แบบจำลองเครือข่าย

Network Simulator

โดย

นายวรชาติ น้ำไชยศรี

นางสาวศิริรักษ์ เชิดชู

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

NETWORK SIMULATOR

BY

MR.VARACHAT NUMCHAISRI

MS.SIRIRAK CHIRDCHOO

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

BACHELOR IN DEPARTMENT OF INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF THECHNOLOGY LADKRABANG

2003

หัวข้อปริญญานิพนธ์	แบบจำลองเครือข่าย	
ชื่อนักศึกษา	นายวรชาติ น้ำไชยศรี	รหัสประจำตัว 43010363
	นางสาวศิริรักษ์ เชิดชู	รหัสประจำตัว 43010432
อาจารย์ที่ปรึกษา	อาจารย์ภุชงค์ หงษ์สุวรรณ	
ระดับการศึกษา	ปริญญาตรี วิศวกรรมศาสตรบัณฑิต	
	สาขาวิศวกรรมสารสนเทศ	
ภาควิชา	วิศวกรรมสารสนเทศ	
ปีการศึกษา	2546	

ปริญญานิพนธ์ฉบับนี้ได้รับความเห็นชอบจากอาจารย์ที่ปรึกษาเป็นที่เรียบร้อยแล้ว

.....

(อาจารย์ภุชงค์ หงษ์สุวรรณ)

อาจารย์ที่ปรึกษา

หัวข้อปริญญานิพนธ์	แบบจำลองเครือข่าย
ชื่อนักศึกษา	นายวรชาติ น้ำไชยศรี รหัสนักศึกษา 43010363
	นางสาวศิริรักษ์ เชิดชู รหัสนักศึกษา 43010432
อาจารย์ที่ปรึกษา	อาจารย์ภุชงค์ หงษ์สุวรรณ
ระดับการศึกษา	ปริญญาตรี วิศวกรรมศาสตรบัณฑิต
	สาขาวิศวสารสนเทศ
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2546

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาซอฟต์แวร์แบบจำลองการทำงานของเครือข่าย โดยสามารถจำลองการทำงานของ เราเตอร์ และลิงค์ระหว่างเราเตอร์ IP และสามารถกำหนดปริมาณ และช่วงเวลาในการส่งได้ โดยมีการแสดงผลแบบกราฟิก และสามารถแสดงประสิทธิภาพในการส่งข้อมูลที่ได้จากการจำลอง

Thesis Title	Network Simulator	
Student	Mr.Varachat Numchaisri	ID. 43010363
	Ms.Sirirak Chirdchoo	ID. 43010432
Advisor	Mr.Puchong Hongsuwan	
Graduate Level	Bachelor Degree of Information Engineering	
Department	Information Engineering	
Academic Year	2003	

ABSTRACT

The purpose of this project is to design and develop a prototype of network traffic monitoring tool using Java programming language. This tool has ability to simulate network-links and router operations, to monitor network traffic IP protocol and to display graphical output report grouped by type and quantity of traffic at a certain interval time frame

กิตติกรรมประกาศ

โครงงานนี้ได้จัดทำขึ้นเป็นผลสำเร็จ ทางคณะผู้จัดทำขอขอบคุณบิดา มารดา ญาติพี่น้อง ที่คอยช่วยเหลือและให้กำลังใจ คณะบูรพาจารย์ทั้งหลาย ท่านผู้เขียนเอกสารและตำราอ้างอิงต่าง ๆ ทุกท่านโดยเฉพาะอาจารย์ที่ปรึกษา รวมถึงรุ่นพี่และมิตรสหายวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่านที่ให้คำปรึกษาแนะนำ และช่วยเหลือในการสนับสนุนในการหาข้อมูลต่าง ๆ ทั้งด้านทฤษฎีและปฏิบัติเป็นอย่างดียิ่ง รวมถึง เว็บไซท์ต่าง ๆ ที่ให้ความรู้ในการจัดทำโครงงาน ทั้งนี้คณะผู้จัดทำต้องขอขอบพระคุณ ภาควิชาวิศวกรรมสารสนเทศ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดการบัง ที่ได้ให้คณะผู้จัดทำได้มีโอกาสเข้ามาศึกษา ณ สถาบันแห่งนี้

สุดท้ายนี้ทางคณะผู้จัดทำต้องขอกราบขอบพระคุณบุคคลผู้ที่มีความสำคัญมากที่สุดที่ทำให้มีวันนี้ คือ บิดา มารดา ที่สนับสนุนและคอยช่วยเหลือคณะผู้จัดทำตลอดเรื่อยมาในทุก ๆ ด้าน จนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดี จึงขอกราบขอบพระคุณมา ณ ที่นี้

คณะผู้จัดทำ

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

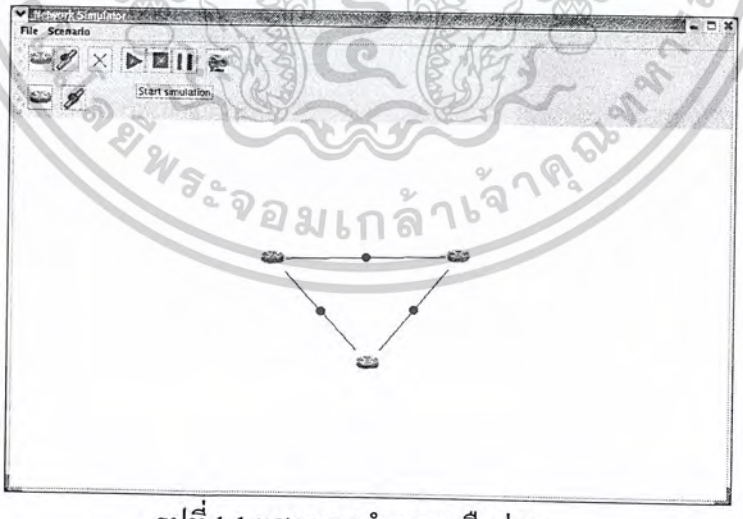
สารบัญรูปภาพ (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 แนวคิดและที่มาของปัญหา

ในการปฏิบัติงานด้านเน็ตเวิร์ค (network) นั้น บุคลากรจะต้องมีความรู้ ความเข้าใจในด้านเน็ตเวิร์คเป็นอย่างดี แต่ในทางการศึกษานั้นเป็นการยากที่จะลงทุนซื้ออุปกรณ์ทางด้านเน็ตเวิร์ค เช่น เราเตอร์ (router) เครื่องคอมพิวเตอร์ (computer) เป็นต้น หลาย ๆ ตัวมาต่อเพื่อศึกษาได้ ดังนั้นผู้จัดทำโครงงานนี้ได้เล็งเห็นว่า ถ้าหากเราสามารถจำลองแบบเครือข่ายเสมือนจริงมาศึกษาได้เสมือนจริง ทั้งคำสั่งและผลที่ได้ออกมาเหมือนการปฏิบัติการจริง ๆ ถึงแม้ว่าไม่ได้มีการต่ออุปกรณ์จริง แต่โครงงานนี้ก็จะแสดงภาพให้เห็นจากการใช้คำสั่งนั้น ๆ กับอุปกรณ์เน็ตเวิร์ค และ แสดงเป็นกราฟเปรียบเทียบการส่งโปรโตคอล (protocol) IP เปรียบเทียบกับช่วงเวลาให้เห็นได้ชัดเจนขึ้น ซึ่งถ้าในการปฏิบัติจริงภาพในการทำงานตรงนี้จะไม่สามารถมองเห็นได้ ซึ่งผู้ปฏิบัติการต้องเข้าใจว่าการใช้คำสั่งใดแล้วจะมีผลที่เกิดขึ้นอย่างไรในเน็ตเวิร์ค ซึ่งซอฟต์แวร์ (software) ของแบบจำลองเครือข่ายในปัจจุบันนั้นก็มีขึ้นมาให้นำมาใช้ในการศึกษา แต่ขีดความสามารถ ความ ซับซ้อนในการทำงานของเน็ตเวิร์คเมื่อเทียบกับเน็ตเวิร์คจริงจะน้อยกว่า ในโครงงานนี้ได้จำลองเน็ตเวิร์คแบบที่ใกล้เคียงกับระบบเน็ตเวิร์คจริงขึ้นมา



รูปที่ 1.1 แสดงการจำลองเครือข่าย

1.2 จุดประสงค์

1.2.1 เพื่อพัฒนาซอฟต์แวร์ของแบบจำลองเน็ตเวิร์ค

1.2.2 เพื่อใช้ในการศึกษาการทำงานของอุปกรณ์ต่าง ๆ ในเน็ตเวิร์ค

1.2.3 เพื่อเปิดโอกาสให้แก่ผู้ที่สนใจศึกษาด้านเน็ตเวิร์คสามารถมีโอกาสได้เสมือนปฏิบัติการจริง เช่น รูปแบบของคำสั่งที่ใช้ติดตั้ง (set) เราเตอร์ เป็นต้น

1.3 ขอบเขตของโครงงาน

1.3.1 เราเตอร์

1.3.1.1 สามารถกำหนด กำหนด (Config) ต่าง ๆ ได้ เช่น ชื่อ โหนด (Node) , หมายเลข ไอพี (IP) , การเพิ่ม (add) อินเตอร์เฟส (Interface) ไปใน เราเตอร์ ได้

1.3.1.2 สามารถกำหนด โปรโตคอล เป็นแบบ เราติ้งอินฟอร์เมชั่นโปรโตคอล (RIP) ที่ใช้ในการ หาเส้นทาง (Routing) ได้

1.3.1.3 สามารถกำหนดการจัดการด้าน การควบคุมความคับคั่งของทราฟฟิก (Congestion Control) ของเราเตอร์ได้

1.3.2 ลิงค์

1.3.2.1 สามารถกำหนดการเชื่อมต่อระหว่าง Router ได้

1.3.2.2 สามารถกำหนดแบนด์วิดท์ (Bandwidth) บนลิงค์ได้

1.3.3 แบบจำลอง (Simulate)

1.3.3.1 สามารถจำลองการส่งข้อมูลแบบ TCP , UDP ผ่านเราเตอร์และลิงค์ โดยกำหนดปริมาณข้อมูลและช่วงเวลาได้

1.3.4 รายงาน (Report)

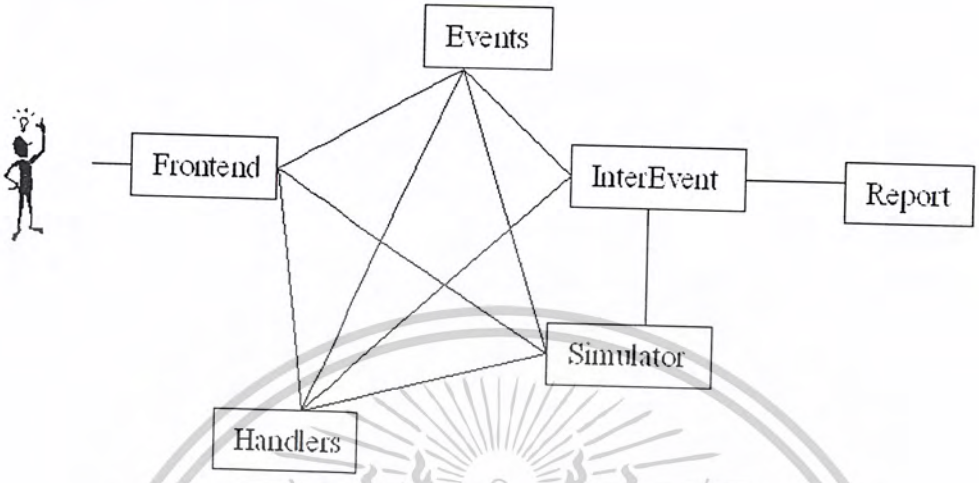
1.3.4.1 สามารถแสดงการใช้ประโยชน์ (Utilized) บนสายส่งสัญญาณเพื่อแสดงปริมาณการส่งข้อมูลแต่ละการเชื่อมต่อได้ในรูปแบบของกราฟ

1.3.4.2 สามารถแสดงค่าเฉลี่ยการยืดเวลา (Delay Time) ของสายส่งข้อมูลแต่ละเซสชัน (session) ได้

1.3.4.3 แสดงค่าเฉลี่ยแบนด์วิดท์ในแต่ละช่วงเวลาได้

1.3.4.4 สามารถแสดง Utilized แบนด์วิดท์ ของเราเตอร์ ที่ถูกใช้งานไปเป็นเปอร์เซ็นต์ ด้วย gressbar

1.4 สถาปัตยกรรมของระบบ



รูปที่ 1.2 แสดงสถาปัตยกรรมของระบบ

1.4.1 ฮาร์ดแวร์ (Hardware) ระบบคอมพิวเตอร์ (Computer Server)

1.4.2 ซอฟท์แวร์ (Software) ที่ใช้จำลองระบบเน็ตเวิร์คคือ Java Programming Language, J2SDK , Borland Jbuilder 9 Personal เป็นต้น

1.5 ตารางการทำงาน

ID	Task Name	2003							2004		
		Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar
1	**Problem Definition & Get Requirement**										
2	Analysis & Design										
3	Software Design										
4	User Interface Design										
5	Implementation										
6	Test & Debug										
7	Documentation										

ตารางที่ 1.1 แสดงตารางการทำงาน

บทที่ 2

ทฤษฎีและหลักการที่ใช้ในโครงงาน

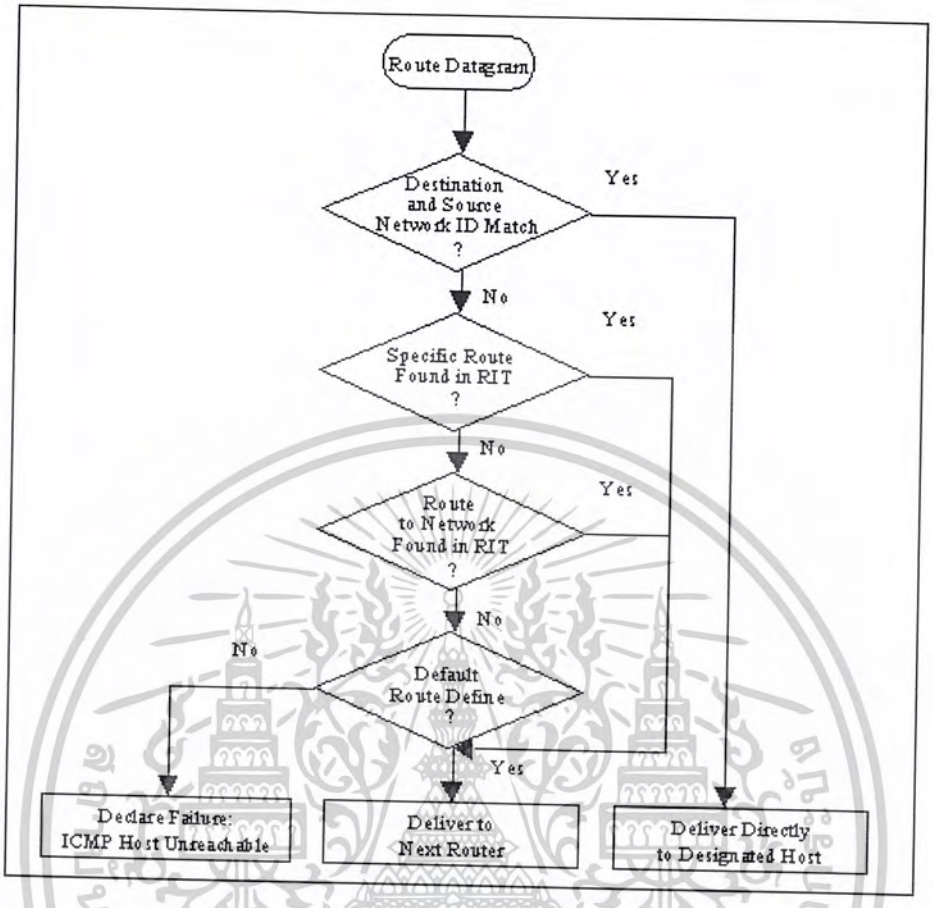
2.1 การหาเส้นทาง (Routing)

การหาเส้นทางจัดเป็นหน้าที่สำคัญของระดับชั้นเน็ตเวิร์ค กระบวนการหาเส้นทางนั้นเกิดขึ้นที่โหนดต่าง ๆ ในเน็ตเวิร์ค คือเมื่อมีแพ็คเกจเข้ามายังโหนดใด ๆ โหนดนั้นจะต้องตัดสินใจว่าควรจะส่งแพ็กเก็ต (packet) ที่เข้ามา นั้นออกไปทางไหนเพื่อให้ข้อมูล ถึงปลายทางได้อย่างถูกต้องและรวดเร็ว โดยเฉพาะในเครือข่าย ไอพี ซึ่งเป็น การทำงานแบบดาต้าแกรม (datagram) การตัดสินใจจึงต้องทำกับทุกแพ็คเกจที่เข้าส่งเข้ามา ปัจจุบัน ไอพี เน็ตเวิร์คจัดว่า เป็นเน็ตเวิร์คที่ขนาดใหญ่ที่สุด (อินเทอร์เน็ต (Internet) ในปัจจุบัน) นั่นหมายความว่ามีเครื่องคอมพิวเตอร์ที่ต่ออยู่กับ ไอพี เน็ตเวิร์คทั่วโลก สิ่งที่น่าสนใจคือ เมื่อเครื่องคอมพิวเตอร์ ก ต้องการติดต่อ ส่งข้อมูลกับเครื่องคอมพิวเตอร์ ข ซึ่งทั้ง ก และ ข ต่างก็ ต่ออยู่กับ ไอพี เน็ตเวิร์ค ข้อมูลดังกล่าวนั้นถูกส่งออกจาก เครื่องคอมพิวเตอร์ ก เข้าไปในเน็ตเวิร์คส่งไปยังเครื่อง คอมพิวเตอร์ โดยรวดเร็วและถูกต้องได้อย่างไร ใน ไอพี เน็ตเวิร์ค นั้นใช้อุปกรณ์ เราเตอร์ ในการเชื่อมต่อเครือข่ายเข้าด้วยกันและเป็นตัวตัดสินใจเลือกเส้นทางให้กับแพ็กเก็ตที่อยู่ ในเน็ตเวิร์ก แล้วส่งข้อมูลไปยังปลายทาง โดยที่ในการตัดสินใจเลือกเส้นทางของ เราเตอร์ นั้นมีสิ่งที่เกี่ยวข้องอยู่ 3 ส่วนคือ

1. อัลกอริธึมค้นหาเส้นทาง (The routing algorithm)
2. ตารางข้อมูลค้นหาเส้นทาง (The routing information table (RIT))
3. การปรับปรุงตารางข้อมูลการค้นหาเส้นทาง (Maintenance of routing information table)

2.1.1 อัลกอริธึมค้นหาเส้นทาง

เมื่อเราเตอร์ได้รับแพ็กเก็จเข้ามาแล้วจะมีขั้นตอนในการตัดสินใจเลือกเส้นทางว่าจะส่งแพ็กเก็จออกไปทางทิศใดดังรูป



รูปที่ 2.1 แสดงอัลกอลิทึมของการหาเส้นทาง

จากรูปเมื่อเราเตอร์ได้รับแพ็คเกจข้อมูลเข้ามาในขั้นแรกจะทำการตรวจสอบว่า แอดเดรสเป้าหมาย (destination address) นั้นอยู่ใน หมายเลขเน็ตเวิร์ค (network address) เดียวกันกับเราเตอร์ หรือไม่ ถ้าใช่ก็จะทำการส่งแพ็คเกจนั้นไปยังอินเตอร์เฟสที่มีหมายเลขเน็ตเวิร์คเดียวกัน เพื่อส่งไปยัง เป้าหมายโฮสท์ (destination host) แต่ถ้าอยู่คนละเน็ตเวิร์คไอดี (Network ID) ก็จะทำการค้นหา หมายเลขเน็ตเวิร์ค ของ เป้าหมาย (destination) จาก ตารางข้อมูลค้นหาเส้นทาง จะกล่าวถึงต่อไป ถ้าพบก็จะส่งแพ็กเก็ต ไปยังจุดหมายปลายทางต่อไป

ขั้นตอนในการค้นหาเส้นทางจาก ตารางข้อมูลค้นหาเส้นทาง มีดังนี้

1. ค้นหาจากหมายเลขโฮสท์ (host address) ปลายทางที่ระบุเจาะจงอยู่ใน ตารางข้อมูลค้นหาเส้นทาง ว่ามีหรือไม่ ถ้ามีก็จะส่งไปให้โฮสท์นั้นตามเส้นทางที่ระบุในตารางข้อมูลค้นหาเส้นทา

2. ถ้าไม่พบโฮสท์นั้นในตารางข้อมูลค้นหาเส้นทาง ก็จะดูว่าโฮสท์นั้นอยู่ในหมายเลขเน็ตเวิร์คไหน และนำหมายเลขเน็ตเวิร์คนั้นไปค้นหาในตารางข้อมูลค้นหาเส้นทาง ถ้าพบก็จะส่งแพ็คเกจไปตามเส้นทางที่ระบุใน ตารางข้อมูลค้นหาเส้นทาง

3. ถ้าไม่พบ หมายเลขเน็ตเวิร์คนั้นในตารางข้อมูลค้นหาเส้นทาง ก็จะส่งแพ็คเกจไปยังดีฟอลทเราท์ (default route) ที่ระบุในตารางข้อมูลค้นหาเส้นทาง แต่ถ้าไม่มีการกำหนดดีฟอลทเราท์ไว้ในตารางข้อมูลค้นหาเส้นทาง ก็จะส่ง แมสเสจ (message) บอกโฮสท์ต้นทางว่าไม่สามารถหาเส้นทางเพื่อส่งข้อมูลได้

2.1.2 ตารางข้อมูลค้นหาเส้นทาง

ตารางข้อมูลค้นหาเส้นทาง เป็นตารางที่เก็บแอดเดรสและเส้นทางที่จะไปยังแอดเดรสนั้น โดยมีข้อมูลที่อยู่ในตารางดังนี้

เป้าหมาย	ดิสแทน (Distance)	แฟล็ก(Flag)	เราเตอร์ถัดไป (Next router)	อินเตอร์เฟส
172.16.14.3	1	UH	-	le0
198.14.0.0	2	UGD	198.14.16.16	le0
192.16.0.0	2	UG	198.14.16.16	le0
0.0.0.0	1	UG	140.202.13.1	le1

ตารางที่ 2.1 แสดงตารางการหาเส้นทาง

เป้าหมาย คือหมายเลขโฮสท์ หรือหมายเลขเน็ตเวิร์คปลายทาง ดิสแทน คือระยะทางหรือจำนวนฮอพ (hop) จากต้นทางคือโฮสท์นี้ไปยังปลายทาง แฟล็ก จะบอกถึงสถานะของข้อมูลรายการนี้ U หมายความว่าเส้นทางนั้นยังใช้งานอยู่ (Up and operational) G บ่งบอกว่าเส้นทางนี้เป็นการส่งดาต้าแกรมไปยัง เกตเวย์ (Gateway) ถ้าหากว่าเส้นทางนั้นสามารถส่งข้อมูลไปถึงปลายทางได้เลยจะไม่ขึ้นแฟล็ก H บ่งบอกว่าเส้นทางนี้ใช้ในการเดินทางไปยังโฮสท์ใดโฮสท์หนึ่ง (แต่ปกติเราจะให้เป็นการเดินทางไปยังเน็ตเวิร์คมากกว่า) D จะถูกติดตั้งเมื่อเส้นทางนี้ถูกสร้างขึ้นเนื่องจากถูกรีไดเร็ก (redirect) จากอินเทอร์เน็ตคอนโทรลแมสเสจโปรโตคอล (ICMP) เราเตอร์ถัดไป เป็นแอดเดรสของเราเตอร์ตัวถัดไปที่จะส่งแพ็คเกจนี้ไปให้ อินเตอร์เฟส คือ เน็ตเวิร์ค อินเตอร์เฟส การ์ด (Network Interface Card) ของ โฮสท์

หรือ เราเตอร์ นั้นที่จะส่งข้อมูลออกไป ในกรณีที่ต้องการใช้ ดีฟอล์ท เราท์ (default route) ให้ระบุ แอดเดรสเป้าหมาย เป็น 0.0.0.0 หรือใช้คำว่า ดีฟอล์ท แล้วระบุเส้นทางที่ต้องการให้เป็น ดีฟอล์ทเราท์

2.1.3 การปรับปรุง ตารางข้อมูลค้นหาเส้นทาง

การปรับปรุงข้อมูลในตารางข้อมูลค้นหาเส้นทางนั้นสามารถทำได้ทั้งแบบไดนามิค(dynamic) โดยใช้โปรโตคอลการหาเส้นทาง หรือแบบสแตติก (static) โดยการกำหนดลงในตารางข้อมูลค้นหาเส้นทางโดยตรงก็ได้การปรับปรุงข้อมูลแบบสแตติกเรียกอีกอย่างหนึ่งว่าแบบไม่ปรับเปลี่ยน (nonadaptive) คือการหาเส้นทางไม่ขึ้นอยู่กับการสัญจรของข้อมูลหรือรูปร่างของเครือข่ายในขณะนั้น แต่จะมีการกำหนดเส้นทางก่อนที่จะใช้งาน เมื่อเริ่มใช้งานก็จะทำการโหลดข้อมูลเข้าไปในเราเตอร์ เพื่อใช้ในการหา เส้นทาง ส่วนแบบ ไดนามิค หรือเรียกอีกอย่างว่าแบบปรับเปลี่ยนได้ (adaptive) นั้นการพิจารณาหาเส้นทางจะขึ้น อยู่กับการสัญจรของข้อมูลและรูปร่างในขณะนั้นของเครือข่าย ซึ่งแบบปรับเปลี่ยนได้นี้แบ่งออกเป็น 3 แบบ ขึ้น อยู่กับข้อมูลที่ใช้ในการตัดสินหาเส้นทาง ได้ดังนี้

- การหาเส้นทางแบบใช้ข้อมูลรวม
- การหาเส้นทางแบบใช้ข้อมูลเฉพาะตัว
- การหาเส้นทางแบบใช้ข้อมูลเฉพาะที่

2.1.4 การหาเส้นทางแบบใช้ข้อมูลรวม

การหาเส้นทางโดยวิธีนี้จะมีการคำนวณตารางหาเส้นทางที่ศูนย์กลางหนึ่งเมื่อคำนวณเรียบร้อยแล้วก็จะส่งตาราง ไปยังโหนดต่าง ๆ ของเครือข่าย เราติ้งคอนโทรลเซ็นเตอร์ (RCC) โดยจะทำ การคำนวณหาเส้นทางทุก ๆ ช่วงเวลาหนึ่ง แต่ละโหนดในเครือข่ายจะส่งข้อมูลที่แสดงสภาวะเครือข่ายให้แก่ เราติ้งคอนโทรลเซ็นเตอร์ ข้อมูลเหล่านี้ได้แก่ จำนวนข้อมูลที่รอการส่งอยู่ในบัฟเฟอร์ จำนวนแพ็คเกจที่ถูกส่งผ่านแต่ละสายของโหนด เป็นต้น เมื่อ เราติ้งคอนโทรลเซ็นเตอร์ ได้รับข้อมูลนี้ก็จะทราบสภาวะของเครือข่ายและคำนวณหาเส้นทางที่ดีที่สุดแล้วจึงส่งตารางที่คำนวณได้ กลับคืนไปยังโหนดต่าง ๆ ข้อดีสำหรับวิธีนี้คือ การหาเส้นทางนั้นใช้ข้อมูลทั้งหมดของเครือข่าย จึงทำให้ได้เส้นทางที่เหมาะสมและยังลด ภาระในการคำนวณของโหนดต่าง ๆ แต่ข้อเสียคือหากเครือข่ายมีการเปลี่ยนแปลงบ่อยจะต้องทำการคำนวณใหม่ ทุกครั้ง รวมทั้งในกรณีที่เครือข่ายมีขนาดใหญ่การคำนวณที่ เราติ้งคอนโทรลเซ็นเตอร์ จะต้องใช้เวลานาน ถึงแม้จะเลือกใช้อุปกรณ์ที่มี ประสิทธิภาพสูงมาเป็น เราติ้งคอนโทรลเซ็นเตอร์ ก็ตามเพราะเนื่องจากว่าต้องรอข้อมูลทั้งหมดจากเครือข่ายก่อนจึงเริ่มทำการ

คำนวณ อีกทั้งถ้า เราติ้งคอนโทรลเซ็นเตอร์ เกิดความเสียหายก็จะไม่สามารถหาเส้นทางที่เหมาะสมสำหรับเครือข่ายได้ ซึ่งอาจจะต้องทำการ ติดตั้ง เราติ้งคอนโทรลเซ็นเตอร์ สำรองไว้สำหรับกรณีนี้ด้วย นอกจากนี้การส่งข้อมูลที่คำนวณได้กลับไปสู่โหนดต่าง ๆ นั้น โหนดที่อยู่ใกล้ เราติ้งคอนโทรลเซ็นเตอร์ จะได้รับข้อมูลก่อนส่วนโหนดที่อยู่ไกลออกไปจะได้รับข้อมูลที่หลังการใช้งานข้อมูลดังกล่าวจึงไม่สอดคล้องกัน

เนื่องจากวิธีการนี้มีปัญหาดังที่กล่าวมาข้างต้น และในปัจจุบันคอมพิวเตอร์มีความสามารถสูงขึ้นในขณะที่ราคา ต่ำลง แทนที่โหนดทุกตัวจะต้องส่งข้อมูลให้แก่ เราติ้งคอนโทรลเซ็นเตอร์ โหนดทุกตัวจะส่งข้อมูลไปยังโหนดข้างเคียงของมันและตัวที่ ได้รับข้อมูลก็จะส่งข้อมูลให้โหนดข้างเคียงต่อไป ทำให้โหนดทุกตัวมีข้อมูลทั้งหมดเครือข่ายและสามารถคำนวณหา เส้นทางที่เหมาะสมได้เอง

2.1.5 การหาเส้นทางแบบใช้ข้อมูลเฉพาะตัว

การคำนวณหาเส้นทางวิธีนี้จะใช้ข้อมูลของแต่ละโหนดเอง ในการตัดสินใจว่าจะส่งข้อมูลออกไปในทิศทางใด ถึงแม้ว่าวิธีนี้จะมีการปรับเปลี่ยนเส้นทางตามลักษณะและปริมาณของข้อมูลที่ทำการส่งอยู่ในขณะนั้นก็ตาม แต่ก็ไม่มีการแลกเปลี่ยนข้อมูลกันระหว่างโหนดเลย วิธีการแบบหนึ่งของการหาเส้นทางแบบนี้คือเมื่อมีแพ็คเกจส่งเข้ามายังโหนด โหนดนั้นจะพยายามส่งแพ็คเกจนั้น ออกไปให้เร็วที่สุด โดยส่งออกไปยังสายที่มีการรอคอยของการส่งข้อมูลที่สั้นที่สุดในขณะนั้น โดยไม่สนใจว่าจะเป็นเส้นทางที่สั้นที่สุดหรือไม่ วิธีการนี้จึงเรียกกันทั่วไปว่า ฮอทโพเตโตอัลกอริทึม (hot potato algorithm)

บางทีวิธีนี้อาจใช้ค่าสถิติการรอคอยการส่งข้อมูลของแต่ละสายเป็นน้ำหนักเพื่อใช้ในการพิจารณาด้วยกล่าวคือเมื่อ ข้อมูลเข้ามายังโหนดแล้ว อาจจะเลือกสายที่มีน้ำหนักดีที่สุดยกเว้นการณีที่ความยาวของแพ็คเกจที่กำลังรอการส่ง ของสายนั้นยาวเกินกว่าค่าวิกฤต (threshold) ค่าหนึ่ง

2.1.6 การหาเส้นทางแบบใช้ข้อมูลเฉพาะที่

การหาเส้นทางโดยวิธีการนี้ แต่ละโหนดจะใช้ข้อมูลของตัวมันเอง ตลอดจนข้อมูลของโหนดที่อยู่ติดกับตัวมัน วิธีนี้แต่ละโหนดจะมีตารางหาเส้นทางซึ่งบอกว่าจากโหนดนั้น ๆ สามารถไปถึงโหนดอื่น ๆ ในเครือข่ายได้โดย มีเวลาหน่วงโดยประมาณเท่าไหร่ เวลาหน่วงโดยประมาณนี้จะถูกปรับปรุงระยะ ๆ โดยทุก ๆ โหนดจะแลกเปลี่ยน ข้อมูลเวลาหน่วงของตัวมันเองกับโหนดข้างเคียง ค่าของความหน่วงโดยประมาณอาจจะเป็นจำนวนของโหนดที่จะ ถึงปลายทาง เวลาล่าช้าของการส่งจริง ๆ หรือความยาวของแถวรอคอยของเส้นทางนั้น เป็นต้น

จะเห็นว่าการหาเส้นทางแบบนี้ทุก ๆ โหนดจะต้องรู้ว่าระยะทางระหว่างตัวเองถึงโหนดอื่น ๆ ทั้งหมดในเครือข่าย เท่าไร และเมื่อมีแพ็คเกจผ่านเข้ามาสวิตชิงโหนดมันก็จะพิจารณาว่าจะต้องส่งแพ็คเกจนั้น ๆ ออกทางสายส่ง ข้อมูลสายไหนเพื่อที่ระยะทางจะได้สั้นที่สุด ดังนั้นวิธีนี้จึงเรียกอีกอย่างหนึ่งว่าเป็นการหาเส้นทางโดยใช้ตาราง ระยะทาง (Distance vector routing)

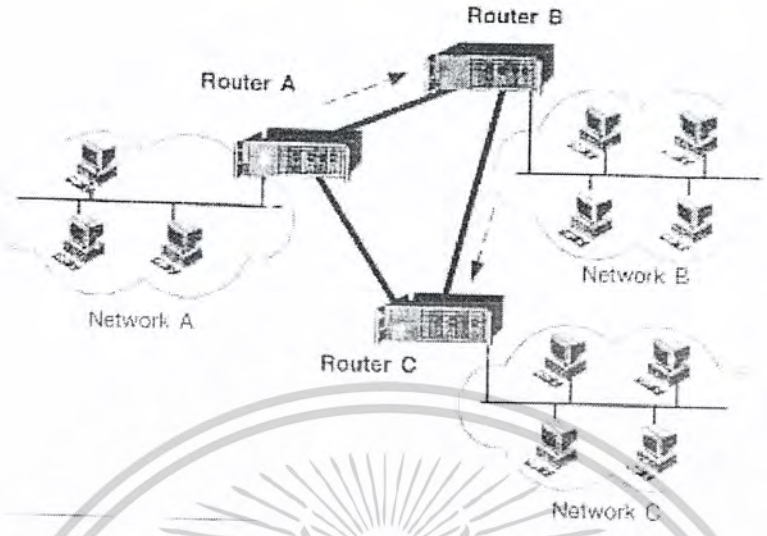
การ อัพเดต (update) ตารางข้อมูลค้นหาเส้นทาง ด้วยโปรโตคอลการหาเส้นทางนั้นแบ่งออกเป็นสองกลุ่มใหญ่ ๆ ด้วยกันคือ อินทีเรียเกตเวย์โปรโตคอล (Interior Gateway Protocol) และ เอ็กซ์ทีเรียเกตเวย์โปรโตคอล (Exterior Gateway Protocol)

2.1.7 อินทีเรียเกตเวย์โปรโตคอล

เป็น โปรโตคอล ที่ใช้ภายในระบบเครือข่ายกลุ่มหนึ่ง ๆ ที่ต่อเชื่อมและถือเป็นหน่วยเดียวกัน ซึ่งเรียกกลุ่ม เครือข่ายเหล่านี้ว่า ออโตโนมัสซีสเท็ม (Autonomous System) ในแต่ละ ออโตโนมัสซีสเท็ม อาจจะประกอบด้วยเครือข่ายย่อย ๆ หลาย เครือข่ายและเชื่อมต่อกันโดยใช้ Gateway หลาย ๆ ตัวก็ได้ หลักการพิจารณาว่าเครือข่ายใดอยู่ใน ออโตโนมัสซีสเท็ม เดียวกันหรือไม่นั้นไม่ค่อยชัดเจน แต่อาจใช้หลักการดังนี้คือ เครือข่ายใดที่อยู่ในองค์กรเดียวกันและระยะทาง ต่อเชื่อมไม่ไกลมากนัก สามารถถือได้ว่าเป็น ออโตโนมัสซีสเท็ม เดียวกัน ตัวอย่างของ โปรโตคอล นี้เช่น เราติ้งอินฟอร์เมชันโปรโตคอล (RIP), โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์ (OSPF), ออนดีมานด์เราติ้ง (ODR), อินทีเรียเกตเวย์โปรโตคอลต้นหาเส้นทาง (IGRP) เป็นต้น

2.1.8 การระบุเส้นทางด้วย สแตติกเราเตอร์ (Static Router) และ โปรโตคอลต้นหาเส้นทาง

ในช่วงแรกสุดนั้นอุปกรณ์ เราเตอร์ จะรู้จักเฉพาะเครือข่ายและลูกข่ายของตนที่เชื่อมต่อกันอยู่เท่านั้น เราเตอร์ จะเรียนรู้การเชื่อมต่อของเครือข่ายอื่น ๆ ได้จาก 2 วิธีคือ การกำหนดเส้นทางการส่งผ่านข้อมูลแบบกำหนดให้ตายตัวหรือเรียกว่า สแตติกเราเตอร์ หรือเรียนรู้จากโปรโตคอลสำหรับการเปลี่ยนแปลงข้อมูลเส้นทางในการส่งผ่านข้อมูลหรือ เราเตอร์ โปรโตคอล ที่มีการติดต่อกัน การกำหนดเส้นทางการส่งผ่านข้อมูลแบบตายตัวหรือ สแตติกเราเตอร์ นี้เป็นการระบุเส้นทางในการส่งผ่านข้อมูล โดยผู้ดูแลระบบเป็นผู้คิดและจัดทำขึ้น ให้แต่ละการเชื่อมต่อระหว่างเครือข่ายและเครื่องปลายทางมีเส้นทางที่ตายตัว จากนั้นเก็บเป็นข้อมูลเส้นทางลงเป็น ตารางหาเส้นทาง (routing table) ในเราเตอร์ ข้อมูลดังกล่าวจะไม่สามารถแก้ไขหรือเปลี่ยนแปลงระหว่างการทำงานของ เราเตอร์ ได้แม้ว่าเครื่องปลายทางจะมีปัญหาหรือวงจรเชื่อมโยงบางช่วงจะล่มไปก็ตาม ตัวอย่างดังรูป



