

โปรแกรมจำลองการตั้งค่าอุปกรณ์เราเตอร์

Router Configuring Simulator



T 0 4 6 1 2 4



นางสาวณัฏฐา ถิ่นทมทอง
นางสาวพนิดา ตั้งวิทย์โมไนย

26/1
26 2544
2544

เลขหมู่.....
เลขทะเบียน.....46124
วัน, เดือน, ปี 20 ส.ค. 2546

b.....
i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

26/1 26 2544

โปรแกรมจำลองการตั้งค่าอุปกรณ์เราเตอร์
Router Configuring Simulator



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมจำลองการตั้งค่าอุปกรณ์เราเตอร์

Router Configuring Simulator

ผู้จัดทำ

1. นางสาวณัฏฐา ลั่นทมทอง รหัสประจำตัว 41014126
2. นางสาวพนิดา ตั้งวิทย์โมไนย รหัสประจำตัว 41014292




อาจารย์ที่ปรึกษา
(อาจารย์ ธนา หงษ์สุวรรณ)

โปรแกรมจำลองการตั้งค่าอุปกรณ์เราเตอร์

นางสาวณัฏฐา ลั่นทมทอง 41014126
 นางสาวพนิดา ตั้งวิทย์โมไนย 41014292
 อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
 ปีการศึกษา 2544

บทคัดย่อ

ปัจจุบันระบบเครือข่ายคอมพิวเตอร์เข้ามามีบทบาทในชีวิตประจำวันมากขึ้น ความต้องการในการนำคอมพิวเตอร์ไปใช้ในการติดต่อสื่อสารหรือเชื่อมต่อเพื่อให้บริการแก่กันก็มากขึ้นตามไปด้วย การติดตั้งเราเตอร์ไม่เป็นเรื่องยากเกินไปสำหรับนักศึกษาระดับปริญญาตรี หรือชั้นสูงกว่านี้ แต่เนื่องจากเราเตอร์ในปัจจุบันนี้ยังมีราคาสูงมาก ทำให้โอกาสในการฝึกหัดใช้ให้เกิดความคุ้นเคยมีน้อยลง มีผลให้ผู้ที่มีความสามารถและความชำนาญในการติดตั้งค่าเราเตอร์มีน้อยไปด้วย ทั้งๆ ที่ความต้องการมีมาก

ปริญญานิพนธ์นี้จัดทำขึ้นเพื่อให้มีความสามารถที่จะจำลองการตั้งค่าอุปกรณ์เราเตอร์ของบริษัทซิสโก้ได้หลายเครื่องโดยไม่จำเป็นต้องมีเราเตอร์จริงๆ สามารถรองรับโปรโตคอลในการเลือกเส้นทางได้ 2 โปรโตคอลคือ อาร์ไอพีและโอเอสพีเอฟ มีการสร้างแผนภาพเน็ตเวิร์กเพื่อให้เห็นภาพรวมของระบบได้ โดยใช้หลักการของออบเจกต์โอเรียนเต็ดโปรแกรมมิ่ง (Object Oriented Programming) และพัฒนาโดยภาษาจาวา

Router Configuring Simulator

Miss Nudlada Lunthomthong

Miss Phanita Thungwitmonai

Thana Hongsuwan Advisor

Abstract

Nowadays, computer network has been closer to our life than before. In addition , the demand in using computer for communication and connection with one another have increased as well. As we know , the way to setup the router is not too complex for any student with a bachelor degree or higher. However, the price of these devices are still high and that is why there are not many people who are experts in these relating subjects even if there is a high demand for them.

This project is established to assume that we can setup Cisco Routers without the real routers, so we can also deal with 2 ways of protocol which are RIP and OSPF. There is a network view for a network system overview education by using Object Oriented Programming and Java Application.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะไม่สามารถเสร็จสมบูรณ์ได้ถ้าไม่ได้รับความช่วยเหลือและการร่วมมือของบุคคลหลายๆ ฝ่ายด้วยกัน โดยเฉพาะอย่างยิ่งบุคคลผู้ซึ่งเป็นผู้จุดประกายความคิดให้เกิดหัวข้อปริญญานิพนธ์นี้ขึ้นมา นั่นก็คืออาจารย์ธนา หงษ์สุวรรณ และอาจารย์อัครเดช วัชรภุพงษ์ อาจารย์ที่คอยให้คำปรึกษาปริญญานิพนธ์ตลอดเวลา และขอขอบคุณคณาจารย์ทุกท่านในภาควิชาวิศวกรรมศาสตร์ที่ได้ให้คำแนะนำ และความรู้ทางด้านคอมพิวเตอร์

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์โดยเฉพาะห้องวิจัยและพัฒนาการรักษาความปลอดภัยข้อมูล (ISAG) ที่ได้เอื้อเฟื้อสถานที่ ให้คณะผู้จัดทำได้ทำการวิจัย และช่วยอำนวยความสะดวกต่างๆ ขอขอบคุณพี่ต่อ, พี่ก่อ, พี่ด้อม, พี่เต๋ และเพื่อนๆ พี่ๆ น้องๆ ชาว ISAG ที่คอยให้ความช่วยเหลือในการทำงานตลอดเวลา เป็นที่ปรึกษายามมีปัญหา รวมทั้งให้ยืมทรัพยากรที่จำเป็นต่างๆ ขอขอบคุณชาว OLALA ที่คอยให้ความบันเทิงตลอดเวลา

ที่สำคัญและขาดมิได้ คือ ต้องขอขอบพระคุณบิดา มารดาที่ได้ให้กำเนิด คอยสั่งสอน และให้การสนับสนุนการศึกษา กิจกรรมต่างๆ นับเป็นพระคุณที่หาได้เปรียบมิได้ ทางคณะผู้จัดทำขอกราบขอบพระคุณมา ณ ที่นี้ด้วย

ณัฏฐา ลั่นทมทอง
พนิดา ตั้งวิทย์โมไนย

สารบัญ

หน้าที่

บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญภาพประกอบ	VII
สารบัญตาราง	IX
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์ของปริญญานิพนธ์	1
1.3 ขอบเขตของปริญญานิพนธ์	1
1.4 ขั้นตอนการดำเนินงาน	1
บทที่ 2 ทฤษฎีไอพีและอินเทอร์เน็ต	2
2.1 การเชื่อมโยงเครือข่าย	2
2.2 ความหมายของโพรโตคอล	2
2.3 สถาปัตยกรรมของอินเทอร์เน็ตและความหมายของเราเตอร์	2
2.4 สถาปัตยกรรมทฤษฎีไอพี	3
บทที่ 3 เทคโนโลยีเครือข่ายคอมพิวเตอร์	8
3.1 แลตามข้อกำหนดของ IEEE	8
3.1.1 IEEE 802.3 (Ethernet)	9
3.1.2 IEEE 802.4 (Token Bus)	10
3.1.3 IEEE 802.5 (Token Ring)	10
3.1.2 เอฟดีดีไอ	11
3.2 อุปกรณ์เครือข่าย	12
3.2.1 ฮับ	12
3.2.2 บริดจ์	13
3.2.3 เราเตอร์	15
บทที่ 4 ไอพีแอดเดรส	16
4.1 ไอพีแอดเดรส	16
4.1.1 ความสำคัญของเลขเครือข่ายและเลขโฮสต์	17
4.1.2 การจัดคลาสเครือข่าย	17
4.1.3 ลักษณะสำคัญของแต่ละคลาส	18

4.2 การแบ่งเครือข่ายย่อย	20
4.2.1 ซับเน็ตมาสก์	21
4.2.2 ดีฟอลต์ซับเน็ตมาสก์	22
4.2.3 การเลือกเส้นทางในซับเน็ต	23
บทที่ 5 โพรโทคอลเลือกเส้นทาง (Routing Protocol)	23
5.1 การเลือกเส้นทาง (Routing)	23
5.2 ตารางเส้นทาง (Routing Table)	23
5.3 ประเภทของการเลือกเส้นทาง	23
5.4 อาร์ไอพี	27
5.4.1 การทำงานของอาร์ไอพี	28
5.4.2 การปรับค่าเมื่อเครือข่ายเปลี่ยน	31
บทที่ 6 โอเอสพีเอฟ (OSPF : Open shorted Path First)	40
6.1 การทำงานของโอเอสพีเอฟโดยสรุป	40
6.2 เรียนรู้เครือข่าย	40
6.3 สร้างฐานข้อมูลเส้นทาง	41
6.4 การคำนวณเส้นทาง	42
6.4.1 ขั้นตอนวิธีของไดจ์สตรา (Dijkstra Algorithm)	42
6.4.2 การปรับเปลี่ยนเมื่อเครือข่ายเปลี่ยน	44
6.5 การจัดองค์ประกอบย่อย	45
6.6 การแบ่งระบบอัตโนมัติ	45
บทที่ 7 การออกแบบและการสร้างโปรแกรม	46
7.1 โครงสร้างของโปรแกรม (Design)	46
7.2 การจัดการกับ User Interface	47
7.3 การจัดการกับ Commmandline	50
7.4 การจัดการกับ Add Router	52
7.5 การจัดการ Interface	53
7.6 การจัดการ RIP	55
7.7 การจัดการ OSPF	53
บทที่ 8 ตัวอย่างและการทดสอบการทำงานของโปรแกรม	59
8.1 ทดสอบการทำงานของทางเลือกเส้นทางแบบสแตติก	59
8.2 ทดสอบการทำงานของทางเลือกเส้นทางแบบโพรโทคอลอาร์ไอพี	61
8.3 ทดสอบการทำงานของทางเลือกเส้นทางแบบโพรโทคอลโอเอสพีเอฟ	62

บทที่ 9 สรุปและวิจารณ์	หน้าที่
9.1 ปัญหาและอุปสรรค	63
9.2 แนวทางการประยุกต์และพัฒนา	63
9.3 สรุปโครงการ	63
ภาคผนวก A สรุปคำสั่งทั้งหมด	65
บรรณานุกรม	82



สารบัญภาพประกอบ

หน้าที่

รูปที่ 2-1 แสดงการเปรียบเทียบเลเซอร์ของโอเอสไอกับเลเซอร์ของทีซีพี/ไอพี	5
รูปที่ 2-2 แสดงข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี	8
รูปที่ 3-1 เป็นแบบจำลองบางส่วนของมาตรฐานซึ่งเป็นที่รู้จักแพร่หลาย	10
รูปที่ 3-2 อีเทอร์เน็ตประเภทต่างๆ	11
รูปที่ 3-3 การทำงานโทเค็นริง	12
รูปที่ 3-4 โทโปโลยีเอฟดีอีไอ	13
รูปที่ 3-5 การใช้ฮับในโทโปโลยี	13
รูปที่ 3-6 แบบจำลองการทำงานของฮับ	14
รูปที่ 3-7 แบบจำลองการทำงานของบริดจ์	15
รูปที่ 3-8 แบบจำลองการทำงานของเราเตอร์	16
รูปที่ 4-1 รูปแบบของไอพีแอดเดรส	17
รูปที่ 4-2 เราเตอร์เชื่อมโยงเครือข่ายที่มีเลขเครือข่ายต่างกัน	18
รูปที่ 4-3 การแบ่งคลาสเครือข่าย	18
รูปที่ 4-4 การแบ่งคลาส D และ E	20
รูปที่ 4-5 ตัวอย่างการแบ่งเครือข่ายย่อยของ 158.105	22
รูปที่ 4-6 การตรวจหาแอดเดรสซ้ำเน็ตเพื่อเลือกเส้นทาง	25
รูปที่ 5-1 เครือข่ายการแสดงผลการเลือกเส้นทางแบบสแตติก	28
รูปที่ 5-2 เครือข่ายสถิติการทำงานของอาร์ไอพี	30
รูปที่ 5-3 การประกาศค่าและปรับค่าตารางของอาร์ไอพี	31
รูปที่ 5-4 การนับเข้าสู่อนันต์	33
รูปที่ 5-5 การประกาศเส้นทางด้วยวิธีสปลิตฮอไรซัน	34
รูปที่ 5-6 การประกาศเส้นทางด้วยวิธีสปลิตฮอไรซันแบบพอยชันรีเวอร์ส	35
รูปที่ 5-7 เครือข่ายที่จำเป็นต้องใช้โฮลดาวน์ร่วมกับทริกเกอร์อัปเดต	36
รูปที่ 5-8 พอร์เมตของเฟรมอาร์ไอพี	36
รูปที่ 5-9 พอร์เมตของเฟรมอาร์ไอพีเวอร์ชัน 2	39
รูปที่ 5-10 เฟรมอาร์ไอพีเวอร์ชัน 2 เมื่อใช้การพิสูจน์ตัวจริง	39
รูปที่ 6-1 แผนที่ทางหลวง	41
รูปที่ 6-2 เราเตอร์ A ส่งแพ็กเก็ตตกทาย	42
รูปที่ 6-3 การคำนวณหาเส้นทางสั้นที่สุดตามขั้นตอนวิธีของไดจ์สตรา	43
รูปที่ 6-4 ปัญหาเส้นทางขัดข้อง	44

รูปที่ 6-5 เส้นทางใหม่ที่คำนวณได้	44
รูปที่ 6-6 เราเตอร์ขอบปลายของระบบอัตโนมัติ	45
รูปที่ 7-1 การออกแบบโปรแกรม	47
รูปที่ 7-2 หน้าจอหลักของโปรแกรม	47
รูปที่ 7-3 หน้าจอสำหรับเลือกกรุ่นของเราเตอร์	49
รูปที่ 7-4 ภาพแสดงภาพรวมของเครือข่าย	49
รูปที่ 7-5 แสดงขั้นตอนการจัดการกับส่วน Command Line	50
รูปที่ 7-6 แสดงขั้นตอนการทำงานของจัดการเพิ่มเราเตอร์	53
รูปที่ 7-7 แสดงขั้นตอนการจัดการอินเทอร์เน็ตของเราเตอร์	54
รูปที่ 7-8 ขั้นตอนการทำงานของโปรโตคอล RIP	56
รูปที่ 7-9 ขั้นตอนการทำงานของโปรโตคอลโอเอสพีเอฟ	57
รูปที่ 8-1 ภาพจำลองเน็ตเวิร์ก	59
รูปที่ 8-2 เมื่อเราเตอร์ B ใช้คำสั่ง ip route	59
รูปที่ 8-3 เมื่อเราเตอร์ B ใช้คำสั่ง show ip route	60
รูปที่ 8-4 เมื่อเราเตอร์ B ใช้คำสั่ง ping	61
รูปที่ 8-5 เมื่อเราเตอร์ A ใช้คำสั่ง show ip route	61
รูปที่ 8-6 ภาพจำลองเน็ตเวิร์ก	62
รูปที่ 8-7 แสดงผลการใช้คำสั่ง Traceroute ของโปรโตคอลโอเอสพีเอฟ	62

สารบัญตาราง

หน้าที่

ตารางที่ 4-1 การจัดแบ่งเครือข่าย 158.108 ด้วยซับเน็ต 8 บิต	20
ตารางที่ 4-2 ค่าดีฟอลต์ซับเน็ตมาส์ก	22
ตารางที่ 6-1 ตารางเส้นทางที่เราเตอร์ A	41
ตารางที่ 6-2 ฐานข้อมูลถึงสเทตประจำแต่ละเราเตอร์	42



บทที่ 1

บทนำ

1.1 ความสำคัญและที่มา

ปัจจุบันระบบเครือข่ายคอมพิวเตอร์เข้ามามีบทบาทในชีวิตประจำวันมากขึ้น ความต้องการในการนำคอมพิวเตอร์ไปใช้ในการติดต่อสื่อสารหรือเชื่อมต่อเพื่อให้บริการแก่กันก็ย่อมมากขึ้นตามไปด้วย ในขณะที่การเชื่อมต่อระบบเน็ตเวิร์กเป็นไปอย่างไม่ง่ายนัก เนื่องมาจากการขาดความรู้และประสบการณ์ถึงแม้ว่าปัจจุบันได้มีการเขียนหนังสือการเชื่อมต่อระบบเน็ตเวิร์กเป็นภาษาไทยมากขึ้น แต่การเชื่อมต่อระบบใดๆ นั้นมีความละเอียดอ่อนมากนัก โดยเฉพาะการกำหนดค่าต่างๆ ให้กับเราเตอร์นั้นเป็นสิ่งที่ไม่ยากเพียงแต่คนโดยทั่วไปไม่ค่อยได้สัมผัสกับเราเตอร์จริงๆ ทำให้การกำหนดค่าเป็นไปด้วยความยากลำบาก และบางครั้งไม่สามารถใช้เราเตอร์ได้ตามความสามารถจริงๆ ของมัน ซึ่งเป็นเรื่องน่าเสียดายอย่างยิ่ง

ดังนั้นการจัดทำโปรแกรมจำลองการติดตั้งเราเตอร์ จึงเป็นแนวคิดที่ต้องการเปิดโอกาสทางการศึกษาให้กับบุคคลทั่วไปที่มีความสนใจทางด้านการติดตั้งเราเตอร์แต่ยังไม่มีเวลาหาข้อมูลที่จะติดตั้งกับเราเตอร์จริงๆ หรือไม่มีเราเตอร์จริงๆ ให้ลองใช้ ได้มีโอกาสได้ทดลองตั้งค่าอุปกรณ์เราเตอร์ของซิสโก้ เพื่อเป็นแนวทางในการตั้งค่าเราเตอร์ต่อไป

1.2 วัตถุประสงค์ของปริญญานิพนธ์

ปริญญานิพนธ์ที่จัดทำขึ้นนี้ จัดทำภายใต้วัตถุประสงค์หลัก 2 ประการ ได้แก่

- (1) เพื่อศึกษารายละเอียดและการทำงานของโปรโตคอลในการเลือกเส้นทางแบบต่างๆ ได้
- (2) เพื่อศึกษาการตั้งค่าเราเตอร์ และกำหนดค่าต่างๆ ให้กับเราเตอร์

1.3 ขอบเขตของปริญญานิพนธ์

- (1) สรุปการทำงานและการติดตั้งค่าให้กับเราเตอร์ รวมทั้งโปรโตคอลที่ใช้ในการเลือกเส้นทาง
- (2) โปรแกรมที่สามารถจำลองการติดตั้งเราเตอร์ได้

1.4 ขั้นตอนการดำเนินงาน

- (1) ศึกษาและรวบรวมคำสั่งที่ใช้ในการตั้งค่าอุปกรณ์เราเตอร์
- (2) ศึกษาโปรโตคอลเลือกเส้นทางแบบอาร์ไอพีและโอเอสพีเอฟ
- (3) ออกแบบคลาสและกำหนดรายละเอียด (Specification) ของโปรแกรมทั้งหมด
- (4) เขียนโปรแกรมติดต่อกับผู้ใช้ให้สามารถจัดการกับคำสั่งพื้นฐานในการตั้งค่าเราเตอร์ทั้งหมด
- (5) เขียนวิธีค้นหาเส้นทางโดยใช้โปรโตคอลอาร์ไอพี
- (6) เขียนวิธีค้นหาเส้นทางโดยใช้โปรโตคอลโอเอสพีเอฟ
- (7) ทำภาพจำลองเครือข่ายและทำบทเรียนสอนการใช้งาน
- (8) ทดสอบโปรแกรมและตรวจสอบข้อผิดพลาด

บทที่ 2

ทีซีพี/ไอพีและอินเทอร์เน็ต

ทีซีพี/ไอพีเป็นโพรโทคอลที่ได้รับการออกแบบให้เป็นอิสระจากชนิดคอมพิวเตอร์และระบบปฏิบัติการ ตัวโพรโทคอลมีความเชื่อถือได้สูงและสามารถปรับเปลี่ยนการทำงานตามสภาพเครือข่ายได้ในกรณีที่บางเส้นทางชำรุด

2.1 การเชื่อมโยงเครือข่าย

จุดประสงค์ของการเชื่อมโยงคอมพิวเตอร์เข้าเป็นเครือข่ายคือต้องการให้คอมพิวเตอร์สามารถสื่อสารและแลกเปลี่ยนข้อมูลระหว่างกันได้ เครือข่ายคอมพิวเตอร์เริ่มจากเครือข่ายขนาดเล็กภายในองค์กรที่เชื่อมโยงกันภายใต้สภาพพื้นที่จำกัดซึ่งเรียกว่า เครือข่ายเฉพาะที่ (LAN : Local Area Network) เมื่อเชื่อมโยงเครือข่ายเข้าด้วยกันและขยายขอบเขตครอบคลุมพื้นที่ระหว่างเมือง หรือระหว่างประเทศ ก็จะเรียกเครือข่านั้นว่า เครือข่ายพื้นที่กว้าง (WAN : Wide Area Network)

เครือข่ายยุคเริ่มต้นมีสถานที่ใช้ฮาร์ดแวร์ประเภทเดียวกันและสามารถทำงานร่วมกันได้อย่างกลมกลืน ต่อมาเมื่อเทคโนโลยีเครือข่ายเพิ่มขึ้นเช่น อินเทอร์เน็ตและโทเค็นริง ปัญหาที่เกิดขึ้นคือจะเชื่อมต่อเครือข่ายต่างเทคโนโลยีเข้าด้วยกันได้อย่างไร โดยไม่จำกัดว่าคอมพิวเตอร์จะอยู่ในเครือข่ายเดียวกัน หรือต่างเครือข่ายกัน

2.2 ความหมายของโพรโทคอล

การเชื่อมโยงเครือข่ายต่างฮาร์ดแวร์จำเป็นต้องกำหนดข้อตกลงร่วม หรือ โพรโทคอล (Protocol) เพื่อให้คอมพิวเตอร์สื่อสารกันตามข้อกำหนด ทีซีพี/ไอพีจัดเป็นโพรโทคอลหนึ่งที่ถูกออกแบบมาเพื่อแก้ไขปัญหาที่เกิดขึ้น

2.3 สถาปัตยกรรมของอินเทอร์เน็ตและความหมายของเราเตอร์

ทีซีพี/ไอพีเป็นแกนสำคัญในการถ่ายโอนข้อมูลระหว่างเครื่องคอมพิวเตอร์ที่อาจอยู่ภายในเครือข่ายเดียวกันหรือภายนอกเครือข่าย โครงสร้างของอินเทอร์เน็ตประกอบด้วยเครือข่ายย่อยจำนวนมากต่อเชื่อมกันผ่าน เราเตอร์ (Router)

เราเตอร์เป็นอุปกรณ์เครือข่ายซึ่งมีหน้าที่เลือกเส้นทางเพื่อนำส่งข้อมูลในรูปแพ็กเก็ต หากเปรียบเทียบกับการส่งจดหมายทางไปรษณีย์แล้ว เราเตอร์ทำหน้าที่เสมือนที่ทำการไปรษณีย์ พนักงานไปรษณีย์จะพิจารณาจุดหมายปลายทางของจดหมายและเลือกเส้นทางส่งจดหมายไปยังที่ทำการไปรษณีย์ถัดไปจนกว่าจะถึงมือผู้รับ

เครือข่ายที่ต่อเชื่อมกันด้วยเราเตอร์เช่นรูปที่ 2-2 เป็นลักษณะทั่วไปของอินเทอร์เน็ตแต่ละเครือข่ายนิยมเขียนแทนด้วยรูป กลุ่มเมฆเครือข่าย (Network Cloud) เพื่อแสดงเครือข่ายโดยไม่กล่าวถึงโทโปโลยีและการจัดการภายใน

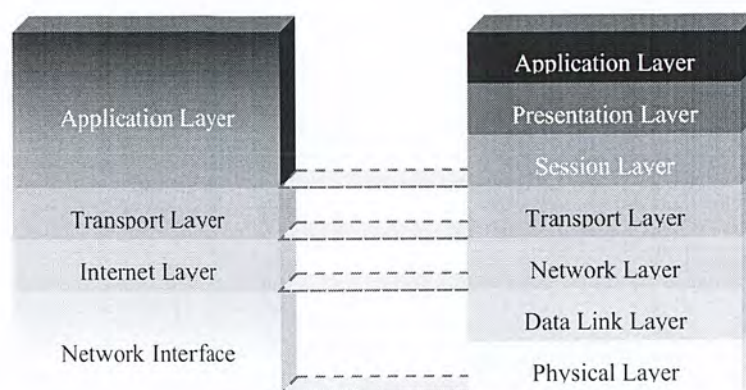
ในยุคต้นของอินเทอร์เน็ตใช้คำว่า เกตเวย์ (Gateway) เพื่อสื่อความหมายถึงอุปกรณ์เชื่อมต่อเครือข่าย แต่ในปัจจุบันนี้เราเตอร์และเกตเวย์เป็นอุปกรณ์ซึ่งมีหน้าที่แตกต่างกัน คำว่าเกตเวย์ที่ใช้เดิมนั้นในปัจจุบันนิยมใช้คำว่าเราเตอร์แทน หน้าที่หลักของเราเตอร์เชื่อมต่อเครือข่ายที่ใช้โปรโตคอลเดียวกันและเลือกเส้นทางส่งข้อมูล ขณะที่เกตเวย์หมายถึงฮาร์ดแวร์หรือซอฟต์แวร์ซึ่งเป็นตัวแปลงระหว่างระบบสองระบบที่มีโปรโตคอล โครงสร้างการจัดข้อมูล หรือสถาปัตยกรรมที่แตกต่างกัน เกตเวย์จะแปลงแพ็กเก็ตจากระบบต้นทางให้อยู่ในรูปแบบของระบบปลายทางเช่นเกตเวย์ระหว่างทีซีพี/ไอพีและโปรโตคอล เอสเอ็นเอ (SNA : Systems Network Architecture) หรืออิเล็กทรอนิกส์เมล์เกตเวย์ที่แปลงรูปแบบจดหมายของโปรโตคอลใดๆ ไปสู่ระบบ x.400 ของไอเอสโอ

2.4 สถาปัตยกรรมทีซีพี/ไอพี

ทีซีพี/ไอพีเป็นโปรโตคอลมาตรฐานที่ใช้กันอยู่ในระบบปฏิบัติการแบบยูนิกซ์ เริ่มพัฒนาโดยกระทรวงกลาโหมของสหรัฐใน ค.ศ. 1969 เพื่อเชื่อมต่อเครื่องคอมพิวเตอร์หลายชนิดที่อยู่ห่างไกลกัน เครือข่ายที่จัดตั้งในระยะแรกชื่อว่า อาร์พาเน็ต (ARPANET)

ต่อมาได้พัฒนาเป็นเครือข่ายอินเทอร์เน็ต โปรโตคอลนี้เหมาะสำหรับเชื่อมต่อคอมพิวเตอร์ทั้งใกล้และไกลเข้าด้วยกัน และมีมาตรฐานรองรับทำให้ผู้ผลิตฮาร์ดแวร์และซอฟต์แวร์ สามารถสร้างอุปกรณ์และโปรแกรมที่จะรองรับการทำงานของโปรโตคอลนี้ ทำให้เครื่องคอมพิวเตอร์สามารถรับส่งข้อมูลกันได้ไม่ว่าจะเป็นเครื่องขนาดเล็กหรือขนาดใหญ่ หรือใช้ระบบปฏิบัติการอะไรก็ตาม ทีซีพี/ไอพี (TCP/IP) เป็นชุดโปรโตคอลที่ประกอบด้วยโปรโตคอลต่างๆ หลายโปรโตคอล แต่ละโปรโตคอลมีคุณลักษณะ และมีความสามารถในการทำงานแตกต่างกัน โดยที่ในบทนี้ได้กล่าวถึงรายละเอียดและคุณสมบัติของโปรโตคอลที่สำคัญบางโปรโตคอล

ทีซีพี/ไอพี (TCP/IP หรือ Transmission Control Protocol/Internet Protocol) เป็นโปรโตคอลในการสื่อสารในระบบอินเทอร์เน็ตและอินทราเน็ต การทำงานของทีซีพี/ไอพีสามารถเปรียบเทียบกับโมเดลอ้างอิงไอเอสโอ (Open System Interconnection Reference Model : OSI) ตามมาตรฐานไอเอสโอ (International Organization for Standardization: ISO) ได้ดังรูปที่ 2-1



รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของโอเอสไอกับเลเยอร์ของทีซีพี/ไอพี

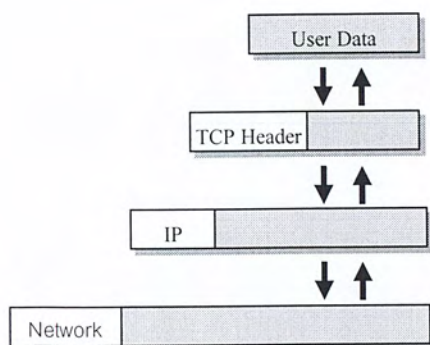
ในแต่ละระดับชั้นของทีซีพี/ไอพีมีการทำงานที่แตกต่างกัน ตั้งแต่การติดต่อกับแอปพลิเคชัน จนกระทั่งแปลงเป็นสัญญาณส่งไปตามสายสัญญาณ ซึ่งการทำงานในแต่ละระดับชั้นของทีซีพี/ไอพี มีดังนี้

ชั้นแอปพลิเคชัน
(Application Layer) ชั้นนี้รองรับการทำงานของแอปพลิเคชันต่างๆ ที่ทำงานเป็นโพรเซสอยู่ในเครื่องต้นทางและปลายทาง โดยจัดการเชื่อมต่อระหว่างโพรเซส หรือแอปพลิเคชันที่อยู่ต่างเครื่องกัน โดยการทำงานของแอปพลิเคชันต่างๆ มีการติดต่อกันตามแต่ละโพรโตคอลเฉพาะแล้วแต่แอปพลิเคชันที่ใช้งาน ซึ่งจะขอบริการจากชั้นทรานสปอร์ตอีกทีหนึ่ง

ชั้นทรานสปอร์ต
(Transport Layer) มีการสร้างการเชื่อมต่อกันระหว่างแอปพลิเคชันแบบ End-to-End โดยจุดที่เชื่อมต่อกันเพื่อรับส่งข้อมูลนี้เรียกว่า พอร์ต (Port) หรือซ็อกเก็ต (Socket) ในชั้นนี้มีบริการหลักอยู่ 2 แบบ คือ Connection Oriented โดยเรียกผ่านโพรโตคอลทีซีพี (TCP: Transmission Control Protocol) และ Connectionless ซึ่งเรียกผ่านโพรโตคอลยูดีพี (UDP: User Datagram Protocol) ซึ่งกล่าวถึงในหัวข้อถัดไป

ชั้นอินเทอร์เน็ต
(Internet Layer) ชั้นนี้มีหน้าที่ส่งผ่านข้อมูลระหว่างเครือข่าย โดยมีโพรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใดๆ ในอินเทอร์เน็ต คือ ไอพี (Internet Protocol: IP) ซึ่งกล่าวถึงในหัวข้อถัดไป นอกจากนี้ในชั้นนี้ยังมีโพรโตคอลทำงานอยู่ด้วยอีก 2 ชนิด คือ ไอซีเอ็มพี (Internet Control Message Protocol: ICMP) และเออาร์พี (Address Resolution Protocol: ARP)

ชั้นเน็ตเวิร์กอินเตอร์เฟซ
(Network Interface Layer) ทำหน้าที่ในการแปลงข้อมูลให้อยู่ในรูปแบบที่เหมาะสมกับเครือข่ายแต่ละแบบ ซึ่งแตกต่างกันออกไป และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่าย



รูปที่ 2-2 แสดงการข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี

ในชุดโพรโทคอลทีซีพี/ไอพีนี้ มีโพรโทคอลหลักที่บอกกล่าวถึง 3 โพรโทคอล ได้แก่

- โพรโทคอลทีซีพี
- โพรโทคอลยูดีพี ซึ่งทำงานในชั้นทรานสปอร์ต
- โพรโทคอลไอพี ซึ่งทำงานในชั้นอินเทอร์เน็ต
- โพรโทคอลไอซีเอ็มพี
- โพรโทคอลเอสเอ็มทีพี
- โพรโทคอลเอฟทีพี
- โพรโทคอลทีเอฟทีพี
- โพรโทคอลเทลเน็ต
- โพรโทคอลดีเอ็นเอส
- โพรโทคอลเอสเอ็มทีพี
- โพรโทคอลบูตพี
- โพรโทคอลดีเอชซีพี

โดยมีรายละเอียดดังต่อไปนี้

โพรโทคอลทีซีพี (TCP : Transmission Control Protocol)

ทีซีพีทำหน้าที่นำส่งข้อมูลโดยรับประกันความเชื่อถือ ทีซีพีด้านส่งต้องส่งแพ็กเก็ตซ้ำใหม่หากแพ็กเก็ตสูญหาย ทีซีพีด้านรับมีหน้าที่จัดแพ็กเก็ตให้ถูกต้องตามลำดับและกำจัดแพ็กเก็ตซ้ำซ้อน ทีซีพีเป็นโพรโทคอลแบบ “Connection Oriented” ก็ต้องสถาปนาการเชื่อมต่อระหว่างสถานีต้นทางและปลายทางก่อนการส่งข้อมูล

ทีซีพีต้นทางจัดแบบข้อมูลเพื่อให้ไอพีดำเนินการ ทีซีพีปลายทางเมื่อรับแพ็กเก็ตจากไอพีก็จะส่งต่อให้โพรโทคอลประยุกต์ โพรโทคอลประยุกต์ที่ใช้บริการผ่านทีซีพีได้แก่ telnet SMTP FTP เป็นต้น

โพรโทคอลยูดีพี (UDP : User Datagram Protocol)

ยูดีพีเป็นโพรโทคอลแบบ “Connectionless” คือไม่ต้องสถาปนาการเชื่อมต่อระหว่างสถานีรับและสถานีส่ง ยูดีพีเป็นโพรโทคอลระดับชั้นเดียวกับทีซีพีแต่ไม่มีกลไกรับประกันความเชื่อถือในการขนถ่ายข้อมูล การข้อมูลสูญหาย ช้าช้อน หรือลำดับไม่ถูกต้อง ยูดีพีจะปล่อยให้โพรโทคอลที่เรียกใช้งานดำเนินการกับปัญหาเหล่านี้เอง

ลักษณะเด่นของยูดีพีคือการใช้การประมวลผลต่ำกว่าทีซีพี เนื่องจากเฮดเดอร์มีขนาดเล็กและไม่ต้องสถาปนาการเชื่อมต่อ แอปพลิเคชันที่ทำงานโดยรับส่งคำถามและคำตอบเป็นรายการ (Transaction) เช่น ดีเอ็นเอส หรือทีเอฟทีพีจะเหมาะกับการใช้บริการผ่านยูดีพี

โพรโทคอลไอพี (IP : Internet Protocol)

ไอพีเป็นโพรโทคอลแกนของทีซีพี/ไอพี ไอพีทำหน้าที่กำหนดรูปแบบของแอดเดรสประจำเครื่องเพื่อใช้ในการลำเลียงข้อมูลจากเครื่องต้นทางไปยังเครื่องปลายทาง นอกจากนี้ยังทำหน้าที่เลือกเส้นทางส่งข้อมูล ตลอดจนแบ่งขนาดข้อมูลให้เหมาะกับฮาร์ดแวร์ระดับล่าง

โพรโทคอลไอซีเอ็มที (ICMP : Internet Control Message Protocol)

ไอซีเอ็มทีเป็นโพรโทคอลซึ่งใช้รายงานสถานะความคิดพลาดที่เกิดขึ้น ตัวอย่างเช่น ในกรณีที่เราเตอร์ไม่สามารถนำข้อมูลส่งไปถึงปลายทางได้ เราเตอร์จะใช้ไอซีเอ็มทีแจ้งสาเหตุ

