โครงงาน ระบบมอนิเตอร์เครือข่ายคอมพิวเตอร์ท้องถิ่น

Network Monitoring System

โดย

นายธนาธิป ธาระวานิช

นางสาวมณีโชติ สมานไทย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

โครงงาน ระบบมอนิเตอร์เครือข่ายคอมพิวเตอร์ท้องถิ่น

Network Monitoring System

โดย

นายธนาธิป ธาระวานิช 38014193

นางสาวมณีโชติ สมานไทย 38014384

อาจารย์ที่ปรึกษา

อาจารย์ ธนา หงษ์สุวรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบมอนิเตอร์เครือข่ายคอมพิวเตอร์ท้องถิ่น

NETWORK MONITORING SYSTEM

ผู้จัดทำ

1. นายธนาธิป ธาระวานิช รหัสประจำตัวนักศึกษา 38014193
2. นางสาวมณีโชติ สมานไทย รหัสประจำตัวนักศึกษา 38014384

..... อาจารย์ที่ปรึกษา

(อาจารย์ ธนา หงษ์สุวรรณ)

โครงงานระบบมอนิเตอร์เครือข่ายคอมพิวเตอร์ท้องถิ่น

ธนาธิป ธาระวานิช 38014193
มณีโชติ สมานไทย 38014384
อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

ในปัจจุบันได้มีการใช้งานระบบเครือข่ายคอมพิวเตอร์กันอย่างกว้างขวาง จนแทบจะกล่าวได้ว่าไม่มีองค์กรทางธุรกิจใดเลยที่ไม่มีการใช้งานระบบเครือข่ายคอมพิวเตอร์ เพราะถือเป็นปัจจัยหนึ่งที่ทำให้มีโอกาสในการแข่งขันมากกว่า ลดค่าใช้จ่ายในการติดต่อสื่อสาร และต้นทุนในการผลิต

เช่นเดียวกับสิ่งต่าง ๆ ในโลกนี้ที่ย่อมเกิดความผิดพลาดได้ ในระหว่างที่ใช้งานระบบเครือข่ายก็มีโอกาสเกิดข้อผิดพลาดซึ่งอาจแก้ไขได้เร็วช้าแตกต่างกันไปตามปัญหาที่เกิดขึ้น แต่ก็ต้องการการแก้ไขที่เร็วที่สุดเท่าที่จะเป็นไปได้ ซึ่งเป็นการยากที่ผู้ดูแลระบบจะทำได้โดยไม่มีเครื่องมือช่วยเลย ดังนั้นจึงมีผู้พัฒนาเครื่องมือช่วยผู้ดูแลระบบขึ้นมาหลายอย่าง ทั้งซอฟต์แวร์และฮาร์ดแวร์ อาทิเช่น โปรแกรม NetXRay , ตัววิเคราะห์โพรโตคอล(Protocol analyzer) เป็นต้น

ปริญญานิพนธ์นี้เป็นการพัฒนาซอฟต์แวร์ที่รันบนแพล็ตฟอร์มดอส ซึ่งช่วยให้ผู้ดูแลระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นอีเธอร์เน็ตสามารถจัดการกับระบบเครือข่ายคอมพิวเตอร์ได้ในเวลาปกติและเวลาที่เกิดปัญหา รวมทั้งวางแผนในการปรับปรุงเปลี่ยนแปลงระบบเครือข่ายคอมพิวเตอร์ให้เหมาะสมกับการใช้งานอยู่ตลอดเวลา โดยทำการจับแพ็คเกจที่ถูกส่งในระบบเครือข่ายท้องถิ่นแล้วนำมาวิเคราะห์ได้เป็นข้อมูลที่มีประโยชน์ในการดูแลระบบ นอกจากประโยชน์ที่กล่าวมาแล้วข้างต้น ซอฟต์แวร์ตัวนี้ยังช่วยให้ผู้ที่ศึกษาทางด้านวิชาระบบเครือข่ายคอมพิวเตอร์สามารถทำความเข้าใจในระบบเครือข่ายคอมพิวเตอร์ได้ดียิ่งขึ้นอีกด้วย

Network Monitoring System

Thanathip Tharavanich
Maneechote Samarnthai
Mr. Thana Hongsuwan Advisor

ABSTRACT

In the present days, computer network is widely used in almost organization especially commercial organization because it's one of factor make the organization have advantage in competition , reduce the cost of communications and manufacuring.

Like everything in the world that have opportunity to work faulty. During working, the network may down, slow down or show unacceptable events which need to repair as soon as possible. But it is difficult to troubleshoot the problem by free hand. Thus, there're many people develop tools for the reason such as NetXRay program and Protocol analyzer.

This thesis is to develop software tool run on DOS platform. By capture packet transmitted in the Ethernet LAN and analyze them can help LAN administrators manage LAN in the usual and unusal behavior. And also leads to LAN adaptation to meet the real user need. The utilities of the software tool also help the network interested persons understand this field better.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน ซึ่งบุคคลแรกที่จะต้องกล่าวถึงเพราะเป็นผู้มีส่วนสำคัญให้เกิดปริญญานิพนธ์ฉบับนี้คือ อาจารย์ธนา หงษ์สุวรรณ ซึ่งเป็นอาจารย์ที่ปรึกษาให้กลุ่มผู้จัดทำ และยังให้คำแนะนำที่ดีเสมอมาในยามที่มีปัญหา กลุ่มผู้จัดทำจึงขอขอบพระคุณไว้ ณ ที่นี้

บุคคลกลุ่มต่อมาที่จะต้องกล่าวถึงก็คือสมาชิกในกลุ่มผู้จัดทำเองที่ได้ร่วมมือร่วมใจกันทำงานชิ้นนี้จนสำเร็จลุล่วงไปได้ นอกจากนี้แล้วยังมีเพื่อน ๆ ทุกคนทั้งที่รู้จักคุ้นเคยเป็นอย่างดีและเพื่อน ๆ ในไซเบอร์สเปซที่มีส่วนร่วมในปริญญานิพนธ์ชิ้นนี้ไม่ว่ามากหรือน้อยก็ตาม และสุดท้ายต้องขอขอบพระคุณบุพการีของกลุ่มผู้จัดทำทุกท่าน ซึ่งก็มีส่วนร่วมในปริญญานิพนธ์ชิ้นนี้เช่นกัน โดยเฉพาะในด้านทุนทรัพย์

ธนาธิป ธาระวานิช
มณีโชติ สมานไทย

สารบัญ

สารบัญตาราง

หน้าที่

สารบัญภาพ

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เนื่องจากในปัจจุบันการใช้งานระบบเครือข่ายคอมพิวเตอร์ได้ทวีความสำคัญมากขึ้น จนสามารถที่จะกล่าวได้ว่าไม่มีองค์กรใดที่ไม่มีการใช้งานระบบเครือข่ายคอมพิวเตอร์ แต่การใช้งานสิ่งใดก็ตาม รวมทั้งระบบเครือข่ายคอมพิวเตอร์นั้นย่อมมีโอกาสเกิดปัญหา เกิดข้อผิดพลาดขึ้นได้ ดังนั้นจึงมีความจำเป็นที่จะต้องมีเครื่องมือที่ช่วยในการดูแลให้ระบบเครือข่ายคอมพิวเตอร์ทำงานได้เป็นปกติ หรือถ้าเกิดข้อผิดพลาดขึ้นก็ช่วยให้ทราบปัญหาได้อย่างรวดเร็ว ทำให้สามารถแก้ไขได้ทันต่อความต้องการของผู้ใช้งาน โครงงานนี้จึงได้พัฒนาซอฟต์แวร์ขึ้นมาเพื่อเป็นเครื่องมือช่วยผู้ดูแลระบบให้ทราบสถานะของระบบเครือข่ายได้ นอกจากนี้ซอฟต์แวร์นี้ยังเป็นประโยชน์แก่ผู้ที่ศึกษาทางด้านระบบเครือข่ายคอมพิวเตอร์คือสามารถช่วยให้เข้าใจระบบการทำงานของเครือข่ายได้มากยิ่งขึ้นอีกด้วย

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อพัฒนาซอฟต์แวร์(Software) ที่สามารถตรวจดูระบบเครือข่ายท้องถิ่น(LAN) ซึ่งช่วยให้การศึกษาวิชาทางด้านระบบเครือข่ายคอมพิวเตอร์เป็นไปด้วยความเข้าใจมากยิ่งขึ้น
2. เพื่อให้มีเครื่องมือ(Tool) ที่ใช้เก็บสถิติเกี่ยวกับการใช้งานระบบเครือข่ายท้องถิ่น
3. เพื่อให้มีเครื่องมือช่วยจัดการ(Management) ระบบเครือข่ายท้องถิ่น ซึ่งจะช่วยในการปรับปรุงระบบเครือข่าย วางแผนในการขยายเครือข่าย และแก้ไขความผิดพลาดของเครือข่ายได้

1.3 ขอบเขตของโครงงาน

งานวิจัยนี้จะพัฒนาซอฟต์แวร์ที่เป็นเครื่องมือช่วยผู้ดูแลระบบในการบริหารระบบเครือข่ายคอมพิวเตอร์ อีกทั้งยังสามารถนำมาใช้ในการศึกษาระบบเครือข่ายคอมพิวเตอร์ได้ โดยมีขอบเขตในการวิจัยดังต่อไปนี้

1. เป็นซอฟต์แวร์สำหรับการตรวจดูระบบเครือข่ายแบบภายนอก(External network monitoring) และใช้ได้ในเครือข่ายท้องถิ่นแบบอีเธอร์เน็ต(Ethernet LAN) โดยทำงานบนคอมพิวเตอร์ส่วนบุคคล(Personal Computer) ที่มีระบบปฏิบัติการดอส(DOS)
2. สามารถดักจับแพ็คเกจ(Packet) และเก็บไว้เพื่อวิเคราะห์ โดยสามารถวิเคราะห์โพรโตคอลหลัก ๆ ในแต่ละเลเยอร์(Layer) ได้ดังนี้

2.1 ดาต้าลิงค์เลเยอร์(Datalink Layer)

- ไอทริปเปิ้ลอี 802.2 (IEEE 802.2)
- ไอทริปเปิ้ลอี 802.3 (IEEE 802.3)
- อีเธอร์เนตทู (Ethernet II)
- อีเธอร์เนตสแนบ (Ethernet SNAP)

2.2 เน็ตเวิร์คเลเยอร์(Network Layer)

- ไอพี (IP)

- ไอพีเอ็กซ์ (IPX)
- เออาร์พี (ARP)
- อาร์เออาร์พี (RARP)
- ไอซีเอ็มพี (ICMP)

2.3 ทรานสปอร์ตเลเยอร์(Transport Layer)

- ทีซีพี (TCP)
- ยูดีพี (UDP)
- เอสพีเอ็กซ์ (SPX)

2.4 แอพพลิเคชั่นเลเยอร์(Application Layer)

- เอฟทีพี (FTP)
- เทลเน็ต (Telnet)
- เอสเอ็มทีพี (SMTP)
- โกเฟอร์ (Gopher)
- เอชทีทีพี (HTTP)
- ป๊อป 3 (POP3)
- เอ็นเอ็นทีพี (NNTP)
- เอสเอ็นเอ็มพี (SNMP)
- ไออาร์ซี (IRC)

3. สามารถกรองแพ็คเกจได้โดยสามารถเลือกรูปแบบการกรองได้ดังนี้

3.1 โดยใช้โพรโตคอลเป็นตัวแยก ประกอบด้วย

3.1.1 ทีซีพี (TCP)

3.1.2 ยูดีพี (UDP)

3.1.3 เอสพีเอ็กซ์ (SPX)

3.1.4 ไอซีเอ็มพี (ICMP)

3.1.5 เออาร์พี (ARP)

3.1.6 อาร์เออาร์พี (RARP)

3.1.7 ไอพี (IP)

3.1.8 ไอพีเอ็กซ์ (IPX)

3.2 โดยใช้ลักษณะของแพ็คเกจเป็นตัวแยก

3.2.1 สามารถระบุว่าให้กรองแพ็คเกจที่มีขนาดเท่าใดได้ ซึ่งทำให้หาแพ็คเกจที่มีขนาดผิดปกติที่ขนาดได้

3.2.2 กำหนดให้กรองเฉพาะแพ็คเกจที่มีแอดเดรสต้นทาง ปลายทางหรือทั้งสองตามที่กำหนด ซึ่งทำให้กรองการสนทนาระหว่างสเตชั่นใด ๆ การบรอดคาสท์(broadcast) รวมทั้งการมัลติคาสท์

(multicast) ได้ทั้งแมคแอดเดรสและไอพีแอดเดรส

4. สามารถแสดงผลการตรวจดูระบบเครือข่ายได้ดังนี้

4.1 เปอร์เซ็นต์การใช้งานจากการเปรียบเทียบขนาดข้อมูลต่อเวลากับแบนด์วิธสูงสุด (ในแลนแบบอีเธอร์เน็ตจะเท่ากับ 10 Mbit/s)

4.2 โหนดที่ใช้งานสูงสุด

4.3 เปอร์เซ็นต์การใช้งานของโหนดที่กำหนดในช่วงเวลาที่ทำการจับข้อมูลจากระบบเครือข่ายโดยระบุได้จากแมคแอดเดรส

4.4 เปอร์เซ็นต์การส่งแพ็คเกจจากต้นทาง และปลายทางที่กำหนดโดยไอพีแอดเดรส

4.5 จำนวนเฟรมที่จับได้ต่อวินาที(Frame per second)

4.6 การกระจายของขนาดเฟรม(Frame Distribution) ที่ไม่คิดฟิลด์ Preamble, SFD และFCS โดยแบ่งเป็น

4.6.1 60 - 420 ไบต์

4.6.2 421 - 780 ไบต์

4.6.3 781 - 1140 ไบต์

4.6.4 1141 - 1514 ไบต์

4.7 การกระจายของโพรโตคอล(Protocol Distribution) แบ่งเป็น

4.7.1 โพรโตคอลในเลเยอร์ดาต้าลิงค์ตามที่ระบุในหัวข้อ 2.1

4.7.2 โพรโตคอลในเลเยอร์เน็ตเวิร์คตามที่ระบุในหัวข้อ 2.2

4.7.3 โพรโตคอลในเลเยอร์ทรานสปอร์ตตามที่ระบุในหัวข้อ 2.3

4.7.4 โพรโตคอลในเลเยอร์แอพพลิเคชั่นตามที่ระบุในหัวข้อ 2.4

5. สามารถเก็บข้อมูลที่วิเคราะห์มาได้เป็นสถิติเพื่อใช้ประโยชน์ในภายหลังได้

6. สามารถสร้างแพ็คเกจขึ้นมา(Generate traffic) เพื่อจำลองสถานการณ์และทดสอบเครือข่าย

1.4 วิธีการดำเนินงาน

โครงงานนี้เริ่มต้นจากการศึกษาเรื่องต่าง ๆ ที่จะเป็นพื้นฐานในการทำโครงงานนี้ให้สำเร็จลุล่วงไป เริ่มตั้งแต่บทที่ 2 ซึ่งจะมีเนื้อหาเกี่ยวข้องกับสถาปัตยกรรมเครือข่ายและระดับชั้นโพรโตคอลต่าง ๆ ตั้งแต่ดาต้าลิงค์เลเยอร์จนถึงแอพพลิเคชั่นเลเยอร์ ส่วนบทที่ 3 จะกล่าวถึงหลักการเบื้องต้นและหลักการออกแบบตัวมอนิเตอร์ระบบเครือข่ายคอมพิวเตอร์ จนถึงบทที่ 4 ซึ่งจะกล่าวถึงการวิธีติดต่อกับแพ็คเกจไดรเวอร์ ส่วนในบทต่อ ๆ มาจะเป็นรายละเอียดในการออกแบบและพัฒนาซอฟต์แวร์ของโครงงานนี้ซึ่งแสดงไว้ในบทที่ 5 รวมถึงการทดสอบแต่ละระบบย่อยในบทที่ 6 จากนั้นจะกล่าวถึงแนวโน้มในอนาคต บทสรุปและวิจารณ์ในบทที่ 7

เพื่อให้ง่ายในการทำความเข้าใจและเป็นการรวบรวมข้อมูลที่จำเป็นไว้รวมกัน ในแต่ละภาคผนวกจึงได้รวบรวมความรู้ที่เกี่ยวข้อง อีกทั้งใช้เป็นที่อ้างอิงไว้ ซึ่งได้แก่ รายละเอียดในการติดต่อกับการ์ดแลน, รูปแบบโพรโตคอลเฮดเดอร์, การพิจารณาสมรรถภาพของ CSMA/CD, การหาสาเหตุความผิดปกติที่ชั้นดาต้าลิงค์ และหมายเลขพอร์ตที่รู้จักกันดี โดยเริ่มจากภาคผนวก ก เป็นลำดับไป

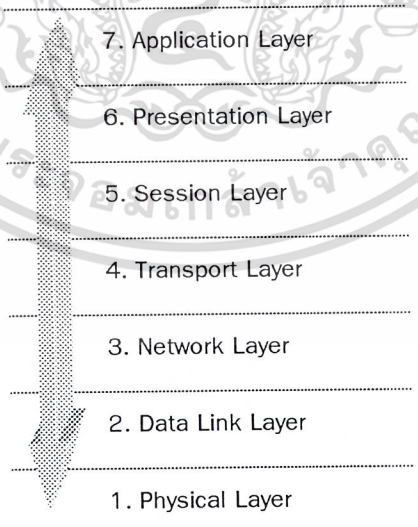
บทที่ 2
สถาปัตยกรรมเครือข่ายและระดับชั้นโพรโตคอล

2.1 แบบอ้างอิงโอเอสไอ(OSI Reference Model)

ในการกล่าวถึงระบบเครือข่ายคอมพิวเตอร์ เราจะต้องใช้คำที่มีความหมายเฉพาะเจาะจงเกี่ยวกับระบบสื่อสารข้อมูล ซึ่งแต่ละคนอาจใช้ไม่เหมือนกัน การมีสิ่งไว้อ้างอิงจึงเป็นสิ่งที่จำเป็นในการทำความเข้าใจความหมายของคำที่ใช้ในการสื่อสารข้อมูล โมเดลแบบสถาปัตยกรรม(architectural model) ที่ถูกพัฒนาโดยองค์การมาตรฐานโลก(International Standards Organization: ISO) ได้ถูกใช้อย่างแพร่หลายในการอธิบายถึงโครงสร้างและหน้าที่ของโพรโตคอลในการสื่อสารข้อมูล โมเดลดังกล่าวนี้มีชื่อว่าแบบอ้างอิงการเชื่อมต่อของระบบเปิด(Open System Interconnection: OSI Reference Model)

โมเดลนี้ประกอบไปด้วยเลเยอร์จำนวน 7 ชั้นที่กำหนดหน้าที่ของโพรโตคอลการสื่อสารข้อมูลแต่ละเลเยอร์แสดงถึงหน้าที่เมื่อข้อมูลถูกส่งระหว่างแอพพลิเคชั่นที่ทำงานร่วมกันข้ามเครือข่าย ดังรูป 2-1 ที่แสดงถึงแต่ละเลเยอร์พร้อมทั้งคำอธิบายเกี่ยวกับหน้าที่ของมันสั้น ๆ มักเรียกโครงสร้างนี้ว่าสแต็กหรือโพรโตคอลสแต็ก

หนึ่งเลเยอร์ไม่ได้กำหนดโพรโตคอลเดียวแต่กำหนดหน้าที่ในการสื่อสารข้อมูลที่สามารถทำได้โดยโพรโตคอลจำนวนเท่าไหร่ก็ได้ ดังนั้นแต่ละเลเยอร์จึงมีได้หลายโพรโตคอลซึ่งให้บริการที่เหมาะสมกับหน้าที่ของเลเยอร์นั้น ตัวอย่างเช่นโพรโตคอลไฟล์ทรานสเฟอร์และโพรโตคอลอิเลคทรอนิคส์เมล์ซึ่งทั้งสองให้บริการแก่ผู้ใช้ จึงเป็นส่วนหนึ่งของเลเยอร์แอพพลิเคชั่น



รูปที่ 2-1 แบบอ้างอิงโอเอสไอ

ทุกโพรโตคอลติดต่อสื่อสารกับเพียร์(peer) ของมัน เพียร์คือการนำโพรโตคอลเดียวกันในเลเยอร์ที่เทียบเท่ากันบนระบบไกล(remote system) มาใช้งานจริง เช่น โพรโตคอลไฟล์ทรานสเฟอร์ของเราคือ

เพียร์ของโพรโตคอลไฟล์ทรานสเฟอร์ที่ใช้อยู่อีกระบบหนึ่ง โดยนามธรรมแล้วแต่ละโพรโตคอลจะเกี่ยวข้องกับการสื่อสารกับเพียร์ของมันโดยที่ไม่ต้องสนใจเลเยอร์ที่อยู่เหนือหรือต่ำกว่ามัน แต่การส่งข้อมูลจริง ๆ จะเกี่ยวข้องกับทุกเลเยอร์ที่จะต้องส่งข้อมูลจากแอพพลิเคชั่นที่เราใช้ไปยังแอพพลิเคชั่นที่เทียบเท่ากันในระบบไกล เลเยอร์ที่สูงกว่าจะเกี่ยวข้องกับเลเยอร์ที่ต่ำกว่าในการส่งข้อมูลไปบนเครือข่ายที่มีอยู่ ข้อมูลถูกส่งลงมาจากสแต็กจากเลเยอร์หนึ่งสู่เลเยอร์ถัดไป จนกระทั่งข้อมูลถูกส่งไปบนเครือข่ายโดยโพรโตคอลในเลเยอร์ฟิสิคัล(physical layer) ที่ปลายอีกข้างหนึ่งก็จะทำในลักษณะตรงกันข้ามคือข้อมูลจะถูกส่งไปบนสแต็กให้แก่แอพพลิเคชั่นที่ต้องการ แต่ละเลเยอร์ไม่จำเป็นต้องรู้ว่าเลเยอร์บนและล่างมันทำหน้าที่อะไร มันจำเป็นต้องรู้เฉพาะว่าทำอย่างไรจึงจะส่งข้อมูลไปให้ได้ การแบ่งหน้าที่ของการสื่อสารข้อมูลออกเป็นหลาย ๆ เลเยอร์จะช่วยลดผลกระทบที่จะต้องเปลี่ยนแปลงทั้งชุดโพรโตคอลเมื่อเทคโนโลยีเปลี่ยนไป ตัวอย่างเช่น เราสามารถเพิ่มแอพพลิเคชั่นเข้าไปใหม่โดยไม่ต้องเปลี่ยนเครือข่ายในระดับฟิสิคัลหรือฮาร์ดแวร์ใหม่สามารถติดตั้งได้โดยไม่ต้องเขียนซอฟต์แวร์ใหม่

แบบอ้างอิงโอเอสไอประกอบด้วยเลเยอร์ต่าง ๆ 7 ชั้นดังนี้

Application Layer เป็นเลเยอร์ที่ติดต่อกับผู้ใช้ ประกอบด้วยแอพพลิเคชั่นโปรแกรมที่ใช้เครือข่าย

Presentation Layer การที่แอพพลิเคชั่นจะทำงานร่วมกันได้จะต้องทำการตกลงว่าจะแทนข้อมูลกันอย่างไร เลเยอร์นี้จะกำหนดมาตรฐานการแทนข้อมูลให้

Session Layer จัดการการติดต่อระหว่างแอพพลิเคชั่นที่ทำงานร่วมกัน

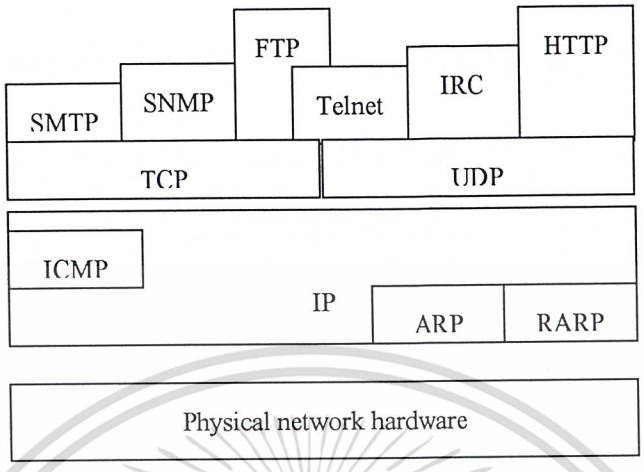
Transport Layer รับประกันว่าผู้รับจะต้องได้รับข้อมูลอย่างถูกต้องครบถ้วน

Network Layer จัดการการติดต่อข้ามเครือข่ายและแยกโพรโตคอลในเลเยอร์ที่สูงกว่าออกจากรายละเอียดของเครือข่าย

Data Link Layer ดูแลความน่าเชื่อถือของการส่งข้อมูลข้ามเครือข่ายฟิสิคัล

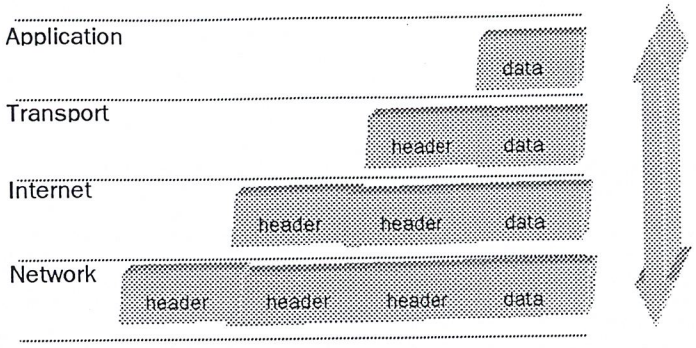
Physical Layer กำหนดคุณลักษณะของฮาร์ดแวร์ที่จำเป็นต้องใช้ในการพาสัญญาณการสื่อสารข้อมูล เช่น ระดับโวลเตจ จำนวนและตำแหน่งของพิน(pins) ที่ใช้ในการอินเทอร์เฟส ตัวอย่างมาตรฐานในเลเยอร์นี้คือ คอนเน็คเตอร์ในการอินเทอร์เฟส เช่น RS232C และ V.35 และมาตรฐานในการเชื่อมสายของเครือข่าย เช่น ไอทริปเปิ้ลอี 802.3

2.2 ชุดโพรโตคอลทีซีพี/ไอพี(TCP/IP Protocol suite)



รูปที่ 2-2 สถาปัตยกรรมทีซีพี/ไอพี

ประกอบด้วยเลเยอร์หลายชั้นเช่นเดียวกับแบบอ้างอิงโอเอสไอดังรูป 2-2 ถ้าพิจารณาจากหน้าที่ของแต่ละเลเยอร์แล้ว 4 เลเยอร์ล่างของทีซีพี/ไอพีสามารถนำมาเปรียบเทียบกับ 4 เลเยอร์ล่างของแบบอ้างอิงโอเอสไอได้ โดยเลเยอร์ 1 และ 2 เป็นเลเยอร์ที่ใช้ร่วมกันได้(compatible) เพราะโอเอสไอกำหนดระบบตัวกลางหลายระบบในเลเยอร์ดังกล่าว และทีซีพี/ไอพีก็ถูกออกแบบให้ใช้ตัวกลางใดก็ได้(medium independent) เมื่อพิจารณาเลเยอร์ 3 และ 4 คืออินเทอร์เน็ตและทรานสปอร์ตของทีซีพี/ไอพี และเน็ตเวิร์คและทรานสปอร์ตของโอเอสไอ จะเห็นได้ว่าโอเอสไอมีทางเลือกมากมายที่สามารถใช้ในเลเยอร์ทั้งสองนี้ได้ และบางตัวก็ทำหน้าที่ในลักษณะคล้ายคลึงกับในทีซีพี/ไอพี ข้อแตกต่างสำคัญระหว่างทีซีพี/ไอพีกับโอเอสไอคือเลเยอร์แอพพลิเคชั่นของทีซีพี/ไอพีซึ่งในโอเอสไอแล้วจะเท่ากับ 3 เลเยอร์บน ถึงแม้จะมีความแตกต่างในการแบ่งเป็นเลเยอร์อยู่บ้าง แต่ลักษณะในการส่งข้อมูลของทีซีพี/ไอพีจะเหมือนกับของโอเอสไอคือข้อมูลจะถูกส่งลงมาจากสแต็กหรือส่งขึ้นไปบนสแต็ก ขณะที่ข้อมูลถูกส่งลงมาแต่ละเลเยอร์ในสแต็กก็จะเพิ่มข้อมูลควบคุม(control information) เข้าไปเพื่อจะแน่ใจได้ว่าการส่งเกิดขึ้นอย่างเหมาะสม ข้อมูลควบคุมนี้เรียกว่าเฮดเดอร์เพราะมันถูกใส่ไว้หน้าข้อมูลที่ถูกส่ง แต่ละเลเยอร์ปฏิบัติกับข้อมูลทั้งหมดที่มันได้รับมาจากเลเยอร์ที่สูงกว่าเหมือนเป็นข้อมูลจริง ๆ และเพิ่มเฮดเดอร์ของมันเองไว้ข้างหน้าข้อมูลทั้งหมดนั้น การทำดังกล่าวนี้เรียกว่าเอ็นแคปซูเลชั่น(encapsulation) ดังรูป 2-3 เมื่อผู้รับได้รับข้อมูลก็จะทำตรงกันข้ามกับที่กล่าวมาคือแต่ละเลเยอร์จะนำเฮดเดอร์ออกมาก่อนที่จะส่งข้อมูลขึ้นไปยังเลเยอร์ที่สูงกว่าข้อมูลก็จะไหลขึ้นไปบนสแต็ก ข้อมูลที่ได้รับก็จะถูกแปลความหมายทั้งเฮดเดอร์และข้อมูล

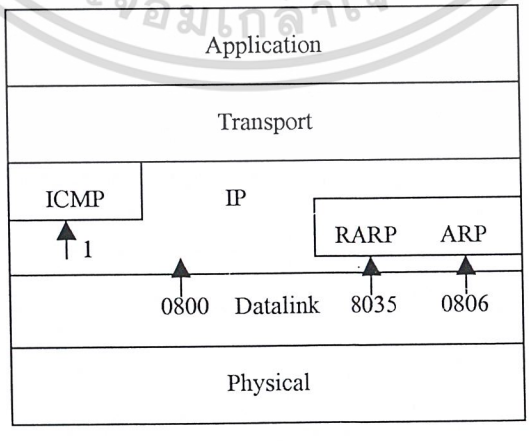


รูปที่ 2-3 การส่งผ่านข้อมูลระหว่างเลเยอร์

เลเยอร์ต่าง ๆ และโพรโตคอลที่เกี่ยวข้องในโพรโตคอลชุดทีซีพี/ไอพีดังนี้

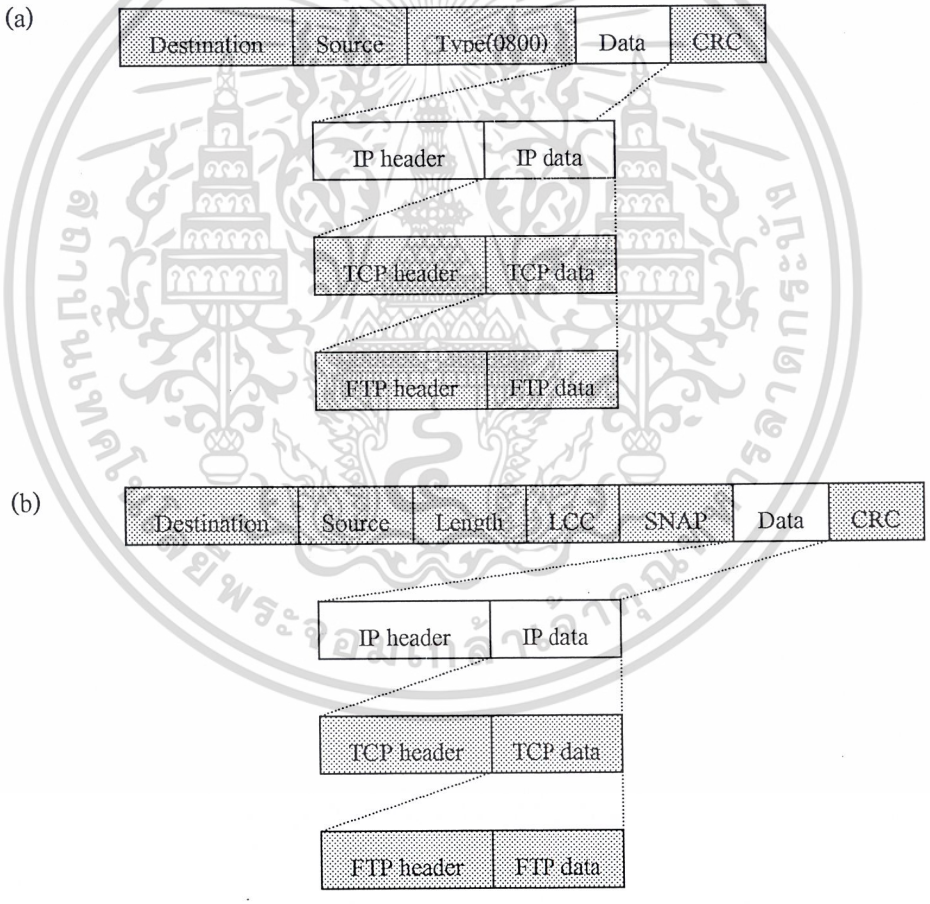
2.2.1 เลเยอร์เน็ตเวิร์คของทีซีพี/ไอพี

มีโพรโตคอลไอพี (Internet Protocol: IP) ทำหน้าที่พื้นฐานในการส่งโพรโตคอลชั้นสูงกว่าของทีซีพี/ไอพีไปบนเครือข่ายฟิสิคัลทั้งหมด ซึ่งทำให้โพรโตคอลในเลเยอร์ที่สูงกว่าไม่จำเป็นต้องรู้อะไรเกี่ยวกับความสามารถของตัวกลางเลย และยังทำให้ผู้พัฒนาแอพพลิเคชั่นที่เขียนโปรแกรมเหนือเลเยอร์ทรานสปอร์ตทำงานได้ง่ายขึ้นด้วยเหตุผลเดียวกัน นอกจากนี้โพรโตคอลไอพียังเกี่ยวข้องกับการส่งข้อมูลไปยังเครื่องและเครือข่ายที่ต้องการอย่างถูกต้องหรือการเร้าท์เส้นทางนั่นเอง ไอพีเป็นบริการแบบไม่ต้องการก่อตั้งการเชื่อมต่อก่อน(connectionless หรือ connectionless datagram service) เพราะไม่มีการเรียก(call) หรือก่อตั้งวงจรเสมือนก่อนที่จะเริ่มส่งข้อมูล เนื่องจากแต่ละดาต้าแกรมมีข้อมูลทั้งหมดที่จำเป็นต้องใช้ในการเร้าท์เส้นทางอยู่แล้ว และระหว่างโหนด 2 โหนดก็ไม่มีเส้นทางที่เฉพาะเจาะจงซึ่งทำให้ง่ายในการเร้าท์ใหม่แม้จะเสียเวลาในการสวิตชิ่งเล็กน้อย เมื่อเครือข่ายเกิดข้อผิดพลาด ส่วนแอดเดรสปลายทางที่ใช้มีทั้งคนเดียว(unique) เป็นกลุ่ม(multicast) หรือทุกคน(broadcast)



รูปที่ 2-4 สถาปัตยกรรมไอพี

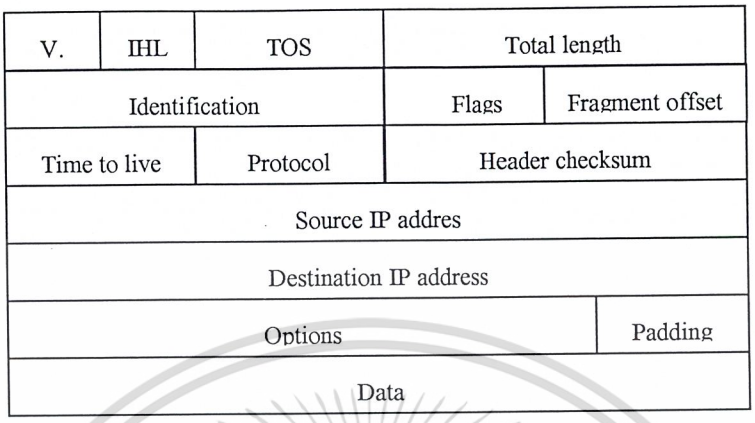
นอกจากไอพีแล้วยังมีโพรโตคอลอื่นในเลเยอร์นี้อีกคือ โพรโตคอลเออาร์พี(Address Resolution Protocol: ARP), โพรโตคอลอาร์เออาร์พี(Reverse Address Resolution Protocol: RARP), โพรโตคอลไอซีเอ็มพี(Internet Control Message Protocol: ICMP) ดังรูป 2-4 โพรโตคอลเออาร์พีและอาร์เออาร์พีแสดงที่ตำแหน่งล่างของชั้นไอพีเพราะโพรโตคอล 2 ตัวนี้ไม่ได้ใช้ไอพีและเป็นที่รู้จักโดยชั้นดาต้าลิงค์ที่สนับสนุนเหมือนเป็นโพรโตคอลที่แยกออกมาต่างหาก ไอซีเอ็มพีแสดงไว้ตำแหน่งบนของชั้นไอพีเพราะมันถูกส่งข้ามเครือข่ายโดยอยู่ในดาต้าแกรมไอพี ชั้นไอพีรู้ว่าเป็นดาต้าแกรมไอซีเอ็มพีโดยค่าโพรโตคอลที่เท่ากับ 1 ทั้งฟิลด์เฮดเดอร์และฟิลด์ข้อมูลของดาต้าแกรมไอพีจะกลายเป็นฟิลด์ข้อมูลของเฟรมในชั้นดาต้าลิงค์ ลักษณะเช่นนี้เรียกว่าการเอ็นแคปซูเลชั่นดังได้กล่าวแล้ว หรือบางครั้งก็เรียกการเอ็นเวลล็อปปิ้ง (enveloping) ซึ่งฟิลด์ข้อมูลของดาต้าแกรมไอพีเองก็บรรจุเฮดเดอร์ของโพรโตคอลในชั้นที่สูงกว่าเช่นเดียวกันดังรูป 2-5 ซึ่งแสดงการเอ็นแคปซูเลชั่นในเฟรมอีเธอร์เน็ตทูและเฟรม SNAP



รูปที่ 2-5 แสดงไอพีใน (a) เฟรมอีเธอร์เน็ตทู (b) เฟรม SNAP

ลักษณะดาต้าแกรมไอพีเวอร์ชั่น 4 เป็นดังรูป 2-6 ซึ่งแสดงในลักษณะกว้าง 32 บิต เมื่อดาต้าแกรมนี้ถูกส่งไปบนเครือข่ายลำดับการส่งจะเป็นจากซ้ายบนไปขวาล่างซึ่งเรียกว่าลำดับไบต์ของเครือข่าย

(network byte order) และตัวเลขในทีซีพี/ไอพีจะถูกส่งโดยให้บิตที่สำคัญสูงสุด(most significant octet) ไปก่อน แต่ละฟิลด์ในรูป 2-6 สามารถอธิบายได้ดังต่อไปนี้



รูปที่ 2-6 รูปแบบของดาต้าแกรมไอพีเวอร์ชั่น 4

1. Version มีขนาด 4 บิต แสดงถึงเวอร์ชั่นของโพรโตคอลไอพี ขณะนี้คือเวอร์ชั่น 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 ระดับ ซึ่งทำให้โหนดของไอพีรู้ว่าดาต้าแกรมใดมีความสำคัญมากกว่าดาต้าแกรมอื่น แต่เราท์เตอร์บางตัวก็ไม่สนใจแฟล็กนี้

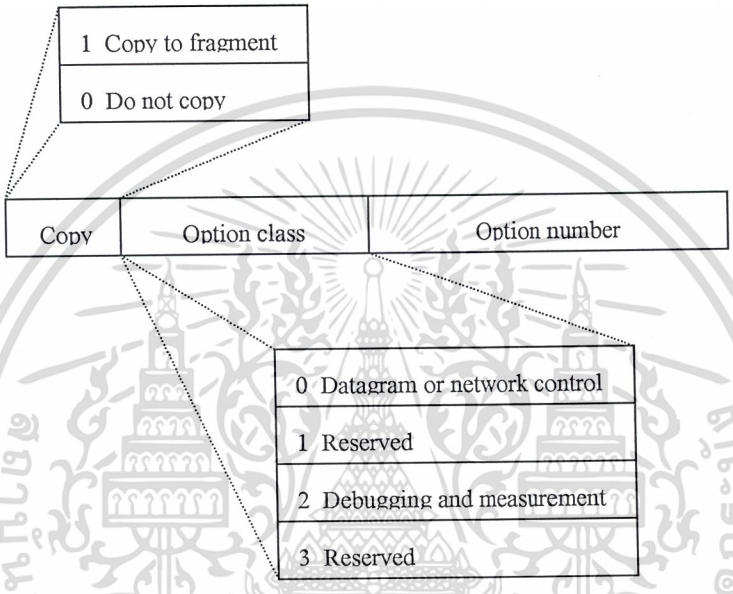
แฟล็กดี(D-flag)	เป็นการร้องขอการเชื่อมต่อที่มีการเสียเวลา(delay) ต่ำ
แฟล็กที(T-flag)	บอกถึงความต้องการทรูพุธ(throughput) สูง
แฟล็กอาร์(R-flag)	บอกถึงความต้องการความเชื่อถือ(reliability) สูง หมายถึงความน่าจะเป็นในการละทิ้ง(discard) ดาต้าแกรมมีต่ำกว่า
แฟล็กซี(C-flag)	บอกถึงความต้องการเสียค่าใช้จ่ายที่ต่ำกว่า
บิตสุดท้ายไม่ใช้	

4. Total Length มีขนาด 16 บิต เป็นการวัดทั้งเฮดเดอร์และข้อมูลในหน่วยออคเต็ด (octets) ซึ่งทำให้คำนวณขนาดข้อมูลโดยคิดจากฟิลด์ Total Length และฟิลด์ IHL ได้ จะเห็นได้ว่าฟิลด์นี้มีขนาด 16 บิตซึ่งหมายความว่าขนาดดาต้าแกรมที่ใหญ่ที่สุดมีขนาดเป็น 65,535 ออคเต็ดซึ่งใหญ่กว่าที่เครือข่ายฟิสิคัลสนับสนุนมาก ถ้าดาต้าแกรมถูกแบ่งย่อย(fragment) ค่าในฟิลด์นี้คือค่าใหม่ ไม่ใช่ค่าเก่าของขนาดดาต้าแกรม

5. **Identification**	มีขนาด 16 บิต บ่งถึงการแบ่งย่อยทั้งหมดของดาต้าแกรม มีลักษณะเฉพาะของใครของมัน(unique) สำหรับแต่ละดาต้าแกรมใหม่ที่ถูกส่งโดยโฮส ฟิลด์นี้ไม่ใช่หมายถึงหมายเลขลำดับ(sequence number) เพราะไอพีเป็นบริการแบบไม่ต้องก่อตั้งการติดต่อก่อนส่งข้อมูล 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 ออคเต็ค ดังนั้นการแบ่งย่อยดาต้าแกรมจึงต้องทำให้หน่วยนี้
8. **Time to live**	มีขนาด 8 บิต ฟิลด์นี้ถูกเซ็ตโดยผู้ส่งดาต้าแกรมและจะถูกลดค่าโดยเราท์เตอร์เมื่อดาต้าแกรมผ่านมัน 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 บอกถึงจุดเริ่มต้นของข้อมูลได้ถูกต้อง เมื่อมีการใช้ฟิลด์ Options ซึ่งความยาวไม่คงที่

