โปรแกรมจำลองการทำงานเครือข่าย

Network Simulator

นาย จักรพงษ์ ศุภพิบูลย์กุล

นาย จิตตินันท์ สุวรรณเรืองศรี

นาย ชัยพร จตุภมรศรี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

โปรแกรมจำลองการทำงานเครือข่าย

Network Simulator

โดย

นาย จักรพงษ์ ศุภพิบูลย์กุล

นาย จิตตินันท์ สุวรรณเรืองศรี

นาย ชัยพร จตุภมรศรี

อาจารย์ที่ปรึกษา

อาจารย์ ธนา หงษ์สุวรรณ

ดร.ศักดิ์ชัย ทิพย์จักษุรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมจำลองการทำงานเครือข่าย

Network Simulator

ผู้จัดทำ

1. นาย จักรพงษ์	ศุภพิบูลย์กุล	รหัสประจำตัว	44010061
2. นาย จิตตินันท์	สุวรรณเรืองศรี	รหัสประจำตัว	44010069
3. นาย ชัยพร	จตุภมรศรี	รหัสประจำตัว	44010102

อาจารย์ที่ปรึกษา

(อาจารย์ ธนา หงษ์สุวรรณ)

อาจารย์ที่ปรึกษา

(ดร.ศักดิ์ชัย ทิพย์จักษุรัตน์)

โปรแกรมจำลองการทำงานของเครือข่าย

นาย จักรพงษ์	สุภพิบูลย์กุล	44010061
นาย จิตตินันท์	สุวรรณเรืองศรี	44010069
นาย ชัยพร	จตุภมรศรี	44010102
อ.ธนา	หงษ์สุวรรณ	อ.ที่ปรึกษา
ดร.ศักดิ์ชัย	ทิพย์จักษุรัตน์	อ.ที่ปรึกษา

ปีการศึกษา 2547

บทคัดย่อ

ปัจจุบันระบบเครือข่ายมีความสำคัญมากขึ้น แต่การติดตั้งระบบเครือข่ายยังคงเป็นไปอย่างไม่ง่ายนัก รวมทั้งอุปกรณ์ในการเชื่อมต่อยังคงมีราคาแพง ทำให้โอกาสในการฝึกหัดใช้ให้เกิดความคุ้นเคยมีน้อยลง มีผลให้ผู้ที่มีความสามารถและความชำนาญในการตั้งค่าเราเตอร์และสวิตช์มีน้อยไปด้วย ทั้งๆ ที่ความต้องการมีมาก

โครงงานนี้เป็นโครงงานที่พัฒนาต่อจากโครงงานฝึกหัดการปรับแต่งค่าให้กับอุปกรณ์เครือข่ายของปีที่ผ่านมาเพื่อจำลองการตั้งค่าเราเตอร์และสวิตช์ของบริษัทซิสโก้ มีความสามารถเพิ่มจากปีที่แล้วคือ โพรโตคอล เอสทีพี และ Per-VLAN สามารถตรวจสอบความถูกต้องของการตั้งค่าได้ด้วย ผู้ใช้สามารถสร้างเครือข่ายของเราเตอร์และสวิตช์ขึ้นมาเองได้เพื่อให้ศึกษาภาพรวมของระบบและยังสามารถกำหนดลักษณะการเปลี่ยนแปลงบนเครือข่ายได้ โดยใช้หลักการของ Object Oriented Programming พัฒนาโดยภาษาจาวา

Network Simulator

Mr. Jakkrapong	Suppiboonkul	44010061
Mr. Jittinan	Suwanroungsri	44010069
Mr. Chaiyaporn	chatupamornsri	44010102
Mr. Thana	Hongsuwan	Advisor
Dr.Sakchai	Thipchaksurat	Advisor

Academic Year 2004

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 Router and Switch. This program is designated for Single user. In addition, this program can also deal with 5 ways of protocol which are RIP, OSPF, BGP, IGRP and STP. And we add Per-VLAN. Users are able to create network view for a network system overview and assign traffic's shape for network base on Object Oriented Programming and Java Application.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะไม่สามารถเสร็จสมบูรณ์ได้ถ้าไม่ได้รับความช่วยเหลือและการร่วมมือของบุคคลหลายๆ ฝ่ายด้วยกัน โดยฉฉพาะอย่างยิ่งบุคคลผู้ซึ่งเป็นผู้จุดประกายความคิดให้เกิดหัวข้อปริญญานิพนธ์นี้ขึ้นมา นั่นก็คืออาจารย์ธนา หงษ์สุวรรณ และดร.ศักดิ์ชัย ทิพย์จักษุรัตน์ อาจารย์ที่คอยให้คำปรึกษาปริญญานิพนธ์ตลอดเวลา และขอบพระคุณคณาจารย์ทุกท่านในภาควิชาวิศวกรรมศาสตร์ที่ได้ให้คำแนะนำ และความรู้ทางด้านคอมพิวเตอร์

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์โดยเฉพาะห้องเน็ตเวิร์ค (Network) ที่ได้เอื้อเฟื้อสถานที่ ให้คณะผู้จัดทำได้ทำการวิจัย และช่วยอำนวยความสะดวกต่างๆ ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ชาว Network ทุกคนที่คอยให้ความช่วยเหลือในการทำงานตลอดเวลา เป็นที่ปรึกษายามมีปัญหา รวมทั้งให้ยืมทรัพยากรที่จำเป็นต่างๆ ขอบคุณชาว Hardware ที่คอยให้ความสนุกยามว่างอย่างสม่ำเสมอ

ที่สำคัญและขาดมิได้ คือ ต้องขอขอบพระคุณบิดา มารดาที่ได้ให้กำเนิด คอยสั่งสอน และให้การสนับสนุนการศึกษา กิจกรรมต่างๆ นับเป็นพระคุณที่หาใดเปรียบมิได้ ทางคณะผู้จัดทำขออกราบขอบพระคุณมา ณ ที่นี้ด้วย

จักรพงษ์ สุภพิบูลย์กุล

จิตตินันท์ สุวรรณเรืองศรี

ชัยพร จตุภมรศรี

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญภาพประกอบ

สารบัญภาพประกอบ(ต่อ)

หน้าที่

สารบัญภาพประกอบ(ต่อ)

สารบัญภาพประกอบ(ต่อ)

หน้าที่

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ปัจจุบันระบบเครือข่ายมีความสำคัญมากขึ้น แต่การติดตั้งระบบเครือข่ายยังคงเป็นไปอย่างไม่ง่ายนัก รวมทั้งอุปกรณ์ในการเชื่อมต่อยังคงมีราคาแพง ทำให้โอกาสในการฝึกหัดใช้ให้เกิดความคุ้นเคยมีน้อยลง มีผลให้ผู้ที่มีความสามารถและความชำนาญในการตั้งค่าเราเตอร์และสวิตช์มีน้อยไปด้วย ทั้งๆ ที่ความต้องการมีมาก

ดังนั้นการจัดทำโปรแกรมฝึกการปรับแต่งค่าให้กับอุปกรณ์เครือข่าย จึงเป็นแนวคิดเพื่อเปิดโอกาสให้กับบุคคลทั่วไปที่มีความสนใจ ในการติดตั้งและตั้งค่าอุปกรณ์สวิตช์ และเราเตอร์ได้ฝึกฝนโดยที่ไม่จำเป็นต้องซื้ออุปกรณ์จริงๆ ทั้งยังสามารถทดลองตั้งค่าเพื่อก่อให้เกิดความชำนาญ และสามารถนำไปปฏิบัติงานจริงได้

1.2 วัตถุประสงค์ของปริญญานิพนธ์

โครงการโปรแกรมฝึกการปรับแต่งค่าให้กับอุปกรณ์เครือข่าย จัดทำขึ้นภายใต้วัตถุประสงค์หลัก ดังนี้

1. เพื่อศึกษารายละเอียดและการทำงานโพรโตคอลในการเลือกเส้นทางแบบต่างๆได้
2. เพื่อศึกษาการตั้งค่าให้กับเราเตอร์และสวิตช์ โดยไม่ต้องใช้อุปกรณ์จริง
3. เพื่อสร้างโปรแกรมต้นแบบจำลองการตั้งค่าเครือข่ายที่สามารถใช้งานคำสั่งในการตั้งค่าเราเตอร์และสวิตช์ได้เพิ่มมากขึ้น
4. เพื่อฝึกการเขียนโปรแกรมภาษาจาวาแบบ Multi-threading

1.3 ขอบเขตของปริญญานิพนธ์

1. โปรแกรมทำงานในสมมุติฐานว่า เครือข่ายไม่มีข้อผิดพลาดคือ ไม่มีเกิดการเสียหาย หรือสูญหายของข้อมูลที่รับส่งกันภายในเครือข่าย
2. โปรแกรมจะทำงานโดยอ้างอิงผลิตภัณฑ์ ของซิสโก้ ซึ่งได้แก่ เราเตอร์ Cisco 1760 และสวิตช์ Cisco 2950 เป็นหลัก
3. การทำงานของสวิตช์เป็นการทำงานในระดับเลเยอร์ 2 (Data link Layer)
4. โปรแกรมจะมีคำสั่งในการปรับตั้งค่าให้แต่สวิตช์ และเราเตอร์ โดยอ้างอิงจาก IOS ของ Cisco version 12.0,12.1
5. การป้อนคำสั่งในการปรับตั้งค่าต่างๆให้แก่สวิตช์ และเราเตอร์ ทำผ่านหน้า Interface Console ของโปรแกรม ซึ่งเป็นการใช้งานแบบผู้ใช้คนเดียว (Stand Alone)

1.4 ขั้นตอนการดำเนินงาน

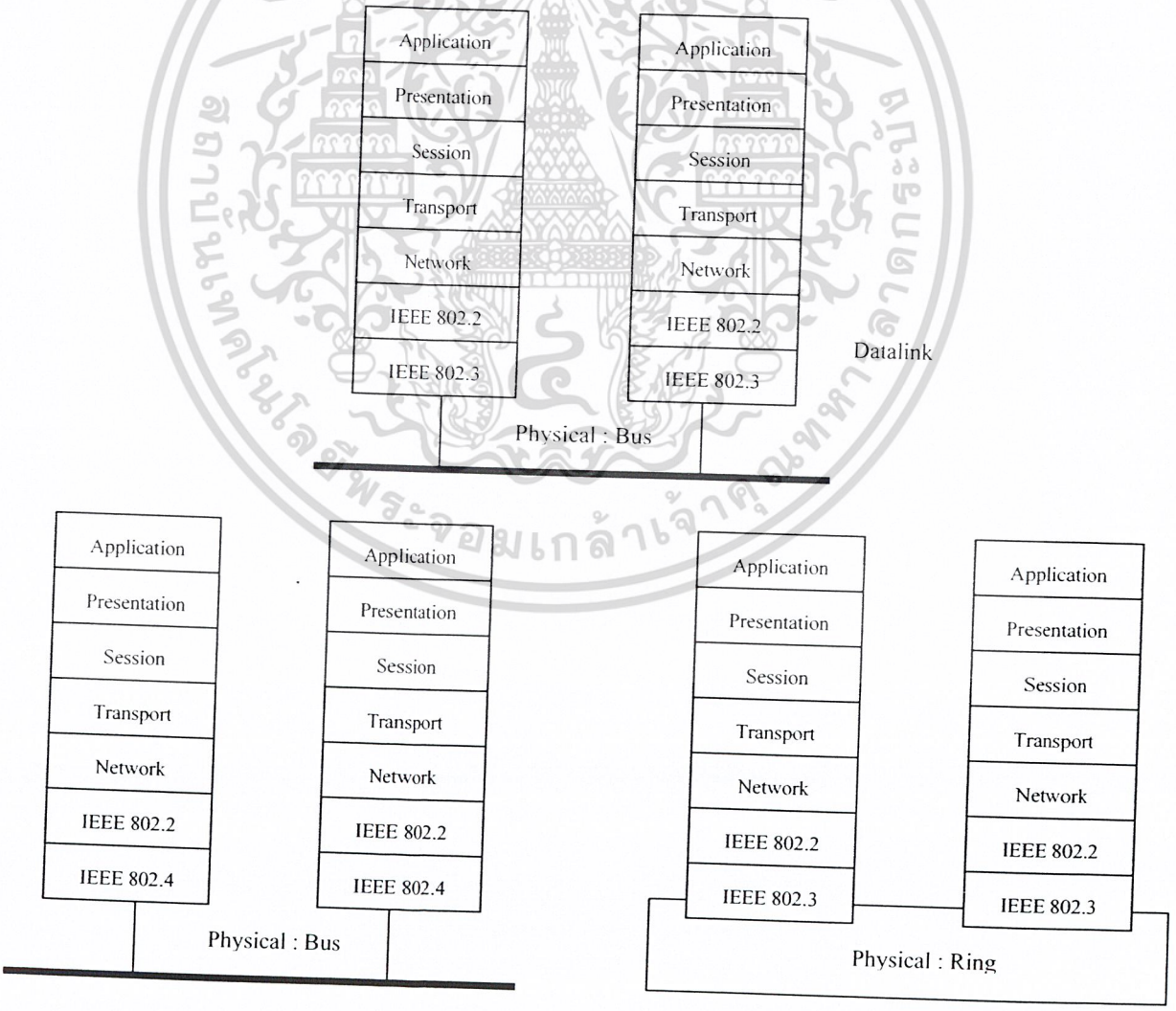
1. ศึกษาคำสั่งของ Router และ Switch จากโปรแกรม Boson และจากโปรเจคปี 2546
2. ศึกษาจุดเด่นจุดด้อยของโปรแกรม Boson และจากโปรเจคปี 2546 เพื่อนำมาแก้ไขและพัฒนา
3. ปรับปรุงโครงสร้างของโปรแกรมและ User Interface
4. เขียนโปรแกรมเพิ่มเติมในส่วนของการสร้าง Router และ Switch ในส่วนของ User Interface
5. แก้ไขส่วนในการออกแบบเครือข่าย และการเชื่อมต่อระหว่าง link
6. เขียน Function ใน Switch
7. เขียน console และเรียกใช้ command ของ Router และ Switch
8. เขียนส่วนการจัดการ Protocol ของ Switch
9. จัดการ Bug ของโปรแกรม และตกแต่ง Interface
10. แก้ไขส่วนที่บกพร่อง

บทที่ 2
เทคโนโลยีเครือข่ายคอมพิวเตอร์

ทีซีพี/ไอพีผ่านการออกแบบมาให้สามารถทำงานกับระบบการสื่อสารระดับล่าง โดยไม่จำกัดประเภท ในปัจจุบันจึงมีฮาร์ดแวร์เครือข่ายจำนวนมากทั้งในกลุ่มของแลนและแวนที่รองรับการทำงานร่วมกับทีซีพี/ไอพี

2.1 แลนตามข้อกำหนดของ IEEE

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



รูปที่ 2-1 เป็นแบบจำลองบางส่วนของมาตรฐานซึ่งเป็นที่รู้จักแพร่หลาย

IEEE กำหนดเครือข่ายเฉพาะที่โดยใช้ตัวเลข 802 ตามด้วยตัวเลขย่อยเป็นรหัสประจำแต่ละมาตรฐาน

- IEEE 802.3 หรืออีเทอร์เน็ต ใช้โพรโตคอลซีเอสเอ็มเอ ซีดีในโทโพโลยีแบบบัส
- IEEE 802.4 หรือโทเค็นบัส ใช้โพรโตคอลส่งผ่านโทเค็นในโทโพโลยีแบบบัส
- IEEE 802.5 หรือโทเค็นริง ใช้โพรโตคอลส่งผ่านโทเค็นในโทโพโลยีแบบวงแหวน

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

2.1.1 IEEE 802.3 (Ethernet)

อีเทอร์เน็ต เป็นเครือข่ายที่มีความเร็วการส่งข้อมูล 10 เมกะบิตต่อวินาที สถานีในเครือข่ายอาจมีโทโพโลยีแบบบัสหรือแบบดาว IEEE ได้กำหนดมาตรฐานอีเทอร์เน็ตไว้หลายชนิดสายสัญญาณ เช่น

- 10Base5 อีเทอร์เน็ตโทโพโลยีแบบบัสซึ่งใช้ในสายโคแอกเชียลแบบหนา (Thick Ethernet) ความยาวของสายในเซ็กเมนต์หนึ่งๆ ไม่เกิน 500 เมตร
- 10Base2 อีเทอร์เน็ตโทโพโลยีแบบบัสซึ่งใช้สายโคแอเชียลแบบบาง (Thin Ethernet) ความยาวของสายในเซ็กเมนต์หนึ่งๆ ไม่เกิน 185 เมตร
- 10BaseT อีเทอร์เน็ตโทโพโลยีแบบดาวซึ่งใช้ฮับเป็นศูนย์กลาง สถานีและฮับเชื่อมด้วยสายยูทีพี (Unshield Twisted Pair) ด้วยความยาวไม่เกิน 100 เมตร

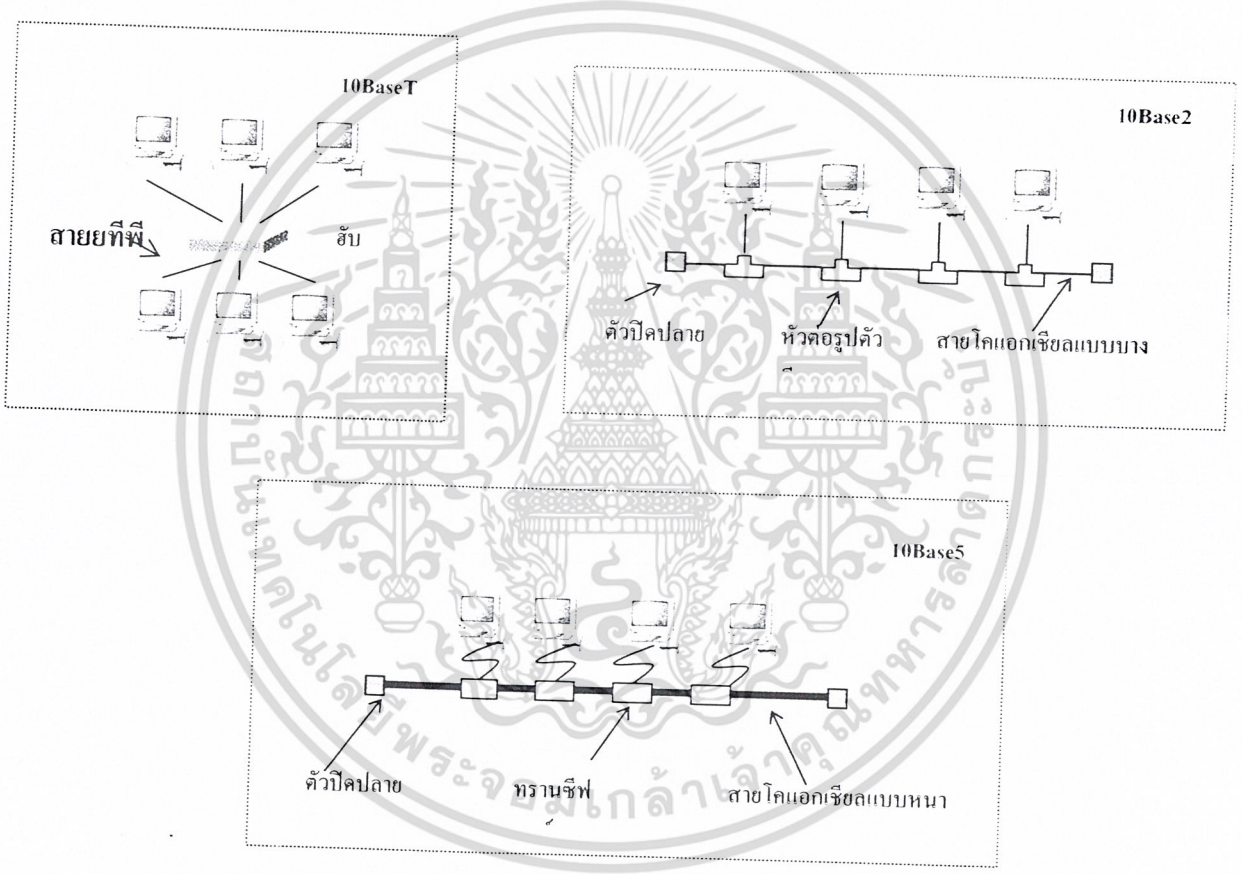
รูปที่ 2-2 แสดงถึงลักษณะเครือข่ายอีเทอร์เน็ตแยกตามประเภทสายสัญญาณรหัสขึ้นต้นด้วย 10 หมายถึงความเร็วสายสัญญาณ 10 เมกะบิตต่อวินาที คำว่า “Base” หมายถึงสัญญาณชนิด “Baseband” รหัสถัดมาหากเป็นตัวเลขหมายถึงความยาวสายต่อเซ็กเมนต์ในหน่อยหนึ่งร้อยเมตร (5=500,2 แทนค่า 185) หากเป็นอักษรจะหมายถึงชนิดของสายสัญญาณ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

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

2.1.2 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

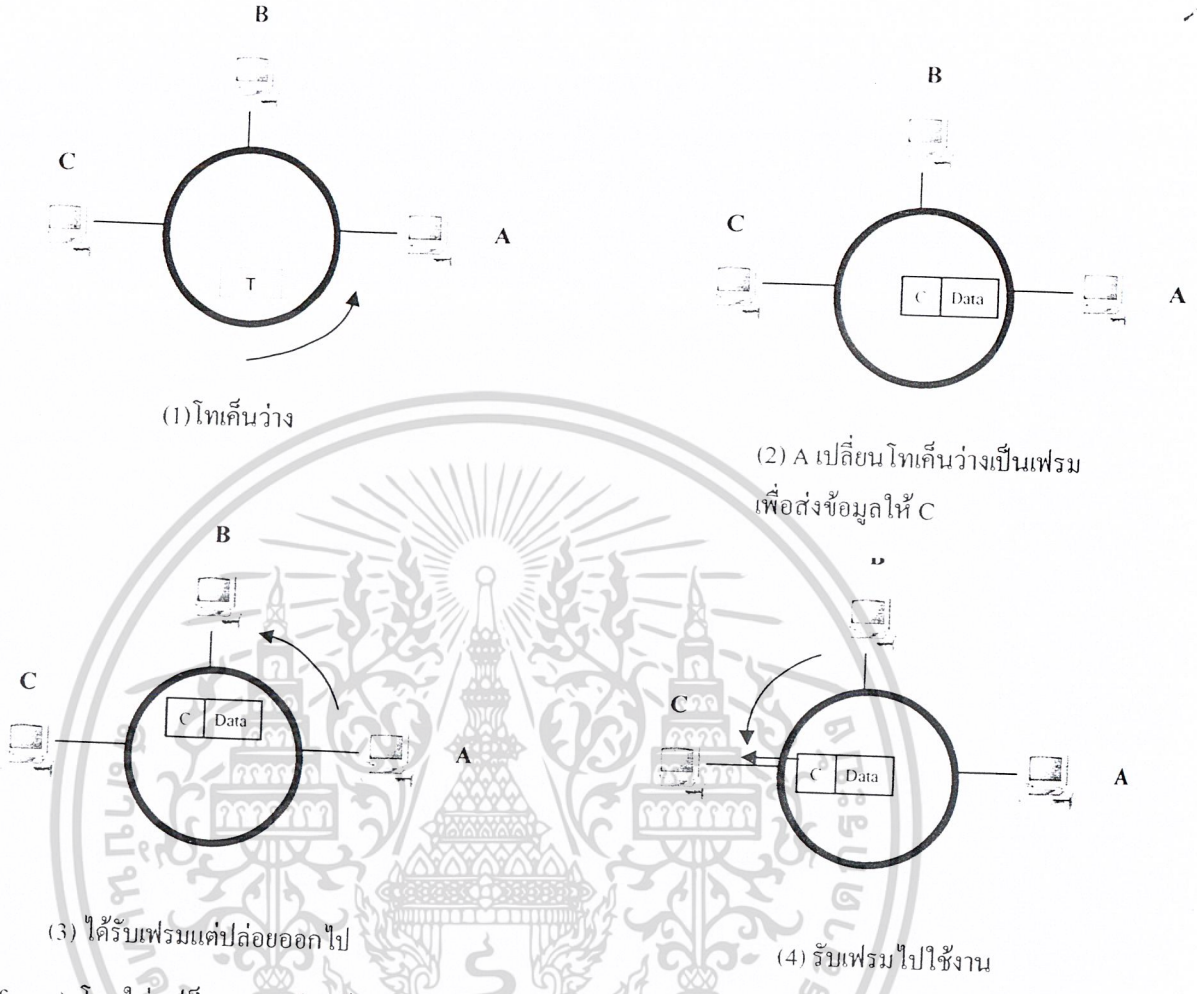


รูปที่ 2-2 อีเทอร์เน็ตประเภทต่างๆ

2.1.3 IEEE 802.5 (Token Ring)

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

รูปที่ 2-3 แสดงการทำงานของโทเค็นริง โดยมีเฟรมพิเศษเรียกว่า โทเค็นว่าง (free token) วิ่งวนอยู่ สถานีที่ต้องการส่งข้อมูลจะรอให้โทเค็นว่างเดินทางมาถึงแล้วรับโทเค็นว่างมาเปลี่ยนเป็น **เฟรมข้อมูล**



data frame) โดยใส่แฟล็กแสดงเฟรมข้อมูลและบรรจุแอดเดรสของสถานีต้นทางและปลายทางตลอดจนข้อมูลอื่นๆ จากนั้นจึงปล่อยเฟรมนี้ออกไป

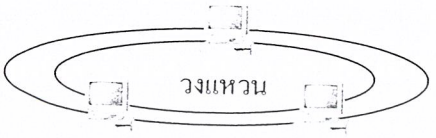
รูปที่ 2-3 การทำงานโทเค็นริง

เมื่อสถานีปลายทางได้รับเฟรมจะสำเนาข้อมูลไว้และปล่อยเฟรมให้วนกลับมายังสถานีส่ง สถานีส่งจะตรวจสอบเฟรมและปล่อยโทเค็นว่างคืนสู่เครือข่ายให้สถานีอื่นมีโอกาสส่งข้อมูลต่อไป กลไกแบบส่งผ่านโทเค็นจัดอยู่ในประเภทประเมินเวลาได้ กล่าวคือสามารถคำนวณเวลาสูงสุดที่สถานีมีสิทธ์จับโทเค็นเพื่อส่งข้อมูลได้ โทเค็นริงจึงเหมาะกับระบบที่ต้องการความแน่นอนทางเวลาหรืองานแบบเวลาจริง

2.1.4 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



รูปที่ 2-4 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)

บทที่ 3
อุปกรณ์เครือข่าย

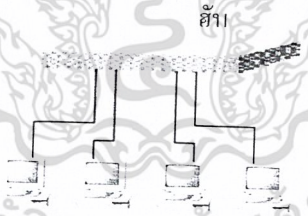
3.1 อุปกรณ์เครือข่าย

เครือข่ายขนาดเล็กมักประกอบด้วยสถานีงานและสถานีเซิร์ฟเวอร์ต่อเชื่อมกัน การเชื่อมเครือข่ายขนาดเล็กจำนวนหลายเครื่องเข้าด้วยกันเพื่อแลกเปลี่ยนข้อมูลระหว่างกันต้องอาศัยอุปกรณ์เครือข่ายที่มีลักษณะสมบัติแตกต่างกันไป อุปกรณ์เครือข่ายพื้นฐานที่พบโดยทั่วไปและจะอธิบายในหัวข้อนี้ได้แก่ ฮับ บริดจ์ และเราเตอร์

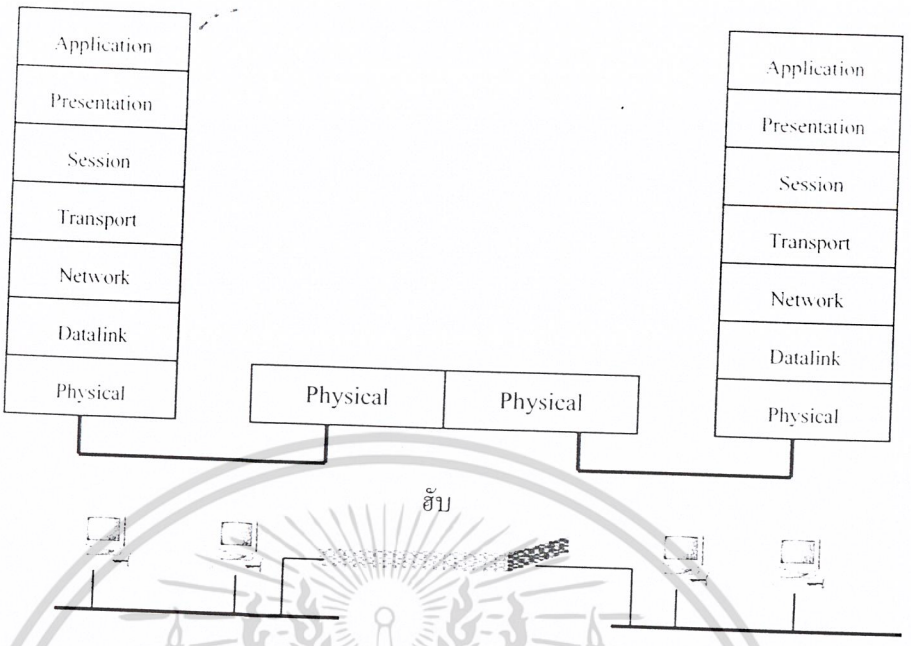
3.1.1 ฮับ

ฮับ (Hub) เป็นอุปกรณ์เชื่อมสถานีเครือข่ายที่ใช้โทโพโลยีแบบดาว เช่น อีเทอร์เน็ต 10BaseT ดังรูปที่ 3-1 ฮับมีพอร์ตได้หลายแบบเช่น พอร์ต RJ-45 ใช้กับสายคู่ดีเกลียวในเครือข่าย 10BaseT หรือพอร์ตไฟเบอร์ใช้กับเส้นใยนำแสงในเครือข่าย 10BaseF

ฮับเป็นอุปกรณ์ในระดับฟิสิคัลเช่นเดียวกับรีพีตเตอร์ดังรูปที่ 3-2 ด้วยเหตุนี้จึงมักเรียกฮับว่าเป็นมัลติพอร์ตรีพีตเตอร์ หน้าที่ของฮับคือขยายสัญญาณและกระจายแพ็กเก็ตไปทุกพอร์ต ฮับใช้เชื่อมต่อเครือข่ายประเภทเดียวกันเท่านั้น เครือข่ายที่เชื่อมด้วยฮับจะรวมเป็นเครือข่ายเดียวกันดังนั้นแพ็กเก็ตที่สร้างจากเครือข่ายหนึ่งผ่านฮับไปอีกเครือข่ายหนึ่ง



รูปที่ 3-1 การใช้ฮับในโทโปโลยีแบบดาว



รูปที่ 3-2 แบบจำลองการทำงานของฮับ

3.1.2 บริดจ์

บริดจ์ (Bridge) ทำหน้าที่เชื่อมเครือข่ายย่อยสองเครือข่ายเข้าด้วยกัน บริดจ์ทำงานในระดับชั้นดาต้าลิงก์จึงสามารถใช้เชื่อมเครือข่ายประเภทเดียวกันหรือต่างกันได้ เช่น เชื่อมอีเทอร์เน็ตเข้ากับโทเค็นริง แบบจำลองการทำงานของบริดจ์แสดงได้ดังรูปที่ 3-3

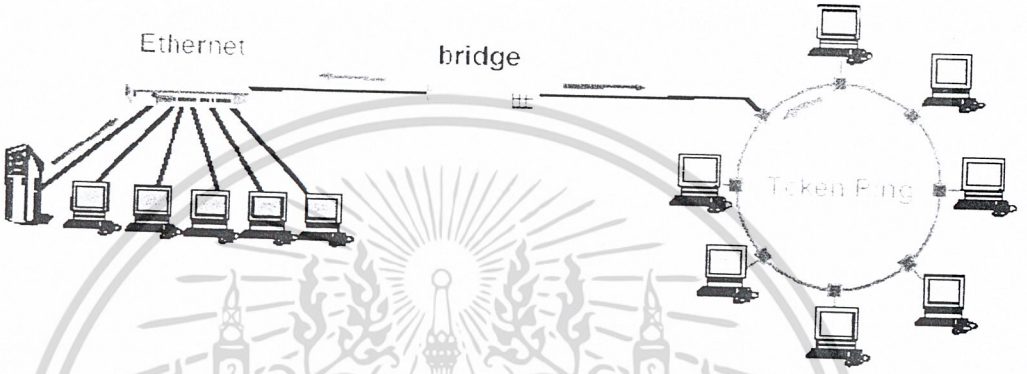
เมื่อใช้บริดจ์เชื่อมเครือข่ายประเภทเดียวกัน บริดจ์จะตรวจสอบว่าจะนำส่งแพ็กเก็ตโดยตรวจสอบฮาร์ดแวร์แอดเดรสปลายทางกับตารางเลือกเส้นทางที่บริดจ์สร้างขึ้น หากมีแพ็กเก็ตที่ต้องข้ามจากบริดจ์จากเครือข่ายหนึ่งไปยังอีกเครือข่ายหนึ่ง บริดจ์จะทำสำเนาแพ็กเก็ตบิตต่อบิต ข้ามไปยังอีกเครือข่ายหนึ่งโดยไม่เปลี่ยนแปลงเฮดเดอร์หรือข้อมูล หากสถานีต้นทางและสถานีปลายทางอยู่ในเครือข่ายเดียวกัน บริดจ์จะได้รับแพ็กเก็ตแต่ไม่ส่งแพ็กเก็ตไปยังเครือข่ายอีกด้านหนึ่ง

ในองค์กรขนาดใหญ่มักจะมีระบบเครือข่ายแลนอยู่หลายระบบ ซึ่งอาศัยการเชื่อมต่อโดยใช้เราเตอร์ (Router) สำหรับเครือข่ายที่ใช้โพรโตคอลอย่างเดียวกันเท่านั้น อุปกรณ์เครือข่ายประเภทบริดจ์ จึงได้รับการพัฒนาขึ้นมา เพื่อใช้ในการเชื่อมต่อระบบเครือข่ายต่างชนิดกันเข้าด้วยกัน โดยอาศัยแมคแอดเดรสในการกำหนดเส้นทางการสื่อสาร ดังนั้นบริดจ์จึงสามารถเรียกอีกอย่างได้ว่า Low-level router เนื่องจากบริดจ์ทำงานในชั้นที่ 2 (Data Link Layer) ดังนั้นจึงมองไม่เห็นความแตกต่างของแพ็กเก็ต IP, IPX และอื่น ๆ ทำให้สามารถรับ-ส่งข้อมูลของโพรโตคอลได้เกือบทุกชนิด แต่การควบคุมเส้นทางในการส่งข้อมูลของบริดจ์ จะมีความยืดหยุ่นน้อยกว่าเราเตอร์ เนื่องจากจะใช้ข้อมูลแมคแอดเดรสเท่านั้นในการกำหนดเส้นทาง ดังนั้น บริดจ์จึงเหมาะสมกับระบบเครือข่ายที่มีความซับซ้อนไม่มากนัก

บริดจ์นำส่งบรอดคาสต์แพ็กเก็ตข้ามเครือข่ายโดยไม่ป้องกัน สถานีอีเทอร์เน็ตหรือโทเคนริงต้องดำเนินการกับแพ็กเก็ตที่สถานีอื่นบรอดคาสต์มาให้ บรอดคาสต์แพ็กเก็ตมักเกิดจากการทำงานของ

โพรโตคอลเออาร์พี บูตพี หรืออาร์ไอพี หากจำนวนบรอดคาสต์แพ็กเก็ตสูงจะส่งผลให้สมรรถนะการทำงานของสถานีในเครือข่ายลดลง

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



รูปที่ 3-3 แสดงระบบเครือข่ายที่ใช้บริดจ์ในการเชื่อมต่อ

3.1.2.1 ชนิดของบริดจ์

บริดจ์มีอยู่ 2 ชนิด คือ ทรานส์แพเร้นท์บริดจ์ (Transparent Bridge) และ ซอสเราท์บริดจ์ (Source Route Bridge)

3.1.2.1.1 ทรานส์แพเร้นท์บริดจ์ (Transparent Bridge)

พัฒนาโดย Digital Equipment Corporation ในต้นปี 1980 ซึ่งต่อมาได้มีการกำหนดมาตรฐาน IEEE 802.1 โดยผู้ใช้งานได้กำหนดความต้องการทรานส์แพเร้นท์บริดจ์ให้มีลักษณะดังนี้ แรกทีเดียวผู้ใช้ (ทุกระบบ) จะต้องไม่มีส่วนเกี่ยวข้องกับการทำงานของทรานส์แพเร้นท์บริดจ์ การมีอุปกรณ์ประเภทนี้อยู่ในระบบ หรือไม่มีอยู่ก็ตาม จะต้องไม่มีผลกระทบใด ๆ ต่อผู้ใช้งาน ผู้ใช้ทั่วไปจะต้องสามารถซื้ออุปกรณ์นี้มาจากตัวแทนจำหน่ายของบริษัทใดก็ได้ การติดตั้งจะต้องมีความยุ่งยากเพียงแค่การเสียบปลั๊กไฟฟ้าและเสียบสายระบบเครือข่ายต่าง ๆ เข้ากับอุปกรณ์นี้ แม้ว่าจะไม่มีการกำหนดค่าพารามิเตอร์ใด ๆ ตัวอุปกรณ์ก็จะต้องสามารถทำงานได้ในทันที ผลที่เกิดขึ้นนั้นน่าแปลกใจเป็นอย่างยิ่งว่า ทรานส์แพเร้นท์บริดจ์นั้นมีตัวตนและทำงานได้อย่างที่ผู้ใช้ต้องการ

ทรานส์แพเร้นท์บริดจ์ ส่วนใหญ่จะใช้ในการเชื่อมต่อระหว่างเซกเมนต์ (Segment) ของอีเทอร์เน็ต โดยการทำงานของทรานส์แพเร้นท์บริดจ์ จะเก็บข้อมูลแมคแอดเดรสของสเตชันที่ต่ออยู่ของพอร์ตต่าง ๆ โดยจะใช้ข้อมูลนั้น เมื่อมีเฟรมส่งเข้ามาที่พอร์ตนั้น

MAC Address Table

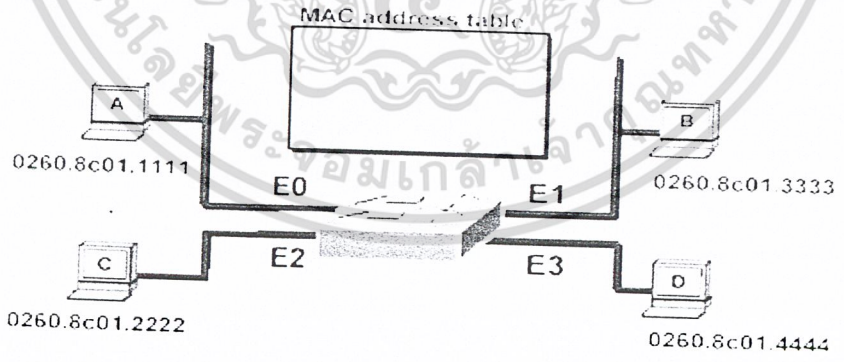
E0: 0260.8c01.1111
E2: 0260.8c01.2222
E1: 0260.8c01.3333
E3: 0260.8c01.4444

รูปที่ 3-4 แสดงตัวอย่าง MAC Address Table

เมื่อบริดจ์ได้รับเฟรมเข้ามา จะตรวจสอบแอดเดรสของสเตชันปลายทางและส่งเฟรมนั้น ออกไปยังพอร์ตที่ระบุ ยกเว้นเฟรมที่มีแอดเดรสของสเตชั่นปลายทางเป็นพอร์ทเดียวกันกับสเตชั่นต้นทาง แต่ละรายการในตารางจะมีเวลากำหนดไว้เรียกว่า Time-To-Live (TTL) โดยรายการนั้นจะถูกลบออกจากตารางเมื่อถึงเวลา TTL ที่กำหนดและ TTL จะถูกกำหนดใหม่เมื่อมีเฟรมจากสเตชันของรายการนั้นเข้ามาอีกครั้ง ซึ่งจะช่วยแก้ปัญหาในกรณีที่มีการย้ายสเตชันไปยังพอร์ตอื่น หรือการนำสเตชันนั้นออกจากระบบเครือข่าย ในกรณีที่บริดจ์ไม่สามารถหารายการที่ตรงกับแอดเดรสของสเตชั่นปลายทางได้ เฟรมนั้นจะถูกส่งไปยังทุกพอร์ตยกเว้นพอร์ตที่รับเฟรมเข้ามา

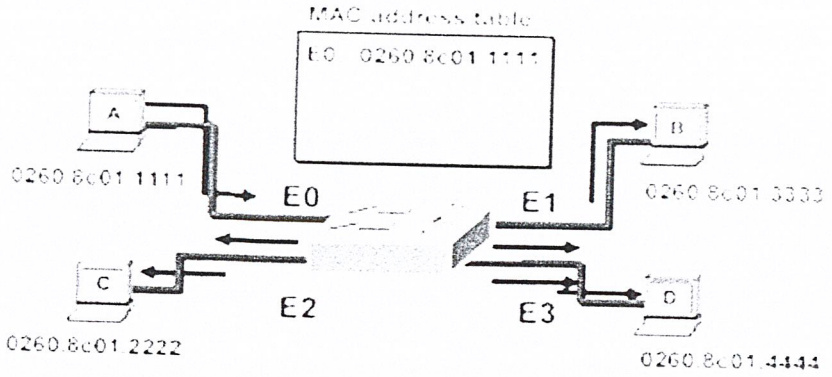
ทรานส์แพเร้นท์บริดจ์จะช่วยให้สามารถแบ่งระบบเครือข่ายออกเป็นเซกเมนต์ย่อย ๆ เพื่อลดปริมาณของการส่งข้อมูลในรูปแบบอีเทอร์เน็ตในอุปกรณ์ฮับไม่ให้คับคั่งมากเกินไป

<u>ตัวอย่างการส่งข้อมูลและการสร้างตารางแมคแอดเดรส</u>



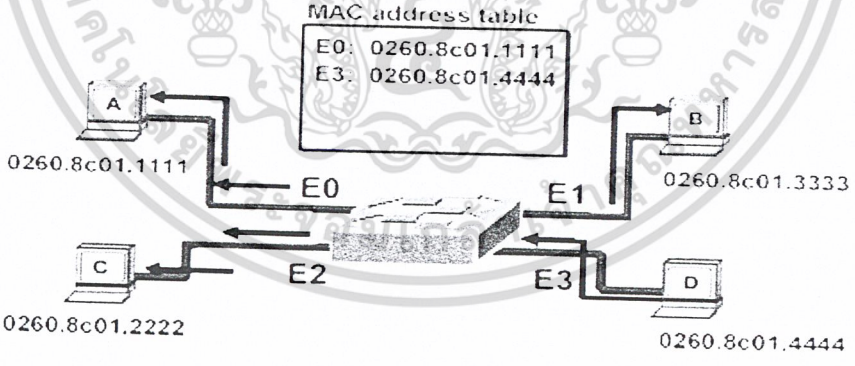
รูปที่ 3-5 แสดงการเรียนรู้แมคแอดเดรส

จากรูป เป็นตอนเริ่มต้น ยังไม่มีการส่งข้อมูล ดังนั้น ในตารางแมคแอดเดรสจึงยังว่างเปล่าอยู่ (empty)



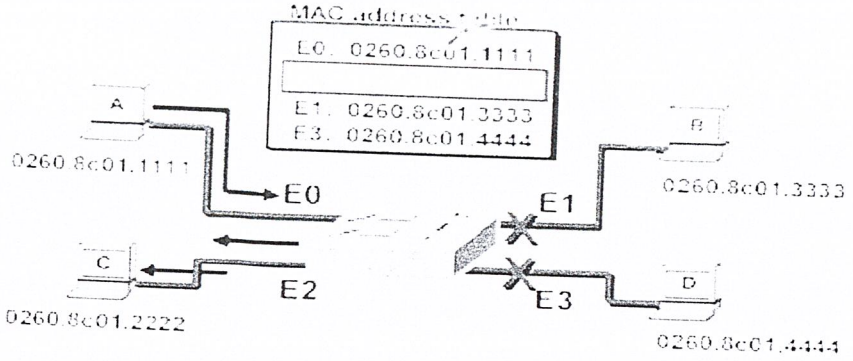
รูปที่ 3-6 แสดงการเรียนรู้แมคแอดเดรส

จากรูปสเตชัน A (Station A) ต้องการส่งข้อมูลไปยังสเตชัน C (Station C) ดังนั้นที่เฟรมข้อมูลของเอ จะมีแอดเดรสของสเตชันต้นทาง (Source Address) เป็นแมคแอดเดรสของ A คือ 0260.8c01.1111 และแอดเดรสของสเตชันปลายทาง (Destination Address) เป็นแมคแอดเดรสของ C คือ 0260.8c01.2222 เมื่อสเตชัน A ส่งเฟรมไปยังสวิตช์ จะทำการดูที่สเตชันต้นทางทำการเรียนรู้ว่า พอร์ต E0 ที่รับข้อมูลเข้ามา คือ 0260.8c01.1111 จากนั้นทำการใส่ลงในตารางแมคแอดเดรสเป็น E0: 0260.8c01.1111 จากนั้นดูต่อไปที่สเตชันปลายทาง แล้วจึงไปค้นหาที่ตารางแมคแอดเดรสว่ามีแมคแอดเดรสนี้อยู่หรือไม่ จากรูป เราจะเห็นได้ว่า ในตารางไม่มีสเตชันปลายทางอยู่ เมื่อเป็นในกรณีนี้ สวิตช์จะทำการส่งออกไปยังทุกพอร์ต (Flood Out)



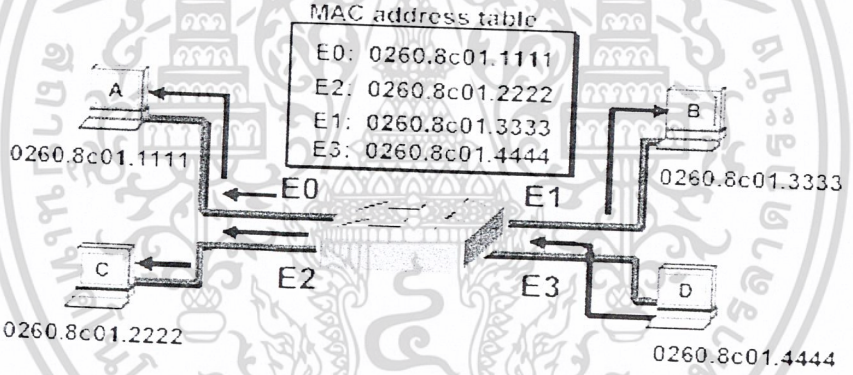
รูปที่ 3-7 แสดงการเรียนรู้แมคแอดเดรส

จากรูป สเตชัน D (Station D) ต้องการส่งเฟรมข้อมูลไปยังสเตชัน C (Station C) จะเป็นกรณีเดียวกับในรูป 3-4 นั่นเอง



