เกตุเขียว	รหัส 43360544

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2546

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	**การพัฒนาโปรแกรมสำหรับเครื่องแม่ข่าย เพื่อการรับส่งจดหมายอิเล็กทรอนิกส์ 2**		
ผู้ดำเนินโครงงาน	นายอารักษ์	คเชนทร์	รหัส 43360692
	นายเจน	โพธิ์ลังกา	รหัส 43360700
	นายรณเชษฐ	เกตุเขียว	รหัส 43360544
อาจารย์ที่ปรึกษา	อาจารย์ภาณุพงษ์	สอนคม	
สาขา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2546		

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร สาขาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(อาจารย์สุชาติ แย้มเม่น)

.....กรรมการ

(อาจารย์พงศ์พันธ์ กิจสนาโยธิน)

.....กรรมการ

(อาจารย์ศิริพร เดชะศิลารักษ์)

หัวข้อโครงงาน	การพัฒนาโปรแกรมสำหรับเครื่องแม่ข่ายเพื่อการรับส่งจดหมายอิเล็กทรอนิกส์ 2		
ผู้ดำเนินโครงงาน	นายอารักษ์	คเชนทร์	รหัส 43360692
	นายเจน	โพธิ์ลังกา	รหัส 43360700
	นายรณเชษฐ	เกตุเขียว	รหัส 43360544
อาจารย์ที่ปรึกษา	อาจารย์ภาณุพงศ์ สอนคม		
อาจารย์ที่ปรึกษาร่วม	ดร.สุชาติ แย้มเม่น		
สาขา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2546		

.....

บทคัดย่อ

โครงงานนี้เป็นการศึกษา และพัฒนาโปรแกรมสำหรับใช้รันบนเครื่องคอมพิวเตอร์แม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ หรือเรียกอีกอย่างหนึ่งว่าโปรแกรมเมล์เซิร์ฟเวอร์ (Mail Server Program) ซึ่งเป็นโปรแกรมที่ให้บริการบนระบบเครือข่าย โดยโปรแกรมสามารถส่งจดหมายอิเล็กทรอนิกส์(e-mail) จากเครื่องคอมพิวเตอร์ลูกข่าย (client) ด้วยการติดต่อกับโปรแกรมเมล์เซิร์ฟเวอร์ที่คอมพิวเตอร์แม่ข่าย (Server) การพัฒนาโปรแกรมนี้ใช้ภาษาจาวา (JAVA) ในการพัฒนาซึ่งสามารถรันได้ทุกระบบปฏิบัติการ โปรแกรมนี้อ้างอิงการทำงานตามโปรโตคอล SMTP อาร์เอฟซี 822 (RFC 822) โปรโตคอล POP3 อาร์เอฟซี 1939 (RFC 1939) และโปรโตคอล IMAP4 อาร์เอฟซี 1730 (RFC 1730)

ผลที่ได้จากการทำโครงงานนี้ คือ ได้โปรแกรมสำหรับเครื่องคอมพิวเตอร์แม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์บนระบบเครือข่าย ตามโปรโตคอล SMTP อาร์เอฟซี 822 (RFC 822) โปรโตคอล POP3 อาร์เอฟซี 1939 (RFC 1939) และโปรโตคอล IMAP4 อาร์เอฟซี 1730 (RFC 1730)ซึ่งเป็นมาตรฐานในการรับส่งจดหมายอิเล็กทรอนิกส์ในปัจจุบัน และโปรแกรมที่ได้จะมีส่วนของ Graphic User Interface ซึ่งจะช่วยการจัดการผู้ใช้งานและสถิติของเซิร์ฟเวอร์เป็นเรื่องง่ายขึ้น

Project Title	**The development of the server programming for electronic mail II**		
Name	Mr.Ar-ruk	Kachen	ID. 42360692
	Mr. Jane	Pholangka	ID. 42360780
	Mr. Ronnachate	Gatekieo	ID. 43360544
Project Advisor	Mr. Panupong	Sornkom	
Co- Project Advisor	Dr. Suchart	Yammen	
Major	Computer Engineering		
Department	Electrical and Computer Engineering		
Academic Year	2003		

.....

ABSTRACT

This project is to study and develop a program for the server to getting an electronic mail. In another way, it can call Mail Server Program that gives the service to a computer network. This program can send the electronic mail from clients by connecting with the program mail server at the server. This developing program uses the JAVA language. It can run on any operating system server. This program involes with protocol SMTP RFC 822 , protocol POP3 RFC 1939 and protocol IMAP4 RFC 1730

The result of this project is the program for the server to sending and getting the electronic mail on the network based on protocol SMTP RFC 822 , protocol POP3 RFC 1939 and protocol IMAP RFC 1730. The program have a Graphic User Interface that can make User Management and Server statistics be easily used.

กิตติกรรมประกาศ

ผู้จัดทำโครงงาน ขอขอบพระคุณ อาจารย์ ภาณุพงศ์ สอนคม อาจารย์ที่ปรึกษา อาจารย์ ดร.สุชาติ แย้มเม่น อาจารย์ที่ปรึกษาร่วม พี่ไพรัตน์ โพธิ์ศรี เป็นอย่างสูงที่กรุณาสละเวลาให้คำปรึกษาและคำแนะนำทฤษฎี วิธีการที่เป็นประโยชน์ในการทำโครงงาน

ขอขอบพระคุณอาจารย์ทุกท่านที่ได้ให้ความรู้ความเข้าใจในวิชาต่างๆจนสามารถนำมาประยุกต์ใช้และให้คำแนะนำในการทำโครงงานครั้งนี้ได้

และขอกราบขอบพระคุณ พ่อ แม่ พี่น้องและเพื่อนๆสำหรับการสนับสนุน และความช่วยเหลือในด้านต่างๆตลอดมา

นายรณเชษฐ เกตุเขียว
นายเจน โพธิ์ลังกา
นายอารักษ์ คเชนทร์

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

การติดต่อสื่อสาร เพื่อแลกเปลี่ยนข้อมูลข่าวสาร มีความสำคัญต่อคนทุกยุคทุกสมัย ตั้งแต่อดีตจนถึงปัจจุบัน รูปแบบของการติดต่อสื่อสารได้มีการพัฒนาอย่างต่อเนื่อง ทั้งทางด้านความรวดเร็วในการแลกเปลี่ยนข้อมูล ความถูกต้องของข้อมูล รูปแบบของการติดต่อสื่อสารที่มีความหลากหลาย ตั้งแต่ จดหมาย โทรเลข โทรศัพท์บ้าน โทรศัพท์มือถือ พัฒนามาเป็นอินเตอร์เนต ซึ่งเทคโนโลยีอินเตอร์เนตเองก็สามารถแยกรูปแบบการสื่อสารได้อีกหลายรูปแบบ อาทิ เช่น chat ,web board, video conference และ electronic mail (จดหมายอิเล็กทรอนิกส์) หรือที่เราเรียกง่าย ๆ ว่า e-mail นั่นเอง

โปรแกรมสำหรับเครื่องแม่ข่ายในการรับส่งจดหมายอิเล็คทรอนิสก์ในประเทศไทย มิได้มีการพัฒนาอย่างกว้างขวาง ทำให้ความรู้ทางด้านนี้ยังไม่เป็นที่แพร่หลายมากนัก ทำให้ต้องใช้โปรแกรมจากต่างประเทศ ต้องสูญเสียเงินไปเป็นจำนวนมาก และเป็นสาเหตุนึงที่ทำให้เกิดปัญหาการละเมิดลิขสิทธิ์ทรัพย์สินทางปัญญา

ทางคณะผู้เสนอโครงการ จึงเล็งเห็นความสำคัญ ในการพัฒนาโปรแกรมสำหรับเครื่องแม่ข่ายเพื่อรับส่งจดหมายอิเล็กทรอนิกส์ (mail server) เพื่อนำไปใช้เป็นกรณีศึกษาและเป็นต้นแบบกระจายความรู้ให้กับคนไทยซึ่งจะทำให้เกิด software ของไทย และลดการละเมิดลิขสิทธิ์ทรัพย์สินทางปัญญา

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อพัฒนาเพิ่มเติมจากระบบเดิมที่มีอยู่ เพื่อให้มีความสามารถมากขึ้นกว่าเดิม และรองรับการใช้งานได้หลากหลายยิ่งขึ้น

1.2.2 เพื่อเป็นกรณีศึกษาการพัฒนาโปรแกรมบนระบบเครือข่าย ซึ่งจะเป็นองค์ความรู้ที่เป็นของคนไทยได้ศึกษาและพัฒนาต่อไป

1.2.3 เพื่อลดการนำเข้าซอฟต์แวร์จากต่างประเทศและลดการละเมิดลิขสิทธิ์ซอฟต์แวร์

1.3 ขอบข่ายการทำงาน

1.3.1 โปรแกรมสามารถรับส่งจดหมายอิเล็กทรอนิกส์ได้ตามมาตรฐานโปรโตคอล คือ SMTPPOP3 และ IMAP4

1.3.2. มีระบบบริหารจัดการของผู้ดูแลระบบ ซึ่งมีความสามารถ ดังนี้

1.3.2.1 สามารถเพิ่มและลบผู้ใช้งาน โดย ผู้ดูแลระบบเป็นผู้กำหนด

1.3.2.2 สามารถลบผู้ใช้งานที่ไม่ได้เข้ามาใช้งานเป็นระยะเวลาที่กำหนดได้โดยอัตโนมัติ

1.3.2.3 สามารถกระจายข่าวให้แก่ผู้ใช้งานในกรณีที่มีข่าวต้องการประชาสัมพันธ์

1.3.2.4 จะมีรายงานสถิติเพื่อใช้ในการเก็บข้อมูล จำนวนผู้ใช้งานทั้งหมด จำนวนผู้ใช้งานในแต่ ละวัน ปริมานการรับ-ส่ง E mail ในแต่ละวัน

1.3.2.5 สามารถกำหนดปริมาณพื้นที่ในการเก็บข้อมูลและ จดหมายของผู้ใช้งานแต่ละคนได้

1.3.2.6 เมื่อพื้นที่เก็บเต็ม ผู้ใช้จะไม่สามารถรับส่งจดหมายได้ แต่จะสามารถ login เพื่อเข้ามาจัด การกับจดหมายตนเองได้

1.3.3 โปรแกรมสามารถติดตั้งและใช้งานได้รันทุกระบบปฏิบัติการที่มี java virtual machine

1.4 ขั้นตอนการทำงาน

1.4.1 เขียนโครงร่างการทำงาน

1.4.2 ศึกษาข้อมูลต่าง ๆ ดังต่อไปนี้

- มาตรฐานโปรโตคอล SMTP (Simple Mail Transfer Protocol) และ POP3 (Post Office Protocol) และ IP address
- เทคโนโลยี RFC 822 , RFC 1939,RFC1730
- การสร้างซ็อกเก็ต
- การพัฒนาโปรแกรมบนทุกระบบปฏิบัติการ

1.4.3 วิเคราะห์และออกแบบโปรแกรม อัลกอริทึม โครงสร้างการทำงาน โดยรวบรวมตามความต้องการที่ได้ศึกษาจากข้อ 1 แล้วนำมาประยุกต์หรือดัดแปลง

1.4.4 พัฒนาโปรแกรมและคู่มือการใช้ โดยโปรแกรมจะถูกพัฒนาบนเครื่องคอมพิวเตอร์ส่วนบุคคล ซึ่งทำงานภายใต้ทุกสภาวะแวดล้อม

1.4.5 ทดสอบโปรแกรม

1.4.6 ปรับปรุงแก้ไขโปรแกรม

1.4.7 จัดทำเอกสาร

1.4.8 ส่งโครงงานฉบับสมบูรณ์

1.5 กิจกรรมการดำเนินการ

ตาราง 1.1 กิจกรรมการดำเนินงาน

กิจกรรม	ปี 2545		ปี 2546									
	พ.ย. 2545	ธ.ค. 2545	ม.ค. 2546	ก.พ. 2546	มี.ค. 2546	เม.ย.2546	พ.ค. 2546	มิ.ย. 2546	ก.ค. 2546	ส.ค. 2546	ก.ย. 2546	ต.ค. 2546
1. เขียนโครงร่างการทำงาน	←	→										
2. ศึกษาข้อมูล	←		→									
3.วิเคราะห์ออกแบบ		←		→								
4.พัฒนาโปรแกรม			←		→							
5.ปรับปรุงแก้ไขโปรแกรม						←					→	
6.จัดทำเอกสาร			←									→
7.ส่งโครงงานฉบับสมบูรณ์											←	→

1.6 ผลที่คาดว่าจะได้รับ

1.6.1 ได้โปรแกรมสำหรับเครื่องแม่ข่ายเพื่อรับส่งจดหมายอิเล็กทรอนิกส์ซึ่งพัฒนาโดยคนไทย

1.6.2 เกิดการกระจายความรู้ทางด้านการเขียนโปรแกรมบนระบบเครือข่ายซึ่งจะกระตุ้นให้เกิดการพัฒนาสิ่งใหม่ๆ ขึ้น

1.6.3 ลดการนำเข้าซอฟต์แวร์จากต่างประเทศ และ ลดปัญหาการละเมิดลิขสิทธิ์ซอฟต์แวร์

1.7 งบประมาณ

1.8.1 ค่ากระดาษ	300 -
1.8.2 ค่าหนังสือ และเอกสารอ้างอิง	1000-
1.8.3 ค่าจัดทำสื่อการนำเสนอโครงงาน	300-
1.8.4 ค่าทำรายงาน และจัดทำรูปเล่ม	500-
1.8.5 ค่าเรียบเรียงเอกสาร	250-

1.8.6 ค่าอุปกรณ์การทำงาน 650-

<u>หมายเหตุ</u> ขออนุมัติถัวเฉลี่ยทุกรายการ

บทที่ 2
หลักการและทฤษฎี

ในบทที่ 2 นี้ เป็นเนื้อหาของหลักการและทฤษฎีในการทำโปรแกรมสำหรับเครื่องคอมพิวเตอร์แม่ข่ายเพื่อรับส่งจดหมายอิเล็กทรอนิกส์ เนื่องจากในการทำโปรแกรมนี้ จะต้องใช้มาตรฐานการรับส่งข้อมูลต่าง ๆ จึงมีความจำเป็นที่ต้องศึกษาและทำความเข้าใจกลไกการทำงานและมาตรฐานต่าง ๆ ให้เข้าใจ ก่อนที่จะนำไปประยุกต์ในการพัฒนาโปรแกรมต่อไป

2.1 โปรโตคอลและ IP address

ในการใช้งานเครือข่ายคอมพิวเตอร์ทั่วไปหรือในเครือข่ายอินเตอร์เน็ตก็ตาม จะมีการส่งผ่านข้อมูลไปมาระหว่างเครื่องคอมพิวเตอร์หรือข้ามเครือข่ายออกไป ระบบคอมพิวเตอร์ที่เชื่อมต่ออยู่ในแต่ละเครือข่ายอาจจะใช้ฮาร์ดแวร์และซอฟร์แวร์ที่เหมือนกันหรือต่างกันได้ ดังนั้นการที่จะทำให้สามารถส่งผ่านข้อมูลถึงกันและตีความได้อย่างถูกต้องตรงกันจะต้องมีการติดต่อกันให้ตรงกัน เปรียบเสมือนกับการสื่อสารกันของมนุษย์เรา ถ้าเราต้องการจะติดต่อกับผู้คนต่างเชื้อชาติต่างภาษากันให้เข้าใจกันได้ถูกต้องตรงกันจะต้องมีการกำหนดกลไกในการสื่อสารกันเสียก่อน เรียกว่าจะต้องกำหนดระเบียบวิธีในการติดต่อกันให้ตรงกัน เปรียบเสมือนกับการสื่อสารกันของมนุษย์เรา ถ้าเราต้องการจะติดต่อกับผู้คนต่างเชื้อชาติต่างภาษากันให้เข้าใจกันได้ถูกต้องตรงกัน ก็จะต้องตกลงกำหนดกันเสียก่อนว่า จะติดต่อกันอย่างไร ด้วยภาษาใดที่จะเข้าใจกันได้ เช่น ปัจจุบันมีการใช้ภาษาอังกฤษเป็นภาษากลางในการติดต่อกันมาก ทำให้เราพูดได้ว่า ภาษาอังกฤษเปรียบเสมือนเป็นภาษามาตรฐานในการสื่อสารของมนุษย์ได้ ถ้าพูดในแง่การสื่อสารข้อมูล เราก็พูดได้ว่าภาษาอังกฤษเป็นโปรโตคอลในการสื่อสารของมนุษย์ที่มีการใช้งานอย่างแพร่หลาย

2.1.1 โปรโตคอลคืออะไร

โปรโตคอล (Protocol) คือ ระเบียบวิธีที่กำหนดขึ้นสำหรับสื่อสารข้อมูล ให้สามารถส่งผ่านข้อมูลไปยังปลายทางได้อย่างถูกต้อง ในปัจจุบันโปรโตคอลในการสื่อสารข้อมูลก็มีหลายโปรโตคอล

2.1.2 IP address

หมายเลข IP address ถูกกำหนดขึ้นมาให้เป็นหมายเลขอ้างอิงประจำตัวของอุปกรณ์ต่าง ๆ ที่เชื่อมต่ออยู่ในเครือข่ายอินเตอร์เน็ต โดยการกำหนดหมายเลข IP address ให้แต่ละเครื่องหรือแต่ละ

อุปกรณ์นี้จะต้องไม่ซ้ำกัน ซึ่งหมายเลข IP address นี้จะไม่ถูกผูกติดกับตัวฮาร์ดแวร์แต่อย่างใด จึงสามารถกำหนดใหม่หรือแก้ไขเปลี่ยนแปลงได้เมื่อมีการเปลี่ยนตัวฮาร์ดแวร์

การทำงานของโปรโตคอล IP จำเป็นต้องอาศัยหมายเลข IP address นี้ เพื่อระบุและอ้างถึงอุปกรณ์ต่าง ๆ ที่ต่ออยู่ในเครือข่ายไม่ว่าจะเป็นเว็บเซิร์ฟเวอร์ เมล์เซิร์ฟเวอร์ อุปกรณ์ Router ฯลฯ หมายเลข IP address จะเป็นค่าตัวเลขขนาด 32 บิต ถูกแบ่งออกเป็นส่วนละ 8 บิต รวมเป็น 4 ส่วน และคั่นแต่ละส่วนด้วยเครื่องหมาย (.) ดังนั้นค่าตัวเลขในแต่ละส่วนจะมีได้ตั้งแต่ 0 ถึง 255 (2^8) ตัวอย่างของ IP address เช่น 205.144.78.1 หรือ 10.0.0.1 เป็นต้น

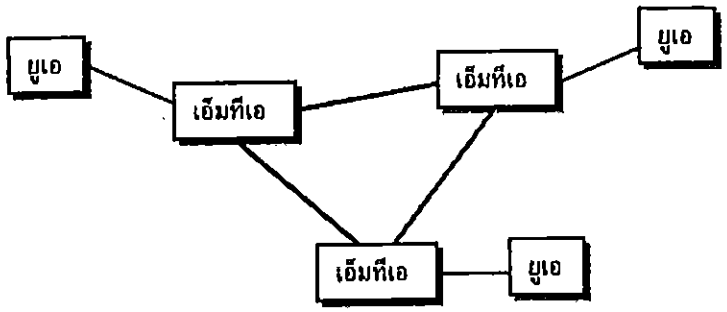
ค่าของ IP address จะถูกกำหนดออกเป็น 2 ความหมายคือ ค่าของหมายเลขอุปกรณ์ในเครือข่าย (host address) และค่าของหมายเลขเครือข่าย (network address) ตัวอย่างเช่น มีเครื่องเว็บเซิร์ฟเวอร์เชื่อมต่ออยู่ในเครือข่าย 2 เครื่อง โดยแต่ละเครื่องมี IP address ประจำตัวคือ 205.144.78.2 และ 205.144.78.3 ตามลำดับ แสดงว่าเครื่องทั้งสองต่อเชื่อมอยู่ในเครือข่ายเดียวกัน บนสายสัญญาณที่เชื่อมโยงเส้นเดียวกันแต่มีหมายเลขประจำตัวเครื่องที่แตกต่างกันคือ 2 และ 3 ตามลำดับ

2.2 สถาปัตยกรรมของระบบเมล์

ไปรษณีย์อิเล็กทรอนิกส์ หรืออิเล็กทรอนิกส์เมล์ซึ่งนิยมเรียกเพียงสั้น ๆ ว่า **อีเมล์** (e-mail) เป็นโปรแกรมประยุกต์ที่แพร่หลายในระบบเครือข่าย ประโยชน์ของอีเมล์ช่วยให้ผู้ใช้ส่งและรับข้อความข้ามเครือข่ายได้ ทีซีพี/ไอพี มีโปรโตคอลสนับสนุนการรับส่งเมล์หลายโปรโตคอล แต่โปรโตคอลที่นิยมใช้ในอินเทอร์เน็ตคือ เอสเอ็มทีพี (SMTP : Simple Mail Transfer Protocol) หน้าที่ของเอสเอ็มทีพีคือกำหนดกรรมวิธีและแบบแผนการนำส่งข้อความระหว่างผู้รับและผู้ส่ง เอสเอ็มทีพีอาศัยทีซีพีเพื่อลำเลียงจดหมายผ่านพอร์ต 25

ระบบเมล์ที่ใช้ในทีซีพีมีองค์ประกอบสองส่วนคือ ยูสเซอร์เอเจนต์ หรือ ยูเอ (UA : User Agent) และ เอ็มทีเอ (MTA : Message Transfer Agent) ทั้งยูเอและเอ็มทีเอ เป็นชื่อที่นำมาจากระบบ x.400 ซึ่งเป็นมาตรฐานนานาชาติกำหนดการนำส่งเมล์

ยูเอเป็นโปรแกรมติดต่อกับผู้ใช้และอำนวยความสะดวกให้ผู้ใช้เขียน แก้ไข และส่งจดหมาย รวมทั้งการเปิดอ่านจดหมายที่ได้รับ และจัดเก็บจดหมายเพื่อนำมาใช้ภายหลัง ส่วนเอ็มทีเอทำหน้าที่หาเส้นทางและส่งจดหมายไปถึงปลายทาง การติดต่อระหว่างเอ็มทีเอใช้ทีซีพีพอร์ต 25 โดยแลกเปลี่ยนคำสั่งตามรูปแบบที่กำหนดในอาร์เอฟซี 822



รูปที่ 2.1 สถาปัตยกรรมระบบเมล์ในทีซีพี/ไอพี

การจัดแบ่งออกเป็นยูเอและเอ็มทีเอมีข้อดีคือ แยกงานของทั้งสองส่วนให้เป็นอิสระจากกัน หน้าที่ของยูเอเน้นการทำงานกับผู้ใช้เพื่อให้ผู้ใช้อ่านเขียนจดหมายได้อย่างสะดวกโดยไม่ต้องยุ่งเกี่ยวกับการทำงานระดับล่างของโปรโตคอล ส่วนเอ็มทีเอทำงานตามเอสเอ็มทีพี เช่น การตรวจสอบความถูกต้องของแอสเดรสผู้รับผู้ส่ง รวมทั้งการหาเส้นทางและนำส่งจดหมายไปยังปลายทาง

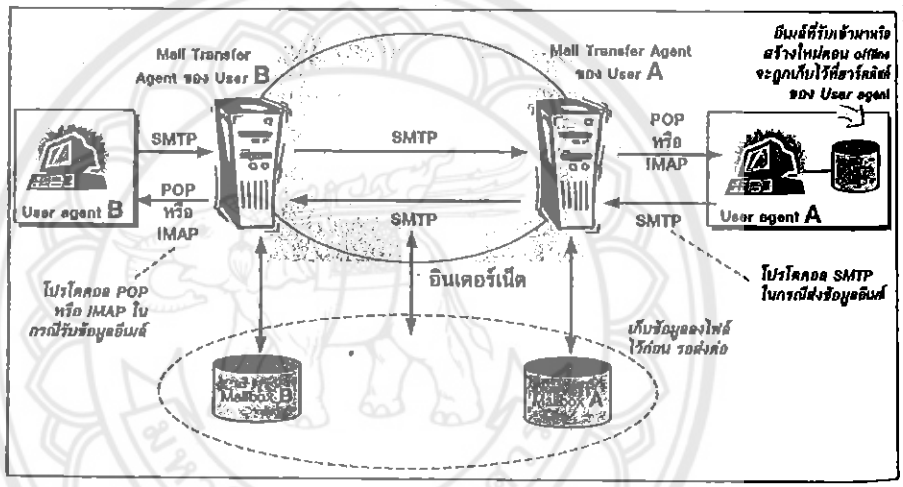
2.3 อีเมล์ และโปรโตคอลของอีเมล์

อีเมล์ (E-mail) หรือจดหมายอิเล็กทรอนิกส์ได้เริ่มใช้งานกันมานานแล้ว ตั้งแต่ยุคของเครื่องเมนเฟรมหรือมินิคอมพิวเตอร์ ซึ่งไอบีเอ็มได้พัฒนาระบบอีเมล์ที่เรียกว่า PROFS (Professional Office System) ออกมาใช้งาน นอกจากนี้ก็มีบนระบบ UNIX ต่อมาหลาย ๆ ค่าย ก็ได้พัฒนาระบบอีเมล์ของตนขึ้นมา โดยส่วนใหญ่จะเป็นองค์ประกอบในแอพพลิเคชันที่ทำงานบนระบบเน็ตเวิร์ก เช่น Microsoft Mail ของไมโครซอฟท์ และ cc:Mail ของโลตัส เป็นต้น ซึ่งต่างก็ใช้เทคโนโลยีของตนเองและเป็นระบบปิด ดังนั้นการส่งเมล์ไปยังอีกผู้ใช้ที่ใช้ระบบเมล์คนละค่ายกันจึงเป็นเรื่องที่ยุ่งยาก

ในยุคต่อมาที่ระบบเน็ตเวิร์กทั้ง LAN และ WAN ต่างมีมาตรฐาน และเป็นระบบเปิด (Open System) มากขึ้น ก็ได้ปรับเปลี่ยนการทำงานของอีเมล์มาเป็นแบบไคลเอนต์เซิร์ฟเวอร์ที่เป็นพื้นฐานแบบที่ใช้กันในระบบ UNIX และมีการพัฒนาอีเมล์เซิร์ฟเวอร์ขึ้นมาโดยเฉพาะ เช่น Exchange Server ของไมโครซอฟท์ หรือ Note Server ของโลตัส เป็นต้น ซึ่งผู้ใช้จะติดต่อเข้าสู่อีเมล์เซิร์ฟเวอร์ได้ทั้งโดยการผ่านระบบ LAN หรือใช้โมเด็มเข้ามาจาก WAN ทำให้ผู้ใช้จะไม่เห็นไฟล์ในฮาร์ดดิสก์บนเซิร์ฟเวอร์เลย ดังนั้นความปลอดภัยของระบบจึงมีมากขึ้น จนในปัจจุบันได้พัฒนาขึ้นมาเป็นระบบ Workflow ที่ใช้อีเมล์เป็นพื้นฐาน การทำงานของอีเมล์ไคลเอนต์และอีเมล์เซิร์ฟเวอร์มีส่วนประกอบดังนี้

