โครงงาน โปรแกรมมอนิเตอร์และวิเคราะห์เครือข่ายแบบตั้งค่าการทำงานได้

Programmable Network Monitoring and Analysis Tool

โดย

นาย ทนงศักดิ์ อิทธิศุภวรรณ 45015367

นาย พิสิฐ ศิริภาพรรณ 45015377

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

โครงงาน โปรแกรมมอนิเตอร์และวิเคราะห์เครือข่ายแบบตั้งค่าการทำงานได้

Programmable Network Monitoring and Analysis Tool

โดย

นาย ทนงศักดิ์ อิทธิศุภวรรณ 45015367

นาย พิสิฐ ศิริภาพรรณ 45015377

อาจารย์ที่ปรึกษา

อาจารย์ ธนัญชัย ตรีภาค อาจารย์ที่ปรึกษา

อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา

อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โครงงาน โปรแกรมมอนิเตอร์และวิเคราะห์เครือข่ายแบบตั้งต่าการทำงานได้
PROGRAMMABLE NETWORK MONITORING AND ANALYSIS TOOL

ผู้จัดทำ

1. นาย ทนงศักดิ์ อิทธิศุภวรรณ รหัสประจำตัวนักศึกษา 45015367
2. นาย พิสิฐ ศิริภาพรรณ รหัสประจำตัวนักศึกษา 45015377

.. อาจารย์ที่ปรึกษา
(อาจารย์ ธนัญชัย ตรีภาค)

.. อาจารย์ที่ปรึกษา
(อาจารย์ ธนา หงษ์สุวรรณ)

.. อาจารย์ที่ปรึกษา
(อาจารย์ อัครเดช วัชระภูพงษ์)

โปรแกรมมอนิเตอร์และวิเคราะห์เครือข่ายแบบตั้งค่าการทำงานได้

นาย ทนงศักดิ์ อิทธิศุภวรรรณ 45015367
นาย พิสิฐ ศิริภาพรรณ 45015377
อาจารย์ ธนัญชัย ตรีภาค อาจารย์ที่ปรึกษา
อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2547

บทคัดย่อ

เครื่องมือสำหรับการมอนิเตอร์และวิเคราะห์เครือข่ายเป็นกลไกหนึ่งที่ช่วยให้ ผู้ดูแลระบบเครือข่าย (Network Administrators) สามารถจัดการกับระบบเครือข่ายคอมพิวเตอร์ได้ในช่วงปกติและในช่วงที่เกิดปัญหา รวมทั้งวางแผนปรับปรุงเปลี่ยนแปลงระบบเครือข่ายคอมพิวเตอร์ให้เหมาะ สมกับการใช้งานอยู่ตลอดเวลา ระบบต่าง ๆ ที่ซับซ้อนย่อมเกิดข้อผิดพลาดได้ ซึ่งต้องการรับรู้ ตรวจสอบและแก้ไขให้ได้อย่างรวดเร็วที่สุดเท่าที่จะเป็นไปได้ ซึ่งเป็นการยากที่ผู้ดูแลระบบจะทำได้โดยที่ไม่มีเครื่องมือใดช่วยเลย อาจก่อให้เกิดความเสียหายที่รุนแรงได้

โปรแกรมที่ใช้ในการมอนิเตอร์และวิเคราะห์เครือข่ายในปัจจุบันมีรูปแบบจำเพาะตายตัว ไม่สามารถเปลี่ยนแปลงรูปแบบทั้งการวิเคราะห์และแสดงผลตามความต้องการแบบชั่วขณะ (ad-hoc) ได้ โครงการนี้มุ่งเน้นการสร้างโปรแกรมมอนิเตอร์และวิเคราะห์เครือข่ายที่สามารถตั้งค่าการทำงานและแสดงผลลัพธ์ตามความต้องการของผู้ดูแลระบบได้อย่างมีประสิทธิภาพและอ่อนตัว

Programmable Network Monitoring and Analysis Tool

Mr. Tanongsak Ittisupawan
Mr. Pisit Siriparphan
Mr. Thanunchai Threepak Advisor
Mr. Thana Hongsuwan Advisor
Mr. Akkradach Watcharapupong Advisor

ABSTACT

The equipment that is used to monitor and analyze the network can assist network administrators to supervise, make an appropriate scheme and properly examine their system all the time. The complex system can encounter with the failure which should be observant, inspected and make them right as soon as possible. Therefore, without supporting tools, they will venture with severe effect.

In this day, the program that is used to monitor and analysis the traffic is provided the framework of pattern which cannot be changed both its analysis pattern and its ad-hoc showing result.

This project focuses on the functions that can be configured and efficiently display the effect by the administrator requirement.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยดี เนื่องจากได้รับการแนะนำสนับสนุน และให้คำปรึกษาเป็นอย่างดีจาก อาจารย์ธนัญชัย ตรีภาค อาจารย์อัครเดช วัชระภูพงษ์ และอาจารย์ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษาปริญญานิพนธ์ ซึ่งต้องขอขอบพระคุณเป็นอย่างสูง รวมทั้งอาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ให้การอบรมสั่งสอนวิชาความรู้ แก่คณะผู้จัดทำมาโดยตลอดและขอขอบพระคุณเป็นอย่างสูงสำหรับบุคคลที่สำคัญที่สุดที่ทำให้คณะผู้จัดทำมีวันนี้ คือ บิดา มารดา ผู้เป็นที่เคารพรักยิ่งของคณะผู้จัดทำ ซึ่งท่านให้การอบรมสั่งสอน เลี้ยงดู และให้โอกาสในการศึกษาอย่างเต็มที่ จึงขอกราบขอบพระคุณมา ณ ที่นี้

สุดท้ายนี้ ขอขอบพระคุณผู้ดูแลระบบคอมพิวเตอร์ ภาควิชาวิศวกรรมศาสตร์ และสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่อำนวยความสะดวกในการใช้งานเครือข่าย และขอขอบคุณเพื่อนๆ ที่ให้ข้อคิดเป็น และเป็นกำลังใจให้เสมอมา

ทนงศักดิ์ อิทธิศุภวรรรณ

พิสิฐ ศิริภาพรรณ

สารบัญ

หน้าที่

สารบัญรูปภาพ

สารบัญตาราง

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เนื่องจากโปรแกรมที่ใช้ในการมอนิเตอร์และวิเคราะห์เครือข่ายในปัจจุบันมีการทำงาน โดยการดึงข้อมูลจากเครือข่ายมาแสดงผล หรือนำมาวิเคราะห์หาผลลัพธ์โดยฟังก์ชั่นต่าง ๆ ก่อนแล้วจึงนำมาแสดงผล ซึ่งอาจอยู่ในรูปของกราฟหรือตารางแต่การทำงานของโปรแกรมดังกล่าวมักมีรูปแบบจำเพาะตายตัว ไม่สามารถเปลี่ยนแปลงรูปแบบทั้งการวิเคราะห์และแสดงผลตามความต้องการแบบชั่วขณะได้

โครงการนี้มุ่งเน้นการสร้างโปรแกรมมอนิเตอร์และวิเคราะห์เครือข่ายที่สามารถตั้งค่า การทำงานและแสดงผลลัพธ์ตามความต้องการของผู้ดูแลระบบได้อย่างมีประสิทธิภาพและอ่อนตัว

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการทำงานของโปรแกรมมอนิเตอร์เครือข่ายที่มีอยู่ในปัจจุบัน พร้อมทั้งปัญหาที่เกิดขึ้นในการใช้งานจริง
2. เพื่อสร้างโปรแกรมมอนิเตอร์และวิเคราะห์เครือข่ายที่ตอบสนองการทำงานของผู้ดูแลระบบได้อย่างหลากหลาย

1.3 ขอบเขตของโครงงาน

1. ศึกษาการออกแบบและพัฒนาโปรแกรมมอนิเตอร์และวิเคราะห์เครือข่ายที่สามารถตั้งค่าการทำงานได้ บนระบบปฏิบัติการวินโดว์
2. โปรโตคอลที่โปรแกรมสามารถวิเคราะห์ได้ คือ
   - TCP/IP
   - ICMP
   - UDP
   - IP
3. สามารถตั้งค่าโดยการตรวจวัด attribute ต่างๆ ใน TCP ,UDP ,ICMP ,IP header และโปรโตคอล ต่าง ๆ ที่เกี่ยวข้อง
4. ไม่เหมาะสำหรับเครือข่ายขนาดใหญ่และเครือข่าย Traffic สูงมาก ๆ เช่น ISP
5. ได้ Software ที่ต้องการและใช้งานง่าย

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษา Algorithm ต่าง ๆ
2. ศึกษาการทำงานที่เกี่ยวข้อง และ กฎเกณฑ์ต่าง ๆ ของ Snort
3. ศึกษาเกี่ยวกับการแสดงผลแบบกราฟ และ Report
4. ศึกษาการเขียนโปรแกรมบน Windows และ Protocol ต่าง ๆ
5. ศึกษาการเขียนโปรแกรมติดต่อกับ Winpcap และ Interface Card
6. ทดลองเขียนโปรแกรม แสดงผลแบบ กราฟ และ Report
7. ทดลองเขียนโปรแกรม ติดต่อกับ Interface Card และ Winpcap
8. ศึกษาการออกแบบและการสร้างส่วนจัดการการตั้งค่า Configuratiom Management เช่น Parser ได้
9. เขียนโปรแกรมที่ใช้ในการมอนิเตอร์และวิเคราะห์เครือข่ายที่สามารถตั้งค่าการทำงานได้ และใช้งานได้อย่างถูกต้อง
10. พัฒนาโปรแกรมให้ใช้ได้อย่างมีประสิทธิภาพ

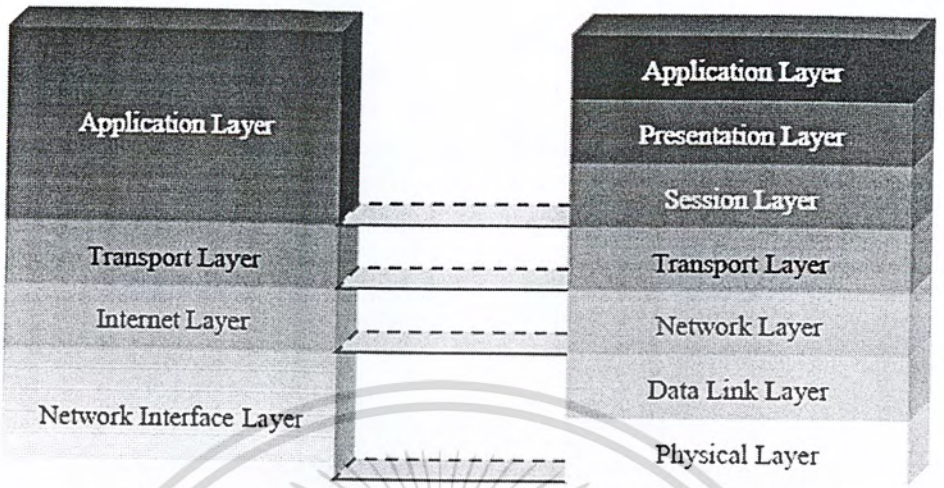
บทที่ 2
โพรโตคอลทีซีพี/ไอพี

2.1 ความเป็นมาของโพรโตคอลทีซีพี/ไอพี

ทีซีพี/ไอพี เป็นมาตรฐานการรับส่งข้อมูลระหว่างคอมพิวเตอร์สองระบบที่มีขึ้นเมื่อ กระทรวงกลาโหมสหรัฐฯ หรือ Department Of Defense (DOD) ทำการทดลองในปี ค.ศ.1969 เชื่อมโยงคอมพิวเตอร์ทางทหารของแต่ละหน่วย ซึ่งเป็นคอมพิวเตอร์ต่างชนิดกันให้สามารถติดต่อรับส่งข้อมูลกันได้ โครงการนี้มีชื่อว่า Advanced Research Projects Agency Network หรือ ARPANET ซอฟต์แวร์ที่ใช้ควบคุมการรับส่งข้อมูลของ ARPANET ประกอบด้วยส่วนหลักๆ 2 ส่วน คือ ทีซีพี (Transmission Control Protocol หรือ TCP) และไอพี (Internet Protocol หรือ IP) ซึ่ง ทีซีพี มีหน้าที่ตรวจสอบการรับ
ส่งข้อมูลระหว่างคอมพิวเตอร์ผู้รับและผู้ส่ง ให้ได้รับข้อมูลถูกต้องครบถ้วน ส่วนไอพีจะมีหน้าที่เลือกเส้นทางที่ใช้รับส่งข้อมูลผ่านระบบเครือข่าย และตรวจสอบที่แอดเดรสของผู้รับเรียกว่าไอพีแอดเดรส (IP Address) ต่อมาในปี ค.ศ.1983 ทีซีพี/ไอพี ถูกกำหนดให้ เป็นมาตรฐานการรับส่งข้อมู ลของกระทรวงกลาโหมสหรัฐฯ และได้รวมเป็นส่วนหนึ่งของระบบปฏิบัติการยูนิกซ์ ส่งผลให้มีการใช้งานกันอย่างกว้างขวาง ในปัจจุบันใช้งานอยู่ในแทบทุกเครือข่าย ไม่ว่าจะเป็นเครือข่ายเฉพาะที่หรือเครือข่ายในบริเวณกว้างทีซีพี/ไอพี เชื่อมกลุ่มเครือข่ายย่อยเข้าด้วยกันเป็นเครือข่ายขนาดใหญ่ หรือ อินเทอร์เน็ต (Internet)

2.2 การเชื่อมต่อของโพรโตคอลทีซีพี/ไอพี (TCP/IP Linking)

ทีซีพี/ไอพี (TCP/IP หรือ Transmission Control Protocol/Internet Protocol) เป็นโพรโตคอลในการสื่อสารในระบบอินเทอร์เน็ตและอินทราเน็ต การทำงานของทีซีพี/ไอพีสามารถเปรียบเทียบกับโมเดลอ้างอิงโอเอสไอ (Open System Interconnection Reference Model: OSI) ตามมาตรฐานไอเอสโอ (International Organization for Standardization: ISO) ได้ดังรูปที่ 2-1

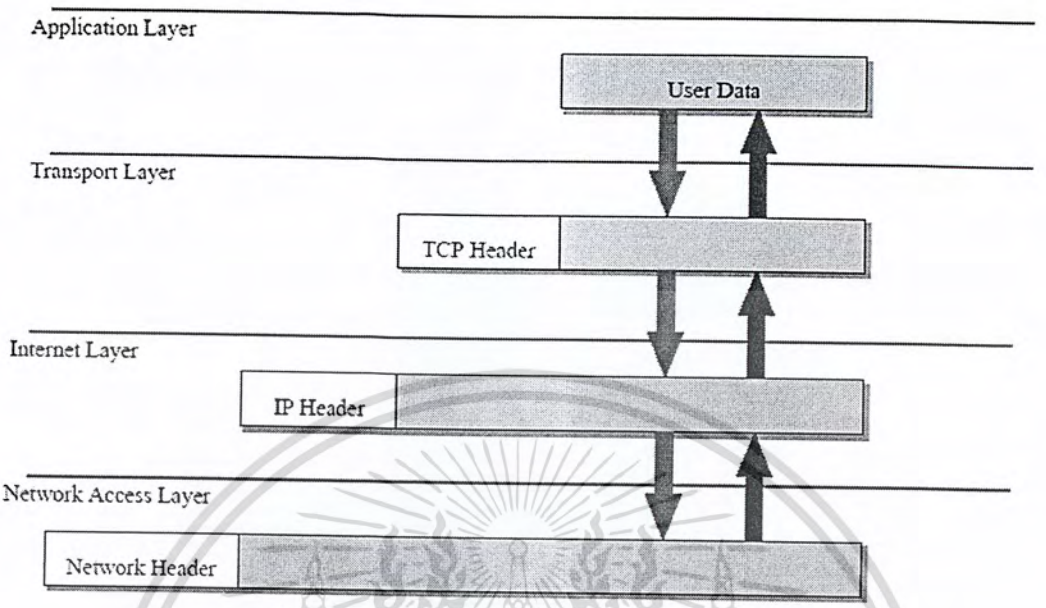


รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของโอเอสไอกับเลเยอร์ของทีซีพี/ไอพี

ในแต่ละระดับชั้นของทีซีพี/ไอพีมีการทำงานที่แตกต่างกัน ตั้งแต่การติดต่อกับแอพพลิเคชั่นจนกระทั่งแปลงเป็นสัญญาณส่งไปตามสายสัญญาณ ซึ่งการทำงานในแต่ละระดับชั้นของทีซีพี/ไอพี มีดังตารางที่ 2-1

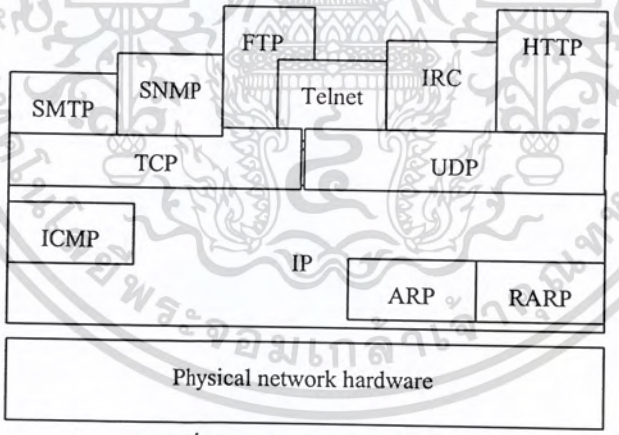
ชื่อระดับชั้น	หน้าที่
1. ชั้นแอพพลิเคชัน (Application Layer)	ชั้นนี้รองรับการทำงานของแอพพลิเคชันต่างๆ ที่ทำงานเป็นโพรเซสอยู่ในเครื่องต้นทางและปลายทาง โดยจัดการเชื่อมต่อระหว่างโพรเซส หรือแอพพลิเคชันที่อยู่ต่างเครื่องกัน โดยการทำงานของแอพพลิเคชันต่างๆมีการติดต่อกันตามแต่ละ โพรโตคอลเฉพาะแล้วแต่แอพพลิเคชันที่ใช้งาน ซึ่งจะขอบริการจากชั้นทรานสปอร์ตอีกทีหนึ่ง
2. ชั้นทรานสปอร์ต (Transport Layer)	มีการสร้างการเชื่อมต่อกันระหว่างแอพพลิเคชันแบบ end-to-end โดยจุดที่เชื่อมต่อกันเพื่อรับส่งข้อมูลนี้เรียกว่า พอร์ต (port) หรือซ็อกเก็ต (Socket) ในชั้นนี้มีบริการหลักอยู่ 2 แบบ คือ Connection Oriented โดยเรียกผ่านโพรโตคอลทีซีพี (TCP: Transmission Control Protocol) และ Connectionless ซึ่งเรียกผ่านโพรโตคอลยูดีพี (UDP: User Datagram Protocol) ซึ่งกล่าวถึงในหัวข้อถัดไป
3. ชั้นอินเทอร์เน็ต (Internet Layer)	ชั้นนี้มีหน้านี้ส่งผ่านข้อมูลระหว่างเครือข่าย โดยมีโพรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใดๆ ในอินเทอร์เน็ต คือ ไอพี (Internet Protocol: IP) ซึ่งกล่าวถึงในหัวข้อถัดไป นอกจากนี้ในชั้นนี้ยังมีโพรโตคอลทำงานอยู่ด้วยอีก 2 ชนิด คือ ไอซีเอ็มพี (Internet Control Message Protocol: ICMP) และเออาร์พี (Address Resolution Protocol: ARP)
4. ชั้นเน็ตเวิร์กอินเตอร์เฟส (Network Interface Layer)	ทำหน้าที่ในการแปลงข้อมูลให้อยู่ในรูปที่เหมาะสมกับเครือข่ายแต่ละแบบ ซึ่งแตกต่างกันออกไป และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่าย

ตารางที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของโอเอสไอกับเลเยอร์ของทีซีพี/ไอพี



รูปที่ 2-2 แสดงการข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี

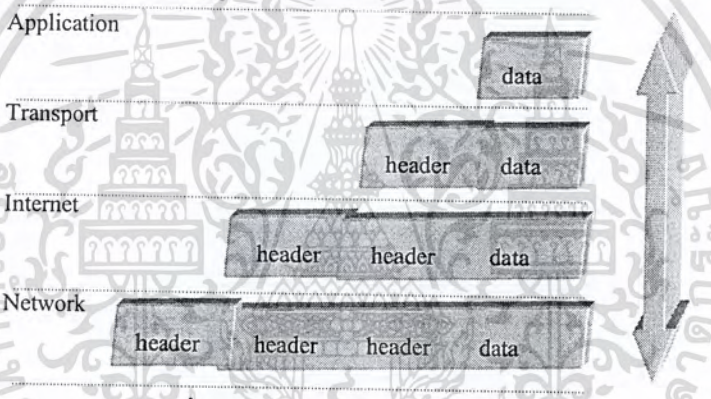
2.3 ชุดโพรโตคอลทีซีพี/ไอพี (TCP/IP Protocol suite)



รูปที่ 2-3 สถาปัตยกรรมทีซีพี/ไอพี

ประกอบด้วยเลเยอร์หลายชั้นเช่นเดียวกับแบบอ้างอิงโอเอสไอดังรูป 2-3 ถ้าพิจารณาจากหน้าที่ของแต่ละเลเยอร์แล้ว 4 เลเยอร์ล่างของทีซีพี/ไอพีสามารถนำมาเปรียบเทียบกับ 4 เลเยอร์ล่างของแบบอ้างอิงโอเอสไอได้ โดยเลเยอร์ 1 และ 2 เป็นเลเยอร์ที่ใช้ร่วมกันได้(compatible) เพราะโอเอสไอกำหนดระบบตัวกลางหลายระบบในเลเยอร์ดังกล่าว และทีซีพี/ไอพีก็ถูกออกแบบให้ใช้ตัวกลางใดก็ได้ (medium independent) เมื่อพิจารณาเลเยอร์ 3 และ 4 คืออินเทอร์เน็ตและทรานสปอร์ตของทีซีพี/ไอพี และเน็ตเวิร์คและทรานสปอร์ตของ

โอเอสไอ จะเห็นได้ว่าโอเอสไอมีทางเลือกมากมายที่สามารถใช้ในเลเยอร์ทั้งสองนี้ได้ และบางตัวก็ทำหน้าที่ในลักษณะคล้ายคลึงกับในทีซีพี/ไอพี ข้อแตกต่างสำคัญระหว่างทีซีพี/ไอพีกับโอเอสไอคือ เลเยอร์แอพพลิเคชั่นของทีซีพี/ไอพีซึ่งในโอเอสไอแล้วจะเท่ากับ 3 เลเยอร์บน ถึงแม้จะมีความแตกต่างในการแบ่งเป็นเลเยอร์อยู่บ้าง แต่ลักษณะในการส่งข้อมูลของทีซีพี/ไอพีจะเหมือนกับของโอเอสไอคือข้อมูลจะถูกส่งลงมาจากสแต็กหรือส่งขึ้นไปบนสแต็ก ขณะที่ข้อมูลถูกส่งลงมาแต่ละเลเยอร์ในสแต็กก็จะเพิ่มข้อมูลควบคุม (control information) เข้าไปเพื่อจะแน่ใจได้ว่าการส่งเกิดขึ้นอย่างเหมาะสม ข้อมูลควบคุมนี้เรียกว่าเฮดเดอร์ เพราะมันถูกใส่ไว้หน้าข้อมูลที่ถูกส่ง แต่ละเลเยอร์ปฏิบัติกับข้อมูลทั้งหมดที่มันได้รับมาจากเลเยอร์ที่สูงกว่าเหมือนเป็นข้อมูลจริง ๆ และเพิ่มเฮดเดอร์ของมันเองไว้ข้างหน้าข้อมูลทั้งหมดนั้น การทำดังกล่าวนี้เรียกว่าเอ็นแคปซูเลชั่น (encapsulation) ดังรูป 2-4 เมื่อผู้รับได้รับข้อมูลก็จะทำตรงกันข้ามกับที่กล่าวมาคือแต่ละเลเยอร์จะนำเฮดเดอร์ออกมาก่อนที่จะส่งข้อมูลขึ้นไปยังเลเยอร์ที่สูงกว่า ข้อมูลก็จะไหลขึ้นไปบนสแต็ก ข้อมูลที่ได้รับก็จะถูกแปลความหมายทั้งเฮดเดอร์และข้อมูล



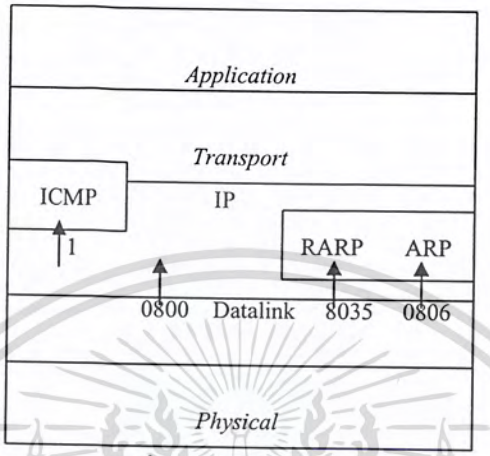
รูปที่ 2-4 การส่งผ่านข้อมูลระหว่างเลเยอร์

เลเยอร์ต่าง ๆ และโพรโตคอลที่เกี่ยวข้องในโพรโตคอลชุดทีซีพี/ไอพีดังนี้

2.3.1 เลเยอร์เน็ตเวิร์กของทีซีพี/ไอพี

มีโพรโตคอลไอพี (Internet Protocol: IP) ทำหน้าที่พื้นฐานในการส่งโพรโตคอลชั้นสูงกว่าของทีซีพี/ไอพีไปบนเครือข่ายฟิสิคัลทั้งหมด ซึ่งทำให้โพรโตคอลในเลเยอร์ที่สูงกว่าไม่จำเป็นต้องรู้อะไรเกี่ยวกับความสามารถของตัวกลางเลย และยังทำให้ผู้พัฒนาแอพพลิเคชั่นที่เขียนโปรแกรมเหนือเลเยอร์ทรานสปอร์ตทำงานได้ง่ายขึ้นด้วยเหตุผลเดียวกัน นอกจากนี้โพรโตคอลไอพียังเกี่ยวข้องกับการส่งข้อมูลไปยังเครื่องและเครือข่ายที่ต้องการอย่างถูกต้องหรือการเร้าท์เส้นทางนั่นเอง ไอพีเป็นบริการแบบไม่ต้องการก่อตั้งการเชื่อมต่อก่อน (connectionless หรือ connectionless datagram service) เพราะไม่มีการเรียก(call) หรือก่อตั้งวงจรเสมือนก่อนที่จะเริ่มส่งข้อมูล เนื่องจากแต่ละดาต้าแกรมมีข้อมูลทั้งหมดที่จำเป็นต้องใช้ในการเร้าท์เส้นทางอยู่แล้วและระหว่างโหนด 2 โหนดก็ไม่มีเส้นทางที่เฉพาะเจาะจงซึ่งทำให้ง่ายในการเร้าท์ใหม่แม้จะเสียเวลาในการ

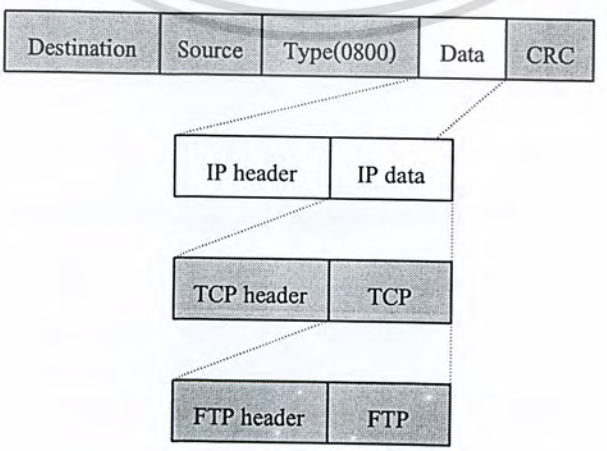
สวิตชิ่งเล็กน้อย เมื่อเครือข่ายเกิดข้อผิดพลาด ส่วนแอดเดรสปลายทางที่ใช้มีทั้งคนเดียว (unique) เป็นกลุ่ม (multicast) หรือทุกคน (broadcast)



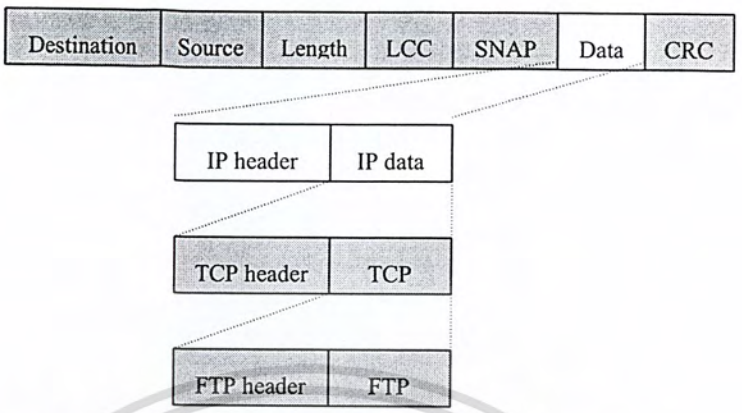
รูปที่ 2-5 สถาปัตยกรรมไอพี

นอกจากไอพีแล้วยังมีโพรโตคอลอื่นในเลเยอร์นี้อีกคือ โพรโตคอล เออาร์พี (Address Resolution Protocol: ARP), โพรโตคอลอาร์เออาร์พี (Reverse Address Resolution Protocol: RARP), โพรโตคอลไอซีเอ็มพี (Internet Control Message Protocol: ICMP) ดังรูป 2-4 โพรโตคอลเออาร์พีและอาร์เออาร์พีแสดงที่ตำแหน่งล่างของชั้นไอพีเพราะโพรโตคอล 2 ตัวนี้ไม่ได้ใช้ไอพีและเป็นที่รู้จักโดยชั้นดาต้าลิงค์ที่สนับสนุน เหมือนเป็นโพรโตคอลที่แยกออกมาต่างหาก ไอซีเอ็มพีแสดงไว้ตำแหน่งบนของชั้นไอพีเพราะมันถูกส่งข้ามเครือข่ายโดยอยู่ในดาต้าแกรมไอพี ชั้นไอพีรู้ว่าเป็นดาต้าแกรมไอซีเอ็มพีโดยค่าโพรโตคอลที่เท่ากับ 1 ทั้งฟิลด์เฮดเดอร์และฟิลด์ข้อมูลของดาต้าแกรมไอพีจะกลายเป็นฟิลด์ข้อมูลของเฟรมในชั้นดาต้าลิงค์ ลักษณะเช่นนี้เรียกว่าการเอ็นแคปซูเลชั่นดังได้กล่าวแล้ว หรือบางครั้งก็เรียกการเอ็นเวลล็อปปิ้ง(enveloping) ซึ่งฟิลด์ข้อมูลของดาต้าแกรมไอพีเองก็บรรจุเฮดเดอร์ของโพรโตคอลในชั้นที่สูงกว่าเช่นเดียวกันดังรูป 2-6 ซึ่งแสดงการเอ็นแคปซูเลชั่นในเฟรมอีเธอร์เน็ตทูและเฟรม SNAP