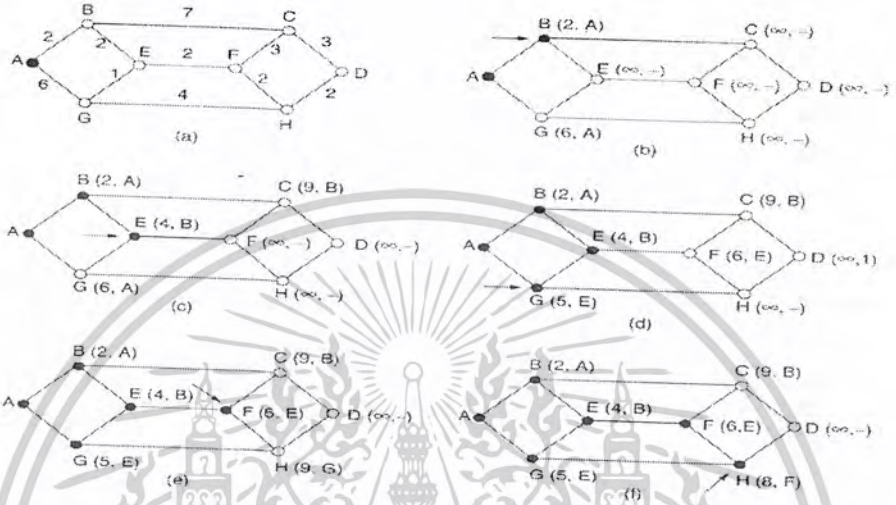
รูปที่ 2.2 แสดงการเชื่อมโยงของเน็ตเวิร์ค

ตัวอย่าง เราเตอร์ A ทำหน้าที่เชื่อมโยงเครือข่าย เน็ตเวิร์ค A กับ เน็ตเวิร์ค B และ เน็ตเวิร์ค C ภายในตัวเราเตอร์ A ถูกกำหนดเส้นทางการส่งผ่านข้อมูลแบบ สแตติกเราเตอร์ โดยมีข้อกำหนดว่า ถ้าต้องการส่งข้อมูลไปที่เครือข่าย เน็ตเวิร์ค C จะต้องผ่านไปที่ เราเตอร์ B เสียก่อน ในรูปแบบดังกล่าวถ้าการเชื่อมโยงระหว่าง เน็ตเวิร์ค A และ เน็ตเวิร์ค B หรือวงจรเชื่อมโยงระหว่าง เน็ตเวิร์ค B และ เน็ตเวิร์ค C ล่มไปหรือไม่สามารถใช้งานได้ การส่งผ่านข้อมูลจาก เน็ตเวิร์ค A ก็จะทำไม่ได้เช่นกันนอกเสียจากว่าจะมีการแก้ไขข้อมูลสแตติกเราเตอร์และตารางหาเส้นทางเสียก่อน ทางแก้ไขที่ดีกว่าเกิดขึ้นตามแนวคิดที่ว่าถ้าให้ เราเตอร์ A สามารถเรียนรู้ว่าการส่งผ่านข้อมูลนี้ สามารถอาศัยเส้นทางอื่นได้ก็จะทำให้การส่งผ่านข้อมูลไม่มีปัญหาแม้ว่าวงจรเชื่อมโยงบางวงจรใช้งานไม่ได้ การเรียนรู้และการกำหนดเส้นทางใหม่โดยอัตโนมัติ จะต้องอาศัยโปรโตคอลที่ทำหน้าที่พิเศษในด้านการพิจารณาเส้นทางในการส่งผ่านข้อมูลเพื่อติดต่อกันระหว่างเราเตอร์ นั้นก็คือมีโปรโตคอลการกาเส้นทาง เข้ามาช่วยเหลือตัว เราติ้ง โปรโตคอลจะทำหน้าที่พิจารณาเส้นทางและแลกเปลี่ยนสถานะของเส้นทางระหว่าง เราเตอร์ ด้วยกันและปรับปรุงข้อมูลเมื่อเกิดการเปลี่ยนแปลงในเครือข่าย

2.2 การเลือกทางเดินที่สั้นที่สุด (Shortest Path Routing)

การเลือกทางเดินที่สั้นที่สุดเป็นวิธีการที่ถูกนำไปใช้มากที่สุดแบบหนึ่ง หลักการทำงานเริ่มต้นด้วยการสร้างรูปกราฟของระบบเครือข่ายย่อย โดยให้แต่ละโหนดในรูปกราฟแทนเราเตอร์แต่ละตัวในเครือข่าย และให้เส้นเชื่อมโหนด (arc) แทนสายสื่อสารที่เชื่อมต่อระหว่างเราเตอร์ การเลือกเส้นทางเดินระหว่างเราเตอร์คู่หนึ่งทำได้โดยการค้นหาเส้นทางที่สั้นที่สุดในรูปกราฟ

นิยามของคำว่าเส้นทางที่สั้นที่สุดอาจมีได้หลายความหมาย เช่นการใช้จำนวนครั้งของการรับส่งข้อมูลหรือจำนวนเส้นเชื่อมในรูปกราฟเป็นหลักพิจารณา เส้นทาง ABCF และ ABEF ในรูปที่ 2.3 จะถือว่ายาวเท่ากัน แต่ถ้าใช้ระยะทาง (เช่นเป็นกิโลเมตร) มาพิจารณาแล้ว เส้นทาง ABEF ย่อมสั้นกว่า



รูปที่ 2.3 แสดงอัลกอลิทึมการหาเส้นทางที่สั้นที่สุด

อีกวิธีการหนึ่งที่สามารถนำมาใช้แทนการนับจำนวนเส้นเชื่อมหรือการวัดระยะทาง คือการกำหนดให้ตัวเลขบนเส้นเชื่อมแต่ละเส้นเป็นตัวเลขที่บอกจำนวนแพ็กเก็ตที่รอการจัดส่งและเวลารอคอย โดยเฉลี่ยที่คำนวณมาจากการรับ-ส่งแพ็กเก็ตมาตรฐานด้วยวิธีการนี้เส้นทางที่สั้นที่สุดจึงหมายถึงเส้นทางที่ส่งข้อมูลได้เร็วที่สุดวิธีการที่นำมาใช้กับกรณีทั่วไปนั้นจะกำหนดให้ตัวเลขบนเส้นเชื่อมคือผลลัพธ์ที่ได้จากการคำนวณโดยมีระยะทาง ความเร็วในการส่งข้อมูล ปริมาณข้อมูลโดยเฉลี่ย ค่าใช้จ่าย ค่าเฉลี่ยมาตรฐานของจำนวนแพ็กเก็ตที่รอการจัดส่ง ระยะเวลารอคอย และอื่น ๆ เป็นตัวประกอบ การเปลี่ยนค่าความสำคัญของตัวประกอบเหล่านี้ ทำให้แต่ละเราเตอร์สามารถใช้อัลกอริทึมเดียวกันในการคำนวณ แต่ให้ความสำคัญกับตัวประกอบไม่เหมือนกันได้

Dijkstra (1959) ได้นำเสนอ อัลกอริทึมสำหรับการค้นหาเส้นทางที่สั้นที่สุด ระหว่างจุดสองจุด ดังตัวอย่างในรูป 2.3 ต้องการหาเส้นทางที่สั้นที่สุดจากจุด A ไปยังจุด D เริ่มด้วยการระบายสีทึบที่จุด A (เพื่อบอกให้ทราบว่า ได้พิจารณาจุดนี้แล้ว) หาระยะทางของแต่ละจุดที่มีเส้นเชื่อมมาที่จุด A แล้วใส่ป้ายบอกระยะทางและโหนด กำกับเส้นเชื่อมเหล่านั้น คือ ใส่ (2,A) ไว้ที่จุด B และใส่ (6,A) ไว้ที่จุด G แล้วเลือกเส้นทางที่สั้นที่สุดจากตัวเลขทั้งหมดที่หาได้ ซึ่งก็คือจุด B ให้ระบายสีทึบที่จุด B แล้วใช้วิธีการเดิม จะได้ (4,B) ที่จุด E , (9,B) ที่จุด C, และของเดิม (6,A) ที่จุด G เลือกจุด E เป็นจุดต่อไป เพราะมีค่าต่ำที่สุด ระบายสีทึบที่จุด E แล้วใช้วิธีการเดิม จะได้ (5,E) ที่จุด G ซึ่งใช้แทน (6,A) เพราะมีค่าต่ำกว่า, (6,E)

ที่จุด F , และของเดิม (9,B) ที่จุด C เลือกจุด G เป็นจุดต่อไป ระบายสีทึบที่จุด G แล้วใช้วิธีการเดิม จะได้ (9,G) ที่จุด H, ของเดิม (6,E) ที่จุด F , และของเดิม (9,B) ที่จุด C เลือกจุด F เป็นจุดต่อไป ระบายสีทึบที่จุด F แล้วใช้วิธีการเดิม จะได้ (8,F) ที่จุด H ซึ่งใช้แทน (9,G) เพราะมีค่าต่ำกว่า , และของเดิม (9,B) ที่จุด C เลือกจุด H เป็นจุดต่อไป ให้ระบายสีทึบที่จุด H แล้วใช้วิธีการเดิมจะได้ (10,H) ที่จุด D , และของเดิม (9,B) ที่จุด C แม้ว่าจะได้เส้นทางมาถึงจุดปลายทางแล้วก็ตามแต่ค่า (9,B) ที่จุด C มีค่าต่ำกว่าจึงต้องคำนวณต่อไปโดยเลือกจุด C เป็นจุดต่อไป ระบายสีทึบที่จุด C แล้วใช้วิธีการเดิม จะได้ (12,C) ที่จุด D แต่ค่าเดิมคือ (10,H) มีค่าต่ำกว่า จึงใช้ค่าเดิม ผลลัพธ์ที่ได้ คือเส้นทาง ABEFHD มีความยาว 10 หน่วย เป็นเส้นทางที่สั้นที่สุด

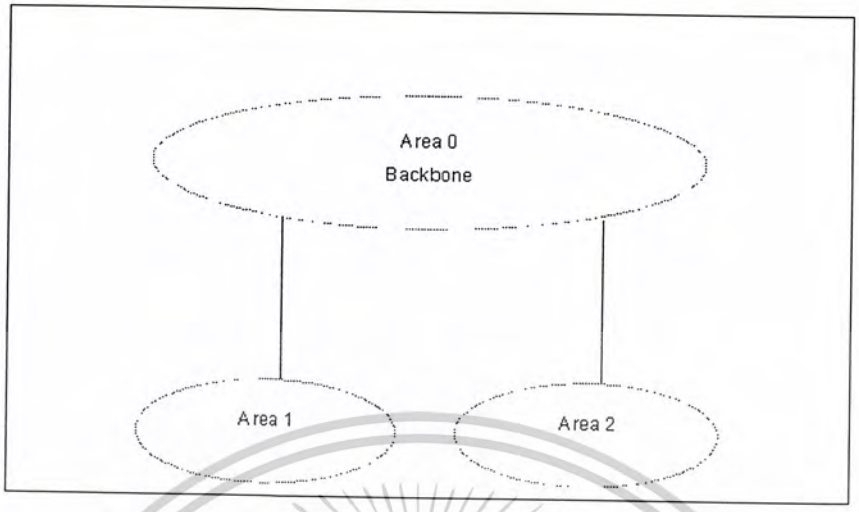
อัลกอริทึมในรูป 2.3 แสดงรายละเอียดการทำงานข้างบนนี้ สิ่งที่แตกต่างกันคือ ในอัลกอริทึมจะทำงานย้อนจากจุด D กลับไปหาจุด A เนื่องจากรูปกราฟที่ใช้เป็นกราฟแบบไม่กำหนดทิศทาง ดังนั้นการพิจารณาจากหน้าไปหลังหรือจากหลังไปหน้าย่อมให้ผลลัพธ์เหมือนกัน ถ้ารูปกราฟมีเส้นทางที่สั้นที่สุดหลายเส้นทางการค้นหาอาจให้ผลลัพธ์ต่างกัน อย่างไรก็ตาม ทุกเส้นทางจะต้องมีค่าเท่ากัน การเลือกใช้เส้นทางใดก็จะให้ผลเหมือนกันคือ เลือกใช้เส้นทางที่สั้นที่สุด

2.2.1 โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์

ในปี 1988 อินเทอร์เน็ตเอ็นจิเนียริงทาร์กฟอร์ท (IETF)ได้ประกาศ โปรโตคอลใหม่แทนที่ เราติ้งอินฟอร์เมชันโปรโตคอล ชื่อ โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์ โดย โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์ จัดอยู่ในกลุ่ม อินทีเรียเกตเวย์โปรโตคอล (Interior Gateway Protocol) นั่นคือเป็น โปรโตคอล ที่ใช้ใน ออโตโนมัสซีสเท็มทุกขนาด และใช้หลักการ ลิงค์สเททอังกอริทึม (Link State Algorithm) ในการหาเส้นทางที่ดีที่สุดโดยที่ โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์ จะสนับสนุน

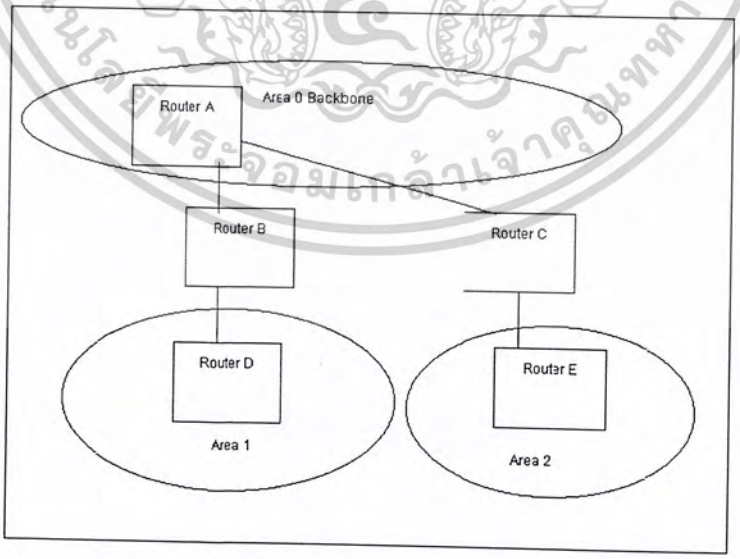
1. ทแรฟฟิคสปลิททิงอะครอสมัลติเพิลพาร์ท (Traffic Splitting across multiple path)
2. เราติงเบสออนไทพ์ออฟเซอร์วิส (Routing Base on type of service)
3. ออเท็นทิเคชันฟอร์เราติ้งอัพเดตแมสเสจ (Authentication for routing update message)

โดย โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์ จัดเป็น นอนโพรไพรอะเทอรี (non-proprietary) เนื่องจากมีการกำหนดเป็นมาตรฐาน โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์ โมเดล (Model) ใน โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์ จะมอง เน็ตเวิร์ค เป็น แอเรีย (Area) โดยที่แต่ละ แอเรีย จะถูกกำหนดหมายเลข พิจารณาจากรูปที่ 2.4 แสดงการแบ่ง แอเรีย ของ โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์ โมเดล



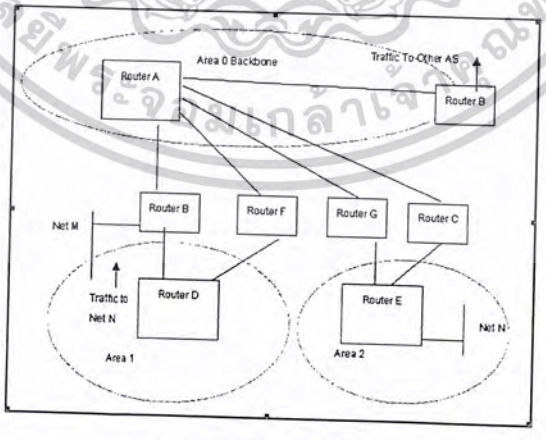
รูปที่ 2.4 แสดงการแบ่ง แอเรีย ของ โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์โมเดล

โดยกำหนดให้ แอเรีย 0 เป็น แบ็คโบน (Backbone) ทำหน้าที่ในการเชื่อมเข้ากับทุก ๆ แอเรีย โดยที่ เราเตอร์ ในแต่ละ แอเรีย จะทำหน้าที่เก็บข้อมูลเฉพาะใน แอเรีย ตัวเองเท่านั้น เมื่อ เน็ตเวิร์ค ใน แอเรีย มีการเปลี่ยนแปลงเช่น ลิงค์ดาวน์ (Link Down) ก็จะทำการส่งข้อมูลให้ทราบทั้ง แอเรีย โดยใช้ ฟลัดดิ้งอัลกอริทึม (Flooding Algorithm) หลักการก็คือเมื่อมีการเปลี่ยนแปลง ของ เน็ตเวิร์ค ส่งมาที่ เราเตอร์ แล้วจะทำการ คอพพี (Copy) ข้อมูลเดิมส่งไปให้ เราเตอร์ ตัวที่ต่อกับมันยกเว้น ลิงค์ ที่ส่ง ข้อมูลมาให้ทำการรับทราบข้อมูลได้เร็วกว่า เราติ้งอินฟอร์เมชันโปรโตคอล ใน ออโตโนมัสซีสเท็ม เดียวกัน



รูปที่ 2.5 แสดง เวอร์ชัวร์เราท์ (Virtual Route) ใน ออโตโนมัสซีสเท็ม

ข้อดีของการแบ่ง ออโตโนมัสซีสเท็ม ออกเป็น แอเรีย ก็คือในระบบ เน็ตเวิร์ค ที่มีความซับซ้อนการแบ่งเป็น แอเรีย จะช่วยมอง ภาพ เน็ตเวิร์ค ให้ซับซ้อนน้อยลงกล่าวคือ เราเตอร์ แต่ละตัวจะทำหน้าที่รับผิดชอบในการเก็บข้อมูลเฉพาะใน แอเรีย ตัวเองและ แบ็คโบน จะทำการเก็บข้อมูลของทุก เราเตอร์ ในแต่ละ แอเรีย และ เราเตอร์ ที่ไม่อยู่ใน แอเรีย พิจารณาจากรูปแสดงการเชื่อมต่อ เราเตอร์ ภายใน ออโตโนมัสซีสเท็มโดยกำหนดให้ แบ็คโบน แอเรีย ประกอบด้วย เราเตอร์ A,B,C และ แอเรีย 1 ประกอบด้วย เราเตอร์ B,D แอเรีย 2 ประกอบด้วย เราเตอร์ C,E และเรียก เราเตอร์ B,C ว่าเป็น Border เราเตอร์ และ กำหนดให้ แอเรีย 1 และ แอเรีย 2 ว่าเป็น Stub เน็ตเวิร์ค โดยที่ เราเตอร์ B จะทราบข้อมูลทั้งหมดของ แอเรีย 1 และ แบ็คโบน โดยที่ เราเตอร์ C จะทราบข้อมูลทั้งหมดของ แอเรีย 2 และ แบ็คโบน จากรูปกรณีส่งข้อมูลจาก เน็ตเวิร์ค M ไปที่ เน็ตเวิร์ค N จะส่งผ่านเส้นทาง B-A-C-E และในกรณีที่ส่ง เราเตอร์ A เสีย จะส่งผ่านเส้นทาง B-D-G-G-E การสร้าง ดาต้าเบส (Database) การหาเส้นทางในแต่ละ แอเรีย การแลกเปลี่ยนข้อมูลด้าน คอนเนคชัน (Connection) และ เมทริกซ์ (Metric) ในแต่ละ เราเตอร์ จะกระทำโดยใช้ เฮลโลแมสเสจ (Hello Message) โดยจะทำการกำหนดเวลาในการแจ้งว่าตัวเองยังสามารถทำงานได้อยู่ (Alive) การกำหนด เดซซิกเนท (Designated) เราเตอร์ ทำการหน้าที่เป็นตัวกลางในการ อัพเดท ข้อมูลใน เราเตอร์ ตัวที่อยู่ใกล้ที่สุด พิจารณาจากรูปที่ 5 แสดงการ อัพเดท ข้อมูลใน เดซซิกเนท เราเตอร์ โดยที่ เราเตอร์ B จะไม่ทำการเชื่อมต่อเข้ากับ เราเตอร์ C,D เนื่องจากมีข้อมูลชุดเดียวกับที่อยู่ใน เราเตอร์ A สร้าง ดาต้าเบสการหาเส้นทาง เช่นกรณีที่ เราเตอร์ B down และ ทำการ Restart จะทำการเริ่มต้นสร้าง ดาต้าเบสการหาเส้นทาง โดยขออัพเดทดาต้าเบสการหาเส้นทาง จาก เราเตอร์ A



รูปที่ 2.6 แสดงการ อัพเดท ข้อมูลใน เดซซิกเนท เราเตอร์

2.2.2 แมสเสจไทพ์ (Message Type)

เราทำการแบ่งแมสเสจที่ใช้ในการอัพเดทดาต้าเบสออกเป็น 5 ชนิดได้แก่ 1. Hello ใช้ในการเลือก เดซิกเนทเราเตอร์ สำหรับ Multi-access เน็ตเวิร์ค 2. ดาต้าเบสเดสคริปชันใช้ในตอนเริ่มต้นเพื่อแลกเปลี่ยนข้อมูลเพื่อให้เราเตอร์สามารหาว่าข้อมูลส่วนใดหายไปจากดาต้าเบส 3. ลิงค์สเตจรีเควส (Link State Request) ใช้ในการสอบถามข้อมูลสำหรับเราเตอร์ที่หายไปจากดาต้าเบส 4. ลิงค์สเตจอัพเดต (Link State Update) ใช้สำหรับตอบลิงค์สเตจรีเควสและรายงานการเปลี่ยนแปลงในเน็ตเวิร์ค 5. ลิงค์สเตจแอ็ค (Link State ACK) ใช้ในการ คันเฟิร์ม (Confirm) ลิงค์สเตจอัพเดต

2.2.3 โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์ แมสเสจ เฮดเดอร์ (Header)

แมสเสจเฮดเดอร์ ของ โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์ แบ่งเป็น 5 ชนิดได้แก่ 1. เฮลโล 2. ดาต้าเบสเดสคริปชัน (Description) 3. ลิงค์สเตจรีเควส 4. ลิงค์สเตจอัพเดต 5. ลิงค์สเตจแอ็ค พิจารณาจากรูปแสดง แพ็คเกจ เฮดเดอร์ ใน โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์

เวอร์ชัน (Version)	ไทพ์ (Type)	แมสเสจ เลนจ์ (Message Length)
ไอพีแอดเดรสของเราเตอร์ ที่ส่งข้อมูลมา		
หมายเลขแอเรีย		
เช็คซัม (Checksum)	ออเทนทิเคชันไทพ์ (Authentication Type)	
ออเทนทิเคชัน		
ออเทนทิเคชัน		

ตารางที่ 2.2 แสดงแพ็คเกจ เฮดเดอร์ ใน โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์

ใน โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์เราติ้งอินฟอร์เมชันจะถูกส่งในลิงค์สเตจอัพเดตไทพ์ (Link State Update Type) โดยข้อมูลในแมสเสจจะถูกเรียกว่าแอดเวอไทซ์เมนท์ (Advertisement) โดยแบ่งไทพ์ของแอดเวอไทซ์เมนท์ ตามตาราง

ไทพ์ ของ แอดเวอไทซ์เมนท์	เดสคริปชัน
เราเตอร์ ลิงค์	แสดงสถานะของ อินเตอร์เฟสของแต่ละ เราเตอร์ โดยที่ เราเตอร์ แต่ละตัวเป็นตัวให้ข้อมูล
เน็ตเวิร์ค ลิงค์	แสดงรายชื่อของ เราเตอร์ ที่ต่ออยู่ใน เน็ตเวิร์ค โดยที่ เดซซิกเนท เราเตอร์ เป็นตัวให้ข้อมูล
ซัมมารีลิงค์ทูอะเน็ตเวิร์ค (Summary Link to a Network)	แสดงข้อมูลเส้นทางในการส่งข้อมูลสู่ภายนอก แต่อยู่ใน ออโตโนมัสซีสเท็ม ให้ข้อมูลโดย บอเดอร์เราเตอร์ (Border router)
ซัมมารีลิงค์ทูอะบาวน์ดารีเราเตอร์ (Summary Link to a Boundary Router)	แสดงเส้นทางในการส่งข้อมูลจาก ออโตโนมัสซีสเท็ม ไปที่ บอเดอร์เราเตอร์
ออโตโนมัสซีสเท็ม เอ็กซ์เทอนอล ลิงค์ (External Link)	แสดงข้อมูลเส้นทางในการส่งข้อมูลสู่ภายนอก และอยู่นอก ออโตโนมัสซีสเท็ม ให้ข้อมูลโดย บอเดอร์เราเตอร์

ตารางที่ 2.3 แสดงการหาเส้นทางแบบ OSPF

กล่าวโดยสรุป โอเพ็นชอร์ทเท็ดพาร์ทเฟริสท์ มีเฟียเจอร์ (Feature) ที่สำคัญคือเนื่องจากเราเตอร์แต่ละตัวทราบข้อมูลทำการเมื่อมีการเปลี่ยนทางเน็ตเวิร์คเช่นลิงค์ดาวน์จะทำการอัพเดตข้อมูลอย่างรวดเร็วและไม่เปลือง โอเวอร์เฮดในการอัพเดตเราติ้งอินฟอร์เมชัน

2.3 การเลือกเส้นทางแบบตารางระยะทาง

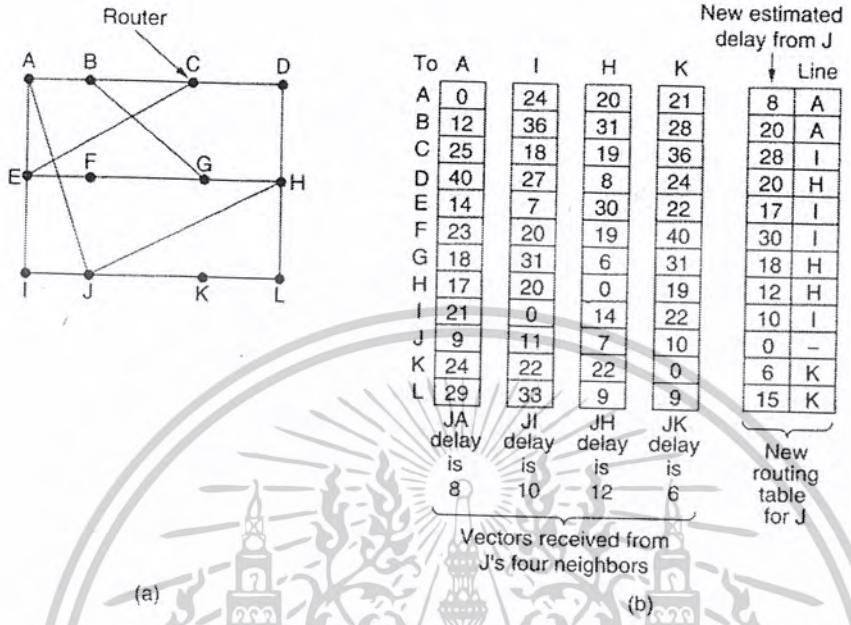
ระบบเครือข่ายรุ่นใหม่เลือกที่จะใช้ อัลกอริทึมเลือก เส้นทางแบบพลวัตรมากกว่าแบบสถิตย์ อัลกอริทึมเลือกเส้นทางแบบตารางระยะทาง (Distance vector routing) เป็นอัลกอริทึมพลวัตรแบบหนึ่งที่ได้รับความนิยมในการนำไปใช้เป็นอย่างมาก อัลกอริทึมนี้ยังเป็นที่รู้จักในนาม อัลกอริทึม เบลมาฟอร์ซ (Bellman-Ford) หรือ อัลกอริทึม ฟอร์ด-เฟอคูสัน (Ford-Fulkerson) ตามชื่อของนักวิจัยผู้คิดค้นขึ้นมา (เบลมาฟอร์ซ ในปี ค.ศ. 1957; และ ฟอร์ด-เฟอคูสัน ในปี ค.ศ. 1962) เดิมทีเดียวเครือข่าย แอดวานซ์ รีเสริร์ช โปรเจค เอเยนซี เน็ตเวิร์ค (ARPANET) ใช้อัลกอริทึมนี้ในการเลือกเส้นทาง เช่นเดียวกับระบบอินเทอร์เน็ต ที่นำไปใช้ในชื่อ เราติ้งอินฟอร์เมชันโปรโตคอล รวมทั้งถูกนำไปใช้ใน DECnet และ โน

เวลล์ ไอพีเอ็กซ์ (Novell's IPX) รุ่นแรก ๆ ในปัจจุบันโปรโตคอล แอปเปิลทอล์ก (AppleTalk) และ ซิสโก (Cisco) ก็ได้นำอัลกอริทึมนี้ไปปรับปรุงใช้งานเช่นกัน

เราเตอร์ที่ใช้อัลกอริทึมเลือกเส้นทางแบบตารางระยะทางจะต้องสร้างตารางเก็บข้อมูลซึ่งจะบอกระยะทางและเส้นทางที่ดีที่สุดในการส่งข้อมูลไปยังเราเตอร์ต่าง ๆ ข้อมูลในตารางนี้จะมีการปรับปรุงอยู่ เสอมโดยการแลกเปลี่ยนข่าวสารกับเราเตอร์ที่อยู่ติดกัน จำนวนข้อมูลในตารางจะเท่ากับจำนวนเราเตอร์ทั้งหมดในเครือข่าย (ไม่นับตัวเอง) ข้อมูลแต่ละตัวประกอบด้วยเส้นทางสำหรับส่งข้อมูลไปยังจุดหมาย และระยะทางหรือเวลาโดยประมาณที่จะต้องใช้ในเส้นทางนั้น มาตรที่ใช้วัดได้แก่จำนวนเราเตอร์ที่อยู่ในเส้นทาง เวลารอคอยตลอดเส้นทาง จำนวนแพ็กเก็ตที่รอการนำส่ง เป็นต้น ในที่นี้จะใช้คำเป็นกลาง ๆ ว่า "ระยะทาง" เท่านั้น

สมมติว่าใช้เวลารอคอย (เป็นผลรวมของเวลารอคอยการนำส่ง กับเวลาที่ใช้ในการส่งข้อมูล) เป็นมาตรในการวัดระยะทาง ในทุก ๆ T มิลลิวินาทีแต่ละเราเตอร์จะต้องส่งตารางข้อมูลที่ตนเองมีอยู่ไปให้เราเตอร์ข้างเคียงทุกตัว ในทางกลับกันเราเตอร์แต่ละตัวจะได้รับตารางข้อมูลจากเราเตอร์ข้างเคียง ข้อมูลที่ได้รับจากเราเตอร์ X จะบอกให้ทราบว่าจะต้องใช้เวลา Xi เพื่อส่งข้อมูลไปยังเราเตอร์ I ถ้าเวลาที่ใช้ส่งข้อมูลไปเราเตอร์ X เป็น m มิลลิวินาที ดังนั้นเวลาที่จะต้องใช้ในการส่งข้อมูลไปยังเราเตอร์ I จะเป็น Xi + m มิลลิวินาที โดยการส่งข้อมูลผ่านเราเตอร์ X เมื่อใช้วิธีการนี้ สำหรับทุกตารางที่ได้รับเราเตอร์แต่ละตัวก็จะทราบเวลาที่ต้องใช้ในการส่งข้อมูลไปยังเราเตอร์ทุกตัวในระบบและสามารถเลือกเส้นทางเดินที่ใช้ระยะเวลาน้อยที่สุดในการส่งข้อมูลไปยังจุดหมายที่ต้องการได้ ซึ่งตัวเลขเหล่านี้ก็จะถูกนำไปเก็บไว้ในตารางของตนเอง สังเกตว่าข้อมูลเก่าในตารางไม่ได้นำมาใช้ในการคำนวณเลย

กระบวนการคำนวณระยะทางนี้ แสดงให้เห็นในรูป 2.7 ในส่วน (a) เป็นรูปโครงสร้างเครือข่ายย่อย ตารางสี่อันดับแรกในส่วน (b) เป็นตารางแสดงระยะเวลาที่ใช้ของเราเตอร์ A,I,H และ K ที่ส่งมาให้เราเตอร์ J ตารางแรกระบุว่าเราเตอร์ A ใช้เวลา 12 มิลลิวินาที ในการส่งข้อมูลไปยังเราเตอร์ B และใช้เวลา 25 มิลลิวินาที ในการส่งข้อมูลไปยังเราเตอร์ C ฯลฯ ให้เวลาในการส่งข้อมูลจาก J ไปยังเราเตอร์ A,I,H และ K เป็น 8,10,12 และ 6 มิลลิวินาที ตามลำดับ

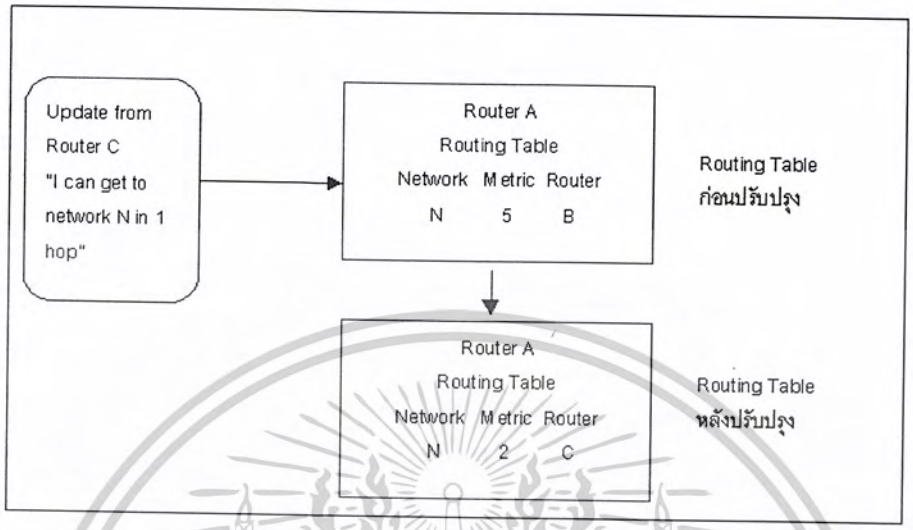


รูปที่ 2.7 แสดงการเลือกเส้นทางตามระยะทาง

ถ้าเราเตอร์ J ต้องการคำนวณเวลาส่งข้อมูลไปเราเตอร์ G เส้นทางแรกที่จะต้องส่งข้อมูลผ่านเราเตอร์ A การส่งข้อมูลจาก J ไปยัง A ใช้เวลา 8 มิลลิวินาที และจาก A ไปยัง G ใช้เวลา 18 มิลลิวินาที เวลารวมเป็น 26 มิลลิวินาที ในทำนองเดียวกันเมื่อส่งข้อมูลผ่าน I,H และ K จะใช้เวลารวม 41(31+10) , 18(6+12) , และ 37(31+6) มิลลิวินาทีตามลำดับ เวลาที่ดีที่สุดคือ 18 มิลลิวินาที ดังนั้นเราเตอร์ J จะบันทึกข้อมูลในตารางของตนเองว่า การส่งข้อมูลไปยังเราเตอร์ G ใช้เวลา 18 มิลลิวินาที โดยการส่งข้อมูลผ่านเราเตอร์ H ผลจากการคำนวณดังที่ได้แสดงไว้ในตารางสุดท้ายในรูปที่ 2.7

2.3.1 เราติ้งอินฟอร์เมชันโปรโตคอล

เราติ้งอินฟอร์เมชันโปรโตคอล เป็น โปรโตคอลการหาเส้นทาง ที่ใช้หลักการ ดิสแทนเวคเตอร์ อัลกอริทึม ในการหาเส้นทางโดยใช้หลัก การง่าย ๆ ได้แก่ ทุก ๆ ฮอพ ที่ส่งข้อมูลผ่านจะประกอบด้วย Cost ค่าหนึ่งและจะทำการรวมค่า Cost ของทุก ๆ ฮอพ ที่ส่งผ่านในทุก ๆ เส้นทางที่มีเป็นไปได้และจะเลือกเส้นทางที่ดีที่สุดจาก เส้นทางที่มี Cost ต่ำที่สุด โดยปกติ ใน เราติ้งอินฟอร์เมชันโปรโตคอล จะมีการ อัพเดท ตารางการหาเส้นทาง โดยจะทำการส่งข้อมูลด้าน เน็ตเวิร์ค ไปให้ neighbor ทุก 30 วินาที โดยวิธีการดังกล่าวจะเรียกว่า advertising routes พิจารณาจากรูปแสดงตัวอย่างการ อัพเดท ตารางการหาเส้นทาง



รูปที่ 2.8 แสดงการปรับปรุง ตารางการหาเส้นทาง โดย เราติ้งอินฟอร์เมชันโปรโตคอล

จากรูป สมมุติว่าการส่งข้อมูลจาก เราเตอร์ A ไปที่ เน็ตเวิร์ค N จะผ่านไปที่ เราเตอร์ B เมื่อมีการส่งอัพเดท ตารางหาเส้นทาง แล้วพบว่าในกรณีที่ส่งข้อมูลผ่าน เราเตอร์ C จะใช้ Cost ต่ำกว่า ส่งผ่าน เราเตอร์ B โดยจะทำการ ส่งข้อมูลผ่าน เราเตอร์ C ในการส่งข้อมูลครั้งต่อไป

2.3.1.1 เราติ้งอินฟอร์เมชันโปรโตคอล แมสเสจโปรโตคอล

เพื่อให้เข้าใจถึง โครงสร้างของ เฮดเดอร์ แพ็คเกจ ของ พิจารณาได้จากรูป

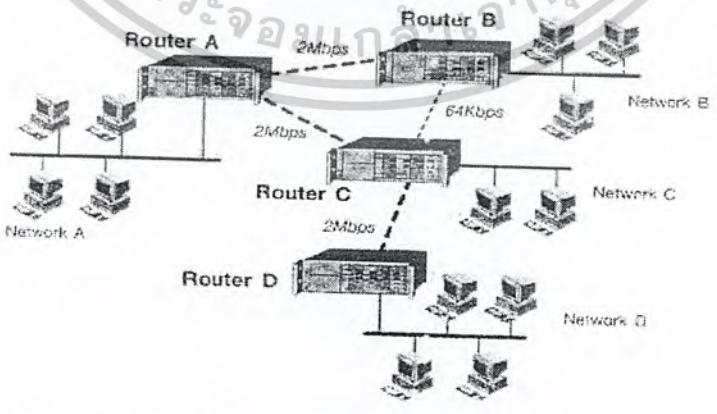
Command	Version	Zero
Address family ident		Zero
IP Address		
Zero		
Zero		
เมทริกซ์		

ตารางที่ 2.4 แสดง เฮดเดอร์ แพ็คเกจ ของ เราติ้งอินฟอร์เมชันโปรโตคอล

Command: เราจะทำการแยก Request เฮดเดอร์ และ Respond เฮดเดอร์ โดยถ้า Command = 1 แสดงว่าเป็น Request เฮดเดอร์ และถ้า Command = 2 แสดงว่าเป็น Respond เฮดเดอร์ หมายเหตุ Request เฮดเดอร์ จะถูกใช้เมื่อต้องการขอให้มีการส่งข้อมูลที่ อัพเดท มาให้ Respond เฮดเดอร์ จะถูกใช้เมื่อต้องการมีการส่งข้อมูลที่ อัพเดท มาให้ IP Address:เป็น Address ของ เน็กท์ ฮอพ (Next Hop)