- **User Agent** เป็นโปรแกรมคอมพิวเตอร์ทางด้านผู้ใช้งาน แบ่งเป็น 2 ส่วน คือ ส่วนของผู้ส่งและส่วนของผู้รับ โดย User agent นี้จะติดต่อเข้าสู่เซิร์ฟเวอร์ของตนโดยผ่านระบบ LAN หรือ Dial-up ซึ่งในส่วนของ User agent นี้จะเป็นส่วนที่ผู้ใช้ติดตั้งโปรแกรมไคลเอนต์ของอีเมล์เพื่อเรียกใช้บริการอีเมล์ เช่น Outlook Express หรือ Eudora เป็นต้น

- **MTA (Mail Tranfer Agent)** เป็นโปรแกรมคอมพิวเตอร์ที่จะส่งอีเมล์จากต้นทางไปยังผู้รับปลายทาง ซึ่งจะต้องส่งผ่านเครื่องจำนวนมากเป็นทอด ๆ จนไปถึงเครื่องที่มี account หรือเมล์บ๊อกซ์ของผู้รับ และหากไม่สามารถส่งเมล์ถึงผู้รับได้ (เพราะใส่ชื่อผิด) ยังทำหน้าที่ส่ง error mail กลับมาส่งผู้ส่งได้อีก ซึ่งเครื่องที่มี MTA ทำงานอยู่มักจะมีเมล์บ๊อกซ์ของผู้ใช้อยู่ด้วย ซึ่งเป็น primary mailbox และเรียกเครื่องนั้นว่า Mail Server



รูปที่ 2.2 องค์ประกอบและโปโตคอลต่าง ๆ ที่ใช้งานอยู่ในระบบการทำงานของอีเมล์ผู้ส่งจะใช้เครื่อง ของ User agent บันทึกอีเมล์และส่งอีเมล์ไปยังเครื่องที่มี MTA ทำงานอยู่ ซึ่งจะส่งต่อไปเครื่องอื่น ๆ ที่มี MTA เป็นทอด ๆ จนถึงเครื่องที่มี account หรือเมล์บ๊อกนั้นอยู่

2.3.1 อีเมล์แอดเดรส

ในการส่งเมล์ให้กับผู้รับนั้น การระบุชื่อผู้รับปลายทางนั้นมักจะเรียกกันว่า อีเมล์แอดเดรส (E-mail address) โดยปกติหากผู้รับมี mailbox อยู่ในเครื่องเดียวกันกับผู้ส่ง ก็สามารถระบุชื่อผู้ใช้หรือ user account ของผู้รับนั้นเป็นอีเมล์แอดเดรสได้เลย แต่หากผู้รับไม่ได้อยู่ในเครื่องเดียวกัน การระบุแอดเดรสปลายทางก็จะซับซ้อนขึ้น

การระบุอีเมล์แอดเดรสของผู้รับโดยทั่วไปถ้าส่งผ่านอินเตอร์เน็ต จะใช้วิธีที่เรียกว่า location-independent คือให้ระบบหาเอาเองว่าผู้รับอยู่ที่ใด ตัวอย่างเช่น edito@provision.co.th โดย editor เป็นชื่อผู้รับหรือ account ที่กำหนดให้ใช้ได้ในเครื่องนั้น ส่วน provision.co.th เป็นชื่อโดเมน ซึ่งอาจเป็นเพียงเครื่อง ๆ หนึ่งหรือเครือข่ายที่มีเครื่องหลายเครื่องที่เชื่อมต่อเข้าด้วยกัน ในลักษณะที่เป็นเครือข่ายนี้บางระบบอาจจำเป็นต้องระบุชื่อเครื่องที่ผู้ใช้นั้นอยู่ คือต้องระบุ host ไปด้วย แต่บางระบบก็อาจระบุชื่อโดเมนก็เพียงพอแล้ว เพราะระบบจะหาให้เองว่าผู้รับอยู่ในเครื่องใดในเครือข่ายนั้น

2.4 โปรโตคอลและประเภทการใช้งาน

การทำงานทั่ว ๆ ไป ของอีเมล์โดยสรุปมีเพียง 2 ประเภท คือ การส่งอีเมล์ และการรับอีเมล์ โดยโปรโตคอล SMTP หรือ Simple Mail Transfer Protocol จะใช้ขณะที่ User agent ส่งอีเมล์มาที่ MTA (เฉพาะแบบ Offline) และใช้ขณะรับและส่งอีเมล์ระหว่าง MTA ด้วยกัน สำหรับการใช้เมล์แบบ Offline คือเครื่องที่ผู้ใช้ใช้อ่านเมล์ไม่ได้ต่อกับเครื่องที่มีเมล์บ๊อกซ์ตลอดเวลา อาจเลือกดาวน์โหลดเมล์มาเก็บไว้ที่เครื่องของตัวเองนั้น จะมีโปรโตคอลสำหรับรับอีเมล์ที่เกี่ยวข้องอีก ที่ใช้งานกันแพร่หลายมีอยู่ 2 แบบ คือ โปรโตคอล POP หรือ Post Office Protocol และ IMAP หรือ Internet Message Access Protocol ซึ่งจะทำหน้าที่ดาวน์โหลดหรืออัพโหลดอีเมล์จากเครื่องของผู้ใช้ไปยังเครื่องที่มี MTA อยู่

รูปแบบของข้อมูลที่ใช้ในโปรโตคอลต่าง ๆ ของอีเมล์นี้ถูกกำหนดไว้ใน RFC 822 ซึ่งแบ่งส่วนประกอบภายในอีเมล์เป็น 2 ส่วน คือ ส่วนที่เป็นจ่าหน้าอีเมล์ และข้อมูลอีเมล์ ในส่วนของจ่าหน้าอีเมล์นี้มีไว้เป็นข้อมูลเพื่อให้ส่งไปถึงผู้รับ รูปแบบของข้อมูลจะเป็นข้อความหรือเท็กซ์ นำหน้าด้วยคำสำคัญ (keyword) เช่น From หมายถึงชื่อผู้ส่ง ส่วน To หมายถึงผู้รับ เป็นต้น ซึ่งจะคล้ายกับการที่ต้องกำหนดเมื่อบันทึกอีเมล์ ถัดจากคำสำคัญก็จะเป็นค่าของข้อมูลในชุดนั้น ๆ เช่น From ก็จะต่อด้วยชื่อผู้ส่ง และ Reply To หมายถึงผู้รับ เป็นต้น ซึ่งจะคล้ายกับการที่ต้องกำหนดเมื่อบันทึกอีเมล์ ถัดจากคำสำคัญก็จะเป็นค่าของข้อมูลในชุดนั้น ๆ เช่น From ก็จะต่อด้วยชื่อของผู้ส่ง และ Reply To ก็จะต่อด้วยชื่อของผู้รับ เป็นต้น โดยแต่ละบรรทัดจะปิดท้ายด้วย Carriage Return และ / หรือ Line Feed (ขึ้นอยู่ระบบปฏิบัติการที่ใช้ เช่น Windows จะปิดท้ายด้วย Carriage Return และ Line Feed ส่วนระบบปฏิบัติการอื่น เช่น Unix ก็อาจจะใช้เพียง Carriage Return เท่านั้น) เป็นเครื่องหมายของการสิ้นสุดบรรทัด จะเห็นได้ว่าในส่วนของจ่าหน้าอีเมล์นี้มีข้อความที่จำเป็นคือรายละเอียดของผู้ส่งและผู้รับ ส่วนรายละเอียดอื่น ๆ เช่น รายชื่อผู้รับสำเนา (Cc) จะมีหรือไม่ก็ได้

มาถึงส่วนที่เป็นข้อมูลของอีเมล์ ซึ่งจะแบ่งย่อยออกได้เป็น 2 ส่วนคือ ส่วนหัว (header) และส่วนเนื้อความของอีเมล์ ส่วนหัวนี้จะถูกสร้างขึ้นอย่างอัตโนมัติโดย User agent ของผู้ส่ง เพื่อ

ให้ MTA ต่าง ๆ ระหว่างทางที่ส่งผ่านอีเมล์ฉบับนั้นได้อ่านไปใช้งาน ซึ่งประกอบด้วยข้อมูลต่าง ๆ หลายประเภท ตัวอย่างของข้อมูลในส่วนหัวของอีเมล์ ได้แก่ เลขทะเบียนของอีเมล์ (Message Header), วันที่และเวลาที่ส่ง เป็นต้น ส่วนที่เป็นเนื้อความของอีเมล์นั้น จะเป็นบรรทัดที่อยู่แยกจากส่วนหัว โดยถูกคั่นด้วยบรรทัดว่าง ๆ (Null Line) และในแต่ละบรรทัดของเนื้อความจะสิ้นสุดบรรทัดด้วย Carriage Return และ/หรือ Line Feed

ตามข้อกำหนด RFC 822 ในการส่งอีเมล์ผ่านอินเตอร์เน็ตนั้น แต่ละบรรทัดจะมีขนาดยาวได้ไม่เกิน 1,000 ไบต์ และขนาดของอีเมล์แต่ละครั้งจะไม่เกิน 64 กิโลไบต์ ซึ่งผู้ส่งไม่จำเป็นต้องสนใจว่าอีเมล์ที่ส่งไปนั้นจะผ่านไปที่ MTA ใดบ้าง เนื่องจากอีเมล์จะถูกเข้ารหัสและส่งไปถึงยัง User agent ของผู้รับปลายทางและผ่านการถอดรหัสได้โดยอัตโนมัติ

จากองค์ประกอบของโปรโตคอล และวิธีการรับส่งอีเมล์ที่กล่าวผ่านมา ทำให้การใช้อีเมล์ในปัจจุบันซึ่งทำงานแบบไคลเอนต์เซิร์ฟเวอร์ สามารถทำงานได้ 3 แบบ คือ

- แบบ Offline หรือเรียกว่า Download and Delete ซึ่งเป็นรูปแบบมาตรฐานทั่วไปในการใช้งานกับอีเมล์ของอินเตอร์เน็ต ซึ่งใช้โปรโตคอล POP หรือ IMAP โดย User agent ของผู้รับจะดาวน์โหลดอีเมล์ทั้งหมดมาจากอีเมล์บางโปรแกรม สามารถให้เลือกได้ว่าต้องการลบอีเมล์ที่ดาวน์โหลดมาแล้วทางฝั่งเซิร์ฟเวอร์นั้นทิ้งหรือไม่) ทำให้ผู้ใช้สามารถอ่านอีเมล์นั้นได้ตลอดเวลาโดยไม่จำเป็นต้องติดต่อเมล์เซิร์ฟเวอร์อีก แต่ User agent จะไม่รู้ว่ามีอีเมล์เข้ามาใหม่จนกว่าจะติดต่อเข้าไปยังเมล์เซิร์ฟเวอร์และดาวน์โหลดอีเมล์เข้ามาใหม่

- แบบ Online เป็นแบบที่อีเมล์ด้าน User agent ของผู้รับจะต้องติดต่อกับเมล์เซิร์ฟเวอร์ของผู้รับเองตลอดเวลาที่ใช้อีเมล์ ซึ่งระบบที่ให้บริการอีเมล์แบบนี้จะสามารถเปิดแชร์เมล์บ็อกซ์ที่เซิร์ฟเวอร์ได้ตลอดเวลา เช่น NFS (Network File System) หรือ CIFS (Common Internet File System) เป็นต้น นอกจากนี้โปรโตคอลแบบ IMAP ยังสามารถใช้งานในแบบ Online นี้ได้อีกด้วย

- แบบ Disconnected เป็นแบบผสมผสานระหว่างแบบ Offline และแบบ Online โดยอาศัยเมล์เซิร์ฟเวอร์ของผู้รับเป็นหลักในการจัดเก็บข้อมูลของอีเมล์ และในส่วนเนื้อที่ของ User agent นี้จะเป็นที่เก็บอีเมล์สำรอง โดยเมื่อมีการดาวน์โหลดอีเมล์มาก็จะทำงานในแบบ Offline เพื่อลดภาระที่ต้องติดต่อกับเมล์เซิร์ฟเวอร์ตลอดเวลา แต่ข้อมูลอีเมล์จะไม่ถูกลบออกจากเมล์เซิร์ฟเวอร์ ผู้ใช้สามารถโหลดอีเมล์ที่แก้ไขแล้วกลับไปยังเมล์เซิร์ฟเวอร์ในภายหลังได้ เช่น การแก้ไขหรือตอบกลับอีเมล์ (Reply to) ที่ส่งมา เป็นต้น ซึ่งโปรโตคอลที่สามารถตอบสนองการใช้งานในแบบนี้ได้ก็คือ IMAP

2.5 POP3 (Post Office Protocol)

สำหรับผู้ที่ใช้งานอีเมล์บนอินเตอร์เน็ตมาแล้ว คงจะคุ้ยเคยกับโปรโตคอลที่เรียกว่า POP หรือ Post Office Protocol กันเป็นอย่างดี เพราะเป็นโปรโตคอลที่ทำหน้าที่โหลดอีเมล์มาจาก MTA ไปยัง User agent ซึ่งในปัจจุบันได้พัฒนามาจนถึงเวอร์ชัน 3 แล้ว หรือเรียกย่อ ๆ ว่า POP3 โปรโตคอลนี้เป็นตัวแรกที่ถูกออกแบบมาเพื่อใช้รับอีเมล์ และเพื่อให้สนับสนุนการทำงานในแบบ Offline โดยติดต่อเข้าไปยังเมล์เซิร์ฟเวอร์แล้วดาวน์โหลดอีเมล์ทั้งหมดไว้ที่ User agent จากนั้นจะลบอีเมล์ที่เซิร์ฟเวอร์นั้นทิ้งไป เพื่อป้องกันการดาวน์โหลดซ้ำ แต่ผู้ใช้จะทำงานแบบ Online กับเซิร์ฟเวอร์ไม่ได้ เนื่องจากการอ่านอีเมล์จะดึงอีเมล์ที่เก็บไว้ใน User agent ขึ้นมาให้อ่านหลังจากที่ดาวน์โหลดมาเก็บไว้ ซึ่งในขณะนั้นอาจจะไม่ได้ออนไลน์อยู่กับเน็ตเวิร์กก็ได้

โปรโตคอลของ POP3 นี้จะทำงานในแบบไคลเอนต์เซิร์ฟเวอร์ คือมีโปรแกรม POP Server ในเมล์เซิร์ฟเวอร์ และ POP client ในเครื่องผู้รับ ซึ่งปกติจะฝังอยู่ในโปรแกรมที่เป็น User agent เลย โปรแกรมทั้งสองจะติดต่อกันโดยใช้ชุดคำสั่งที่เป็นรหัส ASCII คือเมื่อด้านที่รับทำคำสั่งก็จะทำงานตามคำสั่งนั้น แล้วตอบกลับมาโดยมีค่าเป็น +OK หมายถึงทำงานได้เรียบร้อย หรือ –ERR หมายถึงเกิดปัญหาขึ้น

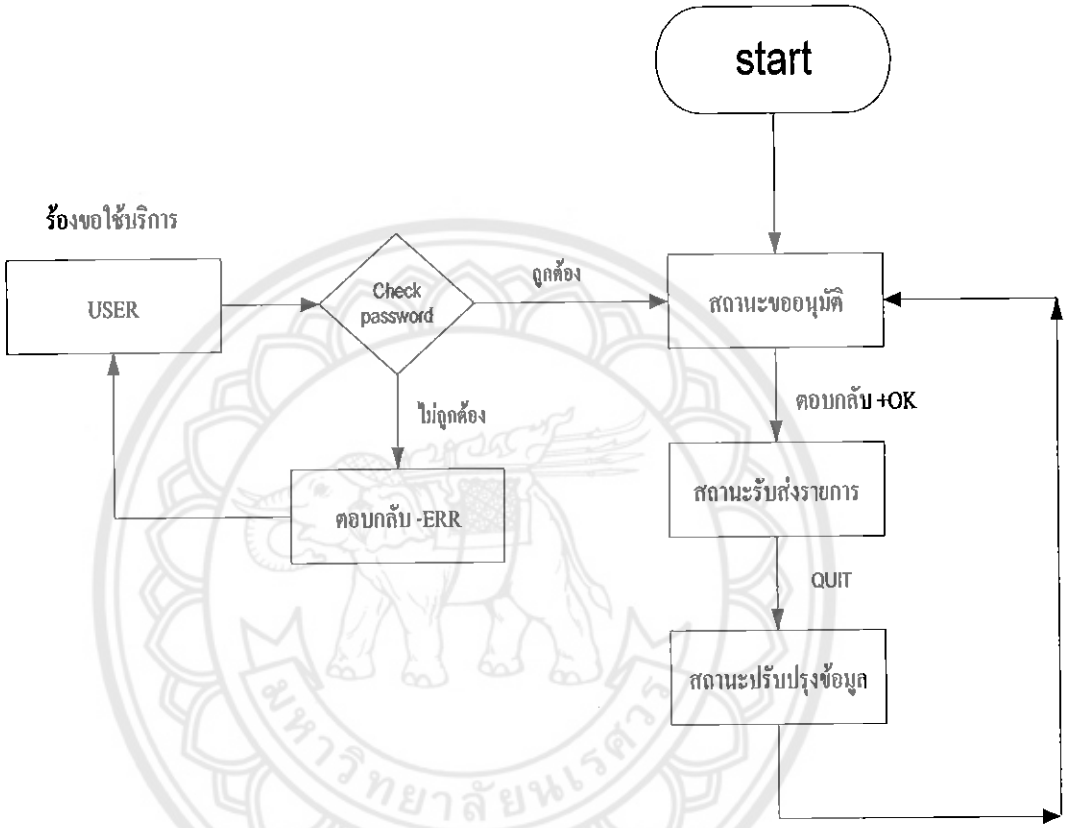
ทำงานไม่ได้ ซึ่งในคำสั่งที่ต้องมีการตอบกลับและส่งข้อมูลกลับมา โดยประกอบด้วยข้อมูลหลาย ๆ บรรทัดนั้น POP3 จะให้บรรทัดสุดท้ายเป็นเครื่องหมายจุด (.) ตามด้วย Carriage Return และ Line Feed หมายถึงการสิ้นสุดชุดข้อมูล แต่ในกรณีที่ข้อมูลบรรทัดสุดท้ายมีข้อมูลที่เป็นจุดด้วย จะใช้เทคนิคที่เรียกว่า Character Stuffing เพื่อแก้ปัญหา โดยจะเติมจุดลงไปอีกหนึ่งตัว เพื่อเป็นตัวบ่งชี้ว่าข้อมูลนั้นเป็นจุด ซึ่งจะแตกต่างจากสัญลักษณ์แสดงการสิ้นสุดของข้อมูล

การทำงานของ POP3 จะทำร่วมกับโปรโตคอล TCP โดยทั่วไปจะใช้ทีซีพีพอร์ต 110 การดำเนินการใดกับจดหมายจะเกิดกับจดหมายที่อ่านมาเก็บไว้ยังไคลเอ็นต์เท่านั้น การเปลี่ยนแปลงสถานะของจดหมายใด ๆ จะเกิดขึ้นที่ไคลเอ็นต์ที่ใช้เท่านั้น โดยไม่เปลี่ยนแปลงสถานะจดหมายที่เซอร์ฟเวอร์หน้าที่หลักของพ็อปจึงเป็นการส่งถ่ายจดหมายจากเซอร์ฟเวอร์มายังไคลเอ็นต์ ในการติดต่อ ขั้นตอนการทำงานของ POP3 ประกอบด้วย 3 สถานะ คือ สถานะขออนุมัติ, สถานะรับส่งรายการ และสถานะปรับปรุงข้อมูล ซึ่งในแต่ละสถานะจะรับรู้คำสั่งต่าง ๆ ของโปรโตคอลแตกต่างกัน โดยมีรายละเอียดต่าง ๆ ดังนี้

- **สถานะขออนุมัติ (Authorization State)** เมื่อเริ่มต้นติดต่อกับเซิร์ฟเวอร์จะเป็นการเข้าสู่สถานะการขออนุมัติ โดยไคลเอนจะต้องแจ้งชื่อผู้ใช้และรหัสผ่าน (Password) เพื่อขออนุมัติจากเซิร์ฟเวอร์ก่อน โดยไคลเอนต์จะใช้คำสั่ง USER เพื่อระบุชื่อผู้ใช้ หรือคำสั่ง PASS เพื่อกำหนด Password แต่ในกรณีที่ชื่อและ Password ถูกเข้ารหัสไว้ และไม่ได้เป็นค่า ASCII ทั่วไป ไคลเอนต์จะใช้คำสั่ง APOP ทำงานแทนคำสั่ง USER และ PASS

- **สถานะรับส่งรายการ (Transaction State)** หลังจากที่ได้รับอนุมัติจากเซิร์ฟเวอร์แล้ว ก็จะเข้าสู่สถานะที่ใช้คำสั่งในการทำงานต่าง ๆ

- **สถานะปรับปรุงข้อมูล (Update State)** เมื่อ User agent เลิกใช้งานด้วยคำสั่ง QUIT ของ POP3 เซิร์ฟเวอร์ก็จะเข้าสู่สถานปรับปรุงข้อมูล เพื่อลบอีเมล์ที่ดาวน์โหลดเรียบร้อยแล้วออกไป จากนั้นก็จะเข้าสู่สถานะขออนุมัติใหม่โดยอัตโนมัติ เพื่อรอรับการทำงานครั้งต่อไป



รูปที่ 2.3 แผนผังลำดับงาน (Flow Chart) ของ POP3

ตาราง 2.1 แสดงรายละเอียดในคำสั่งต่าง ๆ ของ POP 3 ที่ใช้งานอยู่

คำสั่ง	พารามิเตอร์	สถานะ	รายละเอียด
USER	ชื่อผู้ใช้งาน	ขออนุมัติ	แจ้งชื่อผู้ใช้ และระบุบ็อกซ์ที่จะใช้
PASS	Password	ขออนุมัติ	เป็นคำสั่งที่ใช้ระบุ Password โดยจะใช้ต่อจากคำสั่ง USER
APOP	ชื่อ,Password	ขออนุมัติ	ทำหน้าที่เหมือนคำสั่ง USER และ PASS รวมกัน แต่ข้อมูลถูกเข้ารหัสก่อนส่ง
STAT	ไม่ระบุ	รับส่งรายการ	เป็นคำสั่งตรวจสอบสภาพเซิร์ฟเวอร์ เช่น จำนวนอีเมล์ในเซิร์ฟเวอร์, ขนาดของอีเมล์ที่จะดาวน์โหลด
UIDL	หมายเลขข้อความ	รับส่งรายการ	ใช้ตรวจสอบหมายเลขประจำของอีเมล์
LIST	หมายเลขข้อความ	รับส่งรายการ	ใช้ตรวจสอบหมายเลขของอีเมล์และขนาดของอีเมล์
RETR	ข้อความ	รับส่งรายการ	เป็นคำสั่งใช้ส่งข้อมูลของอีเมล์
DELE	ข้อความ	รับส่งรายการ	เป็นการระบุเครื่องหมายการลบลงในอีเมล์ที่จะลบ และอีเมล์เหล่านั้นจะถูกลบออกจากเมล์บ็อกซ์เมื่อใช้คำสั่ง QUIT เมื่อสิ้นสุดการทำงาน
RSET	ไม่ระบุ	รับส่งรายการ	คำสั่งนี้จะยกเลิกเครื่องหมายการลบอีเมล์ที่เคยกำหนดไว้ด้วยคำสั่ง DELE ออกไปทุกๆ อีเมล์
TOP	หมายเลขข้อความ, จำนวนบรรทัด	รับส่งรายการ	เซิร์ฟเวอร์จะส่งข้อมูลย้อนกลับไปเท่ากับจำนวนบรรทัดที่ระบุไว้
NOOP	ไม่ระบุ	รับส่งรายการ	เป็นคำสั่ง No Operation
QUIT	ไม่ระบุ	รับส่งรายการและขออนุมัติ	ใช้เมื่อจบการทำงาน หากมีอีเมล์ซึ่งทำเครื่องหมายว่าจะลบไว้ อีเมล์เหล่านั้นจะ ถูกลบจากเมล์บ็อกซ์ในขั้นตอนนี้

2.5.1 สถานะขออนุมัติ (Authorization State)

เมื่อการเชื่อมต่อ TCP ถูกเปิดโดยไคลเอ็นท์ POP3 เครื่องแม่ข่าย POP3 จะทำการทักทาย ซึ่งจะมีการตอบกลับเป็นบวก ดังตัวอย่าง:

S: +OK POP3 server ready

ขณะ POP3 อยู่ในสถานะอำนาจที่ได้มอบหมายตอนนี้ ไคลเอ็นท์ต้องระบุและรับรองว่าเป็นสมาชิกในเซิร์ฟเวอร์ โดยการ USER และ PASS หรือรวมคำสั่งทั้งสองเข้าด้วยกันคือคำสั่ง APOP

เมื่อเครื่องแม่ข่าย POP3 ใช้คำสั่ง authentication แล้วไคลเอ็นท์ควรจะเข้าไปที่ maildrop ซึ่งเครื่องแม่ข่าย POP3 จะอนุญาติให้เข้าได้เฉพาะ maildrop ของผู้ใช้นั้นๆ เพราะจำเป็นที่จะป้องกันข่าวสารจากการแก้ไขหรือการเคลื่อนย้าย ถ้าการล็อคอินสมบูรณ์ เครื่องแม่ข่าย POP3 จะตอบสนองกลับเป็นสถานะบวก ส่วน POP3 จะเข้าไปสถานะTransaction ถ้า maildrop ไม่สามารถถูกเปิดด้วยอาจจะเหตุผลใดเหตุผลหนึ่ง (เป็นต้นว่า, การล็อคอินไม่สามารถเข้าได้, ไคลเอ็นท์ถูกปฏิเสธไม่ให้เข้าสู่ maildrop, หรือ maildrop ไม่สามารถถูกวิเคราะห์ได้) เครื่องแม่ข่าย POP3 จะตอบสนองกลับเป็น สถานะลบ (ถ้าการล็อคอินได้ แต่เครื่องแม่ข่าย POP3 ตั้งใจตอบสนองกลับเป็นลบ เครื่องแม่ข่าย POP3 ปล่อยการล็อคครั้งก่อนและปฏิเสธคำสั่ง) หลังจากที่กลับค่าตัววัดสถานะเป็นลบ เครื่องแม่ข่ายอาจจะปิดการเชื่อมต่อ แต่ถ้าเครื่องแม่ข่ายไม่ทำการปิดการเชื่อมต่อ, ไคลเอ็นท์อาจใส่คำสั่ง authentication และเริ่มต้นอีกครั้ง หรือไคลเอ็นท์อาจจะใช้คำสั่ง Quit

หลังจากเครื่องแม่ข่าย POP3 ได้เปิด maildrop, จะกำหนดตัวเลขให้ข่าวสารแต่ละอัน, และขนาดของข่าวสารแต่ละอันใน octet ข่าวสารแรกใน maildrop ถูกกำหนดตัวเลขข่าวสาร "1",ตัวที่สองถูกกำหนด "2" ,และอื่นๆ, จนถึงข่าวสาร n th ใน maildrop ถูกกำหนดตัวเลขข่าวสาร "n". ในคำสั่ง POP3 และตอบสนอง, ข่าวสารทั้งตัวเลขและขนาดข่าวสารถูกแสดงใน เลขฐาน 10

คำสั่ง QUIT

Arguments: none

Restrictions: none

Possible Responses: +OK

2.5.2 สถานะรับส่งรายการ (Transaction State)

