ตัวกรองอินเตอร์เน็ตโปรโตคอล

IP FILTER

โดย

นางสาวพรสุดา วงศ์สุโชโต

นายมานะ แซ่ตั้ง

นายวุฒิพล อัศวนิรมล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ผ่านการตรวจชิ้นงานแล้ว

(ลงชื่อ).....ผู้ตรวจ

ผ่านการตรวจรูปเล่มแล้ว

(ลงชื่อ).....ผู้ตรวจ

ตัวกรองอินเตอร์เน็ตโปรโตคอล

IP FILTER

โดย

นางสาวพรสุดา วงศ์สุโชโต 45010512

นายมานะ แซ่ตั้ง 45010612

นายวุฒิพล อัศวนิรมล 45010737

อาจารย์ที่ปรึกษา

ผศ. เกรียงไกร วงศ์โรจนภรณ์

ดร. สุวิพล สิทธิชีวภาค

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ตัวกรองอินเตอร์เน็ตโปรโตคอล**

IP FILTER

ผู้จัดทำ

1. นางสาวพรสุดา วงศ์สุโชโต 45010512
2. นายมานะ แซ่ตั้ง 45010612
3. นายวุฒิพล อัศวนิรมล 45010737

..... อาจารย์ที่ปรึกษา

(ผศ. เกรียงไกร วงศ์โรจนภรณ์)

..... อาจารย์ที่ปรึกษา

(รศ.ดร. สุวิพล สิทธิชีวภาค)

ตัวกรองอินเตอร์เน็ตโปรโตคอล

IP FILTER

โดย นางสาวพรสุดา วงศ์สุโชโต 45010512
นายมานะ แซ่ตั้ง 45010612
นายวุฒิพล อัศวนิรมล 45010737

อาจารย์ที่ปรึกษา ผศ. เกรียงไกร วงศ์โรจนภรณ์
รศ.ดร. สุวิพล สิทธิชีวภาค

บทคัดย่อ

โครงงานนี้เป็นการสร้างและออกแบบตัวกรองข้อมูลโดยตรวจจับหมายเลขไอพี (IP) ในชุดข้อมูลที่รับส่งในระบบเครือข่าย โดยใช้วงจรควบคุมซึ่งมีอุปกรณ์ไมโครโพรเซสเซอร์เป็นตัวควบคุมหลักในการจำแนกหมายเลขไอพี (IP) และกรองข้อมูล

ABSTRACT

This project is built and design IP Filter by using Microprocessor to monitor IP number in Internet packet and filtering the packet

สารบัญ

หน้า

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันองค์กรและผู้ประกอบการด้านธุรกิจต่างๆมีความต้องการที่จะใช้ระบบเครือข่ายเพิ่มขึ้น ทำให้ระบบอินเตอร์เน็ต (Internet) มีความสำคัญมากในโลกธุรกิจ เป็นผลทำให้ความปลอดภัยในการใช้ระบบเครือข่ายเป็นสิ่งที่จำเป็นสำหรับผู้ใช้งานเครือข่ายทุกคน แต่ละองค์กรจึงจำเป็นที่จะต้องมีผู้บริหารระบบเครือข่ายหรือ (Network Administrator) เพื่อทำหน้าที่ในการดูแลการใช้ระบบเครือข่ายภายในองค์กร โดยในแต่ละองค์กรควรจะมีอุปกรณ์รักษาความปลอดภัยจำพวก ไฟร์วอล เอเอเอ เซิฟเวอร์ (Firewall AAA Server) และ VPN(Virtual Private Network) เป็นต้น

อุปกรณ์รักษาความปลอดภัยต่างๆ นั้นส่วนใหญ่แล้วจะมีเงื่อนไขในการป้องกันการเข้าถึงจากผู้ใช้ในระบบเครือข่ายอื่นๆที่เกี่ยวข้องกับ Internet Protocol (IP) โดยอุปกรณ์รักษาความปลอดภัยต่างๆมักจะมีการทำงานขั้นพื้นฐานในการที่จะกรอง ไอพี (IP) ที่จะอนุญาติให้ผ่านหรือห้ามผ่านซึ่งแล้วแต่เงื่อนไขในการกำหนดค่าการทำงานของอุปกรณ์รักษาความปลอดภัย

เนื่องจากเห็นได้ว่าการทำงานของอุปกรณ์รักษาความปลอดภัยนั้นส่วนใหญ่แล้วจะมีการทำงานขั้นพื้นฐานในการตรวจจับไอพีแพ็กเกต(IP Packet) แล้วทำการตรวจสอบกับเงื่อนไขที่กำหนด ว่าจะอนุญาติ (Permit) หรือ ปฏิเสธ (Deny) เฟรมข้อมูลนั้นๆให้ติดต่อไปยังปลายทางได้หรือไม่ จากหลักการข้างต้นทำให้ได้มีการออกแบบสร้างอุปกรณ์ไอพีฟิวเตอร์ (IP Filter) ขึ้น เพื่อใช้ในการป้องกันการเข้าถึงข้อมูลจากผู้บุกรุกที่ไม่ได้อนุญาติ โดยจัดเป็นอุปกรณ์ที่มีความสำคัญภายในวงแลน(LAN) เป็นอย่างมาก ทั้งนี้เพราะอุปกรณ์รักษาความปลอดภัยอย่างอื่น เช่นไฟร์วอลล์นี้จะกันการบุกรุกข้อมูลจากภายนอกวงแลน ได้เท่านั้นไม่สามารถป้องกันผู้บุกรุกที่อยู่ภายในวงแลนได้ ซึ่งจะก่อให้เกิดผลเสียถ้าหากมีผู้บุกรุกจากภายในวงแลน ที่ต้องการจะทำลายข้อมูล หรือพยายามคัดลอกข้อมูลที่เป็นความลับเพื่อออกไปใช้ในทางที่เสียหายกับเจ้าของเครื่อง อุปกรณ์ ไอพีฟิวเตอร์ที่ออกแบบนี้สร้างจากโมดูล (Module) ของแรบบิทไมโครโปรเซสเซอร์(RCM2200) โดยใช้ภาษาไดนามิคซี (Dynamic C) ที่พัฒนามาจากบริษัท ZWORLD ในการสร้างโปรแกรมเพื่อควบคุม การทำงานของไอพีฟิวเตอร์

1.2 วัตถุประสงค์ของโครงงาน

1. ศึกษาและทำความเข้าใจการทำงานของระบบเครือข่ายที่ใช้โปรโตคอลทีซีพีไอพี (TCP/IP)
2. ทำความเข้าใจเกี่ยวกับอุปกรณ์ RCM 2200 เพื่อนำมาสร้างและออกแบบฮาร์ดแวร์ (Hardware)
3. นำความรู้ที่ได้ศึกษามาทางด้านระบบเครือข่ายที่ใช้โปรโตคอลทีซีพีไอพี มาประยุกต์ใช้ในการเขียน ซอฟต์แวร์ควบคุมการทำงานของตัวอุปกรณ์

1.3 ขอบเขตของโครงงาน

โครงงานนี้จะทำการศึกษาระบบการสื่อสารพื้นฐานที่ใช้ภายในเครือข่ายซึ่งใช้โปรโตคอลทีซีพีไอพีเพื่อที่จะนำไปออกแบบสร้างตัวกรองอินเตอร์เน็ตโปรโตคอล (IP Filter) ในรูปแบบลักษณะของฮาร์ดแวร์ (Hardware) ที่มีส่วนของซอฟต์แวร์ (Software) เข้าไปควบคุมการทำงานของตัวอุปกรณ์ IP Filter ซึ่งจะมีลักษณะการทำงานในการรับข้อมูลจากต้นทางด้านหนึ่งเข้ามาเก็บในบัฟเฟอร์ก่อนแล้วทำการดีเอ็นแคปซูลเลท (Deencapsulate) ในส่วนของไอพีแล้วนำมาเปรียบเทียบเพื่อตรวจสอบว่าจะให้ผ่านหรือไม่ถ้าไม่ให้ผ่านก็จะทำการกรองข้อมูลหรือลบข้อมูลในบัฟเฟอร์ทิ้งไปแต่ถ้าหากไม่ก็จะทำการ Translation IP address แล้วส่งออกไปยังปลายทางที่ระบุไว้

1.4 วิธีการดำเนินงาน

ในโครงงานนี้จะเริ่มต้นด้วยการศึกษาทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้องกับโครงงาน ซึ่งประกอบไปด้วย 3 เรื่องหลักๆ ดังนี้

1. การสื่อสารพื้นฐานบนเครือข่ายที่ใช้โปรโตคอลทีซีพีไอพี (TCP/IP)
2. คุณสมบัติและวิธีการใช้งานของตัวอุปกรณ์แรบบิทโมดูล (Rabbit Module)
3. นำความรู้ในการติดต่อสื่อสารในเครือข่ายมาออกแบบวิธีการทำงานของอุปกรณ์ในเชิงอัลกอริธึมเพื่อเขียนโปรแกรมควบคุม
3. ศึกษาการเขียนซอฟต์แวร์ (Dynamic C) เพื่อโปรแกรมการทำงานของอุปกรณ์ซึ่งรายละเอียดอื่นๆจะอธิบายเพิ่มเติมในบทต่อไป

บทที่ 2
ทฤษฎีและหลักการ

2.1 ระบบเครือข่าย (Network)

ระบบเครือข่าย หรือ เน็ตเวิร์ก (Network) คือระบบของฮาร์ดแวร์ (Hardware) และซอฟท์แวร์ (Software) ซึ่งเชื่อมต่อเข้าด้วยกัน เพื่อวัตถุประสงค์ในการติดต่อสื่อสารและใช้ทรัพยากรร่วมกัน (Resources Sharing) ซึ่งเป็นการช่วย ประหยัดค่าใช้จ่าย และเพิ่มความสะดวกในการใช้งาน เช่น การใช้พื้นที่ บนฮาร์ดดิสก์ และเครื่องพิมพ์ร่วมกัน สามารถบริหารจัดการ การทำงานของคอมพิวเตอร์ทุกเครื่องได้จากศูนย์กลาง (Centralized Management) เช่น การสร้างเวิร์กกรุ๊ป(Work group) กำหนดสิทธิ์ในการเข้าถึงข้อมูล และสามารถทำการสำรองข้อมูลของแต่ละเครื่องได้ การสื่อสารข้อมูลโดยใช้เน็ตเวิร์กสามารถรองรับการสื่อสารข้อมูลด้วยความเร็วสูง ในกรณีที่มีความจำเป็นที่ต้องสื่อสารหรือส่งข้อมูลใช้ร่วมกันอย่างรวดเร็ว ทำให้ สามารถย่นระยะเวลาทำงานได้มากขึ้น และไม่มีข้อจำกัดในเรื่องสถานที่ไม่ว่าจะอยู่ภายในอาคารเดียวกัน หรือต่างอาคาร ระบบเครือข่ายนับว่ามีประโยชน์อย่างมาก รวมทั้งการสื่อสารกันได้ในระยะทางไกลๆ โดยทั่วไปแล้วเรามักจะจัดแบ่ง ระบบเครือข่าย (Network) ออกตามขนาดเป็น 2 ประเภท คือ Local Area Network (LAN) และ Wide Area Network (WAN)

2.2 Local Area Network (LAN)

แลน (LAN) คือระบบเครือข่ายขนาดเล็ก ที่เครื่องคอมพิวเตอร์ (Computer) ทั้งหมดจะถูกเชื่อมโยงเข้าด้วยกันมมีระยะห่างไม่เกิน 5 กิโลเมตร ระบบเครือข่ายประเภทนี้ มักเป็นที่นิยมใช้ในบริษัทหรือองค์กรขนาดเล็ก ทั้งนี้เนื่องจากใช้งบประมาณในการสร้างและดูแลรักษาน้อย

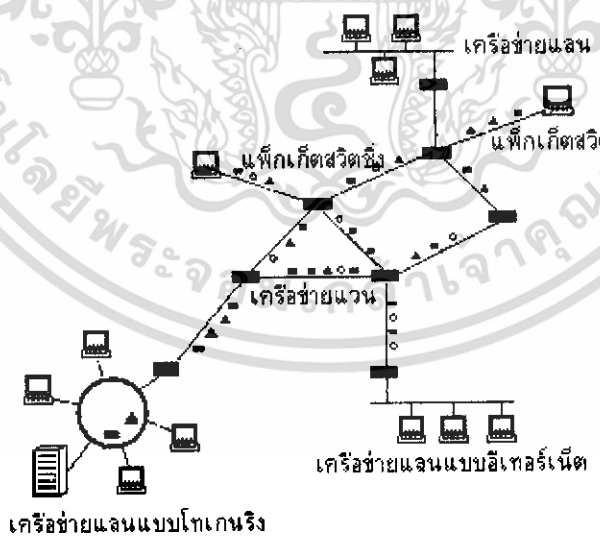
- **โครงสร้างแบบดาว (Star Topology)** เป็นโครงสร้างที่เชื่อมต่อคอมพิวเตอร์แต่ละตัวเข้ากับคอมพิวเตอร์ศูนย์กลาง การรับส่งข้อมูลทั้งหมดจะต้องผ่านคอมพิวเตอร์ศูนย์กลางเสมอ มีข้อดีคือการเชื่อมต่อคอมพิวเตอร์เครื่องใหม่สามารถทำได้ง่าย และไม่กระทบกับเครื่องคอมพิวเตอร์อื่นในระบบเลย แต่ข้อเสียคือมีค่าใช้จ่ายเกี่ยวกับสายสูง และถ้าคอมพิวเตอร์ศูนย์กลางเสียระบบเครือข่ายจะหยุดชะงักทั้งหมดทันที
- **โครงสร้างแบบบัส (Bus Topology)** เป็นโครงสร้างที่เชื่อมคอมพิวเตอร์แต่ละตัวด้วยสายเคเบิล (Cable) ที่ใช้ร่วมกัน ซึ่งสายเคเบิลหรือบัส (Bus) นี้เปรียบเสมือนกับถนนที่ข้อมูลจะถูกส่งผ่านไปมาระหว่างแต่ละเครื่องได้ตลอดเวลา โดยไม่ต้องผ่านไปที่ศูนย์กลางก่อน โครงสร้างแบบนี้มีข้อดีที่ใช้สายน้อย และถ้ามีเครื่องใดเสียก็ไม่มีผลอะไรต่อระบบโดยรวม ส่วนข้อเสียก็คือตรวจหาจุดที่เป็นปัญหาได้ยาก
- **โครงสร้างแบบวงแหวน (Ring Topology)** เป็นโครงสร้างที่เชื่อมคอมพิวเตอร์ทั้งหมดเข้าเป็นวงแหวน ข้อมูลจะถูกส่งต่อๆ กันไปในวงแหวนจนกว่าจะถึงเครื่องผู้รับที่ถูกต้อง ข้อดีของโครงสร้างแบบนี้คือ ใช้สายเคเบิลน้อยและสามารถตัดเครื่องที่เสียออกจากระบบได้ ทำให้ไม่มีผลต่อระบบเครือข่าย ข้อเสียคือหากมีเครื่องที่มีปัญหาอยู่ในระบบจะทำให้เครือข่ายไม่สามารถ

ทำงานได้เลย และการเชื่อมต่อคอมพิวเตอร์เครื่องใหม่เข้าสู่เครือข่ายอาจต้องหยุดระบบทั้งหมดลงก่อน

2.3 Wide Area Network (WAN)

แวน (WAN) คือ ระบบเครือข่าย ที่เครื่องคอมพิวเตอร์ต่างๆ ซึ่งถูกเชื่อมโยงเข้าด้วยกันนั้นอยู่ห่างกันมากกว่า 5 กิโลเมตร ความเร็วในการเชื่อมโยงระหว่างกันอาจไม่สูงมากนัก เนื่องจากระยะทางไกลทำให้มีสัญญาณรบกวนสูง ความเร็วจึงอยู่ในช่วง 9.6-64 Kbps และ 1.5-2 Mbps ขึ้นอยู่กับแอพพลิเคชั่น (Application) และขนาดของข้อมูล ดังนั้น เครือข่ายแวนจึงเป็นเครือข่ายเชื่อมโยงระหว่างองค์กรระหว่างเมือง หรือระหว่างประเทศ และเพื่อให้การใช้งานมีประสิทธิภาพจึงมีองค์กรกลางหรือผู้ให้บริการเครือข่ายสาธารณะเข้ามาช่วยจัดการเพื่อประหยัดค่าใช้จ่าย เช่น เครือข่ายสาธารณะที่ใช้ร่วมกันของทศท. และกสท. หรือ เครือข่ายบริการ เช่น ดาต้าเบต (Database) เป็นต้น

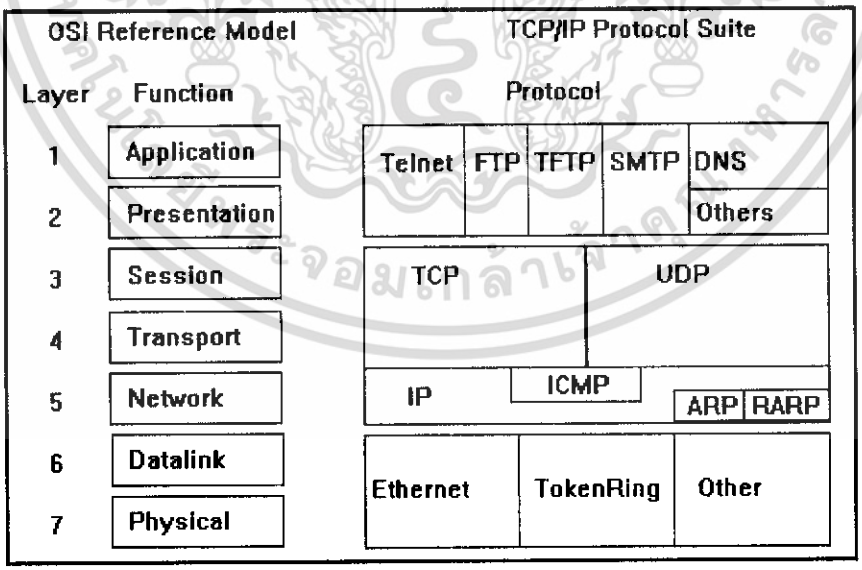
เครือข่ายในปัจจุบันมีการเชื่อมเครือข่ายแลนหลาย ๆ เครือข่ายย่อยเข้าด้วยกัน จะเป็นอีเทอร์เน็ต (Ethernet) หรือ โทเกนริงค์ (Token Ring) ก็ได้ แล้วยังเชื่อมต่อออกจากองค์กรผ่านเครือข่ายแวน ทำให้เครือข่ายทั้งหมดเชื่อมโยงถึงกัน จึงมีการพัฒนาเทคโนโลยีเครือข่ายให้มีความเร็วสูงในการรับส่งข้อมูล ซึ่งเครือข่ายแวนที่ใช้ตัวกลางเป็นเส้นใยแก้วนำแสง (Fiber Optic) สามารถส่งรับข้อมูลได้เร็วไม่น้อยกว่าเครือข่ายแลน การพัฒนาเทคโนโลยีบนเครือข่ายแลน และแวนจึงเป็นสิ่งที่จำเป็น และเป็นโครงสร้างพื้นฐานของการพัฒนาประเทศ และต้องพัฒนาควบคู่ไปกับโครงสร้างพื้นฐานต่าง ๆ ของประเทศ โดยเฉพาะระบบเครือข่ายคอมพิวเตอร์ถือเป็นโครงสร้างพื้นฐานหนึ่งที่ต้องพัฒนาไปด้วย



รูปที่ 2.1 แสดงการเชื่อมต่อของเครือข่ายแวนกับเครือข่าย แลน

2.4 โปรโตคอลทีซีพี/ไอพี (TCP/IP Protocols)

โครงสร้างของโปรโตคอลทีซีพี/ไอพี เมื่อเทียบกับมาตรฐานโอเอสไอ เซเว่น เลเยอร์ (OSI 7 Layer) จากรูปที่ 2.2 แสดงให้เห็นว่าโมเดลของโอเอสไอนั้นได้นิยามหน้าที่ของแต่ละเลเยอร์ไว้โดยชัดเจนและเฉพาะเจาะจง แต่สำหรับโมเดลของโปรโตคอลทีซีพี/ไอพี แล้วได้นิยามแต่ละเลเยอร์ไว้อย่างกว้างๆ โดยเลเยอร์บนสุดจะเกี่ยวข้องกับโปรเซสเซอร์ (Processor) และแอปพลิเคชันต่างๆที่ทำงานอยู่บนเน็ตเวิร์ก ซึ่งครอบคลุมทั้ง 3 เลเยอร์ของโมเดลโอเอสไอ ตัวอย่างของแอปพลิเคชันในเลเยอร์นี้ก็อย่างที่เราคุ้นเคยกันดี เช่น FTP , SMTP , HTTP, POP3 เป็นต้น แอปพลิเคชันเหล่านี้ได้รับการพัฒนาขึ้นให้มีความสามารถในการทำงานครอบคลุมหน้าที่ของโปรโตคอลในระดับ Application / Presentation / Session ตามโมเดลโอเอสไอ โมเดลของโปรโตคอลทีซีพี/ไอพี กำหนดให้มีโปรโตคอลในระดับทรานสปอร์ตเลเยอร์ (Transport Layer) อยู่ 2 ประเภท เพื่อควบคุมการสื่อสารระหว่างโฮสต์ต้นทางกับโฮสต์ปลายทาง ได้แก่โปรโตคอลทีซีพี (TCP:Transmission Control Protocol) และโปรโตคอลยูดีพี (UDP:User Datagram Protocol) ส่วนเลเยอร์ต่ำลงมาเป็นเลเยอร์ของอินเตอร์เน็ตโปรโตคอล (Internet Protocol) ซึ่งรับผิดชอบในเรื่องการรับส่งแพ็กเก็ต (packet) ไปบนเน็ตเวิร์กโดยตรง และเลเยอร์สุดท้ายซึ่งก็คือเน็ตเวิร์กอินเตอร์เฟสเลเยอร์ (Network Interface Layer) ถูกจัดให้เป็นเลเยอร์ของเน็ตเวิร์กประเภทต่างๆที่เข้ามารองรับโปรโตคอลทีซีพี/ไอพี ข้างต้นทั้งหมด เลเยอร์สุดท้ายนี้จะครอบคลุมทั้งดาต้าลิงค์เลเยอร์ (Data Link Layer) และ ฟิสิคัลเลเยอร์ (Physical Layer) ของโมเดลโอเอสไอ อย่างเช่นเน็ตเวิร์กแบบอีเทอร์เน็ตได้กำหนดวิธีการรับส่งข้อมูลในเดาต้าลิงค์เลเยอร์ด้วยอัลกอริทึม (Algorithm) แบบ CSMA / CD และกำหนดมาตรฐานสายเคเบิลและคอนเน็กเตอร์ (Connector)ต่างๆที่ใช้ในอีเทอร์เน็ตอย่างเช่น สาย UTP ประเภทต่างๆและหัวต่อแบบ RJ45 ไว้ด้วย



