ระบบควบคุมการกระจายภาระงานของเครื่องแม่ข่าย

Load Balancing Controller for Server

โดย

นายทศพล บุตรพรหม

นายทิฏฐวัฒน์ ค่าเมือง

นายธเนศ สีเชียงพิมพ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

ระบบควบคุมการกระจายภาระงานของเครื่องแม่ข่าย
Load Balancing Controller for Server

โดย

นายทศพล บุตรพรหม

นายทิฏฐวัฒน์ คำเมือง

นายธเนศ สีเชียงพิมพ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

ระบบควบคุมการกระจายภาระงานของเครื่องแม่ข่าย
Load Balancing Controller for Server

โดย

นายทศพล	บุตรพรหม	รหัสนักศึกษา 53010578
นายทิฏฐวัฒน์	คำเมือง	รหัสนักศึกษา 53010589
นายธเนศ	สีเชียงพิมพ์	รหัสนักศึกษา 53010699

อาจารย์ที่ปรึกษา

รศ.ดร.สุวิพล สิทธิชีวภาค

รศ.เกรียงไกร วงศ์โรจนภรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

ผ่านการตรวจรูปเล่มแล้ว

(.....)
อาจารย์ที่ปรึกษา
10 / ต.ค. / 57

ผ่านการตรวจชิ้นงานแล้ว

(.....)
กรรมการผู้ตรวจชิ้นงาน
1 / พ.ย. / 57

ปริญญานิพนธ์ปีการศึกษา 2556

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมการกระจายภาระงานของเครื่องแม่ข่าย

Load Balancing Controller for Server

ผู้จัดทำ

1. นายทศพล บุตรพรหม 53010578
2. นายทิฏฐวัฒน์ คำเมือง 53010589
3. นายธเนศ สีเชียงพิมพ์ 53010699

..... อาจารย์ที่ปรึกษา

(รศ.ดร.สุวิพล สิทธิชีวภาค)

..... อาจารย์ที่ปรึกษา

(รศ.เกรียงไกร วงศ์โรจนภรณ์)

กิตติกรรมประกาศ

เนื่องด้วยความกรุณาของ รศ.ดร.สุวิพล สิทธิชีวภาค ซึ่งเป็นอาจารย์ที่ปรึกษาปริญญานิพนธ์นี้ ที่ได้ให้คำเสนอแนะ แนวคิด ตลอดจนแก้ไขข้อบกพร่องต่างๆมาโดยตลอด จนปริญญานิพนธ์นี้เสร็จสมบูรณ์ ผู้ศึกษาจึงขอกราบขอบพระคุณเป็นอย่างสูง

การจัดทำปริญญานิพนธ์นี้มีปัญหามากมายในระหว่างการจัดทำ จึงขอขอบคุณ คุณกฤษฎ์ธนิก ศรีธนสาร ที่ให้คำปรึกษาและแก้ไขข้อพร่องต่างๆ ให้เป็นอย่างดี

ขอขอบพระคุณ คุณพ่อ คุณแม่ ผู้ให้กำเนิดและเป็นกำลังใจที่ดีเสมอมา ซึ่งให้ความรักและความอบอุ่น ซึ่งถ้าขาดสิ่งหนึ่งสิ่งใดไปอาจทำให้ปริญญานิพนธ์ไม่สำเร็จ

ขอขอบคุณสำนักบริการคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่เอื้อเฟื้อสถานที่ในการศึกษาและทดลองในการทำปริญญานิพนธ์ในครั้งนี้

สุดท้ายนี้ต้องขอขอบคุณทุกคนที่มีส่วนเกี่ยวข้องทั้งหมด ซึ่งเป็นเหตุให้ปริญญานิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยดี

นายทศพล บุตรพรหม
นายทิฏฐวัฒน์ คำเมือง
นายธเนศ สีเชียงพิมพ์
ผู้จัดทำ

ระบบควบคุมการกระจายภาระงานของเครื่องแม่ข่าย

Load Balancing Controller for Server

โดย นายทศพล บุตรพรหม 53010578
นายทิฏฐวัฒน์ คำเมือง 53010589
นายธเนศ สีเชียงพิมพ์ 53010699