(a)

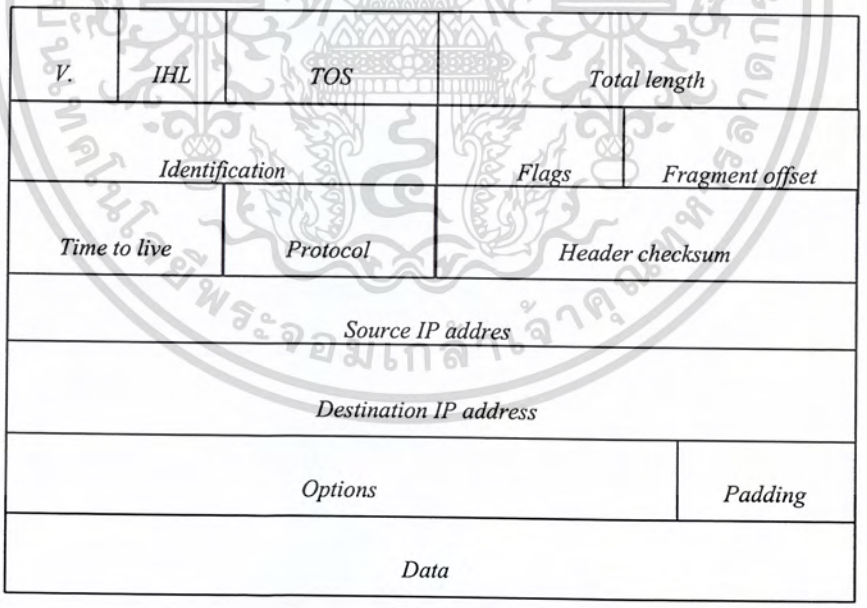


(b)



รูปที่ 2-6 แสดงไอพีใน (a) เฟรมอีเธอร์เน็ตทู (b) เฟรม SNAP

ลักษณะดาต้าแกรมไอพีเวอร์ชั่น 4 เป็นดังรูป 2-7 ซึ่งแสดงในลักษณะกว้าง 32 บิต เมื่อดาต้าแกรมนี้ถูกส่งไปบนเครือข่ายลำดับการส่งจะเป็น จากซ้ายบนไปขวาล่างซึ่งเรียกว่าลำดับไบต์ของเครือข่าย (network byte order) และตัวเลขในทีซีพี/ไอพีจะถูกส่งโดยให้บิตที่สำคัญสูงสุด (most significant octet) ไปก่อน แต่ละฟิลด์ในรูป 2-7 สามารถอธิบายได้ดังต่อไปนี้



รูปที่ 2-7 รูปแบบของดาต้าแกรมไอพีเวอร์ชั่น 4

1. **Version**	มีขนาด 4 บิต แสดงถึงเวอร์ชั่นของโพรโตคอลไอพี ขณะนี้คือ เวอร์ชั่น 4
2. **Internet Header Length**	มีขนาด 4 บิต แสดงถึงความยาวของเฮดเดอร์ หน่วยเป็น 32 บิตเวิร์ด ทำให้พบจุดเริ่มต้นของข้อมูลได้ง่ายถ้ามีการใช้ฟิลด์ Option แต่ปกติจะมีค่าเป็น 5 คือไม่มีการใช้ฟิลด์ Option
3. **Type of Service**	มีขนาด 8 บิต ประกอบไปด้วยแฟล็กที่ใช้สำหรับทีโอเอส (TOS) และลำดับความสำคัญ โดย 3 บิตแรกใช้บ่งถึงลำดับความสำคัญ 8 ระดับซึ่งทำให้โหนดของไอพีรู้ว่าดาต้าแกรมใดมีความสำคัญมากกว่าดาต้าแกรมอื่น แต่เราท์เตอร์บางตัวก็ไม่สนใจแฟล็กนี้
แฟล็กดี (D-flag)	เป็นการร้องขอการเชื่อมต่อที่มีการเสียเวลา (delay) ต่ำ
แฟล็กที (T-flag)	บอกถึงความต้องการทรูพุท (throughput) สูง
แฟล็กอาร์ (R-flag)	บอกถึงความต้องการความเชื่อถือ (reliability) สูง หมายถึงความน่าจะเป็นในการละทิ้ง (discard) ดาต้าแกรมมีต่ำกว่า
แฟล็กซี (C-flag)	บอกถึงความต้องการเสียค่าใช้จ่ายที่ต่ำกว่า บิตสุดท้ายที่ไม่ใช้
4. Total Length	มีขนาด 16 บิต เป็นการวัดทั้งเฮดเดอร์และข้อมูลในหน่วยออคเต็ด (octets) ซึ่งทำให้คำนวณขนาดข้อมูลโดยคิดจากฟิลด์ Total Length และฟิลด์ IHL ได้ จะเห็นได้ว่าฟิลด์นี้มีขนาด 16 บิตซึ่งหมายความว่าขนาดดาต้าแกรมที่ใหญ่ที่สุดมีขนาดเป็น 65,535 ออคเต็ดซึ่งใหญ่กว่าที่ครือข่ายฟิสิคัลสนับสนุนมาก ถ้าดาต้าแกรมถูกแบ่งย่อย (fragment) ค่าในฟิลด์นี้คือค่าใหม่ ไม่ใช่ค่าเก่าของขนาดดาต้าแกรม
5. Identification	มีขนาด 16 บิต บ่งบอกถึงการแบ่งย่อยทั้งหมดของดาต้าแกรมมี ลักษณะเฉพาะของใครของมัน (unique) สำหรับแต่ละดาต้าแกรมใหม่ ที่ถูกส่งโดยโฮส ฟิลด์นี้ไม่ใช่หมายถึงหมายเลขลำดับ (sequence number) เพราะไอพีเป็นบริการแบบไม่ต้อง ก่อตั้งการติดต่อก่อนส่งข้อมูล แต่เป็นเพราะไอพีสนับสนุนบริการการเชื่อมต่อของเลเยอร์ทรานสปอร์ตได้หลายแบบ
6. Flags	มีขนาด 3 บิต ใช้ในการควบคุมการแบ่งย่อย ถ้าบิตลำดับต่ำมีค่าเป็น 0 หมายถึงเป็นส่วนสุดท้ายของดาต้าแกรมที่ถูกแบ่งย่อย บางครั้งจึงเรียกบิตนี้ว่า More Flag หรือบิต MF บิตกลางใช้บ่งถึงว่าดาต้าแกรมนี้ห้ามแบ่งย่อยจึงเรียกว่า Do not fragment หรือบิต DF บิตลำดับสูงไม่ถูกใช้

7. Fragment offset มีขนาด 13 บิต ฟิลด์นี้ใช้ร่วมกับดาต้าแกรมที่ถูกแบ่งย่อยเพื่อบอกถึงตำแหน่งของข้อมูลในดาต้าแกรมเดิม วัดในหน่วย 8 ออคเต็ด ดังนั้นการแบ่งย่อยดาต้าแกรมจึงต้องทำให้หน่วยนี้

8. Time to live มีขนาด 8 บิต ฟิลด์นี้ถูกเซ็ตโดยผู้ส่งดาต้าแกรมและจะถูกลดค่า โดยเราท์เตอร์เมื่อดาต้าแกรมผ่านมัน ถ้าฟิลด์ TTL ถูกลดค่าจนเป็น 0 ดาต้าแกรมนั้นจะถูกละทิ้งเพื่อป้องกันไม่ให้ดาต้าแกรมถูกเร้าท์เป็นลูป (loop) ตลอดไป

9. Protocol มีขนาด 8 บิต บ่งถึงว่าดาต้าแกรมนั้นบรรจุโพรโตคอลใดของเลเยอร์ทรานสปอร์ต ค่าปกติคือ

17	ยูดีพี (UDP)
6	ทีซีพี (TCP)
1	ไอซีเอ็มพี (ICMP)
7	อีจีพี (EGP)
89	โอเอสพีเอฟ (OSPF)

10. Header checksum มีขนาด 16 บิต ป้องกันเฉพาะเฮดเดอร์ไม่รวมข้อมูล เพราะจะต้องคำนวณใหม่ทุกครั้งที่ผ่านเราท์เตอร์ เพราะค่าฟิลด์ TTL Flags และ Fragment offset เปลี่ยนไป ถ้าคำนวณข้อมูลด้วยจะทำให้เสียเวลามากขึ้น

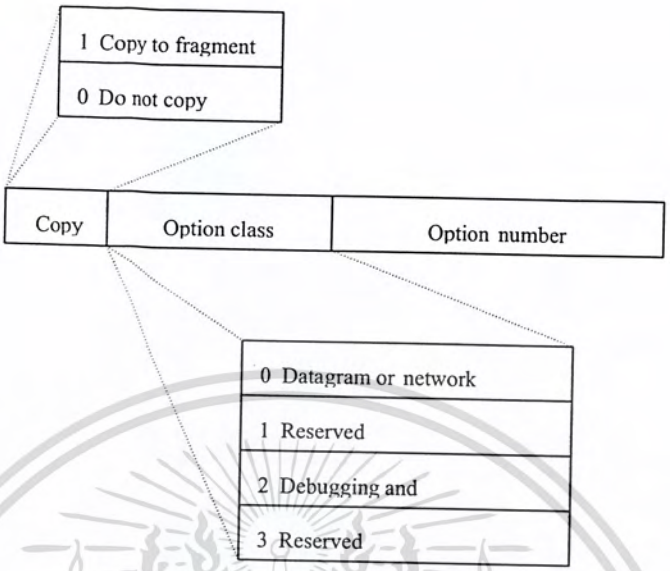
11. Source IP address มีขนาด 32 บิต

12. Destination IP address มีขนาด 32 บิต

13. Data มีขนาดไม่แน่นอน ซึ่งฟิลด์นี้จะรวมเฮดเดอร์ของโพรโตคอลในเลเยอร์ที่สูงกว่าไว้กับข้อมูลจริง ๆ ด้วย

14. Padding มีขนาดไม่แน่นอน ค่าของฟิลด์นี้จะแทนด้วย 0 ใช้เพื่อต่อเฮดเดอร์ให้ครบ 32 บิตเวิร์ด ซึ่งทำให้ IHL บอกถึงจุดเริ่มต้นของข้อมูลได้ถูกต้องเมื่อมีการใช้ฟิลด์ Options ซึ่งความยาวไม่คงที่

15. Options สนับสนุนการดีบัก(debugging) การวัด(measurement) และความปลอดภัย(security) ซึ่งสามารถมีหลาย ออปชั่นได้ในดาต้าแกรมเดียวดังรูป 2-8 ซึ่งฟิลด์นี้ประกอบด้วย



รูปที่ 2-8 ออปชั่นในการวัดและความปลอดภัย

15.1 Copy มีขนาด 1 บิต ใช้ตัดสินใจว่าออปชั่นจะอยู่ในทุกส่วนคาต้าแกรมที่ถูกแบ่งหรือไม่ ถ้ามีค่าเป็น 0 หมายถึงออปชั่นจะปรากฏในส่วนย่อยแรก (fragment) เท่านั้น ตัวอย่างของออปชั่นที่จำเป็นต้องคัดลอก(copy) ให้แก่ทุกส่วนย่อยคือออปชั่นเกี่ยวกับความปลอดภัย

15.2 Option class มีขนาด 2 บิต บอกถึงคลาสของออปชั่น ซึ่งออปชั่นไทม์แสตมป์ (Time stamp option) มีคลาสเป็น 2 นอกจากนั้นคลาสปกติจะเป็น 0

15.3 Option numbers มีขนาด 5 บิต

15.3.1 Security: คือ Option 2 มีการกำหนดระดับของคาต้าแกรมจากธรรมดาไปจนถึงคาต้าแกรมที่เป็นความลับมาก ซึ่งช่วยให้เราท์เตอร์รู้ว่าคาต้าแกรมใดบรรจุข้อมูลที่สำคัญและป้องกันข้อมูลเหล่านั้น ไม่ให้ออกจากสิ่งแวดล้อมที่ปลอดภัย

15.3.2 Time stamp: คือ Option 4 ทำให้คาต้าแกรมที่ถูกส่งไปในเครือข่าย สามารถรวบรวมไทม์แสตมป์จากแต่ละเราท์เตอร์ที่มันผ่านซึ่งเราสามารถนำ สิ่งที่ได้นี้มาใช้ประเมินการเสียเวลา และความเปลี่ยนแปลงใน เครือข่าย ของเราท์เตอร์ได้

15.3.3 Loose source route: คือ Option 3 ในการกำหนดค่าของเราท์เตอร์ เพื่อให้ดาต้าแกรมผ่านตามไอพีแอดเดรสของเราท์เตอร์ ออปชั่นนี้จะอนุญาตให้ใช้เราท์เตอร์อื่นได้ในระหว่างลิสต์ของเราท์เตอร์ ที่ถูกกำหนด

15.3.4 Record route: คือ Option 7 ทำให้แต่ละเราท์เตอร์ใส่ไอพีแอดเดรส ของมันในฟิลด์ Option ของดาต้าแกรมเมื่อดาต้าแกรมเดินทางผ่านเครือข่าย ซึ่งทำให้ค้นหาทางที่ดาต้าแกรมใช้ในการไปถึงโฮสหรือเราท์เตอร์ใดได้

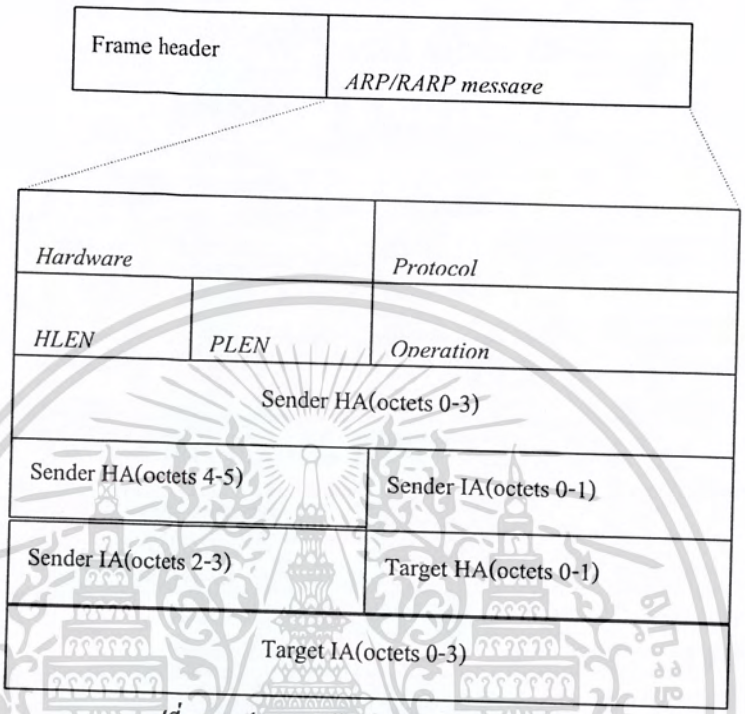
15.3.5 Strict source route: Option 9 คล้ายกับ loose source route ยกเว้นว่า เฉพาะเราท์เตอร์ที่กำหนดในลิสต์เท่านั้นที่สามารถใช้ได้

โพรโตคอลเออาร์พี(Address Resolution Protocol: ARP)

การ์ดแลนส่งและรับแฟรมโดยใช้แมคแอดเดรส(MAC address) แต่ทีซีพี/ไอพีใช้ไอพีแอดเดรสที่กำหนดโดยผู้ดูแลระบบเครือข่าย ณ เวลาติดตั้ง ซึ่งไม่มีความสัมพันธ์โดยตรงกับแมคแอดเดรส การสื่อสารแบบปลายถึงปลาย(end-to-end) ใช้ไอพีแอดเดรส แต่แบบฮ็อพถึงฮ็อพ(hop-to-hop) ใช้แมคแอดเดรส ดังนั้นเลเยอร์แมค(MAC) จึงต้องการแมคแอดเดรสของฮ็อพถัดไประหว่างไอพีแอดเดรสต้นทางและปลายทาง เราสามารถรู้แมคแอดเดรสของไอพีแอดเดรสที่กำหนดโดยใช้โพรโตคอลเออาร์พี แต่จะใช้ได้เฉพาะบนตัวกลางที่สนับสนุนการบรอดคาสท์เท่านั้น และแต่ละโหนดจะมีแคช(cache) ที่เรียกว่าแคชเออาร์พีซึ่งเก็บไอพีแอดเดรสและแมคแอดเดรสที่สัมพันธ์กัน เมื่อไอพีจะส่งดาต้าแกรมไปยังไอพีแอดเดรสอื่นมันจะหาแมคแอดเดรสของไอพีแอดเดรสที่เลเยอร์ดาต้าลิงค์จำเป็นต้องใช้ในการส่งจากแคชเออาร์พีก่อน ถ้าไม่พบมันจะพยายามหาแมคแอดเดรสจากไอพีแอดเดรสโดยใช้โพรโตคอลเออาร์พี ซึ่งการทำดังกล่าวนี้โพรโตคอลเออาร์พีจะส่งดาต้าแกรมร้องขอ(ARP request datagram) ไปยังทุกการ์ดแลนโดยใช้แมคแอดเดรสสำหรับการบรอดคาสท์ (0xFFFF_FFFF_FFFF) พร้อมทั้งไอพีแอดเดรสของแมคแอดเดรสที่ต้องการ การ์ดในเครือข่ายจะอ่านคำขอนี้และทุกการ์ดที่รู้คำตอบจะตอบกลับ(ARP response) ซึ่งเมื่อได้รับคำตอบ คำตอบนี้ก็จะถูกเก็บไว้ในแคชเพื่อใช้ต่อไปในอนาคต แต่ถ้าไม่ได้รับคำตอบภายในเวลาไม่กี่วินาทีเออาร์พีรีเควสก็จะถูกส่งซ้ำ เพราะเออาร์พีอาจถูกละทิ้งได้เนื่องจากความผิดพลาดในการส่งหรือความคับคั่งของบริดจ์(bridge) เพื่อลดความจำเป็นในการบรอดคาสท์เออาร์พี โหนดที่ตอบกลับจะคัดลอกไอพีแอดเดรสและแมคแอดเดรสของผู้ร้องขอเก็บไว้ในแคชเออาร์พีของมันด้วย ดาต้าแกรมเออาร์พีนี้ไม่สามารถผ่านเราท์เตอร์ได้เพราะเราท์เตอร์ทำงานที่เลเยอร์ไอพี และไม่รีเลย์(relay) การบรอดคาสท์โดยใช้แมคแอดเดรสซึ่งทำให้ป้องกันการเกิดข้อมูลจำนวนมาก(flooding) วิ่งไปทั่วทั้งระบบได้

รูปแบบของดาต้าแกรมเออาร์พีแสดงดังรูป 2-9 สามารถใช้กับเครือข่ายแบบใดก็ได้ ไม่เฉพาะทีซีพี/ไอพีเท่านั้น แต่ต้องมีตัวกลางที่สามารถส่งเฟรมบรอดคาสท์ได้ เออาร์พีทำงานโดยตรงบนเลเยอร์ดาต้า

ลิงค์ ดังนั้นจึงถูกเอ็นแคปซูเลชั่นโดยแฟรมดาต้าลิงค์เท่านั้น ทำให้มันต้องการฟิลด์ Ethernet type ของมันเอง คือ 0x0806



รูปที่ 2-9 รูปแบบดาต้าแกรมเออาร์พี

ฟิลด์ในดาต้าแกรมเออาร์พี ประกอบด้วย

1. **Hardware** บอกถึงชนิดของฮาร์ดแวร์ที่ใช้ในเครือข่ายซึ่งสร้างดาต้าแกรมนี้ขึ้นมา ชนิดที่ใช้ได้ คือ

Type	Description
1	Ethernet (10 Mbps)
2	Experimental Ethernet (3 Mbps)
3	Ameteur radio AX.25
4	Proteon ProNET Token Ring
5	Chaos
6	IEEE 802 networks
7	ARCNET
8	Hyperchannel
9	Lanstar

10	Autonet Short address
11	LocalTalk
12	LocalNet (IBM PCNet or Sytek Inc. LocalNet)
2. **Protocol**	แสดงถึงโพรโตคอลที่ร้องขอ ค่าที่ใช้ในฟิลด์นี้จะเหมือนกับฟิลด์ Ethernet type ในเฟรมอีเธอร์เน็ต ซึ่งก็คือ 0x0800 สำหรับ IP
3. **HLEN**	บอกถึงความยาวของฮาร์ดแวร์แอดเดรสในหน่วยออคเต็ด ปกติจะมีค่าเป็น6 สำหรับแมคแอดเดรสของแลนไอทริปเปิ้ลอี (IEEE)
4. **PLEN**	บอกถึงความยาวของแอดเดรสในเลเยอร์เน็ตเวิร์คในหน่วยออคเต็ด ปกติมีค่าเป็น 4 สำหรับ IP
5. **Operation**	มีค่าเป็น 1 สำหรับเออาร์พีรีเควส และ 2 สำหรับเออาร์พีเรสปอนด์ (ARP response) และยังใช้กับอาร์เออาร์พีด้วยโดยมีค่าเป็น 3 สำหรับอาร์เออาร์พีรีเควส และ 4 สำหรับอาร์เออาร์พีเรสปอนด์
6. **Addresses**	ประกอบด้วยฮาร์ดแวร์แอดเดรสของผู้ส่ง (แมคแอดเดรสต้นทาง), ไอพีแอดเดรสต้นทาง, ฮาร์ดแวร์แอดเดรสเป้าหมาย (แมคแอดเดรสปลายทาง), และไอพีแอดเดรสปลายทาง

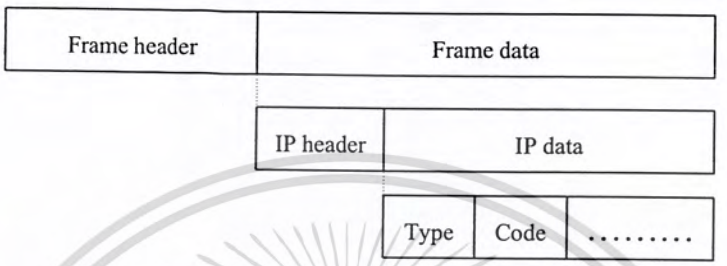
Reverse Address Resolution Protocol (RARP)

ใช้สำหรับอุปกรณ์ที่ไม่สามารถเก็บไอพีแอดเดรสของตัวเองได้ เช่นเวิร์คสเตชั่นที่ไม่มีฮาร์ดดิสก์ อาร์เออาร์พีทำงานในลักษณะตรงกันข้ามกับเออาร์พีคือหาไอพีแอดเดรสจากแมคแอดเดรสที่กำหนด อาร์เออาร์พีทำงานโดยตรงกับเลเยอร์คาต้าลิงค์โดยมีหมายเลขชนิดอีเธอร์เน็ตเท่ากับ 0x8035 โหนดที่ทำหน้าที่เป็นเซิร์ฟเวอร์อาร์เออาร์พี (RARP server) ที่พบแมคแอดเดรสที่กำหนดจะตอบกลับโดยอาร์เออาร์พีเรสปอนด์พร้อมทั้งไอพีแอดเดรสที่ต้องการ รูปแบบของดาต้าแกรมจะเหมือนเออาร์พีแต่ฟิลด์ Operation จะใช้ค่าเป็น 3 สำหรับรีเควส และ 4 สำหรับเรสปอนด์ ถึงแม้ว่าอาร์เออาร์พีจะทำงานได้ดีแต่ก็มีข้อจำกัดมากในทางปฏิบัติจึงถูกแทนที่โดยโพรโตคอลบูท (Boot Protocol: BOOTP) ซึ่งสามารถทำงานผ่านเราท์เตอร์และหาข้อมูลที่เป็นประโยชน์ได้มากกว่าอาร์เออาร์พีเมื่อเวิร์คสเตชั่นที่ไม่มีฮาร์ดดิสก์ทำการบูท

Internet Control Message Protocol (ICMP)

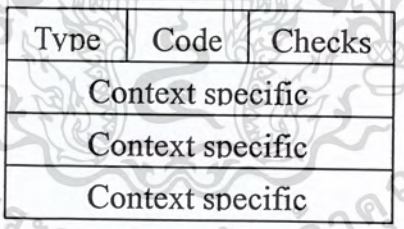
ถึงแม้ว่าไอพีจะไม่รับรองในการส่งข้อมูล แต่ไอซีเอ็มพีซึ่งใช้ได้ในไอพีสามารถสร้างแมสเสจเกี่ยวกับความผิดพลาดเพื่อช่วยเลเยอร์ไอพีในการให้บริการส่งข้อมูลให้ดีที่สุด และยังช่วยผู้ดูแลระบบในการวิเคราะห์หาสาเหตุเกี่ยวกับการทำงานของเครือข่าย ไอซีเอ็มพีใช้ดาต้าแกรมไอพีในการส่งแมสเสจระหว่างโหนด แมสเสจแสดงข้อผิดพลาดของไอซีเอ็มพีจะถูกสร้างโดยโหนดที่พบว่ามีปัญหาในการส่งเกิดขึ้นและจะ

ส่งแมสเสจนี้กลับไปยังแอดเดรสที่เป็นต้นทางของดาต้าแกรม ที่ทำให้เกิดปัญหา รูปที่ 2-10 แสดงถึงแมสเสจไอซีเอ็มพีที่ถูกเอ็นแคปซูเลทในดาต้าแกรมไอพีและแมสเสจแบบต่าง ๆ ที่เป็นไปได้ ไอซีเอ็มพีมีหมายเลขโพรโตคอล (protocol number) ของตัวเองซึ่งเท่ากับ 1 ทำให้ไอพีรู้ว่าได้รับไอซีเอ็มพี ถึงแม้ว่าไอซีเอ็มพีจะใช้เลเยอร์ไอพีแต่มันถูกมองว่าอยู่ภายในไอพีทั้งหมดเพราะไม่ได้ให้บริการแก่เลเยอร์ที่อยู่เหนือมัน



รูปที่ 2-10 แสดงไอซีเอ็มพีชนิดต่าง ๆ ถูกเอ็นแคปซูเลทในไอพี

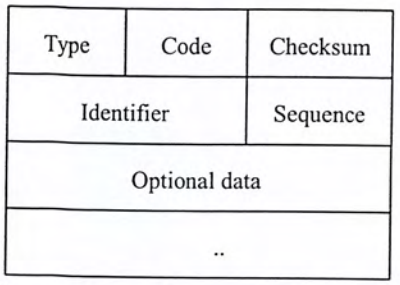
รูปแบบพื้นฐานของดาต้าแกรมไอซีเอ็มพีแสดงได้ดังรูป 2-11 แต่ฟิลด์ต่าง ๆ จะต่างกันซึ่งขึ้นอยู่กับชนิดที่ใช้อยู่ ฟิลด์ Type บ่งถึงชนิดของแมสเสจไอซีเอ็มพี ฟิลด์ Code ใช้แสดงถึงข่าวสารที่ละเอียดมากขึ้น ฟิลด์ Checksum ใช้เพราะไอพีไม่ได้ป้องกันข้อมูลของมันด้วยเช็คซัม(checksum) แต่เมื่อทำงานบนเครือข่ายฟิสิคัลซึ่งมีเฟรมเช็คซีเคว็นส์(Frame Check Sequence: FCS) เช็คซัมของไอซีเอ็มพีอาจเท่ากับ 0 หมายถึงไม่ถูกคำนวณ



รูปที่ 2-11 เฮดเดอร์พื้นฐานของโพรโตคอลไอซีเอ็มพี

1. ไอซีเอ็มพีชนิด 0 และ 8 – echo

ใช้เพื่อจุดประสงค์ในการหาสาเหตุ ถูกสร้างจากโปรแกรมอรรถประโยชน์(utility program) ที่รู้จักกันดีคือ ping ซึ่งจะส่งไอซีเอ็มพีชนิด 8 ไปยังโหนดและคาดว่าจะได้รับไอซีเอ็มพีชนิด 0 ตอบกลับมา รูปแบบของไอซีเอ็มพีสองชนิดนี้เป็นดังรูป 2-11

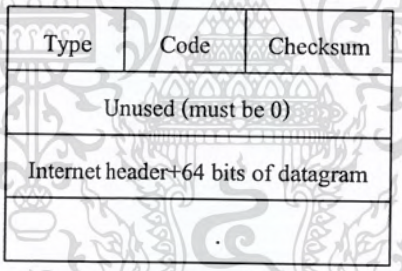


รูปที่ 2-12 รูปแบบดาต้าแกรมไอซีเอ็มพีชนิด echo

ฟิลด์ Identifier และ Sequence number ใช้ในการทำให้ดาต้าแกรมนี้แตกต่างจากดาต้าแกรมอื่น ถ้ามีข้อมูลส่งในฟิลด์ Optional data มันจะต้องถูกส่งกลับในการตอบกลับ

2. ไอซีเอ็มพีชนิด 3 – destination unreachable

ถ้าเราท์เตอร์ไม่สามารถส่งดาต้าแกรมได้ มันจะส่งแมสเสจไอซีเอ็มพีชนิด destination unreachable เพื่อบอกถึงสาเหตุ ฟิลด์ Code จะถูกใช้บอกถึงสาเหตุ ส่วนเฮดเดอร์อินเตอร์เน็ตรวมทั้งดาต้าแกรมพรีฟิกซ์ (datagram prefix) 64 บิตจะใช้ในการบอกถึงดาต้าแกรมที่เป็นสาเหตุของปัญหาดังรูปที่ 2-13



รูปที่ 2-13 ค่า destination unreachable ในฟิลด์ Code

Code value	Meaning
0	Network unreachable
1	Host unreachable
2	Protocol unreachable
3	Port unreachable
4	Fragmentation needed and the do not fragment bit set
5	Source route failed
7	Destination host unknown
11	Network unreachable for type of service

12	Host unreachable for type of service
13	Communication administratively prohibited e.g. firewall blocked
14	Host precedence violation
15	Precedence cut-off in effect

ซึ่งความหมายของแต่ละเหตุผลเป็นดังนี้

2.1 Network unreachable : หมายถึงเครือข่ายที่ระบุในไอพีแอดเดรสไม่สามารถพบได้ ควรจะตรวจที่ไอพีแอดเดรสหรืออาจเกิดความผิดพลาดในตารางเราท์เส้นทาง (routing table) ของเราท์เตอร์ระหว่างทาง แมสเสจแสดงข้อผิดพลาดนี้ถูกสร้างโดยเราท์เตอร์เท่านั้น จุดที่เกิดข้อผิดพลาดจะทราบได้จากแอดเดรสต้นทางในเฮดเดอร์ของไอพีที่บรรจุแมสเสจของไอซีเอ็มพีซึ่งก็คือเราท์เตอร์ที่เจอข้อผิดพลาดนั่นเอง