15. Options สนับสนุนการดีบัก(debugging) การวัด(measurement) และความปลอดภัย(security) ซึ่งสามารถมีหลาย ออปชั่นได้ในดาต้าแกรมเดียว ดังรูป 2-7 ซึ่งฟิลด์นี้ประกอบด้วย



รูปที่ 2-7 ออปชั่นในการวัดและความปลอดภัย

15.1 Copy มีขนาด 1 บิต ใช้ตัดสินใจว่าออปชั่นจะอยู่ในทุกส่วนดาต้าแกรมที่ถูกแบ่งหรือไม่ 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 มีการกำหนดระดับของดาต้าแกรมจากธรรมดาไปจนถึงดาต้าแกรมที่เป็นความลับมาก ซึ่งช่วยให้เราท์เตอร์รู้ว่าดาต้าแกรมใดบรรจุข้อมูลที่สำคัญและป้องกันข้อมูลเหล่านั้นไม่ให้ออกจากสิ่งแวดล้อมที่ปลอดภัย

15.3.2 Time stamp: คือ Option 4 ทำให้ดาต้าแกรมที่ถูกส่งไปในเครือข่ายสามารถรวบรวมไทม์แสตมป์จากแต่ละเราท์เตอร์ที่มันผ่านซึ่งเราสามารถนำสิ่งที่ได้นี้มาใช้ประเมินการเสียเวลาและความเปลี่ยนแปลงในเครือข่ายของเราท์เตอร์ได้

15.3.3 Loose source route: คือ Option 3 ในการกำหนดค่าของเราท์เตอร์เพื่อให้คาต้าแกรมผ่านตามไอพีแอดเดรสของเราท์เตอร์ ออปชั่นนี้จะอนุญาตให้ใช้เราท์เตอร์อื่นได้ในระหว่างลิสต์ของเราท์เตอร์ที่ถูกกำหนด

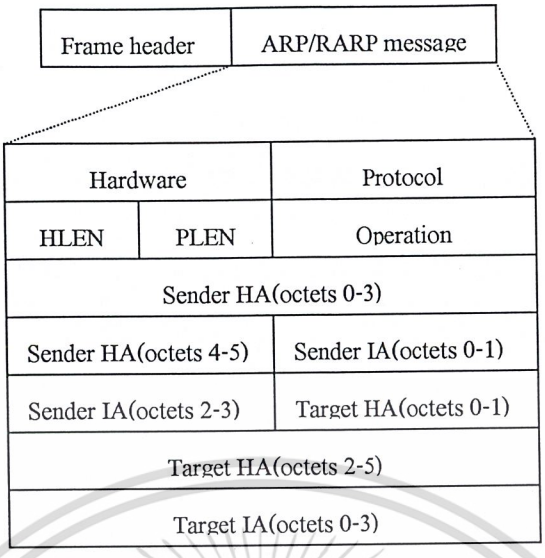
15.3.4 Record route: คือ Option 7 ทำให้แต่ละเราท์เตอร์ใส่ไอพีแอดเดรสของมันในฟิลด์ Option ของคาต้าแกรมเมื่อคาต้าแกรมเดินทางผ่านเครือข่าย ซึ่งทำให้ค้นหาทางที่คาต้าแกรมใช้ในการไปถึงโฮสหรือเราท์เตอร์ใด ๆ ได้

15.3.5 Strict source route: Option 9 คล้ายกับ loose source route ยกเว้นว่าเฉพาะเราท์เตอร์ที่กำหนดในลิสต์เท่านั้นที่สามารถใช้ได้

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 เมื่อ ไอพีจะส่งคาต้าแกรมไปยังไอพีแอดเดรสอื่นมันจะหาแมคแอดเดรสของไอพีแอดเดรสที่เลเยอร์คาต้าลิงค์จำเป็นต้องใช้ในการส่งจากแคชเออาร์พีก่อน ถ้าไม่พบมันจะพยายามหาแมคแอดเดรสจากไอพีแอดเดรสโดยใช้โพรโตคอลเออาร์พี ซึ่งการทำดังกล่าวนี้ โพรโตคอลเออาร์พีจะส่งคาต้าแกรมร้องขอ(ARP request datagram) ไปยังทุกการ์ดแลนโดยใช้แมคแอดเดรสสำหรับการบรอดคาสท์(0xFFFF_FFFF_FFFF) พร้อมทั้งไอพีแอดเดรสของแมคแอดเดรสที่ต้องการ การ์ดในเครือข่ายจะอ่านคำขอนี้และทุกการ์ดที่รู้คำตอบจะตอบกลับ(ARP response) ซึ่งเมื่อได้รับคำตอบคำตอบนี้ก็จะถูกเก็บไว้ในแคชเพื่อใช้ต่อไปในอนาคต แต่ถ้าไม่ได้รับคำตอบภายในเวลาไม่กี่วินาทีเออาร์พีรีเควสก็จะถูกส่งซ้ำ เพราะเออาร์พีอาจถูกละทิ้งได้เนื่องจากความผิดพลาดในการส่งหรือความคับคั่งของบริดจ์(bridge) เพื่อลดความจำเป็นในการบรอดคาสท์เออาร์พี โหนดที่ตอบกลับจะคัดลอกไอพีแอดเดรสและแมคแอดเดรสของผู้ร้องขอเก็บไว้ในแคชเออาร์พีของมันด้วย คาต้าแกรมเออาร์พีนี้ไม่สามารถผ่านเราท์เตอร์ได้เพราะเราท์เตอร์ทำงานที่เลเยอร์ไอพี และไม่รีเลย์(relay) การบรอดคาสท์โดยใช้แมคแอดเดรสซึ่งทำให้ป้องกันการเกิดข้อมูลจำนวนมาก(flooding) วิ่งไปทั่วทั้งระบบได้

รูปแบบของคาต้าแกรมเออาร์พีแสดงดังรูป 2-8 สามารถใช้กับเครือข่ายแบบใดก็ได้ ไม่เฉพาะทีซีพี/ไอพีเท่านั้น แต่ต้องมีตัวกลางที่สามารถส่งเฟรมบรอดคาสท์ได้ เออาร์พีทำงานโดยตรงบนเลเยอร์คาต้าลิงค์ ดังนั้นจึงถูกเอ็นแคปซูเลชั่นโดยแฟรมคาต้าลิงค์เท่านั้น ทำให้มันต้องการฟิลด์ Ethernet type ของมันเองคือ 0x0806



รูปที่ 2-8 รูปแบบดาต้าแกรมเออาร์พี

ฟิลด์ในดาต้าแกรมเออาร์พีประกอบด้วย

1. Hardware บอกถึงชนิดของฮาร์ดแวร์ที่ใช้ในเครือข่ายซึ่งสร้างดาต้าแกรมนี้ขึ้นมา ชนิดที่ใช้ได้คือ

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 ปกติจะมีค่าเป็น 6 สำหรับแมคแอดเดรสของแลนไอทริปเปิ้ลอี

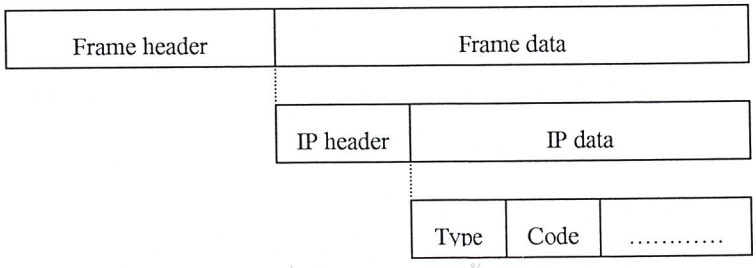
4. **PLEN**	บอกถึงความยาวของแอดเดรสในเลเยอร์เน็ตเวิร์คในหน่วยออคเต็ด ปกติมีค่าเป็น 4 สำหรับ IP
5. **Operation**	มีค่าเป็น 1 สำหรับเออาร์พีรีเควส และ 2 สำหรับเออาร์พีเรสปอนด์(ARP response) และยังใช้กับอาร์เออาร์พีด้วยโดยมีค่าเป็น 3 สำหรับอาร์เออาร์พีรีเควส และ 4 สำหรับอาร์เออาร์พีเรสปอนด์
6. **Addresses**	ประกอบด้วยฮาร์ดแวร์แอดเดรสของผู้ส่ง(แมคแอดเดรสต้นทาง), ไอพีแอดเดรสต้นทาง, ฮาร์ดแวร์แอดเดรสเป้าหมาย(แมคแอดเดรสปลายทาง), และไอพีแอดเดรสปลายทาง

Reverse Address Resolution Protocol(RARP)

ใช้สำหรับอุปกรณ์ที่ไม่สามารถเก็บไอพีแอดเดรสของตัวเองได้ เช่นเวิร์คสเตชั่นที่ไม่มีฮาร์ดดิสค์ อาร์เออาร์พีทำงานในลักษณะตรงกันข้ามกับเออาร์พีคือหาไอพีแอดเดรสจากแมคแอดเดรสที่กำหนด อาร์เออาร์พีทำงานโดยตรงกับเลเยอร์ดาต้าลิงค์โดยมีหมายเลขชนิดอีเธอร์เน็ตเท่ากับ 0x8035 โหนดที่ทำหน้าที่เป็นเซิร์ฟเวอร์อาร์เออาร์พี(RARP server) ที่พบแมคแอดเดรสที่กำหนดจะตอบกลับโดยอาร์เออาร์พีเรสปอนด์พร้อมทั้งไอพีแอดเดรสที่ต้องการ รูปแบบของดาต้าแกรมจะเหมือนเออาร์พีแต่ฟิลด์ Operation จะใช้ค่าเป็น 3 สำหรับรีเควส และ 4 สำหรับเรสปอนด์ ถึงแม้ว่าอาร์เออาร์พีจะทำงานได้ดีแต่ก็มีข้อจำกัดมาก ในทางปฏิบัติจึงถูกแทนที่โดยโพรโตคอลบูท(Boot Protocol: BOOTP) ซึ่งสามารถทำงานผ่านเราท์เตอร์และหาข้อมูลที่เป็นประโยชน์ได้มากกว่าอาร์เออาร์พีเมื่อเวิร์คสเตชั่นที่ไม่มีฮาร์ดดิสค์ทำการบูท

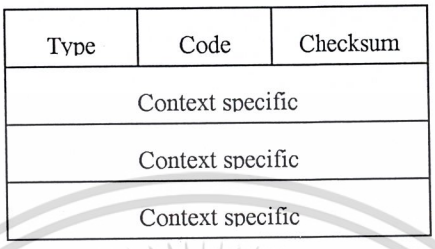
Internet Control Message Protocol(ICMP)

ถึงแม้ว่าไอพีจะไม่รับรองในการส่งข้อมูล แต่ไอซีเอ็มพีซึ่งใช้ได้ในไอพีสามารถสร้างแมสเสจเกี่ยวกับความผิดพลาดเพื่อช่วยเลเยอร์ไอพีในการให้บริการส่งข้อมูลให้ดีที่สุดและยังช่วยผู้ดูแลระบบในการวิเคราะห์หาสาเหตุเกี่ยวกับการทำงานของเครือข่าย ไอซีเอ็มพีใช้ดาต้าแกรมไอพีในการส่งแมสเสจระหว่างโหนด แมสเสจแสดงข้อผิดพลาดของไอซีเอ็มพีจะถูกสร้างโดยโหนดที่พบว่ามีปัญหาในการส่งเกิดขึ้นและจะส่งแมสเสจนี้กลับไปยังแอดเดรสที่เป็นต้นทางของดาต้าแกรมที่ทำให้เกิดปัญหา รูปที่ 2-9 แสดงถึงแมสเสจไอซีเอ็มพีที่ถูกเอ็นแคปซูเลทในดาต้าแกรมไอพีและแมสเสจแบบต่าง ๆ ที่เป็นไปได้ ไอซีเอ็มพีมีหมายเลขโพรโตคอล(protocol number) ของตัวเองซึ่งเท่ากับ 1 ทำให้ไอพีรู้ว่าได้รับไอซีเอ็มพี ถึงแม้ว่าไอซีเอ็มพีจะใช้เลเยอร์ไอพีแต่มันถูกมองว่าอยู่ภายในไอพีทั้งหมดเพราะไม่ได้ให้บริการแก่เลเยอร์ที่อยู่เหนือมัน



รูปที่ 2-9 แสดงไอซีเอ็มพีชนิดต่าง ๆ ถูกเอ็นแคปซูเลทในไอพี

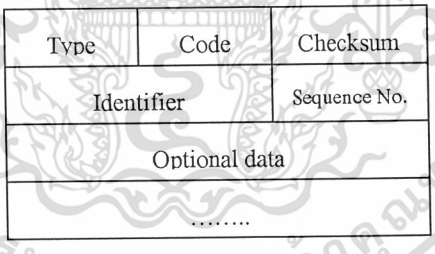
รูปแบบพื้นฐานของคาต้าแกรมไอซีเอ็มพีแสดงได้ดังรูป 2-10 แต่ฟิลด์ต่าง ๆ จะต่างกันซึ่งขึ้นอยู่กับชนิดที่ใช้อยู่ ฟิลด์ Type บ่งถึงชนิดของแมสเสจไอซีเอ็มพี ฟิลด์ Code ใช้แสดงถึงข่าวสารที่ละเอียดมากขึ้น ฟิลด์ Checksum ใช้เพราะไอพีไม่ได้ป้องกันข้อมูลของมันด้วยเช็คซัม(checksum) แต่เมื่อทำงานบนเครือข่ายฟิสิคัลซึ่งมีเฟรมเช็คซีเควินส์(Frame Check Sequence: FCS) เช็คซัมของไอซีเอ็มพีอาจเท่ากับ 0 หมายถึงไม่ถูกคำนวณ



รูปที่ 2-10 เฮดเดอร์พื้นฐานของโพรโตคอลไอซีเอ็มพี

1. ไอซีเอ็มพีชนิด 0 และ 8 – echo

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



รูปที่ 2-11 รูปแบบคาต้าแกรมไอซีเอ็มพีชนิด echo

ฟิลด์ Identifier และ Sequence number ใช้ในการทำให้คาต้าแกรมนี้แตกต่างจากคาต้าแกรมอื่น ถ้ามีข้อมูลส่งในฟิลด์ Optional data มันจะต้องถูกส่งกลับในการตอบกลับ

2. ไอซีเอ็มพีชนิด 3 – destination unreachable

ถ้าเราท์เตอร์ไม่สามารถส่งคาต้าแกรมได้ มันจะส่งแมสเสจไอซีเอ็มพีชนิด destination unreachable เพื่อบอกถึงสาเหตุ ฟิลด์ Code จะถูกใช้บอกถึงสาเหตุ ส่วนเฮดเดอร์อินเตอร์เน็ตรวมทั้งคาต้าแกรมพรีฟิกซ์(datagram prefix) 64 บิตจะใช้ในการบอกถึงคาต้าแกรมที่เป็นสาเหตุของปัญหาดังรูปที่ 2-12

Type	Code	Checksum
Unused (must be 0)		
Internet header+64 bits of datagram prefix		
.....		

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

รูปที่ 2-12 ค่า destination unreachable ในฟิลด์ Code

ซึ่งความหมายของแต่ละเหตุผลเป็นดังนี้

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

2.2 Host unreachable : คาต้าแกรมที่เป็นสาเหตุของความผิดพลาดได้ไปถึงเราท์เตอร์ที่ต่อตรงกับเครือข่ายปลายทางแล้ว แต่เมื่อเราท์เตอร์พยายามที่จะส่งคาต้าแกรมมันกลับไม่สามารถสื่อสารกับโฮสนั้นได้ ซึ่งอาจเกิดจากเออาร์พีล้มเหลวในคาต้าแกรมแรก โฮสดาวน์ หรือเหตุอื่นใดก็ตามซึ่งอาจเพราะไม่มีไอพีแอดเดรสนั้น เช่นเดียวกับ Network unreachable คือแมสเสจนี้จะถูกสร้างจากเราท์เตอร์เท่านั้น จุดที่เกิดข้อผิดพลาดก็คือจุดที่เป็นแอดเดรสต้นทางในเฮดเดอร์ไอพีที่บรรจุแมสเสจไอซีเอ็มพี ซึ่งก็คือเราท์เตอร์ที่พบข้อผิดพลาดนั่นเอง

2.3 Protocol unreachable : ในกรณีนี้คาต้าแกรมได้ไปถึงโฮสปลายทางแล้วแต่ไม่สามารถใช้โพรโตคอลที่ถูกพามาในคาต้าแกรมไอพีได้ 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 : ถูกส่งจากเราท์เตอร์ที่เป็นฮ็อพแรกไปยังโฮสเพื่อบอกว่าไม่สามารถกำหนดลำดับความสำคัญดังที่ร้องขอได้สำหรับโฮสต้นทางหรือปลายทาง เครือข่ายต้นทางหรือปลายทาง โพรโตคอลในเลเยอร์ที่สูงกว่า และพอร์ตต้นทางหรือปลายทาง

2.11 Precedence cut-off in effect : บ่งถึงว่าผู้ดูแลระบบเครือข่ายได้กำหนดค่าระดับความสำคัญที่เส้นทางนี้ต้องการไว้ต่ำสุด คือดาต้าแกรมถูกส่งด้วยลำดับความสำคัญที่ต่ำกว่าที่ต้องการ

3. ไอซีเอ็มพีชนิด 4 และโค้ด 0 – source quence

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

4. ไอซีเอ็มพีชนิด 5 – route change request

มีรูปแบบดังรูปที่ 2-13 ถูกใช้โดยเราท์เตอร์เท่านั้น สำหรับเราท์เตอร์ที่รู้ว่ามันไม่ใช่เราท์เตอร์ที่เหมาะสมที่สุดสำหรับการไปถึงปลายทางที่กำหนดก็จะใช้แมสเสจนี้แนะนำเราท์เตอร์ที่เหมาะสมกว่าแก่ผู้ส่ง คือไอพีแอดเดรสต้นทางของดาต้าแกรม เพื่อความต่อเนื่องในการส่งข้อมูลเราท์เตอร์จะส่งดาต้าแกรมที่เป็นสาเหตุให้เกิดแมสเสจนี้ไปยังเราท์เตอร์ที่เชื่อว่าเข้าถึงเส้นทางที่ดีกว่าด้วย

(a)

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

(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 2-13 (a) รูปแบบของแมสเสจ route change request
(b) ค่าในฟิลด์ Code ที่แมสเสจประเภทนี้ใช้

5. ไอซีเอ็มพีชนิด 9 – router advertisement

ทำให้เราท์เตอร์ประกาศตัวกับโฮสบนระบบเครือข่ายได้ดังรูปที่ 2-14 ซึ่งจะส่งทุก ๆ 7-10 นาที หรือเพื่อตอบสนองโฮสจากแมสเสจไอซีเอ็มพีชนิด 10 แมสเสจนี้ไม่มีข้อมูลว่าจะติดต่อเราท์เตอร์นี้ผ่านเส้นทางไหน เพราะฉะนั้นถ้าโฮสเลือกเราท์เตอร์แรกไม่เหมาะสมก็จะได้รับแมสเสจไอซีเอ็มพีชนิด 5

Type	Code	Checksum
Num addrs	Addr entry size	Life time
Router address [1]		
Preference level [1]		
Router address [2]		
Preference level [2]		
.....		

รูปที่ 2-14 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 2-15

Type	Code	Checksum
Reserved		

รูปที่ 2-15 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 บอกถึงการพยายามรวมส่วนย่อยเพื่อให้เป็นดาต้าแกรมเดิมไม่สำเร็จ การรวมดาต้าแกรมอีกครั้งโดยใช้เวลามากจนเกินไปถ้าเกิดขึ้นไม่บ่อยนักก็ไม่ถือว่าเป็นปัญหาร้ายแรงแต่อย่างใด

8. ไอซีเอ็มพีชนิด 12 – parameter problem

ใช้อาร์กิวเมนท์ผิดในฟิลด์ Option ของเฮดเดอร์ไอพี แต่ถ้าร้ายแรงกว่านั้นก็คือเกิดข้อผิดพลาดในการทำงานของไอพี มันแสดงถึงว่ามีค่าในเฮดเดอร์ที่ไม่สามารถเข้าใจได้ แมสเสจนี้จะไม่ถูกส่งถ้าดาต้าแกรมไม่ถูกละทิ้ง ฟิลด์ pointer บ่งถึงตำแหน่งของอ็อคเต็ดที่สงสัย เช่น ถ้ามีค่าเป็น 1 ก็หมายถึงฟิลด์ TOS ถ้ามีค่าเป็น 20 ก็จะหมายถึงอ็อคเต็ดแรกของฟิลด์ Option เป็นต้น รูปแบบของแมสเสจชนิดนี้เป็นดังรูปที่ 2-16

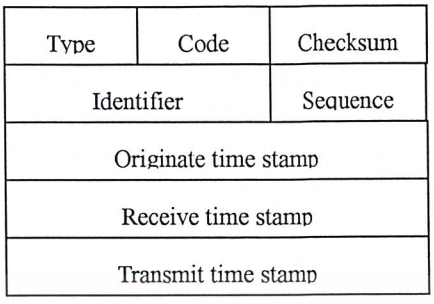
Type	Code	Checksum
Pointer	Unused (must be 0)	
Internet header+64 bits of datagram prefix		
.....		

รูปที่ 2-16 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 2-17 โดยแมสเสจชนิดนี้จะถูก

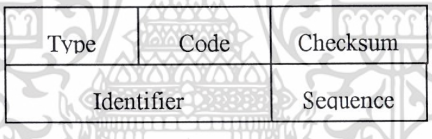
ใช้สำหรับเก็บสถิติเกี่ยวกับสมรรถนะของการเชื่อมต่อไปยังโฮสหรือเพื่อการเข้าจังหวะกันของนาฬิกาในโฮส



รูปที่ 2-17 รูปแบบแมสเสจไอซีเอ็มพี time stamp request/reply

10. ไอซีเอ็มพีชนิด 15 และ 16 – information request

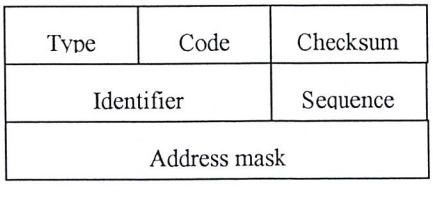
โฮสใช้เพื่อหาหมายเลขของเครือข่าย(network number) ถ้าโฮสนั้นไม่รู้ แอดเดรสที่ใช้ในเฮดเดอร์ไอพีจะมีค่าเป็น 0 หมายถึงเครือข่ายนี้ ซึ่งจะถูกเติมอย่างถูกต้องโดยปลายทางและถูกส่งกลับมา รูปแบบของแมสเสจไอซีเอ็มพีชนิดนี้เป็นดังรูป 2-18 กลไกนี้ใช้กับระบบไดอัลอิน(dial-in) ที่ใช้สลิป(SLIP) เป็นวิธีในการกำหนดเน็ตเวิร์คแอดเดรสที่เหมาะสมให้กับแต่ละปลายของการเชื่อมต่อ



รูปที่ 2-18 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 2-19



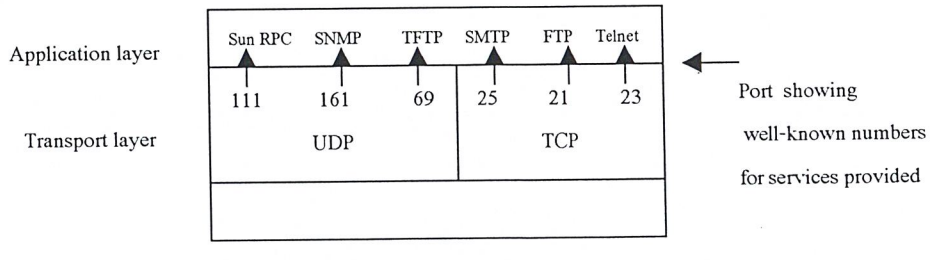
รูปที่ 2-19 รูปแบบแมสเสจไอซีเอ็มพี address mask request

2.2.2 เลเยอร์ทรานสปอร์ตของทีซีพี/ไอพี

บริการที่ไอพีมีให้ยังต้องการให้ดาต้าแกรมส่งไปยังบริการในเลเยอร์แอพพลิเคชั่นที่เหมาะสมโดยตรงและต้องการบริการที่เชื่อถือได้สำหรับแอพพลิเคชั่นที่จำเป็นต้องใช้มัน ซึ่งหน้าที่เหล่านี้เป็นของเลเยอร์นี้ทำโดยโพรโตคอลทรานสปอร์ต 2 ตัวคือ โพรโตคอลยูดีพี(User Datagram Protocol: UDP) และโพรโตคอลทีซีพี(Transmission Control Protocol: TCP) ซึ่งการเลือกใช้นั้นขึ้นอยู่กับประเภทของบริการที่แอพพลิเคชั่นของผู้ใช้ต้องการ

เมื่อข้อมูลถูกส่งไปยังเครื่องที่ต้องการโดยไอพีมันจะถูกส่งไปยังบริการแอพพลิเคชั่นที่เกี่ยวข้องบนเครื่องนั้น ๆ การมัลติเพล็กซ์และดีมัลติเพล็กซ์ข้อมูลไปยังหรือจากเลเยอร์ไอพีและส่งข้อมูลไปยังแอพพลิเคชั่นที่ถูกต้องเป็นหน้าที่อย่างหนึ่งของเลเยอร์ทรานสปอร์ตนี้ รวมทั้งการทำให้ไม่มีข้อผิดพลาด(error-free) และบริการส่งข้อมูลไปยังแอพพลิเคชั่นที่ถูกต้องแบบต้องก่อตั้งหรือไม่ก่อตั้งการเชื่อมต่อก่อนก็เป็นหน้าที่ของเลเยอร์นี้เช่นกัน โพรโตคอลยูดีพีจะให้บริการแบบไม่ต้องก่อตั้งการเชื่อมต่อก่อน (connectionless) ซึ่งเป็นบริการที่ไม่มีความน่าเชื่อถือ เพราะว่ามันจะอนุญาตให้ส่งข้อมูลไปยังเครื่องหรือกลุ่มของเครื่องโดยไม่จำเป็นต้องก่อตั้งการเชื่อมต่อก่อนดังนั้นดาต้าแกรมหนึ่งจะถูกส่งไปยังโหนดอื่นได้โดยไม่ต้องการการตอบสนองว่าดาต้าแกรมดังกล่าวได้ไปถึงแล้วหรือยัง 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) ทีซีพีมีโอเวอร์เฮดมากกว่ายูดีพีในแง่ของการประมวลผลที่ต้องการและขนาดของเฮดเดอร์ที่ต้องใช้ 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 2-20 หมายเลขพอร์ตที่ใช้ในยูดีพีและทีซีพี

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

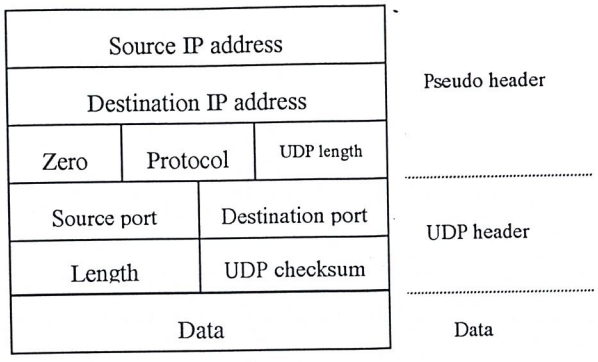
โพรโตคอลยูดีพี(User Datagram Protocol: UDP)

ยูดีพีเพิ่มความสามารถเข้าไปในการให้บริการของไอพีเพียงเล็กน้อย และใช้ฟิลด์ Source Port และ Destination Port ในเฮดเดอร์ของมันเพื่อส่งข้อมูลไปยังบริการในเลเยอร์แอพพลิเค ชั่น ฟิลด์ต่าง ๆ ในโพรโตคอลยูดีพีเป็นดังรูป 2-21 ประกอบด้วย

Source port	Destination port
Length	UDP checksum
Data	

รูปที่ 2-21 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**	เป็นการป้องกันข้อมูลที่ยูดีพีบรรทุกมาซึ่งคิดทั้งฟิลด์ยูดีพีและซูโดเฮดเดอร์ (pseudo header) ดังรูป 2-22 มีขนาด 16 บิต



รูปที่ 2-22 ค่าเช็คซัมประกอบด้วยเฮดเดอร์และซูโดเฮดเดอร์

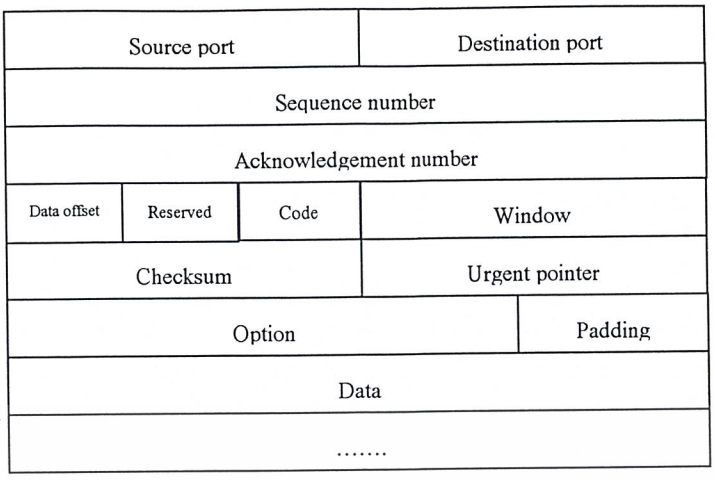
โพรโตคอลทีซีพี(Transmission Control Protocol: TCP)

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

- การตรวจและแก้ไขข้อผิดพลาด: ซึ่งเกี่ยวข้องกับความเป็นไปได้ที่เซกเมนท์จะเสียหายจากสายสื่อสารหรือซอฟต์แวร์ในเลเยอร์ที่สูงกว่า
- โฟลวคอนโทรล: ใช้ในการป้องกันไม่ให้ผู้ส่งทำให้ผู้รับประสบปัญหาเนื่องมาจากข้อจำกัดในทรัพยากร
- การจัดลำดับการส่ง: จำเป็นต้องมีเพราะเลเยอร์ไอพีสามารถส่งดาต้าแกรมซึ่งมีเซกเมนท์ทีซีพีในลำดับใดก็ได้ ซึ่งเกิดขึ้นเมื่อดาต้าแกรมถูกส่งคนละเส้นทาง
- การกำจัดเซกเมนท์ที่ซ้ำ: เกิดเพราะกลไกกู้คืน(error-recovery) ที่ทีซีพีใช้

ทีซีพีได้ทำการเพิ่มความสามารถที่กล่าวข้างต้นได้โดย

- ใช้หมายเลขแสดงลำดับในการแยกแยะข้อมูล
- ต้องได้รับเอคโนวเลดเมนท์ของการส่งข้อมูลในลำดับที่ถูกต้อง
- มีการส่งเซกเมนท์ซ้ำเมื่อไม่ได้รับการตอบกลับในเวลาที่กำหนด



รูปที่ 2-23 เฮดเดอร์ทีซีพี

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

1. **Sequencing number** มีขนาด 32 บิต ในทีซีพีจะนับออคเต็ดในการส่ง แต่โพรโตคอลอื่นที่ใช้เลขลำดับเพื่อควบคุมความผิดพลาดจะนับเซ็กเมนท์ เลขลำดับในเฮดเดอร์นี้จะกำหนดตำแหน่งในข้อมูลทั้งหมดของออคเต็ดแรกในเซ็กเมนท์ ซึ่งช่วยในทีซีพีใส่เซ็กเมนท์ในตำแหน่งที่ถูกต้องในข้อมูลได้ แม้ไอพีจะส่งข้อมูลไม่เป็นลำดับก็ตาม การที่มันมี 32 บิตทำให้ไม่เกิดการซ้ำของค่าแม้เวลาในการส่งจะเร็วมากก็ตาม คือเซ็กเมนท์ที่ได้รับอาจมีเลขลำดับซ้ำกับเซ็กเมนท์ซึ่งแอคโนวเลดไปเรียบร้อยแล้วได้ แต่ถ้าเกิดข้อผิดพลาดเนื่องจากการซ้ำก็ไม่เป็นปัญหา เพราะแค่ไม่สนใจโดยไม่ต้องทำอะไร

2. **Acknowledgement number** มีขนาด 32 บิต บอกให้รู้ว่าได้รับออคเต็ดทั้งหมดอย่างถูกต้องจนถึงเลขแอคโนวเลดลบด้วย 1 เมื่อผู้ส่งได้รับค่านี้ก็ไม่จำเป็นต้องเก็บข้อมูลไว้เพื่อส่งใหม่อีกต่อไป 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 ได้ซึ่งฟิลด์นี้ชี้ไปยังออคเต็ดในฟิลด์ข้อมูลซึ่งเป็นปลายของข้อมูล urgent ซึ่งไม่ถูกมองเป็นข้อมูลปกติและควรจะถูกประมวลผลก่อนข้อมูลอื่น ๆ
- ACK บอกว่าฟิลด์ Acknowledge ใช้ได้ ซึ่งฟิลด์นี้จะใช้ไม่ได้เมื่อก่อตั้งการเชื่อมต่อ คือก่อนที่แต่ละโหนดจะสามารถตัดสินใจได้ว่าจะใช้ค่า sequence และ acknowledge ใด
- PSH คือแฟลก push ซึ่งทำให้เลเยอร์ทีซีพีที่ระยะไกลส่งเซ็กเมนท์นี้ไปให้เลเยอร์แอพพลิเคชั่นอย่างทันทีทันใด ปกติทีซีพีจะหันมาสนใจข้อมูลจากเซ็กเมนท์ที่เข้ามาและส่งข้อมูลนี้ไปยังเลเยอร์แอพพลิเคชั่นในบัฟเฟอร์ที่ใหญ่กว่าเพื่อลดโอเวอร์เฮดในการประมวลผล
- RST คือแฟลก reset ใช้เมื่อเกิดข้อผิดพลาดอื่น ๆ บอกถึงว่ามีข้อผิดพลาดเกิดขึ้นและการควรจะหยุดการติดต่อ
- SYN คือแฟลก synchronize ใช้ขณะเริ่มต้นการก่อตั้งการเชื่อมต่อระหว่าง 2 โหนดซึ่ง ณ เวลานั้นทั้ง 2 โหนดไม่รู้ว่าควรจะใช้เลขแอคโนวเลดใด การก่อตั้งการเชื่อมต่อจะประกอบด้วยการแลกเปลี่ยนเซ็กเมนท์แบบ 2 ทาง(2-way exchange of segment) พร้อมทั้งเซ็ตแฟลก SYN ซึ่งแต่ละอันจะถูกแอคโนวเลดในเซ็กเมนท์โดยเซ็ตแฟลก ACK
- FIN ใช้ในการเลิกการติดต่อเมื่อข้างใดข้างหนึ่งไม่มีข้อมูลที่จะส่งก็จะส่งเซ็กเมนท์ที่มีการเซ็ตแฟลก 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 ค่าในฟิลด์นี้ชี้ไปยังปลายของข้อมูลในฟิลด์ข้อมูลที่เร่งด่วนและต้องการความสนใจทันที จะใช้ได้เมื่อมีการเซ็ตแฟลก URG

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

9. **Padding** ถ้าใช้ฟิลด์ออปชั่นแพดดิ้ง จะทำให้มั่นใจได้ว่าข้อมูลเริ่มที่ขอบ 32 บิตอย่างที่ออปเซ็ตข้อมูลชี้อย่างถูกต้อง

2.2.3 เลเยอร์แอพพลิเคชั่นของทีซีพี/ไอพี

บริการของเลเยอร์แอพพลิเคชั่นจะรับผิดชอบในการเชื่อมต่อ(interface) ระหว่างแอพพลิเคชั่นของผู้ใช้ และบริการในชั้นทรานสปอร์ต บริการของแอพพลิเคชั่นไม่ใช่แอพพลิเคชั่นของผู้ใช้แต่เป็นการเชื่อมต่อกับแอพพลิเคชั่นนั้นกับเครือข่ายสื่อสาร มีบริการของแอพพลิเคชั่นหลายตัวที่เหมาะสมกับแอพพลิเคชั่นหลายชนิด และยังมียูทิลิตี้ในการจัดการอีกจำนวนหนึ่ง ตัวอย่างของบริการในชั้นนี้คือ

1. **FTP** ย่อมาจาก File Transfer Protocol ใช้พอร์ตหมายเลข 20 เป็นโพรโตคอลมาตรฐานและเป็นวิธีที่ง่ายที่สุดในการแลกเปลี่ยนไฟล์กันในอินเทอร์เน็ต FTP เป็นโพรโตคอลแอพพลิเคชั่นที่ใช้โพรโตคอลชุดทีซีพี/ไอพี และมักจะถูกใช้เสมอในการส่งไฟล์เว็บเพจจากผู้สร้างเว็บเพจไปยังคอมพิวเตอร์ที่ทำตัวเป็นเซิร์ฟเวอร์เพื่อให้ใครก็ตามในอินเทอร์เน็ตสามารถใช้ได้ 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 หรือกล่าวได้อีกอย่างว่าปกติแล้วผู้ใช้จะใช้โปรแกรมที่ใช้ 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/เซิร์ฟเวอร์ ซึ่งจะรับอีเมล์ที่ถูกส่งมาและเก็บเอาไว้ให้ในเซิร์ฟเวอร์ของเรา เราสามารถดูเมล์ได้ที่เซิร์ฟเวอร์และดาวน์โหลดได้ นอกจาก POP3 แล้วยังมีโพรโตคอลที่ทำงานคล้ายคลึงกันคือโพรโตคอล IMAP (Interactive Mail Access Protocol) ด้วย IMAP เราสามารถดูอีเมล์ที่เซิร์ฟเวอร์เหมือนกับว่าอีเมล์เหล่านั้นอยู่บนเครื่องคอมพิวเตอร์ของเราเอง 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 คือโพรโตคอลที่ใช้บริหารการจัดการเครือข่ายและการมอนิเตอร์อุปกรณ์ในเครือข่ายและฟังก์ชันของอุปกรณ์เหล่านั้น 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)

2.3 เครือข่ายท้องถิ่นแบบอีเธอร์เน็ต(Ethernet LAN)

ก่อนที่จะอธิบายถึงรายละเอียด เราควรจะรู้ก่อนว่าทำไมเครือข่ายท้องถิ่นแบบนี้มีข้อดีข้อเสียเป็นอย่างไร

ข้อดี :-

1. ง่ายในการติดตั้ง เพราะทุกสเตชั่นสามารถต่อเข้ากับเซ็กเมนท์(segment) ได้โดยใช้คอนเน็คเตอร์แบบที(T-connector) หรือทรานซีฟเวอร์(transceiver) ง่าย ๆ และเครือข่ายท้องถิ่นแบบอีเธอร์เน็ตยังไม่ต้องการฮับ(hub) ในการเชื่อมต่อระบบอีกด้วย
2. เป็นเทคโนโลยีที่รู้จักกันดี เนื่องจากในภาคธุรกิจได้ติดตั้งอีเธอร์เน็ตเป็นเวลาหลายปีแล้ว
3. ราคาการ์ดได้ลดลงทำให้การ์ดมีราคาไม่แพง
4. มีการเชื่อมโยงสายได้หลายแบบ

ข้อเสียหลัก :-

1. ทรูพุธ(throughput) จะลดลงเมื่อมีโหลดในแลนมาก เนื่องจากใช้วิธีในการเข้าถึงสายสื่อสารแบบซีเอสเอ็มเอ/ซีดี(CSMA/CD)
2. เนื่องจากการที่ระบบมีการใช้สารสื่อสารร่วมกันทำให้ยากในการค้นหาสาเหตุเมื่อเกิดข้อผิดพลาดขึ้น(troubleshooting) เช่นถ้าเคเบิ้ลเสียหายเซกเมนท์ของแลนทั้งหมดจะใช้งานไม่ได้(down) และยากมากในการที่จะแยกโหนดที่เป็นต้นเหตุของปัญหาได้ ถึงแม้ว่าเท็นเบสที(10BaseT) จะแก้ปัญหาเหล่านี้ได้บางปัญหาก็ตาม

วิธีที่แลนแบบนี้ใช้ในการเข้าถึงสายสื่อสาร(access method) คือแบบซีเอสเอ็มเอ/ซีดี(Carrier Sense Multiple Access with Collision Detection: CSMA/CD) ซึ่งวิธีเข้าถึงสายสื่อสารนี้จะเป็นตัวกำหนดชนิดของการ์ดที่ใช้ในการอินเทอร์เฟส(interface card) ที่จะต้องติดตั้งไว้ในเวิร์คสเตชั่น และบอกถึงวิธีที่เวิร์คสเตชั่นจะเข้าถึงระบบเคเบิ้ลหรือสายสื่อสาร รวมทั้งวิธีที่ข้อมูลถูกเตรียมเพื่อส่งและถูกส่งออกไป อย่างไร ก่อนที่เราจะทำงานกับอีเธอร์เน็ตได้เราควรจะรู้ว่าวิธีในการเข้าถึงสายสื่อสารที่แลนแบบนี้ใช้เป็นอย่างไร ซึ่งจะช่วยให้เข้าใจถึงสถิติ, ข้อผิดพลาด, และระดับการใช้งานของแลนได้ การทำงานของซีเอสเอ็มเอ/ซีดีอธิบายได้ดังต่อไปนี้

<u>การส่ง</u>(transmitting)

เนื่องจากแลนแบบนี้ใช้ระบบเคเบิ้ลร่วมกัน ดังนั้นจึงต้องมีกฎในการหลีกเลี่ยงไม่ให้มีการส่งพร้อมกัน อย่างไรก็ตามถ้าสเตชั่นหลายสเตชั่นส่งข้อมูลพร้อมกันก็จะต้องมีวิธีที่จะรู้ว่าแพ็คเกจของมันถูกชนหรือไม่และเมื่อไหร่จะเริ่มส่งใหม่ ซึ่งสเตชั่นจะทำตามวิธีดังนี้เมื่อจะส่งข้อมูล

ขั้นตอนที่ 1 : ฟังก่อนส่ง(Listen before transmitting)

สเตชั่นจะคอยตรวจดูเคเบิ้ลว่ามีสัญญาณของสัญญาณพาหะหรือไม่ โดยสัญญาณในเคเบิ้ลจะตรวจพบได้โดยการวัดโวลเตจ(voltage) ที่บ่งถึงการใช้งานเคเบิ้ล ถ้าสเตชั่นไม่พบว่ามีการใช้งานอยู่มันก็จะสรุปว่าเคเบิ้ลว่างและจะเริ่มทำการส่ง แต่ถ้าไม่ว่างเมื่อสเตชั่นทำการส่งข้อมูล ข้อมูลก็จะถูกชนกับสัญญาณที่มีในสายขณะนั้น

ขั้นตอนที่ 2 : รอถ้าพบว่าเคเบิ้ลไม่ว่าง(Defer if the cable is busy)

เพื่อหลีกเลี่ยงการชน สเตชั่นจะต้องรอถ้าพบว่าเคเบิ้ลถูกใช้งานอยู่ ซึ่งจะรอเป็นระยะเวลาหนึ่งจนกระทั่งสายว่างก่อนที่จะพยายามส่งใหม่อีกครั้ง

ขั้นตอนที่ 3 : ส่งและรอฟังการชน(Transmit and listen for collisions)