อาจารย์ที่ปรึกษา รศ.ดร.สุวิพล สิทธิชีวภาค
รศ.เกรียงไกร วงศ์โรจนภรณ์

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการพัฒนาแอพพลิเคชั่นและออกแบบอุปกรณ์แสดงผลเพื่อจัดการการใช้งานเซอร์วิส IPVSADM ให้สามารถใช้งานได้ง่ายขึ้น ซึ่งเซอร์วิส IPVSADM คือเซอร์วิสที่ทำหน้าที่ในการกระจายภาระงานจากเครื่องไคลแอนต์ไปยังเครื่องแม่ข่ายหลายเครื่องให้ทำงานร่วมกันได้อย่างสมดุล เพื่อรองรับการใช้งานจากผู้ใช้บริการในจำนวนมากได้ ทำให้การทำงานของเครือข่ายมีเสถียรภาพมากขึ้น โดยในการพัฒนาแอพพลิเคชั่นจะถูกพัฒนาด้วยภาษาจาวา และโปรเซสบิวเดอร์ไลบรารี่ที่ทำงานร่วมกับลีนุกซ์ เวอชวล เซิร์ฟเวอร์ ซึ่งเป็นเซอร์วิสที่ทำงานภายใต้ระบบปฏิบัติการลีนุกซ์ แอพพลิเคชั่นดังกล่าวสามารถทำการรับค่าพารามิเตอร์จากผู้ดูแลระบบและทำการแปลงค่าพารามิเตอร์ไปเป็นคำสั่งในการใช้งานเซอร์วิส IPVSADM เพื่อเริ่มการทำงานของโหลดบาลานซ์เซิร์ฟเวอร์ ตามที่ผู้ดูแลระบบต้องการได้ และได้สร้างอุปกรณ์ในการแสดงผลและเลือกฟังก์ชั่นการทำงานของเซอร์วิส IPVSADM ให้ใช้งานง่ายขึ้นโดยใช้หน้าจอโมดูลแอลซีดีในการแสดงผลและใช้คีย์แพดในการเลือกฟังก์ชั่นการทำงานเซอร์วิส IPVSADM ให้ทำงานตามที่ผู้ดูแลระบบต้องการได้ โดยผู้จัดทำปริญญานิพนธ์ได้พัฒนาให้อุปกรณ์แสดงผลสามารถทำงานร่วมกับแอพพลิเคชั่นโหลดบาลานซ์เซิร์ฟเวอร์ ได้และสามารถใช้งานง่ายขึ้น

ABSTRACT

This thesis aims to develop an application and design display device to management of IPVSADM applications service. This service provides and distribute the workloads from clients via virtual servers to work together with balanced to support the using of users makes the network more stable. The application is developed by Java. The processes Builder's library that works with Linux Virtual servers is a service that runs under the Linux operating system. The application get the parameters from the system administrator then converts to the commands for use IPVSADM service. LOAD BALANCE SERVER is started by administrator 's requirements and has built in display devices and select functions of the service, IPVSADM is easily to be used by using LCD Module in the display and keypad to select functions of Service IPVSADM. The display and control panel had been developed to work with LOAD BALANCE SERVER application for easy to use.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญตาราง

หน้า

ตารางที่

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันระบบการสื่อสารทางอินเตอร์เน็ตมีความสำคัญเป็นอย่างมาก ซึ่งอินเตอร์เน็ตมีการให้บริการในหลายๆด้านแตกต่างกันออกไป ซึ่งที่ใช้กันอย่างแพร่หลายคือ เว็บไซต์ ซึ่งจะต้องใช้ระบบเครื่องแม่ข่ายหรือเซิร์ฟเวอร์เข้ามาช่วยในการทำงานเพื่อให้บริการเว็บเซิร์ฟเวอร์ ซึ่งในการใช้เครื่องเซิร์ฟเวอร์ขนาดใหญ่นี้ บางเว็บที่ต้องรองรับการใช้งานของผู้ใช้งาน (Users) ในจำนวนมากๆ อาจจะเกิดปัญหากับเซิร์ฟเวอร์ที่ไม่สามารถให้การรองรับการใช้งานในจำนวนมากๆ ในคราวเดียวกันได้ ซึ่งอาจเป็นสาเหตุที่ทำให้เซิร์ฟเวอร์ล่ม และส่งผลให้เว็บไซต์ใช้งานไม่ได้ ดังนั้นมีเทคนิคหนึ่งที่ช่วยให้เซิร์ฟเวอร์สามารถรองรับการใช้งานของผู้ใช้งาน (Users) จำนวนมากได้ ซึ่งเทคนิคดังกล่าวการคือการใช้โหลดบาลานซ์ (Load Balance) ซึ่งจะช่วยในการกระจายภาระงานที่เข้ามาจากผู้ใช้งาน (Users) ให้กระจายไปยังกลุ่มเว็บเซิร์ฟเวอร์ที่ได้ทำการจัดเตรียมไว้จำนวนตั้งแต่สองเครื่องขึ้นไป สำหรับปัญหาดังกล่าวนั้นในบางเว็บไซต์ที่ให้บริการมีปริมาณผู้ใช้งาน (Users) จำนวนมาก Access เข้ามาทำงานพร้อมกันเป็นจำนวนมาก โดยเฉพาะช่วงเวลาสำคัญ (Business times) เช่นการลงทะเบียนเรียนของสถาบันที่มีผู้ใช้งาน (Users) ต้องแย่งกันเข้ามาใช้งานทำให้ระบบเซิร์ฟเวอร์รองรับภาระงานไม่ไหวทำให้เว็บเซิร์ฟเวอร์ทำงานช้ามาก ส่งผลให้ผู้ใช้งาน (Users) ใช้งานได้อย่างไม่เต็มประสิทธิภาพและสิ่งที่แย่ที่สุดในบางครั้งก็คือการเชื่อมต่อล้มเหลว (connection error) จนไม่สามารถใช้งานระบบได้ ถึงแม้จะแก้ไขโดยการเพิ่มประสิทธิภาพของเซิร์ฟเวอร์ให้ดีขึ้น เช่น Memory Sever แต่ก็ไม่สามารถแก้ไขปัญหาได้อย่างเด็ดขาดทำให้เกิดปัญหาทั้งทางด้านธุรกิจและอื่นๆได้ จากปัญหาดังกล่าวผู้ศึกษาจึงได้มีการประยุกต์ใช้โหลดบาลานซ์ (Load Balance) ซึ่งเป็นตัวควบคุมการกระจายภาระงานให้กับเครื่องแม่ข่าย โดยโหลดบาลานซิ่ง (Load balancing) คือจัดเตรียมเซิร์ฟเวอร์หลายๆตัวเพื่อทำการรองรับภาระงานจากการกระจายโหลดของเซอร์วิสโหลดบาลานซ์ (Load balance services) ที่กระจายโหลดการเข้ามาใช้งานของผู้ใช้งาน (Users) ไปยังเครื่องเซิร์ฟเวอร์ในกลุ่ม เพื่อให้สามารถรองรับจำนวนผู้ใช้งาน (Users) ที่เข้ามาใช้งานได้มากขึ้น นอกจากนี้ยังมีคุณสมบัติ Fail Over คือหากมีเครื่องเซิร์ฟเวอร์ตัวใดตัวหนึ่งภายในระบบไม่สามารถทำงานได้เช่น Down อยู่หรือไม่สามารถรับงานหรือผู้ใช้งาน (Users) เพิ่มได้เนื่องจากทรัพยากรที่ใช้งานไม่พอ เซอร์วิสโหลดบา-