เมื่อไคลเอ็นท์ล็อคอินเข้าเครื่องแม่ข่าย POP3 แล้วเครื่องแม่ข่าย POP3 ก็ล็อคและเปิดทางไปสู่ maildrop POP3 ในตอนนี้จะอยู่ในสถานะTransaction ไคลเอ็นท์จะป้อนของคำสั่งต่างๆ ของ POP3 ซึ่งหลังจากที่ป้อนคำสั่งแต่ละอัน, เครื่องแม่ข่าย POP3 ก็ตอบสนอง และเมื่อไคลเอ็นท์ใช้คำสั่ง QUIT และ POP3 จะเข้าไปสู่สถานะ Update คำสั่ง POP3 ที่ใช้ในสถานะรับส่งรายการ (Transaction State) มีดังต่อไปนี้

คำสั่ง STAT

พารามิเตอร์ : ไม่ระบุ

สถานะ : รับส่งรายการ

รายละเอียด : เป็นคำสั่งตรวจสอบสภาพของ server เช่น จำนวนอีเมล์ใน server , ขนาดของอีเมล์ที่จะดาวโหลด

คำสั่ง LIST

พารามิเตอร์ : หมายเลขข้อความ

สถานะ : รับส่งรายการ

รายละเอียด : ใช้ตรวจสอบหมายเลขของอีเมล และ ขนาดของอีเมล

คำสั่ง RETR

พารามิเตอร์ : ข้อความ

สถานะ : รับส่งรายการ

รายละเอียด : เป็นคำสั่งที่ใช้ส่งข้อมูลของอีเมล์

คำสั่ง DELE

พารามิเตอร์ : ข้อความ

สถานะ : รับส่งรายการ

รายละเอียด : เป็นการระบุเครื่องหมายการลบ ลงในอีเมล์ที่จะลบ และอีเมล์เหล่านั้นจะถูกลบออกจากเมล์บ๊อกส์เมื่อใช้คำสั่ง QUIT เมื่อสิ้นสุดการทำงาน

คำสั่ง NOOP

พารามิเตอร์ : ไม่ระบุ

สถานะ : รับส่งรายการ

รายละเอียด : เป็นคำสั่ง No Operation

คำสั่ง RSET

พารามิเตอร์ : ไม่ระบุ

สถานะ : รับส่งรายการ

รายละเอียด : คำสั่งนี้จะยกเลิกเครื่องหมายการลบอีเมล์ ที่เคยกำหนดไว้ด้วยคำสั่ง DELE ออกๆไปทุกๆ อีเมล์

2.5.3 สถานะปรับปรุงข้อมูล (Update State)

เมื่อไคลเอ็นท์ใส่คำสั่ง QUIT จากสถานะ Transaction , POP3 จะเข้าไปสถานะ UPDATE (ถ้าไคลเอ็นท์ใส่คำสั่ง QUIT จากสถานะ Authorization , POP3 ยกเลิกการติดต่อแต่จะไม่เข้าไปในสถานะ UPDATE)

ถ้ายกเลิกสำหรับเหตุผลใดๆ ที่มากกว่าคำสั่ง QUIT , POP3 จะไม่เข้าไปสู่สถานะ UPDATE และไม่ย้ายข่าวสารอย่างใดๆ ออกจาก maildrop

คำสั่ง QUIT

พารามิเตอร์ : ไม่ระบุ

สถานะ : รับส่งรายการและขออนุมัติ

รายละเอียด : ใช้เมื่อจบการทำงาน หากมีอีเมล์ซึ่งทำเครื่องหมายว่าจะลบไว้ อีเมล์เหล่านั้นจะถูกลบจากเมล์บ๊อกซ์ในขั้นตอนนี้

2.5.4 Optional POP3 Commands

นอกจากคำสั่งต่างๆ ที่กล่าวมาในข้างต้นยังมีคำสั่งพิเศษที่จะเอามาใช้กับ POP3 Server ได้ ดังนี้

คำสั่ง UIDL

พารามิเตอร์ : หมายเลขข้อความ

สถานะ : รับส่งรายการ

รายละเอียด : ใช้ตรวจสอบหมายเลขประจำของอีเมล์

คำสั่ง TOP

พารามิเตอร์ : หมายเลขบรรทัด , จำนวนบรรทัด

สถานะ : รับส่งรายการ

รายละเอียด : Server จะส่งข้อมูลย้อนกลับไปเท่ากับจำนวนบรรทัดที่ระบุไว้

คำสั่ง APOP

พารามิเตอร์ : ชื่อ, password

สถานะ : ขออนุมัติ

รายละเอียด : ทำหน้าที่ ให้ ระบุชื่อผู้ใช้ เมล์บ๊อกซ์ที่จะใช้ และ Password และข้อมูลจะถูกเข้ารหัสก่อนที่จะถูกส่งไป

2.6 IMAP4 (Internet Message Access Protocal4)

การทำงานของ IMAP นี้จะเหมือนกับโปรโตคอลอื่น ๆ โดยทำงานร่วมกับ TCP ใช้พอร์ตหมายเลข 143 และจะแบ่งสถานะต่าง ๆออกเป็น 4 สถานะ โดยในแต่ละสถานะจะมีวัตถุประสงค์และคำสั่งที่ใช้งานแตกต่างกันออกไป โดยมีรายละเอียดดังนี้

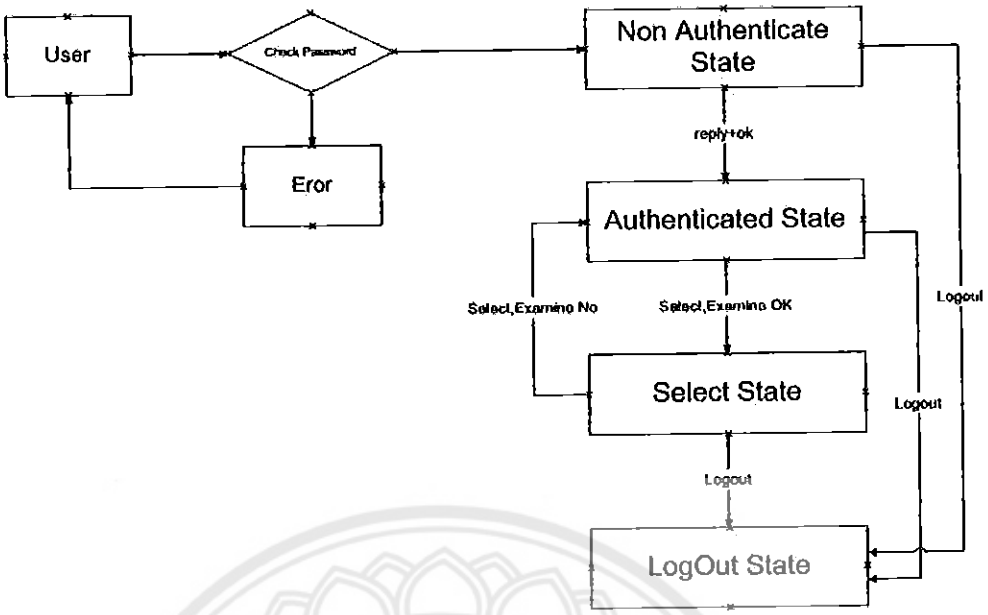
- สถานะก่อนอนุมัติ (Non-authenticated State) เป็นสถานะที่กำลังรอให้ไคลเอนต์ติดต่อเข้ามาเพื่อขออนุมัติใช้ ดังนั้นในด้านไคลเอนต์จะต้องแจ้งชื่อ Login ของ Mail Server นั้นและ password ด้วยคำสั่ง LOGIN หรือ AUTHENTICATE ก่อนจึงจะเริ่มใช้งานได้ จากนั้นจึงเปลี่ยนไปเป็นสถานะได้รับการอนุมัติ

- สถานะได้รับการอนุมัติ (Authenticate State) เป็นสถานะที่สามารถใช้คำสั่ง ต่าง ๆ ที่เกี่ยวข้องกับการเลือกและใช้งาน เมล์บ๊อกซ์ เช่น คำสั่ง SELECT เพื่อเลือกเมล์บ๊อกซ์ หรือคำสั่ง CREATE เพื่อสร้างเมล์บ๊อกซ์ เป็นต้น ในการเลือกเมล์บ๊อกซ์ด้วยคำสั่ง SELECT หรือ EXAMINE นี้จะเป็นการเปลี่ยนไปเป็นสถานะเลือกเมล์บ๊อกซ์

- สถานะเลือกเมลบ๊อกซ์ (Selected State) เป็นสถานะที่จะเข้าไปใช้งานอีเมล์ในแต่ละเมล์บ๊อกซ์ หลังจากที่เลือกเมล์บ๊อกซ์ไว้แล้วในสถานะก่อนหน้านี้

- สถานะเลิกใช้งาน (Logout State) เมื่อต้องการเลิกใช้งาน หรือสิ้นสุดการทำงานของ IMAP จะเข้าสู่สถานะการเลิกใช้งาน โดยคำสั่ง LOGOUT

จากสถานะทั้ง 4 นี้ไม่จำเป็นต้องเรียงกันเสมอไป บางครั้งอาจจะมีการทำงานข้ามจากสถานะหนึ่งไปอีกสถานะหนึ่งได้ ตัวอย่างเช่น เมื่อเข้าสู่สถานะที่ได้รับการอนุมัติ (Authenticate State) และลบอีเมล์ที่ไม่ต้องการใช้งานทิ้งไปด้วยคำสั่ง DELETE แล้ว และไม่ต้องการทำงานอื่นต่อ ก็สามารถใช้คำสั่ง LOGOUTเพื่อเปลี่ยนสถานะเป็นการเลิกใช้งาน (Logout State) ได้โดยไม่ต้องเข้าสู่สถานะการเลือกเมล์บ๊อกซ์ (Select State) ก่อน



รูปที่ 2.4 แผนผังลำดับงาน (Flow Chart) ของ IMAP4

2.6.1 รายละเอียดของคำสั่งต่างๆที่ IMAP4 ใช้อยู่

คำสั่ง : NOOP

สถานะ : ทุกสถานะ

พารามิเตอร์ : ไม่มี

รายละเอียด : เป็นคำสั่ง No Operation และนาการที่เซิร์ฟเวอร์จะถูกตั้งต้นใหม่

คำสั่ง : CAPABILITY

สถานะ : ทุกสถานะ

พารามิเตอร์ : ไม่มี

รายละเอียด : คำสั่งเพื่อตรวจสอบว่าเซิร์ฟเวอร์ใช้โปรโตคอล IMAP ได้

คำสั่ง : LOGOUT

สถานะ : ทุกสถานะ

พารามิเตอร์ : ไม่มี

รายละเอียด : คำสั่งสิ้นสุดการทำงาน

คำสั่ง : AUTHENTICATE

สถานะ : ก่อนอนุมัติ

พารามิเตอร์ : รหัสเพื่อขออนุมัติ

รายละเอียด : คำสั่งที่ใช้เลือกกลไกการรับรอง ประกอบด้วย Kerberos ,V4,S/Key และ GSSAPI

คำสั่ง : LOGIN

สถานะ : ก่อนอนุมัติ

พารามิเตอร์ : ชื่อและ Password

รายละเอียด : คำสั่งเพื่อระบุชื่อผู้ใช้งาน และ Password

คำสั่ง : CREATE

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : คำสั่งสร้างเมล์บ็อกซ์

คำสั่ง : DELETE

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : คำสั่งลบเมล์บ็อกซ์

คำสั่ง : SELECT

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : คำสั่งเลือกเมล์บ็อกซ์

คำสั่ง : EXAMINE

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : คำสั่งเลือกเมล์บ็อกซ์แต่จะเปิดใช้แบบอ่านอย่างเดียว(READ ONLY)

คำสั่ง : RENAME

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์ เดิม/ใหม่

รายละเอียด : คำสั่งเปลี่ยนชื่อเมล์บ็อกซ์

คำสั่ง : SUBSCRIBE

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : คำสั่งเพิ่มชื่อเมล์บ็อกลงในทะเบียน

คำสั่ง : UNSUBSCRIBE

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : ยกเลิกชื่อเมล์บ็อกซ์ออกจากทะเบียน

คำสั่ง : APPEND

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์[FLAG] [วันที่-เวลา]ข้อความ

รายละเอียด : ใช้เพิ่มอีเมล์ลงในเมล์บ็อก

คำสั่ง : LIST

สถานะ : อนุมัติ

พารามิเตอร์ : CONTEXT,ชื่อตู้รับอีเมล์

รายละเอียด : แสดงรายชื่อเมล์บ็อกซ์

คำสั่ง : LSUB

สถานะ : อนุมัติ

พารามิเตอร์ : CONTEXT,ชื่อตู้รับอีเมล์

รายละเอียด : แสดงรายชื่อเมล์บ็อกซ์เฉพาะที่ลงทะเบียนไว้

คำสั่ง : STATUS

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : ตรวจสอบรายละเอียดของเมล์บ็อกซ์

คำสั่ง : FETCH

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : เซ็ตของข้อความ,ชื่อข้อความ

รายละเอียด : อ่านข้อมูลอีเมล์ที่ต้องการ(อ่านได้ทั้งหมด หรือบางส่วน)

คำสั่ง : STORE

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : เซ็ตของข้อความ,ข้อความ

รายละเอียด : ให้ส่งข้อมูลอีเมล์กลับไปอัพเดทที่เซิร์ฟเวอร์

คำสั่ง : CHECK

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : ไม่มี

รายละเอียด : ตรวจสอบสถานะของเมล์บ็อกในขณะนั้น

คำสั่ง : EXPUNG

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : ไม่มี

รายละเอียด : ลบอีเมล์ที่มีเครื่องหมายออกจากเมล์บ็อกซ์

คำสั่ง : SERCH

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : [เซ็ตของตัวอักษร],[เงื่อนไขการค้นหา]

รายละเอียด : ค้นหาอีเมล์ในเมล์บ็อกซ์ที่กำหนด

คำสั่ง : COPY

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : เซ็ตของข้อความ,ชื่อข้อความ

รายละเอียด : คัดลอกอีเมล์ในเมล์บ็อก

คำสั่ง : UID

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : คำสั่ง,พารามิเตอร์ของคำสั่ง

รายละเอียด : คำสั่งแบบแบทช์ คือทำหลาย ๆ คำสั่งต่อเนื่องกันตามที่ระบุไว้ คล้ายกับแบทช์ไฟล์ของ DOS

คำสั่ง : X

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : ขึ้นอยู่กับคำสั่งที่ใช้ร่วม

รายละเอียด : เป็นคำสั่งที่ใช้ทดลองการทำงาน

คำสั่ง : CLOSE

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : ไม่มี

รายละเอียด : ลบอีเมล์ที่มีเครื่องหมายออกจากเมล์บ็อกซ์และเปลี่ยนสถานะเป็นก่อนอนุมัติเพื่อรอรับงานต่อไป

2.7 SMTP (Simple Mail Transfer Protocol)

โปรโตคอลที่คู่กับ POP3 ก็คือ SMTP เพราะเป็นโปรโตคอลที่ใช้ส่งอีเมล์จาก User agent ของผู้ส่งไปยัง MTA ของผู้ส่ง และส่งต่อไปยัง MTA เครื่องอื่น ๆ ที่เป็นจุดผ่านในการเชื่อมต่อไปยังเครื่องของผู้รับ โปรโตคอล SMTP จะทำงานร่วมกับโปรโตคอย TCP โดยใช้พอร์ต 25 ซึ่งคำสั่งต่าง ๆ ของ SMTP จะเป็นลักษณะเดียวกับ POP3 คือ เป็น ASCII ลงท้ายด้วย Carriage Return และ Line Feed ส่วนข้อความที่ตอบกลับมาจะนำหน้าด้วยเลข 3 หลัก เป็นสัญลักษณ์แสดงสถานะการทำงานของคำสั่งที่ได้รับ

เมื่อเริ่มต้นการติดต่อ SMTP จะกำหนดให้ User agent ของผู้ส่งต้องส่งคำสั่ง HELLO พร้อมกับรายละเอียดด้านผู้ส่งออกไป จากนั้นจะส่งคำสั่ง MAIL เพื่อแจ้งให้เซิร์ฟเวอร์เตรียมรับอีเมล์ ในส่วนของเซิร์ฟเวอร์เมื่อพร้อมที่จะรับอีเมล์ก็จะตอบรับกลับมาด้วยคำสั่ง OK จากนั้นที่ด้านส่งก็จะเริ่มส่งโดยใช้คำสั่ง RCPT เพื่อกำหนดอีเมล์แต่ละฉบับที่ส่งไป ซึ่งการส่งข้อมูลของอีเมล์จะถูกระบุด้วยคำสั่ง DATA

การส่งอีเมล์ของโปรโตคอล SMTP ได้จัดเตรียมคำสั่งอื่น ๆ ไว้เพื่ออำนวยความสะดวกและคล่องตัวในการทำงาน ซึ่งประกอบด้วยคำสั่ง VRFY เพื่อให้ด้านที่ส่งตรวจสอบรายชื่อว่าผู้ใช้รายนี้ว่ามีสิทธิ์ใช้งานอีเมล์บ๊อกซ์นั้น ๆ หรือไม่, คำสั่ง EXPN ใช้จัดการและตรวจสอบรายชื่อจากลิสต์รายชื่อ และคำสั่ง TURN ใช้สลับให้ไคลเอนต์ของผู้ส่งทำหน้าที่รับข้อมูลจากเซิร์ฟเวอร์แทน

เมื่อได้รับคำสั่งต่าง ๆ ของผู้ส่งแล้ว เซิร์ฟเวอร์จะมีหน้าที่ตรวจสอบความถูกต้องของคำสั่ง จากนั้นจึงทำงานตามคำสั่งและส่งผลตอบกลับมา ส่วนลักษณะของข้อมูลที่ประกอบด้วยตัวเลขนำหน้าข้อความ 3 หลัก ทำหน้าที่แสดงสถานะการทำงานของเซิร์ฟเวอร์ และเปลี่ยนสถานะการทำงานของโปรโตคอล SMTP ด้วย ถัดจากตัวเลขจะคั่นด้วยช่องว่างแล้วตามด้วยข้อความ ซึ่งปิดท้ายด้วยเครื่องหมาย Carriage Return และ Line Feed ตัวอย่างเช่น 500 Syntax error. Command unrecognized หมายถึงคำสั่งที่ส่งไปไม่ถูกต้อง หรือ 503 Bad sequence of commands หมายถึงลำดับการส่งคำสั่งไม่ถูกต้อง เหล่านี้เป็นต้น

ในการส่งอีเมล์ของโปรโตคอล SMTP นั้น จะใช้วิธีอ้างถึงเซิร์ฟเวอร์อื่น ๆ ตามแบบ DNS หรือ Domain Name System เช่นเดียวกับระบบอื่น ๆ ในอินเตอร์เน็ต และยังสามารถส่งอีเมล์ไปยังผู้รับคนเดียวหรือหลาย ๆ คนพร้อมกันได้ด้วย

2.7.1 กระบวนการขั้นตอนการทำงานของ SMTP (Simple Mail Transfer Protocol)

การติดต่อระหว่างเอ็มทีเอใช้รหัสแอสกีอี วิธีเช่นเดียวกับในเทลเน็ตและเอฟทีพี ไคลเอ็ฯต์จะส่งคำสั่งไปยังเซอร์ฟเวอร์ ส่วนเซอร์ฟเวอร์จะตอบกลับเป็นรหัสตัวเลข โดยอาจมีสายอักขยะขยายความ

การทำงานเริ่มต้นด้วยเอสเอ็มทีพีฝ่ายไคลเอ็นต์ซึ่งเป็นฝ่ายส่งขอสถาปนาการเชื่อมต่อทีซีพีกับเอสเอ็มทีพีฝ่ายเซอร์ฟเวอร์ แต่ก่อนการรับส่งข้อความจะมีกระบวนการแลกเปลี่ยนคำสั่งเพื่อแจ้งผู้รับและผู้ส่งก่อน ในขั้นถัดไปฝ่ายส่งจึงนำส่งข้อความได้ เมื่อสิ้นสุดข้อความจะมีรหัสอักขระส่งปิดท้ายเป็นสัญญาณให้ฝ่ายรับทราบ หลังจากนั้นฝ่ายส่งจึงขอปิดการเชื่อมต่อ

เมื่อไคลเอ็นต์ขอติดต่อไปยังไคลเอ็นต์ เซอร์ฟเวอร์ด้วยคำสั่ง HELO เซอร์ฟเวอร์ให้บริการเอสเอ็มทีพีจะตอบกลับด้วยรหัส 250 ซึ่งจะมีกระบวนการทำงานอยู่ 3 ขั้นตอน ดังนี้

ขั้นตอนแรก

คำสั่งจดหมาย <reverse-path> บรรจุกล่องจดหมายต้นกำเนิด

MAIL < SP >FROM:<reverse-path>< CRLF >

ไคลเอ็นแจ้งเซอร์ฟเวอร์ว่ามีเมล์จากผู้ใช้ patty@nontri.nu.ac.th และทางฝ่ายเซอร์ฟเวอร์ตอบรับตกลง

คำสั่งนี้จะบอก SMTP - เครื่องรับ ว่ารายการจดหมายใหม่จะเริ่มใหม่และจะคืนค่าเนื้อหาและบัพเฟอร์, รวมถึง mail – data ของผู้รับอื่นๆ สถานะของมันทั้งหมดมันจะ reverse-path ไปเป็นส่วนของการรายงานข้อผิดพลาด ถ้ายอมรับ, เครื่องรับ- SMTP รายงานตกลง 250 OK ตอบกลับ

<reverse-path> คือ reverse source routing ของ host และกล่องจดหมายต้นกำเนิด โดยที่ host แรกใน <reverse-path> จะเป็นตัวส่งคำสั่งนี้

ขั้นตอนที่สอง

คำสั่ง RCPT

RCPT < SP >TO:<reverse-path>< CRLF >

ไคลเอ็นต์บอกถึงแอดเดรสของผู้รับเมล์ ถ้ามีผู้รับมากกว่าหนึ่งรายให้ใช้คำสั่ง RCPT สำหรับผู้รับแต่ละราย หากเซอร์ฟเวอร์ตรวจพบว่ามีผู้รับด้านปลายทางก็จะแจ้งตอบตกลงด้วยรหัส 250 หากไม่มีชื่อผู้รับจะแจ้งกลับด้วยรหัส 550 (user unknown)

คำสั่งนี้เป็นการ forward-path ระบุถึงผู้รับ ถ้ายอมรับ, เครื่องรับ - SMTP จะรายงานตกลง 250 OK ตอบกลับ, และเก็บ forward-path ถ้าผู้รับไม่ทราบที่มาเครื่องรับ - SMTP จะรายงานความล้มเหลว 550 Failure ตอบกลับ โดยในขั้นตอนที่สองของกระบวนการทำงานนี้สามารถกระทำซ้ำๆ ได้หลายครั้ง

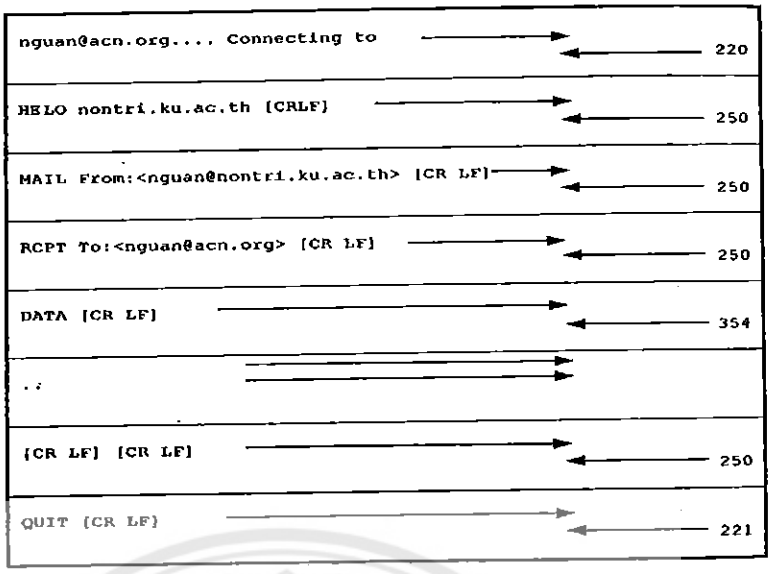
<forward-path> คือ source routing ของ host และกล่องจดหมายปลายทาง โดยที่ host แรกใน <forward-path> ที่เป็นตัวส่งคำสั่งสิ่งนี้

ขั้นตอนที่สาม

คำสั่ง DATAคำสั่ง DATA กำหนดว่าจะนำส่งข้อมูล ทางเซอร์ฟเวอร์จะตอบกลับด้วยรหัส 354 ว่าพร้อมรับข้อความ

ถ้ายอมรับเครื่องรับ- SMTP รายงาน 354 Intermediate ตอบกลับและพิจารณา บรรทัดต่อมาทั้งหมดที่เป็นข้อความข่าวสาร เมื่อจบข้อความแล้ว จะรับและเก็บ จากนั้น SMTP - เครื่องรับ จะส่ง 250 OK ตอบกลับ ไคลเอ็นต์จบการทำงานโดยส่งคำสั่ง QUIT และเซอร์ฟเวอร์ตอบกลับว่าได้ปิดการเชื่อมต่อแล้ว

หากพิจารณาถึงรายละเอียดการส่งคำสั่งและข้อมูลจากไคลเอ็นต์ไปยังเซิฟร์เวอร์ คำสั่งทุกคำสั่งจะปิดท้ายด้วยรหัส CR LF ข้อความที่ส่งหลังจากคำสั่ง DATA จะปิดท้ายด้วย CR LF เช่นกัน ส่วนเครื่องหมายจุดที่พิมพ์ปิดท้ายเป็นสัญลักษณ์บอกให้ยูเอส่งรหัส CR LF และ CR LF ติดกันเพื่อแจ้งว่าข้อความสิ้นสุดแล้ว



รูปที่ 2.5 ตัวอย่างการส่งและโต้ตอบเมล

ตารางที่ 2.2 แสดงรายละเอียดในคำสั่งต่าง ๆ ของ STMP ที่ใช้งานอยู่

คำสั่ง	รายละเอียด
HELLO	ใช้เมื่อไคลเอนต์ของอีเมล์ต้องการติดต่อเซิร์ฟเวอร์
MAIL	เริ่มเข้าสู่สถานะการส่งอีเมล์
RCPT	เป็นคำสั่งเพื่อระบุอีเมล์ที่จะส่งทีละฉบับ โดยเป็นคำสั่งที่ใช้ต่อจาก MAIL
DATA	จะเป็นคำสั่งที่ใช่ต่อจาก RCPT เพื่อส่งข้อมูลของอีเมล์
SEND	ทำหน้าที่เหมือนคำสั่ง DATA แต่ไม่ค่อยมีผู้ใช้งาน
SOML	ทำหน้าที่เหมือนคำสั่ง DATA แต่ไม่ค่อยมีผู้ใช้งาน
SAML	ทำหน้าที่เหมือนคำสั่ง DATA แต่ไม่ค่อยมีผู้ใช้งาน
VRFY	เป็นคำสั่งที่ใช้เพื่อตรวจสอบความถูกต้องของชื่อและเมล์บ๊อกซ์
EXPN	เป็นคำสั่งเพื่อตรวจสอบรายละเอียดของลิสต์รายชื่อ
HELP	ใช้ตรวจสอบคำสั่งที่สามารถใช้งานได้กับเซิร์ฟเวอร์
NOOP	เป็นคำสั่ง No Operation เมื่อเซิร์ฟเวอร์ได้รับคำสั่งนี้ จะตอบ OK กลับมา
QUIT	สิ้นสุดการติดต่อ
RSET	ยกเลิกการส่งข้อมูลในขณะนี้
TURN	เป็นคำสั่งสลับหน้าที่ของผู้ส่งข้อมูลมาทำหน้าที่รับข้อมูลแทน

