โปรแกรมจำลองการทำงานของสวิตช์

Switch Simulator

นางสาวปิยะอร คงศักดิ์ตระกูล

นางสาวพรพรรณ อรรถวานิช

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

โปรแกรมจำลองการทำงานของสวิตช์

Switch Simulator

โดย

นางสาวปิยะอร คงศักดิ์ตระกูล

นางสาวพรพรรณ อรรถวานิช

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

อาจารย์อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมจำลองการทำงานของสวิตช์

Switch Simulator

ผู้จัดทำ

1. นางสาวปิยะอร คงศักดิ์ตระกูล รหัสนักศึกษา 42010207
2. นางสาวพรพรรณ อรรถวานิช รหัสนักศึกษา 42010119

..... อาจารย์ที่ปรึกษา

(อาจารย์ ธนา หงษ์สุวรรณ)

..... อาจารย์ที่ปรึกษา

(อาจารย์ อัครเดช วัชระภูพงษ์)

โปรแกรมจำลองการทำงานของสวิตช์

นางสาวปิยะอร คงศักดิ์ตระกูล 42010207
นางสาวพรพรรณ อรรถวานิช 42010119
อาจารย์ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

ระบบเครือข่ายได้เข้ามามีบทบาทกับชีวิตประจำวันมากขึ้น ความต้องการในการนำคอมพิวเตอร์ไปใช้ในการติดต่อสื่อสารหรือ เชื่อมต่อเพื่อให้บริการแก่กันก็มากขึ้นตามไปด้วย โดยเฉพาะในปัจจุบัน สวิตช์ได้รับความนิยมมากขึ้น เนื่องจากว่าสามารถทำงานได้หลากหลาย ทั้งในเลเยอร์สอง เลเยอร์สาม และเลเยอร์สี่ แต่เนื่องจากความไม่พร้อมในหลาย ๆ ด้าน จึงทำให้มีน้อยคนที่จะมีความชำนาญในการติดตั้งสวิตช์

ดังนั้นการจัดทำโปรแกรมจำลองการทำงานและติดตั้งของสวิตช์ จึงเป็นแนวคิดเพื่อเปิดโอกาสให้กับบุคคลทั่วไปที่มีความสนใจ ในการติดตั้งและตั้งค่าอุปกรณ์สวิตช์ ได้ฝึกฝน โดยที่ไม่จำเป็นต้องมีสวิตช์จริง ๆ เพื่อก่อให้เกิดความชำนาญมากขึ้น โดยโปรแกรมจะมีความสามารถในการจำลองการเรียนรู้แมคแอดเดรส, การใช้สแปนนิ่งทรีโพรโตคอล, การทำแลนเสมือนและสามารถแสดงภาพเครือข่ายจำลองได้

Switch Simulator

Miss Piyaorn Khongsaktrakool
Miss Pornpan Attavanich
Mr. Thana Hongsuwan Advisor
Mr. Akkradach Watcharapupong Advisor

Abstract

Computer network is getting more important in our life these days. The demand of using computer for communication and connection with one another has increased as well. Nowadays, switch is well known because of its plenty of capability. Switch can work in more than one layer such as data link layer, network layer and transport layer. Because of many problems, there are a few people who are experts in setting up the switch.

For this reason, this project is established to support person who is interested in setting up the switch and practicing without real switch. The switch simulator software is able to work on address learning, spanning tree protocol, VLANs and can show the network in graphic mode.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลงได้ ด้วยความช่วยเหลือ และการสนับสนุนเป็นอย่างดีบุคคลหลาย ๆ ฝ่ายด้วยกัน บุคคลสองท่านแรกที่ต้องกล่าวถึงคือ อาจารย์ธนา หงษ์สุวรรณ และ อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษาปริญญานิพนธ์ ซึ่งต้องขอขอบพระคุณเป็นอย่างสูง

โครงงานและปริญญานิพนธ์ฉบับนี้เสร็จสมบูรณ์ลงได้ เนื่องจากคำแนะนำจากอาจารย์ที่ปรึกษาทั้งสองท่านคือ อาจารย์ธนา หงษ์สุวรรณและอาจารย์อัครเดช วัชระภูพงษ์ ที่คอยแนะนำ เป็นที่ปรึกษาและเอาใจใส่กับการทำโครงงานนี้เป็นอย่างดี ซึ่งทางคณะผู้จัดทำขอขอบพระคุณอาจารย์ที่ปรึกษาทั้งสองท่านเป็นอย่างสูง

นอกเหนือจากนี้ ก็ต้องขอขอบพระคุณคณะอาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังเป็นอย่างยิ่งที่ได้ช่วยประสิทธิ์ประสาทวิชาความรู้ให้แก่คณะผู้จัดทำ อีกทั้งภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้เอื้อเฟื้ออุปกรณ์ ทรัพยากรสถานที่และอำนวยความสะดวกต่าง ๆ ในการทำโครงงานนี้ด้วย

ขอขอบใจเพื่อน ๆ และพี่ ๆ ที่ห้อง ISAG ทุกคนที่ช่วยเหลือในการทำงานและแก้ไขอุปสรรคต่าง ๆ ในการทำงานให้ผ่านพ้นไปได้ด้วยดี เป็นที่ปรึกษาและเป็นกำลังใจตลอดมา

สุดท้ายนี้ต้องขอขอบพระคุณบุคคลที่สำคัญที่สุด คือคุณพ่อ คุณแม่ที่เคารพและเป็นที่รักยิ่ง ผู้ที่ให้กำเนิด คอยสั่งสอน ให้ความรัก และให้การศึกษาอย่างสูงสุด พร้อมทั้งการให้การสนับสนุนปัจจัยในด้านต่าง ๆ นับเป็นพระคุณอย่างสูงสุดหาที่เปรียบมิได้ คณะผู้จัดทำขอระลึกพระคุณอันยิ่งใหญ่สุดประมาณนี้ไว้จนกว่าชีวิตจะหาไม่ และกราบขอบพระคุณทุกท่านที่ไม่ได้เอ่ยนามไว้ ณ ที่นี่ด้วย

นางสาวปิยะอร คงศักดิ์ตระกูล
นางสาวพรพรรณ อรรถวานิช

สารบัญ

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เนื่องจากในปัจจุบันนี้ คนที่มีความเชี่ยวชาญในการติดตั้งและตั้งค่าอุปกรณ์สวิตช์มีไม่มาก ทั้งที่การติดตั้งและตั้งค่าอุปกรณ์สวิตช์นั้น ไม่ใช่สิ่งที่ยาก แต่เนื่องมาจากสาเหตุที่ว่า บุคคลทั่วไปมีโอกาสน้อยในการจะเข้าไปสัมผัสกับการใช้งานสวิตช์ ทำให้การติดตั้งและตั้งค่าเป็นไปด้วยความยากลำบาก ซึ่งในบางครั้งอาจทำให้ไม่สามารถใช้งานสวิตช์ได้อย่างเต็มความสามารถของมัน และบุคคลที่ได้เข้าไปใช้งานสวิตช์ ส่วนมากจะต้องอยู่ในองค์กร และทำงานในด้านเกี่ยวกับระบบเครือข่าย แม้กระทั่งบุคคลประเภทนี้ ก็ยังไม่สามารถศึกษาการทำงานของสวิตช์ได้อย่างเต็มที่ ด้วยเหตุที่ว่า การเข้าไปศึกษาและตั้งค่าสวิตช์ อาจก่อให้เกิดผลกระทบกับระบบเครือข่ายขององค์กรได้

ด้วยเหตุนี้ จึงได้มีการคิดโปรแกรมจำลองการทำงานของสวิตช์ เพื่อสร้างโอกาสให้กับบุคคลทั่วไปที่มีความสนใจ ได้ทดลองใช้งานสวิตช์ในการติดตั้งและตั้งค่าอุปกรณ์ รวมทั้งศึกษาการทำงาน เพื่อเป็นแนวทางในการใช้งานสวิตช์ต่อไป

1.2 หลักการและเหตุผล

โปรแกรมจำลองการทำงานของสวิตช์ เป็นโปรแกรมที่พัฒนาขึ้นเพื่อจำลองการทำงานและการติดตั้งอุปกรณ์สวิตช์จริง ๆ โดยใช้คำสั่งเพื่อสร้างสวิตช์จำลองขึ้นมา และติดตั้งค่าอุปกรณ์ให้กับสวิตช์ มีการเรียกใช้โพรโตคอลต่าง ๆ และจำลองการทำงาน การส่งข้อมูลของสวิตช์ในเลเยอร์2 โดยคำสั่งต่าง ๆ ที่เลือกมาใช้ในการติดตั้ง เป็นเพียงบางส่วนของคำสั่งติดตั้งสวิตช์ ที่เป็นคำสั่งหลัก ๆ ที่สำคัญเท่านั้น ซึ่งโปรแกรมนี้เขียนด้วยภาษาจาวา จึงสามารถทำงานได้กับทุกแพลตฟอร์ม

1.3 วัตถุประสงค์ของปริญญานิพนธ์

ปริญญานิพนธ์ที่จัดทำขึ้นนี้ จัดทำภายใต้วัตถุประสงค์หลัก 2 ประการ ได้แก่

1. เพื่อจัดทำโปรแกรมจำลองการทำงานของสวิตช์ ทำให้สามารถทดลองติดตั้งและตั้งค่าอุปกรณ์สวิตช์ รวมไปถึงศึกษาการทำงานของสวิตช์ได้ โดยไม่ต้องใช้สวิตช์จริง ๆ
2. เพื่อศึกษาการทำงานของสวิตช์ และกลไกการทำงานของโพรโตคอลที่ใช้ในสวิตช์ เลเยอร์สอง
3. เพื่อศึกษาและฝึกฝนการเขียนโปรแกรมแพลตฟอร์มจาวา

1.4 ขอบเขตของปริญญานิพนธ์

สร้างโปรแกรมจำลองการทำงานของสวิตช์ในเลเยอร์ 2 โดย

- สามารถตั้งค่าสวิตช์ได้ในระดับพื้นฐาน
- มีการเรียนรู้ค่าแมคแอดเดรส (MAC address) เก็บเป็นตารางแมคแอดเดรส (MAC address Table) คือการเรียนรู้แมคแอดเดรส (Address Learning) นั่นเอง
- สามารถเรียกใช้สแปนนิ่งทรีโพรโตคอล (Spanning Tree Protocol (STP))
- สร้างการทำงานแบบวีแลน (VLANs) ได้
- จำลองระบบเครือข่ายที่สร้างขึ้นในรูปแบบของกราฟฟิกส์ (Graphics) ได้

1.5 ขั้นตอนการดำเนินงาน

1. ศึกษาการทำงานของสวิตช์ในเลเยอร์ 2 (Data Link Layer)
2. ศึกษาและรวบรวมคำสั่งที่ใช้ในการติดตั้งและตั้งค่าอุปกรณ์สวิตช์
3. ศึกษาโพรโตคอลที่เกี่ยวข้อง
4. ออกแบบคลาสและกำหนดรายละเอียดของโปรแกรมทั้งหมด
5. สร้างคลาสที่ใช้ในการทำการเรียนรู้แมคแอดเดรส
6. สร้างคลาสที่ใช้ในการทำสแปนนิ่งทรีโพรโตคอล
7. สร้างคลาสที่ใช้ในการสร้างการทำงานแบบวีแลน
8. เขียนโปรแกรมติดต่อกับผู้ใช้ให้สามารถจัดการกับคำสั่งพื้นฐานในการติดตั้งและตั้งค่าอุปกรณ์สวิตช์
9. ทำภาพจำลองระบบเครือข่ายที่ผู้ใช้สร้างขึ้น
10. ทดสอบโปรแกรม, ตรวจสอบหาข้อผิดพลาดและทำการแก้ไข

บทที่ 2
ความรู้พื้นฐาน

2.1 เครือข่ายคอมพิวเตอร์ (Computer Network)

เครือข่ายคอมพิวเตอร์ หมายถึง เครื่องคอมพิวเตอร์ตั้งแต่สองเครื่องขึ้นไปที่เป็นอิสระต่อกัน นำมาเชื่อมต่อถึงกันได้โดยไม่คำนึงถึงระยะทางระหว่างเครื่องทั้งสอง โดยมีจุดประสงค์เพื่อให้เครื่องคอมพิวเตอร์สามารถติดต่อสื่อสาร และแลกเปลี่ยนข้อมูลระหว่างกันได้ เครือข่ายคอมพิวเตอร์เริ่มจากเครือข่ายขนาดเล็กภายในองค์กรที่เชื่อมโยงกันภายใต้สภาพพื้นที่จำกัดซึ่งเรียกว่า *เครือข่ายท้องถิ่น* (LAN: Local Area Network) เมื่อเชื่อมโยงเครือข่ายเข้าด้วยกัน และขยายขอบเขตครอบคลุมพื้นที่ระหว่างเมือง หรือระหว่างประเทศ ก็จะเรียกเครือข่ายนั้นว่า *เครือข่ายวงกว้าง* (WAN: Wide Area Network)

2.2 โปรแกรมสำหรับเครือข่าย (Network Software)

ลำดับขั้นของโพรโตคอล (Protocol Hierarchies)

เพื่อเป็นการลดความซับซ้อนของการออกแบบโปรแกรมทั้งระบบในคราวเดียวกัน ระบบโปรแกรมเครือข่ายส่วนมากจะแบ่งเป็นการทำงานออกเป็นหลายระดับ หรือหลายชั้น แต่ละชั้นจะสร้างฟังก์ชันการทำงานขึ้นโดยอาศัยการทำงานของฟังก์ชันต่าง ๆ ที่สร้างไว้ในชั้นระดับล่างลงมา จำนวนชั้น, ชื่อที่เรียก และฟังก์ชันการทำงานของเครือข่ายต่าง ๆ จะแตกต่างกันออกไป อย่างไรก็ตาม ทุกระบบจะมีแนวความคิดอย่างเดียวกัน คือการเรียกใช้บริการจากชั้นล่าง และการให้บริการแก่ชั้นบนโดยซ่อนรายละเอียดและ ความซับซ้อนของฟังก์ชันในแต่ละชั้นไว้ภายใน

ในการสื่อสารที่เกิดขึ้นระหว่างผู้ส่งข้อมูลกับผู้รับข้อมูลนั้น จะเสมือนว่าเป็นการติดต่อในชั้นเดียวกัน โดยในแต่ละชั้นจะมีโพรโตคอลของตนเอง ซึ่งจะช่วยให้ผู้ส่งข้อมูลและผู้รับข้อมูลสามารถติดต่อกันได้ แต่ในความจริงแล้ว การสื่อสารจะเกิดขึ้นจริงโดยผ่านสายสื่อสารที่อยู่ชั้นหนึ่งเท่านั้น ข้อมูลที่สื่อสารในระหว่างชั้นต่าง ๆ จะถูกส่งต่อกันเป็นลำดับจากชั้นบนสู่ชั้นล่าง จากฝ่ายของทางผู้ส่งข้อมูล และทางฝ่ายของผู้รับข้อมูลจะทำการรับข้อมูลจากชั้นหนึ่ง แล้งจึงถูกแยกข้อมูลออกไปทีละชั้น จนถึงชั้นบนสุดเพื่อใช้ในการประมวลผลต่อไป

ระบบมาตรฐาน OSI (The OSI Reference Model)

เมื่อคอมพิวเตอร์ของเรามีการรับส่งข้อมูลกับคอมพิวเตอร์เครื่องอื่น ๆ การเชื่อมต่อคอมพิวเตอร์หลาย ๆ เครื่องเข้าด้วยกันเป็นระบบเครือข่ายก็เกิดขึ้น อย่างไรก็ตาม การเชื่อมต่อคอมพิวเตอร์คนละระบบหรือคนละยี่ห้อเป็นสิ่งที่ทำได้ยากในยุคแรก ๆ ของการสื่อสารข้อมูล เนื่องจากขาดมาตรฐานส่วนกลางที่จำเป็นต้องใช้ในการรับส่งข้อมูล ส่วนมากแต่ละยี่ห้อก็จะมีมาตรฐานของตนเอง ซึ่งเข้ากันไม่ได้กับของยี่ห้ออื่น ทำให้ผู้ใช้ต้องผูกติดอยู่กับผู้ผลิตแต่ละยี่ห้อ และเป็นขีดจำกัดในการเชื่อมต่อคอมพิวเตอร์คนละชนิด ไม่ให้รับส่งข้อมูลกันได้ ระบบคอมพิวเตอร์ในยุคนั้น จึงเป็นระบบปิด (Closed System) นั่นเอง

ปัญหานี้ ทำให้หน่วยงานกำหนดมาตรฐานสากล คือ International Standards Organization หรือ ISO จัดการกำหนดโครงสร้างทั้งหมดที่จำเป็นต้องใช้ในการสื่อสารข้อมูลจากคอมพิวเตอร์ระบบหนึ่งไปยังคอมพิวเตอร์อีกระบบหนึ่งขึ้น จุดมุ่งหมายก็เพื่อเปิดช่องทางให้ข้อมูลที่เก็บอยู่ในระบบคอมพิวเตอร์หนึ่ง ๆ รับส่งไปยังคอมพิวเตอร์ที่เป็นระบบเดียวกัน หรือต่างระบบได้อย่างอิสระ โดยไม่ขึ้นกับผู้ผลิตอย่างที่เป็นอยู่ในอดีต ซึ่งเป็นการทำงานแบบที่เรียกว่า ระบบเปิด (Open systems) เราเรียกโครงสร้างของมาตรฐานการรับส่งข้อมูลนี้ว่า (Open System Interconnection) หรือ OSI ซึ่งจัดทำขึ้นราวกลางปี ค.ศ.1970 โดยองค์การ International Standards Organization (ISO) และใช้อ้างอิงกันมาจนถึงในยุคปัจจุบัน

โมเดล OSI มีทั้ง 7 ชั้นด้วยกัน หรือที่เราเรียกว่า OSI 7-Layers Model โดยตัวโครงสร้างจะเน้นความสำคัญของรูปแบบการติดต่อสื่อสาร ระหว่างระบบเปิด (open systems) กับระบบเปิด จึงสามารถนำไปใช้อ้างอิงได้ในระดับสากลอย่างแท้จริง

แนวความคิดของการกำหนดมาตรฐานเป็นแบบชั้นสื่อสาร (layers) คือ

1. ชั้นสื่อสารแต่ละชั้นถูกกำหนดขึ้นมาตามบทบาทที่แตกต่างกัน
2. แต่ละฟังก์ชันในชั้นใด ๆ จะต้องกำหนดขึ้นมาโดยใช้แนวความคิดในระดับสากลเป็นวัตถุประสงค์หลัก
3. ขอบเขตความรับผิดชอบของแต่ละชั้น จะต้องกำหนดขึ้นมา เพื่อจำกัดปริมาณการแลกเปลี่ยนข้อมูลและ ผลกระทบข้างเคียงระหว่างการติดต่อให้มีน้อยที่สุด
4. ในแต่ละชั้น จะเสมือนเชื่อมต่ออยู่กับชั้นเดียวกันของอีกฝั่งหนึ่ง โดยทำงานเสมือนกับว่า มีการติดต่อรับส่งข้อมูลกับกลไกในชั้นเดียวกัน แต่จะมีเพียงชั้นหนึ่ง ซึ่งเป็นชั้นล่างสุดเท่านั้นที่มีการเชื่อมต่อ และรับส่งข้อมูลผ่านสายส่งข้อมูลระหว่างคอมพิวเตอร์ทั้งสองระบบ
5. ในแต่ละชั้น ที่ทำหน้าที่รับส่งข้อมูลจะมีการติดต่อรับส่งข้อมูลกับชั้นที่อยู่ติดกับตนเองเท่านั้น ไม่สามารถติดต่อรับส่งข้อมูลข้ามชั้นได้

Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

รูปที่ 1-1 แสดงลำดับชั้นของมาตรฐาน OSI

ชั้นสื่อสารกายภาพ (The Physical Layer)

ชั้นนี้ คือชั้นที่ 1 หรือชั้นล่างสุด จะทำงานเกี่ยวข้องโดยตรงกับอุปกรณ์สื่อสารต่าง ๆ ทำหน้าที่ในการกำหนดวิธีการควบคุมการรับ และ การส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ในระดับบิต ได้แก่ การส่งบิต 0 และ 1 จะแทนด้วยกระแสไฟฟ้ากี่โวลต์ แต่ละบิตในระยะเวลาในการส่งนานเท่าไหร่ การส่งเป็นแบบทางเดียวหรือสองทาง เป็นต้น จะเห็นได้ว่า การทำงานในชั้นนี้จะเกี่ยวข้องกับการทำงานของอุปกรณ์สัญญาณไฟฟ้า (หรือสัญญาณใด ๆ) ขั้นตอนในการใช้อุปกรณ์เหล่านั้น และความสัมพันธ์กับสื่อที่ใช้รับ – ส่งสัญญาณ

ชั้นสื่อสารการเชื่อมต่อข้อมูล (The Data Link Layer)

หน้าที่หลักของชั้นนี้คือ ทำการรวบรวมข้อมูลจากชั้นที่ 1 ตรวจสอบความถูกต้องของข้อมูล แล้วส่งข้อมูลที่ปราศจากข้อผิดพลาดนี้ให้กับชั้นที่ 3 ต่อไป โดยปกติผู้ส่งข้อมูลจะแบ่งข้อมูลที่มีความยาวมากออกเป็นกลุ่มข้อมูลย่อย ๆ แต่ละส่วนย่อยเรียกว่า ดาต้าเฟรม (Data Frame) ซึ่งจะมีขนาดคงที่ ชุดของดาต้าเฟรมสำหรับข้อมูลที่ต้องการส่งไปให้ผู้รับก็จะถูกส่งไปทีละเฟรม ตั้งแต่เฟรมแรกไปจนครบทุกเฟรม ข้างฝ่ายผู้รับจะตอบสนองโดยการส่งดาต้าเฟรมพิเศษ เรียกว่า เฟรมตอบรับ (Acknowledgement frame) ไปถึงผู้ส่งเพื่อเป็นการบอกให้ทราบว่า ได้รับข้อมูลแล้ว กระบวนการรับ-ส่งข้อมูลนี้ก็จะเสร็จสิ้นสมบูรณ์

การรับ-ส่งข้อมูลในชั้นที่ 1 นั้นจะไม่รับรู้ในเรื่องโครงสร้างข้อมูล คือจะมองเห็นข้อมูลว่าเป็นบิต 0 หรือบิต 1 กลุ่มหรือชุดหนึ่งที่เรียงตามลำดับ เรียกว่า กระแสบิต (Bit Stream) จึงเป็นหน้าที่ของชั้นที่ 2 ในการทำการตรวจสอบความถูกต้อง ซึ่งทำได้โดยการเพิ่มข้อมูลสำหรับการตรวจสอบติดไว้กับข้อมูลทุกเฟรม

การส่งข้อมูลผ่านระบบเครือข่ายใด ๆ ก็ตาม ข้อมูลที่ส่งนั้นมีโอกาสที่จะเสียหายหรือสูญหายไปเลยก็ได้ โปรแกรมในชั้นที่ 2 จะต้องสามารถตรวจสอบความผิดปกตินี้ได้เอง หรืออาจตอบสนองต่อการตรวจพบโดยโปรแกรมในชั้นที่ 1 เมื่อพบความผิดปกตินี้แล้วก็จะต้องมีวิธีในการแก้ไข เช่น แจ้งให้ผู้ส่งข้อมูลได้ส่งข้อมูลชุดเดิมกลับมาใหม่ (เรียกเฟรมนี้ว่า Duplicate Frame) อย่างไรก็ตาม การส่งข้อมูลซ้ำทำให้เกิดปัญหาตามมา