2.3.2 โปรโตคอล อินทีเรียเกตเวย์โปรโตคอลต้นหาเส้นทาง

เครือข่ายอินเทอร์เน็ตที่เป็นมาในอดีตอุปกรณ์เราเตอร์ อุปกรณ์ที่ใช้เชื่อมต่อในเครือข่ายจะใช้โปรโตคอลอินทีเรียเกตเวย์โปรโตคอลต้นหาเส้นทางนี้ในการติดต่อแลกเปลี่ยนข้อมูล ซึ่งเส้นทางการส่งผ่านข้อมูลกับตัวโปรโตคอลอินทีเรียเกตเวย์โปรโตคอลต้นหาเส้นทาง จะใช้วิธีพิจารณาเส้นทางข้อมูลแบบ ดิสแทนเวคเตอร์โปรโตคอล แต่มีการพิจารณาพารามิเตอร์หรือปัจจัยอื่นอีกหลายอย่างประกอบกันเช่น ขนาดของช่องสัญญาณที่เชื่อมต่ออยู่ (แบนด์วิดท์) จำนวน ดีเลย์ (delay) ที่เกิดในช่องสัญญาณนั้น จำนวน load หรือความหนาแน่นของช่องสัญญาณ ความเชื่อถือได้ของช่องสัญญาณรีไลเอเบิล (Reliable) ว่าจะไม่เกิดปัญหาวงจรล้มบ่อย ๆ ครั้ง เมื่อมีการพิจารณาปัจจัยอื่นประกอบการตัดสินใจในการส่งผ่านข้อมูลแล้ว จะเห็นว่าโปรโตคอ อินทีเรียเกตเวย์โปรโตคอลค้นหาเส้นทาง มีความาสามารถดีกว่าและยืดหยุ่นกว่าโปรโตคอลเราติ้งอินฟอร์เมชันโปรโตคอล สิ่งที่สำคัญคือโปรโตคอลอินทีเรียเกตเวย์โปรโตคอลค้นหาเส้นทางได้รับการพัฒนาขึ้นโดยบริษัทซิสโก ซึ่งเป็นผู้ผลิตอุปกรณ์เราเตอร์รายใหญ่และเป็นที่ได้รับความนิยมมากในการนำไปใช้เชื่อมเครือข่ายอินเทอร์เน็ต ทำให้โปรโตคอลอินทีเรียเกตเวย์โปรโตคอลต้นหาเส้นทาง ได้รับความนิยมและมีการใช้งานมากตามไปด้วย ตัวอย่างการใช้งานเช่น เครือข่ายตามรูป



รูปที่ 2.9 แสดงเน็ตเวิร์คแบบ เราติ้งอินฟอร์เมชันโปรโตคอล

ถ้าเครือข่ายในเครือข่าย B ต้องการส่งข้อมูลไปยังเครือข่าย C กรณีที่ใช้โปรโตคอล เราติ้ง อินฟอร์เมชันโปรโตคอล ในการพิจารณาเส้นทางในการส่งผ่านข้อมูล ก็จะได้เส้นทางจากเครือข่าย B ผ่าน เราเตอร์ B และผ่านไปยัง เราเตอร์ C ถูกเลือกใช้เพราะมีจำนวน ฮอพ ที่น้อยกว่าแต่ในกรณีทีใช้โปรโตคอล อินทีเรียเกตเวย์โปรโตคอลต้นหาเส้นทาง พิจารณาจะได้เส้นทางจากเครือข่าย B ไป เราเตอร์ A และผ่านไปยัง เราเตอร์ C ถูกเลือกใช้ เพราะพิจารณาแล้วข้อมูลจะถึงปลายทางได้ดีที่สุดเนื่องจากวงจรที่เชื่อมโยงระหว่าง เราเตอร์ มีขนาดช่องสัญญาณใหญ่กว่าโปรโตคอล อินทีเรียเกตเวย์โปรโตคอลต้นหาเส้นทาง จะมีการ Broadcast สถานะของเครือข่ายและข้อมูล ตารางหาเส้นทาง ของ เราเตอร์ ทุก 90 วินาที

2.4 การเลือกเส้นทางแบบพิจารณาสถานการณ์เชื่อมต่อ (Link State Routing)

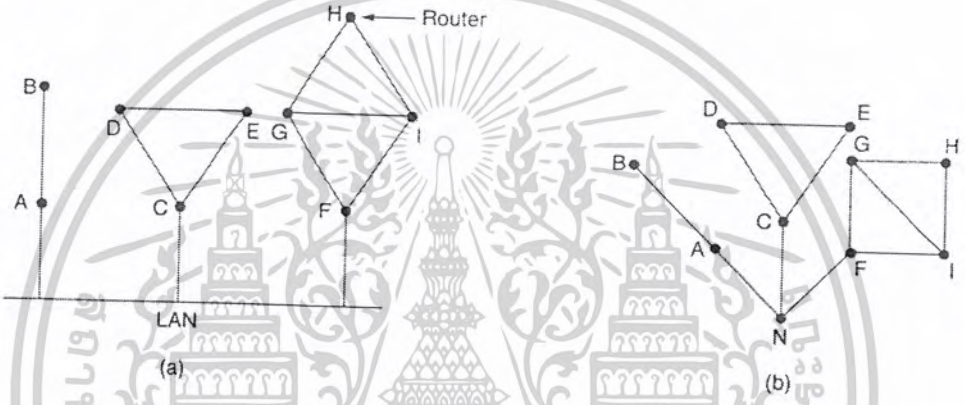
อัลกอริทึมเลือกเส้นทางแบบตารางระยะทางถูกนำไปใช้ใน แอดวานซ์ รีเสริร์ช โปรเจค เอเยนซี เน็ตเวิร์ค ตั้งแต่เริ่มต้นจนกระทั่งใน ค.ศ. 1979 จึงถูกทดแทนโดยอัลกอริทึมพลวัตรแบบพิจารณาสถานการณ์เชื่อมต่อ เนื่องจากสาเหตุสองข้อแรกคือ ข้อแรก แต่เดิมนั้นสายสื่อสารเกือบทั้งหมดมีความเร็วในการส่งข้อมูลที่ 56 กิโลบิตต่อวินาทีเหมือนกันหมด จึงไม่มีความจำเป็นที่จะต้องนำปัจจัยด้านความเร็วเข้าไปพิจารณาด้วย ต่อมาความเร็วในสายสื่อสารบางส่วนได้เพิ่มขึ้นหลายสิบเท่า บางส่วนก็เพิ่มความเร็วเป็นหลายร้อยเท่า ความแตกต่างกันอย่างมากนี้ทำให้ความเร็วกลายเป็นตัวประกอบที่สำคัญมากตัวหนึ่ง ข้อสอง อัลกอริทึมแบบเดิมนั้นใช้เวลาในการเรียนรู้เครือข่ายนานมากเกินไปโดยเฉพาะอย่างยิ่งถ้าเครือข่ายนั้นมีขนาดใหญ่มาก ดังนั้นอัลกอริทึมแบบพิจารณาสถานการณ์เชื่อมต่อจึงถูกนำมาใช้แทนแบบเดิม เราเตอร์แต่ละตัวเมื่อใช้อัลกอริทึมนี้จะต้อง

1. ทำความรู้จักกับเราเตอร์ข้างเคียงและรับรู้ที่อยู่บนเครือข่ายของเราเตอร์เหล่านั้น
2. คำนวณระยะทางเวลารอคอยหรือค่าใช้จ่ายในการติดต่อกับเราเตอร์ข้างเคียง
3. สร้างแพ็กเก็ตสำหรับส่งข้อมูลที่ตนเองรวบรวมมาได้
4. ส่งแพ็กเก็ตนี้ไปยังเราเตอร์ทุกตัว
5. คำนวณระยะทางที่สั้นที่สุดสำหรับการติดต่อไปยังแต่ละเราเตอร์

ผลที่เกิดขึ้นคือ ทุกเราเตอร์มองเห็นภาพโครงสร้างเครือข่ายและที่ได้รับข้อมูลที่จำเป็นต้นทราบขั้นต่อไปจะนำวิธีการ เช่น ดีจิกส์ตรา อัลกอรึทึม (Dijkstra's algolithm) มาใช้ในการคำนวณหาเส้นทางที่สั้นที่สุดจากเราเตอร์หนึ่งไปยังอีกเราเตอร์หนึ่งสำหรับเราเตอร์ทั้งหมดในเครือข่าย ต่อไปนี้เป็นการกล่าวถึงรายละเอียดของขั้นตอนการทำงานในแต่ละข้อ

2.4.1 การทำความรู้จักกับเราเตอร์ข้างเคียง

งานอย่างแรกเมื่อเราเตอร์ที่เพิ่งเริ่มเปิดให้บริการในเครือข่ายต้องทำคือ ทำความรู้จักกับเราเตอร์ข้างเคียงทั้งหมด ซึ่งทำได้โดยการส่งแพ็กเก็ตพิเศษ (เฮลโลแพ็กเกต) เพื่อขอทราบชื่อ(หมายถึงที่อยู่บนเครือข่าย หรือ หมายเลขเน็ตเวิร์ค) ของเราเตอร์เหล่านั้นและจะต้องเป็นชื่อที่ไม่ซ้ำกันเลย มิฉะนั้นจะทำให้ เกิดความสับสนในการส่งข้อมูล ชื่อของเราเตอร์ที่รวบรวมได้จะถูกนำไปสร้างเป็นรูปจำลองโครงสร้างของเครือข่าย

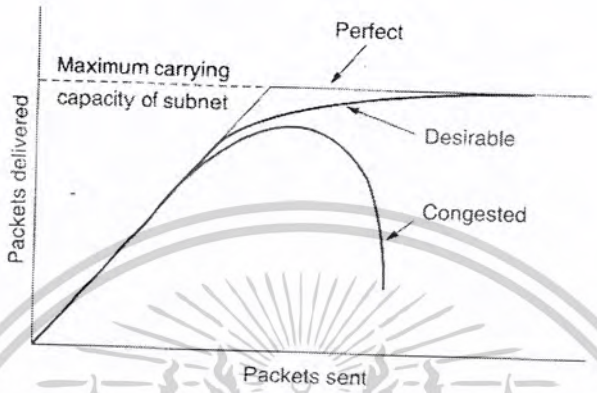


รูปที่ 2.10 การทำความรู้จักกับเราเตอร์ข้างเคียง

2.5 อัลกอริทึมควบคุมความคับคั่งข้อมูล (Congestion Control Algorithms)

ความคับคั่งของข้อมูล (Congestion) คือเหตุการณ์ที่เกิดขึ้นในบางส่วนของระบบเครือข่ายเมื่อมีจำนวนแพ็กเก็ตรอคอยการนำส่งมากเกินไป อันจะทำให้ประสิทธิภาพการทำงานโดยรวมของระบบลดลง กราฟในรูป 2.11 แสดงให้เห็นถึงประสิทธิภาพในการทำงานของระบบเมื่อเปรียบเทียบระหว่างจำนวนแพ็กเก็ตที่ถูกส่งเข้าสู่ระบบ (packets sent) กับจำนวนแพ็กเก็ตที่ส่งถึงตัวผู้รับแล้ว (packet delivered) ในภาวะปกติ โฮสต์ต่าง ๆ จะส่งแพ็กเก็ตเข้าไปในระบบ ซึ่งแพ็กเก็ตทั้งหมดก็จะส่งไปถึงผู้รับได้ (ยกเว้นบางแพ็กเก็ตที่อาจเสียหายหรือสูญหายระหว่างการนำส่ง) แสดงว่าจำนวนแพ็กเก็ตที่ถูกส่งนั้นเป็นอัตราส่วนที่พอเหมาะกับจำนวนแพ็กเก็ตที่ส่งถึงผู้รับ อย่างไรก็ตามเมื่ออัตราการส่งแพ็กเก็ตเข้าสู่ระบบเริ่มมีปริมาณมากขึ้น ๆ เราเตอร์เริ่มไม่สามารถจัดการนำส่งข้อมูลได้ทัน แพ็กเก็ตก็จะเกิดการสูญหายมากขึ้นและก็มีแนวโน้มที่จะแย่ลง ในที่สุดเราเตอร์จะไม่สามารถจัดการนำส่งข้อมูลได้อีก

ต่อไป สมรรถนะในการทำงานของระบบก็จะพังลงอย่างราบคาบ และจะไม่มีแพ็กเก็ตใดส่งถึงผู้รับได้เลย



รูปที่ 2.11 กราฟแสดงค่าความคับคั่งในการส่งข้อมูล

ความคับคั่งเกิดขึ้นได้จากหลายสาเหตุ เช่น แพ็กเก็ตจำนวนหนึ่งถูกส่งมาจากสายสื่อสารหลาย ๆ เส้นในทันทีทันใดพร้อม ๆ กัน และแพ็กเก็ตทั้งหมดจะต้องถูกส่งออกไปทางสายสื่อสารเส้นเดียวกันอีก เราเตอร์นั้นจะต้องสร้าง "แถวคอย (queue)" โดยกำหนดให้หน่วยความจำส่วนหนึ่งเป็นที่สำหรับเก็บแพ็กเก็ตที่พร้อมจะนำส่ง และถ้าหน่วยความจำไม่สามารถรองรับจำนวนแพ็กเก็ตทั้งหมดได้ก็จะทำให้แพ็กเก็ตจำนวนหนึ่งสูญหายไป การเพิ่มขนาดของหน่วยความจำสำหรับแถวคอย อาจช่วยแก้ปัญหานี้ได้ แต่ Nagle (1987) พบว่าถ้าให้เราเตอร์มีหน่วยความจำอย่างไม่จำกัดแล้ว ความคับคั่งจะยิ่งแย่ลงไปอีกเพราะผู้ส่งข้อมูลจะตั้งกำหนดเวลาไว้สำหรับการรอคอยแพ็กเก็ตตอบรับจากผู้รับข้อมูลเรียกว่า "ไทม์ เอาท์ (time out)" เมื่อหมดเวลาดังกล่าวผู้ส่งแพ็กเก็ตจะคิดว่าแพ็กเก็ตที่ส่งออกไปนั้นสูญหายไปแล้วดังนั้นจึงส่งแพ็กเก็ตสำเนา (dupicate packet) เข้ามาในระบบ อันที่จริงแล้วแพ็กเก็ตที่ส่งมาครั้งแรกไม่ได้สูญหายไปไหนแต่ยังคงอยู่ในแถวคอยของเราเตอร์ตัวใดตัวหนึ่ง การกระทำเช่นนี้จึงเป็นการเพิ่มปริมาณงานให้กับระบบโดยไม่จำเป็น

การใช้หน่วยประมวลผลที่ทำงานช้าก็เป็นอีกสาเหตุหนึ่งของความคับคั่ง ถ้าหน่วยประมวลผลของเราเตอร์ทำงานได้ช้าในการบริหารงานภายใน ได้แก่ การสร้างและการจัดการแถวคอย (queue buffer) และการปรับปรุงข้อมูลในตาราง (update table) เป็นต้น ก็จะทำให้แพ็กเก็ตที่รับเข้ามาเริ่มเกิดการสะสมในแถวคอยมากขึ้นเป็นลำดับแม้ว่าสายสื่อสารจะสามารถส่งข้อมูลได้เร็วมากก็ตาม การใช้สายสื่อสารที่มีความเร็วในการส่งต่ำ (low bandwidth line) ก็ทำให้เกิดความคับคั่งข้อมูลได้เช่นเดียวกัน ดังนั้นการปรับปรุงสายสื่อสารแต่ไม่เปลี่ยนหน่วยประมวลผล หรือเปลี่ยนหน่วยประมวลผลแต่ไม่

ปรับปรุงสายสื่อสารก็อาจจะช่วยได้เพียงเล็กน้อย ผลที่เกิดขึ้นเป็นเพียงการขยับจุดที่สร้างปัญหาไปอยู่ที่อื่นเท่านั้น จากที่กล่าวมานี้สามารถสรุปได้ว่าการปรับปรุงส่วนประกอบของระบบเพียงบางส่วนโดยไม่ปรับปรุงทั้งระบบนั้นไม่ได้เป็นการแก้ปัญหาแต่อย่างใด นั่นคือส่วนต่าง ๆ ในระบบยังคงทำงานไม่สอดคล้องกัน

ความคับคั่งของข้อมูลที่เกิดขึ้นในบางส่วนของระบบ มีแนวโน้มที่จะทำให้เกิดความคับคั่งขึ้นที่ส่วนอื่น ๆ ตามไปด้วยอันจะทำให้สถานะการณ์โดยรวมแย่ลงไปอีก ถ้าพื้นที่เก็บข้อมูลในแถวคอยของเราเตอร์เต็ม เราเตอร์จำเป็นต้องละทิ้งแพ็กเก็ตที่เพิ่งรับเข้ามา และเมื่อแพ็กเก็ตถูกทิ้งไปแล้ว เราเตอร์ผู้ส่งจะต้องส่งแพ็กเก็ตนั้นใหม่ บางกรณีอาจส่งซ้ำหลายครั้งจนกว่าจะได้รับแพ็กเก็ตตอบรับจากผู้รับข้อมูล ความคับคั่งของข้อมูลในเราเตอร์ทางฝั่งผู้รับนั้นจะบีบบังคับให้เราเตอร์ทางฝั่งผู้ส่งระงับการลบข้อมูลในแถวคอยทิ้ง ทำให้เกิดปฏิกิริยาลูกโซ่ต่อเราเตอร์ตัวอื่น ๆ ด้วย

การควบคุมความคับคั่งของข้อมูล (congestion control) กับการควบคุมการไหลของข้อมูล (flow control) มีความสัมพันธ์กันอย่างลึกซึ้ง แต่ก็สามารถที่จะชี้ให้เห็น ความแตกต่างระหว่างกันได้ การควบคุมความคับคั่งของข้อมูลนั้นจะมองภาพรวมของทั้งระบบที่จะต้อง ทำให้เกิดความมั่นใจว่าระบบเครือข่ายย่อยสามารถรองรับปริมาณข้อมูลหมุนเวียนภายในระบบได้ องค์ประกอบที่ต้องนำมาพิจารณานั้นครอบคลุมไปทั่วซึ่งจะเกี่ยวข้องกับ การทำงานของโฮสต์และของเราเตอร์ทั้งหมด วิธีการรับส่งข้อมูลของเราเตอร์และปัจจัยอื่น ๆ ที่มีผลกระทบต่อความสามารถในการส่งผ่านข้อมูลภายในระบบเครือข่ายย่อย

การควบคุมการไหลของข้อมูลจะพิจารณาในทางกลับกัน คือจะมองภาพการทำงานที่เกิดขึ้นระหว่างผู้ส่งและผู้รับซึ่งเป็นความสัมพันธ์แบบจุด-ต่อ-จุด (point-to-point) นั่นคือจะสร้างความมั่นใจได้ว่าผู้ส่งที่ทำการส่งข้อมูลได้อย่างรวดเร็วและต่อเนื่องนั้นจะต้องไม่ส่งข้อมูลเร็วเกินกว่าที่ผู้รับจะรับไว้ได้ทั้งหมด ทำให้การควบคุมการไหลของข้อมูลจะเกี่ยวข้องกับการโต้ตอบโดยตรงกับผู้รับเพื่อรายงานสถานะการทำงานให้ผู้ส่งข้อมูลทราบ

เพื่อดูข้อแตกต่างระหว่างหลักการทั้งสองอย่างนี้ พิจารณาตัวอย่างของระบบเครือข่ายที่ใช้สายเคเบิลใยแก้วที่สามารถส่งข้อมูลได้ที่ความเร็ว 1 พันล้านบิตต่อวินาที ในกรณีนี้แม้ว่าจะไม่เกิดความคับคั่งของข้อมูล แต่ก็ยังต้องการการควบคุมการไหลของข้อมูล เพื่อบังคับให้เครื่องซูเปอร์คอมพิวเตอร์หยุดส่งเป็นช่วง ๆ เพื่อเปิดโอกาสให้เครื่องคอมพิวเตอร์ส่วนบุคคลสามารถทำงานตอบได้ทัน อีกมุมมองหนึ่ง ให้พิจารณาระบบเครือข่ายที่ใช้วิธีส่งข้อมูลแบบ รับ-และ-ส่งต่อ (store-and-forward) ที่มีความเร็วในการส่งข้อมูล 1 พันล้านบิตต่อวินาทีและมีเครื่องคอมพิวเตอร์ขนาดใหญ่ 1000 เครื่อง คอมพิวเตอร์จำนวนหนึ่งพยายามส่งแฟ้มข้อมูลด้วยความเร็ว 0.1 ล้านบิตต่อวินาทีให้แก่อีกครึ่งหนึ่งที่

เหลือ ปัญหานี้จึงไม่เกี่ยวข้องกับผู้ส่งที่ส่งข้อมูลอย่างรวดเร็วไปให้กับผู้รับที่ทำงานได้ช้า แต่ว่าเป็นปัญหาเกี่ยวกับปริมาณข้อมูลการสื่อสารที่มีมากเกินกว่าที่ระบบจะบริหารได้

เหตุการณ์ที่ควบคุมความคับคั่งของข้อมูลกับการควบคุมการไหลของข้อมูลเกิดความสับสนกันอยู่บ่อย ๆ ก็เป็นเพราะว่าการทำงานของอัลกอริทึมบางอย่างของวิธีการควบคุมความคับคั่งของข้อมูลนั้นมีการส่งข้อความไปบอกแหล่งส่งข้อมูลหลายแหล่งให้ส่งข้อมูลช้าลงในขณะที่ระบบเครือข่ายเริ่มเกิดปัญหาดังนั้นโฮสต์จึงได้รับข้อความให้ส่งข้อมูลช้าลงด้วยสาเหตุสองประการ คือผู้รับไม่สามารถจัดการข้อมูลในตอนนั้นได้ทัน และระบบเครือข่ายเริ่มมีปัญหาในการให้บริการ

2.5.1 แนวความคิดพื้นฐานในการควบคุมความคับคั่งของข้อมูล

ในระบบที่มีการทำงานแบบซับซ้อนเช่นระบบเครือข่ายคอมพิวเตอร์ จะเกิดปัญหาขึ้นมากมายเริ่มตั้งแต่การออกแบบไปจนถึงปัญหาในขณะปฏิบัติงาน จากทฤษฎีการควบคุมแบบต่าง ๆ ได้แบ่งแนวทางในการแก้ปัญหาออกเป็น 2 กลุ่ม คือ การแก้ปัญหาแบบวงจรเปิด (open loop) และ การแก้ปัญหาแบบวงจรปิด (close loop) หัวใจสำคัญของวิธีแก้ปัญหาแบบวงจรเปิดคือจะเริ่มต้นด้วยการพยายามออกแบบระบบให้ดีที่สุดเพื่อสร้างความมั่นใจว่าจะไม่เกิดปัญหาขึ้นเลย ดังนั้นเมื่อนำระบบไปใช้ทำงานจริงก็จะไม่มีการแก้ไขใด ๆ

กระบวนการแก้ปัญหาแบบวงจรเปิดประกอบด้วย (1) วิธีการตัดสินใจในการรับหรือไม่รับข้อมูลใหม่เข้ามาในระบบ, (2) วิธีการตัดสินใจเมื่อจะต้องละทิ้งแพ็กเก็ตรวมทั้งวิธีการเลือกแพ็กเก็ตที่จะทิ้งไปและ (3) สร้างตารางการตัดสินใจของการทำงานตามจุดที่ต่าง ๆ กันในระบบ ซึ่งวิธีการทั้งหมดที่กล่าวมานี้ไม่ได้นำสภาวะของระบบในขณะทำงานมาร่วมพิจารณาเลย

หัวใจหลักของการแก้ปัญหาแบบวงจรปิดจะตั้งอยู่บนพื้นฐานของการนำข้อมูลสภาวะของระบบในขณะนั้นเข้ามาเป็นข้อมูลสำคัญเพื่อใช้ในการแก้ปัญหา ซึ่งมีขั้นตอนการทำงาน 3 ขั้น เมื่อนำมาใช้แก้ไขความคับคั่งของข้อมูล

1. คอยจับตาดูระบบ เพื่อค้นหาส่วนที่เกิดปัญหาความคับคั่งของข้อมูล
2. ส่งข่าวสารนี้ไปบอกยังหน่วยที่มีความรับผิดชอบในการแก้ปัญหา
3. ปรับการทำงานระบบเพื่อแก้ไขปัญหาที่เกิดขึ้น

มาตรการหลาย ๆ แบบสามารถนำมาใช้ในการตรวจสอบและค้นหาจุดที่เกิดความคับคั่งของข้อมูลภายในเครือข่ายย่อยซึ่งเป็นกระบวนการทำงานในขั้นตอนแรก ที่นิยมนำมาใช้มากคือ (1) จำนวนเปอร์เซ็นต์ของแพ็กเก็ตที่ถูกทิ้งไปเมื่อหน่วยความจำในแถวคอยไม่เพียงพอ, (2) ความยาวเฉลี่ยของจำนวนแพ็กเก็ตในแถวคอย, (3) ปริมาณแพ็กเก็ตที่ถูกส่งใหม่เนื่องจากไม่ได้รับแพ็กเก็ตตอบรับภายในเวลาที่กำหนด, และ (5) ค่าเบี่ยงเบนมาตรฐานของการรอคอยของแต่ละแพ็กเก็ต ซึ่งตัวเลขที่กล่าวมา

ทั้งหมดนี้จะมีค่าเป็นปฏิภาคโดยตรงกับระดับความคับคั่งของข้อมูลในระบบ คือจะมีค่าเพิ่มขึ้นเมื่อระดับความคับคั่งสูงขึ้น

ขั้นตอนที่สอง คือการส่งข่าวสารเกี่ยวกับจุดที่เกิดความคับคั่งของข้อมูลขึ้นให้แก้โหนด (อาจเป็นเราเตอร์หรือโฮสต์) ที่สามารถแก้ไขปัญหานั้นได้ วิธีที่ง่ายและชัดเจนคือ ให้เราเตอร์ผู้ตรวจพบปัญหาทำการแจ้งข่าวนี้ไปยังโหนดที่มีส่วนเกี่ยวข้องกับปัญหาโดยตรง แน่นอนว่าข่าวสารที่ส่งไปนี้จะเป็นการเพิ่มปริมาณข้อมูลเข้าไปในส่วนที่กำลังเกิดปัญหาขึ้น ซึ่งเป็นส่งที่ควรหลีกเลี่ยงอย่างยิ่ง

อย่างไรก็ตาม หนทางหลีกเลี่ยงวิธีการแจ้งเตือนที่กล่าวถึงข้างต้นสามารถทำได้หลายวิธี เช่น การกำหนดให้ 1 บิตหรือหนึ่งเขตข้อมูลของทุก ๆ แพ็กเก็ตสงวนไว้เพื่อให้เราเตอร์ใส่สัญญาณแจ้งเตือนลงไปในบิตพิเศษ หรือใส่ข้อความแจ้งเตือนลงไปในเขตข้อมูลนี้เมื่อระดับความคับคั่งของข้อมูลสูงเกินว่าระดับปลอดภัยที่ได้กำหนดไว้ล่วงหน้า เราเตอร์ที่เกี่ยวข้องกับจุดที่เกิดปัญหาก็จะใส่ข่าวสารนี้ลงในทุก ๆ แพ็กเก็ตเพื่อเตือนเราเตอร์ข้างเคียง วิธีการนี้จึงสามารถแจ้งเตือนปัญหาที่เกิดขึ้นโดยไม่มีการเพิ่มปริมาณข้อมูลเข้าไปในระบบเลย

บางอัลกอริทึมได้กำหนดให้โฮสต์หรือเราเตอร์ส่งแพ็กเก็ตออกเป็นระยะ ๆ เพื่อถามหาข่าวสารเกี่ยวกับความคับคั่งของข้อมูลที่อาจเกิดขึ้นเมื่อข่าวสารนี้ย้อนกลับมาถึง ผู้ส่งจะได้ทราบถึงเส้นทางที่กำลังมีปัญหาและจะสามารถส่งข้อมูลโดยใช้เส้นทางอื่นเพื่อหลีกเลี่ยงพื้นที่นั้นได้ ตัวอย่างเปรียบเทียบที่เห็นได้ชัดเจนคือ การที่สถานีวิทยุบางแห่งได้ใช้เครื่องบินเฮลิคอปเตอร์ทำการบินวนรอบเมืองเพื่อรายงานสภาพการจราจรของถนนสายที่ติดขัดหรือจุดที่มีอุบัติเหตุเกิดขึ้น โดยมุ่งให้ข่าวสารที่การจายเสียงนั้นเป็นการเตือนให้บรรดารถยนต์ทั้งหลาย (ที่รับฟังการกระจายข่าว) หลีกเลี่ยงไปใช้เส้นทางอื่น

วิธีการแบบที่มีการป้อนข้อมูลย้อนกลับได้กล่าวถึงนั้น มีวัตถุประสงค์ที่จะแจ้งข่าวสารเพื่อให้โฮสต์ได้เตรียมหาเส้นทางหลีกเลี่ยงจุดที่เกิดปัญหา หรือจัดการอย่างเหมาะสมเพื่อช่วยลดความคับคั่งของข้อมูล การทำงานที่ถูกต้องนั้นจำเป็นจะต้องมีการจัดเวลาอย่างรอบคอบ เช่นในกรณีที่มีแพ็กเก็ตเข้ามา 2 แพ็กเก็ตติดต่อกัน เราเตอร์จะต้องส่งสัญญาณบอกให้ผู้ส่งหยุดส่งข้อมูลชั่วคราวและทุกครั้งที่เราเตอร์ว่างเป็นเวลา 20 ไมโครวินาทีก็จะส่งสัญญาณบอกให้เราเตอร์ข้างเคียงเริ่มส่งข้อมูลมาได้ ระบบที่ทำงานในลักษณะนี้จะเกิดการไหลเวียนของข้อมูลอย่างสม่ำเสมอและจะไม่เข้าสู่จุดอับ ในอีกกรณีหนึ่งเราเตอร์ได้กำหนดเวลารอคอยไว้ถึง 30 นาทีเพื่อให้เกิดความมั่นใจอย่างเต็มที่ก่อนที่จะส่งสัญญาณบอกให้เราเตอร์ข้างเคียงเริ่มส่งข้อมูลเข้ามาได้ กลไกในการคอบคุมอัลกอริทึมอย่างนี้ตอบสนองช้าเกินไปจนไม่สามารถนำไปใช้งานได้ เพื่อให้ระบบทำงานอย่างมีประสิทธิภาพเวลารอคอยจึงคำนวณมาจากการใช้ค่าเฉลี่ยมาตรฐานบางชนิด แต่การกำหนดค่าเวลาให้เหมาะสมที่สุดนั้นไม่ใช่เรื่องง่ายเลย

อัลกอริทึมสำหรับควบคุมความคับคั่งของข้อมูลมีใช้กันอย่างแพร่หลายมาก แยง (Yang) และรีดดี (Reddy) (1995) ได้พัฒนาวิธีการแยกประเภทสำหรับวิธีการบริหารความคับคั่งของข้อมูลให้เหมาะสมโดยการแบ่งอัลกอริทึมทั้งหมดออกเป็นสองประเภท คือ การแก้ปัญหาแบบวงจรปิด (open loop) และ การแก้ปัญหาแบบวงจรเปิดแบ่งออกเป็นสองพวกย่อย คือ พวกที่กระทำกับแหล่งกำเหนิด และ พวกที่พระทำกับแหล่งที่หมาย การแก้ปัญหาแบบวงจรปิดก็แบ่งออกเป็นสองพวกย่อย คือ การป้อนข้อมูลย้อนกลับด้วยการบังคับ (explicit feedback) และการป้อนข้อมูลย้อนกลับโดยอนุโลม (implicit feedback) อัลกอริทึมที่ใช้การป้อนข้อมูลย้อนกลับด้วยการบังคับ จะได้รับแพ็กเก็ตข้อมูลที่ถูกส่งกลับจากจุดที่เกิดความคับคั่งของข้อมูลโดยตรง ส่วนแบบที่สอง เราเตอร์จะต้องสรุปเองจากวิธีการสังเกตต่าง ๆ เช่น เวลาในการรอคอยแพ็กเก็ตตอบรับ เป็นต้น

การเกิดสภาวะความคับคั่งของข้อมูลนั้นหมายความถึงสภาวะของระบบที่มีปริมาณงานมากเกินกว่าที่ทรัพยากร (resource) ของระบบจัดการได้ดี วิธีการแก้ปัญหาแบบตรงไปตรงมา คือ การเพิ่มจำนวนทรัพยากรหรือการลดปริมาณของงานลง การเพิ่มจำนวนของทรัพยากรทำได้หลายวิธี เช่น ระบบเครือข่ายย่อยอาจเพิ่มเส้นทางสื่อสารเป็นการชั่วคราวด้วยการใช้สายโทรศัพท์ ซึ่งจะช่วยเพิ่มขีดความสามารถในการรับ – ส่งข้อมูลระหว่างโหนดสองโหนดได้ วิธีการนี้สามารถนำไปใช้ในระบบที่ทำงานคล้ายกับระบบ SMDS ที่ต้องการส่งข้อมูลจำนวนหนึ่งเป็นพิเศษซึ่งเป็นความต้องการชั่วคราวเท่านั้น ในระบบสื่อสารดาวเทียม การเพิ่มกำลังส่งก็จะเป็นการเพิ่มขีดความสามารถในการรับ – ส่งข้อมูลด้วย การกระจายข้อมูลเพื่อส่งผ่านเราเตอร์ในหลายเส้นทางแทนที่จะเลือกใช้เส่นทางที่ดีที่สุดเพียงเส้นทางเดียวก็เป็นการเพิ่มประสิทธิภาพของการส่งข้อมูลได้เหมือนกัน ท้ายที่สุดเราเตอร์สำรองที่เตรียมไว้ใช้ทดแทนเราเตอร์ที่กำลังใช้งานจริง สามารถนำมาใช้เพื่อเพิ่มขีดความสามารถในการรับ – ส่งข้อมูลได้เมื่อเกิดความคับคั่งของข้อมูลในขั้นวิกฤต

อย่างไรก็ตาม บางครั้งการเพิ่มขีดความสามารถในการรับส่งข้อมูลก็ไม่อาจทำได้ หรืออาจเป็นเพราะได้เพิ่มขีดความสามารถจนเต็มที่แล้ว หนทางที่เหลืออยู่คือ จะต้องลดปริมาณงานลงให้ได้ การลดปริมาณงานทำได้หลายวิธีซึ่งรวมถึงการงดให้บริการต่อผู้ใช้บางส่วน การลดระดับบริการของผู้ใช้บางคนหรือทุกคน และกำหนดให้ผู้ใช้ทุกคนจัดตารางความต้องการใช้งานเพื่อเป็นการทำนานปริมาณงานที่จะเกิดขึ้นได้ใกล้เคียงกับความเป็นจริง

บางอัลกอริทึมที่จะอธิบายต่อไปนั้นจะเกิดประโยชน์สูงสุดเมื่อนำไปใช้กับการเชื่อมต่อแบบวงจรเสมือน (virtual circuit) สำหรับเครือข่ายย่อยที่ใช้วงจรเสมือนเพื่อการสื่อสานภายในวิธีนี้จะนำไปใช้ในชั้นสื่อสารควบคุมเครือข่าย ส่วนเครือข่ายย่อยที่ใช้ดาต้าแกรมจะนำไปใช้ในชั้นสื่อสารนำส่งข้อมูลในบทนี้จะกล่าวถึงการจัดการความคับคั่งของข้อมูลในชั้นสื่อสารนำส่งข้อมูล

2.5.2 หลักนิยมในการหลีกเลี่ยงความคับคั่งของข้อมูล

การแก้ปัญหาความคับคั่งของข้อมูลแบบวงจรเปิด (open loop) พยายามที่จะออกแบบระบบให้หลีกเลี่ยงโอกาสที่จะทำให้เกิดความคับคั่งของข้อมูล มากกว่าที่จะหาทางแก้ไขเมื่อเกิดความคับคั่งของข้อมูลขึ้น แนวทางที่นำมาใช้คือ การกำหนดหลักเกณฑ์ในการปฏิบัติสำหรับโปรโตคอลในชั้นสื่อสารต่าง ๆ

ชั้นสื่อสาร	หลักนิยมที่ใช้
ชั้นนำส่งข้อมูล	การส่งแพ็กเก็ตซ้ำ การใช้คลังเก็บข้อมูลชั่วคราว (cache) การส่งแพ็กเก็ตตอบรับ การควบคุมการไหลของข้อมูล การกำหนดระยะเวลารอคอยแพ็กเก็ตตอบรับ
ชั้นควบคุมเครือข่าย	การใช้วงจรเสมือน หรือ ดาต้าแกรมของเครือข่ายย่อย การจัดแถวคอยแพ็กเก็ต และการให้บริการ การลบแพ็กเก็ตทิ้ง การเลือกทางเดินข้อมูล การบริหารอายุของแพ็กเก็ต
ชั้นเชื่อมต่อข้อมูล	การส่งแพ็กเก็ตซ้ำ การใช้คลังเก็บข้อมูลชั่วคราว การส่งแพ็กเก็ตตอบรับ การควบคุมการไหลของข้อมูล

ตารางที่ 2.5 แสดงชั้นสื่อสารกับการเชื่อต่อ

ในชั้นสื่อสารเชื่อมต่อข้อมูล การส่งแพ็กเก็ตซ้ำ (retransmission policy) เป็นวิธีการที่นำมาใช้ควบคู่กับการกำหนดระยะเวลารอคอยแพ็กเก็ตตอบรับของผู้ส่งข้อมูลการติดต่อกับผู้ส่งมีเวลารอคอยสั้นและใช้หลักการส่งแพ็กเก็ตทุกตัวใหม่ทุกครั้งจะเป็นการเพิ่มปริมาณงานเข้าสู่ระบบในอัตราที่สูงมากเมื่อเทียบกับผู้ส่งที่มีเวลารอคอยนานกว่าและเลือกการส่งแพ็กเก็ตซ้ำบางตัว (selective repeat) หลักการทำงานของการส่งแพ็กเก็ตซ้ำยังมีลักษณะคล้ายคลึงกับการทำงานแบบการใช้คลังเก็บข้อมูลชั่ว

คราว (caching policy) นั่นคือในกรณีที่ผู้รับข้อมูลใช้วิธีการลบแพ็กเก็ตที่ส่งมาแล้วเกิดการสลับตำแหน่งกันทั้งหมด แพ็กเก็ตเหล่านี้ก็จะถูกส่งซ้ำเข้ามาใหม่อีกอันจะเป็นการเพิ่มปริมาณงานให้กับระบบมากยิ่งขึ้น

การส่งแพ็กเก็ตตอบรับ (acknowledgement policy) มีผลต่อความคับคั่งของข้อมูลเช่นกันถ้าผู้รับส่งแพ็กเก็ตตอบรับสำหรับทุก ๆ แพ็กเก็ตที่ส่งมาถึงตนเอง จะทำให้แพ็กเก็ตตอบรับเหล่านี้กลายเป็นตัวเพิ่มความหนาแน่นของข้อมูลโดยตรง อย่างไรก็ตาม ถ้ามีการสะสมแพ็กเก็ตตอบรับไว้เป็นจำนวนมากแล้ว การกำหนดระยะเวลารอคอยฯ ของผู้ส่งก็จะต้องขยายให้นานออกไป มิฉะนั้นการส่งข้อมูลซ้ำก็จะเกิดขึ้นอย่างแน่นนอน การควบคุมการไหลของข้อมูลอย่างเคร่งครัด เช่น กำหนดให้มีการส่งข้อมูลเป็นระยะเวลาสั้น ๆ จะทำให้อัตราการส่งข้อมูลต่ำลงอันจะเป็นการช่วยลดปัญหาความคับคั่งของข้อมูลได้ด้วย

