โปรแกรม การตรวจจับและวิเคราะห์แพคเก็ตข้อมูลที่รับส่งภายในเครือข่ายคอมพิวเตอร์

โดย

นาย มาโนช พิชผล

นาย ศรายุทธ จันทร์เมืองไทย

ปริญญานิพนธ์เล่มนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

NETWORK MONITORING TOOL

Mr. MANOCH PICHPOL

Mr. SARAYUT CHANMUNGTHAI

Project Report Submitted in Partial Fulfillment of the requriment

For the Bachelor's Degree

Department of Industrial Technology

Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

หัวข้อปริญญานิพนธ์	การตรวจจับ และวิเคราะห์แพคเก็ตข้อมูลที่รับส่งภายในเครือข่ายคอมพิวเตอร์
โดย	นาย มาโนช พิชผล 41012019 นาย ศรายุทธ จันทร์เมืองไทย 41012023
ภาควิชา	เทคนิคอุตสาหกรรม
อาจารย์ที่ปรึกษา	อาจารย์ มยุรี เลิศเวชกุล อาจารย์ พิทักษ์ ธรรมวาริน
ปีการศึกษา	2542

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตร อุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

..... อาจารย์ที่ปรึกษา
( )
..... กรรมการ
( )
..... กรรมการ
( )
..... กรรมการ
( )
..... กรรมการ
( )

ลิขสิทธิ์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Project Report	NETWORK MONITORING TOOL
By	Mr. Manoch Pichpol 41012019
	Mr. Sarayut Chanmuangthai 41012023
Department of	Industrial Technology
Advisor	Assoc. Mayuree Lertwatchakul
	Assoc. Pitak Thammawarin
Year	1999

Accepted by the Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang in partial Fulfillment for the Bachelor' degree

Project Report Committee

..... Chairman
( )

..... Committee
( )

..... Committee
( )

..... Committee
( )

..... Committee
( )

สารบัญ

สารบัญ (ต่อ)

สารบัญภาพ

หน้า

หน้า

สารบัญตาราง

หน้า

หัวข้อปริญญานิพนธ์	การตรวจจับ และวิเคราะห์แพคเก็ตข้อมูลที่รับส่งภายในเครือข่ายคอมพิวเตอร์
โดย	นาย มาโนช พิชผล 41012019 นาย ศรายุทธ จันทร์เมืองไทย 41012023
ภาควิชา	เทคนิคอุตสาหกรรม
อาจารย์ที่ปรึกษา	อาจารย์ มยุรี เลิศเวชกุล อาจารย์ พิทักษ์ ธรรมวาริน
ปีการศึกษา	2542

บทคัดย่อ

ปัจจุบัน การใช้งานคอมพิวเตอร์ในรูปแบบเครือข่าย ได้เข้ามามีบทบาทอย่างมาก โดยเฉพาะองค์กรใหญ่ที่ต้องการมีการแบ่งใช้ทรัพยากรร่วมกัน การปรับปรุงระบบคอมพิวเตอร์ให้มีประสิทธิภาพเพิ่มขึ้นจึงเป็นสิ่งที่สำคัญยิ่ง ด้วยเหตุนี้ จึงได้ทำการพัฒนาโปรแกรมขึ้น เพื่อใช้เฝ้าดูและวิเคราะห์แพคเก็ต ในระบบเครือข่ายคอมพิวเตอร์ท้องถิ่น โดยทำการตรวจสอบแพคเก็ตผ่านการทำงานของแพคเก็ตไดร์เวอร์ เข้ามาวิเคราะห์ในด้านต่าง ๆ เช่น ความหนาแน่น (traffic) และทิศทาง , โพรโตคอลที่ใช้ รวมไปถึงการเก็บข้อมูลในเชิงสถิติเพื่อประโยชน์ทางการปรับปรุงระบบคอมพิวเตอร์ในอนาคต

นอกจากนี้ ยังเป็นโครงงานสำหรับช่วยในการศึกษาการทำงานภายในเครือข่าย เพื่อให้เข้าใจถึงการทำงานของโพรโตคอลในแต่ละชั้นด้วย

Project Report NETWORK MONITORING TOOL

By Mr. Manoch Pitpol 41012019
Mr. Sarayut Chanmuangthai 41012023

Department of Industrial Technology

Advisor Assoc. Mayuree Lertwatchakul
Assoc. Pitak Thammawarin

Year 1999

Abstract

Nowadays, many organizations use the computers in networking environment for sharing the resources. Performance for improvement and maintenance is very important. In order to do this, the Network Monitoring and analysis data packet program are developed. The program receives captured packets from packet driver . The input could be filter by frame type, protocol or address and then save into file or display the analytic result in real time mode.

บทที่ 1 บทนำ

ความสำคัญ และที่มา

ในยุคโลกาภิวัฒน์ ระบบเครือข่ายคอมพิวเตอร์ ได้เข้ามามีบทบาทในด้านต่าง ๆ มากมาย ไม่ว่าจะด้านการดำรงชีวิต ด้านการศึกษา หรือทางด้านธุรกิจ ดังจะเห็นได้จากโครงการต่าง ๆ เช่น ระบบการศึกษาทางไกลผ่านเครือข่ายคอมพิวเตอร์ ระบบอินเตอร์เนต ที่ช่วยให้การศึกษาค้นคว้าในปัจจุบันเป็นไปได้อย่างสะดวกรวดเร็ว และมีประสิทธิภาพ การรื้อปรับระบบองค์กร หรือที่เราได้ยินกันในชื่อว่า การรีเอนจิเนียริง การทำระบบสำนักงานอัตโนมัติ ระบบจดหมายอิเล็กทรอนิกส์ (E – mail) ก็ล้วนแล้วแต่เกี่ยวข้องกับระบบคอมพิวเตอร์ ด้วยกันทั้งสิ้น

ดังนั้นเมื่อมีการใช้ระบบเครือข่ายคอมพิวเตอร์กันอย่างแพร่หลาย จึงเกิดปัญหาเกี่ยวกับการใช้งานระบบเครือข่ายขึ้น ด้วยเหตุนี้การปรับแต่งและแก้ไขระบบคอมพิวเตอร์ ให้มีประสิทธิภาพสูงสุดจึงเป็นสิ่งที่จำเป็นยิ่ง

วัตถุประสงค์ของโครงงาน

1. เพื่อพัฒนาซอร์ฟแวร์ (Software) ที่สามารถตรวจสอบและเฝ้าดูระบบเครือข่ายคอมพิวเตอร์ เพื่อความสะดวกในการจัดการเครือข่าย โดยผู้ดูแลระบบ
2. เพื่อศึกษา การทำงานของโพรโตคอลในแต่ละชั้นของ OSI โมเดล

ขอบเขตของโครงงาน

1. เป็นซอร์ฟแวร์คอมพิวเตอร์ที่ทำงานบนระบบเครือข่ายท้องถิ่น (LAN)
2. สามารถตรวจสอบ เฝ้าดู และดักจับแพคเก็ตเพื่อนำมาวิเคราะห์ โดยวิเคราะห์โพรโตคอลหลัก ๆ ในแต่ละชั้น ดังนี้

2.1 ดาต้าลิงค์เลเยอร์ (Datalink Layer)

- ไออีอีอี 802.2 (IEEE 802.2)
- ไออีอีอี 802.3 (IEEE 802.3)
- อีเทอร์เนต ทู (Ethernet II)
- อีเทอร์เนตสแนบ (Ethernet Snap)

2.2 เน็ตเวอร์กเลเยอร์ (Network Layer)

- ไอพี (IP)
- ไอพีเอ็กซ์ (IPX)
- เออาร์พี (ARP)

2.3 ทรานสปอร์ตเลเยอร์ (Transport Layer)

- ทีซีพี (TCP)
- ยูดีพี (UDP)
- ไอซีเอ็มพี (ICMP)
- เอสพีเอ็กซ์ (SPX)
- เอ็นซีพี (NCP)

2.4 อัพเปอร์เลเยอร์ (Upper Layer)

- เทลเนต (Telnet)
- เอฟทีพี (FTP) เอฟทีพีดาต้า (FTP_DATA)
- อาร์ล็อกอิน (RLOGIN)
- อื่น ๆ

3. สามารถกรองแพคเก็ตในรูปแบบของอีเทอร์เน็ตชนิดต่างๆ ก่อนที่จะแสดงผลหน้าจอ โดยมีชนิดของอีเทอร์เน็ตดังนี้
   - Ethernet II
   - Ethernet 802.2
   - Ethernet 802.3
   - Ethernet Snap
4. วิเคราะห์โพรโตคอล โดยการถอดรหัสแพคเก็ตในแต่ละเลเยอร์ แล้วนำมาแสดงผลบนหน้าจอคอมพิวเตอร์ได้
5. สามารถตรวจสอบโฮสต์ว่ามีการตอบสนองการใช้งานหรือไม่
6. สามารถเก็บข้อมูลการใช้งานของเครือข่ายโดยรวมซึ่งประกอบด้วย ขนาดของแพคเก็ตและข้อมูลทั้งหมด

7. สามารถแสดงผลข้อมูลขณะดักจับแพคเก็ตในรูปแบบต่างๆได้ดังนี้
   - แสดงจำนวนแพคเก็ตต่อหน่วยเวลา (packet per second)โดยสามารถกำหนดเวลาได้
   - แสดงปริมาณข้อมูลต่อหน่วยเวลา (quantity per second)โดยสามารถกำหนดเวลาได้
   - แสดงหมายเลขสถานีต้นทาง,หมายเลขสถานีปลายทาง,ชนิดของอีเทอร์เน็ตและเวลาที่ดักจับแพคเก็ตนั้นได้
   - แสดงปริมาณโพรโตคอลในชั้นอัพเปอร์เลเยอร์ (Upper Layer) ที่ใช้งานอยู่จริง
8. สามารถเฝ้าดูการรับส่งข้อมูลจากเครือข่ายคอมพิวเตอร์ เครื่องใดเครื่องหนึ่งโดยเฉพาะได้

ประโยชน์ที่คาดว่าจะได้รับ

1. เป็นเครื่องมืออำนวยความสะดวกสำหรับผู้ดูแลระบบในการตรวจสอบความหนาแน่น (traffic) ของเครือข่ายคอมพิวเตอร์
2. ประหยัดทรัพยากรบุคคลในการวิเคราะห์ความต้องการใช้งานเครือข่ายคอมพิวเตอร์เพื่อปรับปรุงระบบเครือข่ายให้เหมาะสม

รูปแบบที่ใช้ในปริญญานิพนธ์

ตัวเลขในปริญญานิพนธ์เล่มนี้จะเขียนในรูปแบบเฉพาะ โดยเลขฐานสิบจะเขียนในรูปแบบปกติ เช่น 12 ส่วนเลขฐานสิบหก เขียนในรูป "0x" เช่น 0x0B ทุกๆ ค่าที่อ้างอิงกับเนตเวอร์กฮาร์ดแวร์แอดเดรส (ต้นทาง ปลายทาง) จะเขียนในรูปแบบเลขฐานสิบหกโดยไม่มีตัวนำ เช่น AA–BB–CC–DD–EE–FF

บทที่ 2 ทฤษฎีและหลักการ

2.1 ระบบเครือข่ายคอมพิวเตอร์ตามมาตรฐาน OSI

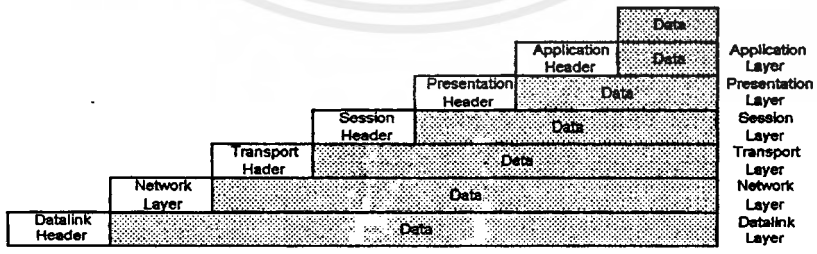
ในยุคต้น ๆ ของระบบเครือข่ายคอมพิวเตอร์ ระบบเครือข่ายมีให้เลือกเพียงไม่กี่ชนิดจากผู้ผลิต ได้แก่ IBM , HP, DEC, HoneyWell อีกทั้งแต่ละระบบยังไม่สามารถเชื่อมต่อกันได้เพราะเป็นแบบปิด (Proprietary) ดังนั้นเมื่อเลือกซื้อระบบใด ผู้ซื้อจะต้องผูกติดกับผู้ขายทำให้เกิดปัญหาหลาย ๆ อย่างตามมา

จึงทำให้ระบบ OSI ได้รับการพัฒนาขึ้นในปี ค.ศ. 1978 โดย **International Organize of Standard (ISO)** เพื่อให้มีมาตรฐานในรูปแบบของระบบเปิด(Open System) และเป็นตัววัดความแตกต่างระหว่างระบบการติดต่อสื่อสาร โดยระบบเครือข่ายใดที่พัฒนากันกับขอบเขตข้อบังคับของ OSI เสมือนกับระบบเหล่านั้นพูดจาภาษาเดียวกัน กล่าวคือใช้การเชื่อมต่อแบบเดียวกัน ทำให้ระบบเครือข่าย จากหลายผู้ผลิตสามารถเชื่อมต่อกันได้ ทำให้มีทางเลือกเพิ่มขึ้น อีกทั้งระบบยังจะมีความยืดหยุ่นสูง เพิ่มการแข่งขันทำให้มีราคาถูกลง

รูปแบบมาตรฐาน OSI ได้กำหนดโดยใช้หลักการ 5 อย่างดังนี้ คือ

1. ชั้น (Layer) จะถูกสร้างขึ้นก็ต่อเมื่อมีความแตกต่างกัน
2. แต่ละชั้นจะต้องถูกกำหนดหน้าที่ไว้อย่างชัดเจน
3. หน้าที่ของแต่ละชั้นจะต้องนิยามได้ในโพรโตคอลมาตรฐาน
4. ขอบเขตของแต่ละชั้นจะต้องมีข้อมูลไหลข้ามผ่านน้อยที่สุด
5. หน้าที่ที่แตกต่างกันจะถูกกำหนดแยกกันในแต่ละชั้น แต่จำนวนชั้นนั้นจะต้องไม่มากเกินกว่าที่สถาปัตยกรรม จะรองรับได้

รูปแบบมาตรฐาน OSI ประกอบไปด้วยชั้น ต่างๆ ดังต่อไปนี้



รูปที่ 2.1 แสดงการใช้งานเฮดเดอร์ในระดับชั้น ต่าง ๆ

ชั้นที่ 1 ชั้นฟิสิคัล (Physical Layer)

ในชั้นฟิสิคัล จะเป็นการส่งและรับบิตข้อมูล ผ่านสื่อกลางที่ใช้สื่อสาร หน่วยของข้อมูลในชั้นนี้จะมีหน่วยเป็น บิต โดยสนใจเพียงการรับและการส่งเท่านั้น จะไม่สนใจในความหมายของตัวกลุ่มบิต โดยจะอธิบายรายละเอียดทางกลศาสตร์และหลักการในการส่งบิต อาทิเช่น การกำหนดลักษณะขาสัญญาณของอุปกรณ์ที่ใช้ในการเชื่อมต่อ ขนาดแรงดันไฟฟ้าของสัญญาณที่ใช้

คุณสามารถเทียบการทำงานในชั้นนี้กับการส่งโทรเลข ที่ประกอบด้วยข้อมูลที่เป็นจุดและขีด ซึ่งทั้งจุดและขีดจะเป็นอิสระต่อกัน ไม่สามารถเข้าใจความหมายได้ หากยังไม่รับการแปลความหมายเสียก่อน ซึ่งนั่นจะเป็นหน้าที่ของระดับชั้นที่อยู่สูงขึ้นไป

โพรโตคอลที่รู้จักกันดีในชั้นนี้คือ **RS-232**

ชั้นที่ 2 ชั้นดาตาลิงค์ (Data Link Layer)

ชั้นดาตาลิงค์ จะรับข้อมูลที่เป็น '1' และ '0' จากชั้นฟิสิคัล มาจำแนกออกเป็นกลุ่มของบิต โดยจะเรียกกลุ่มของบิตเหล่านี้ว่า **เฟรม (FRAME)** ในชั้นดาตาลิงค์จะรวมกฎเกณฑ์การควบคุมการขอใช้ระบบเครือข่าย คือเมื่อไรที่เครื่องลูกข่ายสามารถส่งข้อมูลได้ จะจัดการอย่างไรในกรณีโหนดเครื่องลูกข่ายล้ม จะตรวจสอบความผิดพลาดที่เกิดขึ้นได้อย่างไร

เราเปรียบเทียบการทำงานของชั้นดาตาลิงค์ได้กับการแปลความหมายข้อมูลจุดและขีดในโทรเลข ซึ่งจะต้องแบ่งกลุ่มข้อมูลออกเป็นกลุ่มๆ เพื่อใช้แทนความหมายของคำหรือประโยค ในชั้นนี้จะมีการเพิ่มส่วนเฮดเดอร์ลงไปในตัวข้อมูล โดยมักจะเป็นหมายเลขที่อยู่ (address) ของทั้งฝ่ายส่งและฝ่ายรับข้อมูล มาตรฐานโทเคนริงและอีเทอร์เนตก็มีการเพิ่มเฮดเดอร์ลงไปในลักษณะนี้ หมายเลขที่อยู่นี้จะใช้เพื่อการค้นหาเส้นทางให้กับเฟรมข้อมูล และเพื่อให้เครื่องปลายทางทราบแหล่งที่มาของข้อมูลชุดนั้น หมายเลขที่อยู่ในชั้นนี้มักเป็น *หมายเลขที่อยู่ทางฟิสิคัล (Physical Address)* เพราะมันได้มาจากตัวฮาร์ดแวร์ซึ่งกำหนดโดยบริษัทผู้ผลิตเนตเวอร์กการ์ดนั้น ๆ และเก็บถาวรอยู่บน ROM ของการ์ดนั่นเอง หมายเลขที่อยู่นี้จะเป็นหมายเลขเฉพาะของแต่ละเนตเวอร์กการ์ด จะไม่มีหมายเลขใดที่ซ้ำซ้อนกันเลย

ชั้นที่ 3 ชั้นเนตเวอร์ก (Network Layer)

ในชั้นเนตเวอร์ก รับหน้าที่เกี่ยวกับการค้นหาเส้นทางเพื่อส่งข้อมูลในระบบเครือข่ายคอมพิวเตอร์ที่มีความซับซ้อน หน่วยข้อมูลที่เรียกกันในชั้นนี้คือ **แพคเก็ต (Packet)**

ในระบบเครือข่ายที่ไม่ซับซ้อน หมายเลขที่อยู่ทางฟิสิคัลก็เพียงพอแล้วในการติดต่อ แต่ถ้าหากระบบเครือข่ายมีความซับซ้อนมากขึ้น จำเป็นจะต้องมีข้อมูลเพิ่มขึ้น เพื่อช่วยให้การทำงานสะดวกและมีประสิทธิภาพมากขึ้น ระบบเครือข่ายที่มีความซับซ้อนเรียกกันว่า *อินเตอร์เนตเวอร์ก (Internetwork) หรือ อินเตอร์เนต (Internet)*

ในอินเตอร์เนต จะมีการกำหนด *หมายเลขที่อยู่แบบลอจิคัล (Logical Address)* สำหรับแต่ละระบบเครือข่ายย่อย โดยการกำหนดหมายเลขนี้เป็นหน้าที่ของชั้นเนตเวอร์ก เช่นเนตแวร์จะกำหนดหมายเลขที่อยู่แบบลอจิคัลของเครือข่ายย่อยของแต่ละเครือข่าย โดยใช้เลขฐานสิบหกจำนวนแปดหลัก

ในชั้นเนตเวอร์ก จะรับผิดชอบหน้าที่การค้นหาเส้นทางให้กับตัวแพคเก็ต ซึ่งจะทำให้ในการทำงานในระดับชั้นที่สูงขึ้นไป ไม่ต้องกังวลในส่วนนี้อีก ไม่ต้องสนใจรายละเอียดว่าแต่ละแพคเก็ตจะเดินทาง จากตำแหน่งที่อยู่ต้นทางไปยังที่อยู่ปลายทางได้อย่างไร

เปรียบเทียบกับระบบโทรเลข ในกรณีที่เป็นการส่งโทรเลขข้ามประเทศ จำเป็นต้องส่งข้อมูลในโทรเลขนั้นผ่านสถานีย่อยไปเรื่อยๆ จากต้นทางจนถึงปลายทาง ผู้ส่งเพียงระบุปลายทางที่ชัดเจนให้กับเจ้าหน้าที่โทรเลขของสถานีต้นทาง หลังจากนั้นก็ปล่อยให้เป็นหน้าที่ของเจ้าหน้าที่ว่าจะส่งต่อข้อมูลดังกล่าว ผ่านไปยังสถานีไหน ข้อมูลนั้นก็จะถูกส่งผ่านไปเรื่อย ๆ ตามสถานีย่อยระหว่างทาง จนถึงปลายทาง

ตัวอย่างโพรโตคอลที่รู้จักกันในชั้นนี้ คือ X.25

ชั้นที่ 4 ชั้นทรานสปอร์ต (Transport layer)

รับผิดชอบด้านความน่าเชื่อถือ (Reliability) ในการส่งข้อมูลของโพรเซส (Process) ที่ทำงานกันระหว่างเครื่องคอมพิวเตอร์ต้นทางและปลายทาง ย้ำว่าเป็นการติดต่อกันเองของโพรเซสไม่ใช่เป็นการติดต่อกันกับหมายเลขของเนตเวอร์ก ข้อมูลที่อยู่ในชั้นนี้มีหน่วยเป็น **เซกเมนต์ (Segment)**

การทำงานในชั้นนี้จะตรวจสอบข้อมูล ไม่ให้เกิดความผิดพลาด, ลำดับการส่งข้อมูลจะต้องถูกต้อง, ไม่มีข้อมูลใด ๆ สูญหาย หรือซ้ำซ้อน ความน่าเชื่อถือนี้ไม่ได้หมายถึงจะป้องกันไม่ให้เกิดความผิดพลาดขึ้น แต่หมายถึงการที่เมื่อเกิดมีความผิดพลาดขึ้นแล้ว สามารถรับรู้ได้ทันที ซึ่งจะต้องทำการแก้ไขที่ระดับชั้นนี้ ไม่เช่นนั้นแล้วถ้าผ่านขึ้นไปในระดับชั้นที่สูงขึ้นไปจะทำให้ได้รับข้อมูลที่ผิด ๆ ไป

ระดับชั้นทรานสปอร์ตนี้รับผิดชอบในการนำเอาชุดข้อมูลที่มีความยาวมาก ๆ มาตัดแบ่งให้มีขนาดพอเหมาะพร้อมที่จะผ่านไปให้ระดับชั้นเนตเวอร์กทำงานได้ รวมถึงควบคุมลำดับการส่งข้อมูล การแก้ไขข้อผิดพลาด การจัดลำดับข้อมูลใหม่ และส่งสัญญาณตอบรับเมื่อได้รับข้อมูลจากเครื่องที่กำลังติดต่ออยู่

เจ้าหน้าที่ที่โทรเลข จะเป็นผู้ตรวจสอบว่าเกิดความผิดพลาดในการส่งโทรเลขหรือไม่ อาทิ เช่น เนื้อความไม่สมบูรณ์ หรือไม่ได้รับเครื่องหมายที่ใช้แสดงถึงการสิ้นสุดเนื้อข้อมูล เจ้าหน้าที่จะต้องบอกให้ทำการส่งข้อมูลชุดนั้น ๆ ใหม่

ชั้นที่ 5 ชั้นเซสชัน (Session Layer)

ในชั้นนี้จะดูแลการติดต่อกันของเครื่องคอมพิวเตอร์ 2 เครื่องในเรื่องของ *การสร้าง (Establishing) , การสอดคล้อง (Synchronizing) และการบอกเลิกการติดต่อ (Terminating)*

การที่คนเราจะพูดคุยสนทนากันได้นั้น จำเป็นที่จะต้องมีการสร้างข้อกำหนดเช่น กำหนดว่าจะใช้ภาษาใดเป็นสื่อกลาง และจะต้องใช้มารยาททางการสื่อทางที่ถูกต้อง เมื่อสนทนาเสร็จเรียบร้อยแล้ว จะต้องบอกกล่าวเลิกการสนทนา

เจ้าหน้าที่โทรเลข จะปฏิบัติตามกฎเกณฑ์ของการติดต่อที่วางเอาไว้อย่างเคร่งครัด ตั้งแต่ตอนเริ่มส่งข้อมูล ขณะส่งข้อมูล และเมื่อส่งข้อมูลแล้วเสร็จ

การสื่อสารของคอมพิวเตอร์ที่มีลักษณะคล้ายคลึงกับคนเรา ที่จะต้องมีการสร้างการติดต่อก่อน ตกลงกันว่าจะใช้โพรโตคอลอะไร รูปแบบการสื่อสารแบบไหน หากเกิดความผิดพลาดขึ้นแล้ว จะต้องแก้ไขอย่างไร เมื่อไม่ต้องการสื่อสารแล้ว ก็มีขั้นตอนการบอกเลิกการสื่อสารเช่นเดียวกัน

การสื่อสารทำได้ทั้งแบบ 2 ทิศทางพร้อมกัน (Full – duplex) หรือทีละทิศทาง (Half-duplex) ระดับชั้นเซสชัน จะรับผิดชอบดูแลให้การสื่อสารระหว่างต้นทางและปลายทางเป็นไปอย่างสอดคล้องกัน

โพรโตคอลที่รู้จักกันดีในชั้นนี้คือ Remote Procedure Calls (RPCs) โดยจะนำไปใช้กับทั้งแนตแวร์และชุดโพรโตคอลอื่นๆ อีกมากมาย รวมถึงชุดอินเตอร์เนตโพรโตคอลด้วย

ชั้นที่ 6 ชั้นพรีเซนเตชั่น (Presentation Layer)

ระดับชั้นพรีเซนเตชันจะดูแลให้การสื่อสารระหว่างเครื่องคอมพิวเตอร์ 2 เครื่องให้ถูกต้องตามกฎเกณฑ์ที่วางเอาไว้ งานที่รับผิดชอบส่วนใหญ่จะเป็นการแปลงข้อมูลที่ได้รับจากระดับล่างซึ่งเป็นรูปแบบมาตรฐานไปเป็นรูปแบบข้อมูลที่คอมพิวเตอร์หรือแอพพลิเครชัน นั้น ๆ นำไปใช้ได้

ชั้นพรีเซนเตชัน เปรียบเทียบได้กับการที่มนุษย์มี ล่าม ทำหน้าที่แปลภาษาหนึ่งมาเป็นภาษาที่สามารถเข้าใจได้ หน้าที่อื่น ๆ ก็มี อาทิเช่น *การเข้ารหัสข้อมูล (Data encryption) และการบีบอัดข้อมูล (Data compression)* แต่ในระดับชั้นอื่นก็อาจมีความสามารถในงานเหล่านี้ได้เช่นเดียวกัน

การที่มีชื่อว่าระดับชั้นพรีเซนเตชั่น จึงมักทำให้เกิดความเข้าใจผิดว่าจะดูแลงานการแสดงผลข้อมูลให้กับผู้ใช้ แต่แท้ที่จริงแล้วจะเป็นการแสดงผลข้อมูลให้กับระดับชั้นบน คือชั้นแอพพลิเคชั่น

ชั้นที่ 7 ชั้นแอพพลิเคชั่น (Application Layer)

ชั้นแอพพลิเคชั่น จะเป็นระดับชั้นที่ติดต่อใกล้กับผู้ใช้มากที่สุด ระดับชั้นนี้จะไม่ใช่ระดับชั้นที่ประกอบด้วยโปรแกรมประมวลผลคำ, โปรแกรมทางด้านสเปรดชีต ความเข้าใจผิดนี้น่าจะมีสาเหตุมาจากที่มีชื่อว่า ชั้นแอพพลิเคชั่นนั่นเอง

แต่ระดับชั้นนี้ จะทำให้การเข้าใจระบบเครือข่ายของโปรแกรมที่ผู้ใช้ ใช้อยู่ มีรูปแบบที่มีลักษณะเหมือนกัน เช่นโปรแกรมประมวลผลคำ ไม่จำเป็นจะต้องเข้าใจถึงระบบเครือข่าย แต่ชั้นแอพพลิเคชั่นจะทำให้มองเห็นระบบเครือข่ายมีลักษณะเหมือนกับ DOS ทุก ๆ โปรแกรมก็จะมองระบบเครือข่ายเปรียบเหมือนเป็น DOS รายละเอียดในส่วนของระบบเครือข่ายจะถูกซ่อนไว้ โพรโตคอลในระดับชั้นนี้ที่รู้จักกันดีคือ NFS ทำให้ผู้ใช้สามารถใช้ไฟล์ที่อยู่บนเครื่องอื่น ในระบบเครือข่ายได้ หน่วยของข้อมูลจะเป็น ข้อความ (Message)

Layer	Information Units	Examples	Function	Hardware
Application	Messages	Telnet S.400, NFS, NCOPY	Application Interfaces	Gateways
Presentation		OSI Pres.Protocol Sun's XDR, Netware Password, encryption	Language translator	Gateways
Session		Sun's RPC OSI Sess.Protocol Novell's TTS	Dialog management	Gateways
Transport	Segments	TCP, SPX OSI's TP4	Reliable data Deiivery	Gateways Multiplexors
Network	Packets	IP,IPX	Routing over segments, Message fragmentation And reassembly	Routers
Data-Link	Frames	Ethernet, Token Ring, ARCnet	Packet and Unpacket data	Bridges
Physical	Bits	RS232,V.28	Move bits across Medium	Repeater

ตารางที่ 2.1 สรุปรายละเอียดมาตรฐาน OSI แยกตามระดับชั้น

2.2 LAN (Local Area Network)

รายละเอียดที่ควรสนใจในระบบเครือข่ายท้องถิ่น

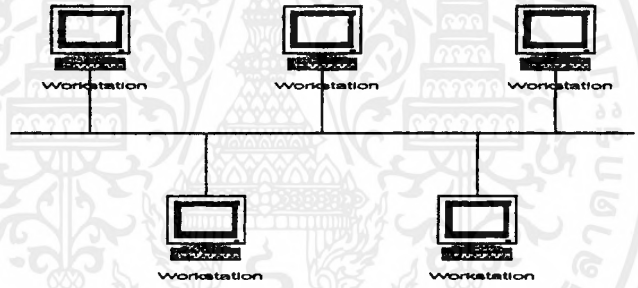
ในระบบเครือข่ายท้องถิ่นมีสิ่งที่ควรให้ความสนใจดังนี้

- เนตเวอร์กโทโพโลยี (Network Topology)
- วิธีการแบ่งเวลาใช้สายสัญญาณ (Access Method)
- ลักษณะของสัญญาณ (Signaling Technique)
- ชนิดของสายสัญญาณ (Transmission Medium)

2.2.1 เนตเวอร์กโทโพโลยี

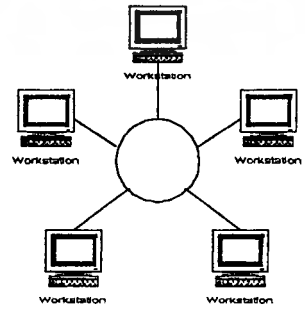
เนตเวอร์กโทโพโลยีในระบบเครือข่ายท้องถิ่น โดยทั่วไปมี 3 รูปแบบคือ

BUS คือใช้สายต่อเครื่องเข้าหาสายใหญ่ที่อยู่ตรงกลางหรือบัส เมื่อเครื่องหนึ่งจะติดต่อกับเครื่องอื่น ๆ ก็สามารถส่งข้อมูลออกมาบนบัสและไปถึงอีกเครื่องหนึ่งได้โดยตรง



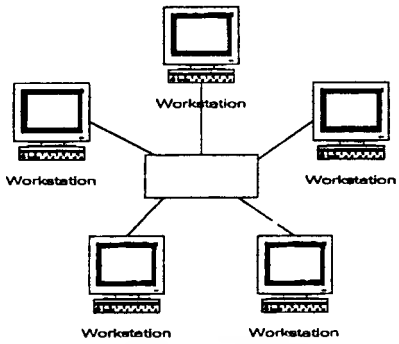
รูปที่ 2.2 เนตเวอร์กโทโพโลยีแบบ BUS

RING ตัวกลางจะร้อยเป็นวงผ่านทุก ๆ เครื่องในระบบจนครบ การส่งข้อมูลจะส่งออกมาในวงแหวน แต่ละเครื่องที่อยู่ระหว่างทางก็จะช่วย ๆ กันส่งข้อมูลที่ผ่านมาต่อเนื่องกันไปจนกว่าจะวนไปถึงปลายทางที่ต้องการ



รูปที่ 2.3 เนตเวอร์กโทโพโลยีแบบวงแหวน

STAR คือทุกเครื่องต่อกับอุปกรณ์ที่อยู่ตรงกลางเพียงตัวเดียว เมื่อเครื่องหนึ่งในระบบจะติดต่อกับเครื่องอื่น ๆ ก็ต้องผ่านตัวกลางนี้ก่อน



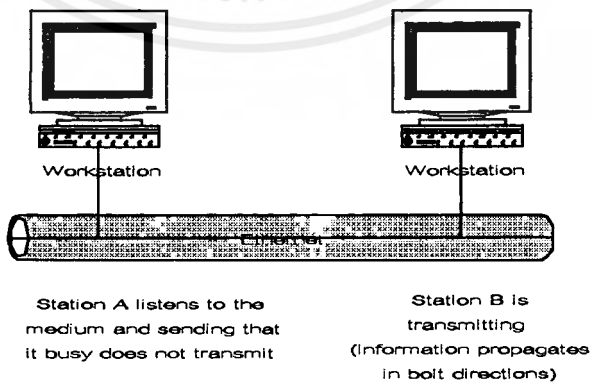
รูปที่ 2.4 เนตเวอร์กโทโพโลยีแบบดาว

2.2.2 วิธีการแบ่งเวลาใช้สายสัญญาณ

เนื่องจากแต่ละเครื่องในระบบเครือข่ายท้องถิ่น ใช้สายสัญญาณชุดเดียวกันในการติดต่อ จึงต้องมีวิธีการที่จะแบ่งเวลาในการใช้สายนี้ให้ทั่วถึง โดยไม่ต้องรอนานเกินไป ซึ่งโดยทั่วไปจะมี 2 แบบ คือ

1. **CSMA/CD (Carrier Sense multiple Access / Collision Detection)**

ใช้ในกรณีที่ทุกเครื่องต่ออยู่กับสายชุดเดียวกัน คือสำหรับระบบที่เป็นบัสนั่นเอง โดยที่ขณะใดขณะหนึ่งคอมพิวเตอร์แต่ละเครื่องจะคอย 'ฟัง' ว่าสายว่างหรือไม่ (carrier sense) ถ้าว่างก็จะเริ่มทำการส่งสัญญาณออกมา ถ้าสายว่างก็จะถึงผู้รับ แต่การเริ่มส่งสัญญาณนี้อาจเกิดขึ้นหลาย ๆ สถานีพร้อมกัน ผลก็คือสัญญาณที่ได้จะชนกันในสายทำให้ข้อมูลใช้ไม่ได้ ถึงตรงนี้แต่ละเครื่องจะต้องสามารถตรวจจับการชนกัน (collision detector) ได้ ซึ่งวิธีแก้ไข เราจะกล่าวโดยละเอียดในส่วนของการแบ่งเวลาใช้สายสัญญาณแบบ CSMA/CD บนเครือข่ายแบบ Ethernet โดยละเอียด

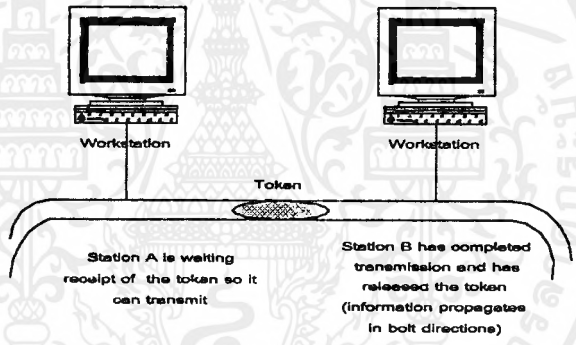


รูปที่ 2.5 แสดงการแบ่งเวลาใช้สายสัญญาณแบบ CSMA/CD

2. **Token Passing**

ใช้กับการต่อสายหรือโทโพโลยีได้หลายแบบ ไม่ว่าจะเป็นแบบบัส , แบบดาว หรือแบบวงแหวน โดยในขณะใดขณะหนึ่งจะมีคอมพิวเตอร์เพียงเครื่องเดียวในระบบเครือข่ายที่มีสิทธิ์ในการส่งข้อมูลโดยมีรหัสที่เรียกว่า โทเกน (Token) เก็บไว้ เมื่อส่งข้อมูลออกไปเสร็จแล้วก็จะส่งรหัสโทเกนนี้ออกไปให้เครื่อง อื่น ๆ ตามลำดับที่กำหนดไว้ล่วงหน้า เครื่องอื่น ๆ เมื่อได้รับ รหัสแล้ว ถ้าเครื่องไหนยังไม่ต้องการส่งข้อมูลก็จะส่งรหัสต่อไปเลย แต่ถ้าต้องการส่งข้อมูลก็ให้ส่งข้อมูลออกมาก่อน แล้วค่อยส่งรหัสตามออกไปให้เครื่องอื่นตามลำดับ ด้วยเครื่องจะได้รับสิทธิในการส่งข้อมูล (โดยมีรหัสโทเกนส่งมาถึง) 1 ครั้ง ภายใน 1 รอบการทำงานหรือ 1 ห้วงเวลาที่กำหนด ทำให้สามารถจำกัดเวลาได้ว่าจะส่งข้อมูลออกไปได้ภายในเวลาไม่เกินกี่ มิลลิวินาที

ในทางปฏิบัติระบบเครือข่ายท้องถิ่น ที่มีผู้ผลิตออกมาจำหน่ายในท้องตลาดจะใช้วิธีการหลาย ๆ แบบมาประกอบกัน บางแบบอาจใช้โทโพโลยีแบบบัส แต่ใช้การควบคุมแบบโทเกนพาสซิง เช่น อาร์กเนต (ARCnet) บางแบบอาจใช้บัสร่วมกับ CSMA/CD เช่น อีเทอร์เนต หรือบางแบบก็ใช้แบบวงแหวนร่วมกับโทเกนพาสซิง เช่น โทเกนริง



รูปที่ 2.6 แสดงการแบ่งเวลาใช้สายสัญญาณแบบ Token - Passing

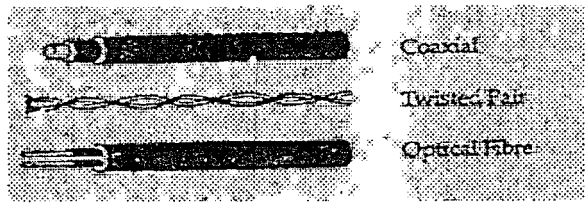
2.2.3 ลักษณะของสัญญาณ

ลักษณะของสัญญาณแบ่งออกได้เป็น 2 ประเภท

1. **Baseband** เป็นการส่งสัญญาณดิจิตอลออกไปบนสายสัญญาณโดยตรง ทำให้ในขณะหนึ่ง ๆ สามารถมี สัญญาณได้เพียงช่อง (Channel) เดียว

2. **Broadband** เป็นการส่งสัญญาณที่เป็นอนาลอกออกไปบนสายสัญญาณ (โดยการมอดูเลตสัญญาณดิจิตอลกับคลื่นพาหะ) ทำให้ในขณะเวลาหนึ่ง ๆ สามารถที่จะมีข้อมูลของหลาย ๆ งานอยู่บนสายสัญญาณได้ โดยใช้เทคนิคการมอดูเลตแบบความถี่ (FDM)

2.2.4 ชนิดของสายสัญญาณ



รูปที่ 2.7 แสดงชนิดของสายสัญญาณ

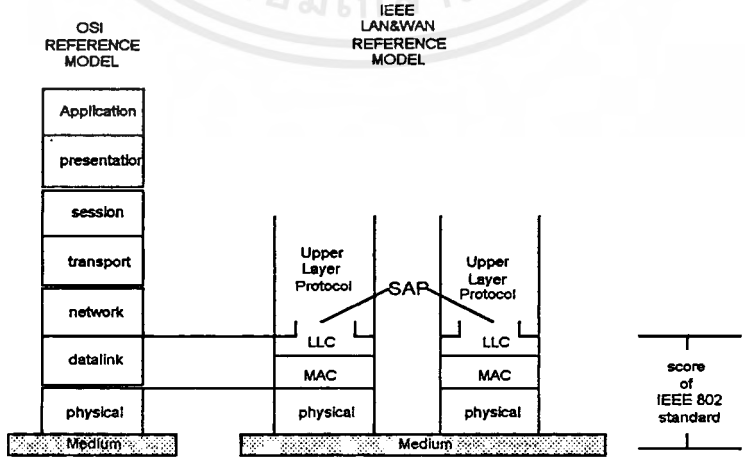
ชนิดของสายสัญญาณที่มักจะนำมาใช้กับระบบเครือข่ายท้องถิ่นคือ

- Broadband coaxial cable
- Baseband coaxial cable
- Twisted pair wire

แต่ในปัจจุบันได้มีการนำเอาเทคโนโลยีทางด้านสายสัญญาณแบบใยแก้ว (Fiber Optics) เข้ามาใช้กันอย่างแพร่หลาย ซึ่งตัวกลางชนิดนี้มีความสามารถในการรับส่งข้อมูลได้ในความเร็วสูง

มาตรฐาน IEEE กับระบบเครือข่ายท้องถิ่น

Institute of Electronic Engineers (IEEE) ได้ประกาศโครงการ 802 ขึ้นในเดือนกุมภาพันธ์ 1980 โดยครอบคลุมมาตรฐานระบบเครือข่ายท้องถิ่นที่มีความเร็วในการติดต่อไม่เกิน 20 ล้านบิตต่อวินาที และเรียกมาตรฐานดังกล่าวว่า "IEEE 802" หมายเลข 802 ก็คือช่วงเวลาที่ได้ประกาศใช้มาตรฐานนี้



รูปที่ 2.8 แสดงความสัมพันธ์ระหว่างมาตรฐาน OSI กับมาตรฐาน IEEE

จะสังเกตได้ว่า IEEE จะเน้นเฉพาะส่วนที่เกี่ยวข้องกับฮาร์ดแวร์ ซึ่งอยู่ในชั้นฟิสิคัล และดาต้าลิงค์ ของ OSI อาทิเช่น ลักษณะของการ์ดแลนด์ รูปแบบการเชื่อมต่อสาย

มาตรฐาน IEEE ได้แบ่งชั้นดาตาลิงค์ของ OSI ออกเป็น 2 ชั้นย่อยคือ ชั้นล่างจะเป็น **Media Access Protocol (MAC)** และชั้นบนจะเป็น **Logical Link Control(LLC)** ในชั้น MAC จะดูแลการควบคุมวิธีแบ่งใช้งานอุปกรณ์ทางฮาร์ดแวร์ที่จำเป็นต่องใช้ร่วมกัน มาตรฐานโทเกนริง และมาตรฐานอีเทอร์เนต ต้องใช้ MAC ที่แตกต่างกันเนื่องจากว่า โทเกนริงและอีเทอร์เนต มีวิธีการแบ่งการใช้งานฮาร์ดแวร์ที่แตกต่างกัน

แต่ทุก ๆ มาตรฐานของ IEEE จะใช้ชั้น LLC เหมือนกันหมดคือ **IEEE 802.2** การใช้มาตรฐานเดียวกันนี้คือใช้ IEEE 802.2 กับระบบเครือข่ายทุกรูปแบบ มีข้อได้เปรียบอย่างหนึ่ง คือ โพรโตคอลที่อยู่ในชั้นบนขึ้นไป ไม่ต้องสนใจว่ามาตรฐานในชั้นล่างจะมีรูปแบบเป็นเช่นไร การติดต่อกับโพรโตคอลในชั้นล่างจะใช้รูปแบบเดียวกันหมด การใช้งานสามารถทำได้ง่ายขึ้น

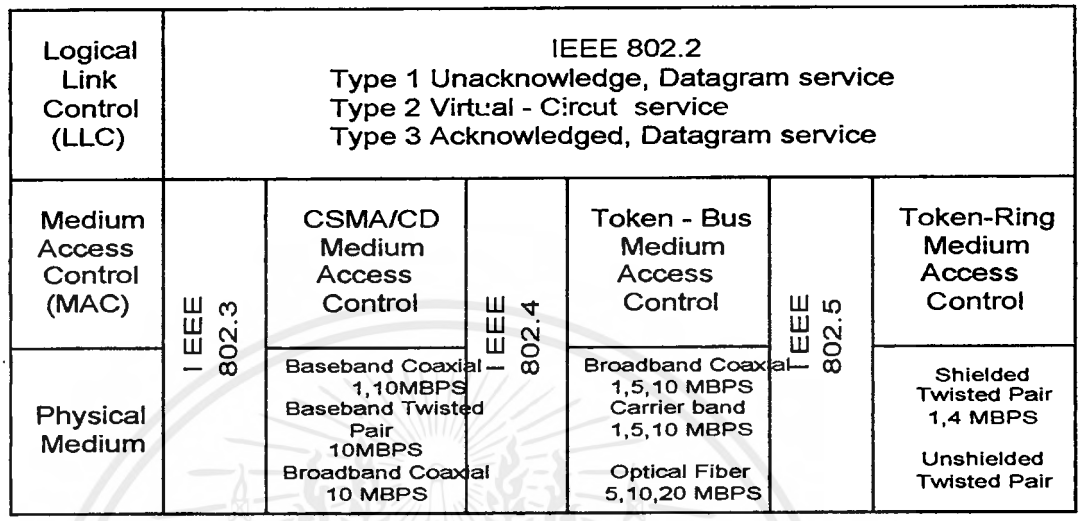
จะสังเกตอีกอย่างหนึ่งว่าการติดต่อกับโพรโตคอลชั้นบนของ LLC มาตรฐาน IEEE จะใช้ผ่าน **Link Service Access Points (LSAPs)** ตัว LSAPs จะเป็นที่อยู่ทางลอจิคัลของชั้นดาตาลิงค์หนึ่งที่อยู่ทางฟิสิคัล (ที่อยู่ MAC) สามารถที่จะมีที่อยู่ LSAPs ได้หลายตัว ทำให้สามารถติดต่อกันได้ในแบบหลายจุด (Multiple end-point connection) ระหว่างโหนด 2 โหนดบนระบบเครือข่าย

ชั้น LLC สามารถเลือกได้ว่าจะให้เป็นการติดต่อแบบวงจรเสมือน **(virtual Circuit)** ซึ่งเป็นแบบ *Connection-Oriented* หรือแบบ ดาตาแกรม **(Datagram)** ซึ่งเป็นแบบ *Connectionless-Oriented* หรือ อาจจะใช้แบบผสมกันทั้ง 2 แบบก็ได้ แบ่งออกได้เป็น 3 แบบดังนี้

แบบที่ 1 ดาตาแกรมในรูป **Unacknowledge** ทุก ๆ แพคเก็ตที่ส่งออกไป จะมีข้อมูลที่อยู่ครบถ้วนทั้งที่อยู่ของทางฝ่ายรับ และที่อยู่ขอองทางฝ่ายส่ง ไม่มีการตรวจสอบว่าแพคเก็ตที่ส่งออกไปจะถึงผู้รับหรือไม่ ลำดับของแพคเก็ตที่ส่งออกไปนั้น เมื่อถึงมือผู้รับแล้วจะถูกต้องหรือไม่ ทำงานได้ทั้งในแบบจุดต่อจุด (point- to –point) แบบหลายจุด (Multipoint) และแบบกระจายทุกจุด (Broadcast)

แบบที่ 2 แบบวงจรเสมือน มีการควบคุมการส่งแพคเก็ต ควบคุมลำดับของแพคเก็ตที่จะต้องตรงกันทั้งฝ่ายรับและฝ่ายส่ง มีการป้องกันการเกิดความผิดพลาด ก่อนที่จะส่งข้อมูลในรูปแบบนี้ต้องสร้างการติดต่อขึ้นมาก่อน และเมื่อการติดต่อจะสิ้นสุดลง จะต้องบอกเลิกการติดต่อเช่นเดียวกัน

แบบที่ 3 ดาตาแกรมในรูปแบบ Acknowledge เป็นรูปแบบผสมระหว่างแบบดาตาแกรม และแบบวงจรเสมือน โดยเมื่อเกิดความผิดพลาดในการส่งข้อมูล จะแก้ไขด้วยการให้ส่งข้อมูลชุดนั้นใหม่



รูปที่ 2.9 แสดงตัวอย่างมาตรฐาน IEEE แบ่งออกเป็นระดับชั้น ต่าง ๆ

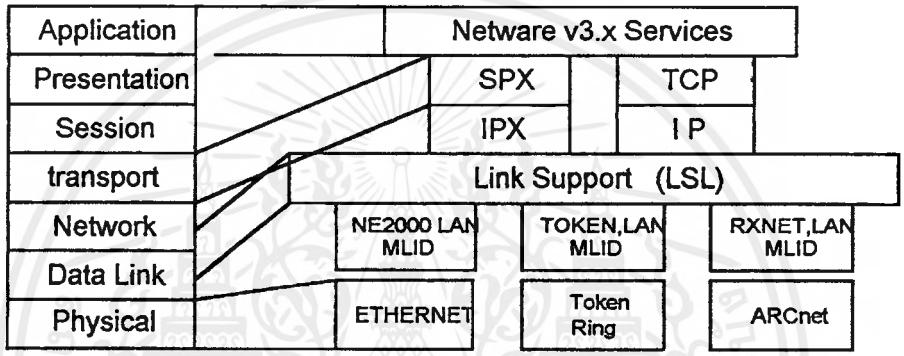
IEEE Standard	Meaning
IEEE 802.1	LAN bridging
IEEE 802.2	Logical Link Control
IEEE 802.3	Standardization of Ethernet Technology
IEEE 802.4	Token Bus station
IEEE 802.5	Token Ring Standard
IEEE 802.6	Metropolitan Area Network (MAN)
IEEE 802.7	Broadband technical advisory
IEEE 802.8	Fiber-optic technical advisory
IEEE 802.9	Integrated Voice/Data (VID)
IEEE 802.10	LAN security
IEEE 802.11	Wireless LANs
IEEE 802.12	100BASE-VG (100VG-anyLAN)