ในกรณีที่ชุดข้อมูลไม่ได้สูญหายไปไหน เพียงแต่ใช้เวลาเดินทางมากกว่าปกติ ดังนั้นข้อมูลชุดเดียวกันก็จะมาถึงผู้ใช้ทั้งสองเฟรม โปรแกรมในชั้นนี้จึงต้องหาวิธีการตรวจสอบและกำจัดเฟรมที่ซ้ำออกไป

ชั้นสื่อสารเครือข่าย (The Network Layer)

ชั้นนี้มีหน้าที่รับผิดชอบในการควบคุมการติดต่อรับ-ส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ (โหนด) ต่าง ๆ ในระบบเครือข่ายให้เป็นไปด้วยความเรียบร้อย สิ่งที่สำคัญที่สุด คือการกำหนดเส้นทางเดินของข้อมูลจากโหนดผู้ส่งข้อมูลไปตามโหนดต่าง ๆ จนถึงโหนดผู้รับข้อมูลในที่สุด โฮสต์บางกลุ่มจะกำหนดเส้นทางเดินข้อมูลโดยศึกษาระบบเครือข่ายแล้วสร้างตารางเส้นทางเดินข้อมูลแบบถาวร โฮสต์บางกลุ่มจะกำหนดเส้นทางเดินข้อมูลในตอนเริ่มต้นของการสื่อสาร ดังนั้นการสื่อสารในครั้งต่อไป (ติดต่อกับโหนดเดิม) อาจจะเปลี่ยนไปใช้เส้นทางอื่นได้ โฮสต์ในกลุ่มที่มีวิธีการซับซ้อนมาก จะกำหนดเส้นทางเดินข้อมูลในระดับแพ็กเก็ต กรณีที่มีผู้ส่งข้อมูลพร้อม ๆ กันหลาย ๆ จุดจะทำให้เกิดความคับคั่งของข้อมูลคล้ายกับสภาวะจราจรในชั่วโมงเร่งด่วน ซึ่งมีปริมาณรถยนต์มากจนทำให้การจราจรติดขัด โฮสต์ในกลุ่มนี้ก็จะปรับเส้นทางเดินข้อมูลของแต่ละแพ็กเก็ตให้เหมาะสมกับสภาวะของระบบเครือข่ายอยู่ตลอดเวลา

การส่งผ่านข้อมูลในระบบเครือข่ายอาจมีการบันทึก ผู้ส่ง ผู้รับ และปริมาณข้อมูลที่ไหลผ่านโฮสต์หรือ เราเตอร์ต่าง ๆ เพื่อประโยชน์ทางด้านการคิดค่าบริการซึ่งจะมีความซับซ้อนมากขึ้นถ้าข้อมูลไหลผ่านระบบเครือข่ายย่อยที่มีการคิดอัตราค่าบริการต่างกัน

ชั้นจัดการนำส่งข้อมูล (The Transport Layer)

โปรแกรมในชั้นนี้ มีหน้าที่หลักในการรับข้อมูลมาจากชั้นที่ 5 ซึ่งอาจต้องแบ่งข้อมูลออกเป็นแพ็กเก็ตขนาดย่อม (ในกรณีที่ข้อมูลมีปริมาณมาก) หลาย ๆ แพ็กเก็ต แล้วจึงส่งข้อมูลทั้งชุดต่อไปให้โปรแกรมในชั้นที่ 3 ทางด้านโปรแกรมในชั้นที่ 4 ก็จะทำหน้าที่ประกอบแพ็กเก็ตชุดนี้ให้กลับมารวมกันเป็นข้อมูลชุดเดิม

ในภาวะปกติ การเชื่อมต่อการสื่อสารจะเป็นการจัดตั้งหน้าต่างสื่อสาร (Session) ระหว่างผู้ส่งและผู้รับตามที่เกิดขึ้น ถ้าต้องการเพิ่มประสิทธิภาพก็อาจสร้างโพสเซสของโปรแกรมในชั้นนี้ขึ้นมาหลาย ๆ โพรเซสเพื่อช่วยกันจัดส่งข้อมูลให้เร็วขึ้น แต่ถ้าเน้นในด้านความประหยัดก็อาจทำในทางตรงกันข้ามนั่นคือ การยุบรวมโพรเซสหลาย ๆ โพรเซสให้เหลือจำนวนน้อยลงแล้วจึงจัดการให้โพรเซสที่เหลืออยู่ทำการส่งข้อมูลทั้งหมดโดยการใช้ช่องสื่อสารร่วมกัน

โปรแกรมในชั้นนี้เป็นผู้กำหนดประเภทของการให้บริการต่าง ๆ รวมไปถึงการอำนวยความสะดวกในการใช้ระบบเครือข่ายซึ่งแบ่งออกได้เป็น 3 ประเภท ประเภทแรกเป็นการให้บริการแบบจุด-ต่อ-จุด โดยเน้นการรับประกันความถูกต้องของข้อมูลเป็นสำคัญ ประเภทที่สองเน้นการให้บริการข้อมูล ข้อมูลในระดับแพ็กเก็ตซึ่งแม้ว่าจะไม่รับประกันการสูญหายของข้อมูลแต่ก็ให้ความคล่องตัวสูงกว่าแบบแรก (การรับประกันความถูกต้องของข้อมูลสามารถทำในชั้นอื่นได้) ประเภทที่สามเป็นการส่งข้อมูลแบบกระจายข่าวเพื่อประโยชน์ในการส่งข้อมูลชุดเดียวกันไปยังผู้ใช้หลายจุดพร้อมกัน

โปรแกรมในชั้นนี้ติดต่อถึงกันผ่านช่องสัญญาณเสมือน (Virtual Channel) ระหว่างผู้ส่งและผู้รับโดยตรงเรียกว่า เป็นการติดต่อกับ end-to-end connection ในขณะที่โปรแกรมในสามชั้นแรกนั้นเป็นการติดต่อแบบจุด-ต่อ-จุด ซึ่งผู้รับอาจไม่ใช่ผู้รับข้อมูลแต่เป็นเพียงโหนดตัวกลางในการรับแล้วส่งข้อมูลต่อไปตามเส้นทางเดินข้อมูลที่ถูกกำหนดไว้

นอกจากการใช้ช่องสื่อสารร่วมกันแล้วโปรแกรมในชั้นนี้ จะต้องมีความสามารถในการจัดตั้งหน้าต่างสื่อสารกับโหนดอื่น ๆ ในระบบเครือข่ายและจัดการยกเลิกเมื่อการสื่อสารสิ้นสุดลง โปรแกรมในชั้นนี้ยังต้องมีวิธีการกำหนดการตั้งชื่อให้แก่ตนเองและแนะนำให้ผู้อื่นในระบบได้รู้จัก รวมทั้งการควบคุมการไหลของข้อมูล ซึ่งมีทั้งในระดับโฮสต์และระดับเราเตอร์โดยมีวัตถุประสงค์ในการควบคุมการรับและส่งข้อมูลโดยเฉพาะในกรณีที่ผู้ส่งจัดการส่งข้อมูลเร็วเกินกว่าทางผู้ใช้จะทำงานได้ทัน

ชั้นหน้าต่างสื่อสาร (The Session Layer)

ชั้นนี้ ทำหน้าที่เป็นผู้กำหนดวิธีการควบคุมการเชื่อมต่อระหว่างผู้รับข้อมูลและผู้ส่งข้อมูลตั้งแต่เริ่มต้นการสื่อสารไปจนยุติการสื่อสาร เช่น การติดต่อขอใช้โฮสต์จากเครื่องคอมพิวเตอร์ที่อยู่ไกลออกไป (Remote Login) โดยภาพรวมแล้ว การให้บริการในชั้นนี้จะคล้ายกับบริการที่มีอยู่ในชั้นที่ 4 แต่ในชั้นนี้จะให้บริการหลายอย่างที่เป็นประโยชน์มากกว่าสำหรับการประยุกต์ใช้งานบางประเภท

หน้าที่สำคัญอย่างหนึ่งคือ บริหารการแลกเปลี่ยนข่าวสาร อันได้แก่การกำหนดให้การแลกเปลี่ยนข่าวสารเป็นไปแบบสองทางในเวลาเดียวกัน (Full duplex) หรือถ้าเป็นการสื่อสารแบบทางเดียวแต่สลับทิศได้ (Half duplex) ก็จะต้องเป็นผู้จัดลำดับให้ทั้งผู้ใช้และผู้ส่งทำการส่งข้อมูลได้คล้ายกับการควบคุมสับหลีกรถไฟ

สำหรับการสื่อสารประเภทที่ต้องใช้โทเก้น (Token) โปรแกรมในชั้นนี้ จะเป็นผู้บริหารการใช้โทเก้นเพื่อให้โหนดต่าง ๆ ในระบบนี้ผลัดเปลี่ยนการครอบครองโทเก้นอย่างเป็นธรรม หรือถูกต้องตามลำดับความสำคัญ

หน้าที่อีกประการหนึ่งได้แก่การแก้ปัญหาความล้มเหลวในการส่งข้อมูลขนาดใหญ่มากระหว่างโหนดต่าง ๆ ในกรณีที่การส่งข้อมูลเกิดล้มเหลวกลางคันโดยไม่มีการแก้ไขใด ๆ โหนดทั้งสองก็จะต้องเริ่มต้นใหม่หมด ถ้าเกิดการล้มเหลวขึ้นอีกก็จะต้องเริ่มต้นใหม่อีก วิธีการแก้ไขวิธีหนึ่งก็คือการแทรกจุดตรวจสอบความถูกต้อง (Checkpoints) เข้าไปจำนวนหนึ่ง โดยจำนวนขึ้นกับปริมาณข้อมูล ในระหว่างการส่งข้อมูล จุดตรวจสอบทั้งหมดจะต้องถูกแทรกเข้าไปในข้อมูล ที่ตำแหน่งเดียวกันของทั้งผู้ส่งและผู้รับซึ่งเรียกว่าการ Synchronization หากเกิดการล้มเหลวขึ้น โปรแกรมในชั้นนี้ของผู้รับก็จะค้นหาจุดตรวจสอบจุดสุดท้ายก่อนการล้มเหลวเพื่อลบข้อมูลส่วนที่อยู่หลังจุดตรวจสอบนั้นทิ้งไป แล้วแจ้งให้ผู้ส่งเริ่มต้นการส่งข้อมูลใหม่จากจุดตรวจสอบนั้นแทนที่จะต้องเริ่มต้นใหม่ทั้งหมด

ชั้นนำเสนอข้อมูล (The Presentation Layer)

โปรแกรมที่ทำงานในระดับชั้นต้น ๆ นั้นที่ได้กล่าวมาแล้วนั้น จะให้ความสนใจในประสิทธิภาพของการรับ-ส่งข้อมูลและมองเห็นว่าข้อมูลคือกระแสบิต (Bit Stream) หรือกระแสไบต์ (Byte Stream) เท่านั้น โปรแกรมในชั้นนี้จะมองข้อมูลว่าเป็นสิ่งที่มีรูปแบบ (Syntax) และความหมาย (Semantics) มากกว่ากระแสบิตหรือไบต์ ความแตกต่างของการให้ความหมายข้อมูลของเครื่องคอมพิวเตอร์ในระบบต่าง ๆ เป็นปัญหาที่จะต้องได้รับการแก้ไขในระดับส่วนรวมไม่ใช่ให้แต่ละฝ่ายแก้ปัญหาโดยลำพัง การควบคุมรูปแบบและความหมายของข้อมูล การใช้รหัสแทนข้อมูล หรือการแทนข้อมูลด้วยระบบต่าง ๆ รวมไปถึงการเข้ารหัสและถอดรหัส สิ่งต่าง ๆ ที่กล่าวมานี้ล้วนแต่เป็นความรับผิดชอบของโปรแกรมในชั้นนี้

ชั้นสื่อสารการประยุกต์ (The Application Layer)

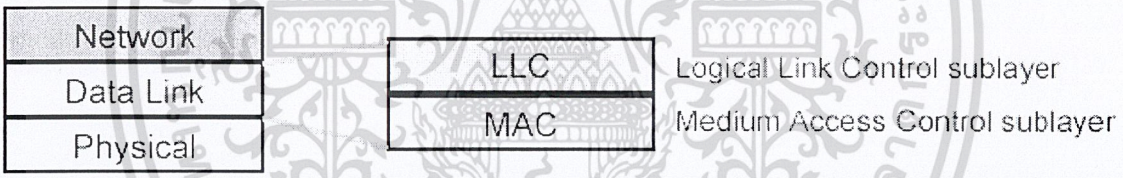
ในปัจจุบัน มีจอภาพเทอร์มินัล (Terminals) อยู่หลายร้อยชนิดทั่วโลกซึ่งส่วนใหญ่จะไม่สามารถใช้ทดแทนหรือใช้งานร่วมกันได้ การติดต่อระหว่างเครื่องคอมพิวเตอร์ที่อยู่คนละระบบเครือข่ายย่อยจึงไม่อาจสื่อสารกันได้โดยสมบูรณ์ โปรแกรมในชั้นประยุกต์จึงเข้ามามีบทบาทสำคัญสองด้านคือ การเป็นตัวกลาง หรือส่วนติดต่อระหว่างโปรแกรมประยุกต์ (Application Programs) กับโปรแกรมใน 6 ชั้นที่เหลือและ การกำหนดแบบมาตรฐานของจอ (Terminal Type)

การกำหนดแบบมาตรฐานของจอนั้น ไม่ได้เป็นการกำหนดวิธีสร้างจอเทอร์มินัลให้เหมือนกัน แต่จะคล้ายกับการสร้างจอเทอร์มินัลเสมือน (Virtual Terminal) ขึ้นบนจอเทอร์มินัลจริง ทั้งนี้เพื่อทำให้จอเทอร์มินัลทุกชนิดในโลกมีความเข้าใจตรงกัน เช่น ขนาดบริเวณที่ในการการแสดงผล การเคลื่อนย้ายตำแหน่งเคอร์เซอร์และการแสดงตัวอักษร ณ ตำแหน่งต่าง ๆ บนจอภาพ เป็นต้น จึงทำให้การใช้จอเทอร์มินัลเพื่อการสื่อสารบนระบบเครือข่ายเกิดขึ้นได้ แม้ว่าจะใช้จอเทอร์มินัลต่างแบบกันก็ตาม

บทที่ 3
ระบบเครือข่ายมาตรฐาน IEEE 802

องค์กร IEEE ได้กำหนดมาตรฐานโพรโตคอลสำหรับระบบเครือข่ายเฉพาะบริเวณไว้จำนวนหนึ่ง เป็นที่รู้จักกันในกลุ่มหมายเลข IEEE 802 โพรโตคอลที่ได้รับความนิยมในการนำมาใช้งาน 3 แบบคือ CSMA/CD, โทเก้นบัสและโทเก้นริง ซึ่งมีความแตกต่างกันในชั้นที่ 1 (Physical Layer) และชั้นที่ 2 (Data Link Layer) มาตรฐานนี้ได้รับการยอมรับจากองค์กรควบคุมมาตรฐานอื่น ๆ ได้แก่ ANSI (American National Standards Institute) และ ISO (International Standard Organization)

ข้อกำหนดเพิ่มเติมที่ IEEE สร้างขึ้นในทุกมาตรฐาน 802 คือการแยกระดับของชั้นที่ 2 (Data Link Layer) ออกเป็น 2 ส่วนย่อย โดยให้มาตรฐาน 802.2 ซึ่งเรียกว่า แอลแอลซี (LLC: Logical Link Control) เป็นส่วนเชื่อมต่อกับชั้นที่ 3 (Network Layer) อินเทอร์เฟซของแอลแอลซีในเครือข่ายแต่ละชนิด (802.3, 802.4 และ 802.5) จะมีรูปแบบเชื่อมต่อกับชั้นที่ 3 เช่นเดียวกันหมด



รูปที่ 3-1 แสดง Sub-Layer ของ Data Link Layer

การกำหนดรายละเอียดได้ถูกแยกเป็นหลายส่วนย่อย ส่วนแรกใช้รหัส 802.1 กำหนดโครงสร้างโดยรวมของมาตรฐาน 802 ทั้งหมด รวมทั้งกำหนดวิธีการสื่อสารขั้นต้น ส่วนที่สองคือ 802.2 อธิบายรายละเอียดของโพรโตคอลในชั้นที่ 2 ส่วนบน (Medium Access Sub-Layer) ซึ่งเป็นการควบคุมการเชื่อมต่อในระดับแอลแอลซีของการสื่อสารในชั้นนี้

3.1 มาตรฐาน IEEE 802.3 และระบบ Ethernet

มาตรฐาน IEEE 802.3 ออกมาสำหรับระบบเครือข่ายแลน (LAN) แบบ CSMA/CD ที่เรียกว่า 1-persistent กล่าวคือ สเตชันที่ต้องการส่งข้อมูล จะฟังสัญญาณในสายสื่อสาร หากพบว่าไม่มีผู้ใดกำลังส่งสัญญาณออกมา หรืออีกในนัยหนึ่งคือสายสัญญาณว่างนั่นเอง สเตชันนั้นก็จะดำเนินการส่งสัญญาณออกมาด้วยความน่าจะเป็นเท่ากับ 1 มิฉะนั้นแล้วก็จะต้องรอจนกว่าสายสัญญาณจะว่าง ในขณะที่กำลังส่งสัญญาณหากพบว่ามีสัญญาณซ้อนเกิดขึ้น ทุกสเตชันที่เกี่ยวข้องในการส่งสัญญาณจะต้องหยุดทำงานในทันทีเป็นช่วง

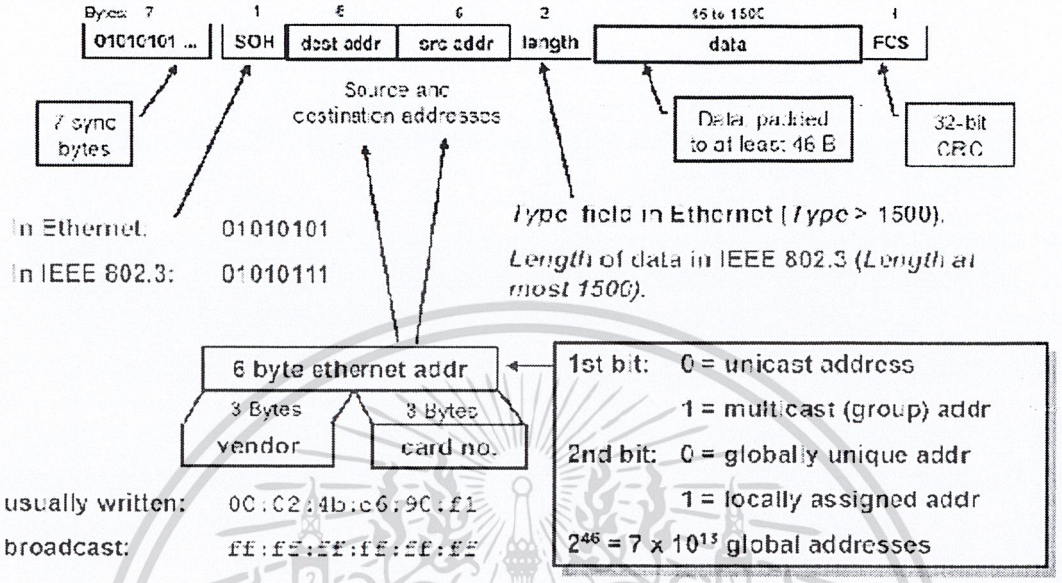
คาบเวลาหนึ่ง (ซึ่งมีค่าไม่แน่นอนและแตกต่างกันออกไปในแต่ละสเตชัน) จากนั้นจึงจะสามารถเริ่มต้นกระบวนการส่งสัญญาณใหม่

ต้นกำเนิดของมาตรฐานนี้มาจากระบบอะโลฮ่า เริ่มต้นในปี ค.ศ. 1970 เกิดจากการค้นคว้าและวิจัยของ Palo Alto Research Center (PARC) ซึ่งเป็นส่วนหนึ่งของบริษัทซีร็อก (Xerox) ต้องการให้สามารถใช้งานสื่อสารระหว่างคอมพิวเตอร์ได้อย่างกว้างขวาง และรวดเร็วจึงต้องใช้ความเร็วสูงในการส่งข้อมูล ระบบนี้ได้รับการตั้งชื่อว่า ระบบอีเทอร์เน็ต (Ethernet) มาจากคำว่า "Lunimiferous Ether" ซึ่งในครั้งหนึ่งในอดีตเคยเชื่อกันว่าสนามแม่เหล็ก (Electromagnetic) สามารถส่งผ่านห้วงอวกาศได้โดยอาศัยสารชนิดนี้เป็นพาหะ ต่อมาภายหลังนักฟิสิกส์ชื่อ Michelson-Morley (1887) ได้ทดลองให้เห็นว่า สนามแม่เหล็กสามารถเดินทางผ่านสุญญากาศได้โดยไม่ต้องอาศัยพาหะใด ๆ

การนำระบบอีเทอร์เน็ตมาใช้งานนั้นประสบความสำเร็จเป็นอย่างมาก บริษัทซีร็อก , DEC (Digital Equipment Corporation, Ltd.) และ Intel Corp. ได้ร่วมกันกำหนดมาตรฐานสำหรับการสื่อสารอีเทอร์เน็ตที่ความเร็ว 10 เมกะบิตต่อวินาที (เมกะบิตต่อวินาที) ซึ่งเป็นพื้นฐานของมาตรฐานรหัส 802.3 ข้อแตกต่างที่สำคัญคือ มาตรฐาน IEEE 802.3 ได้กำหนดไว้สำหรับการสื่อสารแบบ 1 –persistent CSMA/CD ทำงานที่ความเร็ว 1 – 10 เมกะบิตต่อวินาที บนสายสื่อสารชนิดต่าง ๆ แรกทีเดียวได้กำหนดค่าตัวแปรไว้สำหรับการสื่อสารที่ความเร็ว 10 เมกะบิตต่อวินาที บนสายโคแอ็กเชียล (Coaxial) ขนาด 50 โอห์มเท่านั้น ค่าตัวแปรสำหรับตัวเลือกอื่น ๆ ได้รับการกำหนดเพิ่มเติมในภายหลัง

คนส่วนใหญ่มีความเข้าใจสับสนอย่างไม่ถูกต้องนักว่า ระบบอีเทอร์เน็ตหมายถึงระบบที่ใช้โพรโตคอล CSMA/CD ทุกชนิด ในความเป็นจริงนั้น ระบบอีเทอร์เน็ตเป็นระบบที่มีข้อกำหนดมาตรฐาน 802.3 เพียงอย่างเดียว

