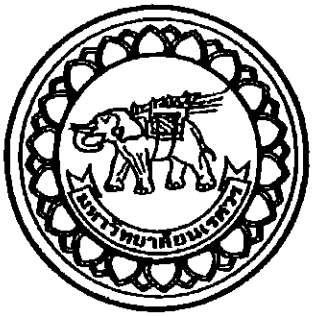



# การพัฒนาโปรแกรมสำหรับเครื่องคอมพิวเตอร์แม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์

## The Development of The Server Programming for Electronic Mail

นายกฤติพล หิรัญมาพร รหัส 42360420
นางสาวปัทมภรณ์ เจียตระกูล รหัส 42360578
นายไพรรัตน์ โพธิ์ศรี รหัส 42360784



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2545

# ใบรับรองโครงงานวิจัย

| | | | |
|---|---|---|---|
| **หัวข้อโครงงาน** | การพัฒนาโปรแกรมสำหรับเครื่องคอมพิวเตอร์แม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ | | |
| **ผู้ดำเนินโครงงาน** | นายกฤติพล | หิรัญมาพร | รหัส 42360420 |
| | นางสาวปัทมภรณ์ | เจียตระกูล | รหัส 42360578 |
| | นายไพรรัตน์ | โพธิ์ศรี | รหัส 42360784 |
| **อาจารย์ที่ปรึกษา** | อาจารย์ภาณุพงศ์ สอนคม | | |
| **สาขา** | วิศวกรรมคอมพิวเตอร์ | | |
| **ภาควิชา** | วิศวกรรมไฟฟ้าและคอมพิวเตอร์ | | |
| **ปีการศึกษา** | 2545 | | |

.....

**คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์**

**คณะกรรมการสอบโครงงานวิจัย**

.....ประธานกรรมการ
(อาจารย์สิทธิโชค เชาวกุล)

.....กรรมการ
(อาจารย์พรพิสุทธิ์ วรจิรันตน์)

.....กรรมการ
(อาจารย์รัฐภูมิ วรานุสาสน์)

.....กรรมการ
(อาจารย์ภาณุพงศ์ สอนคม)

| | | | |
|---|---|---|---|
| **หัวข้อโครงงาน** | **การพัฒนาโปรแกรมสำหรับเครื่องคอมพิวเตอร์แม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์** | | |
| **ผู้ดำเนินโครงงาน** | **นายกฤติพล** | **หิรัญมาพร** | **รหัส 42360420** |
| | **นางสาวปัทมภรณ์** | **เจียตระกูล** | **รหัส 42360578** |
| | **นายไพรรัตน์** | **โพธิ์ศรี** | **รหัส 42360784** |
| **อาจารย์ที่ปรึกษา** | **อาจารย์ภาณุพงศ์ สอนคม** | | |
| **สาขา** | **วิศวกรรมคอมพิวเตอร์** | | |
| **ภาควิชา** | **วิศวกรรมไฟฟ้าและคอมพิวเตอร์** | | |
| **ปีการศึกษา** | **2545** | | |

.....


## บทคัดย่อ

โครงงานนี้เป็นการศึกษา และพัฒนาโปรแกรมสำหรับใช้รันบนเครื่องคอมพิวเตอร์แม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ หรือเรียกอีกอย่างหนึ่งว่าโปรแกรมเมล์เซิร์ฟเวอร์ (Mail Server Program) ซึ่งเป็นโปรแกรมที่ให้บริการบนระบบเครือข่าย โดยโปรแกรมสามารถส่งจดหมายอิเล็กทรอนิกส์(e-mail) จากเครื่องคอมพิวเตอร์ลูกข่าย (client) ด้วยการติดต่อกับโปรแกรมเมล์เซิร์ฟเวอร์ที่คอมพิวเตอร์แม่ข่าย (Server) การพัฒนาโปรแกรมนี้ใช้ภาษาจาวา (JAVA) ในการพัฒนาซึ่งสามารถรันได้ทุกระบบปฏิบัติการ โปรแกรมนี้อ้างอิงการทำงานตามโปรโตคอลเอสเอ็มทีพี อาร์เอฟซี 822 (RFC 822) และโปรโตคอลพ็อป3 อาร์เอฟซี 1939 (RFC 1939)

ผลที่ได้จากการทำโครงงานนี้ คือ ได้โปรแกรมสำหรับเครื่องคอมพิวเตอร์แม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์บนระบบเครือข่าย ตามโปรโตคอลเอสเอ็มทีพี อาร์เอฟซี 822 (RFC 822) และโปรโตคอลพ็อป3 อาร์เอฟซี 1939 (RFC 1939) ซึ่งเป็นมาตรฐานในการรับส่งจดหมายอิเล็กทรอนิกส์ในปัจจุบัน

| | | | |
|---|---|---|---|
| **Project Title** | The Development of The Server Programming for Electronic Mail | | |
| **Name** | Mr. Kridpon | Hiranmaporn | ID. 42360420 |
| | Miss Pattamaporn | Jiatrakul | ID. 42360578 |
| | Mr. Pirut | Phosri | ID. 42360784 |
| **Project Advisor** | Mr. Panupong | Sornkom | |
| **Major** | Computer Engineering | | |
| **Department** | Electrical and Computer Engineering | | |
| **Academic Year** | 2002 | | |


## ABSTRACT

This project is studied and developed a program for server to get the electronic mail. In another way, called Mail Server Program that gives service to computer network. This program can send electronic mail from client by connecting with the program mail server at the server. This program uses the JAVA language. It can run on any operating system servers. This program refer about protocol SMTP RFC 822 and protocol POP3 RFC 1939

The results of this project is the program for server to get on the electronic mail for sending and getting electronic mail on network. In the present, that based protocol SMTP RFC 822 and protocol POP3 RFC 1939, which is the standard of electronic mail sending and getting.

# กิตติกรรมประกาศ

โครงงานชิ้นนี้ต้องอาศัยความรู้เพิ่มเติมมากมาย และต้องการการชี้แนะอย่างสูง ทางคณะผู้จัดทำจึงขอขอบพระคุณ อาจารย์ภาณุพงศ์ สอนคม ที่ให้คำปรึกษาและชี้แนะสิ่งต่างๆมากมาย และขอบคุณผู้ที่คอยให้การสนับสนุนและให้กำลังใจด้วยดีตลอดมา

นายไพรรัตน์ โพธิ์ศรี
นางสาวปิทมภรณ์ เจียตระกูล
นายกฤติพล หิรัญมาพร

# สารบัญ

หน้า

# สารบัญ(ต่อ)

หน้า

# สารบัญตาราง

ตารางที่ หน้า

# สารบัญรูป

# สารบัญรูป(ต่อ)

# บทที่ 1
# บทนำ

## 1.1 ที่มาและความสำคัญของโครงงาน

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

โปรแกรมสำหรับเครื่องแม่ข่ายในการรับส่งจดหมายอิเล็คทรอนิสก์ในประเทศไทย มิได้มีการพัฒนาอย่างกว้างขวาง ทำให้ความรู้ทางด้านนี้ยังไม่เป็นที่แพร่หลายมากนัก ทำให้ต้องใช้โปรแกรมจากต่างประเทศ ต้องสูญเสียเงินไปเป็นจำนวนมาก และเป็นสาเหตุนึงที่ทำให้เกิดปัญหาการละเมิดลิขสิทธิ์ทรัพย์สินทางปัญญา

ทางคณะผู้เสนอโครงการ จึงเล็งเห็นความสำคัญ ในการพัฒนาโปรแกรมสำหรับเครื่องแม่ข่ายเพื่อรับส่งจดหมายอิเล็กทรอนิกส์ (mail server) เพื่อนำไปใช้เป็นกรณีศึกษาและเป็นต้นแบบ กระจายความรู้ให้กับคนไทยซึ่งจะทำให้เกิดซอฟท์แวร์ (software) ของไทย และลดการละเมิดลิขสิทธิ์ทรัพย์สินทางปัญญา

## 1.2 วัตถุประสงค์ของโครงงาน

1. สามารถสร้างโปรแกรมสำหรับเครื่องเครื่องแม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ได้
2. ลดการสูญเสียเงินให้กับต่างประเทศ
3. เป็นแนวทางในการศึกษาและพัฒนาโปรแกรม ทำให้เกิดซอฟท์แวร์ของคนไทย
4. ลดการละเมิดลิขสิทธิ์ทรัพย์สินทางปัญญา

## 1.3 ขอบข่ายการทำงาน

ออกแบบและสร้างโปรแกรมสำหรับเครื่องแม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์

โปรแกรมสำหรับเครื่องแม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ที่สร้างขึ้นสามารถรองรับการติดต่อจากอีเมล์ลูกข่าย (e-mail client) ได้ ภายใต้มาตรฐานการรับส่งข้อมูลโปรโตคอล พ็อป 3 (POP3 : Post Office Protocol)

3. โปรแกรมสำหรับเครื่องแม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ที่สร้างขึ้น สามารถติดต่อกับแม่ข่ายอื่น ๆ ได้ ภายใต้มาตรฐานการรับส่งข้อมูลเอสเอ็มทีพี (SMTP : Simple Mail Transfer Protocol)

สามารถติดตั้งโปรแกรมสำหรับเครื่องแม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ที่สร้างขึ้นในเครื่องแม่ข่ายขององค์กร แล้วเครื่องลูกข่ายในองค์กรนั้น ๆ สามารถรับส่งจดหมายอิเล็กทรอนิกส์ได้

## 1.4 ขั้นตอนการทำงาน

1. เขียนโครงร่างการทำงาน
2. ศึกษาข้อมูลต่าง ๆ ดังต่อไปนี้
   - มาตรฐานโปรโตคอลเอสเอ็มทีพีและพ็อป3 และไอพีแอดเดรส (IP address)
   - เทคโนโลยีอาร์เอฟซี 822 (RFC 822) , อาร์เอฟซี 1939 (RFC 1939)
   - การสื่อสารระหว่างโปรเซส (Inter process communication)
   - การสร้างซ็อกเก็ต (socket)
   - การพัฒนาโปรแกรมบนทุกระบบปฏิบัติการ
3. วิเคราะห์และออกแบบโปรแกรม อัลกอริทึม โครงสร้างการทำงาน โดยรวบรวมตามความต้องการที่ได้ศึกษาจากข้อ 1 แล้วนำมาประยุกต์หรือดัดแปลง
4. พัฒนาโปรแกรมและคู่มือการใช้ โดยโปรแกรมจะถูกพัฒนาบนเครื่องคอมพิวเตอร์ส่วนบุคคล ซึ่งทำงานภายใต้ทุกสภาวะแวดล้อม (โดยใช้ภาษาจาวาในการพัฒนาโปรแกรม)
5. ทดสอบโปรแกรม
6. ปรับปรุงแก้ไขโปรแกรม
7. จัดทำเอกสาร
8. ส่งโครงงานฉบับสมบูรณ์

**ตารางที่ 1.1** กิจกรรมการดำเนินงาน

| กิจกรรม | พ.ย. | ธ.ค. | ม.ค. | ก.พ. | มี.ค. | เม.ย. | พ.ค. | มิ.ย. | ก.ค. | ส.ค. | ก.ย. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1. เขียนโครงร่างการทำงาน | ◄► | | | | | | | | | | |
| 2.ศึกษาข้อมูล | ◄ | — | — | ► | | | | | | | |
| 3.วิเคราะห์ออกแบบ | | | ◄ | — | ► | | | | | | |
| 4.พัฒนาโปรแกรมและคู่มือการใช้ | | | | | ◄ | — | — | — | — | ► | |
| 5.ทดสอบโปรแกรม | | | | | | | ◄ | — | — | ► | |
| 6.ปรับปรุงแก้ไขโปรแกรม | | | | | | | | ◄ | — | ► | |
| 7.จัดทำเอกสาร | | | | | | | | ◄ | — | — | ► |
| 8.ส่งโครงงานฉบับสมบูรณ์ | | | | | | | | | | | ◄► |

## 1.5 ผลที่คาดว่าจะได้รับ

1. โปรแกรมสำหรับเครื่องแม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์
2. เป็นโปรแกรมต้นแบบที่เป็นประโยชน์ในการใช้งาน และการศึกษาค้นคว้าต่อไป

## 1.6 งบประมาณ

| | | |
|---|---|---|
| 1. ค่าหนังสือ | 400 | บาท |
| 2. ค่าจ้างถ่ายเอกสาร และจัดรูปเล่มรายงาน | 1,200 | บาท |
| 3. ค่าจ้าง SCAN รูป | 100 | บาท |
| 4. ค่าวัสดุคอมพิวเตอร์ | 1,300 | บาท |
| รวมค่าใช้จ่าย | 3,000 | บาท (สามพันบาทถ้วน) |

**หมายเหตุ** ขออนุมัติถัวเฉลี่ยทุกรายการ

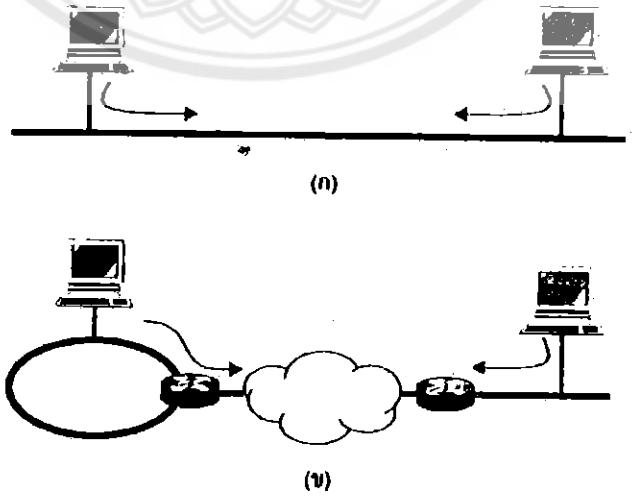
# บทที่ 2
# หลักการและทฤษฎี

ในบทที่ 2 นี้ เป็นเนื้อหาของหลักการและทฤษฎีในการทำโปรแกรมสำหรับเครื่องคอมพิวเตอร์แม่ข่ายเพื่อรับส่งจดหมายอิเล็กทรอนิกส์ เนื่องจากในการทำโปรแกรมนี้ จะต้องติดต่อกันข้ามเครือข่ายและผ่านเครือข่ายอินเทอร์เน็ต (internet) ซึ่งต้องใช้มาตรฐานการรับส่งข้อมูลต่าง ๆ จึงมีความจำเป็นที่ต้องศึกษาและทำความเข้าใจกลไกการทำงานของการติดต่อกันระหว่างเครือข่ายและมาตรฐานต่าง ๆ ให้เข้าใจ ก่อนที่จะนำไปประยุกต์ในการพัฒนาโปรแกรมต่อไป

## 2.1 การเชื่อมโยงเครือข่าย

จุดประสงค์การเชื่อมโยงคอมพิวเตอร์เข้าเป็นเครือข่ายคือต้องการให้คอมพิวเตอร์สามารถสื่อสารและแลกเปลี่ยนข้อมูลระหว่างกันได้ เครือข่ายคอมพิวเตอร์เริ่มจากเครือข่ายขนาดเล็กภายในองค์กรที่เชื่อมโยงกันภายใต้สภาพพื้นที่จำกัดซึ่งเรียกว่า เครือข่ายเฉพาะที่ (LAN : Local Area Network) เมื่อเชื่อมเครือข่ายย่อยเข้าด้วยกันและขยายขอบเขตครอบคลุมพื้นที่ระหว่างเมือง หรือระหว่างประเทศ ก็จะเรียกเครือข่ายนั้นว่า เครือข่ายพื้นที่กว้าง (WAN : Wide Area Network)

เครือข่ายยุคเริ่มต้นมีสถานีที่ใช้ฮาร์ดแวร์ (Hardware) ประเภทเดียวกันและสามารถทำงานร่วมกันได้อย่างกลมกลืน ต่อมาเมื่อมีเทคโนโลยีเครือข่ายเพิ่มขึ้น เช่น อีเทอร์เน็ตและโทเค็นริง ปัญหาที่เกิดขึ้นคือจะเชื่อมเครือข่ายต่างเทคโนโลยีเข้าด้วยกันได้อย่างไร โดยไม่จำกัดว่าคอมพิวเตอร์จะอยู่ในเครือข่ายเดียวกัน หรือต่างเครือข่ายกันโดยอาจใช้ฮาร์ดแวร์เครือข่ายต่างกัน




**รูปที่ 2.1** การเชื่อมโยง (ก) ในเครือข่ายเดียวกัน (ข) ต่างเครือข่ายกัน [2]

## 2.2 โปรโตคอลและไอพีแอดเดรส (Protocol and IP Address)

ในการใช้งานเครือข่ายคอมพิวเตอร์ทั่วไปหรือในเครือข่ายอินเตอร์เน็ตก็ตาม จะมีการส่งผ่านข้อมูลไปมาระหว่างเครื่องคอมพิวเตอร์หรือข้ามเครือข่ายออกไป ระบบคอมพิวเตอร์ที่เชื่อมต่ออยู่ในแต่ละเครือข่ายอาจจะใช้ฮาร์ดแวร์และซอฟร์แวร์ที่เหมือนหรือกันหรือต่างกันได้ ดังนั้นการที่จะทำให้สามารถส่งผ่านข้อมูลถึงกันและตีความได้อย่างถูกต้องตรงกันจะต้องมีการติดต่อกันให้ตรงกัน เปรียบเสมือนกับการสื่อสารกันของมนุษย์เรา ถ้าเราต้องการจะติดต่อกับผู้คนต่างเชื้อชาติต่างภาษากันให้เข้าใจกันได้ถูกต้องตรงกันจะต้องมีการกำหนดกลไกในการสื่อสารกันเสียก่อน เรียกว่าจะต้องกำหนดระเบียบวิธีในการติดต่อกันให้ตรงกัน เปรียบเสมือนกับการสื่อสารกันของมนุษย์เรา ถ้าเราต้องการจะติดต่อกับผู้คนต่างเชื้อชาติต่างภาษากันให้เข้าใจกันได้ถูกต้องตรงกัน ก็จะต้องตกลงกำหนดกันเสียก่อนว่า จะติดต่อกันอย่างไร ด้วยภาษาใดที่จะเข้าใจกันได้ เช่น ปัจจุบันมีการใช้ภาษาอังกฤษเป็นภาษากลางในการติดต่อกันมาก ทำให้เราพูดได้ว่า ภาษาอังกฤษเปรียบเสมือนเป็นภาษามาตรฐานในการสื่อสารของมนุษย์ได้ ถ้าพูดในแง่การสื่อสารข้อมูล เราก็พูดได้ว่า ภาษาอังกฤษเป็นโปรโตคอลในการสื่อสารของมนุษย์ที่มีการใช้งานอย่างแพร่หลาย

### 2.2.1 โปรโตคอลคืออะไร

โปรโตคอล คือ ระเบียบวิธีที่กำหนดขึ้นสำหรับสื่อสารข้อมูล ให้สามารถส่งผ่านข้อมูลไปยังปลายทางได้อย่างถูกต้อง ในปัจจุบันโปรโตคอลในการสื่อสารข้อมูลก็มีหลายโปรโตคอล

### 2.2.2 ไอพีแอดเดรสคืออะไร

ไอพีแอดเดรส คือ หมายเลข ที่ถูกกำหนดขึ้นมาให้เป็นหมายเลขอ้างอิงประจำตัวของอุปกรณ์ต่าง ๆ ที่เชื่อมต่ออยู่ในเครือข่ายอินเตอร์เน็ต โดยการกำหนดหมายเลขไอพีแอดเดรส ให้แต่ละเครื่องหรือแต่ละอุปกรณ์นี้จะต้องไม่ซ้ำกัน ซึ่งหมายเลขไอพีแอดเอรส นี้จะไม่ถูกผูกติดกับตัวฮาร์ดแวร์แต่อย่างใด จึงสามารถกำหนดใหม่หรือแก้ไขเปลี่ยนแปลงได้เมื่อมีการเปลี่ยนตัวฮาร์ดแวร์

การทำงานของโปรโตคอล ไอพี จำเป็นต้องอาศัยหมายเลขไอพีแอดเดรสนี้ เพื่อระบุและอ้างถึงอุปกรณ์ต่าง ๆ ที่ต่ออยู่ในเครือข่ายไม่ว่าจะเป็นเว็บเซิร์ฟเวอร์ เมล์เซิร์ฟเวอร์ อุปกรณ์เร้าเตอร์ (Router) ฯลฯ หมายเลขไอพีแอดเดรส จะเป็นค่าตัวเลขขนาด 32 บิต ถูกแบ่งออกเป็นส่วนละ 8 บิต รวมเป็น 4 ส่วนและคั่นแต่ละส่วนด้วยเครื่องหมาย (.) ดังนั้นค่าตัวเลขในแต่ละส่วนจะมีได้ตั้งแต่ 0 ถึง 255 ($2^8$) ตัวอย่างของไอพีแอดเดรส เช่น 205.144.78.1 หรือ 10.0.0.1 เป็นต้น

ค่าของไอพีแอดเดรส จะถูกกำหนดออกเป็น 2 ความหมายคือ ค่าของหมายเลขอุปกรณ์ในเครือข่าย (host address) และค่าของหมายเลขเครือข่าย (network address) ตัวอย่างเช่น มีเครื่องเว็บเซิร์ฟเวอร์เชื่อมต่ออยู่ในเครือข่าย 2 เครื่อง โดยแต่ละเครื่องมีไอพีแอดเดรส ประจำตัวคือ 205.144.78.2 และ

205.144.78.3 ตามลำดับ แสดงว่าเครื่องทั้งสองต่อเชื่อมอยู่ในเครือข่ายเดียวกัน บนสายสัญญาณที่เชื่อมโยงเส้นเดียวกันแต่มีหมายเลขประจำตัวเครื่องที่แตกต่างกันคือ 2 และ 3 ตามลำดับ [3]

## 2.3 โปรโตคอลทีซีพี/ไอพี (TCP/IP)

ทีซีพี/ไอพีเป็นโปรโตคอลที่ใช้งานอย่างแพร่หลายในแทบทุกเครือข่ายไม่ว่าจะเป็นเครือข่ายเฉพาะที่หรือเครือข่ายในบริเวณกว้าง ทีซีพี/ไอพีเชื่อมกลุ่มเครือข่ายย่อยเข้าด้วยกันเป็นเครือข่ายขนาดใหญ่หรือ อินเทอร์เน็ต

ทีซีพี/ไอพีผ่านการออกแบบให้เป็นอิสระจากชนิดคอมพิวเตอร์ ฮาร์ดแวร์ และระบบปฏิบัติการ กลไกของโปรโตคอลมีความเชื่อถือได้สูงและทำงานได้แม้ในบางภาวะที่การสื่อสารมีความผิดปกติ รวมทั้งสามารถเลือกเส้นทางส่งข้อมูลตามสภาพเครือข่ายได้ในกรณีที่บางเส้นทางชำรุด

ชื่อทีซีพี/ไอพีมีที่มาจากโปรโตคอลสองโปรโตคอลคือ ทีซีพี (TCP : Transmission Control Protocol) และ ไอพี ( IP : Internet Protocol) ไอพีทำหน้าที่กำหนดแอดเดรส จัดแบ่งขนาดข้อมูลให้พอเหมาะ และเลือกเส้นทางส่งข้อมูล ส่วนทีซีพีมีหน้าที่รับประกันความถูกต้องในการลำเลียงข้อมูล

ทีซีพีและไอพีไม่ได้เป็นเพียงสองโปรโตคอลที่มีอยู่เท่านั้น หากแต่ยังมีโปรโตคอลสนับสนุนอีกเป็นจำนวนมากและจัดรวมกันเป็น ชุดโปรโตคอลทีซีพี/ไอพี (TCP/IP protocol suite)

## 2.4 อินเทอร์เน็ต

ในปัจจุบันทั่วโลกให้ความสำคัญกับ เทคโนโลยีสารสนเทศ (Information Technology) หรือเรียกโดยย่อว่า ไอที ซึ่งหมายถึงความรู้ในวิธีจัดเก็บ รวบรวม เรียกใช้ และนำเสนอข้อมูลด้วยอุปกรณ์ประมวลผล เครื่องมือที่จำเป็นต้องใช้สำหรับงานไอทีคือคอมพิวเตอร์และอุปกรณ์สื่อสารโทรคมนาคม ตลอดจนโครงสร้างพื้นฐานด้านการสื่อสารไม่ว่าจะเป็นระบบโทรศัพท์ ดาวเทียม หรือเครือข่ายคอมพิวเตอร์

อินเทอร์เน็ตเป็นเครื่องมือสำคัญอย่างหนึ่งสำหรับไอที หากเราจำเป็นต้องอาศัยข้อมูลข่าวสารในการทำงานประจำวัน อินเทอร์เน็ตเป็นช่องทางให้เข้าถึงข้อมูลที่ต้องการได้ในเวลาอันรวดเร็ว

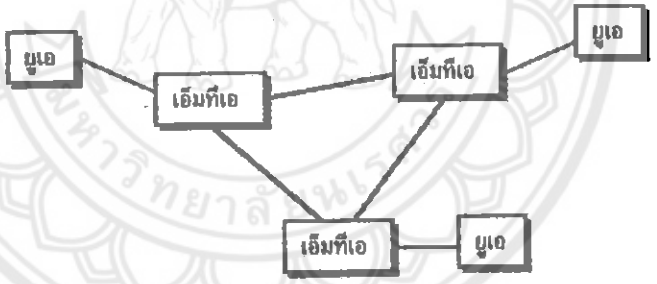
อินเทอร์เน็ตเป็นแหล่งรวบรวมข้อมูลแหล่งใหญ่ที่สุดของโลกและเป็นที่รวมทั้งบริการและเครื่องมือสืบค้นข้อมูลหลากหลายประเภท จนกระทั่งกล่าวได้ว่าอินเทอร์เน็ตเป็นเครื่องมือสำคัญอย่างหนึ่งในการประยุกต์ใช้เทคโนโลยีสารสนเทศทั้งระดับบุคคลและองค์กร ซึ่งบริการในอินเทอร์เน็ตสามารถแบ่งออกได้หลายประเภท เช่น ใช้โปรแกรมบนเครื่องคอมพิวเตอร์อื่น (Remote Login) ถ่ายโอนแฟ้มข้อมูล (ftp) สนทนาทางเครือข่าย (chat , icq) บริการสืบค้นข้อมูล (whois) กระดานข่าว (web broad) เว็บ (WWW : World-Wide WebX) จดหมายข่าวจดหมายเวียน และ ไปรษณีย์อิเล็กทรอนิกส์หรืออีเมล์ (e-mail)

## 2.5 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 2.2** สถาปัตยกรรมระบบเมล์ในทีซีพี/ไอพี

การจัดแบ่งออกเป็นยูเอและเอ็มทีเอมีข้อดีคือ แยกงานของทั้งสองส่วนให้เป็นอิสระจากกัน หน้าที่ของยูเอเน้นการทำงานกับผู้ใช้เพื่อให้ผู้ใช้อ่านเขียนจดหมายได้อย่างสะดวกโดยไม่ต้องยุ่งเกี่ยวกับการทำงานระดับล่างของโปรโตคอล ส่วนเอ็มทีเอทำงานตามเอสเอ็มทีพี เช่น การตรวจสอบความถูกต้องของแอสเดรสผู้รับผู้ส่ง รวมทั้งการหาเส้นทางและนำส่งจดหมายไปยังปลายทาง [3]

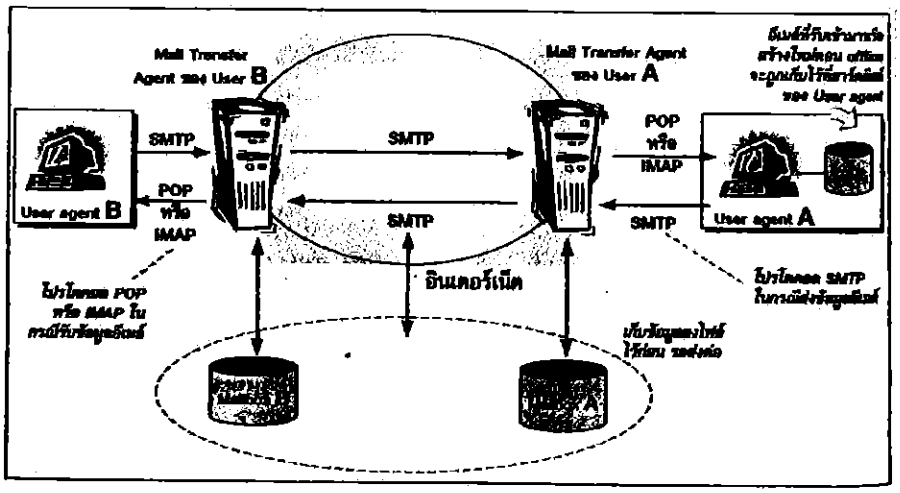
## 2.6 อีเมล์ และโปรโตคอลของอีเมล์

อีเมล์หรือจดหมายอิเล็กทรอนิกส์ได้เริ่มใช้งานกันมานานแล้ว ตั้งแต่ยุคของเครื่องเมนเฟรมหรือมินิคอมพิวเตอร์ ซึ่งไอบีเอ็มได้พัฒนาระบบอีเมล์ที่เรียกว่า PROFS (Professional Office System) ออกมาใช้งาน นอกจากนี้ก็มีบนระบบยูนิกส์ (UNIX) ต่อมาหลาย ๆ ค่าย ก็ได้พัฒนาระบบอีเมล์ของตนขึ้นมา โดยส่วนใหญ่จะเป็นองค์ประกอบในแอพพลิเคชันที่ทำงานบนระบบเน็ตเวิร์ก เช่น ไมโครซอร์ฟเมล์ (Microsoft Mail) ของไมโครซอฟท์ และ ซีซีเมล์ (cc:Mail) ของโลตัส เป็นต้น ซึ่งต่างก็ใช้เทคโนโลยีของตนเองและเป็นระบบปิด ดังนั้นการส่งเมล์ไปยังอีกผู้ใช้ที่ใช้ระบบเมล์คนละค่ายกันจึงเป็นเรื่องที่ยุ่งยาก

