การพัฒนาอุปกรณ์เราท์เตอร์โดยใช้ XORP พร้อมการเชื่อมต่อกับผู้ใช้แบบเว็บเบสยูสเซอร์อินเตอร์เฟส

A DEVELOPMENT OF XORP ROUTER EQUIPMENT WITH WEB-BASED USER INTERFACING

โดย

ชาตรี ชะโลธร

CHATREE CHALOTHORN

อาจารย์ที่ปรึกษา

ผศ.อัครินทร์ คุณกิตติ

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ภาคเรียนที่ 2 ปีการศึกษา 2550

A DEVELOPMENT OF XORP ROUTER EQUIPMENT WITH WEB-BASED USER INTERFACING

CHATREE CHALOTHORN

A SYSTEM DEVELOPMENT PROJECT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2/2007

หัวข้อ	การพัฒนาอุปกรณ์เราท์เตอร์โดยใช้ XORP พร้อมการเชื่อมต่อกับผู้ใช้แบบเว็บเบสยูสเซอร์อินเตอร์เฟส
นักศึกษา	นายชาตรี ชะโลธร
รหัสนักศึกษา	48066410
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2550
อาจารย์ที่ปรึกษา	ผศ.อัครินทร์ คุณกิตติ

บทคัดย่อ

เนื่องจากเราเตอร์ที่มีขายในท้องตลาดปัจจุบันมีราคาสูง การพัฒนาอุปกรณ์เราเตอร์ XORP พร้อมการเชื่อมต่อกับผู้ใช้แบบเว็บเบสยูสเซอร์อินเตอร์เฟส เพื่อสามารถนำมาใช้งานทดแทนเราเตอร์ในท้องตลาดได้ การพัฒนาโครงงานอุปกรณ์เราเตอร์โดยใช้ XORP สร้างจากการใช้ Embedded System ติดตั้งระบบปฏิบัติการ FreeBSD และโปรแกรมจัดการฟังก์ชันเราเตอร์ XORP (eXtensible Open Router Platform) เพื่อทำงานแทนเราเตอร์ราคาแพงได้ การสั่งงานและการปรับแต่งค่าต่างๆ ของเราเตอร์สามารถทำได้โดยการใช้คำสั่งแบบตัวอักษร (Command Line) ที่เป็นรูปแบบเฉพาะของ XORP เป็นการสั่งงานที่ค่อนข้างยาก เพื่อให้การสั่งงานเราเตอร์ง่ายขึ้นจึงเกิดแนวคิดในการพัฒนาการเชื่อมต่อกับผู้ใช้แบบเว็บสำหรับอุปกรณ์เราเตอร์ XORP เพื่อควบคุมเราเตอร์ XORP ด้วยเว็บเทคโนโลยี โครงการพัฒนาระบบงานนี้ พัฒนาอุปกรณ์เราเตอร์ XORP ที่สามารถทำงานได้บน Single Board ของ Soekris การพัฒนาตัวเราเตอร์ XORP และเว็บเบสยูสเซอร์อินเตอร์เฟส ทั้งหมดได้บรรจุเอาไว้ในหน่วยความจำประเภท Compact Flash ที่สามารถทำงานได้เทียบเท่ากับเราเตอร์ได้ โดยใช้การพัฒนาแบบ MiniBSD เพื่อย่อระบบปฏิบัติการและโปรแกรมจัดการฟังก์ชันเราเตอร์ XORP ลงหน่วยความจำ Compact Flash แทน Harddisk ในการพัฒนาโครงงานนี้ได้พัฒนาส่วนติดต่อกับผู้ใช้แบบเว็บเบสยูสเซอร์อินเตอร์เฟสโดยใช้ภาษา PHP เพื่อทำให้ผู้ใช้งานสามารถสั่งงานและเขียน Configuration file สำหรับอุปกรณ์เราเตอร์ XORP ได้ง่ายขึ้นและสั่งงานอุปกรณ์เราเตอร์ผ่านหน้าเว็บได้ทันที ในการพัฒนาโครงงานนี้ได้ทดสอบการทำงานของอุปกรณ์แล้วสามารถทำงานได้จริงอย่างมีประสิทธิภาพ

Title	A development of XORP Router Equipment with Web-based User Interfacing
Student	Mr.Chatree Chalothorn
Student ID.	48066410
Degree	Master of Science in Information Technology
Programme	Information Science
Academic Year	2008
Advisor	Asst. Prof. Akharin Khunkitti

ABSTRACT

Router is the product that it have an expensive. This project is develop a XORP Router Equipment with Web-based User Interfacing to use instead of an expensive Router in the market. Router Development by XORP. It was implemented from Embedded System with FreeBSD and function management on XORP Router Program. It can routing instead of Hardware Router. Normally XORP routing configurations are used by command line. It is not easy to understand so this project want to develop graphic Web-based user interface for configuring XORP Router that running on a Soekris single board. All of components and web-base user-interface are included in Compact flash memory that can work like normal router by MiniBSD development for compaction the operating system and XORP router to compact flash instead of a Harddisk. The graphic user interface is based on web-based development by PHP that's easy for end user to configure XORP via web technology and network technology. The Project can work with successfully and this Project can use in the real world.

กิตติกรรมประกาศ

โครงงานพัฒนาระบบงานนี้เกิดขึ้น และสำเร็จลุล่วงไปได้ด้วยดี ผู้จัดทำโครงงานขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ อัครินทร์ คุณกิตติ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงานที่ได้กรุณาเสียสละเวลาอันมีค่าในการให้คำแนะนำและแนวคิดในการจัดทำโครงงาน และให้คำปรึกษาด้านวิชาการที่เป็นประโยชน์ในการทำโครงงานและให้ความช่วยเหลือด้านอื่นๆ อีกทั้งสถานที่ทำโครงงาน ด้านเครื่องมือและอุปกรณ์ในการจัดทำโครงงาน ด้านการแก้ไขเอกสาร เรียบเรียงเอกสารรวมทั้งได้รับการดูแลเอาใจใส่ ให้ความเมตตา และให้กำลังใจแก่ผู้จัดทำด้วยดีเสมอมา ผู้จัดทำมีความซาบซึ้งในความกรุณาเป็นอย่างยิ่ง จึงขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

และขอกราบขอบพระคุณ คุณพ่อ คุณแม่ ที่ให้กำเนิด ให้การศึกษา ให้กำลังใจและเป็นแรงผลักดันให้ผู้จัดทำมีกำลังใจที่จะมุ่งมั่นในการศึกษาครั้งนี้จนเป็นผลสำเร็จลุล่วงด้วยดี

สุดท้ายขอขอบคุณ คุณชำนาญ ฉลาดแพทย์ เพื่อนๆ และรุ่นน้องทุกๆ ท่านที่ได้ให้คำแนะนำเสนอแนะในการเขียนโปรแกรม การจัดทำเอกสาร การทำอุปกรณ์เราท์เตอร์ให้ประสบผลสำเร็จ

นายชาตรี ชะโลธร

กุมภาพันธ์ 2551

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันการสื่อสารระหว่างเครื่องคอมพิวเตอร์ผ่านเครือข่ายอินเทอร์เน็ตสามารถทำได้อย่างแพร่หลายเรียกได้ว่าเป็นสิ่งจำเป็นของการสื่อสารอีกวิธีหนึ่ง การสื่อสารระหว่างเครื่องคอมพิวเตอร์เป็นการส่งข้อมูลผ่านตัวกลางชนิดต่างๆกัน เพื่อไปยังเครื่องคอมพิวเตอร์ที่อยู่ในเครือข่ายคอมพิวเตอร์เครือข่ายเดียวกันหรืออยู่ต่างเครือข่ายกัน การส่งแพ็กเก็ตไปยังเครื่องคอมพิวเตอร์ต่างเครือข่าย ต้องอาศัยอุปกรณ์เลือกเส้นทางชนิดหนึ่งที่เรียกว่าเราเตอร์ (Router) เราเตอร์จะรับข้อมูลเป็นแพ็กเก็ตเข้ามาตรวจสอบแอดเดรสปลายทาง จากนั้นนำมาเปรียบเทียบกับตารางเส้นทางที่ได้รับการโปรแกรมไว้ เพื่อหาเส้นทางที่ส่งต่อ หากเส้นทาง ที่ส่งมาจากอีเทอร์เน็ต และส่งต่อออกช่องทางของ Port WAN ที่เป็นแบบจุดไปจุด ก็จะมีการปรับปรุงรูปแบบสัญญาณให้เข้ากับมาตรฐานใหม่เพื่อส่งไปยังเครือข่าย WAN ได้ ดังนั้นเราเตอร์จึงนับว่าเป็นอุปกรณ์สำคัญในเครือข่ายอินเทอร์เน็ต เราเตอร์ที่ใช้อยู่ในปัจจุบันนี้มักจะพบว่าเป็นอุปกรณ์เฉพาะที่ผลิตโดยโรงงานผู้ผลิต การใช้งานเราเตอร์เหล่านั้นต้องเรียนรู้การใช้งานคำสั่งเฉพาะของแต่ละบริษัท นอกจากการใช้งานเราเตอร์ที่ผลิตจากโรงงานแล้วนักวิจัยยังสามารถสร้างเราเตอร์ได้โดยการใช้เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer) มาติดตั้งระบบปฏิบัติการ FreeBSD และโปรแกรมจัดการฟังก์ชันเราเตอร์ XORP (eXtensible Open Router Platform) เพื่อให้เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer) ทำหน้าที่เป็นอุปกรณ์เราเตอร์ได้

การพัฒนาอุปกรณ์เราเตอร์ XORP เพื่อใช้งานแทนเราเตอร์ราคาสูง สร้างจากการใช้เครื่องคอมพิวเตอร์ที่มีสถาปัตยกรรม X86 ระบบปฏิบัติการ FreeBSD และโปรแกรมเราเตอร์ XORP โดยนำโปรแกรมเราเตอร์ XORP มาติดตั้งบนระบบปฏิบัติการ FreeBSD การพัฒนาอุปกรณ์เราเตอร์ XORP ในโครงงานนี้ได้ติดตั้งโปรแกรม XORP บนหน่วยความจำประเภท Compact Flash แทนที่ Harddisk เพื่อทำหน้าที่เป็นตัวอุปกรณ์เลือกเส้นทาง (Router) สามารถทำหน้าที่เป็นอุปกรณ์เลือกเส้นทาง (Router) ได้อย่างมีประสิทธิภาพ การสั่งงานและการปรับแต่งค่าต่างๆ ของเราเตอร์สามารถทำได้โดยการใช้คำสั่งแบบตัวอักษร (Command Line) ซึ่งเป็นการสั่งงานที่ยากผู้ใช้ที่ไม่คุ้นเคยอาจจะสั่งงานไม่ได้ตามที่ต้องการ เพื่อทำให้การสั่งงานและควบคุมเราเตอร์ XORP ทำได้ง่ายขึ้น โครงงานนี้จึงได้ทำการศึกษา พัฒนาตัวอุปกรณ์เราเตอร์ XORP วิเคราะห์และออกแบบพัฒนาโปรแกรมยูสเซอร์อินเตอร์เฟสแบบเว็บสำหรับอุปกรณ์เราเตอร์ XORP

โปรแกรมจัดการฟังก์ชันเราเตอร์ XORP นั้นใช้ไฟล์คอนฟิกูเรชั่นในการตั้งค่าของตัวโปรแกรม การสั่งงานอาจใช้การสั่งงานผ่านทาง command line (xorpsh) หรือการตั้งค่าที่ไฟล์คอนฟิกูเรชั่นโดยการพัฒนาโครงงานครั้งนี้ใช้การวิเคราะห์ออกแบบระบบโดย Dataflow Diagram, State Diagram และ Flowchart Diagram เพื่อแสดงขั้นตอนการทำงานของโปรแกรม สำหรับตัวโปรแกรมนั้นได้เลือกพัฒนาโดยใช้ภาษา php, และ java script เพื่อพัฒนาส่วนติดต่อกับผู้ใช้ผ่านเว็บและใช้ lighttpd เวอร์ชั่น 1.4.15 เป็นเว็บเซอร์เวอร์

1.2 วัตถุประสงค์ในการพัฒนาระบบงาน

ในการพัฒนาอุปกรณ์เราเตอร์โดยใช้ XORP มีวัตถุประสงค์ดังนี้

1.2.1 เพื่อพัฒนาเราเตอร์ที่สร้างจากเครื่องคอมพิวเตอร์ส่วนบุคคลและโปรแกรมจัดการฟังก์ชันเราเตอร์ XORP

1.2.2 เพื่อศึกษาการพัฒนาเราเตอร์จากเครื่องคอมพิวเตอร์ส่วนบุคคลและการใช้ compact flash แทนฮาร์ดดิสก์

1.2.3 เพื่อศึกษาการพัฒนาเว็บเบสสำหรับการสั่งงานเราเตอร์ XORP โดยใช้ภาษา script php, java script และ shell script

1.2.4 เพื่อลดความยุ่งยากในการจัดการเราเตอร์

1.3 ขอบเขตของโครงการพัฒนาระบบงาน

ในการพัฒนาโครงการพัฒนาอุปกรณ์เราเตอร์โดยใช้ XORP พร้อมการเชื่อมต่อกับผู้ใช้แบบเว็บเบสยูสเซอร์อินเตอร์เฟส มีขอบเขตการทำงานดังนี้

1.3.1 โครงงานพัฒนาระบบงานนี้มุ่งเน้นการพัฒนาโปรแกรมที่ใช้ในการจัดการฟังก์ชันเราเตอร์ XORP บนระบบปฏิบัติการ FreeBSD เวอร์ชั่น 6.2 โครงงานที่พัฒนาขึ้นนี้ทำงานโดยผ่านทางเว็บบราวเซอร์ โปรแกรมจะเข้าไปจัดการฟังก์ชันเราเตอร์ XORP โดยสร้างไฟล์คอนฟิกและส่งคำสั่งไปยังเราเตอร์แทนการสั่งงานด้วย Command Line

1.3.2 โครงงานนี้ได้ติดตั้งระบบปฏิบัติการ FreeBSD 6.2 โปรแกรมจัดการฟังก์ชันเราเตอร์ XORP v1.4 เว็บเซอร์เวอร์ lighttpd เวอร์ชั่น 1.4.15 รวมทั้งเว็บเบสยูสเซอร์อินเตอร์เฟสทั้งหมดลงบนสื่อบันทึกข้อมูลประเภท Compact flash แทนการใช้งานฮาร์ดดิสก์

1.3.3 การพัฒนาโปรแกรมเพื่อใช้ในการจัดการฟังก์ชันเราเตอร์ XORP ในโครงงานนี้พัฒนาโดยใช้ภาษา PHP , java script, Shell Script และใช้โปรแกรมเว็บเซอร์เวอร์ lighttpd เวอร์ชั่น 1.4.15

1.3.4 การพัฒนาโครงงานครั้งนี้พัฒนาบนพื้นฐานของ IP version 4 เพียงเท่านั้น

1.4 ขั้นตอนและการดำเนินการโครงการ

ในการทำโครงการ มีขั้นตอนการดำเนินการดังนี้คือ

1.4.1 ศึกษาการทำงานของโปรแกรมจัดการฟังก์ชันเราเตอร์ XORP (eXtensible Open Router Platform)

1.4.2 ศึกษาการติดตั้ง การใช้งานและการปรับแต่งค่าคอนฟิกโปรแกรมจัดการฟังก์ชันเราเตอร์ XORP

1.4.3 ศึกษาคุณสมบัติและการเขียนโปรแกรมด้วยโปรแกรม PHP , java script , Shell Script

1.4.4 ศึกษาและทดลองปฏิบัติการย่อระบบปฏิบัติการ FreeBSD ลง compact flash เพื่อใช้งานแทนที่ Harddisk

1.4.5 ศึกษาและทดลองติดตั้งระบบปฏิบัติการ FreeBSD โปรแกรมจัดการฟังก์ชันเราเตอร์ XORP เว็บเซอร์เวอร์ lighttpd ลง compact flash

1.4.6 พัฒนาโครงงาน เขียนโปรแกรมเว็บเบสยูสเซอร์อินเทอร์เฟส

1.4.7 ทดลองการทำงานของเราเตอร์ที่สร้างจาก Personal Computer โดยใช้ compact flash แทนที่ Harddisk

1.4.8 ปรับปรุงตัวโครงงาน

1.4.9 ทดลองและสรุปผลที่ได้จากการทดสอบว่าตัวเราเตอร์สามารถทำงานได้จริงตามความต้องการหรือไม่

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่คาดว่าจะได้รับจากการพัฒนาเราเตอร์ XORP มีดังนี้คือ

1.5.1 ได้เราเตอร์ที่สามารถทำงานได้จริงที่สามารถสร้างได้จากเครื่องคอมพิวเตอร์ส่วนบุคคลโดยใช้ compact flash แทนที่ Harddisk และมีราคาไม่แพง

1.5.2 ได้เราเตอร์ที่สามารถควบคุมการทำงานได้ง่ายด้วยการใช้เว็บบราวเซอร์เข้ามาควบคุมการทำงานและสั่งงานจากเครือข่ายคอมพิวเตอร์ที่ใดก็ได้

1.5.3 ได้เรียนรู้การโปรแกรมด้วยภาษา PHP, java script และ Shell Script

1.5.4 ได้เรียนรู้การย่อระบบปฏิบัติการ FreeBSD ลงสื่อบันทึกข้อมูล compact flash และเรียนรู้การสั่งงานด้วยคำสั่งต่างๆ ของ FreeBSD

1.5.5 สามารถนำความรู้ที่ได้จากการทำโครงงานนี้ไปพัฒนาโครงงานอื่นๆ ที่เป็นการเขียนโปรแกรมและพัฒนาแอพลิเคชั่นผ่านทางเว็บ

บทที่ 2
เราเตอร์ โปรโตคอลสำหรับเลือกเส้นทางและทฤษฎีที่เกี่ยวข้อง

การสื่อสารข้อมูลผ่านเครือข่ายอินเทอร์เน็ต ปัจจุบันใช้เราเตอร์เป็นอุปกรณ์ในการเลือกเส้นทางการสื่อสารเพื่อส่งแพ็กเก็ตข้อมูลลงไปในเครือข่ายแบบ Datagram เพื่อให้ แพ็กเก็ตข้อมูลถึงจุดหมายปลายทางที่ต้องการตามโปรโตคอลที่เลือกใช้ในการสื่อสาร เราเตอร์ (router) คือ อุปกรณ์คอมพิวเตอร์เครือข่ายชนิดหนึ่งที่ทำหน้าที่ส่งต่อแพ็กเก็ตข้อมูล ไปสู่เป้าหมายปลายทางโดยใช้กระบวนการที่รู้จักกันว่าการเราติง (routing process) การเราติงเกิดขึ้นที่ Layer 3 ของ OSI Model (Network Layer)

2.1 หน้าที่การทำงานและส่วนประกอบของของเราเตอร์

2.1.1 หน้าที่การทำงานของเราเตอร์

เราเตอร์ทำหน้าที่เชื่อมต่อเครือข่ายตั้งแต่สองเครือข่ายขึ้นไป เราเตอร์มีหน้าที่ในการค้นหาเส้นทางเดินแพ็กเก็ตข้อมูลจากแหล่งกำเนิดไปยังเป้าหมายปลายทาง เพื่อการส่งต่อแพ็กเก็ตข้อมูลระหว่างเครือข่ายต่างๆ เราเตอร์เชื่อมต่อทั้งเครือข่าย LAN และเครือข่าย WAN ในการส่งต่อแพ็กเก็ตข้อมูล เราเตอร์ติดต่อสื่อสารกับเราเตอร์อื่นๆ ด้วยการใช้เราติงโปรโตคอล (routing protocol) และใช้ข้อมูลนี้ในการสร้างและดูแลตารางเราติง (routing table) ตารางเราติงเก็บค่าของเส้นทางที่ดีที่สุดเพื่อการส่งแพ็กเก็ตไปสู่จุดหมาย

2.1.2 ส่วนประกอบของเราเตอร์

เราเตอร์ก็เป็นคอมพิวเตอร์ชนิดหนึ่งที่ต้องการระบบปฏิบัติการเช่นเดียวกับเครื่องคอมพิวเตอร์ทั่วๆ ไป ในเราเตอร์ของบริษัท ซิสโก้ ระบบปฏิบัติการของเราเตอร์เรียกชื่อได้อีกอย่างหนึ่งว่า Internetwork Operating System (IOS) ระบบปฏิบัติการทำหน้าที่รันคอนฟิกูเรชั่นไฟล์ ในไฟล์คอนฟิกูเรชั่น ประกอบไปด้วยค่าตัวแปรต่างๆ เพื่อทำหน้าที่ควบคุมการจราจรเข้าออกจากเราเตอร์

เราเตอร์เป็นเครื่องคอมพิวเตอร์ชนิดพิเศษ ที่มีส่วนประกอบเช่นเดียวกับเครื่องคอมพิวเตอร์เดสก์ท็อป ทั่วๆไป เราเตอร์ประกอบด้วยส่วนประกอบดังต่อไปนี้คือ

1) CPU
2) หน่วยความจำต่างๆ เช่น RAM, NVRAM, Flash, ROM
3) System bus
4) Input / Output Interfaces

2.2 โปรโตคอลสำหรับเลือกเส้นทาง

เราติงโปรโตคอล คือ กระบวนการที่เราเตอร์ใช้ในการตัดสินใจค้นหาเส้นทางของเราเตอร์เพื่อส่งต่อแพ็กเก็ตข้อมูลต่อไปยังเป้าหมายปลายทาง เราเตอร์ตัดสินใจโดยใช้หมายเลขไอพีแอดเดรสเพื่อส่งแพ็กเก็ตไปสู่เป้าหมายที่ถูกต้อง เราติงโปรโตคอลแบ่งออกเป็นประเภทใหญ่ๆ ได้สองประเภทคือ Static Routing และ Dynamic Routing เมื่อเราเตอร์ใช้ Dynamic Routing ข้อมูลของเราเตอร์อื่นๆ จะถูกเรียนรู้กันโดยการส่งข้อมูลไปถึงกันแต่ถ้าเราเตอร์ใช้ Static Routing ผู้ดูแลระบบจะต้องทำหน้าที่ในการปรับแต่งข้อมูลด้วยตนเอง

2.2.1 Static Routing

การใช้งานเราติงโปรโตคอลแบบ Static Routing เป็น Routing Protocols ที่ไม่ได้ทำงานอย่างอัตโนมัติแต่ผู้ดูแลระบบจะต้องเป็นคนปรับแต่งค่าต่างๆ ด้วยตนเอง ลักษณะและข้อดีข้อเสียเป็นดังนี้คือ

1.ในระบบเครือข่ายขนาดใหญ่จะทำให้สูญเสียเวลาอย่างมาก และอาจจะเกิดความผิดพลาดได้ง่าย เช่น การใส่หมายเลขไอพีแอดเดรสผิด เป็นต้น แต่ในเครือข่ายขนาดเล็กการใช้งาน Static Routing ก็สามารถจัดการได้ง่ายและไม่ยุ่งยาก

2. เมื่อระบบเครือข่ายมีการเปลี่ยนแปลง Topology ผู้ดูแลระบบจะต้องทำการปรับแก้ค่า configuration ต่างๆ เอง เราเตอร์ไม่สามารถเรียนรู้ค่าต่างๆ จากเราเตอร์อื่นๆ

3. การใช้งาน Static Routing มีข้อดีคือ ผู้ดูแลระบบสามารถซ่อนข้อมูลที่ไม่ต้องการให้เราเตอร์อื่นๆ ได้รับไปได้

กระบวนการทำงานของ Static Routing

กระบวนการทำงานของ Static Route มีลำดับขั้นตอนดังนี้

1) ผู้ดูแลระบบเครือข่ายปรับแต่งค่า configuration
2) เราเตอร์ติดตั้งค่าการ Routing ลงในตารางเราติง (Routing Table)
3) แพ็กเก็ตถูกส่งออกไปโดยใช้การเลือกเส้นทางแบบ Static Routing

2.2.2 Dynamic Routing

การค้นหาเส้นทางของเราเตอร์แบบ Dynamic Routing ต่างจากแบบ Static Routing โปรโตคอลการค้นหาเส้นทางแบบ Dynamic Routing นั้นผู้ดูแลระบบไม่ต้องปรับแต่งค่าของตารางเราติงเอง (Routing Table) แต่เราเตอร์จะทำหน้าที่แลกเปลี่ยนข้อมูลซึ่งกันและกัน เราเตอร์จะเรียนรู้เส้นทางจากข้อมูล จากเราเตอร์ข้างเคียงและจะปรับค่าเราติงใน Routing Table เอง ถ้าเกิดการเปลี่ยนแปลง Topology เครือข่ายเกิดขึ้น เราเตอร์ก็จะเปลี่ยนแปลงค่าของเราติงเอง โดยอาศัยกลไกการ Routing ของแต่ละโปรโตคอลของ Dynamic Routing

เราติงโปรโตคอลประเภท Distance Vector

ลักษณะการทำงานพื้นฐานของ Distance Vector Routing Protocol จะส่งสำเนาของ Routing Table ของมันที่สมบูรณ์ซึ่งภายในบรรจุด้วย subnet address ที่เราเตอร์รู้จักทั้งหมด ไม่ว่าจะเป็น subnet address ที่เชื่อมต่อกับตัวมันเองโดยตรงหรือ subnet address ที่เรียนรู้มาจากเราเตอร์ตัวอื่นๆ ไปให้กับเราเตอร์เพื่อนบ้านที่เชื่อมต่ออยู่กับตัวเราเตอร์เอง

เมื่อเราเตอร์ได้รับ Routing Table มาจากเราเตอร์เพื่อนบ้านของมัน มันจะจัดการผนวกรวมเอา Routing Table ของเพื่อนบ้านที่ได้รับมาเข้ากับ Routing Table ของตัวมันเองและเราเตอร์ก็จะเชื่อว่ามันเองสามารถส่งแพ็กเก็ตออกไปตามอินเตอร์เฟสที่ได้รับมา

เมื่อเราเตอร์ทำเช่นนี้ไปเรื่อยๆ สุดท้ายก็จะได้ตารางเราติงอันใหม่ขึ้นมาและมันก็จะส่งตารางเราติงต่อไปให้เราเตอร์ตัวอื่นๆ เป็นทอดๆ ต่อๆไป

2.2.2.1) Routing Information Protocol (RIP)

เป็นโปรโตคอลการเลือกเส้นทางประเภท Distance Vector เพราะอาศัยหลักการส่ง Routing table ทั้งตารางต่อกันไปเป็นทอดๆ ให้เราเตอร์เพื่อนบ้าน เพื่อแลกเปลี่ยนกับเราเตอร์เพื่อนบ้าน ค่าเมตริกของ RIP ก็คือจำนวนของเราเตอร์ทั้งหมดก่อนที่จะไปถึง subnet address ปลายทาง หรือเรียกว่า Hop Count ค่าสูงสุดที่เป็นไปได้คือ 15 Hop Count โดยกรณีที่ hop count =16 จะถือว่าไม่สามารถไปถึงได้ (Unreachable) เพื่อเป็นการป้องกันการเกิด loop ขึ้นในการส่งข้อมูล ในกรณีที่เราเตอร์ได้มีโอกาสเรียนรู้ subnet address เดียวกันจากเราเตอร์อื่นและค่า cost เท่ากัน มันก็จะเพิ่มเส้นทางเข้าไปในตารางเราติงได้มากกว่าหนึ่งเส้นทาง โดย RIP สามารถเก็บจำนวนเส้นทางที่เท่ากันได้สูงสุด 6 เส้นทางต่อหนึ่งเน็ตเวิร์กแอดเดรสปลายทาง

RIP ใช้ routing metric (Hop Count) ในการวัดระยะทางระหว่างต้นทางและเครือข่ายปลายทาง แต่ละ hop ระหว่างทางมีค่าเท่ากับหนึ่ง เมื่อเราเตอร์ปรับปรุงค่า เราเตอร์จะเพิ่มค่าเมตริกเข้าไปหนึ่งสำหรับ hop count ต่อไปจากตัวเราเตอร์

RIP ใช้เวลาในการปรับแต่งประสิทธิภาพและปรับปรุงตารางเราติง ทุกๆ 30 วินาที โปรโตคอล RIP จะติดตามการเปลี่ยนแปลงที่เกิดขึ้นบนเน็ตเวิร์กโดยการส่ง บอร์ดคาสต์ หรือ มัลติคาสต์ตารางเราติงเทเบิลไปให้เราเตอร์เพื่อนบ้านทุกๆระยะเวลา 30 นาที แต่อาจก่อให้เกิดการสิ้นเปลืองในแง่ของซีพียูและแบนด์วิดธ์ของเครือข่ายได้

1.1) RIP Packet Format

รูปต่อไปนี้แสดงรูปแบบของ Packet ของ RIP

1-octet command field	1-octet version number field	2-octet zero field	2-octet AFI field	2-octet zero field	4-octet IP address field	4-octet zero field	4-octet zero field	4-octet metric field

รูปที่ 2.1 แสดง RIP Packet Format

ใน packet ประกอบด้วยแต่ละ field ดังนี้

Command - บอกให้รู้ว่าเป็นการ request หรือ response

- request เป็นการบอกให้เราเตอร์ ส่ง routing table มาให้ทั้งหมดหรือบางส่วน

- response สามารถส่งได้ทั้งที่ไม่มีการร้องขอ บรรจุ Routing Table

Version - บอกให้รู้ว่าใช้ RIP เวอร์ชันใด

Zero – ฟิลด์นี้ยังไม่ได้ใช้ใน RIP RFC 1058 มันถูกเพิ่มเพื่อเตรียมไว้สำหรับความหลากหลายของ RIP ชื่อของฟิลด์นี้มาจากค่าเริ่มต้นคือ ค่าศูนย์

Address-family identifier (AFI) – ใช้สำหรับ Address เพราะว่า RIP ถูกออกแบบมาสำหรับการส่งข้อมูล routing ของหลายๆ โปรโตคอล ฟิลด์นี้เป็นตัวบอกชนิดของ Address เช่น AFI สำหรับ IP คือค่า 2

Address – สำหรับการบรรจุหมายเลข IP Address ของแต่ละ entry

Metric – สำหรับการบอก Hop Count ของปลายทางค่านี้อยู่ระหว่าง 1 ถึง 15 ถ้าค่านี้เป็น 16 แสดงว่าไม่สามารถส่ง packet ไปได้

1.2) RIP2 Packet Format

1-octet command field	1-octet version number field	2-octet unused field	2-octet AFI field	2-octet route tag field	4-octet network address field	4-octet subnet mask field	4-octet next hop field	4-octet metric field

รูปที่ 2.2 แสดง RIP2 Packet Format

ใน packet ประกอบด้วยแต่ละ field ดังนี้

Command - บอกให้รู้ว่าเป็นการ request หรือ response เช่นเดียวกับ RIP Packet Format

Version – บอกเวอร์ชันของโปรโตคอล RIP ที่ใช้ ค่าสำหรับเวอร์ชันนี้ คือ 2

Unused – ค่าที่บรรจุอยู่คือ 0

Address-family identifier (AFI) – สำหรับการบอก Address family ถ้าค่าในฟิลด์นี้เป็น 0xFFFF หมายถึงการข้อมูลของ Authentication

Route tag – จัดเตรียมวิธีการระหว่าง internal route และ (เรียนรู้โดยโปรโตคอล RIP) และ external route (เรียนรู้โดยโปรโตคอลอื่นๆ)

IP address – สำหรับหมายเลข IP

Subnet address - สำหรับ subnet mask สำหรับแต่ละ entry

Next hop – สำหรับหมายเลข IP สำหรับ next hop ต่อไป ที่ packet สำหรับ entry นี้ควรจะส่งต่อไป

Metric – สำหรับจำนวน hop (router) ที่ใช้ส่งต่อข้อมูลไปถึงปลายทาง

เราติงโปรโตคอลประเภท Link State

เราติงโปรโตคอลที่จัดอยู่ในประเภทนี้ได้แก่ Open Shortest Path First (OSPF) และ IS-IS (Intermediate System – Intermediate System) โปรโตคอลการเลือกเส้นทางประเภทนี้เหมาะกับการนำไปใช้บนเน็ตเวิร์กโทโพโลยีทุกรูปแบบ และมีความยืดหยุ่นมากกว่าเราติงโปรโตคอลประเภท Distance Vector

เราติงโปรโตคอลแบบ Link State จะประกาศข้อมูลเกี่ยวกับสถานะของอินเตอร์เฟซของตัวเองเช่น ข้อมูลของอินเตอร์เฟสที่เชื่อมต่ออยู่ ค่าเมตริกที่ใช้และค่าตัวแปรอื่นๆ ที่ใช้โดย OSPF ออกไปให้เราเตอร์ที่อยู่ในพื้นที่ลำดับชั้นเดียวกันและให้เป็นหน้าที่ของเราเตอร์ที่รับข้อมูลไปคำนวณหาเส้นทางที่ดีที่สุดในการส่ง Packet ออกไปเองโดยใช้ SPF Algorithms

2.2.2.2) Open Shortest Path First (OSPF)

OSPF เป็นเราติงโปรโตคอลที่ได้รับการพัฒนาขึ้นมาใช้บนเน็ตเวิร์ก IP โดยคณะทำงาน Interior Gateway Protocol (IGP) ย่อย แห่งคณะกรรมการ Internet Engineering Task Force (IETF) โดยมีพื้นฐานมาจากอัลกอริทึมแบบ Shortest Path First (SPF) อัลกอริทึมนี้เรียกชื่ออีกอย่างได้ว่า Dijkstra's Algorithm ซึ่งเป็นตามชื่อของนักคณิตศาสตร์ที่คิดค้นอัลกอริทึมนี้

OSPF มีสองคุณลักษณะที่เด่นๆ ดังนี้

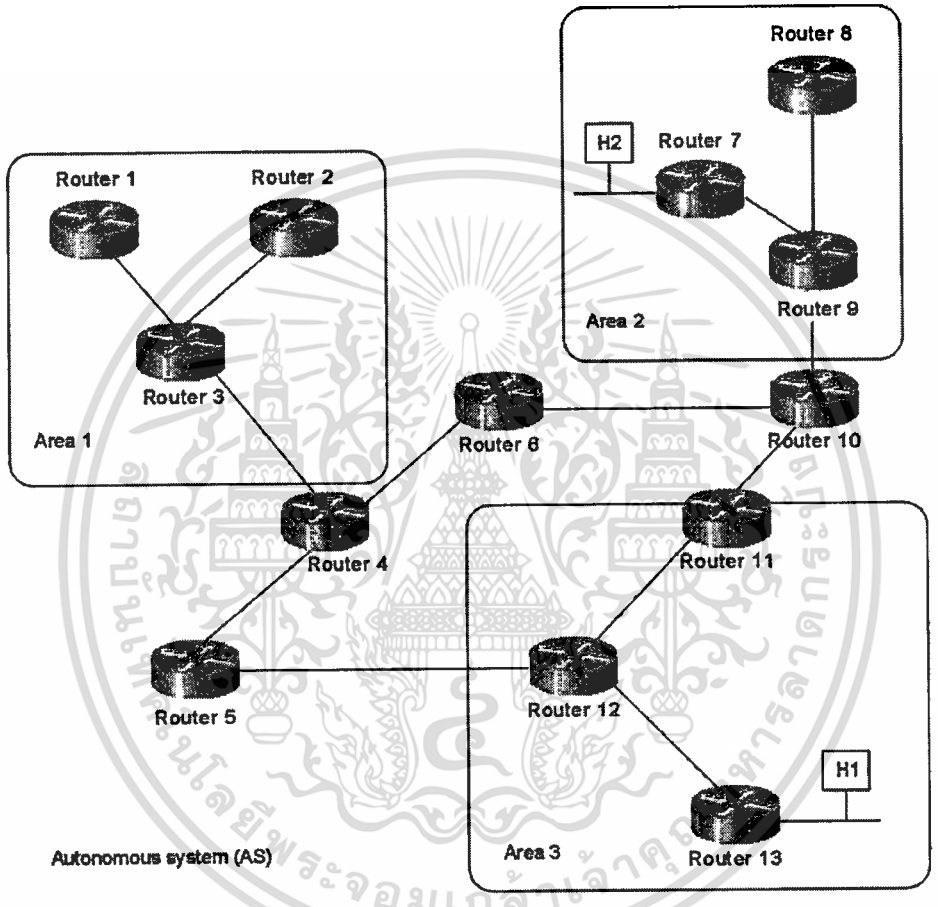
1. เป็นโปรโตคอลมาตรฐานสากล ข้อกำหนดได้ถูกตีพิมพ์ไว้ใน RFC 2328
2. เป็นโปรโตคอลการเลือกเส้นทางที่อาศัยการ update สถานะของเน็ตเวิร์กอินเตอร์เฟซให้กับเราเตอร์เพื่อนบ้าน แล้วให้เราเตอร์เพื่อนบ้านสร้างภาพรวมของเน็ตเวิร์กโทโพโลยีและคำนวณหาเส้นทางเอาเอง แต่จะไม่ส่ง Routing Table ไปให้เหมือน distance vector
3. มีการเลือกเส้นทางที่สั้นที่สุดโดยพิจารณาจาก Bandwidth
4. รองรับการสร้าง "OSPF Area" สามารถจัดแบ่งเน็ตเวิร์กออกเป็นโซนย่อยๆ ได้
5. สามารถทำ "route authentication" ระหว่างเราเตอร์เพื่อตรวจสอบตัวตนของเราเตอร์ได้

ลำดับชั้นของเราติงโปรโตคอล OSPF

OSPF สามารถทำงานแบบลำดับชั้นได้ ลำดับชั้นที่ใหญ่ที่สุดคือ Autonomous System (AS) คือ กลุ่มของ network ที่อยู่ภายใต้การจัดการ administrator และโปรโตคอลการเลือกเส้นทางประเภทเดียวกัน OSPF จัดเป็น intra-AS (interior gateway) routing protocol มันมีความสามารถในการส่งและรับการ route เส้นทางจาก Autonomous System อื่นๆ

AS สามารถแบ่งได้ตามกลุ่มของพื้นที่โดยใช้หมายเลข ของแต่ละพื้นที่ แต่ละกลุ่มก็คือ network ที่ติดกันและเชื่อมต่อกับ Host เราเตอร์ที่มีหลายๆ อินเตอร์เฟซและสามารถติดต่อกับหลายๆ area ได้เราเรียกว่า Area Border Router ที่ดูแล topological database ของแต่ละพื้นที่

Topological database คือฐานข้อมูลที่เก็บความสัมพันธ์ของ router แต่ละ network เอาไว้ topological database บรรจุด้วย LSA (Link-state Advertisements) ที่ได้รับมาจากเราเตอร์ใน area เดียวกัน เพราะว่าเราเตอร์ใน area เดียวกันใช้ข้อมูลร่วมกัน



รูปที่ 2.3 แสดง Autonomous system (AS)

การแบ่งพื้นที่สร้างความแตกต่างของ OSPF routing ขึ้นมาที่ขึ้นอยู่กับต้นทางและปลายทาง ดังนี้คือ Intra-area routing เกิดขึ้นเมื่อต้นทางและปลายทางอยู่ใน area เดียวกัน และ Inter-area routing เกิดขึ้นเมื่อต้นทางและปลายทางอยู่กันคนละ area

OSPF Backbone ตอบสนองต่อการกระจายข้อมูลการเลือกเส้นทาง (routing information) ระหว่าง area ประกอบด้วย backbone router จากภาพ เราเตอร์หมายเลข 4, 5, 6, 10, 11 และ 12 คือ Backbone Router

Backbone Router สร้าง OSPF area และใช้ procedure และ algorithm เดียวกันในการรักษา routing information ภายใน Backbone ที่เราเตอร์ควรจะเป็น

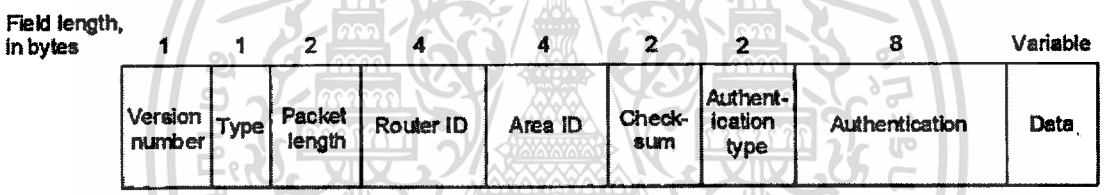
SPF Algorithm

SPF (Shortest Path First) Algorithm คือ พื้นฐานสำคัญของ OSPF เมื่อเราเตอร์ SPF เปิดเครื่องขึ้นมามันจะเริ่มต้นทำงาน protocol ของมัน หลังจากที่เราเตอร์รอจน อินเตอร์เฟซ ทำงานแล้ว เราเตอร์จะใช้ OSPF Hello Protocol ในการติดต่อกับเราเตอร์เพื่อนบ้าน กับเราเตอร์ที่ใช้อินเตอร์เฟซประเภทเดียวกัน เราเตอร์จะส่ง hello packet และรอรับคำตอบกลับเพื่อเป็นการหาเราเตอร์เพื่อนบ้าน ถ้า hello packet ยังคงทำงานส่งถึงกันอยู่แสดงให้ทราบว่าเราเตอร์อื่นๆ ก็ยังคงทำงานอยู่ได้แต่ถ้า hello packet ไม่ทำงานแสดงว่าเราเตอร์นั้นๆ อาจจะ ดาวน์ลงไป

ในกรณีที่เครือข่ายรองรับเราเตอร์มากกว่าสองตัว Hello Protocol จะเลือกเราเตอร์ให้ทำงานตัวหนึ่งและสำหรับการสำรองข้อมูลอีกตัวหนึ่ง สำหรับงานทุกๆอย่าง เราเตอร์ที่ได้รับเลือกจะตอบสนอง LSA ทุกๆอย่างภายในเครือข่ายนี้ เราเตอร์ที่ได้รับการเลือกนี้ทำให้ traffic การสื่อสารภายในเครือข่ายลดลงและขนาดของ topological database ก็ลดลงด้วยเช่นกัน

รูปแบบของแพ็กเก็ต OSPF (Packet Format)

ทุกๆ OSPF แพ็กเก็ตเริ่มต้นด้วยส่วนหัวของแพ็กเก็ตขนาด 24 ไบต์ ดังภาพ



รูปที่ 2.4 แสดง OSPF แพ็กเก็ต

รายละเอียดของแต่ละฟิลด์ สามารถอธิบายได้ดังต่อไปนี้

Version number – กำหนดให้รู้ว่าใช้ OSPF เวอร์ชั่นใดอยู่

Type – บอกให้รู้ว่า OSPF แพ็กเก็ตเป็นชนิดใดดังนี้

- **Hello** : เริ่มต้นและรักษาความสัมพันธ์กับเราเตอร์เพื่อนบ้าน

- **Database Description** : อธิบายเนื้อหาของ topological database

- **Link-state Request** : เป็นการร้องขอส่วนหนึ่งของ topological database จากเราเตอร์เพื่อนบ้าน

- **Link-state Update** : เป็นการตอบสนองต่อการร้องขอของ request packet

- **Link-state Acknowledgement** :เป็นการตอบกลับยืนยันการได้รับLink-state Update แพ็กเก็ต

Packet length – บอกความยาวของ OSPF แพ็กเก็ต รวมทั้งส่วนหัวของแพ็กเก็ตด้วย เป็นหน่วยไบต์

Router ID – บอกที่มาของแพ็กเก็ต

Area ID – แสดงให้ทราบว่าแพ็กเก็ตนั้นเป็นของ area ใด OSPF packet จะต้องเป็นของ area เดียว

Checksum – เอาไว้ใช้สำหรับการตรวจสอบข้อมูลทั้งหมดของ packet ว่าถูกต้องหรือไม่

Authentication type – สำหรับการระบุชนิดของการตรวจสอบ (Authentication) การใช้งาน OSPF Protocol ต้องมีการ Authentication เสมอ

