สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

โปรแกรมจำลองการทำงานเราเตอร์

Router Simulator

นายชำนาญ เพียผิว

นายอนุชา กล้าผจญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

เลขหมู่.....
เลขทะเบียน 49970
วัน,เดือน,ปี 16 มิ.ย. 2547

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

ลิขสิทธิ์ของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น ห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมจำลองการทำงานเราเตอร์

Router Simulator

โดย

นายชำนาญ เพียผิว

นายอนุชา กล้าผจญ

อาจารย์ที่ปรึกษา

อาจารย์ ธนา หงษ์สุวรรณ

อาจารย์ อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมจำลองการทำงานเราเตอร์

Router Simulator

ผู้จัดทำ

1. นายชำนาญ เพียผิว รหัสประจำตัว 43015356
2. นายอนุชา กล้าผจญ รหัสประจำตัว 43015395

_____ อาจารย์ที่ปรึกษา

(อาจารย์ธนา หงษ์สุวรรณ)

_____ อาจารย์ที่ปรึกษา

(อาจารย์อัครเดช วัชระภูพงษ์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมจำลองการทำงานเราเตอร์

นายชำนาญ เพียผิว 43015356
นายอนุชา กล้าผจญ 43015395
อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

ปัจจุบันระบบเครือข่ายมีความสำคัญมากขึ้น แต่การติดตั้งระบบเครือข่ายยังคงเป็นไปอย่างไม่ง่ายนัก รวมทั้งอุปกรณ์ในการเชื่อมต่อยังคงมีราคาแพง ทำให้โอกาสในการฝึกหัดใช้ให้เกิดความคุ้นเคยมีน้อยลง มีผลให้ผู้ที่มีความสามารถและความชำนาญในการตั้งค่าเราเตอร์มีน้อยไปด้วย ทั้งๆ ที่ความต้องการมีมาก

โครงงานนี้เป็นโครงงานที่พัฒนาต่อจากโครงงานโปรแกรมจำลองการตั้งค่าอุปกรณ์เราเตอร์ปีที่แล้วเพื่อจำลองการตั้งค่าเราเตอร์ของบริษัทซิสโก้ มีความสามารถเพิ่มจากปีที่แล้วคือ ออกแบบโปรแกรมเป็นแบบ ไคลเอ็นต์/เซิร์ฟเวอร์ ซึ่งสามารถรองรับการทำงานสำหรับผู้ใช้หลายคน และสามารถใช้งานบนเครื่องคอมพิวเตอร์บุคคลได้ เป็นโปรแกรมที่มีความสามารถจำลองการตั้งค่าเราเตอร์ได้หลายๆ เครื่อง และสามารถตรวจสอบความถูกต้องของการตั้งค่าได้ด้วย รองรับโพรโตคอลในการเลือกเส้นทางได้ 2 โพรโตคอลคือ อาร์ไอพีและโอเอสพีเอฟผู้ใช้สามารถสร้างเครือข่ายของเราเตอร์ขึ้นมาเองได้เพื่อให้ศึกษาภาพรวมของระบบและยังสามารถกำหนดลักษณะของทราฟฟิกบนเครือข่ายได้ โดยใช้หลักการของ Object Oriented Programming และ Remote Method Invocation (RMI) พัฒนาโดยภาษาจาวา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Router Simulator

Mr. Chumana Pearpiw
Mr. Anucha Klaphachon
Mr. Thana Hongsuwan Advisor
Mr. Akkradach Watcharapupong Advisor

Abstract

Nowadays, computer network has been important for our life than before. Since, network systems configuration is rather difficult to work. In addition, the price of network's device 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 developed from the project's last year to assume that we can setup Cisco Routers. This program is designated for Client and Server which is able to support both Singleuser and Multiuser. In addition, this program can also deal with 2 ways of protocol which are RIP and OSPF. Users are able to create network view for a network system overview and assign traffic's shape for network base on Object Oriented Programming , Remote Method Invocation and Java Application.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะไม่สามารถเสร็จสมบูรณ์ได้ถ้าไม่ได้รับความช่วยเหลือและการร่วมมือของบุคคลหลายๆ ฝ่ายด้วยกัน โดยเฉพาะอย่างยิ่งบุคคลผู้ซึ่งเป็นผู้จุดประกายความคิดให้เกิดหัวข้อปริญญานิพนธ์นี้ขึ้นมา นั่นก็คืออาจารย์ธนา หงษ์สุวรรณ และอาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่คอยให้คำปรึกษาปริญญานิพนธ์ตลอดเวลา และขอบพระคุณคณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้ให้คำแนะนำ และความรู้ทางด้านคอมพิวเตอร์

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์โดยเฉพาะห้องวิจัยและพัฒนาการรักษาความปลอดภัยข้อมูล (ISAG) ที่ได้เอื้อเฟื้อสถานที่ ให้คณะผู้จัดทำได้ทำการวิจัย และช่วยอำนวยความสะดวกต่างๆขอขอบคุณพี่เก๋,พี่เดียร์ และเพื่อนๆ พี่ๆ น้องๆ ชาว ISAG ที่คอยให้ความช่วยเหลือในการทำงานตลอดเวลาเป็นที่ปรึกษายามมีปัญหา รวมทั้งให้ยืมทรัพยากรที่จำเป็นต่างๆ

ที่สำคัญและขาดมิได้ คือ ต้องขอขอบพระคุณบิดา มารดาที่ได้ให้กำเนิด คอยสั่งสอน และให้การสนับสนุนการศึกษา กิจกรรมต่างๆ นับเป็นพระคุณที่หาใดเปรียบมิได้ ทางคณะผู้จัดทำขอกราบขอบพระคุณมา ณ ที่นี้ด้วย

คณะผู้จัดทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

	หน้าที่
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญภาพประกอบ	VII
สารบัญตาราง	X
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์ของปริญญานิพนธ์	1
1.3 ขอบเขตของปริญญานิพนธ์	1
1.4 ขั้นตอนการดำเนินงาน	2
บทที่ 2 ทีซีพี/ไอพีและอินเทอร์เน็ต	3
2.1 การเชื่อมโยงเครือข่าย	3
2.2 ความหมายของโพรโตคอล	3
2.3 สถาปัตยกรรมของอินเทอร์เน็ตและความหมายของเราเตอร์	3
2.4.สถาปัตยกรรมทีซีพี/ไอพี	4
บทที่ 3 เทคโนโลยีเครือข่ายคอมพิวเตอร์	9
3.1 เครือข่ายท้องถิ่นตามข้อกำหนดของ IEEE	9
3.1.1 IEEE 802.3 (Ethernet)	10
3.1.2 IEEE 802.4 (Token Bus)	11
3.1.3 IEEE 802.5 (Token Ring)	11
3.1.4 เอฟดีดีไอ(FDDI)	12
3.2 อุปกรณ์เครือข่าย	13
3.2.1 ฮับ	13
3.2.2 บริดจ์	14
3.2.3 เราเตอร์	16
บทที่ 4 ไอพีแอดเดรส	17
4.1 ไอพีแอดเดรส	17
4.1.1 ความสำคัญของเลขเครือข่ายและเลขโฮสต์	18
4.1.2 การจัดคลาสเครือข่าย	18
4.1.3 ลักษณะสำคัญของแต่ละคลาส	19

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

	หน้าที่
4.2 การแบ่งเครือข่ายย่อย	21
4.2.1 ซับเน็ตมาสก์	22
4.2.2 ดีฟอลต์ซับเน็ตมาสก์	22
4.2.3 การเลือกเส้นทางในซับเน็ต	23
บทที่ 5 โพรโตคอลเลือกเส้นทาง (Routing Protocol)	24
5.1 การเลือกเส้นทาง (Routing)	24
5.2 ตารางเส้นทาง (Routing Table)	24
5.3 ประเภทของการเลือกเส้นทาง	25
5.4 อาร์ไอพี(RIP)	28
5.4.1 การทำงานของอาร์ไอพี	29
5.4.2 การปรับค่าเมื่อเครือข่ายเปลี่ยน	32
บทที่ 6 โอเอสพีเอฟ (OSPF : Open Shortest Path First)	41
6.1 การทำงานของโอเอสพีเอฟโดยสรุป	41
6.2 เรียนรู้เครือข่าย	42
6.3 สร้างฐานข้อมูลเส้นทาง	42
6.4 การคำนวณเส้นทาง	43
6.4.1 ขั้นตอนวิธีของไดจ์สตรา (Dijkstra Algorithm)	43
6.4.2 การปรับเปลี่ยนเมื่อเครือข่ายเปลี่ยน	45
6.5 การจัดองค์ประกอบย่อย	46
6.6 การแบ่งระบบออโตโนมัส	46
บทที่ 7 แอคเซสลิสต์ (Access list)	47
7.1 พื้นฐานของแอคเซสลิสต์	48
7.2 ประเภทของแอคเซสลิสต์	49
7.3 การแก้ไขค่าแอคเซสลิสต์	51
7.4 สแตนดาร์ด ไอพี แอคเซสลิสต์(Standard IP Access Lists)	52
7.5 เอกซ์เทนเด็ด ไอพี แอคเซสลิสต์(Extended IP Access Lists)	53
7.6 ทีซีพี แอคเซสลิสต์(TCP Access List)	54
7.7 ยูดีพี แอคเซสลิสต์(UDP Access List)	55
7.8 ไอซีเอ็มพี แอคเซสลิสต์(ICMP Access List)	55
7.9 การเรียกใช้งานแอคเซสลิสต์	56
บทที่ 8 การออกแบบและการสร้างโปรแกรม	57

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้าที่

8.1 โครงสร้างของโปรแกรม 57
8.2 การจัดการส่วนของไคลเอ็นต์ 61
8.2.1 ส่วนของการเลือกโหมดการทำงาน 61
8.2.2 การทำงานในแต่ละโหมด 64
8.3 ส่วนของอาร์เอ็มไอ(RMI) 66
8.4 ส่วนของเซิร์ฟเวอร์ 68
8.4.1 ส่วนจัดการCommand line และคำสั่งพื้นฐาน 68
8.4.2 การจัดการส่วนของการเพิ่มเราเตอร์ 71
8.4.3 การจัดการเกี่ยวกับการเพิ่มสายเชื่อมต่อ 72
8.4.4 การจัดการส่วนของอินเตอร์เฟส 73
8.5 ฟังก์ชันกำหนดทราฟฟิกบนเครือข่าย 74
8.6 การจัดการเกี่ยวกับโพรโตคอลอาร์ไอพี 75
8.7 การจัดการเกี่ยวกับโพรโตคอลโอเอสพีเอฟ 77
บทที่ 9 ตัวอย่างและการทดสอบการทำงานของโปรแกรม 79
9.1 ตัวอย่างการเลือกเส้นทางแบบสแตติก 79
9.2 ตัวอย่างการทำงานของโพรโตคอลอาร์ไอพี 84
9.3 ตัวอย่างการทำงานของโพรโตคอลโอเอสพีเอฟ 87
9.4 ตัวอย่างการทำงานแบบผู้ใช้หลายคน 89
บทที่ 10 สรุปและวิจารณ์ 94
10.1 ปัญหาและอุปสรรค 94
10.2 ขอบเขตและข้อจำกัดของโครงงาน 94
10.3 แนวทางการประยุกต์และพัฒนา 95
10.4 สรุปโครงงาน 95
ภาคผนวก ก สรุปคำสั่งทั้งหมด 96
ภาคผนวก ข รายละเอียดของเราเตอร์แต่ละรุ่น 113
บรรณานุกรม 117

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพประกอบ

หน้าที่

รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของโอเอสไอกับเลเยอร์ของทีซีพี/ไอพี 4
รูปที่ 2-2 แสดงข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี 5
รูปที่ 3-1 เป็นแบบจำลองบางส่วนของมาตรฐานซึ่งเป็นที่รู้จักแพร่หลาย 9
รูปที่ 3-2 อีเทอร์เน็ตประเภทต่างๆ 11
รูปที่ 3-3 การทำงานโทเค็นริง 12
รูปที่ 3-4 โทโปโลยีเอฟดีดีไอ 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 19
รูปที่ 4-4 การแบ่งคลาส D และ E 19
รูปที่ 4-5 ตัวอย่างการแบ่งเครือข่ายย่อยของ 158.105 21
รูปที่ 4-6 การตรวจหาแอดเดรสซับเน็ตเพื่อเลือกเส้นทาง 23
รูปที่ 5-1 เครือข่ายการแสดงการเลือกเส้นทางแบบสแตติก 26
รูปที่ 5-2 เครือข่ายสาธิตการทำงานของอาร์ไอพี 29
รูปที่ 5-3 การประกาศค่าและปรับค่าตารางของอาร์ไอพี 31
รูปที่ 5-4 การนับเข้าสู่อนันต์ 32
รูปที่ 5-5 การประกาศเส้นทางด้วยวิธีสปลิตฮอไรซัน 34
รูปที่ 5-6 การประกาศเส้นทางด้วยวิธีสปลิตฮอไรซันแบบพอยซันรีเวอร์ส 35
รูปที่ 5-7 เครือข่ายที่จำเป็นต้องใช้โฮลดาวน์ร่วมกับทริกเกอร์อัพเดต 36
รูปที่ 5-8 ฟอร์แมตของเฟรมอาร์ไอพี 37
รูปที่ 5-9 ฟอร์แมตของเฟรมอาร์ไอพีเวอร์ชัน 2 39
รูปที่ 5-10 เฟรมอาร์ไอพีเวอร์ชัน 2 เมื่อใช้การพิสูจน์ตัวจริง 40
รูปที่ 6-1 แผนที่ทางหลวง 41
รูปที่ 6-2 เราเตอร์ A ส่งแพ็กเก็ตทักทาย 42
รูปที่ 6-3 การคำนวณหาเส้นทางสั้นที่สุดตามขั้นตอนวิธีของไดจ์สตรา 44
รูปที่ 6-4 ปัญหาเส้นทางขัดข้อง 45
รูปที่ 6-5 เส้นทางใหม่ที่คำนวณได้ 45
รูปที่ 6-6 เราเตอร์ขอบปลายของระบบออโตโนมัส 46

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพประกอบ(ต่อ)

หน้าที่

รูปที่ 7-1 การนำแอคเซสลิสต์ไปใช้งาน 47
รูปที่ 7-2 ลำดับการทำงานของแอคเซสลิสต์ 48
รูปที่ 7-3 ตัวอย่างการจัดการกับแพ็กเก็ตที่ไม่ตรงในแต่ละลำดับชั้น 49
รูปที่ 7-4 ตัวอย่างการคอนฟิกลำดับชั้นของแอคเซสลิสต์ที่ถูกต้องและไม่ถูกต้อง 50
รูปที่ 8-1 การออกแบบโปรแกรม 57
รูปที่ 8-2 คลาสไดอะแกรมของโปรแกรมส่วนไคลเอ็นต์ 58
รูปที่ 8-3 คลาสไดอะแกรมของโปรแกรมส่วนเซิร์ฟเวอร์ 59
รูปที่ 8-4 แสดงลำดับการทำงานของส่วนไคลเอ็นต์ 61
รูปที่ 8-5 หน้าจอสำหรับเลือกโหมดการทำงานของเราเตอร์ 61
รูปที่ 8-6 ตัวอย่างหน้าจอโหมดการทำงานของผู้ดูแลระบบ 62
รูปที่ 8-7 ตัวอย่างหน้าจอโหมดการทำงานของผู้ใช้งาน 62
รูปที่ 8-8 หน้าจอหลักของโปรแกรม 64
รูปที่ 8-9 ภาพแสดงคอนโซล 66
รูปที่ 8-10 สถาปัตยกรรมของอาร์เอ็มไอ 66
รูปที่ 8-11แสดงขั้นตอนการจัดการกับส่วน Command Line 69
รูปที่ 8-12 แสดงขั้นตอนการทำงานของการจัดการการเพิ่มเราเตอร์ 71
รูปที่ 8-13 แสดงขั้นตอนการทำงานของการจัดการการเพิ่มสายเชื่อมต่อ 72
รูปที่ 8-14 แสดงขั้นตอนการจัดการอินเตอร์เฟสของเราเตอร์ 73
รูปที่ 8-15 แสดงรูปไดอะล็อกกำหนดทราฟฟิกบนเครือข่าย 75
รูปที่ 8-16 ขั้นตอนการทำงานของโพรโตคอลอาร์ไอพี 76
รูปที่ 8-17 ขั้นตอนการทำงานของโพรโตคอลโอเอสพีเอฟ 78
รูปที่ 9-1 รูปแสดงเครือข่ายเราเตอร์ 79
รูปที่ 9-2 การคอนฟิกการเลือกเส้นทางแบบสแตติกของเราเตอร์ R1 80
รูปที่ 9-3 การคอนฟิกการเลือกเส้นทางแบบสแตติกของเราเตอร์ R2 80
รูปที่ 9-4 การคอนฟิกการเลือกเส้นทางแบบสแตติกของเราเตอร์ R3 81
รูปที่ 9-5 การคอนฟิกการเลือกเส้นทางแบบสแตติกของเราเตอร์ R4 81
รูปที่ 9-6 การแสดงเส้นทางแบบสแตติกของเราเตอร์ R1 82
รูปที่ 9-7 การแสดงเส้นทางแบบสแตติกของเราเตอร์ R2 82
รูปที่ 9-8 การแสดงเส้นทางแบบสแตติกของเราเตอร์ R3 83
รูปที่ 9-9 การแสดงเส้นทางแบบสแตติกของเราเตอร์ R4 83
รูปที่ 9-10 ผลการใช้คำสั่ง ping 84
รูปที่ 9-11 การคอนฟิกโพรโตคอลอาร์ไอพีของเราเตอร์ R1 85

เอกสารนี้ใช้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพประกอบ(ต่อ)

หน้าที่

รูปที่ 9-12 การคอนฟิกโพรโตคอลอาร์ไอพีของเราเตอร์ R2 85
รูปที่ 9-13 การคอนฟิกโพรโตคอลอาร์ไอพีของเราเตอร์ R3 86
รูปที่ 9-14 การคอนฟิกโพรโตคอลอาร์ไอพีของเราเตอร์ R4 86
รูปที่ 9-15 ผลการใช้คำสั่ง ping 87
รูปที่ 9-16 รูปเครือข่ายโอเอสพีเอฟ 87
รูปที่ 9-17 รูปแสดงเส้นทางจากเราเตอร์ R1 ไปยังเราเตอร์ R3 88
รูปที่ 9-18 ผลการใช้คำสั่ง traceroute 89
รูปที่ 9-19 รูปเครือข่ายที่ผู้ดูแลการทดลองสร้างขึ้น 90
รูปที่ 9-20 รูปเครือข่ายที่ผู้ทำการทดลองโหลดมาจากเซิร์ฟเวอร์ 90
รูปที่ 9-21 หน้าจอคอนโซลของ User1 91
รูปที่ 9-22 หน้าจอคอนโซลของ User2 91
รูปที่ 9-23 หน้าจอคอนโซลของ User3 92
รูปที่ 9-24 แสดงผลลัพธ์ของคำสั่ง ping จาก Router1 ไปยัง Router3 เมื่อทำการคอนฟิกได้ถูกต้อง 92
รูปที่ 9-25 แสดงการคอนฟิกให้อินเตอร์เฟส S1 ของ Router2 มีสถานะเป็น Down 93
รูปที่ 9-26 แสดงผลลัพธ์ของคำสั่ง ping จาก Router1 ไปยัง Router3 เมื่อทำการคอนฟิกไม่ถูกต้อง 93

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

หน้าที่

ตารางที่ 4-1 การจัดแบ่งเครือข่าย 158.108 ด้วยซับเน็ต 8 บิต 21
ตารางที่ 4-2 ค่าดีฟอลต์ซับเน็ตมาสก์ 23
ตารางที่ 6-1 ตารางเส้นทางที่เราเตอร์ A 42
ตารางที่ 6-2 ฐานข้อมูลลิงสเทตประจำแต่ละเราเตอร์ 43
ตารางที่ 7-1 รูปแบบตัวเลขแอคเซสลิสต์ของซิสโก้ 51

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ปัจจุบันระบบเครือข่ายคอมพิวเตอร์มีความสำคัญมากขึ้น ความต้องการในการนำคอมพิวเตอร์ไปใช้ในการติดต่อสื่อสารหรือเชื่อมต่อเพื่อให้บริการแก่กันก็ย่อมมากขึ้นตามไปด้วย ในขณะที่การเชื่อมต่อระบบเน็ตเวิร์กเป็นไปอย่างไม่ง่ายนัก เนื่องมาจากการขาดความรู้และประสบการณ์ รวมทั้งอุปกรณ์ที่เกี่ยวกับระบบเครือข่ายยังมีราคาแพง ทำให้โอกาสในการฝึกหัดใช้ให้เกิดความคุ้นเคยมีน้อยลง การเชื่อมต่อระบบใดๆนั้นต้องมีทักษะในการทำงาน โดยเฉพาะการกำหนดค่าต่างๆให้กับเราเตอร์นั้นเป็นสิ่งที่ไม่ยากเพียงแต่คนโดยทั่วไปไม่ค่อยได้สัมผัสกับเราเตอร์จริงๆ ทำให้การกำหนดค่าเป็นไปด้วยความยากลำบากและบางครั้งไม่สามารถใช้เราเตอร์ได้ตามความสามารถจริงๆ ของมัน ซึ่งเป็นเรื่องน่าเสียดายอย่างยิ่ง

ดังนั้นการจัดทำโปรแกรมจำลองการตั้งค่าเราเตอร์ จึงเป็นแนวคิดที่ต้องการเปิดโอกาสทางการศึกษาให้กับบุคคลทั่วไปที่มีความสนใจทางด้านการตั้งค่าเราเตอร์แต่ยังไม่มีความชำนาญพอที่จะตั้งค่ากับเราเตอร์จริงๆ หรือไม่มีเราเตอร์จริงๆให้ลองใช้ ได้มีโอกาสได้ทดลองตั้งค่าอุปกรณ์เราเตอร์ของซิสโก้ เพื่อเป็นแนวทางในการตั้งค่าเราเตอร์ต่อไป

1.2 วัตถุประสงค์ของปริญญานิพนธ์

ปริญญานิพนธ์นี้จัดทำขึ้นเพื่อสร้างโปรแกรมจำลองการทำงานเราเตอร์ที่มีความสามารถเพิ่มจากปีที่ผ่านมา ได้แก่

(1) เพื่อสร้างโปรแกรมจำลองการตั้งค่าเราเตอร์ที่สามารถใช้งานแบบ ไคลเอ็นต์/เซิร์ฟเวอร์

(2) เพื่อสร้างโปรแกรมจำลองการตั้งค่าเราเตอร์ที่สามารถใช้งานคำสั่งในการตั้งค่าเราเตอร์ได้เพิ่มมากขึ้น

(3) เพื่อประยุกต์โปรแกรมใช้เป็นการทดลอง ประกอบการเรียนการสอนในรายวิชาที่เกี่ยวกับการตั้งเราเตอร์และโพรโตคอลหาเส้นทางแบบ โอเอสพีเอฟและ อาร์ไอพีได้

1.3 ขอบเขตของปริญญานิพนธ์

(1) โปรแกรมที่สามารถจำลองการติดตั้งเราเตอร์ได้

(2) สามารถทำการประมวลผลการทำงานของโพรโตคอลหาเส้นทางแบบต่างๆได้ 2 โพรโตคอลคือ โอเอสพีเอฟและ อาร์ไอพี

(3) จำลองการทำงานรูปแบบการตั้งค่าเราเตอร์ ของเราเตอร์ซิสโก้รุ่น 2500

(4) โปรแกรมสามารถใช้งานแบบไคล์เอ็นต์/เซิร์ฟเวอร์ได้

1.4 ขั้นตอนการดำเนินงาน

(1) ศึกษาและรวบรวมคำสั่งที่ใช้ในการตั้งค่าอุปกรณ์เราเตอร์เพิ่มเติม

เอกสารนี้ใช้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น รวมทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเอกสารทุกครั้งที่มีการนำไปใช้

(2) ศึกษาการทำงานแบบไคลเอ็นต์/เซิร์ฟเวอร์โดยใช้อาร์เอ็มไอ

(3) ออกแบบคลาสและกำหนดรายละเอียด (Specification) ของโปรแกรมทั้งหมดโดยแยกออกเป็นส่วนของไคลเอ็นต์ และส่วนของเซิร์ฟเวอร์

(4) เขียนโปรแกรมส่วนติดต่อกับผู้ใช้

(5) เขียนโปรแกรมในส่วนของเซิร์ฟเวอร์โดยรวมฟังก์ชันการทำงานทั้งหมดของเราเตอร์

(6) ทำบทเรียนสอนการใช้งาน

(7) ทดสอบโปรแกรมและตรวจสอบข้อผิดพลาด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทีซีพี/ไอพีและอินเทอร์เน็ต

ทีซีพี/ไอพีเป็นโพรโตคอลที่ได้รับการออกแบบให้เป็นอิสระจากชนิดคอมพิวเตอร์และระบบปฏิบัติการ ตัวโพรโตคอลมีความเชื่อถือได้สูงและสามารถปรับเปลี่ยนการทำงานตามสภาพเครือข่ายได้ในกรณีที่บางเส้นทางชำรุด

2.1 การเชื่อมโยงเครือข่าย

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

เครือข่ายยุคเริ่มต้นมีสถานีที่ใช้ฮาร์ดแวร์ประเภทเดียวกันและสามารถทำงานร่วมกันได้อย่างกลมกลืน ต่อมาเมื่อเทคโนโลยีเครือข่ายเพิ่มขึ้นเช่น อีเทอร์เน็ตและโทเค็นริง ปัญหาที่เกิดขึ้นคือจะเชื่อมเครือข่ายต่างเทคโนโลยีเข้าด้วยกันได้อย่างไร โดยไม่จำกัดว่าคอมพิวเตอร์จะอยู่ในเครือข่ายเดียวกัน หรือต่างเครือข่ายกัน

2.2 ความหมายของโพรโตคอล

การเชื่อมโยงเครือข่ายต่างฮาร์ดแวร์จำเป็นต้องกำหนดข้อตกลงร่วม หรือ โพรโตคอล (Protocol) เพื่อให้คอมพิวเตอร์สื่อสารกันตามข้อกำหนด ทีซีพี/ไอพีจัดเป็นโพรโตคอลหนึ่งที่ออกแบบมาเพื่อแก้ไขปัญหาที่เกิดขึ้น

2.3 สถาปัตยกรรมของอินเทอร์เน็ตและความหมายของเราเตอร์

ทีซีพี/ไอพีเป็นแกนสำคัญในการถ่ายโอนข้อมูลระหว่างเครื่องคอมพิวเตอร์ที่อาจอยู่ภายในเครือข่ายเดียวกันหรือภายนอกเครือข่าย โครงสร้างของอินเทอร์เน็ตประกอบด้วยเครือข่ายย่อยจำนวนมากต่อเชื่อมกันผ่าน **เราเตอร์** (Router)

เราเตอร์เป็นอุปกรณ์เครือข่ายซึ่งมีหน้าที่เลือกเส้นทางเพื่อนำส่งข้อมูลในรูปแพ็กเก็ต หากเปรียบเทียบกับการส่งจดหมายทางไปรษณีย์แล้ว เราเตอร์ทำหน้าที่เสมือนที่ทำการไปรษณีย์ พนักงานไปรษณีย์จะพิจารณาจุดหมายปลายทางของจดหมายและเลือกเส้นทางส่งจดหมายไปยังที่ทำการไปรษณีย์ถัดไปจนกว่าจะถึงมือผู้รับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครือข่ายที่ต่อเชื่อมกันด้วยเราเตอร์เช่นรูปที่ 2-2 เป็นลักษณะทั่วไปของอินเทอร์เน็ตแต่ละเครือข่ายนิยมเขียนแทนด้วยรูป กลุ่มเมฆเครือข่าย (Network Cloud) เพื่อแสดงเครือข่ายโดยไม่กล่าวถึงโทโปโลยีและการจัดการภายใน

ในยุคต้นของอินเทอร์เน็ตใช้คำว่า เกตเวย์ (Gateway) เพื่อสื่อความหมายถึงอุปกรณ์เชื่อมโยงเครือข่าย แต่ในปัจจุบันนี้เราเตอร์และเกตเวย์เป็นอุปกรณ์ซึ่งมีหน้าที่แตกต่างกัน คำว่าเกตเวย์ที่ใช้เดิมนั้นในปัจจุบันนิยมใช้คำว่าเราเตอร์แทน หน้าที่หลักของเราเตอร์เชื่อมเครือข่ายที่ใช้โพรโตคอลเดียวกันและเลือกเส้นทางส่งข้อมูล ขณะที่เกตเวย์หมายถึงฮาร์ดแวร์หรือซอฟต์แวร์ซึ่งเป็นตัวแปลงระหว่างระบบสองระบบที่มีโพรโตคอล โครงสร้างการจัดข้อมูล หรือสถาปัตยกรรมที่แตกต่างกัน เกตเวย์จะแปลงแพ็กเก็ตจากระบบต้นทางให้อยู่ในรูปแบบของระบบปลายทางเช่นเกตย์เวย์ระหว่างทีซีพี/ไอพีและโพรโตคอล เอสเอ็นเอ (SNA : Systems Network Architecture) หรืออิเล็กทรอนิกส์เมล์เกตเวย์ที่แปลงรูปแบบจดหมายของโพรโตคอลใดๆ ไปสู่ระบบ x.400 ของโอเอสไอ

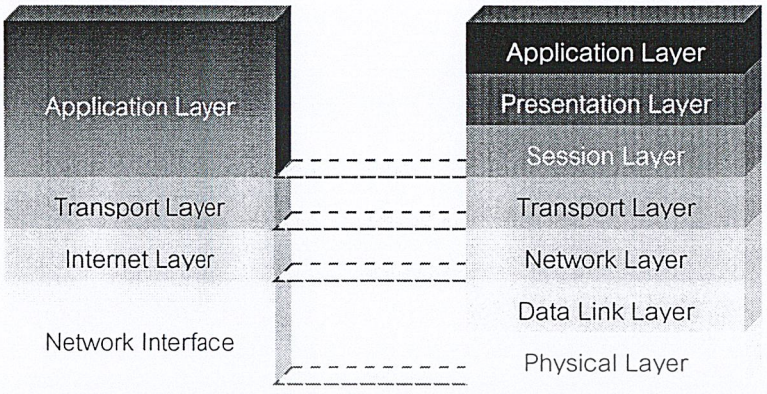
2.4 สถาปัตยกรรมทีซีพี/ไอพี

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

ต่อมาได้พัฒนาเป็นเครือข่ายอินเทอร์เน็ต โพรโตคอลนี้เหมาะสำหรับเชื่อมต่อคอมพิวเตอร์ทั้งใกล้และไกลเข้าด้วยกัน และมีมาตรฐานรองรับทำให้ผู้ผลิตฮาร์ดแวร์และซอฟต์แวร์ สามารถสร้างอุปกรณ์และโปรแกรมที่จะรองรับการทำงานของโพรโตคอลนี้ ทำให้เครื่องคอมพิวเตอร์สามารถรับส่งข้อมูลกันได้ไม่ว่าจะเป็นเครื่องขนาดเล็กหรือขนาดใหญ่ หรือใช้ระบบปฏิบัติการอะไรก็ตาม ทีซีพี/ไอพี (TCP/IP) เป็นชุดโพรโตคอลที่ประกอบด้วยโพรโตคอลต่างๆ หลายโพรโตคอล แต่ละโพรโตคอลมีคุณลักษณะ และมีความสามารถในการทำงานแตกต่างกัน โดยที่ในบทนี้ได้กล่าวถึงรายละเอียดและคุณสมบัติของโพรโตคอลที่สำคัญบางโพรโตคอล

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของโอเอสไอกับเลเยอร์ของทีซีพี/ไอพี

ในแต่ละระดับชั้นของทีซีพี/ไอพีมีการทำงานที่แตกต่างกัน ตั้งแต่การติดต่อกับแอพพลิเคชันจนกระทั่งแปลงเป็นสัญญาณส่งไปตามสายสัญญาณ ซึ่งการทำงานในแต่ละระดับชั้นของทีซีพี/ไอพี มีดังนี้

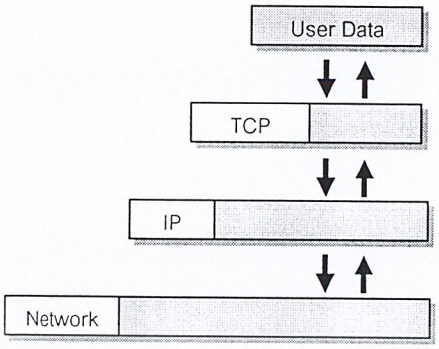
ชั้นแอพพลิเคชัน (Application Layer) ชั้นนี้รองรับการทำงานของแอพพลิเคชันต่างๆ ที่ทำงานเป็นโพรเซสอยู่ในเครื่องต้นทางและปลายทาง โดยจัดการเชื่อมต่อระหว่างโพรเซส หรือแอพพลิเคชันที่อยู่ต่างเครื่องกัน โดยการทำงานของแอพพลิเคชันต่างๆ มีการติดต่อกันตามแต่ละโพรโตคอลเฉพาะแล้วแต่แอพพลิเคชันที่ใช้งาน ซึ่งจะขอบริการจากชั้นทรานสปอร์ตอีกทีหนึ่ง

ชั้นทรานสปอร์ต (Transport Layer) มีการสร้างการเชื่อมต่อกันระหว่างแอพพลิเคชันแบบ 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) ชั้นนี้มีหน้านี้ส่งผ่านข้อมูลระหว่างเครือข่าย โดยมีโพรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใดๆ ในอินเทอร์เน็ต คือ ไอพี (Internet Protocol: IP) ซึ่งกล่าวถึงในหัวข้อถัดไป นอกจากนี้ในชั้นนี้ยังมีโพรโตคอลทำงานอยู่ด้วยอีก 2 ชนิด คือ ไอซีเอ็มพี (Internet Control Message Protocol: ICMP) และเออาร์พี (Address Resolution Protocol: ARP)

ชั้นเน็ตเวิร์กอินเตอร์เฟส (Network Interface Layer) ทำหน้าที่ในการแปลงข้อมูลให้อยู่ในรูปที่เหมาะสมกับเครือข่ายแต่ละแบบซึ่งแตกต่างกันออกไป และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่าย

เอกสารนี้มีลิขสิทธิ์ ห้ามนำไปใช้ในการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2-2 แสดงการข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี

ในชุดโพรโตคอลทีซีพี/ไอพีนี้ มีโพรโตคอลหลักที่ขอกล่าวถึง 3 โพรโตคอล ได้แก่

- โพรโตคอลทีซีพี
- โพรโตคอลยูดีพี ซึ่งทำงานในชั้นทรานสปอร์ต
- โพรโตคอลไอพี ซึ่งทำงานในชั้นอินเทอร์เน็ต
- โพรโตคอลไอซีเอ็มพี
- โพรโตคอลเอสเอ็มทีพี
- โพรโตคอลเอฟทีพี
- โพรโตคอลทีเอฟทีพี
- โพรโตคอลเทลเน็ต
- โพรโตคอลดีเอ็นเอส
- โพรโตคอลเอสเอ็มทีพี
- โพรโตคอลบูตพี
- โพรโตคอลดีเอชซีพี

โดยมีรายละเอียดดังต่อไปนี้

โพรโตคอลทีซีพี (TCP : Transmission Control Protocol)

ทีซีพีทำหน้าที่นำส่งข้อมูลโดยรับประกันความเชื่อถือ ทีซีพีด้านส่งต้องส่งแพ็กเก็ตซ้ำใหม่หากแพ็กเก็ตสูญหาย ทีซีพีด้านรับมีหน้าที่จัดแพ็กเก็ตให้ถูกต้องตามลำดับและกำจัดแพ็กเก็ตซ้ำซ้อน ทีซีพีเป็นโพรโตคอลแบบ “Connection Oriented ” คือต้องสถาปนาการเชื่อมต่อระหว่างสถานีต้นทางและปลายทางก่อนการส่งข้อมูล

ทีซีพีต้นทางจัดแบบข้อมูลเพื่อให้ไอพีดำเนินการ ทีซีพีปลายทางเมื่อรับแพ็กเก็ตจากไอพีก็จะส่งต่อให้โพรโตคอลประยุกต์ โพรโตคอลประยุกต์ที่ใช้บริการผ่านทีซีพีได้แก่ telnet SMTP FTP เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โพรโตคอลยูดีพี (UDP : User Datagram Protocol)

ยูดีพีเป็นโพรโตคอลแบบ "Connectionless" คือไม่ต้องสถาปนาการเชื่อมต่อระหว่างสถานีรับและสถานีส่ง ยูดีพีเป็นโพรโตคอลระดับชั้นเดียวกับทีซีพีแต่ว่าไม่มีกลไกรับประกันความเชื่อถือในการขนถ่ายข้อมูล การข้อมูลสูญหาย ซ้ำซ้อน หรือลำดับไม่ถูกต้อง ยูดีพีจะปล่อยให้โพรโตคอลที่เรียกใช้งานดำเนินการกับปัญหาเหล่านี้เอง

ลักษณะเด่นของยูดีพีคือการใช้การประมวลผลต่ำกว่าทีซีพี เนื่องจากเฮดเดอร์มีขนาดเล็กและไม่ต้องสถาปนาการเชื่อมต่อ แอพพลิเคชันที่ทำงานโดยรับส่งคำถามและคำตอบเป็นรายการ (Transaction) เช่น ดีเอ็นเอส หรือทีเอฟทีพีจะเหมาะกับการใช้บริการผ่านยูดีพี

โพรโตคอลไอพี (IP : Internet Protocol)

ไอพีเป็นโพรโตคอลแกนของทีซีพี/ไอพี ไอพีทำหน้าที่กำหนดรูปแบบของแอดเดรสประจำเครื่องเพื่อใช้ในการลำเลียงข้อมูลจากเครื่องต้นทางไปยังเครื่องปลายทาง นอกจากนี้ยังทำหน้าที่เลือกเส้นทางส่งข้อมูล ตลอดจนแบ่งขนาดข้อมูลให้เหมาะกับฮาร์ดแวร์ระดับล่าง

โพรโตคอลไอซีเอ็มพี (ICMP : Internet Control Message Protocol)

ไอซีเอ็มพีเป็นโพรโตคอลซึ่งใช้รายงานสถานะความผิดพลาดที่เกิดขึ้น ตัวอย่างเช่น ในกรณีที่เราเตอร์ไม่สามารถนำข้อมูลส่งไปถึงปลายทางได้ เราเตอร์จะใช้ไอซีเอ็มพีแจ้งสาเหตุ

โพรโตคอลเอสเอ็มทีพี (SMTP : Simple Mail Transfer Protocol)

บริการพื้นฐานที่มีในทุกเครือข่ายได้แก่บริการไปรษณีย์อิเล็กทรอนิกส์ เอสเอ็มทีพีเป็นโพรโตคอลทำหน้าที่รับส่งจดหมายอิเล็กทรอนิกส์ระหว่างโฮสต์

โพรโตคอลเอฟทีพี (FTP : File Transfer Protocol)

เอฟทีพีให้บริการถ่ายโอนแฟ้มข้อมูลระหว่างเครื่อง เอฟทีพีช่วยให้ผู้ใช้เข้าถึงโฮสต์และจำกัดขอบเขตการทำงานเฉพาะที่เกี่ยวข้องกับแฟ้มข้อมูลเช่น สำเนาข้อมูล ลบแฟ้ม หรือสร้างไดเรกทอรี เป็นต้น

โพรโตคอลทีเอฟทีพี (TFTP : Trivial File Transfer Protocol)

ทีเอฟทีพีทำหน้าที่เป็นโพรโตคอลถ่ายโอนแฟ้มเช่นเดียวกันกับเอฟทีพี แต่ทีเอฟทีพีใช้บริการผ่านยูดีพี ประโยชน์อย่างหนึ่งของการใช้ทีเอฟทีพีได้แก่การใช้ใน สถานีไร้ดิสก์ (Diskless Workstation) ซึ่งไม่มีดิสก์ประจำตัว สถานีไร้ดิสก์จะบูตระบบด้วยโพรโตคอลบูตพีและใช้ทีเอฟทีพีเพื่อถ่ายโอนระบบปฏิบัติการจากทีเอฟทีพีเซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โพรโตคอลเทลเน็ต (TELNET : File Telcommunication Network)

เทลเน็ตเป็นโพรโตคอลสำหรับขอเข้าใช้โฮสต์ระยะไกล หรือเรียกว่า รีโมตล็อกอิน (Remote Login) เทลเน็ตให้บริการเข้าใช้คอมพิวเตอร์ในเครือข่ายโดยเสมือนกับว่ากำลังทำงานอยู่ที่เทอร์มินอลของคอมพิวเตอร์เครื่องนั้น เทลเน็ตเซิร์ฟเวอร์ที่คอมพิวเตอร์ปลายทางจะรอรับการขอบริการจากเทลเน็ตไคลเอ็นต์ ผู้ขอใช้เทลเน็ตจะต้องมีบัญชีประจำเครื่องที่ให้บริการเทลเน็ต

โพรโตคอลดีเอ็นเอส (DNS : Domain Name System)

ดีเอ็นเอสเป็นโพรโตคอลที่ให้บริการสอบถามไอพีแอดเดรสหรือชื่อโดเมน ก่อนการติดต่อไปยังโฮสต์ใดๆ ชื่อโฮสต์จะถูกส่งไปสอบถามผ่านเซิร์ฟเวอร์ที่ให้บริการดีเอ็นเอสเพื่อขอไอพีแอดเดรสกลับมา นอกจากนี้ยังให้บริการเกี่ยวข้อกับฐานข้อมูลประจำเครื่อง

โพรโตคอลเอสเอ็นเอ็มพี (SNMP : Simple Network Management Protocol)

เอสเอ็นเอ็มพีทำหน้าที่เป็นโพรโตคอลช่วยบริหารระบบ เช่น เก็บรวบรวมข้อมูลการทำงานของอุปกรณ์และคอมพิวเตอร์ภายในเครือข่าย ตรวจสอบปริมาณข้อมูลที่ไหลเวียนหรือช่วยวิเคราะห์หาข้อผิดปกติในระบบ เป็นต้น

โพรโตคอลบูตพี(BOOTP : Bootstrap Protocol)

ให้บริการบูตสำหรับสถานีไม่มีดิสก์ สถานีที่เป็นบูตพีไคลเอ็นต์จะติดต่อกับบูตพีเซิร์ฟเวอร์เพื่อขอถ่ายโอนระบบปฏิบัติการ

โพรโตคอลดีเอชซีพี (DHCP : Dynamic Host Configuration Protocol)

ดีเอชซีพีบริการติดตั้งค่าแบบไดนามิกให้โฮสต์ในเครือข่าย การทำงานของดีเอชซีพีเป็บแบบไคลเอ็นต์-เซิร์ฟเวอร์ ดีเอชซีพีเซิร์ฟเวอร์จะให้ค่าแบบไม่ตายตัวกับไคลเอ็นต์ที่ร้องขอบริการ เช่นการให้ไอพีแอดเดรส หรือค่าประจำไคลเอ็นต์

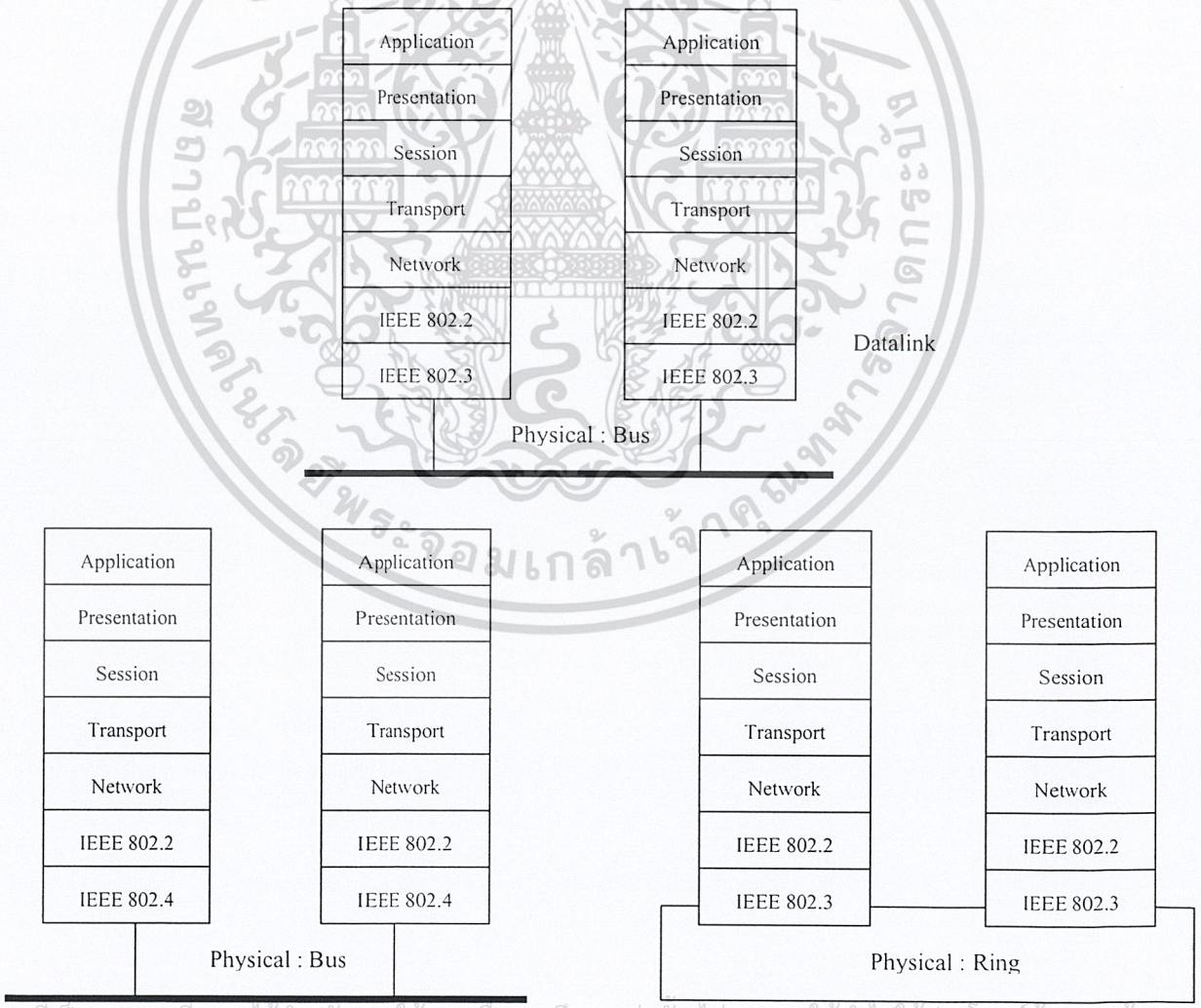
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
เทคโนโลยีเครือข่ายคอมพิวเตอร์

ทีซีพี/ไอพีผ่านการออกแบบมาให้สามารถทำงานกับระบบการสื่อสารระดับล่างโดยไม่จำกัดประเภท ในปัจจุบันจึงมีฮาร์ดแวร์เครือข่ายจำนวนมากทั้งในกลุ่มของแลนและแวนที่รองรับการทำงานร่วมกับทีซีพี/ไอพี

3.1 แลนตามข้อกำหนดของ IEEE

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



รูปที่ 3-1 เป็นแบบจำลองบางส่วนของมาตรฐานซึ่งเป็นที่รู้จักแพร่หลาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเอกสารฉบับนี้ทุกครั้งที่มีการนำไปใช้

IEEE กำหนดเครือข่ายเฉพาะที่โดยใช้ตัวเลข 802 ตามด้วยตัวเลขย่อยเป็นรหัสประจำแต่ละมาตรฐาน

- IEEE 802.3 หรืออีเทอร์เน็ต ใช้โพรโตคอลซีเอสเอ็ม/เอซีดีในโทโพโลยีแบบบัส
- IEEE 802.4 หรือโทเค็นบัส ใช้โพรโตคอลส่งผ่านโทเค็นในโทโพโลยีแบบบัส
- IEEE 802.5 หรือโทเค็นริง ใช้โพรโตคอลส่งผ่านโทเค็นในโทโพโลยีแบบวงแหวน