ในยุคต่อมาที่ระบบเน็ตเวิร์กทั้ง แลน (LAN) และ แวน (WAN) ต่างมีมาตรฐาน และเป็นระบบเปิด (Open System) มากขึ้น ก็ได้ปรับเปลี่ยนการทำงานของอีเมล์มาเป็นแบบไคลเอนต์เซิร์ฟเวอร์ที่เป็นพื้นฐานแบบที่ใช้กันในระบบยูนิกส์ และมีการพัฒนาอีเมล์เซิร์ฟเวอร์ขึ้นมาโดยเฉพาะ เช่น เอ็กซ์เชนจ์เซิร์ฟเวอร์ (Exchange Server) ของไมโครซอฟท์ หรือ โน้ตเซิร์ฟเวอร์ (Note Server) ของโลตัส เป็นต้น ซึ่งผู้ใช้จะติดต่อเข้าสู่อีเมล์เซิร์ฟเวอร์ได้ทั้งโดยการผ่านระบบแลน หรือใช้โมเด็มเข้ามาจากแวน ทำให้ผู้ใช้จะไม่เห็นไฟล์ในฮาร์ดดิสก์บนเซิร์ฟเวอร์เลย ดังนั้นความปลอดภัยของระบบจึงมีมากขึ้น จนในปัจจุบันได้พัฒนาขึ้นมาเป็นระบบเวิร์กโฟล (Workflow) ที่ใช้อีเมล์เป็นพื้นฐาน การทำงานของอีเมล์ไคลเอนต์และอีเมล์เซิร์ฟเวอร์มีส่วนประกอบดังนี้

- ยูสเซอร์เอเจนต์ (User Agent) เป็นโปรแกรมคอมพิวเตอร์ทางด้านผู้ใช้งาน แบ่งเป็น 2 ส่วน คือ ส่วนของผู้ส่งและส่วนของผู้รับ โดยยูสเซอร์เอเจนต์ นี้จะติดต่อเข้าสู่เซิร์ฟเวอร์ของตนโดยผ่านระบบแลน หรือ ไดแอ็ล-อัพ (Dial-up) ซึ่งในส่วนของยูสเซอร์เอเจนต์นี้จะเป็นส่วนที่ผู้ใช้ติดตั้งโปรแกรมไคลเอนต์ของอีเมล์เพื่อเรียกใช้บริการอีเมล์ เช่น เอาท์ลุกค์ เอ็กซ์เพลส (Outlook Express) หรือ ยูโดรา (Eudora)เป็นต้น

- เอ็มทีเอ (MTA : Mail Tranfer Agent) เป็นโปรแกรมคอมพิวเตอร์ที่จะส่งอีเมล์จากต้นทางไปยังผู้รับปลายทาง ซึ่งจะต้องส่งผ่านเครื่องจำนวนมากเป็นทอด ๆ จนไปถึงเครื่องที่มี แอ็คเคาท์ (account) หรือเมล์บ็อกซ์ของผู้รับ และหากไม่สามารถส่งเมล์ถึงผู้รับได้ (เพราะใส่ชื่อผิด) ยังทำหน้าที่ส่งข้อผิดพลาดของจดหมายอิเล็กทรอนิกส์ (error mail) กลับมาส่งผู้ส่งได้อีก ซึ่งเครื่องที่มีเอ็มทีเอทำงานอยู่มักจะมีเมล์บ็อกซ์ของผู้ใช้อยู่ด้วย ซึ่งเป็นกล่องเก็บจดหมายอิเล็กทรอนิกส์ที่สำคัญที่สุด (primary mailbox) และเรียกเครื่องนั้นว่าเมล์เซิร์ฟเวอร์ (Mail Server)




**รูปที่ 2.3** องค์ประกอบและโปโตคอลต่างๆที่ใช้งานอยู่ในระบบการทำงานของอีเมล์ผู้ส่งจะใช้เครื่องของยูสเซอร์เอเจนต์ บันทึกอีเมล์และส่งอีเมล์ไปยังเครื่องที่มีเอ็มทีเอทำงานอยู่ ซึ่งจะส่งต่อไปเครื่องอื่นๆที่มีเอ็มทีเอเป็นทอดๆจนถึงเครื่องที่มีแอ็คเคาน์ท(account) หรือเมล์บ็อกนั้นอยู่ [3]

### 2.6.2 อีเมล์แอดเดรส

ในการส่งเมล์ให้กับผู้รับนั้น การระบุชื่อผู้รับปลายทางนั้นมักจะเรียกกันว่า อีเมล์แอดเดรส โดยปกติหากผู้รับมีกล่องเก็บจดหมายอิเล็กทรอนิกส์อยู่ในเครื่องเดียวกันกับผู้ส่ง ก็สามารถระบุชื่อผู้ใช้หรือยูสเซอร์แอ็คเคาน์ท (user account) ของผู้รับนั้นเป็นอีเมล์แอดเดรสได้เลย แต่หากผู้รับไม่ได้อยู่ในเครื่องเดียวกัน การระบุแอดเดรสปลายทางก็จะซับซ้อนขึ้น

การระบุอีเมล์แอดเดรสของผู้รับโดยทั่วไปถ้าส่งผ่านอินเตอร์เน็ต จะใช้วิธีที่เรียกว่า location-independent คือให้ระบบหาเอาเองว่าผู้รับอยู่ที่ใด ตัวอย่างเช่น edito@provision.co.th โดยอีดิเตอร์ (editor) เป็นชื่อผู้รับหรือแอ็คเคาน์ท ที่กำหนดให้ใช้ได้ในเครื่องนั้น ส่วน provision.co.th เป็นชื่อโดเมน ซึ่งอาจเป็นเพียงเครื่อง ๆ หนึ่งหรือเครือข่ายที่มีเครื่องหลายเครื่องที่เชื่อมต่อเข้าด้วยกัน ในลักษณะที่เป็นเครือข่ายนี้บางระบบอาจจำเป็นต้องระบุชื่อเครื่องที่ผู้ใช้นั้นอยู่ คือต้องระบุโฮสต์ (host) ไปด้วย แต่บางระบบก็อาจระบุชื่อโดเมนก็เพียงพอแล้ว เพราะระบบจะหาให้เองว่าผู้รับอยู่ในเครื่องใดในเครือข่ายนั้น [4]

### 2.6.3 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 เวลาที่ปรากฏอยู่ถือว่าเป็นเวลาท้องถิ่นของโฮสต์ที่รับจดหมายเทียบกับเวลามาตรฐานกรีนิช ตัวเลข +0700 บ่งบอกว่าโฮสต์ nontri.ku.ac.th มีเวลาเร็วกว่ามาตรฐานกรีนิช 7 ชั่วโมง

```
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:
<3.0.2.16.19980309105130.22b778588@mcb.mcb.co.uk>
```

บรรทัดที่กำกับด้วย Received ถัดมาบ่งบอกว่าโฮสต์ mcb.mcb.co.uk ได้รับจดหมายมาจาก pc0107.mcb.co.uk ด้วยโปรโตคอล SMI

X-Sender: cheenan@mcb.co.uk

ใช้บ่งบอกถึงผู้ส่ง

X-Mailer: QUALCOMM Windows Eudora Light Version 3.0.2 (16)

ใช้บ่งบอกถึงโปรแกรมที่ใช้

From: Cris Keenan <ckeenan@mcb.co.uk>

บรรทัดนี้แสดงว่าผู้ส่งจดหมายคือ Chris Keenan และที่อยู่ของผู้ส่งคือ ckeenan@mcb.co.uk

Subject: Internet Research

บรรทัดที่ขึ้นต้นด้วย Subject: เป็นหัวเรื่องอ้างอิงถึงเนื้อความของจดหมาย

To: nguan@ku.ac.th

บรรทัดที่ขึ้นต้นด้วย To: บอกถึงชื่อผู้รับจดหมาย

Date: Mon, 09 Mar 1998 10:51:30

บรรทัดนี้บอกเวลาที่นำส่งจดหมาย

Mime-Version: 1.0

Content-Type: text/plain; charset = "us-ascii"

กำหนดการส่งเนื้อความจดหมายตามมาตรฐาน MIME [2]

## 2.7 โปรโตคอลและประเภทการใช้งาน

การทำงานทั่ว ๆ ไป ของอีเมล์โดยสรุปมีเพียง 2 ประเภท คือ การส่งอีเมล์ และการรับอีเมล์ โดยโปรโตคอลเอสเอ็มทีพี (SMTP : Simple Mail Transfer Protocol) จะใช้ขณะที่ยูสเซอร์เอเจนต์ ส่งอีเมล์มาที่เอ็มทีเอ (เฉพาะแบบ อ็อฟไลน์(Offline)) และใช้ขณะรับและส่งอีเมล์ระหว่างเอ็มทีเอด้วยกัน สำหรับการใช้เมล์แบบอ็อฟไลน์ คือ เครื่องที่ผู้ใช้ใช้อ่านเมล์ไม่ได้ต่อกับเครื่องที่มีเมล์บ็อกซ์ตลอดเวลา อาจเลือกดาวน์โหลดเมล์มาเก็บไว้ที่เครื่องของตัวเองนั้น จะมีโปรโตคอลสำหรับรับอีเมล์ที่เกี่ยวข้องอีกที่ใช้งานกันแพร่หลายมีอยู่ 2 แบบ คือ โปรโตคอลพ็อป (POP : Post Office Protocol) และ ไอแม็ป (IMAP : Internet Message Access Protocol) ซึ่งจะทำหน้าที่ดาวน์โหลดหรืออัพโหลดอีเมล์จากเครื่องของผู้ใช้ไปยังเครื่องที่มีเอ็มทีเออยู่

รูปแบบของข้อมูลที่ใช้ในโปรโตคอลต่าง ๆ ของอีเมล์นี้ถูกกำหนดไว้ในอาร์เอฟซี 822 (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 ยูนิกส์ก็อาจจะใช้เพียง Carriage Return เท่านั้น) เป็นเครื่องหมายของการสิ้นสุดบรรทัด จะเห็นได้ว่าในส่วนของจ่าหน้าอีเมล์นี้มีข้อความที่จำเป็นคือรายละเอียดของผู้ส่งและผู้รับ ส่วนรายละเอียดอื่น ๆ เช่น รายชื่อผู้รับสำเนา (Cc) จะมีหรือไม่ก็ได้

มาถึงส่วนที่เป็นข้อมูลของอีเมล์ ซึ่งจะแบ่งย่อยออกได้เป็น 2 ส่วนคือ ส่วนหัว (header) และส่วนเนื้อความของอีเมล์ ส่วนหัวนี้จะถูกสร้างขึ้นอย่างอัตโนมัติโดย ของผู้ส่ง เพื่อให้เอ็มทีเอ 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

ตามข้อกำหนดอาร์เอฟซี 822 ในการส่งอีเมล์ผ่านอินเตอร์เน็ตนั้น แต่ละบรรทัดจะมีขนาดยาวได้ไม่เกิน 1,000 ไบต์ และขนาดของอีเมล์แต่ละครั้งจะไม่เกิน 64 กิโลไบต์ ซึ่งผู้ส่งไม่จำเป็นต้องสนใจว่าอีเมล์ที่ส่งไปนั้นจะผ่านไปที่ เอ็มทีเอ ใดบ้าง เนื่องจากอีเมล์จะถูกเข้ารหัสและส่งไปถึงยังยูสเซอร์เอเจนต์ ของผู้รับปลายทางและผ่านการถอดรหัสได้โดยอัตโนมัติ

จากองค์ประกอบของโปรโตคอล และวิธีการรับส่งอีเมล์ที่กล่าวผ่านมา ทำให้การใช้อีเมล์ในปัจจุบันซึ่งทำงานแบบไคลเอนต์เซิร์ฟเวอร์ สามารถทำงานได้ 3 แบบ คือ

- **แบบออฟไลน์ (Offline)** หรือเรียกว่า ดาวน์โหลดและลบ (Download and Delete) ซึ่งเป็นรูปแบบมาตรฐานทั่วไปในการใช้งานกับอีเมล์ของอินเตอร์เน็ต ซึ่งใช้โปรโตคอลพ็อป หรือ ไอแม็ปโดยยูสเซอร์เอเจนต์ ของผู้รับจะดาวน์โหลดอีเมล์ทั้งหมดมาจากอีเมล์บางโปรแกรม สามารถให้เลือกได้ว่าต้องการลบอีเมล์ที่ดาวน์โหลมาแล้วทางฝั่งเซิร์ฟเวอร์นั้นทิ้งหรือไม่) ทำให้ผู้ใช้สามารถอ่านอีเมล์นั้นได้ตลอดเวลา โดยไม่จำเป็นต้องติดต่อเมล์เซิร์ฟเวอร์อีก แต่ยูสเซอร์เอเจนต์ จะไม่รู้ว่ามีอีเมล์เข้ามาใหม่จนกว่าจะติดต่อเข้าไปยังเมล์เซิร์ฟเวอร์และดาวน์โหลดอีเมล์เข้ามาใหม่
- **แบบออนไลน์ (Online)** เป็นแบบที่อีเมล์ด้านยูสเซอร์เอเจนต์ ของผู้รบจะต้องติดต่อกับเมล์เซิร์ฟเวอร์ของผู้รับเองตลอดเวลาที่ใช้อีเมล์ ซึ่งระบบที่ให้บริการอีเมล์แบบนี้จะสามารถเปิดแชร์เมล์บ็อกซ์ที่เซิร์ฟเวอร์ได้ตลอดเวลา เช่น NFS (Network File System) หรือ CIFS (Common Internet File System) เป็นต้น นอกจากนี้โปรโตคอลแบบไอแม็ป ยังสามารถใช้งานในแบบออนไลน์ นี้ได้อีกด้วย
- **แบบดิสคอนเน็ค (Disconnected)** เป็นแบบผสมผสานระหว่างแบบออฟไลน์และแบบออนไลน์ โดยอาศัยเมล์เซิร์ฟเวอร์ของผู้รับเป็นหลักในการจัดเก็บข้อมูลของอีเมล์ และในส่วนเนื้อที่ของยูสเซอร์เอเจนต์ นี้จะเป็นที่เก็บอีเมล์สำรอง โดยเมื่อมีการดาวน์โหลดอีเมล์มาก็จะทำงานในแบบออฟไลน์ เพื่อลดภาระที่ต้องติดต่อกับเมล์เซิร์ฟเวอร์ตลอดเวลา แต่ข้อมูลอีเมล์จะไม่ถูกลบออกจากเมล์เซิร์ฟเวอร์ ผู้ใช้สามารถโหลดอีเมล์ที่แก้ไขแล้วกลับไปยังเมล์เซิร์ฟเวอร์ในภายหลังได้ เช่น การแก้ไขหรือตอบกลับอีเมล์ (Reply to) ที่ส่งมา เป็นต้น ซึ่งโปรโตคอลที่สามารถตอบสนองการใช้งานในแบบนี้ได้ก็คือ ไอแม็ป

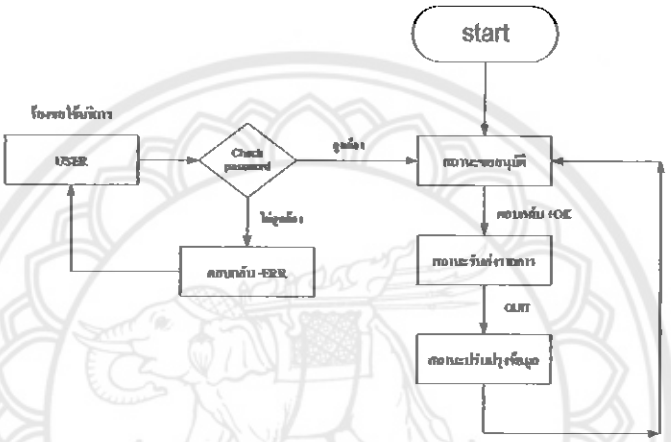
## 2.8 พ็อป 3 (POP3 : Post Office Protocol)

สำหรับผู้ที่ใช้งานอีเมล์บนอินเตอร์เน็ตมาแล้ว คงจะคุ้นเคยกับโปรโตคอลที่เรียกว่าพ็อป หรือ Post Office Protocol กันเป็นอย่างดี เพราะเป็นโปรโตคอลที่ทำหน้าที่โหลดอีเมล์มาจาก เอ็มทีเอ ไปยังยูสเซอร์เอเจนต์ ซึ่งในปัจจุบันได้พัฒนามาจนถึงเวอร์ชัน 3 แล้ว หรือเรียกย่อ ๆ ว่า พ็อป3 โปรโตคอลนี้เป็นตัวแรกที่ถูกออกแบบมาเพื่อใช้รับอีเมล์ และเพื่อให้สนับสนุนการทำงานในแบบออฟไลน์โดยติดต่อเข้าไปยังเมล์เซิร์ฟเวอร์แล้วดาวน์โหลดอีเมล์ทั้งหมดไว้ที่ ยูสเซอร์เอเจนต์ จากนั้นจะลบอีเมล์ที่เซิร์ฟเวอร์นั้นทิ้งไป เพื่อป้องกันการดาวน์โหลดซ้ำ แต่ผู้ใช้จะทำงานแบบออนไลน์ กับเซิร์ฟเวอร์ไม่ได้ เนื่องจากการอ่านอีเมล์จะดึงอีเมล์ที่เก็บไว้ใน ยูสเซอร์เอเจนต์ ขึ้นมาให้อ่านหลังจากที่ดาวน์โหลดมาเก็บไว้ ซึ่งในขณะนั้นอาจจะไม่ได้ออนไลน์อยู่กับเน็ตเวิร์กก็ได้

โปรโตคอลของ พ็อป3 นี้จะทำงานในแบบไคลเอนต์เซิร์ฟเวอร์ คือมีโปรแกรม พ็อปเซิร์ฟเวอร์ในเมล์เซิร์ฟเวอร์ และพ็อปไคลน์เอ็นต์ ในเครื่องผู้รับ ซึ่งปกติจะฝังอยู่ในโปรแกรมที่เป็น ยูสเซอร์เอ

เจนต์เลย โปรแกรมทั้งสองจะติดต่อกันโดยใช้ชุดคำสั่งที่เป็นรหัสแอสกี (ASCII) คือเมื่อด้านที่รับทำคำสั่งก็จะทำงานตามคำสั่งนั้น แล้วตอบกลับมาโดยมีค่าเป็น +OK หมายถึงทำงานได้เรียบร้อย หรือ –ERR หมายถึงเกิดปัญหาขึ้น

ทำงานไม่ได้ ซึ่งในคำสั่งที่ต้องมีการตอบกลับและส่งข้อมูลกลับมา โดยประกอบด้วยข้อมูลหลาย ๆ บรรทัดนั้น พ็อป3 จะให้บรรทัดสุดท้ายเป็นเครื่องหมายจุด (.) ตามด้วย Carriage Return และ Line Feed หมายถึงการสิ้นสุดชุดข้อมูล แต่ในกรณีที่ข้อมูลบรรทัดสุดท้ายมีข้อมูลที่เป็นจุดด้วย จะใช้เทคนิคที่เรียกว่า Character Stuffing เพื่อแก้ปัญหา โดยจะเติมจุดลงไปอีกหนึ่งตัว เพื่อเป็นตัวบ่งชี้ว่าข้อมูลนั้นเป็นจุด ซึ่งจะแตกต่างจากสัญลักษณ์แสดงการสิ้นสุดของข้อมูล




**รูปที่ 2.4** แผนผังลำดับงาน (Flow Chart) ของ พ็อป3

การทำงานของ พ็อป3 จะทำร่วมกับโปรโตคอลทีซีพีโดยทั่วไปจะใช้ที่**ซีพีพอร์ต** 110 การดำเนินการใดกับจดหมายจะเกิดกับจดหมายที่อ่านมาเก็บไว้ยังไคลเอ็นต์เท่านั้น การเปลี่ยนแปลงสถานะของจดหมายใด ๆ จะเกิดขึ้นที่ไคลเอ็นต์ที่ใช้เท่านั้น โดยไม่เปลี่ยนแปลงสถานะจดหมายที่เซิร์ฟเวอร์ หน้าที่หลักของพ็อปจึงเป็นการส่งถ่ายจดหมายจากเซิร์ฟเวอร์มายังไคลเอ็นต์ ในการติดต่อ ขั้นตอนการทำงานของ พ็อป3 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 และไม่ได้เป็นค่าแอสกีทั่วไป ไคลเอนต์จะใช้คำสั่ง APOP ทำงานแทนคำสั่ง USER และ PASS

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

- **สถานะปรับปรุงข้อมูล (Update State)** เมื่อ ยูสเซอร์เอเจนต์ เลิกใช้งานด้วยคำสั่ง QUIT ของ พ็อป3 เซิร์ฟเวอร์ก็จะเข้าสู่สถานปรับปรุงข้อมูล เพื่อลบอีเมล์ที่ดาวน์โหลดเรียบร้อยแล้วออกไป จากนั้นก็จะเข้าสู่สถานะขออนุมัติใหม่โดยอัตโนมัติ เพื่อรอรับการทำงานครั้งต่อไป

**ตาราง 2.1** แสดงรายละเอียดในคำสั่งต่าง ๆ ของ พ็อป 3 ที่ใช้งานอยู่ [3]

| คำสั่ง | พารามิเตอร์ | สถานะ | รายและเอียด |
|---|---|---|---|
| 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 | ใช้ตรวจสอบหมายเลขของอีเมล์ และขนาดของอีเมล์ |
| 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 อีเมล์เหล่านั้นจะถูกลบจากเมล์บ็อกซ์ในขั้นตอนนี้ |

**2.8.1 สถานะขออนุมัติ (Authorization State)**

เมื่อการเชื่อมต่อทีซีพีถูกเปิดโดยไคลเอ็นท์ พ็อป3 เครื่องแม่ข่าย พ็อป3 จะทำการทักทาย ซึ่งจะมีการตอบกลับเป็นบวก ดังตัวอย่าง:

S: +OK POP3 server ready

ขณะ พ็อป3 อยู่ในสถานะอำนาจที่ได้มอบหมายตอนนี้ ไคลเอ็นท์ต้องระบุและรับรองว่าเป็นสมาชิกในเซิร์ฟเวอร์ โดยการ USER และ PASS หรือรวมคำสั่งทั้งสองเข้าด้วยกันคือคำสั่ง APOP

เมื่อเครื่องแม่ข่าย พ็อป3 ใช้คำสั่งขออนุมัติ แล้วไคลเอ็นท์ควรจะเข้าไปที่เมล์ดร็อป (maildrop) ซึ่งเครื่องแม่ข่าย พ็อป3 จะอนุญาติให้เข้าได้เฉพาะเมล์ดร็อปของผู้ใช้นั้นๆ เพราะจำเป็นที่จะป้องกันข่าวสารจากการแก้ไขหรือการเคลื่อนย้าย ถ้าการล็อคอินสมบูรณ์ เครื่องแม่ข่าย พ็อป3 จะตอบสนองกลับเป็นสถานะบวก ส่วนพ็อป3 จะเข้าไปสถานะรับส่งรายการ ถ้าเมล์ดร็อปไม่สามารถถูกเปิดได้อาจจะเกิดจากเหตุผลใดเหตุผลหนึ่ง (เป็นต้นว่า การล็อคอินไม่สามารถเข้าได้ ไคลเอ็นท์ถูกปฏิเสธไม่ให้เข้าสู่เมล์ดร็อป ,หรือเมล์ดร็อปไม่สามารถถูกวิเคราะห์ได้) เครื่องแม่ข่าย พ็อป3 จะตอบสนองกลับเป็นสถานะลบ (ถ้าการล็อคอินได้ แต่เครื่องแม่ข่าย พ็อป3 ตั้งใจตอบสนองกลับเป็นลบ เครื่องแม่ข่ายพ็อป3 ปลดการล็อคครั้งก่อนและปฏิเสธคำสั่ง) หลังจากที่กลับค่าตัววัดสถานะเป็นลบ เครื่องแม่ข่ายอาจจะปิดการเชื่อมต่อ แต่ถ้าเครื่องแม่ข่ายไม่ทำการปิดการเชื่อมต่อไคลเอ็นท์อาจใส่คำสั่งขออนุมัติ และเริ่มต้นอีกครั้ง หรือไคลเอ็นท์อาจจะใช้คำสั่ง Quit

หลังจากเครื่องแม่ข่าย พ็อป3 ได้เปิดเมล์ดร็อป แล้วจะกำหนดตัวเลขให้ข่าวสารแต่ละอัน และขนาดของข่าวสารแต่ละอันในอ็อคเท็ต (octet) ข่าวสารแรกในเมล์ดร็อป ถูกกำหนดตัวเลขข่าวสาร "1", ตัวที่สองถูกกำหนด "2" และอื่นๆ จนถึงข่าวสาร n th ในเมล์ดร็อป จะถูกกำหนดตัวเลขข่าวสาร "n" ในคำสั่ง พ็อป3 และตอบสนอง, ข่าวสารทั้งตัวเลขและขนาดข่าวสารถูกแสดงใน เลขฐาน 10

**2.8.2 สถานะรับส่งรายการ (Transaction State)**

เมื่อไคลเอ็นท์ล็อคอินเข้าเครื่องแม่ข่ายพ็อป3 แล้ว เครื่องแม่ข่ายพ็อป3 ก็ล็อคและเปิดทางไปสู่เมล์ดร็อป พ็อป3 ในตอนนี้จะอยู่ในสถานะรับส่งรายการ ไคลเอ็นท์จะป้อนของคำสั่งต่างๆ ของ พ็อป3 ซึ่งหลังจากที่ป้อนคำสั่งแต่ละอัน เครื่องแม่ข่ายพ็อป3 ก็ตอบสนอง และเมื่อไคลเอ็นท์ใช้คำสั่ง QUIT และ พ็อป3 จะเข้าไปสู่สถานะ Update

**2.8.3 สถานะปรับปรุงข้อมูล (Update State)**

เมื่อไคลเอ็นท์ใส่คำสั่ง QUIT จากสถานะรับส่งรายการ พ็อป3 จะเข้าไปสถานะปรับปรุงข้อมูล ถ้าไคลเอ็นท์ใส่คำสั่ง QUIT จากสถานะขออนุมัติ พ็อป3 ยกเลิกการติดต่อแต่จะไม่เข้าไปในสถานะปรับปรุงข้อมูล

ถ้ายกเลิกด้วยเหตุผลอื่น ๆ ที่ไม่ใช่คำสั่ง QUIT พ็อป3 จะไม่เข้าไปสู่สถานะปรับปรุงข้อมูล และไม่ย้ายข่าวสารอย่างใดๆ ออกจาก maildrop

## 2.9 เอสเอ็มทีพี (SMTP : Simple Mail Transfer Protocol)

โปรโตคอลที่คู่กับ พ็อป3 ก็คือ เอสเอ็มทีพี เพราะเป็นโปรโตคอลที่ใช้ส่งอีเมล์จาก ยูสเซอร์เอเจนต์ ของผู้ส่งไปยัง เอ็มทีเอ ของผู้ส่ง และส่งต่อไปยัง เอ็มทีเอ เครื่องอื่น ๆ ที่เป็นจุดผ่านในการเชื่อมต่อไปยังเครื่องของผู้รับ โปรโตคอล เอสเอ็มทีพี จะทำงานร่วมกับโปรโตคอลทีซีพี โดยใช้พอร์ต 25 ซึ่งคำสั่งต่าง ๆ ของ เอสเอ็มทีพี จะเป็นลักษณะเดียวกับ พ็อป3 คือ เป็นแอสกี ลงท้ายด้วย Carriage Return และ Line Feed ส่วนข้อความที่ตอบกลับมาจะนำหน้าด้วยเลข 3 หลัก เป็นสัญลักษณ์แสดงสถานะการทำงานของคำสั่งที่ได้รับ

เมื่อเริ่มต้นการติดต่อ เอสเอ็มทีพี จะกำหนดให้ ยูสเซอร์เอเจนต์ ของผู้ส่งต้องส่งคำสั่ง HELLO พร้อมกับรายละเอียดด้านผู้ส่งออกไป จากนั้นจะส่งคำสั่ง MAIL เพื่อแจ้งให้เซิร์ฟเวอร์เตรียมรับอีเมล์ ในส่วนของเซิร์ฟเวอร์เมื่อพร้อมที่จะรับอีเมล์ก็จะตอบรับกลับมาด้วยคำสั่ง OK จากนั้นที่ด้านส่งก็จะเริ่มส่งโดยใช้คำสั่ง RCPT เพื่อกำหนดอีเมล์แต่ละฉบับที่ส่งไป ซึ่งการส่งข้อมูลของอีเมล์จะถูกระบุด้วยคำสั่ง DATA

การส่งอีเมล์ของโปรโตคอล เอสเอ็มทีพี ได้จัดเตรียมคำสั่งอื่น ๆ ไว้เพื่ออำนวยความสะดวกและคล่องตัวในการทำงาน ซึ่งประกอบด้วยคำสั่ง VRFY เพื่อให้ด้านที่ส่งตรวจสอบรายชื่อว่าผู้ใช้รายนี้ว่ามีสิทธิ์ใช้งานอีเมล์บ็อกซ์นั้น ๆ หรือไม่ คำสั่ง EXPN ใช้จัดการและตรวจสอบรายชื่อจากลิสต์รายชื่อ และคำสั่ง TURN ใช้สลับให้ไคลเอนต์ของผู้ส่งทำหน้าที่รับข้อมูลจากเซิร์ฟเวอร์แทน

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

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

**ตารางที่ 2.2** แสดงรายละเอียดในคำสั่งต่าง ๆ ของเอสเอ็มทีพีที่ใช้งานอยู่ [3]

| 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 |

### 2.9.1 กระบวนการขั้นตอนการทำงานของ เอสเอ็มทีพี (Simple Mail Transfer Protocol)

การติดต่อระหว่างเอ็มทีเอใช้รหัสแอสกีเอ็นวีทีเช่นเดียวกับในเทลเน็ตและเอฟทีพี ไคลเอ็นต์จะส่งคำสั่งไปยังเซิร์ฟเวอร์ ส่วนเซิร์ฟเวอร์จะตอบกลับเป็นรหัสตัวเลข โดยอาจมีสายอักขระขยายความ

การทำงานเริ่มต้นด้วยเอสเอ็มทีพีฝ่ายไคลเอ็นต์ซึ่งเป็นฝ่ายส่งขอสถาปนาการเชื่อมต่อทีซีพีกับเอสเอ็มทีพีฝ่ายเซิร์ฟเวอร์ แต่ก่อนการรับส่งข้อความจะมีกระบวนการแลกเปลี่ยนคำสั่งเพื่อแจ้งผู้รับและผู้ส่งก่อน ในขั้นถัดไปฝ่ายส่งจึงนำส่งข้อความได้ เมื่อสิ้นสุดข้อความจะมีรหัสอักขระส่งปิดท้ายเป็นสัญญาณให้ฝ่ายรับทราบ หลังจากนั้นฝ่ายส่งจึงขอปิดการเชื่อมต่อ

เมื่อไคลเอ็นต์ขอติดต่อไปยังไคลเอ็นต์ เซิร์ฟเวอร์ด้วยคำสั่ง HELO เซิร์ฟเวอร์ให้บริการเอสเอ็มทีพีจะตอบกลับด้วยรหัส 250 ซึ่งจะมีกระบวนการทำงานอยู่ 3 ขั้นตอน ดังนี้

**ขั้นตอนแรก**

**คำสั่งจดหมาย <reverse-path> บรรจุกล่องจดหมายต้นกำเนิด**

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

**ตัวอย่าง**

>>>MAIL From: <patty@nontri.nu.ac.th> SIZE = 56

250 <nguan@nontri.ku.ac.th>... Sender ok

**ไคลเอ็นแจ้งเซิร์ฟเวอร์ว่ามีเมล์จากผู้ใช้ patty@nontri.nu.ac.th และทางฝ่ายเซิร์ฟเวอร์ตอบรับตกลง**

**คำสั่งนี้จะบอก เอสเอ็มทีพี - เครื่องรับ ว่ารายการจดหมายใหม่จะเริ่มใหม่และจะคืนค่าเนื้อหาและบัพเฟอร์ รวมถึงข้อมูลของเมล์ของผู้รับอื่นๆ สถานะของมันทั้งหมดมันจะกลับเส้นทาง (reverse-path) ไปเป็นส่วนของการรายงานข้อผิดพลาด ถ้ายอมรับ เครื่องรับ- เอสเอ็มทีพี รายงานตกลง 250 OK ตอบกลับ**

**<reverse-path> คือ การกลับเส้นทางของต้นทางของโฮสต์ และกล่องจดหมายต้นกำเนิด โดยที่โฮสต์แรกใน <reverse-path> จะเป็นตัวส่งคำสั่งนี้**

**ขั้นตอนที่สอง**

**คำสั่ง RCPT**

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

**ตัวอย่าง**

RCPT To: <patty@can.org>

250 <patty@can.org>... Recipient ok

**ไคลเอ็นต์บอกถึงแอดเดรสของผู้รับเมล์ ถ้ามีผู้รับมากกว่าหนึ่งรายให้ใช้คำสั่ง RCPT สำหรับผู้รับแต่ละราย หากเซิร์ฟเวอร์ตรวจพบว่ามีผู้รับด้านปลายทางก็จะแจ้งตอบตกลงด้วยรหัส 250 หากไม่มีชื่อผู้รับจะแจ้งกลับด้วยรหัส 550 (user unknown)**

**คำสั่งนี้เป็นการส่งเส้นทางล่วงหน้า (forward-path) ระบุถึงผู้รับ ถ้ายอมรับ เครื่องรับ - เอสเอ็มทีพี จะรายงานตกลง 250 OK ตอบกลับและเก็บเส้นทางล่วงหน้าไว้ ถ้าผู้รับไม่ทราบที่มาของเครื่องรับ - เอสเอ็มทีพี จะรายงานความล้มเหลว 550 Failure ตอบกลับ โดยในขั้นตอนที่สองของกระบวนการทำงานนี้สามารถกระทำซ้ำๆได้หลายครั้ง**

**<forward-path> คือ เส้นทางที่ต้นทางของโฮสต์ไปกล่องจดหมายปลายทาง โดยที่โฮสต์แรกใน<forward-path> ที่เป็นตัวส่งคำสั่งนี้**

**ขั้นตอนที่สาม**

**คำสั่ง DATA**

**DATA < CRLF >**

**ตัวอย่าง**

>>> DATA

354 Enter mail, end with "." on a line by itself

**คำสั่ง DATA กำหนดว่าจะนำส่งข้อมูล ทางเซิร์ฟเวอร์จะตอบกลับด้วยรหัส 354 ว่าพร้อมรับข้อความ**

**ถ้ายอมรับเครื่องรับ-** เอสเอ็มทีพี รายงาน 354 Intermediate ตอบกลับและ**พิจารณา บรรทัดต่อมาทั้งหมดที่เป็นข้อความข่าวสาร เมื่อจบข้อความแล้ว จะรับและเก็บ จากนั้น เอสเอ็มทีพี - เครื่องรับ จะส่ง 250 OK ตอบกลับ**

>>

250 XAA07990 Message accepted for delivery

patty@can.org.... Sent (XAA07990 Message accepted for delivery)

Closing connection to mailgw.can.org.