รูปที่ 2.2 โครงสร้างของโปรโตคอลทีซีพี/ไอพี เมื่อเทียบกับมาตรฐานโอเอสไอ เซเว่น เลเยอร์

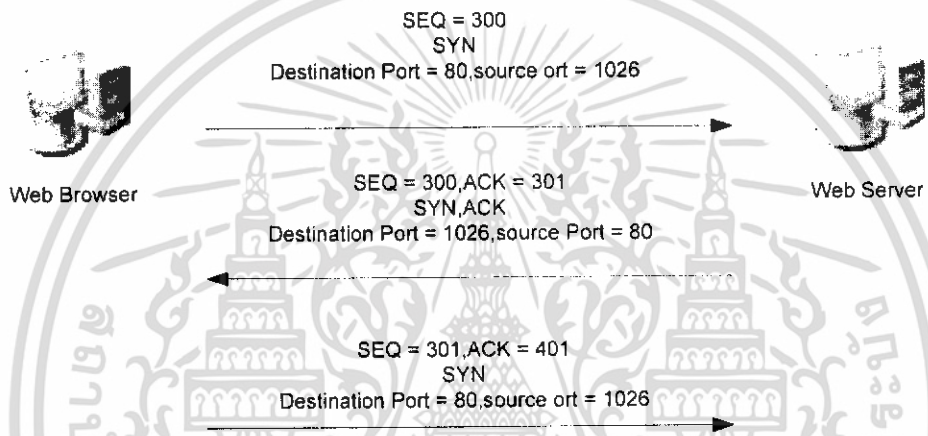
2.5 โปรโตคอลทีซีพี (TCP Protocol)

โปรโตคอลทีซีพีในทรานสปอร์ตเลเยอร์ มีหน้าที่หลักๆก็คือ

- จัดแบ่งข้อมูลจากระดับแอปพลิเคชั่นเลเยอร์ให้มีขนาดพอเหมาะที่จะส่งไปบนเน็ตเวิร์ก หน่วยของข้อมูลในระดับนี้เรียกว่าทีซีพีเซ็กเมนต์ (TCP Segment)
- เริ่มต้นสร้างคอนเน็กชันระหว่างต้นทางและปลายทางให้สำเร็จก่อน ก่อนที่ทั้งต้นทางและปลายทางจะมีการรับส่งข้อมูลกันจริงๆ การรับส่งข้อมูลโดยมีการสร้างคอนเน็กชันก่อนการส่งเรียกว่า การสื่อสารแบบ คอนเน็กชัน-โอเรียนเต็ด (Connection – Oriented) และกระบวนการที่ใช้ในการสร้างคอนเน็กชัน คือกระบวนการทรีเวย์แฮนด์เช็ก (Three Way Handshake)
- มีการใส่หมายเลขซีเควนส์ (Sequence Number (SEQ)) ลงไปในทีซีพีเซ็กเมนต์ ที่ส่งไปเพื่อจัดลำดับการส่งข้อมูล เมื่อปลายทางได้รับทีซีพีเซ็กเมนต์นั้นๆ แล้วจะต้องมีการส่งการยืนยัน (Acknowledgement : ACK) กลับมาให้เครื่องต้นทางทราบว่าได้รับทีซีพีเซ็กเมนต์นั้นๆแล้ว
- ดังที่กล่าวไปในข้อที่ผ่านมาว่าเครื่องต้นทางมีการตรวจสอบว่า แพ็กเกตถูกส่งไปถึงเครื่องคอมพิวเตอร์ปลายทางหรือไม่ โดยการตรวจเช็กว่าได้รับเอซีเค (ACK) กลับมาหรือยัง กรณีที่ไม่ได้รับเอซีเคยืนยันกลับมาภายในเวลารอคอยที่กำหนด เครื่องคอมพิวเตอร์ต้นทางจะเข้าใจว่าทีซีพีเซ็กเมนต์นั้นส่งไปไม่ถึงยังเครื่องคอมพิวเตอร์ปลายทาง ในกรณีนี้เครื่องคอมพิวเตอร์ต้นทางจะมีการส่งแพ็กเกตใหม่ (retransmission) อีกครั้งและเพิ่มเวลารอคอยออกไปอีกระยะหนึ่งจนกว่าจะได้รับเอซีเคกลับมา กลไกนี้คือการทำแอเรอร์ รีโคเวอร์รี่ (Error Recovery) ซึ่งทำให้โปรโตคอลทีซีพีมีความน่าเชื่อถือในการรับส่งข้อมูล (Reliability)
- ในการส่งเอซีเคเพื่อยืนยันว่าได้รับข้อมูลครบถ้วนนั้น เครื่องคอมพิวเตอร์ปลายทางไม่จำเป็นต้องส่งเอซีเคกลับทุกๆทีซีพีเซ็กเมนต์ที่ได้รับ แต่สามารถส่งเอซีเคเมื่อได้รับข้อมูลหลายๆทีซีพีเซ็กเมนต์ตามที่ตกลงกันไว้ก่อนโดยเครื่องคอมพิวเตอร์ต้นทางและเครื่องคอมพิวเตอร์ปลายทางต้องมีการตกลงกันแต่แรกว่าจะให้ผู้ที่ได้รับ ทีซีพีเซ็กเมนต์ตอบยืนยัน (ACK) กลับมาเมื่อได้รับทีซีพีเซ็กเมนต์ไปแล้วเป็นจำนวนเท่าไหร่ ขนาดของทีซีพีเซ็กเมนต์ที่ผู้ส่งสามารถส่งได้ในครั้งหนึ่งๆ โดยไม่ต้องรอคอยให้มีการตอบรับนี้เรียกว่า “Window Size” นั่นหมายความว่า เครื่องคอมพิวเตอร์ปลายทางสามารถรอรับข้อมูลจนครบตามขนาดวินโดว์ไซส์ (Window Size) ก่อนแล้วจึงค่อยส่งเอซีเคตอบกลับไปยังเครื่องต้นทางทีเดียว
- ควบคุมลำดับขั้นตอนของการส่งแพ็กเกตของโปรโตคอลไอพีให้มีความเป็นระเบียบเรียบร้อย จัดเตรียมขนาดบัฟเฟอร์ (Buffer) ข้อมูลที่เหมาะสมไว้ทั้งในขณะรับและขณะส่งข้อมูล และช่วยประกอบรวมไอพีแพ็กเกตที่ได้รับเข้ามาให้เป็นข้อมูลผืนเดียวกันสำหรับส่งต่อขึ้นไปยังแอปพลิเคชันในเลเยอร์ที่สูงขึ้นไป

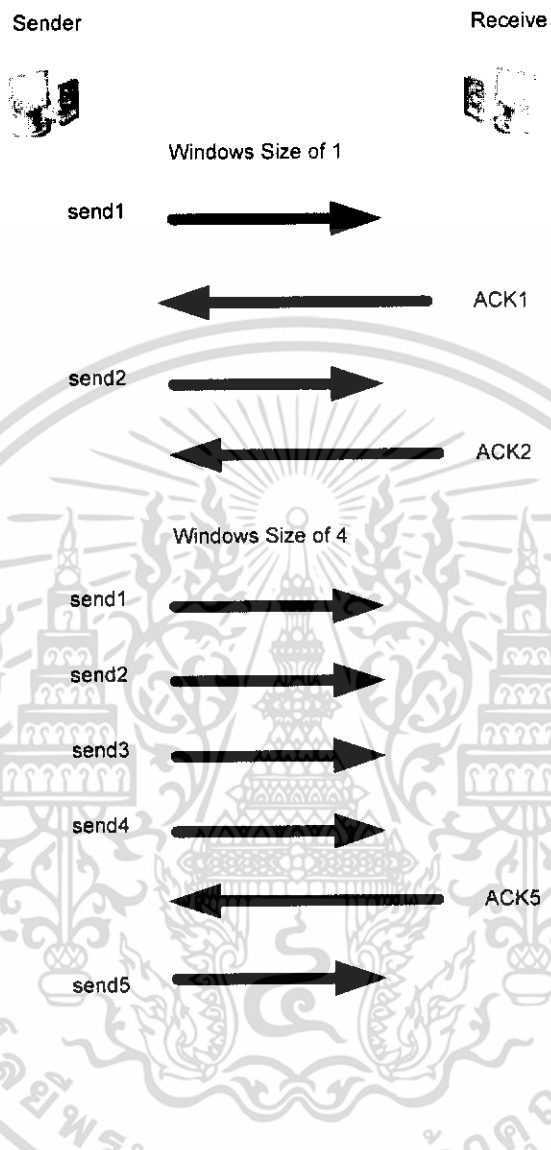
2.5.1 กระบวนการทรีเวย์แฮนด์เชค (Three Way Handshake)

กระบวนการในการสร้างคอนเน็กชันระหว่างโฮสต์ต้นทางกับปลายทางโฮสต์ (Host) ต้นทางจะเริ่มต้นจากการขอสร้างคอนเน็กชันด้วยการส่งทีซีพีแพ็กเกตที่มีการเซตฟิลด์ (Field) เอสวายเอ็น(SYN) ไว้ (ฟิลด์ SYN = Synchronize) และรอให้โฮสต์ปลายทางส่งทีซีพีแพ็กเก็ตที่มีฟิลด์เอสวายเอ็นและเอซีเคกลับมาก่อน จากนั้นต้นทางจึงตอบยืนยันว่าต้องการรับส่งข้อมูลด้วยอีกครั้ง หลังจากผ่านกระบวนการข้างต้นนี้ไป โฮสต์ต้นทางและโฮสต์ปลายทางก็จะพร้อมที่จะเริ่มรับส่งข้อมูลกัน โดยระหว่างกระบวนการนี้ โฮสต์ต้นทางและโฮสต์ปลายทางจะมีการตกลงกันว่าจะใช้ขนาดของวินโดว์ไซส์ขนาดเท่าไหร่ และหมายเลข ซีเควนส์ (Seq) ของโฮสต์แต่ละฝั่งจะมีค่าเริ่มต้นเท่ากับเท่าไหร่ ดังแสดงในรูปที่ 2.3 หมายเลขซีเควนส์ ของต้นทางเท่ากับ 300 และหมายเลขซีเควนส์ของปลายทางเท่ากับ 400



รูปที่ 2.3 ตัวอย่างกระบวนการทรีเวย์แฮนด์เชค

2.5.2 สไลดิ่งวินโดวส์ (Sliding Window) และความน่าเชื่อถือในการรับส่งข้อมูล

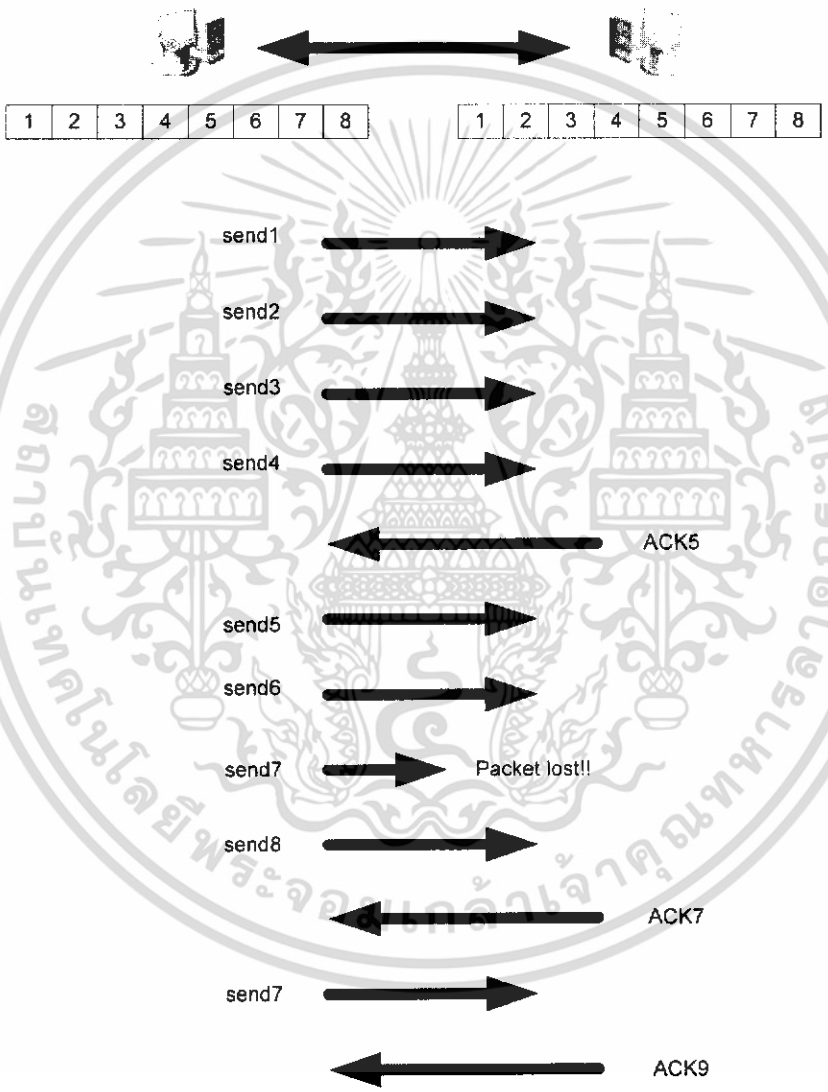


รูปที่ 2.4 แสดงกระบวนการสไลด์ดิ่งวินโดวส์

จากรูปที่ 2.4 แสดงกระบวนการสไลด์ดิ่งวินโดวส์แบบง่ายๆเพื่อช่วยควบคุมโฟลว์การรับส่งข้อมูล (Flow Control) สมมติถ้าขนาดของวินโดวส์(Window) เท่ากับ 1 นั่นหมายความว่าทุกๆ ทีซีพีเซ็กเมนต์ที่ถูกส่งออกไปจะต้องได้รับการตอบยืนยัน (เอซีเค) ว่าได้รับข้อมูลแล้ว โดยเครื่องคอมพิวเตอร์ต้นทางต้องรอคอยให้เครื่องคอมพิวเตอร์ปลายทางส่งเอซีเคสำหรับทีซีพีเซ็กเมนต์แรกก่อนถึงสมารถส่งทีซีพีเซ็กเมนต์ที่ 2 แล้วจึงค่อยส่ง ทีซีพีเซ็กเมนต์ที่ 3 ถัดไปได้เป็นเช่นนี้เรื่อยๆ แต่ในกรณีที่ขนาดของวินโดวส์เท่ากับ 4 นั่นหมายความว่า เครื่องต้นทางสามารถส่งทีซีพีเซ็กเมนต์ไปได้ทีเดียว 4 ชุด

ตามลำดับ และถึงจะหยุดรอคอยให้เครื่องคอมพิวเตอร์ปลายทางตอบรับกลับมา ว่าได้รับทีซีพีเซ็กเมนต์ข้างต้นทั้งหมดครบ 4 ชุดแล้ว ถึงจะส่งทีซีพีเซ็กเมนต์ถัดไปได้

ในการตอบกลับเพื่อยืนยันว่าได้รับข้อมูลจากต้นทางนี้ เครื่องปลายทางจะใช้วิธีการตอบรับ แบบฟอร์เวิร์ด แอ็คโนลีด (Forward Acknowledgement) คือ การบอกเครื่องคอมพิวเตอร์ต้นทางว่าได้รับทีซีพีเซ็กเมนต์ ล่าสุดถึงชุดที่ 4 และพร้อมที่จะรับทีซีพีเซ็กเมนต์ที่ 5 เป็นข้อมูลชุดถัดไป (ด้วยการส่ง เอซีเค 5 ไม่ใช่ เอซีเค 4) เมื่อเครื่องคอมพิวเตอร์ปลายทางได้รับเอซีเค จะทราบทันทีว่าเครื่องคอมพิวเตอร์ปลายทางได้รับทีซีพีเซ็กเมนต์ครบถ้วนถึงทีซีพีเซ็กเมนต์ที่ 4 แล้ว มันจะส่งทีซีพีเซ็กเมนต์ชุดที่ 5 ออกไปให้



รูปที่ 2.5 แสดงกระบวนการสไลด์ดิ้งวินโดวส์ขนาดเท่ากับ 4

จากรูปข้างต้น (รูปที่ 2.5) ขนาดของวินโดวส์เท่ากับ 4 เพราะฉะนั้นเครื่องต้นทางจึงส่งได้ 4 ชุด แล้วจึงค่อยรอคอยการตอบกลับ (เอซีเค) เมื่อปลายทางตอบกลับมาว่าได้รับทีซีพีเซ็กเมนต์ล่าสุดถึงชุดที่ 4 แล้ว และต้องการจะได้ทีซีพีเซ็กเมนต์ชุดที่ 5 ต่อไป (ด้วยการส่ง เอซีเค 5) เครื่องต้นทางจะส่งไปให้อีก 4 ชุด (คือทีซีพีเซ็กเมนต์ที่ 5 , 6 , 7 และ 8 ตามลำดับ) แต่ถ้าในกรณีทีซีพีเซ็กเมนต์ที่ 7 สูญหายระหว่างทาง เช่น เกิดความหนาแน่นของข้อมูลที่เข้าสู่เร้าเตอร์ (Router) หรือเร้าเตอร์มีบัฟเฟอร์ไม่เพียงพอ เร้าเตอร์จำเป็นต้องปฏิเสธทีซีพีเซ็กเมนต์บางส่วนด้วยกฎบางประการของ QoS ส่งผลให้ปลายทางไม่ได้รับทีซีพีเซ็กเมนต์ที่ 7

เมื่อทีซีพีเซ็กเมนต์ที่ 8 เดินทางมาถึงเครื่องคอมพิวเตอร์ปลายทาง จากนั้นจะตอบรับ (เอซีเค) กลับไปว่า เอซีเค 7 ซึ่งหมายความว่าเครื่องปลายทางได้รับทีซีพีเซ็กเมนต์ล่าสุดถึงชุดที่ 6 และต้องการได้รับทีซีพีเซ็กเมนต์ที่ 7 ต่อไป เมื่อเอซีเคถูกส่งกลับมาให้เครื่องคอมพิวเตอร์ต้นทาง ซึ่งจะทราบได้ทันทีว่าทีซีพีเซ็กเมนต์ชุดที่ 7 สูญหายไประหว่างทาง และก็จะส่งทีซีพีเซ็กเมนต์ ชุดที่ 7 กลับไปอีกครั้ง เมื่อเครื่องคอมพิวเตอร์ปลายทางได้รับทีซีพีเซ็กเมนต์และจะตรวจสอบว่าขณะนี้ได้รับทีซีพีเซ็กเมนต์ครบแล้วตั้งแต่ทีซีพีเซ็กเมนต์ที่ 5 – 8 แล้วจึงตอบเอซีเคกลับไปว่ามันได้รับทีซีพีเซ็กเมนต์ที่ 5 – 8 ครบถ้วนแล้ว และต้องการับทีซีพีเซ็กเมนต์ที่ 9 เป็นทีซีพีเซ็กเมนต์ถัดไปด้วยการส่ง เอซีเค 9 กลับมาให้

2.6 โปรโตคอลยูดีพี (UDP Protocol)

เป็นโปรโตคอลในระดับทรานสปอร์ตเลเยอร์ที่มีความแตกต่างไปจากโปรโตคอลทีซีพีเกือบทุกด้าน โปรโตคอลยูดีพีทำการส่งข้อมูลโดยไม่มีการสร้างคอนเน็กชันก่อนจึงเรียกว่าแบบ คอนเน็กชันเลส (Connectionless) คือ ไม่มีการยืนยันจากปลายทางว่าได้รับข้อมูลแล้วและไม่มีการจัดเตรียมขนาดของบัฟเฟอร์สำหรับการรับส่งข้อมูลรวมทั้งไม่มีการจัดลำดับของข้อมูลที่ได้รับ โดยให้หน้าที่ในการยืนยันว่าได้รับข้อมูลแล้วจะให้แอปพลิเคชันที่อยู่สูงขึ้นไปทำหน้าที่แทน และยังไม่สนใจส่วนของการควบคุมการรับส่งข้อมูลด้วย โดยหลักการทำงานข้างต้นนี้จึงทำให้ยูดีพีเป็นโปรโตคอลที่ไม่มีความน่าเชื่อถือ (Unreliable)

ประโยชน์สำคัญที่ได้รับจากการทำงานโดยใช้โปรโตคอลยูดีพีคือ ความรวดเร็ว ลดค่าใช้จ่ายในเรื่องทรัพยากรที่ต้องใช้ในการติดตามข้อมูล ตัวอย่าง การทำงานบนเน็ตเวิร์กที่ใช้โปรโตคอลยูดีพี ได้แก่ โปรโตคอล SNMP (Simple Network Management Protocol) อุปกรณ์เน็ตเวิร์กซึ่งทำหน้าที่เป็นเอสเอ็นเอ็มพีเอเจนท์ (SNMP Agent) จะส่งรายงานสถานะต่างๆกลับไปให้เครื่องเซิร์ฟเวอร์ที่ทำหน้าที่เป็นเอสเอ็นเอ็มพี เซิฟเวอร์ (SNMP Sever) โดยผ่านทางยูดีพีเนื่องจากต้องการความรวดเร็ว อีกทั้งข้อมูลมีจำนวนสถานะจำนวนมาก การที่จะต้องคอยสร้างคอนเน็กชันทุกครั้งเมื่อมีความต้องการส่งข้อมูจะเป็นเรื่องที่เสียเวลาและไม่จำเป็น หรือ อีกตัวอย่างหนึ่งคือ การโอนย้ายไฟล์ด้วยเอฟทีพี (FTP:File Transfer Protocol) จัดเป็นแบบคอนเน็กชันโอเรียนท์เต็ด คือใช้งานโปรโตคอลทีซีพีส่วนการย้ายไฟล์ด้วยโปรโตคอลไทร์วอลเอฟทีพี (TFTP) จะเป็นแบบคอนเน็กชันเลสคือใช้งานโปรโตคอลยูดีพี

2.7 โปรโตคอลไอพี (IP Protocol)

โปรโตคอลไอพีที่อยู่ในเน็ตเวิร์กเลเยอร์จะมีหน้าที่ 3 อย่างต่อไปนี้คือ

2.7.1 แอดเดรสซิ่ง (Addressing)