เมื่อสายสื่อสารว่างเป็นเวลาอย่างน้อย 9.6 มิลลิวินาที สเตชั่นอาจจะส่งข้อมูล ถ้าสเตชั่นอื่น ๆ ในเซ็กเมนท์ก็ทำการส่งแพ็คเกจในเวลาเดียวกันก็จะเกิดการชนกันในสายสื่อสาร แพ็คเกจที่เกี่ยวข้องในการชนคือแพ็คเกจส่วนที่อยู่บนสายสื่อสารเท่านั้น ดังนั้นช่วงที่ทำการส่ง สเตชั่นจะคอยดูว่าเกิดการชนหรือไม่ โดยจับสัญญาณในเคเบิ้ลว่าเท่ากับหรือมากกว่าสัญญาณที่เกิดจากทรานซีฟเวอร์(transceiver) 2 ตัวหรือมากกว่าทำการส่งในเวลาเดียวกัน โดยทรานซีฟเวอร์คืออุปกรณ์ไฟฟ้าที่ส่งและรับข้อมูลบนสายสื่อสาร ถ้าเกิดการชนแต่สเตชั่นอื่นไม่รู้ว่ามีเหตุการณ์นี้เกิดขึ้นก็อาจจะพยายามส่งด้วย ซึ่งสเตชั่นดังกล่าวก็จะทำให้เกิดการชนอีก เพื่อหลีกเลี่ยงเหตุการณ์นี้สเตชั่นที่เกี่ยวข้องกับการชนก็จะส่งสัญญาณแจม(jam) เพื่อบอกให้สเตชั่นอื่นรู้ว่าขณะนี้เคเบิ้ลไม่ว่างดังรูปที่ 2.1.3.5 สัญญาณแจมคือการส่งข้อมูลออกไปอย่างน้อย 32 บิตที่ไม่เท่ากับค่าซีอาร์ซีของการส่งก่อนหน้านี้ หลังจากมีการชนสเตชั่นที่เกี่ยวข้องก็จะเพิ่มค่าพยายามส่ง(transmit attempt) ขึ้นอีกหนึ่ง

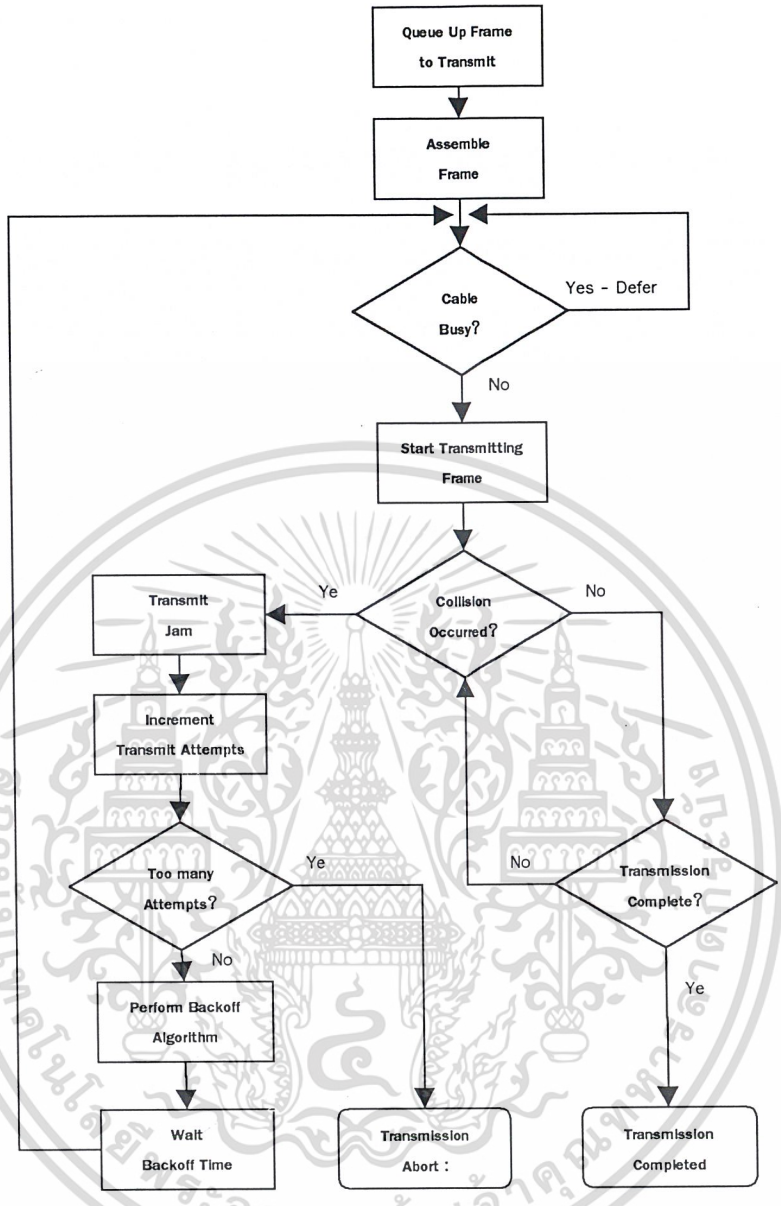
ขั้นตอนที่ 4 : รอก่อนจะส่งอีกครั้ง(Wait before retransmitting)

ถ้าสเตชั่นทำการส่งใหม่ทันทีหลังจากเกิดการชน ก็อาจจะเกิดการชนครั้งที่ 2 ได้อีก จึงจำเป็นต้องสุ่ม(random) เวลาที่สเตชั่นจะต้องรอก่อนพยายามส่งอีกครั้งหนึ่ง สเตชั่นจะใช้อัลกอริธึมแบคออฟ(backoff algorithm) เพื่อที่จะรู้ว่าเมื่อไหร่จะสามารถส่งใหม่ได้ซึ่งจากอัลกอริธึมดังกล่าวจะได้เวลาที่สามารถจะใช้ได้จำนวนมาก เมื่อสเตชั่นสุ่มเลือกเวลาที่จะใช้ได้แล้ว จะทำให้เกิดโอกาสน้อยลงที่จะเกิดการชนกันของข้อมูลจากสองสเตชั่นขึ้นไป

ขั้นตอนที่ 5 : ส่งใหม่หรือเลิกส่ง(Retransmit or Abort)

ถ้าสเตชั่นอยู่บนเซ็กเมนท์ที่มีการใช้งานมาก(busy) อาจทำให้ไม่สามารถส่งโดยไม่มีการชนเกิดขึ้นได้ ซึ่งสเตชั่นจะพยายามส่งจนถึง 16 ครั้งจึงจะหยุดส่ง ถ้าสเตชั่นส่งใหม่แล้วพบว่าไม่มีการชนเกิดขึ้นอีกจะถือว่าส่งสำเร็จ ถ้าส่งไม่สำเร็จเป็นจำนวน 16 ครั้งก็จะเลิกส่ง

เมื่อเข้าใจวิธีที่แต่ละสเตชั่นใช้ในการเข้าถึงสายสื่อสารแล้ว ก็ทำให้เรารู้ได้ว่าสเตชั่นส่งข้อมูลมีประสิทธิภาพหรือทำตามกฎหรือไม่ โฟลวชาร์ตที่แสดงในรูป 2-24 ได้กำหนดแต่ละขั้นตอนที่ทุกสเตชั่นต้องทำเพื่อส่งเฟรมไปบนเครือข่ายแลนซึ่งถ้ามีสเตชั่นใดไม่ทำตามขั้นตอนเหล่านี้ก็จะทำให้เกิดปัญหาขึ้นในเซ็กเมนท์ที่มันใช้งานอยู่ได้



รูปที่ 2-24 โฟลวชาร์ตแสดงการส่ง

<u>การรับ</u>(Receiving)

แต่ละสเตชั่นที่ใช้งานอยู่(active) จะต้องทำงานตามขั้นตอนนี้

ขั้นตอนที่ 1: ดูแพ็คเกจที่เข้ามาและตรวจว่าถูกแบ่งส่วนหรือไม่(View incoming packets and check for fragments)

ทุกสเตชั่นบนเซ็กเมนท์จะคอยดูแต่ละแพ็คเกจที่อยู่บนสายสื่อสารโดยไม่สนใจว่าแพ็คเกจนั้นส่งถึงมันหรือไม่ และสเตชั่นจะตรวจขนาดแพ็คเกจด้วยว่ามีขนาดปกติและไม่ถูกแบ่งส่วนย่อยเนื่องมาจากการชนหรือไม่

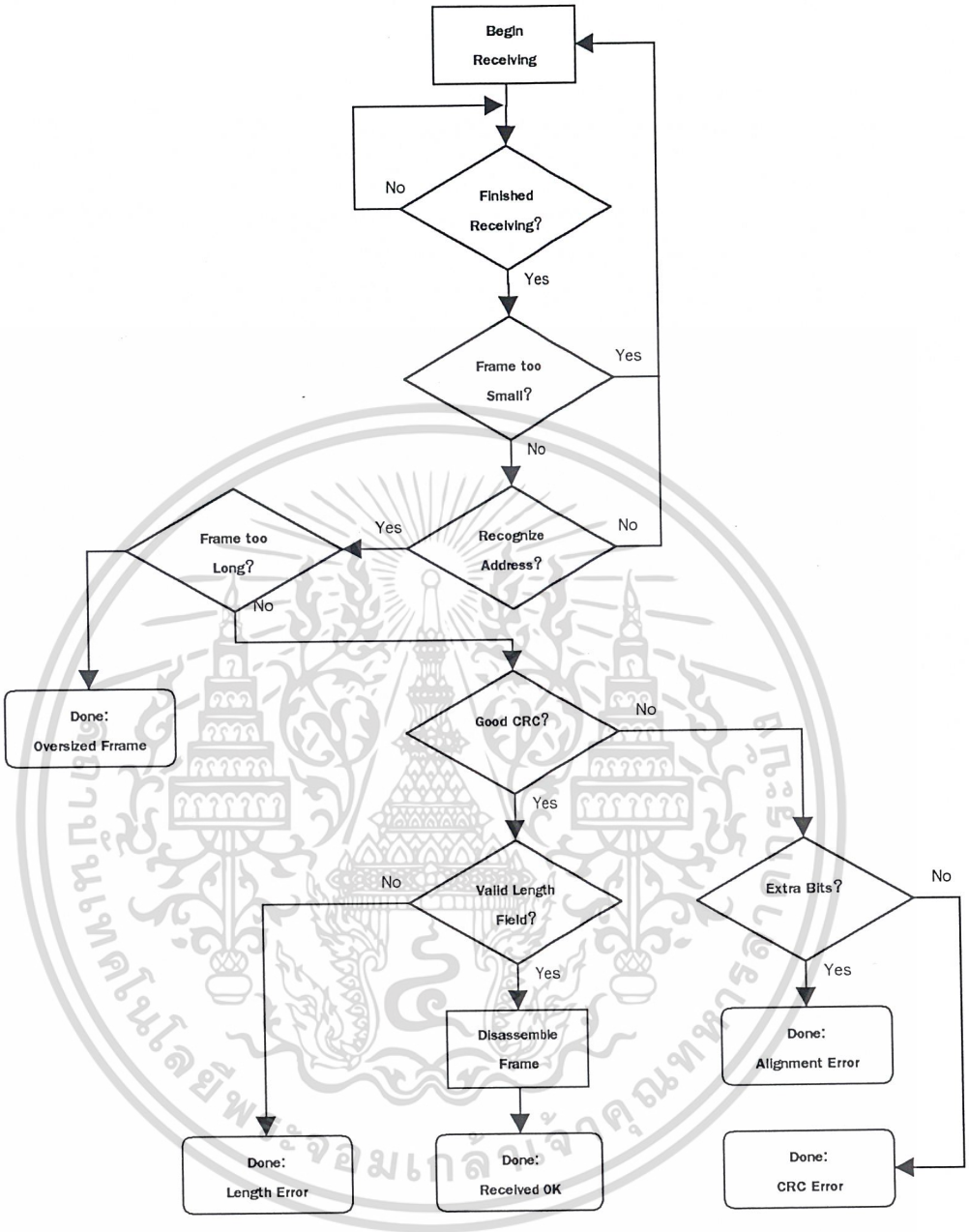
ขั้นตอนที่ 2 : ตรวจแอดเดรสปลายทาง(Check the destination address)

หลังจากพบว่าแพ็คเกจไม่ถูกแบ่งย่อย สเตชั่นก็จะตรวจสอบแอดเดรสปลายทางของแพ็คเกจนั้น เพื่อดูว่าแพ็คเกจนั้นควรถูกประมวลผลต่อไปหรือไม่ ถ้าแพ็คเกจมีแอดเดรสปลายทางเป็นของสเตชั่นนั้น, บรอดคาสท์ หรือมัลติคาสท์ สเตชั่นก็จะตรวจความถูกต้องสมบูรณ์ของแพ็คเกจต่อไป

ขั้นตอนที่ 3 : ตรวจความถูกต้องสมบูรณ์ของแพ็คเกจ(Check the integrity of the packet)

ณ จุดนี้สเตชั่นรู้แล้วว่าแพ็คเกจไม่ถูกแบ่งย่อยและมีแอดเดรสปลายทางเป็นของมัน หรือเป็นแอดเดรสที่มันเป็นส่วนหนึ่งของกลุ่มด้วย แต่สเตชั่นไม่รู้ว่าแพ็คเกจมีรูปแบบที่เหมาะสมหรือไม่ เพราะสเตชั่นอาจจะกำลังอ่านแพ็คเกจที่ถูกทำให้เสียหายบนเซ็กเมนท์หรือถูกจัดรูปแบบไม่เหมาะสมโดยสเตชั่นที่สร้างแพ็คเกจนั้นขึ้นมา(เป็นผู้ส่ง) อยู่ก็ได้ ดังนั้นเพื่อหลีกเลี่ยงที่จะประมวลผลแพ็คเกจเหล่านี้ สเตชั่นที่ได้รับแพ็คเกจจะต้องทำการตรวจคุณลักษณะต่าง ๆ ของแพ็คเกจก่อน โดยอันดับแรกจะต้องทำการตรวจว่าแพ็คเกจนั้นมีความยาวปกติหรือไม่ ถ้ายาวมากกว่า 1,518 ไบต์ จะถือว่าเป็นเฟรมที่ยาวมากเกินไป(oversized frame) ซึ่งอาจเกิดจากความผิดพลาดของไดรเวอร์แวน(WAN driver) ถ้าแพ็คเกจมีขนาดไม่ยาวไปกว่าปกติแล้วก็จะถูกตรวจสอบต่อไปว่าข้อมูลที่มันนำมาด้วยนั้นเหมือนกับตอนที่ถูกส่งหรือไม่ เพราะอาจมีการสลับบิต 0 เป็น 1 หรือบิต 1 เป็น 0 ขณะถูกส่งมาในสายสื่อสารก็ได้ ซึ่งการตรวจสอบนั้นสามารถทำได้โดยตรวจซีอาร์ซี(Cyclic Redundancy Check: CRC) ถ้าพบว่าซีอาร์ซีผิดพลาด แพ็คเกจนั้นก็จะถูกตรวจต่อไปว่าจบที่ขอบ 8 บิตหรือไม่ ถ้าไม่(misaligned packet) จะถือว่าเป็นข้อผิดพลาดที่เกิดจากมีบางไบต์ไม่ครบ 8 บิต(Alignment error) แต่ถ้าเป็นแพ็คเกจที่ทุกไบต์ครบ 8 บิตก็จะถือว่าเป็นแพ็คเกจที่มีข้อผิดพลาดจากซีอาร์ซี(CRC error) ถ้าแพ็คเกจสามารถผ่านการตรวจที่กล่าวมาทั้งหมดได้ สเตชั่นก็จะทำการตรวจสอบความยาวอีกครั้งหนึ่งเพื่อดูว่าแพ็คเกจมีขนาดสั้นเกินไปหรือไม่ ถ้าสั้นกว่า 64 ไบต์แต่มีรูปแบบปกติจะถือว่ามีขนาดสั้นกว่าปกติ(undersized) ซึ่งอาจเกิดจากไดรเวอร์แลน(LAN driver) การตรวจสอบที่กล่าวมาทั้งหมดนี้เพื่อให้แน่ใจได้ว่าแพ็คเกจที่ได้รับมีขนาดและข้อมูลปกติก่อนที่มันจะถูกประมวลผลต่อไป ถ้าเฟรมไม่ผ่านการตรวจจุดใดก็ตามมันจะไม่ถูกส่งขึ้นไปให้โพรโตคอลในเลเยอร์ที่สูงกว่าเพื่อทำงานต่อไป

โฟลวชาร์ตดังรูปที่ 2-25 กำหนดขั้นตอนที่สเตชั่นจะต้องทำก่อนการประมวลผลแพ็คเกจที่ได้รับมาในเครือข่ายที่ใช้วิธีเข้าถึงสายสื่อสารแบบซีเอสเอ็มเอ/ซีดี



รูปที่ 2-25 โฟลวชาร์ตแสดงการรับ

ขั้นตอนที่ 4 : ประมวลผลแพ็คเกจ(Process the packet)

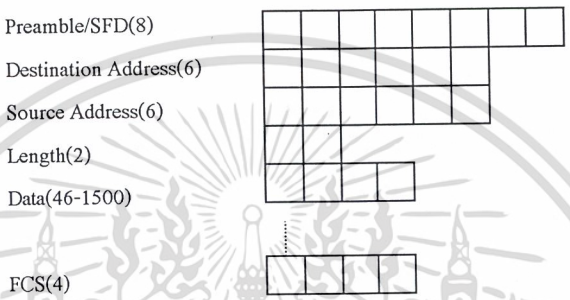
ถ้าแพ็คเกจผ่านการตรวจสอบทุกจุดก็จะถือว่าเป็นแพ็คเกจที่ปกติสามารถนำมาใช้งานได้ ถ้ายังพบปัญหาอีกเราจะต้องดูเข้าไปในแพ็คเกจอีกเพื่อหาปัญหาซึ่งบางทีอาจเป็นเพราะสเตชั่นใช้ชนิดเฟรม(frame type) ผิดหรือมีข้อผิดพลาดที่เฮดเดอร์

2.3.1 เฟรมอีเธอร์เน็ตแบบต่าง ๆ

ข้อมูลจำเป็นจะต้องถูกเอ็นแคปซูเลทในเฟรมเมื่อสเตชั่นเข้าถึงสายสื่อสาร ซึ่งเฟรมจะมีวิธีในการเข้าจังหวะ(synchronizing) กับสเตชั่นที่รับข้อมูล รวมทั้งกำหนดหรือเก็บข่าวสารว่าใครคือต้นทาง, ปลายทาง และโพรโตคอลในเลเยอร์ที่สูงกว่าโพรโตคอลใดที่ใช้เฟรมนี้ โครงสร้างของเฟรมที่ใช้ในอีเธอร์เน็ตจะมีอยู่ 4 ลักษณะดังนี้

2.3.1.1 อีเธอร์เน็ต 802.3

มีโครงสร้างดังรูปที่ 2-26 ซึ่งประกอบด้วยฟิลด์ต่าง ๆ ดังต่อไปนี้



รูปที่ 2-26 เฟรมอีเธอร์เน็ต 802.3

1. **Preamble** และ **Start Frame Delimiter(SFD)** ฟิลด์ Preamble มีขนาด 7 ไบต์ใช้เพื่อเข้าจังหวะกับสเตชั่นที่รับข้อมูลซึ่งจะมีค่าเป็น 1 และ 0 สลับกัน(10101010.....) ส่วนฟิลด์ SFD จะมีขนาด 1 ไบต์อยู่ต่อท้ายฟิลด์ Preamble และมีค่าเป็น 1 และ 0 สลับกันเช่นเดียวกับฟิลด์ Preamble แต่จะลงท้ายด้วย 1 สองตัวติดกัน (10101011) ซึ่งจะช่วยให้รู้จุดเริ่มต้นของเฟรมได้
2. **Destination Address** มีขนาด 6 ไบต์บรรจุฮาร์ดแวร์แอดเดรสหรือโหนดแอดเดรสของสเตชั่นบนเซ็กเมนท์เดียวกันซึ่งแพ็คเกจถูกกำหนดให้ส่งไป ถ้าแอดเดรสมีค่าเป็น 0xFF-FF-FF-FF-FF-FF จะหมายถึงบรอดคาสท์แอดเดรส
3. **Source Address** มีขนาด 6 ไบต์ใช้บรรจุโหนดแอดเดรสของสเตชั่นบนเซ็กเมนท์เดียวกันที่เป็นผู้ส่งแพ็คเกจ
4. **Length** มีขนาด 2 ไบต์ บอกถึงความยาวของข้อมูลจากเลเยอร์ที่สูงกว่าที่บรรจุอยู่ในฟิลด์ข้อมูลของเฟรม ซึ่งค่าจะต้องเป็น 1500 หรือน้อยกว่าสำหรับเฟรมอีเธอร์เน็ต 802.3 ที่ถูกต้อง
5. ฟิลด์ **Data** จะเป็นจุดที่เฮดเดอร์ของไอพีเอ็กซ์เริ่มต้น ความยาวของฟิลด์นี้จะอยู่ระหว่าง 46-1518 ไบต์ เพราะว่าโพรโตคอลไอพีเอ็กซ์/เอสพีเอ็กซ์ (IPX/SPX) ของโนเวล(Novell) เท่านั้นที่ใช้เฟรมอีเธอร์เน็ต 802.3 ดัง

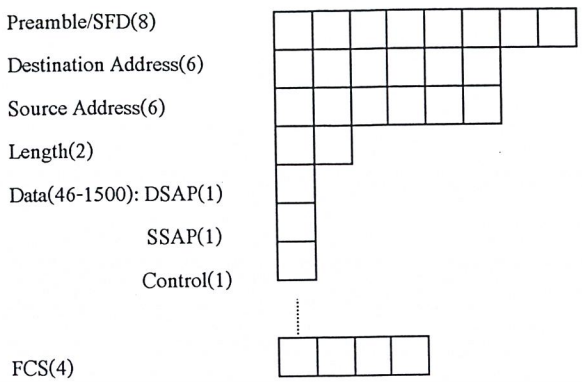
นั้นฟิลด์ข้อมูลจะเริ่มด้วยเฮดเดอร์ของไอพีเอ็กซ์ซึ่งเฮดเดอร์ไอพีเอ็กซ์จะเริ่มต้นด้วยค่า 0xFF-FF

6. **Padding** เพื่อที่จะให้เฟรมมีขนาดอย่างน้อย 64 ไบต์ เพราะฟิลด์ข้อมูลจะต้องมีขนาดอย่างน้อย 46 ไบต์ซึ่งเมื่อรวมกับเฮดเดอร์อีก 18 ไบต์โดยไม่นับฟิลด์ Preamble และ SFD แล้วจะได้ 64 ไบต์ แต่ถ้าข้อมูลที่จะส่งมีไม่ถึง 46 ไบต์ก็จะมีการเติมให้ครบ(padded) อย่างไรก็ตามอาจพบว่ามีการแพด(pad) 1 ไบต์ถึงแม้ว่าตัวข้อมูลจะมีขนาดมากกว่า 46 ไบต์แล้วก็ตาม ทั้งนี้เพราะโนเวลร้องขอให้ผู้ผลิตไดรเวอร์แลนทุกแห่งพัฒนาไดรเวอร์แลนที่สร้างแพ็คเกจที่มีขนาดเป็นจำนวนคู่(evenize) ซึ่งก็คือการเพิ่ม 1 ไบต์เข้าไป เหตุที่แพ็คเกจต้องเป็นเลขคู่ก็เพราะว่าเราท์เตอร์บางตัวสามารถจัดการและประมวลผลได้เฉพาะแพ็คเกจที่มีความยาวเป็นจำนวนคู่เท่านั้น

7. **Frame Check Sequence (FCS)** การตรวจสอบข้อผิดพลาด(error checking) ถูกสร้างให้กับแต่ละเฟรมอีเธอร์เน็ตเพื่อให้มั่นใจได้ว่าเฟรมที่ถูกต้องเท่านั้นที่จะถูกประมวลผลโดยสเตชั่นที่ได้รับแพ็คเกจ ฟิลด์ FCS มีขนาด 4 ไบต์ บรรจุค่าซีอาร์ซี(CRC) โดยสเตชั่นที่เป็นผู้ส่งแพ็คเกจจะทำการคำนวณค่าซีอาร์ซีจากทุกฟิลด์ยกเว้นฟิลด์ Preamble และ SFD ก่อนส่งและใส่ค่านี้ลงในฟิลด์ FCS เมื่อสเตชั่นปลายทางได้รับข้อมูลก็จะทำการคำนวณค่าซีอาร์ซีนี้อีกครั้งเช่นเดียวกันแล้วจะนำค่าที่คำนวณได้นี้มาเปรียบเทียบกับค่าในฟิลด์ FCS ซึ่งถ้าสอดคล้องกัน(เหมือนกัน) ก็แสดงว่าเฟรมที่รับมานั้นถูกต้อง

2.3.1.2 อีเธอร์เน็ต 802.2

ถูกพิจารณาว่าเป็นสิ่งที่สอดคล้องกับไอทริปเปิ้ลอี(IEEE-compliant) เพราะบรรจุทั้งฟิลด์ 802.3 และ 802.2 ฟิลด์ 802.2 ยังถูกเรียกว่าเลเยอร์แอลแอลซี(Logical Link Control: LLC) ภายในเฟรม อีเธอร์เน็ต 802.2 มีโครงสร้างดังรูป 2-27 ซึ่งเริ่มต้นด้วยเฮดเดอร์ 802.3 ส่วนฟิลด์ 802.2 นั้นจะเริ่มต่อจากฟิลด์ Length ของเฮดเดอร์ 802.3 ซึ่งรายละเอียดของฟิลด์ 802.2 เป็นดังนี้



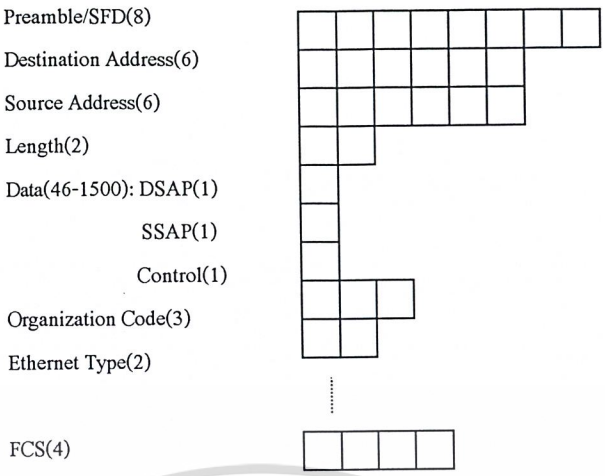
รูปที่ 2-27 โครงสร้างเฟรม 802.2

1. **Destination Service Access Point(DSAP)** มีขนาด 1 ไบต์ บอกถึงชนิดของโพรโตคอลในเลเยอร์ที่สูงกว่า(เลเยอร์เน็ตเวิร์ค) ณ ปลายทางของแพ็คเกจ ถ้าเป็นแพ็คเกจที่ใช้โพรโตคอลไอพีเอ็กซ์/เอสพีเอ็กซ์ ฟิลด์นี้จะมีค่าเป็น 0xE0
2. **Source Service Access Point(SSAP)** มีขนาด 1 ไบต์ บอกถึงชนิดโพรโตคอลในเลเยอร์ที่สูงกว่า (หรือเลเยอร์เน็ตเวิร์ค) เช่นเดียวกับฟิลด์ DSAP ถ้าเป็นแพ็คเกจที่ใช้โพรโตคอลไอพีเอ็กซ์/เอสพีเอ็กซ์ ฟิลด์นี้จะมีค่าเป็น 0xE0
3. **Control** มีขนาด 1 ไบต์ ถ้าเฟรมนั้นใช้โพรโตคอลไอพีเอ็กซ์/เอสพีเอ็กซ์ฟิลด์นี้จะมีค่าเป็น 0x03 ซึ่งหมายถึงเป็นเฟรม 802.2 แบบอันนัมเบอร์ (802.2 unnumbered format) หรือเลเยอร์แอลแอลซีให้บริการแบบไม่ต้องก่อตั้งการเชื่อมต่อก่อน(connectionless) นั่นเอง

ขนาดเฟรมน้อยที่สุดและมากที่สุดยังคงเป็น 64-1518 ไบต์ซึ่งไม่นับฟิลด์ Preamble และ SFD เช่นเดิม

2.3.1.3 อีเธอร์เน็ตสแนบ

สแนบ(SNAP) ย่อมาจาก Sub-Network Access Protocol ซึ่งเฟรมชนิดนี้พัฒนามาจากเฟรม 802.2 ซึ่งแสดงได้ดังรูปที่ 2-28 ซึ่งฟิลด์ต่าง ๆ จะเหมือนกับเฟรม 802.2 แต่มีส่วนที่แตกต่างกันดังนี้



รูปที่ 2-28 โครงสร้างเฟรมอีเธอร์เน็ตสแนป

1. **DSAP, SSAP และ Control** ในอีเธอร์เน็ตสแนปค่าของฟิลด์ DSAP และ SSAP จะเป็น 0xAA เสมอเพื่อบอกว่าเป็นเฟรมแบบอีเธอร์เน็ตสแนป ส่วนฟิลด์ Control นั้นจะมีขนาด 1 ไบต์และมีค่า 0x03(unnumbered) เสมอและจะตามด้วยฟิลด์ Organization code และ Ethernet type

2. **Organization code** มีขนาด 3 ไบต์บอกถึงว่าองค์การใดที่ใช้ฟิลด์ Ethernet type นั้น ๆ ซึ่งโพรโตคอลไอพีเอ็กซ์/เอสพีเอ็กซ์ของเน็ตแวร์(NetWare) จะมีค่าในฟิลด์นี้เป็น 0x00-00-00

3. **Ethernet type** มีขนาด 2 ไบต์ ใช้เพื่อกำหนดโพรโตคอลของเลเยอร์ที่สูงกว่า ซึ่งฟิลด์ Ethernet type ของเน็ตแวร์จะมีค่าเป็น 0x8137(และโนเวลยังได้จองหมายเลข 0x8138 ไว้ด้วยถึงแม้ว่าจะยังไม่ใช้ในขณะนี้) และตามด้วยค่า 0xFF-FF ในฟิลด์ข้อมูลเช่นเดิม ตัวอย่างของค่าในฟิลด์นี้ของโพรโตคอลหลายชนิดเป็นดังนี้

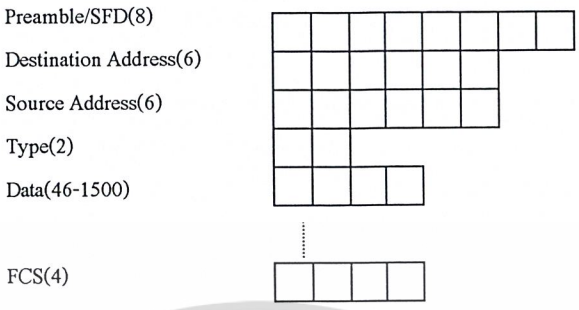
IP(Internet Protocol)	0x0800
ARP	0x0806
RARP	0x8035
AppleTalk	0x809B
AppleTalk ARP	0x80F3
NetWare IPX/SPX	0x8137

ขนาดเฟรมน้อยที่สุดและมากที่สุดยังคงเป็น 64-1518 ไบต์ซึ่งไม่นับฟิลด์ Preamble และ SFD เช่นเดิม

2.3.1.4 อีเธอร์เน็ตทู

เฟรมอีเธอร์เน็ตทูแตกต่างจากเฟรมอื่น ๆ ตรงฟิลด์ Type ซึ่งตามหลังแอดเดรสต้น

ทาง ในขณะที่อีเธอร์เน็ต 802.3, อีเธอร์เน็ต 802.2 และอีเธอร์เน็ตสแนปที่มีฟิลด์ Length ตามหลังแอดเดรสต้นทาง โครงสร้างเฟรมอีเธอร์เน็ตทูแสดงได้ดังรูปที่ 2-29



รูปที่ 2-29 โครงสร้างเฟรมอีเธอร์เน็ตทู

โครงสร้างเฟรมของอีเธอร์เน็ตทูจะมี 2 แห่งที่แตกต่างจากโครงสร้างเฟรมแบบอื่น ๆ คือเฟรม Preamble/SFD และฟิลด์ Type

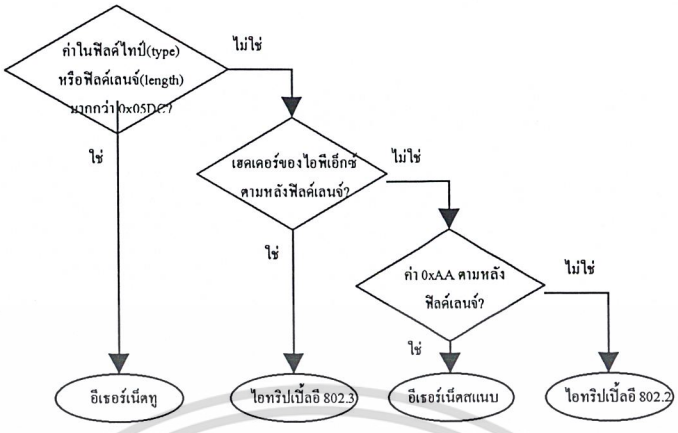
1. **Preamble** มีขนาด 8 ไบต์บรรจุเลข 1 และ 0 สลับกัน เช่นเดียวกับฟิลด์นี้ของเฟรมแบบอื่น ๆ ที่มีขนาด 7 ไบต์ แต่ในเฟรมอีเธอร์เน็ตทูฟิลด์ SFD ที่มีขนาด 1 ไบต์(10101011) จะถูกพิจารณาว่าเป็นส่วนหนึ่งของฟิลด์ preamble ด้วย

2. **Type** ไม่เหมือนเฟรมอื่น ๆ อีเธอร์เน็ตทูบรรจุฟิลด์นี้แทนฟิลด์ Length ซึ่งฟิลด์นี้จะระบุโพรโตคอลในชั้นที่สูงกว่าซึ่งใช้แพ็คเกจนี้ ตัวอย่างของค่าโพรโตคอลที่บรรจุในฟิลด์ Type ได้ซึ่งจะคล้ายกับค่าที่ใช้ในฟิลด์ Type ของอีเธอร์เน็ตสแนปคือ

IP(Internet Protocol)	0x0800
ARP	0x0806
RARP	0x8035
AppleTalk	0x809B
AppleTalk ARP	0x80F3
NetWare IPX/SPX	0x8137

ขนาดเฟรมน้อยที่สุดและมากที่สุดยังคงเป็น 64-1518 ไบต์ซึ่งไม่นับฟิลด์ Preamble และ SFD เช่นเดิม

2.3.2 ความแตกต่างระหว่างเฟรมอีเธอร์เน็ตแบบต่าง ๆ



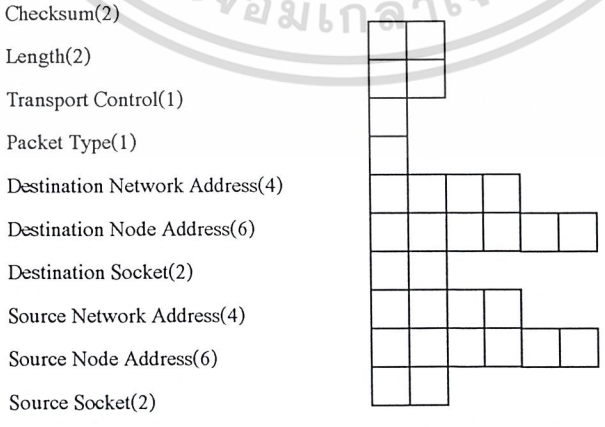
รูปที่ 2-30 โฟลวชาร์ตที่แสดงแบบของเฟรมต่าง ๆ

2.3.3 โพรโตคอลไอพีเอ็กซ์/เอสพีเอ็กซ์ของเน็ตแวร์

เมื่อไคลเอ็นต์และเซิร์ฟเวอร์ของเน็ตแวร์จะติดต่อสื่อสารกัน จะใช้บริการที่ต้องก่อตั้งการเชื่อมต่อหรือไม่ต้องก่อตั้งการเชื่อมต่ออย่างใดอย่างหนึ่ง ในการใช้บริการแบบต้องก่อตั้งการเชื่อมต่อก่อนนั้นจะใช้โพรโตคอลเอสพีเอ็กซ์ แต่ถ้าจะใช้บริการแบบไม่ต้องการการก่อตั้งการเชื่อมต่อก่อนก็จะใช้โพรโตคอลไอพีเอ็กซ์ ซึ่งโพรโตคอลทั้งสองมีข้อแตกต่างกันคือ โพรโตคอลไอพีเอ็กซ์จะดีในแง่ที่จำเป็นต้องใช้เฮดเดอร์เป็นจำนวนที่น้อยกว่าแต่จะไม่รับรองว่าข้อมูลที่ส่งไปจะไปถึงหรือไม่และเป็นลำดับอย่างไร ในขณะที่ไอพีเอ็กซ์จะมีลักษณะตรงกันข้าม

2.3.3.1 เฮดเดอร์ไอพีเอ็กซ์

จะมีความยาว 30 ไบต์และจะเริ่มต้นด้วยค่า 0xFFFF ซึ่งถูกส่งหลังจากเฟรมในเลเยอร์ย่อยแมค (MAC) ของเลเยอร์คาต้าลิงค์แต่จะอยู่ก่อนส่วนที่เป็นข้อมูล



รูปที่ 2-31 แสดงรูปแบบของเฮดเดอร์ไอพีเอ็กซ์

1. **Checksum** โดยปกติแล้วฟิลด์เช็คซัมในเฮดเดอร์ไอพีเอ็กซ์จะไม่ถูกใช้ และจะใส่ค่า 0xFFFF ลงไปในฟิลด์นี้ซึ่งมีขนาด 2 ไบต์แทน โนเวลตั้งใจจะไม่ใช้ฟิลด์นี้เพราะการตรวจซีอาร์ซีก็ครอบคลุมทั้งเฟรมอยู่แล้ว แต่เราก็ยังสามารถใช้งานนี้ได้ถ้าต้องการ แต่จะใช้ไม่ได้เมื่อเฟรมเป็นอีเธอร์เน็ตแบบ 802.3 เพราะฟิลด์นี้เท่านั้นที่ทำหน้าที่ชี้ให้รู้ว่าเป็นเฟรมของเน็ตแวร์ ในขณะที่เฟรมอีเธอร์เน็ตอีก 3 แบบมีฟิลด์ที่ระบุโพรโตคอลในเลเยอร์ที่สูงกว่าอยู่แล้ว เช็คซัมในไอพีเอ็กซ์เฮดเดอร์นี้จะตรวจสอบเฉพาะเฮดเดอร์เท่านั้นไม่รวมไปถึงข้อมูล

2. **Length** มีขนาด 2 ไบต์ บรรจุความยาวของแพ็คเกจซึ่งรวมไอพีเอ็กซ์เฮดเดอร์และส่วนข้อมูลที่ตามมาเท่านั้น ไม่ได้รวมฟิลด์ในส่วนเฟรมอีเธอร์เน็ต

3. **Transport Control** มีขนาด 1 ไบต์ ฟิลด์นี้ถูกใช้โดยเราท์เตอร์ไอพีเอ็กซ์และแสดงถึงจำนวนของเราท์เตอร์ที่แพ็คเกจไอพีเอ็กซ์ได้ผ่านมา สเตชั่นที่เป็นต้นกำเนิดจะเซ็ตฟิลด์นี้ให้มีค่าเป็น 0 แพ็คเกจจะผ่านไปได้ถึง 15 เราท์เตอร์และจะถูกละทิ้งโดยเราท์เตอร์ตัวที่ 16

4. **Packet Type** มีขนาด 1 ไบต์ ฟิลด์นี้แสดงให้ทราบว่าบริการประเภทใดที่แพ็คเกจนี้ใช้อยู่ สำหรับการสื่อสารที่ใช้ไอพีเอ็กซ์ ค่าจะเป็น 0, 4, 5, หรือ 17

IPX-based communications	0 or 4
SPX-based communications	5
NCP communications	17

5. **Destination Network** ฟิลด์นี้บรรจุแอดเดรสของระบบเครือข่ายจำนวน 4 ไบต์ซึ่งโหนดปลายทางตั้งอยู่ ถ้าฟิลด์นี้มีค่าเป็น 0x00-00-00-00 จะหมายถึงว่าปลายทางของแพ็คเกจนี้อยู่ในระบบเครือข่ายเดียวกันกับสเตชั่นต้นทาง

6. **Destination Node** เป็นฟิลด์ที่มีขนาด 6 ไบต์ใช้บรรจุแอดเดรสโหนดของสเตชั่นปลายทางหรือถ้ามีค่าเป็น 0xFF-FF-FF-FF-FF-FF ก็หมายถึงเป็นการบรอดคาสท์

7. **Destination Socket** มีขนาด 2 ไบต์บรรจุหมายเลขซ็อคเก็ตของโพรเซสภายในโหนดที่แพ็คเกจถูกส่งไป หมายเลขซ็อคเก็ตที่รู้จักกันดีคือ

0x451	NetWare Core Protocol
0x452	Service Advertising Protocol packet
0x453	Routing Information Protocol packet
0x455	NetBIOS packet
0x456	Diagnostic packet
0x457	Serialization packet

ที่เวิร์คสเตชั่นหมายเลขซ็อคเก็ตจะถูกกำหนดแบบไดนามิค ซึ่งจะอยู่ในช่วง 0x4000-0x8000

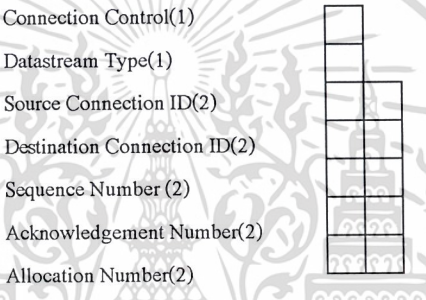
8. **Source Network** มีขนาด 4 ไบต์บรรจุแอดเดรสของระบบเครือข่ายที่โหนดต้นทางตั้งอยู่

9. **Source Node** มีขนาด 6 ไบต์ บรรจุแอดเดรสของสเตชั่นที่ส่งข้อมูล ไม่สามารถบรรจุแอดเดรสสำหรับการบรอดคาสท์ในฟิลด์นี้ได้

10. **Source Socket** บรรจุหมายเลขซ็อคเก็ตของโพรเซสที่ส่งแพ็คเกจนี้ ฟิลด์นี้มีขนาด 2 ไบต์

2.3.3.2 เฮดเดอร์เอสพีเอ็กซ์(Sequence Packet Exchange)

มีความยาว 42 ไบต์และจะเริ่มต้นด้วยฟิลด์ของเฮดเดอร์ไอพีเอ็กซ์ ประกอบด้วยฟิลด์ดังรูปที่ 2-32



รูปที่ 2-32 แสดงรูปแบบเฮดเดอร์ของเอสพีเอ็กซ์

1. **Connection Control** มี 1 ไบต์ ช่วยควบคุมการไหลของข้อมูลแบบสองทิศทาง ซึ่งตัวอย่างค่าของฟิลด์นี้คือ

VALUE	NAME	DESCRIPTION
0x10	End-of-Message	บ่งบอกถึงไคลเอ็นท์ต้องการหยุดการติดต่อ
0x20	Attention	ฟิลด์นี้ยังไม่มีการนำมาใช้
0x40	Acknowledgement Required	ข้อมูลได้ถูกส่งไปแล้ว และต้องการการรับรองว่าข้อมูลได้ส่งไปถึงแล้วจริง ๆ
0x80	System Packet	เป็นแพ็คเก็จแอคโนวเลดเมนท์ ซึ่งไอพีเอ็กซ์ใช้ฟิลด์นี้เป็นการภายใน เพราะมันไม่ได้ถูกส่งไปสู่เซลล์ของสเตชั่นปลายทาง

