การจัดการทราฟฟิกบนอินเตอร์เนต โดยใช้บริการแถวรอคอย

INTERNET TRAFFIC MANAGEMENT

BASED ON QUEUING SERVICES

โดย

นาย ปวีณ ศรีวิโรจน์

นาย มงคล เฟื่องฟูตระกูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

การจัดการทราฟฟิกบนอินเตอร์เนต โดยใช้บริการแถวรอคอย

INTERNET TRAFFIC MANAGEMENT BASED ON QUEUING SERVICES

โดย

นาย ปวีณ ศรีวิโรจน์ 43015076

นาย มงคล เฟื่องฟูตระกูล 43015081

อาจารย์ที่ปรึกษา

ผศ.ดร. สุทธิชัย นพนาคีพงษ์

อาจารย์ นภัทร สระเอี่ยม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การจัดการทราฟฟิกบนอินเตอร์เนตโดยใช้บริการแถวรอคอย

INTERNET TRAFFIC MANAGEMENT BASED ON QUEUING SERVICES

ผู้จัดทำ

1. นาย ปวีณ ศรีวิโรจน์ 43015076
2. นาย มงคล เฟื่องฟูตระกูล 43015081

.....อาจารย์ที่ปรึกษา

(ผศ.ดร. สุทธิชัย นพนาคีพงษ์)

.....อาจารย์ที่ปรึกษา

(อาจารย์ นภัทร สระเอี่ยม)

การจัดการทราฟฟิกบนอินเตอร์เนตโดยใช้บริการแถวรอคอย

INTERNET TRAFFIC MANAGEMENT BASED ON QUEUING SERVICES

โดย นาย ปวีณ ศรีวิโรจน์ 43015076
นาย มงคล เฟื่องฟูตระกูล 43015081
อาจารย์ที่ปรึกษา ผศ.ดร. สุทธิชัย นพนาคีพงษ์
อาจารย์ นภัทร สระเอี่ยม

บทคัดย่อ

การติดต่อสื่อสารข้อมูลได้เข้ามามีบทบาทต่อชีวิตประจำวันเพิ่มมากขึ้นเรื่อยๆ เช่น การใช้งานจดหมายอิเล็กทรอนิกส์(E-mail),การเรียกดูข้อมูลต่างๆในอินเตอร์เนต, การอัพโหลดและดาวน์โหลดข้อมูลต่างๆ ปัญหาสำคัญคือ เมื่อเราเรียกใช้บริการข้อมูลแบบใดในปริมาณมากๆ ทำให้ข้อมูลชนิดอื่นๆมีความเร็วในการรับส่งข้อมูลน้อยลง ดังนั้น โครงงานนี้จึงจัดทำขึ้นเพื่อจำลองการแก้ไขปัญหาดังกล่าวให้ดีขึ้นโดยนำข้อมูลของโปรโตคอล 3 ชนิด คือ HTTP,SMTPและโปรโตคอลอื่นๆนอกเหนือจากนี้ มาทำการจัดเรียงคิว โดยใช้เครื่องคอมพิวเตอร์ที่ติดตั้งระบบปฏิบัติการลีนุกซ์ (Linux) มาเชื่อมต่อระหว่างเน็ตเวิร์คภายในองค์กร กับ Gateway แล้วเขียนโปรแกรมด้วย คอมไพเลอร์ ภาษา C รับ Data Packet แล้วทำการจัดคิวก่อนส่งออกไปที่ Gateway

Abstract

Nowaday, the data communication such as an electronic mail (E-mail), surfing the internet including, upload and download data are extensive to the internet users. However, when these services are very much, the speed of each service is reduced. To overcome this problem, the simulation of the data queuing systems for 3 protocols the HTTP, the SMTP and the one protocol else. The computer based on Linux operating system is connected between network organization and gateway to receive the data packets and then the queue of the data packets by C program. Later, these queued data packets are sent out to the gateway.

สารบัญ

สารบัญภาพ

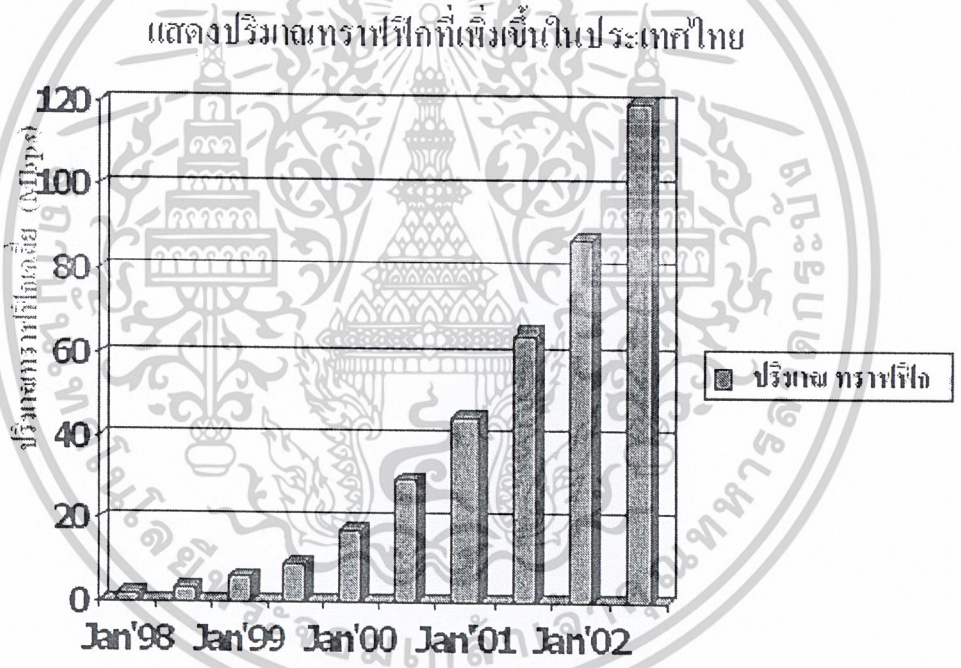
สารบัญภาพ (ต่อ)

สารบัญตาราง

บทที่1
บทนำ

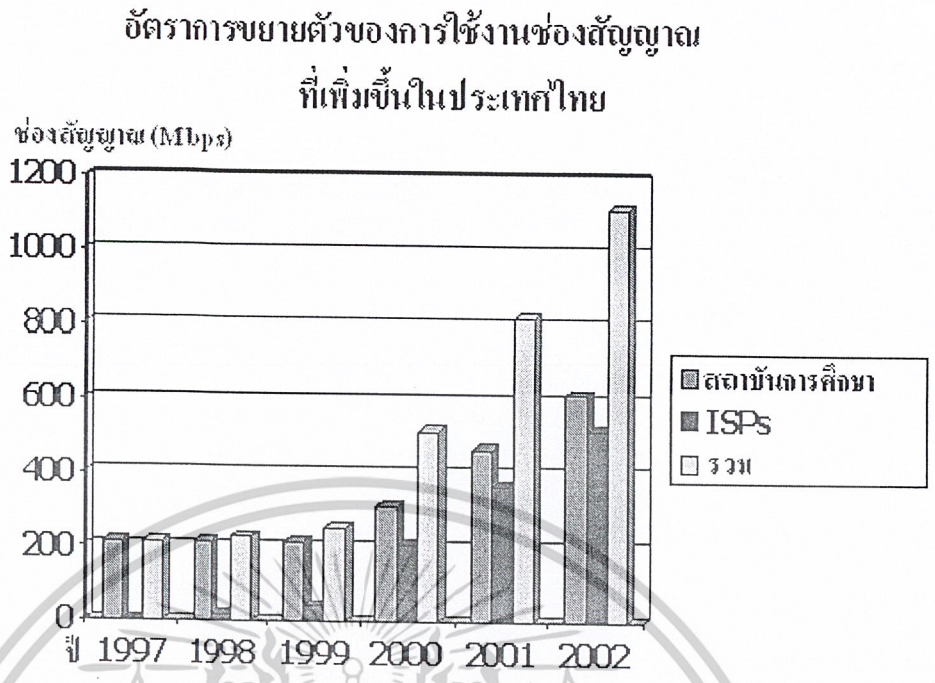
1.1 ความสำคัญและที่มา

ในโลกปัจจุบันนี้ คอมพิวเตอร์ได้เข้ามามีบทบาทเกี่ยวข้องกับชีวิตประจำวันมากขึ้น และเกือบจะทุกหนทุกแห่งมีการใช้งานคอมพิวเตอร์กับการใช้งานที่ซับซ้อนเพื่ออำนวยความสะดวกสบายขึ้น ซึ่งในทุกวันนี้การติดต่อสื่อสารด้วยคอมพิวเตอร์ก็กลายเป็นส่วนหนึ่งในการทำงานที่แทบจะขาดไม่ได้ไปแล้ว ไม่ว่าจะเป็นการติดต่อสื่อสารระดับย่อย อย่างเช่น เครื่องต่อเครื่อง หรือแบบ หลายๆเครื่องในองค์กรต่างๆ และเราก็สามารถที่จะติดต่อกับเครื่องคอมพิวเตอร์ใดๆก็ได้ผ่านเครือข่ายขนาดใหญ่มากอย่างเช่น เครือข่ายอินเตอร์เน็ต ซึ่งการใช้งานในระดับต่างๆนั้นมีการขยายตัวมากขึ้นทุกวัน



รูปที่ 1.1 แสดงการขยายตัวโดยเฉลี่ยของเครือข่ายในประเทศไทย
(ที่มาจาก http://ntl.nectec.or.th/internet)

ซึ่งจากรูปที่1.1แล้ว จะเห็นได้ว่าปัญหาที่เกิดตามมาก็คือความหนาแน่นของข้อมูลที่ต้องการจะติดต่อสื่อสาร ซึ่งในองค์กรหนึ่งๆนั้นมีความต้องการใช้งานเครือข่ายที่มากขึ้น ในขณะที่เซิร์ฟเวอร์ (Server) และสายนำสัญญาณนั้นมีช่องสัญญาณที่จำกัด ถึงแม้ว่าจะมีการขยายตัวได้แต่ก็ไม่เพียงพอต่อความต้องการใช้งานในปัจจุบันซึ่งจะเห็นได้จาก รูปที่1.2



รูปที่1.2 รูปแสดงการขยายช่องสัญญาณของเครือข่ายในประเทศไทย

(ที่มาจาก http://ntl.nectec.or.th/internet)

ซึ่งจะเห็นได้ว่าจากการที่มีช่องสัญญาณที่จำกัดนั้นจะต้องใช้เวลานานในการที่จะให้การตอบสนอง หรือให้บริการข้อมูลในเครือข่ายให้ทั่วถึงกัน

ปริญญานิพนธ์เล่มนี้ นำเสนอรูปแบบของการจำลอง การจัดการสื่อสารข้อมูลที่มีความหนาแน่นในระบบเครือข่าย โดยการพัฒนาโปรแกรมบนเครื่องคอมพิวเตอร์ที่ติดตั้ง ระบบปฏิบัติการ ลีนุกซ์ (Linux) โดยสาเหตุที่เลือกใช้เพราะว่าข้อดีที่มีมากมายของตัวลีนุกซ์ เองที่เป็นโปรแกรมเสรี (freeware) ทำให้ไม่ต้องเสียค่าใช้จ่ายซื้อระบบปฏิบัติการมาใช้งานและลีนุกซ์ เป็นระบบปฏิบัติการที่ใช้ทรัพยากรในตัวเองน้อย ทำงานได้อย่างมีประสิทธิภาพ มีความเสถียรภาพ มั่นคงในการทำงาน พิสูจน์แล้วว่าใช้ได้ดี ใช้งานทรัพยากรน้อย ทำให้เครื่อง PC ขนาดเล็กเป็นเซิร์ฟเวอร์ได้

ซึ่งระบบปฏิบัติการลีนุกซ์ ใช้โปรโตคอลในการติดต่อสื่อสารระหว่างกันคือ โปรโตคอล TCP/IP ซึ่งจะนำโปรโตคอลย่อยของ TCP/IP ที่นิยมใช้งานกันบ่อย 3 ชนิด คือ โปรโตคอล HTTP ที่ใช้งานในการเรียกดูเว็บไซท์ต่าง, โปรโตคอล SMTP ที่ใช้งานในจดหมายอิเล็กทรอนิกส์ หรือ อีเมล์ (Email) และ โปรโตคอลของการใช้งานประเภทอื่นๆ มาทำการจำลองการให้บริการระบบแถวรอคอย (Queuing Service) ในวิธีการแบบต่างๆ บนโปรแกรมจำลอง Network Simulator (NS) เพื่อสังเกตประสิทธิภาพของเครือข่ายที่ได้รับการบริการระบบแถวรอคอยแบบต่างๆกันว่ามีการลดความหนาแน่นของข้อมูลได้ดีเพียงใด

1.2 วัตถุประสงค์

ทุกวันนี้ การติดต่อสื่อสารระหว่างคอมพิวเตอร์ เป็นเรื่องที่จำเป็นและสำคัญไปแล้ว และข้อมูลที่ใช้ติดต่อสื่อสารกันนั้น ก็มีหลายชนิดหลายรูปแบบทำให้เกิดความไม่เป็นธรรมในการที่จะใช้งานในช่อง

สัญญาณที่มีอยู่อย่างจำกัดนั้น และเมื่อมีการใช้งานสูงมากในระบบเครือข่ายโดยจากเครื่องลูกข่ายทั่วๆไป นั้นจะทำให้มีการติดขัดของ packets ข้อมูลที่ Router อย่างมาก ซึ่งวัตถุประสงค์ของปริญญานิพนธ์นี้คือทำการจำลองวิธีการทำงานและศึกษาถึงข้อดีข้อเสียจากวิธีการจัดการคิวแบบ Random Early Detection (RED) ซึ่งเป็นวิธีที่นิยมใช้กันมากในปัจจุบันในการที่จะจัดการปริมาณของทราฟฟิกที่ Router โดยจะศึกษาถึงความสัมพันธ์ของค่าพารามิเตอร์ที่เกี่ยวข้องเพื่อที่จะสามารถทำการปรับแต่งในอุปกรณ์จริงได้อย่างมีประสิทธิภาพ และทั้งยังสามารถออกแบบและพัฒนาจากวิธีดั้งเดิมของ RED ให้มีประสิทธิภาพในการทำงานที่ดีขึ้นได้

1.3 ขอบเขต

รายละเอียดของเนื้อหาได้นำเสนอเกี่ยวกับส่วนที่เกี่ยวข้องกับ รายละเอียดพื้นฐานของ ระบบปฏิบัติการ โปรแกรม Network Simulator , โปรโตคอลพื้นฐานที่ทำงานภายใต้โปรโตคอล TCP/IP, การจัดการโครงสร้างของข้อมูลแบบ คิว ในการออกแบบโปรแกรมในการจำลองการจัดการข้อมูลภายในเครือข่ายภายใต้วิธีการทำงานของคิวชนิด Random Early Detection (RED) ทำการพัฒนาวิธีการแบบดั้งเดิมให้มีประสิทธิภาพได้ดีขึ้น

1.4 วิธีดำเนินการ

ในการดำเนินงาน มีการรวบรวมข้อมูลและรายละเอียดเกี่ยวกับการทำงานและโครงสร้างของข้อมูล, โปรโตคอลในรูปแบบต่างๆ รวมถึงศึกษาการทำงานของ วิธีการจัดการคิวในแบบต่างๆเพื่อใช้ในการเขียนโปรแกรมบนโปรแกรม Network Simulator เพื่อใช้ในการจัดการข้อมูลตามโครงสร้างแบบคิวที่เราจะเลือกใช้

- **บทที่สอง** อธิบายถึงทฤษฎีและหลักการในการสื่อสารระหว่างกันบนเครือข่ายอินเตอร์เนต และโปรแกรมที่ใช้ในการจำลองสร้างระบบเครือข่ายขึ้นมา
- **บทที่สาม** อธิบายถึงหลักการลดความหนาแน่นของทราฟฟิกหรือการทำ (Quality of Service : QoS), วิธีการจัดการข้อมูลแบบคิว (Queuing Management) บางส่วนที่ใช้งานในปัจจุบัน
- **บทที่สี่** แสดงการออกแบบการจัดการคิวแบบ Random Early Detection (RED) ลักษณะการทำงาน, ความสัมพันธ์ของค่าพารามิเตอร์ต่างๆที่จะทำการปรับแต่ง, กราฟแสดงถึงปริมาณทราฟฟิกใน Router และ Throughput, การพัฒนาปรับปรุงวิธีการจัดการให้ดีขึ้น
- **บทที่ห้า** บทสรุปและวิจารณ์

บทที่ 2
ทฤษฎีและหลักการ

2.1 โปรโตคอล (Protocol)

โปรโตคอล (Protocol) คือ ระเบียบวิธีที่กำหนดขึ้นสำหรับการสื่อสารข้อมูล โดยสามารถส่งผ่านข้อมูลไปยังปลายทางได้อย่างถูกต้อง ในปัจจุบันโปรโตคอลในการสื่อสารข้อมูลก็มีอยู่หลายโปรโตคอล นอกเหนือจาก TCP/IP คล้ายกับภาษาต่างๆในโลกนี้ที่นอกจากภาษาอังกฤษแล้วก็ยังมีภาษาจีน ญี่ปุ่น ฝรั่งเศส เยอรมัน และอื่นๆอีกมากมาย ในด้านโปรโตคอลสื่อสารข้อมูลก็เช่นกัน ได้มีการออกแบบโปรโตคอลอื่นๆขึ้นมาใช้งานอีกมาก เช่น โปรโตคอล IPX/SPX, โปรโตคอล NetBEUI และ โปรโตคอล AppleTalk เป็นต้น ซึ่งในการใช้งานปัจจุบันของ internet รวมถึงในโครงงานนี้ได้ใช้ โปรโตคอล TCP/IP ที่จะกล่าวถึงต่อไปนี้

2.1.1 โปรโตคอล TCP/IP

โปรโตคอล TCP/IP เป็นชื่อเรียกของชุดโปรโตคอลที่สำคัญ มีการใช้งานกันอย่างแพร่หลายตามการขยายตัวของอินเตอร์เน็ต/อินทราเน็ต ความจริงแล้วโปรโตคอล TCP/IP เป็นกลุ่มของโปรโตคอลหลายตัวที่ประกอบกันเป็นชุดให้ใช้งาน โดยมีคำเต็มว่า Transmission Control Protocol / Internet Protocol ซึ่งจากชื่อเต็มทำให้เราเห็นว่า อย่างน้อยก็มีโปรโตคอลประกอบกันทำงานร่วมกัน 2 โปรโตคอลคือ TCP และ IP

ตัวอย่างของกลุ่มโปรโตคอลในชุดของ TCP/IP ที่เราพบและใช้งานบ่อยๆ (ส่วนใหญ่เราจะไม่ได้ใช้งานตรงๆ แต่จะใช้งานผ่านแอพพลิเคชั่นต่างๆหรือใช้งานโดยทางอ้อม) เช่น Internet Protocol (IP), Address Resolution Protocol (ARP), Internet Message Control Protocol (ICMP), User Datagram Protocol (UDP), Transport Control Protocol (TCP) และ Simple Mail Transfer Protocol (SMTP) เป็นต้น

โปรโตคอลที่มีบทบาทสำคัญในการทำงานในเครือข่ายอินเตอร์เน็ต คือ Internet Protocol (โปรโตคอล IP) เนื่องจากเมื่อโปรโตคอลอื่นๆต้องการส่งข้อมูลข้ามเครือข่ายอินเตอร์เน็ตนั้น จะต้องอาศัยการผนึกข้อมูล (Encapsulation) ไปกับโปรโตคอล IP ที่มีกลไกการระบุเส้นทาง (Route Service) ผ่าน Gateway หรือ Router เพื่อนำข้อมูลไปยังเครือข่ายและเครื่องปลายทางที่ถูกต้องเนื่องจากกลไกการระบุเส้นทางจะทำงานที่โปรโตคอล IP เท่านั้น และด้วยเหตุนี้เราจึงเรียก IP ว่าเป็นโปรโตคอลที่มีความสามารถระบุเส้นทางการส่งผ่านของข้อมูลได้ (routable)

การอ้างอิงอุปกรณ์ในเครือข่าย

แนวความคิดหลักของระบบเครือข่ายคอมพิวเตอร์คือ การเชื่อมโยงอุปกรณ์เข้าด้วยกัน ไม่ว่าจะเป็นเครื่องเซิร์ฟเวอร์ที่ให้บริการ (หรือบางที่เรียกว่า host) และอุปกรณ์ในเครือข่ายอื่นๆ เช่น Router, เครื่องพิมพ์ เพื่อให้สามารถแชร์การใช้อุปกรณ์ร่วมกันได้ หรือสามารถส่งผ่านข้อมูลไปมาระหว่างกันได้ถูกต้อง เมื่อมีการเชื่อมต่อกันแล้วก็จำเป็นต้องมีการกำหนดหรือระบุหมายเลขของอุปกรณ์ทุกชิ้นทุกชนิด

ในเครือข่าย เพื่ออ้างอิงได้โดยไม่ซ้ำกัน เพราะถ้าซ้ำกันแล้วการรับส่งข้อมูลอาจจะไม่ถึงมือผู้รับปลายทางได้อย่างถูกต้อง เลขหมายดังกล่าวจะเรียกว่า แอดเดรส (address) หรือเลขหมายประจำตัวที่มีข้อกำหนดเป็นมาตรฐาน ซึ่งในการใช้งานโปรโตคอล TCP/IP ที่เชื่อมโยงเครือข่ายอินเตอร์เน็ตนี้ เลขหมายที่ใช้อ้างถึงกันจะใช้เป็นตัวเลขที่เรียกว่า IP Address (Internet Protocol Address)

2.1.2 ไอพี แอดเดรส (IP Address)

IP Address ถูกกำหนดให้เป็นหมายเลขอ้างอิงประจำตัวของอุปกรณ์ต่างๆที่เชื่อมต่ออยู่ในเครือข่ายอินเตอร์เน็ต โดยการกำหนด IP Address ให้แต่ละเครื่องหรือแต่ละอุปกรณ์นี้จะต้องไม่ซ้ำกัน ซึ่ง IP Address นี้จะไม่ผูกติดอยู่กับตัวฮาร์ดแวร์แต่อย่างใด จึงสามารถกำหนดใหม่ หรือแก้ไขเปลี่ยนแปลงได้เมื่อมีการเปลี่ยนตัวฮาร์ดแวร์ ทั้งนี้เนื่องจากเป็นการกำหนดด้วยตัวซอฟท์แวร์แตกต่างกับหมายเลข MAC address (Media Access Control address) ซึ่งเป็นหมายเลขประจำตัวของอุปกรณ์ที่ต่ออยู่ในเครือข่าย ค่า MAC address จะถูกกำหนดจากบริษัทผู้ผลิตอุปกรณ์ตั้งแต่เริ่มผลิต เช่น อุปกรณ์ Network Interface Card (NIC) จะมีค่า MAC address ประจำตัวที่ไม่ซ้ำกันและไม่สามารถแก้ไขได้ ค่า MAC address เป็นการระบุค่าอ้างอิงอุปกรณ์ฮาร์ดแวร์ในระดับล่างสุด (Physical Layer) ของกลไกการรับส่งข้อมูลภายในเครือข่าย ถ้าจะใช้หมายเลข MAC address สำหรับระบุอ้างอิงกันกันภายในเครือข่ายแล้วจะเกิดปัญหามาก เมื่อมีการเปลี่ยนหรือย้ายเครื่องต้องทำการกำหนดระบบเครือข่ายใหม่ (configuration) นอกจากนี้ยังจดจำได้ยากกว่า ตัวอย่างของหมายเลข MAC address คือ 08:0a:0e:12:b5:05 การที่ IP Address ถูกใช้อ้างอิงในการติดต่อกันด้วยโปรโตคอล TCP/IP เพราะการใช้ IP Address จะยืดหยุ่นและคล่องตัวกว่า

การทำงานของโปรโตคอล IP จำเป็นจะต้องอาศัย IP Address เพื่อระบุและอ้างถึงอุปกรณ์ต่างๆที่ต่ออยู่ในเครือข่าย IP Address จะเป็นค่าตัวเลขขนาด 32 บิต ถูกแบ่งออกเป็นส่วนละ 8 บิต รวมเป็น 4 ส่วน และคั่นแต่ละส่วนด้วยเครื่องหมายจุด (.) ดังนั้นค่าตัวเลขในแต่ละส่วนจะมีได้ตั้งแต่ 0 ถึง 255 นอกจากนี้ IP Address บางช่วงจะมีลักษณะการใช้งานในลักษณะความหมายและหน้าที่พิเศษออกไปในการทำงานของโปรโตคอล TCP/IP เช่น IP Address ที่ 127.0.0.0 เป็น IP Address ที่ใช้ทำหน้าที่เป็น loop back address คือใช้กำหนดค่า loop back หรือแอดเดรสย้อนกลับให้กับอุปกรณ์นั้น

ค่าของ IP Address จะถูกกำหนดออกเป็น 2 ความหมายคือ ค่าของ หมายเลขอุปกรณ์ในเครือข่าย (host address) และค่าของ หมายเลขเครือข่าย (network address) ตัวอย่างเช่นมีเครื่องเว็บเซิร์ฟเวอร์เชื่อมต่ออยู่ในเครือข่าย 2 เครื่อง โดยแต่ละเครื่องมี IP Address ประจำตัวคือ 205.144.78.2 และ 205.144.78.3 ตามลำดับ เครื่องทั้งสองมีค่าของหมายเลขเครือข่ายเหมือนกัน คือ 205.144.78 แสดงว่าเครื่องทั้งสองต่อเชื่อมอยู่ในเครือข่ายเดียวกัน บนสายสัญญาณ ที่เชื่อมโยงเส้นเดียวกันแต่มีหมายเลขประจำตัวเครื่องที่แตกต่างกัน คือ 2 และ 3 ตามลำดับ

2.1.3 ดาต้า แพ็คเกจ (Data Packet)

เมื่อมีการรับหรือส่งข้อมูลกันในระบบเครือข่ายอินเตอร์เน็ตนั้น ตัวข้อมูล (เช่น ข้อความในอีเมล์ หรือไฟล์ HTML ที่เห็นในโปรแกรมบราวเซอร์) จะถูกทำให้มีขนาดเล็กลงโดยแบ่งย่อยเป็นส่วนย่อยๆ เรียกว่า Data Packet หรือ datagram ซึ่งการจัดแบ่งข้อมูลให้เป็นส่วนย่อยลงนี้มีประโยชน์คือทำให้เครือข่ายนั้นสามารถรองรับการติดต่อและรับส่งข้อมูลกันได้อย่างราบรื่นไม่ติดขัด หรือพบปัญหาเครือข่าย

ทำงานช้าเมื่อมีการรับส่งข้อมูลขนาดใหญ่ เนื่องจากสายสัญญาณเชื่อมโยงเป็นสื่อที่ต้องแบ่งกันใช้งาน นอกจากนี้การแบ่งข้อมูลออกเป็นส่วนย่อยๆยังสามารถเพิ่มกระบวนการตรวจทานความถูกต้องของข้อมูลที่ปลายทาง และแก้ไขข้อมูลผิดพลาดหรือตกหล่นได้โดยง่ายอีกด้วย

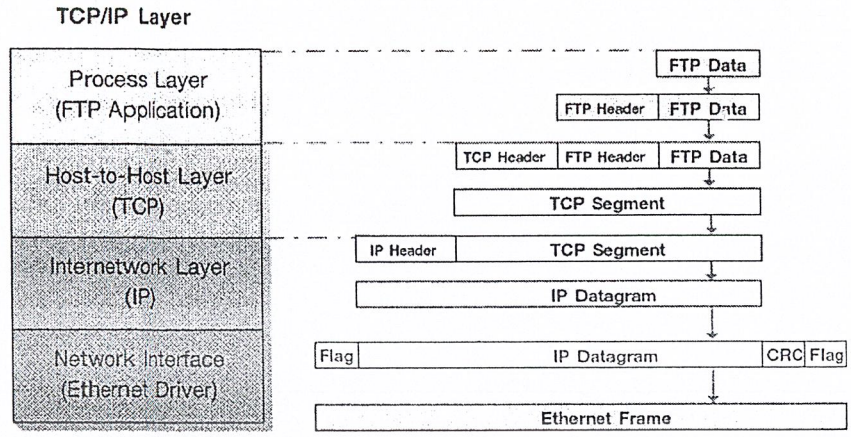
เมื่ออุปกรณ์ใดต้องการส่งข้อมูล อุปกรณ์อื่นๆก็ต้องรอให้การส่งข้อมูลนั้นเสร็จสิ้นก่อนจึงจะสามารถส่งข้อมูลของตนได้ โดยเฉพาะอย่างยิ่งในกรณีที่มีการส่งข้อมูลขนาดใหญ่ ถ้าไม่มีการแบ่งข้อมูลให้เป็นส่วนเล็กเพื่อทยอยส่งไปยังปลายทาง โดยเป็นการแบ่งเวลาให้อุปกรณ์อื่นๆได้ใช้สายด้วยแล้ว เครือข่ายนั้นอาจเกิดปัญหาติดขัดได้ ทั้งนี้เมื่อ datagram ถูกส่งไปยังปลายทางแล้ว ก็จะมีกระบวนการรวมข้อมูลย่อยเหล่านี้ให้กลับคืนสู่สภาพเดิมได้ต่อไป

ประโยชน์อีกประการหนึ่งในการแยกข้อมูลให้เป็นส่วนย่อยๆคือ การแก้ไขและตรวจสอบข้อมูลที่เสียหายในการส่งข้อมูล จะสามารถทำได้อย่างมีประสิทธิภาพ ซึ่งการส่งข้อมูลผ่านสายสัญญาณต่างๆเรามักจะพบปัญหาสัญญาณรบกวนหรือสัญญาณขาดหายไประหว่างการส่งอยู่บ่อยๆ ทำให้ข้อมูลที่ส่งไปยังผู้รับไมถูกต้องครบถ้วน ซึ่งปัญหานี้สามารถแก้ไขได้ เนื่องจากข้อมูลที่ถูกแบ่งเป็น datagram จะมีขนาดเล็กลง ทำให้สามารถเพิ่มการตรวจสอบการรับส่งข้อมูลนั้นๆได้ดีขึ้น เช่น เทคนิคการคำนวณ (check-sum) จะคำนวณค่าของข้อมูลที่ส่งไปและได้รับ ถ้าตรงกันแสดงว่าการรับส่ง datagram นั้นถูกต้องแต่ถ้าผลการคำนวณไม่ตรงกัน ด้านผู้รับจะส่งสัญญาณมาให้เพื่อส่งเฉพาะ datagram นั้นใหม่อีกครั้ง โดยไม่ต้องส่งข้อมูลทั้งหมดมาอีก ทำให้สามารถแก้ไขข้อมูลที่ผิดพลาดได้อย่างรวดเร็ว