รูปที่ 3-8 แสดงการทำงานของสวิตช์ในการทำฟิลเตอร์ (Filter)

จากรูปสเตชัน A ต้องการส่งเฟรมข้อมูลไปยังสเตชัน C เมื่อสวิตช์ทำการเรียนรู้แมคแอดเดรส แล้วจึงทำการค้นหาในตารางและพบว่าสเตชันปลายทางมีจุดหมายอยู่ที่พอร์ต E2 จึงทำการส่งเฟรมข้อมูลไปยังพอร์ต E2 เพียงพอร์ตเดียว ไม่ส่งไปยังพอร์ตอื่น ๆ



รูปที่ 3-9 แสดงการเรียนรู้แมคแอดเดรส

เมื่อสเตชัน D ต้องการส่งเฟรมข้อมูลแบบบอร์ดคาสต์ (Broadcast) หรือ เมาติคาสต์ (Multicast) สวิตช์ก็จะทำการส่งข้อมูลออกไปทุกพอร์ตยกเว้นพอร์ต E3 ซึ่งก็คือ พอร์ตที่รับเฟรมเข้ามานั่นเอง

3.1.2.1.2 ซอสเราท์บริดจ์ (Source-Route Bridge (SRB))

SRB เป็นอัลกอริธึมที่พัฒนาโดย IBM สำหรับการเชื่อมต่อระหว่างแลนแบบโทเก้นริง (IEEE 802.5) โดยการส่งข้อมูลแบบ SRB จะต้องมีการกำหนดเส้นทางก่อนล่วงหน้า ซึ่งมีขั้นตอนในการหาเส้นทาง คือ

เมื่อโฮสต์เอ็กซ์ (Host X) ต้องการส่งเฟรมให้โฮสต์วาย (Host Y) ในครั้งแรก โฮสต์เอ็กซ์จะไม่ทราบว่าโฮสต์วายอยู่ในเครือข่ายแลนเดียวกันหรือไม่ โฮสต์เอ็กซ์จะทำการส่งเฟรมทดสอบ ถ้าเฟรมกลับมายังโฮสต์เอ็กซ์ โดยที่บิต A ในเฟรมโทเก้นริงไม่เป็น 1 แสดงว่า โฮสต์วายอยู่ต่างเซกเมนต์

จากนั้น โฮสต์เอ็กซ์จะทำการส่งเฟรมเอกโพลเรอร์ (Explorer) ไปยังบริดจ์ บริดจ์เมื่อได้รับเฟรมเอกโพลเรอร์ จะส่งเฟรมนั้นไปยังทุกพอร์ตยกเว้นพอร์ตที่รับเฟรม โดยเพิ่มข้อมูลของเส้นทาง (Route) ในเฟรม เอ็กโพลเรอร์ตามบริดจ์นั้น เมื่อเฟรมเอกโพลเรอร์ไปถึงโฮสต์วายจะทำการตอบกลับมายังโฮสต์เอ็กซ์ตามข้อมูลเส้นทาง ซึ่งอาจจะตอบกลับมาหลายครั้งตามจำนวนเฟรมเอกโพลเรอร์ที่ได้รับ

ซึ่งโฮสต์เอ็กซ์จะต้องเลือกเส้นทางใดเส้นทางหนึ่ง โดยที่วิธีในการเลือกนั้นไม่ได้กำหนดไว้ใน IEEE 802.5 แต่สามารถเป็นไปได้หลายอย่าง เช่น

- เลือกเฟรมตอบกลับแรกที่ได้รับ
- เลือกเฟรมที่มีฮอป (Hop) น้อยที่สุด
- เลือกเส้นทางที่ยอมให้มีเฟรมขนาดใหญ่ที่สุด

เป็นต้น

เมื่อเลือกเส้นทางได้แล้ว ข้อมูลเส้นทางจะถูกกำหนดลงในเฟรมที่จะส่งไปที่โฮสต์วายในรูปแบบ Routing Information Field (RIF)

3.1.2.2 การเปรียบเทียบบริดจ์ในระบบ 802

บริดจ์ทั้งแบบทรานส์แพเร้นท์บริดจ์และแบบ SRB มีทั้งข้อดีและข้อเสียที่แตกต่างกันดังที่สรุปไว้ในตารางที่ 3-1

Issue	Transparent bridge	Source routing bridge
Orientation	Connectionless	Connection-oriented
Transparency	Fully transparent	Not transparent
Configuration	Automatic	Manual
Routing	Suboptimal	Optimal
Locating	Backward learning	Discovery frames
Failures	Handled by the bridges	Handled by the hosts
Complexity	In the bridges	In the hosts

ตารางที่ 3-1 ตารางเปรียบเทียบคุณสมบัติของทรานส์แพเร้นท์บริดจ์และแบบ SRB

หัวใจของความแตกต่างระหว่างบริดจ์ทั้ง 2 ชนิดคือ การสื่อสารเครือข่ายแบบมีการติดต่อช่วงสั้น (Connectionless) และแบบมีการติดต่ออย่างต่อเนื่อง (Connection-oriented) ทรานส์แพเร้นท์บริดจ์ไม่ใช้แนวคิดของวงจรเสมือน เส้นทางเดินของแต่ละเฟรมจะถูกเลือกอย่างเป็นอิสระ ส่วน SRB มีลักษณะตรงกันข้ามคือ ต้องมีการค้นหาเส้นทางเดินข้อมูลให้ได้เสียก่อน จากนั้นจึงใช้เส้นทางที่ค้นพบสำหรับการส่งข้อมูลจริงในภายหลัง

การทำงานของทรานส์แพเร้นท์บริดจ์จะไม่เข้าไปเกี่ยวข้องกับโฮสต์เลยแม้แต่น้อย และยังสามารถทำงานเข้ากันได้กับการส่งข้อมูลตามมาตรฐาน 802 ทุกระบบ ในขณะที่ SRB ต้องให้โฮสต์เข้ามาร่วมกระบวนการทำงานด้วยและไม่สามารถทำงานร่วมกับการส่งข้อมูลตามมาตรฐาน 802 บางระบบได้ นั่นคือโฮสต์จะต้องรู้จักโครงสร้างและการทำงานของบริดจ์เป็นอย่างดี และสามารถทำงานร่วมกันได้ การแบ่งระบบเครือข่ายออกเป็น 2 วงที่เชื่อมกันโดยวิธีการเลือกทางเดินโดยผู้ส่งข้อมูลจะต้องทำการเปลี่ยนแปลงโปรแกรมของโฮสต์ด้วย

การใช้ทรานส์แพเร้นท์บริดจ์ไม่จำเป็นต้องมีระบบบริหารเครือข่าย บริดจ์สามารถเรียนรู้ได้ด้วยตัวเองและสามารถปรับตัวให้เข้ากับระบบเครือข่ายนั้น ๆ ได้โดยอัตโนมัติ ส่วน SRB จะต้องให้ผู้บริหารเครือข่ายทำการติดตั้งหมายเลขระบบเครือข่ายและหมายเลขบริดจ์ด้วยตนเองทั้งหมด ความผิดพลาดเช่น ระบบเครือข่ายหรือบริดจ์ใช้หมายเลขซ้ำกันนั้นตรวจสอบได้ยากมาก ซึ่งอาจจะทำให้เกิดการรวบซ้ำของเฟรมข้อมูลได้ นอกจากนี้การติดต่อของระบบเครือข่าย 2 แห่งที่เคยเชื่อมต่อกันในอดีตนั้น สำหรับทรานส์แพเร้นท์บริดจ์แล้วไม่มีสิ่งใดต้องทำยกเว้นการเชื่อมต่อสายเข้าด้วยกันเท่านั้น ส่วนการใช้ SRB อาจมีความจำเป็นจะต้องเปลี่ยนหมายเลขของระบบเครือข่ายหลาย ๆ ระบบที่ใช้หมายเลขซ้ำกัน

ข้อดีประการหนึ่งของ SRB อยู่ที่ระบบนี้สามารถใช้เส้นทางการส่งข้อมูลที่ดีที่สุดได้ (ในทางทฤษฎี) ในขณะที่ทรานส์แพเร้นท์บริดจ์มีข้อจำกัดที่เกิดขึ้นจากการใช้สแปนนิ่งทรี นอกจากนี้ SRB สามารถเลือกใช้งานบริดจ์คู่ขนานระหว่างระบบเครือข่ายได้อย่างเหมาะสม แต่บริดจ์ที่ใช้งานจริงจะฉลาดมากพอที่จะแบ่งงานกันทำให้ได้ตามที่กล่าวไว้หรือไม่นั้น ยังไม่มีการพิสูจน์

การค้นหาตำแหน่งของผู้รับโดยวิธีการเรียนรู้ย้อนหลังในทรานส์แพเร้นท์บริดจ์มีข้อจำกัดตรงที่บริดจ์จะต้องรอจนกระทั่งเฟรมที่ส่งมาจากสถานีต่าง ๆ นั้นมาถึงจึงจะสามารถเรียนรู้จากข้อมูลเหล่านั้นได้ ส่วนการค้นหาเฟรมใน SRB มีปัญหาเกี่ยวกับการเพิ่มจำนวนของเฟรมค้นหาอย่างรวดเร็วมาก โดยเฉพาะในระบบที่มีจำนวนระบบเครือข่ายมากและใช้บริดจ์คู่เชื่อมต่อระหว่างเครือข่ายเข้าด้วยกัน

การจัดการความผิดพลาดของบริดจ์ทั้งสองแบบมีวิธีการที่แตกต่างกัน ในแบบแรกบริดจ์สามารถเรียนรู้เกี่ยวกับบริดจ์และระบบเครือข่ายต่าง ๆ ที่ทำงานผิดพลาดหรือการเปลี่ยนแปลงรูปแบบเครือข่ายได้อย่างรวดเร็วและเป็นไปอย่างอัตโนมัติ ด้วยการดักฟังสัญญาณที่ส่งออกมาจากอุปกรณ์เหล่านั้นเพียงอย่างเดียว โฮสต์จะไม่ต้องเข้ามายุ่งเกี่ยวด้วยเลย

ส่วนการจัดการความผิดพลาดในระบบของ SRB มีวิธีการที่แตกต่างออกไปอย่างสิ้นเชิง เมื่อบริดจ์หยุดทำงาน สเตชันที่เลือกเส้นทางเดินข้อมูลผ่านอุปกรณ์ตัวนั้นจะพบว่าเฟรมที่ส่งออกไปไม่ได้รับการตอบรับกลับมาเลย สเตชันนั้นอาจส่งข้อมูลซ้ำแล้วซ้ำอีก สุดท้ายสเตชันนั้นจะทราบว่าเกิดมีอุปกรณ์บางอย่างทำงานผิดปกติ แต่ก็ยังไม่ทราบว่ามีปัญหาเกิดขึ้นที่สเตชันปลายทางหรืออยู่ในเส้นทางปัจจุบัน การหาคำตอบด้วยการส่งเฟรมค้นหาข้อมูลออกไปใหม่จะทำให้ทราบว่าสเตชันปลายทางยังคงทำงานอยู่หรือไม่ อย่างไรก็ตาม ในกรณีที่บริดจ์หลักเสียหายหรือหยุดทำงานจะทำให้โฮสต์จำนวนมากเสียเวลาไปกับการรอคอยและส่งเฟรมค้นหาออกไปจนกว่าปัญหานั้นจะได้รับการแก้ไขแม้ว่าจะมีเส้นทางอื่นอยู่ก็ตาม การชำรุดของอุปกรณ์เป็นจุดอ่อนหลักของการเชื่อมต่อแบบต่อเนื่องทั้งหมด

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

คุณลักษณะสำคัญของสวิตช์คือมีอิเล็กทรอนิกส์สวิตช์ความเร็วสูงทำหน้าที่ส่งเฟรมจากพอร์ตต้นทางสู่ปลายทางโดยไม่รบกวนพอร์ตอื่น ลักษณะนี้ต่างจากบริดจ์หรือเราเตอร์ซึ่งเฟรมถูกส่งจากพอร์ตหนึ่งไปยังอีกพอร์ตหนึ่งโดยอาศัยการประมวลผลของไมโครโปรเซสเซอร์

3.1.3 สวิตช์ (Switch)

สวิตช์เป็นอุปกรณ์ในระบบเครือข่ายที่ออกแบบมาเพื่อแยกระบบเครือข่ายออกเป็นส่วนย่อย ๆ เพื่อเพิ่มประสิทธิภาพของระบบเครือข่ายและทำให้การควบคุมระบบเครือข่ายทำได้ดีขึ้น โดยแต่ละพอร์ตของสวิตช์จะเป็นเซกเมนต์หนึ่งของระบบเครือข่าย ข้อมูลที่ส่งในเซกเมนต์เดียวกันจะไม่ถูกส่งไปยังเซกเมนต์อื่น เป็นการช่วยลดปัญหาความคับคั่งของข้อมูลได้

สวิตช์จะมีลักษณะคล้ายกับบริดจ์ในการแบ่งระบบเครือข่ายออกเป็นส่วนย่อย ๆ ในกรณีที่มีการส่งข้อมูลข้ามเซกเมนต์ สวิตช์จะส่งเฟรมไปยังพอร์ตที่สเตชันปลายทางอยู่เท่านั้น อีกทั้งสวิตช์ยังสามารถส่งข้อมูลระหว่างเซกเมนต์ได้พร้อม ๆ กัน โดยไม่เกิดปัญหาการชนกันของข้อมูล (Collision) และสามารถส่งข้อมูลได้ในแบบสองทาง (Full-duplex)

สวิตช์จะทำงานที่ชั้น 2 (Data link Layer) ของ OSI Reference Model (โดยในปัจจุบัน สวิตช์สามารถทำงานได้ที่ชั้นที่ 2 ชั้นที่ 3 และชั้นที่ 4 แล้ว) ดังนั้นสวิตช์จะรับและส่งเฟรมข้อมูลตาม MAC Address ของสเตชันที่ต่ออยู่ที่พอร์ตของสวิตช์ โดยการต่อเชื่อมกับสวิตช์แบ่งออกเป็น 2 แบบคือ Segment switch และ Port switch

Segment Switch จะรองรับทราฟฟิกของสเตชันในเซกเมนต์ในแต่ละพอร์ต รวมทั้งเซกเมนต์ที่มีสเตชันเดียวด้วย ซึ่งจะเชื่อมต่อจากสเตชันมาที่พอร์ตของสวิตช์โดยตรง ซึ่งทำให้ผู้ออกแบบระบบเครือข่ายสามารถจัดให้สเตชั่นที่ต้องมีการส่งข้อมูลกันมาก ๆ หรือบ่อย ๆ อยู่ในเซกเมนต์เดียวกัน และสามารถจัดให้เซิร์ฟเวอร์ที่ให้บริการ

Port Switch หรือเรียกอีกอย่างหนึ่งว่า Switch Hub เป็นการใช้งานในลักษณะ 1 พอร์ตต่อ 1 สเตชัน โดยใช้งานแทนที่ฮับ

3.1.3.1 คัททรูสวิตช์ (Cut-Through Switching)

โดยการทำงานปกติของสวิตช์ จะทำการรับเฟรมเข้ามาก่อน แล้วจึงส่งเฟรมนั้นไปยังพอร์ตของสเตชันปลายทาง (Store & Forward) สวิตช์แบบคัททรูจะลดการหน่วงเวลาในขั้นตอนนี้ โดยเมื่อสวิตช์ได้รับข้อมูลเฟรมเพียงพอที่จะกำหนดสเตชั่นเป้าหมายได้แล้ว ก็จะเริ่มต้นการส่งข้อมูลทันทีโดยไม่ต้องรอให้ได้รับเฟรมทั้งหมด

การใช้งานสวิตช์แบบคัททรูอาจจะทำให้เกิดปัญหาการส่งเฟรมที่มีข้อผิดพลาดได้ ดังนั้นจึงควรกำหนดให้สวิตช์ทำการรับเฟรมมาจำนวนหนึ่งก่อน แล้วจึงเริ่มการส่งเฟรม เพื่อให้แน่ใจว่า เฟรมนั้นเป็นเฟรมที่ไม่มีข้อผิดพลาด

3.1.3.2 ชนิดของสวิตช์

Crossbar Switch เป็นสวิตช์ที่พัฒนาขึ้นในยุคแรก ๆ โดยทุกอินพุตจะต่อเข้ากับทุก ๆ เอาต์พุต โดยจะมีบัฟเฟอร์ของอินพุตที่ใช้ในการพักข้อมูลเมื่อพอร์ตเอาต์พุตกำลังใช้งานอยู่

Shared-memory Switch สวิตช์ชนิดนี้จะเก็บข้อมูลที่รับเข้าไว้ในหน่วยความจำและส่งออกไปยังพอร์ตของสเตชั่นปลายทาง ข้อมูลจะเข้าและออกระหว่างพอร์ตกับหน่วยความจำโดยตรง วิธีการนี้มีข้อเสียคือ เกิดความล่าช้าในการเก็บข้อมูลลงในหน่วยความจำ

High-speed bus Switch ข้อมูลที่เข้ามาที่พอร์ตจะส่งผ่านบัสและส่งออกไปยังพอร์ตที่สเตชั่นปลายทางเชื่อมต่ออยู่ บัสที่ใช้จะเป็นบัสความเร็วสูง โดยใช้เทคนิค TDM ในการให้บริการกับพอร์ตต่าง ๆ ซึ่งจะต้องมีบัฟเฟอร์ที่ใช้ในการพักข้อมูลไว้ชั่วคราว

สวิตช์แบบ High-speed bus เป็นชนิดที่มีการนำมาใช้มากที่สุด เช่น สวิตช์รุ่น Catalyst 3000 ของซิสโก้ (Cisco) ที่มีพอร์ต 10Base-T 16 พอร์ตและรองรับการเชื่อมต่อแบบฟาสต์อีเทอร์เน็ต (Fast Ethernet), ATM หรือแวน (WAN) จะใช้บัสความเร็ว 480 Mbps และมีบัฟเฟอร์ขนาด 256 K โดยใช้ชิปโพรเซสเซอร์ Intel i960 ในการควบคุมการเข้าถึงบัสของแต่ละพอร์ต

3.1.3.2.1 สวิตช์เลเยอร์ที่ 3 (Layer 3 Switch)

สวิตช์เลเยอร์ที่ 3 หรือ L3 Switch คือสวิตช์ที่ทำงานในระดับชั้นที่ 3 (Network Layer) ดังนั้นการเลือกเส้นทางการส่งข้อมูลของสวิตช์ L3 จึงต้องอาศัยข้อมูลที่อยู่ในแพ็กเก็ตของชั้นที่ 3 เช่นเดียวกับเราเตอร์ นอกจากนี้ยังต้องทำหน้าที่อื่น ๆ ที่กำหนดในชั้นที่ 3 ด้วย เช่น การตรวจสอบความถูกต้องของข้อมูลโดยการ checksum การตรวจสอบการหมดอายุของแพ็กเก็ต (TTL) การรองรับโพรโตคอลการจัดการต่าง ๆ ของชั้นที่ 3 และระบบควบคุมการปลอดภัย

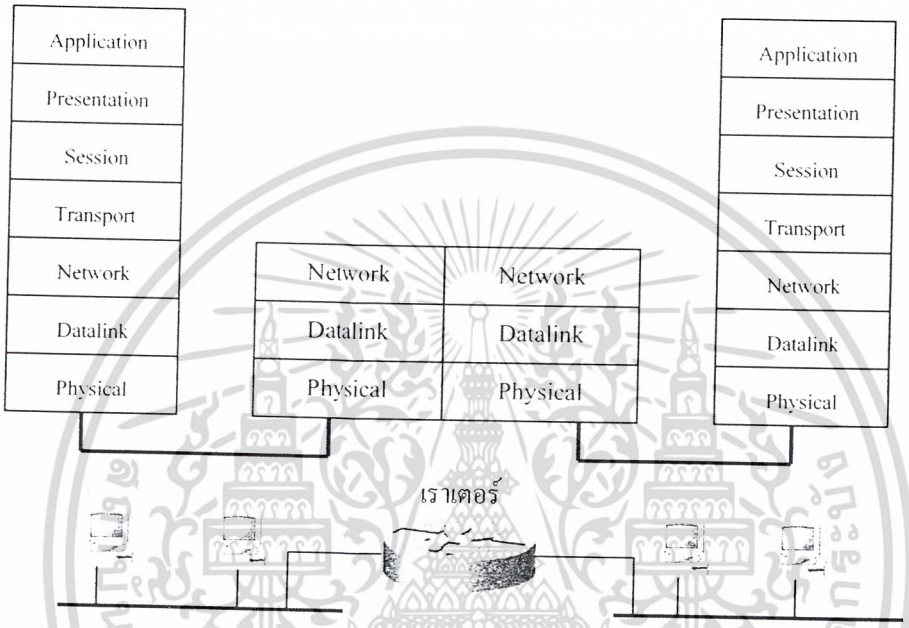
Characteristic	**Layer 3 Switch**	**Router**
LAN Protocol (IP, IPX, Apple Talk)	Yes	Yes
Subnet definition	Layer 2 Switch domain	Port
Forwarding architecture	Hardware	Software (ASIC)
Management	SNMP, RMON	SNMP (RMON)
WAN support	No	Yes
Price	Low	High

ASIC – Application Specific Integrated Circuit

ตารางที่ 3-2 แสดงตารางการเปรียบเทียบระหว่างสวิตช์ชั้นที่ 2 กับสวิตช์ชั้นที่ 3

3.1.4 เราเตอร์

เราเตอร์เป็นอุปกรณ์ที่ทำงานวนระดับชั้นเน็ตเวิร์กตามรูปที่ 3-10 เราเตอร์ทำงานร่วมกับฮาร์ดแวร์ในระดับดาต้าลิงค์ได้หลายรูปแบบ หน้าที่ของเราเตอร์คือจัดแบ่งเครือข่ายและเลือกเส้นทางที่เหมาะสมเพื่อนำส่งแพ็กเก็ต เราเตอร์จะป้องกันการบรอดคาสต์แพ็กเก็ตจากเครือข่ายหนึ่งไม่ให้ข้ามมาอีกเครือข่ายหนึ่ง



รูปที่ 3-10 แบบจำลองการทำงานของเราเตอร์

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

ซอฟต์แวร์ในเราเตอร์จะรับแพ็กเก็ตและเก็บเข้าบัฟเฟอร์ก่อนนำไปประมวลผล วิธีนี้ต่างจากสวิตช์ซึ่งอาศัยฮาร์ดแวร์ตรวจสอบแอดเดรสปลายทางของแพ็กเก็ตและ “สวิตช์” แพ็กเก็ตนั้นจากพอร์ตหนึ่งไปสู่อีกพอร์ตหนึ่ง

บทที่ 4
ทีซีพี/ไอพีและอินเทอร์เน็ต

ทีซีพี/ไอพีเป็นโพรโตคอลที่ได้รับการออกแบบให้เป็นอิสระจากชนิดคอมพิวเตอร์และระบบปฏิบัติการ ตัวโพรโตคอลมีความเชื่อถือได้สูงและสามารถปรับเปลี่ยนการทำงานตามสภาพเครือข่ายได้ในกรณีที่บางเส้นทางชำรุด

4.1 การเชื่อมโยงเครือข่าย

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

เครือข่ายยุคเริ่มต้นมีสถานีที่ใช้ฮาร์ดแวร์ประเภทเดียวกันและสามารถทำงานร่วมกันได้อย่างกลมกลืน ต่อมาเมื่อเทคโนโลยีเครือข่ายเพิ่มขึ้นเช่น อีเทอร์เน็ตและโทเค็นริง ปัญหาที่เกิดขึ้นคือจะเชื่อมเครือข่ายต่างเทคโนโลยีเข้าด้วยกันได้อย่างไรโดยไม่จำกัดว่าคอมพิวเตอร์จะอยู่ในเครือข่ายเดียวกัน หรือต่างเครือข่ายกัน

4.2 ความหมายของโพรโตคอล

การเชื่อมโยงเครือข่ายต่างฮาร์ดแวร์จำเป็นต้องกำหนดข้อตกลงร่วม หรือ โพรโตคอล (Protocol) เพื่อให้คอมพิวเตอร์สื่อสารกันตามข้อกำหนด ทีซีพี/ไอพีจัดเป็นโพรโตคอลหนึ่งที่ออกแบบมาเพื่อแก้ไขปัญหาที่เกิดขึ้น

4.3 สถาปัตยกรรมของอินเทอร์เน็ตและความหมายของเราเตอร์

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

เราเตอร์เป็นอุปกรณ์เครือข่ายซึ่งมีหน้าที่เลือกเส้นทางเพื่อนำส่งข้อมูลในรูปแพ็กเก็ต หากเปรียบเทียบกับการส่งจดหมายทางไปรษณีย์แล้ว เราเตอร์ทำหน้าที่เสมือนที่ทำการไปรษณีย์ พนักงานไปรษณีย์จะพิจารณาจุดหมายปลายทางของจดหมายและเลือกเส้นทางส่งจดหมายไปยังที่ทำการไปรษณีย์ถัดไปจนกว่าจะถึงมือผู้รับ

เครือข่ายที่ต่อเชื่อมกันด้วยเราเตอร์เช่นรูปที่ 4-2 เป็นลักษณะทั่วไปของอินเทอร์เน็ตแต่ละเครือ - ข่ายนิยมเขียนแทนด้วยรูป กลุ่มเมฆเครือข่าย (Network Cloud) เพื่อแสดงเครือข่ายโดยไม่กล่าวถึงโทโปโล ยีและการจัดการภายใน

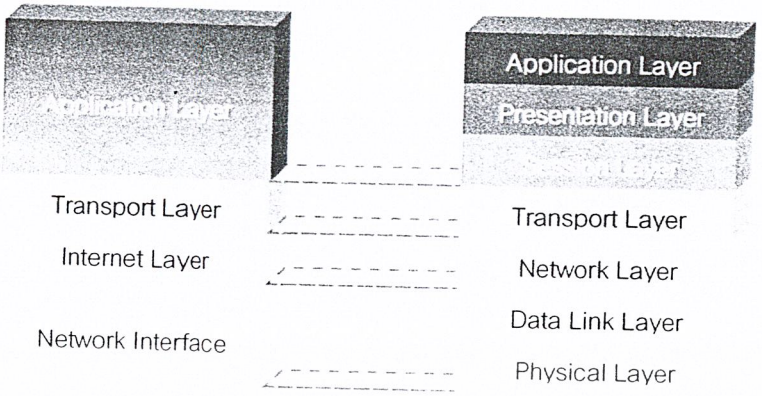
ในยุคต้นของอินเทอร์เน็ตใช้คำว่า เกตเวย์ (Gateway) เพื่อสื่อความหมายถึงอุปกรณ์เชื่อมโยง เครือข่าย แต่ในปัจจุบันนี้เราเตอร์และเกตเวย์เป็นอุปกรณ์ซึ่งมีหน้าที่แตกต่างกัน คำว่าเกตเวย์ที่ใช้เดิมนั้น ในปัจจุบันนิยมใช้คำว่าเราเตอร์แทน หน้าที่หลักของเราเตอร์เชื่อมเครือข่ายที่ใช้โพรโตคอลเดียวกันและ เลือกเส้นทางส่งข้อมูล ขณะที่เกตเวย์หมายถึงฮาร์ดแวร์หรือซอฟต์แวร์ซึ่งเป็นตัวแปลงระหว่างระบบสอง ระบบที่มีโพรโตคอล โครงสร้างการจัดข้อมูล หรือสถาปัตยกรรมที่แตกต่างกัน เกตเวย์จะแปลงแพ็กเก็ต จากระบบต้นทางให้อยู่ในรูปแบบของระบบปลายทางเช่นเกตย์เวย์ระหว่างทีซีพี/ไอพีและโพรโตคอล เอส เอ็นเอ (SNA : Systems Network Architecture) หรืออิเล็กทรอนิกส์เมล์เกตเวย์ที่แปลงรูปแบบจดหมายของ โพรโตคอลใดๆ ไปสู่ระบบ x.400 ของโอเอสไอ

4.4 สถาปัตยกรรมทีซีพี/ไอพี

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

ต่อมาได้พัฒนาเป็นเครือข่ายอินเทอร์เน็ต โพรโตคอลนี้เหมาะสำหรับเชื่อมต่อคอมพิวเตอร์ทั้ง ใกล้และไกลเข้าด้วยกัน และมีมาตรฐานรองรับทำให้ผู้ผลิตฮาร์ดแวร์และซอฟต์แวร์ สามารถสร้างอุปกรณ์ และโปรแกรมที่จะรองรับการทำงานของโพรโตคอลนี้ ทำให้เครื่องคอมพิวเตอร์สามารถรับส่งข้อมูลกัน ได้ไม่ว่าจะเป็นเครื่องขนาดเล็กหรือขนาดใหญ่ หรือใช้ระบบปฏิบัติการอะไรก็ตาม ทีซีพี/ไอพี (TCP/IP) เป็นชุดโพรโตคอลที่ประกอบด้วยโพรโตคอลต่างๆ หลายโพรโตคอล แต่ละโพรโตคอลมีคุณลักษณะ และ มีความสามารถในการทำงานแตกต่างกัน โดยที่ในบทนี้ได้กล่าวถึงรายละเอียดและคุณสมบัติของ โพรโตคอลที่สำคัญบางโพรโตคอล

ทีซีพี/ไอพี (TCP/IP หรือ Transmission Control Protocol/Internet Protocol) เป็นโพรโตคอลใน การสื่อสารในระบบอินเทอร์เน็ตและอินทราเน็ต การทำงานของทีซีพี/ไอพีสามารถเปรียบเทียบกับโมเดล อ้างอิงโอเอสไอ (Open System Interconnection Reference Model : OSI) ตามมาตรฐานไอเอสโอ (International Organization for Standardization: ISO) ได้ดังรูปที่ 4-1



รูปที่ 4-1 แสดงการเปรียบเทียบเลเยอร์ของโอเอสไอกับเลเยอร์ของทีซีพี/ไอพี

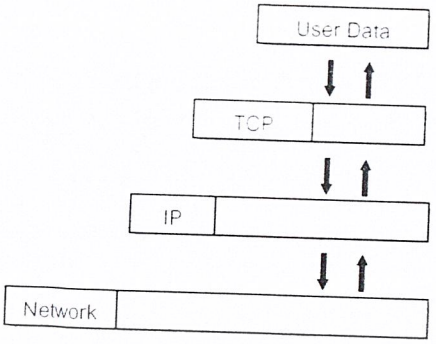
ในแต่ละระดับชั้นของทีซีพี/ไอพีมีการทำงานที่แตกต่างกัน ตั้งแต่การติดต่อกับแอพพลิเคชันจนกระทั่งแปลงเป็นสัญญาณส่งไปตามสายสัญญาณ ซึ่งการทำงานในแต่ละระดับชั้นของทีซีพี/ไอพี มีดังนี้

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



รูปที่ 4-2 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

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

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

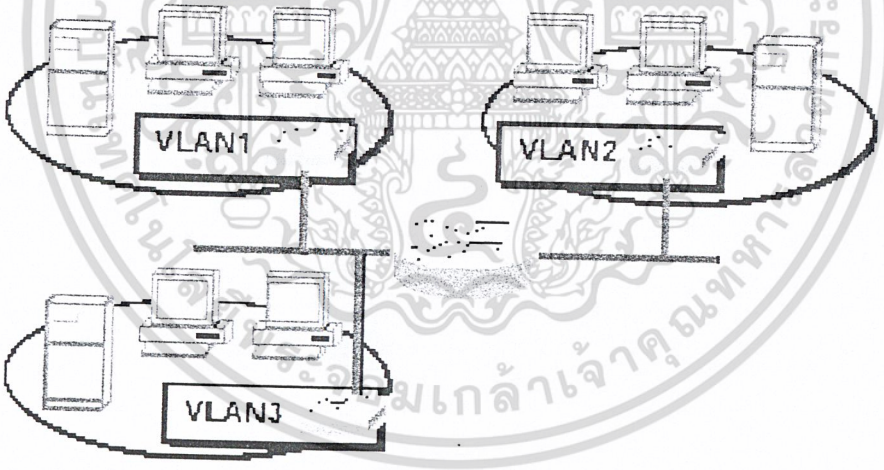
โพรโตคอลดีเอชซีพี (DHCP : Dynamic Host Configuration Protocol)

ดีเอชซีพีบริการติดตั้งค่าแบบไดนามิกให้โฮสต์ในเครือข่าย การทำงานของดีเอชซีพีเป็บแบบไคลเอ็นต์-เซิร์ฟเวอร์ ดีเอชซีพีเซิร์ฟเวอร์จะให้ค่าแบบไม่ตายตัวกับไคลเอ็นต์ที่ร้องขอบริการ เช่นการให้ไอพีแอดเดรส หรือค่าประจำไคลเอ็นต์

บทที่ 5
แลนเสมือน

เมื่อพิจารณาถึงระบบเครือข่ายที่ประกอบด้วยอุปกรณ์ในชั้นที่ 2 เท่านั้น เช่น เซกเมนต์ของอีเทอร์เน็ต, สวิตช์ที่มีหลายพอร์ต หรือเครือข่ายที่ประกอบไปด้วยสวิตช์หลาย ๆ ตัว เครือข่ายแบบนี้เรียกว่า Flat Network Topology เครือข่ายแบบนี้จะมีได้เพียง 1 บอร์ดคาสต์โดเมน (Broadcast Domain) เท่านั้น หมายความว่า เมื่อมีการส่งเฟรมแบบบอร์ดคาสต์จะทำให้ทุก ๆ สเตชันได้รับเฟรมนี้ไป ดังนั้นยิ่งมีจำนวนของอุปกรณ์ (เช่น สวิตช์, ฮับ, สเตชัน) มากขึ้นเท่าใด ก็ยิ่งทำให้เกิดทราฟฟิกขึ้นในเครือข่ายมากขึ้นเท่านั้น จนอาจทำให้เกิดเป็นบอร์ดคาสต์สตอร์ม (Broadcast Storm) ได้

แลนเสมือน หรือวีแลน (Virtual LANs (VLANs)) คือการสร้างเซกเมนต์ของระบบเครือข่ายที่ไม่ขึ้นกับระบบเครือข่ายทางกายภาพ หมายความว่าเราสามารถแบ่งเครือข่ายเราออกเป็นเครือข่ายย่อย ๆ ได้ โดยไม่ขึ้นต่อกัน เมื่อกำหนดวีแลนขึ้นมาแล้ว เราจะถือว่า แต่ละวีแลนเป็น 1 บอร์ดคาสต์โดเมนเป็นเครือข่ายแลนที่ไม่เกี่ยวข้องต่อกัน



รูปที่ 5-1 แสดงเครือข่ายวีแลน

เมื่อกำหนดวีแลนแล้ว สเตชันในวีแลนเดียวกันจะสามารถส่งข้อมูลถึงกันได้ แต่ถ้าเป็นการส่งข้อมูลข้ามเซกเมนต์จะต้องใช้เราเตอร์ในการส่งผ่านข้อมูล ซึ่งผู้ดูแลระบบสามารถกำหนดรายละเอียดของการส่งผ่านข้อมูลระหว่างเซกเมนต์ได้

รูปที่ 5-2 แสดงตัวอย่างการแบ่งวีแลนออกเป็นแผนกงาน

วีแลนมีข้อดีคือ

1. *Segmentation* คือสามารถแบ่งเครือข่ายออกเป็นเครือข่ายย่อยได้ เป็นการแบ่งทราฟฟิกของแต่ละวีแลนออกจากัน
2. *Flexibility* คือ มีความยืดหยุ่นในการเปลี่ยนแปลงสมาชิกที่อยู่ในวีแลนได้ง่าย
3. *Security* คือ เมื่อเราทำการแบ่งวีแลนแล้ว จะถือว่าแต่ละวีแลนเป็น 1 บอร์ดคาสต์โดเมน ทำให้การส่งข้อมูลไม่รั่วไหลออกไปยังวีแลนอื่น ๆ เป็นข้อมูลที่ส่งอยู่ในวีแลนเดียวเท่านั้น

5.1 ประเภทของวีแลน

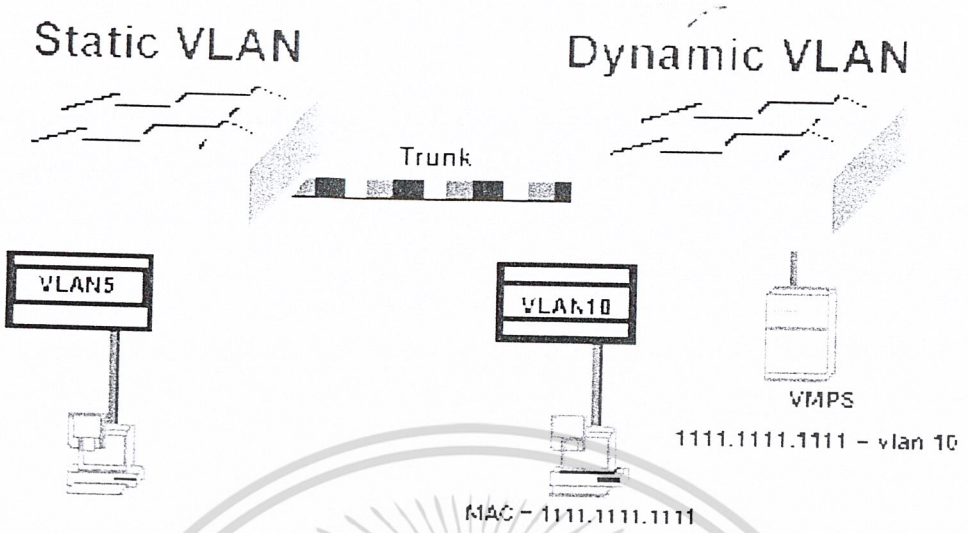
แบ่งออกเป็น 2 ประเภทใหญ่ ๆ คือ

5.1.1 สแตติกวีแลน (Static VLANs)

เป็นการกำหนดวีแลนจากพอร์ตของสวิตช์ว่า ต้องการให้พอร์ตไหนเป็นวีแลนใด เมื่อเรานำอุปกรณ์ไปต่อ ก็จะทำให้อุปกรณ์ชิ้นนั้นเป็นสมาชิกของวีแลนนั้นโดยอัตโนมัติ มีข้อดีคือ กำหนดได้ง่าย และดูแลง่าย (Based on port)

5.1.2 ไดนามิกวีแลน (Dynamic VLANs)

เป็นการกำหนดวีแลนตามค่าแมคแอดเดรสที่ได้กำหนดไว้ โดยจะมีฐานข้อมูล (Database) เก็บไว้ว่าแมคแอดเดรสค่าใดเป็นสมาชิกของวีแลนใด จะมีข้อดีเมื่อเราทำการเคลื่อนย้ายอุปกรณ์ใด ๆ ก็จะยังทำให้อุปกรณ์ตัวนั้นเป็นสมาชิกของวีแลนเดิมอยู่โดยอัตโนมัติ ไม่จำเป็นต้องไปกำหนดค่าใหม่ (Based on MAC Address)



รูปที่ 5-3 แสดงประเภทของวีแลน

นอกจากนี้ เรายังสามารถแบ่งประเภทวีแลนออกได้เป็นอีกหลายแบบ คือ

5.1.2.1 Port-Based & MAC-Based

Port-Based : กำหนดวีแลนตามพอร์ตที่กำหนดไว้ (เหมือน Static VLAN)

MAC-Based : กำหนดวีแลนตามแมคแอดเดรสที่กำหนดไว้ (เหมือน Dynamic VLAN)

5.1.2.2 Protocol-Based & Dynamic-Based

Protocol-Based : กำหนด วีแลน ตามโพรโตคอลที่กำหนดไว้ เช่น

Host X ใช้โพรโตคอล IP ดังนั้น จะเป็นสมาชิกของวีแลน 1

Host Y ใช้โพรโตคอล IPX ดังนั้น จะเป็นสมาชิกของวีแลน 2

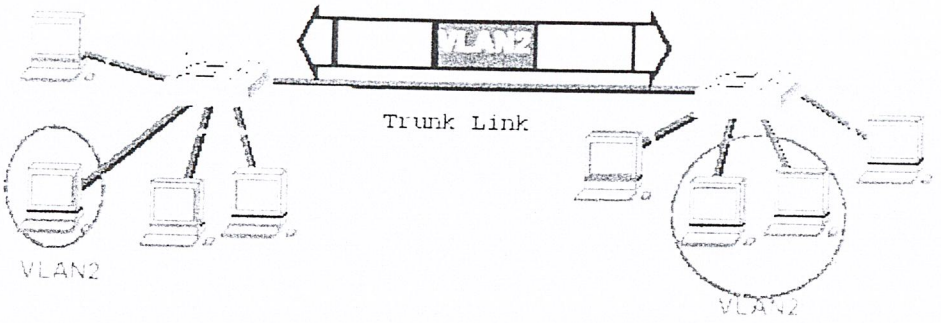
Dynamic-Based : กำหนดวีแลนตาม User Profile ที่กำหนดไว้ โดยเก็บ User Profile ไว้ในฐานข้อมูล เช่น โฮสต์ X ทำการ Log in ตัวโพรไฟล์ (Profile) ของ โฮสต์ X จะเป็นตัวกำหนดให้โฮสต์ X เป็นของ วีแลน 1

5.2 ประเภทของการเชื่อมต่อ

ในการกำหนดวีแลนนั้น บางครั้งอาจมีการกำหนดให้ในวีแลนเดียวกันมีอุปกรณ์ที่เป็นสมาชิกอยู่ในสวิตช์คนละตัวกัน ดังนั้นจึงต้องมีการกำหนดการเชื่อมต่อของวีแลนเพื่อใช้เป็นกฎในการส่งข้อมูลภายในวีแลนเดียวกัน

5.2.1 แอ็กเซสลิงค์ (Access Link) เป็นการเชื่อมต่อที่บอกว่าลิงค์ (Link) นี้เป็นวีแลนใด โดยข้อมูลที่ผ่านจะมีแค่วีแลนเดียว

5.2.2 ทรังค์ลิงค์ (Trunk Link) เป็นการเชื่อมต่อที่ใช้ในการส่งข้อมูลได้หลายๆ วีแลน

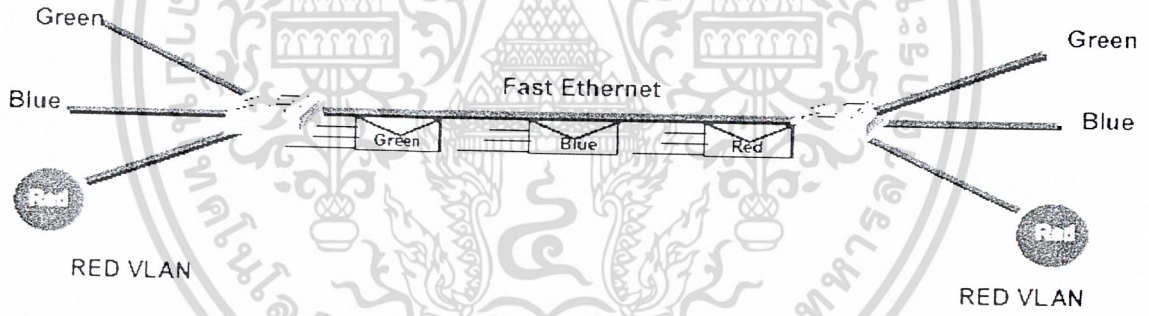


รูปที่ 5-4 แสดงตัวอย่างของทรังค์ลิงค์

5.3 วิธีการระบุถึงวีแลน

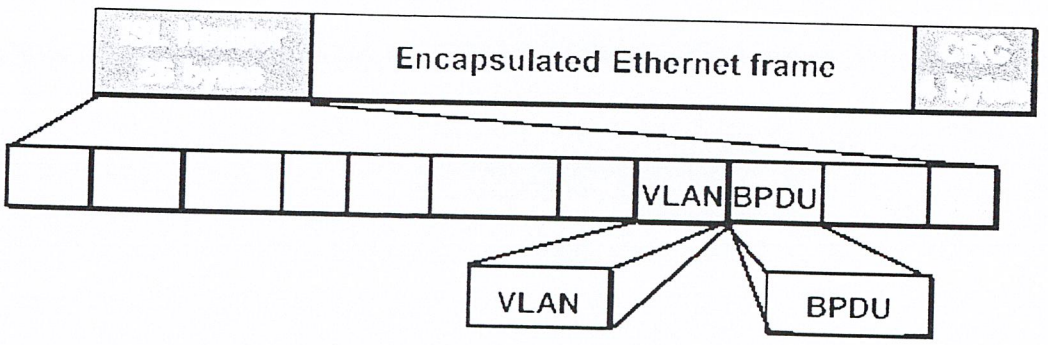
แพ็กเก็ตจะถูกส่งไปตาม Trunk Link โดยบรรจุข้อมูลที่ระบุถึงวีแลนไว้ในส่วนของเฮดเดอร์ (Header) ของแพ็กเก็ต ซึ่งวิธีการระบุนั้นมีอยู่ 2 แบบคือ

- Cisco ISL
- IEEE 802.1Q



รูปที่ 5-5 แสดงเครือข่ายของวีแลนที่ใช้การเชื่อมต่อแบบทรังค์ลิงค์

Cisco ISL



รูปที่ 5-6 แสดงรูปแบบของเฟรมของ ISL

เมื่อได้รับเฟรมอีเทอร์เน็ตมาแล้ว จะทำการ encapsulate ด้วย ISL Header และ CRC ซึ่งสนับสนุนวีแลนได้มากสุดถึง 1024 วีแลน

IEEE 802.1Q

Initial MAC Address	2-Byte TPID 2-Byte TCI	Initial Type/Data	New CRC

รูปที่ 5-7 แสดงรูปแบบของเฟรมของ IEEE 802.1Q

- 2-byte tag protocol identifier (TPID)
- 2-byte tag control information (TCI)

บทที่ 6
ไอพีแอดเดรส

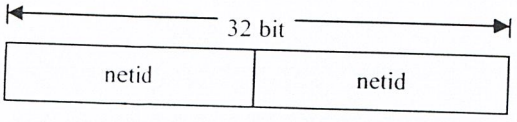
ไอพีแอดเดรสเป็นส่วนประกอบสำคัญอย่างหนึ่งในเครือข่ายทีซีพี/ไอพี ไอพีแอดเดรสเป็นแอดเดรสทางซอฟต์แวร์ประจำสถานีเครือข่ายผู้ออกแบบเครือข่ายจำเป็นต้องศึกษาและเข้าใจรูปแบบของไอพีแอดเดรสโดยละเอียด การจัดแบ่งแอดเดรสออกเป็นคลาส และการจัดแบ่งเครือข่ายย่อยหรือซับเน็ต รวมทั้งสามารถออกแบบซับเน็ตโดยเลือกใช้ไอพีแอดเดรสได้อย่างถูกต้อง เนื้อหาซึ่งจะกล่าวถึงในบทนี้มีดังนี้