Authentication – บรรจุข้อมูลของ authentication

Data – บรรจุข้อมูลที่ encapsulate แล้วสำหรับ Layer ที่สูงขึ้นไป

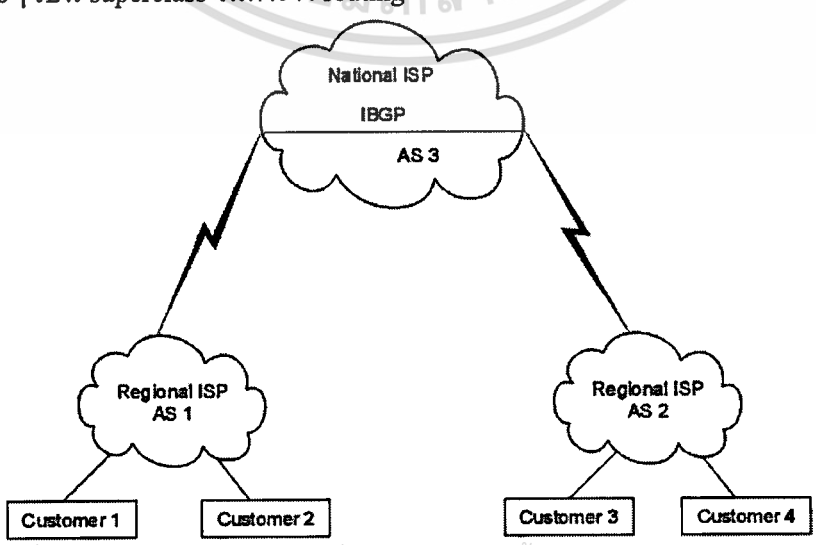
2.2.2.3) Border Gateway Protocol (BGP)

Border Gateway Protocol คือ Protocol ของ Inter Autonomous System โปรโตคอลการเลือกเส้นทางแบบ BGP ใช้ในการแลกเปลี่ยนข้อมูล routing สำหรับอินเตอร์เน็ต และคือโปรโตคอลที่ใช้สำหรับแลกเปลี่ยนข้อมูลระหว่าง Internet Service Provider (ISP) ส่วนเครือข่ายส่วนตัวอื่นๆ เช่น เครือข่ายของมหาวิทยาลัยและของบริษัทมักนิยมใช้ Interior Gateway Protocol (IGP) เช่น RIP หรือ OSPF สำหรับการแลกเปลี่ยน routing information ภายในเครือข่ายของตนเอง

ลูกค้าเชื่อมต่อกับ ISP และ ISP ใช้ BGP เพื่อแลกเปลี่ยนระหว่างลูกค้าและ ISP และเมื่อ BGP ถูกใช้ระหว่าง Autonomous System (AS) โปรโตคอลนั้นจะหมายถึง External BGP (EBGP) แต่ถ้าผู้ให้บริการใช้แลกเปลี่ยนข้อมูลกันภายใน AS โปรโตคอลนั้นหมายถึง Internal BGP (IBGP)

BGP เป็นโปรโตคอลที่มีความมั่นคงและสามารถปรับขยายได้ BGP สามารถรองรับตาราง routing ได้มากกว่า 90,000 เส้นทาง BGP ใช้ตัวแปรในการเลือกเส้นทางมากมายตัวแปรเหล่านั้นเราเรียกว่า attribute เพื่อใช้กำหนดนโยบายในการเลือกเส้นทางในสภาพแตกต่างกัน

ใน BGP มีคุณลักษณะที่เรียกว่า Classless Inter Domain Routing (CIDR) เพื่อใช้ในการรวม class ย่อยๆ เป็น superclass ในตาราง routing



รูปที่ 2.5 แสดง External และ Internal BGP

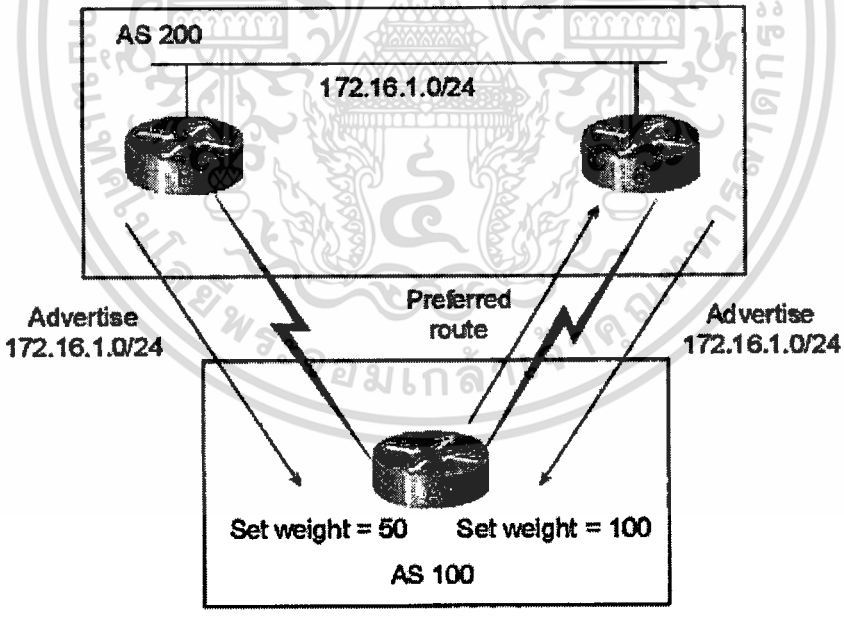
คุณสมบัติของ BGP (BGP Attributes)

การเลือกเส้นทางด้วยโปรโตคอล BGP มีคุณลักษณะร่วมกันที่ใช้ในการตัดสินใจเพื่อเลือกเส้นทางที่ดีที่สุดในการเลือกเส้นทางไปหาจุดหมายปลายทาง คุณลักษณะต่างๆ ที่ BGP ใช้ประกอบการเลือกเส้นทาง ประกอบด้วย

- Weight
- Local preference
- Multi-exit discriminator
- Origin
- AS_path
- Next hop
- Community

Weight Attribute

ถ้าเราเตอร์เรียนรู้เส้นทางในการส่งแพ็กเก็ตไปถึงปลายทางมากกว่าหนึ่งเส้นทาง การเลือกเส้นทางจะใช้ค่า weight เส้นทางที่มีค่า weight มากกว่าจะเป็นเส้นทางที่ถูกเลือก

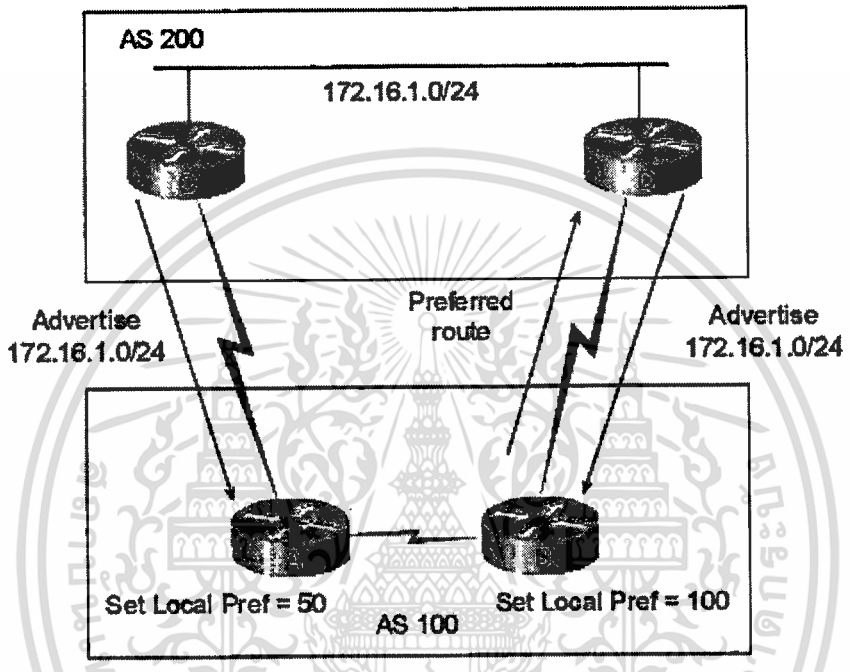


รูปที่ 2.6 แสดง BGP Weight Attribute

จากรูปที่ 2.6 ถ้าเราเตอร์ A เรียนรู้เส้นทางในการส่ง packet ไปยังเครือข่าย 172.16.1.0/24 มาสองเส้นทางคือได้รับการ Advertise มาจากเราเตอร์ B มีค่า weight เท่ากับ 50 และได้รับการ Advertise มาจากเราเตอร์ C มีค่า weight เท่ากับ 100 เราเตอร์จะเลือกเส้นทางที่มีค่า weight มากกว่าเส้นทางที่ผ่านเราเตอร์ C จะถูกเลือกใช้และบันทึกลงตาราง routing

Local preference Attribute

Local preference ใช้เป็นทางออกของ AS ไปยัง AS อื่น แต่ไม่เหมือนกับค่าของ weight เช่นถ้าเราเตอร์สองตัวได้รับการ Advertise ค่า Local preference มาตัวละหนึ่งค่าจาก AS อื่น เราเตอร์ทั้งสองตัวคือ เราเตอร์ A และ เราเตอร์ B จะแลกเปลี่ยนค่า Local preference ซึ่งกันและกัน เราเตอร์ตัวที่มีค่า Local preference สูงกว่าจะถูกเลือกเป็น Local preference ตามภาพเราเตอร์ B จะถูกเลือกให้เป็นทางออกไปยัง AS 200 เพราะค่า Local preference สูงกว่า



รูปที่ 2.7 แสดง BGP Local Preference Attribute

Multi-exit discriminator Attribute

ค่า **multi-exit discriminator** (MED) หรือเรียกว่าค่า metric attribute ใช้ในการสังเกตสำหรับ AS ภายนอกที่ต้องการเลือกเส้นทางมาสู่ภายใน AS ที่ได้ประกาศค่า metric นี้ออกไป การใช้คำว่าสังเกตเพราะว่าบางครั้ง AS ภายนอกอาจใช้ attribute อื่นๆ ในการเลือกเส้นทางมาสู่ AS นี้ก็ได้ เพราะอาจไม่ใช้ค่า metric

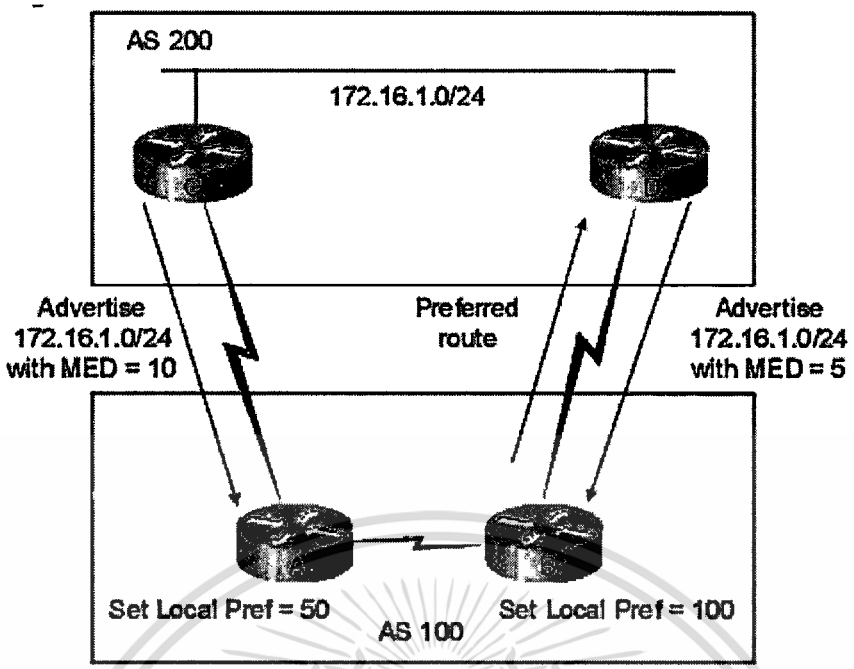
Origin Attribute

Origin Attribute เป็นคุณสมบัติที่ BGP ใช้ในการเลือกเส้นทางโดยเฉพาะ origin attribute ประกอบด้วย ค่าต่อไปนี้

- IGP - การเลือกเส้นทางสำหรับภายใน AS ค่านี้ถูกกำหนดเมื่อคำสั่งของเราเตอร์ ได้เปลี่ยนไปเป็น BGP

- EGP - การเลือกเส้นทางเรียนรู้ผ่าน Exterior Border Gateway Protocol (EBGP)

- Incomplete – จุดเริ่มต้นของเส้นทางไม่สามารถทราบได้หรือเรียนรู้ได้ด้วยวิธีใดๆ

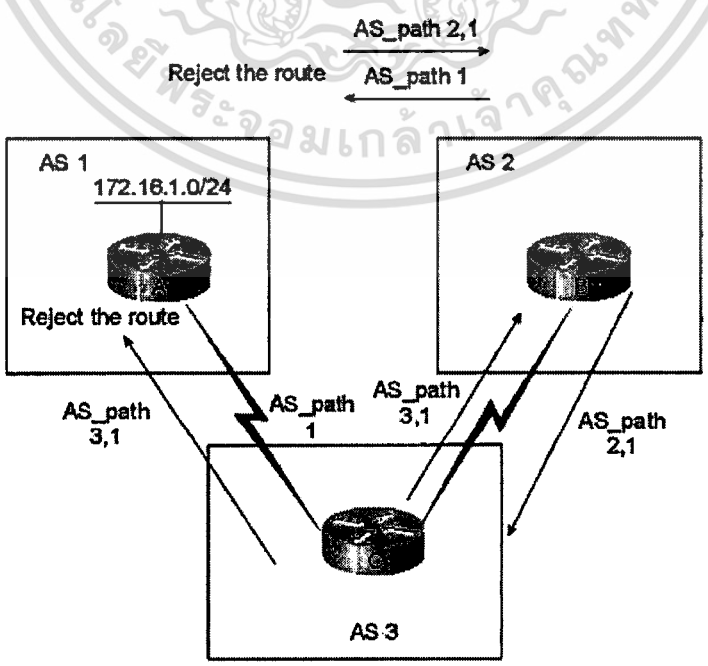


รูปที่ 2.8 แสดง BGP Multi-exit discriminator Attribute

จากรูปที่ 2.8 นั้นเราเตอร์ C และ D Advertise เส้นทางในการเข้าถึงเครือข่าย 172.16.1.0/24 โดยที่เราเตอร์ C Advertise ค่า MED = 10 ส่วนเราเตอร์ D Advertise ค่า MED = 5 ดังนั้น AS 100 จะส่ง packet ไปยัง AS 200 โดยการส่ง packet ผ่านไปยังเราเตอร์ D เพราะมีค่า MED น้อยกว่า

AS_path Attribute

เมื่อเส้นทางได้ประกาศผ่านไปยัง Autonomous System หมายเลข AS จะถูกเพิ่มเข้าไปเรียงลำดับในรายการของหมายเลข AS



รูปที่ 2.9 แสดง BGP AS-path Attribute

AS1 เป็นคนเริ่มต้นประกาศเส้นทางไปยังเครือข่าย 172.16.1.0/24 ไปยัง AS2 และ AS3 ว่า AS-path attribute เท่ากับ {1} , AS3 ประกาศกลับไปยัง AS1 ว่า AS-path attribute คือ {3,1} , และ AS2 ประกาศกลับไปหา AS1 ว่า AS-path attribute คือ {2,1} AS1 จะไม่ใช้เส้นทางนี้เส้นทางของตนเองถูกตรวจสอบว่าอยู่ในการประกาศเส้นทางของตนเอง นี่เป็นกลไกที่ BGP ใช้ในการทำไม่ให้เกิด loop ขึ้นในเครือข่าย AS2 และ AS3 ถ่ายทอดเส้นทางไปสู่กันและกันด้วยหมายเลข AS ของตนเองใน AS-path ของตนเอง เส้นทางนี้จะถูกใส่ลงไปใน routing table เพราะ AS2 และ AS3 เรียนรู้เส้นทางไปสู่เครือข่าย 172.16.1.0 ที่สั้นกว่ามาจาก AS1

Next-hop Attribute

BGP next-hop attribute คือ หมายเลข IP Address ที่ถูกใช้ในการประกาศให้เราเตอร์ส่งแพ็กเก็ตไปยังจุดหมาย สำหรับ EBGP ค่า next-hop address คือ หมายเลข IP Address ของการเชื่อมต่อระหว่าง peer

Community Attribute

community attribute จัดเตรียมเส้นทางของกลุ่มของปลายทาง เรียกว่า community ที่การตัดสินใจเลือกเส้นทางสามารถประยุกต์นำมาใช้ แผนที่การเลือกเส้นทางถูกใช้และกำหนดใน community attribute ค่าต่างๆ ของ community attribute มีดังนี้

- no-export : ไม่ต้องประกาศเส้นทางนี้ไปสู่ EBGP peers
- no-advertise : ไม่ต้องประกาศเส้นทางนี้ไปสู่ทุกๆ peer
- internet : ประกาศเส้นทางนี้ไปสู่ Internet community

2.3 ระบบปฏิบัติการ FreeBSD

ระบบปฏิบัติการ FreeBSD เป็นระบบปฏิบัติการที่มีเสถียรภาพอย่างสูง ระบบปฏิบัติการหนึ่ง สามารถนำมาประยุกต์ใช้งานได้ในหลายลักษณะ เช่น นำมาสร้างเป็น Web server, E-mail server, DNS server , รับการหมุนเข้าใช้งานจากโทรศัพท์ (Dial up), Database server และอื่นๆอีกมากมาย ในการศึกษานี้นำระบบปฏิบัติการ FreeBSD มาสร้างเป็น Router ด้วยโปรแกรม XORP (eXtensible Open Router Platform) นอกจากนี้ระบบปฏิบัติการ FreeBSD ยังมีความสามารถอีกหลากหลาย สามารถ Download ตัวระบบปฏิบัติการมาใช้งานได้โดยไม่เสียค่าใช้จ่ายในการใช้งาน และยังมีเอกสารประกอบการใช้งานได้ที่เว็บไซต์ http://www.freebsd.org ผู้ใช้งานสามารถปรับปรุงและใช้งานตัวระบบปฏิบัติการ FreeBSD ได้ภายใต้ลิขสิทธิ์แบบ BSD

2.3.1 ประวัติความเป็นมาของระบบปฏิบัติการ FreeBSD

ระบบปฏิบัติการ FreeBSD เริ่มต้นพัฒนามาจากระบบปฏิบัติการ Unix ระบบปฏิบัติการ Unix เวอร์ชันแรกเริ่มพัฒนาเมื่อปี ค.ศ. 1969 โดย Ken Thomson แห่ง Bell Laboratories

ระบบปฏิบัติการ Unix มาจากการเล่นคำที่คล้ายกับคำว่า MULTICS ที่เป็นชื่อโครงการระบบปฏิบัติการ Unix พัฒนาขึ้นมาจากภาษา C ในระยะแรกระบบปฏิบัติการ Unix ยังไม่แพร่กระจายออกสู่ภายนอก ยังคงพัฒนาอยู่ภายใน Bell Laboratories จนกระทั่งปี ค.ศ. 1976 ระบบปฏิบัติการ Unix เวอร์ชัน 6 จึงได้กระจายออกสู่มหาวิทยาลัย แต่ก็ยังเป็นส่วนน้อย ไม่มากเท่าไร

หลังจากการแพร่กระจายออกไปของ Unix ในปี ค.ศ. 1978 ได้เกิด Unix Support Group (USG) หลังจาก Unix เวอร์ชัน 7 ได้แพร่กระจายออกไป ระบบปฏิบัติการ Unix ได้กลายเป็นผลิตภัณฑ์ทางการค้าหลังจากที่ระยะแรก Unix เป็นเครื่องมือในการวิจัยแต่นักวิจัยก็ยังไม่ได้หยุดยั้งการวิจัย ยังคงทำการวิจัยในเวอร์ชันของเขาต่อไป จนออกเวอร์ชัน 8 ที่เพิ่ม Stream I/O System ทำให้เกิดความยืดหยุ่นในการปรับปรุง Kernel และในเวอร์ชัน 8 นี้ก็ได้เพิ่มเติม Remote File System คล้ายคลึงกับระบบของ Sun's NFS

หลังจากนั้น University of California at Berkeley เริ่มพัฒนาระบบปฏิบัติการ Unix ในปี ค.ศ. 1978 โดย Bill Joy และ Ozalp Babuaglu ภายใต้ชื่อ 3BSD Unix เวอร์ชันนี้เป็นเวอร์ชันแรกของระบบปฏิบัติการ Unix ทั้งหลาย การที่มี Virtual Memory ขนาดใหญ่ที่ให้สามารถพัฒนาโปรแกรมขนาดใหญ่ได้ จนกระทั่งโครงการ Defense Advance Research Projects Agency (DARPA) ได้ให้ทุนกับ Berkeley เพื่อพัฒนาระบบปฏิบัติการ Unix มาตรฐานเพื่อให้รัฐบาลใช้ จนสุดท้ายได้เกิด 4BSD

FreeBSD เริ่มต้นพัฒนาจาก code ของ 4.4 BSD-Lite เป้าหมายของโครงการระบบปฏิบัติการ FreeBSD ก็เพื่อการสร้างระบบปฏิบัติการที่ใช้งานได้ในหลายๆวัตถุประสงค์ เพื่อให้สามารถสร้างผลกำไรได้สูงสุด ระบบปฏิบัติการ FreeBSD สามารถทำงานได้บนหลายๆ Platform เช่น Intel platforms, Alpha platforms, AMD CPU และ Ultra SPARC

2.3.2 หลักการออกแบบระบบของระบบปฏิบัติการ FreeBSD

ระบบปฏิบัติการ Unix ใช้หลักการ Time-sharing ใช้ระบบ User Interface แบบ shell ระบบไฟล์ข้อมูลเป็นแบบแผนภูมิต้นไม้ (Multilevel Tree) ที่ยินยอมให้ผู้ใช้สามารถสร้าง subdirectory ของตนเอง , Disk file , I/O และ Device Driver ของระบบจัดเก็บไว้ใน Kernel ของระบบปฏิบัติการเอง การบูตระบบ ระบบจะโหลด Kernel เพื่อใช้ในการควบคุมฮาร์ดแวร์อินเทอร์เฟส และการควบคุมหน่วยความจำของเครื่อง ดังนั้นจึงสามารถปรับปรุงระบบปฏิบัติการได้โดยการปรับปรุง Kernel ของระบบ, ระบบปฏิบัติการ Unix สนับสนุน Multiple Processes , process หนึ่งสามารถสร้าง process ใหม่ได้ ระบบ FreeBSD ใช้คำสั่ง paging เป็นกลไกในการควบคุมการจัดการหน่วยความจำและ ตารางการตัดสินใจของซีพียู

ระบบปฏิบัติการ Unix ถูกนำมาใช้งานในหลายด้าน เช่น การใช้งานด้านระบบเครือข่าย, การใช้งานด้านกราฟิก, และการทำงานแบบ real time โดยเฉพาะการใช้งานทางด้านเครือข่ายจะได้รับความนิยมมากจะเห็นได้ว่าเครื่อง server ของหน่วยงานต่างๆ นิยมใช้ระบบปฏิบัติการ Unix เป็นระบบปฏิบัติการของเครื่องเพราะความสามารถทางด้านความมีเสถียรภาพ, ความปลอดภัยจากการถูกโจมตี เป็นต้น

จากที่กล่าวมา ระบบปฏิบัติการ FreeBSD แท้จริงก็คือระบบปฏิบัติการ Unix ที่ได้รับการพัฒนามาจาก code โปรแกรมของ 4.4BSD-Lite ซึ่งโครงสร้างของ 4.4BSD Layer structure แสดงรูปที่ 2

ระบบปฏิบัติการ Unix ประกอบด้วยส่วนประกอบสองส่วนคือ Kernel และ System program จากภาพทุกสิ่งที่อยู่ภายใต้ system-call interface และอยู่เหนือ physical hardware คือ Kernel , Kernel ทำหน้าที่จัดเตรียมระบบไฟล์, ตารางการทำงานของ CPU, การจัดการหน่วยความจำ และฟังก์ชันการใช้งานอื่นๆของระบบปฏิบัติการ

2.4 โปรแกรมสำหรับจัดการ function ของเราเตอร์ XORP (eXtensible Open Routing Platform)

เราเตอร์นอกจากจะใช้ชนิดแบบฮาร์ดแวร์ที่ผู้ผลิต เช่น Cisco ผลิตจำหน่ายแล้วยังสามารถสร้างเราเตอร์เพื่อใช้งานได้จากโปรแกรมแบบ open source ได้จากหลายๆ โปรแกรม เช่น Zebra, MRTd, GateD ในการศึกษานี้ได้ใช้โปรแกรมเราเตอร์XORP (eXtensible Open Routing Platform) ในการสร้างเราเตอร์ดังกล่าว

ในเอกสารฉบับนี้ได้ทำการศึกษาสถาปัตยกรรมของ XORP และการใช้งานโปรแกรม XORP ในรูปแบบของการทำงานบนระบบปฏิบัติการ FreeBSD สำหรับ XORP นั้นประกอบด้วยโปรแกรมสำคัญคือ control plane software: เช่นโปรแกรม Routing Information Base (RIB), firewall management, command-line interface, network management และสำหรับเราเตอร์สมัยใหม่จะมีโปรแกรม address management เพิ่มเข้ามาอีกด้วย

XORP คือ โปรแกรมเราเตอร์ที่สามารถทำงานกับเราท์ติงโปรโตคอล ได้หลายโปรโตคอล คือ

Routing Protocols:

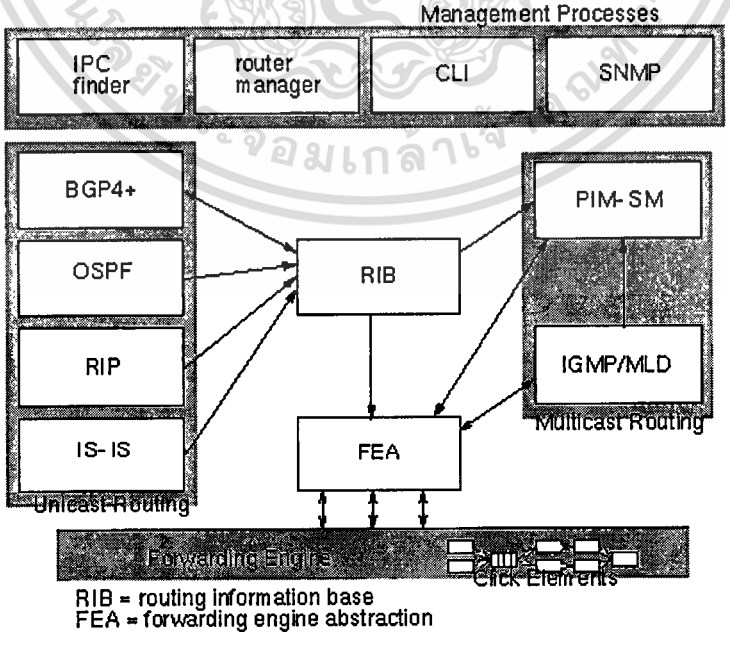
- BGP (Border Gateway Protocol) เป็นโปรโตคอลเลือกเส้นทางประเภท Exterior Gateway Routing ซึ่งมีความเป็นอิสระในการเลือกเส้นทางระหว่างเราเตอร์ และเครือข่ายที่อยู่ต่างโดเมนกันบนอินเทอร์เน็ต

- OSPF4+ (Open Shortest Path First) คือ โปรโตคอลเลือกเส้นทางที่ใช้สถานะ การเชื่อมต่อและตรวจหาเส้นทางที่สั้นที่สุดของเราเตอร์
- RIP (Routing Information Protocol) คือ โปรโตคอลที่ใช้อัลกอริทึม คำนวณหาเส้นทางจากระยะทางเพื่อเปรียบเทียบหาเส้นทางที่ดีที่สุด
- IS-IS (Intermediate System-to-Intermediate System) คือ โปรโตคอลสถานะการเชื่อมต่อ OSI ที่ส่งข้อมูลการเชื่อมต่อไปตามลำดับชั้น หลักการทำงานคล้ายกับ OSPF
- PIM-SM (Protocol Independent Multicast-Sparse Mode) คือ โปรโตคอลสำหรับการหาเส้นทางแบบกลุ่ม มัลติคาสต์ สำหรับเครือข่าย wan และ inter- domain
- IGMPv3/MLD (Internet Group Management Protocol version 3/ Multicast Listener Discovery) คือ โปรโตคอลที่ใช้ใน IP รุ่นที่ 4 ใช้สำหรับรายงานหมายเลข IP แบบมัลติคาสต์ ไปยังเราเตอร์แบบมัลติคาสต์ข้างเคียง

ส่วนโปรแกรมอินเตอร์เฟสในการติดต่อกับเราเตอร์นั้น XORP ได้เตรียมไว้ดังนี้

Management Interfaces:

- SNMP (Simple Network Management Protocol) คือ โปรแกรมที่ออกแบบให้แลกเปลี่ยนข้อมูลการจัดการระหว่างอุปกรณ์เครือข่าย
- Cisco-compatible command interface over ssh and telnet คือ การสั่งงานโปรแกรมเราเตอร์โดยการเชื่อมต่อผ่านโปรแกรม ssh (Secure Shell) และโปรแกรม Telnet
- CLI (Command Line Interface) คือ การสั่งงานโปรแกรมเราเตอร์ XORP โดยใช้คำสั่งแบบตัวอักษร



รูปที่ 2.10 XORP Process Model

จากรูปที่ 2.10 เป็นสถาปัตยกรรมของ XORP เราจะเห็นว่าตัวโปรแกรม RIB (Routing Information Base) ทำหน้าที่เป็นตัวกลางเชื่อมโยงระหว่างโปรโตคอลต่างๆ เช่น โปรโตคอล BGP และโปรโตคอล OSPF ได้รับข้อมูล routing จาก remote route ทั้งสองโปรโตคอลส่งข้อมูลมาที่โปรแกรม RIB โปรแกรม RIB ประมวลผลและหาเส้นทางที่ดีที่สุดแล้วตัดสินใจในการเลือกโปรโตคอลเพื่อส่ง packet ไปสู่ subnet ที่เป็นจุดหมายปลายทาง ดังนั้น RIB เปรียบเสมือนศูนย์กลางของระบบที่จะดูแลทุกเส้นทางการ route ของทุกโปรโตคอล RIB จึงเป็นโปรเซสศูนย์กลาง RIB เป็นผู้ตัดสินชี้ขาดและต้องดูแลทุกโปรโตคอลที่ผ่านเข้ามายัง router software โดย RIB จะต้องใช้เงื่อนไขของแต่ละโปรโตคอล เช่น metric, cost, distance, bandwidth เพื่อพิจารณาเลือกเส้นทางในการส่ง

FEA (Forwarding Engine Abstraction) เป็นโปรแกรมสำหรับจัดเตรียมการติดต่อกับ forwarding engine หรือช่องทางการสื่อสารของเครื่อง เพื่อการส่ง packet โดย FEA จะรับ RIB routing information โดยตรงมาจาก RIB ส่วนตัว PIM-SM (Protocol Independent Multicast-Spare Mode) และ IGMP ทำหน้าที่จัดเตรียม multicast routing function PIM หาเส้นทางโดยการช่วยเหลือและการให้ข้อมูลจาก IGMP โดยที่การหาเส้นทางของ PIM จะส่งผ่านไปยัง FEA โดยตรงโดยไม่ต้องผ่านไปที่ RIB สำหรับรูปที่แสดงให้เห็นนั้นเราสามารถแบ่ง protocol ออกได้สองประเภทคือ

Unicast Routing ประกอบด้วย โปรโตคอล BGP4+, OSPF, RIP และ IS-IS

Multicast Routing ประกอบด้วย โปรโตคอล PIM-SM, IGMP/MLD

โปรแกรม Router Manager ทำหน้าที่ควบคุมการทำงานของเราเตอร์, การปรับแต่งค่า configuration ของเราเตอร์ ควบคุมการเริ่มต้นหรือการหยุดการทำงานของโปรโตคอลและฟังก์ชั่นอื่นๆ ของเราเตอร์ นอกจากนี้ยังทำหน้าที่ในการซ่อนโครงสร้างที่อยู่ภายในของเราเตอร์ออกจากผู้ใช้งานที่ไม่จำเป็นต้องทราบถึงโครงสร้างการทำงานภายใน

2.5 การใช้งานเราเตอร์ XORP

อุปกรณ์เราเตอร์ XORP สร้างขึ้นจากการนำ source codeโปรแกรมเราเตอร์ XORP (http://www.xorp.org) มาแปล (compile) บนระบบปฏิบัติการประเภท Unix เช่น FreeBSD และ Linux เป็นต้น เมื่อคอมไพล์เสร็จแล้วและปรับแต่งคอนฟิกูเรชั่นไฟล์ สามารถทำหน้าที่เป็นอุปกรณ์ router ได้ โดยผู้ใช้สามารถสั่งงานโดยการสั่งงานแบบตัวอักษร (command line) ผ่านโปรแกรมที่เรียกว่า xorpsh ซึ่งคำสั่งในการสั่งงานดังกล่าวและการปรับแต่งคอนฟิกูเรชั่นไฟล์ไม่สะดวกในการควบคุม โปรแกรมเราเตอร์ XORP เพราะอุปกรณ์ เราเตอร์ XORP นั้นเป็นโปรแกรมที่ค่อนข้างใหม่ยังไม่มีผู้ใช้งานอย่างแพร่หลายเช่น เราเตอร์ที่จำหน่ายในท้องตลาดปัจจุบัน ดังนั้นในบทนี้จึง

นำเสนอการออกแบบการสร้างการติดต่อกับผู้ใช้แบบเว็บสำหรับอุปกรณ์เราเตอร์XORP เพื่อให้การควบคุมและปรับแต่งค่าต่างๆของอุปกรณ์เราเตอร์ XORP เป็นไปด้วยความสะดวกมากยิ่งขึ้น

2.5.1 ภาพรวมการทำงานของระบบ

รูปแบบการจัดการและสั่งงาน XORP สามารถทำได้ สองวิธี วิธีแรกคือการเข้าไปแก้ไขคอนฟิกูเรชั่นไฟล์ (configuration file) ส่วนวิธีที่สองคือการใช้ Command line ในตัวโปรแกรมจัดการฟังก์ชั่นเราเตอร์ XORP นั้นใช้โปรแกรมที่เรียกว่า xorpsh (Xorp command shell) เพื่อการสั่งงาน

2.5.1.1) การจัดการโปรแกรมเราเตอร์ XORP โดยการเข้าไปแก้ไขคอนฟิกูเรชั่นไฟล์

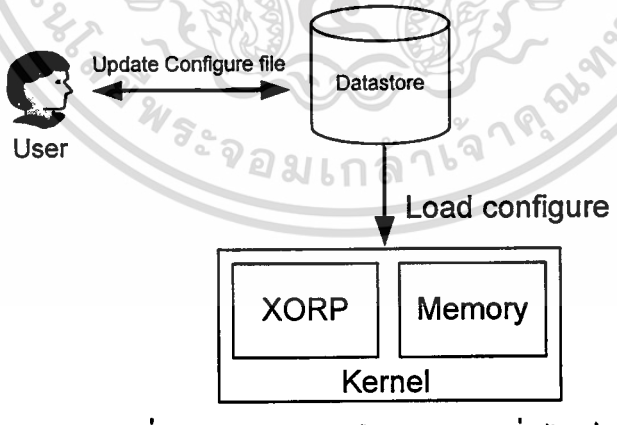
เป็นวิธีที่ง่ายและไม่ยุ่งยากเพียงแต่ผู้ใช้ต้องรู้โครงสร้างของไฟล์ โดยเฉพาะกับผู้ที่คุ้นเคยกับระบบปฏิบัติการแบบ Unix จะสามารถใช้วิธีนี้ได้อย่างไม่ยาก ตำแหน่งของไฟล์ config.boot อยู่ที่ตำแหน่งดังนี้

/usr/local/xorp/rtrmgr/config.boot

แต่ค่าเดิมหลังจากที่ได้แปล (compile) บนระบบปฏิบัติการ FreeBSD นั้นจะเป็นไฟล์ชื่อ config.boot.sample ให้ copy เป็นไฟล์ชื่อ config.boot แก้ไขไฟล์ตามความต้องการของระบบ และสั่งเปลี่ยนไฟล์คอนฟิกโดยสั่ง

#./ xorp_rtrmgr –b config.boot

เพื่อให้ตัวโปรแกรมเราเตอร์XORPได้รู้ว่าคอนฟิกูเรชั่นไฟล์ของโปรแกรมเปลี่ยนเป็นไฟล์ชื่อ config.boot



รูปที่ 2.11 แสดงการแก้ไขคอนฟิกูเรชั่นไฟล์

ตัวอย่างของไฟล์ config.boot

```
interfaces {
  restore-original-config-on-shutdown: false
  interface dc0 {
    description: "data interface"
    disable: false
    /* default-system-config */
    vif dc0 {
      disable: false
      address 10.10.10.10 {
        prefix-length: 24
        broadcast: 10.10.10.255
        disable: false
      }
      /*
      address 10:10:10:10:10:10:10:10 {
        prefix-length: 64
        disable: false
      }
      */
    }
  }
}
```

รูปที่ 2.12 แสดงตัวอย่างไฟล์ config.boot

2.5.1.2) การสั่งงานเราเตอร์ XORP ผ่าน Command line ที่เรียกว่า xorpsh

การจะใช้งานคำสั่ง xorpsh ของโปรแกรมเราเตอร์ XORP ได้ต้องสั่งให้ตัวเราเตอร์เมเนเจอร์ของ XORP ทำงานเป็นเบื้องหลังก่อนโดยสั่งได้ดังนี้

```
# cd /usr/local/xorp/rtrmgr
#./xorp_rtrmgr
```

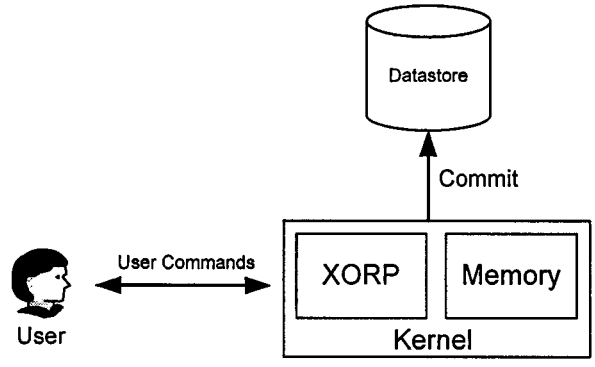
หลังจากนั้นก็เรียกโปรแกรม xorpsh โดย

```
#./xorpsh
```

นอกจากนี้ยังสามารถเรียกใช้คำสั่งของ xorpsh ได้โดยการเขียนเป็น shell script ผ่าน Unix ได้ดังนี้

```
#!/bin/sh
xorpsh <<!
show host os
!
```

เมื่อผู้ใช้งานสั่งงานผ่าน Command line ในกรณีที่เป็นคำสั่งในการเปลี่ยนแปลงค่า เช่น ค่าของ Interface เป็นต้น จากนั้นตัวโปรแกรม ก็จะเข้าอัพเดทค่าในคอนฟิกูเรชั่นไฟล์ในหน่วยความจำสำรองและถ้าต้องการเขียนไฟล์ก็จะมีคำสั่ง commit แต่ถ้าไม่ต้องการเขียนไฟล์ก็สั่ง exit discard

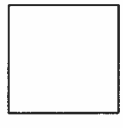
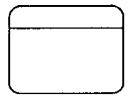
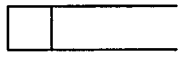


รูปที่ 2.13 แสดงการจัดการระบบโดยใช้ Command line

การใช้งานเราเตอร์ XORP ในโครงการพัฒนาระบบงานนี้ต้องศึกษาโปรโตคอลการเลือกเส้นทางที่ใช้งานเป็นมาตรฐานเช่น Static Routing และ Dynamic Routing แบบต่างๆ เช่น Rip, OSPF , BGP เพื่อให้การใช้งานเราเตอร์ได้อย่างมีประสิทธิภาพและสามารถเลือกเส้นทางได้อย่างถูกต้อง การตั้งค่า configuration ของเราเตอร์ XORP สามารถทำได้สองแบบคือ การใช้คำสั่งแบบ Command Line (xorpsh) และการปรับตั้งค่าใน configuration file

2.6 สัญลักษณ์ของผังการไหลของข้อมูล

การวิเคราะห์งานที่ใช้ในโครงงานนี้ได้ใช้การวิเคราะห์งานเชิงโครงสร้าง เนื่องด้วยการเขียนโปรแกรมเว็บเบสทั้งหมดได้เขียนโปรแกรมแบบโครงสร้างและใช้การบันทึกข้อมูลตัวแปรทั้งหมดลงไฟล์ข้อมูล (configuration file) ซึ่งสัญลักษณ์ที่ใช้ในการวิเคราะห์ระบบทั้งหมดสามารถแสดงได้ดังต่อไปนี้

สัญลักษณ์	คำอธิบาย
 Entity	คือ หน่วยภายนอก (External Entity)
 Process	คือ กระบวนการหรือขั้นตอนการทำงาน (Process)
 Data store	คือ ข้อมูลที่จัดเก็บ (Data store)
 Flow of Data	คือ การไหลของข้อมูล (Flow of data)

ความหมายของแต่ละ Entity

หมายถึง ผู้ใช้ระบบ Web-based User Interface ที่มีสิทธิ์ในการแก้ไขค่า Configuration ของระบบในการพัฒนาระบบนี้ใช้ชื่อ admin

หมายถึง อุปกรณ์เราเตอร์ XORP ที่ทำหน้าที่เป็นเราเตอร์ และเว็บเซอร์เวอร์

บทที่ 3
การวิเคราะห์และออกแบบโครงงานพัฒนาระบบ

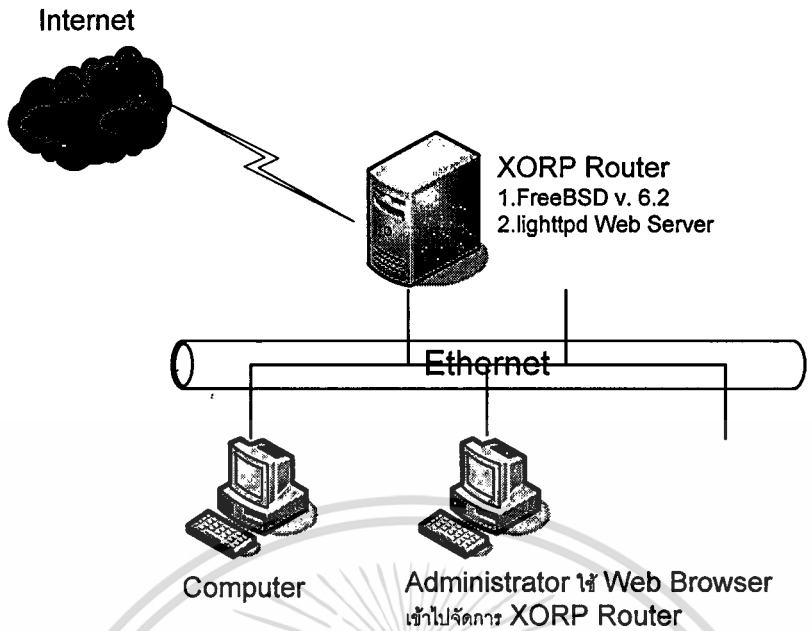
ในบทนี้จะกล่าวถึงการออกแบบการใช้งานเราเตอร์ XORP เป็นโปรแกรมจัดการฟังก์ชันเราเตอร์บนระบบปฏิบัติการ FreeBSD ในระบบเดิมโดยจะกล่าวถึงการตั้งค่าจากการใช้คอนฟิกไฟล์ของระบบเดิมและรูปแบบการใช้งาน command line อีกทั้งจะนำเสนอการออกแบบโครงงานว่ามีรูปแบบการทำงานอย่างไรเพื่อให้สามารถมาทดแทนการใช้งานระบบเดิมได้

3.1 โครงงานการพัฒนาเว็บเบสยูสเซอร์อินเตอร์เฟสสำหรับเราเตอร์ XORP

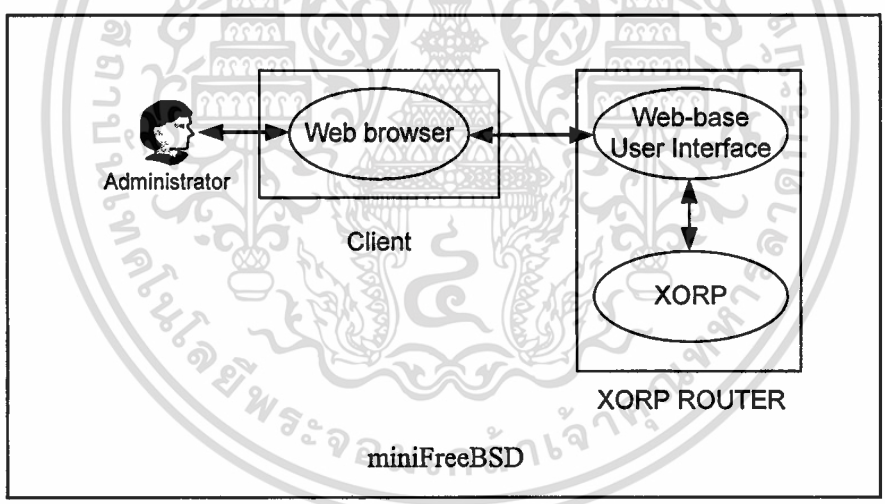
3.1.1 ขอบเขตและวัตถุประสงค์ในการพัฒนาระบบ