ตัวข้อมูลที่ถูกแยกออกเป็น data packet หรือ datagram นี้จะมีลักษณะเป็นข้อมูลแบบต่อเนื่อง (stream byte) คือมีการกำหนดลำดับก่อนหลังของข้อมูล เพื่อให้ประกอบข้อมูลย่อยคืนสู่สภาพเดิมได้อย่างถูกต้อง และมีรูปแบบหรือฟอร์แมต (format) ที่แน่นอน คือ datagram จะประกอบด้วยส่วนของ header และส่วนของตัวข้อมูล (body) โดยในส่วนของ header ต่างๆที่ระบุที่อยู่ปลายทางที่ต้องส่งข้อมูลไป, เลขหมายต้นทางที่ส่งข้อมูลมา, ค่าบอกขนาดความยาวของ datagram นี้ และข้อมูลอื่นๆ สำหรับในส่วนของ body อาจเป็นเนื้อหาข้อมูลใดๆ เช่น ข้อความในอีเมล์, ไฟล์ข้อมูลบางส่วน หรืออาจเป็น datagram ของข้อมูลอื่นๆที่ถูกผนึก (encapsulation) มาด้วยเป็นต้น ซึ่ง datagram ที่ใช้ในเครือข่ายอินเตอร์เน็ตนี้จะเรียกว่า IP datagram

2.1.4 การ Encapsulation

ก่อนที่ข้อมูลใดจะถูกส่งผ่านไปในเครือข่ายอินเตอร์เน็ตได้ ก็จะต้องถูกแยกเป็นส่วนย่อยๆเรียกว่า datagram และถูกผนึกหรือทำ encapsulation เข้าไปกับโปรโตคอล IP หรือเรียกว่าเป็น IP datagram ก่อนจึงจะส่งผ่านไปในเครือข่ายอินเตอร์เน็ตได้ เนื่องจากโปรโตคอล IP มีข้อมูลในการระบุเส้นทางการส่งผ่านไปยังปลายทางได้นั่นเอง การผนึกข้อมูลหนึ่งไปเป็นข้อมูลอีกรูปแบบหนึ่งนี้ เป็นกลไกที่สำคัญของการใช้งานโปรโตคอล TCP/IP มาก โดยขบวนการที่ใช้จะมีขั้นตอนคร่าวๆดังรูปต่อไปนี้

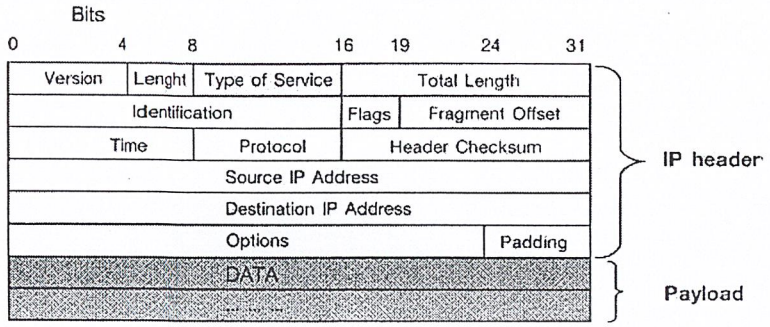


รูปที่ 2.1 ตัวอย่างการ Encapsulation ของข้อมูล FTP เทียบกับ TCP/IP layer

ตามรูปที่ 2.1 เริ่มต้นมีการใช้งานโปรแกรมรับส่งข้อมูล เช่น เมื่อเรียกใช้โปรแกรม FTP โปรแกรมแอพพลิเคชั่นจะเตรียมข้อมูลเพื่อส่งผ่านไปในเครือข่ายอินเตอร์เน็ตหลังจากโปรเซส FTP เตรียมข้อมูลและแยกส่วนเป็น FTP data หรือ FTP datagram แล้ว จะมีส่วนของ FTP header เพิ่มเข้าไปในส่วนของข้อมูล เมื่อมาถึงชั้น Transport หรือ host-to-host layer ซึ่งโปรโตคอล TCP เป็นผู้รับผิดชอบจะมีการสร้าง TCP segment โดยมีการเพิ่มส่วนของ TCP header เข้าไปและมีการผนึกส่วนของ FTP datagram รวมกัน จากนั้น TCP segment นี้จะถูกส่งต่อไปยัง layer ระดับล่างลงไปคือ Internetwork layer ในชั้นนี้โปรโตคอล IP จะทำงานโดยการเพิ่มส่วน IP header รวมกันกับ TCP segment เข้าไป เรียกว่าเป็น IP datagram ก็เป็นอันเสร็จสิ้นขั้นตอนการผนึกหรือ encapsulation ข้อมูลจากระดับบนสุดลงมา เพื่อให้ส่งผ่าน IP datagram นี้ไปยังเครือข่ายอินเตอร์เน็ตได้ และในชั้นสุดท้ายก่อนที่จะส่ง datagram ออกไปยังสายสัญญาณ ในชั้น Network Interface จะมีการแปลงข้อมูลและเพิ่มส่วน error correction และ flag เพื่อให้การส่งข้อมูลนั้นไม่ผิดพลาด จากนั้นก็แปลงข้อมูลเป็นสัญญาณไฟฟ้าส่งผ่านสายสัญญาณที่เชื่อมโยงอยู่ต่อไป ซึ่งจากตัวอย่างนี้มีการส่งผ่านข้อมูลไปในเครือข่ายแบบ Ethernet ดังนั้นในชั้นสุดท้ายข้อมูลก็จะต้องถูกแปลงเป็น Ethernet Frame เสียก่อน

2.1.5 ไอพี ดาต้าแกรม (IP Datagram)

จากขบวนการ Encapsulation นั้น เราทราบว่าข้อมูลในการติดต่อกันไม่ว่าจะเป็นเนื้อความในอีเมล์หรือไฟล์ที่ส่งไปมา จะถูกผนึกข้อมูลหรือ Encapsulate ไปเป็นรูปแบบของ IP datagram และสุดท้ายก็จะถูกแปลงเป็น Ethernet frame หรือเฟรมข้อมูลในรูปแบบอื่นๆตามลักษณะการเชื่อมต่อทางกายภาพ เช่น Ethernet หรือ Token-Ring เป็นต้นเพื่อให้สามารถส่งข้อมูลออกสู่เครือข่ายและข้ามเครือข่ายไปสู่อินเตอร์เน็ตได้ ตัวข้อมูลที่ถูกแปลงมาเป็น IP datagram นี้จะประกอบด้วย 2 ส่วน คือส่วน IP header ที่มีขนาด 32 ไบต์และส่วนเนื้อข้อมูลที่เรียกว่า payload ขนาดของ IP datagram มีขนาดไม่แน่นอน มีลักษณะตามรูปที่ 2.2



รูปที่ 2.2 รูปแบบของ IP datagram ประกอบด้วยส่วน header และ payload

ส่วนของ IP header มีการแบ่งย่อยเพื่อระบุพารามิเตอร์ในการใช้งานต่างๆดังนี้

- ส่วน Version มีขนาด 4 บิต ถูกกำหนดค่าเป็น 4 ในกรณีที่ใช้ IP address เป็น IPv4 และในอนาคตเมื่อ IP address มีการใช้งานเป็น IPv6 หรือ IP เวอร์ชั่น 6 ค่าของ Version ก็จะมีค่าเป็น 6
- ส่วน Length มีขนาด 4 บิต ซึ่งเป็นค่าความยาวของ IP header นี้
- ส่วน Type of Service เป็นฟิลด์ข้อมูลขนาด 8 บิต เพื่อบอกให้ทราบว่าจะดำเนินการกับข้อมูลนี้อย่างไร เช่น low delay, high throughput เป็นต้น แต่ในการใช้งานจริง อุปกรณ์ Router ที่ส่งผ่านข้อมูลจะไม่สนใจข้อมูลนี้
- ส่วน Total Length มีขนาด16 บิต เก็บข้อมูลแสดงค่าความยาวสุทธิของ IP datagram นี้ เป็นจำนวนไบต์ ดังนั้นขนาดของ IP datagram นี้เป็นจำนวนไบต์ ดังนั้นขนาดของ IP datagram จะมีความยาวได้ไม่เกิน 65,535 ไบต์ ซึ่งในส่วนของ IP header จะมีขนาดอย่างน้อย 20 ไบต์ ดังนั้นเนื้อที่ของ payload ของ IP datagram ใดๆจะมีขนาดไม่เกิน 65,515 ไบต์ และในการส่งผ่านข้อมูลกันในอินเตอร์เน็ตตัว IP datagram จะมีขนาดเล็กที่สุดที่ 576 ไบต์(IP header 20 ไบต์, payload 512 ไบต์ และสำหรับข้อมูล option หรือ protocol header อีก 44 ไบต์) ดังนั้นในการส่งผ่านข้อมูล IP datagram ขนาด 576 ไบต์จึงเป็นขนาดเล็กที่สุดซึ่งไม่สามารถแยกย่อยลงไปกว่านี้ได้อีก
- ส่วน Identification เป็นส่วนข้อมูลที่บอกให้ทราบว่า IP datagram นั้นมาจากที่ใด โดยเฉพาะกรณีที่ข้อมูลถูกแยกออกเป็นส่วนย่อยๆแล้ว
- ส่วน Flags และ Fragment Offset เป็นส่วนที่ใช้ระบุการแยกและรวมข้อมูล เพื่อให้ข้อมูลที่ถูกแยกออกเป็นข้อมูลย่อย (fragment) สามารถกลับมารวมกันใหม่ได้ตามลำดับได้ถูกต้อง
- ส่วน Time หรือ Time to Live เป็นข้อมูลแสดงจำนวนเวลาที่มากที่สุดของ IP datagram นี้ซึ่งสามารถจะส่งผ่านเครือข่ายไปยังปลายทางได้ โดยมีหน่วยเป็นวินาที และปกติจะมีค่าเป็น 32 โดยในระหว่างที่ข้อมูล IP datagram ถูกส่งผ่าน Router ตัว Router ก็จะลดค่า Time to Live ลง 1 ค่าเสมอ ทำให้สามารถนำค่า time นี้ไปใช้นับจำนวนเครือข่ายที่ IP datagram นี้ส่งผ่านไปยังปลายทางได้ ซึ่งเรียกว่า hop count

- ส่วน Protocol เป็นข้อมูลการระบุโปรโตคอลที่ทำงานใน layer ข้างบนซึ่งผนึกลงมาใน IP datagram ซึ่งตัวอย่างของโปรโตคอลในชั้นบนที่ถูกผนึกมาให้ IP นี้ได้แก่ โปรโตคอล ICMP , โปรโตคอล TCP และโปรโตคอล UDP เป็นต้น ส่วนค่าที่อยู่ในฟิลด์นี้จะเป็นตัวเลขตามตาราง

โปรโตคอล	ค่าที่กำหนดในฟิลด์	คำอธิบาย
ICMP	1	Internet Control Message Protocol
TCP	6	Transmission Control Protocol
BGP	8	Border Gateway Protocol
UDP	17	User Datagram Protocol
OSPF	89	Open Shortest Path First

ตารางที่ 2.1 ค่าที่ระบุโปรโตคอลที่ใช้บ่อยๆ

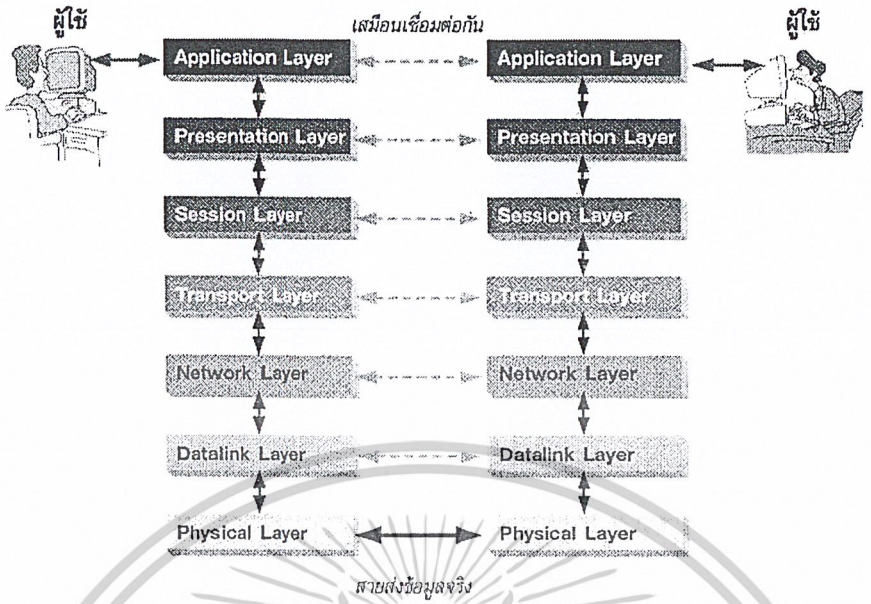
- ส่วน Header Checksum เป็นส่วนของข้อมูลที่ใช้ตรวจสอบความถูกต้องของข้อมูล เฉพาะในส่วนของ IP header โดยจะไม่เกี่ยวกับส่วนของ payload ซึ่งในการตรวจสอบความถูกต้องของข้อมูลนี้ โปรโตคอล IP จะทำหน้าที่ในการคำนวณและตรวจสอบ โดยกรณีที่เกิดความผิดพลาดของข้อมูล IP datagram นั้นจะถูกยกเลิกหรือไม่รับข้อมูลมาใช้งาน
- ส่วน Source IP Address เป็นส่วนเก็บข้อมูลของ IP Address ต้นทางที่ IP datagram นี้ถูกส่งมา
- ส่วน Destination IP Address เป็นส่วนเก็บข้อมูลของ IP Address ปลายทางที่เป็นผู้รับข้อมูล IP datagram นี้
- ส่วน Option เป็นฟิลด์เก็บข้อมูลที่มีขนาดไม่แน่นอน ใช้สำหรับกำหนดค่าพารามิเตอร์ส่วนประกอบปลีกย่อย ซึ่งส่วนใหญ่ไม่มีการนำไปใช้งาน
- และส่วนสุดท้าย คือ Padding ทำหน้าที่เป็นส่วนข้อมูลเติมเต็มเพื่อให้ IP header เต็มครบ 32 ไบต์ ซึ่งเป็นผลมาจาก Option ที่มีขนาดไม่แน่นอนนั่นเอง

2.2 โครงสร้างของโปรโตคอล TCP/IP

2.2.1 OSI Model : มาตรฐานอ้างอิงในการสื่อสารข้อมูล

OSI Model กำหนดให้การสื่อสารข้อมูลจากระบบคอมพิวเตอร์หนึ่งไปยังอีกระบบหนึ่ง แบ่งออกเป็น 7 ชั้นย่อยๆ ซึ่งคอมพิวเตอร์ทั้งสองระบบจะมีขั้นตอนทั้ง 7 ชั้นนี้เหมือนกันทั้งสองฝั่ง เราเรียกชื่อเต็มของแบบการสื่อสารข้อมูลนี้ว่า OSI 7-Layer Reference Model ดังแสดงในรูปที่ 2.3

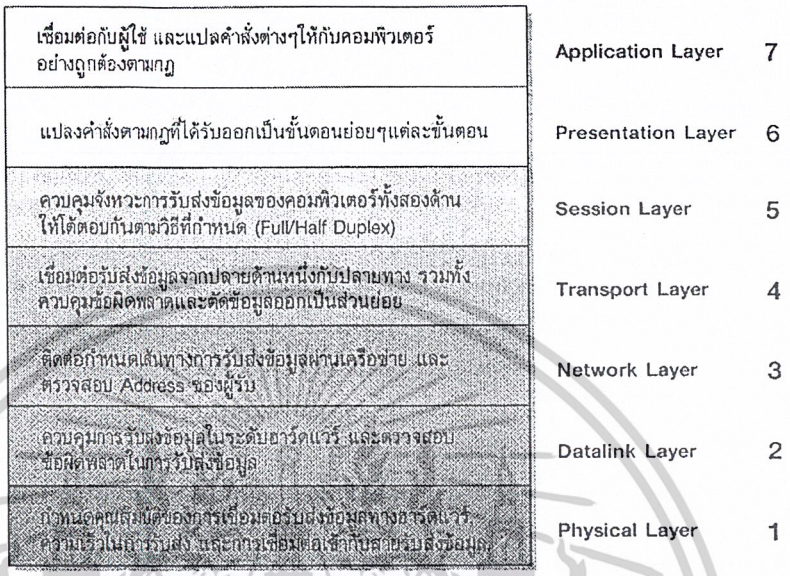
OSI Model ที่แบ่งการรับส่งข้อมูลระหว่างคอมพิวเตอร์ออกเป็น 7 ชั้นนั้น แต่ละชั้นจะมีชื่อเรียกและหน้าที่การทำงานดังนี้



รูปที่ 2.3 การรับส่งข้อมูลของ OSI 7-Layer Reference Model

- **Layer ที่ 7 Application Layer** เป็นชั้นบนสุดของขบวนการรับส่งข้อมูล ทำหน้าที่ติดต่อกับผู้ใช้ โดยจะรับคำสั่งต่างๆจากผู้ใช้ส่งให้คอมพิวเตอร์แปลความหมาย และทำงานตามคำสั่งที่ได้รับในระดับโปรแกรมประยุกต์
- **Layer ที่ 6 Presentation Layer** เป็นชั้นที่ทำหน้าที่ตกลงกับคอมพิวเตอร์อีกด้านหนึ่งในชั้นเดียวกันว่าการรับส่งข้อมูลในระดับโปรแกรมประยุกต์จะมีขั้นตอนและข้อบังคับอย่างไร ข้อมูลที่ทำการรับส่งกันใน Layer ที่ 6 นี้จะอยู่ในรูปแบบของข้อมูลชั้นสูง ซึ่งอยู่ในรูปแบบของคำสั่งที่มีกฎ (Syntax) บังคับอย่างแน่นอน
- **Layer ที่ 5 Session Layer** ทำหน้าที่ควบคุมจังหวะในการรับส่งข้อมูลของคอมพิวเตอร์ทั้งสองด้าน ที่รับส่งแลกเปลี่ยนข้อมูลให้มีความสอดคล้องกัน (Synchronization) และกำหนดวิธีที่ใช้รับส่งข้อมูล
- **Layer ที่ 4 Transport Layer** ทำหน้าที่เชื่อมต่อการรับส่งข้อมูลระดับสูงของ Layer ที่ 5 (ซึ่งมองในลักษณะประโยคของข้อมูลที่ตอบโต้กัน) มาเป็นข้อมูลในระดับฮาร์ดแวร์
- **Layer ที่ 3 Network Layer** ทำหน้าที่เชื่อมต่อคอมพิวเตอร์ของด้านรับและด้านส่งเข้าหากันผ่านระบบเครือข่ายพร้อมทั้ง ควบคุมในการหาเส้นทางในการเชื่อมต่อระหว่างต้นทางและปลายทาง และรูปแบบของข้อมูลในชั้นนี้เป็นลักษณะของกลุ่มข้อมูล (Packet)
- **Layer ที่ 2 Data Link Layer** เป็นชั้นที่ทำหน้าที่เชื่อมต่อการรับส่งข้อมูลในระดับฮาร์ดแวร์และควบคุมการส่งผ่านข้อมูลในลักษณะเฟรม (Frame)

- **Layer ที่ 1 Physical Layer** กำหนดการเชื่อมต่อถึงกันทางกายภาพระหว่างคอมพิวเตอร์กับคอมพิวเตอร์, คอมพิวเตอร์กับเทอร์มินัล, เทอร์มินัลกับเทอร์มินัล มาตรฐานในชั้นนี้กล่าวถึงคุณสมบัติทางไฟฟ้าและทางกายภาพ รวมถึงขั้นตอนวิธีการทำงานต่างๆในการเชื่อมต่อ



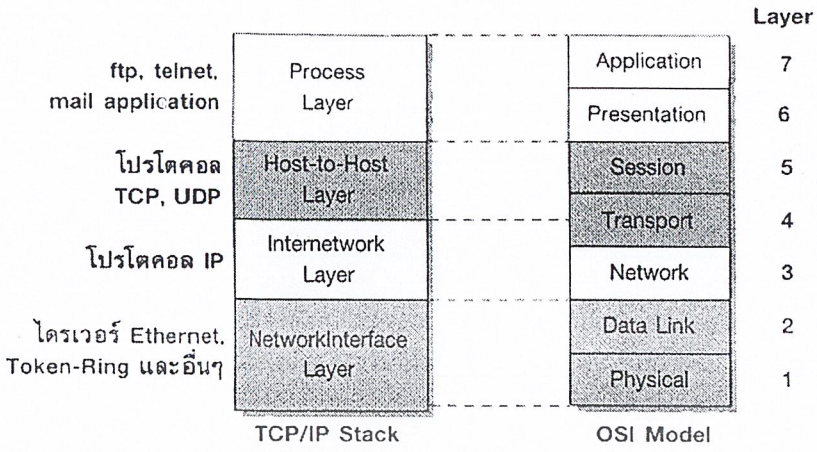
รูปที่ 2.4 หน้าที่แต่ละชั้นใน OSI 7-Layer Reference Model

2.2.2 โครงสร้างของโปรโตคอล TCP/IP

TCP/IP มีการจัดแบ่งกลไกการทำงานออกเป็นชั้นๆหรือ layer เหมือนกับมาตรฐานของ OSI Model ได้ซึ่งในแต่ละ layer ของโปรโตคอล TCP/IP จะประกอบด้วย

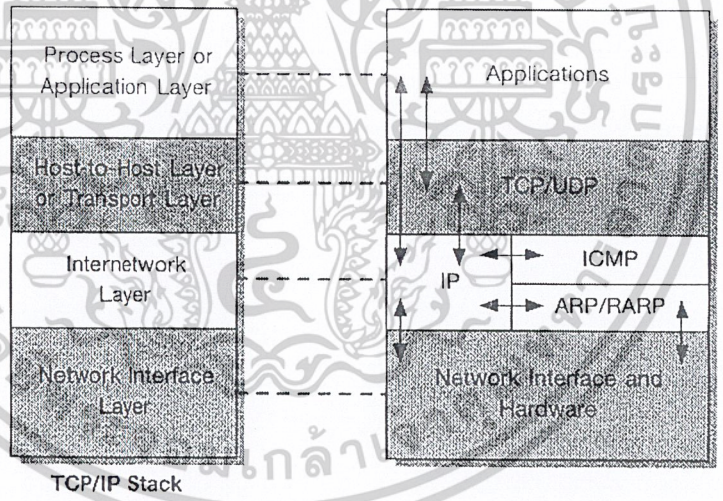
- Process layer หรือ Application Layer
- Host-to-Host layer หรือ Transport Layer
- Internetwork layer
- Network Interface layer

โดยเมื่อเทียบกับมาตรฐาน OSI model แล้วจะเป็นดังรูปที่ 2.5 ซึ่งเราจะเห็นว่าบาง Layer ของโปรโตคอล TCP/IP เทียบได้กับมาตรฐานของ OSI model ถึงสอง Layer และบาง layer ก็จะทำงานคาบเกี่ยวกับหลายๆ Layer ของ OSI model ตัวอย่างเช่น ในส่วน Network Interface layer ของโปรโตคอล TCP/IP จะเทียบได้กับการนำเอา Data Link layer และ Physical layer ของมาตรฐาน OSI model มารวมกัน



รูปที่ 2.5 แสดง TCP/IP stack เปรียบเทียบกับ มาตรฐาน OSI

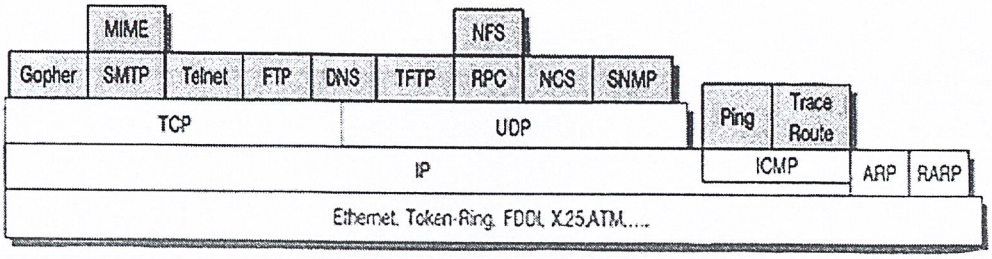
ในแต่ละกลไกของโปรโตคอล TCP/IP จะมีโปรโตคอลอื่นๆในชุดของ TCP/IP ร่วมกันทำงานอยู่ด้วย จึงทำให้เป็นที่มาของชื่อเรียก Protocol Stack เนื่องจากมีโปรโตคอลซ้อนทับกันอยู่เพื่อช่วยกันทำงานดังรูป



รูปที่ 2.6 แสดงความสัมพันธ์ระหว่างโปรโตคอลต่างๆของ TCP/IP

จากรูปจะเห็นว่า มีโปรโตคอลในแต่ละระดับซ้อนทับกันอยู่หลายตัวด้วยกัน การซ้อนทับกันเป็นชั้นๆหรือแต่ละ Layer นี้หากเป็น OSI model จะมีข้อบังคับให้แต่ละชั้นติดต่อได้กับเฉพาะชั้นที่ติดกับตนเองเท่านั้น แต่สำหรับ TCP/IP Stack แล้วจะเห็นว่าบางชั้นสามารถละเลยหรือข้ามไปติดต่อกับชั้นอื่นที่ไม่ติดกับตนเองได้

2.2.3 โปรเซส เลเยอร์ (Process layer) หรือ แอพพลิเคชั่น เลเยอร์ (Application Layer)



รูปที่ 2.7 แสดงแอพพลิเคชั่นต่างๆใน TCP/IP Stack

จากรูปที่ 2.5 แสดงลำดับชั้นการทำงานของโปรโตคอล TCP/IP เทียบกับมาตรฐาน OSI model นั้นในชั้นบนสุดเรียกว่า Process layer ทำงาน 2 หน้าที่เทียบได้กับ Application layer และ Presentation layer ในชั้นนี้จะรองรับการทำงานของแอพพลิเคชั่นต่างๆที่ทำงานเป็นโปรเซส อยู่ในเครื่องเซิร์ฟเวอร์ที่ให้บริการและเครื่องที่ขอใช้บริการ หรือไคลเอนต์ (client) ซึ่งจะติดต่อกันผ่านโปรโตคอลเฉพาะแอพพลิเคชั่นอีกทีหนึ่ง เช่นเมื่อผู้ใช้งานต้องการถ่ายโอนไฟล์หรือ download ข้อมูลจากเครื่องเซิร์ฟเวอร์ที่ให้บริการ โดยอาจจะเรียกโปรแกรม ftp client ทั่วไปติดต่อกับโปรเซส ftp ที่กำลังให้บริการอยู่ที่เครื่องเซิร์ฟเวอร์ จากนั้นโปรเซส ftp ก็จะเรียกใช้โปรโตคอล FTP (File Transfer Protocol) เพื่อทำการถ่านโอนไฟล์นี้ หรือถ้าผู้ใช้ต้องการเรียกใช้โปรแกรม web browser เพื่อเรียกดูเว็บเพจในเว็บไซท์ ก็จะมีโปรเซส HTTP (Hyper Text Transfer Protocol) ทำงานอยู่และจะติดต่อกับผู้ใช้งานผ่านโปรโตคอล HTTP เป็นต้น

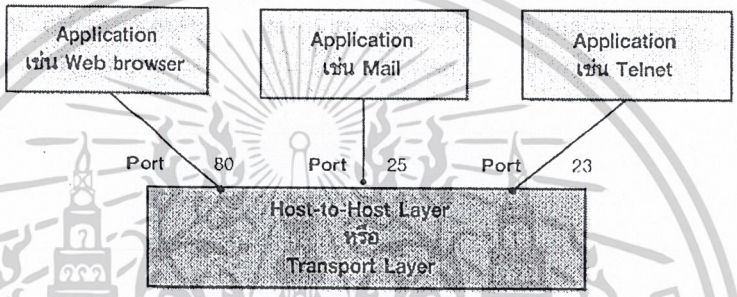
การทำงานของแอพพลิเคชั่นต่างๆจะอยู่ที่ Process layer นี้ และมีการติดต่อกันตามแต่ละโปรโตคอลเฉพาะแล้วแต่แอพพลิเคชั่นที่ใช้งาน จากการที่ Process layer ของ TCP/IP รองรับให้โปรโตคอลอื่นทำงานได้หลายโปรเซสและหลายโปรโตคอลได้พร้อมกันนั้น ทำให้ผู้ใช้สามารถเปิดโปรแกรมใช้งานได้หลายๆอย่างได้พร้อมกัน

โปรโตคอลหลักๆที่ทำงานใน Process layer ซึ่งผู้ใช้มักจะคุ้นเคยกันดีได้แก่ FTP (File Transfer Protocol), Telnet, HTTP (Hyper Text Transfer Protocol) และ SMTP (Simple Mail Protocol) นอกจากนี้ยังมีโปรโตคอลอื่นที่อยู่เบื้องหลัง ซึ่งทำงานโดยไม่สามารถมองเห็นได้จากโปรแกรมหรือไม่ได้มีการใช้งานโดยตรงเช่น

- โปรโตคอล DNS (Domain Name System) ที่ทำหน้าที่แปลงข้อมูลชื่อ domain name หรือชื่อเว็บไซท์ทั้งหลายให้เป็นหมายเลข IP Address
- โปรโตคอล SNMP (Simple Network Management Protocol) ใช้ในการควบคุมและตรวจสอบอุปกรณ์ที่อยู่ในเครือข่าย
- โปรโตคอล DHCP (Dynamic Host Configuration Protocol) ทำหน้าที่แจกจ่ายข้อมูลพารามิเตอร์ของเครือข่ายให้กับเครื่องลูกข่ายที่ต่อเชื่อมอยู่

2.2.4 โฮสท์ ทู โฮสท์ เลเยอร์ (Host-to-Host layer) หรือ ทรานสปอร์ต เลเยอร์ (Transport Layer)