2. **Datastream Type** ฟิลด์นี้มีขนาด 1 ไบต์ ใช้บ่งถึงชนิดของข้อมูลที่บรรจุอยู่ภายในแพ็คเก็จ ฟิลด์นี้ยังสามารถเก็บค่าที่ไคลเอ็นท์เป็นคนกำหนดหรือค่าดังต่อ

ไปนี้

VALUE	NAME	DESCRIPTION
0xFE	End-of-connection	สร้างขึ้นมาเพื่อให้ทราบได้ว่าไคลเอ็นท์ต้องการจะหยุดการสื่อสาร
0xFF	End-of-Connection Acknowledgement	ถูกส่งเมื่อได้รับคำร้องขอในการหยุดการเชื่อมต่อ

3. **Source Connection ID** เก็บเลขจำนวน 2 ไบต์ที่ถูกกำหนดโดยสเตชั่นต้นทางที่ใช้เอสพีเอ็กซ์ ซึ่งใช้สำหรับการดีมัลติเพล็กซ์(demultiplexing) การสื่อสารของเอสพีเอ็กซ์ เนื่องจากการเชื่อมต่อบนเครื่องสามารถใช้หมายเลขซ็อคเก็ตเดียวกันได้ ฟิลด์นี้จึงมีความสำคัญในการแยกแยะแต่ละการเชื่อมต่อเสมือน และหมายเลขที่บรรจุในฟิลด์นี้ก็ยังสามารถใช้กับฟิลด์ Destination Connection ID ได้โดยเอสพีเอ็กซ์อีกฝั่งเพื่อตอบสนอง

4. **Destination Connection ID** มีขนาด 2 ไบต์ ใช้บรรจุหมายเลขการเชื่อมต่อของสเตชั่นปลายทางในระหว่างการก่อตั้งการเชื่อมต่อ ฟิลด์นี้จะถูกเซ็ตค่าให้เป็น 0xFFFF เพราะว่าผู้ส่งยังไม่รู้ว่าหมายเลขการเชื่อมต่อปลายทางที่ผู้รับใช้เป็นเท่าใด

5. **Sequence Number** เป็นฟิลด์ที่มีขนาด 2 ไบต์ซึ่งบรรจุตัวเลขที่นับจำนวนข้อมูลที่ถูกส่งจากสเตชั่น ตัวเลขนี้จะถูกเพิ่มค่าเมื่อได้รับแอคโนวเลดเมนท์แล้วเท่านั้น สเตชั่นจะไม่เพิ่มค่าตัวเลขดังกล่าวนี้เมื่อทำการส่งแพ็คเกจที่ใช้ทำแอคโนวเลดเมนท์

6. **Acknowledge Number** ระหว่างการส่งข้อมูล แพ็คเกจอาจเกิดการสูญหายได้ ฟิลด์แอคโนวเลดเมนท์จะบรรจุค่าของหมายเลขลำดับ(sequence number) ถัดไปที่คิดว่าจะได้รับ ถ้าหมายเลขลำดับไม่ถูกต้อง สเตชั่นที่เป็นผู้รับจะถือว่ามีข้อผิดพลาดเกิดขึ้นในการสื่อสาร ฟิลด์นี้มีขนาด 2 ไบต์

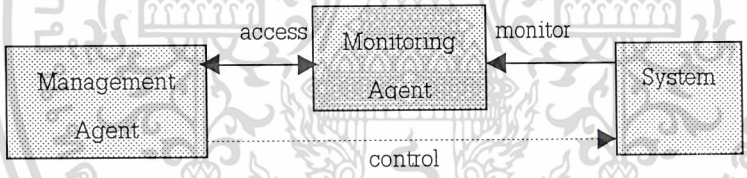
7. **Allocation Number** มีขนาด 2 ไบต์ บ่งถึงจำนวนของบัฟเฟอร์ของการรับที่ว่างในสเตชั่น ซึ่งจะมีค่าเริ่มที่ 0 สำหรับบัฟเฟอร์แรกที่ว่าง ดังนั้นถ้ามีค่าเป็น 6 ก็หมายความว่าบัฟเฟอร์จะสามารถรับแพ็คเกจได้ 7 แพ็คเกจ เมื่อมีการใช้ข้อมูลที่ได้รับมาโดยแอพพลิเคชั่น บัฟเฟอร์ก็จะว่างเพิ่มขึ้น ถ้าสเตชั่นกำลังยุ่งและไม่สามารถเคลียร์แพ็คเกจจากบัฟเฟอร์ได้ จำนวนบัฟเฟอร์ที่ว่างก็จะลดลงทุกครั้งที่สเตชั่นได้รับแพ็คเกจ

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) ทราบสถานะของเครือข่ายและทราบถึงแนวโน้มในระยะยาวของระบบเครือข่ายคอมพิวเตอร์ที่ซับซ้อนได้ ซึ่งตัวอย่างทั่ว ๆ ไปคือระบบการควบคุมแบบป้อนกลับ (feedback system) ดังรูปที่ 3-1 ข้อมูลเกี่ยวกับระบบ(system) จะถูกมอนิเตอร์โดยมอนิเตอริ่งเอเจนท์ (monitoring agent) ซึ่งอาจมีได้มากกว่าหนึ่งตัว โดยมีแมนเนจเมนท์เอเจนท์(management agent)



รูปที่ 3-1 การมอนิเตอร์ในระบบป้อนกลับ

นำข้อมูลที่มอนิเตอริ่งเอเจนท์หามาไปใช้ในการวิเคราะห์และส่งการควบคุมกลับไปยังระบบที่ถูกมอนิเตอร์

ในการมอนิเตอร์ระบบเครือข่ายนั้นจะต้องมีการเกี่ยวข้องกับกิจกรรม 3 อย่างต่อไปนี้

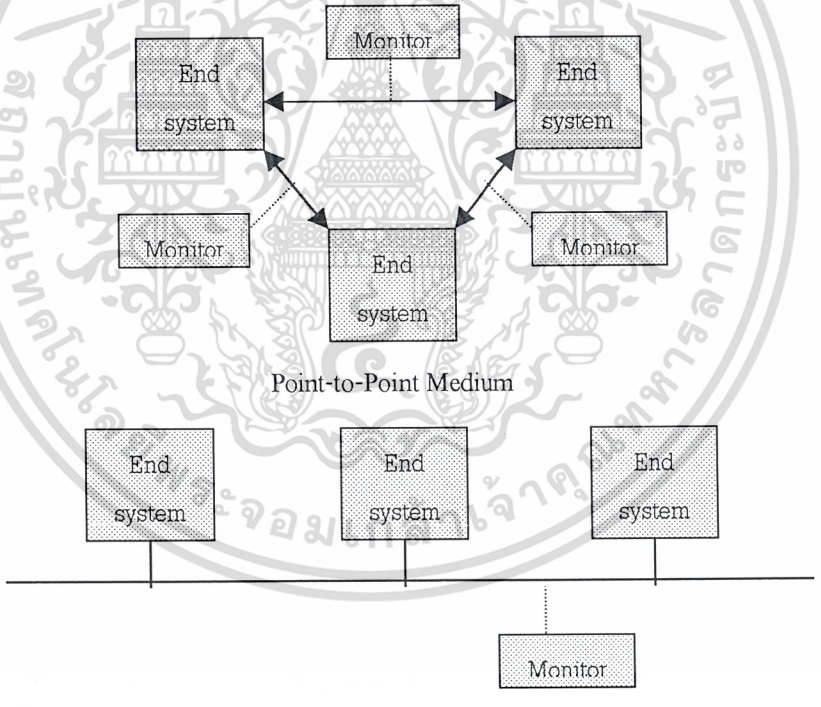
3.1 การเข้าถึงข้อมูลที่จะมอนิเตอร์

จะเกี่ยวข้องกับการพิจารณาว่าจะกำหนดความหมายและรูปแบบของข้อมูลอย่างไรจึงจะทำให้แมนเนจเมนท์เอเจนท์หลายตัวสามารถเข้าถึงและเข้าใจได้(ลูกศรแอคเซส(access) ในรูปที่ 3-1) ซึ่งถ้าต้องการให้สิ่งที่ถูกจัดการซึ่งถูกผลิตจากหลายผู้ผลิตสามารถมอนิเตอร์และจัดการได้เหมือนกันก็จะต้องมีการกำหนดมาตรฐานออกมาใช้

3.2 การออกแบบกลไกในการมอนิเตอร์

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

(component) ของเครือข่าย ซึ่งนำไปสู่การนำไปไว้ที่ระบบปลาย(end systems) และระบบกลาง (intermediate systems) สถานะที่แน่นอนจะสามารถมอนิเตอร์ได้เมื่อมอนิเตอริ่งเอเจนท์เป็นส่วนหนึ่งของสิ่งที่เรากำลังมอนิเตอร์อยู่เท่านั้นซึ่งเอเจนท์ที่มีลักษณะแบบนี้เรียกว่า อินทิเกรตเต็ดมอนิเตอริ่งเอเจนท์ (integrated monitoring agent) ซึ่งมีข้อดีที่เราสามารถจะมอนิเตอร์สถานะของสิ่งที่เราสนใจได้อย่างแม่นยำทันทีทันใด และลึกเท่าที่ต้องการได้ กลไกในการรวบรวมข้อมูลเพื่อใช้ในการจัดการเครือข่ายจะต้องไม่กระทบกับจุดประสงค์หลักของเครือข่ายคือการสื่อสารข้อมูลของแอปพลิเคชั่น(application) ดังนั้นถ้าเครือข่ายมีขนาดใหญ่ประกอบด้วยหลายส่วนประกอบ การใช้ความสามารถในการมอนิเตอร์กับทุกส่วนประกอบจะเป็นการสร้างปัญหาคือทำให้แบนด์วิธ(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 ซึ่งบ่อยครั้งที่ไม่สามารถจะแยกแยะปัญหาได้เร็วนัก เนื่องมาจากความซับซ้อนของปัญหา 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)

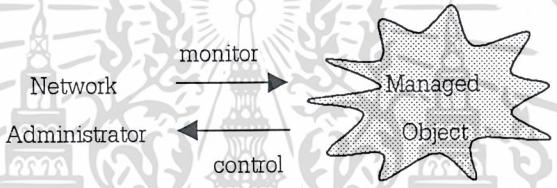
กลไกในการรักษาความปลอดภัยมักจะมีผลกระทบทางลบกับสมรรถภาพและค่าใช้จ่ายของเครือข่าย การตรวจสอบและดักจับการกระทำที่ก่อให้เกิดความไม่ปลอดภัยในระบบ(security violations) จะอยู่ในรูปแบบของการมอนิเตอร์เครือข่ายซึ่งต้องการฟังก์ชันในการวิเคราะห์ความปลอดภัยและตัวกรองรวมเข้าไปกับฟังก์ชันมอนิเตอร์เพื่อให้ตรวจจับปัญหาความปลอดภัยได้อย่างแม่นยำ

5. การบันทึกการใช้งาน(Accounting management)

ทำหน้าที่บันทึกอัตราการใช้ทรัพยากรเครือข่ายเพื่อควบคุมค่าใช้จ่ายที่เกิดจากการใช้งานเครือข่าย ซึ่งการทำเช่นนี้จะทำให้เกิดโอเวอร์เฮด(overhead) ขึ้นในเครือข่ายนั้นมาก การบันทึกการใช้งานนี้จึงใช้กับเครือข่ายที่สร้างขึ้นมาเพื่อผลประโยชน์ทางธุรกิจเท่านั้น เช่นการเก็บค่าใช้บริการจากผู้ใช้ เป็นต้น

3.4 การเข้าถึงข้อมูลที่มอนิเตอร์(Access to monitor information)

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



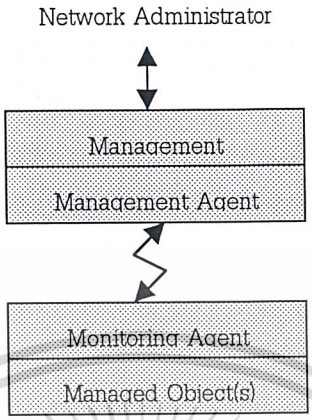
รูปที่ 3-3 แบบระดับสูงของอ็อบเจ็ค

แบบทางฟังก์ชันสำหรับการมอนิเตอร์อ็อบเจ็คที่ถูกจัดการจะประกอบไปด้วย

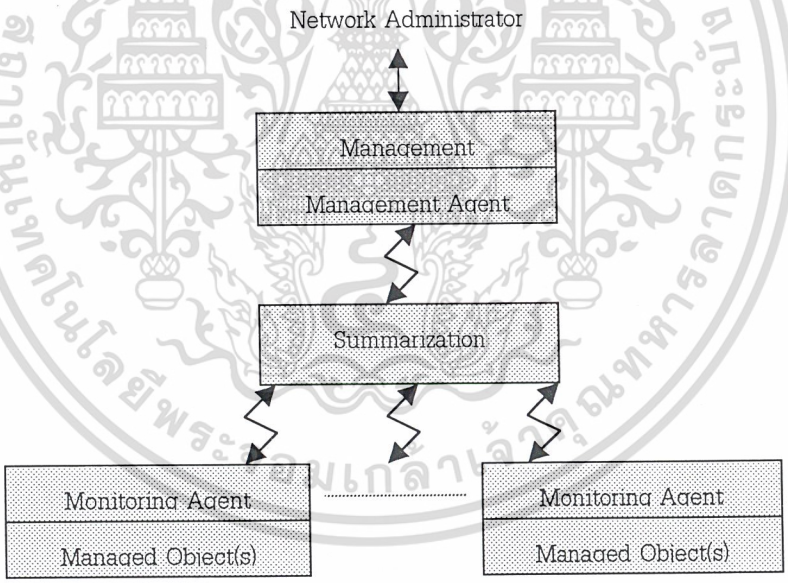
1. แอปพลิเคชั่นที่ใช้ในการจัดการ(The management application) คือโมดูล(module) ของซอฟต์แวร์(software) ที่ทำหน้าที่ช่วยผู้จัดการเครือข่ายในการมอนิเตอร์เครือข่ายและทำงานเกี่ยวกับการจัดการ

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) เพื่อเป็นตัวกลางระหว่างมอนิเตอริ่งเอเจนท์กับแมนเนจเมนท์เอเจนท์เดิมโดยเข้าถึงอ็อบเจ็คผ่านมอนิเตอริ่งเอเจนท์ซึ่งกรณีนี้มอนิเตอริ่งเอเจนท์จะมองมันว่าเป็นแมนเนจเมนท์เอเจนท์ แต่สำหรับแมนเนจเมนท์เอเจนท์แล้วมันจะถูกมองว่าเป็นมอนิเตอริ่งเอเจนท์ดังรูปที่ 3-5



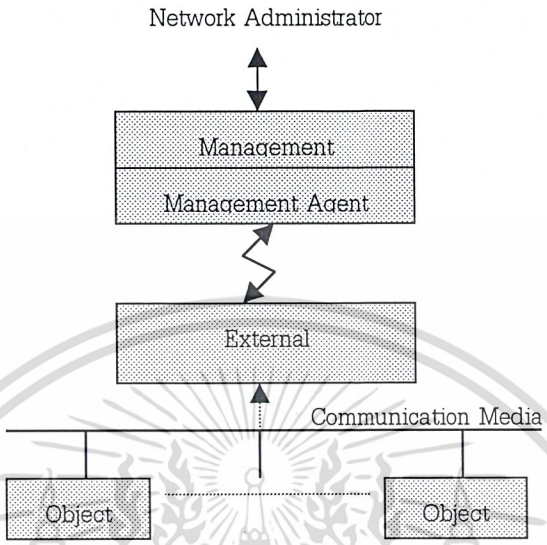
รูปที่ 3-4 แบบของฟังก์ชันพื้นฐานสำหรับการมอนิเตอร์อ็อบเจ็ค



รูปที่ 3-5 ซัมมาไรเซชั่นมอนิเตอริ่งเอเจนท์

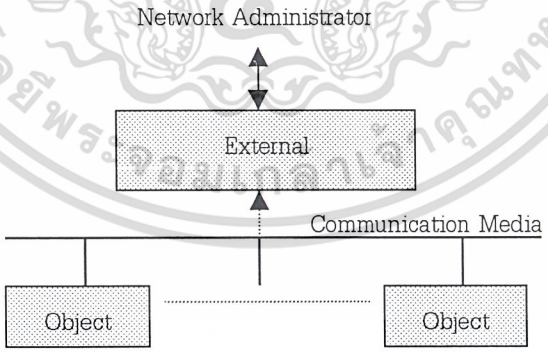
มอนิเตอริ่งเอเจนท์ไม่จำเป็นจะต้องได้มาซึ่งข้อมูลโดยการใช้แอคเซสโพรโตคอล(access protocol) ที่เฉพาะเจาะจงเท่านั้น ในความเป็นจริงมันอาจจะใช้วิธีวิเคราะห์ทราฟฟิค(traffic analysis) ซึ่งถ้าใช้วิธีนี้มันจะถูกเรียกว่า เอ็กเทอร์นอลมอนิเตอริ่งเอเจนท์(external monitoring agent) ดังรูปที่ 3-6 เอเจนท์แบบนี้จะไม่ได้อยู่รวมกับอ็อบเจ็ค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

การสื่อสารกันระหว่างอ็อบเจ็คที่ใช้ตัวกลางในการสื่อสารร่วมกัน และเนื่องจากเป็นการวิเคราะห์ทราฟฟิคเอ็กเทอร์นอลมอนิเตอร์จึงทำให้ผู้จัดการเครือข่ายได้รับข้อมูลที่เป็นประโยชน์เกี่ยวกับการจัดการเครือข่าย



รูปที่ 3-6 เอ็กเทอร์นอลมอนิเตอริ่งเอเจนท์

น้อยลง ถ้าเอ็กเทอร์นอลมอนิเตอริ่งเอเจนท์อยู่รวมกับแมนเนจเมนท์แอปพลิเคชั่นและแมนเนจเมนท์เอเจนท์ดังรูปที่ 3-7 จะเรียกว่าเอ็กเทอร์นอลมอนิเตอร์ ซึ่งทำให้ไม่ต้องการโพรโตคอลที่ใช้ในการจัดการทำให้เอ็กเทอร์นอลมอนิเตอร์นี้สร้างขึ้นมาได้ง่าย



รูปที่ 3-7 เอ็กเทอร์นอลมอนิเตอร์

3.5 การออกแบบตัวมอนิเตอร์เครือข่าย(The design of network monitors)

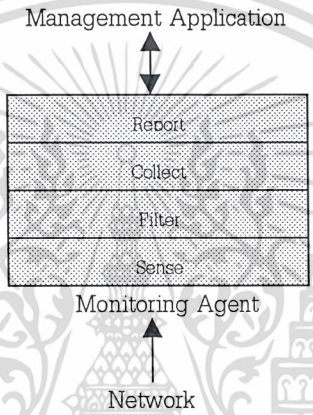
แอปพลิเคชั่นและการนำเอเจนท์ของการมอนิเตอร์เครือข่ายมาใช้อาจจะแตกต่างกัน แต่จะมีฟังก์ชันหลัก ๆ ที่เหมือนกันดังนี้

1. การจับสัญญาณ(Sensing) เป็นการสร้างการติดต่อโดยตรงกับสภาพแวดล้อมทางการสื่อสาร และจับเอาข้อมูลขึ้นมา

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

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

4. การรายงานสรุปผล(Reporting) เป็นการนำข้อมูลที่เก็บไว้มาใช้โดยแมนเนจเมนท์แอปพลิเคชั่นเมื่อเราต้องการ



รูปที่ 3-8 เอเจนท์ในการมอนิเตอร์

3.6 ข้อแตกต่างระหว่างตัวมอนิเตอร์แบบอินทิเกรตเต็ดและเอ็กเทอร์นอล(Integrated VS. External monitors)

เราสามารถแบ่งลักษณะของตัวมอนิเตอร์เครือข่าย(network monitoring) ได้ 2 ประเภทใหญ่ ๆ ซึ่งจะมีผลในการออกแบบและการนำไปใช้งานจริงดังนี้

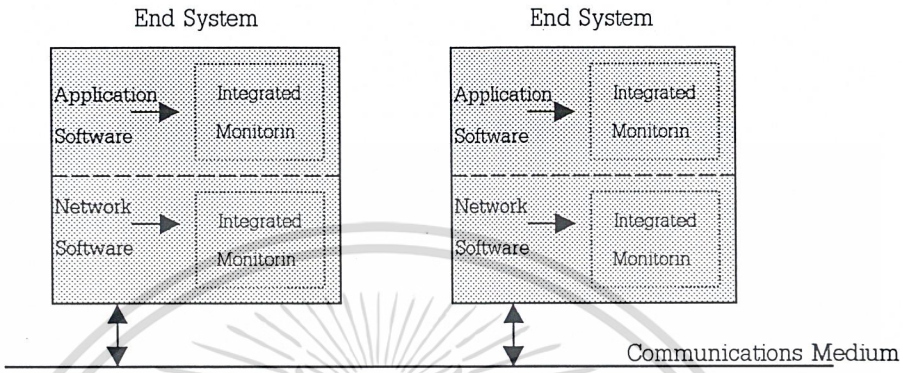
1. ใช้เลเยอร์ในแบบอ้างอิงของโอเอสไอ(OSI-RM) ที่เอเจนท์ทำการมอนิเตอร์เครือข่ายเป็นสิ่งที่ใช้แบ่งประเภทของตัวมอนิเตอร์เครือข่าย

2. ใช้ตำแหน่งที่เอเจนท์อยู่ว่าถูกรวมหรืออยู่ภายนอกส่วนประกอบเครือข่าย, ทรัพยากรที่ถูกมอนิเตอร์

2.1 เอเจนท์แบบอินทิเกรตเต็ด(Integrated monitoring agents)

เอเจนท์แบบนี้สามารถจะเป็นส่วนหนึ่งของแอปพลิเคชั่นเฉพาะในเครือข่าย(particular network application) หรือซอฟต์แวร์เครือข่าย(network software) ได้ ซึ่งถ้าเป็นส่วนหนึ่งในซอฟต์แวร์เครือข่าย จะต้องมีบริการของเครือข่าย(network service) บางอย่างรวมอยู่ด้วย เช่น การเรียกโมดูลย่อยทางไกล(Remote Procedure Call: RPC) หรือบริการทางการสื่อสาร(communications service) ที่เป็นซอฟต์แวร์ด้านเครือข่ายในระบบปฏิบัติการ เช่น เน็ตเวิร์คดีไวซ์ไดรเวอร์(network device drivers) หรือ

ส่วนหนึ่งของส่วนประกอบเครือข่าย เช่น ซอฟต์แวร์หรือเฟิร์มแวร์(firmware) ที่รันในฟรอนต์เอ็นคอนเซ็นเตรเตอร์(front-end concentrator)



รูปที่ 3-9 เอเจนท์แบบอินทิเกรตเต็ด

ข้อดี :-

1. เอเจนท์จะถูกใช้เพื่อความสามารถในการวิเคราะห์ระยะไกล(remote diagnostic) ซึ่งการใช้เอเจนท์แบบนี้จะง่ายในการเพิ่มฟังก์ชันที่ช่วยให้เข้าใจอ็อบเจ็คที่มอนิเตอร์มากขึ้น(more comprehensive)

2. ในกรณีที่เอเจนท์ในการมอนิเตอร์ถูกรวมเข้าไปในแอปพลิเคชั่นเฉพาะ เราก็สามารถที่จะปรับปรุงเอเจนท์ให้ตรงกับความต้องการของแอปพลิเคชั่นนั้น ๆ ได้ ซึ่งดีกว่าที่จะทำแค่ฟังก์ชันที่มีความสามารถทั่ว ๆ ไป

3. เพราะว่าเอเจนท์แบบนี้ถูกรวมอยู่ในฟังก์ชันที่ใช้ในการมอนิเตอร์และเข้าถึงทรัพยากรเดิมบ่อย ๆ ทำให้มันเป็นแหล่งของข้อมูลที่แน่นอน ถูกต้อง และแม่นยำ

4. การมอนิเตอร์ถูกรวมอยู่ภายใน(build-in) ดังนั้นจึงเป็นประโยชน์หรือเป็นข้อได้เปรียบในแง่ของค่าใช้จ่าย(cost) และการจัดการ(administration)

ข้อเสีย :-

1. การมอนิเตอร์อาจจะมีผลกระทบในทางลบกับสมรรถนะโดยรวมของแอปพลิเคชั่น เพราะเอเจนท์รันอยู่บนแพลตฟอร์ม(platform) เดียวกับบริการของเครือข่าย(network service) และแอปพลิเคชั่น

2. ค่าใช้จ่ายที่เสียไปในการเก็บรวบรวมผล(output) จากเอเจนท์ทั้งหมดอาจจะมากกว่าค่าของข้อมูลที่จะใช้ในแมนเนจเมนท์แอปพลิเคชั่น ยิ่งไปกว่านั้นการเสียเวลา(delays) ที่ฝัง(inherent) อยู่กับกระบวนการเก็บรวบรวมข้อมูลจะทำให้ข้อมูลบางส่วนหมดประโยชน์ไปก่อนที่จะถูกนำไปใช้

3. เอเจนท์อาจจะถูกติดตั้งใหม่หลายครั้งขึ้นอยู่กับว่ามันถูกรวมอยู่กับซอฟต์แวร์ที่ส่วนไหนบ้าง ตัวอย่างเช่น ถ้าเอเจนท์เป็นส่วนหนึ่งของแอปพลิเคชั่นซอฟต์แวร์ เอเจนท์ตัวใหม่จะต้องถูกติดตั้งในทุก

แอปพลิเคชั่นของเครือข่าย ถ้ามันถูกรวมเข้าไปในระบบปฏิบัติการ เอเจนท์ตัวใหม่ก็จะต้องถูกใส่เข้าไปในระบบปฏิบัติการทุกตัวด้วย เป็นต้น

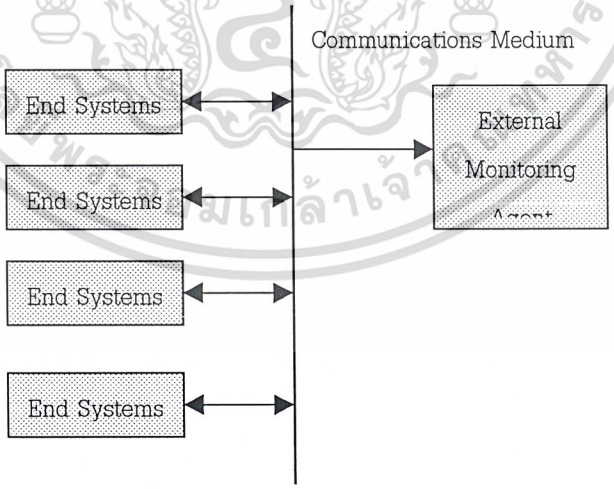
4. มีความเป็นไปได้ที่แต่ละฟังก์ชันของการมอนิเตอร์จะให้รูปแบบของรายงานที่แตกต่างกัน ทำให้งานในการพัฒนาแมนเนจเมนท์แอปพลิเคชั่นยุ่งยากและเสียค่าใช้จ่ายเพิ่มขึ้น ปัญหานี้สามารถหลีกเลี่ยงได้โดยการใช้รูปแบบรายงานที่เป็นมาตรฐาน แต่ก็จะต้องมีการใช้ความพยายามในการสร้างมาตรฐานนั้นขึ้นมา ซึ่งอาจจะก่อให้เกิดการเสียเวลาและค่าใช้จ่าย

5. การมอนิเตอร์เครือข่ายอาจมีลำดับความสำคัญ(priority) ที่ต่ำกว่าแอปพลิเคชั่นและระบบปฏิบัติการอื่น ทำให้มันไม่ได้รับทรัพยากรในการพัฒนาขึ้นมาใช้อย่างเพียงพอ

2.2 เอเจนท์แบบเอ็กเทอร์นอล(External monitoring agents)

เอเจนท์แบบนี้ถูกนำไปใช้กับบางส่วนประกอบที่อยู่ภายนอกซอฟต์แวร์ซึ่งให้บริการในเครือข่ายจริง ๆ วิธีที่เอเจนท์แบบนี้ใช้ในการมอนิเตอร์เครือข่ายคือการลอบฟัง/ดู(eavesdropping) ข้อมูลที่กำลังถูกส่งไปบนเครือข่าย ด้วยเหตุนี้จึงสามารถนำข้อมูลมาได้เฉพาะสถานะที่ใช้ร่วมกัน(shared state) ซึ่งสามารถแสดงให้เห็นได้จากเน็ตเวิร์คโพรโตคอล(network protocols) และไม่สามารถมอนิเตอร์สถานะภายในของส่วนประกอบที่เป็นซอฟต์แวร์ได้ แต่เนื่องจากเอเจนท์แบบนี้ง่ายในการพัฒนาจึงเป็นที่นิยมในเครือข่ายท้องถิ่น(Local Area Network: LAN) ซึ่งอุปกรณ์ปลายทั้งหมดใช้ตัวกลางในการส่งข้อมูลร่วมกัน ดังนั้นด้วยเอเจนท์แบบนี้เพียงตัวเดียวก็สามารถจับข้อมูลที่ต้องการได้ทั้งหมดในที่ ๆ หนึ่ง

เอเจนท์แบบเอ็กเทอร์นอลไม่จำเป็นต้องต่ออยู่กับเครือข่ายท้องถิ่นเท่านั้น เราสามารถนำมันไปต่อกับการเชื่อมต่อแบบจุดต่อจุด(point-to-point) ต่อกับสายบัสในระบบคอมพิวเตอร์ หรือเชื่อมเข้าไปในการติดต่อระหว่างระบบและเครือข่ายก็ได้



รูปที่ 3-10 เอเจนท์แบบเอ็กเทอร์นอล

ข้อดี :-

1. เพราะว่าอยู่ภายนอก เราจึงสามารถที่จะใช้มันในฮาร์ดแวร์ที่ใช้สำหรับการมอนิเตอร์โดยเฉพาะได้โดยไม่ต้องต่อสู้แย่งชิงทรัพยากรกับบริการในเครือข่ายและแอปพลิเคชั่น ซึ่งการทำเช่นนี้จะช่วยทำให้ลดการเสียสมรรถนะของเครือข่ายที่เกิดจากการมอนิเตอร์ได้ และยังสามารถสร้างโอกาสในการใช้ซอฟต์แวร์และฮาร์ดแวร์ที่เจาะจงเพื่อให้เกิดสมรรถนะเพิ่มขึ้น ซึ่งการทำเช่นนี้ไม่สามารถทำได้ในการออกแบบแบบอินทิเกรต

2. ใช้งบประมาณและทรัพยากรในการพัฒนาที่สามารถหาได้และเป็นไปได้ เนื่องจากเอเจนท์เพียงตัวเดียวก็สามารถจัดหาข้อมูลให้แก่ระบบปฏิบัติการและแอปพลิเคชั่นประเภทต่าง ๆ ได้

3. การเป็นส่วนประกอบอิสระทำให้ปรับปรุงเปลี่ยนแปลงและเพิ่มความสามารถให้กับเอเจนท์แบบนี้ได้ง่าย และข้อบกพร่องของมันก็จะส่งผลต่อบริการของเครือข่ายน้อยมาก

ข้อเสีย :-

1. เพราะการทำงานของมันเป็นการลอบฟัง/ดู จึงเป็นการเสี่ยงในเรื่องของความปลอดภัย ถึงแม้จะสามารถออกแบบให้ไม่มีความเสี่ยงนี้ได้แต่จะต้องเสียค่าใช้จ่ายเพิ่ม

2. ไม่สามารถรับประกันได้ว่าข้อมูลที่รวบรวมมาแสดงถึงสถานะที่ถูกต้อง แน่นอน เที่ยงตรงของอ็อบเจ็คที่ถูกมอนิเตอร์ ข้อมูลที่ได้มาเพียงแค่เป็นการประมาณเท่านั้น จึงไม่เป็นที่ยอมรับของแอปพลิเคชั่นบางตัว เช่นการคิดค่าบริการในการใช้งานเครือข่ายจากลูกค้า

3. เพราะว่าจะต้องเก็บรวบรวมข้อมูลจากอุปกรณ์ปลายจำนวนมาก จึงต้องการความจุในการเก็บข้อมูลที่ได้มาสูงมากส่งผลให้เสียค่าใช้จ่ายในการจัดการและผลิตสูงขึ้น

การที่เราจะตัดสินใจว่าควรจะเลือกใช้เอเจนท์แบบใดนั้นขึ้นอยู่กับปัจจัยหลายอย่าง แต่ปัจจัยหลักที่เราควรจะพิจารณาก่อนการเลือกใช้มีดังนี้

1. แบนด์วิธของเครือข่าย(Network bandwidth)

ในยุคแรกของระบบเครือข่ายคอมพิวเตอร์ การเชื่อมต่อทางกายภาพ(physical connections) ส่วนใหญ่ทำได้จากการเชื่อมแบบจุดต่อจุด โดยใช้โมเด็ม(modems) และวงจรโทรศัพท์แบบไดอัล-อัพ(dial-up telephone circuits) ซึ่งโดยทั่ว ๆ ไปมีความเร็วอยู่ที่ 300 bits/s ต่อมาความเร็วก็เพิ่มขึ้นเป็น 1200, 2400, 9600, 19200, ... Bits/s พร้อม ๆ กับมีการพัฒนาส่วนประกอบที่ใช้ในเครือข่ายให้เชื่อถือได้มากขึ้น ทุกวันนี้ได้มีมาตรฐานในการสื่อสารดิจิตอล(digital communications) ในวงจรโทรศัพท์ทั้งแบบส่วนตัว (private) และแบบสาธารณะ(public) สำหรับการเข้าถึงอุปกรณ์ปลายและสำหรับการสื่อสารระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ด้วยกันก็ทำได้เกิน 10 Mbits/s แต่ค่าใช้จ่ายที่สูงของการนำเทคโนโลยีชั้นสูงมาใช้ก็อาจเป็นสาเหตุทำให้ไม่สามารถนำมาใช้ได้ แต่ด้วยเทคโนโลยีแลนที่เสียค่าใช้จ่ายไม่มากนักเพราะมีการใช้สายสัญญาณในการสื่อสารและทรัพยากรร่วมกันทำให้เราสามารถมีเครือข่ายความเร็วสูงใช้งานได้

การเพิ่มขึ้นของอัตราข้อมูล(data rates) มีผลกระทบอย่างมากกับเอเจนท์แบบเอ็กเทอร์นอลเพราะนั่นหมายถึงจำนวนข้อมูลที่จะต้องจัดการมีมากขึ้นตามไปด้วย

2. สายส่งข้อมูลที่ใช้ร่วมกัน(Shared medium)

ในเครือข่ายที่สร้างจากการเชื่อมต่อแบบจุดต่อจุด เราพบว่าการมอนิเตอร์แบบเอ็กเทอร์นอลจะถูกนำมาใช้ไม่มากนักเนื่องมาจากค่าใช้จ่ายที่จะต้องเสียเพราะเอเจนท์หนึ่งตัวสามารถใช้มอนิเตอร์ระบบปลาย(End systems) ได้เพียง 2 ระบบ ดังนั้นการที่จะมอนิเตอร์ทั้งเครือข่ายจึงเป็นสิ่งที่ยากในแง่ของการลงทุน ด้วยเหตุนี้ในเบื้องต้นเอเจนท์แบบเอ็กเทอร์นอลจะถูกนำมาใช้ในการหาสาเหตุของความผิดพลาดเป็นจุด ๆ เท่านั้น

การมอนิเตอร์เครือข่ายสำหรับแวน(Wide Area Networks: WANs) ส่วนใหญ่จะประกอบด้วยเอเจนท์แบบอินทิเกรตเต็ดจำนวนหนึ่งที่ก่อให้เกิดการบันทึกเหตุการณ์(event logging) และเตือนภัย(alarm) โดยทั่วไปจะใช้คำสั่งง่าย ๆ ที่สามารถตั้งค่า(set) และบันทึกสถานะได้ในการเข้าถึงเอเจนท์เหล่านี้ บางครั้งจะอ่านสถานะได้จากแมนเนจเมนท์แอปพลิเคชั่นทางไกล(remote management application) ซึ่งจะบันทึกสถานะเหล่านั้นด้วยฐานข้อมูล(database) ธรรมดาทั่ว ๆ ไป หรือใช้สถานะดังกล่าวเพื่อการแสดงผลแบบเวลาจริง(real-time)

จากการเกิดเทคโนโลยีแลนและความนิยมในการใช้ทำให้เอเจนท์แบบเอ็กเทอร์นอลกลายเป็นทางเลือกที่เป็นไปได้และคุ้มค่าในการลงทุน(cost-effective)

3. ความเร็วของตัวประมวลผลและราคาของหน่วยความจำ(Processor speed and memory cost)

ความเร็วของตัวประมวลผลได้เพิ่มขึ้นมาเรื่อย ๆ (จากในยุคแรก ๆ ที่ 0.1 MIPs) พร้อมกับทางเดินของข้อมูล(data paths) จากเดิม 4 บิตก็เพิ่มเป็น 8, 16, 32, 64, ... ขณะเดียวกันราคาของหน่วยความจำก็ได้ลดลงและขนาดแพ็คเกจ(package) ก็เล็กลง เทคโนโลยีที่พัฒนาไปเรื่อย ๆ นี้กระทบต่อความสามารถและการออกแบบตัวมอนิเตอร์เครือข่าย นอกจากที่กล่าวมาแล้วยังมีอุปกรณ์เก็บข้อมูล(storage device) ที่เล็กลง เร็วขึ้น จุได้มากขึ้น และความเร็วในการเข้าถึง(access speed) ก็เพิ่มมากขึ้นด้วย เทคโนโลยีในการแสดงผลก็ได้รับการพัฒนาจากอุปกรณ์ที่แสดงเป็นอักขระ(character cell device) เป็นจอกราฟฟิค(bit-mapped graphics displays) วินโดส์(windows) , ไอคอน(icon), อุปกรณ์ชี้(pointing devices) และความสะดวกสบายอื่น ๆ ก็หาได้แม้แต่ในระบบที่มีประสิทธิภาพไม่สูง(low-end systems) แหล่งจ่ายไฟ(power supply) และส่วนประกอบอื่น ๆ ก็มีขนาดเล็กลงกว่าเดิม ส่งผลให้ระบบสามารถเคลื่อนย้ายได้สะดวกมากขึ้น และทูล(tools) ที่ช่วยพัฒนาโปรแกรมก็ได้รับการปรับปรุง ทำให้สร้างแอปพลิเคชั่นในการมอนิเตอร์เครือข่ายที่ซับซ้อนได้ง่ายมากขึ้น

3.7 เราควรมอนิเตอร์เครือข่ายที่เลเยอร์ใด(What layer to monitor?)