6.1 ไอพีแอดเดรส

อุปกรณ์ที่เชื่อมเข้าเครือข่ายและสามารถทำงานตามข้อกำหนดของทีซีพี/ไอพีจะต้องมีแอดเดรสประจำอุปกรณ์นั้น อุปกรณ์ดังกล่าวอาจเป็นโฮสต์ เราเตอร์ เครื่องพิมพ์ หรือแม้กระทั่งอุปกรณ์สำนักงาน เช่นโทรศัพท์หรือเครื่องถ่ายเอกสาร 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 10011110 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 6-1 จำนวนบิตที่ใช้สำหรับเลขเครือข่ายและเลขโฮสต์ขึ้นอยู่กับคลาสที่สังกัด



รูปที่ 6-1 รูปแบบของไอพีแอดเดรส

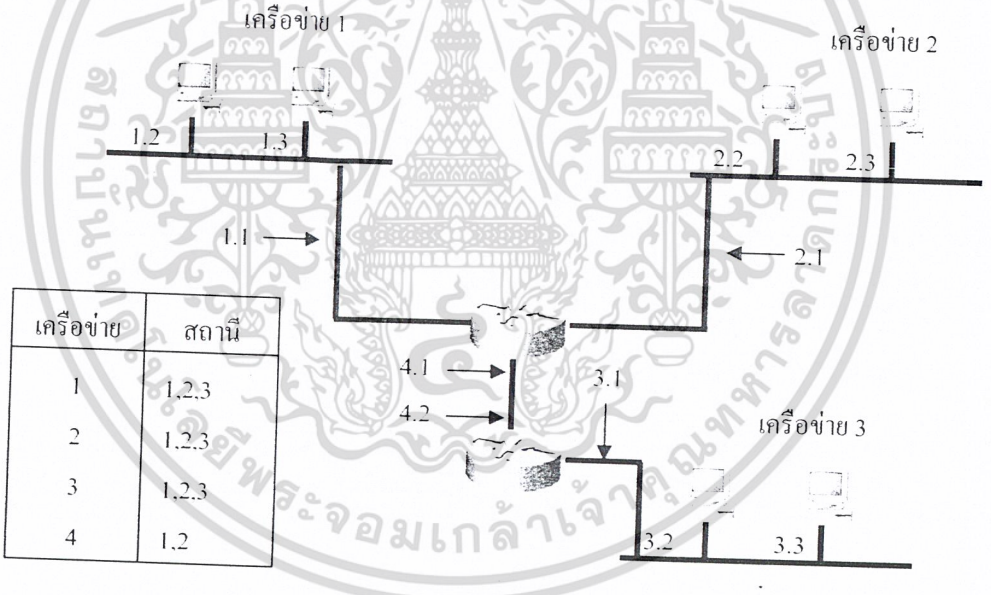
ในปัจจุบันฟิลด์กำหนดเลขเครือข่ายนิยมเรียกว่า พรีฟิกซ์เครือข่าย (Network-Prefix) เพราะทุกโฮสต์ในเครือข่ายจะต้องมีพรีฟิกซ์หรือบิตนำหน้าเหมือนกัน ตัวอย่างเช่นหากมีเลขเครือข่ายจำนวน 16 บิต ก็จะเรียกว่า พรีฟิกซ์ 16 เป็นต้น

6.1.1 ความสำคัญของเลขเครือข่ายและเลขโฮสต์

การจัดแบ่งไอพีแอดเดรสออกเป็นสองส่วนที่ประกอบด้วยเลขเครือข่ายและเลขโฮสต์ก็เพื่อประโยชน์ในการดูแลระบบ เราเตอร์จะอาศัยเลขเครือข่ายเพื่อเลือกเส้นทางส่งแพ็กเก็ตด้วยหลักการต่อไปนี้

โฮสต์ที่มีเลขเครือข่ายชุดเดียวกันย่อมอยู่ภายในเครือข่ายเดียวกัน และสามารถสื่อสารถึงกันด้วยเฟรมดาต้าลิงค์โดยไม่ต้องพึ่งเราเตอร์

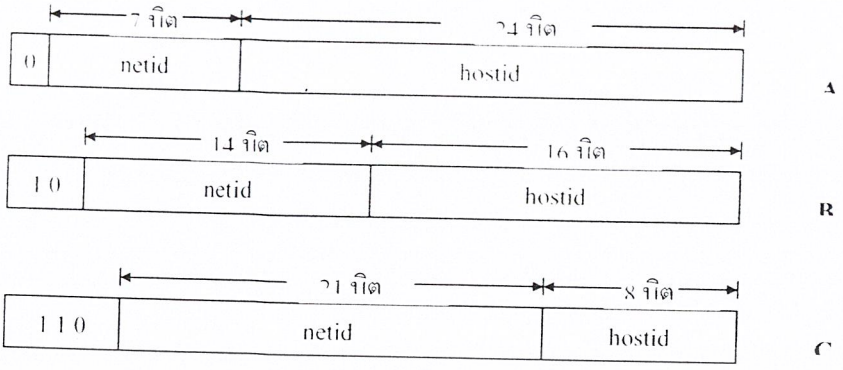
โฮสต์ที่มีเลขเครือข่ายต่างกันจะอยู่ต่างเครือข่ายกัน การสื่อสารระหว่างโฮสต์จะอาศัยเราเตอร์ที่เชื่อมต่อเครือข่ายเป็นผู้นำส่งแพ็กเก็ต เราเตอร์อาจเชื่อมเครือข่ายที่อยู่ติดกันหรือส่งแพ็กเก็ตผ่านเราเตอร์อื่นไปยังปลายทางดังรูป



รูปที่ 6-2 เราเตอร์เชื่อมโยงเครือข่ายที่มีเลขเครือข่ายต่างกัน

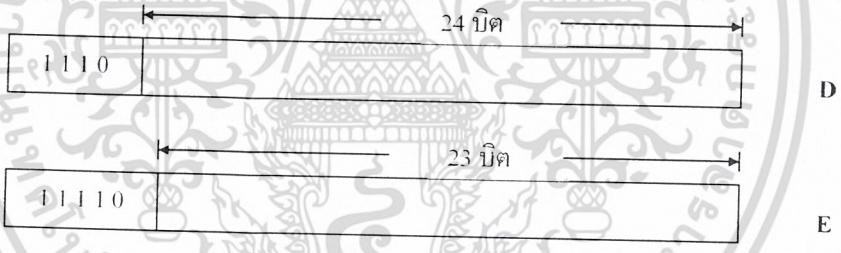
6.1.2 การจัดคลาสเครือข่าย

ไอพีแอดเดรสมีการจัดแบ่งออกเป็นกลุ่มหรือคลาส เครือข่ายที่ใช้งานในปัจจุบันมักสังกัดอยู่ในคลาสใดคลาสหนึ่งคือคลาส A, B หรือ C การแบ่งคลาสอาศัยจำนวนพรีฟิกซ์เครือข่ายที่แตกต่างกันตามรูปที่ 6-3 แต่ละคลาสจึงมีจำนวนเครือข่ายในสังกัดและจำนวนโฮสต์ต่อเครือข่ายไม่เท่ากัน



รูปที่ 6-3 การแบ่งคลาสเครือข่าย

การจัดคลาสตามรูปที่ 6-3 เป็นการจัดแบ่งตามการใช้งานเครือข่ายทั่วไป ในขณะที่ยังมีอีก 2 คลาสซึ่งใช้เพื่อจุดประสงค์เฉพาะได้แก่ คลาส D และ E ดังรูปที่ 6-4 เครือข่ายคลาส D เป็นเครือข่ายแบบมัลติคาสต์ ส่วนคลาส E สงวนไว้ใช้งานหากมีความจำเป็นอื่นใดในอนาคต ทั้งสองคลาสนี้ไม่ได้แบ่งเลขโฮสต์จึงไม่กำหนดจำนวนโฮสต์ไว้



รูปที่ 6-4 การแบ่งคลาส D และ E

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

6.1.3 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

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

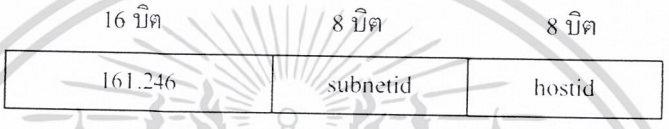
เครือข่ายในคลาส C และ D ไม่มีการจัดแบ่งเลขเครือข่ายและเลขโฮสต์ คลาส D มี 3 บิตแรกเป็น 111 จึงมีแอดเดรสตั้งแต่ 224.0.0.0 ถึง 239.255.255.255 แอดเดรสในคลาสนี้เรียกว่า มัลติคาสต์แอดเดรส (Multicast Address) เนื่องจากใช้ในเครือข่ายมัลติคาสต์

สำหรับคลาส E มีแอดเดรสจาก 240.0.0.0 ถึง 254.255.255.255 ซึ่งสำรองไว้เพื่อความจำเป็นเฉพาะงานในอนาคต

6.2 การแบ่งเครือข่ายย่อย

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

การจัดซับเน็ตใช้วิธีแบ่งบางส่วนของเลขโฮสต์มาใช้เป็นเลขซับเน็ต (SubnetID) เพื่อกำหนดว่าเป็นเครือข่ายย่อยที่เท่าใด ตัวอย่างเช่นเครือข่าย 161.246.0.0 ซึ่งอยู่ในคลาส B อาจใช้ 8 บิตแรกของเลขโฮสต์เป็นเลขซับเน็ต และ 8 บิตที่เหลือใช้สำหรับเลขโฮสต์ดังรูปที่ 6-5



รูปที่ 6-5 ตัวอย่างการแบ่งเครือข่ายย่อยของ 161.246

จำนวนบิตของเลขซับเน็ตเป็นตัวกำหนดจำนวนเครือข่ายย่อย ซับเน็ตขนาด 8 บิตสำหรับเครือข่าย 161.246.0.0 จะมี 254 ซับเน็ต ($2^{subnetid} - 2$) แต่ละซับเน็ตมี 254 โฮสต์ ($2^{hostid} - 2$) ดังตารางที่ 6-1 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

ตารางที่ 6-1 การจัดแบ่งเครือข่าย 161.246 ด้วยซับเน็ต 8 บิต

6.2.1 ซับเน็ตมาสก์

เมื่อผู้วางระบบเลือกขนาดซับเน็ตแล้วจะกำหนดพารามิเตอร์เพื่อใช้บอกให้โฮสต์และเราเตอร์ทราบว่าซับเน็ตที่ใช้งานมีขนาดกี่บิต ค่านี้เรียกว่า ซับเน็ตมาสก์ (Subnet Mask)

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

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

1. นำค่าไอพีแอดเดรส
2. กำหนดบิต "1" และ "0"
3. แปลงเป็นเลขฐานสิบ

8 บิต	8 บิต	8 บิต	8 บิต
161	246	SubnetID	HostID
11111111	11111111	11111111	00000000
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

6.2.2 ดีฟอลต์ซับเน็ตมาสก์ (Default Subnet Mask)

การติดตั้งโฮสต์เข้าเครือข่ายนอกจากจะต้องกำหนดไอพีแอดเดรสแล้วต้องกำหนดค่าซับเน็ตมาสก์ตามที่ผู้ดูแลระบบกำหนดไว้ด้วย ถึงแม้ว่าในบางเครือข่ายเช่นเครือข่ายในคลาส C ซึ่งมีโฮสต์และไม่ได้แบ่งให้มีซับเน็ต ขั้นตอนการติดตั้งโฮสต์ยังจำเป็นต้องใส่ค่าซับเน็ตมาสก์เช่นกัน แต่ค่าซับเน็ตมาสก์นี้เรียกว่า ดีฟอลต์ซับเน็ตมาสก์ (Default Subnet Mask) ดีฟอลต์ซับเน็ตมาสก์ของเครือข่ายคลาส A, B และ C แสดงได้ดังตารางที่ 6-2

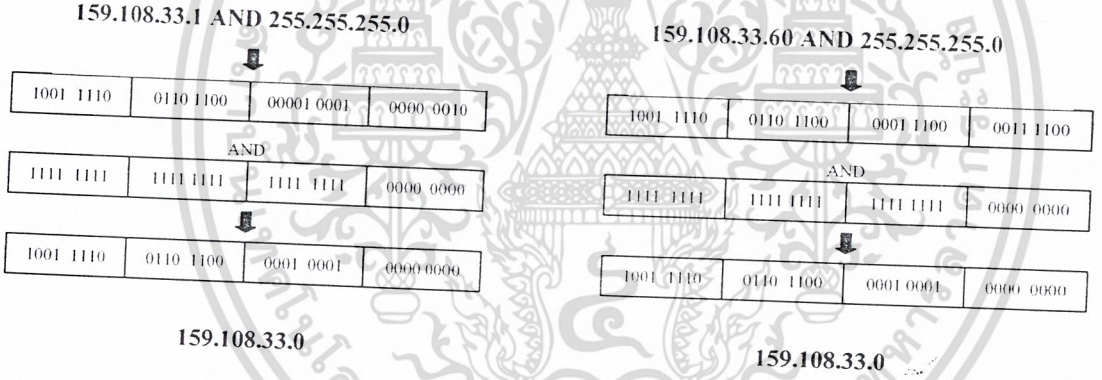
ผู้ดูแลระบบสามารถวางแผนจัดการเครือข่ายโดยเลือกทำซับเน็ตหรือไม่ทำซับเน็ตตามความต้องการ โดยปกติแล้วผู้ดูแลระบบเครือข่ายในคลาส A และ B ไม่สามารถหลีกเลี่ยงการใช้ซับเน็ตได้

คลาส	ดีฟอลต์ซับเน็ตมาสก์	ดีฟอลต์ซับเน็ตมาสก์(ฐาน 2)
A	255.0.0.0	11111111.00000000.00000000.00000000
B	255.255.0.0	11111111. 11111111.00000000.00000000
C	255.255.255.0	11111111.11111111.11111111.00000000

ตารางที่ 6-2 ค่าดีฟอลต์ซับเน็ตมาสก์

6.2.3 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 6-6 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 อยู่ต่างซับเน็ตและจะส่งแพ็กเก็ตไปยังเราเตอร์เพื่อให้เราเตอร์เพื่อให้เราเตอร์นำส่งแพ็กเก็ตต่อไป

บทที่ 7
โพรโตคอลเลือกเส้นทาง (Routing Protocol)

7.1 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 ไอพีแอดเดรสต้นทางและปลายทางในไอพีดาต้าแกรมจะไม่เปลี่ยนแปลง ค่าระหว่างการลำเลียง แต่ฮาร์ดแวร์แอดเดรสจะเปลี่ยนแปลงไปตามเครือข่าย

7.2 ตารางเส้นทาง(Routing Table)

โฮสต์และเราเตอร์จะเก็บแอดเดรสปลายทางสำหรับใช้เป็นเส้นทางส่งดาต้าแกรมไว้ในรูปตารางที่เรียกว่า ตารางเส้นทาง (Routing Table) ค่าในตารางเส้นทางมักประกอบด้วยไอพีแอดเดรสของเครือข่ายปลายทาง และเกตเวย์ซึ่งเป็นทางออกของดาต้าแกรม

ตารางเส้นทางจะบรรจุแอดเดรสหนึ่งซึ่งทำหน้าที่เป็นช่องทางออกไปสู่เครือข่ายใดๆ ที่ไม่ได้ระบุอยู่ในตารางเส้นทาง แอดเดรสนี้เรียกว่า ดีฟอลต์เกตเวย์ (Default Gateway) หรือ ดีฟอลต์เราเตอร์ (Default Router) ซึ่งแทนด้วยแอดเดรส 0.0.0.0 ไอพีจะเลือกเส้นทางโดยตรวจสอบว่าแอดเดรสปลายทางตรงกับ

รายการใดในตาราง แล้วส่งดาต้าแกรมไปยังเกตเวย์ของรายการนั้น หากไม่พบรายการใดในตาราง ก็จะส่งไปยังดีฟอลต์เกตเวย์

7.3 ประเภทของการเลือกเส้นทาง

เมื่อดาต้าแกรมเดินทางออกนอกเครือข่ายที่อาจต้องผ่านเราเตอร์จำนวนมากโดยมีเส้นทางลำเลียงได้หลายเส้นทาง ปัญหาที่สำคัญคือเราเตอร์จะทราบได้อย่างไรว่ามีเส้นทางใดบ้าง และเส้นทางใดที่เป็นเส้นทางที่ดีที่สุด

เราเตอร์จะส่งดาต้าแกรมได้จำเป็นต้องมีแผนที่เครือข่าย ซึ่งก็คือตารางเส้นทาง สำหรับวิธีที่ใช้ในการสร้างตารางเส้นทางนี้มี 2 แบบคือ

7.3.1 การเลือกเส้นทางแบบสแตติก(Static Routing) ตารางเลือกเส้นทางสร้างขึ้นและแก้ไขโดยผู้ดูแลระบบเครือข่าย

7.3.2 การเลือกเส้นทางแบบไดนามิก(Dynamic Routing) ใช้ซอฟต์แวร์คำนวณหาค่าตารางเลือกเส้นทาง ตารางสามารถปรับเปลี่ยนได้หากสภาพเครือข่ายเปลี่ยนไป

การเลือกเส้นทางแบบสแตติก

การเลือกเส้นทางแบบนี้ผู้ดูแลระบบเครือข่ายเป็นผู้พิจารณาและคำนวณหาเส้นทางทั้งหมด โดยใส่ค่าตารางเส้นทางให้กับเราเตอร์ทุกตัว ตารางเลือกเส้นทางนี้จะมีค่าตายตัวตลอดถึงแม้ว่าสภาพของเครือข่ายจะเปลี่ยนไปดังนั้นผู้ดูแลระบบจะต้องคอยตรวจสอบเครือข่ายและปรับเปลี่ยนตารางเส้นทางให้ถูก

รูปที่ 7-1 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)

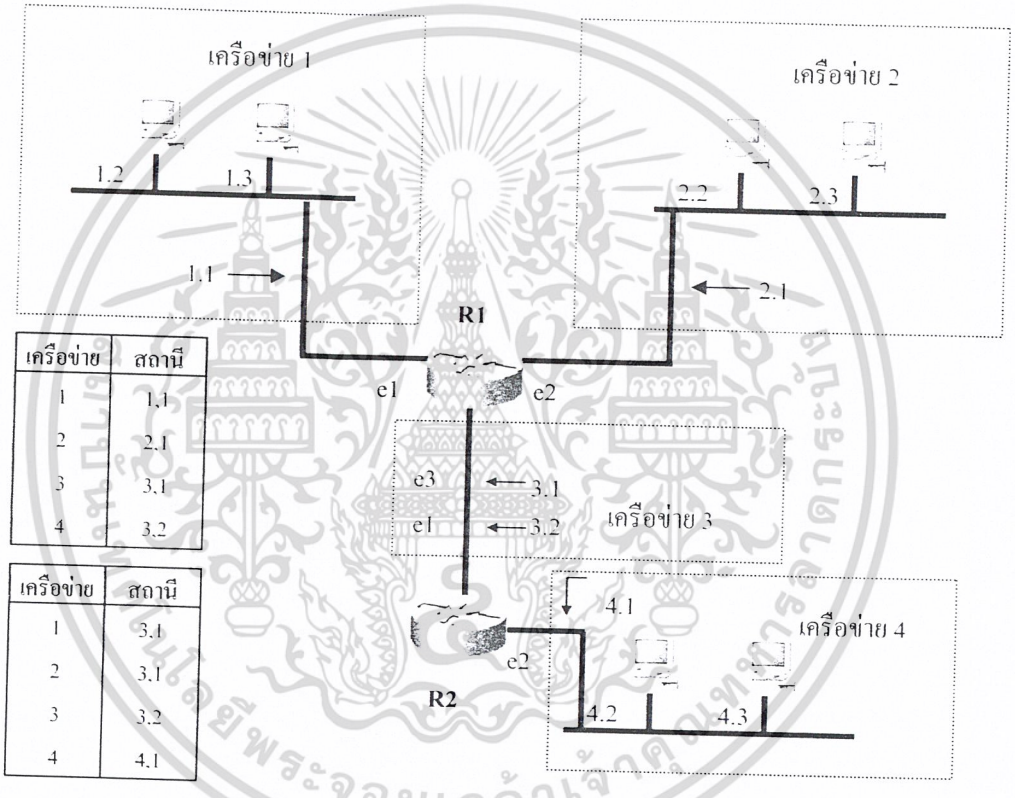
การเลือกเส้นทางแบบสแตติกนิยมใช้กับการเชื่อมโยงแบบจุดต่อจุดระหว่างเราเตอร์สองตัว เช่น เครือข่ายมีทางออกไปสู่ภายนอกหรืออินเทอร์เน็ตเพียงช่องทางเดียว มักจะกำหนดเส้นทางแบบสแตติก การเลือกเส้นทางแบบสแตติกมีข้อดีข้อเสียดังนี้

ข้อดี

1. สะดวกต่อการใช้งานกับเครือข่ายขนาดเล็ก
2. ไม่ต้องใช้ซอฟต์แวร์เลือกเส้นทาง เราเตอร์ไม่จำเป็นต้องมีซีพียูสมรรถนะสูง
3. ประหยัดแบนด์วิดเครือข่ายเนื่องจากไม่ต้องแลกเปลี่ยนข้อมูลระหว่างเราเตอร์
4. สามารถจำกัดให้เข้าถึงได้เฉพาะในเครือข่ายที่ต้องการ

ข้อเสีย

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



รูปที่ 7-1 เครือข่ายการแสดงการเลือกเส้นทางแบบสแตติก

การเลือกเส้นทางแบบไดนามิก

การเลือกเส้นทางแบบนี้จะใช้ซอฟต์แวร์ทำหน้าที่แลกเปลี่ยนข้อมูลการเลือกเส้นทางระหว่างเราเตอร์ด้วยกัน โดยใช้โพรโตคอลเลือกเส้นทาง เราเตอร์จะสร้างตารางเลือกเส้นทางจากสภาพเครือข่ายขณะนั้น หากเครือข่ายมีการเปลี่ยนแปลงตารางเส้นทางก็จะเปลี่ยนแปลงตามไปด้วย

การเลือกเส้นทางแบบไดนามิกต้องอาศัยการแลกเปลี่ยนค่าเส้นทางระหว่างเราเตอร์และใช้ซีพียูในเราเตอร์เพื่อสร้างตารางเส้นทาง เราเตอร์ประเภทนี้จึงมักมีราคาสูงกว่าเราเตอร์ที่มีโพรโตคอลแบบสแตติกอย่างเดียว 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

เลือกใช้เมตริกเพียงประเภทใดประเภทหนึ่ง เช่นอาจใช้เฉพาะระยะทางซึ่งนับจากจำนวนเราเตอร์ที่ต้องส่งผ่านหรือเรียกว่าจำนวนขั้น (Hop Count)

จำนวนขั้นนับจากจำนวนเราเตอร์ที่ดาต้าแกรมต้องเดินผ่านดังตัวอย่างในรูปที่ 7-2 หากดาต้าแกรมไม่ต้องเดินทางผ่านเราเตอร์จำนวนขั้นจะเท่ากับ 0 หากต้องข้ามเราเตอร์หนึ่งตัว จำนวนขั้นจะเพิ่มขึ้นทีละ 1 แต่โพรโตคอลบางชนิดอาจนับจำนวนขั้นตามจำนวนลิงค์ที่ดาต้าแกรมเดินทางผ่านแทนการนับด้วยจำนวนเราเตอร์ หากเครือข่ายอยู่ติดกับเราเตอร์จะมีจำนวนขั้นเท่ากับ 1 แทนที่จะเท่ากับ 0

โพรโตคอลดิสแทนซ์เวกเตอร์และลิงก์สเทต

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

7.4 อาร์ไอพี(RIP)

อาร์ไอพีมีกำเนิดมาจากโพรโตคอลเลือกเส้นทางในระบบเครือข่ายซีร็อกซ์ซึ่งต่อมาได้พัฒนาเป็นอาร์ไอพีและโพรโตคอลอื่นได้แก่ไอพีเอกซ์ (โนเวลล์) และไอจีอาร์พี (ซิสโก้) อาร์ไอพีเป็นโพรโตคอลแบบดิสแทนซ์เวกเตอร์และใช้ระยะทางเป็นค่าเมตริกเพื่อหาเส้นทางที่ดีที่สุดสำหรับการเลือกเส้นทาง

อาร์ไอพีเป็นโพรโตคอลแบบดิสแทนซ์เวกเตอร์ ขั้นตอนพื้นฐานที่อาร์ไอพีใช้คือขั้นตอนวิธีของฟอร์ด/ฟูเกอร์สัน หรือเรียกอีกชื่อว่าเบลแมน-ฟอร์ด เราเตอร์อาร์ไพพีจะเก็บตารางเส้นทางซึ่งประกอบด้วยแอดเดรสเครือข่าย แอดเดรสของเราเตอร์ถัดไปซึ่งเป็นเกตเวย์ไปยังเครือข่ายนั้น และเมตริกประจำเส้นทางซึ่งโดยปกติจะนับตามจำนวนเราเตอร์ระหว่างทาง

7.4.1 การทำงานของอาร์ไอพี

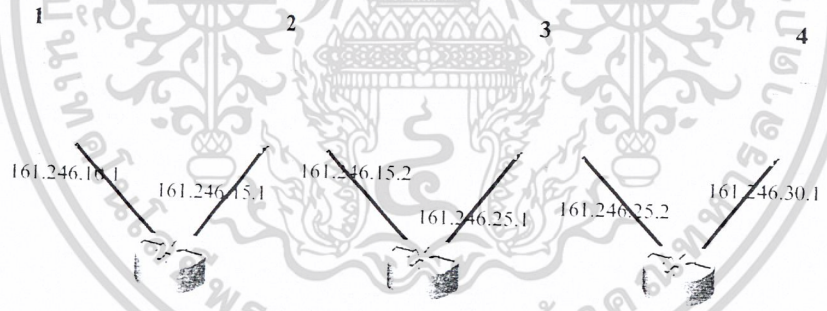
เริ่มแรกเราเตอร์แต่ละตัวจะได้รับการกำหนดแอดเดรสเครือข่ายประจำแต่ละอินเทอร์เฟสของเราเตอร์นั้น ซึ่งเราเตอร์ก็จะสร้างตารางเส้นทางจากแอดเดรสเครือข่ายที่มี แล้วส่งตารางเส้นทางของตัวเองไปให้เราเตอร์ข้างเคียงด้วยการบรอดคาสต์

เมื่อเราเตอร์ได้รับตารางเส้นทางจากเราเตอร์ตัวอื่น ก็จะปรับตารางเส้นทางของตัวเองพร้อมทั้งบรอดคาสต์ตารางของตัวเองไปให้เราเตอร์ข้างเคียงอีกเช่นกัน วิธีนี้ทำให้เราเตอร์แต่ละตัวสามารถคำนวณระยะทางและทิศทางของเครือข่ายอื่นที่เราเตอร์นั้นไม่ได้เชื่อมต่อได้ ในที่สุดเราเตอร์ทุกตัวก็จะทราบแอดเดรสทั้งหมดของเครือข่าย สำหรับการปรับค่าตารางเส้นทางเราเตอร์จะพิจารณาโดย

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

ตัวอย่างการสร้างตารางเส้นทาง

สมมติให้มีเครือข่ายดังรูป 7-2



รูปที่ 7-2 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

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 จะเพิ่มค่าเข้าไปในตารางโดยมองว่า “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
161.246.15.0	0.0.0.0	e1	1
161.246.25.0	0.0.0.0	e2	1

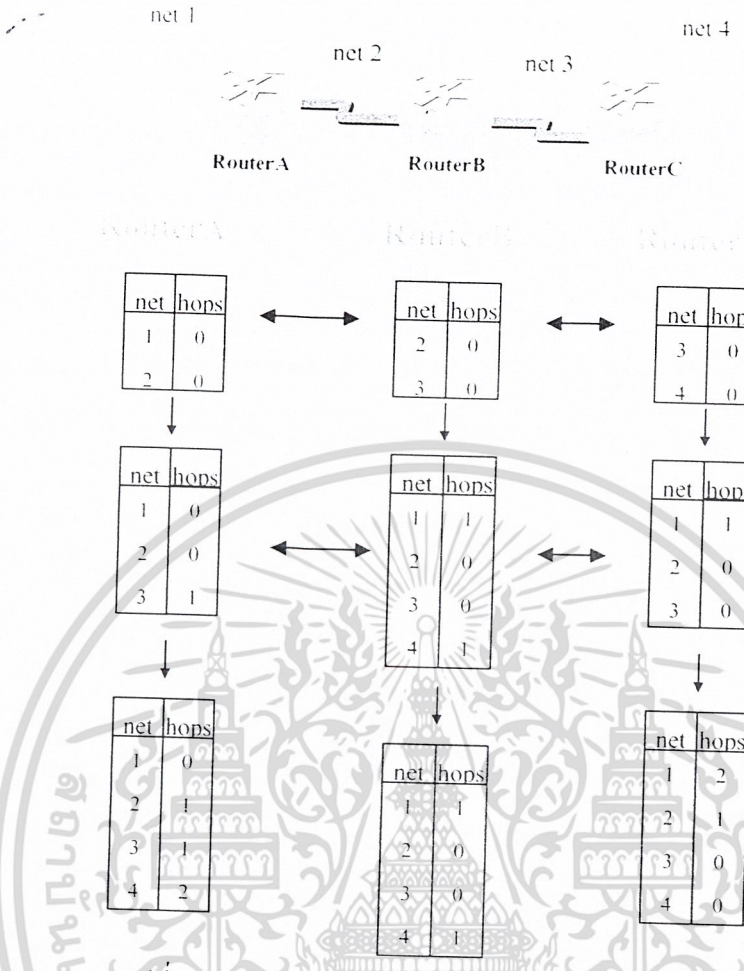
ค่าที่ได้ใหม่จากR1

ถ้าต่อมา R3 บรอดคาสต์เฉพาะเราเตอร์ R2 ก็จะได้รับค่าและเปลี่ยนค่าในตารางเช่นเดียวกัน ดังนั้นตารางเส้นทางของเราเตอร์ R2 จะเป็น

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

ค่าที่ได้ใหม่จาก R3

ตารางของ R2 ในตอนแรกมีเพียงแค่เครือข่ายที่เชื่อมต่อโดยตรงกับเราเตอร์นั้น หลังจากที่ได้รับบรอดคาสต์จาก R1 และ R3 ตารางของ R2 ก็จะมีข้อมูลของเครือข่ายครบทุกเครือข่าย รูปที่ 7-3 แสดงการประกาศและปรับค่าตารางเส้นทางของเราเตอร์ทั้งหมด โดยสมมติให้แต่ละเราเตอร์ประกาศค่าตารางพร้อมกัน



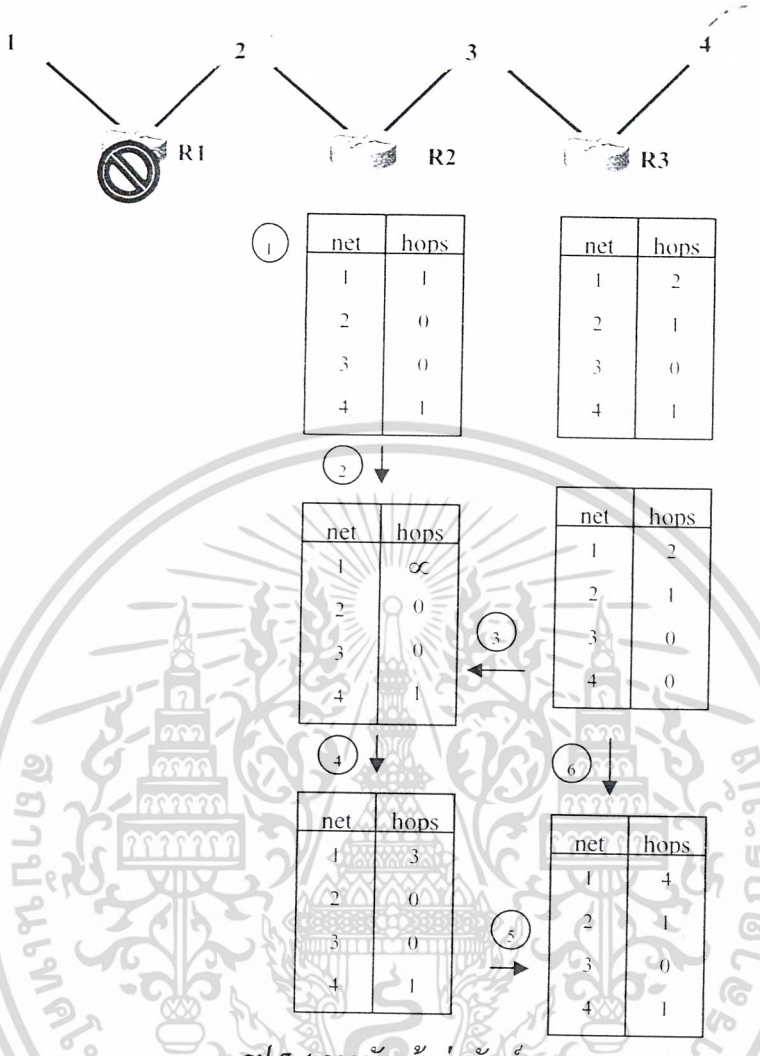
รูปที่ 7-3 การประกาศค่าและปรับค่าและตารางของอาร์ไอพี

7.4.2 การปรับค่าเมื่อเครือข่ายเปลี่ยน

กรณีที่โครงสร้างของเครือข่ายไม่เปลี่ยนแปลง ตารางเส้นทางของอาร์ไอพีก็จะมีเสถียรภาพ ถึงแม้ว่าเราเตอร์จะประกาศตารางออกไปทุกๆ 30 วินาที จึงเป็นการตรวจสอบสภาพเส้นทางและพร้อมที่จะปรับตารางเส้นทางให้เหมาะสมกับสภาพของเครือข่ายถ้าเส้นทางเดิมนั้นมีปัญหา

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

ปัญหาการนับเข้าสู่อนันต์



รูป 7-4 การนับเข้าสู่อนันต์

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 ประกาศตารางเส้นทางตารางของทั้งสองจะปรับค่า

เส้นทางเพื่อไปยังเครือข่าย 1 อย่างไม่มีที่สิ้นสุด ซึ่งอาร์ไอพีได้กำหนดค่าเมตริกไว้ค่าหนึ่งเพื่อแก้ปัญหานี้ ค่าเมตริกนี้ควรเป็นค่าที่ไม่สูงมากเพราะจะได้ไม่ต้องใช้เวลานาน แต่ถ้าน้อยเกินไปก็อาจทำให้เครือข่ายถูกจำกัดให้เล็กลง ซึ่งปกติจะกำหนดไว้ที่ 16 เมื่อเราเตอร์เพิ่มค่าไปถึง 16 ก็จะทราบว่าเส้นทางนั้นเป็นเส้นทางที่ไม่สามารถไปถึงได้ และจะเข้าสู่กระบวนการจับเวลาเพื่อกำจัดเส้นทางออกจากตาราง

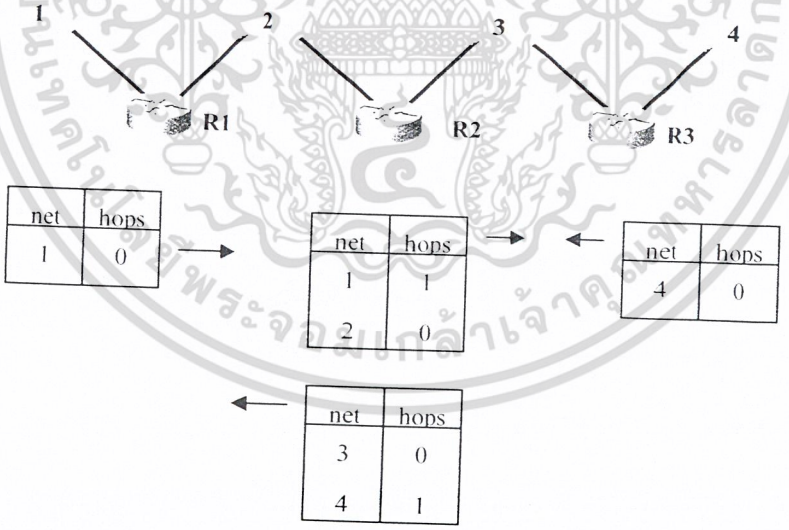
เวลาลู่เข้า

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



รูปที่ 7-5 การประกาศเส้นทางด้วยวิธีสปลิตฮอไรซัน

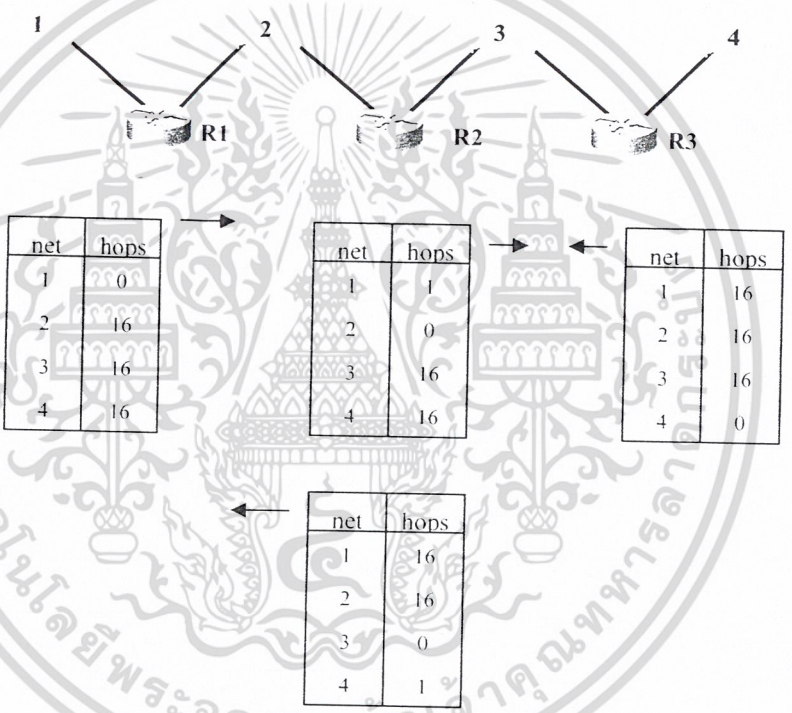
หากเราเตอร์ R1 หยุดทำงานไป ค่าเส้นทางของเครือข่าย 1 ใน R2 จะค่อยๆ ลดลงจนครบ 180 วินาที เส้นทางไปเครือข่าย 1 จะเปลี่ยนเป็นอนันต์ โดยที่ R2 จะไม่ได้รับการประกาศเส้นทางไปเครือข่าย

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



รูปที่ 7-6 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

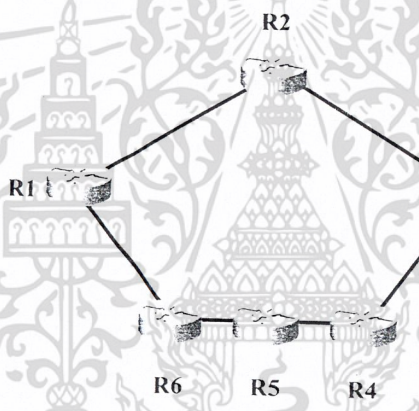
เทคนิคของสปลิตฮอไรซันทั้งสองแบบนี้ไม่สามารถป้องกันปัญหานับเข้าสู่อนันต์ได้ทุกรูปแบบเครือข่าย ช่วยแก้ปัญหาสำหรับเราเตอร์สองตัวที่อยู่ข้างเคียงกันเท่านั้น

ทริกเกอร์อัพเดต

สปลิตฮอไรซันทั้ง 2 วิธีนั้นช่วยแก้ปัญหาการนับเวลาเข้าสู่อนันต์เพื่อลดเวลาลู่เข้า หากแต่ยังต้องใช้เวลาครู่หนึ่งก่อนที่จะตรวจสอบว่าเส้นทางหนึ่งๆ มีปัญหา ทริกเกอร์อัพเดตเป็นเทคนิคเสริมที่ช่วยลดเวลาลู่เข้าของอาร์ไอพี คือกำหนดให้เราเตอร์ประกาศค่าเส้นทางออกไปทันทีที่พบว่าโครงสร้างเครือข่ายมีการเปลี่ยนแปลงโดยไม่ต้องรอให้ครบเวลา 30 วินาที

หากเราเตอร์ทุกตัวใช้ทริกเกอร์อัพเดตก็จะสามารถปรับปรุงตารางได้อย่างรวดเร็ว แต่ก็อาจจะสร้างภาระในเครือข่ายขนาดใหญ่ที่มีเราเตอร์หลายตัว เพราะการเปลี่ยนแปลงใดๆ ก็ย่อมทำให้เราเตอร์ตัวหนึ่งส่งข้อมูลให้เราเตอร์ตัวอื่นประกาศค่าตามกันไปอย่างต่อเนื่องหรือเกิดแพ็กเก็ตบรอดแคสต์เป็นจำนวนมาก

แต่เทคนิคนี้อาจทำให้เกิดเหตุการณ์ที่ไม่คาดฝันขึ้นได้ เช่นในกรณีที่มีเครือข่ายดังรูป



รูปที่ 7-7 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 สามารถใช้งานได้จากเราเตอร์อื่น

เป็นเวลาหนึ่ง โดยช่วงเวลานี้ควรเป็นช่วงเวลาที่มากพอจะให้ทริกเกอร์อัพเดตกระจายไปยังเราเตอร์ทุกตัว ซึ่งการใช้โฮลดาวน์ถึงแม้ว่าจะทำให้ช่วงเวลาลู่เข้านานกว่าแบบไม่ใช้ แต่ทำให้การทำงานของระบบดีกว่าแบบไม่ใช้ด้วย

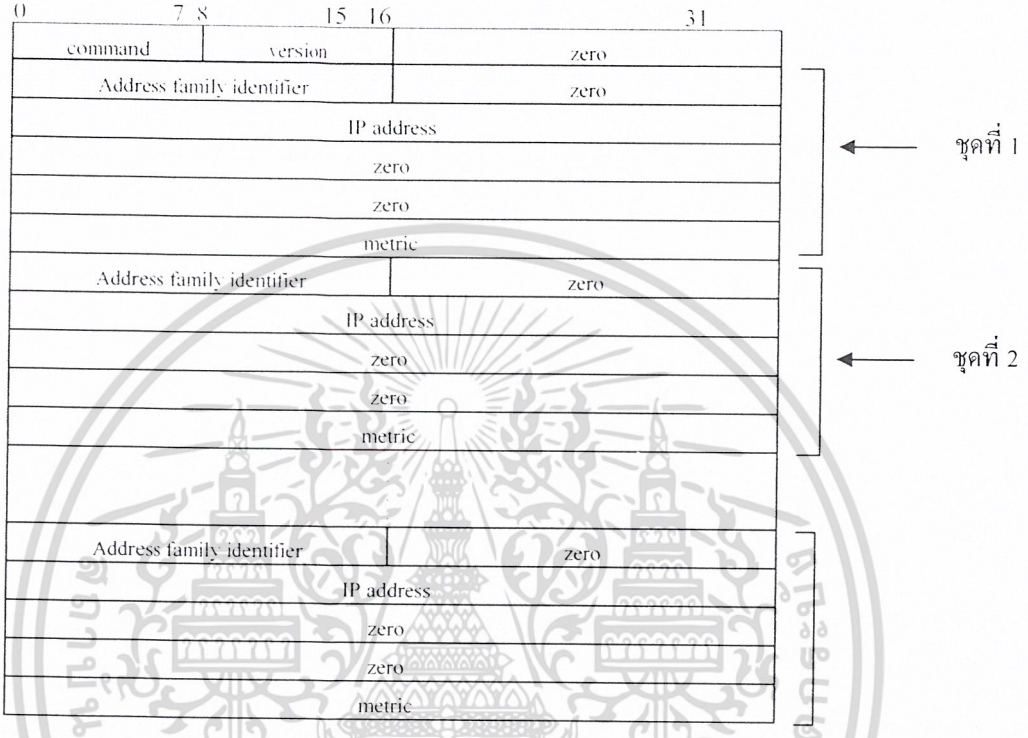
ตัวจับเวลาในอาร์ไอพี

ตัวจับเวลาในอาร์ไอพีจะมีอยู่ 3 ประเภทคือ

1. **ตัวจับเวลาปรับค่า (Update Timer)** เป็นช่วงเวลาที่กำหนดการบรอดคาสต์ของเราเตอร์ ซึ่งปกติจะให้บรอดคาสต์ตารางทุกๆ 30 วินาที แต่ในทางปฏิบัติเราไม่สามารถหลีกเลี่ยงไม่ให้เกิดการบรอดแคสต์พร้อมกัน เช่น ในเราเตอร์ซิสโก้จะสุ่มค่า RIP_JITTER ซึ่งมีค่าได้ไม่เกิน 4.5 วินาที แล้วนำมาลบกับค่า 30 ตัวจับเวลาปรับค่าของเราเตอร์ซิสโก้จึงอยู่ระหว่าง 25.5 ถึง 30 วินาที 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 7-8 ฟอร์แมตของเฟรมอาร์ไอพี

รูปแบบเฟรมในที่นี้แสดงเฉพาะการใช้อาร์ไอพีในเครือข่ายไอพีเท่านั้น หากเป็นเครือข่ายอื่นจะมีรูปแบบแตกต่างออกไป โดยแต่ละฟิลด์มีความหมายดังนี้

- 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 indentifier 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 แต่ได้เพิ่มเติมฟิลด์ใหม่และมีรูปแบบดังรูป7-9

ความหมายของแต่ละฟิลด์

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

Version ขนาด 16 บิต : กำหนดรุ่นของโพรโตคอล ค่าที่ใช้คือ 2

Unused ขนาด 16 บิต : ไม่ได้ใช้งาน

Address family identifier ขนาด 16 บิต : ฟิลด์กำกับชุดแอดเดรสเพื่อแยกแยะชนิดของโพรโตคอลที่อาร์ไอพีทำงานร่วมด้วย สำหรับไอพีใช้ค่าเท่ากับ 2