2.2 Host unreachable : ดาต้าแกรมที่เป็นสาเหตุของความผิดพลาดได้ไปถึงเราท์เตอร์ที่ต่อตรงกับเครือข่ายปลายทางแล้ว แต่เมื่อเราท์เตอร์พยายามที่จะส่งดาต้าแกรมมันกลับไม่สามารถสื่อสารกับโฮสนั้นได้ ซึ่งอาจเกิดจากเออาร์พีล้มเหลวในดาต้าแกรมแรก โฮสดาวน์ หรือเหตุอื่นใดก็ตามซึ่งอาจเพราะไม่มีไอพีแอดเดรสนั้น เช่นเดียวกับ Network unreachable คือแมสเสจนี้จะถูกสร้างจากเราท์เตอร์เท่านั้น จุดที่เกิดข้อผิดพลาดก็คือจุดที่เป็นแอดเดรสต้นทางในเฮดเดอร์ไอพีที่บรรจุแมสเสจไอซีเอ็มพี ซึ่งก็คือเราท์เตอร์ที่พบข้อผิดพลาดนั่นเอง

2.3 Protocol unreachable : ในกรณีนี้ดาต้าแกรมได้ไปถึงโฮสปลายทางแล้วแต่ไม่สามารถใช้โพรโตคอลที่ถูกพามาในดาต้าแกรมไอพีได้ ซึ่งพบได้ไม่บ่อยนัก แต่ก็เป็นไปได้ถ้าเครื่องระยะไกล (remote machine) นั้นถูกกำหนดติดตั้งผิด

2.4 Port unreachable : ส่งจากโฮสเพื่อบอกว่าบริการในเลเยอร์แอพพลิเคชั่นที่โฮสระยะไกลที่การติดต่อกำลังถูกก่อตั้งอยู่นั้นไม่พร้อมที่จะใช้งานได้(not available) ในแต่ละบริการของแอพพลิเคชั่น (application service) ที่โฮสนั้นจะถูกอีแนเบิ้ล(enable) และดิสแอเบิ้ล(disable) ขณะโฮสเริ่มทำงานโดยไฟล์คอนฟิคกูเรชั่น(configuration file) ดังนั้นเมื่อเกิดข้อผิดพลาดขึ้นก็ควรจะตรวจสอบที่ไฟล์นี้

2.5 Fragmentation needed and the do not fragment bit set : ปกติจะมาจากเราท์เตอร์ บ่งถึงความจำเป็นที่จะต้องแบ่งย่อยดาต้าแกรมแต่บิต Do not fragment หรือบิต DF ในฟิลด์ Flags ของเฮดเดอร์ไอพีไม่ยอมให้ทำ

2.6 Source route failed : ออปชั่นสำหรับการจัดการในไอพีอนุญาตให้ดาต้าแกรมไอพีไปตามเส้นทางที่กำหนดไว้ก่อนได้ แมสเสจนี้จะบ่งถึงข้อผิดพลาดที่ดาต้าแกรมไปตามเส้นทางที่กำหนดนี้ไม่สำเร็จ

2.7 Destination host unknown : ถูกสร้างขึ้นมาจากเราท์เตอร์เมื่อรู้จากซอฟต์แวร์สำหรับเชื่อมเลเยอร์ (link layer software) ว่าไม่มีโฮสปลายทาง

2.8 Network unreachable for type of service : สร้างจากเราท์เตอร์เมื่อพาธ (path) ที่จะไปถึงปลายทางไม่สอดคล้องกับที่ TOS ต้องการหรือไม่สอดคล้องกับ TOS ปกติ

2.9 Communication administratively prohibited : ถูกสร้างเมื่อเราท์เตอร์ไม่สามารถส่งแพ็คเกจต่อไปได้เนื่องมาจากการกรองแพ็คเกจ ตัวอย่างเช่น เหตุผลในความปลอดภัยหรือการคิดค่าบริการจากระบบ (local charging)

2.10 Host precedence violation : ถูกส่งจากเราท์เตอร์ที่เป็นฮ็อพแรก ไปยังโฮสเพื่อบอกว่าไม่สามารถกำหนดลำดับความสำคัญดังที่ร้องขอได้สำหรับโฮสต้นทางหรือปลายทาง เครือข่ายต้นทางหรือปลายทาง โพรโตคอลในเลเยอร์ที่สูงกว่า และพอร์ตต้นทางหรือปลายทาง

2.11 Precedence cut-off in effect : บ่งถึงว่าผู้ดูแลระบบเครือข่ายได้กำหนดค่าระดับความสำคัญที่เส้นทางนี้ต้องการไว้ต่ำสุด คือดาต้าแกรมถูกส่งด้วยลำดับความสำคัญที่ต่ำกว่า ที่ต้องการ

3. ไอซีเอ็มพีชนิด 4 และโค้ด 0 – source quence

รูปแบบของไอซีเอ็มพีชนิดนี้จะเหมือนกับไอซีเอ็มพีชนิด destination unreachable แต่จะมี Type เป็น 4 และ Code เป็น 0 เท่านั้น ไอซีเอ็มพีแบบนี้ใช้ในการทำโฟลวคอนโทรล(flow control) เราท์เตอร์ที่พบว่าเครือข่ายหรือโปรเซสเซอร์ถูกใช้งานหนักเกินไปจะส่งแมสเสจไอซีเอ็มพีนี้ไปยังโฮส ที่เป็นสาเหตุหลักของการใช้งานมาก ซึ่งโฮสดังกล่าวเมื่อได้รับแมสเสจก็จะลดอัตราการสร้างแพ็คเกจไปสู่ปลายทางที่ระบุมา

4. ไอซีเอ็มพีชนิด 5 – route change request

มีรูปแบบดังรูปที่ 2-14 ถูกใช้โดยเราท์เตอร์เท่านั้น สำหรับเราท์เตอร์ที่รู้ว่ามันไม่ใช่เราท์เตอร์ที่เหมาะสมที่สุดสำหรับการไปถึงปลายทางที่กำหนดก็จะใช้แมสเสจนี้แนะนำเราท์เตอร์ ที่เหมาะสมกว่าแก่ผู้ส่ง คือไอพีแอดเดรสต้นทางของดาต้าแกรม เพื่อความต่อเนื่องในการส่งข้อมูลเราท์เตอร์จะส่งดาต้าแกรมที่เป็นสาเหตุให้เกิดแมสเสจนี้ไปยังเราท์เตอร์ที่เชื่อว่าเข้าถึงเส้นทางที่ดีกว่าด้วย

(a)

Type	Code	Checksum
Internet address of a more suitable		
Internet header+64 bits of datagram		
.		

(b)

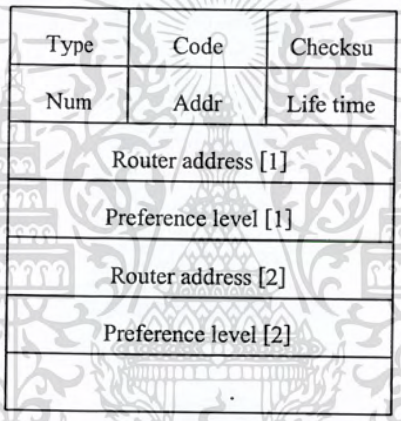
Code value	Meaning
0	Redirect datagrams to go to that network
1	Redirect datagrams to reach that host

2	Redirect datagrams for that network with that TOS
3	Redirect datagrams for that host with that TOS

รูปที่ 2-14 (a) รูปแบบของแมสเสจ route change request
(b) ค่าในฟิลด์ Code ที่แมสเสจประเภทนี้ใช้

5. ไอซีเอ็มพีชนิด 9 – router advertisement

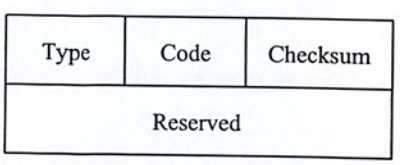
ทำให้เราท์เตอร์ประกาศตัวกับโฮสบนระบบเครือข่ายได้ดังรูปที่ 2-15 ซึ่งจะส่งทุก ๆ 7-10 นาทีหรือเพื่อตอบสนองโฮสจากแมสเสจไอซีเอ็มพีชนิด 10 แมสเสจนี้ไม่มีข้อมูลว่าจะติดต่อเราท์เตอร์นี้ผ่านเส้นทางไหน เพราะฉะนั้นถ้าโฮสเลือกเราท์เตอร์แรกไม่เหมาะสมก็จะได้รับแมสเสจไอซีเอ็มพีชนิด 5



รูปที่ 2-15 รูปแบบแมสเสจไอซีเอ็มพี router advertisement

6. ไอซีเอ็มพีชนิด 10 – router solicitation

สามารถถูกส่งได้โดยโฮสเวลาใดก็ได้ แต่มักจะเป็นตอนเปิดเครื่อง(start-up) เพื่อหาเราท์เตอร์ที่สามารถใช้ได้ในเครือข่ายที่โฮสอยู่ เราท์เตอร์จะตอบสนองแมสเสจนี้ด้วยแมสเสจไอซีเอ็มพีชนิด 9 (router advertisement response) หลังจากส่งแมสเสจนี้ไปแล้ว โฮสจะรอ unsolicited advertisements จากเราท์เตอร์ทุก ๆ 7-10 นาที รูปแบบของแมสเสจชนิดนี้เป็นดังรูป 2-16



รูปที่ 2-16 รูปแบบแมสเสจไอซีเอ็มพี router solicitation

7. ไอซีเอ็มพีชนิด 11 – time exceeded for datagram

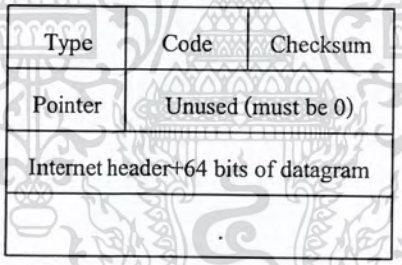
รูปแบบของไอซีเอ็มพีชนิดนี้จะเหมือนกับแมสเสจไอซีเอ็มพี destination unreachable ซึ่งแมสเสจชนิดนี้จะถูกส่งในสถานการณ์ดังต่อไปนี้

7.1 จากเราท์เตอร์ : ใช้บอกถึงว่าค่าฟิลด์ TTL ในเฮดเดอร์ไอพีถูกลดจนมีค่าเป็น 0 ในกรณีนี้ค่า Code จะเป็น 0 ทำให้ดาต้าแกรมถูกละทิ้งก่อนถึงปลายทาง ซึ่งส่วนใหญ่จะแสดงถึงว่าฟิลด์ TTL ที่ตั้งค่าเอาไว้ตอนเริ่มต้นไม่เหมาะสมหรือเกิดจากความเสียหายที่เกิดขึ้นกับเครือข่าย ซึ่งทำให้ความยาวของเส้นทางไม่ปกติ

7.2 จากโหนดปลาย : ค่าในฟิลด์ Code จะเป็น 1 บอกถึงการพยายามรวมส่วนย่อยเพื่อให้ เป็นดาต้าแกรมเดิมไม่สำเร็จ การรวมดาต้าแกรมอีกครั้งโดยใช้เวลามากจนเกินไปถ้าเกิดขึ้นไม่บ่อยนักก็ไม่ถือว่าเป็นปัญหาร้ายแรงแต่อย่างใด

8. ไอซีเอ็มพีชนิด 12 – parameter problem

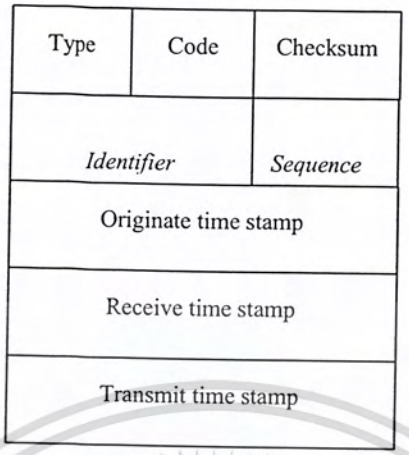
ใช้อาร์กิวเมนท์ผิดในฟิลด์ Option ของเฮดเดอร์ไอพี แต่ถ้าร้ายแรงกว่านั้นก็คือเกิดข้อผิดพลาดในการทำงานของไอพี มันแสดงถึงว่ามีค่าในเฮดเดอร์ที่ไม่สามารถเข้าใจได้ แมสเสจนี้จะไม่ถูกส่งถ้าดาต้าแกรมไม่ถูกละทิ้ง ฟิลด์ pointer บ่งถึงตำแหน่งของอ็อกเต็ดที่สงสัย เช่น ถ้ามีค่าเป็น 1 ก็หมายถึงฟิลด์ TOS ถ้ามีค่าเป็น 20 ก็จะหมายถึงอ็อกเต็ดแรกของฟิลด์ Option เป็นต้น รูปแบบของแมสเสจชนิดนี้เป็นดังรูปที่ 2-17



รูปที่ 2-17 รูปแบบแมสเสจไอซีเอ็มพี parameter problem

9. ไอซีเอ็มพีชนิด 13 และ 14 – time stamp request and reply

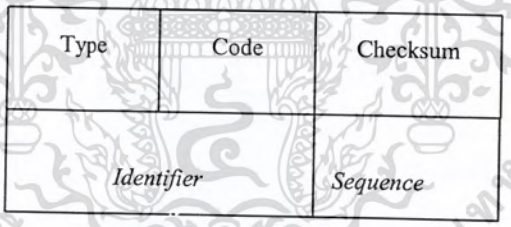
ใช้เก็บเวลาจากนาฬิกา(clock) ของเครื่องระยะไกล ผู้ร้องขอจะส่งแมสเสจไอซีเอ็มพีชนิด 13 และปลายทางจะตอบด้วยแมสเสจชนิด 14 ฟิลด์ Original time stamp จะถูกเติมก่อนดาต้าแกรมถูกส่ง ฟิลด์ Receive time stamp จะเติมทันทีที่ได้รับการร้องขอ และฟิลด์ Transmit time stamp จะถูกเติมทันทีก่อนที่จะตอบกลับไปยังต้นทาง รูปแบบของแมสเสจไอซีเอ็มพีชนิดนี้เป็นดังรูป 2-18 โดยแมสเสจชนิดนี้จะถูกใช้สำหรับเก็บสถิติเกี่ยวกับสมรรถนะของการเชื่อมต่อไปยังโฮสหรือเพื่อการเข้าจังหวะกัน ของนาฬิกาในโฮส



รูปที่ 2-18 รูปแบบแมสเสจไอซีเอ็มพี time stamp request/reply

10. ไอซีเอ็มพีชนิด 15 และ 16 – information request

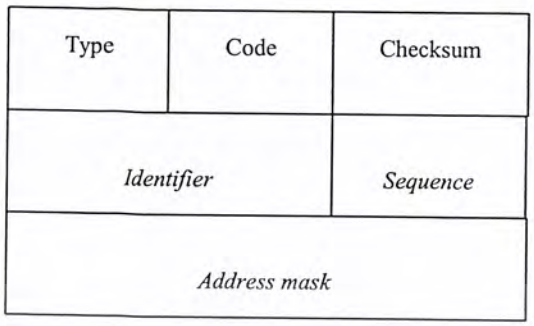
โฮสใช้เพื่อหาหมายเลขของเครือข่าย(network number) ถ้าโฮสนั้นไม่รู้ แอดเดรสที่ใช้ในเฮดเดอร์ไอพีจะมีค่าเป็น 0 หมายถึงเครือข่ายนี้ ซึ่งจะถูกเติมอย่างถูกต้องโดยปลายทางและถูกส่งกลับมา รูปแบบของแมสเสจไอซีเอ็มพีชนิดนี้เป็นดังรูป 2-19 กลไกนี้ใช้กับระบบไดอัลอิน(dial-in) ที่ใช้สลิป(SLIP) เป็นวิธีในการกำหนดเน็ตเวิร์คแอดเดรสที่เหมาะสมให้กับแต่ละปลายของการเชื่อมต่อ



รูปที่ 2-19 รูปแบบแมสเสจไอซีเอ็มพี information request

11. ไอซีเอ็มพีชนิด 17 และ 18 - address mask request

ใช้ร่วมกับการกำหนดแอดเดรสเป็นสับเน็ต (subnet addressing) ได้เพื่อให้โหนดรู้ถึงสับเน็ตมาสค์ (subnet mask) ของเครือข่ายที่มันต่ออยู่ด้วย โหนดสามารถส่งคำร้องขอไปยังแอดเดรสที่มันรู้จักซึ่งอาจเป็นเราท์เตอร์ หรือทำการบรอดคาสท์ไปยังเครือข่ายก็ได้ คำตอบกลับ(reply) จะส่งตรงถ้าโหนดรู้แอดเดรสของมันหรือทำการบรอดคาสท์ก็ได้ สับเน็ตมาสค์จะถูกใส่มาในฟิลด์ Address mask ของการตอบกลับดังรูป 2-20



รูปที่ 2-20 รูปแบบแมสเสจไอซีเอ็มพี address mask request

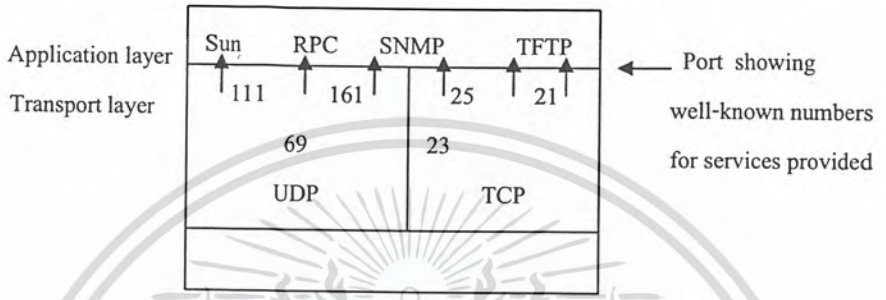
2.3.2 เลเยอร์ทรานสปอร์ตของทีซีพี/ไอพี

บริการที่ไอพีมีให้ยังต้องการให้ดาต้าแกรมส่งไปยังบริการในเลเยอร์แอพพลิเคชั่นที่ เหมาะสมโดยตรงและต้องการบริการที่เชื่อถือได้สำหรับแอพพลิเคชั่นที่จำเป็นต้องใช้มัน ซึ่งหน้าที่เหล่านี้เป็นของเลเยอร์นี้ทำโดยโพรโตคอลทรานสปอร์ต 2 ตัวคือโพรโตคอลยูดีพี (User Datagram Protocol: UDP) และโพรโตคอลทีซีพี(Transmission Control Protocol: TCP) ซึ่งการเลือกใช้นั้นขึ้นอยู่กับประเภทของบริการที่แอพพลิเคชั่นของผู้ใช้ต้องการ

เมื่อข้อมูลถูกส่งไปยังเครื่องที่ต้องการโดยไอพีมันจะถูกส่งไปยังบริการแอพพลิเคชั่น ที่เกี่ยวข้องบนเครื่องนั้น ๆ การมัลติเพล็กซ์และดีมัลติเพล็กซ์ข้อมูลไปยังหรือจากเลเยอร์ไอพีและ ส่งข้อมูลไปยังแอพพลิเคชั่นที่ถูกต้องเป็นหน้าที่อย่างหนึ่งของเลเยอร์ทรานสปอร์ตนี้ รวมทั้งการทำให้ไม่มีข้อผิดพลาด (error-free) และบริการส่งข้อมูลไปยังแอพพลิเคชั่นที่ถูกต้องแบบต้องก่อตั้งหรือไม่ก่อตั้งการเชื่อมต่อก่อนก็เป็นหน้าที่ของเลเยอร์นี้เช่นกัน โพรโตคอลยูดีพีจะให้บริการแบบไม่ต้องก่อตั้งการเชื่อมต่อก่อน (connectionless) ซึ่งเป็นบริการที่ไม่มีความน่าเชื่อถือ เพราะว่ามันจะอนุญาตให้ส่งข้อมูลไปยังเครื่องหรือกลุ่มของเครื่องโดยไม่จำเป็นต้องก่อตั้งการเชื่อมต่อก่อนดังนั้นดาต้าแกรมหนึ่งจะถูกส่งไปยังโหนดอื่นได้ โดยไม่ต้องการการตอบสนองว่าดาต้าแกรมดังกล่าวได้ไปถึงแล้วหรือยัง ในสิ่งแวดล้อมบางอย่างบริการแบบนี้ก็เป็นวิธีที่มีประสิทธิภาพในการทำงานมาก บริการแอพพลิเคชั่นที่ใช้โพรโตคอลนี้ได้แก่ ทีเอฟทีพี(TFTP) เอ็นเอฟเอส(NFS) และการบรอดคาสท์ เป็นต้น โพรโตคอลทีซีพีจะให้บริการแบบจำเป็นต้องก่อตั้งการเชื่อมต่อก่อน (connection-oriented) การเชื่อมต่อมีลักษณะคล้ายท่อของข้อมูลที่อยู่ระหว่างจุด 2 จุด ไม่มีการทำบรอดคาสท์หรือมัลติคาสท์ในโพรโตคอลนี้ โพรโตคอลทีซีพีมีฟีเจอร์(feature) ในการให้บริการที่น่าเชื่อถือระหว่างคอมพิวเตอร์ 2 เครื่อง เพื่อให้มีความน่าเชื่อถือโพรโตคอลทีซีพียังได้เพิ่มโอเวอร์เฮดจำนวนมากเพื่อใช้ในการทำแอคโนวเลดเมนท์(acknowledgement), โฟลวคอนโทรล(flow control), ไทม์เมอร์(timers) และความสะดวกสบายในการจัดการการเชื่อมต่อ(connection management facilities) ทีซีพีมีโอเวอร์เฮดมากกว่ายูดีพีใน

แง่ของการประมวลผลที่ต้องการและขนาดของเฮดเดอร์ที่ต้องใช้ ตัวอย่างของแอพพลิเคชั่นที่ต้องการบริการแบบนี้ได้แก่ เทลเน็ต(Telnet) และเอฟทีพี(FTP) เป็นต้น

ทั้งทีซีพีและยูดีพีต่างก็ใช้การอ้างแอดเดรสเป็นพอร์ต(port addressing) ในการส่งข้อมูลไปยังบริการในชั้นแอพพลิเคชั่นที่สัมพันธ์กัน ซึ่งพอร์ตก็คือแอดเดรสขนาด 16 บิตซึ่งหมายเลขของพอร์ตที่เป็นที่รู้จักกันดี (well-know port) ถูกกำหนดเป็น 0-255 ดังรูป 2-20 นอกจากนี้ยังมีการใช้ซ็อคเก็ต(socket)

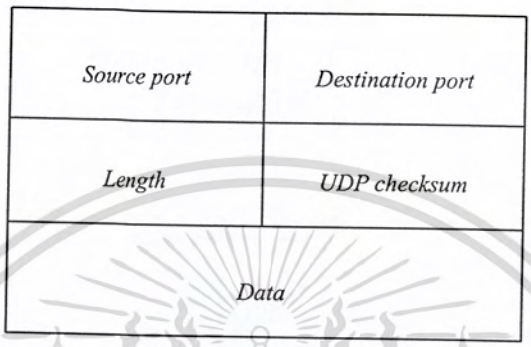


รูปที่ 2-21 หมายเลขพอร์ตที่ใช้ในยูดีพีและทีซีพี

ในทีซีพี/ไอพีอีกด้วย ซึ่งซ็อคเก็ตก็คือการนำไอพีแอดเดรสมาต่อกับหมายเลขพอร์ต เมื่อไอพีแอดเดรสนั้นไม่มีใครซ้ำในแต่ละโหนดและหมายเลขพอร์ตก็ไม่ซ้ำบนโหนดนั้น ดังนั้นซ็อคเก็ตก็จะเป็นการระบุถึงบริการในชั้นแอพพลิเคชั่นที่ไม่มีการใช้ซ้ำกัน เพราะซ็อคเก็ตไม่ซ้ำดังนั้นทั้งโพรโตคอลทีซีพีและยูดีพีจึงได้รวมไอพีแอดเดรสกับหมายเลขพอร์ตเข้าไปคำนวณเช็คซัมด้วยเพื่อให้แน่ใจได้ว่าดาต้าแกรม ที่ส่งไปถึงโฮสผิดจะไม่เป็นที่ยอมรับโดยเลเยอร์ทรานสปอร์ตของโฮสนั้นแม้หมายเลขพอร์ตนั้นจะเป็น ที่รู้จักกันดีก็ตาม บริการในชั้นแอพพลิเคชั่นส่วนใหญ่จะอนุญาตให้มีหลายเซสชั่น(session) ได้ซึ่งทำให้จำเป็นต้องแยกเซสชั่นเหล่านี้ให้ได้เพื่อให้แน่ใจได้ว่าข้อมูลจะถูกส่งกลับคอมพิวเตอร์ที่เหมาะสม ตัวอย่างเช่น ผู้ใช้ทุกคนที่ใช้เทลเน็ตจะติดต่ออยู่กับโฮสเดียวกันด้วยหมายเลขพอร์ตเดียวกันคือ 23 ทางหนึ่งที่จะแยกแยะได้คือดูว่าดาต้าแกรมมาจากไหน แต่ก็เป็นไปได้ที่ผู้ใช้ 2 คนจะติดต่อมาจากโฮสเดียวกัน การแก้ปัญหานี้ทำได้โดยใช้พอร์ตที่รู้จักกันดีกับบริการชั้นแอพพลิเคชั่นของเซิร์ฟเวอร์เท่านั้น และให้โปรแกรมไคลเอ็นท์เลือกหมายเลขพอร์ตที่ยังไม่มีใครใช้ในเครื่องนั้นมาใช้ ด้วยวิธีนี้ถึงแม้ว่า 2 เซสชั่นจะใช้เซิร์ฟเวอร์เดียวกันและยังมาจากโฮสเดียวกันก็ง่ายในการแยกแยะ 2 เซสชั่นนี้

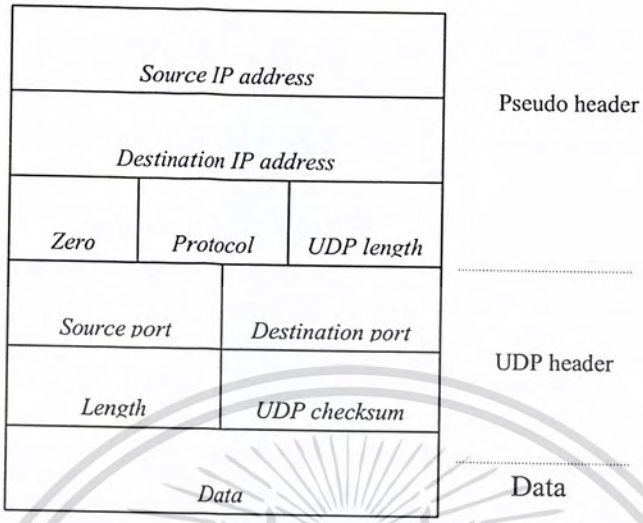
โพรโตคอลยูดีพี(User Datagram Protocol: UDP)

ยูดีพีเพิ่มความสามารถเข้าไปในการให้บริการของไอพีเพียงเล็กน้อย และใช้ฟิลด์ Source Port และ Destination Port ในเฮดเดอร์ของมันเพื่อส่งข้อมูลไปยังบริการในเลเยอร์แอพพลิเคชั่น ฟิลด์ต่าง ๆ ในโพรโตคอลยูดีพีเป็นดังรูป 2-22 ประกอบด้วย



รูปที่ 2-22 ฟิลด์ในเฮดเดอร์ของยูดีพี

1. **Source Port** บอกถึงว่าดาต้าแกรมมาจากพอร์ตหมายเลขอะไร ซึ่งก็คือบริการใดในชั้นแอพพลิเคชั่น มีขนาด 16 บิต
2. **Destination Port** บอกถึงว่าดาต้าแกรมนั้นต้องการส่งไปยังพอร์ตหมายเลขอะไร บริการใดในชั้นแอพพลิเคชั่น มีขนาด 16 บิต
3. **Length** ความยาวของดาต้าแกรมยูดีพี มีขนาด 16 บิต
4. **Checksum** เป็นการป้องกันข้อมูลที่ยูดีพีบรรทุกมา ซึ่งคิดทั้งฟิลด์ยูดีพีและซูโดเฮดเดอร์ (pseudo header) ดังรูป 2-23 มีขนาด 16 บิต



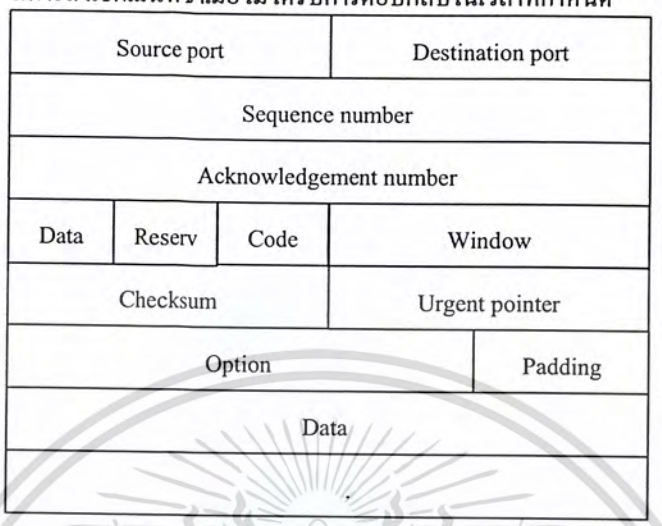
รูปที่ 2-23 ค่าเช็คซัมประกอบด้วยเฮดเดอร์และซูโดเฮดเดอร์

โพรโตคอลทีซีพี(Transmission Control Protocol: TCP)

ช่วยเพิ่มความน่าเชื่อถือให้ไอพีและใช้พอร์ตในการกำหนดแอดเดรสของเลเยอร์ แอพพลิเคชั่น เช่นเดียวกันกับยูดีพี โพรโตคอลทีซีพีเป็นโพรโตคอลที่ต้องการการเชื่อมต่อก่อน(connection-oriented) ดังกล่าวมาแล้ว คือจะต้องเปิดการติดต่อก่อนส่งและเมื่อส่งเสร็จก็จะต้องปิดการติดต่อนั้นด้วย ข้อมูลที่ทีซีพีส่งให้ไอพีนั้นจะประกอบด้วยเฮดเดอร์ของทีซีพีรวมกับข้อมูลจากเลเยอร์แอพพลิเคชั่น ซึ่งรวมกันแล้วเรียกว่าเซ็กเมนท์(segment) เพื่อความน่าเชื่อถือที่มากขึ้นเราต้องการสิ่งดังต่อไปนี้

- การตรวจและแก้ไขข้อผิดพลาด: ซึ่งเกี่ยวข้องกับความเป็นไปได้ที่เซกเมนท์ จะเสียหายจากสายสื่อสารหรือซอฟต์แวร์ในเลเยอร์ที่สูงกว่า
- โฟลวคอนโทรล: ใช้ในการป้องกันไม่ให้ผู้ส่ง ทำให้ผู้รับประสบปัญหาเนื่องมาจาก ข้อจำกัดในทรัพยากร
- การจัดลำดับการส่ง: จำเป็นต้องมีเพราะเลเยอร์ไอพีสามารถส่งดาต้าแกรม ซึ่งมีเซกเมนท์ทีซีพีในลำดับใดก็ได้ ซึ่งเกิดขึ้นเมื่อดาต้าแกรม ถูกส่งคนละเส้นทาง
- การกำจัดเซกเมนท์ที่ซ้ำ: เกิดเพราะกลไกกู้คืน (error-recovery) ที่ทีซีพีใช้ ทีซีพีทำการเพิ่มความสามารถที่กล่าวข้างต้นได้โดย
- ใช้หมายเลขแสดงลำดับในการแยกแยะข้อมูล
- ต้องได้รับเอคโนวเลดเมนท์ของการส่งข้อมูลในลำดับที่ถูกต้อง