2.7.2 FORWARDING

มีบางกรณีที่ข้อมูลปลายทางใน <forward-path> ไม่ถูกต้อง, แต่เครื่องรับ- SMTP รู้ปลายทางที่ถูกต้อง ในกรณีนั้น, จะมีการตอบกลับมาว่าให้ใส่ผู้ส่งติดต่อปลายทางที่ถูกต้อง

251 User not local;will forward to <forward-path>

ตัวที่ตอบกลับมานี้จะบอกว่าเครื่องรับ- SMTP รู้ว่ากล่องจดหมายของผู้ใช้อยู่บน host อื่นๆและบอก forward-path ที่ถูกต้องที่จะใช้ในอนาคต

551 User not local ; please try <forward-path>

สิ่งนี้ตอบกลับชี้บอกว่าเครื่องรับ- SMTP รู้กล่องจดหมายของผู้ใช้อยู่บน host อื่นๆ และจะชี้บอก forward-path ที่จะใช้ เครื่องรับจะปฏิเสธการรับจดหมายนี้, และผู้ส่งต้องส่งตามข้อมูลภายใต้เงื่อนไขหรือบอกข้อผิดพลาดให้ผู้ส่งทราบ

2.7.3 VERIFYING AND EXPANDING

SMTP มีความสามารถเพิ่มเติม คือ คำสั่งพิสูจน์ว่าเป็นผู้ใช้หรือขยายรายชื่อที่จดหมาย ซึ่งจะใช้คำสั่ง VRFY และ EXPN ซึ่งจะเป็นข้อความสตริงตัวอักษร สำหรับคำสั่ง VRFY สตริงคือชื่อผู้ใช้และการตอบสนองรวมถึงชื่อเต็มของผู้ใช้และรวมถึงกล่องจดหมายของผู้ใช้ สำหรับคำสั่ง EXPN , สตริงระบุรายชื่อที่จดหมาย, และการตอบสนอง multiline จะรวมถึงชื่อเต็มของผู้ใช้และบนจดหมายในกล่องจดหมายรายชื่อ

"User name" คือ fuzzy เทอม ถูกใช้เป็นจุดประสงค์ ถ้า host เพิ่ม VRFY หรือ EXPN คำสั่งต่อมาอย่างน้อยที่สุดกล่องจดหมายภายในจะถูกจำ เนื่องจากว่า "user names" ถ้า host เลือกที่จะจำสตริงอื่น ๆ เป็น"user name" นั่นคือถูกอนุญาติ

ใน host จะมีความแตกต่างระหว่างที่จดหมายแสดงรายการและนามแฝงสำหรับกล่องจดหมายเดี่ยวคือ a bit fuzzy , เพราะว่าโครงสร้างข้อมูลธรรมดาจะรับเอาชนิดทั้งสองของรายการที่จดไว้, และจะมีจดหมายที่แสดงรายการของหนึ่งกล่องจดหมาย ถ้าความต้องการตรวจสอบที่จดหมายแสดงรายการตอบสนองที่เป็นบวก สามารถที่จะให้ได้ ถ้าในการรับของข่าวสารตำแหน่งที่อยู่มันจะถูกส่งให้ให้ทุกๆคนบนรายชื่อ, ไม่เช่นนั้นข้อผิดพลาดควรจะถูกรายงาน(e.g .," 550 That is a mailing list,not a user") ถ้าความต้องการถูกทำเพื่อที่จะขยาย user name ตอบสนองจะเป็นบวกแล้วจะถูกสร้างขึ้นโดยการคืนที่บรรจุรายชื่อ 1 ชื่อ, หรือข้อผิดพลาดสามารถถูกรายงาน(e.g .,"550 That is a mailing list,not a mailing list")

ในกรณีของ Multiline reply (มาตรฐานสำหรับ EXPN) หนึ่งกล่องจดหมายจะถูกเจาะจงบนเส้นแต่ละอันของการตอบกลับ ในกรณีของความต้องการคลุมเครือ, เป็นต้นว่า, " VRFY Smith", ซึ่งมี Smith's อยู่ 2 ที่ การตอบสนอง คือ" 553 User ambiguous"

2.7.4 SENDING AND MAILING

จุดประสงค์หลักของ SMTP คือจะส่งข่าวสารให้กล่องจดหมายของผู้ใช้ โดย host จำนวนหนึ่งจะส่งข่าวสารไปให้สถานีปลายทางของผู้ใช้ (ภายใต้เงื่อนไขผู้ใช้ทำงานบน host) การส่งให้กล่องจดหมายของผู้ใช้ถูกเรียกว่า "mailing", การส่งให้สถานีปลายทางของผู้ใช้ถูกเรียก" sending" โดย host ส่วนมากจะยอมให้ผู้ใช้ยอมรับหรือปฏิเสธข่าวสารนั้นได้

ดังต่อไปนี้สามคำสั่งถูกจำกัดความเพื่อสนับสนุนออพชั่นที่ส่ง ซึ่งถูกใช้ในรายการจดหมายแทนที่คำสั่งจดหมายและแจ้งบอกเครื่องรับ- SMTP ความหมายพิเศษของรายการนี้:

SEND <SP> FROM:<reverse-path> <CRLF>

คำสั่งการส่ง ข้อมูลจดหมายจะถูกส่งไปให้สถานีปลายทางของผู้ใช้ ถ้าผู้ใช้ไม่มีตัวตน(หรือไม่ยอมรับข่าวสารที่ปลายทาง) ใน host 450 ตอบกลับอาจจะคืนสู่คำสั่ง RCPT รายการจดหมายจะสมบูรณ์ถ้าข่าวสารถูกส่งให้สถานีปลายทาง

SOML < SP >FROM:<reverse-path>< CRLF >

การส่งคำสั่งจดหมาย ต้องการว่าข้อมูลจดหมายถูกส่งให้สถานีปลายทางของผู้ใช้ถ้าผู้ใช้มีการใช้งานหรือ(ยอมรับข่าวสารสถานีปลายทาง) บน host ถ้าผู้ใช้ไม่มีตัวตนใน(หรือไม่ยอมรับข่าวสารที่ปลายทาง) ข้อมูลจดหมายจะถูกใส่ไปไว้ในกล่องจดหมายของผู้ใช้ รายการจดหมายสมบูรณ์จะถ้าข่าวสารถูกส่งให้หนึ่งในสถานีปลายทางหรือกล่องจดหมาย

SAML < SP > FROM:<reverse-path>< CRLF >

การส่งคำสั่งจดหมาย ต้องการว่า ข้อมูลจดหมายจะถูกส่งให้ถึงสถานีปลายทางของผู้ใช้ถ้าผู้ใช้มีการใช้งานหรือ(ยอมรับข่าวสารสถานีปลายทาง) บน host ในกรณีใดๆ ข้อมูลจดหมายจะใส่เข้าไปในกล่องจดหมายของผู้ใช้ รายการจดหมายจะสมบูรณ์ถ้าข่าวสารถูกส่งให้กล่องจดหมาย

2.6.5 OPENING AND CLOSING

ช่วงหนึ่งของช่องสัญญาณจะมีการการส่งผ่านเพื่อแลกเปลี่ยนข้อมูล โดย host กำลังติดต่อกับ host ที่ถูกต้อง

จะมีสองคำสั่งถูกใช้ในการเปิดช่องสัญญาณการส่งผ่านและการสิ้นสุด:

HELO < SP ><domain>< CRLF >

QUIT < CRLF >

ในคำสั่ง HELO ที่ส่ง host จะระบุตัวเองและคำสั่งอาจจะอธิบายด้วยคำพูดว่า "Hello, I am <domain>".

2.7.6 เมล์รีเลย์

เมื่อผู้ใช้ส่งจดหมาย หน้าที่ของยูเอคือส่งจดหมายไปยังเอ็มทีเอเพื่อให้เอ็มทีเอนำส่งต่อไป เอ็มทีเอต้นทางอาจติดต่อกับเอ็มทีเอปลายทางโดยตรง หรือใช้วิธี รีเลย์ (relay) โดยส่งต่อเป็นทอด คือ จากเอ็มทีเอต้นทางอาจติดต่อกับเอ็มทีเอระหว่างทางซึ่งจะเก็บเมลไว้และนำส่งต่อตามจังหวะเวลาที่เหมาะสมจนกระทั่งเมลไปถึงปลายทาง ระบบเมล์ที่ใช้วิธีส่งต่อเป็นทอด ๆ นี้เรียกว่า ระบบเก็บและส่งต่อ (store-and-forward systems) เมล์รีเลย์ประจำโดเมนหนึ่ง ๆ เรียกว่า ตัวแลกเปลี่ยนเมล์ (mail exchanger) ซึ่งกำหนดในดีเอ็นเอสด้วยเรคอร์ด MX การใช้เมล์รีเลย์มีข้อดีหลายประการ เช่น

- ผู้ใช้สถานีงานขนาดเล็กหรือพีซีที่ไม่มีเอ็มทีเอมักไม่ได้เปิดเครื่องใช้งานอยู่ตลอดเวลา เมื่อมีเมล์เข้ามาจำเป็นต้องอาศัยเมล์รีเลย์เป็นตัวเก็บพักเมล์ไว้จนกว่าจะเปิดใช้พีซีเพื่อขอถ่ายเมล์มาจากเมล์รีเลย์

- เครือข่ายในหลายองค์กรใช้เมล์รีเลย์ทำหน้าที่ติดต่อกับเครือข่ายภายนอกเมล์รีเลย์อาจเป็นจุดเดียวที่อนุญาตให้รับส่งเมล์โดยตรงกับภายนอกได้ โดยมีระบบไฟร์วอล์ห้ามเครื่องอื่นภายในเครือข่ายรับส่งเมล์โดยตรงเพื่อสร้างระบบเมลศูนย์กลางและไม่ให้ชื่อเครื่องอื่นในเครือข่ายแพร่ออกไปภายนอก

- การติดตั้งเอ็มทีเออย่างเช่น sendmail ในยูนิกซ์มีความซับซ้อน ผู้ดูแลระบบบางแห่งจะไม่ติดตั้งเอ็มทีเอกระจายไปทั่ว แต่ให้ใช้บริการผ่านเมล์รีเลย์แทน

forward-path จะเป็นเส้นทางต้นกำเนิดของรูปแบบ"@ONE,@ TWO:JOE@THREE ",ซึ่ง ONE, TWO, และ THREE คือ host รูปแบบนี้ที่ถูกใช้เพื่อเน้นความแตกต่างระหว่างตำแหน่งที่อยู่และเส้นทาง กล่องจดหมายจะเป็นตำแหน่งที่อยู่สมบูรณ์, และเส้นทางคือคำแนะนำเกี่ยวกับวิธีได้ที่นั่น สองความคิดจะต้องไม่ทำให้สับสน

ส่วนประกอบของการ forward-path จะถูกเคลื่อนย้ายสู่การ reverse-path ซึ่งข่าวสารที่ถูกถ่ายทอดจากหนึ่งเครื่องแม่ข่าย- SMTP ถึงที่อื่นๆ การ reverse-path คือการกลับหลังของเส้นทางต้นกำเนิด, (เช่น เส้นทางต้นกำเนิดจากตำแหน่งปัจจุบันของข่าวสารถึง originator ของข่าวสาร) เมื่อเครื่องแม่ข่าย- SMTP ลบผู้ค้นหาของมันจากการ forward-path และใส่มันเข้าไปในการ reverse-path , มันต้องใช้ชื่อที่มันรู้โดยสิ่งแวดล้อมที่มันกำลังส่งเข้าไป, ไม่ใช่สิ่งแวดล้อมที่จดหมายมา ในกรณีของเครื่องแม่ข่าย- SMTP ถูกรู้โดยชื่อที่แตกต่างในสิ่งแวดล้อมแตกต่างกัน

เมื่อข่าวสารมาถึง SMTP ส่วนประกอบแรกของการ forward-path คือไม่มีการค้นหาของ SMTP ส่วนประกอบไม่ถูกลบจากการ forward-path และถูกใช้เพื่อตัดสินใจ SMTP ถัดไปที่จะส่งข่าวสารถึง ในกรณีใดๆ SMTP เพิ่มผู้ค้นหาด้วยตัวเองของมันให้การ reverse-path

การใช้ที่กำเนิดเครื่องรับ- SMTP รับจดหมายถูกถ่ายทอดให้เครื่องแม่ข่ายอื่นๆ- SMTP เครื่องรับ- SMTP จะยอมรับหรือปฏิเสธงานของการถ่ายทอดจดหมาย เช่นเดียวกันมันยอมรับหรือปฏิเสธ จดหมายสำหรับผู้ใช้ภายใน เครื่องรับ- SMTPจะเปลี่ยนรูปแบบข้อความคำสั่งโดยเคลื่อนที่ผู้ค้นหาด้วยตัวเองของมันจากการ forward-path ถึงการเริ่มของการ reverse-path เครื่องรับ- SMTP

ต่อมาจะกลายเป็นผู้ส่ง-SMTP, ช่องสัญญาณจะถูกตั้งขึ้นโดยทำการส่งผ่านให้ SMTP ถัดไปในการ forward-path , และส่งจดหมาย

host ตัวแรกในการ reverse-path ควรจะที่ส่งคำสั่ง Host ให้ SMTP และ host แรกในการ forward-path ควรจะที่รับ host คำสั่ง SMTP

การ forward-path และ reverse-path จะปรากฏในคำสั่ง SMTP และตอบกลับ, แต่ไม่จำเป็นในข่าวสาร กล่าวคือ สิ่งนี้ไม่ต้องการสำหรับเส้นทางเหล่านี้และโดยเฉพาะข้อความนี้ที่จะปรากฏใน "To:","From:"," CC :", อื่นๆ ในส่วนพื้นที่ของหัวข้อข่าวสาร

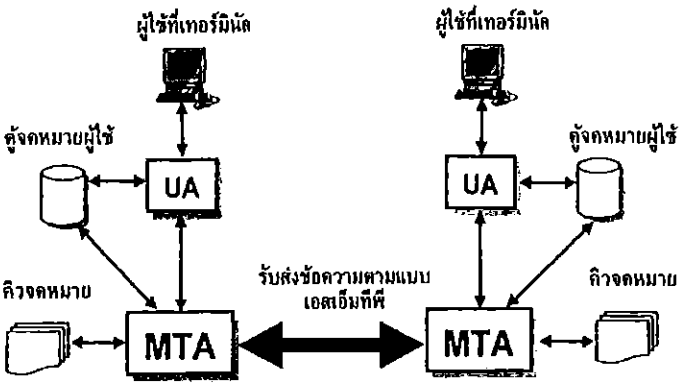
ถ้าเครื่องแม่ข่าย- SMTP ยอมรับงานของการถ่ายทอดจดหมายและต่อมาพบว่าการ forward-path ไม่ถูกต้อง หรือ จดหมายไม่สามารถถูกส่งให้ได้สำหรับเหตุผลอะไรก็ตาม, ต่อมามันต้องสร้าง"จดหมายไม่ส่งให้ "ข่าวสารการประกาศและส่งมันให้ originator ของจดหมายไม่ส่งให้ (ตามที่ที่ชี้บอกโดยการ reverser -path).

ข่าวสารการประกาศนี้จำเป็นต้องสร้างจากเครื่องแม่ข่าย- SMTP ด้วยเหตุนี้ host แน่นอน, เครื่องแม่ข่าย- SMTP ควรจะไม่ส่งข่าวสารให้การประกาศเกี่ยวกับปัญหากับข่าวสารการประกาศหนึ่งทางที่จะป้องกันวนซ้ำในที่รายงานข้อผิดพลาด คือ เจาะจง reverse-path null ในคำสั่งจดหมายของข่าวสารการประกาศ เมื่อข่าวสารถูกถ่ายทอด มันจะยินยอมที่จะออกจากการ reverse-path null คำสั่งจดหมายเกี่ยวกับ null reverse-path ปรากฏดังต่อไปนี้:

MAIL FROM:< >

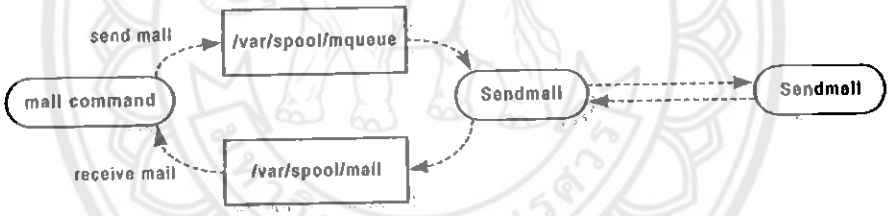
2.7.7 ยูนิกซ์เมล์

ยูนิกซ์เมล์มีแบบทำงานเช่นเดียวกับระบบเมล์ที่กล่าวมาแล้ว เมื่อยูเอนำส่งเมล์เอ็มทีเออาจส่งเมล์ออกไปทันทีหรืออาจ เก็บพัก (spool) ไว้ในหน่วยความจำสำรองก่อนเพื่อรอการนำส่ง (เช่นใน /var/spoolmqueue) การเก็บพักช่วยให้เอ็มทีเอจัดลำดับและบริหารการส่งเมลได้อย่างเป็นระบบ โดยปกติแล้วเอ็มทีเอจะส่งเมล์ไปยังปลายทางทันที แต่ถ้าการส่งล้มเหลว เอ็มทีเอจะจัดเก็บเมล์ไว้ในคิวเพื่อรอส่งใหม่ และควรทดลองส่งซ้ำเป็นช่วงอย่างน้อยช่วงละ 30 นาทีจนกว่าจะส่งได้หรือยกเลิกการส่ง เพราะปัญหาปลายทางไม่สามารถรับเมล์ได้มักเกิดขึ้นเพียงช่วงสั้น ๆ กรณียกเลิกการส่งควรพยายามส่งมาแล้วไม่น้อยกว่า 4-5 วัน



รูปที่ 2.6 แสดงโครงสร้างของยูนิกซ์เมล

ผู้ส่งเรียกใช้เมล์ผ่านทางยูเอ เช่น pine หรือ mail และเอ็มทีเอฝ่ายส่งติดต่อกับเอ็มทีเอฝ่ายรับผ่านทีซีพีพอร์ต 25 จดหมายที่ไปถึงปลายทางจะถูกเก็บอยู่ในตู้ไปรษณีย์ประจำตัวผู้ใช้หรือ เมล์บ็อกซ์ (mailbox) ในยูนิกซ์เก็บเมล์บ็อกในรูปแฟ้มข้อมูล ผู้ใช้แต่ละรายจะมีแฟ้มนี้เป็นของตนเอง (มักอยู่ใน /var/mail หรือ var/spool/mail)



รูปที่ 2.7 แสดงการรับส่งเมล์ในระบบยูนิกซ์

2.7.8 รูปแบบจดหมาย

โครงสร้างของเมล์ประกอบด้วยส่วนสำคัญสองส่วนคือ หัวจดหมาย และ เนื้อความ หัวจดหมายซึ่งอยู่ส่วนต้นประกอบด้วยข้อความแสดงข่าวสารเกี่ยวกับการรับส่งจดหมาย ส่วนเนื้อความคือข้อความที่ได้รับซึ่งจะอยู่ต่อจากหัวจดหมาย เอสเอ็มทีพีกำหนดรูปแบบของหัวจดหมายไว้เพื่อให้ใช้เป็นหลักมาตรฐาน หัวจดหมายแต่ละฉบับอาจมีรายละเอียดมากน้อยต่างกันไป

ตัวอย่างต่อไปนี้เป็นจดหมายที่มีหัวจดหมายสั้นๆ ที่มีเพียงรายละเอียดบ่งบอกว่าใครเป็นผู้ส่ง (From:) เมื่อเวลาใด (Date:) ส่งถึงใคร (To :) ใครบ้างที่ได้รับสำเนาจดหมาย (Cc:) และหัวเรื่องจดหมาย (Subject:)

```
Received: from master.cpe.ku.ac.th ([158.108.33.252]) by ku.ac.th
(8.9.0/8.9.0) with ESMTP id LAA11556; Thu, 30 Jul 1998 11:05:11
+0700 (GMT)
Date: Thu, 30 Jul 1998 10:59:32 +0700
From: Sasinee Panyarat <ann@master.cpe.ku.ac.th>
To: sk@ku.ac.th, nguan@ku.ac.th
Subject: msit/mcpe meeting

Dear all Aj.,
We have a meeting on 4 Aug. 13.00 at the meeeing room.
CU
Ann
```

จดหมายบางฉบับอาจมีรายละเอียดอื่นอีกเช่น เส้นทางลำเลียงหรือลักษณะพิเศษของตัวจดหมาย ดังหัวจดหมายต่อไปนี้

```
Received: from mcb.mcb.co.uk (mcb.mcb.co.uk [193.130.114.132]) by
nontri.ku.ac.th (8.8.2/8.7.3) with SMTP id UAA26572 for
<nguan@ku.ac.th>; Mon, 9 Mar 1998 20:36:12 +0700 (GMT)
Received: from pc0170.mcb.co.uk by mcb.mcb.co.uk (SMI-8.6/PIPEX simple
1.22) id NAA06171; Mon, 9 Mar 1998 13:36:52 GMT Message-ID:
<3.0.2.16.19980309105130.22b77858@mcb.mcb.co.uk>
X-Sender: ckeenan@mcb.mcb.co.uk
X-Mailer: QUALCOMM Windows Eudora Light Version 3.0.2 (16)
Date: Mon, 09 Mar 1998 10:51:30
To: nguan@ku.ac.th
From: Chris Keenan <ckeenan@mcb.co.uk>
Subject: Internet Research
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
Status:
X-Mozilla-Status: 8001

Internet Research
View the journal homepage at: http://www.mcb.co.uk/intr.htm
Dear Surasak,

Thank-you for your interest in Internet Research, I hope you found you
free trial both informative and beneficial.
```

หัวจดหมายแต่ละฉบับอาจมีความแตกต่างกันไปตามระบบและโปรแกรมที่ใช้งานแต่โดยทั่วไปจะอาศัยหลักการที่คล้ายคลึงกัน จากตัวอย่างข้างต้นแต่ละส่วนของจดหมายมีความหมายดังต่อไปนี้

```
Received: from mcb.mcb.co.uk (mcb.mcb.co.uk [193.130.114.132]) by
nontri.ku.ac.th (8.8.2/8.7.3) with SMTP id UAA26572 for
<nguan@ku.ac.th>; Mon, 9 Mar 1998 20:36:12 +0700 (GMT)
```

บรรทัดที่ขึ้นต้นด้วยคำว่า Received: เป็นชื่อเครื่องที่เป็นตัวกลางส่งจดหมาย การเดินทางของจดหมายแต่ละครั้งอาจต้องอาศัยการส่งต่อเป็นช่วง ๆ ลำดับชื่อที่แสดงจะเรียงกลับจากผู้รับย้อนไปหาผู้ส่ง

ข้อมูลข้างต้นบอกการติดต่อระหว่างโฮสต์ล่าสุดที่รับส่งจดหมายระหว่างกัน ในที่นี้คือ nontri.ku.ac.th ข้อมูลส่วนอื่นได้แก่วันเวลาที่โฮสต์ได้รับจดหมาย เวลาที่ปรากฏอยู่ถือว่าเป็นเวลา

```
Received: from pc0170.mcb.co.uk by mcb.mcb.co.uk (SMI-8.6/PIPEX simple
1.22) id NAA06171; Mon, 9 Mar 1998 13:36:52 GMT Message-ID:
<3.0.2.16.19980309105130.22b77858@mcb.mcb.co.uk>
```

ท้องถิ่นของโฮสต์ที่รับจดหมายเทียบกับเวลามาตรฐานกรีนิช ตัวเลข +0700 บ่งบอกว่าโฮสต์ nontri.ku.ac.th มีเวลาเร็วกว่ามาตรฐานกรีนิช 7 ชั่วโมง

2.7.9 โดเมน

เมื่อไม่นานความคิดนี้ได้รับการแนะนำในระบบจดหมายอินเตอร์เน็ต ARPA การใช้ของโดเมนเปลี่ยนที่ว่างตำแหน่งที่อยู่จากที่ว่างทั่วโลกบน host สตริงตัวอักษรง่ายดายตั้งชื่อถึงต้นไม้ที่รากที่โครงสร้างเกี่ยวกับหรือมีลักษณะของ hierar ของตำแหน่งที่อยู่ทั่วโลก ชื่อ host ถูกเคลื่อนย้ายโดยโดเมนและ host designator สิ่งที่มีคือตามลำดับของสตริงส่วนประกอบโดเมนที่แยกเมื่อถึงระยะเวลากับเกี่ยวกับความเข้าใจ ส่วนประกอบโดเมนถูกสั่ง most specific ถึง most general

เป็นต้นว่า, " USC - ISIF.ARPA "," Fred.Cambridge.UK ",และ" PC7.LCS.MIT.ARPA " อาจจะเจ้าของบ้าน-และ-ผู้ค้นหาโดเมน

ชื่อโดเมนที่ถูกใช้ใน SMTP จะเป็นชื่อทางการเท่านั้น การใช้ของชื่อเล่นหรือนามแฝงจะไม่ยอมให้ใช้

2.8 การสร้างซ๊อกเก็ต

ในการเชื่อมต่อระหว่างไคลเอนต์กับเซิร์ฟเวอร์เพื่อจะสามารถรับและส่งข้อมูลได้นั้น ไคลเอนต์จะต้องรู้ตำแหน่งและรู้จักซ็อกเก็ตของเซิร์ฟเวอร์ และเซิร์ฟเวอร์จะต้องตั้งชื่อซ๊อกเก็ตของตัวเองเพื่อให้ไคล์เอนต์สามารถใช้อ้างอิงได้ ชื่อของซ๊อกเก็ต จะสอดคล้องกับ ไอพีแอดเดรส (IP Address) และหมายเลขพอร์ต (Port Number)

2.8.1 โครงสร้างของการสร้างซ๊อกเกต

เราจะเริ่มต้นจากการสรุปโครงสร้างคำสั่งทั้งหมดของ เซิร์ฟเวอร์คลาสเสียก่อน ดังนี้

```
import java.io.*;
import java.net.*;
public class TCPServer implements Runnable{
    Thread serverThread = null;
    Int port = 5000;
    // Constructor
    // runnable/Thead methods
    // run( ) Method
}
```

2.8.2 การสร้างซ๊อกเก็ต (Constructor)

จะต้องสร้าง 1 อากูเม้น: เป็นหมายเลข เพื่อรอการติดต่อเข้ามาซึ่งได้จะต้องพยายามจัดสรรให้พอร์ตServerSocket แต่ถ้าหากไม่สามารถจัดสรรให้ได้ก็จะแจ้ง error ขึ้น เช่น

```
public TCPServer (int p) throws IOException{
    port = p ;
    try{
        s = new ServerSocket (port);
    } catch( IOException e) {
    System.err.printli("Exception allocating ServerSocket: "+e);
    Throw e;
}
```