ในชั้นสื่อสารควบคุมเครือข่ายนั้นการเลือกวิธีการส่งข้อมูลแบบการใช้วงจรเสมือน (virtual circuit) กับดาต้าแกรม (datagram) มีผลต่อการเกิดความคับคั่งของข้อมูล เนื่องจากอัลกอริทึมสำหรับการแก้ไขปัญหาความคับคั่งของข้อมูลหลาย ๆ แบบจะทำงานบนเครือข่ายย่อยที่ใช้วงจรเสมือนเท่านั้น การจัดแถวคอยแพ็กเก็ต (packing queuing policy) และการให้บริการ (service policy) หมายถึงการที่เราเตอร์เลือกจัดแถวคอยประจำสายสื่อสารขาเข้า จัดแถวคอยประจำสายสื่อสารขาออก หรือจัดทั้งสองอย่าง ซึ่งจะมีความสัมพันธ์กับการจัดลำดับในการปฏิบัติงาน (เช่น การให้บริการแบบสลับหรือหมุนเวียนกันไป หรือแบบตามลำดับความสำคัญ) การลบแพ็กเก็ตทิ้ง (discard policy) เป็นกฎที่นำมาใช้ในการเลือกแพ็กเก็ตที่จะต้องลบทิ้งเนื่องจากเราเตอร์ไม่มีที่พอเพียงสำหรับแพ็กเก็ตทั้งหมดไว้ หลักนิยมที่เหมาะสมจะช่วยทำให้สถานการณ์บรรเทาลงได้ และในทางกลับกันหลักนิยมที่ไม่เหมาะสมจะทำให้สถานการณ์ลุกลามมากยิ่งขึ้น

การเลือกทางเดินข้อมูล สามารถช่วยหลีกเลี่ยงการเกิดความคับคั่งของข้อมูลได้โดยทำการแยกเส้นทาง ขนส่งข้อมูลออกไปในทุก ๆ เส้นทางที่เป็นไปได้ อัลกอริทึมที่ไม่เหมาะสมจะเลือกส่งข้อมูลทั้งหมดผ่านเส้นทางเดียวกัน ซึ่งถ้าเป็นเส้นทางที่กำลังเกิดสภาวะติดขัดอยู่แล้วก็จะยิ่งทำให้เส้นทางนั้นติดขัดหนักขึ้นไปอีก หลักการสุดท้ายสำหรับชั้นสื่อสารนี้คือการบริหารอายุของแพ็กเก็ต (lifetime management) การส่งแพ็กเก็ตเข้ามาในระบบจำเป็นต้องมีการกำหนดอายุเอาไว้เผื่อในกรณีที่แพ็กเก็ตหลงทางหรือถูกกักเก็บไว้ที่ใดที่หนึ่ง อายุจะเป็นตัวบอกว่าแพ็กเก็ตจะอยู่ในระบบได้นานเท่าใดก่อนที่มันจะถูกกำจัดออกไป ถ้าให้เวลาแพ็กเก็ตอยู่ได้นานเกินไป แพ็กเก็ตที่หลงทางอาจทำให้ความคับคั่งของข้อมูลคงอยู่นานมาก แต่ถ้ากำหนดเวลาน้อยเกินไป แพ็กเก็ตจะหมดอายุเสียก่อนที่จะเดินทางไปถึงจุดหมายก็จะทำให้ต้องส่งแพ็กเก็ตนั้นใหม่อีก

ปัญหาที่อาจเกิดขึ้นในชั้นสื่อสารเชื่อมต่อข้อมูลก็อาจเกิดขึ้นได้ในชั้นนำส่งข้อมูลเช่นกัน แต่การกำหนดระยะเวลาสำหรับการรอคอยแพ็กเก็ตตอบรับจากผู้รับข้อมูลนั้นมีความสลับซับซ้อนมากกว่ากันมากเนื่องจากการคำนวณระยะเวลารอคอยในการส่งข้อมูลข้ามระหว่างเราเตอร์สองตัวที่อยู่คนละปลายสายสื่อสารนั้นทำได้ง่ายมากและเชื่อถือได้แต่การคำนวณระยะเวลาที่ใช้ในการส่งข้อมูลผ่านเครือข่ายนั้นไม่สามารถทำนายให้เที่ยงตรงได้ ถ้าระยะเวลาการรอคอยฯสั้นเกินไปสำเนาของแพ็กเก็ตที่ถูกส่งไปแล้วก็จะถูกส่งออกไปอีกโดยไม่จำเป็น ถ้ากำหนดระยะเวลายาวเกินไป แม้ว่าจะไม่เพิ่มความคับคั่งของข้อมูลแต่อาจทำให้เวลาในการโต้ตอบระหว่างผู้รับและผู้ส่งยาวนานมากในกรณีที่แพ็กเก็ตเกิดการสูญหาย

2.5.3 การควบคุมรูปแบบการจราจร

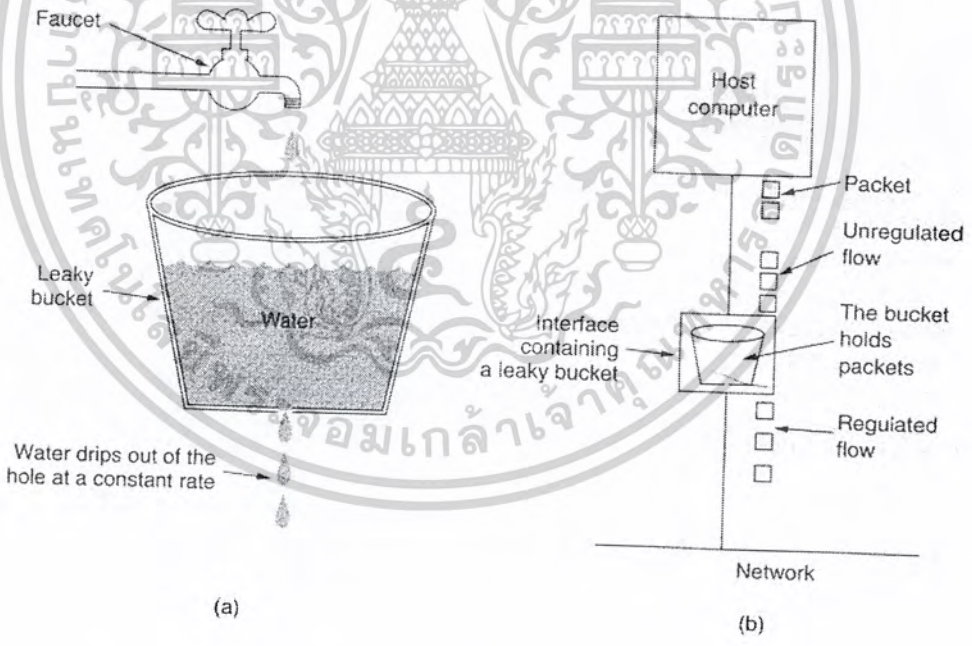
มูลเหตุหลักของการเกิดความคับคั่งของข้อมูลคือ เราเตอร์และโฮสต์ส่วนมากจะส่งข้อมูลเป็นช่วง ๆ ด้วยความเร็วสูงมากภายในระยะเวลาสั้น ๆ ที่เรียกว่า "เบริสท์ (brust)" สลับกับการหยุดนิ่ง ถ้าเราเตอร์และโฮสต์สามารถส่งข้อมูลในอัตราความเร็วสม่ำเสมอ โอกาสที่จะเกิดความคับคั่งของข้อมูลก็จะลดลง การบังคับให้แพ็กเก็ตถูกส่งออกมาในอัตราความเร็วที่สามารถคาดเดาหรือคำนวณได้เป็นอีกวิธีการหนึ่งของการแก้ปัญหาความคับคั่งของข้อมูลแบบวงจรเปิด วิธีการนี้ถูกนำไปใช้อย่างแพร่หลายในระบบเครือข่ายแบบ อะซิงโครนัสทรานส์เฟอร์โหมด (ATM) โดยเรียกว่า "การควบคุมรูปแบบการจราจร (traffic shaping)"

การควบคุมรูปแบบจราจรเกี่ยวกับการกำหนดอัตราเฉลี่ยของการส่งข้อมูลและการจัดการเบริสท์ ซึ่งไม่เหมือนกับโปรโตคอลหน้าต่างสื่อสารเลื่อนไหล (sliding windows) ซึ่งจะกำหนดขีดจำกัดปริมาณข้อมูลสำหรับการส่งในแต่ละครั้งไม่ได้กำหนดอัตราความเร็วในการส่งข้อมูล เมื่อวงจรเสมือนได้ถูกสร้างขึ้นมา ทั้งผู้ใช้ (โฮสต์) และเครือข่ายย่อย (เราเตอร์) ต้องทำความตกลงกันในเรื่องความถี่และอัตราความเร็วในการส่งข้อมูล รวมเรียกว่า "กติกาการส่งข้อมูล" ที่จะนำมาใช้ในวงจรนั้นตราบเท่าที่ผู้ใช้ดำเนินการส่งข้อมูลตามกติกาที่ได้ตกลงกันไว้ ผู้นำส่งข้อมูล (เราเตอร์) จะรับประกันที่จะนำข้อมูลไปส่งให้ตามช่วงเวลาที่เหมาะสม การควบคุมรูปแบบจราจรจะช่วยลดความคับคั่งของข้อมูล และช่วยให้ผู้นำส่งข้อมูลทั้งหลายสามารถทำตามข้อตกลงได้ กติกาที่ตกลงกันนั้นอาจจะไม่มีความสำคัญนักต่อการจัดการส่งแฟ้มข้อมูล แต่อาจจะมีความหมายอย่างยิ่งสำหรับระบบการนำส่งข้อมูลแบบเรียลไทม์ (real time) เช่นการส่งสัญญาณเสียงและสัญญาณภาพเคลื่อนไหว ซึ่งจะไม่สามารถใช้งานได้ในระบบที่มีความคับคั่งของข้อมูลในระดับสูง

การทำงานของการควบคุมรูปแบบจราจรนั้นเปรียบได้กับการที่ผู้ใช้จะบอกกับเราเตอร์ในทำนองที่ว่า "การส่งข้อมูลของฉันมีรูปแบบเป็นอย่างนี้นะ คุณจัดการได้ไหม ?" ถ้าเราเตอร์ตอบตกลงก็จะยึดถือเป็นกติกาที่ทั้งสองฝ่ายต้องปฏิบัติตามซึ่งจะมีระบบการตรวจสอบการไหลของข้อมูลให้เป็นไปตามกติกาเรียกว่า "traffic policing" การตกลงกันในเรื่องการควบคุมรูปแบบจราจรเกิดขึ้นในวงจรเสมือนมากกว่าดาต้าแกรม อย่างไรก็ตามในระบบเครือข่ายย่อยที่ใช้ดาต้าแกรมก็จะมีวิธีการในลักษณะคล้ายคลึงกันนำมาใช้ในชั้นสื่อสารนำส่งข้อมูลได้

2.5.3.1 อัลกอริทึมถังน้ำรั่ว (The Leaky Bucket Algorithm)

สมมติว่าให้ถังใส่น้ำใบหนึ่งมีรูรั่วเล็ก ๆ ที่ก้นถังไปรองน้ำที่กำลังไหลอยู่ ดังที่แสดงในรูป 2.14 ไม่ว่าอัตราการไหลเข้าของน้ำจะเป็นเท่าใดก็ตามอัตราการไหลออกของน้ำจะมีค่าคงที่อยู่สองค่าเสมอ คือ หนึ่งเป็นค่าตัวเลขเมื่อมีน้ำอยู่ในถัง และสองมีค่าเป็นศูนย์เมื่อไม่มีน้ำอยู่ในถังเลย แต่ถ้ามีน้ำเต็มถังและก็ยังคงใส่น้ำลงไปอีก น้ำก็จะไหลล้นออกทางด้านข้างซึ่งถือว่าเป็นส่วนที่สูญเสีย (ไม่นำมาพิจารณา)



รูปที่ 2.12 รูปแสดงอัลกอลิทึมถังน้ำรั่ว

การจัดส่งแพ็กเก็ตข้อมูลก็มีสภาพคล้ายกัน ดังแสดงในรูป 2.12 หลักการมีอยู่ว่าโฮสต์แต่ละตัวจะเชื่อมต่อกับระบบเครือข่ายผ่านตัวกลาง (เราเตอร์) ที่ทำงานแบบถังน้ำรั่ว ซึ่งก็คือการใช้แถวคอยที่มีขนาดจำกัด ถ้าแพ็กเก็ตมาถึงแถวคอยในขณะที่แถวคอยเต็มแพ็กเก็ตนั้นก็จะถูกลบทิ้งไปเหมือนกับน้ำที่ไหลล้นออกจากถังที่มีน้ำอยู่เต็ม หรือกล่าวอีกอย่างว่าถ้าโพรเซสบางตัวหรือหลาย ๆ ตัวของโฮสต์พยายามที่จะส่งแพ็กเก็ตใหม่เข้ามาในขณะที่แถวคอยเต็มแล้ว แพ็กเก็ตที่มาทีหลังก็จะถูกลบทิ้งไป กระบวนการทำงานนี้อาจจะกำหนดให้ตัวอุปกรณ์สื่อสาร (communication device) เป็นผู้จัดการหรือจะให้ระบบปฏิบัติการของโฮสต์ทำการจำลองการทำงานนี้ก็ได้ เทอร์เนอร์ (Turner) เป็นผู้นำเสนออัลกอริทึมนี้ในปี ค.ศ. 1986 โดยเรียกว่า "อัลกอริทึมถังน้ำรั่ว" ซึ่งในเนื้อแท้แล้วกระบวนการนี้ก็คือระบบแถวคอยของผู้ให้บริการเดี่ยวที่มีระยะเวลาการให้บริการคงที่ (single server queuing system with constant service time) นั่นเอง

โฮสต์แต่ละตัวได้รับอนุญาตให้ส่งแพ็กเก็ตเข้าไปในเครือข่ายได้ 1 ตัวต่อ 1 จังหวะสัญญาณนาฬิกา (clock tick) ของเครื่องคอมพิวเตอร์ ซึ่งจะถูกควบคุมโดยตัวอุปกรณ์สื่อสาร หรือระบบปฏิบัติการของโฮสต์เองก็ได้กลไกนี้จะทำการปรับปริมาณการไหลของแพ็กเก็ตข้อมูลของผู้ใช้งานทางฝั่งของโฮสต์ที่มีอัตราความเร็วไม่แน่นอนให้กลายเป็นการไหลของแพ็กเก็ตเข้าสู่ระบบเครือข่ายที่มีอัตราความเร็วสม่ำเสมอ จึงเป็นการลดโอกาสที่จะทำให้เกิดความคับคั่งของข้อมูลลงได้

ในกรณีที่แพ็กเก็ตทั้งหมดมีขนาดเท่า ๆ กัน อัลกอริทึมนี้จะถูกนำมาใช้ได้ตามขั้นตอนที่กล่าวข้างต้น อย่างไรก็ตาม ถ้าแพ็กเก็ตที่มีขนาดไม่เท่ากันถูกส่งเข้ามาในระบบ วิธีการนับจำนวนแพ็กเก็ตจะเปลี่ยนไปเป็นการนับปริมาณข้อมูลแทน เช่น กำหนดให้ส่งข้อมูล 1024 ไบต์ต่อหนึ่งสัญญาณนาฬิกาดังนั้นในหนึ่งสัญญาณนาฬิกาจะส่งแพ็กเก็ตขนาด 1024 ไบต์ได้ 1 ตัว หรือส่งแพ็กเก็ตขนาด 512 ไบต์ ได้ 2 ตัว หรือส่งแพ็กเก็ตขนาด 256 ไบต์ได้ 4 ตัวเป็นต้น แพ็กเก็ตที่มีขนาดใหญ่กว่าจำนวนไบต์ ที่เหลืออยู่จะถูกส่งในรอบสัญญาณนาฬิกาครั้งต่อไป

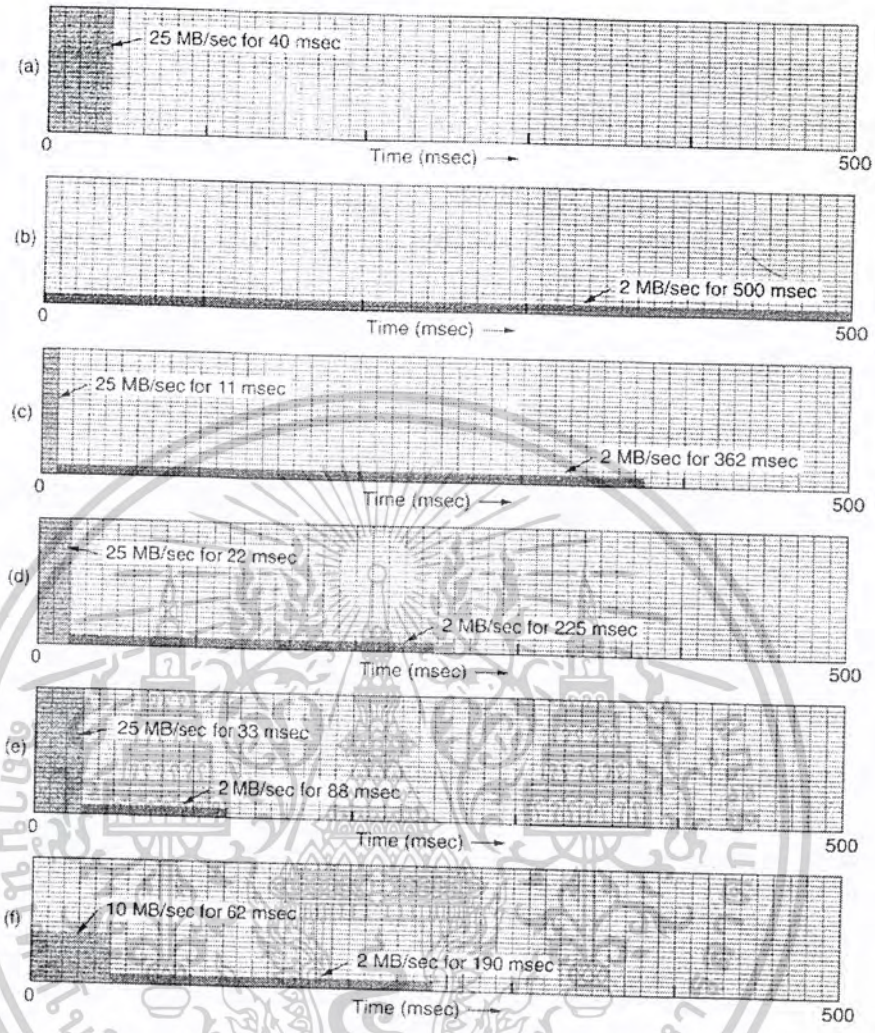
อัลกอริทึมถังน้ำรั่วประกอบด้วยแถวคอยขนาดจำกัดแถวหนึ่ง เมื่อแพ็กเก็ตเดินทางมาถึง ถ้ามีที่ว่างในแถวคอยแพ็กเก็ตก็จะถูกนำมาเก็บไว้ในที่ว่าง ถ้าไม่มีที่ว่างก็จะทิ้งแพ็กเก็ตนั้นไป ทุก ๆ จังหวะสัญญาณนาฬิกาจะมีแพ็กเก็ตถูกส่งเข้าไปในระบบหนึ่งตัว อัลกอริทึมที่ใช้การวัดปริมาณข้อมูลก็ใช้หลักการทำงานเดียวกัน ในทุก ๆ สัญญาณนาฬิกาตัวนับปริมาณข้อมูล (counter) จะเริ่มต้นกำหนดค่าไว้จำนวนหนึ่ง ถ้าแพ็กเก็ตแรกเข้ามาในแถวคอยมีปริมาณข้อมูลน้อยกว่าค่าที่ตั้งไว้แพ็กเก็ตนั้นจะถูกส่งออกแพ็กเก็ตตัวอื่นอาจถูไป และตัวนับปริมาณข้อมูลจะทำการลดค่าลงไปเท่ากับจำนวนไบต์ของแพ็กเก็ตที่เพิ่งส่งออกไป แพ็กเก็ตตัวอื่นอาจถูกส่งออกไปอีกก็ได้ตราบเท่าที่ขนาดของแพ็กเก็ตเล็กกว่าหรือเท่ากับค่าของตัวนับ ปริมาณข้อมูลที่เหลืออยู่มิฉะนั้นแล้วโฮสต์จะหยุดส่งข้อมูลเพื่อรอ

สัญญาณนาฬิกาครั้งต่อไป ซึ่งจะกำหนดค่าตัวนับปริมาณข้อมูลใหม่โดยไม่สนใจว่าค่าเดิมจะเป็นอะไรก็ตาม

สมมติให้คอมพิวเตอร์เครื่องหนึ่งสามารถส่งข้อมูลได้ด้วยความเร็ว 25 ล้านไบต์ต่อวินาที (เท่ากับ 200 ล้านบิตต่อวินาที) ซึ่งสายสื่อสารในระบบก็สามารถสนับสนุนการส่งข้อมูลที่ความเร็วนี้ได้อย่างไรก็ตาม เราเตอร์สามารถทำงานได้ที่ความเร็วขนาดนี้เพียงชั่วระยะเวลาสั้น ๆ ถ้าเป็นการส่งข้อมูลในช่วงระยะเวลายาวนานต่อเนื่องแล้ว เราเตอร์สามารถทำงานได้ที่ความเร็ว 2 ล้านไบต์ต่อวินาที สมมติว่าข้อมูลถูกส่งเข้ามา 1 ล้านไบต์ในเวลา 40 มิลลิวินาที (เทียบเท่ากับความเร็ว 25 ล้านไบต์ต่อวินาที) เราเตอร์สามารถลดอัตราความเร็วในการส่งข้อมูลให้ลงมาอยู่ที่ 2 ล้านไบต์ต่อวินาทีได้โดยการรับข้อมูลทั้งหมดไว้ในตัวเองพร้อม ๆ กับการปล่อยข้อมูลเข้าไปในระบบที่ความเร็ว 2 ล้านไบต์ต่อวินาทีเป็นเวลา 500 มิลลิวินาที ในรูป 2.13 (a) แสดงการไหลของข้อมูลมายังเราเตอร์ที่ทำงานด้วยความเร็ว 25 ล้านไบต์ต่อวินาทีเป็นเวลา 40 มิลลิวินาที รูป 5-25(b) เราเตอร์ส่งข้อมูลเข้าสู่ระบบที่ความเร็ว 2 ล้านไบต์ต่อวินาทีเป็นระยะเวลา 500 มิลลิวินาที

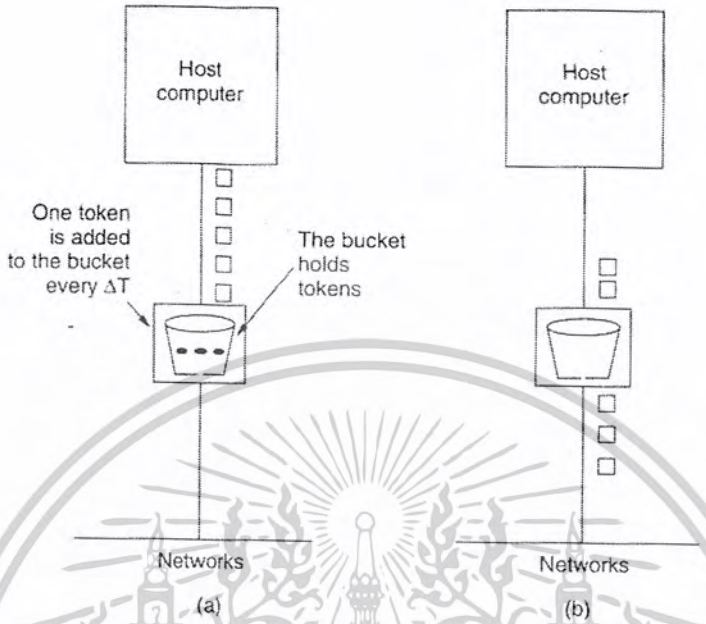
2.5.3.2 อัลกอริทึมโทเก้นบัคเก็ต

อัลกอริทึมถังน้ำรั่วจัดข้อมูลเข้าสู่ระบบด้วยอัตราความเร็วคงที่ไม่ว่าข้อมูลจะถูกส่งออกจากแหล่งกำเหนิดด้วยความเร็วเท่าใดก็ตาม สำหรับโปรแกรมประยุกต์บางประเภท การยอมให้ส่งข้อมูลที่ความเร็วสูงกว่าปกติในช่วงระยะเวลาสั้น ๆ เพื่อให้สอดคล้องกับสภาพที่ข้อมูลถูกส่งออกมาจากแหล่งกำเหนิดเป็นสิ่งที่ทำได้ อัลกอริทึมที่นำมาใช้จึงต้องปรับปรุงให้มีความอ่อนตัวมากขึ้นกลายเป็นวิธีที่เรียกว่า “อัลกอริทึมโทเก้นบัคเก็ต” ซึ่งได้เพิ่มข้อกำหนดให้ถังน้ำหรือบัคเก็ตต้องเก็บรักษาโทเก้น (ทำหน้าที่เหมือนใบกำกับสินค้า) ที่สร้างขึ้นตามจังหวะสัญญาณนาฬิกา เช่นเพิ่มโทเก้นหนึ่งตัวทุก ๆ 500 มิลลิวินาที ในรูป 2.13 ถังน้ำหรือบัคเก็ตมีโทเก้นอยู่ 3 ตัวโดยมีแพ็กเก็ตรอการนำส่งอยู่ 5 ตัว ทุกแพ็กเก็ตที่ถูกส่งออกไปจะต้องใช้โทเก้นหนึ่งตัว ดังนั้นจึงสามารถส่งแพ็กเก็ตได้ 3 ตัวในทันที ที่เหลืออีก 2 ตัวจะต้องรอจนกว่าโทเก้นตัวใหม่จะเกิดขึ้น ถ้าแพ็กเก็ตข้อมูลเป็นสินค้าการส่งสินค้าก็จะต้องมีใบกำกับสินค้าติดไปด้วยทุกครั้ง แต่โทเก้นในที่นี้ไม่ได้นำไปใช้ประโยชน์อย่างอื่นใดเลย การใช้โทเก้นในทางปฏิบัติจึงเป็นการลบโทเก้นหนึ่งตัวทิ้งต่อการส่งข้อมูลหนึ่งแพ็กเก็ตเท่านั้น



รูปที่ 2.13 แสดงอัลกอริทึมโทเก้นบัคเก็ต

อัลกอริทึมโทเก้นบัคเก็ตมีวิธีการนำส่งแพ็กเก็ตข้อมูลที่เข้ามาแบบ เบริสท์ ต่างไปจากวิธีเดิม อัลกอริทึมถังน้ำรั่วจะไม่ยอมให้โฮสต์มีการเก็บสะสมเวลาที่ไม่ได้ใช้งาน เพื่อใช้สำหรับการส่งแพ็กเก็ตจำนวนมากในภายหลัง แต่โทเก้นบัคเก็ตยอมให้มีการเก็บสะสมโทเก้นได้ถึงขีดจำกัดอันหนึ่งที่กำหนดไว้ล่วงหน้าหมายความว่า โทเก้นบัคเก็ตมีความสามารถที่จะส่งแพ็กเก็ตติดต่อกันในทันทีทันใดได้เท่ากับจำนวนสูงสุดของโทเก้นที่มีอยู่ในบัคเก็ต ทำให้รองรับการส่งข้อมูลที่มีความเร็วสูงมากได้เป็นบางครั้งซึ่งจะมีผลตอบสนองโดยตรงกับข้อมูลที่เข้ามาแบบ เบริสท์ ได้เป็นอย่างดี ข้อแตกต่างอีกประการหนึ่งคือโทเก้นบัคเก็ตไม่มีขีดจำกัดของแถวคอยเข้ามาเกี่ยวข้อง จึงไม่มีการลบแพ็กเก็ตข้อมูลทิ้ง



รูปที่ 2.14 อัลกอริทึมโทเก้นบัคเก็ต ก่อน-หลัง ส่งข้อมูล

ระบบเครือข่ายย่อยที่ใช้วิธีกำหนดปริมาณข้อมูลในการส่งแทนการนับจำนวนแพ็กเก็ต โทเก้นแต่ละตัวจะใช้แทนจำนวนไบต์ของข้อมูล แพ็กเก็ตจะถูกนำส่งได้ถ้าจำนวนไบต์ในโทเก้นที่มีอยู่มีขนาดไม่น้อยไปกว่าจำนวนไบต์ของแพ็กเก็ตที่ต้องการส่ง เศษที่เหลืออยู่ของโทเก้นสามารถนำไปรวมกบโทเก้นอื่นได้

ทั้งวิธีถังน้ำรั่วหรือวิธีโทเก้นบัคเก็ตสามารถนำมาใช้ในการช่วยการจราจรระหว่างเราเตอร์สองตัวให้ราบรื่นขึ้น และยังช่วยจัดการส่งข้อมูลของโฮสต์ได้ อย่างไรก็ตามโทเก้นบัคเก็ตสามารถบังคับให้โฮสต์หยุดส่งข้อมูลทั้งที่ยังมีข้อมูลที่ต้องส่งหรือบังคับให้เราเตอร์หยุดส่งข้อมูลในขณะที่ข้อมูลจากแหล่งกำเหนิดยังคงถูกส่งเข้ามาอย่างไม่ขาดสาย อันจะทำให้ข้อมูลสูญหายได้

สมมติว่าโทเก้นบัคเก็ตที่มีความจุของบัคเก็ต 250 กิโลไบต์ โทเก้นถูกสร้างขึ้นในอัตราที่ทำให้สามารถส่งข้อมูลได้ 2 ล้านไบต์ต่อวินาที ถ้าบัคเก็ตมีโทคเก้นสะสมอยู่เต็มในขณะที่ข้อมูลขนาด 1 ล้านไบต์ถูกส่งเข้ามาแบบเบิสท์ (burst-in) จากนั้น เมื่อใช้โทเก้นสะสมหมดแล้ว จึงส่งข้อมูลที่ระดับความเร็วปกติ

การคำนวณหาระยะเวลาเบิสท์ขาออก (burst-out) ที่เราเตอร์สามารถส่งข้อมูลด้วยอัตราความเร็วสูงสุดเป็นเรื่องซับซ้อนเล็กน้อย เนื่องจากในขณะที่ส่งข้อมูลอยู่นั้น โทเก้นใหม่ก้จะถูกสร้างขึ้นมาตลอดเวลา ถ้ากำหนดให้เวลาเบริสท์เป็น S วินาที, โทเก้นถูกนำมาใช้ในการส่งข้อมูลขนาด C

ไบต์, อัตราการสร้างโทเก้นซึ่งทำให้เกิดเป็นอัตราการส่งข้อมูลปกติ เป็น p ไบต์ต่อวินาที, และความเร็วสูงสุดในการส่งข้อมูลเป็น M ไบต์ต่อวินาทีแล้ว ปริมาณการส่งข้อมูลออกในช่วงเบริสท์จะเท่ากับ $C + pS$ ไบต์ซึ่งจะต้องเท่ากับปริมาณข้อมูลที่ถูกส่งด้วยความเร็วสูงสุดในระยะเวลาเดียวกันนั่นคือ MS ดังนั้นจะได้ความสัมพันธ์เป็น

$C + pS = MS$ นั่นคือ $S = C/(M - p)$

ปัญหาหลักของวิธีการโทเก้นบัคเก็ตก็คือการที่วิธีการนี้สนับสนุนการส่งข้อมูลแบบเบริสท์ แม้ว่าจะสามารถควบคุมปริมาณและความถี่ได้ด้วยการกำหนดค่าอัตราการสร้างโทเก้น และความเร็วสูงสุดในการส่งข้อมูลแล้วก็ตาม ระบบทั่วไปก็พยายามจะจำกัดจำนวนครั้งและความถี่ของการส่งข้อมูลที่ความเร็วสูงสุด ในขณะที่ไม่ต้องการส่งข้อมูลที่ความเร็วต่ำเสมอไปดังที่เกิดขึ้นในอัลกอริทึมถังน้ำรั่ว

หนทางหนึ่งในการแก้ปัญหาดังกล่าวคือการใช้วิธีถังน้ำรั่วเป็นตัวรับข้อมูลที่ส่งมาด้วยวิธีโทเก้นบัคเก็ต ความเร็วในการส่งข้อมูลของวิธีถังน้ำรั่วควรจะสูงกว่าอัตราการสร้างโทเก้น (p) แต่จะต้องต่ำกว่าความเร็วสูงสุดในการส่งข้อมูลของเครือข่าย

2.5.4 ข้อกำหนดการไหลของข้อมูล

การส่งผ่านข้อมูลจะมีประสิทธิภาพสูงสุดต่อเมื่อ ผู้ส่ง ผู้รับ และเครือข่ายย่อยสามารถทำความตกลงกันได้ เพื่อที่จะทำความตกลงกันอย่างรอบคอบในรายละเอียดของรูปแบบการจราจรของข้อมูลซึ่งเรียกว่า การจัดทำข้อกำหนดของการไหลของข้อมูล (flow specification) ประกอบด้วยโครงสร้างข้อมูลที่ใช้ในการส่งข้อมูลเข้าสู่ระบบ และคุณภาพของบริการที่โปรแกรมประยุกต์ต้องการ ข้อกำหนดการไหลของข้อมูลสามารถนำไปใช้ได้กับระบบเครือข่ายที่ส่งแพ็กเก็ตผ่านวงจรเสมือนหรือใช้ดาต้าแกรม หรือแม้กระทั่งการส่งแบบหลายเป้าหมายได้

คุณลักษณะของข้อมูล	บริการที่ต้องการให้รับประกัน
ขนาดแพ็กเก็ตสูงสุด (ไบต์)	อัตราการสูญหายของข้อมูล (ไบต์)
อัตราความเร็วของโทเก้นบัคเก็ต	ช่วงเวลาที่สูญเสียของข้อมูล (ไมโครวินาที)
ขนาดของโทเก้นบัคเก็ต (ไบต์)	อัตราการสูญหายของข้อมูลในช่วง burst (แพ็กเก็ต)
ความเร็วสูงสุดในการส่งข้อมูล (ไบต์/วินาที)	ช่วงการรอคอยที่ต่ำที่สุด (ไมโครวินาที) ความแปรปรวนของช่วงการรอคอยที่สูงที่สุด (ไมโครวินาที) ระดับการรับประกัน

ตารางที่ 2.6 แสดงข้อกำหนดการไหลของข้อมูล

ในหัวข้อนี้อธิบายถึงข้อกำหนดการไหลของข้อมูลที่ออกแบบโดย พาทริดจ์ (Partridge) (1992) แนวความคิดหลักคือ ก่อนที่จะมรการจักตั้งวงจรเสมือน หรือก่อนที่จะเริ่มส่งชุดดาต้าแกรม ผู้ส่งข้อมูลจะส่งข้อกำหนดการไหลของข้อมูลไปให้กับเครือข่ายย่อยเพื่อให้พิจารณา เครือข่ายย่อยอาจจะยอมรับหรือปฏิเสธก็ได้ หรืออาจจะแก้ไขเพียงบางส่วนแล้วส่งกลับมาให้ผู้ส่ง เมื่อทั้งสองฝ่ายตกลงกันได้แล้ว ผู้ส่งจึงจะเริ่มต่อรองกับผู้รับข้อมูล ในที่สุดทั้งสามฝ่ายก็จะได้ข้อกำหนดการไหลของข้อมูลที่ยอมรับได้ การสื่อสารจึงจะเริ่มต้นขึ้น

จากตารางข้อกำหนดการไหลของข้อมูลเริ่มจากคอลัมน์ทางด้านซ้ายโดย "ขนาดแพ็กเก็ตสูงสุด (Maximum packet size)" หมายถึงขนาดของแพ็กเก็ตที่ใหญ่ที่สุดที่สามารถนำมาใช้งานได้คุณลักษณะสองข้อถัดมาใช้สมมติฐานว่าเป็นการส่งข้อมูลโดยใช้อัลกอริทึมโทเก้นบัคเก็ตซึ่งบอกอัตราการเกิดของโทเก้นบัคเก็ต (token bucket rate) มีหน่วยเป็นไบต์ต่อวินาที ซึ่งหมายถึงอัตราการส่งข้อมูลเข้ามาในระบบ ตามด้วยขนาดของบัคเก็ต (token bucket size) ถ้าให้อัตราเป็น r ไบต์ต่อวินาทีและขนาดของบัคเก็ตเป็น b ไบต์แล้ว ในช่วงเวลา Δt ใด ๆ ปริมาณข้อมูลที่สามารถส่งได้สูงสุดเท่ากับ $b + r\Delta t$ ในที่นี้ b จะหมายถึงปริมาณข้อมูลที่บรรจุอยู่ในบัคเก็ตเมื่อเริ่มต้นการส่งข้อมูล และ $r\Delta t$ หมายถึงปริมาณข้อมูลที่เกิดการถ่ายทอดในช่วงเวลาต่อมาจนจบช่วงเวลานั้น ๆ "ความเร็วสูงสุดในการส่งข้อมูล (maximum transmission rate)" คือความเร็วในการส่งข้อมูลในระดับสูงสุดที่โฮสต์จะสามารถทำได้ภายใต้สภาวะทั่วไป ซึ่งสามารถนำไปคำนวณหาเวลาที่ต้องใช้น้อยที่สุดในการส่งข้อมูลทั้งหมดที่มีอยู่ในบัคเก็ตได้

ส่วนคอลัมน์ทางขวามือนั้น คือบริการที่โปรแกรมประยุกต์ต้องการรับประกันจากเครือข่ายย่อยสองบรรทัดแรก คือค่าตัวแปรที่ใช้คำนวณหาค่าการสูญเสียของแพ็กเก็ตที่สามารถยอมรับได้เช่น มีอัตราการสูญหายของข้อมูล (loss sensitivity) เป็น 1 ไบต์ต่อชั่วโมง หรือ ช่วงเวลาสูญเสียข้อมูล (loss interval) เป็น 10 ไมโครวินาที เป็นต้น อัตราการสูญหายของข้อมูลในช่วงเบิสท์ (burst loss sensitivity) บอกจำนวนแพ็กเก็ตที่สูญหายติดต่อกันในช่วงเบิสท์ที่บอมรับได้ ซึ่งจะต้องมีการแก้ไขและจะยังคงทำการส่งข้อมูลต่อไป

สองบรรทัดต่อมาอธิบายเกี่ยวกับเวลาในการรอคอย "ช่วงเวลารอคอยที่ต่ำที่สุด (minimum delay noticed)" หมายถึงช่วงเวลาการรอคอยของแพ็กเก็ตที่เกิดขึ้นระหว่าการเดินทางในเครือข่าย แต่โปรแกรมประยุกต์จะไม่ถือว่าช่วงเวลานี้เป็นช่วงเวลาที่เสียไปกับการรอคอย เช่น การคัดลอกสำเนาแฟ้มข้อมูลอาจกำหนดให้มีเวลารอคอยต่ำที่สุดเป็น 1 วินาที หากข้อมูลใดใช้เวลาในการเดินทางภายใน 1 วินาที ก็ยังถือเสมือนว่าไม่มีปัญหาใด ๆ เกิดขึ้น แต่โปรแกรมประยุกต์บางประเภท เช่น การส่งสัญญาณเสียงอาจกำหนดให้มีเวลารอคอยต่ำที่สุดเพียง 3 มิลลิวินาทีก็ได้ ดังนั้นผู้รับข้อมูลจะต้อง