ตารางที่ 2.2 แสดงมาตรฐาน IEEE แบบต่าง ๆ

รูปแบบของ ODI กับมาตรฐาน OSI

ก่อนอื่นจะต้องเข้าใจว่าไม่ว่าจะเป็นมาตรฐานของ ODI , เนตแวร์ หรือของผู้ค้ารายใดๆ ก็ไม่สามารถที่จะจัดแบ่งระดับชั้นตามรูปแบบของ OSI ได้อย่างสมบูรณ์เลย

ODI มีความสามารถทำให้เนตเวอร์กการ์ดหลายรุ่น หลากหลายผู้ค้า สามารถใช้งานได้กับหลากหลายโพรโตคอลสแตกค์ (Protocol Stack) ทั้ง TCP/IP, OSI, SPX/IPX หรือ Apple Talk โดยก่อนหน้าที่จะมีรูปแบบ ODI หรือมีการใช้ NDIS แพคเก็ตไดร์เวอร์ (หรือที่ปัจจุบันเรียกว่า Crynwr ไดร์เวอร์) จำเป็นที่จะต้องใช้ไดร์เวอร์ของเนตเวอร์การ์ดสำหรับแต่ละโพรโตคอลสแตกค์ ทำให้เป็นการยากที่จะใช้หลาย ๆ ไดร์เวอร์ในเครื่อง ๆ เดียว ส่งผลให้ยากต่อการทำให้เครื่องหนึ่งเครื่องทำงานได้มากกว่า 1 โพรโตคอลแสตกค์



รูปที่ 2.10 แสดงการเปรียบเทียบมาตรฐาน OSI กับรูปแบบของ ODI

หลักสำคัญของ ODI คือการใช้ **Link Support Layer (LSL)** และ **Multiple Link Interface Driver (MLD)** จากรูปจะสังเกตได้ว่ามาตรฐานอีเทอร์เนต, โทเกนริงและอาร์กเนต จะสัมพันธ์กันกับการทำงานในชั้นที่ 1 และ 2 ของ OSI โดยมี NE2000.LAN, TOKEN.LAN และ RXNET.LAN เป็นชื่อของไดร์เวอร์ MLID ของแต่ละเนตเวอร์กการ์ด และมีการทำงานสัมพันธ์กับชั้นที่ 2 คือดาตาลิงค์ของ OSI ไดรเวอร์เหล่านี้ติดต่อโดยตรงกับ LSL ตัว LSL จะไม่สอดคล้องกับรูปแบบของ OSI ทีเดียวนัก เพราะทำงานครอบคลุมระหว่างชั้นดาตาลิงค์และเนตเวอร์ก จะสังเกตได้ว่า LSL จะเป็นตัวเชื่อมต่อกันระหว่างโพรโตคอลในชั้นบนกับมาตรฐานแบบต่าง ๆ ในชั้นล่าง

หลักสำคัญของ ODI อยู่ที่ LSL มันช่วยทำให้มองเห็นฟิสิคัลเนตเวอร์กการ์ด เป็นตัวเนตเวอร์กการ์ดเสมือน การทำงานที่อยู่ในระดับชั้นสูงขึ้นไป ไม่จำเป็นต้องเข้าใจในรายละเอียดส่วนการทำงานในชั้นด้านล่าง ไม่ต้องแยกแยะวิธีการใช้งานที่แตกต่างกันในเนตเวอร์กการ์ดแต่ละชนิด การใช้งานมองเห็นเป็นรูปแบบเดียวกันหมด เป็นรูปแบบมาตรฐาน หากมีการผลิตเนตเวอร์กการ์ดแบบใหม่ออกมา เพียงแต่เขียนไดร์เวอร์ให้เข้ากันได้กับ LSL ก็เป็นอันเรียบร้อย ตัวรายละเอียดที่อยู่ในชั้นสูงขึ้นไปไม่ต้องการแก้ไขอย่างไร

ในตัว MLID แต่ละตัว ก็จะใช้งานได้หลายโพรโตคอล ตราบใดก็ตามที่โพรโตคอลเหล่านั้น ยังใช้งานผ่านตัว LSL อยู่ โดย MLID ก็จะไปใช้งานโพรโตคอลเหล่านั้นโดยเรียกผ่านตัว LSL อีกทีหนึ่ง

ในกรณีที่ได้รับข้อมูลมา ตัวไดร์เวอร์จะส่งข้อมูลนั้นให้กับ LSL ทันที ไม่ต้องมีการจำแนกอย่างไรปล่อยให้เป็นหน้าที่ของ LSL ทำการจำแนกข้อมูลนั้น ส่งไปให้โพรโตคอลสแตกค์ที่สอดคล้องกัน ที่ทำเช่นนี้ได้เพราะว่า LSL จะเก็บข้อมูลทั้งตัว MILDs และโพรโตคอลสแตกค์ ที่มันทำงานด้วย โดยเมื่อ ไม่ว่า MILDs ตัวใดถูกเรียกใช้ หรือโพรโตคอลสแตกค์ใดถูกเรียกใช้ จะมีการเก็บ ค่าหมายเลขทางลอจิคัล ไว้ที่ตัว LSL ในส่วนของดาตาเซกเมนต์ (Data Segment)

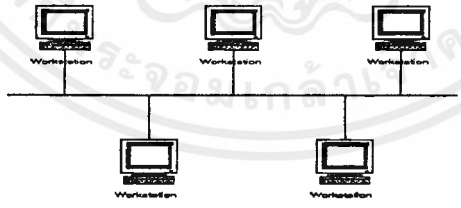
ส่วนของ LSL นี้จะเป็นของแต่ละรูปแบบระบบการปฏิบัติการ หมายความว่า LSL ของระบบปฏิบัติการหนึ่ง ๆ ไม่สามารถใช้ได้กับระบบปฏิบัติการอื่น ๆ ใน ดอส, โอเอสทู, เนตแวร์ ต่างก็ต้องมี LSL เฉพาะที่สัมพันธ์กับระบบปฏิบัติการของตนเอง

2.3 อีเทอร์เนต

อีเทอร์เนตได้ถูกนำมาใช้อย่างกว้างขวางในการทำโพรโตคอลระดับล่าง ซึ่งมีประวัติความเป็นมาและมีผู้ใช้และ การพัฒนามากมายซึ่งในส่วนนี้ จะกล่าวถึงรายละเอียดของอีเทอร์เนต

2.3.1 ประวัติการพัฒนา

อีเทอร์เนตเริ่มต้นในปี ค.ศ. 1970 เกิดจากการค้นคว้าของ Palo Alto Research Center ซึ่งเป็นส่วนหนึ่งของบริษัท Xerox ต้องการให้สามารถใช้งานสื่อสารระหว่างคอมพิวเตอร์ได้อย่างกว้างขวางและรวดเร็วจึงต้องใช้ความเร็วสูงในการส่งข้อมูล ซึ่งนี่คือจุดกำเนิดของ อีเทอร์เนต



รูปที่ 2.11 แสดงการเชื่อมต่อเครื่องคอมพิวเตอร์ในระบบเครือข่าย

วิธีการใช้ก็คือ ฟังสื่อที่ใช้ในการสื่อสารว่าว่างหรือไม่ แล้วจึงส่งข้อมูลลงไป ถ้าส่งไม่ได้เนื่องจากมีการใช้งานอยู่ก็จะรอช่วงเวลาหนึ่ง แล้วจึงส่งข้อมูลอีกที แต่อาจจะมีโอกาสที่ข้อมูลชนกันอีกได้เนื่องจากการส่งข้อมูลพร้อมกัน จึงต้องมีวิธีการในการป้องกันการส่งพร้อมกันซึ่งเรียกว่า แบ็คออฟ อัลกอริทึม (backoff algorothm) ซึ่งจะกล่าวภายหลัง ในตอนเริ่มแรกมีความเร็วเพียง 2.67

เม็กกะบิตต่อวินาที ซึ่งสร้างในปี ค.ศ. 1973 ถึง 1975 ซึ่งรุ่นที่ออกมานั้นยังเป็นรุ่นทดสอบอยู่ ซึ่งมีข้อกำหนดที่ความเร็วและจำนวนสเตชั่นที่เชื่อมต่อ (station)

ปี ค.ศ. 1980 เด็ค, อินเทลคอร์ปเปอเรชั่น และ ซีรอก ร่วมกันกำหนดมาตรฐานอีเทอร์เนต ซึ่งกำหนดมาตรฐานอีเทอร์เนตซึ่งกำหนดเป็นรุ่นที่ 1 โดยมีคุณสมบัติดังนี้

- 10 เมกกะบิตต่อวินาที
- ความยาวสูงสุด 2.8 กิโลเมตร
- จำนวนสเตชันสูงสุด 1024 เครื่อง
- ใช้หลักการส่งเบสแบนด์ (baseband)
- โทโพโลจีแบบบัส
- ขนาดเฟรมเปลี่ยนแปลงได้

2.3.2 หลักการแบ่งเวลาใช้สายสัญญาณ

อีเทอร์เนตใช้หลักการของ CSMA/CD ซึ่งทุกเวอร์กสเตชัน (workstation) ใช้สายสื่อระบบร่วมกันตามรูป เพราะว่าทุกเครื่องสามารถส่งข้อมูลได้ในเวลาเดียวกันบนสาย ซึ่งเป็นสาเหตุให้สายสัญญาณข้อมูลในสายเกิดการชนกันของข้อมูล ดังนั้นจะต้องส่งข้อมูลนั้นใหม่หมด

2.3.2.1 ข้อดีและข้อเสียของอีเทอร์เนต

ข้อดี

ง่ายในการติดตั้ง : คอมพิวเตอร์สามารถเชื่อมต่อกับเซกเมนต์ใด ๆ ได้ง่ายเพียงแต่ใช้ ที – คอนเนคเตอร์ (T – connector) หรือตัวรับส่ง (Transceiver) อื่น ๆ

เป็นวิทยาการที่ใช้กันอย่างแพร่หลาย : มีการใช้งานในอีเทอร์เนตเป็นระยะเวลานาน มีเครื่องมือที่ใช้ในการเชื่อมต่ออย่างมากมายง่ายในการเลือกใช้

ราคาอุปกรณ์ที่ใช้ไม่แพงมาก : อุปกรณ์ที่จำเป็นต่อการเชื่อมต่อ เช่น การ์ดอินเตอร์เฟสนั้นมีราคาไม่แพงมากนัก ทำให้ใช้กันอย่างแพร่หลาย

รูปแบบการใช้งานมีอย่างกว้างขวาง : อีเทอร์เนตสามารถใช้ชนิดของสายได้ไหลายแบบ เช่น 10base5, 10base2, 10baseT, Fiber optic

ข้อเสีย

เมื่อมีการใช้งานเพิ่มขึ้นประสิทธิภาพการใช้งานจะลดลง : หลักการของ CSMA/CD เมื่อมีการใช้งานของเครือข่ายเพิ่มขึ้น ประสิทธิภาพจะลดลงเพราะเนื่องจากการชนของข้อมูลยิ่งมากขึ้น ถ้ามีการใช้งานเพิ่มขึ้นเพราะโอกาสที่จะส่งข้อมูลพร้อมกันในสายสื่อระบบยิ่งมีมากขึ้น

ยากในการตรวจสอบปัญหา : อีเทอร์เนตยากที่จะตรวจสอบปัญหาที่เกิดขึ้นกับสายเพราะถ้าสายขาดแล้วจะทำให้ระบบแลนด์ทั้งเซกเมนต์ไม่ทำงาน

2.3.3 ขั้นตอนการทำงานของ CSMA/CD

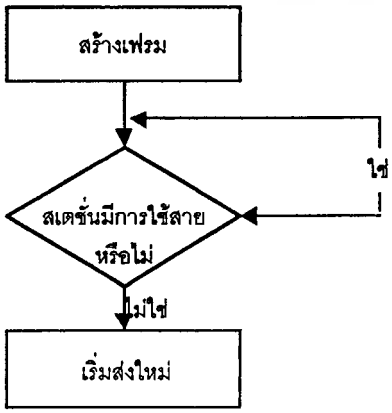
เนื่องจาก CSMA/CD ใช้สายสื่อระบบร่วมกัน ดังนั้นจึงมีข้อกำหนดในการส่งข้อมูลพร้อมกันได้ (CSMA/CD โพรโตคอลคล้ายกับการพูดคุยทางโทรศัพท์ ในขณะที่คนหนึ่งกำลังพูดคุยกันอยู่ในสาย ถ้าอีกคนหนึ่งพูดขึ้นมาพร้อมกันด้วยจะไม่ทราบข้อมูลที่อีกฝ่ายส่งมาได้ ต้องพูดคุยกันใหม่อีกครั้ง)

2.3.3.1 ขั้นตอนในการทำงานในการส่งข่อมูลบน CSMA/CD

มีด้วยกันทั้งหมด 4 ขั้นตอนดังนี้

ขั้นที่ 1 ฟังก่อนที่จะส่ง

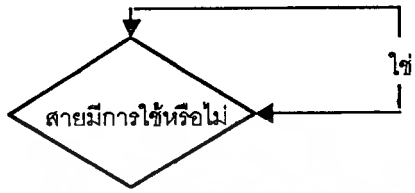
สเตชันจะเฝ้าดูว่าสายที่ส่งนั้นมีสัญญาณแคร์เรีย (carrier) หรือไม่ตามรูปที่ 2.12 สัญญาณนี้วัดโดยค่าระดับความต่างศักย์ซึ่งบ่งบอกถึงการใช้สายนั้น ถ้าสเตชันไม่พบ แคร์เรียออน (carrier on) จะหมายความว่าสายนั้นว่างและพร้อมที่จะส่ง (เหมือนกับการที่เรายกหูโทรศัพท์เพื่อดูว่าสายว่างหรือไม่) ถ้าสายไม่ว่าง (แคร์เรียออน) เมื่อสเตชันกำลังจะส่ง แพคเก็ตที่ส่งนั้นก็จะชนกัน



รูปที่ 2.12 สเตชั่นเฝ้าดูการใช้งาน

ขั้นที่ 2 รอถ้าสายไม่ว่าง

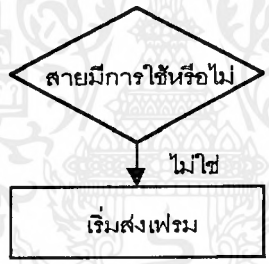
เพื่อป้องกันการชนกัน สเตชันจะรอถ้าสายนั้นถูกใช้งานอยู่ซึ่งแสดงตามรูปที่ 2.13 ซึ่งโดยปกติแล้วการ์ดอินเตอร์เฟสยังไม่ส่งถ้าสายไม่ว่าง (เปรียบเหมือนโทรศัพท์แล้วเหมือนกับว่า เมื่ออีกคนกำลังจะพูดอยู่ต้องรอให้พูดเสร็จก่อนจึงค่อยพูดกลับไป) Deferal time คือช่วงเวลาที่สเตชั่นจะต้องรอก่อนที่จะพยายามส่งใหม่อีกครั้ง



รูปที่ 2.13 สเตชั่นรอเวลาถ้าสายไม่ว่าง

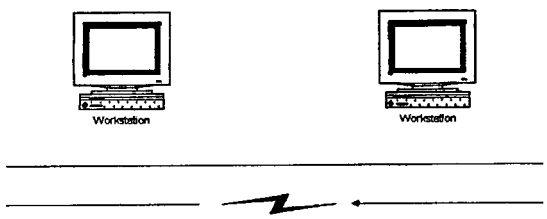
ขั้นที่ 3 ส่ง และ รอว่าแพคเก็ตที่ส่งไปนั้นชนกันหรือไม่

เมื่อสายว่างอย่างน้อยต้องใช้เวลา 9.6 ไมโครวินาที ถึงจะส่งข้อมูลตามรูป แล้วจึงส่งแพคเก็ตลงไปในสายสื่อระบบ



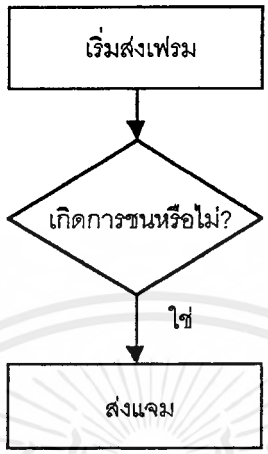
รูปที่ 2.14 ถ้าสายไม่ว่างสเตชั่นจะเริ่มส่งแพคเก็ต

ถ้าสเตชั่นอื่นบนเซกเมนต์ที่ส่งแพ็กเก็ตพร้อมกัน จะชนกันได้ตามรูปที่ 2.15 แสดงถึงการชนกันของแพคเก็ต (ถ้าคุณพูดพร้อมกันในโทรศัพท์ก็จะคุยกันไม่รู้เรื่อง) หลังจากส่งไปแล้วสเตชั่นก็จะทดสอบว่าสายมีการชนกันหรือไม่ ซึ่งการชนกันจะตรวจสอบได้โดยสัญญาณที่เกิดบนสายซึ่งจะเท่ากันหรือมากกว่าสัญญาณที่เกิดจากการส่งข้อมูล หรือมากกว่านั้นพร้อมกัน



รูปที่ 2.15 เมื่อมีการชนกันของแพคเก็ตในสื่อ

ถ้าเกิดการชนขึ้นสเตชั่นอื่นไม่พบสัญญาณว่าชนกัน อาจจะพยายามส่ง แต่จะเกิดการชนกันอีก เพื่อป้องกันการเกิดเหตุการณ์แบบนี้เกิดขึ้น สเตชั่นต้องทำให้ทุกสเตชั่นรู้ถึงสาย โดยส่งแจม (jam) ดังแสดงตามรูปที่ 2.16(แจม คือ การส่งอย่าง 32 บิตไม่เท่ากับค่า ซีอาร์ซี ของการส่งครั้งก่อนหน้านั้น) สเตชั่นจะเพิ่มค่าพยายามในการส่งอีก 1

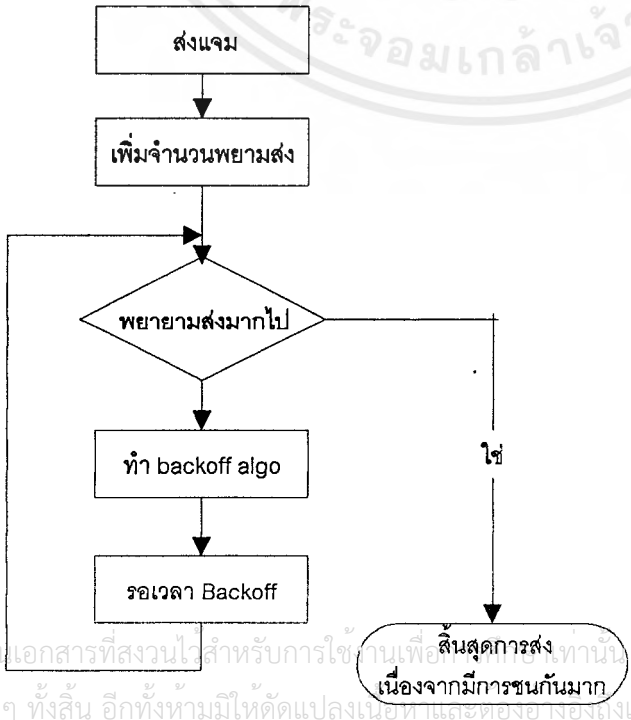


รูปที่ 2.16 ถ้ามีการชนเกิดขึ้นสเตชั่นจะส่งแจม

ขั้นที่ 4 รอก่อนจะส่งใหม่

ถ้าเราส่งทันทีหลังจากการชนจะทำให้เกิดการชนครั้งที่ 2 จำเป็นอย่างยิ่งที่จะกำหนดเวลาสุ่มเพื่อที่จะรอไม่ให้ชนกันอีก

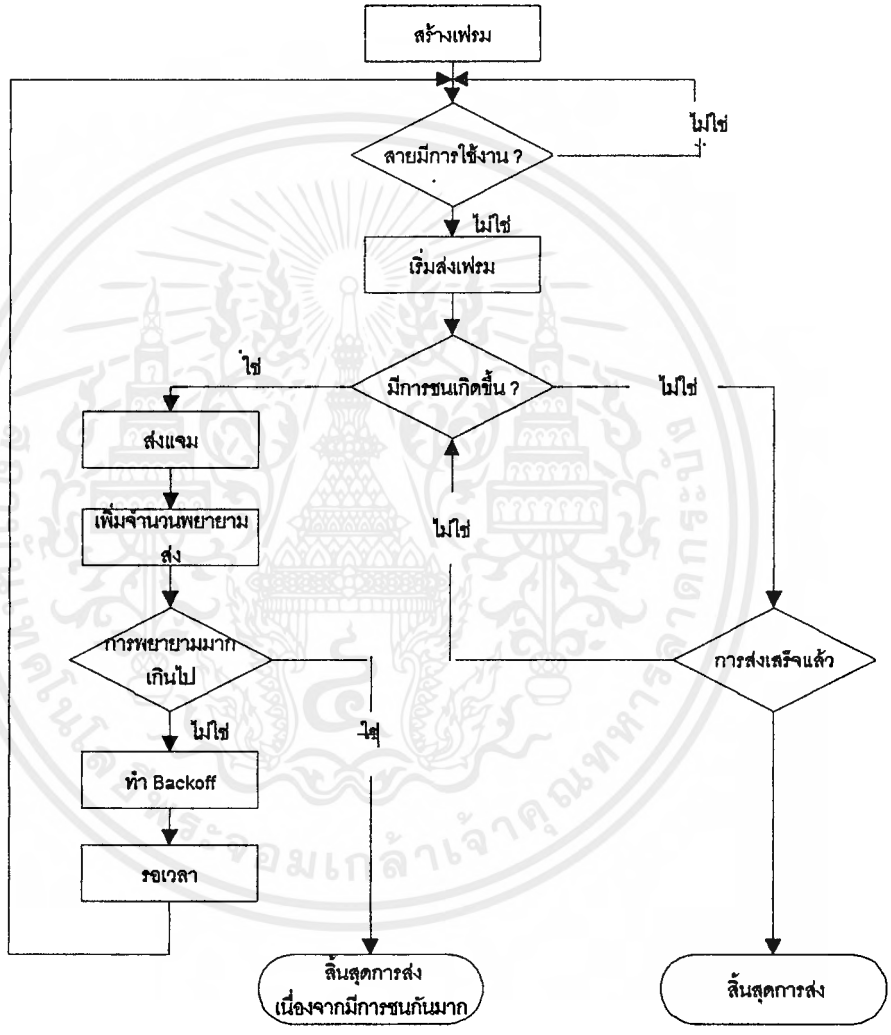
เพื่ออธิบายการส่งใหม่อีกครั้ง สเตชั่นจะทำวิธีการที่เรียกว่า แบ็คออฟอัลกอริทึม ซึ่งเป็นการกำหนดเวลาสุ่มที่จะใช้เพื่อการรอก่อนที่จะส่งแพ็กเก็จอีกครั้งหนึ่ง ตามรูป เพื่อลดการชนกันอีกเมื่อมีการส่งใหม่



รูปที่ 2.17 สเตชั่นจะใช้วิธีการแบคออฟเพื่อที่จะใช้ในการส่งแพคเก็ตอีกครั้ง

ขั้นที่ 5 ส่งอีกครั้งหรือหยุดการส่ง

ถ้ามีการส่งแพคเก็ตไปเรื่อยๆ แต่ไม่สามารถส่งไปในเครือข่ายได้ อาจเป็นไปได้ที่ปัญหาอาจเกิดจาก ปัญหาในเครือข่ายเอง จึงต้องกำหนดจำนวนครั้งในการพยายามที่จะส่งแพคเก็ต โดยทั่วไปจะกำหนดไม่เกิน 16 ครั้ง



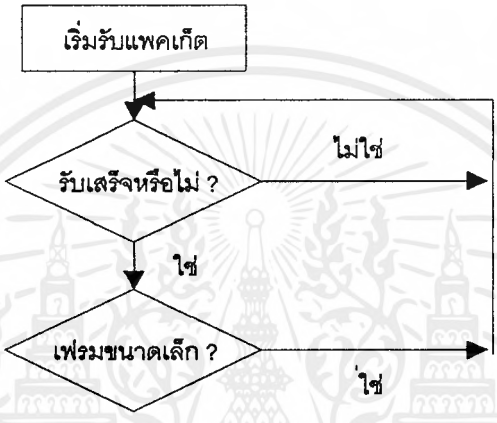
รูปที่ 2.18 ลำดับขั้นการส่งแพคเก็ต

2.3.3.2 ขั้นตอนการทำงานในการรับข้อมูลบน CSMA/CD

มีด้วยกันทั้งหมด 4 ขั้นตอน ดังนี้

ขั้นที่ 1 ตรวจสอบแพ็กเก็จที่ได้รับ

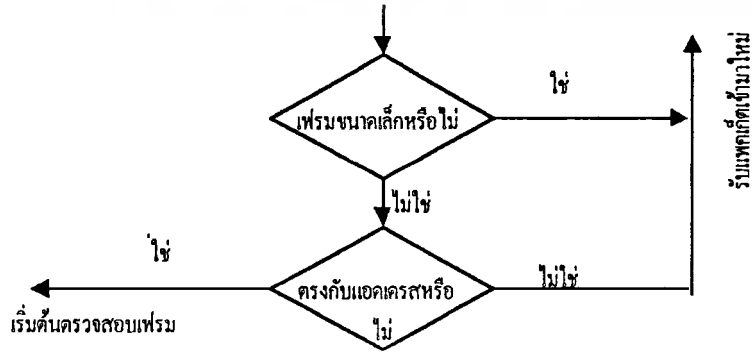
บนอีเทอร์เน็ตทุกสเตชั่นจะดูแพคเก็ตที่อยู่บนสาย เพื่อที่จะดูถึงแพคเก็ตที่ส่งมายังสเตชั่นนั้นๆ และต้องตรวจสอบว่ามีขนาดถูกต้องหรือไม่ (อย่างน้อย 64 ไบต์) และมีครบถ้วนไม่แฟรกเมนต์เนื่องจากการชน ซึ่งดูได้จากรูป



รูปที่ 2.19 เมื่อตรวจสอบแพคเก็ตสเตชั่นจะมองหาเฟรกเมนต์

ขั้นที่ 2 ตรวจสอบแอดเดรสปลายทาง

หลังจากตรวจสอบแอดเดรสแล้วว่าไม่แฟรกเมนต์ ก็จะตรวจสอบว่าแอดเดรสปลายทางของแพคเก็ตที่ตรวจสอบนั้นเป็น บรอดคราส หรือ เป็น มัลติคลาสหรือไม่ ตามรูปที่ 2.20



รูปที่ 2.20 สเตชั่นตรวจสอบที่อยู่ปลายทาง

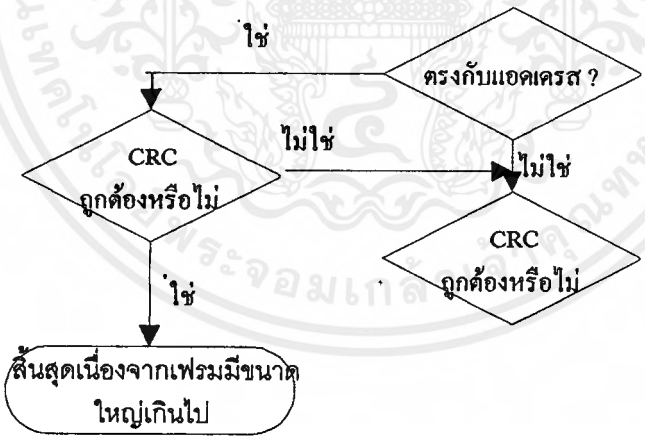
ขั้นที่ 3 ตรวจสอบถึงความถูกต้องของแพ็กเก็จ

ในจุดนี้สเตชั่นจะรับรู้ว่าแพคเก็ตไม่แฟรกเมนต์ และ เป็นแอดเดรสที่รับได้ แต่จะไม่รับรู้ว่าแพคเก็ตอาจจะเสียหายเนื่องจากการส่งข้อมูลในสายหรือไม่ เพื่อที่จะป้องกันการรับแพคเก็ตที่เสียหายนี้สเตชั่นที่จะรับจะต้องตรวจสอบถึงคุณสมบัติของแพคเก็ตตามรูปที่ 2.20 อันดับแรกถ้าจะต้องตรวจสอบขนาดของแพคเก็ต ถ้าเฟรมขนาดใหญ่เกินกว่า 1518 ไบต์ จะถือว่าเป็น โอเวอร์ไซส์เฟรม (oversize frame) ซึ่งจะตรวจสอบในแพคเก็ตไดร์เวอร์

บางทีอาจจะมีการสลับบิตได้ จาก 1 เป็น 0 หรือกลับกัน (ซึ่งทำให้แพคเก็ตเสียหาย) เพราะขณะที่ส่งไปบนสายสื่อระบบอาจจะมีการรบกวนของสภาพแวดล้อมซึ่งเป็นสาเหตุให้เกิดเหตุการณ์เช่นนี้ได้ ซึ่งจะต้องตรวจสอบ ซีอาร์ซี (CRC : Cyclical Redundancy Check) ถ้าตรวจสอบ ซีอาร์ซีแล้วไม่ผ่านจะตรวจสอบเฟรมนั้นต่อว่า อไลน์เมนท์ (alignment) ถูกต้องหรือไม่ แพ็กเก็จ มิส อไลน์ (misaligned) คือแพคเก็ตที่ไม่จบลงด้วยจำนวนเท่าของ 8 บิต เช่น แพคเก็ต ขนาด 72 ไบต์ 3 บิต อาจเป็นไปได้ที่เป็น 72 ไบต์ หรือ 73 ไบต์

เมื่อตรวจสอบว่าแฟรกเมนท์ ยาวเกินไป มีซีอาร์ซีไม่ถูกต้อง จัดเรียงผิดพลาด (มิส อไลน์) หลังจากนั้นก็ตรวจสอบว่าแพคเก็ตมีขนาดเล็กเกินไปหรือไม่ โดยพิจารณาขนาดว่าอย่างน้อยต้องมีขนาดมากกว่า 64 ไบต์

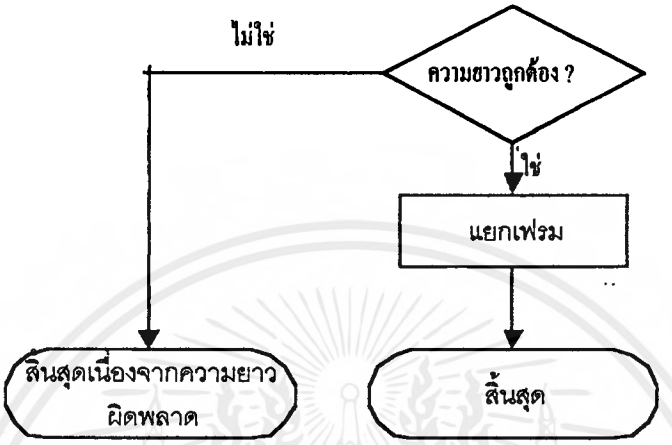
ตามรูปที่ 2.21 อธิบายถึงขั้นตอนการรับแพคเก็ตของสเตชั่นบนเครือข่ายแบบ CSMA/CD



รูปที่ 2.21 แพคเก็ตถูกตรวจสอบสำหรับความถูกต้อง

ขั้นที่ 4 โปรเซสแพ็กเก็จ

ถ้าแพคเก็ตผ่านขั้นตอนการตรวจสอบแต่ละขั้นแล้ว ตามรูปที่ 2.21 จะถือว่าแพคเก็ตนั้นถูกต้องในรูปแบบ ขนาด ถ้าสเตชันยังมีปัญหาอีกให้พิจารณาถึงข้างในแพ็กเก็จเพื่อที่หาปัญหาที่เกิดขึ้น บางทีอาจเกิดขึ้นในโพรโตคอลเลเยอร์อื่น ๆ ก็เป็นได้



รูปที่ 2.22 แพคเก็ตที่ถูกต้องจะถูกจัดการต่อ

2.3.4 ส่วนประกอบของอีเทอร์เน็ตเฟรม

อีเทอร์เน็ตเฟรมมีขนาดประมาณ 46 และ 1500 ไบต์ เฟรมน้อยกว่า 60 ไบต์ในส่วนข้อมูล จะเรียกว่า รั๊น (runt) เฟรม รูปที่เป็นตัวอย่างของอีเทอร์เน็ตเฟรม

Preamble	Destination Address	Sorce Address	Type	Data	FCS

รูปที่ 2.23 โครงสร้างเฟรม อีเทอร์เนตเฟรม

โครงสร้างของเฟรม

เฟรมนี้ประกอบด้วย 6 ฟิลด์ด้วยกัน ทุกฟิลด์มีหน้าที่และขนาดเฉพาะ ซึ่งมีรายละเอียดดังนี้

พรีแอมเบิ้ล (Preamble)

เป็นเลขลำดับ 64 บิตที่ฟิสิคัลย์เลเยอร์ ใช้เพื่อการสร้างสัญญาณพร้อม (synchronization signal) ระหว่างวงจรที่เชื่อมต่อกับสื่อ

ที่อยู่ปลายทาง (Destination Address)

เป็นเลข 48 บิต ฮาร์ดแวร์แอดเดรส ซึ่งเรียกว่า อีเทอร์เน็ตแอดเดรส หรือ แมคแอดเดรส (ทุก ๆ การ์ดอินเทอร์เฟสจะมีเลข 48 บิต ที่บ่งบอกถึง ความแตกต่างของการ์ดอินเทอร์เฟสนั้น) ของสเตชันที่ต้องการส่งไป

ที่อยู่ต้นทาง (Source Address)

เป็นเลข 48 บิต ฮาร์ดแวร์แอดเดรส ของผู้ส่ง

ไทป์ (Type)

เป็นเลข 2 ไบต์ใช้เพื่อบ่งบอกชนิดโพรโตคอลถ้ามีการใช้งานหลายโปรโตคอลในระดับบนสื่อเดียวกัน จะใช้เลขนี้บ่งบอกถึงโพรโตคอลในระดับบน

ข้อมูล (Data)

เป็นข้อมูลขนาดระหว่าง 4 ถึง 1500 ไบต์

เอฟซีเอส (FCS : FRAME CHECK SEQUENCE)

เป็นเลขขนาด 32 บิตที่คำนวณจาก ซีอาร์ซีทุก ๆ ฟิลด์ ยกเว้นฟิลด์ตัวเอง

ในสภาพแวดล้อมทั่วไป ในเครือข่ายจะมีโครงสร้างเฟรมของอีเทอร์เนต ที่ใช้พื้นฐานโครงสร้างของโครงสร้างเฟรมอีเทอร์เนต เพียงแต่แตกต่างกันตรงส่วนการใช้งานของแต่ละแบบ ซึ่งมีด้วยกันดังนี้

2.3.4.1 อีเทอร์เนต 802.3

อีเทอร์เนต 802.3 เฟรมนั้นคล้าย ๆ กับ อีเทอร์เนต II แต่ไม่เหมือนกันตรงฟิลด์ต่าง ๆ ดูได้ตามรูปที่ 2.17 ซึ่งมีหน้าที่และขนาดเฉพาะซึ่งมีรายละเอียดดังนี้

Preamble	Destination Address	Source Address	Length	Data	FCS

รูปที่ 2.24 โครงสร้างเฟรม อีเทอร์เนต 802.3

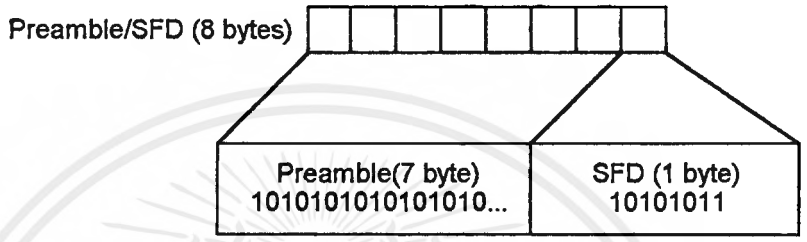
โครงสร้างเฟรม

พรีเอมเบิ้ล (Preamble)

เป็นเลขอันดับ 56 บิต หรือ 7 ไบต์ที่ฟิสิคัลเลเยอร์ใช้เพื่อการสร้างสัญญาณพร้อม (synchronization signal) ระหว่างวงจรที่เชื่อมต่อกับสื่อ

เอสเอฟดี (SFD : Start frame delimiter)

เป็นเลขไบนารี่ (binary) 10101011 ที่ชี้ถึงจุดเริ่มต้นของเฟรมดังตัวอย่างตามรูปที่ 2.18



รูปที่ 2.25 โครงสร้างฟิลด์ เอสเอฟดี

หมายเลขปลายทาง (Destination Address)

เป็นเลข 48 บิต ฮาร์ดแวร์แอดเดรส ซึ่งเรียกว่า อีเทอร์เนตแอดเดรส หรือ แมคแอดเดรส (ทุก ๆ การ์ดอินเทอร์เฟสจะมีเลข 48 บิต ที่บ่งบอกถึงความแตกต่างของการ์ดอินเทอร์เฟสนั้น) ของสเตชั่นที่ต้องการส่งไป แอดเดรส FFFFFFFFFFF หมายถึง บรอดคลาส

หมายเลขต้นทาง (Source Address)

เป็นเลข 48 บิต ฮาร์ดแวร์แอดเดรส ของผู้ส่งที่ไม่ต้องเป็น บรอดคลาส จะต้องเป็นแอดเดรสของ เวิร์คสเตชัน เซิฟเวอร์ หรือ เราท์เตอร์

ความยาว (LENGTH)

เป็นเลข 2 ไบต์ใช้เพื่อบอกขนาดแพคเก็ตซึ่งค่านี้จะต้องน้อยกว่า 1500

ข้อมูล (Data)

เป็นข้อมูลขนาดระหว่าง 46 ถึง 1500 ไบต์

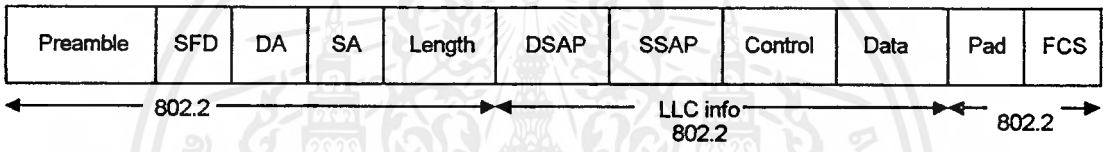
แพดดิ้ง (PADDING)

ฟิลด์นี้เปลี่ยนแปลงได้ ใช้เพื่อตรวจสอบแพคเก็ตให้มีขนาดตรงตามข้อกำหนด เช่นใน อีเทอร์เนตต้องมีขนาดอย่างน้อย 64 ไบต์ ซึ่งถ้ามีการส่งข้อมูลไม่ถึงจะต้องเพิ่มแพดดิ้งเพื่อเข้าไปให้ครบทั้ง 64 ไบต์

หมายเหตุ ถ้าเฟรมถูกต้องและ ค่าความยาวมากกว่า 1500 จะหมายความว่า เป็นอีเทอร์เนตทูเฟรม และเป็นไทป์ฟิลด์

2.3.4.2 อีเทอร์เนต 802.2

เฟรมอีเทอร์เนต 802.2 กำหนด IEEE – compliant เนื่องจากมีข้อมูลทั้ง 802.2 ฟิลด์ และ 802.2 ฟิลด์ กล่าวถึง LLC (Logical Link Control) เลเยอร์ภายในเฟรม ดูได้จากรูปที่ 2.26



รูปที่ 2.26 โครงสร้างเฟรม อีเทอร์เนต 802.2

โครงสร้างของเฟรม

พรีเอมเบิ้ล : 8 ไบต์

ที่อยู่ปลายทาง : 6 ไบต์

ที่อยู่ต้นทาง : 6 ไบต์

ความยาว : 2 ไบต์

ข้อมูลและแพดดิ้ง : 46 – 1500 ไบต์

เอฟซีเอส : 4 ไบต์

ดีเอสเอพี (DSAP : Destination Service Access Point)

เป็นเซอร์วิสแอคเซสพอยต์ปลายทางของโฮสปลายทางซึ่งใช้ในเลเยอร์บน หรือ เน็ตเวิร์ค เลเยอร์

เอสเอสเอพี(SSAP : Source Service Access Point)

เป็นเซอร์วิสแอคเซสพอยต์ต้นทางของโอสต้นทาง ซึ่งใช้ในเลเยอร์บน หรือ เน็ตเวิร์ค เลเยอร์

ควบคุม (Control)

กำหนดถึงการส่งแบบคอนเนคชั่นเลสเซอร์วิส (Connectionless Service)

2.3.4.3 อีเทอร์เนต สแนบ

สแนบมาจากเฟรม อีเทอร์เนต 802.2 ดังแสดงตามรูป

Preamble	SFD	DA	SA	Length	DSAP	SSAP	Control	Organization Code	Data	Pad	FCS

รูปที่ 2.27 โครงสร้างของเฟรม อีเทอร์เนต สแนบ

พรีเอมเบิ้ล : 8 ไบต์

ที่อยู่ปลายทาง : 6 ไบต์

ที่อยู่ต้นทาง : 6 ไบต์

ความยาว : 2 ไบต์

ข้อมูลและแพดดิ้ง : 46-1500 ไบต์

ดีเอสเอพี เอสเอสเอพี และ คอนโทรลฟิลด์

เอฟซีเอส : 4 ไบต์

ดีเอสเอพี เอสเอสเอพี และ คอนโทรล

ในอีเทอน์เนตสแนบค่าใน ดีเอสเอพี และ เอสเอสเอพี จะต้องเป็นค่า 0xAA ซึ่ง 0xAA เป็นตัวบ่งบอกถึง เฟรมนั้นที่เป็นสแนบ

ออร์กาไนซ์เซชั่นโคด (Organization Code)

ฟิลด์นี้กำหนดให้อีเทอร์เนต ไทป์ฟิลด์ซึ่งค่านี้เป็น 0x00-00-00 ในออร์กาไนซ์เซชั่นฟิลด์

อีเทอร์เนตไทป์

ฟิลด์นี้กำหนดถึงโปรโตคอลระดับบน (Upper layer protocol) ซึ่งมีค่าดังนี้

ไอพี	0x0800
เออาร์พี	0x0806
อาร์เออาร์พี	0x8305
เน็ตแวร์ ไอพีเอ็กซ์/เอสพีเอ็กซ์	0x8137

2.3.4.4 อีเทอร์เนตทู

อีเทอร์เนตทู เฟรมแตกต่างจาก 2 แบบที่กล่าวมาเนื่องจากไทป์ซึ่งตามหลังที่อยู่ปลายทาง แต่อีเทอร์เนต 802.3 อีเทอร์เนต 802.2 และ อีเทอร์เนตสแนบ จะเป็นฟิลด์ความยาวแทน ซึ่งแสดงตามรูป

Preamble	Destination Address	Source Address	Type	Data	FCS

รูปที่ 2.28 โครงสร้างของเฟรมอีเทอร์เนตทู

โครงสร้างของเฟรม

พรีเอมเบิ้ล

เป็นเลขลำดับ 64 บิตหรือ 8 ไบต์ที่ฟิสิคัลเลเยอร์ใช้เพื่อสร้างสัญญาณพร้อม (synchronization signal) ระหว่างวงจรที่เชื่อมต่อกับสื่อ กำหนดด้วยค่าสลับกันระหว่าง 1 และ 0 ซึ่งมีด้วยกัน 7 ไบต์ ส่วนไบต์สุดท้ายเป็นเอสเอฟดี

ไทป์

เป็นส่วนที่บอกถึงชนิดโปรโตคอลในระดับบนซึ่งค่าโปรโตคอลที่ใช้มีดังนี้

ไอพี	0x0800
เออาร์พี	0x0806
อาร์เออาร์พี	0x8305
เน็ตแวร์ ไอพีเอ็กซ์/เอสพีเอ็กซ์	0x8137

สังเกตว่าค่าเหล่านี้จะคล้ายกับ อีเทอร์เนตสแนบไทป์ฟิลด์

มาตรฐานการส่งข้อมูลของ IEEE 802.3 ที่ได้รับการปรับปรุงใหม่

เดิมทีเดียวมาตรฐานอีเทอร์เนตจะใช้สายส่งข้อมูลแบบ โคแอกเชียล โดยมี 2 แบบ คือ

- สายส่งข้อมูลแบบบาง (RG-58, 50 โอห์ม) สามารถใช้กับเซกเมนท์ที่มีความยาวไม่เกิน 185 เมตร
- สายส่งข้อมูลแบบหนา (RG-11 , 50 โอห์ม) สามารถใช้กับเซกเมนท์ที่มีความยาวไม่เกิน 500 เมตร ถ้าใช้อุปกรณ์ทวนสัญญาณ (Repeater) จะทำให้สามารถเพิ่มความยาวได้ออกไปถึง 3,000 เมตร

ตามมาตรฐานการส่งข้อมูล IEEE 802.3 ที่ปรับปรุงใหม่ขึ้นนี้ จะส่งข้อมูลด้วยความเร็ว 10 ล้านบิตต่อวินาที ยกเว้นมาตรฐาน 1BASEA5 ที่มีการส่งข้อมูลด้วยความเร็วเพียง 1 ล้านบิตต่อวินาที แต่มีระยะทางไกลกว่า มาตรฐานที่ได้รับความนิยมมากได้แก่ 10Base5 , 10Base2 และ 10BaseT

ตามมาตรฐานที่มีการปรับปรุงใหม่ดังนี้

10Base5 เป็นการส่งข้อมูลโดยใช้สายส่งข้อมูลแบบ RG-11 หรือสายโคเอกเชียลแบบหนา ความยาวสูงสุดของเซกเมนต์ 500 เมตร (1640 ฟุต) สามารถต่อตัวทรานซีฟเวอร์ (Transceiver) ในแต่ละเซกเมนต์ได้สูงสุด 100 ตัว จำนวนสูงสุดของ DTE ในระบบเครือข่ายคือ 1,024 ตัว ระยะห่างต้องไม่ต่ำกว่าข้อมูลแบบ Baseband

10Base2 เป็นการส่งข่อมูลโดยใช้สายส่งข้อมูลแบบ RG-58A/U หรือสายโคเอ็กเชียลแบบบางนั่นเอง ความยาวสูงสุดของเซกเมนต์ต้องไม่เกิน 185 เมตร (600 ฟุต) สามารถต่อตัวทรานซีฟเวอร์สูงสุด 30 ตัว และระยะห่างระหว่างตัวทรานซีฟเวอร์กับเซกเมนต์ต้องไม่ต่ำกว่า 0.5 เมตร การส่งข้อมูลเป็นแบบ Baseband

10BaseT ใช้สายส่งข้อมูลแบบ 20 ถึง 24 AWG UTP หรือสายส่งข้อมูลแบบคู่บิดเกลียว และใช้ตัวเชื่อมต่อแบบ RJ-45 ระยะห่างระหว่างตัวทรานซีฟเวอร์กับฮับ (HUP) จะต้องไม่ยาวกว่า 100 เมตร (328 ฟุต) ฮับ 1 ตัวมีเครื่องลูกข่ายต่ออยู่ได้ไม่เกิน 1023 เครื่อง

10Broad36 ใช้สายส่งข้อมูลแบบโคเอกเชียล (RG-59/U CATV Type) มีความยาว แต่ละเซกเมนต์สูงสุด 3,600 เมตร ใช้วิธีการส่งข้อมูลแบบ Broadband

1Base5 ใช้สายส่งข้อมูลแบบคู่บิดเกลียวที่มีความยาวเซกเมนต์สูงสุดไม่เกิน 500 เมตร ส่งข้อมูลด้วยความเร็ว 1 ล้านบิตต่อวินาที

Rate	Topology	Media	Segment Length in Meters	Data (MBPS)
Ethernet	Bus	50 ohm thick coax	500	10
10Base5	BUS	50 Ohm thick coax	500	10
10Base2	BUS	50 Ohm thin coax	185	10
10BaseT	STAR	Unshielded Twisted pair	100	10
10Broad36	BUS	75 Ohm coax	1800	10
1Base5	STAR	Unshielded Twisted pair	250	1

ตารางที่ 2.3 แสดงรายละเอียดมาตรฐานแบบต่าง ๆ ของ IEEE 802.3

การตั้งชื่อมาตรฐานต่างๆ ของ IEEE 802.3 นี้เป็นการตั้งเพื่อให้สื่อความหมาย โดยตัวเลขด้านหน้าสุด จะแสดงความเร็วในการส่งผ่านข้อมูลบนตัวกลางชนิดนั้นๆ ส่วนคำที่อยู่ตรงกลางจะหมายถึงลักษณะของสัญญาณ (Baseband & Broadband) ส่วนตัวสุดท้ายคือความยาวที่มากที่สุดที่จะสามารถเชื่อมต่อได้ของตัวกลางชนิดนั้น ๆ โดยใช้หน่วยเป็น 100 เมตร

หมายเหตุ T ใน 10BaseT หมายถึง Twisted Pair

36 ใน 10Broad36 ได้มาจากข้อกำหนดที่ว่า 10Broad36 จะต้องประกอบด้วย 2 เซกเมนต์จะมีความยาวมากที่สุดได้ไม่เกิน 1,800 เมตร จึงประมาณเป็น 200 เมตรแทน