ทำหน้าที่ให้บริการในการติดตั้ง ลอจิคัลแอดเดรส (Logical Address)ให้กับเครื่องคอมพิวเตอร์ต่างๆที่ใช้โปรโตคอลไอพี เนื่องจากลอจิคัลแอดเดรสนี้จะไม่ได้ถูกกำหนดในลักษณะตายตัวหรือฝังมากับเน็ตเวิร์กการ์ดตั้งแต่แรก ดังนั้นผู้ออกแบบหรือบริหารระบบเครือข่ายจึงสามารถกำหนดและสามารถเปลี่ยนแปลงแก้ไขได้ตามใจชอบ ข้อดีของการมีลอจิคัลแอดเดรสหรือแอดเดรสในเน็ตเวิร์กเลเยอร์มีดังนี้

- ทำให้เราสามารถออกแบบระบบเน็ตเวิร์กได้ง่ายขึ้น
- ทำให้ระบบเน็ตเวิร์กสามารถขยายเพิ่มเติมได้โดยง่าย
- ทำให้การแก้ไขปัญหาทำได้โดยง่าย

2.7.2 แพ็กเก็จจิ้ง (Packaging)

เป็นการจัดเตรียมไอพีแพ็กเกต ให้อยู่ในสภาพที่พร้อมส่งไปยังเครื่องปลายทางโดยการนำ ทีซีพีเซ็กเมนต์ หรือ ยูดีพีเซ็กเมนต์ จากเลเยอร์ด้านบนมาบรรจุไว้ในฟิลด์ ดาต้า (Data) ของไอพีแพ็กเกต (หากขนาดของ เซ็กเมนต์ ใหญ่เกินกว่าจะส่งได้ภายในไอพีแพ็กเกตเดียว จะต้องแบ่งข้อมูลออกเป็นส่วนๆและทยอยส่งไปทีละแพ็กเกต) จากนั้นจึงใส่ค่าฟิลด์เดสติเนชันไอพีแอดเดรส (Destination IP Address) และซอร์สไอพีแอดเดรส (Source IP Address) ให้เป็นหมายเลขไอพีแอดเดรส (IP Address) ปลายทางและต้นทางตามลำดับ และต้องใส่ค่าฟิลด์โปรโตคอลนัมเบอร์ (Protocal Number) ลงไปด้วยเพื่อระบุว่าเลเยอร์ด้านบนที่ส่งข้อมูลลงมานั้นเป็นทีซีพีหรือยูดีพี(หมายเลข 6 สำหรับทีซีพีและ หมายเลข 17 สำหรับยูดีพี) ไอพีแพ็กเกตหนึ่งๆ บางครั้งถูกเรียกว่า ดาต้าแกรม (Datagram)

2.7.3 เร้าติ้ง (Routing)

เร้าติ้ง คือ การหาเส้นทางในการส่งแพ็กเก็จไปให้ถึงเครื่องคอมพิวเตอร์ปลายทาง หน้าที่หลักของการส่งแพ็กเก็จโดยโปรโตคอลไอพีคือการค้นหาเส้นทางส่งดีที่สุด แต่ไม่รับรองว่าข้อมูลจะถึงปลายทางหรือไม่ โดยจะปล่อยให้เป็นหน้าที่ของโปรโตคอลในระดับสูงกว่า (ทีซีพี) เป็นผู้รับรองให้

2.8 ตัวอย่างโฟลว์ของการส่งข้อมูลจากเลเยอร์บนไปยังเลเยอร์ล่าง

ยกตัวอย่างการใช้โปรแกรมอินเตอร์เน็ตเอ็กซ์โพลเลอร์ (Internet Explorer) บนวินโดวส์เพื่อขอเบราซ์เว็บเพจ (Webpage) จากเว็บเซิฟเวอร์(Web Sever) ในที่สุดก็จะได้เว็บเพจมาปรากฏบนหน้าจอ โดยมีขั้นตอนการทำงานดังนี้

ขั้นที่ 1 โปรโตคอลเว็บหรือ HTTP (Hypertext Transfer Protocol) เป็นโปรโตคอลที่อยู่ในระดับแอปพลิเคชันเลเยอร์ เมื่อมองที่ระดับแอปพลิเคชันเลเยอร์โปรโตคอลเอชทีทีพี (HTTP) จะมีการส่งเอชทีพีพี เมสเสจ (HTTP Message)เพื่อพูดคุยโต้ตอบกันระหว่างเว็บเบราเซอร์(Web browser) และเว็บเซิฟเวอร์ สิ่งสำคัญที่อยากจะให้เข้าใจก็คือ ในมุมมองแบบลอจิคัลนั้น เราจะดูประหนึ่งว่ามันรับส่งเอชทีที

พี เมสเสจ คุยกันโดยตรง แต่ในทางปฏิบัติจริงๆมันไม่ได้เป็นอย่างนั้นเพราะเว็บเบราเซอร์จะต้องใช้วินโดวส์ซอกเก็ตเอพีไอ (Windows Socket API) เพื่อสั่งให้ไดรเวอร์ (Driver)ของโปรโตคอล ทีซีพี/ไอพี ในวินโดวส์เป็นผู้นำพาเอชทีทีพี เมสเสจไปให้

ขั้นที่ 2 เมื่อไดรเวอร์ของโปรโตคอลทีซีพี/ไอพี ในวินโดวส์ได้รับคำสั่งจาก วิดโดวส์ซ็อกเก็ตเอพีไอ ให้เป็นผู้เปิดการสนทนากับเครื่องปลายทางมันก็จะเปิดคอนเน็กชันด้วยกระบวนการทรีเวย์แฮนด์เชค จากนั้นแอปพลิเคชัน (เว็บเบราเซอร์) จะต้องส่งข้อมูลของของมัน (เมสเสจของเอชทีทีพี) มาให้กับไดรเวอร์ของโปรโตคอลทีซีพี/ไอพี โดยข้อมูลจากแอปพลิเคชันจะต้องถูกห่อหุ้ม (เรียกว่าถูกเอ็นแคปซูลเลท (encapsulate) ลงไปในฟิลด์ดาต้าของโปรโตคอลทีซีพี)

เมื่อมองแบบลอจิคัล มันจะดูประหนึ่งว่า ไดรเวอร์ของโปรโตคอลทีซีพีที่ต้นทางรับส่งแพ็กเก็ตคุยกันโดยตรงกับไดร์เวอร์ของโปรโตคอลทีซีพี ที่ปลายทาง ซึ่งในแง่ของการทำความเข้าใจ เราจะมองในลักษณะนี้แต่ในแง่ของกระบวนการที่เกิดขึ้นจริง ไดรเวอ์ของโปรโตคอลในเลเยอร์ล่างก็คือโปรโตคอลไอพี เป็นผู้นำพาไปให้

ขั้นที่ 3 แพ็กเก็ตของโปรโตคอลทีซีพี จะถูกส่งลงให้กับโปรโตคอลไอพี (Internet Protocol) เป็นผู้จัดการต่อ โดบอาศัยหลักการเดิมคือว่า แพ็กเก็ตทีซีพีจะต้องถูกส่งลงมาเอ็นแคปซูลเลท ในส่วนข้อมูลของไอพีแพ็กเก็ต

โปรโตคอลไอพี เป็นโปรโตคอลในระดับเน็ตเวิร์กเลเยอร์ ทำหน้าที่เร้าแพ็กเก็ตหาเส้นทางและอื่นๆดังเคยกล่าวถึงไปแล้ว เช่นเดียวกับโปรโตคอลในเลเยอร์อื่นๆ ก็คือไอพีแพ็กเก็ตจะประกอบด้วย 2 ส่วนใหญ่ๆคือ ฟิล์ดส่วนหัว (Header) และฟิล์ดส่วนข้อมูล (Data),ฟิลด์ส่วนหัว (IP Header) จะประกอบด้วยหมายเลขไอพีของเครื่องต้นทาง , ไอพีแอดเดรสของเครื่องปลายทาง เป็นต้น ส่วนฟิลด์ข้อมูล (IP Data) จะเป็นที่บรรจุแพ็กเก็จของโปรโตคอลทีซีพี (ซึ่งเป็นโปรโตคอลระดับบน) หรือกล่าวในทางกลับกันว่า ทีซีพีแพ็กเก็ตจะถูกเอ็นแคปซูลเลทลงมาในฟิลด์ข้อมูลของไอพีแพ็กเกต

ในขั้นที่ 4 คือเป็นหน้าที่ของโปรโตคอลไอพี ที่เครื่องต้นทางจะต้องส่งแพ็กเก็จไปให้โปรโตคอลไอพีที่เครื่อง ปลายทาง ในขั้นนี้จะมีการคำนวณหาเส้นทางก่อนที่จะส่งไปยังเครื่องปลายทาง เช่น เครื่องปลายทางอยู่ในซับเน็ต(Subnet) เดียวกันหรือไม่ ถ้าอยู่ในซับเน็ตเดียวกัน มันจะสามารถส่งแพ็กเก็ตตรงไปยังเครื่องปลายทางได้เลย แต่ถ้าหากไม่อยู่ มันก็จะต้องส่งแพ็กเก็ตไปให้เร้าเตอร์ก่อนเพื่อให้เร้าเตอร์ส่งแพ็กเก็ตข้ามไปซับเน็ตปลายทางให้

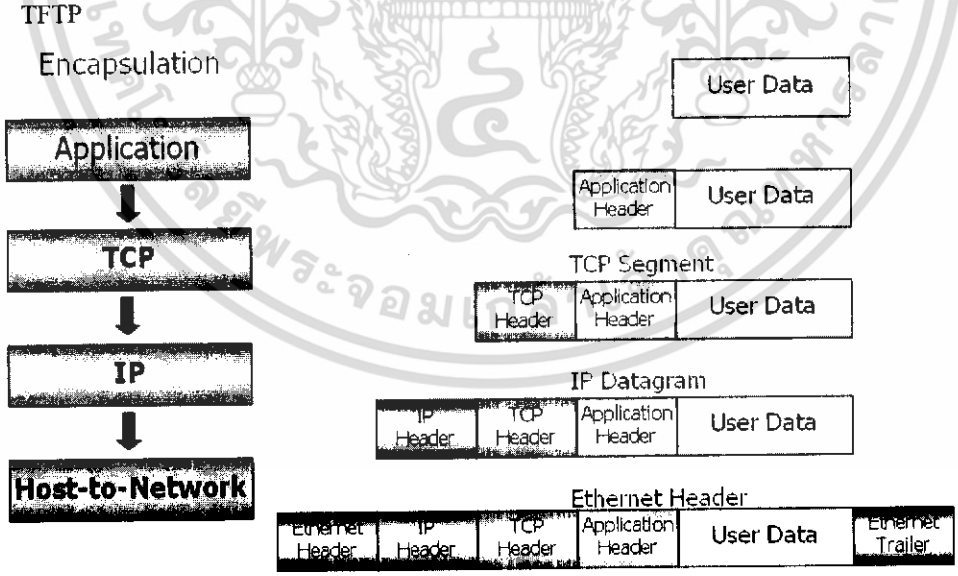
ในลักษณะคล้ายกันกับที่ผ่านมา กล่าวคือ เมื่อมองแบบลอจิคัล มันจะดูประหนึ่งว่า ไดรเวอร์ของโปรโตคอลไอพี ที่ต้นทางรับส่งแพ็กเก็ตคุยกันกับไดรเวอร์ของโปรโตคอลไอพีที่เครื่องปลายทางโดยตรง ซึ่งในแง่ของการทำความเข้าใจ เราจะมองในลักษณะนี้ แต่ในแง่ของกระบวนการที่เกิดขึ้นจริง ไดรเวอร์ของโปรโตคอลไอพีจะต้องส่งแพ็กเก็ตของมันลงมาให้ไดรเวอร์ของโปรโตคอลในเลเยอร์ต่ำกว่าเป็นผู้นำพาไปให้ แล้วในเลเยอร์ของโปรโตคอลต่ำกว่าคือใคร คำตอบก็คือ ขึ้นกับว่าเราทำงานอยู่บนฟิสิคัลเน็ตเวิร์ก (Physical Network)ประเภทไหน เป็น อีเทอร์เน็ต, โทเกนริงค์ หรือ เอฟดีดีไอ (FDDI) หรือ เอทีเอ็ม (ATM) โดยทั่วไป ในปัจจุบันฟิสิคัลเน็ตเวิร์ก ที่เรานิยมเลือกใช้ก็คือแบบอีเทอร์เน็ต สมมติว่าในที่นี้เป็นอีเทอร์เน็ต

ขั้นที่ 5 ไดรเวอร์ของโปรโตคอลไอพี ก็ส่งไอพีแพ็กเก็ต มาให้ไดรเวอร์ของเน็ตเวิร์กการ์ด (Network Card) ที่เชื่อมต่ออยู่บนเน็ตเวิร์กอีเทอร์เน็ต แล้วปล่อยให้เป็นหน้าที่ของเน็ตเวิร์กการ์ดในการส่งเฟรม (frame) ข้อมูลไปให้ถึงเน็ตเวิร์กการ์ดของเครื่องปลายทาง

- อีเทอร์เน็ตเฟรม (Ethernet Frame) เฟรมหนึ่งจะประกอบด้วยสองส่วนใหญ่ๆเหมือนกัน ก็คือ ส่วนหัว (Ethernet Header) และส่วนข้อมูล (Ethernet Data) ส่วนหัวหรือเฮดเดอร์จะประกอบด้วย แอดเดรสของเน็ตเวิร์กการ์ดต้นทางและแอดเดรสของเน็ตเวิร์กการ์ดปลายทาง โดยฟิล์ดในส่วนข้อมูล (Ethernet Data) จะเป็นที่บรรจุหรือ เอ็นแคปซูลเลทไอพีแพ็กเก็ต เอาไว้ภายใน หรือกล่าวอีกอย่างหนึ่งก็คือ ไอพีแพ็กเก็ตจะถูกเอ็นแคปซูลเลทลงมาเก็บในส่วนข้อมูล (Ethernet Data) ของอีเทอร์เน็ตเฟรมก่อน

จากการอธิบายอีเทอร์เน็ตเฟรมข้างต้น แอดเดรสของเน็ตเวิร์กการ์ดเรานิยมเรียกกันว่าแม็คแอดเดรส (MAC Address (MAC = Media Address Control)) แอดเดรสประเภทนี้จะฝังมาพร้อมกับเน็ตเวิร์กการ์ดทันที ตั้งแต่การ์ดถูกผลิตออกมาจากโรงงานผู้ผลิต ประกอบด้วยตัวเลข 6 ไบต์ โดย 3 ไบต์หน้าทางองค์กรกลางคือ IEEE ได้กำหนดให้กับผู้ผลิตแต่ละราย และ 3 ไบต์สุดท้ายนั้น ทางผู้ผลิตสามารถกำหนดได้เอง อย่างเช่น 0A0B1C129A4F

การส่งอีเทอร์เน็ตเฟรมไปยังเครื่องปลายทาง จะไม่สามารถทำได้ถ้าเราทราบเพียงหมายเลขไอพีแอดเดรส ของเครื่องปลายทาง แต่ไม่ทราบแม็คแอดเดรส ดังนั้นจึงต้องมีอีกกระบวนการค้นหาแม็คแอดเดรสของเครื่องปลายทางซึ่งเรียกกระบวนการนี้ว่า ARP (Address Resolution Protocol)



รูปที่ 2.6 แสดงไดอะแกรมสรุปการเอ็นแคปซูลเลทข้อมูลลงมาตามลำดับชั้น

เมื่ออีเทอร์เน็ตเฟรมถูกส่งไปยังเครื่องคอมพิวเตอร์ปลายทาง เน็ตเวิร์กการ์ดที่เครื่องคอมพิวเตอร์ปลายทางจะรับเอาเฟรมเข้าไปเก็บในบัฟเฟอร์ จากนั้นจะสำรวจในฟิลด์ Type ที่อยู่ในอีเทอร์เน็ตเฟรมว่าอ้างอิงถึงโปรโตคอลระดับบนตัวใด ซึ่งถ้ามีไอพีแพ็กเกตอยู่ภายใน ค่าฟิลด์ Typeจะเท่ากับ 0x0800 อีเทอร์เน็ตการ์ดจะส่งข้อมูลต่อไปให้โปรโตคอลไอพี เมื่อโปรโตคอลไอพีได้รับเฟรมข้อมูลจะตรวจสอบฟิลด์โปรโตคอลนัมเบอร์ (Protocol Number) หากพบว่าเท่ากับ 6 จะส่งต่อไปให้กับโปรโตคอลทีซีพีหากพบว่าเท่ากับ 17 ก็จะส่งต่อให้กับโปรโตคอลยูดีพี ท้ายที่สุด เมื่อโปรโตคอล ทีซีพี/ยูดีพี ได้รับเฟรมข้อมูลก็จะอ่านค่าหมายเลขพอร์ตปลายทาง (Destination Port Number) และส่งต่อไปให้กับแอปพลิเคชั่นหรือโปรเซสที่เหมาะสมต่อไป

2.9 จุดเชื่อมต่อการให้บริการ (Service Access Point)

จุดเชื่อมต่อการให้บริการเป็นค่าฟิลด์หนึ่งของโปรโตคอลในแต่ละเลเยอร์ (สมมติเป็นเลเยอร์ที่ N) ใช้ในการเก็บค่าสัญลักษณ์บางอย่าง เพื่อบ่งบอกให้ทราบว่าข้อมูลที่ถูกเอนแคปซูลเลทอยู่ในแพ็กเกตหรือ เฟรมของมันเป็นข้อมูลที่ถูกส่งมายังโปรโตคอลอะไรในเลเยอร์ที่สูงกว่า (ของเลเยอร์ที่ N+1)

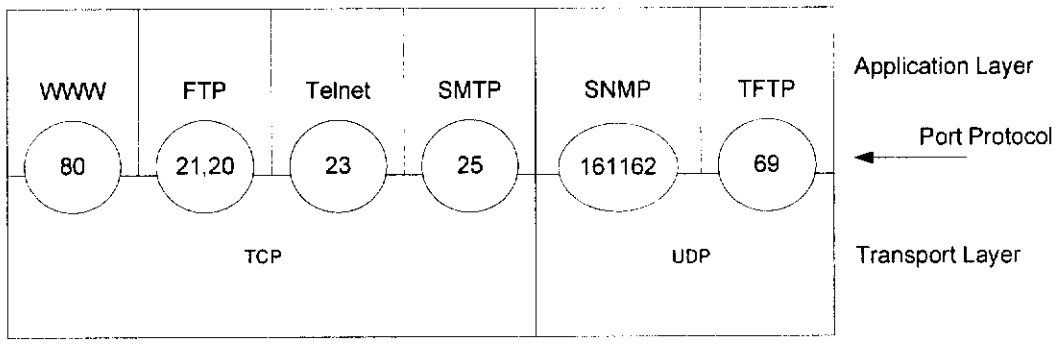
เซอร์วิสแอคเซสพ็อย้ท (Service access point) ที่เรารู้จักกันดี ถ้ามองจากเลเยอร์บนลงมาได้แก่ หมายเลขพอร์ต (Port Number) ในทีซีพีหรือยูดีพีเซ็กเมนต์ , ค่าโปรโตคอลไทพ์ (Protocol Type) ในไอพีแพ็กเก็ต และค่า โปรโตคอลไทพ์ในอีเทอร์เน็ตเฟรม

2.10 หมายเลขพอร์ต (Port Number)

เป็นค่าตัวเลขที่บ่งบอกให้ทราบถึงชื่อของแอปพลิเคชัน หรือชื่อของเซอร์วิสที่ทำงานอยู่ในระดับแอปพลิเคชันเลเยอร์ หมายเลขพอร์ตจะเก็บอยู่ในฟิลด์ทีซีพีซอร์สพอร์ท(TCP Source Port)และทีซีพีเดสติเนชันพอร์ท (TCP Destination Port) ซึ่งอยู่ในส่วนของทีซีพีเฮดเดอร์

ตัวอย่างเช่น หมายเลขพอร์ตของเว็บเซิร์ฟเวอร์เป็น 80 ดังนั้น เมื่อเครื่องไคลเอนต์ต้องการติดต่อกับเว็บเซิร์ฟเวอร์ ไดรเวอร์ของโปรโตคอลทีซีพีที่เครื่องไคลเอนต์จะเซตค่าหมายเลขพอร์ตในฟิลด์ ทีซีพีเดสติเนชันพอร์ทให้เป็น 80 เพื่อบ่งบอกให้ไดรเวอร์ของโปรโตคอลทีซีพี ที่เครื่องเซิร์ฟเวอร์ปลายทางทราบว่ามันต้องการติดต่อกับเว็บเซอร์วิส หลังจากที่ไดรเวอร์ของโปรโตคอลทีซีพี ทราบแล้วมันจะส่งข้อมูลที่อยู่ในฟิลด์ดาต้าของแพ็กเกตขึ้นไปให้เว็บเซิร์ฟเวอร์ที่รันอยู่ได้อย่างถูกต้อง ส่วนค่าทีซีพีซอร์สพอร์ทจะเป็นหมายเลขพอร์ตในฝั่งไคลเอนต์ ซึ่งปกติเราจะไม่สนใจ ดังนั้นเราจะถูกสุ่มขึ้นมาเป็นหมายเลขใดก็ได้ที่มากกว่า 1024

โดยสรุปก็คือ หมายเลขพอร์ตจึงถูกใช้เพื่อบ่งบอกให้ไดรเวอร์ของโปรโตคอลทีซีพีทราบว่าควรจะต้องส่งข้อมูลที่อยู่ในฟิลด์ดาต้าขึ้นไปให้กับแอปพลิเคชันเลเยอร์ดังแสดงในรูปที่ 2.7



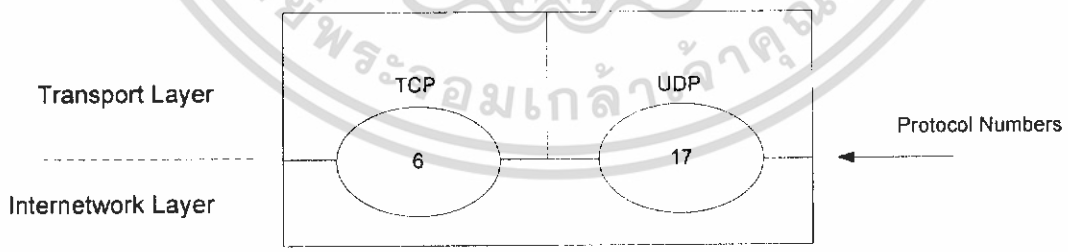
รูปที่ 2.7 หมายเลขพอร์ตที่ใช้กับแอพพลิเคชันต่างๆ

2.11 เวลโนนพอร์ท (Well Known Port)