รูปแบบเฟรมของอีเทอร์เน็ตและ 802.3



รูปที่ 3-2 แสดงรูปแบบเฟรม (Frame Format) ของอีเทอร์เน็ต

สายสื่อสารที่ใช้ตามมาตรฐาน 802.3

คำว่า "Ethernet" นั้นอันที่จริงมีความหมายเกี่ยวข้องโดยตรงกับสายสื่อสารจึงขอเริ่มกล่าวถึงรายละเอียดของระบบนี้ที่สายสื่อสาร สายสื่อสารที่เป็นที่นิยมมีอยู่ 4 ชนิด สายประเภท 10Base5 ได้รับการพัฒนาขึ้นมาใช้งานเป็นประเภทแรกซึ่งนิยมเรียกว่า สายอีเทอร์เน็ตแบบหนา (Thick Ethernet) สายนี้มีลักษณะภายนอกเหมือนกับท่อสายยางรดน้ำต้นไม้ สายจะมีสีเหลือง และจะมีแถบสีใช้บอกระยะ ทุก ๆ 2.5 เมตร สำหรับการเชื่อมต่อไปยังเครื่องคอมพิวเตอร์ของผู้ใช้ วิธีการเชื่อมต่อก็ใช้การต่อแบบง่าย ๆ คือใช้ขั้วต่อแบบที่มีเข็มปักลงไปสายทองแดงที่อยู่ข้างใน คำว่า 10Base5 เป็นรหัสที่มีความหมายว่า สายชนิดนี้ใช้ในการสื่อสารที่ความเร็ว 10 เมกะบิตต่อวินาที ใช้การส่งสัญญาณแบบเบสแบนด์ (Baseband signaling) และมีความยาวในแต่ละเซกเมนต์ไม่เกิน 500 เมตร

ต่อมาได้มีการพัฒนาสายชนิด 10Base2 หรือสายอีเทอร์เน็ตแบบบาง (Thin Ethernet) ซึ่งมีขนาดเล็กกว่าและสามารถดัดให้โค้งงอได้มากกว่าแบบแรก เนื่องจากค่าใช้จ่ายที่ต่ำกว่า 10Base5 ทำให้สายชนิดนี้ได้รับความนิยมอย่างกว้างขวาง ข้อจำกัดที่สำคัญคือความยาวในแต่ละส่วนลดลงเหลือเพียง 200 เมตรและสามารถเชื่อมต่อกับเครื่องผู้ใช้ได้เพียง 30 เครื่องต่อ 1 เซกเมนต์เท่านั้น

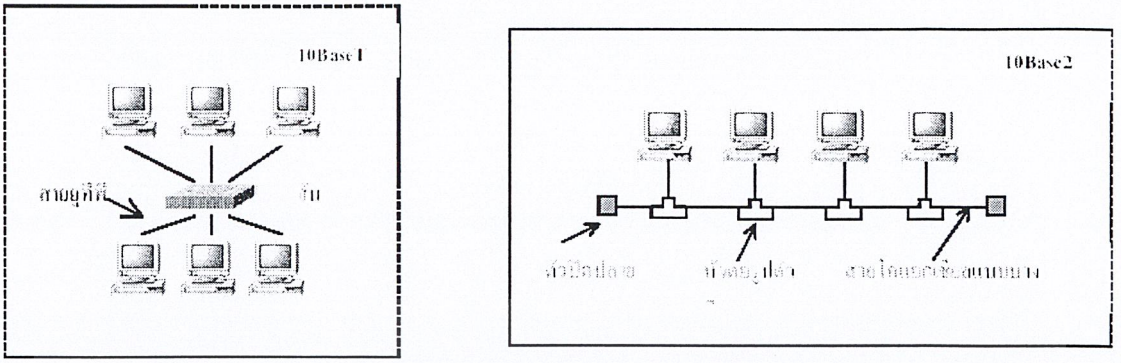
สายทั้งสองแบบนี้มีปัญหาในเรื่องการเชื่อมต่อที่ไม่สมบูรณ์ เช่น สายชำรุดหรือจุดเชื่อมต่อไม่สนิทซึ่งจะทำให้การสื่อสารของทั้งระบบเครือข่ายต้องล่มไป และการตรวจสอบหาจุดบกพร่องก็ทำได้ยาก

ปัญหาเหล่านี้ ได้ผลักดันให้เกิดการพัฒนาการใช้สายแบบใหม่โดยมีอุปกรณ์เรียกว่า ฮับ (Hub) เป็นศูนย์กลางการสื่อสารของเครือข่ายสื่อสารเฉพาะที่หนึ่ง ๆ เครื่องคอมพิวเตอร์ของผู้ใช้จะเชื่อมต่อเข้ากับฮับโดยตรง โดยระบบใหม่นี้ได้กำหนดให้ใช้สายโทรศัพท์เป็นสายสื่อสารโดยตรง ซึ่งมีชื่อว่า 10BaseT

ส่วนมาตรฐานอีเทอร์เน็ตความเร็ว 100 เมกะบิตต่อวินาทีที่นิยมใช้ในปัจจุบัน ได้แก่ 100BaseTX และ 100BaseFX สำหรับอีเทอร์เน็ตคงวามเร็วสูงแบบกิกะบิตอีเทอร์เน็ตเริ่มแพร่หลายมากขึ้น ตัวอย่างของมาตรฐานกิกะบิตอีเทอร์เน็ตในปัจจุบันได้แก่ 1000BaseT, 1000BaseLX และ 1000BaseSX เป็นต้น

Name	Cable	Max. Segment	Nodes/Segment
10Base5	Thick coax	500 m	100
10Base2	Thin coax	185 m	30
10BaseT	Twisted pair	100 m	1024
10BaseFP	Fiber optic	1 km	33
10BaseFL	Fiber optic	2 km	1
10BaseFB	Fiber optic	2 km	1
10Broad36	Coax	3.6 km	?

รูปที่ 3-3 แสดงประเภทของสายสื่อสารในมาตรฐาน 802.3



รูปที่ 3-4 แสดงตัวอย่างของสายสื่อสารประเภทต่าง ๆ

3.2 IEEE 802.4 (โทเก้นบัส)

แม้ว่าระบบ 802.3 จะได้รับความนิยมในการนำไปใช้ระดับสำนักงานเป็นอย่างมาก แต่การใช้งานตามบริษัทขนาดใหญ่ ยังเกิดการประสบปัญหา เช่น ในเรื่องระยะเวลาการรอคอยเมื่อเกิดสัญญาณซ้อน ซึ่งในบางโอกาสบางสถานีจะต้องรอส่งสัญญาณเป็นเวลานานมาก (ตามทฤษฎีแล้ว ระยะเวลาการรอคอยจะไม่มีขอบเขตแต่อย่างใด) อีกปัญหาหนึ่งคือ ระบบ 802.3 ไม่มีการกำหนดลำดับความสำคัญของเฟรม ทำให้ไม่สามารถนำไปใช้ในระบบเรียลไทม์ (real-time system) ซึ่งเฟรมที่มีความสำคัญมากจะต้องได้รับบริการก่อนเฟรมที่มีความสำคัญน้อยกว่า

ระบบเครือข่ายแบบวงแหวน เปิดโอกาสให้สถานีต่าง ๆ ผลัดเปลี่ยนหมุนเวียนกันส่งเฟรมข้อมูลของตนเอง โดยสามารถคำนวณระยะเวลาการรอคอยสูงสุดได้ล่วงหน้า แต่เครือข่ายแบบนี้มีข้อเสียในด้านลักษณะการเชื่อมต่อทางกายภาพเพราะการเสียหายของเคเบิ้ลที่จุดใดก็ตามจะทำให้ระบบทั้งระบบใช้การไม่ได้ ผลที่ได้รับคือ การพัฒนามาตรฐานใหม่เรียกว่า 802.4 ที่นำจุดเด่นของระบบ 802.3 มารวมเข้ากับความสามารถในการคำนวณระยะเวลาการรอคอยสูงสุดได้ล่วงหน้าของระบบเครือข่ายวงแหวน

มาตรฐาน 802.4 (Drivin and Miller, 1986 และ IEEE 1985b) มีชื่อเรียกว่า "โทเก้นบัส (Token Bus)" โดยลักษณะทางกายภาพแล้ว โทเก้นบัสมีการต่อเชื่อมแบบเป็นเส้นตรง ซึ่งสถานีต่าง ๆ สามารถเชื่อมต่อเข้าที่จุดใดก็ได้ แต่ในทางตรรกภาพที่ใช้ในการสื่อสารภายในนั้น สถานีจะถูกจัดกลุ่มในลักษณะของวง (ring) ซึ่งแต่ละสถานีจะทราบหมายเลขที่อยู่ของสถานีที่อยู่ข้างเคียงของตนเองตลอดเวลา เมื่อเริ่มต้นการทำงานสถานีที่มีหมายเลขสูงสุดในวงนั้น ๆ สามารถส่งเฟรมข้อมูลออกมาได้เป็นอันดับแรก หลังจากนั้นก็จะส่งเฟรมข้อมูลพิเศษเรียกว่า "โทเก้น (token)" ไปยังสถานีข้างเคียงของตนเอง เป็นการอนุญาตให้สถานีนั้นส่งเฟรมออกมาได้ โทเก้นจะถูกส่งต่อไปเรื่อย ๆ ตามลำดับจนครบวงรอบ คือย้อนกลับมาที่สถานีแรกแล้วจึงวนต่อไป เนื่องจากในเวลาใดก็ตามจะมีเพียงสถานีเดียวเท่านั้นที่ครอบครองโทเก้นอยู่ ทำให้การส่งสัญญาณซ้อนจะไม่เกิดขึ้นอย่างแน่นอน

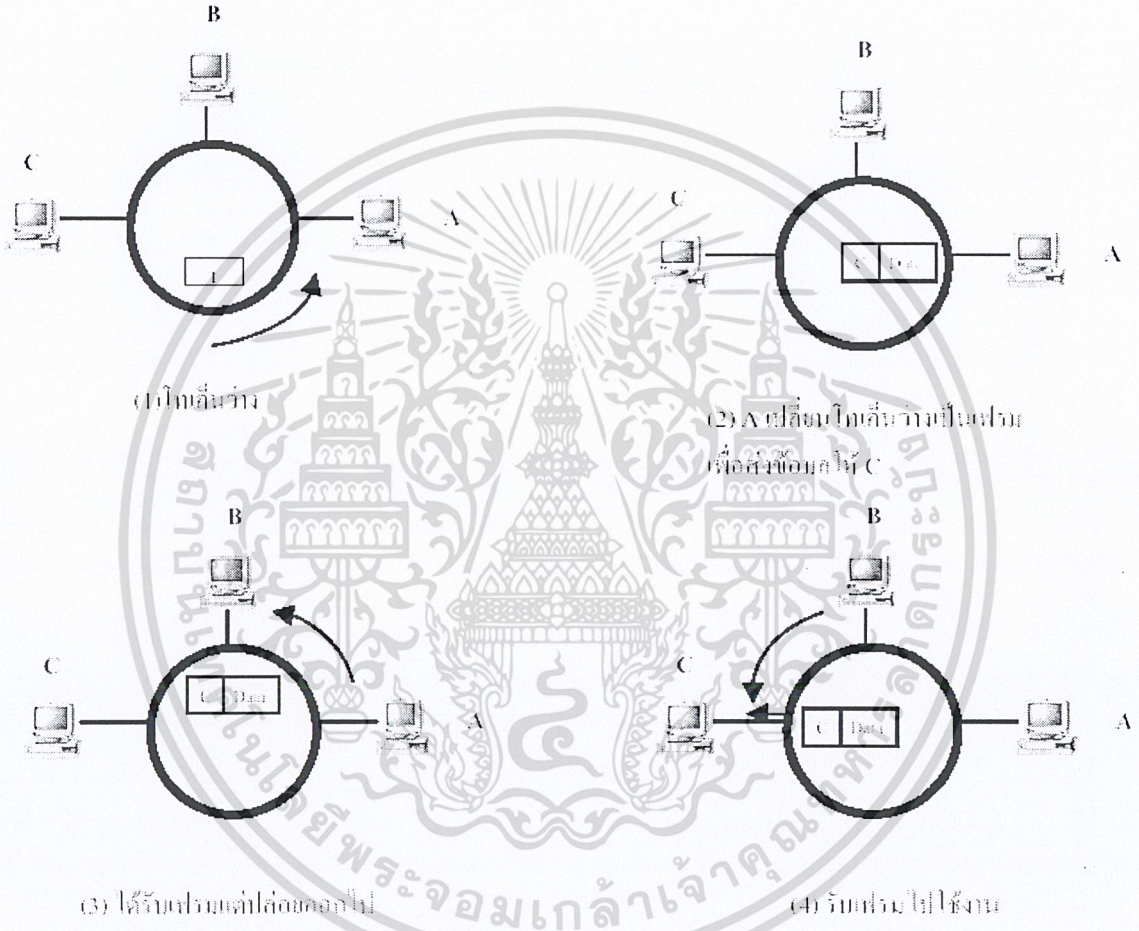
สิ่งที่น่าสนใจประการหนึ่งคือ ลำดับที่อยู่ทางกายภาพนั้นไม่มีความสำคัญแต่อย่างใด เนื่องจากสายสื่อสารมีการส่งสัญญาณแบบกระจายข่าว (broadcast) ทุกสถานีจึงได้รับเฟรมข้อมูลทุกเฟรมเหมือนกันหมดแต่จะอ่านข้อมูลจากเฟรมที่ส่งมาถึงตนเองเท่านั้น เมื่อได้รับโทเก้น แต่ละสถานีจะส่งโทเก้นไปยังสถานีในลำดับต่อไปทางตรรกภาพของตนเอง ซึ่งอาจเป็นสถานีข้างเคียงหรือไกลออกไป (ทางกายภาพ) ก็ได้

สายสื่อสารในโทเก้นบัส ส่วนใหญ่จะใช้สายเคเบิ้ลโคแอ็กเชียล (broadband coaxial) ขนาด 75 โอห์ม ซึ่งเป็นสายแบบเดียวกับที่ใช้กับเคเบิ้ลทีวี สายที่ใช้อาจเป็นแบบสายเดี่ยวหรือสายคู่และจะมีเฮดเอนด์ (head-end) หรือไม่ก็ได้ ระบบการเปลี่ยนสัญญาณเป็นแบบอะนาล็อก ความเร็วในการส่งข้อมูลมีตั้งแต่ 1, 5 และ 10 เมกะบิตต่อวินาที โดยภาพรวมแล้วจะเห็นได้ว่ามาตรฐาน 802.4 นั้นมีความแตกต่างกับมาตรฐาน 802.3 อย่างสิ้นเชิงและมีความซับซ้อนกว่ามากทีเดียว

3.3 IEEE 802.5 (โทเก้นริง)

ระบบเครือข่ายแบบโทเก้นริงได้รับการพัฒนาขึ้นมาใช้งานทั้งในระบบเครือข่ายแบบแลนและระบบเครือข่ายแบบแวนมาเป็นระยะเวลานานพอสมควรแล้ว (Pierce, 1972) โครงสร้างแบบวงแหวนนั้น ที่จริงก็คือการเชื่อมต่อแบบจุด-ต่อ-จุดกลุ่มหนึ่งที่มีการเรียงลำดับเป็นรูปวงกลมพอดี การเชื่อมต่อแบบจุด-ต่อ-จุดนั้นใช้เทคโนโลยีที่ได้ผ่านการปรับปรุงมาจนอยู่ในระดับที่สามารถไว้วางใจได้ ซึ่งสายสื่อสารอาจเป็นแบบธรรมดา เช่น สายคู่ตีเกลียว สายโคแอ็กเชียลหรือสายใยแก้วก็ได้ อัตราการส่งข้อมูลของโทเก้นริงที่ใช้โดยทั่วไปคือ 4 และ 6 เมกะบิตต่อวินาที สัญญาณที่ใช้ในระบบนี้อาจเป็นแบบดิจิตอลก็ได้ ในขณะที่ระบบเครือข่ายแบบ 802.3 นั้นมีอุปกรณ์และโพรโตคอลหลายอย่างที่มีลักษณะเป็นแบบอะนาล็อก ยิ่งไปกว่านั้น ระบบเครือข่ายแบบโทเก้นริงยังสามารถคำนวณระยะเวลารอคอยที่ค่อนข้างคงที่ได้ ด้วยเหตุผลเหล่านี้ บริษัท IBM จึงเลือกระบบเครือข่ายนี้มาใช้ในองค์กร ส่วน IEEE ก็ได้ออกมาตรฐานรองรับโดยกำหนดรหัสหมายเลขเป็น 802.5 (IEEE, 1985c; Latif et al., 1992)

รูปที่ 3-5 แสดงการทำงานของโทเก้นริง โดยมีเฟรมพิเศษเรียกว่า โทเก้นว่าง (Free Token) วิ่งวนไปเรื่อย ๆ สเตชันที่ต้องการส่งข้อมูลจะรอให้โทเก้นว่างเดินทางมาถึงแล้วรับโทเก้นว่างนั้นมาเปลี่ยนเป็นเฟรมข้อมูล (Data Frame) โดยใส่แฟรก (Flag) แสดงเฟรมข้อมูลและบรรจุแอดเดรส (Address) ของสเตชันต้นทางและปลายทางตลอดจนข้อมูลอื่น ๆ จากนั้นจึงปล่อยเฟรมนี้ออกไป



รูปที่ 3-5 แสดงลำดับการทำงานของโทเก้นริง

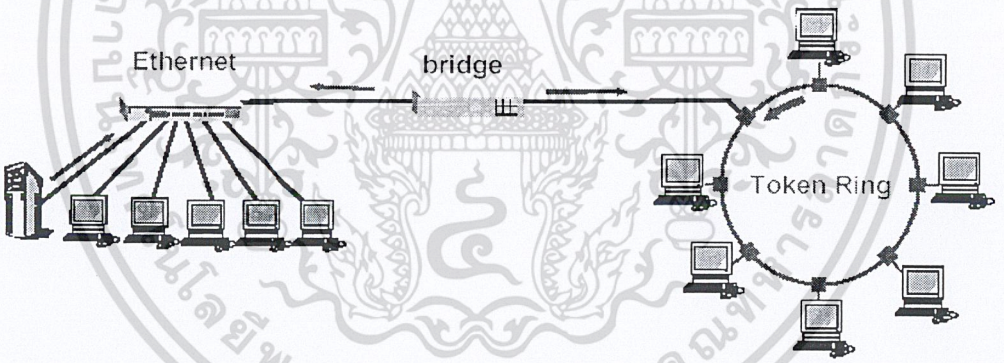
เมื่อสเตชันปลายทางได้รับเฟรม จะสำเนาข้อมูลไว้และปล่อยเฟรมให้วนกลับมายังสเตชันต้นทาง สเตชันต้นทางจะตรวจสอบเฟรมและปล่อยโทเก้นว่างคืนสู่เครือข่ายให้สเตชันอื่นมีโอกาสส่งข้อมูลต่อ กลไกแบบส่งผ่านโทเก้นจัดอยู่ในประเภทประเมินเวลาได้ กล่าวคือสามารถคำนวณเวลาสูงสุดที่สเตชันมีสิทธิ์จัดโทเก้นเพื่อส่งข้อมูลได้ โทเก้นริงจึงเหมาะกับระบบที่ต้องการความแน่นอนทางเวลาหรืองานแบบเวลาจริง

บทที่ 4
พื้นฐานบริดจ์และสวิตช์

4.1 บริดจ์ (Bridge)

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



รูปที่ 4-1 แสดงระบบเครือข่ายที่ใช้บริดจ์ในการเชื่อมต่อ

4.1.1 ชนิดของบริดจ์

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

ทรานส์แพเร้นท์บริดจ์ (Transparent Bridge)

พัฒนาโดย Digital Equipment Corporation ในต้นปี 1980 ซึ่งต่อมาได้มีการกำหนดมาตรฐาน IEEE 802.1 โดยผู้ใช้งานได้กำหนดความต้องการทรานส์แพเร้นท์บริดจ์ให้มีลักษณะดังนี้ แรกทีเดียวผู้ใช้ (ทุกระบบ) จะต้องไม่มีส่วนเกี่ยวข้องกับการทำงานของทรานส์แพเร้นท์บริดจ์ การมีอุปกรณ์ประเภทนี้อยู่ในระบบ หรือไม่มีอยู่ก็ตาม จะต้องไม่มีผลกระทบใด ๆ ต่อผู้ใช้งาน ผู้ใช้ทั่วไปจะต้องสามารถซื้ออุปกรณ์นี้มาจากตัวแทน

จำหน่ายของบริษัทใดก็ได้ การติดตั้งจะต้องมีความยุ่งยากเพียงแค่การเสียบปลั๊กไฟฟ้าและเสียบสายระบบเครือข่ายต่าง ๆ เข้ากับอุปกรณ์นี้ แม้ว่าจะไม่มีการกำหนดค่าพารามิเตอร์ใด ๆ ตัวอุปกรณ์ก็จะต้องสามารถทำงานได้ในทันที ผลที่เกิดขึ้นนั้นน่าแปลกใจเป็นอย่างยิ่งว่า ทรานส์แพเร้นท์บริดจ์นั้นมีตัวตนและทำงานได้อย่างที่ผู้ใช้ต้องการ

ทรานส์แพเร้นท์บริดจ์ ส่วนใหญ่จะใช้ในการเชื่อมต่อระหว่างเซกเมนต์ (Segment) ของอีเทอร์เน็ต โดยการทำงานของทรานส์แพเร้นท์บริดจ์ จะเก็บข้อมูลแมคแอดเดรสของสเตชันที่ต่ออยู่ของพอร์ตต่าง ๆ โดยจะใช้ข้อมูลนั้น เมื่อมีเฟรมส่งเข้ามาที่พอร์ตนั้น

MAC Address Table

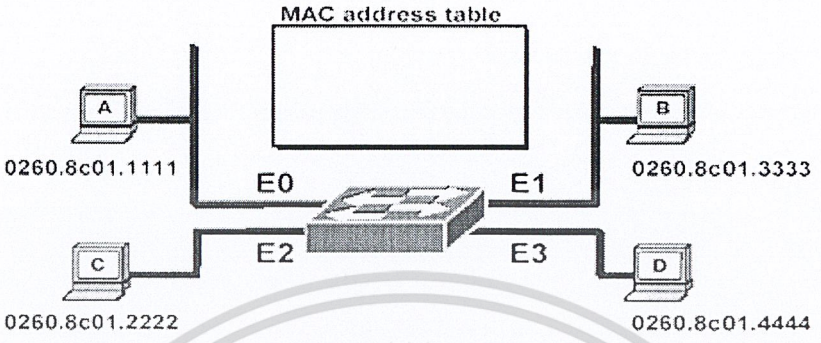
E0: 0260.8c01.1111
E2: 0260.8c01.2222
E1: 0260.8c01.3333
E3: 0260.8c01.4444

รูปที่ 4-2 แสดงตัวอย่าง MAC Address Table

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

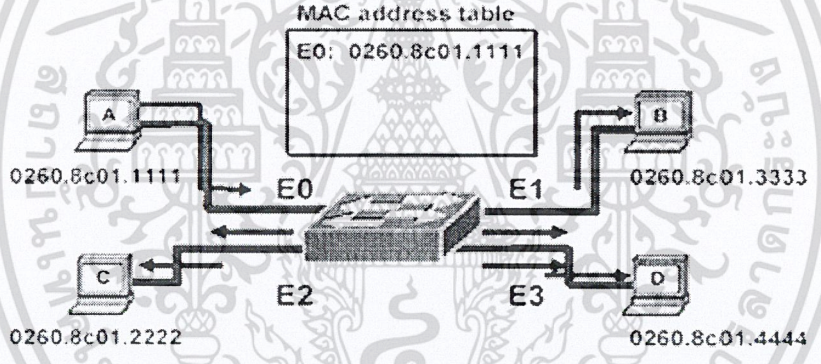
ทรานส์แพเร้นท์บริดจ์จะช่วยให้สามารถแบ่งระบบเครือข่ายออกเป็นเซกเมนต์ย่อย ๆ เพื่อลดปริมาณของการส่งข้อมูลในรูปแบบอีเทอร์เน็ตในอุปกรณ์ฮับไม่ให้คับคั่งมากเกินไป