- มีการส่งเซกเมนท์ซ้ำเมื่อไม่ได้รับการตอบกลับในเวลาที่กำหนด



รูปที่ 2-24 เฮดเดอร์ทีซีพี

และเพื่อให้เกิดหน้าที่ที่ได้กล่าวข้างต้น เฮดเดอร์ทีซีพีจึงซับซ้อนมีฟิลด์มากกว่าเฮดเดอร์ยูดีพี ดังรูป 2-23 ทีซีพีมีฟิลด์ Source port และ Destination port ด้วยเหตุผลเดียวกันกับยูดีพีคือใช้แยกแยะแอพพลิเคชั่น ฟิลด์ที่เหลือส่วนมากมีเพื่อความน่าเชื่อถือและเกี่ยวข้องกับการควบคุมการติดต่อ ดังนี้

1. **Sequencing number** มีขนาด 32 บิต ในทีซีพีจะนับออคเต็ดในการส่ง แต่โพรโตคอลอื่นที่ใช้เลขลำดับเพื่อควบคุมความผิดพลาดจะนับเซ็กเมนท์ เลขลำดับในเฮดเดอร์นี้จะกำหนดตำแหน่งในข้อมูลทั้งหมดของออคเต็ดแรกในเซ็กเมนท์ ซึ่งช่วยในทีซีพีใส่เซ็กเมนท์ในตำแหน่งที่ถูกต้องในข้อมูลได้แม้ไอพี จะส่งข้อมูลไม่เป็นลำดับก็ตาม การที่มันมี 32 บิตทำให้ไม่เกิดการซ้ำของค่าแม้เวลาในการส่งจะเร็วมากก็ตาม คือเซ็กเมนท์ที่ได้รับอาจมีเลขลำดับซ้ำ กับเซ็กเมนท์ซึ่งแอคโนวเลดไปเรียบร้อยแล้วได้ แต่ถ้าเกิดข้อผิดพลาดเนื่องจากการซ้ำก็ไม่เป็นปัญหา เพราะแค่ไม่สนใจโดยไม่ต้องทำอะไร

2. **Acknowledgement number** มีขนาด 32 บิต บอกให้รู้ว่าได้รับออคเต็ดทั้งหมดอย่างถูกต้องจนถึงเลขแอคโนวเลดลบด้วย 1 เมื่อผู้ส่งได้รับค่านี้ก็ไม่จำเป็นต้องเก็บข้อมูลไว้ เพื่อส่งใหม่อีกต่อไป ซึ่งเลขแอคโนวเลดนี้จะใช้ได้เมื่อเซ็ตแฟลก ACK

3. **Data Offset** วัดออฟเซ็ตที่เป็นจุดเริ่มต้นของฟิลด์ข้อมูลในหน่วย 32 บิตเวิร์ด ค่าปกติคือ 5 ซึ่งก็คือเฮดเดอร์ 20 ออคเต็ดเมื่อไม่ใช้ฟิลด์ออปชั่น ฟิลด์นี้มีขนาด 4 บิต

4. **Flags** มีขนาด 6 บิต ต่อจากฟิลด์ Reserve ซึ่งมีขนาด 6 บิตเช่นกัน ใช้บอกว่าฟิลด์อื่นใช้ได้หรือไม่ และสำหรับการควบคุมการติดต่อ ประกอบด้วย 6 แฟลกดังนี้

- URG บอกว่าใช้ฟิลด์ urgent pointer ได้ซึ่งฟิลด์นี้ชี้ไปยังออคเต็ดในฟิลด์ข้อมูล ซึ่งเป็นปลายของข้อมูล urgent ซึ่งไม่ถูกมองเป็นข้อมูลปกติและควร จะถูกประมวลผลก่อนข้อมูลอื่น ๆ
- ACK บอกว่าฟิลด์ Acknowledge ใช้ได้ ซึ่งฟิลด์นี้จะใช้ไม่ได้เมื่อก่อตั้งการเชื่อมต่อ คือก่อนที่แต่ละโหนดจะสามารถตัดสินใจได้ว่าจะใช้ค่า sequence และ acknowledge ใด
- PSH คือแฟลก push ซึ่งทำให้เลเยอร์ทีซีพีที่ระยะไกลส่งเซ็กเมนท์นี้ ไปให้เลเยอร์แอพพลิเคชั่นอย่างทันทีทันใด ปกติทีซีพีจะหันมาสนใจข้อมูลจากเซ็กเมนท์ ที่เข้ามาและส่งข้อมูลนี้ไปยังเลเยอร์แอพพลิเคชั่นในบัฟเฟอร์ที่ใหญ่กว่าเพื่อลด โอเวอร์เฮดในการประมวลผล
- RST คือแฟลก reset ใช้เมื่อเกิดข้อผิดพลาดอื่น ๆ บอกถึงว่ามีข้อผิดพลาดเกิดขึ้นและการควรจะหยุดการติดต่อ
- SYN คือแฟลก synchronize ใช้ขณะเริ่มต้นการก่อตั้งการเชื่อมต่อระหว่าง 2 โหนดซึ่ง ณ เวลานั้นทั้ง 2 โหนด ไม่รู้ว่าควรจะใช้เลขแอคโนวเลดใด การก่อตั้งการเชื่อมต่อจะประกอบด้วยการแลกเปลี่ยนเซ็กเมนท์แบบ 2 ทาง (2-way exchange of segment) พร้อมทั้งเซ็ตแฟลก SYN ซึ่งแต่ละอันจะถูกแอคโนวเลดในเซ็กเมนท์โดยเซ็ตแฟลก ACK
- FIN ใช้ในการเลิกการติดต่อเมื่อข้างใดข้างหนึ่งไม่มีข้อมูลที่จะส่งก็จะส่งเซ็กเมนท์ ที่มีการเซ็ตแฟลก FIN เมื่อทั้ง 2 ข้างส่งแฟลก FIN การติดต่อก็จะปิดลง

5. **Window** มีขนาด 16 บิต บอกถึงขนาดเนื้อที่ในบัฟเฟอร์ของโหนดนี้ที่ใช้ได้ในการเชื่อมต่อนี้ โหนดอื่นจะต้องไม่ส่งข้อมูลที่ยังไม่แอคโนวเลดมาเกินเนื้อที่ในบัฟเฟอร์ที่ระบุนี้

6. **Checksum** มีขนาด 16 บิต ใช้ตรวจสอบเฮดเดอร์และข้อมูล

7. **Urgent pointer** มีขนาด 16 บิต ค่าในฟิลด์นี้ชี้ไปยังปลายของข้อมูลในฟิลด์ข้อมูลที่เร่งด่วน และต้องการความสนใจทันที จะใช้ได้เมื่อมีการเซ็ตแฟลก URG

8. **Options** ขนาดไม่คงที่ มีออปชั่นเดียวที่ใช้เป็นปกติในทีซีพีคือขนาดเซ็กเมนท์มากที่สุด (Maximum Segment Size:MSS) เพื่อบอกเลเยอร์ทีซีพีปลายทางถึงขนาดเซ็กเมนท์มากที่สุดที่ควรจะส่งซึ่งรวมเฮดเดอร์ทีซีพีแล้ว

9. **Padding** ถ้าใช้ฟิลด์ออปชั่นแพดดิ้ง จะทำให้มั่นใจได้ว่าข้อมูลเริ่มที่ขอบ 32 บิต อย่างที่ออปเซ็ตข้อมูลชี้อย่างถูกต้อง

2.3.3 เลเยอร์แอพพลิเคชั่นของทีซีพี/ไอพี

บริการของเลเยอร์แอพพลิเคชั่นจะรับผิดชอบในการเชื่อมต่อ(interface) ระหว่างแอพพลิเคชั่นของผู้ใช้ และบริการในชั้นทรานสปอร์ต บริการของแอพพลิเคชั่นไม่ใช่แอพพลิเคชั่นของผู้ใช้แต่เป็นการเชื่อมต่อกับแอพพลิเคชั่นนั้นกับเครือข่ายสื่อสาร มีบริการของแอพพลิเคชั่นหลายตัวที่เหมาะสมกับแอพพลิเคชั่นหลายชนิด และยังมียูทิลิตี้ในการจัดการอีกจำนวนหนึ่ง ตัวอย่างของบริการในชั้นนี้คือ

1. **FTP** ย่อมาจาก File Transfer Protocol ใช้พอร์ตหมายเลข 20 เป็นโพรโตคอลมาตรฐานและเป็นวิธีที่ง่ายที่สุดในการแลกเปลี่ยนไฟล์กันในอินเทอร์เน็ต FTP เป็นโพรโตคอลแอพพลิเคชั่นที่ใช้โพรโตคอลชุดทีซีพี/ไอพี และมักจะถูกใช้เสมอในการส่งไฟล์เว็บเพจจากผู้สร้างเว็บเพจไปยังคอมพิวเตอร์ที่ทำตัวเป็นเซิร์ฟเวอร์เพื่อให้ใครก็ตามในอินเทอร์เน็ต สามารถใช้ได้ นอกจากนี้ FTP ยังใช้ในการดาวน์โหลดโปรแกรมและไฟล์อื่น ๆ จากเซิร์ฟเวอร์มายังคอมพิวเตอร์ของเราได้ด้วย ในฐานะผู้ใช้เราสามารถใช้ FTP ด้วยอินเทอร์เฟสแบบคอมมานไลน์ง่าย ๆ เช่นจากหน้าต่างคอสพรอมท์ หรือด้วยโปรแกรมที่มีขายซึ่งจะเสนออินเทอร์เฟสแบบกราฟฟิกให้ นอกจากนี้เว็บบราวเซอร์ของเราก็ยังสามารถใช้สำหรับดาวน์โหลดโปรแกรมที่เราเลือกจากเว็บเพจโดยส่ง FTP request ได้ด้วย ในการใช้ FTP เรายังสามารถอัพเดท หมายถึงการลบ การเปลี่ยนชื่อ การย้ายตำแหน่ง และการคัดลอกไฟล์ที่เซิร์ฟเวอร์ได้อีกด้วยแต่เราจำเป็นต้องล็อกออนเข้าไปในเซิร์ฟเวอร์นั้นก่อน อย่างไรก็ตามไฟล์ที่ให้ใครก็ได้ใช้สามารถเข้าถึงได้โดย anonymous FTP

2. **Telnet** ใช้พอร์ตหมายเลข 23 Telnet คือทางที่จะช่วยให้เราสามารถเข้าถึงคอมพิวเตอร์ของใครก็ตามถ้าเขาอนุญาตซึ่งมักจะเรียกคอมพิวเตอร์ลักษณะนี้ว่าโฮสคอมพิวเตอร์ หรือถ้าจะกล่าวให้ลึกกว่านั้นก็สามารถจะพูดได้ว่า Telnet คือคำสั่งของผู้ใช้บนโพรโตคอลทีซีพี/ไอพีสำหรับการเข้าถึงคอมพิวเตอร์ระยะไกล โพรโตคอลเว็บหรือ HTTP และ FTP นั้นอนุญาตให้เราร้องขอไฟล์ที่เจาะจงจากคอมพิวเตอร์ระยะไกลแต่ไม่ได้ให้เราล็อกออนจริง ๆ ในฐานะผู้ใช้ของคอมพิวเตอร์เครื่องนั้น แต่ด้วย Telnet เราสามารถล็อกออนเหมือนเป็นผู้ใช้ปกติด้วยสิทธิอะไรก็ตามที่เราได้รับอนุญาตให้ทำบนเครื่องคอมพิวเตอร์เครื่องนั้น

3. **SMTP** ย่อมาจาก Simple Mail Transfer Protocol ใช้พอร์ตหมายเลข 25 SMTP คือโพรโตคอล TCP/IP ที่ถูกใช้ในการส่งหรือรับอีเมล์(e-mail) อย่างไรก็ตามเพราะข้อจำกัดในความสามารถของมันในการจัดคิวข่าวสารที่ฝั่งผู้รับ เราจึงมักจะใช้โพรโตคอลอื่นแทนเช่น POP3 หรือ IMAP ซึ่งจะให้ผู้ใช้เก็บข่าวสารในกล่องจดหมาย(mail box) ของเซิร์ฟเวอร์และดาวน์โหลดข่าวสารเหล่านั้นจากเซิร์ฟเวอร์เป็นระยะ ๆ หรือกล่าวได้อีกอย่างว่าปกติแล้ว

ผู้ใช้จะใช้โปรแกรมที่ใช้ SMTP สำหรับการส่งอีเมล์และใช้ POP3 หรือ IMAP สำหรับการรับอีเมล์ โปรแกรมเกี่ยวกับการเมล์ส่วนใหญ่ เช่น Eudora จะให้เราระบุทั้งเซิร์ฟเวอร์ SMTP และเซิร์ฟเวอร์ POP

4. **Gopher** ใช้พอร์ตหมายเลข 70 Gopher เป็นโพรโตคอลชั้นแอพพลิเคชั่นในเซิร์ฟเวอร์ซึ่งโครงสร้างไฟล์ถูกจัดการเรียงเป็นลำดับชั้น Gopher ได้จัดหาทางที่จะนำเท็กซ์ไฟล์จากทั่วโลกมายัง viewer บนคอมพิวเตอร์ของเรา โกเฟอร์ได้รับความนิยมเป็นเวลาหลายปีโดยเฉพาะอย่างยิ่งในมหาวิทยาลัย และยังเป็นก้าวหนึ่งที่นำไปสู่ HTTP แต่ด้วยไฮเปอร์เท็กซ์ลิ้งค์ ภาษา HTML และการปรากฏตัวของบราวเซอร์แบบกราฟฟิกทำให้โกเฟอร์เสื่อมความนิยมลงอย่างรวดเร็ว โครงสร้างไฟล์แบบดั้งเดิมจำนวนหนึ่งโดยเฉพาะในมหาวิทยาลัยยังคงใช้อยู่และสามารถเข้าถึงโดยเว็บบราวเซอร์ส่วนใหญ่เพราะมันยังคงสนับสนุนโพรโตคอลโกเฟอร์ Gopher ถูกพัฒนาที่มหาวิทยาลัยมินเนโซต้า(the University of Minnesota) ถึงแม้ว่าบราวเซอร์โกเฟอร์และไฟล์จะเป็นเท็กซ์แต่บราวเซอร์โกเฟอร์ก็ได้ถูกพัฒนาให้แสดงรูปกราฟฟิกได้คือไฟล์ GIF และ JPEG ซึ่งถูกรวมไว้ในไฟล์ไดเรคทอรี่โกเฟอร์

5. **HTTP** ย่อมาจาก Hypertext Transfer Protocol (HTTP) ใช้พอร์ตหมายเลข 80 HTTP เป็นชุดของกฎสำหรับการแลกเปลี่ยนไฟล์ ซึ่งมีทั้งเท็กซ์ กราฟฟิก ภาพ เสียง วีดีโอ และไฟล์มัลติมีเดียอื่น ๆ บนเวิลด์ไวด์เว็บ(World Wide Web) เมื่อเปรียบเทียบกับชุดโพรโตคอลทีซีพี/ไอพีซึ่งเป็นพื้นฐานสำหรับการแลกเปลี่ยนข้อมูลข่าวสารบนอินเทอร์เน็ต HTTP ก็คือแอพพลิเคชั่นโพรโตคอล คอนเซ็ปท์(concepts) สำคัญที่เป็นส่วนหนึ่งของ HTTP ประกอบไปด้วยความคิดที่ว่าไฟล์สามารถอ้างอิงไปยังไฟล์อื่นได้ โดยเว็บเซิร์ฟเวอร์ใด ๆ ก็ตามนอกจากจะเก็บไฟล์ HTML และไฟล์อื่น ๆ แล้วยังมี HTTP daemon ซึ่งเป็นโปรแกรมที่ถูกออกแบบมาให้รอ HTTP requests และจัดการเมื่อคำร้องขอมาถึง เว็บบราวเซอร์ของเราก็คือไคลเอ็นต์ HTTP ซึ่งจะส่งคำร้องขอไปยังเซิร์ฟเวอร์ เมื่อผู้ใช้บราวเซอร์ใส่การร้องขอไฟล์โดยการเปิดเว็บไฟล์(โดยการพิมพ์ URL: Uniform Resource Locator) หรือคลิ๊ก(click) บน hypertext link บราวเซอร์ก็จะสร้าง HTTP request และส่งไปยังไอพีแอดเดรสที่ระบุโดย URL หลังจากนั้น HTTP daemon ที่เซิร์ฟเวอร์ปลายทางจะได้รับคำร้องขอและเมื่อทำการประมวลผลสิ่งที่จำเป็นแล้วไฟล์ที่ถูกร้องขอก็จะถูกส่งกลับ

6. **POP3** ย่อมาจาก Post Office Protocol 3 ใช้พอร์ตหมายเลข 110 POP3 เป็นเวอร์ชั่นล่าสุดของโพรโตคอลมาตรฐานสำหรับการรับอีเมล์ POP3 เป็นโพรโตคอลแบบไคลเอ็นต์/เซิร์ฟเวอร์ซึ่งจะรับอีเมล์ที่ถูกส่งมาและเก็บเอาไว้ให้ในเซิร์ฟเวอร์ของเรา เราสามารถดูเมล์ได้ที่เซิร์ฟเวอร์และดาวน์โหลดได้ นอกจาก POP3 แล้วยังมีโพรโตคอลที่ทำงานคล้ายคลึงกันคือโพรโตคอล IMAP (Interactive Mail Access Protocol) ด้วย IMAP เราสามารถดูอีเมล์ที่

เซิร์ฟเวอร์เหมือนกับว่าอีเมล์เหล่านั้นอยู่บนเครื่องคอมพิวเตอร์ของเราเอง อีเมล์ที่ถูกลบที่เครื่องเราจะยังอยู่บนเซิร์ฟเวอร์เช่นเดิม นอกจากนี้อีเมล์ยังสามารถเก็บและค้นหาได้ที่เซิร์ฟเวอร์ เราสามารถคิดได้ว่า POP คือบริการแบบเก็บและส่งต่อไป (store-and-forward) ส่วน IMAP ก็คือไฟล์เซิร์ฟเวอร์ระยะไกล(remote file server) POP และ IMAP เกี่ยวข้องกับการรับอีเมล์และไม่ยุ่งเกี่ยวกับ SMTP ซึ่งเป็นโพรโตคอลสำหรับส่งอีเมล์ข้ามอินเทอร์เน็ต

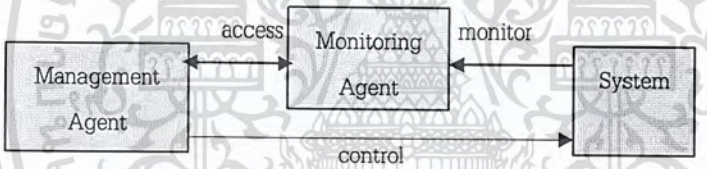
7. **NNTP** ย่อมาจาก Network News Transfer Protocol ใช้พอร์ตหมายเลข 119 NNTP คือโพรโตคอลที่ใช้โดยคอมพิวเตอร์ทั้งเซิร์ฟเวอร์และไคลเอ็นต์สำหรับจัดการข้อความ (notes) ที่ทิ้งไว้บนกลุ่มข่าว Usenet(Usenet newsgroups) NNTP ได้มาแทนที่โพรโตคอล Usenet ดั้งเดิมคือ UUCP(UNIX-to-UNIX Copy Protocol) เซิร์ฟเวอร์ NNTP จะจัดการเครือข่ายของกลุ่มข่าว Usenet ที่ถูกรวบรวมและรวมเซิร์ฟเวอร์เข้าไว้ที่ผู้ให้บริการอินเทอร์เน็ตของเรา ไคลเอ็นต์ NNTP อาจจะถูกรวมเป็นส่วนหนึ่งของ Netscape, Internet Explorer, Opera หรือเว็บบราวเซอร์อื่น ๆ หรือเราอาจจะใช้โปรแกรมแยกต่างหากที่เรียกว่า newsreader ก็ได้

8. **SNMP** ย่อมาจาก Simple Network Management Protocol ใช้พอร์ตหมายเลข 161 SNMP คือโพรโตคอลที่ใช้บริหารการจัดการเครือข่ายและการมอนิเตอร์อุปกรณ์ในเครือข่าย และฟังก์ชั่นของอุปกรณ์เหล่านั้น ซึ่งไม่ได้จำกัดอยู่เฉพาะเครือข่ายที่ใช้ทีซีพี/ไอพี

9. **IRC** ย่อมาจาก Internet Relay Chat (IRC) ใช้พอร์ตหมายเลข 194 IRC คือระบบสำหรับ chat ที่เกี่ยวข้องกับชุดของกฎ ข้อตกลงและซอฟต์แวร์ประเภทไคลเอ็นต์/เซิร์ฟเวอร์ ในเว็บมีไซต์เฉพาะเช่นเมืองแห่งการคุย(Talk City) หรือเครือข่าย IRC และช่วยให้เราดาวน์โหลดไคลเอ็นต์ IRC มายังเครื่องคอมพิวเตอร์ของเรา เราสามารถเริ่มการคุยในกลุ่ม (เรียกว่าแชนแนล) ใดก็ได้ที่มีอยู่ ซึ่งมีโพรโตคอลสำหรับค้นหากลุ่มการคุยที่มีอยู่และสมาชิกของกลุ่มนั้น ๆ ด้วย ผู้ที่เข้าไปร่วมคุยในกลุ่มการคุยใดก็ตามจะใช้ชื่อเล่นซึ่งใช้ได้เฉพาะครั้งนั้น ๆ (เราไม่สามารถเป็นเจ้าของชื่อเล่นนั้นได้คืออาจมีคนใช้ซ้ำกับเราได้)

บทที่ 3
หลักการในการมอนิเตอร์เครือข่ายและการออกแบบตัวมอนิเตอร์
(The overview and design of network monitoring)

การมอนิเตอร์เครือข่ายเป็นการช่วยแก้ปัญหาทั่ว ๆ ไปของการจัดการระบบเครือข่าย ซึ่งไม่ใช่เฉพาะระบบเครือข่ายเท่านั้นที่จำเป็นต้องมีการมอนิเตอร์ ในทุกระบบที่ใหญ่และซับซ้อนต่างก็จำเป็นต้องมีการมอนิเตอร์ด้วย อาทิเช่น ในการจราจรการมอนิเตอร์จะมีไว้เพื่อช่วยในการจัดการการคับคั่งของการใช้เส้นทางจราจร และข้อมูลจากการมอนิเตอร์นี้ก็จะถูกเก็บไว้เพื่อนำมาใช้วางแผนระยะยาวเกี่ยวกับการสร้างถนนเพิ่ม ขยายถนน หรือบำรุงถนน อีกตัวอย่างหนึ่งก็คือ ระบบโทรศัพท์ การมอนิเตอร์สามารถจะช่วยให้เกิดการเตือนเมื่อมีความผิดปกติเกิดขึ้นได้ และที่สำคัญจะช่วยในการตรวจนับอัตราการใช้โทรศัพท์ของลูกค้าเพื่อนำมาคิดค่าบริการได้อย่างถูกต้อง ดังนั้นการมอนิเตอร์เครือข่ายก็คือกลไกหนึ่งที่ช่วยให้ผู้ดูแลระบบเครือข่าย(network administrators) ทราบสถานะของเครือข่ายและทราบถึงแนวโน้มในระยะยาวของระบบเครือข่ายคอมพิวเตอร์ที่ ซับซ้อนได้ ซึ่งตัวอย่างทั่ว ๆ ไปคือระบบการควบคุมแบบป้อนกลับ (feedback system) ดังรูปที่ 3-1 ข้อมูลเกี่ยวกับระบบ(system) จะถูกมอนิเตอร์โดยมอนิเตอริ่งเอเจนท์ (monitoring agent) ซึ่งอาจมีได้มากกว่าหนึ่งตัว โดยมีแมนเนจเมนท์เอเจนท์(management agent)



รูปที่ 3-1 การมอนิเตอร์ในระบบป้อนกลับ

นำข้อมูลที่มอนิเตอริ่งเอเจนท์หามาไปใช้ในการวิเคราะห์และส่งการควบคุมกลับไปยังระบบ ที่ถูกมอนิเตอร์

ในการมอนิเตอร์ระบบเครือข่ายนั้นจะต้องมีการเกี่ยวข้องกับกิจกรรม 3 อย่างต่อไปนี้

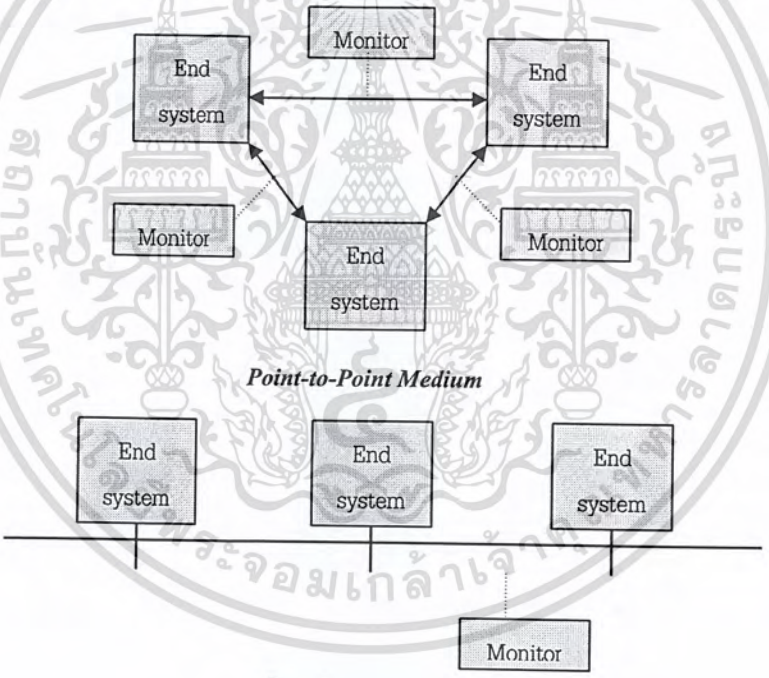
3.1 การเข้าถึงข้อมูลที่จะมอนิเตอร์

จะเกี่ยวข้องกับการพิจารณาว่าจะกำหนดความหมายและรูปแบบของข้อมูลอย่างไร จึงจะทำให้แมนเนจเมนท์เอเจนท์หลายตัวสามารถเข้าถึงและเข้าใจได้(ลูกศรแอคเซส (access) ในรูปที่ 3-1) ซึ่งถ้าต้องการให้สิ่งที่ถูกจัดการซึ่งถูกผลิตจากหลายผู้ผลิตสามารถมอนิเตอร์และจัดการได้เหมือนกัน ก็จะต้องมีการกำหนดมาตรฐานออกมาใช้

3.2 การออกแบบกลไกในการมอนิเตอร์

เกี่ยวข้องกับการหาวิธีที่จะทำให้ได้รับข้อมูลเกี่ยวกับสถานะของเครือข่ายดีที่สุด (ลูกศรมอนิเตอร์ (monitor) และกล่องมอนิเตอริ่งเอเจนท์ในรูปที่ 3-1) จะต้องคิดว่าจะนำมอนิเตอริ่งเอเจนท์ไปไว้ตรงไหน จากแบบอ้างอิงของโอเอสไอ (OSI) เราสามารถจะมอนิเตอร์ได้ทุกเลเยอร์และทุกส่วนประกอบ

(component) ของเครือข่าย ซึ่งนำไปสู่การนำไปไว้ที่ระบบปลาย(end systems) และระบบกลาง (intermediate systems) สถานะที่แน่นอนจะสามารถมอนิเตอร์ได้เมื่อมอนิเตอริ่งเอเจนท์เป็นส่วนหนึ่งของสิ่งที่เรากำลังมอนิเตอร์อยู่เท่านั้นซึ่งเอเจนท์ที่มีลักษณะแบบนี้เรียกว่า อินทิเกรตเต็ดมอนิเตอริ่งเอเจนท์ (integrated monitoring agent) ซึ่งมีข้อดีที่เราสามารถจะมอนิเตอร์สถานะของสิ่งที่เราสนใจได้อย่างแม่นยำทันทีทันใด และลึกเท่าที่ต้องการได้ กลไกในการรวบรวมข้อมูลเพื่อใช้ในการจัดการเครือข่ายจะต้องไม่กระทบกับจุดประสงค์หลักของเครือข่ายคือการสื่อสาร ข้อมูลของแอปพลิเคชั่น(application) ดังนั้นถ้าเครือข่ายมีขนาดใหญ่ประกอบด้วยหลายส่วนประกอบ การใช้ความสามารถในการมอนิเตอร์กับทุกส่วนประกอบจะเป็นการสร้างปัญหาคือทำให้ แบนด์วิธ(bandwidth) ที่จะใช้ส่งข้อมูลจริง ๆ ลดลงเพราะฟังก์ชันในการมอนิเตอร์ จากการพัฒนาและความนิยมของเทคโนโลยีแลน(LAN) ทำให้เกิดความเป็นไปได้ในการใส่กลไกเดี่ยวเข้าไปในตัวกลางที่ใช้สื่อสาร (physical medium) ร่วมกัน ซึ่งจะทำให้สามารถมอนิเตอร์การสื่อสารระหว่างระบบจำนวนมากได้ดังรูปที่ 3-2 ซึ่งลักษณะเช่นนี้เรียกว่าเอ็กเทอร์นอลมอนิเตอริ่งเอเจนท์(external monitoring agent) ซึ่งถูกใช้สำหรับจัดการเครือข่ายที่เป็นแลน แต่มีข้อจำกัดหลายข้อเกี่ยวกับความสามารถของตัว



รูปที่ 3-2 เอ็กเทอร์นอลมอนิเตอริ่งแบบต่าง ๆ

มอนิเตอร์ในการแปลข่าวสารของโพรโตคอล (protocol messages) ระหว่างระบบที่กำลังสื่อสารกันมากมายในเวลาจริง(real time)

3.3 การประยุกต์ใช้ข้อมูลที่ได้มา

เกี่ยวกับการหาวิธีนำข้อมูลที่ได้จากการมอนิเตอร์มาใช้ในฟังก์ชัน(function) ในการจัดการหลาย ๆ ฟังก์ชัน(กล่องแมนเนจเมนท์เอเจนท์ในรูปที่ 3-1) แอพพลิเคชั่นที่ใช้ในการจัดการ (management applications) จะเป็นคนใช้ข้อมูลที่ได้จากการมอนิเตอร์ มันจะกำหนดความต้องการ(เช่นต้องการข้อมูลที่เก็บเป็นสถิติไว้หรือสถานะขณะนั้น) และกำหนดว่าจะมอนิเตอร์อะไร เมื่อไหร่และที่ไหน เป้าหมายของการมอนิเตอร์เครือข่ายคือสามารถจัดการเครือข่ายได้ มันเป็นสิ่งจำเป็นที่จะทำความเข้าใจว่าการจัดการเครือข่ายต้องการอะไรบ้าง การจัดการเครือข่ายจะเกี่ยวข้องกับการวางแผน(planning) การติดตั้ง (installation) และการปฏิบัติงานของส่วนประกอบทั้งหมดในเครือข่ายเพื่อให้ได้มาซึ่งสิ่งที่องค์กรต้องการ ซึ่งความต้องการเหล่านี้สามารถแบ่งได้เป็น 5 กลุ่มดังนี้