ข้อกำหนดเพิ่มเติมที่ IEEE สร้างขึ้นในทุกมาตรฐาน 802 คือการแยกระดับชั้นดาต้าลิงค์ออกเป็น 2 ส่วนย่อย โดยให้มาตรฐาน 802.2 ซึ่งเรียกว่า แอลแอลซี (LLC : Logical Lonk Control) เป็นส่วนเชื่อมต่อกับชั้นเน็ตเวิร์ก อินเทอร์เฟสของแอลแอลซีในเครือข่ายแต่ละชนิด (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) ความยาวของสายในเซ็กเมนต์หนึ่งๆ ไม่เกิน 500 เมตร
- 10Base2 อีเทอร์เน็ตโทโพโลยีแบบบัสซึ่งใช้สายโคแอเชียลแบบบาง (Thin Ethernet) ความยาวของสายในเซ็กเมนต์หนึ่งๆไม่เกิน 185 เมตร
- 10BaseT อีเทอร์เน็ตโทโพโลยีแบบดาวซึ่งใช้ฮับเป็นศูนย์กลาง สถานีและฮับเชื่อมด้วยสายยูทีพี (Unshield Twisted Pair) ด้วยความยาวไม่เกิน 100 เมตร

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

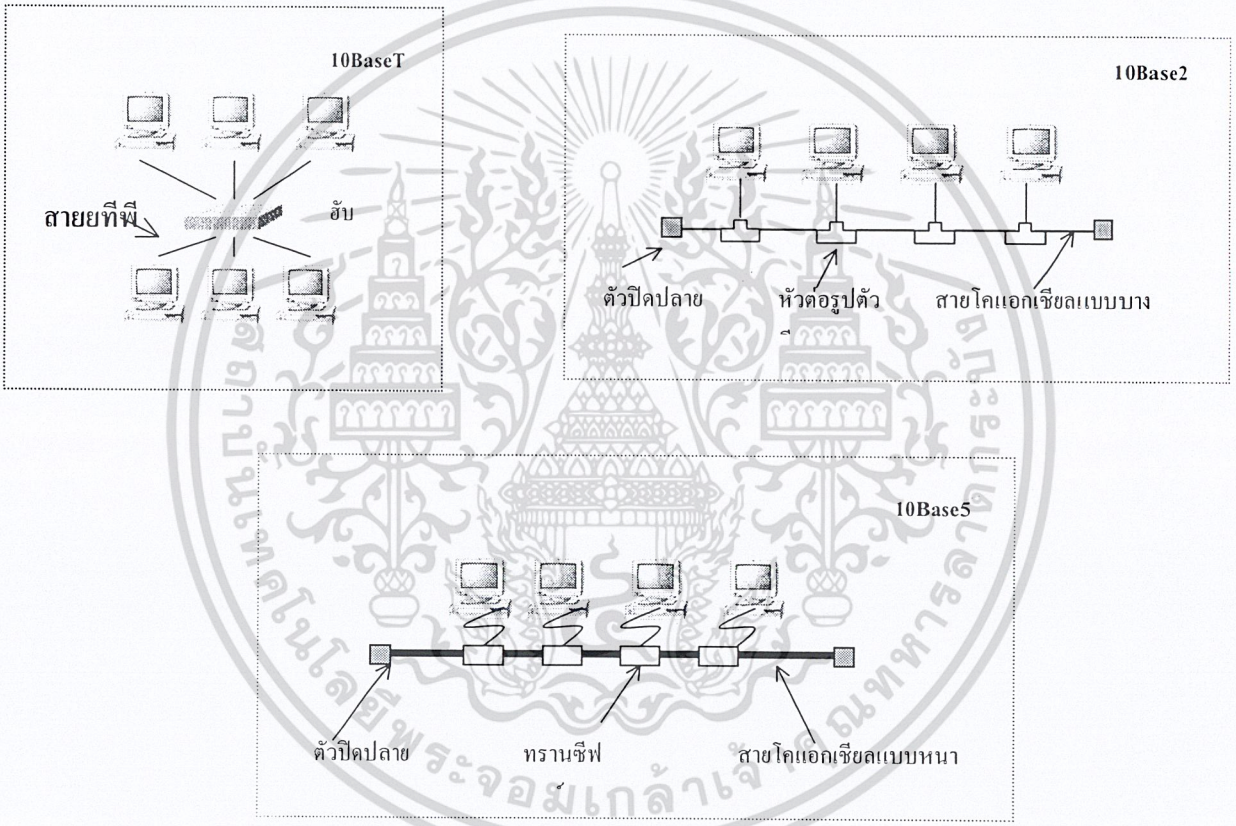
ส่วนมาตรฐานอีเทอร์เน็ตความเร็ว 100 เมกะบิตต่อวินาทีที่นิยมใช้ในปัจจุบันได้แก่ 100BaseTX และ 100BaseFX สำหรับอีเทอร์เน็ตความเร็วสูงแบบกิกะบิตอีเทอร์เน็ตเริ่มแพร่หลายมากขึ้น ตัวอย่างของมาตรฐานกิกะบิตอีเทอร์เน็ตในปัจจุบันได้แก่ 1000BaseT,1000BaseLX และ 1000BaseSX เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.2 IEEE 802.4 (Token Bus)

IEEE 802.4 หรือ โทเค็นบัส (Token Bus) มีโทโพโลยีแบบบัสเช่นเดียวกับ IEEE 802.3 แต่มีข้อกำหนดว่าการเข้าสายใช้สายสื่อสารโดยใช้โทเค็นพิเศษซึ่งทำหน้าที่เป็นเฟรมสัญญาณกำหนดจังหวะให้สถานีเข้าใช้สายสื่อสาร โทเค็นจะถูกนำส่งจากสถานีหนึ่งไปยังอีกสถานีหนึ่งและวนกลับที่เดิมเป็นวงรอบ สถานีที่ได้รับโทเค็นจะมีสิทธิ์ใช้สายสื่อสารเพื่อส่งข้อมูลได้

สายสื่อสารในโทเค็นบัสที่มักใช้สายโคแอกเชียล และมีอัตราเร็วหลายระดับคือ 1,5 หรือ 10 เมกะบิต การใช้โทเค็นช่วยให้สถานีไม่ต้องแย่งยึดช่องสายสัญญาณเหมือนใน IEEE 802.3 หากแต่ความซับซ้อนของโพรโตคอลทำให้ IEEE 802.4 ไม่เป็นที่นิยมใช้



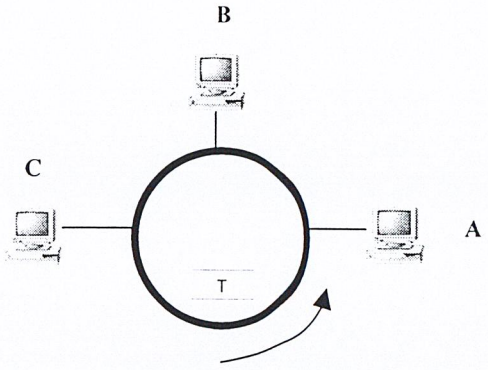
รูปที่ 3-2 อีเทอร์เน็ตประเภทต่างๆ

3.1.3 IEEE 802.5 (Token Ring)

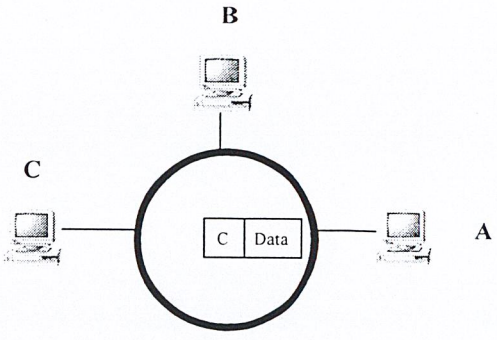
IEEE 802.5 หรือ โทเค็นริง (Token Ring) หรือมักเรียกว่าไอบีเอ็มโทเค็นริงจัดเป็นเครือข่ายที่ใช้โทโพโลยีแบบวงแหวนด้วยสายคู่ตีเกลียวหรือเส้นใยนำแสง อัตราการส่งข้อมูลของโทเค็นริงที่ใช้โดยทั่วไปคือ 4 และ 16 เมกะบิตต่อวินาที

รูปที่ 3-3 แสดงการทำงานของโทเค็นริง โดยมีเฟรมพิเศษเรียกว่า โทเค็นว่าง (free token) วิ่งวนอยู่ สถานีที่ต้องการส่งข้อมูลจะรอให้โทเค็นว่างเดินทางมาถึงแล้วรับโทเค็นว่างมาเปลี่ยนเป็น **เฟรมข้อมูล** data frame) โดยใส่แฟล็กแสดงเฟรมข้อมูลและบรรจุแอดเดรสของสถานีต้นทางและปลายทางตลอดจนข้อมูลอื่นๆ จากนั้นจึงปล่อยเฟรมนี้ออกไป

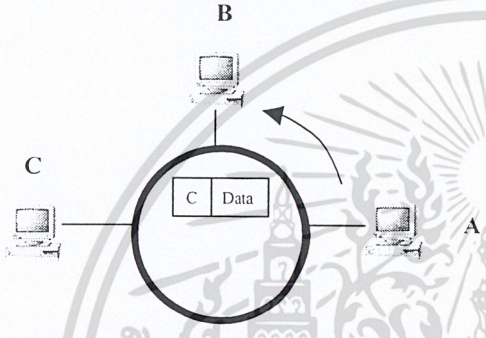
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



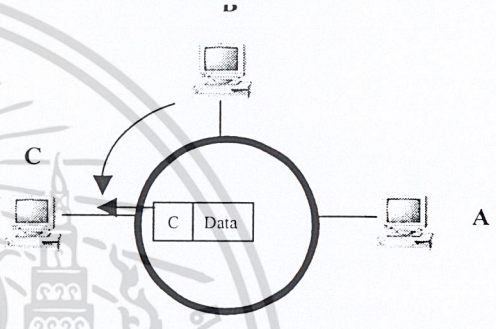
(1)โทเค็นว่าง



(2) A เปลี่ยนโทเค็นว่างเป็นเฟรม
เพื่อส่งข้อมูลให้ C



(3) ได้รับเฟรมแต่ปล่อยออกไป



(4) รับเฟรมไปใช้งาน

รูปที่ 3-3 การทำงานโทเค็นริง

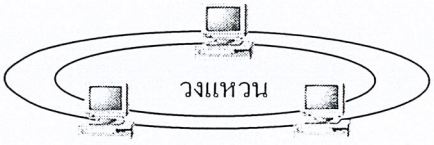
เมื่อสถานีปลายทางได้รับเฟรมจะสำเนาข้อมูลไว้และปล่อยเฟรมให้วนกลับมายังสถานีส่ง สถานีส่งจะตรวจสอบเฟรมและปล่อยโทเค็นว่างคืนสู่เครือข่ายให้สถานีอื่นมีโอกาสส่งข้อมูลต่อไป กลไกแบบส่งผ่านโทเค็นจัดอยู่ในประเภทประเมินเวลาได้ กล่าวคือสามารถคำนวณเวลาสูงสุดที่สถานีมีสิทธ์จับโทเค็นเพื่อส่งข้อมูลได้ โทเค็นริงจึงเหมาะกับระบบที่ต้องการความแน่นอนทางเวลาหรืองานแบบเวลาจริง

3.1.4 เอฟดีดีไอ

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

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-4 โทโปโลยีเอฟดีดีไอ

สายสัญญาณที่ใช้กับเอฟดีดีไอมี 2 ประเภทคือเส้นใยนำแสงและสายคู่ตีเกลียว เส้นใยนำแสงที่ใช้งานมีทั้งแบบ หลายภาวะ (Multi mode) ภาวะเดี่ยว (Single mode) เส้นใยนำแสงแบบหลายภาวะสามารถเชื่อมสถานีที่อยู่ห่างกันได้ 2 กิโลเมตร ขณะที่แบบภาวะเดี่ยวสามารถเชื่อมสถานีที่อยู่ห่างกันได้ระยะทางราว 40 กิโลเมตร ส่วนสายคู่ตีเกลียวจะใช้สายประเภท 5 (Category 5) เอฟดีดีไอที่ใช้สายคู่ตีเกลียวมีชื่อเรียกเฉพาะว่า ซีดีดีไอ (CDDI : Copper Distributed Data Interface)

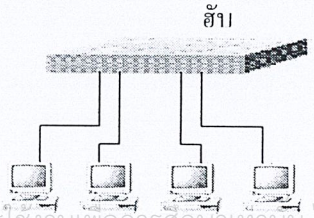
3.2 อุปกรณ์เครือข่าย

เครือข่ายขนาดเล็กมักประกอบด้วยสถานีงานและสถานีเซิร์ฟเวอร์ต่อเชื่อมกัน การเชื่อมเครือข่ายขนาดเล็กจำนวนหลายเครื่องเข้าด้วยกันเพื่อแลกเปลี่ยนข้อมูลระหว่างกันต้องอาศัยอุปกรณ์เครือข่ายที่มีลักษณะสมบัติแตกต่างกันไป อุปกรณ์เครือข่ายพื้นฐานที่พบโดยทั่วไปและจะอธิบายในหัวข้อนี้ได้แก่ ฮับ บริดจ์ และเราเตอร์

3.2.1 ฮับ

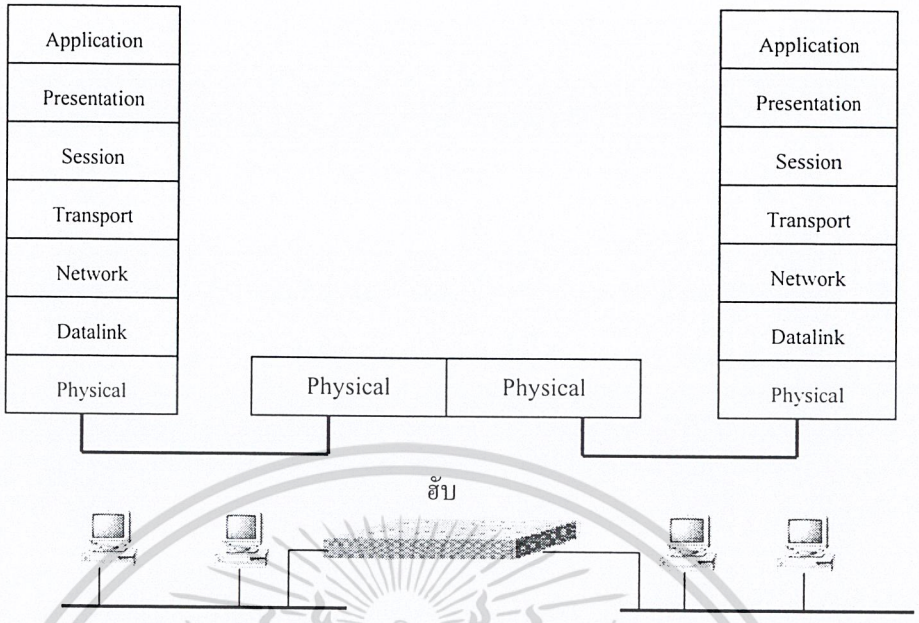
ฮับ (Hub) เป็นอุปกรณ์เชื่อมสถานีเครือข่ายที่ใช้โทโพโลยีแบบดาว เช่น อีเทอร์เน็ต 10BaseT ดังรูปที่ 3-5 ฮับมีพอร์ตได้หลายแบบเช่น พอร์ต RJ-45 ใช้กับสายคู่ตีเกลียวในเครือข่าย 10BaseT หรือพอร์ตไฟเบอร์ใช้กับเส้นใยนำแสงในเครือข่าย 10BaseF

ฮับเป็นอุปกรณ์ในระดับฟิสิคัลเช่นเดียวกับรีพีตเตอร์ดังรูปที่ 3-6 ด้วยเหตุนี้จึงมักเรียกฮับว่าเป็นมัลติพอร์ตรีพีตเตอร์ หน้าที่ของฮับคือขยายสัญญาณและกระจายแพ็กเก็ตไปทุกพอร์ต ฮับใช้เชื่อมต่อเครือข่ายประเภทเดียวกันเท่านั้น เครือข่ายที่เชื่อมด้วยฮับจะรวมเป็นเครือข่ายเดียวกันดังนั้นแพ็กเก็ตที่สร้างจากเครือข่ายหนึ่งผ่านฮับไปอีกเครือข่ายหนึ่ง



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่3-5 การใช้ฮับในโทโปโลยีแบบดาว



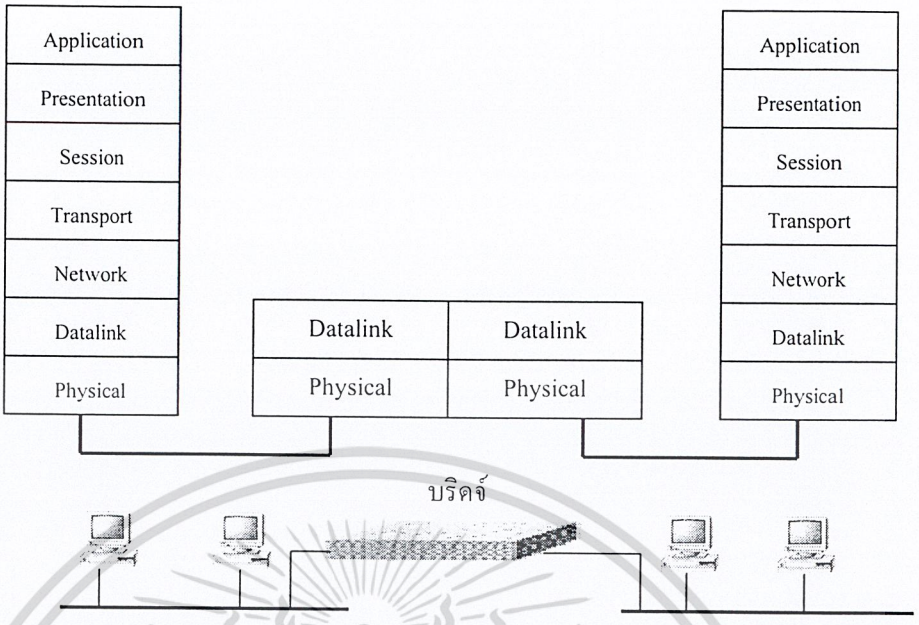
รูปที่3-6 แบบจำลองการทำงานของฮับ

3.2.2 บริดจ์

บริดจ์ (Bridge) ทำหน้าที่เชื่อมเครือข่ายย่อยสองเครือข่ายเข้าด้วยกัน บริดจ์ทำงานในระดับชั้นดาต้าลิงท์จึงสามารถใช้เชื่อมเครือข่ายประเภทเดียวกันหรือต่างกันได้ เช่น เชื่อมอีเทอร์เน็ตเข้ากับโทเค็นริง แบบจำลองการทำงานของบริดจ์แสดงได้ดังรูปที่ 3-7

เมื่อใช้บริดจ์เชื่อมเครือข่ายประเภทเดียวกัน บริดจ์จะตรวจสอบว่าจะนำส่งแพ็กเก็ตโดยตรวจสอบฮาร์ดแวร์แอดเดรสปลายทางกับตารางเลือกเส้นทางที่บริดจ์สร้างขึ้น หากมีแพ็กเก็ตที่ต้องข้ามจากบริดจ์จากเครือข่ายหนึ่งไปยังอีกเครือข่ายหนึ่ง บริดจ์จะทำสำเนาแพ็กเก็ตบิตต่อบิต ข้ามไปยังอีกเครือข่ายหนึ่งโดยไม่เปลี่ยนแปลงเฮดเดอร์หรือข้อมูล หากสถานีต้นทางและสถานีปลายทางอยู่ในเครือข่ายเดียวกัน บริดจ์จะได้รับแพ็กเก็ตแต่ไม่ส่งแพ็กเก็ตไปยังเครือข่ายอีกด้านหนึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่3-7 แบบจำลองการทำงานของบริดจ์

บริดจ์นำส่งบรอดคาสต์แพ็กเก็ตข้ามเครือข่ายโดยไม่ป้องกัน สถานีอีเทอร์เน็ตหรือโทเคนริงต้องดำเนินการกับแพ็กเก็ตที่สถานีอื่นบรอดคาสต์มาให้ บรอดคาสต์แพ็กเก็ตมักเกิดจากการทำงานของโพรโตคอลเออาร์พี บูตพี หรืออาร์ไอพี หากจำนวนบรอดคาสต์แพ็กเก็ตสูงจะส่งผลให้สมรรถนะการทำงานของสถานีในเครือข่ายลดลง

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

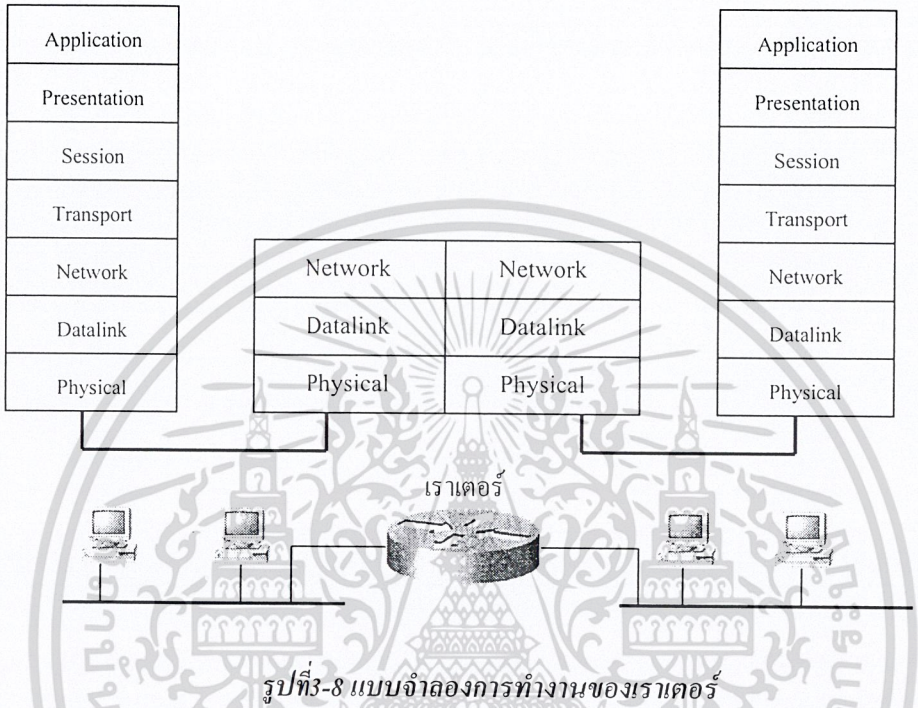
สวิตช์ทำหน้าที่เช่นเดียวกับบริดจ์ในการแบ่งเครือข่ายใหญ่ออกเป็นเครือข่ายย่อย และป้องกันไม่ให้มีการส่งเฟรมที่ไม่จำเป็นจากเครือข่ายหนึ่งข้ามไปยังอีกเครือข่ายหนึ่ง สวิตช์มักประกอบด้วยพอร์ตจำนวนมาก ต่างจากบริดจ์ที่มีเพียงสองพอร์ต แต่ละพอร์ตของสวิตช์สามารถเชื่อมโยงระหว่างสวิตช์ด้วยกันหรือเชื่อมสถานีเข้าสู่สวิตช์โดยตรง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

คุณลักษณะสำคัญของสวิตช์คือมีอิเล็กทรอนิกส์สวิตช์ความเร็วสูงทำหน้าที่ส่งเฟรมจากพอร์ตต้นทางสู่ปลายทางโดยไม่รบกวนพอร์ตอื่น ลักษณะนี้ต่างจากบริดจ์หรือเราเตอร์ซึ่งเฟรมถูกส่งจากพอร์ตหนึ่งไปยังอีกพอร์ตหนึ่งโดยอาศัยการประมวลผลของไมโครโปรเซสเซอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.3 เราเตอร์

เราเตอร์เป็นอุปกรณ์ที่ทำงานวนระดับชั้นเน็ตเวิร์กตามรูปที่ 3-8 เราเตอร์ทำงานร่วมกับฮาร์ดแวร์ในระดับดาต้าลิงค์ได้หลายรูปแบบ หน้าที่ของเราเตอร์คือจัดแบ่งเครือข่ายและเลือกเส้นทางที่เหมาะสมเพื่อนำส่งแพ็กเก็ต เราเตอร์จะห้องกันการบรอดคาสต์แพ็กเก็ตจากเครือข่ายหนึ่งไม่ให้ข้ามมาอีกเครือข่ายหนึ่ง



รูปที่3-8 แบบจำลองการทำงานของเราเตอร์

ในอินเทอร์เน็ตมักเรียกเราเตอร์ว่า ไอพีเราเตอร์ (IP Router) เนื่องจากเราเตอร์ทำงานตามข้อกำหนดของโพรโตคอลไอพี เราเตอร์ทำหน้าที่เลือกเส้นทางโดยสร้างแผนที่เครือข่ายและเก็บอยู่ในรูปตารางเส้นทาง เมื่อเราเตอร์ได้รับแพ็กเก็ตก็จะตรวจสอบแอดเดรสปลายทางและส่งแพ็กเก็ตไปยังอีกอินเทอร์เฟสที่เป็นช่องทางไปสู่เครือข่ายปลายทาง เราเตอร์ประกอบด้วยหลายอินเทอร์เฟสเพื่อเชื่อมต่อกับเครือข่ายต่างชนิดเข้าด้วยกันได้ ตัวอย่างเช่น อินเทอร์เฟสอีเทอร์เน็ต โทเค็นริง เอฟดีดีไอ เอทีเอ็ม หรืออินเทอร์เฟสแบบอนุกรม เป็นต้น

ซอฟต์แวร์ในเราเตอร์จะรับแพ็กเก็ตและเก็บเข้าบัฟเฟอร์ก่อนนำไปประมวลผล วิธีนี้ต่างจากสวิตช์ซึ่งอาศัยฮาร์ดแวร์ตรวจสอบแอดเดรสปลายทางของแพ็กเก็ตและ “สวิตช์” แพ็กเก็ตนั้นจากพอร์ตหนึ่งไปสู่อีกพอร์ตหนึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

บทที่ 4
ไอพีแอดเดรส

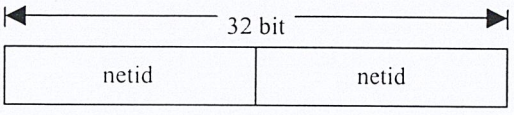
ไอพีแอดเดรสเป็นส่วนประกอบสำคัญอย่างหนึ่งในเครือข่ายทีซีพี/ไอพี ไอพีแอดเดรสเป็นแอดเดรสทางซอฟต์แวร์ประจำสถานีเครือข่ายผู้ออกแบบเครือข่ายจำเป็นต้องศึกษาและเข้าใจรูปแบบของไอพีแอดเดรสโดยละเอียด การจัดแบ่งแอดเดรสออกเป็นคลาส และการจัดแบ่งเครือข่ายย่อยหรือซับเน็ต รวมทั้งสามารถออกแบบซับเน็ตโดยเลือกใช้ไอพีแอดเดรสได้อย่างถูกต้อง เนื้อหาซึ่งจะกล่าวถึงในบทนี้มีดังนี้

4.1 ไอพีแอดเดรส

อุปกรณ์ที่เชื่อมเข้าเครือข่ายและสามารถทำงานตามข้อกำหนดของทีซีพี/ไอพีจะต้องมีแอดเดรสประจำอุปกรณ์นั้น อุปกรณ์ดังกล่าวอาจเป็นโฮสต์ เราเตอร์ เครื่องพิมพ์ หรือแม้กระทั่งอุปกรณ์สำนักงานเช่นโทรศัพท์หรือเครื่องถ่ายเอกสาร ไอพีรุ่นสี่กำหนดให้ใช้ไอพีแอดเดรสขนาด 32 บิต อุปกรณ์ที่เชื่อมกับอินเทอร์เน็ตจะมีไอพีแอดเดรส 32 บิตประจำอินเทอร์เฟสที่ไม่ซ้ำกัน อุปกรณ์อย่างเราเตอร์จะมีหลายอินเทอร์เฟสซึ่งแต่ละอินเทอร์เฟสจะมีไอพีแอดเดรสหลายค่าตามจำนวนอินเทอร์เฟสโดยไม่ซ้ำค่ากัน แต่ถ้าเป็นเครื่องคอมพิวเตอร์หรือโฮสต์ปกติจะมีแค่อินเทอร์เฟสเดียว 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 (หรืออีกนัยหนึ่งคืออินเทอร์เฟสของโฮสต์) ในเครือข่าย ไอพีแอดเดรสจึงแบ่งได้เป็นสองส่วนตามรูปที่ 4-1 จำนวนบิตที่ใช้สำหรับเลขเครือข่ายและเลขโฮสต์ขึ้นอยู่กับคลาสที่สังกัด



รูปที่ 4-1 รูปแบบของไอพีแอดเดรส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

49970

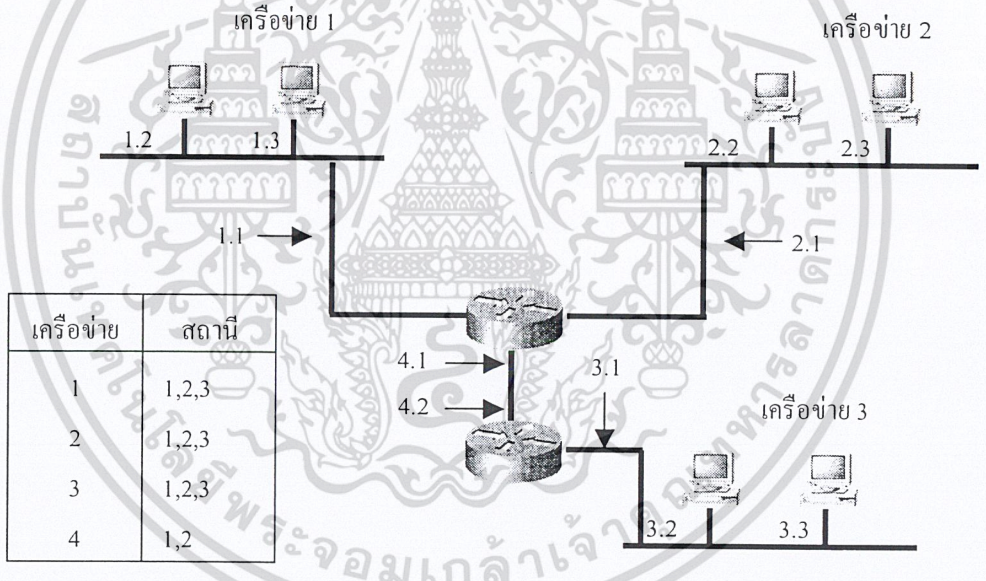
ในปัจจุบันฟิลด์กำหนดเลขเครือข่ายนิยมเรียกว่า พรีฟิกซ์เครือข่าย (Network-Prefix) เพราะทุกโฮสต์ในเครือข่ายจะต้องมีพรีฟิกซ์หรือบิตนำหน้าเหมือนกัน ตัวอย่างเช่นหากมีเลขเครือข่ายจำนวน 16 บิต ก็จะเรียกว่า พรีฟิกซ์ 16 เป็นต้น

4.1.1 ความสำคัญของเลขเครือข่ายและเลขโฮสต์

การจัดแบ่งไอพีแอดเดรสออกเป็นสองส่วนที่ประกอบด้วยเลขเครือข่ายและเลขโฮสต์ก็เพื่อประโยชน์ในการดูแลระบบ เราเตอร์จะอาศัยเลขเครือข่ายเพื่อเลือกเส้นทางส่งแพ็กเก็ตด้วยหลักการต่อไปนี้

โฮสต์ที่มีเลขเครือข่ายชุดเดียวกันย่อมอยู่ภายในเครือข่ายเดียวกัน และสามารถสื่อสารถึงกันด้วยเฟรมดาต้าลิงค์โดยไม่ต้องพึ่งเราเตอร์

โฮสต์ที่มีเลขเครือข่ายต่างกันจะอยู่ต่างเครือข่ายกัน การสื่อสารระหว่างโฮสต์จะอาศัยเราเตอร์ที่เชื่อมต่อเครือข่ายเป็นผู้นำส่งแพ็กเก็ต เราเตอร์อาจเชื่อมเครือข่ายที่อยู่ติดกันหรือส่งแพ็กเก็ตผ่านเราเตอร์อื่นไปยังปลายทางดังรูป



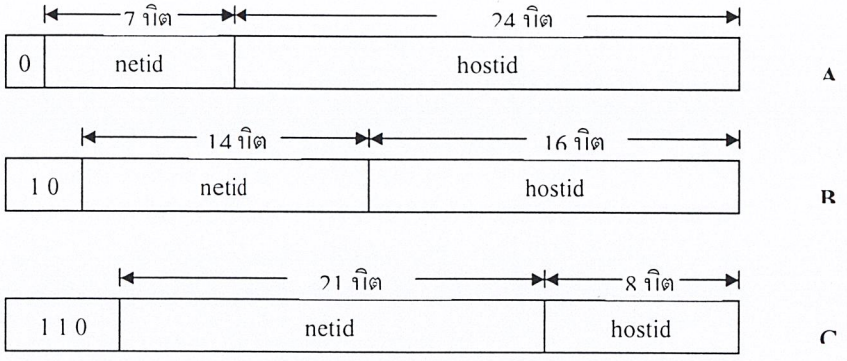
เครือข่าย	สถานี
1	1,2,3
2	1,2,3
3	1,2,3
4	1,2

รูปที่ 4-2 เราเตอร์เชื่อมโยงเครือข่ายที่มีเลขเครือข่ายต่างกัน

4.1.2 การจัดคลาสเครือข่าย

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



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 ตัวอย่างเช่นในคลาส 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

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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 เป็นแอดเดรสลูปแบ็คคือเป็นแอดเดรสที่ใช้เพื่อเชื่อมเข้าสู่อินเทอร์ลูปแบ็ค 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 และมักไม่จัดสรรเครือข่ายในคลาสนี้ให้กับผู้ขอจดทะเบียนรายใหม่หากไม่มีความจำเป็นอย่างแท้จริง

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 แอดเดรสทำให้เครือข่ายเหมาะสำหรับหน่วยงานขนาดเล็ก 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

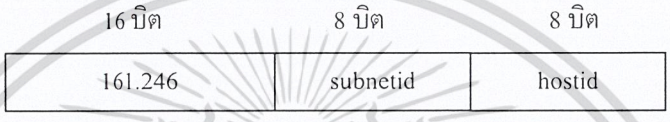
สำหรับคลาส E มีแอดเดรสจาก 240.0.0.0 ถึง 254.255.255.255 ซึ่งสำรองไว้เพื่อความจำเป็นเฉพาะงานในอนาคต

เอกสารนี้จัดทำขึ้นสำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 การแบ่งเครือข่ายย่อย

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

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



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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.1 ซับเน็ตมาสก์

เมื่อผู้วางระบบเลือกขนาดซับเน็ตแล้วจะกำหนดพารามิเตอร์เพื่อใช้บอกให้โฮสต์และเราเตอร์ทราบว่าซับเน็ตที่ใช้งานมีขนาดกี่บิต ค่านี้เรียกว่า ซับเน็ตมาสก์ (Subnet Mask)

ซับเน็ตมาสก์เป็นตัวเลข 32 บิต ซึ่งเขียนอยู่ในรูป Dotted-Decimal เช่นเดียวกับการเขียนไอพีแอดเดรส ซับเน็ตมาสก์จะมีบิตที่ตรงกับเลขเครือข่ายและเลขซับเน็ตเท่ากับ "1" ส่วนบิตที่ตรงกับเลยโฮสต์มีค่าเท่ากับ "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

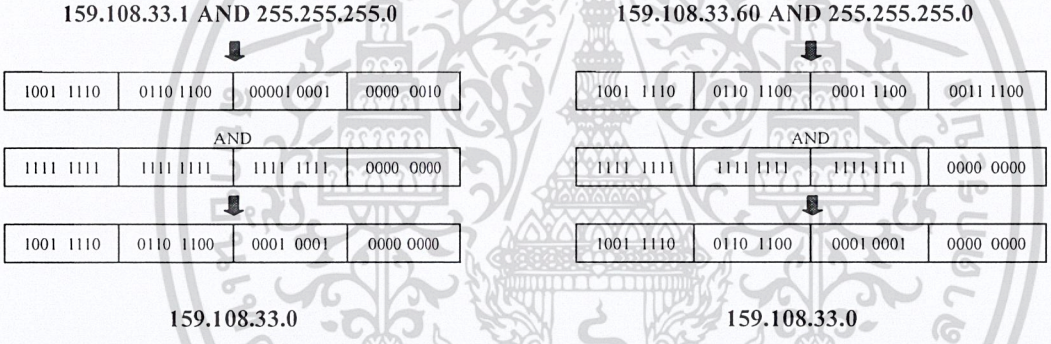
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

ซับเน็ตมาสก์นอกจากจะช่วยจัดแบ่งเครือข่ายย่อยแล้วยังใช้ประโยชน์ในการเลือกเส้นทางส่งไอพีดาต้าแกรมระหว่างเครือข่ายย่อยด้วย เช่นโฮสต์ 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” บิตต่อบิตกับค่าซับเน็ตมาสก์ดังรูป



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 ครอบเอาเลขเครือข่ายออกมา

ในกรณีที่โฮสต์ปลายทางอยู่ต่างเครือข่ายกับโฮสต์ต้นทาง เช่น แอดเดรสของโฮสต์ต้นทางคือ 161.246.33.2 และโฮสต์ปลางทางคือ 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 และจะส่งแพ็กเก็ตไปยังเราเตอร์เพื่อให้เราเตอร์เพื่อให้เราเตอร์นำส่งแพ็กเก็ตต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
โพรโตคอลเลือกเส้นทาง (Routing Protocol)

การเลือกเส้นทางเป็นหน้าที่สำคัญอย่างหนึ่งของไอพี เราเตอร์อาศัยตารางเส้นทาง (Routing Table) เพื่อนำส่งดาต้าแกรมไปยังเครือข่ายปลายทาง ตารางเส้นทางอาจปรับปลี่ยนได้ตามสภาพเครือข่ายอัติโนมัติโดยใช้โพรโตคอลเลือกเส้นทาง หรือผู้ดูแลระบบเป็นคนปรับเปลี่ยนเส้นทางเอง

5.1 การเลือกเส้นทาง (Routing)

การเลือกเส้นทางเป็นกระบวนการที่เกิดขึ้นในระดับชั้นที่ 3 หรือระดับเน็ตเวิร์ก ของแบบจำลองทีซีพี/ไอพี ซอฟต์แวร์ไอพีที่อยู่ในโฮสต์หรือเราเตอร์จะนำส่งดาต้าแกรมไปตามเส้นทาง โดยอาศัยเลขเครือข่ายของไอพีแอดเดรสตามแต่ละคลาส เลขเครือข่ายเป็นเสมือนค่ากำหนดตำแหน่งปลายทางของเครือข่ายซึ่งคล้ายกับรหัส 3 ตัวแรกที่กำหนดชุมสายโทรศัพท์ ระบบรหัสชุมสายจะกำหนดที่ตั้งทางภูมิศาสตร์ด้วยว่าอยู่ทิศทางใด แต่เลขเครือข่ายในไอพีแอดเดรสนี้ไม่ได้มีส่วนสัมพันธ์กับที่ตั้งของเครือข่าย

เครือข่ายโดยทั่วไปประกอบด้วยสถานีส่วนปลาย (End Node) และเราเตอร์หรืออุปกรณ์อื่นทำงานร่วมกัน โดยกระบวนการเลือกหาเส้นทางจะเกิดขึ้นทั้งที่สถานีส่วนปลายและที่เราเตอร์ คือ

1. สถานีส่วนปลายที่เป็นสถานีส่งต้องทราบว่าจะนำส่งดาต้าแกรมให้เราเตอร์ได้อย่างไรและเมื่อใด
2. เราเตอร์ต้องทราบเส้นทางเชื่อมโยงไปยังเราเตอร์ตัวอื่น เพื่อส่งดาต้าแกรมไปตามเส้นทางที่เหมาะสมที่สุด
3. เราเตอร์ที่เชื่อมกับเครือข่ายของสถานีปลายทางต้องทราบถึงวิธีส่งดาต้าแกรมไปยังสถานีปลายทางนั้น

ในตอนแรกสถานีต้นทางจะเป็นผู้ตัดสินใจขั้นแรกว่าต้องส่งดาต้าแกรมไปยังสถานีในเครือข่ายด้วยกันเองหรือต้องส่งผ่านเราเตอร์ โดยสถานีต้นทางจะเปรียบเทียบเลขเครือข่ายของแอดเดรสต้นทางและปลายทางกับค่าซับเน็ตมาสก์ หากได้เลขครือข่ายเหมือนกันแสดงว่าสถานีปลายทางอยู่ในเครือข่ายเดียวกัน สถานีต้นทางจะใช้เออาร์พีสอบถามฮาร์ดแวร์แอดเดรสหรืออ่านจากแคช และบรรจุฮาร์ดแวร์แอดเดรสเข้าสู่เฟรมดาต้าลิงค์เพื่อส่งตรงไปยังสถานีปลายทาง แต่ถ้าเลขเครือข่ายมีค่าต่างกันแสดงว่าสถานีปลายทางอยู่ต่างเครือข่ายกัน สถานีส่งก็จะส่งเฟรมไปให้เราเตอร์เพื่อให้เราเตอร์นำส่งต่อไป เมื่อเราเตอร์ได้รับเฟรมนี้ก็จะส่งต่อไป ไอพีแอดเดรสต้นทางและปลายทางในไอพีดาต้าแกรมจะไม่เปลี่ยนแปลงค่าระหว่างการลำเลียง แต่ฮาร์ดแวร์แอดเดรสจะเปลี่ยนแปลงไปตามเครือข่าย

5.2 ตารางเส้นทาง(Routing Table)

โฮสต์และเราเตอร์จะเก็บแอดเดรสปลายทางสำหรับใช้เป็นเส้นทางส่งดาต้าแกรมไว้ในรูปตารางที่เรียกว่า ตารางเส้นทาง (Routing Table) ค่าในตารางเส้นทางมักประกอบด้วยไอพีแอดเดรสของเครือข่ายปลายทาง และเกตเวย์ซึ่งเป็นทางออกของดาต้าแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางเส้นทางจะบรรจุแอดเดรสหนึ่งซึ่งทำหน้าที่เป็นช่องทางออกไปสู่เครือข่ายใดๆ ที่ไม่ได้ระบุอยู่ในตารางเส้นทาง แอดเดรสนี้เรียกว่า ดีฟอลต์เกตเวย์ (Default Gateway) หรือ ดีฟอลต์เราเตอร์ (Default Router) ซึ่งแทนด้วยแอดเดรส 0.0.0.0 ไอพีจะเลือกเส้นทางโดยตรวจสอบว่าแอดเดรสปลายทางตรงกับรายการใดในตาราง แล้วส่งดาต้าแกรมไปยังเกตเวย์ของรายการนั้น หากไม่พบรายการใดในตาราง ก็จะส่งไปยังดีฟอลต์เกตเวย์

5.3 ประเภทของการเลือกเส้นทาง

เมื่อดาต้าแกรมเดินทางออกนอกเครือข่ายที่อาจต้องผ่านเราเตอร์จำนวนมากโดยมีเส้นทางลำเลียงได้หลายเส้นทาง ปัญหาที่สำคัญคือเราเตอร์จะทราบได้อย่างไรว่ามีเส้นทางใดบ้าง และเส้นทางใดที่เป็นเส้นทางที่ดีที่สุด

เราเตอร์จะส่งดาต้าแกรมได้จำเป็นต้องมีแผนที่เครือข่าย ซึ่งก็คือตารางเส้นทาง สำหรับวิธีที่ใช้ในการสร้างตารางเส้นทางนี้มี 2 แบบคือ

1. **การเลือกเส้นทางแบบสแตติก(Static Routing)** ตารางเลือกเส้นทางสร้างขึ้นและแก้ไขโดยผู้ดูแลระบบเครือข่าย
2. **การเลือกเส้นทางแบบไดนามิก(Dynamic Routing)** ใช้ซอฟต์แวร์คำนวณหาค่าตารางเลือกเส้นทาง ตารางสามารถปรับเปลี่ยนได้หากสภาพเครือข่ายเปลี่ยนไป

การเลือกเส้นทางแบบสแตติก

การเลือกเส้นทางแบบนี้ผู้ดูแลระบบเครือข่ายเป็นผู้พิจารณาและคำนวณหาเส้นทางทั้งหมด โดยใส่ค่าตารางเส้นทางให้กับเราเตอร์ทุกตัว ตารางเลือกเส้นทางนี้จะมีค่าตายตัวตลอดถึงแม้ว่าสภาพของเครือข่ายจะเปลี่ยนไปดังนั้นผู้ดูแลระบบจะต้องคอยตรวจสอบเครือข่ายและปรับเปลี่ยนตารางเส้นทางให้ถูก

รูปที่ 5-1 แสดงเครือข่ายที่ประกอบด้วยเราเตอร์ 2 ตัว เราเตอร์แต่ละตัวจะมีตารางเส้นทางที่ผู้ดูแลระบบป้อนเพื่อใช้กำหนดทิศทางการส่งดาต้าแกรม ทุกเครือข่ายที่เชื่อมโดยตรงกับเราเตอร์อื่นจะมีค่าเกตเวย์เท่ากับไอพีแอดเดรสประจำอินเทอร์เฟสนั้น ส่วนเครือข่ายที่ต้องผ่านเราเตอร์อื่นจะมีค่าเกตเวย์เท่ากับไอพีแอดเดรสของเราเตอร์ขั้นถัดไป (Next Hop Router) เช่นตารางเส้นทางในเราเตอร์ R1 มีค่าเกตเวย์ประจำเครือข่าย 1 และเครือข่าย 2 เท่ากับ 1.1 และ 2.1 ตามลำดับ ส่วนเกตเวย์สำหรับเครือข่าย 4 จะมีค่าเท่ากับ 3.2 ซึ่งเป็นแอดเดรสประจำอินเทอร์เฟส e1 ของ R2 (R2 เป็นเราเตอร์ขั้นถัดไปของ R1)

การเลือกเส้นทางแบบสแตติกนิยมใช้กับการเชื่อมโยงแบบจุดต่อจุดระหว่างเราเตอร์สองตัว เช่น เครือข่ายมีทางออกไปสู่ภายนอกหรืออินเทอร์เน็ตเพียงช่องทางเดียว มักจะกำหนดเส้นทางแบบสแตติก การเลือกเส้นทางแบบสแตติกมีข้อดีข้อเสียดังนี้

<u>ข้อดี</u>

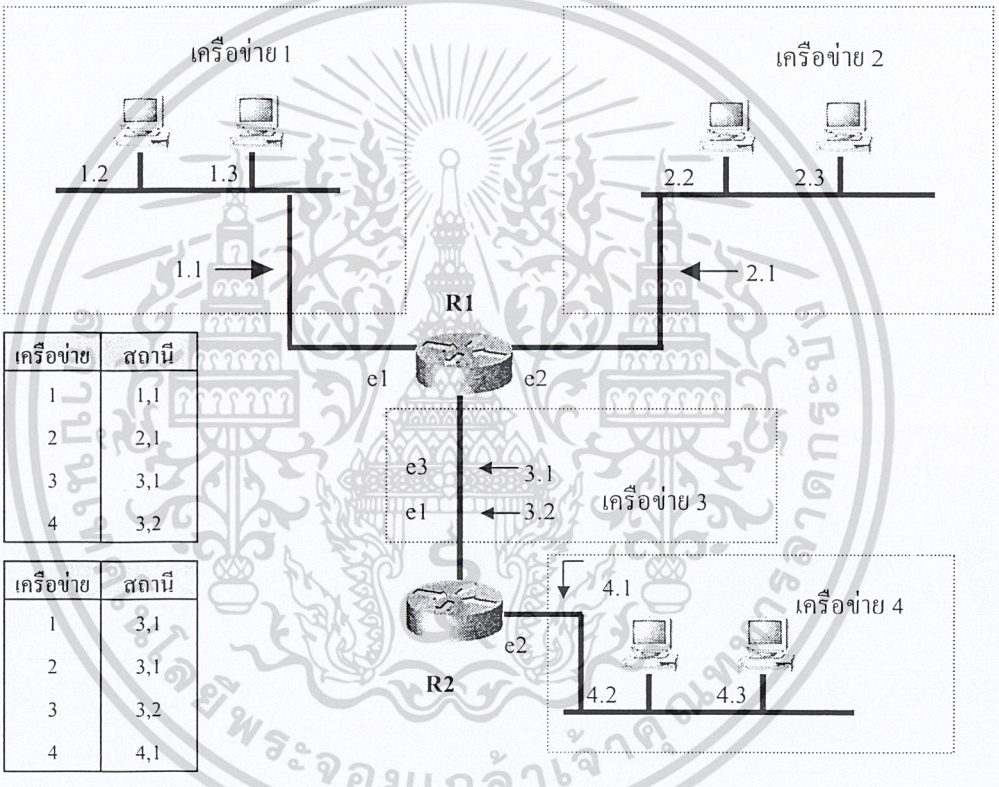
1. สะดวกต่อการใช้งานกับเครือข่ายขนาดเล็ก
2. ไม่ต้องใช้ซอฟต์แวร์เลือกเส้นทาง เราเตอร์ไม่จำเป็นต้องมีซีพียูสมรรถนะสูง
3. ประหยัดแบนด์วิดเครือข่ายเนื่องจากไม่ต้องแลกเปลี่ยนข้อมูลระหว่างเราเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. สามารถจำกัดให้เข้าถึงได้เฉพาะในเครือข่ายที่ต้องการ