Well Know Port หมายถึง หมายเลขพอร์ตที่ทางองค์กรกลางที่ชื่อ IETF (Internet Engineering Task Force) เป็นผู้กำหนดให้กับเซอร์วิสที่รู้จักกันดีและนิยมใช้กันแพร่หลายในเครือข่ายอินทราเน็ต (Intranet) / อินเทอร์เน็ต ค่าของ Well Known Port จะอยู่ระหว่าง 1 ถึง 1,024 (ดังนั้น แอปพลิเคชันหรือเซอร์วิสอื่นที่เขียนขึ้นมาทำงานบนโปรโตคอลทีซีพี/ไอพี ด้วยจึงสามารถตั้งค่าพอร์ตของตนเองได้ เป็นค่าที่มากกว่า 1,024 ขึ้นไป) ตัวอย่างของเวลโนนพอร์ทเช่น พอร์ตหมายเลข 80 เป็นของ WWW, พอร์ตหมายเลข 25 เป็นของเอสทีเอ็มพี เป็นต้น หากต้องการทราบค่าของเวลโนนพอร์ท ต่างๆที่นิยมใช้กันให้ท่านผู้อ่านทดลองเปิดดูได้จากไฟล์ชื่อเซอร์วิส (SERVICES) โดยในวินโดวส์ NT 4.0 , 2000, XP, และ 2003 ไฟล์ SERVICES จะอยู่ที่โฟลเดอร์ C:\<winnt_root>\system32\driver\etc

2.12 หมายเลขโปรโตคอล (Protocol Number)

เป็นฟิลด์หนึ่งที่อยู่ในเฮดเดอร์ของไอพีแพ็กเก็ต ที่ใช้บ่งบอกว่าโปรโตคอลในเลเยอร์บนคือโปรโตคอลทีซีพีหรือ โปรโตคอลยูดีพี ดังแสดงในรูป 2.8



รูปที่ 2.8 แสดงฟิลด์เฮดเดอร์ที่บอกโปรโตคอลในเลเยอร์ว่าเป็นทีซีพีหรือยูดีพี

2.13 โปรโตคอลเออาร์พี ARP (Address Resolution Protocol)

เออาร์พี (ARP) เป็นโปรโตคอลที่ใช้ในการค้นหาหมายเลขแม็คแอดเดรสของเครื่องโฮสต์ปลายทาง ที่เครื่องโฮสต์ต้นทางต้องการส่งแพ็กเก็ตไปหา เนื่องจากการทราบหมายเลขไอพีแอดเดรสของโฮสต์ปลายทางเพียงอย่างเดียวนั้นยังไม่เพียงพอ ในการส่งไอพีแพ็กเก็ต ปยังโฮสต์ปลายทางนั้น เครื่องต้นทางต้องเอ็นแคปซูลเลท (ห่อหุ้ม) ไอพีแพ็กเก็ตลงมาในอีเทอร์เน็ตเฟรมก่อน แล้วค่อยให้เน็ตเวิร์กการ์ดเป็นผู้ส่งเฟรมนั้นผ่านสายเคเบิลไปยังเน็ตเวิร์กการ์ดปลายทาง เมื่อเน็ตเวิร์กการ์ดของเครื่องปลายทางรับเอาเฟรมขึ้นมาแล้ว จึงค่อยดีเอ็นแคปซูลเลท (de-encapsulate) เฟรมเพื่อแยกเอาไอพีแพ็กเก็ตออกมาแล้วส่งไปให้ไดรเวอร์ของโปรโตคอลทีซีพี/ไอพีอีกที

ในอีเทอร์เน็ตเฟรมนั้น ส่วนหัวของเฟรมจะประกอบด้วยหมายเลขแม็คแอดเดรส ของเน็ตเวิร์กการ์ดต้นทาง และแม็คแอดเดรสของเน็ตเวิร์กการ์ดปลายทาง และในส่วนข้อมูลจะเป็นไอพีแพ็กเก็ตที่ถูกเอ็นแคปซูลเลทลงมา หมายเลขแม็คแอดเดรสของเครื่องคอมพิวเตอร์ต้นทางนั้นเครื่องต้นทางต้องทราบอยู่แล้วแต่หมายเลขแม็คแอดเดรสของเครื่องคอมพิวเตอร์ปลายทางนั้นยังไม่ทราบ ดังนั้นเครื่องคอมพิวเตอร์ต้นทางจึงต้องทำเออาร์พี ก่อนเพื่อหาว่าเครื่องคอมพิวเตอร์ปลายทางที่มีหมายเลขไอพีแอดเดรสเบอร์นี้มีหมายเลขแม็คแอดเดรสเป็นเบอร์อะไร จึงจะสามารถส่งอีเทอร์เน็ตเฟรมออกไปได้

ดังนั้นทุกครั้งที่เครื่องคอมพิวเตอร์ที่ใช้โปรโตคอลทีซีพี/ไอพีต้องการติดต่อกันจะต้องมีกระบวนการเออาร์พีเกิดขึ้นก่อนทุกครั้งเสมอ เพียงแต่กระบวนการตรงนี้เรามองไม่เห็นเท่านั้นเอง

โปรโตคอลเออาร์พีทำงานอยู่ระหว่างอินเตอร์เน็ตเวิร์กเลเยอร์กับเน็ตเวิร์กแอคเซสเลเยอร์โดยจะถือว่าเป็นโปรดตคอลเสริมอันหนึ่งของโปรโตคอลไอพี โปรโตคอลเออาร์พีจะมีแพ็กเก็ตเออาร์พี เป็นของตนเอง ได้แก่ เออาร์พีรีเควส (ARP Request) และ เออาร์พีรีไพน์ (ARP Reply) โดยแพ็กเก็ตทั้งสองประเภทนี้จะถูกเอ็นแคปซูลเลทลงไปในอีเทอร์เน็ตเฟรมก่อน แล้วค่อยส่งออกไปเพื่อค้นหาแม็คแอดเดรสที่ต้องการ

2.13.1 หลักการทำงานของเออาร์พี

การทำเออาร์พีจะเกิดขึ้น โดยเครื่องคอมพิวเตอร์ต้นทางเป็นผู้ส่งเฟรมพิเศษที่เรียกว่าเออาร์พีรีเควส โดยส่งแบบบรอดคาสต์ (Broadcast) ออกไปยังเครื่องคอมพิวเตอร์ทุกๆเครื่องในเน็ตเวิร์กเพื่อถามว่ามีเครื่องไหนมีข้อมูลแม็คแอดเดรสของไอพีแอดเดรสเบอร์นี้ไหม ภายในเฟรมเออาร์พีรีเควสประกอบไปด้วยฟิลด์ต่างๆดังนี้

- **ซอร์สไอพีแอดเดรส (Source IP Address)** : คือหมายเลขไอพีแอดเดรสของเครื่องต้นทางเองซึ่งทราบอยู่แล้ว
- **ซอร์สฮาร์ดแวร์แอดเดรส (Source Hardware Address) :** คือหมายเลขแม็คแอดเดรสของเครื่องต้นทางเองซึ่งทราบดีอยู่แล้ว
- **ทาร์เก็ตไอพีแอดเดรส (Target IP Address) :** คือหมายเลขไอพีแอดเดรสของเครื่องปลายทางที่ต้องการติดต่อ ซึ่งทราบอยู่แล้ว

- **ทาร์เก็ตฮาร์ดแวร์แอดเดรส (Target Hardware Address) :** คือหมายเลขแม็คแอดเดรสของเครื่องปลายทาง ซึ่งขณะนี้ยังไม่ทราบ จึงใส่เป็น 00000000000 ไว้ก่อน

แพ็กเก็ตเออาร์พีรีเควสจะถูกเอ็นแคปซูลเลทลงมาในอีเทอร์เน็ตเฟรมก่อน แล้วจึงค่อยถูกส่งผ่านสายเคเบิลออกไป แต่เพื่อเป็นการบังคับให้ทุกๆเครื่องรับอีเทอร์เน็ตเฟรมไปประมวลผล โปรโตคอลเออาร์พีจึงได้เซตหมายเลข เดสติเนชันแม็คแอดเดรส (Destination MAC Adderss) ของอีเทอร์เน็ตเฟรมที่ทำหน้าที่ห่อหุ้มแพ็กเก็ตเออาร์พีรีเควสไว้ให้เป็นบอร์ดคาสต์แอดเดรสในระดับ MAC Layer คือ FFFFFFFFFFFF โดย Source MAC Address เป็นแม็คแอดเดรสของมันเอง

มีข้อกำหนดว่า เน็ตเวิร์กการ์ดจะรับเอาเฟรมเข้าไปประมวลผลต่อ ก็ต่อเมื่อ 2 เงื่อนไขต่อไปนี้ คือ

1. เดสติเนชันแม็คแอดเดรส ตรงกับแม็คแอดเดรสของมันเอง หรือ
2. เดสติเนชันแม็คแอดเดรส เป็นบรอดคลาสต์แอดเดรสคือ FFFFFFFFFFFF

ดังนั้น ในขณะนี้ เครื่องทุกเครื่องจะรับเอาเฟรมดังกล่าวไปพิจารณา แล้วดูว่าตัวเองได้ตั้งหมายเลขไอพีแอดเดรสไว้ตรงกับหมายเลขทาร์เก็ตไอพีแอดเดรสที่สอบถามมาในเออาร์พีรีเควสหรือไม่ หากใช่ มันก็จะส่งคำตอบกลับไปให้ว่ามีหมายเลขแม็คแอดเดรสเป็นเบอร์อะไร คำตอบจะถูกส่งไปในแพ็กเก็ตเออาร์พีรีไพน์ ส่วนเครื่องที่มีไอพีแอดเดรสไม่ตรงกับทาร์เก็ตไอพีแอดเดรสที่เซตมาในเออาร์พีรีเควสก็จะไม่สนใจ และทิ้งเออาร์พีรีเควสทิ้งไปในที่สุด

2.13.2 เออาร์พีแคช (ARP Cache)

หลังจากที่โฮสต์แต่ละเครื่องได้แม็คแอดเดรสของเครื่องปลายทางมาแล้ว เครื่องต้นทางที่ได้รับแม็คแอดเดรส จะเก็บความสัมพันธ์ระหว่างหมายเลขแม็คแอดเดรสของเครื่องปลายทางกับหมายเลขไอพีแอดเดรสของเครื่องปลายทางนั้นๆไว้ในหน่วยความจำ ซึ่งถูกกันไว้เป็นแคชที่เรียกว่าเออาร์พีแคช (ARP Cache) เผื่อว่าในอนาคตหากต้องการติดตั้งกับเครื่องปลายทางที่มีหมายเลขไอพีแอดเดรสเป็นเบอร์นั้นๆอีก จะได้ไม่ต้องทำเออาร์พีใหม่อีกครั้ง เพราะสามารถหาจากเออาร์พีแคชได้เลย ในวินโดวส์ทุกรุ่นเราสามารถวิว(View) ดูเออาร์พีแคช ได้โดยการใช้คำสั่ง arp – a

2.13.3 การทำงานของเออาร์พี ขึ้นกับซับเน็ตแอดเดรสปลายทาง

เมื่อโฮสต์ต้นทางต้องการส่งข้อมูลไปหาโฮสต์ปลายทาง ขั้นตอนสำคัญขั้นแรกที่เกิดขึ้นก็คือจะต้องมีการคำนวณหาก่อนว่า โฮสต์ปลายทางอยู่ในซับเน็ตโฮสต์เดียวกันกับโฮสต์ต้นทางหรือไม่ ขั้นตอนถัดไปจะเกี่ยวข้องกับโปรโตคอลเออาร์พีดังนี้

1. ถ้าอยู่ในซับเน็ตเดียวกันโฮสต์ต้นทางสามารถส่งไอพีแพ็กเก็ตไปหาได้โดยตรง โดยก่อนการส่งไอพีแพ็กเก็ตโอสต์ต้นทางต้องมีการหาแม็คแอดเดรสของโอสต์ปลายทางให้ได้ก่อน อาจหาพบจากเออาร์พีแคช (ถ้าได้หาไว้แล้วก่อนหน้านี้) หรือต้องส่งเออาร์พีรีเควส บรอดคาสต์ออกไปอย่างที่ได้อธิบายไปข้างต้น

2. ถ้าอยู่ต่างซับเน็ตกันโฮสต์ต้นทางจะไม่สามารถส่งแพ็กเก็ตไปยังโอสต์ปลายทางได้โดยตรงแต่จะต้องส่งแพ็กเก็ตไปหาเร้าเตอร์ก่อน แล้วจึงให้เร้าเตอร์เป็นผู้ส่งผ่านแพ็กเก็ตไปให้ถึงโฮสต์ปลายทางอีก

ครั้งหนึ่ง ซึ่งโฮสต์ต้นทางเองก็ต้องมีการตรวจสอบจากเร้าติ้งเทเบิล(Routing Table) อีกเช่นกันว่าจะส่งไปให้เร้าเตอร์ตัวไหนดี ส่วนใหญ่แล้วในวินโดวส์คลเอนต์เรามักเซตให้ส่งไปให้เร้าเตอร์ที่ทำหน้าที่เป็น ดีฟอลต์เกตเวย์ (default gateway) ก่อนที่มันจะส่งไอพีแพ็กเก็ตไปหาดีฟอลต์เกตเวย์ได้ โฮสต์ต้นทางก็จะต้องทำเออาร์พีเพื่อหาแม็คแอดเดรสของเร้าเตอร์นั้นๆก่อน สรุปได้ว่า

- ถ้าโฮสต์ปลายทางอยู่ใน Subnet เดียวกันโฮสต์ต้นทางจะทำเออาร์พีเพื่อหาแม็คแอดเดรสของโฮสต์ปลายทางโดยตรง
- ถ้าโฮสต์ปลายทางอยู่ต่าง Subnet กัน โฮสต์ต้นทางจะทำเออาร์พีเพื่อหาแม็คแอดเดรสของดีฟอลต์เกตเวย์แทน

2.13.4 ข้อสังเกตเกี่ยวกับการทำเออาร์พีของเร้าเตอร์

1. เมื่อเร้าเตอร์ต้องการส่งไอพีแพ็กเก็ตไปยังเครื่องปลายทาง เร้าเตอร์เองก็ต้องทำกระบวนการเออาร์พีเช่นเดียวกัน เพื่อหาแม็คแอดเดรสของเครื่องโฮสต์ปลายทางก่อน แล้วจึงค่อย เอ็นแคปซูลเลทไอพีแพ็กเก็ตลงไปในอีเทอร์เน็ตเวิร์กไปให้โฮสต์ปลายทาง

2. ถ้าเร้าเตอร์จำเป็นต้องส่งไอพีแพ็กเก็ตไปให้เร้าเตอร์ตัวอื่นก่อนอีกช่วงหนึ่ง เร้าเตอร์ตัวแรกก็ต้องทำเออาร์พีเหมือนกัน เพื่อหาแม็คแอดเดรส ของเร้าเตอร์ตัวถัดไปด้วย แล้วจึงค่อยเอ็นแคปซูลเลท ไอพีแพ็กเก็ตลงไปในอีเทอร์เน็ตเฟรมแล้วส่งผ่านสายเน็ตเวิร์กไปให้เร้าเตอร์ตัวถัดไป

ภาพรวมของการสื่อสารระหว่างเครื่องคอมพิวเตอร์สองเครื่องที่ใช้โปรโตคอลทีซีพี/ไอพีจัดว่าเป็นการสื่อสารแบบเอนด์ทูเอนด์ (End to End) โดยแต่ละฝ่ายจะอ้างถึงฝ่ายตรงข้ามด้วยหมายเลขไอพีแอดเดรสแต่ในขณะที่แพ็กเก็ตเดินทางจากจุด (hop) หนึ่งไปยังจุด (hop) หนึ่งในระหว่างทาง (Hop by Hop) อย่างเช่น จากโฮสต์ต้นทางกับอุปกรณ์เร้าเตอร์ หรือเร้าเตอร์ตัวแรกกับเร้าเตอร์ตัวที่สอง หรือเร้าเตอร์ตัวที่สองกับโฮสต์ตัวสุดท้าย เครื่องคอมพิวเตอร์และอุปกรณ์เร้าเตอร์ระหว่างทางเหล่านั้นจำเป็นต้องทราบหมายเลขแม็คแอดเดรสจริงๆ ของ Hop ปลายทางด้าน ดังนั้นกระบวนการ เออาร์พีจึงเข้ามาช่วยในการค้นหาแม็คแอดเดรสของการสื่อสารแต่ละช่วงให้จนกว่าจะถึงปลายทาง

นอกจากเออาร์พีแล้ว ยังมีอีกโปรโตคอลหนึ่งที่ชื่อว่ารีเวอร์สเออาร์พี(Reverse ARP (RARP)) โปรโตคอลนี้ทำหน้าที่ในการค้นหาว่าแม็คแอดเดรสที่มีอยู่สมควรจะจับคู่กับหมายเลขไอพีแอดเดรสเบอร์ใด โปรโตคอลนี้เป็นโปรโตคอลเก่าที่มักถูกใช้โดยระบบปฏิบัติการบางประเภทที่มีการร้องขอหมายเลขไอพีแอดเดรสขณะกำลังบุต (Boot) ระบบจะส่งเมสเสจไปบอกอาร์เออาร์พีเซิฟเวอร์ RARP Server ว่าเครื่องของตนมีแม็คแอดเดรสเบอร์นี้และให้ RARP Server ตรวจเช็คตาราง RARP Table ว่าหมายเลขแม็คแอดเดรสเบอร์นี้สมควรได้รับการจัดสรรหมายเลขไอพีแอดเดรสเบอร์อะไร

2.14 โปรโตคอลไอซีเอ็มพี (ICMP (Internet Control Message Protocol))

ไอซีเอ็มพีเป็นโปรโตคอลที่ใช้ในการตรวจสอบและรายงานสถานภาพของดาต้าแกรม โปรโตคอลนี้ทำงานในระดับเน็ตเวิร์กเลเยอร์ เช่นเดียวกับ ไอพี ในกรณีที่เกิดปัญหากับดาต้าแกรม เช่น เร้าเตอร์ไม่สามารถส่งดาต้าแกรมไปถึงปลายทางได้ไอซีเอ็มพี จะถูกส่งออกไปยังโฮสต์ต้นทางเพื่อรายงาน

ข้อผิดพลาด ที่เกิดขึ้น อย่างไรก็ดี ไม่มีสามารถรับรองได้ว่าไอซีเอ็มพีเมสเสจ (ICMP Message) ที่ส่งไปนั้น จะถึงผู้รับจริงหรือไม่ หากมีการส่งดาต้าแกรมออกไปแล้วไม่มีไอซีเอ็มพีเมสเสจฟ้อง เออเรอร์ กลับมาก็แปลความหมายได้สองกรณีคือ ข้อมูลถูกส่งไปถึงปลายทางอย่างเรียบร้อย หรืออาจจะมีปัญหาในการสื่อสารทั้งการส่งดาต้าแกรม และไอซีเอ็มพีเมสเสจที่ส่งกลับมาก็มีปัญหาระว่างทางก็ได้ไอซีเอ็มพีจึงเป็นโปรโตคอลที่ไม่มีความน่าเชื่อถือ (unreliable) ซึ่งจะเป็นหน้าที่ของ โปรโตคอลในระดับสูงกว่าเน็ตเวิร์กเลเยอร์ในการจัดการให้การสื่อสารนั้นๆ มีความน่าเชื่อถือ

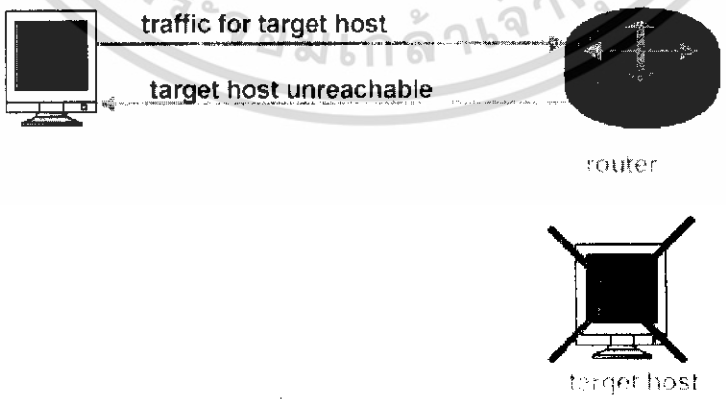
2.14.1 การใช้งานไอซีเอ็มพี

โดยทั่วไปไอซีเอ็มพี มีการใช้งานในสองลักษณะคือ

- **Query** ใช้สอบถามสถานะระหว่างกัน ในรูปที่ 2.9 เป็นการส่งเอคโครีเควส (Echo request) เพื่อถามสถานะของปลายทาง ซึ่งโฮสปลายทางอยู่ในสถานะปกติ สามารถทำการสื่อสารได้จะส่งเอคโค่รีไพน์ (Echo Reply) กลับมา

รูปที่ 2.9 การใช้งานโปรโตคอลไอซีเอ็มพีเพื่อสอบถามสถานะระหว่างกัน

- **Error Report** ใช้รายงานข้อผิดพลาดที่เกิดขึ้น เช่น หากไม่สามารถส่งดาต้าแกรมไปถึงปลายทางได้ เราเตอร์จะส่งไอซีเอ็มพีเมสเสจโฮสต์ อันรีชเอเบิ้ล (ICMP Message Host Unreachable) กลับมารายงานโฮสต้นทาง (รูปที่ 2.10)



รูปที่ 2.10 การใช้งานโปรโตคอลไอซีเอ็มพีเพื่อรายงานข้อผิดพลาดที่เกิดขึ้น

การรายงานข้อผิดพลาดของ ไอซีเอ็มพีนั้น จะ ไม่รายงานข้อผิดพลาดในบางกรณีคือ

- เกิดความผิดพลาดในการส่งไอซีเอ็มพีเสียเอง
- การส่งข้อมูลเป็นการส่งข้อมูลแบบมัลติคาสต์(Multicast)หรือบรอดคาสต์
- มีการแบ่งดาต้าแกรมออกเป็นส่วนย่อยๆ เรียกว่าแฟรกเมนต์ ในกรณีนี้ จะส่งไอซีเอ็มพีเมสเสจสำหรับแฟรกเมนต์แรกเพียงครั้งเดียว ไม่ต้องแจ้งกลับทุกแฟรกเมนต์ เพราะถือว่าเป็นดาต้าแกรมเดียวกัน
- แอดเดรสต้นทางไม่ได้เจาะจงโฮส เช่น แอดเดรส ที่เป็น 0 ทั้งหมดแอดเดรสที่เป็น loopback หรือแอดเดรสที่เป็นแบบมัลติคาสต์หรือบรอดคาสต์

2.15 ไอพีฟิวเตอร์ (IP Filter)