ตัวอย่างการส่งข้อมูลและการสร้างตารางแมคแอดเดรส



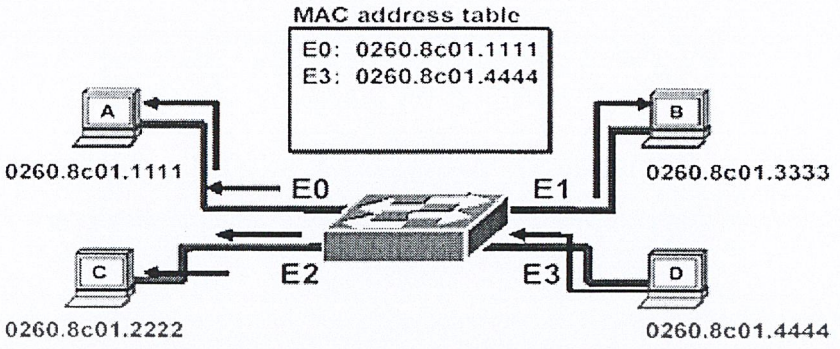
รูปที่ 4-3 แสดงการเรียนรู้แมคแอดเดรส

จากรูป เป็นตอนเริ่มต้น ยังไม่มีการส่งข้อมูล ดังนั้น ในตารางแมคแอดเดรสจึงยังว่างเปล่าอยู่ (empty)



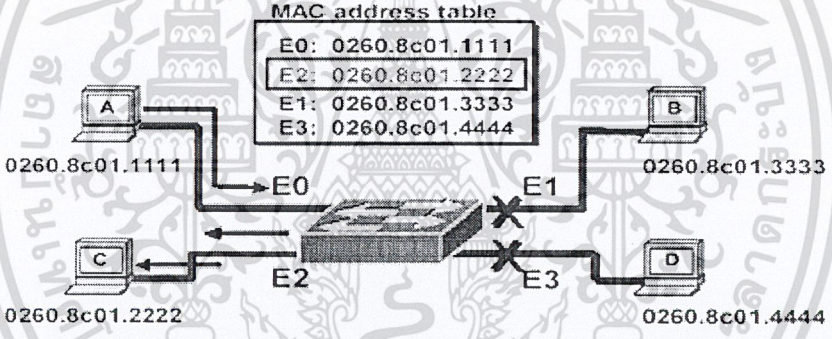
รูปที่ 4-4 แสดงการเรียนรู้แมคแอดเดรส

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



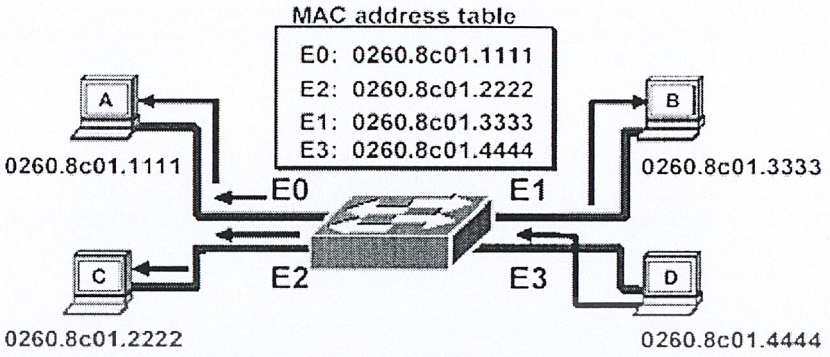
รูปที่ 4-5 แสดงการเรียนรู้แมคแอดเดรส

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



รูปที่ 4-6 แสดงการทำงานของสวิตช์ในการทำฟิลเตอร์ (Filter)

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



รูปที่ 4-7 แสดงการเรียนรู้แมกแอดเดรส

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

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) ในเฟรมเอกโพลเรอร์ตามบริดจ์นั้น เมื่อเฟรมเอกโพลเรอร์ไปถึงโฮสต์วายจะทำการตอบกลับมายังโฮสต์เอ็กซ์ตามข้อมูลเส้นทาง ซึ่งอาจจะตอบกลับมาหลายครั้งตามจำนวนเฟรมเอกโพลเรอร์ที่ได้รับ

ซึ่งโฮสต์เอ็กซ์จะต้องเลือกเส้นทางใดเส้นทางหนึ่ง โดยที่วิธีในการเลือกนั้นไม่ได้กำหนดไว้ใน IEEE 802.5 แต่สามารถเป็นไปได้หลายอย่าง เช่น

- เลือกเฟรมตอบกลับแรกที่ได้รับ
- เลือกเฟรมที่มีฮอป (Hop) น้อยที่สุด
- เลือกเส้นทางที่ยอมให้มีเฟรมขนาดใหญ่ที่สุด

เป็นต้น

เมื่อเลือกเส้นทางได้แล้ว ข้อมูลเส้นทางจะถูกกำหนดลงในเฟรมที่จะส่งไปที่โฮสต์วายในรูปแบบ Routing Information Field (RIF)

4.1.2 การเปรียบเทียบบริดจ์ในระบบ 802

บริดจ์ทั้งแบบทรานส์แพเร้นท์บริดจ์และแบบ SRB มีทั้งข้อดีและข้อเสียที่แตกต่างกันดังที่สรุปไว้ในตารางในรูปที่ 4-8

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

รูปที่ 4-8 ตารางเปรียบเทียบคุณสมบัติของทรานส์แพเร้นท์บริดจ์และแบบ SRB

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

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

การใช้ทรานส์แพเร้นท์บริดจ์ไม่จำเป็นต้องมีระบบบริหารเครือข่าย บริดจ์สามารถเรียนรู้ได้ด้วยตัวเองและสามารถปรับตัวให้เข้ากับระบบเครือข่ายนั้น ๆ ได้โดยอัตโนมัติ ส่วน SRB จะต้องให้ผู้บริหารเครือข่ายทำการติดตั้งหมายเลขระบบเครือข่ายและหมายเขบริดจ์ด้วยตนเองทั้งหมด 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 การหาคำตอบด้วการส่งเฟรมค้นหาข้อมูลออกไปใหม่จะทำให้ทราบว่าสเตชันปลายทางยังคงทำงานอยู่หรือไม่ อย่างไรก็ตาม ในกรณีที่บริดจ์หลักเสียหายหรือหยุดทำงานจะทำให้โฮสต์จำนวนมากเสียเวลาไปกับการรอคอยและส่งเฟรมค้นหาออกไปจนกว่าปัญหานั้นจะได้รับการแก้ไขแม้ว่าจะมีเส้นทางอื่นอยู่ก็ตาม การชำรุดของอุปกรณ์เป็นจุดอ่อนหลักของการเชื่อมต่อแบบต่อเนื่องทั้งหมด

4.2 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

4.2.1 คัททรูสวิตช์ (Cut-Through Switching)

โดยการทำงานปกติของสวิตช์ จะทำการรับเฟรมเข้ามาก่อน แล้วจึงส่งเฟรมนั้นไปยังพอร์ตของสเตชันปลายทาง (Store & Forward) สวิตช์แบบคัททรูจะลดการหน่วงเวลาในขั้นตอนนี้ โดยเมื่อสวิตช์ได้รับข้อมูลเฟรมเพียงพอที่จะกำหนดสเตชั่นเป้าหมายได้แล้ว ก็จะเริ่มต้นการส่งข้อมูลทันทีโดยไม่ต้องรอให้ได้รับเฟรมทั้งหมด

การใช้งานสวิตช์แบบคัททรูอาจจะทำให้เกิดปัญหาการส่งเฟรมที่มีข้อผิดพลาดได้ ดังนั้นจึงควรกำหนดให้สวิตช์ทำการรับเฟรมมาจำนวนหนึ่งก่อน แล้วจึงเริ่มการส่งเฟรม เพื่อให้แน่ใจว่า เฟรมนั้นเป็นเฟรมที่ไม่มีข้อผิดพลาด

4.2.2 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

4.2.3 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

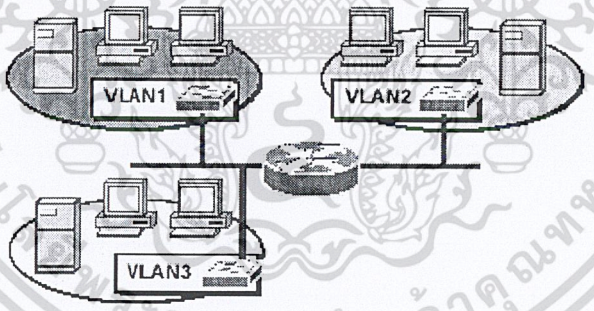
ASIC – Application Specific Integrated Circuit

รูปที่ 4-9 แสดงตารางการเปรียบเทียบระหว่างสวิตช์ชั้นที่ 2 กับสวิตช์ชั้นที่ 3

บทที่ 5
แลนเสมือน

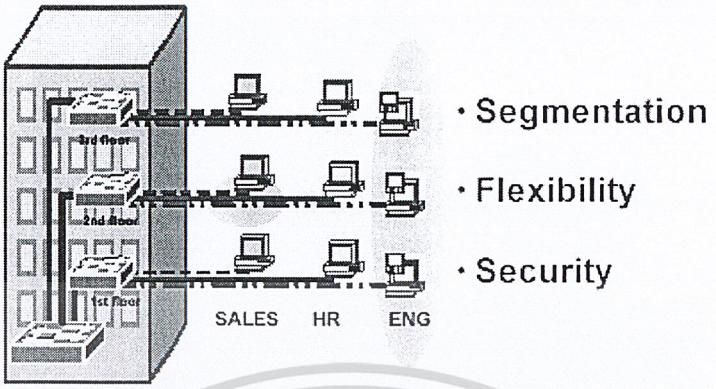
เมื่อพิจารณาถึงระบบเครือข่ายที่ประกอบด้วยอุปกรณ์ในชั้นที่ 2 เท่านั้น เช่น เซกเมนต์ของอีเทอร์เน็ต, สวิตช์ที่มีหลายพอร์ต หรือเครือข่ายที่ประกอบไปด้วยสวิตช์หลาย ๆ ตัว เครือข่ายแบบนี้ เรียกว่า Flat Network Topology เครือข่ายแบบนี้จะมีได้เพียง 1 บอร์ดคาสต์โดเมน (Broadcast Domain) เท่านั้น หมายความว่า เมื่อมีการส่งเฟรมแบบบอร์ดคาสต์จะทำให้ทุก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

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



รูปที่ 5-1 แสดงเครือข่ายวีแลน

เมื่อกำหนดวีแลนแล้ว สเตชันในวีแลนเดียวกันจะสามารถส่งข้อมูลถึงกันได้ แต่ถ้าเป็นการส่งข้อมูลข้ามเซกเมนต์จะต้องใช้เราเตอร์ในการส่งผ่านข้อมูล ซึ่งผู้ดูแลระบบสามารถกำหนดรายละเอียดของการส่งผ่านข้อมูลระหว่างเซกเมนต์ได้



รูปที่ 5-2 แสดงตัวอย่างการแบ่งวีแลนออกเป็นแผนกงาน

วีแลนมีข้อดีคือ

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

5.1 ประเภทของวีแลน

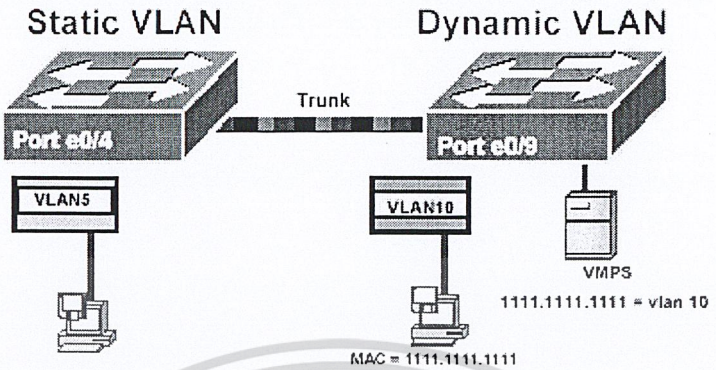
แบ่งออกเป็น 2 ประเภทใหญ่ ๆ คือ

5.1.1 สแตติกวีแลน (Static VLANs)

เป็นการกำหนดวีแลนจากพอร์ตของสวิตช์ว่า ต้องการให้พอร์ตไหนเป็นวีแลนใด เมื่อเรานำอุปกรณ์ไปต่อ ก็จะทำให้อุปกรณ์ชิ้นนั้นเป็นสมาชิกของวีแลนนั้น โดยอัตโนมัติ 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

1. **Port-Based & MAC-Based**

   Port-Based : กำหนดวีแลนตามพอร์ตที่กำหนดไว้ (เหมือน Static VLAN)

   MAC-Based : กำหนดวีแลนตามแมคแอดเดรสที่กำหนดไว้ (เหมือน Dynamic VLAN)

2. **Protocol-Based & Dynamic-Based**

   Protocol-Based : กำหนด วีแลน ตาม โพรโตคอลที่กำหนดไว้ เช่น

   Host X ใช้โพรโตคอล IP ดังนั้น จะเป็นสมาชิกของวีแลน 1

   Host Y ใช้โพรโตคอล IPX ดังนั้น จะเป็นสมาชิกของวีแลน 2

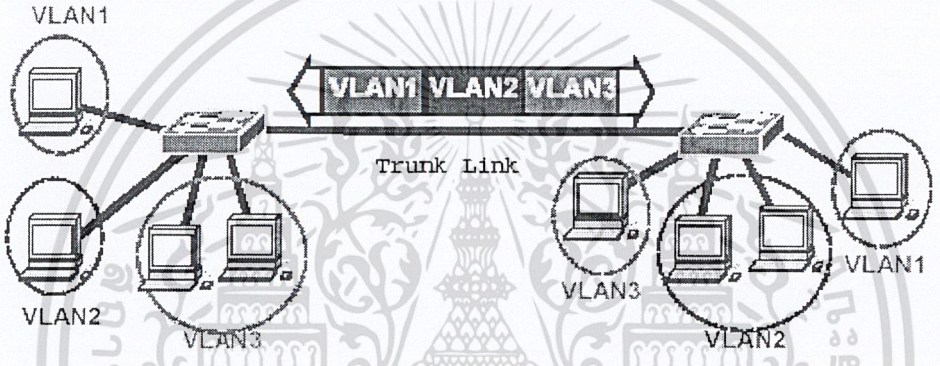
   Dynamic-Based : กำหนดวีแลนตาม User Profile ที่กำหนดไว้ โดยเก็บ User Profile ไว้ในฐานข้อมูล เช่น โฮสต์ X ทำการ Log in ตัวโพรไฟล์ (Profile) ของ โฮสต์ X จะเป็นตัวกำหนดให้โฮสต์ X เป็นของ วีแลน 1

5.2 ประเภทของการเชื่อมต่อ

ในการกำหนดวีแลนนั้น บางครั้งอาจมีการกำหนดให้ในวีแลนเดียวกันมีอุปกรณ์ที่เป็นสมาชิกอยู่ในสวิตช์คนละตัวกัน ดังนั้นจึงต้องมีการกำหนดการเชื่อมต่อของวีแลนเพื่อใช้เป็นกฎในการส่งข้อมูลภายในวีแลนเดียวกัน

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

5.2.2 ทรังค์ลิงค์ (Trunk Link) เป็นการเชื่อมต่อที่ใช้ในการส่งข้อมูลได้หลายๆ วีแลน

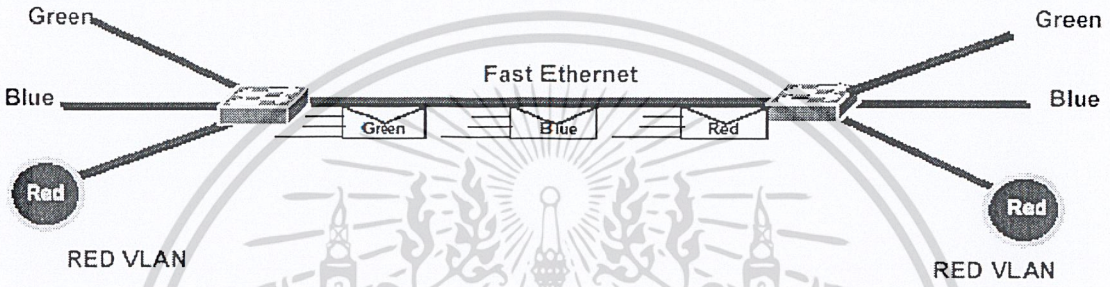


รูปที่ 5-4 แสดงตัวอย่างของทรังค์ลิงค์

5.3 วิธีการระบุถึงวีแลน

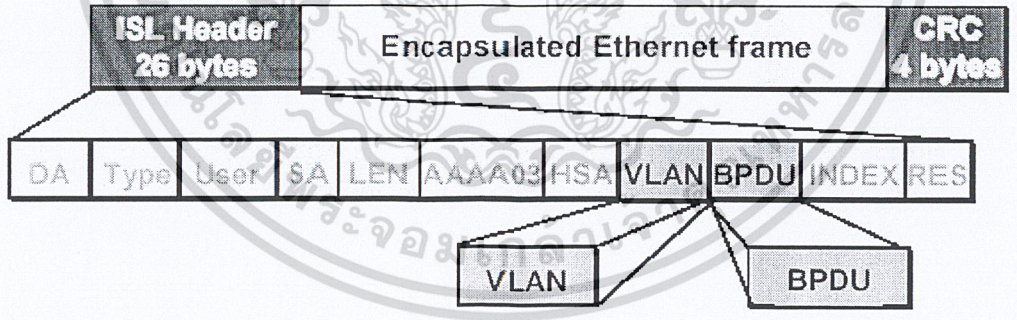
แพ็กเก็ตจะถูกส่งไปตาม Trunk Link โดยบรรจุข้อมูลที่ระบุถึงวีแลนไว้ในส่วนของเฮดเดอร์ (Header) ของแพ็กเก็ต ซึ่งวิธีการระบุนั้นมีอยู่ 2 แบบคือ

- Cisco ISL
- IEEE 802.1Q



รูปที่ 5-5 แสดงเครือข่ายของวีแลนที่ใช้การเชื่อมต่อแบบทรังค์ลิงค์

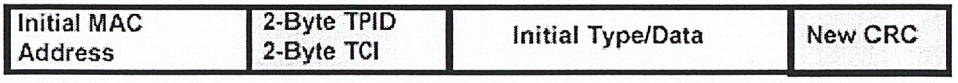
Cisco ISL



รูปที่ 5-6 แสดงรูปแบบของเฟรมของ ISL

เมื่อได้รับเฟรมอีเทอร์เน็ตมาแล้ว จะทำการ encapsulate ด้วย ISL Header และ CRC ซึ่งสนับสนุนวีแลนได้มากสุดถึง 1024 วีแลน

IEEE 802.1Q

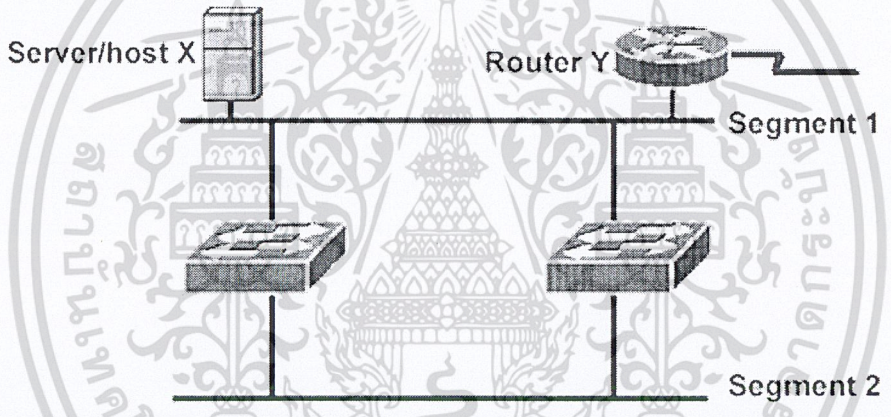


รูปที่ 5-7 แสดงรูปแบบของเฟรมของ IEEE 802.1Q

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

บทที่ 6
สแปนนิ่งทรีโพรโตคอล

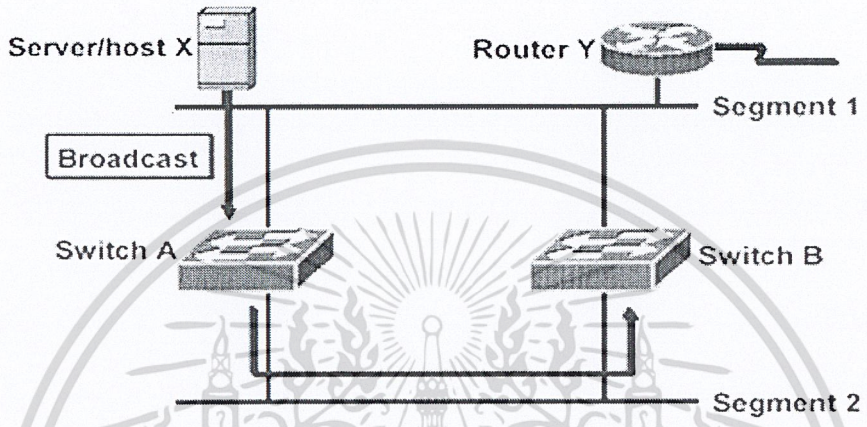
ในการสร้างระบบเครือข่ายที่ดีนั้น ควรต้องคำนึงถึงความน่าจะเป็นในทุกด้าน เมื่อเกิดกรณีฉุกเฉิน ระบบเครือข่ายจะทำการแก้ไขเช่นไร เช่น ถ้าในระบบ สวิตช์เกิดเสีย ใช้งานไม่ได้ จะทำเช่นไร ด้วยเหตุนี้ จึงทำให้เกิดเป็นแนวคิด Redundant Topology ขึ้น คือเป็นการสร้างระบบเครือข่ายแบบต่อให้มี สวิตช์ 2 ตัวในเส้นทาง (path) เดียวกัน เพื่อป้องกันในเหตุที่ถ้ามีสวิตช์ตัวใดตัวหนึ่งเสีย สวิตช์ตัวที่เหลือจะยังสามารถใช้งานต่อไปได้ ทำให้ระบบเครือข่ายไม่หยุดชะงัก