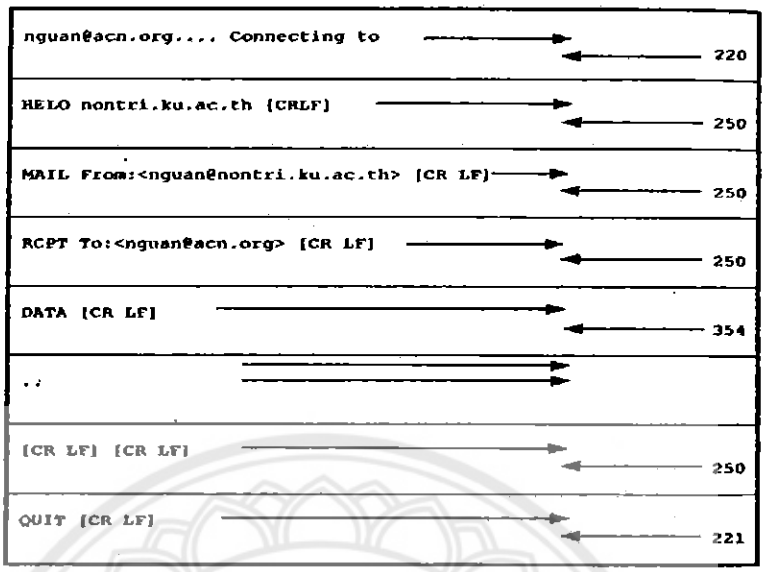
**ไคลเอ็นต์นำส่งข้อมูล (โดยไม่ได้แสดงข้อมูล) และเซิร์ฟเวอร์ตอบรับการส่ง**

>>> QUIT

221 mailgw.can.org closing connection

**ไคลเอ็นต์จบการทำงานโดยส่งคำสั่ง QUIT และเซิร์ฟเวอร์ตอบกลับว่าได้ปิดการเชื่อมต่อแล้ว**

**หากพิจารณาถึงรายละเอียดการส่งคำสั่งและข้อมูลจากไคลเอ็นต์ไปยังเซอร์เวอร์ คำสั่งทุกคำสั่งจะปิดท้ายด้วยรหัส CR LF ข้อความที่ส่งหลังจากคำสั่ง DATA จะปิดท้ายด้วย CR LF เช่นกัน ส่วนเครื่องหมายจุดที่พิมพ์ปิดท้ายเป็นสัญลักษณ์บอกให้ยูเอส่งรหัส CR LF และ CR LF ติดกันเพื่อแจ้งว่าข้อความสิ้นสุดแล้ว**




รูปที่ 2.5 ตัวอย่างการส่งและโต้ตอบเมล์

**2.9.2 การส่งไป (FORWARDING)**

มีบางกรณีที่ข้อมูลปลายทางใน <forward-path> ไม่ถูกต้องแต่เครื่องรับ- เอสเอ็มทีพี รู้ปลายทางที่ถูกต้อง ในกรณีนั้น จะมีการตอบกลับมาว่าให้ใส่ผู้ส่งติดต่อปลายทางที่ถูกต้อง

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

ตัวที่ตอบกลับมานี้จะบอกว่าเครื่องรับ - เอสเอ็มทีพี รู้ว่ากล่องจดหมายของผู้ใช้อยู่บนโฮสต์อื่นๆ และบอกเส้นทางต่อไปที่ถูกต้องซึ่งจะใช้ในอนาคต

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

สิ่งนี้ตอบกลับชี้บอกว่าเครื่องรับ - เอสเอ็มทีพี รู้กล่องจดหมายของผู้ใช้อยู่บนโฮสต์ อื่นๆ และจะชี้บอกเส้นทางต่อไปที่จะใช้ เครื่องรับจะปฏิเสธการรับจดหมายนี้ และผู้ส่งต้องส่งตามข้อมูลภายใต้เงื่อนไขหรือบอกข้อผิดพลาดให้ผู้ส่งทราบ

**2.9.3 การพิสูจน์และขยายรายชื่อจดหมาย (VERIFYING AND EXPANDING)**

เอสเอ็มทีพี มีความสามารถเพิ่มเติม คือ คำสั่งพิสูจน์ว่าเป็นผู้ใช้หรือขยายรายชื่อที่จดหมาย ซึ่งจะใช้คำสั่ง VRFY และ EXPN ซึ่งจะเป็นข้อความสตริงตัวอักษร สำหรับคำสั่ง VRFY สตริงคือชื่อผู้ใช้และการตอบสนองรวมถึงชื่อเต็มของผู้ใช้และรวมถึงกล่องจดหมายของผู้ใช้ สำหรับคำสั่ง EXPN สตริงระบุรายชื่อที่จดหมาย และการตอบสนองหลายเส้นทาง (multiline) จะรวมถึงชื่อเต็มของผู้ใช้และบนจดหมายในกล่องจดหมายรายชื่อ

"User name" คือ เทอมเล็ก ๆ ที่ถูกใช้โดยมีจุดประสงค์ว่า ถ้าโฮสต์เพิ่ม VRFY หรือ EXPN คำสั่งต่อมาอย่างน้อยที่สุดกล่องจดหมายภายในจะถูกจำ เนื่องจากว่า "user names" ถ้า host เลือกที่จะจำสตริงอื่น ๆ เป็นชื่อของผู้ใช้ ก็จะถูกอนุญาติ

ในโฮสต์ จะมีความแตกต่างระหว่างที่จดหมายแสดงรายการและนามแฝงสำหรับกล่องจดหมายเดี่ยวคือ ส่วนย่อย ๆ เพราะว่าโครงสร้างข้อมูลธรรมดาจะรับเอาชนิดทั้งสองของรายการที่จดไว้ และจะมีจดหมายที่แสดงรายการของหนึ่งกล่องจดหมาย ถ้าความต้องการตรวจสอบที่จดหมายแสดงรายการตอบสนองที่เป็นบวก สามารถที่จะให้ได้ ถ้าในการรับของข่าวสารตำแหน่งที่อยู่มันจะถูกส่งให้ให้ทุกๆ คนบนรายชื่อ ไม่เช่นนั้นข้อผิดพลาดควรจะถูกรายงาน (e.g .," 550 That is a mailing list,not a user") ถ้าความต้องการถูกทำเพื่อที่จะขยายชื่อของผู้ใช้ และตอบสนองจะเป็นบวกแล้วจะถูกสร้างขึ้นโดยการคืนที่บรรจุรายชื่อ 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"

**2.9.4 ส่งข่าวสารให้กล่องจดหมายของผู้ใช้ (SENDING AND MAILING)**

จุดประสงค์หลักของ เอสเอ็มทีพี คือจะส่งข่าวสารให้กล่องจดหมายของผู้ใช้ โดยโฮสต์ จำนวนหนึ่งจะส่งข่าวสารไปให้สถานีปลายทางของผู้ใช้ (ภายใต้เงื่อนไขผู้ใช้ทำงานบนโฮสต์) การส่งให้กล่องจดหมายของผู้ใช้ถูกเรียกว่า "mailing" การส่งให้สถานีปลายทางของผู้ใช้ถูกเรียก "sending" โดยโฮสต์ส่วนมากจะยอมให้ผู้ใช้ยอมรับหรือปฏิเสธข่าวสารนั้นได้

ดังต่อไปนี้สามคำสั่งถูกจำกัดความเพื่อสนับสนุนทางเลือก (option) ที่ส่ง ซึ่งถูกใช้ในรายการจดหมายแทนที่คำสั่งจดหมายและแจ้งบอกเครื่องรับ- เอสเอ็มทีพี 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) บนโฮสต์ ถ้าผู้ใช้ไม่มีตัวตนใน(หรือไม่ยอมรับข่าวสารที่ปลายทาง) ข้อมูลจดหมายจะถูกใส่ไปไว้ในกล่องจดหมายของผู้ใช้ รายการจดหมายสมบูรณ์จะถ้าข่าวสารถูกส่งให้หนึ่งในสถานีปลายทางหรือกล่องจดหมาย

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

การส่งคำสั่งจดหมาย ต้องการว่า ข้อมูลจดหมายจะถูกส่งให้ถึงสถานีปลายทางของผู้ใช้ถ้าผู้ใช้มีการใช้งานหรือ(ยอมรับข่าวสารสถานีปลายทาง) บนโฮสต์ ในกรณีใดๆ ข้อมูลจดหมายจะใส่เข้าไปในกล่องจดหมายของผู้ใช้ รายการจดหมายจะสมบูรณ์ถ้าข่าวสารถูกส่งให้กล่องจดหมาย

**2.9.5 การเปิดและปิด (OPENING AND CLOSING)**

ช่วงหนึ่งของช่องสัญญาณจะมีการการส่งผ่านเพื่อแลกเปลี่ยนข้อมูล โดยโฮสต์กำลังติดต่อกับโฮสต์ที่ถูกต้อง

จะมีสองคำสั่งถูกใช้ในการเปิดช่องสัญญาณการส่งผ่านและการสิ้นสุด :

HELO < SP ><domain>< CRLF >

QUIT < CRLF >

ในคำสั่ง HELO ที่ส่งโฮสต์จะระบุตัวเองและคำสั่งอาจจะอธิบายด้วยคำพูดว่า "Hello, I am <domain>".

**2.9.6 เมล์รีเลย์**

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

- ผู้ใช้สถานีงานขนาดเล็กหรือพีซีที่ไม่มีเอ็มทีเอมักไม่ได้เปิดเครื่องใช้งานอยู่ตลอดเวลา เมื่อมีเมล์เข้ามาจำเป็นต้องอาศัยเมล์รีเลย์เป็นตัวเก็บพักเมล์ไว้จนกว่าจะเปิดใช้พีซีเพื่อขอถ่ายเมล์มาจากเมล์รีเลย์
- เครือข่ายในหลายองค์กรใช้เมล์รีเลย์ทำหน้าที่ติดต่อกับเครือข่ายภายนอกเมล์รีเลย์อาจเป็นจุดเดียวที่อนุญาตให้รับส่งเมล์โดยตรงกับภายนอกได้ โดยมีระบบไฟร์วอล์ห้ามเครื่องอื่นภายในเครือข่ายรับส่งเมล์โดยตรงเพื่อสร้างระบบเมล์ศูนย์กลางและไม่ให้ชื่อเครื่องอื่นในเครือข่ายแพร่ออกไปภายนอก
- การติดตั้งเอ็มทีเออย่างเช่น ส่งเมล์ในยูนิกซ์มีความซับซ้อน ผู้ดูแลระบบบางแห่งจะไม่ติดตั้งเอ็มทีเอกระจายไปทั่ว แต่ให้ใช้บริการผ่านเมล์รีเลย์แทน

กลับเส้นทางจะเป็นเส้นทางต้นกำเนิดของรูปแบบ"@ONE,@ TWO:JOE@THREE ",ซึ่ง ONE, TWO, และ THREE คือโฮสต์รูปแบบนี้ที่ถูกใช้เพื่อเน้นความแตกต่างระหว่างตำแหน่งที่อยู่และเส้นทาง กล่องจดหมายจะเป็นตำแหน่งที่อยู่สมบูรณ์ และเส้นทางคือคำแนะนำเกี่ยวกับวิธีได้ที่นั่น สองความคิดจะต้องไม่ทำให้สับสน

ส่วนประกอบของเส้นทางต่อไปจะถูกเคลื่อนย้ายสู่การกลับเส้นทาง ซึ่งข่าวสารที่ถูกถ่ายทอดจากหนึ่งเครื่องแม่ข่าย- เอสเอ็มทีพี ถึงที่อื่นๆ การกลับเส้นทาง คือการกลับหลังของเส้นทางต้นกำเนิด (เช่น เส้นทางต้นกำเนิดจากตำแหน่งปัจจุบันของข่าวสารถึง originator ของข่าวสาร) เมื่อเครื่องแม่ข่าย-เอสเอ็มทีพี ลบผู้ค้นหาของมันจากการกลับเส้นทาง และใส่มันเข้าไปในการกลับเส้นทาง มันต้องใช้ชื่อที่มันรู้โดยสิ่งแวดล้อมที่มันกำลังส่งเข้าไป ไม่ใช่สิ่งแวดล้อมที่จดหมายมา ในกรณีของเครื่องแม่ข่าย-เอสเอ็มทีพี ถูกรู้โดยชื่อที่แตกต่างในสิ่งแวดล้อมแตกต่างกัน

เมื่อข่าวสารมาถึง เอสเอ็มทีพี ส่วนประกอบแรกของการกลับเส้นทาง คือไม่มีการค้นหาของเอสเอ็มทีพี ส่วนประกอบไม่ถูกลบจากการกลับเส้นทาง และถูกใช้เพื่อตัดสินใจ เอสเอ็มทีพี ถัดไปที่จะส่งข่าวสารถึง ในกรณีใดๆ เอสเอ็มทีพี เพิ่มผู้ค้นหาด้วยตัวเองของมันให้การกลับเส้นทาง

การใช้ที่กำเนิดเครื่องรับ - เอสเอ็มทีพี รับจดหมายถูกถ่ายทอดให้เครื่องแม่ข่ายอื่น ๆ เอสเอ็มทีพี เครื่องรับ - เอสเอ็มทีพี จะยอมรับหรือปฏิเสธงานของการถ่ายทอดจดหมาย เช่นเดียวกันมันยอมรับหรือปฏิเสธ จดหมายสำหรับผู้ใช้ภายใน เครื่องรับ - เอสเอ็มทีพีจะเปลี่ยนรูปแบบข้อความคำสั่งโดยเคลื่อนที่

ผู้ค้นหาด้วยตัวเองของมันจากการกลับเส้นทาง ถึงการเริ่มของการกลับเส้นทาง เครื่องรับ- เอสเอ็มทีพี ต่อมาจะกลายเป็นผู้ส่ง-เอสเอ็มทีพี ช่องสัญญาณจะถูกตั้งขึ้นโดยทำการส่งผ่านให้ เอสเอ็มทีพี ถัดไปในการกลับเส้นทาง และส่งจดหมาย

โฮสต์ตัวแรกในการกลับเส้นทาง ควรจะที่ส่งคำสั่งโฮสต์ ให้ เอสเอ็มทีพี และโฮสต์แรกในการ กลับเส้นทาง ควรจะที่รับโฮสต์คำสั่ง เอสเอ็มทีพี

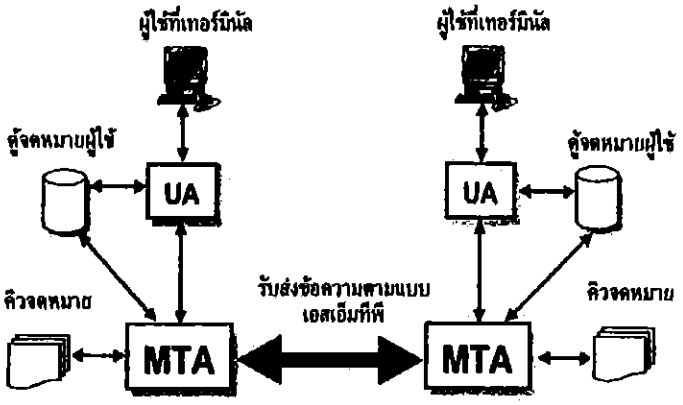
การเลือกเส้นทางต่อไปและการกลับเส้นทาง จะปรากฏในคำสั่ง เอสเอ็มทีพี และตอบกลับ แต่ไม่จำเป็นในข่าวสาร กล่าวคือ สิ่งนี้ไม่ต้องการเส้นทางเหล่านี้และโดยเฉพาะข้อความนี้ที่จะปรากฏใน "To:", "From:", " CC :" และอื่นๆ ในส่วนพื้นที่ของหัวข้อข่าวสาร

ถ้าเครื่องแม่ข่าย -เอสเอ็มทีพี ยอมรับงานของการถ่ายทอดจดหมายและต่อมาพบว่าการเลือกเส้นทางต่อไปไม่ถูกต้อง หรือจดหมายไม่สามารถถูกส่งให้ได้สำหรับเหตุผลอะไรก็ตาม ต่อมามันต้องสร้างจดหมายไม่ส่งให้ ข่าวสารการประกาศและส่งมันให้ originator ของจดหมายไม่ส่งให้ (ตามที่ที่ชี้บอกโดยการกลับเส้นทาง)

ข่าวสารนี้จำเป็นต้องสร้างจากเครื่องแม่ข่าย - เอสเอ็มทีพี ด้วยเหตุนี้โฮสต์เครื่องแม่ข่าย- เอสเอ็มทีพี ควรจะไม่ส่งข่าวสารเกี่ยวกับปัญหา ทางหนึ่งที่จะป้องกันการวนซ้ำในที่รายงานข้อผิดพลาด คือเจาะจงให้การกลับเส้นทางไม่มีผล (null reverse-path) ในคำสั่งจดหมายของข่าวสาร เมื่อข่าวสารถูกถ่ายทอด มันจะยินยอมที่จะออกจากการกลับเส้นทางไม่มีผล

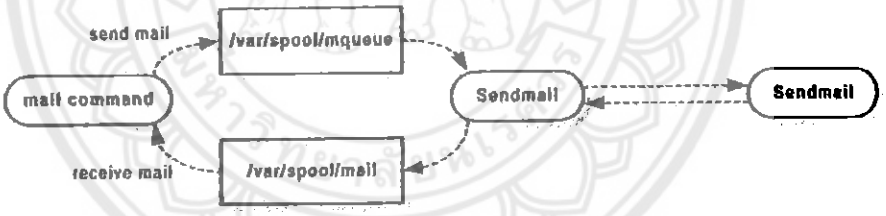
### 2.9.7 ยูนิกซ์เมล์

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




**รูปที่ 2.6** แสดงโครงสร้างของยูนิกซ์เมล์ [2]

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




**รูปที่ 2.7** แสดงการรับส่งเมล์ในระบบยูนิกซ์ [4]

**2.9.8 โดเมน**

โดเมน เมื่อไม่นานความคิดนี้ได้รับการแนะนำในระบบจดหมายอินเตอร์เนต ARPA การใช้ของโดเมนเปลี่ยนที่ว่างของตำแหน่งที่อยู่จากที่ว่างทั่วโลกบนโฮสต์สตริงตัวอักษรง่ายๆ ที่ตั้งชื่อถึงต้นไม้ที่ราก โดยมีโครงสร้างเกี่ยวกับลักษณะของ hierar ของตำแหน่งที่อยู่ทั่วโลก ซึ่งชื่อโฮสต์จะถูกย้ายโดยโดเมนและชื่อของโฮสต์ นั่นก็คือส่วนประกอบโดเมนจะถูกแยกตามลำดับของสตริง เมื่อถึงเวลาและทำความเข้าใจแล้ว ส่วนประกอบโดเมนจะถูกสั่งแบบเฉพาะเจาะจงและแบบทั่วไป

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

ชื่อโดเมนที่ถูกใช้ใน เอสเอ็มทีพี จะเป็นชื่อทางการเท่านั้น การใช้ชื่อเล่นหรือนามแฝงจะไม่ยอมให้ใช้

## 2.10 การติดต่อสื่อสารระหว่างโปรเซส (IPC : Inter process communication)

ในการติดต่อระหว่างโปรเซส 2 โปรเซสนั้น มีวิธีการติดต่อกันได้หลายวิธีดังต่อไปนี้

**ไปป์ (Pipe)** ถูกใช้งานเป็นตัวกลางในการส่งผ่านข้อมูลระหว่างโปรเซส ซึ่งเป็นการส่งข้อมูลแบบทางเดียว (one way flow) ทำงานผ่านเคอร์เนล เพื่อนำเอาท์พุทของโปรเซสเป็นอินพุทของอีกโปรเซสหนึ่ง แต่ไปป์ก็มีข้อจำกัดคือ สามารถใช้งานได้ระหว่างโปรเซสลูกที่มีโปรเซสต้นฉบับเดียวกันเท่านั้น แต่โปรเซสที่ไม่มีความสัมพันธ์ดังกล่าว ไม่สามารถจะใช้งานได้

**ไฟล์ไปป์ : ไฟโฟว์ FIFOs (name pipe)** เป็นไปป์ (ไฟล์ชนิดหนึ่ง) ที่ได้รับการกำหนดให้มีชื่อเฉพาะ (เป็นออบเจ็ค IPC หนึ่ง) เพื่อให้โปรเซสที่ไม่ความสัมพันธ์กัน สามารถอ้างอิงถึงไฟล์ไปป์ เป็นอ็อบเจ็คในการส่งผ่านข้อมูลร่วมกันได้อย่างสะดวกมากยิ่งขึ้น ไฟโฟว์จะไม่เหมือนกับไปป์ที่สร้างขึ้นมาโดยใช้ฟังก์ชั่นระบบไปป์ เนื่องจากไฟโฟว์เป็นชื่อของไฟล์ (ไปป์) มากกว่าจะเป็นชนิดของไฟล์ (file descriptor) ซึ่งก่อนที่จะอ่านหรือเขียนได้ จะต้องมีการเปิดไฟล์ (open) เสียก่อน หลังจากที่ทำงานเสร็จก็ต้องมีการปิดไฟล์ (close) เช่นกัน นอกจากนี้อาร์กิวเมนต์ที่ส่งไปให้ฟังก์ชั่น จะเป็นชื่อไฟล์ไฟโฟว์มากกว่าที่จะเป็นชื่อไฟล์ทั่ว ๆ ไป ข้อจำกัดของไฟล์ไปป์ ก็คือ จะใช้ไปป์เพื่อส่งผ่านข้อมูลเพียงหนึ่งทิศทางเช่นเดียวกับไปป์ทั่วไป

**เมจเซสคิว (Message Queue)** จะมีลักษณะคล้ายกับไปป์ ทำให้สามารถส่งบล็อกข้อมูลจากโปรเซสหนึ่งไปยังโปรเซสหนึ่งผ่านเคอร์เนล เช่นเดียวกัน แต่ละบล็อกข้อมูลจะมีหมายเลขของบล็อกนั้น แต่สิ่งที่แตกต่างกับไฟล์ไปป์ ก็คือ ไม่ต้องกำหนดจังหวะการทำงานร่วมกันของแต่ละโปรเซสที่ใช้งาน ซึ่งข้อดีคือ โปรเซสสามารถจะเขียนข้อมูลไปยังคิวแล้วจบการรันโปรเซส โดยที่โปรเซสอื่นสามารถจะมาอ่านข้อความได้ภายหลัง แต่ก็มีข้อเสียเช่นกันคือ ขนาดของบล็อกถูกจำกัดรวมทั้งจำนวนบล็อกทั้งหมดที่มีบนระบบก็ถูกจำกัดไว้เช่นกัน (ด้วยความจุรวมกันของแต่ละบล็อกข้อมูล) การใช้งานคิวเป็นที่เชื่อถือได้มากกว่า นอกจากนี้บางระบบการใช้งานคิว จะเร็วกว่าการใช้งานไปป์

**เซมาฟอร์** เมื่อต้องเขียนโปรแกรมในระบบ มัลติยูสเซอร์(multi-users) มัลติแทสกิ้ง (multi-tasking) สิ่งหนึ่งที่ต้องเจอบ่อย ๆ ก็คือ การที่โปรแกรมต้องการสิทธิ์ในการควบคุมรีซอร์สนั้นเพียงโปรแกรมเดียวในช่วงเวลาหนึ่ง (critical sections) ซึ่งเซมาฟอร์ก็เป็นแนวความคิดหนึ่งที่ใช้ในการสร้างระบบจัดการ การแอ็คเซสรีซอร์เพียง 1 โปรเซส ซึ่งจุดประสงค์ของเซมาฟอร์ คือ การกำหนดจังหวะการใช้งานรีซอร์สร่วมกันของ หลาย ๆ โปรเซส ค่านี้จะถูกเก็บและควบคุมโดยเคอร์เนล เพื่อที่จะเข้าใช้งานรีซอร์สนั้นเพียงโปรเซสเดียว

**การใช้งานหน่วยความจำร่วมกัน (Shared Memory)** ซึ่งจะเป็นการทำให้โปรเซส 2 โปรเซสหรือมากกว่าใช้ข้อมูลในหน่วยความจำร่วมกันได้โดยตรง ไม่ผ่านเคอร์เนล โดยโปรเซสเหล่านั้นจะต้องจัดการกับการทำงานร่วมกันด้วยตนเอง