การพัฒนาระบบ Web-based User Interfaceสำหรับอุปกรณ์เราเตอร์ XORP เป็นการพัฒนาระบบที่มุ่งหวังเพื่อการควบคุมอุปกรณ์เราเตอร์ XORP ให้สามารถควบคุมได้โดยการใช้เว็บบราวเซอร์ (Web browser) จากเครื่อง client โดยไม่ต้องสั่งงานที่เครื่องเราเตอร์ XORP การพัฒนาระบบนี้ทำดังต่อไปนี้ คือ การใช้เครื่องคอมพิวเตอร์ส่วนบุคคลทำหน้าที่รันโปรแกรม เราเตอร์ XORP (eXtensible Open Router Platform) และโปรแกรมเว็บเซอร์เวอร์ โดยโปรแกรม Web-based User Interface จะสามารถเข้าไปควบคุมการทำงานของ XORP Router ได้ จากการเข้าไปแก้ไขไฟล์ config.boot ของโปรแกรมเราเตอร์ XORP และสั่งงานได้จากการใช้คำสั่งผ่านหน้าเว็บที่เครื่อง client

เมื่อผู้ใช้ต้องการควบคุมเราเตอร์ XORP สามารถทำได้โดยการใช้เครื่อง client ใดก็ได้เปิดโปรแกรม Web browser มายัง URL ของเครื่องเราเตอร์ จะพบหน้าจอ Web-based User Interface ของอุปกรณ์เราเตอร์ XORP จากนั้นผู้ใช้ก็สามารถควบคุมโดยไม่ต้องจดจำคำสั่งแบบ command line ของ XORP เพียงแต่ผู้ใช้จะต้องกำหนดค่าตามที่ต้องการปรับแต่งระบบลงไปเท่านั้น ซึ่งเป็นการใช้งานที่ง่ายกว่าการใช้คำสั่งแบบตัวอักษร(command line) ดังรูปที่ 3.1 เป็นการแสดงให้เห็นถึงสถาปัตยกรรมของระบบที่ user หรือ admin สั่งงานผ่านเครือข่ายคอมพิวเตอร์เข้ามายังอุปกรณ์เราเตอร์ XORP ที่ได้พัฒนาขึ้น ดังนั้นผู้จัดการเราเตอร์ XORP ไม่จำเป็นที่จะต้องทำงานที่เครื่องเราเตอร์ได้ก็สามารถที่จะจัดการปรับแต่งระบบเราเตอร์ได้อย่างง่ายโดยผ่านเทคโนโลยีเว็บและเทคโนโลยีเครือข่ายคอมพิวเตอร์



รูปที่ 3.1 แสดงรูปแบบการเชื่อมต่อโดยรวมของระบบ



รูปที่ 3.2 แสดงองค์ประกอบการทำงานของระบบ

การพัฒนาระบบ Web-based User Interface พัฒนาโดยใช้โปรแกรมจัดการฟังก์ชันเราเตอร์ XORP ทำงานอยู่บนระบบปฏิบัติการ FreeBSD และพัฒนาโปรแกรม Web-based User Interface เพื่อการควบคุมเราเตอร์ XORP มีขอบเขตการทำงานดังนี้คือ

1) Routing Protocol ที่นำมาใช้ในการพัฒนาโครงงานนี้คือ Routing Protocol แบบ **Static Routing** และแบบ Dynamic Routing คือ **RIP** และ **OSPF**

2) ภาษา script ที่ใช้ในการพัฒนาเว็บเบส คือ ภาษา PHP, perl , java script และ shell script

3) ระบบปฏิบัติการที่ใช้ในการพัฒนาระบบ คือ ระบบปฏิบัติการ FreeBSD เวอร์ชั่น 6.2

4)โปรแกรมอื่นๆ ที่ใช้ คือ โปรแกรม Web Server lighttpd เวอร์ชั่น 1.4.15 ทำหน้าที่เป็นเว็บเซอร์เวอร์

5) โปรแกรมแปลภาษา PHP เวอร์ชัน 4.4.7

3.1.2 วัตถุประสงค์ของการพัฒนาระบบ

เนื่องจากอุปกรณ์เราเตอร์ XORP ที่พัฒนาจากการนำโปรแกรมเราเตอร์ มา compile บนระบบปฏิบัติการตระกูล Unix เช่น FreeBSD และ Linux การควบคุมและสั่งงานเราเตอร์ XORP ผู้ใช้สามารถสั่งงานโดยใช้คำสั่งแบบตัวอักษร (command line) ก่อให้เกิดการใช้งานที่ไม่สะดวก ดังนั้นการพัฒนา Web-based user interface จึงมีวัตถุประสงค์ในการพัฒนาดังนี้

1) เพื่อสามารถควบคุมอุปกรณ์เราเตอร์ XORP ได้ง่ายขึ้นเพราะใช้การสั่งงานผ่านรูปแบบเว็บซึ่งผู้ใช้คุ้นเคย

2) เพื่อสามารถควบคุมอุปกรณ์เราเตอร์ XORP ได้จากที่ใดก็ได้บนระบบเครือข่ายอินเทอร์เน็ต (Internet) และเครือข่ายภายใน (Local Area Network)

3) เพื่อให้เกิดการใช้งานอุปกรณ์เราเตอร์ XORP อย่างแพร่หลาย

4) เพื่อให้เข้าใจการทำงานของเราเตอร์ XORP

5) เพื่อเรียนรู้การออกแบบวิเคราะห์ ตลอดจนการพัฒนาโปรแกรมเพื่อจัดการ Application ผ่านทางเว็บ

3.2 การวิเคราะห์และการออกแบบระบบ

การออกแบบระบบในโครงงานนี้ใช้การออกแบบเชิงโครงสร้าง ประกอบด้วยแผนภาพกระแสข้อมูล (Dataflow Diagram) แบบต่างๆ คือ Context Diagram , Diagram0 และ Diagram1 และ state diagram

3.2.1 การวิเคราะห์ คุณสมบัติของระบบและความต้องการของระบบ

โครงงานการพัฒนาอุปกรณ์เราเตอร์โดยใช้ XORP พร้อมการเชื่อมต่อกับผู้ใช้แบบเว็บเบสยูสเซอร์อินเตอร์เฟส เป็นโครงงานที่พัฒนาบนระบบปฏิบัติการ FreeBSD และโปรแกรมจัดการฟังก์ชันเราเตอร์ XORP โดยในการพัฒนาโครงงานนี้ใช้เว็บเบสยูสเซอร์อินเตอร์เฟสที่พัฒนาจากภาษา PHP , Java script และนำระบบที่พัฒนาทั้งหมดบรรจุลงบนสื่อบันทึกข้อมูลประเภท compact flash โปรแกรมที่พัฒนาขึ้นนั้นทำงานบนเว็บเซอร์เวอร์คือ lighttpd และใช้เว็บบราวเซอร์เพื่อติดต่อตั้งค่า และติดต่อดูสถานะกับระบบเราเตอร์ XORP ที่ทำงานอยู่บนระบบปฏิบัติการ FreeBSD บนพื้นฐานความต้องการและฟังก์ชันการทำงานดังต่อไปนี้

3.2.1.1 Functional Requirements

1) สามารถป้อนข้อมูลที่จำเป็นสำหรับการกำหนดค่า Configuration ต่างๆ ของเราเตอร์ XORP ได้สำหรับแต่ละ Interfaces และโปรโตคอลการเลือกเส้นทางต่างๆ

2) ระบบสั่งงานอุปกรณ์เราเตอร์ XORP ผ่านทางหน้าจอเว็บบราวเซอร์ การสั่งงานสามารถทำได้ผ่านเครือข่ายคอมพิวเตอร์โดยใช้โปรโตคอลสื่อสาร TCP/IP และ HTTP

3) ระบบตรวจสอบสิทธิ์การเข้าใช้งานของผู้ใช้ ผู้ใช้ที่มีสิทธิในการติดต่อ กับเว็บเบ๊สคือ Administrator (admin)

4) ระบบเพิ่มและแก้ไขค่าของ Interfaces ต่างๆ ที่มีอยู่ในตัวอุปกรณ์เราเตอร์ XORPได้และสามารถเพิ่มค่าใหม่เข้าสู่ระบบได้

5) ระบบปรับแต่งค่า routing protocols ทั้งแบบ static และแบบ dynamic ได้ เช่น Static, Rip และ OSPF ของเราเตอร์ XORP ได้

6) ระบบสั่งการทำงานของเราเตอร์ XORP ได้ เช่น การเริ่มต้นและการหยุดการทำงานของแต่ละ Routing Protocols, การเริ่มและการหยุดการทำงานของเราเตอร์ XORP ทั้งระบบ

7) ระบบมีผลทันทีที่มีการปรับแต่ง configuration ของโปรโตคอลการเลือกเส้นทางและระบบมีผลเมื่อเริ่มระบบใหม่ที่มีการปรับแต่ง configuration ของโปรโตคอลการเลือกเส้นทาง

8) ระบบสามารถ Backup ไฟล์ configuration ของเราเตอร์ได้

3.2.1.2 Non- Functional Requirements

1) ระบบสามารถแสดงสภาวะการทำงานต่างๆ ของเราเตอร์ XORP เช่น การใช้งาน CPU , ปริมาณ packet ที่ผ่านเข้าออก ข้อมูลแสดงการเราท์เส้นทางของเราเตอร์ เป็นต้น

2) สามารถปิดระบบ (เราเตอร์ XORP)ได้โดยไม่ต้องสั่ง shutdown เครื่อง

3.2.2 ขอบเขตของเราติงโปรโตคอลของ XORP ที่ใช้ในการพัฒนาระบบ

เราติงโปรโตคอลสำหรับการเลือกเส้นทางที่ใช้ในการพัฒนาระบบงานนี้ได้เลือกใช้เราติงโปรโตคอลที่นำมาพัฒนาเป็นเว็บเบสยูสเซอร์อินเตอร์เฟสอยู่สี่แบบคือ

1) Static Routing Protocol

2) RIP Routing Protocol

3) OSPF Routing Protocol

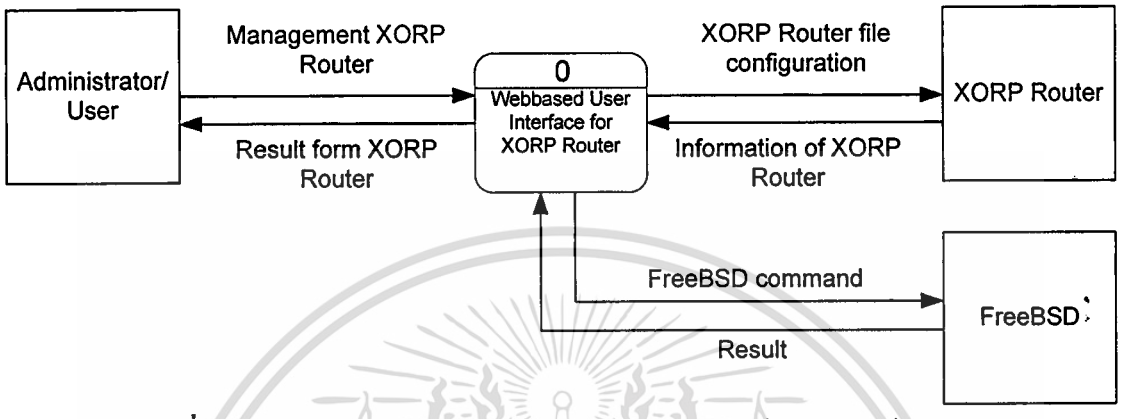
4) BGP Routing Protocol

ส่วนเราติงโปรโตคอลอื่นนอกเหนือจากที่กล่าวมาข้างต้นที่ XORP สามารถจัดการได้นั้น ผู้พัฒนาไม่ได้นำมาทำเป็นเว็บเบสยูสเซอร์อินเตอร์เฟส

3.2.3 การวิเคราะห์การไหลของข้อมูล (Dataflow Diagram)

ระบบ Web-based User Interface for XORP Router สามารถวิเคราะห์การไหลของข้อมูลได้ออกเป็น Diagram ต่างๆ ตั้งแต่ Context Diagram , Diagram0 และ Diagram1 ได้ดังต่อไปนี้

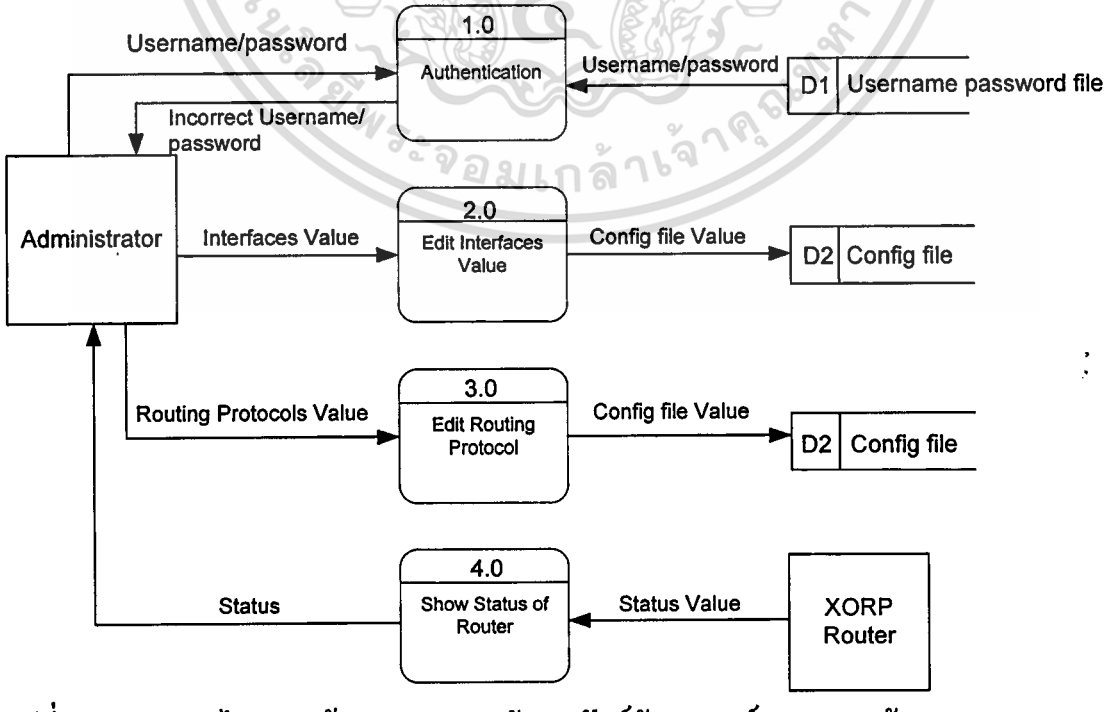
1) Context Diagram ของระบบ สามารถแสดงได้ดังรูปที่ 3.3



รูปที่ 3.3 แสดง Context Diagram ของระบบจัดการฟังก์ชันเราเตอร์ XORP

จากรูปที่ 3.3 แสดงการใช้งานของระบบที่ Administrator หรือ User สามารถใช้งานระบบ Web-based User Interface for XORP Router เข้าไปจัดการฟังก์ชันเราเตอร์ XORP และระบบจะเข้าไปแก้ไข Configuration file ของ XORP Router

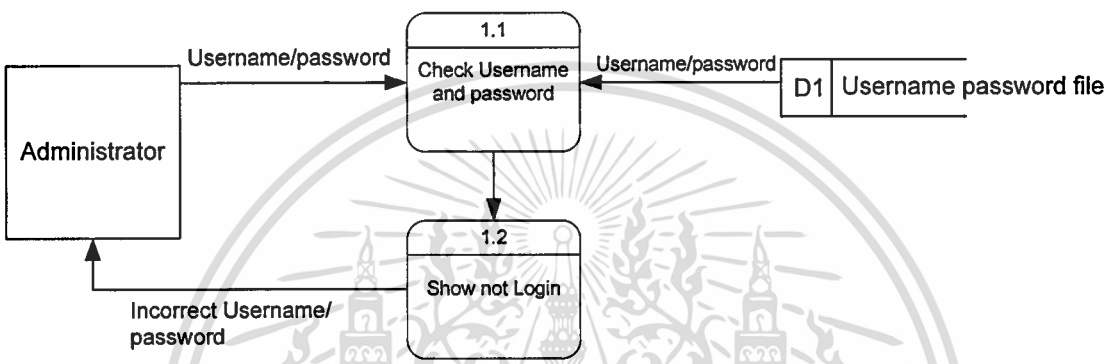
2) Dataflow Diagram Level 0 ของระบบสามารถแสดงได้ดังรูปที่ 3.7



รูปที่ 3.4 แสดงการไหลของข้อมูลของระบบจัดการฟังก์ชันเราเตอร์ XORP ระดับ 0 (DFD Level 0)

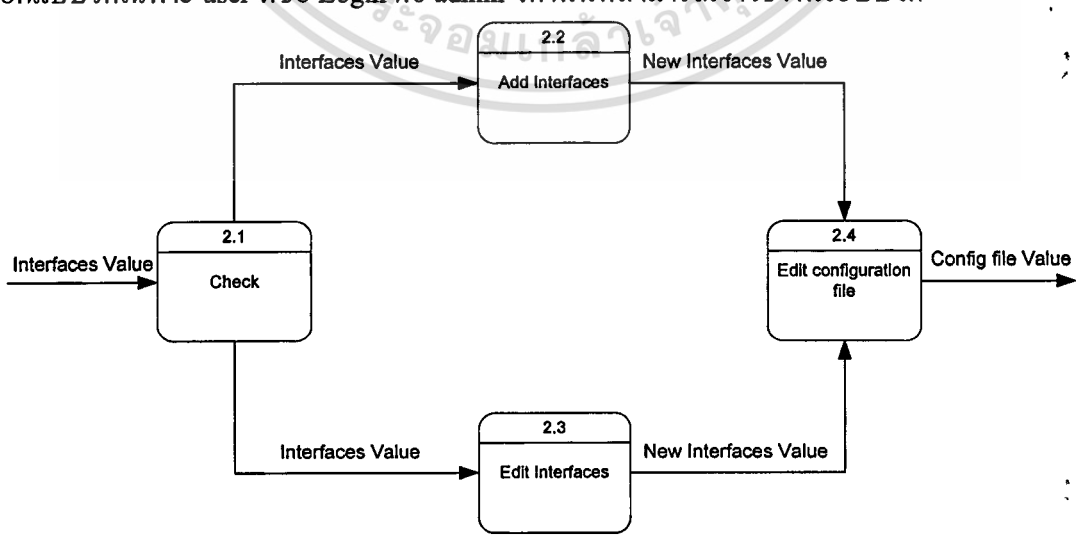
จากรูปแสดงการไหลของข้อมูลระดับ 0 โดยระบบแบ่งออกเป็น process ใหญ่ๆ ได้ 4 process คือ การตรวจสอบผู้ใช้งาน (Authentication) , การแก้ไขค่าอินเตอร์เฟส (Edit Interfaces Value) , การปรับแก้ไขค่าของเราติงโปรโตคอล (Edit Routing Protocol) และโพรเซสแสดงสถานะของเราเตอร์ (Show Status of Router)

3) Dataflow Diagram Level 1 ของระบบสามารถแสดงได้ดังรูปต่อไปนี้



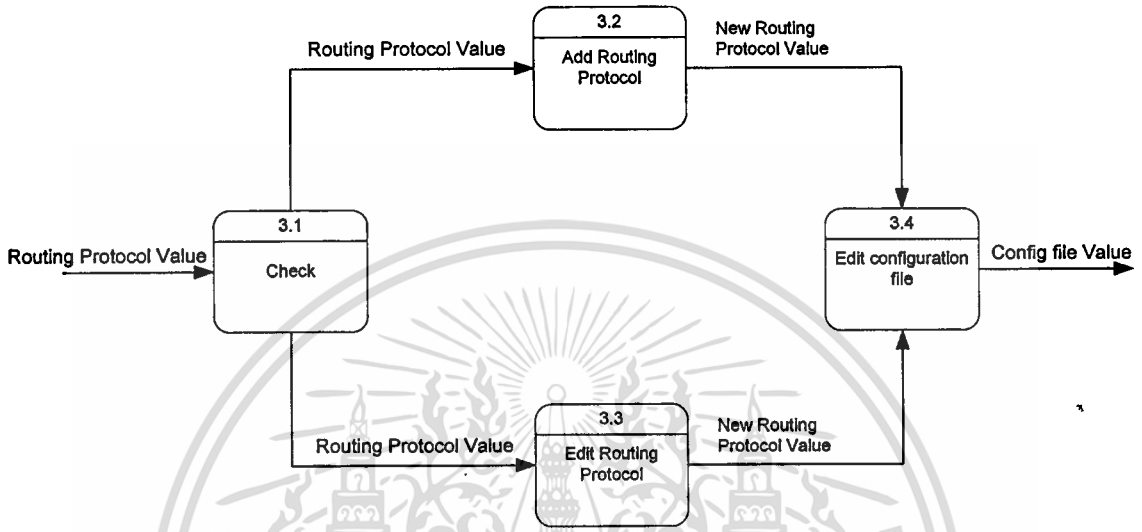
รูปที่ 3.5 แสดงการไหลของข้อมูลของระบบจัดการฟังก์ชันเราเตอร์ XORP ระดับ 1 (DFD Level 1 Process1)

จากรูปแสดงการไหลของข้อมูลระดับ 1 โดยเป็นการแสดงรายละเอียดของ Process1 เมื่อ Administrator ใส่ username และ password ระบบจะทำการตรวจสอบรายชื่อผู้ใช้จากไฟล์ของ Configuration ที่เก็บรายชื่อผู้มีสิทธิ์เข้าใช้งาน ถ้ามีก็สามารถ Login เข้าสู่ระบบได้ ในโครงงานนี้ออกแบบให้เฉพาะ user ที่ชื่อ Login คือ admin เท่านั้นที่สามารถเข้าใช้งานระบบได้



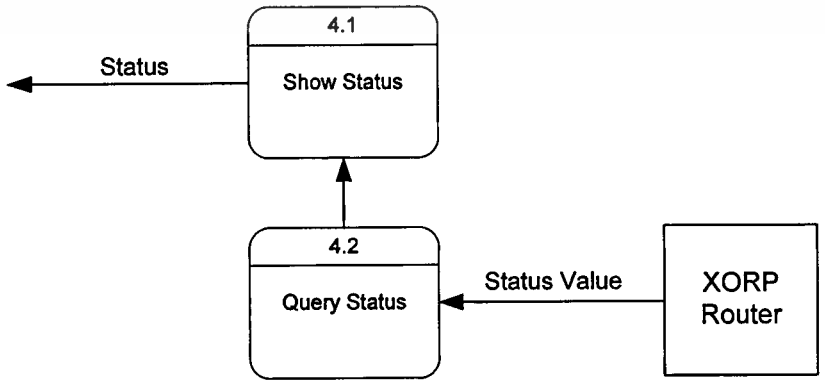
รูปที่ 3.6 แสดงการไหลของข้อมูลของระบบจัดการฟังก์ชันเราเตอร์ XORP ระดับ 1 (DFD Level 1 process2)

จากรูปที่ 3.6 แสดงการไหลของข้อมูลของระบบจัดการฟังก์ชันเราเตอร์ XORP ระดับ 1 ของ process ที่ 2 คือการแก้ไขค่าของ Interfaces ของระบบ (Edit Interfaces) โดยภาพแสดงให้เห็น process ย่อยๆ คือ นำค่า Interfaces ที่ได้รับมาตรวจสอบว่าเป็นการ Add Interfaces หรือ Edit Interfaces แล้วส่งค่าที่ได้รับมาต่อไปยัง process ย่อยๆ ต่อไป เมื่อได้ค่าใหม่แล้วส่งต่อไปยัง process Edit configuration file ต่อไป



รูปที่ 3.7 แสดงการไหลของข้อมูลของระบบจัดการฟังก์ชันเราเตอร์ XORP ระดับ 1 (DFD Level 1 process3)

จากรูปที่ 3.7 แสดงการไหลของข้อมูลของระบบจัดการฟังก์ชันเราเตอร์ XORP ระดับ 1 ของ process ที่ 3 คือการแก้ไขค่าของเราติงโปรโตคอลของระบบ (Edit Routing Protocol) จากภาพแสดงให้เห็น process ย่อยๆ ดังนี้คือ นำค่าของเราติงโปรโตคอลมาตรวจสอบว่าเป็นชนิดใดและต้องการทำอะไรคือการ เพิ่ม หรือการแก้ไข หลังจากนั้นส่งค่าที่รับมาไปยัง process ต่างๆ คือ Add Routing Protocol และ Edit Routing Protocol เมื่อผ่านออกจาก process แล้วนำค่าที่ได้ไปแก้ไขค่าของ configuration file โดย process Edit configuration file

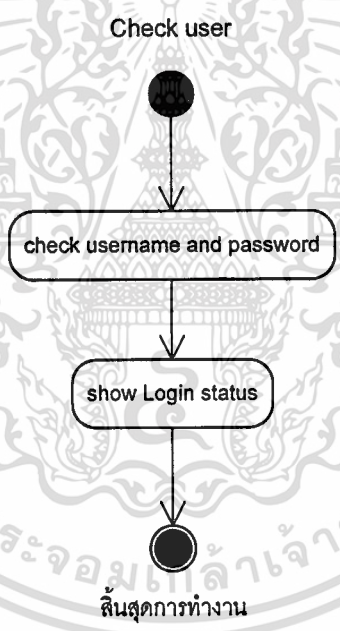


รูปที่ 3.8 แสดงการไหลของข้อมูลของระบบจัดการฟังก์ชันเราเตอร์ XORP ระดับ 1 (DFD Level 1 process4)

จากรูปที่ 3.8 แสดงการไหลของข้อมูลของระบบการจัดการฟังก์ชันเราเตอร์ XORP ระดับ 1 ของ process ที่ 4 คือ การแสดงสถานะของเราเตอร์ (Show Status of Router) จากภาพแสดงให้เห็น process ย่อย 2 process คือ process Query Status จากเราเตอร์ XORP และ process การแสดงสถานะ (Show Status)

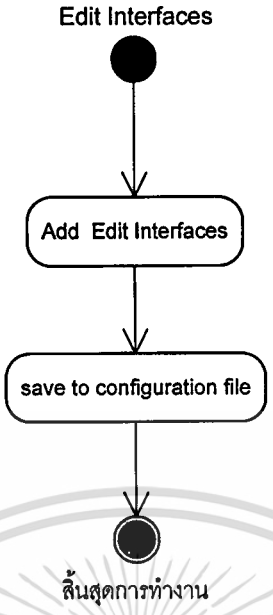
3.3 ฟังก์ชันการทำงาน

ระบบจัดการฟังก์ชันเราเตอร์ XORP ประกอบด้วยฟังก์ชันการทำงานหลักๆ อยู่สี่ฟังก์ชัน คือ การตรวจสอบผู้ใช้งานระบบ (Authentication) , ฟังก์ชันการเพิ่ม , แก้ไข ค่าของอินเทอร์เฟสของเราเตอร์ (Edit Interfaces Value) , ฟังก์ชันการเพิ่ม , แก้ไข ค่าของเราติงโปรโตคอล (Edit Routing Protocol) และฟังก์ชันแสดงสถานการณ์ทำงานของเราเตอร์ (Show Status of Router) ซึ่งกระบวนการขั้นตอนของแต่ละฟังก์ชันสามารถแสดงได้ดังนี้



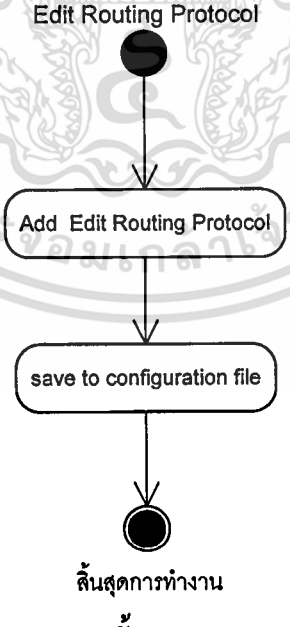
รูปที่ 3.9 Activity Diagram แสดงขั้นตอนการ Authentication

จากรูปที่ 3.9 ไดอะแกรมแสดงขั้นตอนการ Authentication เมื่อผู้ใช้กรอกข้อมูล Username และ password ของ XORP เราเตอร์ลงไประบบจะตรวจสอบว่าสามารถ Login เข้าสู่ระบบได้หรือไม่ แล้วจึงแสดงสถานะ Login ออกมา



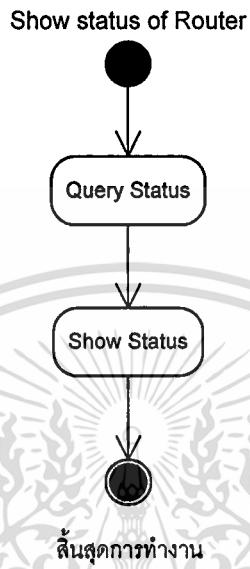
รูปที่ 3.10 Activity Diagram แสดงขั้นตอนการ Edit Interfaces Value

รูปที่ 3.10 ไดอะแกรม แสดงขั้นตอนการเพิ่ม ลบ และแก้ไขค่าของ Interfaces ของระบบ (ค่าของ Lan card) เมื่อ Administrator เข้าสู่ระบบแล้ว Administrator สามารถเพิ่ม ลบ แก้ไข,ค่าต่างๆ ของ Interfaces เช่น ค่า IP Address, Subnet Mask เป็นต้น หลังจากแก้ไขแล้ว จึงบันทึกลงสู่ไฟล์ configuration ของระบบต่อไป



รูปที่ 3.11 Activity Diagram แสดงขั้นตอนการ Edit Routing Protocol Value

รูปที่ 3.11 ไดอะแกรมแสดงขั้นตอนการ Edit Routing Protocol Value เมื่อ Administrator เข้าสู่ระบบ สามารถเพิ่มเราติงโปรโตคอล ถ้าระบบยังไม่ได้มีเราติงโปรโตคอลนั้นทำงานอยู่ หรือแก้ไขถ้าต้องการเปลี่ยนแปลงค่าของเราติงโปรโตคอล และลบเราติงโปรโตคอลนั้นๆ ถ้ามีอยู่แล้วและไม่ต้องการใช้งาน



รูปที่ 3.12 Activity Diagram แสดงขั้นตอนการ Show status of Router

รูปที่ 3.12 ไดอะแกรมแสดงขั้นตอนการ Show status of Router เมื่อ Administrator สั่งให้ระบบแสดงสถานะของระบบออกมา ระบบจะทำการตรวจสอบสถานะต่างๆ แล้วจึงแสดงออกมา

3.4 การออกแบบส่วนของข้อมูลของระบบ

โปรแกรมจัดการฟังก์ชันเราเตอร์ XORP ใช้ configuration file ในการเก็บค่าต่างๆ ของระบบซึ่งไฟล์ดังกล่าวเป็น Text file สำหรับเก็บค่าตัวแปรต่างๆ ที่จำเป็นต้องใช้ของระบบดังแสดงต่อไปนี้

ตัวอย่างไฟล์ config.boot แสดงส่วนของ Interfaces

```
/* $XORP: xorp/rtrmgr/config.boot.sample,v 1.23 2005/03/09 22:50:41 pavlin Exp $ */

interfaces {
    interface dc0 {
        description: "data interface"
        disable: false
        /* default-system-config */
        vif dc0 {
            disable: false
            address 10.10.10.10 {
                prefix-length: 24
                broadcast: 10.10.10.255
                disable: false
            }
/*
            address 10:10:10:10:10:10:10:10 {
                prefix-length: 64
                disable: false
            }
*/
        }
    }
```

3.4.1 การออกแบบโครงสร้างข้อมูลของระบบ

3.4.1.1 ตารางแสดงข้อมูล Interfaces

ตารางที่ 3.1 แสดงฟิลด์ของข้อมูล Interfaces

ชื่อข้อมูล	ชื่อเขตข้อมูล	ชนิดของข้อมูล	คำอธิบาย
ชื่ออินเทอร์เฟส	Interfaces_name	Char(5)	เก็บชื่อของ Interfaces
คำอธิบาย	description	Char(20)	คำอธิบายของ Interfaces
สถานะ	disable	Char(5)	สถานะ การใช้งาน Interfaces
เวอร์ชวลอินเตอร์เฟส	vif	Char(5)	ชื่อเวอร์ชวล Interfaces
สถานะของเวอร์ชวลอินเตอร์เฟส	disable_vif	Char(5)	สถานะของเวอร์ชวลอินเตอร์เฟส
หมายเลข IP Address	address	Char(15)	หมายเลข IP Address ของ Interfaces
ค่า Subnet Mask	prefix-length	Char(2)	Subnet Mask ของ IP Address
ค่า Broadcast	broadcast	Char(15)	Broadcast ของ IP Address
สถานะของ Card	disable_interface	Char(5)	สถานะของ LAN Card

ตัวอย่างไฟล์ config.boot แสดงส่วน tunnel interface

```
interface tun0 {
    description: "tunnel interface"
    disable: false
    /* default-system-config */
    vif tun0 {
        disable: false
        address 10.10.10.11 {
            prefix-length: 32
            destination: 10.20.20.20
            disable: false
        }
    }
}
```

3.4.1.2 ตารางแสดงข้อมูล tunnel interface

ตารางที่ 3.2 แสดงฟิลด์ของข้อมูล interface tun0

ชื่อข้อมูล	ชื่อเซตข้อมูล	ชนิดของข้อมูล	คำอธิบาย
ชื่อ interface tunnel	interface_tun	Char(5)	ชื่อ interface tunnel
คำอธิบาย	description	Char(20)	คำอธิบายของ interface tunnel
สถานะ	disable_tun	Char(5)	สถานะของ interface tunnel
เวอร์ชวลอินเตอร์เฟส	vif	Char(5)	ชื่อ Virtual interface tunnel
สถานะของเวอร์ชวลอินเตอร์เฟส	disable_vif	Char(5)	สถานะของเวอร์ชวลอินเตอร์เฟส
หมายเลข IP Address	address	Char(15)	หมายเลข IP Address ของเวอร์ชวล interface
ค่า Subnet Mask	prefix-length	Char(2)	ค่า Subnet Mask ของ virtual interface
หมายเลข IP Address ของ Destination	destination	Char(15)	หมายเลข IP Address ของ Destination
สถานะของ address	disable_addr	Char(5)	สถานะของ address

ตัวอย่างไฟล์ config.boot แสดงส่วน discard

```
interface discard0 {
    description: "discard interface"
    disable: false
    discard: true
    vif discard0 {
        disable: false
        address 127.127.0.1 {
            prefix-length: 32
            disable: false
        }
    }
}
```

3.4.1.3 ตารางแสดงข้อมูล interface discard

ตารางที่ 3.3 แสดงฟิลด์ของข้อมูล interfaces discard

ชื่อข้อมูล	ชื่อเซตข้อมูล	ชนิดของข้อมูล	คำอธิบาย
ชื่อ interface_discard	interface_discard	Char(8)	ชื่อ interface_discard
คำอธิบาย	description	Char(20)	คำอธิบายของ interface discard
สถานะของ interface	disable	Char(5)	สถานะของ interface
สถานะของ discard	discard	Char(5)	สถานะของ discard
เวอร์ชวลอินเตอร์เฟส	vif	Char(8)	ชื่อของ virtual interface discard
สถานะของ virtual interface	disable_vif	Char(5)	สถานะของ virtual interface discard
หมายเลข IP Address	address	Char(15)	หมายเลข IP Address ของ virtual interface discard
ค่า Subnet Mask	prefix-length	Char(2)	ค่า Subnet Mask ของ virtual Interface discard
สถานะของ address	disable_addr	Char(5)	สถานะของ address

ตัวอย่างไฟล์ config.boot แสดงส่วนของ Routing Protocols แบบ static

```
protocols {
    static {
        route4 10.20.0.0/16 {
            next-hop: 10.10.10.20
            metric: 1
        }
        mrib-route4 10.20.0.0/16 {
            next-hop: 10.10.10.30
            metric: 1
        }
/*
        route6 20:20:20:20::/64 {
            next-hop: 10:10:10:10:10:10:10:20
            metric: 1
        }
        mrib-route6 20:20:20:20::/64 {
            next-hop: 10:10:10:10:10:10:10:30
            metric: 1
        }
*/
    }
}
```

3.4.1.4 ตารางแสดงข้อมูล Routing protocols แบบ static

ตารางที่ 3.4 แสดงฟิลด์ของข้อมูล Routing protocols แบบ static

ชื่อข้อมูล	ชื่อเขตข้อมูล	ชนิดของข้อมูล	คำอธิบาย
หมายเลข subnet	route4	Char(18)	หมายเลข subnet ของเครือข่าย
หมายเลขของ next hop	next-hop	Char(15)	หมายเลขของ next hop ตัวถัดไป ที่ต้องการส่งไป
ค่าเมทริกซ์	metric	Char(1)	ค่าของ metric
หมายเลข mrib	mrib-route4	Char(18)	หมายเลขเครือข่าย mrib
หมายเลขของ next hop	next-hop_mrib	Char(15)	หมายเลขของ next hop ของ mrib
ค่าเมทริกซ์ mrib	metric_mrib	Char(1)	ค่าของ metric mrib

ตัวอย่างไฟล์ config.boot แสดงส่วนของ Routing Protocols แบบ RIP

```
protocols {
    rip {
/* Redistribute routes for connected interfaces */
/*
        export connected {
            metric: 0
            tag: 0
        }
*/
/* Redistribute static routes */
/*
        export static {
            metric: 1
            tag: 0
        }
*/
/* Run on specified network interface addresses */
/*
        interface dc0 {
            vif dc0 {
                address 10.10.10.10 {
                    disable: false
                }
            }
        }
*/
    }
}
```

3.4.1.5 ตารางแสดงข้อมูล Routing protocols แบบ RIP

ตารางที่ 3.5 แสดงฟิลด์ของข้อมูล Routing protocols แบบ RIP

ชื่อข้อมูล	ชื่อเขตข้อมูล	ชนิดของข้อมูล	คำอธิบาย
ค่า export	export	Char(9)	ค่า export
ค่าเมทริกซ์	metric	Char(1)	ค่าเมทริกซ์
ค่า tag	tag	Char(1)	ค่า tag
ค่า export	export	Char(6)	ค่า export
ค่าเมทริกซ์	metric	Char(1)	ค่าเมทริกซ์
ค่า tag	tag	Char(1)	ค่า tag
ชื่อ interface	interface	Char(5)	ชื่อของ interface
ชื่อ virtual interface	vif	Char(5)	ชื่อ virtual interface
หมายเลข IP Address	address	Char(15)	หมายเลข IP Address
สถานะของ interface	disable	Char(5)	สถานะของ interface

ตัวอย่างไฟล์ config.boot แสดงส่วนของ Routing Protocols แบบ OSPF

```
protocols {
    ospf {
        router-id: 10.10.10.10
        area 0.0.0.0 {
            stub: false
            interface dc0 {
                hello-interval: 5
            }
        }
    }
}
```

3.4.1.6 ตารางแสดงข้อมูล Routing protocols แบบ OSPF

ตารางที่ 3.6 แสดงฟิลด์ของข้อมูล Routing protocols แบบ OSPF

ชื่อข้อมูล	ชื่อเขตข้อมูล	ชนิดของข้อมูล	คำอธิบาย
หมายเลข IP Address	router-id	Char(15)	หมายเลข IP Address ของเราเตอร์
หมายเลข area	area	Char(15)	หมายเลข IPAddress ของ area
ค่า stub	stub	Char(5)	ค่าของ stub
ชื่อ interface	interface	Char(5)	ชื่อของ interface
ค่า hello-interval	hello-interval	Char(1)	ค่า hello-interval

ตัวอย่างไฟล์ config.boot แสดงส่วนของ Routing Protocols แบบ BGP

```
protocols {
    bgp {
        bgp-id: 10.10.10.10
        local-as: 65002

        peer 10.30.30.30 {
            local-ip: 10.10.10.10
            as: 65000
            next-hop: 10.10.10.20
/*
            local-port: 179
            peer-port: 179
*/
            /* holdtime: 120 */
            /* disable: false */

            /* Optionally enable other AFI/SAFI combinations */
            /* ipv4-multicast: true */

            /* ipv6-unicast: true */
            /* ipv6-multicast: true */
        }

        /* Originate IPv4 Routes */
/*
        network4 10.10.10.0/24 {
            next-hop: 10.10.10.10
            unicast: true
            multicast: true
        }
*/
```

3.4.1.7 ตารางแสดงข้อมูล Routing protocols แบบ BGP

ตารางที่ 3.7 แสดงฟิลด์ของข้อมูล Routing protocols แบบ BGP

ชื่อข้อมูล	ชื่อเขตข้อมูล	ชนิดของข้อมูล	คำอธิบาย
หมายเลข IP Address ของ bgp	bgp-id	Char(15)	หมายเลข IP Address ของ bgp
หมายเลข Local AS Number	local-as	Char(5)	หมายเลข Local AS Number
หมายเลข IP ของ peer	peer	Char(15)	หมายเลข IP ของ peer
หมายเลข Local IP Address	local-ip	Char(15)	หมายเลข Local IP Address
หมายเลข AS Number	as	Char(5)	หมายเลข AS Number
หมายเลข Next-hop	Next-hop	Char(15)	หมายเลข Next-hop
หมายเลข local port	local-port	Char(3)	หมายเลข local port
หมายเลข peer port	peer-port	Char(3)	หมายเลข peer port
เวลา hold	holdtime	Char(3)	เวลา hold
สถานะ	disable	Char(5)	สถานะ
multicast ของ IP V4	ipv4-multicast	Char(5)	ค่า multicast ของ IP V4
Network4 subnet	network4	Char(18)	ค่า subnet ของ network V4
Next-hop	next-hop	Char(15)	Hop count ต่อไป
Unicast	unicast	Char(5)	สถานะ unicast
Multicast	Multicast	Char(5)	สถานะ multicast

บทที่ 4
การพัฒนาระบบงาน

ในการพัฒนาระบบงาน การพัฒนาอุปกรณ์เราเตอร์โดยใช้ XORP พร้อมการเชื่อมต่อกับผู้ใช้แบบเว็บเบสยูสเซอร์อินเตอร์เฟส ครั้งนี้ได้ติดตั้งระบบทั้งหมดลงสู่ Compact Flash ขนาด 512 MB เพื่อบูตระบบและใช้งานกับ Single board ของ Soekris รุ่น net5501 ดังจะได้แสดงต่อไปนั้น ได้มีการกำหนดแผนการพัฒนาและขั้นตอนการพัฒนาระบบงานดังนี้

4.1 การวางแผนการปฏิบัติงาน

การวางแผนการปฏิบัติงาน ได้เลือกใช้ระบบปฏิบัติการ ฮาร์ดแวร์ และซอฟต์แวร์ เพื่อพัฒนาทั้งตัวอุปกรณ์เราเตอร์ XORP และโปรแกรมเว็บเบสสำหรับการควบคุมสั่งงานดังต่อไปนี้

1. ระบบปฏิบัติการ เลือกใช้ระบบปฏิบัติการ FreeBSD 6.2

เป็นระบบปฏิบัติการแบบยูนิกซ์ที่ได้รับความนิยมอย่างแพร่หลาย และไม่ต้องเสียค่าใช้จ่าย โดยระบบปฏิบัติการ FreeBSD นั้นเป็นระบบปฏิบัติการที่มีเสถียรภาพและมีความปลอดภัยสูงกว่าระบบปฏิบัติการ Linux นอกจากนี้ระบบปฏิบัติการ FreeBSD ยังสามารถย่อขนาดได้เพื่อบรรจุลง compact flash ได้ สามารถ download ได้จาก ftp.FreeBSD.org

2. ซอฟต์แวร์จัดการฟังก์ชันเราเตอร์ XORP v1.4

เป็นโปรแกรมสำหรับจัดการฟังก์ชันเราเตอร์ ที่เป็นแบบ Open Source สามารถดาวน์โหลดมาใช้ได้ฟรีโดยไม่ต้องเสียค่าใช้จ่ายสามารถดาวน์โหลดได้จากเว็บไซต์ของผู้พัฒนา ในการพัฒนาระบบครั้งนี้ผู้พัฒนาได้เลือกใช้โปรแกรมจัดการฟังก์ชันเราเตอร์ XORP เวอร์ชัน 1.4 (http://www.xorp.org) เมื่อดาวน์โหลดมาแล้วจะต้องนำ source code มาคอมไพล์ก่อนจึงจะสามารถใช้งานได้