รูปที่ 6-1 แสดงตัวอย่างของเครือข่ายที่ใช้ Redundant Topology

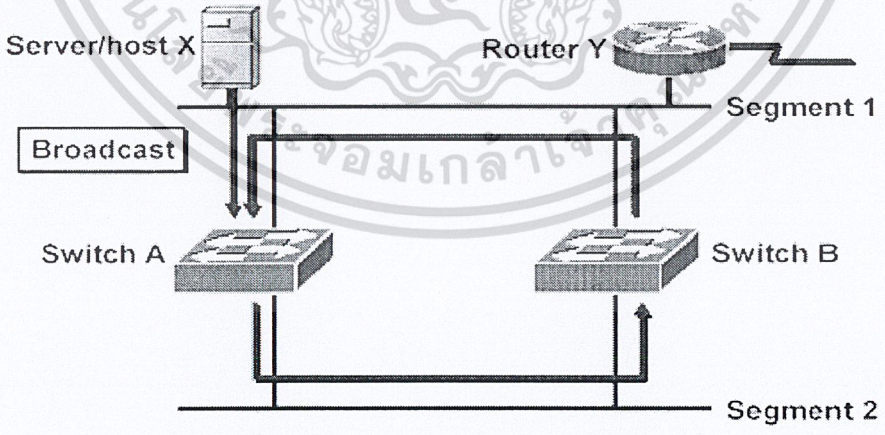
เมื่อเราสร้างระบบเครือข่ายโดยใช้ Redundant Topology แล้ว อาจทำให้เกิดลูป (loop) ขึ้นได้ ซึ่งเป็นสาเหตุทำให้เกิดบอร์ดคาสต์สตอร์มขึ้นในระบบเครือข่ายจนอาจเป็นสาเหตุทำให้เครือข่ายล่มได้

โดยขั้นตอนของการเกิดลูปจะเป็นดังนี้คือ



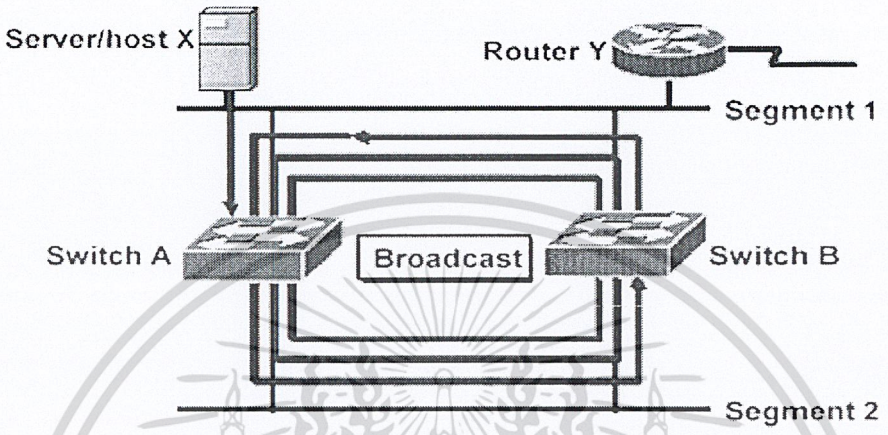
รูปที่ 6-2 แสดงการส่งข้อมูลจากโฮสต์ X ไปยังสวิตช์ A

เมื่อโฮสต์ X ส่งเฟรมข้อมูลออกไปให้กับสวิตช์ A จากนั้นสวิตช์ A จะทำการใส่แอดเดรสต้นทาง (Source Address) เข้าไปในตารางแมคแอดเดรสแล้วจึงทำการส่งออกไปยังทุกพอร์ตออกไป ทำให้สวิตช์ B ได้รับเฟรมข้อมูลนี้ด้วย



รูปที่ 6-3 แสดงการฟลัดเอาท์ (Flood out) ของสวิตช์ A

เมื่อสวิตช์ B ได้รับเฟรมข้อมูล ก็จะทำการใส่แอดเดรสต้นทางลงไปในตารางแมคแอดเดรสแล้วทำการส่งออกไปยังทุกพอร์ต (Flood Out) อีกครั้ง ทำให้สวิตช์ A ได้รับเฟรมข้อมูลนี้อีกครั้ง



รูปที่ 6-4 แสดงการเกิดบอร์ดคาสต์สตอร์ม (Broadcast Storm)

เมื่อสวิตช์ A ได้รับเฟรมข้อมูล ก็จะกระทำการเหมือนเดิม และต่อเนื่องไปถึงสวิตช์ B วนเป็นลูปไปเรื่อย ๆ ไม่รู้จบ และแม้ว่า เฟรมข้อมูลไปถึง ณ จุดหมายที่สเตชันที่ต้องการแล้ว ตัวสวิตช์ A และสวิตช์ B ก็จะยังส่งเฟรมข้อมูลกันต่อไปแบบไม่รู้จบ จนทำให้เกิดเป็นบอร์ดคาสต์สตอร์ม สร้างทราฟฟิกให้กับระบบเครือข่าย จนอาจทำให้เครือข่ายหยุดชะงักได้

6.1 สแปนนิ่งทรีโพรโตคอล (Spanning-Tree Protocol)

สแปนนิ่งโพรโตคอลถูกสร้างมาเพื่อใช้ในการกำจัดลูปที่เกิดขึ้นในระบบเครือข่าย โดยจะเสมือนจัดให้ระบบเครือข่ายมีลักษณะเหมือนกับต้นไม้ (Tree) คือจากจุดเริ่มต้นแล้วมีการแตกกิ่งก้านออกมาเรื่อย ๆ เป็นสายแห่งการติดต่อสื่อสาร จะทำให้เป็นเครือข่ายที่ไม่มีลูปเลย

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

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

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

6.1.1 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)

6.1.2 เลือกรูทพอร์ต

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

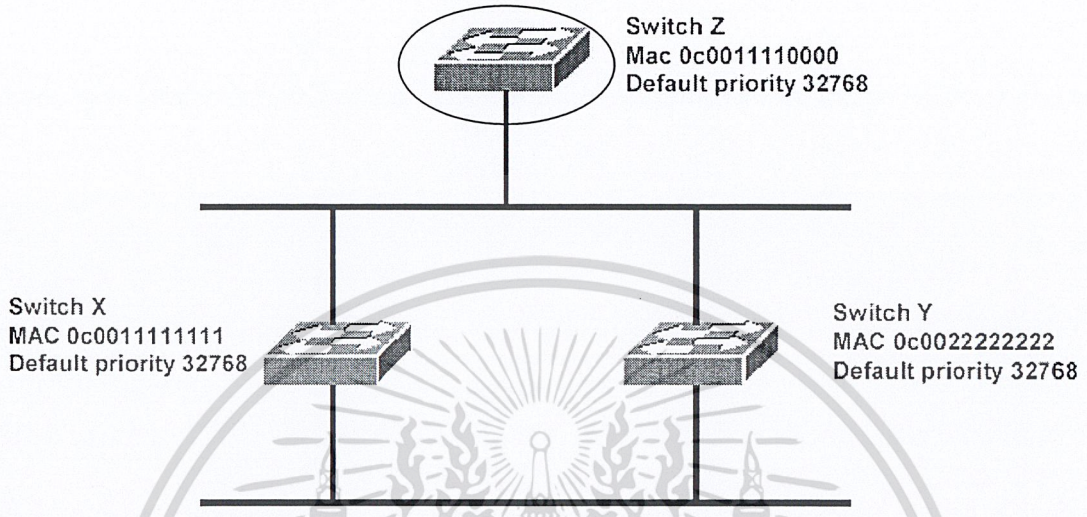
6.1.3 เลือกดีไซเนเตทพอร์ต

ดีไซเนเตทพอร์ตมีคอนเซ็ปต์ว่า ลิงค์เพียงหนึ่งเดียวในเซกเมนต์ที่ใช้ในการส่งและรับทราฟฟิก โดยสำหรับตัวรูทบริดจ์นั้นถือว่า ทุก ๆ พอร์ตของรูทบริดจ์จะเป็นดีไซเนเตทพอร์ตและสำหรับนอนรูทบริดจ์จะให้พอร์ตที่ต่อกับรูทพอร์ตของสวิตช์ตัวข้างเคียงเป็นดีไซเนเตทพอร์ต และสำหรับพอร์ตที่ไม่ใช่รูทพอร์ตและดีไซเนเตทพอร์ตจะถูกบล็อก (Block)

ด้วยหลักการทำงานนี้ จะทำให้แต่ละเครือข่ายแลนสามารถส่งเฟรมข้อมูลไปยังสวิตช์ได้เพียงเครื่องเดียว และเกิดสภาพของต้นไม้ (Tree) ขึ้น โดยพอร์ตที่ไม่ได้ใช้งานจะเป็นพอร์ตสำรอง และเมื่อดำเนินการทำจนเกิดเป็นสแปนนิ่งทรีแล้ว จะได้เครือข่ายที่มีลักษณะ

- **One Root Bridge per Network**

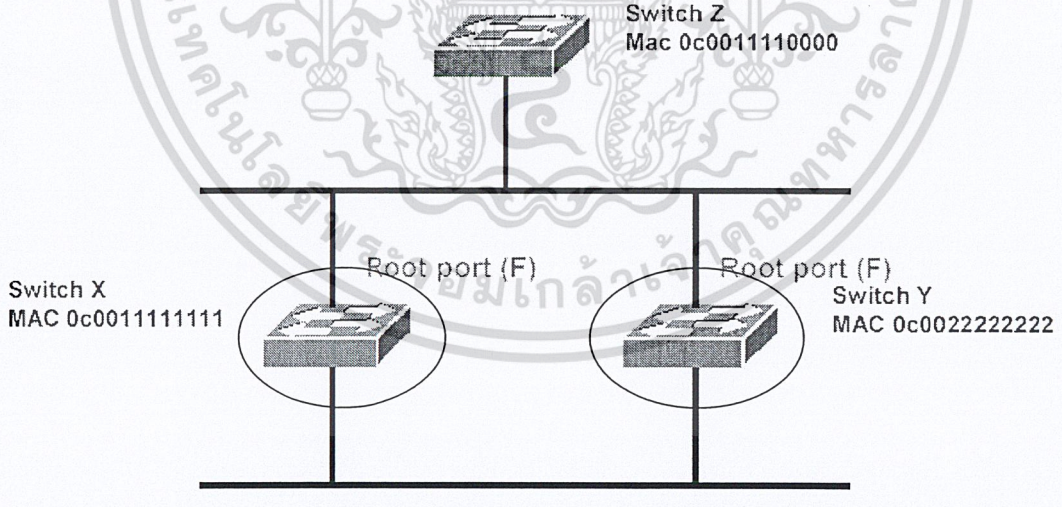
  1 Root Bridge ต่อ 1 ระบบเครือข่าย



รูปที่ 6-5 แสดงระบบเครือข่ายที่มีสวิตช์ Z เป็นรูทบริดจ์

- **One Root port per Non-Root Bridge**

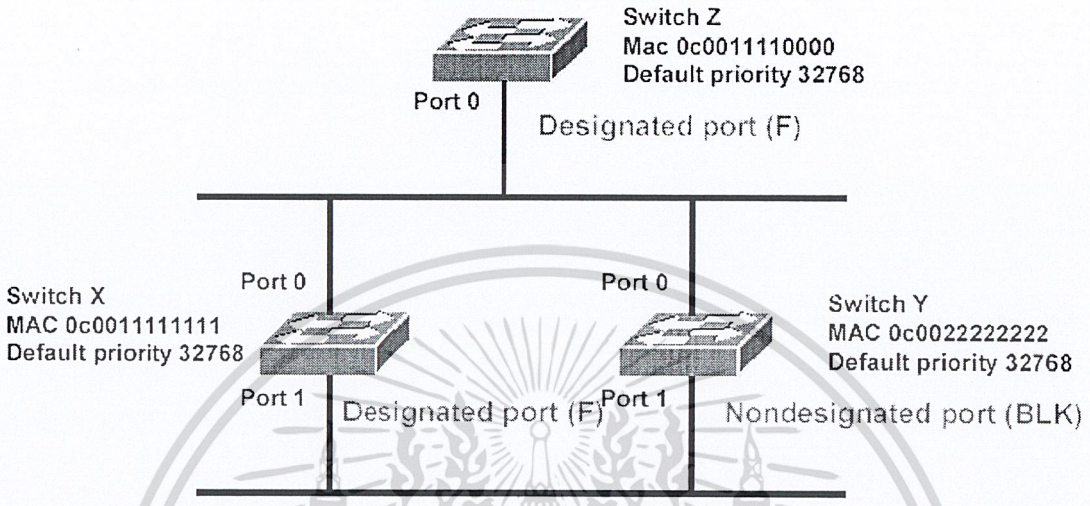
  1 Root Port ต่อ 1 Non-root Bridge



รูปที่ 6-6 แสดงระบบเครือข่ายที่มีสวิตช์ X และสวิตช์ Y เป็นนอนรูทบริดจ์

- **One Designated port per Segment**

  1 Designated port ต่อ 1 เซกเมนต์



รูปที่ 6-7 แสดงระบบเครือข่ายที่มี 1 ดีไซเนเตทพอร์ตต่อ 1 เซกเมนต์

6.2 สแปนนิ่งทรีสเตท (Spanning Tree Port States)

ในการใช้สแปนนิ่งโพรโตคอลทุก ๆ พอร์ตของสวิตช์ จะต้องทำงานเป็นขั้นตอนผ่านไปในแต่ละสเตทโดยจะเริ่มต้นที่ Disabled State และสิ้นสุดที่สเตทสุดท้ายซึ่งอนุญาตให้พอร์ตสามารถรับ-ส่งเฟรมข้อมูลได้ โดยสเตทมีดังต่อไปนี้

6.2.1 **Disabled** – เป็นสเตทของพอร์ตที่ชัทดาวน์ (Shut down) คือไม่สามารถทำอะไรได้เลย สเตทนี้ไม่ถือว่าเป็นส่วนหนึ่งของสแปนนิ่งทรีสเตท

6.2.2 **Blocking** – หลังจากการติดตั้งค่าให้กับพอร์ตแล้ว พอร์ตจะเข้าสู่ Blocking State เพื่อไม่ให้การส่งเฟรมข้อมูลแบบลูป โดยในสเตทนี้จะไม่สามารถรับ – ส่งข้อมูลได้ รวมถึงไม่มีการเรียนรู้แมคแอดเดรส คือ ไม่มีการเพิ่มแมคแอดเดรสลงตารางแมคแอดเดรสอีกด้วย แต่สามารถรับ BPDU จากสวิตช์ข้างเคียงได้ นอกจานี้พอร์ตที่อยู่ในโหมดสแตนด์บาย (Standby Mode) จะเข้าสู่ Blocking State เช่นเดียวกัน

6.2.3 **Listening** – พอร์ตจะเปลี่ยนสเตทจาก Blocking State ไปเป็น Listening State ก็ต่อเมื่อพอร์ตได้รับการเลือกให้เป็นรูทพอร์ตหรือดีไซเนเตทพอร์ต หรือในอีกนัยหนึ่งก็คือ พอร์ตกำลังจะสามารถฟอร์เวิร์ดทราฟฟิก (Forward Traffic) ได้นั่นเอง โดยในสเตทนี้พอร์ตจะสามารถรับ – ส่ง BPDU ได้แต่ถ้าพอร์ตสูญเสียสถานะการเป็นรูทพอร์ตหรือดีไซเนเตทพอร์ต เมื่อใด พอร์ตจะกลับไปสู่ Blocking State อีกครั้ง

6.2.4 **Learning** – เมื่อผ่านช่วงเวลาที่เรียกว่า Forward Delay ใน Listening State แล้วพอร์ตจะเปลี่ยนไปสู่ Learning State โดยที่พอร์ตจะสามารถรับ – ส่ง BPDU ได้และสวิตช์ยังสามารถเรียนรู้แมคแอดเดรสได้อีกด้วย

6.2.5 **Forwarding** – เมื่อผ่านช่วงเวลาที่เรียกว่า Forward Delay ใน Learning State แล้วพอร์ตจะเปลี่ยนไปสู่ Forwarding State โดยพอร์ตจะสามารถรับ-ส่งข้อมูล, เรียนรู้แมคแอดเดรสและรับ-ส่ง BPDU ได้โดยในสเตทนี้ถือว่าได้ทำสแปนนิ่งทรีเสร็จสิ้นแล้วนั่นเอง

6.3 ชนิดของสแปนนิ่งโพรโตคอล (Types of Spanning Tree Protocol)

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

6.3.1 คอมมอนสแปนนิ่งทรี (Common Spanning Tree (CST))

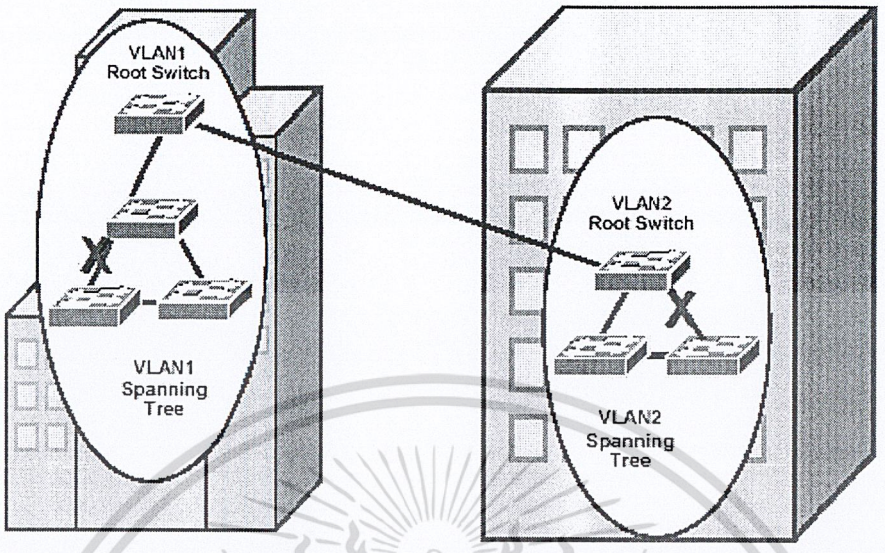
มาตรฐาน IEEE 802.1Q ได้กำหนดถึงการสร้างทรังค์ลิงค์ (Trunk Link) ที่ใช้ระหว่างสวิตช์และได้กำหนดให้ใช้เป็น 1 สแปนนิ่งทรีต่อหนึ่งเครือข่าย ก็คือทุกวีแลนมีชื่อเรียกว่าคอมมอนสแปนนิ่งทรี (Common Spanning Tree (CST)) หรือ โมโนสแปนนิ่งทรี (Mono Spanning Tree (MST)) ทุก ๆ BPDUs จะถูกส่งไปทั่วทั้งเครือข่าย

การใช้เพียง 1 สแปนนิ่งทรีต่อหลาย ๆ วีแลน มีข้อดีคือ สามารถใช้คำสั่งได้ง่าย ไม่ยุ่งยาก และยังลดการใช้ซีพียู (CPU) ของสวิตช์ในการคำนวณอีกด้วย อย่างไรก็ตาม การกระทำเช่นนี้ยังมีข้อเสียอยู่ก็คือ Redundant Link ที่ได้กำหนดขึ้นจะไม่ได้ใช้งาน ถูกบล็อกเพื่อกำจัดลูปและยังมีข้อจำกัดในเรื่องวีแลนคือพอร์ตที่ถูกบล็อค อาจจะเป็นพอร์ตที่ใช้ส่งเฟรมข้อมูลของวีแลนใด ๆ ก็ได้ จึงอาจจะเป็นเหตุให้ไม่สามารถส่งเฟรมข้อมูลได้

6.3.2 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

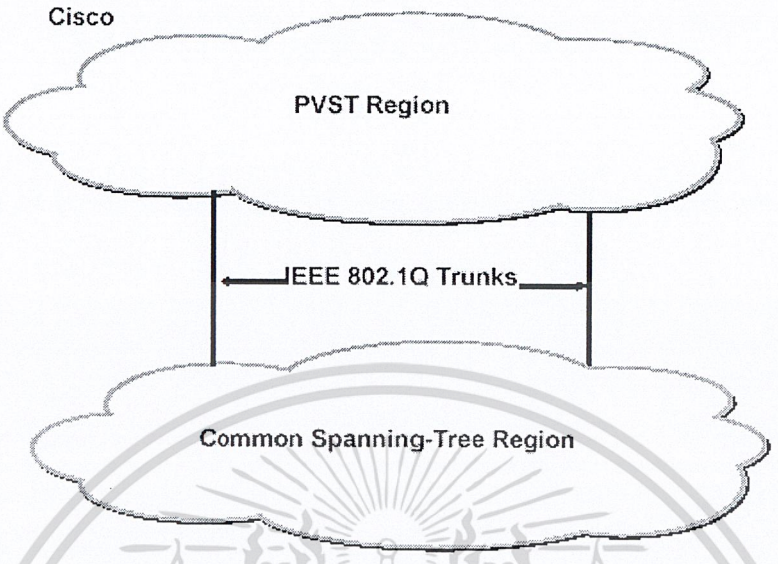


รูปที่ 6-8 แสดงตัวอย่างเครือข่ายที่ใช้เปอร์วีแลนสแปนนิ่งทรี

6.3.3 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 6-9 แสดงการทำงานของเปอร์วีแลนสแปนนิ่งทรีพลัส

บทที่ 7
หลักการสร้างและออกแบบโปรแกรม

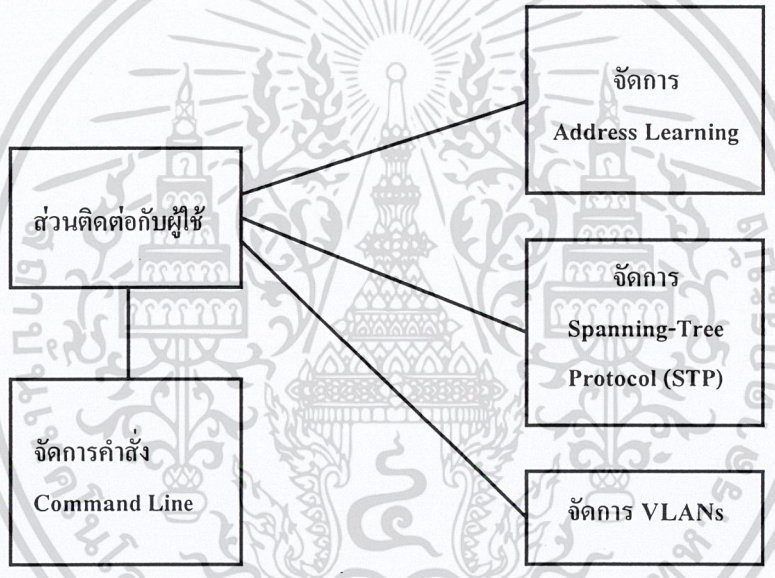
7.1 โครงสร้างของโปรแกรม (Design)

โปรแกรมนี้ ออกแบบโดยใช้หลักการของออบเจ็กต์โอเรียนเต็ดโปรแกรมมิง (Object Oriented Programming) เพื่อให้โปรแกรมง่ายต่อการเพิ่มเติมและดัดแปลงโปรแกรม โดยแบ่งการทำงานของโปรแกรมออกเป็นส่วนย่อย ๆ เป็นคลาสต่าง ๆ แต่ละคลาสจะมีหน้าที่การทำงานที่แตกต่างกันดังนี้