ลานซ์ (Load balance services) ก็จะทำการกระจายโหลดไปยังเซิร์ฟเวอร์เครื่องอื่นๆแทนซึ่งทำให้ไม่สิ้นเปลืองทรัพยากรในการทำงานของระบบ

ดังนั้นโครงงานนี้ผู้ศึกษาจึงได้นำเสนอระบบการทำโหลดบาลานซิ่ง (Load balancing) มาใช้ในการจัดการการเข้ามาใช้งานเว็บไซต์ของผู้ใช้งาน (Users) และกระจายภาระงานไปยังเซิร์ฟเวอร์ภายในกลุ่มเพื่อให้สามารถรองรับจำนวนผู้ใช้งาน (Users) ที่เข้ามาใช้งานได้มากขึ้น โดยได้ศึกษาและพัฒนาแอพพลิเคชั่นการจัดการใช้งานเซอร์วิสโหลดบาลานซ์ (Load balance services) จากระบบปฏิบัติการลีนุกซ์อูบุนตู (Linux Ubuntu) โดยในการพัฒนาแอพพลิเคชั่นจะทำการพัฒนาโดยภาษาจาวา (Java) และทำการสร้างหน้าต่างอินเตอร์เฟส (interface) ด้วยโปรแกรมเน็ตบีนส์ (NetBeans) และทำการติดต่อสื่อสารระหว่างหน้าต่างอินเตอร์เฟส (interface) กับลีนุกซ์อูบุนตู (Linux Ubuntu) โดย Process builder library และได้สร้างอุปกรณ์ในการจัดการการใช้งานโหลดบาลานซ์(Load Balance) ขึ้นมาซึ่งสามารถป้อนค่าผ่านอุปกรณ์โดยคีย์แพดได้โดยส่งคำสั่งไปยังไมโครคอนโทรเลอร์เพื่อติดต่อสื่อสารกับโหลดบาลานซ์ (Load Balance) โดยผ่านพอร์ตอนุกรม RS-232 และแสดงผลผ่านหน้าจอโมดูลแอลซีดี ซึ่งทำให้การจัดการใช้งานเซอร์วิสโหลดบาลานซ์ (Load balance services) ทำได้โดยง่ายและก่อให้เกิดผลที่มีประสิทธิภาพสูงสุดต่อระบบได้

1.2 วัตถุประสงค์

1.2.1 เพื่อจัดการระบบเว็บเซิร์ฟเวอร์ให้สามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ

1.2.2 เพื่อพัฒนาแอพพลิเคชั่นและอุปกรณ์ในการจัดการการใช้งานโหลดบาลานซ์ (Load Balance) ให้สามารถใช้งานได้

1.2.3 เพื่อให้เซิร์ฟเวอร์สามารถรองรับการเข้ามาใช้งานของผู้ใช้งาน (Users) จำนวนมากๆได้

1.2.4 เพื่อแก้ปัญหาระบบเซิร์ฟเวอร์ทำงานช้าหรือล่มและช่วยในการประหยัดทรัพยากรของระบบเว็บเซิร์ฟเวอร์

1.3 ขอบเขตของโครงงาน

1.3.1 พัฒนาแอพพลิเคชั่นด้วย NetBeans IDE และภาษาจาวา (Java) ที่ทำงานใต้ Process Builder Library

1.3.2 สร้างฮาร์ดแวร์ด้วยไมโครคอนโทรลเลอร์ MCS-51 ที่พัฒนาด้วย Firmware ด้วยภาษาซีโดยป้อนค่าอินพุตพารามิเตอร์ด้วยคีย์แพด (Keypad) และแสดงผลด้วยโมดูลแอลซีดี

1.3.3 ฮาร์ดแวร์ทำการติดต่อสื่อสารข้อมูลกับลีนุกซ์ด้วยพอร์ตอนุกรม RS-232 ซึ่งติดต่อสื่อสารกันด้วยการทำอินเตอร์รัพท์และมี Process Builder ในการรับค่าข้อมูลจากไมโครคอนโทรลเลอร์ แปลงเป็น message เพื่อส่งค่าไปยังลีนุกซ์