ข้อเสีย

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



รูปที่ 5-1 เครือข่ายการแสดงการเลือกเส้นทางแบบสแตติก

การเลือกเส้นทางแบบไดนามิก

การเลือกเส้นทางแบบนี้จะใช้ซอฟต์แวร์ทำหน้าที่แลกเปลี่ยนข้อมูลการเลือกเส้นทางระหว่างเราเตอร์ด้วยกัน โดยใช้โพรโตคอลเลือกเส้นทาง เราเตอร์จะสร้างตารางเลือกเส้นทางจากสภาพเครือข่ายขณะนั้น หากเครือข่ายมีการเปลี่ยนแปลงตารางเส้นทางก็จะเปลี่ยนแปลงตามไปด้วย

การเลือกเส้นทางแบบไดนามิกต้องอาศัยการแลกเปลี่ยนค่าเส้นทางระหว่างเราเตอร์และใช้ซีพียูในเราเตอร์เพื่อสร้างตารางเส้นทาง เราเตอร์ประเภทนี้จึงมักมีราคาสูงกว่าเราเตอร์ที่มีโพรโตคอลแบบสแต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ติกอย่างเดียว เพราะต้องออกแบบซอฟต์แวร์ให้ทำงานตามโพรโตคอลเลือกเส้นทาง และซีพียูต้องมีสมรรถนะสูงพอในการคำนวณหาตารางเส้นทาง

ข้อดีของการการเลือกเส้นทางแบบไดนามิก

1. รองรับขนาดเครือข่ายที่ขยายขึ้นเป็นลำดับได้
2. ตารางเส้นทางเปลี่ยนแปลงค่าเองตามการทำงานของซอฟต์แวร์ เส้นทางใดที่ถูกตัดขาดจะมีการหาเส้นทางใหม่ทดแทน

ประเภทโพรโตคอลเลือกเส้นทางแบบไดนามิก

โพรโตคอลเลือกเส้นทางแบบไดนามิกสามารถจัดแยกประเภทออกได้หลายรูปแบบ ในที่นี้จะกล่าวเพียง 2 รูปแบบคือ

1. โพรโตคอลเกตเวย์ภายนอกและภายใน (Interior Gateway Protocol และ Exterior Gateway Protocol)
2. โพรโตคอลดิสแทนซ์เวกเตอร์และลิงค์สเกต (Distance Vector Protocol และ Link State Protocol)

โพรโตคอลเกตเวย์ภายนอกและภายใน

การจัดแบบนี้แบ่งชนิดโพรโตคอลออกเป็นโพรโตคอลเกตเวย์ภายนอกและโพรโตคอลเกตเวย์ภายใน โพรโตคอลเกตเวย์ภายนอกมีหน้าที่แลกเปลี่ยนข้อมูลเส้นทางระหว่างเครือข่ายที่มีการบริหารงานเป็นอิสระออกจากกัน โดยแต่ละเครือข่ายที่มีการบริหารเป็นอิสระออกจากกันจะเรียกว่า ระบบออโตโนมัส (Autonomous System) แต่ละระบบออโตโนมัสมีหมายเลยประจำของตนเองเรียกว่า เลขระบบออโตโนมัส (Autonomous System Number) หมายเลขนี้สามารถขอได้จากหน่วยงานนิก (NIC) ประจำภูมิภาค เลขระบบออโตโนมัสเป็นค่าที่ระบุว่าข้อมูลเส้นทางที่แลกเปลี่ยนระหว่างเครือข่ายนั้นมาจากที่ใด

อินเทอร์เน็ตในยุคแรกใช้โพรโตคอล อีจีพี (EGP: Exterior Gateway Protocol) เป็นโพรโตคอลเกตเวย์ภายนอก แต่ในปัจจุบันโพรโตคอลที่นิยมใช้ระหว่างเครือข่ายคือบีจีพี (BGP : Border Gateway Protocol) และนำมาใช้งานแทนอีจีพี พีจีพีผ่านการพัฒนามาเป็นลำดับกระทั่งปัจจุบันเป็นรุ่นที่ 4 จึงเรียกว่า บีจีพี-4

โพรโตคอลเกตเวย์ภายในเป็นโพรโตคอลที่ออกแบบเพื่อใช้งานในระบบออโตโนมัสหนึ่งๆ เช่น อาร์ไอพี (RIP : Routing Information Protocol) และ โอเอสพีเอฟ (OSPF: Open Shortest Path First) โพรโตคอลทั้งสองนี้เป็นที่ยอมรับเป็นมาตรฐานสากลและใช้งานอย่างแพร่หลายในเครือข่ายทั่วไป ในขณะที่โพรโตคอลเฉพาะที่ออกแบบโดยบริษัทซิสโก้ (Cisco) คืออีไอจีอาร์พี (EIGRP : Enhanced Interior Gateway Routing Protocol) เป็นอีกโพรโตคอลหนึ่งที่นิยมใช้ตามความแพร่หลายของเราเตอร์ของซิสโก้

เมตริก (Metric)

เมตริกเป็นค่าที่นำมาใช้คำนวณหาว่าเส้นทางใดเหมาะต่อการใช้มากกว่าเส้นทางอื่น ค่าเมตริกที่ใช้อาจเป็นได้ทั้งระยะทาง เวลาหน่วง ความน่าเชื่อถือ หรือความเร็ว โพรโตคอลที่ไม่สลับซับซ้อนอาจจะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เลือกใช้เมตริกเพียงประเภทใดประเภทหนึ่ง เช่นอาจใช้เฉพาะระยะทางซึ่งนับจากจำนวนเราเตอร์ที่ต้องส่งผ่านหรือเรียกว่าจำนวนขั้น (Hop Count)

จำนวนขั้นนับจากจำนวนเราเตอร์ที่ดาต้าแกรมต้องเดินผ่านดังตัวอย่างในรูปที่ 5-2 หากดาต้าแกรมไม่ต้องเดินทางผ่านเราเตอร์จำนวนขั้นจะเท่ากับ 0 หากต้องข้ามเราเตอร์หนึ่งตัว จำนวนขั้นจะเพิ่มขึ้นทีละ 1 แต่โพรโตคอลบางชนิดอาจนับจำนวนขั้นตามจำนวนลิงค์ที่ดาต้าแกรมเดินทางผ่านแทนการนับด้วยจำนวนเราเตอร์ หากเครือข่ายอยู่ติดกับเราเตอร์จะมีจำนวนขั้นเท่ากับ 1 แทนที่จะเท่ากับ 0

โพรโตคอลดิสแทนซ์เวกเตอร์และลิงค์สเทต

การจัดแบ่งโพรโตคอลในแบบนี้อาศัยรูปแบบของข้อมูลที่ส่งผ่านระหว่างเราเตอร์และวิธีที่เราเตอร์สร้างตารางเลือกเส้นทางจากข้อมูลนั้น

เราเตอร์ที่ใช้โพรโตคอลดิสแทนซ์เวกเตอร์อาศัยระยะทางเพื่อกำหนดว่าเส้นทางใดเหมาะสมกว่าเส้นทางอื่น ความหมายของดิสแทนซ์เวกเตอร์คือใช้ระยะทางเป็นค่าเมตริก และแอดเดรสของเครือข่ายปลายทางเป็นเวกเตอร์กำหนดจุดหมายปลายทาง

การทำงานพื้นฐานของโพรโตคอลดิสแทนซ์เวกเตอร์จะส่งข้อมูลเลือกเส้นทางไปยังเราเตอร์ที่อยู่ข้างเคียงทุกตัวอย่างสม่ำเสมอเป็นช่วงเวลา ข้อมูลเลือกเส้นทางประกอบด้วยตารางเลือกเส้นทางของตนเองทั้งหมดที่กำกับด้วยเมตริก เราเตอร์แต่ละตัวจะใช้ตารางเส้นทางของตัวเองร่วมกับตารางเส้นทางที่ได้รับมาใหม่เพื่อคำนวณหาระยะทางที่สั้นที่สุด โพรโตคอลดิสแทนซ์เวกเตอร์ที่นิยมใช้ในปัจจุบันได้แก่ อาร์ไอพี ค่าเมตริกที่ใช้คือจำนวนขั้น อาร์ไอพีถือว่าเส้นทางที่ดีที่สุดคือเส้นทางที่มีจำนวนขั้นน้อยที่สุด

โพรโตคอลลิงค์สเทตไม่ได้แลกเปลี่ยนตารางเส้นทางโดยตรงเหมือนกับที่ใช้ในดิสแทนซ์เวกเตอร์ หากแต่เราเตอร์แต่ละตัวจะตรวจสอบ สถานะลิงค์ (Link State) ที่เชื่อมไปยังเราเตอร์ข้างเคียงว่าใช้งานได้หรือไม่พร้อมกับค่าเมตริกซึ่งโดยทั่วไปแล้วเป็นความเร็วของสายสื่อสาร เช่นให้ค่า 1 สำหรับสายสื่อสารที่มีความเร็ว 2 เมกะบิตต่อวินาที และ 10 สำหรับสายสื่อสารที่มีความเร็ว 9600 บิตต่อวินาที โพรโตคอลลิงค์สเทตที่นิยมใช้คือโอเอสพีเอฟ

5.4 อาร์ไอพี(RIP)

อาร์ไอพีมีกำเนิดมาจากโพรโตคอลเลือกเส้นทางในระบบเครือข่ายซีร็อกซ์ซึ่งต่อมาได้พัฒนาเป็นอาร์ไอพีและโพรโตคอลอื่นได้แก่ไอพีเอกซ์ (โนเวลล์) และไอจีอาร์พี (ซิสโก้) อาร์ไอพีเป็นโพรโตคอลแบบดิสแทนซ์เวกเตอร์และใช้ระยะทางเป็นค่าเมตริกเพื่อหาเส้นทางที่ดีที่สุดสำหรับการเลือกเส้นทาง

อาร์ไอพีเป็นโพรโตคอลแบบดิสแทนซ์เวกเตอร์ ขั้นตอนพื้นฐานที่อาร์ไอพีใช้คือขั้นตอนวิธีของฟอร์ด/ฟูเกอร์สัน หรือเรียกอีกชื่อว่าเบลแมน-ฟอร์ด เราเตอร์อาร์ไพพีจะเก็บตารางเส้นทางซึ่งประกอบด้วยแอดเดรสเครือข่าย แอดเดรสของเราเตอร์ถัดไปซึ่งเป็นเกตเวย์ไปยังเครือข่ายนั้น และเมตริกประจำเส้นทางซึ่งโดยปกติจะนับตามจำนวนเราเตอร์ระหว่างทาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.4.1 การทำงานของอาร์ไอพี

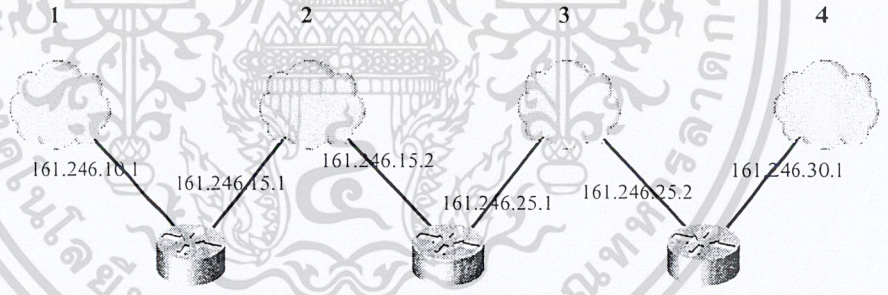
เริ่มแรกเราเตอร์แต่ละตัวจะได้รับการกำหนดแอดเดรสเครือข่ายประจำแต่ละอินเทอร์เฟสของเราเตอร์นั้น ซึ่งเราเตอร์ก็จะสร้างตารางเส้นทางจากแอดเดรสเครือข่ายที่มี แล้วส่งตารางเส้นทางของตัวเองไปให้เราเตอร์ข้างเคียงด้วยการบรอดคาสต์

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



รูปที่5-2 เครือข่ายสาธิตการทำงานของอาร์ไอพี

ในตอนแรกที่เราเตอร์ทั้ง 3 ตัวเริ่มทำงาน ค่าในตารางเส้นทางของเราเตอร์แต่ละตัวที่เกิดจากการติดตั้งค่าของผู้ดูแลเครือข่ายจะเป็นเครือข่ายที่เชื่อมต่อกับเราเตอร์นั้นโดยตรงดังนี้

ตาราง R1

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตาราง R2

เครือข่าย	เกตเวย์	อินเทอร์เฟส	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	อินเทอร์เฟส	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

อินเทอร์เฟส : อินเทอร์เฟสของเราเตอร์

เมตริก : จำนวนขั้น

ในตอนแรกเราเตอร์จะมีตารางเส้นทางเฉพาะเครือข่ายที่อยู่ติดกับเราเตอร์ แต่เส้นทางไปเครือข่ายอื่นนั้นจะได้จากการแลกเปลี่ยนตารางเส้นทางกับเราเตอร์ตัวอื่น 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 จะเพิ่มค่าเข้าไปในตาราง โดยมองว่า “เราเตอร์ R2 สามารถไปถึงเครือข่าย 161.246.10.0 ได้โดยผ่านเราเตอร์ R1 ซึ่งถ้าเราเตอร์ R1 สามารถไปยังเครือข่าย 161.246.10.0 ได้ด้วยระยะทาง 1 จำนวนขั้น ดังนั้นถ้าเราเตอร์ R2 สามารถไปเครือข่ายนั้นโดยผ่านเราเตอร์ R1 ก็สามารถไปถึงได้ด้วยระยะทางที่เราเตอร์ R1 ไป + 1 จึงเท่ากับ 1+1 = 2 จำนวนขั้น” และใช้เกตเวย์ 161.246.15.1 เป็นทางออก ตาราง ของ R2 หลังจากได้รับตารางเส้นทางที่ R1 บรอดแคสต์ให้จะเป็น

เครือข่าย	เกตเวย์	อินเทอร์เฟส	เมตริก	
161.246.10.0	161.246.15.1	e1	2	← ค่าที่ได้ใหม่
161.246.15.0	0.0.0.0	e1	1	
161.246.25.0	0.0.0.0	e2	1	

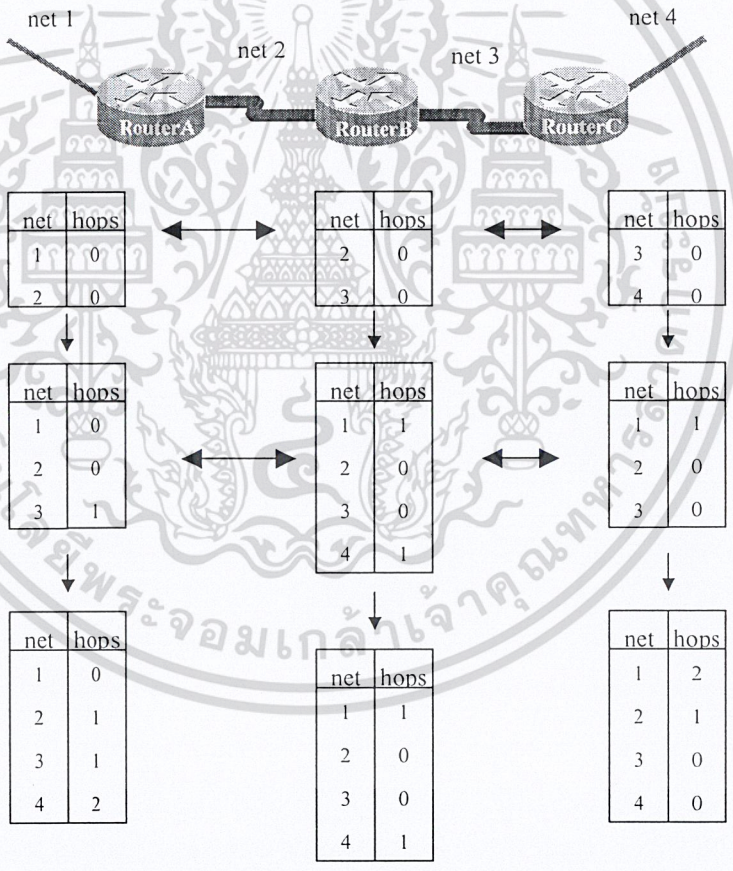
ถ้าต่อมา R3 บรอดคาสต์เฉพาะเราเตอร์ R2 ก็จะได้รับค่าและเปลี่ยนค่าในตารางเช่นเดียวกัน ดังนั้นตารางเส้นทางของเราเตอร์ R2 จะเป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครือข่าย	เกตเวย์	อินเทอร์เฟส	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 หลังจากที่ได้รับบรอดคาสต์จาก R1 และ R3 ตารางของ R2 ก็จะมีข้อมูลของเครือข่ายครบทุกเครือข่าย รูปที่ 5-3 แสดงการประกาศและปรับค่าตารางเส้นทางของเราเตอร์ทั้งหมด โดยสมมติให้แต่ละเราเตอร์ประกาศค่าตารางพร้อมกัน



รูปที่ 5-3 การประกาศค่าและปรับค่าและตารางของอาร์ไอพี

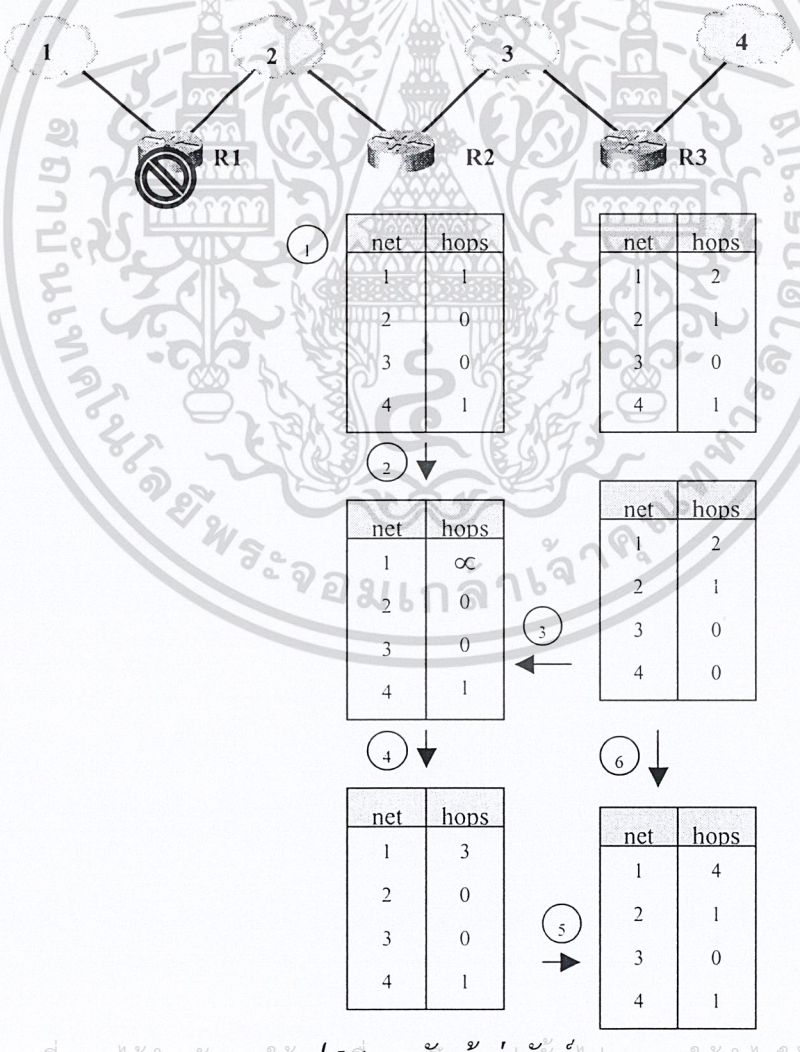
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.4.2 การปรับค่าเมื่อเครือข่ายเปลี่ยน

กรณีที่โครงสร้างของเครือข่ายไม่เปลี่ยนแปลง ตารางเส้นทางของอาร์ไอพีก็จะมีเสถียรภาพถึงแม้ว่าเราเตอร์จะประกาศตารางออกไปทุกๆ 30 วินาที จึงเป็นการตรวจสอบสภาพเส้นทางและพร้อมที่จะปรับตารางเส้นทางให้เหมาะสมกับสภาพของเครือข่ายถ้าเส้นทางเดิมนั้นมีปัญหา

เมื่อพิจารณาในทางปฏิบัติแล้วเราเตอร์อาร์ไอพีใดๆ จะไม่สามารถปรับตารางเส้นทางใหม่ได้หากเราเตอร์ข้างเคียงไม่ประกาศค่าเส้นทางให้ ซึ่งอาจเป็นเพราะเราเตอร์ข้างเคียงตัวนั้นไม่สามารถทำงานได้หรือหยุดทำงานชั่วคราว อาร์ไอพีจึงหาวิธีแก้ปัญหานี้โดยการกำหนดให้เส้นทางทุกเส้นทางมีอายุการใช้งานปกติจะกำหนดไว้ที่ 180 วินาที ถ้าเส้นทางใดไม่ได้รับการประกาศค่าเส้นทางใดๆ เป็นเวลาตามที่กำหนดไว้ จะถือว่าเส้นทางนั้นเป็นเส้นทางทีใช้ไม่ได้อีกต่อไป และเราเตอร์จะเปลี่ยนเมตริกของเส้นทางนั้นให้เป็นอนันต์ แต่จะยังไม่ลบเส้นทางนั้นออกจากตาราง หากมีการประกาศเส้นทางนั้นมาจากเราเตอร์อื่น เราเตอร์จะปรับเส้นทางให้ใช้เส้นทางตามที่ได้รับจากเราเตอร์อื่นนั้น

ปัญหาการนับเข้าสู่อนันต์



รูป 5.4 การนับเข้าสู่อนันต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานของอาร์ไอพีอาจมีปัญหาเวลาที่เราเตอร์ R พบเส้นทางไปยังเครือข่ายหนึ่งผิดปกติ ในขณะเดียวกันเราเตอร์ข้างเคียงตัวหนึ่งแจ้งว่ามีเส้นทางไปยังเครือข่ายที่เกิดผิดปกติ เราเตอร์ R ที่พบปัญหาก็จะปรับค่าเส้นทางตามค่าที่ได้รับมาใหม่ แต่เนื่องจากเป็นเส้นทางที่เราเตอร์ข้างเคียงสามารถไปถึงเครือข่ายนั้นได้โดยผ่านทางเราเตอร์ R ทำให้เกิดการปรับเมตริกต่อไปแบบไม่มีที่สิ้นสุด ปรากฏการณ์นี้เราเรียกว่า การนับเข้าสู่อนันต์ (Count to Infinity)

ให้ตารางเส้นทางที่เราเตอร์ R2 และ R3 มีค่าดังตำแหน่ง ในจังหวะที่เราเตอร์ R1 ทำงานผิดปกติ ทำให้เครือข่าย 1 จะถูกตัดขาดออกไป ค่าเส้นทางของเครือข่าย 1 ในR2 จะลดลงตามการทำงานจองตัวจับเวลา 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 ซึ่งอาร์ไอพีได้กำหนดค่าเมตริกไว้ค่าหนึ่งเพื่อแก้ปัญหานี้ ค่าเมตริกนี้ควรเป็นค่าที่ไม่สูงมากเพราะจะได้ไม่ต้องใช้เวลานาน แต่ถ้าน้อยเกินไปก็อาจทำให้เครือข่ายถูกจำกัดให้เล็กลง ซึ่งปกติจะกำหนดไว้ที่ 16 เมื่อเราเตอร์เพิ่มค่าไปถึง 16 ก็จะทราบว่าเส้นทางนั้นเป็นเส้นทางที่ไม่สามารถไปถึงได้ และจะเข้าสู่กระบวนการจับเวลาเพื่อกำจัดเส้นทางออกจากตาราง

เวลาลู่เข้า

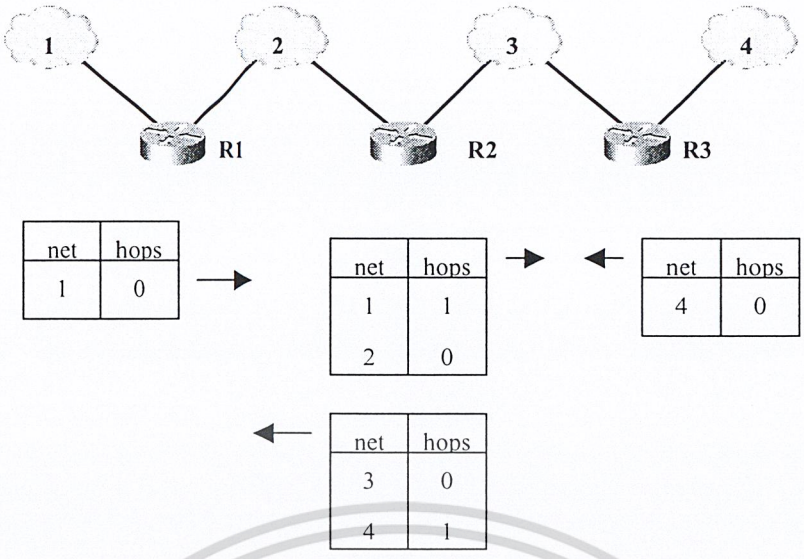
เวลาลู่เข้าของอาร์ไอพีหมายถึงเวลาที่เราเตอร์ใช้ปรับตารางจนกระทั่งตารางมีค่าถูกต้องตามโครงสร้างของเครือข่ายจริง จากข้อกำหนดของการประกาศค่าของเราเตอร์อาร์ไอพีทุกๆ 30 วินาที ปัญหาการนับเข้าสู่อนันต์ในเครือข่ายที่ใช้อาร์ไอพีจะใช้เวลาสูงสุดประมาณ 7 นาที

หากเวลาลู่เข้าของอาร์ไอพีสั้นลง เครือข่ายก็ย่อมที่จะเข้าสู่เสถียรภาพได้เร็วขึ้น วิธีปรับเวลาลู่เข้าของอาร์ไอพีให้สั้นลงมีหลายวิธี ดังนี้

สปลิตฮอไรซัน(Split Horizon)

วิธีนี้จะช่วยแก้ปัญหานับเข้าสู่อนันต์ (เฉพาะเครือข่ายบางรูปแบบเท่านั้น) โดยเราเตอร์จะไม่ประกาศตารางเส้นทางทั้งตารางให้กับเราเตอร์ข้างเคียงแต่จะประกาศเฉพาะเส้นทางที่ไม่ได้เรียนรู้มาจากเราเตอร์ตัวนั้นเท่านั้น เช่นเราเตอร์ R2 ได้รับตารางเส้นทางไปเครือข่าย 1 มาจากเราเตอร์ R1 เราเตอร์ R2 ก็จะประกาศเฉพาะเส้นทางของเครือข่าย 3 และ 4 ไปให้กับ R1 ดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่5-5 การประกาศเส้นทางด้วยวิธีสปลิตฮอไรซัน

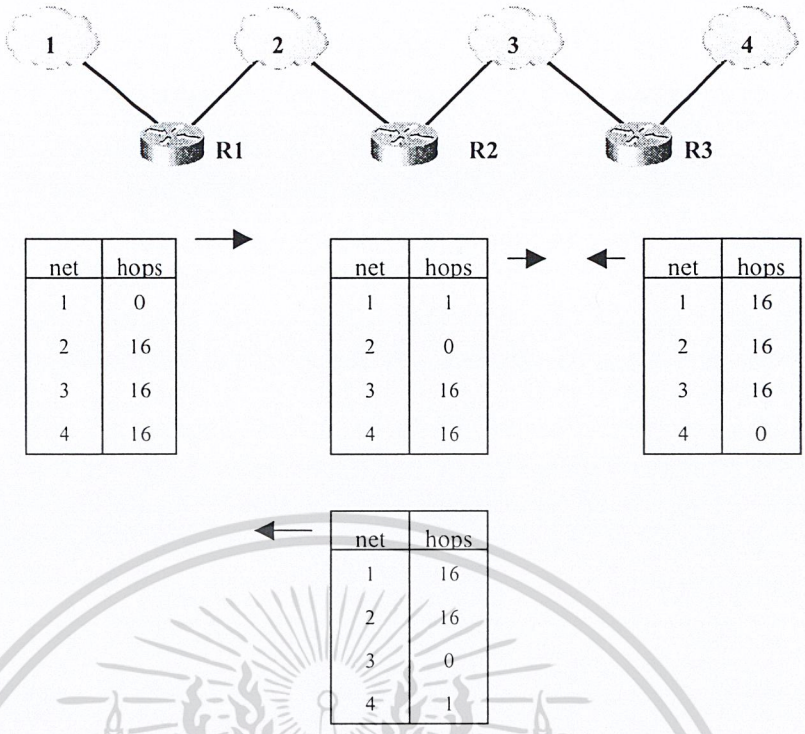
หากเราเตอร์ R1 หยุดทำงานไป ค่าเส้นทางของเครือข่าย 1 ใน R2 จะค่อยๆ ลดลงจนครบ 180 วินาที เส้นทางไปเครือข่าย 1 จะเปลี่ยนเป็นอนันต์ โดยที่ R2 จะไม่ได้รับการประกาศเส้นทางไปเครือข่าย 1 จาก R3 เนื่องจากเป็นเส้นทางที่ R2 ประกาศให้ R3 รู้วิธีไปยังเครือข่าย 1 เอง ดังนั้นจึงไม่เกิดปัญหานับเข้าสู่อนันต์

สปลิตฮอไรซันแบบพอยซันรีเวอร์ส (Split Horizon with Poisoned Reverse)

วิธีนี้เป็นเทคนิคอีกแบบหนึ่งของสปลิตฮอไรซันที่ใช้วิธีแจ้งค่าเส้นทางที่ชัดเจนมากยิ่งขึ้น เพื่อไม่ให้เกิดการนับเข้าสู่อนันต์

วิธีนี้จะยอมให้เราเตอร์ประกาศเส้นทางที่เรียนรู้จากลิงค์หนึ่งๆ กลับไปได้ แต่ให้เมตริกประจำเส้นทางนั้นเป็นอนันต์หรือเท่ากับ 16 เพื่อกำกับว่าเส้นทางนั้นไม่สามารถใช้งานได้ เช่น มีเราเตอร์ต่อกันดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่5-6 การประกาศเส้นทางด้วยวิธีสปลิตฮอไรซันแบบพอยซันรีเวอร์ส

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

วิธีนี้จะสร้างแพ็กเก็ตมากกว่าวิธีสปลิตฮอไรซันแบบธรรมดา เนื่องจากมีค่าที่ต้องประกาศมากกว่า แต่ก็เป็นวิธีที่เราเตอร์ในปัจจุบันนิยมใช้เนื่องจากให้ประสิทธิผลดีกว่า

เทคนิคของสปลิตฮอไรซันทั้งสองแบบนี้ไม่สามารถป้องกันปัญหานับเข้าสู่อนันต์ได้ทุกรูปแบบเครือข่าย ช่วยแก้ปัญหาสำหรับเราเตอร์สองตัวที่อยู่ข้างเคียงกันเท่านั้น

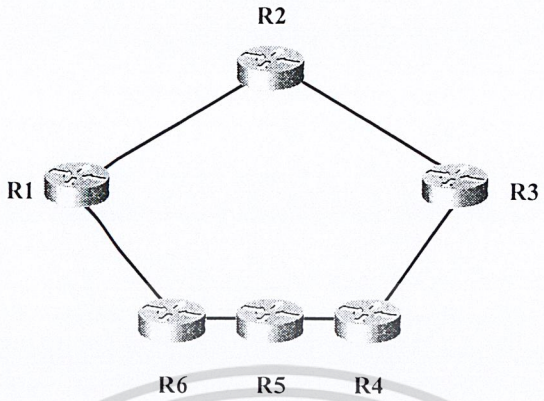
ทริกเกอร์อัพเดต

สปลิตฮอไรซันทั้ง 2 วิธีนั้นช่วยแก้ปัญหาการนับเวลาเข้าสู่อนันต์เพื่อลดเวลาลู่เข้า หากแต่ยังต้องใช้เวลารู่หนึ่งก่อนที่จะตรวจสอบว่าเส้นทางหนึ่งๆ มีปัญหา ทริกเกอร์อัพเดตเป็นเทคนิคเสริมที่ช่วยลดเวลาลู่เข้าของอาร์ไอพี คือกำหนดให้เราเตอร์ประกาศค่าเส้นทางออกไปทันทีที่พบว่าโครงสร้างเครือข่ายมีการเปลี่ยนแปลงโดยไม่ต้องรอให้ครบเวลา 30 วินาที

หากเราเตอร์ทุกตัวใช้ทริกเกอร์อัพเดตก็จะสามารถปรับปรุงตารางได้อย่างรวดเร็ว แต่ก็อาจจะสร้างภาระในเครือข่ายขนาดใหญ่ที่มีเราเตอร์หลายตัว เพราะการเปลี่ยนแปลงใดๆ ก็ย่อมทำให้เราเตอร์ตัวหนึ่งส่งข้อมูลให้เราเตอร์ตัวอื่นประกาศค่าตามกันไปอย่างต่อเนื่องหรือเกิดแพ็กเก็ตบรอดแคสต์เป็นจำนวนมาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แต่เทคนิคนี้อาจทำให้เกิดเหตุการณ์ที่ไม่คาดฝันขึ้นได้ เช่นในกรณีที่มีเครือข่ายดังรูป



รูปที่ 5-7 เครือข่ายที่จำเป็นต้องใช้โฮลดาวน์ร่วมกับทริกเกอร์อัพเดต

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

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

โฮลดาวน์กำหนดช่วงเวลาให้เราเตอร์งดปรับเปลี่ยนค่าที่เพิ่งจะตรวจพบว่าไม่สามารถใช้งานได้ จากตัวอย่างข้างต้นนี้ เมื่อ R1 ปรับเส้นทางระหว่าง R3 กับ R4 ว่าไม่สามารถใช้งานได้กฎโฮลดาวน์จะบังคับให้ R1 รอเวลาโดยไม่รับการแจ้งค่าว่าเส้นทางระหว่าง R3 กับ R4 สามารถใช้งานได้จากเราเตอร์อื่นเป็นเวลาหนึ่ง โดยช่วงเวลานี้ควรเป็นช่วงเวลาที่มากพอจะให้ทริกเกอร์อัพเดตกระจายไปยังเราเตอร์ทุกตัว ซึ่งการใช้โฮลดาวน์ถึงแม้ว่าจะทำให้ช่วงเวลาลู่เข้านานกว่าแบบไม่ใช้ แต่ทำให้การทำงานของระบบดีกว่าแบบไม่ใช้ด้วย

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

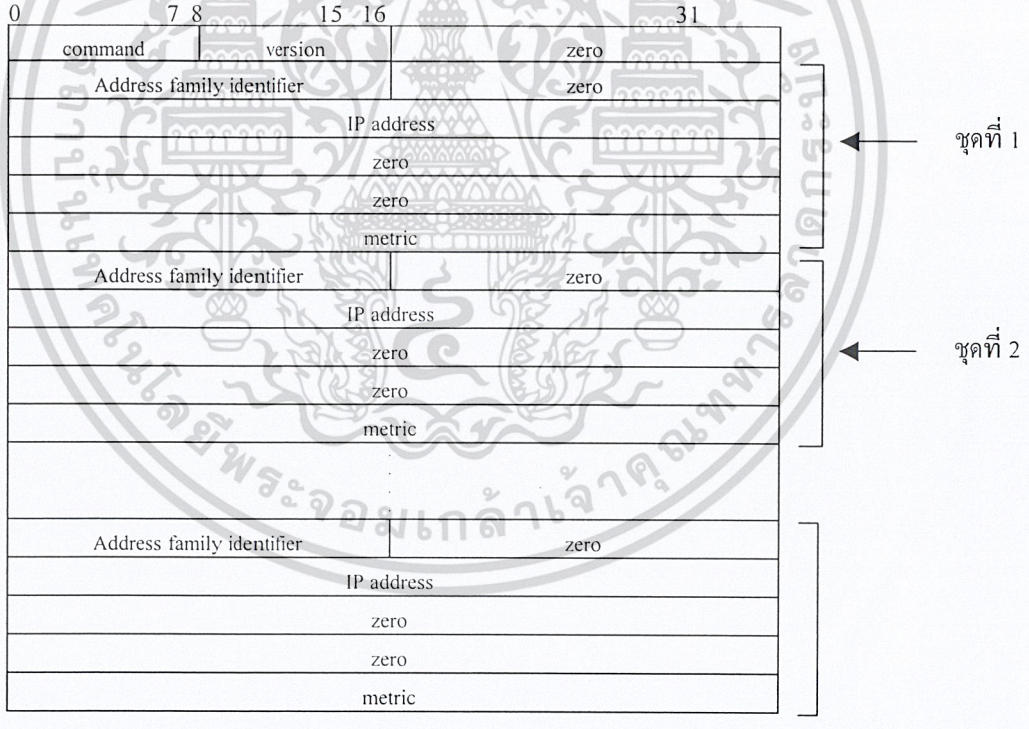
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วิธีนี้จึงช่วยลดโอกาสเกิดการบรอดแคสต์แพ็กเก็ตพร้อมกันเป็นจำนวนมากลงไป นอกจากนี้เพื่อป้องกันการกลับมาเข้าจังหวะของเราเตอร์เมื่อเกิดทริกเกอร์อัพเดต เราเตอร์จะต้องไม่รีเซ็ตตัวจับเวลาให้กลับมาเป็นศูนย์เมื่อเกิดทริกเกอร์อัพเดต แต่ให้จับเวลาต่อไปตามปกติและบรอดคาสต์ออกไปเมื่อถึงเวลา

2. **ตัวจับเวลาหมดอายุ (Expiratin 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 ฟอร์แมตของเฟรมอาร์ไอพี

รูปแบบเฟรมในที่นี้แสดงเฉพาะการใช้อาร์ไอพีในเครือข่ายไอพีเท่านั้น หากเป็นเครือข่ายอื่นจะมีรูปแบบแตกต่างออกไป โดยแต่ละฟิลด์มีความหมายดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- command ขนาด 8 บิต : กำหนดแบบการทำงานว่าเป็นการร้องขอ (ค่าเท่ากับ 1) หรือการตอบรับ (ค่าเท่ากับ 2) การบรอดคาสต์ของอาร์ไอพีจัดเป็นการตอบรับ (เพราะไม่มีการร้องขอมาก่อน)

อาร์ไอพีจะกำหนดคำสั่งให้เป็น 1 เพื่อร้องขอต่อเมื่อต้องการทราบตารางเส้นทางของเราเตอร์ใดเป็นกรณีเฉพาะ

- version ขนาด 8 บิต : กำหนดรุ่นของโพรโตคอล หากเท่ากับ 1 หมายถึงรุ่น 1
- zero ขนาด 16 บิต : สงวนไว้และต้องมีค่าเป็น 0
- address family indentier ขนาด 16 บิต : การออกแบบอาร์ไอพีในขั้นต้นไม่ได้กำหนดให้ใช้เฉพาะกับทีซีพี/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 ค่าในดาต้าแกรมเดียว

เฟรมอาร์ไอพี 2

อาร์ไอพีรุ่น 2 มีส่วนขยายการทำงานเพิ่มเติมจากอาร์ไอพีรุ่นแรกและเพิ่มระบบความปลอดภัยในการส่งข้อมูลเส้นทางระหว่างเราเตอร์ อาร์ไอพีรุ่น 1 ไม่สนับสนุนการใช้งานในเครือข่ายที่มีซับเน็ตแบบแปรค่า ไม่สามารถแจ้งหมายเลขระบบออโตโนมัสเพื่อเชื่อมความปลอดภัยในการส่งตาราง อาร์ไอพีรุ่น 2 จึงได้รับการพัฒนาขึ้นเพื่อรักษาความปลอดภัยในการส่งตาราง อาร์ไอพีรุ่น 2 จึงได้รับการพัฒนาขึ้นเพื่อแก้ปัญหาดังกล่าวและกลายเป็นมาตรฐานใหม่ทดแทนอาร์ไอพีรุ่น 1

ฟิลด์อาร์ไอพีรุ่น 2 ยังคงลักษณะของอาร์ไอพีรุ่น 1 แต่ได้เพิ่มเติมฟิลด์ใหม่ และมีรูปแบบดังรูป 5.10

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 : ฟิลด์กำกับชุดแอดเดรสเพื่อแยกแยะชนิดของโพรโตคอลที่อาร์ไอพีทำงานร่วมด้วย สำหรับไอพีใช้ค่าเท่ากับ 2

Route tag ขนาด 8 บิต : ใช้แยกความแตกต่างระหว่างการได้มาของค่าเส้นทางว่าเรียนรู้จากภายในหรือภายนอก ค่าที่ใช้อาจเป็นเลขระบบออโตโนมัสสำหรับเส้นทางที่ได้มาจากโพรโตคอลภายนอก

IP Address ขนาด 32 บิต : ไอพีแอดเดรสที่กำหนดเส้นทาง

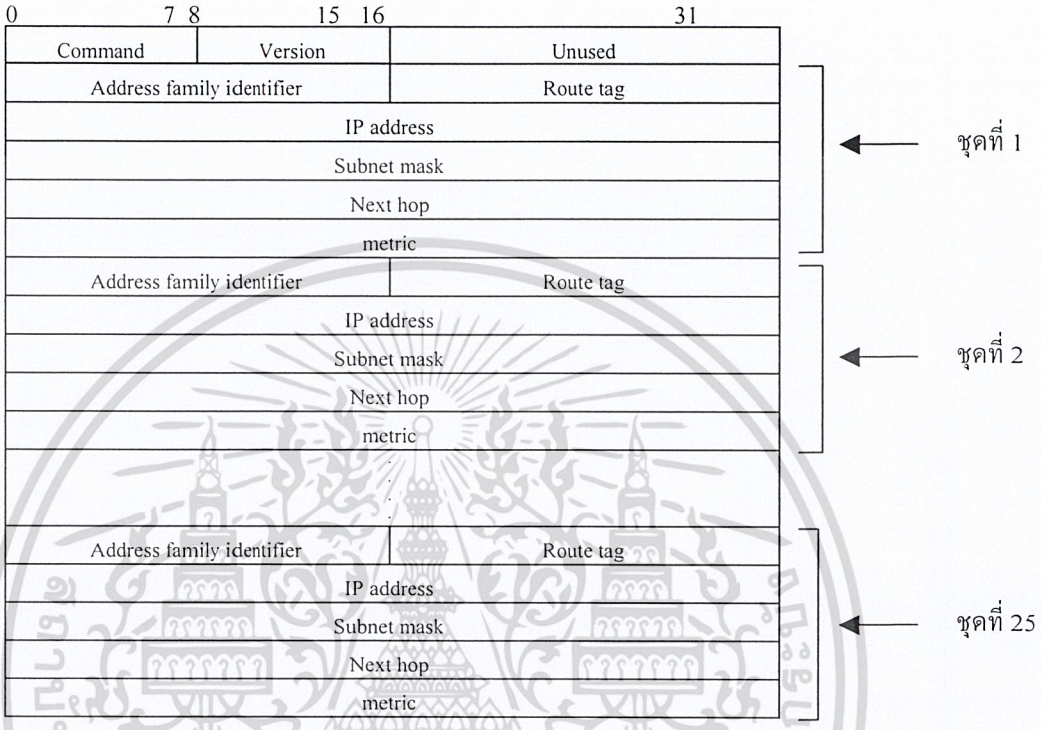
Subnet Mask ขนาด 32 บิต : ซับเน็ตมาสก์กำกับไอพีแอดเดรส หากมีค่าเป็น 0 หมายถึงไม่ได้กำหนดซับเน็ตมาสก์ให้

Next Hop ขนาด 32 บิต : ไอพีแอดเดรสของเราเตอร์ถัดไปที่ควรจะนำส่งแพ็กเก็ตซึ่งมีแอดเดรสปลายทางตามที่กำหนดด้วยค่าเลือกเส้นทางในเฟรมอาร์ไอพีนี้ ประโยชน์ของฟิลด์นี้ใช้เพื่อลดจำนวนเส้นทางที่แพ็ก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น สงวนลิขสิทธิ์โดยมหาวิทยาลัยเกษตรศาสตร์

เก็ตต้องเดินทางผ่านในกรณีที่มีโพรโตคอลเลือกเส้นทางอื่นนอกเหนือไปจากอาร์ไอพีทำงานร่วมกัน หาก มีค่า 0.0.0.0 หมายถึงให้ส่งแพ็กเก็ตไปยังเราเตอร์ที่ประกาศค่าเส้นทางนี้มาให้

Metric ขนาด 32 บิต : เมตริกกำหนดระยะทางยังคงมีค่าได้ตั้งแต่ 1 ถึง 15 เช่นเดียวกับอาร์ไอพีรุ่น 1การพิสูจน์ตัวจริง



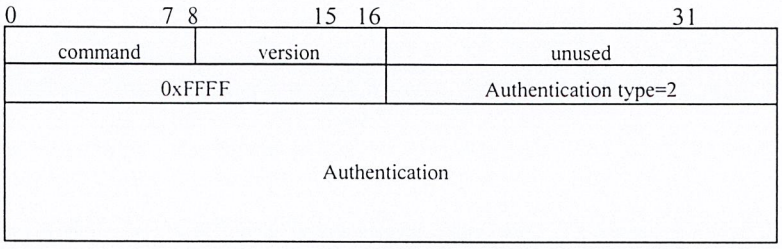
รูปที่ 5-9 ฟอร์แมตของเฟรมอาร์ไอพีเวอร์ชัน 2

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5-10 เฟรมอาร์ไอพีเวอร์ชัน 2 เมื่อใช้การพิสูจน์ตัวจริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

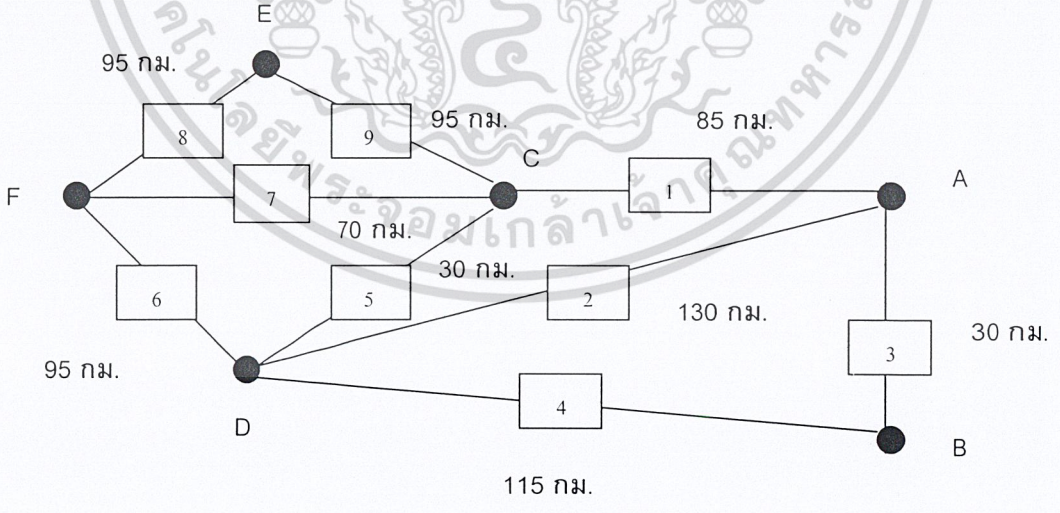
โอเอสพีเอฟ (OSPF : Open Shortest Path First)

โอเอสพีเอฟเป็นโพรโตคอลเกตเวย์ภายในที่นิยมใช้อย่างแพร่หลายเนื่องจากสามารถปรับเปลี่ยนการเลือกเส้นทางตามสภาพเครือข่ายได้รวดเร็วและรองรับการทำงานกับเครือข่ายขนาดใหญ่ได้

หลักการพื้นฐานของโพรโตคอลแบบโอเอสพีเอฟ เป็นโพรโตคอลแบบลิงค์เสตท โดยโอเอสพีเอฟจะทำหน้าที่สร้างแผนที่ให้กับเราเตอร์ 3 ขั้นตอนคือ ขั้นแรกเราเตอร์จะตรวจหาเราเตอร์อื่นที่อยู่ข้างเคียง ขั้นที่สองเราเตอร์จะประกาศค่านี้ออกไปให้เราเตอร์อื่นทุกตัว เมื่อเสร็จสิ้นขั้นตอนนี้ เราเตอร์จะทราบว่ามีเราเตอร์อื่นใดอยู่บ้าง ขั้นสุดท้ายเราเตอร์จะอาศัยข้อมูลจากขั้นที่สองสร้างแผนที่ไปยังเราเตอร์อื่นทุกตัวในเครือข่าย เราเตอร์ทุกตัวต่างก็มีแผนที่ประจำตัวที่ทราบถึงเส้นทางทั้งหมดในเครือข่าย