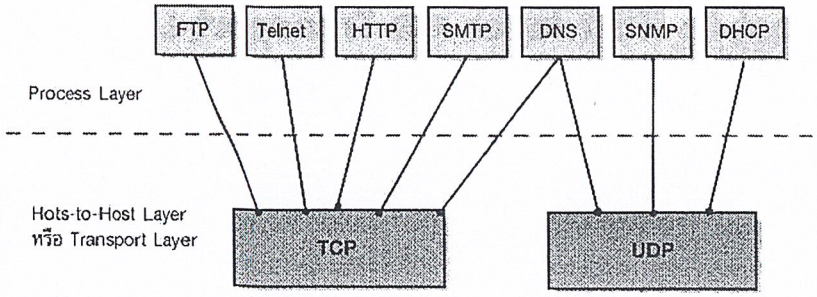
การทำงานที่ชั้นของ Host-to-Host layer นี้จะมีบทบาทในการจัดการต่อจาก Process layer บางครั้งเราเรียกชั้น Host-to-Host layer ว่าเป็น Transport Layer ซึ่งไม่ใช่ชั้น Transport Layer ในมาตรฐานของ OSI model การทำงานของ Host-to-Host layer นี้จะมีการสร้าง connection หรือการเชื่อมต่อกันระหว่างแอพพลิเคชั่นกับ Host-to-Host layer โดยจุดที่เชื่อมกันเพื่อรับส่งข้อมูลนี้เรียกว่า port หรือ socket และในแต่ละแอพพลิเคชั่นก็จะสร้างการเชื่อมต่อผ่าน port ได้พร้อมกันหลายแอพพลิเคชั่น ซึ่งการใช้งาน port ของแต่ละแอพพลิเคชั่นที่อยู่ในชั้น Process layer จะแตกต่างกันตามหมายเลขที่กำหนดไว้ แต่ละโปรโตคอลจะมีการใช้งาน port หมายเลขต่างๆไม่ซ้ำกันตามรูป



รูปที่ 2.8 แสดงหมายเลข port ของการใช้งานแต่ละอย่าง

เมื่อแอพพลิเคชั่นทำงานผ่านโปรโตคอลในชั้น Process layer จะมีการส่งผ่านข้อมูลไปยัง Host-to-Host layer ที่ชั้นนี้จะมีการเชื่อมต่อผ่าน port ที่กำหนด ทำให้การรับส่งข้อมูลในแต่ละโปรโตคอลทำได้ถูกต้อง ถึงแม้ว่าในเครื่องเซิร์ฟเวอร์ที่ให้บริการจะมีการทำงานอยู่หลายโปรเซสที่แตกต่างกันก็ตาม หรือมีผู้ใช้บริการเข้ามาใช้งานพร้อมกันจำนวนมากและหลายแอพพลิเคชั่นในเวลาเดียวกัน ในชั้น Host-to-Host layer หรือ Transport Layer ของ TCP/IP นี้ จะมีโปรโตคอลทำงานอยู่ 2 โปรโตคอลที่แตกต่างกัน คือ โปรโตคอล TCP และโปรโตคอล UDP (User Datagram Protocol) ในการส่งผ่านข้อมูลลงไปในชั้นถัดๆไป เราจะเห็นว่าโปรโตคอล TCP และ UDP จะถูกผนึกเข้าไปในโปรโตคอล IP อีกทีหนึ่งและส่งต่อออกไปยังเครือข่ายอินเตอร์เน็ตต่อไป

ตัวโปรโตคอล TCP และ UDP จะมีแอพพลิเคชั่นเฉพาะเพื่อเรียกใช้งานแยกกันคือแอพพลิเคชั่นที่ใช้โปรโตคอล FTP, Telnet, HTTP และ SMTP จะมีการส่งผ่านข้อมูลโดยเรียกใช้โปรโตคอล TCP ส่วนแอพพลิเคชั่นที่ใช้โปรโตคอล SNMP และ DHCP จะส่งผ่านข้อมูลโดยเรียกใช้โปรโตคอล UDP และสำหรับโปรโตคอล DNS นั้นจะสามารถเรียกใช้งานได้ทั้ง TCP และ UDP ดังรูป ซึ่งเหตุผลที่มีการเรียกใช้โปรโตคอล TCP และ UDP แตกต่างกัน ก็เนื่องจากวิธีการทำงานของทั้งสองโปรโตคอลต่างกันนั่นเอง



รูปที่ 2.9 แสดงโปรเซสต่างๆที่เรียกใช้ Transport layer

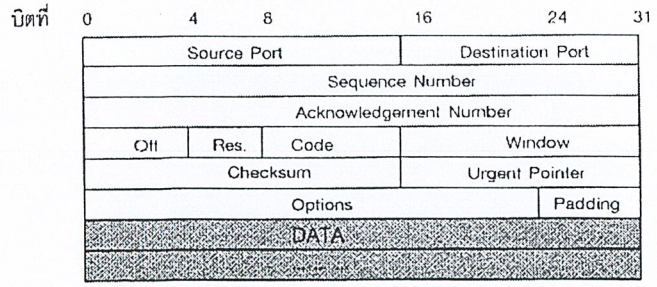
- **โปรโตคอล TCP**

โปรโตคอล TCP (Transmission Control Protocol) เป็นโปรโตคอลที่มีการรับส่งข้อมูลแบบ stream oriented protocol หมายความว่า การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่งไป แต่จะแบ่งข้อมูลเป็นส่วนย่อยๆก่อน แล้วจึงจะส่งไปยังปลายทางอย่างต่อเนื่องเป็นลำดับข้อมูล ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไป ก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้งหนึ่ง สำหรับปลายทางก็จะทำหน้าที่จัดเรียงส่วนของข้อมูล datagram ใหม่ให้ต่อเนื่องกันและประกอบกลับเป็นข้อมูลทั้งหมดได้ ซึ่งจะแยกข้อมูลที่ไม่ถูกต้องออก ดังนั้นแอพพลิเคชั่นหรือส่วนโปรเซสใดที่อาศัยการส่งผ่านข้อมูลด้วยโปรโตคอล TCP จะต้องใช้หน่วยความจำและขนาดของช่องสัญญาณ (bandwidth) มากกว่า UDP

การติดต่อระหว่างกันจะต้องเป็นแบบ connection-oriented คือต้องมีการสร้างการติดต่อกันเป็น session ทั้ง 2 ด้านเสียก่อน แล้วจึงจะรับส่งข้อมูลไปได้พร้อมกัน (full duplex) เหมือนกับการใช้โทรศัพท์ติดต่อกัน เมื่อผู้ติดต่อต้นทางเรียกให้ฝ่ายตรงข้ามรับสายแล้วจึงจะสนทนา เช่นเดียวกับการติดต่อกันด้วยกลไกโปรโตคอล TCP เมื่อแอพพลิเคชั่นต้องการส่งผ่านข้อมูลจะใช้ข้อมูลที่เหมาะสมในชั้น Process layer ติดต่อไปและมีการสร้างช่องส่งข้อมูลผ่าน port ที่กำหนดเพื่อส่งผ่านข้อมูลไปยังโปรโตคอล TCP

ในระหว่างการรับส่งข้อมูลนี้ โปรโตคอล TCP จะเพิ่มขบวนการตรวจทานข้อมูลเพื่อให้ข้อมูลมีความถูกต้องไม่ผิดพลาดไปจากเดิม โดยการส่งสัญญาณตรวจทานข้อมูล (acknowlwdgement) และส่งข้อมูลใหม่อีกครั้งถ้าปลายทางไม่ได้รับหรือเกิดความผิดพลาดขึ้น

ความน่าเชื่อถือของการส่งผ่านข้อมูลโดยโปรโตคอล TCP จะมีมากกว่า แต่ก็ต้องอาศัยทรัพยากรของระบบมากกว่าในการทำงานเช่นกัน



รูปที่ 2.10 รูปแบบของ TCP/IP packet

- **โปรโตคอล UDP**

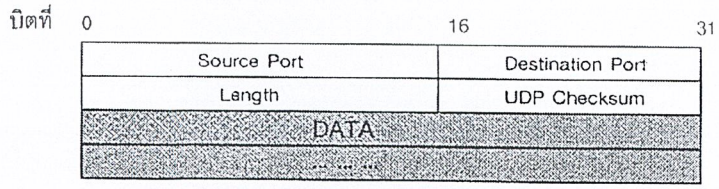
ใน Host-to-Host layer นอกจากจะมีโปรโตคอล TCP ทำงานแล้ว ก็ยังมีโปรโตคอล UDP (User Datagram Protocol) ที่มีคุณสมบัติแตกต่างกันอยู่ด้วย ในการรับส่งข้อมูลผ่านโปรโตคอล UDP จะเป็นแบบที่ทั้งสองด้านไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกัน (connectionless) ระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องที่ขอใช้บริการ โดยไม่ต้องแจ้งให้ฝ่ายรับข้อมูลเตรียมรับข้อมูลเหมือนโปรโตคอล TCP และไม่มีการตรวจสอบความถูกต้องครบถ้วนในการรับส่งข้อมูลนั้นๆด้วยเนื่องจากโปรโตคอล UDP ไม่มีสัญญาณสอบทานข้อมูล (acknowledgement) ในการส่งข้อมูลในแต่ละครั้ง และไม่มีการส่งข้อมูลใหม่อีกครั้งในกรณีที่เกิดความผิดพลาดของการส่งข้อมูล เมื่อเป็นเช่นนี้แอพพลิเคชั่นหรือโปรเซสใดที่ต้องอาศัยโปรโตคอล UDP ในการส่งผ่านข้อมูลอาจต้องสร้างขบวนการตรวจสอบข้อมูลขึ้นมาเอง

ตามรูปจะเห็นว่าโปรโตคอลชั้นบนขึ้นไป ที่ใช้ในการส่งผ่านข้อมูลโดยโปรโตคอล UDP เช่น โปรโตคอล SNMP หรือโปรโตคอล DHCP การส่งข้อมูลเหล่านั้นไม่ต้องรับทราบหรือตรวจสอบว่าข้อมูลไปถึงปลายทางถูกต้องหรือไม่ แต่กลไกการตรวจสอบข้อมูลที่มีการรับส่งจะไปทำในขั้นตอนของโปรโตคอลในชั้นที่สูงกว่าแทน

ตัวอย่างขั้นตอนกลไกการทำงานโดยใช้โปรโตคอล UDP มีดังต่อไปนี้

1. ในชั้นของ Process layer เมื่อโปรแกรมควบคุมอุปกรณ์เครือข่ายเช่น โปรแกรม Network Management ต้องการส่งข้อมูลไปยังอุปกรณ์ที่ต้องการแอพพลิเคชั่นนั้นจะติดต่อผ่านโปรโตคอล SNMP ในชั้น Process layer
2. โปรโตคอล SNMP จะติดต่อกับโปรโตคอล UDP ในชั้นถัดไป เพื่อขอติดต่อผ่าน port ที่กำหนด
3. โปรโตคอล SNMP เตรียมข้อมูลที่จะส่ง รวมทั้งที่อยู่ปลายทาง
4. โปรโตคอล SNMP ส่งผ่านข้อมูลให้โปรโตคอล UDP ที่อยู่ชั้น Host-to-Host layer
5. โปรโตคอล UDP ทำหน้าที่ผนึกข้อมูลหรือ datagram นั้น ไปกับโปรโตคอล IP ในชั้นถัดลงไปเพื่อส่งข้อมูลออกจากเครื่อง

ซึ่งจะเห็นว่ามีกลไกในการส่งข้อมูลด้วยโปรโตคอล TCP ซึ่งจะต้องมีการติดต่อกันก่อนและทั้งสองฝ่ายรับทราบการส่งข้อมูลของช่องการส่งข้อมูลนั้น



รูปที่ 2.11 รูปแบบของ UDP packet

2.2.5 อินเตอร์เน็ตเวิร์ค เลเยอร์ (Internetwork Layer)

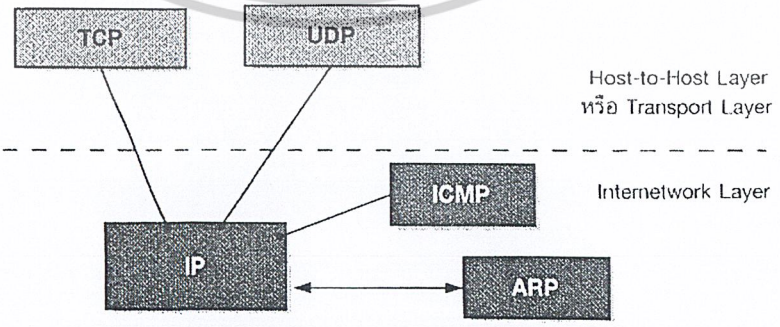
ในระดับล่างต่อมาในชั้น Internetwork Layer มีหน้าที่ส่งผ่านข้อมูลในระหว่างเครือข่าย โดยมีโปรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใดๆบนอินเตอร์เน็ต คือ โปรโตคอล IP (Internet Protocol) นอกจากนี้ในชั้น Internetwork Layer ยังมีโปรโตคอลทำงานอยู่ด้วยกันอีก 2 ชนิดคือ โปรโตคอล Internet Control Message Protocol (ICMP) และโปรโตคอล Address Resolution Protocol (ARP)

- โปรโตคอล IP

โปรโตคอล IP ทำหน้าที่ให้บริการส่งผ่านข้อมูลที่มาจาก Host-to-Host layer เพื่อส่งข้ามไปยังเครือข่ายใดๆได้อย่างถูกต้อง แม้ว่าเครือข่ายเชื่อมต่ออยู่ในอินเตอร์เน็ตเป็นล้านๆเครือข่ายก็ตามเนื่องจากโปรโตคอล IP มีตำแหน่งข้อมูล IP ปลายทางที่จะส่งข้อมูลไปให้ โดยทำงานร่วมกับ อุปกรณ์ Router เพื่อส่งข้อมูลข้ามเครือข่ายออกไปได้ ตัวโปรโตคอล IP จะทำงานแบบ packet switching คือมีการส่งข้อมูลผ่านสวิทซ์ (switch) ไปยังปลายทาง โดยข้อมูลจะเดินทางไปยังเครือข่ายต่างๆผ่านสวิทซ์นี้ไปยังเครือข่ายไปเรื่อยๆจนกว่าจะถึงปลายทาง ตัววงจรผ่านหรือ switch นี้อาจจะเป็น Gateway หรือ Router ในระบบเครือข่ายก็ได้ ซึ่งในข้อมูลของโปรโตคอล IP จะมีข้อมูล IP ของหมายเลข IP ปลายทางที่จะส่งข้อมูลไป และเมื่อส่งข้อมูลถึงเครือข่ายปลายทางแล้ว จะมีกลไกการแปลงหมายเลข IP ให้เป็นหมายเลข ฮาร์ดแวร์ประจำเครื่องที่ถูกต้องอีกทีหนึ่งด้วยโปรโตคอล ARP ตามรูปที่ 2.10 ที่จะแสดงการติดต่อกันระหว่างโปรโตคอลในชั้นของ Host-to-Host layer และ Internetwork Layer

- โปรโตคอล ICMP

ทำหน้าที่หลักของโปรโตคอล ICMP (Internet Control Message Protocol) คือการแจ้งหรือแสดงข้อความจากระบบ เพื่อบอกให้ผู้ใช้ทราบว่าเกิดอะไรขึ้นในการส่งผ่านข้อมูลนั้น ซึ่งปัญหาส่วนมากที่พบคือส่งไปไม่ได้ หรือปลายทางรับข้อมูลไม่ได้ เป็นต้น นอกจากนี้โปรโตคอล ICMP ยังถูกเรียกใช้งานจากเครื่องเซิร์ฟเวอร์และ Router อีกด้วย เพื่อแลกเปลี่ยนข้อมูลที่ใช้ควบคุม ส่วนรูปแบบการทำงานของโปรโตคอล ICMP นั้นจะทำคู่กับโปรโตคอล IP ในระดับเดียวกัน และข้อความต่างๆที่แจ้งให้ทราบจะถูกผนึกอยู่ภายในข้อมูลของ IP (IP datagram) อีกทีหนึ่ง



รูปที่ 2.12 การทำงานร่วมกันของโปรโตคอลต่างๆในชั้นของ Internetwork layer

ข้อความที่โปรโตคอล ICMP ส่งนั้นแบ่งออกได้ 2 แบบคือ ICMP error message หรือ ข้อความแจ้งข้อผิดพลาด และ ICMP query หรือข้อความเรียกขอข้อมูลเพิ่ม เช่น เมื่อมีการส่งผ่านข้อมูลไปยังปลายทางที่ไม่ถูกต้อง หรือขณะนั้นเครื่องปลายทางเกิดปัญหาจนไม่สามารถรับข้อมูลได้ ที่ Router จะส่งข้อความแจ้งเป็น ICMP message ที่ชื่อ destination unreachable ให้กับผู้ส่งข้อมูล นอกจากนี้ตัวข้อมูลที่แจ้งข้อความก็จะมีส่วนของข้อมูล IP datagram ที่เกิดปัญหาด้วย ดังนั้นเมื่อผู้ส่งข้อมูลได้รับข้อความแจ้งแล้วก็จะทราบได้ว่าจุดที่เกิดปัญหานั้นอยู่ที่ใด

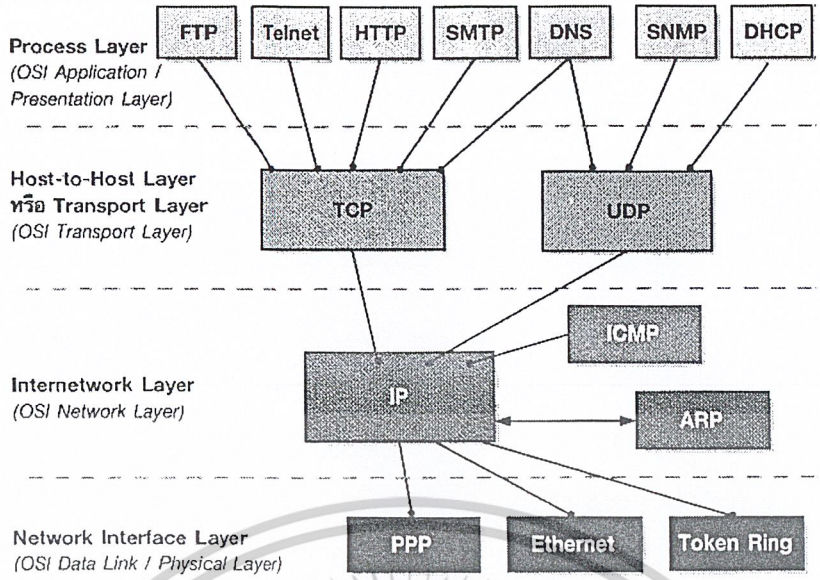
ดังนั้นโปรโตคอล ICMP จึงกลายมาเป็นเครื่องมืออย่างหนึ่งในการช่วยทดสอบเครือข่าย เช่น คำสั่ง ping ที่เรามักใช้ทดสอบว่าเครื่องเซิร์ฟเวอร์ที่ให้บริการหรืออุปกรณ์ที่ต่ออยู่ในเครือข่ายอินเตอร์เน็ตนั้นยังทำงานปกติหรือไม่ แล้วคำสั่ง ping มีการเรียกใช้งานโปรโตคอล ICMP แจ้งเป็นข้อความให้ทราบอีกต่อหนึ่ง

- **โปรโตคอล ARP**

โปรโตคอล ARP (Address Resolution Protocol) ถูกเรียกใช้งานโดยโปรโตคอล IP เพื่อช่วยแปลงหมายเลข IP ไปเป็นหมายเลขฮาร์ดแวร์ปลายทาง ตัวอย่างเช่น เว็บเซิร์ฟเวอร์เครื่องหนึ่งเชื่อมต่ออยู่ในเครือข่ายอินเตอร์เน็ต และในการเชื่อมต่อนี้ต้องอาศัย Network Interface Card (NIC) หรือ LAN card ติดตั้งอยู่ ที่ LAN card นี้เองจะมีหมายเลขเฉพาะประจำฮาร์ดแวร์ที่ไม่ซ้ำกับใคร เพื่อใช้อ้างอิงการส่งข้อมูลในเครือข่าย แต่เมื่อมาใช้งานในโปรโตคอล TCP/IP ก็จะต้องมีการกำหนดหมายเลข IP Address ประจำตัวเพื่อใช้อ้างอิงกัน และโปรโตคอล ARP จะทำหน้าที่แปลงค่าหมายเลข IP ให้เป็นหมายเลขฮาร์ดแวร์จริงให้ในระดับการทำงานที่ Internetwork layer นี้ ซึ่งกลไกการแปลงนี้เรียกว่า address resolution

2.2.6 เน็ตเวิร์คอินเตอร์เฟส เลเยอร์ (Network Interface Layer)

เนื่องจากในด้านกายภาพของเครือข่ายนั้น มีหลายวิธีการและหลายรูปแบบในการเชื่อมต่อระบบให้เป็นเครือข่าย แต่อย่างไรก็ตามในเครือข่ายอินเตอร์เน็ตนี้ ข้อมูลหรือ IP datagram จะถูกถ่ายทอดและส่งผ่านไปยังปลายทางโดยไม่คำนึงถึงรูปแบบการเชื่อมต่อทางกายภาพ การทำงานระดับล่างสุดต่อจาก Internet layer จะเป็นการแปลงข้อมูล IP datagram ให้อยู่ในรูปแบบที่เหมาะสม และแปลงเป็นสัญญาณไฟฟ้าส่งออกเครือข่ายต่อไป ซึ่งในชั้น Network Interface Layer นี้เมื่อเทียบกับมาตรฐาน OSI model แล้วจะเป็นการรวม 2 layer เข้าด้วยกันคือ Data link layer และ Physical layer กล่าวโดยสรุปคือ การทำงานในชั้นต่างๆตามโครงสร้างของโปรโตคอล TCP/IP จะมีลักษณะดังรูปที่ 2.13



รูปที่ 2.13 โครงสร้างของโปรโตคอล TCP/IP ในแต่ละชั้น

โปรโตคอลที่ใช้งาน	Port หรือ socket เชื่อมต่อ (เลขฐาน 10)	โปรโตคอลในระดับ Host-to-Host	รายละเอียด
BootP	67	UDP	BOOTstrap Protocol ด้านเซิร์ฟเวอร์
BootP	68	UDP	BOOTstrap Protocol ด้านไคลเอนต์
DHCP	67	UDP	Dynamic Host Configuration Protocol ด้านเซิร์ฟเวอร์
DHCP	68	UDP	Dynamic Host Configuration Protocol ด้านไคลเอนต์
DNS	53	UDP/TCP	Domain Name System
FTP	21	TCP	File Transfer Protocol ด้านเซิร์ฟเวอร์ที่ควบคุม
FTP	20	TCP	File Transfer Protocol ด้านเซิร์ฟเวอร์ที่ส่งข้อมูล
HTTP	80	TCP/UDP	Hyper Text Transfer Protocol ด้านเซิร์ฟเวอร์
NetBT	138	UDP	NetBIOS datagram service
NetBT	139	TCP	NetBIOS session service
SMTP	25	TCP	Simple Mail Transfer Protocol ด้านเซิร์ฟเวอร์
SNMP	161	UDP	Simple Network Management Protocol ด้าน agent
SNMP	162	UDP	SNMP trap manager

ตารางที่ 2.2 สรุปหมายเลขบางส่วนของ port ที่ใช้งานโดย TCP และ UDP (1)

โปรโตคอลที่ใช้งาน	Port หรือ socket เชื่อมต่อ (เลขฐาน 10)	โปรโตคอลในระดับ Host-to-Host	รายละเอียด
Telnet	23	TCP	Teletype Network Protocol
TFTP	69	UDP	Trivial File Transfer Protocol
WINS	137	UDP	Windows Internet Name Service

ตารางที่ 2.2 สรุปหมายเลขบางส่วนของ port ที่ใช้งานโดย TCP และ UDP (2)

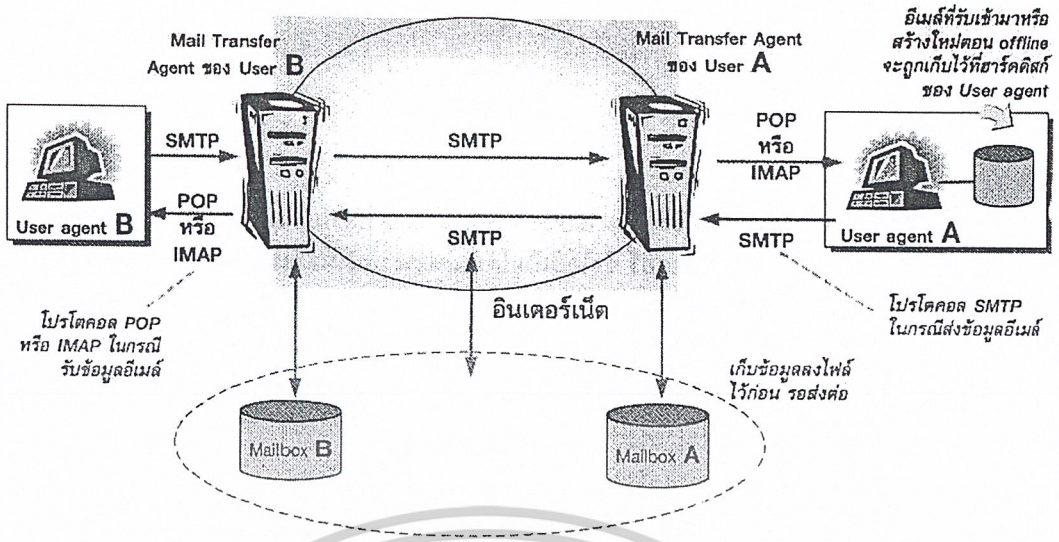
กล่าวโดยสรุปก็คือ โปรโตคอล TCP/IP ทำงานโดยแบ่งเป็นชั้นเทียบกับ OSI model ได้ กลไกในการทำงานของโปรโตคอล TCP/IP มี 4 ชั้น ซึ่งในชั้นแรก คือ Process layer ทำหน้าที่ติดต่อกับแอพพลิเคชั่นและโปรโตคอลและโปรโตคอลที่แอพพลิเคชั่นนั้นใช้งาน และส่งต่อมาให้ชั้น Host-to-Host layer เพื่อติดต่อกันระหว่างเครื่องเซิร์ฟเวอร์ ให้บริการกับเครื่องผู้ขอใช้บริการ ในชั้นนี้จะมีการสร้าง session หรือการเชื่อมต่อระหว่างระบบขึ้นตามแต่ละโปรโตคอลต้องการ ต่อมาเป็นการผนึกข้อมูลไปเป็นแบบ IP datagram ที่ชั้น Internetwork layer โดยอาศัยโปรโตคอล IP เพื่อให้สามารถติดต่อส่งข้อมูลข้ามเครือข่ายไปยังเครือข่ายและเครื่องที่ถูกต้องได้ และสุดท้ายการส่งข้อมูลสู่โลกภายนอก ต้องอาศัยกลไกในชั้น Network Interface layer เพื่อแปลงข้อมูลใหม่ เพิ่มข้อมูลที่จำเป็นในการอ้างอิงตำแหน่งและแปลงข้อมูลเป็นสัญญาณไฟฟ้าส่งออกไปยังเครือข่าย และอาจจะออกไปยัง Gateway หรือ Router เพื่อข้ามเครือข่ายออกไปยังเส้นทางที่กำหนดไว้ในอินเตอร์เน็ตต่อไป

เราจะเห็นว่าในแต่ละชั้นของโครงสร้าง TCP/IP stack มีการใช้งานโปรโตคอลต่างๆอยู่หนึ่งโปรโตคอลหรือมากกว่า ในแต่ละโปรโตคอลเหล่านี้ก็จะรับผิดชอบทำหน้าที่ของตน เพื่อส่งผ่านข้อมูลลงไปในระดับล่าง และออกสู่เครือข่ายอินเตอร์เน็ตในที่สุด

2.3 อีเมล์ และโปรโตคอลของอีเมล์

การทำงานของอีเมล์ไคลเอนต์และอีเมล์เซิร์ฟเวอร์มีส่วนประกอบดังนี้

- **User Agent** เป็นโปรแกรมคอมพิวเตอร์ทางด้านผู้ใช้งาน แบ่งเป็น 2 ส่วนคือ ส่วนของผู้ส่งและส่วนของผู้รับ โดย User agent นี้จะติดต่อเข้าสู่เซิร์ฟเวอร์ของตนโดยผ่านระบบ LAN หรือ Dial-up ซึ่งในส่วนของ User agent นี้จะเป็นส่วนที่ผู้ใช้ติดตั้งโปรแกรมไคลเอนต์ของอีเมล์เพื่อเรียกใช้บริการอีเมล์ เช่น Outlook Express หรือ Eudora เป็นต้น
- **MTA (Mail Transfer Agent)** เป็นโปรแกรมคอมพิวเตอร์ที่จะส่งอีเมล์จากต้นทางไปยังผู้รับปลายทาง ซึ่งจะต้องส่งผ่านเครื่องจำนวนมากที่เชื่อมต่อกันในเครือข่าย โดยโปรแกรมเหล่านี้จะช่วยกันส่งต่ออีเมล์เป็นทอดๆจนถึงเครื่องที่มี account หรือเมล์บ็อกซ์ของผู้รับ



รูปที่ 2.14 องค์ประกอบและโปรโตคอลต่างๆที่ใช้งานในระบบการทำงานของอีเมล์

2.3.1โปรโตคอลและการใช้งาน

ในการทำโครงงานนี้นั้นเราจะใช้โปรโตคอลของ SMTP (Simple Mail Protocol) เป็นหนึ่งในโปรโตคอลที่เราจะทำการจัดการดังนั้นในส่วนนี้จะขอยกตัวอย่างเพียงแค่โปรโตคอล SMTP เท่านั้นซึ่งจะมีลักษณะดังนี้

- SMTP (Simple Mail Transfer Protocol)

เป็นโปรโตคอลที่ใช้ส่งอีเมล์จาก User agent ของผู้ส่งไปยัง MTA ของผู้ส่งและส่งต่อไปยัง MTA เครื่องอื่นๆที่เป็นจุดผ่านในการเชื่อมต่อไปยังเครื่องของผู้รับ โปรโตคอล SMTP จะทำงานร่วมกับโปรโตคอล TCP โดยใช้พอร์ต 25 ซึ่งคำสั่งต่างๆจะเป็น ASCII ลงท้ายด้วย Carriage Return และ Line Feed ส่วนข้อความที่ตอบกลับมาจะนำหน้าด้วยเลข 3 หลัก เป็นสัญลักษณ์แสดงสถานการณ์ทำงานของคำสั่งที่ได้รับ