เตรียมการแก้ไข ถ้าข้อมูลขาดช่วงไปนานเกินกว่า 3 มิลลิวินาที "ความแปรปรวนของช่วงเวลารอคอยที่สูงที่สุด (maximum delay เลิก variation)" หมายถึงช่วงเวลาที่แตกต่างกันของเวลารอคอย เช่น ถ้าให้เวลารอคอยแพ็กเก็ตต่ำสุดเป็น 5 วินาที และสูงสุดเป็น 8 วินาที ค่าของความแปรปรวนของช่วงการรอคอยสูงที่สุด คือ 3 วินาที

ส่วน "ระดับการรับประกัน (quality of guarantee)" บอกระดับความต้องการบริการต่าง ๆ ของโปรแกรมประยุกต์ ตัวอย่างเช่น อัตราการสูญหายของข้อมูล และช่วงการรอคอยที่ต่ำที่สุดของข้อมูลอาจเป็นวัตถุประสงค์ที่ต้องการ แต่โปรแกรมประยุกต์บางส่วนก็ไม่ได้ให้ความสนใจมากนักถ้าเรื่องดังกล่าวไม่เป็นไปตามที่ตกลงไว้ ในขณะที่โปรแกรมประยุกต์อีกบางส่วนอาจจะต้องการทำงานไปในทันทีที่ความต้องการดังกล่าวไม่ได้รับการสนองตอบ โปรแกรมอื่น ๆ ก็อาจมีพฤติกรรมที่แตกต่างกันออกไปได้

แม้ว่าข้อกำหนดการไหลของข้อมูลจะถูกมองไปในด้านความต้องการของโปรแกรมประยุกต์ที่มีต่อเครือข่ายย่อย สิ่งเหล่านี้อาจเป็นข้อมูลที่เครือข่ายย่อยเป็นผู้บอกให้ผู้ที่กำลังจะส่งข้อมูลได้ทราบก็ได้ซึ่งสามารถนำไปใช้ในการต่อรองในภายหลัง

ปัญหาที่เกิดขึ้นกับการกำหนดการไหลของข้อมูล คือการที่โปรแกรมประยุกต์เองอาจไม่ทราบความต้องการที่แท้จริงของตน เช่น ระยะเวลารอคอย 200 มิลลิวินาทีสำหรับการส่งข้อมูลระหว่างกรุงเทพฯกับเชียงใหม่อาจเป็นที่พอใจของผู้ใช้ แต่คงไม่มีผู้ใช้คนใดพอใจกับการรอคอยขนาดนี้เมื่อเป็นการติดต่อระหว่างคอมพิวเตอร์สองเครื่องที่ตั้งอยู่ในห้องติดกัน การให้บริการในระดับต่ำสุด (minimum service) จึงเป็นเรื่องของความเป็นไปได้หรือความน่าจะเป็นที่จะต้องพิจารณาให้เหมาะสมกับแต่ละสถานะการณ์

2.5.5 การควบคุมความคับคั่งของข้อมูลในเครือข่ายย่อยที่ใช้วงจรเสมือน

เทคนิคการควบคุมจำนวนผู้ใช้ (admission control) สามารถนำมาใช้ในเครือข่ายย่อยที่เกิดความคับคั่งขึ้นแล้ว โดยจะนำมาควบคุมไม่ให้ความคับคั่งเพิ่มขนาดหรือขยายตัวออกไปอีกมีหลักการทำงานคือ เมื่อเกิดความคับคั่งขึ้นแล้วจะไม่ยอมให้มีการจัดตั้งวงจรเสมือนขึ้นมาอีกจนกว่าความคับคั่งนั้นจะหายไป นั่นคือเครือข่ายจะขัดขวางการติดต่อในชั้นสื่อสารนำส่งข้อมูลซึ่งเป็นผู้จัดตั้งวงจรเสมือน แม้ว่าวิธีการนี้ค่อนข้างที่จะเด็ดขาดมากไป ทางชุมสายโทรศัพท์จะงดส่งสัญญาณ (tone signal) ไปให้แก่ผู้ใช้คนอื่น (ที่ยังไม่ได้ใช้โทรศัพท์ในขณะนั้น) เป็นการชั่วคราวจนกว่าจำนวนผู้ใช้จะลดลงสู่ระดับปกติ จะเห็นได้ว่าแม้วิธีการจะแตกต่างกันแต่ก็ใช้หลักการในทำนองเดียวกัน

วิธีประยุกต์ทางหนึ่งเพื่อลดความเข้มงวดลงบ้างคือการยอมให้ผู้ใช้อื่นสามารถจัดตั้งวงจรเสมือนขึ้นมาใช้งานได้แต่ก็จะจัดการให้วงจรที่สร้างใหม่นี้หลบไปใช้เส้นทางอื่นที่ไม่มีผลต่อความคับ

คั่งที่กำลังดำเนินอยู่ สมมติว่าโฮสต์ต้องการเชื่อมต่อกับเราเตอร์ A ต้องการสร้างวงจรเสมือนเพื่อติดต่อกับโฮสต์ที่เชื่อมต่อกับเราเตอร์ B ในขณะที่เราเตอร์ทั้งสองตัว A และ B กำลังมีปัญหาความคับคั่งของข้อมูลเกิดขึ้น โดยปกติวงจรเสมือนระหว่างโฮสต์ทั้งสองนี้จะใช้เป็นเส้นทางที่ผ่านเราเตอร์ที่กำลังมีปัญหา

อีกวิธีการหนึ่งที่นำมาใช้คือ การที่โฮสต์ทำการต่อรองกับเครือข่ายย่อยในเวลาขณะจัดตั้งวงจรเสมือนข้อตกลงส่วนใหญ่จะเกี่ยวข้องกับปริมาณและรูปแบบการส่งข้อมูล คุณภาพของการให้บริการและอื่น ๆ เหล่านี้ได้แก่ ตารางข้อมูล หน่วยความจำ และความกว้างของช่องสัญญาณสื่อสาร ด้วยวิธีการดังกล่าวโอกาสที่จะเกิดความคับคั่งของข้อมูลนั้นเกือบจะเป็นไปไม่ได้เลยเนื่องจากทรัพยากรที่จำเป็นต้องใช้ได้ถูกสงวนไว้ให้แล้ว

วิธีการ "จองทรัพยากร" นี้จะสามารถนำไปใช้เป็นขั้นตอนถาวรในกระบวนการสร้างวงจรเสมือนหรือนำไปใช้เฉพาะเมื่อเกิดความคับคั่งขึ้นในระบบก็ได้ ถ้าหากนำไปใช้เป็นการถาวรอาจจะทำให้การใช้งานทรัพยากรต่าง ๆ ในระบบเป็นไปอย่างไม่คุ้มค่า เช่น สมมติว่ามีการสร้างวงจรเสมือนขึ้น 6 วงจร มีปริมาณข้อมูลโดยรวม 6 ล้านไบต์ต่อวินาที ซึ่งเท่ากับขีดความสามารถในการนำส่งข้อมูลของสายสื่อสารพอดี ในการปฏิบัติงานจริงนั้นมีโอกาสน้อยมากที่วงจรเสมือนทั้ง 6 วงจรจะทำการส่งข้อมูลที่ความเร็วสูงสุดในเวลาเดียวกัน หมายความว่าเวลาส่วนใหญ่จึงสูญเปล่าไปโดยไม่ได้ใช้สายสื่อสารให้เกิดประโยชน์อย่างเต็มที่

2.5.6 โช้คแพ็กเก็ต

วิธีการโช้คแพ็กเก็ต ที่จะกล่าวถึงต่อไปนี้สามารถนำไปใช้ได้กับเครือข่ายย่อยที่ใช้การส่งข้อมูลทั้งแบบวงจรเสมือนและแบบบดาต้าแกรม เราเตอร์แต่ละตัวสามารถตรวจสอบประสิทธิภาพการทำงานของสายสื่อสาร (ขาเข้าและขาออก) ที่เชื่อมต่อกับตนเองและทรัพยากรอื่น ๆ ได้ เช่น เราเตอร์สามารถคำนวณค่าประสิทธิภาพการทำงาน (u) ของสายสื่อสารขาออกแต่ละเส้นซึ่งมีค่าอยู่ระหว่าง 0.0 ถึง 1.0 ได้ด้วยการสุ่มตัวอย่างค่าประสิทธิภาพการทำงานของสายสื่อสาร (f) ซึ่งสามารถทำได้เป็นประจำมาคำนวณหาค่า u ให้ใกล้เคียงกับความเป็นจริง ได้ดังนี้

$$u_{new} = au_{old} + (1-a)f$$

เมื่อ a เป็นค่าคงที่ค่าหนึ่งที่ใช้เป็นน้ำหนักบอกความสำคัญของข้อมูลเดิม (au_{old})

การประมาณค่าประสิทธิภาพการทำงานของสายสื่อสารขาออก (u) เป็นตัวบอกปริมาณข้อมูลที่ส่งผ่านสายสื่อสารเส้นหนึ่ง หรือโดยอุปนัย ค่า u ก็คือตัววัดระดับความคับคั่งของข้อมูลในสายสื่อสารนั่นเอง เมื่อใดก็ตามที่ u มีค่ามากกว่าค่าคงที่ที่กำหนดไว้ล่วงหน้าสายสื่อสารขาออกเส้นนั้นถือได้ว่าอยู่ในระดับ "เตือนภัย" เราเตอร์ทุกตัวจะต้องพิจารณาค่า u ก่อนทำการส่งข้อมูลเสมอ ถ้าค่า u อยู่

ในระดับ "เตือนภัย" เราเตอร์จะต้องส่งแพ็กเก็ตพิเศษ เรียกว่า "โช้คแพ็กเก็ต (choke packet)" ไปยังโฮสต์ที่เป็นผู้ส่งข้อมูลเพื่อบอกสถานะความคับคั่งที่อาจเกิดขึ้นให้ทราบ แพ็กเก็ตนั้นจะได้รับการใส่รหัส (togged) เพื่อที่จะได้ไม่เกิดการส่งโช้คแพ็กเก็ตซ้ำอีก ก่อนที่จะได้รับการนำส่งตามปกติ

โฮสต์ที่ได้รับโช้คแพ็กเก็ต จะต้องลดปริมาณหรือความถี่ในการส่งข้อมูลเข้าสู่ระบบลงส่วนหนึ่ง (กำหนดไว้ล่วงหน้า เช่น ลดลง 10 เปอร์เซ็นต์) แม้ว่าจะได้แจ้งปัญหาความคับคั่งไปแล้ว แต่แพ็กเก็ตอื่น ๆ ที่ส่งตามกันมาก็ยังอาจตกอยู่ในสภาวะเดียวกัน ดังนั้นจะมีโช้คแพ็กเก็ตจำนวนหนึ่งทยอยเข้ามาที่โฮสต์เพื่อบอกข่าวสายเดียวกัน โฮสต์จึงต้องเพิกเฉยโช้คแพ็กเก็ตที่เข้ามาติดต่อกันในช่วงระยะเวลาหนึ่งหลังจากนั้นจึงให้ความสนใจกับโช้คแพ็กเก็ตตัวใหม่ ถ้าข่าวสารที่ได้ทำให้ทราบว่าความคับคั่งของข้อมูลยังเป็นเช่นเดิม (หรืออาจแย่กว่าเดิม) โฮสต์จะต้องลดปริมาณหรือความถี่ในการส่งข้อมูลลงไปอีก แต่ถ้าไม่ได้รับข่าวสารจากโช้คแพ็กเก็ตตัวใหม่ แสดงว่าความคับคั่งของข้อมูลได้หายไปแล้ว โฮสต์ก็สามารถเพิ่มปริมาณหรือความถี่ในการส่งข้อมูลให้สูงขึ้นกว่าเดิมได้

โฮสต์สามารถรับอัตราการส่งข้อมูลได้โดยการปรับค่าตัวแปรต่าง ๆ เช่น ขนาดของหน้าต่างสื่อสาร หรือ อัตราการส่งข้อมูลในอัลกอริทึมถังน้ำรั่ว โดยทั่วไปโช้คแพ็กเก็ตตัวแรกที่มาถึงโฮสต์นั้นจะทำให้โฮสต์ลดอัตราการส่งข้อมูลลงไปครึ่งหนึ่งของอัตราปัจจุบัน ครั้งต่อไปก็จะลดลงเหลือครึ่งหนึ่งของอัตราฯในขณะนั้นหรือเท่ากับหนึ่งในสี่ของอัตราเริ่มต้น ครั้งต่อ ๆ ไปก็จะลดลงไปครึ่งหนึ่งเสมอ ส่วนการเพิ่มอัตราก็จะทำแบบค่อยเป็นค่อยไปเพื่อไม่ทำให้เกิดความคับคั่งของข้อมูล

การปรับปรุงในด้านอื่น ๆ สำหรับอัลกอริทึมนี้ เช่น ให้เราเตอร์เก็บค่าคงที่สำหรับการ "เตือนภัย" ได้หลายค่า เราเตอร์ก็มีโอกาสเลือกระดับการเตือนภัยให้เหมาะสม และโฮสต์ก็สามารถเลือกวิธีตอบสนองได้ดียิ่งขึ้น เช่น การเตือนภัยระดับต่ำ การเตือนภัยระดับกลาง และ การเตือนภัยระดับสูงสุด การปรับปรุงอีกวิธีหนึ่งคือการดูปริมาณงานที่ค้างอยู่ในแถวคอยสำหรับส่งแพ็กเก็ต หรือประสิทธิภาพการใช้งานหน่วยความจำ ก็สามารถนำมาใช้ในการตรวจสอบความคับคั่งของข้อมูลได้เช่นกัน

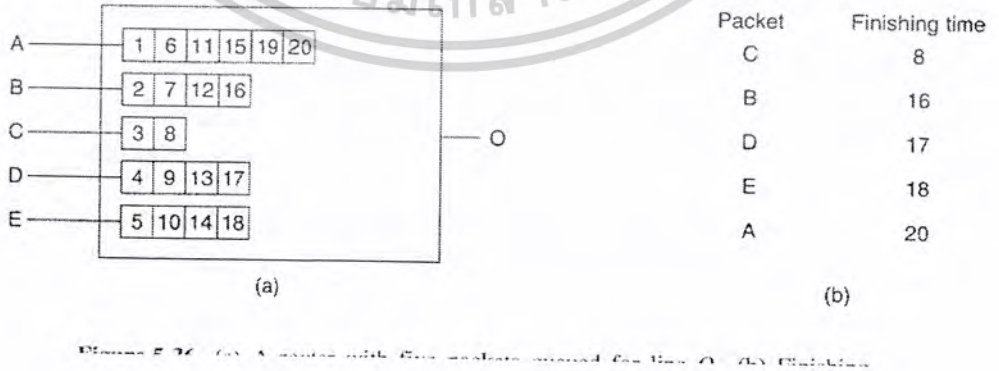
2.5.6.1 แถวคอยแบบการให้น้ำหนักอย่างเป็นธรรม

ปัญหาของการใช้โช้คแพ็กเก็ตคือโฮสต์แต่ละตัว (ซึ่งมีส่วนเกี่ยวข้องโดยตรงกับปัญหาที่เกิดขึ้น) มีอิสระอย่างเต็มที่ในการตอบสนองต่อคำเตือน สมมติว่าเราเตอร์ตัวหนึ่งกำลังเกิดปัญหาความคับคั่งเนื่องจากข้อมูลที่ส่งมาจากโฮสต์สี่ตัว จึงส่งโช้คแพ็กเก็ตไปยังโฮสต์ทั้งสี่นั้น โฮสต์ตัวหนึ่งตัดสินใจลดปริมาณการส่งข้อมูลลงครึ่งหนึ่งตามที่ควรจะทำ แต่โฮสต์อื่น ๆ กลับทำเฉย โฮสต์ที่ซื่อสัตย์ก็เลยส่งข้อมูลได้ช้าลงเพียงผู้เดียว ถ้าสถานะการณ์ยังไม่คลี่คลาย โฮสต์ผู้ซื่อสัตย์นี้ก็จะต้องลดความเร็วในการ

ส่งข้อมูลลงไปเรื่อย ๆ จนอาจต้องหยุดไปในที่สุด ในเวลาเดียวกันโฮสต์อีกสามตัวที่เหลือก็ยังคงส่งข้อมูลที่ความเร็วปกติต่อไป

เพื่อเป็นการแก้ปัญหาดังกล่าว แนเกิล (Nagle) (1987) ได้เสนอแนวทางการแก้ไขโดยใช้อัลกอริทึมการบริหารแพ็กเก็ตในแถวคอยอย่างเป็นธรรม (fair queueing algorithm) วิธีการนี้เราเตอร์จะต้องสร้างแถวคอยสำหรับแต่ละสายสื่อสารขาออก จำนวนแถวคอยในแต่ละสายจะเท่ากับจำนวนโฮสต์ที่ต้องการ ส่งข้อมูลออกทางสายเส้นนั้น เมื่อสายนั้นว่างลง เราเตอร์ก็จะเลือกแพ็กเก็ตขึ้นมาจากแถวคอยทั้งหมดตามลำดับการหมุนเวียนอย่างเป็นธรรม ด้วยวิธีการนี้แพ็กเก็ตที่มาจากโฮสต์ทุกตัวมีโอกาสที่จะได้รับการนำส่งเท่ากันหมดแม้ว่าจำนวนแพ็กเก็ตในแถวคอยจะไม่เท่ากันก็ตาม วิธีการนี้ได้นำไปใช้ในการส่งข้อมูลแบบ อะซิงโครนัสทรานส์เฟอร์โหมด ในหลายระบบ

ด้วยลักษณะการทำงานของอัลกอริทึมนี้ แม้ว่าจะให้โอกาส (จำนวนครั้งในการส่ง) แก่โฮสต์ต่าง ๆ เท่ากัน แต่แพ็กเก็ตที่มีขนาดใหญ่จะได้รับส่วนแบ่งเวลามากกว่าแพ็กเก็ตที่มีขนาดเล็กกว่าเสมอ ดีเมริสท์ (Demers et al.) (1990) ได้เสนอแนวทางแก้ไขโดยการปรับปรุงให้ใช้การวัดปริมาณข้อมูลแทนการนับจำนวนแพ็กเก็ต โดยให้ตรวจข้อมูลในทุกแถวคอยหมุนเวียนกันไปแบบเดิมแต่เป็นการตรวจขนาดข้อมูลไม่ใช่จำนวนแพ็กเก็ต หรืออีกนัยหนึ่งเป็นการสมมติว่าทุกแถวคอยใช้แพ็กเก็ตขนาดเดียวกันหมดในการตรวจนี้จะใส่หมายเลขกำกับสำหรับทุก ๆ แพ็กเก็ตสมมติ เรียงตามลำดับที่ควรจะนำส่ง เมื่อหมุนเวียนครบแล้วทุกแถวคอยจะได้หมายเลขสุดท้ายซึ่งเป็นหมายเลขสูงสุดเป็นของตนเอง จากนั้นจึงจัดเรียงแถวคอยทั้งหมดตามลำดับหมายเลขเหล่านี้จากน้อยไปหามาก เราเตอร์จะส่งข้อมูลแบบหมุนเวียนตามลำดับคือ จะนำแพ็กเก็ตที่มีแถวคอยหมายเลขต่ำสุดออกไปเป็นลำดับแรก ตามด้วยแพ็กเก็ตจากแถวคอยในลำดับที่สอง เวียนไปจนครบแล้วกลับไปที่แถวแรกอีก

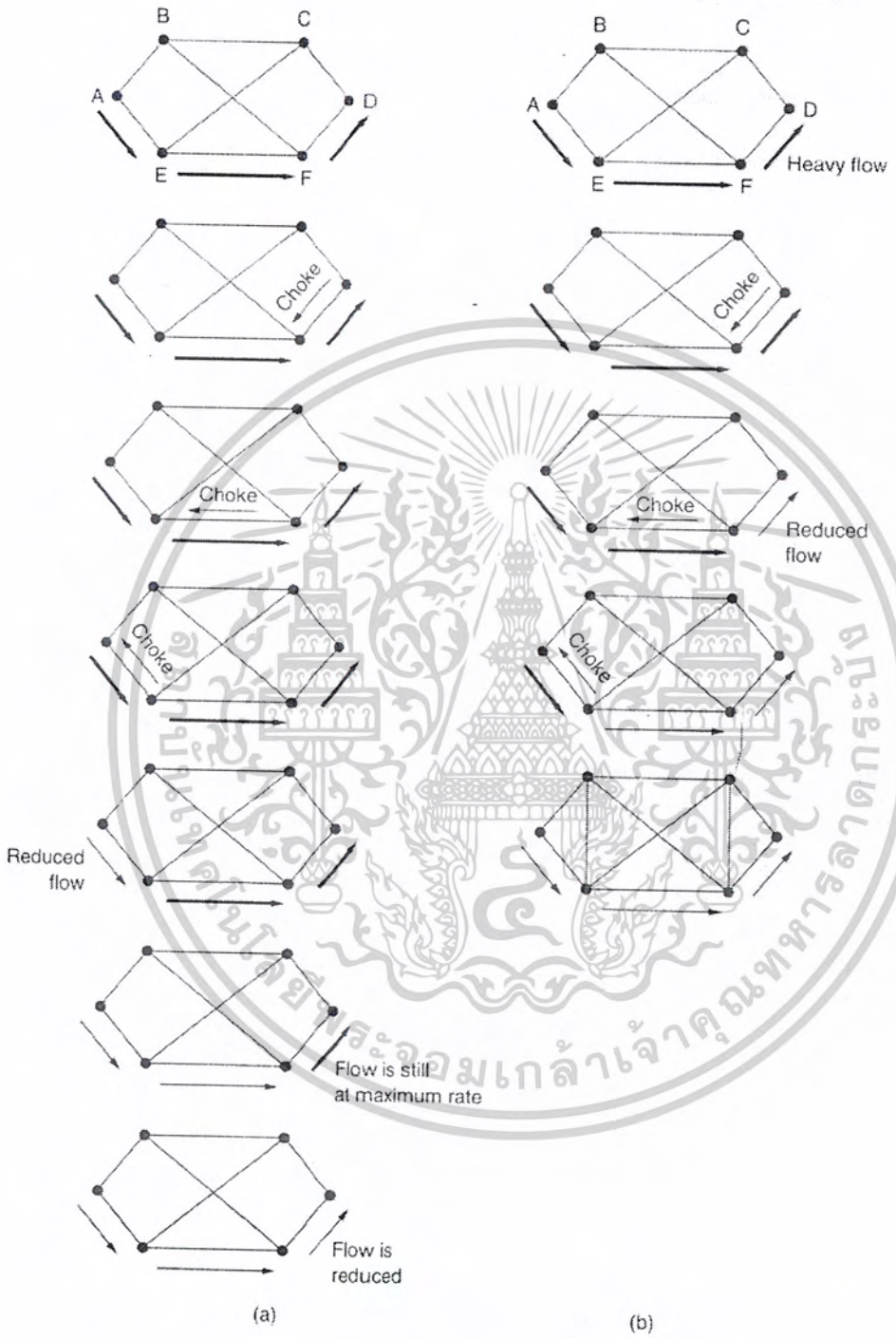


รูปที่ 2.15 แสดงแถวคอยการใช้น้ำหนักอย่างเป็นธรรม

การให้ความสำคัญของทุกแถวคอยเท่ากันหมดเป็นปัญหาสำหรับอัลกอริทึมนี้ ในสภาวะแวดล้อมหลาย ๆ แบบมีความนิยมที่จะให้ความสำคัญแก่แพ็กเก็ตที่ส่งจากผู้ให้บริการ (servers) มากกว่าแพ็กเก็ตที่ส่งมาจากผู้ใช้บริการ (clients) เช่น ส่งข้อมูลที่มาจากผู้ให้บริการเป็นสองเท่าของข้อมูลที่มาจากผู้ใช้บริการ การปรับปรุงอัลกอริทึมในลักษณะนี้เรียกว่า "การจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม (weighted fair queueing algorithm)" เป็นวิธีที่ได้รับความนิยมอย่างกว้างขวาง บางครั้งน้ำหนักที่ให้จะเท่ากับจำนวนวงจรเสมือนที่ส่งออกมาจากโฮสต์เพื่อให้โพรเซสมีโอกาสเท่า ๆ กัน

2.5.6.2 โช้คแพ็กเก็ตแบบช่วงต่อช่วง

การส่งโช้คแพ็กเก็ตกลับไปยังผู้ส่งข้อมูลผ่านเครือข่ายระยะไกล แม้ว่าจะใช้ความเร็วสูงก็ไม่สามารถทำงานได้ตามวัตถุประสงค์ที่ต้องการเนื่องจากรัยะเวลาการตอบรับนั้นนานเกินไป ถ้าเราเตอร์ A เป็นโหนดในกรุงเทพฯกำลังส่งข้อมูลไปยังเราเตอร์ D ที่อยู่ในจังหวัดเชียงใหม่ด้วยความเร็ว 155 ล้านบิตต่อวินาที หน่วยความจำที่ใช้ในการรับข้อมูลของเราเตอร์ D กำลังจะเต็ม โช้คแพ็กเก็ตจากเราเตอร์ D จึงถูกส่งกลับไปบอกเราเตอร์ A เพื่อให้ส่งข้อมูลช้าลง เส้นทางเดินของโช้คแพ็กเก็ตจะต้องใช้เวลาเดินทางนาน 30 มิลลิวินาที ในช่วงเวลาเดินทางนี้ ข้อมูลอีก 4.65 ล้านบิตจะถูกส่งมาที่เราเตอร์ D สมมติว่าโฮสต์ที่เราเตอร์ A จะหยุดส่งข้อมูลในทันทีที่ได้รับโช้คแพ็กเก็ต เราเตอร์ D ก็ยังคงจะต้องหาวิธีจัดการกับข้อมูลอีก 4.6 ล้านบิตที่ถูกส่งสวนทางมาแล้ว แต่จากนิยามที่อธิบายไว้ข้างต้นขั้นตอนสุดท้ายเท่านั้นที่เราเตอร์ D จึงจะได้รับข้อมูลในอัตราที่ช้าลง หนทางแก้ไขที่เป็นไปได้คือจะต้องให้ข่าวสารจากโช้คแพ็กเก็ตมีผลในทันทีกับเราเตอร์ในทุกช่วงของการส่งข้อมูล เราเตอร์ทุกตัวในเส้นทางที่โช้คแพ็กเก็ตเดินทางผ่าน จะต้องอ่านข่าวสารในโช้คแพ็กเก็ตและควบคุมความเร็วในการส่งข้อมูลไปยังเราเตอร์ D ให้ออกไปด้วยอัตราความเร็วที่ช้าลง เมื่อโช้คแพ็กเก็ตไปถึงเราเตอร์ E ข้อมูลที่จะส่งไปเราเตอร์ D ก็จะลดอัตราเร็วลงด้วยเช่นกัน ในที่สุดโช้คแพ็กเก็ตก็เดินทางไปถึงแหล่งผลิตข้อมูลคือเราเตอร์ A จากนี้ข้อมูลตลอดเส้นทางจึงจะถูกส่งด้วยความเร็วที่ช้าลง



รูปที่ 2.16 แสดงโชคแพ็กเก็ตแบบช่วงต่อช่วง

ผลประโยชน์ที่ได้รับจากการใช้โช้คแพ็กเก็ตแบบช่วง-ต่อ-ช่วง (hop-by-hop) นี้ จะทำให้ข่าวสายที่ส่งกลับไปยังผู้ส่งข้อมูลนั้นเกิดผลในทางปฏิบัติทันที ซึ่งจะช่วยแก้ปัญหาความคับคั่งของข้อมูลในจุดที่เกิดเหตุได้ อย่างไรก็ตามเราเตอร์ทุกตัวที่อยู่ในเส่นทางเดินของโช้คแพ็กเก็ตนั้นจะต้องใช้หน่วยความจำเพิ่มขึ้นอย่างมาก

2.5.7 การลบทิ้งข้อมูล

ถ้าวิธีที่กล่าวมาแล้วทั้งหมดไม่สามารถทำให้ความคับคั่งของข้อมูลหายไปได้ วิธีสุดท้ายที่ต้องทำก็คือการลบแพ็กเก็ตทิ้งโดยใช้อัลกอลิทึมที่เข้มงวดมากเรียกว่า "การลบทิ้งข้อมูล (load shedding)" มาใช้ วิธีการนี้ประยุกต์มาจากการแก้ไขปัญหากระแสไฟฟ้าที่ผลิตได้ไม่เพียงพอกับความต้องการของผู้ใช้ เช่น ในช่วงฤดูร้อนผู้คนจะใช้ไฟฟ้นอย่างมาก ถ้าหากว่าการไฟฟ้าไม่สามารถผลิตกระแสไฟฟ้าป้อนเข้าสู่ระบบได้เพียงพอและถ้าไม่มีการแก่ไขใด ๆ แล้ว เมืองทั้งเมืองก็จะไม่มีกระแสไฟฟ้าใช้เลย เพราะเครื่องผลิตกระแสไฟฟ้าจะทำงานเกินกำลังจนชำรุดในที่สุด วิธีแก้ไขก็คือ การไฟฟ้าจะตัดกระแสไฟฟ้าที่ส่งไปยังพื้นที่มีพื้นที่มีความสำคัญน้อยพื่อให้ปริมาณกระแสไฟฟ้าเพียงพอสำหรับป้อนให้กับพื้นที่ส่วนที่เหลือทั้งหมด

ในทำนองเดียวกัน เราเตอร์ที่มีแพ็กเตค้างอยู่ในแถวคอยจำนวนมากและมีแพ็กเก็ตทยอยส่งเข้ามาไม่ขาดสายก็อยู่ในสภาวะที่จะต้องทำอย่างใดอย่างหนึ่งก่อนที่เราเตอร์จะไม่สามารถควบคุมสถานการณ์ได้ โดยทั่วไปเราเตอร์จะเลือกวิธีลบแพ็กเก็ตทิ้งซึ่งอาจจะเป็นแพ็กเก็ตใด ๆ ก็ได้ แต่โดยปกติเราเตอร์จะเลือกลบแพ็กเก็ตทิ้งด้วยวิธีการต่าง ๆ ตามลักษณะของงาน เช่น การคัดลอกสำเนาแฟ้มข้อมูล แพ็กเก็ตที่ส่งมาก่อนจะมีความสำคัญมากกว่าแพ็กเก็ตที่ส่งเข้ามาทีหลังเนื่องจากการลบแพ็กเก็ตเก่าจะทำให้แพ็กเก็ตที่เหลืออยู่ในลำดับที่ไม่ถูกต้อง โปรโตคอลบางชนิดจะไม่ยอมรับแพ็กเก็ตที่เหลืออยู่เหล่านี้และจะบังคับให้ส่งแพ็กเก็ตชุดใหม่ทั้งชุดมาตามลำดับที่ถูกต้อง ส่วนโปรแกรมประยุกต์ประเภทมัลติมีเดียเลือกที่จะลบแพ็กเก็ตเก่าทิ้งมากกว่า

โปรแกรมประยุกต์อีกส่วนหนึ่งมีความซับซ้อนมากกว่าโปรแกรมทั่วไป ทำให้แพ็กเก็ตบางส่วนมีความสำคัญมากกว่าแพ็กเก็ตส่วนอื่น เช่น การส่งข้อมูลภาพเครื่อนไหวที่มีการบีบอัดข้อมูลเฟรม บางเฟรมจะถูกกำหนดให้เป็นเฟรมหลักซึ่งจะเก็บข้อมูลของภาพทั้งภาพ เฟรมอื่น ๆ เป็นเฟรมประกอบซึ่งจะเก็บข้อมูลเฉพาะบางส่วนของภาพที่มีการเปลี่ยนแปลงจากภาพในเฟรมหลัก ดังนั้นแพ็กเก็ตที่เก็บเฟรมหลักจะมีความสำคัญมากกว่าเฟรมที่เก็บเฟรมประกอบ เนื่องจากการสูญหายของเฟรมหลักจะทำ

ให้เฟรมประกบที่เหลืออยู่ไร้ความหมาย หรือการส่งข้อมูลที่เป็นภาพปนกับตัวหนังสือ การสูญเสียบางส่วนของภาพโดยปกติจะมีผลเสียนอยกว่าการสูญเสียข้อมูลตัวหนังสือบางบรรทัดเป็นต้น

วิธีการแบบซับซ้อนในการเลือกแพ็กเก็ตที่จะลบทิ้งนั้น โปรแกรมประยุกต์จะกำหนดระดับความสำคัญให้กับทุกแพ็กเก็ต เมื่อต้องการลบแพ็กเก็ตทิ้ง เราเตอร์จะเลือกลบแพ็กเก็ตที่มีระดับความสำคัญต่ำสุดก่อนตามด้วยแพ็กเก็ตในลับดับเพิ่มขึ้นในอีกครั้งหนึ่งระดับแพ็กเก็ตที่มีความสำคัญในระดับสูงสุดจะถูกลบในลำดับหลังสุด และอาจเป็นไปได้ว่าแพ็กเก็ตบางอย่างจะถูกจัดให้มีความสำคัญชนิด "ห้ามลบทิ้งโดยเด็ดขาด" ก็ได้

การกำหนดระดับความสำคัญสามารถทำได้มากมาย เช่น ในทางธุรกิจจะกำหนดให้แพ็กเก็ตของลูกค้าชั้นดีซึ่งจะต้องจ่ายค่าบริการแพงมาก มีลำดับความสำคัญสูง ในขณะที่ลูกค้าทั่วไปซึ่งจ่ายค่าบริการถูก หรือลูกค้าชั่วคราวเป็นกลุ่มที่ลำดับความสำคัญต่ำสุด ลำดับความสำคัญอาจผูกติดกับรูปแบบของการจราจร เช่น ในระบบโทเก้นบัคเก็ตที่เข้ามายังเราเอตร์ในขณะไม่มีโทเก้นเหลืออยู่อาจขอให้เราเตอร์ส่งระดับต่ำที่สุดซึ่งถ้าเกิดความคับคั่งขึ้นมาในระบบ แพ็กเก็ตประเภทนี้จะถูกลบทิ้งในลำดับแรก ในช่วงที่ระบบมีปริมาณงานต่ำ ผู้ใช้อาจจะนำวิธีการนี้ไปใช้ได้ เมื่อปริมาณงานเพิ่มมากขึ้นจนทำให้แพ็กเก็ตถูกลบทิ้ง ผู้ใช้ก็สามารถยกเลิกและหันกลับไปใช้วิธีการรอโทเก้นตามปกติก็ได้

อีกวิธีหนึ่งคือ การยอมให้โฮสต์ส่งแพ็กเก็ตออกมามากกว่ากติกาตกลงกันไว้ขณะจัดตั้งวงจรเสมือ เช่น โฮสต์อาจส่งข้อมูลด้วยความเร็วสูงกว่าที่กำหนดไว้ อย่างไรก็ตามแพ็กเก็ตส่วนเกินจะต้องถูกกำหนดระดับความสำคัญไว้ที่ระดับต่ำสุด แนวความคิดนี้จะช่วยให้การใช้งานสายสื่อสารมีประสิทธิภาพมากยิ่งขึ้น นั่นคือยอมให้โฮสต์ใช้งานสายสื่อสารได้อย่างเต็มที่ในขณะไม่มีผู้ใดต้องการใช้ แต่ก็ไม่ได้ใช้สิทธิ์อย่างถาวร

การนำระดับความสำคัญมาใช้ทำให้พิ่มข้อมูลนี้เข้าไปในส่วนหัวของแพ็กเก็ต โปรโตคอล อะซิงโครนัสทรานส์เฟอร์โหมด จัดเตรียมไว้ 1 บิตสำหรับบอกระดับความสำคัญสองระดับคือ ระดับความสูงและระดับความต่ำ ซึ่งนำไปใช้ประโยชน์ได้จริง

ในบางระบบเครือข่าย แพ็กเก็ตจะถูกจัดรวมเข้าระบบด้วยกันเพื่อเตรียมไว้ในกรณีที่ต้องการส่งข้อมูลนั้นซ้ำ เช่นโปรโตคอล อะซิงโครนัสทรานส์เฟอร์โหมด จะบอกข้อมูลออกเป็นส่วนเรียกว่าเซลล์ (cell) ซึ่งมีขนาดคงที่ โดยปกติเซลล์จะมีลักษณะเล็กกว่าขนาดข่าวสารที่ต้องการส่ง ดังนั้นแต่ละข่าวสารจะประกอบด้วยเซลล์ตั้งแต่สองเซลล์ขึ้นไป ในกรณีที่เซลล์สูญหายหรือถูกทำลายเพียงเซลล์เดียวจะทำให้ข้อมูลข่าวสารนั้นเสียหายทั้งหมด ซึ่งมีผลให้มีการส่งข้อมูลซ้ำ ในกรณีนี้เราเตอร์ ที่ลบทิ้งสามารถลบเซลล์อื่นในชุดเดียวกันได้ทั้งหมดเพราะเซลล์ที่เหลืออยู่แม้ว่าอาจจะเดินทางไปถึงผู้รับแต่ก็ไร้ค่าและจะต้องถูกส่งซ้ำอยู่ดี

ในการจำลองสภาพการทำงานระบบเครือข่ายย่อยแสดงให้เห็นว่าเมื่อเราเตอร์ตรวจพบปัญหาความคับคั่งในระบบ เราเตอร์ควรจะเริ่มลบแพ็กเก็ตทิ้งทันที ดีกว่าจะรอให้ปัญหาขยายตัวจนยากที่จะแก้ไข รายละเอียดอาจจะศึกษาได้ใน (Floyd and Jacobson,1993;Romanow and Floyd,1994) การทำงานตามข้อสรุปนี้จะช่วยป้องกันความคับคั่งไม่ให้ลุกลามต่อไปได้

2.5.8 การควบคุมความไม่ต่อเนื่องข้อมูล

โปรแกรมประยุกต์ประเภทการถ่ายทอดสัญญาณเสียงและภาพเคลื่อนไหวนั้น แม้ว่าจะมีระยะเวลาในการนำส่งข้อมูล 20 หรือ 30 มิลลิวินาทีต่อแพ็กเก็ตก็จะไม่ทำให้เกิดผลเสียหายตราบเท่าที่ระยะเวลานี้คงที่ สมมติว่าแพ็กเก็ตบางส่วนถูกส่งมาถึงผู้รับใน 20 มิลลิวินาที และแพ็กเก็ตบางส่วนมาถึงใน 30 มิลลิวินาที ผลที่ปรากฏคือ ผู้รับจะได้รับเสียงหรือภาพเคลื่อนไหวที่ไม่ต่อเนื่อง ดังนั้นทั้งโฮสต์และเครือข่ายย่อยจะต้องทำความตกลงในลักษณะการรับประกันระยะเวลาในการนำส่งข้อมูล เช่น จะส่งข้อมูลแต่ละแพ็กเก็ตจากแหล่งกำเหนิดไปถึงผู้รับให้ได้ภายในระยะเวลา 24 มิลลิวินาทีถึง 25 มิลลิวินาที กระบวนการนี้เรียกว่าเป็น "การควบคุมความไม่ต่อเนื่องของข้อมูล (jitter control)" ระยะเวลาที่กำหนดนี้จะต้องเป็นไปได้จริง และสามารถคำนวณได้จากปริมาณความคับคั่งของข้อมูลตลอดเส้นทางเดินที่ต้องการ