2.8.3 วิธีการรันผล

จะต้องมี strart server และมี stop server

```
start server
public synchronized void start() {
if (serverThread == null) {
    serverThread = new Thread (this);
    serverThread.setPriority (Thread.MAX_PRIORITY/4);
    serverThread.Start();
```

```
}
if(s == null){
    try{
    s = new ServerSock (port);
} catch(IOException e){
     System.err.println("Exception allocating ServerSocket: " + e );
 Return; }}}
stop server
public synchronize void stop(){
      if (serverThread != null){
    serverThread.stop();
    serverThread = null;
 }
 if (s != null) {
     try {
     s.close();
     s.null;
 } catch(IOException e){
     System.err.println( "Exception closing ServerSocket: " + e );
 Return;
     }
   }
 }
```

เราจะมี method join เพื่อ block thread ที่ใช้เรียก จนกว่าการทำงาน serverThread จะเสร็จ ถ้ามีการ interrupt เกิดขึ้นก่อนที่จะทำงานเสร็จ ก็จะไปที่ method join แต่ถ้าไม่มี thread ไหนรอเลย method นี้ก็จะ return ค่ากลับไป

```
public final void join() throws InterruptedException{
     if ( serverThread != null) {
     serverThread.join();
  }
  retrun;
  }
```

run() Method

ถ้า เซิร์ฟเวอร์ ทำงานได้ ก็จะทำที่ run() method

```
public void run(){
    InputStream in = null;
    PrintStream out = null;
    Socket con = null;
While (serverThread != null) {
//Wait for and incoming connection
//Get I/O Streams for the Socket
//Talk to the client
}
// Close the ServerSocket
}
```

2.8.4 รอการติดต่อเข้ามา

เมื่อเซิร์ฟเวอร์ทำงานแล้ว ก็จะรอไคลเอนต์ติดต่อเข้ามาที่พอร์ตที่เรากำหนดไว้ ถ้าติดต่อพอร์ตได้สำเร็จเราก็จะแจ้งผลไป ถ้าติดต่อไม่ได้เราก็จะแสดงผลบอกว่าผิดพลาดกลับไป

```
try {
        con = s.accept();
} catch( IOException e){
    System.err.println("Error on accept:" + e );
    return;
}
System.err.println ("Got connection from" + con.getInetAddress() + ":" + con.getPort());
```

2.8.5 นำ I/O ติดต่อกับซ็อกเก็ต

ตอนนี้เรามีคนที่จะติดต่อเข้ามาแล้ว server ต้องการที่จะส่งและรับข้อมูลไปที่ไคลเอนต์ เราก็จะใช้ InputStream เป็นตัวรับข้อมูลจนจบ และ a PrintStream เป็นตัวตอบกลับไปที่ client

```
try {
    out = new PrintStream(con.getOutputStream() );
    in = con.getInputStream();
} catch(Exception e) {
    System.err.println ("Error buiding straams: " + e);
```

2.8.6 การตอบกลับไปที่ไคลเอนต์

เราต้องการเริ่มส่งข้อความตอบกลับไปที่ไคลเอนต์จะต้องมีส่วนจบการติดต่อ

out.println("Hi there! Enter 'bye' to exit, 'DIE!' to stop server.:");

ถ้าเราได้รับข้อความที่ไคลเอนต์ส่งมาให้เราก็จะแสดงว่าเราได้รับข้อความนั้นแล้วโดยแสดงข้อความนั้นออกมา แล้วก็จะเช็คว่า มีข้อความจบไหม ถ้ามี flag ก็จะเป็น true และจบการทำงาน

```
try{
    int nbytes;
   boolean done = false;
    byte b[] = new byte[1024];
while (!done && ((nbytes = in.read (b,0,1024)) ! = -1){
    String str = new String (b,0,0,nbytes);
    Out.println ("Receive: \n" + str);
If (str.trim().compareTo("bye") ==0){
    System.err.printly ("Got by. Closing Connection.");
   Done = true;
}
if (str.trim().compareTo("DIE!") ==0) {
    System.err.println ("Exiting.");
    Stop();
    Retrun;
}
out.println("Bye!");
out.flush();
}
}
catch(Exception e) {
System.err.println ("Error reading :" + e);
}
```

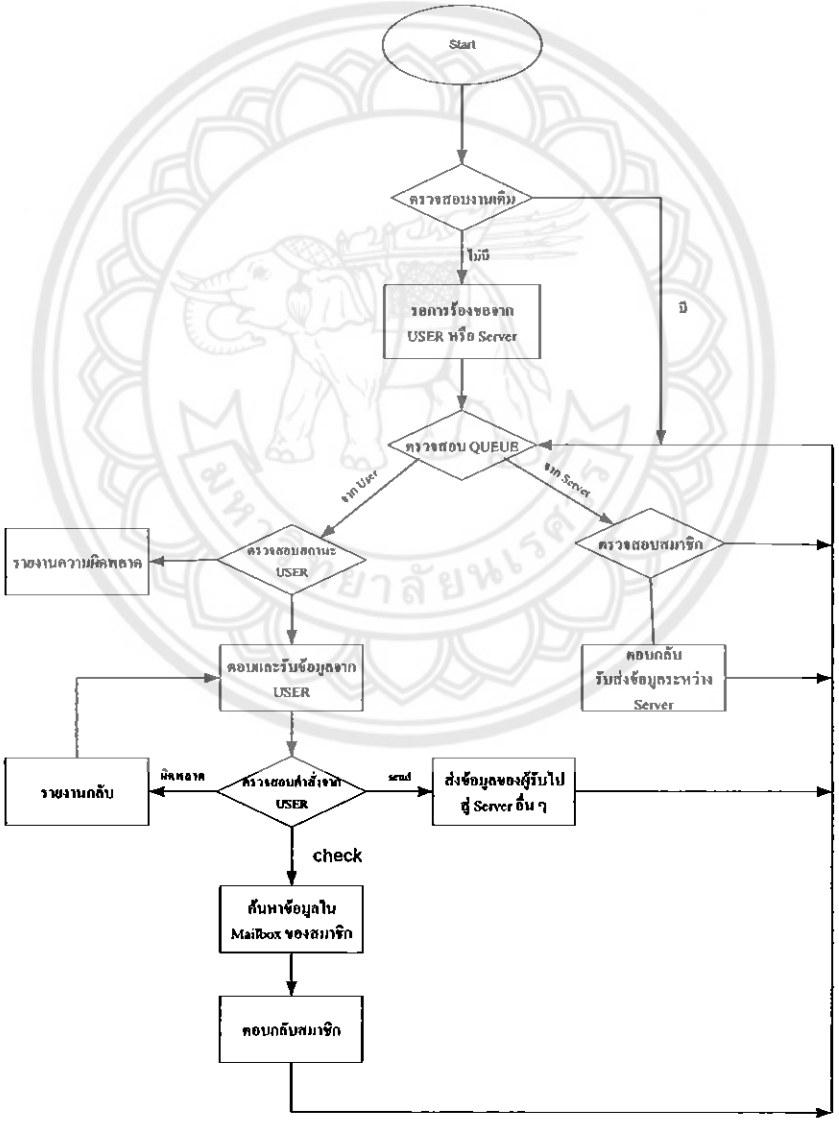
ปิด ServerSocket

ก่อนที่เราจะรับการติดต่ออีกครั้ง เราจะปิดซ๊อกเก็ตที่ติดต่อในขณะนี้เสียก่อน แล้วจึงเริ่มใหม่อีกครั้ง

บทที่ 3
ออกแบบและพัฒนาโปรแกรม

เมื่อได้ทำการศึกษาหลักการและทฤษฎีที่เกี่ยวข้องแล้ว ผู้จัดทำก็ได้ทำการออกแบบและพัฒนาโปรแกรม ได้ดังนี้

3.1 ขั้นตอนการออกแบบโปรแกรม และโครงสร้างของโปรแกรม



รูปที่ 3.1 แผนผังลำดับงาน (Flow Chart) ของโปรแกรม

3.1.1 ลำดับการทำงานของโปรแกรม

หลังจากได้ศึกษาและรวบรวมข้อมูล กำหนดขอบเขตของการพัฒนาโปรแกรมเสร็จสิ้นแล้ว ก็นำความรู้ที่ได้มาออกแบบโปรแกรมที่จะพัฒนาขึ้น โดยการทำงานและโครงสร้างของโปรแกรม อธิบายได้ดังนี้

เริ่มต้นโปรแกรม

3.1.1.1 ตรวจสอบงานเดิมว่ายังมีเหลือค้างหรือไม่

- ถ้ายังมีงานเดิมค้างอยู่ให้ไปตรวจสอบลำดับของ Queue
- ถ้าไม่มีงานเดิมค้างให้ไปรอการร้องขอ User หรือ Server เมื่อมีการร้องขอมาก็จะไปตรวจสอบ Queue ว่าใครมาก่อนได้รับการบริการก่อน

3.1.1.2 เมื่อมีการตรวจสอบ Queue ก็จะทำการตอบสนองการร้องขอแรกสุดที่อยู่ใน Queue ก่อนว่าใครเข้าใช้บริการก่อน ทั้งจาก Server และ User

3.1.1.3 ถ้า User อยู่ในคิวก่อน จะตรวจสอบสถานะ User โดยตรวจสอบจาก password ที่ Log in เข้ามาถูกต้องหรือไม่ ถ้า password ไม่ถูกต้องจะรายงานความผิดพลาดออกมา

3.1.1.4 เมื่อรายงานความผิดพลาดแล้ว จะกลับเข้าไปตรวจสอบ ผู้ที่อยู่ใน Queue ลำดับถัดไป

3.1.1.5 ถ้า password ถูกต้องจะตอบรับและรับคำสั่งจาก User

3.1.1.6 เมื่อรับคำสั่งจาก User แล้ว ก็จะตรวจสอบคำสั่งว่าถูกต้องหรือไม่

3.1.1.7 ถ้าผิดพลาดก็จะรายงานต่อ User แล้วกลับไปรับคำสั่งต่อ

3.1.1.8 ถ้าคำสั่งถูกต้อง User ต้องการส่ง E-mail ไปยังปลายทาง ก็จะทำการส่งข้อมูลต่าง ๆ ไปยัง Server ปลายทาง

3.1.1.9 ถ้า User ต้องการ check mail ก็จะทำการค้นหาใน Mail box

3.1.1.10 ส่งข้อมูลไปให้สมาชิก

3.1.1.11 ก็จะวนกลับไปให้บริการ Queue ต่อไป

3.1.1.12 ถ้า Queue ต่อไป เป็น Server ก็จะตรวจสอบสมาชิก Mail box ของตนว่ามีสมาชิกที่ Server นั้นต้องการหรือไม่

3.1.1.3 ถ้าไม่มีหรือมีข้อผิดพลาด ก็จะรายงานข้อผิดพลาดแล้ววนกลับไปให้บริการ Queueต่อไป

3.1.1.14 ถ้ามี ก็จะรับส่งข้อมูลกันระหว่าง Server

3.2 ขั้นตอนการทดลอง

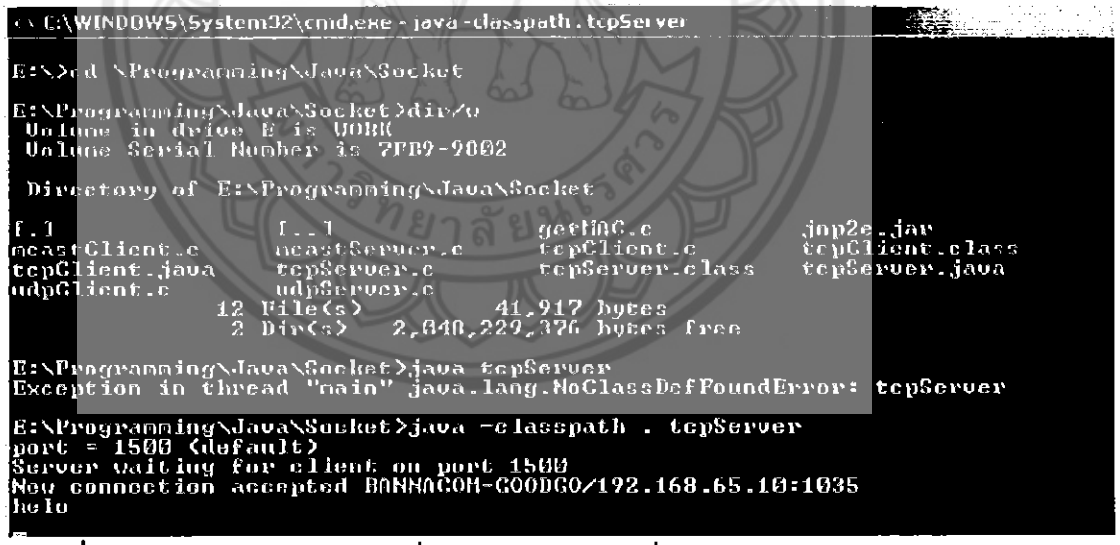
- ทดลองเขียนโปรแกรมติดต่อสื่อสารผ่านพอร์ต
- ทดลองเขียนโปรแกรมสร้างเครื่องแม่ข่ายส่งจดหมายอิเล็กทรอนิกส์ ตามมาตรฐาน SMTP
- ทดลองเขียนโปรแกรมสร้างเครื่องแม่ข่ายส่งจดหมายอิเล็กทรอนิกส์ ตามมาตรฐาน POP3
- ทดลองเขียนโปรแกรมสร้างเครื่องแม่ข่ายส่งจดหมายอิเล็กทรอนิกส์ ตามมาตรฐาน IMAP4

- นำสิ่งที่ทดลองและโปรแกรมส่วนต่าง ๆ มารวมกัน และ ปรับปรุงแก้ไข

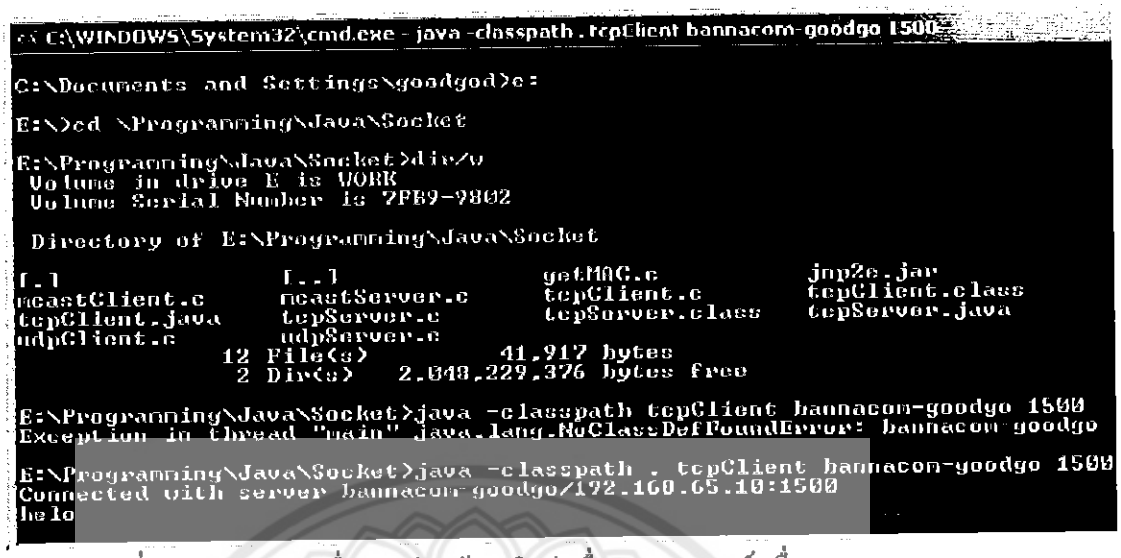
3.3 ทดลองเขียนโปรแกรม

3.3.1 เขียนโปรแกรมติดต่อสื่อสารผ่านพอร์ต

ในการติดต่อระหว่างเครื่องในระบบเครือข่ายจะต้องใช้ พอร์ตสื่อสาร ฝั่งเครื่องแม่ข่ายจะเปิดพอร์ตรอไว้ ส่วนเครื่องลูกข่ายจะเข้ามาติดต่อตามพอร์ตที่เครื่องแม่ข่ายเปิดไว้

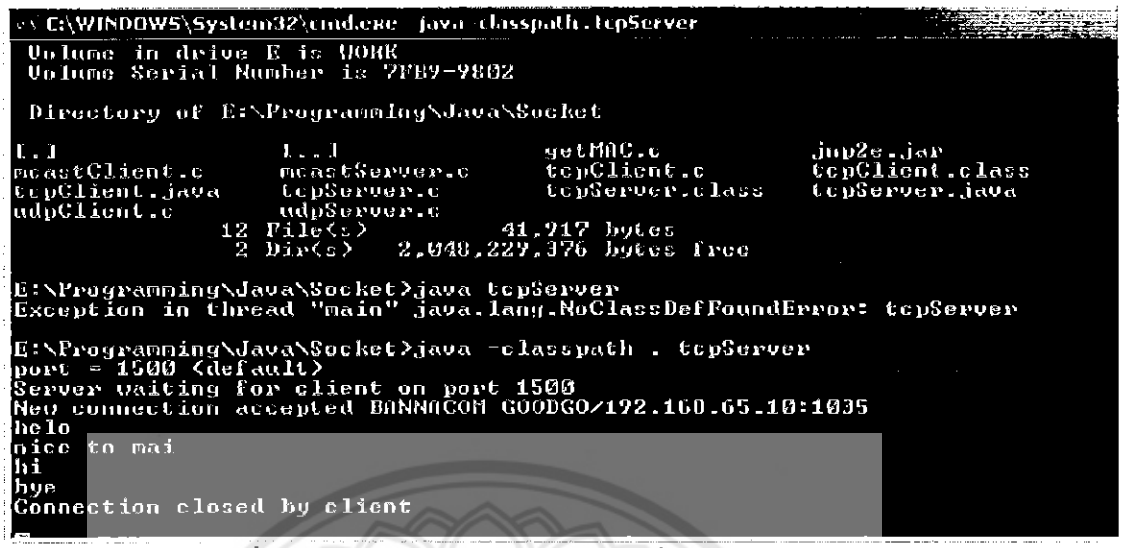


รูปที่ 3.2 โปรแกรมทำตัวเสมือนเครื่องแม่ข่าย เปิดพอร์ตสื่อสาร หมายเลข 1500 รอการติดต่อจากเครื่องลูกข่าย



รูปที่ 3.3 โปรแกรมเครื่องลูกข่ายเข้ามาติดต่อสื่อสาร ทางพอร์ตสื่อสาร หมายเลข 1500

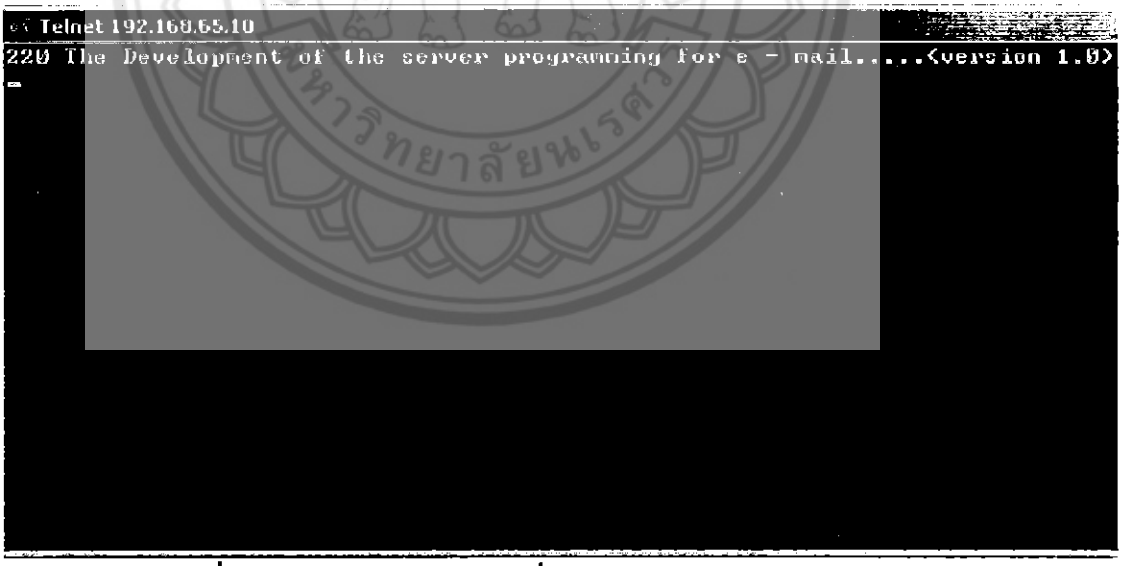
รูปที่ 3.4 แสดงการติดต่อของเครื่องลูกข่ายไปยังเครื่องแม่ข่าย



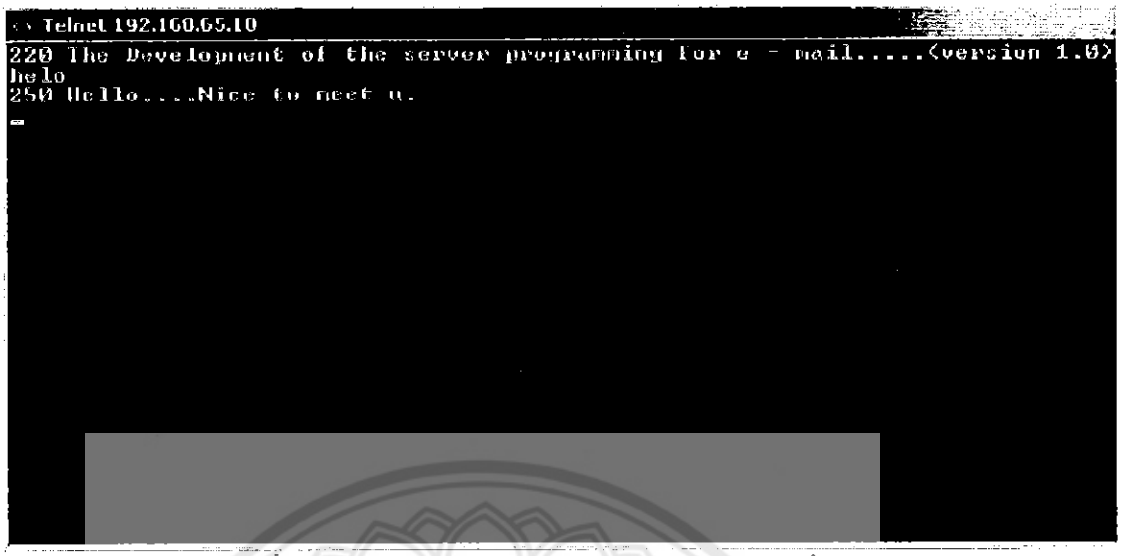
รูปที่ 3.5 แสดงการได้รับการติดต่อจากเครื่องลูกข่าย

3.3.2 ทดลองเขียนโปรแกรมสร้างเครื่องแม่ข่ายส่งจดหมายอิเล็กทรอนิกส์ ตามมาตรฐาน SMTP

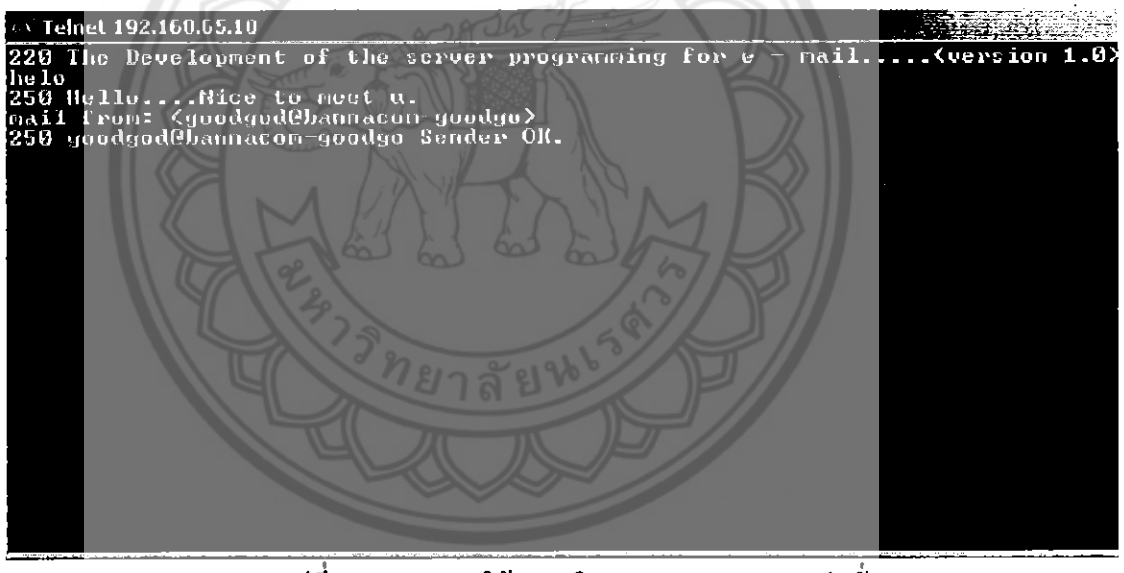
ในการติดต่อสื่อสารของโปรโตคอล SMTP นั้นจะใช้พอร์ตสื่อสารหมายเลข 25



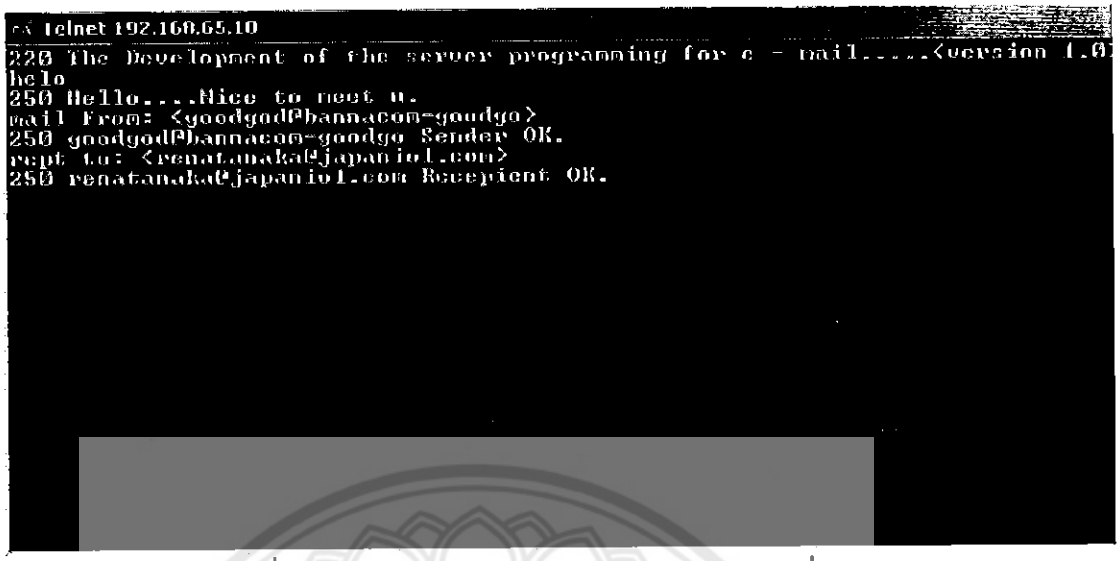
รูปที่ 3.6 แสดงการเปิดการติดต่อสื่อสารตามมาตรฐานโปรโตคอล SMTP