2.4 ชุดอินเตอร์เนตโพรโตคอล

ชุดอินเตอร์เนตโพรโตคอล (Internet Protocol Suite) นั้น คนทั่วไปมักจะรู้จักกันในนามของโพรโตคอล *TCP/IP (Transmission Control and Internet Protocol)* ทั้งๆ ที่ในความเป็นจริงแล้ว การใช้งานของโพรโตคอลในกลุ่มนี้ประกอบด้วยโพรโตคอลมากมาย ไม่ได้มีเพียงแค่ 2 ชนิดนี้เท่านั้น

ถ้าหากคุณได้ศึกษาในรายละเอียดของอินเตอร์เนตโพรโตคอลแล้ว จะสังเกตได้ว่าไม่มีการกำหนดรายละเอียดในส่วนมาตรฐานใน 2 ชั้นล่างคือ ชั้นดาตาลิงค์ และ ชั้นฟิสิคัล เลย ทำให้ชุดอินเตอร์เนตโพรโตคอล สามารถทำงานได้กับระบบเครือข่ายได้หลาย ๆ ชนิด ไม่ว่าจะเป็นอีเทอร์เนต , โทเคนริง , โทเคนบัส, HDLC หรือ X.25 แต่อาจจะต้องมีการปรับแต่งอะไรอีกเล็กน้อย ด้วยความสามารถนี้เองที่ทำให้ชุดอินเตอร์เนตโพรโตคอล ได้รับการยอมรับความสามารถในการรวมหลาย ๆ ระบบ เข้าไว้ด้วยกัน

มีการนำชุดอินเตอร์เนตโพรโตคอลมาใช้ในการเชื่อมระบบเครือข่ายท้องถิ่นเข้าด้วยกัน เพราะสามารถใช้งานได้กับมาตรฐานหลายชนิด อีกทั้งมีความสามารถทำงานได้หลายอย่างไม่ว่าจะเป็น File Tranfer Protocol (FTP), Simple Mail Transfer Protocol (STMP) และ Terminal Service (Telnet) ปัจจุบันชุดอินเตอร์เนตโพรโตคอลได้เป็นมาตรฐานของระบบ Interconnection Lan ทั้งในหน่วยงานของรัฐและเอกชน

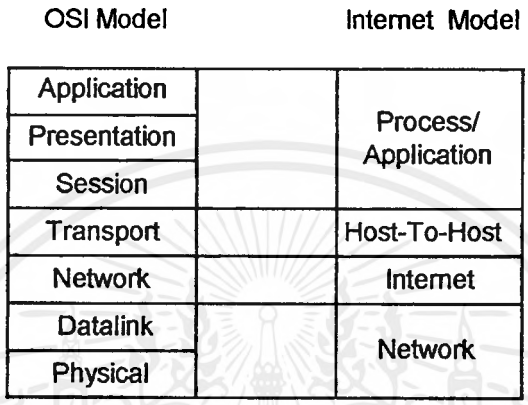
2.4.1 ประวัติความเป็นมาของชุดอินเตอร์เนตโพรโตคอล

ในช่วงปลายทศวรรษที่ 1960 มหาวิทยาลัยในสหรัฐอเมริกา และสถาบันวิจัยต่างๆ มีความต้องการที่จะเชื่อมต่อระบบเครือข่ายคอมพิวเตอร์เข้าด้วยกัน เพื่อใช้ติดต่อส่งข้อมูลข่าวสารระหว่างกัน แต่ติดขัดปัญหาที่ระบบคอมพิวเตอร์ของแต่ละหน่วยงานมีความแตกต่างกัน สถาบันวิจัยโครงงานขั้นสูง (Advanced Reserch Project Agency) หรือ ARPA จึงจัดตั้ง ARPANET ขึ้น เพื่อศึกษาแนวทางและออกแบบพัฒนารูปแบบในการเชื่อมต่อระบบเครือข่ายที่มีอยู่ในแต่ละหน่วยงานเข้าด้วยกัน โดยเริ่มแรกนั้นใช้รูปแบบของ packet switching โดยใช้โครงข่ายโทรศัพท์เป็นช่องทางการเชื่อมต่อหลัก

ARPA รู้จักกันในนามของ DARPA (Department of Defence Advanced Research Project Agency) เพราะก่อตั้งโดยรัฐบาลสหรัฐ เพื่อค้นคว้างานทางด้านที่เกี่ยวกับการทหาร ซึ่งในปัจจุบันนี้ ARPANET ก็ได้เป็นเครือข่ายย่อยหนึ่งใน ระบบเครือข่ายอินเตอร์เนต พร้อมกับเครือข่าย ย่อยๆ อื่นๆ ทางด้านงานวิจัยและทางอุตสาหกรรมอีกมากมายทั่วโลก

ส่วนหนึ่งที่ทำให้ชุดโพรโตคอล ได้รับการยอมรับอย่างแพร่หลายก็คือ ได้มีการนำเอาผลการวิจัยและพัฒนา ออกเผยแพร่ให้กับบุคคลที่ต้องการในนามของ 4.2 BSD (Berkeley System Distribution) โดยผู้ใช้ไม่ต้องเสียค่าใช้จ่ายใด ๆ

เปรียบเทียบรายละเอียดของชุดอินเตอร์เนตโพรโตคอล กับรูปแบบของ OSI



รูปที่ 2.29 แสดงการเปรียบเทียบมาตรฐาน OSI กับชุดอินเตอร์เนตโพรโตคอล

ชั้นที่ 1 : Network

จะครอบคลุมการทำงานในส่วน 2 ชั้นล่างของ OSI คือ ชั้นฟิสิคัล และดาตาลิงค์ โดยข้อกำหนดจะขึ้นอยู่กับชนิดของระบบเครือข่ายที่ใช้ อาทิเช่น อีเทอร์เนต , X.25

ชั้นที่ 2 : Internet

จะครอบคลุมการทำงานในส่วนชั้นเนตเวิร์กของ OSI โดยจะมีโพรโตคอล IP และ ICMP ทำงานในชั้นนี้

ชั้นที่ 3 : Host – to - Host

จะครอบคลุมการทำงานในส่วนชั้นทรานสปอร์ตของ OSI และรวมถึงการทำงานบางอย่างของชั้นเซสชั่นด้วย มีโพรโตคอล TCP และ UDP อยู่ในชั้นนี้

ชั้นที่ 4 : Process/Application

เป็นชั้นสูงสุด ครอบคลุมหน้าที่การทำงานของชั้นแอพพลิเคชั้น , พรีเซนเตชั่น และบางส่วนของเซสชั่น โดยมี TELNET (Virtual Terminal Emulation) , FTP (File Transfer Protocol) , TFTP (Trival File Transfer Protocol) และ SMTP (Simple Mail Transfer Protocol) ทำงานในชั้นนี้

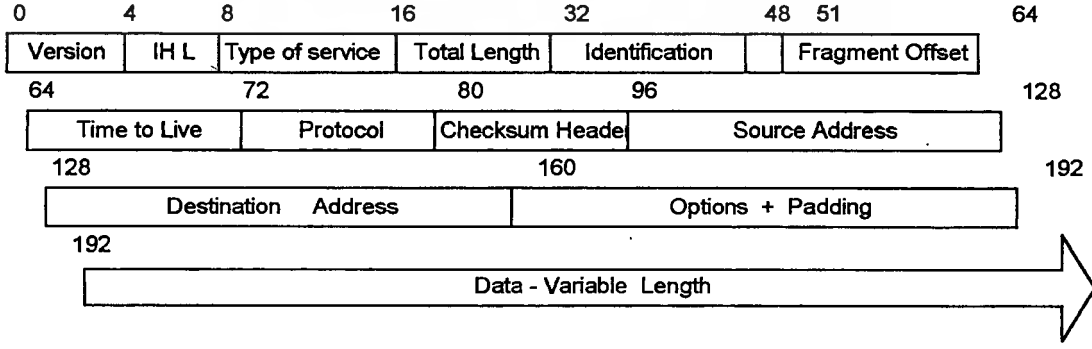
2.4.2 รายละเอียดของแพคเก็ต

2.4.2.1 IP แพคเก็ต

IP เป็นโพรโตคอลที่อยู่ในชั้นเนตเวอร์กของ OSI หรือก็คือชั้นอินเตอร์เนตของชุดอินเตอร์เนตโพรโตคอล ทำหน้าที่รับผิดชอบเกี่ยวกับการค้นหาเส้นทาง (Routing) บนระบบเครือข่ายและหากในกรณีแพคเก็ตมีขนาดใหญ่เกินไป IP ก็จะทำหน้าที่แบ่ง Packet นั้น ๆ ออกเป็นชิ้นย่อย ๆ (Fragment) แล้วจึงส่งผ่านไปในระบบเครือข่ายอื่น ต่อไป และเมื่อไปถึงปลายทาง จึงทำการจัดเรียงลำดับที่ถูกต้องใหม่ เนื่องจากว่าแต่ละแพคเก็ตที่ส่งผ่านมานั้น สามารถใช้เส้นทางคนละเส้นทางได้ โดยในส่วนตัวแพคเก็ตเองจะมีส่วนที่ใช้ในการจัดเรียงลำดับข้อมูล คืนตามรูปแบบเดิมไว้ด้วย

แต่การค้นหาเส้นทางของแพคเก็ต จะต้องใช้ข้อมูลจากตารางการค้นหาเส้นทาง **(Routing Information Table)** ซึ่งในการสร้างและปรับปรุงค่าต่างๆ ในตาราง จะต้องใช้โพรโตคอลชนิดอื่นเข้ามาช่วย ซึ่งจะขอยกตัวอย่างเพียงโพรโตคอลเดียวคือ Routing Information Protocol (RIP) ในส่วนตอนท้ายของบทนี้

รายละเอียดในแต่ละฟิลด์ของ IP แพคเก็ต



รูปที่ 2.30 แสดงฟิลด์ต่าง ๆ ของ IP แพคเก็ต

Version

ขนาด 4 บิต

เป็นเขตข้อมูลแรกสุดมีความยาว 4 Bits ใช้เก็บหมายเลขรุ่นของ Datagram วิธีการนี้ทำให้ Datagram ที่ได้รับการพัฒนาขึ้นมาใหม่สามารถทำงานร่วมกับ Datagram รุ่นเก่าได้ นั่นคือจะมี Datagram หลายๆรุ่นใช้งานอยู่บนเครือข่าย Internet เนื่องจากความยาวของ Datagram ส่วนหัวมีขนาดเปลี่ยนแปลงได้อยู่ระหว่าง 20 Bytes ถึง 60 Bytes เขตข้อมูล ISL จึงใช้กำหนดความยาวของ Datagram โดยบอกเป็นจำนวนคำของข้อมูล Computer ในกรณี Internet 1 คำมีความยาว 4 Bytes ดังนั้นค่าต่ำสุดของ ISL จึงเป็น 5 คำ ซึ่งเท่ากับ 20 Bytes เนื่องจากเขตข้อมูลนี้มีความยาวเพียง 4 Bits ทำให้กำหนดค่าสูงสุดได้เท่ากับ 15 จึงเป็นเหตุผลว่าข้อมูลส่วนหัว Datagram มีได้สูงสุด 15 คำซึ่งเท่ากับ 60 Bytes เมื่อตัดส่วนบังคับออกไปแล้วจะเหลือเนื้อที่สำหรับส่วนขยายเพียง 10 คำหรือ 40 Bytes ซึ่งเนื้อที่ขนาดนี้ไม่เพียงพอในการบรรจุข้อมูลที่สำคัญหลายๆอย่าง เช่น ทางเดินโดยละเอียดของ Datagram จากผู้ส่งไปยังผู้รับ

IP Header Length (IHL)

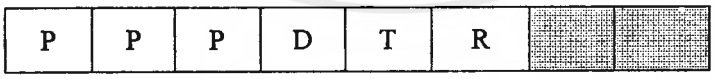
ขนาด 4 บิต

IHL จะระบุความยาวของเฮดเดอร์ของดาตาแกรม ในหน่วยของเวิร์ด (ขนาดเท่ากับ 32 บิต) โดยทั่วไปแล้วเฮดเดอร์จะมีความยาว 5 เวิร์ด (ถ้าไม่มีการใช้ส่วนฟิลด์ Options)

Type of Service

ขนาด 8 บิต

จะเป็นส่วนหนึ่งที่ระบุว่าจะให้โพรโตคอลที่อยู่ในชั้นสูงขึ้นไป จัดการกับแพคเก็ตนั้น ๆ อย่างไร โดยจะแบ่งออกเป็นบิตต่าง ๆ ดังนี้



P = Precedence

D = Delay

T = Throughput

R = Reliability

Grey = Unused

รูปที่ 2.31 แสดงค่าต่าง ๆ ในฟิลด์ Type of Service

Precedence

จะใช้ 3 บิตแรกเป็นตัวกำหนดความสำคัญ (Priority) ของแพคเก็ต โดย '0' จะมีความสำคัญต่ำสุด ส่วน '7' จะมีความสำคัญสูงสุด

Delay

อยู่ในตำแหน่งบิตที่ 4 บิตนี้จะถูกกำหนดเป็น '1' ถ้าต้องการส่งแพคเก็ตแบบที่ไม่ต้องมีการหน่วงเวลา ถ้าหากเป็น '0' จะเป็นการส่งในรูปแบบปกติ

Throughput

ถ้าหากต้องการให้การส่งแพคเก็ตมีประสิทธิภาพสูง ให้กำหนดบิตนี้เป็น '1' ถ้าหากเป็น '0' จะเป็นการส่งในรูปแบบปกติ

Reliability

บิตนี้จะกำหนดเป็น '1' หากเป็นการส่งข้อมูลที่ต้องการให้มีความเชื่อถือสูง อาทิเช่น ในระบบธนาคาร หากกำหนดเป็น '0' จะเป็นการส่งตามรูปแบบปกติ ส่วน 2 บิตสุดท้ายนั้น ยังไม่มีการนำมาใช้ โดยทั่วไปจะกำหนดให้เป็น '0'

Bits	Description	Values
0-2	Precedence	0-7
3	Delay	0 – Normal 1 – Low
4	Throughput	0 – Normal 1 – High
5	Reliability	0 – Normal 1 - High
6-7	Reverse for future use	

ตารางที่ 2.4 แสดงค่าต่าง ๆ ในฟิลด์ Type of Service

Length

ขนาด 16 บิต

จะเป็นฟิลด์ที่แสดงถึงขนาดความยาวของแพคเก็ตนั้น ๆ มีหน่วยเป็นจำนวนไบต์

Identification

ขนาด 8 บิต

ใช้เป็นฟิลด์ระบุหมายเลขของชิ้นส่วนย่อย เมื่อตอนที่ผู้รับได้ชิ้นส่วนย่อยนั้นแล้ว ก็จะได้ทราบว่า เป็นของแพคเก็ตไหน

Flag

ขนาด 3 บิต

Bit 1	Bit 2	Bit 3
Reserved	0 – May Fragment 1 – Do Not Fragment	0 – Last Fragment 1 – More Fragment

ตารางที่ 2.5 แสดงค่าและค่าความหมายในฟิลด์

ใช้เป็นฟิลด์ในการระบุตำแหน่งของชิ้นส่วนย่อย เพื่อให้ผู้รับทราบว่าเป็นตำแหน่งที่เท่าไร ของแพคเก็ตนั้น

Fragment offset

ขนาด 13 บิต

เป็นลำดับของชิ้นส่วนย่อยที่เท่าไร ของตัวแพ็กเก็ตที่จะส่งในกรณีที่แพ็กเก็ตนั้นมีขนาดใหญ่เกินจำเป็นที่จะต้องทอนเป็นส่วนย่อย ๆ เพื่อให้ผู้รับสามารถนำชิ้นส่วนย่อย ๆ เหล่านี้ กลับมาจัดเรียงลำดับได้อย่างถูกต้อง

Time – To – Live

ขนาด 8 บิต

เป็นฟิลด์ที่ใช้กำหนดอายุของแพคเก็ต ว่าจะมีอายุนานสักเท่าไร เมื่อได้ส่งออกไปในระบบเครือข่ายแล้ว โดยค่าในฟิลด์ดังกล่าวจะลดลงไปเรื่อย ๆ ทีละ 1 เมื่อได้ผ่านตัวค้นหาเส้นทาง 1 ครั้ง แต่ในบางครั้งอาจจะลดลงได้มากกว่าหนึ่ง ในกรณีที่ตัวค้นหาเส้นทางนั้น ๆ ทำงานช้า ค่าสูงสุดที่สามารถจะกำหนดได้คือ 255 หรือเทียบเท่ากับ 4.25 นาที การส่งข้อมูลที่สำเร็จ แพคเก็ตจะต้องถึงปลายทาง ก่อนที่ค่าในฟิลด์นี้จะเป็น '0'

Protocol

ขนาด 8 บิต

กำหนดชนิดของโพรโตคอลที่ใช้ในชั้นที่สูงขึ้นไป เพื่อให้ทางฝ่ายรับผ่านแพ็กเก็ตจากชั้นอินเทอร์เนตไปสู่ชั้นโฮสต์ต่อโฮสต์ได้อย่างถูกต้อง หากเป็น TCP จะมีค่าเป็น '6' ส่วน UDP จะมีค่าเป็น 17

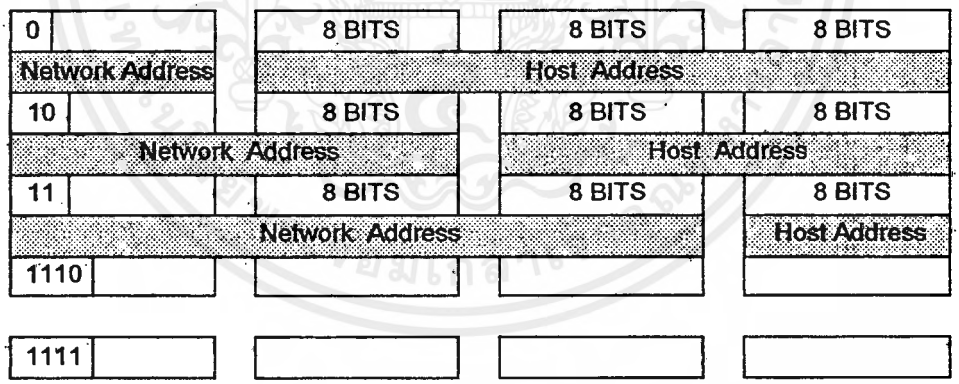
Header Checksum

ขนาด 16 บิต

ใช้เพื่อตรวจสอบความถูกต้องของข้อมูลที่อยู่ในแพคเก็ต ในการส่งบนระบบเครือข่ายทั้งนี้ค่าดังกล่าวจะต้องคำนวณใหม่ทุกครั้งที่มีการค้นหาเส้นทาง เพราะค่าของฟิลด์ Time – To – Live เปลี่ยนไป

Source and Destination Addresses

ในส่วนของ IP แพคเก็ต จะมีฟิลด์เก็บหมายเลขที่อยู่ต้นทาง และปลายทางไว้ ซึ่งหมายเลขที่อยู่นี้จะเป็นแบบลอจิคัลหรือที่คนส่วนใหญ่มักจะเรียกกันว่า IP Address โดยจะเป็นชุดของตัวเลข 4 ไบต์ แต่ละไบต์จะคั่นด้วย '.' สามารถแบ่งออกได้เป็น 5 ชั้น ดังนี้



รูปที่ 2.32 แสดงการจัดแบ่งชั้นของค่าตำแหน่งที่อยู่ IP

Class A

จะเริ่มจากเลข 0-127 หรือถ้าเป็นเลขฐานสองก็จะขึ้นต้นด้วย '0' ในส่วนไบต์แรกจะใช้ระบุหมายเลขที่อยู่เนตเวอร์ก อีก 3 ไบต์หลังจากใช้ระบุหมายเลขที่อยู่โฮสต์ ใน Class A นี้จะใช้กับเนตเวอร์ก ที่ภายในประกอบด้วยโอสต์ จำนวนมาก

Class B

จะเริ่มจากเลข 128-191 หรือถ้าเป็นเลขฐานสองก็จะขึ้นต้นด้วย '10' ในส่วนของสองไบต์แรกจะระบุหมายเลขที่อยู่เนตเวอร์กและอีกสองไบต์หลัง จะใช้ระบุหมายเลขที่อยู่โฮสต์ มักใช้กับเครือข่ายขนาดกลาง

Class C

จะเริ่มจากเลข 192-223 หรือถ้าเป็นเลขฐานสองก็จะขึ้นต้นด้วย '110' ในส่วนสามไบต์แรกใช้ระบุ หมายเลขที่อยู่เนตเวอร์กและไบต์สุดท้ายจะใช้ระบุหมายเลขที่อยู่โฮสต์ จึงใช้กับเครือข่ายที่มีขนาดเล็ก จำนวนเครื่องโฮสต์ ไม่มาก

Class D

จะเริ่มจากเลข 224-239 หรือถ้าเป็นเลขฐานสองก็จะขึ้นต้นด้วย '1110' ใช้กับแพคเก็ตที่ต้องการส่งเป็นแบบหลายจุด

Class E

จะเริ่มจาก 240-255 หรือถ้าเป็นเลขฐานสองก็จะขึ้นต้นด้วย '1111' ยังไม่นำมาใช้จริง กำลังอยู่ในช่วงศึกษา

Options

ขนาด ไม่เกิน 320 บิต

เป็นส่วนออฟชั่นที่เพิ่มขึ้นมา ใช้กับเฉพาะการใช้งานบางประเภทเช่น ใช้เพื่อระบุการค้นหาเส้นทางของแพคเก็ต ใช้ระบุการค้นหาเส้นทางของแพคเก็ต ใช้ระบุค่าเวลา (TimeStamp) ใช้ในด้านการรักษาความปลอดภัยของกระทรวงกลาโหมสหรัฐฯ

Option	Description
Security	Specifies how secret the Datagram is
Strict sotrce routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

ตารางที่ 2.6 ส่วนข้อมูลเพิ่มเติมใน Package IP

Security

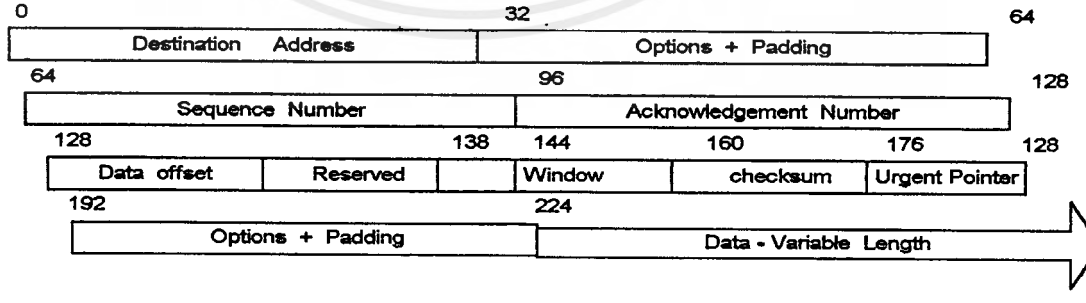
ใช้กำหนดประเภทข้อมูลที่เป็นความลับ เขตข้อมูล Strict source routing ใช้กำหนดที่อยู่ของแม่ขายทั้ง

หมดตั้งแต่ Host ผู้ส่ง Host ที่ถูกบังคับให้รับ-ส่ง Datagram นี้ ไปจนถึง Host ของผู้รับฯ เขตข้อมูล Record route ใช้ในการบันทึกหมายเลขที่อยู่ของ Host ทั้งหมดที่รับ-ส่ง Datagram นี้เป็นต้น

2.4.2.2 TCP แพคเก็ต

TCP เป็นโพรโตคอล ที่ทำหน้าที่ในส่วนของชั้นทรานสปอร์ต ทำงานเป็นแบบ Connection-Oriented ใช้รองรับการทำงานโพรโตคอล Telnet, FTP และ SMTP ในระดับชั้นบน

รายละเอียดในแต่ละฟิลด์ของ TCP แพคเก็ต



รูปที่ 2.33 แสดงฟิลด์ต่าง ๆ ของ TCP แพคเก็ต

Source Port

ขนาด 16 บิต

เป็นฟิลด์ที่ระบุหมายเลขของโพรโตคอลในชั้นบน ไม่ว่าจะเป็น Telnet, FTP หรือ SMTP เพื่อให้ส่งผ่านข้อมูลในระหว่างระดับชั้นอย่างถูกต้อง

Destination Port

ขนาด 16 บิต

ลักษณะเหมือนกับฟิลด์ Source Port แต่เปลี่ยนจากต้นทางมาเป็นปลายทางเท่านั้น

Port Number	Description	Port Number	Description
0	Reserved	23	Telnet
5	Remote Job Entry	25	SMTP
7	Echo	37	Time
9	Discard	53	Name Server
11	Systat	102	ISO-TSAP
13	Daytime	103	X.400
15	Netstat	104	X.400 sending service
17	Quotd(Quote of the	11	Sum RPC
20	day)	139	NetBIOS session source
21	Ftp_data	160--223	Reserved
	Ftp(Control)		

ตารางที่ 2.7 แสดงค่าหมายเลขพอร์ตต่างๆ ในส่วนของฟิลด์ Destination และ Source Port

Sequence Number

ขนาด 32 บิต

ใช้เพื่อทำให้การส่งแพคเก็ต เป็นไปตามลำดับถูกต้อง ตามรูปแบบของวงจรเสมือน

Acknowledgement Number

ขนาด 32 บิต

Acknowedgement Number นำมาใช้เพื่อการตอบรับการได้รับแพคเก็ตแล้ว โดยกำหนดบิต ACK ในฟิลด์ Flag เซต ในฟิลด์นี้จะระบุหมายเลขลำดับไบต์ ของข้อมูลที่คาดว่าจะได้รับในครั้งต่อไป ตัวอย่างเช่น สมมติว่ามีการติดต่อกันระหว่างคู่โฮสต์คู่หนึ่ง ซึ่งต้องมีการส่งแพ็กเก็ตติดต่อกันจำนวนมากและแต่ละแพ็กเก็ตมีขนาดเท่ากับ 20 ไบต์ หมายเลขลำดับของการส่งจะเป็น 40, 60, 80, เพราะฉนั้นเมื่อสิ้นสุดการส่งแพคเก็ตชุดที่ 3 แล้ว แพคเก็ตตอบรับ (Acknowledge Packet) ซึ่งผู้รับจะส่งตอบกลับไปหาผู้ส่ง จะมีหมายเลข Acknowledge เท่ากับ 100 ซึ่งหมายความว่า ข้อมูลก่อนหน้านี้ (40 ถึง 99) ได้รับเรียบร้อยแล้ว (ที่เป็น 100 เพราะว่า 100 เป็นหมายเลขลำดับต่อไปถัดจาก 80)

Data Offset

ขนาด 4 บิต

เพื่อบ่งบอกความยาวของเฮดเดอร์ TCP ในหน่วยเวิร์ด ซึ่งจะเปลี่ยนแปลงได้ตามขนาดของฟิลด์ Option

Reserved

ขนาด 6 บิต

สำรองใช้ในอนาคต

Flags

ขนาด 6 บิต (1 บิต ต่อ 1 Flag)

ในฟิลด์ Flags นี้จะเก็บข้อมูลควบคุมการติดต่อของ TCP ดังนี้

- URG(Urgent) จะเซตเพื่อบอกว่าฟิลด์ Urgent Pointer ทำงานอยู่ด้วย
- ACK(Acknowledge) จะเซตเพื่อบอกว่าเป็นแพคเก็ตตอบรับ โดยมีหมายเลขลำดับการตอบรับอยู่ในฟิลด์ Acknowledge Number
- PUSH ถ้าเป็นฝ่ายส่งจะบอกให้ส่งผ่านข้อมูล ไปยังโพรโตคอลชั้นล่างทันที แต่ถ้าเป็นผู้รับจะทำงานกลับกันคือ ส่งผ่านไปยังชั้นบนทันที่
- SYN (Synchronize) เป็นแพคเก็ตเพื่อใช้สร้างวงจรเสมือนระหว่างผู้รับและผู้ส่ง โดยผู้ส่งจะต้องส่งแพคเก็ตที่มีบิต Syn เซต หมายเลขลำดับเป็น A ฝ่ายผู้รับก็จะส่งแพคเก็ตที่มีบิต

Syn เซต และบิต ACK เซตเพื่อตอบรับกลับ โดยมีหมายเลขการตอบรับกลับเป็น (A)+1 ฝ่ายผู้ส่งครั้งแรกจะตอบรับกลับโดยส่งแพคเก็ตที่มีเฉพาะบิต ACK เซตและหมายเลขลำดับการตอบรับเป็น (B)+1

- FIN (Finish) เพื่อบ่งบอกว่า ส่งข้อมูลครบหมดแล้ว การติดต่อที่ได้สร้างไว้ยกเลิกได้

Window

ขนาด 16 บิต

เพื่อบอกว่าขนาดจำนวนไบต์ที่ต้องการจะได้รับกลับมา

Checksum

ขนาด 16 บิต

เพื่อตรวจสอบความถูกต้องของข้อมูล ถ้าหากว่าค่าที่อยู่ในฟิลด์นี้ กับที่ได้มาจากที่คำนวณของผู้รับข้อมูลมีขนาดไม่เท่ากัน ข้อมูลชุดดังกล่าวจะถูกยกเลิกไป

Urgent Pointer

ขนาด 16 บิต

จะเป็นค่าออฟเซตที่ชี้ไปยังตำแหน่ง ข้อมูลรีบด่วน ของแพคเก็ต

Option

ขนาด ไม่แน่นอน

เป็นออฟชั่นที่เพิ่มขึ้นมา เพื่อเพิ่มความสามารถในการทำงาน อาทิเช่น ขยายขนาดเซกเมนต์ของ TCP

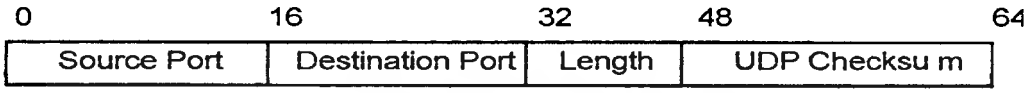
Data

เก็บข้อมูลส่วนที่จะส่งไปยังโพรโตคอลในชั้นบนต่อไป

2.4.2.3 UDP แพคเก็ต

นำมาใช้งานแทนที่ TCP มีการทำงานแบบ **Connectionless – Oriented** ไม่ต้องมีการตอบรับข้อมูล ทำให้ลดภาระ (Overhead) ลงได้มาก หน้าที่หลักของ UDP เพียงเพื่อรับและส่งผ่านข้อมูลไปยังโพนโตคอลในชั้นบนขึ้นไป

รายละเอียดในแต่ละฟิลด์ของ UDP แพคเก็ต



รูปที่ 2.34 แสดงค่าฟิลด์ต่าง ๆ ของ UDP แพคเก็ต

Source Port

ขนาด 16 บิต

เป็นฟิลด์ที่ระบุหมายเลขของโพรโตคอลในชั้นบน ไม่ว่าจะเป็น Telnet , FTP หรือ SMTP เพื่อให้ส่งผ่านข้อมูลในระหว่างระดับชั้นถูกต้อง

Destination Port

ขนาด 16 บิต

ลักษณะเหมือนกับ Source Port แต่เปลี่ยนจากต้นทางมาเป็น ปลายทางเท่านั้น

Length

ขนาด 16 บิต

ใช้บอกความยาวของ UDP แพคเก็ต มีหน่วยเป็นจำนวนไบต์

UDP Checksum

เป็นเพียงออฟชั่น เพื่อตรวจสอบความถูกต้อง เมื่อผู้รับได้รับข้อมูล

หมายเหตุ TCP ทำงานแบบควบคุมลำดับการรับส่งข้อมูล และต้องสร้างการติดต่อกันก่อน แต่ UDP ไม่ต้อง ทำให้ทำงานได้เร็วกว่า เนื่องจากมีภาระน้อย

2.4.2.4 ICMP แพคเก็ต

ICMP ย่อมาจาก Internet Control Message Protocol เป็นกฎระเบียบที่ใช้ในการตรวจสอบ และรายงานสถานภาพของ Datagram โดยทั่วไปมีการรายงานอยู่ 9 แบบคือ

• **Destination unreachable** หมายถึง Router ไม่สามารถหาที่อยู่ผู้รับได้ เกิดขึ้นจาก 2 กรณี คือ 1. Router ไม่มีข้อมูลที่อยู่ของผู้รับ 2. Datagram นั้นเป็นแบบ DF คือไม่ยอมให้แบ่งออกเป็น Datagram เล็กๆ แต่ Datagram ที่ส่งมายัง Router นั้นใหญ่เกินกว่าที่ Router จะจัดการได้

• **Time exceeded** ตามที่ได้กล่าวถึงในข้างต้นเกี่ยวกับค่าที่เก็บไว้ใน Time to live ของ Datagram ซึ่งจะ
ถูกลดค่าลงไปเรื่อยๆ เมื่อค่านี้ถูกลดลงต่ำมากหรือมีค่าเป็น 0 แสดงว่าข้อมูลนี้อยู่ในระบบนานมาก แต่ก็ยังเดินทางไปไม่ถึงผู้รับ Router ที่รับข้อมูลนี้จะทำลายข้อมูลทิ้งและส่งสารกลับไปยังผู้ส่ง

• **Parameter problem** ในทุกครั้งที่ Router รับข้อมูล ก็จะตรวจสอบความถูกต้องของข้อมูล (ตามที่กล่าวไว้ในเรื่อง Header checksum) ถ้าไม่ถูกต้อง Router ที่รับข้อมูลนี้จะทำลายข้อมูลทิ้งและส่งสารกลับไปบอกผู้ส่ง

• **Source quench** หมายถึง Router ที่เป็นฝ่ายรับข้อมูลไม่สามารถจัดการกับข้อมูลที่ส่งเข้ามาได้ทันจึงส่งสารนี้ไปยัง Router ที่ส่งข้อมูลให้หยุดส่งข้อมูลชั่วคราว แต่ในความจริงวิธีการนี้ ไม่เป็นที่นิยมใช้อีกต่อไป

• **Redirect Router** ที่รับข้อมูลจะบอกให้ผู้ส่งทราบว่าไม่สามารถส่งข้อมูลได้อีกต่อไป เนื่องจากเส้นทางเดินที่กำหนดนั้นไม่มีอยู่

• **Echo request** ใช้ในการตรวจสอบว่า Host ที่ต้องการติดต่อด้วยนั้นมีอยู่ในระบบเครือข่ายหรือไม่

• **Echo reply** ใช้ในการตอบการตรวจสอบว่า Host ที่ต้องการติดต่อด้วยนั้นมีอยู่ในระบบเครือข่ายจริง

• **Timestamp request** มีวัตถุประสงค์เช่นเดียวกับ Echo request เพียงแต่เพิ่มเวลาที่สอบถามเข้าไปด้วย

• **Timestamp reply** มีวัตถุประสงค์เช่นเดียวกับ Echo reply เพียงแต่เพิ่มเวลาที่สอบถามและเวลาที่ตอบรับเข้าไปด้วย ใช้ในการตรวจสอบประสิทธิภาพของเครือข่าย

Massage type	Description
Destination unreachable	Package could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo request	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request. But with timestamp
Timestamp reply	Same as Echo reply. But with timestamp

ตารางที่ 2.8 ประเภทของข้อมูลในระบบ ICMP

2.4.2.5 ARP แพคเก็ต

แม้ว่า Computer ทุกเครื่องบน Internet จะมีที่อยู่เป็นหมายเลข IP อย่างน้อยหนึ่งหมายเลข แต่โปรแกรมสื่อสารชั้นเชื่อมต่อข้อมูลไม่สามารถนำหมายเลขที่อยู่นี้ไปใช้ในการส่งแพ็กเก็ตได้ เนื่องจากไม่เข้าใจวิธีการใช้และความหมายของที่อยู่แบบ IP ในปัจจุบัน Host ส่วนมากจะเชื่อมต่อกับเครือข่ายเฉพาะบริเวณผ่านอุปกรณ์สื่อสารเครือข่ายที่เข้าใจวิธีการกำหนดที่อยู่แบบเครือข่ายเฉพาะบริเวณเท่านั้น เช่น อุปกรณ์สื่อสารเครือข่ายแบบ Ethernet จะมีการกำหนดที่อยู่โดยใช้เลขฐานสองขนาด 48 Bits บริษัทผู้ผลิตอุปกรณ์ฯเหล่านี้จะต้องขอหมายเลขที่อยู่จากองค์กรที่ควบคุมการกำหนดหมายเลขที่อยู่แบบ Ethernet เพื่อไม่ให้เกิดมีเลขที่อยู่ซ้ำกัน อุปกรณ์สื่อสารเครือข่ายแบบ Ethernet ที่ถูกนำไปใช้งานจริงจึงมีที่อยู่ของตนเองที่ไม่ซ้ำกับใคร ในเวลาเดียวกันอุปกรณ์เหล่านี้ก็ไม่รู้จักวิธีการกำหนดที่อยู่แบบ IP ซึ่งมีขนาด 32 Bits เลย

ผู้ใช้ของ Host 1 ส่งข้อมูลไปถึงผู้ใช้ของ Host 2 ได้ดังนี้ สมมุติให้ผู้รับที่ Host 2 นั้นคือ mary@eagle.cs.uni.edu ขั้นตอนแรกคือจะต้องค้นหาหมายเลข IP ของ Host 2 ซึ่งใช้ชื่อว่า eagle.cs.uni.edu ในระบบเช่นนี้จะต้องมีเครื่อง Computer เครื่องหนึ่ง เรียกว่า ผู้ให้บริการรายชื่อ (Domain name sever) ซึ่งจะให้คำตอบแก่ Host 1 ว่าหมายเลข IP ของ Host 2 คือ 192.31.65.5

Package ข้อมูลจะถูกสร้างขึ้นมาโดยใช้หมายเลข 192.31.65.5 เป็นที่อยู่ผู้รับ แล้วส่งต่อไปให้โปรแกรม IP ซึ่งจะเป็นผู้ส่ง Package นี้ออกไป โปรแกรม IP ทราบจากหมายเลข IP ของผู้รับว่าผู้รับอยู่ในเครือข่ายเดียวกัน ขั้นต่อไปจึงต้องการทราบหมายเลขที่อยู่ Ethernet ของผู้รับ หนทาง

หนึ่งที่จะได้คือ การสร้างตารางข้อมูลไว้ในเครือข่ายเพื่อแปลงที่อยู่ IP เป็นที่อยู่แบบ Ethernet วิธีการนี้นำมาใช้กับองค์กรขนาดเล็กได้ แต่ไม่เหมาะกับองค์กรขนาดใหญ่

วิธีการ ARP ช่วยแก้ปัญหาในการค้นหาที่อยู่ Ethernet ของข้อมูลที่ใช้การกำหนดที่อยู่แบบ IP แต่ถ้าทราบที่อยู่เป็นแบบ Ethernet แล้วต้องการแปลงที่อยู่เป็น IP จะทำอย่างไร ปัญหานี้มักเกิดขึ้นกับเครื่อง Computer ที่เริ่มทำงานด้วยการอ่านข้อมูลทั้งหมดจากเครื่อง Host (Diskless Workstation) เครื่องประเภทนี้จะทราบเพียงที่อยู่ Ethernet ของตนเองจากอุปกรณ์สื่อสารเครือข่ายเท่านั้น

การค้นหาคำตอบสามารถทำได้โดยวิธีควบคุมการสื่อสารแบบ ARP ย้อนกลับ หรือ RARP (Reverse Address Resolution Protocol) วิธีการนี้ Computer ที่เพิ่งจะเริ่มทำงาน (หรือเครื่องใดก็แล้วแต่) จะส่งคำถามออกไปในทำนอง "ที่อยู่ขนาด 48 Bits แบบ Ethernet ของฉันคือ 14.04.05.18.01.25 มีใครทราบที่อยู่ IP ของฉันบ้าง" เครื่องที่ให้บริการ RARP จะตรวจดูข้อมูลในตารางข้อมูลของตนเองแล้วจึงส่งหมายเลข IP กลับไปให้

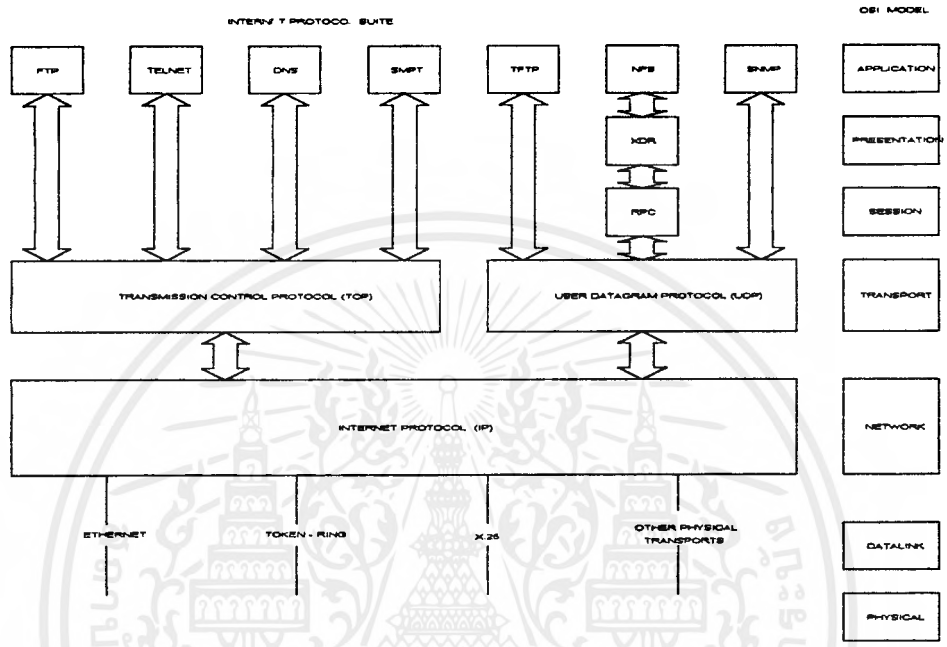
วิธีการนี้ช่วยให้เกิดความอ่อนตัวและเพิ่มประสิทธิภาพในการใช้หมายเลข IP เนื่องจากผู้ใช้ไม่มีหมายเลข IP เป็นของตนเองผู้ควบคุมระบบสามารถกำหนดหมายเลข IP ใดๆที่ไม่มีผู้ใช้งานในขณะนั้นให้ใช้ได้ หมายเลข IP ในที่นี้จึงเป็นเสมือนสมบัติส่วนกลางที่ทุกคนใช้ร่วมกัน

ข้อด้อยของวิธี RAPRP คือการที่ผู้ใช้จะส่งคำถามโดยใช้หมายเลข 1 จำนวน 48 ตัวเป็นที่อยู่ของผู้ให้บริการ หมายเลขนี้เป็นหมายเลขพิเศษที่ Router จะไม่ยอมส่ง Package ผ่านไปยังเครือข่ายอื่นเลย ฉะนั้นผู้ให้บริการ RARP จะต้องมีอยู่ประจำทุกเครือข่าย อย่างไรก็ตาม Protocol แบบ BOOTP ได้รับการพัฒนาขึ้นมาเพื่อแก้ปัญหานี้โดยการใช้ Package UDP แทน Package ชนิดนี้สามารถส่งไปได้ทั่วทุกเครือข่ายและยังให้ข้อมูลอื่นเพิ่มเติม เช่น หมายเลข IP ของผู้ให้บริการแฟ้มข้อมูล หมายเลข IP ของ Router อัตโนมัติ และตารางข้อมูลเครือข่ายย่อยเป็นต้น

ไม่ว่าจะใช้วิธีใดก็ตาม Host 1 จะบรรจุ Package IP ไว้ใน Package Ethernet ที่ส่งไปให้ E3 เมื่อไปถึง Router จะดึง Package IP ออกมาแล้วค้นหาหมายเลข IP ในตางเส้นทางเดินข้อมูลภายในก็จะพบว่า Package นี้จะต้องส่งไปที่ Router 192.31.60.7 ถ้า Router E3 ไม่ทราบที่อยู่ของ FDDI ของ Router 192.31.60.7 ก็จะใช้วิธี ARP เพื่อถามหาที่อยู่ที่ต้องการ เมื่อได้รับคำตอบแล้ว (คือ F3) Router E3 ก็จะบรรจุ Package IP ไว้ใน Package FDDI แล้วส่งไปยัง F3 ที่ Router ของคณะวิศวกรรมไฟฟ้า (E4) Program FDDI จะดึง Package IP ออกมาส่งให้กับ Program IP ซึ่งจะรู้ว่าจะต้องส่ง Package ไปให้ 192.31.63.8 ในทำนองเดียวกันถ้า E3 ไม่มีข้อมูล 192.31.63.8 อยู่ในตารางข้อมูลของตนเองก็จะใช้วิธี ARP เพื่อถามหาที่อยู่ และจะได้รับคำตอบคือ E6 ต่อไป Router E4 ก็จะสร้าง Package Ethernet โดยบรรจุ Package IP ไว้ภายในแล้วส่งออกไปให้ E6 ท้ายที่สุด Package จะมาถึง Host 4 (E6) Package IP จะถูกดึงออกมาแล้วส่งต่อไปให้โปรแกรม IP ดำเนินการประมวลผลต่อไป

2.4.3 โพรโตคอลซึ่งทำงานในระดับชั้นสูงขึ้นไป

เป็นโพรโตคอลซึ่งมีหน้าที่การทำงานอยู่ตั้งแต่ในระดับชั้นเซสชั่น จนถึงชั้นแอพพลิเคชัน ตามมาตรฐานของ OSI การทำงานของแต่ละโพรโตคอลในระดับชั้นนี้ จะต้องอาศัยการทำงานของโพรโตคอลที่อยู่ในระดับล่าง ไม่ว่าจะเป็น IP , TCP หรือ UDP



รูปที่ 2.35 แสดงการทำงานของโพรโตคอลที่อยู่ในระดับชั้นบน

Remote Terminal Emulation – TELNET

เป็นโพรโตคอลที่จะจำลองเครื่องคอมพิวเตอร์ ทำงานเป็นเทอร์มินัลของเครื่องโฮสต์ ได้ ซึ่งส่วนใหญ่แล้วเครื่องคอมพิวเตอร์จะมีความสามารถสูงกว่า ที่ตัวเทอร์มินัลเองจำเป็นต้องใช้อยู่แล้ว

File Transport Protocol (FTP) และ Trivial File Transfer Protocol (TFTP)

FTP จะทำให้สามารถโอนถ่ายแฟ้มข้อมูลต่าง ๆ ระหว่างเครื่องโฮสต์ได้ การทำงานของ FTP จะต้องทำการ Login ก่อนและจะมีการตรวจสอบลิขสิทธิ์ที่ของผู้ใช้นั้นๆ บนแฟ้มข้อมูลก่อน ส่วนการทำงานของ TFTP ไม่ต้องทำการ Login ทำให้ลดภาระในส่วนนี้ไปได้ แต่ยังคงต้องมีการตรวจสอบลิขสิทธิ์ต่าง ๆ ก่อนเช่นเดียวกัน

Simple Mail Transfer Protocol (SMTP)

เป็นโพรโตคอลที่เครื่องโฮสต์ ใช้จัดการกับจดหมายอิเลกทรอนิกส์

Routing Information Protocol (RIP)

RIP เป็นโพรโตคอลที่ใช้ดูแลและจัดการข้อมูลที่เกี่ยวกับการค้นหาเส้นทาง โดยจะมีการส่ง RIP ออกไปตามช่วงเวลาที่กำหนด เพื่อที่จะทำการเปลี่ยนแปลงข้อมูลของตารางการค้นหาเส้นทาง ซึ่งเก็บอยู่ในเครื่องโฮสต์และตัวค้นหาเส้นทาง เพื่อทำให้การค้นหาเส้นทางมีประสิทธิภาพสูงสุด

Network File System (NFS)

เป็นโพรโตคอลที่พัฒนาขึ้นโดย Sun Microsystems บนแฟลตฟอร์ม(Platform) ONC เพื่อเมาท์ hard drive ของเครื่องที่อยู่บนเครือข่าย ให้ทำงานได้เสมือนเป็นโลคัลไดเรกทอรี่หนึ่ง เพื่อที่จะใช้ทรัพยากรณ์เครื่องที่ทำการเมาท์ได้ ถึงแม้ว่าจะเป็นระบบปิด (proprietary) โดยจะต้องใช้ 3 โพรโตคอลประกอบกัน คือ NFS, XDR (External Data Representation) และ RPC(Remote Procedure Call)

External Data Representation (XDR)

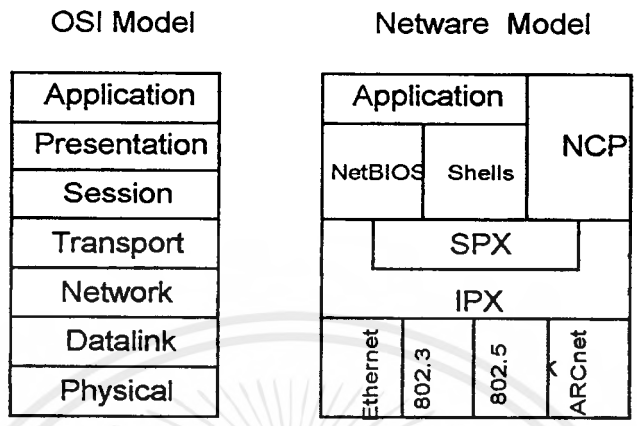
XDR จะเป็นกลุ่มของไบนารีรูทีน ซึ่งเรียกใช้ได้โดยภาษาซี มีความสามารถในการใช้ข้อมูลของระบบอื่นโดยไม่ต้องกังวลในเรื่องของแพลตฟอร์มที่แตกต่างกัน

Remote Procedure Call (RPC)

ให้ระบบสามารถใช้ข้อมูล ได้ทั้งระบบของตนเอง, จากตัวไคลเอนต์ (Client) หรือจากระบบอื่น ๆ

2.5 ชุดเนตแวร์โพรโตคอล

เนตแวร์โพรโตคอล XNS (Xerox Netware System) ซึ่งพัฒนาโดยบริษัท XEROX สถาปัตยากรรมของโพรโตคอล XNS สามารถเทียบเคียงได้กับ OSI ดังนี้