1. การจัดการความผิดพลาด (Fault management)

เมื่อมีความผิดปกติเกิดขึ้นกับส่วนประกอบในเครือข่าย ผู้จัดการเครือข่ายจะต้องสามารถหาความผิดพลาดและแก้ไขสถานการณ์นั้นให้ได้อย่างรวดเร็ว ซึ่งบ่อยครั้งที่ไม่สามารถจะแยกแยะปัญหาได้เร็วนักเนื่องมาจากความซับซ้อนของปัญหา ซึ่งหากเป็นเช่นนี้ถึงจะไม่สามารถหาสาเหตุได้ก็จะต้องแก้ปัญหาให้ได้ก่อน ส่วนการวิเคราะห์สาเหตุของปัญหาก็ยังเป็นสิ่งสำคัญเพื่อป้องกันไม่ให้เกิดปัญหาดังกล่าวอีก การตรวจพบความผิดปกตินี้ขึ้นอยู่กับการมอนิเตอร์สถานะของส่วนประกอบในเครือข่าย (network component) สถานะที่ผิดปกติจะถูกบันทึกไว้ว่าเป็นข้อผิดพลาด(errors) ข้อผิดพลาดที่สำคัญ(critical errors) จะถูกส่งไปให้แก่ผู้จัดการเครือข่ายในรูปแบบของการเตือน(alarm) อย่างไรก็ตามเราไม่สามารถตรวจพบความผิดปกติที่ซับซ้อนมากเนื่องมาจากสถานะที่ตัวมันเอง(local) เท่านั้นได้เสมอไป มันจึงเป็นสิ่งจำเป็นที่จะมอนิเตอร์สถานะโดยรวมเพราะเป็นสิ่งที่ทำได้ที่จะมอนิเตอร์ในแลน แต่การที่จะวิเคราะห์ความผิดปกติอย่างอัตโนมัติโดยตัวมอนิเตอร์แบบโกลบอล(global) ยังเป็นสิ่งที่ทำไม่ได้ และนี่คือเหตุผลที่ผู้จัดการเครือข่ายมักจะแก้ข้อผิดพลาดก่อนเพื่อทำให้เครือข่ายสามารถให้บริการได้ตามปกติและจากนั้นจึงค่อยวิเคราะห์หาสาเหตุของความผิดพลาดในภายหลัง

2. การกำหนดค่า (Configuration management)

เป็นงานที่เป็นพื้นฐานของการวิเคราะห์เครือข่าย เพราะเป็นการกำหนดค่าเริ่มต้นและเชื่อมต่อแต่ละส่วนประกอบของเครือข่ายเข้าไว้ด้วยกันเพื่อให้สามารถให้บริการได้ รวมทั้งกำหนดแอดเดรส(address) และชื่อให้แก่ส่วนประกอบเครือข่ายด้วย ซึ่งการกำหนดค่าของเครือข่ายนี้จะไม่เกี่ยวข้องกับการมอนิเตอร์เครือข่ายโดยตรง แต่การวางแผนในการกำหนดค่าต้องการความเข้าใจในความต้องการทั่วไปของการติดต่อและทรัพยากรเครือข่ายซึ่งจะได้มาจากใช้การมอนิเตอร์ช่วย

3. การควบคุมสมรรถภาพ (Performance management)

เกี่ยวข้องกับการปรับเครือข่ายจนเกิดสมรรถภาพดีที่สุด(optimal) ในทรัพยากรเครือข่ายที่มีอยู่ ซึ่งการมอนิเตอร์เครือข่ายจะสามารถช่วยได้ในการมอนิเตอร์และประเมินการสื่อสารในเครือข่าย (network traffic) และจากนั้นก็ทำการเปลี่ยนค่าต่าง ๆ ที่ตั้งไว้ให้เหมาะสมเพื่อทำให้เกิดสมรรถภาพดีขึ้นถ้าจำเป็น

4. การดูแลความปลอดภัย (Security management)

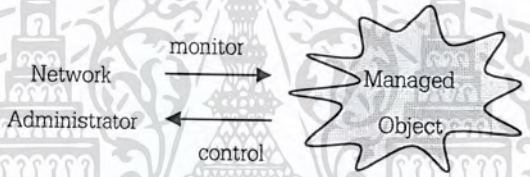
กลไกในการรักษาความปลอดภัยมักจะมีผลกระทบทางลบกับสมรรถภาพ และค่าใช้จ่ายของเครือข่าย การตรวจสอบและดักจับการกระทำที่ก่อให้เกิดความไม่ปลอดภัยในระบบ(security violations) จะอยู่ในรูปแบบของการมอนิเตอร์เครือข่ายซึ่งต้องการฟังก์ชันในการวิเคราะห์ความปลอดภัยและตัวกรองรวมเข้าไปกับฟังก์ชันมอนิเตอร์เพื่อให้ตรวจจับปัญหาความปลอดภัยได้อย่างแม่นยำ

5. การบันทึกการใช้งาน (Accounting management)

ทำหน้าที่บันทึกอัตราการใช้ทรัพยากรเครือข่ายเพื่อควบคุมค่าใช้จ่ายที่เกิดจากการใช้งานเครือข่าย ซึ่งการทำเช่นนี้จะทำให้เกิดโอเวอร์เฮด(overhead) ขึ้นในเครือข่ายนั้นมาก การบันทึกการใช้งานนี้จึงใช้กับเครือข่ายที่สร้างขึ้นมาเพื่อผลประโยชน์ทางธุรกิจเท่านั้น เช่นการเก็บค่าใช้บริการจากผู้ใช้ เป็นต้น

3.4 การเข้าถึงข้อมูลที่มอนิเตอร์(Access to monitor information)

ระบบจัดการเครือข่ายที่ออกแบบมาดีจะมีลักษณะการมองเครือข่ายแบบ นามธรรม(logical) และง่ายต่อการใช้งาน(user-friendly) ซึ่งประกอบไปด้วยอ็อบเจ็คที่ถูกจัดการ(managed object) และสถานะของอ็อบเจ็คเหล่านั้นดังรูปที่ 3-3



รูปที่ 3-3 แบบระดับสูงของอ็อบเจ็ค

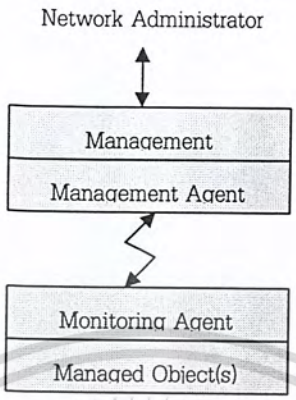
แบบทางฟังก์ชันสำหรับการมอนิเตอร์อ็อบเจ็คที่ถูกจัดการจะประกอบไปด้วย

1. แอปพลิเคชั่นที่ใช้ในการจัดการ(The management application) คือโมดูล(module) ของซอฟต์แวร์(software) ที่ทำหน้าที่ช่วยผู้จัดการเครือข่ายในการมอนิเตอร์เครือข่ายและทำงานเกี่ยวกับการจัดการ

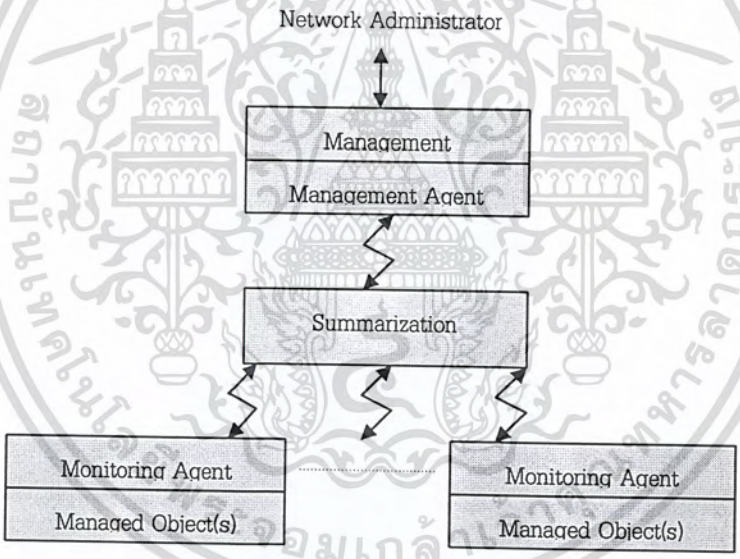
2. แมนเนจเมนท์เอเจนท์(The management agent) จะเข้าถึงข้อมูลของอ็อบเจ็คและจะส่งข้อมูลกลับมายังแมนเนจเมนท์แอปพลิเคชั่นในรูปที่นำไปใช้ประโยชน์ได้

3. มอนิเตอริ่งเอเจนท์(The monitoring agent) เป็นโมดูลของซอฟต์แวร์ที่ทำให้แมนเนจเมนท์เอเจนท์สามารถเข้าถึงข้อมูลของอ็อบเจ็คในเครือข่ายได้โดยมีวิธีพื้นฐานอยู่ 2 วิธีคือ การโพลลิ่ง(polling) และการใช้เหตุการณ์เป็นตัวกระตุ้น(event driven) ซึ่งวิธีการติดต่อระหว่างแมนเนจเมนท์เอเจนท์กับมอนิเตอริ่งเอเจนท์เป็นดังรูปที่ 3-4 และเพื่อความยืดหยุ่นได้(flexibility) และความสามารถในการเปลี่ยนแปลงขนาด(scalability) อาจจะมีมอนิเตอริ่งเอเจนท์อีกตัวซึ่งเรียกว่า ซัมมาไรเซชั่นมอนิเตอริ่งเอเจนท์(summarization monitoring agent) เพื่อเป็นตัวกลางระหว่างมอนิเตอริ่งเอเจนท์กับแมนเนจเมนท์เอเจนท์เดิมโดยเข้าถึงอ็อบเจ็คผ่านมอนิเตอริ่งเอเจนท์ ซึ่งกรณีนี้มอนิเตอริ่งเอเจนท์จะมองมันว่าเป็น

แมนเนจเมนท์เอเจนท์ แต่สำหรับแมนเนจเมนท์เอเจนท์แล้วมันจะถูกมองว่าเป็นมอนิเตอริ่งเอเจนท์ดังรูปที่ 3-5

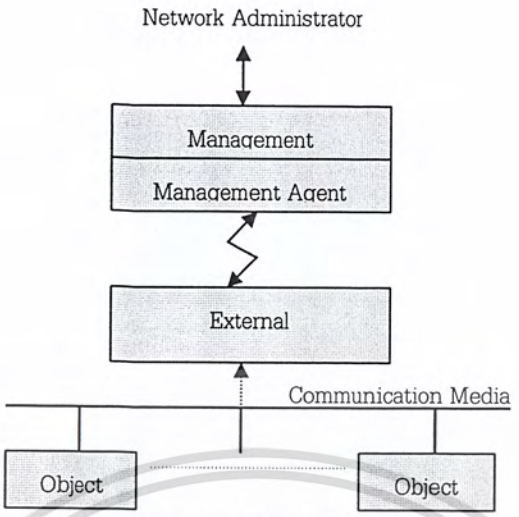


รูปที่ 3-4 แบบของฟังก์ชันพื้นฐานสำหรับการมอนิเตอร์อ็อบเจ็ค



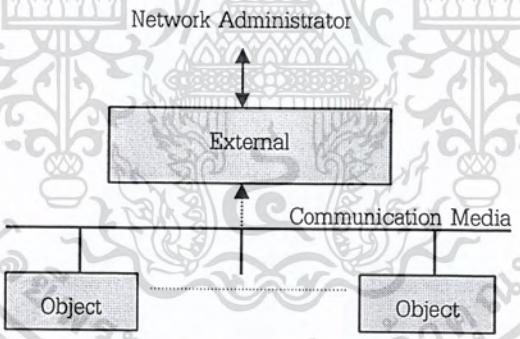
รูปที่ 3-5 ซัมมาไรเซชั่นมอนิเตอริ่งเอเจนท์

มอนิเตอริ่งเอเจนท์ไม่จำเป็นจะต้องได้มาซึ่งข้อมูลโดยการใช้แอคเซสโพรโตคอล (access protocol) ที่เฉพาะเจาะจงเท่านั้น ในความเป็นจริงมันอาจจะใช้วิธีวิเคราะห์ทราฟฟิค(traffic analysis) ซึ่งถ้าใช้วิธีนี้มันจะถูกเรียกว่า เอ็กเทอร์นอลมอนิเตอริ่งเอเจนท์(external monitoring agent) ดังรูปที่ 3-6 เอเจนท์แบบนี้จะไม่ได้อยู่รวมกับอ็อบเจ็ค และวิธีที่มันใช้มอนิเตอร์เครือข่ายก็คือการมอนิเตอร์ทราฟฟิคที่เกิดจากการสื่อสารกันระหว่างอ็อบเจ็คที่ใช้ตัวกลางในการสื่อสารร่วมกัน และเนื่องจากเป็นการวิเคราะห์ทราฟฟิค เอ็กเทอร์นอลมอนิเตอร์จึงทำให้ผู้จัดการเครือข่ายได้รับข้อมูลที่เป็นประโยชน์เกี่ยวกับการจัดการเครือข่าย



รูปที่ 3-6 เอ็กเทอร์นอลมอนิเตอริ่งเอเจนท์

น้อยลง ถ้าเอ็กเทอร์นอลมอนิเตอริ่งเอเจนท์อยู่รวมกับแมนเนจเมนท์แอปพลิเคชั่นและแมนเนจเมนท์เอเจนท์ดังรูปที่ 3-7 จะเรียกว่าเอ็กเทอร์นอลมอนิเตอร์ ซึ่งทำให้ไม่ต้องการโพรโตคอลที่ใช้ในการจัดการ ทำให้เอ็กเทอร์นอลมอนิเตอร์นี้สร้างขึ้นมาได้ง่าย



รูปที่ 3-7 เอ็กเทอร์นอลมอนิเตอร์

3.5 การออกแบบตัวมอนิเตอร์เครือข่าย(The design of network monitors)

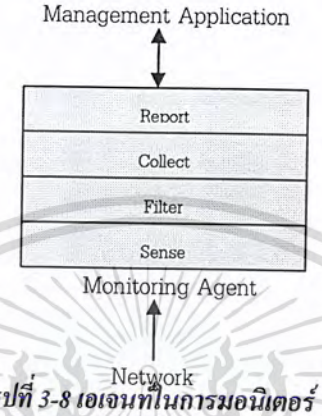
แอปพลิเคชั่นและการนำเอเจนท์ของการมอนิเตอร์เครือข่ายมาใช้อาจจะแตกต่างกัน แต่จะมีฟังก์ชันหลัก ๆ ที่เหมือนกันดังนี้

1. การจับสัญญาณ(Sensing) เป็นการสร้างการติดต่อโดยตรงกับสภาพแวดล้อมทางการสื่อสารและจับเอาข้อมูลขึ้นมา

2. การกรอง(Filtering) เป็นการเอาข้อมูลจำนวนมากที่อยู่ในสตรีมที่ถูกป้อนเข้ามา (input stream) และเลือกไว้เฉพาะที่เราต้องการ

3. การรวบรวม(Collecting) เป็นการเก็บรวบรวมข้อมูลที่ผ่านการกรอง (filter) ซึ่งการเก็บนั้นอาจจะเก็บเป็นการชั่วคราวหรือถาวร หรือจนกระทั่งแมนเนจเมนท์แอปพลิเคชั่นต้องการ

4. การรายงานสรุปผล(Reporting) เป็นการนำข้อมูลที่เก็บไว้มาใช้โดยแมนเนจเมนท์แอปพลิเคชั่นเมื่อเราต้องการ



รูปที่ 3-8 เอเจนท์ในการมอนิเตอร์

3.6 เราควรมอนิเตอร์เครือข่ายที่เลเยอร์ใด (What layer to monitor?)

นอกจากเราจะสามารถแบ่งตัวมอนิเตอร์เครือข่ายเป็นแบบอินทิเกรตเต็ดและเอ็กเทอร์นอล แล้วเรายังสามารถแบ่งตามเลเยอร์(layer) ของเครือข่ายที่มันทำงานอยู่ได้ด้วย

การสื่อสารในเครือข่ายถูกนำมาใช้โดยซอฟต์แวร์ที่ซับซ้อนที่อยู่ในระบบปลาย และระบบที่เชื่อมต่อระบบปลายต่าง ๆ เข้าด้วยกัน(End and Intermediate systems) ซึ่งซอฟต์แวร์เหล่านี้ถูกออกแบบในลักษณะเลเยอร์ซึ่งแต่ละตัวมีโพรโตคอลที่ใช้แตกต่างกันไป รวมทั้งสถานะของมันที่ต้องการการมอนิเตอร์และจัดการ โดยทั่ว ๆ ไปเอเจนท์ที่ทำหน้าที่มอนิเตอร์จะถูกออกแบบให้ทำงานกับเลเยอร์เพียงเลเยอร์เดียว (แต่เป็นไปได้ที่จะทำงานในหลายเลเยอร์) สำหรับตัวมอนิเตอร์แบบอินทิเกรตเต็ด การจะเลือกว่าให้ทำงานบนเลเยอร์ใดนั้นก็ขึ้นอยู่กับฐานข้อมูลที่ใช้จัดการ(Management Information Base: MIB) ที่ถูกกำหนดขึ้นมา สำหรับตัวมอนิเตอร์แบบเอ็กเทอร์นอลจะมีสมรรถนะและได้รับการทำออกมาจำหน่ายมากกว่าเพราะไม่ต้องยุ่งเกี่ยวกับรายละเอียดของแอปพลิเคชั่นทั้งหมด

เราสามารถให้ตัวอย่างของความต้องการที่แตกต่างในการมอนิเตอร์ในเลเยอร์ต่าง ๆ ของเครือข่ายซึ่งจะช่วยให้เข้าใจการออกแบบตัวมอนิเตอร์ได้มากขึ้นดังนี้

1. เลเยอร์ดาต้าลิงค์ (DataLink Layer)

เลเยอร์นี้มีหน้าที่เกี่ยวกับการเคลื่อนย้ายของข้อมูลจริง ๆ จากจุดหนึ่งในเครือข่ายไปยังจุดถัดไป โดยทั่วไปเลเยอร์นี้คือเลเยอร์ต่ำสุดของซอฟต์แวร์เครือข่าย และใกล้กับตัวกลางทางกายภาพ(physical medium) ที่สุด การมอนิเตอร์ในเลเยอร์นี้จะใช้เพื่อตรวจจับข้อผิดพลาดของซอฟต์แวร์และฮาร์ดแวร์ซึ่งเป็นผลให้เกิดการสูญเสียหรือหายไปของข้อมูล ปัญหาที่เกิดขึ้นในเลเยอร์ที่สูงกว่า(upper layer) มักจะถูก

ตรวจจับและแยกแยะโดยการมอนิเตอร์ที่เลเยอร์นี้(datalink layer) เพราะข้อมูลทั้งหมดจะต้องผ่านเลเยอร์นี้

2. เลเยอร์เน็ตเวิร์ค (Network Layer)

เลเยอร์นี้มีหน้าที่ในเครือข่ายย่อย (sub network) เพื่อให้เกิดการเชื่อมต่อปลายต่อปลาย (end-to-end) ของเลเยอร์ทรานสปอร์ต(transport layer) หน้าที่ของเลเยอร์นี้ในสถาปัตยกรรมแบบคอนเน็คชั่นโอเรียนเต็ด(connection-oriented) คือการหาเส้นทาง(route) ให้แพ็คเกจ(packets) ไปสู่ปลายทาง ตัวอย่างของการมอนิเตอร์ในเลเยอร์นี้คือการรายงานเมื่อวงจรใช้ได้(up) หรือใช้ไม่ได้(down) ซึ่งเป็นประโยชน์กับผู้จัดการเครือข่ายในการหลีกเลี่ยงการคับคั่ง(congestion) หรือการสูญเสียการเชื่อมต่อ

3. เลเยอร์ทรานสปอร์ต (Transport Layer)

เลเยอร์นี้จะรับรองว่าการส่งข่าวสารภายในการติดต่อนั้นเชื่อถือได้ การมอนิเตอร์ในเลเยอร์นี้ใช้ในการบันทึกจำนวนการใช้ การทำกิจกรรมภายในเครือข่าย เพราะว่ามันเป็นจุดที่เป็นทางเข้าไปสู่เครือข่ายของแอปพลิเคชั่นทั้งหมด และยังช่วยในจัดการเกี่ยวกับการกำหนดค่าให้กับเครือข่าย(configuration management) และความจุของเครือข่ายโดยการให้ข้อมูลแก่ผู้จัดการเครือข่ายเกี่ยวกับระดับการใช้(level of utilization) ของแต่ละโปรโตคอลในชั้นทรานสปอร์ต

4. เลเยอร์เซสชั่น (Session Layer)

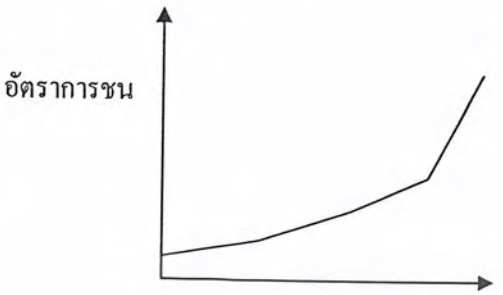
ทำหน้าที่ในการจัดหาฟังก์ชันเพิ่มเติมให้แก่เลเยอร์ทรานสปอร์ตเช่น จัดตั้ง(establish) ตัวตน (identify) ของผู้ใช้(user) และแอปพลิเคชั่น รวมทั้งการทำให้เลเยอร์แอปพลิเคชั่นมองเห็นความเสียหายของการเชื่อมต่อที่เกิดในเลเยอร์ทรานสปอร์ตอย่างชัดเจน การมอนิเตอร์ในชั้นนี้มีเพื่อดูการใช้งาน (workload) บนเครือข่ายทั้งหมดโดยแอปพลิเคชั่นและผู้ใช้ ซึ่งเป็นประโยชน์ในการช่วยทำโหลดบาลานซิ่ง (load balancing) และแอปพลิเคชั่นทางด้านการบันทึกการใช้งาน (Accounting applications)

5. เลเยอร์แอปพลิเคชั่น (Application Layer)

การมอนิเตอร์ในชั้นนี้ มีเพื่อดักจับการเสียหาย(failure) ของโพรเซสของเซิร์ฟเวอร์ในแอบ่พลิเคชั่นแบบไคลเอ็นต์/เซิร์ฟเวอร์(client/server application) โดยการเตือนและแจ้งการเสียหายนี้แก่ผู้จัดการเครือข่าย จะทำให้มีการกู้สถานะเดิมคืนก่อนที่ผู้ใช้แอปพลิเคชั่นจะสังเกตเห็นว่าการทำงานนั้นถูกรบกวน(interrupt)

3.7 ประสิทธิภาพและปัญหาของเน็ตแวร์แลน (Netware LANs Performance and Troubleshooting)

3.7.1 บทบาทของการสื่อสารบนเน็ตเวิร์ค



รูปที่ 3.9 ประสิทธิภาพของอีเทอร์เน็ต

ในการวัดปริมาณการใช้งานนั้น เราวัดขนาดของข้อมูลต่อเวลาเทียบกับแบนด์วิธ (Bandwidth) ซึ่งถ้ามีการใช้งานน้อย นั่นคือน้อยกว่า 5 เปอร์เซ็นต์ของแบนด์วิธทั้งหมด (Total bandwidth) เมื่อมีการใช้งานเพิ่มขึ้นอัตราการชนกัน (Collision rate) ของข้อมูลยิ่งมีมากขึ้น โดยสถิติกล่าวว่าที่ 30 เปอร์เซ็นต์ ประสิทธิภาพการใช้งานของเน็ตเวิร์คจะลดลงอย่างรวดเร็ว

เพราะฉะนั้นสิ่งที่เราต้องวัดเพื่อที่จะตรวจสอบการใช้งานเน็ตเวิร์ค คือ

- การใช้งานของเน็ตเวิร์ค
- การใช้งานสูงสุด (Peak)
- การใช้งานโดยเฉลี่ย

ซึ่งที่กล่าวมานี้จะช่วยในการนำไป รีอาร์เรนท์จิ้ง (rearranging) งานที่ใช้ทรัพยากรของเน็ตเวิร์คสูงได้

ข้อแนะนำ ควรจะวัดทั้งสัปดาห์ โดยเว้นระยะห่างในการจับเป็นครึ่งชั่วโมง

3.7.2 สิ่งที่ควรจะวัดในเน็ตเวิร์คทั่วไป

3.7.2.1 เปอร์เซ็นต์การใช้งานของเน็ตเวิร์ค

ทำได้โดยการวัดทราฟฟิก (traffic) บนเน็ตเวิร์คในช่วงเวลาสั้น ๆ สำหรับเปอร์เซ็นต์การใช้งานของอีเทอร์เน็ตที่ดีต้องไม่เกิน 15 เปอร์เซ็นต์ นอกเหนือจากนี้ ก็จะมีการวัดการชนกัน (collision) ซึ่งมีผลกระทบต่อประสิทธิภาพ

3.7.2.2 เน็ตเวิร์คทรูพุท (Network Throughput)

วัดจำนวนไบต์ทั้งหมดที่ส่งผ่านเน็ตเวิร์คในเวลาใด ๆ ซึ่งจะวัดได้ 2 แบบ คือ

1. ข้อมูลดิบ (raw data) จำนวนไบต์ทั้งหมดที่ได้รับ
2. ข้อมูลที่ใช้จริง (usable data) จำนวนยูเซอร์ดาต้าในแต่ละโปรโตคอล

เนื่องจากโปรโตคอลต่าง ๆ มีโอเวอร์เฮด (overhead) ซึ่งมีผลต่อคาด้าทรูพุท (data throughput) ด้วยเหตุนี้เราจึงนำมาใช้พิจารณาในการออกแบบเน็ตเวิร์คด้วย

ข้อสังเกต อีเทอร์เน็ต 10 เม็กกะบิตต่อวินาที (Megabit per Second) ไม่ใช่หมายถึงให้คาด้าทรูพุทสูงสุด 10 เม็กกะบิตต่อวินาที ในการออกแบบเราคำนึงว่า 2.5 เม็กกะบิตต่อวินาที คือทรูพุทสูงสุด (Maximum throughput)

3.7.2.3 เวลาตอบสนองของโปรเซสไฟล์เซิฟเวอร์ (Response time of the file server peocess)

คือการวัดเวลาตอบสนอง (response) ของไฟล์เซิฟเวอร์ (File Server) ต่อการร้องขอ(request) เป็นการวัดประสิทธิภาพของระบบปฏิบัติการและไฟล์เซิฟเวอร์ที่ทำงานอยู่

3.7.2.4 เน็ตเวิร์คคอนเวอร์เซชั่น (Network Comversation)

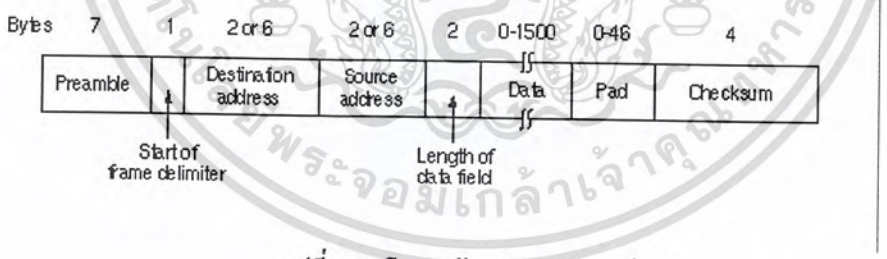
อินดิเซ้นต์ (indecence) และตำแหน่ง (location) ของคอนเวอร์เซชั่น เป็นสิ่งที่สำคัญในการนำมาพิจารณา ถ้าเกิดมีคอขวด (bottlenecks) ขึ้นที่เน็ตเวิร์คฮาร์ดแวร์ เช่น บริดจ์ หรือ เราท์เตอร์

3.7.2.5 เก็บบันทึกข้อผิดพลาดในเน็ตเวิร์ค (recording network error)

เก็บบันทึกข้อผิดพลาดที่เกิดขึ้นในเน็ตเวิร์ค เพื่อที่สามารถดูข้อผิดพลาดที่เกิดขึ้นภายหลังได้

3.7.3 รูปแบบของแพ็กเกจในอีเทอร์เน็ตและประสิทธิภาพ

3.7.3.1 อีเทอร์เน็ตเฟรม (Ethernet frame)



รูปที่ 3.10 โครงสร้างแพ็กเกจอีเทอร์เนต

3.7.3.2 อีเทอร์เนตแอดเดรส (Ethernet Address)

- บรอดคาสท์ (Boardcast)
- มัลติคาสท์ (Multicast)

3.7.3.3 ผลกระทบของขนาดแพ็กเกจกับประสิทธิภาพ

ขนาดของแพ็กเกจยิ่งมีขนาดเล็กเท่าไหร่ โอเวอร์เฮดที่เกิดจากส่วนหัวของโปรโตคอล (Protocol header) ก็ยิ่งมากขึ้นเท่านั้น เช่นเมื่อมีการส่ง ตัวอักษรเพียงตัวเดียวจาก เทอร์มินอล (terminal) จะพบว่า

ข้อมูลที่พบจริงจะเป็น 2 เปอร์เซ็นต์ของเฟรมเท่านั้น แต่ถ้าส่งข้อมูลที่มีขนาดแพ็กเกจสูงสุด พบว่าข้อมูลผู้ใช้คิดเป็น 95 เปอร์เซ็นของแพ็กเกจ จะเห็นว่าขนาดแพ็กเกจเป็นปัจจัยสำคัญของประสิทธิภาพของเน็ตเวิร์ค

ในการส่งข้อมูล ถ้าใช้เฟรมขนาดใหญ่จะช่วยลดจำนวนของแพ็กแกจลง ซึ่งเท่ากับว่าช่วยลดจำนวนการเกิด การชน ดังนั้น โอเวอร์เฮดที่เกิดจาก การชนและการส่งใหม่อีกครั้งจึงลดไปด้วย แต่การเพิ่มขนาดของเฟรมให้ใหญ่ขึ้นก็จะทำให้เกิดผลเสียเช่นกัน ดังนี้ ดูเหมือนเฟรมขนาดใหญ่จะทำให้จำนวนของที่ว่างบนเน็ตเวิร์คลดลงในขณะที่ทราฟฟิกมีการใช้งานสูง และขนาดบัฟเฟอร์ที่ต้องการโดยซอฟท์แวร์ที่ควบคุมเน็ตเวิร์คอินเทอร์เฟสต้องเพิ่มขึ้น

แต่จากการค้นคว้าพบว่าในทางปฏิบัติ ข้อสำคัญที่ใช้พิจารณาคือ คาต้าทรูพุท(ตรงข้ามกับมินิมั่มดีเลย์ (minimum delay) ที่ต้องการในระบบเรียลไทม์ (real-time system) ดังนั้นขนาดเฟรมควรจะมีขนาดใหญ่