การควบคุมความไม่ต่อเนื่องสามารถคำนวณได้จากค่าคาดหวังในการส่งข้อมูลของแต่ละช่วงหรือการส่งข้อมูลจากเราเตอร์หนึ่งไปยังเราเตอร์ถัดไป แต่ละเราเตอร์สามารถตรวจสอบจำนวนแพ็กเก็ตที่อยู่ในแถวคอยว่าแพ็กเก็ตใดส่งไปเร็วหรือช้ากว่ากำหนด ข้อมูลนี้จะถูกฝากไว้ในแพ็กเก็ตและจะได้รับการปรับปรุงทุกครั้งที่เราเตอร์แต่ละตัว ถ้าแพ็กเก็ตได้รับการนำส่งเร็วกว่าที่กำหนดไว้ก็จะถูกบังคับให้รอจนกว่าจะถึงเวลาที่ควรนำส่งตามตาราง ถ้าหากช้ากว่ากำหนด เราเตอร์จะพยายามนำส่งแพ็กเก็ตนั้นโดยเร็วที่สุด อันที่จริงแล้วอัลกอริทึมที่ใช้ในการเลือกแพ็กเก็ตที่ควรนำส่งก่อนจะเลือกนำส่งแพ็กเก็ตที่อยู่ในแถวคอยที่ช้ากว่ากำหนดเป็นลำดับแรก ในเวลาเดียวกันแพ็กเก็ตที่เร็วกว่ากำหนดก็จะถูกบังคับให้รอคอยโดยอัตโนมัติ ทั้งหมดนี้จะมีผลในการลดความไม่ต่อเนื่องของข้อมูลลงได้

2.5.9 การควบคุมความคับคั่งของข้อมูลในระบบการส่งแบบกระจายหลายจุด

อัลกอริทึมสำหรับควบคุมความคับคั่งที่กล่าวถึงแล้วเป็นวิธีจัดการข้อมูลที่ถูกส่งออกจากแหล่งส่งข่าวสารแห่งหนึ่งไปยังผู้รับข่าวสารเพียงแห่งเดียว ในที่นี้จะกล่าวถึงวิธีจัดการข่าวสารที่ส่งจากผู้ผลิตข่าวสารหลายแห่งไปยังผู้รับที่กระจายอยู่หลายแห่งในระบบเครือข่าย ตัวอย่างเช่น สถานีโทรทัศน์วงจรปิดหลายแห่งแพร่กระจายข่าวสารที่เป็นทั้งสัญญาณเสียงและภาพเคลื่อนไหวไปยังสมาชิกกลุ่มหนึ่งซึ่งสมาชิกแต่ละคนสามารถเลือกที่จะรับสัญญาณจากสถานีใดก็ได้และสามารถเปลี่ยนไปรับข่าว

สารจากสถานีอื่นได้ในทุกโอกาสที่ต้องการ ตัวอย่างของโปรแกรมประยุกต์คือ การจัดประชุมผ่านระบบอิเล็กทรอนิกส์ (electronic conference) ซึ่งผู้ร่วมประชุมมีอิสระที่จะเลือกดูสมาชิกที่กำลังอภิปราย หรือดูปฏิกริยาของประธานขณะกำลังรับฟังก็ได้

โปรแกรมประยุกต์ในระบบการส่งกระจายหลายจุดอนุญาตให้สมาชิกของแต่ละกลุ่มสามารถย้ายกลุ่มได้ตลอดเวลา เช่น สมาชิกผู้หนึ่งเกิดเบื่อหน่ายการประชุมก็สามารถเปลี่ยนไปรับข่าวสารจากสถานีข่าวกีฬาได้ ภายใต้เงื่อนไขเหล่านี้ การที่จะให้ผู้ส่งข่าวสารสำรองช่วงสัญญาณสื่อสาร ไว้ล่วงหน้าจะไม่เกิดประสิทธิภาพเท่าที่ควร เพราะผู้ส่งจะคอยเฝ้าดูผู้รับที่มีอยู่ณะนั้นซึ่งสามารถเปลี่ยนไปใช้บริการข่าวสารจากกลุ่มอื่น หรือผู้รับจากกลุ่มอื่นเพิ่งจะเข้ามารับข่าวสารใหม่ที่เกิดขึ้นอยู่ตลอดเวลา ทำให้ผู้ส่งจะต้องสร้างสแปนนิ่งทรีใหม่ทุกครั้งที่เกิดการเปลี่ยนแปลงขึ้น สำหรับสถานีที่ส่งข่าวสารให้กับสมาชิกนับแสนนับล้านคน วิธีการนี้ย่อมไม่สามารถนำมาใช้ได้อย่างแน่นอน

2.5.9.1 โปรโตคอลสำรองทรัพยากร

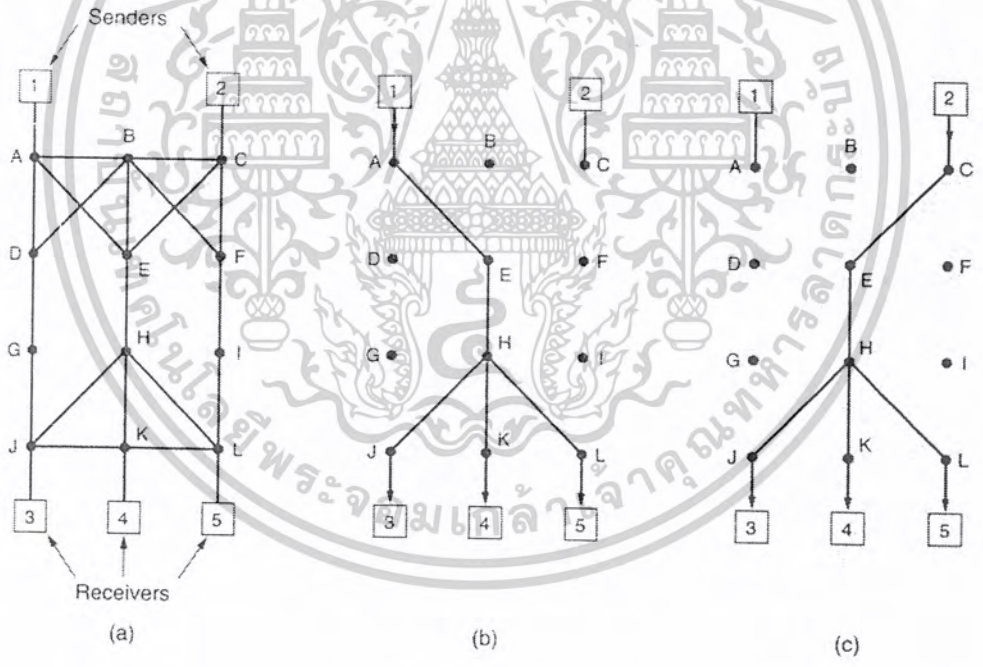
วิธีหนึ่งที่นำมาใช้บริหารระบบภายใต้เงื่อนไขข้างต้นเรียกว่า "โปรโตคอลสำรองทรัพยากร (RSVP: Resource eservation Protocol)" พัฒนาโดย แชง (Zhang et al., 1993) วิธีการนี้ยินยอมให้ผู้ส่งข่าวสารหลายคนสามารถกระจายข้อมูลไปยังผู้รับหลายกลุ่มได้อนุญาตให้ผู้รับแต่ละคนสามารถเปลี่ยนไปรับข่าวสารจากผู้ส่งคนใดก็ได้เมื่อต้องการ และบริหารการใช้ช่วงสัญญาณสื่อสารอย่างมีประสิทธิภาพในขณะที่พยายามกำจัดปัญหาความคับคั่งของข้อมูลด้วย

รูปแบบของโปรโตคอลใช้วิธีการเลือกเส้นทางเดินข้อมูลแบบกระจายหลายจุดโดยการสร้างสแปนนิ่งทรีดังที่ได้กล่าวไว้ข้างต้น แต่ละกลุ่มจะถูกกำหนดหมายเลขของกลุ่ม การส่งแพ็กเก็ตจะต้องระบุหมายเลขของกลุ่มเป็นที่อยู่ของผู้รับ การเลือกเส้นทางเดินข้อมูลแบบกระจายหลายจุดจะถูกนำมาใช้เพื่อสร้างสแปนนิ่งทรีสำหรับสมาชิกของกลุ่มทั้งหมด (วิธีการเลือกเส้นทางไม่ใช่ส่วนหนึ่งของอัลกอริทึมนี้) ขั้นตอนที่เพิ่มเติมขึ้นมาคือการส่งข้อมูลพิเศษตามระเวลาที่กำหนดไปยังเราเตอร์ต่าง ๆ ตลอดเส้นทางในสแปนนิ่งทรีที่สร้างขึ้น เพื่อบอกให้เราเตอร์เหล่านี้ทราบลักษณะโครงสร้างของกลุ่ม

ผู้รับแต่ละคนเมื่อต้องการเปลี่ยนการรับสัญญาณ ก็จะส่งข้อมูลกลับไปหาผู้ส่งตามเส้นทางที่ผู้ส่งใช้ส่งข่าวสารมาถึงตนเองโดยใช้วิธีการส่งข่าวสารย้อนกลับ (reverse path forwording) ตามที่ได้กล่าวมาข้างต้น แต่ละเราเตอร์ (ในเส้นทาง) จะบันทึกการเปลี่ยนแปลงและจัดการสำรองช่วงสัญญาณสื่อสารไว้ให้ซึ่งจะส่งข่าวบอกความล้มเหลวแก่ผู้รับในกรณีที่เราเตอร์ตัวนั้น ๆ ไม่สามารถสำรองทรัพยากรไว้ให้ได้ ในกรณีที่ประสบผลสำเร็จ เราเตอร์ทุกตัวตลอดเส้นทางนั้นจะสำรองทรัพยากรที่จำเป็นต้องใช้ไว้ให้เรียบร้อย

ในขณะที่ร้องขอการสำรองทรัพยากร ผู้รับสามารถที่จะเลือกแหล่งที่มาของข้อมูลได้มากกว่าหนึ่งแห่งในคราวเดียวกัน ผู้ใช้สามารถระบุได้ว่าทางเลือกที่ขอไปนี้เป็นการขอแบบตายตัวหรือแบบที่สามารถจะเปลี่ยนแปลงในภายหลังได้ เราเตอร์จะใช้ข้อมูลนี้ในการวางแผนการใช้ช่องสัญญาณสื่อสารให้มีประสิทธิภาพ ในความเป็นจริงแล้ว ผู้รับสองคน (หรือมากกว่านี้) จะใช้ช่องสัญญาณสื่อสารร่วมกันได้ก็ต่อเมื่อผู้ใช้ทั้งคู่ตกลงที่จะขอการสำรองทรัพยากรจากแหล่งข้อมูลแบบตายตัวเท่านั้น

การที่เครือข่ายย่อยยอมให้ผู้รับใช้ช่วงสัญญาณสื่อสารที่จองไว้มีความอ่อนตัวเป็นแบบ ไดนามิค นั้น มหายถึงการที่ผู้รับสามารถเปลี่ยนไปรับข่าวสารอื่น ๆ จากผู้ส่งได้โดยที่ไม่ต้องมีการขอสำรองทรัพยากรใหม่ เช่น โฮสต์ 2 อาจมีการแพร่ข่าวสารหลายแบบหรือภาพเคลื่อนไหวหลายช่อง เนื่องจากโฮสต์ 3 ได้ทำการขอสำรองทรัพยากรไว้แล้ว โฮสต์ 3 จึงสามารถเปลี่ยนไปรับข่าวสารแบบอื่นหรือภาพเคลื่อนไหวช่องอื่นของโฮสต์ 2 ได้ตลอดเวลา (โฮสต์ 3 ต้องระบุในขณะขอสำรองทรัพยากรว่าเป็นแบบเปลี่ยนแปลงได้) โดยใช้เส้นทางเดินข้อมูลเส้นเดิม



รูปที่ 2.17 โปรโตคอลสำรองทรัพยากร

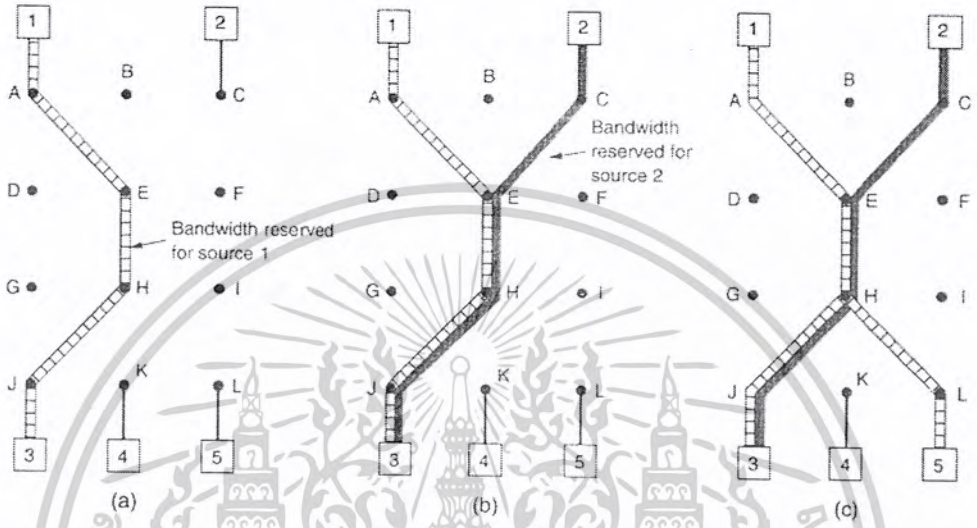


Figure 5-38. (a) Host 3 requests a channel to host 1. (b) Host 3 then requests a

รูปที่ 2.18 โปรโตคอลสำรองทรัพยากร

2.6 การจำลองเหตุการณ์

2.6.1 บทนำ

การจำลองระบบเน็ตเวิร์ค คือ การสมมติเหตุการณ์ที่เกิดขึ้นระหว่างส่งแพ็กเก็ต ออกไปในเครือข่าย เพื่อจำลองการทำงานของเครือข่ายแบบต่าง ๆ โดยในการที่จะจำลองการทำงานของเครือข่าย ผู้ใช้จะต้องกำหนดเหตุการณ์เริ่มต้น กำหนดโครงสร้างของเครือข่ายโดยใช้ออปเจ็คโหนดที่มีให้ จะทำการเริ่มส่งแพคเก็ตเมื่อใดและหยุดส่งเมื่อใด นอกจากออปเจ็คเน็ตเวอร์คแล้ว ยังมีส่วนประกอบหลักอื่น ๆ อีกเช่น ตัวจัดลำดับของเหตุการณ์ (event scheduler) โดยเหตุการณ์ก็คือ ตัวระบุของแต่และแพคเก็ต (packet ID) โดยตัวระบุนี้จะแตกต่างกันในแต่ละแพคเก็ต นอกจากนั้นยังมีเวลาที่ถูกจัดลำดับการทำงาน (scheduled time) ในการทำงาน ตัวจัดลำดับของเหตุการณ์จะคอยติดตามเวลาที่ใช้ในการจำลองการทำงาน และปล่อยให้เหตุการณ์ที่ถูกเก็บอยู่ในคิวของเหตุการณ์เกิดขึ้นเมื่อเวลาที่เหตุการณ์นั้นต้องเกิดดำเนินการมาถึง โดยทำการเรียกใช้เน็ตเวอร์คคอมโพเน้นต์ที่ทำการร้องขอให้เกิดเหตุการณ์นั้น ๆ และปล่อยให้เน็ตเวอร์คคอมโพเน้นต์ตัวนั้นทำงานกับแพคเก็ตต่อไป อย่างไรก็ตามในการติดต่อกันระหว่างเน็ตเวอร์คคอมโพเนนต์และเน็ตเวอร์คอมโพเนนต์อีกตัวที่ทำการส่งแพ็คเก็ตถึงกันไม่ได้กินเวลาจริงที่ทำการจำลองการทำงาน หากเน็ตเวอร์คคอมโพเน้นใดต้องการที่จะใช้เวลาที่ใช้ในการจำลองการทำงาน

เพื่อจัดการกับแพคเก็ต (เช่น ต้องการดีเลย์) ให้ทำการเรียกใช้ตัวจัดลำดับการทำงาน โดยการสร้างเหตุการณ์สำหรับแพคเก็ตนั้นให้เกิดการดีเลย์ขึ้น

2.6.2 แพ็คเก็ต

แพคเก็ต ประกอบด้วยสแตก (stack) ของเฮดเดอร์ (header) ในส่วนของข้อมูลนั้นอาจจะมีหรือไม่มีก็ได้ดังรูปข้างล่าง ดังที่ได้กล่าวไปแล้วในตัวอย่างสคริปต์ที่ใช้ในการจำลองการทำงานของเครือข่ายว่า เฮดเดอร์ของแพคเก็ตจะถูกกำหนดค่าเริ่มต้น เมื่อออปเจ็คที่ชื่อซิมูเลเตอร์ได้ถูกสร้างขึ้น ซึ่งสแตกของของเฮดเดอร์ที่ได้ทำการลงทะเบียน (register) ไว้เช่นเฮดเดอร์ทั่ว ๆ ไป ที่ถูกใช้โดยออปเจ็คต่าง ๆ เช่น เฮดเดอร์ IP ได้ถูกนิยามและออฟเซ็ต (offset) ของแต่ละเฮดเดอร์ในสแตกได้ถูกทำการบันทึก ซึ่งที่ได้กล่าวมานี้หมายความว่า ไม่ว่าแต่ละเฮดเดอร์จะถูกใช้หรือไม่ สแตกของเฮดเดอร์ที่ได้ทำการลงทะเบียนไว้จะต้องถูกสร้างขึ้น และเมื่อแพคเก็ตถูกจับจอง โดยเอเจ้นท์ออปเจ็คของเครือข่ายจะสามารถเข้าถึงเฮดเดอร์ใด ๆ ของแพคเก็ต โดยผ่านทางค่าออฟเซ็ต

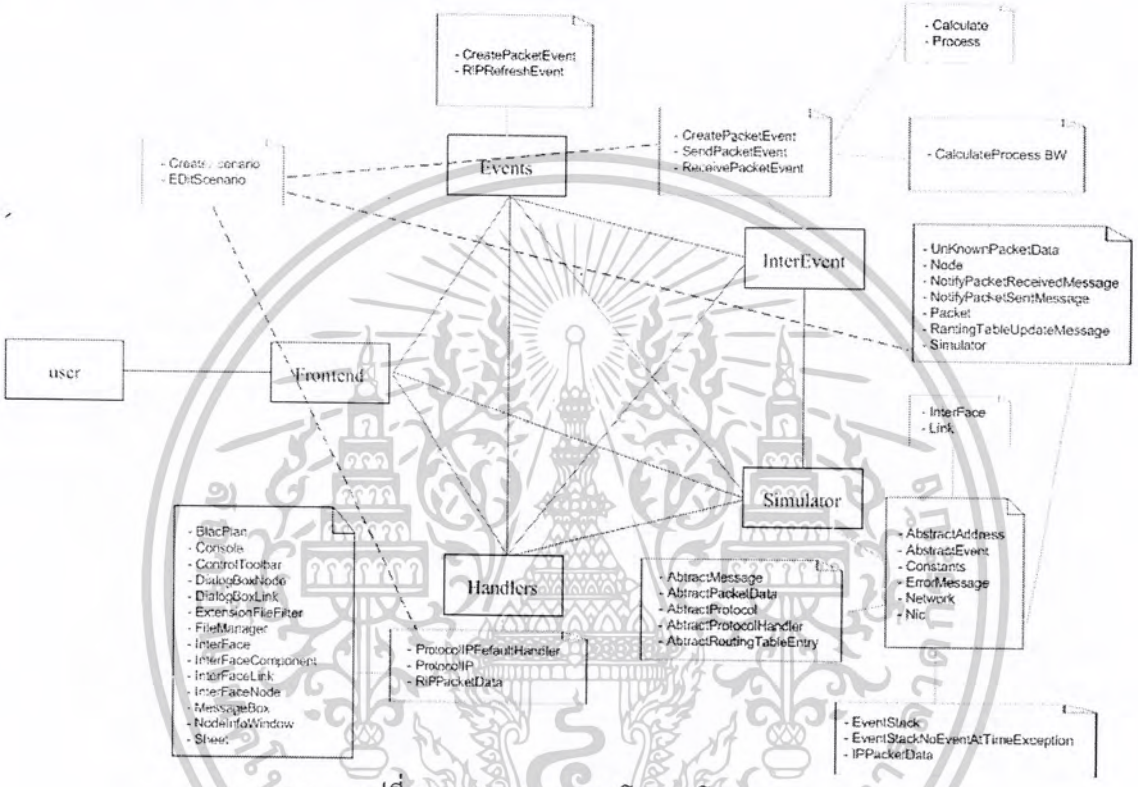
โดยปกติแล้วแพคเก็ตจะมีเพียงสแตกของเฮดเดอร์ (ในส่วนของข้อมูลจะเก็บค่าพ้อยเตอร์ที่เป็นนัลล์) ถึงแม้ว่าแพคเก็ตจะสามารถบรรจุข้อมูลจริง ๆ เข้าไปได้ (จากแอพพลิเคชันใด ๆ) โดยการจับจองเนื้อที่ในส่วนของข้อมูล แต่ในการอิมพลีเมนต์แอพพลิเคชันและเอเจ้นท์แทบจะไม่มีการรองรับคุณสมบัตินี้ เพราะมันไม่มีความหมายที่จะบรรจุข้อมูลในการจำลองการทำงาน ที่เป็นแบบนอนเรียล ไทม์ (non-real-time simulation) แต่ถึงอย่างไรก็ตาม ถ้าหากเราต้องการที่จะพัฒนาแอพพลิเคชันที่สามารถติดต่อกับอีกแอพพลิเคชัน โดยผ่านทางเครือข่าย เราจำเป็นต้องใช้คุณสมบัตินี้ โดยการแก้ไขโค้ดของเอเจ้นท์ หรืออาจจะใช้อีกวิธีหนึ่งคือการสร้างเฮดเดอร์ใหม่สำหรับแอพพลิเคชันนั้น แล้วทำการแก้ไขเอเจ้นท์ให้ทำการเขียนข้อมูลที่ได้รับจากแอพพลิเคชัน แล้วจากนั้นแอพลิเคชันจะนำไปใส่ในเฮดเดอร์ที่สร้างใหม่

2.6.3 หลักการจำลองเหตุการณ์

เมื่อผู้ใช้ทำการกำหนดค่าให้กับเราเตอร์ที่สร้างขึ้น เช่น ชื่อ อินเตอร์เฟส แบนด์วิธ ดีเลย์ และเมื่อทำการกำหนดเหตุการณ์ที่เกิดขึ้น โปรแกรมจะทำการประมวลผล ทีละเราเตอร์เพื่อหาค่าโปรโตคอลการหาเส้นทาง จากนั้นจึงพิจารณาจำนวนแพ็กเก็ตที่เข้ามาในแต่ละเราเตอร์ โดยผ่านเหตุการณ์ต่าง ๆ ที่กำหนดขึ้น เมื่อได้ค่าแพ็กเก็ตที่ผ่านเราเตอร์แล้ว นำมาแสดงผลในแต่ละช่วงเวลาที่กำหนดขึ้น

บทที่ 3
การออกแบบโครงงาน

3.1 โดเมนโมเดล (Domain Model)



รูปที่ 3.1 แสดงแผนภาพโดเมนโมเดล

ในส่วนของการออกแบบของโปรเจ็คได้นำซอฟแวร์ที่มีอยู่แล้วซึ่งได้นำซอฟแวร์ของ Flan Network Simulator มาทำการเพิ่มความสามารถและแก้ไขปรับปรุง ความสามารถเดิมของโปรแกรมสามารถที่จะทำการจำลองการส่งข้อมูลโดยใช้ไอพีโปรโตคอลในการส่งข้อมูล และใช้อาร์ไอพีเป็นโปรโตคอลในการอัพเดทตารางการหาเส้นทาง ซึ่งโปรแกรมนี้จะมีการสร้างเหตุการณ์จำลองได้เพียงครั้งละ 1 เหตุการณ์ ส่วนที่เราได้ทำการออกแบบและแก้ไขเพิ่มเติมมีดังต่อไปนี้

- ส่วนที่เพิ่มความสารถในการแสดงค่าปริมาณแบนวิธที่เราเตอร์ถูกใช้งาน ณ เวลาต่างๆ
- การคำนวณหาค่าปริมาณที่เราเตอร์ถูกใช้งานของแต่ละเราเตอร์ ขณะทำการจำลอง ณ เวลาต่างๆ

ในส่วนนี้จะมีการออกแบบวิธีการคิดคำนวณหาค่าปริมาณแบนวิธที่เราเตอร์ถูกใช้งาน เพื่อจะดูการถูกใช้งานของเราเตอร์แต่ละตัว ซึ่งในหลักความเป็นจริงปัจจัยที่ส่งผลถึงปริมาณแบนวิธที่เราเตอร์

ถูกใช้งาน ในเราเตอร์อาจจะมีค่ามากกว่าที่เรานำมาคิด แต่ปัจจัยที่เราได้นำมาคิดนี้ก็ถือว่าเป็นส่วนหนึ่งที่มีผลกับระบบ หรือตัวเราเตอร์นั่นเอง วิธีการคำนวนสามารถคำนวณได้จากสมการดังต่อไปนี้

$$\text{Process BW} = \frac{(\text{CreatePacketEvent} + \text{ReceivePacketEvent}) \times 8) \times 100\%}{\text{Max process} \times 10^9} \text{ (bit/sec)}$$

Process BW คือ ปริมาณแบนวิธที่เราเตอร์ถูกใช้งาน ณ เวลาใด ๆ ของเราเตอร์

CreatePacketEvent คือ ปริมาณเหตุการณ์ที่ถูกสร้างขึ้นในเราเตอร์ ณ เวลาใด ๆ

ReceivePacketEvent คือ ปริมาณเหตุการณ์ที่เราเตอร์ถูกรับเข้ามา ณ เวลาใด ๆ

Max process คือ ค่าปริมาณแบนวิธที่เราเตอร์สามารถใช้งานได้มากที่สุด

จะเห็นได้ว่าหลังจากการนำค่า (CreatePacketEvent + ReceivePacketEvent) แล้วคูณด้วย 8 นั้นเนื่องจากค่าที่ได้จากการ CreatePacketEvent และ ReceivePacketEvent นั้นจะมีหน่วยเป็นไบต์ แต่เราต้องการคิดในหน่วยของบิตต่อวินาที จึงต้องแลงค่าจากไบต์มาเป็นบิตด้วยการคูณด้วย 8 เนื่องจาก 1 ไบต์มีค่าเท่ากับ 8 บิต แล้วหารด้วยค่า Max process ที่ต้องคูณด้วย 10^9 เนื่องจากว่า ค่าที่เรารับมาจากส่วนอินพุทมีหน่วยเป็น G/Sec เราจึงต้องคูณด้วย 10^9 เพื่อแปลงให้มีค่าเป็นบิต หลังจากนั้นนำค่าที่ได้ทั้งหมดมาคูณด้วย 100% เพื่อคิดค่าเป็นร้อยละ ดังนั้นเราจึงมีการออกแบบในส่วนของยูสเซอร์อินเตอร์เฟสเพิ่มเติมในส่วนที่รับค่า Max process ทางหน้าจอ

3.2 ส่วนแสดงผลปริมาณที่เราเตอร์ถูกใช้งานของแต่ละเราเตอร์ ขณะทำการจำลอง ณ เวลาต่างๆ

ส่วนนี้จะมีการออกแบบในการแสดงผลปริมาณที่เราเตอร์ถูกใช้งานหลังจากการคิดเป็นร้อยละแล้วโดยแบ่งส่วนแสดงผลออกเป็น 2 ส่วนคือ

ส่วนที่ 1 แสดงผลปริมาณที่เราเตอร์ถูกใช้งานของแต่ละตัว ขณะทำการจำลอง ณ เวลาต่างๆ แสดงเป็นกราฟิก เป็นโพรเกรสบาร์ แสดงร้อยละที่ถูกใช้งานไป

ส่วนที่ 2 แสดงผลปริมาณที่เราเตอร์ถูกใช้งานของแต่ละตัว ในรูปแบบร้อยละเป็นเท็กซ์ (Text) นอกจากนี้จะมีการแสดง ชื่อของเราเตอร์ ปริมาณข้อมูลที่ถูกสร้าง เวลาที่ถูกสร้าง หมายเลขไอพี ของแต่ละเราเตอร์ในช่วงเวลาต่าง ๆ

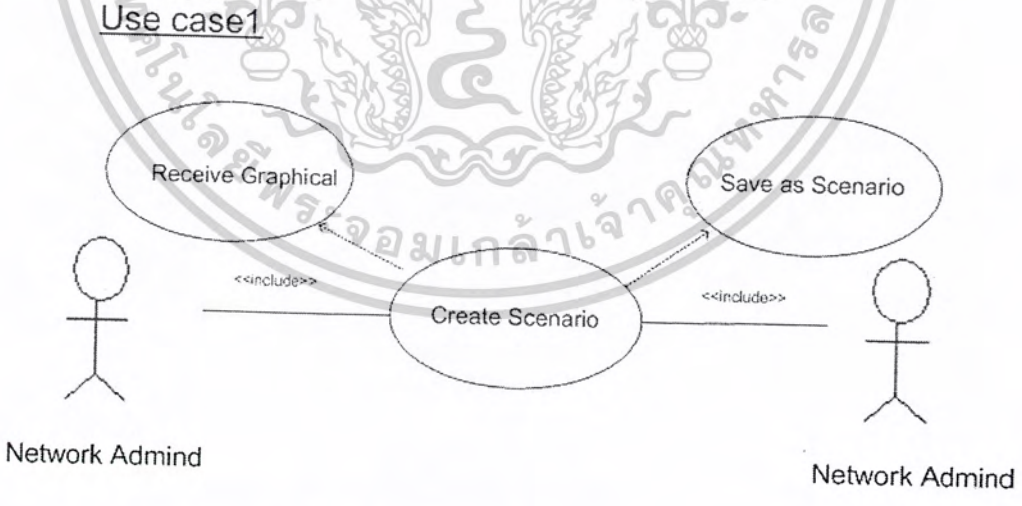
3.3 ส่วนที่ทำการสร้างขึ้นมาใหม่ในส่วนของการสร้างเหตุการณ์และแก้ไขเหตุการณ์ ใน Scenario

ส่วนที่ออกแบบในส่วนนี้ออกแบบเป็นยูสเซอร์อินเตอร์เฟส ที่ผู้ใช้สามารถสร้างเหตุการณ์ เพิ่มเหตุการณ์ และแก้ไขเหตุการณ์ ได้ง่าย ซึ่งค่าที่จำเป็นในการสร้างเหตุการณ์มีดังต่อไปนี้

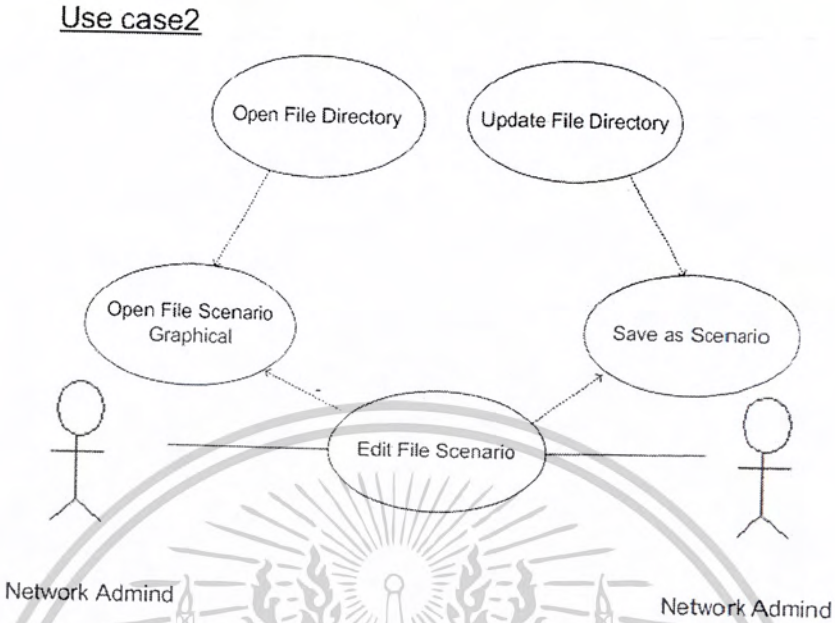
- หมายเลขไอพีต้นทาง
- หมายเลขสับเนตมาคต้นทาง
- หมายเลขไอพีปลายทาง
- หมายเลขสับเนตมาคปลายทาง
- เวลาที่ต้องการให้เกิดเหตุการณ์
- ข้อมูลที่ต้องการส่งไป
- ขนาดของข้อมูลที่ต้องการส่ง

ซึ่งการออกแบบอินเตอร์เฟสในแต่ละส่วนคำนึงถึงความสะดวกแก่ผู้ใช้งาน

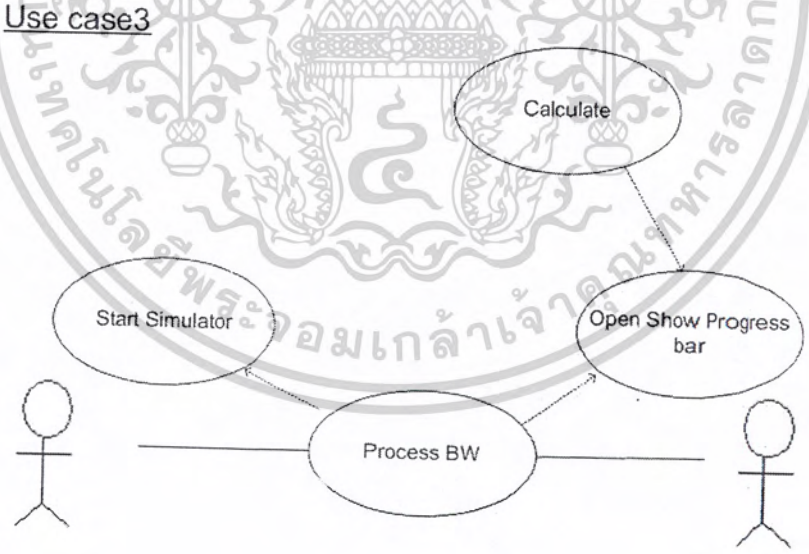
3.4 ยูสเคสไดอะแกรม (USE CASE)



รูปที่ 3.2 แสดงยูสเคสไดอะแกรม

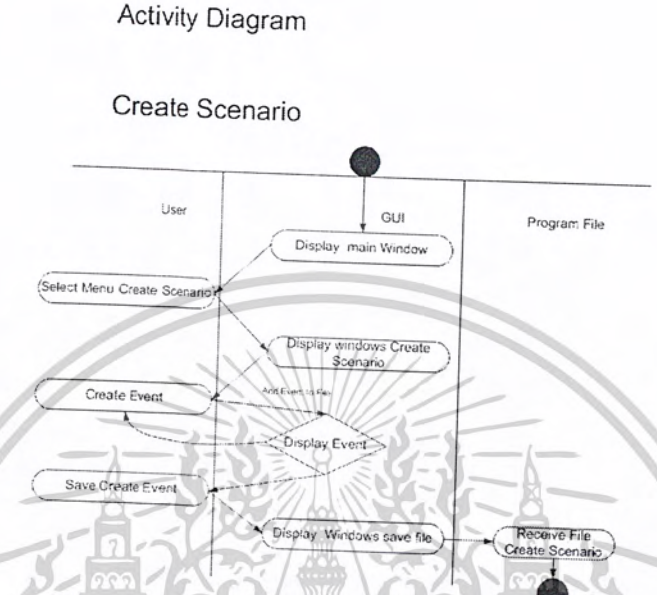


รูปที่ 3.3 แสดงยูสเคสไดอะแกรม

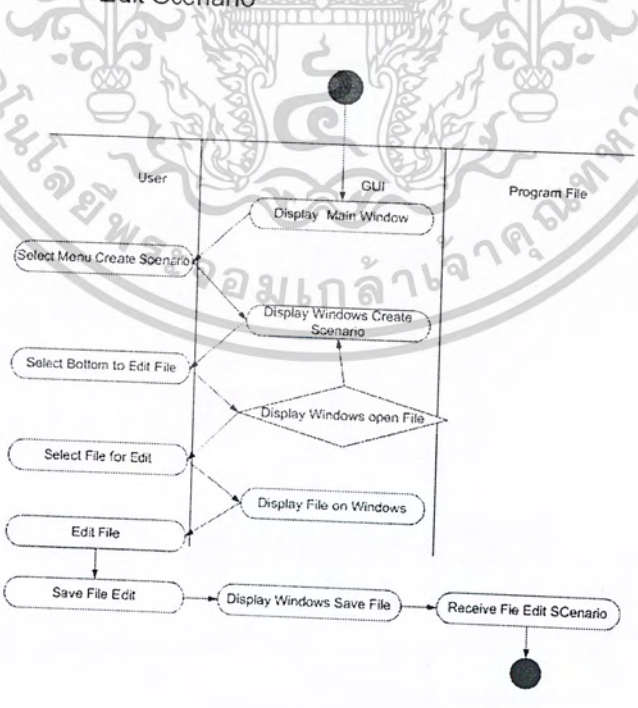


รูปที่ 3.4 แสดงยูสเคสไดอะแกรม

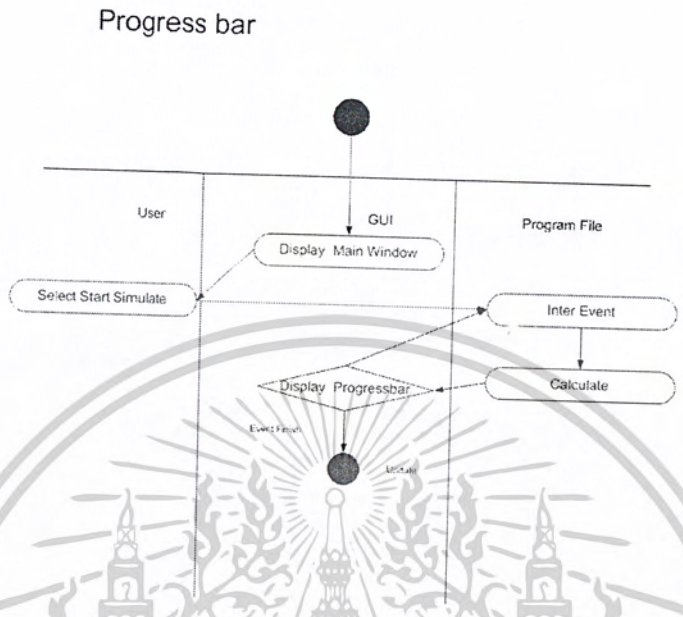
3.5 แอคติวิตี้ไดอะแกรม (Activity Diagram)