3. ซอฟต์แวร์ภาษา เลือกใช้ PHP เวอร์ชัน 4.4.7

ภาษา PHP จัดเป็นภาษา script ที่สามารถทำงานร่วมกับ HTML โดยสามารถเขียน script แทรกเข้าไปใน tag ภาษา HTML ได้ หรือสามารถเขียนเป็นไฟล์ PHP ก็ได้ โดย PHP เป็น open source ที่สามารถใช้งานได้กับหลายๆ ระบบปฏิบัติการ เช่น Windows , Linux , Unix เป็นต้น และยังสามารถรองรับการติดต่อกับฐานข้อมูลหลายๆ ชนิดด้วย

4. เว็บเซอร์เวอร์ เลือกใช้ lighttpd v 1.4.15

เป็นเว็บเซอร์เวอร์ที่มีขนาดเล็กและมีประสิทธิภาพสูง สามารถทำ Authentication ได้ สามารถทำงานตอบสนองต่อระบบงานได้ ดาวน์โหลดได้จากเว็บไซต์ http://www.lighttpd.net/

เนื่องจากระบบงานนี้ต้องบรรจุโปรแกรมและระบบปฏิบัติการทั้งหมดลงใน compact flash ขนาด 512 MB

4.2 การพัฒนาระบบ Embedded System

เนื่องจากโครงงานนี้ได้พัฒนาทั้งตัวโปรแกรมเว็บเบสสำหรับควบคุมเราเตอร์ และตัวอุปกรณ์เราเตอร์เพื่อให้สามารถติดตั้งระบบทั้งหมดลงสู่ Compact Flash ขนาด 512 MB เพื่อทำระบบ Embedded System โดยมีกระบวนการทำงานดังนี้

1) การย่อขนาดของ FreeBSD เพื่อบรรจุระบบปฏิบัติการลงสู่ Compact Flash ขั้นตอนการทำนั้นสามารถศึกษารายละเอียดเพิ่มเติมได้จากเว็บไซต์ต่อไปนี้คือ

- https://neon1.net/misc/minibsd.html
- http://www.ultradesic.com

ขั้นตอนการทำงานโดยสังเขปมีดังนี้

1) ติดตั้ง FreeBSD jail ภายใต้ FreeBSD
2) ปรับแต่ง FreeBSD jail
3) Safety Check ตรวจสอบว่าได้เข้ามาทำงานที่ jail เรียบร้อย
4) Building miniBSD 6.X
5) Create Directory Structure
6) Rebuilding the boot loader
7) Building Dynamic Executables
8) Copy the binaries over
9) Configuration boot files
10) Kernel Compile
11) Copying the libraries
12) Populating /etc
13) สร้าง Binary image
14) Writing the binary image to the media
15) First Boot

รายละเอียดอยู่ในภาคผนวก ท้ายเอกสารนี้

2) การติดตั้งโปรแกรมจัดการฟังก์ชันเราเตอร์ XORP ลงสู่ Compact Flash

การติดตั้งสามารถทำได้โดยการ compile โปรแกรมจัดการเราเตอร์ XORP ในระบบปฏิบัติการ FreeBSD ด้วยโปรแกรมแปลภาษา gcc ก่อน หลังจากนั้นจึงคัดลอก executable file ของ XORP ที่จำเป็นต้องใช้และ configuration file ลงสู่ compact flash

3) ติดตั้งโปรแกรมเว็บเซอร์เวอร์ lighttpd

การติดตั้งโปรแกรม lighttpd เป็นเว็บเซอร์เวอร์เนื่องจากเป็นเว็บเซอร์เวอร์ที่มีขนาดเล็กทำงานได้รวดเร็วมีประสทธิภาพ เมื่อ compile โปรแกรมเสร็จเรียบร้อยจะได้ execute file ที่ /usr/local/sbin ให้ copy ไปไว้ใน miniBSD และ config file ที่ต้อง copy ไปด้วยสองไฟล์คือ lighttpd.conf และ lighttpd.user รายละเอียดการติดตั้งอยู่ในภาคผนวก ก

4) ติดตั้งโปรแกรมแปลภาษา PHP

การติดตั้งตัวโปรแกรมแปรภาษา PHP ในการพัฒนาโครงงานนี้ใช้ PHP 4.4.7 ทำการ download และ compile บนระบบ miniBSD หลังจากนั้นแก้ไฟล์ php.ini นำไปไว้ที่ /usr/local/lib ของ miniBSD รายละเอียดการติดตั้งอยู่ในภาคผนวก ก

5) พัฒนาโปรแกรมเว็บเบสด้วยภาษา PHP และ Java script

การพัฒนาโปรแกรมเว็บเบสยูสเซอร์อินเตอร์เฟสในการพัฒนาระบบงานครั้งนี้ได้เลือกใช้โปรแกรมภาษาสคริปต์คือ PHP สำหรับการสั่งงาน การค้นหาตัดคำต่างๆ การบันทึกไฟล์ การลบ Java Script สำหรับควบคุมการทำงานของฟอร์มในการรับข้อมูลให้ผู้ใช้โปรแกรมกรอกข้อมูลได้อย่างถูกต้อง และ shell script สำหรับการสั่งงานต่างๆ ให้เราเตอร์ XORP ทำงานหรือแสดงผลออกมาทางหน้าจอ

4.3 การออกแบบฟังก์ชันการทำงาน

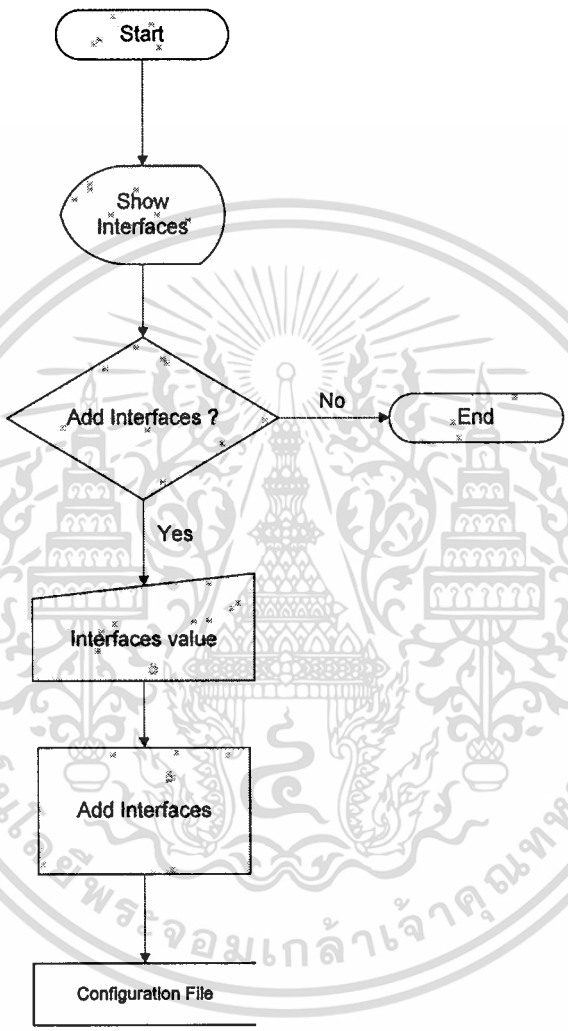
ฟังก์ชันการทำงานของระบบ Web-base User Interface ของเราเตอร์ XORP ประกอบด้วยฟังก์ชันดังต่อไปนี้

4.3.1 ฟังก์ชันเกี่ยวกับ Interface ฟังก์ชันในหมวดหมู่นี้จะเป็นฟังก์ชันที่ทำหน้าที่เกี่ยวกับค่าต่างๆ ของ Interface ที่มีอยู่ในระบบของตัวอุปกรณ์ XORP Router นั้นจะมีชื่อเรียกที่แน่นอนคือ vr0 – vr3 มีทั้งหมด 4 Interface แต่การตั้งค่าที่กำหนดมานั้นยังไม่ตรงกับความต้องการของผู้ใช้งาน ดังนั้นกลุ่มฟังก์ชันนี้ออกแบบให้ผู้ใช้สามารถเพิ่มค่า ลบค่า และแก้ไขค่าต่างๆ ได้

4.3.1.1) ฟังก์ชัน Add Interfaces

ตัวอุปกรณ์เราเตอร์ XORP นั้นใช้ single board ของ soekris รุ่น net5501 ซึ่งมี Interface มาให้จำนวน 4 Interface ใช้ชื่อว่า vr0, vr1, vr2 และ vr3 แต่ใน configuration ของ XORP router ยังไม่ได้เพิ่มอินเตอร์เฟสเข้าไป ขั้นตอนของฟังก์ชันการ add interface มีดังนี้

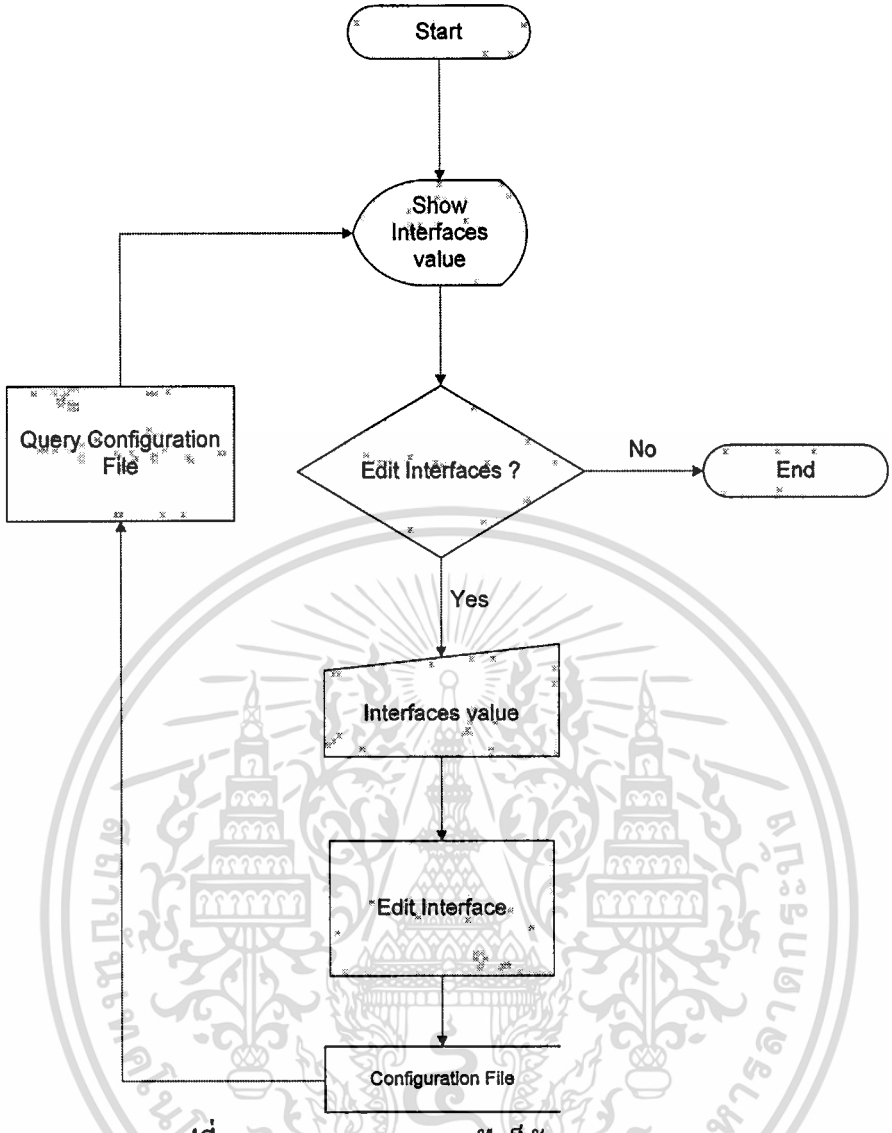
1) หน้าเว็บแสดงรายชื่อ Interface ที่มีอยู่ให้
2) เลือก Interface ของระบบที่ต้องการเพิ่มเข้าไป
3) ใส่ค่าของ Interface ที่ต้องการ ได้แก่ หมายเลข Description, mtu, virtual interface, IP Address , prefix-length, broadcast และ destination ซึ่งเมื่อเพิ่มค่าต่างๆ เข้าไปและสั่ง add ระบบจะบันทึกค่าลง Configuration file



รูปที่ 4.1 แสดง Flow Chart ฟังก์ชัน Add Interfaces

4.3.1.2) ฟังก์ชัน Edit Interfaces

เมื่อมีการเพิ่มค่าของ Interfaces ลงไปในระบบแล้วผู้ใช้งาน (admin) ต้องการแก้ไขค่าของ Interface ของระบบโดยระบบจะเข้าไปค้นค่าที่ได้บันทึกไว้ใน Configuration file ออกมาแสดงในหน้าเว็บเพจ ผู้ใช้งานระบบทำการแก้ไขค่าและบันทึกกลับลงไป



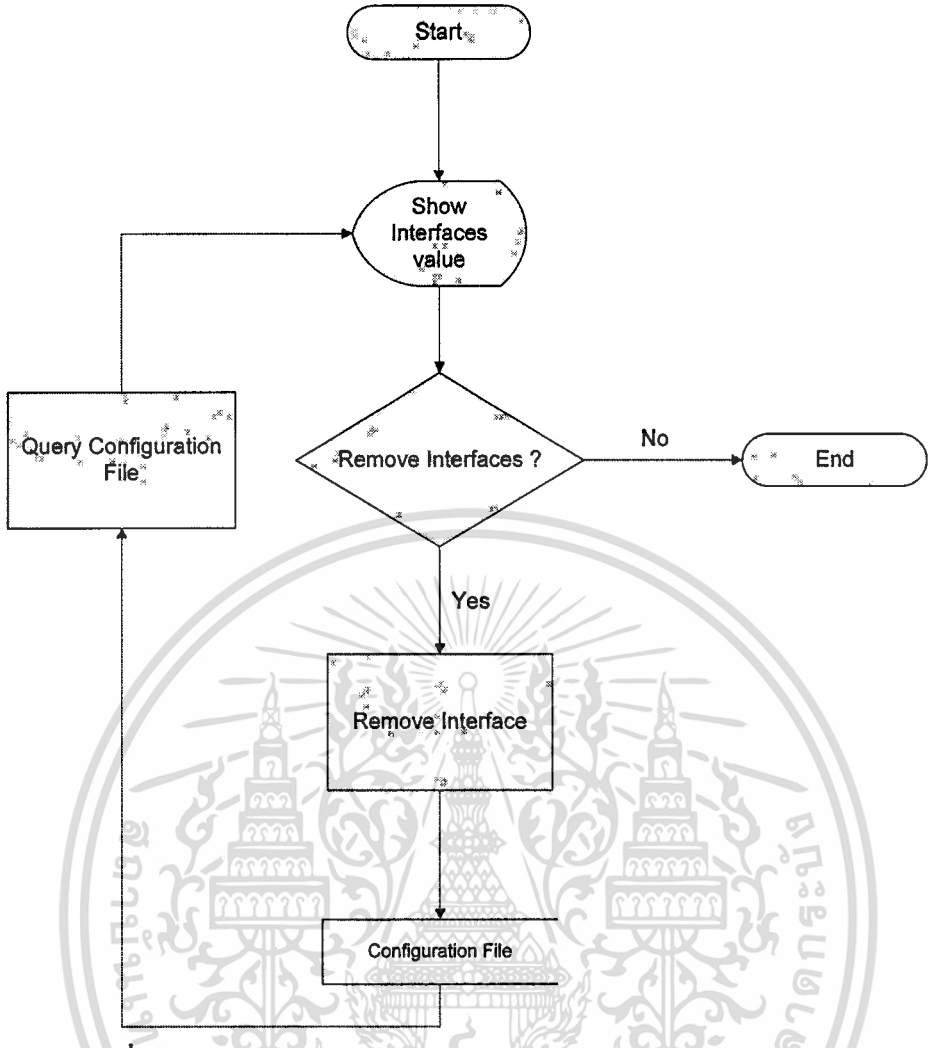
รูปที่ 4.2 แสดง Flow Chart ฟังก์ชัน Edit Interfaces

การทำงานของฟังก์ชันนี้เริ่มจาก

1) ผู้ใช้เปิดหน้าเว็บเพจแสดงรายชื่อ Interface และค่าต่างๆ ขึ้นมาแสดง
2) ผู้ใช้เลือก Interface ที่ต้องการแก้ไขค่าต่างๆ
3) ระบบแสดงหน้า form ของ Interface ที่มีค่าต่างๆ ที่ได้กำหนดเอาไว้
4) ผู้ใช้เปลี่ยนแปลงค่าต่างๆ
5) ผู้ใช้สั่งให้ระบบบันทึกค่าลงสู่ไฟล์ Configuration

4.3.1.3) ฟังก์ชัน Remove Interfaces

เมื่อผู้ใช้ได้เพิ่มค่า Interface ใน configuration file แล้วและไม่ต้องการใช้ค่านั้นอีก ผู้ใช้สามารถที่จะ remove ค่านั้นได้โดยที่ระบบจะแสดงค่า interface ที่มีอยู่ในระบบออกมาหลังจากนั้นผู้ใช้เลือกค่าของ interface ที่ต้องการ remove ออก



รูปที่ 4.3 แสดง Flow Chart ฟังก์ชันการลบ Interfaces ออกจากระบบ

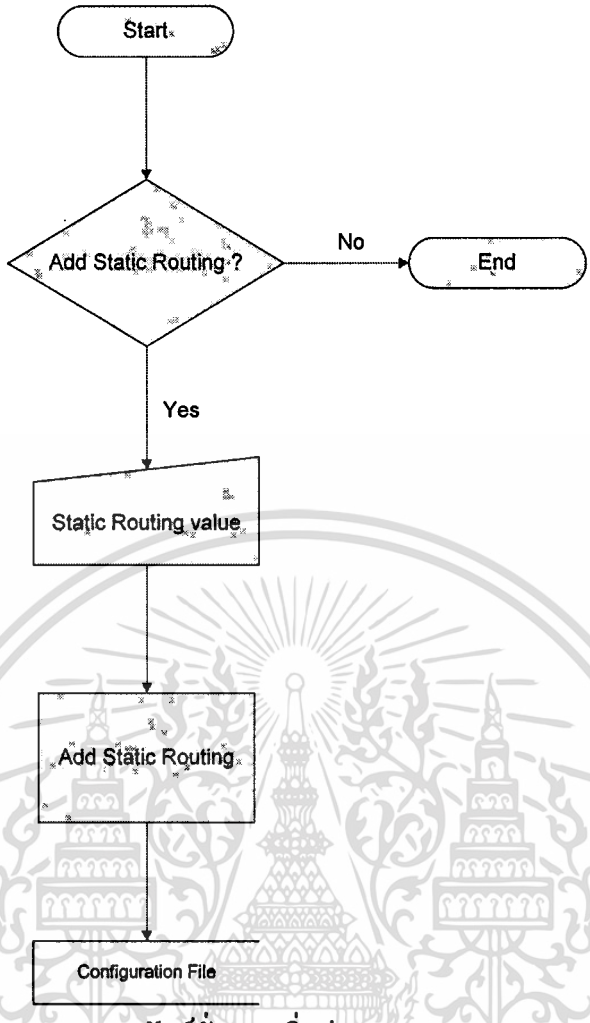
การทำงานของฟังก์ชันนี้เริ่มจาก

1) ผู้ใช้เปิดหน้าเว็บเพจแสดงรายชื่อ Interface และค่าต่างๆ ขึ้นมาแสดง
2) ผู้ใช้เลือก Interface ที่ต้องการ remove ออกจากระบบ
3) ระบบจะถามผู้ใช้ว่าต้องการ remove หรือไม่
4) ถ้าผู้ใช้ไม่ต้องการลบค่า ระบบจะจบการทำงาน
5) ถ้าผู้ใช้ต้องการลบค่า ระบบจะลบค่าออกและบันทึกลงไฟล์ Configuration

4.3.2 ฟังก์ชันเกี่ยวกับ Static Routing

4.3.2.1) ฟังก์ชั่นการเพิ่มค่า Static Routing (Add Static Routing)

เป็นฟังก์ชันที่ทำหน้าที่ในการเพิ่มค่า Static Routing ลงใน configuration file เพื่อให้เราเตอร์สามารถค้นห้าเส้นทางการส่งต่อ packet แบบ static ได้โดยค่าที่เพิ่มลงไปในระบบได้แก่ IP Address, prefix-length, next-hop และ metric ดังนี้



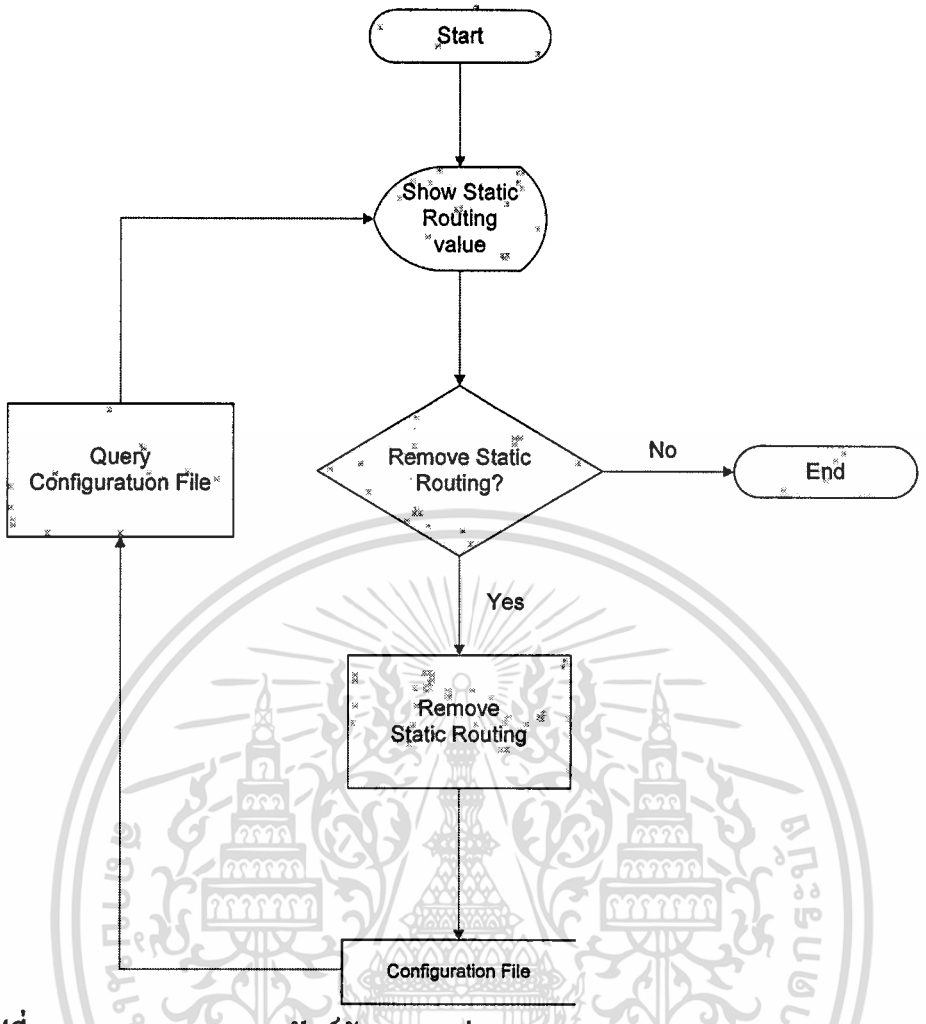
รูปที่ 4.4 แสดง Flow Chart ฟังก์ชั่นการเพิ่มค่า Static Routing (Add Static Routing)

การทำงานของฟังก์ชันนี้เริ่มจาก

1) ผู้ใช้เปิดหน้าเว็บเพจแสดงค่า Static Routing
2) เว็บเพจแสดงหน้าแสดงค่าของ Static Routing
3) ผู้ใช้ใส่ค่าของ Static Routing
4) ผู้ใช้สั่งบันทึกค่าของ Static Routing ลงสู่ Configuration file

4.3.2.2) ฟังก์ชั่นการลบค่า Static Routing (Remove Static Routing)

เป็นฟังก์ชันที่ทำหน้าที่ในการลบค่า Static Routing ที่ไม่ต้องการใช้งานออกจาก Configuration file เพื่อให้ XORP เราเตอร์นั้นส่ง packet ไปยังเส้นทางที่ต้องการเท่านั้น การทำงานของฟังก์ชันนี้เริ่มจาก ผู้ใช้เข้าใช้งานฟังก์ชันแสดง ค่า Static Routing ใน configuration file ออกมา ผู้ใช้เลือกค่าที่ไม่ต้องการใช้งานแล้วสั่งลบออกจากไฟล์



รูปที่ 4.5 แสดง Flow Chart ฟังก์ชันการลบค่า Static Routing (Remove Static Routing)

การทำงานของฟังก์ชันนี้เริ่มจาก

1) ผู้ใช้เปิดหน้าเว็บเพจแสดงค่า Static Routing
2) เว็บเพจแสดงหน้าแสดงค่าของ Static Routing ที่มีอยู่ในระบบ
3) ผู้ใช้ลบค่า Static Routing ที่ไม่ต้องการใช้งานออกออก
4) ระบบบันทึกค่าของ Static Routing ลงสู่ Configuration file

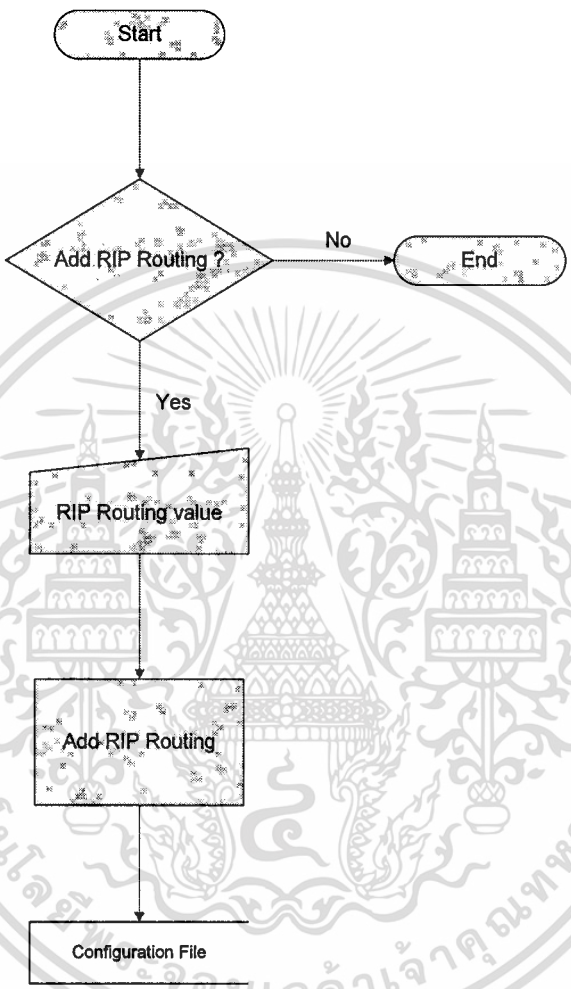
4.3.3 ฟังก์ชันเกี่ยวกับ Dynamic Routing

โปรโตคอลในการค้นหาเส้นทางแบบพลวัตรที่โครงการพัฒนาระบบนี้นำมาใช้คือ โปรโตคอล RIP และ OSPF ซึ่งเป็นโปรโตคอลการค้นหาเส้นทางแบบ Dynamic Routing ที่เลือกพัฒนาโดยสามารถเพิ่มค่าต่างๆ ลงใน Configuration File

Routing Information Protocols (RIP)

4.3.3.1) ฟังก์ชันเพิ่มค่าโปรโตคอลการหาเส้นทางแบบ RIP (Add RIP Protocol)

เป็นฟังก์ชันที่ใช้สำหรับการเพิ่มค่าเพื่อการปรับแต่งโปรโตคอลค้นหาเส้นทางแบบ RIP สำหรับค่าที่ต้องเพิ่มลงไปได้แก่ interface , IP Address เป็นต้น



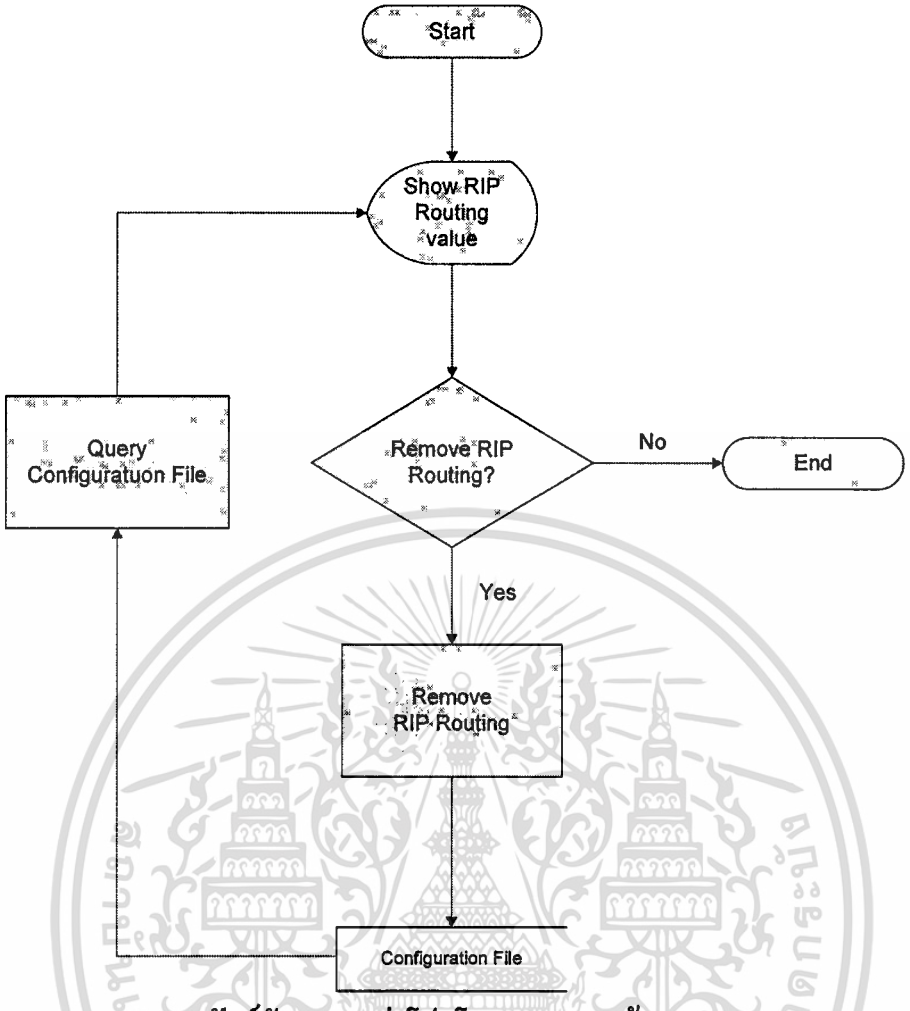
รูปที่ 4.6 แสดง Flow Chart ฟังก์ชันเพิ่มค่าโปรโตคอลการหาเส้นทางแบบ RIP (Add RIP Protocols)

การทำงานของฟังก์ชันนี้เริ่มจาก

1) ผู้ใช้เปิดหน้าเว็บเพจแสดงค่า RIP Routing
2) เว็บเพจแสดงหน้าแสดงหน้า form ค่าของ RIP Routing
3) ผู้ใช้บันทึกค่า RIP Routing
4) ระบบบันทึกค่าของ RIP Routing ลงสู่ Configuration file

4.3.3.2) ฟังก์ชันลบค่าโปรโตคอลการหาเส้นทางแบบ RIP (Remove RIP Protocol)

เป็นฟังก์ชันที่ใช้สำหรับการลบค่าเพื่อการปรับแต่งโปรโตคอลค้นหาเส้นทางแบบ RIP ในกรณีที่ค่านั้นไม่ต้องการใช้งานต่อไป



รูปที่ 4.7 แสดง Flow Chart ฟังก์ชันการลบค่าโปรโตคอลการหาเส้นทางแบบ RIP (Remove RIP Protocols)

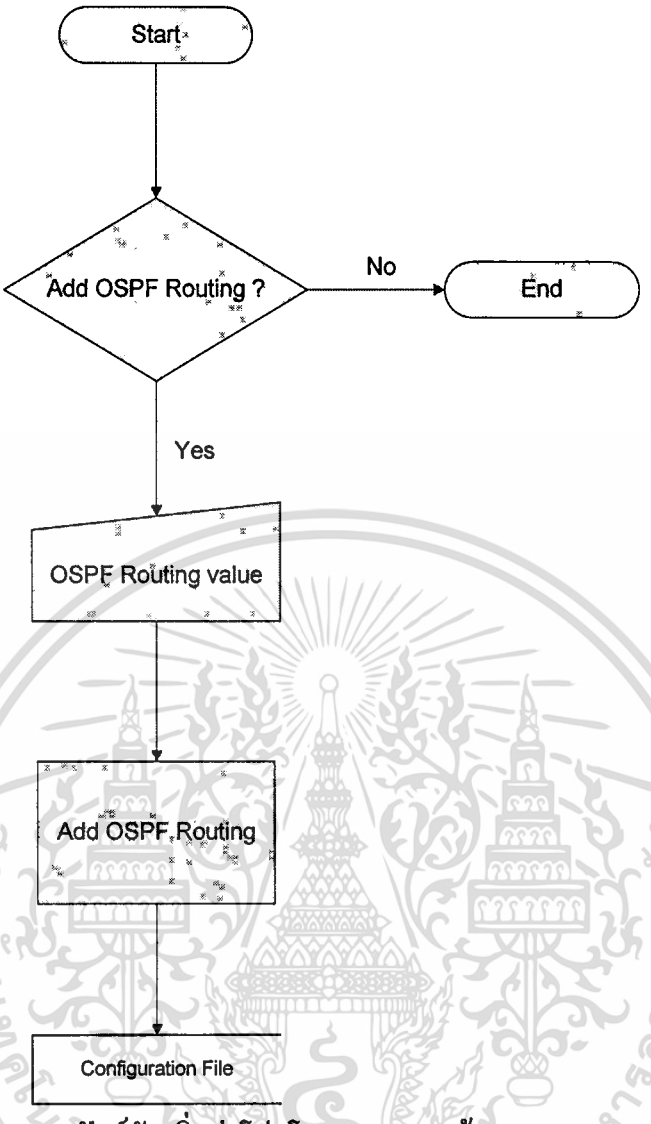
การทำงานของฟังก์ชันนี้เริ่มจาก

1) ผู้ใช้เปิดหน้าเว็บเพจแสดงค่า RIP Routing ที่มีอยู่ในระบบ
2) ระบบแสดงค่า RIP Routing ที่มีอยู่ใน Configuration file ขึ้นมาแสดง
3) ผู้ใช้ลบค่า RIP Routing ที่ไม่ต้องการ
4) ระบบบันทึกค่าของ RIP Routing ลงสู่ Configuration file
5) ออกจากระบบ เมื่อผู้ใช้ไม่ต้องการลบค่า RIP Routing

Open Shortest Path First (OSPF)

4.3.3.3) ฟังก์ชันเพิ่มค่าโปรโตคอลการหาเส้นทางแบบ OSPF (Add OSPF Protocol)

เป็นฟังก์ชันที่ใช้สำหรับการเพิ่มค่าเพื่อการปรับแต่งโปรโตคอลค้นหาเส้นทางแบบ OSPF สำหรับค่าที่ต้องเพิ่มลงไปได้แก่ IP Address , Interface, Area เป็นต้นมีขั้นตอนการทำงานดังนี้



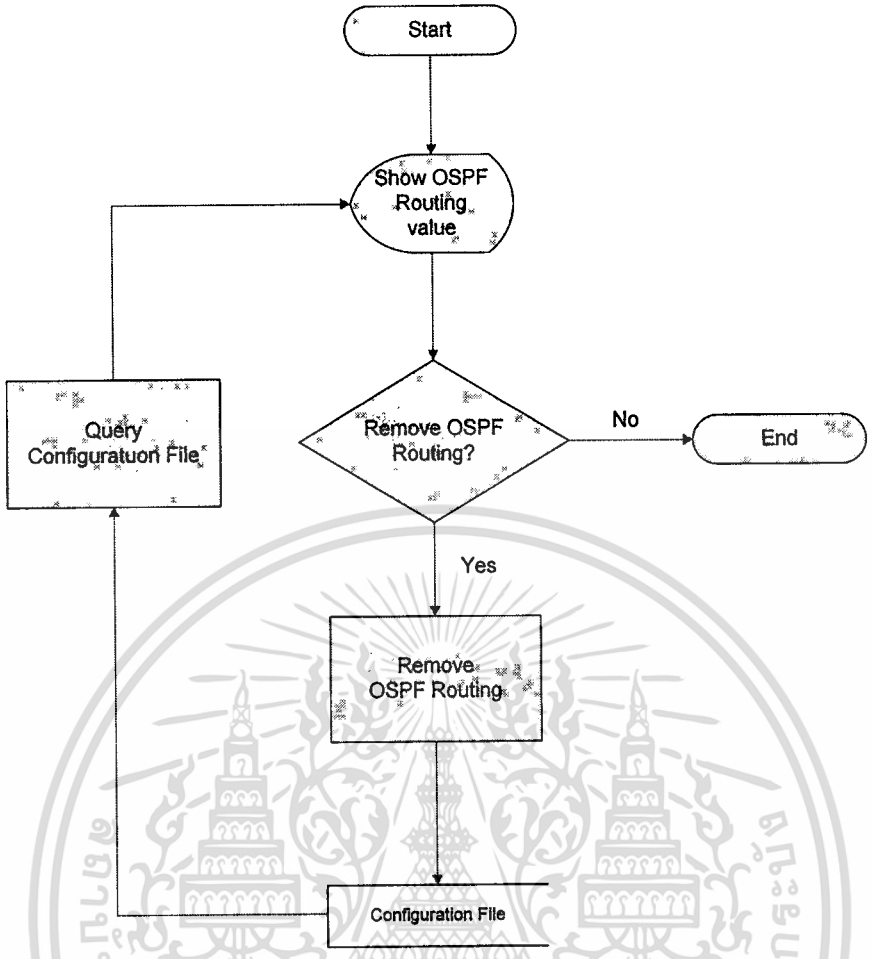
รูปที่ 4.8 แสดง Flow Chart ฟังก์ชันเพิ่มค่าโปรโตคอลการหาเส้นทางแบบ OSPF (Add OSPF Protocol)

การทำงานของฟังก์ชันนี้เริ่มจาก

1) ผู้ใช้เปิดหน้าเว็บเพจแสดงค่า OSPF Routing
2) เว็บเพจแสดงหน้าแสดงหน้า form ค่าของ OSPF Routing
3) ผู้ใช้บันทึกค่า OSPF Routing
4) ระบบบันทึกค่าของ OSPF Routing ลงสู่ Configuration file

4.3.3.4) ฟังก์ชันลบค่าโปรโตคอลการหาเส้นทางแบบ OSPF (Remove OSPF Protocol)

เป็นฟังก์ชันที่ใช้สำหรับการลบค่าของโปรโตคอลการหาเส้นทางแบบ OSPF มีขั้นตอนการทำงานดังนี้



รูปที่ 4.9 แสดง Flow Chart ฟังก์ชันลบค่าโปรโตคอลการหาเส้นทางแบบ OSPF (Remove OSPF Protocol)

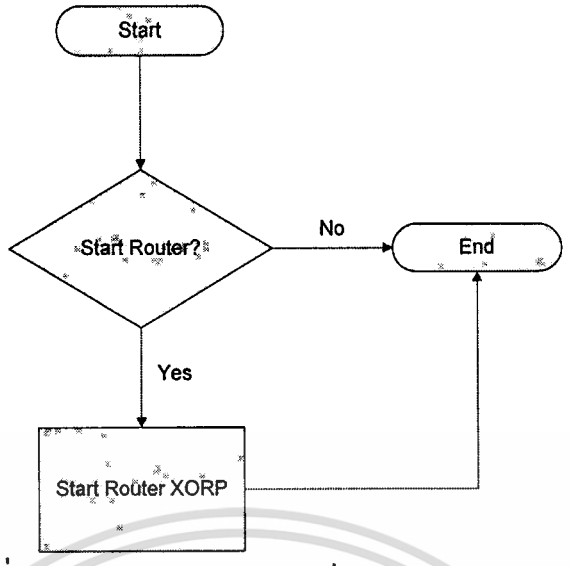
การทำงานของฟังก์ชันนี้เริ่มจาก

1) ผู้ใช้เปิดหน้าเว็บเพจแสดงค่า OSPF Routing ที่มีอยู่ในระบบ
2) ระบบแสดงค่า OSPF Routing ที่มีอยู่ใน Configuration file ขึ้นมาแสดง
3) ผู้ใช้ลบค่า OSPF Routing ที่ไม่ต้องการ
4) ระบบบันทึกค่าของ OSPF Routing ลงสู่ Configuration file
5) ออกจากระบบ เมื่อผู้ใช้ไม่ต้องการลบค่า OSPF Routing

4.3.4 ฟังก์ชันการจัดการทำงานของเราเตอร์

4.3.4.1 ฟังก์ชัน Start Router

เป็นฟังก์ชันการทำงานเพื่อสั่งให้ XORP Router เริ่มทำงานได้ทันทีที่มีการเปลี่ยนแปลงแก้ไขไฟล์ Configuration File โดยที่ไม่ต้องรีสตาร์ทเครื่องเราเตอร์ใหม่ มีการทำงานดังนี้



รูปที่ 4.10 แสดง Flow Chart ฟังก์ชัน Start Router

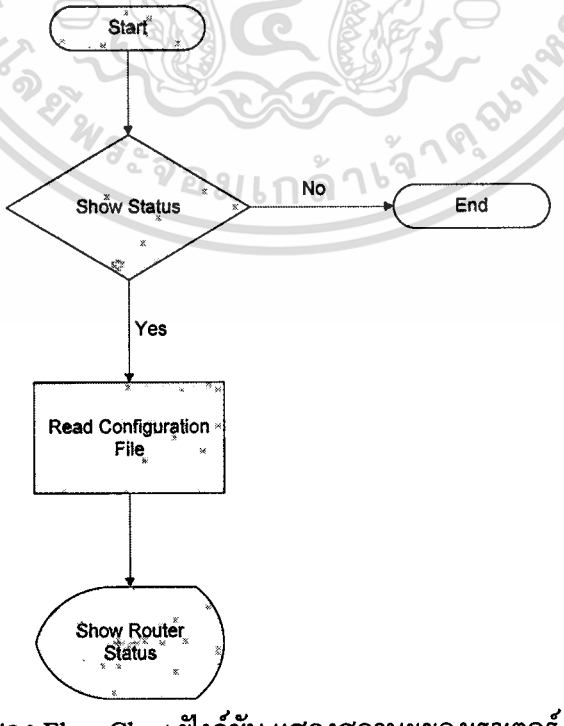
การทำงานของฟังก์ชันนี้เริ่มจาก

1) เมื่อมีการเรียกใช้ฟังก์ชัน Start Router ตัวฟังก์ชันจะไปเรียก shell script ในการรันโปรแกรมเราเตอร์ XORP โดยที่ผู้ใช้ไม่ต้องเริ่มรีสตาร์ทเครื่องใหม่

2) ระบบจะสั่ง Start โปรเซสของ XORP Router Manager เพื่อให้เราเตอร์ XORP ทำงาน

4.3.4.2 ฟังก์ชัน แสดงสถานะของเราเตอร์ Show Status

เป็นฟังก์ชันที่ใช้ในการแสดงสถานะของเราเตอร์คือ ค่าของ Configuration file ต่างๆที่มีอยู่ขณะนั้นเพื่อให้ผู้ใช้สามารถทราบได้ว่าขณะนี้ค่าของ Configuration file ของเราเตอร์ XORP ได้ตั้งค่าไว้เช่นใด มีการทำงานดังนี้



รูปที่ 4.11 แสดง Flow Chart ฟังก์ชัน แสดงสถานะของเราเตอร์ Show Status

การทำงานของฟังก์ชันนี้เริ่มจาก

1) เมื่อมีการเรียกใช้ฟังก์ชัน Show Status ตัวฟังก์ชันจะไปอ่านค่ามาจาก Configuration File ว่าระบบได้ตั้งค่า Routing อะไรเอาไว้บ้าง