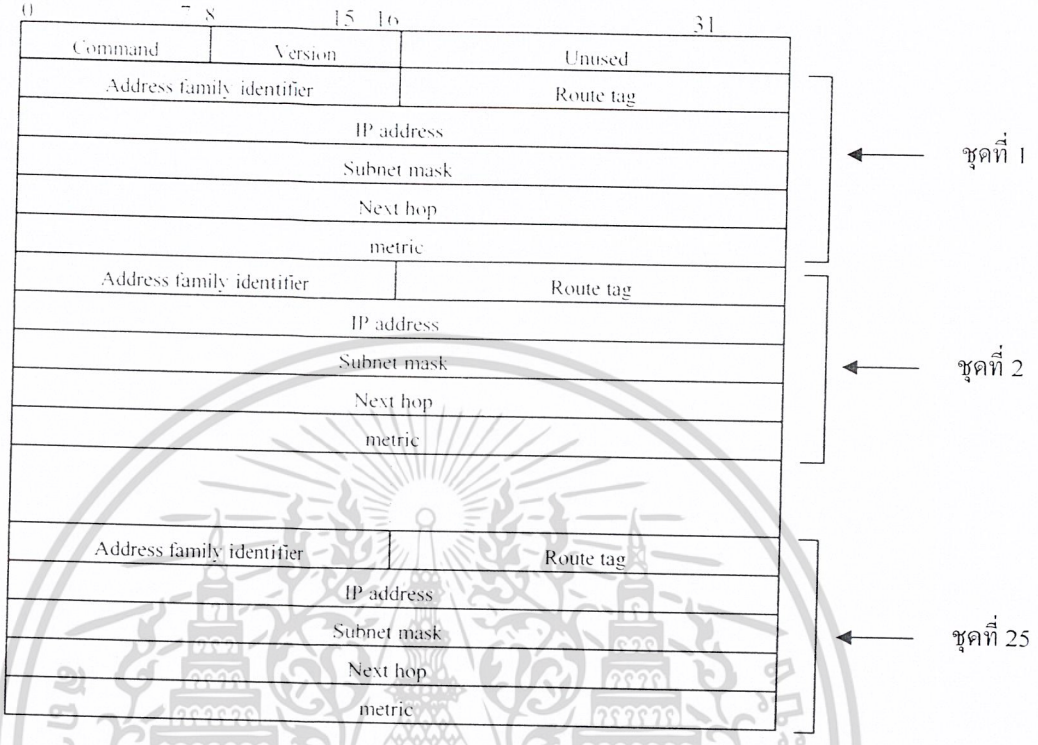
Route tag ขนาด 8 บิต : ใช้แยกความแตกต่างระหว่างการได้มาของค่าเส้นทางว่าเรียนรู้จากภายในหรือภายนอก ค่าที่ใช้อาจเป็นเลขระบบออโตโนมัสสำหรับเส้นทางที่ได้มาจากโพรโตคอลภายนอก

IP Address ขนาด 32 บิต : ไอพีแอดเดรสที่กำหนดเส้นทาง

Subnet Mask ขนาด 32 บิต : ซับเน็ตมาสก์กำกับไอพีแอดเดรส หากมีค่าเป็น 0 หมายถึงไม่ได้กำหนดซับเน็ตมาสก์ให้

Next Hop ขนาด 32 บิต : ไอพีแอดเดรสของเราเตอร์ถัดไปที่ควรจะนำส่งแพ็กเก็ตซึ่งมีแอดเดรสปลายทางตามที่กำหนดด้วยค่าเลือกเส้นทางในเฟรมอาร์ไอพีนี้ ประโยชน์ของฟิลด์นี้ใช้เพื่อลดจำนวนเส้นทางที่แพ็กเก็ตต้องเดินทางผ่านในกรณีที่มีโพรโตคอลเลือกเส้นทางอื่นนอกเหนือไปจากอาร์ไอพีทำงานร่วมกัน หากมีค่า 0.0.0.0 หมายถึงให้ส่งแพ็กเก็ตไปยังเราเตอร์ที่ประกาศค่าเส้นทางนี้มาให้

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

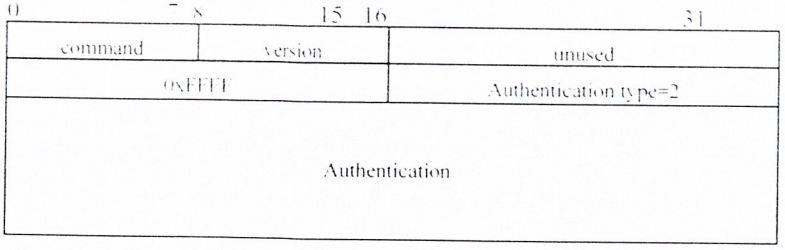


รูปที่ 7-9 ฟอร์แมตของเฟรมอาร์ไอพีเวอร์ชัน 2

ตั้งแต่ฟิลด์ address family identifier ถึงฟิลด์ metric มีค่าซ้ำกันได้ 25 ชุดเช่นเดียวกับเวอร์ชันแรก เพื่อการประกาศเส้นทางไปพร้อมกันได้ 25 เส้นทางในเฟรมเดียว แต่ถ้าฟิลด์ address family identifier ชุดแรกมีค่าเป็น 0xFFFF (ให้ค่านี้มีได้เฉพาะในชุดแรกเท่านั้น) หมายความว่าเป็นเฟรมกำหนดการพิสูจน์ตัวจริง และฟิลด์ถัดมาจะถูกแปลความหมายใหม่ตามรูป 7-10

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



รูปที่ 7-10 เฟรมอาร์ไอพีเวอร์ชัน 2 เมื่อใช้การพิสูจน์ตัวจริง

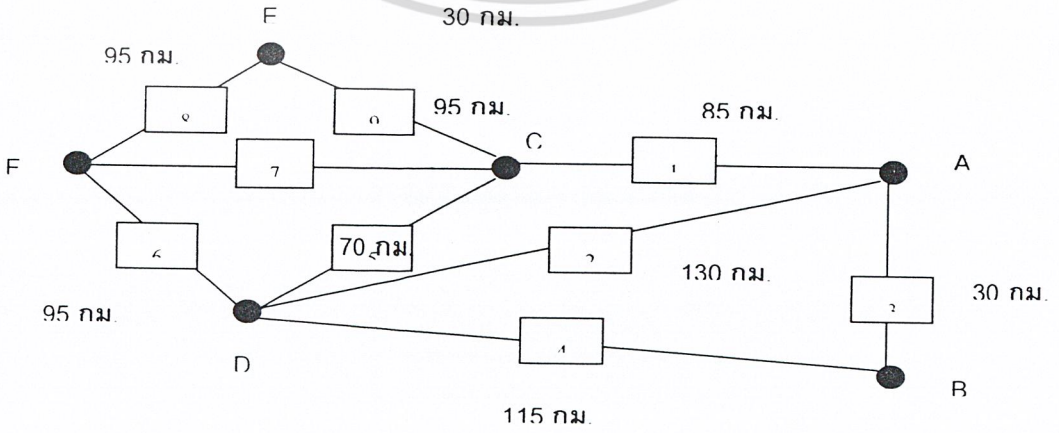
7.5 โอเอสพีเอฟ(OSPF)

โอเอสพีเอฟเป็นโพรโตคอลเกตเวย์ภายในที่นิยมใช้อย่างแพร่หลายเนื่องจากสามารถปรับเปลี่ยนการเลือกเส้นทางตามสภาพเครือข่ายได้รวดเร็วและรองรับการทำงานกับเครือข่ายขนาดใหญ่ได้

หลักการพื้นฐานของโพรโตคอลแบบโอเอสพีเอฟ เป็นโพรโตคอลแบบลิงค์เสตท โดยโอเอสพีเอฟจะทำหน้าที่สร้างแผนที่ให้กับเราเตอร์ 3 ขั้นตอนคือ ขั้นแรกเราเตอร์จะตรวจหาเราเตอร์อื่นที่อยู่ข้างเคียง ขั้นที่สองเราเตอร์จะประกาศค่านี้ออกไปให้เราเตอร์อื่นทุกตัว เมื่อเสร็จสิ้นขั้นตอนนี้ เราเตอร์จะทราบว่ามีเราเตอร์อื่นใดอยู่บ้าง ขั้นสุดท้ายเราเตอร์จะอาศัยข้อมูลจากขั้นที่สองสร้างแผนที่ไปยังเราเตอร์อื่นทุกตัวในเครือข่าย เราเตอร์ทุกตัวต่างก็มีแผนที่ประจำตัวที่ทราบถึงเส้นทางทั้งหมดในเครือข่าย

7.5.1 การทำงานของโอเอสพีเอฟโดยสรุป

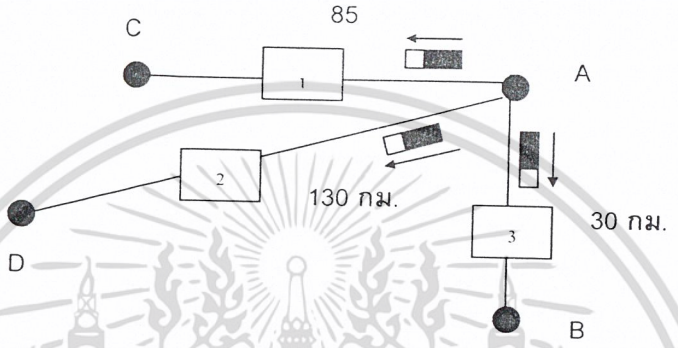
เพื่อให้สามารถศึกษาการทำงานของโอเอสพีเอฟได้อย่างชัดเจนเป็นขั้นตอนจะใช้เส้นทางจากเมืองหนึ่งไปยังอีกเมืองหนึ่งดังรูป 7-11 แต่ละเมืองเปรียบเสมือนเราเตอร์และทางหลวงที่เชื่อมต่อกันถือเป็นสายสื่อสารระหว่างเราเตอร์ เส้นทางในรูปเป็นเพียงเส้นทางที่จำลองจากทางหลวงระหว่างเมืองและระยะทางวัดเป็นกิโลเมตร โดยประมาณ หมายเลยประจำเส้นทางตั้งแต่ 1-9 ใช้เรียกชื่อแทนหมายเลขทางหลวงเพื่อความสะดวกต่อการอ้างอิง



รูปที่ 7-11 แผนที่ทางหลวง

7.5.2 เรียนรู้เครือข่าย

เราเตอร์ในเครือข่ายที่ใช้โอเอสพีเอฟจะเริ่มต้นเรียนรู้ว่ามีเราเตอร์ใดที่อยู่ข้างเคียงโดยการส่งแพ็กเก็ต "ทักทาย" หรือ ฮัลโหลแพ็กเก็ต (Hello Packet) ออกไปทุกๆลิงค์ของเราเตอร์ ตัวอย่างรูปที่ 7-12 เราเตอร์ A จะส่งแพ็กเก็ตทักทายไปยังเราเตอร์ที่ B,C และ D ในขณะที่เราเตอร์ทั้งสามก็ส่งแพ็กเก็ตทักทายไปยังเราเตอร์ A ด้วย



รูปที่ 7-12 เราเตอร์ A ส่งแพ็กเก็ตทักทาย

เมื่อเราเตอร์ A ได้รับแพ็กเก็ตตอบกลับจาก B,C และ D เราเตอร์ A ก็จะทราบถึงเส้นทางข้างเคียงโดยตรง เราเตอร์ A จะสร้างสร้างตารางระยะทางไปยังเราเตอร์ข้างเคียงตามตารางที่ 7-1 ในขณะเดียวกันและในที่สุดเราเอตร์แต่ละจังหวัดต่างก็จะมีข้อมูลระยะทางไปยังเราเตอร์ข้างเคียงทุกตัว

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

ตารางที่ 7-1 ตารางเส้นทางที่เราเตอร์ A

7.5.3 สร้างฐานข้อมูลเส้นทาง

ขั้นตอนที่สองนี้แต่ละเราเตอร์จะสร้างฐานข้อมูลที่เก็บเส้นทางไปยังเราเตอร์ทุกตัวในเครือข่าย วิธีการที่ใช้คือเราเตอร์สร้างแพ็กเก็ตบรรจุเส้นทางที่มีอยู่ทั้งหมดในขณะนั้นและ"ปล่อย" (Flooding) ให้แพ็กเก็ตไหลจากเราเตอร์หนึ่งไปยังอีกเราเตอร์หนึ่ง

เมื่อเราเตอร์ได้รับแพ็กเก็ตก็จะสำเนาแพ็กเก็ตนั้นส่งออกไปยังลิงค์อื่นด้วย เราเตอร์ต้องตรวจสอบว่าเป็นแพ็กเก็ตเก่าซ้ำของเดิมหรือเป็นแพ็กเก็ตใหม่ หากได้รับแพ็กเก็ตเก่าจากเราเตอร์อื่นให้กำจัดทิ้งไป หากเป็นแพ็กเก็ตใหม่จะต้องปล่อยแพ็กเก็ตไปทุกอินเทอร์เฟสยกเว้นอินเทอร์เฟสที่ได้รับแพ็กเก็ต

รูปแบบของแพ็กเก็ตที่เราเตอร์ส่งออกไปในขั้นนี้เรียกว่า **แพ็กเก็ตประกาศลิงก์เสตท** (Link State Advertisement) หรือ **แพ็กเก็ตแอลเอสเอ** (LSA) เมื่อเราเตอร์ทุกตัวปล่อยแพ็กเก็ตแอลเอสเอเช่นเดียวกันเพียงชั่วระยะหนึ่งแพ็กเก็ตจะกระจายไปทั่วทั้งเครือข่าย ท้ายที่สุดแล้วเราเตอร์ทุกตัวจะทราบเส้นทางไปยังเราเตอร์อื่นและได้ตารางเส้นทางเรียกว่า ฐานข้อมูลลิงเสตท (Link State Database) จากตัวอย่างนี้ฐานข้อมูลลิงก์เสตทมีค่าตามตารางที่ 7-2

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 7-2 ฐานข้อมูลลิงสเทตประจำแต่ละเราเตอร์

7.5.4 การคำนวณเส้นทาง

ฐานข้อมูลที่ได้จากขั้นที่แล้วยังไม่ใช่ตารางเส้นทางที่นำไปใช้ได้ ตารางเส้นทางจะได้จากการคำนวณหาเส้นทางที่สั้นที่สุดตามขั้นตอนวิธีของไดจ์สตรา (Dijkstra Algorithm)

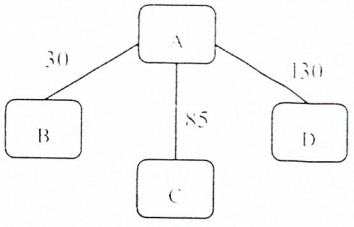
7.5.4.1 ขั้นตอนวิธีของไดจ์สตรา (Dijkstra Algorithm)

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

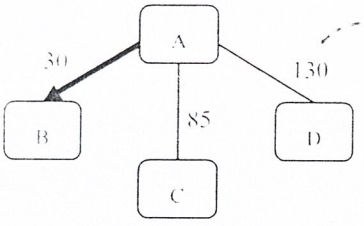
เมื่อเริ่มต้นทำงานที่เราเตอร์ใด เราเตอร์นั้นจะมีระยะทางถึงตัวเองเป็นศูนย์และสร้างเส้นทางชั่วคราวไปยังเราเตอร์ข้างเคียงก่อน ส่วนเส้นทางไปยังเราเตอร์อื่นให้มีค่าเป็นอนันต์

เมื่อใช้ขั้นตอนวิธีของไดจ์สตรากับรูปที่ 7-11 โดยพิจารณาเราเตอร์ A จะได้เส้นทางแรกตามรูปที่ 7-13 (1) โดยมีเส้นทางไปยัง B,C และ D ทั้งสามเส้นทางไม่มีหัวลูกศรซึ่งหมายถึงเส้นทางชั่วคราว ลำดับขั้นที่ (2) ถึง (6) แสดงการหาเส้นทางที่สั้นที่สุดไปยังที่อื่นๆ โดยเส้นทางที่มีหัวลูกศรหมายถึงเส้นทางถาวร

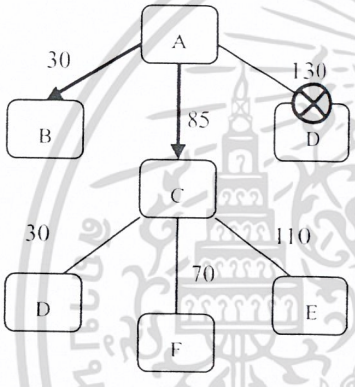
เมื่อเสร็จสิ้นขั้นตอนสุดท้ายแล้วเราเตอร์ A จะมีแผนที่ทั้งเครือข่าย เมื่อต้องการไปยังเส้นทางใดก็เพียงแต่หาเส้นทางจากต้นไม้ที่สร้างขึ้นเท่านั้น ในขณะที่เราเตอร์อื่นก็จะดำเนินการตามขั้นตอนเพื่อหาระยะทางที่สั้นที่สุดไปยังที่อื่นๆในลักษณะเดียวกัน



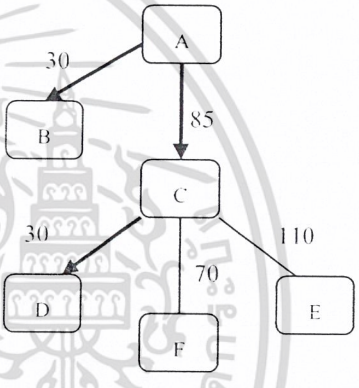
(1) เราเตอร์ A สร้างต้นไม้แสดงเส้นทางไปยังเราเตอร์ข้างเคียง



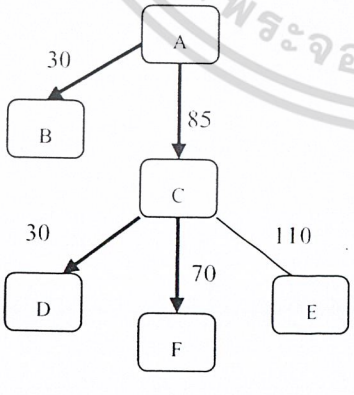
(2) สร้างเส้นทางต่อจากเราเตอร์ B ซึ่งมีเพียงเราเตอร์ D เท่านั้นแต่ระยะทางจาก A ไป D โดยตรงมีค่าเท่ากับ 130 ซึ่งน้อยกว่าการไป D โดยผ่าน B ก่อนซึ่งจะมีค่าเท่ากับ 115+30 = 145 ดังนั้นให้เส้นทางไป B เป็นเส้นทางถาวร



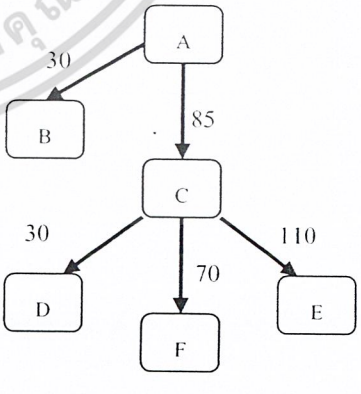
(3) สร้างเส้นทางต่อจากเราเตอร์ C แล้วพบว่าเส้นทางไป D ผ่านทาง C (85+30=115) สั้นกว่าไป D (130) โดยตรง ดังนั้นให้เส้นทางไป C เป็นเส้นทางถาวร และยกเลิกเส้นทางไป D



(4) จากเราเตอร์ D ไม่พบเส้นทางไปเราเตอร์ใดที่สั้นกว่าจึงให้เส้นทางที่ D เป็นเส้นทางถาวร



(5) จากเราเตอร์ F ไม่พบเส้นทางไปเราเตอร์ใดที่สั้นกว่าจึงให้เส้นทางที่ F เป็นเส้นทางถาวร



(6) จากเราเตอร์ E ไม่พบเส้นทางไปเราเตอร์ใดที่สั้นกว่าจึงให้เส้นทางที่ E เป็นเส้นทางถาวร

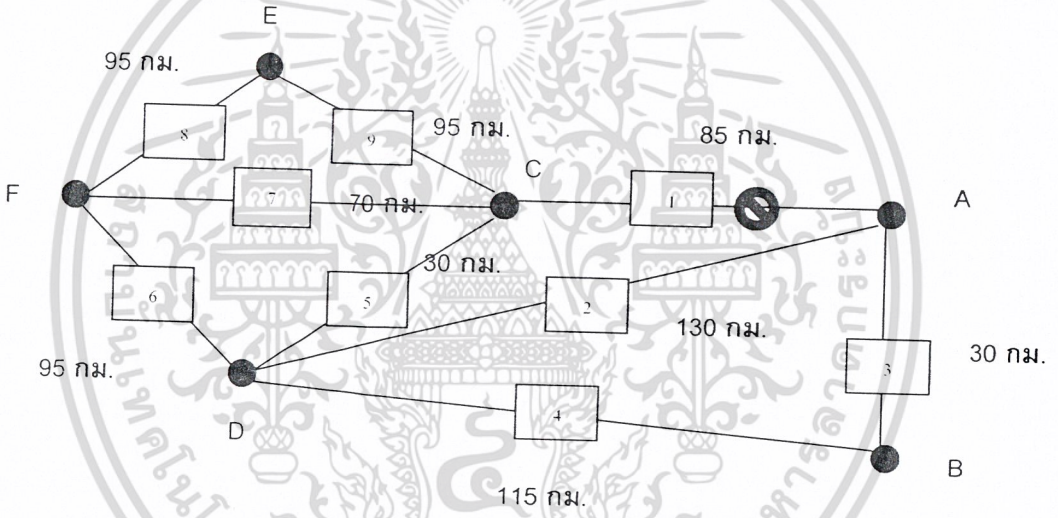
รูปที่ 7-13 การคำนวณหาเส้นทางสั้นที่สุดตามขั้นตอนวิธีของไดจ์สตรา

7.5.4.2 การปรับเปลี่ยนเมื่อเครือข่ายเปลี่ยน

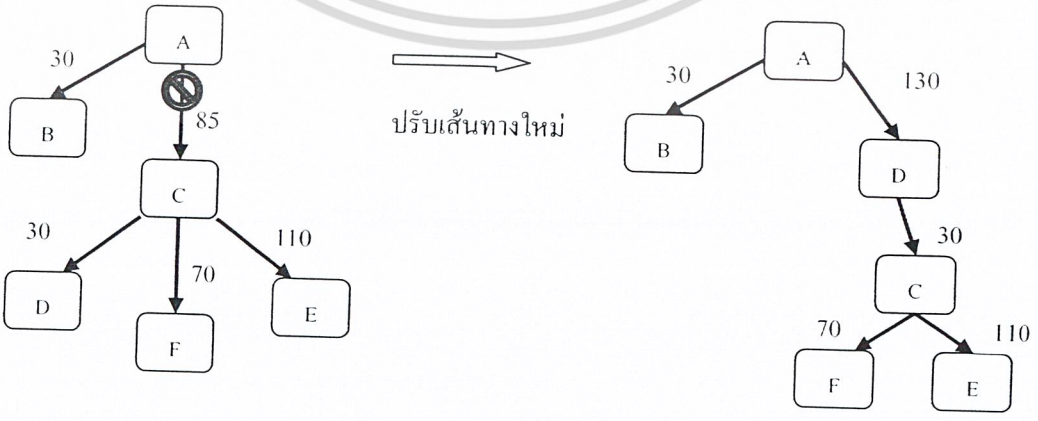
การทำงานของโอเอฟพีเอฟซึ่งประกอบด้วย 3 ขั้นตอนที่กล่าวไปแล้วนั้น ในทางปฏิบัติจะเกิดขึ้นแบบขนานพร้อมกันไปในเราเตอร์แต่ละตัว เราเตอร์จะส่งและได้รับฮัลโหลแพ็กเก็ตกับเราเตอร์ข้างเคียงอยู่ตลอดเวลา หากสายสื่อสารเกิดผิดปกติทำให้เราเตอร์ไม่ได้รับฮัลโหลแพ็กเก็ตภายในระยะเวลาที่กำหนดก็จะทราบว่าเส้นทางไปไม่ถึงและจะเริ่มคำนวณหาเส้นทางใหม่

ตัวอย่างเช่นเส้นทางจากเราเตอร์ A ไปยังเราเตอร์ C มีปัญหาขัดข้องดังรูปที่ 7-14 เราเตอร์ A และ C ต่างไม่ได้รับฮัลโหลแพ็กเก็ตซึ่งกันและกัน เมื่อถึงกำหนดเวลาต่างก็ทราบว่าเส้นทางไม่สามารถใช้งานได้และต้องเปลี่ยนฐานข้อมูลลิงค์เสตทใหม่

หากพิจารณาเฉพาะที่เราเตอร์ A ซึ่งจะคำนวณหาเส้นทางใหม่ กรณีเราเตอร์จะใช้เส้นทางไปยัง D เพื่อไป C โดยเส้นทางอื่นยังคงเดิม ต้นไม้แสดงเส้นทางสั้นที่สุดจะมีลักษณะดังรูปที่ 7-15



รูปที่ 7-14 ปัญหาเส้นทางขัดข้อง



รูปที่ 7-15 เส้นทางใหม่ที่คำนวณได้

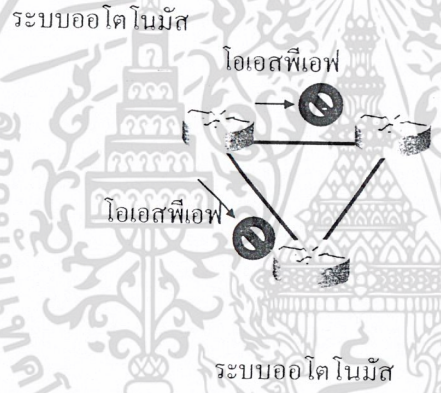
ขั้นตอนวิธีของไดจ์สตราเป็นขั้นตอนวิธีที่ต้องใช้กำลังการคำนวณของซีพียูอย่างมากโดยเฉพาะอย่างยิ่งในเครือข่ายขนาดใหญ่ เราเตอร์จะต้องคำนวณหาต้นไม้ที่แทนเส้นทางสั้นที่สุดใหม่ทุกครั้งที่มีการเปลี่ยนแปลงใดๆในเครือข่าย

แต่โอเอสพีเอฟมีวิธีการจัดรูปแบบเครือข่ายเพื่อลดการคำนวณเมื่อเครือข่ายเปลี่ยนแปลงซึ่งจะกล่าวในหัวข้อถัดไป

7.5.4.3 การจัดองค์ประกอบเครือข่าย

ระบบออโตโนมัสเป็นตัวกำหนดขอบเขตการทำงานของโอเอสพีเอฟ ข้อมูลเส้นทางโอเอสพีเอฟจะแลกเปลี่ยนระหว่างเราเตอร์ที่อยู่ภายในออโตโนมัสหนึ่งๆ เท่านั้น

ในระบบออโตโนมัสหนึ่งๆ ย่อมมีเราเตอร์เชื่อมต่อไปยังระบบอื่นดังตัวอย่างในรูปที่ 7-16 เราเตอร์ในระบบออโตโนมัสที่เชื่อมกับระบบออโตโนมัสอื่นเรียกว่า **เราเตอร์ขอบปลายระบบออโตโนมัส** (AS boundary routers) และเป็นตัวกันไม่ให้ข้อมูลโอเอสพีเอฟออกไปสู่ระบบออโตโนมัสอื่น



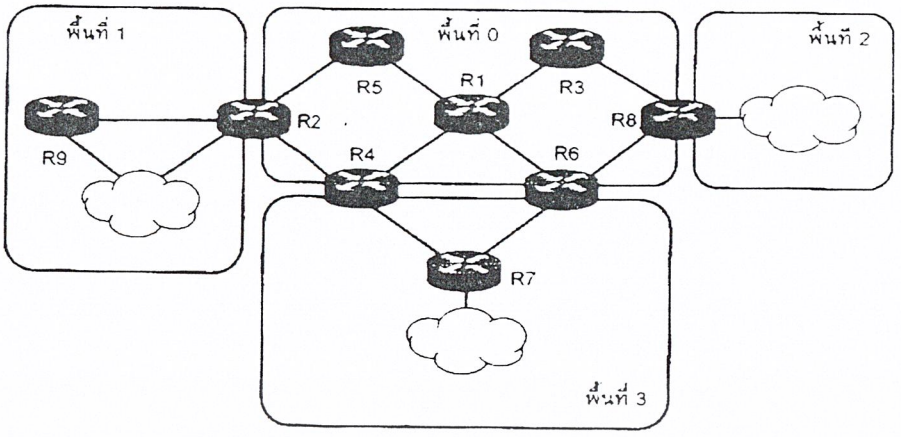
รูปที่ 7-16 เราเตอร์ขอบปลายของระบบออโตโนมัส

7.5.5 การแบ่งระบบออโตโนมัสออกเป็นพื้นที่

เครือข่ายที่ใช้งานโอเอสพีเอฟย่อมอยู่ในระบบออโตโนมัสเดียวกัน และอาจเป็นได้ทั้งเครือข่ายขนาดเล็กไปถึงเครือข่ายขนาดใหญ่ โอเอสพีเอฟสามารถทำงานได้กับเครือข่ายขนาดใหญ่และยังรองรับการขยายขนาดขึ้นไปอีก เพราะมีคุณสมบัติการทำงานที่รองรับการขยายอยู่ในตัว

คุณสมบัติหนึ่งได้แก่การแบ่งเครือข่ายออกเป็นกลุ่มย่อยเพื่อจำกัดขอบเขตการทำงานบางอย่างให้อยู่ในบริเวณใดบริเวณหนึ่งหรือกลุ่มของเราเตอร์กลุ่มหนึ่งๆ ขอบเขตของเครือข่ายที่จัดแบ่งแล้วเรียกว่า **พื้นที่** (Area)

พื้นที่ในความหมายของโอเอสพีเอฟคือกลุ่มของโฮสต์หรือเราเตอร์ภายใต้ขอบเขตที่กำหนด เครือข่ายหนึ่งๆ อาจแบ่งออกเป็นหลายพื้นที่ แต่ละพื้นที่จะเชื่อมโยงกับพื้นที่กลางที่เรียกว่า **พื้นที่แบ็คโบน** Backbone Area) รูปที่ 7-17 แสดงเครือข่ายที่แบ่งออกเป็น 4 ส่วนแต่ละพื้นที่จะมีหมายเลขประจำพื้นที่ (Area ID) โดยแบ็คโบน



รูปที่ 7-17 ตัวอย่างการจัดพื้นที่ในระบบออโตโนมัสของโอเอสพีเอฟ

7.6 บีจีพี(BGP)

การเลือกเส้นทางระหว่างระบบออโตโนมัสมีหลักการที่ต่างจากอาร์ไอพี และโอเอสพีเอฟคือต้องการเส้นทางใดก็ได้ที่ไปถึงปลายทางโดยไม่วนลูป

เทคนิค พาธเวกเตอร์(path vector) ที่ใช้ในบีจีพีจึงไม่ใช้การประกาศค่าเส้นทางที่มีเมตริกกำกับ บีจีพีจะส่งข้อมูลเส้นทางโดยกำหนดลำดับหมายเลขออโตโนมัสในลักษณะของข้อความว่า "สามารถไปถึงเครือข่ายนั้นโดยผ่านระบบออโตโนมัสหมายเลข..."

7.6.1 การเลือกเส้นทางโดยใช้นโยบาย

หากมีเส้นทางไปยังเครือข่ายปลายทางได้หลายเส้นทางบีจีพีจะเลือกเส้นทางตามนโยบาย ที่ผู้ดูแลระบบกำหนด นโยบายเลือกเส้นทางไม่ได้เป็นส่วนหนึ่งของโพรโตคอลแต่เป็นข้อกำหนดตามแนวทางการบริหารและใช้งานเครือข่าย เช่น "ใช้เส้นทางที่ผ่านจำนวนระบบออโตโนมัสน้อยที่สุด" หรือ "ใช้เส้นทางผ่านระบบออโตโนมัส A ก่อนระบบออโตโนมัส B" การเลือกเส้นทางนี้เรียกว่า การเลือกเส้นทางด้วยนโยบาย(policy-based routing)

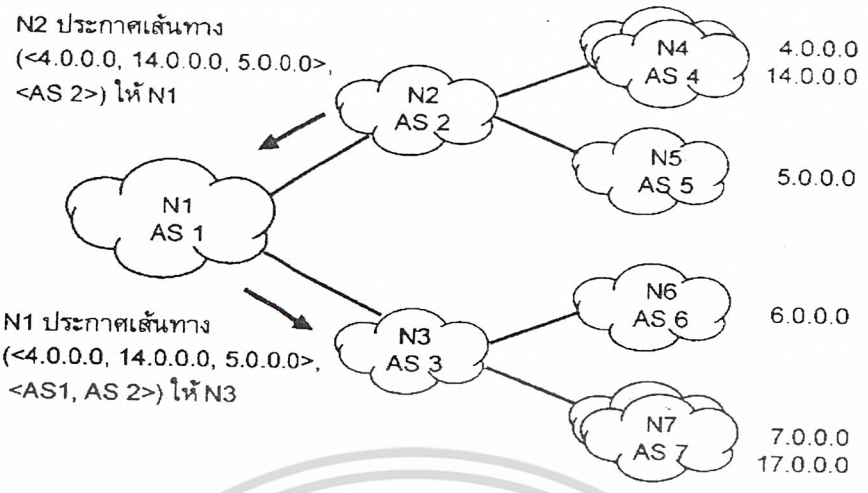
7.6.2 หลักการทำงานของบีจีพี

เส้นทางที่บีจีพีประกาศจะเป็นเส้นทางแสดงลำดับหมายเลขระบบออโตโนมัส เช่น จากรูปที่7-18 เครือข่าย N2 ประกาศเส้นทางให้กับ N1 ว่าไปยังเครือข่าย N4 และ N5 ดังนี้

"4.0.0.0, 14.0.0.0 และ 5.0.0.0 ไปถึงได้โดยผ่าน AS 2"

และ N1 ประกาศเส้นทางให้กับ N3 ว่าไปถึงเครือข่าย N4 และ N5 ในรูปแบบ

"4.0.0.0, 14.0.0.0 และ 5.0.0.0 ไปถึงได้โดยผ่าน AS 1 และ AS 2"



รูปที่ 7-18 ตัวอย่างการเลือกเส้นทางในบีจีพี

7.6.3 ชนิดการส่งข้อมูล

บีจีพีทำงานด้วยทีซีพีผ่านพอร์ต 179 เราเตอร์บีจีพีส่งตารางเส้นทางทั้งตารางระหว่างกันในขั้นต้น หากมีบางค่าในตารางเปลี่ยนแปลงไปก็สามารถส่งเฉพาะค่าที่เพิ่มเติมได้บีจีพีไม่ใช้การตั้งเวลาเพื่อปรับตารางเส้นทางเป็นจังหวะ เราเตอร์จึงต้องรักษาค่าตารางเส้นทางตลอดการทำงาน บีจีพีกำหนดรูปแบบการส่งข้อมูลระหว่างเราเตอร์ 4 แบบคือ

7.6.3.1 การเปิด(open) ข้อความติดต่อเพื่อขอเปิดการเชื่อมโยงระหว่างกัน และเป็นข้อความแรกที่สื่อสารกันหลังจากที่ได้สถาปนาการเชื่อมโยงผ่านทางทีซีพีแล้ว

7.6.3.2 การปรับค่า(update) ข้อความปรับค่าเส้นทางระหว่างบีจีพีเพื่อเพิ่ม เปลี่ยนแปลงหรือยกเลิกเส้นทางที่กำหนด

7.6.3.3 การแจ้ง(notification) ใช้สำหรับแจ้งเหตุเมื่อตรวจพบสิ่งผิดปกติ เพื่อปิดการเชื่อมต่อ และรายงานถึงสาเหตุของความผิดปกติ

7.6.3.4 การรักษา(keep-alive) ใช้สำหรับแจ้งว่าการเชื่อมต่อยังคงสภาพอยู่ บีจีพีจะส่งข้อความเพื่อรักษาสภาพการเชื่อมต่อไม่ให้ขาดหายไปทุก 30 วินาที

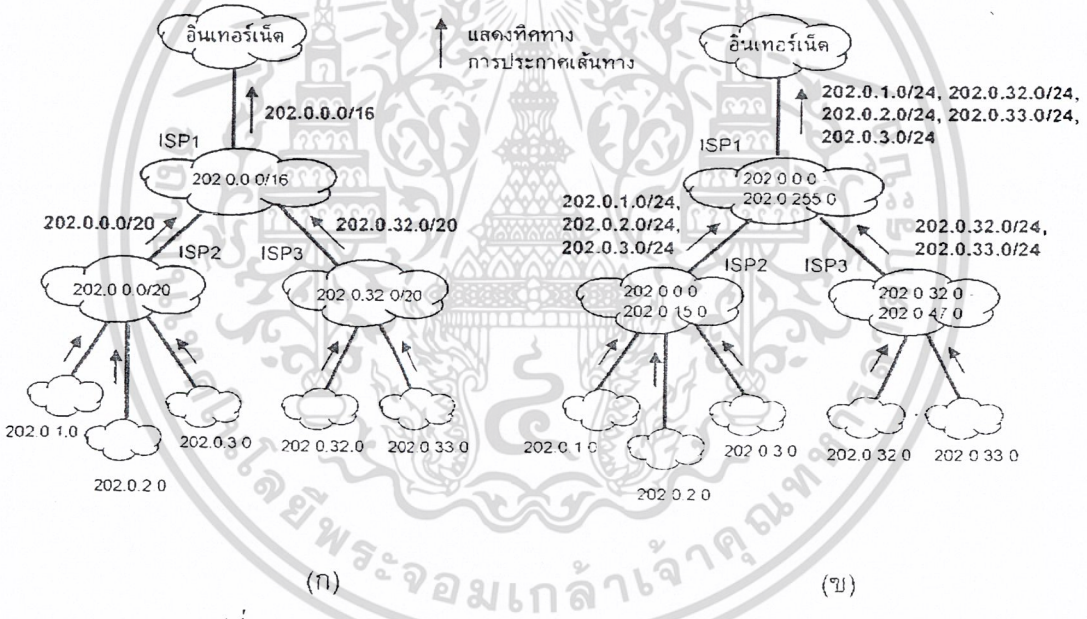
7.6.4 บีจีพีและไซเดอร์

แนวคิดของไซเดอร์คือการจับกลุ่มเครือข่ายเข้าด้วยกันเพื่อให้สามารถรวมเส้นทางลงเหลือเพียงค่าเดียวซึ่งเรียกว่า **เส้นทางสรุป**(route summarization) เครือข่ายที่จับกลุ่มกันได้ต้องมีแอดเดรสเครือข่ายต่อเนื่องกัน และมีจำนวนเครือข่ายเป็นค่า 2^n คือ 2, 4, 8, ... จากรูปที่ 7-19 แสดงตัวอย่างการจัดสรรแอดเดรสและประกาศค่าเส้นทางสรุป

ISP1 เป็นไอเอสพีหลักที่ต่อเชื่อมกับอินเตอร์เน็ตได้รับการจัดสรรแอดเดรสจำนวน 256 ชุด ต่อเนื่องกันคือ 202.0.0.0 ถึง 202.0.255.0 แอดเดรสนี้ได้รับการจัดสรรต่อไปให้ไอเอสพีระดับถัดไปคือ ISP2 และ ISP3

ISP2 ได้แอดเดรสจำนวน 16 ชุดจาก 202.0.0.0 ถึง 202.0.15.0 และ ISP3 ได้แอดเดรสจำนวน 16 ชุดจาก 202.0.32.0 ถึง 202.0.47.0 ทั้ง ISP2 และ ISP3 จัดสรรแอดเดรสให้กับเครือข่ายถัดไปซึ่งเป็นเครือข่ายส่วนปลาย มีการจัดสรรเหมือนกันทั้งรูป 7-19 (ก) และ (ข)

รูปที่ 7-19 (ก) แสดงการสรุปรวมแอดเดรสจากผู้ให้บริการอินเตอร์เน็ตระดับล่างประกาศค่าด้วยบีจีพีไปสู่ผู้ให้บริการอินเตอร์เน็ตระดับบน สังเกตได้ว่าการประกาศค่าสามารถสรุปรวมเป็นค่าเดียวในแต่ละระดับและ ISP1 ก็เพียงแต่ประกาศ่าสรุป 202.0.0.0. ออกไปยังอินเตอร์เน็ตเท่านั้น เมื่อเปรียบเทียบกับรูปที่ 7-19 (ข) ซึ่งไม่ใช้ไซเดอร์ การประกาศคาเส้นทางจะสะสมตามจำนวนเครือข่ายขึ้นตามลำดับทำให้ ISP1 ต้องประกาศเส้นทางออกไปรวม 5 เส้นทาง หรือในกรณีที่ใช้เครือข่ายครบทั้ง 256 ชุดก็จะต้องประกาศออกไปทั้ง 256 ค่า



รูปที่ 7-19 การประกาศเส้นทาง (ก) ใช้ไซเดอร์ (ข) ไม่ใช้ไซเดอร์

7.7 ไอจีอาร์พี(IGRP)

ไอจีอาร์พีเป็นโพรโตคอล ในการเลือกเส้นทางที่ถูกพัฒนาในตอนกลางถึงปีค.ศ.1980 โดยบริษัทซิสโก้ เป้าหมายของการคิด ไอจีอาร์พี คือต้องการเตรียม โพรโตคอลที่มีประสิทธิสำหรับการเลือกเส้นทางภายใน ออโตโนมัสซิสเต็ม ตัวอย่างโพรโตคอลที่เป็นที่รู้จักกันคือ Interior Gateway Routing Protocol

ในช่วงกลางปีค.ศ.1980 Interior Gateway Routing Protocol ที่ได้รับความนิยมสูงคือ โพรโตคอลอาร์ไอพี แม้ว่า อาร์ไอพี จะค่อนข้างเป็นประโยชน์มากสำหรับการเลือกเส้นทางภายใน เน็ตเวิร์กขนาดเล็กถึงขนาดกลาง ซึ่งมันจำกัดการขยายตัวของเน็ตเวิร์ก โดยเฉพาะอย่างยิ่ง ขนาดฮ็อพ-เคานท์ของ อาร์ไอพี จะ

ถูกจำกัดขนาด มันจะสนับสนุนเพียง equal-cost load balancing ในเครือข่ายซิสโก้ เท่านั้น ไม่อนุญาตให้ยืดหยุ่นได้ในสิ่งแวดล้อมที่ซับซ้อน เราท์เตอร์ของ ซิสโก้ เป็นที่นิยมมากและความแข็งแรงของ ไอจีอาร์พี ที่สนับสนุนองค์กรใหญ่ๆหลายองค์กรที่มีเน็ตเวิร์กขนาดใหญ่ก็เปลี่ยนจาก อาร์ไอพี มาเป็น ไอจีอาร์พี ค่าเริ่มต้นของ ไอจีอาร์พี ของ ซิสโก้ ถูกเอามาทำงานใน Internet Protocol networks IGRP ถูกออกแบบมาให้ทำงานในเครือข่ายหลายรูปแบบ อย่างไรก็ตาม ซิสโก้ ทำให้มันสามารถทำงานใน OSI Connectionless-Network Protocol (CLNP) networks Cisco พัฒนาไอจีอาร์พีให้ดีขึ้นในช่วงทศวรรษ 1990 เพื่อให้ ไอจีอาร์พี ทำงานได้อย่างมีประสิทธิภาพ ซึ่งจะพูดถึงการออกแบบ ไอจีอาร์พี แบบพื้นฐานและการอิมพลีเม้นท์ ไอจีอาร์พี

7.7.1 ลักษณะของโพรโตคอลไอจีอาร์พี

ไอจีอาร์พี เป็น ไอจีพี(Interior Gateway Protocol) ชนิด ดีซแท็นซ์เวกเตอร์(distance vector) ซึ่ง ดีซแท็นซ์เวกเตอร์ เป็น เราท์ติ้ง โพรโตคอลแบบที่มีการเปรียบเทียบระยะทาง ซึ่งการวัดระยะทางแบบนี้เป็นที่รู้จักกันในฐานะ ดีซแท็นซ์เวกเตอร์ ซึ่ง เราท์เตอร์ที่ใช้ โพรโตคอลแบบ ดีซแท็นซ์เวกเตอร์ จะส่งตารางที่ปรับปรุงแล้วจาก การเลือกเส้นทางทั้งหมดหรือเพียงบางส่วนให้กับ เราท์เตอร์ข้างเคียง แล้วข้อมูลการ เลือกเส้นทางก็จะแพร่กระจายไปทั่ว เน็ตเวิร์ก และ สุดท้ายจะคำนวณระยะทางที่รู้ทั้งหมด

ดีซแท็นซ์เวกเตอร์เราท์ติ้งโพรโตคอล มักจะถูกเปรียบเทียบกับ ลิงค์สเต็ดเราท์ติ้งโพรโตคอล ซึ่ง ลิงค์สเต็ดจะส่งข้อมูลการติดต่อทั้งหมดให้กับทุก โหนด(node) ใน เน็ตเวิร์ก สำหรับ โอเอสพีเอฟ (OSPF) และ อินเตอร์มีเดียร์ซิสเต็มทูอินเตอร์มีเดียร์ซิสเต็ม(Intermediate System-to-Intermediate System) เป็นอัลกอริทึมส์ของ ลิงค์สเต็ดเราท์ติ้ง

ไอจีอาร์พี ใช้หน่วยต่างๆในการคำนวณซึ่งการคำนวณโดยใช้ ค่าน้ำหนักสำหรับการเกิดดีเลย์, แบนวิทต์, ค่า ความน่าเชื่อถือ และ โหลด ซึ่ง ผู้ดูแลเครือข่าย สามารถตั้งค่า weighting factor สำหรับแต่ละ เมตริกซ์ แม้ว่าจะต้องระมัดระวังอย่างมากในการตั้งค่า ดีฟอลท์ ซึ่ง ไอจีอาร์พี จะเตรียมค่าหลายค่าสำหรับใช้วัด ตัวอย่าง ค่า ความน่าเชื่อถือ และ โหลด มีค่าได้ตั้งแต่ 1-255 ส่วนค่า แบนวิทต์ มี ค่าประมาณ 1200 bps – 10 Gbps ขณะที่ ดีเลย์ มีค่า ได้ประมาณ 1-224 ค่าการวัดที่หลากหลายนี้เป็นการทำให้มันสมบูรณ์ได้มากขึ้นโดยชุดของค่าที่กำหนดได้จะสามารถทำให้ ผู้ดูแลเครือข่าย สามารถเลือกเส้นทางการได้ เพื่อทำให้มันยืดหยุ่นขึ้น ไอจีอาร์พี อนุญาตให้มีการเราท์ติ้งแบบ มัลติพาทส์แบนวิทต์ไลน์(multipath bandwidth lines) ทั้งสองเส้นที่เท่ากันสามารถทำงานแบบ ซิงเกอร์สตรีม(single stream) ในลักษณะ แบบ เราว์โรบิน(round-robin) และมีการเปลี่ยนเส้นทางโดยอัตมัติเมื่อเส้นใดเส้นหนึ่งไม่สามารถทำงานได้ เส้นทางหลายๆ เส้นทางมีความสามารถไม่เท่ากันกับค่า metric แต่ยังคงเป็นทางที่ถูกต้อง ตัวอย่างเช่น ถ้ามีทางๆหนึ่งมีค่าเท่ากับ 3 แต่เป็นเส้นทางที่ดีกว่าเส้นทางอื่นซึ่งมีค่าน้อยกว่า 3 และ ทางที่ดีกว่าจะถูกใช้บ่อย มีเพียงบางเส้นทางที่อยู่ภายในช่วงที่แน่นอนหรือ ทางที่ดีที่สุดถูกใช้เป็น มัลติพาทส์ ค่าที่แตกต่างกันคือ ค่าต่างๆที่สามารถติดตั้งโดย ผู้ดูแลเครือข่าย