1.3.4 ทำงานภายใต้ชั้นของ Application, Shell Script, OS และ Hardware

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

ในการศึกษาและจัดทำปริญญานิพนธ์ในหัวข้อ ระบบควบคุมการกระจายภาระงานของเครื่องแม่ข่าย มีทฤษฎีและเอกสารที่เกี่ยวข้องกับระบบควบคุมการกระจายภาระงานของเครื่องแม่ข่าย โดยได้อธิบายถึงรายละเอียดและหลักการทำงานต่างๆ ที่สำคัญดังต่อไปนี้

2.1 โหลดบาลานซ์ (Load Balance)

2.1.1 โหลดบาลานซิ่ง (Load balancing)

โหลดบาลานซิ่ง (Load balancing) คือการจัดกลุ่มของคอมพิวเตอร์หลายๆตัวเพื่อแบ่งงานกันหรือกระจายโหลดการใช้งานของผู้ใช้งาน (Users) ไปยังคอมพิวเตอร์ภายในกลุ่ม เพื่อให้สามารถรับจำนวนผู้ใช้งาน (Users) ที่เข้ามาใช้งานได้มากขึ้น หรือสามารถรับงานที่เข้ามาได้มากขึ้น นอกจากนั้นยังมีคุณสมบัติของ Fail Over คือหากมีคอมพิวเตอร์ภายในกลุ่มไม่สามารถทำงานได้ เช่น Down อยู่ หรือไม่สามารถรับโหลดเพิ่มได้เนื่องจาก resource ที่ใช้ทำงานไม่พอ โหลดบาลานซ์เซิร์ฟเวอร์ที่เป็นตัวแจกโหลดให้คอมพิวเตอร์ภายในกลุ่มก็จะส่งโหลดไปยังคอมพิวเตอร์เครื่องอื่นๆ แทนจนกว่าคอมพิวเตอร์เครื่องนั้นจะกลับมาใช้งานได้ใหม่

2.1.2 ระบบโหลดบาลานซ์ (Load Balance System)

ระบบโหลดบาลานซ์ (Load Balance System) เป็นระบบที่จะช่วยแก้ไขปัญหาการทำงานของระบบที่มีการใช้งานหนักได้เป็นอย่างดี โดยการทำโหลดบาลานซ์ คือการจัดกลุ่มของเซิร์ฟเวอร์ (Servers) หลายๆ เครื่องเพื่อแบ่งงานกันทำงานหรือกระจายโหลดการใช้งานของผู้ใช้งาน (Users) ไปยังเซิร์ฟเวอร์ (Servers) เครื่องต่างๆ ภายในกลุ่มเพื่อรองรับการทำงาน เช่น เว็บไซต์ที่มีผู้เข้าชมจำนวนมากจะมีการเรียกใช้งาน Web Server, Database Server สูงจนทำให้เซิร์ฟเวอร์ (Server) เครื่องเดียวไม่สามารถรองรับการทำงานได้ทั้งหมด โหลดบาลานซ์สามารถรองรับกับจำนวนผู้ใช้งาน (Users) ที่เข้ามาใช้งานได้มากขึ้น ซึ่งปัญหาหลักๆ ของคนทำเว็บไซต์ขนาดใหญ่คือ จะทำอย่างไรให้ระบบสามารถรองรับการใช้งานของผู้ใช้งาน (Users) จำนวนมากๆได้ และเทคนิคหนึ่งที่มีการใช้กันอย่างแพร่หลายก็คือการทำโหลดบาลานซ์

2.1.3 วิธีการกระจายภาระงานของ Load Balance

2.1.3.1 แบบ Round Robin

วิธีการวนรอบ (Round robin) วิธีการนี้จะสลับการทำงานของแต่ละโหนดของ Real Server แบบวนไปเรื่อยๆ เช่น ครั้งแรกใช้งานเครื่องที่ 1 ครั้งที่สองจะใช้งานเครื่องที่ 2 เมื่อครบรอบจะกลับมาเริ่มต้นใหม่อีกครั้ง ข้อดีของวิธีนี้คือง่ายที่สุดและใช้งบประมาณน้อยที่สุด แต่เป็นวิธีนี้มีข้อเสียค่อนข้างมาก ข้อเสียที่สำคัญคือไม่สามารถควบคุมการทำงานได้ เนื่องจากการทำงานด้วยวิธีนี้จะมีแต่การทำ Round Robin เท่านั้น ไม่สามารถปิดการแบ่งโหลดการทำงานแบบ real-time ได้ และหากต้องการ register server เข้าไปในโหลดจะต้องใช้เวลานานเพราะการทำ mapping DNS นั้นจะมีการ cache ข้อมูลไว้ตามที่ต่างๆ ซึ่งเราไม่สามารถควบคุมได้