เมื่อเริ่มต้นการติดต่อ SMTP จะกำหนดให้ User agent ของผู้ส่งต้องส่งคำสั่ง HELLO พร้อมรายละเอียดด้านผู้ส่งออกไป จากนั้นจะส่งคำสั่ง MAIL เพื่อแจ้งให้เซิร์ฟเวอร์เตรียมรับอีเมล์ ในส่วนของเซิร์ฟเวอร์เมื่อพร้อมที่จะรับก็จะตอบรับกลับมาด้วยคำสั่ง OK จากนั้นด้านส่งก็จะเริ่มส่งโดยใช้คำสั่ง RCPT เพื่อกำหนดอีเมล์แต่ละฉบับที่ส่งไป ซึ่งการส่งข้อมูลของอีเมล์จะถูกระบุด้วยคำสั่ง DATA

การส่งอีเมล์ของโปรโตคอล SMTP ได้จัดเตรียมคำสั่งอื่นๆไว้เพื่ออำนวยความสะดวกและคล่องตัวในการทำงาน ซึ่งประกอบด้วยคำสั่ง VRFY เพื่อให้ด้านที่ส่งตรวจสอบรายชื่อจากลิสต์รายชื่อและคำสั่ง TURN ใช้สลับไคลเอนต์ของผู้ส่งทำหน้าที่รับข้อมูลจากเซิร์ฟเวอร์แทน

เมื่อได้รับคำสั่งต่างๆของผู้ส่งแล้ว เซิร์ฟเวอร์จะมีหน้าที่ตรวจสอบความถูกต้องของคำสั่ง จากนั้นจึงทำงานตามคำสั่งและส่งผลตอบกลับมา ส่วนในลักษณะข้อมูลที่ตอบกลับมา (reply message) นั้นจะเป็นข้อมูลในรูปของ text ที่เป็น ASCII โดยจะประกอบด้วยตัวเลขนำหน้าข้อความ 3 หลักทำหน้าที่แสดงสถานการณ์ทำงานของเซิร์ฟเวอร์ และเปลี่ยนสถานะการทำงานของโปรโตคอล

SMTP ด้วยถัดจากตัวเลขจะคั่นด้วยช่องว่างแล้วตามด้วยข้อความ ซึ่งปิดท้ายด้วยเครื่องหมาย Carriage Return และ Line Feed

ในการส่งอีเมล์ของโปรโตคอล SMTP นั้นจะใช้วิธีอ้างถึงเซิร์ฟเวอร์อื่นๆตามแบบ DNS หรือ Domain Name System เช่นเดียวกับระบบอื่นๆในอินเตอร์เน็ต และยังสามารถส่งอีเมล์ไปยังผู้รับคนเดียวหรือหลายๆคนพร้อมกันได้ด้วย

คำสั่ง	รายละเอียด
HELLO	ใช้เมื่อไคลเอนต์ของอีเมล์ต้องการเริ่มติดต่อกับเซิร์ฟเวอร์
MAIL	เริ่มเข้าสู่สถานะการส่งอีเมล์
RCPT	เป็นคำสั่งเพื่อระบุอีเมล์ที่จะส่งทีละฉบับ โดยเป็นคำสั่งที่ใช้ต่อจาก MAIL
DATA	จะเป็นคำสั่งที่ใช้ต่อจาก RCPT เพื่อส่งข้อมูลของอีเมล์
SEND	ทำหน้าที่เหมือนคำสั่ง DATA แต่ไม่ค่อยมีผู้ใช้งาน
SOML	ทำหน้าที่เหมือนคำสั่ง DATA แต่ไม่ค่อยมีผู้ใช้งาน
SAML	ทำหน้าที่เหมือนคำสั่ง DATA แต่ไม่ค่อยมีผู้ใช้งาน
VRFY	เป็นคำสั่งที่ใช้เพื่อตรวจสอบความถูกต้องของชื่อและเมล์บ็อกซ์
EXPN	เป็นคำสั่งเพื่อตรวจสอบรายละเอียดของลิสต์รายชื่อ
HELP	ใช้ตรวจสอบคำสั่งที่สามารถใช้งานได้กับเซิร์ฟเวอร์
NOOP	เป็นคำสั่ง No Operation เมื่อเซิร์ฟเวอร์ได้รับคำสั่งนี้จะตอบ OK กลับมา
QUIT	สิ้นสุดการติดต่อ
RSET	ยกเลิกการส่งข้อมูลในขณะนี้
TURN	เป็นคำสั่งที่สลับหน้าที่ของผู้ส่งข้อมูลมาทำหน้าที่รับข้อมูลแทน

ตารางที่ 2.3 ตารางแสดงรายละเอียดในคำสั่งต่างๆของ SMTP ที่ใช้งานอยู่

2.4 การรับส่งไฟล์ และระบบไฟล์ FTP (File Transfer Protocol)

FTP เป็นเครื่องมือในการโอนไฟล์ซึ่งเป็นที่รู้จักและได้รับความนิยมที่สุด โดยกำเนิดมาจากการเป็นคำสั่งพื้นฐานของระบบปฏิบัติการ Unix และแพร่หลายอยู่ในระบบปฏิบัติการต่างๆไม่ว่าจะเป็น DOS หรือ Windows 98/ME/2000 ก็ตาม ซึ่งคุณสมบัติของ FTP ก็คือสามารถโหลดไฟล์มาจากเซิร์ฟเวอร์ (download) หรือส่งไฟล์ไปเก็บที่เซิร์ฟเวอร์ (upload) ได้ แต่ในการใช้งานบนอินเตอร์เน็ต ผู้ใช้มักจะใช้เพื่อโหลดไฟล์จากเซิร์ฟเวอร์เสียเป็นส่วนใหญ่

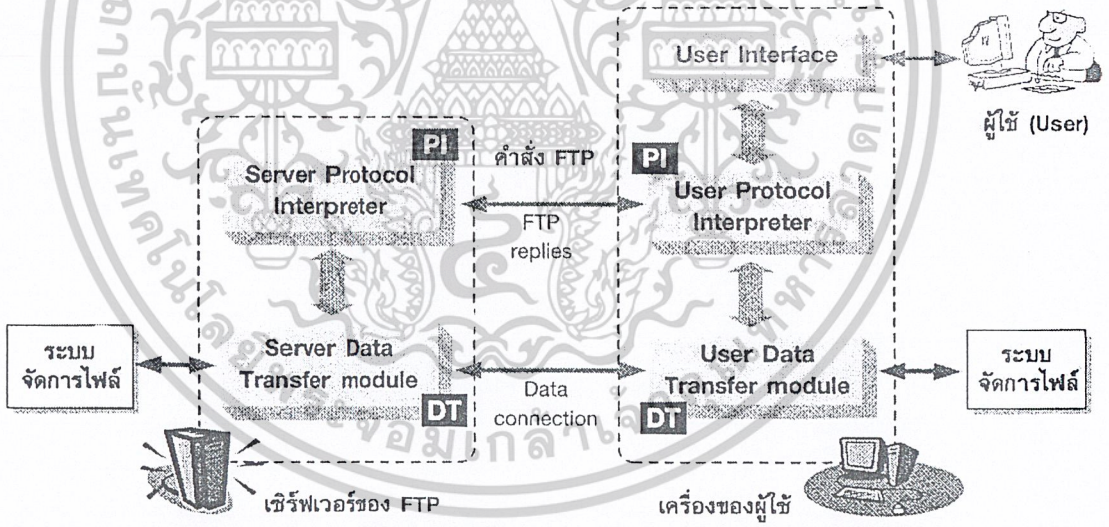
2.4.1 วิธีการทำงานของ FTP

FTP จะทำงานในแบบไคลเอนต์เซิร์ฟเวอร์ โดยพัฒนาขึ้นตามโปรโตคอลพื้นฐาน TCP ซึ่งจะต้องมีการติดต่อเพื่อจองช่องสื่อสาร (Connection Establishment) ก่อนทำการสื่อสารจริง ซึ่งเรียกว่าเป็นการติดต่อแบบที่ต้องขอเชื่อมต่อก่อน (Connection-oriented) ในการใช้งาน FTP เพื่อเริ่มทำการติดต่อสื่อสารนั้น จะต้องระบุหมายเลข IP ปลายทาง และต้องผ่านการแจ้งรหัส Login และ Password ของเซิร์ฟเวอร์ที่จะติดต่อก่อนจึงจะเข้าใช้งานได้

ข้อมูลของ FTP ที่สื่อสารระหว่างกันมี 2 ประเภทคือ

- **ข้อมูล (data)** หมายถึงข้อมูลต่างที่ต้องการรับส่ง รวมถึงไฟล์ที่รับมาจากเซิร์ฟเวอร์ หรือส่งมาจากไคลเอนต์แล้วไปเก็บไว้ที่เซิร์ฟเวอร์ก็ได้
- **ข้อมูลที่เป็นคำสั่ง (command)** FTP จะมีคำสั่งที่ใช้สั่งงานต่างๆ เช่น dir เป็นคำสั่งที่ใช้แสดงชื่อไฟล์หรือไดเรกทอรีในเครื่องเซิร์ฟเวอร์ โดยผู้ใช้จะสั่งงานที่ไคลเอนต์ผ่านโปรแกรม FTP แล้วโปรแกรมจะส่งคำสั่งไปยังเซิร์ฟเวอร์เพื่อทำงาน และแจ้งผลการทำงานกลับมายังไคลเอนต์ ซึ่งผลการทำงานนี้จะนำหน้าด้วยตัวเลข 3 หลัก เป็นรหัสที่ใช้แสดงสถานการณ์ทำงานภายในของ FTP และต่อด้วยข้อความที่เป็น text ต่อท้ายไป ซึ่งก็คือผลของการทำงานหรือคำอธิบายต่างๆ โดยที่ FTP มีกระบวนการภายในที่จะตรวจสอบได้ว่าข้อมูลที่รับส่งนี้เป็นประเภทคำสั่ง ไม่ใช่ตัวข้อมูลที่ต้องการจะโอนย้าย

การที่ FTP สามารถแยกแยะข้อมูลจริงออกจากข้อมูลที่เป็นคำสั่งได้นั้น ถือว่าเป็นหน้าที่การทำงานในโมดูลใน FTP ที่เรียกว่า ตัวแปรโปรโตคอล (Protocol Interpreter module หรือ PI) ซึ่งจะทำหน้าที่รองรับการทำงานคำสั่งต่างๆของ FTP และในส่วนของข้อมูลที่รับส่งนั้นจะเป็นหน้าที่ของโมดูลโอนข้อมูล (Data Transfer module หรือ DT) ซึ่งโมดูลทั้งสองนี้จะต้องทำงานอยู่ทั้งในเครื่องที่เป็นทั้งเซิร์ฟเวอร์และเครื่องที่เป็นไคลเอนต์



รูปที่ 2.15 องค์ประกอบและกลไกการทำงานของโปรโตคอล FTP

2.4.2 ประเภทของข้อมูล (Data Type)

คุณสมบัติที่สำคัญของ FTP คือความสามารถในการแปลงประเภทของข้อมูลให้ถูกต้องตามความเหมาะสม มีประโยชน์กรณีเครื่องเซิร์ฟเวอร์และไคลเอนต์เป็นคอมพิวเตอร์ที่ต่างระบบกัน ซึ่ง FTP ก็จะ

แปลงข้อมูลให้ถูกต้องตามความเหมาะสมได้ โดยที่ FTP จะมีคำสั่ง TYPE เพื่อกำหนดประเภทของข้อมูล ซึ่งประกอบด้วย

- **NVT-ASCII (Network Virtual Terminal ASCII)** เป็นข้อมูลทั่วๆไปที่ใช้รหัส ASCII
- **EBCDIC** เป็นข้อมูลที่ใช้ในเครื่องคอมพิวเตอร์เมนเฟรมของไอบีเอ็ม
- **IMAGE** เป็นข้อมูลที่เป็นไบนารี อาจจะเป็นโปรแกรม หรือรูปภาพโดยข้อมูลเป็น 8 บิต
- **LOCAL** เป็นข้อมูลที่เป็นไบนารีเช่นเดียวกับ IMAGE แต่จำนวนบิตจะต่างกันตามเครื่องคอมพิวเตอร์

2.4.3 โครงสร้างไฟล์ (File Structure)

FTP สามารถกำหนดโครงสร้างของไฟล์ได้หลายแบบ เพื่อรองรับการใช้งานร่วมกับเครื่องคอมพิวเตอร์ต่างชนิดกัน โดยแยกเป็นประเภทได้ดังนี้

- **File** เป็นโครงสร้างไฟล์ที่รับส่งในลักษณะเป็นไบต์ต่อเรียงกัน
- **Record** เป็นโครงสร้างไฟล์ที่ข้อมูลจะเป็นชุดของเรคอร์ด โดยเป็นไฟล์ที่มีความยาวคงที่
- **Page** เป็นการโอนข้อมูลที่เป็นชุดของบล็อกข้อมูล ซึ่งจะใช้กับไฟล์ที่ใช้งานแบบสุ่ม

2.4.5 วิธีการรับส่ง (Transmission Mode)

FTP กำหนดวิธีรับส่งข้อมูลดังนี้

- **Stream Mode** เป็นวิธีการที่จะรับส่งข้อมูลเรียงลำดับไบต์ส่งต่อกันไปเรื่อยๆ จึงสามารถใช้กับไฟล์ทุกประเภท ส่วนไฟล์ที่มีโครงสร้าง ต้องมีรหัสตัวอักษรพิเศษที่กำหนดการสิ้นสุดเรคอร์ด (End of Record หรือ EOR) และจบไฟล์ (End of File หรือ EOF) ด้วย ซึ่งการรับส่งข้อมูลจะสิ้นสุดโดยตรวจสอบจากค่าของ EOF นี้
- **Block Mode** เป็นการรับส่งข้อมูลที่เป็นบล็อก ในแต่ละบล็อกจะมีส่วนหัวที่ระบุขนาดและรายละเอียดต่างๆของบล็อก ขนาดบิตที่ใช้ตรวจสอบว่าข้อมูลที่รับส่งในบล็อกนั้นถูกต้องหรือไม่รวมทั้งข้อมูลที่เรียกว่า Restart Marker ซึ่งเป็นข้อมูลที่ใช้อ้างอิงเพื่อโอนข้อมูลต่อจากข้อมูลของเดิมที่โอนไม่สำเร็จ
- **Compressed Mode** เป็นวิธีเพิ่มประสิทธิภาพการรับส่งข้อมูล โดยใช้เทคนิคการบีบอัดข้อมูลให้เล็กลง โดยมีวิธีการง่ายๆคือ ข้อมูลซ้ำๆที่เรียงต่อกันจะถูกลดลงมาเหลือ 3 ไบต์ ซึ่งใน 2 ไบต์แรกจะบอกว่าข้อมูลชุดดังกล่าวมีค่าซ้ำจำนวนกี่ตัว

2.4.6 การแก้ไขกรณีข้อมูลผิดพลาด (Error Recovery)

ในการรับส่งไฟล์แบบ Stream Mode การแก้ไขกรณีข้อมูลสูญหายระหว่างการรับส่ง จะใช้ความสามารถของโปรโตคอล TCP แต่จะไม่มีการส่งข้อมูลซ้ำ แต่ถ้าหากเป็นแบบ Block Mode หรือ Compressed Mode ข้อมูลสามารถส่งใหม่เฉพาะส่วนที่ตรวจสอบพบข้อผิดพลาดได้

2.5 โปรโตคอลสำหรับ เวิลด์ไวด์เว็บ (World Wide Web)

2.5.1 เทคโนโลยีพื้นฐานของเวิลด์ไวด์เว็บ

เทคโนโลยีพื้นฐานของเวิลด์ไวด์เว็บนี้ จะครอบคลุมในเรื่องของแนวความคิด องค์ประกอบ รวมทั้งโปรโตคอลต่างๆที่ใช้ทั่วไปทางอินเตอร์เนต ซึ่งประกอบด้วย พื้นฐานที่เกี่ยวกับ URL ที่ใช้ในการอ้างอิงถึงเว็บไซต์ต่างๆ, รายละเอียดของโปรโตคอล HTTP ที่เป็นโปรโตคอลพื้นฐานของเวิลด์ไวด์เว็บที่ใช้ติดต่อกันระหว่างเว็บเซิร์ฟเวอร์ และเว็บบราวเซอร์ ซึ่งยังมีส่วนอื่นๆที่จะยังไม่ได้กล่าวถึงในหัวข้อเวิลด์ไวด์เว็บเพราะนอกเหนือขอบเขตของโครงงาน เช่น HTML, CGI

2.5.2 URL (Uniform Resource Locator)

URL เป็นหลักการกำหนดชื่ออ้างอิงของทรัพยากรต่างๆที่อยู่ภายในเครือข่ายอินเตอร์เนต หากเปรียบเทียบกับการจัดเก็บข้อมูลในคอมพิวเตอร์ทั่วไปก็คือ ชื่อของไฟล์, ชื่อของไดเรกทอรี หรือชื่อโฟลเดอร์ เป็นต้น แต่เนื่องจาก URL นั้นต้องรองรับการทำงานภายใต้เครือข่าย ดังนั้นรูปแบบของ URL จึงซับซ้อนมากกว่าชื่อไฟล์ หรือชื่อเครื่องคอมพิวเตอร์ทั่วๆไป ซึ่ง URL จะต้องสามารถบ่งบอกชื่อหรือแอดเดรสของเครื่องคอมพิวเตอร์ภายในเครือข่าย โปรโตคอลที่ใช้งาน รวมทั้งพารามิเตอร์และออปชั่นต่างๆด้วย

สำหรับเวิลด์ไวด์เว็บที่ใช้งานเว็บเพจต่างๆนั้น URL สามารถระบุชื่อของเว็บ เซิร์ฟเวอร์จนถึงที่เก็บไฟล์ HTML ของเว็บเพจนั้นๆและในการลิงค์ไปยังเพจอื่นด้วยคำสั่ง HTML โดยระบุเป็น URL ลงไปช่วยให้บราวเซอร์ทำงานร่วมกับ HTML ในแบบไฮเปอร์เท็กซ์ได้ ซึ่งในหน้าจอของโปรแกรมบราวเซอร์จะแสดง URL ที่ถูกอ้างถึงไว้ตลอดเวลาการทำงาน เพื่อให้ผู้ใช้ทราบว่ากำลังใช้งานที่เว็บไซต์ใดอยู่ นอกจากนั้นในบริการอื่นๆก็สามารถอ้างถึงในลักษณะของ URLได้ เช่น การดาวน์โหลดไฟล์ด้วย FTP การอ่านข่าวสารผ่าน News Group หรือการใช้งาน gopher และ finger ก็สามารถทำได้โดยระบุชื่อโปรโตคอลที่ต้องการใช้งานตามรูปแบบที่ URL กำหนดไว้ ซึ่งมีรูปแบบของ URL มาตรฐานประกอบด้วย

```
<Protocol>:<Protocol-specific name>
```

ส่วนต่างๆของรูปแบบ URL มีรายละเอียดดังนี้

<Protocol> จะทำหน้าที่กำหนดโปรโตคอล (หรือบริการที่ต้องการ) ที่จะใช้งาน ตัวอย่างเช่น HTTP ใช้อ้างถึงเว็บไซต์, FTP ใช้อ้างถึงเซิร์ฟเวอร์ที่ต้องการดาวน์โหลดไฟล์เป็นต้น

<Protocol-specific name> เป็นส่วนกำหนดรายละเอียดของแต่ละโปรแกรมเพื่อให้ทราบถึงรายละเอียดเพิ่มเติมในการใช้งาน เช่นชื่อของเซิร์ฟเวอร์หรือไดเรกทอรีที่เก็บไฟล์เป็นต้น ดังนั้นจึงสามารถปรับรูปแบบ URL มาตรฐานใหม่ได้ดังนี้

```
<Protocol>://<user>:<password>@<server>:<port>/<path>
```

<user> จะกำหนดชื่อของผู้ใช้งานพร้อมกับรหัสความปลอดภัย

<password> จะต้องระบุกรณีที่ใช้งานบางโปรโตคอลที่ต้องการ เช่น การใช้งาน FTP ที่ระบุผู้ใช้แตกต่างจาก anonymous ต้องกำหนด Password เป็นต้น

<server> จะใช้ระบุชื่อโดเมนเซิร์ฟเวอร์ที่ต้องการใช้งาน หรือสามารถระบุเป็นหมายเลข IP แทนได้

<port> ในกรณีเซิร์ฟเวอร์มีการใช้งานหมายเลขพอร์ตพิเศษ แตกต่างจากหมายเลขพอร์ตทั่วๆ ไปของแต่ละโปรโตคอลนั้น ผู้ใช้สามารถระบุหมายเลขพอร์ตใน URL

<path> ใช้เมื่อต้องการอ้างอิงถึงชื่อไฟล์หรือไดเรกทอรี เช่น การดาวน์โหลดด้วยโปรแกรม FTP หรือการดึงไฟล์เว็บเพจจากไดเรกทอรีซึ่งต่างจากที่กำหนดไว้

จากรูปแบบมาตรฐานของ URL นั้น จะเห็นได้ว่าชื่อของโปรโตคอลมีส่วนสำคัญในการใช้งานอย่างมาก เวิลด์ไวด์เว็บสามารถรองรับการทำงานของโปรโตคอลได้หลายชนิด แต่ละชนิดก็สามารถเรียกใช้งานได้โดยกำหนดเป็นชื่อย่อแทน ซึ่งรายชื่อของโปรโตคอลที่ใช้งานมีดังนี้

ชื่อโปรโตคอล	รายละเอียด
http	HyperText Transfer Protocol ที่ใช้งานในเวิลด์ไวด์เว็บ
https	เป็นโปรโตคอล HTTP ที่ใช้งานภายใต้ระบบรักษาความปลอดภัย SSL (Secure Socket Layer)
mailto	กำหนดแอดเดรสของ e-mail
ftp	ใช้กำหนดโปรโตคอล FTP ที่ใช้ดาวน์โหลดข้อมูล
finger	เป็นโปรโตคอลของโปรแกรม finger
gopher	เป็นโปรโตคอลของโปรแกรม gopher
wais	เป็นโปรโตคอลของโปรแกรม Wide Area Information Server
news	เป็นโปรโตคอลของโปรแกรม Usenet News
nntp	เป็นโปรโตคอลของโปรแกรม Usenet News กับโปรโตคอล NNTP (Network News Transfer Protocol)
snews	เป็นโปรโตคอลของโปรแกรม Usenet News กับระบบรักษาความปลอดภัย SSL ร่วมกับโปรโตคอล NNTP
file	ใช้กำหนดเพื่ออ่านไฟล์จากเซิร์ฟเวอร์
jdbc	ใช้กำหนดเพื่ออ่านข้อมูลจากออบเจ็คของ JDBC (Java Database Connector Database Object) ซึ่งเป็นไดรเวอร์ที่ติดต่อกับโปรแกรมฐานข้อมูลของภาษา Java คล้าย ODBC ที่ใช้กับ Windows ทั่วไป
irc	เป็นโปรโตคอลของโปรแกรม IRC (Internet Relay Chat)

ตารางที่ 2.4 รายชื่อของโปรโตคอลที่ใช้งานบน เวิลด์ไวด์เว็บ (1)

ชื่อโปรโตคอล	รายละเอียด
telnet	เป็นโปรโตคอลของโปรแกรม telnet
tn3270	ใช้กับโปรแกรม telnet ที่ทำงานผ่านการจำลองเทอร์มินัลแบบ 3270 ซึ่งใช้กับเครื่องเมนเฟรมในตระกูลไอบีเอ็ม
afs	ใช้งานกับไฟล์ในรูปแบบของ Andrew File System
nfs	ใช้งานกับไฟล์ในรูปแบบของ NFS (Network File System)
cid	เป็นโปรโตคอลที่กำหนด Content identifier
mid	เป็นโปรโตคอลที่กำหนด Message identifier

ตารางที่ 2.4 รายชื่อของโปรโตคอลที่ใช้งานบน เวิลด์ไวด์เว็บ (2)

โดยทั่วไปตัวอักษรภาษาอังกฤษสามารถอ้างอิงตามรูปแบบของ URL ได้ แต่มีตัวอักษรที่ถือว่าเป็นที่สงวนไว้ ห้ามใช้งานใน URL เนื่องจากถูก URL นำไปอ้างอิงไว้แล้ว เช่นตัวอักษร @ ใช้อ้างอิงระหว่าง Password และชื่อเซิร์ฟเวอร์ เป็นต้น ซึ่งอักษรสงวนนั้นได้แก่

: | / ? @ = < ~ { [

; ^ \ % & # > " }]

อย่างไรก็ตามหากมีความจำเป็นที่ต้องอ้างถึงตัวอักษรเหล่านี้ใน URL ก็ทำได้โดยใช้เครื่องหมาย % แล้วตามด้วยเลขฐานสิบหก 2 ตัวแทนตัวอักษรนั้นๆ

2.5.3 HTTP (Hyper Text Transfer Protocol)

HTTP เป็นกลไกหรือโปรโตคอลหลักที่ใช้แลกเปลี่ยนข้อมูลกันระหว่างเซิร์ฟเวอร์และไคลเอนต์ของเวิลด์ไวด์เว็บ โดยถูกออกแบบมาให้มีความกะทัดรัด สามารถทำงานได้รวดเร็ว มีกระบวนการทำงานที่ไม่ซับซ้อน และมีคำสั่งที่ใช้งานไม่มากนัก แต่สามารถรองรับข้อมูลได้ทุกแบบ ไม่ว่าจะเป็นข้อมูลทั่วไปที่เข้ารหัสแบบ MIME หรือข้อมูลที่เป็นกราฟฟิก เช่น ไฟล์ที่เป็น GIF หรือ JPEG เป็นต้น

หลักการทำงานทั่วๆไปของ HTTP ก็คือ จะแบ่งการทำงานออกเป็น 2 ด้านคือ ด้านเว็บเซิร์ฟเวอร์และด้านไคลเอนต์ โดยไคลเอนต์จะติดต่อเข้ามายังเซิร์ฟเวอร์โดยใช้โปรแกรมบราวเซอร์และอ้างถึงแอดเดรสของเซิร์ฟเวอร์โดยใช้รูปแบบของ URL ส่วนด้านเซิร์ฟเวอร์จะส่งข้อมูลกลับมาในรูปแบบที่เป็นภาษา HTML (Hyper Text Markup Language) โดยที่โปรโตคอล HTTP ใช้วิธีการเข้ารหัสในแบบ MIME เป็นมาตรฐานของการทำงาน

โครงสร้างข้อมูลของ HTTP จะแบ่งออกเป็น 2 ส่วนใหญ่ๆคือ ส่วนเฮดเดอร์ หรือที่เรียกว่า metadata จะเป็นส่วนเก็บข้อมูลที่จำเป็นต้องใช้ภายในโปรโตคอล ส่วนที่สองเป็นส่วนเป็นข้อมูลจริงที่ต้องการรับส่ง ทั้งนี้ HTTP ถูกออกแบบมาให้สามารถรับส่งข้อมูลผ่าน Proxy หรือ Firewall ต่างๆได้โดยการทำงาน HTTP จะอาศัยโปรโตคอลพื้นฐาน TCP/IP ซึ่งทั่วไปจะใช้หมายเลขพอร์ตที่ 80

โปรโตคอล HTTP ในปัจจุบันได้พัฒนาขึ้นมาเป็นเวอร์ชั่น 1.1 (จากเดิม 1.0) ซึ่งโปรแกรมบราวเซอร์ที่แพร่หลายทั่วไปนั้นจะสามารถรองรับโปรโตคอลในเวอร์ชั่นใหม่นี้ได้ โดยใน HTTP เวอร์ชั่น

1.1 นี้ได้เพิ่มประสิทธิภาพการทำงานให้สูงขึ้น และปรับปรุงในด้านต่างๆที่ทำให้มีความสามารถมากขึ้น ดังนี้

- ลดภาระของการเชื่อมต่อผ่านโปรโตคอล TCP และสามารถใช้ประสิทธิภาพของ TCP ได้อย่างเต็มที่
- สามารถทำการบีบอัดข้อมูลที่รับส่งระหว่างเซิร์ฟเวอร์และไคลเอนต์ได้
- รองรับการทำงานแบบ virtual host หมายถึง เว็บเซิร์ฟเวอร์เครื่องหนึ่งมีชื่อโดเมนมากกว่า 1 ชื่อได้
- สามารถรองรับการทำงานได้หลายภาษา
- โอนไฟล์ข้อมูลเฉพาะบางส่วนได้ ซึ่งคุณสมบัตินี้จะมีประโยชน์มากในกรณีที่มีการโอนไฟล์ข้อมูลขนาดใหญ่ และเกิดปัญหาขึ้นระหว่างการทำงาน ซึ่งโปรโตคอล HTTP 1.1 มีจุดเด่นที่สามารถตรวจสอบได้ และโอนไฟล์ข้อมูลต่อจากส่วนที่เคยโอนมาแล้วได้

➢ คำสั่งของโปรโตคอล HTTP

HTTP มีคำสั่งต่างๆไม่มากนัก เพื่อให้สามารถใช้งานได้อย่างสะดวกและรวดเร็ว โดยที่มีคำสั่งที่ใช้งานแพร่หลายอยู่เพียง 3 คำสั่งคือ GET, HEAD และ POST ส่วนคำสั่งอื่นๆอีก 4 คำสั่งคือ PUT, DELETE, LINK และ UPLINK มีให้ใช้งานเช่นกัน แต่ไม่เป็นที่นิยมมากนัก รายละเอียดของคำสั่งมีดังนี้

คำสั่ง	รายละเอียด
GET	ใช้อ่านข้อมูลจากเว็บเซิร์ฟเวอร์และส่งไปยังไคลเอนต์ โดยมีรูปแบบดังนี้ GET <URL>HTTP/1.0 ตัวอย่างเช่น ต้องการให้เว็บเซิร์ฟเวอร์ส่งไฟล์ sale.html จากโดเมน www.netcorp.com ไปยังไคลเอนต์จะใช้รูปแบบของคำสั่ง GET ดังนี้ GET www.netcorp.com/sale.html/HTTP/1.0 นอกจากนี้คำสั่ง GET ยังสามารถกำหนดเงื่อนไขให้อ่านข้อมูลจากเว็บเ ซิร์ฟเวอร์เฉพาะที่มีการเปลี่ยนแปลงแก้ไขได้ด้วย
HEAD	คำสั่งนี้จะทำงานคล้ายกับคำสั่ง GET แต่เว็บเซิร์ฟเวอร์จะส่งข้อมูลกลับมาให้เฉพาะในรายละเอียดของ metadata หรือข้อมูลในเฮดเดอร์เท่านั้น ส่วนข้อมูลที่เป็น HTML จะไม่ถูกส่งมาด้วย ซึ่งคำสั่ง HEAD นี้จะใช้เพื่อทดสอบว่าข้อมูลตาม URL นั้นๆมีการเปลี่ยนแปลงหรือไม่เท่านั้น