- *Frame1* เป็นคลาสสำหรับสร้างหน้าอินเทอร์เฟซเพื่อติดต่อกับผู้ใช้ โดยหน้าอินเทอร์เฟซจะประกอบไปด้วยเมนูบาร์ ทูลบาร์ และส่วนที่ใช้แสดงภาพเครือข่ายจำลองในรูปแบบกราฟฟิกส์
- *Interface* เป็นคลาสที่ประกอบไปด้วยข้อมูลของอินเทอร์เฟซแต่ละตัว เช่น ชื่ออินเทอร์เฟซ ประเภทอินเทอร์เฟซ และสถานะของอินเทอร์เฟซ
- *Workstation* เป็นคลาสที่ประกอบไปด้วยข้อมูลต่างๆ ของคอมพิวเตอร์ ได้แก่ ชื่อคอมพิวเตอร์ และค่าแมคแอดเดรส
- *Dialog4* เป็นคลาสสำหรับให้ผู้ใช้เชื่อมต่อสวิตช์แต่ละตัวเข้าด้วยกัน หรือเชื่อมต่อคอมพิวเตอร์เข้ากับสวิตช์
- *AddressLearning* เป็นคลาสสำหรับจัดการการเรียนรู้ค่าแมคแอดเดรส
- *Switch* เป็นคลาสที่ประกอบไปด้วยข้อมูลต่างๆ ของสวิตช์ เช่น ชื่อสวิตช์ อินเทอร์เฟซทั้งหมดของสวิตช์ ตารางการเรียนรู้ค่าแมคแอดเดรส โดยคลาสนี้จะมีฟังก์ชันการทำงานสำหรับเรียนรู้ค่าแมคแอดเดรส ฟังก์ชันการทำงานสำหรับรับ-ส่งเฟรมทั้งที่เป็น เฟรมอีเทอร์เน็ตและวีแลนเฟรม และฟังก์ชันการทำงานสำหรับรับ-ส่งเฟรมที่ใช้ในการทำสแปนนิ่งโพรโตคอล
- *SwitchImg* เป็นคลาสที่เก็บข้อมูลของภาพสวิตช์แต่ละตัว
- *ComImg* เป็นคลาสที่เก็บข้อมูลของภาพคอมพิวเตอร์แต่ละตัว
- *InterfaceImg* เป็นคลาสที่เก็บตำแหน่งของอินเทอร์เฟซแต่ละตัวที่มีการเชื่อมต่อ
- *Vlan* เป็นคลาสที่ใช้ในการเก็บข้อมูลของแต่ละวีแลนว่ามีสมาชิกเป็นสวิตช์ตัวใด อินเทอร์เฟซ (Interface) ไหนบ้าง รวมไปถึงเก็บค่าของเปอร์วีแลนสแปนนิ่งทรี (Per-VLAN Spanning Tree) ของวีแลนด้วย
- *VlanDatabase* เป็นคลาสที่เก็บข้อมูลของวีแลนแต่ละวีแลน เช่น หมายเลขวีแลน ชื่อวีแลน
- *SpanningTree* เป็นคลาสที่ใช้ในการสร้างสแปนนิ่งทรีให้กับเครือข่ายโดยควบคุมการทำงานของสวิตช์ทุกตัว
- *Panel2* เป็นคลาสที่ใช้สำหรับสร้างเครือข่าย และแสดงผลในรูปของกราฟฟิกส์ (Graphics)

- *SwitchCMD* เป็นคลาสที่ใช้สำหรับรับคำสั่งจากผู้ใช้มาตรวจสอบว่าเป็นคำสั่งที่ถูกต้องก่อนจะไปเลือกการทำงานของแต่ละคำสั่ง ซึ่งจะมีฟังก์ชันหลักเป็นตัวตรวจสอบว่าเป็นคำสั่งที่ถูกต้อง และส่งไปทำงานตามแต่ละคำสั่ง
- *SwitchConsole* เป็นคลาสสำหรับให้ผู้ใช้ป้อนคำสั่งต่างๆ ที่ใช้ในการตั้งค่าสวิตช์ และแสดงผลออกมาในรูปของ Text Mode
- *Panel1* เป็นคลาสสำหรับแสดงสถานะของสวิตช์แต่ละตัว และสถานะของอินเทอร์เฟซต่างๆ ของสวิตช์

โดยคลาสเหล่านี้สามารถแบ่งตามการจัดการได้ดังนี้

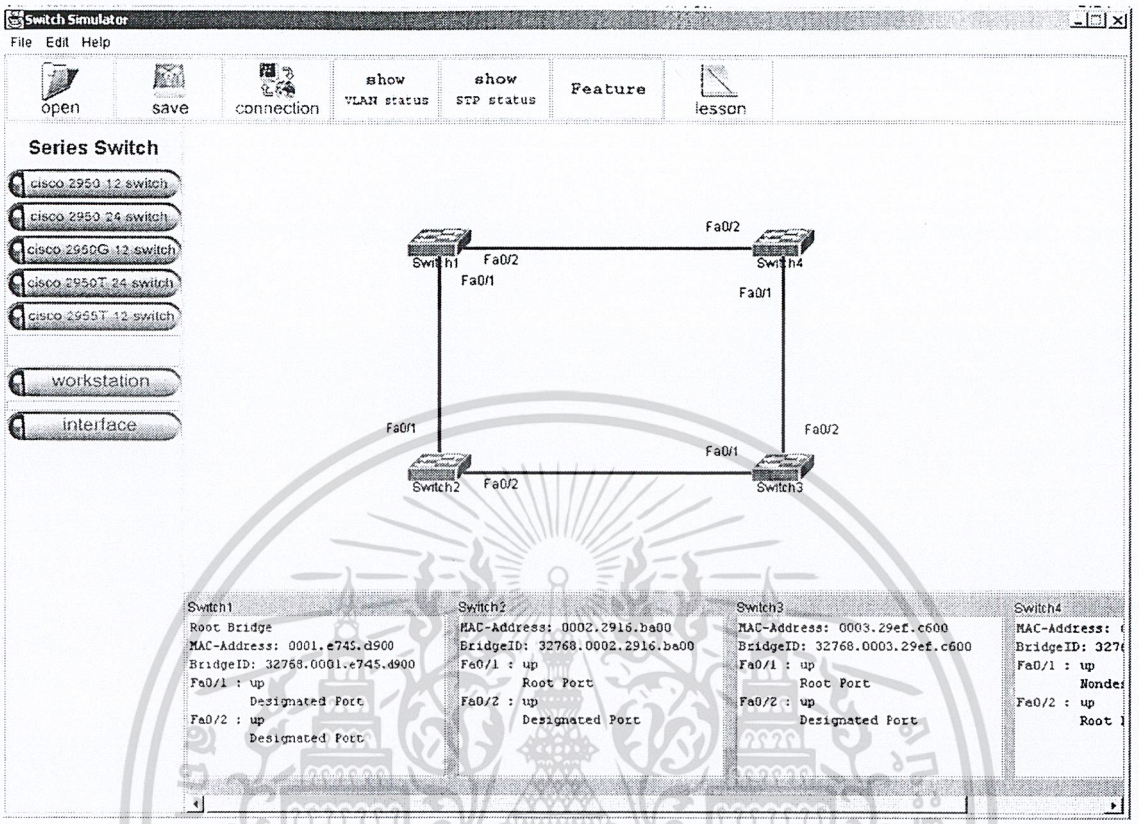


รูปที่ 7-1 การออกแบบโปรแกรม

7.2 การจัดการส่วนติดต่อกับผู้ใช้ (Graphic User Interface : GUI)

แบ่งออกเป็น 3 ส่วนหลักๆ คือ

1. ส่วนหน้าจอหลักของโปรแกรม ประกอบด้วยทูลบาร์ ส่วนของหน้าจอแสดงแผนภาพเครือข่าย และส่วนแสดงสถานะของสวิตช์ และอินเทอร์เฟซของสวิตช์แต่ละตัว การออกแบบส่วนนี้จะแบ่งเป็น 3 คลาส คือ คลาส Frame1 ซึ่งเป็นหน้าจอหลัก คลาส Panel2 ซึ่งเป็นหน้าจอแสดงผลภาพรวมของเครือข่าย โดยจะแสดงการเชื่อมต่อของสวิตช์และคอมพิวเตอร์แต่ละตัวตามที่กำหนด นอกจากนี้ผู้ใช้สามารถเพิ่มสวิตช์ คอมพิวเตอร์ และสร้างการเชื่อมต่อระหว่างสวิตช์กับสวิตช์ หรือสวิตช์กับคอมพิวเตอร์ได้ คลาส Panel1 ซึ่งเป็นส่วนแสดงสถานะของสวิตช์แต่ละตัว เมื่อมีการเพิ่มสวิตช์เข้าไปในระบบก็จะมีการสร้างออบเจ็กต์ของ Panel1 ของสวิตช์ตัวนั้นขึ้นมาด้วย แล้วนำมาเรียงกันในหน้าจอหลักของโปรแกรม



รูปที่ 7-2 หน้าจอหลักของโปรแกรม

ฟังก์ชันการทำงานสำหรับส่วนนี้ประกอบไปด้วยคลาส Frame1, Panel1 และ Panel2 มีฟังก์ชันการทำงานดังนี้

<u>คลาส Frame1</u>

- Frame1() เป็นคอนสตรัคเตอร์ของคลาส ในฟังก์ชันนี้จะแสดงหน้าจอโปรแกรมหลักทั้งส่วนของเมนูบาร์ และทูลบาร์
- newSwitch() เป็นฟังก์ชันสำหรับสร้างสวิตช์แต่ละตัว โดยจะมีการตรวจสอบชนิดของสวิตช์ที่เลือกก่อน และตรวจสอบจำนวนสวิตช์ทั้งหมดด้วยว่าสามารถเพิ่มสวิตช์เข้าไปในระบบได้อีก
- ส่วนฟังก์ชันอื่นๆ เป็นฟังก์ชันสำหรับจัดการการกระทำของผู้ใช้สำหรับแต่ละคอมโพเนนท์ ซึ่งเป็นการจัดการการกระทำที่ผู้ใช้กระทำกับคอมโพเนนท์หนึ่งๆ ในขณะที่คอมโพเนนท์หนึ่งๆ ถูกโฟกัสอยู่ เช่น jNewButton1_actionPerformed() เป็นการจัดการกับการกระทำของผู้ใช้กับปุ่มเพิ่มสวิตช์

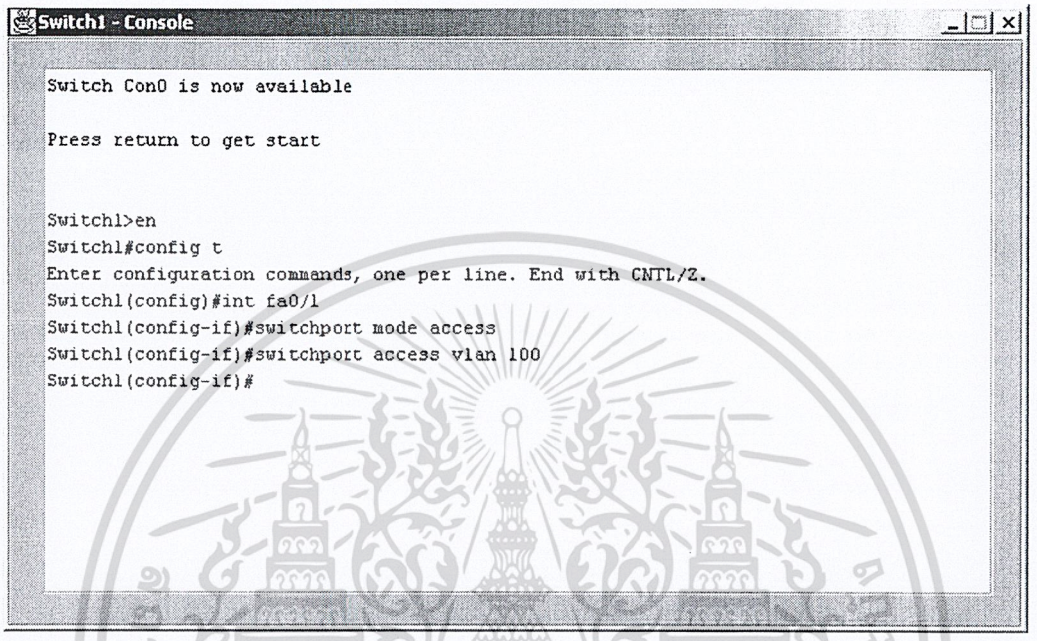
คลาส Panel1

- Panel1() เป็นคอนสตรัคเตอร์ของคลาส
- showDetail() เป็นฟังก์ชันสำหรับแสดงรายเอียดของสวิตช์แต่ละตัว รวมไปถึงรายละเอียดของอินเทอร์เฟซของสวิตช์นั้นๆ ด้วย

คลาส Panel2

- Panel2() เป็นคอนสตรัคเตอร์ของคลาสสำหรับแสดงผลภาพรวมของเครือข่าย
- delSwitch_actionPerformed() เป็นการจัดการกับการกระทำของผู้ใช้ กับเมนูป๊อปอัพ (Menu popup) สำหรับลบสวิตช์
- delComp_actionPerformed() เป็นการจัดการกับการกระทำของผู้ใช้กับเมนูป๊อปอัพ สำหรับลบคอมพิวเตอร์
- delInt_actionPerformed() เป็นการจัดการกับการกระทำของผู้ใช้กับเมนูป๊อปอัพ สำหรับลบอินเทอร์เฟซ
- console_actionPerformed() เป็นการจัดการกับการกระทำของผู้ใช้กับเมนูป๊อปอัพ สำหรับแสดงคอนโซลของสวิตช์ตัวที่เลือก
- connect_actionPerformed() เป็นการจัดการกับการกระทำของผู้ใช้กับเมนูป๊อปอัพ สำหรับเปลี่ยนอินเทอร์เฟซที่ใช้ในการเชื่อมต่อ
- this_mousePressed() เป็นฟังก์ชันจัดการกับการคลิกเมาส์
- this_mouseReleased() เป็นฟังก์ชันจัดการกับการปล่อยเมาส์
- this_mouseDragged() เป็นฟังก์ชันจัดการกับการคลิกเมาส์ และเลื่อนเมาส์ไปพร้อมกัน

2. หน้าจอสำหรับแสดงคอนโซลของสวิตช์แต่ละตัว เมื่อมีการเพิ่มสวิตช์เข้าไปในระบบก็จะมีการสร้างออบเจ็กต์ของ SwitchConsole ของสวิตช์ตัวนั้นขึ้นมาด้วย

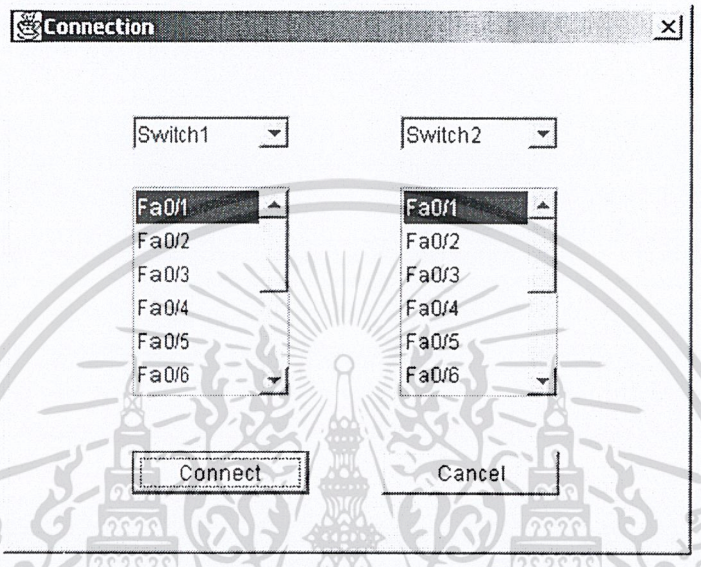


รูปที่ 7-3 หน้าจอสำหรับแสดงคอนโซลของสวิตช์

ฟังก์ชันการทำงานสำหรับส่วนนี้จะอยู่ในคลาส SwitchConsole มีฟังก์ชันการทำงานดังนี้

- SwitchConsole() เป็นคลาสคอนสตรัคเตอร์ของคลาสทำหน้าที่แสดงส่วนที่จะนำหน้าจอไปวางบนแท็ปเพน
- ShowPrompt() เป็นฟังก์ชันสำหรับแสดงพร้อมพ์ โดยจะดูจากโหมดการทำงานของสวิตช์แล้วจึงแสดงพร้อมพ์ของโหมดการทำงานนั้นๆ สำหรับฟังก์ชันนี้จะรับอาร์กิวเมนต์เข้าไปเป็นสตริงเพื่อนำไปแสดงผลต่อจากพร้อมพ์
- ส่วนฟังก์ชันอื่นๆ เป็นฟังก์ชันที่จัดการกับการกระทำที่เกี่ยวกับคีย์บอร์ดและเมาส์ เช่น ฟังก์ชัน jTextArea1_keyPressed() จัดการการกระทำที่เกี่ยวกับคีย์บอร์ดในขณะที่ TextArea ถูกโฟกัสอยู่ ฟังก์ชัน jTextArea1_MouseClicked() จัดการการกระทำของการคลิกเมาส์ในขณะที่ TextArea ถูกโฟกัสอยู่ และฟังก์ชัน jTextArea1_mouseReleased() จัดการกับการกระทำของการปล่อยเมาส์ ทำให้ผู้ใช้ไม่สามารถเลือกข้อความได้

3. หน้าจอสำหรับเชื่อมต่อสวิตช์แต่ละตัวเข้าด้วยกัน หรือ เชื่อมต่อสวิตช์กับคอมพิวเตอร์เข้าด้วยกัน โดยผู้ใช้สามารถเลือกสวิตช์ 2 ตัว ที่จะเชื่อมต่อเข้าด้วยกัน หรือเลือกสวิตช์และคอมพิวเตอร์ที่จะเชื่อมต่อเข้าด้วยกัน เมื่อเลือกสวิตช์แล้ว สามารถเลือกพอร์ตของสวิตช์ในการเชื่อมต่อได้



รูปที่ 7-4 หน้าจอสำหรับเชื่อมต่อสวิตช์แต่ละตัวเข้าด้วยกัน หรือเชื่อมต่อคอมพิวเตอร์เข้ากับสวิตช์

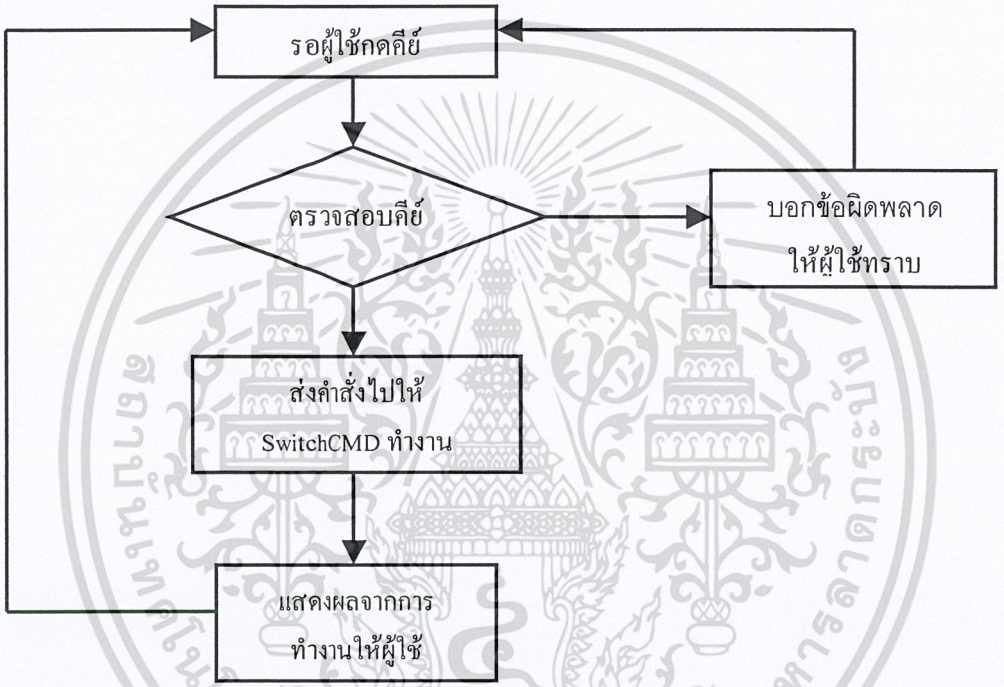
ฟังก์ชันการทำงานสำหรับส่วนนี้จะอยู่ในคลาส Dialog4 มีฟังก์ชันการทำงานดังนี้

- Dialog4() เป็นคอนสตรัคเตอร์ของคลาสสำหรับแสดงหน้าจอนี้
- InitValue() เป็นฟังก์ชันที่ใช้สำหรับแสดงสวิตช์ทั้งหมดให้ผู้ใช้เลือกเพื่อเชื่อมต่อสวิตช์เข้าด้วยกัน
- jButton1_actionPerformed() เป็นฟังก์ชันสำหรับจัดการการกระทำของผู้ใช้กับปุ่ม Connect
- jButton2_actionPerformed() เป็นฟังก์ชันสำหรับจัดการการกระทำของผู้ใช้กับปุ่ม Cancel
- ส่วนฟังก์ชันอื่นๆ เป็นฟังก์ชันสำหรับจัดการการกระทำของผู้ใช้ที่เกี่ยวกับการเลือกสวิตช์แต่ละตัว เช่น jComboBox1_actionPerformed() เป็นการจัดการให้แสดงพอร์ตทั้งหมดที่สามารถใช้ได้ของสวิตช์ตัวแรกที่ผู้ใช้เลือก และแสดงสวิตช์ตัวที่เหลือให้ผู้ใช้เลือกสวิตช์ตัวที่สองที่ต้องการเชื่อมต่อเข้ากับสวิตช์ตัวแรก
- ส่วนฟังก์ชันอื่นๆ เป็นฟังก์ชันสำหรับจัดการการกระทำของผู้ใช้ที่เกี่ยวกับการเลือกสวิตช์ เช่น jComboBox1_actionPerformed() เป็นการจัดการให้แสดงพอร์ตทั้งหมดที่สามารถใช้ได้ของสวิตช์ตัวแรกที่ผู้ใช้เลือก และแสดงพอร์ตทั้งหมดที่สามารถใช้ได้ของสวิตช์ตัวที่เลือก

7.3 การจัดการส่วนของ Command Line

มีคลาสที่จัดการส่วนนี้ 1 คลาส คือ คลาส SwitchCMD จะคอยรับคำสั่งที่ส่งมาจากคลาส Frame1 ที่จัดการในส่วนที่ติดต่อกับผู้ใช้ เมื่อรับคำสั่งมาแล้วจะนำมาตรวจสอบว่าเป็นคำสั่งที่ถูกต้อง สามารถทำงานได้หรือไม่ ถ้าถูกต้องจะส่งคำสั่งไปทำงาน และรอรับผลการทำงาน เพื่อนำมาแสดงผลให้ผู้ใช้ทราบ

ขั้นตอนการทำงานของโปรแกรม



รูปที่ 7-5 แสดงขั้นตอนการจัดการกับส่วน Command Line

ฟังก์ชันการทำงานสำหรับส่วนจะอยู่ในคลาส SwitchCMD มีฟังก์ชันการทำงานดังนี้

คลาส SwitchCMD

- rumCommand() เป็นฟังก์ชันหลักในการทำงานของคลาสนี้ ค่าที่รับเข้ามาในฟังก์ชันนี้จะมีอยู่ 3 ค่าคือ สวิตช์ คำสั่งที่ต้องการทำงานกับสวิตช์นั้นๆ และคอนโซลของสวิตช์นั้นๆ โดยเริ่มต้นที่ดูที่โหมดการทำงานของสวิตช์เพื่อเลือกชุดคำสั่งที่สามารทำงานได้กับโหมดนั้น หลังจากนั้นจะตัดคำมาทีละคำเพื่อเปรียบเทียบกับคำสั่งที่มีทุกคำสั่งในโหมดนั้น ถ้าเจอคำสั่งที่สอดคล้องกันก็จะนับจำนวนคำสั่งนั้นไว้ เมื่อเปรียบเทียบครบทุกคำสั่ง ก็จะนำค่าของจำนวนคำสั่งที่สอดคล้องกันมาตรวจสอบว่ามากกว่า 1 คำสั่งหรือไม่ ถ้ามากกว่าก็จะไม่เลือกคำสั่งใดทำงาน และรายงานผลกลับไปให้กับผู้ ถ้าเท่ากับ 1 ก็จะส่งไปให้ฟังก์ชันของแต่ละคำสั่งทำงาน