โพรโทคอลเอสเอ็มทีพี (SMTP : Simple Mail Transfer Protocol)

บริการพื้นฐานที่มีในทุกเครือข่ายได้แก่บริการไปรษณีย์อิเล็กทรอนิกส์ เอสเอ็มทีพีเป็นโพรโทคอลทำหน้าที่รับส่งจดหมายอิเล็กทรอนิกส์ระหว่างโฮสต์

โพรโทคอลเอฟทีพี (FTP : File Transfer Protocol)

เอฟทีพีให้บริการถ่ายโอนแฟ้มข้อมูลระหว่างเครื่อง เอฟทีพีช่วยให้ผู้ใช้เข้าถึงโฮสต์และจำกัดขอบเขตการทำงานเฉพาะที่เกี่ยวข้องกับแฟ้มข้อมูลเช่น สำเนาข้อมูล ลบแฟ้ม หรือสร้างไดเรกทอรี เป็นต้น

โพรโทคอลทีเอฟทีพี (TFTP : Trivial File Transfer Protocol)

ทีเอฟทีพีทำหน้าที่เป็นโพรโทคอลถ่ายโอนแฟ้มเช่นเดียวกันกับเอฟทีพี แต่ทีเอฟทีพีใช้บริการผ่านยูดีพี ประโยชน์อย่างหนึ่งของการใช้ทีเอฟทีพีได้แก่การใช้ใน สถานีไร้ดิสก์ (Diskless Workstation) ซึ่งไม่มีดิสก์ประจำตัว สถานีไร้ดิสก์จะบูตระบบด้วยโพรโทคอลบูตพีและใช้ทีเอฟทีพีเพื่อถ่ายโอนระบบปฏิบัติการจากทีเอฟทีพีเซิร์ฟเวอร์

โพรโทคอลเทลเน็ต (TELNET : File Telecommunication Network)

เทลเน็ตเป็นโพรโทคอลสำหรับขอเข้าใช้โฮสต์ระยะไกล หรือเรียกว่า รีโมตล็อกอิน (Remote Login) เทลเน็ตให้บริการเข้าใช้คอมพิวเตอร์ในเครือข่ายโดยเสมือนกับว่ากำลังทำงานอยู่ที่เทอร์มินอลของคอมพิวเตอร์เครื่องนั้น เทลเน็ตเซิร์ฟเวอร์ที่คอมพิวเตอร์ปลายทางจะรอรับการขอบริการจากเทลเน็ตไคลเอนต์ ผู้ขอใช้เทลเน็ตจะต้องมีบัญชีประจำเครื่องที่ให้บริการเทลเน็ต

โพรโทคอลดีเอ็นเอส (DNS : Domain Name System)

ดีเอ็นเอสเป็นโพรโทคอลที่ให้บริการสอบถามไอพีแอดเดรสหรือชื่อโดเมน ก่อนการติดต่อไปยังโฮสต์ใดๆ ชื่อโฮสต์จะถูกส่งไปสอบถามผ่านเซิร์ฟเวอร์ที่ให้บริการดีเอ็นเอสเพื่อขอไอพีแอดเดรสกลับมา นอกจากนี้ยังให้บริการเกี่ยวข้องกับฐานข้อมูลประจำเครื่อง

โพรโทคอลเอสเอ็นเอ็มพี (SNMP : Simple Network Management Protocol)

เอสเอ็นเอ็มพีทำหน้าที่เป็นโพรโทคอลช่วยบริหารระบบ เช่น เก็บรวบรวมข้อมูลการทำงานของอุปกรณ์และคอมพิวเตอร์ภายในเครือข่าย ตรวจสอบปริมาณข้อมูลที่ไหลเวียนหรือช่วยวิเคราะห์หาข้อผิดพลาดในระบบ เป็นต้น

โพรโทคอลบูตพี (BOOTP : Bootstrap Protocol)

ให้บริการบูตสำหรับสถานีไม่มีดิสก์ สถานีที่เป็นบูตพีไคลเอนต์จะติดต่อกับบูตพีเซิร์ฟเวอร์เพื่อขอถ่ายโอนระบบปฏิบัติการ

โพรโทคอลดีเอชซีพี (DHCP : Dynamic Host Configuration Protocol)

ดีเอชซีพีบริการติดตั้งค่าแบบไดนามิกให้โฮสต์ในเครือข่าย การทำงานของดีเอชซีพีเป็นแบบไคลเอนต์-เซิร์ฟเวอร์ ดีเอชซีพีเซิร์ฟเวอร์จะให้ค่าแบบไม่ตายตัวกับไคลเอนต์ที่ร้องขอบริการ เช่นการให้ไอพีแอดเดรส หรือค่าประจำไคลเอนต์

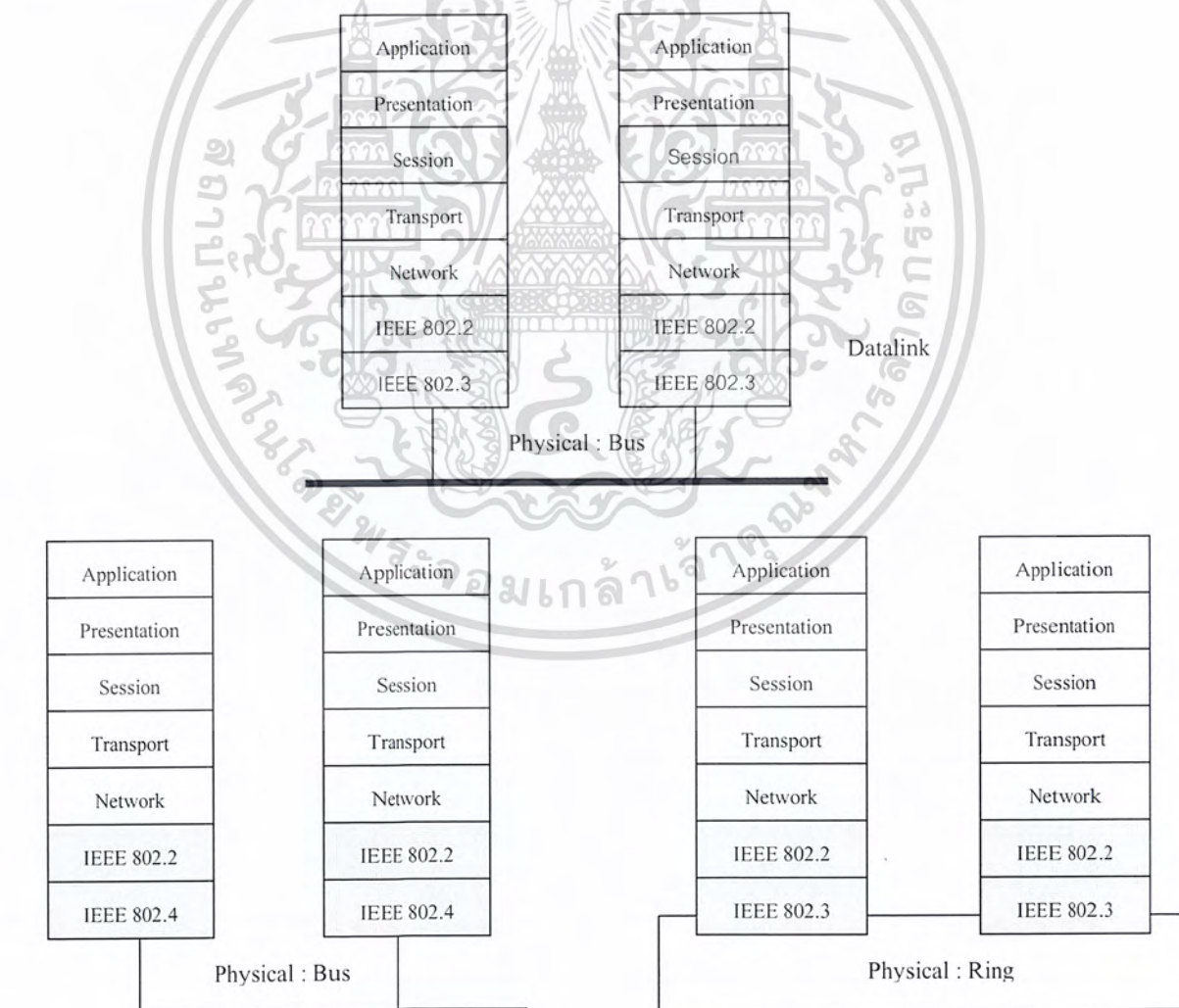
บทที่ 3

เทคโนโลยีเครือข่ายคอมพิวเตอร์

ทีซีพี/ไอพีผ่านการออกแบบมาให้สามารถทำงานกับระบบการสื่อสารระดับล่างโดยไม่จำกัดประเภท ในปัจจุบันจึงมีฮาร์ดแวร์เครือข่ายจำนวนมากทั้งในกลุ่มของแลนและแวนที่รองรับการทำงานร่วมกับทีซีพี/ไอพี

3.1 แลนตามข้อกำหนดของ IEEE

เครือข่ายเฉพาะที่หรือแลนมีลักษณะโดยทั่วไปคือมีอัตราการส่งข้อมูลสูงและมักมีรัศมีเครือข่ายครอบคลุมระยะทางประมาณ 5 กิโลเมตร ในหัวข้อนี้จะกล่าวถึงแลนตามมาตรฐานที่กำหนดโดย IEEE (Institute of Electronic and Electronics Engineers)



รูปที่ 3-1 เป็นแบบจำลองบางส่วนของมาตรฐานซึ่งเป็นที่รู้จักแพร่หลาย

IEEE กำหนดเครือข่ายเฉพาะที่ใช้ตัวเลข 802 ตามด้วยตัวเลขย่อยเป็นรหัสประจำแต่ละมาตรฐาน

- IEEE 802.3 หรืออีเทอร์เน็ต ใช้โปรโตคอลซีเอสเอ็ม/เอชดีในโทโพโลยีแบบบัส
- IEEE 802.4 หรือโทเค็นบัส ใช้โปรโตคอลส่งผ่านโทเค็นในโทโพโลยีแบบบัส
- IEEE 802.5 หรือโทเค็นริง ใช้โปรโตคอลส่งผ่านโทเค็นในโทโพโลยีแบบวงแหวน

ข้อกำหนดเพิ่มเติมที่ IEEE สร้างขึ้นในทุกมาตรฐาน 802 คือการแยกระดับชั้นดาต้าลิงค์ออกเป็น 2 ส่วนย่อย โดยให้มาตรฐาน 802.2 ซึ่งเรียกว่า แอลแอลซี (LLC : Logical Link Control) เป็นส่วนเชื่อมต่อกับชั้นเน็ตเวิร์ก อินเทอร์เน็ตของแอลแอลซีในเครือข่ายแต่ละชนิด (802.3, 802.4 และ 802.5) จะมีรูปแบบเชื่อมต่อกับชั้นเน็ตเวิร์กเช่นเดียวกันหมด

3.1.1 IEEE 802.3 (Ethernet)

อีเทอร์เน็ต เป็นเครือข่ายที่มีความเร็วการส่งข้อมูล 10 เมกะบิตต่อวินาที สถานีในเครือข่ายอาจมีโทโพโลยีแบบบัสหรือแบบดาว IEEE ได้กำหนดมาตรฐานอีเทอร์เน็ตไว้หลายชนิดสายสัญญาณ เช่น

- 10Base5 อีเทอร์เน็ตโทโพโลยีแบบบัสซึ่งใช้ในสายโคแอกเชียลแบบหนา (Thick Ethernet) ความยาวของสายในเซกเมนต์หนึ่งๆ ไม่เกิน 500 เมตร
- 10Base2 อีเทอร์เน็ตโทโพโลยีแบบบัสซึ่งใช้สายโคแอกเชียลแบบบาง (Thin Ethernet) ความยาวของสายในเซกเมนต์หนึ่งๆ ไม่เกิน 185 เมตร
- 10BaseT อีเทอร์เน็ตโทโพโลยีแบบดาวซึ่งใช้อัปเป็นศูนย์กลาง สถานีและอัปเชื่อมด้วยสายยูทีพี (Unshield Twisted Pair) ด้วยความยาวไม่เกิน 100 เมตร

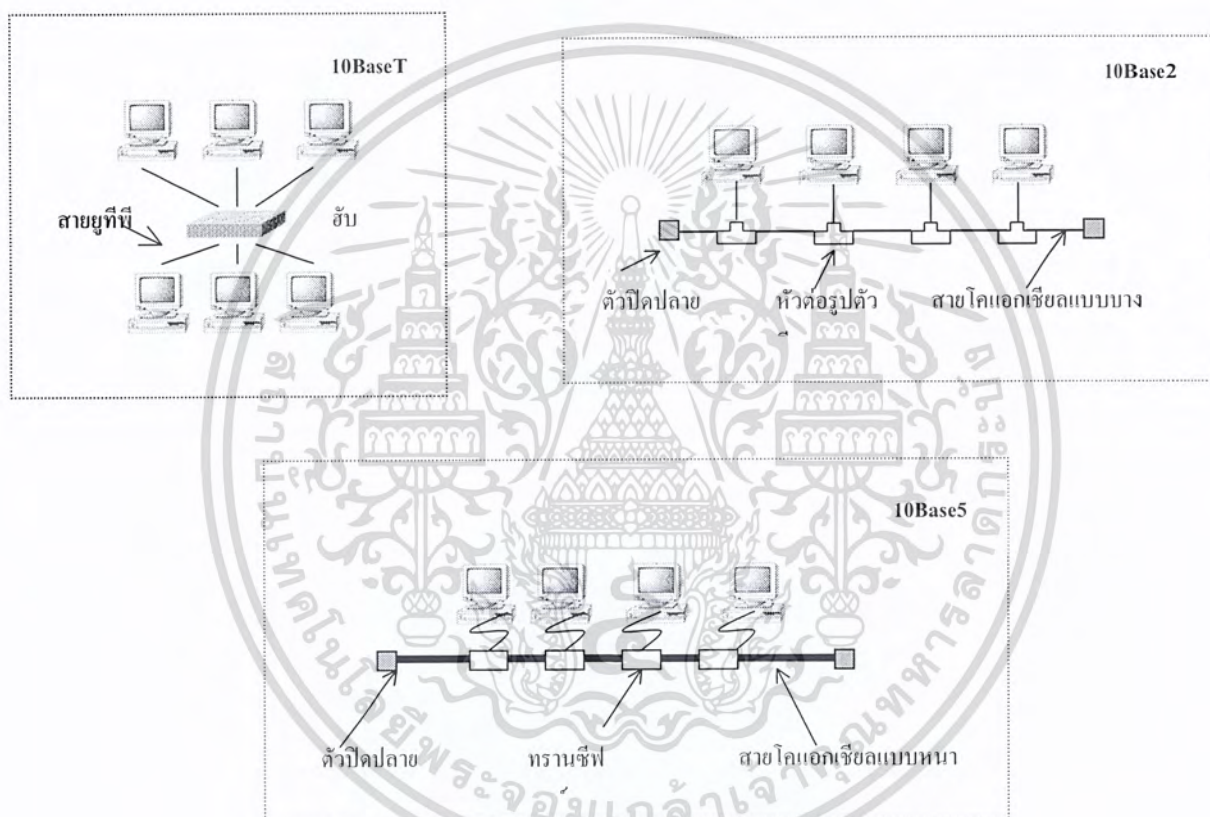
รูปที่ 3-2 แสดงถึงลักษณะเครือข่ายอีเทอร์เน็ตแยกตามประเภทสายสัญญาณรหัสขึ้นต้นด้วย 10 หมายถึงความเร็วสายสัญญาณ 10 เมกะบิตต่อวินาที คำว่า “Base” หมายถึงสัญญาณชนิด “Baseband” รหัสถัดมาหากเป็นตัวเลขหมายถึงความยาวสายต่อเซกเมนต์ในหน่วยหนึ่งร้อยเมตร (5=500, 2 แทนค่า 185) หากเป็นอักษรจะหมายถึงชนิดของสายสัญญาณ เช่น T คือ Twisted pair หรือ F คือ Fiber optics

ส่วนมาตรฐานอีเทอร์เน็ตความเร็ว 100 เมกะบิตต่อวินาทีที่นิยมใช้ในปัจจุบันได้แก่ 100BaseTX และ 100BaseFX สำหรับอีเทอร์เน็ตความเร็วสูงแบบกิกะบิตอีเทอร์เน็ตเริ่มแพร่หลายมากขึ้น ตัวอย่างของมาตรฐานกิกะบิตอีเทอร์เน็ตในปัจจุบันได้แก่ 1000BaseT, 1000BaseLX และ 1000BaseSX เป็นต้น

3.1.2 IEEE 802.4 (Token Bus)

IEEE 802.4 หรือ โทเค็นบัส (Token Bus) มีโทโพลีแบบบัสเช่นเดียวกับ IEEE 802.3 แต่มีข้อกำหนดว่าการเข้าสายใช้สายสื่อสาร โดยใช้โทเค็นพิเศษซึ่งทำหน้าที่เป็นเฟรมสัญญาณกำหนดจังหวะให้สถานีเข้าใช้สายสื่อสาร โทเค็นจะถูกนำส่งจากสถานีหนึ่งไปยังอีกสถานีหนึ่งและวนกลับที่เดิมเป็นวงรอบ สถานีที่ได้รับโทเค็นจะมีสิทธิ์ใช้สายสื่อสารเพื่อส่งข้อมูลได้

สายสื่อสารในโทเค็นบัสที่มักใช้สายโคแอกเชียล และมีอัตราเร็วหลายระดับคือ 1.5 หรือ 10 เมกะบิต การใช้โทเค็นช่วยให้สถานีไม่ต้องแย่งชิงช่องสายสัญญาณเหมือนใน IEEE 802.3 หากแต่ความซับซ้อนของโพรโทคอลทำให้ IEEE 802.4 ไม่เป็นที่นิยมใช้



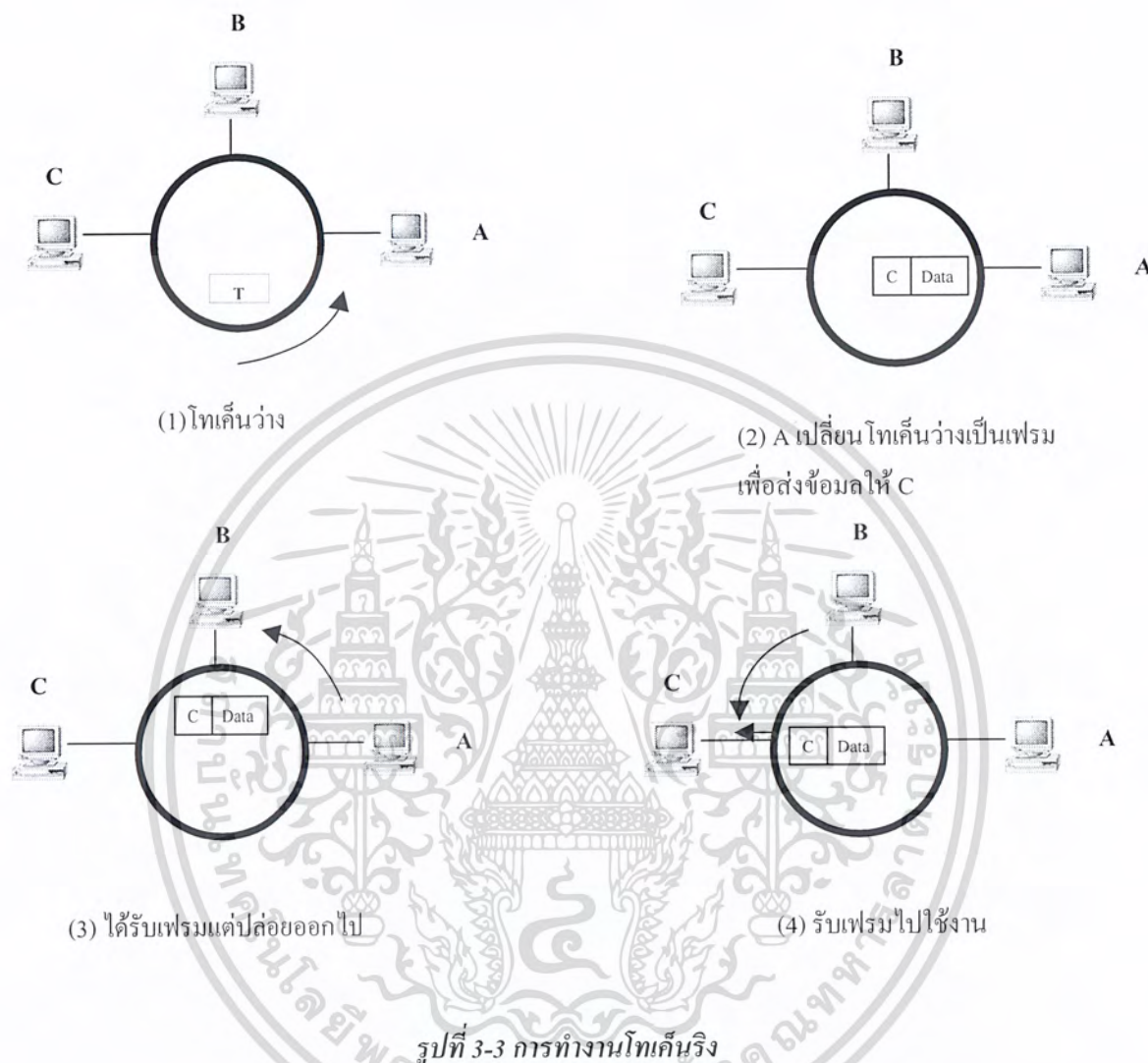
รูปที่ 3-2 อีเทอร์เน็ตประเภทต่างๆ

3.1.3 IEEE 802.5 (Token Ring)

IEEE 802.5 หรือ โทเค็นริง (Token Ring) หรือมักเรียกว่าไอบีเอ็มโทเค็นริงจัดเป็นเครือข่ายที่ใช้โทโพลีแบบวงแหวนด้วยสายคู่เกลียวหรือเส้นใยนำแสง อัตราการส่งข้อมูลของโทเค็นริงที่ใช้โดยทั่วไปคือ 4 และ 16 เมกะบิตต่อวินาที

รูปที่ 3-3 แสดงการทำงานของโทเค็นริง โดยมีเฟรมพิเศษเรียกว่า โทเค็นว่าง (free token) วิ่งวนอยู่ สถานีที่ต้องการส่งข้อมูลจะรอให้โทเค็นว่างเดินทางมาถึงแล้วรับโทเค็นว่างมาเปลี่ยนเป็น เฟรมข้อมูล

(data frame) โดยใส่แฟล็กแสดงเฟรมข้อมูลและบรรจุแอดเดรสของสถานีต้นทางและปลายทางตลอดจนข้อมูลอื่นๆ จากนั้นจึงปล่อยเฟรมนี้ออกไป



เมื่อสถานีปลายทางได้รับเฟรมจะสำเนาข้อมูลไว้และปล่อยเฟรมให้วนกลับมายังสถานีส่ง สถานีส่งจะตรวจสอบเฟรมและปล่อยโทเค้นว่างคืนสู่เครือข่ายให้สถานีอื่นมีโอกาสดำเนินการส่งข้อมูลต่อไป กลไกแบบส่งผ่านโทเค้นจัดอยู่ในประเภทประเมินเวลาได้ กล่าวคือสามารถคำนวณเวลาสูงสุดที่สถานีมีสิทธิจับโทเค้นเพื่อส่งข้อมูลได้ โทเค้นริงจึงเหมาะสำหรับระบบที่ต้องการความแน่นอนทางเวลาหรืองานแบบเวลาจริง

3.1.4 เอฟดีดีไอ

เอฟดีดีไอ (FDDI : Fiber Distributed Data Interface) มีอัตราส่งข้อมูล 100 เมกะบิตต่อวินาทีจึงมักใช้เป็น แกนหลัก (Backbone) ซึ่งเป็นส่วนของเครือข่ายที่จะต้องรับภาระการสื่อสารในปริมาณมาก และนิยมใช้กับระบบงานเวลาจริงเช่นเดียวกับโทเค้นริงเนื่องจากใช้หลักการของโทเค้นเช่นเดียวกัน

โทโพลีของเอฟดีดีไอเป็นแบบวงแหวนคู่ดังรูป 3.4 ในขณะที่โทเค็นริงมีเพียงวงเดียวเท่านั้น วงแหวนประกอบด้วยวงแหวนหลักเรียกว่า วงแหวนปฐมภูมิ (Primary Ring) และวงแหวนรองเรียกว่า วงแหวนทุติยภูมิ (Secondary Ring) การออกแบบให้มีวงแหวนสองวงเพื่อมีเส้นทางสำรอง หากวงแหวนหลักเกิดชำรุด ระบบจะเปลี่ยนไปใช้วงแหวนสำรองแทน



รูปที่ 3-4 โทโพลีเอฟดีดีไอ

สายสัญญาณที่ใช้กับเอฟดีดีไอมี 2 ประเภทคือเส้นใยนำแสงและสายคู่ตีเกลียว เส้นใยนำแสงที่ใช้ งานมีทั้งแบบ หลายภาวะ (Multi mode) ภาวะเดี่ยว (Single mode) เส้นใยนำแสงแบบหลายภาวะสามารถ เชื่อมสถานีที่อยู่ห่างกันได้ 2 กิโลเมตร ขณะที่แบบภาวะเดี่ยวสามารถเชื่อมสถานีที่อยู่ห่างกันได้ระยะทาง ราว 40 กิโลเมตร ส่วนสายคู่ตีเกลียวจะใช้สายประเภท 5 (Category 5) เอฟดีดีไอที่ใช้สายคู่ตีเกลียวมีชื่อ เรียกเฉพาะว่า ซีดีดีไอ (CDDI : Copper Distributed Data Interface)

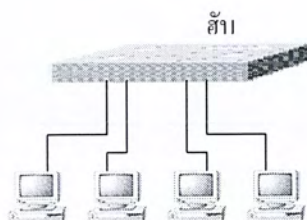
3.2 อุปกรณ์เครือข่าย

เครือข่ายขนาดเล็กมักประกอบด้วยสถานีงานและสถานีเซิร์ฟเวอร์ต่อเชื่อมกัน การเชื่อมเครือข่าย ขนาดเล็กจำนวนหลายเครื่องเข้าด้วยกันเพื่อแลกเปลี่ยนข้อมูลระหว่างกันต้องอาศัยอุปกรณ์เครือข่ายที่มี ลักษณะสมบัติแตกต่างกันไป อุปกรณ์เครือข่ายพื้นฐานที่พบโดยทั่วไปและจะอธิบายในหัวข้อนี้ได้แก่ ฮับ บริดจ์ และเราเตอร์

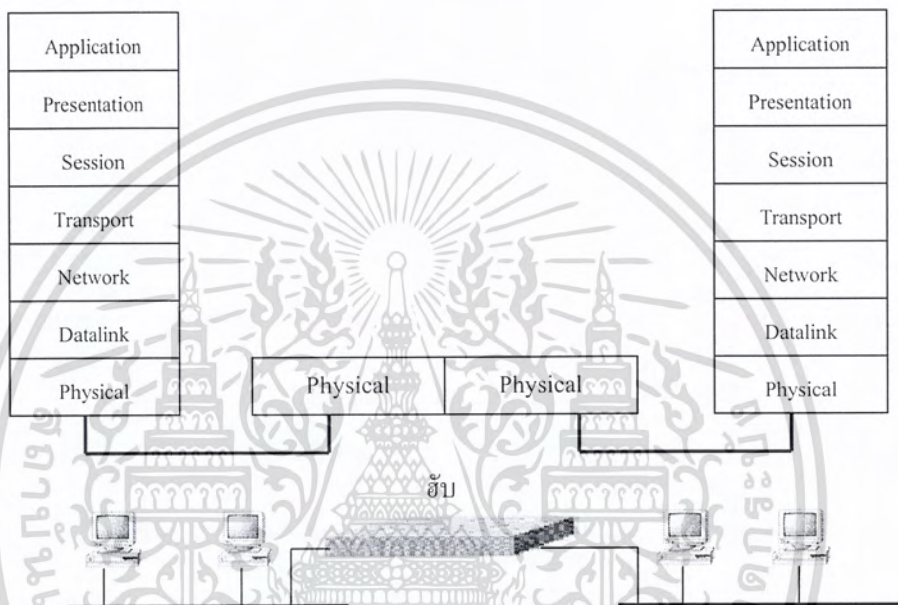
3.2.1 ฮับ

ฮับ (Hub) เป็นอุปกรณ์เชื่อมสถานีเครือข่ายที่ใช้โทโพลีแบบดาว เช่น อีเทอร์เน็ต 10BaseT ดัง รูปที่ 3-5 ฮับมีพอร์ตได้หลายแบบเช่น พอร์ต RJ-45 ใช้กับสายคู่ตีเกลียวในเครือข่าย 10BaseT หรือพอร์ต ไฟเบอร์ใช้กับเส้นใยนำแสงในเครือข่าย 10BaseF

ฮับเป็นอุปกรณ์ในระดับฟิสิคัลเช่นเดียวกับรีพีตเตอร์ดังรูปที่ 3-6 ด้วยเหตุนี้จึงมักเรียกฮับว่าเป็น มัลติพอร์ตรีพีตเตอร์ หน้าที่ของฮับคือขยายสัญญาณและกระจายแพ็กเก็ตไปทุกพอร์ต ฮับใช้เชื่อมต่อเครือ ข่ายประเภทเดียวกันเท่านั้น เครือข่ายที่เชื่อมด้วยฮับจะรวมเป็นเครือข่ายเดียวกันดังนั้นแพ็กเก็ตที่สร้างจาก เครือข่ายหนึ่งผ่านฮับไปอีกเครือข่ายหนึ่ง



รูปที่ 3-5 การใช้ฮับในโทโปโลยีแบบดาว

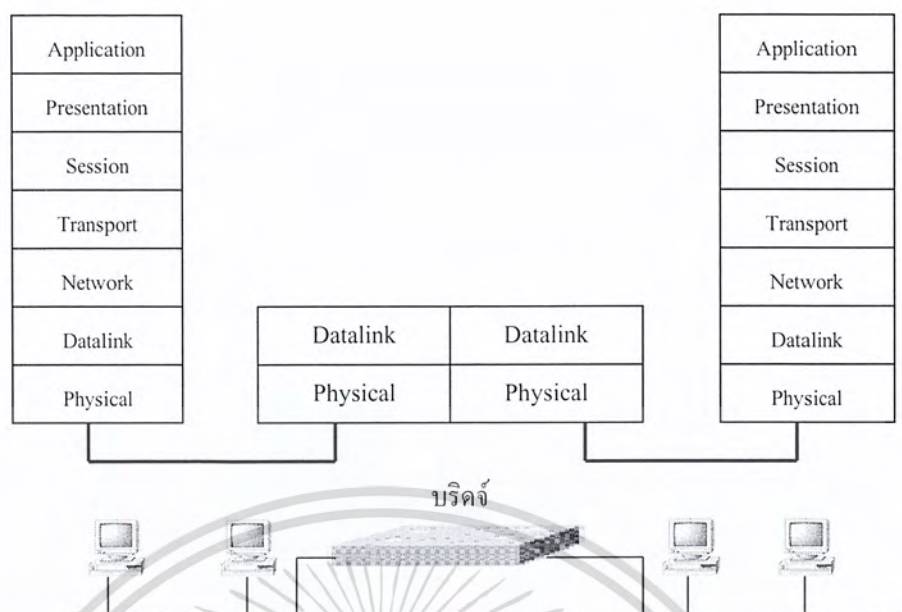


รูปที่ 3-6 แบบจำลองการทำงานของฮับ

3.2.2 บริดจ์

บริดจ์ (Bridge) ทำหน้าที่เชื่อมต่อเครือข่ายย่อยสองเครือข่ายเข้าด้วยกัน บริดจ์ทำงานในระดับชั้นดาต้าลิงก์จึงสามารถใช้เชื่อมต่อเครือข่ายประเภทเดียวกันหรือต่างกันก็ได้ เช่น เชื่อมอีเทอร์เน็ตเข้ากับโทเค็นริงแบบจำลองการทำงานของบริดจ์แสดงได้ดังรูปที่ 3-7

เมื่อใช้บริดจ์เชื่อมต่อเครือข่ายประเภทเดียวกัน บริดจ์จะตรวจสอบว่าจะนำส่งแพ็กเก็ตโดยตรวจสอบฮาร์ดแวร์แอดเดรสปลายทางกับตารางเลือกเส้นทางที่บริดจ์สร้างขึ้น หากมีแพ็กเก็ตที่ต้องข้ามจากบริดจ์จากเครือข่ายหนึ่งไปยังอีกเครือข่ายหนึ่ง บริดจ์จะทำสำเนาแพ็กเก็ตบิตต่อบิต ข้ามไปยังอีกเครือข่ายหนึ่งโดยไม่เปลี่ยนแปลงเฮดเดอร์หรือข้อมูล หากสถานีต้นทางและสถานีปลายทางอยู่ในเครือข่ายเดียวกัน บริดจ์จะได้รับแพ็กเก็ตแต่ไม่ส่งแพ็กเก็ตไปยังเครือข่ายอีกด้านหนึ่ง



รูปที่ 3-7 แบบจำลองการทำงานของบริดจ์

บริดจ์นำส่งบรอดคาสต์แพ็กเก็ตเข้าเครือข่ายโดยไม่ป้องกัน สถานีอีเทอร์เน็ตหรือโทเคนริงต้องดำเนินการกับแพ็กเก็ตที่สถานีอื่นบรอดคาสต์มาให้ บรอดคาสต์แพ็กเก็ตมักเกิดจากการทำงานของโพรโตคอลเออาร์ที บูตพี หรืออาร์ไอพี หากจำนวนบรอดคาสต์แพ็กเก็ตสูงจะส่งผลกระทบต่อสมรรถนะการทำงานของสถานีในเครือข่ายลดลง

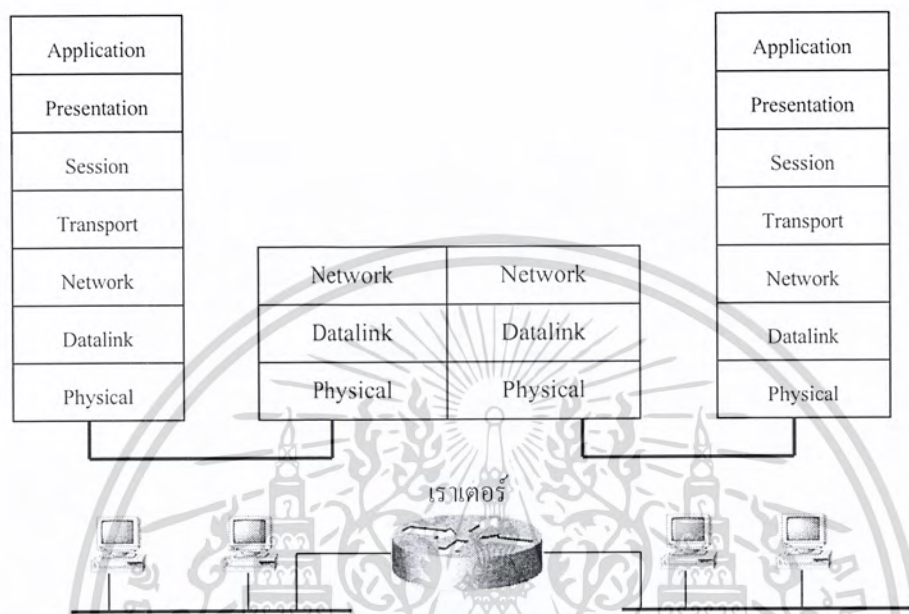
หากสถานีเครือข่ายสร้างบรอดคาสต์แพ็กเก็ตขึ้นพร้อมกันจำนวนมากหรือเรียกว่าพายุบรอดคาสต์ (Broadcast Storms) จะส่งผลกระทบต่อเครือข่าย การสื่อสารในเครือข่ายจะช้าลงจนสังเกตได้หากมีบรอดคาสต์แพ็กเก็ตตั้งแต่ 40 แพ็กเก็ตต่อวินาทีขึ้นไป หรือหากมีบรอดคาสต์แพ็กเก็ตในอัตรา 100 แพ็กเก็ตต่อวินาทีอาจทำให้การสื่อสารทั้งหมดหยุดชะงักได้