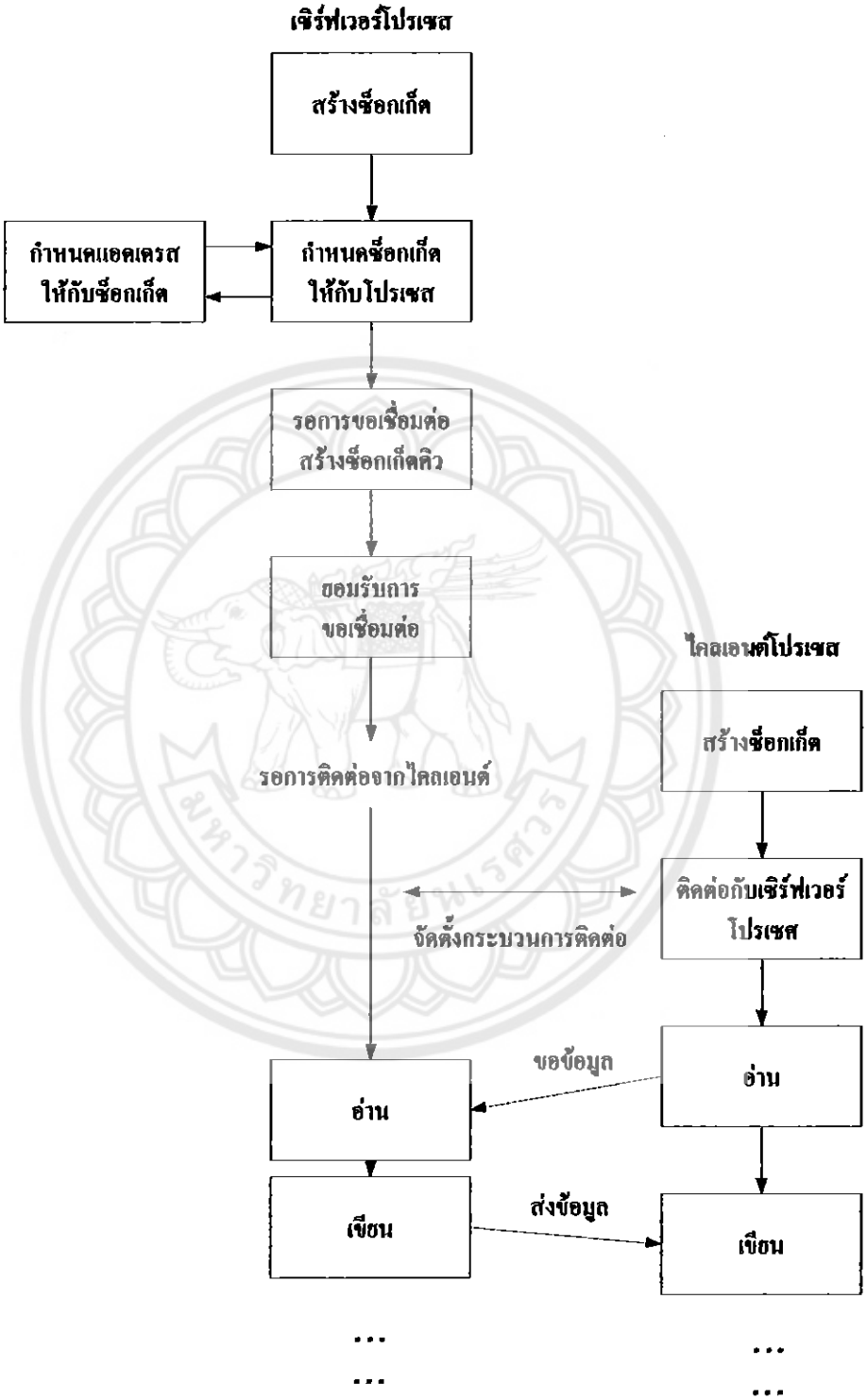
**ซ๊อกเก็ต (Socket)** เป็นวิธีการติดต่อของโปรเซสอีกวิธีหนึ่งแต่จะเพิ่มความพิเศษแตกต่างจากแบบอื่น ๆ คือ แบบอื่นๆ เป็นการเขียนโปรแกรมเพื่อใช้งานรีซอร์สของระบบร่วมกัน แต่โปรแกรมจะอยู่บนคอมพิวเตอร์เพียงเครื่องเดียว โปรเซสทุกโปรเซสรันอยู่บนเครื่องเดียว แต่การติดต่อแบบซ๊อกเก็จจะสามารถใช้งานเช่นเดียวกับไปป์ แต่สามารถทำให้โปรเซสที่อยู่คนละเครื่องของเน็ตเวิร์คสามารถติดต่อถึงกันได้ ซึ่งเป็นพื้นฐานที่สำคัญของ ไคลเอ็นต์/เซิร์ฟเวอร์ เน็ตเวิร์ค (Client/Server Network) ที่โปรเซสจะต้องรันผ่านเน็ตเวิร์ค

## 2.11 ซ๊อกเก็ต

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

### 2.11.1 โครงสร้างของการสร้างซ๊อกเกต

การทำงานแบบซ็อกเกตจะสามารถสรุปเป็นโครงสร้างได้ดังต่อไปนี้




**รูปที่ 2.8** โครงสร้างของการติดต่อระหว่างโปรเซสด้วยซ๊อกเก็ต

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
}
```

**2.11.2 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.println("Exception allocating ServerSocket: "+e);
        Throw e;
        }
```

**2.11.3 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
}
```

**2.11.4 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());
```