7.7.2 ความเสถียรของไอจีอาร์พี

ไอจีอาร์พีมีจุดเด่นที่ถูกออกแบบมาเพื่อให้มันเสถียรขึ้น รวมถึง โฮลดาวน์(Holddown) ถูกใช้เพื่อป้องกันการปรับปรุงค่าจากการเลือกเส้นทางที่ไม่เหมาะสม เมื่อเราท์เตอร์ไม่ทำงาน เราท์เตอร์อื่นๆ จะตรวจสอบผ่านทางการตารางการปรับปรุงค่า(update schedule) หลังจากนั้นเราท์เตอร์เหล่านั้น จะทำการคำนวนหาทางใหม่และส่งผลการปรับปรุงค่าไปบอก เราท์เตอร์อื่นๆให้ เปลี่ยนเส้นทาง การทำงานแบบนี้จะทำในรูปของสัญญาณทริกเกอร์ ไปอัพเดตการป้องกันผ่านเครือข่าย ซึ่งสัญญาณทริกเกอร์เหล่านี้จะไม่ไปถึงอย่างทันที ในทุกๆอุปกรณ์ในเครือข่าย ดังนั้น มันเป็นปัญหาสำหรับอุปกรณ์ ที่ยังไม่รู้ว่าเครือข่ายล้มเหลว การส่งข่าวสารการอัพเดตซึ่งมันจะแจ้งความล้มเหลวนี้ไปบอกอุปกรณ์ในเครือข่ายในกรณีนี้ อุปกรณ์ตัวหลังๆ จะได้รับข้อมูลผิดๆ โฮลดาวน์บอก เราท์เตอร์ให้ หยุดเปลี่ยนแปลงข้อมูลกับเราท์เตอร์ที่มีปัญหาขณะนั้น ช่วงเวลาโฮลดาวน์จะถูกคำนวณให้มากพอ สำหรับการอัพเดตทั้งเครือข่ายด้วยการเปลี่ยนเส้นทาง

สปลิตฮอไรซันจะไม่มีการส่งข้อมูลกลับในทิศที่มันมา ตามรูปในรูปที่ 7-20

รูปที่ 7-20 แสดงถึงกฎสปลิตฮอไรซันเพื่อป้องกันการเกิดลูป

เราท์เตอร์ R1 มันจะมีเส้นทางไปยัง network A ไม่มีเหตุผลที่ R2 จะไปเลือกเส้นทางนั่นเพราะว่า R1 ใกล้ network A มากกว่า กล่าวว่า R2 จะไปยังเส้นทางนั้นได้ต้องส่งผ่านไปยัง R1 กฎของ สปลิตฮอไรซัน ช่วยป้องกันไม่ให้เกิด loop พิจารณาดูตัวอย่างดังนี้ R1 มีส่วนเชื่อมต่ออยู่กับ network A ไม่ได้เป็นแบบสปลิตฮอไรซัน และ R2 จะได้รับเส้นทางอย่างต่อเนื่องเหมือนกับเป็นทางเลือกเมื่อการเชื่อมต่อ ล้มเหลว เป็นสาเหตุให้เกิดลูปแม้ว่าโฮลดาวน์จะป้องกันสิ่งนี้ได้ แต่สปลิตฮอไรซันก็จะถูกใช้ใน ไอจีอาร์พีเพราะว่ามันมีอัลกอริธึมส์ที่เสถียรเป็นพิเศษ

สปลิตฮอไรซันควรจะป้องกันลูประหว่าง เราเตอร์ข้างเคียง(adjacent router) แต่การกำจัดการส่งค่าปรับปรุงย้อนกลับ(Poison-reverse updates)ก็เป็นสิ่งจำเป็น เพื่อกำจัดลูปขนาดใหญ่ การเพิ่มขึ้นของค่าแมตทริกร์โดยปกติจะเกิดขึ้นในเส้นทางเดิม การกำจัดการส่งค่าปรับปรุงย้อนกลับ จะทำการปรับปรุงค่าแล้วจะถูกส่งไปลบเส้นทาง และ แทนที่ภายในช่วงเวลาโฮลดาวน์ สำหรับ ไอจีอาร์พี ของ ซิสโก้ การกำจัดการส่งค่าปรับปรุงย้อนกลับ จะถูกส่งไป ถ้าค่าของเส้นทางมีค่าเพิ่มขึ้นโดยแฟกเตอร์เป็น 1.1 หรือมากกว่า

IGRP จะเก็บรักษาค่าของเวลา และ ตัวแปรที่บรรจุช่วงเวลาไว้ ซึ่งจะประกอบไปด้วย เวลาในการปรับปรุง(timer update) และ ช่วงห่างของเวลา(invalid timer) และ คาบเวลาโฮลไทม์ และ

เวลาในการผลัก(flush timer) เวลาในการปรับปรุง บอกได้ว่าการส่ง ข้อความปรับปรุงเส้นทาง ควรถูกส่งบ่อยแค่ไหน ไอจีอาร์พี กำหนดค่าเริ่มต้นไว้ 90 วินาที ค่า ช่วงห่างของเวลา เอาไว้บอกว่า เราท์เตอร์จะรอก่านานเท่าไหร่ที่ ส่งข้อความปรับปรุงเส้นทาง เกี่ยวกับเส้นทาง ก่อนประกาศยกเลิกเส้นทางนั้น โดยไอจีอาร์พี กำหนดค่าเริ่มต้นไว้คือ 3 เท่าของช่วงเวลาในการปรับปรุงเส้นทาง ตัวแปร โฮลไทม์ จะบอกคาบของโฮลดาวน์ ซึ่งไอจีอาร์พีกำหนดค่าเริ่มต้น 3 เท่าของ ช่วงเวลาในการปรับปรุงเส้นทางและบวกอีก 10 วินาที สุดท้าย เวลาในการผลัก บอกได้ว่าจะใช้เวลาเท่าไหร่ก่อนที่เอาออกจากตารางเส้นทาง ไอจีอาร์พี ได้กำหนดค่าเริ่มต้น ไว้เป็น 7 เท่าของช่วงเวลาในการปรับปรุงเส้นทาง

7.8 สแปนนิ่งทรีโพรโตคอล (Spanning-Tree Protocol)

สแปนนิ่งโพรโตคอลถูกสร้างมาเพื่อใช้ในการกำจัดลูปที่เกิดขึ้นในระบบเครือข่าย โดยจะเสมือนจัดให้ระบบเครือข่ายมีลักษณะเหมือนกับต้นไม้ (Tree) คือจากจุดเริ่มต้นแล้วมีการแตกกิ่งก้านออกมาเรื่อย ๆ เป็นสายแห่งการติดต่อสื่อสาร จะทำให้เป็นเครือข่ายที่ไม่มีลูปเลย

หลักการทำงานของสแปนนิ่งโพรโตคอลคือ สวิตช์ทุกตัวจะทำการส่งเฟรมชื่อว่า Bridge Protocol Data Unit (BPDU) ออกไปยังทุกพอร์ต เพื่อแสดงถึงความมีตัวตนของสวิตช์ แล้วสวิตช์ทุกตัวจะได้รับ BPDU จากสวิตช์ตัวข้างเคียงเพื่อนำมาใช้ในการคำนวณโดยใช้สแปนนิ่งอัลกอริธึม (Spanning Tree Algorithm) ซึ่งมีหลักการคือ

- เลือกรูทบริดจ์ (Root Bridge)
- เลือกรูทพอร์ต (Root Port)
- เลือกดีไซเนเตทพอร์ต (Designated Port)

สวิตช์ทุกตัวจะทำการส่ง BPDU ออกไปยังทุกพอร์ต เป็นการแสดงถึงความมีตัวตนของสวิตช์ จากนั้นสวิตช์ทุกตัวจึงจะได้รับ BPDU มาเพื่อทำการคำนวณโดยใช้สแปนนิ่งอัลกอริธึมจะได้เป็นสแปนนิ่งทรีโพรเซส (Spanning-Tree Process) ซึ่งก็คือ

7.8.1 เลือกรูทบริดจ์

Root Bridge คือจุดที่เป็นจุดอ้างอิง (Reference) ของสแปนนิ่งทรี ซึ่งก็คือ จุดยอดของต้นไม้นั่นเอง โดยรูทบริดจ์จะเป็นสวิตช์ตัวที่มีบริดจ์ไอดี (Bridge ID) น้อยที่สุด

บริดจ์ไอดีประกอบด้วย

- Bridge Priority (2 bytes) ไพรออริตี้ (Priority) หรือ Weight ของสวิตช์สามารถมีแค่ได้ตั้งแต่ 0 – 65535 และมีค่าดีฟอลต์ (default) คือ 32768
- แมคแอดเดรส (8 bytes) เป็นสิ่งที่แสดงในเห็นถึงความเป็นเอกของแต่ละสวิตช์ เนื่องจากแมคแอดเดรสมีคุณสมบัติคือ มีเพียงแมคแอดเดรสเดียวในโลก ไม่เหมือนใคร และไม่สามารถเปลี่ยนแปลงได้

ในตอนเริ่มต้นนั้น สวิตช์จะตั้งค่ารูทบริดจ์เป็นบริดจ์ไอดีตัวเองก่อน จากนั้นจึงมีการรับ BPDU จากสวิตช์ข้างเคียง แล้วนำมาคำนวณหาตัวที่น้อยกว่า ทำเช่นนี้ไปเรื่อย ๆ จนกว่า

สวิตช์ทุกตัวจะมี Root Bridge เป็นตัวเดียวกัน และหลังจากนั้นสวิตช์ก็ยังจะส่ง BPDU ทุก ๆ 2 วินาที (เป็นค่าดีฟอลต์)

7.8.2 เลือกรูทพอร์ต

สำหรับทุก ๆ นอนรูทบริดจ์ (Non-root Bridge) คือสวิตช์ที่ไม่ใช่รูทบริดจ์ต้องทำการเลือกรูทพอร์ต โดยเลือกพอร์ตที่ดีที่สุดในการสื่อสารไปยังรูทบริดจ์ โดยคำนวณจากพอร์ตคอสท์ (Port cost) หรือรูทพาร์ธคอสท์ (Root path cost – จำนวนฮอปทั้งหมดจากรูทบริดจ์จนถึงสวิตช์)

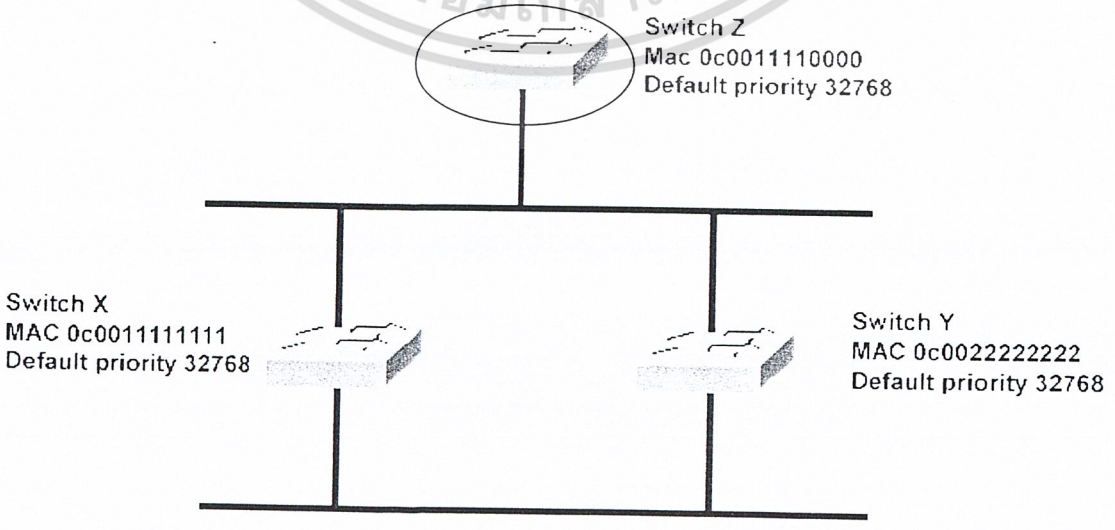
7.8.3 เลือกดีไซเนเตทพอร์ต

ดีไซเนเตทพอร์ตมีคอนเซ็ปต์ว่า ลิงค์เพียงหนึ่งเดียวในเซกเมนต์ที่ใช้ในการส่งและรับทราฟฟิก โดยสำหรับตัวรูทบริดจ์นั้นถือว่า ทุก ๆ พอร์ตของรูทบริดจ์จะเป็นดีไซเนเตทพอร์ตและสำหรับนอนรูทบริดจ์จะให้พอร์ตที่ต่อกับรูทพอร์ตของสวิตช์ตัวข้างเคียงเป็นดีไซเนเตทพอร์ต และสำหรับพอร์ตที่ไม่ใช่รูทพอร์ตและดีไซเนเตทพอร์ตจะถูกบล็อก (Block)

ด้วยหลักการทำงานนี้ จะทำให้แต่ละเครือข่ายแลนสามารถส่งเฟรมข้อมูลไปยังสวิตช์ได้เพียงเครื่องเดียว และเกิดสภาพของต้นไม้ (Tree) ขึ้น โดยพอร์ตที่ไม่ได้ใช้งานจะเป็นพอร์ตสำรอง และเมื่อดำเนินการทำจนเกิดเป็นสแปนนิ่งทรีแล้ว จะได้เครือข่ายที่มีลักษณะ

- One Root Bridge per Network

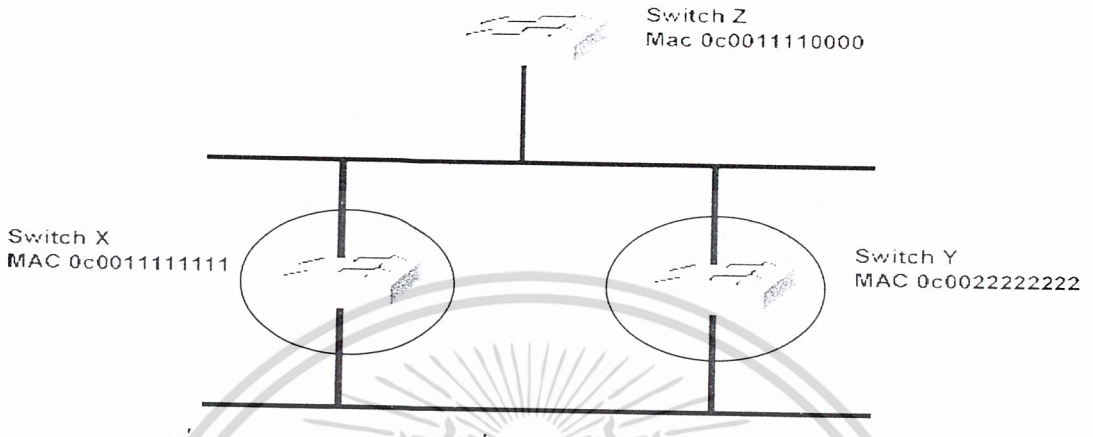
  1 Root Bridge ต่อ 1 ระบบเครือข่าย



รูปที่ 7-21 แสดงระบบเครือข่ายที่มีสวิตช์ Z เป็นรูทบริดจ์

- One Root port per Non-Root Bridge

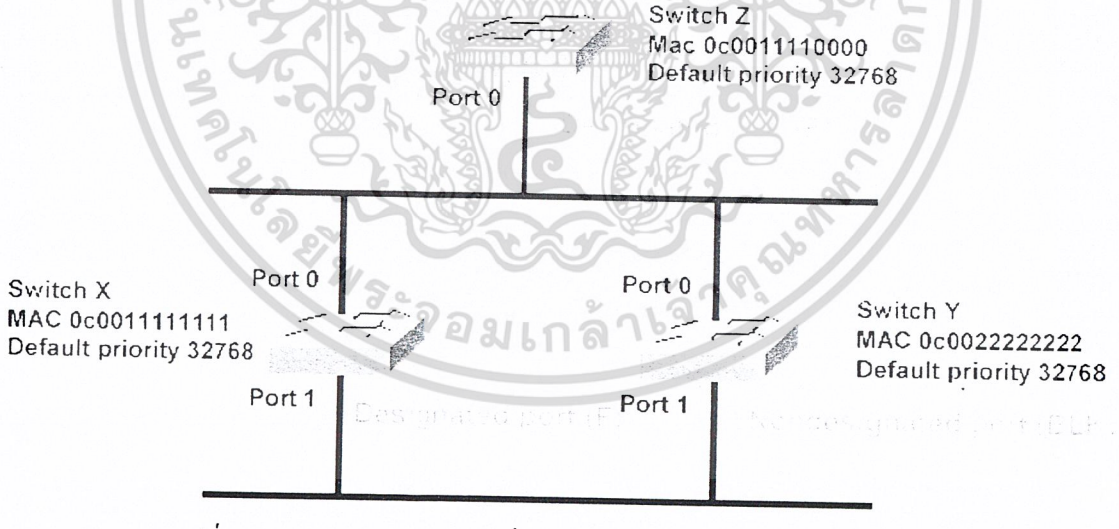
  1 Root Port ต่อ 1 Non-root Bridge



รูปที่ 7-22 แสดงระบบเครือข่ายที่มีสวิตช์ X และสวิตช์ Y เป็นนอนรูทบริดจ์

- One Designated port per Segment

  1 Designated port ต่อ 1 เซกเมนต์



รูปที่ 7-23 แสดงระบบเครือข่ายที่มี 1 ดีไซเนเตทพอร์ตต่อ 1 เซกเมนต์

7.8.4 สแปนนิ่งทรีสเตท (Spanning Tree Port States)

ในการใช้สแปนนิ่งโพรโตคอลทุก ๆ พอร์ตของสวิตช์ จะต้องทำงานเป็นขั้นตอนผ่านไปในแต่ละสเตทโดยจะเริ่มต้นที่ Disabled State และสิ้นสุดที่สเตทสุดท้ายซึ่งอนุญาตให้พอร์ตสามารถรับ-ส่งเฟรมข้อมูลได้ โดยสเตทมีดังต่อไปนี้

1. Disabled – เป็นสเตทของพอร์ตที่ชัทดาวน์ (Shut down) คือไม่สามารถทำอะไรได้เลย สเตทนี้ไม่ถือว่าเป็นส่วนหนึ่งของสแปนนิ่งทรีสเตท
2. Blocking – หลังจากการติดตั้งค่าให้กับพอร์ตแล้ว พอร์ตจะเข้าสู่ Blocking State เพื่อไม่ให้การส่งเฟรมข้อมูลแบบลูป โดยในสเตทนี้จะไม่สามารถรับ – ส่งข้อมูลได้ รวมถึงไม่มีการเรียนรู้แมคแอดเดรส คือ ไม่มีการเพิ่มแมคแอดเดรสลงตารางแมคแอดเดรสอีกด้วย แต่สามารถรับ BPDU จากสวิตช์ข้างเคียงได้ นอกจานี้พอร์ตที่อยู่ในโหมดสแตนด์บาย (Standby Mode) จะเข้าสู่ Blocking State เช่นเดียวกัน
3. Listening – พอร์ตจะเปลี่ยนสเตทจาก Blocking State ไปเป็น Listening State ก็ต่อเมื่อพอร์ตได้รับการเลือกให้เป็นรูทพอร์ตหรือดีไซเนเตทพอร์ต หรือในอีกนัยหนึ่งก็คือ พอร์ตกำลังจะสามารถฟอร์เวิร์ดทราฟฟิก (Forward Traffic) ได้นั่นเอง โดยในสเตทนี้พอร์ตจะสามารถรับ – ส่ง BPDU ได้แต่ถ้าพอร์ตสูญเสียสถานะการเป็นรูทพอร์ตหรือดีไซเนเตทพอร์ต เมื่อใด พอร์ตจะกลับไปสู่ Blocking State อีกครั้ง
4. Learning – เมื่อผ่านช่วงเวลาที่เรียกว่า Forward Delay ใน Listening State แล้วพอร์ตจะเปลี่ยนไปสู่ Learning State โดยที่พอร์ตจะสามารถรับ – ส่ง BPDU ได้และสวิตช์ยังสามารถเรียนรู้แมคแอดเดรสได้อีกด้วย
5. Forwarding – เมื่อผ่านช่วงเวลาที่เรียกว่า Forward Delay ใน Learning State แล้วพอร์ตจะเปลี่ยนไปสู่ Forwarding State โดยพอร์ตจะสามารถรับ-ส่งข้อมูล. เรียนรู้แมคแอดเดรสและรับ-ส่ง BPDU ได้โดยในสเตทนี้ถือว่าได้ทำสแปนนิ่งทรีเสร็จสิ้นแล้วนั่นเอง

7.8.5 ชนิดของสแปนนิ่งโพรโตคอล (Types of Spanning Tree Protocol)

ในตอนเริ่มต้นนั้นสแปนนิ่งโพรโตคอลได้รับการพัฒนามาเพื่อใช้กับเครือข่ายแลน (VLAN) เดียวเท่านั้น การสร้างสแปนนิ่งโพรโตคอลเพื่อให้สามารถใช้ได้กับ Multiple VLANs นั้นจำเป็นต้องประกอบไปด้วยหลาย ๆ สิ่ง ด้วยเหตุนี้ จึงได้มีการสร้างสแปนนิ่งโพรโตคอลที่แตกต่างกันออกมา ซึ่ง ณ ที่นี้เราจะขอหยิบขึ้นมา 3 ประเภทด้วยกันคือ

7.8.5.1 คอมมอนสแปนนิ่งทรี (Common Spanning Tree (CST))

มาตรฐาน IEEE 802.1Q ได้กำหนดถึงการสร้างทรังค์ลิงค์ (Trunk Link) ที่ใช้ระหว่างสวิตช์ และได้กำหนดให้ใช้เป็น 1 สแปนนิ่งทรีต่อหนึ่งเครือข่าย ก็คือทุกวีแลนมีชื่อเรียกว่าคอมมอนสแปนนิ่งทรี (Common Spanning Tree (CST)) หรือ โมโนสแปนนิ่งทรี (Mono Spanning Tree (MST)) ทุก ๆ BPDUs จะถูกส่งไปทั่วทั้งเครือข่าย

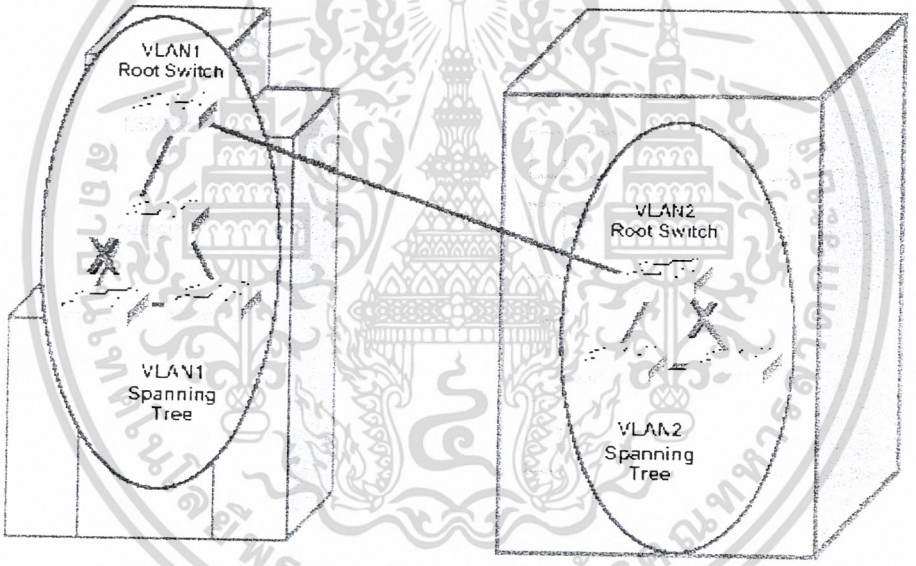
การใช้เพียง 1 สแปนนิ่งทรีต่อหลาย ๆ วีแลน มีข้อดีคือ สามารถใช้คำสั่งได้ง่าย ไม่ยุ่งยากและยังลดการใช้ซีพียู (CPU) ของสวิตช์ในการคำนวณอีกด้วย อย่างไรก็ตาม การกระทำเช่นนี้ยังมีข้อเสียอยู่ก็คือ Redundant Link ที่ได้กำหนดขึ้นจะไม่ได้ใช้งาน ถูกบล็อกเพื่อกำจัดลูป

และยังมีข้อจำกัดในเรื่องวีแลนคือพอร์ตที่ถูกบล็อค อาจจะเป็นพอร์ตที่ใช้ส่งเฟรมข้อมูลของวีแลนใด ๆ ก็ได้ จึงอาจจะเป็นเหตุให้ไม่สามารถส่งเฟรมข้อมูลได้

7.8.5.2 เปอร์วีแลนสแปนนิ่งทรี (Per-VLAN Spanning Tree (PVST))

เปอร์วีแลนสแปนนิ่งทรี (Per-VLAN Spanning Tree) เป็นสแปนนิ่งทรีที่มีความยืดหยุ่นมากกว่า CST เนื่องจากมีหลักการทำงานในการสร้างสแปนนิ่งทรีแยกเป็นของแต่ละวีแลนเลย เพื่อสามารถใช้คำสั่งได้อย่างเป็นอิสระ และยังมีประสิทธิภาพมากกว่า Multiple Spanning Tree สามารถสร้าง Load Balancing บน Redundant Links เมื่อลิงค์นั้นถูกกำหนดไว้ในคนละวีแลนกัน

เนื่องจากคุณสมบัติของ PVST จึงจำเป็นต้องใช้ Cisco Inter-Switch Link (ISL) ในการส่งข้อมูลผ่านทรังค์ลิงค์ระหว่างสวิตช์



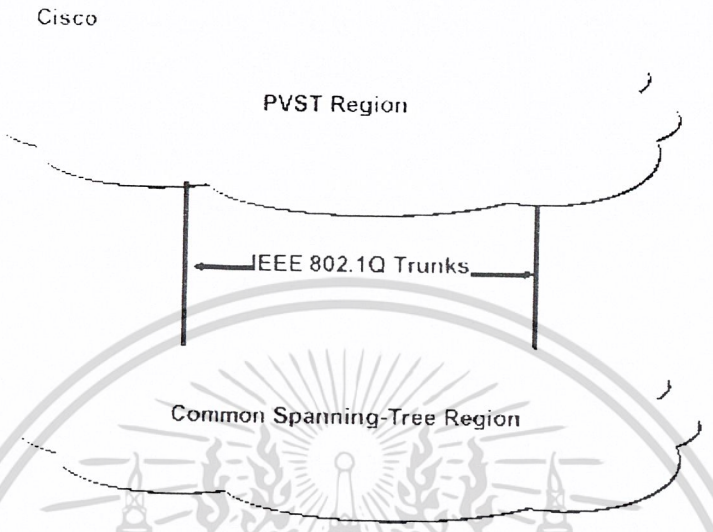
รูปที่ 7-24 แสดงตัวอย่างเครือข่ายที่ใช้เปอร์วีแลนสแปนนิ่งทรี

7.8.5.3 เปอร์วีแลนสแปนนิ่งทรีพลัส (Per-VLAN Spanning Tree Plus (PVST+))

เปอร์วีแลนสแปนนิ่งทรีพลัส (Per-VLAN Spanning Tree Plus) มีความสามารถในการติดต่อสื่อสารระหว่าง CST และ PVST PVST+ สามารถสนับสนุนการทำงานของสแปนนิ่งทรีทั้ง 3 กลุ่มคือ Catalyst ที่ใช้ PVST, Catalyst ที่ใช้ PVST+ และสวิตช์ที่ใช้งาน CST/MST บน 802.1Q

หลักในการทำงานคือ PVST+ จะเป็นเสมือนตัวที่ใช้ติดต่อสื่อสารระหว่างกลุ่มของ CST สวิตช์และกลุ่มของ PVST สวิตช์โดย PVST+ สามารถติดต่อกับ PVST ได้โดยตรงโดยผ่าน ISL Trunks และติดต่อกับ CST อย่างไรก็ตาม PVST+ จะแลกเปลี่ยน BPDUs กับ CST บนวีแลน1 สำหรับ BPDUs ที่มาจากสแปนนิ่งทรีของแต่ละวีแลนนั้นจะเดินทางไปยังส่วนของ CST ในระบบเครือข่ายด้วย Tunnel PVST+ จะส่ง BPDUs ที่กล่าวมานี้โดยการใช้ Multicast Address

ดังนั้น CST สวิตช์จะสามารถ Forward BPDUs ไปยังสวิตช์ข้างเคียงได้ จนในที่สุด BPDUs จึงจะสามารถเดินทางไปถึง PVST+ สวิตช์ได้



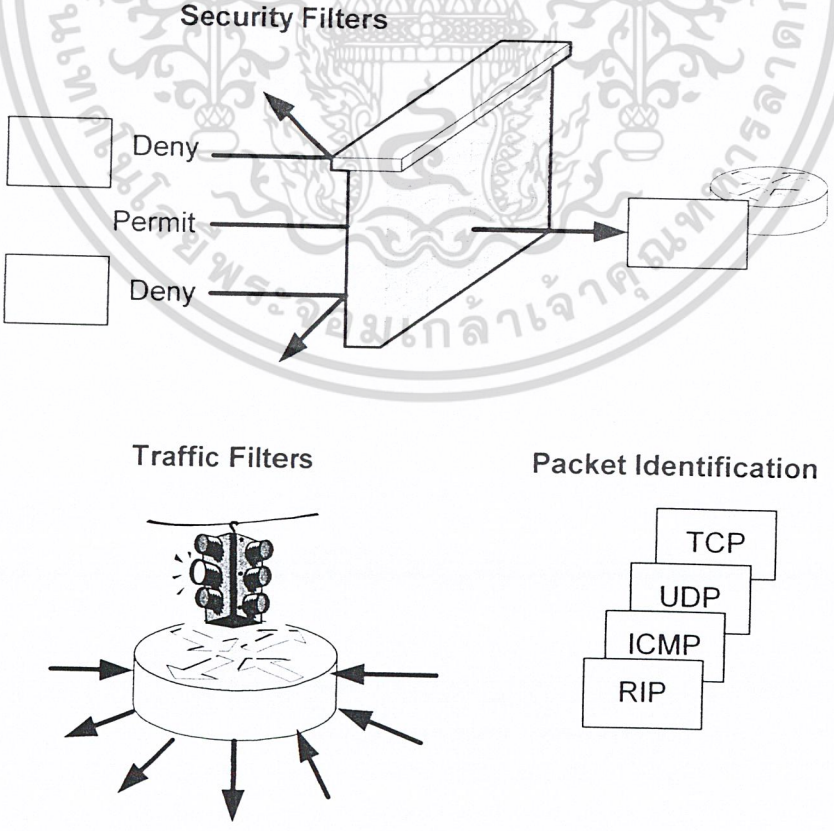
รูปที่ 7-25 แสดงการทำงานของเปอร์วีแลนสแปนนิ่งทรีพลัส

บทที่ 8

แอคเซสลิสต์(Access list)

แอคเซสลิสต์ เป็นวิธีการที่รู้จักกันโดยทั่ว ไปในปัจจุบันซึ่งการทำงานประกอบด้วยการอนุญาตและการป้องกันของแพ็กเก็ตที่เข้าหรือออกจากเราเตอร์ ซึ่งแอคเซสลิสต์กลายเป็นเครื่องมือสำหรับใช้ควบคุมแพ็กเก็ตและภายในเครือข่ายที่มีประสิทธิภาพในปัจจุบัน ซึ่งมีรูปแบบการทำงานหลัก 3 แบบดังนี้ คือ

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

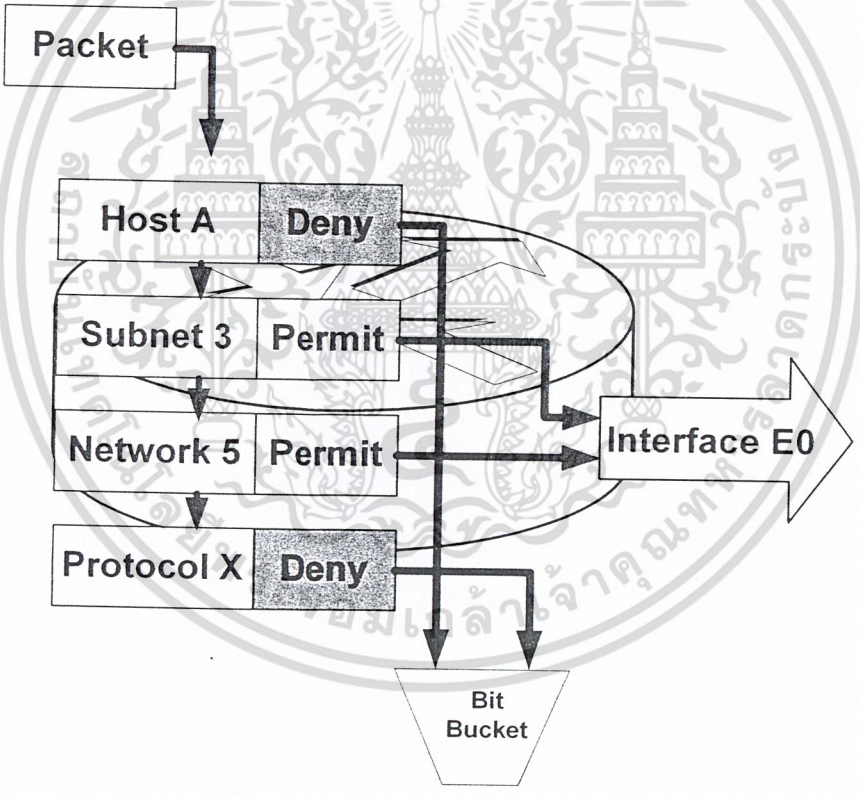


รูปที่ 8-1 การนำแอคเซสลิสต์ไปใช้งาน

8.1 พื้นฐานของแอคเซสลิสต์

แอคเซสลิสต์เป็นวิธีการตรวจสอบแบบเป็นลำดับและต่อเนื่องโดยการทดสอบจะเปรียบเทียบจาก แพ็กเก็ตข้อมูลที่เข้ามา และจะเลือกว่าจะกระทำกับชุดข้อมูลนั้นอย่างไร ระหว่างการอนุญาตหรือป้องกัน ในส่วนการเปรียบเทียบจะทำได้ตั้งแต่การเปรียบเทียบแอดแดรสต้นทาง หรืออาจมีความซับซ้อนยิ่งขึ้น เช่น การตรวจสอบทั้งแอดแดรสต้นทางและแอดแดรสปลายทาง รูปแบบทางโพรโตคอล หมายเลขพอร์ต เป็นต้น

แพ็กเก็ตจะถูกนำเข้ามาไว้ที่บนสุดของ สแตกฟิวล์เตอร์ ดังรูปที่ 8-2 และในแต่ละขั้นตอนการตรวจเช็คถ้าตรวจเช็คแล้วเจอเงื่อนไขที่ตรงกันก็จะไปทำต่อในการทำงานที่เลือกไว้โดยอาจเป็นการอนุญาตหรือการป้องกันแพ็กเก็ตนั้นแต่ถ้าการเปรียบเทียบไม่ตรงกับเงื่อนไขแพ็กเก็ตนั้นจะถูกส่งไปเพื่อการเปรียบเทียบในขั้นตอนอื่นต่อ และการเปรียบเทียบก็จะเกิดขึ้นต่อไป

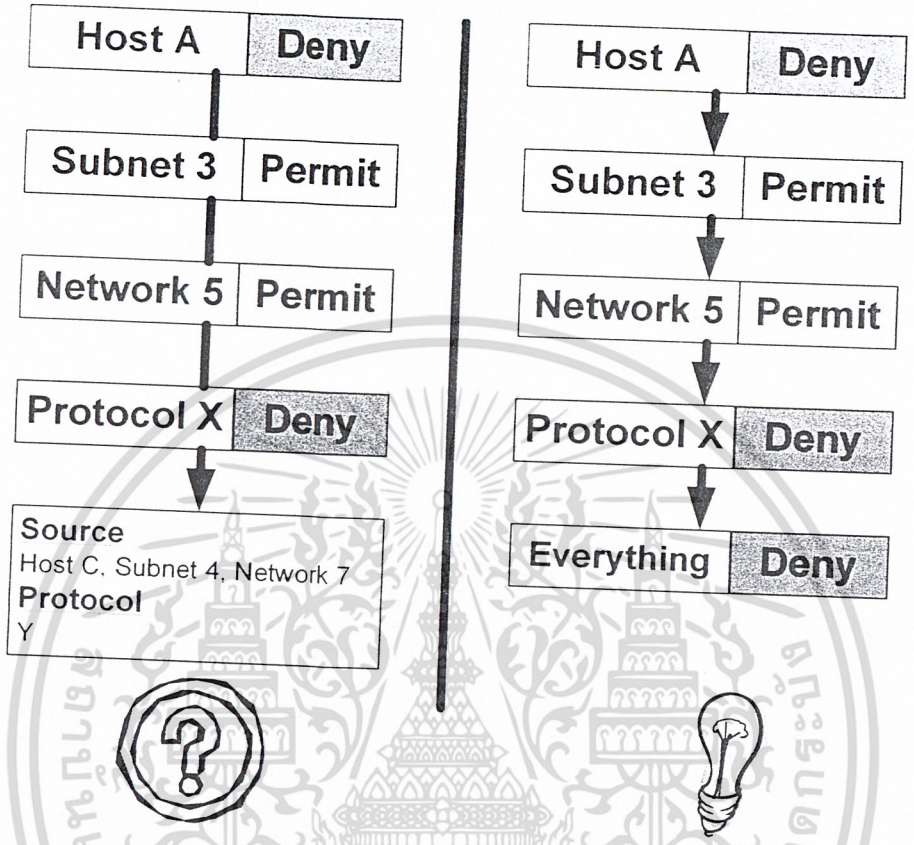


รูปที่ 8-2 ลำดับการทำงานของแอคเซสลิสต์

ในรูปที่ 8-2 คำว่า *Permit* หมายถึงแพ็กเก็ตได้รับอนุญาตให้ออกไปจาก อินเทอร์เฟสอีเทอร์เน็ตหมายเลข 0 (E0) และ *Deny* จะหมายถึงแพ็กเก็ตจะถูกกำจัดออกไป ตามตัวอย่างในรูป สมมุติให้แพ็กเก็ตมีรูปแบบดังนี้

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

เปรียบเทียบกับ Network 5 จะทำให้แพ็กเก็ตนั้นได้รับการส่งออกไปจาก อินเทอร์เฟสอีเทอร์เน็ตหมายเลข 0ในที่สุด



รูปที่ 8-3 ตัวอย่างการจัดการกับแพ็กเก็ตที่ไม่ตรงในแต่ละลำดับชั้น

ส่วนในกรณีที่การเปรียบเทียบในทุกขั้นตอนแล้วไม่ตรงการเปรียบเทียบใดๆ จะใช้ค่าสำรองที่จะมีอยู่แล้วในที่นี้ใน เราเตอร์ของ ซิสโก้ จะเป็น *Deny Any* หรือหมายความว่าให้ทำการกำจัดแพ็กเก็ตที่เหลือออกไป เราสามารถทำกรเปลี่ยนค่าสำรองนี้ได้โดยอาจกำหนด ค่าให้เป็น *Permit Any* เพื่อทำการอนุญาตแพ็กเก็ตที่เหลือจากการเปรียบเทียบทั้งหมด

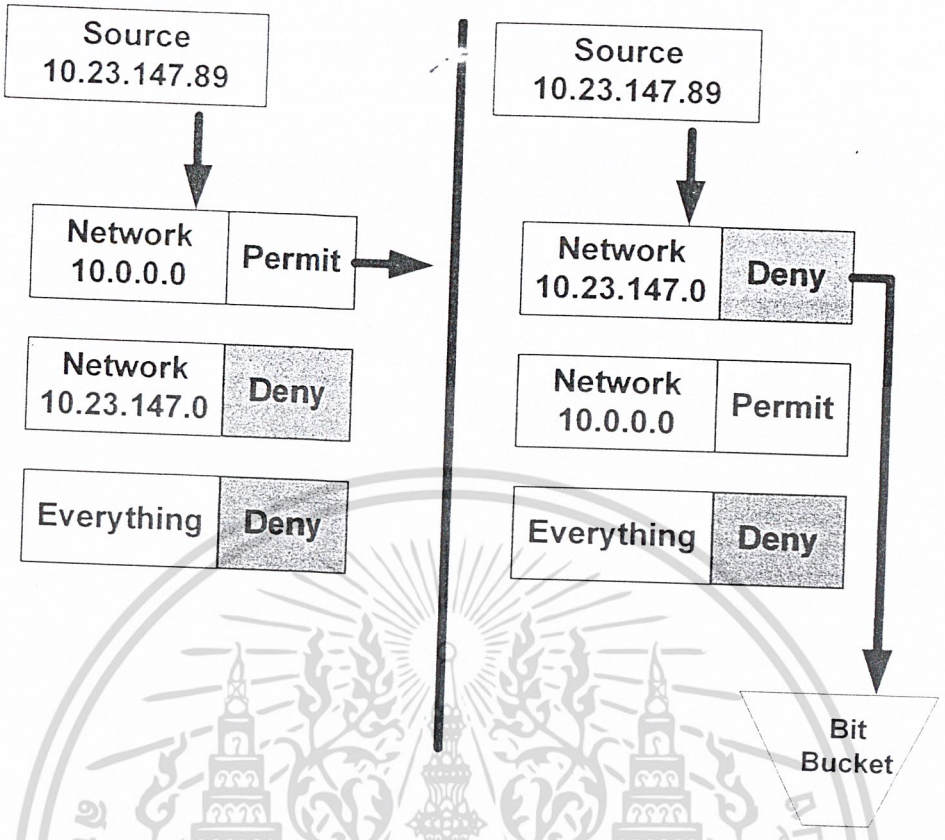
8.2 รูปแบบของแอคเซสลิสต์

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

จากการคอนฟิกค่าของแอคเซสลิสต์ต่อแสดงได้ในรูปที่ 8-4



รูปที่ 8-4 ตัวอย่างการคอนฟิกลำดับชั้นของแอคเซสลิสต์ที่ถูกต้องและไม่ถูกต้อง

ในทุกๆชั้นของแอคเซสลิสต์จะแทนด้วยหนึ่งคำสั่งการคอนฟิกแต่จะเห็นว่ามีเลข 9 ในทั้งสองบรรทัดในการคอนฟิกตามตัวอย่าง ตัวเลขนี้เรียกว่า หมายเลขของแอคเซสลิสต์ ซึ่งจุดประสงค์เพื่อ

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

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 8-1 รูปแบบตัวเลขแอคเซสลิสต์ของซิสโก้

8.3 การแก้ไขค่าแอคเซสลิสต์

การใช้งานของแอคเซสลิสต์จากค่าเก่าที่ได้ทำการคอนฟิกไว้ก่อนแล้วสามารถทำได้โดยใช้คำสั่งการลบแอคเซสลิสต์ดังนี้

no access-list 101 permit tcp 10.2.5.4 0.0.0.255

ในที่นี้แอคเซสลิสต์หมายเลขที่ 101 จะถูกลบออกไป

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

8.4 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

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

8.5 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 นี้เป็น ไอพี *(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 ได้รับการได้รับการอนุญาต

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

8.6 ทีซีพี แอกเซสลิสต์ (TCP Access List)

รูปแบบของการคอนฟิกแบบนี้คือ

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

ในที่นี้จะเป็นเอ็กซ์เทนเด็ดแอกเซสลิสต์(*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

8.7 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 โดยโพรโตคอลนี้จะมีลักษะการเรียกใช้งานเหมือนกับโพรโตคอลทีซีพีแต่ลักษณะที่ต่างกันคือโพรโตคอล 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

8.8 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

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

8.9 การเรียกใช้งานแอคเซ็สลิสต์

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

บทที่ 9

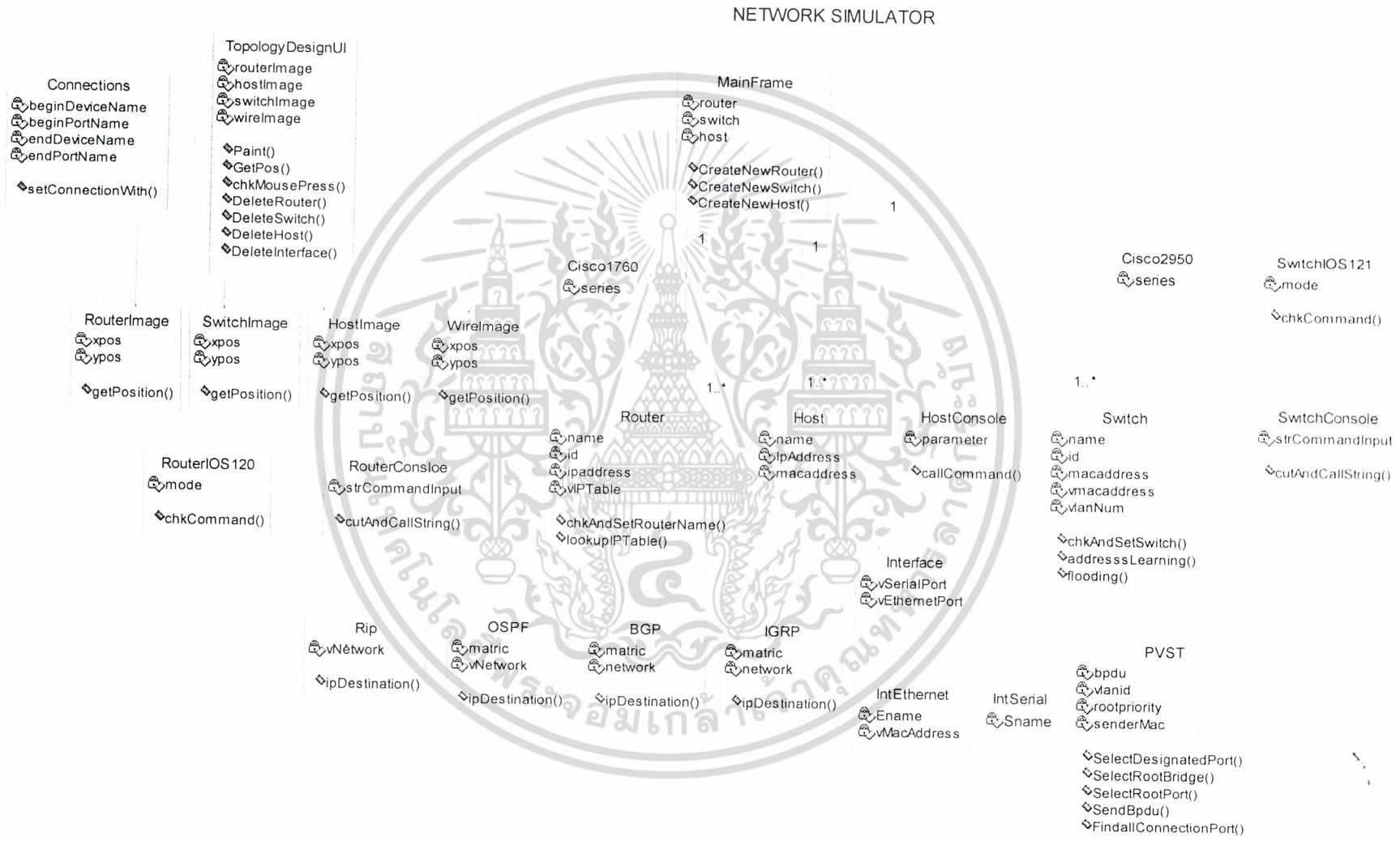
การออกแบบและการสร้างโปรแกรม

9.1 โครงสร้างของโปรแกรม

การออกแบบโปรแกรมจะใช้หลักการออปเจกต์ออเรียนเต็ด (O-O) ซึ่งจะแบ่งการทำงานส่วนต่างๆออกเป็นส่วนย่อยๆ ดังนี้