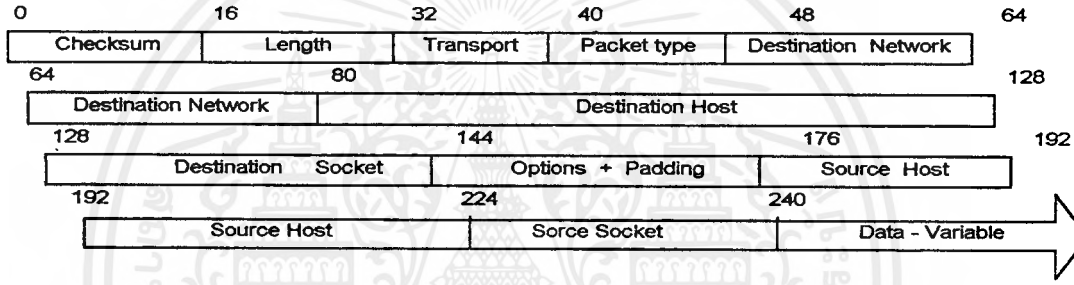
รูปที่ 2.36 แสดงการเปรียบเทียบมาตรฐาน OSI กับชุดอินเตอร์เนตโพรโตคอล

ระดับชั้นฟิสิคัลและดาตาลิงค์ จะใช้มาตรฐานของอีเทอร์เนต , IEEE 802.3 ,802.5 และอาร์กเนต ระดับชั้นเนตเวิร์กเทียบได้กับ IPX (Internet Packet Exchange) ชั้นทรานสปอร์ตเทียบได้กับ Seqienced Packet Exchange (SPX) ชั้นเซสชันจะเป็น NetBIOS (Network Basic Input Output System) ชั้นพรีเซนเตชั่นและชั้นแอพพลิเคชันเปรียบได้กับ Netware Core Protocol (NCP) และ Netware Value – Added Services

2.5.1 IPX แพคเก็ต

Internet Packet Exchange (IPX) เป็นโพรโตคอลที่ทำงานแบบ Connectionless รับหน้าที่การค้นหาเส้นทางให้กับแพ็กเก็ต จากที่อยู่ต้นทางไปยังที่อยู่ปลายทาง เดิมที IPX ทำงานได้เหมือนกับ IDP (Internetwork Datagram Protocol) ของ XNS แต่เนื่องจากต้องทำงานการค้นหาเส้นทางด้วยโดยจะเป็น RIP ของ IPX โดยทำงานคล้ายคลึงกับ RIP ของชุดอินเตอร์เนตโพรโตคอล เนตแวร์เซอร์ฟเวอร์และตัวค้นหาเส้นทาง จะส่งข้อมูลที่อยู่กระจายออกไปยังโหนดอื่นๆ ทุก ๆ ช่วงเวลาที่กำหนดเพื่อเป็นข้อมูลในการจัดทำตารางการค้นหาเส้นทาง

รายละเอียดในแต่ละฟิลด์ของ IPX แพคเก็ต



รูปที่ 2.37 แสดงฟิลด์ต่าง ๆ ของ IPX แพคเก็ต

Checksum

ขนาด 16 บิต

ใช้ตรวจสอบความถูกต้องของตัวเฮดเดอร์ XNS แต่ฟิลด์นี้จะไม่ใช้ในกรณีที่เป็นระบบเครือข่ายเดี่ยว เพราะถือว่าในระบบเครือข่ายเดี่ยว มีความน่าเชื่อถือสูงอยู่แล้ว

Length

ขนาด 16 บิต

จะแสดงความยาวทั้งหมดของ IPX แพคเก็ต โดยความยาวนั้นจะมีค่าตั้งแต่ 30 ไบต์ไปจนถึง 576 ไบต์

Transport Control

ขนาด 8 บิต

ใช้กำหนดอายุของแพคเก็ตที่จะส่งออกไปบนระบบเครือข่าย ผู้ส่งจะกำหนดค่าในฟิลด์นี้เป็นศูนย์ เมื่อแพคเก็ตนี้ผ่านตัวค้นหาเส้นทางแต่ละตัว ด้วยค้นหาเส้นทางจะเพิ่มค่าในฟิลด์นี้ทีละ 1 และเมื่อไรก็ตามที่ค่าในฟิลด์นี้เป็น 16 แพคเก็ตชิ้นนี้จะถูกยกเลิกทิ้ง

Packet Type

ขนาด 8 บิต

จะเป็นตัวระบุโพรโตคอลในชั้นบนที่ IPX จะส่งผ่านข้อมูลขึ้นไปให้ ค่าที่ใช้ในฟิลด์นี้มีดังนี้

0 Unknown Packet

1 Routing Information Protocol (RIP)

2 Echo Packet

3 Error Packet

4 Packet Exchange Packet (REP)

5 Sequence Packet Protocol (SPP)

17 Netware Core Protocol (NCP)

Destination Network

ขนาด 32 บิต

จะเป็นที่อยู่ของเครือข่ายปลายทาง หากมีค่าเป็นศูนย์ หมายความว่าเครื่องต้นทางและเครื่องปลายทางอยู่บนระบบเครือข่ายเดียวกัน

Destination Host

ขนาด 48 บิต

จะเป็นที่อยู่ทางฟิสิคัลของเครื่องปลายทาง ถ้าหากใช้มาตรฐาน IEEE 802.3 หรือ IEEE 802.5 จะใช้ครบหมดทุกบิต แต่ถ้าใช้อาร์กเน็ต ซึ่งที่อยู่ในมาตรฐานนี้จะใช้ไม่ครบทุกไบต์ หากไบต์ไหนไม่มีการใช้จะเติมเลขศูนย์ที่หน้าไบต์เหล่านั้น แต่หากเป็นการส่งข้อมูลแบบกระจายทุกจุด ค่าในฟิลด์นี้จะเป็น หนึ่งทุกบิต

Destination Socket

ขนาด 16 บิต

จะระบุโพรเซสของเครื่องปลายทางที่ ใช้งานกับแพคเก็ตนี้ ซึ่งมีค่าต่าง ๆ ดังนี้

0451H File server packet

0452H Service advertising packet

0453H Routing Information packet

0455H NetBIOS packet

0456H Diagnostic packet

IPX จะใช้ฟิลด์ Destination Socket และ Source Eocket ระบุโพรเซสที่จะใช้งานในระดับชั้นบนขึ้นไป

Source Network

ขนาด 32 บิต

ลักษณะเช่นเดียวกันกับตัว Destination Network แต่จะเป็นที่อยู่ระบบเครือข่ายของต้นทางแทน ถ้าหากมีค่าเป็นศูนย์ จะเป็นระบบเครือข่ายที่ไม่รู้จัก

Source Host

ขนาด 48 บิต

จะเป็นที่อยู่ทางฟิสิคัลของเครื่องที่เป็นฝ่ายส่งข้อมูล มีรูปแบบคล้ายกับฟิลด์ Destination Host และจะใส่ศูนย์ลงไปแทนไบต์ที่ไม่มีการใช้งาน ถ้าหากเป็นการส่งข้อมูลที่มีการ กระจายทุกจุดทุกบิตในฟิลด์นี้จะเป็นหนึ่ง

Source Socket

ขนาด 16 บิต

มีรูปแบบและการทำงานเช่นเดียวกับ Destination Socket

Data

ขนาด ไม่คงที่

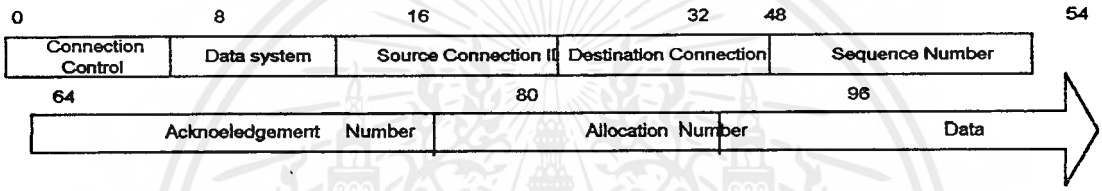
จะเป็นฟิลด์ที่เก็บข้อมูล เพื่อส่งผ่านขึ้นไปยังระดับชั้นบนในกรณีที่เป็นผู้รับข้อมูล หรือใช้เก็บข้อมูลที่ส่งผ่านลงมาจากระดับชั้นบนในกรณีที่เป็นฝ่ายส่ง

2.5.2 SPX แพคเก็ต

Sequenceed Packet Exchange (SPX) ทำงานแบบ Connection-Oriented มีการตรวจสอบการส่งข้อมูลระหว่างเครือข่ายกับเครื่องเซอร์ฟเวอร์ ตัวอย่างการนำโพรโตคอลนี้คือ Rconsole , การพิมพ์ทางไกล (Remote printing) และ SNA เกตเวย์ เพราะการใช้งานเหล่านี้จำเป็นต้องใช้ความน่าเชื่อถือสูงระหว่างที่มีการรับส่งข้อมูล

การทำงานของ SPX มีรูปแบบเป็นวงจรเสมือน ในระหว่างแต่ละคู่ของเครื่องมือที่มีการติดต่อ จะมีหมายเลขเฉพาะ เรียกว่า Connection ID ทำให้ในหนึ่งช่องการติดต่อ (Socket) สามารถมีได้หลายคู่ Connection ID

รายละเอียดในแต่ละฟิลด์ของ SPX แพคเก็ต



รูปที่ 2.38 แสดงฟิลด์ต่าง ๆ ของ SPX แพคเก็ต

Connection Control

ขนาด 8 บิต

ประกอบด้วยค่าแฟลก แบบ 1 บิตจำนวน 4 แฟลก ซึ่งใช้ควบคุมการรับส่งของข้อมูล

01H – 08H	ยังไม่ได้มีการนำมาใช้
10H	สิ้นสุดข้อความ
20H	เตรียมพร้อม
40H	ต้องการการตอบรับกลับ
80H	เป็นแพคเก็ตของระบบ

Datastream Type

ขนาด 8 บิต

เพื่อแสดงประเภทของข้อมูลในด้านที่อยู่ของแพคเก็ต มีลักษณะคล้ายฟิลด์ Type ของอีเทอร์เนต

00H-FDH	กำหนดโดยตัวไคลเอนต์
FEH	บอกสิ้นสุดการติดต่อ
FFH	ตอบรับการการสิ้นสุดการติดต่อ

Source Connection ID

ขนาด 16 บิต

ค่าในฟิลด์นี้ จะถูกกำหนดโดยฝ่ายส่งแพคเก็ต

Destination Connection ID

ขนาด 16 บิต

ใช้เพื่อแยกความแตกต่างของข้อมูล ในกรณีที่มีคู่การติดต่อหลายตัว ติดต่อผ่านทางช่องการติดต่อเดียว

Sequence Number

ขนาด 16 บิต

จะเป็นค่าของหมายเลขแพคเก็ตที่คาดว่าจะได้รับในครั้งต่อไปจากคู่ติดต่ออีกฝ่ายหนึ่ง เพื่อเพิ่มความน่าเชื่อถือให้กับการรับส่งข้อมูล

Allocation Number

ขนาด 16 บิต

แสดงหมายเลขของแพคเก็ต ซึ่งจะสัมพันธ์กับขนาดของบัฟเฟอร์ (Buffer) ที่ทางฝ่ายรับเตรียมไว้ในการรับแพคเก็ต ฝ่ายส่งจะสามารถส่งแพคเก็ตไปให้ฝ่ายรับเรื่อย ๆ จนกว่าค่า Sequence Number กับ Allocation Number เท่ากัน

Data

เก็บข้อมูลที่จะใช้ส่งผ่านขึ้นไปให้กับระดับการทำงานชั้นบนต่อไป

2.6 การเปรียบเทียบการทำงานของ TCP/IP และ SPX/IPX

สิ่งที่เหมือนกันของ IP และ IPX

- ทำงานอยู่ที่ระดับชั้นเนตเวอร์กของ OSI
- การทำงานเป็นแบบดาตาแกรม
- มีโพรโตคอลช่วยในการค้นหาเส้นทางคือ RIP
- การรับส่งข้อมูลจะเป็นแบบ Connectionless
- มีการกำหนดอายุของแพคเก็ตที่ส่งออกไปในระบบเครือข่าย

สิ่งที่เหมือนกันของ SPX และ TCP

- ทำงานอยู่ที่ระดับชั้นทรานสปอร์ตของ OSI
- มีหน้าที่เพื่อเพิ่มความน่าเชื่อถือในการรับส่งข้อมูล โดยใช้การส่งข้อมูลตอบรับเมื่อได้รับข้อมูล เป็นการทำงานแบบวงจรเสมือน
- รูปแบบการทำงานเป็นแบบ ควบคุมการรับส่งข้อมูล

สิ่งที่มีเฉพาะใน TCP/IP แต่ไม่มีใน SPX/IPX

- ข้อมูลของ TCP/IP ได้มีการเผยแพร่ออกสู่สาธารณะ
- การรับส่งข้อมูลเป็นแบบ Sliding วินโดวส์
- IP จะใช้ฟิลด์ Time-To-Live เป็นตัวกำหนดอายุของแพคเก็ตในเครือข่าย
- ที่อยู่แบบ IP จะรวมทั้งที่อยู่ของระบบเครือข่ายและเครื่อง ไว้ในข้อมูลตัวเดียวกัน

สิ่งที่มีเฉพาะใน SPX/IPX แต่ไม่มีใน TCP/IP

- ข้อมูลของ SPX/IPX ไม่ได้นำออกเผยแพร่สู่สาธารณะ
- การรับส่งข้อมูลจะเป็นแบบนับจำนวนบัฟเฟอร์ที่ว่าง พร้อมที่จะรับข้อมูล
- IPX จะใช้ฟิลด์ Transport Control กำหนดอายุของแพคเก็ต
- ที่อยู่แบบ IPX จะแยกกันเป็น 2 ตัวคือที่อยู่ของระบบเครือข่าย และที่อยู่ของเครื่อง

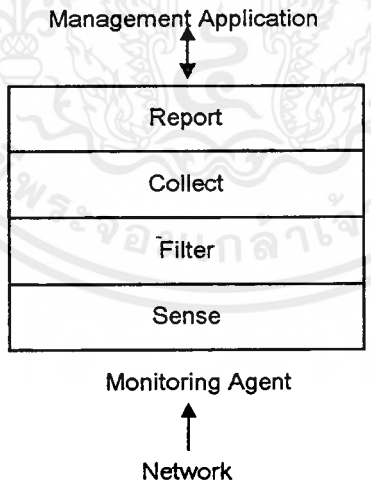
บทที่ 3
หลักการทำงานของโปรแกรมเน็ตเวอร์กมอนิเตอร์ริงโดยทั่ว ๆ ไป

3.1 หลักการทำงานของโปรแกรมเนตเวอร์กมอนิเตอร์ริง
(Design of Network Monitors)

3.1.1 หน้าที่การทำงานทั่วไปของเนตเวอร์กมอนิเตอเตอร์ริงเอเจนท์

มอนิเตอร์ริงเอเจนท์ (monitoring agent) บางตัวไม่ได้มีฟังก์ชันครบทั้งหมด เช่น เรียลไทม์ คาต้าอานาไลเซอร์ (real-time data analyzer) จะแสดงผลข้อมูลไปยังหน้าจอแสดงผล โดยตรงไม่ต้องเก็บข้อมูลลงในสื่อบรรจุ (storage media) เช่น ฮาร์ดดิสก์ แต่จะมีฟังก์ชันพื้นฐานเหมือนกันดังนี้

1. ส่วนรับข้อมูล (Sensing)
2. ส่วนกรองข้อมูล (Filter)
3. ส่วนรวบรวมข้อมูล (Collecting)
4. ส่วนแสดงผล (Reporting)



รูปที่ 3.1 หลักการของแต่ละฟังก์ชันในเนตเวอร์กมอนิเตอร์ริง

3.1.2 ประเภทของการมอนิเตอร์

เนตเวอร์กมอนิเตอร์ริงเอเจนท์ สามารถแบ่งออกได้เป็นประเภทได้ 2 วิธีด้วยกัน

1. แบ่งตามเลเยอร์ที่เอเจนท์ทำการมอนิเตอร์อยู่
2. แบ่งตามลักษณะการมอนิเตอร์คือ เอเจนท์นั้นเป็นแบบ อินทิเกรตเต็ด (Integrated) หรือ เอ็กเทอร์นอล (External) กับ ส่วนประกอบหรือทรัพยากรที่ทำการมอนิเตอร์

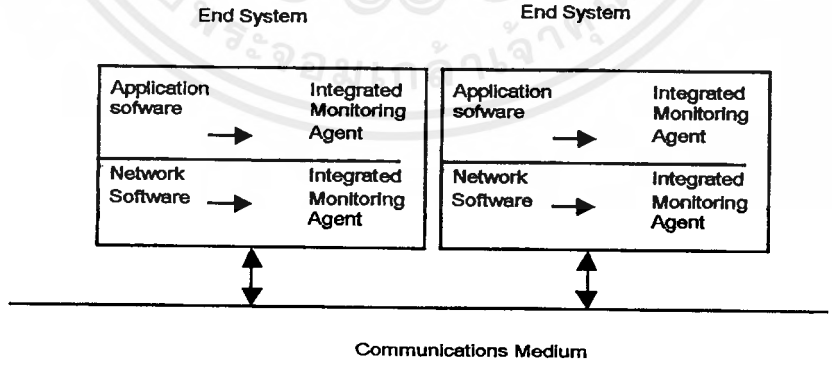
วิธีการแบ่งประเภททั้ง 2 วิธี มีผลต่อการออกแบบและการใช้งานมอนิเตอร์ ดังนั้นเราจะมากล่าวถึงแต่ละประเภทว่ามีลักษณะอย่างไร

3.1.2.1 อินทิเกรตเต็ดมอนิเตอร์ริงเอเจนท์ (Integrated monitoring agents)

อินทิเกรตเต็ดมอนิเตอร์ริงเอเจนท์ รวมไว้เป็นส่วนหนึ่งของเนตเวอร์กแอพพลิเคชัน หรือในตัวของเนตเวอร์กซอฟแวร์เลยก็ได้ ซึ่งในกรณีหลังนั้น เราต้องทำการรวมเอาส่วนต่าง ๆ ต่อไปนี้เข้ามาไว้ด้วย

- เนตเวอร์กเซอร์วิส (network service) ของบางระดับชั้น เช่น รีโมทโพรซีเดอร์คอล (RPC : Remote Procedure Call) คอมมิวนิเคชั่นเซอร์วิส (communication service)
- เนตเวอร์กซอฟแวร์ ในพื้นฐานระบบปฏิบัติการ เช่น เนตเวอร์กดีไวซ์ ไดรเวอร์ (network device driver)
- ส่วนหนึ่งของเนตเวอร์ก (network component) เช่น ซอฟแวร์ หรือ เฟิร์มแวร์ (firmware) ที่ทำงานในฟรอนท์เอ็น คอนเซนเตรเตอร์ (front-end concentrator)

จากตัวอย่างที่กล่าวมานี้ เราจำเป็นต้องมี ตัวนับเหตุการณ์ (event counters) ในทุก ๆ เนตเวอร์กเลเยอร์ ซอฟแวร์ที่ทำหน้าที่นี้ ก็คือ มอนิเตอร์ริงเอเจนท์นั่นเอง



รูปที่ 3.2 ไดอะแกรมของเนตเวอร์กมอนิเตอร์ริงเอเจนท์

ประโยชน์ของอินทิเกรตเต็ดมอนิเตอร์

1. สามารถทำการวิเคราะห์ในระยะไกล (remote diagnostic capability) ได้
2. ในกรณีที่มอนิเตอร์ริงเอเจนท์ รวมอยู่กับแอพพลิเคชั่น เฉพาะอย่าง มันสามารถสนองความต้องการของแอพพลิเคชั่นนั้น ๆ มากกว่าจะทำฟังก์ชันทั่วไป
3. ประหยัด เนื่องจากบางระบบมีบริการของมอนิเตอร์อยู่แล้ว ไม่ต้องเสียค่าใช้จ่ายในส่วนนี้อีก

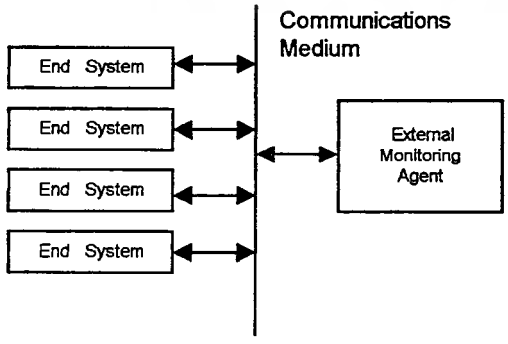
ข้อเสียของอินทิเกรตเต็ดมอนิเตอร์

1. ถ้ามอนิเตอร์ริงเอเจนท์ทำงานบนรูปแบบ (platform) เดียวกันกับตัวบริการด้าน เครือข่าย และแอพพลิเคชั่น ตัวมอนิเตอร์จะทำให้ประสิทธิภาพ ของแอพพลิเคชั่นลดลง
2. ยุ่งยากในการเก็บข้อมูลที่ได้จากมอนิเตอร์ริงเอเจนท์ และค่าหน่วงเวลาที่เกิดขึ้นในช่วงเก็บข้อมูล จะทำให้ข้อมูลบางตัวไม่สามารถเก็บได้ทัน
3. มอนิเตอร์ริงเอเจนท์ ต้องปรับเปลี่ยนไปตามซอฟแวร์ที่มันไปรวมอยู่ด้วย
4. ตัวมอนิเตอร์ จะให้รูปแบบของรายงานที่ต่างกัน สามารถแก้ได้โดยใช้รูปแบบของรายงานที่เป็นมาตรฐาน
5. การมอนิเตอร์เครือข่าย มีความสำคัญน้อย เมื่อเทียบกับแอพพลิเคชั่น หรือ โอเอส

3.1.2.2 เอกซ์เทอร์นอลมอนิเตอร์ (External monitoring agents)

เอกซ์เทอร์นอลมอนิเตอร์เอเจนท์ทำงานในคอมโพเนนท์ที่ไม่เกี่ยวข้องกับซอฟแวร์ที่ทำหน้าที่เนตเวอร์กเซอร์วิส เอเจนท์ชนิดนี้สามารถดักจับได้เฉพาะ แชร์สเตท (share state) ที่ถูกอ้างจากเนตเวอร์กโพรโตคอลเท่านั้น และสามารถดัดแปลงได้ง่าย นิยมใช้ในระบบแลน

เอกซ์เทอร์นอลมอนิเตอร์เอเจนท์ที่ต้องติดต่อกับระบบแลน สามารถนำไปติดตั้งในระหว่างพอยน์ทูพอยน์คอมมิวนิเคชันลิงค์ (point to point communication link) หรือติดต่อกับบัสในระบบคอมพิวเตอร์



รูปที่ 3.3 ไดอะแกรมของเอ็กซ์เทอร์นอลมอนิเตอร์

ประโยชน์ของเอกซ์เทอร์นอลมอนิเตอร์เอเจนท์

1. สามารถทำเป็นฮาร์ดแวร์ได้โดยไม่ไปแบ่งทรัพยากรกับเนตเวอร์กเซอร์วิส และแอพพลิเคชั่น ช่วยลดสาเหตุที่ทำให้ประสิทธิภาพของเนตเวอร์กลดลงโดยมอนิเตอร์ได้ สามารถใช้ฮาร์ดแวร์และซอฟต์แวร์พิเศษ เพื่อใช้สำหรับนำมาทำมอนิเตอร์ริงซึ่งจะไม่มีในมอนิเตอร์ริงเอเจนท์แบบอินทิเกรตเต็ด
2. ประหยัดทั้งด้านราคา และ ทราฟฟิก (traffic)
3. เนื่องจากความเป็นอิสระในการใช้งาน ทำให้ปรับปรุงและขยายการใช้งานได้

ข้อเสีย

1. ความปลอดภัยในการจัดการน้อย
2. ข้อมูลที่ได้ไม่สามารถรับประกันได้ว่าแสดงสถานะจริงๆ ของคอมโพเนนท์ที่เรามอนิเตอร์ จะเป็นเพียงค่าประมาณเท่านั้น
3. เนื่องจากต้องเก็บข้อมูลของเอ็นด์ซิสเต็มจำนวนมาก ตัวเอเจนท์ต้องมีความสามารถสูงมาก ทำให้เพิ่มความยุ่งยากในการผลิตโปรแกรมที่ไม่มีข้อจำกัดในการทำงาน

3.1.3 การประยุกต์ใช้งาน

ปัจจัยที่มีผลต่อการตัดสินใจใช้อินทิเกรตเต็ดหรือเอกซ์เทอร์นอลมอนิเตอร์ริงค์เอเจนท์มีดังนี้

1. แบนด์วิธ (Bandwidth) ของเนตเวอร์กคอมมิวนิเคชั่นมีเดีย (network communication media)
2. สิทธิในการใช้สื่ออื่น ๆ
3. ความเร็วของหน่วยประมวลผล
4. ราคาของหน่วยความจำ

เนตเวอร์กแบนด์วิธ (Network bandwidth)

ถ้าแบนด์วิธของเนตเวอร์กเพิ่มขึ้น จะมีผลต่อเอกซ์เทอร์นอลมอนิเตอร์ริงเอเจนท์เพราะว่ามีข้อมูลจำนวนมากขึ้นที่ต้องการประมวลผล

ใช้สื่อร่วมกัน (Shared media)

ใช้เนตเวอร์กที่เป็นแบบพอยน์ทูพอยน์ลิงค์ที่เหมาะสม ที่จะใช้เอกซ์เทอร์นอลมอนิเตอร์ริงเอเจนท์เนื่องจาก เอเจนท์หนึ่ง ๆ สามารถมอนิเตอร์การติดต่อระหว่าง 2 ระบบปลายทาง (End system) เท่านั้น การจะให้มีการมอนิเตอร์ทุก ๆ ส่วนในเนตเวอร์กต้องใช้ค่าใช้จ่ายสูง

จากเหตุผลที่กล่าวมา ทำให้เราใช้เอกซ์เทอร์นอลมอนิเตอร์ริงเอเจนท์ สำหรับทดสอบ (diagnostic) เป็นส่วนใหญ่ โดยใช้ในการวินิจฉัยเฉพาะลงไปในกรณีที่มีปัญหาเกิดขึ้นในส่วนใดส่วนหนึ่ง

3.1.4 สิ่งที่จะมอนิเตอร์ในแต่ละเลเยอร์

ในแต่ละเนตเวอร์กเลเยอร์ต้องการรายละเอียดในการมอนิเตอร์ดังนี้

1. ดาต้าลิงค์เลเยอร์ ใช้มอนิเตอร์ริงในการตรวจสอบซอฟต์แวร์และฮาร์ดแวร์ที่ผิดพลาด ซึ่งเป็นผลมาจากการคอรัปชั่น(corruption) หรือการสูญหายของข้อมูล

2. เนตเวอร์กเลเยอร์ ใช้มอนิเตอร์ริงในการรายงานถึงเส้นทางการเชื่อมต่อวงจร ว่าใช้การได้หรือไม่

3. ทรานสปอร์ตเลเยอร์ ในเลเยอร์นี้มอนิเตอร์ริงสามารถช่วยในการจัดการเนตเวอร์กและการทำการวางแผน โดยดูจากระดับที่สัมพันธ์ของแต่ละทรานสปอร์ตของโพรโตคอล

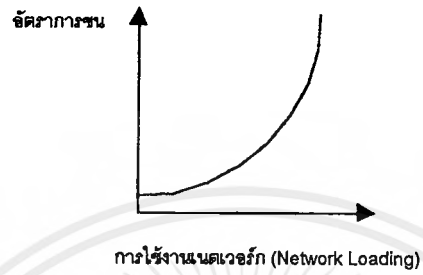
4. เซสชั่นเลเยอร์ ดูเกี่ยวกับปริมาณการใช้งาน (work load) ของเนตเวอร์กที่ใช้โดยแอพพลิเคชั่นและหรือ ผู้ใช้ซึ่งเป็นประโยชน์ต่อการทำ โหลดบาลานซิ่ง (load balancing) และการทำแอกเคานท์ติ้งแอพพลิเคชั่น (accounting application)

5. แอพพลิเคชั่นเลเยอร์ ใช้ตรวจสอบความผิดพลาดของ เซิฟเวอร์โปรเซส (server process) ในไคลน์เซิฟเวอร์แอพพลิเคชั่น เป็นต้น

3.2 ประสิทธิภาพและปัญหาของเนตแวร์แลน

(Netware LANs Performance and Troubleshooting)

3.2.1 บทบาทของการสื่อสารบนเนตเวอร์ก



รูปที่ 3.4 ประสิทธิภาพของอีเทอร์เนต

ในการวัดปริมาณการใช้งานนั้น เราวัดขนาดของข้อมูลต่อเวลาเทียบกับแบนด์วิธ (bandwidth) ซึ่งถ้ามีการใช้งานน้อย นั่นคือน้อยกว่า 5 เปอร์เซนต์ของแบนด์วิธทั้งหมด (total bandwidth) เมื่อมีการใช้งานเพิ่มขึ้นอัตราการชนกัน (collision rate) ของข้อมูลยิ่งมากขึ้น โดยสถิติกล่าวว่าที่ 30 เปอร์เซ็นต์ ประสิทธิภาพการใช้งานของเนตเวอร์กจะลดลงอย่างรวดเร็ว

เพราะฉะนั้นสิ่งที่เราต้องวัดเพื่อที่จะตรวจสอบการใช้งานเนตเวอร์กคือ

- การใช้งานของเนตเวอร์ก
- การใช้งานสูงสุด(peak)
- การใช้งานโดยเฉลี่ย

ซึ่งที่กล่าวมานี้จะช่วยในการนำไป รีอาร์เรนท์จิ้ง (rearranging) งานที่ใช้ทรัพยากรของเนตเวอร์กสูงได้

3.2.2 สิ่งที่ควรจะวัดในเนตเวอร์กทั่วไป

3.2.2.1 เปอร์เซนต์การใช้งานของเนตเวอร์ก

ทำได้โดยการวัดทราฟฟิก (traffic) บนเนตเวอร์กในช่วงเวลาสั้น ๆ สำหรับเปอร์เซนต์การใช้งานของอีเทอร์เนตที่ดีต้องไม่เกิน 15 เปอร์เซ็นต์ นอกเหนือจากนี้ ก็จะมีการวัดการชนกัน (collision) ซึ่งมีผลกระทบต่อประสิทธิภาพ

3.2.2.2 เนตเวอร์กทรูพุท (Network throughput)

วัดจำนวนไบต์ทั้งหมดที่ส่งผ่านเนตเวอร์กในเวลาใด ๆ ซึ่งจะวัดได้ 2 แบบคือ

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

เนื่องจากโพรโตคอลต่าง ๆ มีโอเวอร์เฮด (overhead) ซึ่งมีผลต่อดาต้าทรูพุท ด้วยเหตุนี้เราจึงนำมาใช้พิจารณาในการออกแบบเนตเวอร์กด้วย

ข้อสังเกต อีเทอร์เนต 10 เม็กกะบิตต่อวินาที (Megabit per Second) ไม่ใช่หมายถึงให้ดาต้าทรูพุทสูงสุด 10 เม็กกะบิตต่อวินาที ในการออกแบบเราคำนึงว่า 2.5 เม็กกะบิตต่อวินาที คือ ทรูพุทสูงสุด (Maximum throughput)

3.2.2.3 เวลาตอบสนองของโปรเซสเซิฟเวอร์

(Response time of the file server process)

คือการวัดเวลาตอบสนอง (response) ของไฟล์เซิฟเวอร์ (file server) ต้อการร้องขอ (request) เป็นการวัดประสิทธิภาพของระบบปฏิบัติการและไฟล์เซิฟเวอร์ที่ทำงานอยู่

3.2.2.4 เนตเวอร์กคอนเวอร์เซชั่น (Network conversation)

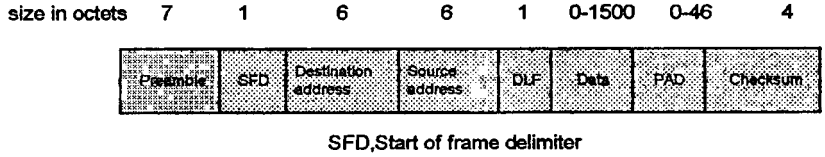
อินดิเซนต์ (indecence) และตำแหน่ง (location) ของคอนเวอร์เซชั่น เป็นสิ่งสำคัญในการนำมาพิจารณาถ้าเกิดมีคอขวด (bottleneck) ขึ้นที่เนตเวอร์กฮาร์ดแวร์ เช่น บริดจ์ และ เราท์เตอร์

3.2.2.5 เก็บบันทึกข้อผิดพลาดในเนตเวอร์ก (recording network error)

เก็บบันทึกข้อผิดพลาดที่เกิดขึ้นในเนตเวอร์ก เพื่อที่สามารถดูข้อมูลผิดพลาดที่เกิดขึ้นภายหลังได้

3.2.3 รูปแบบของแพคเก็ตในอีเทอร์เนตและประสิทธิภาพ

3.2.3.1 อีเทอร์เนตเฟรม



SFD,Start of frame delimiter

DLF,Length of data field

PAD,(optional) padding field

รูปที่ 3.5 โครงสร้างแพคเก็ตอีเทอร์เนต

3.2.3.2 อีเทอร์เนตแอดเดรส (Ethernet address)

บรอดคาสท์ (Broadcast)

มัลติคาสท์ (Multicast)

3.2.3.3 ส่วนหัวของโพรโตคอลระดับบน (Higher level protocol header)

3.2.3.3.1 เนตเวอร์กเลเยอร์

ส่วนหัวโนเวลโพรโตคอล (Novell's IPX header)

ส่วนหัวอินเตอร์เนตโพรโตคอล (IP header) ขนาดเล็กสุดมีขนาด 20 ไบต์

3.2.3.3.2 ทรานสปอร์ตเลเยอร์

ส่วนหัวโนเวลเอสพีเอ็กซ์ (Novell's SPX header) มีขนาด 12 ไบต์

ทีซีพีเฮดเดอร์ ขนาดเล็กที่สุดมีขนาด 20 ไบต์

เพราะฉะนั้นถ้าใช้ ทีซีพี/ไอพี โพรโตคอลจะมีโอเวอร์เฮดอย่างน้อย 66 ไบต์ ส่วนหัวโนเวล ไอพีเอ็กซ์/เอสพีเอ็กซ์ จะมีโอเวอร์เฮด 68 ไบต์

ผลกระทบของขนาดแพคเก็ตกับประสิทธิภาพ

ขนาดของแพคเก็ตยิ่งมีขนาดเล็กเท่าไร โอเวอร์เฮดที่เกิดจากส่วนหัวของโพรโตคอล (protocol header) ก็ยิ่งจะมากขึ้นเท่านั้น เช่น เมื่อมีการส่ง ตัวอักษรเพียงตัวเดียวจาก เทอร์มินอล (terminal) จะพบว่าข้อมูลที่พบจริงจะเป็น 2 เปอร์เซนต์ของเฟรมเท่านั้น แต่ถ้าส่งข้อมูลที่มีขนาดแพคเก็ตสูงสุด พบว่าข้อมูลผู้ใช้คิดเป็น 95 เปอร์เซนต์ของแพคเก็ต จะเห็นว่าขนาดแพคเก็ตเป็นปัจจัยสำคัญของเนตเวอร์ก

ในการส่งข้อมูล ถ้าใช้เฟรมขนาดใหญ่จะช่วยลดจำนวนของแพคเก็ตลง ซึ่งเท่ากับช่วยลดจำนวนการเกิด การชน ดังนั้น โอเวอร์เฮดที่เกิดจาก การชนและการส่งใหม่อีกครั้งจึงลดลงไปด้วย แต่การเพิ่มขนาดของเฟรมให้ใหญ่ขึ้นก็เกิดผลเสียเช่นเดียวกัน ดังนี้ ดูเหมือนเฟรมขนาดใหญ่จะทำให้จำนวนของที่ว่างบนเนตเวอร์กลดลงในขณะที่ทราฟฟิกมีการใช้งานสูง และขนาดบัฟเฟอร์ที่ต้องการโดยซอฟต์แวร์ที่ควบคุมเนตเวอร์กอินเทอร์เฟสต้องเพิ่มขึ้น

แต่จากการค้นคว้าในทางปฏิบัติพบว่า ข้อสำคัญที่ใช้พิจารณาคือ ดาต้าทรูพุท (ตรงข้ามกับมินิมั่มดีเลย์(minimum delay)) ที่ต้องการในระบบเรียลไทม์ (real-time system) ดังนั้นขนาดของเฟรมควรจะมีขนาดใหญ่

3.2.4 สิ่งที่ต้องทำการวัดประสิทธิภาพ

3.2.4.1 สิ่งที่ต้องทำการวัด สำหรับวัดประสิทธิภาพของเนตเวอร์ก

- การใช้งานเนตเวอร์ก (protocol in use)
- การกระจายขนาดของเฟรม (frame size distribution)
- โหนดที่ใช้งานสูงสุด (busy nodes)
- โหนดที่ไม่ได้ใช้งาน (idle node)
- โหนดที่ไม่ได้ตอบสนอง (unresponsive node)
- การสนทนาสูงสุด (busiest conversation)
- เวลาและระดับการใช้งานสูงสุด (peak load time and levels)
- เวลาและระดับการใช้งานต่ำสุด (minimum load time and levels)
- แบนด์วิธที่ใช้ในแต่ละโหนด (bandwidth usage by node)
- แบนด์วิธที่ใช้ในแต่ละโพรโตคอล (bandwidth usage by protocol)

ทั้งหมดนี้เป็น ข้อกำหนดขั้นต่ำของเครื่องมือในการวัดที่ควรพิจารณา นอกเหนือจากนี้แล้ว ยังมีอย่างอื่นที่ต้องการวัด ซึ่งมีประโยชน์เมื่อใช้ในการออกแบบ และการพัฒนาสำหรับประสิทธิภาพของเนตเวอร์กคือ

- โพรโตคอลที่ใช้ในการสนทนา (protocol use in conversation)
- กำหนดว่าในการแจ้งเตือนเหตุร้ายที่เกิดขึ้น
- การเตือนเมื่อมีเหตุเกิดขึ้นแก่เนตเวอร์ก

สิ่งสำคัญอย่างอื่นที่ช่วยในการวัดประสิทธิภาพคือ

- สร้างทราฟฟิกเนตเวอร์ก (generating network traffics) ช่วยให้สามารถจำลองเนตเวอร์กได้โดยการสร้างแพคเก็ตและส่งเข้าไปในเนตเวอร์ก
- ผลของการแสดงต้องช่วยให้เข้าใจง่าย และสามารถจับเก็บ (Capture) ข้อมูลที่กำหนดเพื่อมาวิเคราะห์ภายหลังได้

3.2.4.2 การวัดประสิทธิภาพ

3.2.4.2.1 การใช้งานแบนด์วิธ (bandwidth usage)

หาค่าเฉลี่ยการใช้งานช่วงเวลาคงที่ (ทุก ๆ 1 นาที หรือทุก ๆ 1 วินาที) ทำได้โดยนับจำนวนทราฟฟิกในช่วงเวลาที่กำหนด หารด้วยค่าทรูพุทสูงสุด ค่าที่คำนวณได้จะไม่ต่ำกว่า 5 เปอร์เซนต์และพิจารณาดังนี้

- 10-15 เปอร์เซ็นต์ แสดงว่าเนตเวอร์กมีการใช้งานน้อยถึงปานกลาง

- 25 เปอร์เซ็นต์ แสดงว่าเนตเวอร์กมีการใช้งานสูงสุด เนตเวอร์กแจมมิ่ง (network jamming) การตอบสนองช้า มีแพคเก็ตเสียมาก หรือ มีการส่งซ้ำปริมาณมาก

3.2.4.2.2 ข้อผิดพลาดในการส่ง (Transmission error)

แพคเก็ตที่เกิดการชนจะต้องทำการส่งใหม่อีกครั้งหนึ่งโดยส่วนใหญ่จะถูกรายงานว่า ซีอาร์ซี ผิดพลาด หรือแพคเก็ตสั้นผิดปกติ (runt) ฯลฯ ซึ่งสาเหตุมาจากการชนกัน สำหรับเนตเวอร์กที่มีการใช้งานสูง (ประมาณ 20 เปอร์เซ็นต์) อัตราการชนที่ยอมรับได้จะอยู่ระหว่าง 1-2 เปอร์เซ็นต์

ถ้าทั้งการใช้งานแบบแบนด์วิธและ อัตราแพคเก็ตเสีย (failed rate) สูงทั้งคู่ หมายความว่าประสิทธิภาพของเนตเวอร์กไม่ได้ดีดังที่ควรเป็น ซีอาร์ซีผิดพลาดอาจมีสาเหตุมาจากเนตเวอร์กอินเตอร์เฟส เช่น เราทราบว่าแต่ละโหนดมีการใช้งานต่ำ ในขณะที่เนตเวอร์กมีการใช้งานสูง สาเหตุส่วนใหญ่จะเกิดจากเนตเวอร์กอินเตอร์เฟส สาเหตุอย่างอื่น เช่น ตัวเชื่อมต่อ (connector) หลวม

3.2.4.2.3 การวัดที่จำเป็นสำหรับการออกแบบเพื่อแบ่งเนตเวอร์ก (Measurements needed for design to partition network)

ทำได้หลายวิธีดังนี้

- แบ่งตามชนิดของโพรโตคอลที่ใช้ออกจากกันใช้ ในกรณีที่พีซีแลนด์ (PC LANs) และไมโครคอมพิวเตอร์อยู่ในเนตเวอร์กเดียวกัน เป็นวิธีง่ายต่อการนำไปปฏิบัติ แต่ในกรณีที่มีการใช้งานที่หลากหลายอยู่ด้วยกัน ไม่เหมาะอย่างยิ่งที่จะแบ่งเป็นโพรโตคอล
- แบ่งตามฟิสิคัลแอดเดรส ใช้ในกรณีที่เรามี เวิร์คกรุป (workgroup) ในแต่ละตำแหน่งที่ต้องการติดต่อกับเนตเวอร์กเป็นบางครั้ง ส่วนใหญ่จะติดต่อกันเองภายในกลุ่มการที่เราจะทราบว่าผู้ใช้ใดจะอยู่ในกลุ่มเดียวกัน เราจะต้องเก็บข้อมูลของการสนทนาในแต่ละโหนด

3.2.4.2.4 ขนาดของแพคเก็ตและประสิทธิภาพ (Packet size and performance)

ทรูพุทของดาต้าที่ใช้งานจริง (usable data throughput) กับ เนตเวอร์กทรูพุทที่แตกต่างกัน แอพพลิเคชั่นจะเป็นตัวกำหนดขนาดของแพคเก็ตที่เหมาะสม (optimum packet size) เช่น ถ้าแอพพลิเคชันมีการรับส่งข้อมูลที่ขนาดใหญ่ ซึ่งจำเป็นต้องแบ่งข้อมูลออกเป็นหลายแพคเก็ต เรากำหนดให้ใช้ขนาดแพคเก็ตสูงสุดของ เนตเวอร์กโพรโตคอลที่ใช้จะดีที่สุด (1518 ไบต์ในกรณีอีเทอร์เนต) ซึ่งจะช่วยลดจำนวนของแพคเก็ตที่ใช้ในการส่งข้อมูลและช่วยลดผลของโพรโตคอลโอเวอร์เฮดด้วย

ปัจจัยที่เป็นข้อจำกัดของ การกำหนดขนาดของเฟรม คือ โครงสร้างของเนตเวอร์กซึ่งการใช้ บริดจ์ และ เราท์เตอร์ จะไม่ยอมให้แพคเก็ตใหญ่ผ่าน (บางที) หรือว่าแอพพลิเคชั่นอื่นที่ต้องการพารามิเตอร์ที่ต่างกันออกไป

ไม่มีกฎตายตัวสำหรับการกำหนดขนาดเฟรม ทางเดียวก็คือ เมื่อมีการเปลี่ยนแปลงเราต้องคอยดูผลที่เกิดขึ้น ซึ่งได้จากข้อมูลการมอนิเตอร์ (monitoring information) โดยไม่เพียงแต่ดูเฉพาะส่วนที่มีการเปลี่ยนแปลง เราต้องดูผลที่เกิดกับเนตเวอร์กโดยรวมด้วย

3.2.4.2.5 โหนดใช้งานและไม่ใช้งาน (Active and inactive nodes)

โหนดที่ไม่สนองตอบ คือ โหนดที่ได้รับการติดต่อแต่ไม่ตอบสนองต่อการร้องขอและไม่ได้ส่งการติดต่อ

3.2.4.2.6 กำหนดช่วงเวลาใช้งานสูงสุด (definition of peak usage time)

เราต้องบันทึกเวลาที่เกิด การใช้งานสูงสุดและการใช้งานน้อยสุด เพื่อนำไปทำแผนการทำงาน เช่น ออโตเมติคแบ็คอัพ (automatic backup) การอัพเกรดครั้งใหญ่ (large batch update) หรือการส่งข้อมูล

มอนิเตอร์ริงแพคเก็ต (monitoring packet) จะช่วยได้มาก และยังสามารถทราบทราฟฟิกเพื่อจำลองการทำงานได้

บทที่ 4

การออกแบบ และพัฒนาโปรแกรม

4.1 หลักการเบื้องต้นของการพัฒนา Software

4.1.1 ศึกษาระบบ (system Study)

ทำการศึกษาระบบของ Network การติดต่อกับ Packet driver ศึกษาสิ่งแวดล้อมที่มีความสัมพันธ์กับระบบ network ที่เรากำลังศึกษาอยู่ เช่น ในการดักจับ Packet นั้นจำเป็นจะต้องใช้อะไรบ้างในการดักจับและนำมาวิเคราะห์

4.1.2 ศึกษาความเป็นไปได้ (Feasibility Study)

ศึกษาความเป็นไปได้ของระบบว่าสามารถตรวจจับข้อมูลอะไรมาวิเคราะห์ได้บ้างและสามารถวิเคราะห์อะไร จากข้อมูลนั้น แล้วนำไปใช้ประโยชน์อะไรได้บ้าง

4.1.3 กำหนดรายละเอียดความต้องการ (Software Specification)

เป็นรายละเอียด และข้อกำหนดของแต่ละความต้องการ ที่ผู้ใช้สามารถคาดหวังจากโปรแกรมที่พัฒนานี้

4.1.4 ออกแบบซอฟต์แวร์ (Software Design)

ออกแบบซอฟต์แวร์โดยเลือกภาษาที่ใช้ในการพัฒนา ในการออกแบบจะแยกส่วนประกอบต่าง ๆ ออกเป็นโมดูลเพื่อ สะดวกในการสร้างและสามารถแก้ไขข้อผิดพลาดได้ง่ายหากโปรแกรมเกิดข้อผิดพลาด โดยจะแบ่งออกเป็น

- การสร้างหน้าจอเพื่อติดต่อกับผู้ใช้
- การกรองข้อมูล
- การติดต่อรับข้อมูล
- การเก็บข้อมูลลงใน disk
- การอ่านข้อมูลจาก disk
- การวิเคราะห์และแสดงผลข้อมูล

4.1.5 จัดทำและทดสอบการใช้งาน (Implementation and Unit Testing)

จัดแยกแล้วทดสอบโมดูลย่อยที่สร้างขึ้นว่า ทำงานถูกต้องตามต้องการหรือไม่

4.1.6 รวบรวมและทดสอบระบบ (Integration and Testing)

เมื่อทำการทดสอบโมดูล ย่อยๆ เสร็จแล้ว ก็นำโมดูลต่าง ๆ มารวมกันเป็นโปรแกรมที่ต้องการแล้วทำการทดสอบการทำงานโดยรวมอีกครั้งหนึ่ง

4.2 การวางแผนและพัฒนาระบบ

ขั้นตอนในการวางแผนและพัฒนาระบบมีดังต่อไปนี้

- ศึกษาวิธีการทำงาน โดยพิจารณาความเป็นไปได้ในขอบเขตของโครงการ โดยต้องคำนึงถึงส่วนประกอบต่าง ๆ ที่มีความสัมพันธ์กับโปรแกรมด้วย เช่น ความเร็วของโปรแกรม
- แยกส่วนต่าง ๆ ออกเป็นส่วนย่อยๆ และเริ่มพัฒนาโปรแกรมเป็นส่วนๆ แล้วนำมาประกอบกันในภายหลัง โดยออกแบบโครงสร้างข้อมูลของแต่ละโมดูล และเขียนแผนภาพลำดับการทำงานของโมดูลย่อย
- เลือกพัฒนาโปรแกรมบนภาษาปาสคาล

4.3 ส่วนประกอบของแต่ละโมดูล

4.3.1 โมดูลในการตรวจจับ

โมดูลนี้ออกแบ่งเป็นส่วนย่อย ๆ คือ

- ส่วนในการติดต่อกับ Packet Driver
- ส่วนแสดงข้อมูลของ Packet Driver ที่ตรวจจับได้และแสดงผลข้อความ เพื่อแจ้ง หากติดต่อกับ Packet Driver ไม่ได้
- ส่วนดักจับข้อมูล
- ส่วนกรองข้อมูล ตามข้อกำหนดที่ตั้งไว้

4.3.2 โมดูลการตั้งค่าก่อนการตรวจจับ

- ส่วนตั้งค่าเวลา สำหรับใช้ดักจับข้อมูลทุกๆหน่วยเวลา
- ส่วนตั้งค่าข้อกำหนด ในการกรอง Packet
    - กรองชนิดของอีเทอร์เนต
    - กรองหมายเลขที่อยู่ (Mac Address)
    - กรองชนิดของโพโตคอล
    - กรองหมายเลขที่อยู่ (IP Address)
- ส่วนแสดงผลการตั้งค่าก่อนการตรวจจับทั้งหมด