สวิตช์ทำหน้าที่เช่นเดียวกับบริดจ์ในการแบ่งเครือข่ายใหญ่ออกเป็นเครือข่ายย่อย และป้องกันไม่ให้เกิดการส่งเฟรมที่ไม่จำเป็นจากเครือข่ายหนึ่งเข้าไปยังอีกเครือข่ายหนึ่ง สวิตช์มักประกอบด้วยพอร์ตจำนวนมาก ต่างจากบริดจ์ที่มีเพียงสองพอร์ต แต่ละพอร์ตของสวิตช์สามารถเชื่อมโยงระหว่างสวิตช์ด้วยกันหรือเชื่อมสถานีเข้าสู่สวิตช์โดยตรง สวิตช์บางรุ่นสามารถจัดแบ่งเครือข่ายย่อยตามแต่ละพอร์ตได้โดยใช้ วีแลน (VLAN : Virtual Lan) ตามมาตรฐาน IEEE 802.1q

คุณลักษณะสำคัญของสวิตช์คือมีอิเล็กทรอนิกส์สวิตช์ความเร็วสูงทำหน้าที่ส่งเฟรมจากพอร์ตต้นทางสู่ปลายทางโดยไม่รับกวนพอร์ตอื่น ลักษณะนี้ต่างจากบริดจ์หรือเราเตอร์ซึ่งเฟรมถูกส่งจากพอร์ตหนึ่งไปยังอีกพอร์ตหนึ่งโดยอาศัยการประมวลผลของไมโครโพรเซสเซอร์

3.2.3 เราเตอร์

เราเตอร์เป็นอุปกรณ์ที่ทำงานในระดับชั้นเน็ตเวิร์กตามรูปที่ 3-8 เราเตอร์ทำงานร่วมกับฮาร์ดแวร์ในระดับดาต้าลิงก์ได้หลายรูปแบบ หน้าที่ของเราเตอร์คือจัดแบ่งเครือข่ายและเลือกเส้นทางที่เหมาะสมเพื่อนำส่งแพ็กเก็ต เราเตอร์จะห้องกันการบรอดคาสต์แพ็กเก็ตเกิดจากเครือข่ายหนึ่งไม่ให้ข้ามมาอีกเครือข่ายหนึ่ง



รูปที่ 3-8 แบบจำลองการทำงานของเราเตอร์

ในอินเทอร์เน็ตมักเรียกเราเตอร์ว่า ไอพีเราเตอร์ (IP Router) เนื่องจากเราเตอร์ทำงานตามข้อกำหนดของโพรโตคอลไอพี เราเตอร์ทำหน้าที่เลือกเส้นทางโดยสร้างแผนที่เครือข่ายและเก็บอยู่ในรูปตารางเส้นทาง เมื่อเราเตอร์ได้รับแพ็กเก็ตก็จะตรวจสอบแอดเดรสปลายทางและส่งแพ็กเก็ตไปยังอีกอินเทอร์เน็ตเฟสที่เป็นช่องทางไปสู่เครือข่ายปลายทาง เราเตอร์ประกอบด้วยหลายอินเทอร์เน็ตเฟสเพื่อเชื่อมต่อกับเครือข่ายต่างชนิดเข้าด้วยกันได้ ตัวอย่างเช่น อินเทอร์เน็ตเฟสไอพีเน็ต-โทเค็นริง เอพีดีดีไอ เอทีเอ็ม หรืออินเทอร์เน็ตเฟสแบบอนุกรม เป็นต้น

ซอฟต์แวร์ในเราเตอร์จะรับแพ็กเก็ตและเก็บเข้าบัฟเฟอร์ก่อนนำไปประมวลผล วิธีนี้ต่างจากสวิตช์ซึ่งอาศัยฮาร์ดแวร์ตรวจสอบแอดเดรสปลายทางของแพ็กเก็ตและ “สวิตช์” แพ็กเก็ตนั้นจากพอร์ตหนึ่งไปสู่อีกพอร์ตหนึ่ง

บทที่ 4

ไอพีแอดเดรส

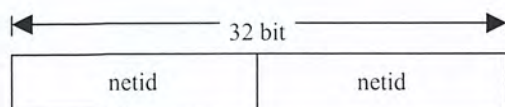
ไอพีแอดเดรสเป็นส่วนประกอบสำคัญอย่างหนึ่งในเครือข่ายที่ซีพี/ไอพี ไอพีแอดเดรสเป็นแอดเดรสทางซอฟต์แวร์ประจำสถานีเครือข่ายผู้ออกแบบเครือข่ายจำเป็นต้องศึกษาและเข้าใจรูปแบบของไอพีแอดเดรสโดยละเอียด การจัดแบ่งแอดเดรสออกเป็นคลาส และการจัดแบ่งเครือข่ายย่อยหรือซับเน็ต รวมทั้งสามารถออกแบบซับเน็ตโดยเลือกใช้ไอพีแอดเดรสได้อย่างถูกต้อง เนื้อหาซึ่งจะกล่าวถึงในบทนี้มีดังนี้

4.1 ไอพีแอดเดรส

อุปกรณ์ที่เชื่อมต่อเข้าเครือข่ายและสามารถทำงานตามข้อกำหนดของซีพี/ไอพีจะต้องมีแอดเดรสประจำอุปกรณ์นั้น อุปกรณ์ดังกล่าวอาจเป็นโฮสต์ เราเตอร์ เครื่องพิมพ์ หรือแม้กระทั่งอุปกรณ์สำนักงาน เช่น โทรศัพท์หรือเครื่องถ่ายเอกสาร ไอพีรุ่นสี่กำหนดให้ใช้ไอพีแอดเดรสขนาด 32 บิต อุปกรณ์ที่เชื่อมกับอินเทอร์เน็ตจะมีไอพีแอดเดรส 32 บิตประจำอินเทอร์เน็ตที่ไม่ซ้ำกัน อุปกรณ์อย่างเราเตอร์จะมีหลายอินเทอร์เน็ตเฟสซึ่งแต่ละอินเทอร์เน็ตเฟสจะมีไอพีแอดเดรสหลายค่าตามจำนวนอินเทอร์เน็ตเฟสโดยไม่ซ้ำค่ากัน แต่ถ้าเป็นเครื่องคอมพิวเตอร์หรือโฮสต์ปกติจะมีแค่อินเทอร์เน็ตเฟสเดียว จึงมักเรียกว่าไอพีแอดเดรสเป็นแอดเดรสประจำโฮสต์

แอดเดรสขนาด 32 บิตมีจำนวนแอดเดรสรวมเท่ากับ 2^{32} (4,294,967,296) แต่เมื่อนำมาจัดสรรแล้วไม่สามารถใช้งานได้ครบทั้งหมด ไอพีแอดเดรสนิยมเขียนในรูปเลขฐานสิบ โดยแบ่งเลข 32 บิตเป็น 4 ไบต์ แต่ละไบต์แทนด้วยตัวเลขฐานสิบหนึ่งตัวคั่นแต่ละไบต์ใช้ด้วยเครื่องหมายจุด เช่น แอดเดรส 1001110 01101100 00000010 00000001 จะเขียนได้เป็น 161.246.2.1

แอดเดรสขนาด 32 บิต ประกอบขึ้นจากหมายเลขสองส่วนคือ เลขเครือข่าย (Network Number หรือ Network Identifier หรือ NetID) และ เลขโฮสต์ (Host Number หรือ Host Identifier หรือ HostID) เลขเครือข่ายใช้สำหรับจัดคลาสเครือข่าย ส่วนเลขโฮสต์ใช้ระบุหมายเลขโฮสต์ (หรืออีกนัยหนึ่งคืออินเทอร์เน็ตเฟสของโฮสต์) ในเครือข่าย ไอพีแอดเดรสจึงแบ่งได้เป็นสองส่วนตามรูปที่ 4-1 จำนวนบิตที่ใช้สำหรับเลขเครือข่ายและเลขโฮสต์ขึ้นอยู่กับคลาสที่สังกัด



รูปที่ 4-1 รูปแบบของไอพีแอดเดรส

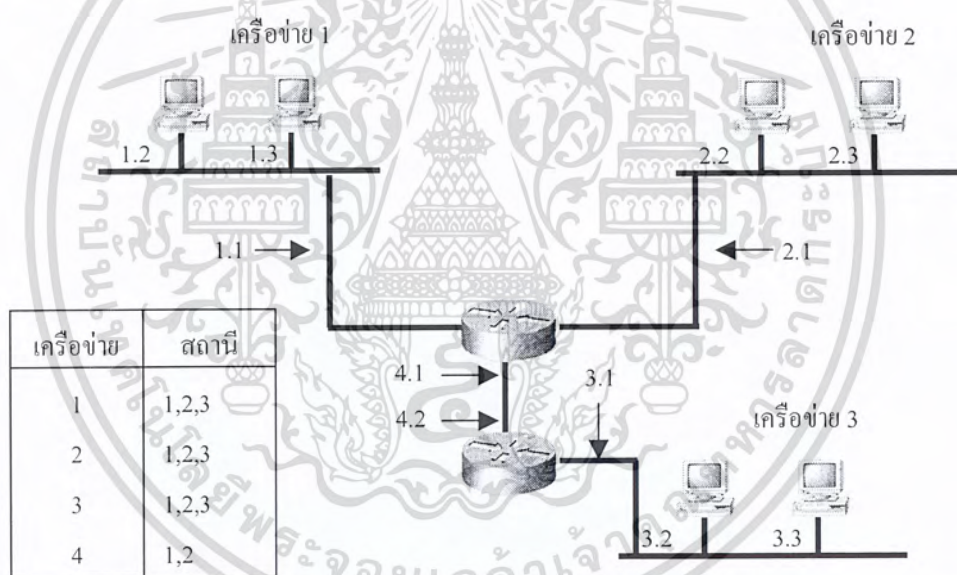
ในปัจจุบันฟิลด์กำหนดเลขเครือข่ายนิยมเรียกว่า พรีฟิกซ์เครือข่าย (Network-Prefix) เพราะทุกโฮสต์ในเครือข่ายจะต้องมีพรีฟิกซ์หรือบิตนำหน้าเหมือนกัน ตัวอย่างเช่นหากมีเลขเครือข่ายจำนวน 16 บิตก็จะเรียกว่า พรีฟิกซ์ 16 เป็นต้น

4.11 ความสำคัญของเลขเครือข่ายและเลขโฮสต์

การจัดแบ่งไอพีแอดเดรสออกเป็นสองส่วนที่ประกอบด้วยเลขเครือข่ายและเลขโฮสต์ก็เพื่อประโยชน์ในการดูแลระบบ เราเตอร์จะอาศัยเลขเครือข่ายเพื่อเลือกเส้นทางส่งแพ็กเก็ตด้วยหลักการต่อไปนี้

โฮสต์ที่มีเลขเครือข่ายชุดเดียวกันย่อมอยู่ภายในเครือข่ายเดียวกัน และสามารถสื่อสารถึงกันด้วยเฟรมดาต้าลิงค์โดยไม่ต้องพึ่งเราเตอร์

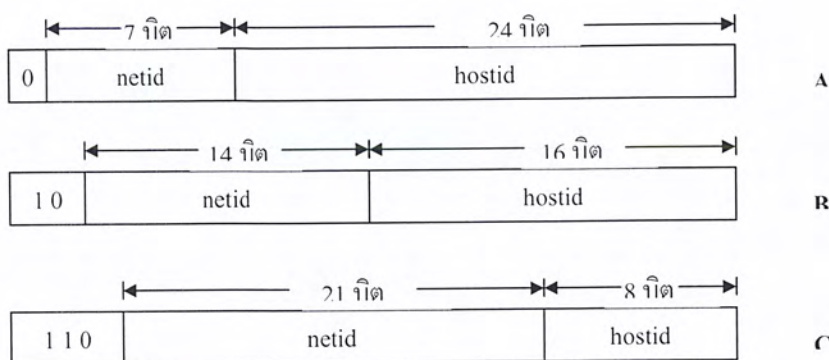
โฮสต์ที่มีเลขเครือข่ายต่างกันจะอยู่ต่างเครือข่ายกัน การสื่อสารระหว่างโฮสต์จะอาศัยเราเตอร์ที่เชื่อมต่อเครือข่ายเป็นผู้นำส่งแพ็กเก็ต เราเตอร์อาจเชื่อมเครือข่ายที่อยู่ติดกันหรือส่งแพ็กเก็ตผ่านเราเตอร์อื่นไปยังปลายทางดังรูป



รูปที่ 4-2 เราเตอร์เชื่อมโยงเครือข่ายที่มีเลขเครือข่ายต่างกัน

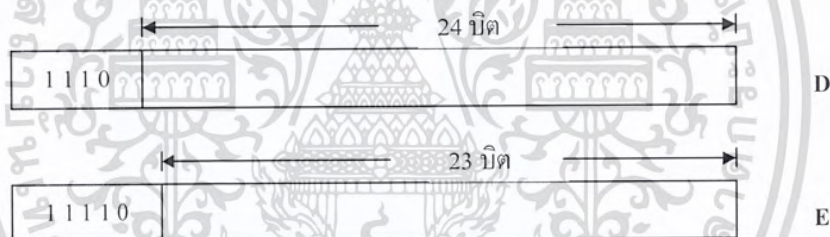
4.1.2 การจัดคลาสเครือข่าย

ไอพีแอดเดรสมีการจัดแบ่งออกเป็นกลุ่มหรือคลาส เครือข่ายที่ใช้งานในปัจจุบันมักสังกัดอยู่ในคลาสใดคลาสหนึ่งคือคลาส A, B หรือ C การแบ่งคลาสอาศัยจำนวนพรีฟิกซ์เครือข่ายที่แตกต่างกันตามรูปที่ 4-3 แต่ละคลาสจึงมีจำนวนเครือข่ายในสังกัดและจำนวนโฮสต์ต่อเครือข่ายไม่เท่ากัน



รูปที่ 4-3 การแบ่งคลาสเครือข่าย

การจัดคลาสตามรูปที่ 4-3 เป็นการจัดแบ่งตามการใช้งานเครือข่ายทั่วไป ในขณะที่ยังมีอีก 2 คลาสซึ่งใช้เพื่อจุดประสงค์เฉพาะได้แก่ คลาส D และ E ดังรูปที่ 4-4 เครือข่ายคลาส D เป็นเครือข่ายแบบมัลติคาสต์ซึ่งจะกล่าวในบทที่ 12 ส่วนคลาส E สงวนไว้ใช้งานหากมีความจำเป็นอื่นใดในอนาคต ทั้งสองคลาสนี้ไม่ได้แบ่งเลขโฮสต์จึงไม่กำหนดจำนวนโฮสต์ไว้



รูปที่ 4-4 การแบ่งคลาส D และ E

การจัดคลาสโดยใช้พรีฟิกซ์เป็นการผนวกข้อมูลเพื่อใช้ในการเลือกเส้นทาง เช่น หากตรวจพบว่าพรีฟิกซ์ 2 บิตแรกมีค่าเป็น 10 แสดงว่าเป็นแอดเดรสในคลาส B ซึ่งมีค่า 16 บิตแรกกำหนดกลุ่มเครือข่ายและ 16 บิตถัดมาเป็นเลขโฮสต์

4.1.3 ลักษณะสำคัญของแต่ละคลาส

จำนวนเครือข่ายในแต่ละคลาสและจำนวนโฮสต์สูงสุดที่มีได้ สามารถคำนวณได้จากจำนวนบิตที่ใช้งานตามสูตร 2^n เมื่อ n คือจำนวนบิต ตัวอย่างเช่น ในคลาส B มีเลขโฮสต์จำนวน 16 บิต จึงมีโฮสต์ได้ไม่เกิน 2^{16} ซึ่งเท่ากับ 65,536 แต่เลขโฮสต์ที่ทุกบิตเป็น “0” และเป็น “1” จะสงวนไว้ใช้งานกรณีเฉพาะจำนวนโฮสต์จึงลดลงไป 2 โฮสต์ทุกเครือข่าย หรือมีโฮสต์ไม่เกิน $2^{16} - 2 = 65,534$ สูตร $2^n - 2$ นี้จะใช้กับการคำนวณจำนวนเครือข่ายในคลาสและจำนวนโฮสต์ ทั้งคลาส A, B และ C ดังนี้

คลาส A

เครือข่ายในคลาส A มีบิตซ้ายสุดเป็น 0 และใช้ 7 บิตถัดมากำหนดเครือข่าย ส่วนอีก 24 เป็นเลขโฮสต์ คลาส A จึงมีเลขเครือข่ายได้ 2^7 หรือ 128 ค่า แต่เครือข่าย 0.0.0.0 และ 127.0.0.0 สงวนไว้เป็นแอดเดรสเฉพาะงานคือ 0.0.0.0 เป็นแอดเดรสกำหนดเส้นทางโดยปริยาย (Default Route) ส่วน 127.0.0.0 เป็นแอดเดรสลูปแบ็คคือเป็นแอดเดรสที่ใช้เพื่อเชื่อมต่อเข้าสู่อินเทอร์เน็ต ดังนั้นจำนวนเครือข่ายในคลาส A จึงมีได้ 126 เครือข่ายคือเลขที่ขึ้นต้นด้วย 1.0.0.0 ถึง 126.0.0.0

แต่ละเครือข่ายในคลาส A มีแอดเดรสได้ $2^{24} - 2$ หรือเท่ากับ 16,777,214 คือตั้งแต่ 0.0.1 ถึง 255.255.254 เครือข่ายในคลาส A ใช้กับหน่วยงานขนาดใหญ่ที่ต้องการแอดเดรสเป็นจำนวนมาก เครือข่ายคลาสนี้จัดสรรให้กับหน่วยงานในยุคแรกเริ่มของอินเทอร์เน็ต แอดเดรสเครือข่ายที่เหลืออยู่ส่วนใหญ่จะสงวนไว้

สังเกตว่าในคลาส A นี้เมื่อกล่าวถึงเฉพาะเลขเครือข่ายก็จะเขียนเฉพาะค่าที่แสดงเลขเครือข่ายที่ขนาด 8 บิต เท่านั้นเช่น 2 หรือ 26 ในทำนองเดียวกันเมื่อกล่าวเฉพาะเลขโฮสต์ก็จะเขียนเฉพาะหมายเลขเครือข่ายโดยให้เลขโฮสต์เป็น “0” เช่น 2.0.0.0 รูปแบบการเขียนเช่นนี้ใช้กับคลาส B และ C เช่นกัน

คลาส B

เครือข่ายในคลาส B มีบิตแรกเริ่มเป็น 10 และใช้ 14 บิตถัดมากำหนดเลขเครือข่ายจำนวนบิตที่กำหนดเลขโฮสต์มีขนาด 16 บิต คลาส B จึงมีสมาชิกเครือข่ายได้ $2^{14} - 2$ หรือ 16,382 คือตั้งแต่ 128.1.0.0 ถึง 192.254.0.0 แต่ละเครือข่ายมีเลขโฮสต์ได้ $2^{16} - 2$ หรือเท่ากับ 65,534 แอดเดรส หรือตั้งแต่ 0.1 ถึง 255.254

เครือข่ายในคลาส B มักจัดสรรให้กับหน่วยงานขนาดกลาง ในปัจจุบันมีเครือข่ายในคลาส B เหลือไม่มากนัก และมักไม่จัดสรรเครือข่ายในคลาสนี้ให้กับผู้จดทะเบียนรายใหม่หากไม่มีความจำเป็นอย่างแท้จริง

คลาส C

เครือข่ายในคลาส C มีพรีฟิกซ์ 110 และใช้ 21 บิตถัดมาเป็นเลขเครือข่าย จำนวนบิตที่เป็นเลขโฮสต์มีเพียง 8 บิต คลาส C จึงมีเลขเครือข่ายได้ตั้งแต่ 192.0.1.0 ถึง 223.255.254.0 รวม 2,097,150 เครือข่าย แต่ละเครือข่ายมีเลขโฮสต์ได้ตั้งแต่ 1 ถึง 254

จำนวนแอดเดรสได้จำกัดเพียง 254 แอดเดรสทำให้เครือข่ายเหมาะสมสำหรับหน่วยงานขนาดเล็ก หากจำเป็นต้องใช้โฮสต์มากกว่านี้ต้องขอใช้เครือข่ายคลาส C หลายเครือข่าย

คลาส D และ E

เครือข่ายในคลาส C และ D ไม่มีการจัดแบ่งเลขเครือข่ายและเลขโฮสต์ คลาส D มี 3 บิตแรกเป็น 111 จึงมีแอดเดรสตั้งแต่ 224.0.0.0 ถึง 239.255.255.255 แอดเดรสในคลาสนี้เรียกว่า มัลติคาสต์แอดเดรส (Multicast Address) เนื่องจากใช้ในเครือข่ายมัลติคาสต์

สำหรับคลาส E มีแอดเดรสจาก 240.0.0.0 ถึง 254.255.255.255 ซึ่งสำรองไว้เพื่อความจำเป็นเฉพาะงานในอนาคต

4.2 การแบ่งเครือข่ายย่อย

เครือข่ายที่สังกัดในคลาส A และ B เป็นเครือข่ายที่มีจำนวนโฮสต์ได้เป็นจำนวนมาก กล่าวคือ 16,777,214 และ 65,534 ตามลำดับ ในทางปฏิบัติแล้วเราไม่สามารถต่อเชื่อมโฮสต์ทั้งหมดในเครือข่ายเดียวกัน ได้เพราะข้อจำกัดทางฮาร์ดแวร์ ผู้วางระบบจึงต้องจัดแบ่งเครือข่ายขนาดใหญ่ให้เล็กลงเป็นเครือข่ายขนาดเล็กย่อย หรือซับเน็ต (Subnet) การแบ่งซับเน็ต นอกจากจะจัดจำนวนโฮสต์ให้เหมาะสมกับฮาร์ดแวร์ของเครือข่ายแล้วยังช่วยอำนวยความสะดวกในการบริหารเครือข่าย

การจัดซับเน็ตใช้วิธีแบ่งบางส่วนของเลขโฮสต์มาใช้เป็นเลขซับเน็ต (SubnetID) เพื่อกำหนดว่าเป็นเครือข่ายย่อยที่เท่าใด ตัวอย่างเช่นเครือข่าย 161.246.0.0 ซึ่งอยู่ในคลาส B อาจใช้ 8 บิตแรกของเลขโฮสต์เป็นเลขซับเน็ต และ 8 บิตที่เหลือใช้สำหรับเลขโฮสต์ดังรูปที่ 4-5

16 บิต	8 บิต	8 บิต
161.246	subnetid	hostid

รูปที่ 4-5 ตัวอย่างการแบ่งเครือข่ายย่อยของ 161.246

จำนวนบิตของเลขซับเน็ตเป็นตัวกำหนดจำนวนเครือข่ายย่อย ซับเน็ตขนาด 8 บิตสำหรับเครือข่าย 161.246.0.0 จะมี 254 ซับเน็ต ($2^{subnetid} - 2$) แต่ละซับเน็ตมี 254 โฮสต์ ($2^{hostid} - 2$) ดังตารางที่ 4.1 เลขซับเน็ตที่ทุกบิตเป็น “1” และ “0” จะสงวนไว้ใช้งานเฉพาะ ดังนั้นซับเน็ต 161.246.0.0 และ 161.246.255.0 จึงนำมาใช้ไม่ได้

ซับเน็ตที่	เครือข่ายย่อย	แอดเดรสเริ่มต้น	แอดเดรสสุดท้าย
1	161.246.1.0	161.246.1.1	161.246.1.254
2	161.246.2.0	161.246.2.1	161.246.2.254
3	161.246.3.0	161.246.3.1	161.246.3.254
..	..	..	..
..	..	..	..
252	161.246.252.0	161.246.252.1	161.246.252.254
253	161.246.253.0	161.246.253.1	161.246.253.254
254	161.246.254.0	161.246.254.1	161.246.254.254

ตาราง 4-1 การจัดแบ่งเครือข่าย 161.246 ด้วยซับเน็ต 8 บิต

4.2.1 ซับเน็ตมาสก์

เมื่อผู้วางระบบเลือกขนาดซับเน็ตแล้วจะกำหนดพารามิเตอร์เพื่อใช้บอกให้โฮสต์และเราเตอร์ทราบว่ซับเน็ตที่ใช้งานมีขนาดกี่บิต ค่านี้เรียกว่า ซับเน็ตมาสก์ (Subnet Mask)

ซับเน็ตมาสก์เป็นตัวเลข 32 บิต ซึ่งเขียนอยู่ในรูป Dotted-Decimal เช่นเดียวกับการเขียนไอพีแอดเดรส ซับเน็ตมาสก์จะมีบิตที่ตรงกับเลขเครือข่ายและเลขซับเน็ตเท่ากับ “1” ส่วนบิตที่ตรงกับเลขโฮสต์มีค่าเท่ากับ “0” การเลือกซับเน็ตมาสก์ควรใช้ค่าที่มีบิต “1” อยู่ติดกันจากทางซ้ายมือไปทางขวามือเสมอ

ตัวอย่างเครือข่าย 161.246.0.0 ซึ่งแบ่งให้มีเลขซับเน็ตและเลขโฮสต์อย่างละ 8 บิตจะมีค่าซับเน็ตมาสก์เท่ากับ 255.255.255.0 ค่านี้คำนวณได้จากการเขียนไอพีแอดเดรสทั้ง 4 หลัก และใส่เลขฐานสองค่า “1” ให้ครบทุกบิตที่เป็นเลขเครือข่ายและเลขซับเน็ต จากนั้นให้ใส่ค่า “0” สำหรับเลขโฮสต์ แล้วจึงแปลงเลขฐานสองที่

	8 บิต	8 บิต	8 บิต	8 บิต
1. นำค่าไอพีแอดเดรส	161	246	SubnetID	HostId
2. กำหนดบิต “1” และ “0”	11111111	11111111	11111111	00000000
3. แปลงเป็นเลขฐานสิบ	255	255	255	0

เครือข่าย 161.246.0.0 ซึ่งใช้ซับเน็ตมาสก์เท่ากับ 255.255.255.0 เรียกว่ามีซับเน็ตมาสก์ 24 บิต เนื่องจากมีบิตที่มีค่า “1” จำนวน 24 บิต หรือเขียนตามรูปแบบที่นิยมใช้ในปัจจุบันคือ 161.246.0.0/24 โดยเรียกว่าเครือข่าย 161.246.0.0 มีพรีฟิกซ์ 24 บิต

สังเกตว่า 161.246.0.0/24 ใช้เลขซับเน็ตจำนวน 8 บิต ดังนั้นนอกจากจะเรียกว่ามีพรีฟิกซ์ 24 บิตแล้ว ยังเรียกได้อีกว่าใช้ซับเน็ตบิตจำนวน 8 บิต

4.2.2 ดีฟอลต์ซับเน็ตมาสก์ (Default Subnet Mask)

การติดตั้งโฮสต์เข้าเครือข่ายนอกจากจะต้องกำหนดไอพีแอดเดรสแล้วต้องกำหนดค่าซับเน็ตมาสก์ตามที่ผู้ดูแลระบบกำหนดไว้ด้วย ถึงแม้ว่าในบางเครือข่ายเช่นเครือข่ายในคลาส C ซึ่งมีโฮสต์และไม่ได้แบ่งให้มีซับเน็ต ขั้นตอนการติดตั้งโฮสต์ยังจำเป็นต้องใส่ค่าซับเน็ตมาสก์เช่นกัน แต่ค่าซับเน็ตมาสก์นี้เรียกว่า ดีฟอลต์ซับเน็ตมาสก์ (Default Subnet Mask) ดีฟอลต์ซับเน็ตมาสก์ของเครือข่ายคลาส A, B และ C แสดงได้ดังตารางที่ 4.2

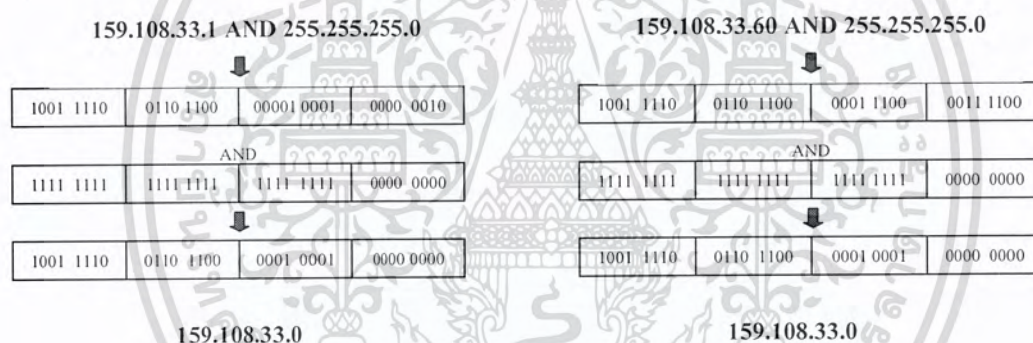
ผู้ดูแลระบบสามารถวางแผนจัดการเครือข่ายโดยเลือกทำซับเน็ตหรือไม่ทำซับเน็ตตามความต้องการ โดยปกติแล้วผู้ดูแลระบบเครือข่ายในคลาส A และ B ไม่สามารถหลีกเลี่ยงการใช้ซับเน็ตได้

คลาส	ดีฟอลต์ซับเน็ตมาสก์	ดีฟอลต์ซับเน็ตมาสก์(ฐาน 2)
A	255.0.0.0	11111111.00000000.00000000.00000000
B	255.255.0.0	11111111.11111111.00000000.00000000
C	255.255.255.0	11111111.11111111.11111111.00000000

ตารางที่ 4-2 ค่าดีฟอลต์ซับเน็ตมาสก์

4.2.3 การเลือกเส้นทางในซับเน็ต

ซับเน็ตมาสก์นอกจากจะช่วยจัดแบ่งเครือข่ายย่อยแล้ว ยังใช้ประโยชน์ในการเลือกเส้นทางส่งไอพีตามแอมระหว่างเครือข่ายย่อยด้วย เช่น โฮสต์ 161.246.33.2 ในเครือข่าย 161.246.0.0/24 (ซับเน็ตมาสก์ 255.255.255.0) ต้องการส่งข้อมูลไปยังโฮสต์ 161.246.33.60 โพรโตคอลไอพีจะทำหน้าที่เลือกเส้นทางโดยนำแอดเดรส 161.246.33.2 และ 161.246.33.60 มาผ่านลอจิก “AND” บิตต่อบิตกับค่าซับเน็ตมาสก์ดัง



รูปที่ 4-6 การตรวจหาแอดเดรสซับเน็ตเพื่อเลือกเส้นทาง

ผลลัพธ์จากลอจิก “AND” ของแอดเดรสและเน็ตมาสก์ข้างต้นได้ค่าแอดเดรสซับเน็ต 161.246.33.0 เท่ากัน ซึ่งหมายความว่าโฮสต์ทั้งสองอยู่ในซับเน็ตเดียวกัน หากเครือข่ายที่ใช้คืออีเทอร์เน็ตแล้ว โฮสต์ 161.246.33.2 จะสร้างแพ็กเก็ตโดยระบุอีเทอร์เน็ตแอดเดรสของ 161.246.33.60 โดยไม่ต้องส่งแพ็กเก็ตให้เราเตอร์ดำเนินการ โปรดสังเกตว่าการใช้ลอจิก “AND” เป็นการใช้ซับเน็ตมาสก์เพื่อ “มาสก์” ให้ได้เฉพาะเลขเครือข่าย ค่าซับเน็ตมาสก์จึงเป็นเสมือน หน้ากาก กรอบเอาเลขเครือข่ายออกมา

ในกรณีที่โฮสต์ปลายทางอยู่ต่างเครือข่ายกับโฮสต์ต้นทาง เช่น แอดเดรสของโฮสต์ต้นทางคือ 161.246.33.2 และโฮสต์ปลายทางคือ 161.246.40.5 ผลจากลอจิก “AND” ระหว่าง 161.246.40.5 กับมาสก์ 255.255.255.0 จะได้ค่า 158.0108.40.0 ซึ่งต่างจาก 161.246.33.0 ดังนั้นโฮสต์ 161.246.33.2 จะสรุปว่า 161.246.40.0 ซึ่งต่างจาก 161.246.33.0 ดังนั้นโฮสต์ 161.246.33.2 จะสรุปว่า 161.246.40.0 อยู่ต่างซับเน็ต และจะส่งแพ็กเก็ตไปยังเราเตอร์เพื่อให้เราเตอร์เพื่อที่เราเตอร์นำส่งแพ็กเก็ตต่อไป

บทที่ 5

โปรโตคอลเลือกเส้นทาง (Routing Protocol)

การเลือกเส้นทางเป็นหน้าที่สำคัญอย่างหนึ่งของไอพี เราเตอร์อาศัยตารางเส้นทาง (Routing Table) เพื่อนำส่งดาต้าแกรมไปยังเครือข่ายปลายทาง ตารางเส้นทางอาจปรับเปลี่ยนได้ตามสภาพเครือข่าย อินเทอร์เน็ตใช้โปรโตคอลเลือกเส้นทาง หรือผู้ดูแลระบบเป็นคนปรับเปลี่ยนเส้นทางเอง

5.1 การเลือกเส้นทาง (Routing)

การเลือกเส้นทางเป็นกระบวนการที่เกิดขึ้นในระดับชั้นที่ 3 หรือระดับเน็ตเวิร์ก ของแบบจำลอง ทีซีพี/ไอพี ซอฟต์แวร์ไอพีที่อยู่ในโฮสต์หรือเราเตอร์จะนำส่งดาต้าแกรมไปตามเส้นทาง โดยอาศัยเลขเครือข่ายของไอพีแอดเดรสตามแต่ละคลาส เลขเครือข่ายเป็นเสมือนค่ากำหนดตำแหน่งปลายทางของเครือข่ายซึ่งคล้ายกับรหัส 3 ตัวแรกที่กำหนดหมายเลขโทรศัพท์ ระบบรหัสชุมสายจะกำหนดที่ตั้งทางภูมิศาสตร์ ด้วยว่าอยู่ที่ทิศทางใด แต่เลขเครือข่ายในไอพีแอดเดรสไม่ได้มีส่วนสัมพันธ์กับที่ตั้งของเครือข่าย

เครือข่ายโดยทั่วไปประกอบด้วยสถานีปลายทาง (End Node) และเราเตอร์หรืออุปกรณ์อื่นทำงานร่วมกัน โดยกระบวนการเลือกหาเส้นทางจะเกิดขึ้นทั้งที่สถานีปลายทางและที่เราเตอร์ คือ