ตารางที่ 2.5 รายละเอียดคำสั่งของ HTTP (1)

คำสั่ง	รายละเอียด
POST	เป็นคำสั่งที่ตรงข้ามกับคำสั่ง GET และ HEAD โดยทำหน้าที่ส่งข้อมูลจากไคลเอนต์ไปยังเว็บเซิร์ฟเวอร์ แต่โดยปกติแล้วการส่งข้อมูลจากไคลเอนต์ไปยังเว็บเซิร์ฟเวอร์นั้นจะไม่ค่อยมีใช้งาน นอกจากในกรณีที่ HTML ทำงานในลักษณะที่ให้ผู้ใช้กรอกข้อมูลลงตามแบบฟอร์ม (เช่น รายละเอียดส่วนตัวของผู้ใช้งาน) และส่งข้อมูลนี้กลับมาเก็บไว้ที่เว็บเซิร์ฟเวอร์
PUT	เป็นคำสั่งที่ทำงานเหมือนกับคำสั่ง POST แต่ไม่เป็นที่นิยม
DELETE	เพื่อให้ไคลเอนต์สั่งเว็บเซิร์ฟเวอร์ลบ URL ที่กำหนดไว้ออกจากเซิร์ฟเวอร์ แต่เป็นคำสั่งที่ไม่นิยมใช้มากนัก เนื่องจากเว็บเซิร์ฟเวอร์ทั่วไปมักจะทำงานในแบบอ่านข้อมูลได้เท่านั้น (read-only)
LINK	เป็นคำสั่งที่เชื่อม URL ที่ต้องการไปยังเว็บเซิร์ฟเวอร์อื่น
UNLINK	ยกเลิกคำสั่ง LINK ให้กลับมาใช้เซิร์ฟเวอร์เดิมตามที่กำหนดไว้ใน URL

ตารางที่ 2.5 รายละเอียดคำสั่งของ HTTP (2)

➢ สถานะการทำงานของ HTTP

ในโปรโตคอล HTTP ได้กำหนดรหัสแสดงการทำงานของโปรโตคอลไว้ โดยแบ่งกลุ่มของรหัสสถานะออกเป็น 5 กลุ่มคือ

รหัสสถานะ	ประเภท	รายละเอียด
100-199	Infomational	เป็นรหัสสถานะกลุ่มที่เปิดให้โปรแกรมประยุกต์ต่างๆ กำหนดใช้งานได้เอง
200-299	Successful	กลุ่มรหัสที่แสดงว่าการทำงานสำเร็จ
300-399	Redirection	กลุ่มรหัสนี้จะใช้ภายในโปรโตคอล HTTP เอง โดยเป็นการทำงานที่ต่อเนื่องมาจากโปรเซสก่อนหน้านี้ ซึ่งไคลเอนต์เป็นผู้สั่งงาน
400-499	Client Error	ใช้แสดงการปัญหาที่เกิดขึ้นกับไคลเอนต์
500-599	Server Error	ใช้แสดงการปัญหาที่เกิดขึ้นกับเซิร์ฟเวอร์

ตารางที่ 2.6 รหัสแสดงสถานการณ์ทำงานต่างๆของ HTTP

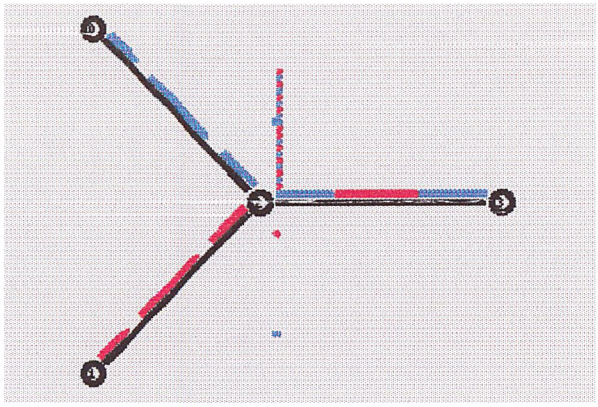
รหัสแสดงสถานะในแต่ละตัวจะนำหน้าด้วยตัวเลข 3 หลักและตามด้วยตัวอักษร ซึ่งรหัสในกลุ่ม 100-199 จะเปิดกว้างให้ผู้ที่พัฒนาโปรแกรมประยุกต์สามารถกำหนดค่าขึ้นมาใช้งานได้เอง ส่วนรายละเอียดของรหัสในกลุ่มอื่นๆมีดังนี้

รหัสสถานะ	รายละเอียด
200 OK	การทำงานสำเร็จเรียบร้อย
201 Created	คำสั่ง POST ทำงานเสร็จสมบูรณ์
202 Accepted	ได้รับคำสั่งให้ทำงานเรียบร้อย แต่ไม่ต้องมีการตอบกลับ
204 No Content	ทำงานตามคำสั่งเรียบร้อย แต่ไม่ต้องแสดงข้อความใดๆบนหน้าจอ
300 Multiple Choices	ถ้าค้นหาและพบแหล่งข้อมูลที่ต้องการหลายแห่ง เซิร์ฟเวอร์จะตอบกลับไปทั้งหมดเพื่อให้ไคลเอนต์สามารถเลือกแหล่งข้อมูลที่ต้องการเองได้
301 Moved Permanently	URL ที่ร้องขอได้ถูกย้ายไปที่อื่นแล้ว ดังนั้นการร้องขอใช้งาน URL จะต้องเปลี่ยนเป็นแอดเดรสใหม่
302 Moved Temporarily	URL ที่ร้องขอมาได้ถูกย้ายไปที่อื่นชั่วคราว
304 Not Modify	ใช้แสดงสถานะเมื่อใช้คำสั่ง GET ที่กำหนดเงื่อนไขเฉพาะเว็บไซท์ที่มีการเปลี่ยนแปลง ส่วนเว็บไซท์ที่ไม่มีการเปลี่ยนแปลงจะแสดงด้วยสถานะนี้
400 Bad Request	คำสั่งจากไคลเอนต์ไม่ถูกต้อง
401 Unauthorized	ปฏิเสธการทำงานจากไคลเอนต์ที่ไม่ได้รับอนุญาต
403 Forbidden	เซิร์ฟเวอร์ไม่อนุญาตให้ใช้งาน หรือไคลเอนต์มีสิทธิในการใช้งานไม่เพียงพอ
404 Not Found	ไม่พบเว็บเซิร์ฟเวอร์ตาม URL ที่กำหนด
500 Internal Server Error	เซิร์ฟเวอร์มีปัญหา
501 Not Implemented	เซิร์ฟเวอร์ไม่รองรับคำสั่งที่สั่งไป
502 Bad Gateway	Proxy Server รับคำสั่งที่ไม่ถูกต้องจากเว็บเซิร์ฟเวอร์
503 Service Unavailable	เซิร์ฟเวอร์กำลังทำงานอื่นอยู่ ไม่สามารถให้บริการได้ในขณะนี้

ตารางที่ 2.7 รายละเอียดแสดงสถานะกลุ่มอื่นๆของ HTTP

2.6 โปรแกรม Network Simulator (NS)

NS เป็นโปรแกรมที่ใช้ในการจำลองระบบเครือข่ายที่นักวิจัยทางด้านโครงข่ายใช้กันอย่างกว้างขวางทั่วโลกซึ่งโปรแกรม NS ถูกเขียนขึ้นมาด้วยภาษา C++ และภาษา OTcl จาก VINT Project ซึ่งเป็นการพัฒนาร่วมกันของนักวิจัยที่ UC Berkeley, LBNL, USC/ISI, และ Xerox PARC ซึ่งทำงานบนระบบปฏิบัติการยูนิกซ์เป็นหลักแต่ก็สามารถใช้งานบนระบบปฏิบัติการวินโดวส์ได้ การใช้งานโปรแกรม NS จะต้องเขียนโปรแกรมภาษา OTcl Script, Tcl Script ในการปรับแต่งค่าพารามิเตอร์และสั่งให้โปรแกรม NS ทำงาน รูปแบบการสั่งให้โปรแกรมทำงานจะเป็นดังนี้ ns <tclscript>

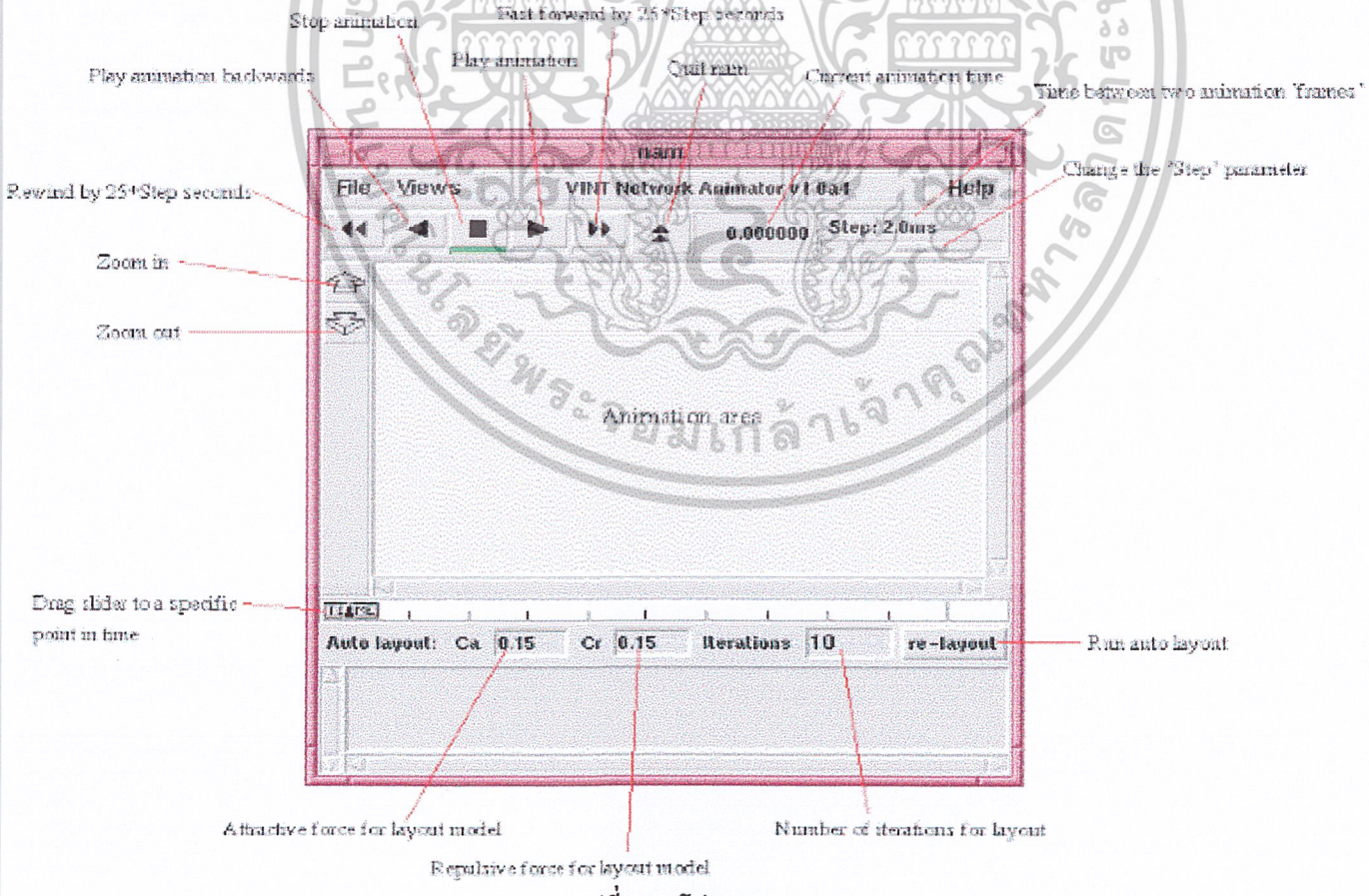


รูปที่ 2.16 ผลโปรแกรม NS

2.7 เครื่องมือที่ใช้ร่วมกับโปรแกรม NS

2.7.1 NAM (Network Animator)

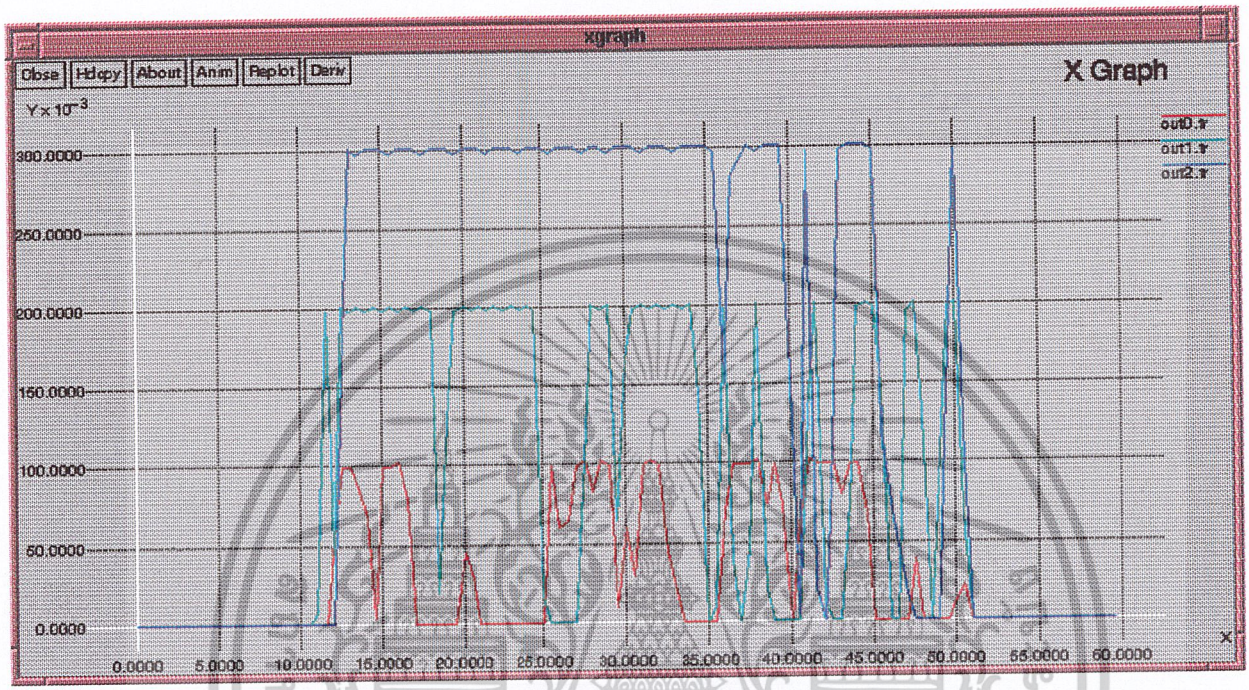
โปรแกรม NAM (Network Animator) เป็นโปรแกรมที่ใช้ในการแสดงผลที่ออกแบบมาสำหรับโปรแกรม NS โดยเฉพาะโดยจะมีปุ่มควบคุมการทำงานคล้ายกับเครื่องเล่นซีดี ซึ่งจะสามารถสั่งให้ NAM ทำงานได้ด้วยคำสั่ง $ns namtrace-all <$nam-trace-file>



รูปที่ 2.17 โปรแกรม NAM

2.7.2 Xgraph

Xgraph เป็นโปรแกรมที่ใช้พล็อตกราฟในแกน x-y โดยจะรับค่าจากไฟล์หรืออินพุตมาตรฐานในรูปแบบแอสกี (ASCII format) ด้วยค่าตัวเลขสองค่าในแต่ละบรรทัดและจะวาดกราฟในแกน x-y ตามค่าที่ใส่เข้าไป โดยสามารถสั่งให้ทำงานได้ด้วยคำสั่งตามนี้ xgraph <$file>



รูปที่ 2.18 โปรแกรม xgraph

2.8 คำสั่ง NS พื้นฐาน

2.8.1 ตัวตารางงานเหตุการณ์

2.8.1.1 สร้างตัวตารางงาน

set ns [new Simulator]

2.8.1.2 เหตุการณ์ตัวตารางงาน

$ns at <time> <event>

2.8.1.3 เริ่มตัวตารางงาน

$ns run

2.8.2 สร้างโทโปโลยี (Topology)

2.8.2.1 สร้างโหนด

set n0 [$ns node]

2.8.2.2 การลิงค์(Link) และ คิว

$ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>

2.8.3 สร้างการเชื่อมต่อกับ UDP

set udp [new Agent/UDP]

set null [new Agent/Null]

$ns attach-agent $n0 $udp

$ns attach-agent $n1 $null

$ns connec $udp $null

2.8.4 สร้างการเชื่อมต่อกับ TCP

set tcp [new Agent/TCP]

set tcpsink [new Agent/TCPSink]

$ns attach-agent $n0 $tcp

$ns attach-agent $n1 $tcpsink

$ns connect $tcp $tcpsink

2.8.5 สร้างทราฟฟิก(Traffic)

2.8.5.1 FTP

set ftp [new Application/FTP]

$ftp attach-agent $tcp

2.8.5.2 Telnet

set telnet [new Application/Telnet]

$telnet attach-agent $tcp

2.8.5.3 CBR

set src [new Application/Traffic/CBR]

2.8.5.4 Exponential or Pareto on-off

set src [new Application/Traffic/Exponential]

set src [new Application/Traffic/Pareto]

2.8.6 การกำหนดเส้นทางเดิน

2.8.6.1 ยูนิแคส (Unicast)

$ns rtproto <type>

<type>: Static, Session, DV, cost, multi-path

2.8.6.2 มัลติแคส (Multicast)

$ns multicast

$ns mrtproto <type>

<type> : CtrMcast, DM, ST,BST

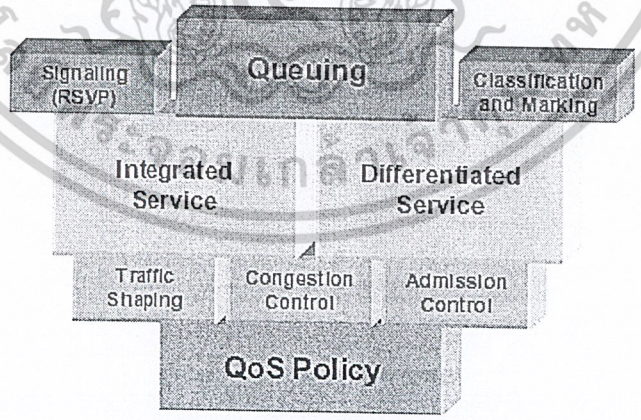
บทที่ 3
วิธีการลดความหนาแน่นของทราฟฟิกแบบต่างๆ

3.1 ความหมายของ QUALITY OF SERVICE (QoS)

การทำ Quality of Service (QoS) นี้สามารถที่จะให้คำจำกัดความได้ว่า เป็นวิธีการที่ระบบเครือข่ายที่ใช้งานนั้นมีการเปลี่ยนแปลงคุณภาพและระดับของการให้บริการ เพื่อรองรับการใช้งานลักษณะต่างๆที่ดีขึ้น ในเวลาเดียวกันกับที่ช่องสัญญาณในระบบเครือข่าย มีการใช้งานสูงที่สุด ซึ่งวงจรสวิทช์ซึ่งจะมีประสิทธิภาพสูงที่สุดขนาดไหนที่จะรองรับการใช้งานต่างๆที่จะสร้างทราฟฟิกจำนวนมาก โดยการทำ QoS นี้จะมีการปรับช่องสัญญาณ ให้คงที่ และมีการล่าช้าน้อยที่สุด แต่อย่างไรก็ตาม การใช้งานรูปแบบต่างๆนั้นจะขึ้นอยู่กับอัตราส่วนระหว่างทราฟฟิกที่เปลี่ยนแปลงไปกับช่องสัญญาณนั้น เพราะเป็นผลมาจากการพยายามที่จะให้การบริการที่ดีที่สุดภายในเครือข่ายที่มีความหนาแน่นสูงนั่นเอง

ซึ่งมีความพยายามที่จะใช้งานโดยลดผลกระทบอันนี้ โดยใช้งานอุปกรณ์อย่างเช่น Packet Switching แต่ก็ยังไม่สามารถรองรับข้อมูลที่มีคุณภาพสูงอย่างเช่น ภาพวีดีโอและเสียง เป็นต้น ซึ่งเทคโนยีระบบเครือข่ายในปัจจุบันและอนาคตอย่างเช่น ระบบ Asynchronous Transfer Mode (ATM), Internet Protocol Differentiated Service (IP DiffServ) และ Multiprotocol Label Switching (MPLS) เป็นต้น นั้นมีวัตถุประสงค์หลักในการเลือกใช้งานระบบเครือข่ายเหล่านี้ก็คือการทำอย่างไรให้วิธีการ QoS มีการจัดการปริมาณ ทราฟฟิกที่มีการใช้ช่องสัญญาณสูงสุดในเวลาเดียวกันได้ง่ายและมีประสิทธิภาพสูงสุด

- **องค์ประกอบพื้นฐานของการทำ QoS**



รูปที่ 3.1 องค์ประกอบพื้นฐานของการทำ QoS

ซึ่งจากรูปที่ 3.1 นั้นเราแสดงส่วนประกอบของการทำ Quality of Service โดยหน้าที่ในส่วนต่างๆนั้นจะมีหน้าที่ดังนี้

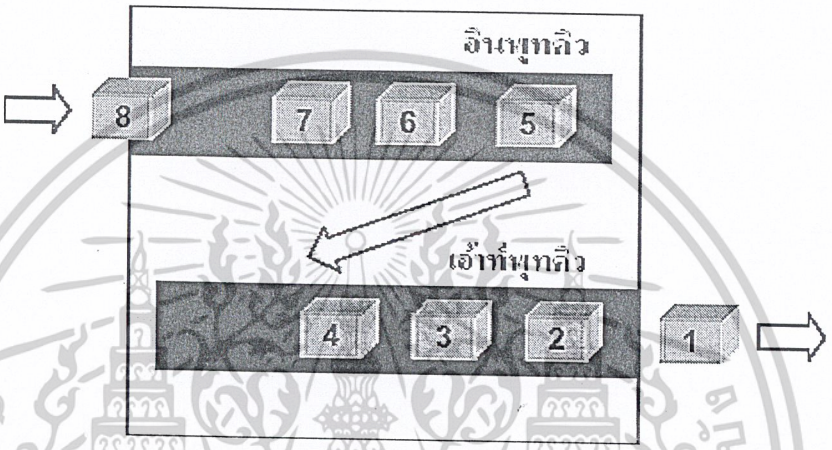
- **Signaling** เป็นสัญญาณในการที่บอกว่าต้องการจองช่องสัญญาณของระบบ
- **Queuing** จัดลำดับความสำคัญของ Packets ต่างๆตามกฎเกณฑ์ของคิวชนิดที่ใช้งาน
- **Classification** ตรวจจับทราฟฟิกที่ได้รับการแบ่งลำดับชั้นมาแล้วหรือปริมาณที่ไหลมา
- **Marking** Packets ของข้อมูลบางชนิดจะมีการถูกทำเครื่องหมายพิเศษเอาไว้เมื่อมีความต้องการที่จะได้รับบริการที่พิเศษที่ต่างจากข้อมูลชนิดอื่นๆ
- **Integrated and Differentiated Service** เป็นการทำงานที่ให้บริการข้อมูลที่ต่างกัน 2 ชนิดในวิธีการทำงาน QoS จะอยู่ในส่วนการทำงานภาคหลังๆของการให้บริการ
- **Traffic shaping** ทำการลดปริมาณทราฟฟิกที่มีความหนาแน่นสูงให้เบาบางลง
- **Congestion control** ลดอัตราการไหลของ Packets เมื่อระบบเครือข่ายมีความหนาแน่นของปริมาณทราฟฟิกเพิ่มขึ้น
- **Admission control** แบ่งแยกความสามารถหรือคุณสมบัติต่างๆของ QoS ตามการใช้งานในรูปแบบต่างๆจากผู้ใช้
- **QoS policy** กฎเกณฑ์หรือข้อกำหนดต่างๆที่จะใช้ในการจัดการทราฟฟิกต่างๆในระบบอย่างไร

ซึ่งในส่วนของการจัดการความหนาแน่นของปริมาณทราฟฟิกในระบบเครือข่ายนั้น จะอยู่ในส่วนของการทำ Queuing Management ในบล็อค Queuing ดังรูปที่ 3.1 ซึ่งในปริญญานิพนธ์นี้ได้นำเอาวิธีการจัดการคิวแบบต่างๆบางส่วนมาทำการศึกษาและหาความสามารถของแต่ละชนิดว่ามีข้อดีหรือด้อยอย่างไร โดยวิธีการจัดการคิวที่จะนำเสนอเช่น

- **First In First Out (FIFO), DropTail Queuing**
- **Fair Queuing (FQ)**
- **Stochastic Fairness Queuing (SFQ)**
- **Deficit Round Robin (DRR) Queuing**
- **Random Early Detection (RED) Queuing**
- **Priority Queuing**
- **Class-Based Queuing (CBQ)**

3.2 First In First Out (FIFO), DropTail Queuing

FIFO Queuing เป็นกลไกการทำงานพื้นฐานที่สุดที่จะนึกถึงได้ ซึ่งในการทำงานของมันนั้น Packets จะถูกนำเข้ามาอย่างต่อเนื่องภายในระบบเครือข่าย ซึ่งภายในนี้จะมีคิวอยู่เพียงคิวเดียวเท่านั้นเป็นคิวเดี่ยว ซึ่งมันไม่ต้องการการจัดแบ่งหรือการจัดเรียงของ Packets ใดๆเลย โดยจะเข้ามาทางท้ายของคิวและเคลื่อนย้ายออกไปทางส่วนหัวของคิว FIFO เป็นวิธีการดั้งเดิมที่ใช้กันกันอย่างแพร่หลายในอุปกรณ์เครือข่ายอย่างง่าย แต่อย่างไรก็ดีมันไม่ได้เป็นการทำงานที่ดี ในการที่จะให้บริการระบบที่มีความหนาแน่นของปริมาณทราฟฟิกสูงๆ เช่น มันไม่ได้มีความยืดหยุ่นในระบบ QoS ซึ่งต้องการการให้บริการเฉพาะอย่างที่แตกต่างออกไป ซึ่งโครงสร้างของ FIFO จะแสดงได้ดังรูปที่ 3.2



รูปที่ 3.2 โครงสร้างของ FIFO Queuing

ซึ่งลักษณะของ FIFO Queuing ก็คือ

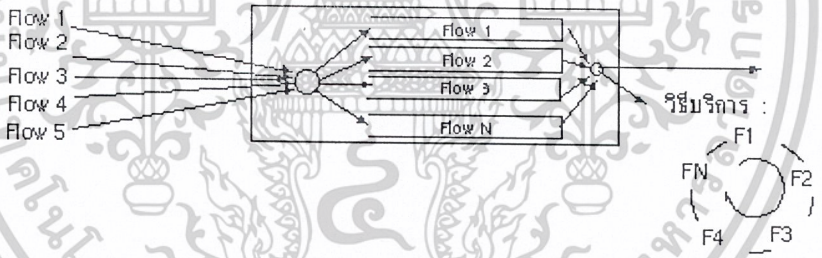
- จะมีความเร็วและง่ายที่สุดเพราะ Packets ที่เข้ามาในระบบจะได้รับการบริการทันที
- ไม่มีการแยกแยะชนิดของข้อมูลที่เข้ามาว่ามีความแตกต่างกันอย่างไร
- DropTail คือวิธีการจัดการบัฟเฟอร์ของ FIFO Queuing
- วิธีการนี้เมื่อมี Packets ของข้อมูลจำนวนมากไหลเข้ามาในคิวของระบบจนเต็ม Packets ต่างๆที่เหลือจะถูกหยุด (drop) ไม่สามารถเข้ามาในระบบได้ ทำให้มีปัญหาเรื่อง การสูญเสียของ Packets มาก

3.3 Fair Queuing (FQ)

Round Robin เนื่องจากการทำงาน ของ FIFO นั้นไม่ได้มีการทำงานที่จะเป็นการลดภาระการทำงานของ Router ที่ต้องรองรับปริมาณทราฟฟิกสูงๆได้ และไม่มีความเป็นธรรม (unfairness) ในการให้บริการที่มาจากต่างเส้นทาง และปัญหาหลักๆของ FIFO ก็คือเมื่อมีการส่งข้อมูลมากขึ้นความต้องการในการใช้ช่องสัญญาณจะมากขึ้นซึ่งจะทำให้กลายเป็นตัวที่สร้างความหนาแน่นซะเอง

รูปที่ 3.3 เปรียบเทียบโครงสร้างของ FIFO และ Round Robin

ดังนั้นจึงมีการสร้างวิธีการที่จะให้ความเป็นธรรม (fairness) ในการที่จะให้บริการกับข้อมูล โดยที่จะทำการจัดเก็บ Packets ลงในคิวที่ต่างกันโดยใช้วิธีแบ่งตาม แอดเดรสต้นทางของ Packets ที่ส่งมาแล้วให้บริการส่งทีละ Packet ต่อรอบเท่านั้นซึ่งอาจจะเรียกได้ว่าเป็นวิธีแบบ Packet Round Robin ก็ได้

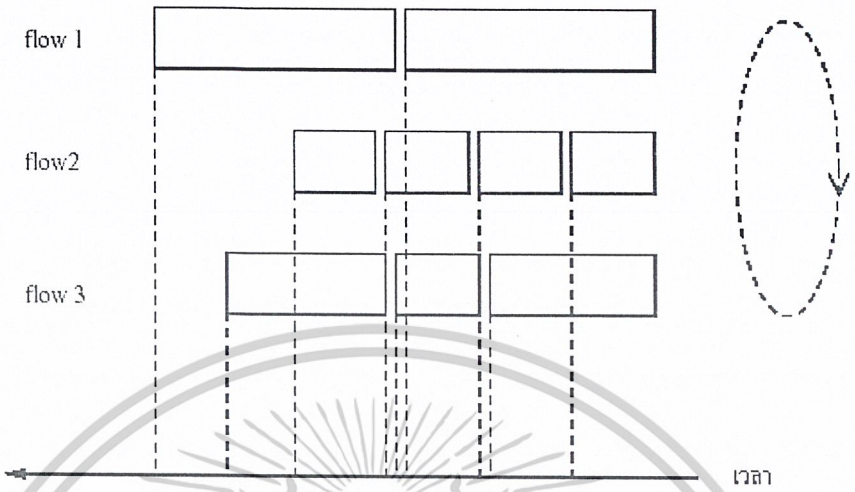


รูปที่ 3.4 ลักษณะการทำงานของ Round Robin

ซึ่งจากรูปที่ 3.4 นั้นจะเห็นว่าเมื่อมีการไหลของข้อมูลมาจากต้นทางที่ต่างกัน Router ที่มีการให้บริการจัดการข้อมูลแบบ Roud Robin จะให้บริการทีละคิวตามลำดับ โดยที่จะส่งทีละ Packet ต่อการให้บริการใน 1 คิว แล้วจึงจะไปทำการส่ง Packet ของคิวตัวถัดไป