6.1 การทำงานของโอเอสพีเอฟโดยสรุป

เพื่อให้สามารถศึกษาการทำงานของโอเอสพีเอฟได้อย่างชัดเจนเป็นขั้นตอนจะใช้เส้นทางจากเมืองหนึ่งไปยังอีกเมืองหนึ่งดังรูป 6.1 แต่ละเมืองเปรียบเสมือนเราเตอร์และทางหลวงที่เชื่อมต่อกันถือเป็นสายสื่อสารระหว่างเราเตอร์ เส้นทางในรูปเป็นเพียงเส้นทางที่จำลองจากทางหลวงระหว่างเมืองและระยะทางวัดเป็นกิโลเมตรโดยประมาณ หมายเลยประจำเส้นทางตั้งแต่ 1-9 ใช้เรียกชื่อแทนหมายเลขทางหลวงเพื่อความสะดวกต่อการอ้างอิง

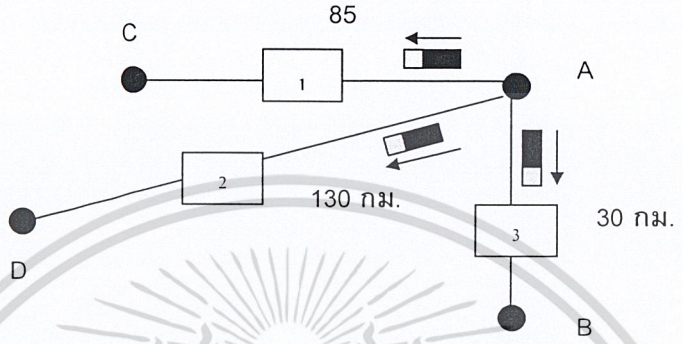


รูปที่ 6-1 แผนที่ทางหลวง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.2 เรียนรู้เครือข่าย

เราเตอร์ในเครือข่ายที่ใช้โอเอสพีเอฟจะเริ่มต้นเรียนรู้ว่ามีเราเตอร์ใดที่อยู่ข้างเคียงโดยการส่งแพ็กเก็ต “ทักทาย” หรือ ฮัลโหลแพ็กเก็ต (Hello Packet) ออกไปทุกๆลิงค์ของเราเตอร์ ตัวอย่างรูปที่ 6-2 เราเตอร์ A จะส่งแพ็กเก็ตทักทายไปยังเราเตอร์ที่ B,C และ D ในขณะที่เราเตอร์ทั้งสามก็ส่งแพ็กเก็ตทักทายไปยังเราเตอร์ A ด้วย



รูปที่ 6-2 เราเตอร์ A ส่งแพ็กเก็ตทักทาย

เมื่อเราเตอร์ A ได้รับแพ็กเก็ตตอบกลับจาก B,C และ D เราเตอร์ A ก็จะทราบถึงเส้นทางข้างเคียงโดยตรง เราเตอร์ A จะสร้างสร้างตารางระยะทางไปยังเราเตอร์ข้างเคียงตามตารางที่ 6.1 ในขณะเดียวกันและในที่สุดเราเอตร์แต่ละจังหวัดต่างก็จะมีข้อมูลระยะทางไปยังเราเตอร์ข้างเคียงทุกตัว

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 หากเป็นแพ็กเก็ตใหม่จะต้องปล่อยแพ็กเก็ตไปทุกอินเทอร์เฟสยกเว้นอินเทอร์เฟสที่ได้รับแพ็กเก็ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปแบบของแพ็กเก็ตที่เราเตอร์ส่งออกไปในขั้นนี้เรียกว่า **แพ็กเก็ตประกาศลิงค์เสตท** (Link State Advertisement) หรือ **แพ็กเก็ตแอลเอสเอ** (LSA) เมื่อเราเตอร์ทุกตัวปล่อยแพ็กเก็ตแอลเอสเอเช่นเดียวกัน เพียงชั่วระยะหนึ่งแพ็กเก็ตจะกระจายไปทั่วทั้งเครือข่าย ท้ายที่สุดแล้วเราเตอร์ทุกตัวจะทราบเส้นทางไปยังเราเตอร์อื่นและได้ตารางเส้นทางเรียกว่า ฐานข้อมูลลิงเสตท (Link State Database) จากตัวอย่างนี้ฐานข้อมูลลิงค์เสตทมีค่าตามตารางที่ 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 ฐานข้อมูลลิงสเทตประจำแต่ละเราเตอร์

6.4 การคำนวณเส้นทาง

ฐานข้อมูลที่ได้จากขั้นที่แล้วยังไม่ใช่ตารางเส้นทางที่นำไปใช้ได้ ตารางเส้นทางจะได้จากการคำนวณหาเส้นทางที่สั้นที่สุดตามขั้นตอนวิธีของไดจ์สตรา (Dijkstra Algorithm)

6.4.1 ขั้นตอนวิธีของไดจ์สตรา (Dijkstra Algorithm)

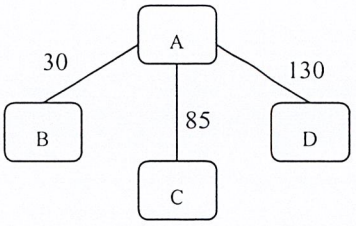
ตามขั้นตอนวิธีไดจ์สตรา เราเตอร์จะสร้างเส้นทางที่เขียนแทนด้วยต้นไม้กำกับทิศทางโดยมีกิ่งเชื่อมไปยังเราเตอร์อื่น รากของต้นไม้ก็คือเราเตอร์ที่เป็นจุดตั้งต้นคำนวณหาเส้นทาง แต่ละเส้นจะกำกับด้วยเลเบลซึ่งแบ่งเส้นทางออกเป็นสองชนิดคือ **เส้นทางชั่วคราว** และ **เส้นทางถาวร** เส้นทางชั่วคราวใช้กำกับเส้นทางที่ค้นพบแต่ยังไม่ผ่านการคำนวณว่าเป็นเส้นทางที่สั้นที่สุด ส่วนเส้นทางถาวรใช้กำกับเส้นทางที่คำนวณแล้วว่าสั้นที่สุด

เมื่อเริ่มต้นทำงานที่เราเตอร์ใด เราเตอร์นั้นจะมีระยะทางถึงตัวเองเป็นศูนย์และสร้างเส้นทางชั่วคราวไปยังเราเตอร์ข้างเคียงก่อน ส่วนเส้นทางไปยังเราเตอร์อื่นให้มีค่าเป็นอนันต์

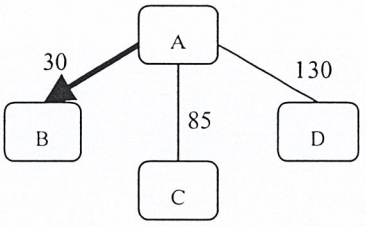
เมื่อใช้ขั้นตอนวิธีของไดจ์สตรากับรูปที่ 6-1 โดยพิจารณาเราเตอร์ A จะได้เส้นทางแรกตามรูปที่ 6-3 (1) โดยมีเส้นทางไปยัง B,C และ D ทั้งสามเส้นทางไม่มีหัวลูกศรซึ่งหมายถึงเส้นทางชั่วคราว ลำดับขั้นที่ (2) ถึง (6) แสดงการหาเส้นทางที่สั้นที่สุดไปยังที่อื่นๆ โดยเส้นทางที่มีหัวลูกศรหมายถึงเส้นทางถาวร

เมื่อเสร็จสิ้นขั้นตอนสุดท้ายแล้วเราเตอร์ A จะมีแผนที่ทั้งเครือข่าย เมื่อต้องการไปยังเส้นทางใดก็เพียงแต่หาเส้นทางจากต้นไม้ที่สร้างขึ้นเท่านั้น ในขณะที่เราเตอร์อื่นก็จะดำเนินการตามขั้นตอนเพื่อหาระยะทางที่สั้นที่สุดไปยังที่อื่นๆในลักษณะเดียวกัน

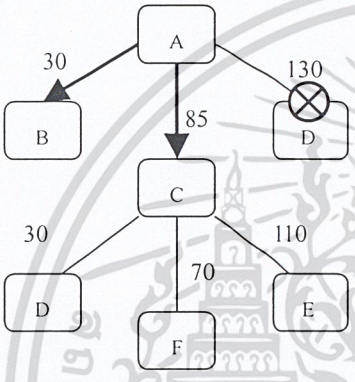
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



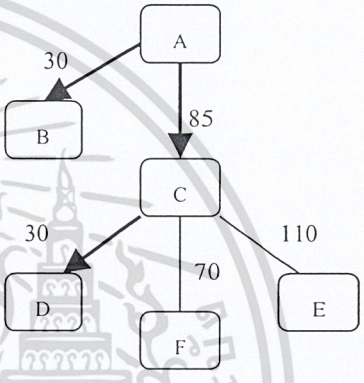
(1) เราเตอร์ A สร้างต้นไม้แสดงเส้นทางไปยังเราเตอร์ข้างเคียง



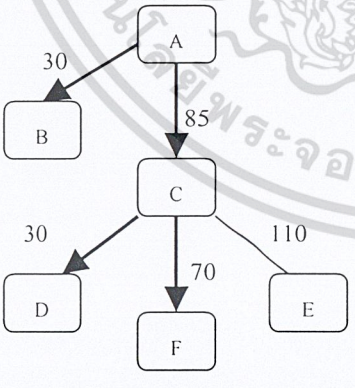
(2) สร้างเส้นทางต่อจากเราเตอร์ B ซึ่งมีเพียงเราเตอร์ D เท่านั้นแต่ระยะทางจาก A ไป D โดยตรงมีค่าเท่ากับ 130 ซึ่งน้อยกว่าการไป D โดยผ่าน B ก่อนซึ่งจะมีค่าเท่ากับ 115+30 = 145 ดังนั้นให้เส้นทางไป B เป็นเส้นทางถาวร



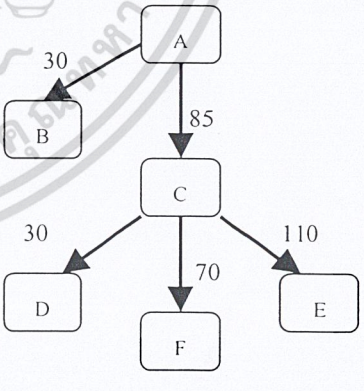
(3) สร้างเส้นทางต่อจากเราเตอร์ C แล้วพบว่า เส้นทางไป D ผ่านทาง C (85+30=115) สั้นกว่าไป D (130) โดยตรง ดังนั้นให้เส้นทางไป C เป็นเส้นทางถาวร และยกเลิกเส้นทางไป D



(4) จากเราเตอร์ D ไม่พบเส้นทางไปเราเตอร์ใดที่สั้นกว่าจึงให้เส้นทางที่ D เป็นเส้นทางถาวร



(5) จากเราเตอร์ F ไม่พบเส้นทางไปเราเตอร์ใดที่สั้นกว่าจึงให้เส้นทางที่ F เป็นเส้นทางถาวร



(6) จากเราเตอร์ E ไม่พบเส้นทางไปเราเตอร์ใดที่สั้นกว่าจึงให้เส้นทางที่ E เป็นเส้นทางถาวร

รูปที่ 6-3 การคำนวณหาเส้นทางสั้นที่สุดตามขั้นตอนวิธีของไดจ์สตรา

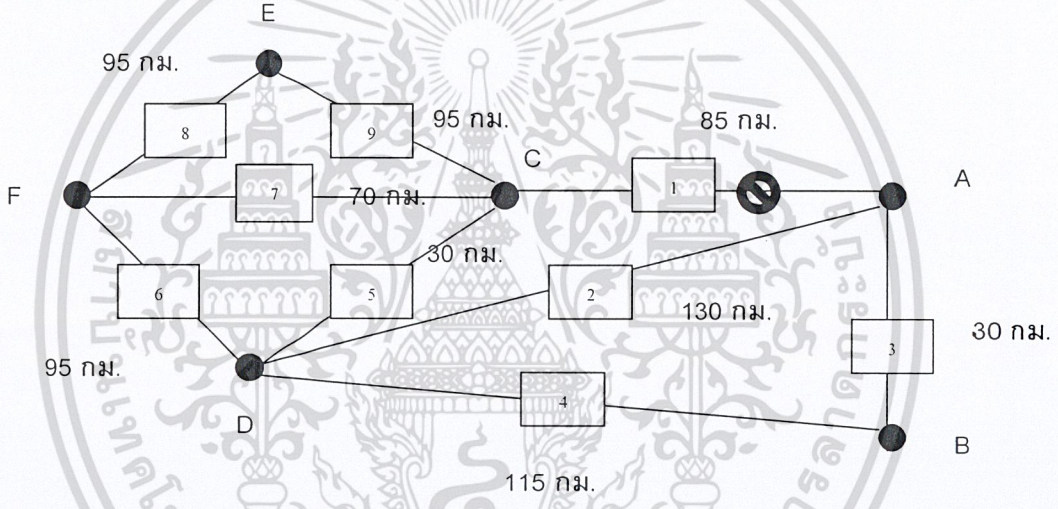
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.4.2 การปรับเปลี่ยนเมื่อเครือข่ายเปลี่ยน

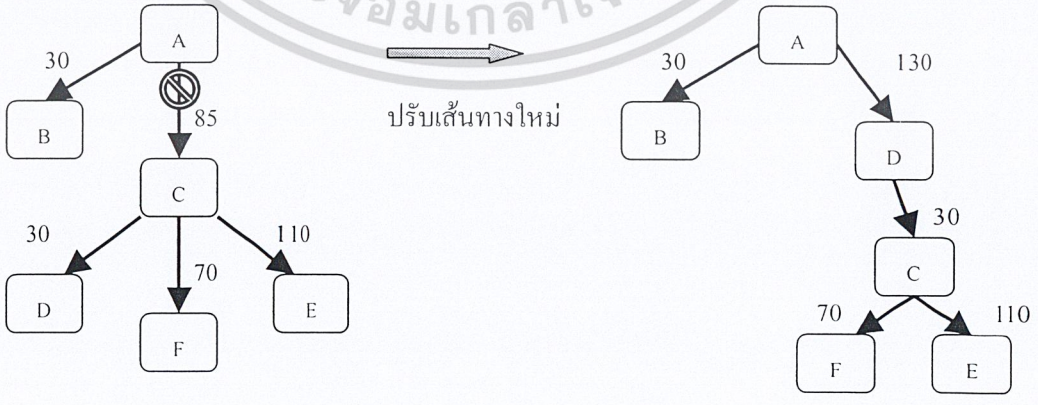
การทำงานของโอเอฟพีเอฟซึ่งประกอบด้วย 3 ขั้นตอนที่กล่าวไปแล้วนั้น ในทางปฏิบัติจะเกิดขึ้นแบบขนานพร้อมกันไปในเราเตอร์แต่ละตัว เราเตอร์จะส่งและได้รับฮัลโหลแพ็กเก็ตกับเราเตอร์ข้างเคียงอยู่ตลอดเวลา หากสายสื่อสารเกิดผิดปกติทำให้เราเตอร์ไม่ได้รับฮัลโหลแพ็กเก็ตภายในระยะเวลาที่กำหนดก็จะทราบว่าเส้นทางไปไม่ถึงและจะเริ่มคำนวณหาเส้นทางใหม่

ตัวอย่างเช่นเส้นทางจากเราเตอร์ A ไปยังเราเตอร์ C มีปัญหาขัดข้องดังรูปที่ 6-4 เราเตอร์ A และ C ต่างไม่ได้รับฮัลโหลแพ็กเก็ตซึ่งกันและกัน เมื่อถึงกำหนดเวลาต่างก็ทราบว่าเส้นทางไม่สามารถใช้งานได้และต้องเปลี่ยนฐานข้อมูลลิงค์เสตทใหม่

หากพิจารณาเฉพาะที่เราเตอร์ A ซึ่งจะคำนวณหาเส้นทางใหม่ กรณีเราเตอร์จะใช้เส้นทางไปยัง D เพื่อไป C โดยเส้นทางอื่นยังคงเดิม ต้นไม้แสดงเส้นทางสั้นที่สุดจะมีลักษณะดังรูปที่ 6-5



รูปที่ 6-4 ปัญหาเส้นทางขัดข้อง



รูปที่ 6-5 เส้นทางใหม่ที่คำนวณได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

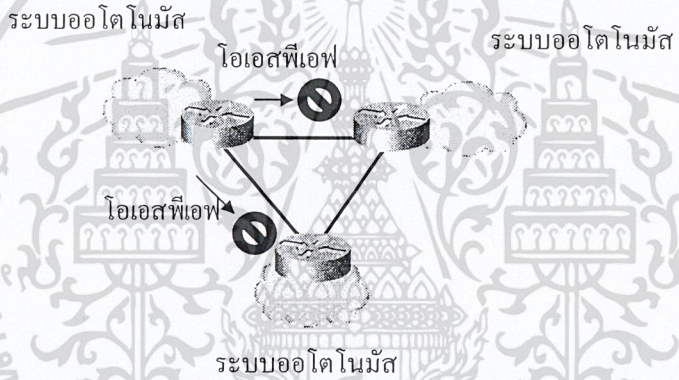
ขั้นตอนวิธีของไดจ์สตราเป็นขั้นตอนวิธีที่ต้องใช้กำลังการคำนวณของซีพียูอย่างมากโดยเฉพาะอย่างยิ่งในเครือข่ายขนาดใหญ่ เราเตอร์จะต้องคำนวณหาต้นไม้ที่แทนเส้นทางสั้นที่สุดใหม่ทุกครั้งที่มีการเปลี่ยนแปลงใดๆในเครือข่าย

แต่โอเอสพีเอฟมีวิธีการจัดรูปแบบเครือข่ายเพื่อลดการคำนวณเมื่อเครือข่ายเปลี่ยนแปลงซึ่งจะกล่าวในหัวข้อถัดไป

6.5 การจัดองค์ประกอบเครือข่าย

ระบบออโตโนมัสเป็นตัวกำหนดขอบเขตการทำงานของโอเอสพีเอฟ ข้อมูลเส้นทางโอเอสพีเอฟจะแลกเปลี่ยนระหว่างเราเตอร์ที่อยู่ภายในออโตโนมัสหนึ่งๆ เท่านั้น

ในระบบออโตโนมัสหนึ่งๆ ย่อมมีเราเตอร์เชื่อมต่อไปยังระบบอื่นดังตัวอย่างในรูปที่ 6-6 เราเตอร์ในระบบออโตโนมัสที่เชื่อมกับระบบออโตโนมัสอื่นเรียกว่า **เราเตอร์ขอบปลายระบบออโตโนมัส** (AS boundary routers) และเป็นตัวกันไม่ให้ข้อมูลโอเอสพีเอฟออกไปสู่ระบบออโตโนมัสอื่น



รูปที่ 6-6 เราเตอร์ขอบปลายของระบบออโตโนมัส

6.6 การแบ่งระบบออโตโนมัสออกเป็นพื้นที่

เครือข่ายที่ใช้งานโอเอสพีเอฟย่อมอยู่ในระบบออโตโนมัสเดียวกัน และอาจเป็นได้ทั้งเครือข่ายขนาดเล็กไปถึงเครือข่ายขนาดใหญ่ โอเอสพีเอฟสามารถทำงานได้กับเครือข่ายขนาดใหญ่และยังรองรับการขยายขนาดขึ้นไปอีก เพราะมีคุณสมบัติการทำงานที่รองรับการขยายอยู่ในตัว

คุณสมบัติหนึ่งได้แก่การแบ่งเครือข่ายออกเป็นกลุ่มย่อยเพื่อจำกัดขอบเขตการทำงานบางอย่างให้อยู่ในบริเวฯใดบริเวณหนึ่งหรือกลุ่มของเราเตอร์กลุ่มหนึ่งๆ ขอบเขตของเครือข่ายที่จัดแบ่งแล้วเรียกว่า **พื้นที่** (Area)

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

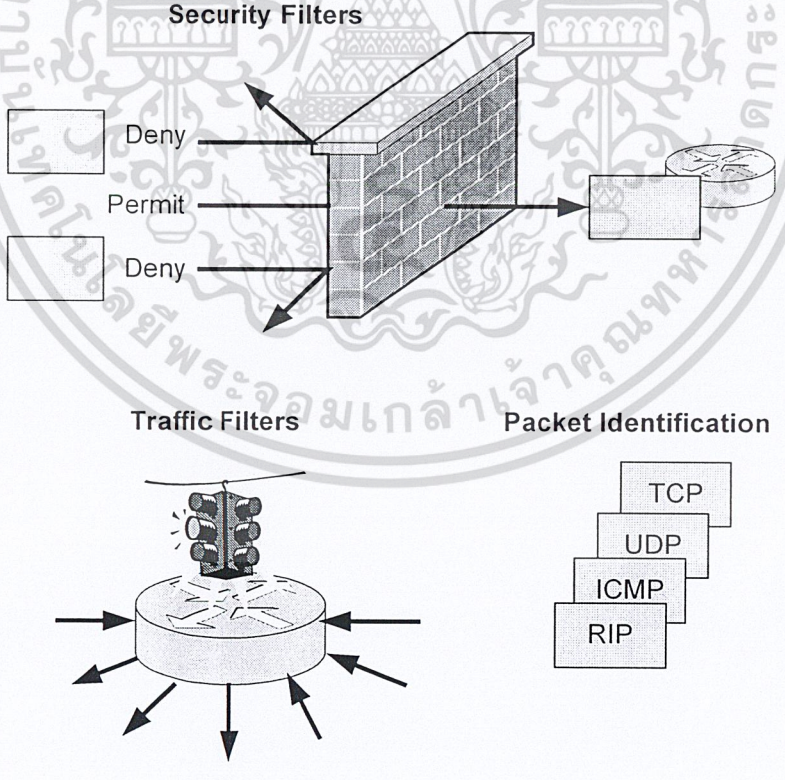
เอกสารนี้มีลิขสิทธิ์ ห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
ไม่อนุญาตให้นำไปใช้เพื่อการค้า

บทที่ 7

แอคเซสลิสต์(Access list)

แอคเซสลิสต์เป็นวิธีการที่รู้จักกันโดยทั่วไปในปัจจุบันซึ่งการทำงานประกอบด้วยการอนุญาตและการป้องกันของแพ็กเก็ตที่เข้าหรือออกจากเราเตอร์ ซึ่งแอคเซสลิสต์กลายเป็นเครื่องมือสำหรับใช้ควบคุมแพ็กเก็ตและภายในเครือข่ายที่มีประสิทธิภาพในปัจจุบัน ซึ่งมีรูปแบบการทำงานหลัก 3 แบบดังนี้ คือ

1. การตรวจสอบความปลอดภัย จะทำการอนุญาตเฉพาะแพ็กเก็ตที่รู้จักและทำการป้องกันแพ็กเก็ตที่เหลือออกไปทั้งหมด
2. การตรวจสอบการจราจรในเครือข่ายจะทำการป้องกันแพ็กเก็ตที่มีความสำคัญน้อยหรือไม่มีความสำคัญออกไป ซึ่งจะทำให้ไม่เสียขนาดแบนด์วิดธ์ในเครือข่ายไป วิธีการนี้คล้ายกับ การตรวจสอบความปลอดภัย แต่จะใช้เทคนิคการใช้งานตรงข้ามกัน โดยจะทำการป้องกันแพ็กเก็ตที่ไม่ต้องการออก และจะอนุญาตแพ็กเก็ตที่เหลือให้อยู่ต่อไป
3. เป็นเครื่องมือที่ใช้ในเราเตอร์ของซิสโก้ ยกตัวอย่างเช่น ไดอัลเลอร์ลิส เราเตอร์ฟิวล์เตอร์ เราแม็บ และ คิวริ่งลิส ซึ่งสามารถบ่งชี้ได้ว่ารูปแบบแพ็กเก็ตเป็นแบบใด



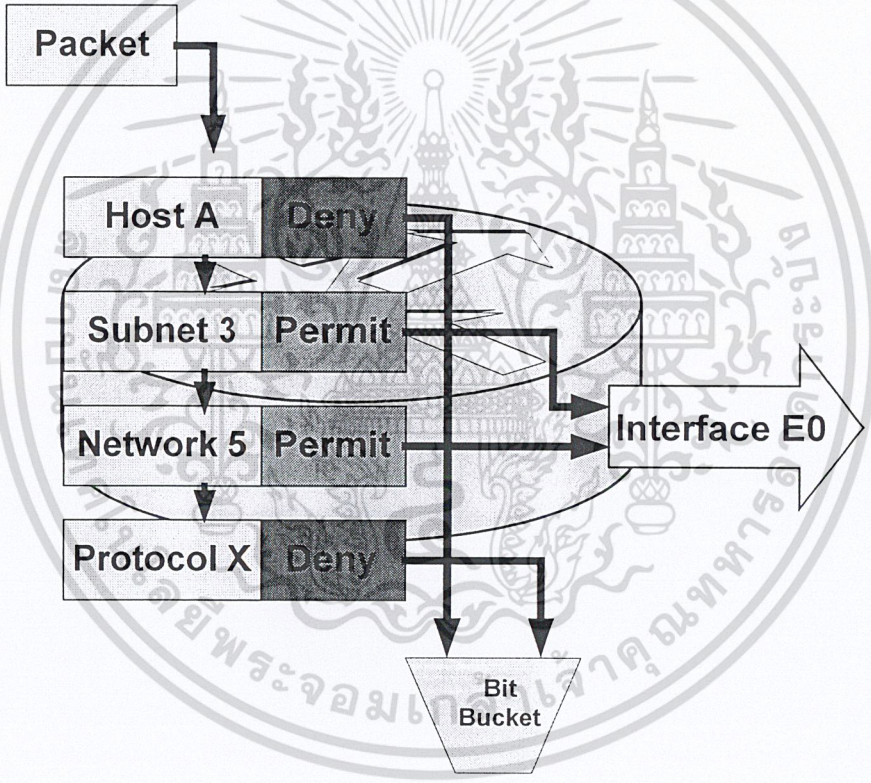
รูปที่ 7-1 การนำแอคเซสลิสต์ไปใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7.1 พื้นฐานของแอคเซสลิสต์

แอคเซสลิสต์เป็นวิธีการตรวจสอบแบบเป็นลำดับและต่อเนื่องโดยการทดสอบจะเปรียบเทียบจาก แพ็กเก็ตข้อมูลที่เข้ามา และจะเลือกว่าจะกระทำกับชุดข้อมูลนั้นอย่างไร ระหว่างการอนุญาตหรือป้องกัน ในส่วนการเปรียบเทียบจะทำได้ตั้งแต่การเปรียบเทียบแอดแดรสต้นทาง หรืออาจมีความซับซ้อนยิ่งขึ้น เช่น การตรวจสอบทั้งแอดแดรสต้นทางและแอดแดรสปลายทาง รูปแบบทางโพรโตคอล หมายเลขพอร์ต เป็นต้น

แพ็กเก็ตจะถูกนำเข้ามาไว้ที่บนสุดของ สแตกฟิวล์เตอร์ ดังรูปที่2 และในแต่ละขั้นตอนการตรวจเช็คถ้าตรวจเช็คแล้วพบว่ามีเงื่อนไขที่ตรงกันก็จะไปทำต่อในส่วนการทำงานที่เลือกไว้โดยอาจเป็นการอนุญาตหรือการป้องกันแพ็กเก็ตนั้น แต่ถ้าการเปรียบเทียบไม่ตรงกับเงื่อนไข แพ็กเก็ตนั้นจะถูกส่งไปเพื่อการเปรียบเทียบในขั้นตอนอื่นต่อ และการเปรียบเทียบก็จะเกิดขึ้นต่อไป



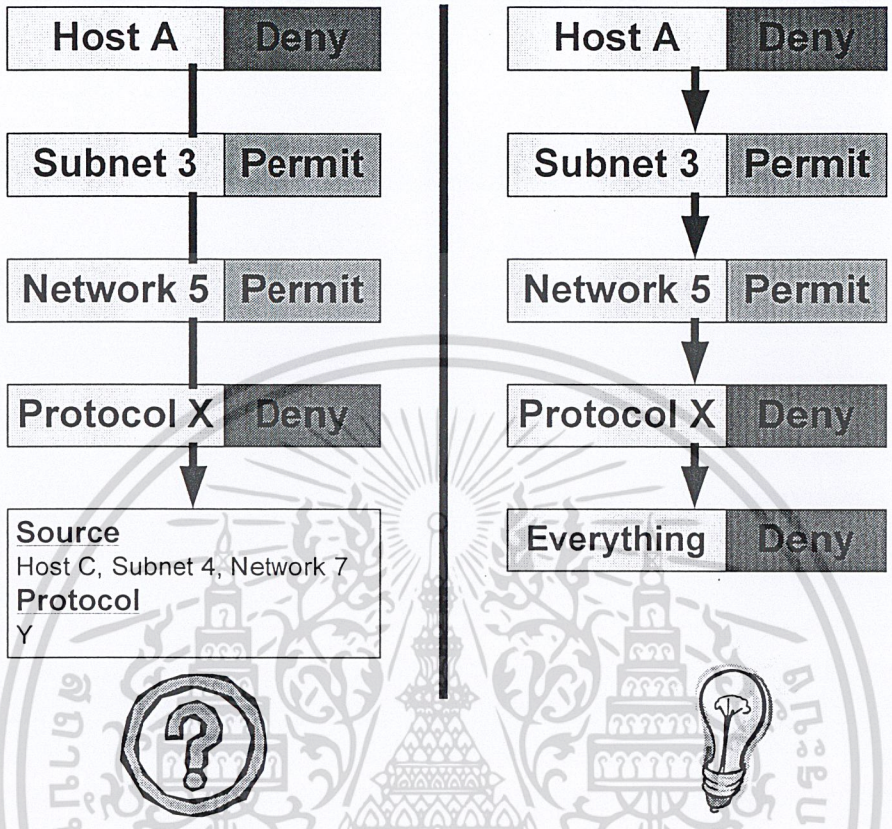
รูปที่ 7-2 ลำดับการทำงานของแอคเซสลิสต์

ในรูปที่ 7-2 คำว่า *Permit* หมายถึงแพ็กเก็ตได้รับอนุญาตให้ออกไปจาก อินเทอร์เฟสอีเทอร์เน็ตหมายเลข 0 (E0) และ *Deny* จะหมายถึงแพ็กเก็ตจะถูกกำจัดออกไป ตามตัวอย่างในรูป สมมุติให้แพ็กเก็ตมีรูปแบบดังนี้

Host D - Subnet 2 - Network 5 แล้วในการเทียบครั้งแรกซึ่งมีเงื่อนไขเป็น Host A จะไม่ตรงซึ่งจะทำให้แพ็กเก็ตถูกส่งไปยังข้างด้านล่างต่อมาเพื่อเปรียบเทียบกับ Subnet 3 ซึ่งจะยังไม่ตรงอีก สุดท้ายเมื่อเปรียบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น ทั้งนี้ห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เทียบกับ Network 5 จะทำให้แพ็กเก็ตนั้นได้รับการส่งออกไปจาก อินเทอร์เฟสอีเทอร์เน็ตหมายเลข 0 ในที่สุด



รูปที่ 7-3 ตัวอย่างการจัดการกับแพ็กเก็ตที่ไม่ตรงในแต่ละลำดับชั้น

ส่วนในกรณีที่การเปรียบเทียบในทุกขั้นตอนแล้วไม่ตรง การเปรียบเทียบใดๆ จะใช้ค่าสำรองที่จะมีอยู่แล้วในที่นี้ใน เราเตอร์ของ ซิสโก้ จะเป็น *Deny Any* หรือหมายความว่าให้ทำการกำจัดแพ็กเก็ตที่เหลือ ออกไป เราสามารถทำการเปลี่ยนค่าสำรองนี้ได้โดยอาจกำหนด ค่าให้เป็น *Permit Any* เพื่อทำการอนุญาตแพ็กเก็ตที่เหลือจากการเปรียบเทียบทั้งหมด

7.2 รูปแบบของ แอคเซสลิสต์

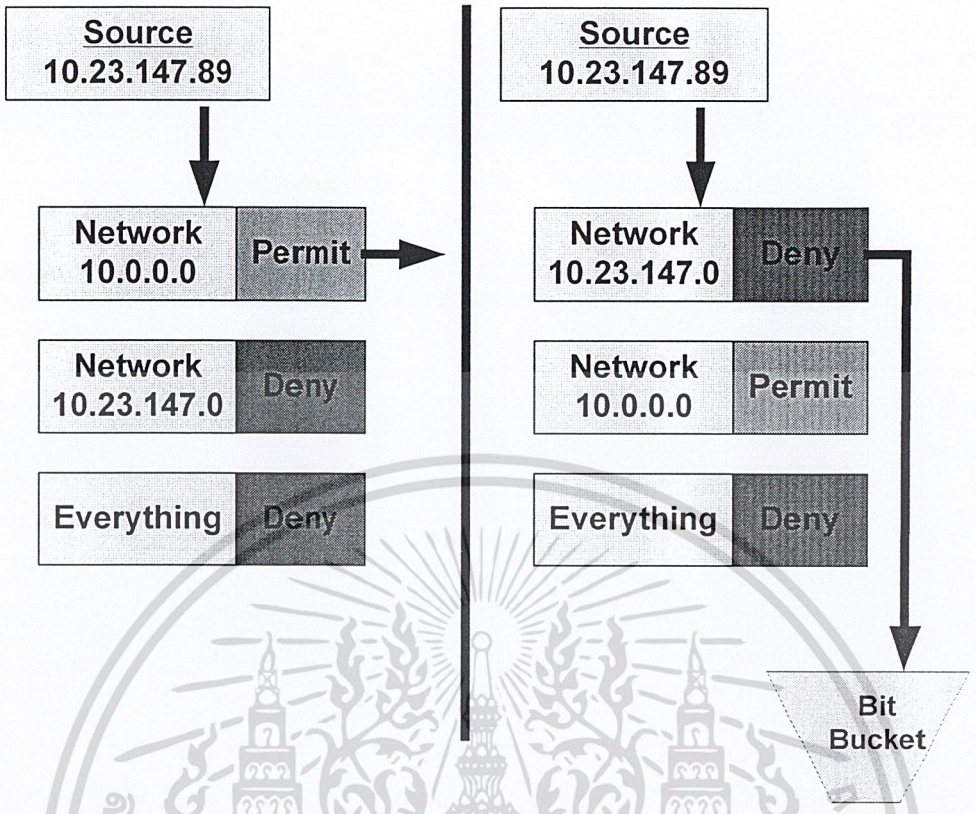
ตัวอย่างการคอนฟิกค่าของแอคเซสลิสต์

access-list 9 deny 10.23.147.0 0.0.0.255

access-list 9 permit 10.0.0.0 0.255.255.255

จากการคอนฟิกค่าของแอคเซสลิสต์แสดงได้ในรูปที่ 7-4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7-4 ตัวอย่างการคอนฟิกลำดับชั้นของแอคเซสลิสต์ที่ถูกต้องและไม่ถูกต้อง

ในทุกๆชั้นของแอคเซสลิสต์จะแทนด้วยหนึ่งคำสั่งการคอนฟิกแต่จะเห็นว่ามีเลข 9 อยู่ทั้งสองบรรทัดในการคอนฟิกตามตัวอย่าง ตัวเลขนี้เรียกว่า หมายเลขของแอคเซสลิสต์ ซึ่งมีจุดประสงค์เพื่อ

- เชื่อมต่อหลายคำสั่งให้อยู่ในแอคเซสลิสต์เดียวกัน และเพื่อความแตกต่างระหว่างแอ็กเซสลิสต์ชุดอื่นที่อาจมีการคอนฟิกไว้ก่อน
- ทำให้เราเตอร์สามารถแยกความแตกต่างระหว่างรูปแบบของแอคเซสลิสต์แบบต่างๆได้ อย่างเช่นในซิสโก้ ไอโอเอส (IOS) มีแอคเซสลิสต์ จะมี IP , IPX , AppleTalk รวมทั้งโพรโตคอลต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Access List Type	Range
Standard IP	1-99
Extended IP	100 – 199
Ethernet type code	200-299
Ethernet address	700-799
Transparent bridging (protocol type)	200-299
Transparent bridging (vendor code)	700-799
Extended Transparent bridging	1100-1199
DECnet and extended DECnet	300-399
XNS	400-499
Extended XNS	500-599
AppleTalk	600-699
Source-route bridging (protocol type)	200-299
Source-route bridging (vendor code)	700-799
Standard IPX	800-899
Extended IPX	900-999
IPX SAP	1000-1099
NLSP route summery	1200-1299
Standard VINES	1-99
Extended VINES	100-199
Simple VINES	200-299

ตารางที่ 7-1 รูปแบบตัวเลขแอคเซสลิสต์ของซิสโก้

7.3 การแก้ไขค่าแอคเซสลิสต์

การใช้งานของแอคเซสลิสต์จากค่าเก่าที่ได้ทำการคอนฟิกไว้ก่อนแล้วสามารถทำได้โดยใช้คำสั่งการลบแอคเซสลิสต์ดังนี้

no access-list 101 permit tcp 10.2.5.4 0.0.0.255

ในที่นี้แอคเซสลิสต์หมายเลขที่ 101 จะถูกลบออกไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อีกวิธีที่เราจะทำการคอนฟิกค่าแอคเซสลิสต์ได้คือ ทำการอัพโหลดไฟล์คอนฟิกที่เราได้เขียนเอาไว้ก่อน โดยใช้ ทีเอ็ฟทีพี เซิร์ฟเวอร์ (TFTP server) โดยรูปแบบในไฟล์ที่เราทำการเขียนไว้ควรจะมี ประโยคที่ทำการลบแอคเซสลิสต์ตัวเดิมออกก่อน ซึ่งมีรูปแบบดังนี้

no access-list #

เมื่อเครื่องหมาย # แทนหมายเลขของแอคเซสลิสต์ที่ต้องการใช้งาน ตัวอย่างมีดังนี้

no access-list 5
access-list 5 permit 10.0.0.1 0.0.0.0
access-list 5 permit 10.0.1.0 0.0.0.255
access-list deny any

ในบรรทัดแรก *no access-list 5* เป็นการลบแอคเซสลิสต์หมายเลข 5 ของเก่าลงเพื่อจะนำค่าการคอนฟิกใหม่ลงแทน เพื่อหลีกเลี่ยงปัญหาที่นำแอคเซสลิสต์ใหม่ไปต่อกับ แอคเซสลิสต์ที่มีอยู่เดิม

7.4 สแตนดาร์ด ไอพี แอคเซสลิสต์ (Standard IP Access Lists)

รูปแบบของการคอนฟิกแบบนี้คือ

access-list access-list-number {deny | permit} source [source-wildcard]

รูปแบบคำสั่งนี้จะใช้กับหมายเลขของแอคเซสลิสต์ตามตาราง คือ 1 ถึง 99 ส่วน *deny* กับ *permit* เป็นการเลือกการปฏิบัติงานเมื่อการเปรียบเทียบถูกต้อง *source* เป็น ไอพีแอดเดรสของต้นทาง และ ส่วน *source-wildcard* เป็นการกำหนดช่วงของ *source* นั้นเอง

ตัวอย่าง

access-list 1 permit 161.246.5.27 0.0.0.0
access-list 1 permit 161.246.5.15 0.0.0.0
access-list 1 deny 161.246.5.0 0.0.0.255
access-list 1 permit 161.246.0.0 0.0.31.255
access-list 1 deny 161.246.0.0 0.0.255.255
access-list permit 0.0.0.0 255.255.255.255

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตัวอย่างสองบรรทัดแรกเป็นการกำหนด ให้หมายเลขโฮสต์ที่กำหนดไว้คือ 161.246.5.27 และ 161.246.5.15 สามารถผ่านได้โดยค่า 0.0.0.0 เป็นค่าไวร์คาร์ด *(wildcard)* ซึ่งจะเรียกอีกอย่างว่า อินเวิร์สแมส *(inverse mask)* ส่วนบรรทัดที่สามเป็นการป้องกันโฮสต์อื่นที่มาจากสับเน็ต 161.246.5.0 บรรทัดที่สี่ บอกว่าให้หมายเลขโฮสต์ตั้งแต่ 161.246.0.1 ถึง 161.246.31.255 ผ่านไปได้ โดยตัว อินเวิร์สแมส เป็นตัวกำหนดช่วงของแอดเดรส บรรทัดที่ห้าเป็นการป้องกันโฮสต์อื่นๆจากสับเน็ต 161.246.0.0 และบรรทัดสุดท้ายเป็นการอนุญาตให้โฮสต์อื่นๆผ่านไปได้

7.5 เอกซ์เทนเด็ด ไอพี แอคเซสลิสต์(Extended IP Access Lists)

รูปแบบของการคอนฟิกแบบนี้คือ

access-list access-list-number { deny | permit } protocol source source-wildcard destination destination-wildcard

รูปแบบคำสั่งนี้จะใช้กับหมายเลขของแอคเซสลิสต์ตามตาราง คือ 100 ถึง 199 ส่วนที่เพิ่มเข้ามาเป็นโพรโตคอล ซึ่งสามารถกำหนดได้ดังนี้ *eigrp ,gre , icmp , igmp , igrp , ip , ipinip , nos ,ospf , tcp , udp* โดยถ้าเรากำหนดค่าโพรโตคอล นี้เป็นไอพี *(ip)* จะสามารถใช้งานได้กับทุกๆโพรโตคอล ที่กล่าวมา ส่วนค่า *destination* และ *destination-wildcard* เป็นการกำหนดค่าปลายทางและช่วงของปลายทางนั้นเอง

ตัวอย่าง

access-list 101 permit ip 172.22.30.6 0.0.0.0 10.0.0.0 0.255.255.255
access-list 101 permit ip 172.22.30.95 0.0.0.0 10.11.12.0 0.0.0.255
access-list 101 deny ip 172.22.30.0 0.0.0.255 192.168.18.27 0.0.0.0
access-list 101 permit ip 172.22.0.0 0.0.31.255 192.168.18.0 0.0.0.255
access-list 101 deny ip 172.22.0.0 0.0.255.255 192.168.18.64 0.0.0.63
access-list 101 permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255

บรรทัดที่ 1 กำหนดว่าเป็นไอพีแพ็กเก็ต *(IP Packet)* ที่มีหมายเลขต้นทางเป็น 172.22.30.6 และหมายเลขปลายทางเป็นเครือข่าย 10.0.0.0 ได้รับการอนุญาต *(permit)*

บรรทัดที่ 2 กำหนดว่าไอพีแพ็กเก็ต ที่หมายเลขต้นทางเป็น 172.22.30.95 และหมายเลขปลายทางมีเครือข่ายที่มีสับเน็ตเป็น 10.11.12.0/24 ได้รับการ ได้รับการอนุญาต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรทัดที่ 3 กำหนดว่าเป็นไอพีแพ็กเก็ต ที่มีหมายเลขต้นทางอยู่ในสับเน็ต 172.22.30.0/24 และ หมายเลขปลายทางมีแอดเดรสเป็น 192.168.18.27 จะถูกกำจัดไป

บรรทัดที่ 4 กำหนดว่าเป็นไอพีแพ็กเก็ตที่มีหมายเลขต้นทางระหว่าง 172.22.0.0 และ 172.22.31.255 และมีหมายเลขปลายทางอยู่ในเครือข่าย 192.168.18.0 จะได้รับการอนุญาต

บรรทัดที่ 5 กำหนดว่าเป็นไอพีแพ็กเก็ตที่มีหมายเลขต้นทางอยู่ในเครือข่าย 172.22.0.0 และมี หมายเลขปลายทางเป็น 26 บิตแรกของ 192.168.18.64 จะถูกกำจัดออกไป

บรรทัดที่ 6 กำหนดว่าเป็นไอพีแพ็กเก็ต ที่เหลือจะได้รับการอนุญาต

7.6 ทีซีพี แอคเซสลิสต์ (TCP Access List)

รูปแบบของการคอนฟิกแบบนี้คือ

access-list access-list-number { deny | permit } tcp source source-wildcard
[operator [port]] destination destination-wildcard [operator [port]]

ในที่นี้จะเป็นเอ็กซ์เทนเด็ดแอคเซสลิสต์*(Extended access list)*ประเภทที่มีโพรโตคอลเป็นทีซีพีซึ่งสามารถทำการตรวจสอบ แพ็กเก็ตที่เป็นทีซีพีและสามารถตรวจสอบหมายเลขพอร์ต ได้

- Operator เป็นการกำหนดการเปรียบเทียบสำหรับหมายเลขพอร์ต ได้แก่ eq , neq , gt , lt ซึ่งความหมายคือ เท่ากับ ไม่เท่ากับ มากกว่า และ น้อยกว่าตามลำดับ
- Port เป็นหมายเลขพอร์ตที่กำหนด ยกตัวอย่างเช่น เทลเน็ต (23) , เอฟทีพี (20 และ 21) , เอสเอ็นทีพี (25) เป็นต้น

ตัวอย่าง

access-list 110 permit tcp 10.0.0.0 0.255.255.255 eq 80 172.22.144.0 0.0.0.255 eq 80

จากตัวอย่าง เป็นการอนุญาต ซึ่งกำหนดโพรโตคอลเป็นทีซีพี โดยมีแอดเดรสต้นทางเป็น 10.0.0.0 ที่เป็นการติดต่อผ่านพอร์ต 80 ไปยังปลายทาง ที่แอดเดรสปลายทาง 172.22.144.0.0/24

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7.7 ยูดีพี แอคเซสลิสต์ (UDP Access List)

รูปแบบของการคอนฟิกแบบนี้คือ

access-list access-list-number { deny | permit } udp source source-wildcard [operator [port]] destination destination-wildcard [operator [port]]

ในที่นี้จะเป็น เอ็กซ์เทนเด็ดแอคเซสลิสต์ *(Extended access list)* ประเภทที่มีโพรโตคอล เป็น ยูดีพี โดยโพรโตคอลนี้จะมีลักษณะการเรียกใช้งานเหมือนกับโพรโตคอลทีซีพีแต่ลักษณะที่ต่างกันคือโพรโตคอล ยูดีพี จะใช้การทำงานแบบคอนเน็กชั่นเลส *(Connection less)* ในการติดต่อระหว่างเครื่อง ซึ่งหมายความว่าไม่มีการสร้างเส้นทางการเชื่อมต่อไว้ก่อน

ตัวอย่าง

access-list 109 permit udp 10.0.0.0 0.0.0.255 11.0.0.0 0.0.0.255 eq 161

จากตัวอย่าง เป็นการอนุญาต ซึ่งกำหนดโพรโตคอลเป็นยูดีพี โดยมีแอดเดรสต้นทางเป็น 10.0.0.0 ที่เป็นการติดต่อผ่านไปยังปลายทาง ที่แอดเดรสปลายทาง 172.22.144.0.0/24 ผ่านพอร์ต 161 หรือเป็นเอสเอ็นทีพี แพ็คเก็ต

7.8 ไอซีเอ็มพี แอคเซสลิสต์ (ICMP Access List)

รูปแบบของการคอนฟิกแบบนี้คือ

access-list access-list-number { deny | permit } icmp source source-wildcard destination destination-wildcard [icmp-type [icmp-code]]

ในที่นี้จะเป็นเอ็กซ์เทนเด็ดแอคเซสลิสต์ *(Extended access list)* ประเภทที่มีโพรโตคอล เป็นไอซีเอ็มพี โดยโพรโตคอลนี้จะไม่มีการกำหนดพอร์ตในการเชื่อมต่อของแอดเดรสต้นทางและปลายทางโดย โพรโตคอลแบบไอซีเอ็มพี จะเป็นโพรโตคอลที่อยู่ในชั้นเครือข่าย *(network layer)* ซึ่งสามารถทำการตรวจสอบข้อความของไอซีเอ็มพีซึ่งประกอบด้วย

- icmp-type เป็นตัวเลขระหว่าง 0 – 255 โดยสามารถดูรายละเอียดได้ใน อาร์เอ็ฟซี(RFC) 1700
- icmp-code เป็นการกำหนดสับเซ็ต ของรูปแบบไอซีเอ็มพีแพ็กเก็ต ซึ่งมีค่าระหว่าง 0 – 255

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่าง

access-list 111 deny icmp 172.22.0.0 0.0.255.255 0.0.0.0 255.255.255.255 3 9

จากตัวอย่างเป็นการป้องกันแพ็กเก็ตไอซีเอ็มพีที่มีเครือข่ายแอดเดรสต้นทาง เป็น 172.22.0.0 ไปยังปลายทางทุกตัวด้วยซึ่งเป็น ICMP destination unreachable packet (type 3) และมีโค้ดเป็นเลข 9 ซึ่งมีความหมายว่า Network Administratively Prohibited