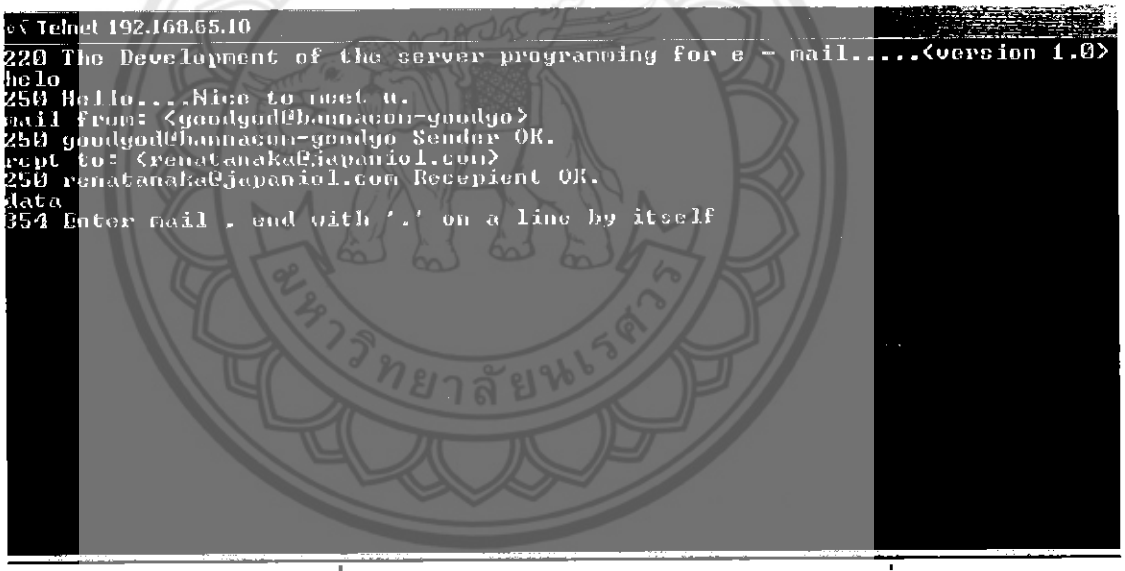
รูปที่ 3.7 แสดงการใช้งานบริการมาตรฐาน SMTP คำสั่ง helo



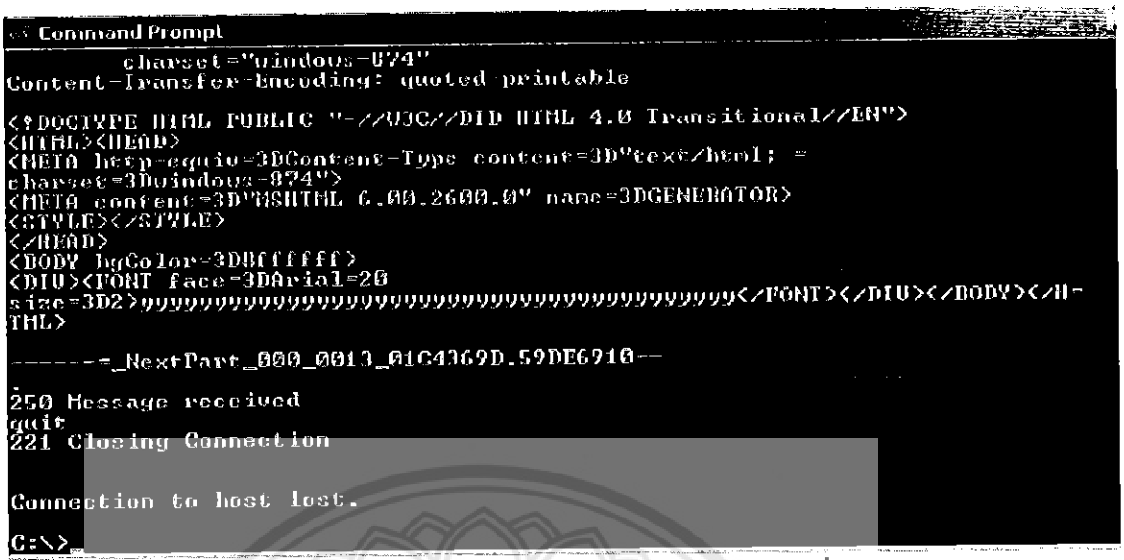
รูปที่ 3.8 แสดงการใช้งานบริการมาตรฐาน SMTP คำสั่ง mail to



รูปที่ 3.9 แสดงการใช้งานบริการมาตรฐาน SMTP คำสั่ง rcpt to



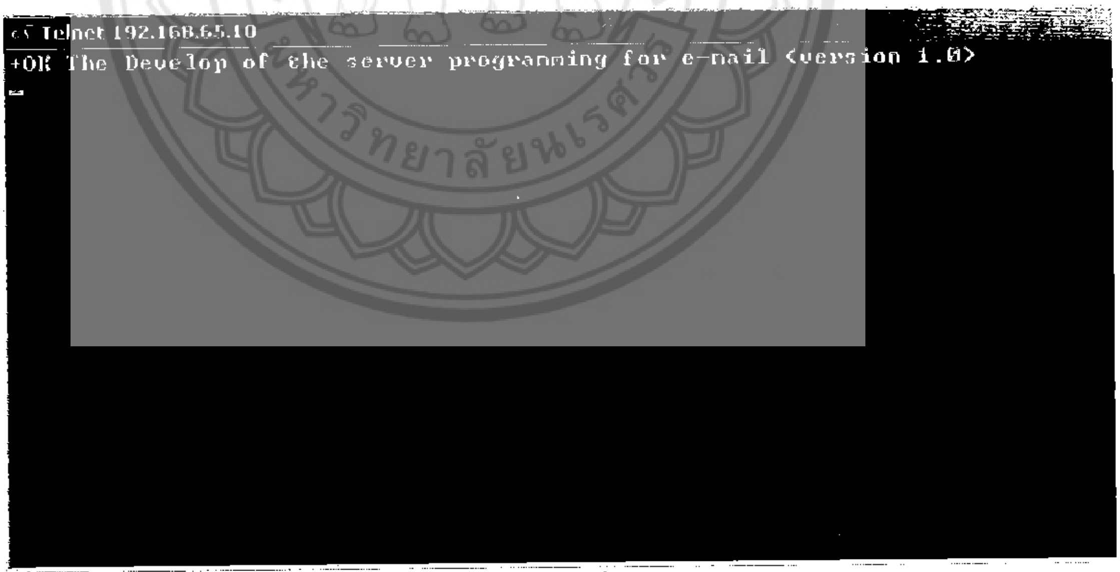
รูปที่ 3.10 แสดงการใช้งานบริการมาตรฐาน SMTP คำสั่ง data



รูปที่ 3.11 แสดงการใช้งานบริการมาตรฐาน SMTP คำสั่ง quit

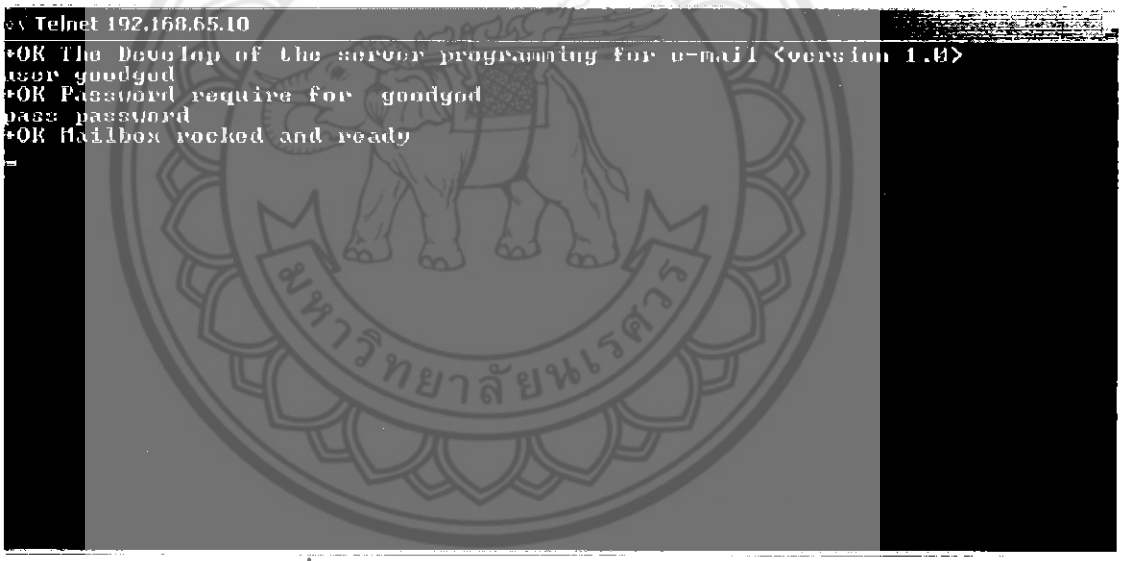
3.3.2 ทดลองเขียนโปรแกรมสร้างเครื่องแม่ข่ายส่งจดหมายอิเล็กทรอนิกส์ ตามมาตรฐาน POP3

ในการติดต่อสื่อสารของโปรโตคอล SMTP นั้นจะใช้พอร์ตสื่อสารหมายเลข 110



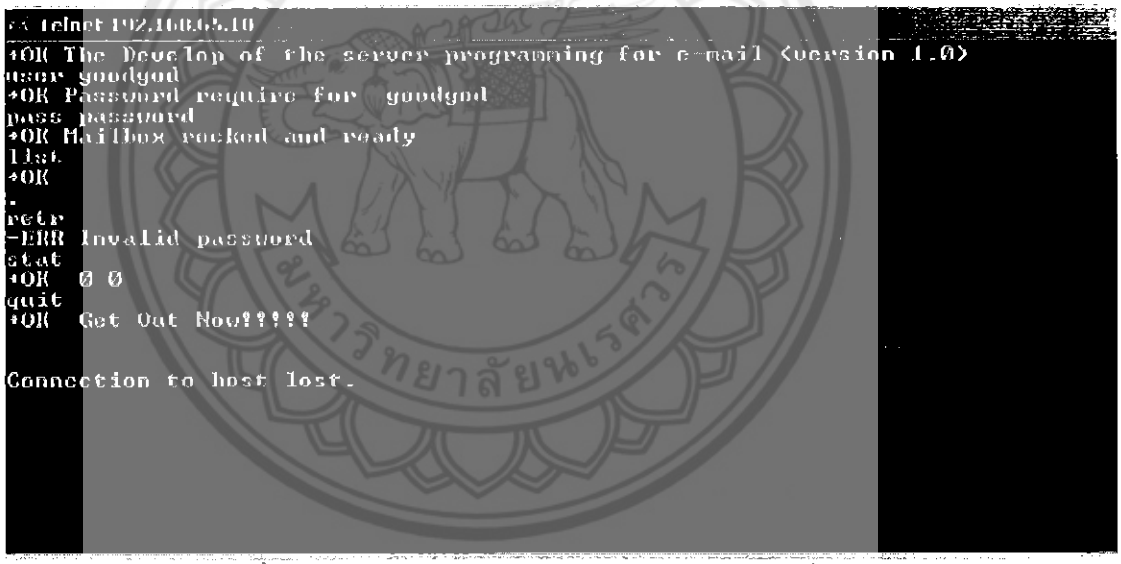
รูปที่ 3.12 แสดงการเปิดการติดต่อสื่อสารตามมาตรฐานโปรโตคอล POP3

รูปที่ 3.13 แสดงการใช้งานบริการมาตรฐาน POP3 คำสั่ง user



รูปที่ 3.14 แสดงการใช้งานบริการมาตรฐาน POP3 คำสั่ง pass

รูปที่ 3.15 แสดงการใช้งานบริการมาตรฐาน POP3 คำสั่ง list

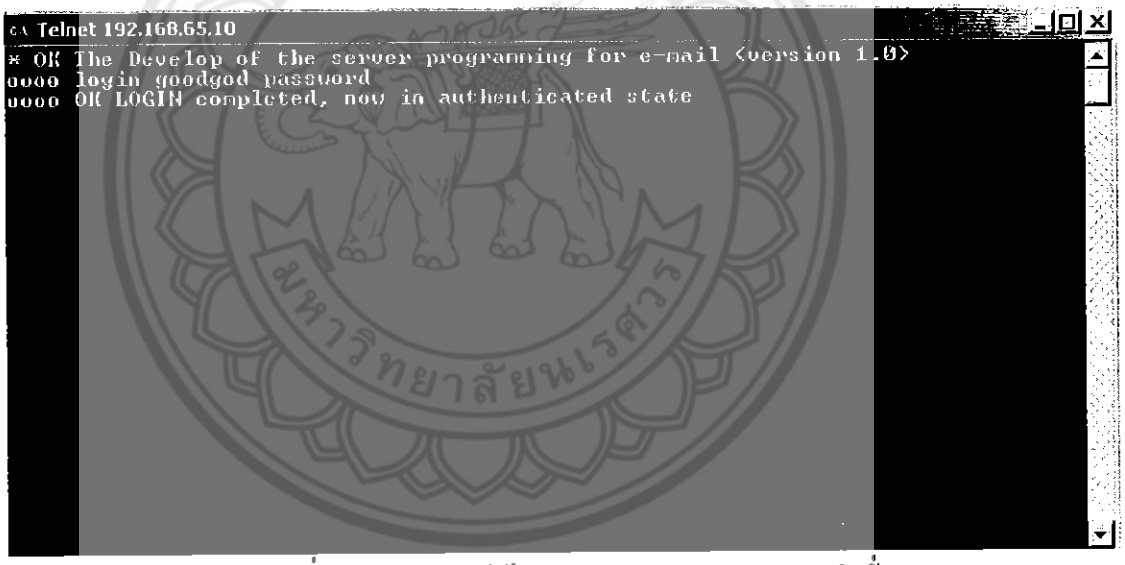


รูปที่ 3.16 แสดงการใช้งานบริการมาตรฐาน POP3 คำสั่ง quit

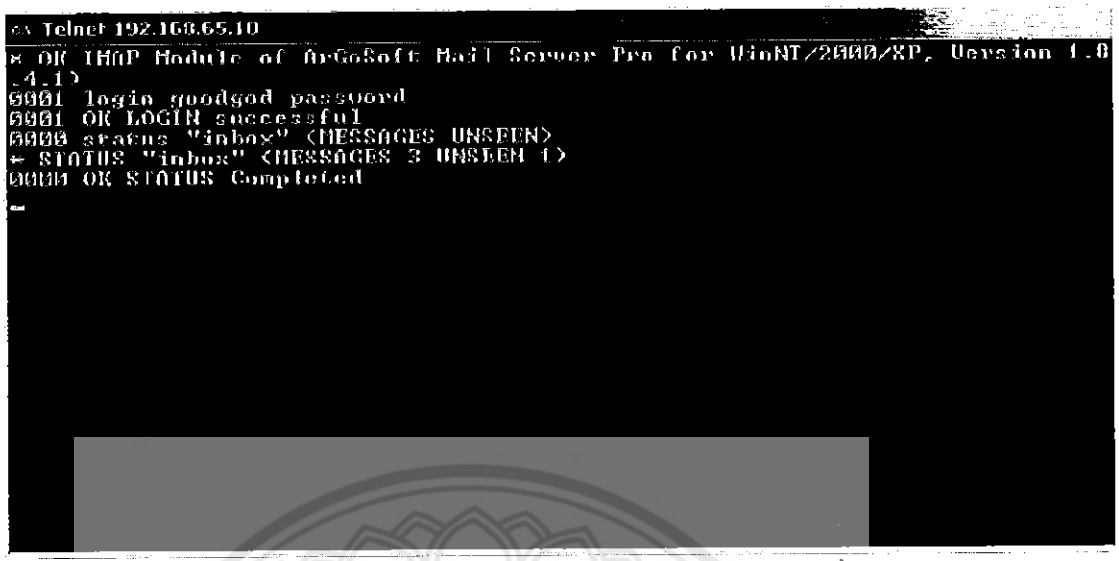
3.3.2 ทดลองเขียนโปรแกรมสร้างเครื่องแม่ข่ายส่งจดหมายอิเล็กทรอนิกส์ ตามมาตรฐาน IMAP4

ในการติดต่อสื่อสารของโปรโตคอล SMTP นั้นจะใช้พอร์ตสื่อสารหมายเลข 143

รูปที่ 3.17 แสดงการเปิดการติดต่อสื่อสารตามมาตรฐานโปรโตคอล IMAP4

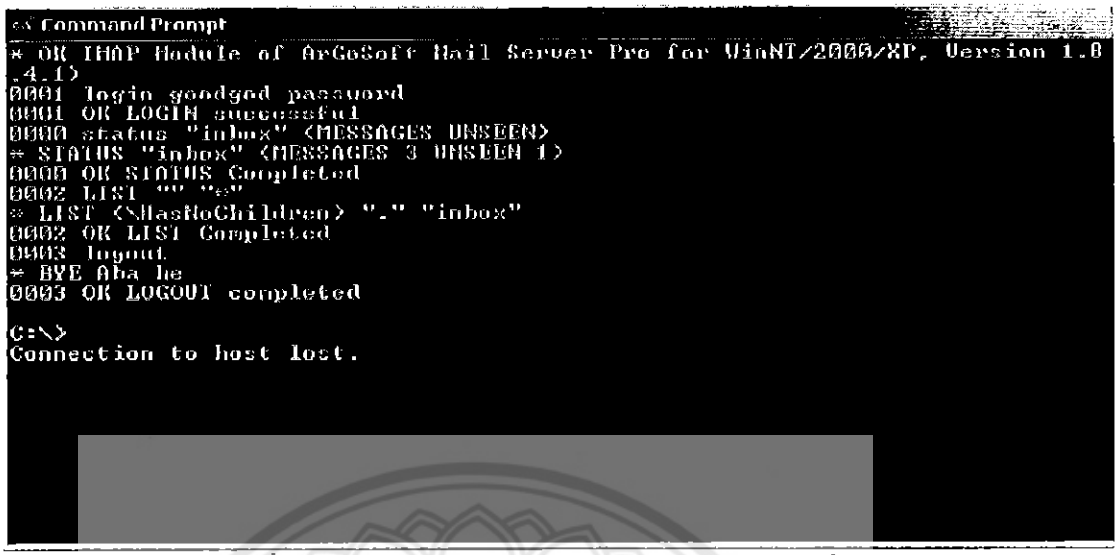


รูปที่ 3.18 แสดงการใช้งานบริการมาตรฐาน IMAP4 คำสั่ง login



รูปที่ 3.19 แสดงการใช้งานบริการมาตรฐาน IMAP4 คำสั่ง status

รูปที่ 3.20 แสดงการใช้งานบริการมาตรฐาน IMAP4 คำสั่ง list



รูปที่ 3.21 แสดงการใช้งานบริการมาตรฐาน IMAP4 คำสั่ง logout

บทที่ 4
ผลการทดสอบโปรแกรมและวิเคราะห์ผล

บทที่ 4 นี้ กล่าวถึงการทดสอบโปรแกรมว่ามีขั้นตอนการทดสอบอย่างไรและหลังจากทดสอบโปรแกรมแล้ว ผลการทำงานเป็นอย่างไร

4.1 จุดประสงค์ของการทดสอบโปรแกรม

4.1.1. เพื่อทดสอบโปรแกรมว่าสามารถทำงานได้ผลและมีประสิทธิภาพมากน้อยเพียงใดบรรลุตามวัตถุประสงค์หรือไม่

4.1.2. เพื่อทดสอบว่าโปรแกรมสามารถรองรับการให้บริการได้ดีเพียงใด

4.2 ขั้นตอนการทดสอบการทำงานของโปรแกรม

4.2.1 ติดตั้งโปรแกรมลงบนคอมพิวเตอร์ เพื่อให้เครื่องคอมพิวเตอร์เครื่องนั้นๆ ทำงานเป็นเครื่องแม่ข่ายในการรับส่งจดหมายอิเล็คทรอนิสก์และรอรับส่งจดหมายอิเล็กทรอนิกส์จากไคลเอนต์

4.2.2 ทดสอบทำงานจัดการระบบผู้ใช้งานต่างๆ การเพิ่ม ลบแก้ไขการกระจายข่าวรายงานสถิติ

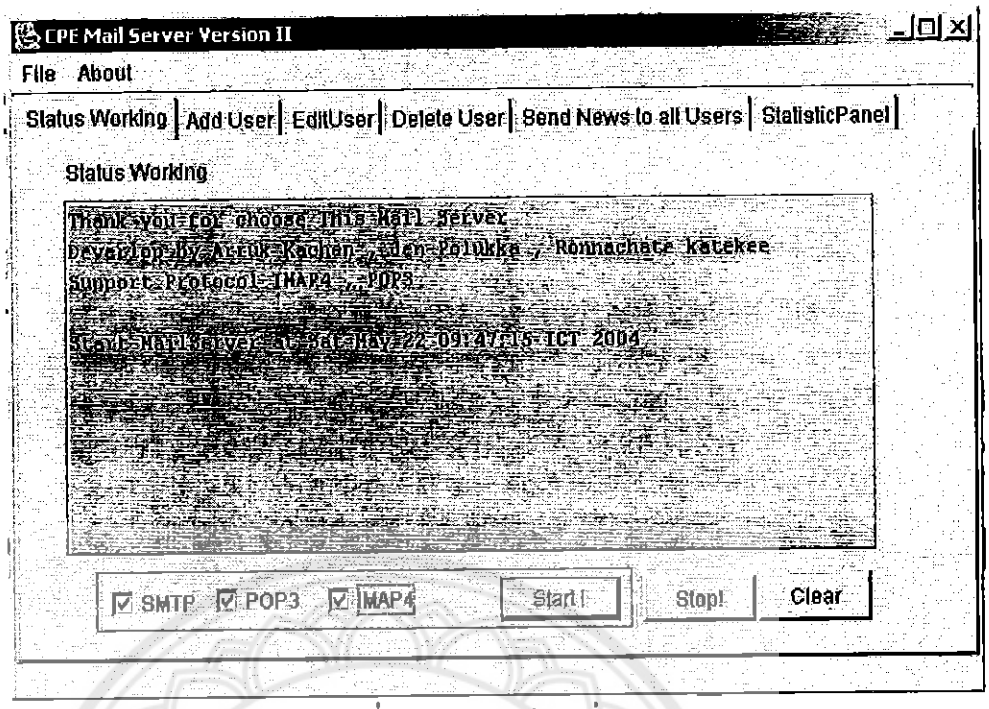
4.2.3 ทดสอบการส่งจดหมายเตือนเมื่อขนาดของตู้จดหมายที่เกินขอบเขต

4.2.4 ทดสอบการทำงานของโปรแกรมโดยใช้โปรแกรมเอาท์ลุคเอ็กเพรส ทำการเข้ามาอ่านจดหมายโดยผ่านโปรโตคอล POP3และ IMAP4 และส่งจดหมายโดยผ่านโปรโตคอล SMTP

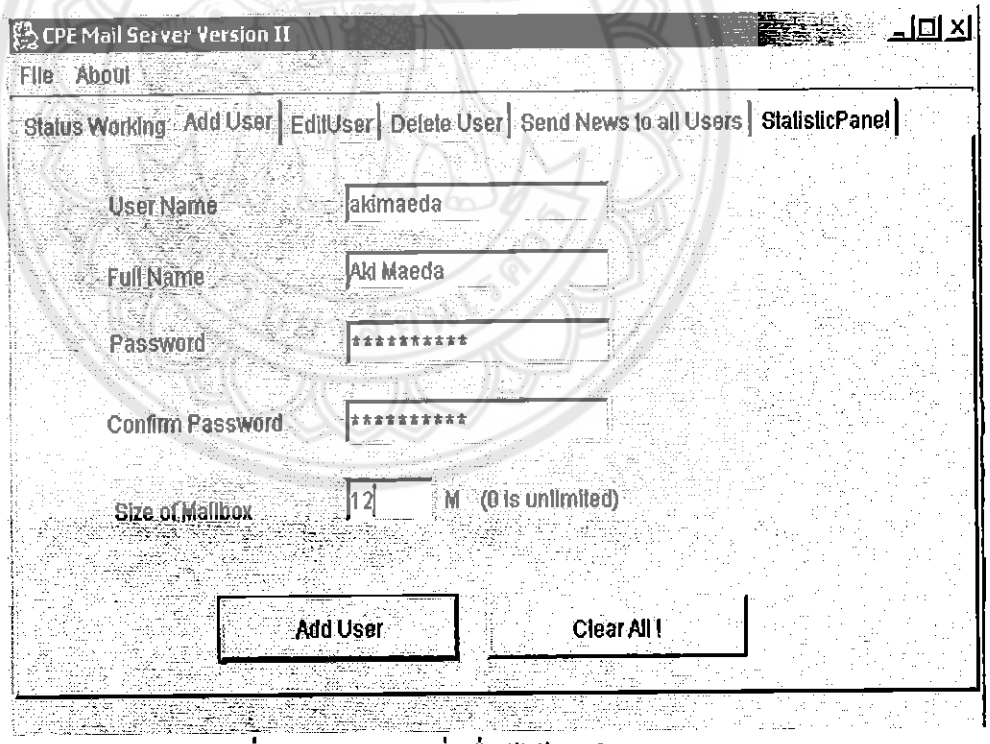
4.2.5 ตรวจสอบข้อผิดพลาดและทำการปรับปรุงแก้ไข

4.3 ผลการทดสอบ

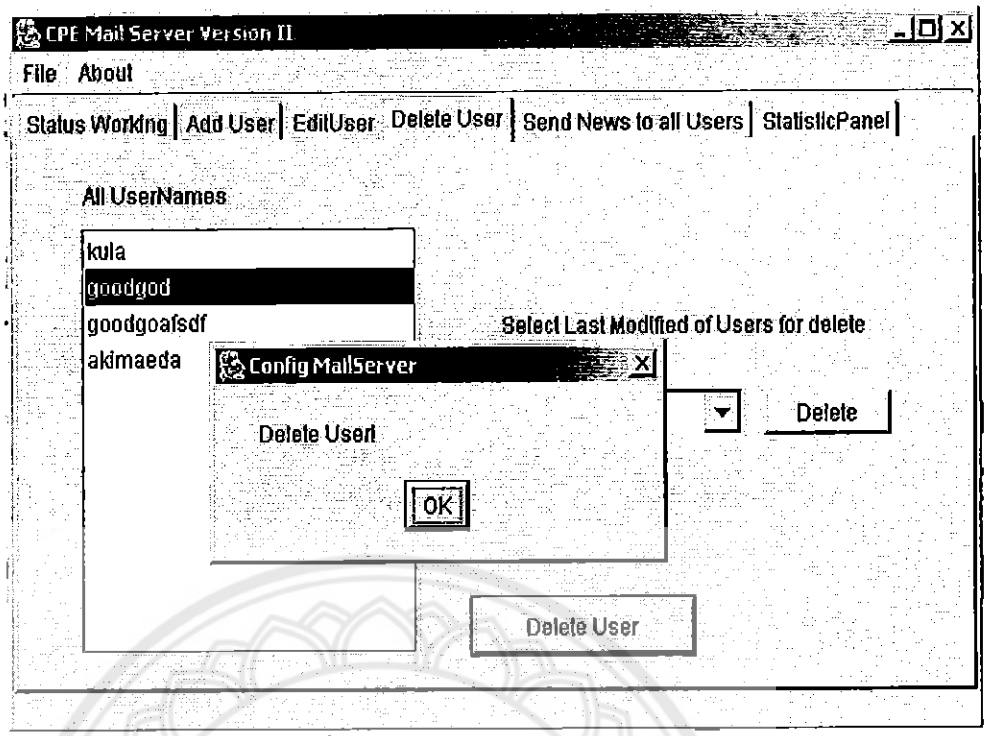
4.3.1 ทดสอบการจัดการระบบผู้ใช้งานต่างๆ โดยจะกำหนดผู้ใช้ขึ้นมา 2 คนคือ akimaeda กับ renatanaka ทำการสร้างผู้ใช้ทั้งสอง แก้ไข และ ลบ กระจายข่าวให้กับผู้ใช้ทุกคน