โดยทั่วไปไอพีฟิวเตอร์จะถูกใช้ในการป้องกันการเข้าถึงคอมพิวเตอร์จากไอพีแอดเดรส อื่นๆ โดยในการกรองเฟรมข้อมูลไอพี นั้นจำเป็นที่จะต้องทราบถึงรูปแบบของไอพีเฮดเดอร์ (IP Header) เพื่อจะนำมาดีเอ็นแคปซูลเลท ส่วนที่เป็นซอร์สไอพีหรือเดสติเนชันไอพีมาใช้ในการตัดสินใจว่าจะกรองทิ้งหรือไม่ โดยรูปแบบของไอพีเฮดเดอร์จะมีลักษณะดังรูปที่ 2.11

ตำแหน่ง	ชื่อ	อธิบาย
0-3	Version	มีขนาด 4 บิตเป็นเวอร์ชั่นของ IP ปัจจุบันค่านี้ถูกกำหนดให้เป็น 4
4-7	Lenght	มีขนาด 4 บิตเป็นค่าความยาวของ Header นี้ โดยปกติจะเป็น 5 หมายความว่า 5*32 บิต = 20 ไบต์
8-15	Type of Service	เป็นข้อมูลขนาด 8 บิต ปัจจุบันไม่ได้ใช้งานแล้ว
16-31	Total length	เป็นฟิลด์ที่บอกจำนวนไบต์ทั้งหมดของ IP Datagram ด้วยขนาด 16 บิตทำให้ Datagram มีขนาดสูงสุดไม่เกิน 65535 ไบต์ และมีขนาดเล็กสุดไม่ต่ำกว่า 512 ไบต์
32-47	Identification	ใช้ในกรณีที่มีการแบ่งดาต้าแกรมออกเป็นแฟรกเมนต์ เมื่อนำกลับมารวมกันใหม่จะได้รู้ว่ามาจากดาต้าแกรมเดียวกัน
48-50	Flag	ใช้ในกรณีที่มีการแบ่งข้อมูลออกเป็นแฟรกเมนต์ มีความหมายดังนี้ บิต 0 : reserved เป็น 0 เสมอ บิต 1 (DF) 0 = May Fragment, 1 = Don't Fragment บิต 2 (MF) 0 = Last Fragment, 1 = More Fragments.
51-63	fragment offset	เป็นส่วนระบุข้อมูลที่ใช้แยกรวมข้อมูล เพื่อให้ผู้มูลที่ถูกแยกออกเป็นแฟรกเมนต์กลับมารวมกันได้อย่างถูกต้องตามลำดับ
64-71	Time to Live (TTL)	เป็นจำนวนครั้งสูงสุดที่ดาต้าแกรมนี้จะถูกส่งผ่านเครือข่ายไปยังปลายทางได้ เพื่อ ป้องกันไม่ให้ดาต้าแกรมถูกเราต์ไปเรื่อยๆอย่างไม่สิ้นสุด ปกติค่านี้จะเริ่มต้นที่ 32 และจะถูกลดค่าลงทีละ 1 เมื่อมีการเราต์ จนค่านี้มีค่าเป็น 0 ก็จะไม่ถูกเราต์อีกต่อไป
72-79	Protocol	เป็นข้อมูลที่ระบุโปรโตคอลที่ส่งดาต้าแกรมนี้มา ตัวอย่างโปรโตคอลที่ใช้บ่อยๆ ได้แก่ โปรโตคอลที่ใช้ในฟิลด์ Protocol / อธิบาย ICMP 1 Internet Comtrol Message Protocol TCP 6 Transmission Control Protocol UDP 17 User Datagram Protocol
80-95	Header Checksum	เป็นส่วนตรวจสอบความถูกต้องของข้อมูลใน Header โดยไม่เกี่ยวกับส่วนข้อมูลที่อยู่ภายใน payload ค่านี้จะถูกคำนวนใหม่ทุกครั้งที่มีการเปลี่ยนแปลงข้อมูลใน Header (เช่น TTL ที่มีการเปลี่ยนแปลงทุกครั้งที่ IP datagram ถูกส่งผ่านเราเตอร์)
96-127	Source IP Address	คือ IP Address ของผู้ส่งดาต้าแกรม
128-163	Destination IP Address	คือ IP Address ของผู้รับดาต้าแกรม
ไม่แน่นอน	Option	มีขนาดข้อมูลไม่แน่นอน ใช้สำหรับกำหนดค่าพารามิเตอร์ปลีกย่อย ซึ่งส่วนใหญ่ไม่มีการนำไปใช้งาน
ขึ้นอยู่กับ Option	Padding	มีข้อมูลว่างเปล่า ใช้เป็นส่วนเติมเต็มของฟิลด์ Option ให้ครบ 32 ไบต์

รูปที่ 2.11 รูปแบบของไอพีเฮดเดอร์

เมื่อเราทราบตำแหน่งบิตหรือไบท์ของข้อมูลที่เป็นไอพี ของต้นทางและปลายทางแล้วส่วนที่เหลือก็จะเป็นเพียงการกำหนดเงื่อนไขว่าไอพีแอดเดรส ที่ได้ทำการแยกมาจากเฟรมข้อมูลทั้งหมดนั้นได้ตรงกับข้อมูลในรายการหรือปล่าวถ้าเกิดตรงกันก็ให้ทำตามเงื่อนไขที่กำหนดว่าจะให้กรองข้อมูลทั้งชุดนั้นๆทิ้งหรือไม่

2.16 เอ็นแคปซูลเลชัน (Encapsulation)

เป็นกระบวนการสอดใส่ข้อมูลโดยทำการห่อหุ้มข้อมูลที่จะส่งด้วยข้อมูลควบคุมที่จำเป็นสำหรับโปรโตคอลนั้นๆ ก่อนที่จะส่งเข้าสู่ระบบเครือข่ายต่อไป โดยหลังจากที่ข้อมูลนั้นผ่านลงมาจาก โอเอสไอโมเดลในแต่ละชั้นก็จะใส่ข้อมูลในส่วนหัว(Header) ซึ่งจะบรรจุข้อมูลที่เกี่ยวกับการควบคุมการทำงานสำหรับอุปกรณ์เครือข่ายและผู้รับข้อมูล เพื่อรับประกันได้ว่าข้อมูลที่จะนำส่งได้รับการดูแลเป็นอย่างดีและสามารถนำส่งได้อย่างเรียบร้อย

2.17 ดีเอ็นแคปซูลเลชัน (Deencapsulation)

เป็นกระบวนการปลดปล่อยการห่อหุ้มข้อมูลโดยทั่วไปข้อมูลที่ส่งมาเป็นกระแสบิตที่อุปกรณ์รับเข้ามานั้นก่อนการจะส่งเป็นกระแสบิตนั้นได้ผ่านการเอ็นแคปซูลเลชัน มาก่อนซึ่งก็คือการนำส่วนของข้อมูลมาหุ้มด้วยส่วนของเฮดเดอร์ ในทางกลับกันกระบวนการดีเอ็นแคปซูลเลชันก็จะเป็นการนำส่วนของข้อมูลออกจากเฮดเดอร์นั่นเอง

2.18 เน็ตเวิร์กแอดเดสทรานส์เลชัน (Network Address Translation)

การทำ แน็ท (NAT) เป็นการแทนที่หมายเลขไอพีแอดเดรสต้นทางของไอพีแพ็กเก็ตที่วิ่งผ่านอุปกรณ์ออกไปให้เป็นหมายเลขไอพีแอดเดรสกำหนด มีอุปกรณ์หลายชนิดที่สามารถทำแน็ทได้ เช่น เร้าเตอร์ และไฟร์วอลล์ เป็นต้น โดยในเนื้อหานี้เราจะทำแบบสเตติกแน็ท (Static NAT)

- สเตติกแน็ท: เป็นการทำ แน็ท แบบหนึ่งต่อหนึ่ง (One to One) ความหมายก็คือ กำหนดให้มีแมปว่า หมายเลขไพรเวทแอดเดรส(Private Address) เบอร์นี้จะถูกแทนที่ (Translate) ออกไปเป็นหมายเลขพับบลิคแอดเดรส (Public Address) เบอร์ใดเบอร์หนึ่ง

2.19 โพรมิสคิวอัสโหมด (Promiscuous mode)

โพรมิสคิวอัสโหมดในด้านที่เกี่ยวข้องกับคอมพิวเตอร์นั้น จะหมายถึงการกำหนดให้เน็ตเวิร์กการ์ดนั้นรับทุกๆแพ็กเก็ตที่ผ่านเข้ามาและส่งไปให้ซีพียูซึ่งโดยปกติแล้วในแต่ละแพ็กเก็ตนั้นจะมีการระบุแม็คแอดเดรสปลายทางอยู่ในแพ็กเก็ต และเมื่อ เน็ตเวิร์กการ์ดรับแพ็กเก็ตเข้ามาก็จะตรวจสอบว่า แม็คแอดเดรสปลายทางในแพ็กเก็ตนั้นตรงกับของตัวเองหรือไม่ ถ้าเกิดว่าตรงก็จะรับแพ็กเก็ตนั้น แต่ถ้าไม่ตรงก็จะทำการดร็อปแพ็กเก็ต แต่เมื่อทำการกำหนดโพรมิสคิวอัสโหมดแล้ว เน็ตเวิร์กการ์ดจะไม่ดร็อปแพ็กเก็ตใดๆเลย แต่จะอ่านแพ็กเก็ตทั้งหมดแทน

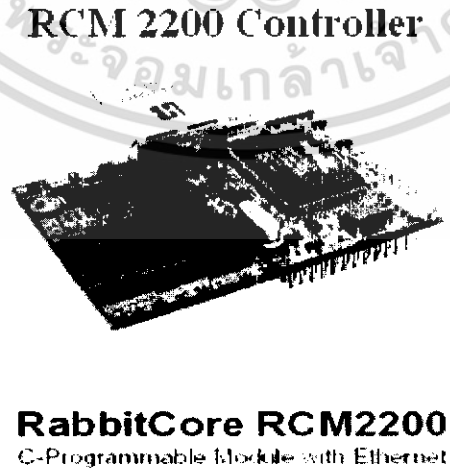
บทที่ 3
ไมโครโปรเซสเซอร์ Rabbit 2200

ไมโครโปรเซสเซอร์ตระกูล Rabbit ซึ่งตัวที่ใช้ คือRCM2200 มีความสามารถมากโดยมีความแตกต่างจากไมโครโปรเซสเซอร์ตัวอื่น คือมีความเร็วประมวลผลและความสามารถด้านการติดต่อ ระบบเครือข่าย อีกทั้งยังมีพอร์ตอินพุตและเอาต์พุตมากมาย รายละเอียดต่างๆจะอธิบายดังนี้

3.1 คุณลักษณะพื้นฐานของ Rabbit 2200

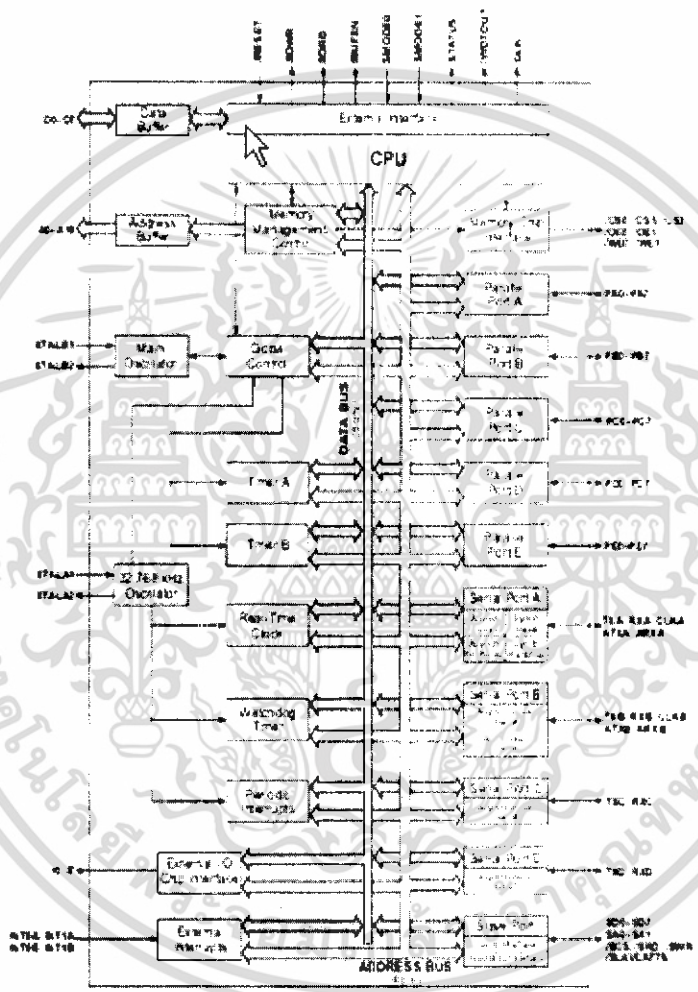
RCM 2200 เป็น module ซึ่งใช้ microprocessor Rabbit 2000 และมีความสามารถดังนี้มี

- มีขนาดเล็ก: 1.60" × 2.30" × 0.86" (41 mm × 58 mm × 22 mm)
- Microprocessor: Rabbit 2200 ใช้สัญญาณนาฬิกาที่ 29.5 MHz
- หน่วยประมวลผลกลางขนาด Rabbit 8 bit processors
- หน่วยความจำโปรแกรมภายในขนาด Flash memory 256 KB
- หน่วยความจำแบบ Static RAM (SRAM) ภายในจำนวน 128 KB
- Digital I/O port
- Timer/RTC
- Parallel I/O
- Slave port
- Serial port
- 4 high-speed ports: 2 configurable as clocked ports, 1 clocked port dedicated to programming port use.
- 10 Base-T Ethernet port 1 port

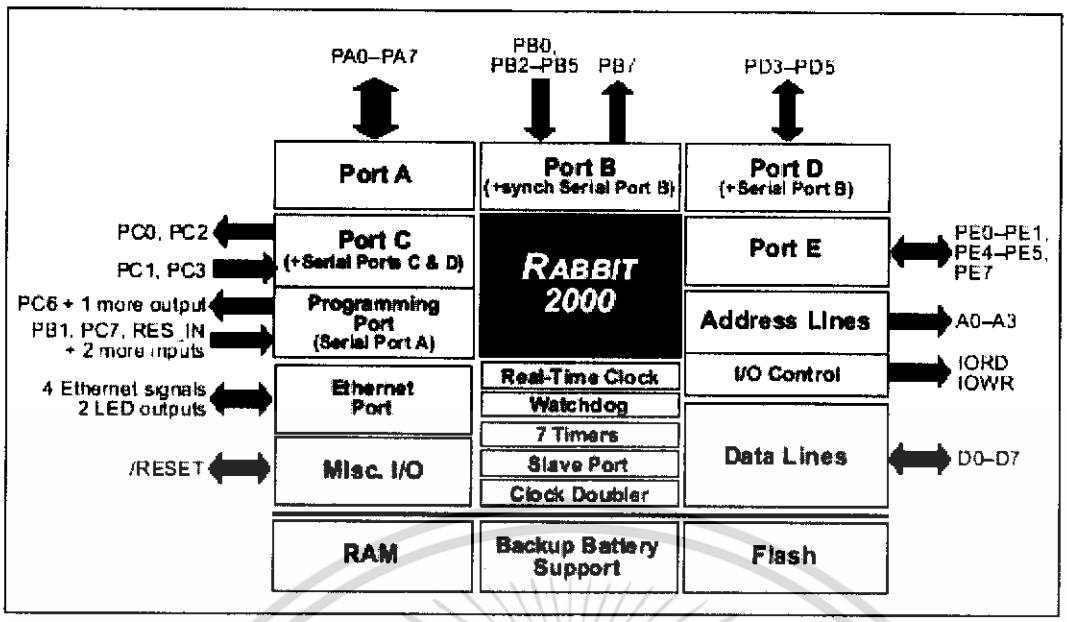


รูปที่ 3.1 แรบบิทโมดูล RCM 2200

Rabbit 2200 จัดเป็น Microprocessor ที่มีความกะทัดรัดและมีกำลังความสามารถมาก อีกทั้งยังมีส่วนประกอบที่จำเป็น เช่น มี 40 I/O, 4 CMOS-compatible serial ports, มี Timer 8 - bit 5 ตัว และ Timer 10 - bit 1 ตัว กับชุด registers 2 ชุด และ a fast number-crunching clock ส่วน Flash และ SRAM จะอยู่บนบอร์ด โดยตัวอุปกรณ์ Controllerได้เลือกใช้ RCM2000 จะประกอบด้วย Standard 10-pin programming port ซึ่งจำเป็นสำหรับ circuit emulators โดยเราสามารถที่จะทำการ Download และ debug software ได้โดยใช้ Dynamic C ซึ่งเป็น การศึกษารูปแบบการทำงานที่เราไม่คุ้นเคยมากนัก ต้องเริ่มต้นในการศึกษาพื้นฐานของภาษาชนิดนี้ใหม่

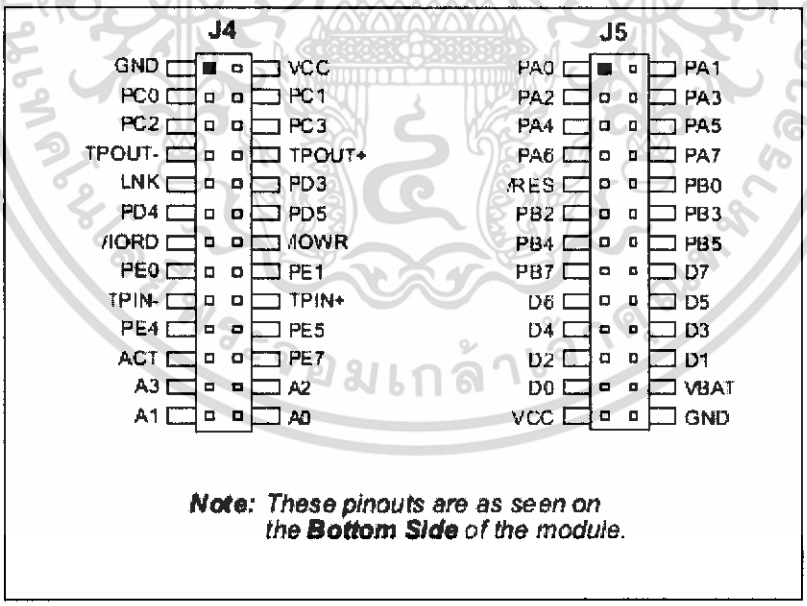


รูปที่ 3.2 Block Diagram of the Rabbit Microprocessor



รูปที่ 3.3 Rabbit Subsystems

RCM2200 มีพอร์ตขนานสำหรับใช้งานทั้งหมด 26 พอร์ต โดยเป็นพอร์ตที่แบ่งการทำงานเป็น 5 กลุ่มๆ ละ 8 บิต โดยจะอยู่ในหัวเชื่อมต่อของจั๊มเปอร์ 4 และ จั๊มเปอร์ 5 โดยพอร์ตอินพุตและพอร์ตเอาท์พุตแบบสองทิศทางจะอยู่ที่ตำแหน่งที่ PA0 – PA7, PD3 – PD5, PE0 – PE1, PE4, PE5 และ PE7



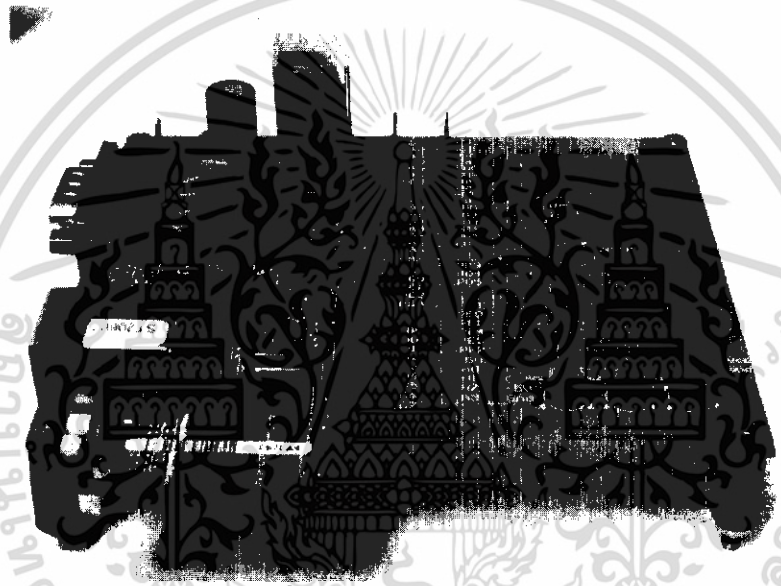
รูปที่ 3.4 RCM2200 I/O Pin outs

3.2 Prototyping Board

เป็นอุปกรณ์ส่วนที่ช่วยในการทำงานเพื่อเชื่อมต่อกับ Controller RCM 2200 ต้องมีการออกแบบให้สามารถทำงานในหน้าที่ต่างๆ ดังนี้

- เป็นตัวจ่ายไฟ ให้กับ RCM 2200 module จำเป็นต้องมีส่วนวงจรในการแปลงกระแส
- ทำหน้าที่ขยายส่วน Port การทำงานให้สามารถทำงานได้สะดวกยิ่งขึ้น
- ทำหน้าที่เชื่อมต่อกับสาย Flash

และเพื่อศึกษาและการทดลองการทำงาน (Test Board) จำเป็นต้องสร้าง Prototyping Board เพราะจะได้ความสะดวกในการทำงานเนื่องจากขาอุปกรณ์และ Port ที่ใช้มีขนาดเล็ก จำเป็นต้อง สร้างเพื่อทำให้ port มีขนาดที่เป็นมาตรฐาน

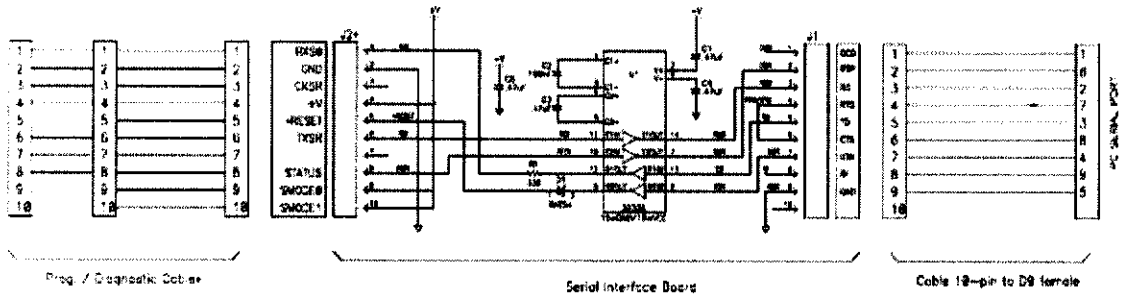


รูปที่ 3.5 Prototyping Board

ส่วนของ Prototyping Board นั้นถูกออกแบบโครงสร้างโดยการเน้นให้สามารถใช้งานในส่วน port ต่างๆให้มีความสะดวกในการใช้งาน มีการออกแบบให้มีชุด ของส่วนแปลงกระแสไฟฟ้า ส่วนเชื่อมต่อกับ serial port และ Battery Backup ซึ่งเป็นอีกส่วนหนึ่งที่สำคัญในการรักษาข้อมูลในขณะมีการเกิดไฟฟ้าขัดข้อง