นอกจากเราจะสามารถแบ่งตัวมอนิเตอร์เครือข่ายเป็นแบบอินทิเกรตเต็ดและเอ็กเทอร์นอลแล้ว เรายังสามารถแบ่งตามเลเยอร์(layer) ของเครือข่ายที่มันทำงานอยู่ได้ด้วย

การสื่อสารในเครือข่ายถูกนำมาใช้โดยซอฟต์แวร์ที่ซับซ้อนที่อยู่ในระบบปลายและระบบที่เชื่อมต่อระบบปลายต่าง ๆ เข้าด้วยกัน(End and Intermediate systems) ซึ่งซอฟต์แวร์เหล่านี้ถูกออกแบบในลักษณะเลเยอร์ซึ่งแต่ละตัวมีโพรโตคอลที่ใช้แตกต่างกันไป รวมทั้งสถานะของมันที่ต้องการการมอนิเตอร์และจัดการ โดยทั่ว ๆ ไปเอเจนท์ที่ทำหน้าที่มอนิเตอร์จะถูกออกแบบให้ทำงานกับเลเยอร์เพียงเลเยอร์เดียว (แต่เป็นไปได้ที่จะทำงานในหลายเลเยอร์) สำหรับตัวมอนิเตอร์แบบอินทิเกรตเต็ด การจะเลือกว่าให้ทำงานบนเลเยอร์ใดนั้นก็ขึ้นอยู่กับฐานข้อมูลที่ใช้จัดการ(Management Information Base: MIB) ที่ถูกกำหนดขึ้น

มา สำหรับตัวมอนิเตอร์แบบเอ็กเทอร์นอลจะมีสมรรถนะและได้รับการทำออกมาจำหน่ายมากกว่าเพราะไม่ต้องยุ่งเกี่ยวกับรายละเอียดของแอปพลิเคชั่นทั้งหมด

เราสามารถให้ตัวอย่างของความต้องการที่แตกต่างในการมอนิเตอร์ในเลเยอร์ต่าง ๆ ของเครือข่าย ซึ่งจะช่วยให้เข้าใจการออกแบบตัวมอนิเตอร์ได้มากขึ้นดังนี้

1. เลเยอร์ดาต้าลิงค์(DataLink Layer)

เลเยอร์นี้มีหน้าที่เกี่ยวกับการเคลื่อนย้ายของข้อมูลจริง ๆ จากจุดหนึ่งในเครือข่ายไปยังจุดถัดไป โดยทั่วไปเลเยอร์นี้คือเลเยอร์ต่ำสุดของซอฟต์แวร์เครือข่าย และใกล้กับตัวกลางทางกายภาพ(physical medium) ที่สุด การมอนิเตอร์ในเลเยอร์นี้จะใช้เพื่อตรวจจับข้อผิดพลาดของซอฟต์แวร์และฮาร์ดแวร์ซึ่งเป็นผลให้เกิดการสูญเสียหรือหายไปของข้อมูล ปัญหาที่เกิดขึ้นในเลเยอร์ที่สูงกว่า(upper layer) มักจะถูกตรวจจับและแยกแยะโดยการมอนิเตอร์ที่เลเยอร์นี้(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) บนเครือข่ายทั้งหมดโดยแอปพลิเคชั่น และ/หรือผู้ใช้ ซึ่งเป็นประโยชน์ในการช่วยทำโหลดบาลานซิ่ง (load balancing) และแอปพลิเคชั่นทางด้านการบันทึกการใช้งาน (Accounting applications)

5. เลเยอร์แอปพลิเคชั่น(Application Layer)

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

บทที่ 4

การติดต่อใช้งานแพ็คเกจไดรเวอร์(Packet driver usage)

ข้อกำหนดของ พีซี / ทีซีพีแพ็คเกจไดร์เวอร์ (PC/TCP Packet Driver Specification)

แพ็คเกจไดรเวอร์ คือ โปรแกรมที่ช่วยอินเทอร์เฟสกับการ์ดแลน ซึ่งหลักการในการติดต่อกับการ์ดแลนด้วยวิธีที่จะอธิบายนี้สามารถใช้ได้กับแพ็คเกจไดรเวอร์ทุกแบบที่เป็นของ Crynwr Software เช่น NE2000, NE1000, 3C505, AT1500, ISOLAN เป็นต้น ซึ่งในการติดต่อกับแพ็คเกจไดร์เวอร์จะต้องอาศัยข้อมูลประกอบต่อไปนี้

4.1 ข้อกำหนดในเอกสารฉบับนี้ (Document Conventions)

ตัวเลขที่ปรากฏในเอกสารฉบับนี้จะแสดงในรูปแบบของภาษาซี (C-style representation) เช่น เลข 11 จะแทนด้วยในรูปของเลขฐานสิบหก 0x0B และเลขฐานแปดจะแทนด้วย 013 ซึ่งจะใช้ในการอ้างอิงหมายเลขฮาร์ดแวร์เครือข่าย (network hardware address) , หมายเลขเครื่องปลายทาง (destination address) ,หมายเลขเครื่องต้นทาง (source address) , ข้อมูลการแยกสัญญาณ (demultiplexing) เป็นต้น

4.2 ความรู้เบื้องต้น (Introduction and Motivation)

ในเอกสารฉบับนี้จะอธิบายถึงการเขียนโปรแกรมเพื่อติดต่อกับ FTP Software Packet Driver ซึ่งสามารถใช้งานได้ง่ายและ การเขียนโปรแกรมติดต่อกับแพ็คเกจไดร์เวอร์สามารถทำให้โปรแกรมหลายหลายโปรแกรมร่วมกันใช้อินเตอร์เฟสที่ดาต้าลิงค์ระดับเดียวกันได้ แพ็คเกจไดร์เวอร์จะทำการคัดแยกแพ็คเกจที่เข้ามาระหว่างโปรแกรมประยุกต์ (application) โดยการแยกคัดจากชนิดของสื่อของเครือข่ายมาตรฐาน (network media 's standard packet type) หรือ การเข้าถึงของการบริการของแพ็คเกจไดว์เวอร์

จุดประสงค์สำคัญของข้อกำหนดนี้คือ การที่สามารถให้ระดับชั้นของโพรโตคอล (Protocol Stack) ทำการจัดการโดยอิสระต่อชนิดของการ์ดเชื่อมต่อเครือข่าย (Network interface) ที่อยู่ในเครื่องแต่ละชนิด

แพ็คเกจไดร์เวอร์ได้จัดเตรียมฟังก์ชั่นในการเข้าถึงชนิดของแพ็คเกจไดว์เวอร์ , การสิ้นสุดการเข้าถึงแพ็คเกจ,การส่งแพ็คเกจ, การเก็บสถิติบนการ์ดเชื่อมต่อเครือข่าย และ การเก็บข้อมูลเกี่ยวกับการ์ดเชื่อมต่อเครือข่าย

การจัดการเกี่ยวกับโพรโตคอลที่ใช้แพ็คเกจไดรเวอร์สามารถใช้งานได้ร่วมกันในเครื่องคอมพิวเตอร์ส่วนบุคคล (PC) อย่างสมบูรณ์และสามารถใช้ประโยชน์จากบริการอื่นๆได้อีก แต่โปรแกรมประยุกต์หลายหลายโปรแกรม ที่ไม่ได้ใช้ไดร์เวอร์ จะไม่สามารถอยู่ร่วมกันในเครื่องเดียวกันได้อย่างสมบูรณ์ โดยการใช้แพ็คเกจไดร์เวอร์ ผู้ใช้สามารถรันโพรโตคอล ทีซีพี/ไอพี (TCP/IP), เอ็กเอ็นเอส (XNS), และโพรโตคอลที่จดทะเบียนเช่น DECnet , Banyan's , LifeNet's , Novell's หรือ 3Com's ได้ โปรแกรมประยุกต์สามารถใช้ แพ็คเกจไดร์เวอร์ในการรันบนอุปกรณ์ฮาร์ดแวร์ตัวใหม่ได้ โดยไม่ต้องทำการปรับแต่งอย่างอื่นอย่างใดอื่น

ระดับของแพ็คเกจไดร์เวอร์มีหลายระดับซึ่งได้อธิบายไว้ในข้อกำหนดนี้ไว้เรียบร้อยแล้ว ระดับการใช้งานของแพ็คเกจไดร์เวอร์ได้อธิบายไว้ในข้อกำหนดนี้เรียบร้อยแล้ว อันแรกคือ แพ็คเกจไดร์เวอร์

ขั้นพื้นฐาน (basic) ซึ่งได้จัดเตรียมฟังก์ชั่นขนาดเล็ก แต่ง่ายต่อการใช้งานและใช้ทรัพยากรของเครื่องน้อย แพ็คเกจไดรเวอร์ขั้นพื้นฐานมีการทำงาน ในการบรอดคาสท์ (broadcast) และการรับแพ็คเกจ (receive packet) แพ็คเกจไดร์เวอร์อันดับที่สองคือ แพ็คเกจไดร์เวอร์ขยาย (extend packet driver) ซึ่งเป็นแพ็คเกจไดร์เวอร์ที่ครอบคลุมมากกว่าแพ็คเกจไดร์เวอร์ขั้นพื้นฐาน แพ็คเกจไดร์เวอร์แบบขยายนี้จะสนับสนุนฟังก์ชั่นที่ค่อยข้างใช้งานน้อยเช่น ฟังก์ชั่นมัลติคลาส (multicast function) และ เก็บรวบรวมสถิติในการใช้งานของการ์ดอินเตอร์เฟสและ ให้ใช้งานได้ในโปรแกรมประยุกต์ทั่วๆไป แพ็คเกจไดร์เวอร์อันดับสามได้แก่ ประสิทธิภาพสูง (high-performance) ซึ่งจะสนับสนุนการพัฒนาประสิทธิภาพ และการปรับแต่งแพ็คเกจ

4.3 การแยกแยะอินเตอร์เฟสเครือข่าย (Identifying network interfaces)

อินเตอร์เฟสเครือข่ายประกอบตัวเลข 3 ส่วนคือ ตัวเลขตัวแรกคือ คลาสอินเตอร์เฟส (class interface) คลาสจะเป็นตัวบอกชนิดของสื่อที่การ์ดสนับสนุน : DEC/Intel/Xerox (DIX) Ethernet , IEEE 802.3 Ethernet , IEEE 802.5 Token Ring , ProNET-10 , Appletalk , serial line etc.

ตัวเลขส่วนที่สองคือ ชนิดของการ์ด (type of interface) : ส่วนนี้จะระบุถึงการ์ดที่สนับสนุนคลาสของสื่อเครือข่าย ได้แก่ 3Com 3C503 หรือ 3C505 , Interlan NI5210 , Univation , BICC Data Networks ISOLAN ,Ungermann-Bass NIC , etc. ชนิดของการ์ดสำหรับ IEEE 802.5 อาจมีชื่อเป็น IBMToken Ring adapter , Proteon p1340 และอื่นๆ

ตัวเลขส่วนสุดท้ายคือ ตัวเลขของการ์ด (number) ถ้าเครื่องคอมพิวเตอร์ใดที่มีการ์ดมากกว่าหนึ่งการ์ด การ์ดนั้นต้องมีตัวเลขของการ์ดที่ต้องแตกต่างจากตัวเลขของคลาสและชนิดของการ์ด

ในส่วนของภาคผนวก ก จะแสดงรายละเอียดของตัวเลขคลาสและชนิดของการ์ด โดยตัวเลขคลาสจะเป็นตัวเลขขนาด 8 บิต และ ตัวเลขที่แสดงชนิดของการ์ดจะมีขนาด 16 บิต โดยตัวเลขเหล่านี้จะถูกตั้งขึ้นโดย FTP Software .

สำหรับชนิดของการ์ด 0xFFFF หมายถึง ชนิดของการ์ดใดๆซึ่งจะเข้าคู่กับการ์ดที่กำหนดในตัวเลขคลาส และไม่จำเป็นที่ต้องกำหนดการ์ดเป็นชนิดใดๆก็ได้ซึ่งเป็น 0 ซึ่งมักจะมีค่าตรงกับอินเตอร์เฟส (interface) ส่วนแรกตามที่กำหนดโดยคลาสและชนิดของการ์ด

4.4 การเริ่มใช้งานของไดร์เวอร์ (Initiating driver operations)

แพ็คเกจไดร์เวอร์จะถูกเรียกโดยซอฟต์แวร์อินเตอร์รัพท์ ตั้งแต่ 60 ถึง 80 ซึ่งในเอกสารฉบับนี้มิได้กำหนดเจาะจงว่าเป็นอินเตอร์รัพท์เบอร์ใด แต่จะอธิบายถึงกลไกในการทำงานของการระบุถึงอินเตอร์รัพท์ที่ไดร์เวอร์นั้นเรียกใช้

ตัวควบคุม (handler) สำหรับอินเตอร์รัพท์ถูกกำหนดให้เริ่มด้วย 3 ไบต์ของโค๊ดที่ทำงานได้ (executable code) แล้วตามด้วยสตริงที่มีรหัสจบ (null - terminated ASCII text) “PKT DRVR” การค้นหาอินเตอร์รัพท์ที่ถูกใช้โดยไดร์เวอร์ทำได้โดยผ่านทางตัวควบคุมตั้งแต่อินเตอร์รัพท์ที่ 60 ถึง อินเตอร์รัพท์ที่ 80 จนกว่าจะเจอสตริง “PKT DRVR” ใน 12 ไบต์แรกนับจากจุดตั้งต้น

4.5 การคัดแยกในชั้นลิงค์เลเยอร์ (Link-layer demultiplexing)

ถ้าสื่อเครือข่ายมาตรฐานสนับสนุนโพรโตคอลต่างๆกัน (เช่น TCP/IP ,XNS, OSI) มันจะต้องกำหนดกลไกของระดับการลิงค์ซึ่งจะต้องอนุญาตให้โฮสท์ (host) เป็นตัวตัดสินใจว่าโพรโตคอลใดที่แพ็คเกจนั้น เกี่ยวข้องด้วย ใน DIX Ethernet , จะเรียกส่วนนี้ว่า "ethertype" ซึ่งมีขนาด 16 บิตแล้วตามด้วยแอดเดรสของตำแหน่งต้นทางและปลายทางซึ่งมีขนาด 6 บิต ใน IEEE 802.3 ที่ซึ่งใช้ ส่วนหัว (header) ของ 802.2 ข้อมูลในส่วนนี้จะอยู่ในส่วนหัวของ 802.2 (ใน Proteon 's ProNET-10 ใช้ส่วนนี้ในขนาด 32 บิต เรียกว่า "ฟิลด์ชนิด" (type field) ในสื่อมาตรฐานอื่น ๆ อาจจะใช้การคัดแยกผ่านกระบวนการของมันเอง

ฟังก์ชั่น *access_type()* ทำหน้าที่ในการเข้าถึงการคัดแยกในชั้นลิงค์เลเยอร์ โดยการเรียกใช้แต่ละครั้งจะทำการติดตั้งจุดมุ่งหมายสำหรับแพ็คเกจ

4.6 การเขียนโปรแกรมติดต่อกับแพ็คเกจไดร์เวอร์ (Programming interface)

ทุกๆฟังก์ชั่นจะถูกเรียกได้โดยผ่านทางซอฟต์แวร์อินเตอร์รัพท์ที่ถูกกำหนดโดยกลไกของแพ็คเกจไดร์เวอร์ที่ได้กล่าวมาแล้วในข้างต้น โดยจะเรียกฟังก์ชั่นที่ต้องการผ่านทางรีจิสเตอร์ AH

แฮนเดิล (handle) เป็นค่าตัวเลขที่เกี่ยวข้องกับแต่ละระดับการคัดแยกของการ์ดแต่ละชนิด ที่ได้มาจากฟังก์ชั่น *access type ()* ซึ่งมันอาจจะเป็นพอยเตอร์หรือตารางออฟเซ็ตก็ได้ ถ้าแอพพลิเคชันใดใช้มากว่า 2 แพ็คเกจไดร์เวอร์ ค่าตัวเลขแฮนเดิลที่ได้จากแพ็คเกจไดร์เวอร์คนละตัวกันต้องเป็นค่าเดียวกัน

เราสังเกตเห็นว่าฟังก์ชันที่กำหนดไว้ข้างล่างนี้เป็นฟังก์ชั่นของแพ็คเกจไดร์เวอร์แบบขยาย และแพ็คเกจไดร์เวอร์ประสิทธิภาพสูง เพราะฟังก์ชันเหล่านั้นไม่ต้องการการทำงานขั้นพื้นฐานของเครือข่าย และเราอาจพิจารณาว่าเป็นฟังก์ชั่นที่ไว้เผื่อเลือกก็ได้ โปรแกรมที่ใช้ฟังก์ชันเหล่านี้ควรจะเรียกใช้ฟังก์ชัน *driver_info()* เพื่อเป็นตัวกำหนด ถ้าฟังก์ชันเหล่านั้นยังมีอยู่ในแพ็คเกจไดร์เวอร์ที่ใช้

4.6.1 เงื่อนไขการใช้งาน (Entry condition)

แอพพลิเคชั่น FTP Software ซึ่งเรียกใช้แพ็คเกจไดร์เวอร์ถูกเขียนขึ้นโดยภาษาไมโครซอฟท์ซี (Microsoft C) และภาษาแอสเซมบลี้ (Assembly) รีจิสเตอร์ที่จำเป็นทุกตัวจะถูกเซฟโดยรูทีน FTP's ก่อนที่จะเรียกคำสั่ง INT เพื่อเรียกแพ็คเกจไดร์เวอร์ขึ้นมาทำงาน โดยฟังก์ชัน *receiver()* จะทำการเซฟรีจิสเตอร์ DS,SS,SP และแฟล็ก และเรียกกลับมาใช้คืนได้ ส่วนรีจิสเตอร์ส่วนที่เหลืออื่นๆอาจจะเปลี่ยนแปลงและถูกเซฟโดยแพ็คเกจไดร์เวอร์ โพรเซสเซอร์อินเตอร์รัพท์ (Processor interrupts) อาจจะมีสถานะเอนเนเบิ้ล (enable) ในขณะที่อยู่ในระหว่างการเรียกใช้

เมื่อเราเรียกใช้แพ็คเกจไดร์เวอร์ในคลาสที่ 1 , PC/TCP โดยปกติจะสร้าง *access_type()* ขึ้นมา 5 ชุดเพื่อเรียกโพรโตคอล IP, ARP และอีก 3 ชนิดของแพ็คเกจไดร์เวอร์ Berkeley Trailer

4.6.2 การเรียงลำดับระดับไบต์และบิต (Byte and Bit Ordering)

นักพัฒนาโปรแกรมควรจะสังเกตเห็นว่าในบรรดาระบบเครือข่ายและโพรโตคอลทั้งหลายมักจะมีการเรียกใช้งานระดับบิตขนาด 16 บิต ซึ่งตรงกันข้ามกับการเรียกใช้งานแบบเนทีฟของเครื่อง PC (ยกเว้น 802.5 Token Ring) ซึ่งหมายความว่า ค่าของ DEC-Intel-Xerox ethertype จะผ่านค่ามายัง *access_type()* ฟังก์ชันจะต้องถูกสลับค่า ใน IEEE 802.3 ค่าฟิลด์ *length* ต้องการค่าแฮนเดิลที่คล้ายกันและสนใจต่อแพ็คเกจไดร์เวอร์ที่จะถูกส่งผ่านให้ *send_pkt()* ฟังก์ชัน ดังนั้นทุกๆฟิลด์จะต้องเรียงตาม

ลำดับจริงๆ นักพัฒนาโปรแกรมที่ ทำงานเกี่ยวข้องกับ MSB LANs (802.5 และ FDDI) ควรจะต้องทราบว่าค่าที่แสดงหมายเลขเครื่อง (MAC address) จะเปลี่ยนลำดับบิต ทั้งนี้ขึ้นอยู่กับว่ามันแสดงอยู่ในส่วนหัวหรือส่วนข้อมูล

4.6.3 ฟังก์ชัน *driver_info()*

driver_info(handle) *AH == 1, AL == 255*

int handle; BX */* Optional */*

error return:

carry flag set

error code *DH*

possible errors:

BAD_HANDLE */* older drivers only */*

non-error return:

carry flag clear

version *BX*

class *CH*

type *DX*

number *CL*

name *DS:SI*

functionality *AL*

1 == basic functions present.

2 == basic and extended presen

5 == basic and high-performance.

6 == basic, high-performance, extended.

255 == not installed.

ฟังก์ชันนี้จะส่งค่าข้อมูลเกี่ยวกับการ์ด ค่า *Version* แสดงรุ่นของฮาร์ดแวร์ไดร์เวอร์ ค่า *handle* เป็นค่าที่ได้จากการเรียกใช้ฟังก์ชัน *access_type()*

4.6.4 ฟังก์ชัน *access_type()*

int access_type(if_class, if_type, if_number, type, typelen, receiver) *AH==2*

int if_class; *AL*

int if_type; *BX*

int if_number; *DL*

*char far *type;* *DS:SI*

unsigned typelen; *CX*

*int (far *receiver)();* *ES:DI*

error return:

carry flag set

error code *DH*

possible errors:

NO_CLASS

NO_TYPE

NO_NUMBER

BAD_TYPE

NO_SPACE

TYPE_INUSE

non-error return:

carry flag clear

handle *AX*

receiver call:

*(*receiver)(handle, flag, len [, buffer])*

int handle; *BX*

int flag; *AX*

unsigned len; *CX*

if AX == 1,

*char far *buffer;* *DS:SI*

การเริ่มต้นเข้าถึงแพ็คเกจไดร์เวอร์ตามที่ระบุโดยค่า *type* โดยค่า *type* จะเป็นค่าพอยเตอร์ที่ชี้ไปยังชนิดของแพ็คเกจตามที่ระบุ ค่า *typelen* แสดงถึงความยาวเป็นจำนวนไบต์ใน ฟิลด์ *type receiver* เป็นพอยเตอร์ที่ของซับรูทีนซึ่งจะถูกเรียกใช้เมื่อได้รับแพ็คเกจ ถ้าค่า *typelen* มีค่าเป็น 0 นั่นหมายถึงตัวเรียก (caller) ตัวการทุกๆแพ็คเกจ เมื่อเราได้รับแพ็คเกจ ตัว *receiver* จะถูกเรียกจากแพ็คเกจไดร์เวอร์ 2 ครั้ง โดยครั้งแรกจะถูกเรียกเพื่อร้องขอ *buffer* จากแอพพลิเคชั่นเพื่อใช้เป็นที่เก็บแพ็คเกจ ค่า AX จะมีค่าเป็น 0 ในการเรียกครั้งนี้ แอพพลิเคชั่นจะส่งค่าพอยเตอร์สำหรับ *buffer* โดยผ่านทาง *ES:DI* ถ้าแอพพลิเคชั่นไม่มี *buffer* นั้นมันจะส่งค่ากลับเป็น 0 ในค่า *ES:DI* และไดร์เวอร์จะไม่สนใจแพ็คเกจนั้น

ใน AX=0 ค่า CX จะมีความสำคัญมากซึ่งจะแสดงค่าความยาวของแพ็คเกจ ดังนั้น *receiver* จะเป็นตัวจองขนาด *buffer* จริงๆ ซึ่งค่า *length* นี้จะปรากฏอยู่ในส่วนหัวของ MAC และข้อมูลที่ได้รับทั้งหมด

4.6.5 ***release_type()***

int release_type(handle) *AH == 3*

int handle; *BX*

error return:

carry flag set

error code *DH*

possible errors:

BAD_HANDLE

non-error return:

carry flag clear

ฟังก์ชันนี้จะสิ้นสุดการเข้าถึงแพ็คเกจซึ่งค่าแฮนเดิล ที่ได้รับจากฟังก์ชั่น *access_type()* จะสิ้นสุดความถูกต้องทันที

4.6.6 ***send_pkt()***

int send_pkt(buffer, length) *AH == 4*

*char far *buffer;* *DS:SI*

unsigned length; *CX*

error return:

carry flag set

error code *DH*

possible errors:

CANT_SEND

non-error return:

carry flag clear

เป็นฟังก์ชันในการส่งข้อมูล แอพพลิเคชันจะบรรจุข้อมูลและส่วนหัวของเครือข่าย

4.6.7 ***terminate()***

terminate(handle) *AH == 5*

int handle; *BX*

error return:

carry flag set

error code *DH*

possible errors:

BAD_HANDLE

CANT_TERMINATE

non-error return:

carry flag clear

เป็นฟังก์ชันในการจบแพ็คเกจไดร์เวอร์ พร้อมด้วยค่าแฮนเดิล ถ้าการเรียกฟังก์ชันสมบูรณ์ จะออกจาก MS-DOS และคืนหน่วยความจำให้ดั้งเดิม

4.6.8 ***get_address()***

get_address(handle, buf, len) *AH == 6*

int handle; *BX*

*char far *buf;* *ES:DI*

int len; *CX*

error return:

carry flag set

error code *DH*

possible errors:

BAD_HANDLE

NO_SPACE

non-error return:

carry flag clear

length *CX*

จะทำการคัดลอกค่าแอดเดรสของเครื่องท้องถิ่น (local net address) ลงใน *buffer* โดยจะแสดงขนาดผ่านทาง *CX* ถ้าไม่มี *buffer* จะแสดงค่าผิดพลาด *NO_SPACE* ออกมาซึ่งจะหมายถึงมีเนื้อที่ไม่เพียงพอในการเก็บแอดเดรส ถ้าค่าแอดเดรสมีการเปลี่ยนแปลงโดยฟังก์ชัน *set_address()* มันจะส่งค่าแอดเดรสใหม่ลงใน *buffer* นี้

4.6.9 ***reset_interface()***

reset_interface(handle) *AH == 7*

int handle; *BX*

error return:

carry flag set

error code *DH*

possible errors:

BAD_HANDLE

CANT_RESET

non-error return:

carry flag clear

ทำการรีเซ็ตอินเตอร์เฟสพร้อมกับค่าแฮนเดิล ทำการยกเลิกการส่งแพ็คเกจและทำการเริ่มต้นติดตั้ง *receiver* ใหม่ ค่าแอดเดรสของเครื่องท้องถิ่นจะถูกรีเซ็ตเป็นค่าดีฟอลต์ (default) , ค่า *multicast* จะถูกเคลียร์ *receiver mode* จะถูกเซ็ตค่าเป็น 3

4.6.10 ***get_parameters()***

high-performance driver function

get_parameters() *AH = 10*

error return:

carry flag set

error code *DH*

possible errors:

BAD_COMMAND */* No high-performance support */*

non error return:

carry flag clear

*struct param far *;* *ES:DI*

```
struct param {
unsigned char   major_rev;      /* Revision of Packet Driver spec */
unsigned char   minor_rev;      /*  this driver conforms to. */
unsigned char   length;         /* Length of structure in bytes */
unsigned char   addr_len;       /* Length of a MAC-layer address */
unsigned short  mtu;            /* MTU, including MAC headers */
unsigned short  multicast_aval; /* Buffer size for multicast addr */
unsigned short  rcv_bufs;       /* (# of back-to-back MTU rcvs) - 1 */
unsigned short  xmt_bufs;       /* (# of successive xmits) - 1 */
unsigned short  int_num;        /* Interrupt # to hook for post-EOI
                                   processing, 0 == none */
```

};

ประสิทธิภาพของแอพพลิเคชั่นจะได้รับประโยชน์ จากฟังก์ชันนี้ การเรียกใช้ฟังก์ชัน *get_parameter()* จะได้รับค่าตัวเลขไดร์เวอร์พารามิเตอร์ (driver parameter)

ค่า *major_rev* และ *minor_rev* ฟิลด์เป็นค่าตัวเลขเมเจอร์และไมเนอร์ของการแก้ไขปรับปรุงข้อกำหนดของไดร์เวอร์นี้

ค่า *length* ฟิลด์ ถูกใช้เป็นตัวตัดสินความถูกต้อง (*valid*) ในเอกสารฉบับนี้ *length* จะมีค่าเป็น 14

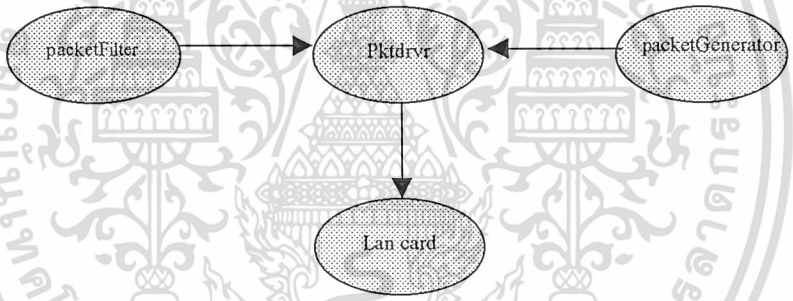
บทที่ 5
การออกแบบและการพัฒนาซอฟต์แวร์
(Software Design and Development)

5.1 การออกแบบซอฟต์แวร์(Software Design)

โครงงานนี้เป็นการพัฒนาซอฟต์แวร์เครื่องมือช่วยผู้ดูแลระบบในการบริหารระบบเครือข่ายคอมพิวเตอร์ ซึ่งในการออกแบบได้ใช้วิธีการออกแบบซอฟต์แวร์แบบอ็อบเจ็คโอเรียนเต็ด(Object-Oriented Design: OOP)ในการพัฒนาโดยสามารถแบ่งซอฟต์แวร์ออกเป็นอ็อบเจ็ค(Object) ส่วนต่าง ๆ ได้ดังต่อไปนี้

1. อ็อบเจ็คจับและส่งแพ็คเกจ(pktdrvr object)

ทำหน้าที่ติดต่อกับการ์ดแลนเพื่อสั่งให้การ์ดแลนจับแพ็คเกจในแลนหรือส่งแพ็คเกจออกสู่แลน และอ็อบเจ็คนี้จะมีการติดต่อกับอีก 2 อ็อบเจ็คคืออ็อบเจ็คกรองแพ็คเกจและอ็อบเจ็คสร้างแพ็คเกจดังรูปที่ 5-1



รูปที่ 5-1 แสดงการติดต่อกับอ็อบเจ็คอื่นของอ็อบเจ็คจับและส่งแพ็คเกจ

2. อ็อบเจ็คกรองแพ็คเกจ(packetFilter object)

ทำหน้าที่กรองแพ็คเกจซึ่งได้มาจากการจับโดยอ็อบเจ็คจับและส่งแพ็คเกจที่กล่าวถึงข้างต้น โดยสามารถกรองแพ็คเกจได้ดังนี้

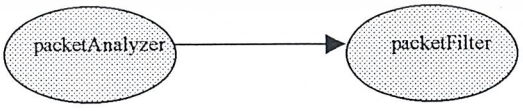
- กรองโดยไอพีแอดเดรสตามที่ผู้ใช้กำหนด (method filterByIP)
- กรองโดยแมคแอดเดรสตามที่ผู้ใช้กำหนด (method filterByMac)
- กรองโดยโพรโตคอลในดาต้าลิงค์เลเยอร์ 4 ชนิดคือ Ethernet 802.3, Ethernet 802.2, Ethernet SNAP, Ethernet II (method filterByDtlProtocol)
- กรองโดยโพรโตคอลในทรานสปอร์ตเลเยอร์ 4 โพรโตคอลคือ TCP, UDP, SPX และ ICMP ซึ่งเป็นโพรโตคอลในเน็ตเวิร์คเลเยอร์แต่จะถูกส่งโดยมีเฮดเดอร์ของโพรโตคอลไอพีนำ (method filterByTransportProtocol)

- กรองโดยโพรโตคอลในเน็ตเวิร์คเลเยอร์ 4 โพรโตคอลคือ IP, ARP, RARP, IPX (method filterByNetworkProtocol)
- กรองโดยขนาดแพ็คเกจตามที่ผู้ใช้กำหนด (method filterBySize)
- กรองโดยหมายเลขพอร์ตตามที่ผู้ใช้กำหนด (method filterByPortNumber)

3. อ็อบเจ็ควิเคราะห์แพ็คเกจ(packetAnalyzer object)

มีการติดต่อกับอ็อบเจ็คกรองแพ็คเกจดังรูปที่ 5-2 อ็อบเจ็คนี้ทำหน้าที่นำแพ็คเกจที่ได้จากการกรองมาวิเคราะห์หาสิ่งดังต่อไปนี้

- เปอร์เซ็นต์ในการใช้งานโพรโตคอลระดับแอพพลิเคชั่นเลเยอร์ 9 โพรโตคอลคือ FTP, Telnet, SMTP, Gopher, HTTP, POP3, NNTP, SNMP, IRC (method usedApp)
- เปอร์เซ็นต์ในการใช้งานโพรโตคอลระดับทรานสปอร์ตเลเยอร์ 4 โพรโตคอลคือ TCP, UDP, SPX และ ICMP ซึ่งเป็นโพรโตคอลในชั้นเน็ตเวิร์คเลเยอร์แต่จะมีเฮดเดอร์ของโพรโตคอลไอพีนำ (method usedTransport)
- เปอร์เซ็นต์ในการใช้งานโพรโตคอลระดับเน็ตเวิร์คเลเยอร์ 4 โพรโตคอลคือ IP, ARP, RARP, IPX (method usedNetwork)
- เปอร์เซ็นต์ในการใช้งานโพรโตคอลระดับดาต้าลิงค์เลเยอร์ 4 ชนิดคือ Ether 802.3, Ethernet 802.2, Ethernet SNAP, Ethernet II (method usedEthernet)
- เปอร์เซ็นต์ในการกระจายขนาดของแพ็คเกจโดยแบ่งขนาดออกเป็น 4 กลุ่มคือ 60 - 420 ไบต์, 421 – 780 ไบต์, 781 – 1140 ไบต์, 1141 – 1514 ไบต์ (method distributedSize)
- โหนดที่ส่งแพ็คเกจออกมามากที่สุดในจำนวนแพ็คเกจที่จับมาได้ทั้งหมด (method frequentSendingNode)
- เปอร์เซ็นต์การใช้งานโหนดที่กำหนดโดยแมคแอดเดรส (method percentUsedNode)
- เปอร์เซ็นต์การใช้งานสเตชั่นที่กำหนดโดยไอพีแอดเดรส (method percentUsedPacket)

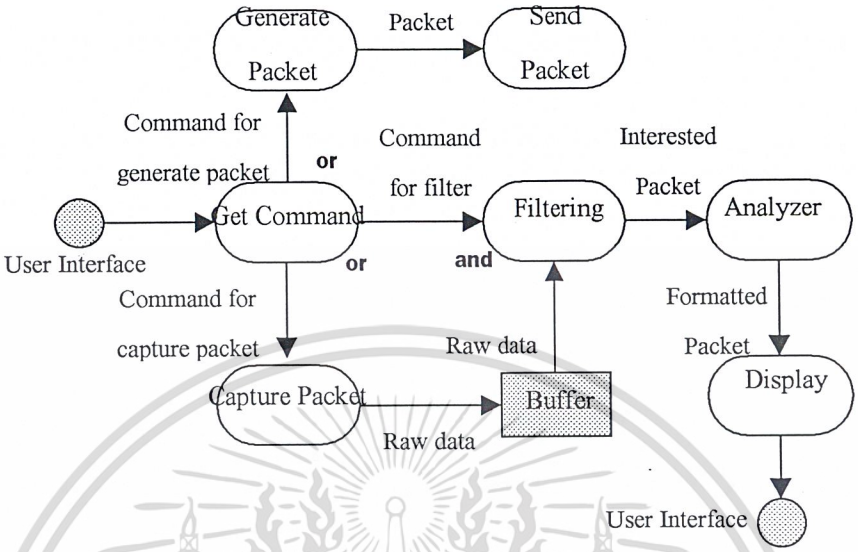


รูปที่ 5-2 แสดงการติดต่อกับอ็อบเจ็คอื่นของอ็อบเจ็ควิเคราะห์แพ็คเกจ

4. อ็อบเจ็คสร้างแพ็คเกจ(generatePacket object)

ทำหน้าที่สร้างแพ็คเกจแล้วทำการสั่งให้อ็อบเจ็ครับและส่งแพ็คเกจส่งแพ็คเกจนั้นออกสู่ระบบเครือข่าย โดยสามารถสร้างแพ็คเกจดังต่อไปนี้ได้

- สร้างแพ็คเกจที่ส่งเข้าไปในระบบเครือข่ายเพื่อทดสอบสมรรถนะเมื่อมีโหลดเป็นจำนวนต่าง ๆ กัน



รูปที่ 5-3 แสดงไดอะแกรมดาต้าโฟล์วของซอฟต์แวร์ที่พัฒนา

5.2 การพัฒนาซอฟต์แวร์(Software Development)

ซอฟต์แวร์ที่ทำการพัฒนาขึ้นมานี้สามารถรันได้บนแพล็ตฟอร์มดอส ในการพัฒนาได้ใช้ภาษา C++ ซึ่งสอดคล้องกับการออกแบบซอฟต์แวร์แบบอ็อบเจ็คโอเรียนเต็ด ซึ่งสามารถอิมพลีเมนท์ (implement) ได้ดังนี้

5.2.1 อ็อบเจ็คกรองแพ็คเกจ(packetFilter object)

มีเมทธอด(method) ดังต่อไปนี้

1. การกรองแพ็คเกจโดยใช้ไอพีแอดเดรส(method filterByIP)

สามารถกรองแพ็คเกจจากไอพีแอดเดรสได้ โดยต้องผ่านอาร์กิวเมนท์ให้กับเมทธอดนี้ 3 ตัวคือ

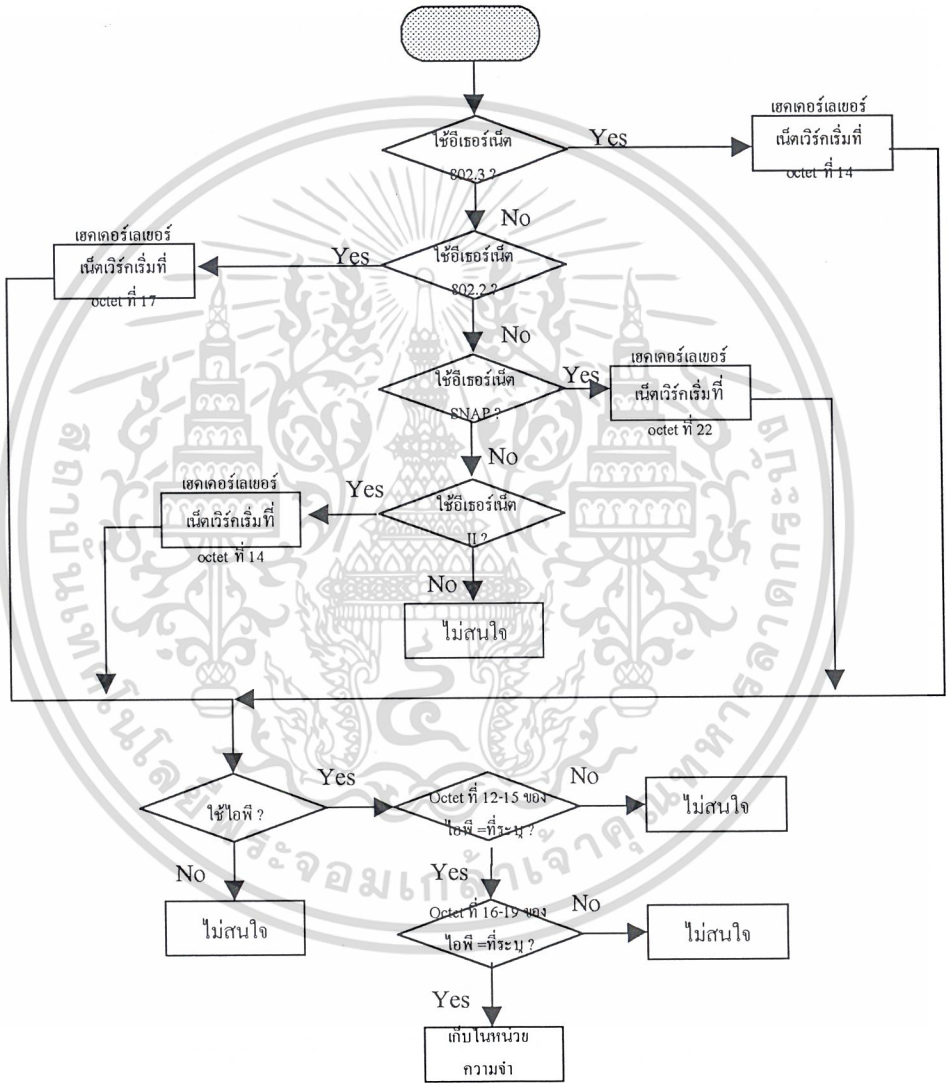
- แพ็คเกจที่จับมาได้
- ไอพีแอดเดรสของสเตชั่นต้นทาง
- ไอพีแอดเดรสของสเตชั่นปลายทาง

สามารถแสดง pseudo code ของเมทธอดได้ดังนี้

```
algorithm filterByIP   /* use both source IP and destination IP */
input:  current frame
        source IP
        destination IP
output: flag to determine whether current frame match
{
        if(this frame not IP)
                return 0;      /* this frame don't pass filtering */
        else {
```

```
                start point=start point of network layer header;
                compare source IP with current frame byte (start point+(12 to 15));
                compare source IP with current frame byte (start point+(16 to 19));
                if(address match both)
                        return 1;
                else
                        return 0;
        }
}
```

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-4



รูปที่ 5-4 แสดงโฟล์วชาร์ตของเมทธอด filterByIP

2. การกรองแพ็คเกจโดยใช้แมคแอดเดรส(method filterByMac)

สามารถกรองแพ็คเกจจากแมคแอดเดรสได้ โดยต้องผ่านอาร์กิวเมนท์ให้กับเมทธอดนี้ 3 ตัวคือ

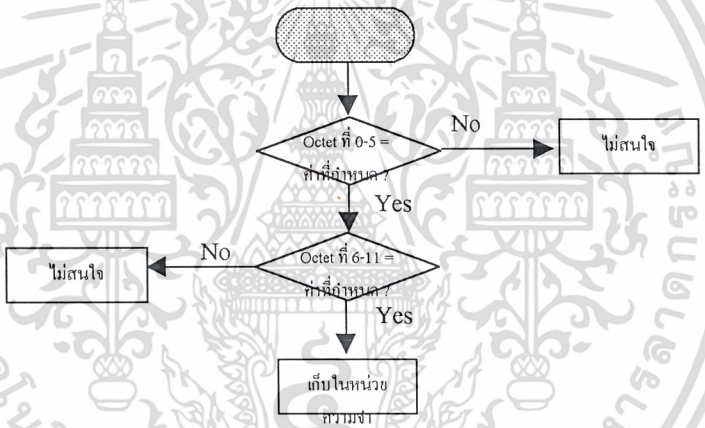
- แพ็คเกจที่จับมาได้
- แมคแอดเดรสของสเตชั่นต้นทาง

- แมคแอดเดรสของสเตชั่นปลายทาง

สามารถแสดง pseudo code ของเมทธอดได้ดังนี้

```
algorithm filterByMAC   /* use both source and destination MAC address */
input:  current frame
        source MAC
        destination MAC
output: flag to determine whether this frame is match
{
        compare source MAC with current frame byte 6 to 11;
        compare destination MAC with current frame byte 0 to 5;
        if(match both)
                return 1;
        else
                return 0;
}
```

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-5



รูปที่ 5-5 แสดงโฟล์วชาร์ตของเมทธอด filterByMac

3. การกรองแพ็คเกจโดยใช้โพรโตคอลชั้นเน็ตเวิร์ค(method filterByNetworkProtocol)

สามารถกรองแพ็คเกจจากโพรโตคอลที่ใช้ในเน็ตเวิร์คเลเยอร์ได้ โดยต้องผ่านอาร์กิวเมนท์ให้กับเมทธอดนี้ 2 ตัวคือ

- หมายเลขโพรโตคอลที่ต้องการกรองซึ่งกำหนดโดยผู้พัฒนาซอฟต์แวร์นี้
- แพ็คเกจที่จับมาได้

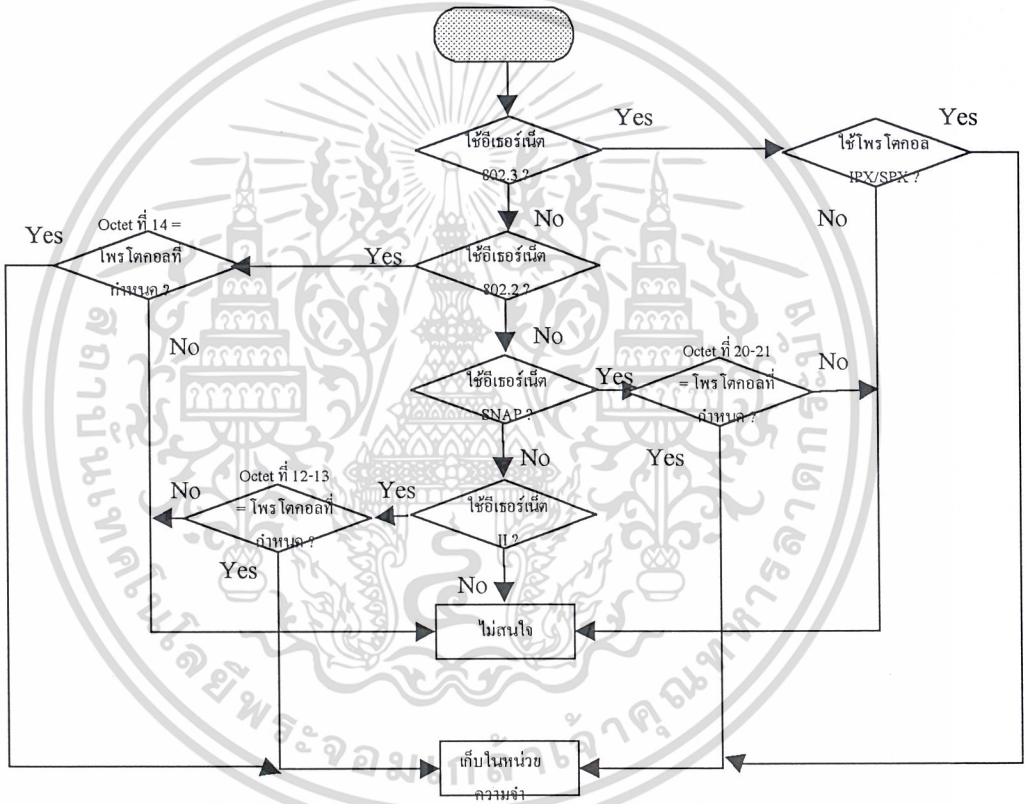
สามารถอธิบายเมทธอดนี้โดย pseudo code ได้ดังนี้

```
algorithm filterByNetworkProtocol   /* known protocol is IP,IPX,ARP,RARP */
input:  protocol id
        current frame
output: flag to determine whether this frame is match
{
        ethernet type=ethernet type of current frame;
        if(ethernet type=Ethernet 802.3)
```

```
            network protocol=ipx;
        else if(ethernet type=Ethernet 802.2)
            network protocol=current frame byte 14
        else if(ethernet type=Ethernet SNAP)
            network protocol=current frame byte 20 to 21
        else if(ethernet type=Ethernet II)
            network protocol=current frame byte 12 to 13

        if(network protocol==protocol id)
            return 1;
        else
            return 0;
}
```

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-6



รูปที่ 5-6 แสดงโฟล์วชาร์ตของเมทธอด filterByNetworkProtocol

4. การกรองแพ็คเกจโดยใช้โพรโตคอลชั้นดาต้าลิงค์(method filterByDtlProtocol)

สามารถกรองแพ็คเกจจากโพรโตคอลที่ใช้ในดาต้าลิงค์เลเยอร์ได้ โดยต้องผ่านอาร์กิวเมนท์ให้กับเมทธอดนี้ 2 ตัวคือ

- หมายเลขโพรโตคอลที่ต้องการกรองซึ่งกำหนดโดยผู้พัฒนาซอฟต์แวร์นี้
- แพ็คเกจที่จับมาได้

สามารถแสดง pseudo code ของเมทธอดได้ดังนี้


```
algorithm filterByDtlProtocol
input:  protocol id
```

```
        current frame
output: flag determine whether current frame is match
{
        determine value in current frame byte 12 to 13;
        if greater than 0x05DC
                it's Ethernet II;
        else if current frame byte 14 to 15 equal 0xFF
                it's Ethernet 802.3;
        else if current frame byte 14 equals 0xAA
                it's Ethernet SNAP;
        else it's Ethernet 802.2;
}
```

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 2-30 ในบทที่ 2

5. การกรองแพ็คเกจโดยใช้โพรโตคอลชั้นทรานสปอร์ต(method filterByTransportLayer)