3.7.4 สิ่งที่ต้องการทำการวัดประสิทธิภาพ

3.7.4.1 สิ่งที่ต้องทำการวัด สำหรับวัดประสิทธิภาพของเน็ตเวิร์ค

- การใช้งานเน็ตเวิร์ค (Protocol in use)
- การกระจายขนาดของเฟรม (Frame size distribution)
- โหนดที่ใช้งานสูงสุด (Busy modes)
- โหนดที่ไม่ได้ใช้งาน (Idle mode)
- โหนดที่ไม่ได้ตอบสนอง (Unresponsive mode)
- การสนทนาสูงสุด (Busiest Conversation)
- เวลาและระดับการใช้งานสูงสุด (Peak load time and levels)
- เวลาและระดับการใช้งานต่ำสุด (Minimum load time and levels)
- แบนด์วิธที่ใช้ในแต่ละโหนด (Bandwidth usage by node)
- แบนด์วิธที่ใช้ในแต่ละโปรโตคอล (Bandwidth usage by protocol)
- ที่หมดนี้เป็น ข้อกำหนดขั้นต่ำของเครื่องมือในการวัดที่ควรพิจารณา นอกเหนือจากนี้แล้วยังมีอย่างอื่นที่ต้องการวัด ซึ่งมีประโยชน์เมื่อใช้ในการพัฒนาสำหรับประสิทธิภาพของเน็ตเวิร์ค
- โปรโตคอลที่ใช้ในการสนทนา (Protocol use in Conversation)
- กำหนดค่าในการแจ้งเตือนเหตุร้ายที่เกิดขึ้น
- การเตือนเมื่อมีเหตุเกิดขึ้นแก่เน็ตเวิร์ค
- สิ่งสำคัญอย่างอื่นที่ช่วยในการวัดประสิทธิภาพ
- สร้างทราฟฟิกเน็ตเวิร์ค (Generating network traffics) ช่วยให้สามารถจำลองเน็ตเวิร์คได้โดยการสร้างแพ็กเกจและส่งเข้าไปในเน็ตเวิร์ค

- ผลของการแสดงต้องช่วยให้เข้าใจง่าย และสามารถจับเก็บ (Capture) ข้อมูลที่กำหนดเพื่อนำมาวิเคราะห์ภายหลังได้

3.7.4.2 การวัดประสิทธิภาพ

3.7.4.2.1 การใช้งานแบนด์วิธ (Bandwidth usage)

หาค่าเฉลี่ยการใช้งานช่วงเวลาคงที่ (ทุก ๆ 1 นาที หรือ ทุก ๆ 1 วินาที) ทำได้โดยนับจำนวนทราฟฟิกในช่วงเวลาที่กำหนด หารด้วยทรูพุทสูงสุด ค่าที่คำนวณได้จะไม่ต่ำกว่า 5 เปอร์เซ็นต์และพิจารณา ดังนี้

- 10-15 เปอร์เซ็นต์ แสดงว่าเน็ตเวิร์คมีการใช้งานน้อยถึงปานกลาง
- 25 เปอร์เซ็นต์ แสดงว่าเน็ตเวิร์คมีการใช้งานสูงสุด เน็ตเวิร์คแจมมิ่ง(Network jamming) การตอบสนองช้า มีแพ็กเกจเสียมากหรือมีการส่งซ้ำปริมาณมาก

3.7.4.2.2 ข้อผิดพลาดในการส่ง (Transmission Error)

แพ็กเกจที่เกิดการชนกันจะต้องทำการส่งใหม่อีกครั้งหนึ่ง โดยส่วนใหญ่จะถูกรายงานว่า ซีอาร์ซีผิดพลาด หรือแพ็กเกจสั้นขนาดผิดปกติ (runts) ฯลฯ ซึ่งสาเหตุมาจากการชนกัน สำหรับเน็ตเวิร์คที่มีการใช้งานสูง (ประมาณ 20 เปอร์เซ็นต์) อัตราการชนกันที่ยอมรับได้จะอยู่ระหว่าง 1-2 เปอร์เซ็นต์

ถ้าทั้งการใช้งานแบบแบนด์วิธและ อัตราแพ็กเกจเสีย (Failed Packet Rates) สูงที่คู่ หมายความว่าประสิทธิภาพของเน็ตเวิร์คไม่ได้ดีดังที่ควรเป็นซีอาร์ซีผิดพลาดอาจมีสาเหตุมาจากเน็ตเวิร์คอินเทอร์เฟส เช่น เราทราบว่าแต่ละโหนดมีการใช้งานต่ำ ในขณะที่เน็ตเวิร์คมีการใช้งานสูง สาเหตุหลักส่วนใหญ่จะเกิดจากเน็ตเวิร์คอินเทอร์เฟส สาเหตุอย่างอื่นเช่น ตัวเชื่อมต่อ (Connector) หลวม

3.7.4.2.3 การวัดที่จำเป็นสำหรับการออกแบบเพื่อแบ่งเน็ตเวิร์ค (Measurements needed for design to partition network)

ทำได้หลายวิธี ดังนี้

- แบ่งตามชนิดของโปรโตคอลที่ใช้ออกจากกันใช้ ในกรณีที่พีซีแลนด์ (PC Lan) และไมโครคอมพิวเตอร์อยู่ในเน็ตเวิร์คเดียวกัน เป็นวิธีที่ง่ายต่อการนำไปปฏิบัติ แต่ในกรณีที่มีการใช้งานที่หลากหลายอยู่ด้วยกัน ไม่เหมาะสมอย่างยิ่งที่จะแบ่งเป็นโปรโตคอล
- แบ่งตามฟิสิคอลแอดเดรส ใช้ในกรณีที่เรามี เวิร์คกรุ๊ป (Workgroup) ในแต่ละตำแหน่งที่ต้องการติดต่อกับเน็ตเวิร์คเป็นบางครั้ง ส่วนใหญ่จะติดต่อกันเองภายในกลุ่ม การที่เราจะทราบว่าผู้ใช้ใดจะอยู่ในกลุ่มเดียวกัน เราจะต้องเก็บข้อมูลของการสนทนาในแต่ละโหนด

3.7.4.2.4 ขนาดของแพ็กเกจและประสิทธิภาพ (Packet size and performance)

ทรูพุทของดาต้าที่ใช้งานจริง (usable data throughput) กับ เน็ตเวิร์คทรูพุทแตกต่างกัน แอพพลิเคชั่น จะเป็นตัวกำหนดขนาดแพ็กเกจที่เหมาะสม (optimum packet size) เช่นถ้าแอพพลิเคชั่นมีการรับส่งข้อมูลที่ขนาดใหญ่ ซึ่งจำเป็นต้องแบ่งข้อมูลออกเป็นหลายแพ็กเกจ เรากำหนดให้ใช้ขนาดแพ็กเกจสูงสุดของ เน็ตเวิร์คโปรโตคอลมราใช้จะดีที่สุด (1518 ไบต์ในกรณีอีเทอร์เนต) ซึ่งจะช่วยลดจำนวนของแพ็กเกจที่ใช้ในการส่งข้อมูลและช่วยลดผลของโปรโตคอลโอเวอร์เฮดด้วย

ปัจจัยที่เป็นข้อจำกัดของ การกำหนดขนาดของเฟรมคือ โครงสร้างของเน็ตเวิร์คซึ่งการใช้บริดจ์และเราท์เตอร์ จะไม่ยอมให้แพ็กเกจใหญ่ผ่าน (บางที) หรือว่าแอพพลิเคชั่นอื่นต้องการพารามิเตอร์ที่ต่างกันออกไป

ไม่มีกฎตายตัวสำหรับการกำหนดขนาดของเฟรม ทางเดียวก็คือ เมื่อมีการเปลี่ยนแปลงเราต้องคอยดูผลที่เกิดขึ้น ซึ่งที่ได้จากข้อมูลการมอนิเตอร์ริง (Monitoring Information) โดยไม่เพียงแต่ดูเฉพาะส่วนที่มีการเปลี่ยนแปลง เราต้องดูผลที่เกิดกับเน็ตเวิร์คโดยรวมด้วย

3.7.4.2.5 โหนดใช้งานและไม่ใช้งาน (Active and inactive nodes)

โหนดที่ไม่สนองตอบ คือ โหนดที่ได้รับการติดต่อแต่ไม่ตอบสนองต่อการร้องขอ แต่ไม่ได้ส่งการติดต่อ

3.7.4.2.6 กำหนดช่วงเวลาใช้งานสูงสุด (Definition of peak usage time)

เราต้องบันทึกเวลาที่เกิด การใช้งานสูงสุด และการใช้งานน้อยที่สุด เพื่อนำไปทำแผน การทำงาน เช่น ออโตเมติคแบ็คอัพ (Automatic backup) การอัพเดตครั้งใหญ่ (Large batch update) หรือการส่งข้อมูล

มอนิเตอร์ริงแพ็กเกจ (Monitoring Package) จะช่วยได้มาก และยังสามารถทราบทราฟฟิกเพื่อจำลองการทำงานได้

3.8 ภาษาที่ใช้ในการพัฒนาซอฟแวร์

เนื่องจากการใช้งานจะต้องติดต่อกับ การ์ดเชื่อมต่อเน็ตเวิร์ค (Network Interface Card) โดยตรง และเพื่อให้มีประสิทธิภาพมากขึ้นโดนไม่ผ่านขั้นตอนการทำงานของสแต็กโปรโตคอล (Stack Protocol) อื่น ๆ และประกอบกับได้รับตัวโปรแกรมมาจากการพัฒนาต่อเนื่องมาจากรุ่นที่แล้ว จึงเลือกภาษา Virsual C++ ซึ่งต่อการพัฒนาเครื่องมือให้ใช้งานง่าย

ภาษา Virsual C++ เหมาะสำหรับการพัฒนาโปรแกรม บนพื้นฐานระบบวินโดส์ (Windows) และเนื่องจากเคยใช้มาก่อนจึงง่ายต่อการเข้าใจ และการพัฒนาและมีเครื่องมือช่วยในการทำส่วนติดต่อผู้ใช้

การพัฒนาโปรแกรมนั้นจะต้องเรียนแพ็กเกจไดรเวอร์ โดยเรียนผ่าน วินพีแคป (Winpcap) และเพื่อให้โปรแกรมนั้นง่ายต่อการพัฒนาจึงได้แบ่งเป็นโมดูลย่อย ๆ แล้วจึงเรียกผ่านวินพีแคป

บทที่ 4
การวางแผนออกแบบและการสร้างโปรแกรม

4.1 หลักการเบื้องต้นในการสร้าง Software

4.1.1 ศึกษาระบบ (System Study)

ทำการศึกษาระบบโดยพิจารณาถึงตวามต้องการ และศึกษาสิ่งแวดล้อมที่มีความสัมพันธ์กับระบบที่กำลังศึกษาอยู่ เช่น การใช้งานในระบบเครือข่ายจำเป็นต้องใช้อะไรบ้างในการวิเคราะห์ ซึ่ง Software ในปัจจุบันไม่ยืดหยุ่นพอ จึงไม่เหมาะสมกับความต้องการบางรูปแบบ บน Platform วินโดว์ ซึ่งเป็นแบบแอปพิเคชั่นเบสโดยการจัดการแบบวิเคราะห์แบบตั้งค่าและแสดงผล ต้องทำบน คาต้าลิงค์เลเยอร์ เพื่อทำการตรวจสอบข้อมูลทั้งหมดในเครือข่าย

4.1.2 ศึกษาถึงความเป็นไปได้ (Feasibility Study)

ศึกษาความเป็นไปได้ของระบบว่าเราสามารถนำข้อมูลอะไรได้บ้าง ใช้วิเคราะห์อะไรได้บ้าง ถ้าออกแบบเป็น เอเจ็นต์(Agent) จำเป็นจะต้องมีการติดต่อแบบ ไคลเอ็นเซิฟเวอร์ (Client/Server) การรับส่งข้อมูลอาจจะไม่ปลอดภัยจนเป็นเหตุให้ผู้อื่นสามารถได้ข้อมูลที่จะนำไปวิเคราะห์นำไปใช้ เป็นปัญหาของความปลอดภัยแทน แต่ถ้าทำเป็นแอปพลิเคชั่นเบส ทุกอย่างจะสามารถจัดได้ในตัวเดียว ข้อมูลไม่ต้องรับส่งผ่านเน็ตเวิร์คอีก

4.1.3 วิเคราะห์ความต้องการ (Requirement Analysis)

ศึกษาและวิเคราะห์ระบบปัจจุบันที่มีอยู่ พิจารณาข้อดีและข้อเสียของระบบ ทำความเข้าใจระบบให้เป็นอย่างดี และหาความต้องการ (Requirement) ของระบบ อาจจะทำการจำลองระบบต่าง ๆ เพื่อช่วยในการทำความเข้าใจ ได้ดังนี้

- ดักจับข้อมูลเพื่อวิเคราะห์ข้อมูลและกรองข้อมูลบนเน็ตเวิร์คเซกเมนต์
- ดูข้อมูลภายในซับเน็ต (Subnet) หรือแล้วแต่ผู้ใช้จะกำหนดเฉพาะงานที่ใช้
- เปรียบเทียบระหว่างเครื่องหรือกลุ่ม เฉพาะการใช้งานบางอย่างได้ หรือเครื่องที่ผู้ใช้ระบุ
- แสดงการใช้งานในระดับ แอปพลิเคชั่นเลเยอร์ที่ใช้งานเยอะ ๆ หรือเครื่องที่ใช้งานเยอะ ๆ
- แสดงข้อมูลที่ได้วิเคราะห์เป็น แบบกราฟ

4.1.4 กำหนดความต้องการ (Requirement Definition)

เป็นกิจกรรมที่ทำการเปลี่ยนข้อมูลต่าง ๆ ที่รวบรวมได้จากการวิเคราะห์ให้เป็นเอกสารที่กำหนดความต้องการต่าง ๆ ที่ตรงตามที่ผู้ใช้ต้องการ เอกสารต้องเขียนด้วยภาษาที่เข้าใจง่าย ซึ่งผู้ใช้ระดับต้นสามารถอ่านเข้าใจได้ง่าย ดังนี้

- รับการคอนฟิก (Config)
    - IP Number ที่ต้องการ เฉพาะเครื่องหรือบางเครื่องต้องรับการตั้งค่า
    - โปรโตคอล (Protocol) ที่ต้องการเฉพาะเครื่อง
    - พอร์ต (Port) ที่ระบุได้ว่าจะเป็นพอร์ตหมายเลขใด และเป็นขาเข้าหรือออก
- การวิเคราะห์ข้อมูลให้ถูกต้องและทันวลา เช่น ทุก ๆวินาที
- การแสดงผล เช่น วงกลม หรือ กราฟแท่ง ว่าต้องการจะดูอะไรข้อมูลใดในแต่ละแท่ง

4.1.5 กำหนดรายละเอียดความต้องการ (Requirement Specification)

เป็นรายละเอียด และข้อกำหนดของแต่ละความต้องการ อธิบายถึงหน้าที่ความต้องการ ในรายละเอียดที่ลึกลงไปควรเขียนด้วยภาษาที่ไม่มีความกำกวม เพื่อไม่ให้สับสนระหว่างผู้อ่านและผู้เขียน เราจึงได้แบ่งการทำงานออกเป็น 5 ส่วนหลัก ๆ คือ

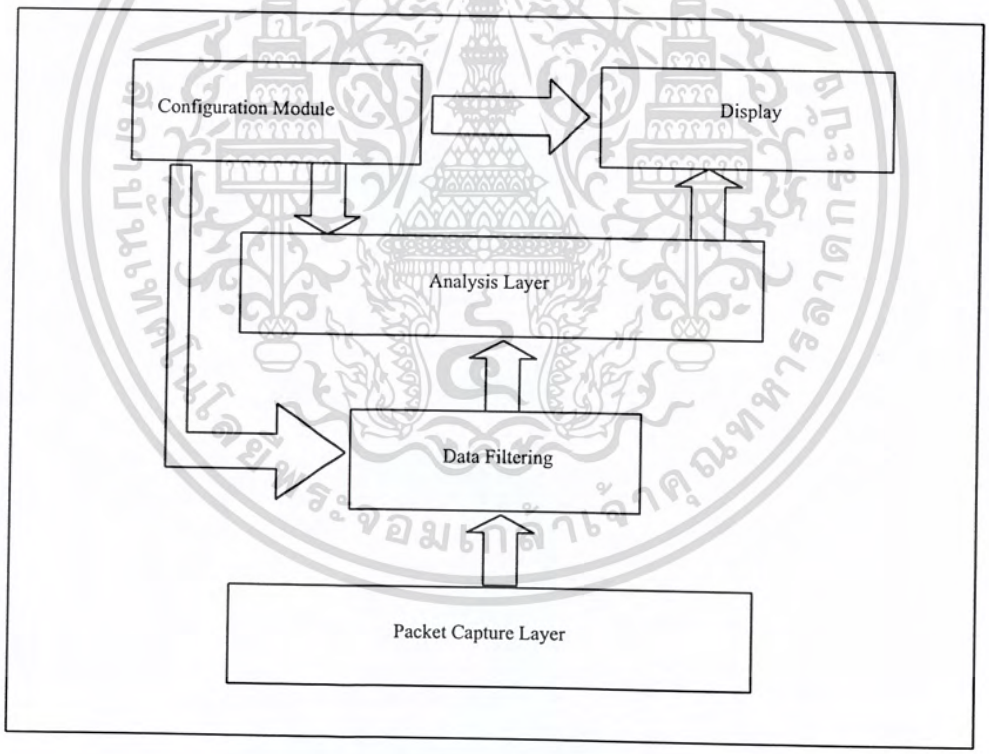
- ส่วนของการควบคุมการดักจับจับข้อมูล
    - Select Interface
    - Start/Stop
    - จัดเก็บข้อมูลลงโครงสร้าง
- ส่วนของการกรองข้อมูล
    - ไอพี (IP)
    - โปรโตคอล (Protocol)
    - พอร์ต (Port)
- วิเคราะห์ข้อมูลประมวลผลได้ทันทุก ๆ แพ็กเก็ต โดยแบ่งข้อมูลออกเป็น แพ็กเก็ต เพื่อแสดงในกรณีกลับมาดูใหม่
- ส่วนของการรับค่า
    - รับค่าไอพี โดยรับเป็น แพทเทอร์ (Pattern) เช่น ww.xx.yy.zz แล้วตามด้วยเครื่องหมายลูกน้ำ (,) เมื่อต้องการเพิ่มไอพีอื่น ๆ อีก และเครื่องหมายลบ (-) เพื่อบอกให้ทราบถึงช่องของ ไอพีได้
    - รับค่าโปรโตคอล (Protocol) หลัก ๆ เช่น IP,TCP,UDP,ICMP,IGMP
    - รับค่าพอร์ต (Port) เช่นรับ IN,OUT,IN or OUT,IN and OUT พร้อมระบุหมายเลขพอร์ตได้
    - การแสดงผล ข้อมูลที่ได้รับแต่ละ แพ็กเก็ต(Packet) นั้นจะเป็นแพ็กเก็ตที่ ถูกกรองข้อมูล เรียบร้อยแล้ว

4.1.6 ออกแบบซอฟแวร์ (Software Design)

ออกแบบซอฟแวร์โดยเลือกภาษาที่ใช้ในการพัฒนา ออกแบบวิธีการที่ใช้ในการเขียนซอฟท์โดยเลือกภาษาที่ใช้ในการพัฒนา ออกแบบวิธีการที่ใช้ในการเขียนซอฟท์แวร์เลือกใช้กลยุทธ์ในการเขียนซอฟท์แวร์ เช่น การเขียนออบเจ็ค (Object-Oriented) เพิ่มรายละเอียดลงไปในข้อกำหนดที่ทำขึ้นในการออกแบบ โครงสร้างข้อมูลที่จะใช้ การออกแบบแบ่งเป็น

- ออกแบบสถาปัตยกรรม (Architectural Design)
- ข้อกำหนดเบื้องต้น (Abstract Specification)
- ออกแบบส่วนติดต่อ (Interface Design)
- ออกแบบส่วนเชื่อมต่อ (Component Design)
- ออกแบบโครงสร้างข้อมูล (Data Structure Design)
- ออกแบบวิธีการทำงาน (Algorithm Design)

4.1.6.1 ออกแบบสถาปัตยกรรม (Architectural Design)



รูปที่ 4.1 โครงสร้างการทำงานของโปรแกรม

- **Configuration Module** รับการตั้งค่าทั้ง Input ที่มาจากการดักจับข้อมูล และ Output ที่เกิดจากการตรวจจับและวิเคราะห์ข้อมูลให้เป็นไปตามความต้องการของผู้ใช้
- **Display** ส่วนจัดการการแสดงผลของข้อมูล เช่นเป็น Report แบบ Text หรือ จะเป็นกราฟ (กราฟแท่งหรือกราฟวงกลม) โดยข้อมูลการแสดงผลจะสอดคล้อง กับการ Configuration
- **Analysis Layer** ส่วนของการวิเคราะห์ข้อมูลที่ได้จากการกรองข้อมูล (Filtering) ให้ตรงตามที่ Output ต้องการ โดยนำข้อมูลมาแยกเป็นจำนวนของข้อมูล และชนิดของข้อมูลเพื่อจะนำไปแสดงผลต่อไป
- **Data Filtering** ส่วนของการคัดแยกเอาเฉพาะข้อมูลที่สนใจ จากการดักจับข้อมูลขึ้นมา เป็นไปตามที่ได้ทำการ Configuration ไว้
- **Packet Capture Layer** ส่วนของการดึงข้อมูลจากเน็ตเวิร์ค ขึ้นมาผ่านทาง Network Interface Card ที่ได้เลือกไว้ ข้อมูลที่ได้จะเป็นข้อมูลแบบดิบที่ยังไม่ได้วิเคราะห์และคัดเลือก โดยอาศัย Library Winpcap เป็นตัวช่วยในการจัดการให้

4.1.6.2 ข้อกำหนดเบื้องต้น (Abstract Specification)

ติดต่อ Library Winpcap ให้ได้ ระบุโหมด Promiscuous เพื่อรับทุกแพ็กเก็ต (Packet) ซึ่งการรับข้อมูลนั้นจะทำแบบ เทรด (Thread) เพื่อความรวดเร็วในการกระจายงานโปรแกรม

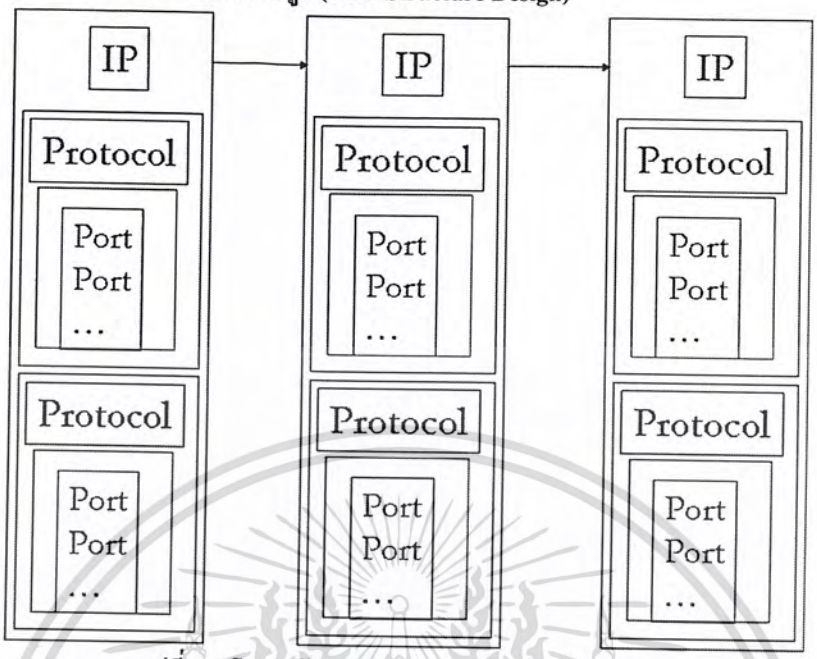
4.1.6.3 ออกแบบส่วนติดต่อ (Interface Design)

ให้ทำเป็นแบบไดอะล็อก (Dialog) เมื่อต้องการตั้งค่า ก็ให้เลือกไดอะล็อก (Dialog) เพื่อทำการตั้งค่า เพื่อจะส่งให้ข้อมูลนั้นไปยังในส่วนของการแสดงผล ให้ผู้ใช้เห็นได้โดยง่าย

4.1.6.4 ออกแบบส่วนเชื่อมต่อ (Component Design)

ติดต่อกับ Winpcap โดยใช้ เทรด (Thread) ควบคุมการหยุดหรือเริ่มการทำงาน ภายในเทรด (Thread) จะต้องมีการวิเคราะห์ข้อมูลจากการดักจับขึ้นมา โดยแอปพลิเคชั่นหลักเป็นตัวรับผลจากการวิเคราะห์ แล้วส่งไปแสดงผล ให้เทรด (Thread) ไปตัวจัดการข้อมูลและจัดเก็บข้อมูลลงใน Structure เพื่อเป็นข้อมูลทางสถิติ

4.1.6.5 ออกแบบโครงสร้างข้อมูล (Data Structure Design)



รูปที่ 4.2 โครงสร้างการทำงานแบบลิงค์ลิสต์ (Linklist)

เป็นลิงค์ลิสต์ (Linklist) ที่เก็บการนับของข้อมูลที่ตรงตามกฎที่ได้ตั้งค่าไว้ ได้แก่ หมายเลขไอพี (IP Number) หมายเลขของเครื่องหรือกลุ่มที่ได้ระบุไว้, โปรโตคอล (Protocol), พอร์ต (Port) โดยจะเก็บเป็นคลาส (Class) ที่มีค่า อินเด็ก (Index) ชี้ที่โหนดถัดไป แล้วถ้าไม่มีก็จะจองพื้นที่ใหม่ โดยภายในมีมัมเบอร์ฟังก์ชั่น (Member Function) เพื่อเรียกค่าของสถิติออกมาแสดงผลต่อไป

4.1.6.6 ออกแบบวิธีการทำงาน (Algorithm Design)

การรับการตั้งค่ากฎจากการคอนฟิก(Config) เช่น การกรองข้อมูลหรือจากการแสดงผลแล้วก็ไปค้นหาใน Structure ว่ามีตรงไหน ถ้ามีก็จะนำผลลัพธ์มารวมกันแล้วนำไปแสดงผลต่อไป เมื่อเริ่มการทำงานจำเป็นต้องติดต่อกับอินเตอร์เฟส (Interface) ก่อน แล้วข้อมูลจาก Winpcap จะถูกอ่านออกมาจากหน่วยความจำ จากนั้นแบ่งแยกข้อมูลออกตามโปรโตคอล (Protocol) ในฟิลด์ต่าง ๆ เพื่อแบ่งแยกประเภทของแพ็กเก็ต (Packet) แล้วการแบ่งนี้ต้องตรงตามโปรโตคอล (Protocol) ด้วย และจะทำการวิเคราะห์ตามโครงสร้างข้อมูลเพื่อจะนำไปเก็บเป็นสถิติลงใน Structure ซึ่งการทำงานทั้งหมดจะอยู่ในเทรด (Thread) โดยแอปพลิชั่น หลักนั้นเป็นเพียงส่วนควบคุมและเป็น ยูสเซอร์อินเตอร์เฟส (User Interface) แสดงผลทุก ๆ ครึ่งวินาที โดยจะแบ่งแยกการทำงานต่างออกเป็นคลาส (Class) แล้วกำหนดมัมเบอร์ (Member) แล้วจึงนำมาเชื่อมต่อกันบน แอปพลิเคชั่นหลัก

4.1.7 จัดทำและทดสอบการใช้งาน (Implementation and Unit Testing)

จัดทำแยกทดสอบโมดูลย่อยที่สร้างขึ้นว่า ทำงานถูกต้องตามต้องการหรือไม่ โดยสามารถทำได้ดังนี้

- ส่วนของการกัดจับแพ็กเก็ต (Packet) ซึ่งต้องอาศัย ไลบารารี Winpcap นั้น ซึ่งต้องขึ้นอยู่กับ แอปพลิเคชั่นอินเตอร์เฟสนี้ จากการทดสอบต้องใช้ เวอร์ชั่น 3.1 beta 4 หรือสูงกว่ามาใช้งาน และการทำการติดตั้งและรับข้อมูล เพื่อนำไปวิเคราะห์ต่อไป ต้องเปิด Promiscuous Mode ก่อน ซึ่งข้อมูลที่ได้เป็น hex Dump ซึ่งจะต้องวิเคราะห์ตามโครงสร้างของโปรโตคอล (Protocol) ข้อมูลที่ได้ก็เปรียบเสมือนการตรวจสอบบน เน็ตเวิร์คโดยการ Tap สามารถทำงานได้ดี

- ส่วนของการวิเคราะห์ข้อมูลถูกประยุกต์เข้ากับการดักจับข้อมูลโดยมีการทำงานภายใน Thread ต้องจัดเก็บข้อมูลลง ไฟล์ และลง Structure ตามที่ได้คอนฟิก(Config)ไว้ โดยอาศัยโครงสร้างโปรโตคอล (Protocol) หลัก ซึ่งถ้าจะทำให้ครบและสามารถเข้าใจทุก โปรโตคอล (Protocol) เลยคงต้องใช้เวลาและทรัพยากรเป็นจำนวนมาก เลยทำตามโปรโตคอลหลัก ๆ ที่พบได้บ่อยก่อน

- ส่วนของการกรองข้อมูลและการตั้งค่า จะเป็นตัวกำหนดโครงสร้างข้อมูลและกำหนดการแสดงผลจำเป็นต้องติดต่อกับหลายส่วนเกิดความยุ่งยากและใช้เวลา โดยเฉพาะการส่งค่าระหว่างคลาส (Class)

- ส่วนของการแสดงผล ให้ผู้ใช้เข้าใจง่ายและเปลี่ยนแปลงรวดเร็วได้ จึงได้ใช้เทคนิคต่าง ๆ เช่น ดิบเบิ้ลบัฟเฟอร์ (Double Buffer) เป็นต้น ซึ่งเป็นผลลัพธ์ของ แอปพลิเคชั่นด้วย ยังจะต้องสอดคล้องกับการตั้งค่า ซึ่งเป็นส่วนที่มีความยืดหยุ่นมาก

4.1.8 รวบรวมและทดสอบระบบ (Integration and System Testing)

เมื่อทำการทดสอบโมดูลย่อย ๆ เสร็จแล้ว จึงทำโมดูลต่าง ๆ มารวมกันเป็นโปรแกรมที่ต้องการแล้วทำการทดสอบการทำงานโดยรวมอีกครั้งหนึ่ง เมื่อรวบรวมคลาส (Class) ต่าง ๆ เข้าด้วยกันโดยแต่ละ โมดูล กมีจะคลาส (Class) ที่ทำหน้าที่เฉพาะแต่การทำงานกับเทรด(Thread) จะติดต่อยากจึงต้องให้เทรด (Thread) เป็นฝ่ายติดต่อโดยส่ง พอยต์เตอร์ (Pointer) โดยการส่งค่าระหว่างคลาส (Class) เมื่อทำสำเร็จก็จะประกอบ โมดูล (Module) เข้าด้วยกัน