Fair Queuing จากลักษณะของการให้บริการแบบ Round Robin นั้นจะเห็นว่าเป็นการส่งทีละ Packet ต่อคิว แล้วจึงไปให้บริการในคิวถัดไป ซึ่งจะเกิดปัญหาก็คือ จะเกิดความไม่เป็นธรรมในการจัดแบ่งช่องสัญญาณ เพราะว่าถ้าเกิดคิวใดมี Packet ที่มีความยาวมากๆแล้วนั้นจะทำให้มีความต้องการที่จะใช้งานช่องสัญญาณสูงกว่า Packet ที่มีสั้น ซึ่งทำให้เกิดความล่าช้าในการส่งและเสี่ยงต่อการที่จะเกิดการสูญหายของ Packets ได้มาก ซึ่งเป็นผลกระทบอย่างเดียวกันกับ FIFO ดังนั้น ดังนั้นถ้าต้องการให้ผลของการ

ให้บริการมีความเป็นธรรมทั้งในการบริการและการจัดสรรช่องสัญญาณ เราจะใช้วิธีการส่งแบบ Bit-by-Bit Round Robin ดังรูปที่ 3.5



รูปที่ 3.5 Bit-by-Bit Round Robin

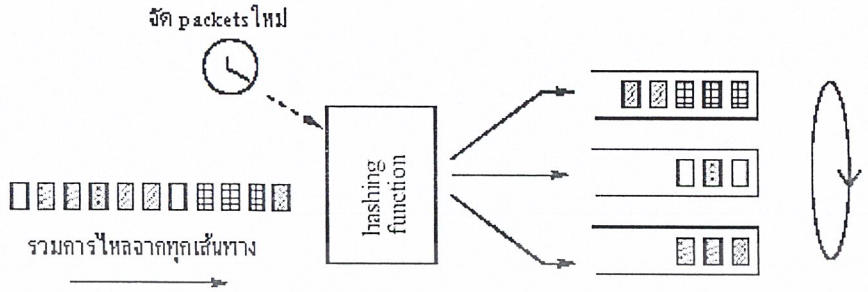
ลักษณะของ Bit-by-Bit Round Robin นั้น จะส่งทีละ Bit ต่อคิว ซึ่งจะต่างกับการส่งแบบ Packet Round Robin ที่จะส่งทีละ Packet ที่จะกำหนดค่าเป็น byte ซึ่งมีค่ามากกว่า โดยจากรูปที่ 3.5 แล้วเมื่อมีการไหลของข้อมูล 3 เส้นทาง ก็จะให้บริการทีละคิวซึ่งจะทำการส่งทีละ bit ของ Packet ต่อคิว ซึ่งในวิธีของ Bit-by-Bit Round Robin นี้ Packets จะถูกทำเครื่องหมายซึ่งแสดงลักษณะของ Packets นั้นๆเอาไว้ ไม่ว่าจะเป็นเวลาในการส่งมา ขนาด และอื่นๆ เพื่อที่จะบอกว่าได้ทำการส่ง bit ข้อมูลของ Packet นั้นไปหมดแล้ว วิธีการ Bit-by-Bit Round Robin นี้ สามารถที่จะกำหนดให้มีการส่งต่อครั้งว่าเป็นกี่ bit ของคิวใดๆก็ได้ ซึ่งเรียกว่าเป็นการให้น้ำหนัก (Weighted) นั่นเอง

3.4 Stochastic Fairness Queuing (SFQ)

จากวิธีของ Fair Queuing นั้นจะเห็นว่าเมื่อมีการไหลของข้อมูลมาจากหลายๆเส้นทางนั้นก็จะทำให้ต้องมีจัดสรรการใช้งานช่องสัญญาณให้เพียงพอในแต่ละคิวมากขึ้น ซึ่งหมายความว่าช่องสัญญาณรวมของ Router ก็จะต้องเพิ่มมากขึ้นไปด้วย ซึ่งจะทำให้รายจ่ายในการที่จะแลกกับการเพิ่มบัฟเฟอร์เพื่อที่จะรองรับปริมาณทราฟฟิกที่เพิ่มสูงขึ้นตามไปด้วย ซึ่งในความเป็นจริงแล้วนั้น เราไม่สามารถที่จะเพิ่มบัฟเฟอร์เพื่อที่จะให้จำนวนของช่องสัญญาณเพิ่มขึ้นทุกครั้งได้ เพราะค่าใช้จ่ายที่สูงในการที่จะเปลี่ยนอุปกรณ์ตัวใหม่

ดังนั้นเราจะใช้วิธีการของ Stochastic Fairness Queuing เพื่อจัดการกับปัญหานี้โดยที่จะใช้การรวมการไหลของข้อมูลจากต่างเส้นทางกันนั้นให้รวมกันเป็นอันเดียวแล้ว Stochastic Fairness Queuing นี้จะทำการกระจาย Packets จากแอดเดรสที่ต่างกันนั้นลงในคิวต่างๆที่ถูกกำหนดไว้คงที่โดยจะไม่เปลี่ยนแปลงตามจำนวนเส้นทางการไหลของข้อมูล ซึ่งส่วนที่จะทำหน้าที่กระจาย Packets ต่างลงไปในคิวที่มี

จำนวนคงที่เหล่านี้นั้นคือ Hashing Function นั่นเองซึ่งลักษณะของ Stochastic Fairness Queuing จะแสดงได้ดังรูปที่ 3.6 นั่นเอง



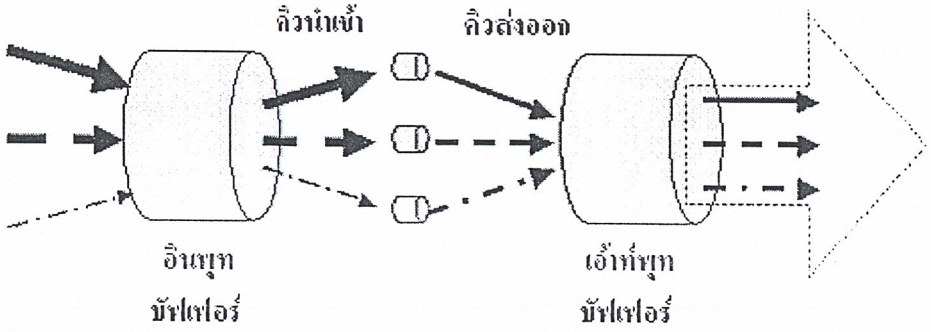
รูปที่ 3.6 Stochastic Fairness Queuing

ซึ่งการทำงานนั้นจะเห็นว่ามี Packets อยู่ 6 ชนิดซึ่งแทนว่าเป็น Packets ที่มาจาก 6 เส้นทางการไหลของข้อมูล แล้วถูกนำมารวมกันเป็นเส้นทางเดียวก่อนที่จะถูก Hashing Function กระจาย Packets เหล่านี้ไปลงในคิว ซึ่งจะเห็นว่าคิวที่มีนั้นได้แบ่งเพียง 3 คิวเท่านั้นเพราะว่า มีช่องสัญญาณจำกัด ซึ่งจะแบ่งเป็น 6 คิวเท่ากับจำนวนเส้นทางการไหลทั้งหมดก็จะไม่มีประสิทธิภาพเพียงพอที่จะให้บริการที่ดีกับแต่ละคิว ดังนั้นจึงทำการกำหนดค่าของคิวให้คงที่ เพียงแค่ 3 คิวเท่านั้น ซึ่งหลักการกระจาย Packets ของ Hashing Function ก็คือการจัด แอดเดรสที่มาจากเส้นทางต่างกันลงไปในคิว ซึ่งเมื่อแอดเดรสใดถูกจัดลงไปในคิวแล้ว Packets ที่มาจากแอดเดรสนั้นก็จะถูกนำไปวางที่คิวตัวเดิมเสมอ และจากการที่มีจำนวนของคิวที่น้อยกว่าเส้นทางการไหลของข้อมูลทั้งหมดดังนั้น ใน 1 คิวนั้นอาจจะต้องมี packets ที่มาจากแอดเดรสต่างกันอยู่ ก็ได้ ซึ่งจากรูปที่ 3.6 นั้นในแต่ละคิวจะต้องรองรับ Packets ที่มาจากแอดเดรสที่ต่างกันอยู่ 2 แอดเดรส ซึ่งค่าเหล่านี้นั้นจะถูกเก็บเอาไว้ใน Hash Table ซึ่งจะเก็บข้อมูลเหล่านี้เอาไว้เพื่อใช้ในการกระจาย Packets ลงในคิวต่างๆ

โดยการส่ง Packets ออกไปนั้นจะถูกให้บริการแบบ Round Robin หรือ Bit-by-Bit Round Robin ก็ได้ ซึ่งจากการที่จำนวนของคิวน้องกว่าเส้นทางการไหลของข้อมูลทั้งหมดจะทำให้ ความเป็นธรรมในการให้บริการส่ง Packets นั้นขึ้นอยู่กับค่าของ Hashing Table และลำดับการไหลเข้าของ Packets นั่นเอง

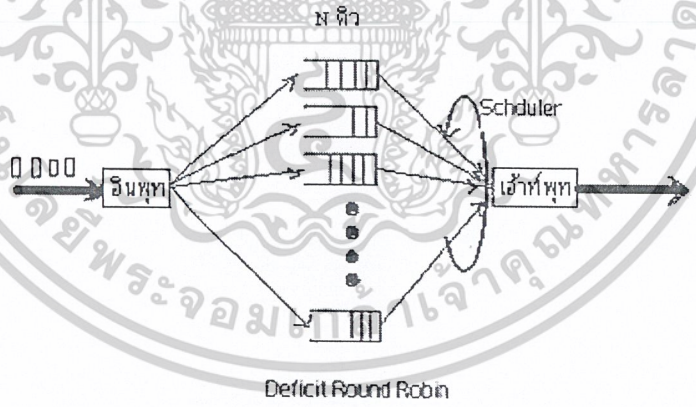
3.5 Deficit Round Robin (DRR) Queuing

วิธีการของ Deficit Round Robin (DRR) จะแตกต่างกับวิธี Fair Queuing ซึ่งมันจะลดข้อบกพร่องของ Fair Queuing ซึ่งการเลือกหรือกระจาย Packets ลงในคิวนั้นจะใช้วิธีการของ Stochastic Fairness Queuing โดยที่โครงสร้างของมันนั้นสามารถแบ่งได้ออกเป็น 3 ส่วนดังนี้คือ อินพุทบัฟเฟอร์ ส่วนที่ทำการจัดการการไหลของ Packets ในคิว และส่วน อินพุทบัฟเฟอร์ ซึ่งเมื่อมีการไหลเข้าของ Packets ข้อมูลใน Router จะทำการจัดเรียงคิวใน อินพุทบัฟเฟอร์ ซึ่งมันจะกลายเป็นคิวที่ไหลเข้ามาเพื่อที่กระบวนการของการจัดการคิวแบบ DRR ก่อนจะปล่อยเข้าไปในเอ้าท์พุทบัฟเฟอร์



รูปที่ 3.7 โครงสร้างของ Deficit Round Robin

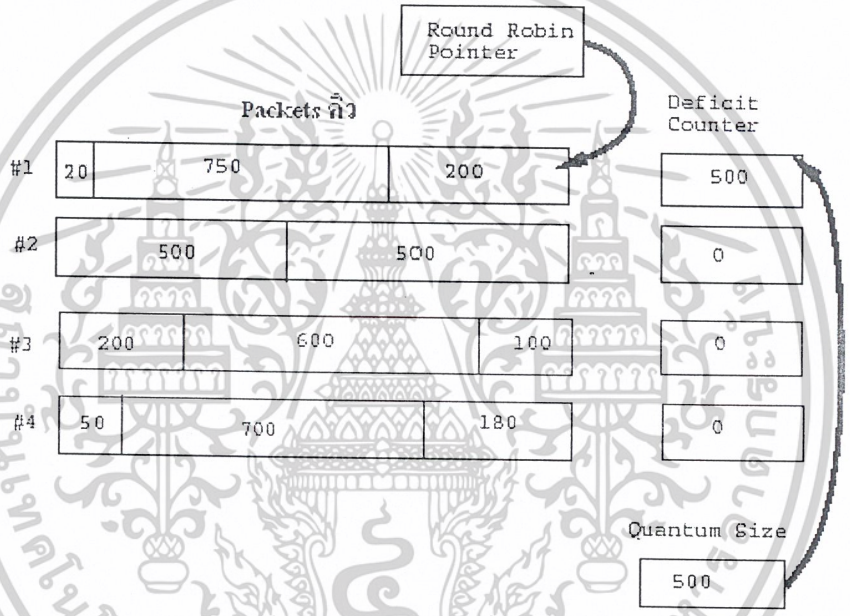
ในวิธีปกติของการทำ Round Robin แบบวิธีการของ Fair Queuing นั้น จะสามารถทำได้ในเวลาที่คงที่ ซึ่งปัญหาหลักก็คือมีความเป็นไปได้ที่จะเกิดการไม่เป็นธรรมในการให้บริการขึ้นได้เพราะจากขนาดที่แตกต่างกันของ Packets ที่ไหลเข้าไปในคิวที่ต่างกันเพราะว่าในการทำ Fair Queuing ธรรมดานั้นจะไม่สามารถจัดการกับ Packets ที่มีขนาดใหญ่มากโดยที่ไม่มีการเพิ่มขนาดช่องสัญญาณของคิวนั้นได้ ซึ่งปัญหาเหล่านี้นั้นได้รับการแก้ไขโดยใช้วิธีการที่จะนำเสนอต่อไปนี้ที่เรียกว่าวิธี Deficit Round Robin โดยใช้หลักการให้บริการแต่ละช่องสัญญาณที่ถูกแบ่งไว้เป็นคิวที่แตกต่างกันเหมือนกับ Fair Queuing นั่นคือวิธี Round Robin นั่นเองดังรูปที่ 3.8



รูปที่ 3.8 วิธีการทำงานของ Deficit Round Robin

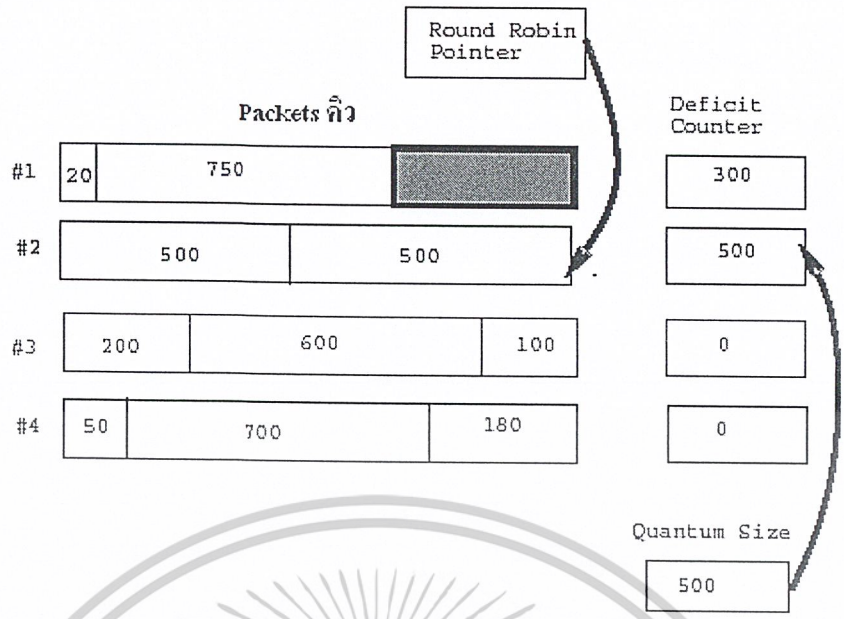
โดยวิธีการแบบ Deficit Round Robin นี้ มีวิธีการทำงานอย่างคร่าวๆก็คือ การกำหนดขนาดในการให้บริการของแต่ละคิวนี้ (Quantum size) ที่แตกต่างจากวิธีของ Round Robin แบบดั้งเดิมก็คือ ถ้าคิวนี้ไม่มีความสามารถในการที่จะส่ง Packets ในรอบที่แล้วเพราะว่ามีขนาดของ Packets ที่ใหญ่เกินไปนั้น Packets ในส่วนที่หลงเหลืออยู่จากการส่งในรอบที่แล้วนั้นจะถูกรวมเข้าไปกับขนาดของในรอบการส่งถัดไป ซึ่งส่วนที่มันไม่ได้รับการบริการนี้นี่เองจะถูกได้รับการบริการชดเชยในรอบถัดไปนั่นเอง

เรากำหนดให้ขนาดข้อมูลในคิวนี้มีขนาดหนึ่งๆ ซึ่งเป็นค่าของจำนวนบิตที่จะส่งในแต่ละรอบ เมื่อมี Packets เข้ามาในคิวที่ต่างกันนั้น เราจะอนุญาตให้มีการส่ง Packets ในรอบแรกนี้ได้นั่นก็คือ Packets ที่เข้ามานี้จะต้องมีขนาดน้อยกว่าหรือเท่ากับ Quantum Size นี้ ซึ่งถ้าไม่มี Packets อื่นภายในคิวนี้จะมีตัวแปรหรือตัวนับตัวหนึ่งที่เรียกว่า Deficit Counter ซึ่งมันจะทำการรีเซ็ตค่าตัวเองให้เป็น 0 แต่ถ้าในกรณีที่มี Packets มาต่อท้ายรอรับการบริการเช่นกันแต่ไม่สามารถให้บริการได้ในรอบนี้นั้นซึ่งค่าที่ Deficit Counter นี้จะเก็บเอาไว้ก็คือ ค่าแตกต่างระหว่าง Quantum Size กับขนาดของ Packets ที่ส่งไปแล้ว และในรอบของการบริการถัดไปนั้น ช่องสัญญาณที่ถูกแบ่งไว้ในแต่ละคิวนั้นจะสามารถใช้ได้เพิ่มกับการส่ง Packets ที่ยังคงเหลืออยู่จากการส่งในรอบที่แล้วนั่นคือค่า Quantum Size รวมกับค่าของ Deficit Counter นั่นเองซึ่งอธิบายได้ดังในรูปที่ 3.9 และ 3.10 นั่นเอง



รูปที่ 3.9 วิธีการส่ง Packets ขนาดต่างๆของ DRR ขณะเริ่มต้นการทำงาน

จากรูปที่ 3.9 นั้นขณะเริ่มต้นการทำงาน Deficit Counter จะถูกกำหนดค่าเริ่มต้นให้เป็นศูนย์ โดยตัวชี้ (Round Robin Pointer) จะชี้ไปที่ส่วนหัวของคิวที่มีข้อมูลอยู่ เมื่อคิวตัวแรกมีการให้บริการไปแล้วค่า Quantum Size ที่มีการกำหนดค่า 500 ไบท์ไว้จะถูกนำไปไว้ใน Deficit Counter ก่อนหน้านั้นเช่นกัน ซึ่งส่วนที่ยังเหลืออยู่จากการให้บริการในรอบแรกนั้นก็จะ ทำให้ Deficit Counter มีค่า 300 ไบท์ (500-200) เพื่อนำไปรวมกับรอบต่อไปดังรูปที่ 3.10



รูปที่ 3.10 วิธีการส่ง Packets ขนาดต่างๆของ DRR ขณะที่ส่ง Packets ไปแล้ว

หลังจากที่ส่ง Packets ขนาด 200ไบท์ ไปแล้วภายในคิวก็จะยังคงเหลือ Packets อยู่ในคิวที่กำหนด Quantum Size ไว้ที่ 500ไบท์ เป็น 300 ไบท์ เพราะว่ามันไม่สามารถส่ง Packets ขนาดนี้ได้ในรอบนี้ ที่มี Packets ที่รออยู่ขนาดถึง 750 ไบท์ ดังนั้นจำนวนที่เหลืออยู่นั้นจะถูกยกไปรวมกับรอบหน้าเมื่อมันสามารถมีจำนวนในการส่ง Packets รวมเป็น 300 ไบท์ (จำนวนที่เหลือมาจากรอบที่แล้ว) + 500 (Quantum Size)

ในกรณีของคิวที่ไม่มี Packets เข้ามานั้น เรามีตัวชี้ (Round Robin Pointer) ซึ่งปกติแล้วมันจะชี้อยู่ที่ส่วนหัวของคิวเมื่อในคิวนั้นมี Packets เข้ามา ซึ่งค่าของคิวนั้นจะมีค่าตามที่ได้กล่าวมาแล้ว แต่ในกรณีที่ไม่มี Packets เข้ามาในคิวเลย ตัวชี้นี้จะชี้ที่ท้ายของคิวซึ่งนั่นจะทำให้ มีการเซ็ตให้ค่าของ Deficit Counter มีค่าเท่ากับ 0 หรือเท่ากับค่าเมื่อเริ่มต้นการทำงานนั่นเอง

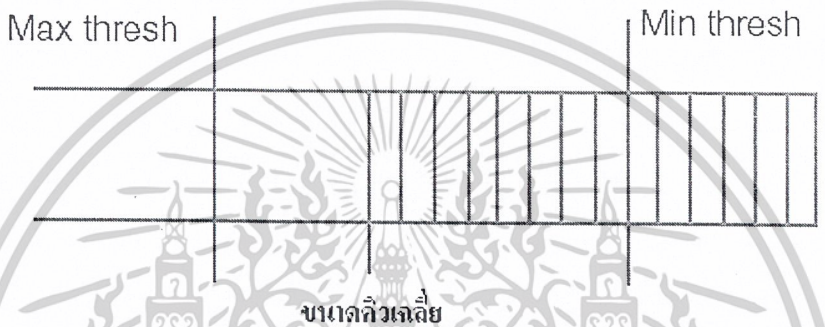
3.6 Random Early Detection (RED) Queuing

การที่การทำงานบนเครือข่ายไม่สามารถทำงานได้เต็มประสิทธิภาพ อาจจะเป็นเพราะมีความหนาแน่นของทราฟฟิกสูงเกินไป จึงทำให้มีการลดทอนประสิทธิภาพของการทำงานลงไป วิธีการแบบ RED นี้ถูกออกแบบมาสำหรับปรับปรุงประสิทธิภาพของระบบเครือข่ายให้ดีขึ้น โดยการปรับปรุงที่ต่างออกไปจากคิวแบบ FIFO ซึ่งมันมีจุดประสงค์คือ

- ตรวจจับความหนาแน่นในระบบเครือข่ายเบื้องต้น
- อนุญาตให้มีความหนาแน่นของปริมาณทราฟฟิกสูงๆ ได้ชั่วคราวโดยใช้หลักการดูดซับ (Absorbing)ความหนาแน่นเอาไว้

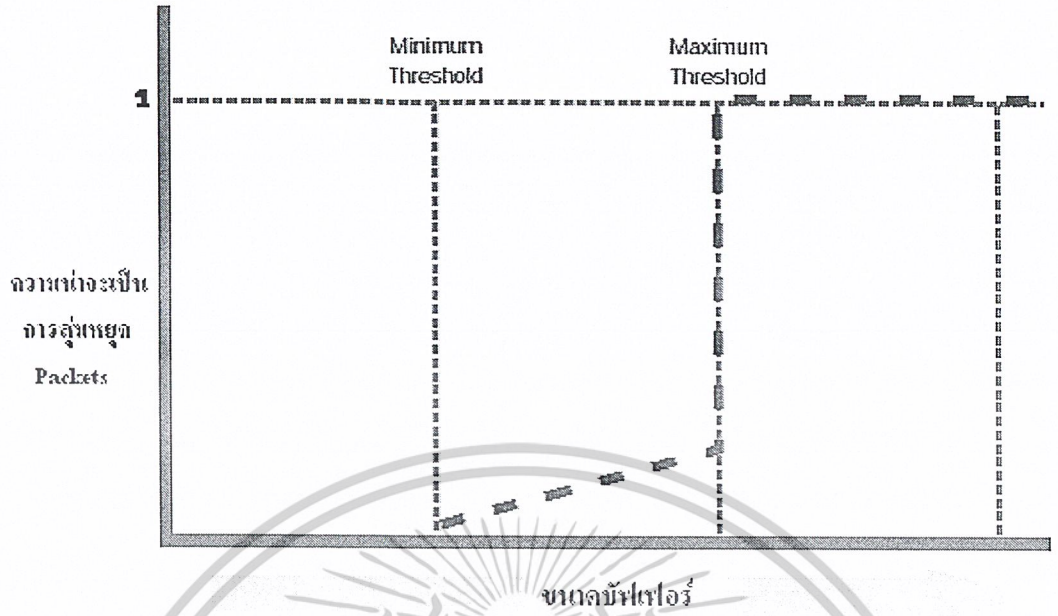
- ป้องกันการเกิดความหนาแน่นเพิ่มมาอีก
- หลีกเลี่ยงการเกิดความหนาแน่นขึ้นพร้อมๆกันของการเชื่อมต่อต่างๆ โดยใช้การเลือกสุ่มการเชื่อมต่อหรือการไหลของ Packets ที่ถูกทำเครื่องหมายไว้

เป้าหมายของ RED ที่จะทำตามดังที่กล่าวก็คือ มันจะมีการทำเครื่องหมายหรือสัญลักษณ์ลงบนความยาวเฉลี่ยของคิวกับ จุดจำกัด ต่ำสุดและสูงสุด (Minimum & Maximum Threshold) โดยค่าของความยาวเฉลี่ยของใน RED คิวนี้ จะใช้การคำนวณจากทุกช่วงเวลาที่ มี Packets เข้าออกจากคิวนั่นเอง ซึ่งก็จะทำให้ได้ค่าของความยาวเฉลี่ยของคิว



รูปที่ 3.11 การหาความยาวเฉลี่ยของคิว

ซึ่งถ้าความยาวเฉลี่ยของคิวมีค่าต่ำกว่าจุดจำกัดต่ำสุด Packets ต่างๆก็จะไหลเข้าไปในคิวโดยที่จะยังไม่มีการเครื่องหมายลงบน Packets เหล่านี้ แต่เมื่อมีความเป็นไปได้ในการที่จะมีปริมาณของ Packets เพิ่มสูงขึ้นจนทำให้ค่าความยาวเฉลี่ยของคิวเพิ่มสูงขึ้นจนเกินระดับจุดจำกัดต่ำสุด ก็จะมีการทำเครื่องหมายไว้บน Packets บางส่วน และจะ หยุด Packets บางส่วนเหล่านี้ไว้เพื่อที่จะทำให้ยังสามารถรองรับปริมาณทราฟฟิกภายในคิวนี้ได้หรืออาจเรียกการทำงานแบบนี้ว่าเป็นการดูดซับ (Absorbing) ความหนาแน่นก็ได้ แต่เมื่อมีจำนวนของ Packets ที่ไหลเข้ามาสูงมากจนทำให้ความยาวเฉลี่ยของคิวนี้สูงเกินจุดจำกัดสูงสุด Packets ทั้งหมดก็จะถูกหยุดไว้และระบบก็จะไม่สามารถใช้งานได้ซึ่งสามารถแสดง ได้ดังรูปกราฟที่ 3.12 ต่อไปนี้



รูปที่ 3.12 กราฟแสดงการทำงานของ RED

ซึ่งวิธีการทำงานของ RED นั้นสามารถแบ่งได้เป็น 2 ขั้นตอนก็คือ

อันดับแรก มันจะคำนวณหาค่าของความยาวเฉลี่ยของคิวจากจุดจำกัด ต่ำสุดและสูงสุดนี้ในทันทีที่เริ่มทำงานซึ่งจะขึ้นอยู่กับการตั้งค่าของอุปกรณ์อย่างเกตเวย์เพื่อรองรับทราฟฟิกเป็นต้น

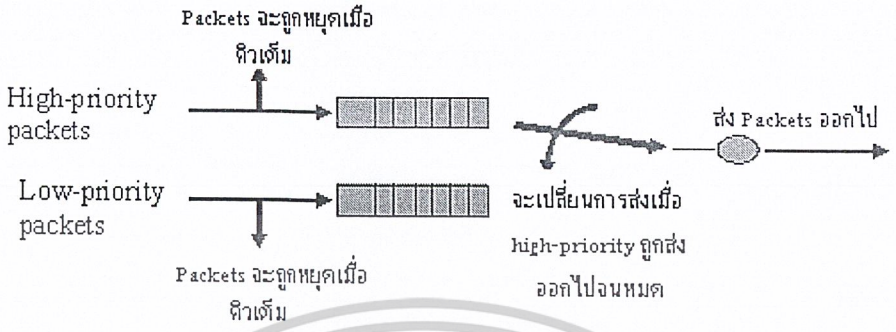
อันดับสอง เป็นวิธีการที่จะเลือกสุ่มทำเครื่องหมาย Packets ที่จะทำการหยุด ซึ่งจะให้มีความถี่เท่าใดที่จะทำการหยุด Packets ซึ่งจากกราฟที่ 3.12 นั้นจำนวนของ Packets ที่จะถูกหยุดนั้นจะสามารถเขียนได้เป็นความน่าจะเป็นที่จะหยุดจำนวนของ Packets เท่าใดจากจำนวน Packets ทั้งหมด (Discard Probability)

ทั้งนี้ถ้าความหนาแน่นของปริมาณทราฟฟิกยังคงมีอยู่ต่อไป RED ก็จะทำการสุ่มเพื่อทำการหยุด Packets เพิ่มขึ้นเรื่อยๆ จนกว่าจะมีความหนาแน่นที่ลดลง ซึ่งความถี่ในการที่จะสุ่มทำเครื่องหมายกับ Packets ต่างๆที่เพิ่มขึ้นอยู่กับ ความยาวเฉลี่ยของคิวที่จะเข้าถึงจุดจำกัดสูงสุดนั่นเอง

3.7 Priority Queuing

Priority Queuing เป็นวิธีในเพิ่มความสามารถในการจัดการทราฟฟิกใน Router ที่นิยมใช้งานกันอีกวิธีหนึ่ง ซึ่งในการทำงานของมันนั้นจะทำการจัดเรียง Priority ของ Packets ในระดับต่างๆกันและแยก Packets เหล่านั้นเรียงตามลำดับของ Priority ลงในคิวเดียว โดยในการแบ่งลำดับชั้น Priority ของ Packets ต่างๆเหล่านี้นั้นจะขึ้นอยู่กับ ชนิดข้อมูลมาแล้วปลายทางที่จะส่งไปอย่างเช่น ชนิดของโปรโตคอล,หมายเลขของพอร์ตปลายทาง หรืออาจจะขึ้นอยู่กับปัจจัยอื่นๆก็ได้ วิธีในการส่ง Packets ออกจากคิวนั้นจะส่ง