**2.11.5 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);
}
```

**2.11.6 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

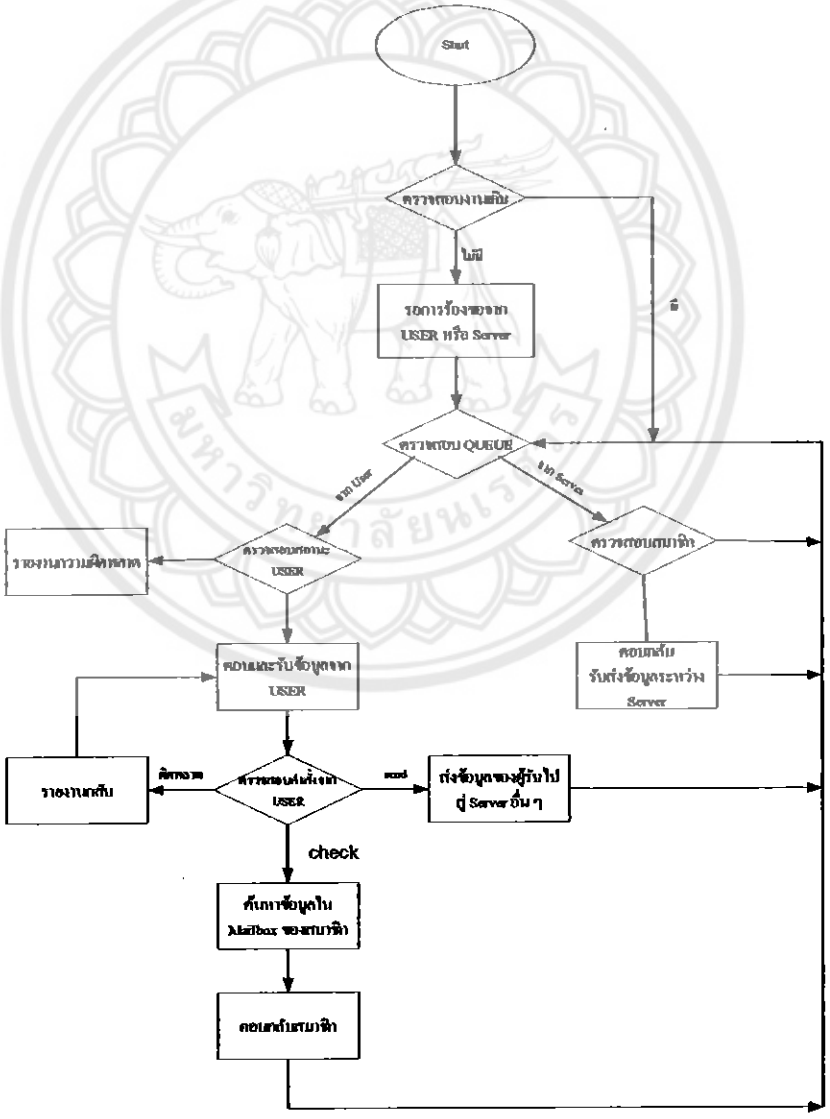
```
try {
        con.close();
}
catch (Exception e){
        System.err.println ("Error on close: " + e );
}
```

# บทที่ 3
# ออกแบบและพัฒนาโปรแกรม

เมื่อได้ทำการศึกษาหลักการและทฤษฎีที่เกี่ยวข้องแล้ว ผู้จัดทำก็ได้ทำการออกแบบและพัฒนาโปรแกรม ได้ดังนี้

## 3.1 ขั้นตอนการออกแบบโปรแกรม และโครงสร้างของโปรแกรม

จากการศึกษาหลักการและทฤษฎีที่เกี่ยวข้องและผู้จัดทำได้ออกแบบโปรแกรมโดยแสดงเป็นโครงสร้างของโปรแกรมได้ดังรูป




**รูปที่ 3.1** แผนผังลำดับงาน (Flow Chart) ของโปรแกรม

**3.1.1 ลำดับการทำงานของโปรแกรม**

หลังจากได้ศึกษาและรวบรวมข้อมูล กำหนดขอบเขตของการพัฒนาโปรแกรมเสร็จสิ้นแล้ว ก็นำความรู้ที่ได้มาออกแบบโปรแกรมที่จะพัฒนาขึ้น โดยการทำงานและโครงสร้างของโปรแกรมอธิบายได้ดังนี้

1. เริ่มต้นโปรแกรม
2. ตรวจสอบงานเดิมว่ายังมีเหลือค้างหรือไม่
   - ถ้ายังมีงานเดิมค้างอยู่ให้ไปตรวจสอบลำดับของ Queue
   - ถ้าไม่มีงานเดิมค้างให้ไปรอการร้องขอยูสเซอร์หรือเซิร์ฟเวอร์ เมื่อมีการร้องขอมาก็จะไปตรวจสอบคิวว่าใครมาก่อนได้รับการบริการก่อน
3. เมื่อมีการตรวจสอบคิว ก็จะทำการตอบสนองการร้องขอแรกสุดที่อยู่ในคิว ก่อนว่าใครเข้าใช้บริการก่อน ทั้งจากเซิร์ฟเวอร์และยูสเซอร์
4. ถ้ายูสเซอร์อยู่ในคิวก่อน จะตรวจสอบสถานะของยูสเซอร์ โดยตรวจสอบจากพาสเวิร์ด ที่ล็อคอิน (Login) เข้ามาถูกต้องหรือไม่ ถ้าพาสเวิร์ด ไม่ถูกต้องจะรายงานความผิดพลาดออกมา
5. เมื่อรายงานความผิดพลาดแล้ว จะกลับเข้าไปตรวจสอบ ผู้ที่อยู่ในคิวลำดับถัดไป
6. ถ้าพาสเวิร์ดถูกต้องจะตอบรับและรับคำสั่งจากยูสเซอร์
7. เมื่อรับคำสั่งจากยูสเซอร์แล้ว ก็จะตรวจสอบคำสั่งว่าถูกต้องหรือไม่
8. ถ้าผิดพลาดก็จะรายงานต่อยูสเซอร์แล้วกลับไปรับคำสั่งต่อ
9. ถ้าคำสั่งถูกต้องยูสเซอร์ต้องการส่งอีเมล์ไปยังปลายทาง ก็จะทำการส่งข้อมูลต่าง ๆ ไปยังเซิร์ฟเวอร์ปลายทาง
10. ถ้ายูสเซอร์ต้องการเช็คเมล์ก็จะทำการค้นหาในเมล์บ็อกซ์
11. ส่งข้อมูลไปให้สมาชิก
12. ก็จะวนกลับไปให้บริการคิวต่อไป
13. ถ้าคิวต่อไป เป็นเซิร์ฟเวอร์ก็จะตรวจสอบสมาชิกเมล์บ็อกซ์ของตนว่ามีสมาชิกที่เซิร์ฟเวอร์นั้นต้องการหรือไม่
14. ถ้าไม่มี หรือ มีข้อผิดพลาด ก็จะรายงานข้อผิดพลาด แล้ววนกลับไปให้บริการคิวต่อไป
15. ถ้ามี ก็จะรับส่งข้อมูลกันระหว่างเซิร์ฟเวอร์

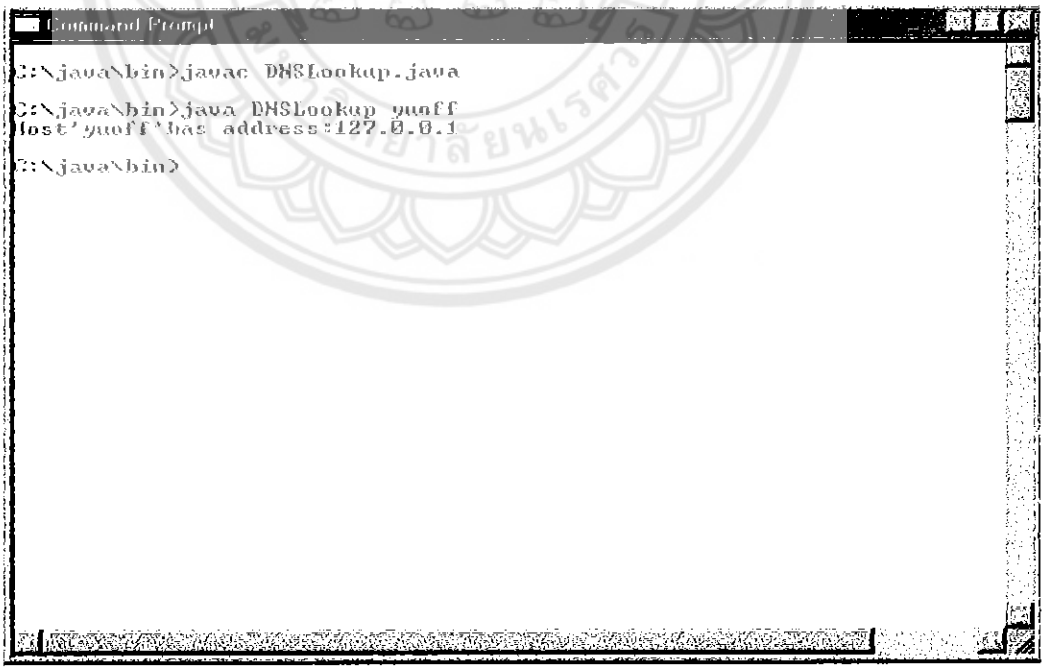
## 3.2 ขั้นตอนการทดลอง

- ทดลองเขียนโปรแกรมหาค่าไอพีแอดเดรสของเครื่องภายในเครือข่ายเดียวกัน
- เขียนโปรแกรมสำหรับเครื่องแม่ข่ายในการรับส่งจดหมายอิเล็คทรอนิกส์
- ทดลองสร้างโปรแกรมส่งจดหมายอิเล็กทรอนิกส์ไปยังเมล์เซิร์ฟเวอร์อื่น ๆ ที่มีอยู่แล้วตามมาตรฐานเอสเอ็มทีพี และ พ็อป3 เพื่อดูค่าที่ตอบกลับ แล้วนำมาทดลองกับเมล์เซิร์ฟเวอร์ของเราที่สร้างขึ้นเพื่อให้เป็นไปตามมาตรฐาน
- ทดลองนำโปรแกรมส่งจดหมายอิเล็กทรอนิกส์ที่เราได้สร้างขึ้น ส่งถึงโปรแกรมสำหรับเครื่องแม่ข่ายในการรับส่งจดหมายอิเล็คทรอนิกส์ที่เราเขียนขึ้นให้ได้ผลเหมือนกับเมล์เซิร์ฟเวอร์อื่น ๆ ที่มีอยู่แล้วตามมาตรฐานเอสเอ็มทีพี และพ็อป3
- นำสิ่งที่ทำการทดลองและนำโปรแกรมส่วนต่าง ๆ มารวมกัน แล้วนำมาปรับปรุงแก้ไข

## 3.3 ทดลองเขียนโปรแกรม

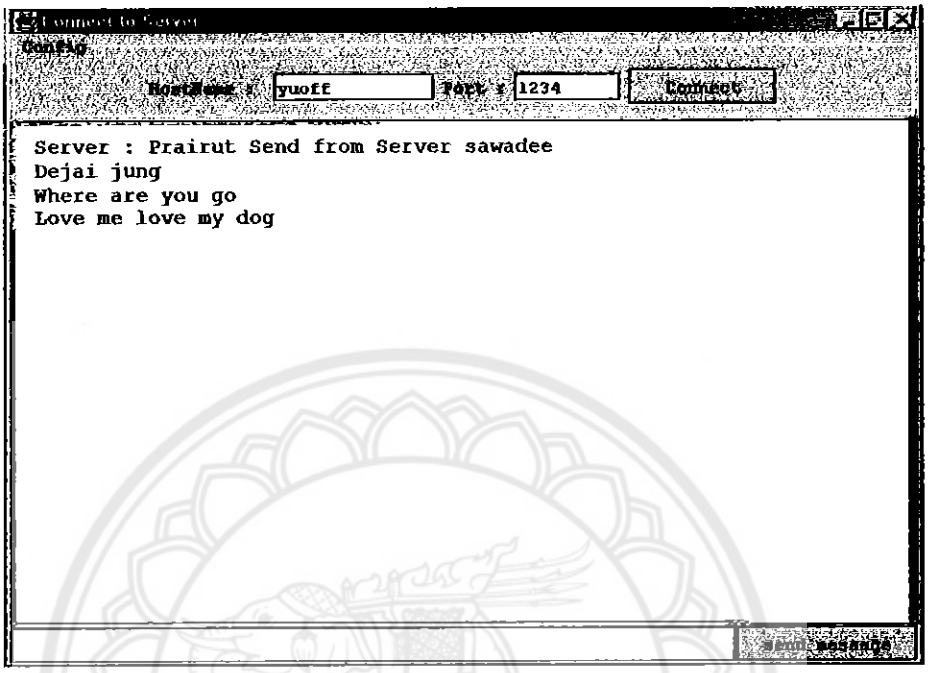
### 3.3.1 เขียนโปรแกรมหาค่าไอพีแอดเดรสของเครื่องภายในเครือข่ายเดียวกัน

ในการติดต่อระหว่างเครื่องนั้นจำเป็นจะต้องรู้ค่าไอพีแอดเดรสเพราะค่าไอพีแอดเดรสเป็นหมายเลขอ้างอิงประจำตัวของอุปกรณ์ต่าง ๆ ที่เชื่อมต่ออยู่ในเครือข่าย




**รูปที่ 3.2 ผลของโปรแกรมหาค่าไอพีแอดเดรสของเครื่องภายในเครือข่ายเดียวกัน**

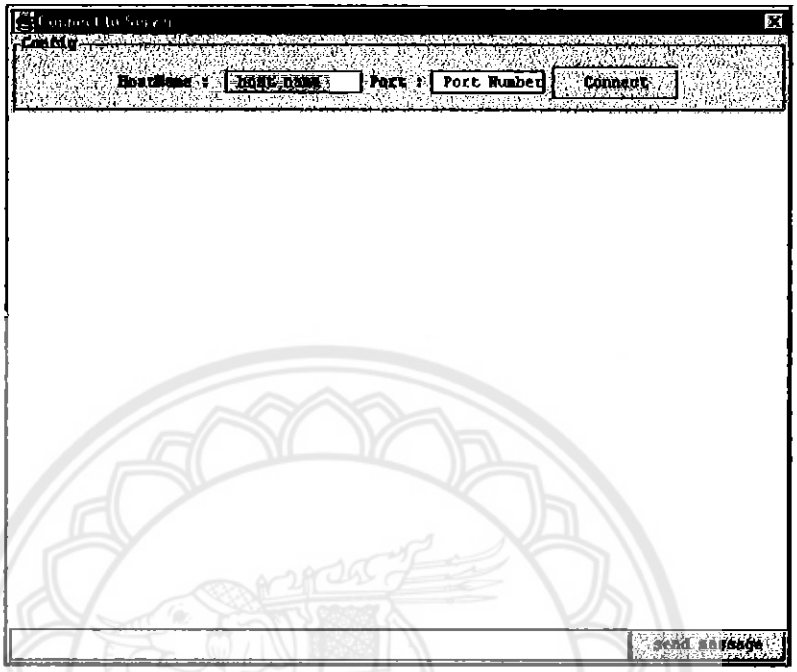
**3.3.2 เขียนโปรแกรมเมล์เซิร์ฟเวอร์ โดยส่งข้อความจากไคลเอนต์ที่เขียนขึ้นมาเอง โดยกำหนดต้องใส่ชื่อ Hostname เป็นชื่อเครื่อง และเลขพอร์ต 1234**



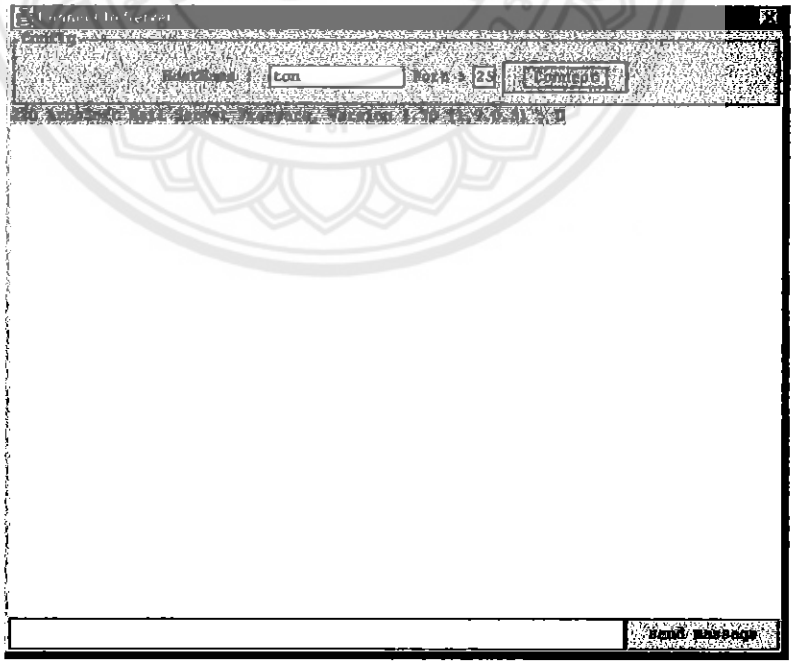

**รูปที่ 3.3** โปรแกรมเมล์ไคลเอนต์ส่งข้อความถึงเมล์เซิร์ฟเวอร์ที่เราเขียนขึ้น


**รูปที่ 3.4** โปรแกรมเมล์เซิร์ฟเวอร์ที่ได้รับข้อความจากโปรแกรมเมล์ไคลเอนต์

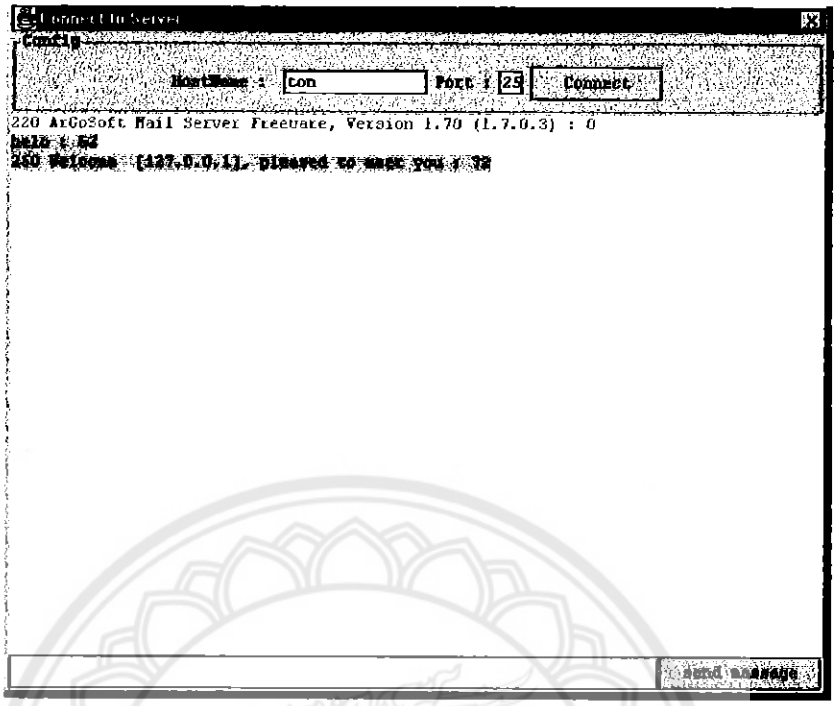
**3.3.3 ตรวจสอบการรับค่าและส่งค่าของเมล์เซิร์ฟเวอร์ที่มีอยู่แล้วตามมาตรฐานเอสเอ็มทีพีเพื่อนำมาใช้ในเมล์เซิร์ฟเวอร์ที่ได้เขียนขึ้น**



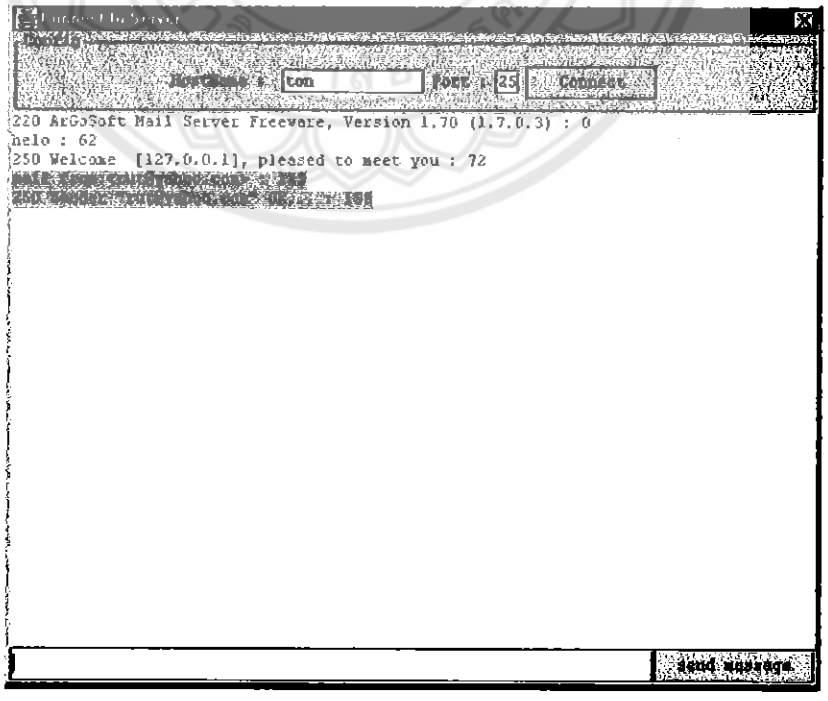

รูปที่ 3.5 เมื่อเปิดโปรแกรมเมล์ไคลเอนต์ทดสอบจะมีลักษณะเช่นนี้



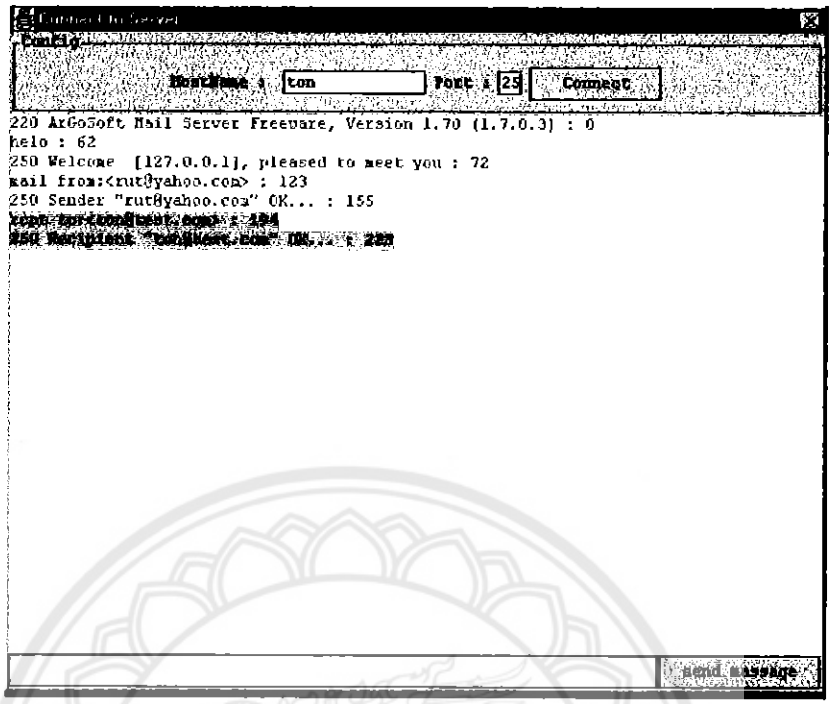

**รูปที่ 3.6 ทำการติดต่อไปที่เซิร์ฟเวอร์โดยใส่ชื่อของเซิร์ฟเวอร์ ตามด้วยหมายเลขพอร์ตสำหรับเอสเอ็มทีพี จะใช้พอร์ต 25 แล้วกด connect จะมีข้อความขึ้นมาในโปรแกรม**



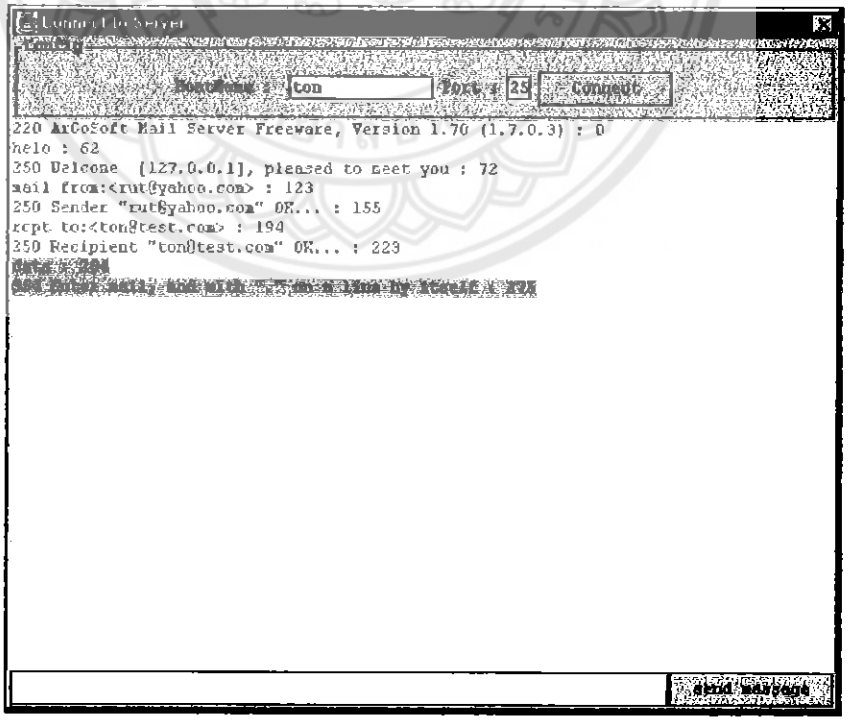

**รูปที่ 3.7** ใส่คำสั่ง hello เพื่อบอกเซิร์ฟเวอร์ ว่าเริ่มการติดต่อ และเซิร์ฟเวอร์จะตอบกลับมา



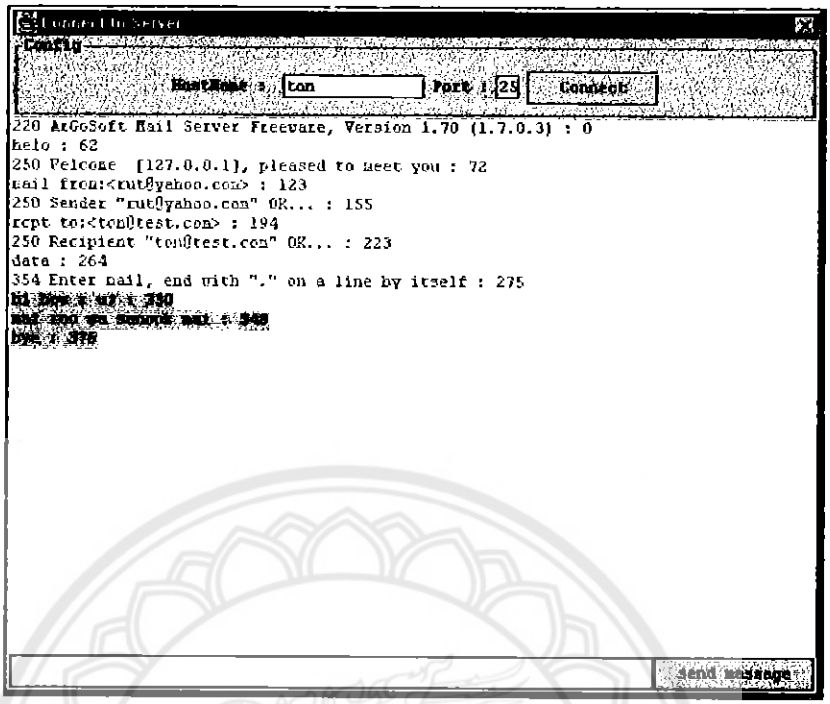

**รูปที่ 3.8** ใส่คำสั่ง mail from:<อีเมล์ของผู้รับ> เพื่อเป็นการแสดงผู้ส่งจดหมาย



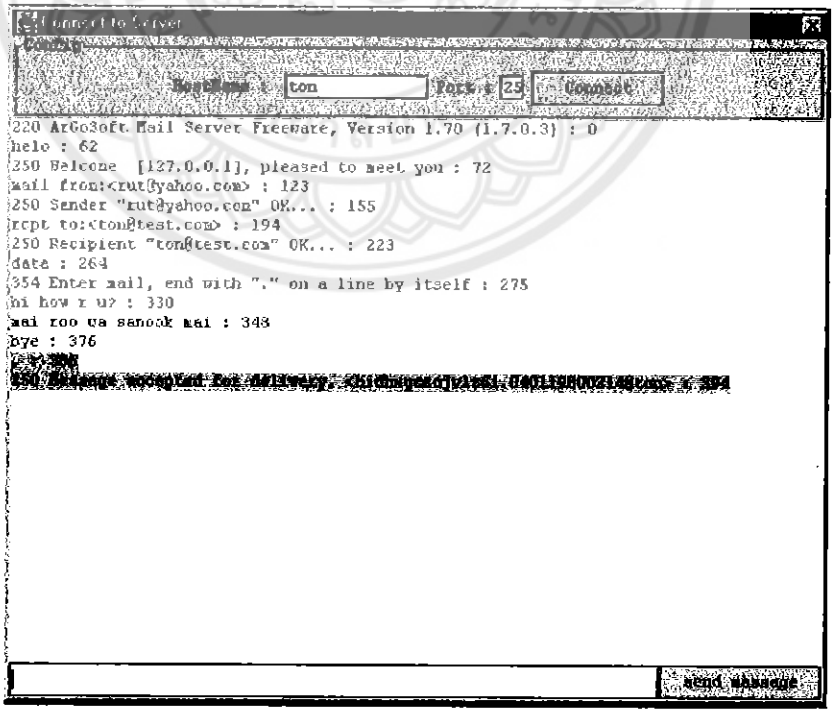

**รูปที่ 3.9** ใส่คำสั่ง rcpt to:<อีเมล์ของผู้รับ> โดยเป็นการบอกถึงผู้รับ**จดหมาย**



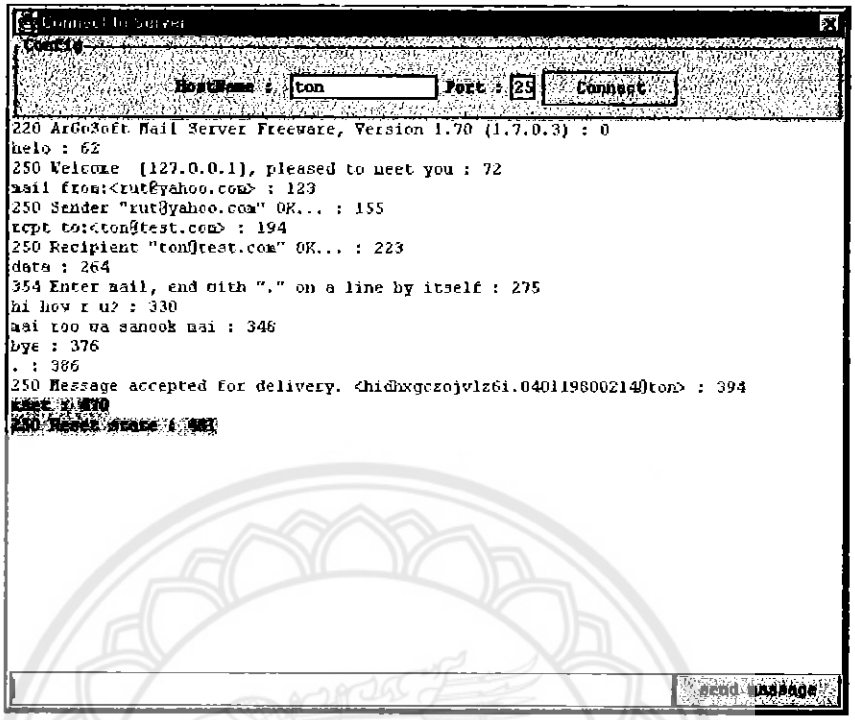

**รูปที่ 3.10** ใส่คำสั่ง data แล้วกดส่งข้อมูลเซิร์ฟเวอร์จะทำการตอบกลับมาว่า ให้เริ่มเขียน ข้อความ ของจดหมายได้



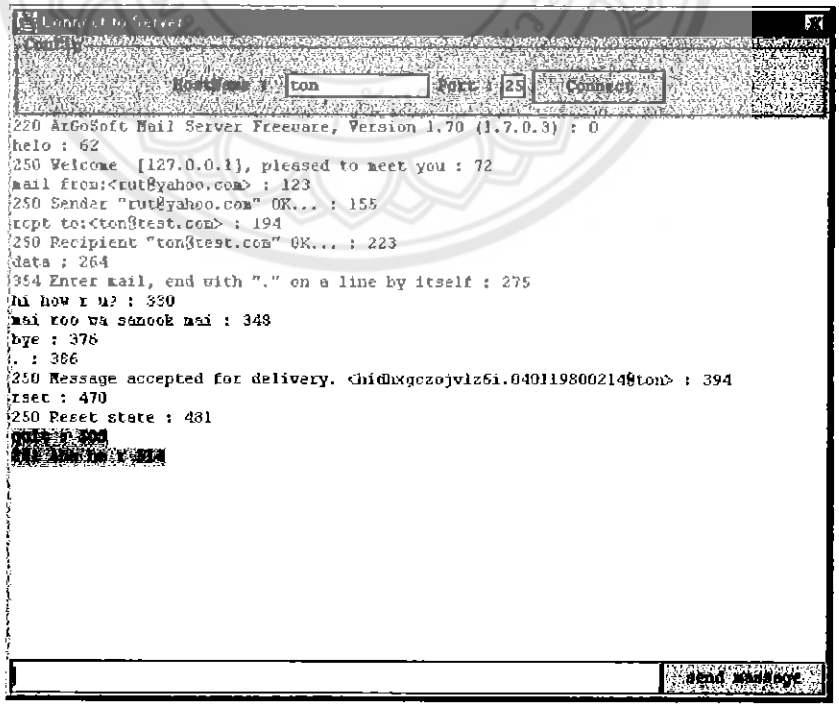

**รูปที่ 3.11 เริ่มเขียนข้อความที่ต้องการส่ง**




**รูปที่ 3.12 เมื่อเขียนข้อความจบแล้ว ให้สั่ง . (จุด) เพื่อบอกเซิร์ฟเวอร์ว่าต้องการสิ้นสุดข้อความ**

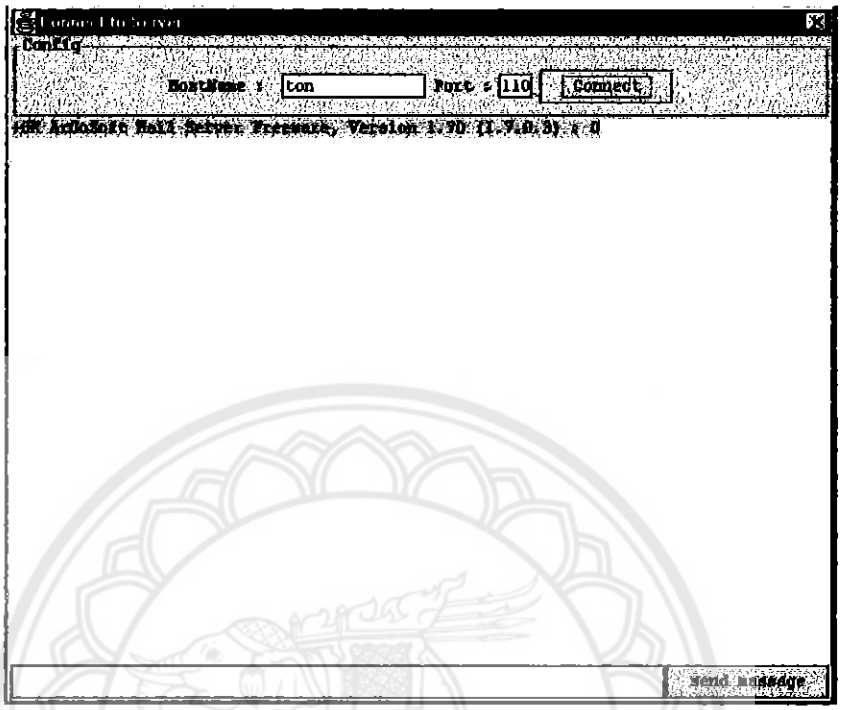



**รูปที่ 3.13** ถ้าต้องการเลิกการทำงานที่กล่าวมาทั้งหมด ให้ใส่คำสั่ง rset

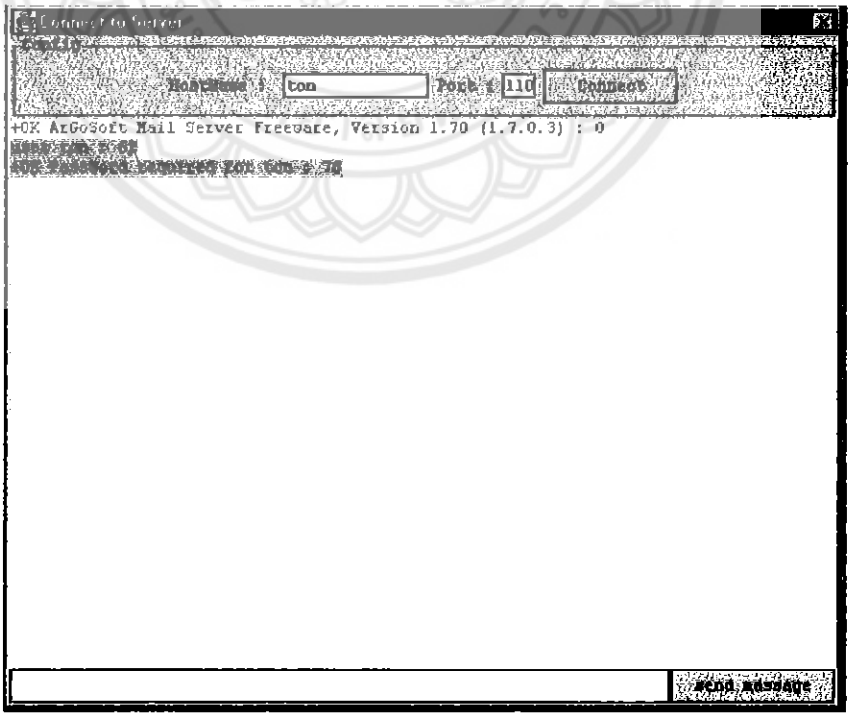



**รูปที่ 3.14** ใส่คำสั่ง quit เพื่อจบการติดต่อ กับเซิร์ฟเวอร์

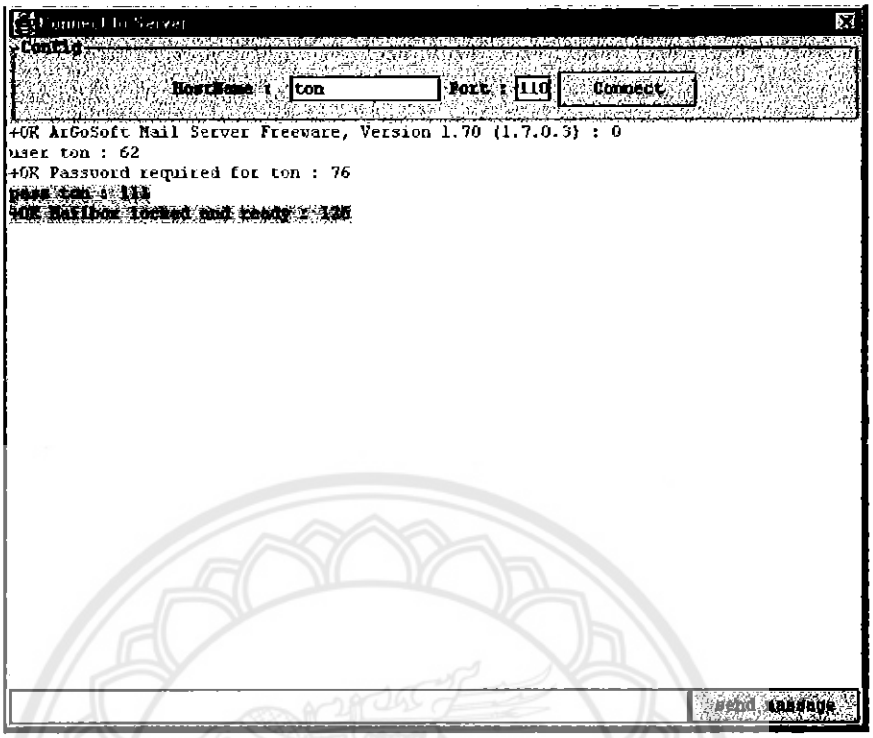
**3.3.4 ตรวจสอบการรับค่าและส่งค่าของ MailServer ที่มีอยู่แล้วตามมาตรฐานPOP3 เพื่อนำมาใช้ใน MailServer ที่ได้เขียนขึ้น**



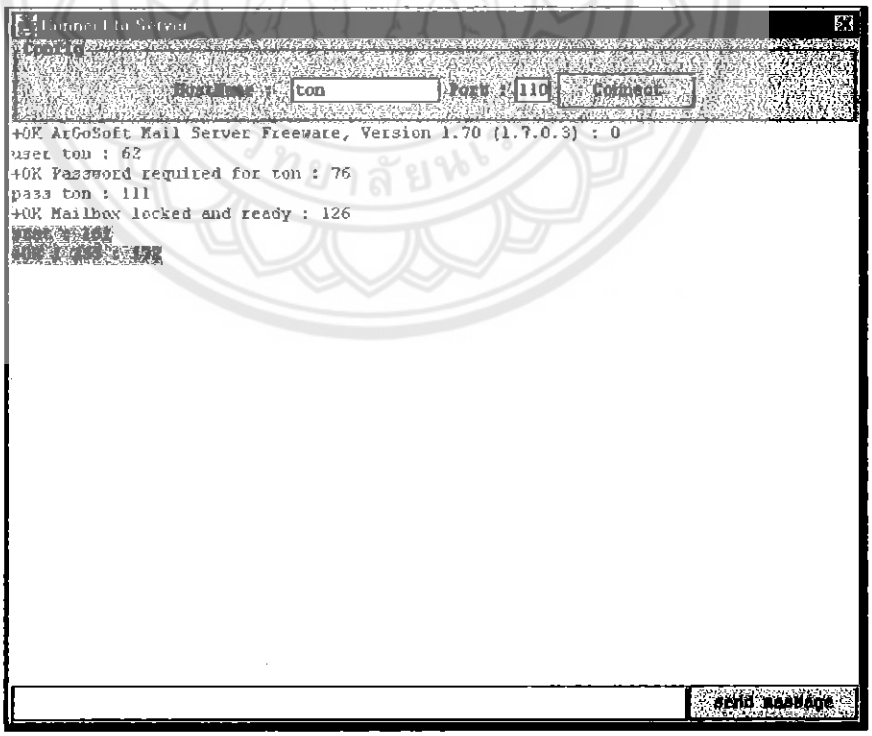

**รูปที่ 3.15** ทำการติดต่อไปที่เซิร์ฟเวอร์โดยใส่ชื่อของเซิร์ฟเวอร์ตามด้วยหมายเลขพอร์ตสำหรับพ็อป3 จะใช้ port 110 แล้วกด connect จะมีข้อความขึ้นมาในโปรแกรม



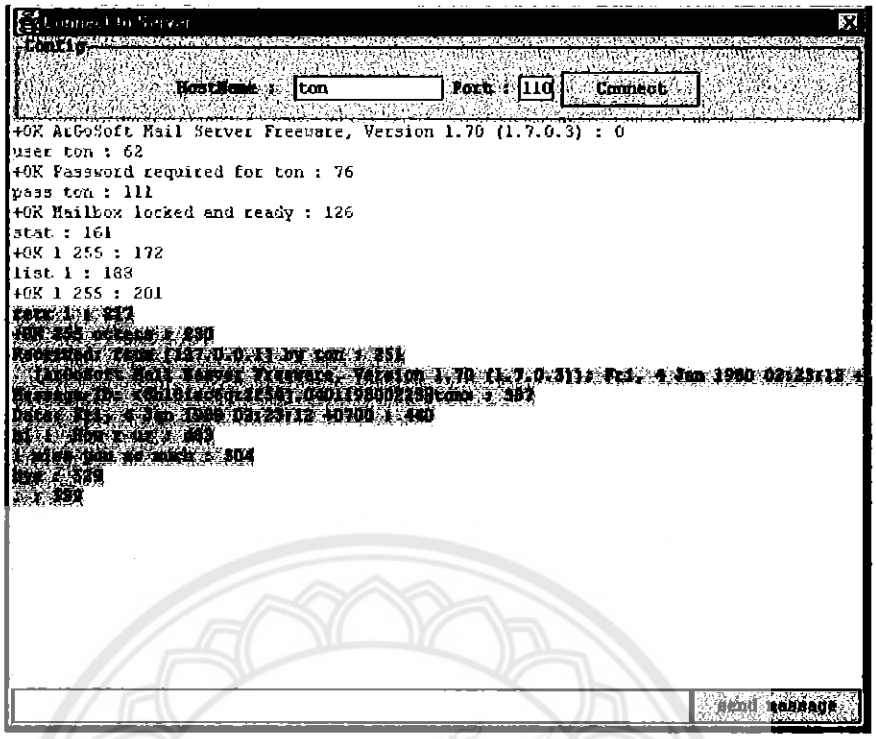

**รูปที่ 3.16** ใส่คำสั่ง user ตามด้วย user name ของเรา



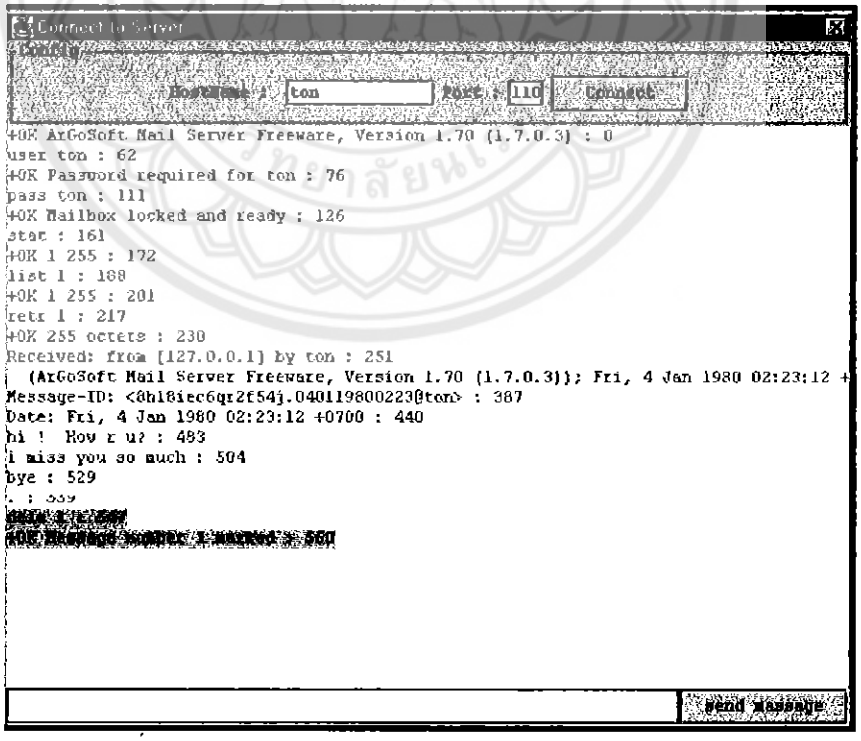

รูปที่ 3.17 ใส่คำสั่ง pass ตามด้วย password ของเรา



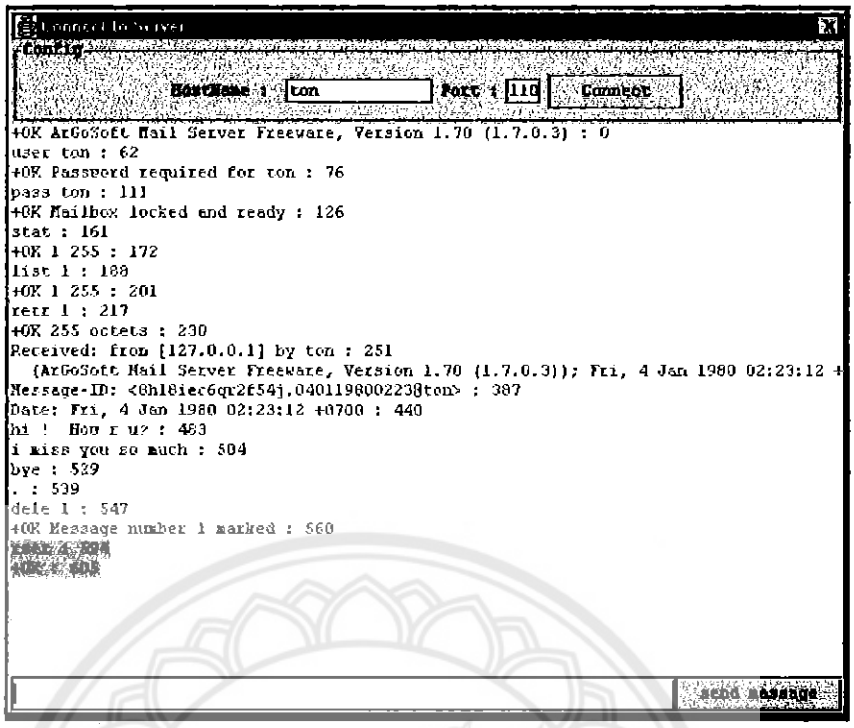

**รูปที่ 3.18 ใส่คำสั่ง stat เพื่อดูสภาพของเซิร์ฟเวอร์ เช่น จำนวน และ ขนาดของอีเมล์ ในเมล์บ็อกซ์**



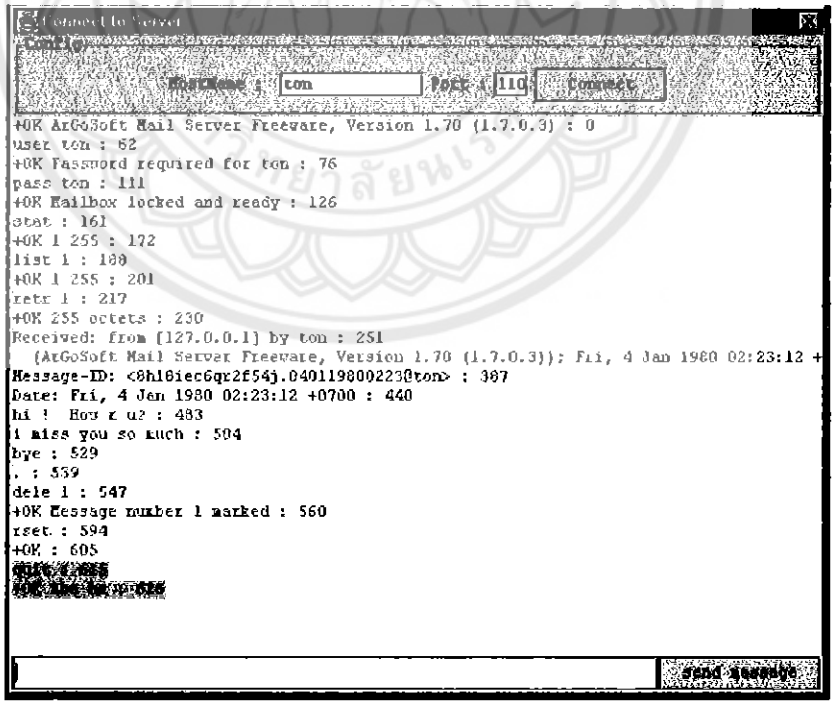

**รูปที่ 3.19** ใส่คำสั่ง retr ตามด้วยหมายเลขของจดหมายที่ต้องการจะรับข้อมูล จะพบข้อมูลและ ข้อความของจดหมาย




**รูปที่ 3.20** ใส่คำสั่ง dele ตามด้วยหมายเลขของจดหมาย เมื่อต้องการจะลบจดหมายที่รับข้อมูลมาแล้ว




**รูปที่ 3.21** ใส่คำสั่ง rset เมื่อต้องการจะยกเลิกการลบจดหมาย ที่ใช้คำสั่ง dele ไปแล้ว ในกรณีที่เรายังต้องการเก็บจดหมายไว้ที่เซิร์ฟเวอร์




**รูปที่ 3.22** ถ้าต้องการยกเลิกการติดต่อกับเซิร์ฟเวอร์ให้ใช้คำสั่ง quit

### 3.4 สรุปผลการทดลอง

◆ ผลจากการทดลองเขียนโปรแกรมหาค่าไอพีแอดเดรสของเครื่องภายในเครือข่ายเดียวกัน นั้นจะสามารถหาค่าไอพีแอดเดรสได้ตรงตามค่าไอพีแอดเดรสของเครื่อง

◆ ผลจากการทดลองนำโปรแกรมส่งจดหมายอิเล็กทรอนิกส์ที่เราได้สร้างขึ้น ส่งถึงโปรแกรมสำหรับเครื่องแม่ข่ายในการรับส่งจดหมายอิเล็คทรอนิกส์ที่เราเขียนขึ้นได้ผลเหมือนกับเมล์เซิร์ฟเวอร์อื่น ๆ ที่มีอยู่แล้วตามมาตรฐานเอสเอ็มทีพี และพ็อป3

# บทที่ 4
# ผลการทดสอบโปรแกรมและวิเคราะห์ผล

บทที่ 4 นี้ กล่าวถึงการทดสอบโปรแกรมว่ามีขั้นตอนการทดสอบอย่างไรและหลังจากทดสอบโปรแกรมแล้ว ผลการทำงานเป็นอย่างไร รวมถึงการวิเคราะห์ผลการทำงานว่ามีประสิทธิภาพมากน้อยเพียงใด มีข้อบกพร่องหรือต้องปรับปรุงแก้ไขในส่วนใดบ้าง

## 4.1 จุดประสงค์ของการทดสอบโปรแกรม

1. เพื่อทดสอบโปรแกรมว่าสามารถทำงานได้ผลและมีประสิทธิภาพมากน้อยเพียงใด บรรลุตามวัตถุประสงค์หรือไม่
2. เพื่อทดสอบว่าโปรแกรมสามารถรองรับการให้บริการได้ดีเพียงใด
3. เพื่อหาข้อผิดพลาดของโปรแกรมแล้วทำการปรับปรุงแก้ไขให้ดีขึ้น

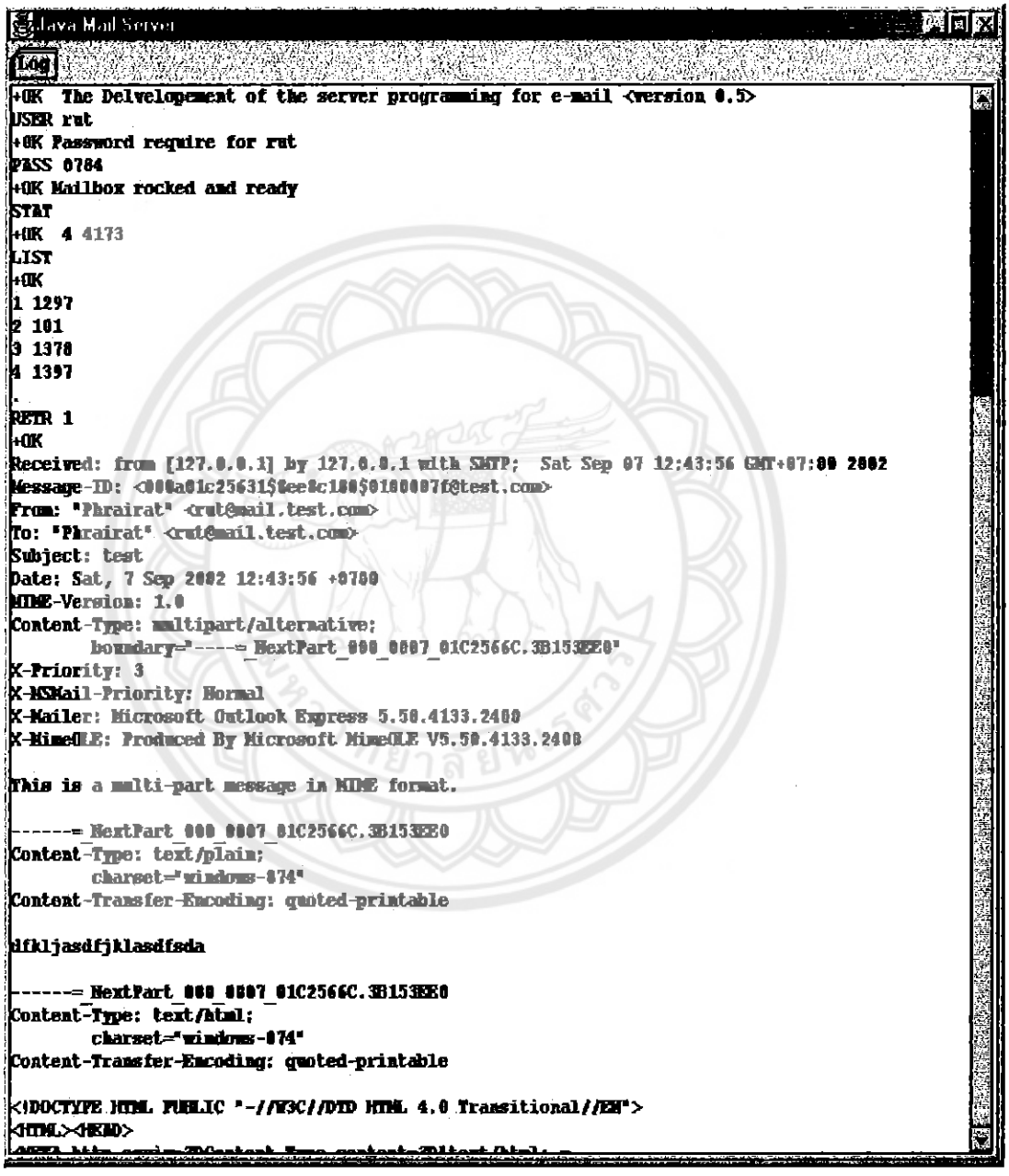
## 4.2 ขั้นตอนการทดสอบการทำงานของโปรแกรม

1. ติดตั้งโปรแกรมลงบนเครื่องคอมพิวเตอร์ แล้วทดลองส่งด้วยโปรแกรมเอาท์ลุกค์เอกซ์เพรสส่งไปที่โปรแกรมสำหรับเครื่องแม่ข่ายในการรับส่งจดหมายอิเล็คทรอนิกส์ให้ทำงานบนเครื่องเดียวกันซึ่งจะกำหนดให้ส่งจากยูสเซอร์หนึ่งไปอีกยังยูสเซอร์หนึ่งบนเครื่องเดียวกันว่าสามารถรับส่งจดหมายอิเล็คทรอนิกส์ได้หรือไม่
2. ทดสอบโปรแกรมการรับส่งอีเมล์กันระหว่างเครื่องโดยใช้โปรแกรมวีเอ็มแวร์(VM Ware) ซึ่งเป็นโปรแกรมจำลองการทำงานของคอมพิวเตอร์ทั่วไป ซึ่งสามารถจำลองทำงานหลาย ๆ เครื่อง บนเครื่องเดียวกันได้ และสามารถกำหนดได้ว่าอยู่ในวงแลนเดียวกันหรือต่างวงแลนกันก็ได้
3. ทดสอบโปรแกรมการรับส่งอีเมล์กันระหว่างเครื่องบนระบบเครือข่าย
4. ทดสอบโปรแกรมการรับส่งอีเมล์กันบนระบบเครือข่ายอินเตอร์เน็ต
5. ตรวจสอบผลที่ได้จากการทดสอบทั้ง 4 รูปแบบ ว่ามีความถูกต้องและมีประสิทธิภาพการทำงานเป็นอย่างไร ผลการทำงานมีความแตกต่างกันเมื่อทดสอบในรูปแบบที่ต่างกันหรือไม่
6. หาข้อผิดพลาดและทำการปรับปรุงแก้ไข

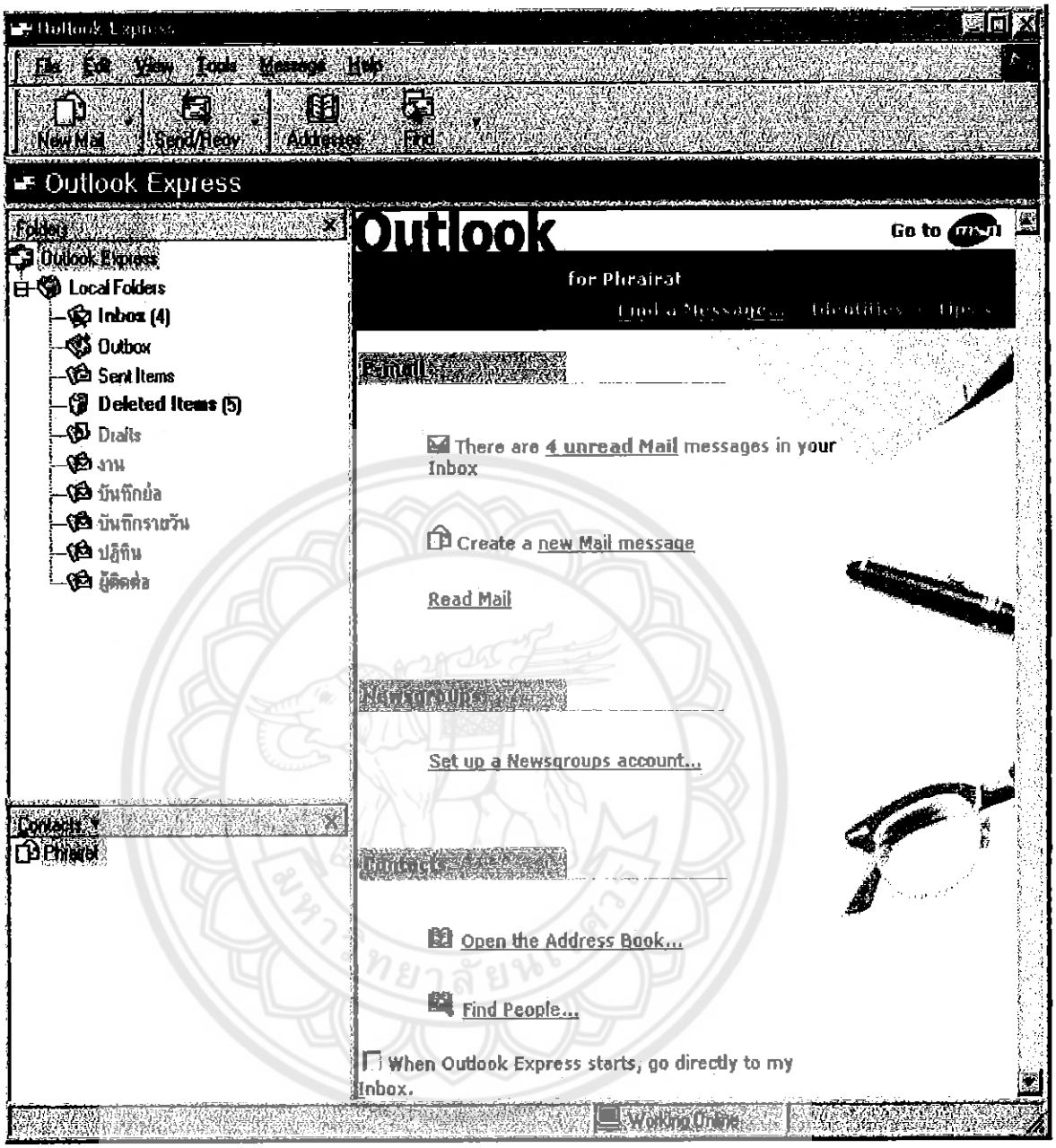
## 4.3 ผลการทดสอบ

### 4.3.1 การทดสอบการรับส่งอีเมล์ภายในเครื่องเดียวกัน

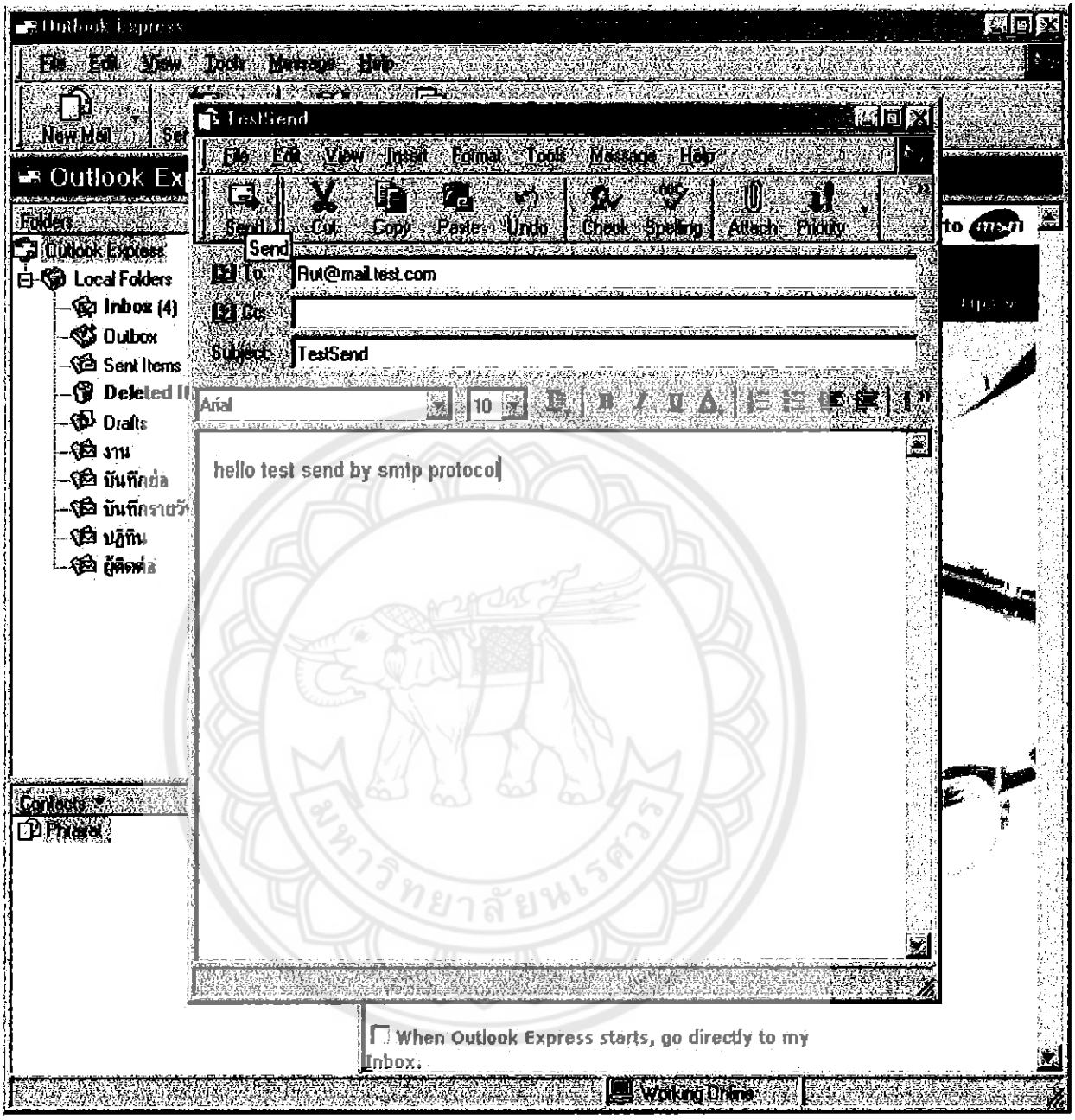
โดยส่งจะกำหนดให้ส่งจากยูสเซอร์หนึ่งไปยังอีกยูสเซอร์หนึ่งบนเครื่องเดียวกันโดยส่งจากโปรแกรมเอาท์ลุกส์เอ็กซ์เพรส




รูปที่ 4.1 แสดงการทดสอบการใช้เอาท์ลุกส์เอ็กซ์เพรสติดต่อกับโปรแกรมเมล์เซิร์ฟเวอร์ที่เขียนขึ้นโดยผ่านโปรโตคอลพ็อป3




**รูปที่ 4.2** เมื่อเอาท์ลุคค์เอ็กเพรสรับอีเมล์จากเมล์เซิร์ฟเวอร์ที่เราเขียนขึ้นจะแสดงผลว่าได้รับเมล์ใหม่มา 4 เมล์



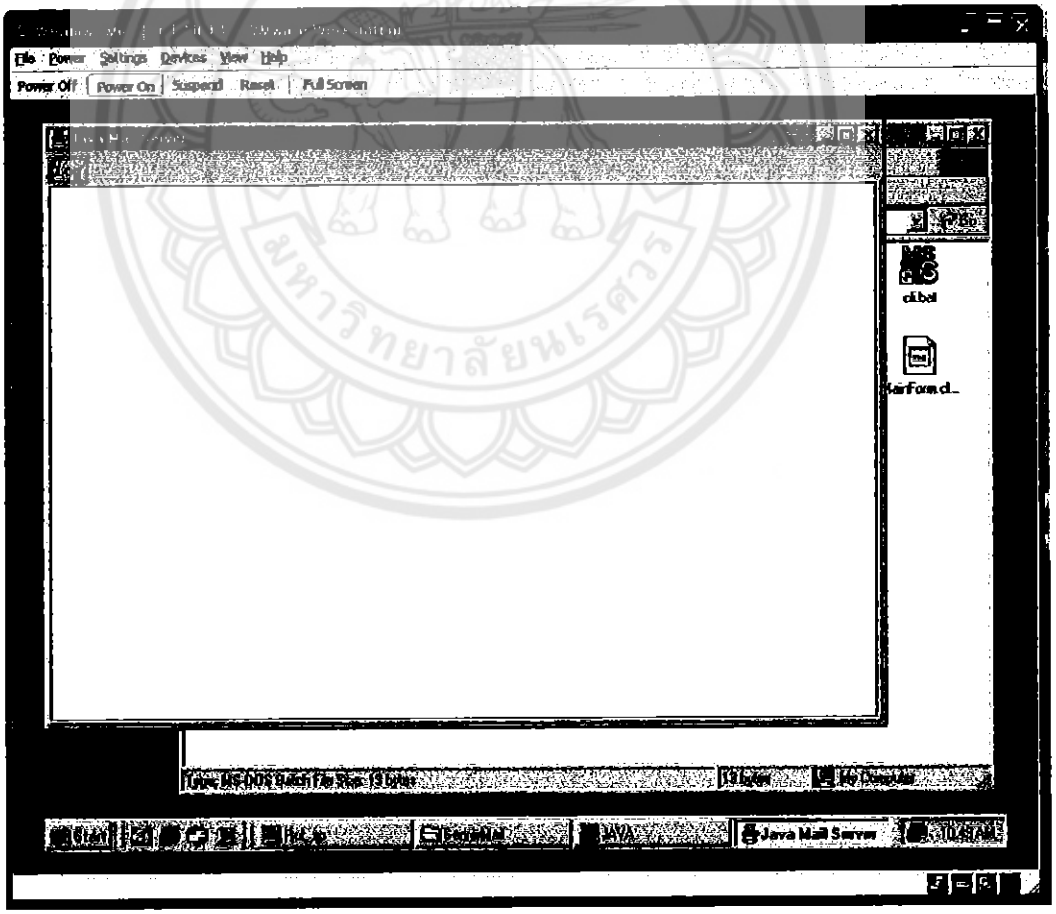

**รูปที่ 4.3** แสดงการทดสอบส่งเมล์ให้กับตัวเองโดยใช้โปรโตคอลเอสเอ็มทีพีผ่านโปรแกรมเมล์เซิร์ฟเวอร์ที่เราเขียนขึ้น


รูปที่ 4.4 ผลลัพท์ของการให้เอาท์ลุกค์เอกซ์เพรสส่งเมล์

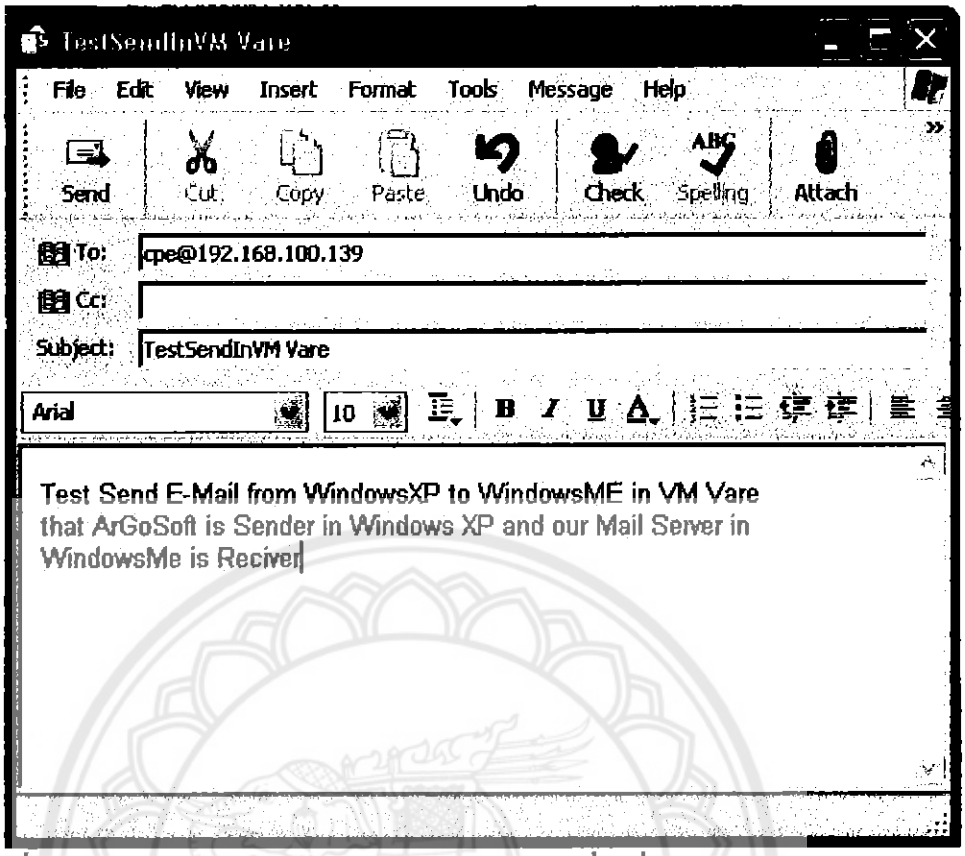
**4.3.2 การทดสอบการรับส่งอีเมล์ภายในเครื่องเดียวกันโดยใช้โปรแกรมวีเอ็มแวร์ (VM Ware)**

โปรแกรมวีเอ็มแวร์เป็นโปรแกรมจำลองการทำงานของคอมพิวเตอร์ทั่วไป ซึ่งสามารถจำลองการทำงานหลาย ๆ เครื่อง บนเครื่องเดียวกันได้ และสามารถกำหนดได้ว่าอยู่ในวงแลนเดียวกันหรือต่างวงแลนกันก็ได้ มีขั้นตอนการทดสอบ ดังนี้

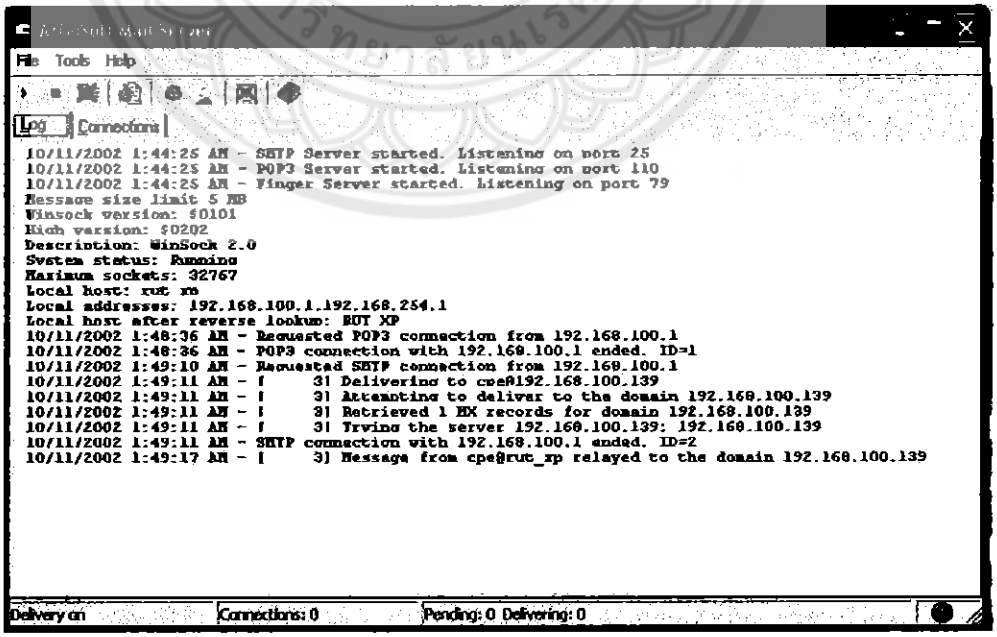
1. ลงโปรแกรมวีเอ็มแวร์ และลงระบบปฏิบัติการอื่นในเครื่องเดียวกัน ซึ่งจากการทดสอบได้ลงระบบปฏิบัติการวินโดวส์เอ็มอี (Windows Me) ในโปรแกรมวีเอ็มแวร์ และเครื่องที่ทดลองลงระบบปฏิบัติการวินโดวส์เอ็กซ์พี (Windows XP)
2. กำหนดให้โปรแกรมอาร์โก้เมล์ (Argo Mail) ซึ่งเป็นโปรแกรมเมล์เซิร์ฟเวอร์ทั่วไปที่เป็นไปตามมาตรฐานเอสเอ็มทีพี และพ็อป3 ทำงานในระบบปฏิบัติการวินโดวส์เอ็มอี
3. กำหนดให้โปรแกรมเมล์เซิร์ฟเวอร์ที่สร้างขึ้นทำงานในระบบปฏิบัติการวินโดวส์เอ็กซ์พี
4. ทดสอบส่งอีเมล์จากระบบปฏิบัติการวินโดวส์เอ็กซ์พีโดยใช้โปรแกรมเอาท์ลุกค์เอ็กซ์เพรส โดยกำหนดปลายทางเป็นยูสเซอร์หนึ่งในระบบปฏิบัติการวินโดว์เอ็มอี
5. ทำเช่นเดียวกับข้อข้างต้นแต่สลับหน้าที่ระหว่างผู้รับกับผู้ส่ง



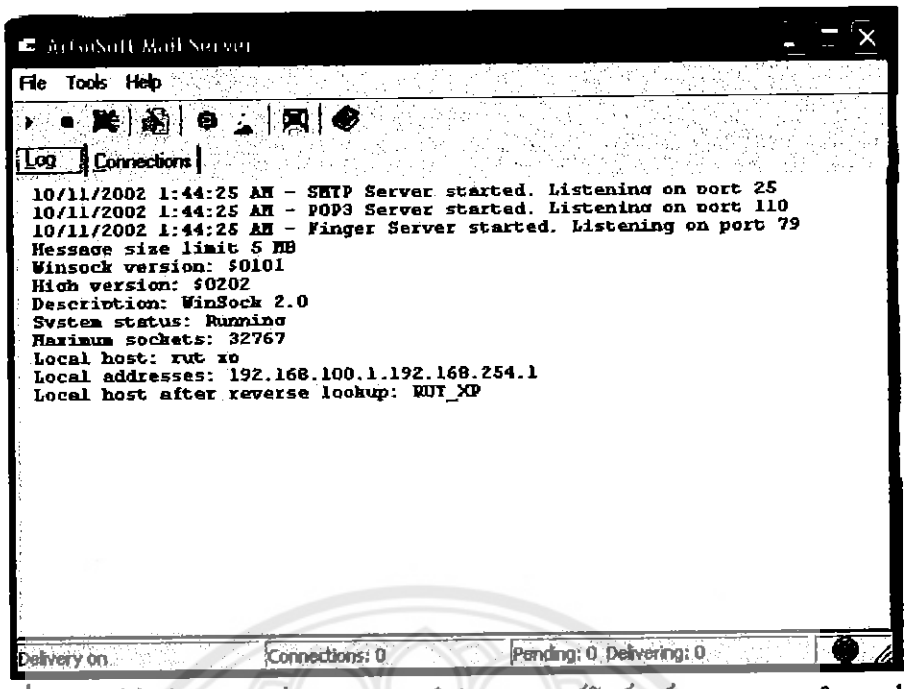

**รูปที่ 4.5** เริ่มต้นการทำงานโดยการเริ่มโปรแกรมเมล์เซิร์ฟเวอร์ของระบบปฏิบัติการวินโดวส์ เอ็มอีภายในโปรแกรมวีเอ็มแวร์



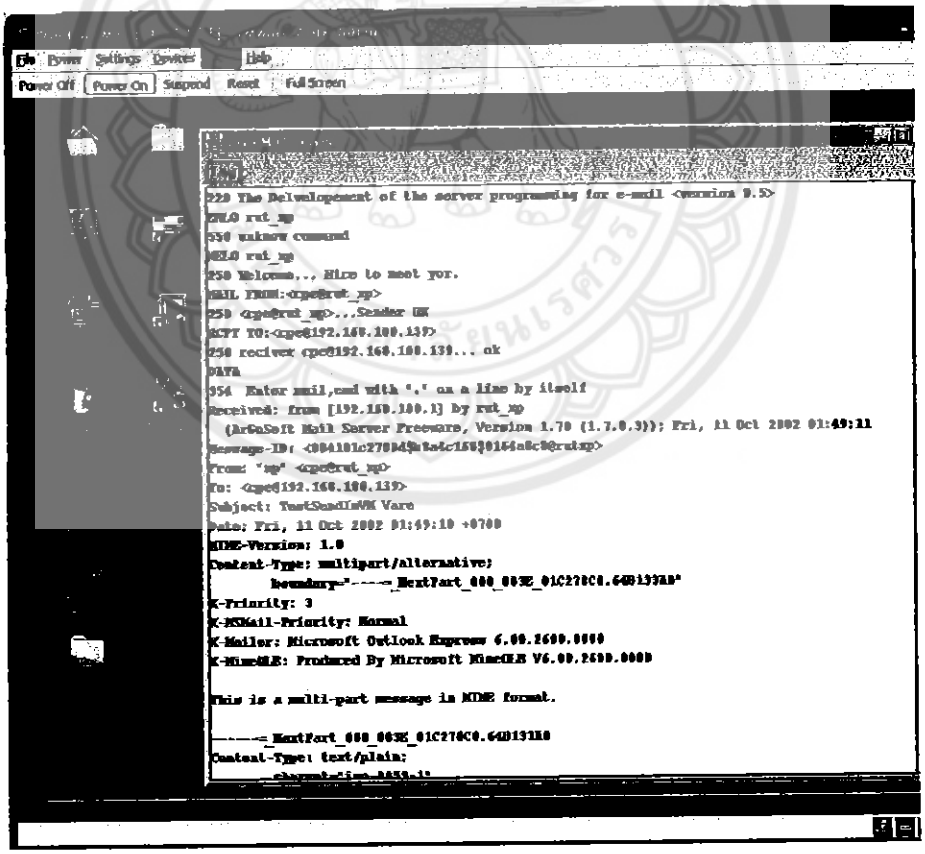

**รูปที่ 4.6** เริ่มทำงานโปรแกรมเมล์เซิร์ฟเวอร์อาร์โกว์เมล์บนเครื่องที่เป็นระบบปฏิบัติการวินโดวส์ เอ็กซ์พี เพื่อใช้ทำหน้าที่เป็นเมล์เซิร์ฟเวอร์มาตรฐาน ในการรับส่งเมล์ล์ กับโปรแกรมเมล์เซิร์ฟเวอร์ที่ได้เขียนขึ้น




**รูปที่ 4.7** แสดงถึงแอดเดรสและ ชื่อของผู้รับปลายทางและข้อความของอีเมล์ที่ใช้ทดสอบในการส่ง




**รูปที่ 4.8** เมื่อส่งเมล์ล์แล้วในส่วนล็อค (log)ของโปรแกรมอาร์โกว์เมล์จะแสดงการทำงานว่าได้ติดต่อไปที่เครื่องปลายทางแล้ว และเครื่องปลายทางก็ได้รับอีเมล์ที่ส่งไปแล้ว




**รูปที่ 4.9** แสดงการทำงานของเมล์เซิร์ฟเวอร์ที่พัฒนาขึ้น ว่ามีการรับเมล์จากอาร์โกว์เมล์แล้ว

**4.3.4 การทดสอบการรับส่งอีเมล์บนระบบเครือข่าย**

ในส่วนของการทดสอบผ่านระบบเครือข่ายนั้นมี ผลการทดสอบเหมือนกับผลการทดสอบโดยใช้โปรแกรมวีเอ็มแวร์ทุกประการ

**4.3.5 การทดสอบการรับส่งอีเมล์ผ่านระบบเครือข่ายอินเตอร์เน็ต**

การทำสอบของโปรแกรมผ่านระบบอินเตอร์เน็ตนั้นไม่สามารถทำได้เนื่องจากขณะที่ทดสอบนั้น การเชื่อมต่อระบบเครือข่ายอินเตอร์เน็ตได้เชื่อมต่อกับระบบเครือข่ายอินเทอร์เน็ตของมหาวิทยาลัย ซึ่งทำให้ไม่ได้ค่าไอพีจริง ได้ค่าไพรเวทไอพี (Private IP) ซึ่งอาจจะติดที่ไฟร์วอล์ล(Fire wall)

## 4.4 วิเคราะห์ผล

จากผลการทดสอบโปรแกรมปรากฏว่า ผลการทำงานในการทดสอบการรับส่งอีเมล์ภายในเครื่องเดียวกัน ผลการทดสอบการรับส่งอีเมล์ภายในเครื่องเดียวกันโดยใช้โปรแกรมวีเอ็มแวร์ (VM Ware) ผลการทดสอบการรับส่งอีเมล์บนระบบเครือข่ายกัน ให้ผลการทำงานที่เหมือนกัน คือสามารถรับส่งจดหมายอิเล็กทรอนิกส์ได้ แสดงว่ารูปแบบหรือสภาวะแวดล้อมของระบบเครือข่ายไม่มีผลต่อการทำงานของโปรแกรมที่สร้างขึ้น แต่ในการทดสอบโปรแกรมผ่านระบบเครือข่ายอินเตอร์เน็ตนั้นยังไม่สามารถได้เนื่องจากขณะที่ทดสอบนั้น การเชื่อมต่อระบบเครือข่ายอินเตอร์เน็ตได้เชื่อมต่อกับระบบเครือข่ายอินเทอร์เน็ตของมหาวิทยาลัย ซึ่งทำให้ไม่ได้ค่าไอพีจริง ได้ค่าไพรเวทไอพี (Private IP) ซึ่งอาจจะติดที่ไฟร์วอล์ล(Fire wall) และในการรับอีเมล์อาจจะต้องมีการลงทะเบียนทางอินเตอร์เน็ตเสียก่อน

# บทที่ 5
# สรุปผลและข้อเสนอแนะ

ในบทนี้เป็นการสรุปผลทั้งหมดของการทำโครงงานนี้ ซึ่งประกอบด้วยส่วนสรุปผล ปัญหาในการทำงาน ข้อเสนอแนะ และแนวทางในการพัฒนาต่อไป หากมีผู้ที่สนในที่จะนำโครงงานนี้ไปพัฒนาและปรับปรุงให้มีประสิทธิภาพดีขึ้นต่อไป

## 5.1 สรุปผล

1. โปรแกรมสำหรับเครื่องแม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ (mail server)สามารถให้บริการการรับและส่งจดหมายอิเล็กทรอนิกส์ ภายใต้โปรโตคอลเอสเอ็มทีพี (อาร์เอฟซี 822) ได้

2. โปรแกรมสำหรับเครื่องแม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ (mail server) ที่ทำขึ้นสามารถให้บริการรับส่งจดหมายอิเล็กทรอนิกส์ได้ ภายใต้โปรโตคอลพ็อป3 (อาร์เอฟซี1939) ได้

3. สามารถใช้โปรแกรมสำหรับเครื่องแม่ข่าย เพื่อรับส่งจดหมายอิเล็กทรอนิกส์ที่สร้างขึ้น ในระบบเครือข่ายได้

4. ยังไม่สามารถใช้โปรแกรมผ่านระบบอินเตอร์เน็ตได้

## 5.2 ปัญหาในการทำงาน

1. ในการทดสอบการรับส่งอีเมล์ภายในเครื่องเดียวกันโดยใช้โปรแกรมวีเอ็มแวร์ (VM Ware) นั้น จะทำงานได้ช้าเนื่องจากโปรแกรมวีเอ็มแวร์จะดึงทรัพยากรในคอมพิวเตอร์ไปใช้ร่วมกัน

2. ในการทดสอบร่วมกับเมล์เซิร์ฟเวอร์มาตรฐานอื่น ๆ นั้น หากเซ็ตโปรแกรมไม่ถูกต้องทำให้ไม่สามารถทำงานร่วมกันได้