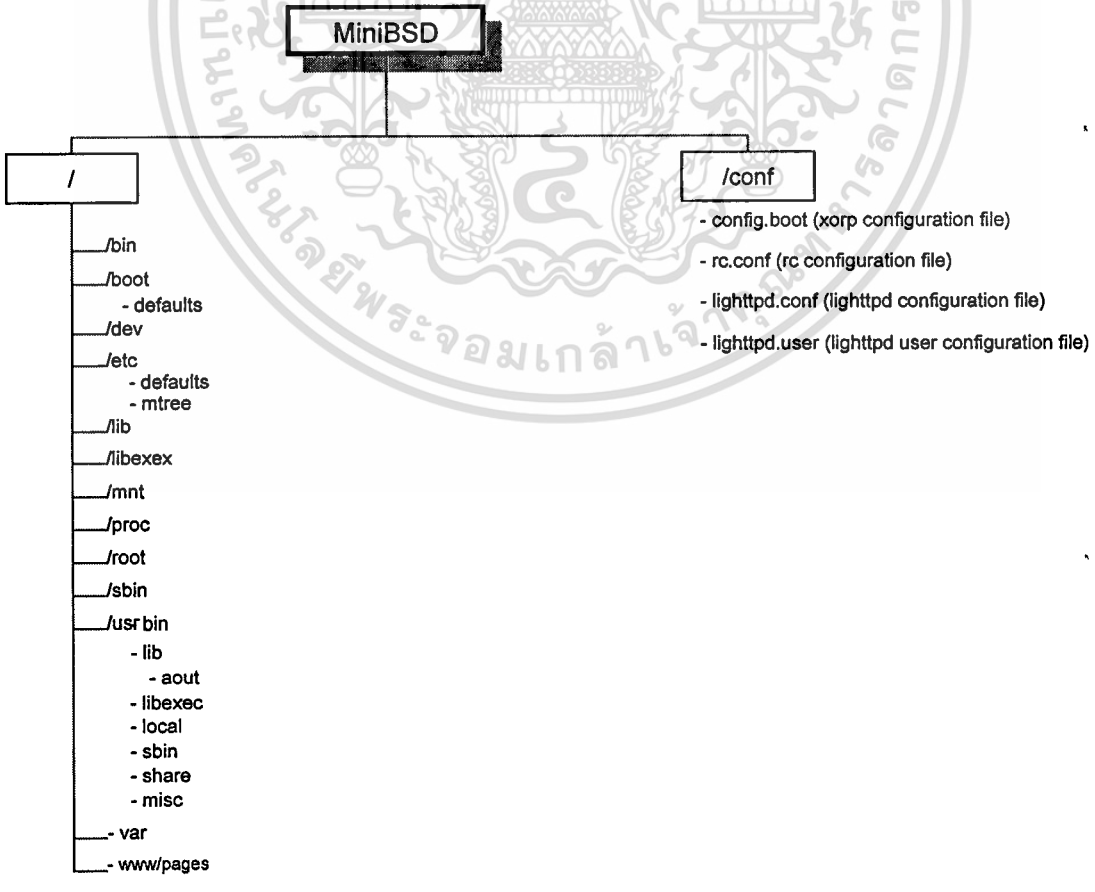
2) ระบบแสดงค่าที่ได้บันทึกเอาไว้ออกมา

4.4 การออกแบบ Embeded Router

ในโครงงานนี้ได้ติดตั้งระบบทั้งหมดคือ ระบบปฏิบัติการ FreeBSD, XORP, lighttpd web server, ตัวแปรภาษา php และโปรแกรมเว็บเบสยูสเซอร์อินเตอร์เฟสที่พัฒนาขึ้นลงสู่ Compact Flash ขนาด 512 MB โดยได้แบ่งออกเป็น slice สอง slice คือ / (root) และ /conf

1) Slice ที่เป็น / (root) เป็น slice ที่มีขนาด 511 MB สำหรับเอาไว้เก็บระบบทั้งหมด คือ FreeBSD, XORP , lighttpd web server, ตัวแปรภาษา php และโปรแกรมเว็บเบสยูสเซอร์อินเตอร์เฟส

2) Slice ที่เป็น /conf เป็น slice ที่มีขนาด 1 MB สำหรับเอาไว้เก็บ configuration file ของ XORP, lighttpd web server, และ rc.conf ของ FreeBSD โครงสร้างการจัดแบ่งแสดงได้ดังรูปต่อไปนี้



รูปที่ 4.12 แสดงโครงสร้างการจัดแบ่ง slice สำหรับ Compact Flash

4.5 การออกแบบโครงสร้างของเว็บเบสยูสเซอร์อินเตอร์เฟส

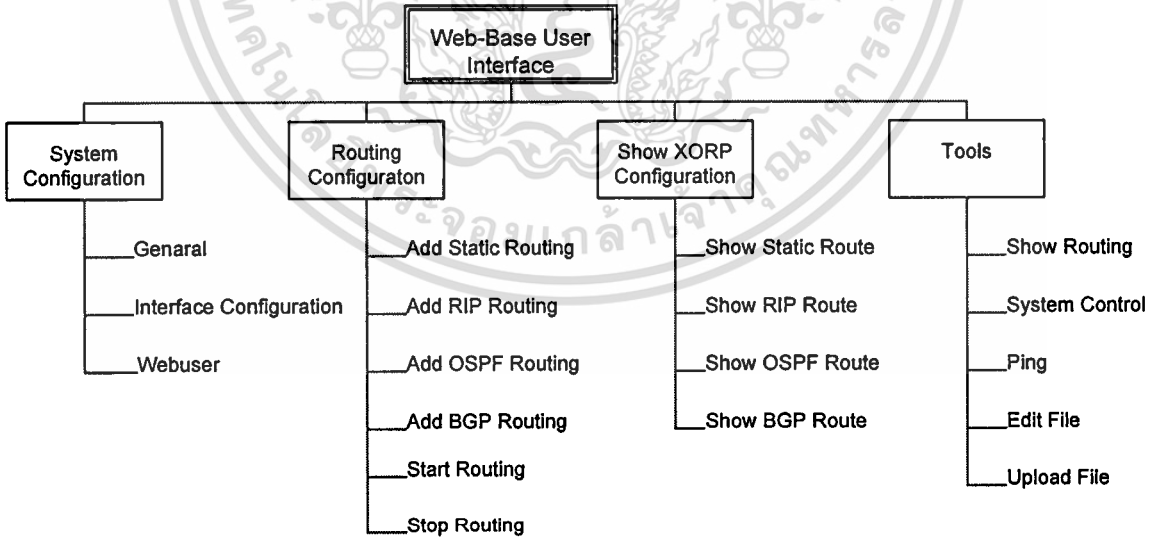
โครงสร้างของเว็บเบสยูสเซอร์อินเตอร์เฟสที่พัฒนาในโครงงานนี้ได้จัดแบ่งตามหน้าที่การทำงานของฟังก์ชันที่ใช้ ได้ 4 เมนูหลักได้แก่

1) System Configuration เป็นเมนูที่สำหรับการจัดการเกี่ยวกับ Interfaces ของระบบคือเลขหมาย IP Address, Subnet mask , Broadcast Address และเกี่ยวกับผู้ใช้ของเว็บเซอร์เวอร์ lighttpd ด้วย

2) Routing Configuration เป็นเมนูที่ใช้เกี่ยวกับการเพิ่มค่าของเราติงโปรโตคอลต่างๆ คือ Static Routing, RIP Routing, OSPF Routing และ BGP Routing นอกจากนี้ยังมีคำสั่งสำหรับเริ่มการทำงานและหยุดการทำงานของการเราติงอีกด้วย

3) Show XORP Configuration เป็นเมนูที่ใช้เกี่ยวกับการแสดงค่าการเราติงของแต่ละโปรโตคอลของ XORP configuration ไฟล์ ที่จะแสดงให้ทราบว่าในไฟล์ configuration ของ XORP นั้นได้มีค่าการเราติงอะไรอยู่บ้าง และยังได้มีคำสั่งสำหรับการลบค่าการเราติงใน configuration ไฟล์ที่ไม่ต้องการออกด้วย

4) Tools เป็นเมนูสำหรับเครื่องมืออำนวยความสะดวกต่างๆ คือ การ Upload ไฟล์, การแก้ไข configuration ไฟล์, การแสดงค่าของ Routing table ของ XORP และของระบบ FreeBSD โดยโครงสร้างที่กล่าวมาทั้งหมดสามารถแสดงได้ดังรูป



รูปที่ 4.13 แสดงโครงสร้างฟังก์ชันของ Web-base User Interface

4.6 สคริปต์ที่ใช้ในการสั่งงานผ่านหน้าเว็บเบสยูสเซอร์อินเตอร์เฟส

การเขียนเว็บเบสยูสเซอร์อินเตอร์เฟสในโครงงานนี้ได้กำหนดให้ผู้ใช้สามารถสั่งงานและดูค่าของเราติงเทเบิลของเราเตอร์ XORP ได้ผ่านหน้าเว็บเบสโดยไม่ต้องรู้คำสั่งสามารถแสดงสคริปต์ต่างๆ ได้ดังนี้

1) สคริปต์สำหรับการสั่งให้เราเตอร์ทำงาน

```
/usr/local/xorp/rtrmgr/xorp_rtrmgr -b /conf/config.boot
```

คำสั่งนี้เป็นคำสั่งที่สั่งให้เราเตอร์ XORP เริ่มทำงานโดยไปอ่าน configuration file ที่ slice /conf ชื่อไฟล์ config.boot

2) สคริปต์สำหรับการสั่งให้เราเตอร์หยุดการทำงาน

```
pkill xorp_rtrmgr
```

คำสั่งนี้เป็นการ kill process ของเราเตอร์ xorp_rtrmgr เพื่อให้เราเตอร์หยุดการทำงาน

3) สคริปต์สำหรับการแสดงค่า Static Routing Table ของ XORP

```
#!/bin/sh
/usr/local/xorp-1.4/rtrmgr/xorpsh <<!
show route table ipv4 unicast static
!
```

4) สคริปต์สำหรับการแสดงค่า RIP Routing Table ของ XORP

```
#!/bin/sh
/usr/local/xorp-1.4/rtrmgr/xorpsh <<!
show route table ipv4 unicast rip
!
```

5) สคริปต์สำหรับการแสดงค่า OSPF Routing Table ของ XORP

```
#!/bin/sh
/usr/local/xorp-1.4/rtrmgr/xorpsh <<!
show ospf4 neighbor detail
!
```

6) สคริปต์สำหรับการแสดงค่า BGP Routing Table ของ XORP

```
#!/bin/sh
/usr/local/xorp-1.4/rtrmgr/xorpsh <<!
show bgp routes
!
```

7) สคริปต์สำหรับการแสดงเราติงเทเบิลของระบบปฏิบัติการ FreeBSD

netstat -nr

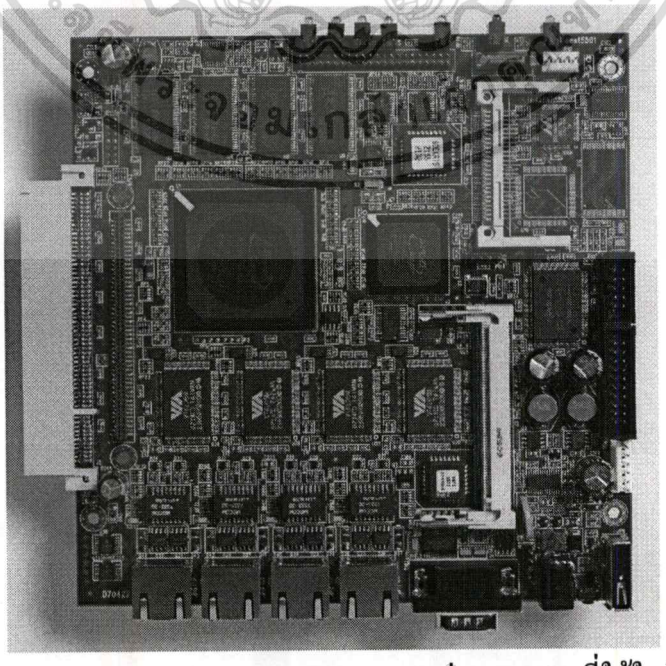
สคริปต์ที่แสดงทั้งหมดเป็น สคริปต์ที่สั่งผ่าน Button ที่หน้าเว็บเพจเพื่อให้ผู้ใช้ระบบได้ตรวจสอบค่าของเราติงเทเบิลของเราเตอร์ XORP ได้โดยไม่ต้องทราบวิธีการเขียนภาษาสคริปต์ สำหรับสคริปต์ข้อที่ 3-6 เป็นสคริปต์ที่สั่งผ่าน เว็บเบสไปสู่ xorpsh (XORP shell) ส่วนสคริปต์ข้อ 7 เป็นการแสดงเราติงเทเบิลของระบบปฏิบัติการ FreeBSD

4.7 การวางแผนโครงสร้างการทดสอบระบบ

การทดสอบระบบอุปกรณ์เราเตอร์ XORP พร้อมเว็บเบสยูสเซอร์อินเตอร์เฟสในโครงงานนี้ได้แยกการทดสอบออกเป็นสองแบบคือ 1) การทดสอบตัวอุปกรณ์เราเตอร์ XORP ที่ใช้ single board ของ Soekris และ 2) การทดสอบการทำงานของตัวเราเตอร์ XORP ในการเราติง packet ข้อมูลตามรูปแบบ topology ที่ได้ออกแบบเอาไว้และทดสอบการเราติงด้วยเราติงโปรโตคอลสามแบบคือ การเราติงด้วย Static Routing protocol, RIP Routing Protocols และ OSPF Routing Protocols

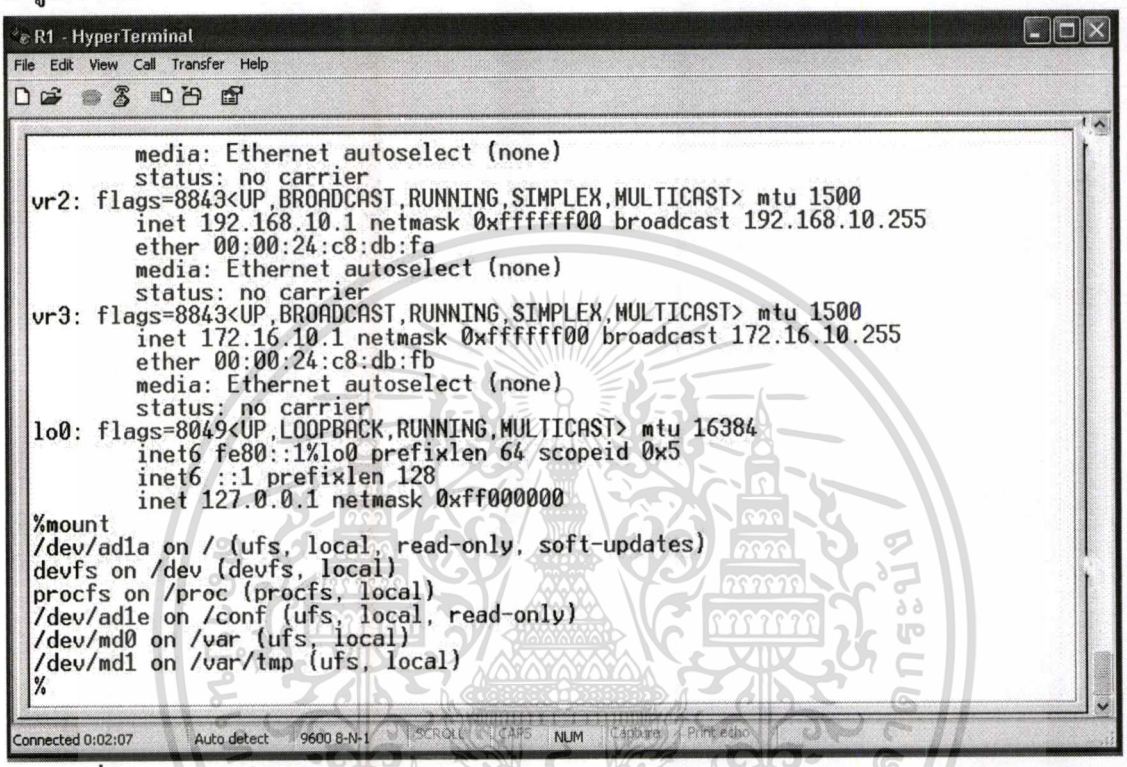
4.7.1) การบูตระบบด้วย Single Board ของ Soekris

หลังจากที่ได้พัฒนาอุปกรณ์เราเตอร์ XORP และได้ติดตั้งระบบทั้งหมดลงสู่ Compact Flash เรียบร้อยแล้วจึงนำ Compact Flash มาติดตั้งลงสู่ Single Board ของ Soekris ดังรูป และบูตระบบเพื่อดูว่าสามารถทำงานได้หรือไม่ โดยภาพ Single Board ของ Soekris แสดงได้ดังรูป ซึ่งเป็นรุ่น net 5501 มีการ์ดแลน ติดตั้งบนบอร์ดจำนวน 4 การ์ด และใช้ CPU ของ AMD



รูปที่ 4.14 แสดงภาพ Single Board ของ Soekris รุ่น net5501 ที่ใช้ในโครงงาน

การบูตระบบด้วย Single Board ของ Soekris สามารถแสดงการบูตได้โดยการใช้การแสดงผลด้วยโปรแกรม Hyper Terminal และผ่านทาง serial port ดังภาพต่อไปนี้เมื่อบูตระบบแล้วจะแสดงให้เห็นว่าระบบที่ทำขึ้นมานั้นได้สร้าง slice เอาไว้สอง slice ที่ mount เป็นแบบ read only คือไม่สามารถเขียนได้ แต่สามารถอ่านได้อย่างเดียว slice แรกคือ / (root) และ slice ที่สองคือ /conf ดังรูปที่ 4.15

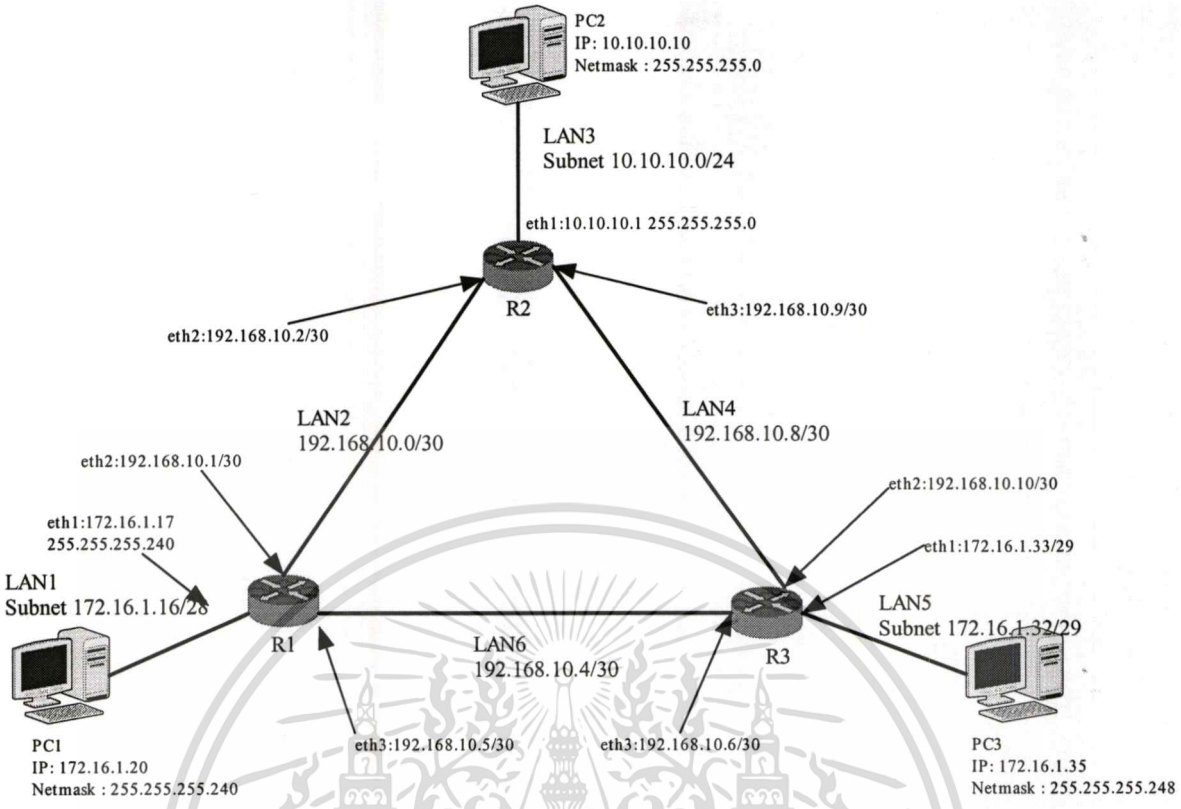


รูปที่ 4.15 แสดงภาพการบูตระบบด้วย Single Board ของ Soekris ผ่านทาง Hyper Terminal

การทดสอบการเราติงของเราเตอร์ XORP ผู้พัฒนาได้ออกแบบให้การทดสอบมีการใช้งานเราเตอร์สามเครื่องเชื่อมต่อกันแบบสองทาง เราเตอร์แต่ละเครื่องมีเครือข่ายภายในเชื่อมต่ออยู่และใช้รูปแบบการเราท์ติงในการทดสอบสามแบบคือ Static Routing , Rip Routing และ OSPF Routing ตามรูปแบบต่อไปนี้

4.7.2) การทดสอบการทำงานของตัวเราเตอร์ XORP ในการเราติง packet ข้อมูล

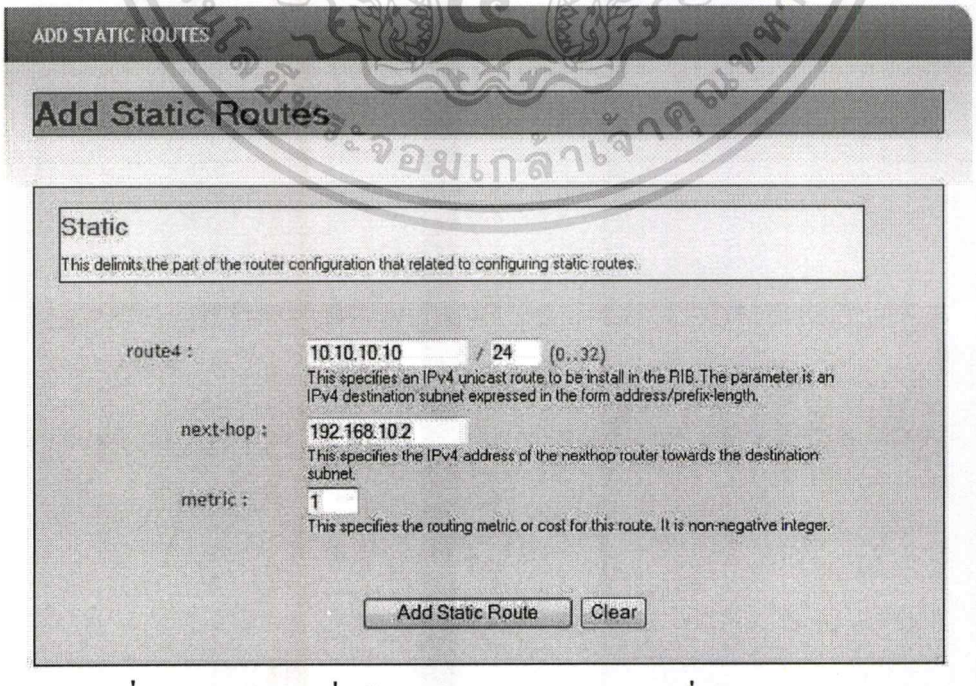
การทดสอบการทำงานของตัวอุปกรณ์เราเตอร์ XORP ในการเราติง packet ข้อมูลในโครงงานนี้ได้ทดสอบในระบบปิดคือใช้โปรแกรม vmware และจำลองเครื่องเป็นอุปกรณ์เราเตอร์ XORP จำนวน 3 เครื่องตาม topology คือ R1, R2 และ R3 ทั้งสามเครื่องมีเว็บเบสยูสเซอร์อินเตอร์เฟสติดตั้งเอาไว้เพื่อการปรับแต่งค่า configuration ไฟล์ ส่วนเครื่องคอมพิวเตอร์ลูกข่ายใช้ระบบปฏิบัติการ Windows XP เป็นระบบปฏิบัติการสองเครื่องคือ PC2 และ PC3 ส่วน PC1 เป็นเครื่อง notebook ที่ติดตั้ง Windows vista ตามรูปที่ 4.16 ที่แสดง



รูปที่ 4.16 แสดงการเชื่อมต่อที่ใช้ในการทดสอบระบบ

4.7.3) การทดสอบการเราติงแบบ Static Routing

ในการเราติงแบบ Static Routing นั้นผู้ใช้จะต้องป้อนค่าของ Network ที่ต้องการส่งต่อ packet และ IP Address ของ Next-hop ที่ต้องการให้เป็นเส้นทางส่งต่อข้อมูล ดังภาพ



รูปที่ 4.17 แสดงการเพิ่มข้อมูลลง configuration file เพื่อทำ Static Route

การเพิ่มข้อมูลดังภาพนั้นจะต้องทำลงในเราเตอร์ทุกๆ เครื่องเพื่อให้ configuration file ของเราเตอร์แต่ละเครื่องทั้งสามเครื่อง สำหรับผลการทำสอบแสดงได้โดยการทดสอบ ping และ trace route ไปยังเครื่อง PC ต่างๆ ดังนี้

4.7.3.1) แสดงการคอนฟิก Static Route

SHOW STATIC ROUTES

XORP Static Route

Route	Next-hop	Action
10.10.10.10/24	192.168.10.2	Remove Route
192.168.10.8/30	192.168.10.2	Remove Route
192.168.10.8/30	192.168.10.6	Remove Route
172.16.1.32/29	192.168.10.6	Remove Route

รูปที่ 4.18 แสดงค่าของ Static Route ใน configuration file ในรูปแบบตาราง

เมื่อได้ทำการเพิ่มค่าคอนฟิกไฟล์เรียบร้อยแล้ว ผู้ใช้ต้องสั่งให้โปรแกรมเราเตอร์ XORP ทำงานก่อนเราเตอร์จึงจะสามารถทำงานได้ดังแสดงในภาพเป็นการสั่งให้เราเตอร์ XORP ทำงานผ่านหน้าเว็บยูสเซอร์อินเทอร์เฟส

ACTIVE ROUTER

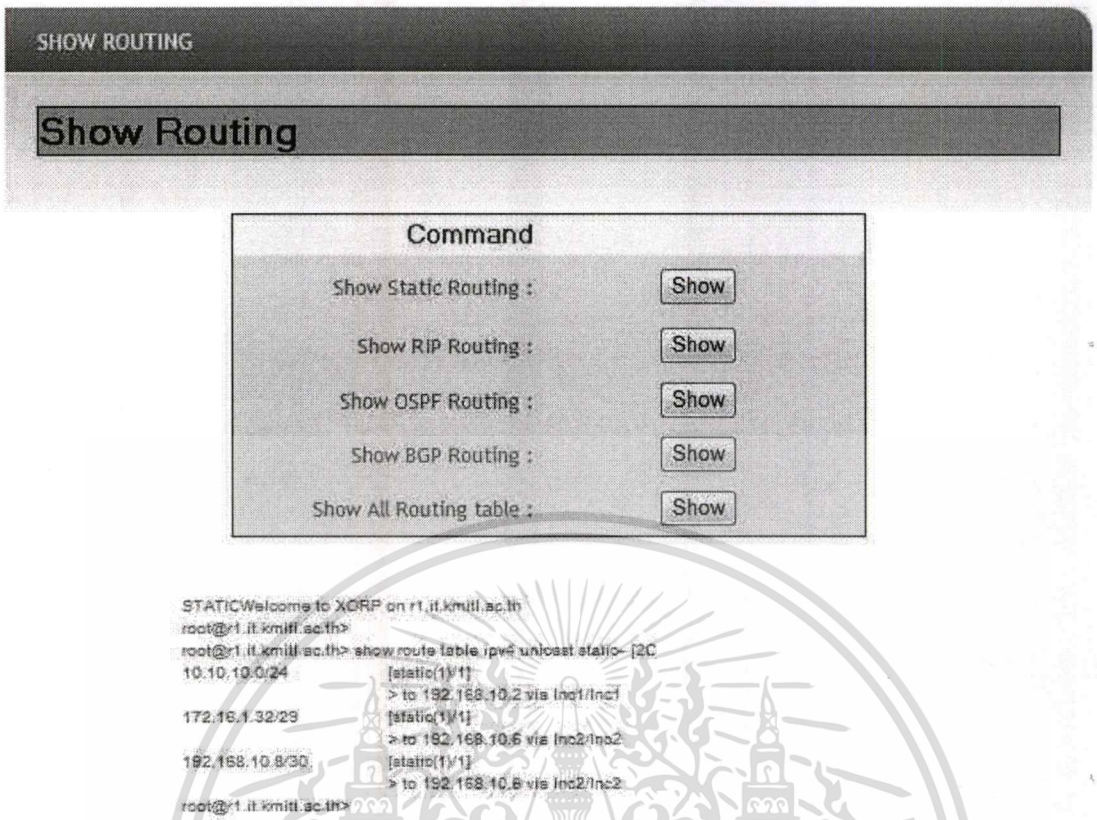
XORP Router Configuration has been Actived.

Please wait for 20 second.
Next time . You can use XORP Router.

[2008/02/14 19:17:34 INFO xorp_fea MFEA] MFEA enabled
[2008/02/14 19:17:34 INFO xorp_fea MFEA] CLI enabled
[2008/02/14 19:17:34 INFO xorp_fea MFEA] CLI started
[2008/02/14 19:17:34 INFO xorp_fea MFEA] MFEA enabled
[2008/02/14 19:17:34 INFO xorp_fea MFEA] CLI enabled
[2008/02/14 19:17:34 INFO xorp_fea MFEA] CLI started

รูปที่ 4.19 แสดงการสั่งงานให้เราเตอร์ทำงาน

ผู้ใช้สามารถสั่งงานให้โปรแกรมเราเตอร์ XORP แสดง Routing table ของ static route ได้ผ่านหน้าเวบเบสยูสเซอร์อินเตอร์เฟสดังแสดงในภาพ



รูปที่ 4.20 แสดงตารางเราติงแบบ static routing

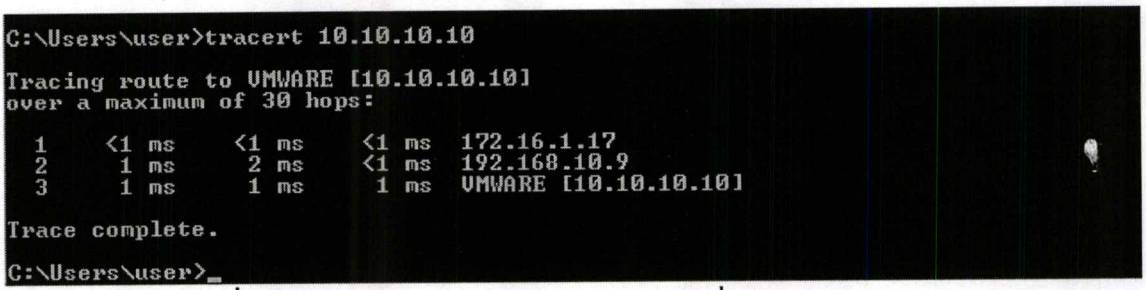
4.7.3.2) ทดสอบการทำงานของการเราท์ติงแบบ Static Routing

1) ทดสอบ ping จากเครื่อง PC1 ไปยังเครื่อง PC3

```
   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::d9a3:3228:d2c6:2c67%13
   IPv4 Address. . . . . . . . . . . : 172.16.1.20
   Subnet Mask . . . . . . . . . . . : 255.255.255.240
   Default Gateway . . . . . . . . . : 172.16.1.17

Tunnel adapter Local Area Connection* 6:

   Connection-specific DNS Suffix  . :
   Default Gateway . . . . . . . . . :

Tunnel adapter Local Area Connection* 9:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::5efe:172.16.1.20%14
   Default Gateway . . . . . . . . . :

Tunnel adapter Local Area Connection* 10:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\user>ping 172.16.1.35

Pinging 172.16.1.35 with 32 bytes of data:

Reply from 172.16.1.35: bytes=32 time=2ms TTL=126
Reply from 172.16.1.35: bytes=32 time=1ms TTL=126
Reply from 172.16.1.35: bytes=32 time=1ms TTL=126
Reply from 172.16.1.35: bytes=32 time=14ms TTL=126

Ping statistics for 172.16.1.35:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 14ms, Average = 4ms

C:\Users\user>_
```

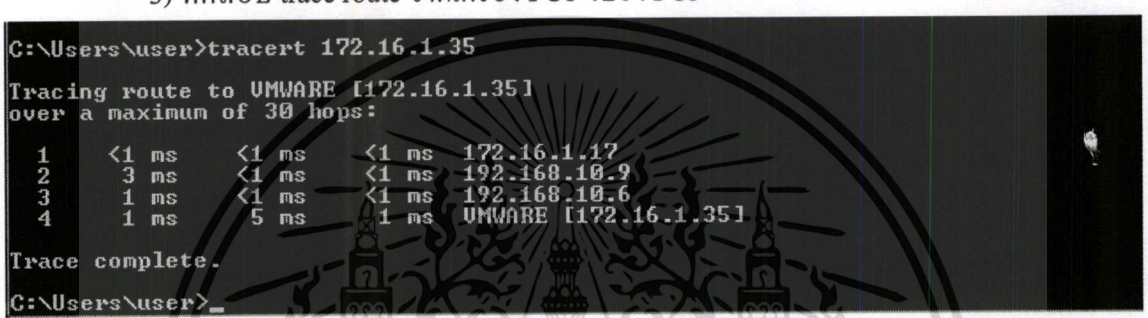
รูปที่ 4.21 แสดงการทดสอบ ping จากเครื่อง PC1 ไปยังเครื่อง PC3

2) ทดสอบ trace route จากเครื่อง PC1 ไปยัง PC2



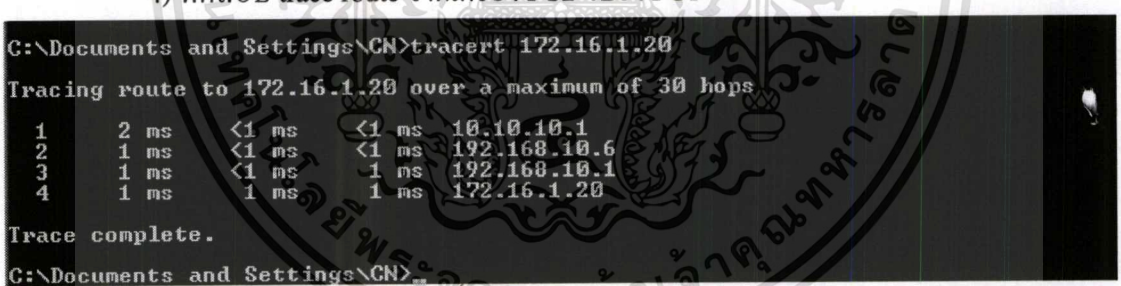
รูปที่ 4.22 แสดงการทดสอบ trace route จากเครื่อง PC1 ไปยัง PC2

3) ทดสอบ trace route จากเครื่อง PC1 ไปยัง PC3



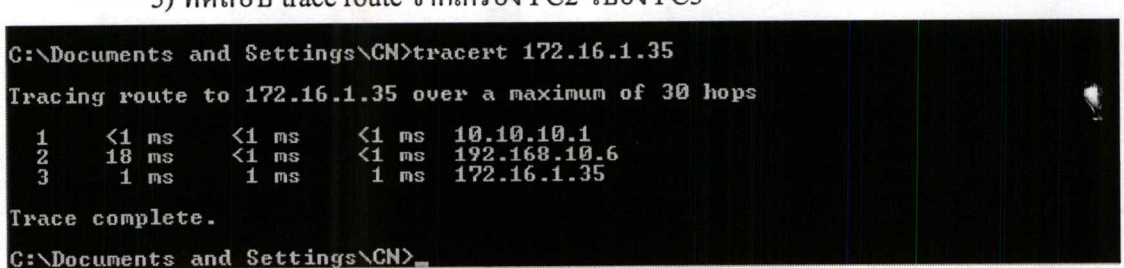
รูปที่ 4.23 แสดงการทดสอบ trace route จากเครื่อง PC1 ไปยัง PC3

4) ทดสอบ trace route จากเครื่อง PC2 ไปยัง PC1



รูปที่ 4.24 แสดงการทดสอบ trace route จากเครื่อง PC2 ไปยัง PC1

5) ทดสอบ trace route จากเครื่อง PC2 ไปยัง PC3



รูปที่ 4.25 แสดงการทดสอบ trace route จากเครื่อง PC2 ไปยัง PC3

6) ทดสอบ trace route จากเครื่อง PC3 ไปยัง PC1

```
C:\Documents and Settings\CN>tracert 172.16.1.20

Tracing route to 172.16.1.20 over a maximum of 30 hops

  1     6 ms     <1 ms     <1 ms  172.16.1.33
  2     2 ms     <1 ms      1 ms  192.168.10.1
  3     1 ms     23 ms      1 ms  172.16.1.20

Trace complete.

C:\Documents and Settings\CN>_
```

รูปที่ 4.26 แสดงการทดสอบ trace route จากเครื่อง PC3 ไปยัง PC1

7) ทดสอบ trace route จากเครื่อง PC3 ไปยัง PC2

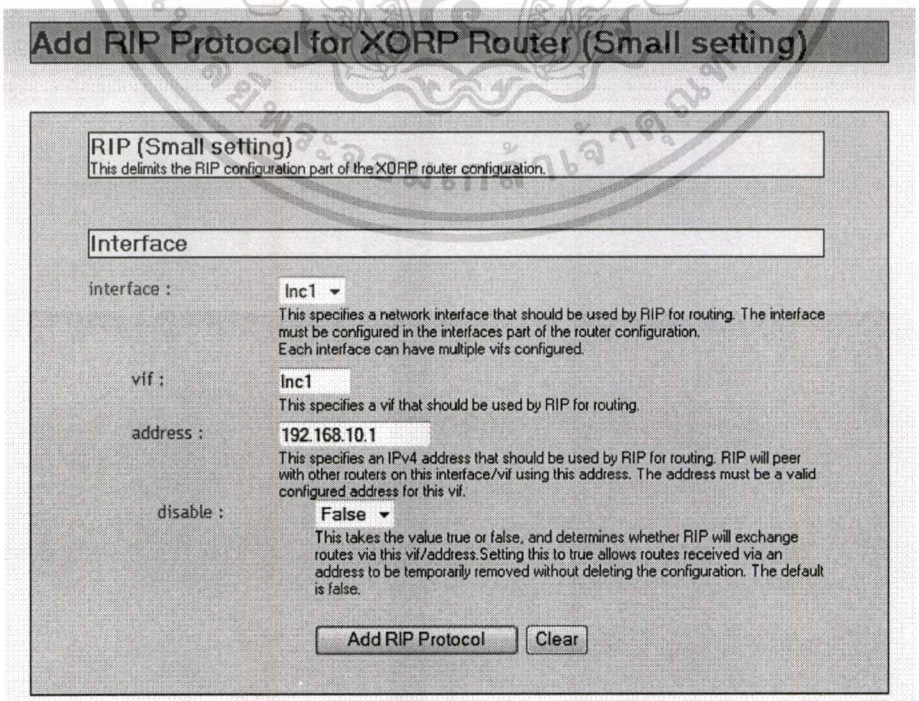
```
C:\Documents and Settings\CN>tracert 10.10.10.10

Tracing route to 10.10.10.10 over a maximum of 30 hops

  1    <1 ms     <1 ms     <1 ms  172.16.1.33
  2     1 ms      1 ms     <1 ms  192.168.10.1
  3     1 ms     <1 ms      2 ms  192.168.10.9
  4     1 ms      1 ms      1 ms  10.10.10.10

Trace complete.

C:\Documents and Settings\CN>_
```

รูปที่ 4.27 แสดงการทดสอบ trace route จากเครื่อง PC3 ไปยัง PC2

4.7.4) การเราติงแบบ RIP Routing

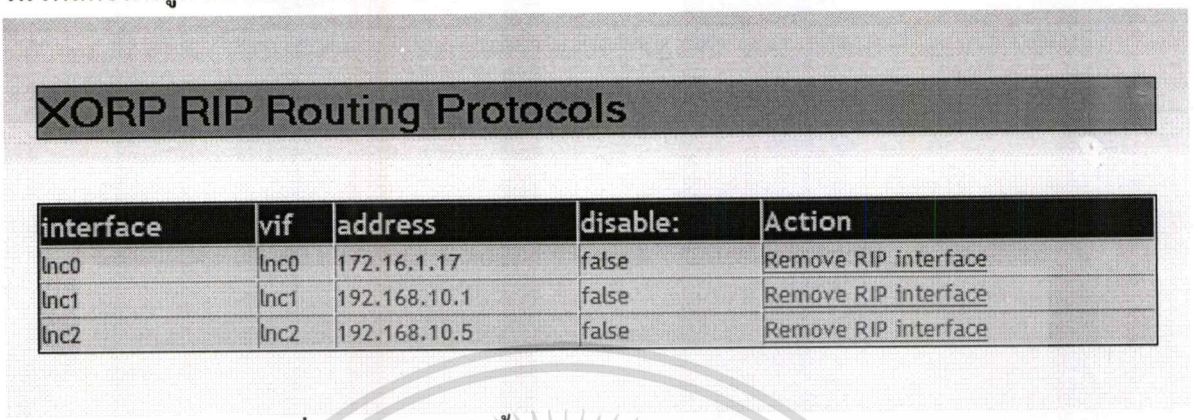
การ Routing แบบ RIP เป็นการเราท์ติงที่เราเตอร์จะประกาศ Routing table ของตัวเองทั้งตารางออกไปยังอินตอร์เฟสที่รัน RIP อยู่โดยการตั้งค่า ของโปรแกรม XORP สามารถคอนฟิกได้ตามภาพต่อไปนี้ซึ่งเป็นตัวอย่างการตั้งค่าการเราติงแบบ RIP ของ R1

4.7.4.1) แสดงการตั้งค่าของ RIP Routing



รูปที่ 4.28 แสดงการเพิ่มค่าการเราท์ติงแบบ RIP routing

เมื่อทำการตั้งค่าทุกอินเตอร์เฟสของเราเตอร์แล้วผู้ใช้ต้องสั่งงานให้โปรแกรมเราเตอร์ XORP ทำงานเช่นเดียวกับการเราท์ติงแบบ static routing ผู้ใช้สามารถแสดงค่าการเราท์ติงแบบ RIP ในไฟล์คอนฟิกูเรชั่นได้ดังแสดงในภาพต่อไปนี้



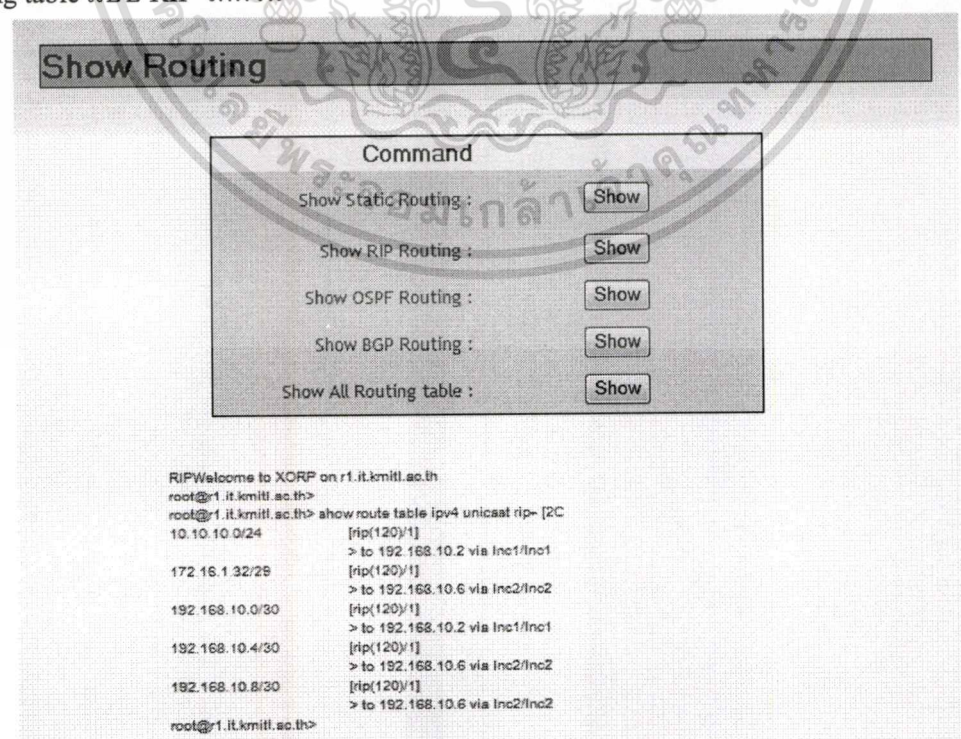
XORP RIP Routing Protocols

interface	vif	address	disable:	Action
lnc0	lnc0	172.16.1.17	false	Remove RIP interface
lnc1	lnc1	192.168.10.1	false	Remove RIP interface
lnc2	lnc2	192.168.10.5	false	Remove RIP interface