7.9 การเรียกใช้งาน แอคเซ็สลิสต์

ในการกำหนดค่าแอคเซ็สลิสต์ต้องกำหนดให้แต่ละอินเทอร์เฟสด้วยคำสั่ง

ip access-group access-list-number { in | out }

ซึ่งคำสั่งนี้เป็นการกำหนดค่าความปลอดภัยหรือตรวจสอบให้กับอินเทอร์เฟสที่เรียกใช้คำสั่งนี้โดยการกำหนดหมายเลขของแอคเซสลิสต์ จะเป็นการเลือกหมายเลขของแอคเซสลิสต์ที่ทำการคอนฟิกกูเลชั่นไว้ โดยแต่ละอินเทอร์เฟสจะกำหนดการตรวจสอบว่าจะตรวจสอบในขณะที่แพ็กเก็ตเข้ามาหรือออกไปโดยการกำหนด *in* หรือ *out*

ตัวอย่าง การใช้งาน

Router(config-int)# ip access-group 15 in

หมายถึงการกำหนดแอคเซสลิสต์กลุ่มที่มีหมายเลขเป็น 15 ให้กับอินเทอร์เฟสโดยจะตรวจสอบทางเข้าของอินเทอร์เฟส

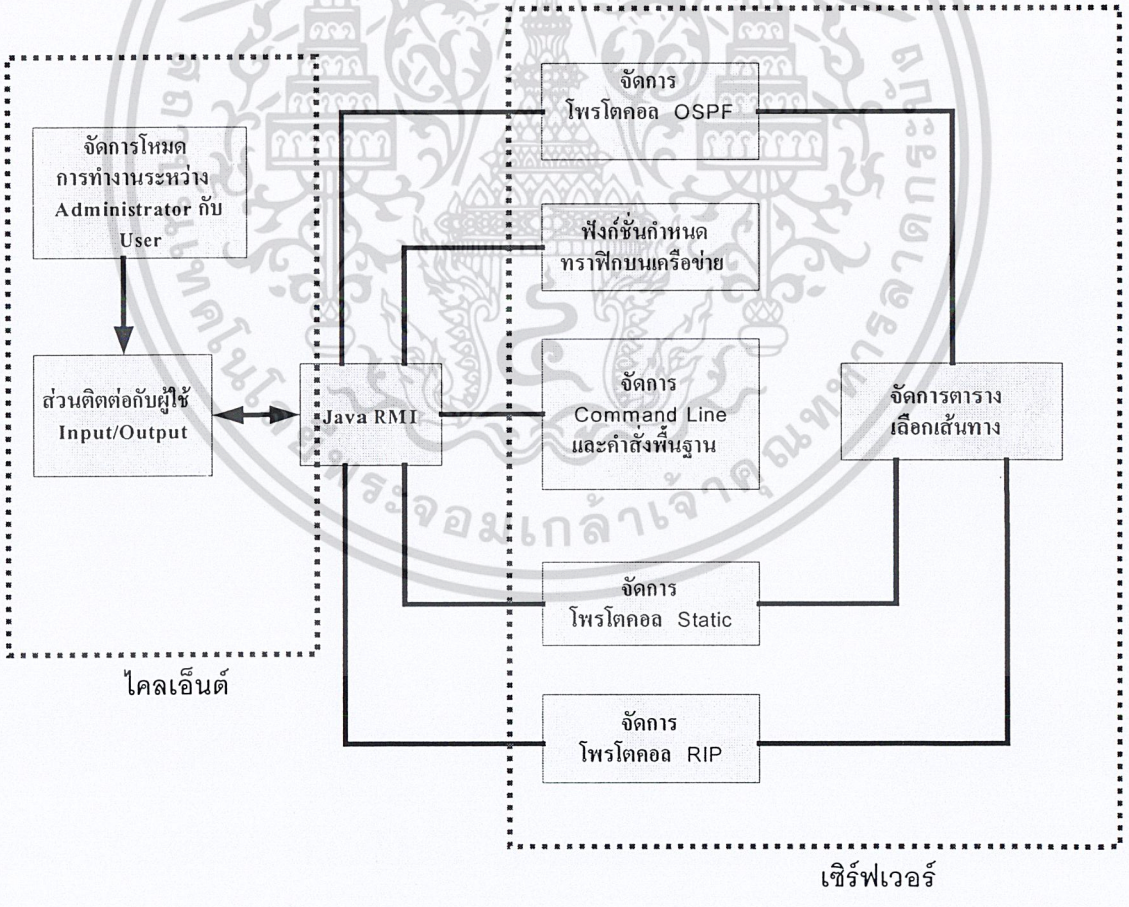
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8
หลักการสร้างและออกแบบโปรแกรม

8.1 โครงสร้างของโปรแกรม

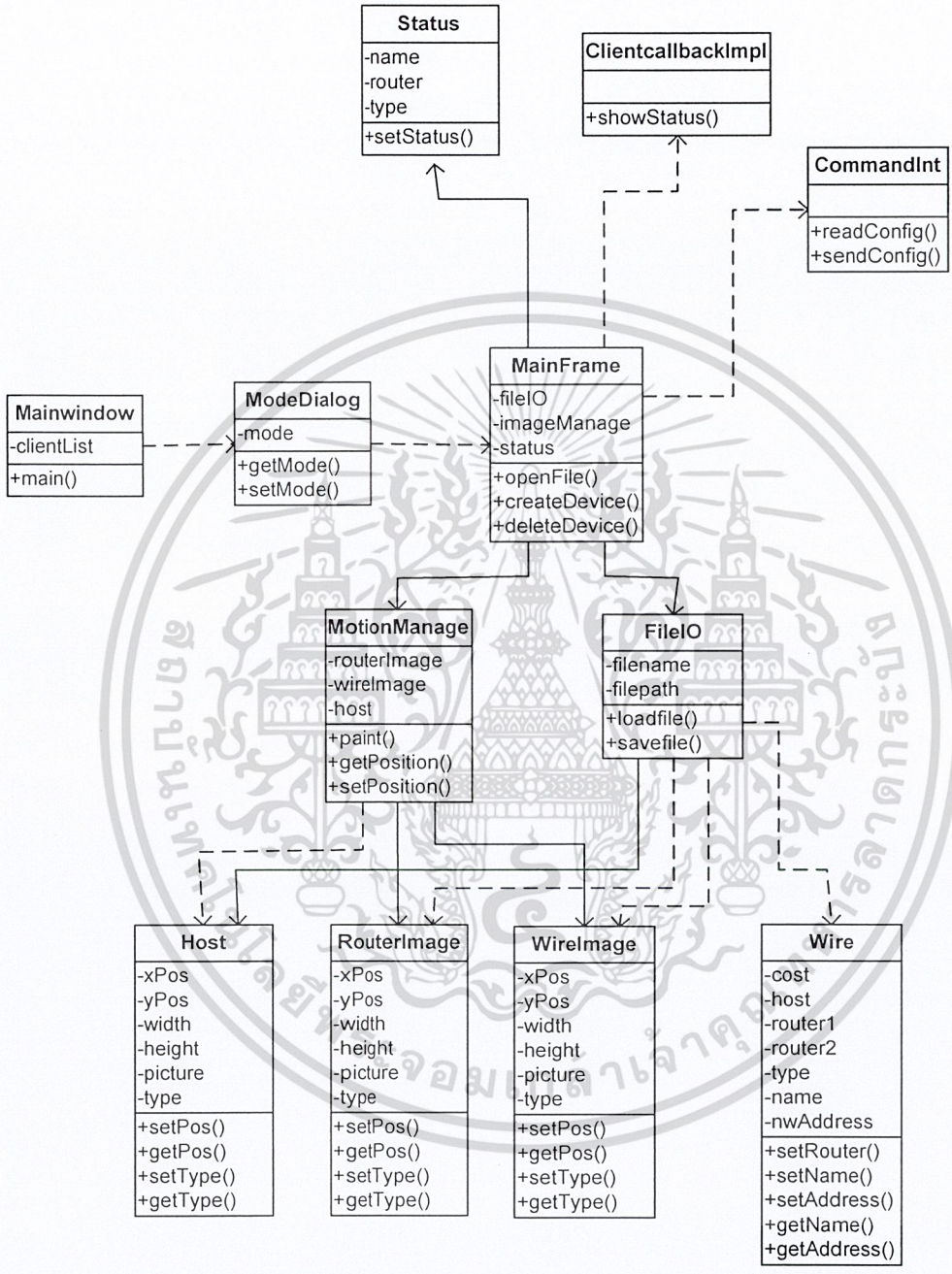
โปรแกรมนี้ออกแบบโดยใช้วิธี ออบเจ็กต์โอเรียนเต็ดโปรแกรมมิง (Object Oriented Programming)เพื่อให้การเขียนโปรแกรมเป็นไปได้ง่ายต่อการเพิ่มและเปลี่ยนแปลงโปรแกรม โดยโครงสร้างของโปรแกรมแบ่งออกเป็น 3 ส่วนหลักๆ ดังนี้

1. ส่วนของ ไคลเอ็นต์ เป็นส่วนที่ใช้ติดต่อกับผู้ใช้เป็นทั้ง Input และ Output รวมถึงเป็นส่วนจัดการโหมดการทำงานระหว่าง Administrator และ User
2. ส่วนของ Java RMI เป็นส่วนที่ใช้ติดต่อระหว่าง ไคลเอ็นต์ กับ เซิร์ฟเวอร์ ซึ่งพัฒนาโดย Java RMI
3. ส่วนของ เซิร์ฟเวอร์ เป็นส่วนทำงานหลักของโปรแกรม



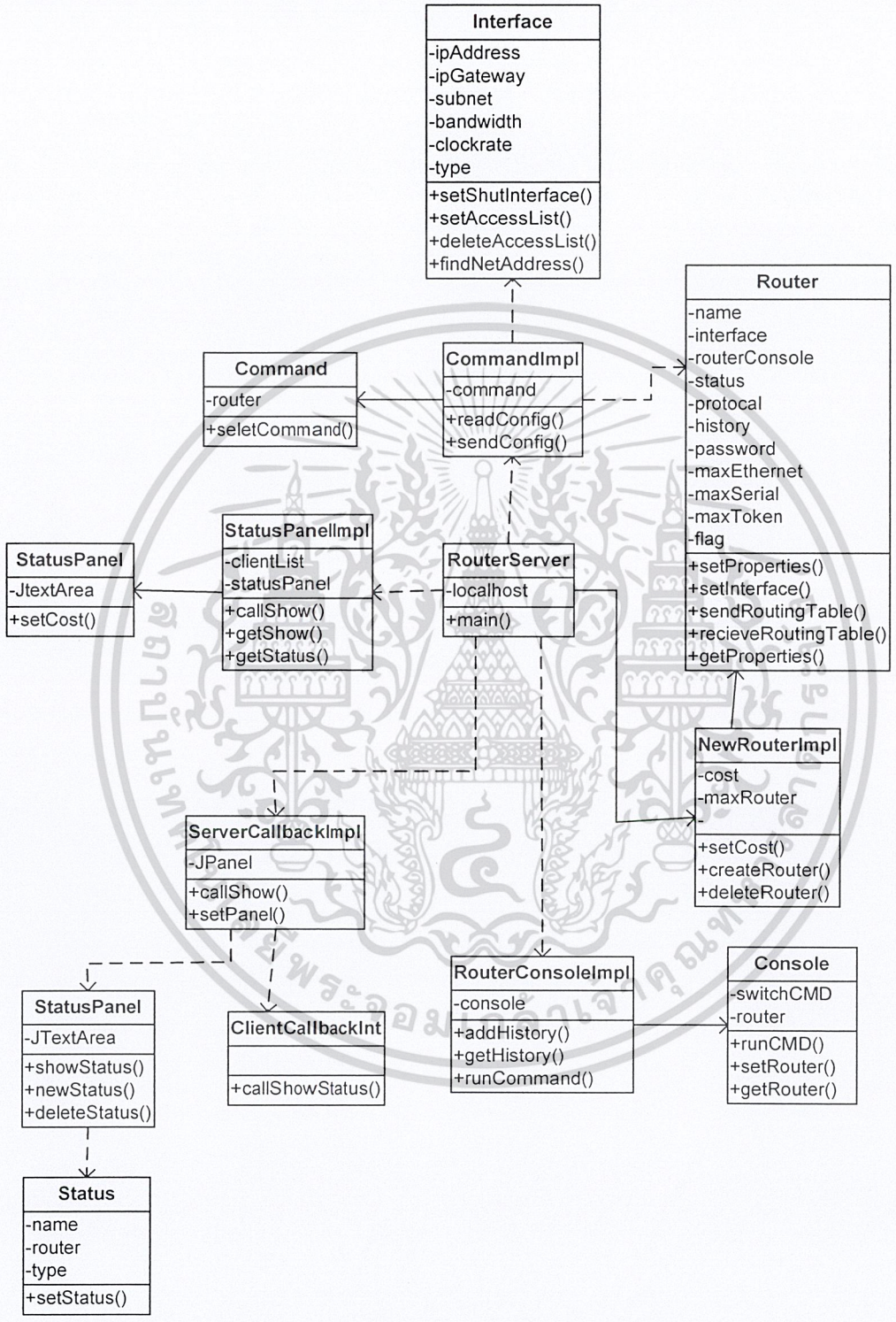
รูปที่ 8-1การออกแบบโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8-2 คลาสไดอะแกรมของโปรแกรมส่วนไคลเอ็นต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8-3 คลาสไดอะแกรมของโปรแกรมส่วนเซิร์ฟเวอร์

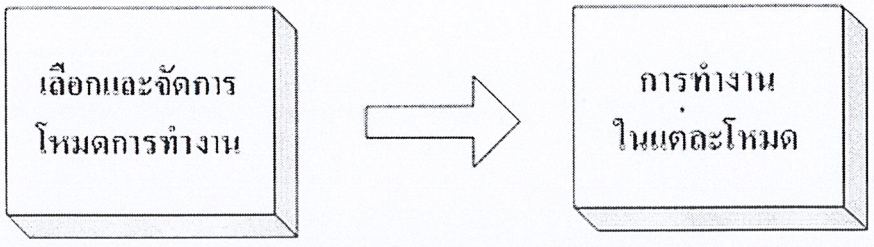
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **Frame3** เป็นคลาสสำหรับสร้างหน้าอินเทอร์เฟสเพื่อติดต่อกับผู้ใช้ โดยหน้าต่างอินเทอร์เฟสจะประกอบไปด้วยเมนูบาร์ ทูลบาร์ เน็ตเวิร์กไดอะแกรมสำหรับสร้างเครือข่าย และส่วนที่เป็นหน้าต่างทำงานของแต่ละเราเตอร์
- **ImageMotion** เป็นคลาสที่จัดการเกี่ยวกับการวาดไดอะแกรมของเครือข่ายที่ผู้ใช้งานสร้างขึ้น
- **FrameR** เป็นคลาสที่เก็บหน้าต่างสำหรับทำงานของเราเตอร์แต่ละตัว
- **RouterConsole** เป็นคลาสที่เก็บหน้าต่างสำหรับทำงานของเราเตอร์แต่ละตัวซึ่งผู้ใช้สามารถป้อนคำสั่งและแสดงผลของคำสั่งที่ป้อนเข้าไป
- **Router** เป็นคลาสที่ประกอบไปด้วยข้อมูลต่างๆ ของเราเตอร์เช่น ชื่อเราเตอร์ สถานะการทำงานของเราเตอร์ โหมดการทำงานของเราเตอร์ อินเทอร์เฟสของเราเตอร์ และตารางการค้นหาเส้นทางของเราเตอร์ โดยฟังก์ชันการทำงานของคลาสนี้จะมีฟังก์ชันสำหรับส่งตารางค้นหาเส้นทาง และรับตารางค้นหาเส้นทางจากเราเตอร์ตัวอื่น
- **RouterImg** เป็นคลาสที่เก็บข้อมูลเกี่ยวกับภาพของเราเตอร์แต่ละตัว
- **Wire** เป็นคลาสที่ประกอบไปด้วยข้อมูลต่างๆ ของสายเชื่อมต่อเช่น ชื่อสายเชื่อมต่อ แบนด์วิดธ์ของสายเชื่อมต่อ ค่าคอสท์ของสายเชื่อมต่อ
- **WireImg** เป็นคลาสที่เก็บข้อมูลเกี่ยวกับภาพของสายเชื่อมต่อแต่ละตัว
- **StatusPanel** เป็นคลาสที่จัดการเกี่ยวกับการแสดงค่าสถานะของเราเตอร์แต่ละตัวเช่น ชื่อเราเตอร์ รุ่นของเราเตอร์ อินเทอร์เฟส ไอพีแอดเดรส และ สถานะของแต่ละอินเตอร์เฟส
- **Traffic** เป็นคลาสที่จัดการเกี่ยวกับการกำหนดลักษณะของทราฟฟิกของเครือข่ายที่สร้างขึ้นมีด้วยกัน 3 แบบได้แก่ แบบค่าคงที่(constants) แบบสุ่มค่า(random) และแบบพัลส์(pulse)
- **Interface** เป็นคลาสที่ประกอบไปด้วยข้อมูลของอินเทอร์เฟสแต่ละตัว เช่น ชื่ออินเทอร์เฟส ไอพีแอดเดรส ประเภทของอินเทอร์เฟส ซับเน็ต และสถานะของอินเทอร์เฟส
- **RoutingTable** เป็นคลาสที่เก็บข้อมูลของตารางค้นหาเส้นทางเช่น เครือข่ายปลายทาง เกตเวย์ที่ออก ค่าเมตริก หรือประเภทของข้อมูลค้นหาเส้นทางว่าเป็นแบบสแตติก อาร์ไอพีหรือแบบอื่น ฟังก์ชันการทำงานหลักก็จะเป็นการเพิ่มข้อมูลลงในตารางค้นหาเส้นทาง
- **SwitchCMD** มีหน้าที่รับคำสั่งจากผู้ใช้มาตรวจสอบว่าเป็นคำสั่งที่ถูกต้องก่อนจะไปเลือกการทำงานของแต่ละคำสั่ง ซึ่งจะมีฟังก์ชันหลักเป็นตัวตรวจสอบว่าเป็นคำสั่งที่ถูกต้อง และส่งไปทำงานตามแต่ละคำสั่ง แต่จะมีเป็นฟังก์ชันย่อยที่ช่วยฟังก์ชันใหญ่ในการตรวจสอบ เช่นฟังก์ชันสำหรับหาช่องว่าง
- **Command** เป็นคลาสที่จัดการเกี่ยวกับคำสั่งคำสั่งของเราเตอร์
- **Rip** เป็นคลาสที่จัดการเกี่ยวกับ การเลือกเส้นทางแบบ RIP โดยคลาสนี้มีการทำงานเป็น Threads รับส่งตารางเลือกเส้นทางเป็นช่วงเวลาที่ตั้งไว้ตามข้อกำหนดของโพรโตคอล
- **Ospf** เป็นคลาสที่จัดการเกี่ยวกับ การเลือกเส้นทางแบบ OSPF มีการทำงานเป็น Threads

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.2 การจัดการส่วนของ ไคลเอ็นต์

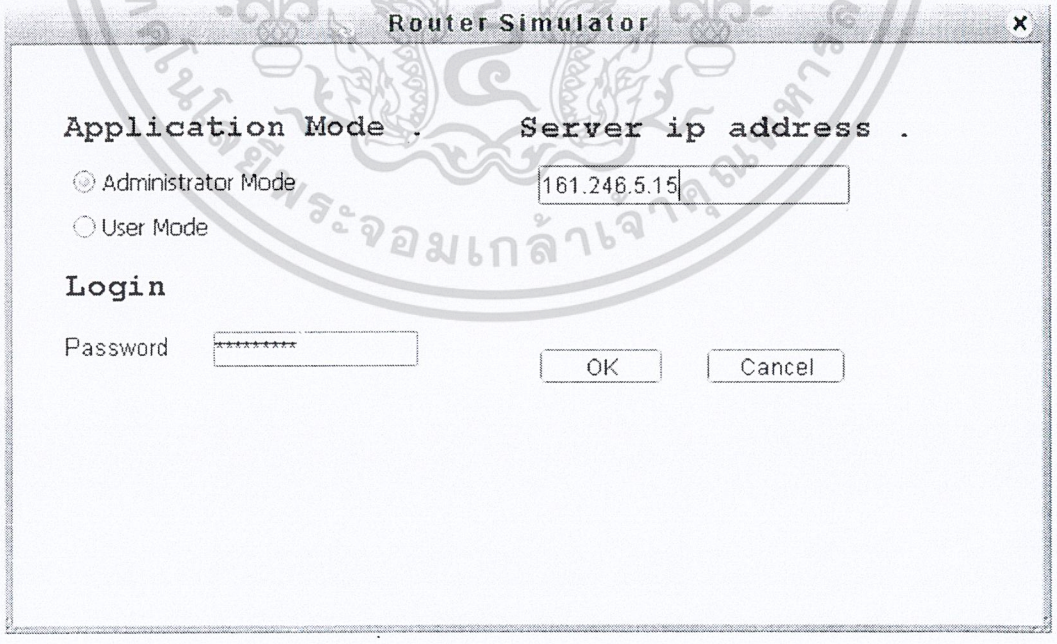
ส่วนของไคลเอ็นต์เป็นส่วนติดต่อกับผู้ใช้ (Graphic User Interface : GUI)ส่วนของ GUI จะมี 2 ส่วนหลักๆ ซึ่งมีลำดับการทำงานดังนี้



รูปที่ 8-4 แสดงลำดับการทำงานของส่วนไคลเอ็นต์

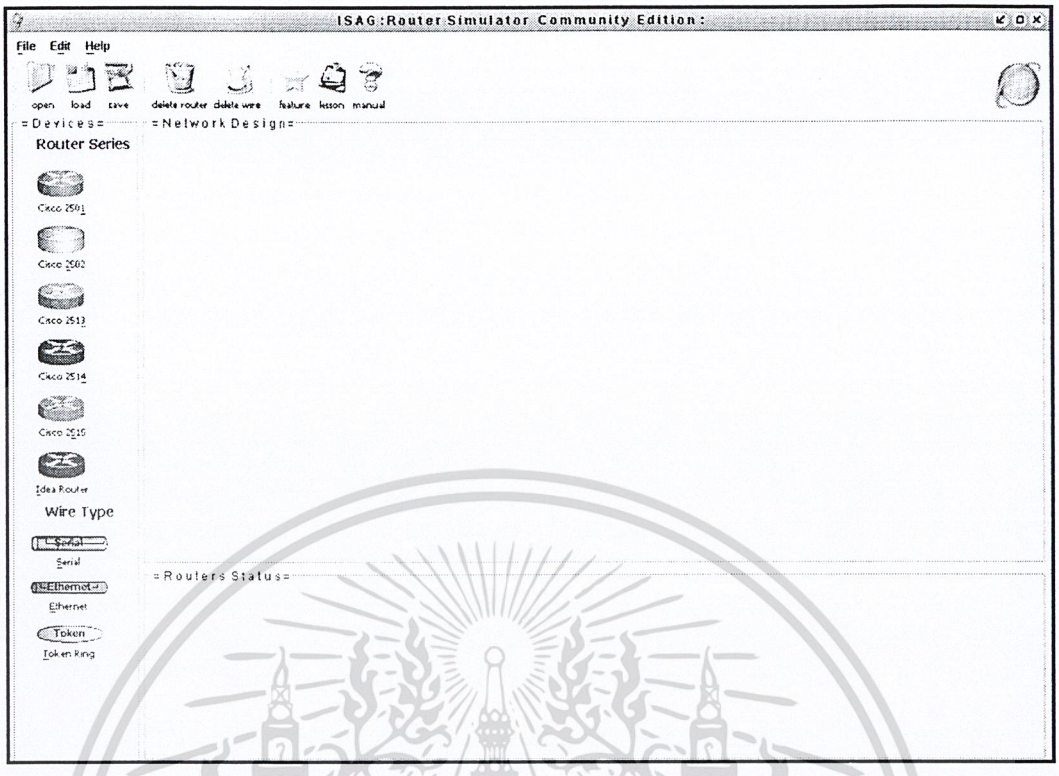
8.2.1 ส่วนของการเลือกโหมดการทำงาน

ส่วนของการเลือกโหมดการทำงานเป็นส่วนที่มีไว้สำหรับให้ผู้ใช้งานเลือกโหมดและจัดการ สิทธิการทำงานของแต่ละโหมดของโปรแกรม และกำหนดไอพีแอดเดรสของเครื่องที่ทำงานเป็นเซิร์ฟเวอร์ โหมดการทำงานมีอยู่ 2 โหมด ได้แก่ โหมดการทำงานของผู้ดูแลระบบ(Administrator mode) และโหมดการทำงานของผู้ใช้งาน(User mode) แต่ละโหมดจะทำงานต่างกันคือ โหมดของผู้ดูแลระบบนั้นสามารถทำการสร้างเปลี่ยนแปลงและแก้ไขเครือข่ายที่สร้างขึ้นได้ นอกจากนั้นยังสามารถตั้งค่าต่างของเราเตอร์ได้ทุกอย่าง ส่วนโหมดการทำงานของผู้ใช้งานนั้น สามารถตั้งค่าของเราเตอร์ที่ผู้ดูแลระบบอนุญาตเท่านั้น

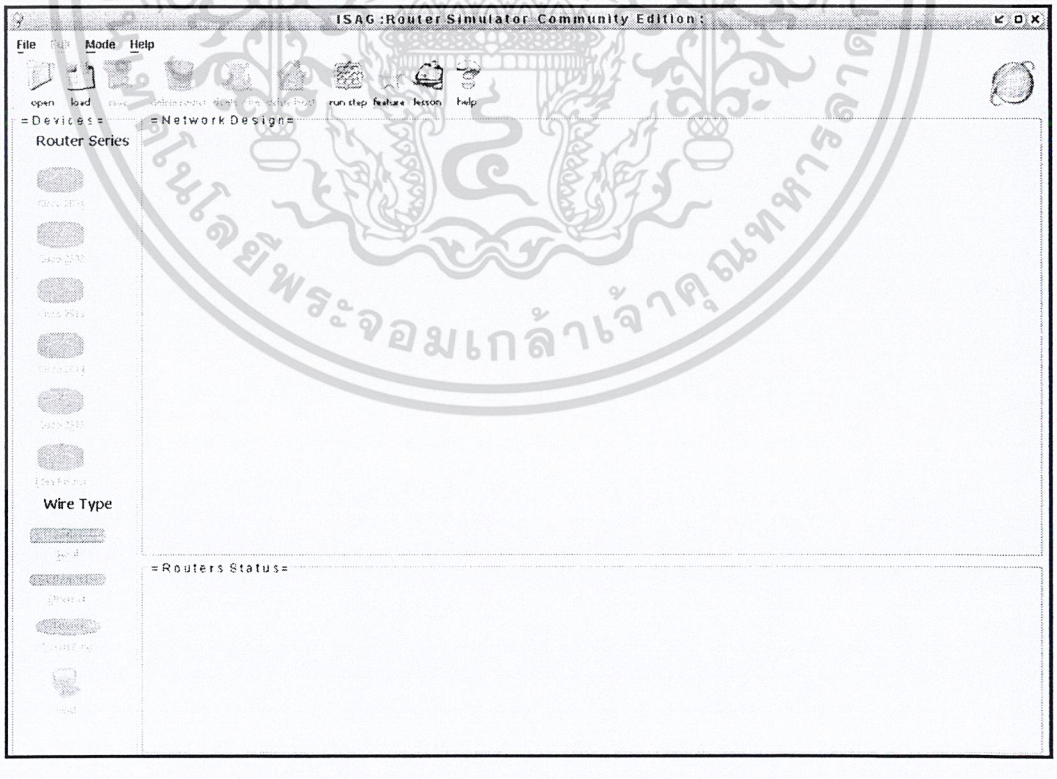


รูปที่ 8-5 หน้าจอสำหรับเลือกโหมดการทำงานของเราเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8-6 ตัวอย่างหน้าจอโหมดการทำงานของผู้ดูแลระบบ



รูปที่ 8-7 ตัวอย่างหน้าจอโหมดการทำงานของผู้ใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฟังก์ชันการทำงานสำหรับส่วนนี้จะอยู่ในคลาส DialogMode มีฟังก์ชันการทำงานดังนี้

- DialogMode() เป็นคอนสตรัคเตอร์สำหรับแสดงหน้าจอไดอะล็อกนี้ ประกอบไปด้วย RadioButton ที่จัดเป็นกลุ่มเดียวกันเพื่อให้ผู้ใช้เลือกตัวเลือกได้เพียงตัวเลือกเดียวเท่านั้น TextArea สำหรับป้อนไอพีของเครื่องที่ทำงานเป็นเซิร์ฟเวอร์ และปุ่ม OK กับ Cancel สำหรับตกลงหรือยกเลิกการเลือกตัวเลือก
- getSelectChoice() เป็นฟังก์ชันสำหรับตรวจสอบว่าผู้ใช้เลือกตัวเลือกใด แล้วให้ค่าของตัวเลือกที่ผู้ใช้เลือกออกมาเป็นค่าตัวเลข
- jButton1_actionPerformed() เป็นฟังก์ชันสำหรับจัดการการกระทำของผู้ใช้กับปุ่ม OK
- jButton2_actionPerformed() เป็นฟังก์ชันสำหรับจัดการการกระทำของผู้ใช้กับปุ่ม Cancel

นอกจากนั้นยังมีส่วนที่ใช้ในการป้อนรหัสผ่านเพื่อให้ยืนยันว่าผู้ใช้งานนั้นเป็นตัวจริง โดยรหัสผ่านจะถูกเก็บไว้ที่เซิร์ฟเวอร์ ในการส่งรหัสผ่านไปเก็บไว้ที่ส่วนของเซิร์ฟเวอร์นั้น ได้มีการเข้ารหัสของรหัสผ่านอีกที โดยใช้วิธีการของ DES (Data Encryption Standard) และใช้วิธีการแลกเปลี่ยนคีย์ระหว่างไคล์เอ็นต์และเซิร์ฟเวอร์แบบ Deffie-Hellman โดยมีคลาสที่เกี่ยวข้องในส่วนนี้ดังนี้
ฟังก์ชันการทำงานสำหรับการเข้ารหัสข้อมูลแบบ DES จะอยู่ในคลาส DesEncrypter มีฟังก์ชันการทำงานดังนี้

- DesEncrypter(SecretKey key) เป็นคอนสตรัคเตอร์ที่รับคีย์ลับเข้ามาเพื่อใช้ในการเข้ารหัสและถอดรหัสข้อมูล
- Encrypt(String cleartext) เป็นฟังก์ชันที่ใช้ในการเข้ารหัสของข้อมูล โดยจะนำข้อมูลที่ส่งมาหรือ cleartext และคีย์ลับทำการเข้ารหัสได้ผลลัพธ์เป็นไซเฟอร์เท็กซ์(Cipher Text) ซึ่งเป็นข้อมูลที่ไม่สามารถอ่านหรือตีความได้
- Decrypt(String ciphertext) เป็นฟังก์ชันที่ใช้ในการถอดรหัส จะใช้คีย์ลับตัวเดียวกันกับที่ใช้ในการเข้ารหัส โดยจะทำกระบวนการย้อนกลับกับกระบวนการการเข้ารหัส คือนำเอาไซเฟอร์เท็กซ์ที่ได้จากการเข้ารหัสกับคีย์ลับมาทำการถอดรหัสจะได้เป็น cleartext ออกมา

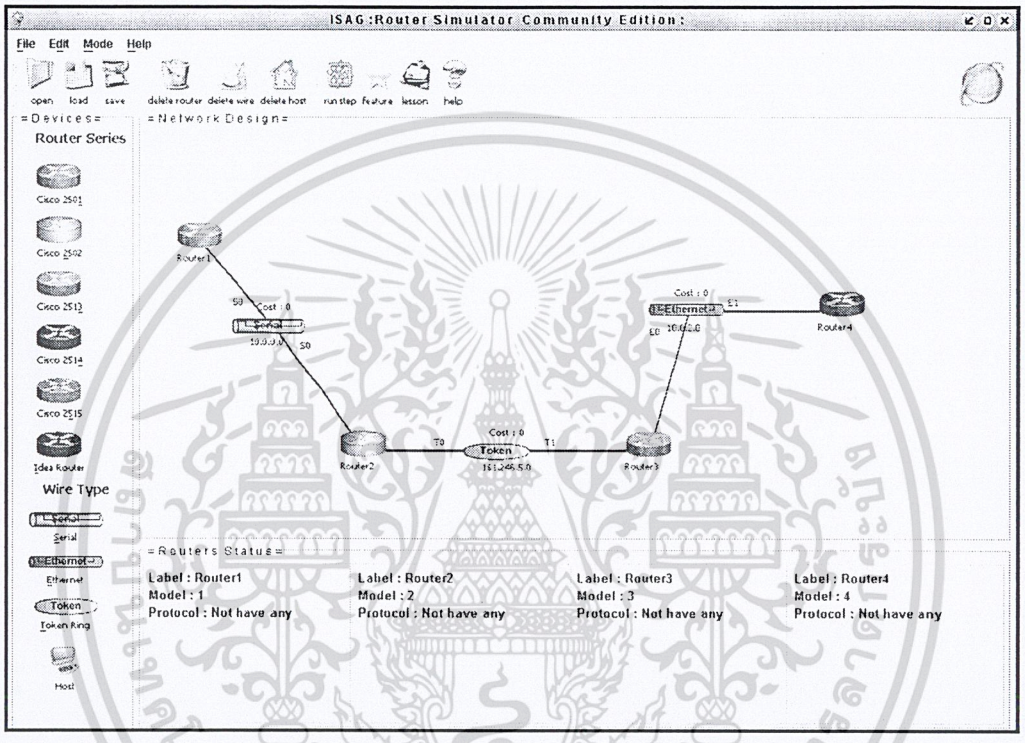
ฟังก์ชันการทำงานสำหรับการแลกเปลี่ยนคีย์ จะอยู่ในคลาส KeyManager มีฟังก์ชันการทำงานดังนี้

- GenDhParams() เป็นฟังก์ชันในการกำหนดค่าตัวแปรที่ใช้ในการเข้ารหัสซึ่งค่าตัวแปรนี้สามารถใช้ได้ทั้งฝั่งของไคล์เอ็นต์และเซิร์ฟเวอร์ได้
- GenerateKey() เป็นฟังก์ชันที่ทำการสร้างคีย์ส่วนตัว(PrivateKey)และคีย์สาธารณะ(PublicKey) โดยทั้ง 2 ฝั่งมีการแลกเปลี่ยนคีย์สาธารณะระหว่างกันเพื่อนำไปใช้ในการสร้างคีย์ลับ
- GenerateSecretkey() เป็นฟังก์ชันที่ทำการสร้างคีย์ส่วนตัวและคีย์สาธารณะที่ได้จากการแลกเปลี่ยน กันระหว่างคู่สื่อสาร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.2.2 การทำงานในแต่ละโหมด

การทำงานในแต่ละโหมดนั้นจะเป็นส่วนหน้าจอหลักของโปรแกรมซึ่งประกอบด้วยเมนูบาร์ ทูลบาร์ ส่วนของอุปกรณ์เชื่อมต่อได้แก่ เราเตอร์รุ่นต่างๆสายเชื่อมต่อ ส่วนของเน็ตเวิร์กไดอะแกรม(drag and drop)สำหรับให้ผู้ใช้งานสร้างเครือข่าย และส่วนแสดงสถานะของเราเตอร์ สำหรับแสดงสถานะของเราเตอร์ได้แก่ ชื่อ รุ่น ชื่ออินเตอร์เฟส ไอพีแอดเดรสของแต่ละอินเตอร์เฟส และสถานะของอินเตอร์เฟสนั้นว่าทำงาน(Up) หรือไม่ทำงาน(Down)



รูปที่ 8-8 หน้าจอหลักของโปรแกรม

ฟังก์ชันการทำงานสำหรับส่วนนี้จะอยู่ในคลาส Frame3 , ImageMotion ,RouterConsole และ FrameR มีฟังก์ชันการทำงานหลักๆดังนี้

คลาส Frame3

- Frame3() เป็นคอนสตรัคเตอร์ของคลาส ในฟังก์ชันนี้จะแสดงหน้าจอโปรแกรมหลักทั้งส่วนของเมนูบาร์ ทูลบาร์ และส่วนของเน็ตเวิร์กไดอะแกรม
- CreateWire() เป็นฟังก์ชันสำหรับสร้างสายเชื่อมต่อของเราเตอร์ เมื่อผู้ใช้คลิ้กเพิ่มสายเชื่อมต่อเราเตอร์
- CreateRouter()เป็นฟังก์ชันสำหรับสร้างเราเตอร์ เมื่อผู้ใช้คลิ้กเพิ่มเราเตอร์
- OpenDiagram() เป็นฟังก์ชันสำหรับเปิดไฟล์ที่ทำการเซฟค่าคอนฟิกและไดอะแกรมที่สร้างเอาไว้ขึ้นมาเพื่อกำหนดให้กับค่าเราเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คลาส ImageMotion

- ImageMotion() เป็นคอนสตรัคเตอร์ของคลาสในฟังก์ชันนี้จะทำการวาดไดอะแกรมของเครือข่าย
- PaintComponent() เป็นฟังก์ชันในการวาดไดอะแกรมของเครือข่ายซึ่งจะเรียกใช้ ฟังก์ชัน paintRouter() และPaintWire() อีกที
- paintRouter() เป็นฟังก์ชันในการวาดรูปเราเตอร์ที่สร้างขึ้น
- PaintWire() เป็นฟังก์ชันในการวาดสายเชื่อมต่อที่สร้างขึ้น

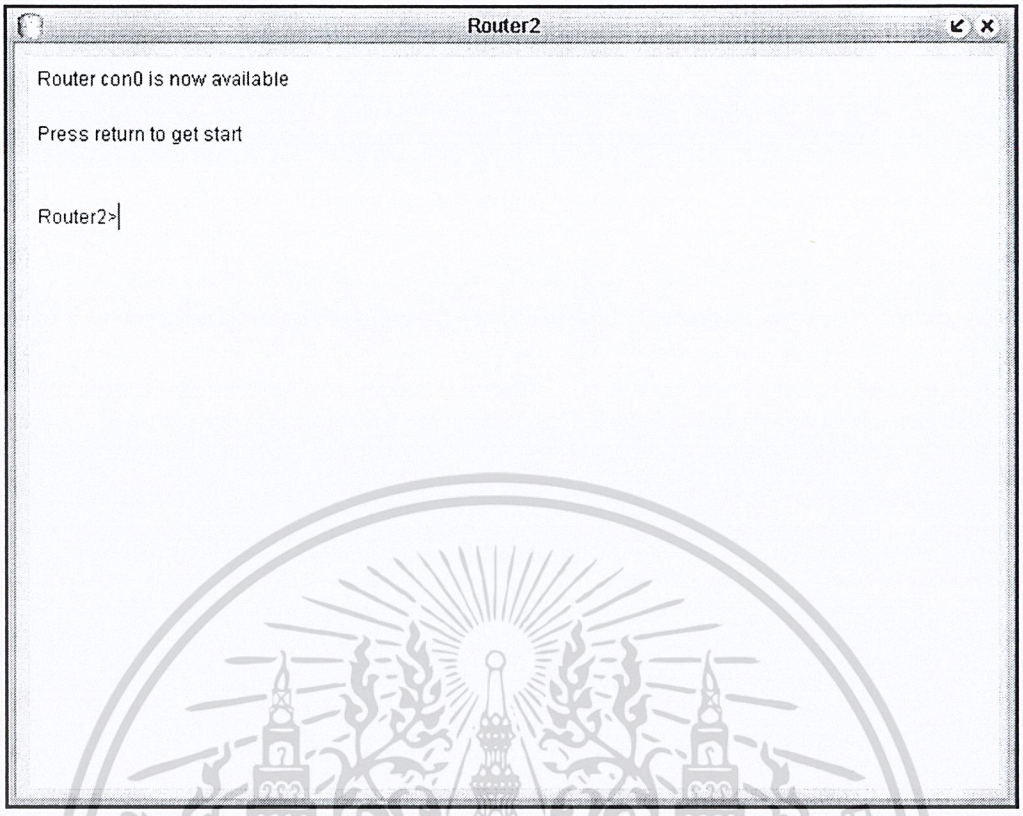
คลาส RouterConsole

- RouterConsole() เป็นคอนสตรัคเตอร์ของคลาสทำหน้าที่แสดงส่วนหน้าจอคอนโซล
- EnterKey() เป็นฟังก์ชันที่จัดการกับการทำงานของคีย์ Enter เพื่อเริ่มต้นการทำงาน เนื่องจากเราเตอร์สามารถกำหนดระดับความปลอดภัยของผู้ใช้ได้ เมื่อผู้ใช้กดคีย์ Enter ก็สามารถทำงานกับเราเตอร์ได้เลยหรือจะต้องใส่พาสเวิร์ดก่อนถึงจะเริ่มทำงานได้
- ShowPrompt() เป็นฟังก์ชันสำหรับแสดงพรอมต์ โดยจะดูจากโหมดการทำงานของเราเตอร์แล้วจึงแสดงพรอมต์ของโหมดการทำงานนั้นๆ สำหรับฟังก์ชันนี้จะรับอาร์กิวเมนต์เข้าไปเป็นสตริงเพื่อนำไปแสดงผลต่อจากพรอมต์
- ส่วนฟังก์ชันที่เหลือเป็นฟังก์ชันที่จัดการกับการกระทำที่เกี่ยวกับคีย์บอร์ดและเมาส์ เช่น ฟังก์ชัน jTextArea1_keyPressed() จัดการการกระทำที่เกี่ยวกับคีย์บอร์ดในขณะที่ TextArea ถูกโฟกัส ฟังก์ชัน jTextArea1_mouseClicked() จัดการการกระทำของการคลิกเมาส์ในขณะที่ TextArea ถูกโฟกัส และฟังก์ชัน jTextArea1_mouseReleased() จัดการกับการกระทำของการปล่อยเมาส์ ทำให้ผู้ใช้ไม่สามารถเลือกข้อความได้

คลาส FrameR

- FrameR() เป็นคอนสตรัคเตอร์ของคลาสในฟังก์ชันนี้จะทำการแอดคอมโพเนนต์ของ RouterConsole เข้าไปในเฟรมของคลาส

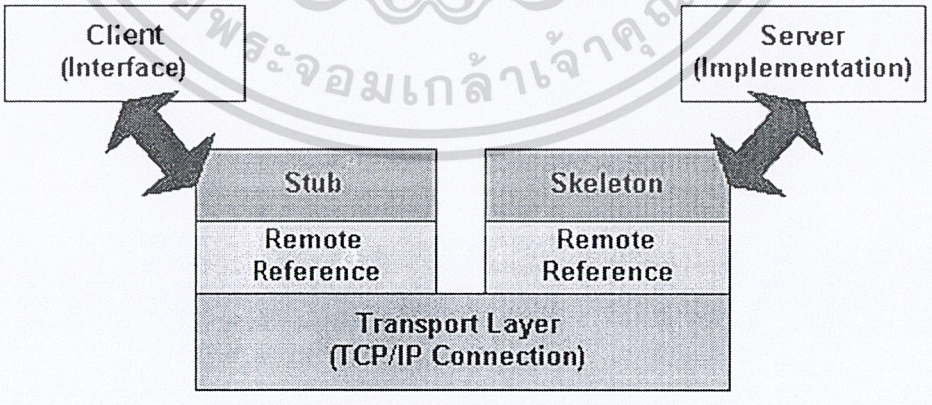
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8-9 ภาพแสดงคอนโซล

8.3 ส่วนของอาร์เอ็มไอ(RMI)

ส่วนของ Java RMI เป็นส่วนที่ใช้ติดต่อระหว่าง ไคลเอ็นต์ กับ เซิร์ฟเวอร์ การทำงานของโปรแกรมจะเป็นการติดต่อผ่านกระบวนการของอาร์เอ็มไอโดยอาร์เอ็มไอมีสถาปัตยกรรมดังนี้



รูปที่ 8-10 สถาปัตยกรรมของอาร์เอ็มไอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่งการติดต่อผ่านกระบวนการอาร์เอ็มไอนั้นมีอยู่ 3 ส่วน ที่สำคัญได้แก่ ส่วนของอินเตอร์ฟส

(Interface) ส่วนของอิมพลีเมนต์เทชัน(Implementation) และ ระบบของอาร์เอ็มไอ(RMI Systems)ในส่วนของระบบอาร์เอ็มไอก็มีอยู่3 เลเยอร์คือ สตับ/สเกลตัน(Stub/Skeleton) ส่วนที่สองคือ รีโมตรีเฟอร์เรนซ์เลเยอร์(Remote reference) และส่วนที่สาม คือทรานสปอร์ตเลเยอร์(Transport) ซึ่งสร้างการเชื่อมต่อระหว่างจาวาเวอร์ชวลมาชีนด้วย TCP/IP

แต่ในโปรแกรมนี้ทำการพัฒนาอยู่สองส่วนคือ ส่วนของอินเตอร์เฟส และ ส่วนของอิมพลีเมนต์เท-ชัน

ในส่วนนี้มีคลาสต่างๆที่เกี่ยวข้องดังนี้

คลาส NewRouterInt และ คลาส NewRouterImpl เป็นคลาสที่ทำหน้าที่ในการส่งค่าออบเจ็กต์และอาร์กิวเมนต์ของเราเตอร์ผ่านกระบวนการของอาร์เอ็มไอ โดยคลาส NewRouterInt เป็นคลาสอินเตอร์และคลาส NewRouterImpl จะทำการอิมพลีเมนต์คลาส NewRouterInt อีกที เมื่อผู้ใช้กระทำการใดๆที่เกี่ยวข้องกับเราเตอร์ เช่น สร้างเราเตอร์ ลบเราเตอร์ เป็นต้น ซึ่งค่าต่างๆที่เกี่ยวข้องกับเราเตอร์ จะถูกเก็บไว้ที่เซิร์ฟเวอร์ เมื่อทางด้านไคลเอ็นต์เรียกใช้ค่าเหล่านี้จะต้องเรียกใช้ผ่านคลาสทั้งสองนี้

คลาส NewRouterInt และ คลาส NewRouterImpl มีฟังก์ชันการทำงานหลักๆดังนี้

- getvRouter() เป็นฟังก์ชันที่รับค่าเวกเตอร์ที่เก็บจำนวนของเราเตอร์ทั้งหมดจากเซิร์ฟเวอร์
- DelRouter(int index) เป็นฟังก์ชันที่จัดการลบเราเตอร์ออกจากเวกเตอร์ที่เก็บอยู่ที่เซิร์ฟเวอร์
- getvStatus()เป็นฟังก์ชันที่รับค่าเวกเตอร์ที่เก็บคอมโพเนนต์ที่แสดงค่าสถานะของเราเตอร์จากเซิร์ฟเวอร์

คลาส CommandInt และคลาส CommandImpl เป็นคลาสที่ทำหน้าที่ในการจัดการเกี่ยวกับไฟล์ที่เก็บค่าคอนฟิกของเราเตอร์และไดอะแกรมที่สร้างขึ้น โดยคลาส CommandInt เป็นคลาสอินเตอร์และคลาส CommandImpl จะทำการอิมพลีเมนต์คลาส CommandInt อีกที

คลาส CommandInt และ คลาส CommandImpl มีฟังก์ชันการทำงานหลักๆดังนี้

- constructor () เป็นคอนสตรัคเตอร์ของคลาสและในคอนสตรัคเตอร์นี้มีการเรียกใช้ฟังก์ชัน readStartup() ของคลาส Command เพื่ออ่านค่าคอนฟิกที่เซฟไว้มากำหนดให้กำเราเตอร์เมื่อมีการเปิดไฟล์ไดอะแกรมขึ้นมา
- getFileBuffer() เป็นฟังก์ชันที่รับค่าไฟล์คอนฟิกที่อยู่เซิร์ฟเวอร์มาเขียนลงในไฟล์คอนฟิกเหมือนกัน แต่เป็นการเขียนลงที่ไคลเอ็นต์เมื่อไคลเอ็นต์ทำการเปิดไฟล์ไดอะแกรมขึ้นมา
- copyFile(String f) () เป็นฟังก์ชันที่ส่งค่าคอนฟิกของเราเตอร์ไปเขียนลงในไฟล์คอนฟิกที่เซร์ฟเวอร์ ในกรณีที่มีการเปลี่ยนแปลงค่าคอนฟิกรวมไปถึงเครือข่ายไดอะแกรมด้วย

คลาส RouterConsoleInt และคลาส RouterConsoleImpl เป็นคลาสที่ทำหน้าที่ในการจัดการเกี่ยวกับคำสั่งของเราเตอร์ทั้งหมดโดยคลาสRouterConsoleInt เป็นคลาสอินเตอร์และคลาส RouterConsoleImpl จะทำการอิมพลีเมนต์คลาส RouterConsoleInt อีกที เมื่อผู้ใช้งานป้อนคำสั่งที่คอนโซล คำสั่งจะถูกส่งไปดำเนินการที่เซิร์ฟเวอร์และส่งค่าผลลัพธ์กลับมาแสดงที่คอนโซลโดยผ่านคลาส 2 คลาสนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คลาส RouterConsoleInt และคลาส RouterConsoleImpl มีฟังก์ชันการทำงานหลักๆดังนี้