3. ในการทดสอบโปรแกรมผ่านระบบอินเตอร์เน็ตยังไม่สามารถทำได้

4. โปรแกรมเมล์เซิร์ฟเวอร์เป็นโปรแกรมที่ซับซ้อนเป็นอย่างมาก จึงยากต่อการออกแบบให้ทำงานได้ภายในครั้งเดียว ต้องมีการออกแบบแล้วลองนำมาสร้างให้ทำงานจริง เมื่อมีข้อผิดพลาดเกิดขึ้นในส่วนที่ไม่เป็นไปตามจุดประสงค์ก็ต้องออกแบบใหม่หมดเลยซึ่งทำให้ล่าช้าเป็นอย่างมาก

5. เนื่องจากข้อมูลที่ใช้ในการค้นคว้า ทั้งในเรื่องของโปรโตคอล และการติดต่อสื่อสารระหว่างโปรเซส มีข้อมูลเป็นจำนวนมากทำให้เสียเวลาในการศึกษา

6. การพัฒนาโปรแกรมนี้ใช้ภาษาจาวา (JAVA) ในการพัฒนา ซึ่งเป็นภาษาที่ผู้จัดทำไม่คุ้นเคยในการใช้งานมาก่อน ทำให้ต้องใช้เวลาในการศึกษาการใช้งานภาษาจาวามาก มีผลทำให้การพัฒนาโปรแกรมเป็นไปอย่างล่าช้า

## 5.3 แนวทางแก้ไขและข้อเสนอแนะ

1. การทดสอบการรับส่งอีเมล์ภายในเครื่องเดียวกันโดยใช้โปรแกรมวีเอ็มแวร์ (VM Ware) นั้นทำงานได้ช้า อาจจะต้องเพิ่มทรัพยากรให้กับคอมพิวเตอร์ หรือ นำเครื่องอีกเครื่องมาทดสอบด้วยกันจะรวดเร็วขึ้น

2. ในการทดสอบร่วมกับเมล์เซิร์ฟเวอร์มาตรฐานอื่น ๆ ควรศึกษาคู่มือการใช้ เพื่อเซ็ตค่าต่าง ๆ ให้ถูกต้องจะได้ไม่มีปัญหาในการทำงานร่วมกัน

3. ในการทดสอบการรับส่งอีเมล์ผ่านระบบเครือข่ายอินเตอร์เน็ตน่าจะทดสอบโปรแกรมผ่านระบบอินเตอร์เน็ตที่ไม่ได้เชื่อมต่อกับทางมหาวิทยาลัยหรือองค์กรต่าง ๆ ที่จะได้ค่าไอพีจริง

4. ในการออกแบบโปรแกรมควรออกแบบโครงสร้างของโปรแกรมในรูปแบบแผนผังลำดับงาน (Flow Chart) ในทุก ๆ ด้านของการทำงาน เพื่อให้ง่ายต่อการออกแบบและพัฒนาโปรแกรม

5. ควรพัฒนาให้ได้ตามมาตรฐานที่อ้างอิง เพื่อให้สามารถใช้งานร่วมกับโปรแกรมเมล์เซิร์ฟเวอร์มาตรฐานอื่น ๆ ได้

6. ควรศึกษาการทำงานของโปรโตคอล เอสเอ็มทีพี และ พ็อป3 การติดต่อสื่อสารระหว่างโปรเซส ภาษาที่จะใช้ในการพัฒนา ตลอดจนข้อมูลอื่น ๆ เพื่อจะนำไปใช้ในการปรับปรุงแก้ไขโปรแกรมในโอกาสต่อไป

## 5.4 แนวทางในการพัฒนา

1. ศึกษาข้อมูลและการทำงานของ อาร์เอฟซี ส่วนขยายอื่น ๆ ที่เกี่ยวข้อง แล้วนำมาพัฒนาส่วนของคำสั่งส่วนขยายอื่นๆเพื่อให้ได้โปรแกรมสำหรับเครื่องแม่ข่ายในการรับส่งจดหมายอิเล็กทรอนิกส์ที่มีประสิทธิภาพในการทำงาน

2. เพิ่มประสิทธิภาพการทำงานในส่วนต่าง ๆ ของโปรแกรม เช่น MIME (Multipurpose Internet Mail Extensions) เป็นข้อกำหนดขยายการทำงานของอินเทอร์เน็ตเมล์ให้สามารถรับส่งข้อมูลไบนารีได้โดยไม่ต้องใช้โปรแกรมแปลงข้อมูล และมีการเข้ารหัส ทำให้มีความปลอดภัยในจดหมายอิเล็กทรอนิกส์ของเรา

3. พัฒนาโปรแกรมเพิ่มเติมเพื่อให้โปรแกรมสามารถรองรับการทำงานแบบ IMAP ที่ซับซ้อนกว่า พ็อป และมีขีดความสามารถหลายด้านเหนือกว่า พ็อป

# เอกสารอ้างอิง

[1] จริยา มุ่งคีมกลาง และคณะ. **การพัฒนาโปรแกรมสำหรับเครื่องคอมพิวเตอร์แม่ข่ายเพื่อถ่ายโอนข้อมูลแบบเอชทีทีพี (HTTP).** ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์. 2543

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

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

[4] วิภา เพิ่มทรัพย์, สาโรจน์ ไพชยนต์ฤทธา. **Linux Red Hat 6.1.** กรุงเทพมหานคร : โปรวิชั่น, 2543

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

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

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

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

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

# ภาคผนวก

## พ็อป 3 (POP3 : Post Office Protocol)

**สถานะขออนุมัติ (Authorization State)**

**คำสั่ง พ็อป3 ที่ใช้ในสถานะขออนุมัติ มีดังต่อไปนี้**

**เมื่อการเชื่อมต่อ TCP ถูกเปิดโดยไคลเอ็นท์ พ็อป3 เครื่องแม่ข่าย พ็อป3 จะทำการทักทาย ซึ่งจะมีการตอบกลับเป็นบวก ดังตัวอย่าง:**

S: +OK POP3 server ready

**คำสั่ง QUIT**

พารามิเตอร์ : ไม่ระบุ

สถานะ : ขออนุมัติ

รายละเอียด : เมื่อขออนุมัติใหม่แล้วจะตอบกลับ +OK

**ตัวอย่าง**

C: QUIT

S: +OK dewey POP3 server signing off

**สถานะรับส่งรายการ (Transaction State)**

**คำสั่ง พ็อป3 ที่ใช้ในสถานะรับส่งรายการ มีดังต่อไปนี้**

**คำสั่ง STAT**

พารามิเตอร์ : ไม่ระบุ

สถานะ : รับส่งรายการ

รายละเอียด : เป็นคำสั่งตรวจสอบสภาพของ server เช่น จำนวนอีเมล์ใน server , ขนาดของอีเมล์ที่จะดาวโหลด

**ตัวอย่าง**

C: STAT

S: + OK 2 320

**คำสั่ง LIST**

พารามิเตอร์ : หมายเลขข้อความ

สถานะ : รับส่งรายการ

รายละเอียด : ใช้ตรวจสอบหมายเลขของอีเมล์ และ ขนาดของอีเมล์

**ตัวอย่าง**

C: LIST

S: +OK 2 messages (320 octets)

S: 1 120

S: 2 200

S: .

...

C: LIST 2

S: +OK 2 200

...

C: LIST 3

S: -ERR no such message, only 2 messages in maildrop

**คำสั่ง RETR**

พารามิเตอร์ : ข้อความ

สถานะ : รับส่งรายการ

รายละเอียด : เป็นคำสั่งที่ใช้ส่งข้อมูลของอีเมล์

**ตัวอย่าง**

C: RETR 1

S: +OK 120 octets

S: <the POP3 server sends the entire message here>

S: .

**คำสั่ง DELE**

พารามิเตอร์ : ข้อความ

สถานะ : รับส่งรายการ

รายละเอียด : เป็นการระบุเครื่องหมายการลบ ลงในอีเมล์ที่จะลบ และอีเมล์เหล่านั้นจะถูกลบออกจากเมล์บ๊อกส์เมื่อใช้คำสั่ง QUIT เมื่อสิ้นสุดการทำงาน

**ตัวอย่าง**

C: DELE 1

S: +OK message 1 deleted

...

C: DELE 2

S: -ERR message 2 already deleted

**คำสั่ง NOOP**

พารามิเตอร์ : ไม่ระบุ

สถานะ : รับส่งรายการ

รายละเอียด : เป็นคำสั่ง No Operation

**ตัวอย่าง**

C: NOOP

S: +OK

**คำสั่ง RSET**

พารามิเตอร์ : ไม่ระบุ

สถานะ : รับส่งรายการ

รายละเอียด : คำสั่งนี้จะยกเลิกเครื่องหมายการลบอีเมล์ ที่เคยกำหนดไว้ด้วยคำสั่ง DELE ออกๆไปทุกๆ อีเมล์

**ตัวอย่าง**

C: RSET

S: +OK maildrop has 2 messages (320 octets)

**สถานะปรับปรุงข้อมูล (Update State)**

**คำสั่ง พ็อป3 ที่ใช้ในสถานะปรับปรุงข้อมูล มีดังต่อไปนี้**

**<u>คำสั่ง QUIT</u>**

**พารามิเตอร์** : ไม่ระบุ

**สถานะ** : รับส่งรายการและขออนุมัติ

**รายละเอียด** : ใช้เมื่อจบการทำงาน หากมีอีเมล์ซึ่งทำเครื่องหมายว่าจะลบไว้ อีเมล์เหล่านั้นจะถูกลบจากเมล์บ๊อกซ์ในขั้นตอนนี้

**ตัวอย่าง**

C: QUIT

S: +OK dewey POP3 server signing off (maildrop empty)

...

C: QUIT

S: +OK dewey POP3 server signing off (2 messages left)

...

**2.5.4 คำสั่งพิเศษของป๊อฟ 3 (Optional POP3 Commands)**

**นอกจากคำสั่งต่างๆ ที่กล่าวมาในข้างต้นยังมีคำสั่งพิเศษที่จะเอามาใช้กับ พ็อป3 Server ได้ ดังนี้**

**<u>คำสั่ง UIDL</u>**

**พารามิเตอร์** : หมายเลขข้อความ

**สถานะ** : รับส่งรายการ

**รายละเอียด** :ใช้ตรวจสอบหมายเลขประจำของอีเมล์

**ตัวอย่าง**

C: UIDL

S: +OK

S: 1 whqtswO00WBw418f9t5JxYwZ

S: 2 QhdPYR:00WBw1Ph7x7

S: .

...

C: UIDL 2

S: +OK 2 QhdPYR:00WBw1Ph7x7

...

C: UIDL 3

S: -ERR no such message, only 2 messages in maildrop

**คำสั่ง TOP**

พารามิเตอร์ : หมายเลขบรรทัด , จำนวนบรรทัด

สถานะ : รับส่งรายการ

รายละเอียด : Server จะส่งข้อมูลย้อนกลับไปเท่ากับจำนวนบรรทัดที่ระบุไว้

**ตัวอย่าง**

C: TOP 1 10

S: +OK

S: <the POP3 server sends the headers of the

message, a blank line, and the first 10 lines

of the body of the message>

S: .

...

C: TOP 100 3

S: -ERR no such message

**คำสั่ง APOP**

พารามิเตอร์ : ชื่อ, password

สถานะ : ขออนุมัติ

รายละเอียด : ทำหน้าที่ ให้ ระบุชื่อผู้ใช้ เมล์บ๊อกซ์ที่จะใช้ และ Password และข้อมูลจะถูกเข้ารหัสก่อนที่จะถูกส่งไป

**ตัวอย่าง**

S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>

C: APOP mrose c4c9334bac560ecc979e58001b3e22fb

S: +OK maildrop has 1 message (369 octets)

## ตัวอย่างของพ็อป3

S: <wait for connection on TCP port 110>

C: <open connection>

S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>

C: APOP mrose c4c9334bac560ecc979e58001b3e22fb

S: +OK mrose's maildrop has 2 messages (320 octets)

C: STAT

S: +OK 2 320

C: LIST

S: +OK 2 messages (320 octets)

S: 1 120

S: 2 200

S: .

C: RETR 1

S: +OK 120 octets

S: <the POP3 server sends message 1>

S: .

C: DELE 1

S: +OK message 1 deleted

C: RETR 2

S: +OK 200 octets

S: <the POP3 server sends message 2>

S: .

C: DELE 2

S: +OK message 2 deleted

C: QUIT

S: +OK dewey POP3 server signing off (maildrop empty)

C: <close connection>

S: <wait for next connection>

## เอสเอ็มทีพี (SMTP : Simple Mail Transfer Protocol)

### ตัวอย่างการทำงาน

patty@can.org…. Connecting to mailgw.can.org. via smtp…

250 milgw.can.org SMTP Sendmail 8.8.5; Mon, 26 Aug 2002 23:30:22 –0400

>>> HELLO patty.nu.ac.th

250 mailgw.can.org Hello patty@nontri.nu.ac.th, pleased to meet you

>>> MAIL From: <patty@nontri.nu.ac.th> SIZE = 56

250 <patty@nontri.nu.ac.th>… Sender ok

>>> RCPT To:<patty@can.org>

250 <patty@can.org>… Recipient ok

>>> DATA

354 Enter mail, end with "." on a line by itself

>>> .

250 XAA07990 Message accepted for delivery

patty@can.org…. Sent (XAA07990 Message accepted for delivery)

Closing connection to mailgw.can.org.

>>> QUIT

221 mailgw.can.org closing connection

### ตัวอย่างของการส่งไป

S: RCPT TO:<Postel@USC-ISI.ARPA>

R: 251 User not local; will forward to <Postel@USC-ISIF.ARPA>

**หรือ**

S: RCPT TO:<Paul@USC-ISIB.ARPA>

R: 551 User not local; please try <Mockapetris@USC-ISIF.ARPA>

### ตัวอย่างของการพิสูจน์และขยายรายชื่อจดหมาย

S: VRFY Smith

R: 250 Fred Smith <Smith@USC-ISIF.ARPA>

**หรือ**

S: VRFY Smith

R: 251 User not local; will forward to <Smith@USC-ISIQ.ARPA>

หรือ

S: VRFY Jones

R: 550 String does not match anything.

หรือ

S: VRFY Jones

R: 551 User not local; please try <Jones@USC-ISIQ.ARPA>

หรือ

S: VRFY Gourzenkyinplatz

R: 553 User ambiguous.

ตัวอย่างส่งข่าวสารให้รายชื่อผู้ใช้

S: EXPN Example-People

R: 250-Jon Postel <Postel@USC-ISIF.ARPA>

R: 250-Fred Fonebone <Fonebone@USC-ISIQ.ARPA>

R: 250-Sam Q. Smith <SQSmith@USC-ISIQ.ARPA>

R: 250-Quincy Smith <@USC-ISIF.ARPA:Q-Smith@ISI-VAXA.ARPA>

R: 250-<joe@foo-unix.ARPA>

R: 250 <xyz@bar-unix.ARPA>

หรือ

S: EXPN Executive-Washroom-List

R: 550 Access Denied to You.

ตัวอย่างของการติดต่อเพื่อเปิด

R: 220 BBN-UNIX.ARPA Simple Mail Transfer Service Ready

S: HELO USC-ISIF.ARPA

R: 250 BBN-UNIX.ARPA

ตัวอย่างของการติดต่อเพื่อปิด

S: QUIT

R: 221 BBN-UNIX.ARPA Service closing transmission channel

**ตัวอย่างของเมล์ที่ไม่ได้ส่งซึ่งจะแจ้งด้วยข้อความ**

S: MAIL FROM:<>

R: 250 ok

S: RCPT TO:<@HOSTX.ARPA:JOE@HOSTW.ARPA>

R: 250 ok

S: DATA

R: 354 send the mail data, end with .

S: Date: 23 Oct 81 11:22:33

S: From: SMTP@HOSTY.ARPA

S: To: JOE@HOSTW.ARPA

S: Subject: Mail System Problem

S:

S: Sorry JOE, your message to SAM@HOSTZ.ARPA lost.

S: HOSTZ.ARPA said this:

S: "550 No Such User"

S: .

R: 250 ok

**ตัวอย่างของการกลับเส้นทางเดินและรับเวลาที่จำกัด**

MAIL FROM:< >

Return-Path: <@GHI.ARPA,@DEF.ARPA,@ABC.ARPA:JOE@ABC.ARPA>

Received: from GHI.ARPA by JKL.ARPA ; 27 Oct 81 15:27:39 PST

Received: from DEF.ARPA by GHI.ARPA ; 27 Oct 81 15:15:13 PST

Received: from ABC.ARPA by DEF.ARPA ; 27 Oct 81 15:01:59 PST

Date: 27 Oct 81 15:01:01 PST

From: JOE@ABC.ARPA

Subject: Improved Mailing System Installed

To: SAM@JKL.ARPA

This is to inform you that ...

The following are the SMTP commands:

HELO <SP> <domain> <CRLF>

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

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

DATA <CRLF>

RSET <CRLF>

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

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

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

VRFY <SP> <string> <CRLF>

EXPN <SP> <string> <CRLF>

HELP [<SP> <string>] <CRLF>

NOOP <CRLF>

QUIT <CRLF>

TURN <CRLF>

**ตัวอย่างของการกลับเส้นทางเดิน**

Return-Path: <@CHARLIE.ARPA,@BAKER.ARPA:JOE@ABLE.ARPA>

**ตัวอย่างของเส้นเวลาที่จำกัด**

Received: FROM ABC.ARPA BY XYZ.ARPA ; 22 OCT 81 09:23:59 PDT

Received: from ABC.ARPA by XYZ.ARPA via TELENET with X25

id M12345 for Smith@PDQ.ARPA ; 22 OCT 81 09:23:59 PDT

**ตัวอย่างของเหตุการณ์ที่เอสเอ็มทีพีติดต่อไม่สำเร็จ**

R: 220 MIT-Multics.ARPA Simple Mail Transfer Service Ready

S: HELO ISI-VAXA.ARPA

R: 250 MIT-Multics.ARPA

S: MAIL FROM:<Smith@ISI-VAXA.ARPA>

R: 250 OK

S: RCPT TO:<Jones@MIT-Multics.ARPA>

R: 250 OK

S: RCPT TO:<Green@MIT-Multics.ARPA>

R: 550 No such user here

S: RSET

R: 250 OK

S: QUIT

R: 221 MIT-Multics.ARPA Service closing transmission channel

**ตัวอย่างของรีเลย์เมล์**

**ระดับที่ 1 จากต้นทางโฮสต์ไปรีเลย์โฮสต์**

R: 220 USC-ISIE.ARPA Simple Mail Transfer Service Ready

S: HELO MIT-AI.ARPA

R: 250 USC-ISIE.ARPA

S: MAIL FROM:<JQP@MIT-AI.ARPA>

R: 250 OK

S: RCPT TO:<@USC-ISIE.ARPA:Jones@BBN-VAX.ARPA>

R: 250 OK

S: DATA

R: 354 Start mail input; end with <CRLF>.<CRLF>

S: Date: 2 Nov 81 22:33:44

S: From: John Q. Public <JQP@MIT-AI.ARPA>

S: Subject: The Next Meeting of the Board

S: To: Jones@BBN-Vax.ARPA

S:

S: Bill:

S: The next meeting of the board of directors will be

S: on Tuesday.

S: John.

S: .

R: 250 OK

S: QUIT

R: 221 USC-ISIE.ARPA Service closing transmission channel

**ระดับที่ 2 จากรีเลย์โฮสต์ไปปลายทางโฮสต์**

R: 220 BBN-VAX.ARPA Simple Mail Transfer Service Ready

S: HELO USC-ISIE.ARPA

R: 250 BBN-VAX.ARPA

S: MAIL FROM:<@USC-ISIE.ARPA:JQP@MIT-AI.ARPA>

R: 250 OK

S: RCPT TO:<Jones@BBN-VAX.ARPA>

R: 250 OK

S: DATA

R: 354 Start mail input; end with <CRLF>.<CRLF>

S: Received: from MIT-AI.ARPA by USC-ISIE.ARPA ; 2 Nov 81 22:40:10 UT

S: Date: 2 Nov 81 22:33:44

S: From: John Q. Public <JQP@MIT-AI.ARPA>

S: Subject: The Next Meeting of the Board

S: To: Jones@BBN-Vax.ARPA

S:

S: Bill:

S: The next meeting of the board of directors will be

S: on Tuesday.

S: John.

S: .

R: 250 OK

S: QUIT

R: 221 USC-ISIE.ARPA Service closing transmission channel

**ซอสโค้ดของโปรแกรม (Source of program)**

**MainForm.java**

ฟอร์มหลักของโปรแกรมเป็นคลาสที่ใช้ในการสร้าง frame หลักในการทำงานทั้งการแสดงผลคำสั่งต่าง ๆ ที่ใช้ติดต่อกันและส่วนที่เป็น serversocket เพื่อใช้ในการ listen รอการติดต่อจาก user agent