รูปที่ 3.5 แสดงแอคติวิตี้ไดอะแกรม Create Scenario



รูปที่ 3.6 แสดงแอคติวิตี้ไดอะแกรม Edit Scenario



รูปที่ 3.7 แสดงแอคติวิตี้ไดอะแกรม Progress bar

บทที่4

ผลการทดลอง

สำหรับบทนี้จะแบ่งเนื้อหาออกเป็น 2 ส่วนคือ

- ส่วนที่แสงโมดูลการทำงานของโปรแกรม

  การเริ่มทำการเรียกรันโปรแกรม

  เข้าสู่โปรแกรม

  ส่วนของหน้าต่างหลักประกอบด้วยเครื่องมือดังนี้

- ส่วนที่ทำการทดลองและผลการทดลอง

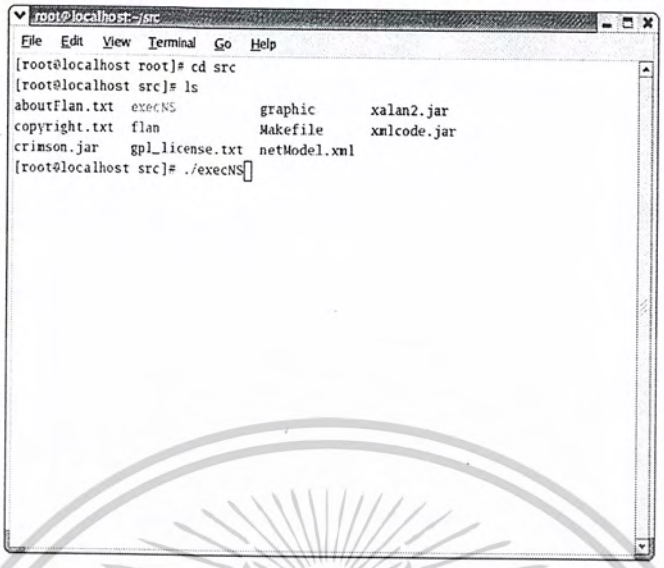
4.1 ส่วนที่แสงโมดูลการทำงานของโปรแกรม

4.1.1 การเริ่มทำการเรียกรันโปรแกรม

เนื่องจากการพัฒนาของโปรเจ็คนี้ ได้ทำการพัฒนาบนระบบปฏิบัติการลินุกซ์ (Linux) ผู้ใช้สามารถพิมพ์คำสั่งผ่านหน้าคอมมานด์ในสับไดเร็คทอรีรูท

คำสั่ง

- cd src
- ./execNS

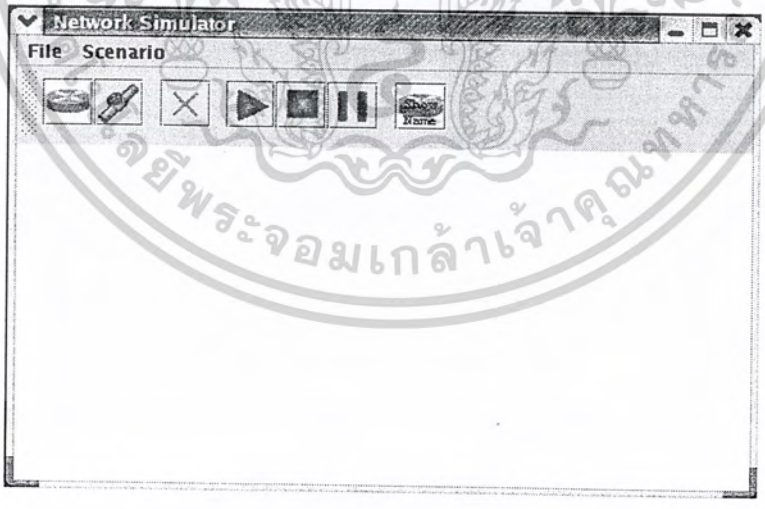


รูปที่ 4.1 แสดงการรันเรียกโปรแกรม

4.1.2 เข้าสู่โปรแกรม

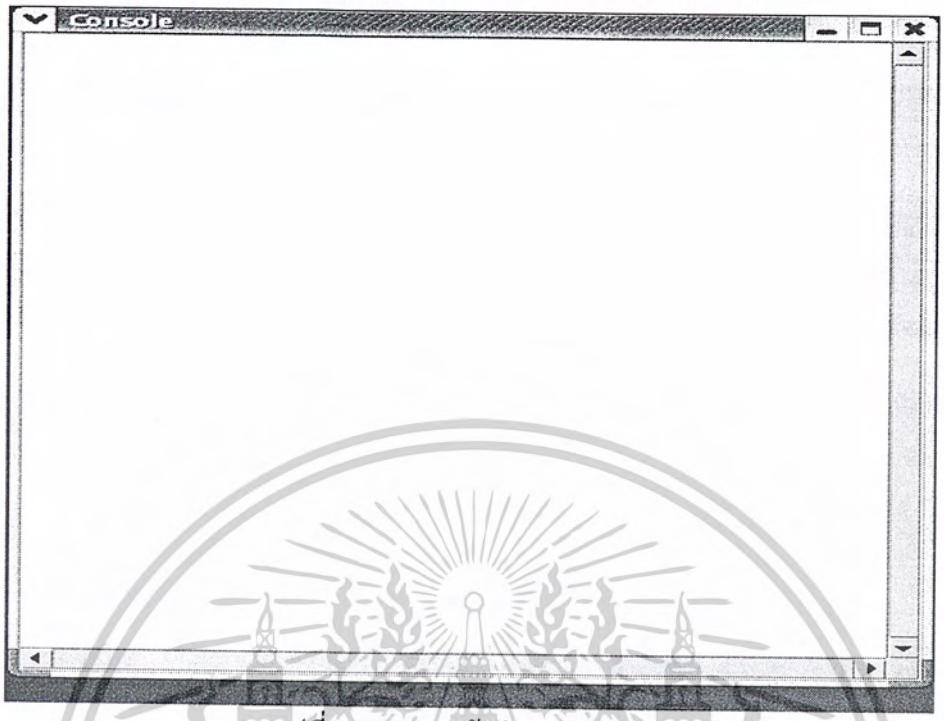
หลังจากทำการรันเพื่อเรียกโปรแกรมแล้วจะปรากฏหน้าต่าง 3 ส่วนคือ

ส่วนที่ 1 เป็นหน้าต่างหลักสำหรับผู้ใช้ในการเรียกใช้เครื่องมือต่าง ๆ



รูปที่ 4.2 แสดงหน้าต่างผู้ใช้หลัก

ส่วนที่ 2 เป็นหน้าต่างส่วนที่สำหรับแสดงสถานะขณะที่รันทดสอบการจำลองระบบเน็ตเวิร์คที่ได้ออกแบบ



รูปที่ 4.3 แสดงหน้าต่างคอนโซล

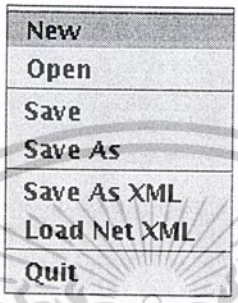
ส่วนที่ 3 เป็นหน้าต่างสำหรับแสดงกราฟการเกิดโพรเซสทั้งหมดของเหตุการณ์ที่ได้จำลองทั้งหมด โดยมีแกนนอนแสดงเวลา แนวแกนตั้งแสดงสเกลของข้อมูลที่ส่ง

รูปที่ 4.4 แสดงหน้าต่างกราฟ

4.1.3 ส่วนของหน้าต่างหลักประกอบด้วยเครื่องมือดังนี้

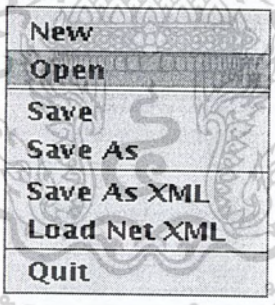
4.1.3.1. File จะมีเมนูให้เลือกทำดังนี้

- เลือกเมนู New หน้าจอใหม่



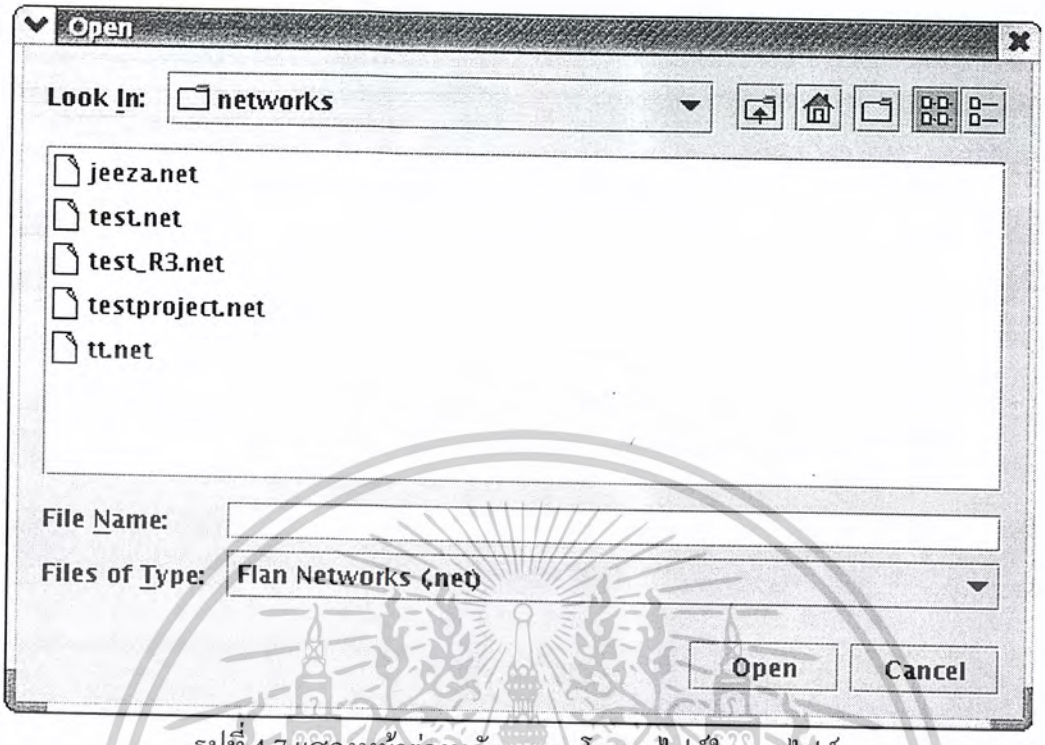
รูปที่ 4.5 แสดงหน้าต่างไฟล์ ส่วนของเมนู New

- เลือกเมนู Open เพื่อเลือกไฟล์ที่ทำการออกแบบเน็ตเวิร์คที่ได้เคยออกแบบไว้



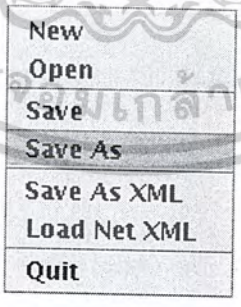
รูปที่ 4.6 แสดงเมนูไฟล์ Open

- หลังจากนั้นก็จะเปิดหน้าต่างมาให้เลือกไฟล์ที่เคยสร้างไปแล้ว



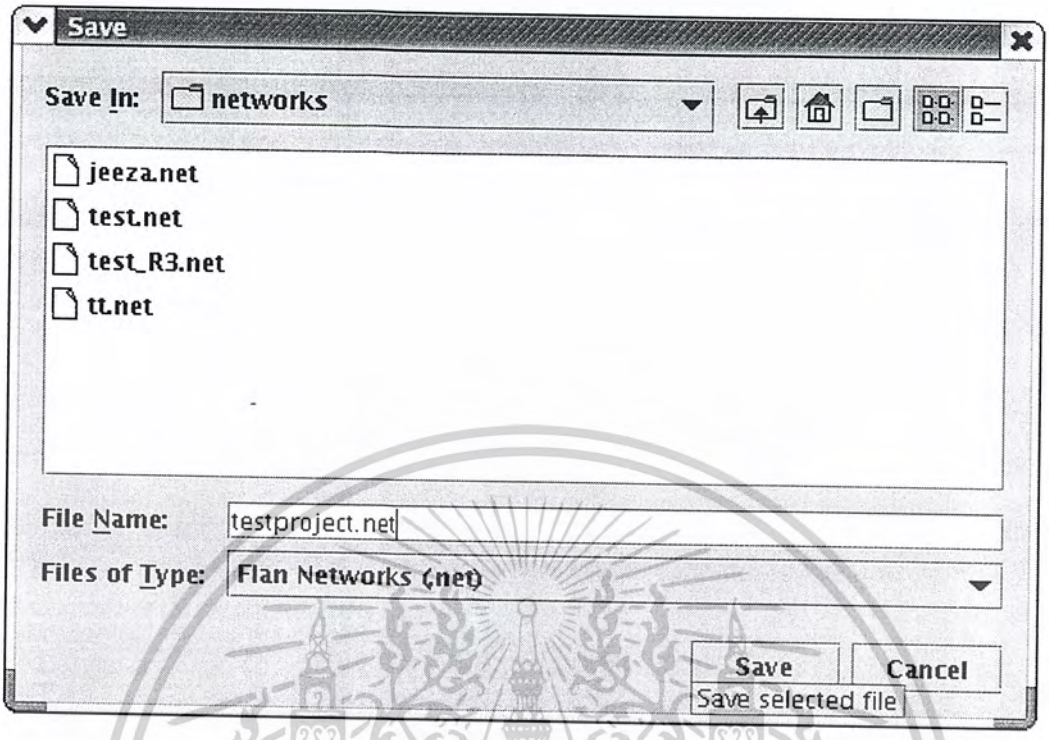
รูปที่ 4.7 แสดงหน้าต่างหลังจากกดโอเพนไฟล์ในเมนูไฟล์

- เลือกเมนู save ได้ก็ต่อเมื่อได้เปิดไฟล์ที่เคยสร้างไปแล้ว แล้วมีการแก้ไข แล้วต้องการเซฟที่ได้แก้ไขใหม่ทับไฟล์เดิม
- เลือกเมนู save as เมื่อได้ทำการออกแบบระบบเน็ตเวิร์คตามที่ต้องการแล้ว ต้องการจะเก็บระบบเน็ตเวิร์คที่ได้ออกแบบนั้นไว้ก็ให้เลือกเมนู save as



รูปที่ 4.8 แสดงเมนู save as ในเครื่องมือไฟล์

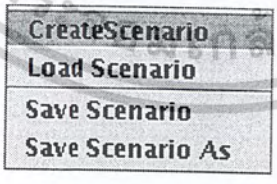
หลังจากนั้นก็จะเกิดหน้าต่างหลังจากกดเมนู save as ขึ้นมาเพื่อตั้งชื่อไฟล์ที่ต้องการจะเก็บ โดยไฟล์นั้นจะต้องมีจุดนามสกุลเป็น .net เท่านั้น



รูปที่ 4.9 แสดงหน้าต่างการ save as ไฟล์

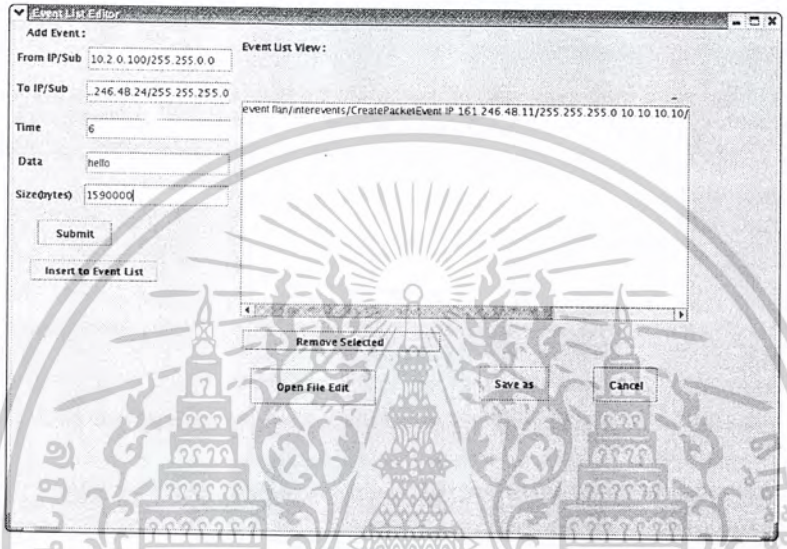
4.1.3.2. Scenario เป็นเมนูที่ใช้ในการเลือกการสร้างเหตุการณ์ในการทดสอบระบบเน็ตเวิร์คที่ได้ทำการออกแบบไป โดยจะมีเมนูย่อยให้เลือกดังนี้

- Create Scenario เป็นเมนูย่อยเพื่อใช้ในการสร้างเหตุการณ์ใหม่สำหรับการทดสอบระบบเน็ตเวิร์คที่ได้ออกแบบ



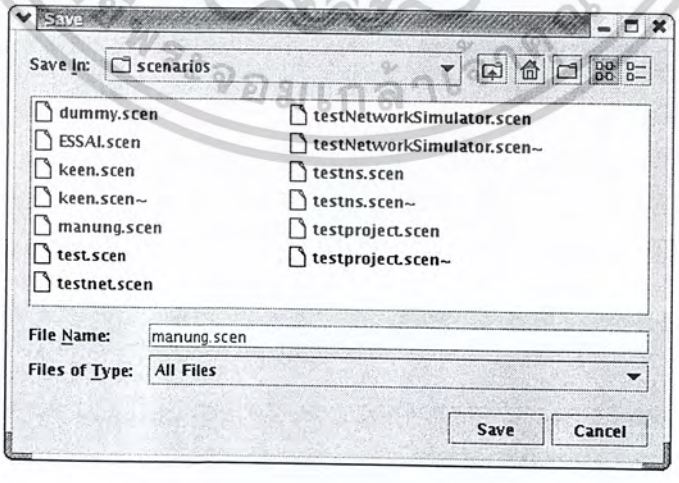
รูปที่ 4.10 แสดงเมนู Create Scenario

หลังจากที่เราเลือกเมนู Create Scenario แล้วก็จะเกิดหน้าต่างสำหรับสร้างเหตุการณ์ต้องการทดสอบส่งค่าต่าง ๆ ในระบบเน็ตเวิร์ค ลงในช่องรับ ประกอบด้วย ช่องรับค่าไอพีและสับเนตมาคของต้นทาง ช่องรับค่าไอพีของสับเนตมาคปลายทาง ช่องรับเวลาที่ต้องการส่งข้อมูล ช่องรับข้อมูลที่ต้องการส่งไป และช่องรับขนาดของข้อมูลซึ่งมีหน่วยเป็นไบต์ หลังจากนั้นก็กดปุ่ม Submit ตามด้วยกดปุ่ม Insert to Event List เพื่อส่งค่าเหตุการณ์ที่ได้กำหนดไปแสดงในช่องของเทคแอเรีย



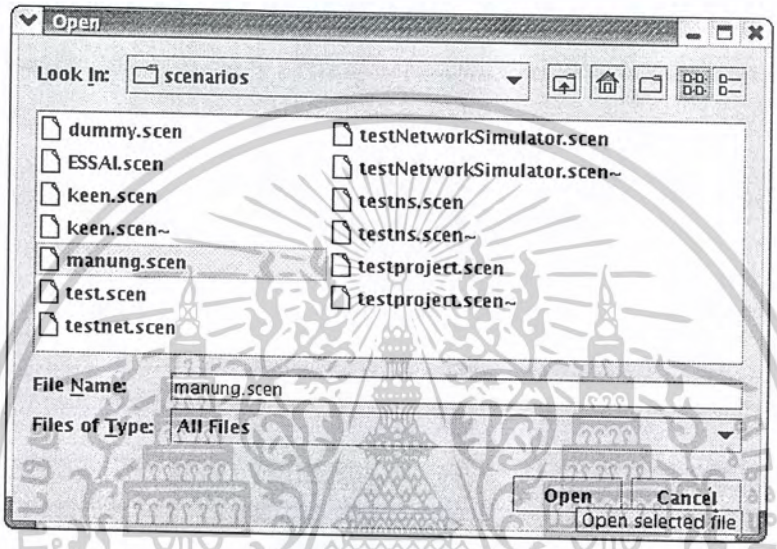
รูปที่ 4.11 แสดงหน้าต่างของการสร้างเหตุการณ์ใหม่

หลังจากที่เราได้ทำการสร้างเหตุการณ์แล้วจะต้องมีการบันทึกเหตุการณ์ที่ได้ออกแบบ โดยกดบันทึกที่ปุ่ม Save as ก็จะเกิดต่างขึ้นมา ดังรูปที่ 4.12



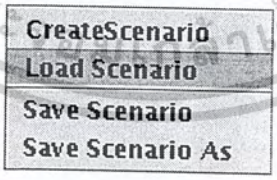
รูปที่ 4.12 แสดงหน้าต่างของการบันทึก

เมื่อเราต้องการแก้ไขหรือเพิ่มเหตุการณ์ที่สร้างขึ้นก็กดที่ปุ่ม Open Edit File ไฟล์ที่ต้องการจะแก้ไขก็จะถูกเปิดขึ้นมาบนเทคแอเรียดังรูป



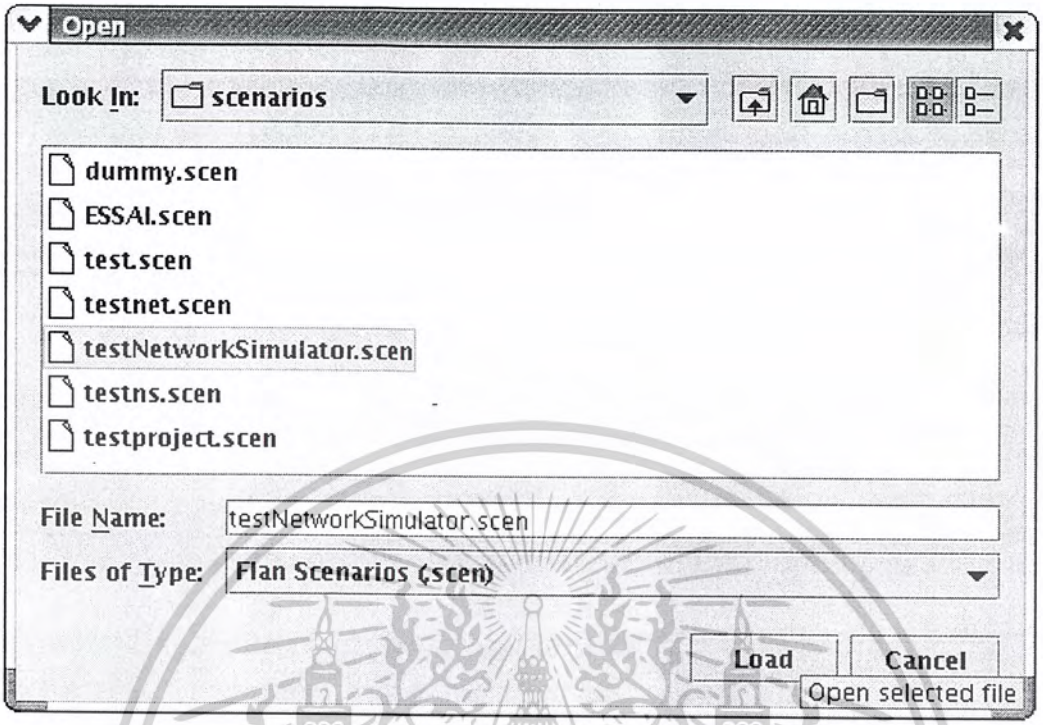
รูป 4.13 แสดงหน้าต่างการแก้ไขหรือการเพิ่มเหตุการณ์

- Load Scenario ใช้การเรียกไฟล์เหตุการณ์ที่ได้สร้างเอาไว้มาใช้เพื่อรันทดสอบระบบเน็ตเวิร์คที่ได้ออกแบบจำลองไว้



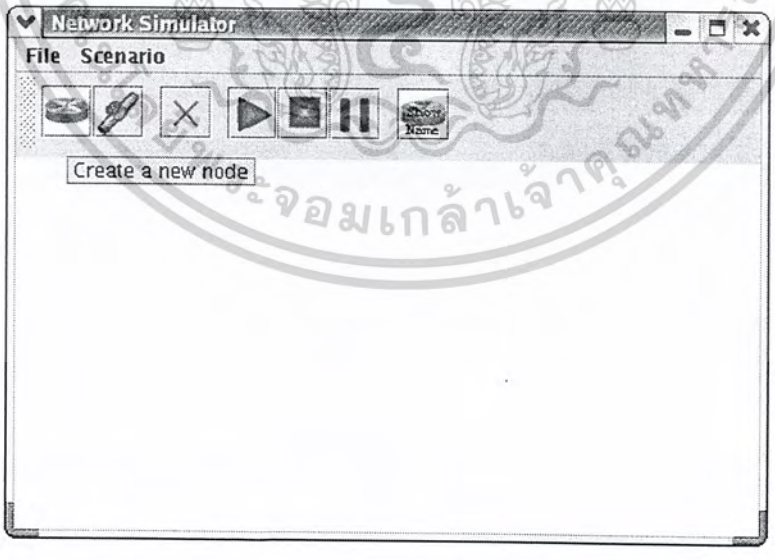
รูปที่ 4.14 แสดงเมนูLoad Scenario

หลังจากนั้นก็จะเปิดหน้าต่างสำหรับเลือกไฟล์ที่เก็บไว้ในไดเร็คทอรี ที่เคยสร้างไฟล์เหตุการณ์ไปเก็บไว้



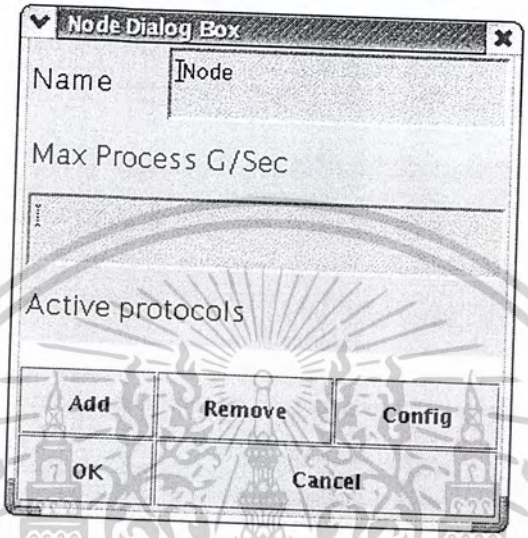
รูปที่ 4.15 แสดงหน้าต่างเลือกเหตุการณ์ที่ทำการทดสอบเน็ตเวิร์คที่จำลอง

4.1.3.3. Icon Node ใช้ในการสร้างเราเตอร์



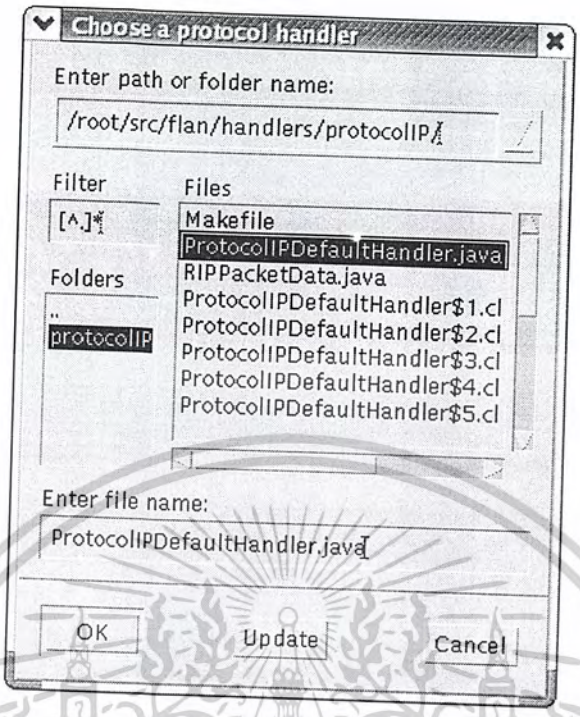
รูปที่ 4.16 แสดง Icon ของเราเตอร์

เมื่อคลิกที่ไอคอนเราเตอร์ก็จะเกิดหน้าต่างมาเพื่อให้ผู้ใช้กำหนดค่าต่าง ๆ ให้กับเราเตอร์ คือ ชื่อโพรเซสสูงสุดที่เราเตอร์สามารถทำงานได้ (G/Sec) สำหรับหน้าต่างนี้เรายังไม่ต้องมีการกำหนดค่าต่าง ๆ ให้ก็ได้ เราสามารถกำหนดค่าต่าง ๆ ได้หลังจากเราวาดแบบจำลองเสร็จเรียบร้อยแล้ว



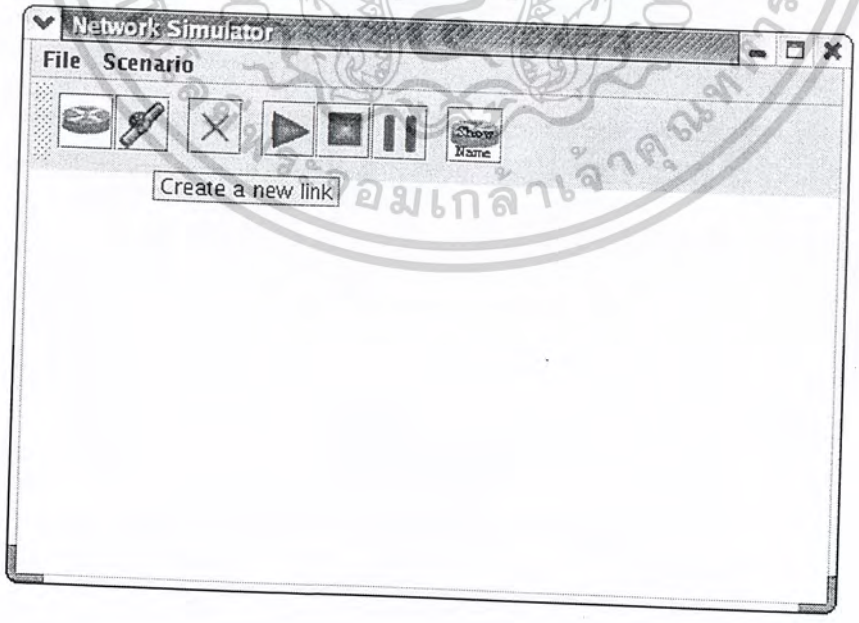
รูปที่ 4.17 แสดงหน้าต่างสำหรับกำหนดค่าต่าง ๆ ให้กับเราเตอร์

เราจะทำการเลือกโปรโตคอลสำหรับนำส่งข้อมูลด้วย ไอพีโปรโตคอล จากรูปที่ 4.15 เมื่อเราทำการคลิคเลือกที่ปุ่ม Add ก็จะเปิดหน้าต่างสำหรับกำหนดโปรโตคอล โดยเลือกไฟล์ที่ชื่อ ProtocolIPDefaultHandler.java หลังจากนั้นก็เลือกปุ่ม ok



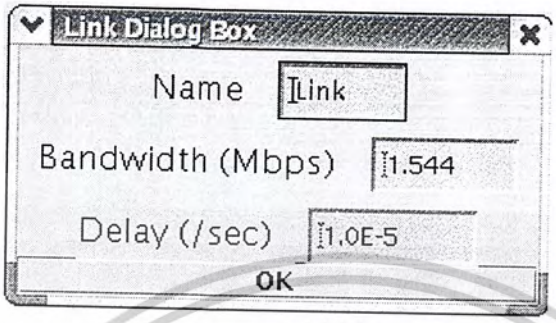
รูปที่ 4.18 แสดงหน้าต่างของการเลือกไอพีโปรโตคอล

4.1.3.4. Icon Link เป็นไอคอนที่ใช้ในการสร้างลิงค์เชื่อมต่อระหว่างเราเตอร์โดยสามารถกำหนดชื่อลิงค์ แบนวิธ และค่าดีเลย์เวลาได้



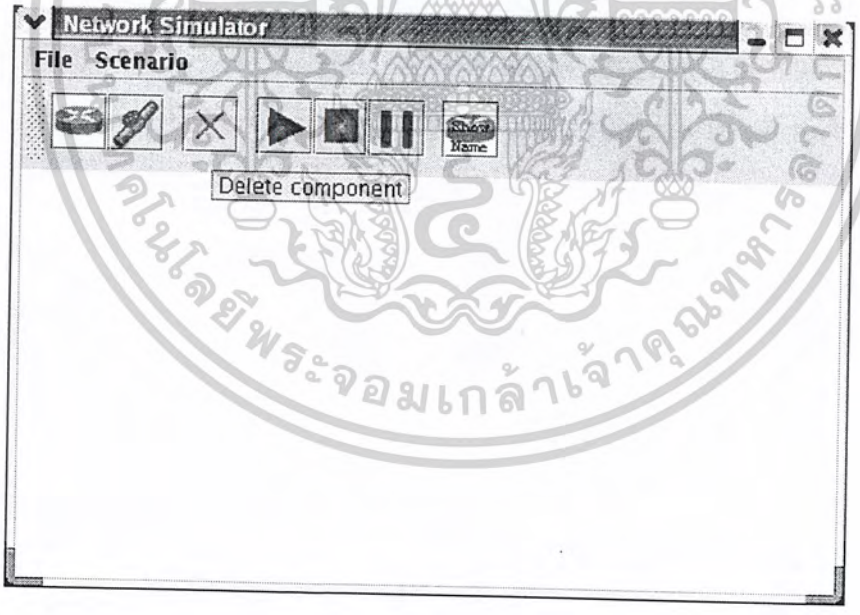
รูปที่ 4.19 แสดง Icon Link

หลังจากกดไอคอนลิงค์ก็จะเกิดหน้าต่างสำหรับกำหนดค่าให้กับลิงค์



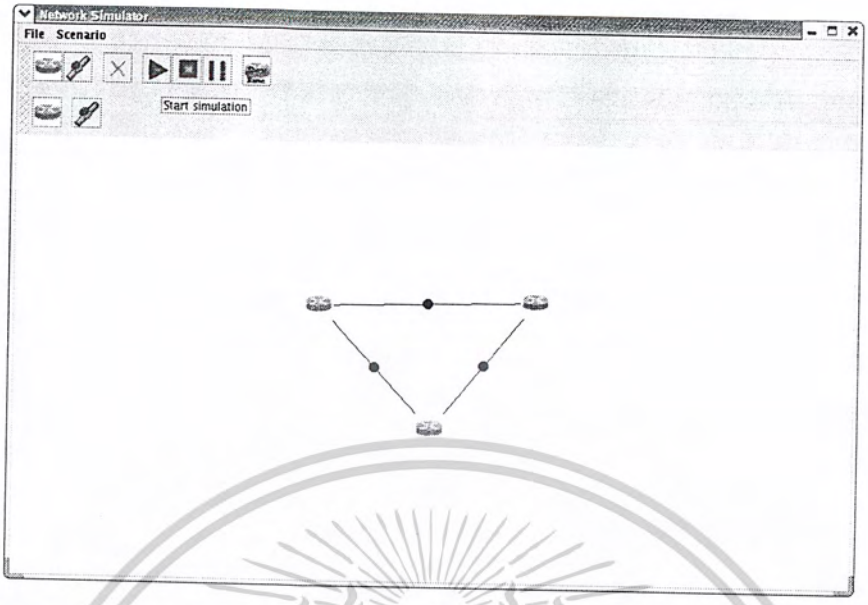
รูปที่ 4.20 แสดงหน้าต่างสำหรับกำหนดค่าต่าง ๆ ให้กับลิงค์

4.1.3.5. Icon Delete ใช้สำหรับลบรูปของเราเตอร์ที่เราไม่ต้องการเวลาออกแบบ แบบจำลองระบบเน็ตเวิร์ค โดยคลิกที่ไอคอนดิลิท แล้วค่อยไปเลือกรูปของเราเตอร์ที่ต้องการลบ



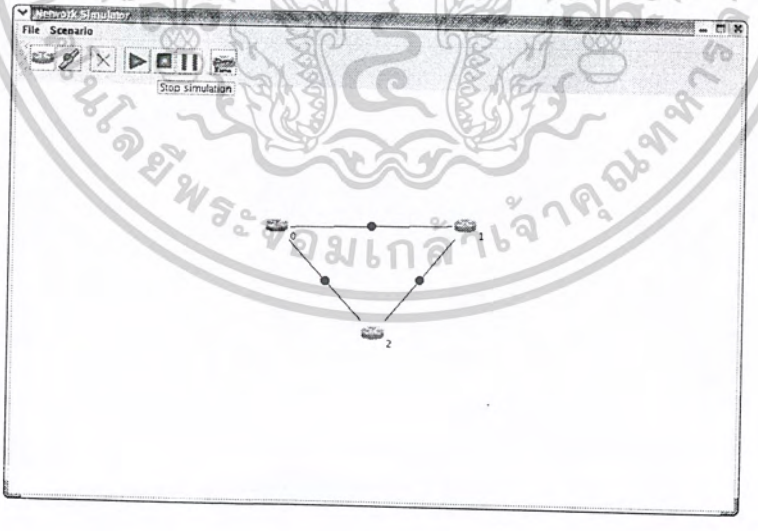
รูปที่ 4.21 แสดงไอคอนดิลิท

4.1.3.6. Icon Start ใช้สำหรับการเริ่มต้นการรันโปรแกรม



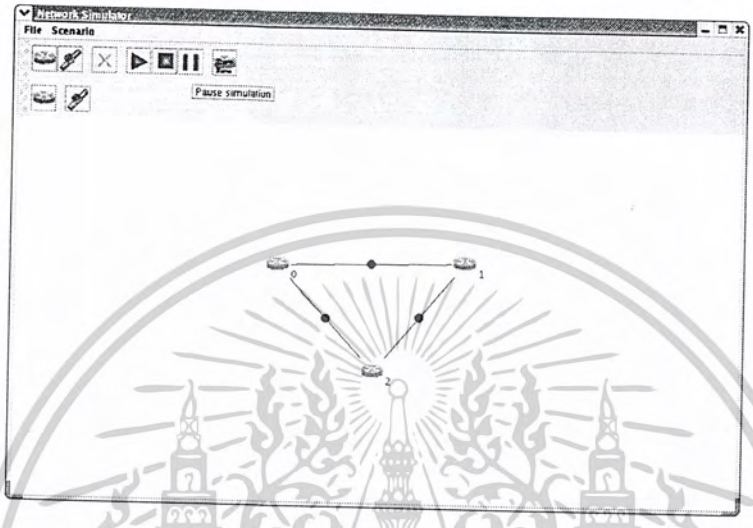
รูปที่ 4.22 แสดงไอคอนที่ใช้ในการเริ่มรันระบบจำลองเน็ตเวิร์ค

4.1.3.7. Icon Stop ใช้ในกรณีที่เราต้องการหยุดเหตุการณ์ที่กำลังทดสอบระบบเน็ตเวิร์คที่กำลังรันอยู่



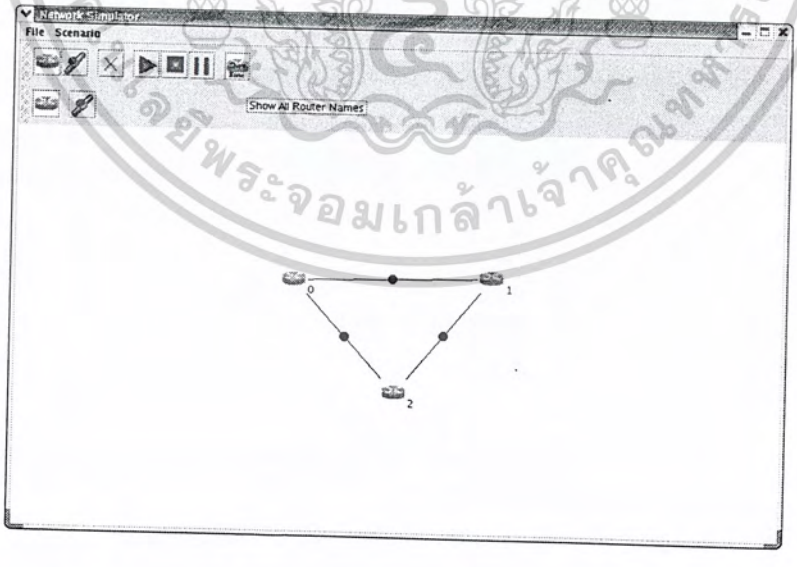
รูปที่ 4.23 แสดงไอคอนสำหรับกดพื่อหยุดการรันทดสอบเหตุการณ์กับแบบจำลองระบบเน็ตเวิร์ค