1. ส่วนติดต่อกับผู้ใช้
2. ส่วนจัดการเราเตอร์
3. ส่วนจัดการสวิตช์
4. ส่วนจัดการพอร์ตและอินเทอร์เฟส

รูปที่ 9-1 แสดงโครงสร้างของโปรแกรม



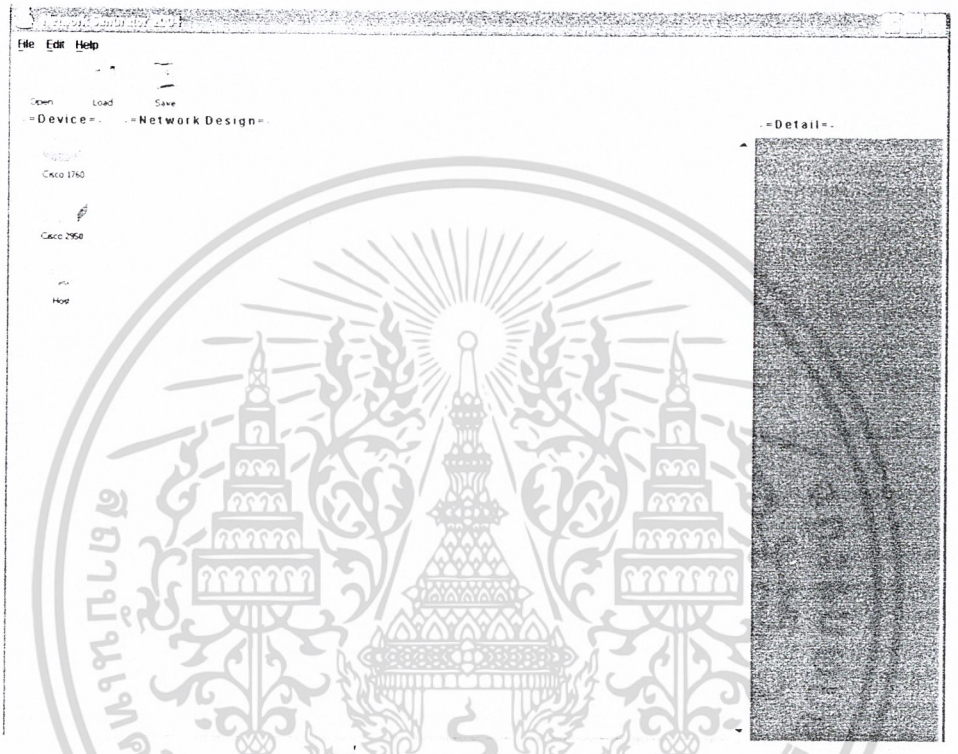
รูปที่ 9-2 แสดง class diagram ของโปรแกรม

9.2 รายละเอียดส่วนต่างๆของคลาส

9.2.1 ส่วนติดต่อกับผู้ใช้

- class MainFrame

เป็นคลาสที่ใช้สำหรับแสดงหน้าจอหลัก ภายในประกอบไปด้วย Tool Panel, Device Panel, Detail Panel และเป็นส่วนที่เรียกใช้ในการสร้างอุปกรณ์เครือข่าย



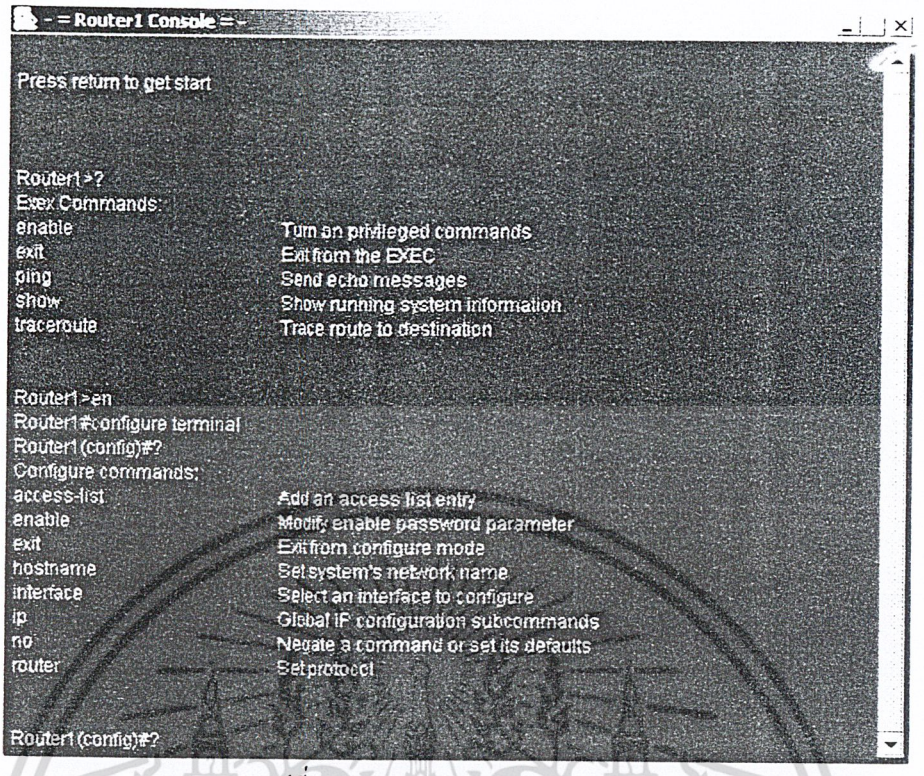
รูปที่ 9-3 แสดงคลาส MainFrame

- class TopologyDesignUI

เป็นคลาสที่ใช้สำหรับ แสดงผลและรับเมาส์อีเวนท์ของโปรแกรมในการเชื่อมต่อ หรือออกแบบเครือข่าย โดยคลาสนี้ถูกสร้างแยกออกจากคลาส Mainframe เพื่อสะดวกในการทำงานโต้ตอบกับผู้ใช้ และจะถูกบรรจุอยู่ในคลาส MainFrame ด้วย

- class RouterConsole

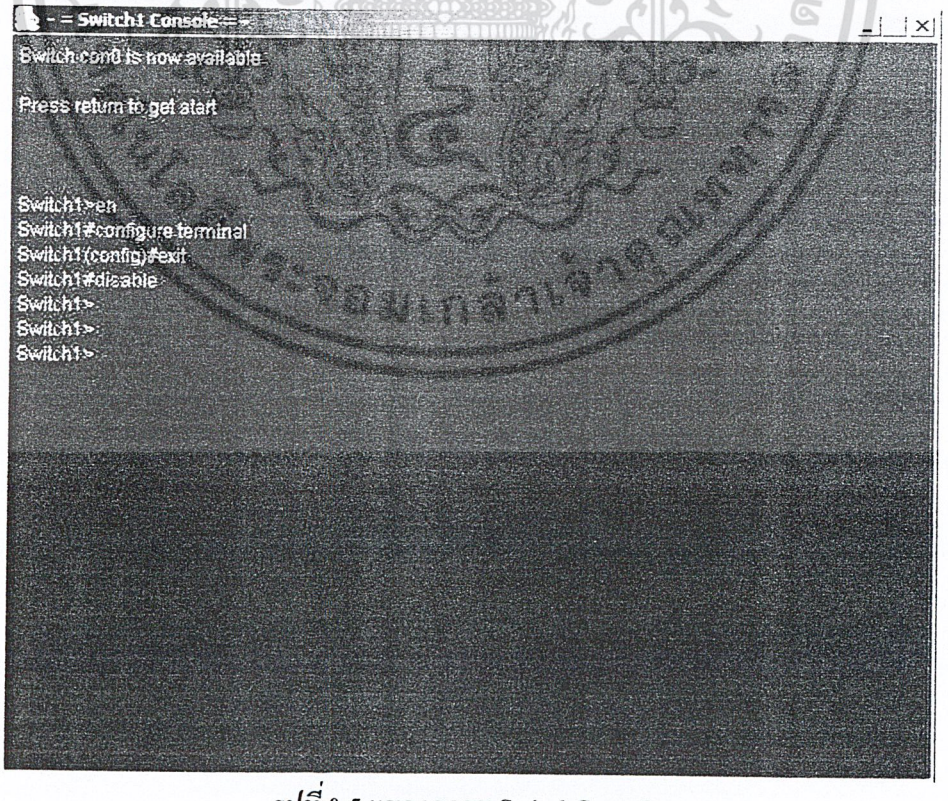
เป็นคลาสที่ติดต่อกับผู้ใช้แบบ command line โดยจะให้ผู้ใช้ทำการ ป้อนคำสั่งลงไปเพื่อให้ Router Console เรียกใช้งานจาก RouterIOS120 ต่อไป โดยส่วนสำคัญจะอยู่ที่การตัดคำสั่งเพื่อส่งไปทำงาน



รูปที่ 9-4 แสดงคลาส RouterConsole

- class SwitchConsole

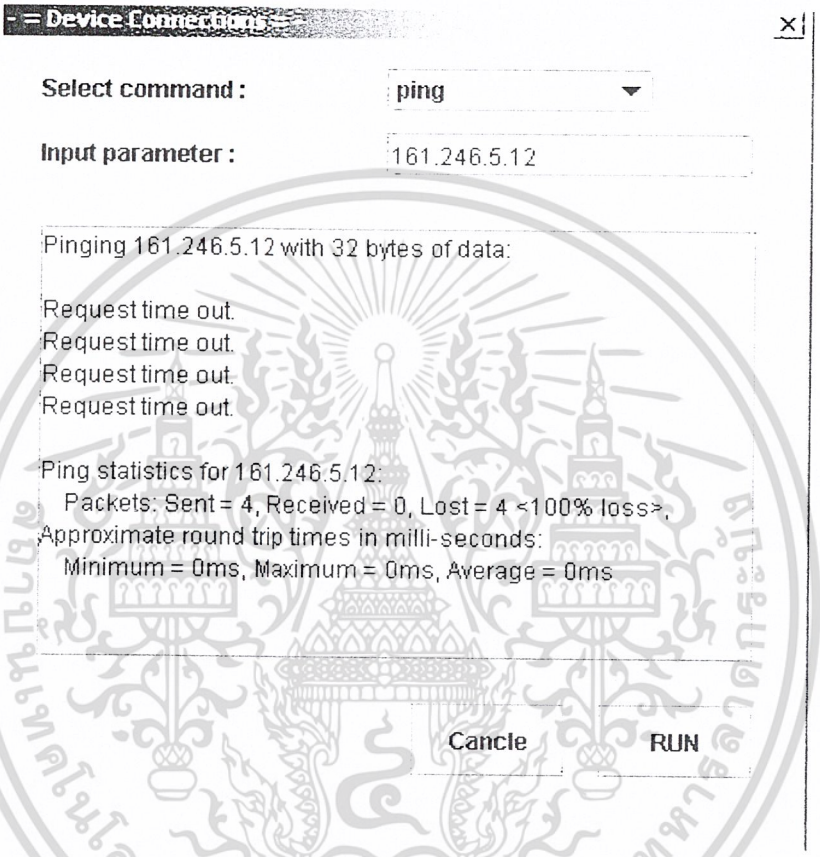
เป็นคลาสที่ติดต่อกับผู้ใช้แบบ command line เหมือนกันกับคลาส RouterConsole แต่ในการเรียกใช้งานจะส่งคำสั่งไปเรียกใช้งานจากคลาส SwitchIOS121



รูปที่ 9-5 แสดงคลาส SwitchConsole

- class Host และ HostConsole

คลาส Host จะทำการสร้าง MacAddress ประจำเครื่องขึ้นมา ส่วนคลาส HostConsole นี้จะต่างจาก RouterConsole และ SwitchConsole ตรงที่ในการติดต่อกับผู้ใช้นั้น จะเป็น Dialog ให้ผู้ใช้ป้อนค่าพารามิเตอร์ไม่ใช่ผ่าน command line โดยคำสั่งต่างๆที่จะต้องใช้จะอยู่ใน ComboBox และพารามิเตอร์จะต้องป้อนใน jTextField นอกจากนั้นการส่งค่าไปทำงานจะต้องใช้ปุ่ม Run Button



รูปที่ 9-6 แสดงคลาส HostConsole

9.2.2 ส่วนจัดการเราเตอร์

- class Router

คลาสนี้จะเป็นคลาสที่ใช้สำหรับเก็บรายละเอียดของเราเตอร์รวมทั้งการกระทำและเมทธอทต่างๆ โดยส่วนที่สำคัญสำหรับคลาสนี้คือส่วนของการ chkAndSetRouterName และส่วนในการ lookupIPTable ซึ่งจะใช้ในการค้นหา ip ปลายทาง

- class Router1760

คลาสนี้เป็นคลาสที่ถูกสืบทอดมาจากคลาส Router ซึ่งจะรองรับเราเตอร์รุ่นใหม่ๆได้ในอนาคต

- class RouterIOS120

- คลาสนี้เป็นคลาสที่สำคัญ มีไว้เพื่อกำหนดการทำงานต่างๆของเราเตอร์ เช่น setHostname, setIPAddress เป็นต้น โดยภายในจะบรรจุคำสั่งต่างๆที่ใช้ในการ config เราเตอร์ซิสโก้เวอร์ชั่น 12.0
- class RouterImg

คลาสนี้เป็นคลาสที่มีไว้เพื่อกำหนดรูปภาพของเราเตอร์ในการแสดงผล รวมทั้งเมื่อมีการเปลี่ยนแปลงตำแหน่งของเราเตอร์ไปยังตำแหน่งต่างๆ ก็จะทำการ repaint ให้ตรงกับที่ผู้ใช้ต้องการอีกด้วย

- class RIP

เป็นคลาสที่จัดการเกี่ยวกับ การเลือกเส้นทางแบบ RIP โดยคลาสนี้มีการทำงานเป็น Threads รับส่งตารางเลือกเส้นทางเป็นช่วงเวลาที่ตั้งไว้ตามข้อกำหนดของโพรโตคอล

- class OSPF

เป็นคลาสที่จัดการเกี่ยวกับ การเลือกเส้นทางแบบ OSPF มีการทำงานเป็น Threads

9.2.3 ส่วนจัดการสวิตช์

- class Switch

เป็นคลาสที่เก็บรายละเอียดแอตทริบิวของสวิตช์รวมถึงแมทธอทการทำงานต่างๆ โดยส่วนสำคัญของคลาสนี้คือส่วนการทำ AddressLearning และการ Flooding

- class Switch2950

เป็นคลาสที่สืบทอดมาจากคลาส Switch มีไว้เพื่อแสดงว่ามีสามารถสร้างสวิตช์ซีรีส์ใหม่ๆได้ในอนาคต โดยระบุจำนวนพอร์ตที่ต้องการใช้งาน

- class SwitchIOS

เป็นคลาสที่รับค่ามาจาก SwitchConsole แล้วทำการเปรียบเทียบกับคำสั่งที่มีอยู่ เมื่อพอว่าคำสั่งตรงกันจึงเรียกใช้งานคำสั่งนั้น

- class SwitchImg

เป็นคลาสที่เก็บรูปสวิตช์ในการแสดงผล รวมถึงตำแหน่งที่จะแสดงผลให้ผู้ใช้ดูอีกด้วย

- class STP

เป็นคลาสที่มีไว้จัดการเกี่ยวกับการทำ สแปนนิ่งทรีโปรโตคอล มีการทำงานเป็น Threads

- class PVST

เป็นคลาสที่ไว้จัดการเกี่ยวกับการทำเปอร์วีแลนสแปนนิ่งทรี มีการทำงานเป็น Threads เพื่อกำจัด ลูปใน เลเยอร์สองสวิตช์

9.2.4 ส่วนจัดการพอร์ตและอินเตอร์เฟส

- class Interface

เป็นคลาสที่มีไว้สร้าง port ให้กับอุปกรณ์ต่างๆที่เรียกใช้ โดยจะต้องบอกจำนวนของพอร์ตแต่ละประเภทให้ หลังจากนั้นจะมีเมทธอทสำหรับ กำหนดชื่อและรายละเอียดของแต่ละพอร์ตว่ามีอะไรบ้างอีกด้วย

- class IntSerial

เป็นคลาสที่จัดการเกี่ยวกับการทำงานของพอร์ต Serial

- class IntEthernet

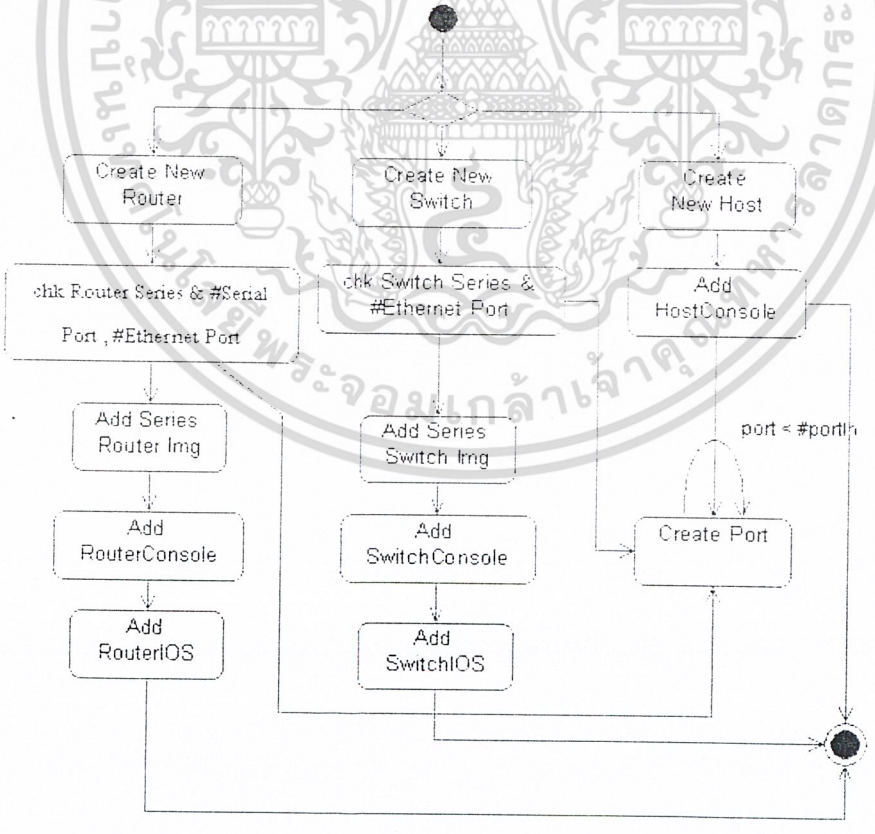
เป็นคลาสที่จัดการเกี่ยวกับการทำงานของพอร์ต Ethetnet

- class Connections

เป็นคลาสที่คอยบอกว่า อุปกรณ์ใดต่อกันอยู่ และใช้พอร์ตใดบ้างในการเชื่อมต่อ หลังจากนั้นก็จะทำการส่งไปบอกให้ class TopologyDesignUI ทำการวาดรูปว่ามีการเชื่อมต่อแบบใดอยู่บ้าง

9.3 ขั้นตอนการทำงานของโปรแกรม

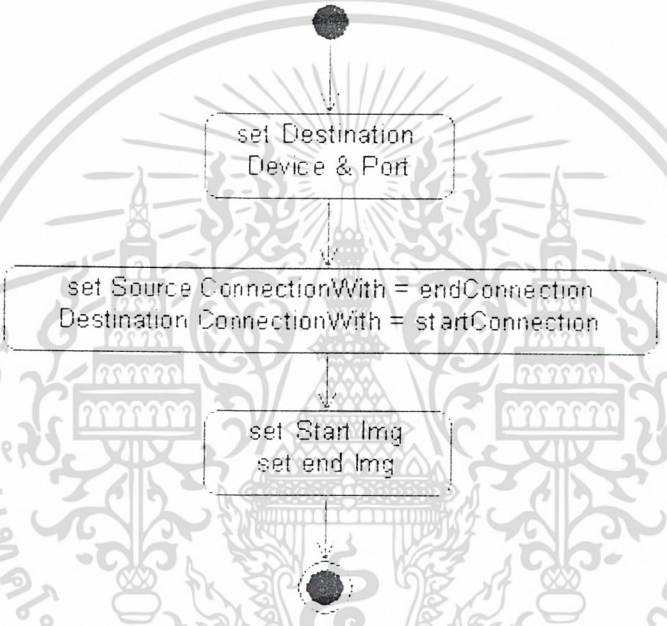
9.3.1 การทำงานของ MainFrame



รูปที่ 9-7 แสดงขั้นตอนการทำงานของ MainFrame

MainFrame จะเริ่มจากรอให้ผู้ใช้เลือกการทำงาน เช่นการสร้าง Router, Switch หรือ Host หลังจากที่ผู้ใช้เลือกว่าจะสร้างอุปกรณ์ใดแล้ว โปรแกรมจะทำการ get ค่าว่าจะสร้างพอร์ตเป็นจำนวนเท่าไหร่จาก series ของอุปกรณ์นั้น หรือจากอุปกรณ์ที่ผู้ใช้กำหนดขึ้นเอง แต่ว่าหากสร้าง Host โปรแกรมจะทำการสร้างพอร์ตแบบ Ethernet เพียง 1 พอร์ตเท่านั้น และหลังจากสร้างพอร์ตเสร็จแล้ว โปรแกรมจะทำการ add อุปกรณ์ลงใน vector ของ MainFrame และ ใน TopologyDesignUI ซึ่งการ add ดังกล่าวใน TopologyDesignUI จะเป็นการ add image ของอุปกรณ์ที่สร้างขึ้นมา เมื่อ add เสร็จเรียบร้อยแล้ว โปรแกรมจะทำการ add console เข้าไปใน vector เช่นกัน

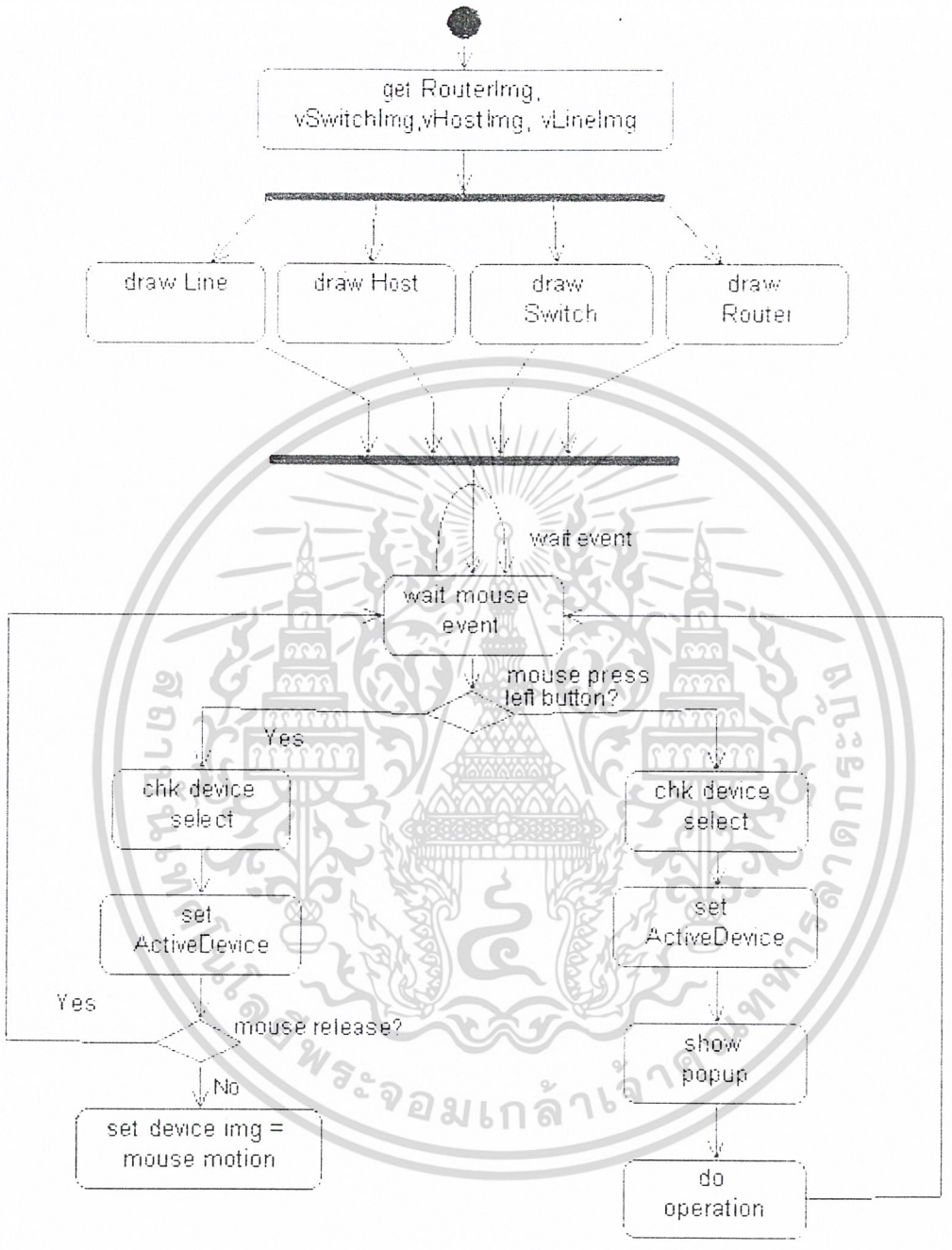
9.3.2 การทำงานของ Connections



รูปที่ 9-8 แสดงการทำงานของ Connections

ในการทำงานของ Connections อาจดูง่าย เพียงแต่การทำงานจะยุ่งยากมากในการ ตรวจสอบว่า ต้นทางที่จะต่อนั้นเป็นอุปกรณ์ใด และจากพอร์ตใด หลังจากที่ รู้แล้วว่า ต้นทางมาจากที่ใดแล้ว ต่อไปก็จะเป็นการตรวจสอบปลายทาง ว่าต่ออยู่กับอุปกรณ์ใด พอร์ตใด หลังจากที่ทราบแล้วว่าอุปกรณ์ใดต่อกันอยู่แล้ว ก็จะทำการ set ค่า ปลายทางของกันและกันเก็บไว้ใน attribute ของอุปกรณ์ที่ต่อกันอยู่ และ add ค่าลงใน vector ที่จะใช้วาดรูป

9.3.3 การทำงานของ TopologyDesignUI

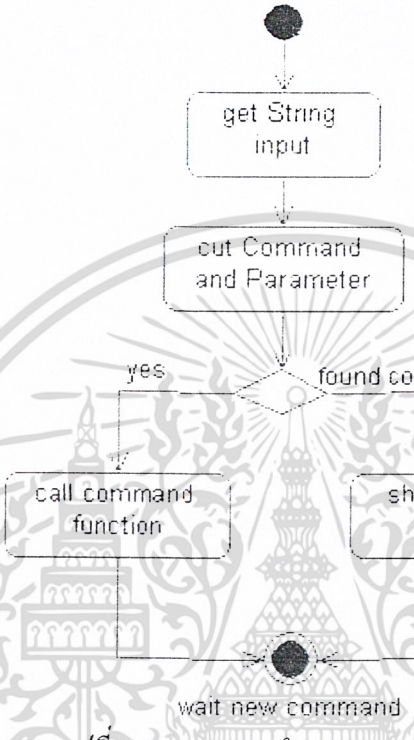


รูปที่ 9-9 การทำงานของ TopologyDesignUI

การทำงานของ TopologyDesignUI นั้น จะเริ่มจากการที่ไป get ค่าของอุปกรณ์ที่ต้องการจะวาดรูป มาทีละค่า แล้วทำการวาด โดยการวาดนั้นจะเริ่มจากการวาด เส้นก่อน, หลังจากนั้นก็จะวาด Host, Switch และ Router ตามลำดับ เมื่อวาดเสร็จแล้ว ก็จะเป็นการรอรับ event จากผู้ใช้ว่าต้องการทำอะไร โดยถ้ามีการคลิกที่อุปกรณ์ตัวใดก็ตาม อันดับแรกจะเป็นการ active ให้อุปกรณ์นั้น แสดงผลอยู่หน้าสุด และรอรับคำสั่งจากผู้ใช้ต่อไป หากเป็นการคลิกค้างแล้วทำการเคลื่อนย้ายเมาส์ โปรแกรม

ก็จะทำการ get ค่าตำแหน่งของเมาส์ที่ชี้บนอุปกรณ์ตัวนั้น แล้วทำการ set ค่าเพื่อวาดใหม่ หากผู้ใช้ทำการคลิกขวาบนรูปอุปกรณ์ใดๆ โปรแกรมจะทำการเรียกใช้ popup ของอุปกรณ์ตัวนั้นขึ้นมาให้ผู้ใช้สั่งงานตาม operation ต่างๆที่แสดงขึ้น

9.3.4 การทำงานของ Console



รูปที่ 9-10 แสดงการทำงานของ Console

การทำงานของคอนโซลจะเริ่มจากการที่ ผู้ใช้ ป้อนคำสั่งไปบน command line ที่รอรับอยู่ หลังจากกด Enter แล้ว โปรแกรมจะทำการนำค่า String ล่าสุดที่ป้อนลงไป ทำการ ตัดคำ ออกเป็นส่วนๆ หลังจากนั้น จะนำเอาส่วนแรก มาตรวจสอบว่ามีอยู่ใน vector หรือไม่ ถ้ามีก็จะเป็นการเรียกคำสั่งนั้นขึ้นมาใช้งาน

9.3.5 การทำงานของ PVST(Per-VLAN Spanning Tree)

รูปที่ 9-11 แสดงการทำงานของ Per-VLAN Spanning Tree

Create Port Buffer-สร้าง buffer สำหรับแต่ละ port ขึ้นมาในที่นี้คือ switch 2950 มีจำนวน 24 ports เอาไว้สำหรับรับข้อมูลที่ switch ตัวอื่นส่งมาให้ เนื่องจาก Per-VLAN Spanning-Tree แต่ละตัวทำงานเป็น 1 thread ดังนั้นจึงจำเป็นต้องมีส่วน shared memory เอาไว้ติดต่อกันระหว่าง thread

Select Root Bridge-ทำการหา root bridge โดยเลือกจาก switch ตัวที่มี bridge id ต่ำสุดให้เป็น root bridge

Select Root Port-switch ตัวที่ไม่ใช่ root bridge จะทำการเลือก root port (เส้นทางที่มีค่า cost ที่ต่ำที่สุดที่ไปยัง root bridge) โดยเลือกจาก ค่า root path cost ภายใน bpdu ที่ได้รับมาจากตัวข้างเคียง ของ

แต่ละ port นำค่า root path cost ที่ต่ำสุดที่หาได้นั้นมาบวกกับค่า path cost ของ link ที่ต่อกับ port นั้น แล้วเก็บเป็นค่า root path cost ของหากมีมากกว่า 1 port ที่ bpdu มีค่า root path cost เท่ากัน ให้เลือกจาก port ที่หมายเลขต่ำสุดเป็น root port และตัวเอง เพื่อใช้ในการส่ง bpdu ให้กับตัวข้างเคียง

Select Designated Port-หากภายใน segment หนึ่งนั้น packet สามารถไปได้มากกว่าหนึ่งทางจะทำให้เกิด loop ภายในระบบซึ่งทำให้สูญเสีย band width และ processing time ของ ระบบ ดังนั้นจะต้องมีทางเดียวเท่านั้นที่ packet สามารถผ่านไปได้ เรียกว่า "designated port" โดยทำการเลือก port จาก switch ตัวที่มี root path cost ที่ต่ำสุดถ้าเท่ากันให้เลือกจาก switch ที่มี bridge id ต่ำสุด

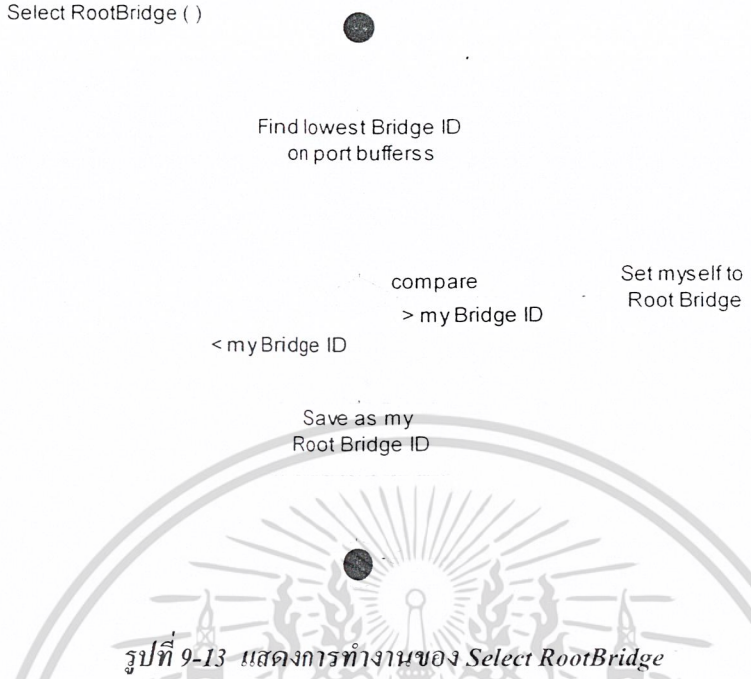
Send Bpdu-ทำการส่ง bpdu ไปให้ตัวข้างเคียงทุกๆ hello time

ตัวอย่าง

ลักษณะการทำงานของ PVST

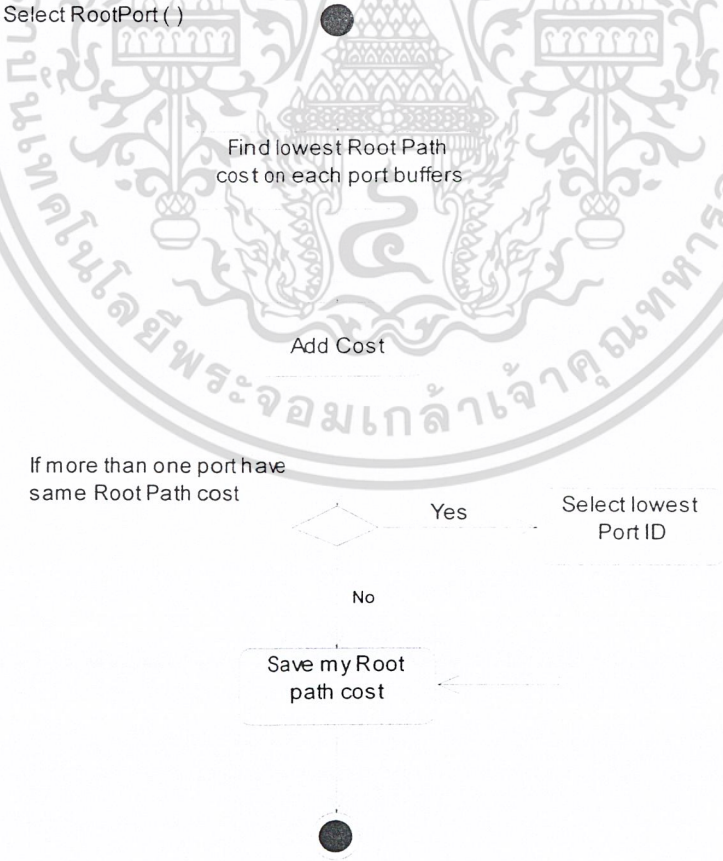
รูปที่ 9-12 แสดงตัวอย่างและลักษณะการทำงานของ Per-VLAN Spanning Tree

การทำงานของ Select RootBridge



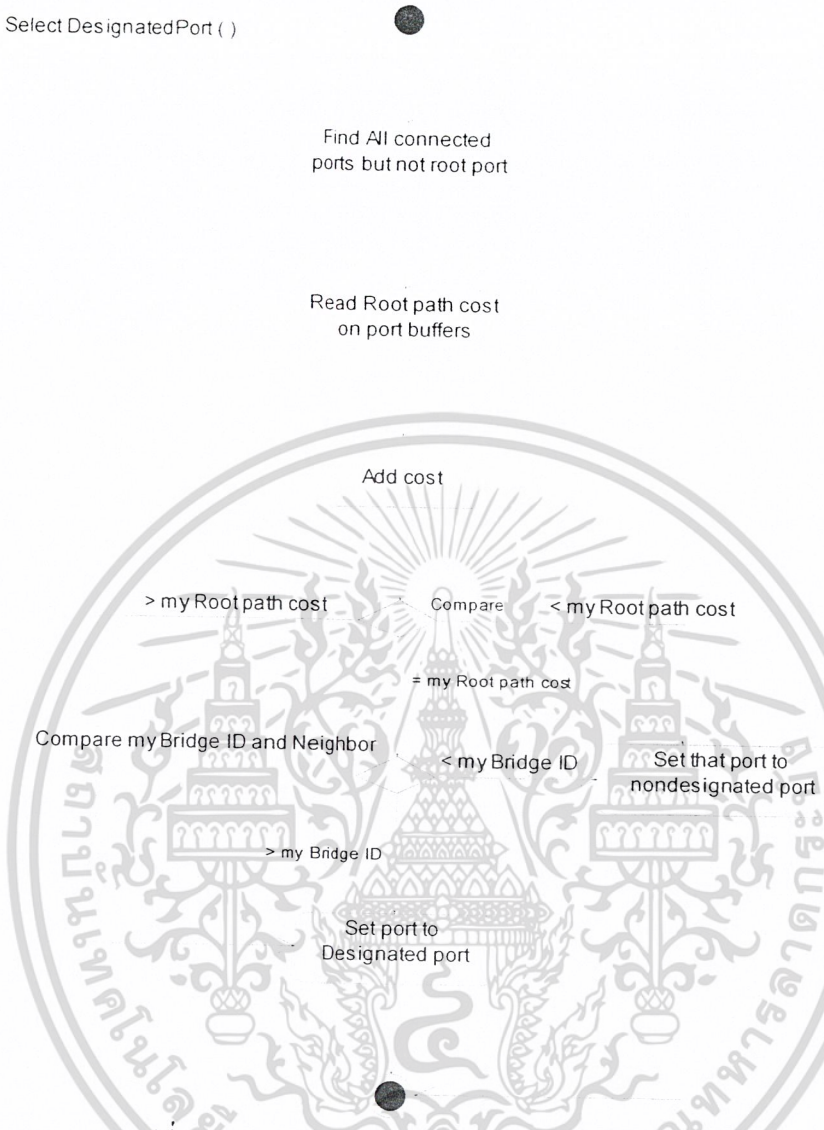
รูปที่ 9-13 แสดงการทำงานของ Select RootBridge

การทำงานของ **Select RootPort**



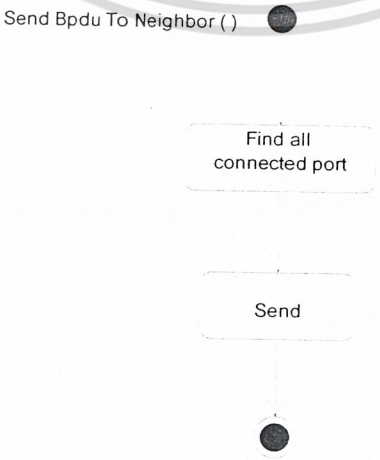
รูปที่ 9-14 แสดงการทำงานของ Select RootPort

การทำงานของ Select DesignatedPort



รูปที่ 9-15 แสดงการทำงานของ Select DesignatedPort

การทำงานของ Send Bpdu To Neighbor



รูปที่ 9-16 แสดงการทำงานของ Send Bpdu To Neighbor

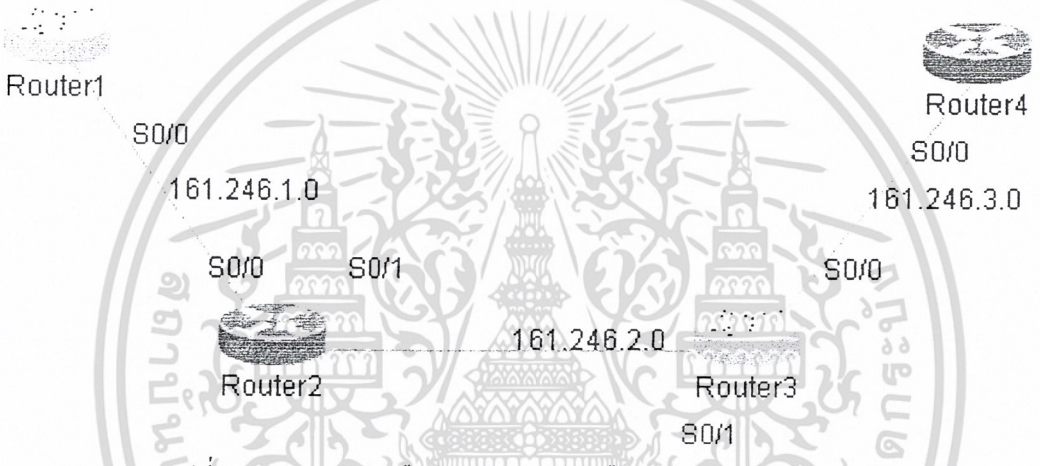
บทที่ 10
ตัวอย่างและการทดสอบการทำงานของโปรแกรม

10.1 ตัวอย่างการเลือกเส้นทางของเราเตอร์

10.1.1 ตัวอย่างการเลือกเส้นทางแบบสแตติก

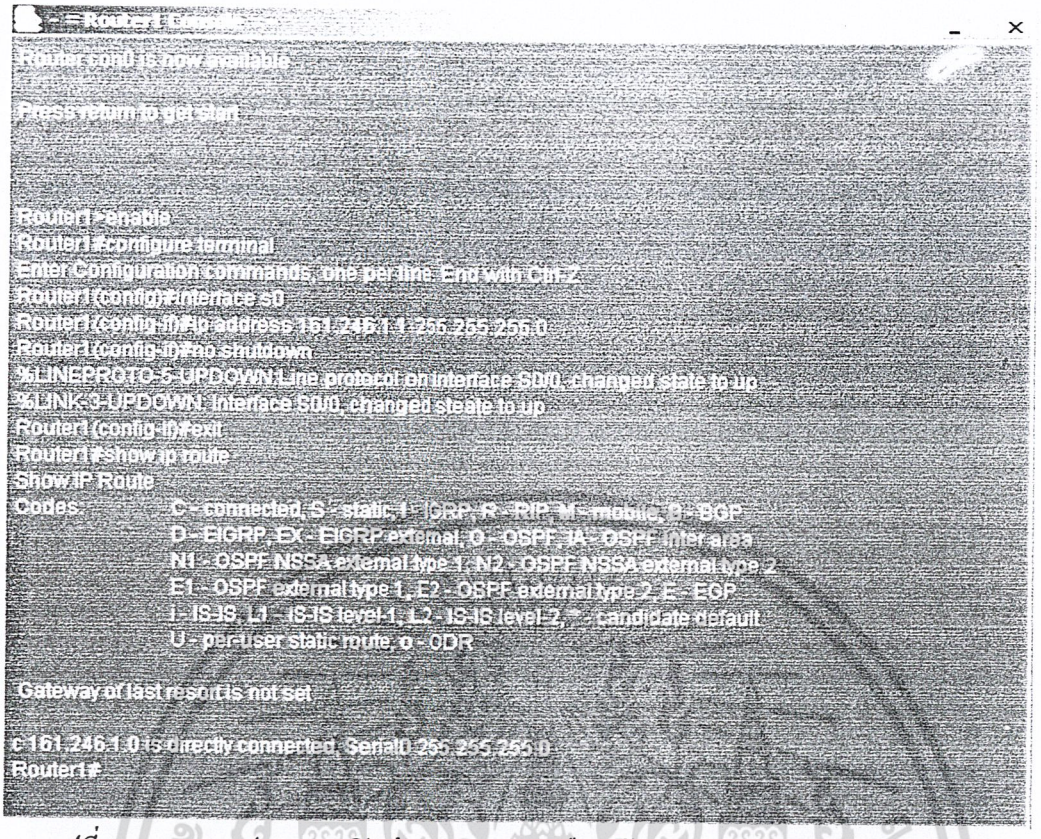
ตัวอย่างนี้เป็นการคอนฟิกเราเตอร์โดยใช้การเลือกเส้นทางแบบสแตติกมีลำดับการทำงานดังนี้