- ฟังก์ชันอื่นๆ จะเป็นฟังก์ชันการทำงานของคำสั่งแต่ละคำสั่ง

7.4 การจัดการในส่วนของการรับ-ส่งเฟรม และการเรียนรู้ค่าแมคแอดเดรสของสวิตช์

การจัดการในส่วนของการรับ-ส่งเฟรมจะมีคลาส AddressLearning เป็นคลาสหลักในการจัดการ โดยคลาสนี้มีการสืบทอด (Inherit) มาจากคลาสแม่ที่เป็นเทรด (Thread) เนื่องจากมีการจัดการให้คอมพิวเตอร์มีการส่งเฟรมตลอดเวลา เพื่อให้สวิตช์มีการเรียนรู้ค่าแมคแอดเดรส ดังนั้นการแตกโพรเซสให้โปรแกรมจัดการให้คอมพิวเตอร์ส่งเฟรม และสวิตช์มีการเรียนรู้ค่าแมคแอดเดรสเป็นโพรเซสแบบเทรด ทำให้สิ้นเปลืองเวลาน้อยกว่ารอให้โปรแกรมทำงานเอง

ในการรับและส่งเฟรมของคอมพิวเตอร์ เฟรมที่รับและส่งจะมีลักษณะเป็นเฟรมอีเทอร์เน็ต ส่วนการรับและส่งเฟรมของสวิตช์ เฟรมที่รับและส่งจะมี 2 ประเภท คือ เฟรมอีเทอร์เน็ต และ เฟรมที่เป็นวีแลน แต่เนื่องจากเป็นโปรแกรมที่จำลองเฟรมอีเทอร์เน็ตจึงมีเฉพาะค่าแมคแอดเดรสของคอมพิวเตอร์ต้นทาง (Source Address) และค่าแมคแอดเดรสของคอมพิวเตอร์ปลายทาง (Destination Address) ส่วนเฟรมที่เป็นวีแลนจะเพิ่มข้อมูลในส่วนที่ระบุวีแลนเข้าไปในเฟรมอีเทอร์เน็ต

สำหรับการเรียนรู้ค่าแมคแอดเดรสของสวิตช์ เมื่อสวิตช์ได้รับเฟรมมาแล้ว จะนำค่าแมคแอดเดรสต้นทางของเฟรมนั้นมา แล้วค้นหาค่าแมคแอดเดรสนั้นในตารางเรียนรู้ค่าแมคแอดเดรส ซึ่งในตารางจะเก็บค่าแมคแอดเดรส และพอร์ตที่เชื่อมต่อกับแมคแอดเดรสนั้นๆ โดยสามารถตรวจสอบได้ดังนี้

1. ถ้าในตารางไม่มีแมคแอดเดรสนี้ โปรแกรมจะเพิ่มแมคแอดเดรส และพอร์ตที่รับเฟรมนี้มา ลงในตาราง
2. ถ้าในตารางมีแมคแอดเดรสนี้ จะตรวจสอบพอร์ตที่รับเฟรมนี้มากับพอร์ตที่ตรวจพบในตารางของแมคแอดเดรสนี้ ถ้าพอร์ตไม่ตรงกัน จะเปลี่ยนพอร์ตในตารางเป็นพอร์ตที่รับเฟรมนี้มา

สำหรับการส่งเฟรมของสวิตช์ มีขั้นตอนการทำงานของโปรแกรมดังนี้

1. ถ้าเฟรมที่ได้รับมาเป็นเฟรมอีเทอร์เน็ต โปรแกรมจะตรวจสอบว่าพอร์ตที่ได้รับเข้ามามีค่าวีแลนเป็นอะไร จากนั้นนำค่าแมคแอดเดรสไปค้นหาในตารางเรียนรู้ค่าแมคแอดเดรส
   - ถ้าพบแมคแอดเดรสนี้ในตาราง จะตรวจสอบว่าพอร์ตที่ระบุไว้เป็นวีแลนเดียวกันกับพอร์ตที่รับเข้ามา ถ้าเป็นวีแลนเดียวกันสวิตช์จะส่งเฟรมนี้ออกไปทางพอร์ตที่ระบุในตาราง
   - ถ้าไม่พบแมคแอดเดรสนี้ในตาราง สวิตช์จะส่งเฟรมนี้ออกไปทุกพอร์ต
2. ถ้าพอร์ตที่ส่งออกไปเป็นแอ็กเซสลิงค์ จะตรวจสอบว่าเป็นวีแลนเดียวกันก่อนจึงส่งออกไป แต่ถ้าเป็นทรังค์ลิงค์ โปรแกรมจะเพิ่มข้อมูลที่ระบุวีแลนเข้าไปในเฟรมนั้น และตรวจสอบว่าวีแลนนี้สามารถส่งผ่านทรังค์ลิงค์นี้ได้ก่อนที่จะส่งออกไป

ฟังก์ชันการทำงานจะอยู่ในคลาส AddressLearning คลาส Switch และคลาส Workstation มีฟังก์ชันการทำงานดังนี้

คลาส AddressLearning

- run() เป็นฟังก์ชันสำหรับให้เทรดทำงาน โดยจะทำงานทุกๆ 1 วินาที เพื่อเรียกให้ฟังก์ชันของคอมพิวเตอร์ส่งเฟรมออกไป

คลาส Switch

- accessMACAddressTable() เป็นฟังก์ชันสำหรับเรียนรู้ค่าแมคแอดเดรส
- sendFrame() ฟังก์ชันนี้จะตรวจสอบว่าเชื่อมต่ออยู่กับสวิตช์ และคอมพิวเตอร์ตัวใดบ้าง และตรวจสอบกับตารางแมคแอดเดรส ตรวจสอบพอร์ตที่ใช้ในการส่งเฟรม แล้วจึงเรียกฟังก์ชัน receiveFrame() ของคลาส Switch เพื่อให้สวิตช์รับเฟรมมาเก็บไว้ และฟังก์ชัน receiveFrame() ของคลาส Workstation เพื่อให้คอมพิวเตอร์รับเฟรม
- receiveFrame() ฟังก์ชันนี้จะรับเฟรมเข้ามา แล้วเรียกฟังก์ชัน accessMACAddressTable() เพื่อเรียนรู้ค่าแมคแอดเดรส

คลาส Workstation

- sendFrame() ฟังก์ชันนี้จะตรวจสอบว่าเชื่อมต่ออยู่กับสวิตช์ตัวใด แล้วจึงเรียกฟังก์ชัน receiveFrame() ของสวิตช์ตัวนั้นเพื่อรับเฟรม
- receiveFrame() เป็นฟังก์ชันสำหรับรับเฟรมจากสวิตช์

7.5 การจัดการในส่วนของสแปนนิ่งทรีโพรโตคอล

การจัดการในการสร้างสแปนนิ่งทรีเพื่อทำการกำจัดลูปในระบบเครือข่ายนั้น จะใช้คลาส SpanningTree เป็นคลาสหลักในการจัดการ ควบคุมการทำงานของสวิตช์ทั้งหมด เพื่อทำการสร้างสแปนนิ่งทรีขึ้นในระบบเครือข่าย โดยคลาส SpanningTree จะทำหน้าที่สั่งการให้สวิตช์ทั้งหมดเริ่มต้นรับ-ส่ง BPDU เพื่อทำการเรียนรู้และสร้างสแปนนิ่งทรี

ฟังก์ชันการทำงานของสแปนนิ่งโพรโตคอลจะอยู่ในคลาส SpanningTree และคลาส Switch โดยมีฟังก์ชันในการทำงานดังนี้

คลาส SpanningTree

- enableCommonSpanningTree() เป็นฟังก์ชันที่ใช้ในการสั่งงานสวิตช์ทั้งหมดโดยเรียก doCommonSpanningTree() ของสวิตช์และควบคุมการสร้างคอมมอนสแปนนิ่งทรี (Common Spanning Tree -- เป็นการกำจัดลูปของทั้งเครือข่าย) จนเสร็จสิ้น
- disableCommonSTP() เป็นฟังก์ชันที่ใช้ในการยกเลิกคอมมอนสแปนนิ่งทรีที่เคยสร้างไว้ โดยทำการตั้งค่าต่าง ๆ ให้กลับไปเป็นเหมือนตอนเริ่มต้น

- enablePerVLANSpanningTree()เป็นฟังก์ชันที่ใช้ในการสั่งงานสวิตช์ทั้งหมดที่เป็นสมาชิกของวีแลน โดยเรียก doPerVLANSpanningTree() ของสวิตช์และควบคุมการสร้างเปอร์วีแลนสแปนนิ่งทรี (Per-VLAN Spanning Tree -- เป็นการกำจัดลูปของวีแลนที่กำหนด) จนเสร็จสิ้น
- disablePerVLANSTP เป็นฟังก์ชันที่ใช้ในการยกเลิกเปอร์วีแลนสแปนนิ่งทรีที่เคยสร้างไว้ โดยทำการตั้งค่าต่าง ๆ ให้กลับไปเป็นเหมือนตอนเริ่มต้น

คลาส Switch

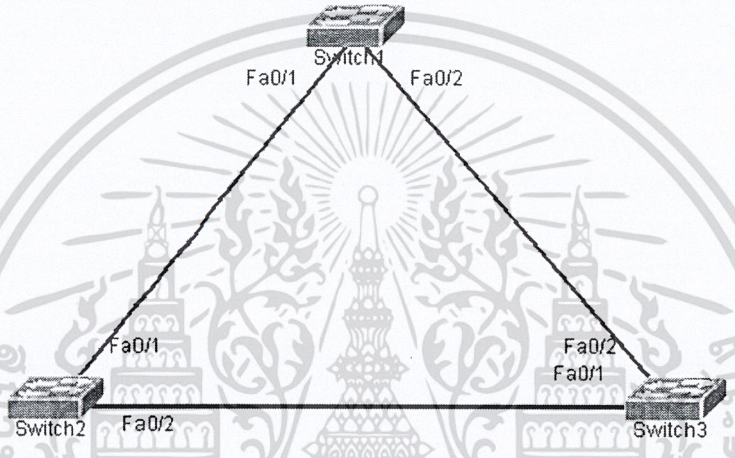
- doCommonSpanningTree() เป็นฟังก์ชันที่ใช้ในส่ง-รับ BPDUs ให้กับสวิตช์ข้างเคียงทั้งหมด โดยเรียกฟังก์ชัน receiveBPDU() เพื่อรับ BPDUs ไปคำนวณ
- receiveBPDU() เป็นฟังก์ชันที่ใช้ในการรับ BPDU มาเพื่อทำการคำนวณค่าต่าง ๆ เพื่อสร้างคอมมอนสแปนนิ่งทรี
- doPerVLANSpanningTree()เป็นฟังก์ชันที่ใช้ในส่ง-รับ BPDUs ให้กับสวิตช์ข้างเคียงที่เป็นสมาชิกของวีแลนเดียวกัน โดยเรียกฟังก์ชัน receiveBPDU() เพื่อรับ BPDUs ไปคำนวณ
- receiveBPDUforVlan()เป็นฟังก์ชันที่ใช้ในการรับ BPDU มาเพื่อทำการคำนวณค่าต่าง ๆ เพื่อสร้างเปอร์วีแลนสแปนนิ่งทรี (Per-VLAN Spanning Tree)

บทที่ 8

ตัวอย่างและการทดสอบการทำงานของโปรแกรม

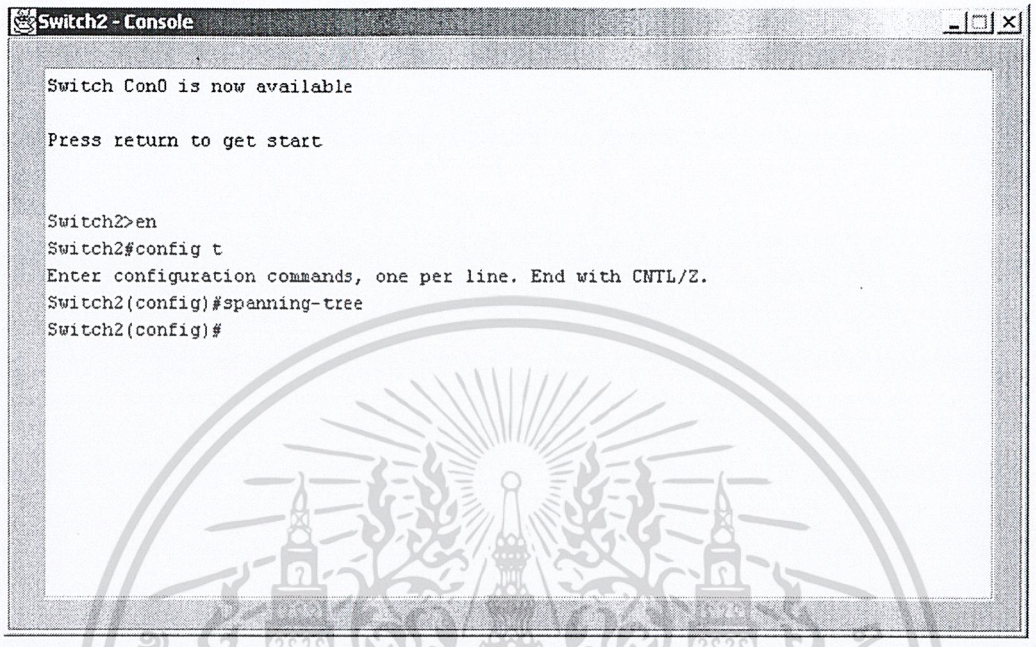
ทดสอบการทำงานของการใช้สแปนนิ่งทรีโพรโตคอล

เมื่อสร้างเครือข่ายจำลองให้เป็นดังรูป



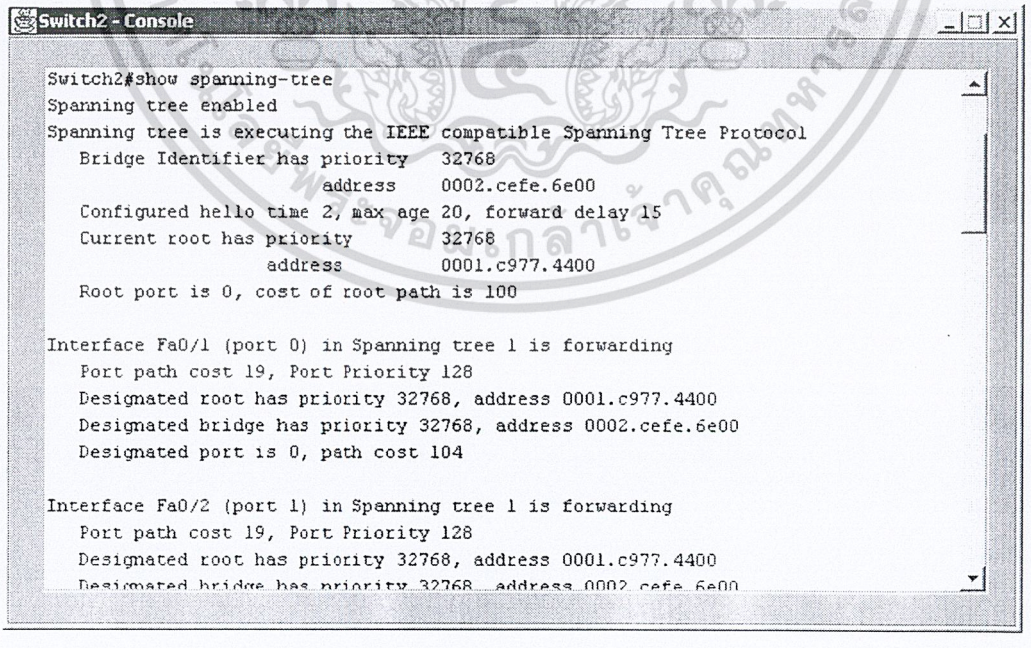
รูปที่ 8-1 แสดงภาพจำลองเครือข่าย

ใช้คำสั่ง Spanning-tree เพื่อสั่งการให้สวิตช์ทุกตัวทำการสร้างคอมมอนสแปนนิ่งทรี



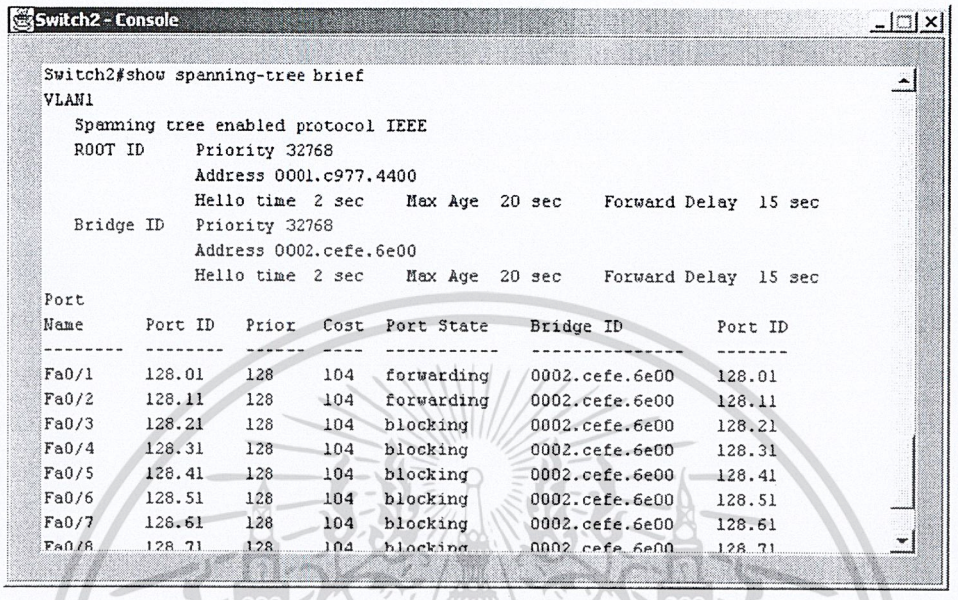
รูปที่ 8-2 แสดงการใช้คำสั่ง Spanning-tree

ใช้คำสั่ง Show spanning-tree เพื่อดูผลลัพธ์ในการสร้างคอมมอนสแปนนิ่งทรี



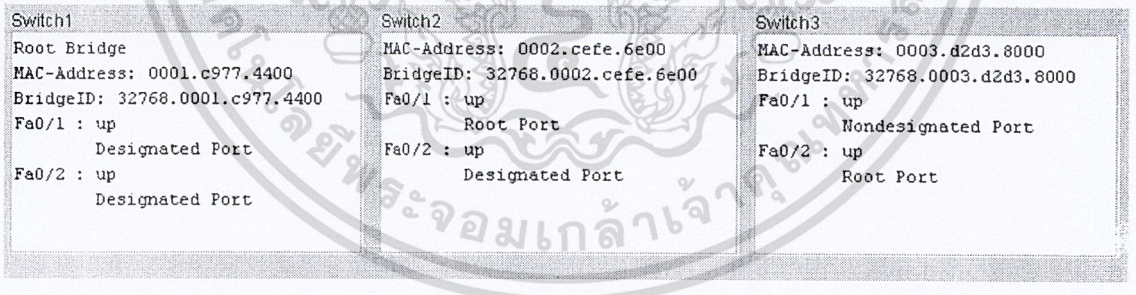
รูปที่ 8-3 แสดงการใช้คำสั่ง show spanning-tree

ใช้คำสั่ง Show spanning-tree brief เพื่อดูผลลัพธ์ของการสร้างคอมมอนสแปนนิ่งทรีอย่างย่อ ๆ



รูปที่ 8-4 แสดงการใช้คำสั่ง show spanning-tree brief

เมื่อมีการสร้างคอมมอนสแปนนิ่งทรี ผู้ใช้สามารถดูสถานะของสแปนนิ่งทรีได้จากที่หน้าจอหลักของโปรแกรม บริเวณที่แสดงสถานะของสวิตช์และอินเทอร์เฟซ



รูปที่ 8-5 แสดงสถานะของสวิตช์และอินเทอร์เฟซเมื่อมีการสร้างคอมมอนสแปนนิ่งทรี

บทที่ 9
สรุปและวิจารณ์

โครงงานที่จัดทำขึ้นนี้ เป็นโครงงานที่มีจุดประสงค์เพื่อเปิดโอกาสทางการศึกษาให้กับบุคคลทั่วไปที่สนใจทางด้านการติดตั้งและตั้งค่าอุปกรณ์สวิตช์เลเยอร์สอง รวมไปถึงการศึกษาการทำงานและโพรโตคอลของสวิตช์ด้วย โดยโครงงานนี้คือซอฟท์แวร์ที่สามารถจำลองการตั้งค่าของสวิตช์เลเยอร์สองของซิสโก้ และจำลองการทำงานของโพรโตคอลที่ใช้ในการกำจัดลูปได้ อีกทั้งยังสามารถแสดงเครือข่ายจำลองออกมาเป็นภาพ 2 มิติเพื่อให้มองเห็นภาพรวมของเครือข่ายได้อย่างชัดเจนอีกด้วย

แม้ว่าในทุกวันนี้ สวิตช์จะมีการออกซีรี่ส์ใหม่ออกมาตลอด ทำให้การตั้งค่าสวิตช์ต้องมีการเปลี่ยนแปลงไป แต่ถ้าผู้ดูแลระบบมีความเข้าใจในพื้นฐานของการทำงานของสวิตช์ และหลักการพื้นฐานในการตั้งค่าสวิตช์ จะทำให้สามารถตั้งค่าสวิตช์โดยส่วนใหญ่ได้

9.1 ขอบเขตและข้อจำกัดของโปรแกรม

- สามารถติดตั้งสวิตช์ได้ไม่เกิน 10 ตัว
- ไม่สามารถใช้ได้ทุกคำสั่งที่มีในสวิตช์ของซิสโก้ (Cisco)
- เป็นการตั้งค่าที่อิงตามคำสั่งที่มีในสวิตช์ของซิสโก้ (Cisco) เท่านั้น

9.2 กลุ่มผู้ใช้โปรแกรม

บุคคลโดยทั่วไปที่มีความต้องการศึกษาและฝึกฝนการติดตั้งและ ตั้งค่าอุปกรณ์สวิตช์ หรือศึกษาการทำงานของสวิตช์ เหมาะกับผู้ที่ต้องการศึกษาแต่ไม่มีสวิตช์ให้ศึกษา หรือมีแต่ยังไม่เข้าใจการติดตั้งและตั้งค่าอุปกรณ์สวิตช์ หรือติดตั้งโดยไม่รู้ความสามารถที่แท้จริงที่สวิตช์สามารถทำได้