3.3 สาย Flash programmable

การสร้างสาย Flash programmable เพื่อทำการเชื่อมต่อ ข้อมูลที่ต้องการ program ลงในส่วน Flash และRam เพื่อให้สามารถใช้งานได้อย่างสมบรูณ์ โดยการติดต่อผ่าน serial port A ของ controller RCM-2200 การออกแบบใช้อุปกรณ์ ง่ายๆในการเชื่อมต่อ โดยใช้ jumper ขนาดเล็กในกาเชื่อมต่อกับ serial port A ของ controller RCM-2200 เลือกใช้การเชื่อมต่อโดยใช้ IC MAX232 เพื่อให้สามารถติดต่อข้อมูลผ่าน serial port ได้ รูปแบบการออกแบบต้องเป็นไปตาม Schematicsที่แสดงการเชื่อมต่ออุปกรณ์ต่างๆภายใน



รูปที่ 3.6 สาย Flash Programable

3.4 ส่วนของ Software ที่ใช้ในการพัฒนา Rabbit 2200

Software ที่นำมาใช้ในการโปรแกรม และ Debug คือ Dynamic C (windows 98/ME/XP) โดยมีคุณสมบัติดังนี้ คือ

- โครงสร้างภาษาเหมือนภาษาซี
- มี Library ที่สนับสนุนการเขียนโปรแกรมเชื่อมต่อกับ Network
- สามารถโปรแกรมลง RCM2200 Chip โดยผ่านทาง Serial port หรือ Ethernet และยังสามารถ program ลงบน RAM หรือ flash ก็ได้

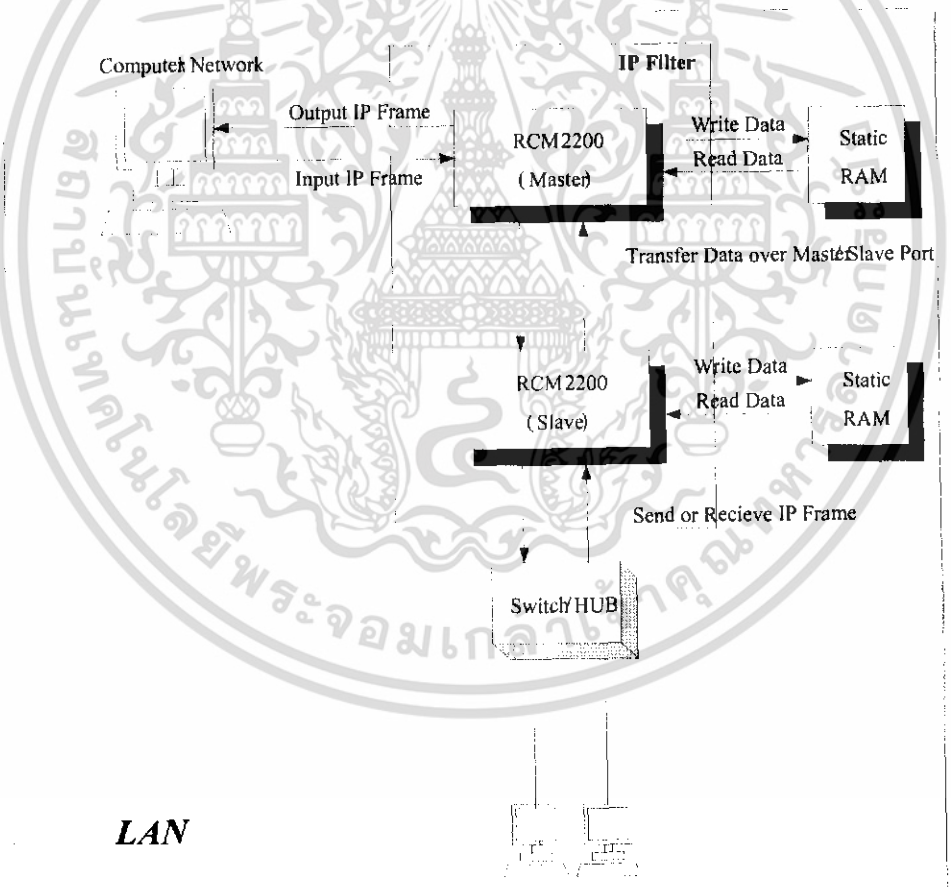
บทที่ 4
หลักการสร้างและการออกแบบ

ในการสร้างและออกแบบอุปกรณ์ IP Filter นั้นได้แบ่งออกเป็นสองส่วนหลักๆคือ

- **ฮาร์ดแวร์ (Hardware)** เนื่องจากอุปกรณ์จำเป็นต้องมีพอร์ต Ethernet จำนวน 2 พอร์ตในการรับส่งข้อมูลระหว่างต้นทางและปลายทางดังนั้นส่วนนี้จึงเป็นส่วนเชื่อมต่อระหว่างอุปกรณ์ RCM 2200 เข้าด้วยกัน
- **ซอฟต์แวร์ (Software)** เป็นส่วนโปรแกรมการทำงานให้ไมโครโปรเซสเซอร์แรบบิททำการควบคุมฮาร์ดแวร์ให้ทำงานในลักษณะของ IP Filter ได้ตามที่ต้องการ

4.1 โครงสร้างของอุปกรณ์ IP Filter

โดยโครงสร้างของอุปกรณ์เป็นดังรูปที่ 4.1

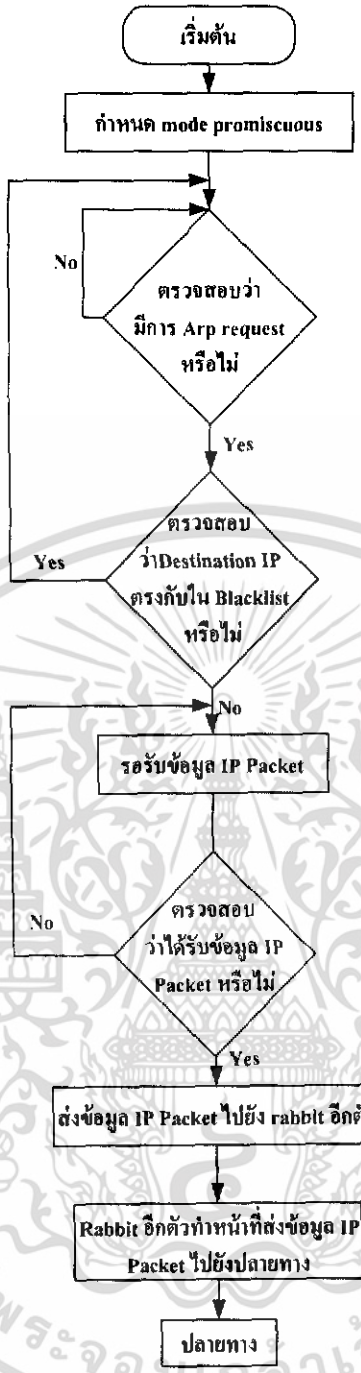


รูปที่ 4.1 โครงสร้างการทำงานของอุปกรณ์ IP FILTER ที่ออกแบบด้วยโมดูล RCM 2200

จากรูปที่ 4.1 แสดงโครงสร้างการทำงานโดยรวมของ IP FILTER การติดต่อส่วนต่างๆเข้ากับ RCM2200 โดยส่วนแรกที่ติดต่อกับ RCM2200 Board และถือว่ามีความสำคัญคือ ส่วนที่เชื่อมต่อกับทางด้าน Input ที่เป็นคอมพิวเตอร์ โดยจะเป็นส่วนที่ใช้ในการรับ Data Input ที่เป็นข้อมูล IP เมื่อมีการส่งผ่านข้อมูลเข้ามาแล้วข้อมูลจะถูกนำไปเก็บไว้ในหน่วยความจำของ RCM2200 และนำส่วนของ IP header มาทำการเปรียบเทียบกับ IP ในแบล็คลิสท์ ถ้าหากตรงกันก็ให้ทำการกรองหรือส่งผ่านต่อไปยัง RCM2200 อีกตัวหนึ่งแล้วตัว RCM 2200 ตัวหลังนี้จะทำหน้าที่ฟอร์เวิร์ดข้อมูลไปยังปลายทางที่ต้องการอีกที

4.2 โปรแกรมที่ไว้ติดต่อกับทางด้าน Input

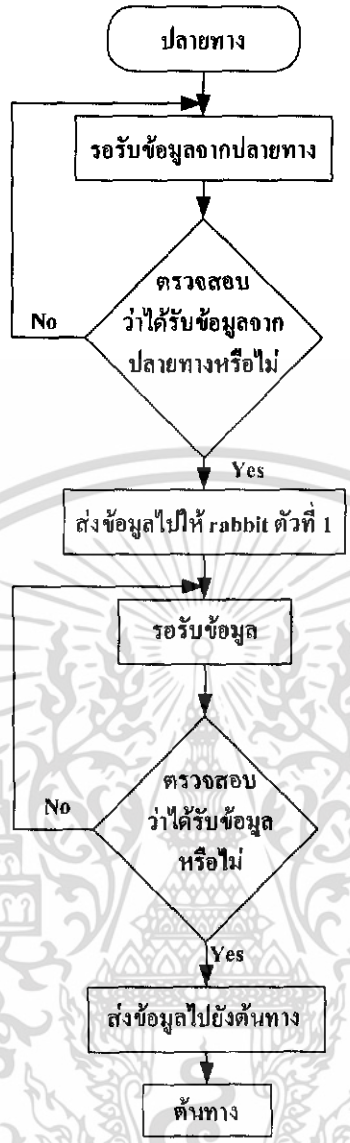
ในส่วนนี้จะเขียนโปรแกรมรับ Input IP ใดๆแล้วส่งไปยังทางด้านสเลฟถ้าหากไอพีแอดเดรสนั้นไม่ได้อยู่ในแบล็คลิสท์โดยที่จะเป็นไปตาม Flow Chart ดังรูปที่ 4.2



รูปที่ 4.2 Flow Chart การส่งข้อมูลจากต้นทางไปยังปลายทาง

4.3 โปรแกรมที่ไว้ติดต่อกับทางด้าน Output

ในส่วนนี้จะเขียนโปรแกรมในการรับข้อมูลจากทางด้านมาสเตอร์แล้วส่งออกไปยังเน็ตเวิร์กโดยที่จะเป็นไปตาม Flow Chart ดังรูปที่ 4.3

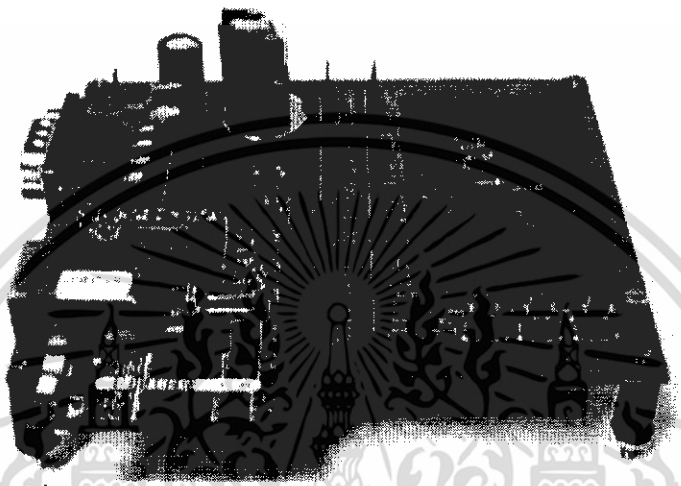


รูปที่ 4.3 Flow Chart การตอบกลับข้อมูลจากปลายทาง

บทที่ 5
การทดลองและผลการทดลอง

การทดลองที่ 5.1 การใช้งานอุปกรณ์แรบบิทเบื้องต้นและหมายเลข Ethernet Address ของอุปกรณ์

1. นำอุปกรณ์แรบบิทโมดูล (RCM 2200) มาต่อเข้ากับโปรโตไทป์บอร์ด (Prototype Board) ดังรูปที่ 5.1.1

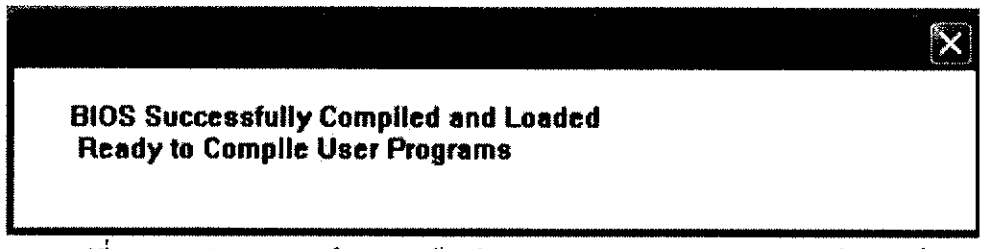


รูปที่ 5.1.1 รูปแสดงการต่อแรบบิทโมดูลเข้ากับโปรโตไทป์บอร์ด

2. นำปลายสายแฟลชโปรแกรม (Flash program) ด้าน 10 pin ต่อเข้ากับอุปกรณ์แรบบิทแล้วนำด้านที่เป็น DB-9 ต่อเข้ากับพอร์ตซีเรียลที่คอมพิวเตอร์ดังรูปที่ 5.1.2

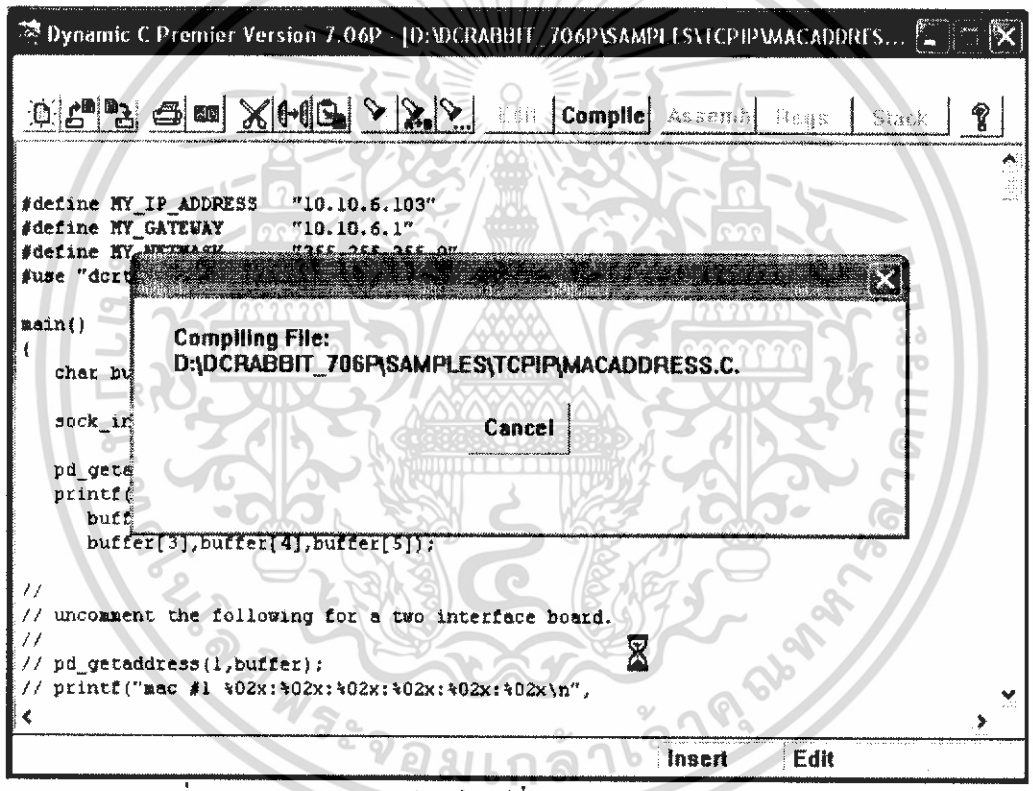
รูปที่ 5.1.2 รูปแสดงการต่อสายโปรแกรมเข้ากับตัวอุปกรณ์เพื่อเตรียมการลงโปรแกรม

3. ทำการเปิดโปรแกรมไดนามิคซี (Dynamic C7.06 P) ถ้าอุปกรณ์แรบบิทโมดูลทำงานได้ถูกต้องจะแสดงผลการตรวจพบแรบบิทไมโครโปรเซสเซอร์ดังรูปที่ 5.1.3



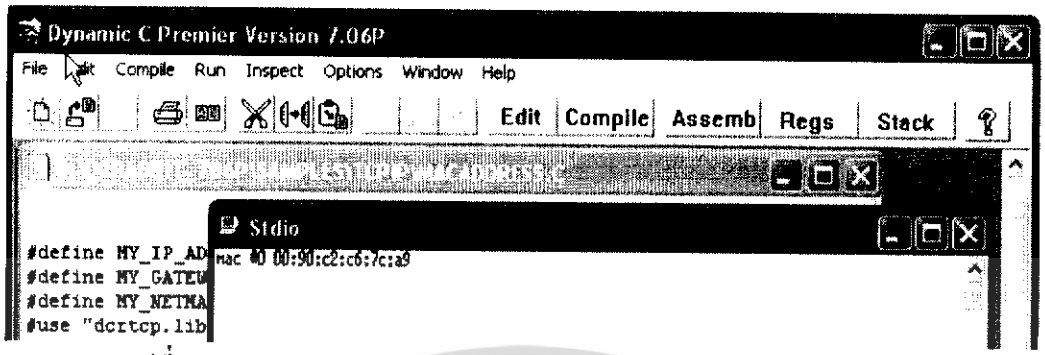
รูปที่ 5.1.3 รูปภาพแสดงถึงความพร้อมในการติดต่อแรบบิทโมดูลกับคอมไพเลอร์

4. ทำการเปิดไฟล์คำสั่งที่เขียนไว้สำหรับดูแมคแอดเดรสของแรบบิทโมดูลแล้วสั่งคอมไพล์ดังรูปที่ 5.1.4



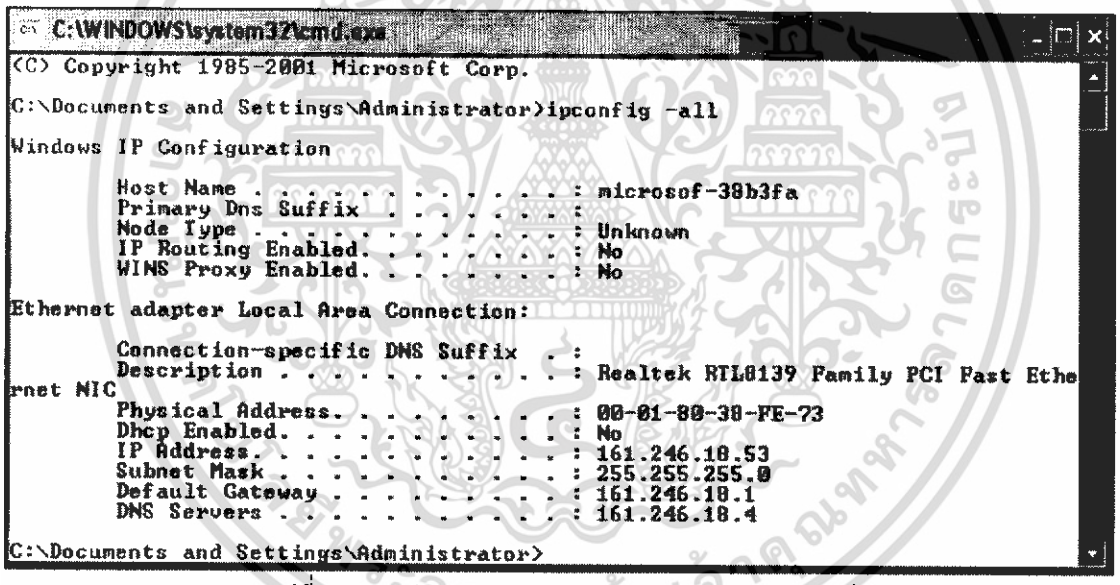
รูปที่ 5.1.4 แสดงการคอมไพล์ไฟล์ที่เขียนมาแสดงผล Ethernet Address

5. ทำการสั่งรันโปรแกรมจะมีการแสดงผลหมายเลขอีเทอร์เน็ตแอดเดรสของอุปกรณ์ทางหน้าต่าง Stdio ของอุปกรณ์ดังรูปที่ 5.1.5



รูปที่ 5.1.5 แสดง Ethernet Address ผ่านทางหน้าต่าง Stdio ของ Dynamic C

6. ทำการดูหมายเลขอีเทอร์เน็ตแอดเดรสของคอมพิวเตอร์โดยใช้คำสั่ง ipconfig –all จะมีการแสดงผลดังรูปที่ 5.1.6

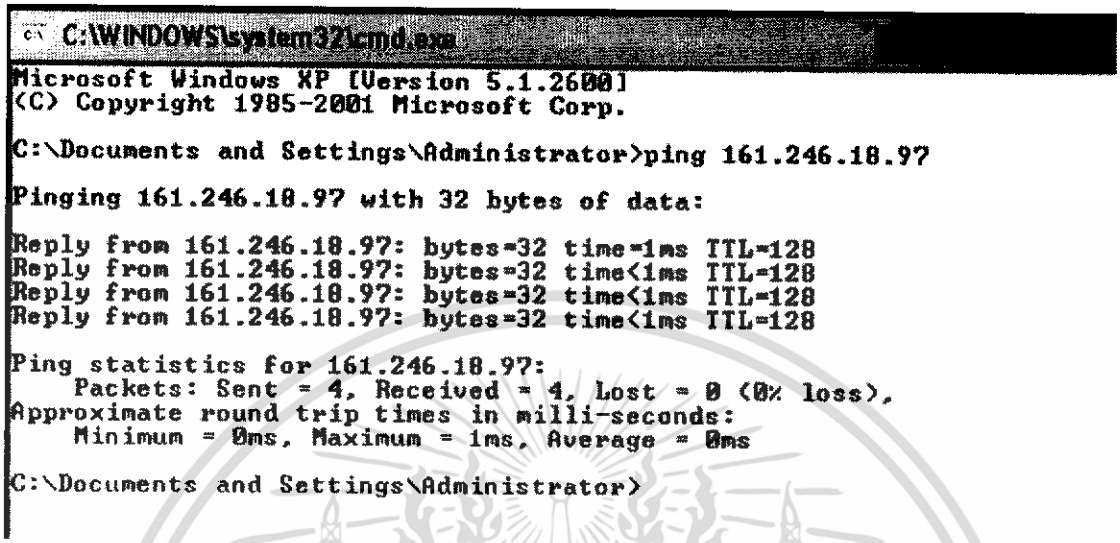


รูปที่ 5.1.6 รูปแสดง Ethernet Address ของคอมพิวเตอร์

การทดลองที่ 5.2 การกำหนดไอพีแอดเดรส (IP Address) ให้กับอุปกรณ์

1. ในการทดลองนี้จะกำหนดหมายเลขไอพีแอดเดรสให้กับอุปกรณ์แรบบิทและคอมพิวเตอร์แล้วทดสอบการติดต่อผ่านทางพอร์ตอีเทอร์เน็ตของอุปกรณ์แรบบิทและคอมพิวเตอร์โดยทำการลงโปรแกรมที่มีการกำหนดหมายเลข IP Address ให้กับอุปกรณ์แรบบิทโมดูลแล้วทำการกำหนดค่าไอพีแอดเดรสทางฝั่งคอมพิวเตอร์แล้วจึงทำการกดปุ่มรีเซ็ตที่ Prototype Board เพื่อให้อุปกรณ์แรบบิทเริ่มทำงานตามโปรแกรมที่ได้เขียนไว้ควบคุม