- runCMD(int index ,String cmd) เป็นฟังก์ชันที่จัดการกับคำสั่ง เมื่อผู้ใช้ป้อนคำสั่งคอนฟิกเราเตอร์ คำสั่งก็จะถูกส่งผ่านฟังก์ชันนี้ไปดำเนินการตามคำสั่งนั้นๆที่เซิร์ฟเวอร์
- tabCmd(int index ,String cmd)) เป็นฟังก์ชันที่จัดการกับคำสั่งแท็ปโดยเฉพาะ เมื่อผู้ใช้กดแท็บรหัสของแท็บก็จะถูกส่งผ่านฟังก์ชันนี้ไปดำเนินการที่เซิร์ฟเวอร์ เพื่อแสดงรายการคำสั่งทั้งหมดที่มีอยู่ในโหมดนั้นๆ

คลาส StatusPanelInt และคลาส StatusPanelImpl เป็นคลาสที่ทำหน้าที่ในการจัดการเกี่ยวกับการสร้างส่วนแสดงสถานะของเราเตอร์แต่ละตัว ตลอดจนปรับเปลี่ยนค่าสถานะ เมื่อมีการคอนฟิกเพื่อเปลี่ยนค่าสถานะต่างๆของเราเตอร์ ได้แก่ อินเตอร์เฟส ไอพีแอดเดรส ชื่อเราเตอร์ เป็นต้น คลาสStatusPanelInt เป็นคลาสอินเตอร์เฟสและคลาส StatusPanelImpl จะทำการอิมพลีเมนต์คลาส StatusPanelInt อีกที
คลาส StatusPanelInt และคลาส StatusPanelImpl มีฟังก์ชันการทำงานหลักๆดังนี้

- NewStatus () เป็นฟังก์ชันที่ทำการสร้างคอมโพเนนต์ที่ประกอบไปด้วยข้อความที่แสดงสถานะของเราเตอร์แต่ละตัว
- getPanel1() เป็นฟังก์ชันที่ทำการรับคอมโพเนนต์ที่ประกอบไปด้วยข้อความที่แสดงสถานะของเราเตอร์แต่ละตัวมาแสดงที่แถบแสดงสถานะของหน้าจอหลัก
- showStatusname(Router r1) เป็นฟังก์ชันที่ทำการแสดงค่าสถานะของเราเตอร์ที่เปลี่ยนแปลงอันเนื่องมาจากผลของการคอนฟิกค่าสถานะของเราเตอร์แต่ละตัว
- registerClient(ChangeStatus st) เป็นฟังก์ชันที่เซิร์ฟเวอร์ทำการเก็บออบเจ็กต์ของไคลเอ็นต์แต่ละตัวที่ทำการติดต่อกับเซิร์ฟเวอร์ในขณะที่เซิร์ฟเวอร์กำลังทำงานอยู่

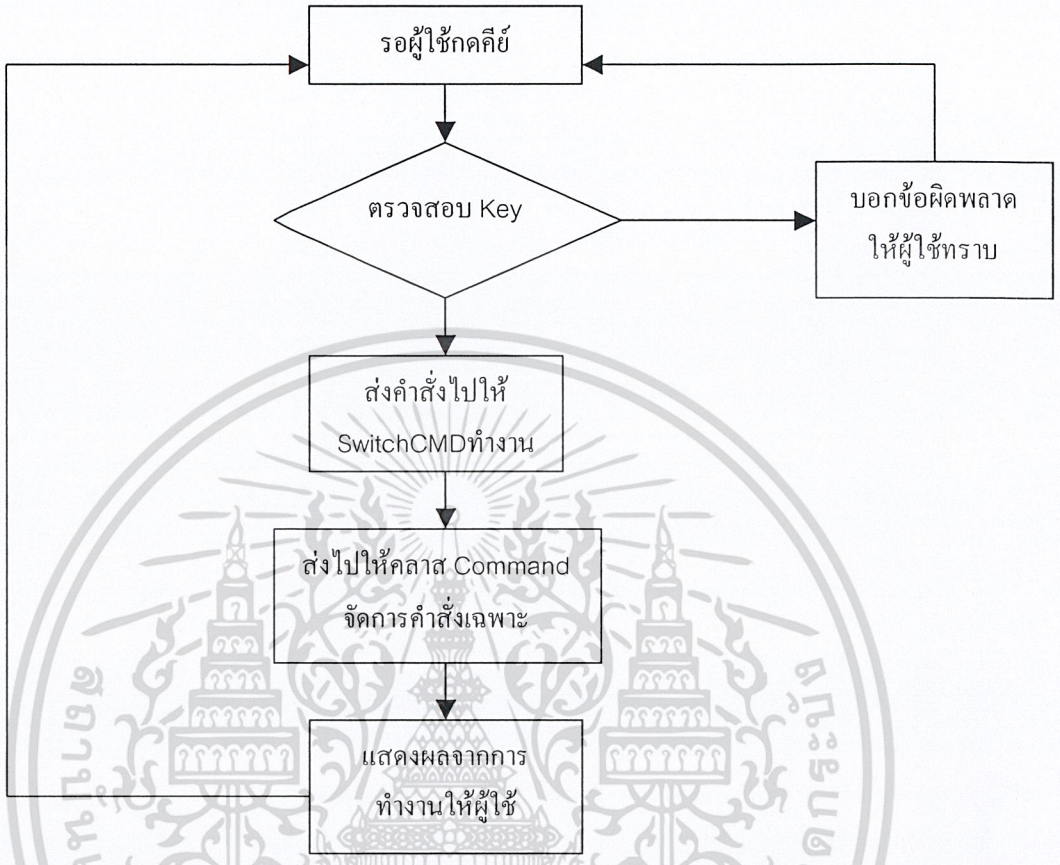
8.4 ส่วนของ เซิร์ฟเวอร์ เป็นส่วนทำงานหลักของโปรแกรม ได้แก่

8.4.1 ส่วนจัดการ Command Line และคำสั่งพื้นฐาน

มีคลาสที่จัดการส่วนนี้ 1 คลาสคือ คลาส SwitchCMD จะคอยรับคำสั่งที่ส่งมาจากคลาส RouterConsole ที่จัดการในส่วนที่ติดต่อกับผู้ใช้ ผ่านกระบวนการของอาร์เอ็มไอโดยคลาส RouterConsole Int และ คลาส RouterConsoleImpl เมื่อรับคำสั่งมาแล้วจะนำมาตรวจสอบว่าเป็นคำสั่งที่ถูกต้อง สามารถทำงานได้หรือไม่ ถ้าถูกต้องจะส่งคำสั่งไปทำงาน และรอรับผลการทำงาน เพื่อนำมาแสดงผลให้ผู้ใช้ทราบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนการทำงานของโปรแกรม



รูปที่ 8-11 แสดงขั้นตอนการจัดการกับส่วน Commands line

ฟังก์ชันการทำงานสำหรับส่วนนี้จะอยู่ในคลาส SwitchCMD และ Command มีฟังก์ชันการทำงานดังนี้

คลาส SwitchCMD

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คลาส Command

- eraseStartUp() จะเปิดไฟล์ config.dat ขึ้นมาอ่าน แล้วนำค่าที่อ่านได้เก็บไว้เรื่อยๆ จนกระทั่งตรวจสอบพบว่าเป็นส่วนที่เก็บค่าของเราเตอร์ตัวที่ต้องการลบค่า ก็จะไม่นำค่าที่อ่านมาได้ไปเก็บจนหมดของเราเตอร์ตัวดังกล่าวก็จะอ่านค่าแล้วเก็บไว้เช่นเดิม จนกระทั่งหมดไฟล์ ก็จะปิดไฟล์แล้วเปิดขึ้นมาเพื่อเขียนไฟล์ใหม่ จากนั้นนำค่าที่ได้จากการอ่านครั้งที่แล้วเขียนทับลงไปที่ไฟล์ config.day แล้วจึงปิดไฟล์
- ping() ฟังก์ชันนี้จะทำงานแบบรีเคอร์ซีพ โดยอ่านค่าที่เก็บไว้ในตารางค้นหาเส้นทางเพื่อตรวจสอบว่ามีการเก็บข้อมูลสำหรับไปยังจุดหมายปลายทางนั้นๆ แล้วไปตรวจสอบที่เราเตอร์ต่อๆ ไปจนกว่าจะถึงจุดหมายปลายทาง หรือไม่สามารถหาข้อมูลเพื่อไปยังจุดหมายปลายทางนั้นได้
- readStartUp() ฟังก์ชันนี้จะอ่านค่าที่เก็บไว้ในไฟล์ config.dat มาแล้วสร้างเราเตอร์ขึ้นมา แล้วกำหนดค่าตามที่กำหนดไว้ในไฟล์ที่ละเราเตอร์ จนกระทั่งหมดไฟล์
- saveRunningConfig() เช่นเดียวกับฟังก์ชัน eraseStartUp() เมื่อพบเราเตอร์ที่ต้องการจะอ่านค่าที่กำหนดให้กับเราเตอร์ในขณะนั้นมาแทนการอ่านจากไฟล์ แล้วจึงเขียนทับไฟล์เดิม
- show_ip_ospf_inf() ทำหน้าที่แสดงผลจากคำสั่ง show ip ospf <interface name> เพื่อบอกว่าที่อินเทอร์เฟสนั้นๆ มีสถานะของโพรโตคอลโอเอสพีเอฟเป็นเช่นไร
- show_ip_ospf_neighbor() ทำหน้าที่แสดงเราเตอร์ข้างเคียงที่อยู่ในโพรโคตอลโอเอสพีเอฟเช่นเดียวกัน
- showCdp() ฟังก์ชันนี้จะแสดงผลออกมาโดยดึงค่า CDP ที่เก็บไว้ของเราเตอร์ออกมาแสดงผล
- showCdpEntry() ฟังก์ชันนี้จะหาเราเตอร์ข้างเคียงก่อน หลังจากนั้นจึงนำค่า CDP ของเราเตอร์ข้างเคียงมาแสดงผล
- showCdpNeighbors() เช่นเดียวกับ showCdpEntry() แต่ข้อมูลที่นำมาแสดงเป็นข้อมูลคนละประเภทกันคือ เป็นข้อมูลเกี่ยวกับอินเทอร์เฟสที่ต่อกัน
- showHistory() ฟังก์ชันนี้จะนำข้อมูลเกี่ยวกับคำสั่งที่เคยใช้งานของเราเตอร์ออกมาแสดงผล
- showInt() ฟังก์ชันนี้จะแสดงข้อมูลของแต่ละอินเทอร์เฟสของเราตอร์
- showIPProtocol() ฟังก์ชันนี้จะแสดงข้อมูลโพรโตคอลของเราเตอร์ โดยดูจากค่าตัวแปร RouteType ของเราเตอร์
- showIpRoute() ฟังก์ชันนี้จะแสดงข้อมูลของแต่ละเส้นทางในตารางค้นหาเส้นทาง
- showRunningConfig() ฟังก์ชันนี้จะดูข้อมูลการกำหนดค่าของเราเตอร์ในขณะนั้นมาแสดงผล
- showStartUp()ฟังก์ชันนี้จะทำงานคล้ายกับฟังก์ชัน eraseStartUp() แต่กลับกันโดยเมื่อเจอเราเตอร์ที่ต้องการแล้วจึงจะนำค่าที่อ่านได้มาเก็บไว้ แต่หลังจากที่อ่านเสร็จจะไม่มีการเขียนไฟล์ แต่จะนำค่าที่อ่านได้แสดงผลออกมาแทน
- showVersion() ฟังก์ชันนี้จะแสดงผลเกี่ยวกับเวอร์ชันของ IOS และข้อมูลของเราเตอร์ โดยจึงจากค่าตัวแปรของเราเตอร์

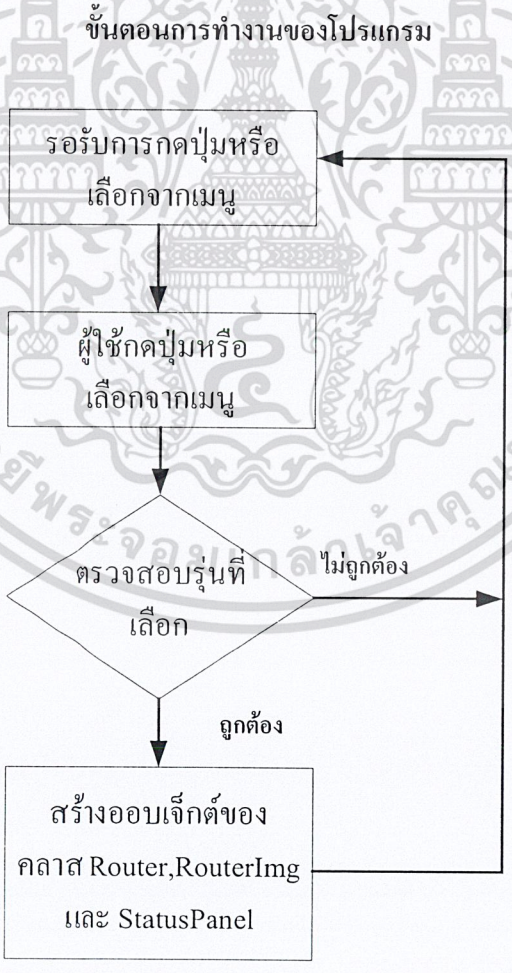
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น และห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- telnetCmd() ฟังก์ชันนี้จะตรวจสอบว่าไอพีที่ต้องการเทลเน็ตเข้าไปนั้นถูกต้องหรือไม่ และตรวจสอบต่อว่าพอร์ตเทลเน็ตเปิดไว้หรือไม่
- trace() คล้ายกับฟังก์ชัน ping() แต่จะแสดงผลแต่ละเส้นทางที่ค้นหาได้ออกมาด้วย

8.4.2 การจัดการส่วนของการ เพิ่มเราเตอร์

คลาสที่จัดการส่วนนี้คือคลาส Frame3 เป็นการจัดการกับไอคอน โดยโปรแกรมจะดักจับการกดปุ่มเพิ่มเราเตอร์ เมื่อผู้ใช้กดปุ่มเพิ่มเราเตอร์ โปรแกรมจึงจะสร้างออบเจ็กต์ของเราเตอร์ขึ้นมา พร้อมกับออบเจ็กต์ของสถานะของเราเตอร์ที่ใช้ในส่วนของการแสดงสถานะของเราเตอร์ขึ้นมา จากนั้นจะนำเก็บค่าตัวอ้างอิง (Reference) ของพื้นที่นั้นไว้ เพื่อใช้สำหรับอ้างถึงภายหลัง คลาสที่เกี่ยวข้องในส่วนนี้ได้แก่

- คลาส Router เป็นคลาสที่เก็บคุณสมบัติต่างๆของเราเตอร์ไว้เช่น ชื่อ อินเตอร์เฟส โพรโตคอล เป็นต้น
- คลาส RouterImg เป็นคลาสที่แสดงรูปภาพของเราเตอร์แต่ละรุ่น
- คลาส StatusPanel เป็นคลาสที่แสดงสถานะต่างของเราเตอร์ที่สร้างขึ้นมาใหม่



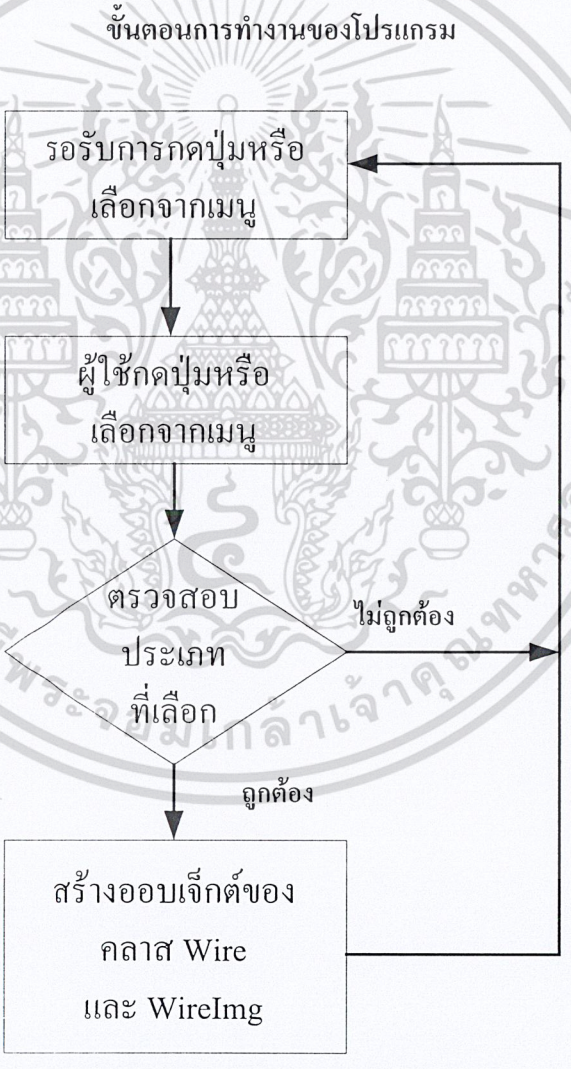
รูปที่ 8-12 แสดงขั้นตอนการทำงานของการจัดการเพิ่มเราเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.4.3 การจัดการส่วนของการ เพิ่มสายเชื่อมต่อ

คลาสที่จัดการส่วนนี้คือคลาส Frame3 เป็นการจัดการกับไอคอน โดยโปรแกรมจะดักจับการกดปุ่มเพิ่มสายเชื่อมต่อ เมื่อผู้ใช้กดปุ่มเพิ่มสายเชื่อมต่อ โปรแกรมจึงจะสร้างออบเจ็กต์ของสายเชื่อมต่อขึ้นมา จากนั้นจะนำเก็บค่าตัวอ้างอิง (Reference) ของพื้นที่นั้นไว้ เพื่อใช้สำหรับอ้างถึงภายหลัง คลาสที่เกี่ยวข้องในส่วนนี้ได้แก่

- คลาส Wire เป็นคลาสที่เก็บคุณสมบัติของสายเชื่อมต่อไว้เช่น ประเภท, แบนด์วิดธ์ และ ค่าคอสท์ เป็นต้น
- คลาส WireImg คลาส RouterImg เป็นคลาสที่แสดงรูปภาพของสายเชื่อมต่อแต่ละประเภท

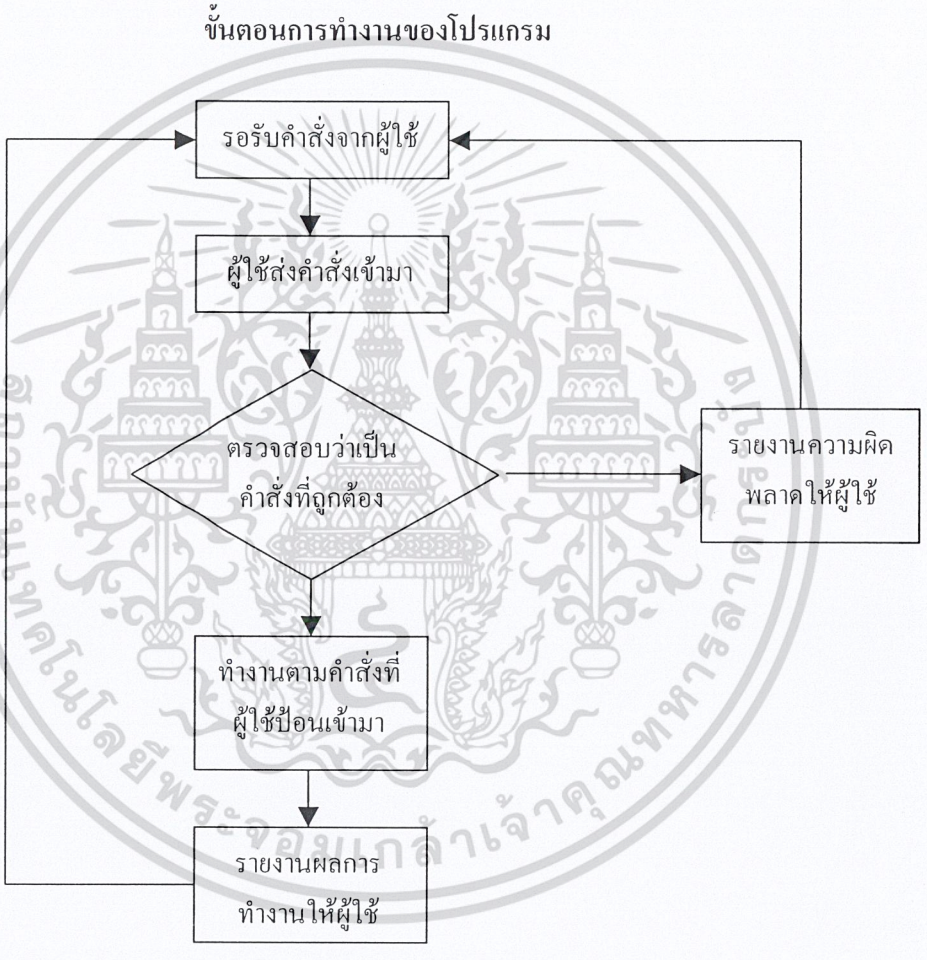


รูปที่ 8-13 แสดงขั้นตอนการทำงานของการจัดการเพิ่มสายเชื่อมต่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.4.4 การจัดการส่วนของอินเตอร์เฟส

คลาสที่จัดการส่วนนี้คือคลาส Interface โดยการทำงานจะเป็นการทำงานตามคำสั่งที่ผู้ใช้ส่งมา เช่น เพิ่มอินเทอร์เฟสให้กับเราเตอร์ โปแกรมจะตรวจสอบค่าไอพีแอดเดรสที่ส่งมากับคำสั่งว่าเป็นค่าที่ใช้งานได้ แล้วสร้างออบเจ็กต์ของอินเทอร์เฟสขึ้นมา แล้วนำค่าอ้างอิงไปเก็บเป็นแบบเวกเตอร์ในออบเจ็กต์เราเตอร์ เพื่อใช้ในการอ้างอิงภายหลัง แต่ถ้าเป็นคำสั่งอื่นๆ ที่จัดการกับอินเทอร์เฟสที่มีอยู่แล้ว โปรแกรมจะใช้ค่าอ้างอิงออบเจ็กต์เพื่อหาอินเทอร์เฟสที่ต้องการ แล้วจึงทำงานกับอินเทอร์เฟสนั้นๆ ตามคำสั่งที่ส่งมา



รูปที่ 8-14 แสดงขั้นตอนการจัดการอินเทอร์เฟสของเราเตอร์

ฟังก์ชันการทำงานนี้จะอยู่ในคลาส Interface มีฟังก์ชันการทำงานดังนี้

- Interface() เป็นคอนสตรัคเตอร์ของคลาส Interface ทำหน้าที่กำหนดค่าเริ่มต้นให้กับอินเทอร์เฟส เช่น ชื่อของอินเทอร์เฟส เป็นต้น
- AddNewInt() เป็นฟังก์ชันสำหรับเพิ่มอินเทอร์เฟสให้กับเราเตอร์ แต่จะต้องตรวจสอบก่อนว่าจำนวนของอินเทอร์เฟสประเภทนั้นเกินหรือไม่ ถ้าไม่เกินจึงจะสามารถเพิ่มอินเทอร์เฟสได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- findNetAdd() เป็นฟังก์ชันสำหรับหาค่าเลขประจำเครือข่าย (Net Address) เพื่อนำไปใช้ในการทำงานอื่น สำหรับฟังก์ชันที่ไม่ส่งค่าอาร์กิวเมนต์ใดๆ มาให้ถือว่าให้หาค่าของอินเทอร์เฟสที่เรียกฟังก์ชันนั้น แต่ถ้าส่งค่าไอพีแอดเดรสกับค่าของซับเน็ตมาให้จะหาค่าของเลขประจำเครือข่ายโดยพิจารณาจากค่าทั้ง 2 ค่านี้
- isIp() ฟังก์ชันนี้จะรับอาร์กิวเมนต์มาเป็นสตริง เพื่อนำมาตรวสอบว่าค่าไอพีแอดเดรสนั้นเป็นค่าไอพีที่ถูกต้องหรือไม่ โดยค่าของไอพีแอดเดรสจะต้องมี 4 ชุด แต่ละชุดจะต้องมีค่าอยู่ระหว่าง 0-255
- noshutInt() เป็นการกำหนดให้อินเทอร์เฟสนั้นอยู่ในสถานะที่ทำงาน
- shutInt() เป็นการกำหนดให้อินเทอร์เฟสอยู่ในสถานะที่ไม่สามารถทำงานได้
- setIpAddress() เป็นการกำหนดค่าไอพีแอดเดรสให้กับอินเทอร์เฟสล่าสุดที่พึ่งทำงานไป โดยจะดึงอินเทอร์เฟสตัวสุดท้ายมาแล้วกำหนดค่าให้กับอินเทอร์เฟสนั้น

8.5 ฟังก์ชันกำหนดทราฟฟิกบนเครือข่าย

คลาส Traffic เป็นคลาสที่จัดการเกี่ยวกับการกำหนดลักษณะของทราฟฟิกของเครือข่ายที่สร้างขึ้นด้วยกัน 3 แบบได้แก่ แบบค่าคงที่(constants) แบบสุ่มค่า(random) และแบบพัลส์(pulse)

- แบบค่าคงที่เป็นการให้ผู้ใช้กำหนดค่าของความเร็วในการส่งข้อมูลมีหน่วยเป็น เมกกาบิทต่อวินาที โดยมีค่าอยู่ระหว่าง 0 –1000 เมกกาบิทต่อ วินาที แล้วนำค่าที่ได้ไปคำนวณค่าคอสท์ของสายเชื่อมต่อ

- แบบสุ่มค่าเป็นการโปรแกรมจะทำการสุ่มค่าความเร็วในการส่งข้อมูลขึ้นมา ค่าที่สุ่มจะอยู่ระหว่าง 0 –1000 เมกกาบิทต่อ วินาที แล้วนำค่าที่ได้ไปคำนวณค่าคอสท์ของสายเชื่อมต่อ

- แบบพัลส์เป็นการให้ผู้ใช้กำหนดค่าของความเร็วในการส่งข้อมูลขึ้นมา ค่าจะอยู่ระหว่าง 0 – 1000 เมกกาบิทต่อ วินาที กำหนดค่า t1 และ t2 มีหน่วยเป็น วินาที แล้วนำค่าที่ได้ไปคำนวณค่าคอสท์ของสายเชื่อมต่อ

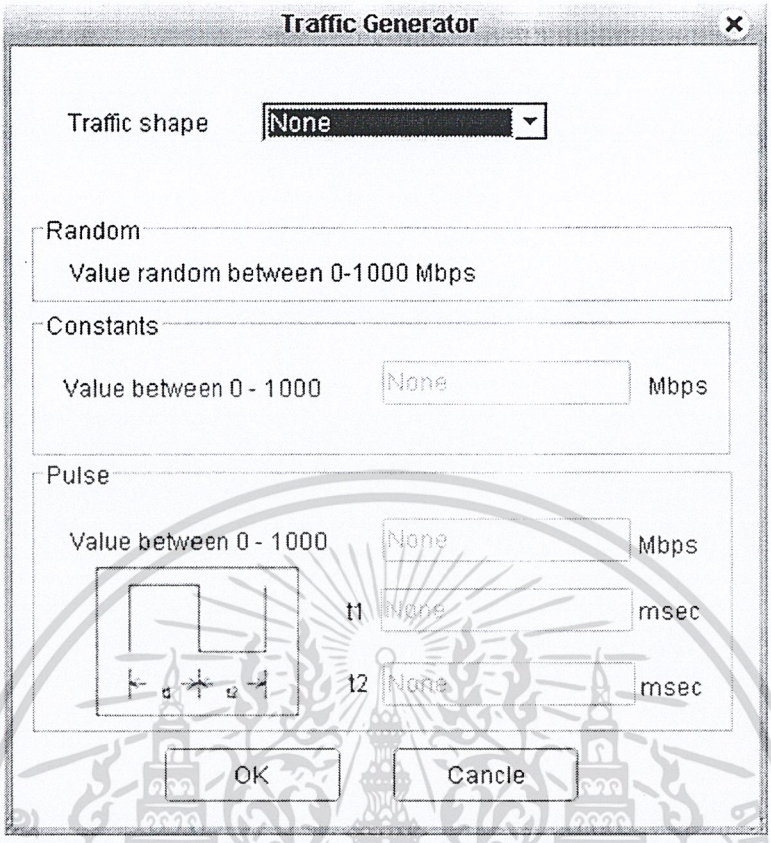
คลาสที่เกี่ยวข้องในฟังก์ชันนี้คือ คลาส Traffic ซึ่งเป็นคลาสที่จัดการในส่วนนี้มีฟังก์ชันการทำงานหลักๆดังนี้

- Calculate_Cost(int BW_usedAll) เป็นฟังก์ชันที่ทำการคำนวณค่าคอสท์ของสายเชื่อมต่อเราเตอร์โดยนำค่าต่างๆที่ผู้ใช้งานป้อนเข้ามาทำการคำนวณ ซึ่งสูตรในการคำนวณค่าคอสท์ของโอเอสพีเอฟคือ

  คอสท์ $= 10^8$ / แบนด์วิดธ์ของสาย

  ค่าคอสท์ที่คำนวณได้จะได้เป็นจำนวนเต็มถ้ามีเศษจะตัดเศษทิ้ง
- randomValue() เป็นฟังก์ชันสุ่มค่าความเร็วในการส่งข้อมูลซึ่งค่าที่สุ่มจะอยู่ระหว่าง 0-1000 ซึ่งเป็นค่าจำนวนเต็ม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8-15 แสดงรูปไดอะล็อกกำหนดทราฟฟิกบนเครือข่าย

8.6 การจัดการเกี่ยวกับโพรโตคอลอาร์ไอพี

คลาส RIP นี้เป็นคลาสที่สืบทอด (Inherit) มาจากคลาสแม่ที่เป็นเทรด (Thread) เนื่องจากการทำงานโพรโตคอล RIP จะส่งตารางค้นหาเส้นทางให้กันตลอดเวลา ดังนั้นการแตกโพรเซสให้โปรแกรมจัดการกับส่วนที่แลกเปลี่ยนตารางเส้นทางเป็นโพรเซสแบบเทรด จะสิ้นเปลืองเวลาที่ใช้น้อยกว่า รอให้โปรแกรมทำงานเอง สำหรับส่วนของตารางค้นหาเส้นทาง เราเตอร์แต่ละตัวจะมีเวกเตอร์ที่เก็บค่าอ้างอิงของออบเจ็กต์ของคลาส RoutingTable

วิธีแลกเปลี่ยนตารางค้นหาเส้นทาง เนื่องจากเป็นโปรแกรมจำลองจึงไม่มีการส่งข้อมูลจริงๆ แต่จะตรวจสอบว่าเป็นเราเตอร์ตัวที่อยู่ติดกันเท่านั้นจึงจะสามารเข้าไปนำข้อมูลของตารางค้นหาเส้นทางออกมาได้ เปรียบเสมือนกับการทำงานจริงๆ ที่เราเตอร์ข้างเคียงเท่านั้นที่จะได้รับข้อมูลตารางค้นหาเส้นทางเมื่อได้ ข้อมูลของตารางค้นหาเส้นทางมาแล้วการพิจารณาว่าจะจัดการกับข้อมูลนั้นอย่างไรจะทำตามโพรโตคอลอาร์ไอพี คือแบ่งการตรวจสอบออกเป็น 3 กรณีดังนี้

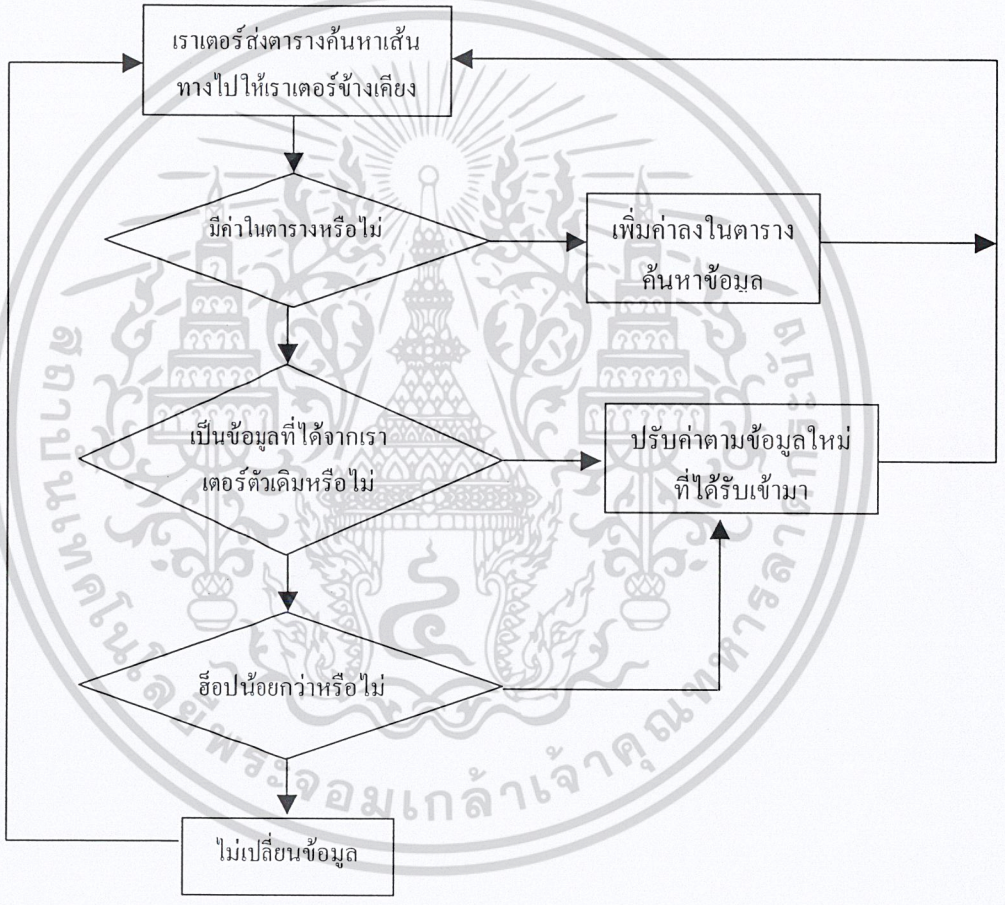
1. ถ้าข้อมูลตารางค้นหาเส้นทางที่ได้รับมานั้น เป็นข้อมูลของจุดหมายปลายทางที่ไม่มีอยู่ในตารางค้นหาเส้นทาง โปรแกรมจะเพิ่มเส้นทางนั้นลงในตารางค้นหาเส้นทางโดยสร้างออบเจ็กต์ของRoutingTable แล้วเก็บค่าอ้างอิงไว้ที่เราเตอร์
2. ถ้าข้อมูลตารางค้นหาเส้นทางที่ได้รับมานั้น เป็นข้อมูลของจุดหมายปลายทางที่มีอยู่แล้วในตารางค้นหาเส้นทางให้พิจารณาค่าของเมตริกซ์ต่อไปโดย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ถ้าค่าเมตริกใหม่มีค่าน้อยกว่าเส้นทางเดิม จะเปลี่ยนค่าในตารางเลือกเส้นทางตามทางเส้นทางใหม่
- ถ้าค่าเมตริกใหม่มีค่ามากกว่าหรือเท่ากับเส้นทางเดิม จะไม่เปลี่ยนค่าใดๆ ในตารางค้นหาเส้นทาง

3. ถ้าข้อมูลตารางค้นหาเส้นทางที่ได้รับมานั้น เป็นข้อมูลที่มีเกตเวย์เป็นเราเตอร์ตัวเดียวกันกับเราเตอร์ตัวที่ส่งตารางค้นหาเส้นทางมาให้ ให้ปรับค่าตารางค้นหาเส้นทางนั้นตามค่าที่ได้รับมาใหม่ โดยไม่สนใจว่าค่าเมตริกซ์ใหม่นั้นจะมีค่ามากกว่าหรือน้อยกว่าเส้นทางเดิม

ขั้นตอนการทำงานของโปรแกรม



รูปที่ 8-16 ขั้นตอนการทำงานของโพรโตคอล RIP

ฟังก์ชันการทำงานนี้จะอยู่ในคลาส RIP และ Router มีฟังก์ชันการทำงานดังนี้

คลาส Rip

- run() เป็นฟังก์ชันสำหรับให้เทรดทำงาน โดยจะทำงานทุกๆ 30 วินาที เพื่อเรียกให้ฟังก์ชันของเราเตอร์ให้ส่งตารางค้นหาเส้นทาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คลาส Router

- sendRoutingTable() ฟังก์ชันนี้จะตรวจสอบว่ามีเราเตอร์ใดอยู่ข้างเคียงบ้าง และเราเตอร์นั้นจะต้องใช้โพรโตคอลอาร์ไอพีด้วย แล้วจึงเรียกฟังก์ชัน recieveRoutingTable() เพื่อให้เราเตอร์นั้นรับค่าตารางค้นหาเส้นทาง
- recieveRoutingTable()ฟังก์ชันนี้จะตรวจสอบค่าตารางเส้นทางที่ได้รับมาและปรับเปลี่ยนตามโพรโตคอล

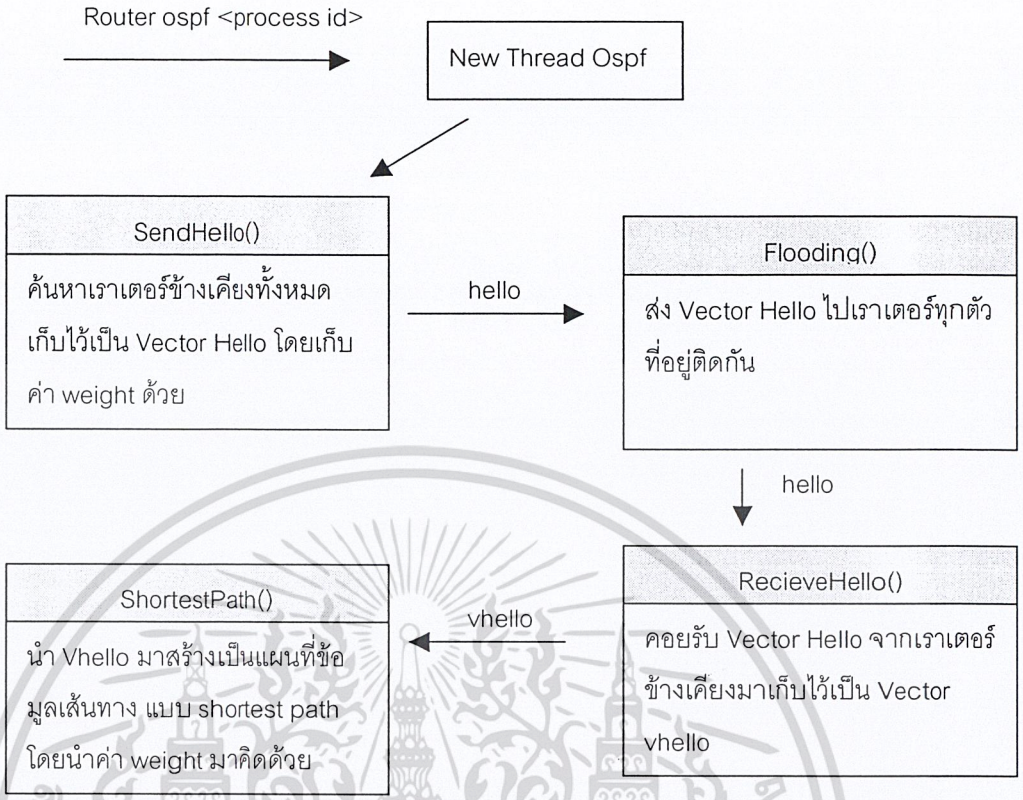
8.7 การจัดการเกี่ยวกับโพรโตคอลโอเอสพีเอฟ

เมื่อผู้ใช้ป้อนคำสั่ง router ospf <Process id> โปรแกรมจะทำการสร้าง Thread ของคลาส Ospf ขึ้นมาใหม่

การทำงานของโพรโตคอลโอเอสพีเอฟหลังจากสร้าง Thread มีขั้นตอนหลัก 3 ขั้นตอน คือ

1. ตรวจหาเราเตอร์อื่นที่อยู่ข้างเคียงโดยใช้ฟังก์ชัน sendHello() โดยฟังก์ชันนี้จะทำการตรวจสอบหาเราเตอร์ที่อยู่ข้างเคียงมาเก็บไว้แบบเวกเตอร์
2. เราเตอร์ประกาศเวกเตอร์นี้ออกไปให้เราเตอร์อื่นทุกตัวโดยใช้ฟังก์ชัน Flooding() และเราเตอร์อื่นจะรับโดยใช้ฟังก์ชัน recieveHello() เมื่อเสร็จสิ้นขั้นตอนนี้เราเตอร์ จะทราบว่ามีเราเตอร์อื่นใดอยู่ในระบบบ้าง
3. เราเตอร์จะอาศัยข้อมูลจากขั้นที่สองสร้างแผนที่ไปยังเราเตอร์อื่นทุกตัวในเครือข่ายโดยใช้ฟังก์ชัน shortestPath() และเก็บแผนที่ข้อมูลเส้นทางนี้เป็นแบบเวกเตอร์ โดยจะเป็นเส้นทางต้นทางไปยังทุกเราเตอร์ปลายทาง
4. เนื่องจากมีการทำงานเป็นแบบ Thread จึงทำให้แผนที่ข้อมูลเส้นทางถูกปรับปรุงทุกๆ ช่วงเวลาที่กำหนด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8-17 ขั้นตอนการทำงานของโพรโตคอลโอเอสพีเอฟ

ฟังก์ชันการทำงานนี้จะอยู่ในคลาส Ospf และ Router มีฟังก์ชันการทำงานดังนี้

คลาส Ospf

- run() เป็นฟังก์ชันสำหรับให้เทรดทำงาน โดยจะทำงานทุกๆ ช่วงเวลา เพื่อเรียกให้ฟังก์ชันของเราเตอร์ให้ทำฟังก์ชันดังต่อไปนี้
- sendHelloPackage(r) ทำหน้าที่ตรวจสอบหาเราเตอร์ข้างเคียงมาเก็บไว้แบบเวกเตอร์
- flooding(r) ทำหน้าที่ส่งเวกเตอร์ที่ได้จาก sendHelloPackage(r) ไปแบบทุกทิศทุกทางที่ติดกับตัวเอง
- shortest_path(r) คำนวณหาระยะทางที่สั้นที่สุดจากตารางข้อมูลที่ได้
- recieveHelloTable() ทำหน้าที่คอยรับเวกเตอร์ที่ได้จากฟังก์ชัน sendHelloTable()มาเก็บไว้เป็นฐานข้อมูลเพื่อใช้ในการหาเส้นทางที่สั้นที่สุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

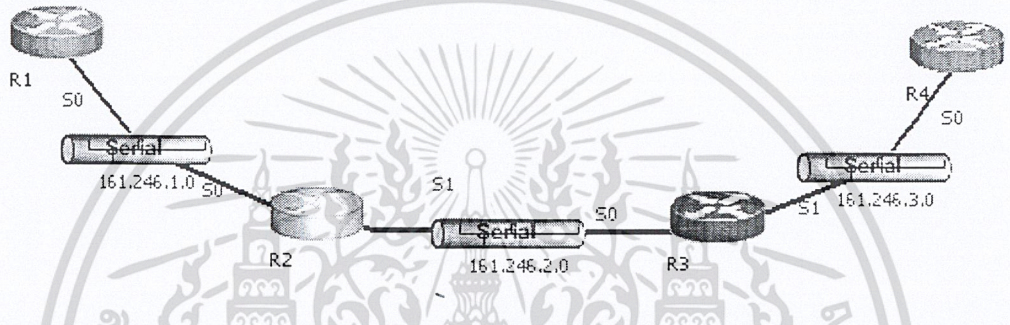
บทที่ 9

ตัวอย่างและการทดสอบการทำงานของโปรแกรม

9.1 ตัวอย่างการเลือกเส้นทางแบบสแตติก

ตัวอย่างนี้เป็นการคอนฟิกเราเตอร์โดยใช้การเลือกเส้นทางแบบสแตติกมีลำดับการทำงานดังนี้

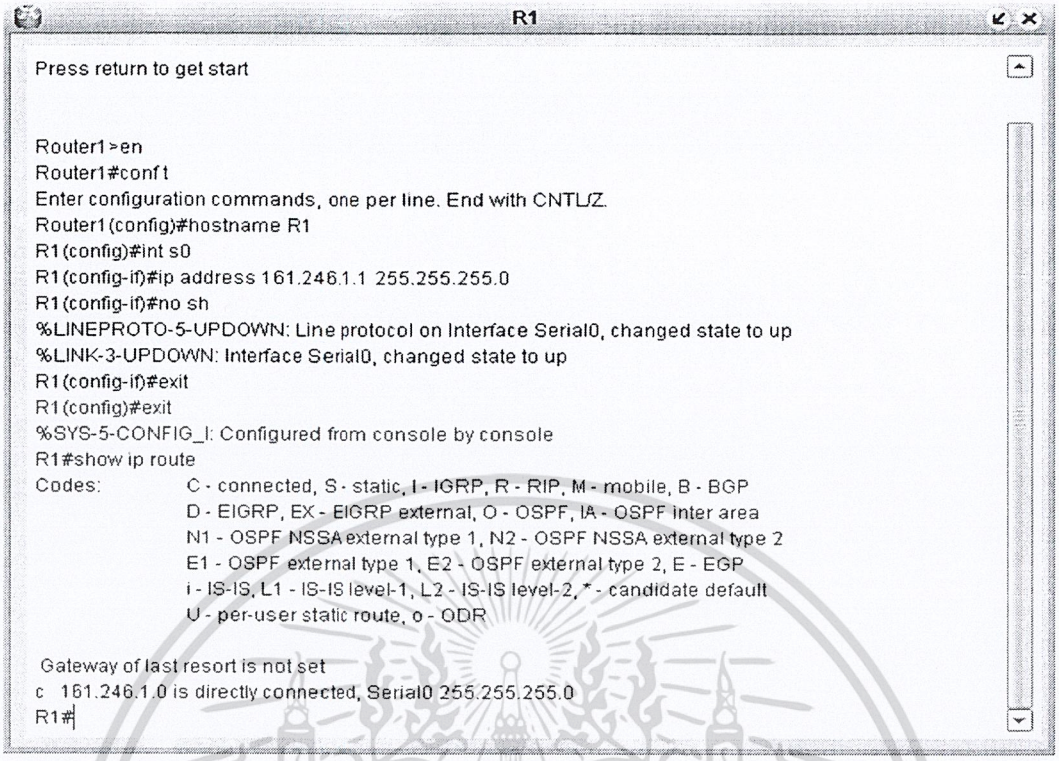
1. เปิดโปรแกรมขึ้นมาทำการสร้างเครือข่ายเราเตอร์ดังรูป



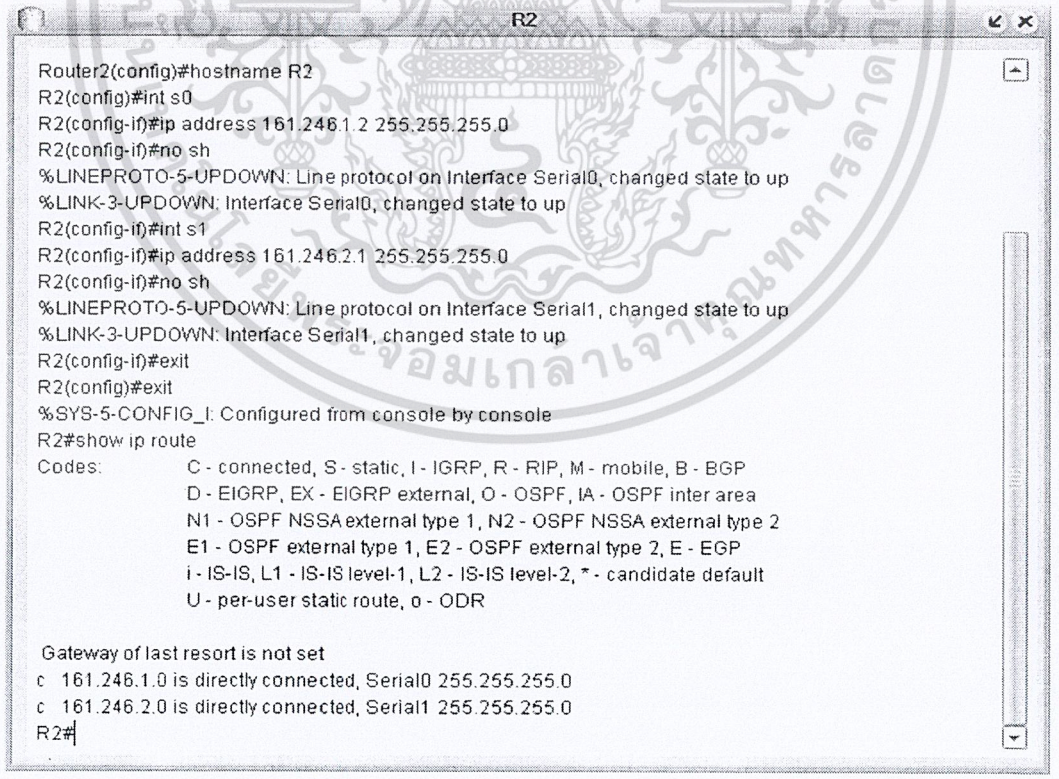
รูปที่ 9-1 รูปเครือข่ายเราเตอร์

2. จากนั้นทำการดับเบิ้ลคลิ้กที่ตัวเราเตอร์แต่ละตัวก็จะแสดงหน้าจอคอนโซลขึ้นมา แล้วทำการคอนฟิกเราเตอร์ให้ครบทั้ง 4 ตัวโดยกำหนดอินเตอร์เฟสตามเครือข่ายที่สร้างขึ้นแล้วกำหนดแต่ละอินเตอร์เฟสให้อยู่ในสถานะที่ทำงาน(UP) จากนั้นลองใช้คำสั่ง show ip route เพื่อดูค่าตารางเลือกเส้นทางดังตัวอย่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