9.3 ปัญหาและอุปสรรค

- เนื่องจากคำสั่งในการตั้งค่าสวิตช์มีเป็นจำนวนมาก ทำให้ต้องใช้เวลาในการศึกษาและรวบรวมคำสั่งเป็นเวลาพอสมควร
- เนื่องจากสวิตช์ทำงานในระดับเลเยอร์สอง ทำให้ยากต่อการจำลอง เช่น เมื่อทำการต่อพอร์ตของสวิตช์นั้น จะอยู่ในระดับกายภาพ (Physical) ดังนั้นเมื่อทำการส่งข้อมูลไปตามพอร์ต ในสวิตช์จริง ๆ นั้น สามารถส่งข้อมูลผ่านไปตามสายได้เลย

9.4 แนวทางการประยุกต์และพัฒนา

โครงงานนี้ จัดทำขึ้นแบบ OOP (Object Oriented Programming) ซึ่งทำให้สามารถเพิ่มเติมและพัฒนาต่อได้โดยง่าย โดยสามารถเพิ่มคำสั่งในการตั้งค่าอุปกรณ์สวิตช์เข้าไปได้เลย รวมทั้งยังสามารถเพิ่มความสามารถต่าง ๆ ให้กับสวิตช์ได้ เช่น ในการทำสแปนนิ่งทรีโพรโตคอล สามารถเพิ่มเติมให้ทำเป็นแบบเปอร์วีแลนสแปนนิ่งทรีพลัส, เพิ่มความสามารถของสวิตช์ ให้เป็นสวิตช์เลเยอร์สาม หรือ สวิตช์เลเยอร์สี่, สร้างการเชื่อมต่อผ่านเครือข่ายจริงโดยสามารถเชื่อมโยงเครือข่ายของผู้ใช้หลาย ๆ คนเข้าด้วยกัน และสามารถส่งข้อมูลหากันได้

ภาคผนวก ก

สรุปคำสั่งทั้งหมด

Configure Terminal
Command Mode: Privileged EXEC Commands
เป็นคำสั่งที่ใช้สำหรับเปลี่ยนโหมดจากโหมด Privileged EXEC Commands เป็นโหมด Global Configuration
ตัวอย่างการใช้งาน Switch#**configuration terminal** Switch(config)#

Disable
Command Mode: Privileged EXEC Commands
เป็นคำสั่งที่ใช้สำหรับออกจากโหมด Privileged EXEC Commands เพื่อเข้าสู่โหมด User EXEC Commands
ตัวอย่างการใช้งาน Switch#**disable** Switch>

Enable
Command Mode: User EXEC Commands
เป็นคำสั่งที่ใช้สำหรับเปลี่ยนโหมดจากโหมด User EXEC Commands เป็นโหมด Privileged EXEC Commands
ตัวอย่างการใช้งาน Switch>**enable** Switch#

End

Command Mode: Interface Commands

เป็นคำสั่งที่ใช้สำหรับออกจากโหมด Interface Commands เพื่อเข้าสู่โหมด Global Configuration

ตัวอย่างการใช้งาน

Switch(config-if)#**end**

Switch(config)#

Exit

Command Mode: Global Configuration Commands, Interface Commands, VLAN Configuration Commands, Line Commands

เป็นคำสั่งที่ใช้สำหรับออกจากโหมดที่กำลังใช้งานอยู่ขณะนั้น เพื่อเข้าสู่โหมด Privileged EXEC Commands

ตัวอย่างการใช้งาน

Switch(config-if)#**exit**

Switch#

Hostname

Command Mode: Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดชื่อสวิตช์

ตัวอย่างการใช้งาน

Switch(config)#**hostname SwitchA**

SwitchA(config)#

Interface

Command Mode: Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับเข้าสู่การตั้งค่าอินเทอร์เฟสต่างๆ ของสวิตช์ โดยคำสั่งนี้จะตามด้วยชื่อของอินเทอร์เฟสที่ต้องการตั้งค่า

รูปแบบคำสั่ง: Interface <interface name>

<interface name> เป็นชื่อของอินเทอร์เฟส

ตัวอย่างการใช้งาน

Switch(config)#**int fa0/1**

Switch(config-if)#

MAC-address-table Aging-time

Command Mode: Global Configuration Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดค่าเวลาในการมีอยู่ของค่าในตารางแมคแอดเดรส นับตั้งแต่ค่านั้นๆ ไม่มีการใช้งานหรือไม่เกิดการเปลี่ยนแปลง

รูปแบบคำสั่ง: mac-address-table aging-time <number>

<number> เป็นค่าเวลา มีหน่วยเป็นวินาที

ตัวอย่างการใช้งาน

Switch(config)#**mac-address-table aging-time 200**

Switch(config)#

Show Interfaces

Command Mode: Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงรายละเอียดของอินเทอร์เฟส

รูปแบบคำสั่ง: Show Interface <interface name>

<interface name> เป็นตัวเลือกจะใส่หรือไม่ใส่ก็ได้ ถ้าใส่จะเป็นการระบุให้แสดงรายละเอียดเฉพาะอินเทอร์เฟสที่ระบุ ถ้าไม่เช่นนั้นจะแสดงทุกอินเทอร์เฟส

ตัวอย่างการใช้งาน

Switch#**show int fa0/1**

FastEthernet0/1 is up, line protocol is up

Hardware is Fast Ethernet, address is 0001.254b.a301 (bia 0001.254b.a301)

```
MTU 1500 bytes, BW 10000 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, 0CRC, 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
Switch#
```

Show Interfaces Switchport

Command Mode: Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงรายละเอียดในการตั้งค่า และการทำงานต่างๆ ของอินเทอร์เฟส

รูปแบบคำสั่ง: Show Interface <interface name> **switchport**

<interface name> เป็นชื่อของอินเทอร์เฟส

ตัวอย่างการใช้งาน

Switch#**show int fa0/1 switchport**

Name: FastEthernet0/1

Switchport: Enabled

Administrative Mode: static access

Operational Mode: static access

Administrative Trunking Encapsulation: dot1q

Negotiation of Trunking: Disables

Access Mode VLAN: 1 (default)

Trunking VLANs Enabled: NONE

Pruning VLANs Enabled: NONE

Priority for untagged frames: 0

Override vlan tag Priority: FALSE

Voice vlan: none

Appliance trust: none

Switch#

Show Interfaces Switchport Allowed-vlan

Command Mode: Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงวีแลนทั้งหมดที่สามารถผ่าน Trunk Link นี้ได้

รูปแบบคำสั่ง: Show Interface <interface name> **switchport allowed-vlan**

<interface name> เป็นชื่อของอินเทอร์เฟส

ตัวอย่างการใช้งาน

Switch#**show int fa0/1 switchport allowed-vlan**

"1-100,250,500-1005"

Switch#

Show MAC-address-table

Command Mode: Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงตารางแมคแอดเดรส

ตัวอย่างการใช้งาน

Switch#**show mac-address-table**

Dynamic Addresses Count: 9

Secure Addresses Count: 0

Static Addresses (User-defined) Count: 0

System Self Addresses Count: 41

Total MAC addresses: 50

Maximum MAC addresses: 8192

Non-static Address Table:

Destination Address Address Type VLAN Destination Port

0010.0de0.e289 Dynamic 1 Fa0/1

0010.7b00.1540 Dynamic 2 Fa0/5

0010.7b00.1545 Dynamic 2 Fa0/5

0060.5cf4.0076 Dynamic 1 Fa0/1

0060.5cf4.0077 Dynamic 1 Fa0/1

0060.5cf4.1315 Dynamic 1 Fa0/1

0060.70cb.f301 Dynamic 1 Fa0/1

00e0.1e42.9978 Dynamic 1 Fa0/1

00e0.1e9f.3900 Dynamic 1 Fa0/1

Switch#

Show Spanning-tree

Command Mode: Privileged EXEC Commands

เป็นคำสั่งที่ใช้ในการแสดงสแปนนิ่งทรีของวีแลนที่กำหนด

รูปแบบคำสั่ง: show spanning-tree [**brief**] | [**vlan** *stp-list*]

stp-list คือหมายเลขของวีแลนที่ต้องการ โดยสามารถเลือกได้ว่าจะใส่หรือไม่ใส่ โดยถ้าไม่ใส่วีแลน จะหมายถึง คอมมอนสแปนนิ่งทรี แต่ถ้าใส่วีแลน จะหมายถึง เปอร์วีแลนสแปนนิ่งทรีที่กำหนด

ตัวอย่างการใช้งาน

Switch#**show spanning-tree**

Spanning tree enabled

Spanning tree is executing the IEEE compatible Spanning Tree Protocol

Bridge Identifier has priority 32768

Address 0004.a367.c100

Configured hello time 2, max age 20, forward delay 15

Current root has priority 32768

Address 0001.61c5.b900

Root port is 1, cost of root path is 100

Interface Fa0/1 (port 0) in Spanning tree 1 is forwarding

Port path cost 19, Port Priority 128

Designated root has priority 32768, address 0001.61c5.b900

Designated bridge has priority 32768, address 0004.a367.c100

Designated port is 0, path cost 104

Interface Fa0/2 (port 1) in Spanning tree 1 is forwarding

Port path cost 19, Port Priority 128

Designated root has priority 32768, address 0001.61c5.b900

Designated bridge has priority 32768, address 0004.a367.c100

Designated port is 1, path cost 104

|

|

|

Interface Gi0/2 (port 26) in Spanning tree 1 is blocking

Port path cost 19, Port Priority 128

Designated root has priority 32768, address 0001.61c5.b900

Designated bridge has priority 32768, address 0004.a367.c100

Designated port is 25, path cost 104

Switch#

Show Spanning-tree Interface

Command Mode: Privileged EXEC Commands

เป็นคำสั่งที่ใช้ในการแสดงรายละเอียดสแปนนิ่งทรีของอินเทอร์เฟซที่กำหนด

รูปแบบคำสั่ง: show spanning-tree interface <interface name>

<interface name> เป็นชื่อของอินเทอร์เฟส

ตัวอย่างการใช้งาน

Switch#**show spanning-tree interface fa0/3**

Interface Fa0/3 (port 3) in Spanning tree is down

Port path cost 100, Port priority 128

Designated root has priority 6000, address 009D.2bba.7a40

Designated bridge has priority 32768, address 00e0.1e9f.4abc

Designated port is 3, path cost 410

Timers: message age 0, forward delay 0, hold 0

Switch#

Show VLAN

Command Mode: Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับแสดงรายละเอียดของวีแลน

ตัวอย่างการใช้งาน

```
Switch(config)#show vlan
VLAN Name                Status    Ports
---- ------------------------- --------- -------------------------------
1    default              active    Fa0/1, Fa0/2, Fa0/3, Fa0/4,
                                    Fa0/7, Fa0/8, Fa0/9, Fa0/10,
                                    Fa0/11, Fa0/12, Fa0/13, Fa0/14,
                                    Fa0/15, Fa0/16, Fa0/17, Fa0/18,
                                    Fa0/19, Fa0/20, Fa0/21, Fa0/22,
                                    Fa0/23, Fa0/24, Gi0/1, Gi0/2
100  VLAN0100            active    Fa0/5, Fa0/6
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
100  enet  100100 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
Switch#
```

Shutdown/No Shutdown

Command Mode: Interface Commands

เป็นคำสั่งที่ใช้สำหรับเปลี่ยนสถานะของอินเทอร์เฟสของสวิตช์ ถ้าต้องการใช้อินเทอร์เฟสมีสถานะเป็นเปิดจะใช้คำสั่ง No Shutdown และในทางกลับกัน ใช้คำสั่ง Shutdown เพื่อเปลี่ยนสถานะของอินเทอร์เฟสเป็นปิด

ตัวอย่างการใช้งาน

Switch(config)#**int fa0/1**

Switch(config-if)#**no shutdown**

%LINEPROTO-5-UPDOWN: Line protocol on Interface Fa0/1, changed state to up

%LINK-3-UPDOWN: Interface Fa0/1, changed state to up

Spanning-tree/No Spanning-tree

Command Mode: Global Configuration

เป็นคำสั่งที่ใช้ในการสร้างคอมมอนสแปนนิ่งทรี โดยคำสั่ง Spanning-tree จะเป็นการสร้าง และในทางกลับกัน คำสั่ง No Spanning-tree เป็นการ disable สแปนนิ่งทรี

รูปแบบคำสั่ง: spanning-tree | no spanning-tree [**vlan** *stp-list*]

stp-list คือหมายเลขของวีแลนที่ต้องการ โดยสามารถเลือกได้ว่าจะใส่หรือไม่ใส่ โดยถ้าไม่ใส่วีแลน จะหมายถึงคอมมอนสแปนนิ่งทรี แต่ถ้าใส่วีแลน จะหมายถึง เปอร์วีแลนสแปนนิ่งทรีที่กำหนด

ตัวอย่างการใช้งาน

Switch#**spanning-tree**

Switch#

Spanning-tree/No Spanning-tree Forward-time

Command Mode: Global Configuration

เป็นคำสั่งที่ใช้ในการเปลี่ยนค่า Forward time โดยมีขอบเขตได้ตั้งแต่ 4-200 วินาที ซึ่งคำสั่ง spanning-tree forward-time เป็นการเปลี่ยนค่า และในทางกลับกัน คำสั่ง no spanning-tree forward-time เป็นการเปลี่ยนไปใช้ค่าดีฟอลต์ของ forward-time คือ 15 วินาที

รูปแบบคำสั่ง: spanning-tree | no spanning-tree forward-time <number>

<number> เป็นค่าเวลา

ตัวอย่างการใช้งาน

Switch#**spanning-tree forward-time 20**

Switch#

Spanning-tree/No Spanning-tree Hello-time

Command Mode: Global Configuration

เป็นคำสั่งที่ใช้ในการเปลี่ยนค่า Hello time โดยมีขอบเขตได้ตั้งแต่ 1-10 วินาที ซึ่งคำสั่ง spanning-tree hello-time เป็นการเปลี่ยนค่า และในทางกลับกัน คำสั่ง no spanning-tree hello-time เป็นการเปลี่ยนไปใช้ค่าดีฟอลต์ของ hello-time คือ 2 วินาที

รูปแบบคำสั่ง: spanning-tree | no spanning-tree hello-time <number>

<number> เป็นค่าเวลา

ตัวอย่างการใช้งาน

Switch#**spanning-tree hello-time 8**

Switch#

Spanning-tree/No Spanning-tree Max-age

Command Mode: Global Configuration

เป็นคำสั่งที่ใช้ในการเปลี่ยนค่า Hello time โดยมีขอบเขตได้ตั้งแต่ 6-200 วินาที ซึ่งคำสั่ง spanning-tree max-age เป็นการเปลี่ยนค่า และในทางกลับกัน คำสั่ง no spanning-tree max-age เป็นการเปลี่ยนไปใช้ค่าดีฟอลต์ของ max-age คือ 20 วินาที

รูปแบบคำสั่ง: spanning-tree | no spanning-tree max-age <number>

<number> เป็นค่าเวลา

ตัวอย่างการใช้งาน

Switch# **spanning-tree max-age 100**

Switch#

Spanning-tree/No Spanning-tree Priority

Command Mode: Global Configuration

เป็นคำสั่งที่ใช้ในการเปลี่ยนค่า Priority ของสวิตช์ โดยมีขอบเขตได้ตั้งแต่ 0-65535 ซึ่งคำสั่ง spanning-tree priority เป็นการเปลี่ยนค่า และในทางกลับกัน คำสั่ง no spanning-tree priority เป็นการเปลี่ยนไปใช้ค่าดีฟอลต์ของ priority คือ 32768

รูปแบบคำสั่ง: spanning-tree | no spanning-tree priority <number>

<number> เป็นค่า priority ของสวิตช์

ตัวอย่างการใช้งาน

Switch# **spanning-tree priority 5000**

Switch#

Switchport Access Vlan

Command Mode: Interface Commands

เป็นคำสั่งที่ใช้สำหรับกำหนด VLAN ให้กับอินเทอร์เฟสชนิด Access Link

รูปแบบคำสั่ง: switchport access vlan <vlan>

<vlan> เป็นหมายเลขวีแลน (VLAN)

ตัวอย่างการใช้งาน

Switch(config)#**int fa0/1**

Switch(config-if)#**switchport mode access**

Switch(config-if)#**switchport access vlan 100**

Switch(config-if)#

Switchport Mode Access/Trunk

Command Mode: Interface Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดชนิดของอินเทอร์เฟส โดยถ้าต้องการให้อินเทอร์เฟสนั้นเป็น Access Link จะใช้คำสั่ง switchport mode access และถ้าต้องการให้เป็น Trunk Link จะใช้คำสั่ง switchport mode trunk

ตัวอย่างการใช้งาน

Switch(config)#**int fa0/1**

Switch(config-if)#**switchport mode trunk**

Switch(config-if)#

Switchport Trunk Allowed VLAN

Command Mode: Interface Commands

เป็นคำสั่งที่ใช้สำหรับกำหนด VLAN ที่อนุญาตให้ผ่าน และไม่อนุญาตให้ผ่านอินเทอร์เฟสที่เป็นชนิด Trunk Link ได้ โดยถ้าอนุญาตให้ผ่านจะใช้คำสั่ง switchport trunk allowed vlan add และถ้าไม่อนุญาตให้ผ่านจะใช้คำสั่ง switchport trunk allowed remove

รูปแบบคำสั่ง: switchport trunk allowed vlan add | remove <vlan list>

<vlan list> เป็นค่า VLANs ต่างๆ โดยมีค่าตั้งแต่ 2-1001

ตัวอย่างการใช้งาน

Switch(config)#**int fa0/1**

Switch(config-if)#**switchport mode trunk**

Switch(config-if)#**switchport mode allowed vlan add 2-100,250,500-1001**

Switch(config-if)#

Vlan/No Vlan

Command Mode: VLAN Configuration Commands

เป็นคำสั่งที่ใช้สำหรับเพิ่มค่า VLAN ลงใน VLAN Database และลบค่า VLAN ออกจาก VLAN Database ของสวิตช์ โดยถ้าต้องการเพิ่ม VLAN ลงใน VLAN Database จะใช้คำสั่ง vlan และถ้าต้องการลบ VLAN ออกจาก VLAN Database จะใช้คำสั่ง no vlan

รูปแบบคำสั่ง: vlan | no vlan <vlan>

<vlan> เป็นหมายเลข VLAN

ตัวอย่างการใช้งาน

Switch(vlan)#**vlan 100**

VLAN 100 added:

Name: VLAN0100

Switch(vlan)#

Vlan Database

Command Mode: Privileged EXEC Commands

เป็นคำสั่งที่ใช้สำหรับเปลี่ยนโหมดจาก Privileged EXEC Commands โหมดเป็น VLAN Configuration โหมด

ตัวอย่างการใช้งาน

Switch#**vlan database**

Switch(vlan)#

Vlan Name

Command Mode: VLAN Configuration Commands

เป็นคำสั่งที่ใช้สำหรับกำหนดชื่อ VLAN ใน VLAN Database

รูปแบบคำสั่ง: vlan <vlan> **name** <vlan name>

<vlan> เป็นหมายเลขวีแลน

<vlan name> เป็นชื่อของวีแลน

ตัวอย่างการใช้งาน

Switch(vlan)#**vlan 100 name marketing**

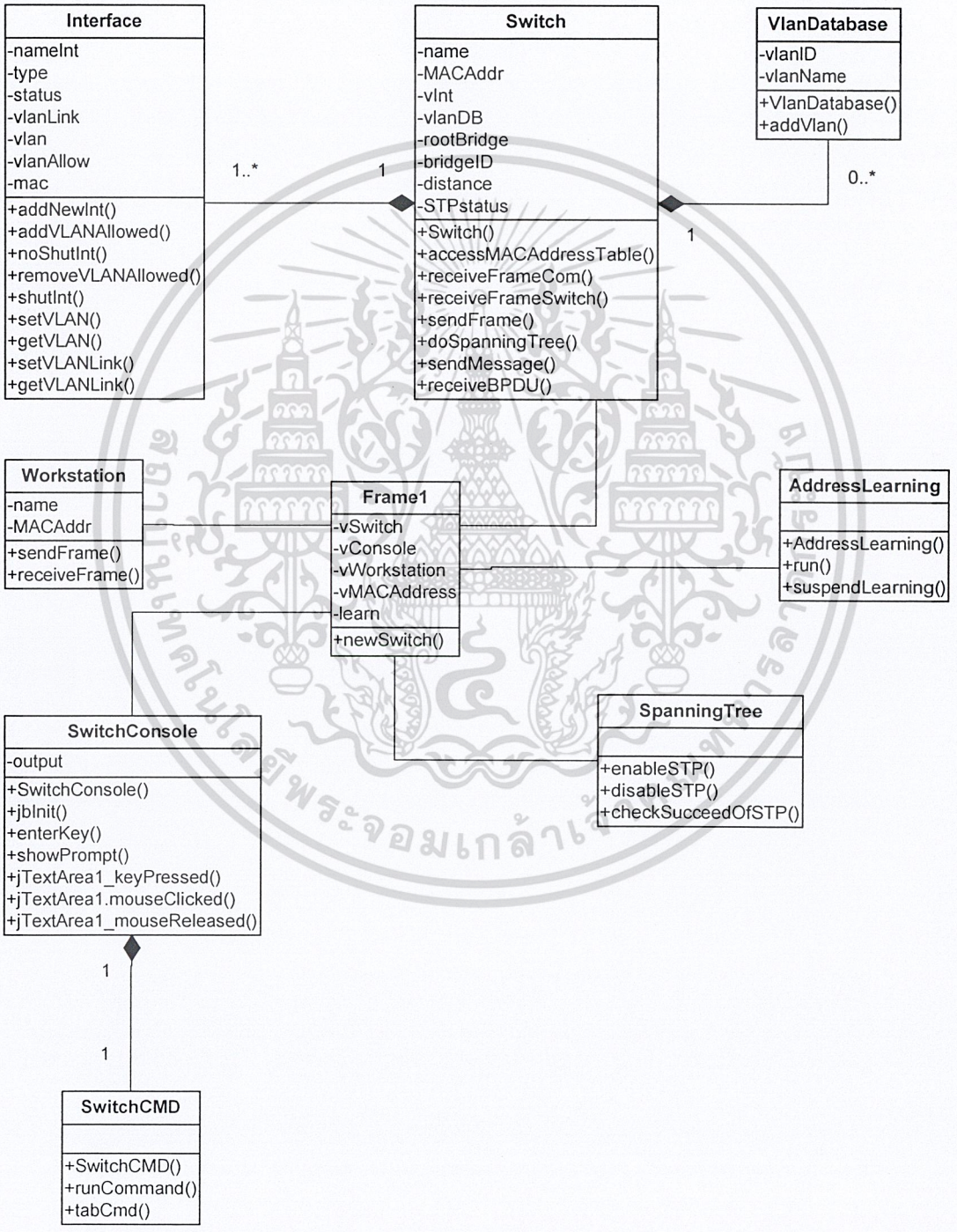
VLAN 100 modified:

Name: marketing

Switch(vlan)#

ภาคผนวก ข

คลาสไดอะแกรม



บรรณานุกรม

[1] Tim Boyles and Dave Hucaby : "Cisco CCNP Switching Exam Certification Guide", Indianapolis : Cisco Press, 2000.

[2] Louis R. Rossi, Louis D. Rossi and Thomas L. Rossi : "Cisco Catalyst LAN Switching", New York : McGraw Hill, 1976.

[3] "Catalyst 2950 Desktop Switch Command Reference", San Jose : Cisco Press, 2002.

[4] Andrew S. Tanenbaum : "Computer Networks", New York : Prentice Hall, 2002.

[5] www.cisco.com

[6] http://www.erg.abdn.ac.uk/users/gorry/course/lan-pages/bridge.html