4.3.3 โมดูลในการเก็บข้อมูล

ในการเก็บข้อมูลนั้น จะมีส่วนที่ต้องการเก็บดังนี้ คือ

- จำนวน packet ทั้งหมดที่จับได้
- ขนาดของแต่ละ packet ที่ตรวจจับได้
- ข้อมูลดิบทั้งหมด

ในการบันทึกข้อมูลจะเป็นแบบ "Text file" และแสดงเป็นเลขฐาน 16 เพื่อให้สามารถเข้าใจได้ง่าย ส่วนในการใส่ชื่อหรือตั้งชื่อไฟล์จะกำหนด นามสกุลเป็น nmt (network monitoring tool)

4.3.4 โมดูลแสดงผลการดักจับข้อมูล

มีการแสดงผลของข้อมูลเป็นดังนี้

- เวลา
- ปริมาณแพคเก็ตข้อมูลต่อหน่วยเวลา
- ขนาดข้อมูล (กิโลไบต์) ต่อหน่วยเวลา
- ชนิดของอีเทอร์เน็ต
- จำนวนและขนาด (กิโลไบต์) ของโพรโตคอลในชั้นบน

4.3.5 โมดูลการอ่านข้อมูลที่บันทึกไว้เพื่อนำมาวิเคราะห์ในภายหลัง

- ส่วนการอ่านข้อมูลแบบ text file
- ส่วนของการเลือกไฟล์ข้อมูลในดิสก์

4.3.6 โมดูลในการวิเคราะห์และแสดงผล

โดยจะแยกเป็นโมดูลย่อยดังนี้

- แยกชั้นโพรโตคอล โดยรวมและกำหนด byte เริ่มต้นของแต่ละโพรโตคอลออกมา
- แสดงผลในส่วนโพรโตคอลของชั้น Datalink แบ่งเป็น
  - ไออีอีอี 802.2 (IEEE 802.2)
  - ไออีอีอี 802.3 (IEEE 802.3)
  - อีเทอร์เนต ทู (Ethernet II)
  - อีเทอร์เนตสแนบ (Ethernet Snap)

- แสดงผลในส่วนโพรโตคอลของชั้น Network
  - ไอพี (IP)
  - ไอพีเอ็กซ์ (IPX)
  - เออาร์พี (ARP)

- แสดงผลในส่วนโพรโตคอลของชั้น Transport
  - ทีซีพี (TCP)
  - ยูดีพี (UDP)
  - ไอซีเอ็มพี (ICMP)
  - เอสพีเอ็กซ์ (SPX)
  - เอ็นซีพี (NCP)
- แสดงผลในส่วนโพรโตคอลของชั้น Upper

4.3.7 โมดูลในส่วนของ Menu

- ส่วนแสดงผล Menu
- ส่วนรับค่าคีย์
- ส่วนประมวลผลตามคำสั่ง

4.4 ส่วนอัลกอริทึมของโปรแกรม

ส่วนการออกแบบวิธีการทำงานในส่วนต่างๆ ซึ่งได้แบ่งเป็น โมดูลหลักแยกจากกันเพื่อง่ายในการพัฒนาโปรแกรมมีส่วนต่างๆดังนี้

- การตั้งค่าเริ่มต้นของแพคเก็ตไดรเวอร์
- การจับข้อมูล
- การกรองแพคเก็ต
- การวิเคราะห์และแสดงผล

4.4.1 การตั้งค่าเริ่มต้นของแพคเก็ตไดรเวอร์

ในการดักจับแพคเก็ตข้อมูลนั้นเราจะต้องอาศัยการติดต่อกับแพคเก็ตไดรเวอร์ โดยไม่ต้องไปติดต่อกับการ์ดอินเตอร์เฟสโดยตรง ซึ่งวิธีนี้เราสามารถที่จะนำไปใช้ในเน็ตเวิร์คอื่นๆ ซึ่งอยู่ในคลาสเดียวกันได้ โดยไม่ต้องแก้ไขโปรแกรม เพียงแต่หาค่าหมายเลขอินเตอร์รัพท์แพ็คเก็ตไดรเวอร์อันใหม่เท่านั้น ซึ่งหมายเลขอินเตอร์รัพท์นี้เราสามารถนำมาใช้เรียกฟังก์ชั่นต่างๆของแพคเก็ตไดรเวอร์มาใช้งานได้ โดยเราสามารถแบ่งระดับชั้นของฟังก์ชั่นได้ดังนี้

- ฟังก์ชั่นพื้นฐาน (Basic Function) มีฟังก์ชั้นการทำงานพื้นฐานซึ่งง่ายในการใช้งานเนื่องจากทรัพยากรน้อย
- ฟังก์ชั่นเพิ่มเติม (Extended Function) มีฟังชั้นมากกว่าแบบฟังก์ชั้นพื้นฐาน สนับสนุนมัลติคาสต์ และเก็บสถิติ
- ฟังก์ชั่นประสิทธิภาพสูง (High Performance Function) สนับสนุนการปรับปรุงประสิทธิภาพ

ในการหาค่าหมายเลขอินเตอร์รัพท์ เราจะพิจารณาหมายเลขอินเตอร์รัพท์ในช่วง 0x60 ถึง 0x80 โดยเราจะใช้คำสั่งในการหาค่าตำแหน่งเริ่มต้นของแต่ละหมายเลขอินเตอร์รัพท์ทีละหมายเลข เพื่อนำค่าในตำแหน่งตั้งแต่ไบต์ที่ 4 จำนวน 12 ไบต์ มาพิจารณาว่ามีข้อความ "PKT DRVR" หรือไม่ ถ้าข้อความตรง แสดงว่าหมายเลขอินเตอร์รัพท์หมายเลขนี้เป็นหมายเลขอินเตอร์รัพท์ที่แพคเก็ตไดรเวอร์ใช้อยู่ ถ้าไม่ใช้ให้พิจารณาหมายเลขอินเตอร์รัพท์ตัวอื่นๆต่อไป

เมื่อเราได้หมายเลขอินเตอร์รัพท์แล้ว เราก็สามารถเรียกใช้ฟังก์ชั่นของแพคเก็ตไดรเวอร์ได้ โดยฟังชั่นที่เราใช้ในโครงงานนี้ได้แก่

- ฟังก์ชั่นในการหาข้อมูลเกี่ยวกับแพคเก็ตไดรเวอร์
- ฟังก์ชั่นในการกำหนดค่าเริ่มต้นในการทำงานของแพคเก็ตไดรเวอร์
- ฟังก์ชั่นหาค่าหมายเลขที่อยู่ของการ์ดอินเตอร์เฟส
- ฟังก์ชั่นในการกำหนดโหมดการรับข้อมูล ซึ่งแต่ละโหมดมีค่าต่างๆดังนี้คือ
    1. ไม่รับแพ็คเก็ต
    2. รับเฉพาะแพ็คเก็ตที่ส่งมาอินเทอร์เฟสนี้
    3. โหมด 2 รวมกับบรอดคาสท์แพ็คเก็ต
    4. โหมด 3 รวมกับลิมิตเตดมัลติคาสท์แพ็ตเก็ต
    5. โหมด 3 รวมกับมัลติคาสท์แพ็คเก็ต
    6. ทุกแพ็คเก็ต

4.4.2 การจับข้อมูล

ส่วนเริ่มต้นของการเก็บแพคเก็ต โดยจะต้องคำนึงถึงเรื่องการเก็บแพคเก็ต ซึ่งมักจะมีปัญหาตรงที่ไม่สามารถรับแพคเก็ตได้ทัน ทำให้ต้องสูญเสียแพคเก็ตไป ดังนั้นควรจะมีบัฟเฟอร์เพื่อสำรองข้อมูลและทำการจัดเรียงข้อมูล เพื่อให้สามารถรับแพ็คเก็ตได้ใกล้เคียงกับความเป็นจริงมากที่สุด

การใช้งานจะเรียกผ่านแพ็คเก็ตไดรเวอร์ ซึ่งต้องติดต่อผ่านทางบริการอินเตอร์รัพท์ (User Interface Interrupt Service) ที่มีให้ ซึ่งจะต้อผ่านข้อมูลมายังโมดูลที่เรากำหนด แล้วค่อยรวบรวมข้อมูลหรือจัดทำสถิติต่อไป

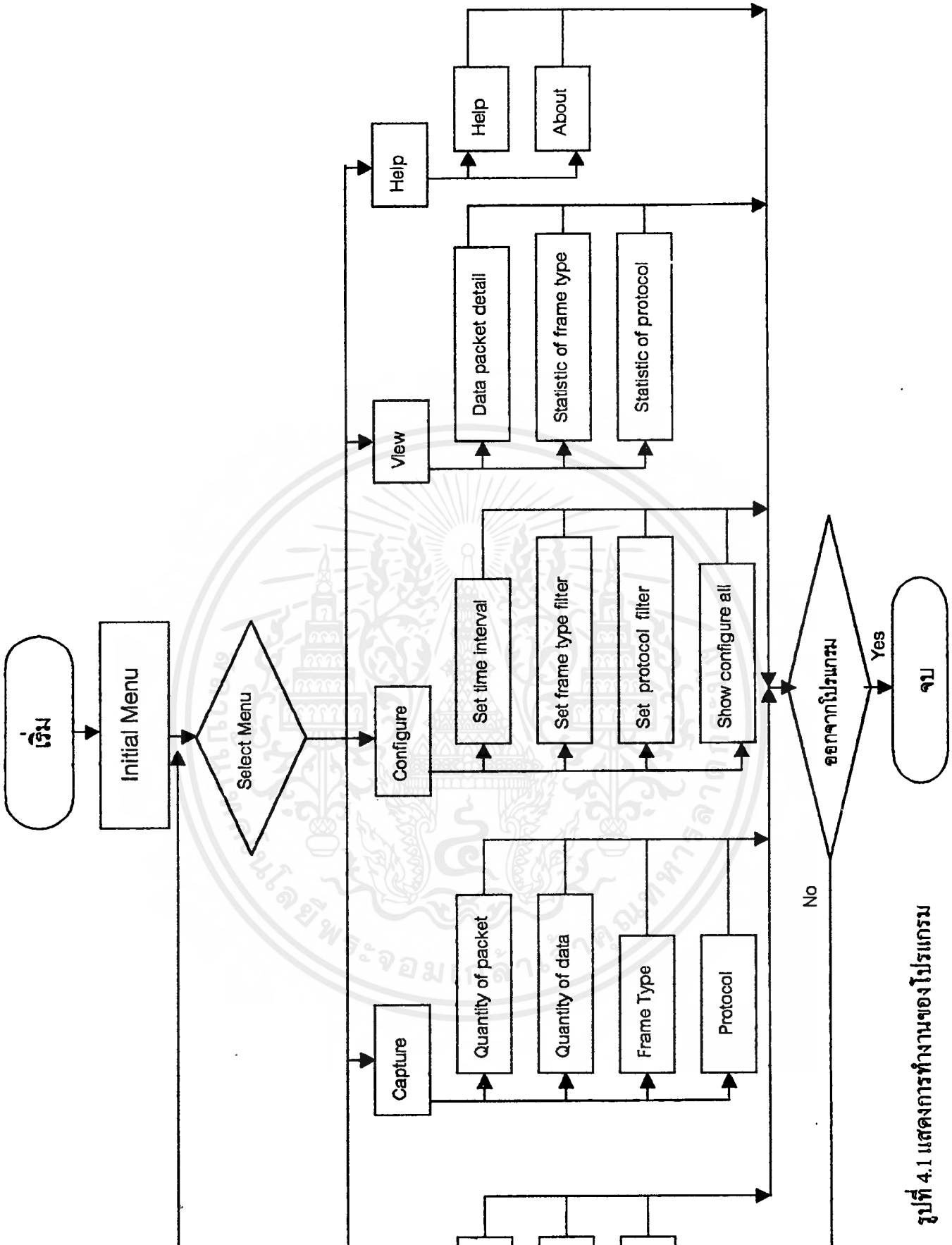
4.4.3 การกรองแพคเก็ต

เมื่อมีข้อมูลผ่านเข้ามาแล้วก่อนที่เราจะนำข้อมูลนั้นเก็บลงในบัฟเฟอร์ เราจะทำการกรองข้อมูลนั้นเสียก่อนโดย โดยจะทำการกรองตามค่าที่เราตั้งไว้ และเราจะเก็บข้อมูลเฉพาะที่ผ่านการกรองเท่านั้น ถ้าไม่ได้ตั้งค่าการกรองจะถือว่าเก็บทุกแพ็คเก็ต ซึ่งทำให้ขนาดข้อมูลอาจจะมีขนาดใหญ่มาก ซึ่งในที่นี้อาจจะมีข้อมูลที่เราไม่ได้สนใจรวมอยู่ด้วย ดังนั้นโปรแกรมจะทำการกรองเพื่อให้สามารถดูได้เฉพาะแพ็คเก็ตที่ต้องการ โดยใช้การตรวจสอบแพ็คเก็ตที่เข้ามาโดยนำค่าฟิลด์ต่างมาวิเคราะห์เพื่อกรองแพคเก็ตก่อนเก็บลงบัฟเฟอร์

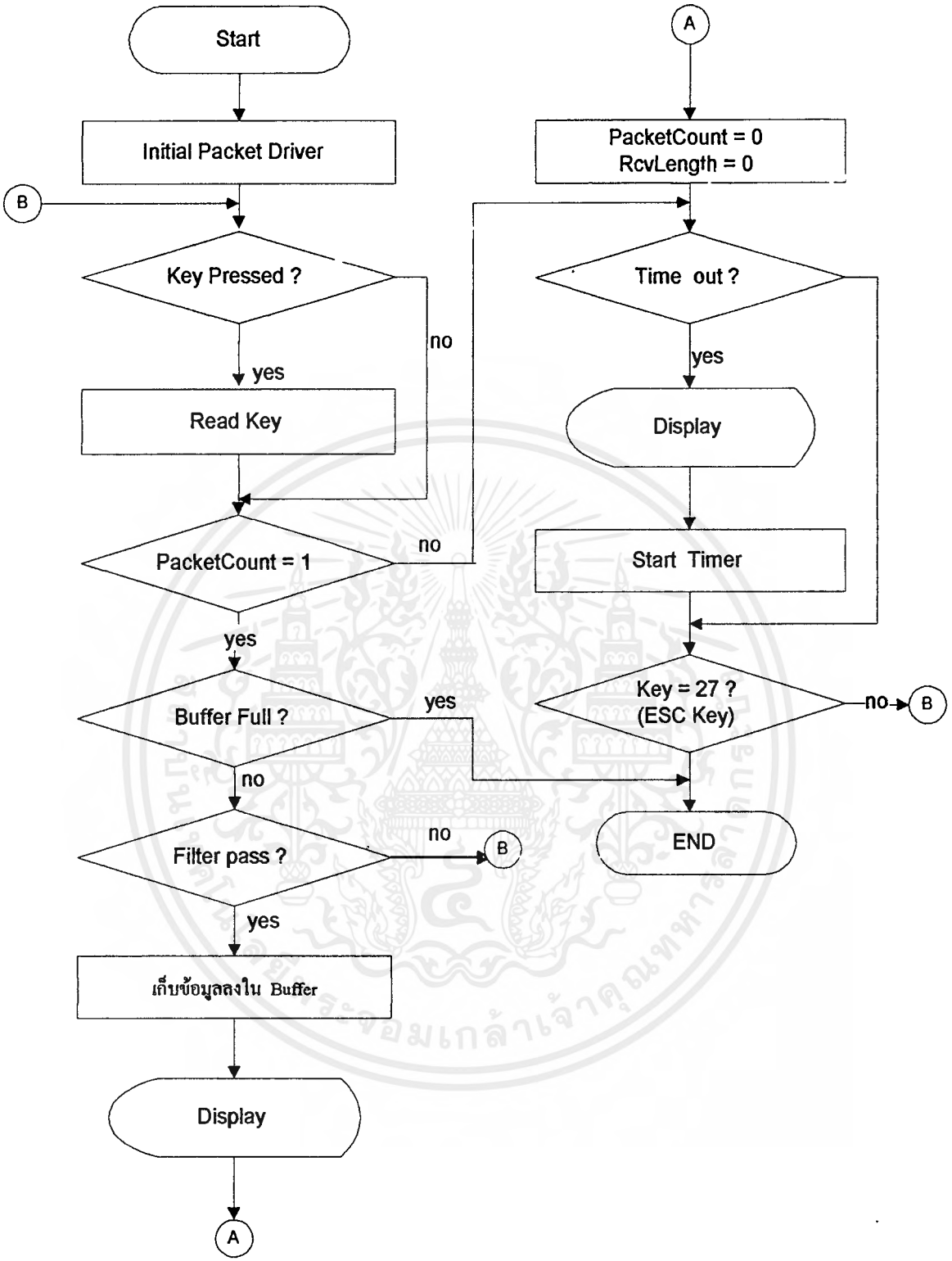
4.4.4 การวิเคราะห์และแสดงผล

ในการรายงานผลนั้นสามารถรวบรวมข้อมูลสำคัญต่างๆ เกี่ยวกับเครือข่าย เพื่อมาแสดงผลให้ผู้ใช้สามารถดูได้ โดยในที่นี้เราจะทำการวิเคราะห์แพ็คเก็ต และแสดงค่าฟิลด์ของโพโตรคอลต่างๆของแพ็คเก็ตข้อมูลที่ส่งมา และทำการแสดงสถิติปริมาณการใช้งานของแพ็คเก็ตตามชนิดของเฟรมและโพโตรคอล

4.5 รูปแสดงแผนผังลำดับการทำงานของโปรแกรม

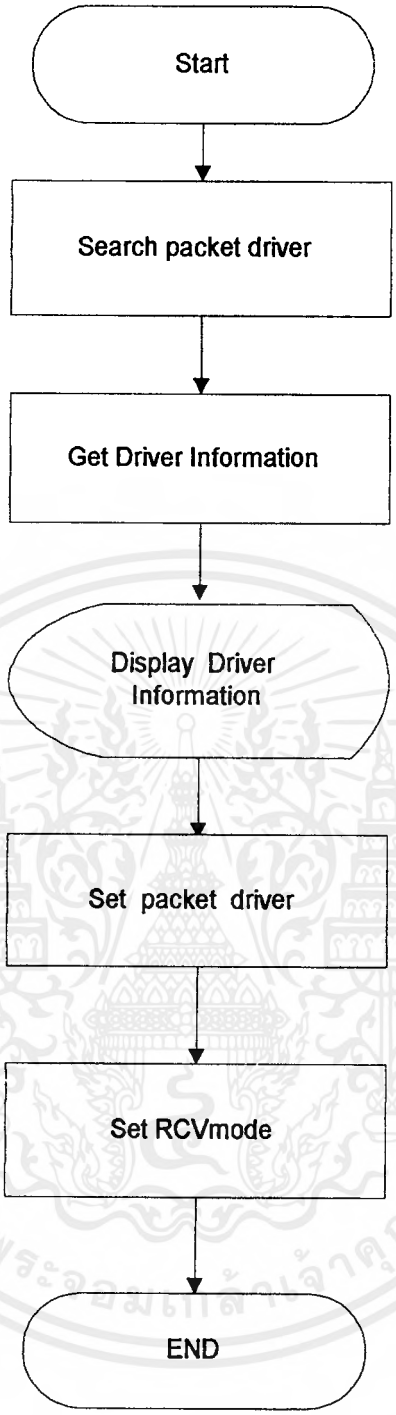


รูปที่ 4.1 แสดงการทำงานของโปรแกรม



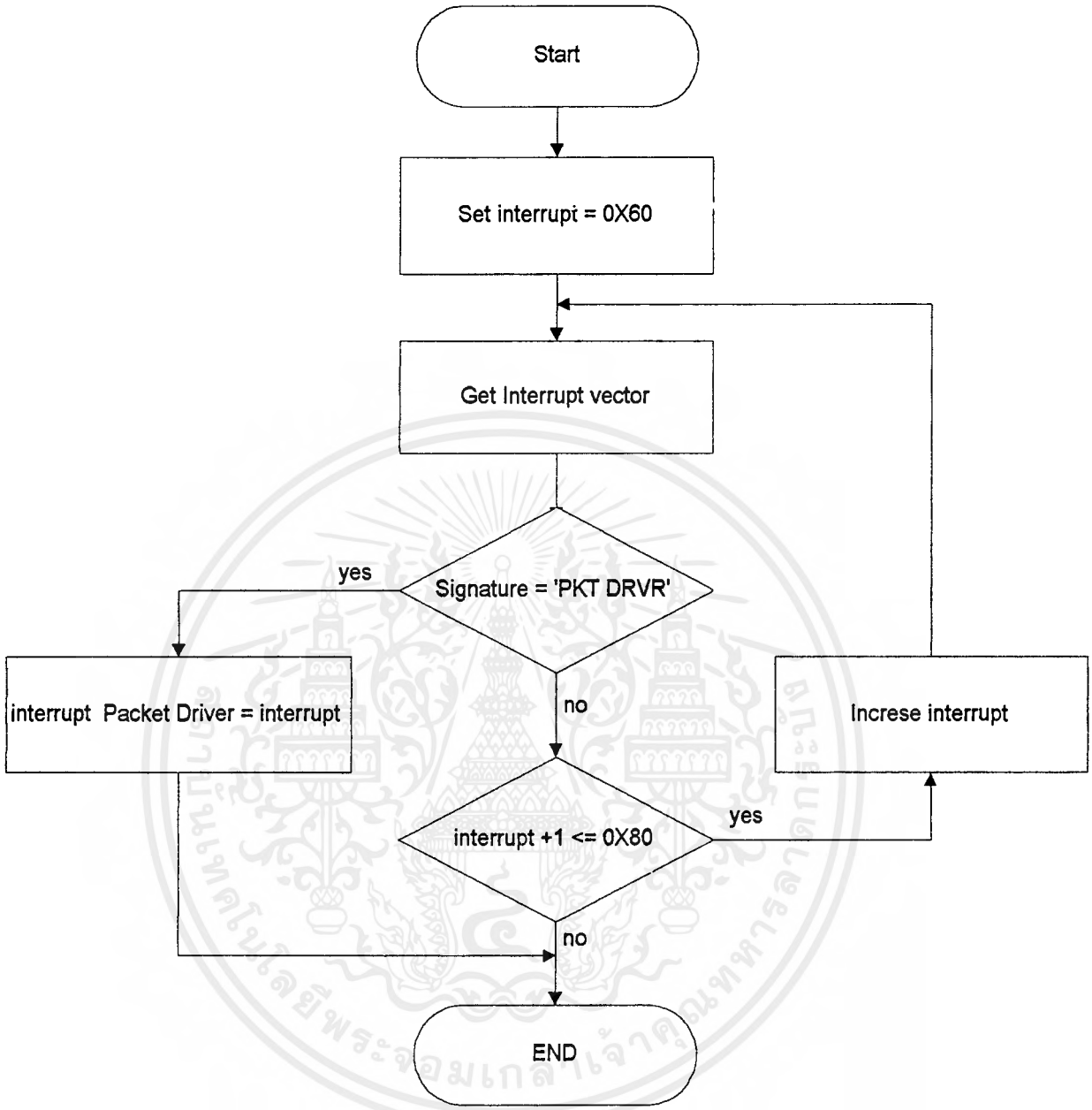
ขั้นตอนการทำงานของโมดูล ดักจับข้อมูล

รูปที่ 4.2



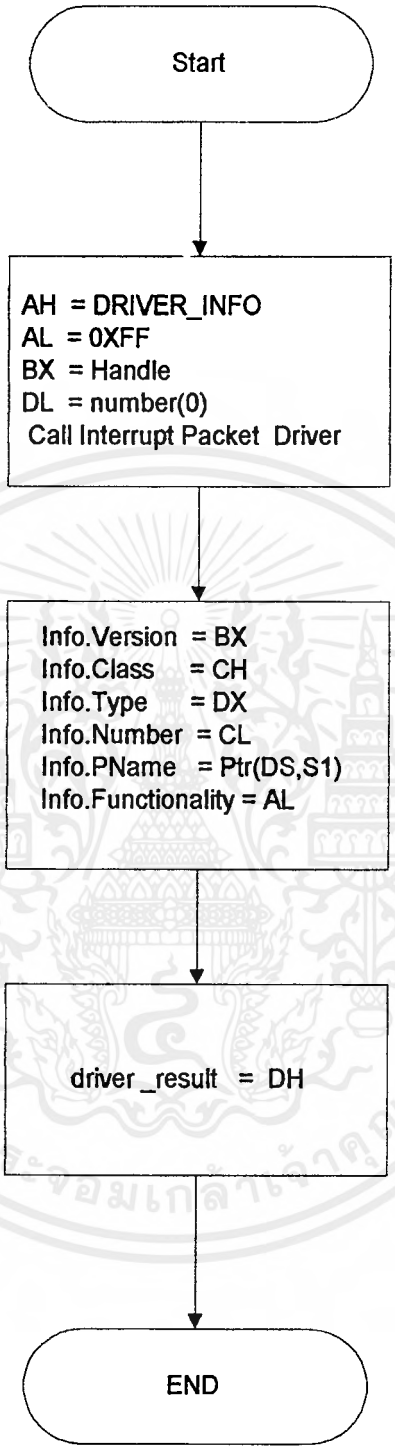
การกำหนดค่าเริ่มต้นให้กับ packet driver

รูปที่ 4.3



การหาค่า interrupt ของ packet Driver

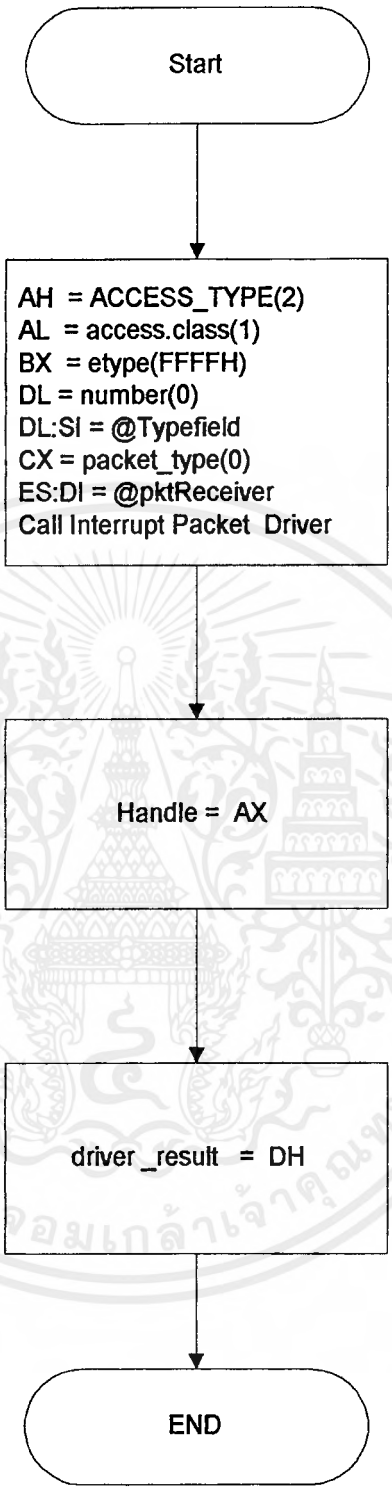
รูปที่ 4.4



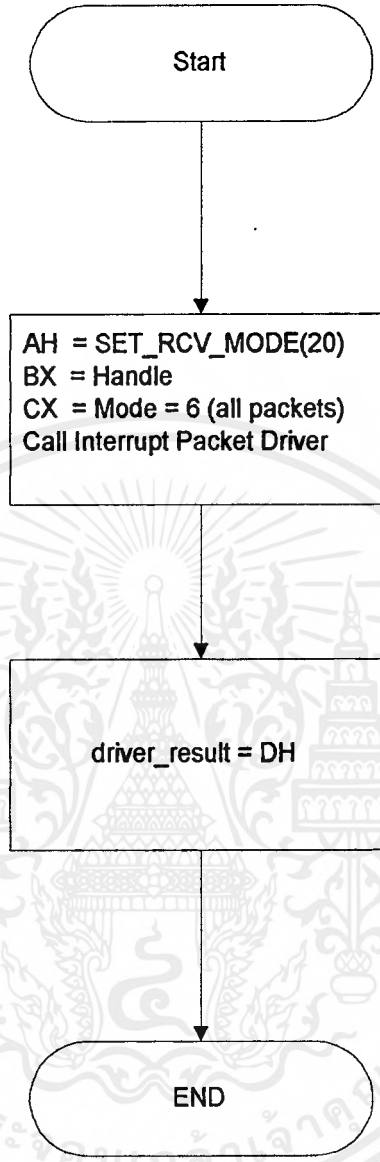
รับค่าข้อมูลของแพ็คเก็ตไดร์เวอร์

(Packet Driver Information)

รูปที่ 4.5

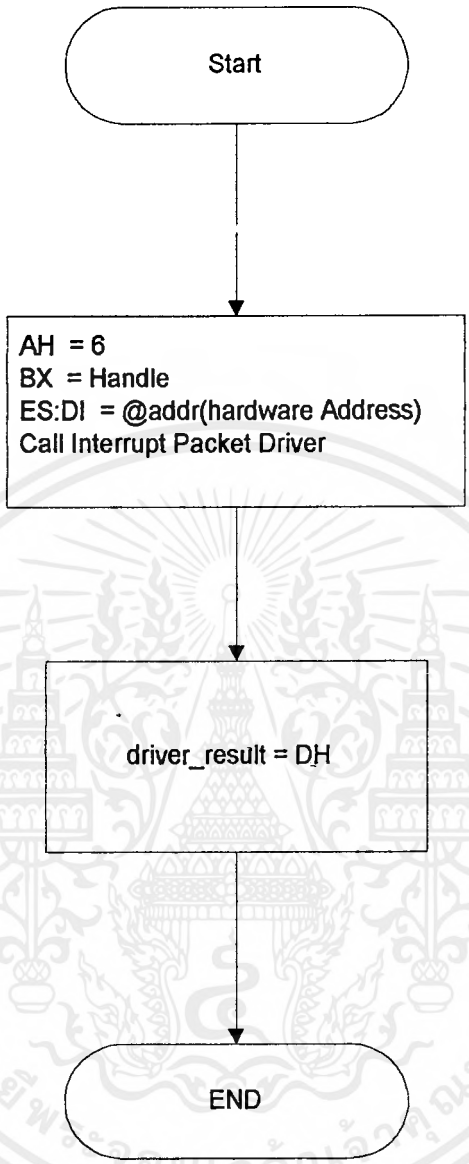


ขั้นตอนการตั้งค่าตัวแปรของ
แพคเก็ตไดร์เวอร์
รูปที่ 4.6



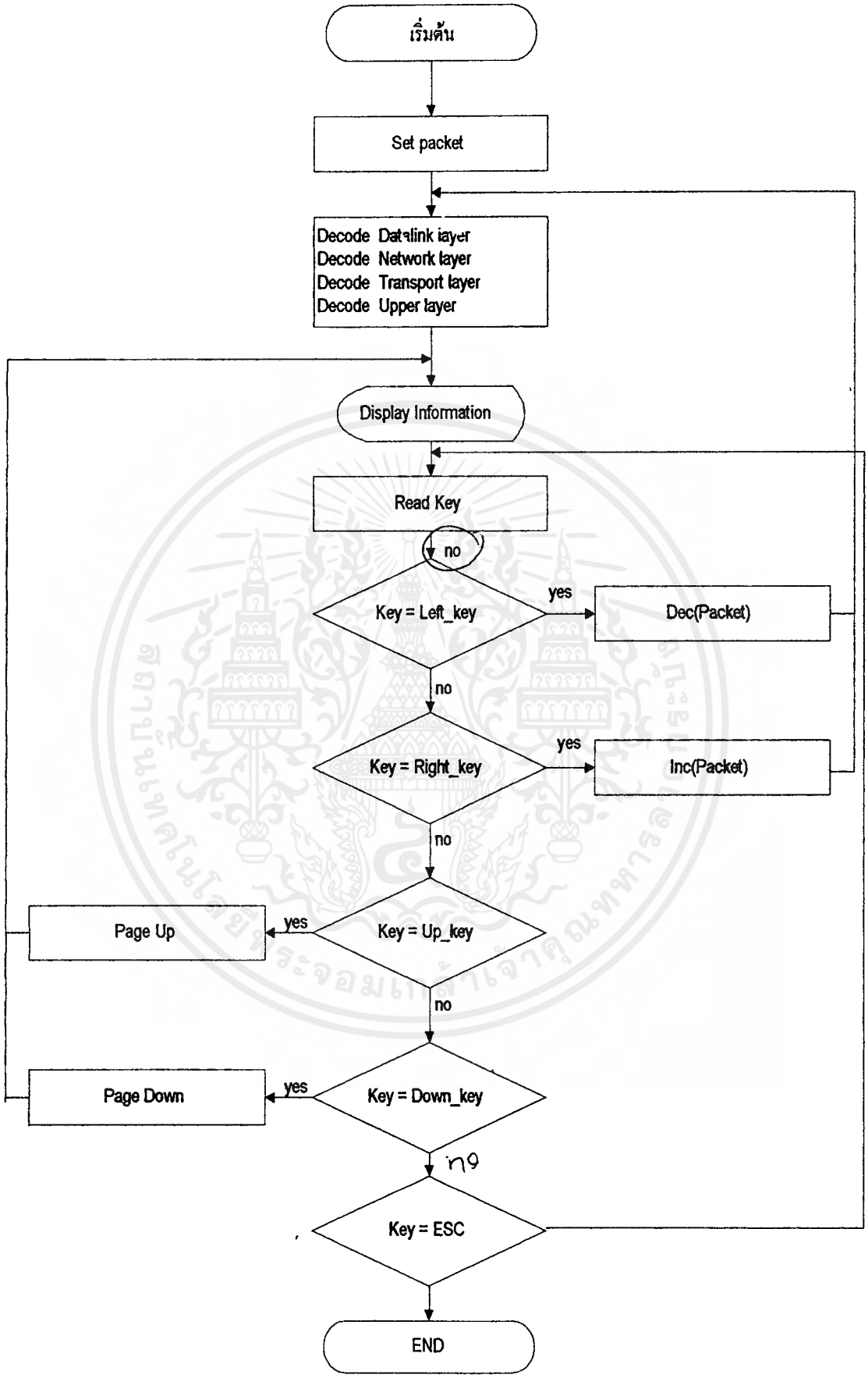
ขั้นตอนการตั้งค่าวิธีการรับแพคเก็ต

รูปที่ 4.7



รับค่าหมายเลขที่อยู่ของ Hardware

รูปที่ 4.8



โมดูลการวิเคราะห์โพรโตคอล

รูปที่ 4.9

บทที่ 5

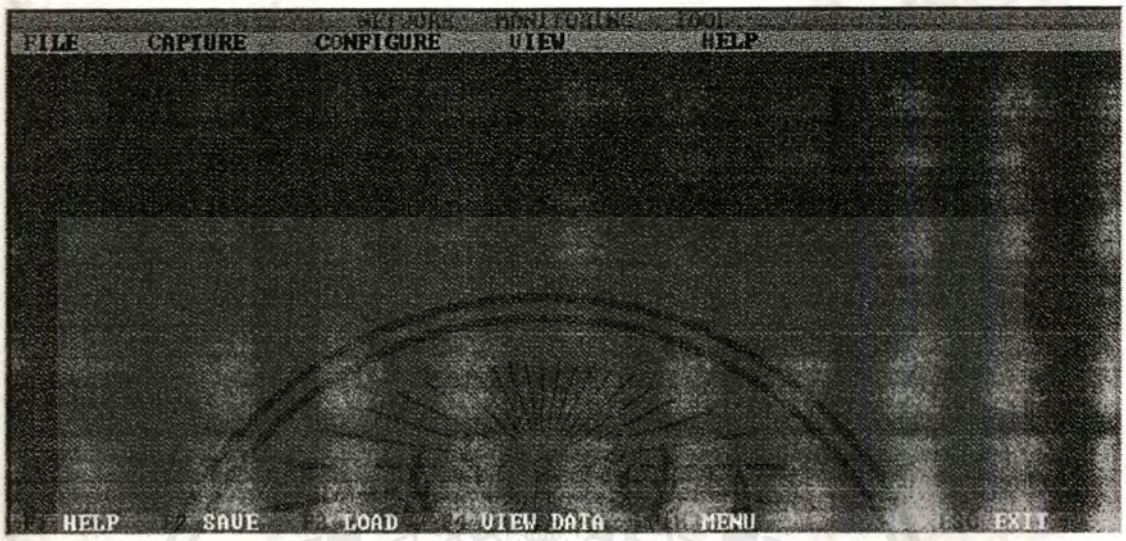
การใช้งาน และการทดสอบโปรแกรม

5.1 การใช้งานโปรแกรม

โปรแกรมนี้จะเป็นการตรวจจับแพคเก็ตที่มีการรับส่งในขณะเวลานั้นๆ โดยมีการนำเมนูมาใช้ในการเรียกใช้คำสั่งต่าง เพื่อเพิ่มความสะดวกให้แก่ผู้ใช้งาน โดยมีเมนูต่างๆ ดังนี้คือ

- **FILE**
  - LOAD
  - SAVE
  - EXIT
- **CAPTURE**
  - QUANTITY OF PACKET
  - QUANTITY OF DATA
  - FRAM TYPE
  - PROTOCOL (UPPER LAYER)
- **CONFIGURE**
  - SET TIME INTERVAL
  - SET FRAME TYPE FILTER
    - FRAME TYPE
    - MAC ADDRESS
  - SET PROTOCOL FILTER
    - PROTOCOL TYPE
    - IP ADDRESS
  - SHOW CONFIGURE ALL
- **VIEW**
  - PACKET DATA DETAIL
  - STATISTIC OF FRAME TYPE
  - STATISTIC OF PROTOCOL

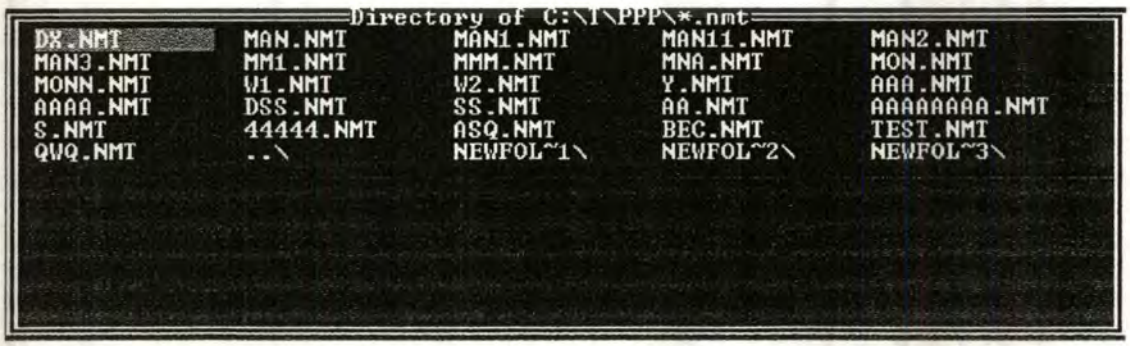
- **HELP**
  - HELP
  - ABOUT



รูปที่ 5.1 แสดงหน้าจอปกติของโปรแกรม

5.1.1 FILE เมนูนี้จะใช้ประกอบไปด้วย 3 เมนูย่อยคือ

- **LOAD** เป็นการนำข้อมูลในดิสก์มาใส่ไว้ในตัวแปรบัฟเฟอร์ของโปรแกรม เพื่อที่จะสามารถนำมาวิเคราะห์ในภายหลัง เมื่อเราเลือกเมนูนี้แล้วจะปรากฏหน้าต่างออกมาและแสดงชื่อไฟล์ที่มีนามสกุลเป็น NMT ให้เราเลือกดังแสดงในรูป โดยสามารถที่จะเปลี่ยนไดร์ฟได้โดยกดคีย์ตัวอักษรของไดร์ฟนั้นเช่นกดอักษร 'A' โปรแกรมจะเปลี่ยนเป็นแสดงไฟล์ข้อมูลของไดร์ฟ 'A' ทันที

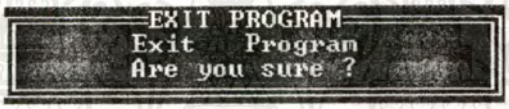


รูปที่ 5.2 แสดงหน้าต่างการเลือกไฟล์ข้อมูลของเมนู LOAD

- **SAVE** เป็นการบันทึกข้อมูลในตัวแปรบัฟเฟอร์ลงในดิสก์โดยจะมีหน้าต่างให้ใส่ชื่อของไฟล์ที่จะบันทึก หากเราไม่ใส่นามสกุล โปรแกรมจะใส่นามสกุล NMT โดยอัตโนมัติ และถ้าหากเราใส่นามสกุลอื่นโปรแกรมจะเปลี่ยนนามสกุลนั้นให้เป็นนามสกุล NMT เช่นกัน ดังแสดงใน

รูปที่ 5.3 แสดงหน้าต่างการใส่ชื่อไฟล์ข้อมูลของเมนู SAVE

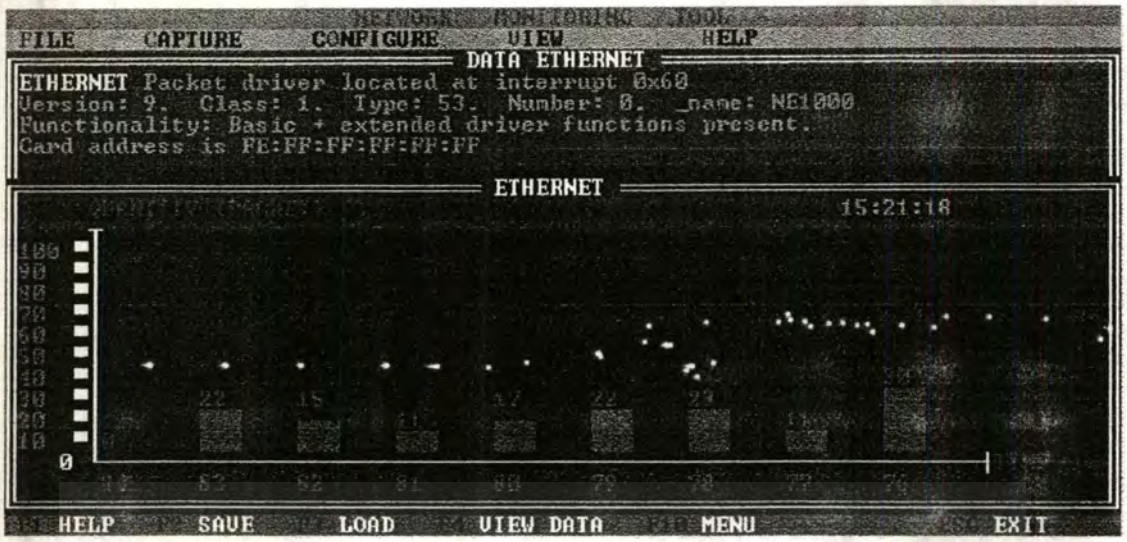
- **EXIT** เป็นเมนูออกจากโปรแกรม เราสามารถ กด **ESC** เพื่อออกจากโปรแกรมก็ได้ โดยจะมีหน้าต่างแสดงข้อความถามความแน่ใจที่จะออกจากโปรแกรม ถ้าเราจะออกจากโปรแกรมก็ให้กด 'Y' หรือ 'y' หากไม่ต้องการออกจากโปรแกรมให้กดคีย์ ESC , 'N' หรือ 'n' ก็จะกลับเข้าสู่โปรแกรมและสามารถเลือกเมนูต่างๆ ต่อไปได้



รูปที่ 5.4 แสดงหน้าต่างยืนยันการออกจากโปรแกรม

5.1.2 CAPTURE เป็นเมนูที่จะให้เลือกรูปแบบการแสดงผลการดักจับแพคเก็ตในรูปแบบต่างๆ โดยมีเมนูย่อยดังนี้คือ

- **QUANTITY OF PACKET** เป็นการแสดงจำนวนแพคเก็ตในแต่ละช่วงเวลาที่ตั้งเอาไว้ โดยจะแสดงเป็นกราฟแท่ง ที่มีค่าสูงสุดของกราฟคือ 100 และ 1000 แพคเก็ต โดยจะมีการเปลี่ยนสเกลโดยอัตโนมัติหากกราฟที่แสดงบนหน้าจอมีค่าเกิน 100 แพคเก็ต ก็จะเปลี่ยนเป็นสเกล ที่มีค่าสูงสุดเป็น 1000 แพคเก็ต และหากทุกกราฟมีค่าต่ำกว่า 100 แพคเก็ต ก็จะเปลี่ยนสเกลกลับไปที่สเกลเดิมทันที โดยกราฟจะเลื่อนไปทางซ้ายหากมีกราฟเกินหนึ่งหน้าจอแสดงผล

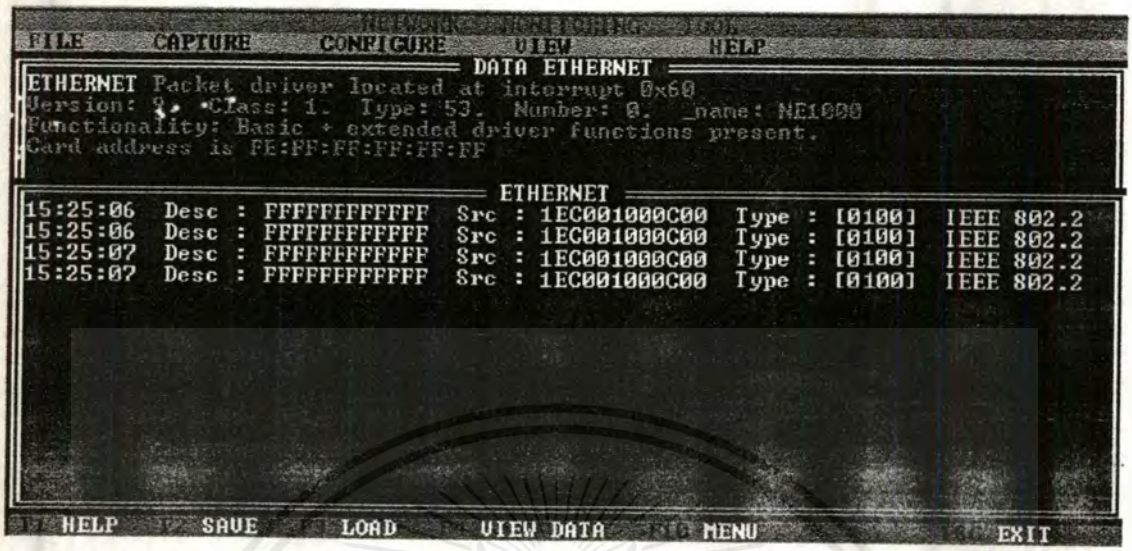


รูปที่ 5.5 แสดงการแสดงผลของเมนู QUANTITY OF PACKET

- **QUANTITY OF DATA** เป็นการแสดงขนาดข้อมูลมีหน่วยเป็น กิโลไบต์ ในแต่ละช่วงเวลา โดยมีสเกลที่ มีค่าสูงสุด 10 และ 40 กิโลไบต์ โดยสามารถปรับเปลี่ยนสเกลตามค่าของข้อมูลได้

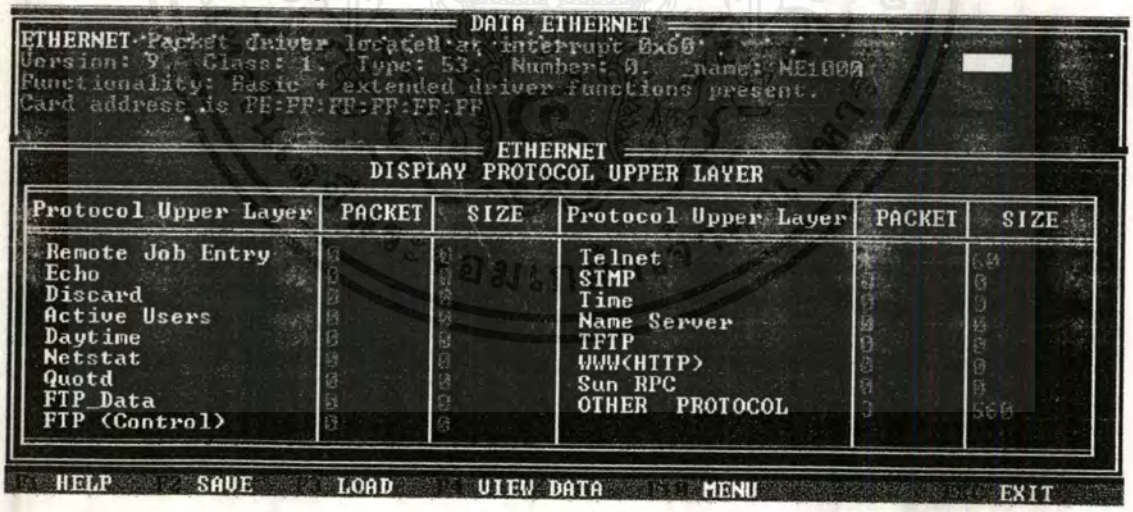
รูปที่ 5.6 แสดงการแสดงผลของเมนู QUANTITY OF DATA

- **FRAME TYPE** เป็นการแสดงหมายเลขต้นทางและปลายทางของแพคเก็ต และวิเคราะห์หาชนิดของอีเทอร์เน็ตมาแสดงผลพร้อมทั้งเวลาที่ดักจับแพคเก็ตนั้นได้



รูปที่ 5.7 แสดงการแสดงผลของเมนู ETHERNET TYPE

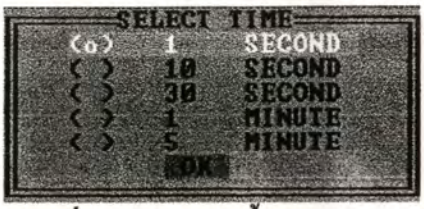
- **PROTOCOL (UPPER LAYER)** เป็นการแสดงจำนวนแพคเก็ตและขนาดรวมของแพคเก็ตมีหน่วยเป็น กิโลไบต์ ของโพรโตคอลต่างในชั้นอัพเปอร์เลเยอร์ โดยจะแสดงเป็นตัวเลขดังรูป