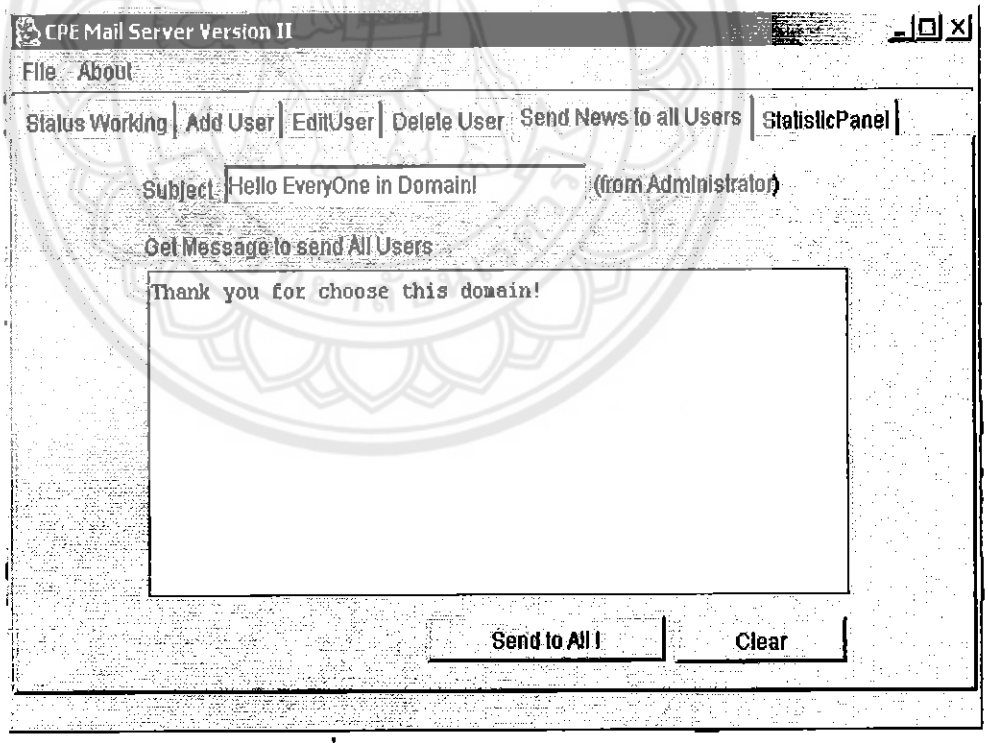
รูปที่4.1 แสดงหน้าจอเริ่มทำงาน



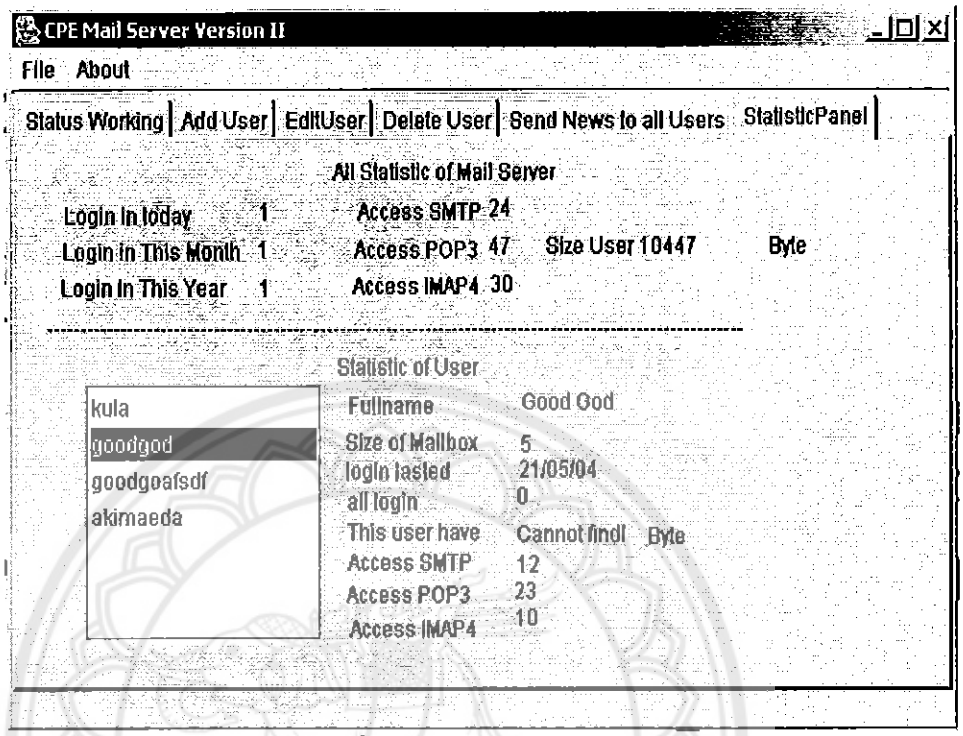
รูปที่ 4.2 แสดงการเพิ่มชื่อผู้ใช้ของในระบบ



รูปที่4.3 แสดงการแก้ไขผู้ใช้ของในระบบ



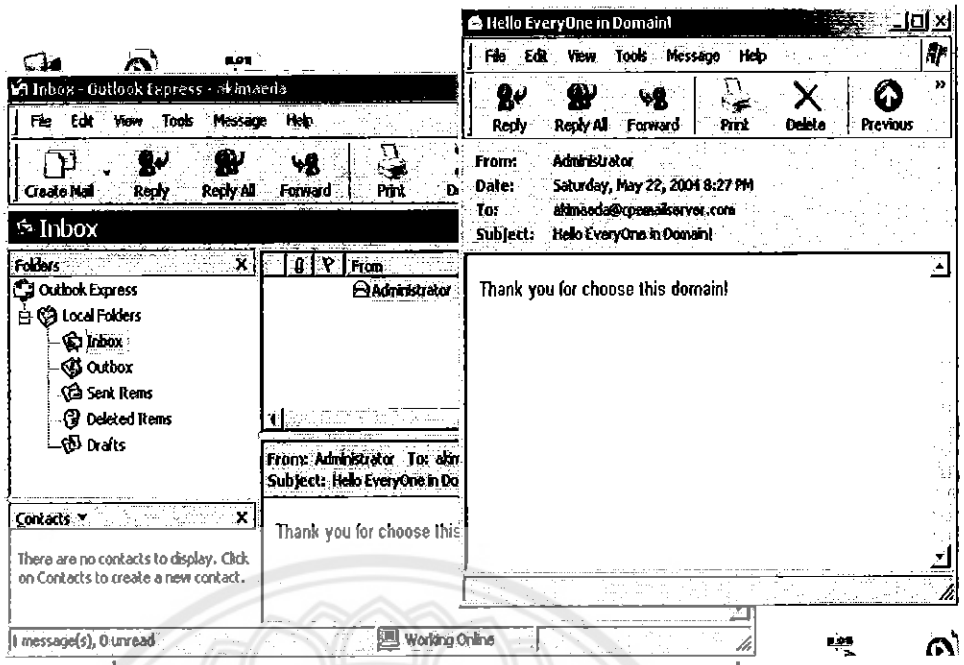
รูปที่4.4 แสดงหน้าจอการส่งข่าวแก่ผู้ใช้



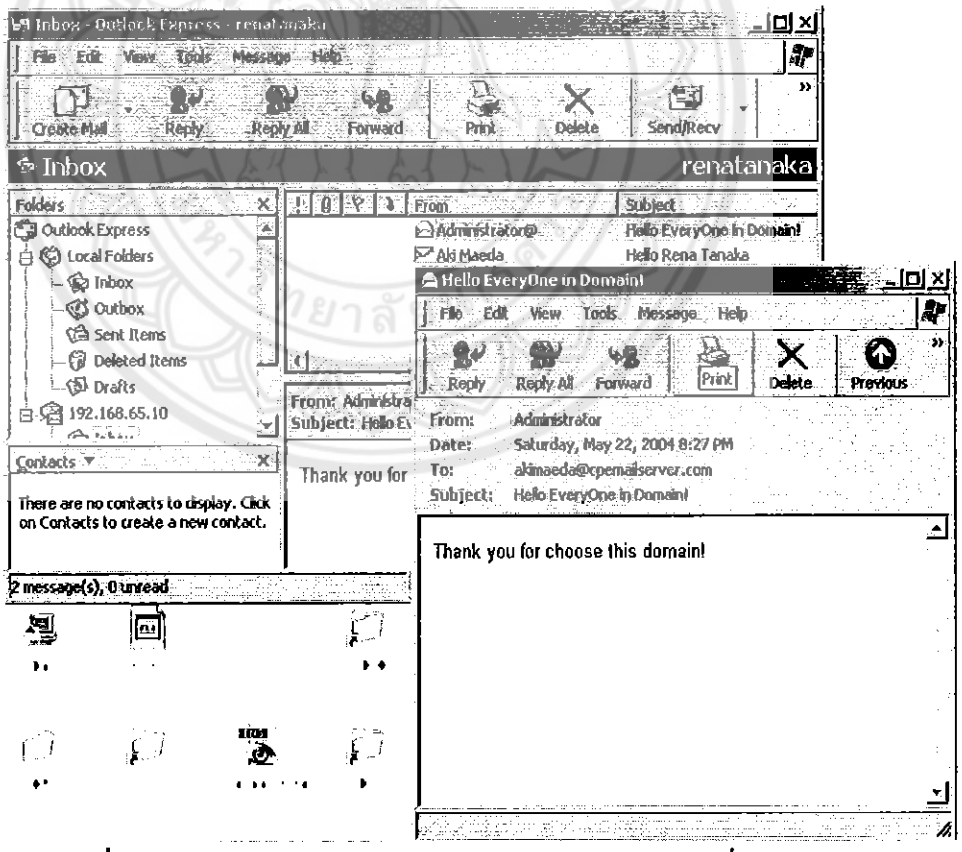
รูปที่4.5 แสดงหน้าจอรายงานสถิติ

4.3.2 ทดสอบการทำงานรับจดหมายผ่านโปรโตคอล POP3และ IMAP4

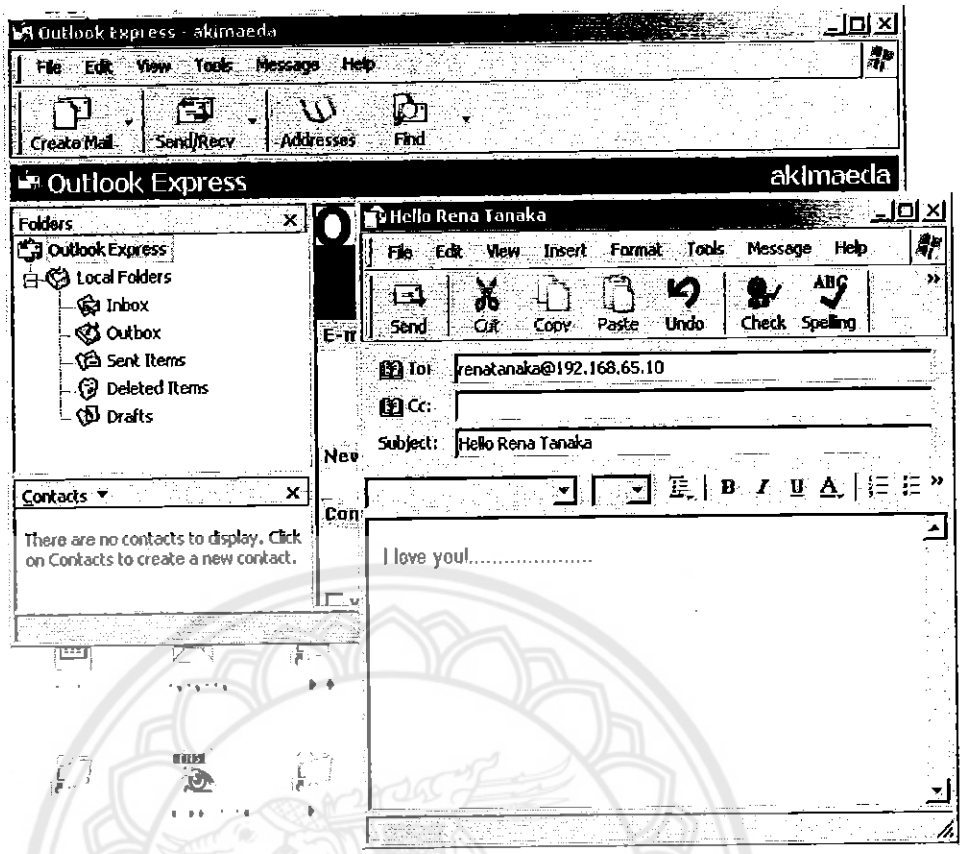
โดยจะกำหนดให้ผู้ใช้ akimaeda เข้าสู่บริการ POP3 และ renatanaka เข้าสู่บริการ IMAP4



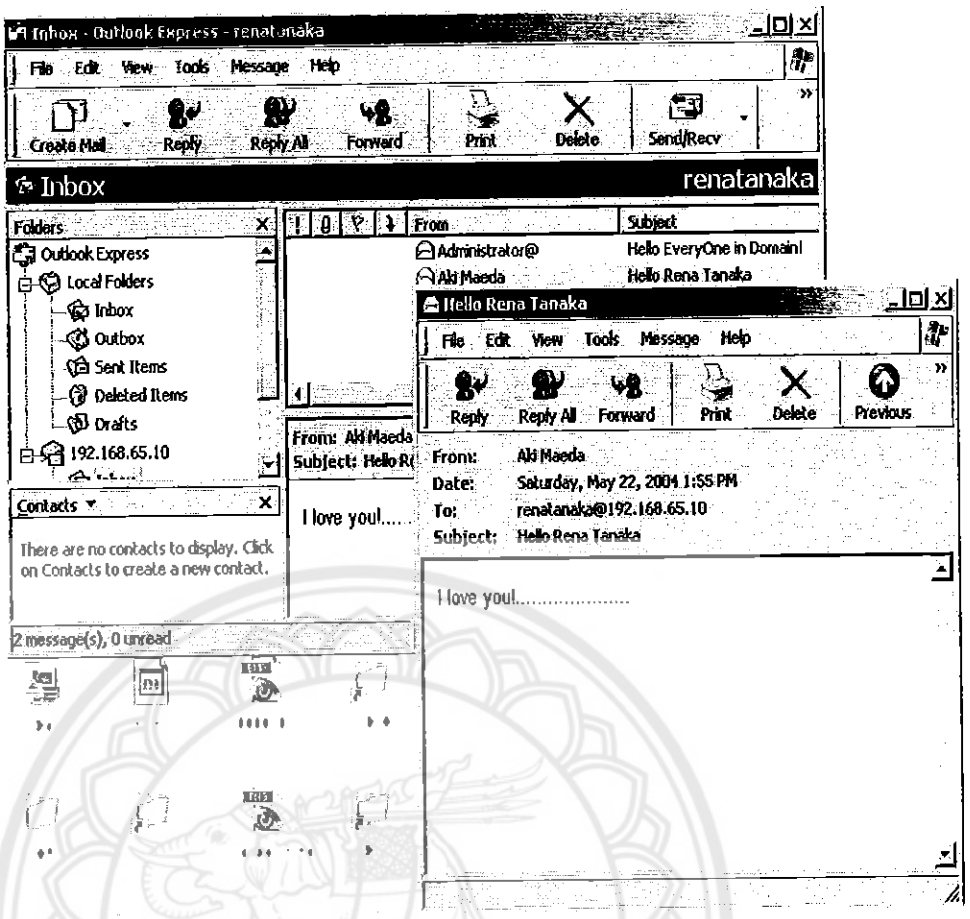
รูปที่4.6 แสดงหน้าจอการได้รับข่าวสารของระบบของผู้ใช้ที่เข้าใช้ระบบ POP3



รูปที่4.7 แสดงหน้าจอการได้รับข่าวสารของระบบของผู้ใช้ที่เข้าใช้ระบบIMAP4

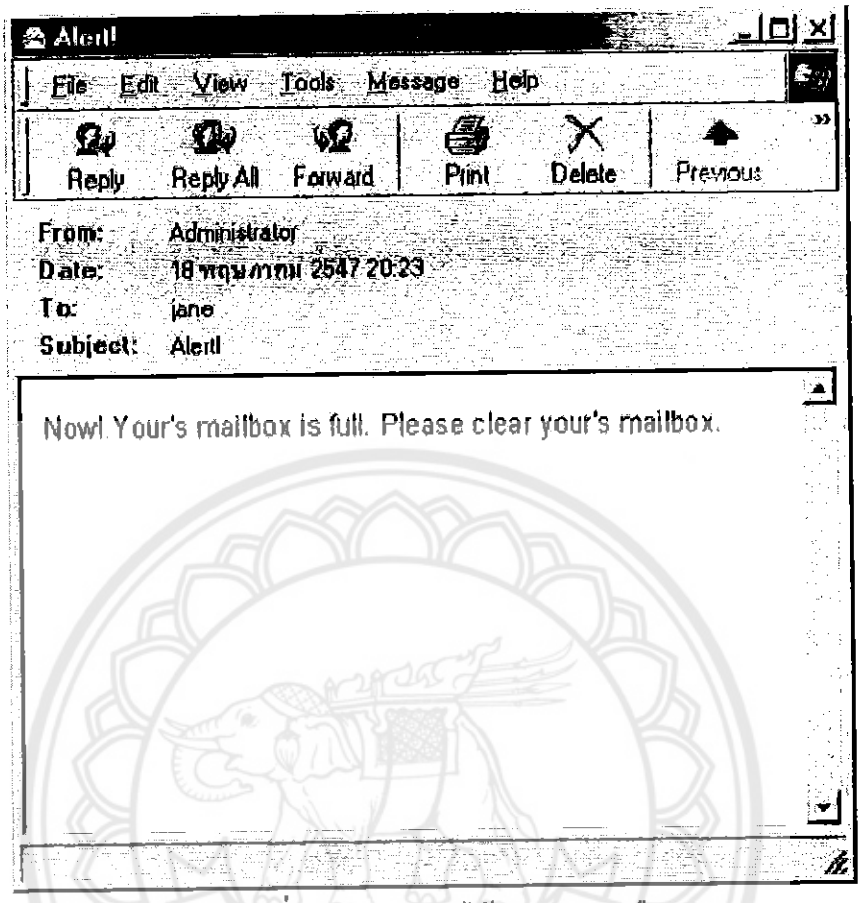


รูปที่4.8 แสดงการส่งจดหมายจากผู้ใช้ระบบPOP3สู่ระบบIMAP4



รูปที่4.9 แสดงการได้รับจดหมายจากผู้ใช้ระบบPOP3ของผู้ใช้ระบบIMAP4

4.3.3 ทดสอบการได้รับจดหมายเตือนว่าขนาดของตู้จดหมายเกินขอบเขตแล้ว



รูปที่4.10 แสดงการได้รับจดหมายเตือนจากระบบ

4.4 วิเคราะห์งาน

จากผลการทดสอบโปรแกรมปรากฏว่า ผลการทำงาน ของการทดสอบ สามารถใช้งานได้ดีมาก รวมทั้งเมื่อนำโปรแกรมไปทดสอบการรับส่งอีเมล์บน ระบบเครือข่ายหรือ เครื่องเดียวกันก็สามารถทำงานได้เหมือนกัน แสดงว่า รูปแบบหรือสภาวะแวดล้อมของระบบเครือข่ายนั้น จะไม่มีผลต่อการทำงานของโปรแกรม แต่การทดสอบโปรแกรมผ่านระบบเครือข่ายอินเตอร์เน็ตนั้นยังไม่สามารถทำได้ เนื่องจากระบบเครือข่ายอินเตอร์เน็ตในมหาวิทยาลัย ซึ่งไม่ได้มีการใช้ค่าไอพีที่แท้จริง และมีการทำงานผ่านพร็อกซี่เซิร์ฟเวอร์ ทำให้ไม่สามารถทดสอบได้

บทที่ 5
สรุปผลและข้อเสนอแนะ

ในบทนี้เป็นการสรุปผลทั้งหมดของการทำโครงงานนี้ ซึ่งประกอบด้วยส่วนสรุปผล ปัญหาในการทำงาน ข้อเสนอแนะ และแนวทางในการพัฒนาต่อไป หากมีผู้ที่สนในที่จะนำโครงงานนี้ไปพัฒนาและปรับปรุงให้มีประสิทธิภาพดีขึ้นต่อไป

5.1 สรุปผล

5.1.1 โปรแกรมสำหรับเครื่องแม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ สามารถให้บริการรับและส่งจดหมายอิเล็กทรอนิกส์ ภายใต้โปรโตคอล SMTP (RFC 822) และ POP 3 (RFC 1939) ได้

5.1.2 โปรแกรมสำหรับเครื่องแม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ สามารถให้บริการการรับและส่งจดหมายอิเล็กทรอนิกส์ ภายใต้โปรโตคอล IMAP (RFC 1730)

5.1.3 โปรแกรมมีส่วนติดต่อกับผู้ใช้(User Interface) ในลักษณะหน้าต่างทำให้สะดวกและง่ายต่อการใช้งานมากขึ้น

5.2 ปัญหาในการทำงาน

5.2.1 การพัฒนาโปรแกรมนี้ใช้ภาษาจาวา (JAVA) ในการพัฒนา ซึ่งเป็นภาษาที่ผู้จัดทำคุ้นเคยในการใช้งานมาก่อน ทำให้ต้องใช้เวลาในการศึกษาการใช้งานภาษาจาวามาก มีผลทำให้การพัฒนาโปรแกรมเป็นไปอย่างล่าช้า

5.2.2 การพัฒนาโปรแกรมในส่วนคำสั่งต่าง ๆ และตามมาตรฐานโปรโตคอลไม่สามารถทำได้ทัน เนื่องจากมีความซับซ้อนของระบบการทำงานเป็นอย่างมาก ในการทดสอบร่วมกับเมล์เซิร์ฟเวอร์ตามมาตรฐานอื่นๆ หากเซ็ตโปรแกรมไม่ถูกต้อง จะทำให้ไม่สามารถทำงานร่วมกันได้

5.2.3 ในการทดสอบโปรแกรมผ่านระบบอินเตอร์เน็ตไม่สามารถทำได้

5.3 ข้อเสนอแนะ

5.3.1 ควรพัฒนาโปรแกรมให้เสร็จสิ้นตามวัตถุประสงค์ เพื่อให้ได้ตามมาตรฐานที่อ้างอิง

5.3.2 ควรดำเนินงานแต่ละขั้นตอนให้ได้ตามระยะเวลาที่กำหนด เพื่อไม่ให้เกิดความล่าช้าในการทำโครงงานตลอดจนต้องศึกษาข้อมูลให้ถ่องแท้เสียก่อนเพื่อไม่ให้เกิดข้อผิดพลาดขึ้นในการทำงาน

5.3.3 ศึกษาการทำงานของโปรโตคอล SMTP และ POP 3 ภาษาที่จะใช้ในการพัฒนาข้อมูลตลอดจนข้อมูลอื่น ๆ เพื่อจะนำไปใช้ในการปรับปรุงแก้ไขโปรแกรมในโอกาสต่อไป

5.4 แนวทางในการพัฒนา

5.4.1 ศึกษาโปรโตคอลให้เข้าใจอย่างจริงแท้เพราะ จะเสียเวลาในการศึกษาส่วนนี้เป็นอย่ามาก

5.4.2 ศึกษาข้อมูลและการทำงานของ RFC ส่วนขยายอื่น ๆ ที่เกี่ยวข้อง แล้วนำมาพัฒนาโปรแกรมในส่วนที่เหลือ

5.4.3 เพิ่มประสิทธิภาพการทำงานในส่วนต่าง ๆ ของโปรแกรม เช่น MIME(Multipurpose Internet Mail Extensions) เป็นข้อกำหนดขยายการทำงานของอินเทอร์เน็ตเมลให้สามารถรับส่งข้อมูลไบนารีได้โดยไม่ต้องใช้โปรแกรมแปลงข้อมูล และมีการเข้ารหัส ทำให้มีความปลอดภัยในจดหมายอิเล็กทรอนิกส์ของเรา

5.4.4 เพิ่มการพัฒนาการให้บริการรับส่งจดหมายอิเลกทรอนิกส์ผ่านทางเวปไซต์ ตัวอย่าง เช่น HotMail ThaiMail เป็นต้น

เอกสารอ้างอิง

[1] ไพรัตน์ โพธิ์ศรี และคณะ. **การพัฒนาโปรแกรมสำหรับเครื่องคอมพิวเตอร์แม่ข่ายเพื่อการรับส่งจดหมายอิเล็กทรอนิกส์.** วิทยานิพนธ์ วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรม คอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์. 2545

[2] สุรศักดิ์ สงวนพงษ์. **สถาปัตยกรรมและโปรโตคอลทีซีพี/ไอพี.** กรุงเทพมหานคร:ซีเอ็ดยูเคชั่น, 2545.

[3] สุวัฒน์ ปุณณชัย และคณะ. **เปิดโลก TCP/IP และโปรโตคอลของอินเตอร์เน็ต.** กรุงเทพมหานคร:โปรวิชั่น. 2543.

[4] วีระศักดิ์ ซึงถาวร. **JAVA PROGRAMMING Volume I,II.** กรุงเทพมหานคร : ซีเอ็ดยูเคชั่น, 2545

[5] Alexander Newman. **Special Edition Using Java Written.** 1996

[6] Laurence Vanhelsuwe. **Mastering Java.** TECH BUBLICTIONS PTE LTD, 1996

[7] Madha Siddalingaiah, Stephen D. Lockwood. **JAVA HOW-TO THE DEFINITIVE.** 1996

[8] Mike Fletchr. **Java™ UNLEASHED.** 1996.