1. สถานีปลายทางที่เป็นสถานีส่งต้องทราบว่าจะนำส่งดาต้าแกรมให้เราเตอร์ได้อย่างไรและเมื่อใด
2. เราเตอร์ต้องทราบเส้นทางเชื่อมโยงไปยังเราเตอร์ตัวอื่น เพื่อส่งดาต้าแกรมไปตามเส้นทางที่เหมาะสมที่สุด
3. เราเตอร์ที่เชื่อมกับเครือข่ายของสถานีปลายทางต้องทราบถึงวิธีส่งดาต้าแกรมไปยังสถานีปลายทางนั้น

ในตอนแรกสถานีต้นทางจะเป็นผู้ตัดสินใจขั้นแรกว่าต้องส่งดาต้าแกรมไปยังสถานีใดในเครือข่ายด้วยตนเองหรือต้องส่งผ่านเราเตอร์ โดยสถานีต้นทางจะเปรียบเทียบเลขเครือข่ายของแอดเดรสต้นทางและปลายทางกับค่าซับเน็ตมาสก์ หากได้เลขเครือข่ายเหมือนกันแสดงว่าสถานีปลายทางอยู่ในเครือข่ายเดียวกัน สถานีต้นทางจะใช้เออาร์พีสอบถามฮาร์ดแวร์แอดเดรสหรืออ่านจากแคช และบรรจุอาร์ดแวร์แอดเดรสเข้าสู่เฟรมดาต้าลิงค์เพื่อส่งตรงไปยังสถานีปลายทาง แต่ถ้าเลขเครือข่ายมีค่าต่างกันแสดงว่าสถานีปลายทางอยู่ต่างเครือข่ายกัน สถานีส่งก็จะส่งเฟรมไปให้เราเตอร์เพื่อให้เราเตอร์นำส่งต่อไป เมื่อเราเตอร์ได้รับเฟรมนี้ก็จะส่งต่อไป ไอพีแอดเดรสต้นทางและปลายทางในไอพีดาต้าแกรมจะไม่เปลี่ยนแปลงค่าระหว่างการลำเลียง แต่ฮาร์ดแวร์แอดเดรสจะเปลี่ยนแปลงไปตามเครือข่าย

5.2 ตารางเส้นทาง(Routing Table)

โฮสต์และเราเตอร์จะเก็บแอดเดรสปลายทางสำหรับใช้เป็นเส้นทางส่งดาต้าแกรมไว้ในรูปตารางที่เรียกว่า ตารางเส้นทาง (Routing Table) ค่าในตารางเส้นทางมักประกอบด้วยไอพีแอดเดรสของเครือข่ายปลายทาง และเกตเวย์ซึ่งเป็นทางออกของดาต้าแกรม

ตารางเส้นทางจะบรรจุแอдресหนึ่งซึ่งทำหน้าที่เป็นช่องทางออกไปสู่เครือข่ายใด ๆ ที่ไม่ได้ระบุอยู่ในตารางเส้นทาง แอдресนี้เรียกว่า ดีฟอลต์เกตเวย์ (Default Gateway) หรือ ดีฟอลต์เราเตอร์ (Default Router) ซึ่งแทนด้วยแอдрес 0.0.0.0 ไอพีจะเลือกเส้นทางโดยตรวจสอบว่าแอдресปลายทางตรงกับรายการใดในตาราง แล้วส่งค่าแอดเรสไปยังเกตเวย์ของรายการนั้น หากไม่พบรายการใดในตาราง ก็จะส่งไปยังดีฟอลต์เกตเวย์

5.3 ประเภทของการเลือกเส้นทาง

เมื่อค่าแอดเรสเดินทางออกนอกเครือข่ายที่อาจต้องผ่านเราเตอร์จำนวนมากโดยมีเส้นทางได้หลายเส้นทาง แต่เราเตอร์จะทราบได้อย่างไรว่ามีเส้นทางใดบ้าง และเส้นทางใดที่เป็นเส้นทางที่ดีที่สุด

เราเตอร์จะส่งค่าแอดเรสได้จำเป็นต้องมีแผนที่เครือข่าย ซึ่งก็คือตารางเส้นทาง สำหรับวิธีที่ใช้ในการสร้างตารางเส้นทางนี้มี 2 แบบคือ

1. การเลือกเส้นทางแบบสถิติก(Static Routing) ตารางเลือกเส้นทางสร้างขึ้นและแก้ไขโดยผู้ดูแลระบบเครือข่าย
2. การเลือกเส้นทางแบบไดนามิก(Dynamic Routing) ใช้ซอฟต์แวร์คำนวณค่าตารางเลือกเส้นทาง ตารางสามารถปรับเปลี่ยนได้หากสภาพเครือข่ายเปลี่ยนไป

การเลือกเส้นทางแบบสถิติก

การเลือกเส้นทางแบบนี้ผู้ดูแลระบบเครือข่ายเป็นผู้พิจารณาและคำนวณหาเส้นทางทั้งหมด โดยใส่ค่าตารางเส้นทางให้กับเราเตอร์ทุกตัว ตารางเลือกเส้นทางนี้จะมีค่าตายตัวตลอดถึงแม้ว่าสภาพของเครือข่ายจะเปลี่ยนไปดังนั้นผู้ดูแลระบบจะต้องคอยตรวจสอบเครือข่ายและปรับเปลี่ยนตารางเส้นทางให้ถูก

รูปที่ 5-1 แสดงเครือข่ายที่ประกอบด้วยเราเตอร์ 2 ตัว เราเตอร์แต่ละตัวจะมีตารางเส้นทางที่ผู้ดูแลระบบป้อนเพื่อใช้กำหนดทิศทางการส่งค่าแอดเรส ทุกเครือข่ายที่เชื่อมโดยตรงกับเราเตอร์อื่นจะมีค่าเกตเวย์เท่ากับไอพีแอдресประจำอินเทอร์เฟซนั้น ส่วนเครือข่ายที่ต้องผ่านเราเตอร์อื่นจะมีค่าเกตเวย์เท่ากับไอพีแอдресของเราเตอร์ขั้นถัดไป (Next Hop Router) เช่นตารางเส้นทางในเราเตอร์ R1 มีค่าเกตเวย์ประจำเครือข่าย 1 และเครือข่าย 2 เท่ากับ 1.1 และ 2.1 ตามลำดับ ส่วนเกตเวย์สำหรับเครือข่าย 4 จะมีค่าเท่ากับ 3.2 ซึ่งเป็นแอдресประจำอินเทอร์เฟซ e1 ของ R2 (R2 เป็นเราเตอร์ขั้นถัดไปของ R1)

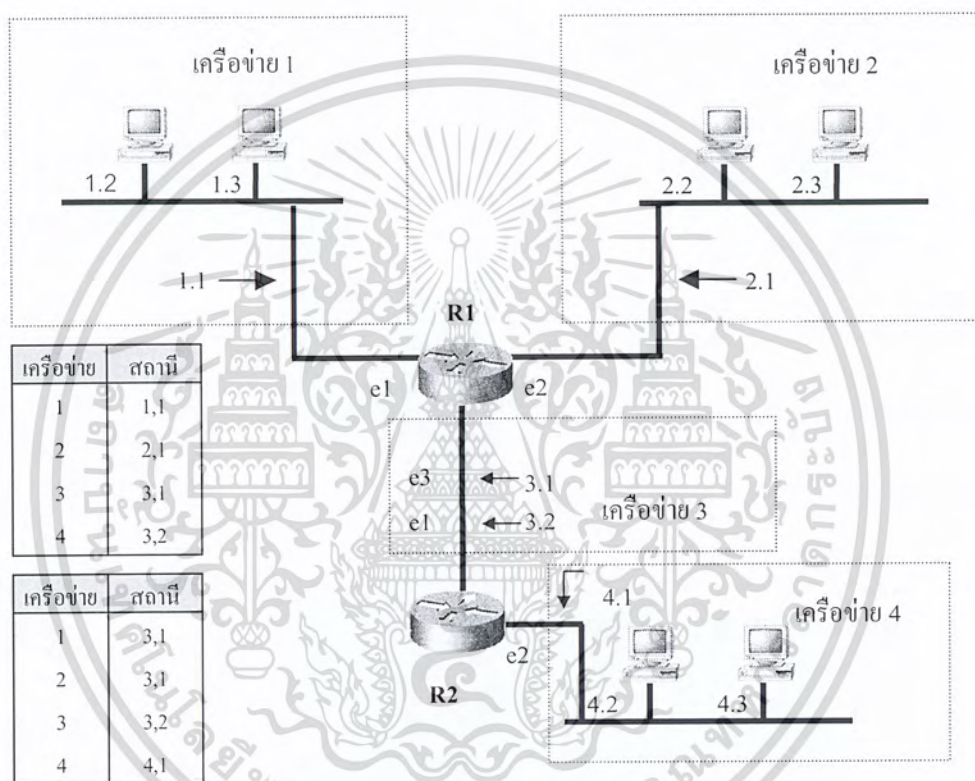
การเลือกเส้นทางแบบสถิติกนิยมใช้กับการเชื่อมโยงแบบจุดต่อจุดระหว่างเราเตอร์สองตัว เช่นเครือข่ายมีทางออกไปสู่ภายนอกหรืออินเทอร์เน็ตเพียงช่องทางเดียว มักจะกำหนดเส้นทางแบบสถิติก การเลือกเส้นทางแบบสถิติกมีข้อดีข้อเสียดังนี้

ข้อดี

1. สะดวกต่อการใช้งานกับเครือข่ายขนาดเล็ก
2. ไม่ต้องใช้ซอฟต์แวร์เลือกเส้นทาง เราเตอร์ไม่จำเป็นต้องมีซอฟต์แวร์สมรรถนะสูง
3. ประหยัดแบนด์วิดธ์เครือข่ายเนื่องจากไม่ต้องแลกเปลี่ยนข้อมูลระหว่างเราเตอร์
4. สามารถจำกัดให้เข้าถึงได้เฉพาะในเครือข่ายที่ต้องการ

ข้อเสีย

1. ไม่เหมาะกับเครือข่ายขนาดใหญ่ เพราะผู้ดูแลระบบสามารถคำนวณหาเส้นทางในเครือข่ายที่มีขนาดเล็กได้ แต่ในเครือข่ายขนาดใหญ่ที่มีเราเตอร์เป็นจำนวนมากการคำนวณและป้อนค่าเข้าสู่เราเตอร์โดยตรงเป็นสิ่งที่เกินวิสัย
2. ไม่สะดวกต่อการเปลี่ยนโครงสร้างของเครือข่าย เพราะผู้ดูแลต้องคำนวณหาเส้นทางใหม่ทุกครั้งที่มีการเปลี่ยนโครงสร้างของเครือข่าย
3. ตารางเส้นทางเป็นตารางคงตัวไม่สามารถเปลี่ยนแปลงได้เอง ถ้าเส้นทางใดถูกตัดขาดไป ผู้ดูแลระบบจะต้องคอยตรวจสอบและแก้ไขเอง



รูปที่ 5-1 เครือข่ายการแสดงการเลือกเส้นทางแบบสแตติก

การเลือกเส้นทางแบบไดนามิก

การเลือกเส้นทางแบบนี้จะใช้ซอฟต์แวร์ทำหน้าที่แลกเปลี่ยนข้อมูลการเลือกเส้นทางระหว่างเราเตอร์ด้วยกัน โดยใช้โพรโทคอลเลือกเส้นทาง เราเตอร์จะสร้างตารางเลือกเส้นทางจากสภาพเครือข่ายขณะนั้น หากเครือข่ายมีการเปลี่ยนแปลงตารางเส้นทางก็จะเปลี่ยนแปลงตามไปด้วย

การเลือกเส้นทางแบบไดนามิกต้องอาศัยการแลกเปลี่ยนค่าเส้นทางระหว่างเราเตอร์และใช้ซีพียูในเราเตอร์เพื่อสร้างตารางเส้นทาง เราเตอร์ประเภทนี้จึงมักมีราคาสูงกว่าเราเตอร์ที่มีโพรโทคอลแบบสแตติกอย่างเดียว เพราะต้องออกแบบซอฟต์แวร์ให้ทำงานตามโพรโทคอลเลือกเส้นทาง และซีพียูต้องมีประสิทธิภาพสูงพอในการคำนวณหาตารางเส้นทาง

ข้อดีของการการเลือกเส้นทางแบบไดนามิก

1. รองรับขนาดเครือข่ายที่ขยายขึ้นเป็นลำดับได้
2. ตารางเส้นทางเปลี่ยนแปลงค่าเองตามการทำงานของซอฟต์แวร์ เส้นทางใดที่ถูกตัดขาดจะมีการหาเส้นทางใหม่ทดแทน

ประเภทโพรโทคอลเลือกเส้นทางแบบไดนามิก

โพรโทคอลเลือกเส้นทางแบบไดนามิกสามารถจัดแยกประเภทออกได้หลายรูปแบบ ในที่นี้จะกล่าวเพียง 2 รูปแบบคือ

1. โพรโทคอลเกตเวย์ภายนอกและภายใน (Interior Gateway Protocol และ Exterior Gateway Protocol)
2. โพรโทคอลดิสเทนซ์เวกเตอร์และลิงก์สเตต (Distance Vector Protocol และ Link State Protocol)

โพรโทคอลเกตเวย์ภายนอกและภายใน

การจัดแบบนี้แบ่งชนิดโพรโทคอลออกเป็นโพรโทคอลเกตเวย์ภายนอกและโพรโทคอลเกตเวย์ภายใน โพรโทคอลเกตเวย์ภายนอกมีหน้าที่แลกเปลี่ยนข้อมูลเส้นทางระหว่างเครือข่ายที่มีการบริหารงานเป็นอิสระออกจากกัน โดยแต่ละเครือข่ายที่มีการบริหารเป็นอิสระออกจากกันจะเรียกว่า ระบบบอโตโนมัส (Autonomous System) แต่ละระบบบอโตโนมัสมีหมายเลขประจำของตนเองเรียกว่า เลขระบบบอโตโนมัส (Autonomous System Number) หมายเลขนี้สามารถขอได้จากหน่วยงานนิค (NIC) ประจำภูมิภาค เลขระบบบอโตโนมัสเป็นค่าที่ระบุข้อมูลเส้นทางที่แลกเปลี่ยนระหว่างเครือข่ายนั้นมาจากที่ใด

อินเทอร์เน็ตในยุคแรกใช้โพรโทคอล อีจีพี (EGP: Exterior Gateway Protocol) เป็นโพรโทคอลเกตเวย์ภายนอก แต่ในปัจจุบันโพรโทคอลที่นิยมใช้ระหว่างเครือข่ายคือบีจีพี (BGP : Border Gateway Protocol) และนำมาใช้งานแทนอีจีพี บีจีพีผ่านการพัฒนามาเป็นลำดับกระทั่งปัจจุบันเป็นรุ่นที่ 4 จึงเรียกว่า บีจีพี-4

โพรโทคอลเกตเวย์ภายในเป็นโพรโทคอลที่ออกแบบเพื่อใช้งานในระบบบอโตโนมัสหนึ่งๆ เช่น อาร์ไอพี (RIP : Routing Information Protocol) และ โอเอสพีเอฟ (OSPF: Open Shortest Path First) โพรโทคอลทั้งสองนี้เป็นที่ยอมรับเป็นมาตรฐานสากลและใช้งานอย่างแพร่หลายในเครือข่ายทั่วไป ในขณะที่โพรโทคอลเฉพาะที่ออกแบบโดยบริษัทซิสโก้ (Cisco) คืออีไอจีอาร์พี (EIGRP : Enhanced Interior Gateway Routing Protocol) เป็นอีกโพรโทคอลหนึ่งที่นิยมใช้ตามความแพร่หลายของเราเตอร์ของซิสโก้

เมตริก (Metric)

เมตริกเป็นค่าที่นำมาใช้คำนวณหาว่าเส้นทางใดเหมาะต่อการใช้มากกว่าเส้นทางอื่น ค่าเมตริกที่ใช้อาจเป็นได้ทั้งระยะทาง เวลาหน่วง ความน่าเชื่อถือ หรือความเร็ว โพรโทคอลที่ไม่สลับซับซ้อนอาจจะเลือกใช้เมตริกเพียงประเภทใดประเภทหนึ่ง เช่นอาจใช้เฉพาะระยะทางซึ่งนับจากจำนวนเราเตอร์ที่ต้องส่งผ่านหรือเรียกว่าจำนวนขั้น (Hop Count)

จำนวนขั้นนับจากจำนวนเราเตอร์ที่ค่าค่าแกรมต้องเดินผ่านดังตัวอย่างในรูปที่ 5-2 หากค่าค่าแกรมไม่ต้องเดินผ่านเราเตอร์จำนวนขั้นจะเท่ากับ 0 หากต้องข้ามเราเตอร์หนึ่งตัว จำนวนขั้นจะเพิ่มขึ้นทีละ 1 แต่โปรโตคอลบางชนิดอาจนับจำนวนขั้นตามจำนวนลิงค์ที่ค่าค่าแกรมเดินผ่านแทนการนับด้วยจำนวนเราเตอร์ หากเครือข่ายอยู่ติดกับเราเตอร์จะมีจำนวนขั้นเท่ากับ 1 แทนที่จะเท่ากับ 0

โปรโตคอลดิสเทนซ์เวกเตอร์และลิงก์สเทต

การจัดแบ่งโปรโตคอลในแบบนี้อาศัยรูปแบบของข้อมูลที่ส่งผ่านระหว่างเราเตอร์และวิธีที่เราเตอร์สร้างตารางเลือกเส้นทางจากข้อมูลนั้น

เราเตอร์ที่ใช้โปรโตคอลดิสเทนซ์เวกเตอร์อาศัยระยะทางเพื่อกำหนดว่าเส้นทางใดเหมาะสมกว่าเส้นทางอื่น ความหมายของดิสเทนซ์เวกเตอร์คือใช้ระยะทางเป็นค่าเมตริก และแอคเตสของเครือข่ายปลายทางเป็นเวกเตอร์กำหนดจุดหมายปลายทาง

การทำงานพื้นฐานของโปรโตคอลดิสเทนซ์เวกเตอร์จะส่งข้อมูลเลือกเส้นทางไปยังเราเตอร์ที่อยู่ข้างเคียงทุกตัวอย่างสม่ำเสมอเป็นช่วงเวลา ข้อมูลเลือกเส้นทางประกอบด้วยตารางเลือกเส้นทางของตนเองทั้งหมดที่กำกับด้วยเมตริก เราเตอร์แต่ละตัวจะใช้ตารางเส้นทางของตนเองร่วมกับตารางเส้นทางที่ได้รับมาใหม่เพื่อคำนวณหาระยะทางที่สั้นที่สุด โปรโตคอลดิสเทนซ์เวกเตอร์ที่นิยมใช้ในปัจจุบันได้แก่ อาร์ไอพี ค่าเมตริกที่ใช้คือจำนวนขั้น อาร์ไอพีถือว่าเส้นทางที่ดีที่สุดคือเส้นทางที่มีจำนวนขั้นน้อยที่สุด

โปรโตคอลลิงก์สเทตไม่ได้แลกเปลี่ยนตารางเส้นทางโดยตรงเหมือนกับที่ใช้ในดิสเทนซ์เวกเตอร์ หากแต่เราเตอร์แต่ละตัวจะตรวจสอบ สถานะลิงก์ (Link State) ที่เชื่อมไปยังเราเตอร์ข้างเคียงว่าใช้งานได้หรือไม่พร้อมกับค่าเมตริกซึ่งโดยทั่วไปแล้วเป็นความเร็วของสายสื่อสาร เช่นให้ค่า 1 สำหรับสายสื่อสารที่มีความเร็ว 2 เมกะบิตต่อวินาที และ 10 สำหรับสายสื่อสารที่มีความเร็ว 9600 บิตต่อวินาที โปรโตคอลลิงก์สเทตที่นิยมใช้คือ โอเอสพีเอฟ

5.4 อาร์ไอพี

อาร์ไอพีมีกำเนิดมาจากโปรโตคอลเลือกเส้นทางในระบบเครือข่ายซีร็อกซ์ซึ่งต่อมาได้พัฒนาเป็นอาร์ไอพีและโปรโตคอลอื่นได้แก่ไอพีเอกซ์ (โนเวลล์) และไอจีอาร์พี (ซิสโก้) อาร์ไอพีเป็นโปรโตคอลแบบดิสเทนซ์เวกเตอร์และใช้ระยะทางเป็นค่าเมตริกเพื่อหาเส้นทางที่ดีที่สุดสำหรับการเลือกเส้นทาง

อาร์ไอพีเป็นโปรโตคอลแบบดิสเทนซ์เวกเตอร์ ขั้นตอนพื้นฐานที่อาร์ไอพีใช้คือขั้นตอนวิธีของฟอร์ด/ฟูกอร์สัน หรือเรียกอีกชื่อว่าเบลแมน-ฟอร์ด เราเตอร์อาร์ไอพีจะเก็บตารางเส้นทางซึ่งประกอบด้วยแอคเตสเครือข่าย แอคเตสของเราเตอร์ถัดไปซึ่งเป็นเกตเวย์ไปยังเครือข่ายนั้น และเมตริกประจำเส้นทางซึ่งโดยปกติจะนับตามจำนวนเราเตอร์ระหว่างทาง

5.4.1 การทำงานของอาร์ไอพี

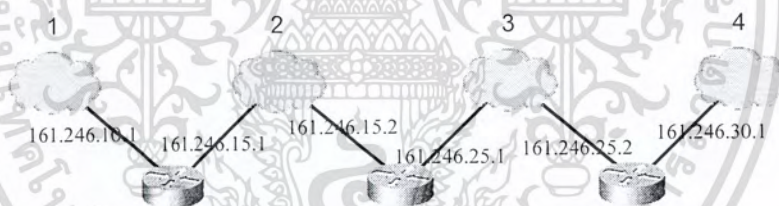
เริ่มแรกเราเตอร์แต่ละตัวจะได้รับการกำหนดแอดเดรสเครือข่ายประจำแต่ละอินเทอร์เฟซของเราเตอร์นั้น ซึ่งเราเตอร์ก็จะสร้างตารางเส้นทางจากแอดเดรสเครือข่ายที่มี แล้วส่งตารางเส้นทางของตัวเองไปให้เราเตอร์ข้างเคียงด้วยการบรอดคาสต์

เมื่อเราเตอร์ได้รับตารางเส้นทางจากเราเตอร์ตัวอื่น ก็จะปรับตารางเส้นทางของตัวเองพร้อมทั้งบรอดคาสต์ตารางของตัวเองไปให้เราเตอร์ข้างเคียงอีกเช่นกัน วิธีนี้ทำให้เราเตอร์แต่ละตัวสามารถคำนวณระยะทางและทิศทางของเครือข่ายอื่นที่เราเตอร์นั้นไม่ได้เชื่อมต่อได้ ในที่สุดเราเตอร์ทุกตัวก็จะทราบแอดเดรสทั้งหมดของเครือข่าย สำหรับการปรับค่าตารางเส้นทางเราเตอร์จะพิจารณาโดย

1. ถ้าเป็นเส้นทางใหม่ที่ไม่มีในตารางเส้นทาง จะใส่เส้นทางนั้นเข้าตารางเลย
2. ถ้าเป็นเส้นทางที่มีข้อมูลอยู่ในตารางเส้นทางแล้ว เราเตอร์จะพิจารณาว่าเป็นเส้นทางที่สั้นกว่าข้อมูลที่มีอยู่ในตาราง จะแทนที่ข้อมูลในตารางด้วยข้อมูลใหม่
3. ถ้าได้รับข้อมูลเส้นทางจากเราเตอร์ R ใดๆ และตรวจพบว่าในตารางมีเส้นทางซึ่งเราเตอร์ R เป็นเกตเวย์อยู่แล้ว ให้ปรับค่าเส้นทางใหม่ตามค่าที่ได้รับจากเราเตอร์ R นั้น

ตัวอย่างการสร้างตารางเส้นทาง

สมมติให้มีเครือข่ายดังรูป 5.2



รูปที่ 5-2 เครือข่ายสาธิตการทำงานของอาร์ไอพี

ในตอนแรกที่เราเตอร์ทั้ง 3 ตัวเริ่มทำงาน ค่าในตารางเส้นทางของเราเตอร์แต่ละตัวที่เกิดจากการติดตั้งค่าของผู้ดูแลเครือข่ายจะเป็นเครือข่ายที่เชื่อมต่อกับเราเตอร์นั้น โดยตรงดังนี้

ตาราง R1

เครือข่าย	เกตเวย์	อินเทอร์เฟซ	เมตริก
161.246.10.0	0.0.0.0	e1	1
161.246.15.0	0.0.0.0	e2	1

ตาราง R2

เครือข่าย	เกตเวย์	อินเทอร์เน็ตเฟส	เมตริก
161.246.15.0	0.0.0.0	e1	1
161.246.25.0	0.0.0.0	e2	1

ตาราง R3

เครือข่าย	เกตเวย์	อินเทอร์เน็ตเฟส	เมตริก
161.246.25.0	0.0.0.0	e1	1
161.246.30.0	0.0.0.0	e2	1

ตารางเลือกเส้นทางตามตัวอย่างข้างต้นแสดงเพียงฟิลด์สำคัญซึ่งประกอบด้วย

เครือข่าย : ไอพีแอดเดรสของเครือข่ายปลายทาง

เกตเวย์ : ไอพีแอดเดรสของเราเตอร์ซึ่งเป็นทางออกไปสู่เครือข่ายปลายทาง

อินเทอร์เน็ตเฟส : อินเทอร์เน็ตเฟสของเราเตอร์

เมตริก : จำนวนชั้น

ในตอนแรกเราเตอร์จะมีตารางเส้นทางเฉพาะเครือข่ายที่อยู่ติดกับเราเตอร์ แต่เส้นทางไปเครือข่ายอื่นนั้นจะได้อาจจากการแลกเปลี่ยนตารางเส้นทางกับเราเตอร์ตัวอื่น ซึ่งอาร์ไอพีนี้กำหนดให้เราเตอร์ประกาศเส้นทางให้กับเราเตอร์ข้างเคียงทุกๆ 30 วินาที หากสมมติให้ R1 บรอดแคสต์ R2 จะได้รับแอดเดรส 161.246.10.0 และ 161.246.15.0 จาก R1

เครือข่าย 161.246.15.0 เป็นค่าที่มีอยู่ในตารางแล้วและเป็นเครือข่ายที่อยู่ติดกับเราเตอร์ R2 จึงไม่เปลี่ยนค่า ส่วนเครือข่าย 161.246.10.0 เป็นค่าใหม่ R2 จะเพิ่มค่าเข้าไปในตารางโดยมองว่า “เราเตอร์ R2 สามารถไปถึงเครือข่าย 161.246.10.0 ได้โดยผ่านเราเตอร์ R1 ซึ่งถ้าเราเตอร์ R1 สามารถไปยังเครือข่าย 161.246.10.0 ได้ด้วยระยะทาง 1 จำนวนชั้น ดังนั้นถ้าเราเตอร์ R2 สามารถไปเครือข่ายนั้นโดยผ่านเราเตอร์ R1 ก็จะสามารถไปถึงได้ด้วยระยะทางที่เราเตอร์ R1 ไป + 1 จึงเท่ากับ $1+1 = 2$ จำนวนชั้น” และใช้เกตเวย์ 161.246.15.1 เป็นทางออก ตาราง ของ R2 หลังจากได้รับตารางเส้นทางที่ R1 บรอดแคสต์ให้จะเป็น

เครือข่าย	เกตเวย์	อินเทอร์เน็ตเฟส	เมตริกซ์
161.246.10.0	161.246.15.1	e1	2
161.246.15.0	0.0.0.0	e1	1
161.246.25.0	0.0.0.0	e1	1

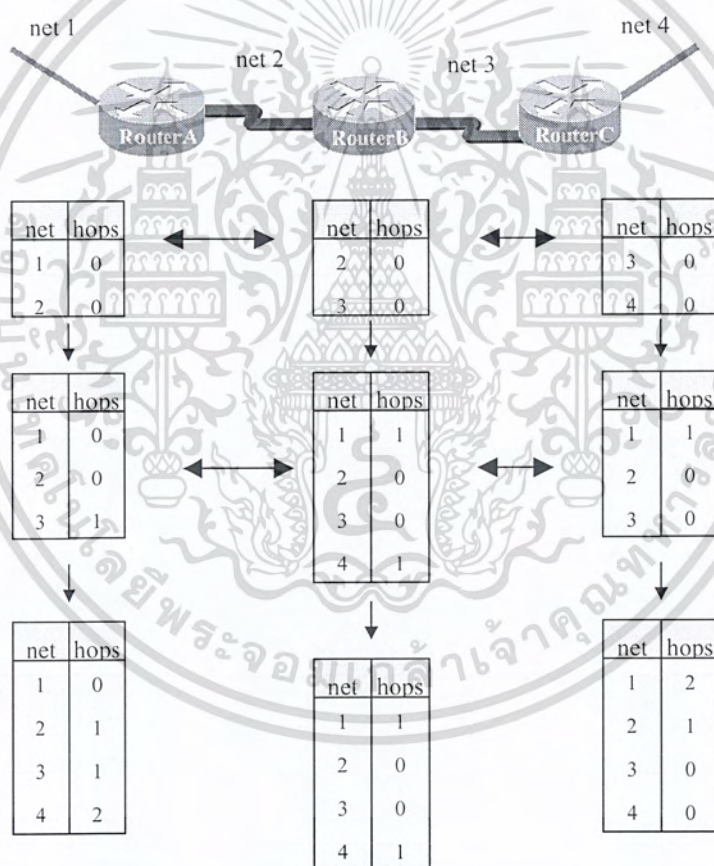
← ค่าที่ได้ใหม่

ถ้าต่อมา R3 บรอดแคสต์เฉพาะเราเตอร์ R2 ก็จะได้รับค่าและเปลี่ยนค่าในตารางเช่นเดียวกัน ดังนั้นตารางเส้นทางของเราเตอร์ R2 จะเป็น

เครือข่าย	เกตเวย์	อินเทอร์เฟซ	เมตริก
161.246.10.0	161.246.15.1	e1	2
161.246.15.0	0.0.0.0	e1	1
161.246.25.0	0.0.0.0	e2	1
161.246.30.0	161.246.25.2	e2	2

ค่าที่ได้ใหม่

ตารางของ R2 ในตอนแรกมีเพียงแค่เครือข่ายที่เชื่อมต่อโดยตรงกับเราเตอร์นั้น หลังจากที่ได้รับ broadcast จาก R1 และ R3 ตารางของ R2 ก็จะมีข้อมูลของเครือข่ายครบทุกเครือข่าย รูปที่ 5-3 แสดงการประกาศและปรับค่าตารางเส้นทางของเราเตอร์ทั้งหมด โดยสมมติให้แต่ละเราเตอร์ประกาศค่าตารางพร้อมกัน



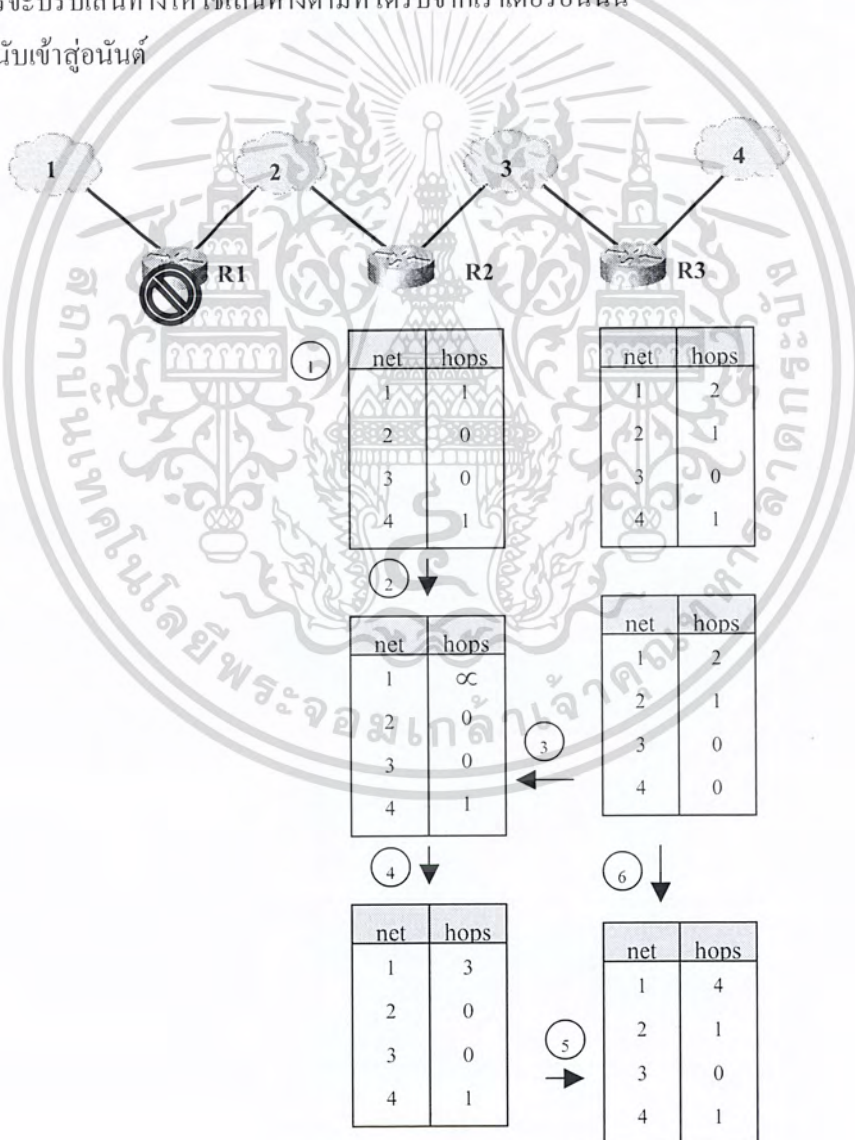
รูปที่ 5-3 การประกาศค่าและปรับค่าและตารางของอาร์ไอพี

5.4.2 การปรับค่าเมื่อเครือข่ายเปลี่ยน

กรณีที่โครงสร้างของเครือข่ายไม่เปลี่ยนแปลง ตารางเส้นทางของอาร์โอพีก็จะมีเสถียรภาพถึงแม้ว่าเราเตอร์จะประกาศตารางออกไปทุกๆ 30 วินาที จึงเป็นการตรวจสอบสภาพเส้นทางและพร้อมที่จะปรับตารางเส้นทางให้เหมาะสมกับสภาพของเครือข่ายถ้าเส้นทางเดิมนั้นมีปัญหา

เมื่อพิจารณาในทางปฏิบัติแล้วเราเตอร์อาร์โอพีใดๆ จะไม่สามารถปรับตารางเส้นทางใหม่ได้หากเราเตอร์ข้างเคียงไม่ประกาศค่าเส้นทางให้ ซึ่งอาจเป็นเพราะเราเตอร์ข้างเคียงตัวนั้นไม่สามารถทำงานได้หรือหยุดทำงานชั่วคราว อาร์โอพีจึงหาวิธีแก้ปัญหานี้โดยการกำหนดให้เส้นทางทุกเส้นทางมีอายุการใช้งานปกติจะกำหนดไว้ที่ 180 วินาที ถ้าเส้นทางใดไม่ได้รับการประกาศค่าเส้นทางใดๆ เป็นเวลาตามที่กำหนดไว้ จะถือว่าเส้นทางนั้นเป็นเส้นทางที่ใช้ไม่ได้อีกต่อไป และเราเตอร์จะเปลี่ยนเมตริกของเส้นทางนั้นให้เป็นอนันต์ แต่จะยังไม่ลบเส้นทางนั้นออกจากตาราง หากมีการประกาศเส้นทางนั้นมาจากเราเตอร์อื่น เราเตอร์จะปรับเส้นทางให้ใช้เส้นทางตามที่ได้รับจากเราเตอร์อื่นนั้น