- การทดลองตรวจสอบบางเครื่องภายในเครือข่ายเดียวกันเทียบกับการตวจสอบเป็นกลุ่ม ๆ เช่น 161.246.5.9,11-13 TCP Port 80 in และ out แอปพลิเคชั่นก็จะค้นหาข้อมูลดังกล่าวและมีข้อมูลอื่น ๆ ตามที่ได้ตั้งค่าไว้ ในที่นี้เป็นที่นี้เป็น เว็ป(Web) ก็จะรีเทอร์(Return) ค่าออกมาแล้วนำไปแสดงผลบนกราฟ

- ทดสอบบนสภาพแวดล้อมที่มี Traffic หนาแน่น แต่ละโมดูลก็ทำงานหนักโดยเฉพาะการวิเคราะห์แต่ก็ยังสามารถทรงตัวอยู่ได้

4.1.9 ใช้งานและดูแล (Operation and Maintenance)

นำซอฟแวร์นั้นไปใช้งานจริง และบางครั้งอาจมีการเปลี่ยนแปลงเกิดขึ้นจึงต้องมีการดูแล ดังนั้น ซอฟท์แวร์ที่ดีควรออกแบบให้สามารถดูแลได้ง่ายซึ่งสามารถใช้งานและวิเคราะห์ข้อมูลบนสภาพแวดล้อมเสมือนจริงที่ผู้ใช้จะนำไปใช้ เช่น Mirror Port เพื่อพัฒนาโปรแกรมที่จะนำไปใช้งานได้จริงได้

4.2 การวางแผนและพัฒนาระบบ

ขั้นตอนในการวางแผนและพัฒนาระบบมี ดังนี้

- ศึกษาการทำงานของเนื้อหาข้อมูลในส่วนต่าง ๆ ที่ใช้ในการออกแบบโปรแกรมโดยพิจารณาถึงข้อดีข้อเสียของแต่ละส่วนว่าควรจะปรับปรุงอะไร โดยศึกษาถึงการทำงานของโปรแกรมเดิม
- ทำการศึกษาลักษณะการทำงานของโปรแกรมเก่าทั้งหมด และทำการออกแบบโครงสร้างของโปรแกรมใหม่ เพื่อให้เพิ่มประสิทธิภาพในส่วนต่าง ๆ ที่จำเป็นในการใช้งาน
- เลือกถึงวิธีการทำงานของโปรแกรมให้เหมาะสมตามส่วนต่าง ๆ ที่กำหนดไว้คือเลือกพัฒนาโปรแกรมบนภาษา Virsual C++ และใช้เครื่องมือต่าง ๆ

4.3 ส่วนของอัลกอริทึมของโปรแกรม

ช่วงการออกแบบวิธีการทำงานในส่วนต่าง ๆ ซึ่งได้แบ่งเป็น โมดูลหลักแยกจากกันเพื่อง่ายในการพัฒนาโปรแกรมโดยมีส่วนต่าง ๆ มีดังนี้

- การจับข้อมูล
- การกรองแพ็กเกจ
- การวิเคราะห์และแสดงผล

4.3.1 การจับข้อมูล

ส่วนเริ่มต้นของการเก็บแพ็กเกจ โดยจะต้องคำนึงถึงเรื่องการรับแพ็กเกจ ซึ่งมักจะมีปัญหาตรงที่ไม่สามารถรับแพ็กเกจได้ทัน ทำให้ต้องสูญเสียแพ็กเกจไป ดังนั้นควรจะมีบัฟเฟอร์เพื่อสำรองข้อมูลและทำการจัดลำดับ เพื่อให้สามารถได้ค่าที่ใกล้เคียงความจริงมากที่สุด

การใช้งานจะเรียกผ่านวินพีแคป ซึ่งต้องติดต่อผ่านทาง เน็ตเวิร์คอินเทอร์เฟสการ์ด (Network Interface Card) ซึ่งจะผ่านข้อมูลมายังโมดูลต่าง ๆ ที่เรากำหนด แล้วค่อยรวบรวมข้อมูลหรือจัดทำสถิติต่อไป

4.3.2 การกรองแพ็กเกต

เมื่อผ่านการกรองขั้นแรกมาแล้วก็จะทำการเก็บข้อมูลลงในบัฟเฟอร์ ในที่นี้สามารถที่จะเก็บข้อมูลในบัฟเฟอร์ที่เป็นหน่วยความจำ (Memory) หรือเป็นแบบไฟล์ (File) ก็ได้

ถ้าไม่ได้กำหนดการกรองจะถือว่าให้เก็บทุกแพ็กเกจ ซึ่งอาจทำให้บัฟเฟอร์มีขนาดใหญ่และอาจมีข้อมูลที่เราไม่สนใจรวมอยู่ด้วย ในกรณีนี้จะเป็นการยากที่จะดูแพ็กเกจที่สำคัญหรือมีปัญหา เนื่องจากมีข้อมูลมาก ดังนั้นโปรแกรมสามารถจะทำการกรองอีกขั้นหนึ่งเพื่อให้สามารถดูได้เฉพาะแพ็กเกจที่ต้องการ

4.3.2.1 การกรองขั้นแรก

แบ่งเป็น 2 ชนิดด้วยกัน คือ ลักษณะของแพ็กเกจ หรือ ค่าของฟิลด์

4.3.2.1.1 ลักษณะของแพ็กเกจ

ในโปรแกรมสามารถที่จะกรองขั้นแรกได้ซึ่งลักษณะขึ้นอยู่กับรูปแบบแพ็กเกจและความถูกต้องยกตัวอย่าง เช่น

- ทุกแพ็กเกจ
- ทุกแพ็กเกจที่ดี
- ทุกแพ็กเกจที่ผิดพลาด
- แพ็กเกจที่มีขนาดผิดพลาด
- แพ็กเกจที่มีขนาดตามต้องการ

ซึ่งเกณฑ์ดังกล่าวนี้ เป็นอิสระกับชนิดของ โปรโตคอลที่จับได้ เช่น ในเครือข่ายอีเทอร์เน็ต แพ็กเกจที่ดี มีขนาดตามที่ต้องการจะเป็น โปรโตคอลทีซีพี/ไอพี หรือเน็ตแวร์ก็ได้ ดังตาราง แสดงถึงคุณสมบัติแพ็กเกจและความหมาย

ลักษณะแพ็กเกจ	ความหมาย
ทุกแพ็กเกจ	ทุกแพ็กเกจทั้งดีและผิดพลาด
ทุกแพ็กเกจที่ดี	ทุกแพ็กเกจไม่มีข้อผิดพลาด
แพ็กเกจที่มีขนาดผิดพลาด	แพ็กเกจที่มีขนาดน้อยกว่า 64 หรือมากกว่า 1518 ไบต์
แพ็กเกจที่มีขนาดตามต้องการ	แพ็กเกจที่มีขนาดตรงกับที่กำหนดไว้

ตารางที่ 4.1 ตารางแสดงประเภทของการรับแพ็กเกจ

ในการกรองสามารถเลือก ขนาดแพ็กเกจที่ต้องการได้ โดยกำหนดขนาดแพ็กเกจต่ำสุด และสูงสุดดังตัวอย่างต่อไปนี้ ต้องการเก็บข้อมูลซึ่งสั้นกว่าปกติ (แพ็กเกจที่มีขนาดน้อยกว่า 64 ไบต์และ CRC ถูกต้อง) อาจจะเลือกโดยกำหนดดังนี้ คือ เลือกขนาดแพ็กเกจที่มีขนาดน้อยกว่า 64

4.3.2.1.2 ค่าของฟิลด์

ในการกรองแพ็กเกจโดยการกำหนดประเภทส่วนหัว ประเภทของโปรโตคอลระดับเน็ตเวิร์คโปรโตคอลระดับขนส่ง และโปรโตคอลระดับบนที่ต้องการนั้น จะต้องบอกถึงตำแหน่งของข้อมูลของโปรโตคอลที่นำมาใช้ในการกรอง เช่น ต้องการเก็บข้อมูลที่เป็น RIP บนแลนเน็ตเวิร์ค ขั้นแรกก็ต้องระบุว่าใน ฟิลด์ซ็อกเกตของผู้ส่ง (Source Socket field) มีค่า 0x0453 และในฟิลด์ที่อยู่ของผู้รับ (Destination Address field) มีค่า FF-FF-FF-FF-FF-FF

4.3.2.2 การกรองขั้นสุดท้าย

เมื่อคุณใช้การกรองขั้นสุดท้าย (หรือเรียกอีกอย่างว่าการกรองก่อนการแสดงผล) ก่อนที่จะใช้ต้องพิจารณาดูว่า ข้อมูลที่อยู่ในบัฟเฟอร์นั้น มีข้อมูลที่เราต้องการหรือไม่ ดังเช่น คุณต้องการเก็บทุกแพ็กเกจของเน็ตเวิร์คที่ใช้เฟรมแบบ อีเทอร์เนต 802.2 ในบัฟเฟอร์ ซึ่งผลที่ได้อาจจะมากเกินไปจึงต้องทำการกรองเฉพาะแพ็กเกจที่ต้องการ โดยกรองขั้นสุดท้ายเพื่อเอาเฉพาะแพ็กเกจที่เป็นเน็ตแวร์แซฟ (SAP) เป็นต้น

4.3.3 การวิเคราะห์และแสดงผล

ส่วนของการวิเคราะห์ข้อมูลที่ได้จากการ Filtering ให้ตรงตามที่ Output ต้องการ โดยนำข้อมูลมาแยกเป็นจำนวนของข้อมูล และชนิดของข้อมูลเพื่อจะนำไปแสดงผลต่อไป

ส่วนจัดการการแสดงผลของข้อมูล เช่นเป็น Report แบบ Text หรือจะเป็นกราฟ(กราฟแท่งหรือกราฟวงกลม) โดยข้อมูลการแสดงผลจะสอดคล้องกับการ Configuration

4.4 การพัฒนาโปรแกรม (Software Development)

โดยซอฟแวร์ที่ทำการพัฒนาขึ้นจาก Visual C++ เวอร์ชัน 6 ร่วมกับไลบราลี่ WinPCap โดยไลบราลีดังกล่าวจะทำงานอยู่ในชั้นล่างสุดของโปรแกรมการทำงานของไลบราลี จะทำหน้าที่ควบคุมการทำงานของการ์ดอีเธอร์เน็ต การทำงานในส่วนของการเก็บข้อมูลที่ได้จากไลบราลี WinPcap และการวิเคราะห์ข้อมูลจะแยก กันทำงานกันอย่างอิสระในรูปแบบมัลติเธรด โดยส่วนของการเก็บข้อมูลจะนำข้อมูลจากไลบราลีมาเก็บ ในบัฟเฟอร์ของโปรแกรมที่พัฒนาจากโครงสร้างการเก็บข้อมูล แบบลิงส์ลิสต์ เมื่อระบบต้องการวิเคราะห์จะทำการงานข้อมูลจากบัฟเฟอร์ดังกล่าวมาประมวลผล เสมือนว่าบัฟเฟอร์เป็นตัวกลางเชื่อมการ ทำงานระหว่างไลบราลีกับการวิเคราะห์ข้อมูล

4.5 รายละเอียดการทำงานของโปรแกรมแต่ละส่วน

เนื่องจากแต่ละส่วนทำงานแยกจากกันอย่างอิสระ จากความลักษณะการทำงานแบบคลาส ฟังก์ชันการทำงานแต่ละส่วนสามารถแบ่งตามคลาสได้ดังนี้

4.5.1 ควบคุมการทำงานทั้งหมดของระบบ

ทำหน้าที่กำหนดค่าต่างๆ และควบคุมส่วนต่างๆ ให้ทำงานร่วมกัน ในส่วนของการควบคุมจะใช้คลาส CCapturepacketDlg และโกลบอลฟังก์ชั่น โดยจะมีฟังก์ชั่นการทำงานดังต่อไปนี้

- Class Sniffer()

จุดมุ่งหมาย

เป็นส่วนที่ใช้ในเก็บข้อมูล โดยการทำงานในส่วนนี้จะทำงานแยกออกมาจากฟังก์ชั่นหลัก

ขั้นตอนการทำงาน

1. เช็คว่ามีการสั่งหยุดการทำงานหรือยัง
2. สั่งให้ทำการเก็บข้อมูลจากการ์ดแลนด้วย ฟังก์ชั่น PacketFromdevice() ของคลาส Sniffer

- Void CCapturepacketDlg::OnStartCapture()

จุดมุ่งหมาย

เป็นส่วนที่ใช้ในการเริ่มการทำงานของฟังก์ชั่นต่าง ๆ และสั่งให้ฟังก์ชั่นเริ่มการทำงาน

ขั้นตอนการทำงาน

1. ตรวจว่ามีการเปิดโพรมิสคูอัสของการ์ดแลนหรือยัง
2. ทำการเปิดโพรมิสคูอัสของการ์ดแลน
3. สั่งงานให้ class Sniffer() ทำงาน

- Void CCapturePacketDlg::OnStopCapture()

จุดมุ่งหมาย

เป็นส่วนที่ใช้ในการหยุดการทำงานของฟังก์ชั่นต่าง ๆ

ขั้นตอนการทำงาน

1. สั่งให้ Sniffer() หยุดการทำงาน
2. ตรวจว่ามีการเปิดโพรมิสคูอัสของการ์ดแลนหรือยัง
3. ทำการปิดโพรมิสคูอัสของการ์ดแลน

- Void CCapturePacketDlg::OnSelectAdpter()

จุดมุ่งหมาย

เป็นส่วนที่ใช้การเลือก การ์ดอีเทอร์เน็ต

ขั้นตอนการทำงาน

1. สั่งให้ Sniffer() หยุดการทำงาน
2. ตรวจว่ามีการเปิดโพรมิสคูอัสของการ์ดแลนหรือยัง
3. ทำการปิดโพรมิสคูอัสของการ์ดแลน
4. ทำการเลือก การ์ดแลน

- Class CNSChartCtrl

จุดมุ่งหมาย

เป็นส่วนที่ใช้ในการวาดกราฟ โดยการทำงานในส่วนนี้จะทำงานแยกออกมาจากฟังก์ชั่นหลัก

ขั้นตอนการทำงาน

1. เมื่อเริ่มโปรแกรมก็จะทำการวาดกราฟออกมา

4.5.2 Winpcap

WinPcap เป็นไลบรารีที่ทำการติดต่อกับการ์ดแลน เพื่อทำการควบคุมการทำงานของการ์ดแลน ฟังก์ชันที่สำคัญในไลบรารีนี้มีดังนี้

• ULONG PacketGetAdapterNames()

เป็นฟังก์ชั่นแรกที่ใช้ในการติดต่อกับไดร์เวอร์ โดยที่ฟังก์ชั่นนี้จะส่งชื่อของการ์ดแลนที่ถูกติดตั้งอยู่ภายในระบบออกมา

• LPADAPTER PacketOpenAdapter()

เป็นฟังก์ชั่นที่รับชื่อของการ์ดแลนและจะทำการรีเทิร์นพอยต์เตอร์ เกี่ยวกับการ initialized การ์ดแลนโดยจะได้มาจากฟังก์ชั่น PacketGetAdapterNames

• BOOLEAN PacketSetHwFilter()

เป็นฟังก์ชั่นที่ใช้ในการกำหนดการทำงานของการ์ดแลนให้รับแพ็กเก็ต รูปแบบใดมีรายละเอียดดังนี้ Hardware Filter โดยมี parameter ดังนี้

- PACKET_TYPE_PROMISCUOUS
- PACKET_TYPE_DIRECTED
- PACKET_TYPE_BROADCAST
- PACKET_TYPE_MULTICAST
- PACKET_TYPE_ALL_MULTICASTNDIS_PACKET_TYPE_ALL_LOCAL

• BOOLEAN PacketSetBuff()

เป็นฟังก์ชั่นที่กำหนดขนาดของบัฟเฟอร์

• BOOLEAN PacketSetReadTimeout ()

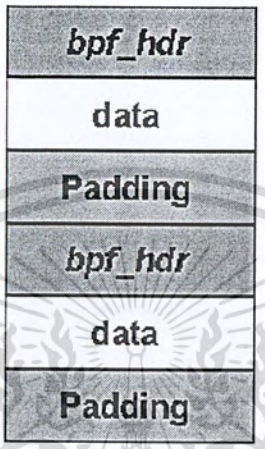
เป็นฟังก์ชั่นที่กำหนดค่าไทมเอาท์ ซึ่งมีหน่วยเป็นมิลลิวินที หากกำหนดเป็น 0 แสดงว่าไม่มีค่าไทม์เอาท์

• LPPACKET PacketAllocatePacket()

กำหนดโครงสร้างแพ็กเก็ต และส่งค่ากลับเป็นพอยต์เตอร์

• BOOLEAN PacketReceivePacket()

เป็นฟังก์ชั่นที่ใช้ในการเก็บข้อมูลขึ้นมาจากการ์ดแลน สำหรับโครงสร้างข้อมูลทำการเก็บใน บัฟเฟอร์มีลักษณะดังรูปที่ 5-1



รูปที่ 4-3 แสดงการเก็บข้อมูลในบัฟเฟอร์ของไลบราลี

• BOOLEAN PacketGetStats()

เป็นฟังก์ชั่นที่ใช้ในการแสดงสถานะของการทำงาน เช่นปริมาณแพ็กเก็ตที่รับได้และเกิดการสูญหาย

• VOID PacketFreePacket()

เป็นฟังก์ชั่นที่ใช้ในการเคลียร์ข้อมูลแพ็กเก็ตในบัฟเฟอร์

• VOID PacketCloseAdapter()

เป็นฟังก์ชั่นที่ใช้สำหรับหยุดการติดต่อกับการ์ดแลน

4.5.3 การเก็บข้อมูล

ในการเก็บข้อมูลมาทำการวิเคราะห์ โปรแกรมจะเก็บข้อมูลส่วนหัว (Header) ของแต่ละแพ็กเก็ตในชั้นต่างๆ เช่นเก็บเฉพาะข้อมูลส่วนหัวของแพ็กเก็ตในชั้นไอพีและทีซีพี ในการเก็บข้อมูลจะใช้คลาส Sniffer เป็นตัวเก็บข้อมุล โดยมีฟังก์ชั่นการทำงานดังต่อไปนี้

• BOOL OpenPromiscuous()

จุดมุ่งหมาย

เพื่อทำการเปิดโพรมิสคูอัสของการ์ดแลน

ขั้นตอนการทำงาน

1. เช็คว่ามีการเปิดโพรมิสคูอัสของการ์ดแลนอยู่หรือไม่
2. ทำการเปิดโพรมิสคูอัสของการ์ดแลนตามค่า NumberDevice

• BOOL ClosePromiscuous()

จุดมุ่งหมาย

เพื่อทำการปิดโพรมิสคูอัสของการ์ดแลน

ขั้นตอนการทำงาน

1. เช็คว่ามีการเปิดโพรมิสคูอัสของการ์ดแลนอยู่หรือไม่
2. ทำการปิดโพรมิสคูอัสของการ์ดแลนที่เปิดอยู่

• BOOL GetStatusPacket()

จุดมุ่งหมาย

เพื่อตรวจสอบว่ามีการรับแพ็กเก็ตเข้ามาเท่าไหร่ แล้วมีการสูญหายไปเท่าไหร่

ขั้นตอนการทำงาน

1. เช็คว่ามีการเปิดโพรมิสคูอัสของการ์ดแลนอยู่หรือไม่
2. ทำการดึงค่าขึ้นมาจาก Winpcap

• BOOL GetPacket()

จุดมุ่งหมาย

เพื่อนำค่าเฮดเดอร์ของแพ็กเก็ตที่เก็บไว้ส่งออกไปให้ฟังก์ชั่นอื่นทีละ 1 แพ็กเก็ต

ขั้นตอนการทำงาน

1. เช็คว่ามีแพ็กเก็ตอยู่ในลิงค์ลิสต์หรือไม่
2. อ่านค่าใน head ของลิงค์ลิสต์ออกมา
3. ลบส่วนหัวของลิงค์ลิสต์ทิ้ง

• BOOL PacketFromDevice()

จุดมุ่งหมาย

เพื่อนำข้อมูลเฮดเดอร์แพ็กเก็ตในการ์ดแลนออกมา แล้วนำมาเข้าลิงค์ลิสต์

ขั้นตอนการทำงาน

1. เช็คว่ามีการเปิดโพรมิสคูอัสของการ์ดแลนอยู่หรือไม่
2. จัดการนำข้อมูลใน buffer ของการ์ดแลนออกมา
3. ทำการแยกออกเป็นแพ็กเก็ตโดยจะแยกเอาเฉพาะส่วนหัวของแพ็กเก็ต
4. นำเข้าไปเก็บในลิงค์ลิสต์โดยที่ 1 โหนด จะเก็บ 1 แพ็กเก็ต

• BOOL ClearSniff()

จุดมุ่งหมาย

เพื่อลบข้อมูล ในลิงค์ลิสต์ทั้งหมด

ขั้นตอนการทำงาน

1. จะทำการลบข้อมูลในลิงค์ลิสต์

• BOOL GetStatusPromiscuous()

จุดมุ่งหมาย

เพื่อเช็คดูว่ามีการเปิดโพรมิสคูอัสของการ์ดแลนหรือไม่

ขั้นตอนการทำงาน

1. เช็คว่ามีการเปิดโพรมิสคูอัสของการ์ดแลนอยู่หรือไม่
2. ส่งผลลัพธ์กลับสู่ฟังก์ชั่นหลักว่าเปิดหรือไม่

4.5.4 การวิเคราะห์ข้อมูล

ในส่วนของการวิเคราะห์ข้อมูล จะใช้คลาสที่ทำหน้าที่วิเคราะห์ข้อมูลดังนี้ คลาส class Packet : public CWinThread เป็นคลาสที่สืบทอดมาจาก CWinThread เป็นThread ที่ใช้ในการวิเคราะห์โดยจะมีการทำงานพื้นฐานที่เกี่ยวกับ Thread เช่น packet->Create Thread(); , packet->SuspendThread(); , packet->ResumeThread(); เป็นต้น

Packet->CreateThread();

จุดมุ่งหมาย

เป็นการสร้าง Thread ขึ้นมาใหม่ ตอนเริ่มต้นการทำงาน

ขั้นตอนการทำงาน

1. สร้างคลาสที่เป็น Thread

Packet->SuspendThread();

จุดมุ่งหมาย

เป็นการหยุดพักการทำงานของ Thread นั้น ชั่วคราวจนกว่าที่จะมีการสั่งให้ทำงานใหม่

ขั้นตอนการทำงาน

1. หยุดการกระทำภายใน Thread นั้นทั้งหมด
2. รอคอยเพื่อจะกลับมาทำงานใหม่

Packet->ResumeThread();

จุดมุ่งหมาย

เพื่อเรียกการทำงานของ Thread ที่ได้ถูกหยุดพักไว้ ขึ้นมาใหม่ให้ทำงานเป็นปกติ

ขั้นตอนการทำงาน

1. เรียนการทำงานของ Thread ที่ระบุกลับขึ้นมาทำงานตามปรกติ

Class FProtocol

จุดมุ่งหมาย

เป็นการระบุโปรโตคอลที่ต้องการวิเคราะห์ โดยส่งไปทำการกรองข้อมูลออกมา

ขั้นตอนการทำงาน

1. รับค่าจากผู้ใช้ แล้วตรวจสอบในโครงสร้าง โปรโตคอลหลัก
2. ส่งค่าเข้าไปทำได้กรองข้อมูลใหม่

Class FIPAddress

จุดมุ่งหมาย

เป็นการระบุหมายเลขเครื่องที่ต้องการวิเคราะห์ โดยส่งไปทำการกรองข้อมูลออกมา

ขั้นตอนการทำงาน

1. รับค่าจากผู้ใช้ แล้วตรวจสอบในโครงสร้าง ของหมายเลขเครื่อง
2. ส่งค่าเข้าไปทำได้กรองข้อมูลใหม่

Class FPort

จุดมุ่งหมาย

เป็นการระบุหมายเลขพอร์ตที่ต้องการวิเคราะห์ โดยส่งไปทำการกรองข้อมูลออกมา

ขั้นตอนการทำงาน

1. รับค่าจากผู้ใช้ แล้วตรวจสอบต้นทางปลายทางของ พอร์ต
2. ส่งค่าเข้าไปทำได้กรองข้อมูลใหม่

บทที่ 5
การทดสอบการทำงาน

5.1 การดำเนินงานในภาคเรียนที่ 1/2547

รายละเอียดและขั้นตอนการดำเนินงานมีดังนี้

1. ศึกษา Algorithm ต่าง ๆ
2. ศึกษาการทำงานที่เกี่ยวข้อง และ กฎเกณฑ์ต่าง ๆ ของ Snort
3. ศึกษาเกี่ยวกับการแสดงผลแบบกราฟ และ รีพอร์ต
4. ศึกษาการเขียนโปรแกรมบน วินโดส์ และ โปรโตคอล ต่าง ๆ
5. ศึกษาการเขียนโปรแกรมติดต่อกับ วินพีแคป และ อินเตอร์เฟสการ์ด
6. ทดลองเขียนโปรแกรมแสดงผลแบบ กราฟ และ รีพอร์ต
7. ทดลองเขียนโปรแกรม ติดต่อกับ อินเตอร์เฟสการ์ด และ วินพีแคป

5.2 ผลการดำเนินงาน

ศึกษาขั้นตอนการทำงานของโปรแกรมต่าง ๆ ที่เกี่ยวข้องกับการดักจับข้อมูล ซึ่งได้เลือกวิธีการแบบ เฝ้าดูภายนอก นั้นคือการดักจับข้อมูลที่วิ่งผ่าน การ์ดเชื่อมต่อเครือโดยตรง ซึ่งยังต้องตรวจสอบถึงความสามารถของ ฮาร์ดแวร์ และ ซอฟแวร์ ว่าสามารถดักจับได้ทันหรือไม่ และศึกษาถึงการทำงานและการเรียกใช้ วินพีแคป กับตัว Microsoft Virsual C++ เวอรชั่น 6 ว่าสามารถที่จะใช้งานร่วมกันได้หรือไม่

5.3 การดำเนินงานในภาคเรียนที่ 2/2547

รายละเอียดและขั้นตอนการดำเนินงานมีดังนี้

1. ศึกษาการออกแบบระบบ
2. เขียนโปรแกรมที่ใช้ในการมอนิเตอร์และวิเคราะห์เครือข่ายที่สามารถตั้งค่าการทำงานได้และใช้งานได้อย่างถูกต้อง
3. พัฒนาโปรแกรมให้ใช้ได้อย่างมีประสิทธิภาพ
4. จัดทำคู่มือการใช้งาน

5.3.1 การออกแบบระบบ

- ออกแบบโครงสร้าง จะทำการออกแบบโครงสร้างของซอฟแวร์ว่าประกอบด้วยส่วนใดบ้าง แต่ละส่วนแบ่งเป็น คลาสย่อย ๆ ใดบ้าง และสร้างคลาสต่าง ๆ ขึ้นเพื่อช่วยในการทำความเข้าใจซอฟแวร์
- ออกแบบรายละเอียด จะแบ่งซอฟแวร์ออกเป็นคลาสย่อย ๆ และทำการออกแบบแต่ละคลาสตามหลัก อ็อบเจคโดนออกแบบบริการต่าง ๆ รวมถึง

การอินเทอร์เฟสระหว่างคลาสด้วย และออกแบบอัลกอริทึม ที่ดีที่สุดแล้ว ออกแบบลงในรายละเอียดของแต่ละคลาส

- ออกแบบส่วนติดต่อผู้ใช้ ออกแบบรูปแบบหน้าจอเพื่ออินเทอร์เฟส กับผู้ใช้ โดยพยายามให้ใช้งานง่ายโดยคำนึงถึงความสะดวกและง่ายต่อการใช้งาน

5.3.2 การพัฒนาและการทดสอบส่วนย่อย

การเขียนโปรแกรมที่ได้ทำการออกแบบไว้ และทำการทดสอบว่าโปรแกรมที่ได้สามารถทำงานได้หรือไม่ และถูกต้องที่กำหนดไว้หรือไม่

5.3.3 รวบรวมและทดสอบระบบ

นำคลาสและส่วนประกอบต่าง ๆ ที่ได้ทำการเขียนและทดสอบว่าถูกต้องและมารวมกัน เป็นโปรแกรมเดียว และทำการทดสอบโปรแกรมนั้นกับระบบจริง

5.3.4 ทำคู่มือประกอบการใช้งาน

บันทึกรายละเอียดและขั้นตอนการติดตั้ง และใช้งานโปรแกรมที่สร้างขึ้นเพื่อเป็นเอกสารอ้างอิงประกอบกับการใช้งาน

5.3.5 ใช้งานและดูแล

เมื่อโปรแกรมทำงานได้อย่างถูกต้องแล้ว ก็จะถูกนำไปติดตั้งใช้งานจริง และในอนาคตหากมี การเปลี่ยนแปลงความต้องการบางอย่างก็จะต้องทำการแก้ไขพัฒนาโปรแกรมให้ได้ตามต้องการ

5.4 การทดสอบประสิทธิภาพของระบบ

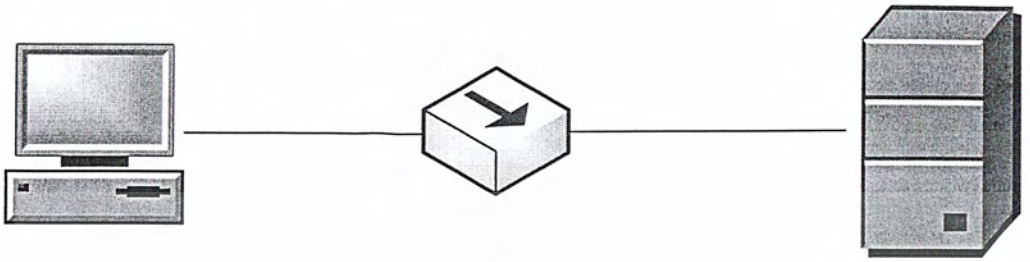
ขั้นตอนการทดสอบโปรแกรมมอนิเตอร์และวิเคราะห์เครือข่ายได้ทำการทดสอบโดยมีรายละเอียดของคอมพิวเตอร์ที่ใช้ทดสอบดังนี้

เครื่องที่ติดตั้งโปรแกรม IsagNetMon

- หน่วยประมวลผลความเร็ว 500 Mhz
- หน่วยความจำหลัก 256 MB
- ระบบปฏิบัติการไมโครซอฟท์วินโดว์ 2000
- การ์ดแลนความเร็ว 100 Mb

5.5 โครงสร้างทางเครือข่ายของระบบทดสอบ

ระบบเครือข่ายที่ใช้ทดสอบในบทนี้ เป็นระบบที่ทำงานบนการ์ดแลนและอุปกรณ์ที่มีความเร็วทั้งระบบ 100 เมกกะบิต มีลักษณะดังรูป