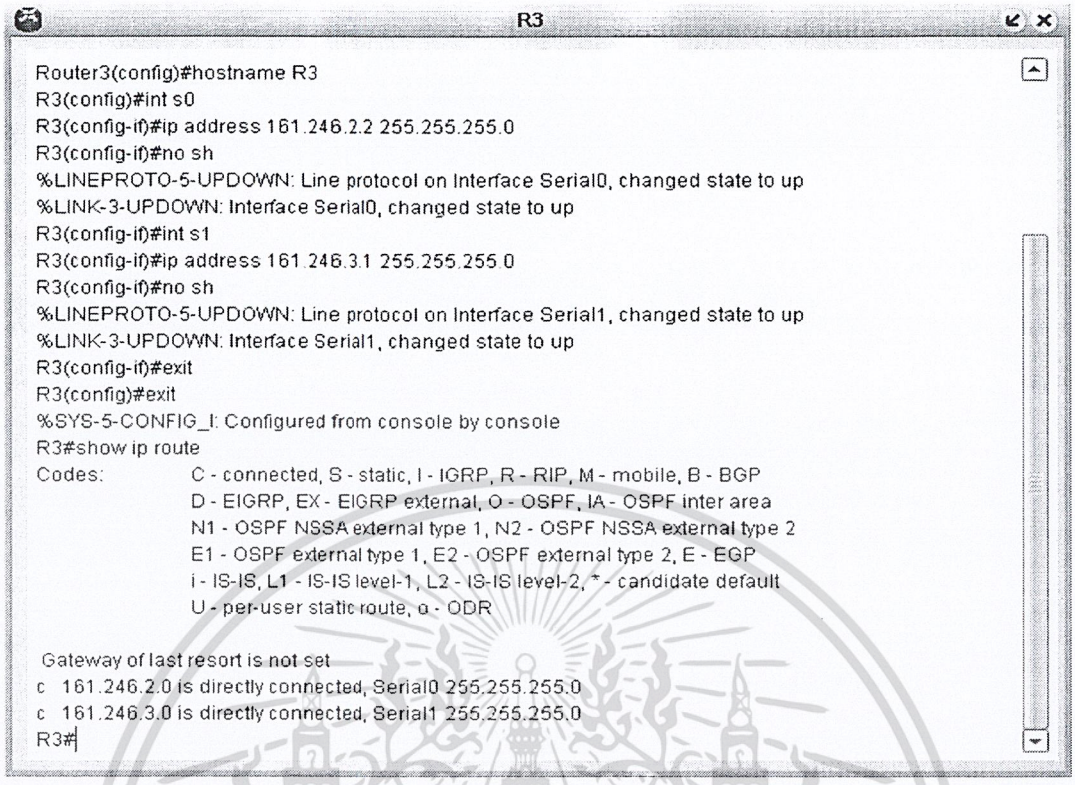


รูปที่ 9-2 การคอนฟิกการเลือกเส้นทางแบบสแตติกของเราเตอร์ R1

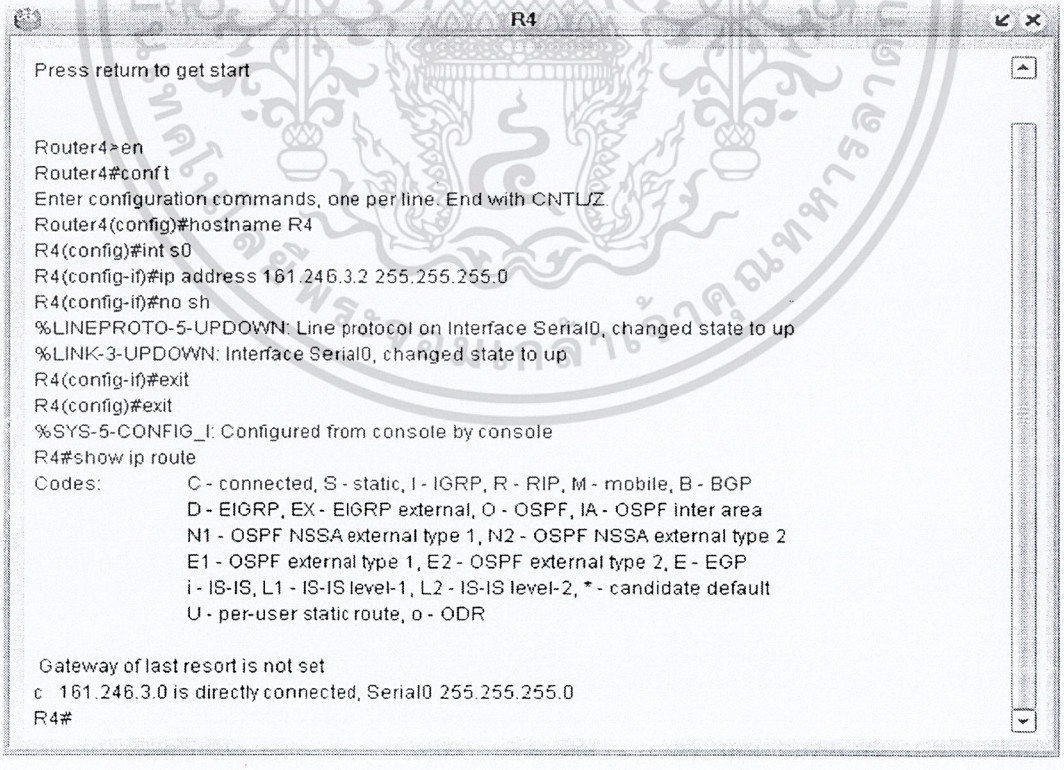


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 9-3 การคอนฟิกการเลือกเส้นทางแบบสแตติกของเราเตอร์ R2



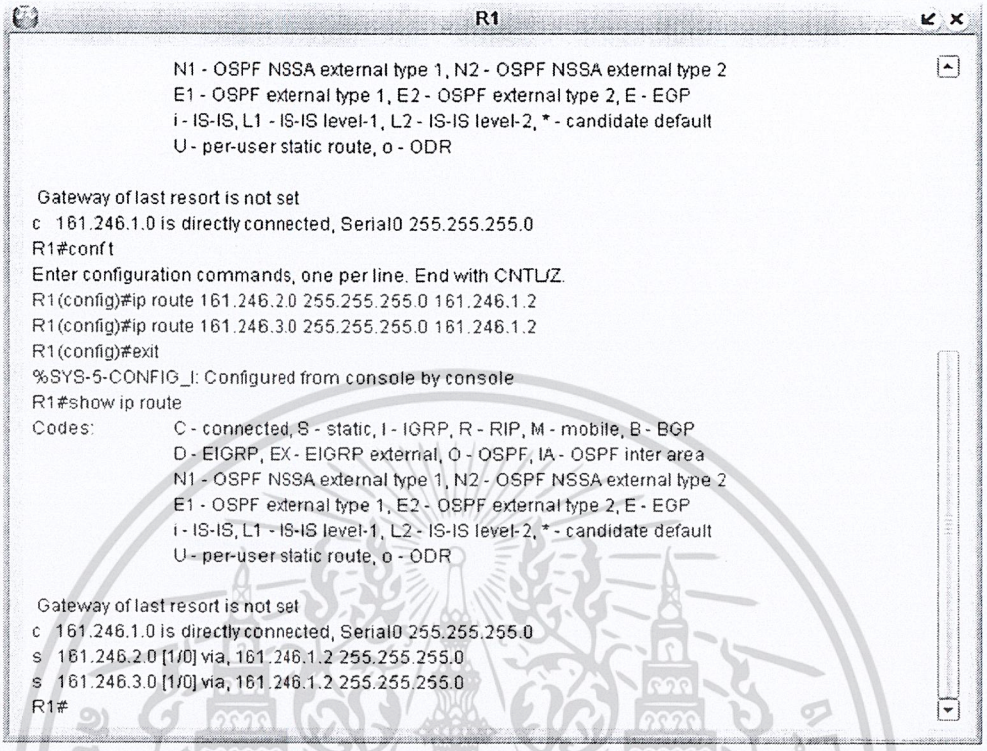
รูปที่ 9-4 การคอนฟิกการเลือกเส้นทางแบบสแตติกของเราเตอร์ R3



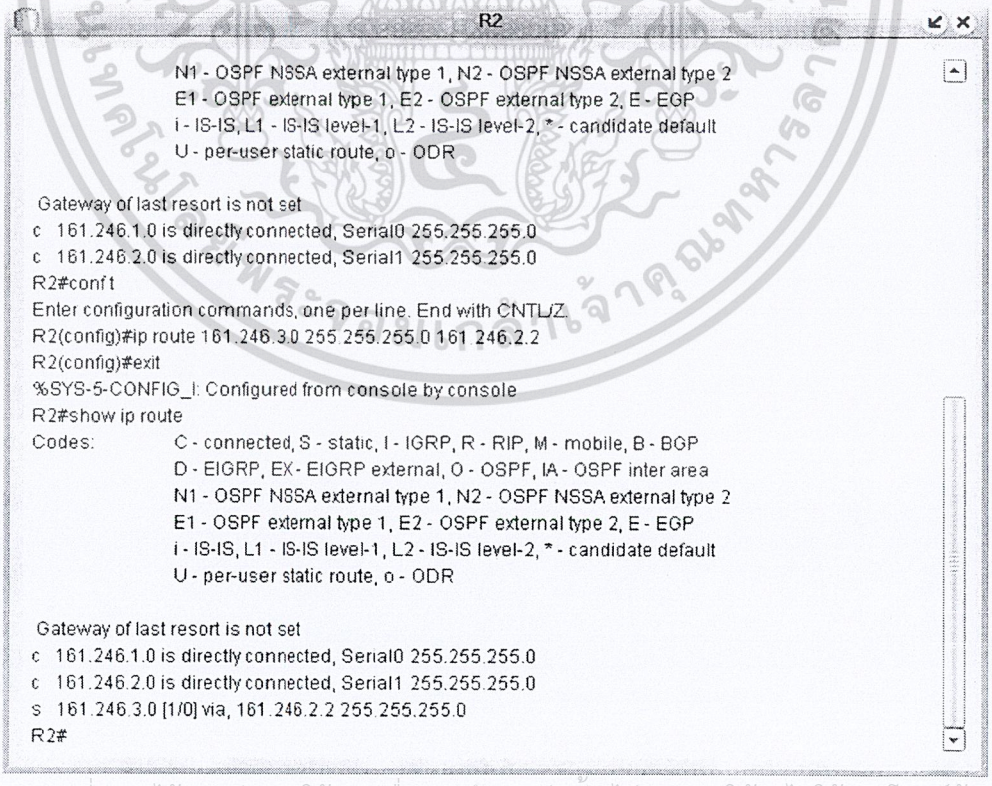
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 9-5 การคอนฟิกการเลือกเส้นทางแบบสแตติกของเราเตอร์ R4

3. สร้างการเลือกเส้นทางแบบแสตติกโดยใช้คำสั่ง ip route แล้วใช้คำสั่ง show ip route ดูค่าตาราง เลือกเส้นทางอีกที

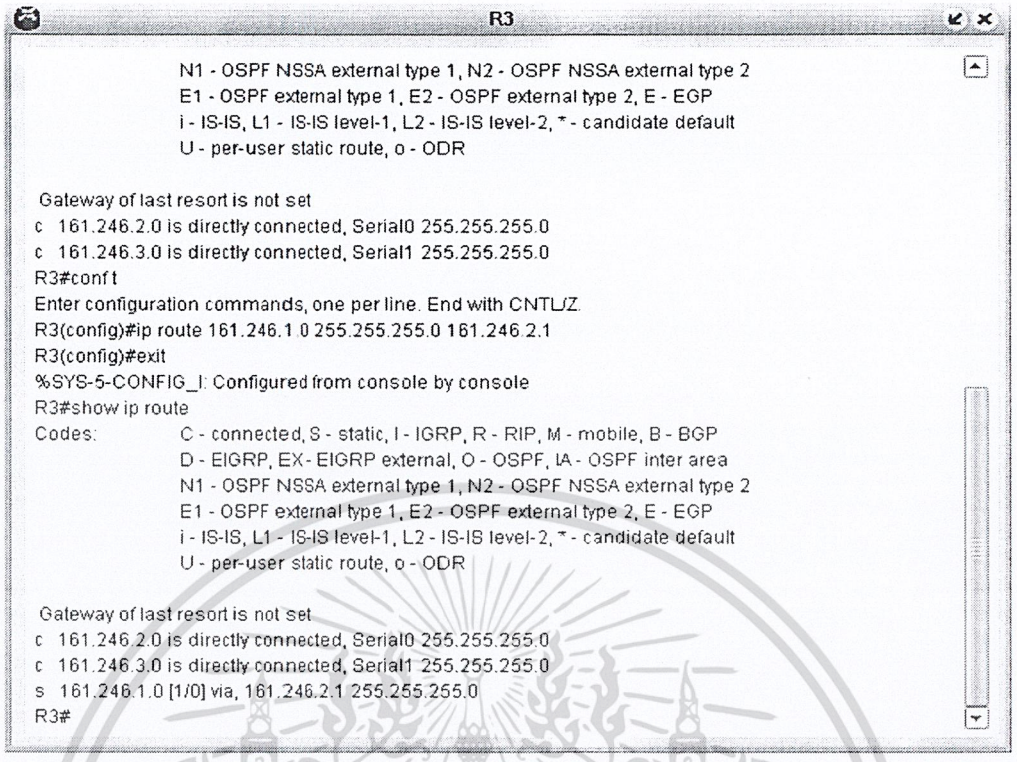


รูปที่ 9-6 การแสดงเส้นทางแบบสแตติกของเราเตอร์ R1

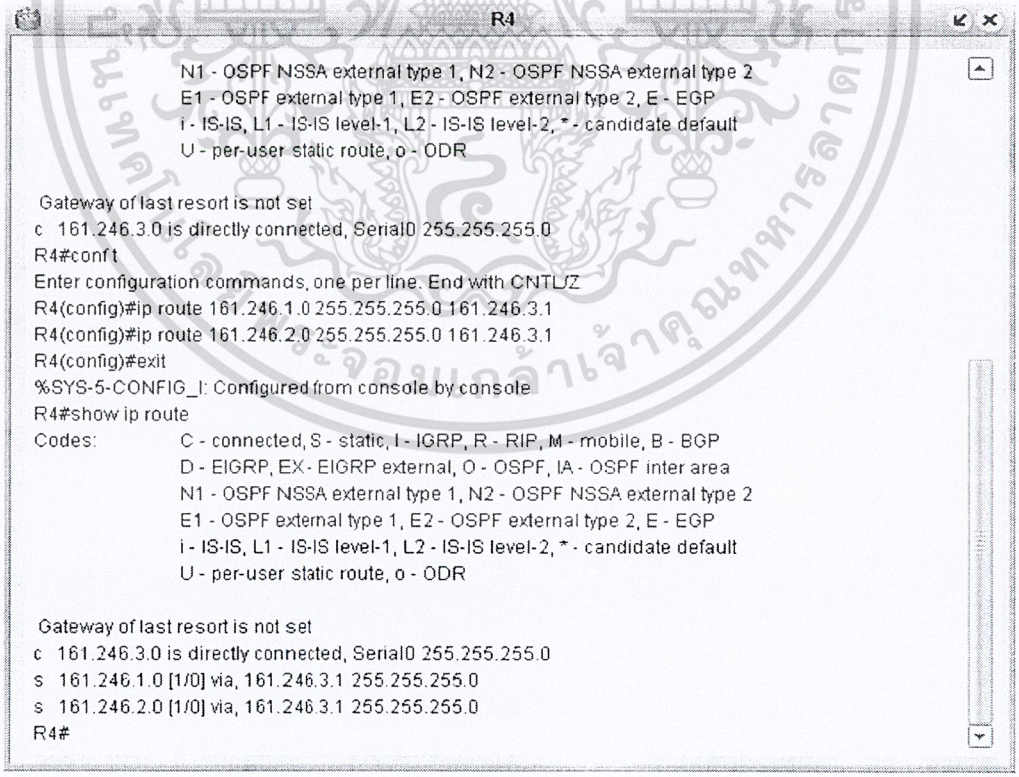


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 9-7 การแสดงเส้นทางแบบสแตติกของเราเตอร์ R2



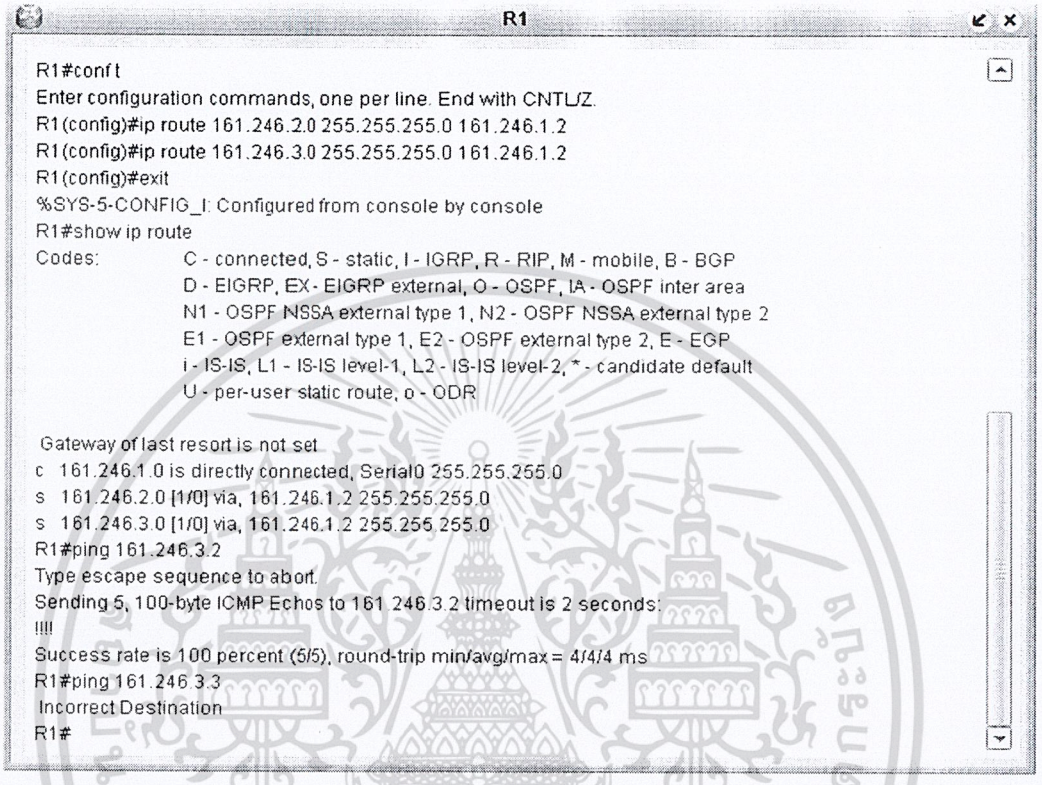
รูปที่ 9-8 การแสดงเส้นทางแบบสแตติกของเราเตอร์ R3



รูปที่ 9-9 การแสดงเส้นทางแบบสแตติกของเราเตอร์ R4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. ทดสอบโดยใช้คำสั่ง ping เพื่อตรวจสอบว่าไปถึงปลายทางได้หรือไม่ ในที่นี่จะใช้เราเตอร์ R1 ping ไปยังไอพี 161.246.3.2 ซึ่งเป็นอินเทอร์เฟสของเราเตอร์ R4 ซึ่งถ้าการเชื่อมต่อถูกต้อง จะต้องสามารถ ping ไปได้ และไอพี 161.246.3.3 ซึ่งไม่มีในตารางเลือกเส้นทางดังนี้

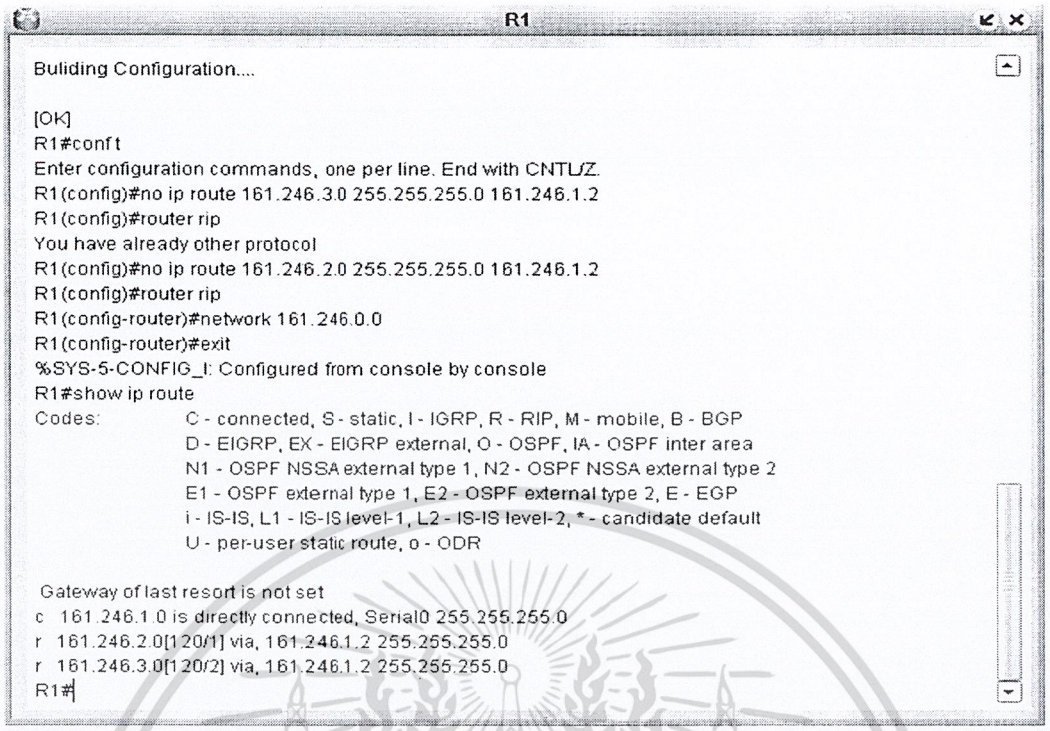


รูปที่ 9-10 ผลการใช้คำสั่ง ping

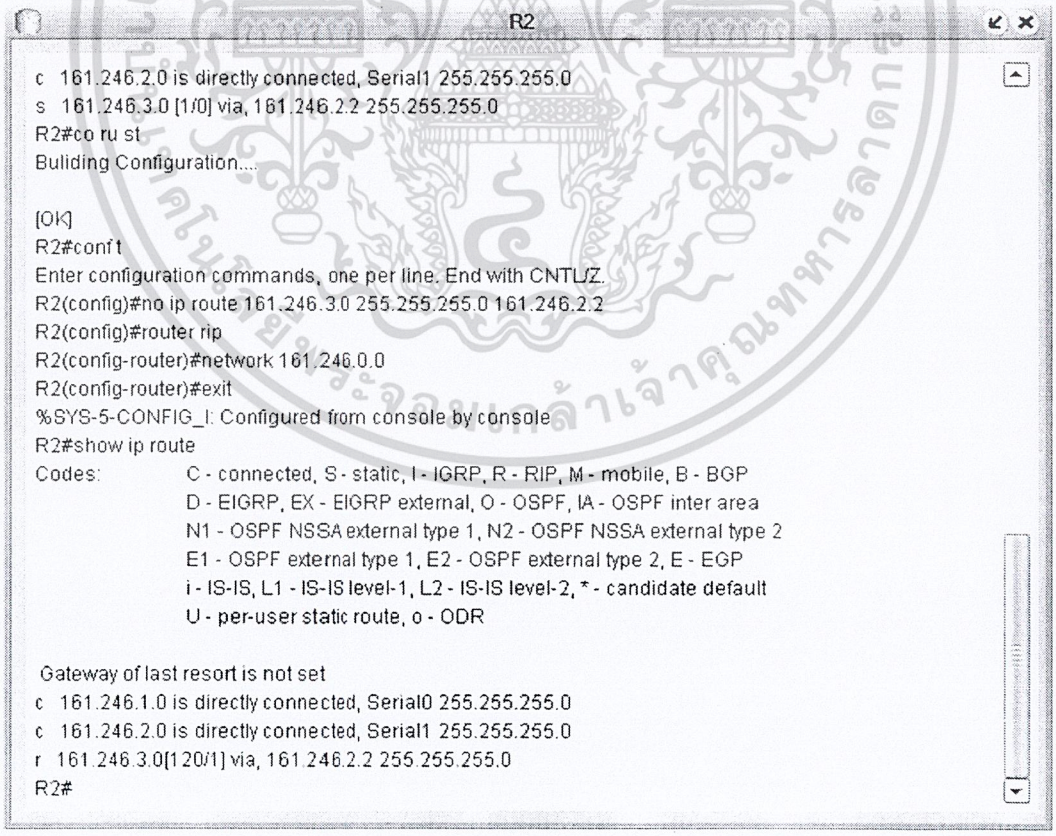
9.2 ตัวอย่างการทำงานของโพรโตคอลอาร์ไอพี

1. จากตัวอย่างที่แล้วเราจะใช้เครือข่ายเดิมในการจำลองการทำงานของเราเตอร์โดยใช้โพรโตคอลอาร์ไอพี โดยการยกเลิกโพรโตคอลแบบสแตติกทั้งหมดด้วยคำสั่ง no ip route หลังจากนั้นก็ทำการกำหนดโพรโตคอลอาร์ไอพีให้เราเตอร์โดยใช้คำสั่ง router rip โดยทำกับเราเตอร์ทุกตัว จากนั้นเมื่อลองใช้คำสั่ง show ip route ดูก็จะได้ผลลัพธ์ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

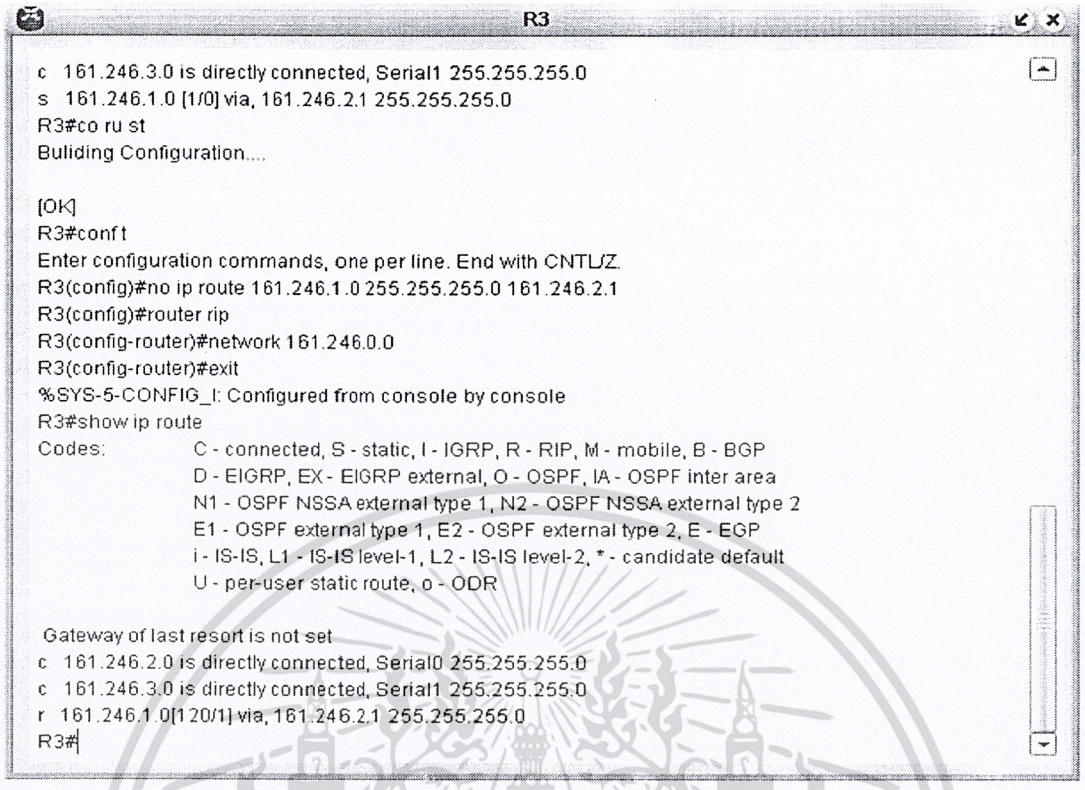


รูปที่ 9-11 การคอนฟิกโพรโตคอลอาร์ไอพีของเราเตอร์ R1

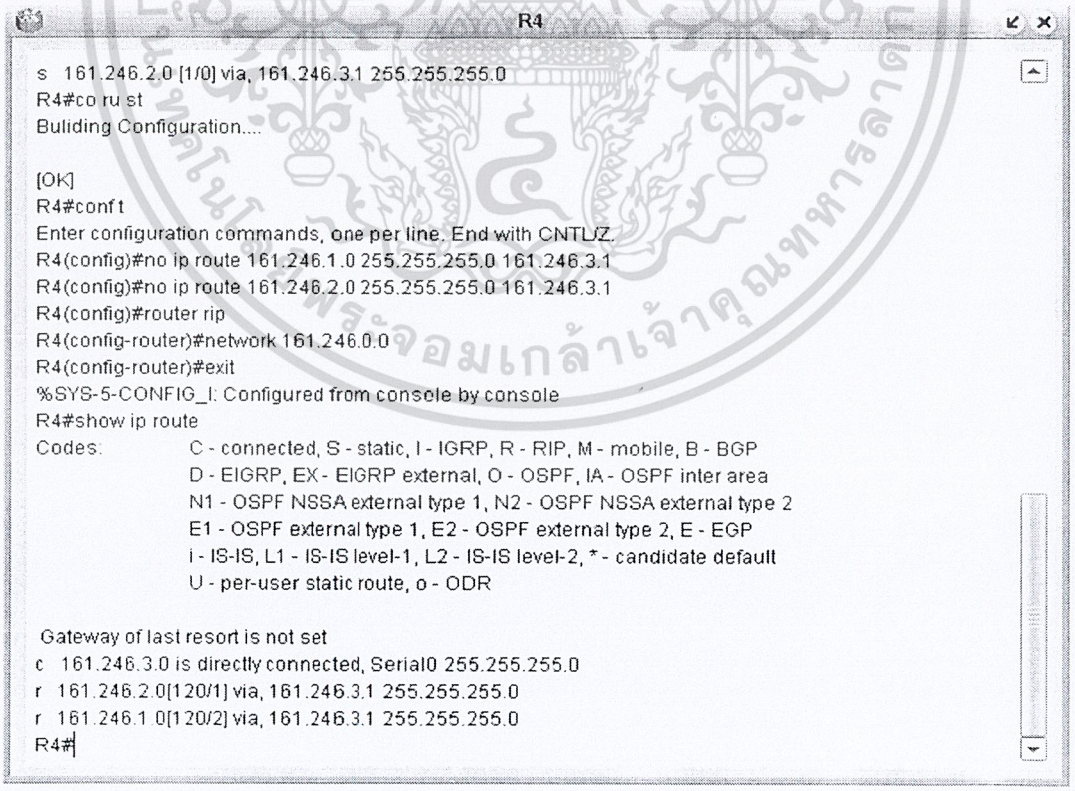


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 9-12 การคอนฟิกโพรโตคอลอาร์ไอพีของเราเตอร์ R2



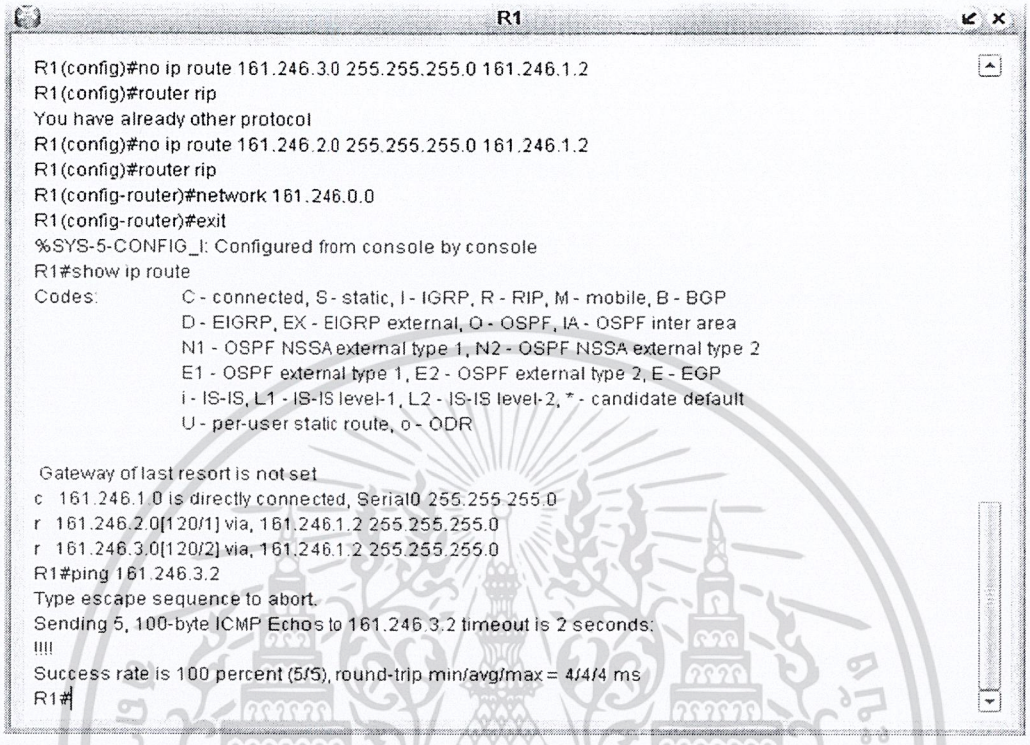
รูปที่ 9-13 การคอนฟิกโพรโตคอลอาร์ไอพีของเราเตอร์ R3



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 9-14 การคอนฟิกโพรโตคอลอาร์ไอพีของเราเตอร์ R4

2. ลองใช้คำสั่ง ping ตรวจสอบการเชื่อมต่อโดย ping จากเราเตอร์ R1 ไปยัง R4 ที่ไอพี 161.246.3.2 จะได้ผลลัพธ์ดังนี้

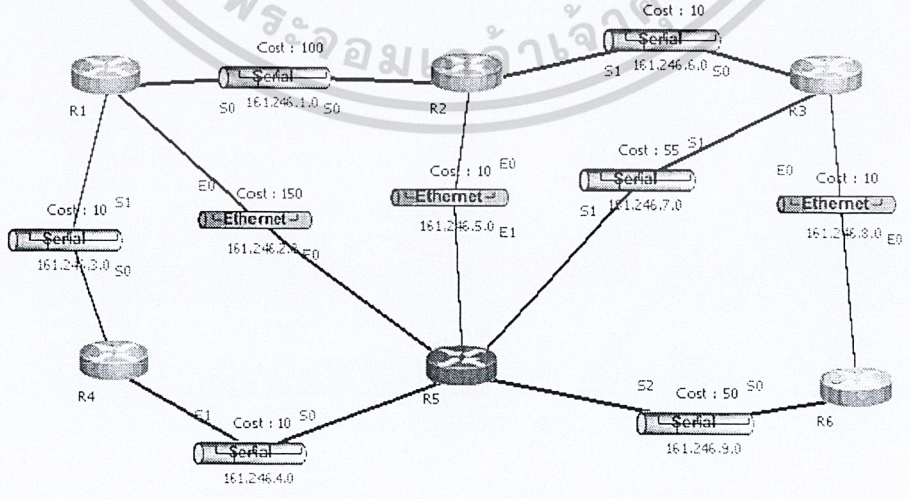


รูปที่ 9-15 การใช้คำสั่ง ping

9.3 ตัวอย่างการทำงานของโพรโตคอลโอเอสพีเอฟ

เป็นตัวอย่างการจำลองการทำงานของเราเตอร์ที่ใช้โพรโตคอลโอเอสพีเอฟ

1. สร้างเครือข่ายดังรูป



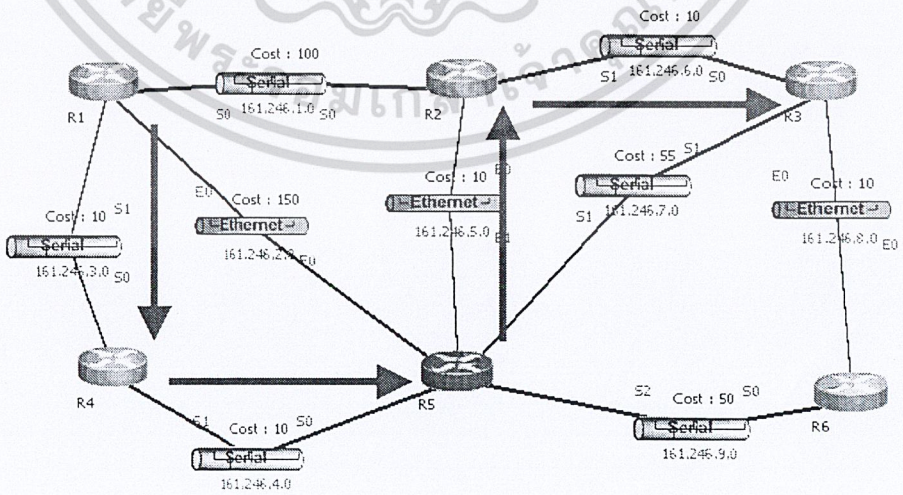
รูปที่ 9-16 รูปเครือข่ายโอเอสพีเอฟ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ทำการกำหนดค่าอินเตอร์เฟสตามรูป แล้วก็กำหนดโพรโตคอลให้กับเราเตอร์ทุกตัวเป็นแบบโอเอสพีเอฟด้วยคำสั่ง router ospf [id] โดย id ให้ใส่เป็นเลขอะไรก็ได้ซึ่งไม่มีผลต่อการทำงานของโปรแกรม เช่นที่เราเตอร์ R1 ก็ใส่เป็น router ospf 1 เป็นต้น หลังจากนั้นก็กำหนดค่าคอสท์ให้กับสายเชื่อมต่อระหว่างเราเตอร์ด้วยคำสั่ง ip ospf [cost] ซึ่งค่าคอสท์แสดงถึงระยะเวลาในการส่งข้อมูลผ่านเครือข่ายถ้าค่าคอสท์มากก็ต้องใช้เวลาในการส่งข้อมูลมาก ซึ่งโพรโตคอลโอเอสพีเอฟต้องสามารถหาเส้นทางที่คอสท์ต่ำที่สุดได้ ให้กำหนดค่าคอสท์ตามรูปเครือข่ายข้างบน
3. ใช้คำสั่ง traceroute เพื่อดูว่าเส้นทางจากเราเตอร์ R1 ไปยังไอพี 161.246.8.2 ที่เป็นอินเตอร์เฟส E0 ของเราเตอร์ R3 ว่าสามารถไปทางไหนได้บ้าง โดยถาดูจากรูปที่ 9-16 แล้วจะเห็นได้ว่าจากเราเตอร์ R1 ไปยังเราเตอร์ R3 มีด้วยกันทั้งหมด 9 เส้นทางได้แก่

เส้นทางที่1 1.0, 6.0, 8.0
เส้นทางที่2 1.0, 5.0, 9.0, 8.0
เส้นทางที่3 1.0, 5.0, 7.0, 8.0
เส้นทางที่4 2.0, 5.0, 6.0, 8.0
เส้นทางที่5 2.0, 7.0, 8.0
เส้นทางที่6 2.0, 9.0, 8.0
เส้นทางที่7 3.0, 4.0, 5.0, 6.0, 8.0
เส้นทางที่8 3.0, 4.0, 7.0, 8.0
เส้นทางที่9 3.0, 4.0, 9.0, 8.0

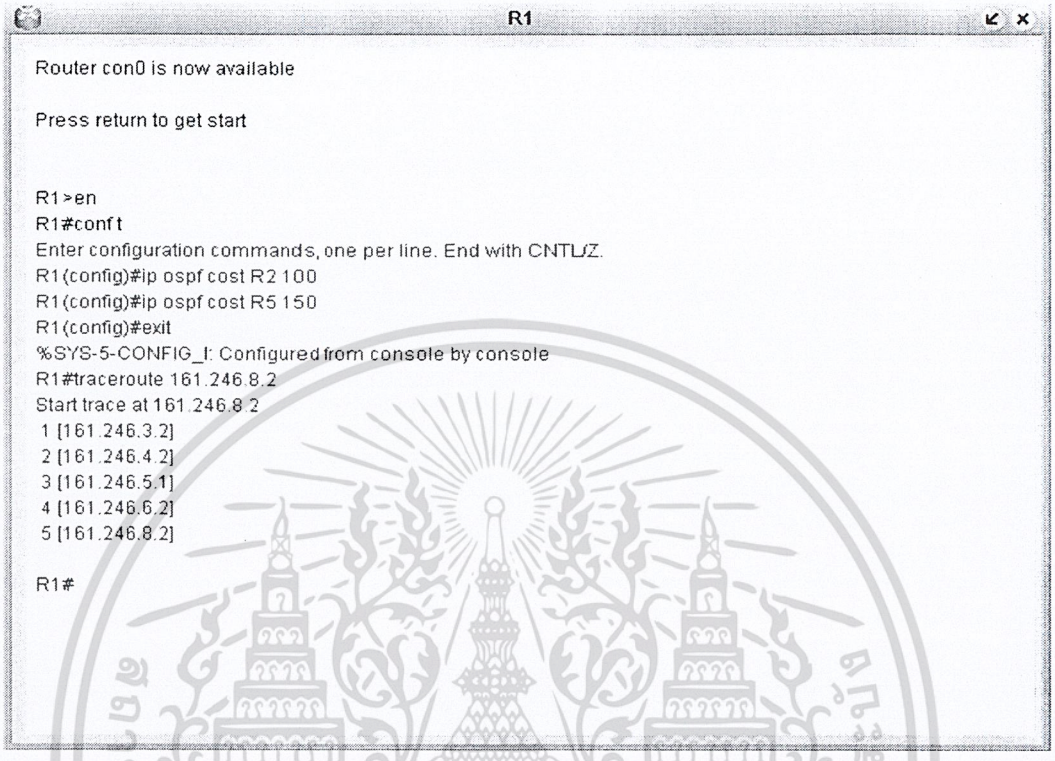
โพรโตคอลโอเอสพีเอฟจะเลือกเส้นทางที่ดีที่สุดโดยใช้อัลกอริทึมของไดจ์สตรา(Dijkstra Algorithm) (ดูรายละเอียดได้ในบทที่ 6) โดยคิดจากค่าคอสท์ที่ได้กำหนดให้กับสายเชื่อมต่อระหว่างเราเตอร์ R3 ดังรูปที่ 9-17 และผลจากคำสั่ง traceroute ดังรูปที่ 9-18



รูปที่ 9-16 รูปแสดงเส้นทางจากเราเตอร์ R1 ไปยังเราเตอร์ R3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในที่นี้หมายเลข 1.0 หมายถึง 161.246.1.0 และหมายเลขอื่นๆก็มีความหมายเช่นเดียวกันเช่น 2.0 ก็หมายถึง 161.246.2.0 เป็นต้น



รูปที่ 9-18 การใช้คำสั่ง traceroute

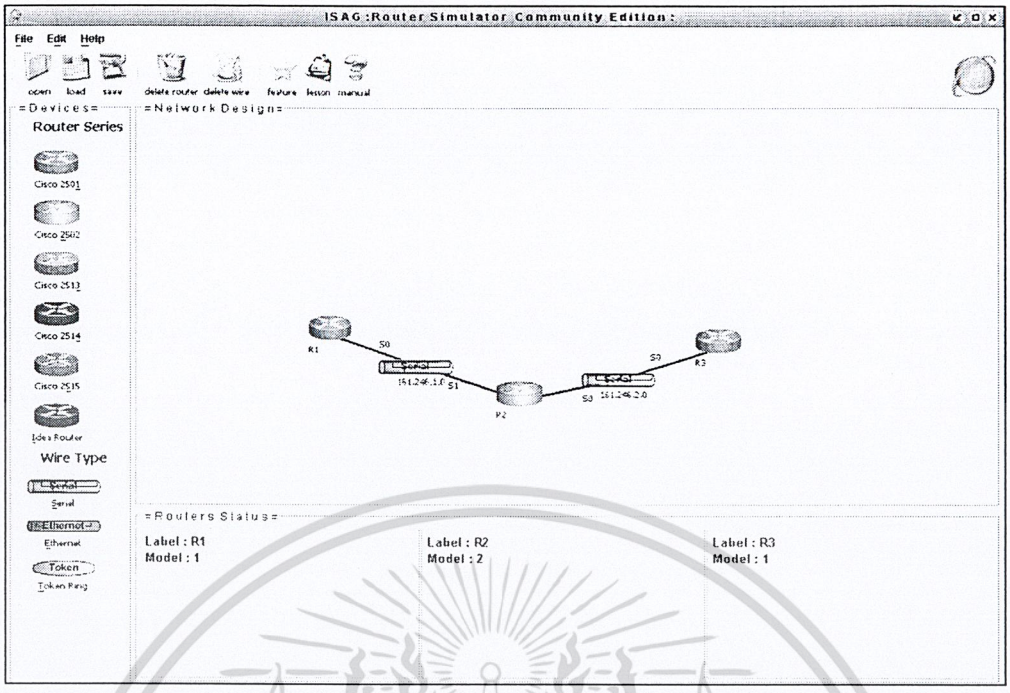
9.4 ตัวอย่างการทำงานแบบผู้ใช้หลายคน

ตัวอย่างนี้เป็นการใช้งานสำหรับผู้ใช้หลายคน โดยสมมติให้มีผู้ทำการทดลองอยู่ 3 คน ได้แก่ User1,User2 และ User3 โดยมีผู้ดูแลการทดลอง 1 คนคือ Admin1

1. ผู้ดูแลการทดลองสร้างเครือข่ายที่มีเราเตอร์อยู่ 3 ตัวและกำหนดพาสเวิร์ดให้เราเตอร์แต่ละตัว ดังนี้คือ
    - เราเตอร์ R1 พาสเวิร์ดคือ U1
    - เราเตอร์ R2 พาสเวิร์ดคือ U2
    - เราเตอร์ R3 พาสเวิร์ดคือ U3