ปัญหาการนับเข้าสู่อนันต์



รูป 5.4 การนับเข้าสู่อนันต์

การทำงานของอาร์โอทีอาจมีปัญหาเวลาที่เรเตอร์ R พบเส้นทางไปยังเครือข่ายหนึ่งผิดปกติ ในขณะที่เรเตอร์ข้างเคียงตัวหนึ่งแจ้งว่ามีเส้นทางไปยังเครือข่ายที่เกิดผิดปกติ เรเตอร์ R ที่พบปัญหาก็จะปรับค่าเส้นทางตามค่าที่ได้รับมาใหม่ แต่เนื่องจากเป็นเส้นทางที่เรเตอร์ข้างเคียงสามารถไปถึงเครือข่ายนั้นได้โดยผ่านทางเรเตอร์ R ทำให้เกิดการปรับเมตริกต่อไปแบบไม่มีที่สิ้นสุด ปรากฏการณ์นี้เราเรียกว่า การนับเข้าสู่อนันต์ (Count to Infinity)

ให้ตารางเส้นทางที่เรเตอร์ R2 และ R3 มีค่าดังตำแหน่ง ในจังหวะที่เรเตอร์ R1 ทำงานผิดปกติ ทำให้เครือข่าย 1 จะถูกตัดขาดออกไป ค่าเส้นทางของเครือข่าย 1 ใน R2 จะลดลงตามการทำงานของตัวจับเวลา เมื่อเมื่อ 180 วินาที เส้นทางไปเครือข่าย 1 จะเปลี่ยนเป็นอนันต์ดังตำแหน่ง

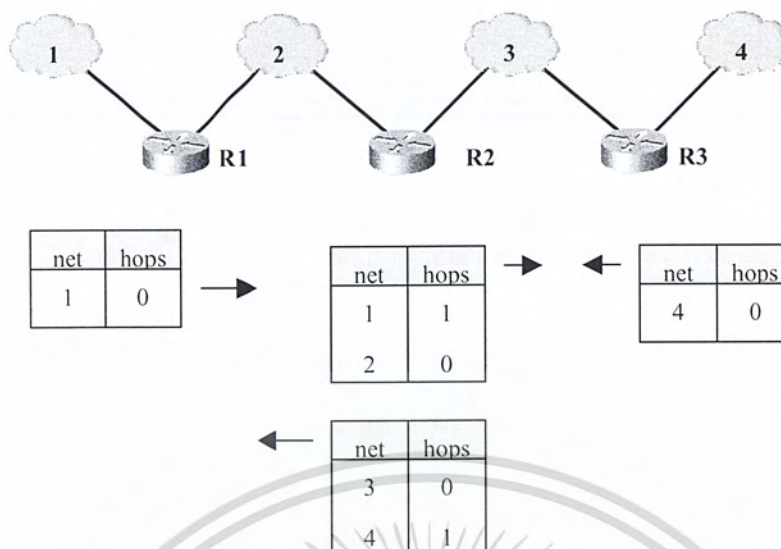
หากเวลานั้นถึงเวลาที่ R3 บรอดแคสต์ R3 จะส่งค่าเส้นทางมาให้ R2 ทำให้ R2 ได้รับเส้นทางไปเครือข่าย 1 มาด้วยเมตริก 2 ดังนั้น R2 จึงเปลี่ยนค่าในตารางเส้นทางให้เส้นทางไปเครือข่าย 1 สามารถไปโดยผ่านทาง R3 ด้วยเมตริก 3 แต่ที่จริงแล้วเครือข่าย 1 นั้นไม่สามารถใช้งานได้อยู่ในขณะนั้น R2 และ R3 ก็ไม่ทราบได้โดยตรง ทำให้เมื่อถึงเวลา R2 และ R3 ประกาศตารางเส้นทางตารางของทั้งสองจะปรับค่าเส้นทางเพื่อไปยังเครือข่าย 1 อย่างไม่มีที่สิ้นสุด ซึ่งอาร์โอทีได้กำหนดค่าเมตริกไว้ค่าหนึ่งเพื่อแก้ปัญหานี้ ค่าเมตริกนี้ควรเป็นค่าที่ไม่สูงมากเพราะจะได้ไม่ต้องใช้เวลานาน แต่ถ้าน้อยเกินไปก็อาจทำให้เครือข่ายถูกจำกัดให้เล็กลง ซึ่งปกติจะกำหนดไว้ที่ 16 เมื่อเรเตอร์เพิ่มค่าไปถึง 16 ก็จะทราบว่าเส้นทางนั้นเป็นเส้นทางที่ไม่สามารถไปถึงได้ และจะเข้าสู่กระบวนการจับเวลาเพื่อกำจัดเส้นทางออกจากตารางเวลาลู่เข้า

เวลาลู่เข้าของอาร์โอทีหมายถึงเวลาที่เรเตอร์ใช้ปรับตารางจนกระทั่งตารางมีค่าถูกต้องตามโครงสร้างของเครือข่ายจริง จากข้อกำหนดของการประกาศค่าของเรเตอร์อาร์โอทีทุกๆ 30 วินาที ปัญหาการนับเข้าสู่อนันต์ในเครือข่ายที่ใช้อาร์โอทีจะใช้เวลาสูงสุดประมาณ 7 นาที

หากเวลาลู่เข้าของอาร์โอทีสั้นลง เครือข่ายก็ย่อมที่จะเข้าสู่เสถียรภาพได้เร็วขึ้น วิธีปรับเวลาลู่เข้าของอาร์โอทีให้สั้นลงมีหลายวิธี ดังนี้

สปลิตซอไรซัน(Split Horizon)

วิธีนี้จะช่วยแก้ปัญหานับเข้าสู่อนันต์ (เฉพาะเครือข่ายบางรูปแบบเท่านั้น) โดยเรเตอร์จะไม่ประกาศตารางเส้นทางทั้งตารางให้กับเรเตอร์ข้างเคียงแต่จะประกาศเฉพาะเส้นทางที่ไม่ได้เรียนรู้มาจากเรเตอร์ตัวนั้นเท่านั้น เช่นเรเตอร์ R2 ได้รับตารางเส้นทางไปเครือข่าย 1 มาจากเรเตอร์ R1 เรเตอร์ R2 ก็จะประกาศเฉพาะเส้นทางของเครือข่าย 3 และ 4 ไปให้กับ R1 ดังรูป



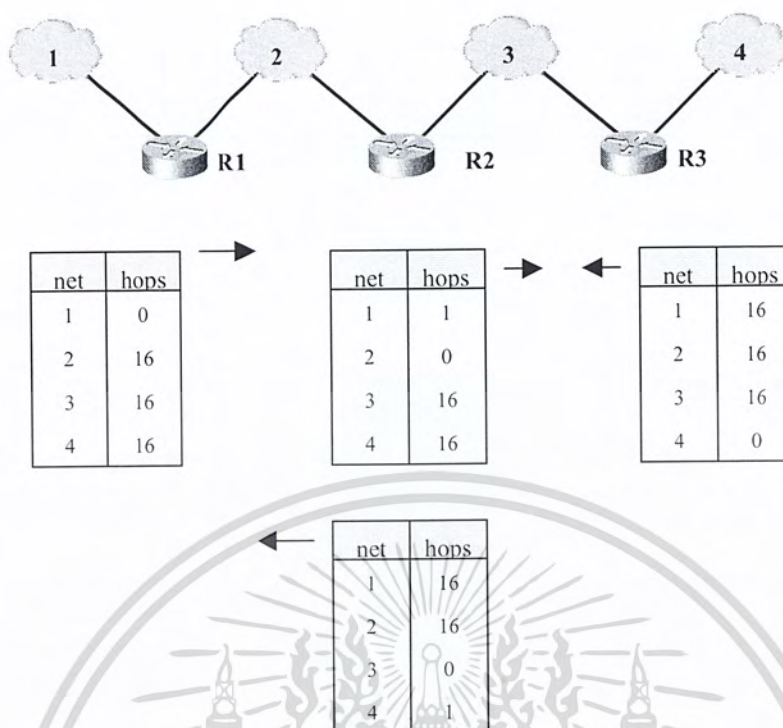
รูปที่ 5-5 การประกาศเส้นทางด้วยวิธีสปลิตฮอไรซัน

หากเราเตอร์ R1 หยุดทำงานไป ค่าเส้นทางของเครือข่าย 1 ใน R2 จะค่อยๆ ลดลงจนครบ 180 วินาที เส้นทางไปเครือข่าย 1 จะเปลี่ยนเป็นอนันต์ โดยที่ R2 จะไม่ได้รับการประกาศเส้นทางไปเครือข่าย 1 จาก R3 เนื่องจากเป็นเส้นทางที่ R2 ประกาศให้ R3 รู้วิธีไปยังเครือข่าย 1 เอง ดังนั้นจึงไม่เกิดปัญหานับเข้าสู่อนันต์

สปลิตฮอไรซันแบบพอยซันรีเวอร์ส (Split Horizon with Poisoned Reverse)

วิธีนี้เป็นเทคนิคอีกแบบหนึ่งของสปลิตฮอไรซันที่ใช้วิธีแจ้งค่าเส้นทางที่ชัดเจนมากยิ่งขึ้น เพื่อไม่ให้เกิดการนับเข้าสู่อนันต์

วิธีนี้จะยอมให้เราเตอร์ประกาศเส้นทางที่เรียนรู้จากลิงก์อื่นๆ กลับไปได้ แต่ให้เมตริกประจำเส้นทางนั้นเป็นอนันต์หรือเท่ากับ 16 เพื่อกำกับว่าเส้นทางนั้นไม่สามารถใช้งานได้ เช่น มีเราเตอร์ต่อกัน ดังรูป



รูปที่ 5-6 การประกาศเส้นทางด้วยวิธีสปลิตฮอไรซันแบบพอยซ์รีเวอร์ส

หากเราเตอร์ R1 หยุดทำงาน ค่าเส้นทางของเครือข่าย 1 ใน R2 จะลดลงตามเวลาการทำงานของตัวจับเวลา และเมื่อครบ 180 วินาที เส้นทางไปเครือข่าย 1 จะเปลี่ยนเป็นอนันต์ เมื่อทั้ง R2 และ R3 ต่างประกาศค่าซึ่งแจ้งว่าระยะทางไปเครือข่าย 1 มีค่าเท่ากับ 16 ส่วนทางกันก็จะทราบได้ว่าเครือข่าย 1 ใช้งานไม่ได้ ทั้ง R2 และ R3 ก็จะกำจัดเส้นทางไปเครือข่าย 1 โดยไม่เกิดปัญหานับเข้าสู่อนันต์

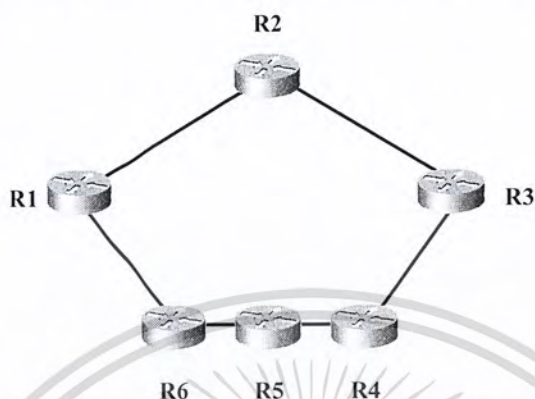
วิธีนี้จะสร้างแพ็กเก็ตมากกว่าวิธีสปลิตฮอไรซันแบบธรรมดา เนื่องจากมีค่าที่ต้องประกาศมากกว่า แต่ก็เป็วิธีที่เรเตอร์ในปัจจุบันนิยมใช้เนื่องจากให้ประสิทธิผลดีกว่า

เทคนิคของสปลิตฮอไรซันทั้งสองแบบนี้ไม่สามารถป้องกันปัญหานับเข้าสู่อนันต์ได้ทุกรูปแบบ เครือข่าย ช่วยแก้ปัญหาสำหรับเรเตอร์สองตัวที่อยู่ข้างเคียงกันเท่านั้น

ทริกเกอร์อัปเดต

สปลิตฮอไรซันทั้ง 2 วิธีนั้นช่วยแก้ปัญหาการนับเวลาเข้าสู่อนันต์เพื่อลดเวลาเข้าสู่ หากแต่ยังต้องใช้เวลาคำนวณหนึ่งก่อนที่จะตรวจสอบว่าเส้นทางหนึ่งๆ มีปัญหา ทริกเกอร์อัปเดตเป็นเทคนิคเสริมที่ช่วยลดเวลาเข้าสู่ของอาร์ไอพี คือกำหนดให้เรเตอร์ประกาศค่าเส้นทางออกไปทันทีที่พบว่าโครงสร้างเครือข่ายมีการเปลี่ยนแปลงโดยไม่ต้องรอให้ครบเวลา 30 วินาที

หากเราเตอร์ทุกตัวใช้ทริกเกอร์อัปเดตก็จะสามารถปรับปรุงตารางได้อย่างรวดเร็ว แต่ก็อาจจะสร้างภาระในเครือข่ายขนาดใหญ่ที่มีเราเตอร์หลายตัว เพราะการเปลี่ยนแปลงใดๆ ก็ย่อมทำให้เราเตอร์ตัวหนึ่งส่งข้อมูลให้เราเตอร์ตัวอื่นประกาศค่าตามกันไปอย่างต่อเนื่องหรือเกิดแพ็กเก็ตบรอดแคสต์เป็นจำนวนมาก แต่เทคนิคนี้อาจทำให้เกิดเหตุการณ์ที่ไม่คาดฝันขึ้นได้ เช่นในกรณีที่มีเครือข่ายดังรูป



รูปที่ 5-7 เครือข่ายที่จำเป็นต้องใช้โฮลดาวน์ร่วมกับทริกเกอร์อัปเดต

ถ้าเส้นทางที่เชื่อมระหว่าง R3 กับ R4 ไม่สามารถใช้งานได้และใช้เทคนิคทริกเกอร์อัปเดตแจ้งไปยังเราเตอร์ที่อยู่ข้างเคียง ปัญหาจะเกิดขึ้นเมื่อ R1 ได้รับการแจ้งค่าจาก R2 ก่อน R6 ทำให้ R1 ปรับตารางว่าเส้นทางที่เชื่อมระหว่าง R3 กับ R4 ใช้งานไม่ได้

ต่อมาก่อนที่ R6 จะได้รับแจ้งจาก R5 ถ้าถึงเวลาประกาศเส้นทางออกไป ทำให้ R1 ได้รับแจ้งว่ามีเส้นทางระหว่าง R3 กับ R4 ใช้งานได้ผ่าน R6 ทำให้ R1 ปรับตารางเส้นทางกลับไปใหม่ว่าเส้นทางระหว่าง R3 กับ R4 ใช้งานได้ ทำให้กว่าที่เราเตอร์ทุกตัวจะรู้ว่าเส้นทางระหว่าง R3 กับ R4 ไม่สามารถใช้งานได้ ก็ใช้เวลานานพอสมควร เพื่อแก้ปัญหาเรื่องนี้จึงใช้กฎเพิ่มเติมสำหรับการปรับตารางที่เรียกว่าโฮลดาวน์ (Hold Down)

โฮลดาวน์กำหนดช่วงเวลาให้เราเตอร์ปรับเปลี่ยนค่าที่เพิ่งจะตรวจพบว่าไม่สามารถใช้งานได้ จากตัวอย่างข้างต้นนี้ เมื่อ R1 ปรับเส้นทางระหว่าง R3 กับ R4 ว่าไม่สามารถใช้งานได้กฎโฮลดาวน์จะบังคับให้ R1 รอเวลาโดยไม่รับการแจ้งว่าเส้นทางระหว่าง R3 กับ R4 สามารถใช้งานได้จากเราเตอร์อื่นเป็นเวลาหนึ่ง โดยช่วงเวลานี้ควรเป็นช่วงเวลาที่มากพอจะให้ทริกเกอร์อัปเดตกระจายไปยังเราเตอร์ทุกตัว ซึ่งการใช้โฮลดาวน์ถึงแม้ว่าจะทำให้ช่วงเวลาลู่อ่านนานกว่าแบบไม่ใช่ แต่ทำให้การทำงานของระบบดีกว่าแบบไม่ใช่ด้วย

ตัวจับเวลาในอาร์ไอพี

ตัวจับเวลาในอาร์ไอพีจะมีอยู่ 3 ประเภทคือ

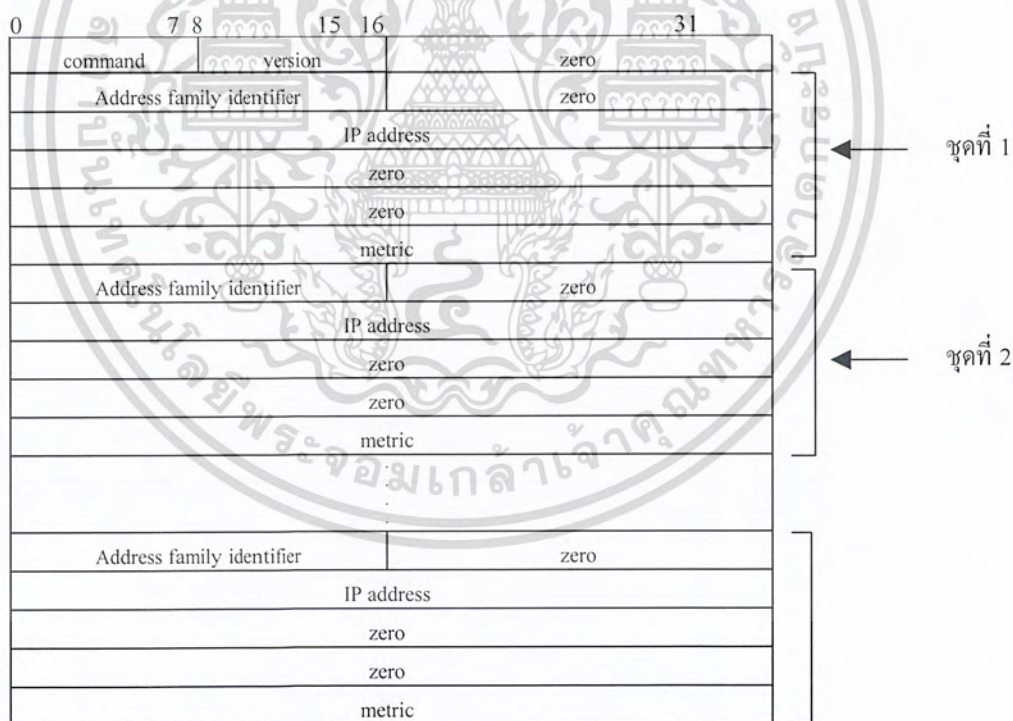
1. ตัวจับเวลาปรับค่า (Update Timer) เป็นช่วงเวลาที่กำหนดการบรอดคาสต์ของเราเตอร์ ซึ่งปกติจะให้บรอดคาสต์ตารางทุกๆ 30 วินาที แต่ในทางปฏิบัติเราไม่สามารถหลีกเลี่ยงไม่ให้เกิดการบรอดแคสต์พร้อมกัน เช่น ในเราเตอร์ซิสโก้จะสุ่มค่า RIP_JITTER ซึ่งมีค่าได้ไม่เกิน 4.5

วินาทีแล้วนำมาลบกับค่า 30 ตัวจับเวลาปรับค่าของเราเตอร์ซิสโก้จึงอยู่ระหว่าง 25.5 ถึง 30 วินาที วิธีนี้จึงช่วยลดโอกาสเกิดการบรอดแคสต์แพ็กเก็ตพร้อมกันเป็นจำนวนมากลงไป นอกจากนี้เพื่อป้องกันการกลับมาเข้าจังหวะของเราเตอร์เมื่อเกิดทริกเกอร์อัปเดต เราเตอร์จะต้องไม่รีเซ็ตตัวจับเวลาให้กลับมาเป็นศูนย์เมื่อเกิดทริกเกอร์อัปเดต แต่ให้จับเวลาต่อไปตามปกติและบรอดคาสต์ออกไปเมื่อถึงเวลา

2. ตัวจับเวลาหมดอายุ (Expiration Timer) เป็นตัวกำหนดอายุของแต่ละเส้นทาง โดยถ้าเส้นทางใดไม่ได้รับประกาศเส้นทางนั้นมาจากเราเตอร์อื่นเป็นเวลาเท่าที่กำหนดนี้ ค่าเมตริกของเส้นทางนั้นก็จะถูกเปลี่ยนเป็น 16 เพื่อแสดงว่าไม่สามารถไปถึงได้ โดยปกติจะกำหนดเวลานี้เป็น 180 วินาที
3. ตัวจับเวลากำจัดเส้นทาง (Garbage Collection Timer) สำหรับเส้นทางที่หมดอายุจากการจับเวลา 180 วินาทีข้างต้นแล้วจะยังไม่ถูกลบออกจากตารางทันที แต่จะนับเวลาถอยหลังเป็นจำนวนเวลาตามที่กำหนดนี้เพื่อลบเส้นทางนี้ออกไปจากตาราง แต่ในระหว่างที่นับเวลาถอยหลังนี้ก็จะบรอดแคสต์เส้นทางนี้ออกไปด้วย เพื่อให้เราเตอร์ตัวอื่นนับเวลาถอยหลังเพื่อลบเส้นทางนี้ด้วย

เฟรมอาร์ไอพี

เฟรมอาร์ไอพีจะนำส่งโดยบรรจุอยู่ในยูดีพี ลักษณะของเฟรมอาร์ไอพีจะเป็นดังรูป



รูปที่ 5-8 ฟอรัมเมตของเฟรมอาร์ไอพี

รูปแบบเฟรมในที่นี้แสดงเฉพาะการใช้อาร์ไอพีในเครือข่ายไอพีเท่านั้น หากเป็นเครือข่ายอื่นจะมีรูปแบบแตกต่างออกไป โดยแต่ละฟิลด์มีความหมายดังนี้

- command ขนาด 8 บิต : กำหนดแบบการทำงานว่าเป็นการร้องขอ (ค่าเท่ากับ 1) หรือการตอบรับ (ค่าเท่ากับ 2) การบรอดคาสต์ของอาร์ไอพีจัดเป็นการตอบรับ (เพราะไม่มีการร้องขอมาก่อน) อาร์ไอพีจะกำหนดค่าส่งให้เป็น 1 เพื่อร้องขอต่อเมื่อต้องการทราบตารางเส้นทางของเราเตอร์ใดเป็นกรณีเฉพาะ
- version ขนาด 8 บิต : กำหนดรุ่นของโพรโตคอล หากเท่ากับ 1 หมายถึงรุ่น 1
- zero ขนาด 16 บิต : สงวนไว้และต้องมามีค่าเป็น 0
- address family identifier ขนาด 16 บิต : การออกแบบอาร์ไอพีในขั้นต้นไม่ได้กำหนดให้ใช้เฉพาะกับที่ซีพี/ไอพีเท่านั้น จึงมีฟิลด์กำกับชุดแอดเดรสเพื่อแยกแยะชนิดโพรโตคอลที่อาร์ไอพีทำงานด้วย ค่านี้เท่ากับ 2 สำหรับไอพี
- IP address ขนาด 32 บิต : ไอพีแอดเดรสของเครือข่ายที่ประกาศค่าออกไป
- Metric ขนาด 32 บิต : เมตริกกำหนดระยะทางให้ใช้ค่าได้ตั้งแต่ 1 ถึง 15 หากมีค่าเป็น 16 หมายถึงเครือข่ายปลายทางไม่สามารถไปถึงได้

ตั้งแต่ฟิลด์ address family identifier กระทั่งถึงเมตริกสามารถมีค่าซ้ำได้ 25 ชุด เพื่อให้เราเตอร์สามารถบรอดคาสต์เส้นทางได้ 25 ค่าในค่าเดียว

เฟรมอาร์ไอพี 2

อาร์ไอพีรุ่น 2 มีส่วนขยายการทำงานเพิ่มเติมจากอาร์ไอพีรุ่นแรกและเพิ่มระบบความปลอดภัยในการส่งข้อมูลเส้นทางระหว่างเราเตอร์ อาร์ไอพีรุ่น 1 ไม่สนับสนุนการใช้งานในเครือข่ายที่มีซับเน็ตแบบแปรค่า ไม่สามารถแจ้งหมายเลขระบบบอโคโนมัสเพื่อเชื่อมความปลอดภัยในการส่งตาราง อาร์ไอพีรุ่น 2 จึงได้รับการพัฒนาขึ้นเพื่อรักษาความปลอดภัยในการส่งตาราง อาร์ไอพีรุ่น 2 จึงได้รับการพัฒนาขึ้นเพื่อแก้ปัญหาดังกล่าวและกลายเป็นมาตรฐานใหม่ทดแทนอาร์ไอพีรุ่น 1

ฟิลด์อาร์ไอพีรุ่น 2 ยังคงลักษณะของอาร์ไอพีรุ่น 1 แต่ได้เพิ่มเติมฟิลด์ใหม่ และมีรูปแบบดังรูป

5.9

ความหมายของแต่ละฟิลด์

Command ขนาด 8 บิต : กำหนดแบบการทำงานว่าเป็นการร้องขอ (ค่าเท่ากับ 1) หรือเป็นการตอบรับ (ค่าเท่ากับ 2)

Version ขนาด 16 บิต : กำหนดรุ่นของโพรโตคอล ค่าที่ใช้คือ 2

Unused ขนาด 16 บิต : ไม่ได้ใช้งาน

Address family identifier ขนาด 16 บิต : ฟิลด์กำกับชุดแอดเดรสเพื่อแยกแยะชนิดของโพรโตคอลที่อาร์ไอพีทำงานร่วมกับ สำหรับไอพีใช้ค่าเท่ากับ 2

Route tag ขนาด 8 บิต : ใช้แยกความแตกต่างระหว่างการได้มาของค่าเส้นทางว่าเรียนรู้จากภายในหรือภายนอก ค่าที่ใช้อาจเป็นเลขระบบบอโดโนมัสสำหรับเส้นทางที่ได้มาจากโพรโตคอลภายนอก

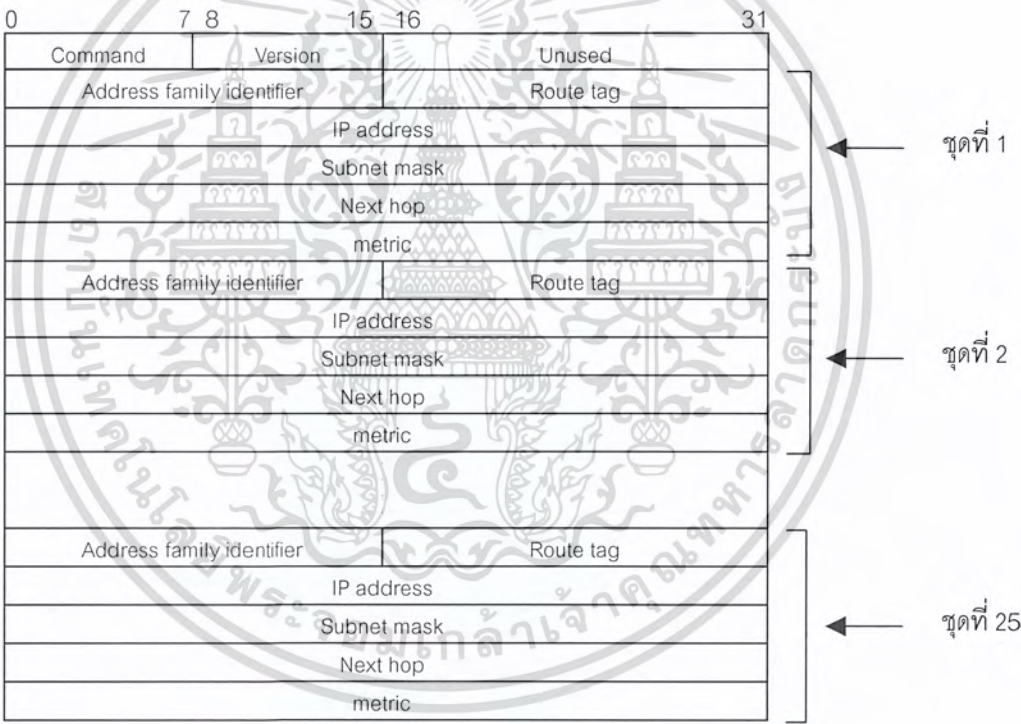
IP Address ขนาด 32 บิต : ไอพีแอดเดรสที่กำหนดเส้นทาง

Subnet Mask ขนาด 32 บิต : ซับเน็ตมาสก์กำกับไอพีแอดเดรส หากมีค่าเป็น 0 หมายถึงไม่ได้กำหนดซับเน็ตมาสก์ให้

Next Hop ขนาด 32 บิต : ไอพีแอดเดรสของเราเตอร์ถัดไปที่ควรจะนำส่งแพ็กเก็ตซึ่งมีแอดเดรสปลายทางตามที่กำหนดด้วยค่าเลือกเส้นทางในเฟรมอาร์ไอพีนี้ ประโยชน์ของฟิลด์นี้ใช้เพื่อลดจำนวนเส้นทางที่แพ็กเก็ตต้องเดินทางผ่านในกรณีที่มีโพรโตคอลเลือกเส้นทางอื่นนอกเหนือไปจากอาร์ไอพีทำงานร่วมกัน หากมีค่า 0.0.0.0 หมายถึงให้ส่งแพ็กเก็ตไปยังเราเตอร์ที่ประกาศค่าเส้นทางนี้มาให้

Metric ขนาด 32 บิต : เมตริกกำหนดระยะทางยังคงมีค่าได้ตั้งแต่ 1 ถึง 15 เช่นเดียวกับอาร์ไอพีรุ่น 1

การพิสูจน์ตัวจริง



รูปที่ 5-9 ฟอร์เมตของเฟรมอาร์ไอพีเวอร์ชัน 2

ตั้งแต่ฟิลด์ address family identifier ถึงฟิลด์ metric มีค่าซ้ำกันได้ 25 ชุดเช่นเดียวกับเวอร์ชันแรก เพื่อการประกาศเส้นทางไปพร้อมกันได้ 25 เส้นทางในเฟรมเดียว แต่ถ้าฟิลด์ address family identifier ชุดแรกมีค่าเป็น 0xFFFF (ให้ค่านี้มีได้เฉพาะในชุดแรกเท่านั้น) หมายความว่าเฟรมกำหนดการพิสูจน์ตัวจริง และฟิลด์ถัดมาจะถูกแปลความหมายใหม่ตามรูป 5.10

มีฟิลด์ authentication type ขนาด 16 บิตอยู่ต่อจาก address family identifier และถัดไปอีก 128 บิตเป็นฟิลด์ authentication สำหรับใช้ตรวจสอบว่าเราเตอร์ที่ส่งเฟรมนี้เป็นเราเตอร์ตัวจริงที่ได้รับอนุญาต

ในปัจจุบันกำหนดให้ฟิลด์ authentication type มีค่าเท่ากับ 2 ซึ่งหมายถึงการพิสูจน์ด้วยรหัสผ่าน (ชนิดที่ไม่เข้ารหัสลับ) และค่ารหัสผ่านนั้นจะต้องส่งมาในฟิลด์ authentication ด้วยเหตุนี้เฟรมอาร์ไอพีที่ใช้การพิสูจน์ตัวจริงจะมีจำนวนเส้นทางที่บรอดคาสต์พร้อมกันในหนึ่งเฟรมได้เพียง 24 เส้นทาง

0	7 8	15 16	31
command	version	unused	
0xFFFF		Authentication type=2	
Authentication			

รูปที่ 5-10 เฟรมอาร์ไอพีเวอร์ชัน 2 เมื่อใช้การพิสูจน์ตัวจริง



บทที่ 6

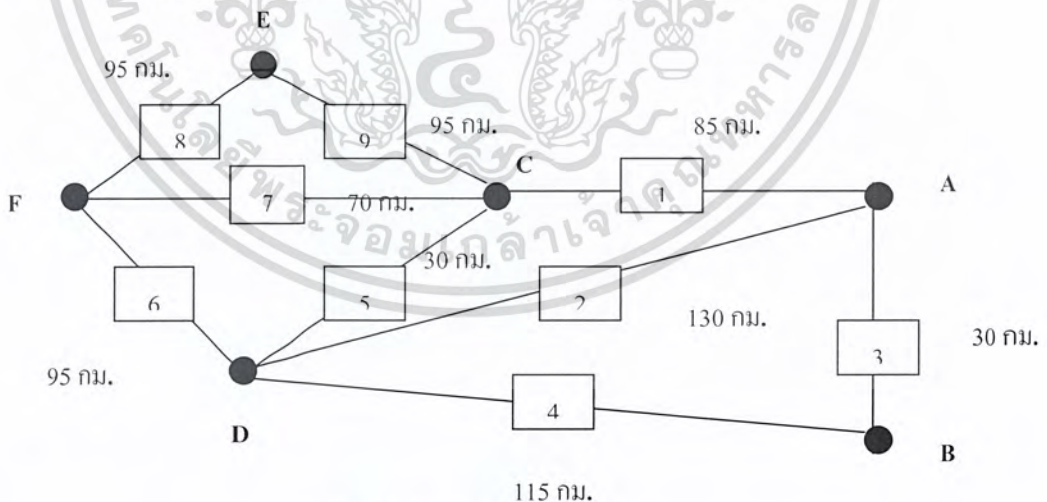
โอเอสพีเอฟ (OSPF : Open shorted Path First)

โอเอสพีเอฟเป็นโพรโตคอลเกตเวย์ภายในที่นิยมใช้อย่างแพร่หลายเนื่องจากสามารถปรับเปลี่ยนการเลือกเส้นทางตามสภาพเครือข่ายได้รวดเร็วและรองรับการทำงานกับเครือข่ายขนาดใหญ่ได้

หลักการพื้นฐานของโพรโตคอลแบบโอเอสพีเอฟจะเป็นโพรโตคอลแบบลิงค์สเตต โดยโอเอสพีเอฟจะทำหน้าที่สร้างแผนที่ให้กับเราเตอร์ 3 ขั้นตอนคือ ขั้นแรกเราเตอร์จะตรวจหาเราเตอร์อื่นที่อยู่ข้างเคียง ขั้นที่สองเราเตอร์จะประกาศค่านี้ออกไปให้เราเตอร์อื่นทุกตัว เมื่อเสร็จสิ้นขั้นตอนนี้เราเตอร์จะทราบว่ามีเราเตอร์อื่นใดอยู่บ้าง ขั้นสุดท้ายเราเตอร์จะอาศัยข้อมูลจากขั้นที่สองสร้างแผนที่ไปยังเราเตอร์อื่นทุกตัวในเครือข่าย เราเตอร์ทุกตัวต่างก็มีแผนที่ประจำตัวที่ทราบถึงเส้นทางทั้งหมดในเครือข่าย

6.1 การทำงานของโอเอสพีเอฟโดยสรุป

เพื่อให้สามารถศึกษาการทำงานของโอเอสพีเอฟได้อย่างชัดเจนเป็นขั้นตอนจะใช้เส้นทางจากเมืองหนึ่งไปยังอีกเมืองหนึ่งดังรูป 6.1 แต่ละเมืองเปรียบเสมือนเราเตอร์และทางหลวงที่เชื่อมต่อกันถือเป็นสายสื่อสารระหว่างเราเตอร์ เส้นทางในรูปเป็นเพียงเส้นทางที่จำลองจากทางหลวงระหว่างเมืองและระยะทางวัดเป็นกิโลเมตร โดยประมาณ หมายเลขประจำเส้นทางตั้งแต่ 1-9 ใช้เรียกชื่อแทนหมายเลขทางหลวงเพื่อความสะดวกต่อการอ้างอิง

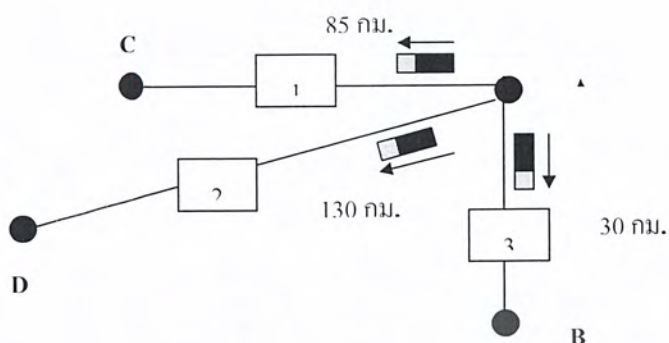


รูปที่ 6-1 แผนที่ทางหลวง

6.2 เรียนรู้เครือข่าย

เราเตอร์ในเครือข่ายที่ใช้โอเอสพีเอฟจะเริ่มค้นเรียนรู้ว่ามีเราเตอร์ใดที่อยู่ข้างเคียงโดยการส่งแพ็กเก็ต “ทักทาย” หรือ ฮัลโลแพ็กเก็ต (Hello Packet) ออกไปทุกๆลิงค์ของเราเตอร์ ตัวอย่างรูปที่ 6-2 เรา

เตอร์ A จะส่งแพ็กเก็ตตกท้ายไปยังเราเตอร์ที่ B,C และ D ในขณะที่เราเตอร์ทั้งสามก็ส่งแพ็กเก็ตตกท้ายไปยังเราเตอร์ A ด้วย



รูปที่ 6-2 เราเตอร์ A ส่งแพ็กเก็ตตกท้าย

เมื่อเราเตอร์ A ได้รับแพ็กเก็ตตอบกลับจาก B,C และ D เราเตอร์ A ก็จะทราบถึงเส้นทางข้างเคียงโดยตรง เราเตอร์ A จะสร้างตารางระยะทางไปยังเราเตอร์ข้างเคียงตามตารางที่ 6.1 ในขณะเดียวกันและในที่สุดเราเตอร์แต่ละจังหวัดต่างก็จะมีข้อมูลระยะทางไปยังเราเตอร์ข้างเคียงทุกตัว

เราเตอร์ข้างเคียง	เส้นทาง	ระยะทาง
B	3	30 กม.
C	1	85 กม.
D	2	130 กม.