2.1.3.2 แบบ Weighted Round Robin

วิธีการวนรอบแบบถ่วงน้ำหนัก (Weighted Round Robin) วิธีการทำงานของวิธีนี้คือจะทำงานคล้ายกับ Round Robin แต่เราสามารถที่จะทำการกระจายงานโดยคำนึงถึงประสิทธิภาพในการทำงานที่แตกต่างกันของเซิร์ฟเวอร์ต่างๆที่อยู่ในระบบโดยวิธีการนี้จะทำการกำหนดค่าตัวแบ่งน้ำหนักที่บ่งบอกถึงประสิทธิภาพในการทำงานของเซิร์ฟเวอร์แต่ละเครื่องที่อยู่ในระบบคลัสเตอร์ วิธีการนี้ไม่ได้คำนึงถึงจำนวนการติดต่อของแต่ละเซิร์ฟเวอร์ที่อยู่ในระบบ ซึ่งทำให้มี Overhead น้อย และส่งผลให้ระบบสามารถมีจำนวนเซิร์ฟเวอร์ภายในระบบคลัสเตอร์ได้มาก แต่วิธีการนี้อาจจะนำไปสู่การกระจายงานอย่างไม่สมดุลภายในระบบคลัสเตอร์ได้ถ้ามีภาระงานของการร้องขอบริการมีความแตกต่างกันมาก กล่าวคือภาระการร้องขอบริการจำนวนมากที่มีการทำงานมากอาจจะถูกส่งไปให้กับเซิร์ฟเวอร์เครื่องที่อยู่ภายในระบบว่าเครื่องอื่นได้ การวนรอบแบบถ่วงน้ำหนัก (Weighted Round Robin) เพื่อให้เครื่องกระจายงานในปริมาณที่เหมาะสมกับขีดความสามารถของเครื่องแต่ละเครื่องในระบบ เป็นต้น ในส่วนนี้ถ้าเราออกแบบหรือใช้อัลกอริทึมที่เหมาะกับประเภทของงาน จะทำให้เราสามารถใช้ประสิทธิภาพของระบบได้เพิ่มมากขึ้น

2.1.3.3 แบบ Fast Response

Fast Response การกระจายโหลดแบบนี้จะพิจารณาถึง Response Time ของระบบว่า Real Server ตัวใดมีเวลาในการตอบสนองที่เร็วที่สุดโหลดบาลานซ์เซิร์ฟเวอร์ก็จะทำการกระจายโหลดไปให้ Real Server ตัวนั้นก่อนที่จะกระจายโหลดไปให้ตัวต่อๆไปที่มี Response Time มากกว่า (Response Time ที่มีค่าน้อยแสดงว่าใช้เวลาในการตอบสนองต่อระบบได้เร็ว) ซึ่งจากการกระจายโหลดแบบ Fast Response จะให้ระบบสามารถกระจายโหลดได้อย่างมีประสิทธิภาพมากขึ้น ทำให้เกิดความรวดเร็วในการส่งข้อมูล โดยการกระจายโหลดแบบนี้ เงื่อนไขที่พิจารณาได้ในระบบคือพิจารณาได้จากการทำ Polling ที่ได้ทำไปก่อนหน้านี้ ซึ่งการทำ Polling ได้แสดงถึง Response Time ของ Real Server แต่ละตัวทำให้การกระจายโหลดแบบ Fast Response นำเงื่อนไขนี้ไปใช้ได้

2.1.3.4 แบบ Least Connection

การเชื่อมต่อน้อยที่สุด (Least Connection) การกระจายโหลดแบบนี้พิจารณาถึงจำนวนภาระงานที่ Real Server แต่ละตัวได้รับซึ่งในการกระจายโหลดของโหลดบาลานซ์เซิร์ฟเวอร์จะกระจายโหลดไปให้ Real Server ตัวที่มีภาระงานน้อยที่สุดก่อน ส่วนตัวที่รับภาระงานเยอะก็หยุดกระจายโหลดชั่วคราวซึ่งโหลดบาลานซ์เซิร์ฟเวอร์จะพยายามกระจายโหลดไปให้แก่ Real Server ในอัตราส่วนที่เหมาะสมตามประสิทธิภาพเครื่องของ Real Server แต่ละตัว (ในกรณีที่ Real Server แต่ละตัวมีประสิทธิภาพในการทำงานต่างกัน) ในการกระจายโหลดแบบนี้จะทำให้การกระจายโหลดมีความสมดุลไม่ต้องรอว่า Real Server ตัวใดตัวหนึ่งทำงานหนักอยู่แต่จะกระจายโหลดไปให้ตัวที่ทำงานน้อยๆ (Real Server ประสิทธิภาพสูงกว่า) ทำแทนซึ่งเป็นการบริหารประสิทธิภาพของเครื่อง Real Server แต่ละเครื่องให้ทำงานกับระบบได้สมดุลทำให้การส่งข้อมูลมีความเสถียรและรวดเร็วขึ้น ซึ่งจะสังเกตว่าการกระจายโหลดแบบนี้จะมีความคล้ายคลึงกับวิธีการกระจายโหลดแบบ Ratio (Weight Round Robin) แต่แบบ Ratio จะเป็นแบบที่ผู้ดูแลระบบทำการประเมินประสิทธิภาพเองและเลือกอัตราส่วนการกระจายโหลดเอง แต่การกระจายโหลดแบบ Least Connection ระบบจะทำการวิเคราะห์ระบบเองและเลือกอัตราส่วนในการกระจายโหลดเองซึ่งจุดประสงค์ในการใช้งานของแต่ละระบบก็มีความแตกต่างกันไปขึ้นอยู่กับว่าการกระจายโหลดแบบไหนที่เหมาะกับระบบมากกว่าและให้ผลลัพธ์ที่ต้องการ