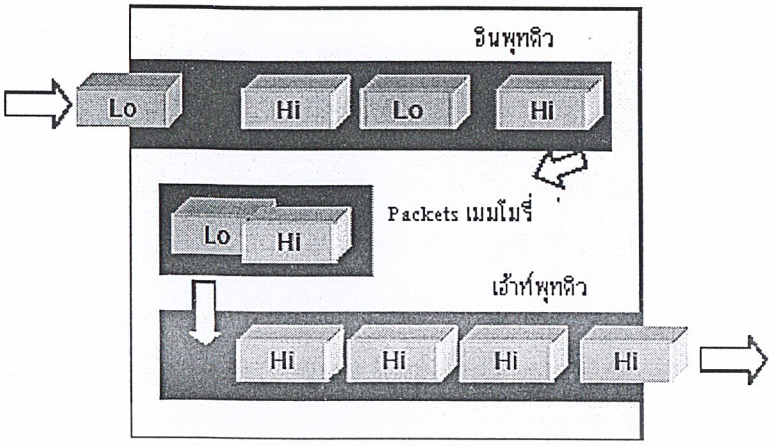
Packets ที่มีค่า Priority สูงที่สุดเป็นอันดับแรก และเมื่อส่งหมดไปแล้วนั้น Packets ที่มี Priority รองลงมาจึงจะได้รับการบริการถัดไป โดยมีการทำงานคร่าวเขียนได้ดังรูปที่ 3.13



รูปที่ 3.13 แสดงการเลือกส่ง Packets ที่มี Priority ต่างกัน

ภายในจุดเชื่อมต่อนั้นถ้ามีทราฟฟิกที่มี Priority สูงๆแล้ว Packets ที่มี Priority ต่ำกว่านั้นจะถูกสกัดกั้นการไหลเอาไว้ซึ่งไม่สามารถที่จะนำออกไปจากคิวได้ ซึ่งเป็นปัญหาที่สำคัญมาก เพราะมีบางการใช้งานที่มีความต้องการด้วยเหมือนกันแม้ว่าจะไม่มากก็ตาม ซึ่งการไหลข้อมูลที่เป็น TCP นั้นส่วนใหญ่จะทำให้เกิดปัญหานี้เกือบทั้งหมด เพราะในระบบของ Priority Queuing นี้จะให้ความสำคัญกับข้อมูลที่เป็น TCP packets ในการทำงานในชั้น Transport Layer มากกว่าข้อมูลที่เป็นอย่างอื่นทั้งหมด หรือนั่นคือข้อมูลประเภท TCP Packets จะมี Priority สูงที่สุดนั่นเอง นั่นหมายความว่ามีกรณีน้อยมากที่ข้อมูลที่มี Priority ต่ำทำให้เกิดความหนาแน่นของปริมาณทราฟฟิกที่จุดเชื่อมต่อ

และปัญหาที่ควรให้ความใส่ใจก็คือผลกระทบของ FIFO Queuing ก็จะยังคงเกิดอยู่กับ การใช้งาน Priority Queuing เช่นกัน เพราะเมื่อมีปริมาณของข้อมูลที่มี Priority ระดับเดียวกันเข้ามาในคิวจำนวนมากนั้น การทำงานก็จะเหมือน FIFO Queuing ทันที เพราะฉะนั้นผลกระทบที่มาจาก FIFO ก็มีผลเหมือนกันกับ Priority Queuing อย่างเช่น ถ้ามีข้อมูลใน Priority เดียวเข้ามาในคิว ข้อมูลที่เข้ามาก่อน ก็จะได้รับการบริการหรือส่งออกไปก่อน และเมื่อมีปริมาณข้อมูลที่เข้ามาสูงจนคิวเต็ม ข้อมูลที่เหลือก็จะไม่สามารถเข้ามาในคิวได้อีกเหมือนกับ FIFO Queuing



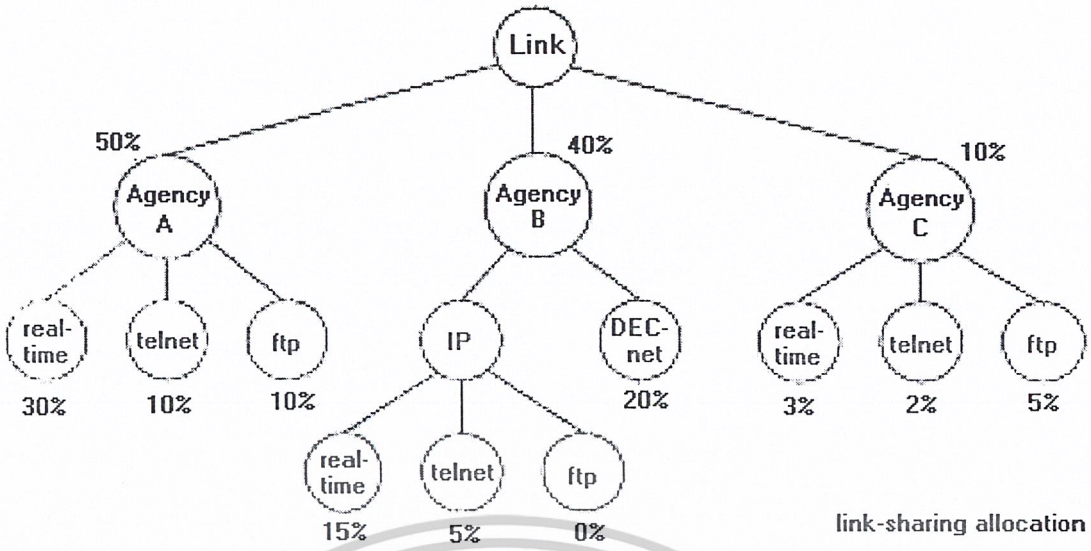
รูปที่ 3.14 การจัดส่งข้อมูลที่มี Priority ต่างกันออกจากคิว

ซึ่งลักษณะของ Priority Queuing ก็คือ

- Packets ของข้อมูลที่มี Priority สูงจะถูกนำออกจากคิวก่อนข้อมูลธรรมดา
- ในกรณีที่มีข้อมูลหลากหลายนั้นจะเลือกส่งข้อมูลที่มี Priority แน่นอนกว่าอย่างเช่น เลือกส่ง IP ก่อน SNA, TCP ก่อน UDP, TELNET ก่อน FTP
- ในการส่งข้อมูลของแต่ละ Priority จะใช้เวลานานมาก
- การไหลของปริมาณทราฟฟิกเกือบทั้งหมดที่เกิดจากข้อมูลที่มี Priority สูงนั้นจะทำให้เกิดปัญหาในการบริการข้อมูลที่มี Priority ต่ำ ซึ่งจะทำให้มีการหยุดของข้อมูลเหล่านี้
- มีความล่าช้า (delay) สูงสำหรับข้อมูลที่มี Priority ต่ำ เพื่อรอคอยที่จะได้รับการบริการการถูกนำส่งออกจากคิว

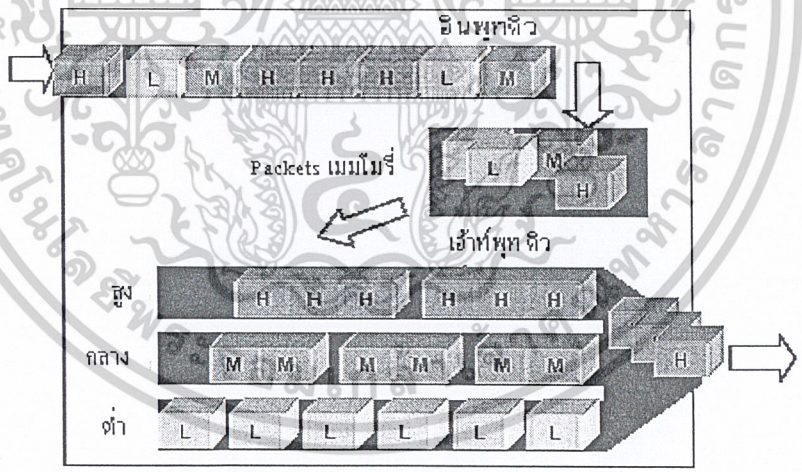
3.8 Class-Based Queuing (CBQ)

Class-Based Queuing (CBQ) นี้เริ่มแรกใช้เพื่อความต้องการในการที่จะควบคุมการทำงานของจุดใช้งานร่วม (Link-Sharing) ระหว่างผู้ใช้งาน, องค์การที่แตกต่างกัน หรือระหว่างการใช้งานที่ต้องมีการรับประกันว่าจะได้รับการใช้งานช่องสัญญาณอย่างแน่นอนในแต่ละการทำงาน หรือการใช้งานที่มีความพิเศษต่างๆ ซึ่งวิธีการควบคุมจุดใช้งานร่วมนี้จะแสดงในรูปของการเชื่อมต่อแบบต่างๆในการเชื่อมต่อสื่อสารที่อยู่ในส่วนบนของโครงข่ายดังรูปที่ 3.15 โดยช่องสัญญาณทั้งหมดจะมีการถูกแบ่งปันระหว่างกันและแต่ละกลุ่มจะมีการแบ่งขนาดของช่องสัญญาณที่น้อยที่สุดที่จะมีได้ของตัวเองซึ่งในลำดับชั้นซึ่งมีการแบ่งย่อยลำดับชั้นมากก็จะได้รับการรับประกันการมีช่องสัญญาณในเปอร์เซ็นต์ที่ต่ำลงไป



รูปที่ 3.15 การแบ่งลำดับชั้นของการใช้งานช่องสัญญาณในโครงข่าย

ภายใน CBQ นี้มีการใช้งานที่มีสิทธิพิเศษร่วมกันนั้นจะกำหนดได้โดย เมื่อมีการไหลของทราฟฟิกที่สูงขึ้นภายในระบบเครือข่ายนั้นก็จะมีกฎเกณฑ์ที่จะอนุญาตให้ใช้งานช่องสัญญาณที่ไม่ได้ใช้งานได้ชั่วคราว ซึ่งภายใน CBQ นี้มีความสามารถในการที่จะควบคุม การแบ่งแยกชนิดของ Packets ออกเป็นลำดับชั้น, กำหนดแบบแผนโครงสร้างของ Packets และการจัดการคิวภายในระบบได้



รูปที่ 3.16 การใช้ CBQ กับ Packets ที่มีการกำหนด Priority หลายระดับ

ซึ่งลักษณะของ CBQ ก็คือ

- สามารถประยุกต์ใช้งานได้ดีกว่า Priority Queue
- ซึ่งในกรณีที่ Packets มี Priority หลายระดับในคิวซึ่ง ตัวที่มี Priority สูงกว่าจะถูกระบายออกไปได้เร็วกว่าตัวที่มี Priority ต่ำกว่า (แต่จะได้รับการบริการแน่นอน)

- Router จะมีการให้บริการที่แตกต่างกันในแต่ละลำดับชั้น ของปริมาณทราฟฟิกดังนั้น วิธีการนี้จึงเรียกว่า “Class-Based Queuing” หรือเรียกว่า “Custom Queuing”
- มีความเป็นธรรม (Fairness) ในการแบ่งช่องสัญญาณเพื่อใช้ในงานต่างๆ

บทที่ 4

การออกแบบและทดลอง

4.1 ขั้นตอนการทำงานของ RED

ในการทดลองนี้เราได้เลือกใช้วิธีการจัดการคิวจากที่ได้เสนอไปในบทที่ 3 คือวิธีแบบ Random Early Detection (RED) ซึ่งเป็นวิธีการที่นิยมใช้งานจริงในปัจจุบัน และมีประสิทธิภาพสูง ซึ่งตามทฤษฎีที่ได้นำเสนอไปในบทที่ 3 นั้นเราสามารถเขียนเป็นอัลกอริทึมได้ดังรูปที่ 4.1

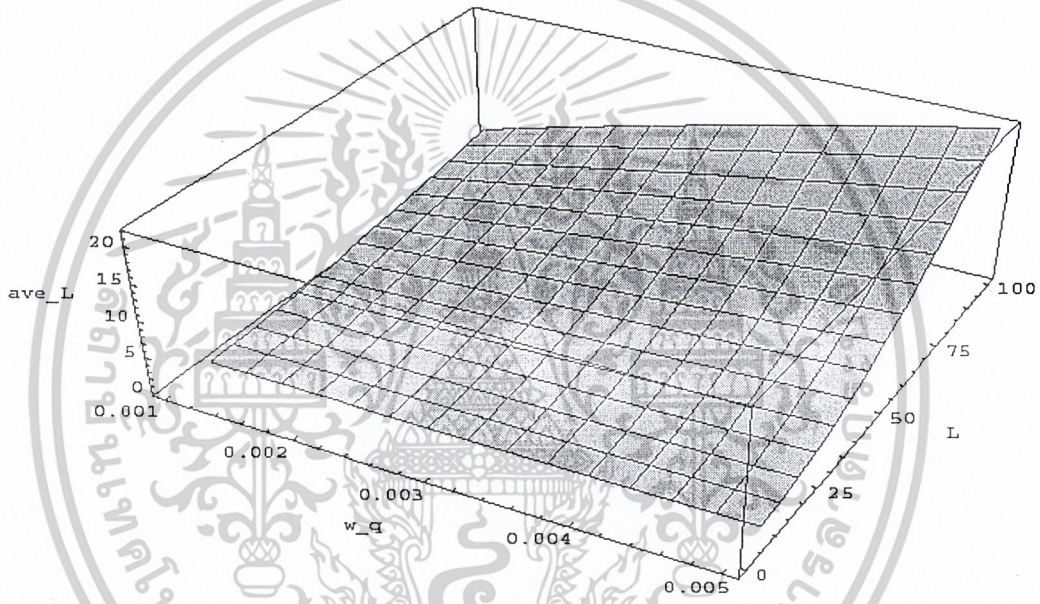
```
Initialization:
    avg ← 0
    count ← −1
for each packet arrival
    calculate the new average queue size avg:
        if the queue is nonempty
            avg ← (1 − w_q)avg + w_q q
        else
            m ← f(time − q_time)
            avg ← (1 − w_q)^m avg
    if min_th ≤ avg < max_th
        increment count
        calculate probability p_a:
            p_b ← max_p(avg − min_th)/(max_th − min_th)
            p_a ← p_b/(1 − count · p_b)
        with probability p_a:
            mark the arriving packet
            count ← 0
    else if max_th ≤ avg
        mark the arriving packet
        count ← 0
    else count ← −1
when queue becomes empty
    q_time ← time
```

รูปที่ 4.1 อัลกอริทึมการทำงานของ RED

ซึ่งจากรูปที่ 4.1 นี้จะเป็นในรายละเอียดเมื่อมี packets ไหลเข้าใน RED Router โดยจะทำการหาค่าเฉลี่ยของความยาวคิว (avg) ตลอดเวลาเมื่อมี packets เข้ามาซึ่งในส่วนของการออกแบบและพัฒนานั้นเราจะแบ่งส่วนในการพิจารณาออกเป็น 2 ส่วน คือในส่วนของการออกแบบจุดจำกัดสูงสุดและต่ำสุด (Maximun & Minimum Threshold : $\max_{th}, \max_{th}$) และส่วนของการคำนวณหาค่าสุ่มในการหยุด packets (p_b) ซึ่งจะนำเสนอในส่วนถัดไป

4.2 การออกแบบค่าพารามิเตอร์แบบต่างๆ

ในวิธีการจัดการคิวของ RED นี้ จะมีการกำหนดค่าน้ำหนักของคิว (queue weight : w_q) ซึ่งจะใช้วิธีการเปรียบเสมือนวงจรกรองความถี่ต่ำ (Low Pass Filter) ซึ่งจะใช้วิธีการตรวจจับความหนาแน่นของปริมาณทราฟฟิกซึ่งค่า w_q นี้จะเสมือนกับจุดคัทออฟ (Cut-Off) ซึ่งในการกำหนดค่า w_q นี้ถ้ามีการกำหนดให้มากจนเกินไปก็จะทำให้เมื่อมีปริมาณของทราฟฟิกใน Router สูงก็จะทำให้ไม่มีการสุ่มหยุด packets ซึ่งจะทำให้ Router กระทำตัวเองเหมือนกับ FIFO ที่เมื่อภายในคิวเต็มก็จะหยุดทุก packets ที่เข้ามา แต่ถ้ากำหนดค่า w_q ต่ำจนเกินไปก็จะมีการสุ่มหยุด packets มากจนเกินไปจนอาจทำให้มีการหยุด packets ของข้อมูลที่สำคัญๆได้ซึ่งการออกแบบค่า w_q นี้จะมีความสัมพันธ์กับค่าขนาดของบัฟเฟอร์ที่เก็บ packets (L) โดยจะทำให้ค่าของค่าเฉลี่ยของความยาวคิว (avg_L) เปลี่ยนแปลงไป ดังจากรูปที่ 4..2



รูปที่ 4.2 แสดงความสัมพันธ์ของค่า w_q , L, avg_L

โดยจากรูปที่ 4.2 นั้นค่า avg_L จะเปลี่ยนแปลงไปตามสมการต่อไปนี้

$$avg_L = \sum_{i=1}^{L} i w_q (1-w_q)^{L-i}$$

$$avg_L = w_q (1-w_q)^L \sum_{i=1}^{L} i \left(\frac{1}{1-w_q}\right)^i$$

$$\boxed{avg_L = L + 1 + \frac{(1-w_q)^{L+1} - 1}{w_q}} \qquad \text{---------- (1)}$$

ซึ่งจากสมการของ avg_L นี้จะแตกต่างกับค่า avg ซึ่งค่า avg_L นั้นจะคิดในกรณีที่มี packets ไหลเข้ามาในบัฟเฟอร์คิวจนเต็ม L packets แต่ค่า avg นั้นจะเปลี่ยนแปลงไปตามสภาวะของคิวในขณะเวลานั้นที่มี packets เข้ามาและออกไปจากคิวดังสมการที่ (2)

$$avg = (1 - w_q)avg + w_q q \qquad \text{---------- (2)}$$

และเมื่อพิจารณาที่สมการที่ (1) นั้นเราจะคิดในกรณีที่มี packets เข้ามาเต็มคิว นั่นคือถ้าต้องการออกแบบ RED Router ให้มีประสิทธิภาพสูงที่สุดนั่นหมายความว่าค่า avg_L จะต้องมีค่าที่น้อยกว่าจุดจำกัดต่ำสุด ($\min_{th}$) ที่จะทำให้ packets ไม่มีการถูกหยุดดังนั้นเราจะได้เงื่อนไขเป็น

$$L + 1 + \frac{(1 - w_q)^{L+1} - 1}{w_q} < \min_{th}$$

หรือ

$$\boxed{avg_L < \min_{th}}$$

ซึ่งในส่วนของการกำหนดจุดจำกัดสูงสุด($\max_{th}$)นั้นจากเอกสารอ้างอิงที่ [19] ได้แนะนำว่าควรจะใช้ค่า $\max_{th}$ มากกว่าอย่างน้อยที่สุดเป็นสองเท่าของ $\min_{th}$ ซึ่งค่าที่ใช้งานจริงนั้นจะเป็น

$$\boxed{\max_{th} < 3\min_{th}}$$

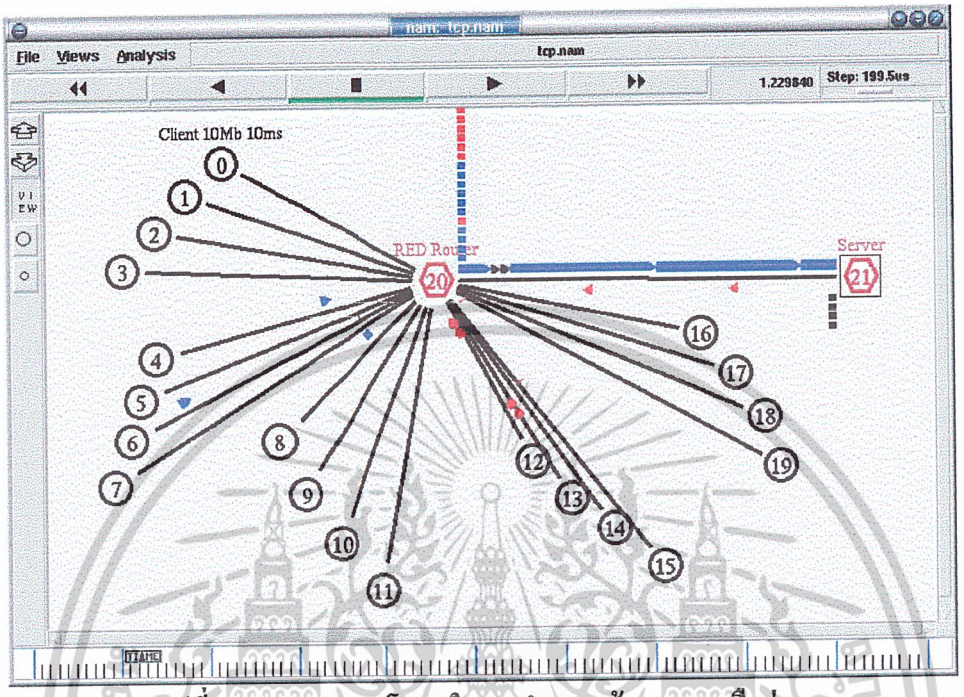
และในส่วนที่เราจะพิจารณาเพิ่มเติมนั้นเป็นการกำหนดค่าการสุ่มหยุด packets (p_b) ซึ่งจะเป็นไปตามสมการที่ (3)

$$p_b = \max_p \frac{(avg - \min_{th})}{(\max_{th} - \min_{th})} \qquad \text{---------- (3)}$$

โดยที่ค่า $\max_p$ นั้นเป็นค่าความน่าจะเป็นในการสุ่มหยุด packets สูงที่สุดโดยค่าทั่วไปที่เรากำหนดให้ค่า $\max_p$ นั้นจะเป็น 0.1 นั่นเอง

4.3 การทดลอง

เราจะใช้โปรแกรม Network Simulator (NS) ในการจำลองการทำงานของ RED Router ซึ่งจะใช้การวางโหนด (Node) แทนการใช้งานของเครื่องลูกข่ายซึ่งสามารถแสดงได้ตามรูปที่ 4.3



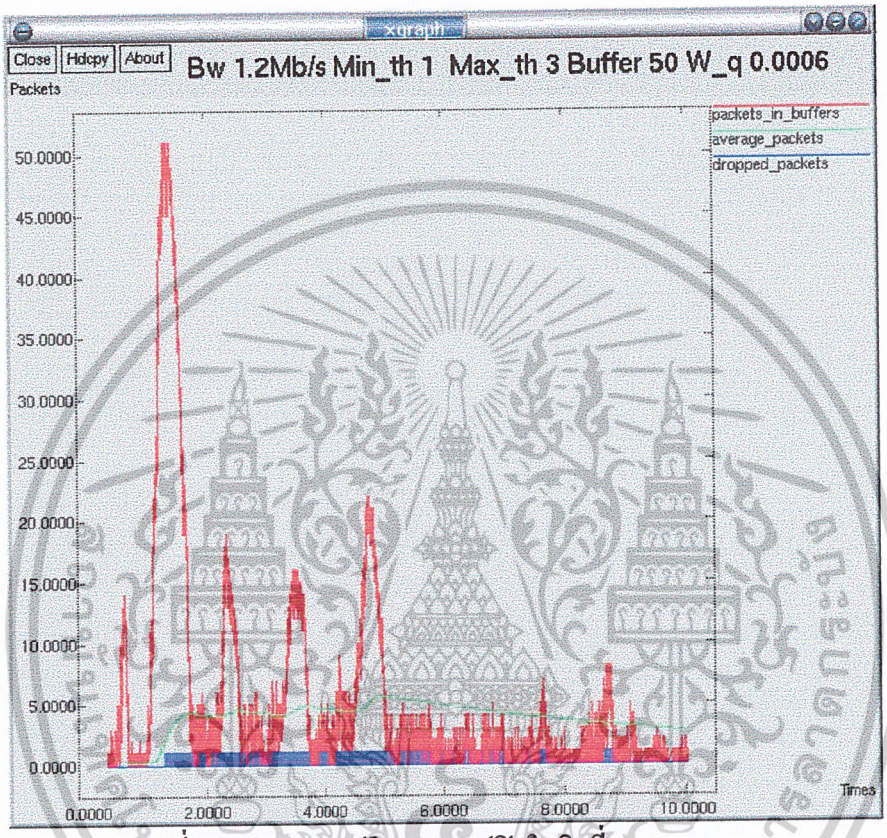
รูปที่ 4.3 แสดงการวางโหนดในการจำลองสร้างระบบเครือข่าย

จากรูปที่ 4.3 เราจะใช้จำนวนของเครื่องลูกข่ายเป็น 20 โหนด โดยแต่ละโหนดนั้นจะกำหนดให้มีการส่งปริมาณ packets ไปยังโหนดปลายทาง (โหนดที่ 21) ผ่านทาง Router (โหนดที่ 20) ในแบบสุ่ม ซึ่งในการทดลองนี้เราจะทดลองกำหนดค่าพารามิเตอร์ (Parameter) ที่จะเป็นตัวกำหนดประสิทธิภาพการจัดการคิวแบบ RED นี้ซึ่งมีพารามิเตอร์คือ ค่าของ w_q โดยเราจะทำการปรับเปลี่ยนค่าของ w_q โดยเหตุที่เลือกปรับค่าพารามิเตอร์ตัวนี้เพราะ w_q จะเป็นตัวกำหนดขนาดของ $\min_{th}$ และ $\max_{th}$ ที่จะทำการสุ่มหยุด packets ใน Router ซึ่งในความจริงแล้วนั้นเรามีขนาดของบัฟเฟอร์ (L) คงที่จึงขึ้นอยู่กับการปรับค่าพารามิเตอร์ของ Router นั่นเองที่จะทำให้ความสามารถในการทำงานมีประสิทธิภาพซึ่งค่าพารามิเตอร์ที่เราจะทำการกำหนดให้คงที่นั้นมีดังนี้

1. ขนาดของบัฟเฟอร์ $L = 50$ packets
2. ความน่าจะเป็นสูงสุดในการสุ่มหยุด packets $\max_p = 0.1$

โดยแต่ละค่าของ w_q ที่เปลี่ยนไปเราจะทำการเลือกแสดงกราฟปริมาณของ packets ในคิวและ packets ในส่วนที่ถูกหยุดไว้และ กราฟแสดงค่า Throughput ต่อช่องสัญญาณ (Bandwidth) รวมด้วย ดังแสดงจากรูปต่อไปนี้

ที่ Wq = 0.0006

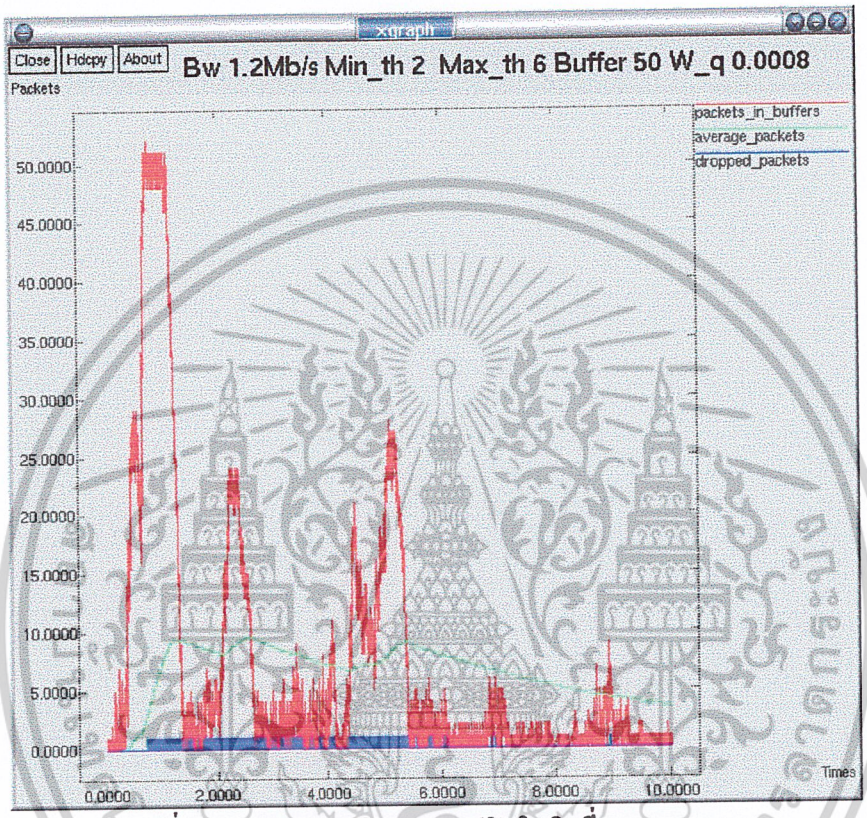


รูปที่ 4.4 กราฟแสดงปริมาณทราฟฟิกในคิวที่ $w_q = 0.0006$

Bandwidth (Mbps)	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
Packets (Mbps)	0.176	0.358	0.508	0.656	0.706	0.840	0.941	0.976	0.947	0.958

ตารางที่ 4.1 แสดง packets ที่รับได้ที่โหนดปลายทางต่อช่องสัญญาณค่าต่างๆที่ค่า w_q =0.0006

ที่ Wq = 0.0008

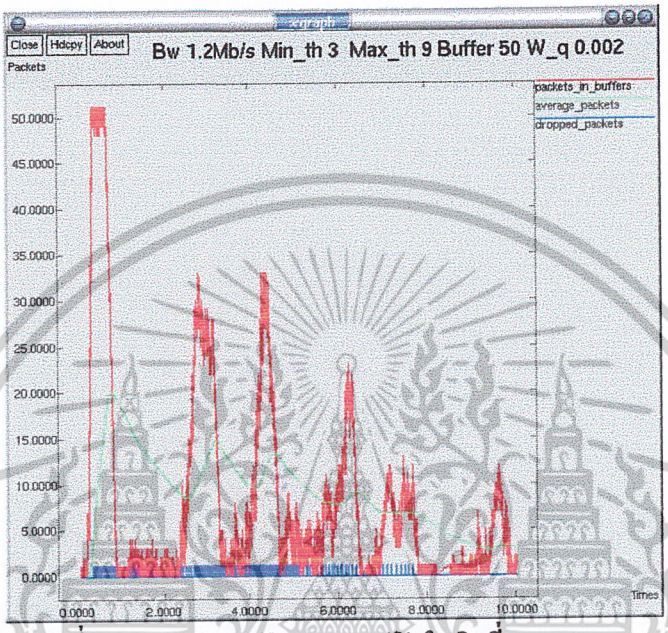


รูปที่ 4.5 กราฟแสดงปริมาณทราฟฟิกในคิวที่ $w_q = 0.0008$

Bandwidth (Mbps)	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
Packets (Mbps)	0.167	0.334	0.547	0.697	0.755	0.856	0.926	0.958	0.993	0.997