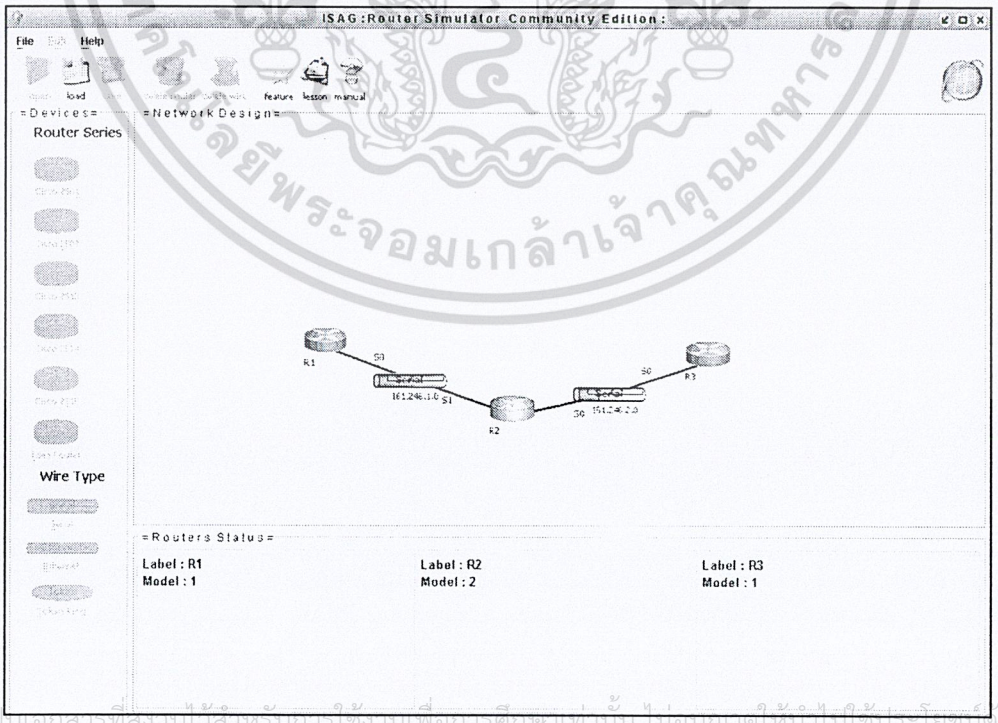
โดยกำหนดให้ User1 ทำการคอนฟิกเราเตอร์ R1, User2 ทำการคอนฟิกเราเตอร์ R2 และ User3 ทำการคอนฟิกเราเตอร์ R3 โดยมีข้อกำหนดว่าให้คอนฟิกเราเตอร์ให้ใช้โพรโตคอลอาร์ไอพี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



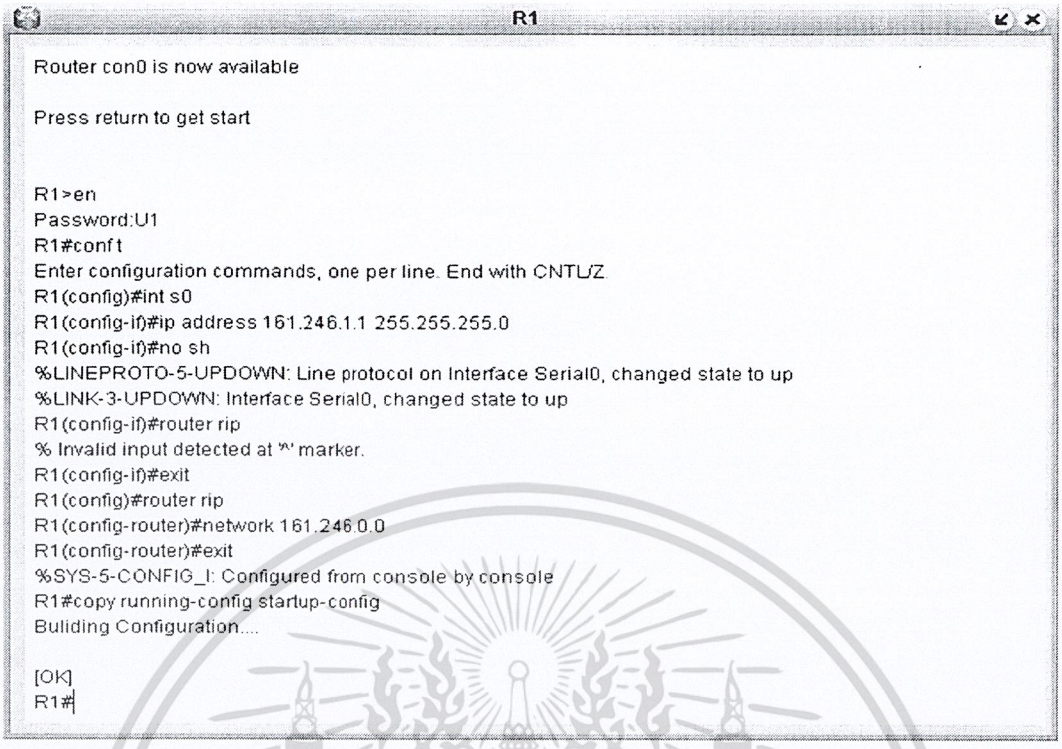
รูปที่ 9-19 รูปเครือข่ายที่ผู้ดูแลการทดลองสร้างขึ้น

2. ผู้ทำการทดลองแต่ละคนก็ทำการเปิดโปรแกรมขึ้นมาแล้วเลือกโหมด User จากนั้นก็กดปุ่มโหลดเพื่อโหลดไดอะแกรมที่ผู้ดูแลการทดลองสร้างไว้ แล้วทำการคอนฟิกเราเตอร์ตัวที่ได้รับมอบหมายจากผู้ดูแลการทดลอง จะได้ผลดังนี้

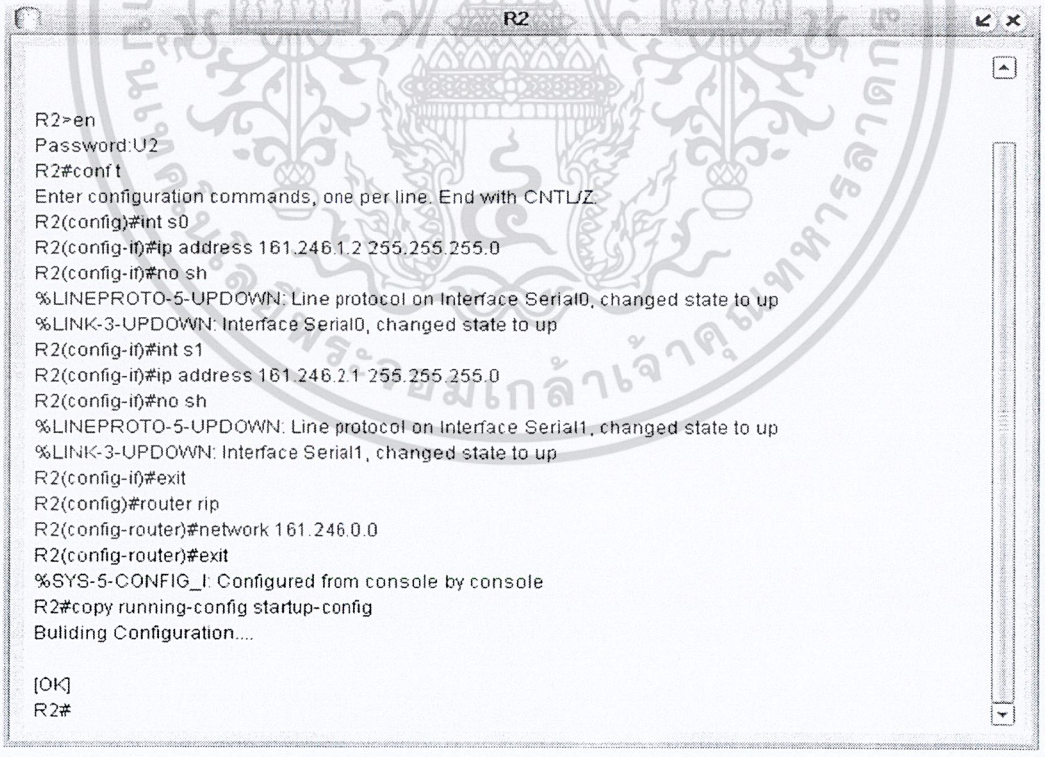


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 9-20 รูปเครือข่ายที่ผู้ทำการทดลองโหลดมาจากเซิร์ฟเวอร์

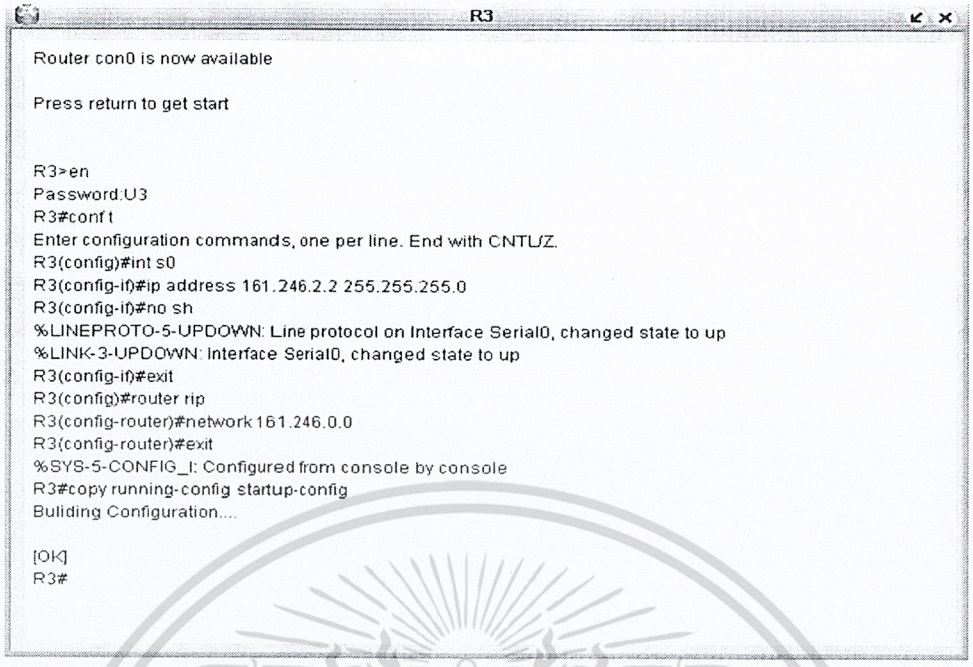


รูปที่ 9-21 แสดงผลการคอนฟิกเราเตอร์ของ User1



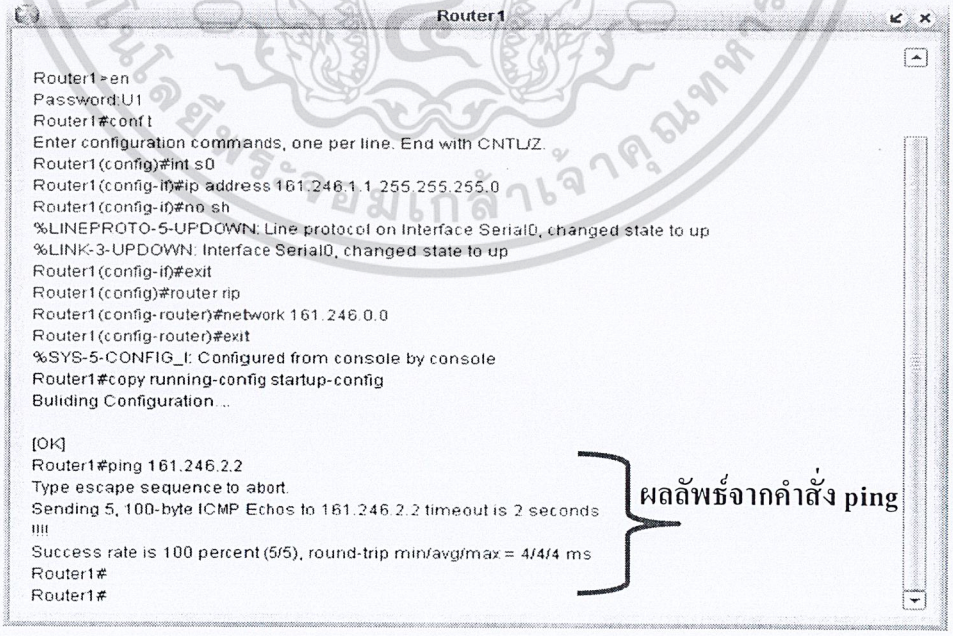
รูปที่ 9-22 แสดงผลการคอนฟิกเราเตอร์ของ User2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 9-23 แสดงผลการคอนฟิกเราเตอร์ของ User3

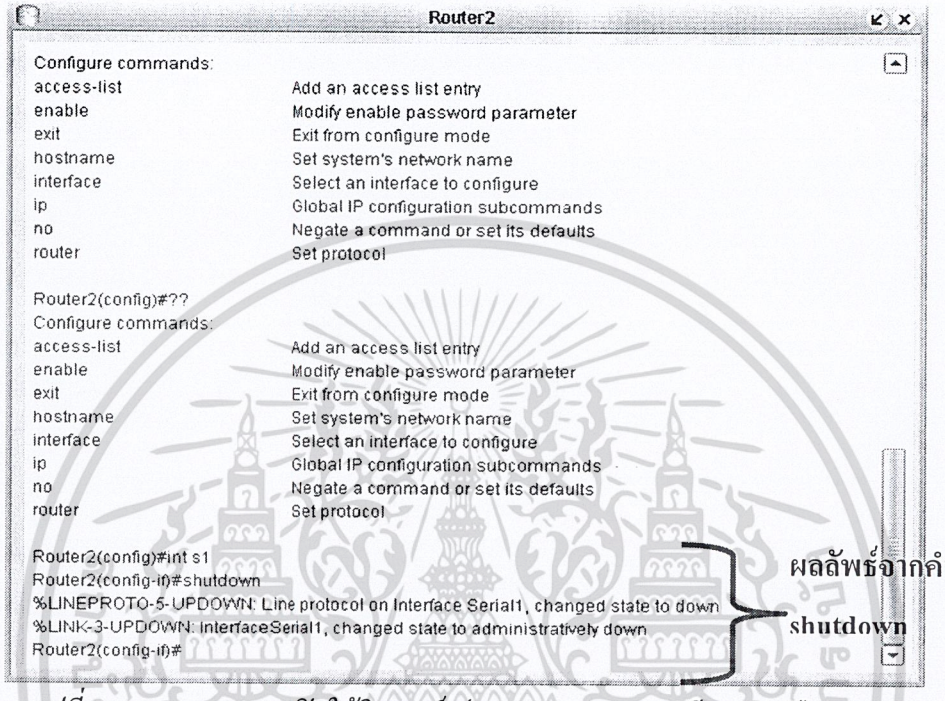
3. เมื่อ User ทั้ง 3 คนทำการคอนฟิกเราเตอร์ที่ได้รับมอบหมายได้อย่างถูกต้องตามรูปที่ 9-21, 9-22 และ 9-23 แล้วเมื่อใช้คำสั่ง ping ตรวจสอบการเชื่อมต่อระหว่างเครือข่ายวง 161.246.1.0 กับเครือข่ายวง 161.246.2.0 โดย ping จาก Router1 ไปยัง Router3 ผลลัพธ์ที่ได้คือ Router1 สามารถไปถึง Router3 ได้ ผลลัพธ์ที่ได้ดังรูปที่ 9-24



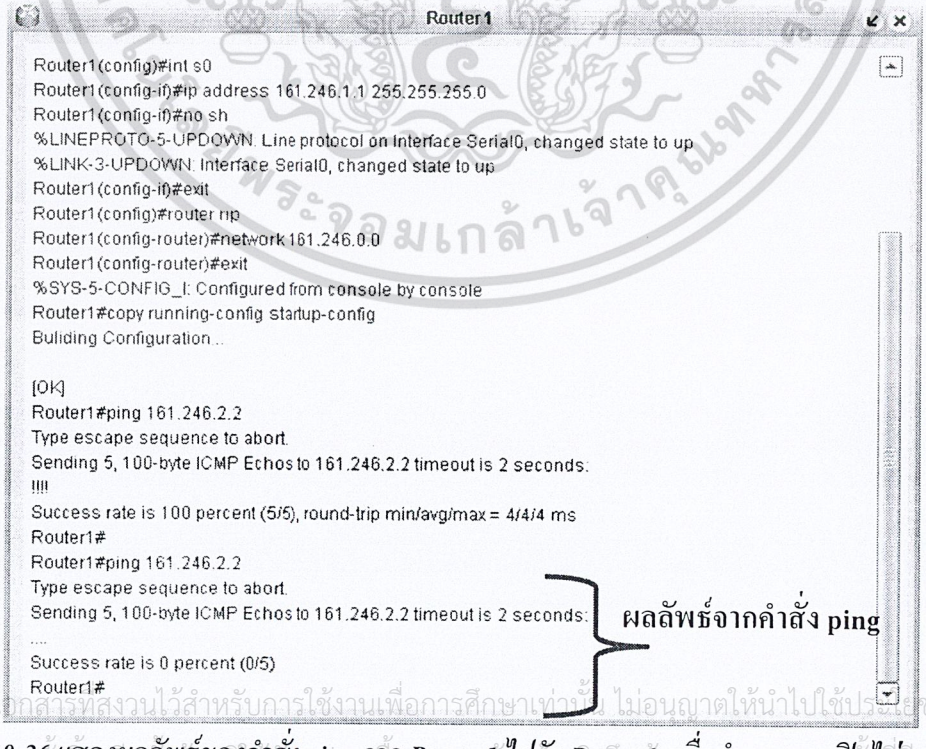
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 9-24 แสดงผลลัพธ์ของคำสั่ง ping จาก Router1 ไปยัง Router3 เมื่อทำการคอนฟิกได้ถูกต้อง

4. สมมติเมื่อ User2 ทำการคอนฟิกอินเตอร์เฟส S1 ของ Router2 ให้มีสถานะ down จะทำให้เครือข่ายวง 161.246.1.0 และเครือข่ายวง 161.246.2.0 ไม่สามารถติดต่อสื่อสารกันได้ ผลการคอนฟิกดังรูปที่ 9-25 ดังนั้นเมื่อใช้คำสั่ง ping จาก Router1 ไปยัง Router3 หรือ Router3 ไปยัง Router1 ผลลัพธ์จากการ ping ไม่สามารถไปถึงปลายทางได้ดังรูปที่ 9-26



รูปที่ 9-25 แสดงการคอนฟิกให้อินเตอร์เฟส S1 ของ Router2 มีสถานะเป็น Down



รูปที่ 9-26 แสดงผลลัพธ์ของคำสั่ง ping จาก Router1 ไปยัง Router3 เมื่อทำการคอนฟิกไม่ถูกต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารนี้ทุกครั้งที่มีการนำไปใช้

บทที่ 10
สรุปและวิจารณ์

ในการทำงานของปริญญานิพนธ์นี้ สามารถที่จะสร้างซอฟต์แวร์จำลองการตั้งค่าเราเตอร์ของซิสโก้ และจำลองการทำงานของโพรโตคอลที่ใช้เลือกเส้นทางได้ ทั้งแบบแสตติกและไดนามิก (อาร์ไอพีและโอเอสพีเอฟ) ทำให้สามารถจำลองการทำงานของเครือข่ายได้ตั้งแต่เครือข่ายขาดเล็กไปจนถึงเครือข่ายขนาดใหญ่ สามารถที่จะบันทึกค่าคอนฟิกและรูปไดอะแกรมของเครือข่ายที่สร้างไว้ลงไฟล์เก็บไว้ได้ ผู้ใช้สามารถออกแบบสร้างเครือข่ายเองได้เพื่อให้สามารถมองภาพรวมของเครือข่ายได้อย่างชัดเจน อีกทั้งโปรแกรมสามารถทำงานแบบผู้ใช้หลายคนได้(Multiuser) พร้อมทั้งมีบทเรียนสอนเพื่อที่ผู้สนใจใช้ซอฟต์แวร์นี้สามารถทดลองใช้ตามบทเรียน

การตั้งค่าอุปกรณ์เราเตอร์มีคำสั่งมากและเราเตอร์มีการออกซีรี่ใหม่มาเรื่อยๆ การที่ผู้ดูแลระบบจะดูแลและใช้ความสามารถของเราเตอร์ได้อย่างเต็มที่นั้น ผู้ดูแลจะต้องทราบถึงความสามารถในการตั้งค่าของเราเตอร์ ต้องทราบว่าเราเตอร์มีคำสั่งอะไรบ้าง แต่ถึงอย่างไรก็ตามหลักการพื้นฐานในการตั้งค่าอุปกรณ์เราเตอร์รวมทั้งคำสั่งพื้นฐานก็ไม่แตกต่างกันมากนัก การที่เราเข้าใจหลักการพื้นฐานของการตั้งค่าอุปกรณ์เราเตอร์จึงเป็นสิ่งที่จำเป็น

10.1 ปัญหาและอุปสรรค

- คำสั่งในการตั้งค่าเราเตอร์มีเยอะมาก ต้องใช้เวลาในการศึกษาและรวบรวมเฉพาะคำสั่งที่น่าสนใจ และเป็นประโยชน์
- ใช้เวลาในการศึกษา RMI นานพอสมควร

10.2 ขอบเขตและข้อจำกัดของโครงงาน

- จำลองการทำงานรูปแบบการตั้งค่าเราเตอร์ของเราเตอร์ซิสโก้ รุ่น 2500
- ใช้โพรโตคอลในการค้นหาเส้นทางแบบสแตติกและแบบไดนามิก ประกอบด้วยอาร์ไอพีและโอเอสพีเอฟเท่านั้น
- ไม่สามารถใช้ได้ทุกคำสั่งที่มีในเราเตอร์ของ ซิสโก้รุ่น 2500
- สำหรับการใช้งานแบบผู้ใช้หลายคนเราเตอร์หนึ่งตัวไม่สามารถตั้งค่าพร้อมกันในเวลาเดียวกันได้
- สำหรับการทำงานแบบผู้ใช้หลายคน การติดต่อระหว่างไคลเอ็นต์และเซิร์ฟเวอร์ไม่สามารถทำงานผ่านไฟร์วอลล์ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

10.3 แนวทางการประยุกต์และพัฒนา

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

10.4 สรุปโครงงาน

โครงงานนี้มีประโยชน์มาก เพราะสามารถให้ผู้ที่ต้องการฝึกการตั้งค่าเราเตอร์และเรียนรู้การทำงานของโพรดตคอลที่เราเตอร์ใช้งาน ไม่ว่าจะเป็นผู้ที่อยู่ในระดับการศึกษา หรือใช้งานจริงก็ตาม สามารถฝึกฝนด้วยตนเองได้ โดยมีตัวอย่างการทดลองเบื้องต้นให้ฝึกฝน ซึ่งโครงงานนี้นอกจากสามารถทำเพื่อการศึกษาด้านเครือข่ายคอมพิวเตอร์แล้ว ยังเป็นการลดความต้องการ การสั่งซื้ออุปกรณ์เราเตอร์หรือซอฟต์แวร์เชิงพาณิชย์ที่จำลองการตั้งค่าเราเตอร์จากต่างประเทศ โครงงานนี้เป็นโปรแกรมที่ทำงานได้ทั้งแบบผู้ใช้คนเดียวและผู้ใช้หลายคนอีกทั้งเป็นโปรแกรมแบบมัลติแพลตฟอร์ม (Multi platform) เท่าที่สำรวจมายังไม่มีซอฟต์แวร์เชิงพาณิชย์ตัวไหนที่สามารถทำได้ และไม่มีซอฟต์แวร์เชิงพาณิชย์ตัวไหนที่สามารถใช้โพรโตคอลการเลือกเส้นทางแบบโอเอสพีเอฟได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก
สรุปคำสั่งทั้งหมด

Bandwidth

Command Mode Interface Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดค่าแบนวิดให้กับอินเทอร์เฟสประเภท Serial

รูปแบบคำสั่ง Bandwidth <number>

<number> เป็นค่าขนาดของแบนวิด มีหน่วยเป็นกิโลบิต

ตัวอย่างการใช้งาน

RouterA(config)#**int s0**

RouterA(config-if)#**bandwidth 56**

RouterA(config-if)#

Cdp Timer

Command Mode 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

Command Mode Interface Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดค่า clock rate ให้กับอินเทอร์เฟสประเภท Serial

รูปแบบคำสั่ง Clock Rate <number>

<number> เป็นค่าขนาดของ clock rate มีหน่วยเป็นบิต

ตัวอย่างการใช้งาน

RouterA(config)#**int s0**

RouterA(config-if)#**clock rate 56000**

RouterA(config-if)#

Clock Set

Command Mode Privileged EXEC Commands

เอกสารการค้า
ไม่ว่าและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้เปลี่ยนโหมดจาก privilege mode เป็น global configuration mode

ตัวอย่างการใช้งาน

Router#**configuration terminal**

Router(config)#

Copy Running-Config Startup-Config

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับจัดเก็บค่าที่กำหนดให้กับเราเตอร์ในขณะนั้นลงในหน่วยความจำเพื่อเวลาเปิดโปรแกรม

ขึ้นมาใหม่จะได้ไม่ต้องกำหนดค่าใหม่

ตัวอย่างการใช้งาน

RouterA#**copy running-config startup-config**

RouterA#

Copy Startup-Config Running-Config

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับโหลดค่าที่เก็บอยู่ในหน่วยความจำมากำหนดให้กับเราเตอร์

ตัวอย่างการใช้งาน

RouterA#**copy startup-config running-config**

Buliding Configuration

....

[OK]

RouterA#

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Debug ip rip

Command Mode Exex Commands

เป็นคำสั่งที่ใช้สำหรับแสดงการรับและส่งข้อมูลตารางเส้นทางของเราเตอร์

รูปแบบคำสั่ง Debug ip rip

ตัวอย่างการใช้งาน

RouterA#**debug ip rip**

RIP protocol debugging is on

RouterA#

Undebug ip rip

Command Mode Exex Commands

เป็นคำสั่งที่ใช้สำหรับยกเลิกการแสดงการรับและส่งข้อมูลตารางเส้นทางของเราเตอร์

รูปแบบคำสั่ง Undebug ip rip

ตัวอย่างการใช้งาน

RouterA#**undebug ip rip**

RIP protocol debugging is off

RouterA#

Description

Command Mode Interface Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดรายละเอียดเพิ่มเติมให้กับอินเทอร์เฟส

รูปแบบคำสั่ง Description <String>

<String> เป็นข้อความที่เป็นรายละเอียดเพิ่มเติมของอินเทอร์เฟส

ตัวอย่างการใช้งาน

RouterA(config)#**int e0**

RouterA(config-if)#**description lan link to ISAG room**

RouterA(config-if)#

Disable

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับเปลี่ยนโหมดจาก privilege โหมดเป็น user execution โหมด

ตัวอย่างการใช้งาน

Router#**disable**

Router>

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น ใช้

Enable

Command Mode User EXEC Commands

เป็นคำสั่งที่ใช้สำหรับเปลี่ยนโหมดจาก user execution เป็น privilege โหมด

ตัวอย่างการใช้งาน

Router>**enable**

Router#

Enable Secret

Command Mode Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดให้ใส่พาสเวิร์ดก่อนเข้าสู่ระดับของ Privileged EXEC

รูปแบบคำสั่ง Enable Secret <Password>

<Password> เป็นคีย์เวิร์ดที่เป็นพาสเวิร์ดในการผ่าน

ตัวอย่างการใช้งาน

RouterA(config)#**enable secret cisco**

RouterA(config)#

Erase Startup-Config

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับลบค่าที่กำหนดให้กับเราเตอร์ในหน่วยความจำ

ตัวอย่างการใช้งาน

RouterA#**copy startup-config running-config**

RouterA#

Exit

Command Mode User EXEC Commands

เป็นคำสั่งที่ใช้สำหรับ log out ออกจากเราเตอร์

ตัวอย่างการใช้งาน

Router>**exit**

Exit

Command Mode Global Configuration Commands, Interface Commands, Routing Engine Commands, Line Commands

เป็นคำสั่งใช้สำหรับออกจากโหมดที่กำลังใช้งานอยู่ขณะนั้น เพื่อเข้าสู่ Privileged EXEC Commands โหมด

ตัวอย่างการใช้งาน

Router(**Config-if)#exit**

Router#

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Hostname

Command Mode Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดชื่อเราเตอร์

ตัวอย่างการใช้งาน

Router(config)#**hostname RouterA**

RouterA(config)#

Interface

Command Mode Global Configuration Commands

เป็นคำสั่งเพื่อเข้าสู่การตั้งค่าอินเทอร์เฟสต่างๆ ของเราเตอร์โดยคำสั่งนี้จะตามด้วยชื่อของอินเทอร์เฟสที่ต้องการตั้งค่า

รูปแบบคำสั่ง Int <interface_name>

โดย <Interface_name> เป็นชื่อของอินเทอร์เฟส

ตัวอย่างการใช้งาน

Router(config)#**int e0**

Router(config-if)#

Ip address

Command Mode Interface Commands

เป็นคำสั่งที่ใช้กำหนดค่าไอพีแอดเดรสให้กับอินเทอร์เฟสที่กำลัง config อยู่

รูปแบบคำสั่ง ip address <ip_address> <netmask>

<Ip_address> เป็นค่าไอพีแอดเดรสที่ต้องการกำหนดให้กับอินเทอร์เฟส

<Netmask> เป็นค่าเน็ตเวิร์คมาสค์ของไอพีแอดเดรส

ตัวอย่างการใช้งาน

Router(config)#**int e0**

Router(config-if)#**ip address 161.246.5.1 255.255.255.0**

Router(config-if)#

Ip Ospf Cost

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดค่า weight ให้กับเราเตอร์ตัวที่เชื่อมต่อกันอยู่

รูปแบบคำสั่ง Ip Ospf Cost <Router_Name> <Weight>

<Router_Name> เป็นชื่อเราเตอร์ตัวติดกันที่ต้องการใส่ค่า weight

<Weight> เป็นค่า weight ระหว่างเราเตอร์

ตัวอย่างการใช้งาน

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น และห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

RouterA#**ip ospf cost Router2 10**

RouterA#

Ip Route/No Ip Route

Command Mode Global Configuration Commands

เป็นคำสั่งเพื่อกำหนดการค้นหาเส้นทางเป็นแบบสแตติกหรือยกเลิกข้อมูลการค้นหาเส้นทางแบบสแตติก

รูปแบบคำสั่ง Ip Route/No Ip Route <Destination> <Netmask> < Gateway>

<Destination> เป็นเลขเครือข่ายของเครือข่ายปลายทาง

<Netmask> เป็นค่าเน็ตเวิร์คมาสค์ของไอพีแอดเดรส

<Gateway> เป็นค่าไอพีแอดเดรสของเราเตอร์เครื่องถัดไป

ตัวอย่างการใช้งาน

Router(config)#**ip route 161.246.10.1 255.255.255.0 161.246.5.2**

Router(config)#

Line Auxiliary 0

Command Mode Global Configuration Commands

เป็นคำสั่งที่ใช้ระบุว่าค่าที่กำหนดต่อไปนี้เป็นการกำหนดที่ auxiliary port

ตัวอย่างการใช้งาน

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

RouterA(config-line)#

Line Console 0

Command Mode Global Configuration Commands

เป็นคำสั่งที่ใช้ระบุว่าค่าที่กำหนดต่อไปนี้เป็นการกำหนดที่ console port

ตัวอย่างการใช้งาน

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

RouterA(config-line)#

Line vty 0 4

Command Mode Global Configuration Commands

เป็นคำสั่งที่ใช้ระบุว่าค่าที่กำหนดต่อไปนี้เป็นการกำหนดที่ port telnet

ตัวอย่างการใช้งาน

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

RouterA(config-line)#

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น ไม่อนุญาตให้นำไปใช้

Login/No Login

Command Mode Line Commands

เป็นคำสั่งที่ใช้สำหรับเปลี่ยนกำหนดให้เปิดพอร์ตเทลเน็ตสำหรับ Login หรือปิดพอร์ตเทลเน็ตสำหรับ No Login

ตัวอย่างการใช้งาน

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

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

RouterA(config-line)#

Network/No Network

Command Mode Router Configuration Commands

เป็นคำสั่งที่ใช้สำหรับกำหนด network address ให้กับเครือข่ายที่ทำการคอนฟิกหรือยกเลิก network address

ตัวอย่างการใช้งาน

RouterA(config-router)#**network 161.246.0.0**

RouterA(config-router)#

RouterA(config-router)#**no network 161.246.0.0**

No Enable Secret

Command Mode Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับยกเลิกการใส่พาสเวิร์ดก่อนเข้าสู่ระดับของ Privileged EXEC

ตัวอย่างการใช้งาน

RouterA(config)#**no enable secret**

RouterA(config)#

Passive-interface/No Passive-interface

Command Mode Router Configuration Commands

เป็นคำสั่งที่ใช้สำหรับหยุดหรือไม่หยุดการ update routingtable บนอินเตอร์เฟส

รูปแบบคำสั่ง Passive-interface/No passive-interface <Interface_name>

<Interface_name> เป็นชื่อของอินเตอร์เฟส

ตัวอย่างการใช้งาน

RouterA(config-router)#**passive-interface s0**

RouterA(config-router)#

Password

Command Mode Line Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดค่าพาสเวิร์ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปแบบคำสั่ง Password <Password>

<Password> เป็นคีย์เวิร์ดที่ใช้สำหรับผ่าน

ตัวอย่างการใช้งาน

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

RouterA(config-line)#**password 123**

RouterA(config-line)#

Ping

Command Mode 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

Command Mode Interface Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดค่า Ring Speed ให้กับอินเทอร์เฟสประเภทโทเคนริงค์

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

Command Mode Routing Engine Commands

เป็นคำสั่งเพื่อกำหนดให้เราเตอร์เครื่องนั้นใช้โพรโตคอลอาร์ไอพีในการสร้างตารางค้นหาเส้นทาง

สำหรับคำสั่ง router rip และยกเลิกการใช้โพรโตคอลอาร์ไอพีสำหรับคำสั่ง no router rip

ตัวอย่างการใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

RouterA(config)#**router rip**

RouterA(config-router)#

Router Ospf/No Router Ospf

Command Mode Routing Engine Commands

เป็นคำสั่งเพื่อกำหนดให้เราเตอร์เครื่องนั้นใช้โพรโตคอลโอเอสพีเอฟในการสร้างตารางค้นหาเส้นทาง สำหรับคำสั่ง router ospf และยกเลิกการใช้โพรโตคอลโอเอสพีเอฟสำหรับคำสั่ง no router ospf

ตัวอย่างการใช้งาน

RouterA(config)#**router spf**

RouterA(config-router)#

Show Cdp

Command Mode 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

Command Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดง 10 คำสั่งล่าสุดที่เคยใช้งาน

ตัวอย่างการใช้งาน

RouterA#**show history**

//----this is history commands---//

RouterA#

Show Interface

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงรายละเอียดของอินเทอร์เฟส

รูปแบบคำสั่ง Show Interface <interface name>

<interface name> เป็นตัวเลือกจะใส่หรือไม่ใส่ก็ได้ ถ้าใส่จะเป็นการระบุให้แสดงรายละเอียดเฉพาะอินเทอร์เฟสที่ระบุ ถ้าไม่เช่นนั้นจะแสดงทุกอินเทอร์เฟส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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 minu te 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 Ospf database

Command Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดง databaseของตารางเส้นทางโอเอสพีเอฟ ของเราเตอร์

ตัวอย่างการใช้งาน

RouterA#**show ip ospf database**

161.246.5.0 is directly connected, serial0

161.246.4.0 is directly connected, serial1

161.246.3.0 is directly connected, ethernet0

RouterA#

เอกสารนี้มีไว้เพื่อเป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Show Ip Ospf interface

Command Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดง databaseของตารางเส้นทางโอเอสพีเอฟ ของเราเตอร์

รูปแบบคำสั่ง**Show Ip Ospf interface<interface_name>**

<interface_name> เป็นชื่อของอินเตอร์เฟส

ตัวอย่างการใช้งาน

RouterA#**show ip ospf interface s0**

s0 is Up , line protocol is up

Internet Address 161.246.4.2, Area 0

Process ID 1, Router ID RouterA, Network Type BROADCAST, Cost: 64

Transmit Delay is 1 sec, State DR, Priority 1

Designated Router (ID) RouterA, Interface address161.246.4.2

Backup Designated router (ID) RouterB, Interface address 161.246.4.1

Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5

Hello due in 00:00:00

Neighbor Count is 1, Adjacent neighbor count is 1

Adjacent with neighbor RouterB(Backup Designated Router)

Suppress hello for 0 neighbor(s)

RouterA#

Show Ip Ospf Neighbors

Command Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงข้อมูลของเราเตอร์ข้างเคียง

ตัวอย่างการใช้งาน

RouterA#**show ip ospf neighbors**

Neighbor	ID	Pri	State	Date Time	Address	Interface
R1	1	FULL/DR		0.00.34	161.246.4.2	s0

RouterA#

Show Ip Protocol

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงโพรโตคอลที่เราเตอร์ใช้งานอยู่ขณะนั้น

ตัวอย่างการใช้งาน

เอกสารนี้เป็นลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย ที่ได้อนุญาตให้นำไปใช้ประโยชน์ด้านการศึกษาเท่านั้น
ไม่ว่ากรณีใดๆ ทั้งสิ้น ห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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 Rip Database

Command Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดง databaseของตารางเส้นทางอาร์ไอพี ของเราเตอร์

ตัวอย่างการใช้งาน

RouterA#**show ip rip database**

161.246.5.0 is directly connected, serial0

161.246.4.0

[1] via 161.246.5.16, serial0

RouterA#

how Ip Route

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงตางรางการค้นหาเส้นทาง

รูปแบบคำสั่ง Show Ip Route <route_type>

<route_type> เป็นตัวเลือกที่จะใส่หรือไม่ใส่ก็ได้ ถ้าใส่จะเป็นการระบุว่าต้องการดูเฉพาะตารางค้นหาเส้นทางประเภทใด โดย

s : สแตติก

r : อาร์ไอพี

o : โอเอสพีเอฟ

ตัวอย่างการใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

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 Ip Route Rip

Command Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงตารางเส้นทางอาร์ไอพี ของเราเตอร์

ตัวอย่างการใช้งาน

RouterA#**show ip route rip**

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

Gateway of last resort is not set

r 161.246.5.0 [120/2] via 161.246.1.1 255.255.255.0

Destination Gateway Interface metric

161.246.5.0 161.246.1.1 s0 1

RouterA#

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Show Running-Config

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงการตั้งค่าของเราเตอร์ขณะนั้น

ตัวอย่างการใช้งาน

RouterA#**show running-config**

version 11.3

service timestamps debug uptime

service uimestamps log uptiom

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

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงค่าที่กำหนดให้กับเราเตอร์สำหรับตอนเปิดโปรแกรม

ตัวอย่างการใช้งาน

RouterA#**show startup-config**

version 11.3

service timestamps debug uptime

service uimestamps log uptiom

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

Command Mode 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 11.1(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)

เอกสารนี้เป็นที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

Command Mode Interface Commands

เป็นคำสั่งเปลี่ยนสถานะของอินเทอร์เฟสของเราเตอร์ ถ้าต้องการให้อินเทอร์เฟสมีสถานะเป็นเปิดจะใช้คำสั่ง No Shutdown และในทางกลับกันใช้คำสั่ง Shutdown เพื่อเปลี่ยนสถานะของอินเทอร์เฟสเป็นปิด

ตัวอย่างการใช้งาน

Router(config)#**int e0**

Router(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

Router(config)#

Telnet

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับ telnet ไปยังเราเตอร์เครื่องอื่นเพื่อตั้งค่าให้กับเราเตอร์นั้น

รูปแบบคำสั่ง Telnet <Ip Address>

<Ip Address> เป็นเลขไอพีแอดเดรสของอินเทอร์เฟสของเราเตอร์เครื่องที่ต้องการเข้าไปกำหนดค่า

ตัวอย่างการใช้งาน

RouterA#**telnet 161.246.5.1**

Router2#

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Traceroute

Command Mode 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>

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข

รายละเอียดของเราเตอร์แต่ละรุ่น

Overview of the Router

Table 1-1 lists the router models discussed in this publication and provides a summary of the interfaces supported on each model. These router models are similar in functionality, but differ in the number of interfaces supported.

Table 1-1: Summary of Router Interfaces

Model	Ethernet AUI[1] (DB-15)	Token Ring (DB-9)	Serial (DB-60)	ISDN[2] BRI[3] (RJ-45)
Cisco 2501/ CPA2501	1	-	2	-
Cisco 2502/ CPA2502	-	1	2	-
Cisco 2503/ CPA2503	1	-	2	1
Cisco 2504/ CPA2504	-	1	2	1
Cisco 2513/ CPA2513	1	1	2	-
Cisco 2514/ CPA2514	2	-	2	-
Cisco 2515[4]	-	2	2	-

[1] AUI = attachment unit interface.

[2] ISDN = Integrated Services Digital Network.

[3] BRI = Basic Rate Interface.

[4] A CPA2515 model is not available.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Note Throughout the remainder of the publication, one model number will be used in text references. For example, references to the model 2501 router will apply to both the Cisco 2501 and CPA2501 routers

Hardware Features

In addition to the interfaces listed in Table 1-1, the routers include the following hardware features:

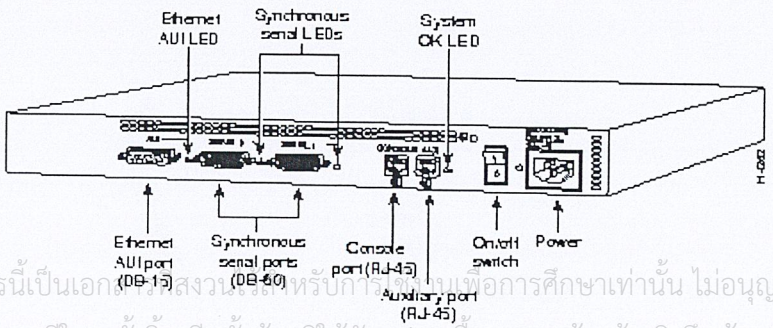
- Dynamic random-access memory (DRAM) for main memory and shared memory
- Nonvolatile random-access memory (NVRAM) for storing configuration information
- Flash memory for running the Cisco IOS software
- EIA/TIA-232 console port for local system access using a console terminal
- EIA/TIA-232 auxiliary port for remote system access using a modem

---

Note EIA/TIA-232 and EIA/TIA-449 were known as recommended standards RS-232 and RS-449 before their acceptance as standards by the Electronic Industries Association (EIA) and Telecommunications Industry Association (TIA).

Figure 1-1 to Figure 1-7 show the rear panels of the router models discussed in this publication.

Figure 1-1: Model 2501 Router Rear Panel



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Figure 1-2: Model 2502 Router Rear Panel

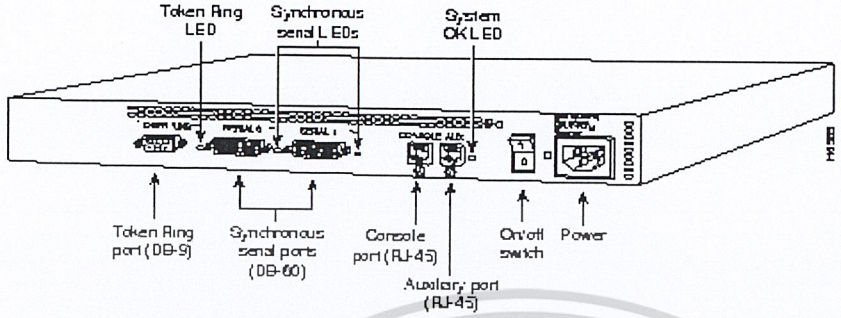


Figure 1-3: Model 2503 Router Rear Panel

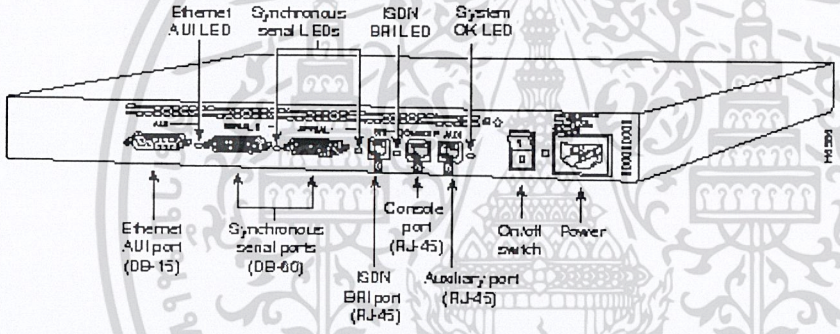


Figure 1-4: Model 2504 Router Rear Panel

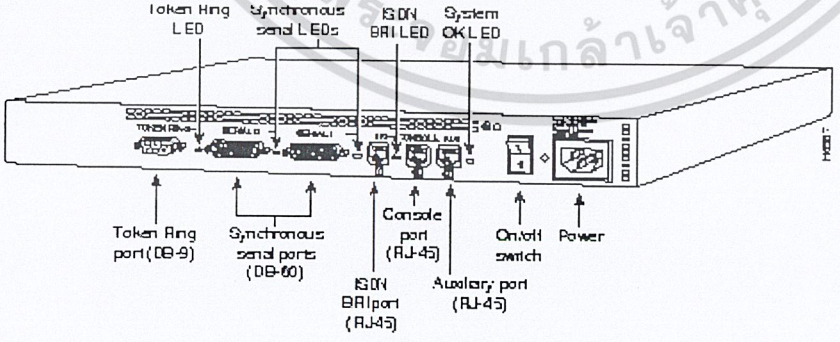


Figure 1-5: Model 2513 Router Rear Panel

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

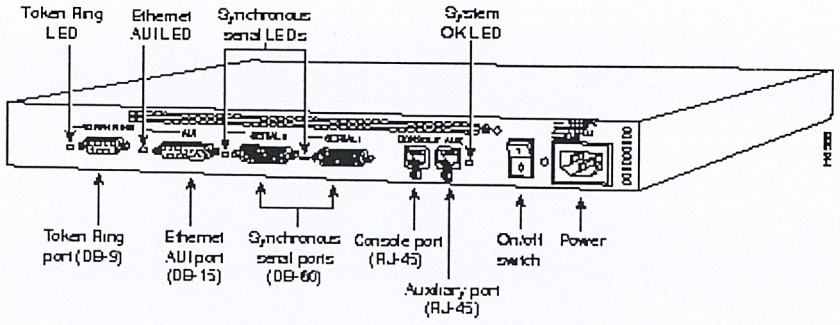


Figure 1-6: Model 2514 Router Rear Panel

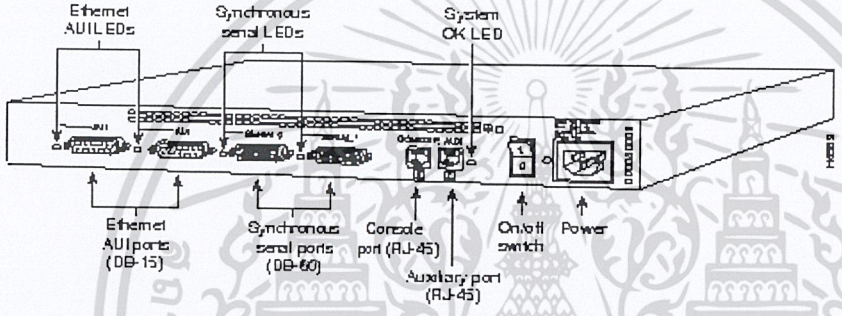
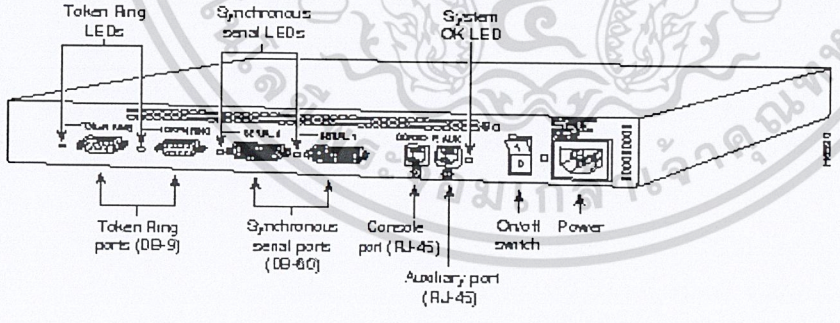


Figure 1-7: Model 2515 Router Rear Panel



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

หนังสืออ้างอิง

[1] วีระศักดิ์ ซึงถาวร,"Java Programming Volume 1", ซีเอ็ดยูเคชั่น,432 หน้า.2543.

[2] วีระศักดิ์ ซึงถาวร,"Java Programming Volume 2", ซีเอ็ดยูเคชั่น,432 หน้า.2545.

[3] สุรศักดิ์ สงวนพงษ์,"สถาปัตยกรรมและโพรโตคอลทีซีพี/ไอพี",ซีเอ็ดยูเคชั่น,528 หน้า,2543.

[4] Deitel,"Java How to program",third edition,,Prentice Hall,1355 pages,2000.

[5] Jeff Doyle,"CCIE Professional Development:Routing TCP/IP",volume I,Macmillan Technical Publishing,1025pages,1998.

เว็บไซต์อ้างอิง

[1] http://www.acm.org/crossroads/xrds6-5/ovp65.html

[2] http://www.ccs.neu.edu/home/kenb/com3337/rmi_tut.html

[3] http://www.cisco.com

[4] http://developer.java.sun.com/developer/onlineTraining/rmi/RMI.html

[5] http://www-ece2.engr.ucf.edu/~xhu/rmi_tutor.html

[6] http://www.javacoffeebreak.com/articles/javarmi/javarmi.html

[7] http://www.ryerson.ca/~dgrimsha/courses/cps841/RMICallbacks.html

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้