ภาคผนวก

IMAP4 (Internet Message Access Protocol 4 : RFC 1730)

คำสั่ง NOOP

สถานะ : ทุกสถานะ

พารามิเตอร์ : ไม่มี

รายละเอียด : เป็นคำสั่ง No Operation และค่าที่เซิร์ฟเวอร์จะถูกตั้งต้นใหม่

ตัวอย่าง :

C : a002 NOOP

S : a002 OK NOOP completed

คำสั่ง CAPABILITY

สถานะ : ทุกสถานะ

พารามิเตอร์ : ไม่มี

รายละเอียด : คำสั่งเพื่อตรวจสอบว่าเซิร์ฟเวอร์ใช้โปรโตคอล IMAP ได้

ตัวอย่าง : C : abcd CAPABILITY

S : * CAPABILITY IMAP4rev1 STARTTLS AUTH=GSSAPI LOGINDISABLED

S : abcd OK CAPABILITY completed

C : ijkl CAPABILITY

S : * CAPABILITY IMAP4rev1 AUTH=GSSAPI AUTH=PLAIN

S : ijkl OK CAPABILITY completed

คำสั่ง LOGOUT

สถานะ : ทุกสถานะ

พารามิเตอร์ : ไม่มี

รายละเอียด : คำสั่งสิ้นสุดการทำงาน

ตัวอย่าง: C : A023 LOGOUT

S : * BYE IMAP4rev1 Server logging out

S : A023 OK LOGOUT completed(Server and client then close the connection)

คำสั่ง AUTHENTICATE

สถานะ : ก่อนอนุมัติ

พารามิเตอร์ : รหัสเพื่อขออนุมัติ

รายละเอียด : คำสั่งที่ใช้เลือกกลไกการรับรอง ประกอบด้วย Kerberos ,V4,S/Key และ GSSAPI

ตัวอย่าง: S : * OK IMAP4rev1 Server

C : A001 AUTHENTICATE GSSAPI

S : +

C : YIIB+wYJKoZIhvcSAQICAQBuggHqMIIB5qADAgE

S : A001 OK GSSAPI authentication successful

คำสั่ง LOGIN

สถานะ : ก่อนอนุมัติ

พารามิเตอร์ : ชื่อและ Password

รายละเอียด : คำสั่งเพื่อระบุชื่อผู้ใช้งาน และ Password

ตัวอย่าง: C : a001 LOGIN SMITH SESAME

S : a001 OK LOGIN completed

คำสั่ง CREATE

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : คำสั่งสร้างเมล์บ็อกซ์

ตัวอย่าง: C : A003 CREATE owatagusiam/

S : A003 OK CREATE completed

C : A004 CREATE owatagusiam.blurdybloop

S : A004 OK CREATE completed

คำสั่ง DELETE

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : คำสั่งลบเมล์บ็อกซ์

ตัวอย่าง: S : A682 OK LIST completed

C : A683 DELETE blurdybloop

S : A683 OK DELETE completed

C : A684 DELETE foo

S : A684 NO Name "foo" has inferior hierarchical names

C : A685 DELETE foo.bar

S : A685 OK DELETE Comp leted

คำสั่ง SELECT

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : คำสั่งเลือกเมล์บ็อกซ์

ตัวอย่าง: C : A142 SELECT INBOX

S : * 172 EXISTS

S : * 1 RECENT

S : * OK [UNSEEN 12] Message 12 is first unseen

S : * OK [UIDVALIDITY 3857529045] UIDs valid

S : * OK [UIDNEXT 4392] Predicted next UID

S : * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)

S : * OK [PERMANENTFLAGS (\Deleted \Seen *)] Limited

S : A142 OK [READ-WRITE] SELECT completed

คำสั่ง EXAMINE

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : คำสั่งเลือกเมล์บ็อกซ์แต่จะเปิดใช้แบบอ่านอย่างเดียว(READ ONLY)

ตัวอย่าง: C : A932 EXAMINE blurdybloop

S : * 17 EXISTS

S : * 2 RECENT

S : * OK [UNSEEN 8] Message 8 is first unseen

S : * OK [UIDVALIDITY 3857529045] UIDs valid

S : * OK [UIDNEXT 4392] Predicted next UID

S : * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)

S : * OK [PERMANENTFLAGS ()] No permanent flags permitted

S : A932 OK [READ-ONLY] EXAMINE completed

คำสั่ง RENAME

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์ เดิม/ใหม่

รายละเอียด : คำสั่งเปลี่ยนชื่อเมล์บ็อกซ์

ตัวอย่าง: C : A683 RENAME blurdybloop sarasoop

S : A683 OK RENAME completed

C : A684 RENAME foo zowie

S : A684 OK RENAME Completed

คำสั่ง SUBSCRIBE

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : คำสั่งเพิ่มชื่อเมล์บ็อกลงในทะเบียน

ตัวอย่าง: C : A002 SUBSCRIBE #news.comp.mail.mime

S : A002 OK SUBSCRIBE completed

คำสั่ง UNSUBSCRIBE

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : ยกเลิกชื่อเมล์บ็อกซ์ออกจากทะเบียน

ตัวอย่าง: C : A002 UNSUBSCRIBE #news.comp.mail.mime

S : A002 OK UNSUBSCRIBE completed

คำสั่ง APPEND

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์[FLAG] [วันที่-เวลา]ข้อความ

รายละเอียด : ใช้เพิ่มอีเมล์ลงในเมล์บ็อก

ตัวอย่าง: C : A003 APPEND saved-messages (\Seen) {310}

S : + Ready for literal data

C : Date: Mon, 7 Feb 1994 21:52:25 -0800 (PST)

C : From: Fred Foobar <foobar@Blurdybloop.COM>

C : Subject: afternoon meeting

C : To: mooch@owatagu.siam.edu

C : Message-Id: <B27397-0100000@Blurdybloop.COM>

C : MIME-Version: 1.0

C : Content-Type: TEXT/PLAIN; CHARSET=US-ASCII

C : Hello Joe, do you think we can meet at 3:30 tomorrow?

S : A003 OK APPEND completed

คำสั่ง LIST

สถานะ : อนุมัติ

พารามิเตอร์ : CONTEXT,ชื่อตู้รับอีเมล์

รายละเอียด : แสดงรายชื่อเมล์บ็อกซ์

ตัวอย่าง: C : A101 LIST "" ""

S : * LIST (\Noselect) "/" ""

S : A101 OK LIST Completed

C : A102 LIST #news.comp.mail.misc ""

S : * LIST (\Noselect) "." #news.

S : A102 OK LIST Completed

คำสั่ง LSUB

สถานะ : อนุมัติ

พารามิเตอร์ : CONTEXT,ชื่อตู้รับอีเมล์

รายละเอียด : แสดงรายชื่อเมล์บ็อกซ์เฉพาะที่ลงทะเบียนไว้

ตัวอย่าง: C : A002 LSUB "#news." "comp.mail.*"

S : * LSUB () "." #news.comp.mail.mime

S : * LSUB () "." #news.comp.mail.misc

S : A002 OK LSUB completed

C : A003 LSUB "#news." "comp.%"

S : * LSUB (\NoSelect) "." #news.comp.mail

S : A003 OK LSUB completed

คำสั่ง STATUS

สถานะ : อนุมัติ

พารามิเตอร์ : ชื่อตู้รับอีเมล์

รายละเอียด : ตรวจสอบรายละเอียดของเมล์บ็อกซ์

ตัวอย่าง: C : A042 STATUS blurdybloop (UIDNEXT MESSAGES)

S : * STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292)

S : A042 OK STATUS completed

คำสั่ง FETCH

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : เซ็ตของข้อความ,ชื่อข้อความ

รายละเอียด : อ่านข้อมูลอีเมล์ที่ต้องการ(อ่านได้ทั้งหมด หรือบางส่วน)

ตัวอย่าง: C : A654 FETCH 2:4 (FLAGS BODY[HEADER.FIELDS (DATE FROM)])

S : * 2 FETCH ....

S : * 3 FETCH ....

S : * 4 FETCH ....

S : A654 OK FETCH completed

คำสั่ง STORE

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : เซ็ตของข้อความ,ข้อความ

รายละเอียด : ให้ส่งข้อมูลอีเมล์กลับไปอัพเดทที่เซิร์ฟเวอร์

ตัวอย่าง: C : A003 STORE 2:4 +FLAGS (\Deleted)

S : * 2 FETCH (FLAGS (\Deleted \Seen))

S : * 3 FETCH (FLAGS (\Deleted))

S : * 4 FETCH (FLAGS (\Deleted \Flagged \Seen))

S : A003 OK STORE completed

คำสั่ง CHECK

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : ไม่มี

รายละเอียด : ตรวจสอบสถานะของเมล์บ็อกในขณะนั้น

ตัวอย่าง: C : FXXZ CHECK

S : FXXZ OK CHECK Completed

คำสั่ง EXPUNGE

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : ไม่มี

รายละเอียด : ลบอีเมล์ที่มีเครื่องหมายออกจากเมล์บ็อกซ์

ตัวอย่าง: C : A202 EXPUNGE

S : * 3 EXPUNGE

S : * 3 EXPUNGE

S : * 5 EXPUNGE

S : * 8 EXPUNGE

S : A202 OK EXPUNGE completed

คำสั่ง SERCH

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : [เซ็ตของตัวอักษร],[เงื่อนไขการค้นหา]

รายละเอียด : ค้นหาอีเมล์ในเมล์บ็อกซ์ที่กำหนด

ตัวอย่าง: C : A282 SEARCH FLAGGED SINCE 1-Feb-1994 NOT FROM "Smith"

S : * SEARCH 2 84 882

S : A282 OK SEARCH completed

C : A283 SEARCH TEXT "string not in mailbox"

S : * SEARCH

S : A283 OK SEARCH completed

C : A284 SEARCH CHARSET UTF-8 TEXT {6}

C : XXXXXX

S : * SEARCH 43

S : A284 OK SEARCH completed

คำสั่ง COPY

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : เซ็ตของข้อความ,ชื่อข้อความ

รายละเอียด : คัดลอกอีเมล์ในเมล์บ็อก

ตัวอย่าง: C : A003 COPY 2:4 MEETING

S : A003 OK COPY completed

คำสั่ง UID

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : คำสั่ง,พารามิเตอร์ของคำสั่ง

รายละเอียด : คำสั่งแบบแบทช์ คือทำหลาย ๆ คำสั่งต่อเนื่องกันตามที่ระบุไว้ คล้ายกับแบทช์ไฟล์ของ DOS

ตัวอย่าง: C : A999 UID FETCH 4827313:4828442 FLAGS

S : * 23 FETCH (FLAGS (\Seen) UID 4827313)

S : * 24 FETCH (FLAGS (\Seen) UID 4827943)

S : * 25 FETCH (FLAGS (\Seen) UID 4828442)

S : A999 OK UID FETCH completed

คำสั่ง X

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : ขึ้นอยู่กับคำสั่งที่ใช้ร่วม

รายละเอียด : เป็นคำสั่งที่ใช้ทดลองการทำงาน

ตัวอย่าง: C : a441 CAPABILITY

S : * CAPABILITY IMAP4rev1 XPIG-LATIN

S : a441 OK CAPABILITY completed

C : A442 XPIG-LATIN

S : * XPIG-LATIN ow-nay eaking-spay ig-pay atin-lay

S : A442 OK XPIG-LATIN ompleted-cay

คำสั่ง CLOSE

สถานะ : เลือกตู้รับอีเมล์

พารามิเตอร์ : ไม่มี

รายละเอียด : ลบอีเมล์ที่มีเครื่องหมายออกจากเมล์บ็อกซ์และเปลี่ยนสถานะเป็นก่อนอนุมัติเพื่อรอรับงานต่อไป

ตัวอย่าง: C : A341 CLOSE

S : A341 OK CLOSE completed

ซอร์สโค้ดของโปรแกรม(Source of program)

Application1.java

คลาส Application1 เป็น คลาสหลักในของโปรแกรมในการแสดงผลจะเรียกคลาส Frame1 ที่เป็นคลาสรูปแบบฟอร์มของโปรแกรม

```
import javax.swing.UIManager;
import java.awt.*;
import java.io.*;

/**
 * <p>Title: MailServer</p>
 * <p>Description: MailServer Support POP3 , IMAP4</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: GoodGod CO.LTD</p>
 * @author unascribed
 * @version 1.0
 */

public class Application1 {
  boolean packFrame = false;

  //Construct the application
  public Application1() {
    Frame1 frame = new Frame1();
    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from their layout
    if (packFrame) {
      frame.pack();
    }
    else {
      frame.validate();
```

```
    }
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height) {
      frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width) {
      frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
    frame.setVisible(true);
  }
  //Main method
  public static void main(String[] args) throws IOException{
    try {
      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    new Application1();
  }
}
```

Frame1.java

เป็นคลาสที่เป็นรูปแบบของตัวโปรแกรมจะเรียกใช้คลาสต่างๆในการทำงาน

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.net.*;
import javax.swing.border.*;
public class Frame1 extends JFrame {
private Date date;
 String textFullName, textUserName,textSize;
 String textConfirm ;
 String textPassword ;
 JPanel contentPane;
 JMenuBar jMenuBar1 = new JMenuBar();
 JMenu jMenuFile = new JMenu();
 JMenuItem jMenuFileExit = new JMenuItem();
 JMenu jMenuHelp = new JMenu();
 JLabel statusBar = new JLabel();
 BorderLayout borderLayout1 = new BorderLayout();
 Box box1;
 JTabbedPane jTabbedPane1 = new JTabbedPane();
 JPanel AddUserPanel = new JPanel();
 JPanel EditUserPanel = new JPanel();
 JPanel DeletePanel = new JPanel();
 JPanel SendNEWSPanel = new JPanel();
 JPanel StatisticPanel = new JPanel();
 JTextField UserNameTextField = new JTextField();
 JTextField FullNameTextField = new JTextField();
 JLabel jLabel1 = new JLabel();
```

```
JLabel jLabel2 = new JLabel();
JLabel jLabel3 = new JLabel();
JLabel jLabel4 = new JLabel();
JPasswordField PasswordPasswordField1 = new JPasswordField();
JPasswordField ConfirmPasswordField2 = new JPasswordField();
JButton AddUserButton = new JButton();
JButton ClearAllButton = new JButton();
JLabel jLabel5 = new JLabel();
String[] limitmonth = {"1 month","2 month","3 month","6 month"};
JComboBox MonthComboBox = new JComboBox(limitmonth);
JLabel jLabel6 = new JLabel();
JButton DeleteUserButton = new JButton();
JButton DeleteFromMonthButton = new JButton();
JLabel jLabel8 = new JLabel();
JButton SendButton = new JButton();
JButton ClearButton = new JButton();
JScrollPane jScrollPane1 = new JScrollPane();
JTextArea jTextArea1 = new JTextArea();
JLabel jLabel11 = new JLabel();
JLabel jLabel12 = new JLabel();
JLabel jLabel13 = new JLabel();
JLabel LoginTodayLabel = new JLabel();
JLabel jLabel15 = new JLabel();
JLabel jLabel16 = new JLabel();
JLabel LoginMonthLabel = new JLabel();
JLabel LoginYearLabel = new JLabel();
JLabel jLabel19 = new JLabel();
JLabel jLabel20 = new JLabel();
JLabel jLabel21 = new JLabel();
JLabel jLabel22 = new JLabel();
JLabel LoginUserLabel = new JLabel();
JLabel SizeMailUserLabel = new JLabel();
```

```
JLabel SizeUserLabel = new JLabel();
JScrollPane jScrollPane2 = new JScrollPane();
static int MAXUSER = 1000;
Vector diruser = new Vector();
JList UserList = new JList(diruser);
String DirUser = "";
File f = new File(DirUser);
File pro = new File(DirUser,"property.user");
String listFile[] = new String[MAXUSER];
JButton StartWorkingButton = new JButton();
JScrollPane jScrollPane3 = new JScrollPane();
JLabel jLabel7 = new JLabel();
JButton StopWorkingButton = new JButton();
JPanel StatusWorkingPanel = new JPanel();
JTextArea StatusWorkingTextArea = new JTextArea();
JMenuItem jMenuItem1 = new JMenuItem();
JButton ClearWorkingButton = new JButton();
JLabel jLabel9 = new JLabel();
JScrollPane jScrollPane4 = new JScrollPane();
JList jList1 = new JList(diruser);
JLabel jLabel17 = new JLabel();
JLabel UserAllLoginLabel2 = new JLabel();
JLabel jLabel10 = new JLabel();
JLabel ShowNameLabel = new JLabel();
JScrollPane jScrollPane5 = new JScrollPane();
JList UserEditList = new JList(diruser);
JTextField jTextField2 = new JTextField();
JLabel jLabel18 = new JLabel();
JLabel jLabel110 = new JLabel();
JLabel jLabel111 = new JLabel();
JLabel jLabel112 = new JLabel();
JButton SaveEditButton = new JButton();
```

```
JLabel jLabel14 = new JLabel();
TitledBorder titledBorder1;
JCheckBox SMTPCheckBox = new JCheckBox();
JCheckBox POP3CheckBox = new JCheckBox();
JCheckBox IMAP4CheckBox = new JCheckBox();
JTextField AddSizeMailField = new JTextField();
JLabel jLabel23 = new JLabel();
JLabel jLabel24 = new JLabel();
JTextField EditSizeMailField = new JTextField();
JLabel jLabel25 = new JLabel();
JTextField SubjectField = new JTextField();
JLabel jLabel26 = new JLabel();
JLabel jLabel113 = new JLabel();
JLabel jLabel114 = new JLabel();
JLabel jLabel115 = new JLabel();
JLabel IMAPLabel = new JLabel();
JLabel POPLabel = new JLabel();
JLabel SMTPLabel = new JLabel();
JLabel jLabel116 = new JLabel();
JLabel jLabel117 = new JLabel();
JLabel jLabel118 = new JLabel();
JLabel jLabel119 = new JLabel();
JLabel jLabel1110 = new JLabel();
JLabel jLabel1111 = new JLabel();
JLabel UserIMAPLabel = new JLabel();
JLabel UserPOPLabel = new JLabel();
JLabel UserSMTPLabel = new JLabel();

MainConfigUser mcu = new MainConfigUser();
JPasswordField jPasswordField1 = new JPasswordField();
JPasswordField jPasswordField2 = new JPasswordField();
JLabel jLabel27 = new JLabel();
```

```
SMTPRun smtprun = new SMTPRun();
POP3Run pop3run = new POP3Run();
IMAP4Run imap4run = new IMAP4Run();



public Frame1() {
 enableEvents(AWTEvent.WINDOW_EVENT_MASK);
 try {
  jbInit();
 }
 catch(Exception e) {
  e.printStackTrace();
 }
}
        static void write(String n)
        {
          try
          {
                  File f1 = new File(n,"property.user");
               File f2 = new File(n,"propetry2.user");
                  f1.createNewFile();
             f2.createNewFile();
            }
            catch (IOException e)
            {
                  System.out.println(n);
             System.out.println(e);
            }

       }
    static void makeDir(String d)
    {
```

```
                File f = new File(d);
                f.mkdir();
              System.out.println(d);
    }

private void jbInit() throws Exception  {
  contentPane = (JPanel) this.getContentPane();
  box1 = Box.createVerticalBox();
  titledBorder1 = new TitledBorder("");
  contentPane.setLayout(borderLayout1);
  this.setSize(new Dimension(575, 380));
  this.setTitle("CPE Mail Server Version II");
  statusBar.setText(" ");
  jMenuFile.setText("File");
  jMenuFileExit.setText("Exit");
  jMenuFileExit.addActionListener(new ActionListener()  {
    public void actionPerformed(ActionEvent e) {
      jMenuFileExit_actionPerformed(e);
    }
  });
  jMenuHelp.setActionCommand("About");
  jMenuHelp.setText("About");
  AddUserPanel.setLayout(null);
  jLabel1.setText("User Name");
  jLabel1.setBounds(new Rectangle(55, 20, 141, 23));
  jLabel2.setText("Full Name");
  jLabel2.setBounds(new Rectangle(54, 57, 141, 23));
  jLabel3.setText("Password");
  jLabel3.setBounds(new Rectangle(55, 91, 141, 23));
  jLabel4.setText("Confirm Password");
  jLabel4.setBounds(new Rectangle(53, 133, 141, 23));
  AddUserButton.setBounds(new Rectangle(117, 235, 142, 35));
```

```
AddUserButton.setText("Add User");
AddUserButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e){
  AddUserButton_actionPerformed(e);
 }
});
ClearAllButton.setBounds(new Rectangle(276, 237, 152, 32));
ClearAllButton.setText("Clear All !");
ClearAllButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  ClearAllButton_actionPerformed(e);
 }
});
DeletePanel.setLayout(null);
jLabel5.setText("All UserNames");
jLabel5.setBounds(new Rectangle(39, 12, 202, 31));
jLabel6.setText("Select Last Modified of Users for delete");
jLabel6.setBounds(new Rectangle(285, 81, 230, 31));
DeleteUserButton.setBounds(new Rectangle(266, 238, 133, 32));
DeleteUserButton.setText("Delete User");
DeleteUserButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  DeleteUserButton_actionPerformed(e);
 }
});
DeleteFromMonthButton.setBounds(new Rectangle(439, 131, 75, 23));
DeleteFromMonthButton.setText("Delete");
DeleteFromMonthButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  DeleteFromMonthButton_actionPerformed(e);
 }
});
```

```
SendNEWSPanel.setLayout(null);
jLabel8.setText("Get Message to send All Users");
jLabel8.setBounds(new Rectangle(75, 41, 404, 27));
SendButton.setBounds(new Rectangle(241, 248, 140, 24));
SendButton.setText("Send to All !");
SendButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  SendButton_actionPerformed(e);
 }
});
ClearButton.setBounds(new Rectangle(387, 249, 100, 24));
ClearButton.setText("Clear");
ClearButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  ClearButton_actionPerformed(e);
 }
});
jLabel11.setText("Size of Mailbox");
jLabel11.setBounds(new Rectangle(57, 180, 127, 20));
StatisticPanel.setLayout(null);
jLabel12.setHorizontalAlignment(SwingConstants.CENTER);
jLabel12.setText("All Statistic of Mail Server");
jLabel12.setBounds(new Rectangle(184, 4, 148, 24));
jLabel13.setText("Login in today");
jLabel13.setBounds(new Rectangle(29, 31, 81, 17));
LoginTodayLabel.setBackground(Color.lightGray);
LoginTodayLabel.setToolTipText("");
LoginTodayLabel.setBounds(new Rectangle(146, 29, 43, 16));
jLabel15.setText("Login in This Month");
jLabel15.setBounds(new Rectangle(28, 49, 112, 18));
jLabel16.setText("Login in This Year");
jLabel16.setBounds(new Rectangle(27, 68, 112, 19));
```

```
LoginMonthLabel.setToolTipText("");
LoginMonthLabel.setBounds(new Rectangle(145, 50, 44, 15));
LoginYearLabel.setToolTipText("");
LoginYearLabel.setBounds(new Rectangle(146, 69, 43, 17));
jLabel19.setText("Statistic of User");
jLabel19.setBounds(new Rectangle(193, 108, 96, 20));
jLabel20.setText("login lasted");
jLabel20.setBounds(new Rectangle(198, 165, 81, 14));
jLabel21.setText("Size of Mailbox");
jLabel21.setBounds(new Rectangle(198, 150, 86, 15));
jLabel22.setText("This user have");
jLabel22.setBounds(new Rectangle(199, 196, 86, 17));
LoginUserLabel.setToolTipText("");
LoginUserLabel.setBounds(new Rectangle(303, 163, 96, 16));
SizeMailUserLabel.setToolTipText("");
SizeMailUserLabel.setBounds(new Rectangle(302, 197, 71, 17));
SizeUserLabel.setToolTipText("");
SizeUserLabel.setBounds(new Rectangle(304, 151, 37, 14));
ConfirmPasswordField2.setBounds(new Rectangle(193, 133, 156, 23));
PasswordPasswordField1.setBounds(new Rectangle(194, 92, 156, 23));
FullNameTextField.setBounds(new Rectangle(194, 56, 156, 21));
UserNameTextField.setBounds(new Rectangle(194, 21, 156, 21));
jScrollPane2.setBounds(new Rectangle(38, 45, 197, 223));
MonthComboBox.setBounds(new Rectangle(304, 130, 123, 25));
jScrollPane1.setBounds(new Rectangle(77, 67, 413, 171));
UserList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
StopWorkingButton.setEnabled(false);
StartWorkingButton.setBounds(new Rectangle(283, 243, 71, 24));
StartWorkingButton.setText("Start !");
StartWorkingButton.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(ActionEvent e) {
    StartWorkingButton_actionPerformed(e);
```

```
 }
});
jScrollPane3.setBounds(new Rectangle(28, 33, 477, 197));
jLabel7.setText("Status Working");
jLabel7.setBounds(new Rectangle(29, 7, 197, 21));
StopWorkingButton.setBounds(new Rectangle(367, 243, 66, 25));
StopWorkingButton.setText("Stop!");
StopWorkingButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  StopWorkingButton_actionPerformed(e);
 }
});
StatusWorkingPanel.setLayout(null);
jMenuItem1.setText("About");
ClearWorkingButton.setBounds(new Rectangle(435, 241, 66, 27));
ClearWorkingButton.setText("Clear");
ClearWorkingButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  ClearWorkingButton_actionPerformed(e);
 }
});
StatusWorkingTextArea.setBackground(Color.lightGray);
StatusWorkingTextArea.setEditable(false);
jLabel9.setText("----------------------------------------------------" +   "----");
jLabel9.setBounds(new Rectangle(19, 90, 421, 16));
jScrollPane4.setBounds(new Rectangle(42, 128, 142, 134));
jLabel17.setBounds(new Rectangle(199, 180, 70, 16));
jLabel17.setText("all login");
UserAllLoginLabel2.setBounds(new Rectangle(302, 177, 70, 16));
jLabel10.setText("Fullname");
jLabel10.setBounds(new Rectangle(200, 129, 67, 19));
ShowNameLabel.setBounds(new Rectangle(305, 127, 111, 19));
```

```
EditUserPanel.setLayout(null);
jScrollPane5.setBounds(new Rectangle(36, 32, 192, 228));
jTextField2.setBounds(new Rectangle(355, 66, 162, 26));
jLabel18.setBounds(new Rectangle(237, 68, 77, 21));
jLabel18.setText("Full Name");
jLabel110.setBounds(new Rectangle(237, 100, 77, 21));
jLabel110.setText("Password");
jLabel111.setBounds(new Rectangle(236, 130, 108, 21));
jLabel111.setText("Confirm Password");
jLabel112.setBounds(new Rectangle(238, 162, 106, 21));
jLabel112.setText("Size of MailBox");
SaveEditButton.setBounds(new Rectangle(311, 215, 176, 30));
SaveEditButton.setText("Save");
SaveEditButton.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(ActionEvent e) {
  SaveEditButton_actionPerformed(e);
 }
});
```

ประวัติผู้ทำโครงงาน

ชื่อ นายอารักษ์ คเชนทร์

ภูมิลำเนา 1690 ถนนประชาสงเคราะห์ ดินแดง กรุงเทพมหานคร

ประวัติการศึกษา

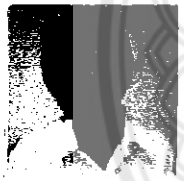
- จบระดับมัธยมศึกษาจากโรงเรียนสุรศักดิ์มนตรี
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4 สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

Email : goodgod_th@hotmail.com

ชื่อ นายเจน โพธิ์ลังกา

ภูมิลำเนา 7 หมู่ 23 ต.บ้านไร่ อ.ลาดยาว จ.นครสวรรค์

ประวัติการศึกษา



- จบระดับมัธยมศึกษาจากโรงเรียนลาดยาววิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4 สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

Email : cpejane@hotmail.com

ชื่อ นายรณเชษฐ เกตุเขียว

ภูมิลำเนา 219 หมู่5 ต.น้ำอ่าง อ.ตรอน จ.อตรดิตถ์

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนตรอนตรีสินธุ์
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4 สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

Email : Under_schola@hotmail.com