2.1.3.5 แบบ Policy Routing

Policy Routing เป็นเทคนิคที่ใช้เพื่อการกำหนดเส้นทางและการตัดสินใจในการส่งข้อมูลหรือโหลดจากต้นทางไปยังปลายทางโดยผู้ดูแลระบบเป็นผู้กำหนด Policy Routing โดยทั่วไปแล้วการหาเส้นทางในการส่งข้อมูลของอุปกรณ์ Layer 3 นั้น จะใช้ปลายทางของ ทราฟฟิคนั้น เป็นตัวตัดสินใจในการเลือกเส้นทางว่าจะส่งไปในเส้นทางใด เช่น ต้องการส่งข้อมูลไปยัง Server ที่อยู่ต่างประเทศ เมื่อทราฟฟิคนั้นเดินทางมาถึงเราเตอร์ เราเตอร์ก็จะดูว่าปลายทางของทราฟฟิคนั้นสามารถส่งไปยังเส้นทางใดบนตัวมันได้บ้าง จากนั้นมันจึงจะทำการส่งทราฟฟิคไปยังเส้นทางที่ดีที่สุดเพื่อไปยังปลายทางที่ต้องการ แต่ด้วยการใช้งาน Policy Routing เราจะสามารถใช้ต้นทาง (source) หรือชนิดของทราฟฟิค ในการเลือกเส้นทางที่จะใช้ในการส่งข้อมูลได้ เช่น เราสามารถเลือกได้ว่าให้ทราฟฟิคที่มีต้นทางเป็น 192.168.1.10 ถึง 192.168.1.20 ให้ไปใช้งานเส้นทางหนึ่ง ส่วนทราฟฟิคชนิดอื่น ๆ ก็ให้ใช้งานอีกเส้นทางหนึ่งได้

2.1.3.6 แบบ Link Backup

Link Backup เป็นรูปแบบการใช้งานอินเทอร์เน็ตโดยลิงค์สำรองจะทำงานต่อเมื่อลิงค์หลักไม่สามารถใช้งานได้ โดยสามารถกำหนดแบนด์วิดท์หรือความเร็วในการใช้งานอินเทอร์เน็ต (Internet Bandwidth) และรูปแบบการใช้งาน (Internet Application) ให้กับกลุ่มของผู้ใช้งานที่อยู่ในระบบเครือข่ายได้ เช่น กำหนดให้การใช้งานเว็บดาวน์โหลดข้อมูล ใช้งานที่ลิงค์ที่ WAN1 เท่านั้น และ กำหนดให้การใช้งานรับ-ส่งอีเมลล์ ใช้งานที่ลิงค์ที่ WAN2 เท่านั้นหรือสามารถกำหนดความเร็วที่ต้องการให้ใช้งานได้ เช่น กำหนดให้ผู้ใช้งานไอพีแอดเดรสในช่วง 192.168.1.10-192.168.1.20 ใช้งานเว็บที่ลิงค์ที่ 1 โดยสามารถใช้ความเร็วได้ในช่วง 512-1024 Kbps และผู้ใช้งานไอพีแอดเดรสในช่วง 192.168.1.21-192.168.1.100 ใช้งานเว็บที่ลิงค์ที่ 2 โดยสามารถใช้ความเร็วได้ในช่วง 512-2048 Kbps เป็นต้น

2.2 IPVSADM

IPVSADM คือ เซอร์วิสที่ใช้สร้างเครื่องแม่ข่ายเสมือนเพื่อใช้ในการตั้งค่าเครื่องแม่ข่ายเสมือนในระบบปฏิบัติการลีนุกซ์ และสร้างเครื่องข่ายแม่เสมือนเพื่อให้บริการแก่ Real Server ตั้งแต่ 2 เครื่องขึ้นไป ซึ่ง IPVSADM ให้การรองรับเซอร์วิสของ TCP และ UDP และสนับสนุนวิธีการส่งต่อแพ็กเก็ต 3 วิธี คือ NAT, Tunneling และ Direct routing และรองรับวิธีการการกระจายภาระงานทั้งแบบ Round robin, Weighted Round Robin, Fast Response และ Least Connection ด้วย

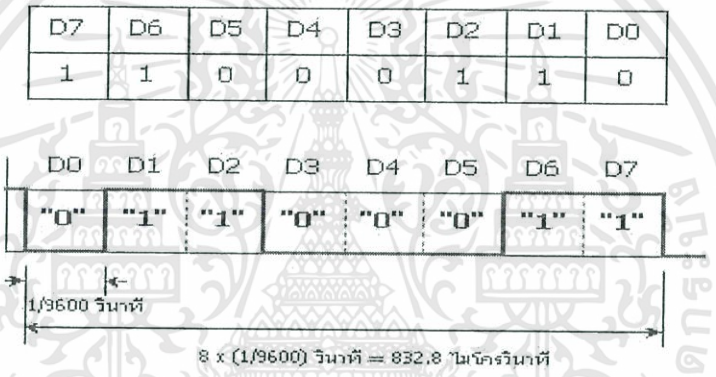
2.3 การสื่อสารพอร์ตอนุกรม

ข้อมูลในไมโครคอนโทรลเลอร์ที่ใช้ศึกษาจะเป็นข้อมูลที่มีความยาวขนาด 1 ไบต์ หรือ 8 บิต โดยมีวิธีการส่งข้อมูลเป็นแบบขนาน คือเป็นการส่งข้อมูลขนาด 8 บิตพร้อมกันไปยังอุปกรณ์ภายนอก และจะต้องมีจำนวนของสายสัญญาณจำนวน 8 เส้น เพื่อให้พอดีกับจำนวนของบิตที่ต้องการจะส่ง การส่งข้อมูลแบบขนานจึงทำให้มีการส่งข้อมูลที่มีความรวดเร็ว แต่ถ้าหากมีการสื่อสารข้อมูลในระยะไกล ก็จะต้องใช้จำนวนของสายและระยะทางของสายมากขึ้นจึงทำให้มีการสิ้นเปลืองค่าใช้จ่ายสูง