```
import javax.swing.*;
import java.awt.*;
import server.*;
import components.*;
import MainTabbedPane.*;
import java.lang.*;
import system.*;
public class MainForm extends JFrame
{
 Server serv;
 MainTabbedPane mtp;
 String message_in;
 public system sys;
 Relay relay;
 MainForm(String args[])
 {
  super("Java Mail Server");
  sys = new system("mailsys.conf");

  Integer  Iport =new Integer(sys.con.getProperty("Port Pop3"));
  int iport = Iport.intValue();
  serv= new Server(iport,this);

  Iport =new Integer(sys.con.getProperty("Port SMTP"));
  iport = Iport.intValue();
  serv= new Server(iport,this);
```

```
    // for Relay
    relay = new Relay(this);
    //

    Container cp = getContentPane();
    setSize(700,500);
    //setMenuBar(new MainMenuBar());
    mtp = new MainTabbedPane();
    cp.add(mtp);
    if(args.length>0)
    {
     if(args[0].equals("-b"))
     {
      setVisible(false);
     }else{setVisible(true);}
    }else{setVisible(true);}


  }
  public void read(String mess)
  {
   message_in=mess;
   mtp.read(message_in);

  }
  public static void main(String args[])
  {
   new MainForm(args);
  }
}
```

**Server.java**

class Server เป็นคลาสที่ทำหน้าที่รอการติดต่อจาก User agent เมื่อมีการติดต่อ คลาสนี้จะสร้างคลาส Service ขึ้นมาเพื่อให้บริการแก่เครื่องที่เข้ามาติดต่อ

```
package server;
import MainForm;
import server.services.*;
import java.net.*;
public class Server extends Thread
{
 private ServerSocket socket_server;
 public Socket socket_client;
 String message_in;
 public int port;
 public MainForm mf;
 public Server(int port,MainForm mf)
 {
  try
  {
   this.mf=mf;
   this.port=port;
   //create Server socket wait for connect form client
   socket_server=new ServerSocket(port);

   char n[]=new char[255];
   String m;
   m = socket_server.getInetAddress().getLocalHost().getHostAddress();
   System.out.println("Host name :" + m);
   //socket_client.getInetAddress()
  }catch(Exception e)
  {
   e.printStackTrace();
```

```
    }
    this.start();
  }
  public void read(String mess)
  {
    message_in=mess;
    mf.read(message_in);
  }
  public void run()
  {
    try
    {
      while(true)
      {
        //if client connect to this server
        if((socket_client=socket_server.accept())!=null)
        {
          //give service to client
          service service_client = new service(socket_client,this);

          service_client.start();
        }
      }
    }catch(Exception e)
    {
      e.printStackTrace();
    }
  }
}
```

**service.java**

คลาส service จะสร้าง คลาส ViaIn และ ViaOut ขึ้นเพื่อใช้ในการติดต่อกับ User agent ทางพอร์ตที่ถูกร้องขอมาและจะสร้าง คลาส MsgCon เพื่อเป็นตัวควบคุมสถานะการทำงานของการให้บริการ

```
package server.services;
import java.net.*;
import server.*;
import server.services.via.*;
import server.MsgCon.*;
public class service extends Thread
{
 private viaIn in;
 private viaOut out;
 public Server ser;
 String message_in;
 MsgCon msg_con;
 Socket Client;
 public service(Socket Client,Server ser)
 {
  this.Client=Client;
  in = new viaIn(Client,this);
  out = new viaOut(Client);
  this.ser=ser;
  msg_con= new MsgCon(this);

  ////////////////////
  String m = new String(Client.getInetAddress().getHostAddress());
  System.out.println(m);
  ////////////////////////////
 }
```

```
public void read(String mess)
{
 message_in=mess;
 ser.read(message_in);
 msg_con.read(message_in);
}
public void write(String mess)
{
 if(mess.equals("Stop service")) disconnect();
 out.send(mess);

}
private void disconnect()
{
 try
 {
  Client.close();
  in.stop();
  this.stop();
 }catch(Exception e){e.printStackTrace();}
}
public void run()
{
 in.start();
 try{
 in.join();
 }catch(Exception e)
 {
  e.printStackTrace();
 }
 System.out.println("Exit from method run Server Thread");
}
```

```
 public String getIPClient()
 {
  return Client.getInetAddress().getHostAddress();
 }
}
```

**MsgCon.java**

คลาส MsgCon เป็นคลาสที่ให้เป็นตัวแยกว่าการติดต่อจาก User agent นั้นใช้บริการของ protocol ใด แล้ว จะสร้าง คลาส pop3 หรือ smtp แล้วแต่การร้องขอของ User agent และคลาสนี้ทำหน้าที่เป็นตัวควบคุมเส้นทางในการส่งข้อมูลของคำสั่งจาก User agent ไปยังคลาส pop3 หรือ smtp อย่างถูกต้อง

```
package server.MsgCon;

import java.util.*;
import java.lang.Integer;
import system.*;
import server.services.*;
import server.MsgCon.protocol.*;
public class MsgCon
{
 public service sv;
 int port;
 config p ;
 String strProtocol;
 //Protocol ptcProtocol;
 SMTP smtp;
 POP3 pop3;
 public MsgCon(service sv)
 {

  this.sv=sv;
```

```
 this.port=this.sv.ser.port;
 p = this.sv.ser.mf.sys.con;
 findProtocol();
 //ptcProtocol= new protocol(this);
}
private void findProtocol()
{
 Enumeration e = p.propertyNames();
 String pName=new String();
 String strPort = Integer.toString(port);
 while(e.hasMoreElements())
 {
  pName=(String)e.nextElement();
  if(strPort.equals(p.getProperty(pName)))
  {
   if(pName=="Port Pop3")
   {
    strProtocol = new String("POP3");
    pop3 =new POP3(this);
   }
   if(pName=="Port SMTP")
   {
    strProtocol = new String("SMTP");
    smtp = new SMTP(this);
   }
  }
 }
}
public void read(String mess)
{
 if(strProtocol.equals("POP3"))
 {
```

```
    pop3.read(mess);
   }if(strProtocol.equals("SMTP"))
   {
    smtp.read(mess);
   }
  }
  public void write(String mess)
  {
   sv.ser.read(mess);
   sv.write(mess);
  }
  void control(String mess)
  {

  }
}
```

**POP3.java**

คลาส pop3 ทำหน้าที่เป็นตัวควบคุมคำสั่งที่ได้รับมาและควบคุมสถานะการทำงานการให้บริการของโปรโตคอลพ็อป3

```
package server.MsgCon.protocol;
import server.MsgCon.*;
import system.*;
import java.io.*;
public class POP3
{
 MsgCon mc;
 int state;
 int count_of_command;
 int del[]=new int[255];//for delete mail <Max = 20 mails each a loging >
 User u;
```

```
String MessageCommand;
String command[]=
//for command
{
 new String("user"), // in order = 0
 new String("pass"), // in order = 1
 new String("apop"), // in order = 2
 new String("stat"), // in order = 3
 new String("uidl"), // in order = 4
 new String("list"), // in order = 5
 new String("retr"), // in order = 6
 new String("dele"), // in order = 7
 new String("rset"), // in order = 8
 new String("top"),  // in order = 9
 new String("noop"), // in order = 10
 new String("quit")  // in order = 11

};
String respond[]=
{
 new String("+OK "),
 new String("+0K Password require for "),
 new String("+OK Mailbox rocked and ready")
};
String error[]=
{
 new String("-ERR "),                    // index 0
 new String("-ERR unknow command"),      // index 1
 new String("-ERR Invalid password"),    // index 2
 new String("-ERR Not yet Authenticated"), // index 3
 new String("-ERR Unknow user"),         // index 4
 new String("-ERR need parameter"),      // index 5
```

```
 new String("-ERR Incorrect password"),       // index 6
 new String("-ERR Incorrect number of message"),// index 7
 new String("-ERR Already authenticated")      // index 8
};
public POP3(MsgCon mc)
{
 this.mc=mc;
 count_of_command=command.length;
 mc.write(respond[0] + " The Delvelopement of the server programming for e-mail <version 0.5>");
 state = 1;
}
public void read(String msg)
{
 MessageCommand = msg; //for getParameter
 String strCommand = lexical(msg);
 //toLower becouse in table all charect is Lower
 control(strCommand.toLowerCase());
}

//open command order in table
public void control(String msg)
{
 int i=0;
 while(!msg.equals(command[i])){ i++; };
 exec(i);

}

public int exec(int orderCommand)
{
 switch(orderCommand)
 {
```

```
case 0: // for user command
 if(!(state==1))
 {
  write(error[8]);
  return 0;
 }
 user();
 break;
case 1: //for pass command
 if(state==1)
 // not yet command user
 {
  write(error[2]);
  return 0;
 }
 if(state==3)
 {
  write(error[8]);
 }
 pass();
 break;
case 2:  //for apop command
 apop();
 break;
case 3:  //for stat command
 if(!(state==3))
 {
  write(error[3]);
  return 0;
 }
 stat();
 break;
```