1. เปิดโปรแกรมขึ้นมาและทำการสร้างเครือข่ายดังรูป

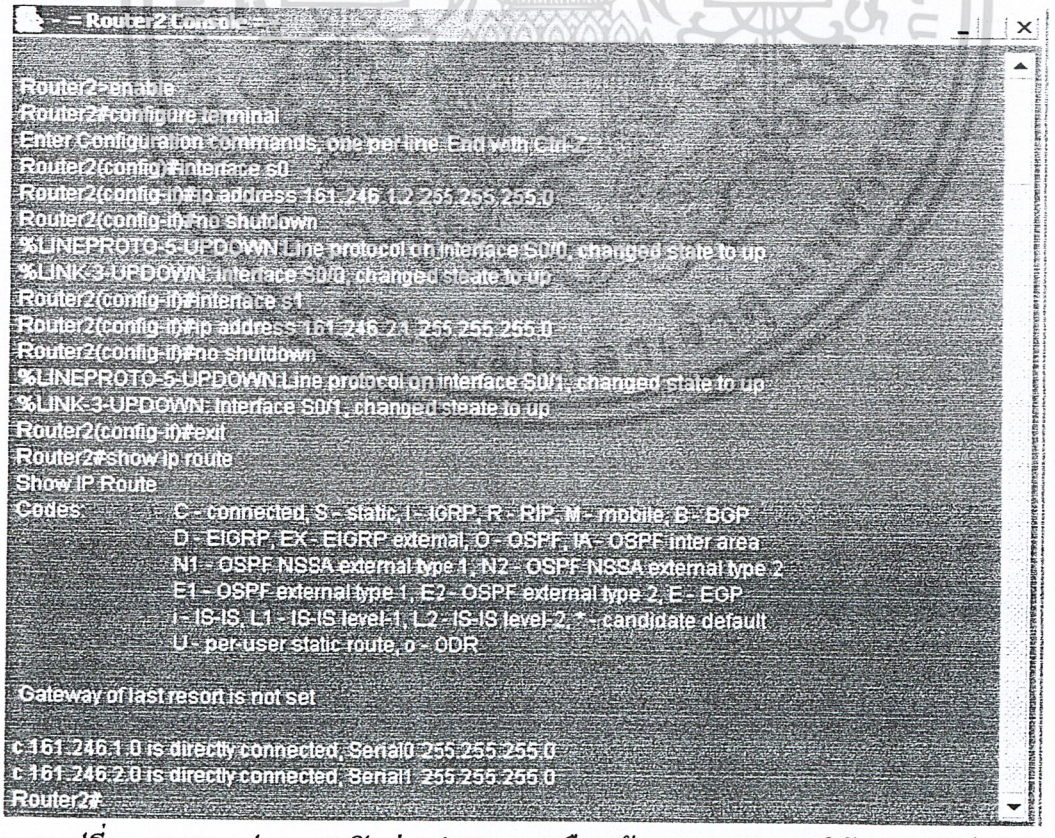


รูปที่ 10- 1 แสดงรูปเครือข่ายทดสอบการเลือกเส้นทางแบบสแตติก

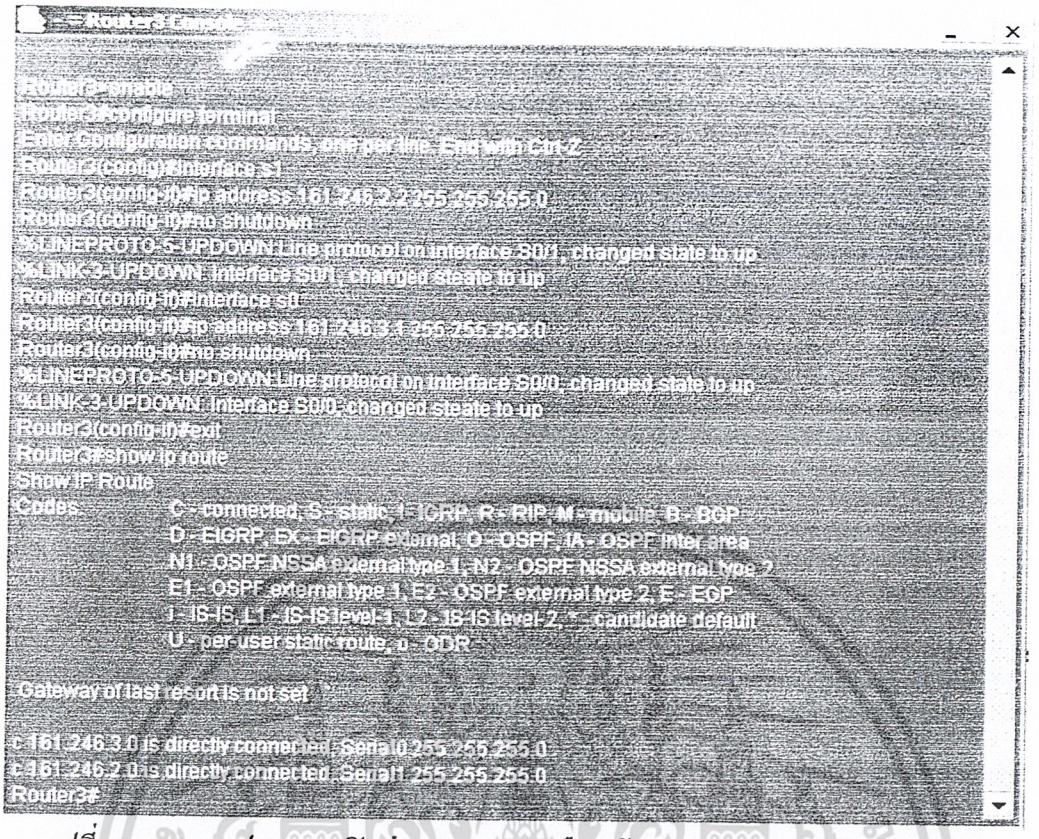
2. ดับเบิ้ลคลิกที่เราเตอร์1 ถึง เราเตอร์4 เพื่อเรียกคอนโซลขึ้นมาทำการคอนฟิกโดยกำหนดอินเตอร์เฟสตามเครือข่ายที่สร้างขึ้นแล้วกำหนดแต่ละอินเตอร์เฟสให้อยู่ในสถานะที่ทำงาน(UP) จากนั้นลองใช้คำสั่ง show ip route เพื่อดูค่าตารางเลือกเส้นทางดังตัวอย่าง



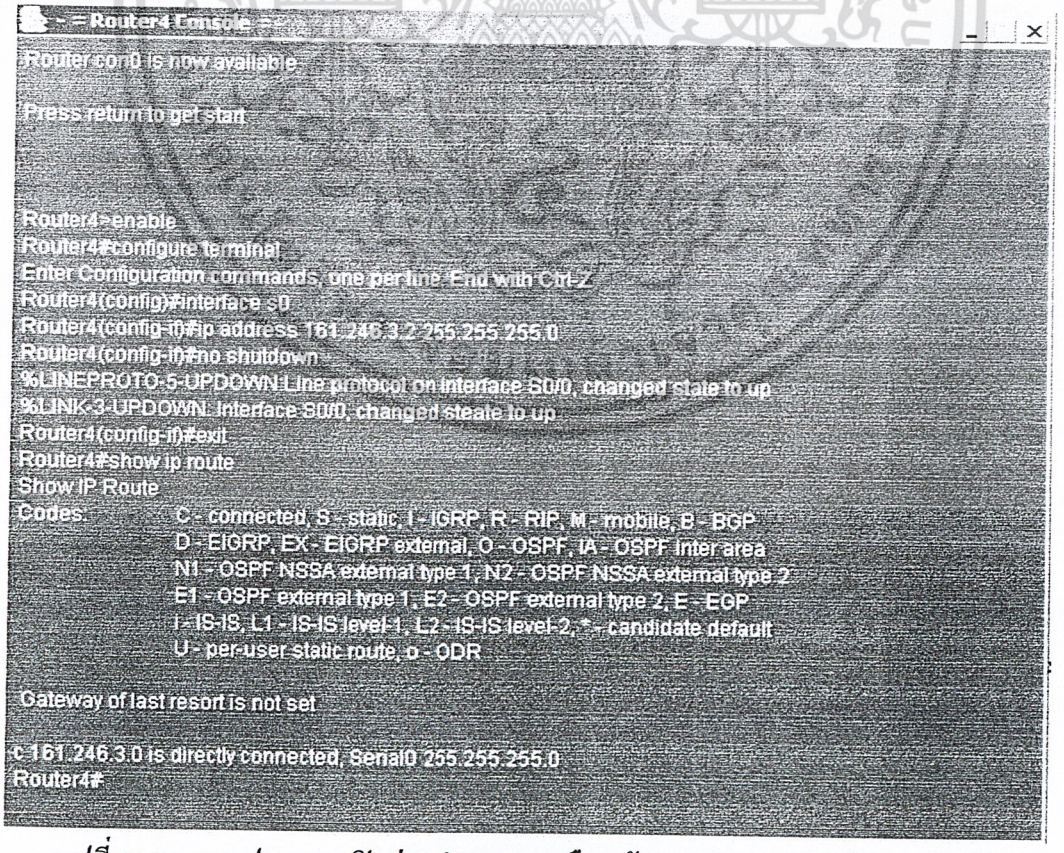
รูปที่ 10- 2 แสดงรูปการคอนฟิกก่อนกำหนดการเลือกเส้นทางแบบสแตติกให้กับเราเตอร์1



รูปที่ 10-3 แสดงรูปการคอนฟิกก่อนกำหนดการเลือกเส้นทางแบบสแตติกให้กับเราเตอร์2

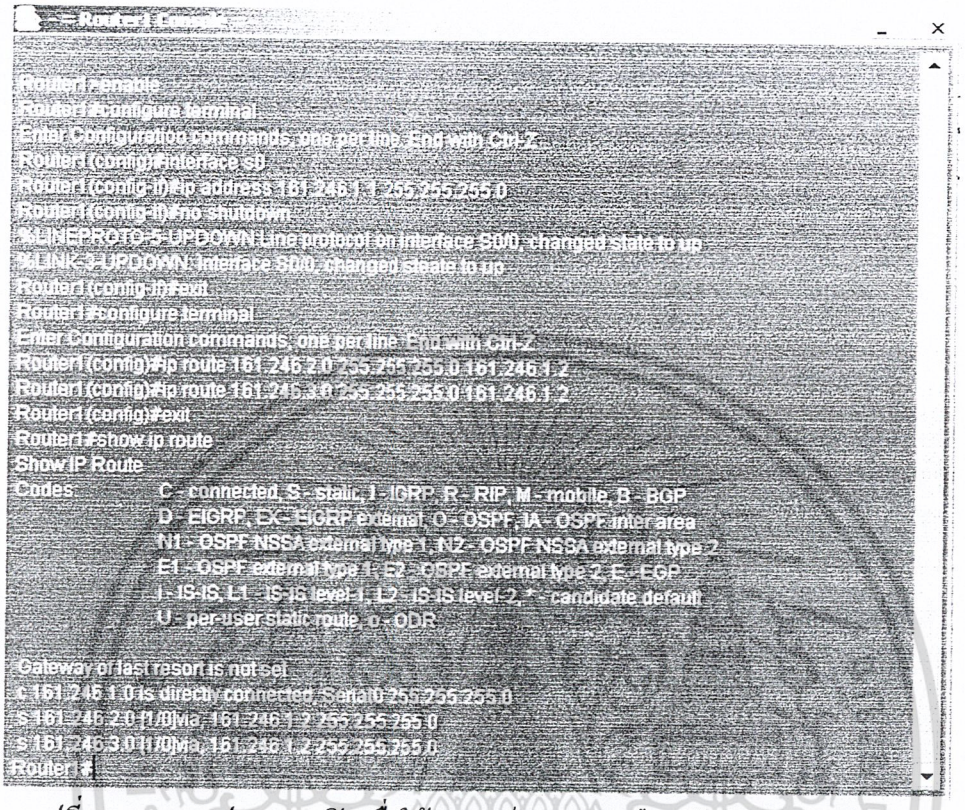


รูปที่ 10- 4 แสดงรูปการคอนฟิกก่อนกำหนดการเลือกเส้นทางแบบสแตติกให้กับเราเตอร์3

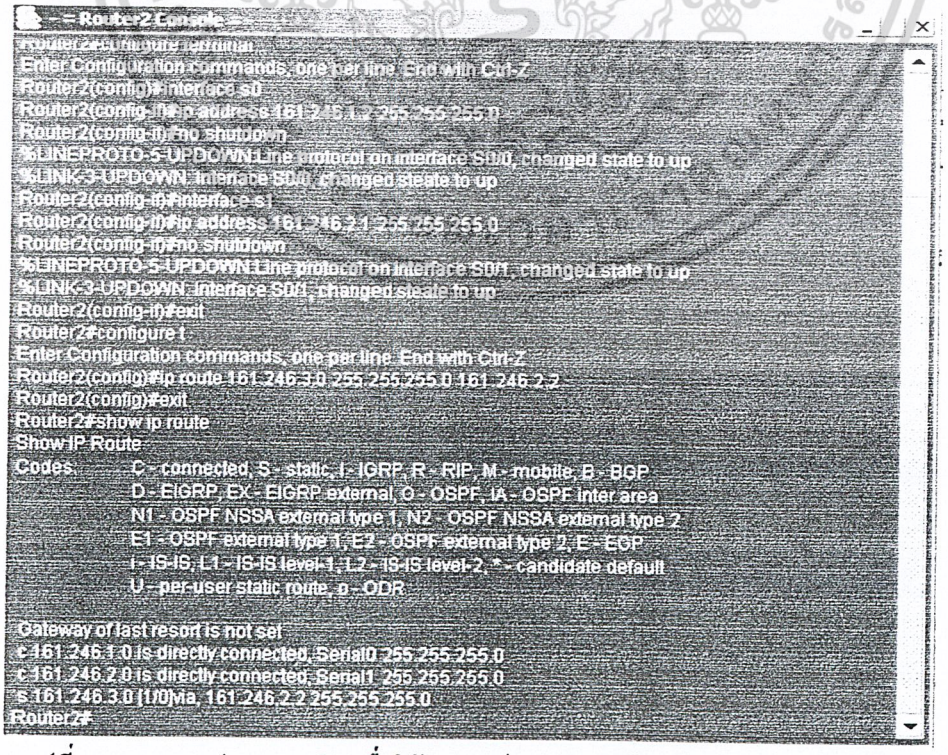


รูปที่ 10-5 แสดงรูปการคอนฟิกก่อนกำหนดการเลือกเส้นทางแบบสแตติกให้กับเราเตอร์4

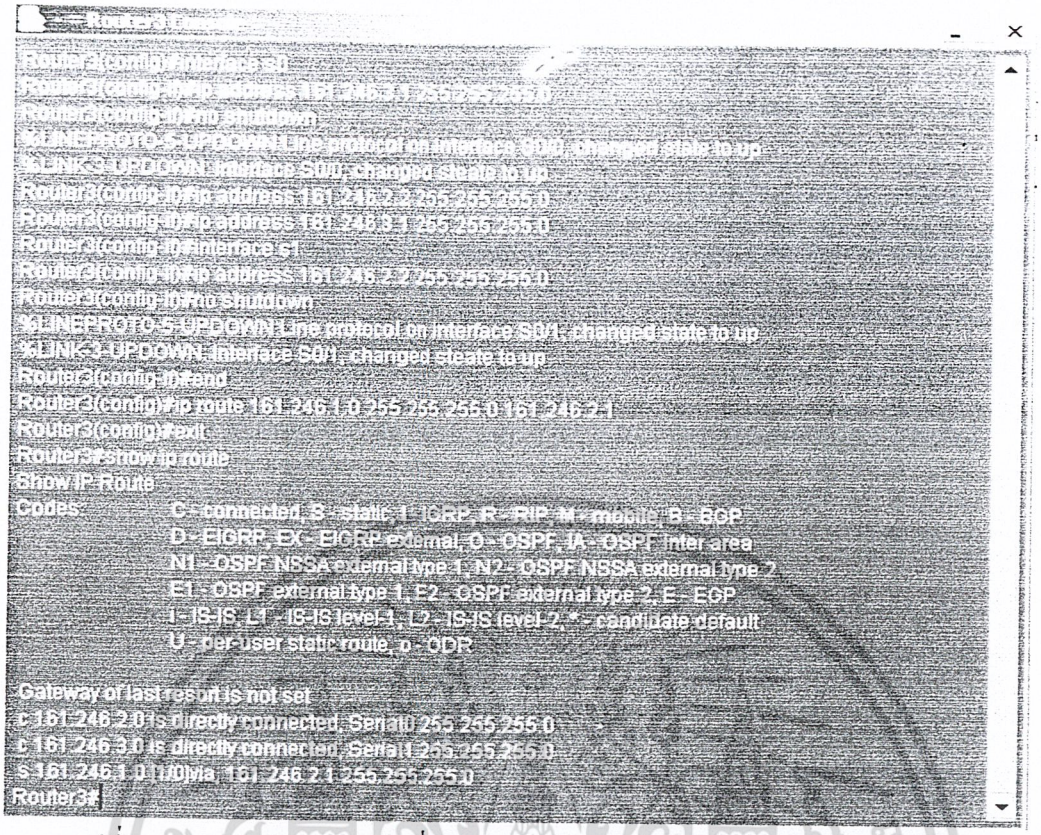
3. หลังจากคอนฟิกค่าให้แต่ละอินเทอร์เฟสแล้ว ต่อไปก็ให้ทำการคอนฟิกเราเตอร์แต่ละตัวให้สามารถเราท์เส้นทางไปยัง เราเตอร์ตัวอื่นๆได้โดยใช้คำสั่ง ip route <destination network ip address> <subnet mask> <destination interface ip address>



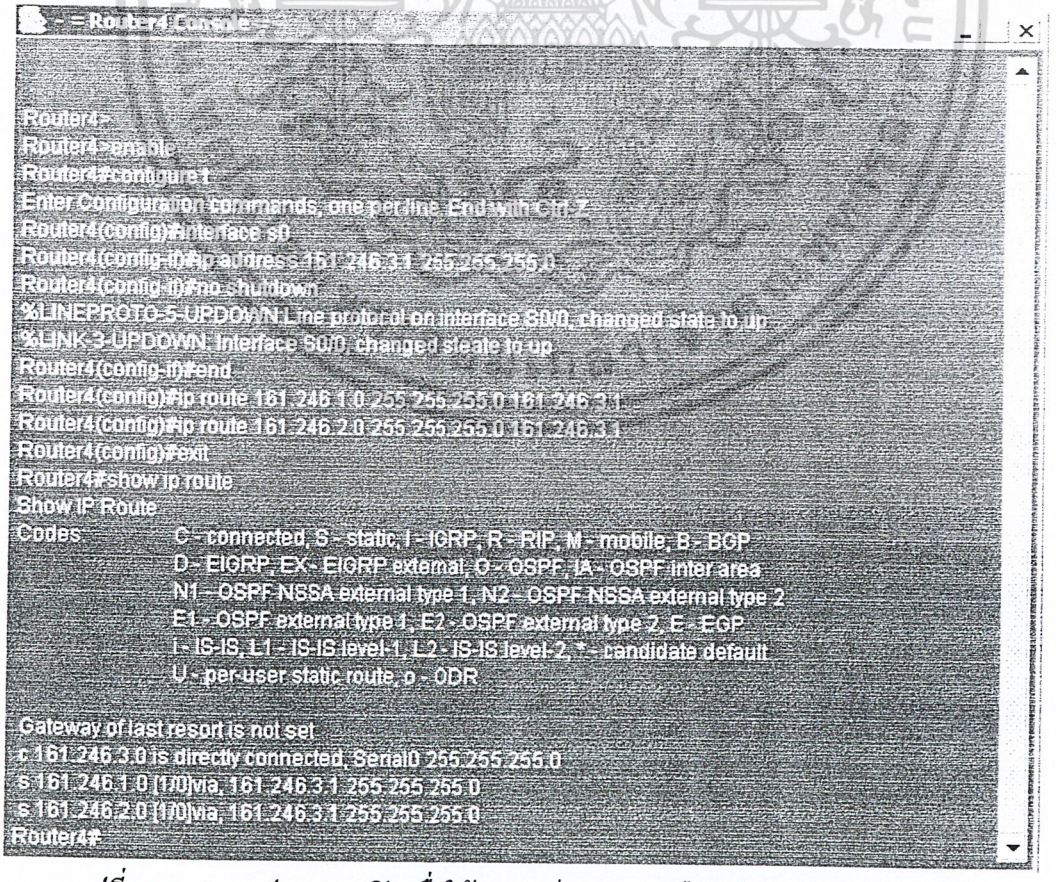
รูปที่ 10-6 แสดงรูปการคอนฟิกเพื่อให้เราเตอร์1 สามารถเลือกเส้นทางแบบสแตติกได้



รูปที่ 10-7 แสดงรูปการคอนฟิกเพื่อให้เราเตอร์2 สามารถเลือกเส้นทางแบบสแตติกได้

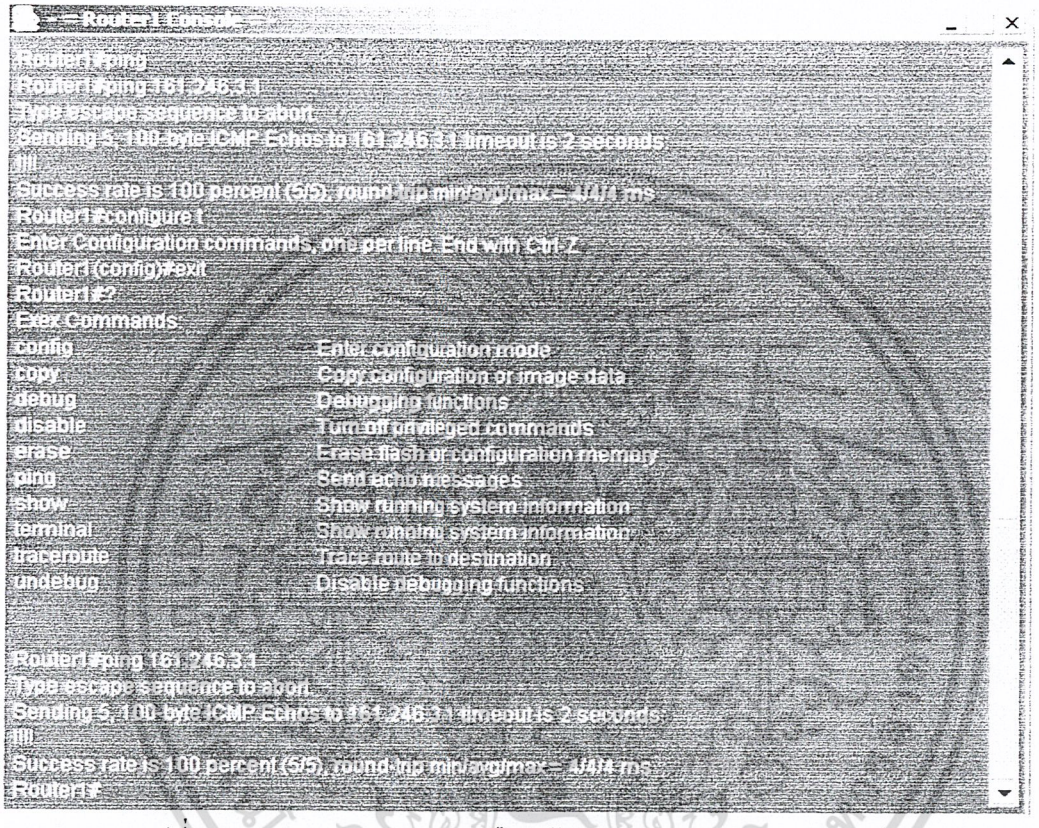


รูปที่ 10-8 แสดงรูปการคอนฟิกเพื่อให้เราเตอร์3 สามารถเลือกเส้นทางแบบสแตติกได้



รูปที่ 10-9 แสดงรูปการคอนฟิกเพื่อให้เราเตอร์4 สามารถเลือกเส้นทางแบบสแตติกได้

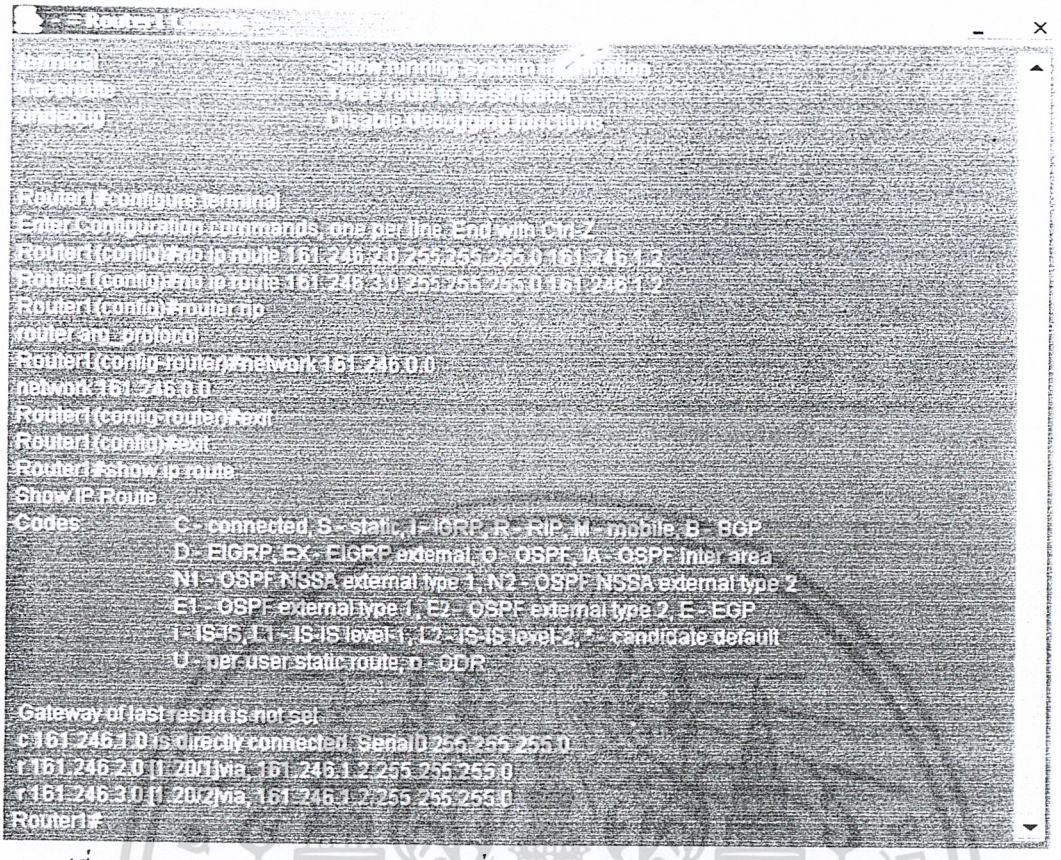
4. หลังจากกำหนดให้เราเตอร์แต่ละตัวสามารถเลือกเส้นทางไปยังเราเตอร์ต่างๆได้แล้ว ขั้นตอนต่อไปคือการตรวจสอบว่า เราเตอร์สามารถติดต่อกับเราเตอร์ในเครือข่ายที่ได้ทำการ เราท์ไว้แล้วได้หรือไม่ โดยใช้คำสั่ง ping ซึ่งในการทดสอบจะทำการ ping จาก เราเตอร์ 1 ไปยัง เราเตอร์ 4 หากสามารถเชื่อมต่อกันได้แสดงว่าการเลือกเส้นทาง ทำงานได้เป็นปกติ



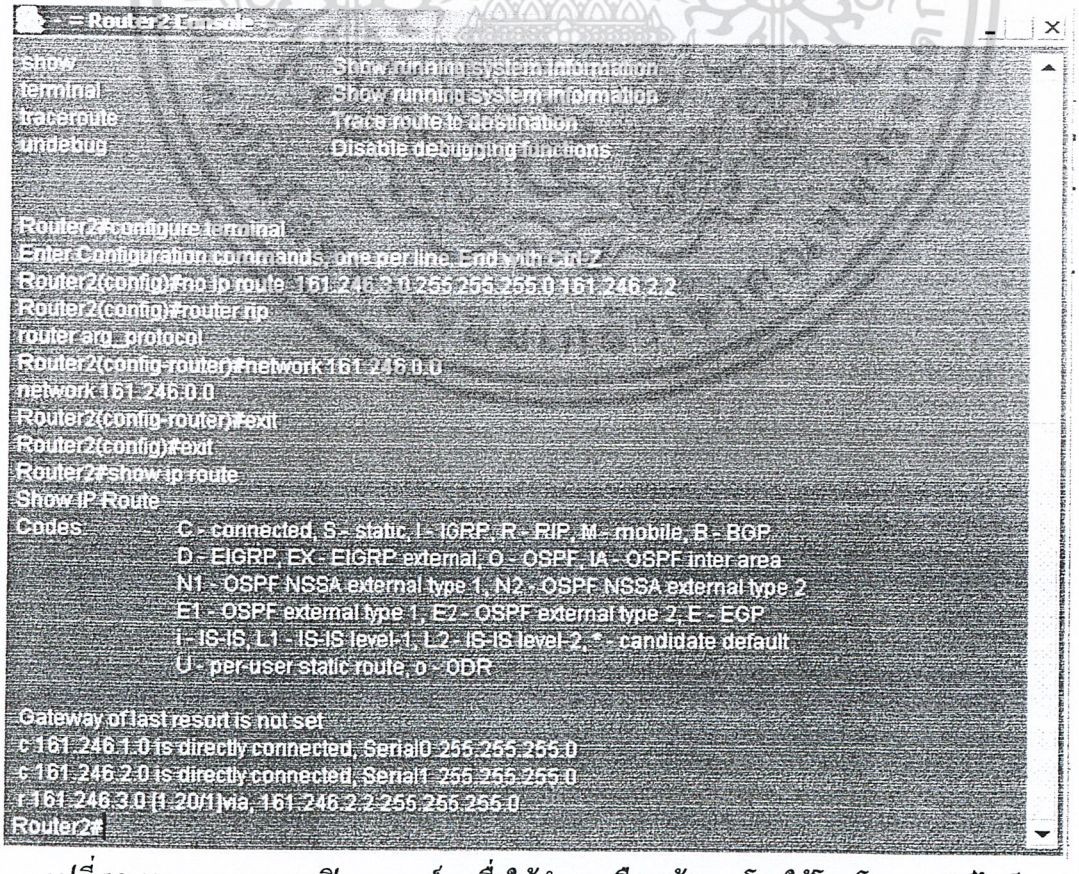
รูปที่ 10-10 แสดงการทำงาน การเลือกเส้นทางแบบสแตติกของเราเตอร์1

10.1.2 ตัวอย่างการเลือกเส้นทางโดยใช้โพรโตคอลอาร์ไอพี

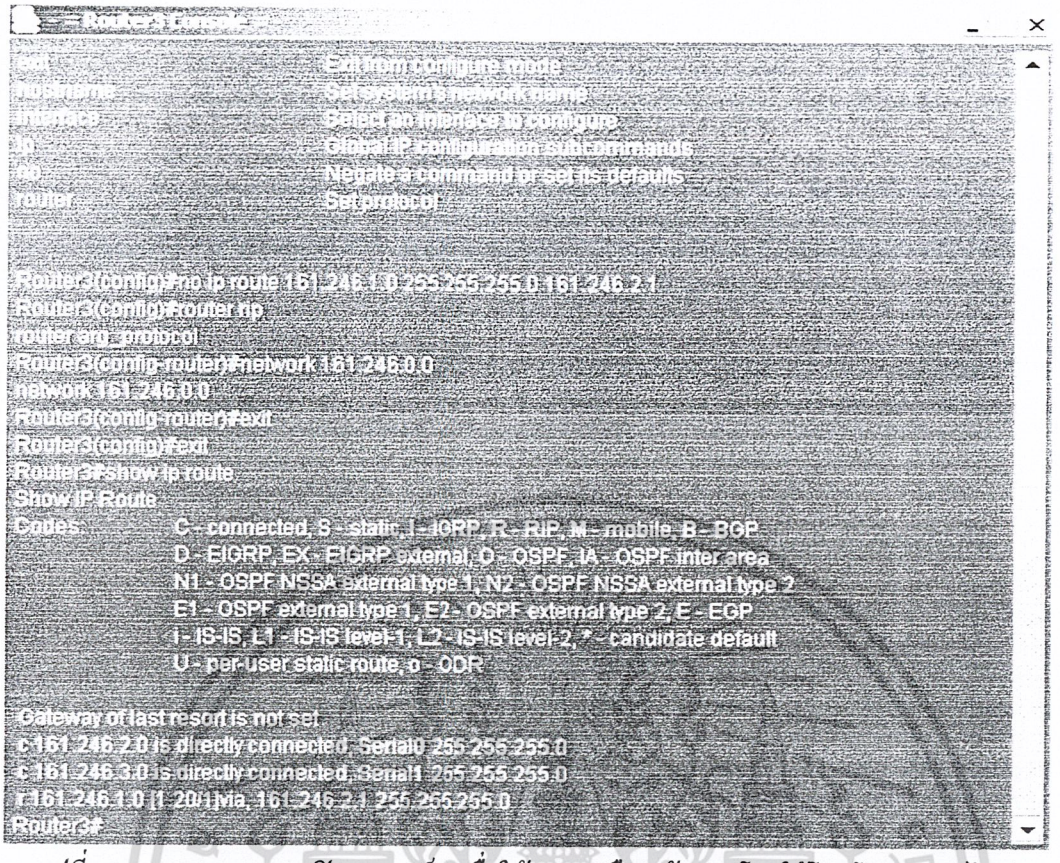
ก่อนที่จะทำการใช้โพรโตคอลเลือกเส้นทางแบบอาร์ไอพีนั้น จำเป็นต้องยกเลิกการเลือกเส้นทางแบบอื่นๆก่อน โดยใช้คำสั่ง no ip route หลังจากนั้นจึงใช้คำสั่ง router rip เพื่อเข้าไปกำหนดค่าในการใช้โพรโตคอลอาร์ไอพี ดังรูป



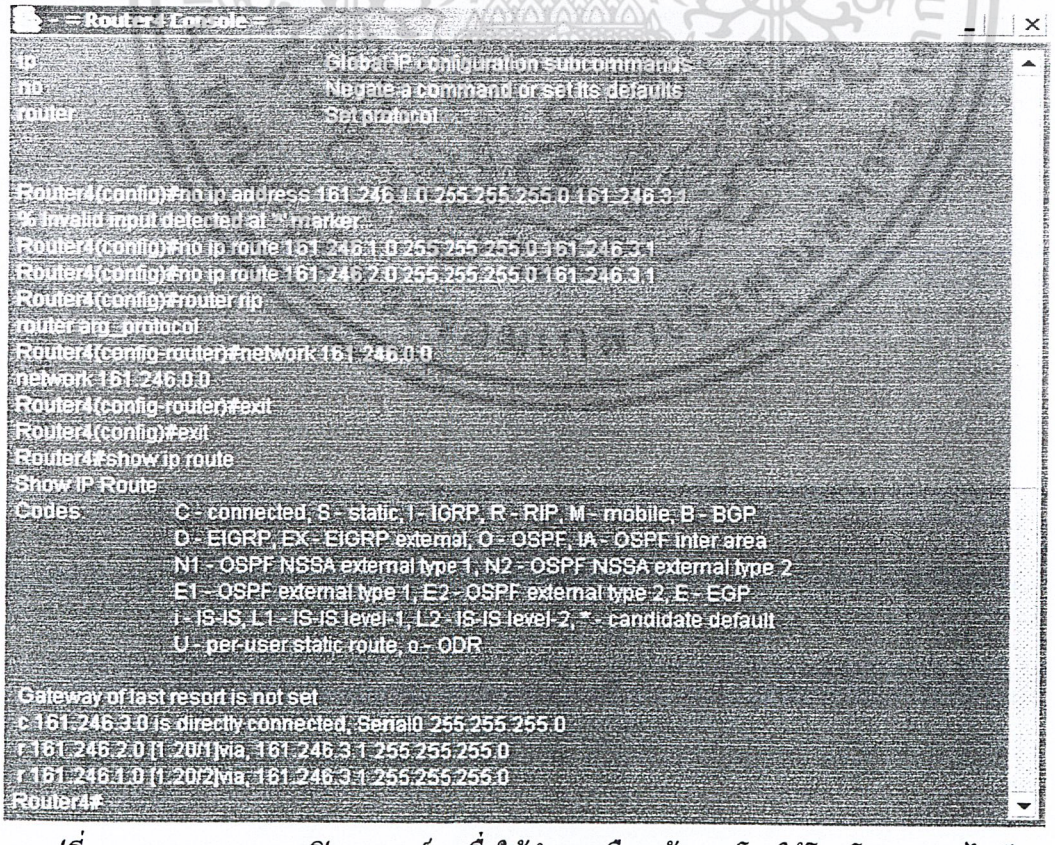
รูปที่ 10-11 แสดงการคอนฟิกเราเตอร์ 1 เพื่อให้ทำการเลือกเส้นทางโดยใช้โพรโตคอลอาร์ไอพี



รูปที่ 10-12 แสดงการคอนฟิกเราเตอร์ 2 เพื่อให้ทำการเลือกเส้นทางโดยใช้โพรโตคอลอาร์ไอพี

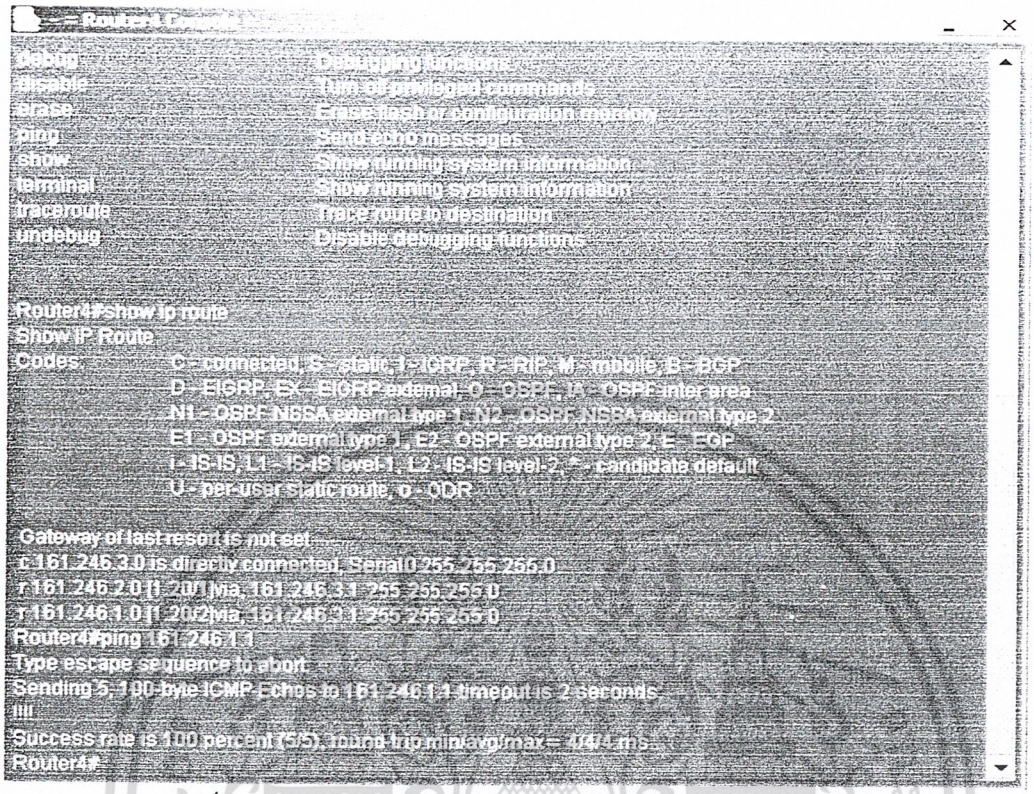


รูปที่ 10-13 แสดงการคอนฟิกเราเตอร์ 3 เพื่อให้ทำการเลือกเส้นทางโดยใช้โพรโตคอลอาร์ไอพี



รูปที่ 10-14 แสดงการคอนฟิกเราเตอร์ 4 เพื่อให้ทำการเลือกเส้นทางโดยใช้โพรโตคอลอาร์ไอพี

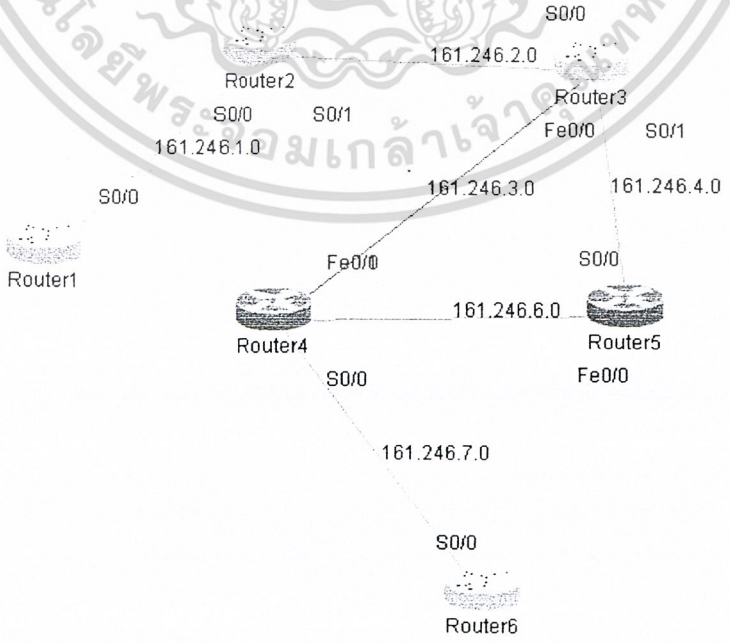
หลังจากที่ทำการคอนฟิกเรียบร้อยแล้ว ก็เป็นการทดสอบว่าสามารถทำงานได้อย่างถูกต้อง โดยจะทำการใช้คำสั่ง ping จากเราเตอร์4 ไปยังเราเตอร์1 ดังรูป



รูปที่ 10- 15 แสดงผลลัพธ์การ ping จากการใช้โพรโตคอลอาร์ไอพี

10.1.3 ตัวอย่างการทำงานของโพรโตคอลโอเอสพีเอฟ

1. สร้างเครือข่ายดังรูป



รูปที่ 10- 16 แสดงเครือข่ายเพื่อใช้ในการทดสอบโพรโตคอลโอเอสพีเอฟ

2. ทำการกำหนดค่าอินเตอร์เฟสตามรูป แล้วก็กำหนดโพรโตคอลให้กับเราเตอร์ทุกตัวเป็นแบบโอเอสพีเอฟด้วยคำสั่ง router ospf [id] โดย id ให้ใส่เป็นเลขอะไรก็ได้ซึ่งไม่มีผลต่อการทำงานของโปรแกรม เช่นที่เราเตอร์ R1 ก็ใส่เป็น router ospf 1 เป็นต้น หลังจากนั้นก็กำหนดค่าคอสท์ให้กับสายเชื่อมต่อระหว่างเราเตอร์ด้วยคำสั่ง ip ospf [cost] ซึ่งค่าคอสท์แสดงถึงระยะเวลาในการส่งข้อมูลผ่านเครือข่ายถ้าค่าคอสท์มากก็ต้องใช้เวลาในการส่งข้อมูลมาก ซึ่งโพรโตคอลโอเอสพีเอฟต้องสามารถหาเส้นทางที่คอสท์ต่ำที่สุดได้ โดยให้กำหนดค่าทุกเส้นทางเป็น 1 เพื่อสะดวกในการคำนวณ
3. ใช้คำสั่ง traceroute เพื่อดูว่าเส้นทางจากเราเตอร์ R1 ไปยังไอพี 161.246.7.2 ที่เป็นอินเตอร์เฟส S0/0 ของเราเตอร์6 ว่าสามารถไปทางไหนได้บ้าง โดยถาดูจากรูปที่ 11-16 แล้วจะเห็นได้ว่าจากเราเตอร์ R1 ไปยังเราเตอร์6 มีด้วยกันทั้งหมด 2 เส้นทางได้แก่

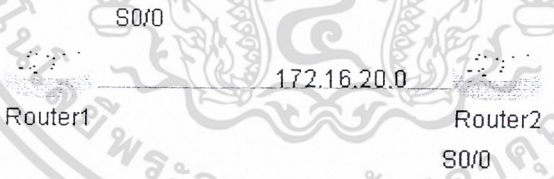
เส้นทางที่1 1.0, 2.0, 3.0, 7.0

เส้นทางที่2 1.0, 2.0, 4.0, 6.0, 7.0

โพรโตคอลโอเอสพีเอฟจะเลือกเส้นทางที่ดีที่สุดโดยใช้อัลกอริทึมของไดจ์สตรา(Dijkstra Algorithm) โดยคิดจากค่าคอสท์ที่ได้กำหนดให้ ซึ่งจะเห็นได้ว่าค่าคอสท์ของเส้นทางจาก R1 ไปยังเราเตอร์ 6 ในเส้นทางที่1 มีค่าคอสท์รวมเป็น 4 และในเส้นทางที่ 2 มีค่าคอสท์รวมเป็น 5 แสดงว่าในการเลือกเส้นทางของโพรโตคอลโอเอสพีเอฟจะเลือกเส้นทางที่1

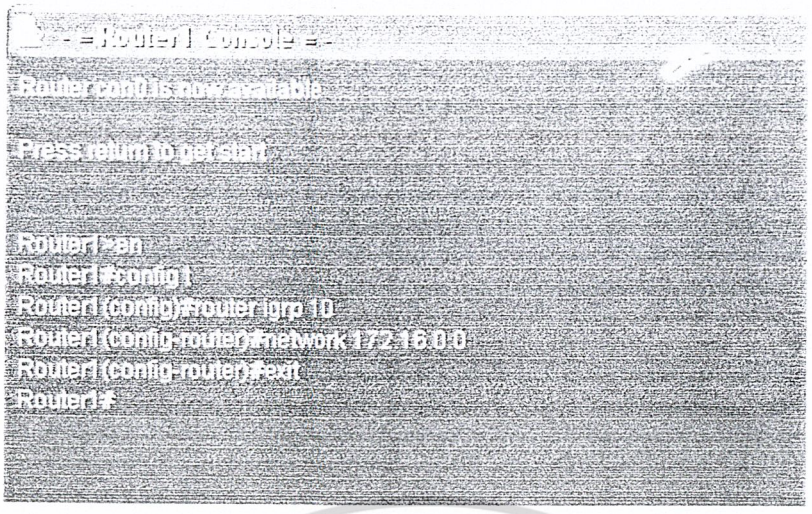
10.1.4 ตัวอย่างการทำงานของโพรโตคอลไอจีอาร์พี

1. สร้างเครือข่ายดังรูป

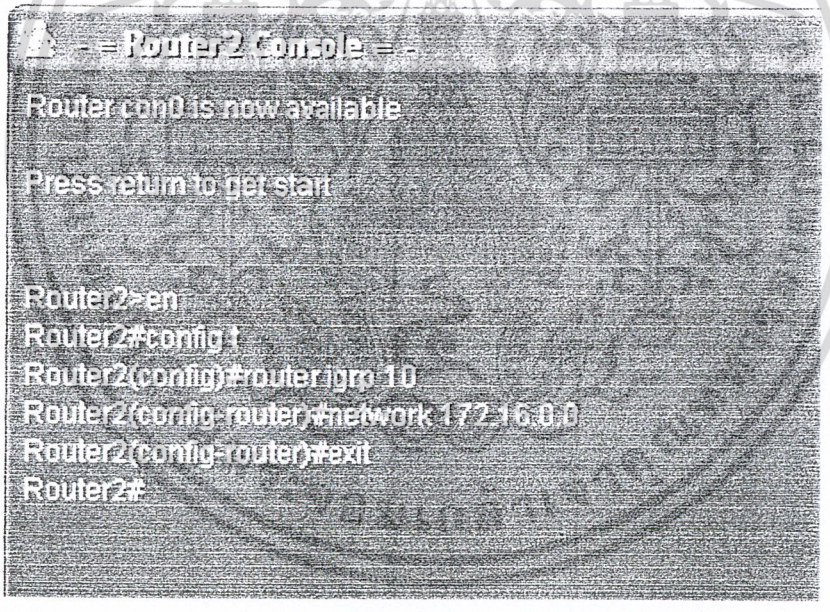


รูปที่ 10- 17 แสดงเครือข่ายเพื่อใช้ในการทดสอบโพรโตคอลไอจีอาร์พี

2. คอนฟิกเราเตอร์1 และเราเตอร์2 เพื่อเปิดให้ทำงานโดยใช้ ไอจีอาร์พี โพรโตคอล



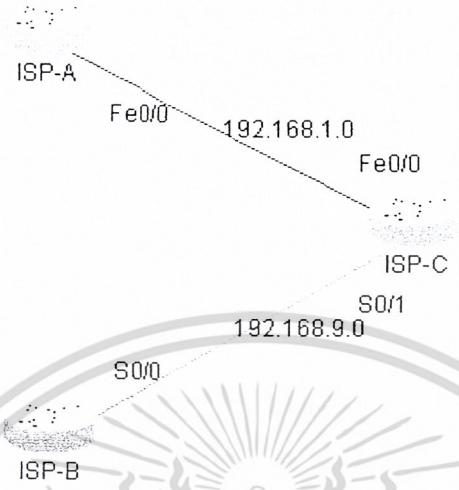
รูปที่ 10-18 แสดงการคอนฟิกเราเตอร์ 1 เพื่อให้ทำการเลือกเส้นทางโดยใช้โพรโตคอลไอจีอาร์พี