ดังนั้นการสื่อสารข้อมูลแบบอนุกรมจึงถูกนำมาใช้ ในการสื่อสาร โดยจะใช้สายเพียงเส้นเดียวในการส่งข้อมูล หรือรับข้อมูล (คำว่าเส้นเดียวหมายความว่าสายส่ง (TxD) 1 เส้น สายรับ (RxD) 1 เส้น และสายกราวด์ร่วม (Ground 1 เส้น) นำมาใช้สื่อสารข้อมูลกับอุปกรณ์ภายนอกในระยะทางที่ไกล ถ้าหากต้องการส่งข้อมูลขนาด 8 บิต ก็จะทำการส่งข้อมูลออกไปทีละบิตเป็นลำดับไป จนกว่าจะครบจำนวนทั้ง 8 บิต โดยข้อมูลจะถูกส่งไปตามสายสัญญาณที่ละบิตตามจังหวะเวลาที่กำหนด เป็นความกว้างของพัลส์ โดยจังหวะเวลาที่กล่าวนี้จะต้องมีมาตรฐานของฝ่ายส่งและฝ่ายรับด้วย ในการรับสัญญาณที่ส่งมาทีละบิต จะทำการตรวจสอบระดับแรงดันของสัญญาณที่เข้ามาเพื่อแปลงเป็นลอจิก "1" หรือ "0" เมื่อรับข้อมูลเข้ามาครบใน 1 ไบต์ที่กำหนดไว้ก็จะถูกเปลี่ยนให้อยู่ในรูปแบบของข้อมูลแบบขนานเหมือนเดิม

2.3.1 จังหวะเวลาของการสื่อสารข้อมูลอนุกรม

ในการสื่อสารข้อมูลแบบอนุกรม เพื่อรับหรือส่งข้อมูลจะเป็นลักษณะของกลุ่มข้อมูล ดังนั้น อัตราความเร็วจะต้องมีค่าเท่ากันระหว่างการรับและการส่งโดยทั่วไปเราจะระบุความเร็วของจำนวนบิตในการรับและส่งข้อมูล เป็นจำนวนของบิตที่จะส่งใน 1 วินาที โดยเรียกความเร็วในการส่งข้อมูลว่า อัตราบอด (Baud Rate) ซึ่งมีหน่วยเป็นสัญลักษณ์ต่อวินาที (Bd) เช่น 300, 1,200, 2,400, 4,800 และ 9,600 สัญลักษณ์ต่อวินาที ในรูปที่ 2.1 ถ้าหากมีการส่งข้อมูลด้วยความเร็ว 9600 สัญลักษณ์ต่อวินาที จะใช้เวลาในการรับส่งข้อมูลหนึ่งบิตมีค่าเท่ากับ 1/9600 หรือ 104.1 ไมโครวินาที และเวลาในการรับส่งข้อมูลทั้ง 8 บิตจะมีค่าเท่ากับ 8 x 104.1 หรือ 832.8 ไมโครวินาที



รูปที่ 2.1 การส่งข้อมูลแบบอนุกรมด้วยความเร็ว 9600 สัญลักษณ์ต่อวินาที

2.3.2 รูปแบบของการสื่อสารข้อมูลอนุกรม

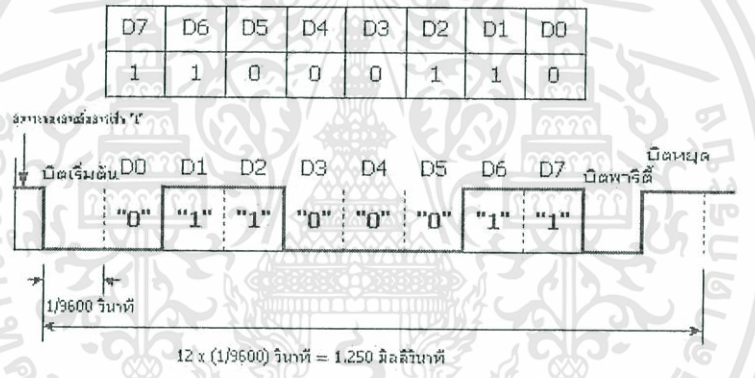
การสื่อสารข้อมูลอนุกรมแบบอะซิงโคนัส เป็นวิธีการรับและส่งข้อมูลโดยไม่ต้องอาศัยสัญญาณนาฬิกาส่งร่วมไปด้วย แต่จะใช้อัตราความเร็วของจำนวนข้อมูลต่อวินาที และจะทำการเพิ่มบิตข้อมูลบางอย่างร่วมไปกับการส่งข้อมูลจริง เพื่อจะได้ทำการตรวจสอบข้อมูลได้อย่างถูกต้องมากยิ่งขึ้นแสดงดังรูปที่ 2.2 ซึ่งประกอบด้วยกัน 4 ส่วนคือ