ตารางที่ 6-1 ตารางเส้นทางที่เราเตอร์ A

6.3 สร้างฐานข้อมูลเส้นทาง

ขั้นตอนที่สองนี้แต่ละเราเตอร์จะสร้างฐานข้อมูลที่เก็บเส้นทางไปยังเราเตอร์ทุกตัวในเครือข่าย วิธีการที่ใช้คือเราเตอร์สร้างแพ็กเก็ตบรรจุเส้นทางที่มีอยู่ทั้งหมดในขณะนั้นและ”ปล่อย” (Flooding) ให้แพ็กเก็ตไหลจากเราเตอร์หนึ่งไปยังอีกเราเตอร์หนึ่ง

เมื่อเราเตอร์ได้รับแพ็กเก็ตก็จะสำเนาแพ็กเก็ตนั้นส่งออกไปยังลิงค์อื่นด้วย เราเตอร์ต้องตรวจสอบว่าเป็นแพ็กเก็ตเก่าซ้ำของเดิมหรือเป็นแพ็กเก็ตใหม่ หากได้รับแพ็กเก็ตเก่าจากเราเตอร์อื่นให้กำจัดทิ้งไป หากเป็นแพ็กเก็ตใหม่จะต้องปล่อยแพ็กเก็ตไปทุกอินเทอร์เฟซยกเว้นอินเทอร์เฟซที่ได้รับแพ็กเก็ต

รูปแบบของแพ็กเก็ตที่เราเตอร์ส่งออกไปในขั้นนี้เรียกว่า แพ็กเก็ตประกาศลิงก์สเตต (Link State Advertisement) หรือ แพ็กเก็ตแอลเอสเอ (LSA) เมื่อเราเตอร์ทุกตัวปล่อยแพ็กเก็ตแอลเอสเอเช่นเดียวกันเพียงชั่วระยะหนึ่งแพ็กเก็ตจะกระจายไปทั่วทั้งเครือข่าย ท้ายที่สุดแล้วเราเตอร์ทุกตัวจะทราบเส้นทางไปยังเราเตอร์อื่นและได้ตารางเส้นทางเรียกว่า ฐานข้อมูลลิงก์สเตต (Link State Database) จากตัวอย่างนี้ฐานข้อมูลลิงก์สเตตมีค่าตามตารางที่ 6-2

Router A	Router B	Router C	Router D	Router E	Router F
B : 30	A : 30	A : 85	A : 130	C : 110	C : 70
C : 85	D : 115	D : 30	B : 115	F : 95	D : 95
D : 130		E : 110	C : 30		E : 95
		F : 70	F : 95		

ตารางที่ 6-2 ฐานข้อมูลถึงสหภาพประจำแต่ละเราเตอร์

6.4 การคำนวณเส้นทาง

ฐานข้อมูลที่ได้จากขั้นที่แล้วยังไม่ใช้ตารางเส้นทางที่นำไปใช้ได้ ตารางเส้นทางจะได้อีกจากการคำนวณหาเส้นทางที่สั้นที่สุดตามขั้นตอนวิธีของไดจ์สตรา (Dijkstra Algorithm)

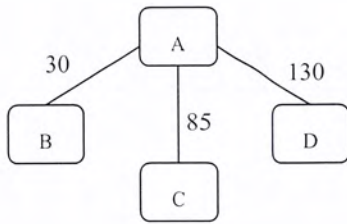
6.4.1 ขั้นตอนวิธีของไดจ์สตรา (Dijkstra Algorithm)

ตามขั้นตอนวิธีไดจ์สตรา เราเตอร์จะสร้างเส้นทางที่เขียนแทนด้วยต้นไม้กำกับทิศทางโดยมีกิ่งเชื่อมไปยังเราเตอร์อื่น รากของต้นไม้ก็คือเราเตอร์ที่เป็นจุดตั้งต้นคำนวณหาเส้นทาง แต่ละเส้นจะกำกับด้วยเลเบลซึ่งแบ่งเส้นทางออกเป็นสองชนิดคือ เส้นทางชั่วคราว และ เส้นทางถาวร เส้นทางชั่วคราวใช้กำกับเส้นทางที่ค้นพบแต่ยังไม่ผ่านการคำนวณว่าเป็นเส้นทางที่สั้นที่สุด ส่วนเส้นทางถาวรใช้กำกับเส้นทางที่คำนวณแล้วว่าสั้นที่สุด

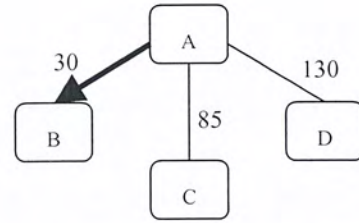
เมื่อเริ่มต้นทำงานที่เราเตอร์ใด เราเตอร์นั้นจะมีระยะทางถึงตัวเองเป็นศูนย์และสร้างเส้นทางชั่วคราวไปยังเราเตอร์ข้างเคียงก่อน ส่วนเส้นทางไปยังเราเตอร์อื่นให้มีค่าเป็นอนันต์

เมื่อใช้ขั้นตอนวิธีของไดจ์สตรากับรูปที่ 6-1 โดยพิจารณาเราเตอร์ A จะได้เส้นทางแรกตามรูปที่ 6-3 (1) โดยมีเส้นทางไปยัง B, C และ D ทั้งสามเส้นทางไม่มีหัวลูกศรซึ่งหมายถึงเส้นทางชั่วคราว ลำดับขั้นที่ (2) ถึง (6) แสดงการหาเส้นทางที่สั้นที่สุดไปยังที่อื่นๆ โดยเส้นทางที่มีหัวลูกศรหมายถึงเส้นทางถาวร

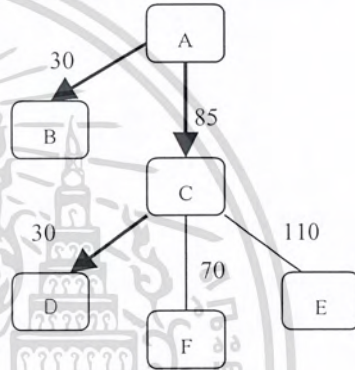
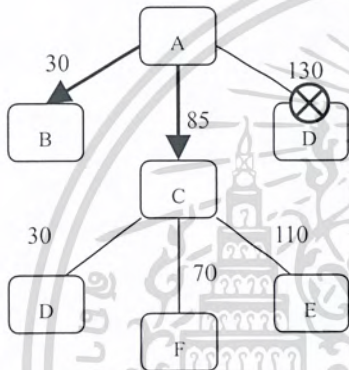
เมื่อเสร็จสิ้นขั้นตอนสุดท้ายแล้วเราเตอร์ A จะมีแผนที่ทั้งเครือข่าย เมื่อต้องการไปยังเส้นทางใดก็เพียงแต่หาเส้นทางจากต้นไม้ที่สร้างขึ้นเท่านั้น ในขณะที่เราเตอร์อื่นก็จะดำเนินการตามขั้นตอนเพื่อหาระยะทางที่สั้นที่สุดไปยังที่อื่นๆในลักษณะเดียวกัน



(1) เราเตอร์ A สร้างต้นไม้แสดงเส้นทางไปยังเราเตอร์ข้างเคียง

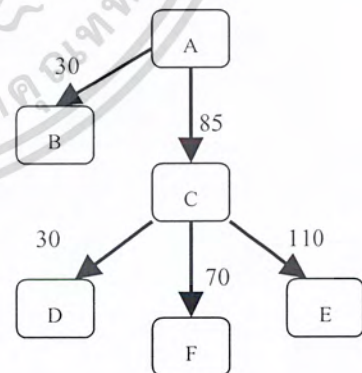
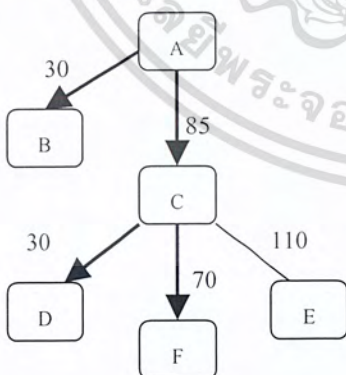


(2) สร้างเส้นทางต่อจากเราเตอร์ B ซึ่งมีเพียงเราเตอร์ D เท่านั้นแต่ระยะทางจาก A ไป D โดยตรงมีค่าเท่ากับ 130 ซึ่งน้อยกว่าการไป D โดยผ่าน B ก่อนซึ่งจะมีค่าเท่ากับ $115+30=145$ ดังนั้นให้เส้นทางไป B เป็นเส้นทางถาวร



(3) สร้างเส้นทางต่อจากเราเตอร์ C แล้วพบว่า เส้นทางไป D ผ่านทาง C ($85+30=115$) สั้นกว่าไป D (130) โดยตรง ดังนั้นให้เส้นทางไป C เป็นเส้นทางถาวร และยกเลิกเส้นทางไป D

(4) จากเราเตอร์ D ไม่พบเส้นทางไปเราเตอร์ใดที่สั้นกว่า จึงให้เส้นทางที่ D เป็นเส้นทางถาวร



(5) จากเราเตอร์ F ไม่พบเส้นทางไปเราเตอร์ใดที่สั้นกว่า จึงให้เส้นทางที่ F เป็นเส้นทางถาวร

(6) จากเราเตอร์ E ไม่พบเส้นทางไปเราเตอร์ใดที่สั้นกว่า จึงให้เส้นทางที่ E เป็นเส้นทางถาวร

รูปที่ 6-3 การคำนวณหาเส้นทางสั้นที่สุดตามขั้นตอนวิธีของไดจ์สตรา

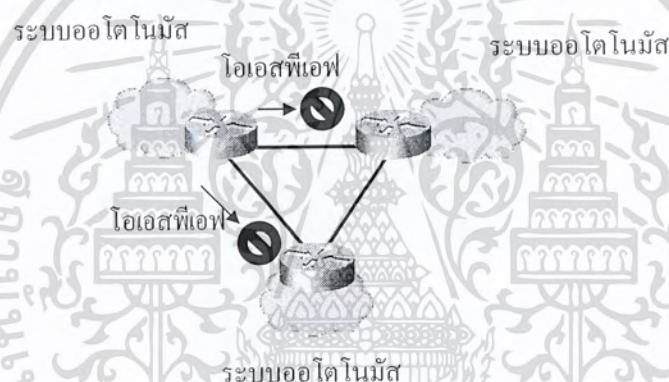
ขั้นตอนวิธีของไดจ์สตราเป็นขั้นตอนวิธีที่ต้องใช้กำลังการคำนวณของซีพียูอย่างมากโดยเฉพาะอย่างยิ่งในเครือข่ายขนาดใหญ่ เราเตอร์จะต้องคำนวณหาต้นไม้ที่แทนเส้นทางสั้นที่สุดใหม่ทุกครั้งที่มีการเปลี่ยนแปลงใดๆในเครือข่าย

แต่ไอเอสพีเอฟมีวิธีการจัดรูปแบบเครือข่ายเพื่อลดการคำนวณเมื่อเครือข่ายเปลี่ยนแปลงซึ่งจะกล่าวในหัวข้อถัดไป

6.5 การจัดองค์ประกอบเครือข่าย

ระบบบอโดโนมัสเป็นตัวกำหนดขอบเขตการทำงานของไอเอสพีเอฟ ข้อมูลเส้นทางไอเอสพีเอฟจะแลกเปลี่ยนระหว่างเราเตอร์ที่อยู่ภายในบอโดโนมัสหนึ่งๆ เท่านั้น

ในระบบบอโดโนมัสหนึ่งๆ ย่อมมีเราเตอร์เชื่อมต่อไปยังระบบอื่นดังตัวอย่างในรูปที่ 6-6 เราเตอร์ในระบบบอโดโนมัสที่เชื่อมกับระบบบอโดโนมัสอื่นเรียกว่า เราเตอร์ขอบปลายระบบบอโดโนมัส (AS boundary routers) และเป็นตัวกันไม่ให้ข้อมูลไอเอสพีเอฟออกไปสู่ระบบบอโดโนมัสอื่น



รูปที่ 6-6 เราเตอร์ขอบปลายของระบบบอโดโนมัส

6.6 การแบ่งระบบบอโดโนมัสออกเป็นพื้นที่

เครือข่ายที่ใช้งานไอเอสพีเอฟย่อมอยู่ในระบบบอโดโนมัสเดียวกัน และอาจเป็นได้ทั้งเครือข่ายขนาดเล็กไปถึงเครือข่ายขนาดใหญ่ ไอเอสพีเอฟสามารถทำงานได้กับเครือข่ายขนาดใหญ่และยังรองรับการขยายขนาดขึ้นไปอีก เพราะมีคุณสมบัติการทำงานที่รองรับการขยายอยู่ในตัว

คุณสมบัติหนึ่งได้แก่การแบ่งเครือข่ายออกเป็นกลุ่มย่อยเพื่อจำกัดขอบเขตการทำงานบางอย่างให้อยู่ในบริเวณใดบริเวณหนึ่งหรือกลุ่มของเราเตอร์กลุ่มหนึ่งๆ ขอบเขตของเครือข่ายที่จัดแบ่งแล้วเรียกว่า พื้นที่ (Area)

พื้นที่ในความหมายของไอเอสพีเอฟคือกลุ่มของโฮสต์หรือเราเตอร์ภายใต้ขอบเขตที่กำหนดเครือข่ายหนึ่งๆ อาจแบ่งออกเป็นหลายพื้นที่ แต่ละพื้นที่จะเชื่อมโยงกับพื้นที่กลางที่เรียกว่า พื้นที่แบ็คโบน (Backbone Area) รูปที่ 6-7 แสดงเครือข่ายที่แบ่งออกเป็น 4 ส่วนแต่ละพื้นที่จะมีหมายเลขประจำพื้นที่ (Area ID) โดยแบ็คโบน

บทที่ 7

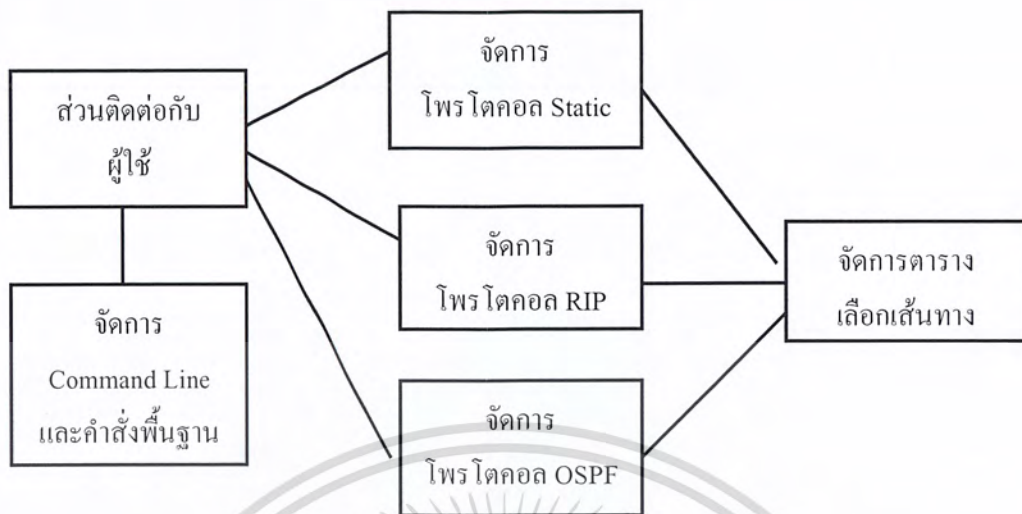
หลักการสร้างและออกแบบโปรแกรม

7.1 โครงสร้างของโปรแกรม (Design)

โปรแกรมนี้ออกแบบโดยใช้วิธี ออบเจกต์โอเรียนเต้ดโปรแกรมมิง (Object Oriented Programming) เพื่อให้การเขียนโปรแกรมเป็นไปได้ง่ายต่อการเพิ่มและเปลี่ยนแปลงโปรแกรม โดยแบ่งการทำงานของโปรแกรมออกเป็นส่วนย่อยๆ เป็นคลาสต่างๆ แต่ละคลาสจะมีหน้าที่การทำงานที่แตกต่างกันดังนี้

- **Frame1** เป็นคลาสสำหรับสร้างหน้าต่างอินเทอร์เน็ตเพื่อติดต่อกับผู้ใช้ โดยหน้าต่างอินเทอร์เน็ตประกอบไปด้วยเมนูบาร์ ทูลบาร์ และส่วนที่เป็นหน้าต่างทำงานของแต่ละเราเตอร์
- **Frame2** เป็นคลาสที่แสดงผลภาพรวมของระบบ รวมทั้งจัดการกับการเลื่อนตำแหน่งของภาพ
- **RouterConsole** เป็นคลาสที่เก็บหน้าต่างสำหรับทำงานของเราเตอร์แต่ละตัว
- **Router** เป็นคลาสที่ประกอบไปด้วยข้อมูลต่างๆ ของเราเตอร์เช่น ชื่อเราเตอร์ สถานะการทำงานของเราเตอร์ โหมดการทำงานของเราเตอร์ อินเทอร์เน็ตของเราเตอร์ และตารางการค้นหาเส้นทางของเราเตอร์ โดยฟังก์ชันการทำงานของคลาสนี้จะมีฟังก์ชันสำหรับส่งตารางค้นหาเส้นทาง และรับตารางค้นหาเส้นทางจากเราเตอร์ตัวอื่น
- **RouterImg** เป็นคลาสที่เก็บข้อมูลเกี่ยวกับภาพของเราเตอร์แต่ละตัว
- **Interface** เป็นคลาสที่ประกอบไปด้วยข้อมูลของอินเทอร์เน็ตแต่ละตัว เช่น ชื่ออินเทอร์เน็ต ไอพีแอดเดรส ประเภทของอินเทอร์เน็ต ชับเน็ต และสถานะของอินเทอร์เน็ต
- **RoutingTable** เป็นคลาสที่เก็บข้อมูลของตารางค้นหาเส้นทางเช่น เครือข่ายปลายทาง เกตเวย์ที่ออก ค่าเมตริก หรือประเภทของข้อมูลค้นหาเส้นทางว่าเป็นแบบสแตติก อาร์ไอพีหรือแบบอื่น ฟังก์ชันการทำงานของหลักก็จะเป็นการเพิ่มข้อมูลลงในตารางค้นหาเส้นทาง
- **SwitchCMD** มีหน้าที่รับคำสั่งจากผู้ใช้มาตรวจสอบว่าเป็นคำสั่งที่ถูกต้องก่อนจะไปเลือกการทำงานของแต่ละคำสั่ง ซึ่งจะมีฟังก์ชันหลักเป็นตัวตรวจสอบว่าเป็นคำสั่งที่ถูกต้อง และส่งไปทำงานตามแต่ละคำสั่ง แต่จะมีเป็นฟังก์ชันย่อยที่ช่วยฟังก์ชันใหญ่ในการตรวจสอบ เช่นฟังก์ชันสำหรับหาช่องว่าง
- **Command** เป็นคลาสที่จัดการเกี่ยวกับคำสั่งคำสั่งของเราเตอร์
- **Rip** เป็นคลาสที่จัดการเกี่ยวกับ การเลือกเส้นทางแบบ RIP โดยคลาสนี้มีการทำงานเป็น Threads รับส่งตารางเลือกเส้นทางเป็นช่วงเวลาที่ตั้งไว้ตามข้อกำหนดของโปรโตคอล
- **Ospf** เป็นคลาสที่จัดการเกี่ยวกับ การเลือกเส้นทางแบบ OSPF มีการทำงานเป็น Threads

โดยคลาสต่างๆ นี้สามารถแบ่งตามการจัดการได้ดังนี้



รูปที่ 7-1 การออกแบบโปรแกรม

7.2 การจัดการส่วนติดต่อผู้ใช้ (Graphic User Interface : GUI)



รูปที่ 7-2 หน้าจอหลักของโปรแกรม

ส่วนของ GUI จะมี 3 ส่วนหลักๆ คือ

1. ส่วนหน้าจอหลักของโปรแกรม ประกอบด้วยทูลบาร์และส่วนของแท็บเพน หน้าจอแท็บเพนแต่ละแท็บจะแทนหน้าจอคอนโซลของเราเตอร์แต่ละตัว การออกแบบส่วนนี้จะแบ่งเป็น 2 คลาส คือ คลาส Frame1 ซึ่งเป็นหน้าจอหลัก และคลาส RouterConsole ซึ่งเป็นส่วนของหน้าจอคอนโซลของเราเตอร์ 1 ตัว เมื่อมีการเพิ่มเราเตอร์เข้าไปในระบบก็เป็นการสร้างออบเจกต์ของ RouterConsole เพิ่มขึ้น

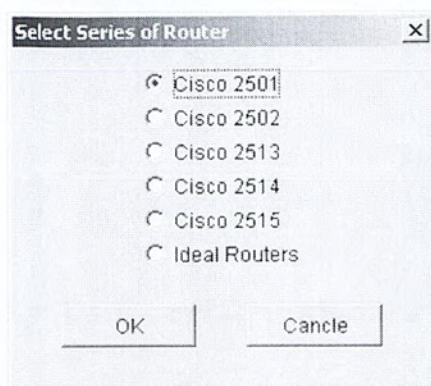
ฟังก์ชันการทำงานสำหรับส่วนนี้จะอยู่ในคลาส Frame1 และ RouterConsole มีฟังก์ชันการทำงานดังนี้

- Frame1() เป็นคอนสตรัคเตอร์ของคลาส ในฟังก์ชันนี้จะแสดงหน้าจอโปรแกรมหลักทั้งส่วนของเมนูบาร์ ทูลบาร์ และแท็บเพน
- AddTab() เป็นฟังก์ชันสำหรับเพิ่มหน้าต่างแท็บเพน เมื่อผู้ใช้คลิกปุ่มเพิ่มเราเตอร์
- SetTab() เป็นฟังก์ชันสำหรับกำหนดข้อความที่แสดงตรงแท็บเพน จะใช้ในกรณีที่ใช้ใช้คำสั่งเปลี่ยนชื่อของเราเตอร์
- ส่วนฟังก์ชันที่เหลือเป็นฟังก์ชันสำหรับการจัดการการกระทำของผู้ใช้สำหรับแต่ละคอมโพเนนท์ ซึ่งประเภทของการจัดการกับการกระทำของผู้ใช้ออกเป็น 2 ประเภท คือการกระทำที่ผู้ใช้กระทำกับคอมโพเนนท์หนึ่งๆ และการกระทำที่เกี่ยวกับคีย์บอร์ดในขณะที่คอมโพเนนท์หนึ่งๆ ถูกโฟกัสอยู่ เช่น `jNewButton_actionPerformed()` เป็นการจัดการกับการกระทำของผู้ใช้ที่คลิกปุ่มเพิ่มเราเตอร์ และ `jNewButton_keyPressed()` จัดการการกระทำของการกดคีย์บอร์ดขณะที่ปุ่มเพิ่มเราเตอร์ถูกโฟกัสอยู่

คลาส RouterConsole

- RouterConsole() เป็นคอนสตรัคเตอร์ของคลาสทำหน้าที่แสดงส่วนที่จะนำหน้าจอไปวางบนแท็บเพน
- EnterKey() เป็นฟังก์ชันที่จัดการกับการทำงานของคีย์ Enter เพื่อเริ่มต้นการทำงาน เนื่องจากเราเตอร์สามารถกำหนดระดับความปลอดภัยของผู้ใช้ได้ เมื่อผู้มิจฉาชีพคีย์ Enter ก็สามารถทำงานกับเราเตอร์ได้เลยหรือจะต้องใส่พาสเวิร์ดก่อนถึงจะเริ่มทำงานได้
- ShowPrompt() เป็นฟังก์ชันสำหรับแสดงพร้อมท์ โดยจะดูจากโหมดการทำงานของเราเตอร์แล้วจึงแสดงพร้อมท์ของโหมดการทำงานนั้นๆ สำหรับฟังก์ชันนี้จะรับอาร์กิวเมนต์เข้าไปเป็นสตริงเพื่อนำไปแสดงผลต่อจากพร้อมท์
- ส่วนฟังก์ชันที่เหลือเป็นฟังก์ชันที่จัดการกับการกระทำที่เกี่ยวกับคีย์บอร์ดและเมาส์ เช่น ฟังก์ชัน `jTextArea1_keyPressed()` จัดการการกระทำที่เกี่ยวกับคีย์บอร์ดในขณะที่ TextArea ถูกโฟกัส ฟังก์ชัน `jTextArea1_mouseClicked()` จัดการการกระทำของการคลิกเมาส์ในขณะที่ TextArea ถูกโฟกัส และฟังก์ชัน `jTextArea1_mouseReleased()` จัดการกับการกระทำของการปล่อยเมาส์ ทำให้ผู้ใช้ไม่สามารถเลือกข้อความได้

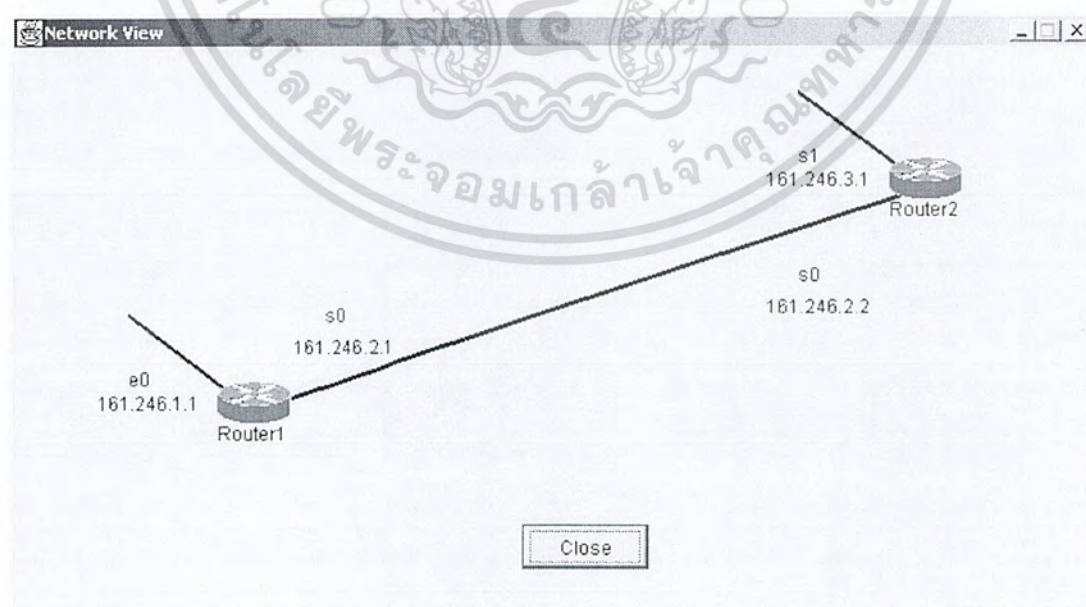
2. หน้าจอสำหรับเลือกรุ่นของเราเตอร์ เมื่อเพิ่มเราเตอร์เข้าไปในระบบ ผู้ใช้จะต้องเลือกรุ่นของเราเตอร์ที่ต้องการ ซึ่งแต่ละรุ่นก็จะมีอินเทอร์เฟซที่ต่างกัน



รูปที่ 7-3 หน้าจอสำหรับเลือกรุ่นของเราเตอร์

ฟังก์ชันการทำงานสำหรับส่วนนี้จะอยู่ในคลาส Dialog1 มีฟังก์ชันการทำงานดังนี้

- Dialog1() เป็นคอนสตรัคเตอร์สำหรับแสดงหน้าจอไดอะล็อก
 - getSelectChoice() เป็นฟังก์ชันสำหรับตรวจสอบว่าผู้ใช้เลือกตัวเลือกใด แล้วให้ค่าของตัวเลือกที่ผู้ใช้เลือกออกมาเป็นค่าตัวเลข
 - jButton1_actionPerformed() เป็นฟังก์ชันสำหรับจัดการการกระทำของผู้ใช้กับปุ่ม OK
 - jButton2_actionPerformed() เป็นฟังก์ชันสำหรับจัดการการกระทำของผู้ใช้กับปุ่ม Cancel
3. หน้าจอสำหรับแสดงภาพรวมของเครือข่าย ใช้ Frame ในการแสดงผล จะแสดงการเชื่อมต่อระหว่างเราเตอร์ตามที่กำหนดให้กับเราเตอร์แต่ละตัว



รูปที่ 7-4 ภาพแสดงภาพรวมของเครือข่าย

ฟังก์ชันการทำงานสำหรับส่วนนี้จะอยู่ในคลาส Frame2 และ Panel2 มีฟังก์ชันการทำงานดังนี้

คลาส Frame2

- Frame2() เป็นคอนสตรัคเตอร์ของคลาสสำหรับแสดงผลภาพรวมของเครือข่าย

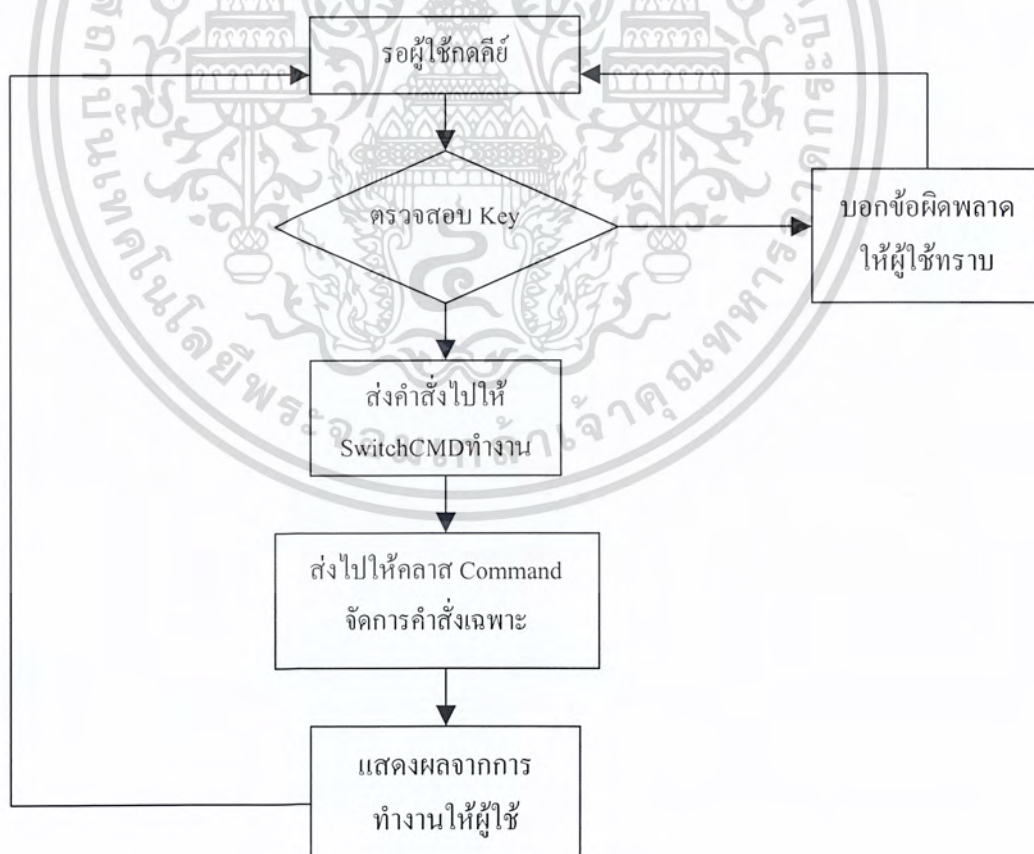
คลาส Panel2

- Panel2() เป็นคอนสตรัคเตอร์ของคลาส ฟังก์ชันนี้จะแสดงเฉพาะส่วนพื้นของหน้าจอ
- paintComponent() เป็นฟังก์ชันสำหรับวาดภาพเรเตอร์ และลากเส้นของแต่ละอินเทอร์เฟซ
- this_mousePressed() เป็นฟังก์ชันจัดการกับการคลิกเมาส์
- this_mouseReleased() เป็นฟังก์ชันจัดการกับการปล่อยเมาส์
- this_mouseDragged() เป็นฟังก์ชันจัดการกับการคลิกเมาส์และเลื่อนเมาส์ไปพร้อมกัน