รูปที่ 5-1 แสดงโครงสร้างเครือข่ายในการทดสอบ

5.6 ปัญหาและอุปสรรคที่พบในขณะปฏิบัติงาน

1. ขาดประสบการณ์ในการเขียนโปรแกรมขนาดใหญ่ ๆ ที่ถูกนำไปใช้งานจริง
2. มีปัญหาเรื่องการประมวณผลข้อมูลไม่ทันกับแพ็กเกจที่เข้ามาได้ทำให้บางครั้งค่าได้ออกมาจากการคำนวณอาจจะคลาดเคลื่อนไปจากค่าที่เป็นจริงไปบ้าง ซึ่งในปฏิบัติจริง ๆ แล้วจะใช้ฮาร์ดแวร์เพื่อนวิเคราะห์แทน
3. ขาดประสบการณ์ในการทำส่วนติดต่อแบบกราฟฟิก จึงทำให้เสียเวลาในการเขียนโปรแกรมมากกว่าการเขียนโปรแกรมอย่างอื่น
4. มีปัญหาเกี่ยวกับการเก็บข้อมูลของตัวแปรประเภทต่าง ๆ ทำให้เสียเวลาในการหาข้อผิดพลาด

บทที่ 6
บทวิจารณ์และสรุป

6.1 วิจารณ์โครงงาน

1. การพัฒนาโปรแกรม

ในการเขียนโปรแกรมภาษา Virsual C++ นั้นมีใกล้เคียงกับการเขียนภาษา C++ มาก แต่บางคำสั่งที่ภาษา Virsual C++ มีมากกว่าภาษา C++ ก็ยังต้องการการเปิดคู่มืออ้างอิง หรืออ่านระบบช่วยเหลือของคอมไพเลอร์อยู่บ้าง แต่ก็พบปัญหาซึ่งไม่สามารถหาได้จากคู่มืออ้างอิงหรือระบบช่วยเหลือของคอมไพเลอร์ คือ การเขียนฟังก์ชันในการอินเทอร์รัพท์ในรูปแบบของอ็อบเจ็คโอเรียนเต็ด ซึ่งผู้พัฒนาคิดว่าต้องอยู่ในอ็อบเจ็คนั้น แต่ในความเป็นจริงแล้วต้องเขียนแยกไว้ในไฟล์หลัก(ไฟล์ที่มีฟังก์ชัน main()) อีกทั้งการคอมไพล์เป็นเม็มโมรีโมเดลแบบต่าง ๆ ก็ยังเข้าใจไม่ลึกซึ้งทำให้อาจจะเลือกรูปแบบการคอมไพล์ไม่เหมาะสมนัก และการเขียนโปรแกรมเพื่อให้สามารถรวมเฮดเดอร์ได้นั้น ควรเขียนประกาศคลาสแยกไว้คนละไฟล์กับส่วนที่แสดงการทำงานของเมทธอดต่าง ๆ ของคลาส และสุดท้ายการพัฒนาโปรแกรมเพื่อให้ออปติไมซ์ที่สุดนั้นก็ยังทำได้ไม่ดีพอแต่ก็ได้พยายามทำให้ประหยัดทรัพยากรที่สุดแล้ว

2. การกรองแพ็คเกจ

ในตอนแรกที่จะทำการพัฒนาส่วนกรองแพ็คเกจนั้น ก็ไม่แน่ใจว่าในความเป็นจริงแล้ว แพ็คเกจที่วิ่งอยู่ในเน็ตเวิร์คมีการเรียงลำดับไบต์อย่างไร แต่ในภายหลังก็ค้นพบว่าในอีเธอร์เน็ตนั้นการเรียงลำดับไบต์เป็นแบบง่าย ๆ คือเรียงจากบิตสูงไปหาบิตต่ำในไบต์เดียวกัน และจากไบต์ลำดับต่ำไปหาไบต์ลำดับสูงในแพ็คเกจเดียวกัน

6.2 สรุปผลโครงงาน

ถึงแม้ว่าโครงงานแนวนี้จะเคยมีผู้พัฒนามาก่อนแล้ว แต่ในโครงงานก็เป็นการพัฒนาใหม่เองทั้งหมด ทั้งในส่วนการพัฒนาโปรแกรมซึ่งได้นำเทคโนโลยีอ็อบเจ็คโอเรียนเต็ดมาใช้ ส่วนกรองแพ็คเกจซึ่งเขียนมาจากการศึกษาทฤษฎี เป็นอัลกอริทึมที่คิดขึ้นมาเอง ไม่ได้ศึกษาจากโปรแกรมที่มีผู้เขียนไว้แล้ว อีกทั้งผู้พัฒนาโครงงานนี้ก็ไม่มีความคุ้นเคยและไม่มีประสบการณ์ ในการใช้เทคโนโลยีอ็อบเจ็คโอเรียนเต็ด จึงอาจมีปัญหาเกิดขึ้นบ้างแต่ก็ผ่านไปได้ด้วยดี ทำให้ได้รับความรู้จากการทำโครงงานชิ้นนี้เป็นอย่างมาก ผู้จัดทำโครงงานยังได้เรียนรู้การทำงานร่วมกันเป็นทีม ได้รู้ถึงจุดบกพร่องของตนเอง ซึ่งนับว่าเป็นประโยชน์ในการปรับปรุงตัว เพื่อนำไปใช้ในชีวิตการทำงานจริง ๆ ได้ดียิ่ง ๆ ขึ้นไป

6.3 ข้อเสนอแนะและแนวทางการพัฒนาต่อไป

1. ควรพัฒนาให้โปรแกรมรู้จักกับโพรโตคอลที่หลากหลายกว่านี้ เพราะยิ่งรู้จักโพรโตคอลมากขึ้นเท่าใดก็จะยิ่งเกิดประโยชน์กับผู้ใช้มากขึ้นเท่านั้น

2. ไม่ควรพัฒนาโปรแกรมโดยเริ่มต้นใหม่เองทั้งหมด ถ้าจะทำโปรแกรมประเภทมอนิเตอร์เครือข่ายเช่นเดียวกับที่โครงงานนี้ได้ทำแล้ว ควรทำต่อจากที่มีอยู่เพราะจะทำให้ประหยัดเวลา ทำให้มีโปรแกรมที่มีประสิทธิภาพมากยิ่งขึ้นไปเป็นลำดับ

3. ถ้าต้องการจะพัฒนาใหม่โดยใช้ภาษาอื่น ภาษาจาวาก็เป็นภาษาที่น่าสนใจ เพราะเขียนครั้งเดียวรันได้หลายแพลตฟอร์ม

ภาคผนวก ก

หมายเลขพอร์ตที่รู้จักกันดี(Well-known ports)

Keyword	*Decimal*	*Description*
-	0/tcp/udp	Reserved
tcpmux	1/tcp/udp	TCP Port Service Multiplexer
compressnet	2/tcp/udp	Management Utility
compressnet	3/tcp/udp	Compression Process
rje	5/tcp/udp	Remote Job Entry
echo	7/tcp/udp	Echo
discard	9/tcp/udp	Discard
systat	11/tcp/udp	Active Users
daytime	13/tcp/udp	Daytime
qotd	17/tcp/udp	Quote of the Day
msp	18/tcp/udp	Message Send Protocol
chargen	19/tcp/udp	Character Generator
ftp-data	20/tcp/udp	File Transfer [Default Data]
ftp	21/tcp/udp	File Transfer [Control]
telnet	23/tcp/udp	Telnet
-	24/tcp/udp	Any private mail system
smtp	25/tcp/udp	Simple Mail Transfer
nsw-fe	27/tcp/udp	NSW User System FE
msg-icp	29/tcp/udp	MSG ICP
msg-auth	31/tcp/udp	MSG Authentication
dsp	33/tcp/udp	Display Support Protocol
-	35/tcp/udp	Any private printer server
time	37/tcp/udp	Time
rap	38/tcp/udp	Route Access Protocol
rlp	39/tcp/udp	Resource Location Protocol
graphics	41/tcp/udp	Graphics
nameserver	42/tcp/udp	Host Name Server
nicname	43/tcp/udp	Who Is
mpm-flags	44/tcp/udp	MPM FLAGS Protocol

Keyword	*Decimal*	*Description*
mpm	45/tcp/udp	Message Processing Module [recv]
mpm-snd	46/tcp/udp	MPM [default send]
ni-ftp	47/tcp/udp	NI FTP
auditd	48/tcp/udp	Digital Audit Daemon
login	49/tcp/udp	Login Host Protocol
re-mail-ck	50/tcp/udp	Remote Mail Checking Protocol
la-maint	51/tcp/udp	IMP Logical Address Maintenance
xns-time	52/tcp/udp	XNS Time Protocol
domain	53/tcp/udp	Domain Name Server
xns-ch	54/tcp/udp	XNS Clearinghouse
isi-gl	55/tcp/udp	ISI Graphics Language
xns-auth	56/tcp/udp	XNS Authentication
-	57/tcp/udp	Any private terminal access
xns-mail	58/tcp/udp	XNS Mail
-	59/tcp/udp	Any private file service
-	60/tcp/udp	Unassigned
ni-mail	61/tcp/udp	NI MAIL
acas	62/tcp/udp	ACA Services
covia	64/tcp/udp	Communications Integrator(CI)
covia	64/udp/udp	Communications Integrator(CI)
tacacs-ds	65/tcp/udp	TACACS-Database Service
aql*net	66/tcp/udp	Oracle SQL *NET
bootps	67/tcp/udp	Bootstrap Protocol Server
bootpc	68/tcp/udp	Bootstrap Protocol Client
tftp	69/tcp/udp	Trivial File Transfer
gopher	70/tcp/udp	Gopher
netrjs-1	71/tcp/udp	Remote Job Service
netrjs-2	72/tcp/udp	Remote Job Service
netrjs-3	73/tcp/udp	Remote Job Service
netrjs-4	74/tcp/udp	Remote Job Service
-	75/tcp	Any private dial out service
deos	76/tcp/udp	Distributed External Object Store

Keyword	*Decimal*	*Description*
-	77/tcp/udp	Any private RJE service
vettcp	78/tcp/udp	vettcp
finger	79/tcp/udp	Finger
www-http	80/tcp/udp	World Wide Web HTTP
hosts2-ns	81/tcp/udp	HOSTS2 Name Server
xfer	82/tcp/udp	XFER Utility
mit-ml-dev	83/tcp/udp	MIT ML Device
ctf	84/tcp/udp	Common Trace Facility
mit-ml-dev	85/tcp/udp	MIT ML Device
mfcobol	86/tcp/udp	Micro Focus Cobol
-	87/tcp/udp	Any private terminal link
kerberos	88/tcp/udp	Kerberos
su-mit-tg	89/tcp/udp	SU/MIT Telnet Gateway
dnsix	90/tcp/udp	DNSIX Security Attribute Token Map
mit-dov	91/tcp/udp	MIT Dover Spooler
npp	92/tcp/udp	Network Printing Protocol
dcp	93/tcp/udp	Device Control Protocol
objcall	94/tcp/udp	Tivoli Object Dispatcher
supdup	95/tcp/udp	SUPDUP
dixie	96/tcp/udp	DIXIE Protocol Specification
swift-rvf	97/tcp/udp	Swift Remote Virtual File Protocol
tacnews	98/tcp/udp	TAC News
metagram	99/tcp/udp	Metagram Relay
newacct	100/tcp	[unauthorized use]
hostname	101/tcp/udp	NIC Host Name Server
iso-tsap	102/tcp/udp	ISO-TSAP
gppitnp	103/tcp/udp	Genesis Point-to-Point Trans Net
acr-nema	104/tcp/udp	ACR-NEMA Digital Imag. & Comm. 300
csnet-ns	105/tcp/udp	Mailbox Name Nameserver
3com-tsmux	106/tcp/udp	3COM-TSMUX
rtelnet	107/tcp/udp	Remote Telnet Service
snagas	108/tcp/udp	SNA Gateway Access Server

Keyword	*Decimal*	*Description*
pop2	109/tcp/udp	Post Office Protocol – Version 2
pop3	110/tcp/udp	Post Office Protocol – Version 3
sunrpc	111/tcp/udp	SUN Remote Procedure Call
mcidas	112/tcp/udp	McIDAS Data Transmission Protocol
auth	113/tcp/udp	Authentication Service
audionews	114/tcp/udp	Audio News Multicast
sftp	115/tcp/udp	Simple File Transfer Protocol
ansanotify	116/tcp/udp	ANSA REX Notify
uucp-path	117/tcp/udp	UUCP Path Service
sqlserv	118/tcp/udp	SQL Services
nntp	119/tcp/udp	Network News Transfer Protocol
cfdptkt	120/tcp/udp	CFDPTKT
erpc	121/tcp/udp	Encore Expedited Remote Pro.Call
smakynet	122/tcp/udp	SMAKYNET
ntp	123/tcp/udp	Network Time Protocol
ansatrader	124/tcp/udp	ANSA REX Trader
locus-map	125/tcp/udp	Locus PC-Interface Net Map Ser
unitary	126/tcp/udp	Unisys Unitary Login
locus-con	127/tcp/udp	Locus PC-Interface Conn Server
gss-xlicen	128/tcp/udp	GSS X License Verification
pwdgen	129/tcp/udp	Password Generator Protocol
cisco-fna	130/tcp/udp	cisco FNATIVE
cisco-tna	131/tcp/udp	cisco TNATIVE
cisco-sys	132/tcp/udp	cisco SYSMAINT
statsrv	133/tcp/udp	Statistics Service
ingres-net	134/tcp/udp	INGRES-NET Service
loc-srv	135/tcp/udp	Location Service
profile	136/tcp/udp	PROFILE Naming System
netbios-ns	137/tcp/udp	NETBIOS Name Service
netbios-dgm	138/tcp/udp	NETBIOS Datagram Service
netbios-ssn	139/tcp/udp	NETBIOS Session Service
emfis-data	140/tcp/udp	EMFIS Data Service

Keyword	*Decimal*	*Description*
emfis-cntl	141/tcp/udp	EMFIS Control Service
bl-idm	142/tcp/udp	Britton-Lee IDM
imap2	143/tcp/udp	Interim Mail Access Protocol v2
news	144/tcp/udp	NewS
uaac	145/tcp/udp	UAAC Protocol
iso-tp0	146/tcp/udp	ISO-IP0
iso-ip	147/tcp/udp	ISO-IP
cronus	148/tcp/udp	CRONUS-SUPPORT
aed-512	149/tcp/udp	AED 512 Emulation Service
sql-net	150/tcp/udp	SQL-NET
hems	151/tcp/udp	HEMS
bftp	152/tcp/udp	Background File Transfer Program
sgmp	153/tcp/udp	SGMP
netsc-prod	154/tcp/udp	NETSC
netsc-dev	155/tcp/udp	NETSC
sqlsrv	156/tcp/udp	SQL Service
knet-cmp	157/tcp/udp	KNET/VM Command/Message Protocol
pcmail-srv	158/tcp/udp	PCMail Server
nss-routing	159/tcp/udp	NSS-Routing
sgmp-traps	160/tcp/udp	SGMP-TRAPS
snmp	161/tcp/udp	SNMP
snmptrap	162/tcp/udp	SNMPTRAP
cmip-man	163/tcp/udp	CMIP/TCP Manager
cmip-agent	164/tcp/udp	CMIP/TCP Agent
xns-courier	165/tcp/udp	Xerox
s-net	166/tcp/udp	Sirius Systems
namp	167/tcp/udp	NAMP
rsvd	168/tcp/udp	RSVD
send	169/tcp/udp	SEND
print-srv	170/tcp/udp	Network PostScript
multiplex	171/tcp/udp	Network Innovations Multiplex
cl/1	172/tcp/udp	Network Innovations CL/1

Keyword	*Decimal*	*Description*
xyplex-mux	173/tcp/udp	Xyplex
mailq	174/tcp/udp	MAILQ
vmnet	175/tcp/udp	VMNET
genrad-mux	176/tcp/udp	GENRAD-MUX
xdmcp	177/tcp/udp	X Display Manager Control Protocol
nextstep	178/tcp/udp	NextStep Window Server
bgp	179/tcp/udp	Border Gateway Protocol
ris	180/tcp/udp	Intergraph
unify	181/tcp/udp	Unify
audit	182/tcp/udp	Unisys Audit SITP
ocbinder	183/tcp/udp	OCBinder
ocserver	184/tcp/udp	OCServer
remote-kis	185/tcp/udp	Remote-KIS
kis	186/tcp/udp	KIS Protocol
aci	187/tcp/udp	Application Communication Interface
mumps	188/tcp/udp	Plus Five's MUMPS
qft	189/tcp/udp	Queued File Transport
gacp	190/tcp/udp	Gateway Access Control Protocol
prospero	191/tcp/udp	Prospero Directory Service
osu-nms	192/tcp/udp	OSU Network Monitoring System
srmp	193/tcp/udp	Spider Remote Monitoring Protocol
irc	194/tcp/udp	Internet Relay Chat Protocol
dn6-nlm-aud	195/tcp/udp	DNSIX Network Level Module Audit
dn6-smm-red	196/tcp/udp	DNSIX Session Mgt Module Audit Redir
dls	197/tcp/udp	Directory Location Service
dls-mon	198/tcp/udp	Directory Location Service Monitor
smux	199/tcp/udp	SMUX
src	200/tcp/udp	IBM System Resource Controller
at-rtmp	201/tcp/udp	AppleTalk Routing Maintenance
at-nbp	202/tcp/udp	AppleTalk Name Binding
at-3	203/tcp/udp	AppleTalk Unused
at-echo	204/tcp/udp	AppleTalk Echo

Keyword	*Decimal*	*Description*
at-5	205/tcp/udp	AppleTalk Unused
at-zis	206/tcp/udp	AppleTalk Zone Information
at-7	207/tcp/udp	AppleTalk Unused
at-8	208/tcp/udp	AppleTalk Unused
tam	209/tcp/udp	Trivial Authenticated Mail Protocol
z39.50	210/tcp/udp	ANSI Z39.50
914c/g	211/tcp/udp	Texas Instruments 914C/G Terminal
anet	212/tcp/udp	ATEXSSTR
ipx	213/tcp/udp	IPX
vmpwscs	214/tcp/udp	VM PWSCS
softpc	215/tcp/udp	Insignia Solutions
atls	216/tcp/udp	Access Technology License Server
dbase	217/tcp/udp	dBASE Unix
mpp	218/tcp/udp	Netix Message Posting Protocol
uarps	219/tcp/udp	Unisys ARPs
imap3	220/tcp/udp	Interactive Mail Access Protocol v3
fln-spx	221/tcp/udp	Berkeley rlogind with SPX auth
rsh-spx	222/tcp/udp	Berkeley rshd with SPX auth
cdc	223/tcp/udp	Certificate Distribution Center
sur-meas	243/tcp/udp	Survey Measurement
link	245/tcp/udp	LINK
dsp3270	246/tcp/udp	Display Systems Protocol
pdap	344/tcp/udp	Prospero Data Access Protocol
pawserv	345/tcp/udp	Perf Analysis Workbench
zserv	346/tcp/udp	Zebra server
fatserv	347/tcp/udp	Fatmen Server
csi-sgwp	348/tcp/udp	Cabletron Management Protocol
clearcase	371/tcp/udp	Clearcase
ulistserv	372/tcp/udp	Unix Listserv
legent-1	373/tcp/udp	Legent Corporation
legent-2	374/tcp/udp	Legent Corporation
hassle	375/tcp/udp	Hassle

Keyword	*Decimal*	*Description*
nip	376/tcp/udp	Amiga Envoy Network Inquiry Protocol
tnETOS	377/tcp/udp	NEC Corporation
dsETOS	378/tcp/udp	NEC Corporation
is99c	379/tcp/udp	TIA/EIA/IS-99 modem client
is99s	380/tcp/udp	TIA/EIA/IS-99 modem server
hp-collector	381/tcp/udp	hp performance data collector
hp-managed-node	382/tcp/udp	hp performance data managed node
hp-alarm-mgr	383/tcp/udp	hp performance data alarm manager
arns	384/tcp/udp	A Remote Network Server System
ibm-app	385/tcp/udp	IBM Application
asa	386/tcp/udp	ASA Message Router Object Def.
aurp	387/tcp/udp	Appletalk Update-Based Routing Pro.
unidata-ldm	388/tcp/udp	Unidata LDM Version 4
ldap	389/tcp/udp	Lightweight Directory Access Protocol
uis	390/tcp/udp	UIS
synotics-relay	391/tcp/udp	SynOptics SNMP Relay Port
synotics-broker	392/tcp/udp	SynOptics Port Broker Port
dis	393/tcp/udp	Data Interpretation System
embl-ndt	394/tcp/udp	EMBL Nucleic Data Transfer
netcp	395/tcp/udp	NETscout Control Protocol
netware-ip	396/tcp/udp	Novell Netware over IP
mptn	397/tcp/udp	Multi Protocol Trans. Net.
kryptolan	398/tcp/udp	Kryptolan
work-sol	400/tcp/udp	Workstation Solutions
ups	401/tcp/udp	Uninterruptible Power Supply
genie	402/tcp/udp	Genie Protocol
decap	403/tcp/udp	decap
nced	404/tcp/udp	nced
ncld	405/tcp/udp	ncld
imsp	406/tcp/udp	Interactive Mail Support Protocol
timbuktu	407/tcp/udp	Timbuktu

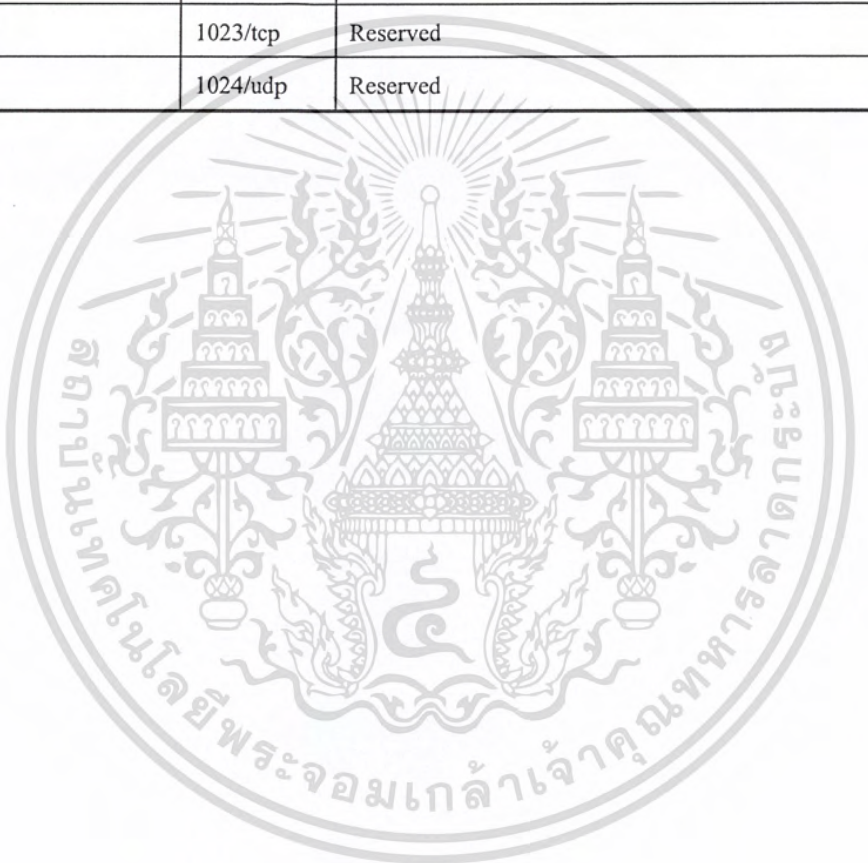
Keyword	*Decimal*	*Description*
prm-sm	408/tcp/udp	Prospero Resource Manager Sys. Man.
prm-nm	409/tcp/udp	Prospero Resource Manager Node. Man.
decladebug	410/tcp/udp	DECLadebug Remote Debug Protocol
rmt	411/tcp/udp	Remote MT Protocol
synoptics-trap	412/tcp/udp	Trap Convention Port
smsp	413/tcp/udp	SMSP
infoseek	414/tcp/udp	Infoseek
bnet	415/tcp/udp	BNet
silverplatter	416/tcp/udp	Silverplatter
onmux	417/tcp/udp	Onmux
hyper-g	418/tcp/udp	Hyper-G
ariel1	419/tcp/udp	Ariel
smpte	420/tcp/udp	SMPTE
ariel2	421/tcp/udp	Ariel
ariel3	422/tcp/udp	Ariel
opc-job-start	423/tcp/udp	IBM Operations Planning and Control Start
opc-job-track	424/tcp/udp	IBM Operations Planning and Control Track
icad-el	425/tcp/udp	ICAD
smartsdp	426/tcp/udp	smartsdp
svrloc	427/tcp/udp	Server Location
ocs_cmu	428/tcp/udp	OCS_CMU
ocs_amu	429/tcp/udp	OCS_AMU
utmpsd	430/tcp/udp	UTMPSD
utmpcd	431/tcp/udp	UTMPCD
iasd	432/tcp/udp	IASD
nnsp	433/tcp/udp	NNSP
mobileip-agent	434/tcp/udp	MobileIP-Agent
mobilip-mn	435/tcp/udp	MobilIP-MN
dna-cml	436/tcp/udp	DNA-CML
comscm	437/tcp/udp	comscm
dsfgw	438/tcp/udp	dsfgw
dasp	439/tcp/udp	dasp

Keyword	*Decimal*	*Description*
sgcp	440/tcp/udp	sgcp
decvms-sysmgt	441/tcp/udp	decvms-sysmgt
cvc_hostd	442/tcp/udp	cvc_hostd
https	443/tcp/udp	https MCom
snpp	444/tcp/udp	Simple Network Paging Protocol
microsoft-ds	445/tcp/udp	Microsoft-DS
ddm-rdb	446/tcp/udp	DDM-RDB
ddm-dfm	447/tcp/udp	DDM-RFM
ddm-byte	448/tcp/udp	DDM-BYTE
as-servermap	449/tcp/udp	AS Server Mapper
tserver	450/tcp/udp	TServer
exec	512/tcp	Remote process execution
biff	512/udp	Used by mail system to notify users
login	513/tcp	Remote login a la telnet
who	513/udp	Maintains databases showing who's
cmd	514/tcp	Like exec, but automatic
syslog	514/udp	
printer	515/tcp/udp	Spooler
talk	517/tcp/udp	Like tenex link, but across tcp connection is established
ntalk	518/tcp/udp	
utime	519/tcp/udp	unixtime
efs	520/tcp	Extended file name server
router	520/udp	Local routing process (on site)
timed	525/tcp/udp	Timeserver
tempo	526/tcp/udp	Newdate
courier	530/tcp/udp	rpc
conference	531/tcp/udp	chat
netnews	532/tcp/udp	readnews
netwall	533/tcp/udp	For emergency broadcasts
apertus-ldp	539/tcp/udp	Apertus Technologies Load Determination
uucp	540/tcp/udp	uucpd
uucp-rlogin	541/tcp/udp	uucp-rlogin

Keyword	*Decimal*	*Description*
klogin	543/tcp/udp	
kshell	544/tcp/udp	krcmd
new-rwho	550/tcp/udp	new-who
dsf	555/tcp/udp	
remotefs	556/tcp/udp	rfs server
rmonitor	560/tcp/udp	rmonitord
monitor	561/tcp/udp	
chshell	562/tcp/udp	chcmd
9pfs	564/tcp/udp	Plan 9 file service
whoami	565/tcp/udp	whoami
meter	570/tcp/udp	demon
meter	571/tcp/udp	udemon
ipcserver	600/tcp/udp	Sun IPC server
urm	606/tcp/udp	Cray Unified Resource Manager
nqs	607/tcp/udp	nqs
sift-uft	608/tcp/udp	Sender-Initiated/Unsolicited File Transfer
npmp-trap	609/tcp/udp	npmp-trap
npmp-local	610/tcp/udp	npmp-local
npmp-gui	611/tcp/udp	npmp-gui
ginad	634/tcp/udp	ginad
mdqs	666/tcp/udp	
doom	666/tcp/udp	doom Id Software
elcsd	704/tcp/udp	errlog copy/server daemon
entrustmanager	709/tcp/udp	EntrustManager
netviewdm1	729/tcp/udp	IBM NetView DM/6000 Server/Client
netviewdm2	730/tcp/udp	IBM NetView DM/6000 send/tcp
netviewdm3	731/tcp/udp	IBM NetView DM/6000 receive/tcp
netgw	741/tcp/udp	netGW
netrcs	742/tcp/udp	Network Based Rev. Cont. Sys.
flexlm	744/tcp/udp	Flexible License Manager
fujitsu-dev	747/tcp/udp	Fujitsu Device Control
ris-cm	748/tcp/udp	Russell Info Sci Calendar Manager

Keyword	*Decimal*	*Description*
kerberos-adm	749/tcp/udp	kerberos administration
rfile	750/tcp	
loadav	750/udp	
pump	751/tcp/udp	
qrh	752/tcp/udp	
rrh	753/tcp/udp	
tell	754/tcp/udp	Send
nlogin	758/tcp/udp	
con	759/tcp/udp	
ns	760/tcp/udp	
rxe	761/tcp/udp	
quotad	762/tcp/udp	
cycleserv	763/tcp/udp	
omserv	764/tcp/udp	
webster	765/tcp/udp	
phonebook	767/tcp/udp	Phone
vid	769/tcp/udp	
cadlock	770/tcp/udp	
rtip	771/tcp/udp	
cycleserv2	772/tcp/udp	
submit	773/tcp	
notify	773/udp	
rpasswd	774/tcp	
acmaint_dbd	774/udp	
entomb	775/tcp	
acmaint_transd	775/udp	
wpages	776/tcp/udp	
wpgs	780/tcp/udp	
concert	786/tcp/udp	Concert
mdbs_daemon	800/tcp/udp	
device	801/tcp/udp	
xtreelic	996/tcp/udp	Central Point Software

Keyword	*Decimal*	*Description*
maitrd	997/tcp/udp	
busboy	998/tcp	
puparp	998/udp	
garcon	999/tcp	
applix	999/udp	Applix ac
puprouter	999/tcp/udp	
cadlock	1000/tcp	
ock	1000/udp	
	1023/tcp	Reserved
	1024/udp	Reserved



บรรณานุกรม

[1] ยุทธนา ลีลาศวัฒนกุล (2544) : "คู่มือการเขียนโปรแกรมและใช้งาน Visual C++ 6.0

[2] Dave Roberts. Internet Protocols Handbook. U.S.A: The Coriolis Group, 1996.

Ed Taylor. TCP/IP Complete. U.S.A: McGraw-Hill, 1998.

[3] สุรศักดิ์ สงวนพงษ์. สถาปัตยกรรมและโปรโตคอลทีซีพีไอพี. กรุงเทพมหานคร: ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์, 2543.

[4] สุวัฒน์ ปุณณชัยยะ และคณะ. เปิดโลกของ TCP/IP และโปรโตคอลของอินเตอร์เน็ท. กรุงเทพมหานคร: บริษัท โปรวิชั่น จำกัด, 2543.

เว็ปไซด์อ้างอิง

http://www.cert.org

http://www.nmap.org

http://www.securityfocus.com

http://www.snort.org/docs

http://www.thaidev.com