ตารางที่ 4.2 แสดง packets ที่รับได้ที่โหนดปลายทางต่อช่องสัญญาณค่าต่างๆที่ค่า w_q =0.0008

ที่ Wq = 0.002

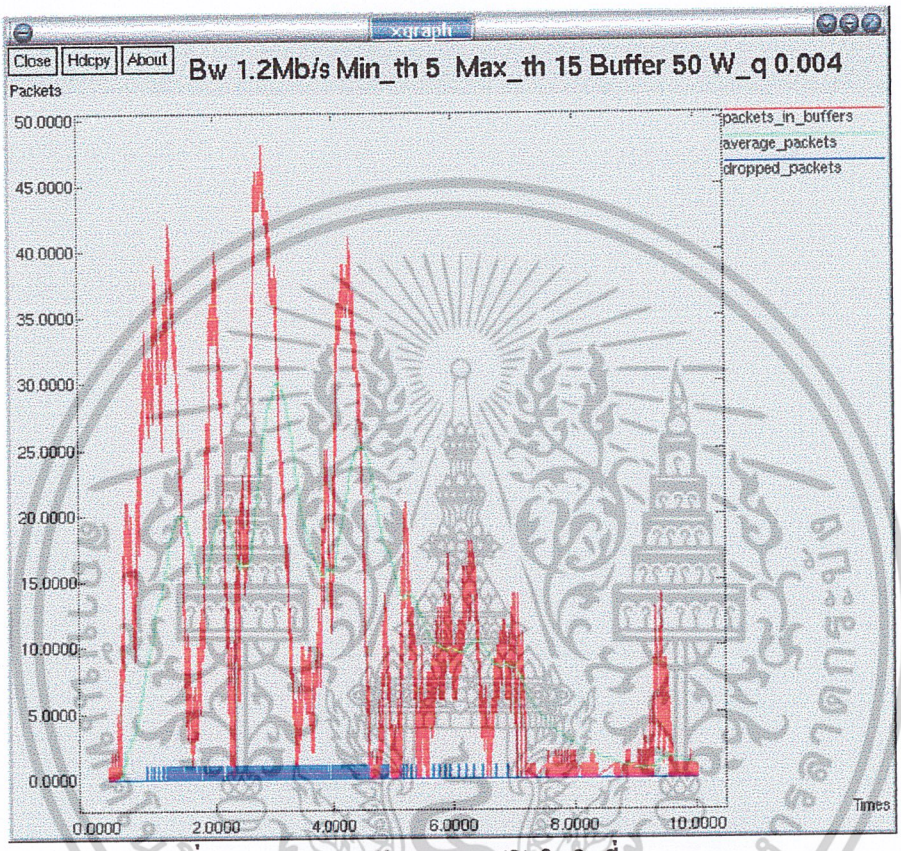


รูปที่ 4.6 กราฟแสดงปริมาณทราฟฟิกในคิวที่ w_q = 0.002

Bandwidth (Mbps)	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
Packets (Mbps)	0.169	0.365	0.574	0.707	0.837	0.893	0.924	0.937	0.994	0.997

ตารางที่ 4.3 แสดง packets ที่รับได้ที่โหนดปลายทางต่อช่องสัญญาณค่าต่างๆที่ค่า w_q =0.002

ที่ Wq = 0.004

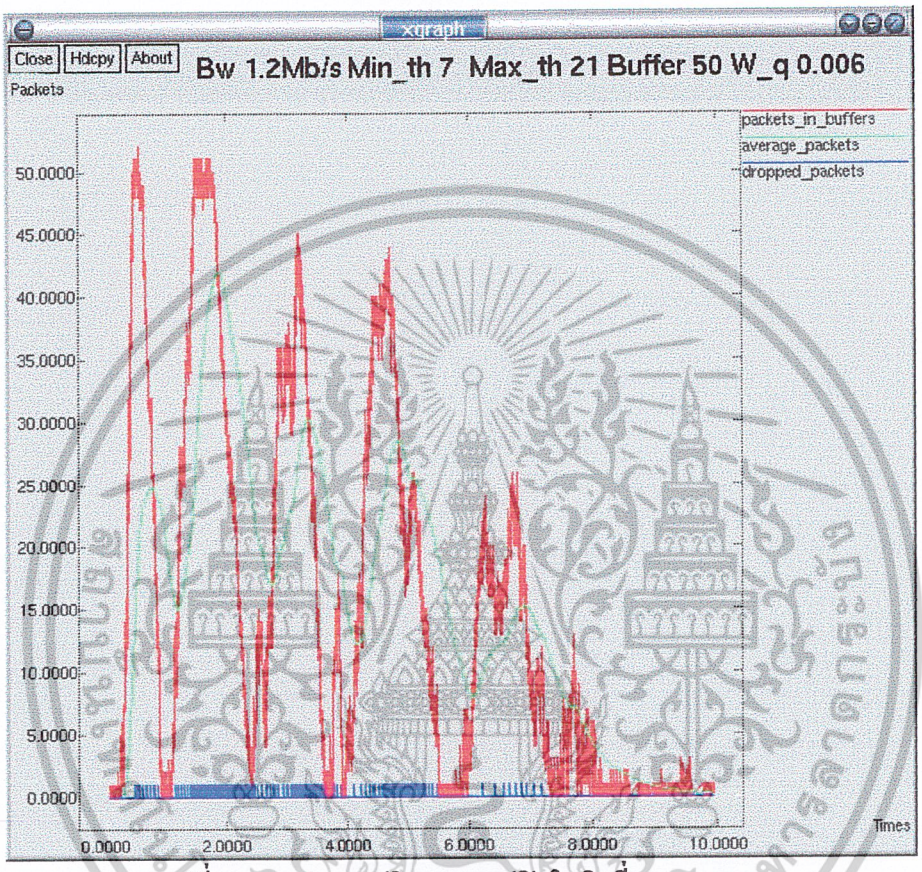


รูปที่ 4.7 กราฟแสดงปริมาณทราฟฟิกในคิวที่ w_q = 0.004

Bandwidth (Mbps)	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
Packets (Mbps)	0.177	0.388	0.567	0.706	0.798	0.828	0.895	0.945	0.970	0.994

ตารางที่ 4.4 แสดง packets ที่รับได้ที่โหนดปลายทางต่อช่องสัญญาณค่าต่างๆที่ค่า w_q =0.004

ที่ Wq = 0.006

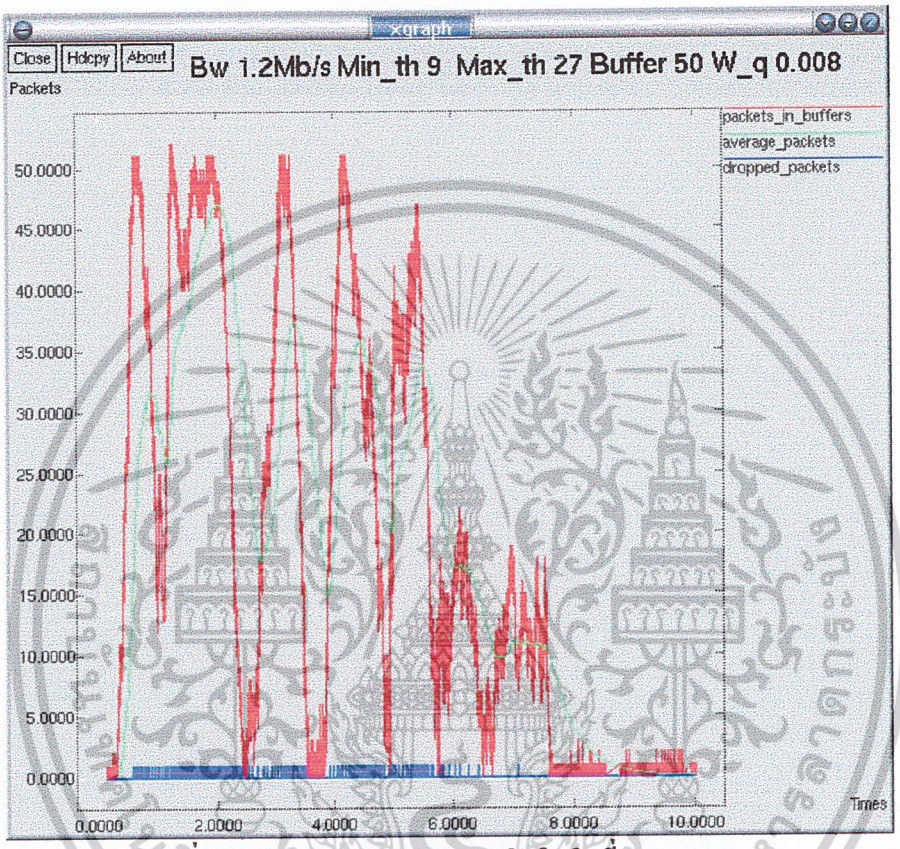


รูปที่ 4.8 กราฟแสดงปริมาณทราฟฟิกในคิวที่ w_q = 0.006

Bandwidth (Mbps)	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
Packets (Mbps)	0.192	0.385	0.583	0.706	0.897	0.888	0.954	0.951	0.946	0.958

ตารางที่ 4.5 แสดง packets ที่รับได้ที่โหนดปลายทางต่อช่องสัญญาณค่าต่างๆที่ค่า w_q =0.006

ที่ Wq = 0.008

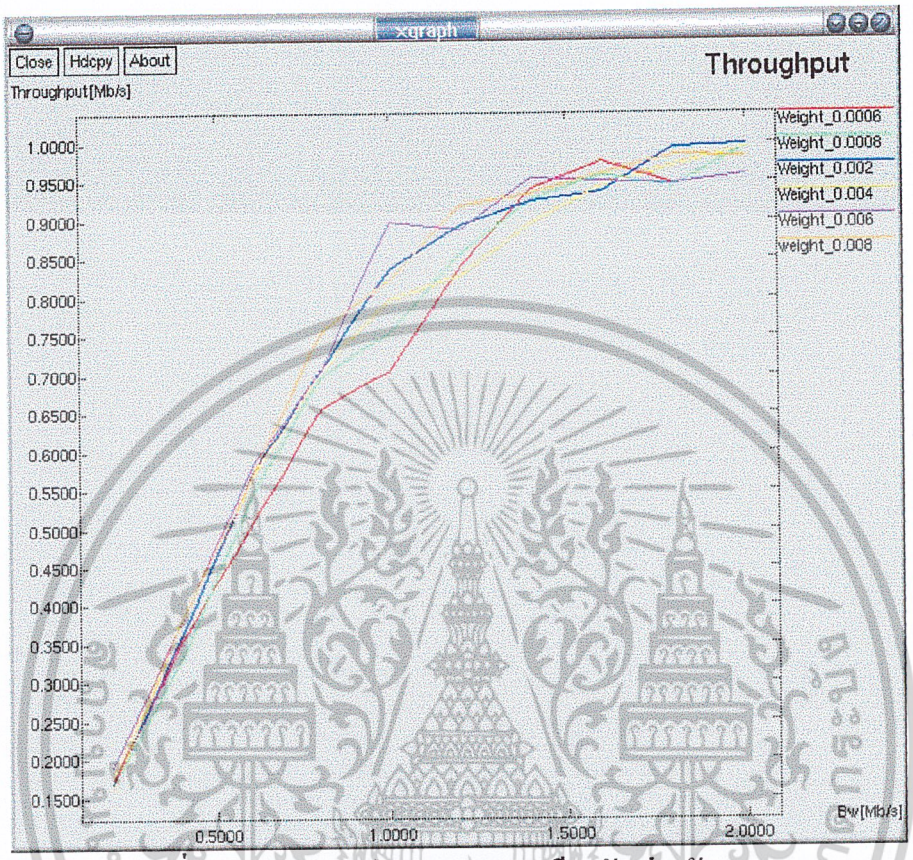


รูปที่ 4.9 กราฟแสดงปริมาณทราฟฟิกในคิวที่ w_q = 0.008

Bandwidth (Mbps)	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
Packets (Mbps)	0.194	0.388	0.570	0.756	0.822	0.918	0.934	0.961	0.983	0.981

ตารางที่ 4.6 แสดง packets ที่รับได้ที่โหนดปลายทางต่อช่องสัญญาณค่าต่างๆที่ค่า w_q =0.008

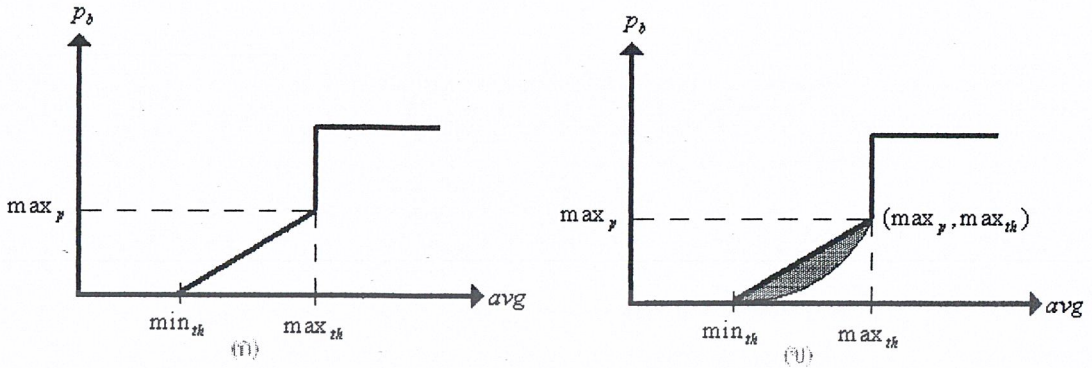
โดยจากตารางที่ 4.1 - 4.6 นั้นจะเป็นการเก็บค่าของ packets ที่สามารถรับได้จริงจากจำนวนของช่องสัญญาณที่ค่าต่างๆซึ่งสามารถแสดงได้ดังรูปที่ 4.10



รูปที่ 4.10 กราฟแสดงค่า Throughput เทียบกับ ช่องสัญญาณ

จะเห็นได้ว่าเมื่อเพิ่มค่าของ w_q จะทำให้เพิ่มค่าของ Throughput ขึ้นเช่นกันแต่จากการพิจารณาจากรูปของกราฟ 4.4 – 4.9 นั้นก็จะพบว่า การที่เพิ่มค่า w_q แล้วทำให้ค่า Throughput เพิ่มนั้นก็เป็นเพราะว่า packets นั้นจะมีการถูกสุ่มหยุด packets ที่น้อยลงทำให้มีการสูญเสียที่ปลายทางนั้นลดน้อยลงไป แต่การที่เราใช้ค่า w_q ที่สูงไปนั้นก็จะทำภายใน Router มีความหนาแน่นของปริมาณทราฟฟิกได้สูงมากเช่นกัน และอาจจะประพฤติตัวเป็นการทำงานแบบ FIFO ก็ได้

4.4 การพัฒนา



รูปที่ 4.11 ความแตกต่างระหว่างวิธีการปกติกับวิธีการที่พัฒนาขึ้น

การพัฒนานี้เป็นการประยุกต์จากค่าการสุ่มหยุด packets (p_b) ซึ่งจากสมการเส้นตรงที่ว่า

$$\frac{p_b - 0}{avg - \min_{th}} = \frac{\max_p - 0}{\max_{th} - \min_{th}}$$

$$p_b = \max_p \frac{(avg - \min_{th})}{(\max_{th} - \min_{th})}$$

ซึ่งจะได้ค่าของสมการการสุ่มหยุด packets (p_b) เหมือนกับสมการที่ (3) นั่นเอง

และในการพัฒนาขึ้นใหม่นี้เราจะแสดงได้จากรูปที่ 4.11 (ข) โดยที่เราสามารถจะลดปริมาณการสุ่มหยุด packets ได้ตามขนาดจำนวนที่เป็นการแรงเงา โดยการหาค่าของสมการ p_b ใหม่นี้จะอ้างอิงจากสมการพาราโบลาที่ว่า

$$(avg - \min_{th})^2 = 4C(p_b - 0)$$

$$p_b = \frac{(avg - \min_{th})^2}{4C} \qquad \text{---------- (4)}$$

จากสมการที่ (4) เมื่อสมการพาราโบลาของเรานี้ผ่านจุด $(\max_p, \max_{th})$ เราจะหาค่าคงที่ 4C ที่เหลือได้โดยการแทนจุดนี้ลงไปจะได้

$$\max_p = \frac{(\max_{th} - \min_{th})^2}{4C}$$

$$4C = \frac{(\max_{th} - \min_{th})^2}{\max_p} \qquad \text{---------- (5)}$$

นำสมการที่ (5) แทนลงไปในสมการที่ (4) จะได้สมการการสุ่มหยุด packet (p_b) เป็น

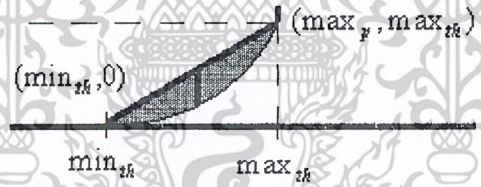
$$p_b = \max_p \frac{(avg - \min_{th})^2}{(\max_{th} - \min_{th})^2}$$

$$\boxed{p_b = \max_p \left[\frac{avg - \min_{th}}{\max_{th} - \min_{th}} \right]^2} \qquad \text{---------- (6)}$$

โดยเมื่อเรากำหนดให้ $\max_{th} = 3\min_{th}$ จะได้สมการใหม่เป็น

$$p_b = \frac{\max_p}{4} \left[\frac{avg - \min_{th}}{\min_{th}} \right]^2 \qquad \text{---------- (7)}$$

ซึ่งสมการใหม่นี้คือสมการที่สามารถจะลดปริมาณของ packets ที่ถูกหยุดไปจากสมการที่ (3) ได้ โดยเราที่สามารถจะหาจำนวนของ packets ที่ลดไปได้จากรูปที่ 4.11 (ข) ได้ดังรูปที่ 4.12



รูปที่ 4.12 การหาพื้นที่ที่สามารถลดจำนวน packets ได้จากสมการใหม่

จะทำการแก้สมการได้โดยให้ N เป็นจำนวนปริมาณของ packets ที่ลดได้จากสมการใหม่

$$N = \int_{\min_{th}}^{\max_{th}} \left[\left[avg \frac{\max_p}{2\min_{th}} - \frac{\max_p}{2} \right] - \left[\frac{\max_p}{4} \left(\frac{avg - \min}{\min} \right)^2 \right] \right] davg$$

$$N = \int_{\min_{th}}^{\max_{th}} \left[\left[avg \frac{\max_p}{2\min_{th}} - \frac{\max_p}{2} \right] - \left[\frac{\max_p}{4\min_{th}^2} (avg^2 - 2avg\min_{th} + \min_{th}^2) \right] \right] davg$$

$$N = \frac{\max_p}{2\min_{th}} \int_{\min_{th}}^{\max_{th}} [avg - 1] davg - \frac{\max_p}{4\min_{th}^2} \int_{\min_{th}}^{\max_{th}} [avg^2 - 2avg\min_{th} + \min_{th}^2] davg$$

จาก $\max_{th} = 3\min_{th}$ ดังนั้น $\max_{th} - \min_{th} = 2\min_{th}$

$$N = \frac{\max_p}{2\min_{th}}\left[2\min_{th}^2 - 2\min_{th}\right] - \frac{\max_p}{4\min_{th}^2}\left[\frac{8}{3}\min_{th}^3 - 4\min_{th}^3 + 2\min_{th}^3\right]$$

$$N = \max_p \min_{th} - \max_p - \frac{\max_p}{6}\min_{th}$$

$$N = \frac{5}{6}\max_p(5\min_{th} - 6)$$

ด้วยวิธีการที่ได้พัฒนาขึ้นจะทำให้สามรถลดจำนวนของ packets ที่จะถูกสุ่มหยุดจากความน่าจะเป็น (p_b) ได้ซึ่งจำนวนปริมาณ packets ที่ลดไปได้นี้จะมีปริมาณเท่ากับ

$$\boxed{N = \frac{5}{6}\max_p(5\min_{th} - 6)}$$

บทที่ 5
สรุปและวิจารณ์

5.1 สรุป

จากการศึกษาและค้นคว้าเกี่ยวกับการจัดการข้อมูลที่อยู่ในระบบเครือข่าย ทำให้ทราบว่าการรับส่งข้อมูลในลักษณะของ โปรโตคอล TCP/IP นั้น มีการรับส่งข้อมูลหลายประเภท ซึ่งมีการแบ่งชนิดการรับส่งข้อมูลชนิดต่างๆด้วยโปรโตคอลย่อยที่ทำงานภายใต้โปรโตคอล TCP/IP เช่น โปรโตคอล HTTP, โปรโตคอล FTP และโปรโตคอลอื่นๆ ซึ่งภายใต้ระบบเครือข่ายที่มีการใช้งานอย่างหนาแน่น แต่จำนวนช่องสัญญาณที่มีอยู่จำกัดจะทำให้เกิดความคับคั่งของข้อมูลภายในเครือข่ายอย่างมาก ทำให้เกิดจัดการความหนาแน่นของข้อมูล (Congestion-Management) โดยการจัดลำดับของข้อมูลใหม่โดยใช้หลักการจัดคิว (Queuing) โดยใช้วิธีการแบบต่างซึ่งมีลักษณะการให้บริการข้อมูล ที่มีลักษณะข้อดี ข้อเสีย ที่แตกต่างกันไปซึ่งการจะใช้งานของวิธีการจัดการคิวแบบใดนั้นขึ้นอยู่กับลักษณะของการใช้งานของผู้ใช้ว่าใช้งานข้อมูลแบบใดเป็นหลักในเครือข่ายซึ่งจะมีการปรับแต่งองค์ประกอบในวิธีการจัดการคิวในแบบดังในบทที่ 3 นั่นเอง

ซึ่งปริญญานิพนธ์ฉบับนี้นั้นจะได้เลือกใช้วิธีการจัดการคิวโดยวิธี Random Early Detection (RED) ที่มีการใช้งานอย่างแพร่หลายทั่วไปในปัจจุบัน โดยผู้จัดทำได้ทำการศึกษาถึงการทำงานและความสัมพันธ์ในการที่จะปรับแต่งค่าพารามิเตอร์ที่เกี่ยวข้องให้มีประสิทธิภาพในการจัดการทราฟฟิกที่หนาแน่นใน Router ซึ่งจะมีลักษณะการทำงานดังนี้

1. มีการคำนวณความยาวเฉลี่ยของคิว (Average Queue Length)
2. ต้องทำการกำหนดจุดจำกัดสูงสุดและต่ำสุด (Maximum & Minimum Threshold) เพื่อกำหนดความหนาแน่นของปริมาณทราฟฟิกที่สามารถจะเกิดได้บน Router
3. จะมีการสุ่มหยุด packets ด้วยความน่าจะเป็น p_b เมื่อความยาวเฉลี่ยของคิวมีค่าเกินจุดจำกัดต่ำสุดเป็นต้นไป
4. ใช้หลักการคล้ายคลึงกับวงจรกรองความถี่ต่ำ (Low Pass Filter) ในวงจรอิเล็กทรอนิกส์ที่จะอนุญาตให้เกิดความหนาแน่นของปริมาณทราฟฟิกได้ไม่เกินจุดๆหนึ่ง ซึ่งจะถูกกำหนดโดยพารามิเตอร์ w_q (Queue Weight)

โดยความสัมพันธ์ของการปรับแต่งค่าพารามิเตอร์ใน RED ให้มีประสิทธิภาพนั้นได้แสดงอย่างละเอียดเอาไว้ในบทที่ 4 แล้วซึ่งจะสรุปโดยสังเขปได้ถึงผลของการปรับแต่งคือ จะมีพารามิเตอร์ที่สำคัญคือ w_q ซึ่งจะมีความสัมพันธ์กับค่าจุดจำกัดสูงสุดและต่ำสุดดังสมการที่ (1) ซึ่งผลของการปรับค่า w_q นั้นจะได้ผลคือ เมื่อกำหนดให้ค่าของ w_q มีค่าต่ำจะทำให้การสุ่มหยุด packets ด้วยความน่าจะเป็น p_b นั้นสูง

ขึ้นเพราะว่าจะทำให้จุดจำกัดต่ำสุดนั้นมีค่าลดลง ส่งผลให้ปริมาณของ packets ที่รับได้ที่ปลายทางมีค่าน้อยทำให้ Throughput น้อย และเมื่อกรณีที่ปรับค่า w_q ให้สูงนั้นก็จะทำให้การสุ่มหยุด packets นั้นน้อยลงเนื่องจากจุดจำกัดต่ำสุดมีค่าสูงขึ้น ส่งผลให้ปริมาณของ packets ที่รับได้ที่ปลายทางมีค่าสูงขึ้นทำให้ Throughput สูงไปด้วย แต่อย่างไรก็ตามปริมาณทราฟฟิกที่จะเกิดใน Router ก็จะสูงมากขึ้นไปด้วยนั่นเอง นั่นหมายความว่าถ้ามีการกำหนดค่า w_q ที่สูงเกินไปก็อาจจะทำให้ RED Router ประพฤติตัวมันเองเสมือน FIFO Router ที่ไม่สามารถรับ packets ได้อีกเมื่อมีปริมาณของ packets ในบัฟเฟอร์เต็ม

5.2 แนวทางการพัฒนาต่อ

ในปริญญานิพนธ์นี้ผู้จัดทำได้ทำการออกแบบวิธีการใหม่เพื่อพัฒนาวิธีการจัดการคิวแบบ RED นี้ให้มีประสิทธิภาพมากขึ้นโดยจะทำการสร้างสมการของความน่าจะเป็นในการสุ่มหยุด packets p_b ขึ้นมาใหม่โดยอ้างอิงตามสมการพาราโบลา ซึ่งจะช่วยลดปริมาณของ packets ที่จะถูกหยุดไว้ได้บางส่วน ซึ่งเป็นการเพิ่ม Qos ให้ RED อีกด้วยดังแสดงการวิเคราะห์สมการไว้ในบทที่ 4 ส่วนแนวทางในการพัฒนาต่อไปนั้น ผู้พัฒนาควรจะศึกษาวิธีการจัดการคิวแบบอื่นๆซึ่งได้แสดงไว้แล้วดังบทที่ 3 โดยควรจะเข้าใจในการทำงานอย่าถ่องแท้และควรทำอย่างไรที่จะทำให้ระบบคิวนั้นมีประสิทธิภาพที่ดีขึ้น อาจจะด้วยการดัดแปลงสมการของการทำงานให้ดีขึ้น หรือ ประยุกต์ใช้งานนำวิธีที่ต่างกันมาผสมผสานกัน ที่จะให้ประสิทธิภาพดีขึ้น ซึ่งจะนำประโยชน์อย่างมากในการจัดการข้อมูลบนเครือข่ายที่มีขนาดใหญ่ และมีการใช้งานอย่างมากมาย ดังเช่น เครือข่ายอินเตอร์เน็ตนั่นเอง

กิตติกรรมประกาศ

ขอขอบพระคุณ อาจารย์ นภัทร สระเอี่ยม และ ผศ.ดร. สุทธิชัย นพนาคีพงษ์ อาจารย์ที่ปรึกษาทั้งสองท่านที่มอบโอกาสในการทำโครงงานนี้และคำแนะนำที่ดีมาโดยตลอด พี่ๆปริญญาโท, เพื่อนๆ และรุ่นน้องทุกๆท่าน ในภาควิชาวิศวกรรมโทรคมนาคม ที่ให้ความช่วยเหลือและกำลังใจเสมอมา พ่อแม่ผู้ให้กำเนิดและสนับสนุนทุกๆเรื่องมาทั้งชีวิต สุดท้ายนี้ ขอบคุณผู้สร้างสรรค์วิทยาการต่างๆในโลกนี้ และ อินเตอร์เนตที่เป็นแหล่งข้อมูลหลักในการทำโครงงานครั้งนี้

หนังสือและเอกสารอ้างอิง

[1] สุวัฒน์ ปุณณชัยยะ " เปิดโลก TCP/IP และโปรโตคอลของอินเตอร์เนต " โปรวิชั่น , 2545

[2] Brent B. welch, " Practical Programming in Tcl and Tk ," Prentice Hall Ptr, 1995

[3] http://ntl.nectec.or.th/internet

[4] http://antares.math.tau.ac.il/~alx/courses03/notes/Icc4.ppt

[5] http://www.cs .cmu.edu/~srini/15-744/S01/lectures/

[6] http://www.ict.tuwien.ac.at/skripten/datenkomm/print/35-IPQoSPrimer.pdf

[7] http://netlab1.bu.edu/~staro/546projects/RED/SC546/what_is_red.html

[8] http://www.info.fundp.ac.be/~infonet/coursenligne/Info2231/INFO2231-3/

[9] CS459:Special Topics in Computer Networking Research Spring 2002," http://www-courses.cs.uiuc.edu/~cs497hou/ "

[10] S. Floyd, RED:Discussion of Setting Parameter," http:// www.aciri.org/floyd/ REDparameters.txt "

[11] The ns Manuals," http://www.isi.edu/nsnam/ns/ns-documentation.html "

[12] OTcl Tutorial," ftp://ftp.tns.lcs.mit.edu/pub/otcl/doc/ "

[13] A. Haider, H. Sirisena, K. Pawlikowski and M. J. Ferguson," Congestion Control Algorithms in High Speed Telecommunication Networks "

[14] P. Rai, S. K. Yadav," Queue Scheduling for TCP Traffic ," M. Tech. CSE IITD

[15] M. Shreedhar and G. Varghese," Efficient fair queuing using deficit round robin," IEEE/ACM Transactions on Networking, 1996

[16] J.-S. Li," An Evaluation of Deficit Round Fair Queuing Applied in Router Congestion Control," Journal of Information Science and Engineering, Vol. 18/2 (SCI, EI)

[17] S. Floyd and V. Jacobson," Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Trans. Networking, August 1993

[18] S. Floyd and V. Jacobson," Link-sharing and Resource Management Medels for Packet Networks," IEEE/ACM Trans. Networking, August 1995

[19] P. E. McKenny," Stochastic fairness queuing," Internetworking: Research and Experence, Vol. 2, 1991

[20] Christophe Deleuze," Scheduling ," report final COST237, 1999