รูปที่ 5.8 แสดงการแสดงผลของเมนู PROTOCOL (UPPER LAYER)

5.1.3 CONFIGURE เป็นการตั้งค่าต่างๆในการดักจับแพคเก็ตข้อมูลโดยมีเมนูย่อยดังนี้

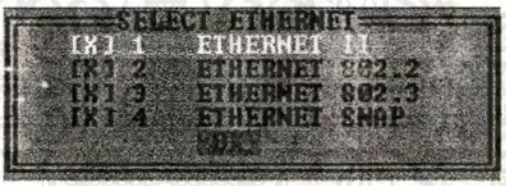
- **SET TIME INTERVAL** เป็นการตั้งค่าเวลาของการดักจับแพคเก็ตเพื่อใช้สำหรับการแสดงผลที่เป็นการรวมค่าในแต่ละช่วงเวลามาแสดงผลในรูปของกราฟ



รูปที่ 5.9 แสดงการตั้งค่าของเวลา

- **SET FRAME TYPE FILTER** เป็นการเซตค่าการกรองในส่วนของชนิดของเฟรม โดยแบ่งได้ดังนี้

  - FRAME TYPE เป็นการกรองชนิดของเฟรมตามชนิดของอีเทอร์เนตที่ใช้ โดยใช้คีย์ ENTER ในการเลือกเมนู เมื่อเรากด ENTER อีกครั้งก็จะเป็นการยกเลิก ดังรูป

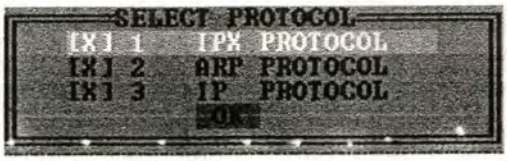


รูปที่ 5.10 แสดงการเลือกชนิดของเฟรมที่จะรับ

  - MAC ADDRESS เป็นการกรองที่อยู่ (MAC ADDRESS) โดยใส่ค่าที่อยู่ที่จะกรองทั้งต้นทางหรือปลายทาง ก็ได้ โดยเมื่อเลือกเมนูแล้วจะกด TAB เพื่อเข้าไปเปลี่ยนหมายเลขที่อยู่ดังรูป

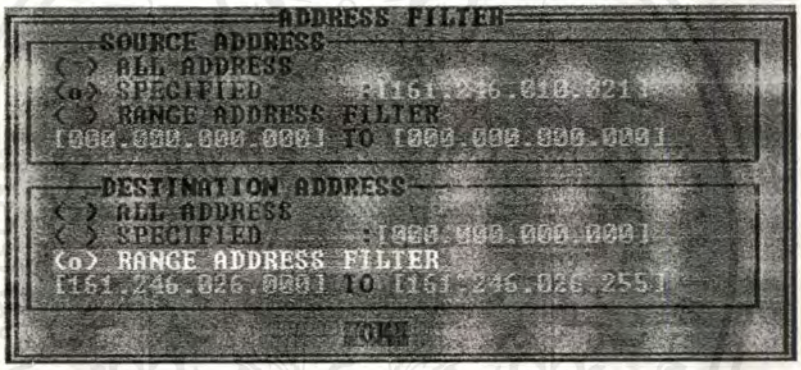
รูปที่ 5.11 แสดงการตั้งค่า MAC ADDRESS ที่จะรับ

- **SET PROTOCOL FILTER** เป็นการเซตค่าการกรองในส่วนของชนิดของโพรโตคอลโดยแบ่งได้ดังนี้
  - PROTOCOL TYPE เป็นการกรองชนิดของโพรโตคอลดังแสดงดังรูป



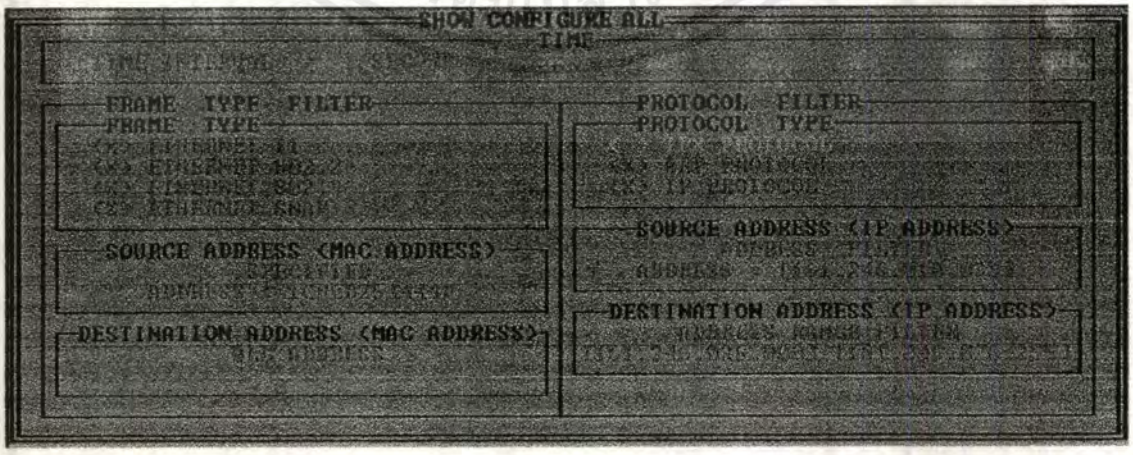
รูปที่ 5.12 แสดงการเลือกชนิดของโพรโตคอลที่จะรับ

  - IP ADDRESS เป็นการกรอง IP ADDRESS โดยสามารถกำหนดให้กรองเฉพาะหมายเลข ที่อยู่ กรองในช่วงหมายเลข หรือรับทั้งหมด



รูปที่ 5.13 แสดงการเลือกหมายเลข IP ADDRESS ที่จะรับ

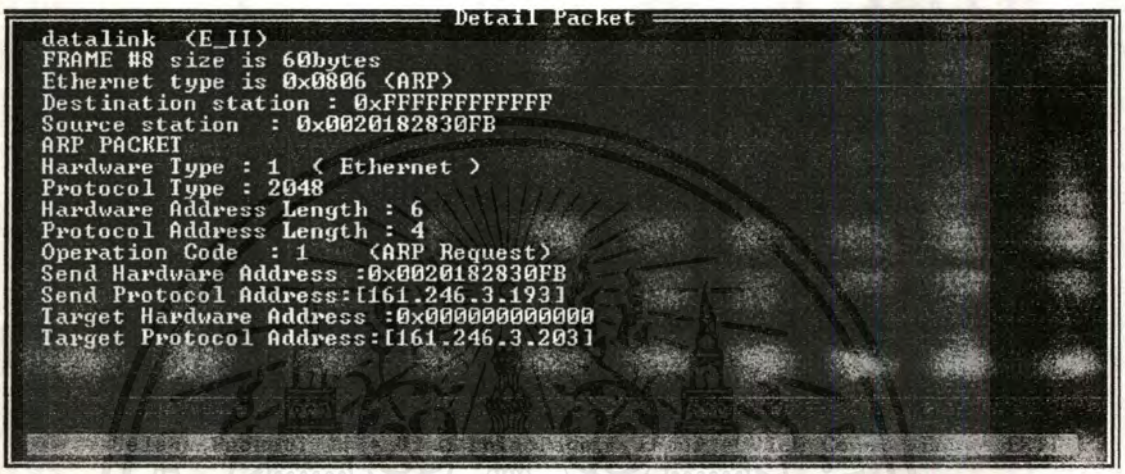
- **SHOW CONFIGURE ALL** เป็นการแสดงค่าที่เราตั้งทั้งหมดดังรูป



รูปที่ 5.14 แสดงผลการตั้งค่าทั้งหมด

5.1.4 VIEW จะมีเมนูย่อยในการวิเคราะห์และแสดงผล 3 เมนูคือ

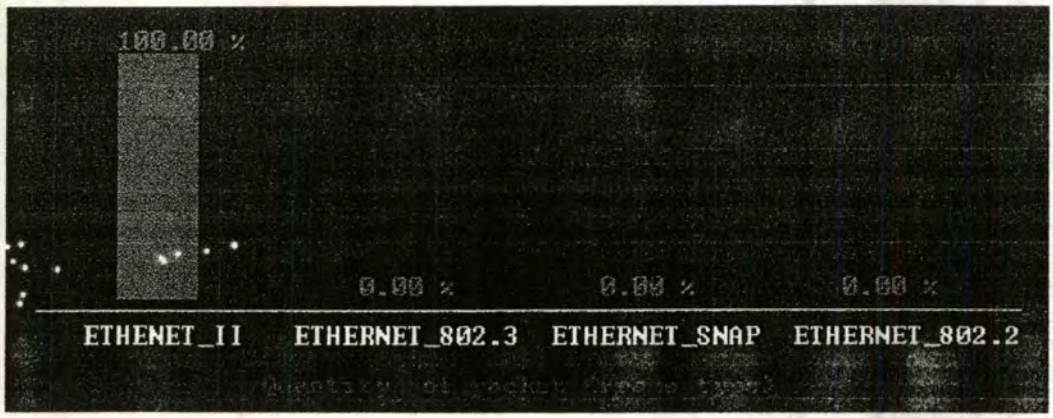
- PACKET DATA DETAIL เป็นการนำข้อมูลจากตัวแปรบัฟเฟอร์มาวิเคราะห์โพรโตคอลในระดับชั้นต่างๆ และนำมาแสดงผลที่หน้าจอโดยสามารถที่จะดูข้อมูลดิบได้โดยการกดคีย์ Enter ในการดูข้อมูลหากข้อมูลมีขนาดยาวกว่าหนึ่งหน้าจอแสดงผลสามารถกด คีย์ลูกศรขึ้นลงเพื่อดูข้อมูลทั้งหมด และกดคีย์ซ้าย,ขวาเพื่อเพิ่มหรือลดลำดับของแพคเก็ตที่จะนำมาวิเคราะห์ได้



รูปที่ 5.15 การแสดงผลการวิเคราะห์และถอดรหัสของแพคเก็ต

รูปที่ 5.16 แสดงการนำข้อมูลดิบออกมาแสดงผล

- **STATISTIC OF FRAME TYPE** เป็นการแสดงผลสถิติโดยเปรียบเทียบชนิดของเฟรมแต่ละชนิด



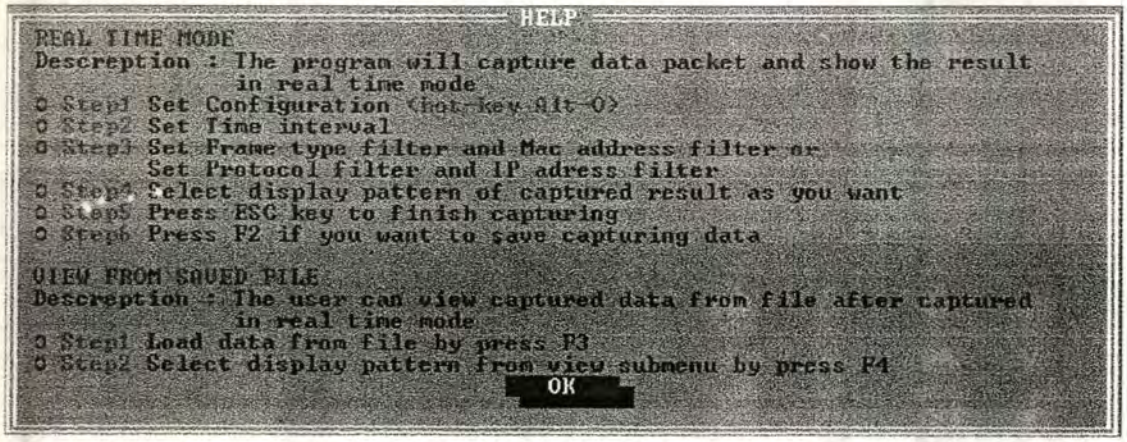
- รูปที่ 5.17 แสดงผลสถิติ โดยเปรียบเทียบชนิดของเฟรมแต่ละชนิด

- **STATISTIC OF PROTOCOL** เป็นการแสดงผลสถิติโดยเปรียบเทียบชนิดของโพรโตคอลแต่ละชนิด

รูปที่ 5.18 แสดงผลสถิติโดยเปรียบเทียบชนิดของโพรโตคอลแต่ละชนิด

- HELP

- HELP เป็นการแสดงวิธีการดักจับข้อมูลเบื้องต้น



รูปที่ 5.19 รูปแสดงผลการใช้เมนู HELP

-ABOUT เป็นการแสดงรายละเอียดเฉพาะของการจัดทำโปรแกรม

รูปที่ 5.20 รูปแสดงผลการใช้เมนู ABOUT

5.2 การทดสอบโปรแกรม

5.2.1 ขั้นตอนการทดสอบโปรแกรม

- Run โปรแกรมบนระบบปฏิบัติการ DOS ที่เครื่องคอมพิวเตอร์เชื่อมต่อกับระบบเครือข่ายอีเทอร์เนต
- ทดสอบการดักจับโปรแกรม โดยใช้วิธีแสดงผลบนหน้าจอคอมพิวเตอร์ในทุกรูปแบบให้สามารถใช้งานได้ตามรูปแบบที่กำหนด
- ทดลองตั้งเวลาในเมนู SET TIME INTERVAL ในทุกค่าเวลาที่กำหนดไว้ ให้สามารถทำงานตามเวลาที่กำหนดได้

- ทำการบันทึกข้อมูลลงในดิสก์ และลองทดสอบเปิดแฟ้มข้อมูลด้วยโปรแกรม อิดิเตอร์ (Editor) ว่าโปรแกรมได้เก็บข้อมูลได้ถูกต้องหรือไม่ดังแสดงในรูป

รูปที่ 5.20 แสดงข้อมูลในแฟ้มข้อมูลที่บันทึกไว้

- LOAD ข้อมูลจากดิสก์มาเก็บไว้ที่ตัวแปรบัฟเฟอร์
- ทดลองทำการวิเคราะห์และแสดงผล และดูผลการถอดรหัสว่าถูกต้องหรือไม่ สถิติและข้อมูลดิบที่แสดงผลมีผลลัพท์ตรงกันหรือไม่

บทที่ 6
บทสรุปและวิจารณ์

6.1 บทสรุป

โครงงานที่จัดทำขึ้นนี้เป็นโครงการสำหรับช่วยในการศึกษาการทำงานภายในเครือข่าย เพื่อให้เข้าใจถึงการทำงานของโพรโตคอลในแต่ละชั้น และสามารถเฝ้าดูการใช้งานในระบบเครือข่ายเพื่อประโยชน์ทางการปรับปรุงและพัฒนาระบบเครือข่าย โดยที่สามารถทราบถึงปริมาณการใช้งาน ที่เกิดขึ้นภายในระบบเครือข่าย เนื่องจากปัจจุบันการใช้งานคอมพิวเตอร์ (Computer) ในรูปแบบระบบเครือข่าย (Network) ได้เข้ามามีบทบาทเพื่อช่ายเพิ่มประสิทธิภาพในการทำงานมากยิ่งขึ้นโดยเฉพาะในองค์กรใหญ่ๆ จะสามารถใช้ประโยชน์จากการใช้ทรัพยากรร่วมกันหรือจากการทำงานเป็นระบบเครือข่าย จึงได้ทำการพัฒนาโปรแกรม (Program) ขึ้นเพื่อใช้เฝ้าดูและวิเคราะห์แพคเก็ต (Packet) ในระบบเครือข่ายท้องถิ่น (LAN : Local Area Network) โดยทำการตรวจสอบแพคเก็ตที่ได้รับจากสื่อ ผ่านการทำงานของแพคเก็ตไดร์เวอร์ (Packet Driver) ซึ่งจะรับแพคเก็ตจากการ์ดเชื่อมต่อเน็ตเวอร์ก (NIC : Network Interface Card) เข้ามาวิเคราะห์ถึงส่วนต่าง ๆ ที่จำเป็นต่อการศึกษาการทำงานและตรวจสอบระบบเครือข่าย

6.2 ปัญหาในการพัฒนาโปรแกรม

ส่วนของปัญหาที่เกิดขึ้นในการพัฒนาโปรแกรมสามารถแบ่งเป็นข้อๆ ได้ดังนี้คือ

- ข้อมูลของการถอดรหัสโพรโตคอลต่างๆ นั้นไม่สามารถหาได้ครบทำให้ไม่สามารถถอดรหัสของโพรโตคอลได้ทุกโพรโตคอลที่มีการใช้งานในปัจจุบัน

- ในการดักจับแพคเก็ตนั้น หากเราทำการเก็บข้อมูลลงดิสก์เลยทันทีจะทำให้การทำงานช้าลงทำให้ประสิทธิของการดักจับแพคเก็ตนั่นก็จะลดลง จึงต้องใช้ตัวแปรบัฟเฟอร์มาเก็บข้อมูลไว้ก่อน ซึ่งในแต่และแพคเก็ตสามารถมีค่าสูงสุดได้ 1500 ไบต์ โครงสร้างของโปรแกรมภาษาป่าสคาลนั่นไม่สามารถจองตัวแปรไว้ได้มากจึงต้องทำการเก็บเฉพาะข้อมูลที่ใช้งานต่อๆกันไป โดยต้องเก็บขนาดของแต่ละแพคเก็ตไว้เป็นตัวแยกข้อมูลแต่ละแพคเก็ตออกจากกัน

6.3 แนวทางการพัฒนาต่อ

- สามารถวิเคราะห์โพรโตคอลในแต่ละเลเยอร์ได้มากขึ้น โดยเฉพาะอย่างยิ่งโพรโตคอลที่ใช้ในเน็ตแวร์แลน ทำให้สามารถดูการใช้งานของเครือข่ายได้มากขึ้น
- สามารถดักจับแพคเก็ตได้มีความถูกต้องเพิ่มมากขึ้น
- สามารถพัฒนาไปใช้ในระบบวินโดว์ได้โดยผ่าน NDIS (Network Driver Interface Specification) ของวินโดว์
- สามารถใช้งานกับระบบเครือข่ายชนิดอื่นๆได้เช่น โทเค็นริง เป็นต้น

เอกสารอ้างอิง

1. สุวิพล สุทธิชีวภาค " เทคโนโลยีของโลคัลเนตเวอร์ค" ,พิมพ์ที่ Japan International Cooperation Agency (JICA), 2535, หน้า 46-48
2. Luara A. Chappep and Dan E. Hakes, "Netware Lan Analysis", SYBEX Inc., 693p.,1994
3. Novell. "LANanalyzer for Windows 2.2 Installation and User's Guide",Novell Inc., 244p.,1996
4. J.L. Hammond, "Performance Analysis of Local Computer Network",Addison-Wesley.,725p.,1986
5. Ed Tayler, "Internetworking Handbook", McGraw-Hill Inc.,450p.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ทั้งด้านข้อมูลและด้านโปรแกรมด้วยความช่วยเหลือจาก อาจารย์ มยุรี เลิศเวชกุล และ อาจารย์ พิทักษ์ ธรรมวาริน ซึ่งเป็นอาจารย์ที่ปรึกษา ที่คอยให้คำแนะนำและปรึกษาปัญหาที่เกิดขึ้น รวมทั้งชี้แนะแนวทางในการพัฒนาโปรแกรมให้เป็นไปอย่างมีประสิทธิภาพ

ขอขอบคุณ ห้องสมุดของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่เป็นแหล่งรวบรวมความรู้ที่ใช้อ้างอิงในการทำโครงงานครั้งนี้ รวมทั้งห้องปฏิบัติการ คอมพิวเตอร์ ตึก A ที่เป็นสถานที่ทดสอบโปรแกรม

นาย มาโนช พิชผล

นาย ศรายุทธ จันทร์เมืองไทย

ภาคผนวก

ภาคผนวก ก

รูปแบบส่วนหัวของแต่ละโพรโตคอล

Ethernet II Frame	
Destination Address	6 Bytes
Source Address	6 Bytes
Type	2 Bytes
Data	46-1500 Bytes

Ethernet 802.3 Frame	
Destination Address	6 Bytes
Source Address	6 Bytes
Length	2 Bytes
Data	46-1500 Bytes (เริ่มต้นด้วย 0xFFFF)

Ethernet 802.2 Frame	
Destination Address	6 Bytes
Source Address	6 Bytes
Type	2 Bytes
DSAP	1 Bytes
SSAP	1 Bytes
Control	1 Bytes
Data	43 – 1497 Bytes

Ethernet SNAP Frame	
Destination Address	6 Bytes
Source Address	6 Bytes
Type	2 Bytes
DSAP	1 Bytes
SSAP	1 Bytes
Control	1 Bytes
Organization Code	3 Bytes

Ethernet Type	2 Bytes
Data	38 – 1492 Bytes

IPX Packet	
Check Sum	2 Bytes
Length	2 Bytes
Transport	1 Bytes
Packet Type	1 Bytes
Destination Network	4 Bytes
Destination Host	6 Bytes
Destination Socket	2 Bytes
Source Host	6 Bytes
Source Socket	2 Bytes
Data	

IP Packet	
Version + Header Length	1 Bytes (1/2 Bytes + ? Bytes)
Type Of Service	1 Bytes
Length	2 Bytes
Identifier	2 Bytes
Flag	3 Bytes
Fragment Offset	13 Bytes
Time To Live	1 Bytes
Protocol	1 Bytes
Header Check Sum	1 Bytes
Source Address	4 Bytes
Destination Address	4 Bytes
Option	Variable
Data	

ARP Packet	
Hardware Type	2 Bytes
Protocol Type	2 Bytes
Hardware Address Length	1 Bytes
Protocol Address Length	1 Bytes
Operation Code	2 Bytes
Send Hardware Address	6 Bytes
Send Protocol Address	4 Bytes
Target Hardware Address	6 Bytes
Target Protocol Address	4 Bytes

SPX Packet	
Control	1 Bytes
Data Type	1 Bytes
Source ID	2 Bytes
Destination ID	2 Bytes
Sequence Number	2 Bytes
Acknowledgement Number	2 Bytes
Allocation Number	2 Bytes
Data	

NCP Request Packet	
Request Type	2 Bytes
Sequence Number	1 Bytes
Connection Number Low	1 Bytes
Task Number	1 Bytes
Connection Number High	1 Bytes
Data	

NCP Reply Packet	
Request Type	2 Bytes
Sequence Number	1 Bytes
Connection Number Row	1 Bytes
Task Number	1 Bytes
Connection Number High	1 Bytes
Completion Code	1 Bytes
Connection Status	1 Bytes
Data	

ICM Packet	
Type	1 Bytes
Code	1 Bytes
Check Sum	2 Bytes
Data	

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

Echo and Echo Reply Message	
Type	1 Bytes (0 or 8)
Code	1 Bytes
Check Sum	2 Bytes
Identifier	2 Bytes
Sequence Number	2 Bytes
Data	

Information and Information Reply Message	
Type	1 Bytes (0 or 8)
Code	1 Bytes
Check Sum	2 Bytes
Identifier	2 Bytes
Sequence Number	2 Bytes

Destination Unreachable Message + Source Quench Message + Time Exceeded Message	
Type	1 Bytes (3,4 or 11)
Code	1 Bytes
Check Sum	2 Bytes
Unused	4 Bytes
Data	

Redirect Message	
Type	1 Bytes (5)
Code	1 Bytes
Check Sum	2 Bytes
Gateway Internet Address	4 Bytes
Data	

Parameter Problem Message	
Type	1 Bytes (12)
Code	1 Bytes
Check Sum	2 Bytes
Pointer	1 Bytes
Unused	3 Bytes
Data	
Timestamp and Timestamp Reply Message	
Type	1 Bytes (13 or 14)
Code	1 Bytes
Check Sum	2 Bytes
Original Timestamp	4 Bytes
Receive Timestamp	4 Bytes
Transmit Timestamp	4 Bytes

TCP Packet	
Source Port	2 Bytes
Destination Port	2 Bytes
Sequence Number	4 Bytes
Acknowledgement Number	4 Bytes
Header Length	1 Bytes
Code Bits	1 Bytes
Windows	2 Bytes
Check Sum	2 Bytes
Urgent Pointer	2 Bytes
Option	Variable
Data	

UDP Packet	
Source Port	2 Bytes
Destination Port	2 Bytes
Length	2 Bytes
Check Sum	2 Bytes
Data	

RIP Packet	
Command	1 Bytes
Version	1 Bytes
Zero	2 Bytes
Address family ID	2 Bytes
Zero	2 Bytes
Address Family ID	2 Bytes
Zero	2 Bytes
IP Address	4 Bytes
Zero	8 Bytes
Distance to network n	4 Bytes
Can Repeat from first Address Family Id Fields	