7.3 การจัดการส่วนของ Command line

มีคลาสที่จัดการส่วนนี้ 1 คลาสคือ คลาส SwitchCMD จะคอยรับคำสั่งที่ส่งมาจากคลาส Frame1 ที่จัดการในส่วนที่ติดต่อกับผู้ใช้ เมื่อรับคำสั่งมาแล้วจะนำมาตรวจสอบว่าเป็นคำสั่งที่ถูกต้อง สามารถทำงานได้หรือไม่ ถ้าถูกต้องจะส่งคำสั่งไปทำงาน และรอรับผลการทำงานเพื่อนำมาแสดงผลให้ผู้ใช้ทราบ

ขั้นตอนการทำงานของโปรแกรม



รูปที่ 7-5 แสดงขั้นตอนการจัดการกับส่วน Commands line

ฟังก์ชันการทำงานสำหรับส่วนนี้จะอยู่ในคลาส SwitchCMD และ Command มีฟังก์ชันการทำงานดังนี้

คลาส SwitchCMD

- sendCMD() เป็นฟังก์ชันหลักในการทำงานของคลาสนี้ ค่าที่รับเข้ามาในฟังก์ชันนี้จะมีอยู่ 2 อย่างคือ เราเตอร์และคำสั่งที่ต้องการทำงานกับเราเตอร์นั้นๆ โดยเริ่มต้นที่ดูโหมมการทำงานของเราเตอร์เพื่อเลือกชุดคำสั่งที่สามารถทำงานได้กับโหมมคนั้น หลังจากนั้นจะตัดคำมาทีละคำเพื่อเปรียบเทียบกับคำสั่งที่มีทุกคำสั่ง ถ้าเจอคำสั่งที่สอดคล้องกันก็จะนับจำนวนคำสั่งนั้นไว้ เมื่อเปรียบเทียบครบทุกคำสั่ง ก็จะนำค่าของจำนวนคำสั่งที่สอดคล้องกันมาตรวจสอบว่ามากกว่า 1 คำสั่งหรือไม่ ถ้ามากกว่าก็จะไม่เลือกคำสั่งใดทำงาน และรายงานผลกลับไปให้กับผู้ใช้ ถ้าเท่ากับ 1 ก็จะส่งไปให้ฟังก์ชันของแต่ละคำสั่งทำงาน
- ฟังก์ชันที่เหลือจะเป็นฟังก์ชันของแต่ละคำสั่ง โดยบางคำสั่งจะทำงานที่ฟังก์ชันนี้เลย แต่บางคำสั่งจะส่งไปให้คลาส Command จัดการต่อ ซึ่งคำสั่งพวกนี้มักเป็นคำสั่งที่ต้องตรวจสอบอะไรมากมาย

คลาส Command

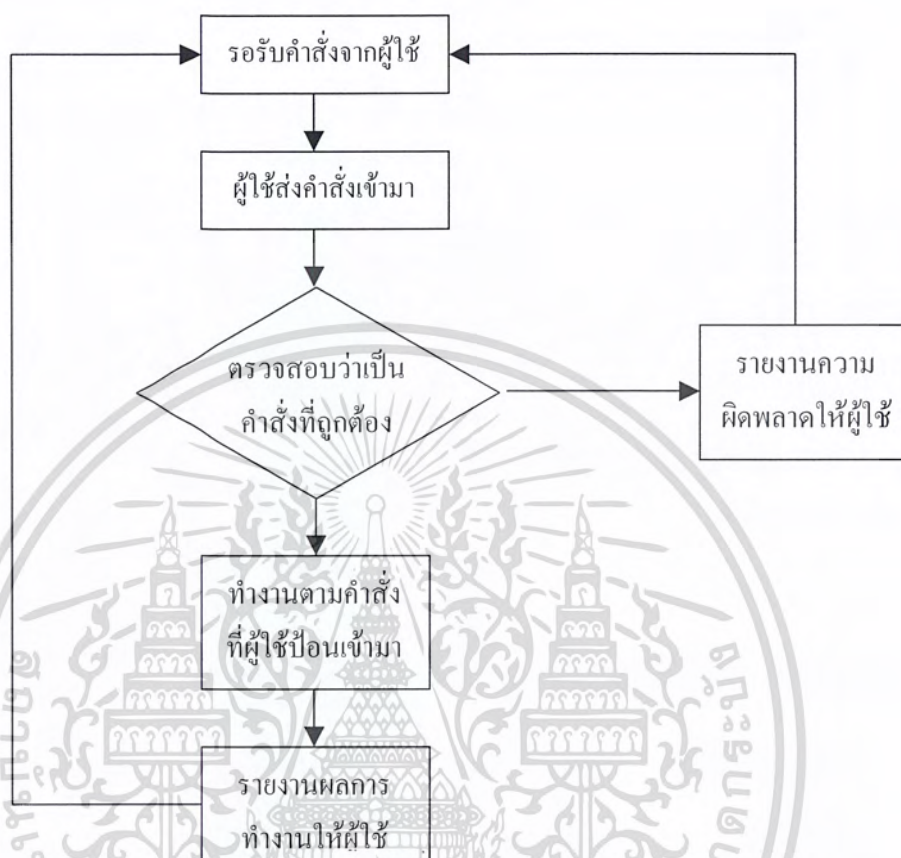
- eraseStartUp() จะเปิดไฟล์ history.dat ขึ้นมาอ่าน แล้วนำคำที่อ่านได้เก็บไว้เรื่อยๆ จนกระทั่งตรวจสอบพบว่าเป็นส่วนที่เก็บค่าของเราเตอร์ตัวที่ต้องการลบค่า ก็จะไม่นำคำที่อ่านมาได้ไปเก็บจนหมดของเราเตอร์ตัวดังกล่าวก็จะอ่านค่าแล้วเก็บไว้เช่นเดิม จนกระทั่งหมดไฟล์ ก็จะปิดไฟล์แล้วเปิดขึ้นมาเพื่อเขียนไฟล์ใหม่ จากนั้นนำค่าที่ได้จากการอ่านครั้งที่แล้วเขียนทับลงไปทีไฟล์ history.dat แล้วจึงปิดไฟล์
- ping() ฟังก์ชันนี้จะทำงานแบบรีเคอร์ซีฟ โดยอ่านค่าที่เก็บไว้ในตารางค้นหาเส้นทางเพื่อตรวจสอบว่ามีการเก็บข้อมูลสำหรับไปยังจุดหมายปลายทางนั้นๆ แล้วไปตรวจสอบที่เราเตอร์ต่อๆ ไปจนกว่าจะถึงจุดหมายปลายทาง หรือไม่สามารถหาข้อมูลเพื่อไปยังจุดหมายปลายทางนั้นได้
- readStartUp() ฟังก์ชันนี้จะอ่านค่าที่เก็บไว้ในไฟล์ history.dat มาแล้วสร้างเราเตอร์ขึ้นมา แล้วกำหนดค่าตามที่กำหนดไว้ในไฟล์ที่เราเตอร์ จนกระทั่งหมดไฟล์
- saveRunningConfig() เช่นเดียวกับฟังก์ชัน eraseStartUp() เมื่อพบเราเตอร์ที่ต้องการจะอ่านค่าที่กำหนดให้กับเราเตอร์ในขณะนั้นมาแทนการอ่านจากไฟล์ แล้วจึงเขียนทับไฟล์เดิม
- show_ip_ospf_inf() ทำหน้าที่แสดงผลจากคำสั่ง show ip ospf <interface name> เพื่อบอกว่าที่อินเทอร์เฟซนั้นๆ มีสถานะของโพรโตคอลโอเอสพีเอฟเป็นเช่นไร
- show_ip_ospf_neighbor() ทำหน้าที่แสดงเราเตอร์ข้างเคียงที่อยู่ในโพรโตคอลโอเอสพีเอฟเช่นเดียวกัน
- showCdp() ฟังก์ชันนี้จะแสดงผลออกมาโดยดึงค่า CDP ที่เก็บไว้ของเราเตอร์ออกมาแสดงผล
- showCdpEntry() ฟังก์ชันนี้จะหาเราเตอร์ข้างเคียงก่อน หลังจากนั้นจึงนำค่า CDP ของเราเตอร์ข้างเคียงมาแสดงผล

- showCdpNeighbors() เช่นเดียวกับ showCdpEntry() แต่ข้อมูลที่นำมาแสดงเป็นข้อมูลคนละประเภทกันคือ เป็นข้อมูลเกี่ยวกับอินเทอร์เฟซที่ต่อกัน
- showHistory() ฟังก์ชันนี้จะนำข้อมูลเกี่ยวกับคำสั่งที่เคยใช้งานของเราเตอร์ออกมาแสดงผล
- showInt() ฟังก์ชันนี้จะแสดงข้อมูลของแต่ละอินเทอร์เฟซของเราเตอร์
- showIPProtocol() ฟังก์ชันนี้จะแสดงข้อมูลโพรโทคอลของเราเตอร์ โดยดูจากค่าตัวแปร RouteType ของเราเตอร์
- showIpRoute() ฟังก์ชันนี้จะแสดงข้อมูลของแต่ละเส้นทางในตารางค้นหาเส้นทาง
- showRunningConfig() ฟังก์ชันนี้จะดูข้อมูลการกำหนดค่าของเราเตอร์ในขณะนั้นมาแสดงผล
- showStartup() ฟังก์ชันนี้จะทำงานคล้ายกับฟังก์ชัน eraseStartup() แต่กลับกันโดยเมื่อเราเตอร์ที่ต้องการแล้วจึงจะนำค่าที่อ่านได้มาเก็บไว้ แต่หลังจากที่อ่านเสร็จจะไม่มีกรเขียนไฟล์ แต่จะนำค่าที่อ่านได้แสดงผลออกมาแทน
- showVersion() ฟังก์ชันนี้จะแสดงผลเกี่ยวกับเวอร์ชันของ IOS และข้อมูลของเราเตอร์ โดยจึงจากค่าตัวแปรของเราเตอร์
- telnetCmd() ฟังก์ชันนี้จะตรวจสอบว่าไอพีที่ต้องการเทลเน็ตเข้าไบนั้นถูกต้องหรือไม่ และตรวจสอบต่อว่าพอร์ตเทลเน็ตเปิดไว้หรือไม่
- trace() คล้ายกับฟังก์ชัน ping() แต่จะแสดงผลแต่ละเส้นทางที่ค้นหาได้ออกมาด้วย

7.4 การจัดการส่วนของการ Add Router

คลาสที่จัดการส่วนนี้คือคลาส Frame1 เป็นการจัดการกับไอคอน โดยโปรแกรมจะดักจับการคลิกปุ่มเพิ่มเราเตอร์ เมื่อผู้ใช้คลิกปุ่มเพิ่มเราเตอร์ โปรแกรมจะแสดงหน้าต่างให้ผู้ใช้เลือกรุ่นของเราเตอร์ก่อน ในส่วนนี้ผู้ใช้จำเป็นต้องเลือก โปรแกรมจึงจะสร้างออบเจกต์ของเราเตอร์ขึ้นมา พร้อมกับออบเจกต์ของเราเตอร์ที่ใช้ในส่วนของการแสดงภาพเครือข่ายโดยรวมขึ้นมา จากนั้นจะนำเก็บค่าตัวอ้างอิง (Reference) ของพื้นที่นั้นไว้เพื่อใช้สำหรับอ้างอิงถึงภายหลัง

ขั้นตอนการทำงานของโปรแกรม



รูปที่ 7-7 แสดงขั้นตอนการจัดการอินเทอร์เน็ตของเรเตอร์

ฟังก์ชันการทำงานนี้จะอยู่ในคลาส Interface มีฟังก์ชันการทำงานดังนี้

- Interface() เป็นคอนสตรัคเตอร์ของคลาส Interface ทำหน้าที่กำหนดค่าเริ่มต้นให้กับอินเทอร์เน็ต เช่น ชื่อของอินเทอร์เน็ต เป็นต้น
- AddNewInt() เป็นฟังก์ชันสำหรับเพิ่มอินเทอร์เน็ตให้กับเรเตอร์ แต่จะต้องตรวจสอบก่อนว่าจำนวนของอินเทอร์เน็ตประเภทนั้นเกินหรือไม่ ถ้าไม่เกินจึงจะสามารถเพิ่มอินเทอร์เน็ตได้
- findNetAdd() เป็นฟังก์ชันสำหรับหาค่าเลขประจำเครือข่าย (Net Address) เพื่อนำไปใช้ในการทำงานอื่น สำหรับฟังก์ชันที่ไม่ส่งค่าอาร์กิวเมนต์ใดๆ มาให้ถือว่าให้หาค่าของอินเทอร์เน็ตที่เรียกฟังก์ชันนั้น แต่ถ้าส่งค่าไอพีแอดเดรสกับค่าของซับเน็ตมาให้จะหาค่าของเลขประจำเครือข่ายโดยพิจารณาจากค่าทั้ง 2 ค่านี้

- `isIp()` ฟังก์ชันนี้จะรับอาร์กิวเมนต์มาเป็นสตริง เพื่อนำมาตรวจสอบว่าค่าไอพีแอดเดรสนั้นเป็นค่าไอพีที่ถูกต้องหรือไม่ โดยค่าของไอพีแอดเดรสจะต้องมี 4 ชุด แต่ละชุดจะต้องมีค่าอยู่ระหว่าง 0-255
- `noshutInt()` เป็นการกำหนดให้อินเทอร์เฟซนั้นอยู่ในสถานะที่ทำงาน
- `shutInt()` เป็นการกำหนดให้อินเทอร์เฟซอยู่ในสถานะที่ไม่สามารถทำงานได้
- `setIpAddress()` เป็นการกำหนดค่าไอพีแอดเดรสให้กับอินเทอร์เฟซล่าสุดที่เพิ่งทำงานไป โดยจะดึงอินเทอร์เฟซตัวสุดท้ายมาแล้วกำหนดค่าให้กับอินเทอร์เฟซนั้น

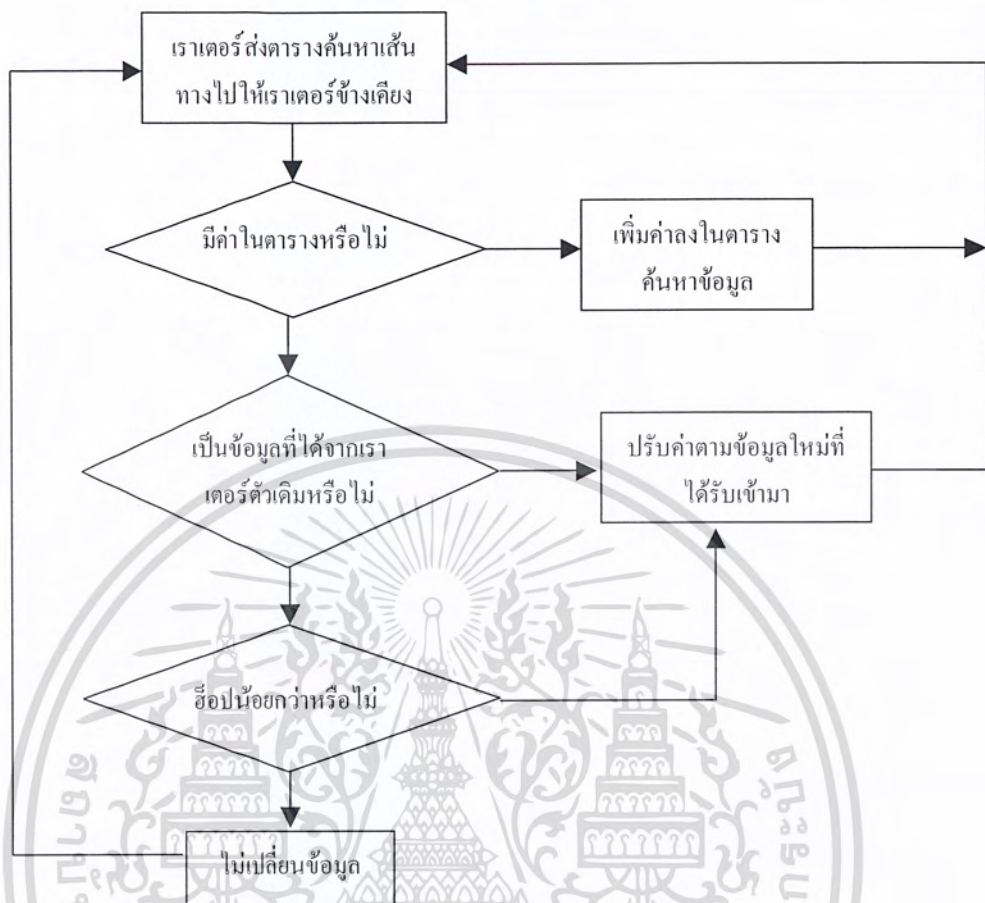
7.6 การจัดการส่วนของ RIP

คลาส `RIP` นี้เป็นคลาสที่สืบทอด (Inherit) มาจากคลาสแม่ที่เป็นเธรด (Thread) เนื่องจากการทำงานโปรโตคอล `RIP` จะส่งตารางค้นหาเส้นทางให้กันตลอดเวลา ดังนั้นการแตกโปรเซสให้โปรแกรมจัดการกับส่วนที่แลกเปลี่ยนตารางเส้นทางเป็นโปรเซสแบบเธรด จะสิ้นเปลืองเวลาที่ใช้้น้อยกว่า รอให้โปรแกรมทำงานเอง สำหรับส่วนของตารางค้นหาเส้นทาง เราเตอร์แต่ละตัวจะมีเวกเตอร์ที่เก็บค่าอ้างอิงของออบเจกต์ของคลาส `RoutingTable`

วิธีแลกเปลี่ยนตารางค้นหาเส้นทาง เนื่องจากเป็นโปรแกรมจำลองจึงไม่มีการส่งข้อมูลจริงๆ แต่จะตรวจสอบว่าเป็นเราเตอร์ตัวที่อยู่ติดกันเท่านั้นจึงจะสามารถเข้าไปนำข้อมูลของตารางค้นหาเส้นทางออกมาได้ เปรียบเสมือนกับการทำงานจริงๆ ที่เราเตอร์ข้างเคียงเท่านั้นที่จะได้รับข้อมูลตารางค้นหาเส้นทางเมื่อได้ ข้อมูลของตารางค้นหาเส้นทางมาแล้วการพิจารณาว่าจะจัดการกับข้อมูลนั้นอย่างไรจะทำตามโปรโตคอลอาร์ไอพี คือแบ่งการตรวจสอบออกเป็น 3 กรณีดังนี้

1. ถ้าข้อมูลตารางค้นหาเส้นทางที่ได้รับมานั้น เป็นข้อมูลของจุดหมายปลายทางที่ไม่มีอยู่ในตารางค้นหาเส้นทาง โปรแกรมจะเพิ่มเส้นทางนั้นลงในตารางค้นหาเส้นทางโดยสร้างออบเจกต์ของ `RoutingTable` แล้วเก็บค่าอ้างอิงไว้ที่เราเตอร์
2. ถ้าข้อมูลตารางค้นหาเส้นทางที่ได้รับมานั้น เป็นข้อมูลของจุดหมายปลายทางที่มีอยู่แล้วในตารางค้นหาเส้นทางให้พิจารณาค่าของเมตริกซ์ต่อไปโดย
 - ถ้าค่าเมตริกใหม่มีค่าน้อยกว่าเส้นทางเดิม จะเปลี่ยนค่าในตารางเลือกเส้นทางให้ไปตามทางเส้นทางใหม่
 - ถ้าค่าเมตริกใหม่มีค่ามากกว่าหรือเท่ากับเส้นทางเดิม จะไม่เปลี่ยนค่าใดๆ ในตารางค้นหาเส้นทาง
3. ถ้าข้อมูลตารางค้นหาเส้นทางที่ได้รับมานั้น เป็นข้อมูลที่มีเกตเวย์เป็นเราเตอร์ตัวเดียวกันกับเราเตอร์ตัวที่ส่งตารางค้นหาเส้นทางมาให้ ให้ปรับค่าตารางค้นหาเส้นทางนั้นตามค่าที่ได้รับมาใหม่ โดยไม่สนใจว่าค่าเมตริกซ์ใหม่นั้นจะมีค่ามากกว่าหรือน้อยกว่าเส้นทางเดิม

ขั้นตอนการทำงานของโปรแกรม



รูปที่ 7-8 ขั้นตอนการทำงานของโปรโตคอล RIP

ฟังก์ชันการทำงานนี้จะอยู่ในคลาส RIP และ Router มีฟังก์ชันการทำงานดังนี้

คลาส Rip

- run() เป็นฟังก์ชันสำหรับให้เธรดทำงาน โดยจะทำงานทุกๆ 30 วินาที เพื่อเรียกให้ฟังก์ชันของเราเตอร์ให้ส่งตารางค้นหาเส้นทาง

คลาส Router

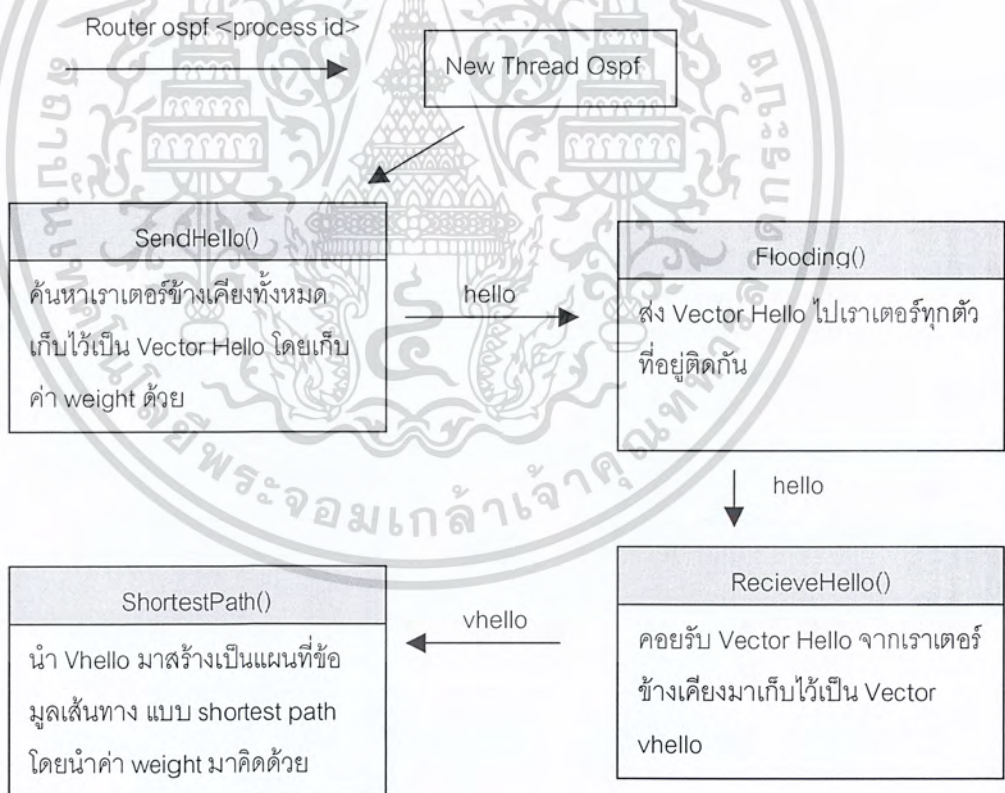
- sendRoutingTable() ฟังก์ชันนี้จะตรวจสอบว่ามีเราเตอร์ใดอยู่ข้างเคียงบ้าง และเราเตอร์นั้นจะต้องใช้โปรโตคอลอะไรอีกด้วย แล้วจึงเรียกฟังก์ชัน recieveRoutingTable() เพื่อให้เราเตอร์นั้นรับค่าตารางค้นหาเส้นทาง
- recieveRoutingTable() ฟังก์ชันนี้จะตรวจสอบค่าตารางเส้นทางที่ได้รับมาและปรับเปลี่ยนตามโปรโตคอล

7.7 การจัดการเกี่ยวกับโปรโตคอลโอเอสพีเอฟ

เมื่อผู้ใช้ป้อนคำสั่ง router ospf <Process id> โปรแกรมจะทำการสร้าง Thread ของ คลาส Ospf ขึ้นมาใหม่

การทำงานของโปรโตคอลโอเอสพีเอฟหลังจากสร้าง Thread มีขั้นตอนหลัก 3 ขั้นตอน คือ

1. ตรวจสอบเราเตอร์อื่นที่อยู่ข้างเคียงโดยใช้ฟังก์ชัน sendHello() โดยฟังก์ชันนี้จะทำการตรวจสอบหาเราเตอร์ที่อยู่ข้างเคียงมาเก็บไว้แบบเวกเตอร์
2. เราเตอร์ประกาศเวกเตอร์นี้ออกไปให้เราเตอร์อื่นทุกตัวโดยใช้ฟังก์ชัน Flooding() และเราเตอร์อื่นจะรับโดยใช้ฟังก์ชัน recieveHello() เมื่อเสร็จสิ้นขั้นตอนนี้เราเตอร์ จะทราบว่า มีเราเตอร์อื่นใดอยู่ในระบบบ้าง
3. เราเตอร์จะอาศัยข้อมูลจากขั้นที่สองสร้างแผนที่ไปยังเราเตอร์อื่นทุกตัวในเครือข่ายโดยใช้ฟังก์ชัน shortestPath() และเก็บแผนที่ข้อมูลเส้นทางนี้เป็นแบบเวกเตอร์ โดยจะเป็นเส้นทางค้นทางไปยังทุกเราเตอร์ปลายทาง
4. เนื่องจากมีการทำงานเป็นแบบ Thread จึงทำให้แผนที่ข้อมูลเส้นทางถูกปรับปรุงทุกๆ ช่วงเวลาที่กำหนด



รูปที่ 7-9 ขั้นตอนการทำงานของโปรโตคอลโอเอสพีเอฟ

ฟังก์ชันการทำงานนี้จะอยู่ในคลาส Ospf และ Router มีฟังก์ชันการทำงานดังนี้

คลาส Ospf

- run() เป็นฟังก์ชันสำหรับให้เทรดทำงาน โดยจะทำงานทุกๆ ช่วงเวลา เพื่อเรียกให้ฟังก์ชันของเราเตอร์ให้ทำฟังก์ชันดังต่อไปนี้

sendHelloPackage(r)	ทำหน้าที่ตรวจสอบหาเราเตอร์ข้างเคียงมาเก็บไว้แบบเวกเตอร์
flooding(r)	ทำหน้าที่ส่งเวกเตอร์ที่ได้จาก sendHelloPackage(r) ไปแบบทุกทิศทางทุกทางที่ติดกับตัวเอง
shortest_path(r)	คำนวณหาระยะทางที่สั้นที่สุดจากตารางข้อมูลที่ได้

คลาส Router

- recieveHelloTable() ทำหน้าที่คอยรับเวกเตอร์ที่ได้จากฟังก์ชัน sendHelloTable() มาเก็บไว้เป็นฐานข้อมูลเพื่อใช้ในการหาเส้นทางที่สั้นที่สุด

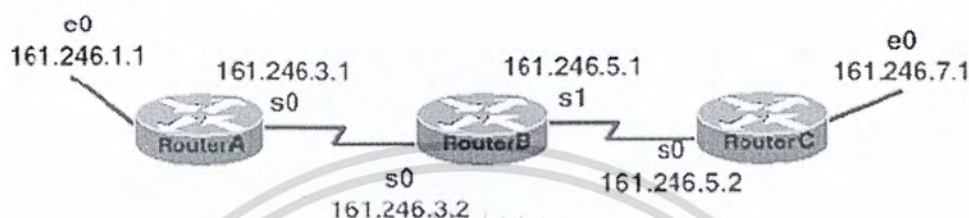


บทที่ 8

ตัวอย่างและการทดสอบการทำงานของโปรแกรม

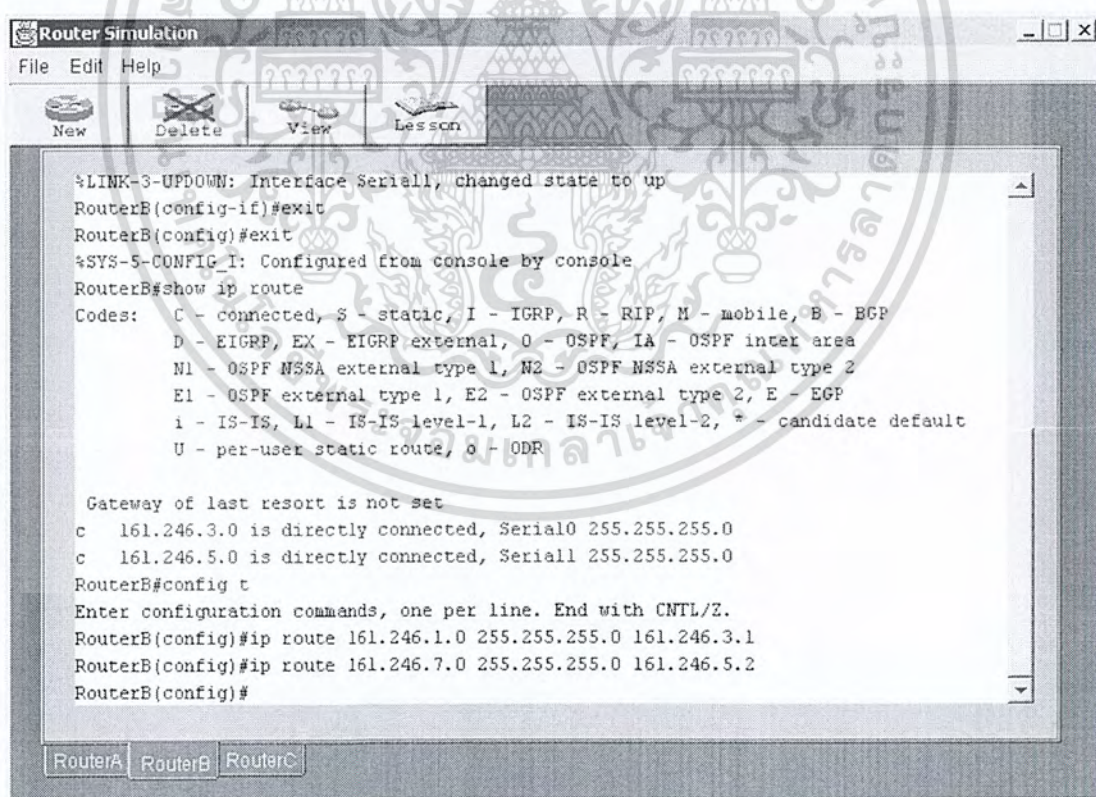
8.1 ทดสอบการทำงานของทางเลือกเส้นทางแบบสแตติก

เมื่อจำลองเน็ตเวิร์กดังรูป



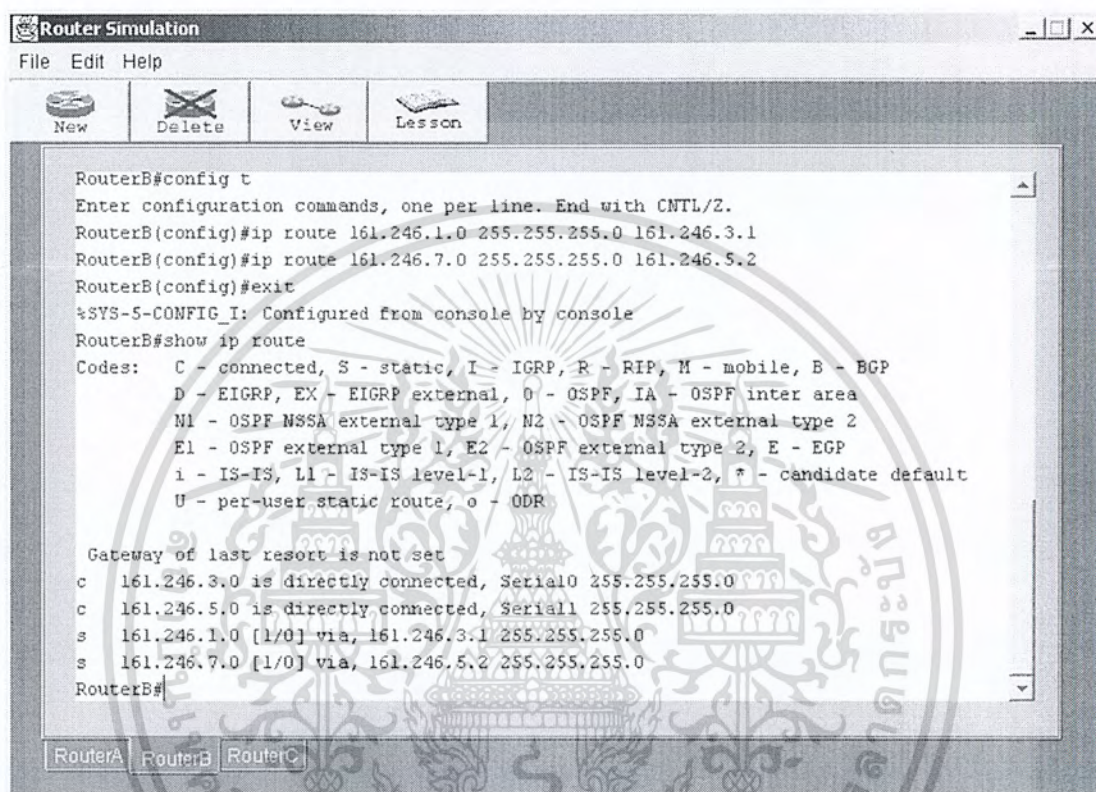
รูปที่ 8-1 ภาพจำลองเน็ตเวิร์ก

สร้างการเลือกเส้นทางแบบสแตติกโดยใช้คำสั่ง ip route



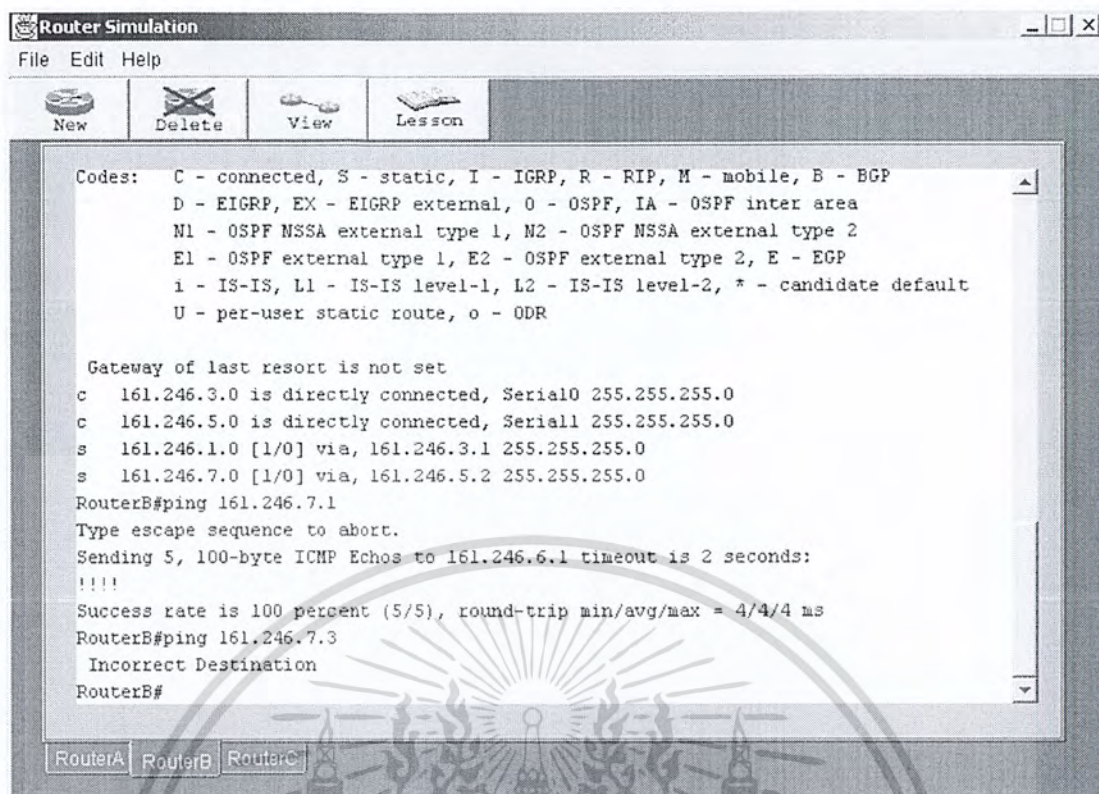
รูปที่ 8-2 เมื่อใช้เราเตอร์ B ใช้คำสั่ง ip route