2. ใช้คำสั่ง ping IP Address ของแรบบิทที่คอมมานด์พร้อมพ์ของคอมพิวเตอร์จะได้ผลลัพธ์ดังรูปที่ 5.2 ซึ่งแสดงถึงสถานการณ์เชื่อมต่อของอุปกรณ์แรบบิทกับคอมพิวเตอร์นั้นสามารถเชื่อมต่อกันได้ทางพอร์ตอีเทอร์เน็ต



อ

รูปที่ 5.2 แสดง ICMP Reply จากอุปกรณ์แรบบิท

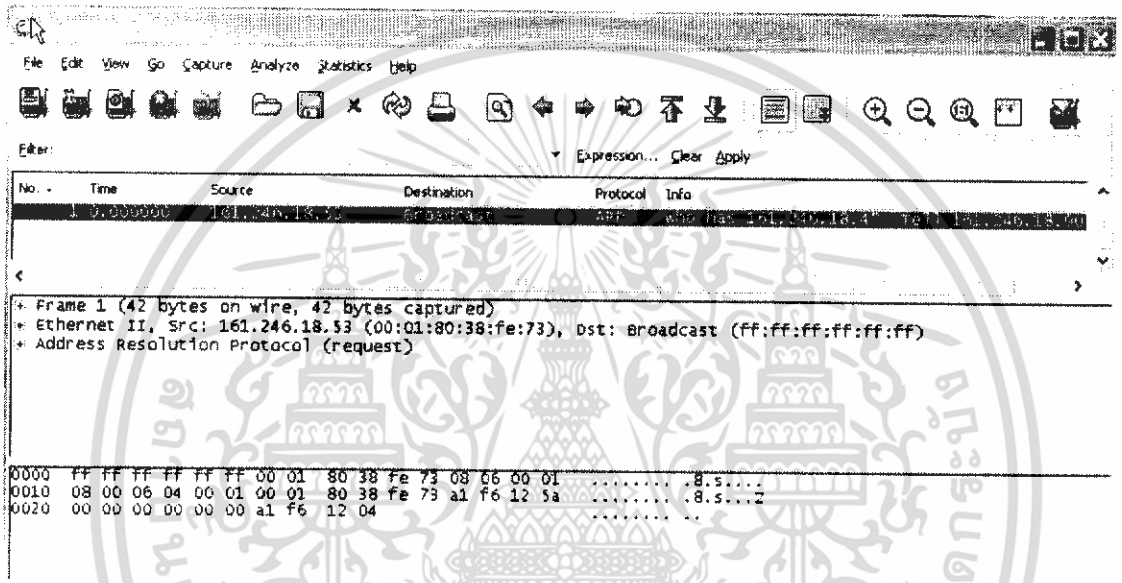
การทดลองที่ 5.3 การกำหนด Promiscusous mode

1. การทำงานในส่วนนี้จะเป็นการกำหนดโหมดโพรมิสคิวอัส (promiscuous mode) เพื่อให้อุปกรณ์สามารถรับข้อมูลไอพีได้ถึงแม้ว่าไอพีปลายทางที่รับมานั้นจะไม่ใช่ไอพีของอุปกรณ์ RCM 2200 ซึ่งโดยหลักการปกติแล้วหากไม่ได้กำหนดโหมดนี้อุปกรณ์จะไม่รับแพ็คเกตใดๆที่มีหมายเลขไอพีแอดเดรสไม่ตรงกับของมันโดยการทดลองจะทำการลงโปรแกรมที่กำหนดโหมดนี้ให้กับอุปกรณ์แรบบิทเพื่อตรวจจับแพ็คเกตที่ผ่านเข้าออกพอร์ตอีเทอร์เน็ตของอุปกรณ์แรบบิทโดยจะมีการแสดงผลออกทางหน้าต่าง Stdio เสมือนเป็นโปรแกรมสนิฟเฟอร์ทั่วไปดังรูปที่ 5.3

รูปที่ 5.3 รูปแสดงแพ็คเกตที่รับได้ทางหน้าต่าง Stdio

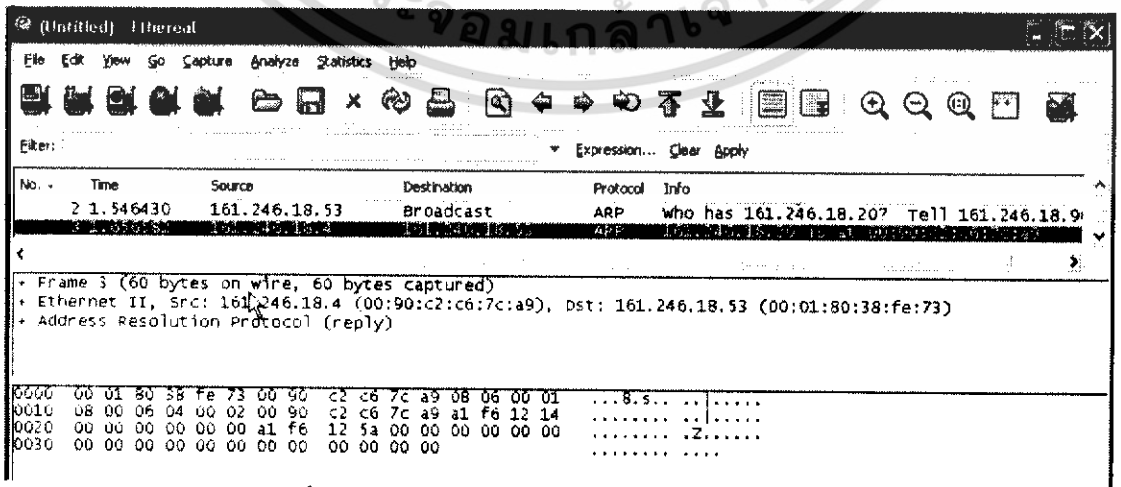
การทดลองที่ 5.4 การทำงานของ ARP

1. การทำงานในส่วนนี้จะเป็นกระบวนการของ ARP เนื่องจากต้องให้ข้อมูลจากต้นทางเริ่มต้นส่งข้อมูลออกมาจึงจำเป็นที่ทางต้นทางจะต้องได้รับการตอบกลับ Arp reply โดยการทดลองจะเริ่มต้นจากใช้คำสั่ง ping ไปยังไอพีแอดเดรสที่ไม่ใช่ไอพีแอดเดรสของแรบบิทซึ่งจะเป็นการส่งเฟรม ICMP Request ไปยังแรบบิทในเบื้องต้นจะพบว่ามีแต่เฟรม Arp request จากคอมพิวเตอร์ซึ่งในการทดลองนี้จะใช้โปรแกรม Ethereal ดักจับแพ็คเกตที่ฝั่งคอมพิวเตอร์ซึ่งจะแสดงผลได้ดังรูปที่ 5.4.1



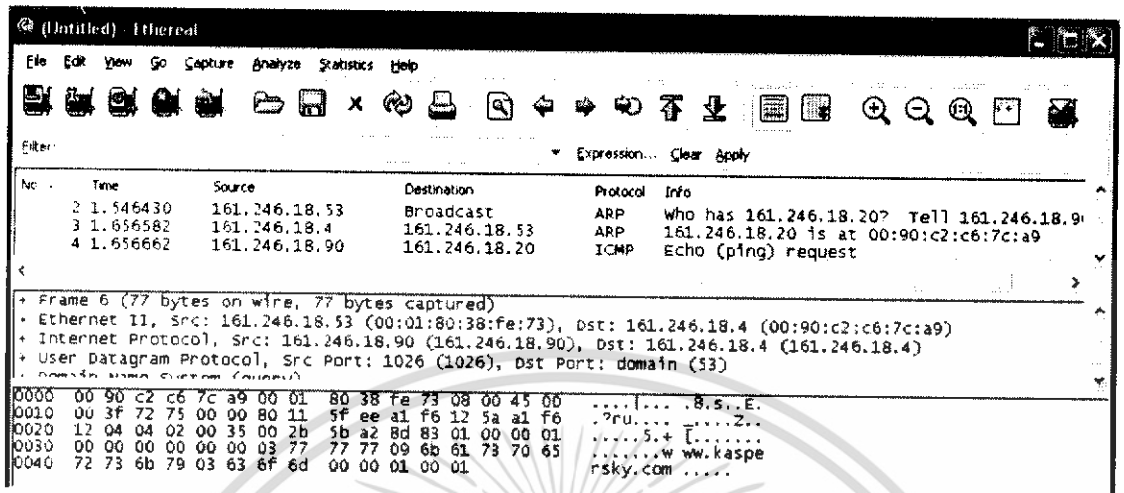
รูปที่ 5.4.1 แสดงถึงเฟรม Arp request จากคอมพิวเตอร์ต้นทาง

2. ทำการลงโปรแกรมควบคุมการตอบกลับ Arp reply ให้กับแรบบิทซึ่งจะมีการแสดงผลเฟรม arp reply ที่ส่งมายังคอมพิวเตอร์ทางโปรแกรม Ethereal ดังรูปที่ 5.4.2



รูปที่ 5.4.2 แสดงถึงเฟรม Arp reply ที่ส่งมาจากแรบบิท

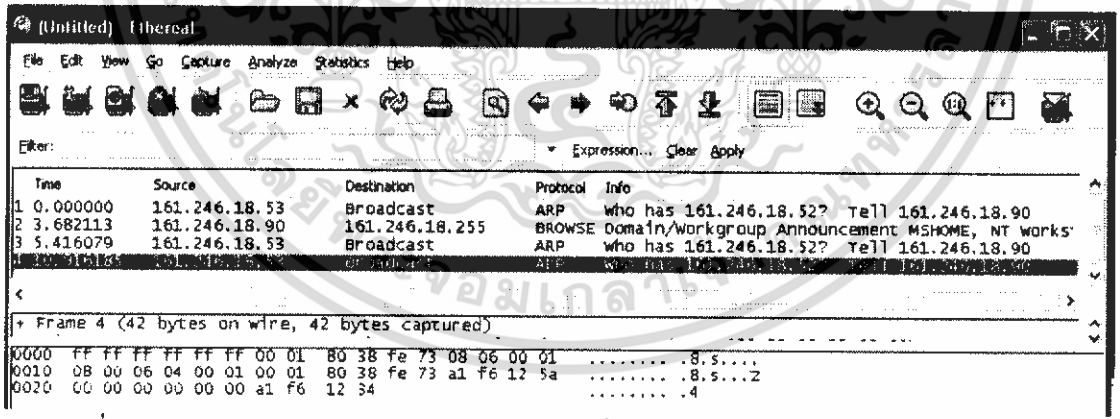
3. พบว่าจะมีการแสดงผลการเริ่มต้นส่งข้อมูลเฟรม ICMP request มาจากคอมพิวเตอร์ดังรูปที่ 5.4.3



รูปที่ 5.4.3 แสดงผลการเริ่มต้นส่งข้อมูลจากเครื่องต้นทาง

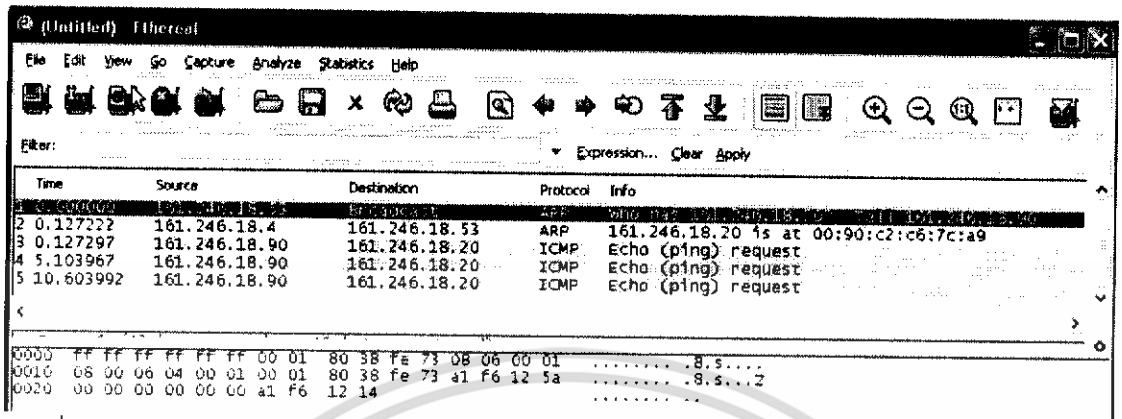
การทดลองที่ 5.5 การ ทำงานของ Arp reply ร่วมกับส่วนเปรียบเทียบ IP

1. ในการทดลองนี้จะทดลองการตอบกลับ Arp reply ร่วมกับส่วนเปรียบเทียบ IP โดยจะลงโปรแกรมที่มีการตอบกลับ Arp reply เฉพาะไอพีแอดเดรสปลายทางของเฟรม Arp request ที่ไม่ได้อยู่ในแบล็คลิสท์เท่านั้น(ทุกไอพีแอดเดรสยกเว้น 161.246.18.50, 161.246.18.51, 161.246.18.52, 161.246.18.54) ซึ่งจะมีการแสดงผลเฉพาะ Arp request โดยที่ไม่มีการตอบกลับ Arp reply ดังรูปที่ 5.5.1



รูปที่ 5.5.1 แสดงถึงการส่งเฟรม Arp request จากเครื่องต้นทางแต่ไม่มีการได้รับเฟรม Arp reply

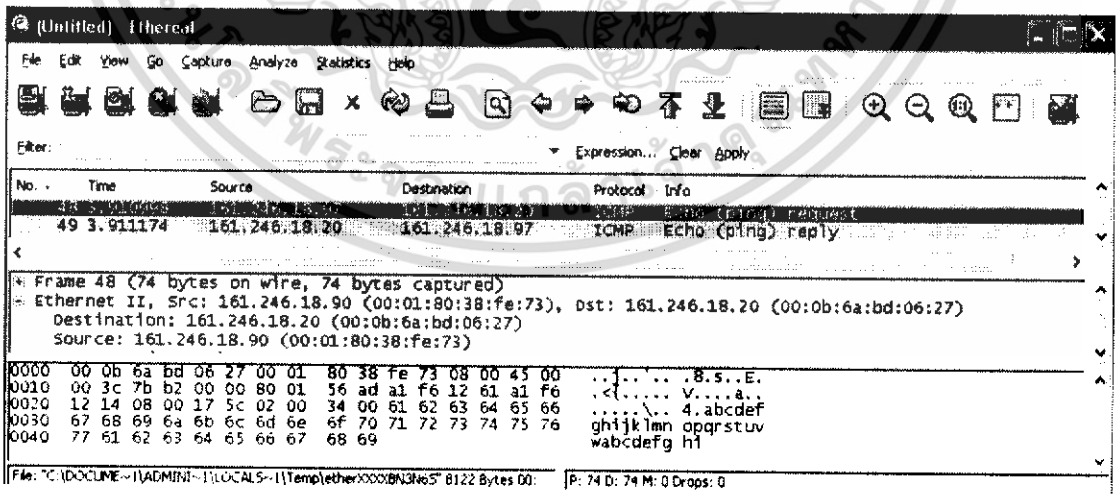
2. แต่ถ้าหากไอพีแอดเดรสปลายทางไม่ตรงกับหมายเลขไอพีแอดเดรสในแบล็คลิสท์ก็จะมีการตอบกลับ Arp reply ซึ่งแสดงผลทางโปรแกรม Ethereal ได้ดังรูปที่ 5.5.2



รูปที่ 5.5.2 รูปแสดงถึงการตอบ Arp reply กลับมาที่เครื่องต้นทางทำให้เครื่องต้นทางเริ่มต้นส่งข้อมูลได้

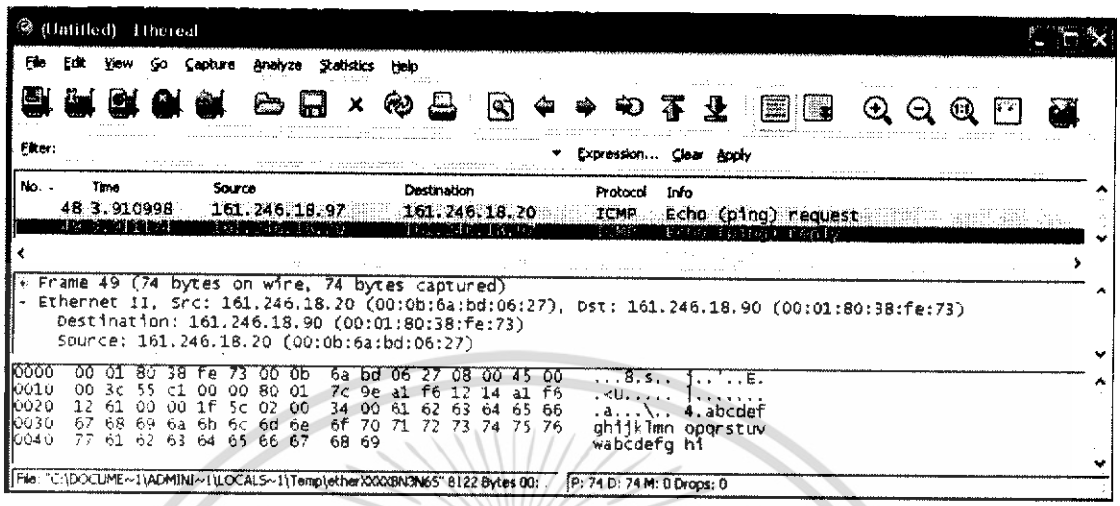
การทดลองที่ 5.6 การส่งข้อมูลระหว่างอุปกรณ์แรบบิท

1. ในการทดลองนี้จะเป็นการทดลองการส่งข้อมูลระหว่างอุปกรณ์แรบบิทโดยที่จะลงโปรแกรม ที่มีการเพิ่มส่วนเชื่อมต่อระหว่างตัวอุปกรณ์แรบบิท ซึ่งจะมีการส่งหมายเลขไอพีแอดเดรสปลายทางจากเฟรมข้อมูลที่รับได้จากแรบบิทตัวแรกไปยังแรบบิทตัวที่สองแล้วให้แรบบิทตัวที่สองทำการ Ping แล้วตรวจสอบว่าได้รับ ICMP reply จากไอพีแอดเดรส ที่แรบบิทตัวที่ 1 ส่งไปหรือไม่ซึ่งจะแสดงผลโดยการใช้โปรแกรม Ethereal ดักจับเฟรม ICMP request ที่แรบบิทตัวที่ 2 ส่งซึ่งจะแสดงผลดังรูปที่ 5.6.1



รูปที่ 5.6.1 แสดงเฟรม Arp request ที่ส่งมาจากแรบบิทตัวที่ 2

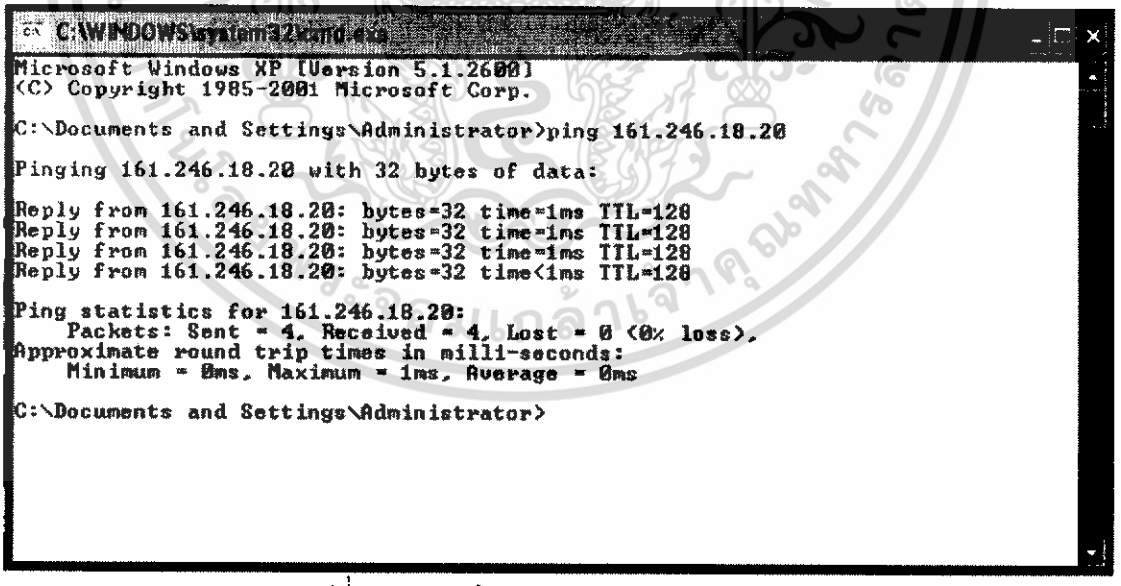
2. ซึ่งทางคอมพิวเตอร์ก็จะตอบกลับ ICMP reply ซึ่งจะแสดงผลดังรูปที่ 5.6.2