รูปที่ 10-19 แสดงการคอนฟิกเราเตอร์ 2 เพื่อให้ทำการเลือกเส้นทางโดยใช้โพรโตคอลไอจีอาร์พี

10.1.5 ตัวอย่างการทำงานของโพรโตคอลบีจีพี

1. สร้างเครือข่ายดังรูป



รูปที่ 10- 20 แสดงเครือข่ายเพื่อใช้ในการทดสอบโพรโตคอลบีจีพี

2. คอนฟิกเราเตอร์ ISP-B และ ISP-C เพื่อให้ทำงานแบบบีจีพี ซึ่งให้สามารถเลือกเส้นทางไป ยังออโตโนมัสหมายเลข 200 ได้

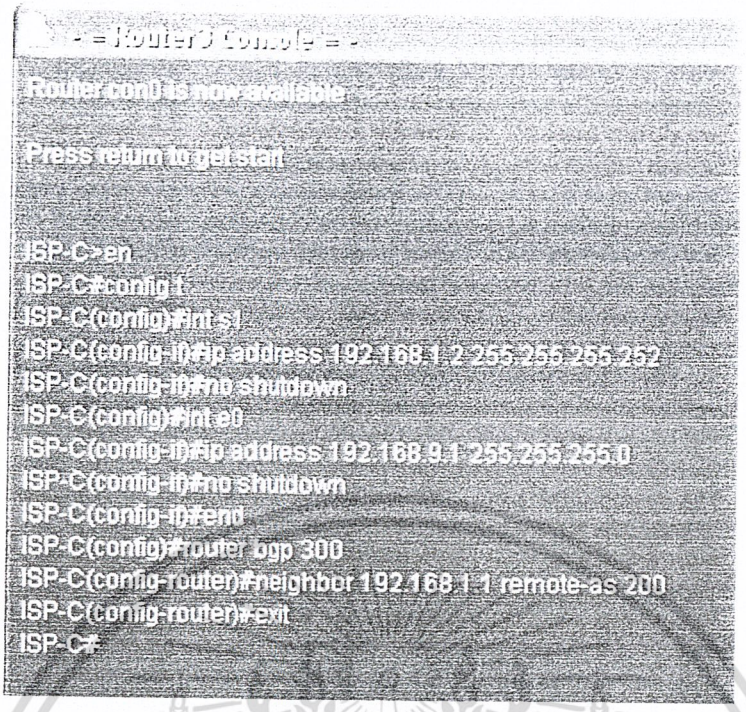
```
= = Router2 Console = =

Router con0 is now available

Press return to get start

ISP-B>en
ISP-B#config t
ISP-B(config)#int s0
ISP-B(config-if)#ip address 192.168.1.1 255.255.255.252
ISP-B(config-if)#no shutdown
ISP-B(config)#int e0
ISP-B(config-if)#ip address 192.168.4.1 255.255.255.0
ISP-B(config-if)#no shutdown
ISP-B(config-if)#end
ISP-B(config)#router bgp 100
ISP-B(config-router)#neighbor 192.168.1.1 remote-as 200
ISP-B(config-router)#exit
ISP-B#
```

รูปที่ 10-21 แสดงการคอนฟิกเราเตอร์ ISP-B เพื่อให้ทำการเลือกเส้นทางโดยใช้โพรโตคอลบีจีพี

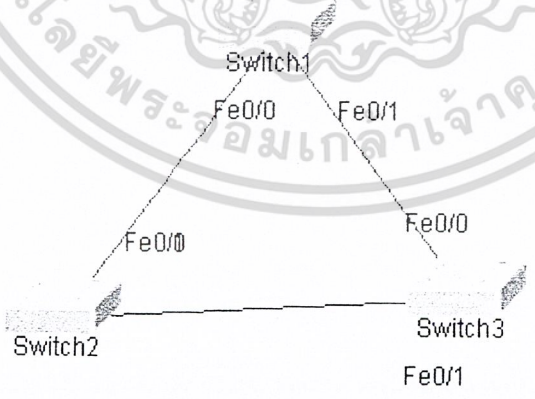


รูปที่ 10-22 แสดงการคอนฟิกเราเตอร์ ISP-C เพื่อให้ทำการเลือกเส้นทางโดยใช้โพรโตคอลบีจีพี

10.2 ตัวอย่างการทำงานโพรโตคอลของสวิตช์

10.2.1 คอมมอนสแปนนิ่งทรี

1. สร้างเครือข่ายดังรูป



รูปที่ 10-23 แสดงเครือข่ายสำหรับใช้ทดสอบโพรโตคอลคอมมอนสแปนนิ่งทรี

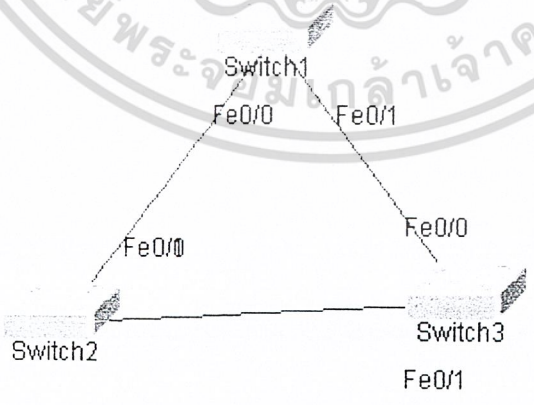
2. การทำงานของสแปนนิ่งทรีจะทำงานโดยอัตโนมัติ ทำการตรวจสอบโดยใช้คำสั่งคำสั่ง show spanning-tree brief

```
Switch3>en
Switch3#show spanning-tree brief
VLAN1
  Spanning tree enabled protocol IEEE
  ROOT ID    Priority 32768
             Address                 20d8.0b57.0233
             Hello time  2 sec   Max Age  20 sec   Forward Delay  15 sec
  Bridge ID  Priority 32768
             Address                 20d8.0b57.2eb8
             Hello time  2 sec   Max Age  20 sec   Forward Delay  15 sec
Interface
Name            Port ID         Cost            Port State
----            -------         ----            ----------
Fe0/0           0               19              block
Fe0/1           1               19              forward
Fe0/2           2               19              block
Fe0/3           3               19              block
Fe0/4           4               19              block
```

รูปที่ 10-24 แสดงคำสั่ง show spanning-tree brief เพื่อแสดงรายละเอียด

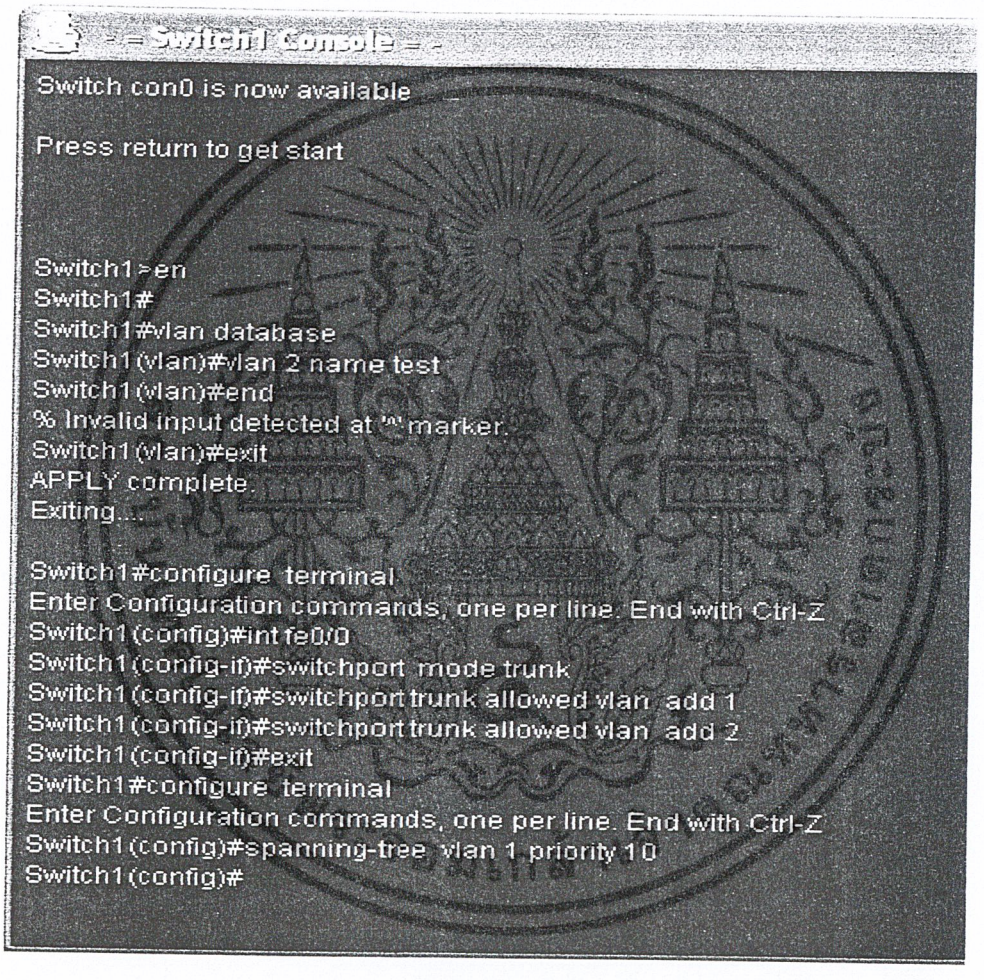
10.2.2 เปอร์วีแลนสแปนนิ่งทรี

1. สร้างเครือข่ายดังรูป เพื่อทำ เปอร์วีแลนสแปนนิ่งทรี โดยจะกำหนดให้ สวิตซ์1 เป็น รูท ของ วีแลน 1 โดยตั้งค่า ไพออริตี้ ของ วีแลน 1 ในสวิตซ์ 1 ให้มีค่าเป็น 10 และกำหนดให้ สวิตซ์ 2 เป็นรูทของ วีแลน 2 โดยตั้งค่าไพออริตี้ ของ วีแลน 2 ในสวิตซ์ 2 ให้มีค่าเป็น 10

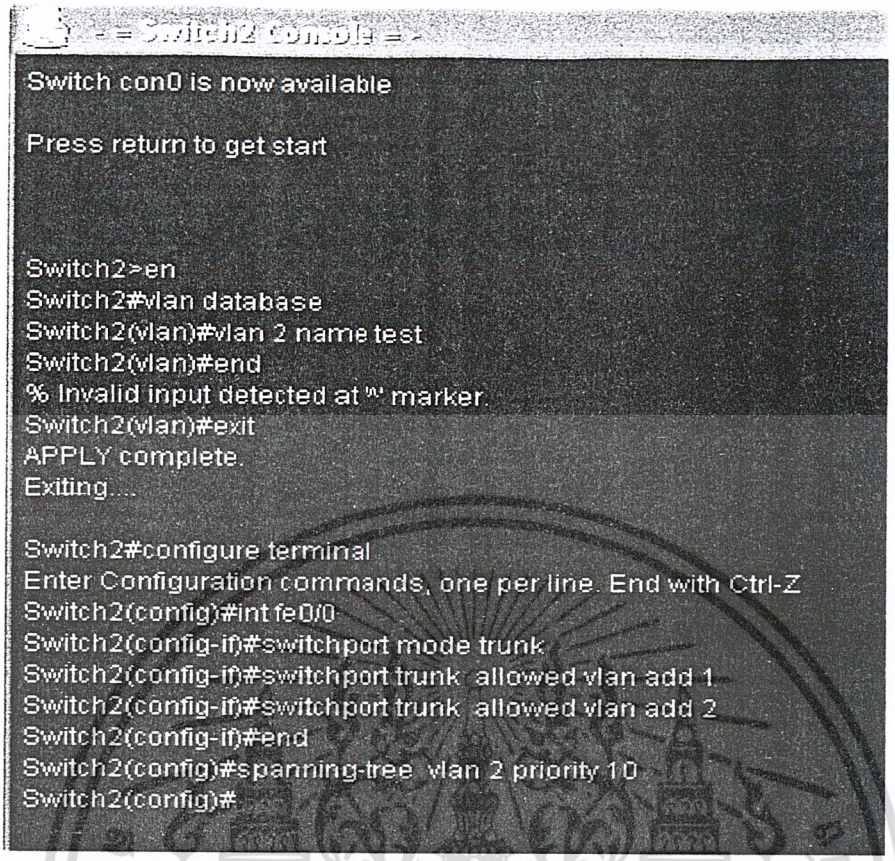


รูปที่ 10-25 แสดงเครือข่ายสำหรับใช้ทดสอบโพรโตคอลเปอร์วีแลนสแปนนิ่งทรี

2. สร้างวีแลนดาต้าเบสให้กับวีแลน 2 โดยใช้คำสั่ง vlan database เพื่อเข้าสู่ vlan mode จากนั้น สร้างวีแลน 2 ขึ้นมาโดยใช้คำสั่ง vlan < vlan num > name <vlan name> เมื่อสร้าง vlan แล้ว ขั้นตอนต่อไปกำหนด link ให้เป็น trunk link โดยใช้คำสั่ง switch port mode trunk และใส่วีแลนเข้าไปใน trunk โดยใช้คำสั่ง switchport trunk allowed vlan add <vlan id> โดยขั้นตอนดังกล่าวตั้งค่าในสวิตช์ทั้ง 3 ตัว ขั้นตอนต่อไป ทำการตั้งค่า ไพออริตี้ ให้กับ switch 1 และ switch 2 โดยใช้คำสั่ง spanning-tree vlan <vlan num> priority < priority>

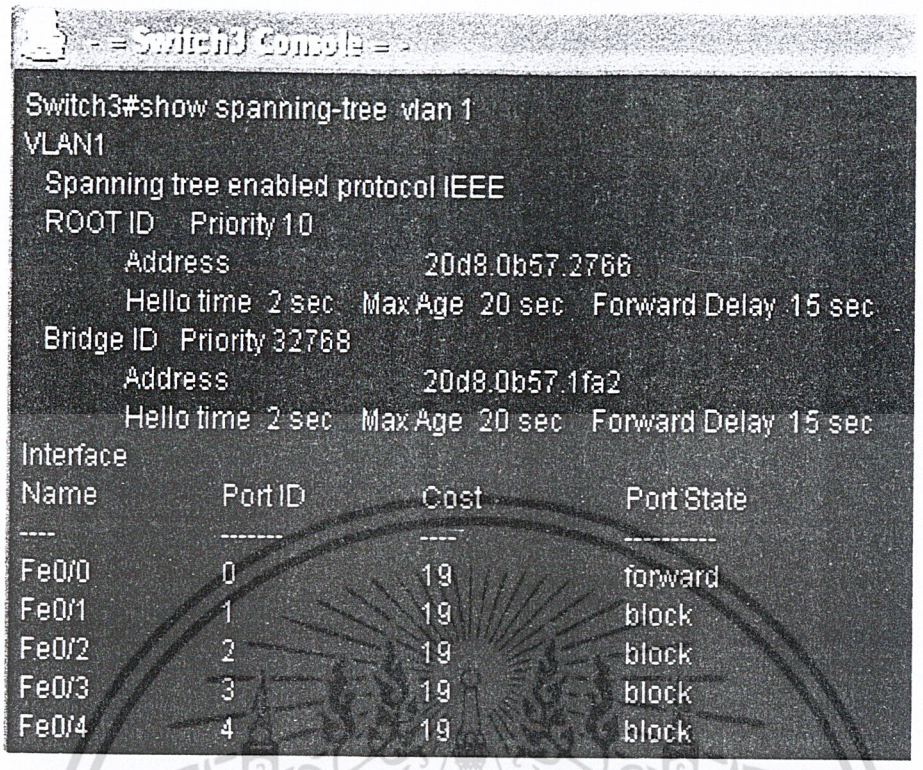


รูปที่ 10-26 แสดงการตั้งค่าวีแลนด์ อินเตอร์เฟส และไพออริตี้ เพื่อทำสแปนนิ่งทรีให้กับสวิตช์ 1

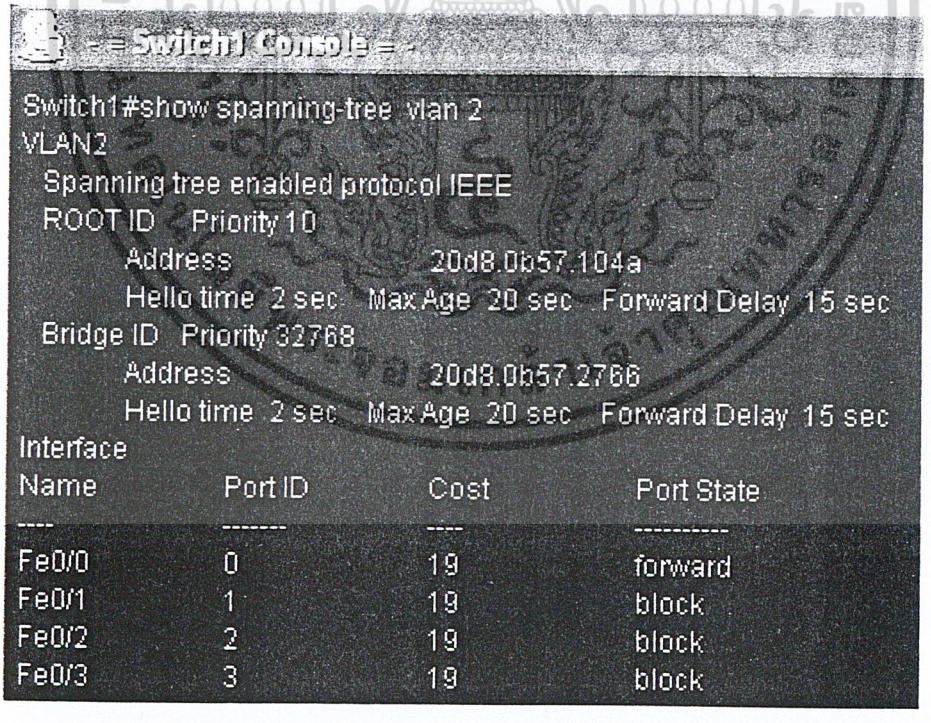


รูปที่ 10-27 แสดงการตั้งค่าวีแลนด์ อินเตอร์เฟส และไพออริตี้ เพื่อทำสแปนนิ่งทรีให้กับสวิตซ์ 2

3. ตรวจสอบการทำงานโดยใช้คำสั่ง show spanning-tree vlan เพื่อดูความถูกต้องโดยถ้าหากตั้งค่าได้ถูกต้องสายที่ถูกบล็อก ของทั้งสอง วีแลน จะเป็นคนละตัวกันดังตัวอย่าง วีแลน 1 จะบล็อกที่ สวิตซ์ 3 Interface Fe0/1 และวีแลน 2 จะบล็อกที่ สวิตซ์ 1 Interface Fe0/1 ซึ่งเป็นคนละตัวกันทำให้ใช้งานสายได้มีประสิทธิภาพมากขึ้น



รูปที่ 10-28 แสดงสแปนนิ่งทรีของวีแลน 1



รูปที่ 10-29 แสดงสแปนนิ่งทรีของวีแลน 2

บทที่ 11
สรุปและวิจารณ์

ในการทำงานของปริญญานิพนธ์นี้ สามารถที่จะสร้างซอฟต์แวร์จำลองการตั้งค่าเราเตอร์ของซิสโก้ และจำลองการทำงานของโพรโตคอลที่ใช้เลือกเส้นทางได้ ทั้งแบบแสตติกและไดนามิก (อาร์ไอพี, โอเอสพีเอฟ,บีจีพี,ไอจีอาร์พี) ทำให้สามารถจำลองการทำงานของเครือข่ายได้ตั้งแต่เครือข่ายขาดเล็กไปจนถึงเครือข่ายขนาดใหญ่ สามารถที่จะบันทึกค่าคอนฟิกและรูปไดอะแกรมของเครือข่ายที่สร้างไว้ลงไฟล์เก็บไว้ได้ ผู้ใช้สามารถออกแบบสร้างเครือข่ายเองได้เพื่อให้สามารถมองภาพรวมของเครือข่ายได้อย่างชัดเจน พร้อมทั้งมีบทเรียนสอนเพื่อที่ผู้สนใจใช้ซอฟต์แวร์นี้สามารถทดลองใช้ตามบทเรียน

การตั้งค่าอุปกรณ์เราเตอร์และสวิตช์มีคำสั่งมากและมีการออกซีรี่ใหม่มาเรื่อยๆ การที่ผู้ดูแลระบบจะดูแลและใช้ความสามารถของเราเตอร์และสวิตช์ได้อย่างเต็มที่นั้น ผู้ดูแลจะต้องทราบถึงความสามารถในการตั้งค่าของเราเตอร์และสวิตช์ ต้องทราบว่ามีคำสั่งอะไรบ้าง แต่ถึงอย่างไรก็ตามหลักการพื้นฐานในการตั้งค่าอุปกรณ์เราเตอร์และสวิตช์รวมทั้งคำสั่งพื้นฐานก็ไม่แตกต่างกันมากนัก การที่เราเข้าใจหลักการพื้นฐานของการตั้งค่าอุปกรณ์เราเตอร์และสวิตช์จึงเป็นสิ่งที่จำเป็น

11.1 ปัญหาและอุปสรรค

- คำสั่งในการตั้งค่าเราเตอร์และสวิตช์มีจำนวนมาก อีกทั้งคำสั่งในการตั้งค่านั้นในแต่ไอโอเอส (IOS)ก็มีความแต่ต่างกัน จำเป็นต้องใช้เวลาในการศึกษาและรวบรวมเฉพาะคำสั่งที่น่าสนใจ และเป็นประโยชน์โดยการเจาะจงลงไปว่าจะเป็น ไอโอเอส รุ่นใดรุ่นหนึ่ง
- โปรแกรมจากปีก่อนขยายแต่เพียงส่วนของ Hardware โดยไม่ได้ทำ Function ของทั้งสวิตช์และเราเตอร์ ให้รองรับการทำงานเลย

11.2 ขอบเขตและข้อจำกัดของโครงงาน

- จำลองการทำงานรูปแบบการตั้งค่าของเราเตอร์ซิสโก้ รุ่น 1760
- จำลองการทำงานรูปแบบการตั้งค่าสวิตช์ของซิสโก้ รุ่น 2950 สามารถจำลองการทำ Per-VLAN ได้ แต่ยังไม่สามารถจำลองการทำ อินเตอร์วีแลนได้
- ใช้โพรโตคอลในการค้นหาเส้นทางแบบสแตติกและแบบไดนามิก ประกอบด้วยอาร์ไอพี โอเอสพีเอฟ บีจีพี และ ไอจีอาร์พี เท่านั้น
- ไม่สามารถใช้ได้ทุกคำสั่งที่มีในเราเตอร์และสวิตช์

11.3 แนวทางการประยุกต์และพัฒนา

- เพิ่มโพรโตคอล เช่น IGRP,BGP เพื่อจำลองเป็นเราเตอร์ระดับสูง โปรแกรมในขณะนี้ยังเป็น LAN Router ต้องพัฒนาต่อเป็น WAN Router
- พัฒนาให้โปรแกรมสามารถจำลองการทำงานของ อินเตอร์วีแลน เพื่อให้ผู้ใช้สามารถฝึกหัดการตั้งค่าอินเตอร์วีแลนให้กับสวิตช์ได้

11.4 สรุปโครงงาน

โครงงานนี้มีประโยชน์มาก เพราะสามารถให้ผู้ที่ต้องการฝึกการตั้งค่าเราเตอร์และสวิตช์ อีกทั้งยังสามารถเรียนรู้การทำงานของโพรโตคอลที่เราเตอร์และสวิตช์ใช้งาน ไม่ว่าจะเป็นผู้ที่อยู่ในระดับการศึกษาหรือใช้งานจริงก็ตาม สามารถฝึกฝนด้วยตนเองได้ โดยมีตัวอย่างการทดลองเบื้องต้นให้ฝึกฝน ซึ่งโครงงานนี้นอกจากสามารถทำเพื่อการศึกษาด้านเครือข่ายคอมพิวเตอร์แล้ว ยังเป็นการลดความต้องการการสั่งซื้ออุปกรณ์เราเตอร์และสวิตช์ หรือซอฟต์แวร์เชิงพาณิชย์ที่จำลองการตั้งค่าเราเตอร์และสวิตช์จากต่างประเทศ โครงงานนี้เป็นโปรแกรมที่ทำงานได้ทั้งแบบผู้ใช้คนเดียวและผู้ใช้หลายคนอีกทั้งเป็นโปรแกรมแบบมัลติแพลตฟอร์ม (Multi platform) เท่าที่สำรวจมายังไม่มีซอฟต์แวร์เชิงพาณิชย์ตัวไหนที่สามารถทำได้ และไม่มีซอฟต์แวร์เชิงพาณิชย์ตัวไหนที่สามารถใช้โพรโตคอลการเลือกเส้นทางแบบโอเอสพีเอฟ บีจีพี และ ไอจีอาร์พี ได้

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

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#

Debug ip rip

Command Mode Exex Commands

เป็นคำสั่งที่ใช้สำหรับแสดงการรับและส่งข้อมูลตารางเส้นทางของเราเตอร์

รูปแบบคำสั่ง Debug ip rip

ตัวอย่างการใช้งาน

RouterA#debug ip rip

RIP protocol debugging is on

RouterA#

Description

Command Mode Exec 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>

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#

Hostname

Command Mode Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดชื่อเราเตอร์

ตัวอย่างการใช้งาน

Router(config)#hostname RouterA

RouterA(config)#

Command Mode Global Configuration Commands

เป็นคำสั่งเพื่อเข้าสู่การตั้งค่าอินเทอร์เฟสต่างๆ ของเราเตอร์โดยคำสั่งนี้จะตามด้วยชื่อของอินเทอร์เฟสที่ต้องการตั้งค่า

รูปแบบคำสั่ง Int <interface_name>

โดย <Interface_name> เป็นชื่อของอินเทอร์เฟส

ตัวอย่างการใช้งาน

Router(config)#int e0

Router(config-if)#

Command Mode Interface Commands

เป็นคำสั่งที่ใช้กำหนดค่าไอพีแอดเดรสให้กับอินเทอร์เฟสที่กำลัง config อยู่

รูปแบบคำสั่ง ip address <ip_address> <netmask>

<Ip_address> เป็นค่าไอพีแอดเดรสที่ต้องการกำหนดให้กับอินเทอร์เฟส

<Netmask> เป็นค่าเน็ตเวิร์คมาสก์ของไอพีแอดเดรส

ตัวอย่างการใช้งาน

Router(config)#int e0

Router(config-if)#ip address 161.246.5.1 255.255.255.0

Router(config-if)#

Command Mode Privileged EXEC Command

เป็นคำสั่งที่ใช้สำหรับกำหนดค่า weight ให้กับเราเตอร์ตัวที่เชื่อมต่อกันอยู่

รูปแบบคำสั่ง Ip Ospf Cost <Router_Name> <Weight>

<Router_Name> เป็นชื่อเราเตอร์ตัวติดกันที่ต้องการใส่ค่า weight

<Weight> เป็นค่า weight ระหว่างเราเตอร์

ตัวอย่างการใช้งาน

RouterA#ip ospf cost Router2 10

RouterA#

Command Mode Global Configuration Commands

เป็นคำสั่งเพื่อกำหนดการค้นหาเส้นทางเป็นแบบสแตติกหรือยกเลิกข้อมูลการค้นหาเส้นทางแบบสแตติก

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

<Destination> เป็นเลขเครือข่ายของเครือข่ายปลายทาง

<Netmask> เป็นค่าเน็ตเวิร์คมาสก์ของไอพีแอดเดรส

<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)#

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)#

Command Mode Router Configuration Commands

เป็นคำสั่งที่ใช้สำหรับกำหนด network address ให้กับเครือข่ายที่ทำการ์กอนฟิกหรือยกเลิก 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

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

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)#

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

ตัวอย่างการใช้งาน

RouterA#show int e0

Ethernet0 is up, line protocol is up

Hardware is Lance

Internet address is 161.246.1.1/24

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

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

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

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

Last clearing of "show interface" counters never

Queueing strategy: fifo

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

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

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

0 packets input, 0 bytes, 0 no buffer

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

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

0 input packets with dribble condition detected

472 packets output, 47379 bytes, 0 underruns

0 output errors, 0 collisions, 2 interface resets

0 babbles, 0 late collision, 0 deferred

0 lost carrier, 0 no carrier

0 output buffer failures, 0 output buffers swapped out

RouterA#

Show Ip 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#

Show Ip Ospf interface

Command Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดง databaseของตารางเส้นทางโอเอสพีเอฟ ของเราเตอร์

รูปแบบคำสั่งShow 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#

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#

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงโพรโตคอลที่เราเตอร์ใช้งานอยู่ขณะนั้น

ตัวอย่างการใช้งาน

RouterA#show ip protocol

Routing Protocol is "rip"

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

Outgoing update filter list for all interfaces is not set

Incoming update filter list for all interfaces is not set

Redistributing : rip

Routing for Networks :

161.246.0.0

Routing Information Sources :

Gateway Distance Last Update

161.246.0.0 120 0:00:00

Distance: (default is 120)

RouterA#

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#

show ip Route

Command Mode Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงตารางการค้นหาเส้นทาง

รูปแบบคำสั่ง Show Ip Route <route_type>

<route_type> เป็นตัวเลือกที่จะใส่หรือไม่ใส่ก็ได้ ถ้าใส่จะเป็นการระบุว่าต้องการดูเฉพาะตารางค้นหาเส้นทางประเภทใด โดย

s : สแตติก

r : อาร์ไอพี

o : โอเอสพีเอฟ

ตัวอย่างการใช้งาน

RouterA#show ip route

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

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

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

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

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

U - per-user static route, o - ODR

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#

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#
```

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 timestamps 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#

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 tcj

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)

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 เพื่อเปลี่ยนสถานะของ

อินเทอร์เฟสเป็นปิด

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#

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>

ภาคผนวก ข

สรุปคำสั่งทั้งหมดของ สวิตช์

Configure terminal
Command Mode : Privileged EXEC Commands
เป็นคำสั่งที่ใช้เปลี่ยนโหมดจาก privilege mode เป็น global configuration mode **ตัวอย่างการใช้งาน** Switch1#configure terminal Switch1(config)#

Disable
Command Mode : Privileged EXEC Commands
เป็นคำสั่งที่ใช้สำหรับเปลี่ยนโหมดจาก privilege โหมดเป็น user execution โหมด **ตัวอย่างการใช้งาน** Switch11#disable Switch1>

Enable
Command Mode : User EXEC Commands
เป็นคำสั่งที่ใช้สำหรับเปลี่ยนโหมดจาก user execution เป็น privilege โหมด **ตัวอย่างการใช้งาน** Switch1>enable Switch1#

End
Command Mode : Global Configuration Commands, Interface Commands
เป็นคำสั่งที่ใช้สำหรับสิ้นสุดการ ใช้คำสั่งย่อย **ตัวอย่างการใช้งาน** Switch1(config-if)#end Switch1(config)#

Exit
Command Mode : User EXEC Commands, Global Configuration Commands, Interface Commands,

VLAN Commands

เป็นคำสั่งใช้สำหรับออกจากโหมดที่กำลังใช้งานอยู่ขณะนั้น เพื่อเข้าสู่ Privileged EXEC Commands โหมด และหากใช้ใน user mode จะเป็นการ log out

ตัวอย่างการใช้งาน

Switch1(config-if)#exit

Switch1#

Hostname <name>

Command Mode : Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดชื่อสวิตช์

ตัวอย่างการใช้งาน

Switch1(config)#hostname SwitchA

SwttchA(config)#

Mac-address-table aging-time <time>

Command Mode : Global Configuration Commands

เป็นคำสั่งที่ใช้กำหนดค่าเวลาคงอยู่ของ mac-address มีค่าตั้งแต่ 10-1000000

ตัวอย่างการใช้งาน

Switch1(config)#mac-address-table aging-time 1000

Switch1(config)#

Shutdown / No shutdown

Command Mode : Interface Commands

เป็นคำสั่งที่ใช้กำหนดสถานะของอินเทอร์เฟสที่ต้องการใช้งาน

ตัวอย่างการใช้งาน

Switch1(config-if)#shutdown

Switch1(config-if)#

Switch1(config-if)#no shutdown

Switch1(config-if)#

Show interface <name>

Command Mode : Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงรายละเอียดของอินเทอร์เฟส <name> เป็นตัวเลือกจะใส่หรือไม่ใส่ก็ได้ ถ้าใส่จะเป็นการระบุให้แสดงรายละเอียดเฉพาะอินเทอร์เฟสที่ระบุ ถ้าไม่เช่นนั้นจะแสดงทุกอินเทอร์เฟส

ตัวอย่างการใช้งาน

```
Switch1#show interface fe0/0
FastEthernet0/0 is up ,line protocol is up
Hardware is Fast Ethernet , address is 0001.2484.dfd1 (bia 0001.2484.dfd1)
MTU 1500 bytes, BW 100000 Kbit, DLY 1000 usec,
   reliability 255/255, txload 1/255, rxload 1/255
Encapsulation ARPA, loopback not set
Keepalive set (10 sec)
Auto-duplex, Auto-speed
input flow-control is off, output flow-control is off
Last input never, output 4d21h, output hang never
Last clearing of "show interface" counters never
Input queue:0/75/0/0 (size/max/drops/flushes); Total output drops:0
Queueing strategy:fifo
Output queue :0/40 (size/max)
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
   1 packets input, 64 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 watchdog, 0 multicast, 0 pause input
   0 input packets with dribble condition detected
   1 packets output, 64 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 PAUSE output
   0 output buffer failures, 0 output buffers swapped out
Switch1#
```

Show interface <name> switchport
Command Mode : Privileged EXEC Commands
เป็นคำสั่งดูรายละเอียดของ swithport ใน interface ที่ต้องการจะดู

ตัวอย่างการใช้งาน

Switch1#show interface fe0/0 switchport

Name: fe0/0

Switchport: Enabled

Administrative Mode: static access

Operational Mode: static access

Administrative Trunking Encapsulation: dot1q

Operational Trunking Encapsulation: native

Negotiation of Trunking: Disables

Access Mode VLAN: 1 (default)

Trunking Native Mode VLAN: 1 (default)

Trunking VLANs Enabled: NONE

Trunking VLANs Enabled: NONE

Priority for untagged frames: 0

Override vlan tag priority: FALSE

Voice vlan: none

Appliance trust: none

Switch1#

Show interface <name> switchport allowed-vlan

Command Mode : Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับดูว่า ใน interface นั้นใช้ vlan อะไรอยู่

ตัวอย่างการใช้งาน

Switch1#show interface fe0/0 switchport allowed-vlan

"NONE"

Switch1#

Show mac-address-table

Command Mode : User EXEC Commands

เป็นคำสั่งสำหรับดู mac-address-table ของสวิตช์

ตัวอย่างการใช้งาน

Switch1>show mac-address-table

```
-- Mac address table--

Switch1>
```

Show spanning-tree

Command Mode : Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับ ดู spanning-tree

ตัวอย่างการใช้งาน

```
Switch1#show spanning-tree
Spanning-tree disabled
Switch1#
```

Show spanning-tree brief

Command Mode : Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดง spaning-tree แบบสรุป

ตัวอย่างการใช้งาน

```
Switch1#show spanning-tree brief
VLAN1
  Spanning tree enabled protocol IEEE
  Root ID    Priority 32768
             Address 0001.67ac.fc00
             Hello time 2 sec   Mac Age  20 sec   Forward Delay  15 sec
  Bridge ID  Priority 32768
             Address 0001.67ac.fc00
             Hello time 2 sec   Mac Age  20 sec   Forward Delay  15 sec
Port
Name      Port ID   Prior  Cost  Port State  Bridege ID        Port ID
-------   ---------  -----  ------  ----------  --------------  ----------
Fe0/1     128.01    128    104   blocking    0001.67ac.fc00    128.01
Fe0/2     128.11    128    104   blocking    0001.67ac.fc00    128.11
Fe0/3     128.21    128    104   blocking    0001.67ac.fc00    128.21
Fe0/4     128.31    128    104   blocking    0001.67ac.fc00    128.31
Fe0/5     128.41    128    104   blocking    0001.67ac.fc00    128.41
Fe0/6     128.51    128    104   blocking    0001.67ac.fc00    128.51
```

Fe0 7	128.61	128	104	blocking	0001.67ac.fc00	128.61
Fe0 8	128.71	128	104	blocking	0001.67ac.fc00	128.71
Fe0 9	128.81	128	104	blocking	0001.67ac.fc00	128.81
Fe0 10	128.91	128	104	blocking	0001.67ac.fc00	128.91
Fe0 11	128.101	128	104	blocking	0001.67ac.fc00	128.101
Fe0 12	128.111	128	104	blocking	0001.67ac.fc00	128.111

Switch1#

Show spanning-tree interface <interface name>

Command Mode : Privileged EXEC Commands

เป็นการแสดง spanning-tree ของ interface ที่กำหนด

ตัวอย่างการใช้งาน

Switch1#show spanning-tree interface fe0 0

VLAN1

Spanning tree enabled protocol IEEE

Root ID Priority 32768

Address 0001.67ac.fc00

Hello time 2 sec Mac Age 20 sec Forward Delay 15 sec

Bridge ID Priority 32768

Address 0001.67ac.fc00

Hello time 2 sec Mac Age 20 sec Forward Delay 15 sec

Port

Name	Port ID	Prior	Cost	Port State	Bridege ID	Port ID
Fe0 1	128.01	128	104	blocking	0001.67ac.fc00	128.01

Switch1#

Show spanning-tree vlan <vlan id>

Command Mode : Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดง spanning-tree ของ vlan ที่ระบุ

ตัวอย่างการใช้งาน

Switch1#show spanning-tree vlan 1

No parameter have been configured :

Switch1#

Show spanning-tree summary

Command Mode : Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับ แสดงการใช้ spanning-tree แบบสรุป

ตัวอย่างการใช้งาน

Switch1#show spaning-tree summary

UplinkFast is Disabled

Name	Blocking	Listening	Learning	Forwarding	STP Active

Switch1#

Show version

Command Mode : User EXEC Commands, Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดง version ของ IOS ที่กำลังใช้งานอยู่

ตัวอย่างการใช้งาน

Switch1>show version

Cisco Internetwork Operating System Software

IOS (tm) C2950 Software (C2950-I6Q4L2-M), Version 12.1(9)EAI

Copyright (c) 1986-2002 by cisco Systems, Inc.

Compiled Wed 27-Feb-02 06:51 by antonio

Image text-base:0x80010000, data-base:0x804E2000

ROM:Bootstrap program is C2950 boot loader

sw.getName()+ " uptime is 1 hour, 54 minutes

System returned to ROM by power-on

System image file is "flash:c2950-16q412-mz.121-0.0.9.EAI.bin

Cisco WS-C2950G-12-EI (RC32300) processor with 20830K bytes of memory.

Last reset from system-reset

Running Enhanced Image

12 FastEthernet/IEEE 802.3 interface(s)

2 Gigabit Ethernet/IEEE 802.3 interface(s)

32K bytes of flash-simulated non-volatile configuration memory.

```
Base ethernet MAC Address:00:05:74:28:09:C0
Configuration register is 0x1
Switch1>
```

Show vlan

Command Mode : User EXEC Commands, Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดง vlan ทั้งหมด

ตัวอย่างการใช้งาน

```
Switch1>show vlan
VLAN  Name                                Status    Ports
----  ------------------------------------ --------- -------------------------------
1     default                             active    Fe0 1, Fe0 2, Fe0 3, Fe0 4,
                                                    Fe0 5, Fe0 6, Fe0 7, Fe0 8,
                                                    Fe0 9, Fe0 10, Fe0 11, Fe0 12,
1002  fddi-default                        active
1003  token-ring-default                  active
1004  fddinet-default                     active
1005  trnet-default                       active

VLAN  Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1 Trans2
----  ----- ---------- ----- ------ ------ -------- ---- -------- ------ ------
1     enet  100001     1500  -      -      -        -    -        0      0
1002  fddi  101002     1500  -      -      -        -    -        0      0
1003  tr    101003     1500  -      -      -        -    -        0      0
1004  fdnet 101004     1500  -      -      -        ieee -        0      0
1005  trnet 101005     1500  -      -      -        ibm  -        0      0
Switch1>
```

Show vlan brief

Command Mode : User EXEC Commands, Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดง vlan ทั้งหมดแบบย่อ

ตัวอย่างการใช้งาน

```
Switch1>show vlan
```

```
VLAN  Name                                Status    Ports
----  ------------------------------------ --------- -------------------------------
1     default                             active    Fe0/1, Fe0/2, Fe0/3, Fe0/4,
                                                    Fe0/5, Fe0/6, Fe0/7, Fe0/8,
                                                    Fe0/9, Fe0/10, Fe0/11, Fe0/12,
1002  fddi-default                        active
1003  token-ring-default                  active
1004  fddinet-default                     active
1005  trnet-default                       active
Switch1>
```

Spanning-tree / No spanning-tree

Command Mode : Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับเปิด/ปิด การใช้ spanning-tree

ตัวอย่างการใช้งาน

```
Switch1(config)#spanning-tree
Switch1(config)#
Switch1(config)#no spanning-tree
Switch1(config)#
```

Switch mode access / No switch mode

Command Mode : Interface Commands

เป็นคำสั่งที่ใช้สำหรับ เปิดให้ interface นั้นใช้การเชื่อมต่อแบบ access

ตัวอย่างการใช้งาน

```
Switch1(config-if)#switch mode access
Switch1(config-if)#
Switch1(config-if)#no switch mode access
Switch1(config-if)#
```

Switch mode trunk / No switch mode

Command Mode : Interface Commands

เป็นคำสั่งที่ใช้สำหรับ เปิดให้ interface นั้นใช้การเชื่อมต่อแบบ trunk

ตัวอย่างการใช้งาน

```
Switch1(config-if)#switch mode trunk
Switch1(config-if)#
Switch1(config-if)#no switch mode trunk
Switch1(config-if)#
```

Switchport access vlan <vlan id>

Command Mode : Interface Commands

เป็นคำสั่งที่ใช้กำหนดค่า vlan ให้กับ interface

ตัวอย่างการใช้งาน

```
Switch1(config-if)#switchport access vlan 20
Switch1(config-if)#
```

Switchport trunk allowed vlan all

Command Mode : Interface Commands

เป็นคำสั่งที่ใช้กำหนดค่า interface ให้สามารถเชื่อมต่อโดยมี vlan ทุกตัวผ่านไปได้

ตัวอย่างการใช้งาน

```
Switch1(config-if)#switchport trunk allowed vlan all
Switch1(config-if)#
```

Switchport trunk allowed vlan add <vlan id>

Command Mode : Interface Commands

เป็นคำสั่งที่ใช้กำหนดค่า interface ให้เพิ่มหมายเลข vlan ที่สามารถผ่านไปได้ใน trunk link

ตัวอย่างการใช้งาน

```
Switch1(config-if)#switchport trunk allowed vlan add vlan 20
Switch1(config-if)#
```

Switchport trunk allowed vlan remove <vlan id>

Command Mode : Interface Commands

เป็นคำสั่งที่ใช้กำหนดค่า interface ให้ลบหมายเลข vlan ที่สามารถผ่านไปได้ออกจาก trunk link

ตัวอย่างการใช้งาน

```
Switch1(config-if)#switchport trunk allowed vlan remove vlan 20
Switch1(config-if)#
```

Vlan <vlan id> mtu <mtu size>
Command Mode : VLAN Commands
เป็นคำสั่งที่ใช้กำหนดค่า MTU ให้กับ vlan ที่กำหนด **ตัวอย่างการใช้งาน** Switch1(vlan)#vlan 20 mtu 2500 Switch1(vlan)#

Vlan <vlan id> name <vlan name>
Command Mode : VLAN Commands
เป็นคำสั่งที่กำหนด ชื่อให้กับ vlan **ตัวอย่างการใช้งาน** Swtich1(vlan)#vlan 20 name isag_group Switch1(vlan)#

No vlan <vlan id>
Command Mode : Global Configuration Commands, VLAN Commands
เป็นคำสั่งที่ใช้ถอดหมายเลข vlan ออกโดย interface ที่ถูกถอดออกนั้นจะกลายเป็น default vlan id (1) **ตัวอย่างการใช้งาน** Switch1(config)#no vlan 20 Switch1(config)#

No spanning-tree priority
Command Mode : Global Configuration Commands
เป็นคำสั่งที่ใช้สำหรับ disable การคิดค่า priority ในการทำ spanning-tree โดยเมื่อใช้คำสั่งนี้แล้ว ค่า priority จะกลายเป็นค่า 32768 (default) **ตัวอย่างการใช้งาน** Switch1(config)#no spanning-tree priority Switch1(config)#

No spanning-tree forward-time
Command Mode : Global Configuration Commands
เป็นคำสั่งที่ใช้สำหรับ disable ค่า forward-time โดยเมื่อใช้คำสั่งนี้แล้ว ค่า forward-time จะกลายเป็นค่า 15 (default)

ตัวอย่างการใช้งาน

Switch1(config)#no spanning-tree forward-time

Switch1(config)#

No spanning-tree hello-time

Command Mode : Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับ disable ค่า hello-time ในการทำ spanning-tree โดยเมื่อใช้คำสั่งนี้แล้ว ค่า hello-time จะกลายเป็นค่า 2 (default)

ตัวอย่างการใช้งาน

Switch1(config)#no spanning-tree hello-time

Switch1(config)#

No spanning-tree max-age

Command Mode : Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับ disable ค่า max-age ในการทำ spanning-tree โดยเมื่อใช้คำสั่งนี้แล้ว ค่า max-age จะกลายเป็นค่า 20 (default)

ตัวอย่างการใช้งาน

Switch1(config)#no spanning-tree max-age

Switch1(config)#

No spanning-tree vlan <vlan id>

Command Mode : Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับ disable spanning-tree กับ vlan ที่กำหนดให้

ตัวอย่างการใช้งาน

Switch1(config)#no spanning-tree vlan 20

Switch1(config)#

Spanning-tree cost <cost>

Command Mode : Interface Commands

เป็นคำสั่งที่ใช้สำหรับ ตั้งค่า cost ให้ spanning-tree

ตัวอย่างการใช้งาน

Switch1(config-if)#spanning-tree cost 10

Switch1(config-if)#

Spanning-tree vlan <vlan id> priority <priority>
Command Mode : Global Configuration Commands
เป็นคำสั่งที่ใช้สำหรับ ตั้งค่า priority ให้กับ vlan สำหรับทำ เปอร์วีแลนสแปนนิ่งทรี **ตัวอย่างการใช้งาน** Switch1(config)#spanning-tree vlan 2 priority 15 Switch1(config)#

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.cisco.com

[3] http://developer.java.sun.com/developer/onlineTraining/

[4] http://www-ece2.engr.ucf.edu/~xhu/

[5] http://www.javacoffeebreak.com/articles/

[6] http://www.ryerson.ca/~dgrimsha/courses/cps841/