ภาคผนวก ข

SOURCE CODE

```
program ether;
{
 Compiler      :  Turbo Pascal 7.0
 OS-Version    :  MS-DOS  6.22
 Last edit     :  May,10,2000
 Version       :  1.0
}
{$S-}      { stack checking off }
{$F+}      { force far calls }
uses  Dos,Crt,Screen,KeyBoard,Win,menu1,dir2;
type
     hwaddr = array[0..5] of byte;
     ipaddr = array[1..4] of byte;
     frame = Array[00..1524] of Byte;

{ the information returned from a Get_Driver_Info() call. }
     drvinfo = record
      version : word;
      e_class : byte;
      e_type : word;
      number : byte;
      _name : pointer;
      functionality : byte;       { 1=basic, 2=basic+extended }
     end;

{ Specification of parameters for access_type() call }
     eth_drv_access = record
      class : byte;
      etype : word;               { ...of interface }
      number : byte;              { ...of the interface }
      packet_type : pointer;      { ...two-byte type spec. }
      packet_type_length : word;  { ...of above data. }
      receiver : pointer;         { ...to ether_receiver routine }
```

```
    end;
{the data of upper layer}
    cupture_upper = record
      value      : word;      {quantity packet of upper layer}
      size       : word;      {quantity data of upper layer}
    end;


const
    window_c      = $30;
    text_c        =$30;
    text_c_inv    = $1f;
    tab  = #24;
var
    src_all       : boolean;                { status of source address filter }
    des_all       : boolean;                { status of Destination address filter }
    ip_src,ip_des,                          { specified ip address}
    ip_src1,ip_src2,                        { rang address of source ip address }
    ip_des1,ip_des2  : ipaddr;              { rang address of destination ip
address }
    ip_add_src,ip_add_des : byte;           { value of menu protocol filter }
    des_add,src_add  : string;              { specified  mac address}
    menutime      :  byte ;                 { value of menu timeinterval}
    cap_ok        : boolean;                { status of filter packet}
    capture_up    : array[0..16] of cupture_upper;   { Array of  size and quantity
packet}
    filtereth     : array[1..4] of boolean;          { status of frame type filter}
    filternet     : array[1..3] of boolean;          { status of protocol type filter}

    Hour,Minute,Second,Sec100                  { pressent time  }
    ,Hour2,Minute2,Second2,Sec1002    : word;      { capture time }

    value_array   : array[00..1000] of integer;    {array of quantity  packet}
    size_array    : array[00..1000] of word ;      {array of quantity  data}
    value_all     : word;                          {sum of quantity  packet}
```

```
    size_all        : word;                    {sum of quantity  data}
    timeAssing      : integer;                   {value of time assignment}
    maxtime         : integer  ;                 {max time }
    menu_show       : byte;                      {value of menu display pattern}
    dataf           : array[0..40000] of byte;     {array of raw data}
    long            : array[0..250] of word;     {array of length packet}
    line            : array[0..60] of string[80];  {array of string buffer}
    sline,pline     : byte;                    {pointer of string buffer}
    datafp1,datafp  : word;                      { pointer of raw data }
    longp,longp1    : integer;                   { packet length }
    driver_intr,                               { interupt vecter }
    driver_Result   : byte;                      { Result of Call Function }
    driver_loaded   : boolean;                   { Status of driver (load or not)  }
    Regs            : Registers;               { Register }
    TypeField       : Word;                    { Length Field}
    showPacket                                 { RCV buffer for analysis}
    ,RCVpacket      : frame ;                    { Rcv buffer      }
    RcvLength       : Word;                    { Length of packet }
    PacketCount     : Word;                      { Packet counter   }
    Layer2,layer3,Layer4  : string;              {name of protocol  }
    P_Type                                     {field type of ethernet}
    ,slayer3,slayer4  : word;                    {start byte of layer}
    my_address      : hwaddr;                    {address of interface}
    save_data       : boolean;                 {status of saving data}
    k_scale,                                   {max scale}

    g,h,i,j,k: integer;
    st          : string;

{ The receiver procedure:                                  }
{$S-}
PROCEDURE pktReceiver; ASSEMBLER;
ASM
    PUSH AX              { Push registers onto stack }
    PUSH BX
```

```
PUSH CX
PUSH DX

CMP  AX,0001            { AX=1 means frame copied }
JZ   @@FrameCopied
CMP  AX,0000            { AX=0 means allocate memory please }
JZ   @@AllocMemory
JMP  @@EXIT             { Invalid register contents for AX so exit}

@@AllocMemory:

MOV  DX,0               { ES:DI = 0000:0000, we don't want the packet }
MOV  ES,DX
MOV  DI,0               { We don't grab the packet }

MOV  DX,SEG PacketCount     { Set correct data segment }
MOV  DS,DX
MOV  DX,PacketCount
CMP  DX,0

JNZ  @@Exit             { buffer is not free ! }

MOV  DX,SEG rcvPacket
MOV  ES,DX
MOV  DI,OFFSET rcvPacket

MOV  DX,SEG rcvLength
MOV  DS,DX
MOV  SI,OFFSET RcvLength
MOV  WORD PTR DS:[SI],CX    { Store length of frame in PacketLength }

JMP  @@Exit

@@FrameCopied:
```

```
    MOV  DX,SEG PacketCount      { Set correct data segment }
    MOV  DS,DX
    MOV  PacketCount,1           { Set Flag to 1 }

    @@Exit:

    POP  DX                  { Pop registers from stack }
    POP  CX
    POP  BX
    POP  AX
END;
{$S+}

function Access_Type( access : eth_drv_access ) : word;
{   Initiates access to packets of IEEE 802.3 type.
    Some typical values for the fields of 'access'.
    class = 1                packet class (802.3)
    etype = $ffff            packet type ($ffff=match any incoming)
    number = 0                packet number (0=first device)
    packet_type = 0:0          seg pointer to packet type spec
                          typelen= 0 to match all packet
}

begin
    Regs.AH := 2;                  { select function          }
    with access do begin
      Regs.AL := class;              { packet class (802.3)          }
      Regs.BX := etype;              { packet type ($ffff=match any)   }
      Regs.DL := number;               { packet number (0=first device)  }
      Regs.DS := Seg( packet_type );    { seg pointer to packet type spec }
      Regs.SI := Ofs( packet_type );    { ofs   "   "   "   "   "  }
      Regs.CX := packet_type_length;    { typelen= 0 to match all packets }
      Regs.ES := Seg( receiver^ );      { seg & ofs of receiving routine  }
      Regs.DI := Ofs( receiver^ );
    end;
```

```
      Intr(driver_Intr,Regs );
      driver_Result := Regs.DH;
      access_Type := Regs.AX;
end;


PROCEDURE SetRCVmode(handle,Mode : Word);{ Sets the receive mode of the driver }
begin
      Regs.AH := 20;
      Regs.BX := Handle;
      Regs.CX := Mode;
      Intr(driver_Intr,Regs );
      driver_Result := Regs.DH;
end;


procedure Get_Driver_Info(    handle : word; var info : drvinfo );
{ Returns some information on the assigned packet driver}
begin
      Regs.AH := 1;
      Regs.AL := 255;
      Regs.BX := Handle;
      Intr(driver_Intr,Regs );
      With info do begin
         version := Regs.BX;
         e_class := Regs.CH;
         e_type := Regs.DX;
         number := Regs.CL;
         _name := Ptr( Regs.DS,Regs.SI );
         functionality := Regs.AL;
      end;
      driver_Result := Regs.DH;
end;
```

```
procedure Get_Pkt_hwaddr(   handle : word;var addr : hwaddr );
{ Get the hardware address of the card, write it into 'addr'}
begin
    Regs.AH := 6;
    Regs.BX := Handle;
    Regs.ES := Seg( addr );
    Regs.DI := Ofs( addr );
    Intr(driver_Intr,Regs );
    driver_Result := Regs.DH;
end;

function Driver_Present : byte;
var   i,j,c : byte;
    vec : pointer;
    sstr : string[13];
{ search for an ftp spec. packet driver between interrupts $60 and $80 }
begin
    j := 0;
    for i := $60 to $80 do
    begin
      GetIntVec( i, vec );
      sstr[0] := Chr(8);
      for c := 3 to 10 do sstr[c-2] := Chr(mem[Seg(vec^):Ofs(vec^)+c]);
      if (sstr='PKT DRVR') then j := i;
    end;
    Driver_Present := j;
end;

function hex_byte( s1 : char ) : byte; { change charecter(0-9,A-F) to byte}
begin
    case s1 of
      '0' : hex_byte := 0;
      '1' : hex_byte := 1;
```

```
        '2' : hex_byte := 2;
        '3' : hex_byte := 3;
        '4' : hex_byte := 4;
        '5' : hex_byte := 5;
        '6' : hex_byte := 6;
        '7' : hex_byte := 7;

        '8' : hex_byte := 8;
        '9' : hex_byte := 9;
        'A' : hex_byte :=10;
        'B' : hex_byte :=11;
        'C' : hex_byte :=12;
        'D' : hex_byte :=13;
        'E' : hex_byte :=14;
        'F' : hex_byte :=15;
    end;
end;



function Get_hex( i : byte ) : string;{ returns hex digit where i is a nibble }
var   t : byte;
    u,l : char;
begin
    t := i mod 16;
    i := i div 16;
    if (t<10) then l := chr(t+48) else l := chr(t+55);
    if (i<10) then u := chr(i+48) else u := chr(i+55);
    Get_hex := Concat( u,l );
end;



function Word_hex(i: word) : string; {change word to string(heximal)}
var   str : string;
begin
```

```
        1: Writeln( 'Basic driver functions present.' );
        2: Writeln( 'Basic + extended driver functions present.' );
        5: Writeln( 'Basic + high-performance functions present.' );
        6: Writeln( 'Basic, high-performance + extended functions.' );
      end;
      end;
end;


procedure fwriteln(s1 : string);{ write result of anlysis to string buffer}
begin
      line[sline] := s1;
      sline := sline+1;
end;


function strr(s1 : word) : string;{ change word to string(decimal)}
var    st : string;
begin
      str(s1,st);
      strr := st;
end;



function twohex(b1,b2 : byte) : word;

begin
      twohex := (b1*$100 + b2);
end;


function disbin(b1 : byte) : string;{chang byte to string(binary)}
begin
       case b1 of
```

```
        0  : disbin := '0000';
        1  : disbin := '0001';
        2  : disbin := '0010';
        3  : disbin := '0011';
        4  : disbin := '0100';
        5  : disbin := '0101';
        6  : disbin := '0110';
        7  : disbin := '0111';
        8  : disbin := '1000';
        9  : disbin := '1001';
        10 : disbin := '1010';
        11 : disbin := '1011';
        12 : disbin := '1100';
        13 : disbin := '1101';
        14 : disbin := '1110';
        15 : disbin := '1111';
    end;
end;


function chrr(b1 : byte) : char;
{ change byte to charecter}
begin
    if b1 > $80 then b1 := b1-$80;
    chrr := chr(b1);
end;


function kbyte(b1 : longint) : integer; {change longint to integer}
var div1,mod1,div2,mod2,div3,mod3 : integer;
begin
    div1 :=  b1 div 1000;
    mod1 :=  b1 mod 1000;
    div2 := mod1 div 100;
    mod2 := mod1 mod 100;
```

```
    str := Get_hex(hi(i));
    str := concat(str,Get_hex(lo(i)));
    Word_hex := str;
end;


function String_hwaddr( a : hwaddr ) : string;{change 6 byte (hardware address) to
string}
begin
    String_hwaddr := Concat( Get_hex(a[0]),':', Get_hex(a[1]),':',
                    Get_hex(a[2]),':', Get_hex(a[3]),':',
                    Get_hex(a[4]),':', Get_hex(a[5])    );
end;


procedure write_driver_info( info : drvinfo );{ display information of packet driver }
var   count : byte;
    by : byte;
    n_seg, n_ofs : word;
begin
    with info do begin
    Write( 'Version: ',version, '.  ' );
    Write( 'Class: ',e_class, '.  ' );
    Write( 'Type: ',e_type,'.  ' );
    Write( 'Number: ',number, '.  ' );
    Write( '_name: ' );
    count := 0;
    repeat
    by := Mem[Seg(_name^):Ofs(_name^)+count];
    if by <> 0 then Write( Chr(by) );
    inc(count);
    until ((count>70) or (by = 0));
    WriteIn;
    write( 'Functionality: ' );
    case functionality of
```

```
        div3 := mod2 div 10;
        mod3 := mod2 mod 10;
        if mod3 >=5 then div3  := div3+1;
        if div3 >=5 then div2  := div2+1;
        if div2 >=5 then div1  := div1+1;
        kbyte := div1;
    end;


    procedure filterprotocol;{ set  Filter mac address }
    var key              : char;
      msg                : array[1..4] of string  ;
      old                : array[1..4] of boolean;
    begin
        for i := 1 to 4 do  old[i] := filtereth[i];
        k := 1;
        msg[1] := ' 1   ETHERNET II    ' ;
        msg[2] := ' 2   ETHERNET 802.2 ' ;
        msg[3] := ' 3   ETHERNET 802.3 ' ;
        msg[4] := ' 4   ETHERNET SNAP  ' ;
        Cursoroff;
        SetwinHeader('SELECT ETHERNET');
        SetWinAttr($30);
        SetBoxAttr($30);
        SetCharAttr($30);
        Setboxstyle(double);
        SetHeadAttr($30);
        Windowopen(25,10,55,16);
        repeat
        for  i := 1 to 4 do
        begin
        gotoxy(4,i*1);
        SetAttr($30);
        if i = k  then  setAttr($7f);
        if  filtereth[i] then st := '[X]' else st := '[ ]';
```

```
    write(st,msg[i]);
    end;

    setAttr($40);
    if k = 5 then  setAttr(S7e);
    gotoxy(12,5);
    write(' OK ');

    key := readkey;

        case (Key) of
         up_key   :   if k = 1 then k := 5 else k := k - 1;
         Dn_Key   :   if k = 5 then k := 1 else k := k + 1;
         PgUp_Key :   k := 1;
         PgDn_Key :   k := 4;
         return_key:

         if (k >0) and (k <=4) then
         if  filtereth[k] = true then filtereth[k] := false else filtereth[k] := true;
         esc_key :    for i := 1 to 4  do   filtereth[i]  :=  old[i]
        end;

    until  (key = esc_key)or((key = return_key) and (k = 5)) ;
     windowclose;
     end;

procedure filterprotocol2;
{ Set  filter protocol }
var key              : char;
   msg               : array[1..3] of string  ;
   old               : array[1..3] of boolean;
begin
    for i := 1 to 3 do  old[i] := filternet[i];
```

```
k := 1;
msg[1] := ' 1   IPX PROTOCOL   ' ;
msg[2] := ' 2   ARP PROTOCOL   ' ;
msg[3] := ' 3   IP  PROTOCOL   ' ;
Cursoroff;
SetwinHeader('SELECT PROTOCOL');
SetWinAttr($30);
SetBoxAttr($30);
SetCharAttr($30);
Setboxstyle(double);
SetHeadAttr($30);
Windowopen(25,10,55,15);
repeat
for  i := 1 to 3 do
begin
gotoxy(4,i*1);
SetAttr($30);
if i = k  then  setAttr($7f);
if  filternet[i] then st := '[X]' else st := '[ ]';
write(st,msg[i]);
end;

setAttr($40);
if k = 4 then  setAttr($7e);
gotoxy(12,4);
write(' OK ');

key := readkey;

    case (Key) of
      up_key    :    if k = 1 then k := 4 else k := k - 1;
      Dn_Key    :    if k = 4 then k := 1 else k := k + 1;
      PgUp_Key  :    k := 1;
      PgDn_Key  :    k := 3;
      return_key:
```

```
            if (k >0) and (k <=3) then
            if  filternet[k] = true then filternet[k] := false else filternet[k] := true;
            esc_key :    for i := 1 to 3  do   filternet[i]  :=  old[i]
          end;



    until  (key = esc_key)or((key = return_key) and (k = 4)) ;
     windowclose;
     end;


procedure desired(b1 : byte);
{ write desired access to string buffer}
begin
    fwriteln('  ...'+strr((b1 and $10)div$10)+' ....    = Exclusive Access  ');
    fwriteln('  .... '+strr((b1 and $08)div$08)+'...    = Deny Write        ');
    fwriteln('  ..... .'+strr((b1 and $04)div$04)+'..   = Deny Only         ');
    fwriteln('  .... ..'+strr((b1 and $02)div$02)+'.    = Write Permission  ');
    fwriteln('  .... ...'+strr((b1 and $01)div$01)+'    = Read Permission   ');
end;

procedure dosatt(b1 : byte);
{ write dos attribute to string buffer}
begin
    fwriteln('  ..'+strr((b1 and $20)div$20)+'. ....    = Archive     ');
    fwriteln('  ...'+strr((b1 and $10)div$10)+' ....    = Directory   ');
    fwriteln('  .... '+strr((b1 and $08)div$08)+'...    = Volume Label');
    fwriteln('  ..... .'+strr((b1 and $04)div$04)+'..   = System      ');
    fwriteln('  .... ..'+strr((b1 and $02)div$02)+'.    = Hidden      ');
    fwriteln('  .... ...'+strr((b1 and $01)div$01)+'    = Read Only   ');
end;
```

```
procedure netwareatt(b1 : byte);
{ write netware attribute to string buffer}
begin
    fwriteln('   .'+strr((b1 and $40)div$40)+'.. ....    = Sharable    ');
    fwriteln('   ..'+strr((b1 and $20)div$20)+'. ....    = Archive     ');
    fwriteln('   ...'+strr((b1 and $10)div$10)+' ....    = Directory   ');
    fwriteln('   .... '+strr((b1 and $08)div$08)+'...    = Excute Only ');
    fwriteln('   .... .'+strr((b1 and $04)div$04)+'..    = System      ');
    fwriteln('   .... ..'+strr((b1 and $02)div$02)+'.    = Hidden      ');
    fwriteln('   .... ...'+strr((b1 and $01)div$01)+'    = Read Only   ');
end;

procedure checkall(spacket :frame);{ analysis layer of frame packet }
var packet_long     : word;
begin
    LAYER2 := ' ';
    LAYER3 := ' ';
    LAYER4 := ' ';
    SLAYER3 := 0;
    SLAYER4 := 0;

    packet_long := (sPacket[12]*$100) + sPacket[13];
    if  packet_long  >  1500  then
      begin
       layer2 := 'E_II' ;
       slayer3 := 14   ;
      end
    else
    if (sPacket[14] and sPacket[15] ) = $ff then
      begin
       layer2 := 'E_802.3';
       sLayer3 := 14    ;
      end
    else
```

```
if (sPacket[14] and sPacket[15] ) = $aa then
   begin
    layer2 := 'E_Snap' ;
    sLayer3 := 22     ;
   end
else
   begin
    Layer2 := 'E_802.2' ;
    sLayer3 := 17;
   end;

if layer2 = 'E_II'   then P_Type :=  twohex(sPacket[12],sPacket[13]);
if layer2 = 'E_Snap' then P_type :=  twohex(sPacket[20],sPacket[21]);
if (layer2 = 'E_802.3') or (Layer2 = 'E_802.2') then P_type := $8137;
case P_type of
  $0800 :  begin
           Layer3 := 'IP';
           slayer4 :=  (slayer3)+((spacket[slayer3+0] mod 16 )*4);
         end;
  $0806 :    Layer3 := 'ARP';
  $8137 :  begin
           Layer3 := 'IPX';
           slayer4 := slayer3+30;
         end;
  else      Layer3 := 'UNKNOW';
end;

if layer3 = 'IPX' then
case sPacket[slayer3+5] of
  $01  : Layer4 := 'RIP';
  $02  : Layer4 := 'ECHO';
  $03  : Layer4 := 'ERROR';
  $04  : Layer4 := 'PEP';
  $05  : Layer4 := 'SPX';
  $11  : Layer4 := 'NCP';
```

```
        else   layer4 := 'UNKNOWN';
      end;

      if Layer3 = 'IP' then
      case  spacket[slayer3+9] of
        $01 : Layer4 :=  'ICMP';
        $06 : Layer4 :=  'TCP' ;
        $11 : Layer4 :=  'UDP' ;
        else  Layer4 :=  'UNKNOWN';
      end;
end;



procedure checklayer2; { analysis information in datalink layer}
begin
      fwriteln('  datalink  ('+layer2+')');
      fwriteln('  FRAME #'+strr(longp1)+' size is '+strr(long[longp1])+'bytes');
      if (layer2 = 'E_II')   then fwriteln('  Ethernet type is 0x'+get_hex(showPacket[12])
                                  +get_hex(showPacket[13])+' ('+layer3+')');
      if (layer2 = 'E_Snap') then fwriteln('  Ethernet type is 0x'+get_hex(showPacket[20])
                                  +get_hex(showPacket[21])+' ('+layer3+')');
      fwriteln('  Destination station : 0x'+ get_hex(showpacket[0])+ get_hex(showpacket
[1])
                                  + get_hex(showpacket[2])+ get_hex(showpacket[3])
                                  + get_hex(showpacket[4])+ get_hex(showpacket[5]));



      fwriteln('  Source station  : 0x'    + get_hex(showpacket[6])+ get_hex(showpacket[7])
                                  + get_hex(showpacket[8])+ get_hex(showpacket[9])
                                  + get_hex(showpacket[10])+ get_hex(showpacket[11]));
end;



procedure checkarp;{ analysis information in ARP protocol}
```

```
begin
    if layer3  = 'ARP' then
      begin
      case twohex(showpacket[slayer3+6],showpacket[slayer3+7])  of
         1,2  : fwriteln('  ARP PACKET');
         3,4  : fwriteln('  RARP PACKET');
      else
         writeln('  UNKNOW PACKET');
      end;

    if twohex(showpacket[slayer3+0],showpacket[slayer3+1]) = 1 then st := ('  ( Ethernet
)')
    else st := '';
    fwriteln('  Hardware Type : '+strr(twohex(showpacket[slayer3+0],showpacket
[slayer3+1]))+st);
    fwriteln('  Protocol Type : '+strr(twohex(showpacket[slayer3+2],showpacket
[slayer3+3])));
    fwriteln('  Hardware Address Length : '+strr(showpacket[slayer3+4]));
    fwriteln('  Protocol Address Length : '+strr(showpacket[slayer3+5]));

    case twohex(showpacket[slayer3+6],showpacket[slayer3+7])  of
       1  : st := ('(ARP Request)  ');
       2  : st := ('(ARP Response) ');
       3  : st := ('(RARP Request) ');
       4  : st := ('(RARP Response)');
       else st := ('               ');
    end;
    fwriteln('  Operation Code  : '+strr(twohex(showpacket[slayer3+6],showpacket
[slayer3+7]))+
                          ('    ')+st);

    fwriteln('  Send Hardware Address :0x'+get_hex(showpacket[slayer3+8])+get_hex
(showpacket[slayer3+9])
                          +get_hex(showpacket[slayer3+10])+get_hex(showpacket
[slayer3+11])
```

```
                    +get_hex(showpacket[slayer3+12])+get_hex(showpacket
[slayer3+13]));

    fwriteln('  Send Protocol Address:['
        +strr(showpacket[slayer3+14])+'.'+strr(showpacket[slayer3+15])+'.'
        +strr(showpacket[slayer3+16])+'.'+strr(showpacket[slayer3+17])+']');

    fwriteln('  Target Hardware Address :0x'
        +get_hex(showpacket[slayer3+18])+get_hex(showpacket[slayer3+19])
        +get_hex(showpacket[slayer3+20])+get_hex(showpacket[slayer3+21])
        +get_hex(showpacket[slayer3+22])+get_hex(showpacket[slayer3+23]));

    fwriteln('  Target Protocol Address:['
        +strr(showpacket[slayer3+24])+'.'+strr(showpacket[slayer3+25])+'.'
        +strr(showpacket[slayer3+26])+'.'+strr(showpacket[slayer3+27])+']');
    end;
    end;


procedure checkipx;{ analysis information in IPX protocol}
begin
   if Layer3 = 'IPX' then
   begin
    fwriteln('  --------IPX--------');
    fwriteln('  Check Sum  : 0x'+get_hex(showpacket[slayer3+0])+get_hex(showpacket
[slayer3+1]));
    fwriteln('  Length     : '+strr(twohex(showpacket[slayer3+2],showpacket
[slayer3+3]))+' (0x'+
    get_hex(showpacket[slayer3+2])+get_hex(showpacket[slayer3+3])+') bytes');
    fwriteln('  Transport Control  : 0x'+get_hex(showpacket[slayer3+4]));
   fwriteln('     '+disbin(showpacket[slayer3+4] div 16) + '  ....    Reserved');
    fwriteln('     ....  '+disbin(showpacket[slayer3+4] mod 16)+'    Hop Count');
    fwriteln('  Packet Type  : '+strr(showpacket[slayer3+5])+'   (0x'+get_hex(showpacket
[slayer3+5])+' '+layer4+')');
```

```
    fwriteln('  Destination Network  : 0x'+get_hex(showpacket[slayer3+6])+get_hex
(showpacket[slayer3+7])
                              +get_hex(showpacket[slayer3+8])+get_hex(showpacket
[slayer3+9]));
    fwriteln('  Destination Host     : 0x'+get_hex(showpacket[slayer3+10])+get_hex
(showpacket[slayer3+11])
                              +get_hex(showpacket[slayer3+12])+get_hex(showpacket
[slayer3+13])
                              +get_hex(showpacket[slayer3+14])+get_hex(showpacket
[slayer3+15]));
    case twohex(showpacket[slayer3+16],showpacket[slayer3+17]) of
      $0451 : st := (' ( File Server )');
      $0452 : st := (' ( Service Advertising )');
      $0453 : st := (' ( Routing Information )');
      $0455 : st := (' ( NetBIOS Packet )');
      $0456 : st := (' ( Diagnostic Packet )');
    else st := ' ';
    end;
    fwriteln( '  Destination Socket   : 0x'+get_hex(showpacket[slayer3+16])+get_hex
(showpacket[slayer3+17])+st);
    fwriteln('  Source  Netwok       : 0x'  +get_hex(showpacket[slayer3+18])+get_hex
(showpacket[slayer3+19])
                              +get_hex(showpacket[slayer3+20])+get_hex(showpacket
[slayer3+21]));
    fwriteln('  Source  Host         : 0x'  +get_hex(showpacket[slayer3+22])+get_hex
(showpacket[slayer3+23])
                              +get_hex(showpacket[slayer3+24])+get_hex(showpacket
[slayer3+25])
                              +get_hex(showpacket[slayer3+26])+get_hex(showpacket
[slayer3+27]));
    case twohex(showpacket[slayer3+28],showpacket[slayer3+29]) of
      $0451 : st := (' ( File Server )');
      $0452 : st := (' ( Service Advertising )');
      $0453 : st := (' ( Routing Information )');
      $0455 : st := (' ( NetBIOS Packet )');
```

```
        $0456 : st := (' ( Diagnostic Packet )');
      else      st := ' ';
      end;
      fwriteln( ' Source Socket      : 0x'+get_hex(showpacket[slayer3+28])+get_hex
(showpacket[slayer3+29])+st);
      end;
end;


procedure checkip;{ analysis information in IP protocol}
begin
   if layer3 = 'IP' then
   begin
      fwriteln(' ---------IP----------');{0x'+get_hex(showpacket[slayer3+7]));}
      fwriteln(' Version  : '+strr(showpacket[slayer3+0] div 16 ));
      fwriteln(' Header Length  : '+strr(showpacket[slayer3+0] mod 16 ));
      fwriteln(' Type Of Service  : '+strr(showpacket[slayer3+1]));
      fwriteln(' Total Packet Length  : '+strr(twohex(showpacket[slayer3+2],showpacket
[slayer3+3])));
      fwriteln(' Identifier  : '+strr(twohex(showpacket[slayer3+4],showpacket[slayer3+5])));
      fwriteln(' Flag    : '+strr(showpacket[slayer3+7] and 07)+
            ' Fragment Offset : '+strr((twohex(showpacket[slayer3+6],showpacket
[slayer3+7])) shr 3 ));
      fwriteln(' Time To live (second)  : '+strr(showpacket[slayer3+8]));
      fwriteln(' Protocol   : '+strr(showpacket[slayer3+9])+' ('+layer4+')');
      fwriteln(' Header Check Sum  : 0x'+get_hex(showpacket[slayer3+10])+get_hex
(showpacket[slayer3+11]));
      fwriteln(' Source Address      : ['+strr(showpacket[slayer3+12])+'.'+strr(showpacket
[slayer3+13])+
                  '.'+strr(showpacket[slayer3+14])+'.'+strr(showpacket[slayer3+15])+']');
      fwriteln(' Destination Address : ['+strr(showpacket[slayer3+16])+'.'+strr(showpacket
[slayer3+17])+
                  '.'+strr(showpacket[slayer3+18])+'.'+strr(showpacket[slayer3+19])+']');
    end;
end;
```

```
procedure checkncp; { analysis information in NCP protocol}
begin
    if layer4 = 'NCP' then
    begin
     fwriteln('  ----------NCP----------');
     if  (showpacket[slayer4+0] = $33) and (showpacket[slayer4+1] = $33) then
     begin
       fwriteln('  Action         :  0x3333 (Response)');
       fwriteln('  Sequence Number : '+strr(showpacket[slayer4+2]));
       fwriteln('  Connection ID   : '+strr(showpacket[slayer4+3]));
       fwriteln('  Process ID      : '+strr(showpacket[slayer4+4]));
       if (showpacket[slayer4+6]) = 00 then st := ('  (OK)')
       else st :=('  (Failed)');
       fwriteln('  Status         : 0x'+get_hex(showpacket[slayer4+6])+st);
     end;

     if  (showpacket[slayer4+0] = $22) and (showpacket[slayer4+1] = $22) then
     begin
      fwriteln('  Action         :  0x2222 (Request)');
      fwriteln('  Sequence Number : '+strr(showpacket[slayer4+2]));
      fwriteln('  Connection ID   : '+strr(showpacket[slayer4+3]));
      fwriteln('  Process ID      : '+strr(showpacket[slayer4+4]));
     case  showpacket[slayer4+6] of
$03 :
   begin
       fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Lock File)');
       fwriteln('  Dirtrory Handle : 0x'+get_hex(showpacket[slayer4+7]));
       fwriteln('  Lock Flags     : 0x'+get_hex(showpacket[slayer4+8]));
       fwriteln('  Timeout        : '+strr(twohex(showpacket[slayer4+10],showpacket
[slayer4+9])));

       st := ('  Filname        : '+chr(96));
       i:= 0;
       while (i < showpacket[slayer4+11]) and (showpacket[slayer4+12+i]<>0) do
```

```
        begin
         st := st + (chrr(showpacket[slayer4+12+i]));
         i := i+1 ;
         end;
        st := st +(chr(39));
        fwriteln(st);
    end;
S04  : begin
    fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Lock All File)');
    fwriteln('  Timeout        : '+strr(twohex(showpacket[slayer4+7],showpacket
[slayer4+8])));
     end;

$07   : begin
    fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Unlock File)');
    fwriteln('  Dirtrory Handle : 0x'+get_hex(showpacket[slayer4+7]));
    st := ('  Filname        : '+chr(39));

    i:= 0;
    while (i < showpacket[slayer4+8]) and (showpacket[slayer4+9+i]<>0) do
    begin
    st := st +(chrr(showpacket[slayer4+9+i]));
    i := i+1 ;
    end;
    st := st+(chr(39));
    fwriteln(st);
   end;
$13  : fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Get Station
ID)');
$14  : fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Read Server
Clock)');
$18  : fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Peocess
Exit)');
$19  : fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Logout)');
$1a  : begin
```

```
    fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Lock Byte
Range)');
    fwriteln('  Handle       : 0x'+get_hex(showpacket[slayer4+8])+get_hex(showpacket
[slayer4+9])+
                    get_hex(showpacket[slayer4+10])+get_hex(showpacket
[slayer4+11]));
    fwriteln('  Position     : 0x'+get_hex(showpacket[slayer4+14])+get_hex(showpacket
[slayer4+15])+
                    get_hex(showpacket[slayer4+16])+get_hex(showpacket
[slayer4+17]));
    fwriteln('  Record Size  : 0x'+get_hex(showpacket[slayer4+18])+get_hex(showpacket
[slayer4+19])+
                    get_hex(showpacket[slayer4+20])+get_hex(showpacket
[slayer4+21]));

    end;
$1E  : fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Unlock Byte
Range)');
$21  : begin
    fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Negotiate Buffer
Size)');
    fwriteln('  Requested Buffer size  : '+strr(twohex(showpacket[slayer4+7],showpacket
[slayer4+8])));
    end;
$3E  : begin
    fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Set Direrory
Handle)');
    fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+7]));
    st := ('  Path        : '+chr(39));

    i:= 0;
    while (i < showpacket[slayer4+8]) and (showpacket[slayer4+9+i]<>0) do
    begin
    st := st+(chrr(showpacket[slayer4+9+i]));
    i := i+1 ;
```

```
    end;
    fwriteln(st+(chr(39)));
    end;
$3F  : begin
    fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Search)');
    fwriteln('  Databyte  : 0x'+get_hex(showpacket[slayer4+7]));
    fwriteln('  Unknown Word  : 0x'+get_hex(showpacket[slayer4+8])+get_hex
(showpacket[slayer4+9]));
    if (twohex(showpacket[slayer4+11],showpacket[slayer4+10]) = $ffff) then st := ('
(Find First)')
    else st := ('  (Find Next '+strr(twohex(showpacket[slayer4+11],showpacket
[slayer4+10]))+')');
    fwriteln  ('  Entry Number  :  '+strr(twohex(showpacket[slayer4+11],showpacket
[slayer4+10]))+st);
    fwriteln('  Netware Attribute  : 0x'+get_hex(showpacket[slayer4+12]));
    netwareatt(showpacket[slayer4+12]) ;
    st := ('  Filename  : '+chr(39));
    i:= 0;
    while (i < showpacket[slayer4+13]) and (showpacket[slayer4+14+i]<>0) do
    begin
    st := st + (chrr(showpacket[slayer4+14+i]));
    i := i+1 ;
    end;
    fwriteln(st+chr(39));
    end;

$40  : begin
    fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Find Unique)');
    fwriteln('  Entry Number  : '+strr(twohex(showpacket[slayer4+8],showpacket
[slayer4+7])));
    fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+9])+get_hex
(showpacket[slayer4+10]));
    fwriteln('  Dos Attribute  : 0x'+get_hex(showpacket[slayer4+11]));
    dosatt(showpacket[slayer4+11]);
    st := ('  Pathname  : '+chr(39));
```

```
    i:= 0;
    while (i < showpacket[slayer4+12]) and (showpacket[slayer4+13+i]<>0) do
    begin
    st := st+(chrr(showpacket[slayer4+13+i]));
    i := i+1 ;
    end;
    fwriteln(st+chr(39));
    end;

$41  : begin
    fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Open FCB)');
    fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+7]));
    fwriteln('  Dos Attribute  : 0x'+get_hex(showpacket[slayer4+8]));
    dosatt(showpacket[slayer4+8]);
    st := ('  Filename  : '+chr(39));
    i:= 0;
    while (i < showpacket[slayer4+9]) and (showpacket[slayer4+10+i]<>0) do
    begin
    st := st+(chrr(showpacket[slayer4+10+i]));
    i := i+1 ;
    end;
    fwriteln(st+chr(39));
    end;
$42  : begin
    fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Close)');
    fwriteln('  Handle  : 0x'+get_hex(showpacket[slayer4+8])+get_hex(showpacket
[slayer4+9])+
                    get_hex(showpacket[slayer4+10])+get_hex(showpacket
[slayer4+11]));
    end;

$43  : begin
    fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Create)');
    fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+7]));
    st := ('  Filename        : '+chr(39));
```

```
        i:= 0;
        while (i < showpacket[slayer4+8]) and (showpacket[slayer4+9+i]<>0) do
        begin
        st := st+(chrr(showpacket[slayer4+9+i]));
        i := i+1 ;
        end;
        fwriteln(st+chr(39));
        end;

$44  : begin
        fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Delete)');
        fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+7]));
        fwriteln('  Data Byte  : '+strr(showpacket[slayer4+8])+'  Byte');
        st := ('  Filename        : '+chr(39));

        i:= 0;
        while (i < showpacket[slayer4+9]) and (showpacket[slayer4+10+i]<>0) do
        begin
        st := st+(chrr(showpacket[slayer4+10+i]));
        i := i+1 ;
        end;
        fwriteln(st+chr(39));
        end;

$45  : begin
        fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Rename)');
        fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+7]));
        fwriteln('  Data Byte  : '+strr(showpacket[slayer4+8])+'  Byte');
        st := ('  SourceName        : '+chr(39));

        i:= 0;
        while (i < showpacket[slayer4+9]) do
        begin
        st := st+(chrr(showpacket[slayer4+10+i]));
```

```
        i := i+1 ;
        end;
        fwriteln(st+chr(39));

        st := ('  TargetName        : '+chr(39));

        j:= 0;
        while (j < showpacket[slayer4+9+2+i]) and (showpacket[slayer4+10+i+2+j]<>0) do
        begin
        st := st+(chrr(showpacket[slayer4+10+i+2+j]));
        j := j+1 ;
        end;
        fwriteln(st+chr(39));
        end;

$46  : begin
        fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  Set File
Information)');
        fwriteln('  Netware Attribute  : 0x'+get_hex(showpacket[slayer4+7]));
        netwareatt(showpacket[slayer4+7]);
        fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+8]));
        fwriteln('  Dos Attribute  : 0x'+get_hex(showpacket[slayer4+9]));
        dosatt(showpacket[slayer4+9]);
        st := ('  Filename         : '+chr(39));
        i:= 0;
        while (i < showpacket[slayer4+10]) and (showpacket[slayer4+11+i]<>0) do
        begin
        st := st+(chrr(showpacket[slayer4+11+i]));
        i := i+1 ;
        end;
        fwriteln(st + chr(39));
        end;
$47  : begin
        fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Get File Size)');
```

```
    fwriteln('  Handle  : 0x'+get_hex(showpacket[slayer4+8])+get_hex(showpacket
[slayer4+9])+
                    get_hex(showpacket[slayer4+10])+get_hex(showpacket
[slayer4+11]));
  end;

$48  : begin
    fwriteln('  Command       : 0x'+get_hex(showpacket[slayer4+6])+'  (Read)');
    fwriteln('  Handle      : 0x'+get_hex(showpacket[slayer4+8])+get_hex(showpacket
[slayer4+9])+
                    get_hex(showpacket[slayer4+10])+get_hex(showpacket
[slayer4+11]));
    fwriteln('  Position    : 0x'+get_hex(showpacket[slayer4+14])+get_hex(showpacket
[slayer4+15])+
                    get_hex(showpacket[slayer4+16])+get_hex(showpacket
[slayer4+17]));
    fwriteln('  Byte to Transfer  : 0x'+get_hex(showpacket[slayer4+18])+get_hex
(showpacket[slayer4+19]));
    end;
$49  : begin
    fwriteln('  Command       : 0x'+get_hex(showpacket[slayer4+6])+'  (Write)');
    fwriteln('  Handle      : 0x'+get_hex(showpacket[slayer4+8])+get_hex(showpacket
[slayer4+9])+
                    get_hex(showpacket[slayer4+10])+get_hex(showpacket
[slayer4+11]));
    fwriteln('  Position    : 0x'+get_hex(showpacket[slayer4+14])+get_hex(showpacket
[slayer4+15])+
                    get_hex(showpacket[slayer4+16])+get_hex(showpacket
[slayer4+17]));
    fwriteln('  Byte to Transfer  : 0x'+get_hex(showpacket[slayer4+18])+get_hex
(showpacket[slayer4+19]));
    end;
$4b  : begin
    fwriteln('  Command       : 0x'+get_hex(showpacket[slayer4+6])+'  (Set File Time)');
```

```
      fwriteln('  Handle  : 0x'+get_hex(showpacket[slayer4+8])+get_hex(showpacket
[slayer4+9])+
                        get_hex(showpacket[slayer4+10])+get_hex(showpacket
[slayer4+11]));
      end;
$4c   : begin
      fwriteln('  Command         : 0x'+get_hex(showpacket[slayer4+6])+'  (Open)');
      fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+7]));
      fwriteln('  Dos Attribute  : '+strr(showpacket[slayer4+8]));
      dosatt(showpacket[slayer4+8]);
      fwriteln('  Desired Access  : '+strr(showpacket[slayer4+9]));
      desired(showpacket[slayer4+9]);
      st := ('  Filename        : '+chr(39));
      i:= 0;
      while (i < showpacket[slayer4+10]) and (showpacket[slayer4+11+i]<>0) do
      begin
      st := st+(chrr(showpacket[slayer4+11+i]));
      i := i+1 ;
      end;
      fwriteln(st + chr(39));
      end;
$4d  : begin
      fwriteln('  Command         : 0x'+get_hex(showpacket[slayer4+6])+'  (Create New
File)');
      fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+7]));
      fwriteln('  Dos Attribute  : '+strr(showpacket[slayer4+8]));
      dosatt(showpacket[slayer4+8]);
      st := ('  Filename        : '+chr(39));
      i:= 0;
      while (i < showpacket[slayer4+9]) and (showpacket[slayer4+10+i]<>0) do
      begin
      st := st +(chrr(showpacket[slayer4+10+i]));
      i := i+1 ;
      end;
      fwriteln(st+chr(39));
```

```
    end;

$17 : begin
    fwriteln('  Command        : 0x'+get_hex(showpacket[slayer4+6])+'  (Access
Control)');
    fwriteln('------ACCESS CONTROL PROTOCOL------');
    case  showpacket[slayer4+9] of
    $00 : begin
        fwriteln('  Function Code  : 0x'+get_hex(showpacket[slayer4+9])+'  (Login)');
        st := ('  Username       : '+chr(39));

        i:= 0;
        while (i < showpacket[slayer4+10]) do
        begin
        st := st+(chr(showpacket[slayer4+11+i]));
        i := i+1 ;
        end;
        fwriteln(st+chr(39));
        st := ('  Password       : '+chr(39));
        j:= 0;
        while (j < showpacket[slayer4+11+i]) and (showpacket[slayer4+12+i+j]<>0) do
        begin
        st := st+(chr(showpacket[slayer4+12+i+j]));
        j := j+1 ;
        end;
        fwriteln(st+chr(39));
        end;
    $11 :  fwriteln('  Function Code  : 0x'+get_hex(showpacket[slayer4+9])+'  (Check
Server Version)');
    $16 :  begin
        fwriteln('  Function Code  : 0x'+get_hex(showpacket[slayer4+9])+'  (Get
Connection Information)');
        fwriteln('  Connection ID  : 0x'+get_hex(showpacket[slayer4+10]));
        end;
    $35 :  begin
```

```
        fwriteln('  Function Code   : 0x'+get_hex(showpacket[slayer4+9])+'  (Get Trustee
Mapping)');
        fwriteln('  Object Type     : 0x'+get_hex(showpacket[slayer4+10])+get_hex
(showpacket[slayer4+11]));

        st := ('  Username       : '+chr(39));
        i:= 0;
        while (i < showpacket[slayer4+12]) and (showpacket[slayer4+13+i]<>0) do
        begin
        st := st +(chrr(showpacket[slayer4+13+i]));
        i := i+1 ;
        end;
        fwriteln(st+chr(39));
        end;
    $36 : begin
        fwriteln('  Function Code   : 0x'+get_hex(showpacket[slayer4+9])+'  (Mapp
Trustee To User)');
        fwriteln('  Object ID       : 0x'+get_hex(showpacket[slayer4+10])+get_hex
(showpacket[slayer4+11])
                          +get_hex(showpacket[slayer4+12])+get_hex(showpacket
[slayer4+13]));
        end;

    $37 : begin
        fwriteln('  Function Code   : 0x'+get_hex(showpacket[slayer4+9])+'  (Search
Bindery For Objec)');
        fwriteln('  Search Key      : 0x'+get_hex(showpacket[slayer4+10])+get_hex
(showpacket[slayer4+11])
                          +get_hex(showpacket[slayer4+12])+get_hex(showpacket
[slayer4+13]));
        fwriteln('  Object Type     : 0x'+get_hex(showpacket[slayer4+14])+get_hex
(showpacket[slayer4+15]));
        st := ('  Name           : '+chr(39));
        i:= 0;
        while (i < showpacket[slayer4+16]) and (showpacket[slayer4+16+i]<>0) do
```

```
            begin
            st := st+(chrr(showpacket[slayer4+17+i]));
            i := i+1 ;
            end;
            fwriteln(st+chr(39));
            end;

      $3f : begin
            fwriteln('  Function Code   : 0x'+get_hex(showpacket[slayer4+9])+'  (Check
Object Password)');
            fwriteln('  Object Type     : 0x'+get_hex(showpacket[slayer4+10])+get_hex
(showpacket[slayer4+11]));

            st := ('  Username        : '+chr(39));

            i:= 0;
            while (i < showpacket[slayer4+12]) do
            begin
            st := st+(chrr(showpacket[slayer4+13+i]));
            i := i+1 ;
            end;
            fwriteln(st+chr(39));

            st := ('  Password        : '+chr(39));
            j:= 0;
            while (j < showpacket[slayer4+13+i]) and (showpacket[slayer4+14+i+j]<>0) do
            begin
            st := st+(chrr(showpacket[slayer4+14+i+j]));
            j := j+1 ;
            end;
            fwriteln(st+chr(39));
            end;
      $3D :  begin
            fwriteln('  Function Code   : 0x'+get_hex(showpacket[slayer4+9])+'  (Get Trustee
Mapping)');
```

```
            fwriteln('  Object Type     : 0x'+get_hex(showpacket[slayer4+10])+get_hex
(showpacket[slayer4+11]));

            st := ('  Object Name     : '+chr(39));
            i:= 0;


            while (i < showpacket[slayer4+12]) and (showpacket[slayer4+13+i]<>0) do
            begin
            st := st+(chrr(showpacket[slayer4+13+i]));
            i := i+1 ;
            end;
            fwriteln(st+chr(39));
            fwriteln('  Segment         : 0x'+get_hex(showpacket[slayer4+13+i])+get_hex
(showpacket[slayer4+14+i]));

            st := ('  Property Name   : '+chr(39));
            j:= 0;
            while (j < showpacket[slayer4+15+i]) and (showpacket[slayer4+16+i+j]<>0) do
            begin
            st := st+(chrr(showpacket[slayer4+16+i+j]));
            j := j+1 ;
            end;
            fwriteln(st+chr(39));
            end;
      else fwriteln(  'Undecode Acess Control');
      end;
      end;
$16  : begin
      fwriteln('  Command         : 0x'+get_hex(showpacket[slayer4+6])+'   (Directory
Control)');
      fwriteln('------DIRECTORY CONTROL PROTOCOL-------');
      case   showpacket[slayer4+9] of
      $00 :  begin
            fwriteln('  Frame Length  : 0x'+get_hex(showpacket[slayer4+8]));
```

```
        fwriteln('  Operation  : 0x'+get_hex(showpacket[slayer4+9])+' (Set Current
Directory)');
        fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+10]));
        fwriteln('  Data Byte         : 0x'+get_hex(showpacket[slayer4+11]));
        st := ('  Path           : '+chr(39));
        i:= 0;
        while (i < showpacket[slayer4+12]) and (showpacket[slayer4+13+i]<>0) do
        begin
        st := st+(chrr(showpacket[slayer4+13+i]));
        i := i+1 ;
        end;
        fwriteln(st+chr(39));
        end;
    $01 :  begin
        fwriteln('  Data Byte  : 0x'+get_hex(showpacket[slayer4+8]));
        fwriteln('  Operation  : 0x'+get_hex(showpacket[slayer4+9])+' (Get Current
Directory)');
        fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+10]));
        fwriteln('  Data Byte         : 0x'+get_hex(showpacket[slayer4+11]));
        end;
    $12 :  begin
        fwriteln('  Data Byte  : 0x'+get_hex(showpacket[slayer4+8]));
        fwriteln('  Operation  : 0x'+get_hex(showpacket[slayer4+9])+' (Create Permanent
Directory Handle)');
        fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+10]));
        fwriteln('  Drive name        : '+chrr(showpacket[slayer4+11])+':');
        st := ('  Path           : '+chr(39));
        i:= 0;
        while (i < showpacket[slayer4+12]) and (showpacket[slayer4+13+i]<>0) do
        begin
        st := st+(chrr(showpacket[slayer4+13+i]));
        i := i+1 ;
        end;
        fwriteln(st+chr(39));
        end;
```

```
        $13 :  begin
            fwriteln('  Data Byte  : 0x'+get_hex(showpacket[slayer4+8]));
            fwriteln('  Operation  : 0x'+get_hex(showpacket[slayer4+9])+' (Create  Directory
Handle)');
            fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+10]));
            fwriteln('  Drive name        : '+chrr(showpacket[slayer4+11])+':');
            st := ('  Path           : '+chr(39));
            i:= 0;
            while (i < showpacket[slayer4+12]) and (showpacket[slayer4+13+i]<>0) do
            begin
            st := st+(chrr(showpacket[slayer4+13+i]));
            i := i+1 ;
            end;
            fwriteln(st+chr(39));
            end;

        $14 :  begin
            fwriteln('  Data Byte  : 0x'+get_hex(showpacket[slayer4+8]));
            fwriteln('  Operation  : 0x'+get_hex(showpacket[slayer4+9])+' (Delete Directory
Handle)');
            fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+10]));
            end;

        $15 :  begin
            fwriteln('  Data Byte  : 0x'+get_hex(showpacket[slayer4+8]));
            fwriteln('  Operation  : 0x'+get_hex(showpacket[slayer4+9])+' (Get Volume
Information)');
            fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+10]));
            fwriteln('  Data Byte         : 0x'+get_hex(showpacket[slayer4+11]));
            end;
        else  begin
            fwriteln('  Data Byte  : 0x'+get_hex(showpacket[slayer4+8]));
            fwriteln('  Directory Control : 0x'+get_hex(showpacket[slayer4+9]));
            fwriteln('  Directory Handle  : 0x'+get_hex(showpacket[slayer4+10]));
```

```
            fwriteln('  Data Byte        : 0x'+get_hex(showpacket[slayer4+11]));
            end;
        end;
        end;
    end;
    end;
    end;
    end;

    procedure checktcp;{ analysis information in TCP protocol}
    begin
      if layer4 = 'TCP' then .
      begin
        fwriteln('  --------TCP--------');
        case twohex(showpacket[slayer4+0],showpacket[slayer4+1]) of
            05    : st := '  (Remote Job Entry)';
            07    : st := '  (Echo Service)';
            09    : st := '  (Discard)';
            11    : st := '  (Active Users)';
            13    : st := '  (Day Time)';
            15    : st := '  (NetStat)';
            17    : st := '  (Quote of The Day)';
            20    : st := '  (FTP Data)';
            21    : st := '  (FTP Control)';
            23    : st := '  (TELNET)';
            25    : st := '  (SMTP)';
            37    : st := '  (Time)';
            53    : st := '  (Name Server)';
        else     st := '  (UNKNOWN)';
        end;
        fwriteln('  Source Port  = 0x'+get_hex(showpacket[slayer4+0])+get_hex(showpacket
    [slayer4+1])+st);
        case twohex(showpacket[slayer4+2],showpacket[slayer4+3]) of
            05    : st := '  (Remote Job Entry)';
            07    : st := '  (Echo Service)';
```

```
            09    : st := ' (Discard)';
            11    : st := ' (Active Users)';
            13    : st := ' (Day Time)';
            15    : st := ' (NetStat)';
            17    : st := ' (Quote of The Day)';
            20    : st := ' (FTP Data)';
            21    : st := ' (FTP Control)';
            23    : st := ' (TELNET)';
            25    : st := ' (SMTP)';
            37    : st := ' (Time)';
            53    : st := ' (Name Server)';
            else    st := ' (UNKNOWN)';
        end;
        fwriteln(' Destination Port  = 0x'+get_hex(showpacket[slayer4+2])+get_hex
    (showpacket[slayer4+3])+st);
        fwriteln(' Sequence Number  = 0x' +get_hex(showpacket[slayer4+4])+get_hex
    (showpacket[slayer4+5])
                                +get_hex(showpacket[slayer4+6])+get_hex(showpacket
    [slayer4+7]));
        fwriteln(' Acknowledgement Number  = 0x'
    +get_hex(showpacket[slayer4+8])+get_hex(showpacket[slayer4+9])
                                +get_hex(showpacket[slayer4+10])+get_hex(showpacket
    [slayer4+11]));
        fwriteln(' Header Length  = '+strr(showpacket[slayer4+12]));
        fwriteln(' Code Bits  = '+strr(showpacket[slayer4+13]));
        fwriteln(' Windows  = 0x'+get_hex(showpacket[slayer4+14])+get_hex(showpacket
    [slayer4+15]));
        fwriteln(' Check Sum  = 0x'+get_hex(showpacket[slayer4+16])+get_hex
    (showpacket[slayer4+17]));
        fwriteln(' Urgent Pointer  = 0x'+get_hex(showpacket[slayer4+18])+get_hex
    (showpacket[slayer4+19]));
        end;
    end;
```

```
procedure checkudp;{ analysis information in UDP protocol}
begin
    if layer4 = 'UDP' then
    begin
     fwriteln('  --------UDP---------');
     case twohex(showpacket[slayer4+0],showpacket[slayer4+1]) of
        05    : st := '  (Remote Job Entry)';
        07    : st := '  (Echo Service)';
        09    : st := '  (Discard)';
        11    : st := '  (Active Users)';
        13    : st := '  (Day Time)';
        15    : st := '  (NetStat)';
        17    : st := '  (Quote of The Day)';
        20    : st := '  (FTP Data)';
        21    : st := '  (FTP Control)';
        23    : st := '  (TELNET)';
        25    : st := '  (SMTP)';
        37    : st := '  (Time)';
        53    : st := '  (Name Server)';
        else    st := '  ';{(UNKNOWN)';}
    end;
    fwriteln('  Source Port  = 0x'+get_hex(showpacket[slayer4+0])+get_hex(showpacket
[slayer4+1])+st);
    case twohex(showpacket[slayer4+2],showpacket[slayer4+3]) of
        05    : st := '  (Remote Job Entry)';
        07    : st := '  (Echo Service)';
        09    : st := '  (Discard)';
        11    : st := '  (Active Users)';
        13    : st := '  (Day Time)';
        15    : st := '  (NetStat)';
        17    : st := '  (Quote of The Day)';
        20    : st := '  (FTP Data)';
        21    : st := '  (FTP Control)';
        23    : st := '  (TELNET)';
        25    : st := '  (SMTP)';
```

```
        37    : st := ' (Time)';
        53    : st := ' (Name Server)';
        else    st := ' ';{(UNKNOWN)';}
    end;
    fwriteln(' Destination Port  = 0x'+get_hex(showpacket[slayer4+2])+get_hex
(showpacket[slayer4+3])+st);
    fwriteln(' Length  =  0x'+get_hex(showpacket[slayer4+4])+get_hex(showpacket
[slayer4+5]));
    fwriteln(' Check Sum  =  0x'+get_hex(showpacket[slayer4+6])+get_hex(showpacket
[slayer4+7]));
    end;
end;

procedure checkicmp; { analysis information in ICMP protocol}
begin
   if layer4 = 'ICMP' then
   begin
    fwriteln(' --------ICMP--------');
    case showpacket[slayer4+0] of
       0 :
       begin
        fwriteln(' Type  =  '+strr(showpacket[slayer4+0])+' (Echo Message)');
        fwriteln(' Code  =  '+strr(showpacket[slayer4+1]));
        fwriteln(' Check sum  =  0x'+get_hex(showpacket[slayer4+2])+get_hex
(showpacket[slayer4+3]));
        fwriteln(' Identifier  =  0x'+get_hex(showpacket[slayer4+4])+get_hex
(showpacket[slayer4+5]));
        fwriteln(' Sequence Number  =  0x'+get_hex(showpacket[slayer4+6])+get_hex
(showpacket[slayer4+7]));
        end;

       8 :
       begin
        fwriteln(' Type  =  '+strr(showpacket[slayer4+0])+' (Echo Reply Message)');
        fwriteln(' Code  =  '+strr(showpacket[slayer4+1]));
```

```
        fwriteln(' Check sum = 0x'+get_hex(showpacket[slayer4+2])+get_hex
(showpacket[slayer4+3]));
        fwriteln(' Identifier = 0x'+get_hex(showpacket[slayer4+4])+get_hex
(showpacket[slayer4+5]));
        fwriteln(' Sequence Number = 0x'+get_hex(showpacket[slayer4+6])+get_hex
(showpacket[slayer4+7]));
        end;

      3 :
      begin
        fwriteln(' Type = '+strr(showpacket[slayer4+0])+' (Destination Unreachable
Message)');
         case showpacket[slayer4+1] of
          0 : st := ' (Network Unreachable)';
          1 : st := ' (Host Unreachable)';
          2 : st := ' (Protocol Unreachable)';
          3 : st := ' (Port Unreachable)';
          4 : st := ' (Fragmentation needed and DF set)';
          5 : st := ' (Source Route Failed)';
         else st := ' ';
         end;

        fwriteln(' Code = '+strr(showpacket[slayer4+1])+st);
        fwriteln(' Check sum = 0x'+get_hex(showpacket[slayer4+2])+get_hex
(showpacket[slayer4+3]));
        fwriteln(' Unused = 0x'+get_hex(showpacket[slayer4+4])+get_hex
(showpacket[slayer4+5])
              +get_hex(showpacket[slayer4+6])+get_hex(showpacket[slayer4+7]));
      end;

      11 :
      begin
        fwriteln(' Type = '+strr(showpacket[slayer4+0])+' (Time Exceeded Message)');
        case showpacket[slayer4+1] of
          0 : st := ' (Time To Live Exceeded In Transit)';
```

```
            1 : st := '  (Fragment Reassembly Time Exceeded)';
            else st := ' ';
          end;
          fwriteln('  Code  =  '+strr(showpacket[slayer4+1])+st);
          fwriteln('  Check sum  =  0x'+get_hex(showpacket[slayer4+2])+get_hex
(showpacket[slayer4+3]));
          fwriteln('  Unused  =  0x'+get_hex(showpacket[slayer4+4])+get_hex
(showpacket[slayer4+5])
                  +get_hex(showpacket[slayer4+6])+get_hex(showpacket[slayer4+7]));
          end;

        4 :
        begin
          fwriteln('  Type  =  '+strr(showpacket[slayer4+0])+' (Source Quench Message)');
          fwriteln('  Code  =  '+strr(showpacket[slayer4+1]));
          fwriteln('  Check sum  =  0x'+get_hex(showpacket[slayer4+2])+get_hex
(showpacket[slayer4+3]));
          fwriteln('  Unused  =  0x'+get_hex(showpacket[slayer4+4])+get_hex
(showpacket[slayer4+5])
                  +get_hex(showpacket[slayer4+6])+get_hex(showpacket[slayer4+7]));
        end;

        15 :
        begin
          fwriteln('  Type  =  '+strr(showpacket[slayer4+0])+' (Information Request
Message)');
          fwriteln('  Code  =  '+strr(showpacket[slayer4+1]));
          fwriteln('  Check sum  =  0x'+get_hex(showpacket[slayer4+2])+get_hex
(showpacket[slayer4+3]));
          fwriteln('  Identifier  =  0x'+get_hex(showpacket[slayer4+4])+get_hex
(showpacket[slayer4+5]));
          fwriteln('  Sequence Number  =  0x'+get_hex(showpacket[slayer4+6])+get_hex
(showpacket[slayer4+7]));
        end;
```

```
        16 :
        begin
         fwriteln('  Type  =  '+strr(showpacket[slayer4+0])+' (Information Reply
Message)');
         fwriteln('  Code  =  '+strr(showpacket[slayer4+1]));
         fwriteln('  Check sum  =  0x'+get_hex(showpacket[slayer4+2])+get_hex
(snowpacket[slayer4+3]));
         fwriteln('  Identifier  =  0x'+get_hex(showpacket[slayer4+4])+get_hex
(showpacket[slayer4+5]));
         fwriteln('  Sequence Number  =  0x'+get_hex(showpacket[slayer4+6])+get_hex
(showpacket[slayer4+7]));
        end;



        5 :
        begin
         fwriteln('  Type  =  '+strr(showpacket[slayer4+0])+' (Redirect Message)');
         case showpacket[slayer4+1] of
          0 : st := '  (Redirect Datagrams For the Network)';
          1 : st := '  (Redirect Datagrams For the Host)';
          2 : st := '  (Redirect Datagrams For The Type Of Service And Network)';
          3 : st := '  (Redirect Datagrams For The Type Of Service And Host)';
         else st := ' ';
         end;
         fwriteln('  Code  =  '+strr(showpacket[slayer4+1])+st);
         fwriteln('  Check sum  =  0x'+get_hex(showpacket[slayer4+2])+get_hex
(showpacket[slayer4+3]));
         fwriteln('  Gateway Internet Address  =  ['+strr(showpacket[slayer4+4])+'.'+strr
(showpacket[slayer4+5])+
                          '.'+strr(showpacket[slayer4+6])+'.'+strr(showpacket
[slayer4+7])+']');
        end;

        12 :
        begin
```

```
        fwriteln('  Type  =  '+strr(showpacket[slayer4+0])+' (Parameter Problem
Message)');
        case showpacket[slayer4+1] of
         0 : st := '  (Pointer Indirect The Error)';
         else st := ' ';
        end;
        fwriteln('  Code  =  '+strr(showpacket[slayer4+1])+st);
        fwriteln('  Check sum  =  0x'+get_hex(showpacket[slayer4+2])+get_hex
(showpacket[slayer4+3]));
        fwriteln('  Pointer  =  '+strr(showpacket[slayer4+4]));
        fwriteln('  Unused  =  0x'+get_hex(showpacket[slayer4+5])+get_hex
(showpacket[slayer4+6])
                +get_hex(showpacket[slayer4+7]));
      end;

      13 :
      begin
        fwriteln('  Type  =  '+strr(showpacket[slayer4+0])+' (Timestamp Message)');
        fwriteln('  Code  =  '+strr(showpacket[slayer4+1]));
        fwriteln('  Check sum  =  0x'+get_hex(showpacket[slayer4+2])+get_hex
(showpacket[slayer4+3]));
        fwriteln('  Original Timestamp  =  0x'+get_hex(showpacket[slayer4+4])+get_hex
(showpacket[slayer4+5])
                +get_hex(showpacket[slayer4+6])+get_hex(showpacket[slayer4+7]));
        fwriteln('  Receive Timestamp  =  0x'+get_hex(showpacket[slayer4+8])+get_hex
(showpacket[slayer4+9])
                +get_hex(showpacket[slayer4+10])+get_hex(showpacket[slayer4+11]));
        fwriteln('  Transmit Timestamp  =
0x'+get_hex(showpacket[slayer4+12])+get_hex(showpacket[slayer4+13])
                +get_hex(showpacket[slayer4+14])+get_hex(showpacket[slayer4+15]));
      end;

      14 :
      begin
```

```
            fwriteln(' Type = '+strr(showpacket[slayer4+0])+' (Timestamp Reply
Message)');
            fwriteln(' Code = '+strr(showpacket[slayer4+1]));