สามารถกรองแพ็คเกจจากโพรโตคอลที่ใช้ในทรานสปอร์ตเลเยอร์ได้ โดยต้องผ่านอาร์กิวเมนท์ให้กับเมทธอดนี้ 2 ตัวคือ

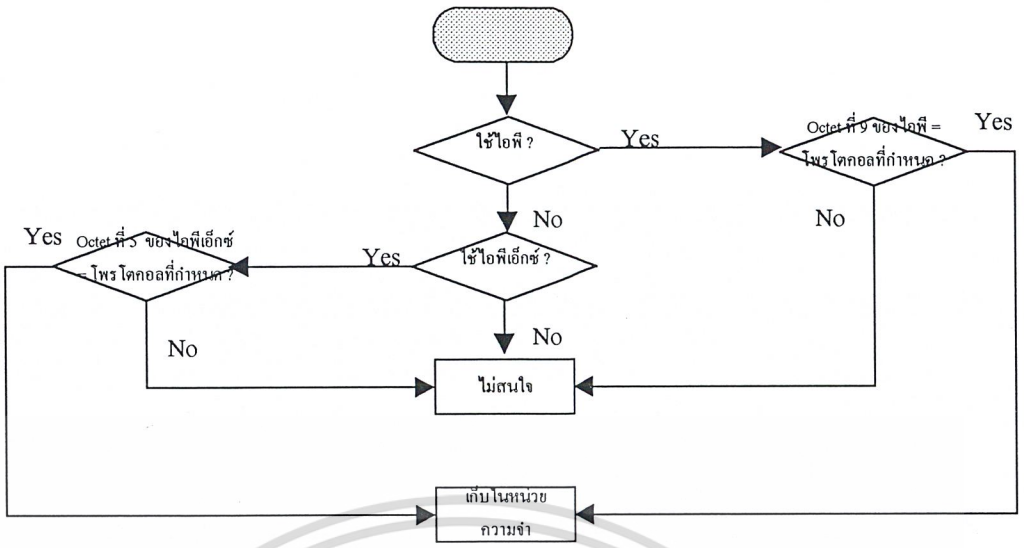
- หมายเลขโพรโตคอลที่ต้องการกรองซึ่งกำหนดโดยผู้พัฒนาซอฟต์แวร์นี้
- แพ็คเกจที่จับมาได้

สามารถแสดง pseudo code ของเมทธอดนี้ได้ดังนี้

```
algorithm filterByTransportLayer
input:  protocol id
        current frame
output: flag to determine whether current frame is match
{
        if(current frame is IP)
                transport protocol=current frame byte 9 from IP header start point;
        else if(current frame is IPX)
                transport protocol=current frame byte 5 from IPX header start point;

        if(protocol id==transport protocol)
                return 1;
        else
                return 0;
}
```

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-7



รูปที่ 5-7 แสดงโฟล์วชาร์ตของเมทธอด filterByTransportProtocol

6. การกรองแพ็คเกจโดยใช้ขนาดแพ็คเกจ(method filterBySize)

สามารถกรองแพ็คเกจจากขนาดแพ็คเกจที่กำหนดให้ได้ โดยต้องผ่านอาร์กิวเมนท์ให้กับเมทธอดนี้ 3 ตัวคือ

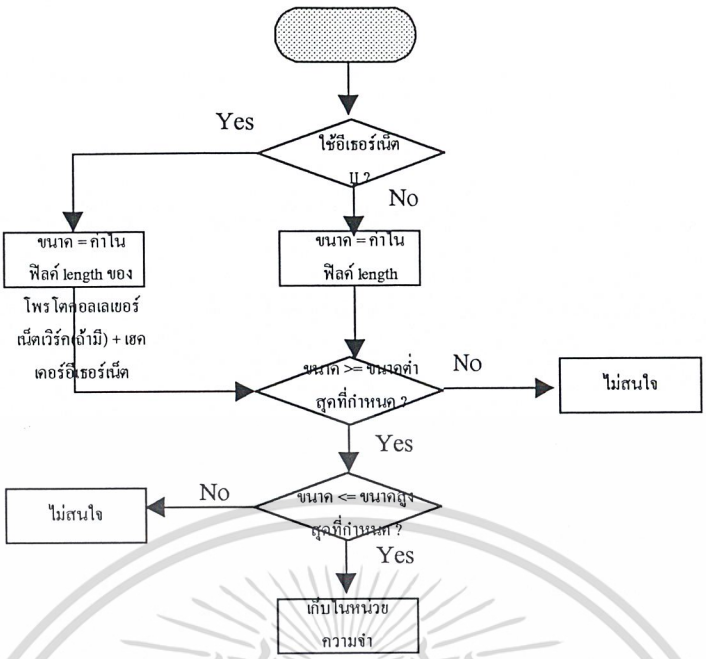
- แพ็คเกจที่จับมาได้
- ขนาดต่ำสุดของแพ็คเกจที่ต้องการกรอง
- ขนาดสูงสุดของแพ็คเกจที่ต้องการกรอง

สามารถแสดง pseudo code ของเมทธอดนี้ได้ดังนี้

```
algorithm filterBySize
input:  current frame
        minimum size
        maximum size
output: flag to determine whether current frame is match
{
        if(current frame is Ethernet II)
        {
                /* length of network PDU plus size of Ethernet II header(is 14) */
                length=current frame byte 2 to 3 from IP or IPX start point plus 14;
        } else length=current frame byte 12 to 13

        if(minimum size <= length <= maximum size)
                return 1;
        else
                return 0;
}
```

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-8



รูปที่ 5-8 แสดงโฟล์วชาร์ตของเมทธอด filterBySize

7. การกรองแพ็คเกจโดยใช้หมายเลขพอร์ต(method filterByPortNumber)

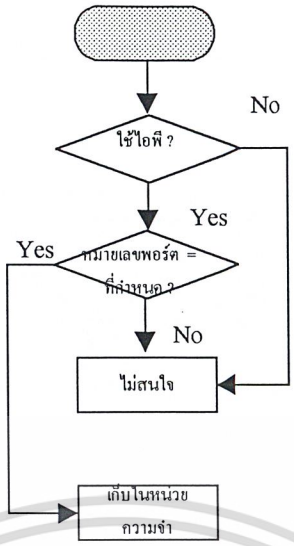
สามารถกรองแพ็คเกจจากขนาดแพ็คเกจที่กำหนดให้ได้ โดยต้องผ่านอาร์กิวเมนท์ให้กับเมทธอดนี้ 2 ตัวคือ

- หมายเลขพอร์ตที่ต้องการกรอง
- แพ็คเกจที่จับมาได้

สามารถอธิบาย pseudo code ของเมทธอดนี้ได้ดังนี้

```
algorithm filterByPortNumber
input:  Port number
        current frame
output: flag to determine whether current frame is match
{
        if(current frame not use IP)
                return 0;
        else {
                find start point of transport layer protocol:  /* TCP or UDP */
                /* byte 0-1 is source port, byte 2-3 is destination port */
                if(current frame byte 0 to 1 or 2 to 3 from start point of TCP or
UDP==Port number)
                        return 1;
                else
                        return 0;
        }
}
```

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-9



รูปที่ 5-9 แสดงโฟล์วชาร์ตของเมทธอด filterByPortNumber

5.2.2 อ็อบเจ็ควิเคราะห์แพ็คเกจ(packetAnalyzer object)

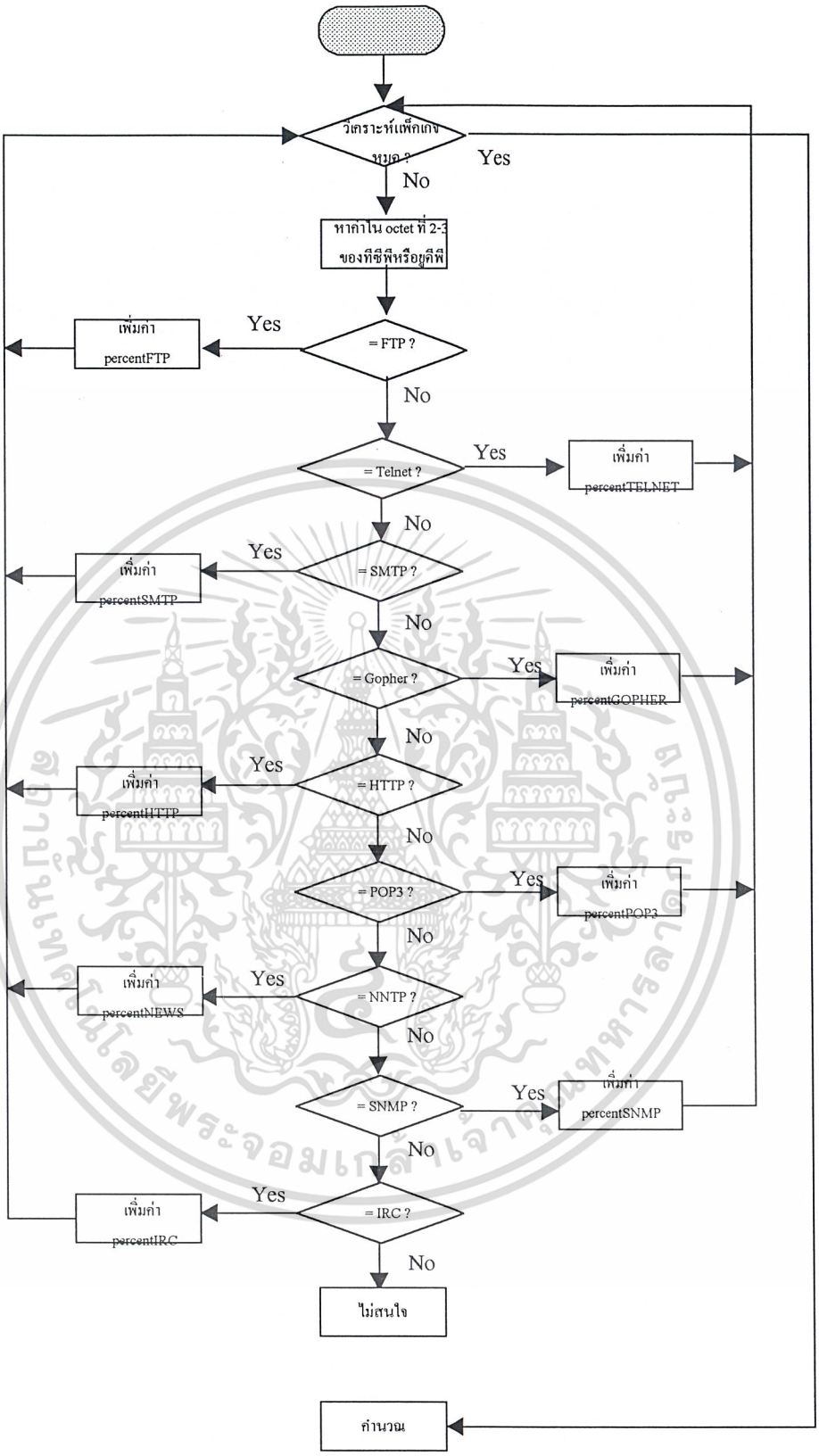
มีเมทธอดดังต่อไปนี้

1. การวิเคราะห์หาเปอร์เซ็นต์การใช้งานโพรโตคอลชั้นแอพพลิเคชั่น(method usedApp)

สามารถหาเปอร์เซ็นต์การใช้งานของโพรโตคอล 9 โพรโตคอลดังกล่าวแล้วได้จากแพ็คเกจที่จับมาได้ขณะนั้น ซึ่งจะต้องผ่านอาร์กิวเมนท์ให้กับเมทธอดนี้ 2 ตัวดังนี้

- จำนวนแพ็คเกจทั้งหมดที่จะนำมาวิเคราะห์
- พอยเตอร์ชี้ไปยังหน่วยความจำที่เก็บแพ็คเกจที่จับมาได้

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-10



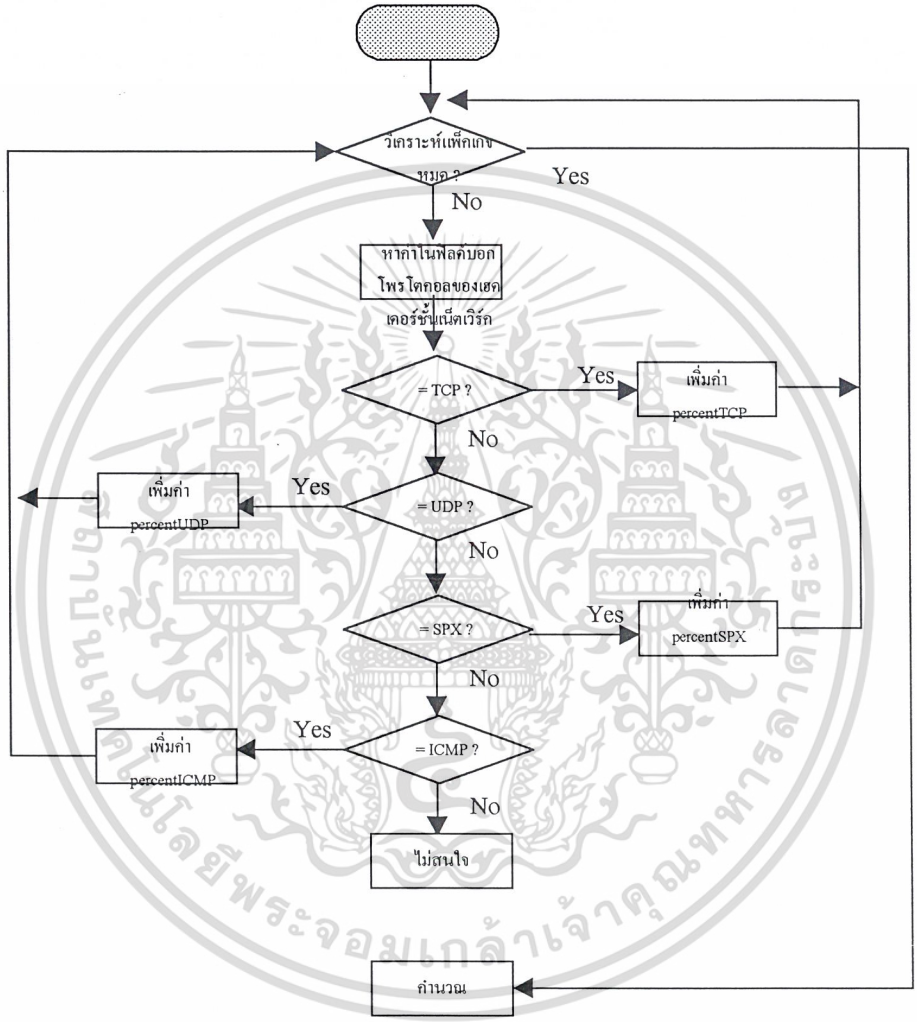
รูปที่ 5-10 แสดงโฟล์วชาร์ตของเมทธอด usedApp

2. การวิเคราะห์หาเปอร์เซ็นต์การใช้โพรโตคอลชั้นทรานสปอร์ต(method usedTransport)

สามารถวิเคราะห์หาเปอร์เซ็นต์การใช้งานของแต่ละโพรโตคอลในทรานสปอร์ตเลเยอร์ดังกล่าวมาแล้ว จากแพ็คเกจที่จับมาได้ขณะนั้น ซึ่งจะต้องผ่านอาร์กิวเมนท์ให้กับเมทธอดนี้ 2 ตัวดังนี้

- จำนวนแพ็คเกจทั้งหมดที่จะนำมาวิเคราะห์
- พอยเตอร์ชี้ไปยังหน่วยความจำที่เก็บแพ็คเกจที่จับมาได้

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-11



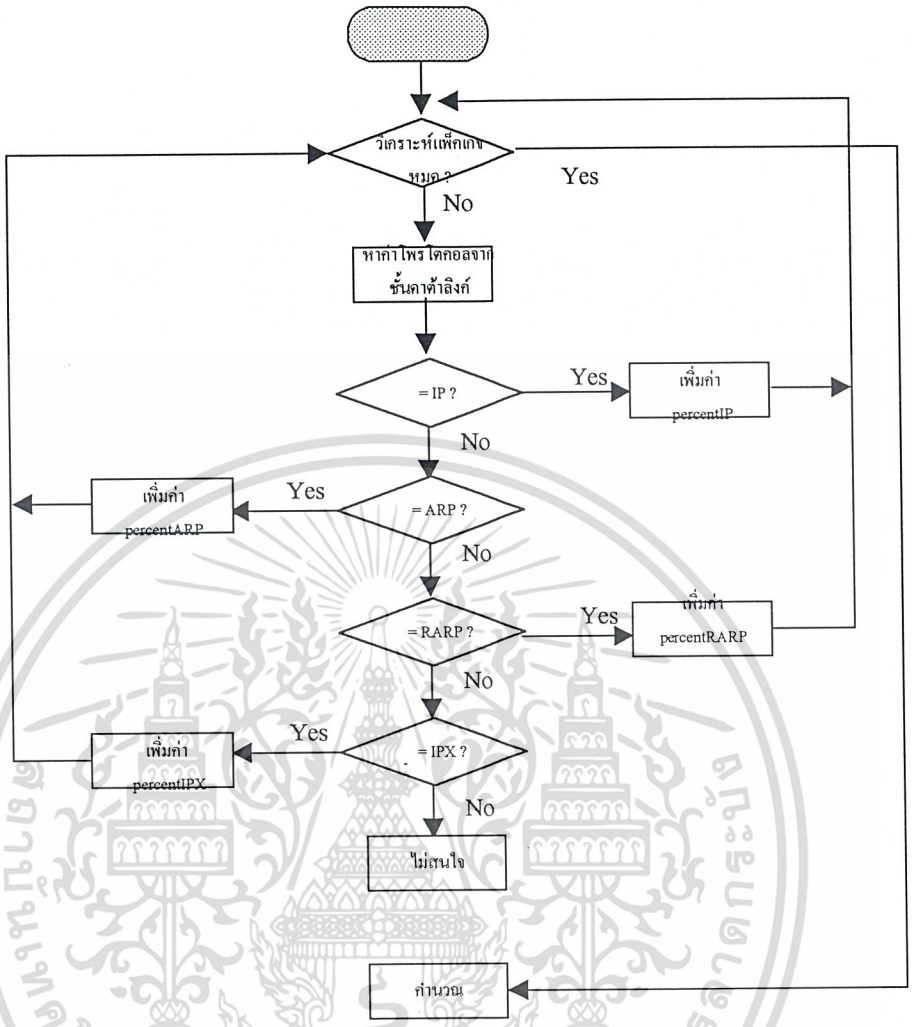
รูปที่ 5-11 แสดงโฟล์วชาร์ตของเมทธอด usedTransport

3. การวิเคราะห์หาเปอร์เซ็นต์การใช้โพรโตคอลชั้นเน็ตเวิร์ค(method usedNetwork)

สามารถวิเคราะห์หาเปอร์เซ็นต์การใช้งานของแต่ละโพรโตคอลในเน็ตเวิร์คเลเยอร์ดังกล่าวมาแล้ว จากแพ็คเกจที่จับมาได้ขณะนั้น ซึ่งจะต้องผ่านอาร์กิวเมนท์ให้กับเมทธอดนี้ 2 ตัวดังนี้

- จำนวนแพ็คเกจทั้งหมดที่จะนำมาวิเคราะห์
- พอยเตอร์ชี้ไปยังหน่วยความจำที่เก็บแพ็คเกจที่จับมาได้

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-12



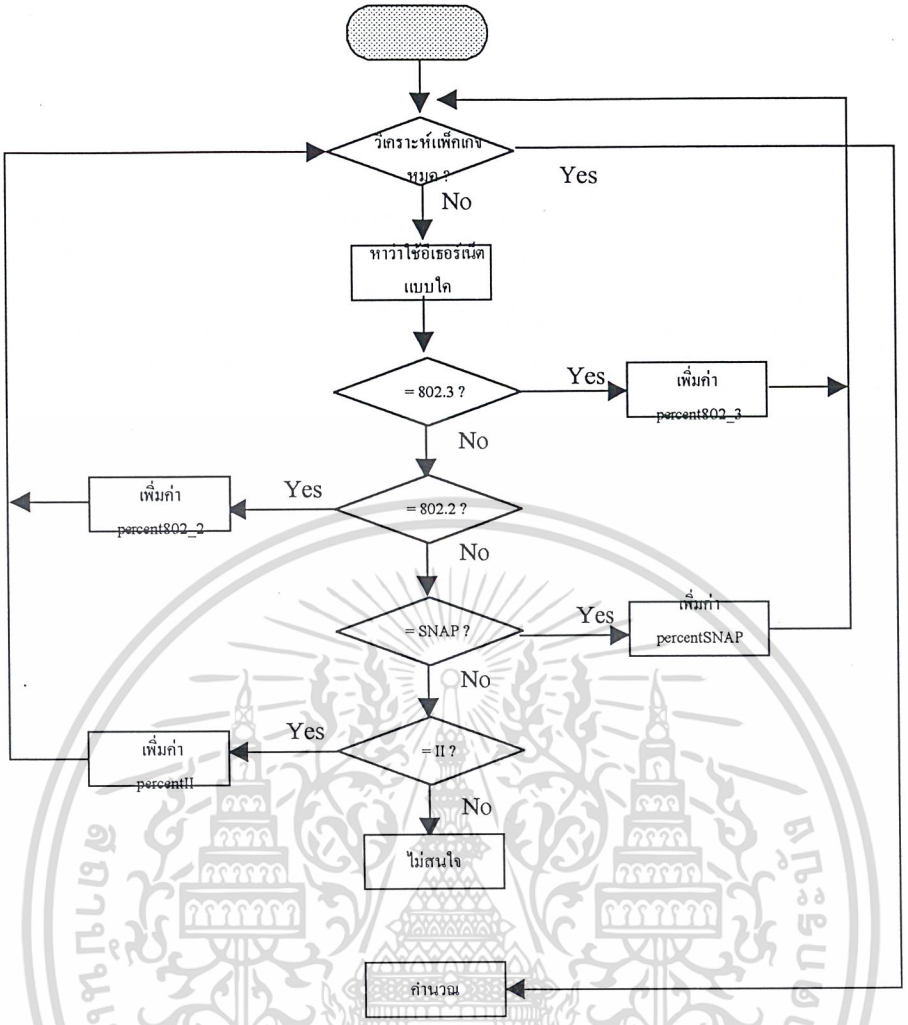
รูปที่ 5-12 แสดงโฟล์วชาร์ตของเมทธอด usedNetwork

4. การวิเคราะห์หาเปอร์เซ็นต์การใช้งานอีเธอร์เน็ตแบบต่าง ๆ (method usedEthernet)

สามารถวิเคราะห์หาเปอร์เซ็นต์การใช้งานของแต่ละโพรโตคอลในดาต้าลิงค์เลเยอร์ดังกล่าวมาแล้ว จากแพ็คเกจที่จับมาได้ขณะนั้น ซึ่งจะต้องผ่านอาร์กิวเมนท์ให้กับเมทธอดนี้ 2 ตัวดังนี้

- จำนวนแพ็คเกจทั้งหมดที่จะนำมาวิเคราะห์
- พอยเตอร์ชี้ไปยังหน่วยความจำที่เก็บแพ็คเกจที่จับมาได้

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-13



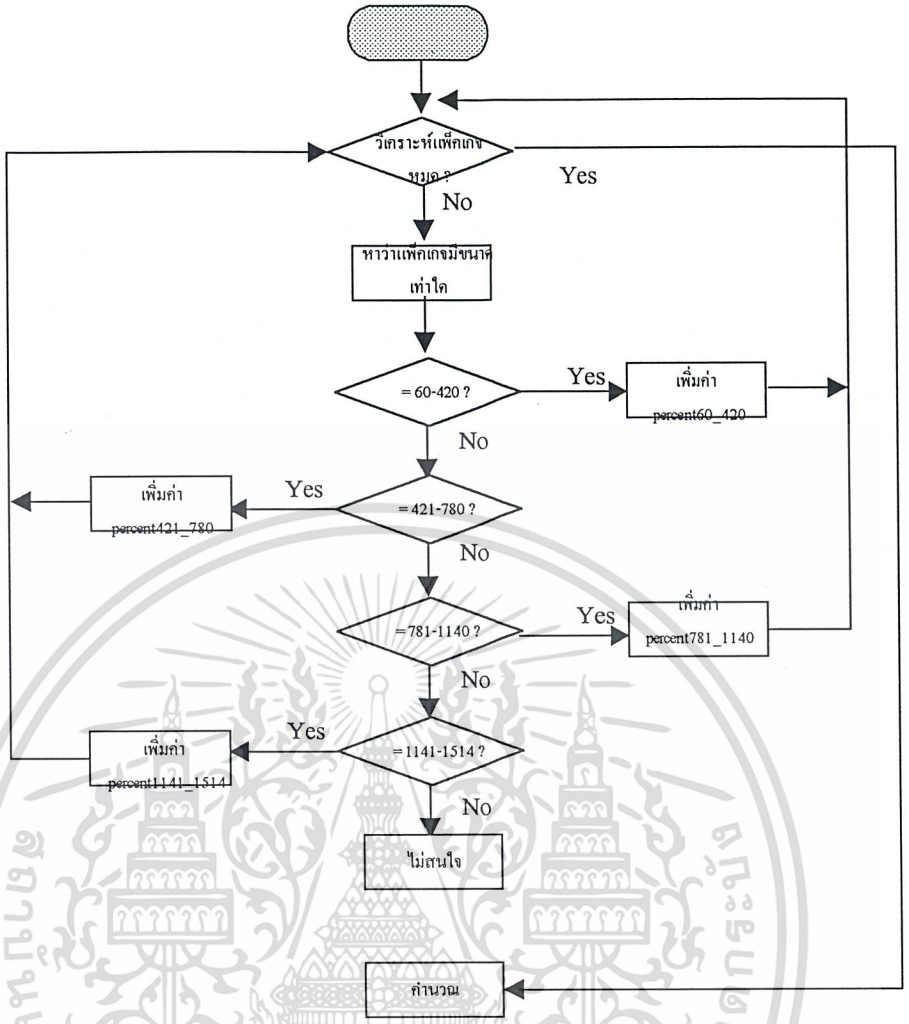
รูปที่ 5-13 แสดงโฟล์วชาร์ตของเมทธอด usedEthernet

5. การวิเคราะห์การกระจายของขนาดแพ็คเกจ(method distributedSize)

สามารถวิเคราะห์การกระจายของขนาดแพ็คเกจที่จับมาได้ขณะนั้น โดยต้องผ่านอาร์กิวเมนท์ 2 ตัวให้เมทธอดนี้คือ

- จำนวนแพ็คเกจทั้งหมดที่จะนำมาวิเคราะห์
- พอยเตอร์ชี้ไปยังหน่วยความจำที่เก็บแพ็คเกจที่จับมาได้

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-14



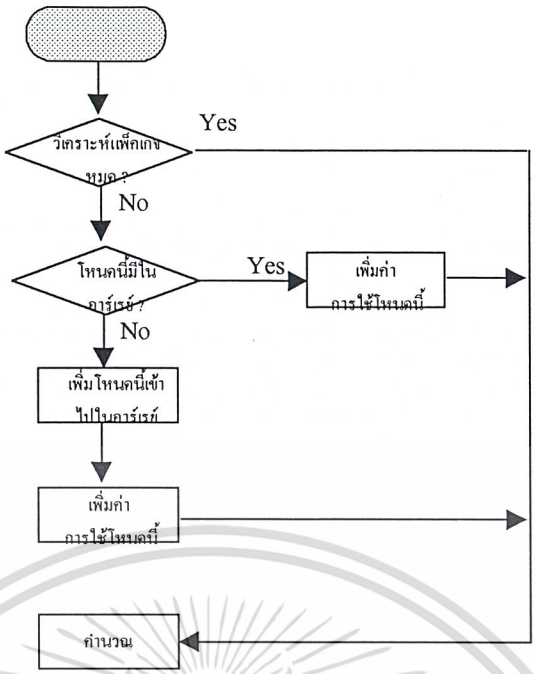
รูปที่ 5-14 แสดงโฟล์วชาร์ตของเมทธอด distributedSize

6. การวิเคราะห์หาโหนดที่ส่งแพ็คเกจออกมามากที่สุด(method frequentSendingNode)

สามารถหาโหนดที่ใช้งานแลนมากที่สุดในจำนวนทั้งหมดที่จับมาได้ขณะนั้น โดยต้องผ่านอาร์กิวเมนท์จำนวน 2 ตัวให้กับเมทธอดนี้คือ

- จำนวนแพ็คเกจทั้งหมดที่จะนำมาวิเคราะห์
- พอยเตอร์ชี้ไปยังหน่วยความจำที่เก็บแพ็คเกจที่จับมาได้

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-15



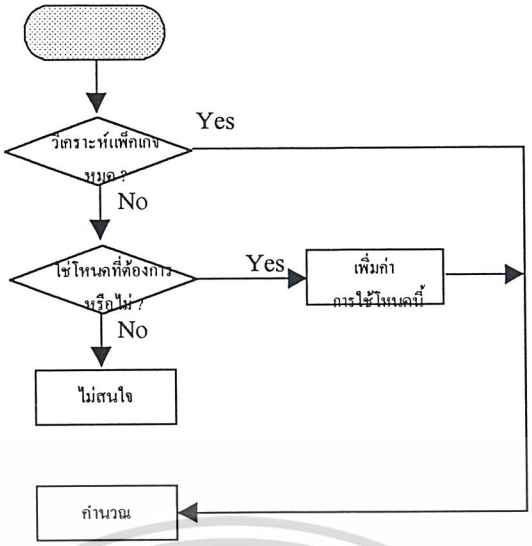
รูปที่ 5-15 แสดงโฟล์วชาร์ตของเมทธอด frequentUsedNode

7. การวิเคราะห์หาเปอร์เซ็นต์การใช้งานของโหนดที่กำหนด(method percentUsedNode)

สามารถหาเปอร์เซ็นต์การใช้งานโหนดที่กำหนด หรือระหว่างโหนดที่กำหนดจากแพ็คเกจทั้งหมดที่จับมาได้ขณะนั้น โดยต้องผ่านอาร์กิวเมนท์จำนวน 4 ตัวให้กับเมทธอดนี้คือ

- จำนวนแพ็คเกจทั้งหมดที่จะนำมาวิเคราะห์
- พอยเตอร์ชี้ไปยังหน่วยความจำที่เก็บแพ็คเกจที่จับมาได้
- แมคแอดเดรสของโหนดต้นทาง
- แมคแอดเดรสของโหนดปลายทาง

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-16



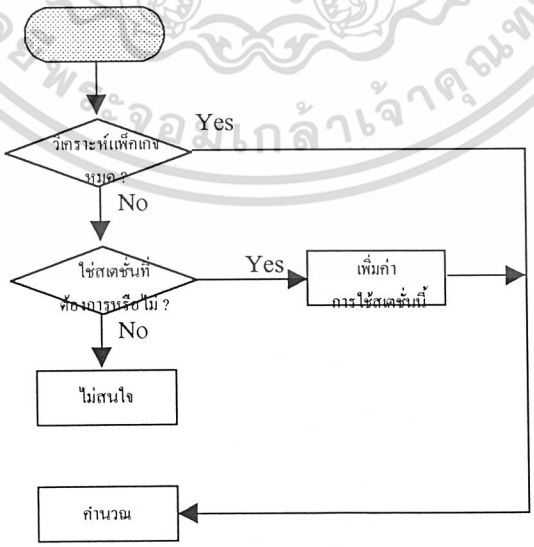
รูปที่ 5-16 แสดงโฟล์วชาร์ตของเมทธอด percentUsedNode

8. การวิเคราะห์หาเปอร์เซ็นต์การใช้งานสเตชั่นที่กำหนด(method percentUsedPacket)

สามารถหาเปอร์เซ็นต์การใช้งานสเตชั่นที่กำหนด หรือระหว่างสเตชั่นที่กำหนดจากแพ็คเกจทั้งหมดที่จับมาได้ขณะนั้น โดยต้องผ่านอาร์กิวเมนท์จำนวน 4 ตัวให้กับเมทธอดนี้คือ

- จำนวนแพ็คเกจทั้งหมดที่จะนำมาวิเคราะห์
- พอยเตอร์ชี้ไปยังหน่วยความจำที่เก็บแพ็คเกจที่จับมาได้
- ไอพีแอดเดรสของโหนดต้นทาง
- ไอพีแอดเดรสของโหนดปลายทาง

การทำงานของเมทธอดนี้สามารถแสดงโดยโฟล์วชาร์ตดังรูปที่ 5-17



รูปที่ 5-17 แสดงโฟล์วชาร์ตของเมทธอด percentUsedPacket

5.2.3 อ็อบเจ็คจับและส่งแพ็คเกจ(PktDriver object)

เป็นอ็อบเจ็คที่ใช้สำหรับจับและส่งเฟรมออกไปในระบบเครือข่ายคอมพิวเตอร์ โดยมีการอินเทอร์เฟสกับแพ็คเกจไดรเวอร์ NE2000.EXE อ็อบเจ็คนี้เขียนขึ้นตามข้อกำหนดที่ได้กล่าวถึงแล้วในบทที่ 4 แต่ปรับปรุงเล็กน้อยเพื่อให้สะดวกในการเรียกใช้มากขึ้น อ็อบเจ็คนี้มีเมทธอดดังต่อไปนี้

1. เมทธอด InitPktDriver

ใช้ในการกำหนดค่าเริ่มต้น(initial) ไลบรารีฟังก์ชัน PKT โดยมีการเรียกใช้เมทธอด ScanForPktDriver และเมทธอด ShowInfo รูปแบบการเรียกใช้งานเป็นดังนี้

```
InitPktDriver();
```

2. เมทธอด AccessType

ใช้ในการเข้าถึงแพ็คเกจไดรเวอร์ ทำให้สามารถรับแพ็คเกจตามชนิดที่ต้องการได้ เมทธอดนี้มีรูปแบบการเรียกใช้งานดังนี้

```
int AccessType(int if_type,int if_number,char *type,unsigned typelen,int interrupt
receiver());
```

อากิวเมนท์ที่ผ่านให้เมทธอดนี้ประกอบด้วย ชนิดการอินเทอร์เฟส, หมายเลขการอินเทอร์เฟส, พอยเตอร์ที่ชี้ไปยังชนิดของแพ็คเกจ, ความยาวเป็นจำนวนไบต์, และพอยเตอร์ที่ชี้ไปยังฟังก์ชันในการรับแพ็คเกจ ตามลำดับ และหลังจากเรียกใช้งานเมทธอดนี้แล้วจะให้ค่าแฮนเดิลกลับมา

3. เมทธอด ReleaseType

ใช้ในการเลิกใช้แฮนเดิล และหยุดรับแพ็คเกจจากดีไวซ์ไดรเวอร์ มีรูปแบบการเรียกใช้งานดังนี้

```
int ReleaseType(int handle);
```

อากิวเมนท์ที่ผ่านให้เมทธอดนี้คือ ค่าแฮนเดิลซึ่งได้มาจากการเรียกใช้ฟังก์ชัน AccessType นั่นเอง ส่วนค่าที่ได้กลับมาหลังจากเรียกใช้เมทธอดนี้ก็คือค่าที่แสดงว่าทำงานสำเร็จหรือไม่

4. เมทธอด SendPacket

ใช้ในการเข้าถึงแพ็คเกจไดรเวอร์เพื่อทำการส่งแพ็คเกจซึ่งมีขนาดตามที่กำหนดออกไป ซึ่งแพ็คเกจดังกล่าวจะต้องมีการใส่เฮดเดอร์รวมไว้กับข้อมูลจริง ๆ แล้ว มีรูปแบบการเรียกใช้งานดังนี้

```
int SendPacket(char far *buffer,unsigned length);
```

อากิวเมนท์ที่ผ่านให้ประกอบด้วย พอยเตอร์ชี้ไปยังข้อมูลที่ต้องการส่ง และขนาดของแพ็คเกจที่ต้องการส่ง ตามลำดับ หลังจากเรียกใช้แล้วจะส่งค่ากลับเพื่อระบุว่าทำงานสำเร็จหรือไม่

5. เมทธอด TerminateDriver

ใช้ในการจบการทำงานของดีไวซ์ไดรเวอร์ที่สอดคล้องกับแฮนเดิลที่กำหนด ไดรเวอร์จะหยุดทำงานและจะคืนหน่วยความจำให้กับดอสถ้าเป็นไปได้ มีรูปแบบการเรียกใช้งานดังนี้

```
int TerminateDriver(int handle);
```

อากิวเมนท์ที่ส่งให้คือ แฮนเดิล และเมื่อเรียกใช้เมทธอดนี้แล้วจะให้ค่ากลับเพื่อระบุว่าทำงานสำเร็จหรือไม่

6. เมทธอด GetAddress

ใช้ในการหาแมคแอดเดรสของเครื่องที่กำลังรันเมทธอดนี้อยู่ หลังจากเรียกใช้จะให้ค่ากลับเพื่อระบุว่าทำงานสำเร็จหรือไม่ มีรูปแบบการเรียกใช้งานดังนี้

```
int GetAddress();
```

7. เมทธอด ResetInterface

ใช้ในการรีเซ็ทดีไวซ์ไดรเวอร์ หยุดการส่งทุกอย่างที่กำลังดำเนินอยู่ และทำการกำหนดค่าเริ่มต้นให้อีกครั้ง มีรูปแบบการเรียกใช้งานดังนี้

```
int ResetInterface(int handle);
```

อากิวเมนท์ที่ต้องผ่านให้เมทธอดนี้คือ แฮนเดิล ค่าที่ส่งกลับจากเมทธอดนี้เป็นค่าที่บ่งถึงว่าทำงานสำเร็จหรือไม่

8. เมทธอด GetParameters

ใช้ในการหาค่าพารามิเตอร์จากไดรเวอร์ มีรูปแบบการเรียกใช้งานดังนี้

```
void GetParameters();
```

9. เมทธอด SetRCVmode

ใช้ในการเลือกโหมดในการรับแพ็คเกจ มีรูปแบบในการเรียกใช้งานดังนี้

```
int SetRCVmode(int handle,int mode);
```

อากิวเมนท์ที่ต้องผ่านไปให้คือ แฮนเดิล และโหมดในการรับแพ็คเกจที่ต้องการ เมทธอดนี้จะส่งค่ากลับมาเพื่อระบุว่าทำงานสำเร็จหรือไม่

10. เมทธอด GetRCVmode

ใช้ในการหาค่าโหมดที่ใช้ในการรับแพ็คเกจอยู่ขณะนั้น มีรูปแบบการเรียกใช้งานดังนี้

```
int GetRCVmode(int handle);
```

อากิวเมนท์ที่ต้องส่งให้คือ แฮนเดิล และหลังจากเมทธอดทำงานเสร็จแล้วจะส่งค่ากลับมาเป็นโหมดที่ใช้ขณะนั้น ถ้าไม่สำเร็จจะส่งรหัสแสดงข้อผิดพลาดกลับมา

11. เมทธอด GetHandle

ใช้ในการหาแฮนเดิลที่กำลังใช้งานอยู่ เมื่อเมทธอดทำงานเสร็จจะส่งค่าแฮนเดิลกลับมาให้ โดยมีรูปแบบการเรียกใช้งานดังนี้

```
int GetHandle();
```

5.2.4 อ็อบเจ็คสร้างแพ็คเกจ

สร้างแพ็คเกจที่ส่งเข้าไปในระบบเครือข่ายเพื่อทดสอบสมรรถนะเมื่อมีโหลดเป็นจำนวนต่าง ๆ กันตั้งแต่ 1% - 25% ตัวอย่างเช่น

10% load	10 Kbytes packet every 1 seconds
20% load	20 Kbytes packet every 1 seconds
30% load	30 Kbytes packet every 1 seconds
40% load	40 Kbytes packet every 1 seconds
50% load	50 Kbytes packet every 1 seconds
60% load	60 Kbytes packet every 1 seconds

บทที่ 6

การทดสอบแต่ละระบบย่อย(Sub-system testing)

ในการพัฒนาซอฟต์แวร์นั้นมีหลายขั้นตอน ซึ่งการทดสอบความถูกต้องของซอฟต์แวร์ก็เป็นหนึ่งในหลายขั้นตอนของการพัฒนาซอฟต์แวร์ซึ่งเป็นขั้นตอนที่นับได้ว่ามีความสำคัญมาก เพราะจะทำให้เรามั่นใจได้ว่าซอฟต์แวร์ที่เราพัฒนาขึ้นมานั้นสามารถนำไปใช้งานได้อย่างถูกต้อง อีกทั้งในขั้นตอนการทดสอบความถูกต้องของซอฟต์แวร์นี้จะทำให้เราค้นพบข้อผิดพลาดที่ยังไม่ปรากฏออกมาในซอฟต์แวร์ของเรา ซึ่งทำให้เราแก้ไขได้ทันก่อนที่จะมีการนำไปใช้งานจริง

ในการทดสอบซอฟต์แวร์ของโครงงานนี้จะใช้การทดสอบที่เรียกว่าการทดสอบกล่องดำ(Black-box testing) คือการป้อนอินพุทเข้าไปในระบบ จากนั้นเราจะดูเอาท์พุทที่ออกมาว่าเป็นไปตามที่คาดหมายไว้หรือไม่

6.1 การทดสอบอ็อบเจ็ค packetFilter

อ็อบเจ็คนี้เป็นการนำเฟรมที่จับได้จากแลนมากรองตามลักษณะที่ต้องการดังได้กล่าวไว้แล้วในบทที่ 4 ดังนั้นเราจะทดสอบอ็อบเจ็คนี้โดยการจำลองเฟรมทุกชนิดที่อ็อบเจ็คนี้สามารถกรองได้เพื่อมาทดสอบว่าอ็อบเจ็คนี้ทำงานถูกต้องหรือไม่ ซึ่งเฟรมที่จำลองขึ้นมามีทั้งหมด 7 เฟรมและมีลักษณะดังต่อไปนี้

ชื่อ	Datalink	Network	Transport	Size
tcpFrame	Ethernet II Source: 0x1a-2b-3c-4d-5e-6f Destination: 0xff-ff-ff-ff-ff-ff	IP Source IP: 161.246.6.92 Destination IP: 161.246.4.3	TCP Source port: 0x10-00 Destination port: 80	71 bytes
udpFrame	Ethernet 802.2 Source: 0x1a-2b-3c-4d-5e-6f Destination: 0xff-ff-ff-ff-ff-ff	IP Source IP: 161.246.6.85 Destination IP: 161.246.10.21	UDP Source port: 0x10-00 Destination port: 23	68 bytes
arpFrame	Ethernet II Source: 0x1a-2b-3c-4d-5e-6f Destination:	ARP Operation: 0x00-01 (ARP request)		42 bytes

	0xff-ff-ff-ff-ff-ff			
rarpFrame	Ethernet II Source: 0x1a-2b-3c-4d-5e-6f Destination: 0xff-ff-ff-ff-ff-ff	RARP Operation: 0x00-04 (RARP response)		42 bytes
icmpFrame	Ethernet II Source: 0x1a-2b-3c-4d-5e-6f Destination: 0xff-ff-ff-ff-ff-ff	IP Source IP: 161.246.6.92 Destination IP: 161.246.4.3	ICMP Type: 0x00 (Echo request)	54 bytes
ipxFrame	Ethernet SNAP Source: 0x1a-2b-3c-4d-5e-6f Destination: 0xff-ff-ff-ff-ff-ff	IPX	IPX	93 bytes
spxFrame	Ethernet 802.3 Source: 0x1a-2b-3c-4d-5e-6f Destination: 0xff-ff-ff-ff-ff-ff	IPX	SPX	67 bytes