รูปที่ 4.29 แสดงการตั้งค่าการเราท์ติงแบบ RIP routing

ถ้าผู้ใช้ไม่ต้องการการเราท์ติงแบบ RIP ของอินเตอร์เฟสใด สามารถสั่งให้โปรแกรมลบค่าของเราติงของอินเตอร์เฟสนั้นออกจากค่าใน configuration file นั้นๆได้ทันทีโดยการ click ที่ Remove RIP interface ด้านท้ายของแต่ละบรรทัด ตัวโปรแกรมจะไปลบค่าใน configuration file ของ XORP

การสั่งงานให้เราเตอร์ทำงานเพื่อเราท์ติงแบบ RIP สามารถทำได้เช่นเดียวกับการเราท์ติงแบบ static routing เมื่อสั่งงานให้โปรแกรม XORP ทำการเราท์ติงแล้วสามารถแสดงผลการเราท์ใน Routing table แบบ RIP ได้ดังนี้



Show Routing

Command

Show Static Routing : Show

Show RIP Routing : Show

Show OSPF Routing : Show

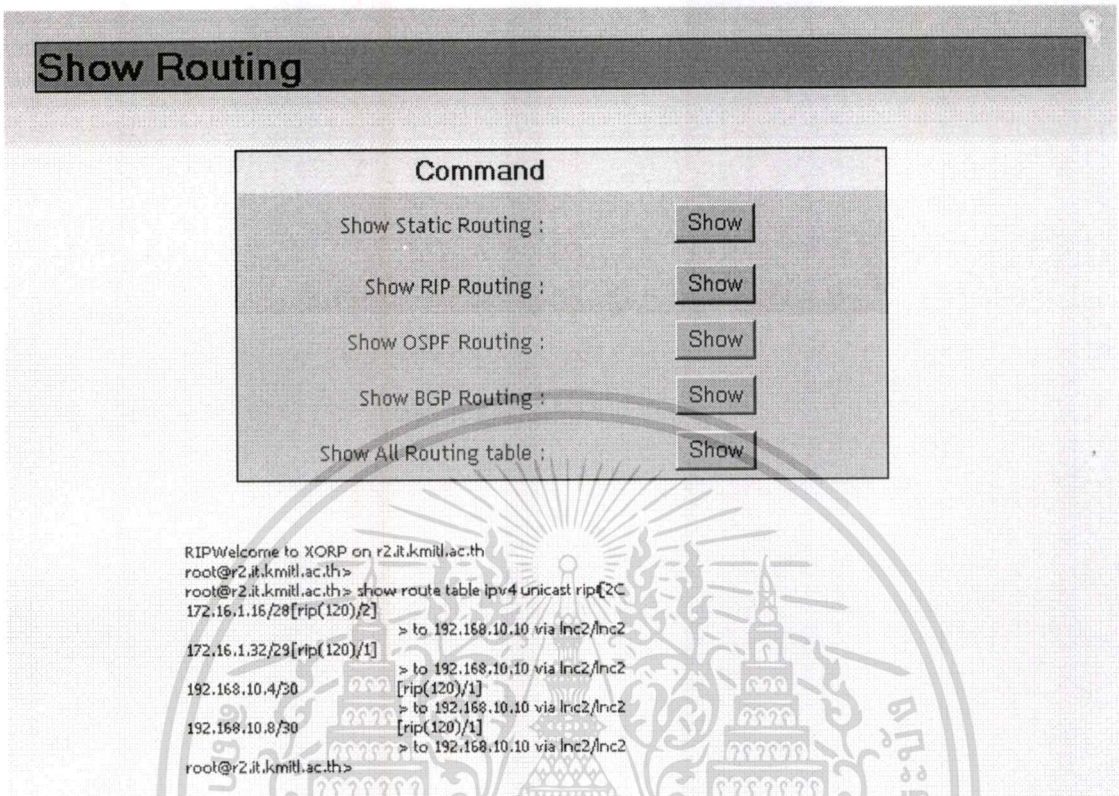
Show BGP Routing : Show

Show All Routing table : Show

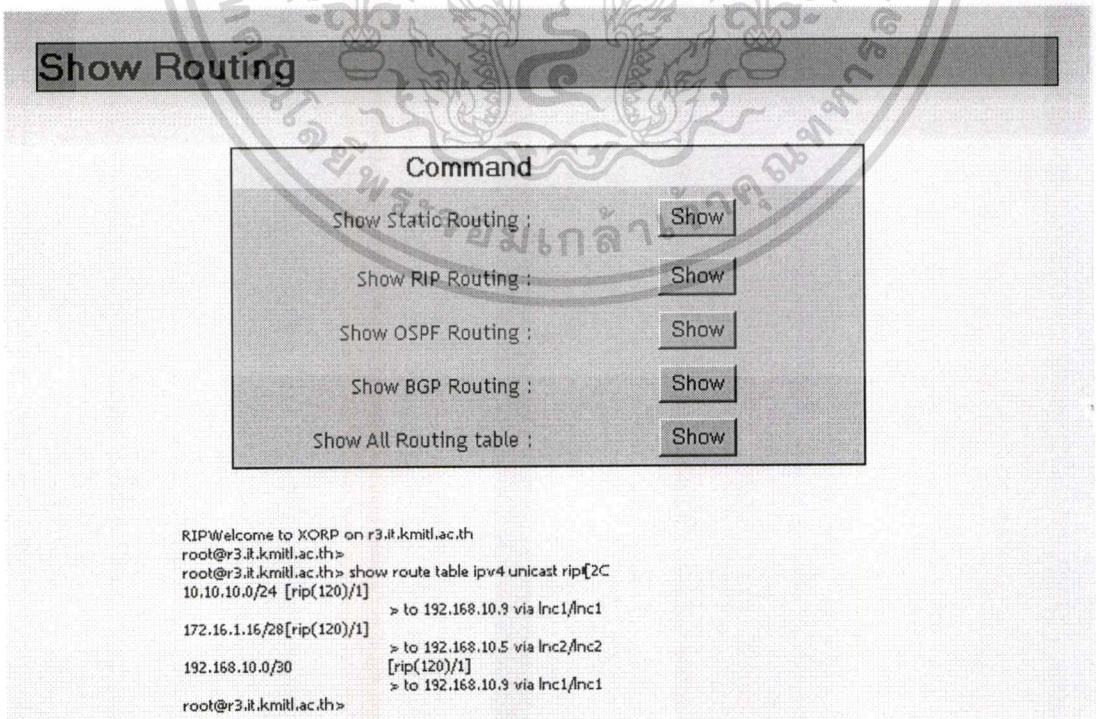
```
RIPWelcome to XORP on r1.it.kmitl.ac.th
root@r1.it.kmitl.ac.th>
root@r1.it.kmitl.ac.th> show route table ipv4 unicast rip> [2C
10.10.10.0/24          [rip(120)/1]
                       > to 192.168.10.2 via lnc1/lnc1
172.16.1.32/29         [rip(120)/1]
                       > to 192.168.10.6 via lnc2/lnc2
192.168.10.0/30        [rip(120)/1]
                       > to 192.168.10.2 via lnc1/lnc1
192.168.10.4/30        [rip(120)/1]
                       > to 192.168.10.6 via lnc2/lnc2
192.168.10.8/30        [rip(120)/1]
                       > to 192.168.10.6 via lnc2/lnc2
root@r1.it.kmitl.ac.th>
```

รูปที่ 4.30 แสดงเราติงเทเบิลแบบ RIP ของ R1

การตั้งค่าของเราติงแบบ RIP ต้องทำที่เราเตอร์ทุกตัวในเครือข่ายซึ่งการทดลองนั้นได้ตั้งค่าแบบ RIP ทั้งสามตัวและรัน RIP Routing Protocol ได้ผลการรันแสดงได้ดังภาพต่อไปนี้



รูปที่ 4.31 แสดงเราติงเทเบิลแบบ RIP ของ R2



รูปที่ 4.32 แสดงเราติงเทเบิลแบบ RIP ของ R3

4.7.4.2) แสดงการทำงานของ RIP Routing

ผลการทำงานของโปรแกรมเราเตอร์ XORP ที่ทำงานกับ RIP Routing Protocol สามารถทำการทดสอบได้โดยการ ping และ trace route ตามผลการทดลองต่อไปนี้

1) ทดสอบการ ping จาก PC1 ไป PC2

```
Ethernet adapter Local Area Connection 3:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::d9a3:3228:d2c6:2c67%12
   IPv4 Address. . . . . . . . . . . : 172.16.1.20
   Subnet Mask . . . . . . . . . . . : 255.255.255.240
   Default Gateway . . . . . . . . . : 172.16.1.17

Tunnel adapter Local Area Connection* 6:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 9:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::5efe:172.16.1.20%13
   Default Gateway . . . . . . . . . :

Tunnel adapter Local Area Connection* 10:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\user>ping 10.10.10.10

Pinging 10.10.10.10 with 32 bytes of data:

Reply from 10.10.10.10: bytes=32 time=11ms TTL=126
Reply from 10.10.10.10: bytes=32 time=14ms TTL=126
Reply from 10.10.10.10: bytes=32 time=1ms TTL=126
Reply from 10.10.10.10: bytes=32 time=1ms TTL=126

Ping statistics for 10.10.10.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 14ms, Average = 6ms

C:\Users\user>_
```

รูปที่ 4.33 แสดงการทดสอบการ ping จาก PC1 ไป PC2

2) ทดสอบการ ping จาก PC3 มายัง PC1

```
Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 172.16.1.35
        Subnet Mask . . . . . . . . . . . : 255.255.255.248
        Default Gateway . . . . . . . . . : 172.16.1.33

C:\Documents and Settings\CN>ping 172.16.1.20

Pinging 172.16.1.20 with 32 bytes of data:

Reply from 172.16.1.20: bytes=32 time=6ms TTL=126
Reply from 172.16.1.20: bytes=32 time=1ms TTL=126
Reply from 172.16.1.20: bytes=32 time=1ms TTL=126
Reply from 172.16.1.20: bytes=32 time=1ms TTL=126

Ping statistics for 172.16.1.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 6ms, Average = 2ms

C:\Documents and Settings\CN>_
```

รูปที่ 4.34 แสดงการทดสอบการ ping จาก PC3 ไป PC1

3) ทดสอบการ ping จาก PC2 มายัง PC3

```
Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 10.10.10.10
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 10.10.10.1

C:\Documents and Settings\CN>ping 172.16.1.35

Pinging 172.16.1.35 with 32 bytes of data:

Reply from 172.16.1.35: bytes=32 time=3ms TTL=126
Reply from 172.16.1.35: bytes=32 time=5ms TTL=126
Reply from 172.16.1.35: bytes=32 time=1ms TTL=126
Reply from 172.16.1.35: bytes=32 time=2ms TTL=126

Ping statistics for 172.16.1.35:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 5ms, Average = 2ms

C:\Documents and Settings\CN>_
```

รูปที่ 4.35 แสดงการทดสอบการ ping จาก PC2 ไป PC3

4) ทดสอบการ trace route จาก PC1 ไปยัง PC2

```
C:\Users\user>tracert 10.10.10.10

Tracing route to VMWARE [10.10.10.10]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  172.16.1.17
  2    <1 ms    <1 ms    <1 ms  192.168.10.2
  3     1 ms    <1 ms    <1 ms  VMWARE [10.10.10.10]

Trace complete.

C:\Users\user>_
```

รูปที่ 4.36 แสดงการทดสอบการ trace route จาก PC1 ไปยัง PC2

5) ทดสอบการ trace route จาก PC1 ไปยัง PC3

```
C:\Users\user>tracert 172.16.1.35

Tracing route to VMWARE [172.16.1.35]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  172.16.1.17
  2    <1 ms    <1 ms    <1 ms  192.168.10.6
  3     1 ms     1 ms    <1 ms  VMWARE [172.16.1.35]

Trace complete.

C:\Users\user>_
```

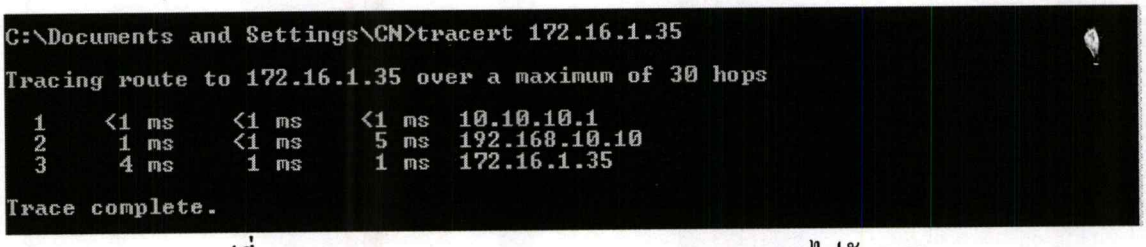
รูปที่ 4.37 แสดงการทดสอบการ trace route จาก PC1 ไปยัง PC3

6) ทดสอบการ trace route จาก PC2 ไปยัง PC1

```
C:\Documents and Settings\CN>tracert 172.16.1.20

Tracing route to 172.16.1.20 over a maximum of 30 hops

  1    30 ms    <1 ms    <1 ms  10.10.10.1
  2     1 ms     1 ms    <1 ms  192.168.10.1
  3     1 ms    11 ms     1 ms  172.16.1.20

Trace complete.
```

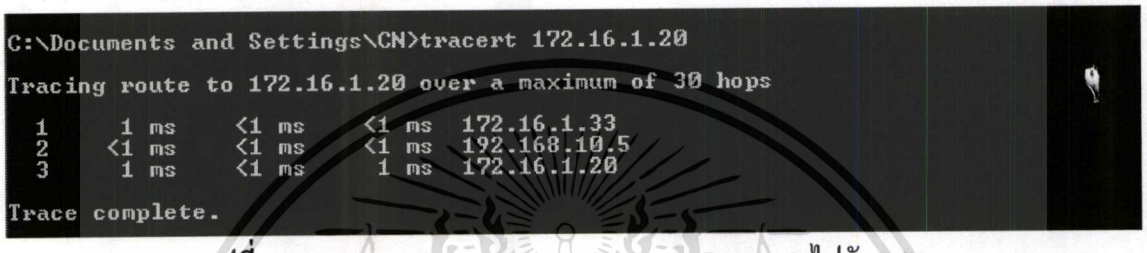
รูปที่ 4.38 แสดงการทดสอบการ trace route จาก PC2 ไปยัง PC1

7) ทดสอบการ trace route จาก PC2 ไปยัง PC3



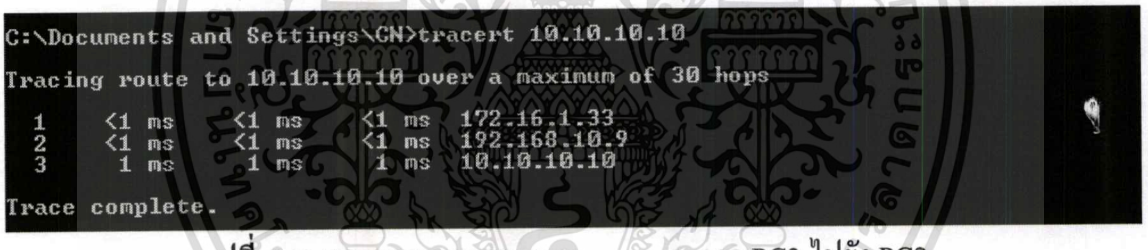
รูปที่ 4.39 แสดงการทดสอบการ trace route จาก PC2 ไปยัง PC3

8) ทดสอบการ trace route จาก PC3 ไปยัง PC1



รูปที่ 4.40 แสดงการทดสอบการ trace route จาก PC3 ไปยัง PC1

9) ทดสอบการ trace route จาก PC3 ไปยัง PC2



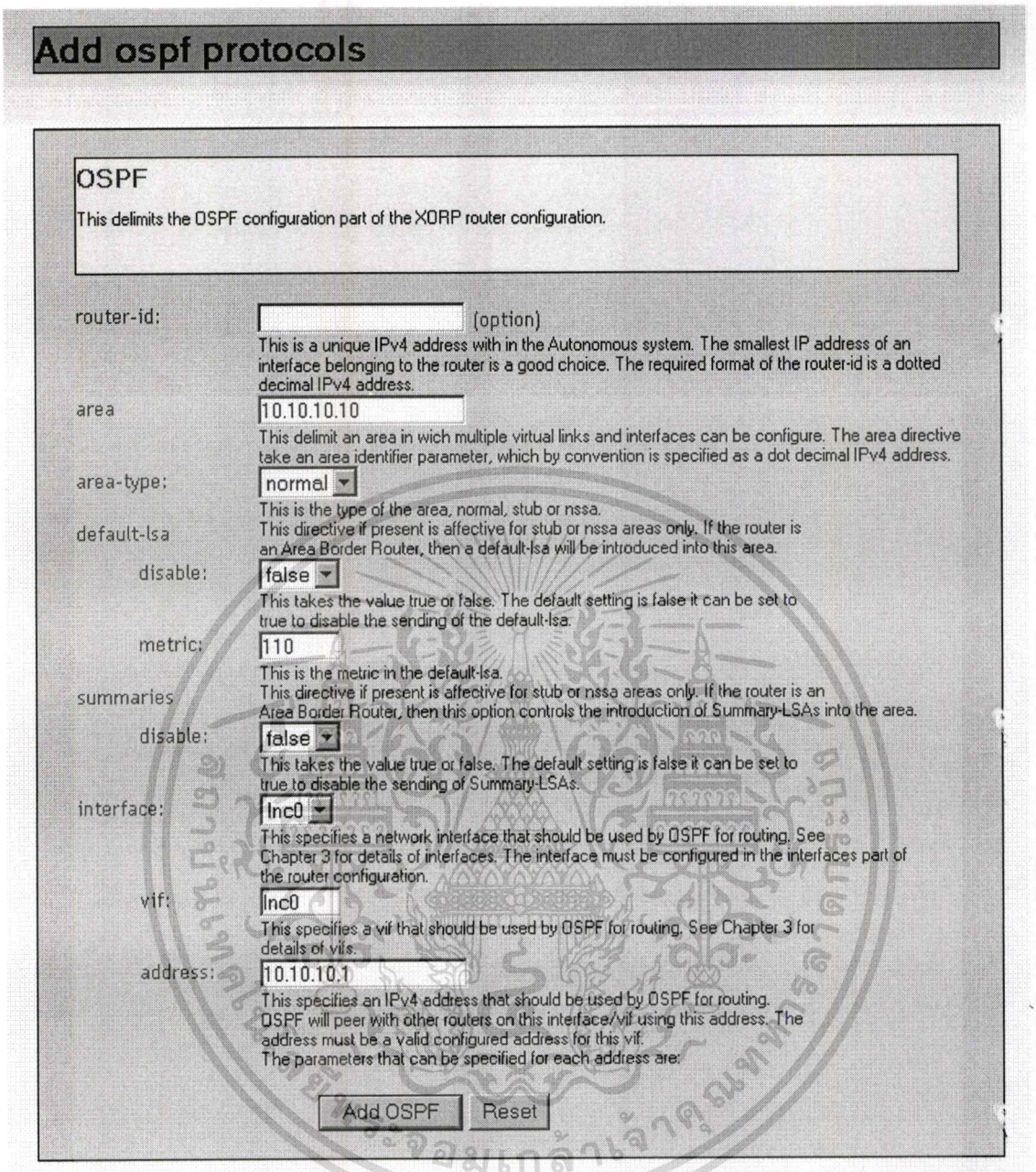
รูปที่ 4.41 แสดงการทดสอบการ trace route จาก PC3 ไปยัง PC2

4.7.5) การเราติงแบบ OSPF Routing

การเราท์ติงแบบ OSPF เป็นโปรโตคอลการเราท์ติงที่เราเตอร์จะประกาศสถานะของตัวเองออกไปให้เราเตอร์เพื่อนบ้านได้รับทราบแล้วเราเตอร์เพื่อนบ้านจะสร้างตารางเราท์ติง ของตัวเองเพื่อใช้ในการส่ง packet ออกไปยังเราเตอร์เพื่อนบ้าน

4.7.5.1) แสดงการตั้งค่าของ OSPF Routing

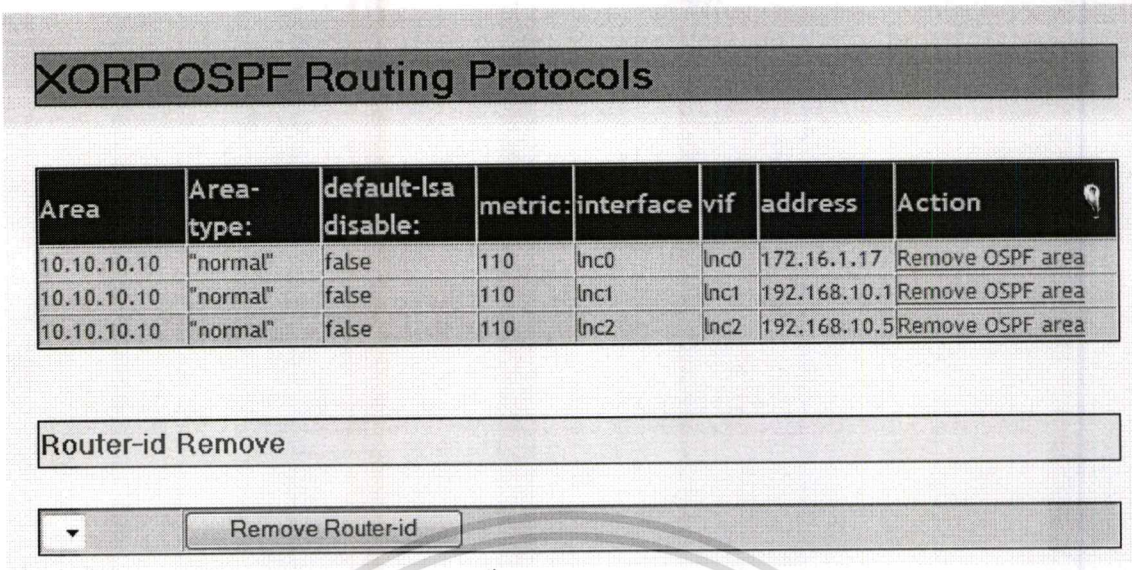
การตั้งค่าการเราท์ติงของ OSPF ผ่านหน้าเว็บเบสยูสเซอร์อินเตอร์เฟสของโปรแกรมสามารถทำได้ดังตัวอย่างต่อไปนี้



รูปที่ 4.42 แสดงการเพิ่มค่าการเราท์ติงแบบ OSPF routing

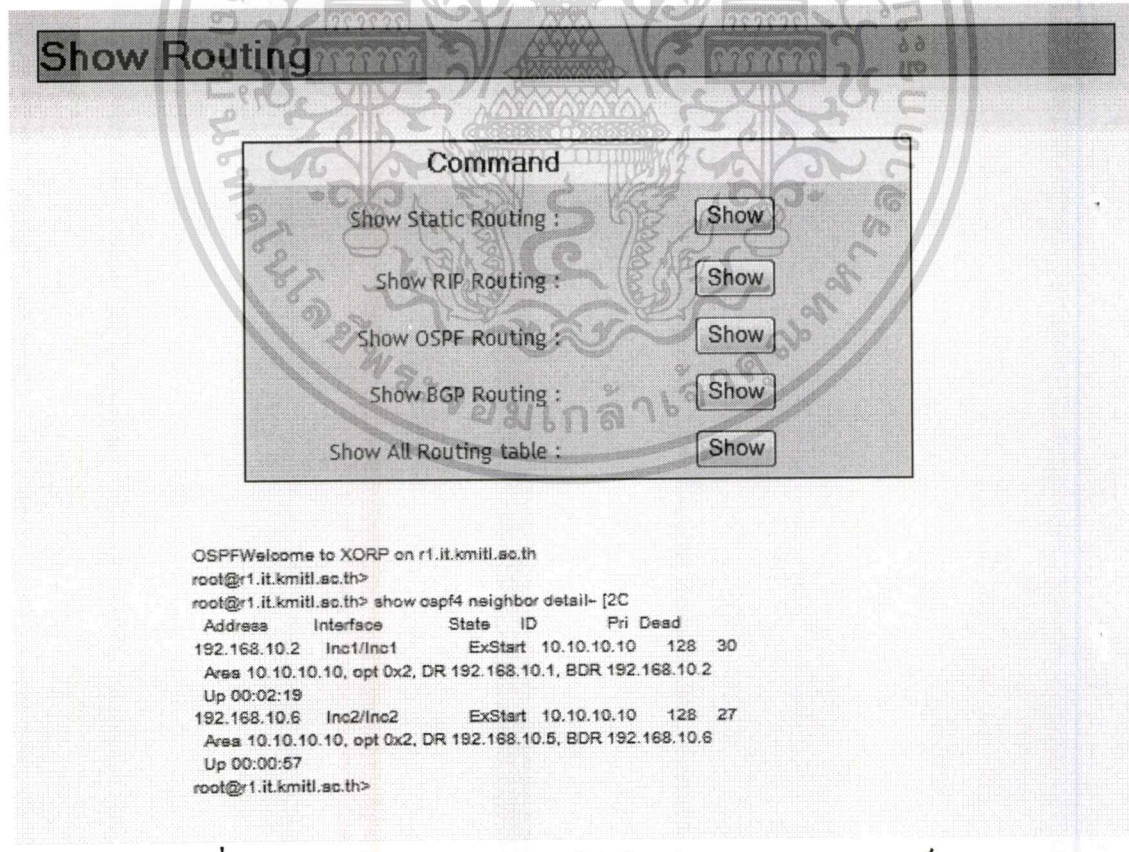
การเพิ่มค่าเราติงของโปรโตคอล OSPF ผู้ใช้ต้องเพิ่มค่าของทุกเราเตอร์ตามที่ต้องการให้เราเตอร์นั้นทำการเราท์ติงแบบ OSPF ออกไปในการทดลองครั้งนี้ได้เพิ่มค่าการ Routing แบบ OSPF ลงในเราเตอร์ทั้งสามตัวแล้วทดลองเราท์ติงทั้งสามเครื่องด้วยโปรโตคอลแบบ OSPF

เมื่อได้เพิ่มค่า เราท์ติงแบบ OSPF ลงใน configuration file แล้วสามารถทำการแสดงค่า configuration file ของโปรโตคอล OSPF ในไฟล์ได้ดังแสดงในภาพ

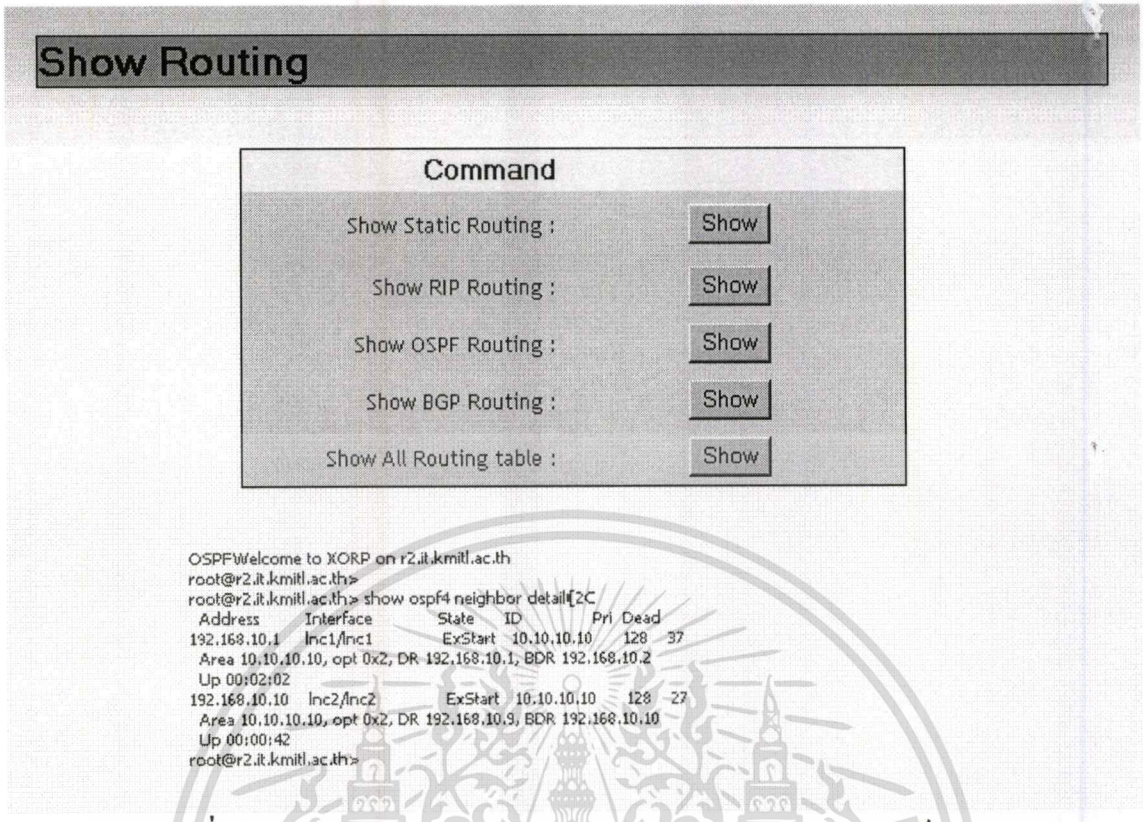


รูปที่ 4.43 แสดงการตั้งค่าการเราท์ติงแบบ OSPF routing

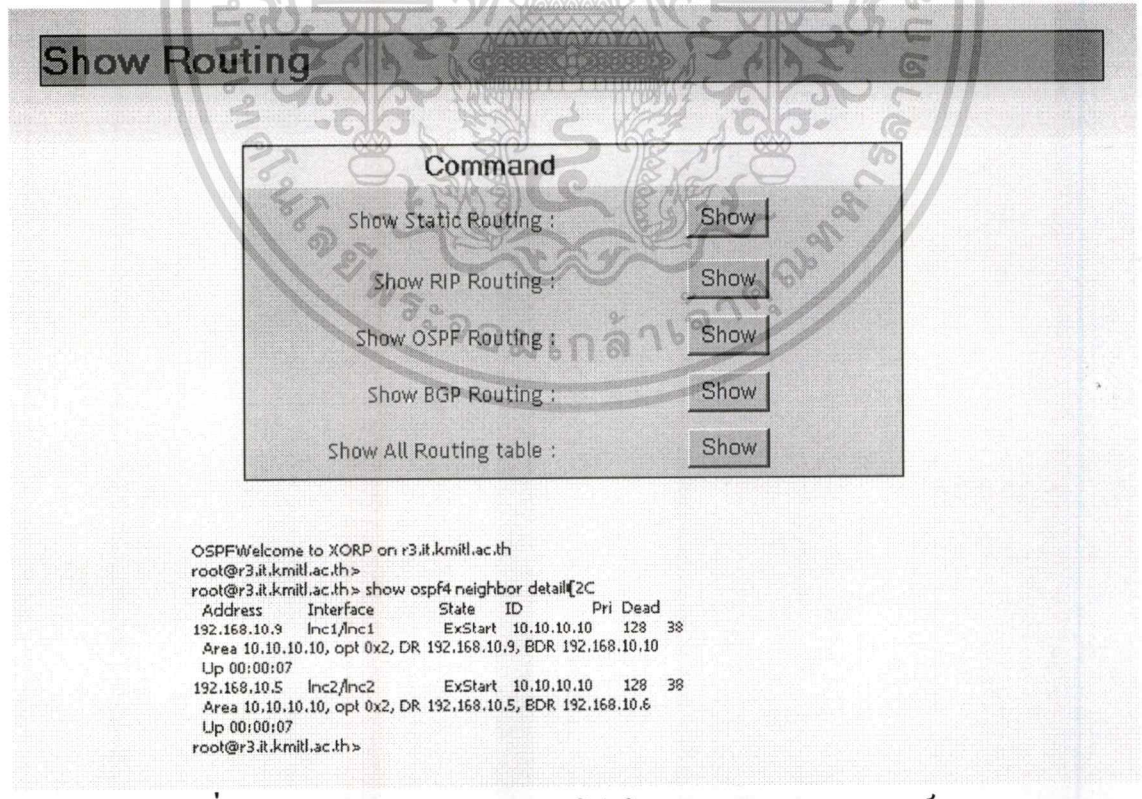
หลังจากที่ผู้ใช้ได้ทำการเพิ่มค่าการเราท์ติงให้กับเราเตอร์ทั้งสามเครื่องเป็นที่เรียบร้อยแล้ว ผู้ใช้ต้องสั่งให้เราเตอร์ทำงานเช่นเดียวกับการเราท์ติงประเภทอื่นๆและสามารถสั่งให้โปรแกรมแสดงค่าการเราท์ออกมาทางเว็บเพจได้ดังภาพ



รูปที่ 4.44 แสดง Routing table ของโปรโตคอลแบบ OSPF เราเตอร์ R1



รูปที่ 4.45 แสดง Routing table ของโปรโตคอลแบบ OSPF เราเตอร์ R2



รูปที่ 4.46 แสดง Routing table ของโปรโตคอลแบบ OSPF เราเตอร์ R3

4.7.5.2) แสดงการทำงานของ OSPF Routing

การทำงานของเราเตอร์ที่ทำการทดลองเราท์ติงโปรโตคอลแบบ OSPF สามารถทำการทดสอบได้จากการ ping และ trace route ในการทดลองนี้ได้ผลการทดลองตามนี้

1) ทดสอบการ ping จาก PC1 ไป PC2

```
Ethernet adapter Local Area Connection 3:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::d9a3:3228:d2c6:2c67%12
   IPv4 Address. . . . . . . . . . . : 172.16.1.20
   Subnet Mask . . . . . . . . . . . : 255.255.255.240
   Default Gateway . . . . . . . . . : 172.16.1.17

Tunnel adapter Local Area Connection* 6:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 9:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::5efe:172.16.1.20%13
   Default Gateway . . . . . . . . . :

Tunnel adapter Local Area Connection* 10:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\user>ping 10.10.10.10

Pinging 10.10.10.10 with 32 bytes of data:

Reply from 10.10.10.10: bytes=32 time=1ms TTL=125
Reply from 10.10.10.10: bytes=32 time=1ms TTL=125
Reply from 10.10.10.10: bytes=32 time=1ms TTL=125
Reply from 10.10.10.10: bytes=32 time=24ms TTL=125

Ping statistics for 10.10.10.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 24ms, Average = 6ms
```

รูปที่ 4.47 แสดงการทดสอบการ ping จาก PC1 ไป PC2

2) ทดสอบการ ping จาก PC2 ไป PC3

```
Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 10.10.10.10
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 10.10.10.1

C:\Documents and Settings\CN>ping 172.16.1.35

Pinging 172.16.1.35 with 32 bytes of data:

Reply from 172.16.1.35: bytes=32 time=6ms TTL=125
Reply from 172.16.1.35: bytes=32 time=1ms TTL=125
Reply from 172.16.1.35: bytes=32 time=1ms TTL=125
Reply from 172.16.1.35: bytes=32 time=2ms TTL=125

Ping statistics for 172.16.1.35:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 6ms, Average = 2ms
```

รูปที่ 4.48 แสดงการทดสอบการ ping จาก PC2 ไป PC3

3) ทดสอบการ ping จาก PC3 ไป PC1

```
Windows IP Configuration

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 172.16.1.35
        Subnet Mask . . . . . . . . . . . : 255.255.255.248
        Default Gateway . . . . . . . . . : 172.16.1.33

C:\Documents and Settings\CN>ping 172.16.1.20

Pinging 172.16.1.20 with 32 bytes of data:

Reply from 172.16.1.20: bytes=32 time=2ms TTL=125
Reply from 172.16.1.20: bytes=32 time=12ms TTL=125
Reply from 172.16.1.20: bytes=32 time=1ms TTL=125
Reply from 172.16.1.20: bytes=32 time=1ms TTL=125

Ping statistics for 172.16.1.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 12ms, Average = 4ms
```

รูปที่ 4.49 แสดงการทดสอบการ ping จาก PC3 ไป PC1

4) ทดสอบการ trace route จาก PC1 ไปยัง PC2

```
C:\Users\user>tracert 10.10.10.10

Tracing route to VMWARE [10.10.10.10]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  172.16.1.17
  2     1 ms    <1 ms    <1 ms  192.168.10.9
  3     1 ms     1 ms     1 ms  VMWARE [10.10.10.10]

Trace complete.
```

รูปที่ 4.50 แสดงการทดสอบการ trace route จาก PC1 ไปยัง PC2

5) ทดสอบการ trace route จาก PC1 ไปยัง PC3

```
C:\Users\user>tracert 172.16.1.35

Tracing route to VMWARE [172.16.1.35]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  172.16.1.17
  2     1 ms    <1 ms     1 ms  192.168.10.9
  3     1 ms    <1 ms    <1 ms  192.168.10.6
  4     1 ms    13 ms     1 ms  VMWARE [172.16.1.35]

Trace complete.
```

รูปที่ 4.51 แสดงการทดสอบการ trace route จาก PC1 ไปยัง PC3

6) ทดสอบการ trace route จาก PC2 ไปยัง PC1

```
C:\Documents and Settings\CN>tracert 172.16.1.20

Tracing route to 172.16.1.20 over a maximum of 30 hops

  1     1 ms    <1 ms    11 ms  10.10.10.1
  2     1 ms    14 ms    <1 ms  192.168.10.6
  3    13 ms     2 ms     1 ms  192.168.10.1
  4     1 ms     1 ms     1 ms  172.16.1.20

Trace complete.
```

รูปที่ 4.52 แสดงการทดสอบการ trace route จาก PC2 ไปยัง PC1

7) ทดสอบการ trace route จาก PC2 ไปยัง PC3

```
C:\Documents and Settings\CN>tracert 172.16.1.35

Tracing route to 172.16.1.35 over a maximum of 30 hops

  1    <1 ms    <1 ms    <1 ms  10.10.10.1
  2     1 ms     1 ms     1 ms  192.168.10.6
  3     2 ms     1 ms     1 ms  172.16.1.35

Trace complete.
```

รูปที่ 4.53 แสดงการทดสอบการ trace route จาก PC2 ไปยัง PC3

8) ทดสอบการ trace route จาก PC3 ไปยัง PC1

```
C:\Documents and Settings\CN>tracert 172.16.1.20

Tracing route to 172.16.1.20 over a maximum of 30 hops

  1    20 ms    <1 ms    <1 ms  172.16.1.33
  2     1 ms     1 ms    <1 ms  192.168.10.1
  3     1 ms    17 ms     1 ms  172.16.1.20

Trace complete.
```

รูปที่ 4.54 แสดงการทดสอบการ trace route จาก PC3 ไปยัง PC1

9) ทดสอบการ trace route จาก PC3 ไปยัง PC2

```
C:\Documents and Settings\CN>tracert 10.10.10.10

Tracing route to 10.10.10.10 over a maximum of 30 hops

  1     1 ms    <1 ms    <1 ms  172.16.1.33
  2     1 ms     1 ms     1 ms  192.168.10.1
  3     1 ms     1 ms     1 ms  192.168.10.9
  4    18 ms     1 ms     1 ms  10.10.10.10

Trace complete.
```

รูปที่ 4.55 แสดงการทดสอบการ trace route จาก PC3 ไปยัง PC1

สรุปผลการทดสอบ

หลังจากที่ได้ทดสอบเราเตอร์ XORP และระบบการจัดการฟังก์ชันเราเตอร์ XORP ผ่านเว็บเบสยูสเซอร์อินเตอร์เฟสแล้ว ได้ผลการทดลองการทำเราท์ติงต่างๆ ทั้งสามแบบคือ แบบ Static Routing, RIP Routing และ OSPF Routing ว่าอุปกรณ์เราเตอร์ที่สร้างขึ้นสามารถทำงานและใช้งานได้จริงและเว็บเบสยูสเซอร์อินเตอร์เฟสสามารถทำการสั่งงานได้โดยผู้ใช้ไม่ต้องรู้คำสั่งของโปรแกรมเราเตอร์ XORP และการเขียน configuration file

บทที่ 5
บทสรุปและแนวทางพัฒนาในอนาคต

การพัฒนาโครงงานในครั้งนี้อยู่ภายใต้สภาวะแวดล้อมแบบปิดคือได้ทดลองโปรแกรมทั้งหมดบนโปรแกรม Vmware และโปรแกรมที่ใช้คือระบบปฏิบัติการ FreeBSD , โปรแกรมจัดการฟังก์ชันเราเตอร์ XORP , โปรแกรมเว็บเซอร์เวอร์ lighttpd , โปรแกรมภาษา PHP และพัฒนาให้ทำงานได้จริงบน single board ของ soekris

5.1 ข้อจำกัดของระบบ

1. ตัวโปรแกรมจัดการฟังก์ชันเราเตอร์ XORP เป็นโปรแกรมเราเตอร์ที่ทำงานโดยอาศัย kernel ของ FreeBSD ในการส่งต่อ packet ดังนั้นเราเตอร์ที่ใช้ XORP ในการทำงานจริงๆ อาจจะมีประสิทธิภาพไม่สูงเทียบเท่ากับเราเตอร์ที่ได้รับการออกแบบ Hardware มาเพื่อการทำงานสำหรับเราท์ข้อมูลแต่สามารถทำงานได้เหมือนกันซึ่งต้องได้รับการทดสอบในระบบจริงต่อไป การพัฒนาโครงงานในครั้งนี้ได้ทำการทดสอบระบบภายใต้ระบบปิดคือภายในโปรแกรม Vmware อาจจะทำให้ไม่สามารถเห็นประสิทธิภาพในการทำงานได้อย่างชัดเจน

2. อุปกรณ์เราเตอร์ XORP ที่พัฒนาครั้งนี้ไม่ได้เพิ่มเติมคุณสมบัติอื่นๆ เช่น การทำหน้าที่แจก IP Address (DHCP Server) , การทำหน้าที่เป็น NAT (Network Address Translation) ดังนั้นเมื่อนำไปใช้งานจริงในเครือข่ายจะต้องจัดหาเครื่อง Server ที่ทำหน้าที่ดังกล่าวมาแล้วเพื่อเป็นการลดภาระงานของอุปกรณ์เราเตอร์ XORP

3. การพัฒนาครั้งนี้เลือกใช้โปรแกรม lighttpd มาทำเว็บเซอร์เวอร์ที่มีความสามารถใกล้เคียงกับโปรแกรมเว็บเซอร์เวอร์อื่นๆ แต่มีขนาดไม่ใหญ่เพื่อสะดวกในการทำเป็น MiniBSD นอกจากนี้ lighttpd ยังสามารถทำ Authentication ได้แต่การพัฒนาโครงงานนี้การทำ Authentication ยังไม่ได้ทำการเข้ารหัสของ username และ password ซึ่งอาจโดนดักจับได้จึงควรมีการพัฒนาเพิ่มเติมในส่วนนี้

5.2 สรุปแนวทางในการพัฒนาในอนาคต

การนำระบบเราเตอร์ XORP ที่พัฒนาไปใช้ในระบบเครือข่ายจริงๆ นั้นสามารถนำไปใช้แทนเราเตอร์ที่มีขายทั่วไปได้แต่ควรมีการนำไปทดสอบกับระบบเครือข่ายทั้งขนาดเล็ก ขนาดกลางและขนาดใหญ่ก่อนเพราะการทดสอบครั้งนี้ได้ทำในระบบปิดคือทำภายใต้โปรแกรม Vmware ตามแผนผังการทดสอบในบทที่ 4 และมีเครื่องลูกข่ายของแต่ละเครือข่ายเพียงแค่เครือข่ายละเครื่องเดียว

เท่านั้น แต่ระบบเครือข่ายที่ใช้งานจริงๆ จะต้องมีเครื่องลูกข่ายที่มีจำนวนมากกว่าและปริมาณ packet ที่ส่งต่อจะมีจำนวนมากกว่านี้

การ Login ผ่านหน้าเว็บยูสเซอร์อินเตอร์เฟสนั้นยังไม่ได้เข้ารหัสของ username และ password การนำไปพัฒนาต่อไปควรได้ทำการแก้ไขปรับปรุงในส่วนนี้เพื่อความปลอดภัยจากการดักจับข้อมูลของผู้ไม่หวังดี

การทำเว็บเบสยูสเซอร์อินเตอร์เฟสของการตั้งค่า configuration file ของเราติงแต่ละแบบการเราติงบางอย่างอาจจะยังไม่ได้ทำหน้า configuration เพื่อเข้าไปเพิ่มค่าการเราท์ติงให้กับเราเตอร์ XORP ดังนั้นการพัฒนาต่อไปจึงควรปรับปรุงโปรแกรมให้สามารถเพิ่มเติมการ configuration ได้มากกว่าเดิมแต่สำหรับการพัฒนาครั้งนี้นั้นเป็นการปรับแต่งค่า configuration ที่สามารถทำให้เราเตอร์ทำงานได้แล้ว

บรรณานุกรม

กิตติพงษ์ สุวรรณราช. 2537. **การบริหารจัดการเครือข่ายอินเตอร์เน็ต ด้วยระบบปฏิบัติการ FreeBSD** กรุงเทพฯ : ออฟเซ็ท เพรส.