            fwriteln(' Check sum = 0x'+get_hex(showpacket[slayer4+2])+get_hex
(showpacket[slayer4+3]));
            fwriteln(' Original Timestamp = 0x'+get_hex(showpacket[slayer4+4])+get_hex
(showpacket[slayer4+5])
                    +get_hex(showpacket[slayer4+6])+get_hex(showpacket[slayer4+7]));
            fwriteln(' Receive Timestamp = 0x'+get_hex(showpacket[slayer4+8])+get_hex
(showpacket[slayer4+9])
                    +get_hex(showpacket[slayer4+10])+get_hex(showpacket[slayer4+11]));
            fwriteln(' Transmit Timestamp =
0x'+get_hex(showpacket[slayer4+12])+get_hex(showpacket[slayer4+13])
                    +get_hex(showpacket[slayer4+14])+get_hex(showpacket[slayer4+15]));
          end;
       end;
     end;
end;

procedure showraw;{ Show raw data }
begin
     setwinattr($1f);
     SetHeadAttr($1f);
     setBoxattr($1f);
     setcharattr($1f);
     setwinheader(' DATA ETHERNET ');
     setboxstyle(2);
     h :=  (long[longp1] div 16);
     if h > 18 then h := 18 ;
     windowopen(2,4,78,h+7);
     k := 0;
     g := 0;
     key := dn_key;
     repeat
      if (key = up_key) or (key = dn_key) then
```

```
          begin
          for j := 1 to h+1 do
           begin
             gotoxy(1,j);
             write(get_hex(g div $100),get_hex(g mod $100));
             for i := 2 to 17 do
              begin
              gotoxy(i*3,j);
              if i > 9 then gotoxy((i*3)+2,j);
              write(get_hex(showpacket[g]),'');
              gotoxy(i+55,j);
              if i > 9 then gotoxy((i+57),j);
              if showpacket[g] >32 then  write(chr(showpacket[g]))else write('.');
              g := g+1
              end;
           end;
          gotoxy(1,h+7);
          write('  ');
          SetAttr($35);
          write('  Press Key ',char(24),char(25),' to Up and Down View This Page   Press
Key Enter  to EXIT ');
          SetAttr($1f);
         end;

         key := readkey;
         if (key =dn_key) then
         if (k+304 < long[longp1]) then
         begin
          k := k+16 ;
          g := k ;
         end
        else
        key := #0;

        if (key =up_key) then
```

```
        if (k-16 >=0) then
        begin
         k := k-16 ;
         g := k;
        end
       else
       kev := #0;
       until (key = Return_key) or (key = esc_key);
       key := null_key;
       windowclose;
end;



procedure showtime(hour,min,sec : word);{ Show time }
var times,h,m,s : string;
begin
      str(hour,h);
      if hour < 10 then
      h := concat('0',h);
      str(min,m);
      if min  < 10 then
      m := concat('0',m);
      str(sec,s);
      if sec < 10 then
      s := concat('0',s);
      write(h,':',m,':',s);
end;



procedure Init;
{ Check for presence of packet driver, get and store hardware address,
  print info.}
var x : byte;
    driver_info : drvinfo;
```

```
   access     : eth_drv_access  ;
begin
    TextColor( Yellow );
    Write( 'ETHERNET ' );
    TextColor( LightRed );
    x := Driver_present;
    if (x <> 0) then begin
     Writeln( 'Packet driver located at interrupt 0x',Get_Hex( x) );
     driver_Intr := x;
     Get_Driver_Info( 0, driver_info );
     Write_Driver_Info( driver_info );
     Get_Pkt_hwaddr( 0, my_address );
     Writeln('Card address is ',String_hwaddr(my_address));
     Driver_loaded := true;
    end else begin
     Writeln( 'Packet driver not found.' );
     Driver_loaded := false;
     readkey  ;
     exit;
     end;
      with  access  do begin
      class := driver_info.e_class;
      etype := $ffff;              { ...of interface }
      number := 0   ;              { ...of the interface }
      packet_type := @typeField;     { two-byte type spec. }
      packet_type_length := 0   ;    { ...of above data. }
      receiver := @pktReceiver;          { ...to ether_receiver routine }
      end;
      Access_Type(access) ;
      SetRCVmode( 0,6);
end;


procedure savebuffer; { Save buffer to disk }
var filvar : text;
```

```
    namef : string;
    i     : integer;
begin
      checkfile(namef);
      Assign(filvar,namef) ;
      rewrite(filvar);
      writeln(filvar,longp);
      for i := 1 to longp do
      writeln(filvar,long[i]);
      writeln(filvar,datafp);
      for i := 1 to datafp do
        begin
        write(filvar,Get_hex(dataf[i]));
        if (i mod 50) = 0 then writeln(filvar);
        end;
      close(filvar);
      windowclose;
      save_data := false;
end;

procedure savedata; { check siveing data}
var k     : byte;
    key  : char;
begin
      SetwinHeader('SAVE DATA');
      SetWinAttr($1f);
      SetBoxAttr($1f);
      SetCharAttr($1f);
      Setboxstyle(double);
      SetHeadAttr($1f);
      Windowopen(23,10,57,14);
      writeln(' The data has not been saved yet');
      write  ('       Save it now ?');
      k := 1;
      repeat
```

```
setAttr($7e);
gotoxy(10,3);
if k = 1 then
begin
setAttr($40);
write('<YES>');
end
else
write(' YES ');

setAttr($7e);
gotoxy(20,3);
if k = 2 then
begin
setAttr($40);
write(' <NO> ');
end
else
write('  NO  ');


key := readkey;
    case (Key) of
      lt_key   :   if k = 1 then k := 2 else k := k-1;
      rt_Key   :   if k = 2 then k := 1 else k := k+1;
      return_key : if k = 1 then  savebuffer;
      end;
until (key = return_key)or(key = esc_key) ;
windowclose;
 end;


procedure loadbuffer; { Load file form disk to buffer }
var filvar : text;
```

```
    namef,p : string;
    i    : integer;
    s1,s2 : char;

begin
      if (save_data) and (longp >= 1)  then savedata;
      ReadDir('*.nmt',p,Namef);
      if namef <> #27 then
      begin
       namef := concat(p,namef);
       Assign(filvar,namef) ;
       reset(filvar);
       readln(filvar,longp);
       for i := 1 to longp do
       readln(filvar,long[i]);
       readln(filvar,datafp);
       for i := 1 to datafp do
        begin
        read(filvar,s1);
        read(filvar,s2);
        dataf[i] := (hex_byte(s1)*16)+hex_byte(s2);
        if (i mod 50) = 0 then readln(filvar);
       end;
       close(filvar);
       setwinHeader('LOAD');
       SetWinAttr($1f);
       SetBoxAttr($1f);
       SetCharAttr($1f);
       Setboxstyle(double);
       SetHeadAttr($1f);
       Windowopen(25,10,55,13);
       writeln('         LOAD DATA  ');
       Delay(500);
       windowclose;
      end;
```

```
end;


procedure showscale_v;{ display scale of quantity packet}
var i,j : integer;
   maxt : integer;
begin
      maxt := maxtime;
      for j := 1 to 10 do
      begin
        gotoxy(1,j);
        write('                                                    ');
      end;
      k_scale := 10;
      for i := 1 to  9 do
      begin
        if value_array[maxt]  > 100 then
        k_scale := 100;
        if maxt - 1 >= 0 then maxt := maxt-1;
      end;
      gotoxy(1,1);
      textcolor(red);
      writeln('    QUANTITY (PACKET)');
      textcolor(white);
      writeln('     ย');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
```

```
      write('   0
ภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤ
ฤด');
    textcolor(red);
    writeln('TIME');
    textcolor(green);
    j := 10;
    for i := 1 to 10 do
    begin
     gotoxy(1,2+i);
     write(j*k_scale);
     j := j-1
    end;
end;

procedure showgraph_v;{ display graph of quantity packet}
var div1,div2,mod1,mod2,mods,i,j : byte;
   maxt : integer;
begin
    maxt := maxtime;
    for j := 1 to 09 do
    begin
     if j = 1 then textcolor(red) else textcolor(brown);
     if k_scale = 10 then
     begin
       div1 := value_array[maxt] div 10;
       mod1 := value_array[maxt] mod 10;
     end;
     if k_scale = 100 then
     begin
       div1 := value_array[maxt] div 100;
       mod1 := value_array[maxt] mod 100;
       div2 := mod1 div 10;
       mod2 := mod1 mod 10;
       if mod2 >= 5 then div2 := div2+1;
```

```
  mod1 := div2;
end;
if mod1 <= 3 then mods := 0;
if( mod1 > 3) and (mod1 < 8) then mods := 1;
if(mod1 >= 8) then
begin
  div1 := div1+1;
  mods := 0;
end;
if div1 >10 then
begin
  div1 := 10;
  mods := 0;
end;
for i := 12 downto 3  do
begin
  gotoxy(j*7,i);
  write('□□ □');
end;
for i := 12 downto 13-div1 do
begin
  gotoxy(j*7,i);
  write('□□□');
end;
if mods = 1 then
begin
  gotoxy(j*7,13-div1-1);
  write('');
  gotoxy(j*7,13-div1-2);
end
else
gotoxy(j*7,14-div1-2);
if (value_array[maxt] >= 1) and (value_array[maxt] < 4) then
begin
  gotoxy(j*7,12);
```

```
          write('');
          gotoxy(j*7,11);
        end;
        if maxt <> 0 then write(value_array[maxt]);
        gotoxy(j*7,14);
        if maxt <> 0 then write(maxt);
        if maxt < 1 then maxt := 0 else  maxt := maxt-1;
      end;
end;

procedure showscale_s;{ display scale of quantity data}
var i,j : integer;
   maxt : integer;
begin
      maxt := maxtime;
      for j := 1 to 10 do
      begin
        gotoxy(1,j);
        write('                                        ');
      end;
      k_scale := 1;
      for i := 1 to  9 do
      begin
        if kbyte(size_array[maxt])  > 10 then
        k_scale := 4;
        if maxt - 1 >= 0 then maxt := maxt-1;
      end;
      gotoxy(1,1);
      textcolor(red);
      writeln('    SIZE (KILOBYTE) ');
      textcolor(white);
      writeln('     ย');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
```

```
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
      writeln('    ฿ณ');
       write('   0
ภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤ
ฤด');
      textcolor(red);
      writeln('TIME');
      textcolor(green);
      j := 10;
      for i := 1 to 10 do
      begin
        gotoxy(1,2+i);
        write(j*k_scale);
        j := j-1
      end;
end;


procedure showgraph_s; { display graph of quantity data}
var   div1,div2,mod1,mod2,mods : byte;
      maxt : integer;
begin
      maxt := maxtime;
      for j := 1 to 09 do
      begin
       if j = 1 then textcolor(red) else textcolor(brown);
       if k_scale = 1 then div1 :=  kbyte(size_array[maxt]) ;
       if k_scale = 4 then
       begin
         div1 :=  kbyte(size_array[maxt]) div 4;
```

```
        mod1 :=  kbyte(size_array[maxt]) mod 4;
        if mod1 > 2 then div1 := div1+1
      end;
      for i := 12 downto 3  do
      begin
        gotoxy(j*7,i);
        write('□□□');
      end;
      for i := 12 downto 13-div1 do
      begin
        gotoxy(j*7,i);
        write('□□□');
      end;
      gotoxy(j*7,14-div1-2);
      if ( div1 = 0) then
      begin
        gotoxy(j*7,12);
        write('___');
        gotoxy(j*7,11);
      end;
      if maxt <> 0 then write((size_array[maxt]));
      gotoxy(j*7,14);
      if maxt <> 0 then write(maxt);
      if maxt < 1 then maxt := 0 else  maxt := maxt-1;
    end;
end;


procedure showdatalink; { display pattern 'frame type' of capturing}
var   long : word;
    layer2 : string;
begin
    textcolor(white);
    long := (rcvPacket[12]*$100) + rcvPacket[13];
    if  long   >   1500                then  layer2 := 'Ethernet II'    else
```

```
        if (rcvPacket[14] and rcvPacket[15] ) = $ff then  layer2 := 'IEEE 802.3' else
        if (rcvPacket[14] and rcvPacket[15] ) = $aa then  layer2 := 'Ethernet Snap'  else
                                     layer2 := 'IEEE 802.2' ;

        showtime(Hour,Minute,Second);
        write('  Desc : '+ get_hex(showpacket[0])+ get_hex(Rcvpacket[1])
               + get_hex(Rcvpacket[2])+ get_hex(Rcvpacket[3])
               + get_hex(Rcvpacket[4])+ get_hex(Rcvpacket[5]));
        write('  Src : ' + get_hex(Rcvpacket[6])+ get_hex(Rcvpacket[7])
               + get_hex(Rcvpacket[8])+ get_hex(Rcvpacket[9])
               + get_hex(Rcvpacket[10])+ get_hex(Rcvpacket[11]));
        writeln('  Type : [',get_hex(Rcvpacket[6])+ get_hex(Rcvpacket[7]),']  ',layer2);
end;



procedure showprotocol_as;{ display background pattern 'protocol upper layer' }
begin
        gotoxy(1,1);
        writeln('                    DISPLAY PROTOCOL UPPER LAYER');
        writeln('ออออออออออออออออออออออฅอัฅอออออัอออออออัอออออออออออออฒอออออออัอ
ออออัออออออออธ');
        writeln('ณProtocol Upper Layerณ PACKETณ  SIZE  ณProtocol Upper Layerณ
PACKETณ  SIZE  ณ');
        writeln('ฦอออออออออออออออออออออุอออออออุอออออออุอออออออออออออออออออออุ
อออออออุออออออออต');
        writeln('ณ Remote Job Entry  ณ      ณ      ณ Telnet          ณ      ณ      ณ');
        writeln('ณ Echo              ณ      ณ      ณ STMP            ณ      ณ      ณ');
        writeln('ณ Discard           ณ      ณ      ณ Time            ณ      ณ      ณ');
        writeln('ณ Active Users      ณ      ณ      ณ Name Server     ณ      ณ      ณ');
        writeln('ณ Daytime           ณ      ณ      ณ TFTP            ณ      ณ      ณ');
        writeln('ณ Netstat           ณ      ณ      ณ WWW(HTTP)       ณ      ณ      ณ');
        writeln('ณ Quotd             ณ      ณ      ณ Sun RPC         ณ      ณ      ณ');
        writeln('ณ FTP_Data          ณ      ณ      ณ OTHER  PROTOCOL ณ      ณ      ณ');
        writeln('ณ FTP (Control)     ณ      ณ      ณ                 ณ      ณ      ณ');
```

```
    write ('ออออออออออออออออออออออฯออออออออฯออออออออฯออออออออออออออออออออ
ออฯออออออออฯออออออออพ');
end;

procedure showprotocol; { display pattern 'protocol upper layer' of capturing}
begin
    checkall(rcvpacket);
    if (layer4 = 'TCP') or (Layer4 = 'UDP') then
    begin
     case (twohex(showpacket[slayer4+0],showpacket[slayer4+1])or(twohex
(showpacket[slayer4+0],showpacket[slayer4+1])) of
        05   : begin
               inc(capture_up[0].value);
               capture_up[0].size := capture_up[0].size+rcvlength;
              end;
        07   : begin
               inc(capture_up[1].value);
               capture_up[1].size := capture_up[1].size+rcvlength;
              end;
        09   : begin
               inc(capture_up[2].value);
               capture_up[2].size := capture_up[2].size+rcvlength;
              end;
        11   : begin
               inc(capture_up[3].value);
               capture_up[3].size := capture_up[3].size+rcvlength;
              end;
        13   : begin
               inc(capture_up[4].value);
               capture_up[4].size := capture_up[4].size+rcvlength;
              end;
        15   : begin
               inc(capture_up[5].value);
               capture_up[5].size := capture_up[5].size+rcvlength;
              end;
```

```
17   : begin
        inc(capture_up[6].value);
        capture_up[6].size := capture_up[6].size+rcvlength;
       end;
20   : begin
        inc(capture_up[7].value);
        capture_up[7].size := capture_up[7].size+rcvlength;
       end;
21   : begin
        inc(capture_up[8].value);
        capture_up[8].size := capture_up[8].size+rcvlength;
       end;
23   : begin
        inc(capture_up[9].value);
        capture_up[9].size := capture_up[9].size+rcvlength;
       end;
25   : begin
        inc(capture_up[10].value);
        capture_up[10].size := capture_up[10].size+rcvlength;
       end;
37   : begin
        inc(capture_up[11].value);
        capture_up[11].size := capture_up[11].size+rcvlength;
       end;
53   : begin
        inc(capture_up[12].value);
        capture_up[12].size := capture_up[12].size+rcvlength;
       end;
69   : begin
        inc(capture_up[13].value);
        capture_up[13].size := capture_up[13].size+rcvlength;
       end;
80   : begin
        inc(capture_up[14].value);
        capture_up[14].size := capture_up[14].size+rcvlength;
```

```
                end;
          111   :  begin
                   inc(capture_up[15].value);
                   capture_up[15].size := capture_up[15].size+rcvlength;
                  end;
          else     begin
                   inc(capture_up[16].value);
                   capture_up[16].size := capture_up[16].size+rcvlength;
                  end;
          end;
          end;
      for i := 0 to 8 do
      begin
        gotoxy(23,5+i);
        write(capture_up[i].value);
        gotoxy(31,5+i);
        write(capture_up[i].size);
      end;

      for i := 9 to 16 do
      begin
        gotoxy(61,5+i-9);
        write(capture_up[i].value);
        gotoxy(69,5+i-9);
        write(capture_up[i].size);
      end;
end;


procedure checkfilter; { filter packet income }
var   src_mac,des_mac : string;
     src_ip,des_ip  : ipaddr;
begin
      checkall(rcvPacket);
      if (layer2 = 'E_II') then if (filtereth[1]) then   cap_ok  := true else
```

```
begin
  cap_ok := false;
  exit;
end;

if (layer2 = 'E_802.2') then if (filtereth[2]) then  cap_ok  := true else
begin
  cap_ok := false;
  exit;
end;

if (layer2 = 'E_802.3') then if (filtereth[3]) then  cap_ok  := true else
begin
  cap_ok := false;
  exit;
end;

if (layer2 = 'E_Snap') then if (filtereth[4]) then  cap_ok  := true else
begin
  cap_ok := false;
  exit;
end;

if (layer3 = 'IPX') then if (filternet[1]) then   cap_ok  := true else
begin
  cap_ok := false;
  exit;
end;

if (layer3 = 'ARP') then if (filternet[2]) then  cap_ok  := true else
begin
  cap_ok := false;
  exit;
end;
```

```
 if (layer3 = 'IP') then if (filternet[3]) then  cap_ok  := true else
 begin
  cap_ok := false;
  exit;
 end;

 des_mac := ( get_hex(rcvpacket[0])+ get_hex(rcvpacket[1])
         +get_hex(rcvpacket[2])+ get_hex(rcvpacket[3])
         +get_hex(rcvpacket[4])+ get_hex(rcvpacket[5]));
 src_mac  :=( get_hex(rcvpacket[6])+ get_hex(rcvpacket[7])
         +get_hex(rcvpacket[8])+ get_hex(rcvpacket[9])
         +get_hex(rcvpacket[10])+ get_hex(rcvpacket[11]));  .

 if (src_all = false) then if  (src_add = src_mac) then cap_ok  := true else
 begin
  cap_ok := false;
  exit;
 end;

 if (des_all = false) then if  (des_add = des_mac) then cap_ok  := true else
 begin
  cap_ok := false;
  exit;
 end;

 if layer3 = 'IP' then
 begin
  for i := 1 to 4 do  src_ip[i] := rcvpacket[slayer3+11+i];
  for i := 1 to 4 do  des_ip[i] := rcvpacket[slayer3+15+i];
 end;

 if layer3 = 'ARP' then
 begin
  for i := 1 to 4 do  src_ip[i] := rcvpacket[slayer3+13+i];
  for i := 1 to 4 do  des_ip[i] := rcvpacket[slayer3+23+i];
```

```
end;

if (layer3 = 'IP') or (layer3 = 'ARP') then
begin
  case ip_add_src of
    1:  cap_ok := true;
    2:  if  (src_ip[1] = ip_src[1]) and  (src_ip[2] = ip_src[2])
        and (src_ip[3] = ip_src[3]) and  (src_ip[4] = ip_src[4]) then
        cap_ok  := true else
        begin
         cap_ok := false;
         exit;
        end;
    3:  begin
         for i := 1 to 4 do
         begin
          if  (src_ip[i] >= ip_src1[i]) and  (src_ip[i] <= ip_src2[i]) then
          begin
          cap_ok  := true;
          if (src_ip[i] > ip_src1[i]) and (src_ip[i] < ip_src2[i]) then  exit;
          end
          else
          begin
           cap_ok := false;
           exit;
          end;
        end;
   end
end;

  case ip_add_des of
    1:  cap_ok := true;
    2:  if  (des_ip[1] = ip_des[1]) and  (des_ip[2] = ip_des[2])
        and (des_ip[3] = ip_des[3]) and  (des_ip[4] = ip_des[4]) then
        cap_ok  := true else
```

```
            begin
             cap_ok := false;
             exit;
            end;
        3:  begin
            for i := 1 to 4 do
            begin
             if  (des_ip[i] >= ip_des1[i]) and  (des_ip[i] <= ip_des2[i]) then
             begin
             cap_ok  := true;
             if (des_ip[i] > ip_des1[i]) and (des_ip[i] < ip_des2[i]) then  exit;
             end
             else
             begin
               cap_ok := false;
               exit;
             end;
            end;
          end
      end;
    end;
end;

procedure message_ip ; { check vaule of ip address and show message}
begin
    SetwinHeader('');
    SetWinAttr($1f);
    SetBoxAttr($1f);
    SetCharAttr($1f);
    Setboxstyle(double);
    SetHeadAttr($1f);
    Windowopen(23,10,55,13);
    writeln('   Range of data is 000-255 ');
    readkey;
    windowclose;
```

```
end;

function showtext(byt : byte) : string;
var st,st1 : string;
begin
    st := '';
    if byt < 100 then st := '0';
    if byt < 10 then st := '00';
    str(byt,st1);
    showtext := st+st1;
end;

function showip_add(bty : ipaddr) : string; { change 4 byte to ip address }
begin
    showip_add := ('['+showtext(bty[1])+'.'+showtext(bty[2])+'.'
            +showtext(bty[3])+'.'+showtext(bty[4])+']');
end;

function ip2text(ip1 : ipaddr) : string;
var ipt,temp : STRING;
   div1,mod1 : byte;
begin
    ipt  := '';
    for i := 1 to 4 do
    begin
     temp := '';
     Str((ip1[i] div 100),temp) ;
     ipt := ipt+temp;
     temp := '';
     str(((ip1[i] mod 100) div 10),temp);
     ipt := ipt+temp;
     temp := '';
     str(((ip1[i] mod 100) mod 10),temp);
     ipt := ipt+temp;
     ip2text := ipt;
```

```
      end;
  end;


  var     key1     : char;
  procedure read_add_ip(b1 : ipaddr ; x,y : byte; var b2 :ipaddr) ;{ get ip address from
  user}
- var  old_add  : STRING;
     char_add : string;
     k :  byte;
     temp,code : integer;
  begin
       k := 1;
       char_add := ip2text(b1);
       repeat
        for i := 1 to 4 do
        begin
         val(char_add[1+((i-1)*3)]+char_add[2+((i-1)*3)]+char_add[3+((i-
  1)*3)],temp,code);
          if (temp > 255)and ( (k < (1+((i-1)*3))) or (k > (3+((i-1)*3))))   then
          begin
           message_ip;
           char_add[3+((i-1)*3)] := char_add[2+((i-1)*3)];
           char_add[2+((i-1)*3)] := char_add[1+((i-1)*3)];
           char_add[1+((i-1)*3)] := '0';
          end;
        end;
        for i := 1 to 12 do
        begin
         case i of
         1,2,3    : gotoxy(x+i,y);
         4,5,6    : gotoxy(x+i+1,y);
         7,8,9    : gotoxy(x+i+2,y);
         10,11,12 : gotoxy(x+i+3,y);
         end;
        SetAttr($3A);
```

```
        if i = k  then  setAttr($4B);
        write(char_add[i]);
      end;
      key1 := readkey ;
      case key1 of
        lt_key   :   if k = 1 then k := 12 else k := k - 1;
        rt_Key   :   if k = 12 then k := 1 else k := k + 1;
        home_Key :   k := 1;
        end_Key  :   k := 12;
        '1','2','3','4','5','6','7','8','9','0' :
                    begin
                     char_add[k] := key1;
                     if k = 12 then k := 1 else k := k + 1;
                    end;
        esc_key :char_add  :=  ip2text(b1);
       end;
     until  (key1 = tab) or (key1 = esc_key) or (key1 = return_key) or (key1 = up_key) or
(key1 = dn_key);
     for i := 1 to 4 do
     begin
      val(char_add[1+((i-1)*3)]+char_add[2+((i-1)*3)]+char_add[3+((i-1)*3)],temp,code);
      b2[i] := temp;
     end;
     SetAttr($3a);
     gotoxy(x,y);
     write(showip_add(b2));
end;


procedure filter_add;{set filter of menu protocol filter }
VAR   MSG : ARRAY[1..7] OF STRING[22];
     old_ip_src,old_ip_des : BYTE;
     old_src_add,old_des_add : string;
     kk : byte;              .
begin
```

```
kk := 1;
old_ip_src   := ip_add_src;
old_ip_des   := ip_add_des;
MSG[1] := ' ALL ADDRESS        ';
MSG[2] := ' SPECIFIED        :';
MSG[3] := ' RANGE ADDRESS FILTER :';
MSG[4] := ' ALL ADDRESS        ';
MSG[5] := ' SPECIFIED        :';
MSG[6] := ' RANGE ADDRESS FILTER :';
SetwinHeader('ADDRESS FILTER');
SetWinAttr($30);
SetBoxAttr($30);
SetCharAttr($30);
Setboxstyle(double);
SetHeadAttr($30);
Windowopen(16,7,64,21);
gotoxy(1,1);
writeln('ฤฤฤSOURCE  ADDRESSฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฟ' );
writeln('ณ                                    ณ');
writeln('ณ                                    ณ');
writeln('ณ                                    ณ');
writeln('ณ                                    ณ');
writeln('ภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฐ');
writeln('ฤฤฤDESTINATION  ADDRESSฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฟ' );
writeln('ณ                                    ณ');
writeln('ณ                                    ณ');
writeln('ณ                                    ณ');
writeln('ณ                                    ณ');
  write('ภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฐ');
K := 1;
repeat
  FOR I := 1 TO 3  do
  begin
    gotoxy(3,1+i);
    SetAttr($31);
```

```
 if i = k  then  setAttr($7f);
 st := '( )';
 if  (ip_add_src = i) then st := '(o)' ;
 write(st + msg[i]);
 if i = 2  then
 begin
  SetAttr($3A);
  write(showip_add(ip_src));
  gotoxy(3,5);
  write(showip_add(ip_src1));
  Setattr($31);
  write(' TO');
  SetAttr($3A);
  gotoxy(24,5);
  write(showip_add(ip_src2));
 end;
 SetAttr($31);
 if k = 7  then  setAttr($7f);
 setAttr($40);
 if k = 7 then  setAttr($7e);
 gotoxy(21,13);
 write(' OK ');
end;
FOR I := 4 TO 6  do
begin
 gotoxy(3,4+i);
 SetAttr($31);
 if i = k  then  setAttr($7f);
 st := '( )';
 if  (ip_add_des = i-3) then st := '(o)' ;
 write(st + msg[i]);
 if i = 5  then
 begin
  SetAttr($3A);
  write(showip_add(ip_des));
```

```
        gotoxy(3,11);
        write(showip_add(ip_des1));
        Setattr($31);
        write(' TO');
        SetAttr($3A);
        gotoxy(24,11);
        write(showip_add(ip_des2));
      end;
    end;
    key := readkey;
    case (Key) of
      up_key   :   if k = 1 then k := 7 else k := k - 1;
      Dn_Key   :   if k = 7 then k := 1 else k := k + 1;
      PgUp_Key :   k := 1;
      PgDn_Key :   k := 6;
      esc_key  :   begin
                     ip_add_src := old_ip_src;
                     ip_add_des := old_ip_des;
                   end;
      tab      :   begin
                     if (k = 2) and ( ip_add_src = 2) then  read_add_ip(ip_src,23,3,ip_src);
                     if (k = 5) and ( ip_add_des = 2) then
read_add_ip(ip_des,23,9,ip_des);
                     if (k = 3) and ( ip_add_src = 3) then
                     begin
                      kk := 1;
                      read_add_ip(ip_src1,3,5,ip_src1);
                      repeat
                       if (key1 = tab) and (kk  = 1) then
                       begin
                         read_add_ip(ip_src2,24,5,ip_src2);
                         kk := 2;
                       end;
                       if (key1 = tab) and (kk  = 2) then
                       begin
```

```
                    read_add_ip(ip_src1,3,5,ip_src1);
                    kk := 1;
                   end;
                  until  (key1 = esc_key) or (key1 = return_key) or (key1 = up_key) or
(key1 = dn_key);
                end;
                if (k = 6) and ( ip_add_des = 3) then
                begin
                 kk := 1;
                 read_add_ip(ip_des1,3,11,ip_des1);
                 repeat
                  if (key1 = tab) and (kk  = 1) then
                  begin
                   read_add_ip(ip_des2,24,11,ip_des2);
                    kk := 2;
                  end;
                 if (key1 = tab) and (kk  = 2) then
                 begin
                  read_add_ip(ip_des1,3,11,ip_des1);
                  kk := 1;
                 end;
                 until  (key1 = esc_key) or (key1 = return_key) or (key1 = up_key) or
(key1 = dn_key);
                end;
               end;

        return_key:   begin
                case k of
                  1  : ip_add_src := 1;
                  2  : ip_add_src := 2;
                  3  : ip_add_src := 3;
                  4  : ip_add_des := 1;
                  5  : ip_add_des := 2;
                  6  : ip_add_des := 3;
                 end;
```

```
                    end;
        end;

        until  (key = esc_key) or ((k = 7) and (key = return_key));
        windowclose;
end;


procedure assigntime;{menu set time interval }
var    i,old,k  : byte;
          key : char;
          msg : array[1..5] of string  ;
           st : string;
begin
      old := menutime;
      k := 1;
      msg[1] := '   1    SECOND '  ;
      msg[2] := '   10   SECOND '  ;
      msg[3] := '   30   SECOND '  ;
      msg[4] := '   1    MINUTE '  ;
      msg[5] := '   5    MINUTE '  ;
      Cursoroff;
      SetwinHeader('SELECT TIME');
      SetWinAttr(window_c);
      SetBoxAttr(window_c);
      SetCharAttr(text_c);
      Setboxstyle(double);
      SetHeadAttr(window_c);
      Windowopen(30,10,55,17);
      repeat
       for  i := 1 to 5 do
       begin
         gotoxy(4,i);
         SetAttr(window_c);
         if i = k then setAttr($7f);
```

```
        if i = menutime then st := '(o)' else st := '( )';
        write(st,msg[i]);
      end;
      setAttr($40);
      if k = 6 then  setAttr($7e);
      gotoxy(10,6);
      write(' OK ');
      key := readkey;
      case (Key) of
          up_key    :   if k = 1 then k := 6 else k := k - 1;
          Dn_Key    :   if k = 6 then k := 1 else k := k + 1;
          PgUp_Key  :   k := 1;
          PgDn_Key  :   k := 5;
          return_key :  if (k > 0) and(k <= 5) then  menutime := k;
          Esc_key   :   menutime := old;
        end;
    until ((key = return_key)and(k = 6))or(key = esc_key) ;
    windowclose;
end;

var  fmenu1,fmenu2 : integer;

procedure menufilter(k : byte);
var   key  : char;
    msg  : array[1..2] of string;
    fmenu : integer;
    x,y  : byte;
begin
    if k = 1 then
    begin
     msg[1] := '  FRAME TYPE    ' ;
     msg[2] := '  MAC ADDRESS   ' ;
     x := 40;
     y := 6 ;
     fmenu := fmenu1;
```

```
end;
if k = 2 then
begin
 msg[1] := '  PROTOCOL TYPE   ' ;
 msg[2] := '  IP ADDRESS      ' ;
 x  := 40;
 y  :=  7;
 fmenu := fmenu2;
end;
SetWinAttr (choice_c);
SetBoXAttr (choice_c);
SetCharAttr(choice_c);
SetBoxStyle(single);
SetWinHeader('');
Windowopen(x,y,x+21,y+3);
repeat
 for  i := 1 to 2 do
 begin
  gotoxy(2,i);
  SetAttr(window_c);
  if i = fmenu then setAttr($7f);
  write(msg[i]);
 end;
 key := readkey;
 case (Key) of
     up_key   :   if fmenu = 1 then fmenu := 2 else fmenu := fmenu - 1;
     Dn_Key   :   if fmenu = 2 then fmenu := 1 else fmenu := fmenu + 1;
     PgUp_Key :   fmenu := 1;
     PgDn_Key :   fmenu := 2;
   end;
until (key = return_key) or(key = esc_key) ;
case k of
 1 : fmenu1 := fmenu;
 2 : fmenu2 := fmenu;
end;
```

```
      windowclose;
end;



function read_add_mac(st : string ; x,y : byte) : string;
{get mac address from user }
var    char_add  : string;
       key1      : char;
       k         :  byte;
begin
       k := 1;
       char_add := st;
       repeat
        for i := 1 to 12 do
        begin
          gotoxy(x+i,y);
          SetAttr($3A);
          if i = k  then  setAttr($4B);
          write(char_add[i]);
        end;
        key1 := readkey ;
        case key1 of
            lt_key    :    if k = 1 then k := 12 else k := k - 1;
            rt_Key    :    if k = 12 then k := 1 else k := k + 1;
            home_Key  :    k := 1;
            end_Key   :    k := 12;
            '1','2','3','4','5','6','7','8','9','0','A','B','C','D','E','F' :
                           begin
                            char_add[k] := key1;
                            if k = 12 then k := 1 else k := k + 1;
                           end;
           'a','b','c','d','e','f' :
                           begin
                            for i := 1 to 32 do dec(key1);
                            char_add[k] := key1;
```

```
                    if k = 12 then k := 1 else  k := k + 1;
                  end;
         return_key,up_key,dn_key :  read_add_mac := char_add;
         esc_key   :    read_add_mac := st;
         end;
    until  (key1 = esc_key) or (key1 = return_key) or (key1 = up_key) or(key1 = dn_key);
end;


procedure filter_add_mac;   { menu set mac filter}
var   MSG  :  ARRAY[1..6] OF STRING[20];
    old_src_all,old_des_all : boolean;
    old_src_add,old_des_add : string;
begin
    old_src_all  := src_all;
    old_des_all  := des_all;
    old_src_add  := src_add;
    old_des_add  := des_add;
    MSG[1] := ' ALL ADDRESS      ';
    MSG[2] := ' SPECIFIED      :';
    MSG[3] := ' ALL ADDRESS      ';
    MSG[4] := ' SPECIFIED      :';
    SetwinHeader('ADDRESS FILTER');
    SetWinAttr($30);
    SetBoxAttr($30);
    SetCharAttr($30);
    Setboxstyle(double);
    SetHeadAttr($30);
    Windowopen(16,9,64,19);
    gotoxy(1,1);
    writeln('ฤฤฤSOURCE  ADDRESSฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฟ');
    writeln('ณ                               ณ');
    writeln('ณ                               ณ');
    writeln('ภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤู');
    writeln('ฤฤฤDESTINATION  ADDRESSฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฟ');
```

```
writeln('ณ                              ณ');
writeln('ณ                              ณ');
 write('ภฤฤฤ
```

```
        write(st + msg[i]);
        if i = 4  then
        begin
          SetAttr($3A);
          write('  ',des_add);
        end;
      end;
      key := readkey;
      case (Key) of
          up_key   :   if k = 1 then k := 5 else k := k - 1;
          Dn_Key   :   if k = 5 then k := 1 else k := k + 1;
          PgUp_Key :   k := 1;
          PgDn_Key :   k := 4;
          esc_key  :
          begin
          src_all  := old_src_all;
          des_all  := old_des_all;
          src_add  := old_src_add;
          des_add  := old_des_add;
          end;
          tab     :  begin
                     if (k = 2) and (src_all = false) then  src_add := read_add_mac
(src_add,24,3);
                     if (k = 4) and (des_all = false) then  des_add := read_add_mac
(des_add,24,7);
                  end;
          return_key:
                  begin
                   case k of
                       1 : src_all :=  true;
                       2 : src_all := false;
                       3 : des_all :=  true;
                       4 : des_all := false;
                   end;
                  end;
```

```
        end;
      until  (key = esc_key) or ((k = 5) and (key = return_key));
      windowclose;
 end;


 procedure show_status; { show configure all }
 begin
      SetwinHeader('SHOW STATUS ALL');
      SetWinAttr($30);
      SetBoxAttr($30);
      SetCharAttr($30);
      Setboxstyle(double);
      SetHeadAttr($30);
      Windowopen(1,4,80,24);
      gotoxy(1,1);
      SetAttr($31);
      writeln('
ฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤTIMEฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤ
ฤฤฤฤฤฤฤฤฤฤฟ');
      writeln(' ณ                                              ณ');
      writeln('
ภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤ
ฤฤฤฤฤฤฤฤฤฤฤู');
      writeln('ฺฤฤฤFRAME    TYPE    FI LTERฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤยฤฤฤฤฤฤPROT OCOL    FI LTER
ฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฟ');
      writeln(' ณฺฤฤฤฤFRAME  TYPEฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฟณฺฤฤฤฤPROTOCOL  TYPE
ฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฟณ');
      writeln(' ณณ                        ณณณ                        ณณ');
      writeln(' ณณ                        ณณณ                        ณณ');
      writeln(' ณณ                        ณณณ                        ณณ');
      writeln(' ณณ
ณณภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤูณ');
      writeln(' ณภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤูณฺฤฤฤฤSOURCE
ADDRESS (IP ADDRESS)ฤฤฤฤฟณ');
```

```
      writeln(' ณฺฤฤฤSOURCE ADDRESS (MAC ADDRESS)ฤฤฤฟณณ
ณณ');
      writeln(' ณณ                          ณณณ                          ณณ');
      writeln(' ณณ
ณณภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤูณ');
      writeln(' ณภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤูณฺฤฤDESTINATION
ADDRESS (IP ADDRESS)ฤฟณ');
      writeln(' ณฺฤDESTINATION ADDRESS (MAC ADDRESS)ฟณณ
ณณ');
      writeln(' ณณ                          ณณณ                          ณณ');
      writeln(' ณณ
ณณภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤูณ');
      writeln(' ณภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤูณ
ณ');
       write('
ภฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤมฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤ
ฤฤฤฤฤฤฤฤฤฤู');
      SetAttr($36);
      gotoxy(3,2);
      write('   TIME INTERVAL  =  ');
      SetAttr($39);
      write(timeAssing,' SECOND');
      SetAttr($37);
      st := '( )';
      if filtereth[1] then
      begin
        st := '(X)';
        SetAttr($36);
      end;
      gotoxy(6,6);
      writeln(st,' ETHERNET II');
      SetAttr($37);
      st := '( )';
      if filtereth[2] then
      begin
```

```
  st := '(X)';
  SetAttr($36);
end;
gotoxy(6,7);
writeln(st,' ETHERNET 802.2');
SetAttr($37);
st := '( )';
if filtereth[3] then
begin
  st := '(X)';
  SetAttr($36);
end;
gotoxy(6,8);
writeln(st,' ETHERNET 802.3');
SetAttr($37);
st := '( )';
if filtereth[4] then
begin
  st := '(X)';
  SetAttr($36);
end;
gotoxy(6,9);
writeln(st,' ETHERNET SNAP');
SetAttr($37);
st := '( )';
if filternet[1] then
begin
  st := '(X)';
  SetAttr($36);
end;
gotoxy(43,6);
writeln(st,' IPX PROTOCOL ');
SetAttr($37);
st := '( )';
if filternet[2] then
```

```
begin
  st := '(X)';
  SetAttr($36);
end;
gotoxy(43,7);
writeln(st,' ARP PROTOCOL');
SetAttr($37);
st := '( )';
if filternet[3] then
begin
  st := '(X)';
  SetAttr($36);
end;
gotoxy(43,8);
writeln(st,' IP PROTOCOL');
SetAttr($36);
gotoxy(7,12);
if src_all then  writeln('        ALL ADDRESS')
else
begin
  write('        SPECIFIED   ');
  gotoxy(7,13);
  write('   ADDRESS : ');
  Setattr($39);
  writeln(src_add);
end;
SetAttr($36);
gotoxy(7,16);
if des_all then  writeln('        ALL ADDRESS ')
else
begin
  write('        SPECIFIED   ');
  gotoxy(7,17);
  write('   ADDRESS : ');
  Setattr($39);
```

```
  writeln(des_add);
end;
SetAttr($36);
gotoxy(43,11);
case ip_add_src of
  1: writeln('         ALL ADDRESS ');
  2: begin
       write('       ADDRESS  FILTER');
       gotoxy(41,12);
       write('    ADDRESS : ');
       Setattr($39);
       write(showip_add(ip_src));
     end;
  3: begin
       write('     ADDRESS RANGE FILTER');
       gotoxy(41,12);
       Setattr($39);
       write(showip_add(ip_src1));
       write('-');
       write(showip_add(ip_src2));
     end;
end;
SetAttr($36);
gotoxy(43,15);
case ip_add_des of
  1: writeln('         ALL ADDRESS ');
  2: begin
       write('       ADDRESS FILTER');
       gotoxy(41,16);
       write('    ADDRESS : ');
       Setattr($39);
       write(showip_add(ip_des));
     end;
  3: begin
       write('     ADDRESS RANGE FILTER');
```

```
            gotoxy(41,16);
            Setattr($39);
            write(showip_add(ip_des1));
            write('-');
            write(showip_add(ip_des2));
          end;
      end;
      readkey;
      windowclose;
end;

procedure capture; { capture packet data }
var   value_p,size_p,sec_use : word;
      full : boolean;
begin
      if (save_data) and (longp >= 1)  then savedata;
      maxtime := 1;
      datafp  := 0;
      longp   := 0;
      full    := false;
      setwinattr($1f);
      setHeadAttr($1f);
      setBoxattr($1f);
      setcharattr($1f);
      setwinheader(' DATA ETHERNET ');
      setboxstyle(2);
      windowopen(1,3,80,9);
      value_p := 0;
      size_p  := 0;
      value_all := 0;
      size_all  := 0;
      sec_use := 0;
      Init;
      IF Driver_loaded = true then
      begin
```

```
  GetTime(Hour,Minute,Second,Sec100);
  setwinattr($1f);
  SetHeadAttr($1f);
  setBoxattr($1f);
  setcharattr($1f);
  setwinheader(' ETHERNET ');
  setboxstyle(2);
  windowopen(1,9,80,24);
  case menu_show of
     1: showscale_v;
     2: showscale_s;
     5: showprotocol_as;
  end;
repeat
  IF KeyPressed THEN Key := ReadKey;
  IF (PacketCount = 1)or(key = 'a') THEN
BEGIN
checkfilter;
if  (datafp+rcvlength+1  < 40000 ) then
begin
  if cap_ok  then
  begin
    inc(value_p);
    size_p := size_p+rcvlength;
    value_all := value_all+1;
    size_all := size_all+rcvlength;
    TextColor( green );
    case menu_show of
      5 : showprotocol;
      6 : showdatalink;
    end;
    datafp := datafp+1;
    longp := longp+1;
    j := 0;
    for i := datafp to datafp+rcvlength do
```

```
    begin
      dataf[i] :=  rcvpacket[j];
      j := j+1;
    end;
    datafp := datafp+rcvlength;
    long[longp] := rcvlength;
  end;
end
else
full := true;
RcvLength   := 0;
PacketCount := 0;
key := '0';
end;
GetTime(Hour2,Minute2,Second2,Sec1002);
if (menu_show = 1) or ( menu_show = 2) then
begin
  textcolor(7);
  gotoxy(60,1);
  showtime(Hour,Minute,Second);
end;
if (second2 <> second) then
begin
  inc(sec_use);
  second := second2;
  hour   := hour2;
  minute := minute2;
end;
if sec_use =  timeAssing then
begin
  value_array[maxtime] := value_p;
  size_array[maxtime] := size_p;
  case menu_show of
  1:  begin
        showscale_v;
```

```
            showgraph_v;
           end;
        2:  begin
            showscale_s;
            showgraph_s;
           end;
       end;
       value_p := 0;
       size_p := 0;
       sec_use := 0;
       maxtime := maxtime+1 ;
      end;
      until (Key = #27)or(full);
      save_data := true;
      if full then
      begin
       SetwinHeader('');
       SetWinAttr($1f);
       SetBoxAttr($1f);
       SetCharAttr($1f);
       Setboxstyle(double);
       SetHeadAttr($1f);
       Windowopen(25,10,55,13);
       writeln('      Data is Full');
       readkey;
       windowclose;
      end;
      writeln(datafp);
      WindowClose;
      end;
      WindowClose;
end;

procedure ABOUT;{ display message about }
var   ch:char;
```

```
begin
    if activemenu = true then windowclose;
    activemenu := false;
    DisplayMainMenu(0);
    SetWinAttr ($7f);
    SetBoxAttr ($7f);
    SetCharAttr($7f);
    SetBoxstyle(double);
    SetHeadAttr($7f);
    setwinheader(' ABOUT ');
    Windowopen(18,5,62,20);
    GotoXY(0,2);
        writeln;
SetAttr($7b); Write('   Network Monitoring Tool');
SetAttr($74); Writeln(' Version 1.0');
        writeln;
SetAttr($7a); Writeln('            Authorized            ');
SetAttr($79); Writeln('  Mr. Manoch  Pichpol      ID  41012019  ');
        Writeln('  Mr. Sarayut Chanmuangthai ID  41012023  ');
        writeln;
SetAttr($7a); writeln('              Advisor');
SetAttr($79); writeln('          Mayuree   Lertwatchakul       ');
        Writeln('          Pituk     Thammawarin        ');


    {*************** ok buttom***************}
        gotoxy(16,13);
        SetAttr($1f); Write('  OK   ');
SetAttr($70); write('');
        gotoxy(17,14);
SetAttr($70); write('฿฿฿฿฿฿฿฿');
      repeat
      ch := readkey;
      until (ch = esc_key) or (ch = return_key) ;
      windowclose;
```

```
end;


procedure HELP;{display message help}
var   ch:char;
begin
     if activemenu = true then windowclose;
     activemenu := false;
     DisplayMainMenu(0);
     SetWinAttr ($7f);
     SetBoxAttr ($7f);
     SetCharAttr($7f);
     SetBoxstyle(double);
     SetHeadAttr($7f);
     setwinheader(' HELP ');
     Windowopen(2,4,79,23);
     GotoXY(0,2);
SetAttr($74); writeln(' REAL TIME MODE');
SetAttr($71); writeln(' Descreption : The program will capture data packet and show the
result');
          writeln('              in real time mode');
SetAttr($74); write(' ',char(9));
SetAttr($75); write(' Step1');
SetAttr($71); write(' Set Configuration');
SetAttr($76); writeln(' (hot-key Alt-O)');
SetAttr($74); write(' ',char(9));
SetAttr($75); write(' Step2');
SetAttr($71); writeln(' Set Time interval');
SetAttr($74); write(' ',char(9));
SetAttr($75); write(' Step3');
SetAttr($71); write(' Set Frame type filter and Mac address filter');
SetAttr($74); writeln(' or');
SetAttr($71); writeln('        Set Protocol filter and IP adress filter');
SetAttr($74); write(' ',char(9));
SetAttr($75); write(' Step4');
SetAttr($71); writeln(' Select display pattern of captured result as you want');
```

```
SetAttr($74); write(' ',char(9));
SetAttr($75); write(' Step5');
SetAttr($71); writeln(' Press ESC key to finish capturing');
SetAttr($74); write(' ',char(9));
SetAttr($75); write(' Step6');
SetAttr($71); writeln(' Press F2 if you want to save capturing data');
            writeln;
SetAttr($74); writeln(' VIEW FROM SAVED FILE');
SetAttr($71); writeln(' Descreption : The user can view captured data from file after
captured');
          writeln('            in real time mode');
SetAttr($74);write(' ',char(9));
SetAttr($75); write(' Step1');
SetAttr($71); writeln(' Load data from file by press F3');
SetAttr($74); write(' ',char(9));
SetAttr($75); write(' Step2');
SetAttr($71); write(' Select display pattern from view submenu by press F4');

    {************** ok buttom**************}
          gotoxy(35,17);
          SetAttr($1f); Write('  OK   ');
SetAttr($70);  write('');
          gotoxy(36,18);
SetAttr($70);  write('ßßßßßßßß');
        repeat
        ch := readkey;
        until (ch = esc_key) or (ch = return_key) ;
        windowclose;
      end;



procedure showdata; { analysis protocol and display}
begin
```

```
if longp > 0 then
begin
 setWinAttr ($1f);
 setBoxAttr ($1f);
 SetCharAttr($1f);
 SetBoxstyle(double);
 setwinheader(' Detail Packet ');
 Windowopen(1,3,79,24);
 datafp1 := 1;
 longp1  := 1;
 key := Lt_key;
 repeat
  if (key = Lt_key) or (key = Rt_key) then
  begin
   pline := 0;
   j := 0;
   sline := 0;
   for i := datafp1 to datafp1+long[longp1] do
   begin
    showpacket[j] :=  dataf[i]  ;
    j := j+1;
   end;
   sline := 0;
   gotoxy(1,1);
   checkall(showpacket);
   checklayer2;
   checkarp;
   checkipx;
   checkip;
   checkncp;
   checktcp;
   checkudp;
   checkicmp;
  end;
  if (key = Lt_key) or (key = Rt_key )or(key = Up_key ) or (key = Dn_key) then
```

```
begin
 clrscr;
 if sline < 18 then
 for i := 0 to sline-1 do
 writeln(line[i]) else
 for i := pline to pline+18 do writeln(line[i]);
 gotoxy(1,20);
 write(' ');
 SetAttr($35);
 write(char(255),char(27),char(26),' = Select Packet, ',char(24),char(25),
 ' = View This Page,  Enter = View Code,  Ecs = EXIT□ ');
 SetAttr($1f);
end;
key := readkey;
if key = return_key then showraw;
if key = Dn_key then
if pline+20 <= sline then
begin
 pline := pline+1;
 clrscr;
end
else key := null_key;
if key = Up_key then
if pline-1 >= 0 then
begin
 pline := pline-1;
 clrscr;
end
else key := null_key;
if key = rt_key then
if longp1 < longp then
begin
 datafp1 := datafp1+long[longp1];
 datafp1 := datafp1+1;
 longp1 := longp1+1;
```

```
          end
          else key := null_key ;
          if key = lt_key then
          if longp1 > 1 then
          begin
           datafp1 := datafp1-long[longp1-1];
           datafp1 := datafp1-1;
           longp1 := longp1-1;
          end
          else key := null_key;
         until (key = esc_key);
         key := null_key;
         windowclose;
        end
        else
        begin
         SetwinHeader('');
         SetWinAttr($1f);
         SetBoxAttr($1f);
         SetCharAttr($1f);
         Setboxstyle(double);
         SetHeadAttr($1f);
         Windowopen(25,10,55,13);
         writeln('      Data is empty');
         readkey;
         windowclose;
        end;
end;

procedure graphdatalink; {display statistic of frame type}
var   eth : array[1..4] of real;
      per : real;
      layer2 : string;
      div1 : real;
      graph,g1,g2 : integer;
```

```
      packet_long      : word;
 begin
      for i := 1 to 4 do eth[i] := 0;
      if longp > 0 then
      begin
        setWinAttr ($1f);
        setBoxAttr ($1f);
        SetCharAttr($1f);
        SetBoxstyle(double);
        setwinheader(' Graph (Data link) ');
        Windowopen(1,3,79,25);
        datafp1 := 1;
        longp1  := 1;
        for  k := 1 to longp do
        begin
         j := 0;
         for i := datafp1 to datafp1+long[longp1] do
         begin
           showpacket[j] :=  dataf[i]  ;
           j := j+1;
         end;
         packet_long := (showPacket[12]*$100) + showPacket[13];
         if  packet_long   >   1500   then   eth[1] :=eth[1]+1 else
         if (showPacket[14] and showPacket[15] ) = $ff then  eth[2] := eth[2]+1 else
         if (showPacket[14] and showPacket[15] ) = $aa then  eth[3] := eth[3]+1 else
         eth[4] := eth[4]+1 ;
         datafp1 := datafp1+long[longp1];
         datafp1 := datafp1+1;
         longp1 := longp1+1;
        end;
       gotoxy(10,16);

writeln('ฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤ
ฤฤ');
```

```
        writeln('          ETHENET_II  ETHERNET_802.3  ETHERNET_SNAP
ETHERNET_802.2');
        setattr($14);
        writeln;
        writeln('                    Quantity  of packet (frame type)');
        per := (longp / 100);
        for j := 1 to 4 do
        begin
         div1 := eth[j]/per ;
         if div1 = 0 then graph := 0;
         if (div1 > 0) and (div1 <= 5) then graph := 1;
         if (div1 > 5) and (div1 <=10) then graph := 2;
         if (div1 >10) and (div1 <=15) then graph := 3;
         if (div1 >15) and (div1 <=20) then graph := 4;
         if (div1 >20) and (div1 <=25) then graph := 5;
         if (div1 >25) and (div1 <=30) then graph := 6;
         if (div1 >30) and (div1 <=35) then graph := 7;
         if (div1 >35) and (div1 <=40) then graph := 8;
         if (div1 >40) and (div1 <=45) then graph := 9;
         if (div1 >45) and (div1 <=50) then graph := 10;
         if (div1 >50) and (div1 <=55) then graph := 11;
         if (div1 >55) and (div1 <=60) then graph := 12;
         if (div1 >60) and (div1 <=65) then graph := 13;
         if (div1 >65) and (div1 <=70) then graph := 14;
         if (div1 >70) and (div1 <=75) then graph := 15;
         if (div1 >75) and (div1 <=80) then graph := 16;
         if (div1 >80) and (div1 <=85) then graph := 17;
         if (div1 >85) and (div1 <=90) then graph := 18;
         if (div1 >90) and (div1 <=95) then graph := 19;
         if (div1 >95) and (div1 <=100) then graph := 20;
         {graph := 20;  }
         g1 := graph div 2;
         g2 := graph mod 2;
         setattr($13);
         for i := 15 downto 16-g1 do
```

```
          begin
            gotoxy(j*15,i);
            write('□□□□□');
          end;
          gotoxy(j*15,16-g1-1);
          if g2 > 0 then
          begin
            gotoxy(j*15,16-g1-1);
            write('');
            gotoxy(j*15,16-g1-2);
          end;
          write(eth[j]/per :1:2,' %');
        end;
        readkey;
        windowclose;
      end
      else
      begin
        SetwinHeader('');
        SetWinAttr($1f);
        SetBoxAttr($1f);
        SetCharAttr($1f);
        Setboxstyle(double);
        SetHeadAttr($1f);
        Windowopen(25,10,55,13);
        writeln('      Data is empty');
        readkey;
        windowclose;
      end;
end;

procedure graphnetwork;{display statistic of protocol type}
var   net : array[1..4] of real;
      per : real;
      layer2 : string;
```

```
    div1 : real;
    slayer3,graph,g1,g2 : integer;
    packet_long     : word;
begin
    for i := 1 to 4 do net[i] := 0;
    if longp > 0 then
    begin
     setWinAttr ($1f);
     setBoxAttr ($1f);
     SetCharAttr($1f);
     SetBoxstyle(double);
     setwinheader(' Graph Network Layer ');
     Windowopen(1,3,79,25);
     datafp1 := 1;
     longp1  := 1;
     for  k := 1 to longp do
     begin
      j := 0;
      for i := datafp1 to datafp1+long[longp1] do
      begin
        showpacket[j] :=  dataf[i]  ;
        j := j+1;
      end;
      packet_long := (showPacket[12]*$100) + showPacket[13];
      if  packet_long  >  1500   then layer2 := 'E_II'  else
      if (showPacket[14] and showPacket[15] ) = $ff then layer2 := 'E_802.3'else
      if (showPacket[14] and showPacket[15] ) = $aa then layer2 := 'E_Snap' else
      Layer2 := 'E_802.2' ;
      if layer2 = 'E_II'   then P_Type :=  twohex(showPacket[12],showPacket[13]);
      if layer2 = 'E_Snap' then P_type :=  twohex(showPacket[20],showPacket[21]);
      if (layer2 = 'E_802.3') or (Layer2 = 'E_802.2') then P_type := $8137;
      CASE P_type of
        $0800 :  net[1] := net[1]+1;
        $0806 :  net[2] := net[2]+1;
        $8137 :  net[3] := net[3]+1;
```

```
      else    net[4] := net[4]+1;
     end;
     datafp1 := datafp1+long[longp1];
     datafp1 := datafp1+1;
     longp1 := longp1+1;
    end;
    gotoxy(10,16);

writeln('ฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤฤ
ฤฤ');
    writeln('          IP        ARP        IPX        OTHER    ');
    setattr($14);
    writeln;
    writeln('                  Quantity  of packet (protocol)');
    per := (longp / 100);
    for j := 1 to 4 do
    begin
     div1 := net[j]/per ;
     if div1 = 0 then graph := 0;
     if (div1 > 0) and (div1 <= 5) then graph := 1;
     if (div1 > 5) and (div1 <=10) then graph := 2;
     if (div1 >10) and (div1 <=15) then graph := 3;
     if (div1 >15) and (div1 <=20) then graph := 4;
     if (div1 >20) and (div1 <=25) then graph := 5;
     if (div1 >25) and (div1 <=30) then graph := 6;
     if (div1 >30) and (div1 <=35) then graph := 7;
     if (div1 >35) and (div1 <=40) then graph := 8;
     if (div1 >40) and (div1 <=45) then graph := 9;
     if (div1 >45) and (div1 <=50) then graph := 10;
     if (div1 >50) and (div1 <=55) then graph := 11;
     if (div1 >55) and (div1 <=60) then graph := 12;
     if (div1 >60) and (div1 <=65) then graph := 13;
     if (div1 >65) and (div1 <=70) then graph := 14;
     if (div1 >70) and (div1 <=75) then graph := 15;
     if (div1 >75) and (div1 <=80) then graph := 16;
```

```
        readkey;
        windowclose;
      end;
end;

procedure DoMenu;
begin
    case statusMenu of
     1 : begin
          case (ChoiceMenu[StatusMenu]) of
           1    : begin
                   closemenu;
                   loadbuffer;
                  end;
           2    : begin
                   closemenu;
                   savebuffer;
                  end;
           4    : begin
                   if (save_data) and (longp >= 1)  then savedata;
                   quit;
                  end;
          end;
        end;
     2 : begin
          case (ChoiceMenu[StatusMenu]) of
           1    : begin
                   closemenu;
                   menu_show := 1;
                   capture;
                  end;
           2    : begin
                   closemenu;
                   menu_show := 2;
                   capture;
```

```
            end;
        3   : begin
              closemenu;
              menu_show := 6;
              capture;
            end;
        4   : begin
              closemenu;
              menu_show := 5;
              capture;
            end;
        5   : begin
              closemenu;
              menu_show := 1;
              capture;
            end;
      end;
    end;
3 : begin
    case (ChoiceMenu[StatusMenu]) of
      1   : begin
              closemenu;
              assigntime;
              case menutime of
                1 :timeAssing := 1;
                2 :timeAssing := 10;
                3 :timeAssing := 30;
                4 :timeAssing := 60;
                5 :timeAssing := 300;
              end;
            end;
      2   : begin
              menufilter(1);
              case fmenu1 of
                1: begin
```

```
                closemenu;
                filter_add_mac;
              end;
            2: begin
                closemenu;
                filterprotocol;
              end;
          end;
        end;

    3   : begin
            menufilter(2);
            case fmenu2 of
             1: begin
                closemenu;
                filter_add;
              end;
             2: begin
                closemenu;
                filterprotocol2;
              end;
          end;
        end;

    5   : begin
            closemenu;
            show_status;
          end;
      end;
  end;
4 : begin
    case (ChoiceMenu[StatusMenu]) of
     1 : begin
          closemenu;
          showdata;
```

```
                end;
            2 :  begin
                  closemenu;
                  graphdatalink;
                end;
            3 :  begin
                  closemenu;
                  graphnetwork;
                end;
          end;
        end;
    5 :  begin
          case (ChoiceMenu[StatusMenu]) of
            1 : help;
            3 : about;

          end;
        end;

  end;
end;


procedure  processkey(key : char);
begin
    if FuncKey then
    case (Key) of
        F3_Key   :  begin
                     if (activemenu = true) then closemenu;
                     loadbuffer;
                    end;
        F2_Key   :  begin
                     if (activemenu = true) then closemenu;
```

```
                savebuffer;
               end;
        F1_Key  :  begin
                if (activemenu = true) then closemenu;
                HELP;
               end;
    end
    else
    if (activemenu = true) and (Key = Return_Key) then  DoMenu;
    if  (key =  Esc_key) then
    if activemenu = false then quit else
    begin
     DisplayMainMenu(0);
     activemenu := false;
     windowclose;
    end;
end;


begin
    datafp := 0;
    longp := 0;
    menutime := 1;
    timeAssing := 1;
    fmenu1 := 1;
    fmenu2 := 2;
    for i := 1 to 4 do filtereth[i] := true;
    for i := 1 to 3 do filternet[i] := true;
    ip_add_src := 1;
    ip_add_des := 1;
    cap_ok := true;
    src_all := true;
    des_all := true;
    src_add := 'FFFFFFFFFFFF';
    des_add := 'FFFFFFFFFFFF';
```

```
        save_data := false;
        intmenu;
        repeat
          ReadFuncKey(key);
          Testkey(key);
          processkey(key);
        until Finish;
    end.
```