ตารางที่ 6-1 แสดงเฟรมที่จำลองขึ้นมาทดสอบ

เมื่อเราได้เฟรมครบทุกประเภทซึ่งสามารถจะนำมาทดสอบโปรแกรมแล้ว ก็จะทำการทดสอบโดยแยกเป็นแต่ละเมทธอดดังนี้

6.1.1 เมทธอด filterByIP

แบบ 1 แอดเดรส คือป้อนอินพุทเพียงไอพีแอดเดรสเดียวแล้วกรองทุกเฟรมที่มีไอพีแอดเดรสต้นทางหรือปลายทางตรงกับไอพีแอดเดรสที่กำหนด จะมีกรณีให้ทดสอบ 3 กรณีคือ

- ไอพีแอดเดรสที่กำหนดเท่ากับไอพีแอดเดรสต้นทาง
- ไอพีแอดเดรสที่กำหนดเท่ากับไอพีแอดเดรสปลายทาง

- ไม่ตรงกับไอพีแอดเดรสทั้งต้นทางและปลายทาง

ผลจากการรันโปรแกรมทดสอบเป็นดังนี้

```
*** filterByIP 1 address ***
*   tcpFrame : sAddr 161.246.6.92
*            : dAddr 161.246.4.3

Time 1 : need address 161.246.6.92
(expect true) : 1
Time 2 : need address 161.246.4.3
(expect true) : 1
Time 3 : need address 161.246.6.93
(expect false) : 0
```

แบบ 2 แอดเดรส คือป้อนอินพุททั้งไอพีแอดเดรสต้นทางและปลายทางแล้วกรองเฟรมที่มีแอดเดรสตรงกัน โดยต้นทางตรงกับไอพีแอดเดรสต้นทางที่ระบุ และปลายทางตรงกับไอพีแอดเดรสปลายทางที่ระบุ จะมีการทดสอบ 4 กรณีคือ

- ไอพีแอดเดรสต้นทางไม่ตรงกับที่ระบุ แต่ไอพีแอดเดรสปลายทางตรง
- ไอพีแอดเดรสตรงกับที่ระบุทั้งต้นทางและปลายทาง
- ไอพีแอดเดรส ไม่ตรงกับที่ระบุเลยทั้งต้นทางและปลายทาง
- ไอพีแอดเดรสต้นทางตรงกับที่ระบุแต่ไอพีแอดเดรสปลายทางไม่ตรง

ผลการรันโปรแกรมทดสอบเป็นดังนี้

```
*** filterByIP 2 address ***
*   tcpFrame : sAddr 161.246.6.92
*            : dAddr 161.246.4.3

Time 1 : need 161.246.6.93 and 161.246.4.3
(expect false) : 0
Time 2 : need 161.246.6.92 and 161.246.4.3
(expect true) : 1

*** filterByIP 2 address ***
*   udpFrame : sAddr 161.246.6.85
*            : dAddr 161.246.10.21

Time 1 : need 161.246.6.93 and 161.246.4.3
(expect false) : 0
Time 2 : need 161.246.6.85 and 161.246.4.3
(expect false) : 0
Time 3 : need 161.246.6.85 and 161.246.10.21
(expect true) : 1
```

6.1.2 เมทธอด filterByMAC

แบบ 1 แอดเดรส คือ การกรองโดยใส่อินพุทเข้าไปเพียงแมคแอดเดรสเดียว แล้วกรองเฉพาะเฟรมที่มีแมคแอดเดรสต้นทางหรือปลายทางตรงกับแมคแอดเดรสที่ระบุไว้ มีการทดสอบ 3 กรณีคือ

- แมคแอดเดรสที่ระบุไม่ตรงกับทั้งแมคแอดเดรสต้นทางและปลายทาง
- แมคแอดเดรสที่ระบุตรงกับแมคแอดเดรสปลายทาง
- แมคแอดเดรสที่ระบุตรงกับแมคแอดเดรสต้นทาง

ผลของการรันโปรแกรมทดสอบเป็นดังนี้

```
*** filterByMAC 1 address ***
*   arpFrame : MAC sAddr 0x1a 2b 3c 4d 5e 6f
*              MAC dAddr 0xff ff ff ff ff ff

Time 1 : need 0x11 22 33 44 55 66
(expect false) : 0
Time 2 : need 0xff ff ff ff ff ff
(expect true) : 1
Time 3 : need 0x1a 2b 3c 4d 5e 6f
(expect true) : 1
```

แบบ 2 แอดเดรส คือ การใส่อินพุทที่เป็นแมคแอดเดรสเข้าไปทั้งแมคแอดเดรสต้นทางและปลายทาง แล้วทำการกรองเฟรมที่มีแมคแอดเดรสต้นทางตรงกับแมคแอดเดรสต้นทางที่ระบุ และแมคแอดเดรสปลายทางตรงกับแมคแอดเดรสปลายทางที่ระบุไว้ สามารถทดสอบได้ 4 กรณีคือ

- แมคแอดเดรสตรงกับที่ระบุทั้งต้นทางและปลายทาง
- แมคแอดเดรสตรงเฉพาะต้นทาง
- แมคแอดเดรสไม่ตรงทั้งต้นทางและปลายทาง
- แมคแอดเดรสตรงกับที่ระบุเฉพาะปลายทาง

ผลการรันโปรแกรมทดสอบเป็นดังนี้

```
*** filterByMAC 2 address ***
*   arpFrame : MAC sAddr 0x1a 2b 3c 4d 5e 6f
*              MAC dAddr 0xff ff ff ff ff ff

Time 1 : need 0x1a 2b 3c 4d 5e 6f and 0xff ff ff ff ff ff
(expect true) : 1
Time 2 : need 0x1a 2b 3c 4d 5e 6f and 0xff 11 aa 99 22 fe
(expect false) : 0
Time 3 : need 0x1a 2b 3c 4d 5e 7f and 0xff 11 aa 99 22 fe
(expect false) : 0
Time 4 : need 0x1a 2b 3c 4d 5e 7f and 0xff ff ff ff ff ff
(expect false) : 0
```

6.1.3 เมทธอด filterByDtlProtocol

คือการกรองโดยใช้โพรโตคอลในชั้นดาต้าลิงค์เป็นตัวแยก ทดสอบได้ 16 กรณีคือ

- นำเฟรมที่ใช้ Ethernet II มากรองโดย Ethernet II
- นำเฟรมที่ใช้ Ethernet 802.2 มากรองโดย Ethernet II
- นำเฟรมที่ใช้ Ethernet SNAP มากรองโดย Ethernet II
- นำเฟรมที่ใช้ Ethernet 802.3 มากรองโดย Ethernet II
- นำเฟรมที่ใช้ Ethernet II มากรองโดย Ethernet 802.2
- นำเฟรมที่ใช้ Ethernet 802.2 มากรองโดย Ethernet 802.2
- นำเฟรมที่ใช้ Ethernet SNAP มากรองโดย Ethernet 802.2
- นำเฟรมที่ใช้ Ethernet 802.3 มากรองโดย Ethernet 802.2
- นำเฟรมที่ใช้ Ethernet II มากรองโดย Ethernet SNAP
- นำเฟรมที่ใช้ Ethernet 802.2 มากรองโดย Ethernet SNAP
- นำเฟรมที่ใช้ Ethernet SNAP มากรองโดย Ethernet SNAP
- นำเฟรมที่ใช้ Ethernet 802.3 มากรองโดย Ethernet SNAP
- นำเฟรมที่ใช้ Ethernet II มากรองโดย Ethernet 802.3
- นำเฟรมที่ใช้ Ethernet 802.2 มากรองโดย Ethernet 802.3
- นำเฟรมที่ใช้ Ethernet SNAP มากรองโดย Ethernet 802.3
- นำเฟรมที่ใช้ Ethernet 802.3 มากรองโดย Ethernet 802.3

ผลของการรันโปรแกรมทดสอบเป็นดังนี้

```
*** filterByDtlProtocol ***
*   tcpFrame : EthII
*   udpFrame : Eth802.2
*   ipxFrame : EthSNAP
*   spxFrame : Eth802.3

Time 1 : filter tcpFrame, need Ethernet II
(expect true) : 1
Time 2 : filter udpFrame, need Ethernet II
(expect false) : 0
Time 3 : filter ipxFrame, need Ethernet II
(expect false) : 0
Time 4 : filter spxFrame, need Ethernet II
(expect false) : 0
Time 5 : filter tcpFrame, need Ethernet 802.2
(expect false) : 0
Time 6 : filter udpFrame, need Ethernet 802.2
(expect true) : 1
Time 7 : filter ipxFrame, need Ethernet 802.2
(expect false) : 0
Time 8 : filter spxFrame, need Ethernet 802.2
(expect false) : 0
```

```
Time 9 : filter tcpFrame, need Ethernet SNAP
(expect false) : 0
Time 10 : filter udpFrame, need Ethernet SNAP
(expect false) : 0
Time 11 : filter ipxFrame, need Ethernet SNAP
(expect true) : 1
Time 12 : filter spxFrame, need Ethernet SNAP
(expect false) : 0
Time 13 : filter tcpFrame, need Ethernet 802.3
(expect false) : 0
Time 14 : filter udpFrame, need Ethernet 802.3
(expect false) : 0
Time 15 : filter ipxFrame, need Ethernet 802.3
(expect false) : 0
Time 16 : filter spxFrame, need Ethernet 802.3
(expect true) : 1
```

6.1.4 เมทธอด filterByNetworkProtocol

คือ การกรองเฟรมโดยใช้โพรโตคอลในชั้นเน็ตเวิร์คเป็นตัวแยก ซึ่งสามารถจะทดสอบได้ 16 กรณีคือ

- นำเฟรมที่ใช้ IP มากรองโดย IP
- นำเฟรมที่ใช้ ARP มากรองโดย IP
- นำเฟรมที่ใช้ RARP มากรองโดย IP
- นำเฟรมที่ใช้ IPX มากรองโดย IP
- นำเฟรมที่ใช้ IP มากรองโดย ARP
- นำเฟรมที่ใช้ ARP มากรองโดย ARP
- นำเฟรมที่ใช้ RARP มากรองโดย ARP
- นำเฟรมที่ใช้ IPX มากรองโดย ARP
- นำเฟรมที่ใช้ IP มากรองโดย RARP
- นำเฟรมที่ใช้ ARP มากรองโดย RARP
- นำเฟรมที่ใช้ RARP มากรองโดย RARP
- นำเฟรมที่ใช้ IPX มากรองโดย RARP
- นำเฟรมที่ใช้ IP มากรองโดย IPX
- นำเฟรมที่ใช้ ARP มากรองโดย IPX
- นำเฟรมที่ใช้ RARP มากรองโดย IPX
- นำเฟรมที่ใช้ IPX มากรองโดย IPX

ผลจากการรันโปรแกรมทดสอบเป็นดังนี้

```
*** filterByNetworkProtocol ***
*   udpFrame : IP
```

```
*   arpFrame : ARP
*   rarpFrame : RARP
*   ipxFrame : IPX

Time 1 : filter udpFrame, need IP
(expect true) : 1
Time 2 : filter arpFrame, need IP
(expect false) : 0
Time 3 : filter rarpFrame, need IP
(expect false) : 0
Time 4 : filter ipxFrame, need IP
(expect false) : 0
Time 5 : filter udpFrame, need ARP
(expect false) : 0
Time 6 : filter arpFrame, need ARP
(expect true) : 1
Time 7 : filter rarpFrame, need ARP
(expect false) : 0
Time 8 : filter ipxFrame, need ARP
(expect false) : 0
Time 9 : filter udpFrame, need RARP
(expect false) : 0
Time 10 : filter arpFrame, need RARP
(expect false) : 0
Time 11 : filter rarpFrame, need RARP
(expect true) : 1
Time 12 : filter ipxFrame, need RARP
(expect false) : 0
Time 13 : filter udpFrame, need IPX
(expect false) : 0
Time 14 : filter arpFrame, need IPX
(expect false) : 0
Time 15 : filter rarpFrame, need IPX
(expect false) : 0
Time 16 : filter ipxFrame, need IPX
(expect true) : 1
```

6.1.5 เมทธอด filterByTransportProtocol

คือ การกรองเฟรมต่าง ๆ โดยใช้โพรโตคอลชั้นทรานสปอร์ตเป็นตัวแยก สามารถทดสอบได้ 20 กรณี ดังนี้

- นำเฟรมที่ใช้ TCP มากรองโดย TCP
- นำเฟรมที่ใช้ UDP มากรองโดย TCP
- นำเฟรมที่ใช้ ICMP มากรองโดย TCP
- นำเฟรมที่ใช้ IPX มากรองโดย TCP
- นำเฟรมที่ใช้ SPX มากรองโดย TCP
- นำเฟรมที่ใช้ TCP มากรองโดย UDP
- นำเฟรมที่ใช้ UDP มากรองโดย UDP
- นำเฟรมที่ใช้ ICMP มากรองโดย UDP

- นำเฟรมที่ใช้ IPX มากรองโดย UDP
- นำเฟรมที่ใช้ SPX มากรองโดย UDP
- นำเฟรมที่ใช้ TCP มากรองโดย ICMP
- นำเฟรมที่ใช้ UDP มากรองโดย ICMP
- นำเฟรมที่ใช้ ICMP มากรองโดย ICMP
- นำเฟรมที่ใช้ IPX มากรองโดย ICMP
- นำเฟรมที่ใช้ SPX มากรองโดย ICMP
- นำเฟรมที่ใช้ TCP มากรองโดย SPX
- นำเฟรมที่ใช้ UDP มากรองโดย SPX
- นำเฟรมที่ใช้ ICMP มากรองโดย SPX
- นำเฟรมที่ใช้ IPX มากรองโดย SPX
- นำเฟรมที่ใช้ SPX มากรองโดย SPX

ผลที่ได้จากการรันโปรแกรมทดสอบเป็นดังนี้

```
*** filterByTransportProtocol ***
*   tcpFrame : TCP
*   udpFrame : UDP
*   icmpFrame : ICMP
*   ipxFrame : IPX
*   spxFrame : SPX

Time 1 : filter tcpFrame, need TCP
(expect true) : 1
Time 2 : filter udpFrame, need TCP
(expect false) : 0
Time 3 : filter icmpFrame, need TCP
(expect false) : 0
Time 4 : filter ipxFrame, need TCP
(expect false) : 0
Time 5 : filter spxFrame, need TCP
(expect false) : 0
Time 6 : filter tcpFrame, need UDP
(expect false) : 0
Time 7 : filter udpFrame, need UDP
(expect true) : 1
Time 8 : filter icmpFrame, need UDP
(expect false) : 0
Time 9 : filter ipxFrame, need UDP
(expect false) : 0
Time 10 : filter spxFrame, need UDP
(expect false) : 0
Time 11 : filter tcpFrame, need ICMP
(expect false) : 0
Time 12 : filter udpFrame, need ICMP
(expect false) : 0
Time 13 : filter icmpFrame, need ICMP
```

```
(expect true) : 1
Time 14 : filter ipxFrame, need ICMP
(expect false) : 0
Time 15 : filter spxFrame, need ICMP
(expect false) : 0
Time 16 : filter tcpFrame, need SPX
(expect false) : 0
Time 17 : filter udpFrame, need SPX
(expect false) : 0
Time 18 : filter icmpFrame, need SPX
(expect false) : 0
Time 19 : filter ipxFrame, need SPX
(expect false) : 0
Time 20 : filter spxFrame, need SPX
(expect true) : 1
```

6.1.6 filterByPortNumber

คือ การนำเฟรมมากรองโดยใช้หมายเลขพอร์ตเป็นตัวแยกโดยเฟรมที่จะกรองผ่านจะต้องมีพอร์ตต้นทางหรือปลายทางตรงกับที่ต้องการ สามารถทดสอบได้ 3 กรณีคือ

- นำเฟรมที่มีพอร์ตปลายทางตรงกับที่ต้องการมากรอง
- นำเฟรมที่ไม่ได้ใช้ทั้งพอร์ตต้นทางและปลายทางตามที่ต้องการมากรอง
- นำเฟรมที่มีพอร์ตต้นทางตรงกับที่ต้องการมากรอง

ผลจากการรันโปรแกรมทดสอบเป็นดังนี้

```
*** filterByPortNumber ***
*   tcpFrame : port 80
*   udpFrame : port 23

Time 1 : filter tcpFrame, need port 80
(expect true) : 1
Time 2 : filter tcpFrame, need port 800
(expect false) : 0
Time 3 : filter tcpFrame, need port 0x1000
(expect true) : 1
Time 4 : filter udpFrame, need port 80
(expect false) : 0
Time 5 : filter udpFrame, need port 23
(expect true) : 1
Time 6 : filter udpFrame, need port 0x1000
(expect true) : 1
```

6.1.7 เมทธอด filterBySize

คือ การนำเฟรมมากรองโดยแยกจากขนาดของเฟรม ซึ่งจะต้องป้อนอินพุทให้ทั้งขนาดต่ำสุดและขนาดสูงสุดที่ต้องการกรอง เฟรมที่ผ่านการกรองจะอยู่ในช่วงดังกล่าวรวมทั้งขนาดต่ำสุดและสูงสุดด้วย สามารถทดสอบได้ 8 กรณีดังนี้

- นำเฟรมที่ใช้ Ethernet II ซึ่งไม่มีฟิลด์บอกขนาดมากรองที่ขนาดไม่ตรงกับขนาดเฟรม
- นำเฟรมที่ใช้ Ethernet II ซึ่งไม่มีฟิลด์บอกขนาด และขนาดของเฟรมอยู่ในช่วงที่ต้องการกรอง
- นำเฟรมที่ใช้ Ethernet 802.2 ซึ่งมีฟิลด์บอกขนาดมากรองที่ขนาดตรงกับขนาดเฟรม
- นำเฟรมที่ใช้ Ethernet 802.2 ซึ่งมีฟิลด์บอกขนาดมากรองที่ขนาดไม่ตรงกับขนาดเฟรม
- นำเฟรมที่ใช้ Ethernet SNAP ซึ่งมีฟิลด์บอกขนาดมากรองที่ขนาดไม่ตรงกับขนาดเฟรม
- นำเฟรมที่ใช้ Ethernet SNAP ซึ่งมีฟิลด์บอกขนาดมากรองที่ขนาดตรงกับขนาดเฟรม
- นำเฟรมที่ใช้ Ethernet 802.3 ซึ่งมีฟิลด์บอกขนาดมากรองที่ขนาดตรงกับขนาดเฟรม
- นำเฟรมที่ใช้ Ethernet 802.3 ซึ่งมีฟิลด์บอกขนาดมากรองที่ขนาดไม่ตรงกับขนาดเฟรม

ผลที่ได้จากการรันโปรแกรมเป็นดังนี้

```
*** filterBySize ***
* tcpFrame : EthII 71 bytes
* udpFrame : 68 bytes
* arpFrame : EthII 42 bytes
* icmpFrame : EthII 54 bytes
* ipxFrame : 93 bytes
* spxFrame : 67 bytes

Time 1 : filter tcpFrame, need 60-70
(expect false) : 0
Time 2 : filter tcpFrame, need 60-71
(expect true) : 1
Time 3 : filter tcpFrame, need 71-71
(expect true) : 1
Time 4 : filter udpFrame, need 60-71
(expect true) : 1
Time 5 : filter udpFrame, need 100-150
(expect false) : 0
Time 6 : filter arpFrame, need 0-150
(expect true) : 1
Time 7 : filter icmpFrame, need 0-150
(expect true) : 1
Time 8 : filter ipxFrame, need 100-150
(expect false) : 0
Time 9 : filter ipxFrame, need 90-100
(expect true) : 1
Time 10 : filter spxFrame, need 60-70
(expect true) : 1
Time 11 : filter spxFrame, need 90-60
(expect false) : 0
```

6.2 การทดสอบอ็อบเจ็ค Fields

เป็นการนำเฟรมที่จับได้หรือกรองได้ก็ตาม มาทำการแยกออกเป็นฟิลด์ต่าง ๆ ตามเฮดเดอร์ที่มีอยู่ในเฟรมนั้นรวมทั้งแสดงข้อมูลจริงเป็นแอสกีด้วย การทดสอบในส่วนนี้จะนำเฟรมที่เราจำลองขึ้นมาใช้ในหัวข้อ 6.1 มาทดสอบ ซึ่งประกอบด้วย เฟรมที่ 1 คือ tcpFrame, เฟรมที่ 2 คือ udpFrame, เฟรมที่ 3 คือ arpFrame,

เฟรมที่ 4 คือ rarpFrame, เฟรมที่ 5 คือ icmpFrame, เฟรมที่ 6 คือ ipxFrame และเฟรมที่ 7 คือ spxFrame ผลที่ได้หลังจากทดสอบเป็นดังนี้

```
Frame 1 :
*** Ethernet II ***
Destination MAC Address : 0xff ff ff ff ff ff
Source MAC Address : 0x1a 2b 3c 4d 5e 6f
Type field : 0x800
*** IP protocol ***
Version : 0x4
Internet Header Length : 0x5
Type of Service
     Priority : 0
     D - flag : 0
     T - flag : 0
     R - flag : 0
     C - flag : 0
     not used : 0
Total length : 57
Identification : 0x0
Flags
     Not used : 0
     Do not fragment : 1
     More flag : 0
Fragment offset : 0x0
Time to live : 254
Protocol : 0x6 (TCP)
Header checksum : 0x0
Source IP address : 161.246.6.92
Destination IP address : 161.246.4.3
*** TCP protocol ***
Source port : 4096
Destination port : 80
Sequence number : 0
Acknowledgement number : 0
Data offset : 5
Control bits
     URG : 1
     ACK : 1
     PSH : 1
     RST : 1
     SYN : 1
     FIN : 1
Window : 0
Checksum : 0x0
Urgent pointer : 0
*** Data section ***
This is TCP .....
--------------------------------------------------------------------------------
Frame 2 :
*** Ethernet 802.2 ***
Destination MAC Address : 0xff ff ff ff ff ff
Source MAC Address : 0x1a 2b 3c 4d 5e 6f
Length field : 68
DSAP : 0x6
SSAP : 0x6
```

```
Control : 0x0
*** IP protocol ***
Version : 0x4
Internet Header Length : 0x5
Type of Service
     Priority : 0
     D - flag : 0
     T - flag : 0
     R - flag : 0
     C - flag : 0
     not used : 0
Total length : 51
Identification : 0x0
Flags
     Not used : 0
     Do not fragment : 1
     More flag : 0
Fragment offset : 0x0
Time to live : 254
Protocol : 0x11 (UDP)
Header checksum : 0x0
Source IP address : 161.246.6.85
Destination IP address : 161.246.10.21
*** UDP protocol ***
Source port : 4096
Destination port : 23
Length : 31
UDP checksum : 0x0
*** Data section ***
This is UDP packet ....
--------------------------------------------------------------------------------
Frame 3 :
*** Ethernet II ***
Destination MAC Address : 0xff ff ff ff ff ff
Source MAC Address : 0x1a 2b 3c 4d 5e 6f
Type field : 0x806
*** ARP protocol ***
Hardware : 1
Protocol : 0x800
HLEN : 6
PLEN : 4
Operation code : 1 (ARP request)
Sender Hardware Address : 0x0 0 0 0 0 0
Sender Internet Address : 0.0.0.0
Target Hardware Address : 0x0 0 0 0 0 0
Target Internet Address : 0.0.0.0
--------------------------------------------------------------------------------
Frame 4 :
*** Ethernet II ***
Destination MAC Address : 0xff ff ff ff ff ff
Source MAC Address : 0x1a 2b 3c 4d 5e 6f
Type field : 0x8035
*** RARP protocol ***
Hardware : 1
Protocol : 0x800
HLEN : 6
PLEN : 4
Operation code : 4 (RARP response)
```

```
Sender Hardware Address : 0x0 0 0 0 0 0
Sender Internet Address : 0.0.0.0
Target Hardware Address : 0x0 0 0 0 0 0
Target Internet Address : 0.0.0.0
--------------------------------------------------------------------------------
Frame 5 :
*** Ethernet II ***
Destination MAC Address : 0xff ff ff ff ff ff
Source MAC Address : 0x1a 2b 3c 4d 5e 6f
Type field : 0x800
*** IP protocol ***
Version : 0x4
Internet Header Length : 0x5
Type of Service
     Priority : 0
     D - flag : 0
     T - flag : 0
     R - flag : 0
     C - flag : 0
     not used : 0
Total length : 40
Identification : 0x0
Flags
     Not used : 0
     Do not fragment : 1
     More flag : 0
Fragment offset : 0x0
Time to live : 254
Protocol : 0x1 (ICMP)
Header checksum : 0x0
Source IP address : 161.246.6.92
Destination IP address : 161.246.4.3
*** ICMP protocol ***
Type : 0 (Echo reply)
Code : 0
Checksum : 0x0
*** Data section ***
Context specific
--------------------------------------------------------------------------------
Frame 6 :
*** Ethernet SNAP ***
Destination MAC Address : 0xff ff ff ff ff ff
Source MAC Address : 0x1a 2b 3c 4d 5e 6f
Length field : 93
DSAP : 0xaa
SSAP : 0xaa
Control : 0x0
Organization code : 0 0 0
Ethernet type : 0x8137
*** IPX protocol ***
Checksum : 0x0
Length : 71
Transport control : 15
Packet type : 4 (IPX)
Destination network address : 0x0 0 0 0
Destination node address : 0xff ff ff ff ff ff
Destination socket : 0x4 51
Source network address : 0x0 0 0 0
```

```
Source node address : 0x0 0 0 0 0 0
Source socket : 0x0 0
*** IPX protocol ***
Checksum : 0x0
Length : 41
Transport control : 15
Packet type : 17 (NCP communications)
Destination network address : 0x0 0 0 0
Destination node address : 0xff ff ff ff ff ff
Destination socket : 0x4 51
Source network address : 0x0 0 0 0
Source node address : 0x0 0 0 0 0 0
Source socket : 0x0 0
*** Data section ***
This is NCP
--------------------------------------------------------------------------------
Frame 7 :
*** Ethernet 802.3 ***
Destination MAC Address : 0xff ff ff ff ff ff
Source MAC Address : 0x1a 2b 3c 4d 5e 6f
Length field : 67
*** IPX protocol ***
Checksum : 0xffff
Length : 53
Transport control : 15
Packet type : 5 (SPX)
Destination network address : 0x0 0 0 0
Destination node address : 0xff ff ff ff ff ff
Destination socket : 0x4 51
Source network address : 0x0 0 0 0
Source node address : 0x0 0 0 0 0 0
Source socket : 0x0 0
*** SPX protocol ***
Connection control : 0x10
Datastream type : 0xff
Source connection ID : 0
Destination connection ID : 0
Sequence number : 0
Acknowledgement number : 0
Allocation number : 0
*** Data section ***
This is SPX
```

6.3 การทดสอบอ็อบเจ็ค packetAnalyzer

อ็อบเจ็คนี้เป็นการหาสถิติเกี่ยวกับเฟรมที่สนใจ ซึ่งในการทดสอบอ็อบเจ็คนี้นั้นจะนำเฟรมที่ได้จำลองไว้เพื่อทดสอบอ็อบเจ็ค packetFilter มาใช้อีกครั้งรวมทั้งได้ทำการจำลองเฟรมขึ้นมาใหม่อีก 8 เฟรมดังตารางข้างล่างนี้

ชื่อ	Datalink	Network	Transport	Size
ftpFrame	Ethernet SNAP	IP	UDP	420 bytes

	Source: 0x1a-2b-3c-4d-5e-6f Destination: 0xff-ff-ff-ff-ff-ff	Source IP: 161.246.6.85 Destination IP: 161.246.10.21	Source port: 0x10-00 Destination port: 20	
smtpFrame	Ethernet SNAP Source: 0x11-22-33-44-55-66 Destination: 0x01-02-03-04-05-06	IP Source IP: 161.246.6.82 Destination IP: 161.246.10.22	UDP Source port: 0x10-00 Destination port: 25	421 bytes
gopherFrame	Ethernet SNAP Source: 0x1a-2b-3c-4d-5e-6f Destination: 0x01-02-03-04-05-06	IP Source IP: 161.246.6.85 Destination IP: 161.246.10.22	UDP Source port: 0x10-00 Destination port: 70	780 bytes
pop3Frame	Ethernet SNAP Source: 0x11-22-33-44-55-66 Destination: 0xff-ff-ff-ff-ff-ff	IP Source IP: 161.246.6.82 Destination IP: 161.246.10.21	UDP Source port: 0x10-00 Destination port: 110	781 bytes
nntpFrame	Ethernet II Source: 0x1a-2b-3c-4d-5e-6f Destination: 0xff-ff-ff-ff-ff-ff	IP Source IP: 161.246.6.85 Destination IP: 161.246.10.21	UDP Source port: 0x10-00 Destination port: 119	1140 bytes
snmpFrame	Ethernet II Source: 0x1a-2b-3c-4d-5e-6f Destination: 0xff-ff-ff-ff-ff-ff	IP Source IP: 161.246.6.85 Destination IP: 161.246.10.21	UDP Source port: 0x10-00 Destination port: 161	1141 bytes
ircFrame	Ethernet II Source:	IP Source IP:	UDP Source port:	1514 bytes

	0x1a-2b-3c-4d-5e-6f Destination: 0xff-ff-ff-ff-ff-ff	161.246.6.85 Destination IP: 161.246.10.21	0x10-00 Destination port: 194	
anyFrame	Ethernet II Source: 0x1a-2b-3c-4d-5e-6f Destination: 0xff-ff-ff-ff-ff-ff	IP Source IP: 161.246.6.85 Destination IP: 161.246.10.21	UDP Source port: 0x10-00 Destination port: 0x50-00	60 bytes

ตารางที่ 6-2 แสดงเฟรมที่จำลองขึ้นมาทดสอบ(เพิ่มเติม)

6.3.1 เมทธอด usedApp

คือ การหาเปอร์เซ็นต์ของพอร์ต 20, 23, 25, 70, 80, 110, 119, 161 และ 194 จากจำนวนเฟรมทั้งหมดที่นำมาหา ในการทดสอบครั้งนี้จะมีเฟรมทั้งหมด 16 เฟรม โดยจะมีการใช้แต่ละพอร์ตเท่ากับ 1 ยกเว้นพอร์ต 20 และ 80 ซึ่งจะมีการใช้ 2 และ 5 ตามลำดับ

ผลจากการรันโปรแกรมทดสอบเป็นดังนี้

```
*** usedApp ***
*   ftpFrame : FTP port 20
*   udpFrame : Telnet port 23
*   smtpFrame : SMTP port 25
*   gopherFrame : Gopher port 70
*   tcpFrame : HTTP port 80
*   pop3Frame : POP3 port 110
*   nntpFrame : NNTP port 119
*   snmpFrame : SNMP port 161
*   ircFrame : IRC port 194
*   arpFrame : not use port
*   anyFrame : port 0x50 00

FTP 2/16 : 12.5 %
TELNET 1/16 : 6.25 %
SMTP 1/16 : 6.25 %
GOPHER 1/16 : 6.25 %
HTTP 5/16 : 31.25 %
POP3 1/16 : 6.25 %
NNTP 1/16 : 6.25 %
SNMP 1/16 : 6.25 %
IRC 1/16 : 6.25 %
```

6.3.2 เมทธอด usedTransport

คือ การคำนวณหาเปอร์เซ็นต์การใช้งานโพรโตคอลชั้นทรานสปอร์ต ซึ่งจะหาเปอร์เซ็นต์การใช้งาน

โพรโตคอล TCP, UDP, ICMP และ SPX การทดสอบครั้งนี้จะหาเปอร์เซ็นต์จากเฟรมทั้งหมด 5 เฟรมโดยมี 1 เฟรมที่ไม่ได้ใช้โพรโตคอลใดเลยใน 4 โพรโตคอลข้างต้น คือ ipxFrame ซึ่งใช้โพรโตคอลไอพีเอ็กซ์ต่อจากไอพีเอ็กซ์ในชั้นเน็ตเวิร์คซ้ำอีกครั้ง

ผลจากการทดสอบเป็นดังนี้

```
*** usedTransport ***
*   tcpFrame : TCP
*   udpFrame : UDP
*   icmpFrame : ICMP
*   spxFrame : SPX
*   ipxFrame : IPX over IPX

TCP 1/5 : 20 %
UDP 1/5 : 20 %
ICMP 1/5 : 20 %
SPX 1/5 : 20 %
```

6.3.3 เมทธอด usedNetwork

คือ การหาเปอร์เซ็นต์การใช้งานของโพรโตคอลที่ใช้ในชั้นเน็ตเวิร์ค ซึ่งสามารถหาได้ 4 โพรโตคอล คือ IP, ARP, RARP และ IPX

ผลการรันโปรแกรมทดสอบเป็นดังนี้

```
*** usedNetwork ***
*   udpFrame : IP
*   arpFrame : ARP
*   rarpFrame : RARP
*   ipxFrame : IPX
*   icmpFrame : IP
*   spxFrame : IPX

IP 2/6 : 33.333332 %
ARP 1/6 : 16.666666 %
RARP 1/6 : 16.666666 %
IPX 2/6 : 33.333332 %
```

6.3.4 เมทธอด usedEthernet

คือ การหาเปอร์เซ็นต์การใช้งานโพรโตคอลในชั้นดาต้าลิงค์ ซึ่งจะมี 4 ชนิดคือ Ethernet 802.3, Ethernet 802.2, Ethernet SNAP และ Ethernet II

ผลการรันโปรแกรมทดสอบเป็นดังนี้

```
*** usedEthernet ***
*   tcpFrame : Ethernet II
*   udpFrame : Ethernet 802.2
*   ipxFrame : Ethernet SNAP
```

```
*   spxFrame : Ethernet 802.3

Ethernet 802.3 1/4 : 25 %
Ethernet 802.2 1/4 : 25 %
Ethernet SNAP 1/4 : 25 %
Ethernet II 1/4 : 25 %
```

6.3.5 เมทธอด distributedSize

คือ การหาเปอร์เซ็นต์การใช้งานตามขนาดของเฟรม ซึ่งเราได้แบ่งออกเป็น 4 ช่วงคือ

- ช่วงของเฟรมที่มีขนาด 60-420 ไบต์
- ช่วงของเฟรมที่มีขนาด 421-780 ไบต์
- ช่วงของเฟรมที่มีขนาด 781-1140 ไบต์
- ช่วงของเฟรมที่มีขนาด 1141-1514 ไบต์

ผลหลังจากการรันโปรแกรมทดสอบเป็นดังนี้

```
*** distributedSize ***
*   arpFrame : not have length field, 42 bytes
*   anyFrame : not have length field, 60 bytes
*   ftpFrame : have length field, 420 bytes
*   smtpFrame : have length field, 421 bytes
*   gopherFrame : have length field, 780 bytes
*   pop3Frame : have length field, 781 bytes
*   nntpFrame : not have length field, 1140 bytes
*   snmpFrame : not have length field, 1141 bytes
*   ircFrame : not have length field, 1514 bytes

60-420 bytes 2/9 : 22.222221 %
421-780 bytes 2/9 : 22.222221 %
781-1140 bytes 2/9 : 22.222221 %
1141-1514 bytes 2/9 : 22.222221 %
```

6.3.6 เมทธอด percentUsedNode

แบบ 2 แอดเดรส คือ การหาเปอร์เซ็นต์การใช้งานของเฟรมที่มาจากโหนดที่เราสนใจโดยเฟรมดังกล่าวจะต้องมีแมคแอดเดรสตรงตามที่ระบุทั้งต้นทางและปลายทาง ในการทดสอบนี้ได้ทดสอบกับเฟรมที่จำลองขึ้นมา 4 เฟรมซึ่งก็คือ

- มีแมคแอดเดรสต้นทางและปลายทางตรงกับที่ต้องการ
- มีแมคแอดเดรสต้นทางและปลายทางไม่ตรงกับที่ต้องการ
- มีแมคแอดเดรสต้นทางเท่านั้นที่ตรงกับความต้องการ
- มีแมคแอดเดรสปลายทางเท่านั้นที่ตรงกับความต้องการ

ผลการรันโปรแกรมทดสอบเป็นดังนี้

```
*** percentUsedNode 2 address ***
*   need Source : 0x 1a 2b 3c 4d 5e 6f
*        Destination : 0xff ff ff ff ff ff

1/4 : 25 %
```

แบบ 1 แอดเดรส คือ การหาเปอร์เซ็นต์การใช้งานของเฟรมที่มาจากโหนดที่เราสนใจโดยเฟรมดังกล่าวจะต้องมีแมคแอดเดรสตรงตามที่ระบุซึ่งอาจตรงกับแอดเดรสต้นทางหรือปลายทางก็ได้ ในการทดสอบนี้ได้ทดสอบกับเฟรมที่จำลองขึ้นมา 4 เฟรมซึ่งก็คือ

- มีแมคแอดเดรสต้นทางและปลายทางตรงกับที่ต้องการ
- มีแมคแอดเดรสต้นทางและปลายทางไม่ตรงกับที่ต้องการ
- มีแมคแอดเดรสต้นทางเท่านั้นที่ตรงกับความต้องการ
- มีแมคแอดเดรสปลายทางเท่านั้นที่ตรงกับความต้องการ

ผลการรันโปรแกรมทดสอบเป็นดังนี้

```
*** percentUsedNode 1 address ***
*   need any MAC address : 0x1a 2b 3c 4d 5e 6f

2/4 : 50 %
```

6.3.7 เมทธอด percentUsedPacket

แบบ 2 แอดเดรส คือ การหาเปอร์เซ็นต์การใช้งานของเฟรมที่มาจากสเตชั่นที่เราสนใจโดยเฟรมดังกล่าวจะต้องมีไอพีแอดเดรสตรงตามที่ระบุทั้งต้นทางและปลายทาง ในการทดสอบนี้ได้ทดสอบกับเฟรมที่จำลองขึ้นมา 4 เฟรมซึ่งก็คือ

- มีไอพีแอดเดรสต้นทางและปลายทางตรงกับที่ต้องการ
- มีไอพีแอดเดรสต้นทางและปลายทางไม่ตรงกับที่ต้องการ
- มีไอพีแอดเดรสต้นทางเท่านั้นที่ตรงกับความต้องการ
- มีไอพีแอดเดรสปลายทางเท่านั้นที่ตรงกับความต้องการ

ผลการรันโปรแกรมทดสอบเป็นดังนี้

```
*** percentUsedPacket 2 address ***
*   need Source : 161.246.6.85
*        Destination : 161.246.10.21

2/8 : 25 %
```

แบบ 1 แอดเดรส คือ การหาเปอร์เซ็นต์การใช้งานของเฟรมที่มาจากสเตชั่นที่เราสนใจโดยเฟรมดังกล่าวอาจจะมีไอพีแอดเดรสตรงกับแอดเดรสต้นทางหรือปลายทางก็ได้ ในการทดสอบนี้ได้ทดสอบกับเฟรมที่

จำลองขึ้นมา 4 เฟรมซึ่งก็คือ

- มีไอพีแอดเดรสต้นทางและปลายทางตรงกับที่ต้องการ
- มีไอพีแอดเดรสต้นทางและปลายทางไม่ตรงกับที่ต้องการ
- มีไอพีแอดเดรสต้นทางเท่านั้นที่ตรงกับความต้องการ
- มีไอพีแอดเดรสปลายทางเท่านั้นที่ตรงกับความต้องการ

ผลการรันโปรแกรมทดสอบเป็นดังนี้

```
*** percentUsedPacket 1 address ***
*   need any IP address : 161.246.10.21

3/7 : 42.857143 %
```

6.4 การทดสอบอ็อบเจ็ค PktDriver

อ็อบเจ็คนี้มีหน้าที่ในการจับและส่งแพ็คเกจออกไป ในการทดสอบนี้ฟังก์ชันที่มีไว้สำหรับการรับหรือก็คือ receiver จะต้องอยู่ในโปรแกรมหลักหรือที่มี main() เพราะเป็นฟังก์ชันที่จะถูกเรียกใช้เมื่อมีแพ็คเกจเข้ามาทุกครั้งโดยการอินเทอร์รัพท์

ผลจากการรันโปรแกรมทดสอบเป็นดังนี้

```
0000  ff ff ff ff ff ff 00 20 18 62 38 aa 00 50 ff ff  ....... .b8*.P..
0010  00 50 02 14 a1 f6 06 00 ff ff ff ff ff ff 04 55  .P..!v.........U
0020  a1 f6 05 00 00 aa 00 69 68 2c 04 55 a1 f6 05 00  !v...*.ih,.U!v..
0030  19 92 07 03 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040  00 00 00 00 00 00 00 00 00 00 00 00 00 01 4f 52  ..............OR
0050  49 4f 4e 20 20 20 20 20 20 20 20 20 20 1b        ION           .
0000  00 a0 24 b3 54 e0 00 20 18 2c a1 63 08 00 45 00  . $3T`. .,!c..E.
0010  00 2c b8 0a 40 00 80 06 ee 6a a1 f6 06 55 a1 f6  ..8.@...nj!v.U!v
0020  0a 15 04 9b 0c 38 00 bd 19 19 00 00 00 00 60 02  .....8.=......`.
0030  20 00 f9 26 00 00 02 04 05 b4 20 20              .y&.....4
0000  00 20 18 2c a1 63 00 a0 24 b3 54 e0 08 00 45 00  . .,!c. $3T`..E.
0010  00 2c 62 e3 40 00 fd 06 c6 91 a1 f6 0a 15 a1 f6  ..bc@.}.F.!v..!v
0020  06 55 0c 38 04 9b fd 36 be 82 00 bd 19 1a 60 12  .U.8..}6>..=..`.
0030  22 38 3b 24 00 00 02 04 05 b4 14 0a              "8;$.....4..
0000  00 a0 24 b3 54 e0 00 20 18 2c a1 63 08 00 45 00  . $3T`. .,!c..E.
0010  01 6a ba 0a 40 00 80 06 eb 2c a1 f6 06 55 a1 f6  .j:.@...k,!v.U!v
0020  0a 15 04 9b 0c 38 00 bd 19 1a fd 36 be 83 50 18  .....8.=..}6>.P.
0030  22 38 f2 9f 00 00 47 45 54 20 68 74 74 70 3a 2f  "8r...GET http:/
0040  2f 70 61 6e 74 69 70 2e 69 6e 65 74 2e 63 6f 2e  /pantip.inet.co.
0050  74 68 2f 62 61 6e 6e 65 72 2f 74 65 63 68 2f 62  th/banner/tech/b
0060  61 6e 5f 74 65 63 68 2e 73 68 74 6d 6c 20 48 54  an_tech.shtml HT
0070  54 50 2f 31 2e 30 0d 0a 50 72 6f 78 79 2d 43 6f  TP/1.0..Proxy-Co
0080  6e 6e 65 63 74 69 6f 6e 3a 20 4b 65 65 70 2d 41  nnection: Keep-A
0090  6c 69 76 65 0d 0a 55 73 65 72 2d 41 67 65 6e 74  live..User-Agent
00a0  3a 20 4d 6f 7a 69 6c 6c 61 2f 34 2e 30 35 20 5b  : Mozilla/4.05 [
00b0  65 6e 5d 20 28 57 69 6e 39 35 3b 20 49 20 3b 4e  en] (Win95; I ;N
00c0  61 76 29 0d 0a 50 72 61 67 6d 61 3a 20 6e 6f 2d  av)..Pragma: no-
00d0  63 61 63 68 65 0d 0a 48 6f 73 74 3a 20 70 61 6e  cache..Host: pan
```

```
00e0  74 69 70 2e 69 6e 65 74 2e 63 6f 2e 74 68 0d 0a  tip.inet.co.th..
00f0  41 63 63 65 70 74 3a 20 69 6d 61 67 65 2f 67 69  Accept: image/gi
0100  66 2c 20 69 6d 61 67 65 2f 78 2d 78 62 69 74 6d  f, image/x-xbitm
0110  61 70 2c 20 69 6d 61 67 65 2f 6a 70 65 67 2c 20  ap, image/jpeg,
0120  69 6d 61 67 65 2f 70 6a 70 65 67 2c 20 69 6d 61  image/pjpeg, ima
0130  67 65 2f 70 6e 67 2c 20 2a 2f 2a 0d 0a 41 63 63  ge/png, */*..Acc
0140  65 70 74 2d 4c 61 6e 67 75 61 67 65 3a 20 65 6e  ept-Language: en
0150  0d 0a 41 63 63 65 70 74 2d 43 68 61 72 73 65 74  ..Accept-Charset
0160  3a 20 69 73 6f 2d 38 38 35 39 2d 31 2c 2a 2c 75  : iso-8859-1,*,u
0170  74 66 2d 38 0d 0a 0d 0a                          tf-8....
0000  00 20 18 2c a1 63 00 a0 24 b3 54 e0 08 00 45 00  . .,!c. $3T`..E.
0010  00 28 62 e4 40 00 fd 06 c6 94 a1 f6 0a 15 a1 f6  .(bd@.}.F.!v..!v
0020  06 55 0c 38 04 9b fd 36 be 83 00 bd 1a 5c 50 10  .U.8..}6>..=.\P.
0030  22 38 51 9f 00 00 02 04 05 b4 14 0a              "8Q.....4..
```