จิตเกษม พัฒนาศิริ. 2541. **เสริมแต่งโฮมเพจให้มีชีวิตีวา ด้วย Java Script.** กรุงเทพฯ : บริษัท วิตตี้ กรุ๊ป จำกัด

ฐิติมา มโนหมั่นศรัทธา. 2537.**โคตรเซียน PHP.** กรุงเทพฯ : บริษัท โอเอวัน จำกัด .

นิรุธ อำนวยศิลป์. 2521. **PHP เพื่อการประยุกต์ใช้งาน.** กรุงเทพฯ : บริษัท ซัคเซส มีเดีย จำกัด.

บัณฑิต จารมรภูมิ. 2549. **คู่มือระบบยูนิกซ์ FreeBSD เล่ม 1.** กรุงเทพฯ : บริษัทบัณฑิตเพลส จำกัด.

สมศักดิ์ โชคชัยชุติกุล. 2537. **อินไซต์ PHP5.** กรุงเทพฯ : บริษัท โปรวิชั่น จำกัด.

เอกสิทธิ์ วิริยจารี. 2548. **"เรียนรู้ระบบเน็ตเวิร์ก จากอุปกรณ์ของ Cisco ภาคปฏิบัติ".** กรุงเทพฯ. บริษัท ซีเอ็ดยูเคชั่น,

Appendix A The FreeBSD System [Online]. Available URL : http://www.freebsd.org

Bassam Halabi, **Internet Routing Archtectures,** Cisco Press, Indianapolis, 1997.

International Computer Science Institute Berkeley,CA 94704,USA. XORP Design Overview Version 1.1. [Online]. Available URL : http://www.xorp.org

Lowell Jay Arther, **"Unix Shell programming"**, 2nd ed, John Wiley & Sons, Singapore, 1990.

Michael K. Glass, Yann Le Scouarnec, Elizabeth Naramore, Gary Mailer, Jeremy Stolz, Jason Gerner, **"Beginning PHP,Apache,MySQL Web Development"**, Wiley Publishing,Inc.,2004.

Mark Handly, Eddie Kohler, Atuna Ghosh, Orion Hodson, Pavlin Radoslavov. **Designing Extensible IP Router Software.**[Online]. Available URL: http://www.xorp.org

Mark Handly, Orion Hodson, Eddie Kohler. **XORP: An Open Platform for Network Research.** [Online]. Available URL : http://www.xorp.org

Mark Handley. 2003, November 30. **Proposal to Develop an Extensible Open Router Platform.** [Online]. Available URL : http://www.xorp.org

XORP Project. 2005, April 13. **XORP User Manual Version 1.1** [Online]. Available URL : http://www.xorp.org

XORP Project.2006, August 2. **"XORP User Manual version1.3"** [online]. Available URL: http://www.xorp.org

XORP Project.2007, March 20. **"XORP User Manual version1.4"** [online]. Available URL: http://www.xorp.org

ภาคผนวก ก

การติดตั้งโปรแกรมที่เกี่ยวข้อง

ขั้นตอนการติดตั้งโปรแกรมต่างๆ ที่เกี่ยวข้องกับระบบเราเตอร์ XORP และเว็บเบสสำหรับติดต่อกับผู้ใช้ การพัฒนาระบบนี้ใช้โปรแกรมทั้งหมดที่ download มาได้โดยไม่เสียค่าใช้จ่ายทั้งสิ้นและมีขั้นตอนการติดตั้งแต่ละโปรแกรมดังนี้

1. การติดตั้งระบบปฏิบัติการ FreeBSD 6.2 เป็นระบบปฏิบัติการของระบบ

1) Boot เครื่องด้วยแผ่น FreeBSD 6.2

2) เลือกประเทศไทย รหัส 212

3) เลือก Keymap

4) เมื่อเข้าเมนู Sysinstall ให้เลือกการติดตั้งแบบ Custom

5) เมื่อเข้าเมนู Option ไม่ต้องเปลี่ยนค่าใดๆ

6) การกำหนด partition ให้เลือก A คือใช้ทั้งหมด

7) เมื่อเข้าสู่การ Install Boot Manager ให้เลือกแบบ Standard

8) กำหนด Label ที่ต้องการสร้างและขนาด เช่น / = 120 MB , swap = 128 MB , /usr = 700 MB และ /data = เนื้อที่ส่วนที่เหลือทั้งหมด

9) ส่วนของ Distribution ให้เลือกการติดตั้งแบบ minimal

10) เลือก Media คือ CD/DVD

11) Commit รอการติดตั้งเสร็จแล้ว Boot เครื่องหนึ่งครั้ง

2. การติดตั้ง Lighttpd Web server

1) Download โปรแกรม lighttpd-1.4.15.tar.gz จากเว็บไซต์ http://www.lighttpd.net จากนั้นทำการ unzip

2) ติดตั้งด้วยคำสั่ง

```
#gzip –cd lighttpd-1.4.15.tar.gz | tar xvf –
#cd lighttpd-1.4.15
#./configure
#make
#make install
```

3) ปรับแต่งไฟล์ configuration ของ lighttpd

3. การติดตั้ง PHP

1) Download โปรแกรม php-4.4.7.tar.gz จากเว็บไซต์ http://www.php.net จากนั้นทำการ unzip

2) ติดตั้งด้วยคำสั่ง

#gzip –cd php-4.4.7.tar.gz | tar xvf –

#cd php-4.4.7

#./configure

#make

#make install

3) ปรับแต่งไฟล์ php.ini แล้วบันทึกไว้ที่ /usr/local/lib/php.ini

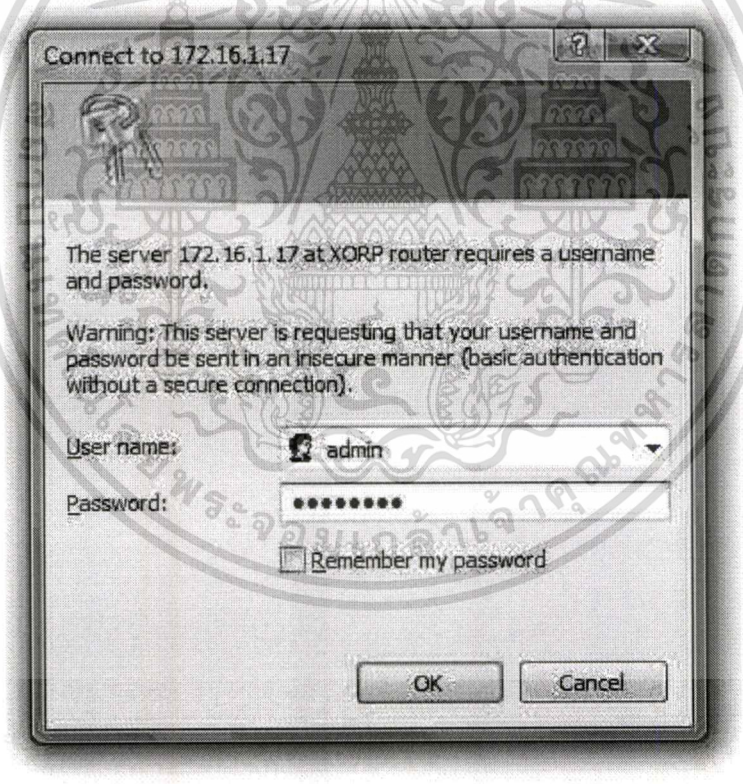
ภาคผนวก ข

คู่มือการใช้งานเว็บเบสยูสเซอร์อินเตอร์เฟส สำหรับอุปกรณ์เราเตอร์ XORP

1.เริ่มต้นเข้าใช้งาน

การเริ่มเข้าใช้งานเริ่มจากผู้ใช้พิมพ์ http://ip_address:portnumber/frm_general.php ซึ่งเป็นหน้าแรกของเว็บเบสหลังจากนั้นผู้ใช้จะได้พบการตรวจสอบผู้ใช้ของเว็บเซอร์เวอร์ lighttpd ดังนี้

http://172.16.1.17:81/frm_general.php



รูปที่ ข.1 แสดงหน้าจอการ Login

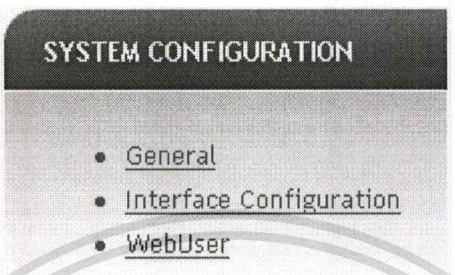
ให้กรอก User name : admin

Password : password

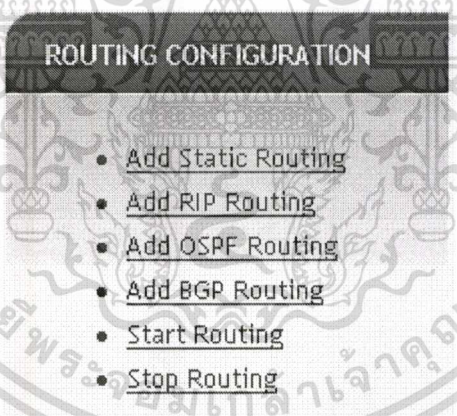
2. เมนูการใช้งาน

ระบบประกอบด้วยเมนูหลักๆ ดังนี้คือ

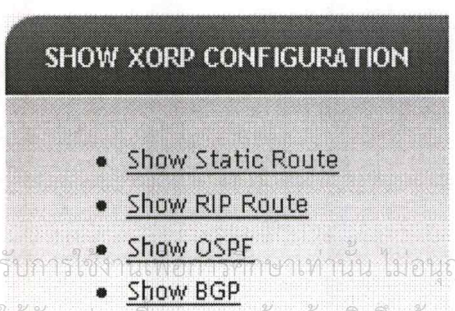
2.1) เมนู System Configuration เป็นเมนูสำหรับการตั้งค่าพื้นฐานของระบบคือค่าของแต่ละ Interface การตั้งค่าของ Web server



2.2) เมนู Routing Configuration เป็นเมนูสำหรับการเพิ่มค่า Routing Configuration แบบต่างๆ คือ Static Routing, RIP Routing, OSPF Routing , BGP Configuration และใช้ในการ Stop และ Start Router อีกด้วย



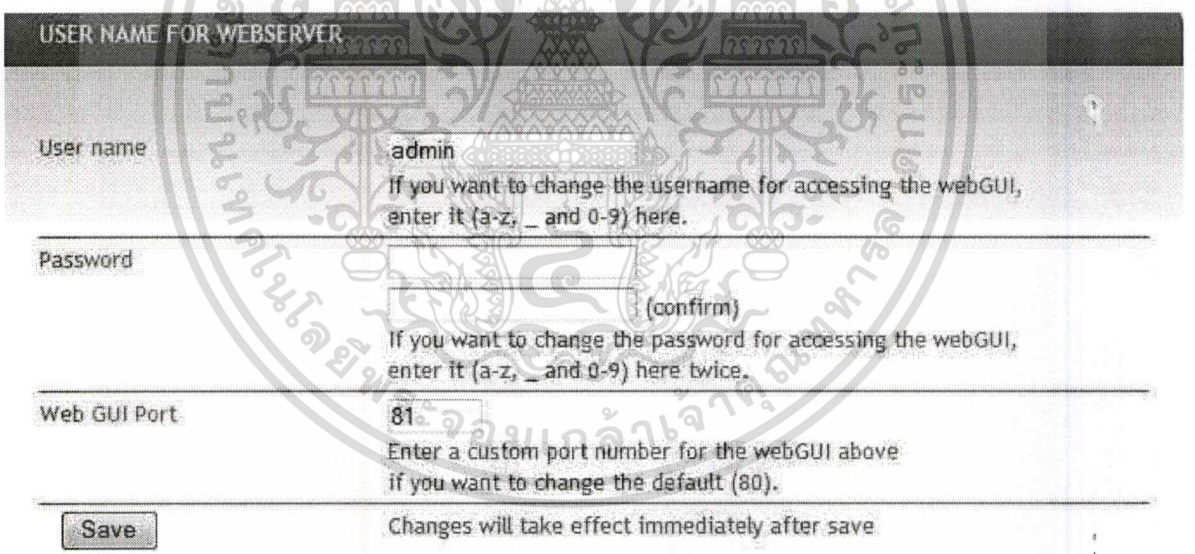
2.3) เมนู Show XORP Configuration เป็นเมนูสำหรับการแสดงค่าของ configuration file ออกมาในรูปของตารางที่ดูง่าย ประกอบด้วยเราท์ติงโปรโตคอลต่างๆ คือ Static Routing, RIP Routing, OSPF Routing และ BGP Routing



2.4) เมนู Tools เป็นเมนูสำหรับรวบรวมคำสั่งเพื่ออำนวยความสะดวกแก่การใช้งานและแสดงสถานะของ Routing Table แบบต่างๆ , เมนูสำหรับการแก้ไขไฟล์ configuration และการ Upload file เข้าสู่ระบบ

3. เปลี่ยนชื่อผู้ใช้ (User name) รหัสผ่าน (password) และ หมายเลข port

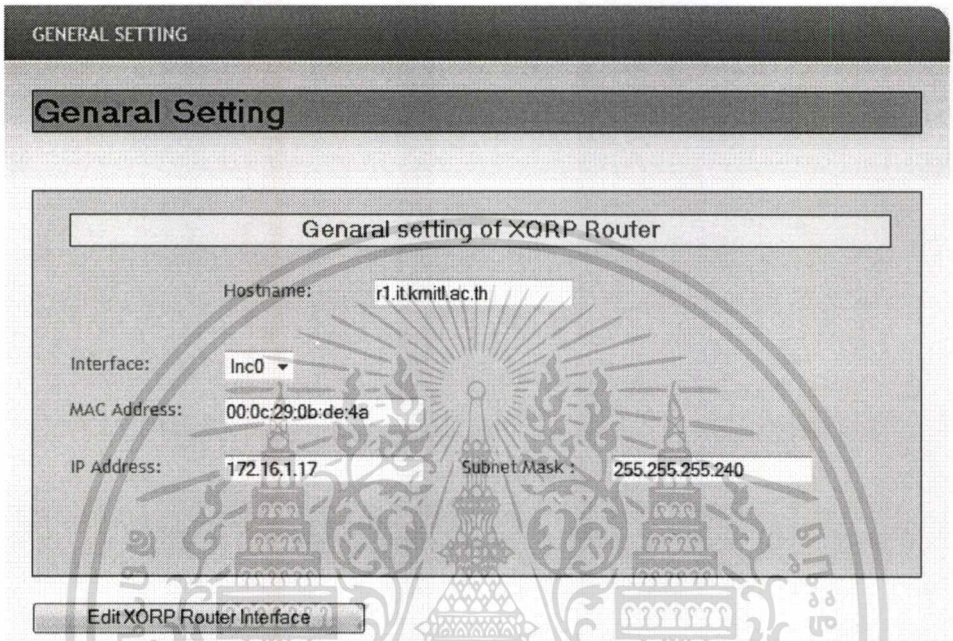
ผู้ใช้สามารถเปลี่ยนค่าของ User name , password และ port number ได้ดังนี้



รูปที่ ข.2 แสดงหน้าจอการเปลี่ยน User name , password และ Port number

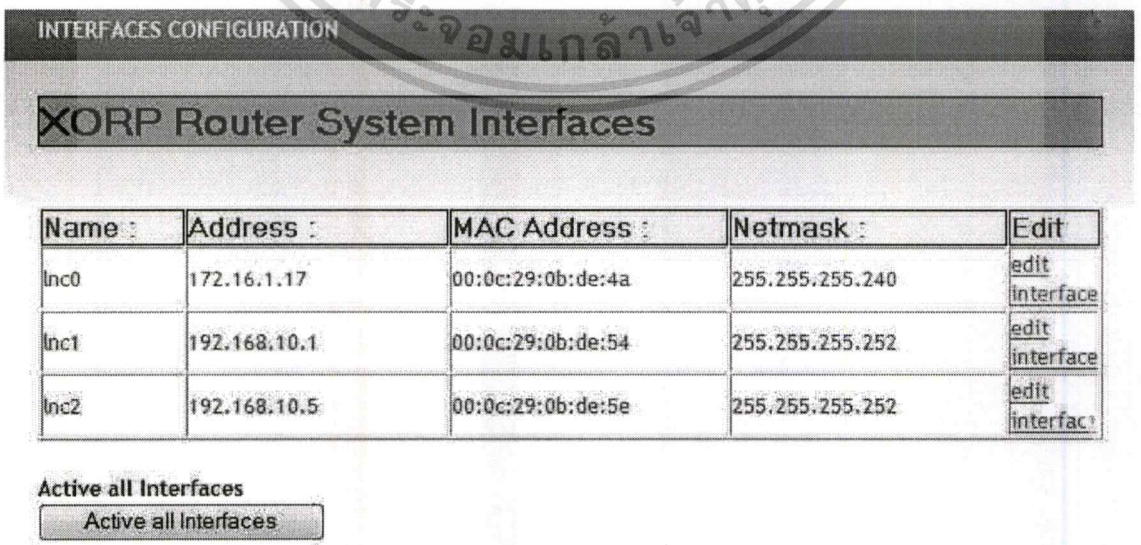
4. การตั้งค่าหมายเลข IP Address ของแต่ละ interfaces

การตั้งค่าทำได้โดยเมื่อเข้าสู่ระบบแล้วผู้ใช้จะพบหน้าจอแสดงค่าที่มีอยู่เดิมของระบบในแต่ละ interfaces ผู้ใช้ต้องเปลี่ยนหมายเลข IP Address และ Subnet Mask ตามเครือข่ายที่ต้องการ โดยผู้ใช้คลิกเลือกที่ปุ่ม Edit XORP Router Interfaces



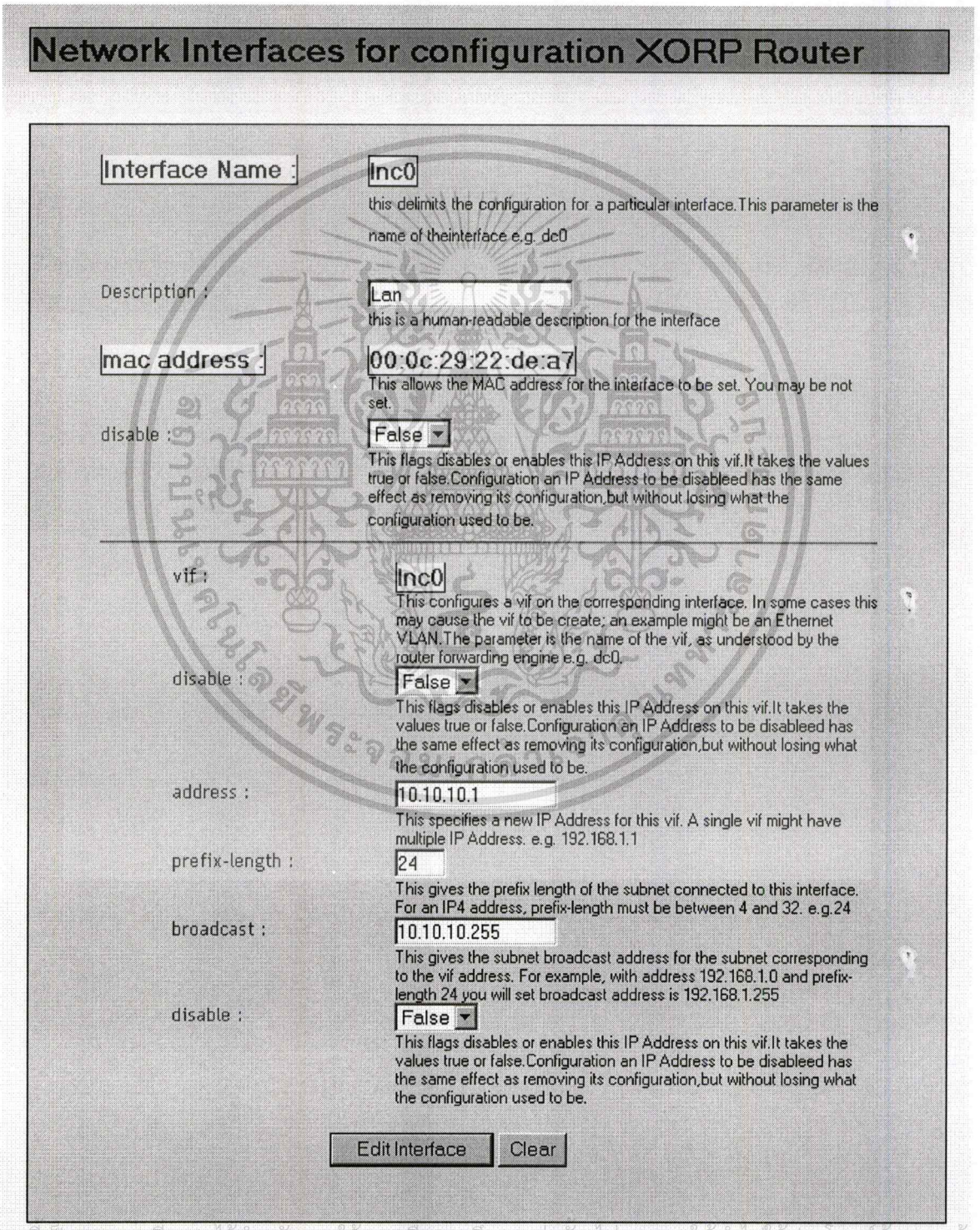
รูปที่ ข.3 แสดงหน้าจอแสดงค่าของ Interface

เมื่อเลือกที่ปุ่ม Edit XORP Router Interface แล้วระบบจะโชว์ภาพรวมของ Interface อีกครั้งดังภาพให้คลิกที่ Edit interface ตามแต่ละ Interface ที่ต้องการเปลี่ยนหมายเลข IP Address และค่าที่ต้องการ



รูปที่ ข.4 แสดงภาพรวมของทุก Interface ที่มีอยู่ในระบบ

ระบบจะให้ผู้ใช้เข้าไปแก้ไขในแต่ละ Interface ตามที่ต้องการแก้ไขดังภาพที่ ข.4 แสดง ซึ่งค่าที่ป้อนเข้าไปในหน้าจอนี้จะถูกบันทึกในไฟล์ configuration ของ XORP และไฟล์ configuration ของ FreeBSD คือไฟล์ /etc/rc.conf หลังจากได้แก้ไขทั้งหมดแล้วและต้องการให้แต่ละ Interface ทำงานได้ทันทีให้คลิกที่ปุ่ม Active all Interfaces ในหน้า XORP Router System Interfaces ดังภาพที่ ข.3

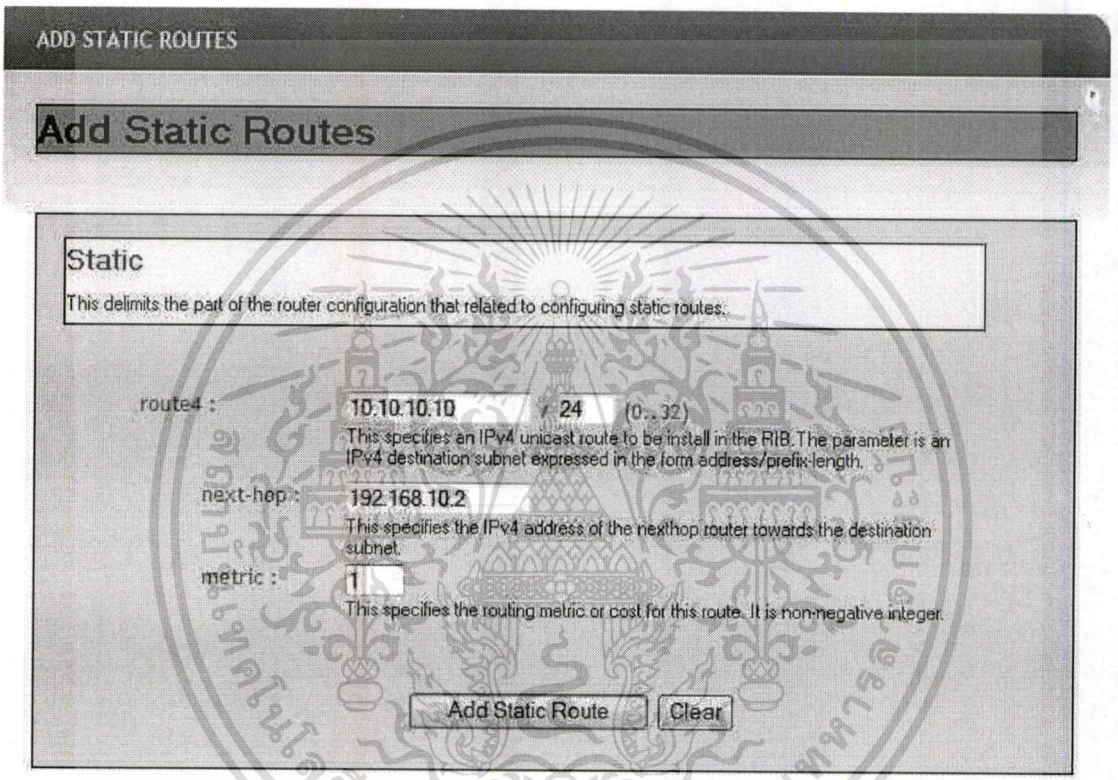


รูปที่ ข.5 แสดงหน้าจอป้อนค่าของ Interface แยกตาม Interface

5. การแก้ไขและเพิ่มค่า Configuration file ของการเราท์ติงแบบต่างๆ

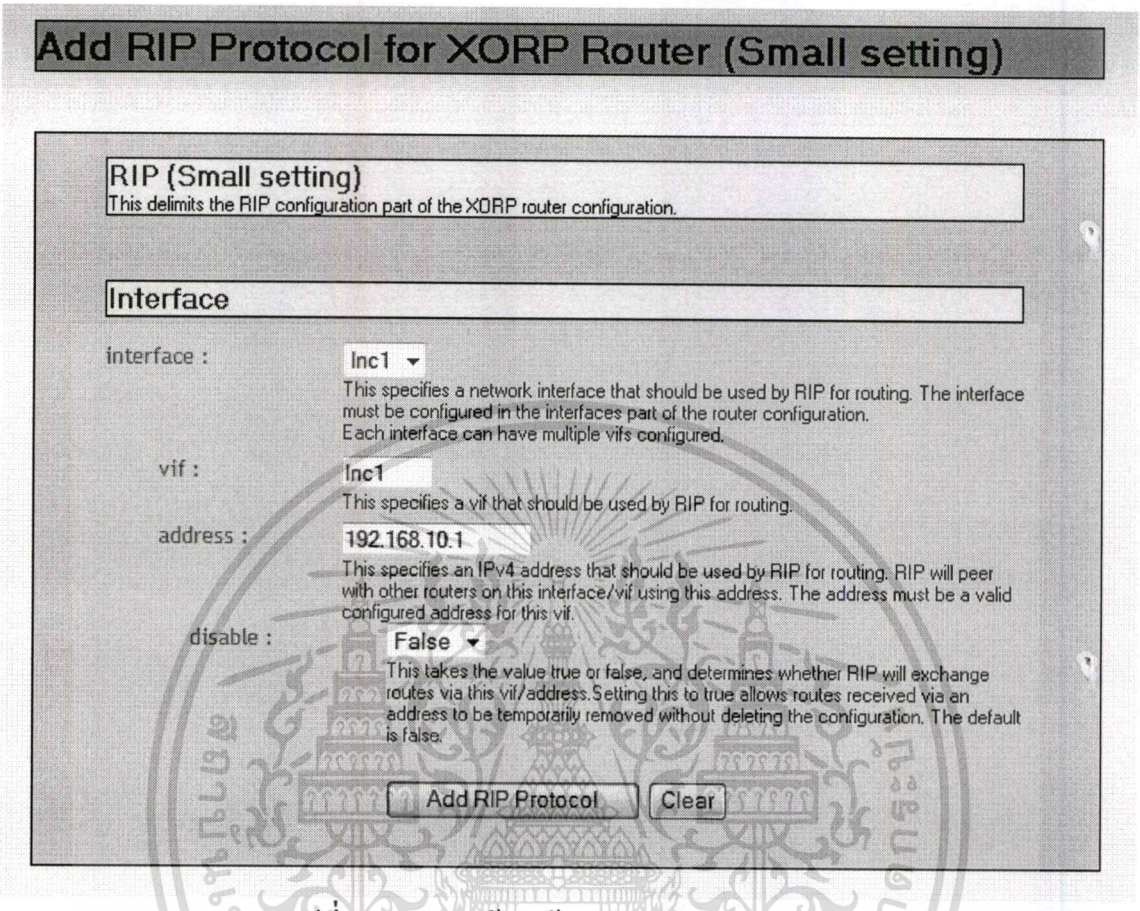
หลังจากที่ผู้ใช้ได้แก้ไขค่าของแต่ละ Interface ที่ตัวเองต้องการแล้วนั้นต่อไปก็คือผู้ใช้ต้องเลือกว่าจะใช้การ เราท์ติงแบบไหนและเข้าไปแก้ไขค่าการเราท์ติงแบบต่างดังที่ได้อธิบายไปแล้วในรายละเอียดของการทำงาน

5.1) การเพิ่มค่าของ Static Routing Protocol



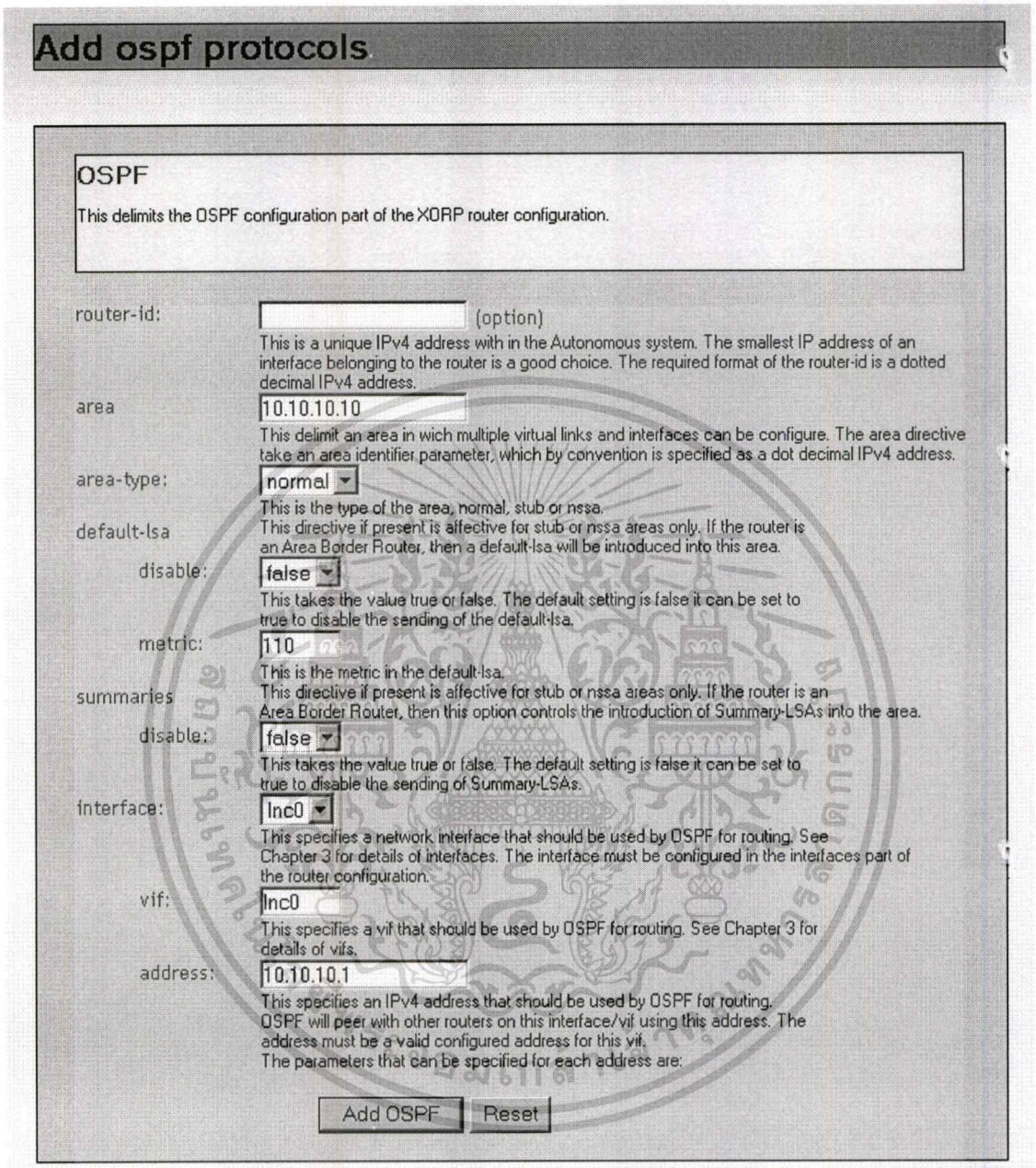
รูปที่ ข.6 แสดงหน้าจอป้อนค่าของ Static Route

5.2) การเพิ่มค่าของ RIP Routing Protocol



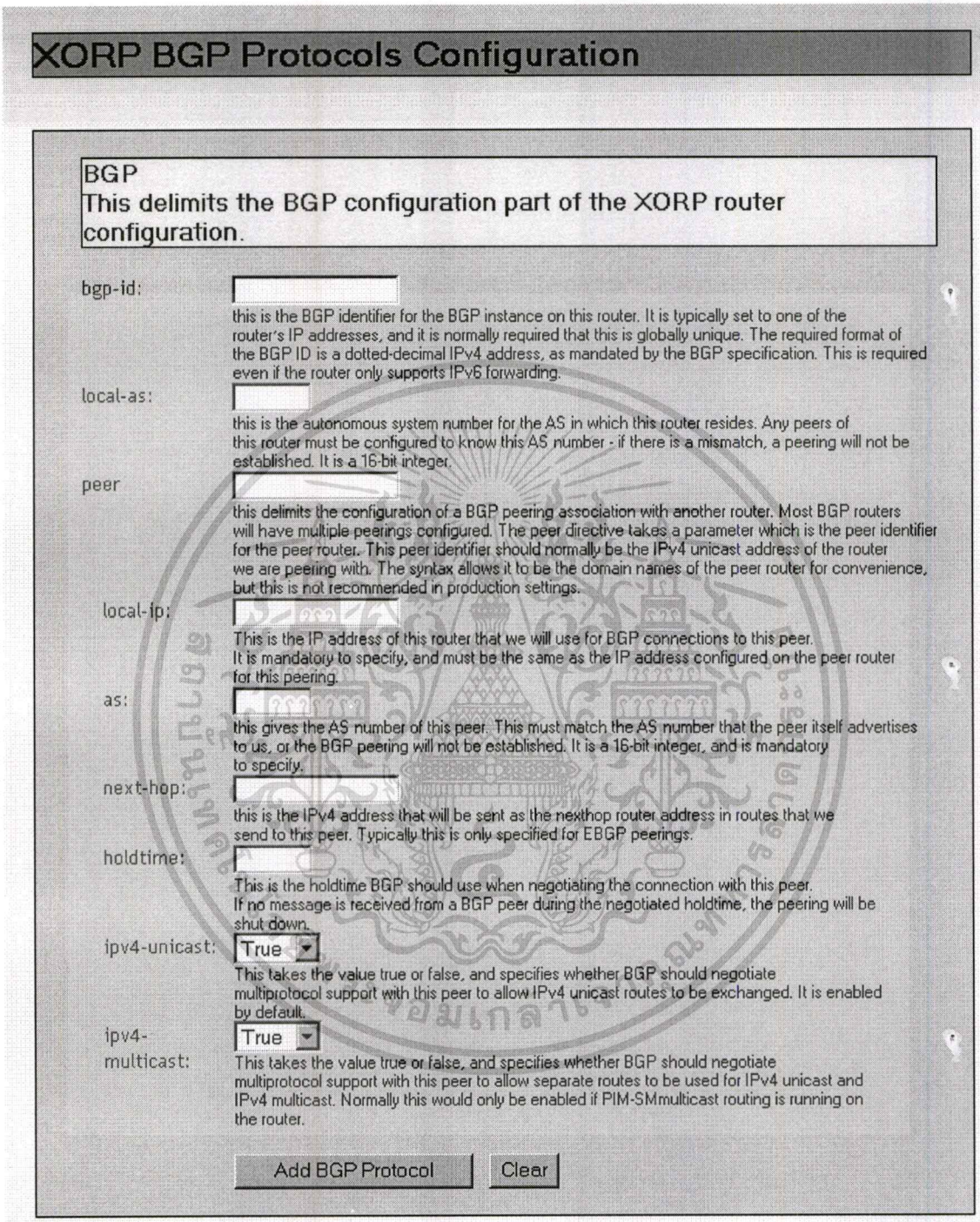
รูปที่ ข.7 แสดงหน้าจอป้อนค่าของ RIP Routing

5.3) การเพิ่มค่าของ OSPF Routing Protocol



รูปที่ ข.8 แสดงหน้าจอป้อนค่าของ OSPF Routing

5.4) การเพิ่มค่าของ BGP Routing Protocol

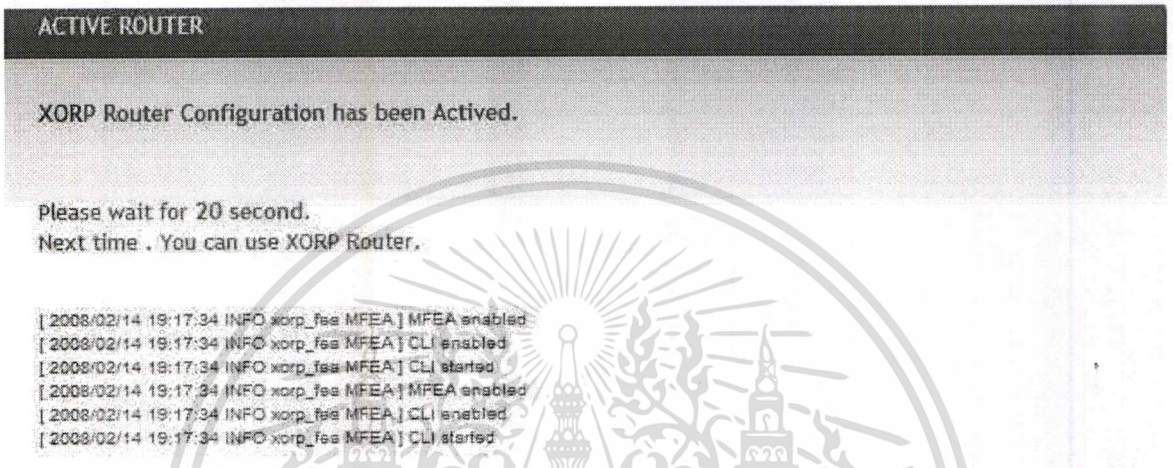


รูปที่ ข.9 แสดงหน้าจอป้อนค่าของ BGP Routing

6. การสั่งให้เราเตอร์ทำงานและหยุดการทำงาน

เมื่อผู้ใช้ได้เลือกใช้โปรโตคอลการเราท์ติงแบบใดแล้วและได้ตั้งค่าของเราเตอร์ทุกตัวแล้ว ผู้ใช้ต้องสั่งงานให้ตัวเราเตอร์ XORP เริ่มทำงานทันทีโดยที่ไม่ต้อง Boot เครื่องใหม่โดยกานเลือกที่เมนู Start Routing และถ้าต้องการให้เราเตอร์ XORP หยุดการเราท์ติง สามารถสั่งงานที่เมนู Stop Routing ดังภาพ

6.1) การสั่งงานให้เราเตอร์ XORP ทำงาน



รูปที่ ข.10 แสดงหน้าจอเมื่อ XORP เริ่มทำงาน

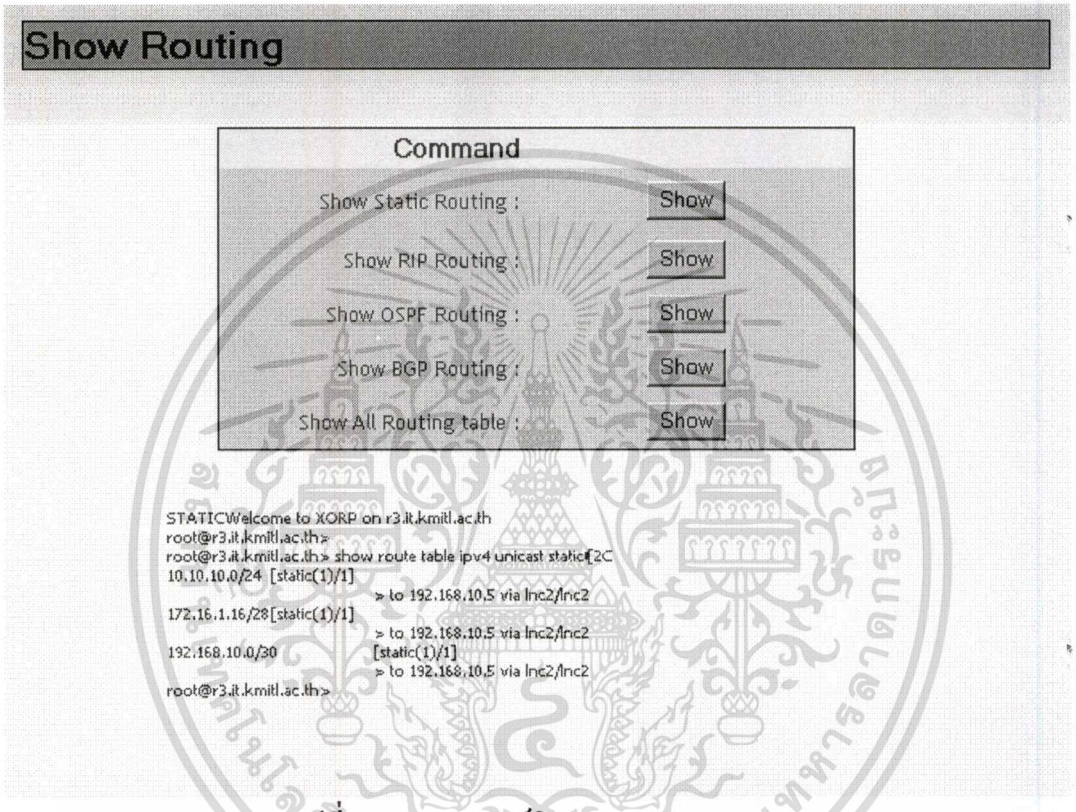
6.2) การสั่งงานให้เราเตอร์ XORP หยุดการทำงาน