ดูค่าตารางเลือกเส้นทางโดยใช้คำสั่ง show ip route



รูปที่ 8-3 เมื่อเราเตอร์ B ใช้คำสั่ง show ip route

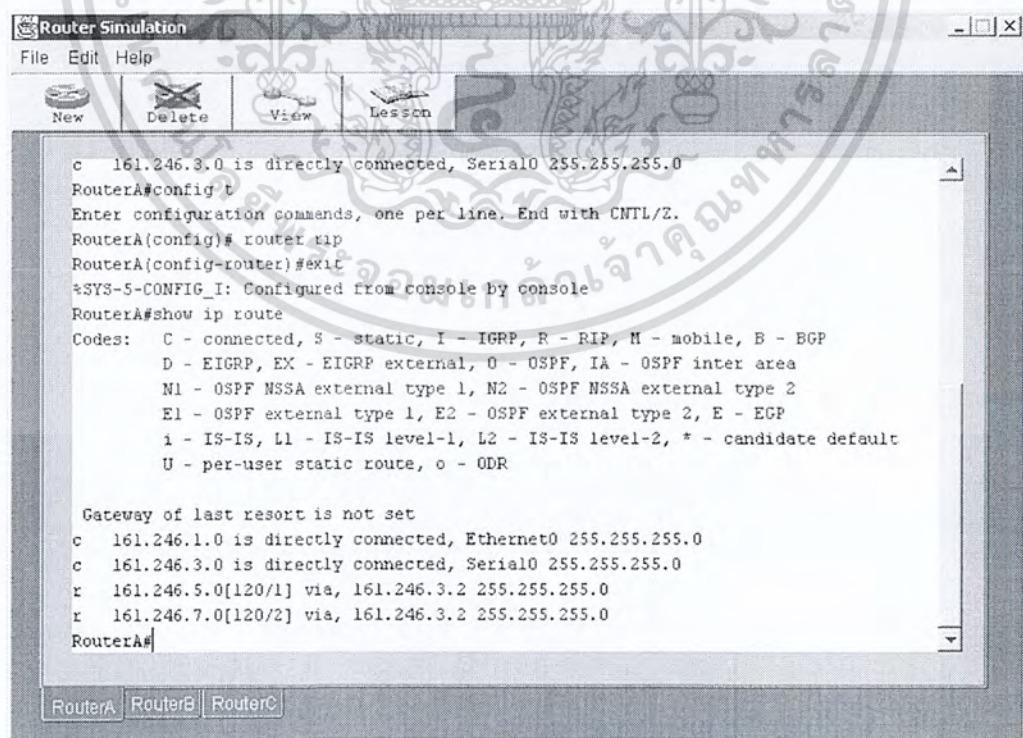
ทดสอบโดยใช้คำสั่ง ping เพื่อตรวจสอบว่าไปถึงปลายทางได้หรือไม่ ในที่นี้จะใช้เราเตอร์ B ping ไปยังไอพี 161.246.7.1 ซึ่งเป็นอินเทอร์เฟซของเราเตอร์ C ซึ่งถ้าการเชื่อมต่อถูกต้อง จะต้องสามารถ ping ไปได้ และไอพี 161.246.7.3 ซึ่งไม่มีในตารางเลือกเส้นทางดังนี้



รูปที่ 8-4 เมื่อเราเตอร์ B ใช้คำสั่ง ping

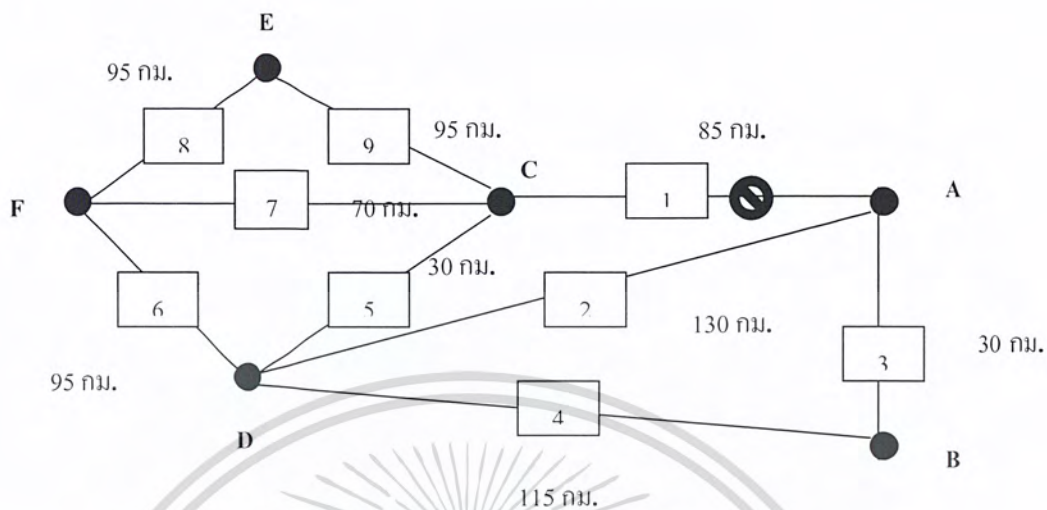
8.2 ทดสอบการทำงานของเส้นทางแบบโพรโทคอลสตาร์ไอพี

เมื่อใช้คำสั่ง router rip ที่เราเตอร์ทุกตัวและใช้คำสั่ง show ip route ที่เราเตอร์ จะได้ผลดังรูป

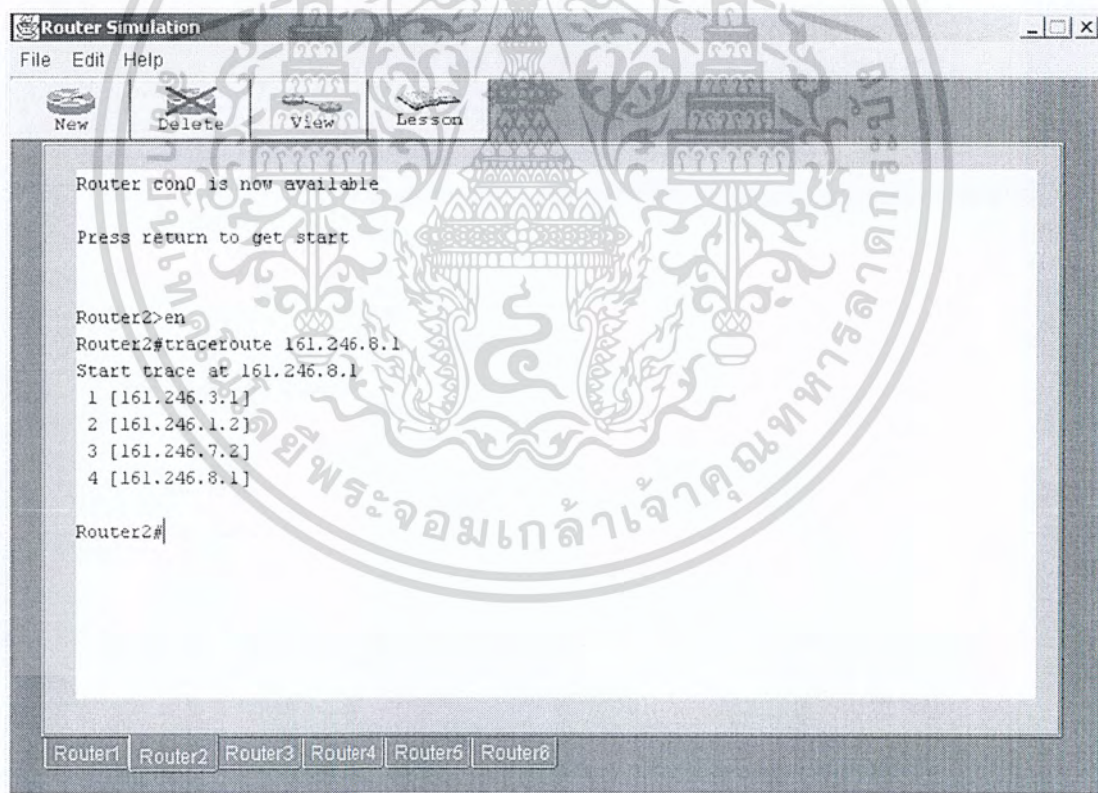


รูปที่ 8-5 เมื่อเราเตอร์ A ใช้คำสั่ง show ip route

8.3 ทดสอบการทำงานของกรเลือกเส้นทางแบบโปรโตคอลโอเอสพีเอฟ เมื่อจำลองเน็ตเวิร์กดังรูป



รูปที่ 8-6 ภาพจำลองเน็ตเวิร์ก



รูปที่ 8-7 แสดงผลจากการใช้คำสั่ง Traceroute ของโปรโตคอลโอเอสพีเอฟ

บทที่ 9

สรุปและวิจารณ์

ในการทำงานของปริญญานิพนธ์นี้ สามารถที่จะสร้างซอฟต์แวร์ที่จำลองการตั้งค่าเราเตอร์ของซิสโก้ และจำลองการทำงานของโพรโตคอลที่ใช้เลือกเส้นทางได้ ทั้งแบบเสถียรและไดนามิก (อาร์ไอพีรีและโอเอสพีเอฟ) ทำให้สามารถจำลองการทำงานของเครือข่ายได้ตั้งแต่เครือข่ายขนาดเล็กไปจนถึงเครือข่ายขนาดใหญ่ สามารถที่จะบันทึกค่าคอนฟิกลงไฟล์เก็บไว้ได้ อีกทั้งยังสามารถแสดงผลเครือข่ายจำลองออกมาเป็นภาพ 2 มิติให้สามารถมองภาพรวมของเครือข่ายได้อย่างชัดเจน พร้อมทั้งมีบทเรียนสอนเพื่อที่ผู้สนใจใช้ซอฟต์แวร์นี้สามารถทดลองใช้ตามบทเรียน

การตั้งค่าอุปกรณ์เราเตอร์มีคำสั่งมากและเราเตอร์มีการออกซิริใหม่มาเรื่อยๆ การที่ผู้ดูแลระบบจะดูแลและใช้ความสามารถของเราเตอร์ได้อย่างเต็มที่มีนั้น ผู้ดูแลจะต้องทราบถึงความสามารถในการตั้งค่าของเราเตอร์ ต้องทราบว่าเราเตอร์มีคำสั่งอะไรบ้าง แต่ถึงอย่างไรก็ตามหลักการพื้นฐานในการตั้งค่าอุปกรณ์เราเตอร์รวมทั้งคำสั่งพื้นฐานก็ไม่แตกต่างกันมากนัก การที่เราเข้าใจหลักการพื้นฐานของการตั้งค่าอุปกรณ์เราเตอร์จึงเป็นสิ่งจำเป็น

9.1 ปัญหาและอุปสรรค

- คำสั่งในการตั้งค่าเราเตอร์มีเยอะมาก ต้องใช้เวลาในการศึกษาและรวบรวมเฉพาะคำสั่งที่น่าสนใจและเป็นประโยชน์
- การเลือกเส้นทางแบบ OSPF นั้นต้องใช้ปริมาณข้อมูลที่ส่งในขณะนั้นเข้ามาเป็นองค์ประกอบในการคำนวณค่า weight ของเส้นทางด้วย แต่เนื่องจากการจำลองเราเตอร์ จึงไม่มีการส่งข้อมูลจริงๆ ทำให้ต้องกำหนดค่า weight ด้วยตัวเอง

9.2 แนวทางการประยุกต์และพัฒนา

- เพิ่มโพรโตคอล เช่น IGRP, BGP เพื่อจำลองเป็นเราเตอร์ระดับสูง โปรแกรมในขณะนี้ยังเป็น LAN Router ต้องพัฒนาต่อเป็น WAN Router
- พัฒนาให้หนึ่งโปรแกรมกลายเป็นหนึ่งเราเตอร์ แล้วนำโปรแกรมไปติดตั้งที่เครื่องอื่นๆ ที่เชื่อมต่อกันในเครือข่ายเดียวกัน ทำเป็น Virtual Router

9.3 สรุปโครงการ

โครงการนี้มีประโยชน์มาก เพราะสามารถให้ผู้ที่ต้องการฝึกการติดตั้งค่าเราเตอร์ ไม่ว่าจะเป็นผู้ที่อยู่ในระดับการศึกษา หรือใช้งานจริงก็ตาม สามารถฝึกฝนด้วยตนเองได้ โดยมีตัวอย่างการทดลองเบื้องต้นให้ฝึกฝน ซึ่งโครงการนี้นอกจากสามารถทำเพื่อการศึกษาด้านเครือข่ายคอมพิวเตอร์แล้ว ยังเป็นการลดความต้องการ การสั่งซื้ออุปกรณ์เราเตอร์ หรือซอฟต์แวร์เชิงพาณิชย์ที่จำลองการติดตั้งเราเตอร์จากต่าง

ประเทศ โครงการนี้เป็นโปรแกรมแบบมัลติแพลตฟอร์ม (Multi platform) เท่าที่สำรวจมายังไม่มีซอฟต์แวร์เชิงพาณิชย์ตัวไหนที่สามารถทำได้ และไม่มีซอฟต์แวร์เชิงพาณิชย์ตัวไหนที่สามารถใช้โปรโตคอลการเลือกเส้นทางแบบโอเอสพีเอฟได้



ภาคผนวก A

สรุปคำสั่งทั้งหมด

Bandwidth
Interface Commands
เป็นคำสั่งที่ใช้สำหรับกำหนดค่าแบนวิดให้กับอินเทอร์เฟซประเภท Serial
รูปแบบคำสั่ง Bandwidth <number>
<number> เป็นค่าขนาดของแบนวิด มีหน่วยเป็นกิโลบิต
ตัวอย่างการใช้งาน
RouterA(config)#int s0
RouterA(config-if)#bandwidth 56
RouterA(config-if)#
Cdp Timer
Global Configuration Commands
เป็นคำสั่งที่ใช้สำหรับกำหนดค่า cdp ให้กับเราเตอร์
รูปแบบคำสั่ง Cdp Timer <number>
<number> เป็นช่วงเวลาสำหรับส่งแพ็กเกจ cdp ออกไป มีค่าอยู่ระหว่าง 5-900 วินาที
ตัวอย่างการใช้งาน
RouterA(config) #cdp timer 90
RouterA(config)#
Clock Rate
Interface Commands
เป็นคำสั่งที่ใช้สำหรับกำหนดค่า clock rate ให้กับอินเทอร์เฟซประเภท Serial
รูปแบบคำสั่ง Clock Rate <number>
<number> เป็นค่าขนาดของ clock rate มีหน่วยเป็นบิต
ตัวอย่างการใช้งาน
RouterA(config)#int s0
RouterA(config-if)#clock rate 56000

Clock Set
Privileged EXEC Commands
เป็นคำสั่งที่ใช้สำหรับตั้งค่าวันเวลาให้กับเราเตอร์ รูปแบบคำสั่ง Clock Set <time> <date> <month> <year> <time> เป็นเวลาขณะนั้น ใช้รูปแบบ hh:mm:ss <date> วันที่ขณะนั้น <month> เดือนขณะนั้น <year> ปีขณะนั้น
ตัวอย่างการใช้งาน RouterA#clock set 10:30:00 14 February 2002 RouterA#
Configure Terminal
Privileged EXEC Commands
เป็นคำสั่งที่ใช้เปลี่ยนโหมดจาก privilege mode เป็น global configuration mode
ตัวอย่างการใช้งาน Router#configuration terminal Router(config)#
Copy Running-Config Startup-Config
Privileged EXEC Commands
เป็นคำสั่งที่ใช้สำหรับจัดเก็บค่าที่กำหนดให้กับเราเตอร์ในขณะนั้นลงในหน่วยความจำเพื่อเวลาเปิดโปรแกรมขึ้นมาใหม่จะได้ไม่ต้องกำหนดค่าใหม่
ตัวอย่างการใช้งาน RouterA#copy running-config startup-config RouterA#

Copy Startup-Config Running-Config
Privileged EXEC Commands
เป็นคำสั่งที่ใช้สำหรับโหลดค่าที่เก็บอยู่ในหน่วยความจำมากำหนดให้กับเราเตอร์
ตัวอย่างการใช้งาน
RouterA#copy startup-config running-config
Buliding Configuration
....
[OK]
RouterA#

Description
Interface Commands
เป็นคำสั่งที่ใช้สำหรับกำหนดรายละเอียดเพิ่มเติมให้กับอินเทอร์เฟซ
รูปแบบคำสั่ง Description <String>
<String> เป็นข้อความที่เป็นรายละเอียดเพิ่มเติมของอินเทอร์เฟซ
ตัวอย่างการใช้งาน
RouterA(config)#int e0
RouterA(config-if)#description lan link to ISAG room
RouterA(config-if)#

Disable
Privileged EXEC Commands
เป็นคำสั่งที่ใช้สำหรับเปลี่ยน โหมดจาก privilege โหมดเป็น user execution โหมด
ตัวอย่างการใช้งาน
Router#disable
Router>

Enable
User EXEC Commands
เป็นคำสั่งที่ใช้สำหรับเปลี่ยน โหมดจาก user execution เป็น privilege โหมด
ตัวอย่างการใช้งาน
Router> enable
Router#

Enable Secret
Global Configuration Commands
เป็นคำสั่งที่ใช้สำหรับกำหนดคีย์ให้ใส่พาสเวิร์ดก่อนเข้าสู่ระดับของ Privileged EXEC
รูปแบบคำสั่ง Enable Secret <Password>
<Password> เป็นคีย์เวิร์ดที่เป็นพาสเวิร์ดในการผ่าน
ตัวอย่างการใช้งาน
RouterA(config)# enable secret cisco
RouterA(config)#

Erase Startup-Config
Privileged EXEC Commands
เป็นคำสั่งที่ใช้สำหรับลบค่าที่กำหนดให้กับเราเตอร์ในหน่วยความจำ
ตัวอย่างการใช้งาน
RouterA# copy startup-config running-config
RouterA#

Exit
User EXEC Commands
เป็นคำสั่งที่ใช้สำหรับ log out ออกจากเราเตอร์
ตัวอย่างการใช้งาน
Router> exit

Exit
Global Configuration Commands, Interface Commands, Routing Engine Commands, Line Commands
เป็นคำสั่งใช้สำหรับออกจากโหมดที่กำลังใช้งานอยู่ขณะนั้น เพื่อเข้าสู่ Privileged EXEC Commands โหมด
ตัวอย่างการใช้งาน
Router(Config-if)#exit
Router#

Hostname
Global Configuration Commands
เป็นคำสั่งที่ใช้สำหรับกำหนดชื่อเราเตอร์
ตัวอย่างการใช้งาน
Router(config)#hostname RouterA
RouterA(config)#

Interface
Global Configuration Commands
เป็นคำสั่งเพื่อเข้าสู่การตั้งค่าอินเทอร์เฟซต่างๆ ของเราเตอร์ โดยคำสั่งนี้จะตามด้วยชื่อของอินเทอร์เฟซที่ต้องการตั้งค่า
รูปแบบคำสั่ง Int <interface_name>
โดย <Interface_name> เป็นชื่อของอินเทอร์เฟซ
ตัวอย่างการใช้งาน
Router(config)#int e0
Router(config-if)#

Ip Address
Interface Commands
เป็นคำสั่งที่ใช้กำหนดค่าไอพีแอดเดรสให้กับอินเทอร์เฟซที่กำลัง config อยู่
รูปแบบคำสั่ง ip address <ip_address> <netmask>
<Ip_address> เป็นค่าไอพีแอดเดรสที่ต้องการกำหนดให้กับอินเทอร์เฟซ
<Netmask> เป็นค่าเน็ตเวิร์กมาสก์ของไอพีแอดเดรส
ตัวอย่างการใช้งาน
RouterA(config)#int e0
RouterA(config-if)#ip address 161.246.5.1 255.255.255.0
RouterA(config-if)#

Ip Ospf Cost
Privileged EXEC Commands
เป็นคำสั่งที่ใช้สำหรับกำหนดค่า weight ให้กับเราเตอร์ตัวที่เชื่อมต่อกันอยู่
รูปแบบคำสั่ง Ip Ospf Cost <Router_Name> <Weight>
<Router_Name> เป็นชื่อเราเตอร์ตัวติดกันที่ต้องการใส่ค่า weight
<Weight> เป็นค่า weight ระหว่างเราเตอร์
ตัวอย่างการใช้งาน
RouterA#ip ospf cost Router2 10
RouterA#

Ip Route/No Ip Route
Global Configuration Commands
เป็นคำสั่งเพื่อกำหนดการค้นหาเส้นทางเป็นแบบสแตติกหรือยกเลิกข้อมูลการค้นหาเส้นทางแบบสแตติก
รูปแบบคำสั่ง Ip Route/No Ip Route <Destination> <Netmask> < Gateway>
<Destination> เป็นเลขเครือข่ายของเครือข่ายปลายทาง
<Netmask> เป็นค่าเน็ตเวิร์กมาสก์ของไอพีแอดเดรส
<Gateway> เป็นค่าไอพีแอดเดรสของเราเตอร์เครื่องถัดไป
ตัวอย่างการใช้งาน
RouterA(config)#ip route 161.246.10.1 255.255.255.0 161.246.5.2
RouterA(config)#

Line Auxiliary 0

Global Configuration Commands

เป็นคำสั่งที่ใช้ระบุค่าที่กำหนดต่อไปนี้เป็นกำหนดที่ auxiliary port

ตัวอย่างการใช้งาน

```
RouterA(config)#line aux 0
```

```
RouterA(config-line)#
```

Line Console 0

Global Configuration Commands

เป็นคำสั่งที่ใช้ระบุค่าที่กำหนดต่อไปนี้เป็นกำหนดที่ console port

ตัวอย่างการใช้งาน

```
RouterA(config)#line con 0
```

```
RouterA(config-line)#
```

Line vty 0 4

Global Configuration Commands

เป็นคำสั่งที่ใช้ระบุค่าที่กำหนดต่อไปนี้เป็นกำหนดที่ port telnet

ตัวอย่างการใช้งาน

```
RouterA(config)#line vty 0 4
```

```
RouterA(config-line)#
```

Login/No Login

Line Commands

เป็นคำสั่งที่ใช้สำหรับเปลี่ยนกำหนดให้เปิดพอร์ตเทลเน็ตสำหรับ Login หรือปิดพอร์ตเทลเน็ตสำหรับ No Login

ตัวอย่างการใช้งาน

```
RouterA(config)#line vty 0 4
```

```
RouterA(config-line)#login
```

```
RouterA(config-line)#
```

No Enable Secret
Global Configuration Commands
เป็นคำสั่งที่ใช้สำหรับยกเลิกการใส่พาสเวิร์ดก่อนเข้าสู่ระดับของ Privileged EXEC
ตัวอย่างการใช้งาน
RouterA(config)#no enable secret
RouterA(config)#

Password
Line Commands
เป็นคำสั่งที่ใช้สำหรับกำหนดค่าพาสเวิร์ด
รูปแบบคำสั่ง Password <Password>
<Password> เป็นคีย์เวิร์ดที่ใช้สำหรับผ่าน
ตัวอย่างการใช้งาน
RouterA(config)#line con 0
RouterA(config-line)#password 123
RouterA(config-line)#

Ping
User EXEC Commands, Privileged EXEC Commands
เป็นคำสั่งที่ใช้สำหรับตรวจสอบการเชื่อมต่อว่าสามารถไปถึงยังไอพีที่ต้องการได้หรือไม่
รูปแบบคำสั่ง Ping <Ip Address>
<Ip Address> เป็นเลขไอพีแอดเดรสของจุดหมายปลายทางที่ต้องการตรวจสอบ
ตัวอย่างการใช้งาน
RouterA>ping 161.246.5.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 161.246.5.1 timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
RouterA>

Ring-Speed
Interface Commands
เป็นคำสั่งที่ใช้สำหรับกำหนดค่า Ring Speed ให้กับอินเทอร์เฟซประเภทโทเคนริงค์ รูปแบบคำสั่ง Ring-Speed <number> <number> เป็นความเร็วของโทเคน
ตัวอย่างการใช้งาน RouterA(config)# int to0 RouterA(config-if)# ring-speed 16 RouterA(config-if)#
Router Rip/No Router Rip
Routing Engine Commands
เป็นคำสั่งเพื่อกำหนดให้เราเตอร์เครื่องนั้นใช้โปรโตคอลอาร์ไอพีในการสร้างตารางค้นหาเส้นทางสำหรับคำสั่ง router rip และยกเลิกการใช้โปรโตคอลอาร์ไอพีสำหรับคำสั่ง no router rip
ตัวอย่างการใช้งาน RouterA(config)# router rip RouterA(config-router)#
Router Ospf/No Router Ospf
Routing Engine Commands
เป็นคำสั่งเพื่อกำหนดให้เราเตอร์เครื่องนั้นใช้โปรโตคอลโอเอสพีเอฟในการสร้างตารางค้นหาเส้นทางสำหรับคำสั่ง router ospf และยกเลิกการใช้โปรโตคอลโอเอสพีเอฟสำหรับคำสั่ง no router ospf
ตัวอย่างการใช้งาน RouterA(config)# router spf RouterA(config-router)#

Show Cdp

User EXEC Commands, Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงค่าที่กำหนดให้กับการส่งแพ็กเกจ cdp

ตัวอย่างการใช้งาน

RouterA#show cdp

Global CDP Information:

Sending CDP packets every 90 seconds

Sending a holdtime value of 180 seconds

RouterA#

Show History

Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดง 10 คำสั่งล่าสุดที่เคยใช้งาน

ตัวอย่างการใช้งาน

RouterA#show history

enable

config t

int s0

ip address 161.246.1.1 255.255.255.0

no shut

int s1

ip address 161.246.2.1 255.255.255.0

no shut

int e0

ip address 161.246.3.1 255.255.255.0

no shut

exit

exit

RouterA#

Show Interface

Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงรายละเอียดของอินเทอร์เฟซ

รูปแบบคำสั่ง Show Interface <interface name>

<interface name> เป็นตัวเลือกจะใส่หรือไม่ใส่ก็ได้ ถ้าใส่จะเป็นการระบุให้แสดงรายละเอียดเฉพาะอินเทอร์เฟซที่ระบุ ถ้าไม่เช่นนั้นจะแสดงทุกอินเทอร์เฟซ

ตัวอย่างการใช้งาน

RouterA#show int e0

Ethernet0 is up, line protocol is up

Hardware is Lance

Internet address is 161.246.1.1/24

MTU 1500 bytes, BW 56 Kbit, DLY 1000 usec, rely 255/255, load 1/255

Encapsulation ARPA, loopback not set, keepalive set (10 sec)

ARP type: ARPA, ARP Timeout 04:00:00

Last input never, output 00:00:07, output hang never

Last clearing of "show interface" counters never

Queueing strategy: fifo

Output queue 0/40, 0 drops; input queue 0/75, 0 drops

5 minute input rate 0 bits/sec, 0 packets/sec

5 minute output rate 0 bits/sec, 0 packets/sec

0 packets input, 0 bytes, 0 no buffer

Received 0 broadcasts, 0 runts, 0 giants, 0 throttles

0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort

0 input packets with dribble condition detected

472 packets output, 47379 bytes, 0 underruns

0 output errors, 0 collisions, 2 interface resets

0 babbles, 0 late collision, 0 deferred

0 lost carrier, 0 no carrier

0 output buffer failures, 0 output buffers swapped out

RouterA#

Show Ip Protocol

Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงโปรโตคอลที่เราเตอร์ใช้งานอยู่ขณะนั้น

ตัวอย่างการใช้งาน

RouterA#**show ip protocol**

Routing Protocol is "rip"

Sending updates every 30 seconds, hold down 180, flushed after 240

Outgoing update filter list for all interfaces is not set

Incoming update filter list for all interfaces is not set

Redistributing : rip

Routing for Networks :

161.246.0.0

Routing Information Sources :

Gateway	Distance	Last Update
161.246.0.0	120	0:00:00

Distance: (default is 120)

RouterA#

Show Ip Route

Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงตารางการค้นหาเส้นทาง

รูปแบบคำสั่ง Show Ip Route <route_type>

<route_type> เป็นตัวเลือกที่จะใส่หรือไม่ใส่ก็ได้ ถ้าใส่จะเป็นการระบุว่าการดูเฉพาะตารางค้นหาเส้นทางประเภทใด โดย

s : สเตติก

r : อาร์ไอพี

o : โอเอสพีเอฟ

ตัวอย่างการใช้งาน

RouterA#show ip route

Codes: C – connected, S - static, I - IGRP, R - RIP, M – mobile, B – BGP

D – EIGRP, EX - EIGRP external, O - OSPF, IA – OSPF inter area

N1 – OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 – OSPF external type 1, E2 - OSPF external type 2, E – EGP

i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

U - per-user static route, o - ODR \n\n Gateway of last resort is not set

c 161.246.1.0 is directly connected, Ethernet0

c 161.246.2.0 is directly connected, Serial0

s 161.246.7.0 [1/0] via 161.246.2.1

r 161.246.5.0 [120/2] via 161.246.1.1 255.255.255.0, Ethernet0

r 161.246.3.0 [120/1] via 161.246.2.1 255.255.255.0, Serial0

RouterA#

Show Running-Config

Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงการตั้งค่าของเราเตอร์นั้น

ตัวอย่างการใช้งาน

RouterA#show running-config

version 11.3

service timestamps debug uptime

service uimstamps log upiom

no service password-encryption

!

hostname Router1

!

interface e0

ip address 161.246.1.1 255.255.255.0

no shutdown

```
!  
no ip classless  
!  
line con 0  
line aux 0  
line vty 0 4  
!  
end  
  
RouterA#
```

Show Startup-Config

Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงค่าที่กำหนดให้กับเราเตอร์สำหรับตอนเปิดโปรแกรม

ตัวอย่างการใช้งาน

```
RouterA#show startup-config  
version 11.3  
service timestamps debug uptime  
service uimestamps log uption  
no service password-encryption  
!  
hostname Router1  
!  
interface e0  
ip address 161.246.1.1 255.255.255.0  
no shutdown  
!  
no ip classless  
!
```

```
line con 0
line aux 0
line vty 0 4
!
end
RouterA#
```

Show Version

Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงรายละเอียดของซอฟต์แวร์ที่ใช้งานกับเราเตอร์

ตัวอย่างการใช้งาน

```
RouterA#show version
Cisco Internetwork Operating System Software
IOS(tm) 3600 Software (C3640-J-M), Version 11.2(6)P, SHARED PLATFORM, RELEASE SOFTWARE
(fc1)
Copy right(c) 1986-1977 by cisco Systems, Inc.
Compiled Mon 12-May-97 15:07 by tej
Image text-base:0x600088A0, data-base: 0x6075C000

ROM: System Bootstrap, Version 1.1(7)AX SOFTWARE

ROM: System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE
BOOTFLASH: 3000 Bootstrap Software (IGS-RXBOOT), Version 10.2(8a), RELEASE SOFTW
ARE (fc1)

RouterA uptime is 11 minutes
System restarted by power-on
System image file is \"flash:c2500-d-l_113-5.bin\", booted via flash
```

Bridging software.
X.25 software, Version 3.0.0.
2 Ethernet/IEEE 802.3 interface(s)
4 Serial network interface(s)
32K bytes of non-volatile configuration memory.
8192K bytes of processor board System flash (Read ONLY)

Configuration register is 0x2102
RouterA#

Shutdown/No Shutdown

Interface Commands

เป็นคำสั่งเปลี่ยนสถานะของอินเทอร์เฟซของเราเตอร์ ถ้าต้องการให้อินเทอร์เฟซมีสถานะเป็นเปิดจะใช้คำสั่ง
No Shutdown และในทางกลับกันใช้คำสั่ง Shutdown เพื่อเปลี่ยนสถานะของอินเทอร์เฟซเป็นปิด

ตัวอย่างการใช้งาน

RouterA(config)#int e0
RouterA(config-if)#no shut
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0, changed state to up
%LINK-3-UPDOWN: Interface Ethernet0, changed state to up

Telnet

Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับ telnet ไปยังเราเตอร์เครื่องอื่นเพื่อตั้งค่าให้กับเราเตอร์นั้น
รูปแบบคำสั่ง Telnet <Ip Address>
<Ip Address> เป็นเลขไอพีแอดเดรสของอินเทอร์เฟซของเราเตอร์เครื่องที่ต้องการเข้าไปกำหนดค่า

ตัวอย่างการใช้งาน

RouterA#telnet 161.246.5.1
Router2#

Traceroute
User EXEC Commands, Privileged EXEC Commands
เป็นคำสั่งที่ใช้สำหรับดูเส้นทางที่เดินทางไปยังไอพีที่ต้องการตรวจสอบ
รูปแบบคำสั่ง Traceroute <Ip Address>
<Ip Address> เป็นเลขไอพีแอดเดรสของจุดหมายปลายทางที่ต้องการตรวจสอบ
ตัวอย่างการใช้งาน
RouterA>traceroute 161.246.5.1
1 [161.246.1.1]
2 [161.246.7.1]
3 [161.246.5.1]
RouterA>



บรรณานุกรม

- [1] Todd Lammle,Donald porter with James Chellis: "CCNA Cisco Certified Network Associate",SYBEX,page 1-729,1997
- [2] Todd Lammle,"CCNA Cisco Certified Network Associate Study Guide second Edition",SYBEX,page 1-716,2000
- [3] สุรศักดิ์ สงวนพงษ์, "สถาปัตยกรรมและโพรโตคอลที่ซีพี/ไอพี",ซีเอ็ดยูเคชั่น,หน้า 1-528,2543
- [4] <http://www.java.sun.com>
- [5] <http://www.cisco.com>
- [6] <http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/>
- [7] http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/ip_c/ipcprt2/1cdrip.htm
- [8] <http://www.geocities.com/Heartland/4394/work/ospf.html>
- [9] http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fipr_c/ipcprt2/1cfospf.htm
- [10] <http://www.cisco.com/warp/public/104/14.html>
- [11] <http://rfc.sunsite.dk/rfc/rfc1793.html>
- [12] <http://community.roxen.com/developers/idoocs/rfc/rfc1247.html>
- [13] <http://www.cisco.com/warp/public/104/16.html>
- [14] <http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/Dijkstra.shtml.en>
- [15] <http://www.math.grin.edu/~rebelsky/Courses/CS152/98S/Outlines/outline.50.html#sp-dijkstra-code>
- [16] <http://www.whatis.com>
- [17] <http://www.theparticle.com/javadata2.html#Trees>
- [18] <http://carnap.ss.uci.edu/java/dijkstra/DijkstraApplet.html>
- [19] <http://www.dcs.qmul.ac.uk/~pablo/coursework/test.html>