4.1.3.8. Icon Pause ใช้ในกรณีที่เราต้องการหยุดเหตุการณ์ที่กำลังรันทดสอบแบบจำลองระบบเน็ตเวิร์คที่กำลังรันอยู่ แต่เมื่อกดไอคอนสตาร์ทอีกครั้งก็จะทำการรันเหตุการณ์ต่อไปหลังจากเหตุการณ์ที่ได้หยุดไป



รูปที่ 4.24 แสดงไอคอนสำหรับหยุดเหตุการณ์ที่กำลังรันอยู่ชั่วคราว

4.1.3.9. Icon Showname ใช้ในการแสดงชื่อของเราเตอร์ที่ได้กำหนดไว้



รูปที่ 4.25 แสดงไอคอนสำหรับแสดงชื่อของเราเตอร์ที่ออกแบบ

4.2 ส่วนที่ทำการทดลองและผลการทดลอง

การทดลอง

- ทดลองสร้างเราเตอร์ 3 ตัวดังนี้

Router 1

- Name : 0
- Max Process : 1 G/Sec
- IP Interface 1 : 10.2.0.4/255.255.0.0
- IP Interface 2 : 10.2.0.100/255.255.0.0

Router 2

- Name : 1
- Max Process : 2 G/Sec
- IP Interface 1 : 161.246.48.11/255.255.255.0
- IP Interface 2 : 161.246.48.24/255.255.255.0

Router 3

- Name : 2
- Max Process : 2 G/Sec
- IP Interface 1 : 10.10.10.10/255.255.255.255
- IP Interface 2 : 10.10.10.20/255.255.255.255

Link ทุกเส้นมีค่าแบนวิธ 1.554 Mbps และ ค่าดีเลย์เวลา มีค่าเท่ากับ 1.0E-5 โดยมีเหตุการณ์ทั้งหมดดังนี้

event flan/events/RIPRefreshAllEvent 0 30

event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255 161.246.48.24/255.255.255.0 5 hello 100000000

event flan/interevents/CreatePacketEvent IP 161.246.48.24/255.255.255.0 10.2.0.4/255.255.0.0 5 hello 90500000

event flan/interevents/CreatePacketEvent IP 10.2.0.100/255.255.0.0 10.10.10.10/255.255.255.255 5 hello 99000000

event flan/interevents/CreatePacketEvent IP 161.246.48.24/255.255.255.0
10.10.10.10/255.255.255.255 6 hello 80000000

event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255 10.2.0.100/255.255.0.0 6
hello 99000000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0
161.246.48.24/255.255.255.0 6 hello 92000000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0
161.246.73.168/255.255.255.0 6 hello 85000000

event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255 10.2.0.4/255.255.0.0 8
hello 19000000

event flan/interevents/CreatePacketEvent IP 10.2.0.100/255.255.0.0 10.10.10.10/255.255.255.255 8
hello 19000000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0
161.246.48.11/255.255.255.0 8 hello 19000000

event flan/interevents/CreatePacketEvent IP 10.2.0.2/255.255.0.0 161.246.73.168/255.255.255.0 6
hello 92000000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.4/255.255.0.0 8
hello 150000000

event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255
161.246.48.24/255.255.255.0 10 hello 19900000

event flan/interevents/CreatePacketEvent IP 10.10.10.20/255.255.255.255
161.246.48.24/255.255.255.0 10 hello 2910000

event flan/interevents/CreatePacketEvent IP 161.246.48.24/255.255.255.0 10.2.0.100/255.255.0.0 12
hello 3900000

event flan/interevents/CreatePacketEvent IP 10.2.0.100/255.255.0.0 10.10.10.10/255.255.255.255 13
hello 3990000

event flan/interevents/CreatePacketEvent IP 10.2.0.4/255.255.0.0 10.10.10.20/255.255.255.255 15
hello 5099000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0
161.246.48.11/255.255.255.0 15 hello 19000000
event flan/interevents/CreatePacketEvent IP 10.2.0.100/255.255.0.0 10.10.10.10/255.255.255.255 16
hello 19000000
event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255
161.246.48.24/255.255.255.0 16 hello 19900000
event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255
161.246.48.24/255.255.255.0 18 hello 2900000
event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.100/255.255.0.0 19
hello 2990000
event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.100/255.255.0.0 20
hello 3900000
event flan/interevents/CreatePacketEvent IP 10.2.0.100/255.255.0.0 161.246.48.11/255.255.255.0 20
hello 10900000
event flan/interevents/CreatePacketEvent IP 161.246.48.24/255.255.255.0 10.2.0.100/255.255.0.0 20
hello 3900900
event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255 10.2.0.4/255.255.0.0 22
hello 19000000
event flan/interevents/CreatePacketEvent IP 10.2.0.100/255.255.0.0 10.10.10.10/255.255.255.255 23
hello 19000000
event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.100/255.255.0.0 24
hello 3900000
event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.100/255.255.0.0 25
hello 5090000
event flan/interevents/CreatePacketEvent IP 10.2.0.4/255.255.0.0 10.10.10.10/255.255.255.255 25
hello 3909000
event flan/interevents/CreatePacketEvent IP 10.2.0.100/255.255.0.0 10.10.10.10/255.255.255.255 29
hello 19000000

event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255
161.246.48.24/255.255.255.0 30 hello 2900000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.100/255.255.0.0 33
hello 5090000

event flan/interevents/CreatePacketEvent IP 10.2.0.4/255.255.0.0 10.10.10.10/255.255.255.255 35
hello 3909000

event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255 10.2.0.4/255.255.0.0 36
hello 19000000

event flan/interevents/CreatePacketEvent IP 10.2.0.100/255.255.0.0 10.10.10.10/255.255.255.255 38
hello 99000000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.100/255.255.0.0 40
hello 2990000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.100/255.255.0.0 42
hello 5090000

event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255 10.2.0.4/255.255.0.0 44
hello 19000000

event flan/interevents/CreatePacketEvent IP 10.2.0.100/255.255.0.0 10.10.10.10/255.255.255.255 45
hello 19000000

event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255 10.2.0.4/255.255.0.0 46
hello 19000000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.100/255.255.0.0 46
hello 5090000

event flan/interevents/CreatePacketEvent IP 10.2.0.100/255.255.0.0 10.10.10.10/255.255.255.255 49
hello 19000000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.100/255.255.0.0 49
hello 5090000

event flan/interevents/CreatePacketEvent IP 10.2.0.100/255.255.0.0 10.10.10.10/255.255.255.255 50
hello 19000000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.100/255.255.0.0 50 hello 2990000

event flan/interevents/CreatePacketEvent IP 161.246.48.11/255.255.255.0 10.2.0.100/255.255.0.0 50 hello 5090000

event flan/interevents/CreatePacketEvent IP 10.10.10.10/255.255.255.255 161.246.48.24/255.255.255.0 50 hello 2900000

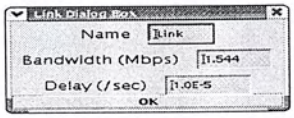
4.2.1 ทำการออกแบบบนโมดูลเครื่องมือ

4.2.1.1 ทำการออกแบบจำลองระบบเน็ตเวิร์ค

- เลือกไอคอนเราเตอร์และเลือกไอคอนลิงค์

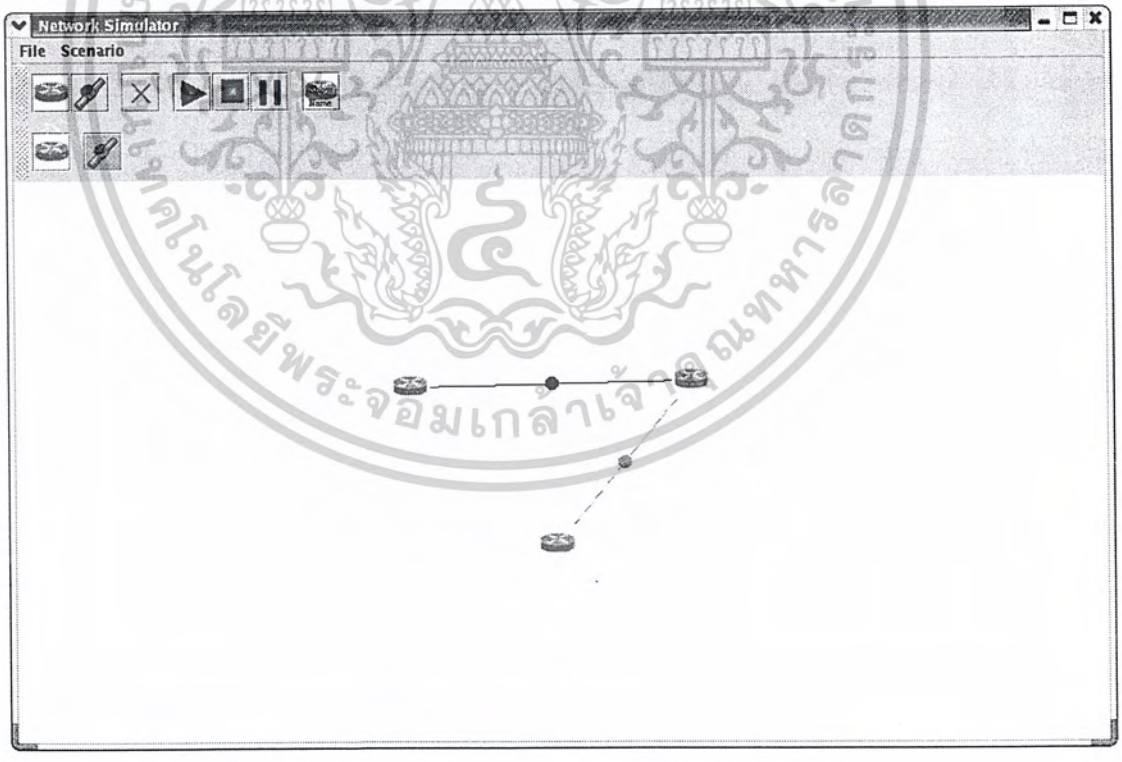
รูปที่ 4.26 แสดงการเลือกไอคอนจากเมนูเครื่องมือของเราเตอร์และลิงค์

กำหนด Link ทุกเส้นมีค่าแบนวิธ 1.554 Mbps และ ค่าดีเลย์เวลา มีค่าเท่ากับ 1.0E-5



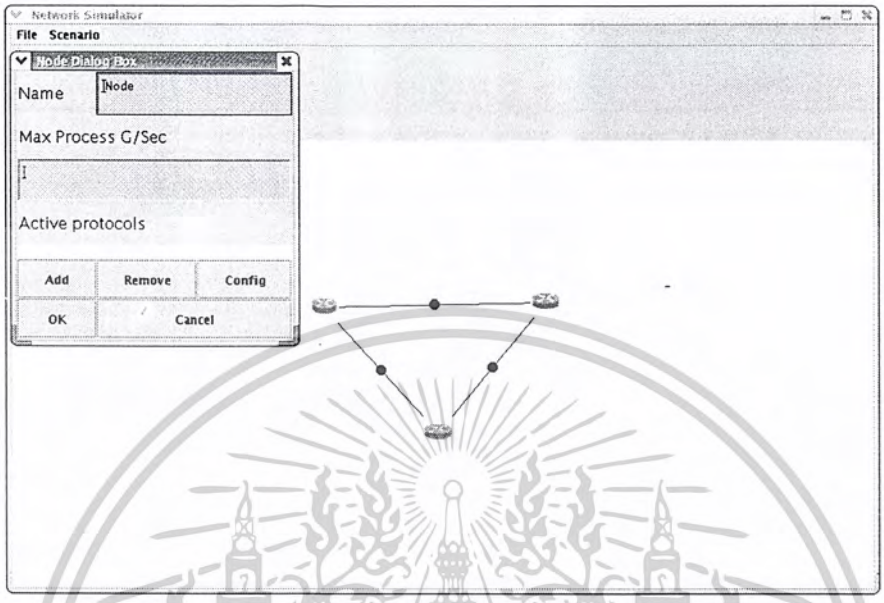
รูปที่ 4.27 แสดงหน้าต่างกำหนดค่าต่าง ๆ ให้กับลิงค์

- ทำการวาดเราเตอร์และลิงค์ทั้ง 3 ตัว



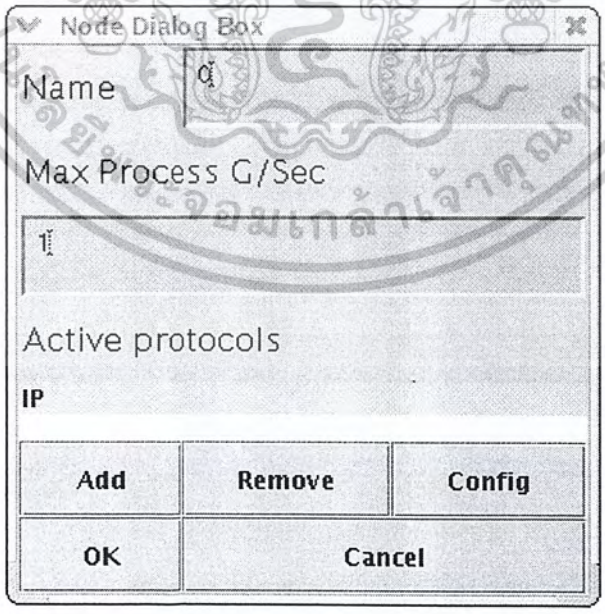
รูปที่ 4.28 แสดงการออกแบบจำลองระบบเน็ตเวิร์ค

- คลิกขวาที่เราเตอร์ที่ 1 เพื่อเปิดหน้าต่างในการกำหนดค่าต่าง ๆ



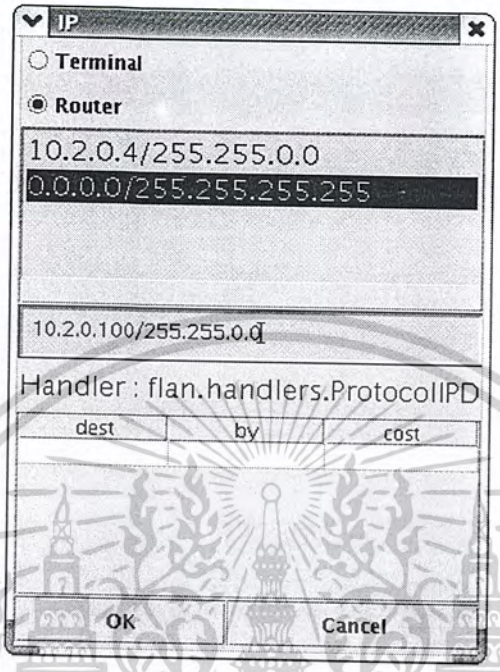
รูปที่ 4.29 แสดงคลิกขวาที่เราเตอร์ เพื่อเปิดหน้าต่างในการกำหนดค่าต่าง ๆ

- กำหนด ชื่อ และ ขนาดของแมคโพรเซสให้กับเราเตอร์ หลังจากนั้นก็คลิกที่ปุ่ม Add เลือกไฟล์โปรโตคอล

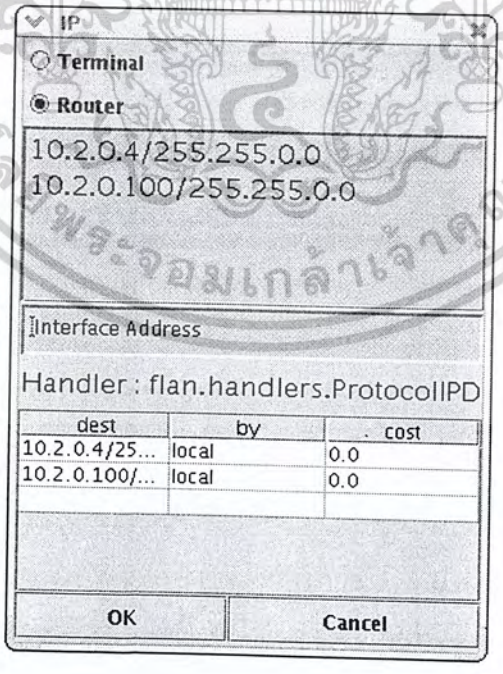


รูปที่ 4.30 แสดงหน้าต่างในการกำหนดค่าให้กับเราเตอร์

หลังจากนั้นก็คลิกที่ปุ่ม Config เพื่อกำหนดค่าไอพีให้กับอินเตอร์เฟสต่าง ๆ หลังจากนั้นก็คลิก ok

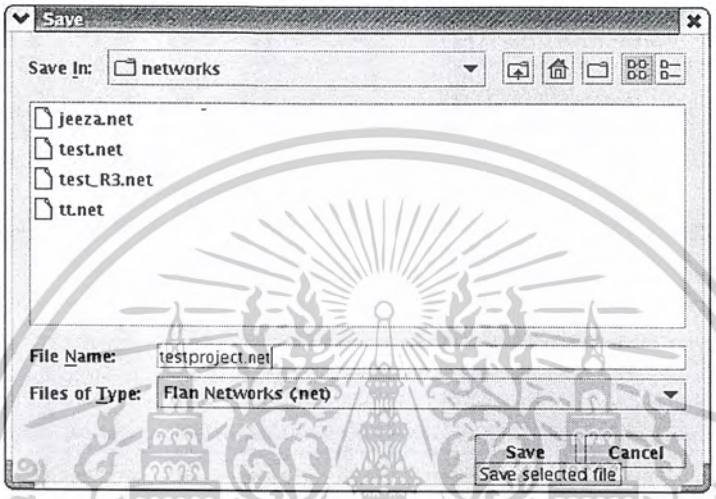


รูปที่ 4.31 แสดงการกำหนดค่าไอพีให้กับอินเตอร์เฟสของเราเตอร์



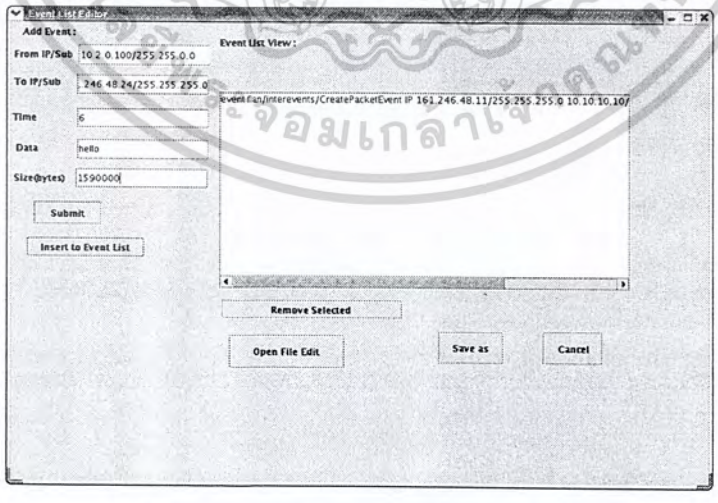
รูปที่ 4.32 แสดงค่าของเราติงเทเบิ้ลก่อนการรัน

แล้วทำการกำหนดค่าต่าง ๆ ที่กำหนดไว้ให้กับเราเตอร์ทุกตัวที่เหลือ ซึ่งก่อนที่มีการรันทดสอบ ค่าของเราติงเทเบิ้ล จะยังไม่ทำการรู้จักกับเราเตอร์ข้างเคียง และหากต้องการเซฟแบบจำลองที่ออกแบบไว้ก็ไปคลิกที่เมนูไฟล์ เลือก save as ก็จะเปิดหน้าต่างเพื่อทำการเซฟขึ้นมา โดยในตัวอย่างนี้จะเซฟเป็นชื่อ testproject.net



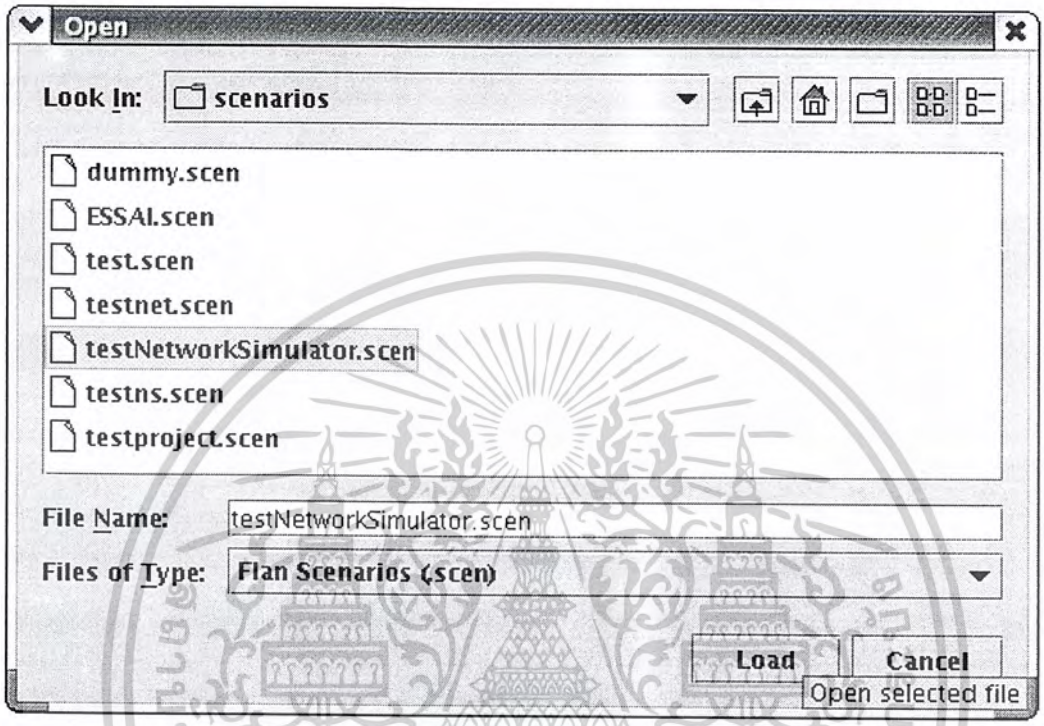
รูปที่ 4.33 แสดงหน้าต่างเพื่อทำการเซฟแบบจำลอง

- ทำการสร้างเหตุการณ์จำลองในเมนู Create Scenario ตามเหตุการณ์ที่กำหนดโดยหลังจากคลิก ok แล้วเหตุการณ์ที่ออกแบบจะถูกเก็บไว้ในไฟล์ชื่อ testNetworkSimulator.scen



รูปที่ 4.34 แสดงหน้าต่างสร้างเหตุการณ์ที่กำหนด

- ทำการโหลดเหตุการณ์ที่สร้างไว้เพื่อรัน โดยเลือกที่เมนู Scenario Load ก็จะเปิดหน้าต่างเพื่อเปิดไฟล์เหตุการณ์ขึ้นมา



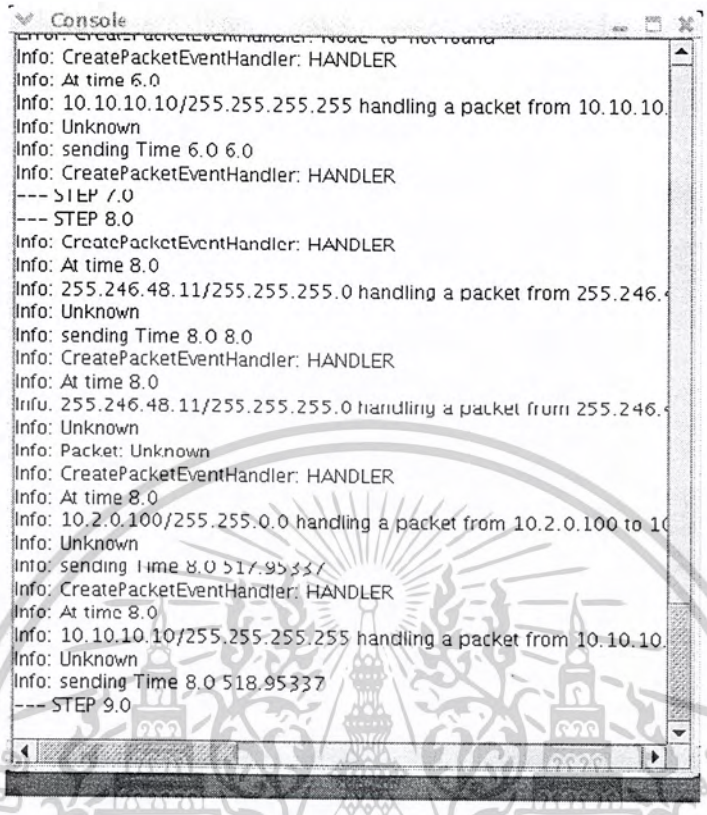
รูปที่ 4.35 แสดงหน้าต่างโหลดเหตุการณ์สร้างไว้มาใช้

- ทำการรันโปรแกรมโดยกดที่เมนูบาร์ คลิกที่ปุ่ม start

4.2.2 ผลการทดลอง

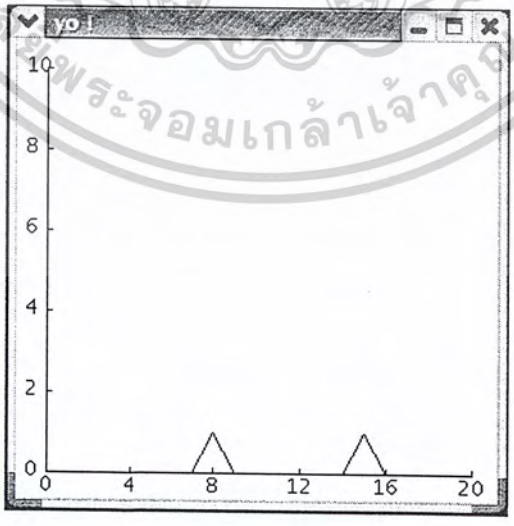
ในส่วนของผลการทดลองจะมีการแสดงผลต่าง ๆ ดังนี้

- ที่หน้าต่าง Console จะแสดงสถานะขณะที่รันโปรแกรมว่ามีการสร้างข้อมูลหรือเกิดเหตุการณ์ใดขึ้นในเวลาต่าง ๆ โดยจะแสดงผลออกมาเป็นในรูปแบบของเท็ค



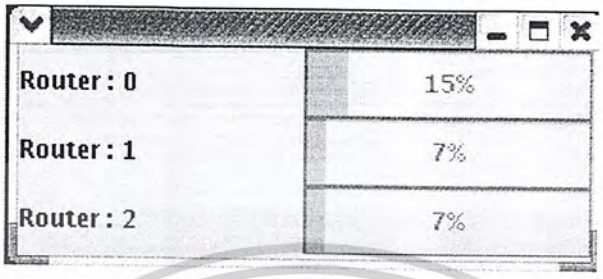
รูปที่ 4.36 แสดงหน้าต่าง console ขณะที่รันโปรแกรม

- สามารถดูการยูทิไลท์ ของเราเตอร์ทั้งหมดได้ว่ามีเหตุการณ์เกิดขึ้นเทียบกับเวลามากน้อยแค่ไหนดูได้จากหน้าต่างนั้น



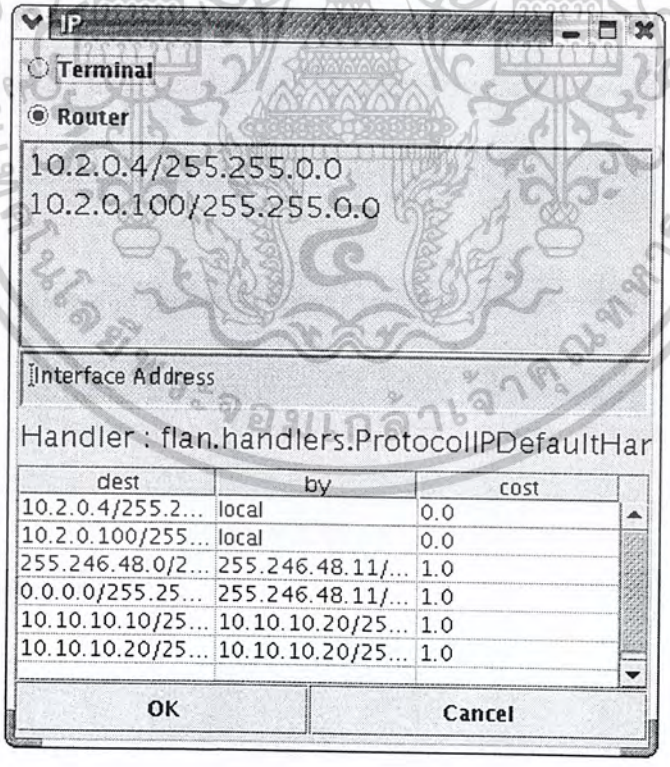
รูปที่ 4.37 แสดงยูทิไลท์ของทั้งหมด

- สามารถดูค่าของโพรเซสของแต่ละเราเตอร์ที่ถูกใช้ไปโดยดูได้จากโพรเกรสบาร์ที่ช่วงเวลาต่าง ๆ



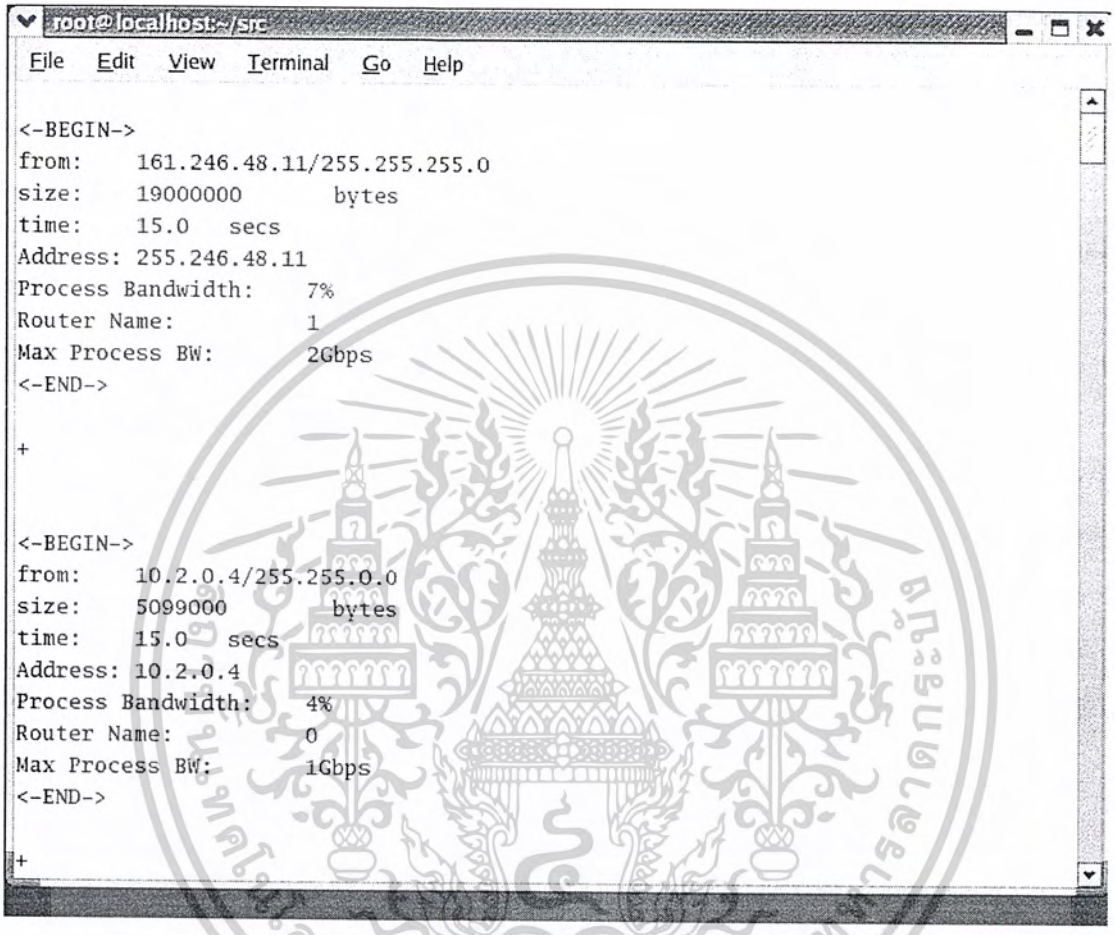
รูปที่ 4.38 แสดงโพรเกรสบาร์ของแต่ละเราเตอร์

- ส่วนตารางการหาเส้นทางหลังจากการรันก็จะมีการอัพเดทตารางการหาเส้นทางใหม่โดยสามารถรู้จักกับเราเตอร์ข้างเคียงด้วยตารางการหาเส้นทางแบบ RIP



รูปที่ 4.39 แสดงตารางการหาเส้นทางหลังการอัพเดทแล้วของเราเตอร์ตัวที่ 1

- เราสามารถดูค่าอื่น ๆ ในรูปแบบเท็คไฟล์จากหน้าต่างของเทอร์มินอลที่ใช้รันได้ เช่น มีการส่งไอพีอะไร ชื่อของเราเตอร์ โพรเซสของเราเตอร์ ณ เวลานั้น ค่ามากที่สุดของโพรเซสที่เราเตอร์



รูปที่ 4.40 แสดงผลทางหน้าเทอร์มินอล

บทที่ 5
สรุปและวิจารณ์ผลการทดลอง

โปรแกรมประยุกต์ที่ได้พัฒนาขึ้นมานี้ เป็นโปรแกรมที่ช่วยให้ผู้ใช้ในระดับเริ่มต้น ได้เรียนรู้เกี่ยวกับเรื่อง การกำหนด กำหนด ต่าง ๆ ได้ เช่น ชื่อ โหนด ,หมายเลข ไอพี ในเราเตอร์ การเพิ่ม อินเตอร์เฟส ไปใน เราเตอร์ หรือการกำหนด โปรโตคอล เราติ้งอินฟอร์เมชั่นโปรโตคอล(RIP) ที่ใช้ในการ หาเส้นทาง โดยโปรแกรมสามารถ กำหนดการจัดการด้าน การควบคุมความคับคั่งของทราฟฟิกของเราเตอร์ได้ ในส่วนของ ลิงค์ เป็นการเชื่อมต่อระหว่างเราเตอร์ และกำหนดแบนด์วิธบนลิงค์ โดยแบบจำลอง สามารถจำลองการส่งข้อมูลแบบ IP ผ่านเราเตอร์และลิงค์ โดยกำหนดปริมาณข้อมูล,ช่วงเวลาและสามารถสร้างเหตุการณ์จำลองที่ต้องการทดสอบระบบเครือข่ายที่ได้ออกแบบได้ทีละหลายเหตุการณ์ผ่านหน้าอินเทอร์เฟส โดยแสดงการยูทิไลทซ์ของแต่ละเราเตอร์ ว่าถูกใช้งานแบนด์วิธไปกี่เปอร์เซ็นต์ โดยแสดงด้วย gressbar ปริมาณการส่งข้อมูลแต่ละการเชื่อมต่อในรูปแบบของกราฟ ซึ่งทำให้ผู้จัดทำได้รับความรู้ ประสบการณ์ การรับรู้ถึงปัญหาต่าง ๆ ทั้งจากการเรียนรู้การทำงานของระบบปฏิบัติการลินุกซ์ และการเขียนโปรแกรมด้วยภาษาจาวา ทำให้เห็นภาพรวมของการทำงานทั้งสองส่วนได้ดียิ่งขึ้น และสามารถพัฒนาการทำงานในอนาคตได้

5.1 ปัญหาในการทดลอง

5.1.1 ในส่วนของการเขียน JAVA Programming Language มีความยุ่งยากในการจัดการและการเขียน

5.1.2 ทฤษฎีเครือข่าย เป็นเรื่องละเอียดอ่อน ต้องศึกษาหาความรู้พื้นฐานจากหลาย ๆ ที่ และใช้เวลานานในการทำความเข้าใจ

5.1.3 ในการทำงานของระบบโดยรวมยังมีปัญหาเกิดขึ้นในบางส่วน

5.2 แนวทางแก้ไขปัญหา

5.2.1 ศึกษาค้นคว้าเพิ่มเติม ใช้เวลาให้มากในการเรียนรู้ พยายามเขียนและแบ่งงานออกเป็นส่วน ๆ เพื่อง่ายในการจัดการ

5.2.2 ศึกษา ค้นคว้า สอบถามจากผู้รู้ และพยายามทำความเข้าใจเป็นพิเศษในส่วนที่เกี่ยวข้องกับโครงงานเพื่อความรวดเร็ว

5.2.3 ทำการพัฒนาซอฟท์แวร์และฮาร์ดแวร์เพื่อแก้ปัญหาของระบบ และทำให้ระบบมีเสถียรภาพมากขึ้น

5.3 แนวทางการพัฒนา

5.3.1 แก้ไขการทำงานของโครงการ โดยรับค่าต่าง ๆ ของระบบ ทำให้สามารถจำลองเหตุการณ์ต่าง ๆ ได้หลากหลายยิ่งขึ้น

5.3.2 ทำการศึกษาและ เพิ่ม โปรโตคอล ทำให้สามารถแสดงผลในรูปแบบที่หลากหลายขึ้น

5.3.3 สามารถดัดแปลงและนำไปใช้คาดเดาความน่าจะเป็นในการเกิดเหตุการณ์ต่าง ๆ เพื่อนำไปใช้ในเชิงพาณิชย์ได้

บรรณานุกรม

1. วรวุฒิ เที่ยงธรรม, สันติ ศรีลาศักดิ์, "*รู้จักลีนุกซ์*" : สำนักพิมพ์ออฟเซ็ทเพรส, 2542

2. วรรณิกา เนตรงาม, "*คู่มือในการเขียนโปรแกรมภาษา JAVA*" : สำนักพิมพ์อินโฟเพรส,2545

3. วีระศักดิ์ ซึงถาวร, "*Java Prog amming Volume II*" : สำนักพิมพ์ซีเอ็ดยูเคชั่น, 2545

4. Andrew S.Tanenbaum, "*Computer Networks*" , Prentice-Hall, Inc. 1996. third edition

5. Heather Osterloh, "*CCNA 2.0 Prep Kit 640-507 Routing and Switching*" ,Que Corporation . 2000

6. เอกสารจากเว็บไซต์ http://java.sun.com/j2se/1.4.2/docs/api

7. เอกสารจากเว็บไซต์ http://k12linux.mesd.k12.or.us/nag2/lnag.html

8. เอกสารจากเว็บไซต์ http://picolibre.enst-bretagne.fr/projects/flan/FlanDocFolder/FlanDoc.html