รูปที่ ข.11 แสดงหน้าจอเมื่อ XORP หยุดการทำงาน

7. การแสดงสถานะการทำงานของการเราท์ติงแต่ละแบบ

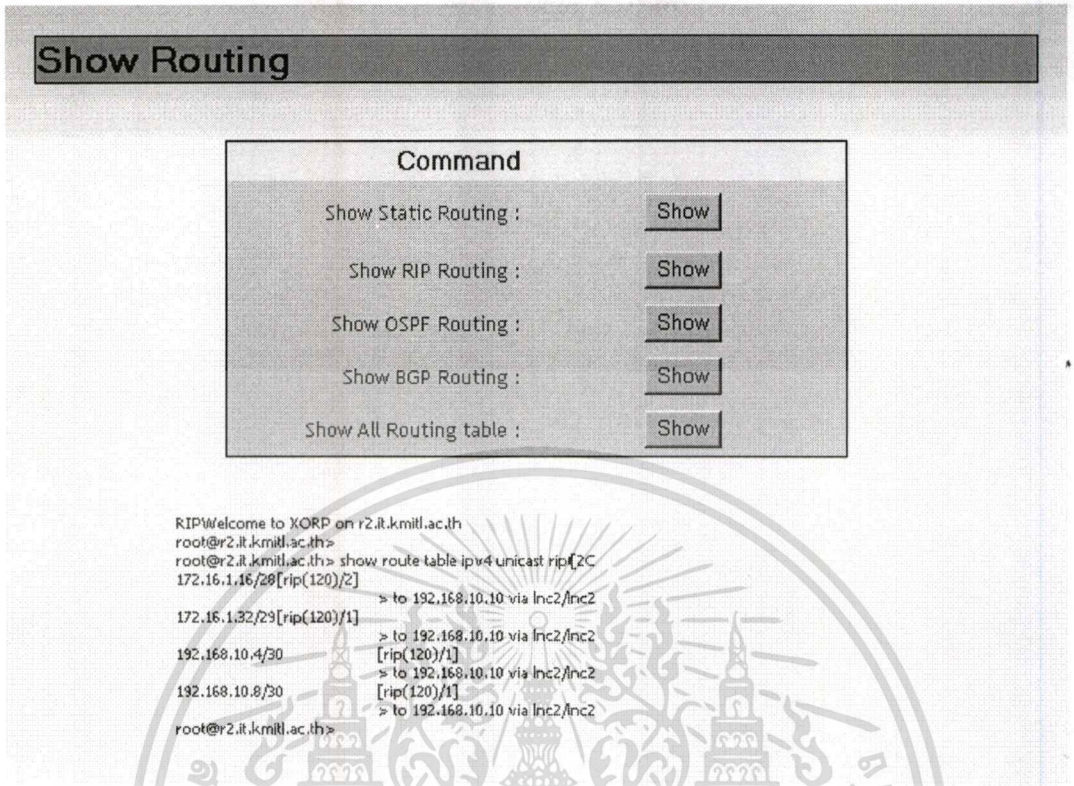
สำหรับเว็บเบสยูสเซอร์อินเตอร์เฟส ได้มีระบบแสดงสถานะการเราท์ติงแต่ละแบบที่สามารถดูค่าของเราท์ติงเทเบิลของเราเตอร์ XORP ขึ้นมาผ่านหน้าเว็บได้ ในการพัฒนาระบบครั้งนี้ผู้พัฒนาได้เตรียมแสดงค่าเราท์ติงเทเบิลต่างๆ ไว้ดังนี้คือ

7.1) แสดงการเราท์ติงแบบ Static Routing Protocol



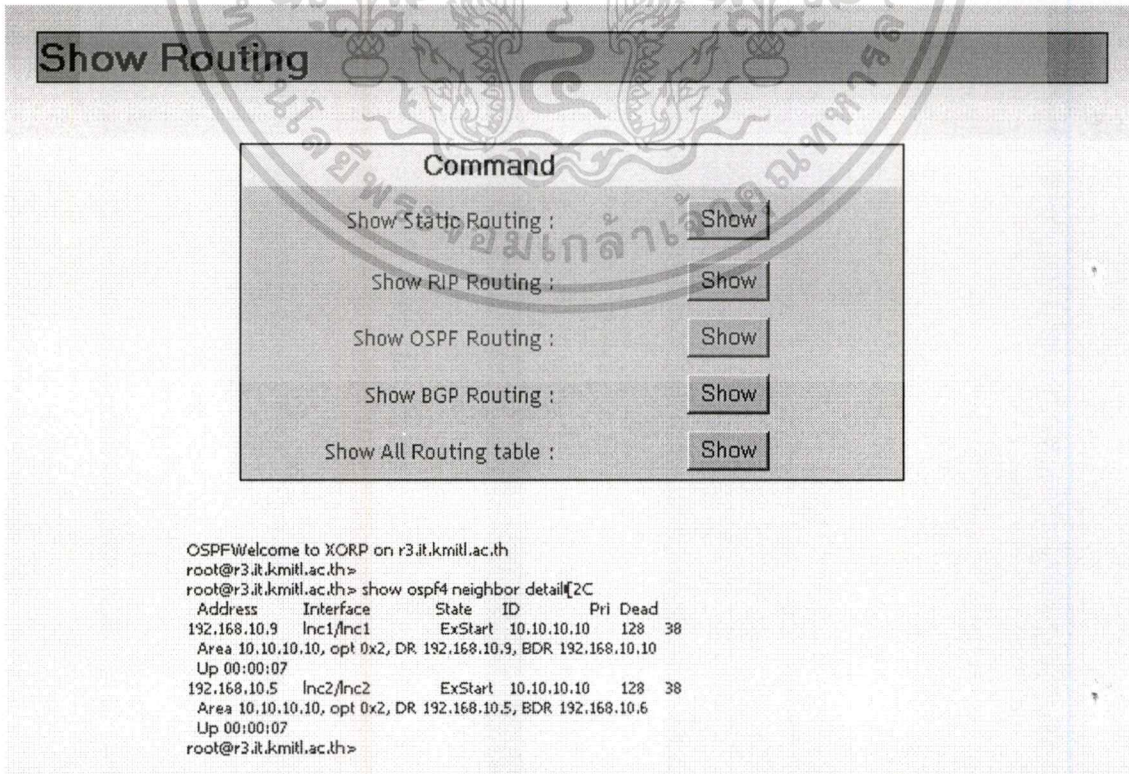
รูปที่ ข.12 แสดงเราท์ติงแบบ Static Routing

7.2) แสดงการเราท์ติงแบบ RIP Routing Protocol



รูปที่ ข.13 แสดงเราท์ติงแบบ RIP Routing

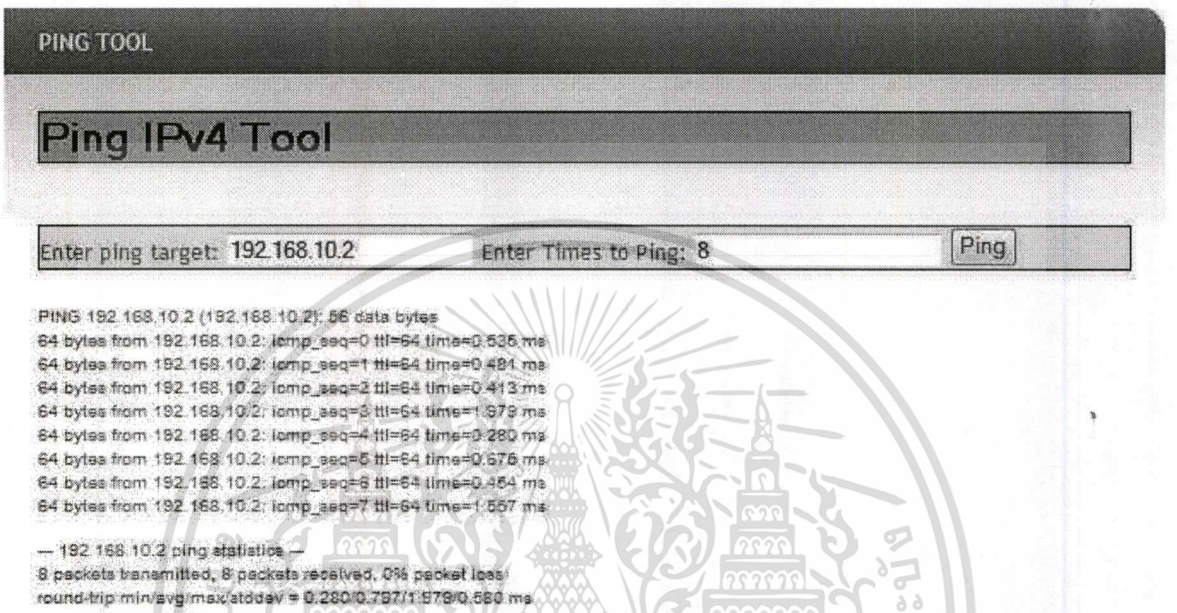
7.2) แสดงการเราท์ติงแบบ OSPF Routing Protocol



รูปที่ ข.14 แสดงเราท์ติงแบบ OSPF Routing

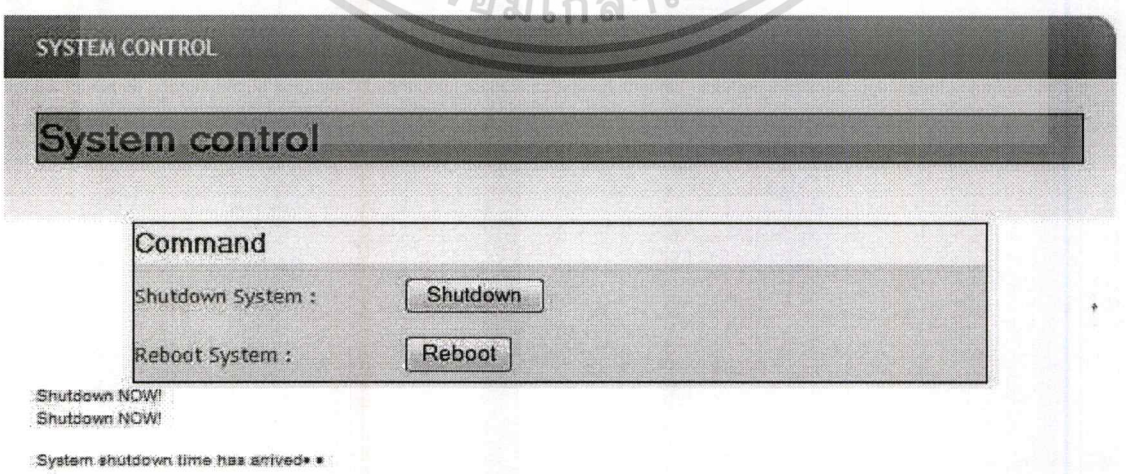
8. การทำงานแบบอื่นๆ ที่ได้เตรียมไว้คือ การทดสอบการ Ping , การ Shutdown และ Restart ระบบ , การแก้ไข Configuration file ของ XORP และการ Upload file

8.1) การทดสอบการ Ping เป็นการทดสอบการ ping จากเครื่องเราเตอร์ XORP ไปยังเครื่องอื่นๆ ในเครือข่ายผ่านหน้าเว็บเบสดังแสดงในรูป



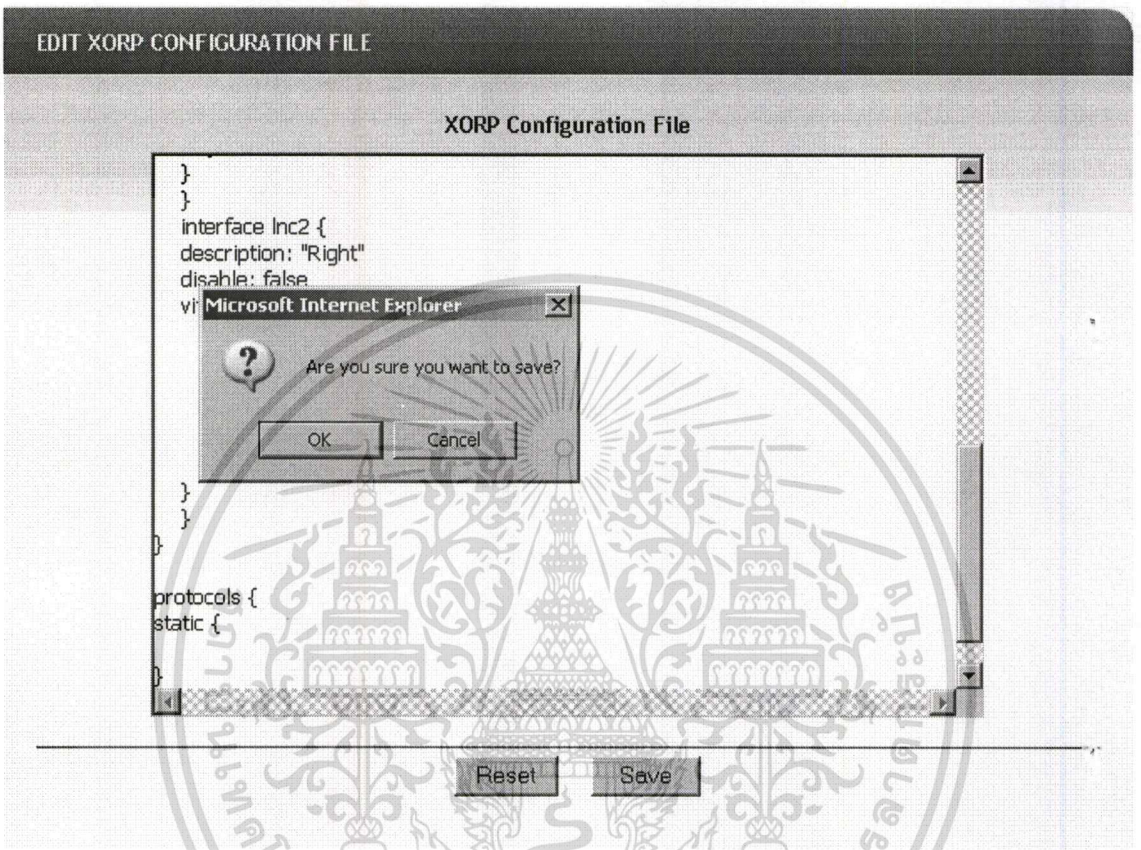
รูปที่ ข.15 แสดงการทดสอบการ Ping

8.2) การ Shutdown และ Restart ระบบ เป็นระบบที่เตรียมไว้ให้ผู้ใช้ได้สั่งงานระบบให้สามารถสั่งให้เราเตอร์ XORP บูตระบบใหม่หรือ Shutdown ได้ตามที่ต้องการ ซึ่งการใช้งานจริงๆ อาจจะไม่ต้องใช้เพราะสามารถปิดเครื่องได้เลยโดยไม่ต้อง Shutdown



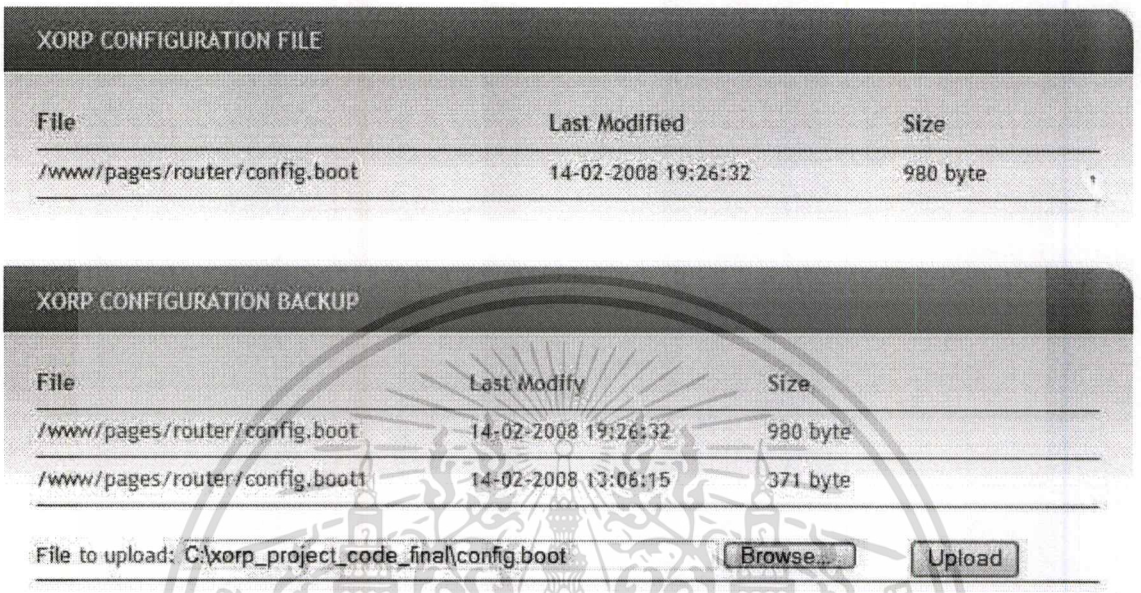
รูปที่ ข.16 แสดงการสั่งงาน Shutdown ระบบ

8.3) การแก้ไข XORP Configuration file บางครั้งการเพิ่มค่า แก้ไขค่าของ XORP configuration file อาจจะไม่สมบูรณ์ หรือเพื่อความถูกต้องผู้ใช้อาจจะใช้หน้าเว็บเบสนี้สำหรับการดู configuration file หรือแก้ไขได้ ดังรูป



รูปที่ ข.17 แสดงการแก้ไข XORP Configuration file และบันทึก

8.4) การ Upload file เมื่อผู้ใช้ได้เตรียมไฟล์ XORP Configuration file เอาไว้เรียบร้อยแล้วสามารถที่จะ Upload ไฟล์ที่ได้เตรียมเอาไว้ไปแทนที่ไฟล์เดิมและสั่งให้เราเตอร์ทำงานได้เลย ซึ่งเป็นความสะดวกถ้าต้องจัดการในปริมาณที่มากๆ ดังรูป



รูปที่ ข.18 แสดงการ Upload XORP Configuration file

ภาคผนวก ค

การทำระบบ MiniBSD

ระบบเราเตอร์ XORP ที่ได้ทำในโครงงานพัฒนาระบบในครั้งนี้นั้นต้องเตรียมให้โปรแกรมระบบปฏิบัติการ FreeBSD , โปรแกรมจัดการฟังก์ชันเราเตอร์ XORP และเว็บเบสยูสเซอร์อินเตอร์เฟส ทั้งหมดนั้นบันทึกสู่หน่วยความจำประเภท Compact Flash เพื่อสามารถนำมาบูตระบบได้บนอุปกรณ์ Single Board ของ Soekris รุ่น net5501 ซึ่งต้องทำตามแบบของ MiniBSD ตามขั้นตอนดังต่อไปนี้

ก่อนการทำตามขั้นตอนต่อไปนี้ต้องติดตั้งระบบปฏิบัติการ FreeBSD 6.2 ตามปกติให้เรียบร้อยเสีย ก่อนจึงจะทำตามขั้นตอนต่อไปนี้ได้

1. ติดตั้ง FreeBSD jail

- ให้ผู้ใช้ Logon เป็น root เข้าสู่ FreeBSD Host Machine และ เรียกโปรแกรม Sysinstall โดยการพิมพ์

#sysinstall

- จากเมนู ของ Sysinstall เลือก

- Custom

- Options

เมื่อพบหน้าจอดำๆที่ประกอบด้วยข้อความมากมายให้เลื่อนลูกศรมาที่ Install Root เพื่อเป็นการเปลี่ยนตำแหน่ง root ที่ต้องการติดตั้งใหม่เป็น

/usr/jail

- หลังจากเลือกได้แล้วให้เคาะ Enter และ Q เพื่อออกจากหน้าจอนี้
- ยกเว้น Over Partition และ Label และให้ไปเลือก Distributions
- ที่หน้าจอ Distributions ให้เลื่อนลูกศรมาที่ Minimal Options และเลือก Minimal distribution package โดยการเคาะ space bar และเลือก custom option เลือก src ให้เลือกเพิ่มคือ

  - lib

  - sys

- ออกไปสู่เมนูหลักแล้วเลือก สื่อที่เป็นแหล่งของไฟล์คือ CD/DVD แล้ว Commit

2. ปรับแต่ง FreeBSD jail

เมื่อได้ทำการติดตั้ง FreeBSD อีกครั้งไว้ใน jail เป็นที่เรียบร้อยแล้วให้เริ่มทำการปรับแต่ง โดยการเข้าไปแก้ไข การปรับแต่งนี้ได้ใช้ script ของ ultradesic สามารถ download ได้จากเว็บไซต์ http://www.ultradesic.com โดยสั่งดังนี้

```
cd /root
fetch http://www.ultradesic.com/pub/miniBSD_Files/minichroot.sh
```

บันทึกไฟล์แล้วเปลี่ยนสิทธิ์ของไฟล์

```
chmod 700 minichroot.sh
```

ก่อนเริ่มการปรับแต่งให้เริ่ม copy ไฟล์ /etc/resolv.conf และ /etc/localtime จาก host machine ไปยัง /usr/jail/etc โดยการพิมพ์

```
cp /etc/resolve.conf /usr/jail/etc
cp /etc/localtime /usr/jail/etc
```

ถ้าเครื่องคอมพิวเตอร์ของเราเวลาอยู่ใน UTC เราสามารถที่จะยกเว้นการกระทำที่ต่อไปได้ แต่ถ้าเครื่องคอมพิวเตอร์ของเราเวลาอยู่ใน local time ต้องแน่ใจว่าได้ copy /etc/wall_cmos_clock โดยการพิมพ์

```
cp /etc/wall_cmos_clock /usr/jail/etc
```

ต่อไปคือเข้าไปแก้ไข /usr/jail/root/.cshrc ไฟล์และเพิ่มคำสั่งต่อไปนี้ในบรรทัดสุดท้าย

```
set prompt = "MiniBSD %~ %# "
```

ที่จะทำหน้าที่ set เครื่องหมาย prompt เมื่อเปลี่ยน root เข้าสู่ FreeBSD jail และเมื่อเสร็จเรียบร้อยแล้วให้เข้าสู่ jail โดยการใช้ script

```
/root/minichroot.sh
```

ถ้าคุณไม่ใช้ script พิมพ์คำสั่งต่อไปนี้

```
touch /root_check
mount -t devfs devfs /usr/jail/dev
chroot /usr/jail /bin/csh
```

3. Safety Checks

ต่อไปเป็นการเริ่มต้นทำระบบ MiniBSD แต่ก่อนจะเริ่มทำต้องแน่ใจก่อนว่าได้อยู่ใน jail เสียก่อนและ dev ได้ mount เรียบร้อยแล้วใน FreeBSD jail ให้คุณพิมพ์

```
ls /root_check
```

ถ้าคุณอยู่ใน jail คุณจะต้องไม่เห็นไฟล์นี้ถ้าคุณเห็นไฟล์นี้แสดงว่ามีบางอย่างผิดพลาดให้หยุดแล้วเริ่มทำใหม่

เมื่อคุณอยู่ใน jail เรียบร้อยแล้ว ให้ตรวจสอบเพื่อความแน่ใจว่า device nodes มีอยู่แล้ว โดยพิมพ์ดังนี้

```
ls /dev
```

ถ้าคุณไม่เห็นไฟล์แสดงว่ามีบางอย่างผิดพลาดให้หยุดแล้วเริ่มต้นใหม่

4. Building miniBSD 6.x

ส่วนต่อไปนี้เป็นการสร้าง miniBSD ในการทำงานครั้งนี้ได้ทำกับ FreeBSD 6.2 ถ้าคุณรู้สึกว่าการทำงานบางอย่างไม่ค่อยเป็นตามที่คาดคุณสามารถออกจาก jail ได้โดยการ unmount /usr/jail/dev และ ลบไดเรคทอรี่ /usr/jail ได้

5. Create Directory Structure

สิ่งแรกที่คุณคสรทำคือการสร้างโครงสร้างของ ไดเรคทอรี่ ของระบบ miniBSD คุณสามารถใช้คำสั่ง mkdir สร้างเองหรือใช้ script โดยทำดังนี้

```
cd /root
fetch http://www.ultradesic.com/pub/miniBSD_Files/create-minibsd-dirs.sh
chmod 0700 /root/create-minibsd-dirs.sh
/root/create-minibsd-dirs.sh
```

ถ้าคุณต้องการสร้างเองสามารถใช้โครงสร้างดังนี้

- usr/minibsd
  - bin
  - boot
    - defaults
  - dev
  - etc
    - defaults
    - mtree
  - lib
  - libexec
  - mnt
  - proc
  - root
  - sbin
  - usr
    - bin
    - lib
      - aout
    - libexec
    - local
    - sbin
    - share
      - misc
  - var

คุณต้องแน่ใจว่าได้ ตั้งค่า permissions ของ ไดเรคทอรี proc เป็น 555 และ permission ของ root ไดเรคทอรีเป็น 700 แล้วสร้าง symlink จาก /var/tmp ไป /tmp

6. Rebuilding the boot loader

ไบออสของเครื่อง Soekris มี bug อยู่ที่ serial console emulation ถ้าคุณใช้ FreeBSD boot loader แบบมาตรฐานอาจจะมีตัวอักษรแปลกๆออกมา เช่น (- \ | / - ...) แก้ไขได้โดยแก้ที่ไฟล์ต่อไปนี้

Edit /sys/boot/i386/libi386/Makefile and comment out the line that reads:

CFLAGS+= -DTERM_EMU

บันทึกลงไฟล์

พิมพ์คำสั่งต่อไปนี้

mkdir -p /usr/share/man/man{1,2,3,4,5,6,7,8}

ขณะนี้คุณต้อง compile และ install bootloader โดยพิมพ์

cd /sys/boot

make clean

make

make install

7. Building Dynamic Executables

ขั้นตอนต่อไปนี้อาจจะยกเว้นได้แต่ถ้าต้องการประหยัดเนื้อที่ของ miniBSD ให้ทำดังนี้ แก้ไขที่ไฟล์ /etc/make.conf โดยการเพิ่มบรรทัดนี้ลงไป

NO_SHARED=no

ตารางต่อไปนี้เป็นเนื้อที่ของแต่ละ partition ที่คุณสามารถประหยัดได้ถ้าทุกๆไบต์มีคุณค่าสำหรับคุณ

Default Size	Size after compiling with NO_SHARED=no
/bin = 802K	/bin = 802K
/sbin = 1.2M	/sbin = 766K
/usr/bin = 3.3M	/usr/bin = 3.3M
/usr/sbin = 1.0M	/usr/sbin = 1.0M

ก่อนที่คุณจะ rebuild ไฟล์ binary คุณต้องรัน sysibstall จาก jail ของคุณดังนี้

/usr/sbin/sysinstall

เลือกจากเมนู

- Configure
  - Distributions

คุณต้องเลือก src option แล้วเลือกต่อไปนี้

- contrib
- libexec
- release
- bin
- sbin
- ubin
- usbin

ออกมาจาก sysinstall เลือก Media source แล้ว commit รอให้ติดตั้ง

ให้ download script จากเว็บไซต์ดังนี้

```
cd /root
fetch http://www.ultradesic.com/pub/miniBSD_Files/build-dynamic-libs-6.sh
chmod 0700 /root/build-dynamic-libs-6.sh
/root/build-dynamic-libs-6.sh
```

ถ้ารัน script แล้วไม่ เกิดข้อผิดพลาดใดๆ ก็ทำขั้นตอนต่อไปได้เลยแต่ถ้าเกิดข้อผิดพลาดให้ทำการต่อไปนี้เอง

```
cd /usr/src/lib/libsm
make clean
make
```

คอมไพล์และติดตั้ง bin

```
cd /usr/src/bin
make clean
make
make install
```

ต่อไปพิมพ์

```
cd /usr/src/contrib/ipfilter/tools
make clean
make

cd /usr/src/sbin/ipf
make clean
make
```

ต่อไป comile และติดตั้ง sbin ต้องแน่ใจว่า /usr/share/docatm ไดเรคทอรีมีอยู่ไม่เช่นนั้นจะเกิดข้อผิดพลาดแต่ถ้ายังไม่มีให้พิมพ์บรรทัดบนก่อนเพื่อสร้าง

```
mkdir -p /usr/share/doc/atm
cd /usr/src/sbin
make clean
make
make install
```

ต่อไป compile และติดตั้ง usr.bin ก่อนอื่น usr.bin ต้องการ libraries ก่อนต้องสร้างให้ดังนี้

```
cd /usr/src/lib/bind
make clean
make

cd /usr/src/lib/libtelnet
make clean
make

cd /usr/src/lib/libsmutil
make clean
make

cd /usr/src/lib/libsmdb
make clean
make
```

ติดตั้ง usr.bin

```
cd /usr/src/usr.bin
make clean
make
mkdir -p /usr/share/info
make install
```

8. Copy the binaries over

ขณะนี้คุณต้อง download mkmini.sh และ minibsd6.files ที่จะทำให้สามารถ copy binary มาสู่ miniBSD ได้

```
cd /root
fetch http://www.ultradesic.com/pub/miniBSD_Files/minibsd6.files
fetch http://www.ultradesic.com/pub/miniBSD_Files/mkmini.sh
```

ถ้าคุณใช้ FreeBSD เวอร์ชั่นที่สูงกว่า 6.1 คุณอาจจะต้องแก้ minibsd6.files และเปลี่ยน lacation ของ /sbin/setkey ไปเป็น /usr/sbin/setkey

แนะนำให้บันทึกไฟล์ใน root ไดเรคทอรีของ FreeBSD jail ก่อน

```
cd /root
chmod 0700 /root/mkmini.sh
/root/mkmini.sh /root/minibsd6.files
```

ถ้าคุณติดตั้ง source package ใน FreeBSD jail คุณสามารถ copy termcap และ services ไฟล์โดยการทำดังนี้

```
cp /usr/src/release/picobsd/mfs_tree/etc/termcap /usr/minibsd/usr/share/misc
cp /usr/src/release/picobsd/mfs_tree/etc/services /usr/minibsd/usr/share/misc
```

9. Configuration boot files

เมื่อคุณบูตระบบ miniBSD อาจจะไม่สามารถหา /boot/beasties.4th เพราะเราไม่ได้ copy เอามาถ้าคุณต้องการ beasties menu แสดงตอน boot คุณต้อง copy มาตามนี้

```
cp /boot/beastie.4th /usr/minibsd/boot
cp /boot/screen.4th /usr/minibsd/boot
cp /boot/frames.4th /usr/minibsd/boot
```

แต่ถ้าคุณไม่สนใจหรือต้องการ beasties menu นั้นคุณสามารถ comment หรือลบได้ที่ /usr/minibsd/boot/loader.rc

ดังนี้ โดยการเพิ่มเครื่องหมาย \ ลงไปหน้าข้อความ

```
\ Load in the boot menu
\ include /boot/beastie.4th
\ Start the boot menu
\ beastie-start
```

ถ้าคุณต้องการให้ระบบไม่มีการ delay หรือ delay น้อยคุณต้องแก้ไขดังนี้ที่ไฟล์ /usr/minibsd/boot/loader.rc และพิมพ์

```
autoboot 0
```

10. Kernel Compile

ถ้าคุณวางแผนที่จะใช้ GENERIC kernel ให้พิมพ์ดังนี้

```
mkdir -p /usr/minibsd/boot/kernel
cp /boot/kernel/kernel /usr/minibsd/boot/kernel
cd /usr/minibsd/boot/kernel
gzip -9 kernel
```

แต่ถ้าคุณต้องการที่จะ compile kernel เองให้เริ่มทำตามนี้

```
cd /usr/src/sys/i386/conf
cp GENERIC MINIBSD
```

ถ้าคุณใช้ soekris รุ่น net4801/net4826 คุณต้องการที่จะใช้ kernel config ของ ultradesic ให้ทำการ download ตามนี้

```
cd /usr/src/sys/i386/conf
fetch http://www.ultradesic.com/pub/miniBSD_Files/MINIBSD
```

เข้าไปแก้ไข ไฟล์ MINIBSD แล้วบันทึก

หลังจากนั้นทำตามนี้

```
/usr/sbin/config MINIBSD
cd ../compile/MINIBSD
make clean && make cleandepend && make depend && make
gzip -9 kernel
mkdir -p /usr/minibsd/boot/kernel
cp kernel.gz /usr/minibsd/boot/kernel
```

11. Copying the libraries

เรายังไม่ได้คัดลอก libraries มาสู่ /usr/lib ซึ่งไม่มีรายการอยู่ใน minibsd6.files เพราะรายการของ libraries ที่ต้องการต้องอาศัยการติดตั้ง ให้ download ไฟล์ mklibs.sh ดังนี้

```
cd /root
fetch http://www.ultradesic.com/pub/miniBSD_Files/mklibs.sh
chmod 0700 /root/mklibs.sh
/root/mklibs.sh > /root/minibsd.libs
```

หลังดูที่ output ของ mimibsd.libs แล้วถูกต้องให้รันสคริปต์ดังนี้

```
/root/mkmini.sh minibsd.libs
```

Library ที่ยังขาดอยู่คือ PAM ให้ copy ดังนี้

```
cp -p /usr/lib/pam* /usr/minibsd/usr/lib
cp -p /usr/lib/*opie* /usr/minibsd/usr/lib
```

12. Populating /etc

ถึงตอนนี้คุณต้องการที่จะปรับแต่ง miniBSD ของคุณเช่นการตั้งค่ารหัสผ่าน เพิ่ม user เป็นต้น คุณอาจจะ copy จาก /etc ของเดิมกได้ภายใต้ FreeBSD jail แต่เมื่อตอนติดตั้งครั้งแรกเรายังไม่ได้ตั้งรหัสผ่านที่ FreeBSD jail ให้ตั้งรหัสผ่านและเพิ่ม user ก่อน เมื่อคุณได้ตั้งรหัสผ่าน เพิ่ม user เรียบร้อยแล้วและปรับแต่งที่ FreeBSD jail แล้วจึงเริ่ม copy มายัง MiniBSD แต่ถ้าคุณมีการเพิ่ม service คุณจะต้อง copy มาด้วยแล้ว ใช้ script นี้ในการ copy ไดเรคทอรี /etc

```
cd /root
fetch http://www.ultradesic.com/pub/miniBSD_Files/copy-etc-6.sh
chmod 0700 /root/copy-etc-6.sh
/root/copy-etc-6.sh
```

ซึ่ง script จะ copy ไดเรคทอรี /etc ตามนี้

- /etc/auth.conf
- /etc/crontab
- /etc/defaults/devfs.rules
- /etc/defaults/periodic.conf
- /etc/defaults/rc.conf
- **/etc/fstab**
- /etc/ftpusers
- /etc/gettytab
- /etc/group
- /etc/host.conf
- /etc/hosts
- /etc/hosts.allow
- /etc/inetd.conf
- /etc/localtime
- /etc/login.access
- /etc/login.conf
- /etc/master.passwd
- /etc/mtree/ (copy the directory plus its contents)
- /etc/network.subr
- /etc/networks
- /etc/newsyslog.conf
- /etc/nsswitch.conf
- /etc/pam.d/ (copy the directory plus its contents)
- /etc/passwd
- /etc/profile
- /etc/protocols
- /etc/pwd.db
- /etc/rc
- **/etc/rc.conf**
- /etc/rc.d/ (copy the directory plus its contents)
- /etc/rc.firewall
- /etc/rc.shutdown

- /etc/rc.subr
- **/etc/resolv.conf**
- /etc/services
- /etc/shells
- /etc/spwd.db
- /etc/sysctl.conf
- /etc/syslog.conf
- /etc/termcap -> /usr/share/misc/termcap
- **/etc/ttys**

ไฟล์ /etc/fstab ยังไม่มีอยู่ใน FreeBSD jail /etc ไดเรคทอรี คุจะต้องสร้างใหม่ได้โดยไม่ยากเพราะคุณต้องพิมพ์แค่สองบรรทัดคือ (อย่างน้อยคุณตั้งเอาไว้สอง partition) โดยการสร้างไฟล์ /usr/minibsd/etc/fstab ด้วย text editor ที่คุณชอบแล้วพิมพ์สองบรรทัดนี้ลงไป

```
/dev/ad0a       /       ufs     ro      1   1
proc            /proc   procfs  rw      0   0
```

ไฟล์ host.conf ถูกสร้างขึ้นโดยอัตโนมัติเมื่อ ระบบ FreeBSD บูตขึ้นมาถ้าไม่มีอยู่และไม่มีอยู่ใน FreeBSD jail จะต้องสร้าขึ้นหรือสร้าง symlink ถ้าต้องการทำ symlink ให้พิมพ์ดังนี้

```
cd /usr/minibsd/etc

ln -s /tmp/host.conf
```

ไฟล์ nsswitch.conf ถูกสร้างขึ้นโดยอัตโนมัติเมื่อบูตระบบแต่ถ้าไม่มีทำดังนี้

```
cd /usr/minibsd/etc

ln -s /tmp/nsswitch.conf
```

FreeBSD jail ของเรายังมีค่าการปรับแต่งที่ไม่ถูกต้องให้เข้าไปแก้ไขดังนี้ เช่น

```
hostname="workbox.ultradesic.build"
ifconfig_sis0="192.168.1.4 netmask 255.255.255.0"
defaultrouter="192.168.1.1"
sshd_enable="NO"
usbd_enable="NO"
sendmail_enable="NONE"
inetd_enable="NO"
portmap_enable="NO"
update_motd="NO"
varsize=8192
varmfs="YES"
tmpmfs="YES"
tmpsize=8192
```

ในบรรทัดที่ varmfs และ tmpmfs ใน /usr/minibsd/etc/rc.conf เราสามารถใส่ไว้ในไฟล์ /usr/minibsd/etc/fstab ได้โดยไม่ต้องใส่ใน /usr/minibsd/etc/rc.conf ดังนี้

```
md /var mfs rw,-s8m 2 0

md /tmp mfs rw,-s8m 2 0
```

ซึ่งจะทำงานเหมือนกันแต่ผู้คนมักจะ กำหนด mount point ไว้ในไฟล์ fstab

ถ้าบูตระบบแล้วพบข้อความ

```
/etc/rc: WARNING: Dump device does not exist. Savecore not run.
```

ข้อความนี้จะปรากฏเมื่อขาด swap partition เพื่อที่จะกำจัดข้อความนี้ให้แก้ไฟล์ /usr/minibsd/etc/rc.conf ตามนี้

dumpdev="NO"

บนระบบ embedded แนะนำให้แก้ไขไฟล์ /usr/minibsd/etc/ttys และปิด virtual console ทั้งหมดซึ่งไม่ได้ใช้และสิ้นเปลืองหน่วยความจำ อาจจะเปิดไว้เท่าที่ต้องการดังนี้

```
# If console is marked "insecure", then init will ask for the root password
# when going to single-user mode.
console    none                 unknown    off secure
#
ttyv0    "/usr/libexec/getty Pc"        cons25    off  secure
# Virtual terminals
ttyv1    "/usr/libexec/getty Pc"        cons25    off  secure
ttyv2    "/usr/libexec/getty Pc"        cons25    off  secure
ttyv3    "/usr/libexec/getty Pc"        cons25    off  secure
ttyv4    "/usr/libexec/getty Pc"        cons25    off  secure
ttyv5    "/usr/libexec/getty Pc"        cons25    off  secure
ttyv6    "/usr/libexec/getty Pc"        cons25    off  secure
ttyv7    "/usr/libexec/getty Pc"        cons25    off  secure
ttyv8    "/usr/X11R6/bin/xdm -nodaemon"    xterm    off secure
# Serial terminals
# The 'dialup' keyword identifies dialin lines to login, fingerd etc.
ttyd0    "/usr/libexec/getty std.9600"    vt100    on secure
```

13. สร้าง Binary image

ขั้นตอนสุดท้ายคือการสร้างทั้งหมดไว้ใน binary ไฟล์เพียงไฟล์เดียวเพื่อนำไป write ลงสู่ cf card หรือ Harddisk ถ้าต้องการใช้ script ให้ download แล้วทำดังนี้

```
cd /root

fetch http://www.ultradesic.com/pub/miniBSD_Files/build-image.sh

chmod 0700 /root/build-image.sh

/root/build-image.sh
```

เมื่อสั่งรัน script แล้วควรได้ผลลัพธ์ดังนี้

```
MiniBSD ~ # ./build-image.sh 0.1.0

Generating an empty binary image file...[ OK ]
Linking binary image to device file.....[ OK ]
Creating MBR on binary image.............[ OK ]
Partitioning binary image.................[ OK ]
Formatting the binary image..............[ OK ]
Mounting temporary mount point..........[ OK ]
Writing miniBSD to the binary image.....[ OK ]
Unmounting temporary mount point........[ OK ]
Unlinking binary image...................[ OK ]
Compressing the binary image.............[ OK ]
Checking file integrity...................[ OK ]
Generating MD5 SUM of binary image......[ OK ]
Done!
```

แต่ถ้าต้องการทำเองตามขั้นตอนให้ทำตามนี้

Genarating the flash image

เราใช้ vnconfig เพื่อกำหนด virtual disk และเราสามารถใช้ bsdlabel เพื่อรู้ขนาดของ flash card ในหน่วย sectors (512 byte units) ให้พิมพ์คำสั่งนี้เพื่อทราบขนาด

```
bsdlabel -Awn ad[n] auto | grep sectors/unit
```

ถ้า bsdlabel รายงานข้อผิดพลาดให้พิมพ์

```
dd if=/dev/zero of=/dev/ad[n] bs=1k count=20
```

นี่เป็นการลบข้อมูลทุกพาร์ติชันที่รายงานบน CF card

ต่อไปเป็นการสร้าง disk image ของขนาดเท่ากับ CF Card ดังนี้

```
dd if=/dev/zero of=/usr/minibsd-disk.bin bs=512 count=[number of sectors on your flash card]
```

อนุญาติให้ disk image files เป็นเช่นเดียวกับ vn device ดังนี้

```
mdconfig -a -t vnode -u 0 -f /usr/minibsd-disk.bin
```

virtual disk ของเราต้องเป็น partitions และสร้าง file system ดังนี้

```
bsdlabel -Bw md0 auto
bsdlabel -e md0
```

คุณต้องใช้ text editor ที่ชอบเพื่อหาบรรทัดที่เริ่มด้วย C: และ copy มันเพื่อเปลี่ยนเป็น a: และเปลี่ยนชนิดเป็น 4.2BSD นี่คือ root partition ที่ใช้ทั้ง slice ถ้าคุณต้องการ partition ที่สองเพื่อเก็บ configuration ให้ลดขนาดของ a: slice โดยการกำหนดค่าของ partition ที่สองที่คุณต้องการ หลังจากนั้นให้ทำอีกบรรทัดที่เริ่มจาก e: และลบขนาดออกจาก a: บันทึก จำไว้ว่าเราไม่สามารถสร้าง swap partition ได้บน CF Card เพราะมันช้ามากๆ

หลังจากที่คุณบันทึกแล้วคุณควรสร้าง ASCII file ที่บรรจุข้อมูลตาราง คุณสามารถนำเข้า ASCII file ได้ในอนาคตตามคำสั่งนี้

```
bsdlabel md0 > /root/minibsd-disklabel
```

ในครั้งต่อไปคุณสามารถรัน bsdlabel –e md0 คุณสามารถใช้คำสั่งนี้แทนได้

```
bsdlabel -R md0 /root/minibsd-disklabel
```

ต่อไปสร้าง newfs และ mount พาร์ติชันของเราที่สร้างขึ้น newfs ยังไม่ได้มีใน /etc/fstab เพราะคุณไม่มี /etc/fstab ใน FreeBSD jail คุณสามารถสร้างไฟล์ว่างๆก่อนที่จะรัน newfs พิมพ์ดังนี้

```
touch /etc/fstab
newfs -b 8192 -f 1024 -U /dev/md0a
mount /dev/md0a /mnt
```

ถ้าคุณได้สร้าง partition ที่สองอย่าลืม newfs พาร์ติชันนั้นด้วย

ต่อไปคือการคัดลอกไฟล์จาก minibsd ไปยัง /mnt ดังนี้

```
(cd /usr/minibsd ; tar cPf - .) | (cd /mnt ; tar xf - );
```

หลังจากนั้น clean up ดังนี้

```
cd /
umount /mnt
mdconfig -d -u 0
```

14. Writing the binary image to the media

คำสั่งสำหรับการเขียนลง CF Card ดังนี้

```
dd if=/usr/minibsd-disk.bin of=/dev/ad[n] ibs=8k obs=8k
```

ต้องระวัง [n] ต้องกำหนดให้ถูกต้องตาม CF card หรือสื่อที่ใช้

ถ้าคุณได้ บีบอัดไฟล์ไว้ให้พิมพ์ดังนี้

```
gzcat /usr/minibsd-disk.bin.gz | dd of=/dev/ad[n] ibs=8k obs=8k
```

หรือ

```
cat /usr/minibsd-disk.bin.gz | gunzip | dd of=/dev/ad[n] ibs=8k obs=8k
```

ถ้าคุณต้องการดูการทำงานให้พิมพ์ Ctrl - T

15. First Boot

หลังจากที่ได้ทำตามขั้นตอนนั้นแล้วให้ทดลอง boot ระบบด้วย CF Card ที่ได้ทำมา

ประวัติผู้เขียน

ชื่อผู้เขียน	นายชาตรี ชะโลธร
วัน เดือน ปีเกิด	21 มีนาคม 2515
สถานที่เกิด	จังหวัดสมุทรสงคราม
วุฒิระดับการศึกษา	ครุศาสตรบัณฑิต สาขาคอมพิวเตอร์ศึกษา
สถาบันที่สำเร็จการศึกษา	สถาบันราชภัฏกาญจนบุรี
ปีที่สำเร็จการศึกษา	2548