รูปที่ 5.6.2 แสดงเฟรม Arp reply ที่ส่งมาจากเครื่องปลายทางไปยังแรบบิทตัวที่ 2

การทดลองที่ 5.7 การทำงานของส่วนการตอบกลับ ICMP Reply

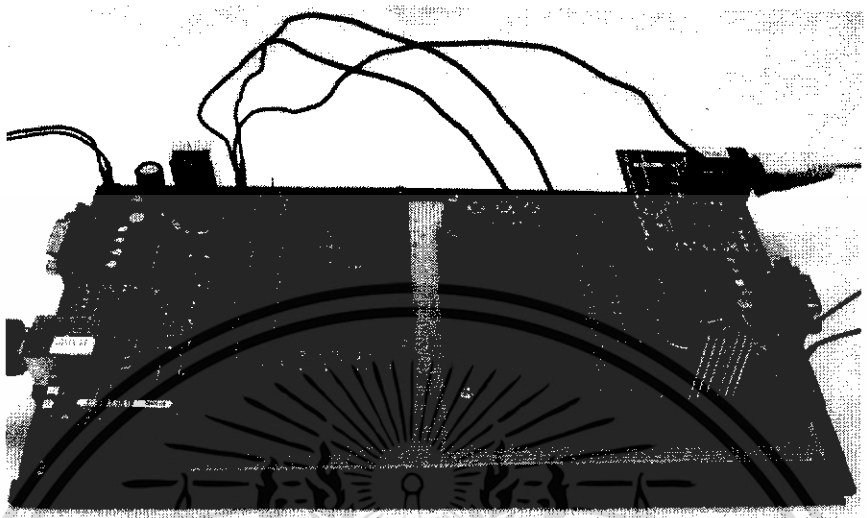
1. ในการทดลองนี้จะเป็นการทดลองโดยจะลงโปรแกรมที่มีหน้าที่การทำงานคือตอบกลับ ICMP reply ทุกๆ หมายเลข ไอพีแอดเดรสที่ส่ง ICMP request มาซึ่งจะแสดงผลการได้รับ ICMP replyที่ทางเครื่องที่ส่งเฟรม ICMP requestได้ดังรูปที่ 5.7



รูปที่ 5.7 แสดงถึงการตอบกลับ ICMP reply

การทดลองที่ 5.8 การทำงานของ IP Filter

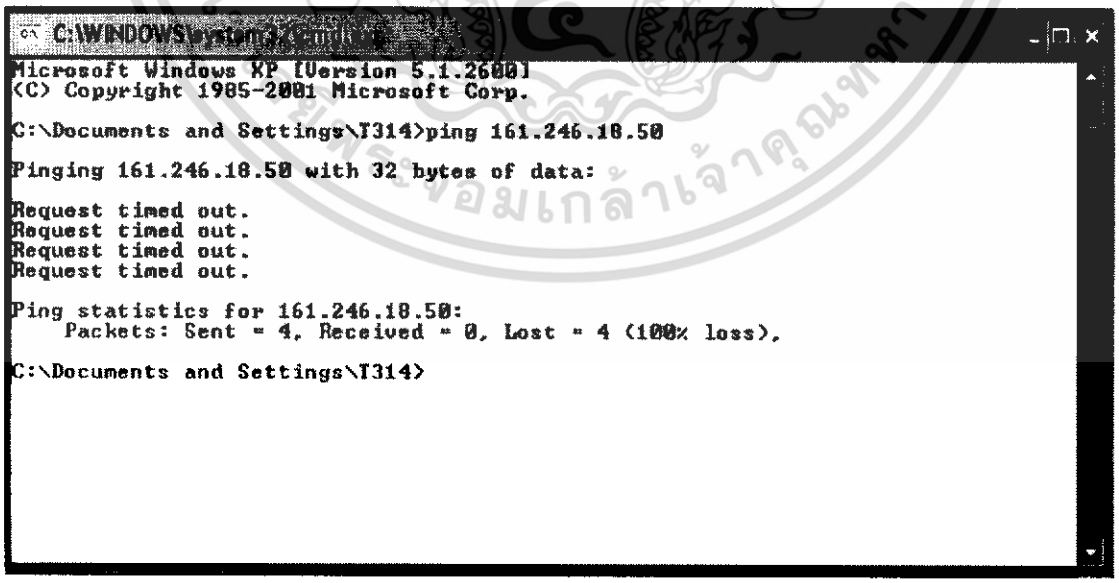
1. จากการทดลองทั้งหมดข้างต้นจะนำมารวมกันเพื่อให้อุปกรณ์แรบบิททั้งสองตัวทำหน้าที่เป็น IP Filter ซึ่งจะมีลักษณะการจัดวางอุปกรณ์ดังรูปที่ 5.8.1



รูปที่ 5.8.1 แสดงอุปกรณ์ IP Filter ที่ได้ออกแบบไว้

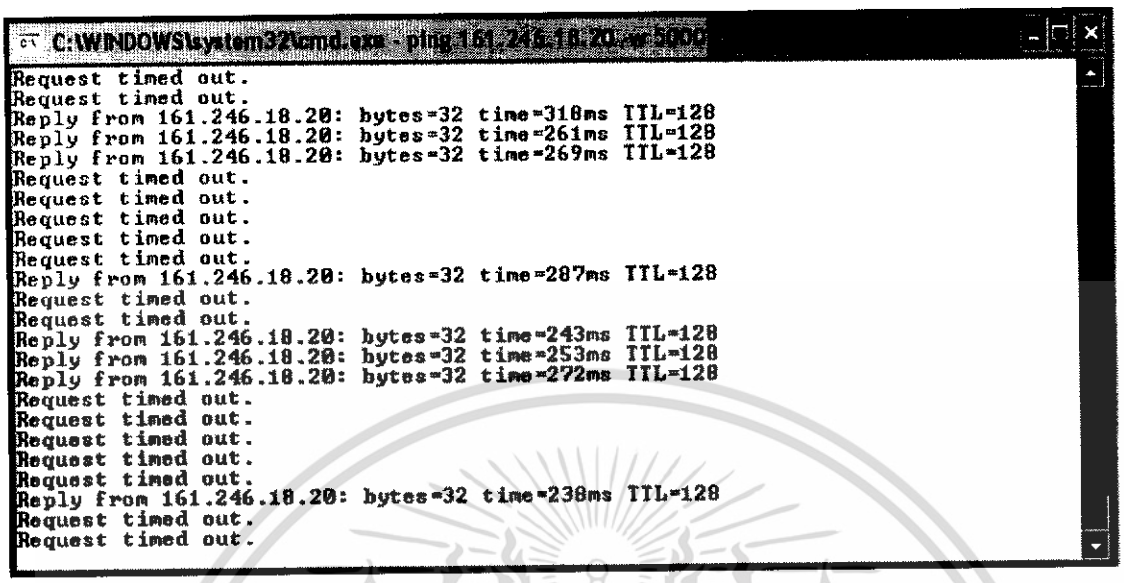
2. ทำการลงโปรแกรม IP Filter.c ไปยังอุปกรณ์แรบบิททั้งสองตัว

3. เมื่อทำการ Ping ไปยังหมายเลขไอพีแอดเดรสที่อยู่ในแบล็คลิสท์จะมีการแสดงผลดังรูปที่ 5.8.2 จะไม่สามารถ Ping ไปยังไอพีแอดเดรสปลายทางได้



รูปที่ 5.8.2 แสดงผลที่ไม่สามารถส่งข้อมูลไปยังปลายทางได้

4. เมื่อทำการ Ping ไปยังหมายเลข IP address ที่ไม่ได้อยู่ในแบล็คลิสท์จะมีการแสดงผลดังรูปที่ 5.8.3 จะ



รูปที่ 5.8.3 แสดงผลว่าสามารถส่งข้อมูลไปยังปลายทางได้

บทที่ 6

บทวิจารณ์และสรุป

จากการทดลอง เราสามารถสรุปการทำงานของไอพีฟิวเตอร์ ได้เป็นส่วนๆดังต่อไปนี้

- **รับข้อมูล (Input)** เป็นส่วนที่รับข้อมูลไอพีแพ็กเกตจากคอมพิวเตอร์เพื่อนำมาประมวลผลโดยจะมีการแยกเฮดเดอร์ของแพ็กเกต ออกเพื่อนำมาเปรียบเทียบกับค่าในแบล็คลิสท์เพื่อทำการตัดสินใจในการกรองแพ็กเกต
- **ประมวลผล (Microprocessor)** ทำหน้าที่ในการประมวลผล และเชื่อมต่อกับ Interface ต่างๆในที่นี้เราเลือกใช้ RCM 2200 โดยจะทำงานประมวลผลในการอ่านเขียนข้อมูลสู่หน่วยความจำ (Static RAM) แล้วมีการประมวลผลแพ็คเกตในการส่งต่อไปให้แรบบิทโมดูลอีกตัวหรือส่งออกพอร์ตอีเทอร์เน็ต
- **โปรแกรมมิ่ง (Programming)** ใช้เครื่องคอมพิวเตอร์ในการโปรแกรมมิ่งผ่านแฟลชโปรแกรม - เมเบิ้ล (Flash programmable) โดยต่อจากซีเรียลพอร์ท (Serial Port) ของเครื่องคอมพิวเตอร์เข้ากับตัวจัมป์ปิ้ง (Jumping) ซึ่งมีอยู่บนแรบบิทโมดูล

ระบบไอพีฟิวเตอร์ที่ได้ออกแบบนี้จะประกอบจากฟังก์ชันต่างๆ โดยเริ่มจากการกำหนดอินเตอร์เฟส (Interface) ให้กับอุปกรณ์การเซตโหมดโพรมิสคิวอัสโหมดโพรมิสคิวอัสเพื่อให้รับแพ็กเกตทุกชนิด ถึงแม้แพ็คเกตนั้นจะไม่ใช่แพ็กเกตของมันเองการตอบกลับเออาร์พีรีไพน์ โดยมีการเปลี่ยนหมายเลขไอพี ต้นทางของอุปกรณ์ RCM 2200 ให้เป็นหมายเลขไอพีปลายทางที่คอมพิวเตอร์ส่งมาเมื่อมีการตอบกลับเออาร์พีรีไพน์ ก็จะเริ่มมีการส่งข้อมูลจากคอมพิวเตอร์แล้วข้อมูลนั้นก็จะถูกรับแล้วนำมาถอดเฮดเดอร์ เพื่อไปใช้ในการวิเคราะห์และเปรียบเทียบกับแบล็คลิสท์เพื่อกรองหรือจะส่งต่อไปให้แรบบิทอีกตัวซึ่งต้องผ่านฟังก์ชันการส่งข้อมูลระหว่างอุปกรณ์และอุปกรณ์แรบบิทตัวที่รับข้อมูลจากอีกฝั่งก็จะทำหน้าที่ในการเขียนข้อมูลออกทางพอร์ตอีเทอร์เน็ตต่อไป

ปัญหาในการทดลอง

- เนื่องจากเป็นไมโครโปรเซสเซอร์ ใหม่ที่ยังมีผู้ใช้ไม่แพร่หลายมากนัก การศึกษาการใช้งานจึงค่อนข้างลำบากและใช้เวลานาน
- เกิดปัญหาเนื่องจากเกิดการดีเลย์ (delay) นานเกินไปซึ่งทำให้เกิดการสูญเสียของแพ็กเกตโดยสามารถคำนวณค่าทรูพุท (through put) ซึ่งจะได้เท่ากับ 0.26
- รายละเอียดในเอกสารที่มากับอุปกรณ์ขาดรายละเอียดสำคัญบางอย่างซึ่งทำให้เกิดปัญหาการติดขัดในการทำงานเช่นวิธีการเซต promiscuous mode
- เกิดปัญหาคอขวดเนื่องมาจากการรับข้อมูลจากเครือข่ายแลน มีความเร็วในการรับข้อมูลที่ 10 Mb/s แต่การรับส่งข้อมูลระหว่างแรบบิทโมดูลสองตัวนั้นเป็นการรับส่งข้อมูลทางซีเรียลพอร์ท ซึ่งเราสามารถกำหนดความเร็วได้ที่ 9600 b/s เท่านั้น เนื่องมาจากโปรแกรมที่เราเขียนขึ้นเพื่อใช้ควบคุมการทำงานของแรบบิทโมดูลมีจำนวนคำสั่ง (บรรทัด) มากและแต่ละ

คำสั่งอาจจะต้องใช้สัญญาณนาฬิกาหลายไซเคิล เป็นผลให้การประมวลผลนั้นต้องใช้เวลานานมากขึ้น ทำให้ไม่สามารถกำหนดบิตเรทที่มากกว่า 9600 b/s ได้เพราะจะทำให้การรับข้อมูลจากซีเรียลพอร์ทเกิดความผิดพลาด

แนวทางในการพัฒนา

- ปรับปรุงการเขียนโปรแกรมให้มีความรัดกุมมากขึ้น พยามยามลดหรือใช้คำสั่งที่กระชับมากขึ้น
- เพิ่มความเร็วของสัญญาณนาฬิกาให้มากขึ้น
- การเชื่อมต่อข้อมูลระหว่างแรบบิททั้งสองตัวอาจจะใช้การรับส่งแบบขนาน (Parallel)

เอกสารอ้างอิง

[1] www.zworld.com

[2] www.rabbitsemiconductor.com

[3] Jan Axelson."Embedded ETHERNET AND INTERNET COMPLETE".Lakeview research LLC,2003

ภาคผนวก

```
/* IP Filter This program burn with rcm2200 which is connect with computer*/
#define TCPCONFIG 100

#define MY_IP_ADDRESS "161.246.18.97" //define IP of RCM2200
#define MY_NETMASK "255.255.255.0"

#define PD_WR1  0xC000      // I/O Bank Control Register IB6CR Address
#define PD_RD1  0x4000          // I/O Bank Control Register IB2CR Address
#define PD_RCR 0x0C         // Receive Configuration Reg. W-0, R-2
#define PD_TCR  0x0D                // Transmit Configuration Reg. W-0, R-2

#define arp_TypeEther  0x100        /* ARP type of Ethernet address */

#define ARP_REPLY   0x0200
#define TIMEOUT 20UL
#define CINBUFSIZE  63
#define COUTBUFSIZE 63
#define DINBUFSIZE  63
#define DOUTBUFSIZE 63

#memmap xmem
#use "dcrtcp.lib"
#use "IP.LIB"
#use "arp.lib"
#use "icmp.lib"
unsigned int packetno;

#asm debug
pd_intt::
   push    af
   push    hl

   ld  a, 0x20
   ioe ld (PD_WR1),a                    ; abort DMA, nic#1

   printf("\nTest\n");
   ld  a, 0xFF
   ioe ld (PD_WR1+PD_INTSTATUS),a       ; clear all interrupts, nic#1

   pop hl
   pop af

   ipres
   ret
#endasm

void chg_hex(unsigned long inp, char* outp)
   outp[0] = inp >> 24;
   outp[1] = (0x00FFFFFF&inp) >> 16;
   outp[2] = (0x0000FFFF&inp) >> 8;
   outp[3] = 0x000000FF&inp;


void rtl_regpage(char pgnum)     // Change register page to pgnum
   char temp;

   temp = pgnum << 6;
   #asm /*debug*/
       LD      A, (temp)
       LD      B, A
       IOE LD  A, (PD_RD1)
       AND 0x3F
       OR      B
       IOE LD (PD_WR1), A // Change to pgnum register page
   #endasm


char rtl_getreg(char reg)        // Get value from reg register
   unsigned int temp;
   char ans;

   temp = PD_RD1+reg;
   #asm /*debug*/
       LD      HL, (temp)
       IOE LD A, (HL)
       LD      (ans), A
   #endasm
   return ans;
```

```
oid rtl_putreg(char reg, char inp) // Set reg register value = inp
   unsigned int temp;

   temp = PD_WR1+reg;
   #asm /*debug*/
       LD      HL, (sp+@sp+inp)
       LD      A, L
       LD      HL, (temp)
       IOE LD (HL), A
   #endasm


nt COMPARE_IP(longword ip1,longword ip2,longword ip3,longword ip4,longword ip5)

   if (ip1 == ip2)
   {
       return 1;
   }else if (ip1 == ip3)
   {
       return 1;
   }else if (ip1 == ip4)
   {
       return 1;
   }else if (ip1 == ip5)
   {
       return 1;
   }else
   {
       return 0;
   }

odebug void _arp_reply( longword ip, longword ip2 ,byte *comMAC)

  auto arp_Header *op;

  op = (arp_Header *)_eth_formatpacket(comMAC, 0x608);
  op->hwType = arp_TypeEther;
  op->protType = 0x008;     /* IP */
  op->hwProtAddrLen = sizeof(eth_address) + (sizeof(longword)<<8);
  op->opcode = ARP_REPLY;
  op->srcIPAddr = intel( ip );
  movmem(&_eth_addr, &op->srcEthAddr, sizeof(eth_address));
  op->dstIPAddr = intel(ip2);

  /* ...and send the packet */
  _eth_send( sizeof(arp_Header) );


har *manage_arppacket(void *inpbuff) //
   struct ether * buff;
   unsigned int framelen,i;
   char *pchar;
   byte *b_pchar,b_dip[4],b_sip[4],des_mac[6];
   longword *ldip_pchar, *lsip_pchar;
   buff = (struct ether *)inpbuff;
   if (buff != NULL && buff->type == 1544) // IP packet
   {   pchar = (char *)buff;
       b_pchar = (byte *)pchar;
       framelen = 42;
       if (*(pchar + 21) == 1)
       {
           printf("this is arp request\n");
           for (i=0;i<4;i++ )
           {
           b_dip[i] = *(b_pchar + 38 + i);
           }//end for i = 0
           for (i=0;i<4 ;i++ )
           {
           b_sip[i] = *(b_pchar + 28 + i);
           }
           for (i=0;i<6 ;i++ )
           {
           des_mac[i] = *(b_pchar + 6 + i);
           }
           ldip_pchar = &b_dip;
           lsip_pchar = &b_sip;
```

```
            if(COMPARE_IP(ntohl (*ldip_pchar),resolve("161.246.18.50"),resolve("161.246.18.
."),resolve("161.246.18.52"),resolve("161.246.18.53"))== 1)
            {
          printf("this is block ip");
            }else
            {
          printf("this is permit ip");
          _arp_reply( ntohl (*ldip_pchar),ntohl (*lsip_pchar),des_mac);

            }//end else
      }// end check request
   }//end if
'/end fn manage_arp packet

)id *send_icmpreply(char *buff)

1ar *icmp,buffer[74];
1signed int i;
:mp = &buffer;
   *(icmp) = *(buff+6);
   *(icmp + 1) = *(buff+7);
   *(icmp + 2) = *(buff+8);
   *(icmp + 3) = *(buff+9);
   *(icmp + 4) = *(buff+10);
   *(icmp + 5) = *(buff+11);
   *(icmp + 6) = *(buff);
   *(icmp + 7) = *(buff+1);
   *(icmp + 8) = *(buff+2);
   *(icmp + 9) = *(buff+3);
   *(icmp + 10) = *(buff+4);
   *(icmp + 11) = *(buff+5);
   for (i=12;i<26;i++)
   {
   *(icmp + i) = *(buff + i);
   }
   for (i=26;i<30;i++ )
   {
   *(icmp + i) = *(buff+4+i);
   }
   for (i=30;i<34 ;i++ )
   {
   *(icmp + i) = *(buff+i-4);
   }
   *(icmp + 34) = 0;
   //place fn receive from slave for set condition
   *(icmp + 35) = 0;
   *(icmp + 36) = *(buff + 36) + 8;
   for (i=37;i<74 ;i++ )
   {
   *(icmp + i) = *(buff + i);
   }
   pkt_send( icmp, 74 );

'place fn s_pack
'place fn r_pack
1ar *manage_icmppacket(void *inpbuff)

   struct ether * buff;
   unsigned int framelen,i,*piplen;
   char *pchar,mssg[6];
   longword *ldip_pchar, *lsip_pchar;
   byte *b_pchar,b_dip[6],b_sip[4],icmpc1,icmpc2;
   buff = (struct ether *)inpbuff;
   if (buff != NULL && buff->type == 8)
       {
       pchar = (char *)buff;
       b_pchar = (byte *)pchar;
       for (i=0;i<4;i++ )
           {
           b_dip[i] = *(b_pchar + 30 + i);
           }//end for i = 0

           for (i=0;i<4 ;i++ )
           {
           b_sip[i] = *(b_pchar + 26 + i);
           }
       ldip_pchar = &b_dip;
       lsip_pchar = &b_sip;
```

```
      if (*(pchar + 23) == 1)
        {
      printf("This is ICMP PROTOCOL\n");  // ICMP packet
      piplen = pchar+16;                  // Pointer to IP total length
      framelen = ntohs(*piplen) + 14; // IP total length + Ethernet length
             if (*(pchar + 34) == 8)
             {
                 b_dip[4] = 0xab;
                 b_dip[5] = 0;
                 printf("This is ICMP REQUEST\n");
                 serCopen(9600);
                 serDopen(9600);
/            if (s_pack(b_dip) == 1)
                 serDputs(b_dip);
                 while ((serDread(mssg, 5, TIMEOUT)) < 5) ;
                 mssg[5] = 0;
                 if (mssg[4] == 0xbb)
                 {
                     send_icmpreply(pchar);
                 }
             }//end if check request
          }// end if check icmp protocol
      }// end if NULL
//end fn manage_icmp


/ êèÇ¹â»Ãá¡ÃÁËÅÑ¡
ain()
   {
   struct ether *pbuff;
   char rcrval, tcrval,mssg[6];
/  char *buffer;
   unsigned int count,i;
   longword seq,time_out,*desip,tmp_seq;
   packetno = 0;
   rtl_regpage(2);
   rcrval = rtl_getreg(PD_RCR);
   tcrval = rtl_getreg(PD_TCR);
   serCopen(9600);
   serDopen(9600);
   rtl_regpage(0);
   rtl_putreg(PD_RCR, 0xDC);        // Promiscuous mode and multicast accepted

       sock_init();
       while(1)
       {
       costate {
           pbuff = (struct ether *) pkt_received();
       if (pbuff != NULL)
       {
           printf("\nReceive #%d, type: %d\n", ++packetno, pbuff->type);
       //  if (pbuff->type == 1544)manage_arppacket(pbuff);    // ARP packet
       //  if (pbuff->type == 8)manage_icmppacket(pbuff);
       } /*end if pbuff != NULL*/else yield;
                 }

   costate {
           while ((serDread(mssg, 5, TIMEOUT)) < 5) {yield;}
           printf("Received 5 byte from serial port\n");
           mssg[5] = 0;
           if (mssg[4]==0xab)
           {
           printf("mssg[4] = 0xab\n");
           desip = &mssg;
           _ping(ntohl(*desip),seq++);
           time_out=_chk_ping(ntohl(*desip),&tmp_seq);
           if(time_out!=0xffffffff)
               {
               printf("timeout != 0xffffffff\n");
                   mssg[4] = 0xbb;
                   serDputs(mssg);
               }
           }
           }//end  costate

       costate     {
           tcp_tick(NULL);
                   }//end costate
```