```
case 4: //for uidl command
 uidl();
 break;
case 5: //for list command
 if(!(state==3))
 {
  write(error[3]);
  return 0;
 }
 list();
 break;
case 6: //for retr command
 if(!(state==3))
 {
  write(error[3]);
  return 0;
 }
 retr();
 break;
case 7: //for dele command
 if(!(state==3))
 {
  write(error[3]);
  return 0;
 }
 dele();
 break;
case 8: //for rset command
 rset();
 break;
case 9: // for top command
 top();
```

```
      break;
    case 10:  // for noop command
      noop();
      break;
    case 11:  //for quit command
      quit();
      break;
    default:  // for another
      write(error[1]);
   }
   return 0;
}

// for execute in Protocol only
  private void user()
  {
    String param=getParameter();
    if(!param.equals(error[0]))
    {
      u = new User(param);
      if(u.isMember())
      {
        write(respond[1] + param);
        state=2;
      }else { write(error[4]); }
    }else { write(error[5]); }
  }
  private void pass()
  {
    String param = getParameter();
    if(!param.equals(error[0]))
    {
```

```
    if(u.isPassword(param))
    {
     write(respond[2]);
     state=3;
    }else write(error[6]);
   }else write(error[5]);
}
private void apop()
{
 write(error[1]);
}
private void stat()
{
 write(respond[0] + " " + u.stat());
}
private void uidl()
{
 write(error[1]);
}
private void list()
{

 String list[] = u.list();
 String param = getParameter();
 if(param.equals(error[0]))
 // if command "list" not parameter
 {
  write(respond[0]);
  for(int i=0;i<list.length;i++)
  {
   write(list[i]);
  }
```

```
    write(".");
   }else
   {
    int x = Integer.parseInt(param);
    if(x<(list.length+1))
    {
     write(respond[0] + "" + list[x-1]);
    }else write(error[7]);
   }
  }
  private int retr()
  {
   String param = getParameter();
   if(param.equals(error[0]))
   {
    write(error[5]);
   }else
   {
    try{
     int p = Integer.parseInt(param);
     String list[] = u.list();
     if(!(p<(list.length+1)))
     {
      write(error[7]);
      return 0;
     }
     write(respond[0]);
     File send = u.retr(p);
     FileInputStream fin = new FileInputStream(send);
     BufferedInputStream bin = new BufferedInputStream(fin);
     DataInputStream din = new DataInputStream(bin);
     String s;
```

```
    while((s = din.readLine())!=null) write(s);
    write(new String("."));
    din.close();
   }catch(Exception e){ e.printStackTrace(); }
  }
  return 0;
}
private int dele()
{
  String param = getParameter();
  if(param.equals(error[0]))
  {
   write(error[5]);
   return 0;
  }
  String list[] = u.list();
  int p = Integer.parseInt(param);
  if(!(p<(list.length+1)))
  {
   write(error[7]);
   return 0;
  }
  int index = Integer.parseInt(param);
  if(index>list.length&&index<0)
  {
   write(error[0] + "message: " + index + " not exite");
   return 0;
  }else
  {
   for(int i=0;i<del.length;i++)
   {
    if(del[i]==index) break ;
```

```
      if(del[i]==0)
      {
       del[i]=index;
       break;
      }
     }
     write(respond[0] + " message: " + index + " is marked ");
     return 0;
   }
  }
  private void rset()
  {
   for(int i=0;i<del.length;i++)
   {
    if(del[i]==0) break;
    del[i]=0;
   }
   write(respond[0]);
  }
  private void top()
  {
   write(respond[0]);
  }
  private void noop()
  {
   write(respond[0]);
  }
  private void quit()
  {
   write(respond[0]);
   u.POP3quit(del);
   write("Stop service");
```

```
    }

//_____________________________________________

//choose only command
private String lexical(String msg)
{
 int i=1;
 if(msg.length()<=4) return msg;
 for(;!(msg.charAt(i)==' ')&&i!=(msg.length()-1);i++);
 if(i>4) return new String("err");
 return new String(msg.substring(0,i));
}
private void write(String msg)
{
 mc.write(msg);
}
private String getParameter()
{
 int l = MessageCommand.length();
 for(int i=0;i<l;i++)
 {
  if(MessageCommand.charAt(i)==' ')
  {
   return MessageCommand.substring((i+1),l);
  }
 }
 return error[0];
}
}
```

**SMTP.java**

คลาส smtp ทำหน้าที่เป็นตัวควบคุมคำสั่งที่ได้รับมาและควบคุมสถานะการทำงานการให้บริการของโปรโตคอลเอสเอ็มทีพี

```
package server.MsgCon.protocol;

import java.util.*;
import java.net.*;
import server.MsgCon.protocol.*;
import server.MsgCon.*;
import system.*;
import java.io.*;
import funcDef;

public class SMTP
{
 //FOR mail message from sender
 Vector fileRecive = new Vector();

 //For reciver who Our Member (this mail server)
 Vector memberReciver = new Vector();

 Vector anotherReciver = new Vector();

 //for name of recive
 String AReciver;
 String sender = new String();
 String reciver = new String();

 String ip;
 public Vector Comm;
 Properties helo;
```

```
Properties mail;
Properties rcpt;
Properties data;
Properties rset;
Properties noop;
Properties quit;

String fulCommand;
String state_1;
String state_2;
int state;
/*
 SMTP protocol has 2 state
   1.before rcpt command
   2.after rcpt command
*/
MsgCon mc;
public SMTP(MsgCon mc)
{
 this.mc=mc;
 Comm=new Vector();
 ip=mc.sv.getIPClient();
 initCommand();
 STMPconnected();

}

void initCommand()
{
 String comm = new String("command");
 String res = new String("respond");
 String sta1 = new String("state_1");
```

```
String sta2 = new String("state_2");

helo = new Properties();
helo.put(comm,"helo");
helo.put(res,"helo");
helo.put(sta1,"enable");
helo.put(sta2,"enable");
Comm.insertElementAt(helo,Comm.size());

mail = new Properties();
mail.put(comm,"mail from:");
mail.put(res,"mail");
mail.put(sta1,"enable");
mail.put(sta2,"enable");
Comm.insertElementAt(mail,Comm.size());

rcpt = new Properties();
rcpt.put(comm,"rcpt to:");
rcpt.put(res,"rcpt");
rcpt.put(sta1,"enable");
rcpt.put(sta2,"enable");
Comm.insertElementAt(rcpt,Comm.size());

data = new Properties();
data.put(comm,"data");
data.put("respond","data");
data.put(sta1,"disable");
data.put(sta2,"enable");
Comm.insertElementAt(data,Comm.size());

rset = new Properties();
rset.put(comm,"rset");
```

```
rset.put(res,"rset");
rset.put(sta1,"enable");
rset.put(sta2,"enable");
Comm.insertElementAt(rset,Comm.size());

noop = new Properties();
noop.put(comm,"noop");
noop.put(res,"noop");
noop.put(sta1,"enable");
noop.put(sta2,"enable");
Comm.insertElementAt(noop,Comm.size());

quit = new Properties();
quit.put(comm,"quit");
quit.put(res,"puit");
quit.put(sta1,"enable");
quit.put(sta2,"enable");
Comm.insertElementAt(quit,Comm.size());

//find local address for write information in header mail
try
{
 ServerSocket socket_server=new ServerSocket(9999);
 String localIP = socket_server.getInetAddress().getLocalHost().getHostAddress();
 socket_server.close();

 //find data time
 Date d = new Date();
 ////////////////////

 fileRecive.add(new String("Received: from [" + ip + "] by " + localIP + " with SMTP;  " + d));
}catch(Exception e){}
```

```
}
public void read(String mess)
{
 // not's state recive message of e-mail
 if(state!=3)
 {
  control(mess);
 }else
 {
  reciveMsgState(mess);
 }
}
public void write(String msg)
{
 mc.write(msg);
}
public void control(String fullMsg)
{
 String fullLowMsg = fullMsg.toLowerCase();
 String strCommand = Lexical(fullLowMsg);
 String extendMsg = fullMsg.substring(strCommand.length(),
                          fullMsg.length());
 if(!strCommand.equals("unknow"))
 {
  runCommand(strCommand,extendMsg);
 }else mc.write("550 unknow command");
}
//get only command
private String Lexical(String fullmess)
{
 String refComm[] =
 {
```

```
  new String("helo"),new String("mail from:"),new String("rcpt to:"),
  new String("data"),new String("rest"),new String("noop"),
  new String("quit")
 };
 char charTmp[]=new char[20];
 for(int i=0;i<7;i++)
 {
  if(refComm[i].length()<=fullmess.length())
  {
   fullmess.getChars(0,refComm[i].length(),charTmp,0);
   String strTmp=new String(charTmp,0,refComm[i].length());
   if(strTmp.equals(refComm[i]))
   {
    return strTmp;
   }
  }
 }
 return new String("unknow");
}
private void runCommand(String strComm,String extendMsg)
{
 Properties temp;
 String comm;
 for(int i = 0;i<Comm.size();i++)
 {
  temp=(Properties)Comm.elementAt(i);
  comm=temp.getProperty("command");
  if(comm.equals(strComm))
  {
   if(chkState(temp))
   {
    exec(strComm,extendMsg,true);
```

```
      }else exec(strComm,extendMsg,false);
    }
   }
}
private boolean chkState(Properties comm)
{
 if(state==1)
 {
  if(comm.getProperty("state_1").equals("enable"))
  {
   return true;
  }else { return false; }
 }else if(state==2)
 {
  if(comm.getProperty("state_2").equals("enable"))
  {
   return true;
  }else { return false; }
 }
 return true;
}

// this part under is protocol SMTP
private void exec(String command,String extendMsg,boolean stat)
{
 if(stat)
 {
  if(command.equals("helo")) helo();
  if(command.equals("mail from:")) mail_from(extendMsg);
  if(command.equals("rcpt to:")) rcpt_to(extendMsg);
  if(command.equals("data")) data(true);
  if(command.equals("quit")) quit();
```

```
 }else if(!stat)
 {
  if(command.equals("data")) data(false);
 }
}
private void STMPconnected()
{
 mc.write("220 The Delvelopement of the server programming for e-mail <version 0.5>");
 state=1;
}
private void helo()
{
 mc.write(new String("250 Welcome... Nice to meet yor."));
}
private void mail_from(String send)
{
 sender = send;
 mc.write("250 " + send + "...Sender OK");
}
private int rcpt_to(String extend)
{
 String strEmail = funcDef.toEmailAddress(extend);
 String destDomain = funcDef.getDestHost(strEmail);
 String userName = funcDef.getName_From_Email(strEmail);
 //ArrayReciver.add(strEmail);
 state=2;
 User u =new User(userName);
 if(funcDef.isLocolHost(destDomain))
 {
  if(u.isMember())
  {
   mc.write("250 reciver " + strEmail + "... ok");
```

```
      User user =new User(userName);
      memberReciver.add(user);
      reciver.concat(new String(u.path + "/" + userName));
      return 0;
     }else
     {
      mc.write("550 unknown user ");
      return 0;
     }
   }
   mc.write("250 reciver " + extend + "... ok");
   reciver.concat(new String(u.path + "/relay" ));
   String r[] = {userName,destDomain};
   anotherReciver.add(r);
   return 0;
 }
 private void data(boolean stat)
 {
   if(stat)
   {
    mc.write("354  Enter mail,end with \".\" on a line by itself");
    state=3;
   }else
   {
    mc.write("550 Incomplete envelope informatioin");
   }
 }
 private void reciveMsg(String mess)
 {
   fileRecive.addElement(mess);
 }
 private void reciveMsgState(String mess)
```

```
{
 if(!mess.equals("."))
 {
  reciveMsg(mess);
 }else
 {
  mc.write("250 Message accepted for delivery");
  //this create file mail
  for(int i=0;i<memberReciver.size();i++)
  {
   writeMail((User)memberReciver.elementAt(i));
  }
  for(int i=0;i<anotherReciver.size();i++)
  {
   String recv[] = (String[])anotherReciver.elementAt(i);
   Socket soc = funcDef.connect(recv[1],25);
   if(soc == null)
   {
    User u = new User();
    AReciver = new String(recv[0] + "@" + recv[1]);
    writeMail(u);
   }else
   {
    new SMTPsend(this,soc,sender,recv[0] + "@" + recv[1],fileRecive);
   }
  }
  //funcDef.creatEmail(fileRecive,reciver);
  state=1;
 }
}
private void writeMail()
{
```

```
String s = funcDef.GenerateFileName("system/user/relay");
File f = new File(s);

try
{
 f.createNewFile();
 FileOutputStream fout = new FileOutputStream(f);
 BufferedOutputStream bout = new BufferedOutputStream(fout);
 PrintStream pout = new PrintStream(bout);

 // write header for transmittion to destination reciver
 writeHeader(pout);
 for(int i=0;i<fileRecive.size();i++)
 {
  String message = new String((String)fileRecive.elementAt(i));
  pout.println(message);
 }
 pout.close();
}catch(Exception e){e.printStackTrace(); }
}
private void writeMail(User u)
{
 String pathUser = u.folderUser.getAbsolutePath();
 String s = funcDef.GenerateFileName(pathUser);
 File f = new File(s);
 try
 {
  f.createNewFile();
  FileOutputStream fout = new FileOutputStream(f);
  BufferedOutputStream bout = new BufferedOutputStream(fout);
  PrintStream pout = new PrintStream(bout);
```

```
    // write header for transmittion to destination reciver
    if(!u.isMember())
    {
     writeHeader(pout);
    }
    for(int i=0;i<fileRecive.size();i++)
    {
     String message = new String((String)fileRecive.elementAt(i));
     pout.println(message);
    }
    pout.close();
   }catch(Exception e){e.printStackTrace(); }
  }
  private void writeHeader(PrintStream pout)
  {
   try{
   pout.println("From:" + sender);
   pout.println("To:<" + AReciver + ">");
   pout.println("---");
   }catch(Exception e){e.printStackTrace();}
  }
  private void quit()
  {
   mc.write("221 mail.com closing connectioin");
   mc.write("Stop service");
  }
 }
```

**SMTPsend.java**

คลาส SMTPsend เป็นคลาสที่ทำหน้าที่ส่งอีเมล์ที่ได้รับจาก User agent ส่งต่อไปยังเมล์เซิร์ฟเวอร์ ปลายทางที่ผู้ส่งได้ระบุไว้

```
package server.MsgCon.protocol;

import server.MsgCon.*;
import system.*;
import java.net.*;
import java.util.*;
import server.services.via.*;
import funcDef;

public class SMTPsend
{
 MsgCon mc;
 boolean MsgReady=false;
 String strRespond;
 int state=0;
 Socket conn;
 SMTP smtp;
 viaIn in;
 viaOut out;
 String sender;
 String reciver;
 Vector msg;
 public SMTPsend(SMTP smtp,Socket conn,String sender,String reciver ,Vector msg )
 {
  this.mc=mc;
  this.smtp=smtp;
  this.sender=sender;
  this.reciver=reciver;
```

```
  this.conn=conn;
  this.msg=msg;
  try
  {
   in = new viaIn(conn,this);
   out = new viaOut(conn);
   in.start();
  }catch(Exception e){
  e.printStackTrace();
  }
}
public void read(String msg)
{
  //strRespond = msg;
  strRespond = getCodeRespond(msg);
  control();
  System.out.println("read :" + msg);
}
public void write(String msg)
{
  out.send(msg);
  System.out.println("write :" + msg);
}
public String getCodeRespond(String msg)
{
  String code = msg.substring(0,3);
  return code;
}
public void control()
{

  switch(state)
```

```
  {
   case 0:
    helo();
    break;
   case 1:
    mail_from();
    break;
   case 2:
    rcpt_to();
    break;
   case 3:
    data();
    break;
   case 4:
    sendMsg();
    break;
   case 5:
    quit();
    break;
   default:
    end();
  }
 }
 public void helo()
 {
  if(strRespond.equals("220"))
  {
   write("helo");
   state=1;
  }else{ end(); }
 }
 public void mail_from()
```

```
{
 if(strRespond.equals("250"))
 {
  write("mail from:<" + sender + ">");
  state=2;
 }else{ end();}
}
public void rcpt_to()
{
 if(strRespond.equals("250"))
 {
  write("rcpt to:<" + reciver + ">");
  state = 3;
 }else{ end(); }
}
public void data()
{
 if(strRespond.equals("250"))
 {
  write("data");
  state = 4;
 }else { end(); }
}
public void sendMsg()
{
 if(strRespond.equals("354"))
 {
  for(int i=0;i<msg.size();i++)
  {
   write((String)msg.elementAt(i));
  }
  write(".");
```

```
      state=5;
     }else{end();};
    }
    public void quit()
    {
     if(strRespond.equals("250"))
     {
      write("quit");
      state=5;
     }
     end();
    }
    public void end()
    {
     try
     {
      in.stop();
      conn.close();
      smtp=null;
     }catch(Exception e){}
    }
}
```

**ViaIn.java**

**คลาส ViaIn ทำหน้าที่เป็นตัวที่คอยรับข้อมูลจาก User agent โดยผ่านพอร์ตที่ร้องขอมา**

```
package server.services.via;
package server.services.via;
import java.io.*;
import java.net.*;
import server.services.*;
import server.MsgCon.protocol.*;
```

```
public class viaIn extends Thread
{
 String message;
 private BufferedReader read;
 service sv;
 SMTPsend smtp;
 String reader;
 public viaIn(Socket connect,service sv)
 {
  this.sv=sv;
  reader = new String("service");
  try
    {
     read = new BufferedReader(new InputStreamReader(connect.getInputStream()));
    }catch(Exception e)
    {
      e.printStackTrace();
    }
 }
 public viaIn(Socket connect,SMTPsend smtp)
 {
  reader = new String("SMTPsend");
  this.smtp=smtp;
  try
    {
     read = new BufferedReader(new InputStreamReader(connect.getInputStream()));
    }catch(Exception e)
    {
      e.printStackTrace();
    }
 }
 public void run()
```

```
{
 try
 {
  while(true)
  {
    if( (message = read.readLine())!=null)
    {
      System.out.println(message);
      if(reader.equals("service"))
      {
       sv.read(message);
      }else
      {
       smtp.read(message);
      }
    }
  }
 }catch(IOException e)
 {
  System.out.println("massage.addElement(new String(read.readLine()));");
  e.printStackTrace();
 }
}
}
```

**ViaOut.java**

คลาส ViaOut เป็นคลาสที่ทำหน้าที่ส่งข้อมูล ไปยัง User agent เพื่อเป็นการบอกการตอบสนอง หรือแจ้ง ข้อผิดพลาดที่เกิดขึ้น

```
import java.net.*;
import java.io.*;
public class viaOut
{
 PrintWriter out;
 public viaOut(Socket connect)
 {
  try
  {
   out = new PrintWriter(
        new OutputStreamWriter(connect.getOutputStream()),
        true);
  }catch(Exception e)
  {
            e.printStackTrace();
  }
 }
 public void send(String mess)
 {
  try
  {
   out.println(mess);
  }catch(Exception e)
        {
                e.printStackTrace();
        }
 }
}
```

**Relay.java**

คลาส Relay ทำหน้าที่เป็นตัวคอยตรวจสอบว่า ยังคงมีอีเมล์ที่ยังคงต้องส่งไปยังเมล์เซิร์ฟเวอร์อื่น ๆ หรือเปล่า ถ้ายังมีให้ทำการส่งต่อไปยังเซิร์ฟเวอร์นั้น ถ้าส่งไม่ได้ ก็ให้รอการส่งตรวจสอบครั้งต่อไปจาก คลาสนี้

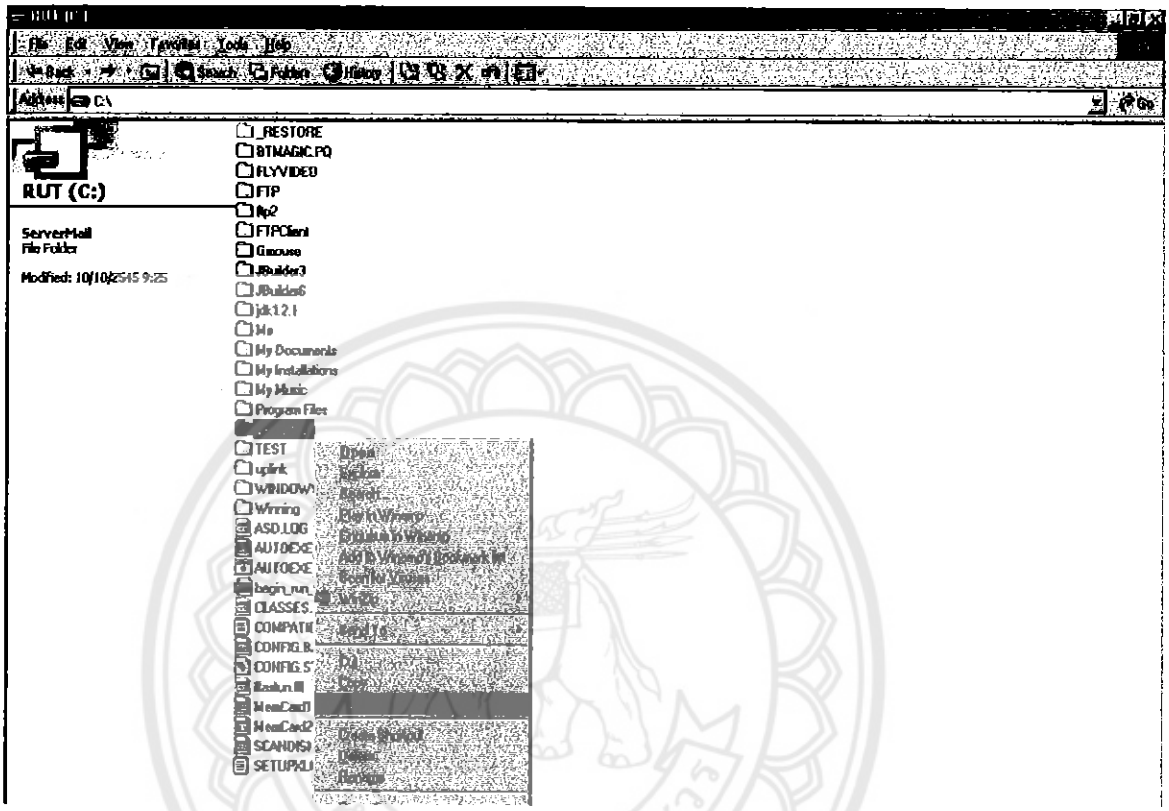
```
package server;


import system.*;
import java.io.*;
import server.MsgCon.protocol.*;
import java.util.*;
import java.net.*;
import funcDef;
import MainForm;
public class Relay extends Thread
{
 User r = new User();
 String t[]=r.list();
 static long seccond = 1000;
 static long minute = 60*seccond;
 static long hour = 60*minute;
 static long day = 24*hour;
 MainForm mf;
 public Relay(MainForm mf)
 {
  this.mf=mf;
  this.start();
 }
 public void run()
 { int i=0;
  while(true)
  {
```

```
      if(r.countMail()>0)
      {
       File mail[] =r.retrAll();
       for(int j=0;j<mail.length;j++)
       {
        new sendRelay(mail[j]);
       }
      }
      try
      {
       this.sleep(minute);
      }catch(Exception e){e.printStackTrace();}
    }
  }
}
class sendRelay
{
 DataInputStream din;
 Vector message = new Vector();
 String senderAddr;
 String reciverAddr;
 public sendRelay(File mail)
 {
  try
  {
   FileInputStream fin = new FileInputStream(mail);
   BufferedInputStream bin = new BufferedInputStream(fin);
   din = new DataInputStream(bin);
   String msg;
   while((msg=din.readLine())!=null)
   {
    message.add(msg);
```

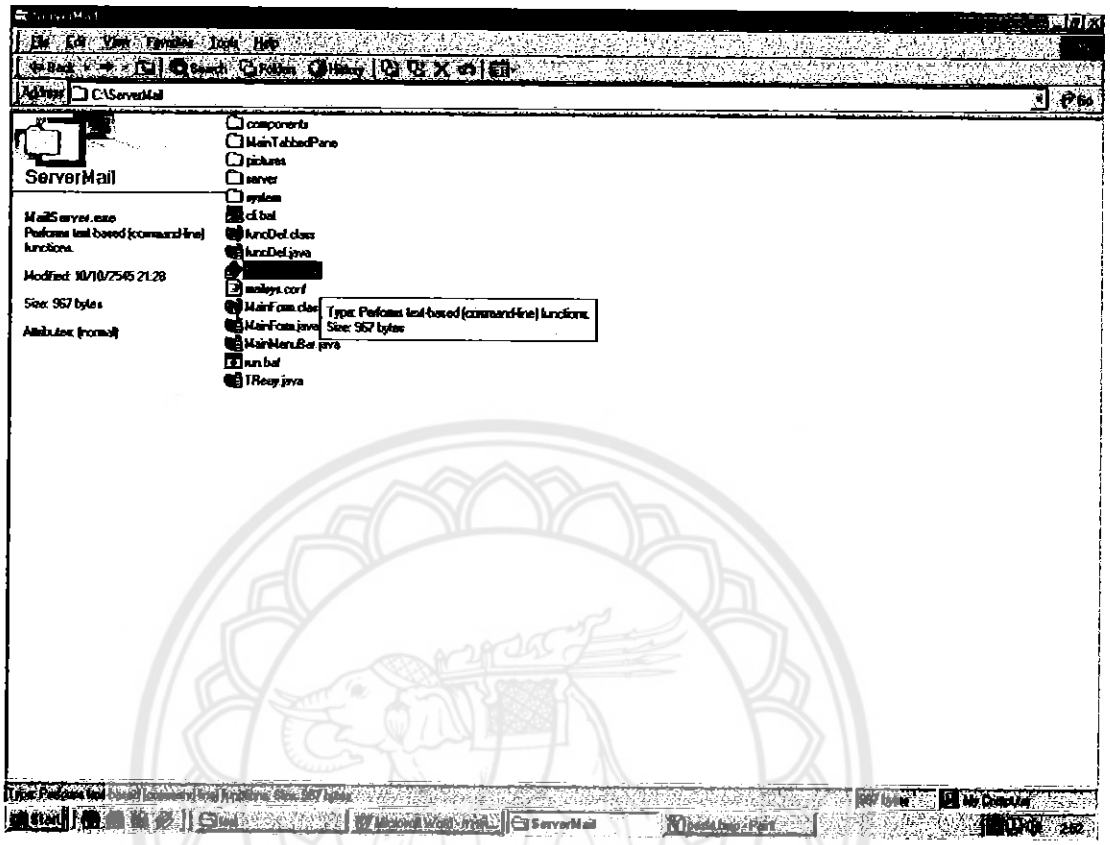
```
        }
        senderAddr=(String)message.elementAt(0);
        reciverAddr=(String)message.elementAt(1);
        for(int i=0;i<=2;i++)
        {
         //clear header in massge
         message.remove(0);
        }
        String sender = funcDef.toEmailAddress(senderAddr);
        String reciver = funcDef.toEmailAddress(reciverAddr);
        String dest = funcDef.getDestHost(reciver);
        System.out.println(dest);
        Socket soc = funcDef.connect(dest,25);
        if(soc!=null)
        {
         new SMTPsend(null,soc,sender,reciver ,message);
         din.close();
         mail.delete();
        } else
        {
         din.close();
        }
       }catch(Exception e){e.printStackTrace();}
     }
    }
```

## การติดตั้งโปรแกรมและคู่มือการใช้

1. การติดตั้งทำได้โดย copy โฟลเดอร์ ServerMail ไปที่ใดก็ได้



2. ดับเบิ้ลคลิ๊กที่ ไฟล์ run.bat หรือ MailServer.exe ( เป็น shotcut ที่สร้างขึ้น )




**หมายเหตุ** โปรแกรมที่พัฒนาขึ้นสามารถรันได้ทุก ๆ ระบบประฏิบัติการ ที่มี JAVA Viatual machine

# ประวัติผู้เขียนโครงงาน

**ชื่อ นายไพรรัตน์ โพธิ์ศรี**
**เกิดเมื่อ วันที่ 9 สิงหาคม พ.ศ. 2523**
**ที่อยู่ปัจจุบัน 3/9 หมู่ 9 ตำบลวัดพริก อำเภอเมือง จังหวัดพิษณุโลก 65230**
**ประวัติการศึกษา**

- **ชั้นอนุบาลและประถมศึกษา จากโรงเรียนวัดท่าโรงฝั่งตะวันตก อ.เมือง พิษณุโลก**
- ชั้นมัธยมศึกษาตอนต้น จากโรงเรียนพุทธชินราชพิทยา พิษณุโลก
- ชั้นมัธยมศึกษาตอนปลาย จากโรงเรียนรัตนโกสินสมโภชลาดกระบัง กรุงเทพ
- ปัจจุบัน กำลังศึกษาในระดับปริญญาตรี สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร พิษณุโลก

ชื่อ นางสาวปัทมภรณ์ เจียตระกูล
เกิดเมื่อ วันที่ 6 กรกฎาคม พ.ศ. 2523
ที่อยู่ปัจจุบัน 390 ม.11 ต.ต้อม อ.เมือง จ.พะเยา 56000
ประวัติการศึกษา

- ชั้นอนุบาลและประถมศึกษา จากโรงเรียนเทศบาล6(รัฐประชาอุทิศ) อ.เมือง จ.พะเยา
- **ชั้นมัธยมศึกษาตอนต้นและตอนปลายจากโรงเรียนพะเยาพิทยาคม**
- **ปัจจุบัน กำลังศึกษาในระดับปริญญาตรี สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร พิษณุโลก**

# ประวัติผู้เขียนโครงงาน(ต่อ)

**ชื่อ** นายกฤติพล หิรัญมาพร

**เกิด** วันที่ 7 พฤษภาคม 2524

**ที่อยู่ปัจจุบัน** 9 ม.3 ซ.นาทราย ถ.อุดรฯ-สกลฯ ต.หนองบัว

อำเภอเมือง จ.อุดรธานี 41000 โทรศัพท์ 042-323326

**ประวัติการศึกษา**

- ศึกษาขั้นอนุบาลและประถมศึกษาที่โรงเรียนเมืองใหม่วิทยา จ.หนองบัวลำภู
- ศึกษาขั้นมัธยมศึกษาตอนต้นและตอนปลายที่โรงเรียนอุดรพิทยานุกูล จ.อุดรธานี
- ปัจจุบัน กำลังศึกษาในระดับปริญญาตรี สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร พิษณุโลก