บทที่ 7
บทวิจารณ์และสรุป

7.1 วิจารณ์โครงงาน

1. การพัฒนาโปรแกรม

ในการเขียนโปรแกรมภาษา C++ นั้นมีใกล้เคียงกับการเขียนภาษา C มาก แต่บางคำสั่งที่ภาษา 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

7.2 สรุปผลโครงงาน

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

7.2 ข้อเสนอแนะและแนวทางการพัฒนาต่อไป

1. ในการพัฒนาโปรแกรมประเภทนี้ต่อไปนั้น ควรทำให้เป็นโปรแกรมประเภท 32 บิต เพื่อจะได้รันบนระบบปฏิบัติการแบบวินโดวส์ซึ่งสะดวกในการทำงาน ถึงแม้ว่าโปรแกรมที่โครงงานนี้ได้พัฒนาขึ้นมาจะสามารถทำให้รันบนวินโดวส์ได้ก็ตาม แต่ไม่สะดวกเพราะต้องกำหนดค่าเพื่อบังคับให้ไดรเวอร์รับข้อมูลได้ถูกต้องแบบ 16 บิต ซึ่งทำให้โปรแกรมแบบ 32 บิตทำงานไม่ได้

2. ในการพัฒนาครั้งนี้ได้ทดสอบกับแพ็คเกจไดรเวอร์เพียงชนิดเดียวเท่านั้นคือ NE2000.COM

ซึ่งอาจเข้ากันไม่ได้กับไดรเวอร์ชนิดอื่น ดังนั้นในการพัฒนาต่อไปควรทำให้ใช้ได้กับไดรเวอร์ทุก ๆ ชนิด หรือชนิดที่เป็นที่นิยม

3. ในการทำให้ติดต่อกับการ์ดแลนได้เกือบทุกชนิดนั้นควรใช้ไดรเวอร์ NDIS(Network Driver Interface Specification)

4. ควรพัฒนาให้โปรแกรมรู้จักกับโพรโตคอลที่หลากหลายกว่านี้ เพราะยิ่งรู้จักโพรโตคอลมากขึ้นเท่าใดก็จะยิ่งเกิดประโยชน์กับผู้ใช้มากขึ้นเท่านั้น

5. ไม่ควรพัฒนาโปรแกรมโดยเริ่มต้นใหม่เองทั้งหมด ถ้าจะทำโปรแกรมประเภทมอนิเตอร์เครือข่ายเช่นเดียวกับที่โครงงานนี้ได้ทำแล้ว ควรทำต่อจากที่มีอยู่เพราะจะทำให้ประหยัดเวลา ทำให้มีโปรแกรมที่มีประสิทธิภาพมากยิ่งขึ้นไปเป็นลำดับ

6. ถ้าต้องการจะพัฒนาใหม่โดยใช้ภาษาอื่น ภาษาจาวาก็เป็นภาษาที่น่าสนใจ เพราะเขียนครั้งเดียวรันได้หลายแพลตฟอร์ม

ภาคผนวก ก

รายละเอียดของการติดต่อกับการ์ดแลน

ก.1 รายละเอียดแสดงเน็ตเวิร์กอินเตอร์เฟสคลาส (Network interface classes)

โดยแบ่งแยกตามคลาสได้ดังนี้

DEC/Intel/Xerox "Bluebook" Ethernet

Class 1

3COM 3C500/3C501	1
3COM 3C505	2
Interlan Ni5010	3
BICC Data Networks 4110	4
BICC Data Networks 4117	5
MICOM-Interlan NP600	6
Ungermann-Bass PC-NIC	8
Univation NC-516	9
TRW PC-2000	10
Interlan Ni5210	11
3COM 3C503	12
3COM 3C523	13
Western Digital WD8003	14
Spider Systems S4	15
Torus Frame Level	16
10NET Communications	17
Gateway PC-bus	18
Gateway AT-bus	19
Gateway MCA-bus	20
IMC PCnic	21
IMC PCnic II	22
IMC PCnic 8bit	23
Tigan Communications	24
Micromatic Research	25
Clarkson "Multiplexor"	26
D-Link 8-bit	27

D-Link 16-bit	28
D-Link PS/2	29
Research Machines 8	30
Research Machines 16	31
Research Machines MCA	32
Radix Microsys. EXM1 16-bit	33
Interlan Ni9210	34
Interlan Ni6510	35
Vestra LANMASTER 16-bit	36
Vestra LANMASTER 8-bit	37
Allied Telesis PC/XT/AT	38
Allied Telesis NEC PC-98	39
Allied Telesis Fujitsu FMR	40
Ungermann-Bass NIC/PS2	41
Tiara LANCard/E AT	42
Tiara LANCard/E MC	43
Tiara LANCard/E TP	44
Spider Comm. SpiderComm 8	45
Spider Comm. SpiderComm 16	46
AT&T Starlan NAU	47
AT&T Starlan-10 NAU	48
AT&T Ethernet NAU	49
Intel smart card	50

ProNET-10

Class 2

Proteon p1300	1
Proteon p1800	2

IEEE 802.5/ProNET-4

Class 3

IBM Token ring adapter	1
Proteon p1340	2
Proteon p1344	3

Gateway PC-bus 4

Gateway AT-bus 5

Gateway MCA-bus 6

Omninet

Class 4

Appletalk

Class 5

Serial line

Class 6

Clarkson 8250-SLIP 1

Clarkson "Multiplexor" 2

Starlan

Class 7 (NOTE: Class has been subsumed by Ethernet)

ArcNet

Class 8

Datapoint RIM 1

AX.25

Class 9

KISS

Class 10

IEEE 802.3 w/802.2 hdrs

Class 11

Types as given under DIX Ethernet

FDDI w/802.2 hdrs

Class 12

Internet X.25

Class 13

Western Digital	1
Frontier Technology	2

N.T. LANSTAR (encapsulating DIX)

Class 14

NT LANSTAR/8	1
NT LANSTAR/MC	2

ก.2 หมายเลขฟังก์ชัน (Function call number)

driver_info	1
access_type	2
release_type	3
send_pkt	4
terminate	5
get_address	6
reset_interface	7
+get_parameters	10
+as_send_pkt	11
*set_rcv_mode	20
*get_rcv_mode	21
*set_multicast_list	22
*get_multicast_list	23
*get_statistics	24
*set_address	25

ก.3 รหัสแสดงความผิดพลาด (Error Codes)

1	BAD_HANDLE	Invalid handle number
2	NO_CLASS	No interfaces of specified class found
3	NO_TYPE	No interfaces of specified type found
4	NO_NUMBER	No interfaces of specified number found
5	BAD_TYPE	Bad packet type specified
6	NO_MULTICAST	This interface does not support multicast
7	CANT_TERMINATE	This packet driver cannot terminate
8	BAD_MODE	An invalid receiver mode was specified
9	NO_SPACE	Operation failed because of insufficient space
10	TYPE_INUSE	The type had previously been accessed and not released
11	BAD_COMMAND	The command was out of range or not implemented
12	CANT_SEND	The packet couldn't be sent (usually hardware error)
13	CANT_SET	Hardware address couldn't be changed (more than 1 handle open)
14	BAD_ADDRESS	Hardware address has bad length or format
15	CANT_RESET	Couldn't reset interface (more than 1 handle open)

Ethernet II Frame

Destination Address	Source Address	Type	Data
6 bytes	6 bytes	2 bytes	46 - 1500 bytes

Ethernet 802.3 Frame

Destination Address	Source Address	Length	Data (เริ่มต้นด้วย 0xFFFF)
6 bytes	6 bytes	2 bytes	46 - 1500 bytes

Ethernet 802.2 Frame

Destination Address	Source Address	Length	DSAP	SSAP	Control	Data
6 bytes	6 bytes	2 bytes	1 byte	1 byte	1 byte	43 - 1497 byte

Ethernet SNAP Frame

Destination Address	Source Address	Length	DSAP	SSAP	Control	Organization Code	Ethernet Type	Data
6 bytes	6 bytes	2 bytes	1 byte	1 byte	1 byte	3 bytes	2 bytes	38 - 1492 bytes

IPX Packet

Check Sum	Length	Trans-port	Packet Type	Destination Network	Destination Host	Destination Socket	Source Network	Source Host	Source Socket	Data
2 bytes	2 bytes	1 byte	1 byte	4 bytes	6 bytes	2 bytes	4 bytes	6 bytes	2 bytes	

IP Packet

Version+Header Length	Type of Service	Length	Identifier	Flag	Fragment Offset	Time to live	Protocol	Header Check Sum	Source Address	Destination Address	Option	Data
1 byte (1/2+1/2)	1 byte	2 bytes	2 bytes	3 bits	13 bits	1 byte	1 byte	2 bytes	4 bytes	4 bytes	variable	

ARP Packet

Hardware Type	Protocol Type	Hardware Address Length	Protocol Address Length	Operation Code	Send Hardware Address	Send Protocol Address	Target Hardware Address	Target Protocol Address
2 bytes	2 bytes	1 byte	1 byte	2 bytes	6 bytes	4 bytes	6 bytes	4 bytes

TCP Packet

Source Port	Destination Port	Sequence Number	Acknowledge-ment Number	Data offset	Rese-rved	Code	Windows	Check Sum	Urgent Pointer	Option	Data
2 bytes	2 bytes	4 bytes	4 bytes	4 bits	6 bits	6 bits	2 bytes	2 bytes	2 bytes	variable	

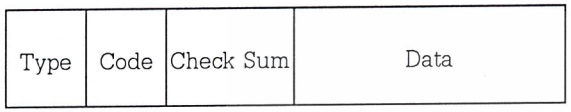
UDP Packet

Source Port	Destination Port	Length	Check Sum	Data
2 bytes	2 bytes	2 bytes	2 bytes	

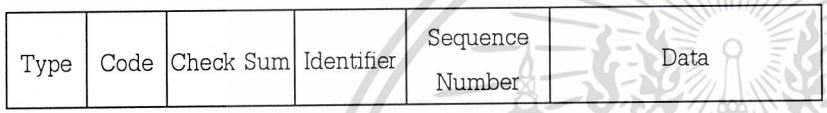
SPX Packet

Control	Data Type	Source ID	Destination ID	Sequence Number	Acknowledge-ment Number	Allocation Number	Data
1 byte	1 byte	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	

ICMP Packet

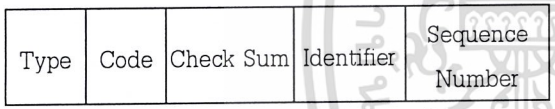


Echo and Echo Reply Message



(0 or 8)

Information and Information Reply Message

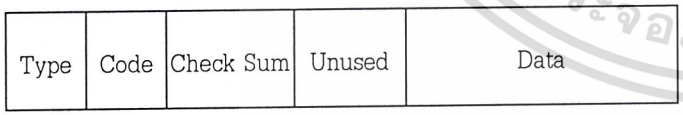


(15 or 16)

Destination Unreachable Message

Source Quench Message

Time Exceeded Message

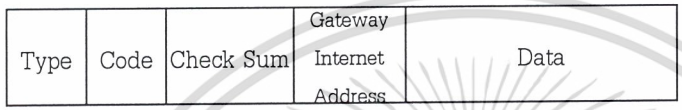


(3, 4 or 11)

ICMP Message Types

ชนิด	ความหมาย
0	Echo Reply
3	Destination Unreachable
4	Source Quench
5	Redirect
8	Echo
11	Time Exceeded
12	Parameter Problem
13	Timestamp
14	Timestamp Reply
15	Information Request
16	Information Reply

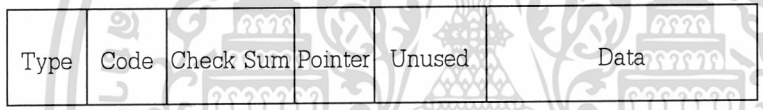
Redirect Message



1 byte 1 byte 2 bytes 4 bytes

(5)

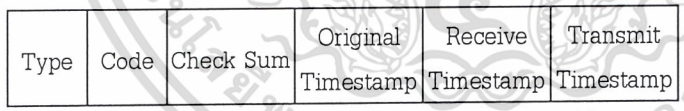
Parameter Problem Message



1 byte 1 byte 2 bytes 1 byte 3 bytes

(12)

Timestamp and Timestamp Reply Message



1 byte 1 byte 2 bytes 4 bytes 4 bytes 4 bytes

(13 or 14)

ภาคผนวก ค

การพิจารณาถึงสมรรถภาพของ CSMA/CD

ถ้าเราทำตามหลักเกณฑ์ในการส่งและรับแพ็คเกจเราก็จะสามารถติดต่อสื่อสารได้บนแลนอีเธอร์เน็ต แต่ก็ยังมีปัจจัยอื่นที่จะมากระทบความสามารถในการรับส่งแพ็คเกจได้ ตัวอย่างเช่นอัตราการใช้งานของระบบเคเบิ้ล ข้อผิดพลาดบนแลน การกำหนดติดตั้งที่ไม่มีประสิทธิภาพ เป็นต้น

ค.1 อัตราการใช้งานแบนด์วิธ

ถ้าเรามองเคเบิ้ลเป็นถนน ในช่วงเวลาเร่งด่วน การติดต่อสื่อสารก็จะช้าเพราะมีรถมากและอาจมีจำนวนอุบัติเหตุสูงกว่าช่วงเวลาอื่นทำให้การจราจรทวีความติดขัดมากยิ่งขึ้นไปอีก ในอีเธอร์เน็ตก็เช่นกัน โดยช่วงเวลาเร่งด่วนของแลนอีเธอร์เน็ตก็อาจจะเกิดขึ้นได้ในเวลา 8.00 น. และ 17.00 น. ซึ่งเป็นเวลาที่ผู้ใช้ทำการล็อกอิน และล็อกเอาท์ออกจากระบบตามลำดับ หรืออาจจะเกิดขึ้นในเวลาที่มีการแบ็คอัพข้ามเครือข่าย ซึ่งในช่วงเวลานี้ผู้ใช้จะพบว่าเครือข่ายช้ากว่าปกติ เพราะมีกิจกรรมมากมายเกิดขึ้นในสายเคเบิ้ล ซึ่งจะนำไปสู่การชนกันของแพ็คเกจ ถึงแม้ว่าการชนกันจะเป็นเรื่องปกติในแลนแบบอีเธอร์เน็ตนี้ก็ตาม แต่ถ้ามันเกิดมากจนเกินไปก็จะทำให้ประสิทธิภาพของเครือข่ายลดลงได้

ค.1.1 การหาอัตราการใช้งานแบนด์วิธในขณะใดขณะหนึ่ง

แบนด์วิธมากที่สุดในอีเธอร์เน็ตจะเท่ากับ 10 Mbit/s อัตราการใช้งานแบนด์วิธก็คือ แบนด์วิธทั้งหมดที่ใช้อยู่ ตัวอย่างเช่น เซ็กเมนท์ที่มีผู้ใช้งานอยู่มากอาจพบว่ามีอัตราการใช้งาน 60 เปอร์เซ็นต์ของแบนด์วิธหรือ 60 เปอร์เซ็นต์ของ 10 Mbit/s การหาอัตราการใช้งานแบนด์วิธทำได้โดยหา Kbytes/s ที่ใช้อยู่แล้วนำมาคิดหน่วยเป็นเปอร์เซ็นต์ของแบนด์วิธทั้งหมดเลย

<u>การใช้งานระดับใดที่ถือว่าปกติ</u>

แลนแบบอีเธอร์เน็ตแต่ละแลนจะถูกกำหนดติดตั้งมาต่างกัน ความปกติในแลนหนึ่งอาจไม่ปกติสำหรับอีกแลนหนึ่งก็เป็นได้ เราจำเป็นจะต้องมีข้อมูลที่เพียงพอซึ่งจะทำให้เราตอบคำถามดังต่อไปนี้ได้

- โหลดในเครือข่ายของเราเพิ่มขึ้นหรือไม่
- มีพีค(peak) ที่ไม่ปกติเกิดขึ้นหรือไม่
- เมื่อใดที่เกิดพีคสูงสุดและต่ำสุดในแบนด์วิธ
- โพรโตคอลชนิดใดที่ใช้แบนด์วิธ
- ผู้ใช้คนใดใช้แบนด์วิธมากที่สุด

การจะให้ได้มาซึ่งข้อมูลเหล่านี้จะต้องใช้ทูลประเภทโพรโตคอลอนาไลเซอร์(Protocol Analyzer) ช่วยซึ่งข้อมูลนี้สามารถนำไปเปรียบเทียบกับเหตุการณ์ในอนาคตได้

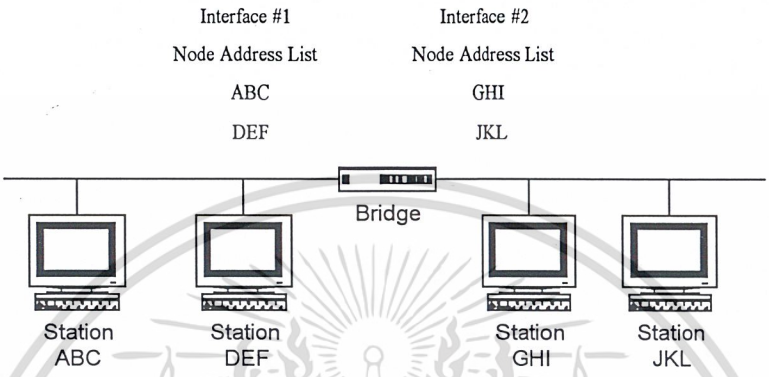
<u>การเปลี่ยนแปลงของอัตราการใช้งานแบนด์วิธ</u>

ถ้ามีผู้ใช้แจ้งให้ทราบถึงสมรรถภาพที่ต่ำลงของระบบ เราสามารถนำข้อมูลที่ได้มาเปรียบเทียบกับสมรรถภาพของเครือข่ายขณะนั้นได้ และสามารถใช้ทูลช่วยในการหาสาเหตุของการใช้งานแบนด์วิธที่เพิ่มขึ้นได้ เช่นหาว่าขณะนั้นใครที่ใช้แบนด์วิธมากที่สุด ซึ่งเราสามารถแจ้งให้เขาทราบเพื่อให้ลดการใช้งานลงได้ แต่ถ้าเราพบว่าแบนด์วิธของเครือข่ายถูกใช้งานเพิ่มขึ้นตลอดเวลาของแต่ละวัน เราก็สามารถเพิ่ม

สมรรถภาพของเครือข่ายได้โดยแยกระบบเคเบิ้ลออกจากกันโดยใช้บริดจ์หรือเราเตอร์ หรือก็คือการทำโหลดบาลานซิ่ง(load balancing) นั่นเอง

<u>การทำโหลดบาลานซ์โดยใช้บริดจ์</u>

บริดจ์ติดต่อกับ 2 เซ็กเมนท์และไม่ยอมให้มีการส่งข้อมูลที่ปลายทางอยู่เซ็กเมนท์เดียวกันข้ามไปได้ ดังรูปที่ ค-1 การที่บริดจ์รู้ว่าเครื่องหรือสเตชั่นใดอยู่ที่เซ็กเมนท์ใดนั้นทำได้โดยเก็บ



รูปที่ ค-1 บริดจ์ทำหน้าที่เก็บลิสท์ของสเตชั่นแต่ละเซ็กเมนท์ไว้

ฮาร์ดแวร์แอดเดรสของแต่ละเครื่องในแต่ละเซ็กเมนท์จากการสังเกตซอร์สแอดเดรสในแต่ละแพ็คเกจแล้วเพิ่มเข้าไปในตารางของเซ็กเมนท์ที่เหมาะสม

<u>การทำโหลดบาลานซ์โดยใช้เราท์เตอร์</u>

เราท์เตอร์แยกเครือข่ายได้โดยใช้เน็ตเวิร์คแอดเดรสซึ่งเราท์เตอร์จะไม่ยอมให้แพ็คเกจผ่านถ้าเดสทิเนชั่นแอดเดรส(destination address) ของแพ็คเกจนั้นไม่ได้อยู่อีกข้างของเครือข่ายเช่นกัน

การเลือกใช้บริดจ์หรือเราท์เตอร์นั้นให้พิจารณาสิ่งดังต่อไปนี้

1. บริดจ์เร็วกว่าเราท์เตอร์
2. บริดจ์อาจเชื่อมเครือข่ายที่ไม่เหมือนกัน เช่น อีเธอร์เน็ตกับโทเคนริง ไม่ได้
3. บริดจ์ทำหน้าที่แค่ส่งต่อไปถ้าปลายทางไม่ได้อยู่ในเซ็กเมนท์นั้น ขณะที่เราท์เตอร์จะมีตารางเก็บเกี่ยวกับการเชื่อมต่อแบบปลายถึงปลาย เช่น เราท์เตอร์ตัวต่อไปควรจะเป็นตัวไหนเพื่อให้ได้ทางที่สั้นที่สุด

ค.2 การทดสอบการเข้าถึงตัวกลาง(การทดสอบโหลด: Load Test)

ถ้าเราต้องการเพิ่มผู้ใช้ แต่กลัวว่าจะกระทบกับสมรรถภาพของเครือข่ายหรือไม่ เราสามารถทดสอบได้โดยการจำลองโหลดเพิ่มเข้าไปในระบบเพื่อดูว่าจะเกิดผลอะไรขึ้นบ้าง

<u>การทดสอบความเครียดของระบบเคเบิ้ลในเครือข่าย(Stress-Testing)</u>

การทดสอบนี้จะทำให้เราทราบว่าสมรรถภาพจะลดลงมากเพียงไรถ้ามีโหลดเพิ่มขึ้นซึ่งอาจทำได้โดยการทดลองส่งไฟล์ที่โหลดปกติ จากนั้นเพิ่มโหลดแล้วดูว่าเวลาที่ใช้ส่งไฟล์เพิ่มขึ้นมากน้อยเพียงไร

ภาคผนวก ง

การหาสาเหตุความผิดปกติที่เกิดขึ้นที่เลเยอร์ดาต้าลิ้งค์

(Trobleshooting at the Data Link Layer)

ทุกสเตชั่นไม่ได้ส่งเฟรมรูปแบบที่ถูกต้องเสมอไป ทรานซีฟเวอร์(transceivers) การ์ดอินเทอร์เฟสเครือข่าย หรือไดรเวอร์แลนที่ผิดปกติสามารถทำให้เกิดการส่งเฟรมรูปแบบที่ผิดได้ นอกจากนี้ระบบเคเบิ้ลอาจเป็นสาเหตุให้เกิดการเสียหายของเฟรมในสายได้ การวิเคราะห์และแปลความหมายของเฟรมที่เสียหายจะช่วยให้เรารู้ว่าเราใช้งานเครือข่ายมากเกินความสามารถที่จะรองรับได้ของมัน หรือเกิดความผิดปกติในระบบเคเบิ้ล การ์ดอินเทอร์เฟสแลน ไดรเวอร์แลน หรือทรานซีฟเวอร์หรือไม่

ความผิดปกติของเฟรมมี 4 ประเภท

- การชนกันในเซ็กเมนท์เราหรือเซ็กเมนท์อื่น(local and remote collisions)
- ความผิดปกติที่เกิดจากซีอาร์ซี/การมีบางไบต์ไม่ครบ 8 บิต(CRC/alignment errors) และการชนแบบเลท(late collisions)
- ความผิดปกติที่เกิดจากขนาดเฟรม
- แจบเบอร์(Jabber)

ง.1 การมอนิเตอร์การชนกันในเซ็กเมนท์เราหรือเซ็กเมนท์อื่น

ถึงแม้ว่าจะเป็นเรื่องปกติที่จะเกิดการชนกันในแลนแบบนี้ แต่เมื่อไรก็ตามที่เกิดการชนกันมากเกินไป สมรรถภาพของระบบเครือข่ายก็จะตกต่ำลง จากการดูข้อมูลที่เกี่ยวข้องจะทำให้เราทราบได้ว่าการชนกันที่เกิดขึ้นเกิดเพราะเครือข่ายถูกใช้งานมากเกินไป(overload) หรือเกิดมาจากอุปกรณ์ทำงานผิดพลาด ผู้ดูแลระบบควรจะมอนิเตอร์ระบบของพวกเขาเพื่อดูว่าการชนกันในเซ็กเมนท์ของตนและเซ็กเมนท์อื่นมีจำนวนเท่าไรเพื่อนำไปเป็นข้อมูลในการติดตั้งระบบ ในตารางที่ ง-1 จะเปรียบเทียบระหว่างการชนทั้ง 2 แบบ

COLLISION	< 64 BYTES	BAD CRC	CR PAIR TRIGGERED
Local	Yes	Yes	Yes
Remote	Yes	Yes	No

ตารางที่ ง-1 แสดงความแตกต่างของการชนกันในเซ็กเมนท์เราและเซ็กเมนท์อื่น

การชนกันในเซ็กเมนท์เรา(Local Collisions)

แฟรกเมนท์ที่เกิดจากการชนจะมีความยาวน้อยกว่า 64 ไบต์ ซึ่งวงจรตรวจจับการชนกัน(collisions-detection circuitry) บนการ์ดอินเทอร์เฟสแลนหรือทรานซีฟเวอร์จะเป็นตัวที่ตรวจพบการชนแบบนี้ การตรวจจับการชนโดยวงจรตรวจจับการชนนี้ช่วยให้แยกการชนในเซ็กเมนท์เราออกจากการชนในเซ็กเมนท์อื่นได้ เพราะการชนในเซ็กเมนท์อื่นไม่สามารถตรวจพบได้จากวงจรนี้

การชนกันที่เกิดขึ้นในเซ็กเมนท์อื่น(Remote Collisions)

การชนแบบนี้เกิดขึ้นในเซ็กเมนท์อีกข้างของเครือข่ายที่แยกกันด้วยรีพีทเตอร์ (repeater) ซึ่งแฟรกเมนท์จะมีความยาวน้อยกว่า 64 ไบต์และมีซีอาร์ซีไม่ถูกต้อง รีพีทเตอร์จะส่งผ่านแฟรกเมนท์ที่เกิดจากการชนไปยังทุกเซ็กเมนท์ที่ติดต่ออยู่ ในขณะที่บริดจ์และเราท์เตอร์จะไม่ทำเช่นนั้น ถ้าเราพบว่ามีการชนกันจากเซ็กเมนท์อื่นมากเกินไป เราอาจจะต้องทำการติดตั้งบริดจ์เพื่อกรองการชนแบบนี้ออกไป อย่างไรก็ตามเราสามารถลดโหลดในระบบซึ่งมีการชนกันมากจนเกินไปโดยการคอนฟิกระบบเคเบิ้ลใหม่หรือซ่อมแซมอุปกรณ์ที่มีปัญหา

การหาสาเหตุของการชนกันมากจนเกินไปในเซ็กเมนท์เราและเซ็กเมนท์อื่น

แต่ละเครือข่ายจะมีจำนวนการชนกันที่ทำให้สังเกตเห็นว่าสมรรถภาพของระบบตกต่ำลงแตกต่างกัน เมื่อไรก็ตามที่ผู้ใช้เริ่มร้องเรียนเกี่ยวกับสมรรถภาพของระบบ เมื่อนั้นแสดงว่าการชนกันเกิดขึ้นมากจนเกินไปแล้ว เมื่อเราดูสถิติเกี่ยวกับการชน เราจำเป็นจะต้องดูสถิติระดับการใช้งานเครือข่ายด้วย ข้อมูลเหล่านี้จะช่วยเราในการหาสาเหตุที่ทำให้เกิดการชนที่มากจนเกินไปว่าเกิดจากโหลดในเซ็กเมนท์มากเกินไปหรืออุปกรณ์บางตัวทำงานผิดปกติ(เพราะปัญหาจากฮาร์ดแวร์หรือทราฟฟิคมากเกินไป)

เซ็กเมนท์ที่มีโหลดมากเกินไป

ถ้าอัตราการใช้งานเครือข่ายสูงขึ้นจากการเพิ่มขึ้นของโหนดหรือแอพพลิเคชั่นใหม่ และจำนวนการชนเกิดขึ้นสูง ก็หมายความว่าการชนกันเกิดจากการเพิ่มขึ้นของทราฟฟิคบนเซ็กเมนท์นั้น ทางแก้ไขที่เป็นไปได้ทางหนึ่งก็คือการคอนฟิกระบบเคเบิ้ลใหม่โดยใช้บริดจ์หรือเราท์เตอร์(รีพีทเตอร์ไม่สามารถกรองทราฟฟิคได้) ซึ่งก็คือการทำโหลดบาลานซิ่งนั่นเอง

เคเบิ้ลที่ใช้ในเซ็กเมนท์มีความยาวมาก

ถ้าเคเบิ้ลมีความยาวมากกว่าที่กำหนดไว้ในข้อกำหนด(Specification) โหนดที่อยู่ปลายเคเบิ้ลอาจจะคิดว่าเคเบิ้ลว่างแล้วทำการส่ง แต่ในความเป็นจริงแล้วขณะนั้นมีแพ็คเกจวิ่งอยู่ในสาย ซึ่งจะเป็นสาเหตุให้เกิดการชนกันของข้อมูลได้ ในกรณีนี้การชนไม่ได้เกิดมาจากสาเหตุที่ระบบเครือข่ายถูกใช้งานมาก ดังนั้นระดับการใช้งานระบบเครือข่ายจึงไม่สูงขึ้นจากปกติ ถ้าเกิดเหตุการณ์เช่นนี้ให้ลองตรวจสอบความยาวของเคเบิ้ลดูว่ายาวกว่าที่ระบุในข้อกำหนดหรือไม่

การใช้งานมากเกินไปในเซ็กเมนท์อื่น

ถ้าเราพบว่ามีการชนกันจำนวนมากในเซ็กเมนท์อื่น เซ็กเมนท์ที่ต่ออยู่กับเซ็กเมนท์ดังกล่าวโดยผ่านรีพีทเตอร์ก็จะเกิดการชนกันจำนวนมากตามไปด้วย เพื่อเป็นการแก้ไขปัญหานี้เราควรเปลี่ยนมาใช้บริดจ์หรือเราท์เตอร์แทนการใช้รีพีทเตอร์ ถ้าเซ็กเมนท์ใดเกิดการชนกันของเฟรมมาก สาเหตุที่เป็นไปได้ก็อาจจะมาจากฮาร์ดแวร์เสีย เช่น การ์ดแลนหรือทรานซีฟเวอร์ และเพราะว่าสเตชั่นที่เกี่ยวข้อง

ในการชนกันจะพยายามส่งอีก ดังนั้นการมอนิเตอร์การชนกันหลาย ๆ ครั้งและดูว่าสเตชั่นใดที่มักจะทำการส่งหลังจากการชน จะทำให้สรุปได้ว่าสเตชั่นดังกล่าวอาจจะมีฮาร์ดแวร์ที่เสียซึ่งทำให้เกิดการชนแบบเลท

ง.2 การมอนิเตอร์การชนแบบเลทและความผิดพลาดที่เกิดขึ้นจากซีอาร์ซี/การมีบางไบต์ไม่ครบ 8 บิต

ความผิดพลาดแบบนี้จะบ่งชี้ถึงปัญหาที่เกิดจากเคเบิ้ลหรือคอมโพเนนท์(component) ในระบบเครือข่ายมีความผิดปกติ

<u>การชนกันแบบเลท</u>

คือการที่แพ็คเกจมีขนาดมากกว่า 64 ไบต์และมีซีอาร์ซีผิด และวงจรตรวจจับการชนกันก็จะตรวจพบได้ถ้าเกิดการชนแบบนี้ในเซ็กเมนท์เรา การชนแบบเลทนี้พิจารณาจากที่เกิดในเซ็กเมนท์เราเท่านั้น ถ้าแพ็คเกจลักษณะนี้ถูกส่งมาจากเซ็กเมนท์อื่นข้ามเราท์เตอร์ แพ็คเกจดังกล่าวจะถือว่ามีความผิดพลาดที่ซีอาร์ซีหรือมีบางไบต์ไม่ครบ 8 บิต

การชนแบบเลทนี้ถือว่าเป็นความผิดปกติในแลนอีเธอร์เน็ต ซึ่งบ่งถึงว่าแม้สเตชั่นจะส่งจำนวนไบต์ที่เพียงพอ คือ 64 ไบต์แล้วก็ตาม แต่สเตชั่นอื่นก็ยังทำการส่งข้อมูลออกมาด้วยเช่นกัน หมายความว่าถ้าทุกโหนดทำตามข้อกำหนดของซีเอสเอ็มเอ/ซีดี การชนแบบเลทก็จะไม่เกิดขึ้น

<u>ข้อผิดพลาดจากซีอาร์ซีและแพ็คเกจที่มีบางไบต์ไม่ครบ 8 บิต</u>

แพ็คเกจจะมีข้อผิดพลาดในซีอาร์ซีเมื่อค่าที่เก็บในฟิลด์เอฟซีเอส(FCS) ไม่ตรงกับค่าที่สเตชั่นปลายทางคำนวณได้ ส่วนแพ็คเกจที่ไม่สิ้นสุดเป็นจำนวนครบไบต์จะเป็นแพ็คเกจที่มีบางไบต์ไม่ครบ 8 บิต ตารางที่ ง-2 แสดงถึงข้อแตกต่างระหว่างข้อผิดพลาดทั้งสอง

ERROR TYPE	< 64 BYTES	BAD CRC	CD PAIR TRIGGERED
Late collisions	No	Yes	Yes
CRC/alignment	No	Yes	No

ตารางที่ ง-2 แสดงข้อแตกต่างระหว่างการชนแบบเลทกับข้อผิดพลาดที่ซีอาร์ซีและเฟรมมีบางไบต์ไม่ครบ 8 บิต

การหาสาเหตุของการเกิดการชนแบบเลทและความผิดพลาดจากซีอาร์ซีรวมทั้งการที่แพ็คเกจมีบางไบต์ไม่ครบ 8 บิต

มีสาเหตุหลักอยู่ 2 สาเหตุคือ ปัญหาจากสายเคเบิ้ลและปัญหาจากคอมโพเนนท์

<u>ปัญหาจากสายเคเบิ้ล</u>

ปัญหาเช่น การช็อต(short) หรือสัญญาณรบกวน(noise) ที่เกิดขึ้นจากการรบกวนของสัญญาณแม่เหล็กไฟฟ้ามักจะทำให้เกิดข้อผิดพลาดในซีอาร์ซีและการมีบางไบต์ไม่ครบ 8 บิต ส่วนการชนแบบเลทมักจะเกิดขึ้นเพราะการต่อสายเคเบิ้ลไม่เหมาะสมคือไม่ถูกต้องตามข้อกำหนด หรือโหนดที่ไม่รับรู้(deaf

node) เพราะส่วนตรวจจับสัญญาณบนสายเสีย ถ้าเราพบว่ามีปัญหาจากการชนกันแบบเลทและข้อผิดพลาดที่ซีอาร์ซีรวมทั้งแพ็คเกจมีบางไบต์ไม่ครบ 8 บิตเกิดขึ้นมากให้ลองตรวจสอบสิ่งดังต่อไปนี้

- เซ็กเมนท์ยาวจนเกินไป ทำให้โหนดที่อยู่ปลายข้างหนึ่งส่งแพ็คเกจออกมา โดยรู้ว่าโหนดที่ปลายอีกข้างหนึ่งได้ครอบครองสายส่งข้อมูลอยู่แล้ว
- เคเบิ้ลหรือสายส่งข้อมูลเสียหาย ข้อมูลเดินทางผ่านสายที่เกิดการช็อตหรือเสียหายซึ่งจะทำให้ข้อมูลดังกล่าวผิดเพี้ยนไปก่อนถึงปลายทาง
- กราวด์เซ็กเมนท์ไม่ถูกต้อง ทำให้เกิดการเหนี่ยวนำสัญญาณรบกวนมาทำความเสียหายให้แก่ข้อมูลได้
- สิ้นสุดเซ็กเมนท์ไม่เหมาะสม ทำให้สัญญาณถูกดูดกลืนที่ปลายของเซ็กเมนท์ และสัญญาณบางส่วนก็อาจจะสะท้อนกลับและชนเข้ากับสัญญาณในสายขณะนั้นได้
- แท็บ(Taps) ใกล้กันเกินไป ควรทำตามที่กำหนดไว้ในข้อกำหนด
- สายมีสัญญาณรบกวน ทำให้สัญญาณข้อมูลผิดเพี้ยนและทำให้เกิดข้อผิดพลาดที่ซีอาร์ซีและการที่แพ็คเกจมีบางไบต์ไม่ครบ 8 บิต

ปัญหาจากคอมโพเนนท์

คอมโพเนนท์ที่ทำงานผิดพลาดสามารถทำให้เกิดข้อผิดพลาดได้ทั้งซีอาร์ซีและการมีบางไบต์ไม่ครบ 8 บิต ปัญหาที่เกิดจากคอมโพเนนท์ในระบบเครือข่ายโดยทั่วไปมี 2 แบบคือ

- โหนดที่ไม่รับรู้ สเตชั่นที่ไม่สามารถรู้ว่ามีกิจกรรมอยู่บนสายส่งข้อมูลขณะนั้นได้คือโหนดที่ไม่รับรู้ ซึ่งแก้ได้โดยการเปลี่ยนการ์ดแลนหรือทรานซีฟเวอร์
- รีพีทเตอร์, ทรานซีฟเวอร์, หรือการ์ดแลนทำงานผิดพลาด จะก่อกวนสัญญานในระบบเครือข่าย ส่งสัญญาณที่ผิดปกติลงไปในสายสัญญาณ หรือไม่สนใจแพ็คเกจที่เข้ามา การแก้ไขก็คือทำการเปลี่ยนอุปกรณ์ที่ทำงานผิดพลาด

ง.3 การมอนิเตอร์ความผิดพลาดที่เกิดจากขนาดของเฟรม

จากที่ทราบกันอยู่แล้วว่าเฟรมในอีเธอร์เน็ตต้องมีความยาวอยู่ระหว่าง 64-1518 ไบต์ซึ่งรวมเฮดเดอร์และเอฟซีเอสแล้ว ดังนั้นเฟรมที่มีความยาวน้อยกว่าหรือมากกว่าความยาวดังกล่าวแต่มีซีอาร์ซีที่ถูกต้องจะถือว่ามีความผิดปกติที่ความยาว

การหาสาเหตุของความผิดปกติในขนาดเฟรม

เฟรมที่มีขนาดผิดปกติเกิดขึ้นเพราะสเตชั่นที่ทำการส่งมันออกมา และเราสามารถหาสเตชั่นนั้นได้ง่ายเพราะเฟรมมีรูปแบบถูกต้องและบรรจุแอดเดรสต้นทางที่เฮดเดอร์ เพียงแค่ผิดปกติตรงความยาวของเฟรมเท่านั้น

ปกติเฟรมที่มีความยาวผิดปกติจะเกิดจากไดรเวอร์แลนทำงานผิดพลาด ซึ่งแก้โดยตรวจสอบเวอร์ชั่นของไดรเวอร์และเปลี่ยนมันถ้ามันเก่าเกินไป แต่ถ้าไดรเวอร์เป็นเวอร์ชั่นล่าสุดแล้วก็อาจเป็นไปได้ที่ไฟล์ของมันจะถูกทำให้เสียหาย เราท์เตอร์ก็สามารถทำให้เกิดขนาดเฟรมผิดปกติได้ ถ้าเราท์เตอร์เชื่อมต่ออยู่กับ 2 ระบบเครือข่ายที่ไม่เหมือนกันและไม่ได้ส่งขนาดเฟรมที่ถูกต้องให้กับแต่ละข้าง

ง.4 การมอนิเตอร์แจบเบอร์(Jabber)

แจบเบอร์คือเฟรมที่มีขนาดมากกว่า 1518 ไบต์และมีซีอาร์ซีผิด ซึ่งเกี่ยวข้องกับการที่ทรานซีฟเวอร์ทำงานผิดพลาด โดยข้อกำหนดแล้ว ทรานซีฟเวอร์จะส่งข้อมูลได้ครั้งละ 150 มิลลิวินาทีเท่านั้นซึ่งเพียงพอที่จะส่งข้อมูลจำนวน 1518 ไบต์ได้ แต่ถ้าทรานซีฟเวอร์ไม่หยุดส่งข้อมูลหลังจาก 1518 ไบต์ก็จะเกิดแจบเบอร์ขึ้น ถ้าเราคิดว่าทรานซีฟเวอร์จะมีอาการดังกล่าวให้สังเกตที่ไฟกะพริบข้างนอกว่ามันยังคงส่งต่อไปหรือไม่ ถ้าเป็นไปตามที่คิดไว้ก็ให้เปลี่ยนทรานซีฟเวอร์นั้นเพื่อจะได้ไม่กระทบกับประสิทธิภาพของระบบเครือข่าย

ERROR	TRIGGERING CONDITION	POSSIBLE CAUSE
Lacal collisions	Less than 64 bytes Bad CRC CD triggered	Overloaded segment Cable segment too longA
Remote collisions	Less than 64 bytes Bad CRC	Remote segment overload Remote cable segment too long
Late collisions	Equal to or greater than 64 bytes Bad CRC CD triggered	Deaf node Segment too long Component problem
CRC/alignment	From 64 to 1518 bytes long Bad CRC Doesn't end on 8-bit boundary	Failing cable Noisy cable Component problem
Short frames	Less than 64 bytes Good CRC	Faulty LAN driver
Long frames	Greater than 1518 bytes Good CRC	Faulty LAN driver
Jabber	Greater than 1518 bytes Bad CRC	Faulty transceiver

ตารางที่ ง-3 แสดงเงื่อนไขของข้อผิดพลาดต่าง ๆ

ภาคผนวก จ

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] Laura A. Chappell , Dan E. Hakes : *"Novell's Guide to NetWare LAN Analysis"* , Nevell Press , 2nd Edition

[2] Kevin Washburn , Jim Evans : *"TCP/IP running a successful network"* , Addison-Wesley , 2nd Edition

[3] พ.อ. เจนวิทย์ เหลืองอร่าม (2538) : *"การเขียนโปรแกรมแบบ OOP ด้วย Borland C++"*

[4] Craig Hunt : *"TCP/IP Network Administration"* , O'Reilly & Associates Inc. , 1992

[5] Ian Sommerville : "Software Engineering" , Addison-Wesley , 5th Edition