1. บิตเริ่มต้น (Start bit) จะมีขนาด 1 บิต จะเป็นระดับลอจิกตรงกันข้ามกับระดับลอจิกของสภาวะสายสื่อสาร ขณะที่ยังไม่มีการส่งข้อมูล

2. บิตข้อมูล (Data bit) จะเริ่มจากบิตที่มีนัยสำคัญต่ำสุดก่อนหรือ บิต LSB ก่อน โดยข้อมูลที่จะส่งอาจจะมีขนาด 5 ,6, 7 หรือ 8 บิตก็ได้

3. บิตแสดงสภาวะเลขคู่หรือเลขคี่ (Parity bit) มีขนาด 1 บิตโดยบิตนี้จะนำไปต่อท้ายกับบิตข้อมูล ค่าของบิตนี้ขึ้นอยู่กับจำนวนค่าของข้อมูลที่เป็น "1" โดยเลือกการส่งข้อมูลเป็นแบบพาริตี้คู่ หรือ พาริตี้คี่ ตัวอย่าง ถ้ากำหนดให้มีการส่งข้อมูลแบบพาริตี้คู่ แต่ข้อมูลมีเลข 1 เป็นจำนวนคี่ ก็จะให้บิตพาริตี้นี้เป็น "1"เพื่อจะได้จำนวนเลข "1" เป็นคู่นั้นเอง ทำนองเดียวกันทางด้านรับเองก็ต้องมีการตรวจสอบจำนวนข้อมูลที่รับเข้ามาเป็น "1" รวมทั้งบิตพาริตี้ 1 บิต ถ้ามีค่า "1"เป็นจำนวนคู่แสดงว่าข้อมูลที่รับเข้ามาถูกต้อง หรือสามารถกำหนดการรับและส่งข้อมูลเป็นแบบ NONE โดยไม่ต้องมีการตรวจสอบพาริตี้บิตก็ได้

4. บิตสุดท้ายหรือบิตหยุด (Stop bit) เป็นการระบุถึงขอบเขตของการสิ้นสุดข้อมูล โดยจะทำให้ขาข้อมูลมีสถานะ ลอจิกเป็น "1" ซึ่งอาจมีจำนวนมากกว่า หนึ่งบิตก็ได้ เช่น 1 บิต 1.5 บิต หรือ 2 บิต

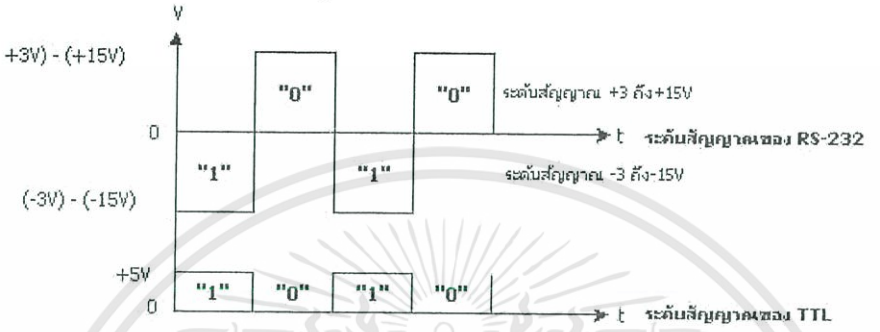


รูปที่ 2.2 การส่งข้อมูลขนาด 8 บิตแบบอนุกรมพร้อมด้วย บิตเริ่มต้น บิตพาริตี้ บิตหยุด ด้วยความเร็ว 9600 บิตต่อวินาที

2.3.3 การเชื่อมต่อพอร์ตอนุกรมมาตรฐาน RS-232

การกำหนดมาตรฐานการเชื่อมต่อแบบอนุกรม EIA RS-232 (x) เป็นมาตรฐานอุตสาหกรรมโดยคณะกรรมการสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association) ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบ อะซิงโครนัส 2 ทิศทาง เพื่อให้มีการใช้งานในการเชื่อมต่อที่สอดคล้องกัน ระหว่างอุปกรณ์คอมพิวเตอร์ต่างๆ การรับส่งสัญญาณจะกำหนดความยาวสูงสุดไว้ที่ไม่เกิน 50 ฟุตโดยมีระดับ สัญญาณตั้งแต่ 3 โวลท์ จนถึง 15 โวลท์ สำหรับลอจิก "0" และมีระดับแรงดันที่ -3 โวลท์ จนถึง -15 โวลท์ สำหรับลอจิก "1" ดังแสดงในรูป 2.3

ดังนั้นสังเกตได้ว่าจะมีระดับแรงดันที่ใช้ในสถานะลอจิก "0" และ ลอจิก "1" แตกต่างออกไปจากระบบไอซีดิจิตอลทั่วๆไป การต่อใช้งานจึงต้องมีอุปกรณ์ที่ทำหน้าที่เปลี่ยนระดับแรงดันจาก 0 - 5 โวลท์ จากไมโครคอนโทรลเลอร์ ให้เป็นระดับแรงดันที่สูงกว่า +3 หรือต่ำกว่า - 3 โดยจะมีไอซีสำเร็จรูปพร้อมใช้งาน หรืออาจจะต่อวงจรจากทรานซิสเตอร์ได้



รูปที่ 2.3 ระดับแรงดันสัญญาณของพอร์ตอนุกรม RS-232 กับ TTL ในสถานะลอจิก "1" และ "0"

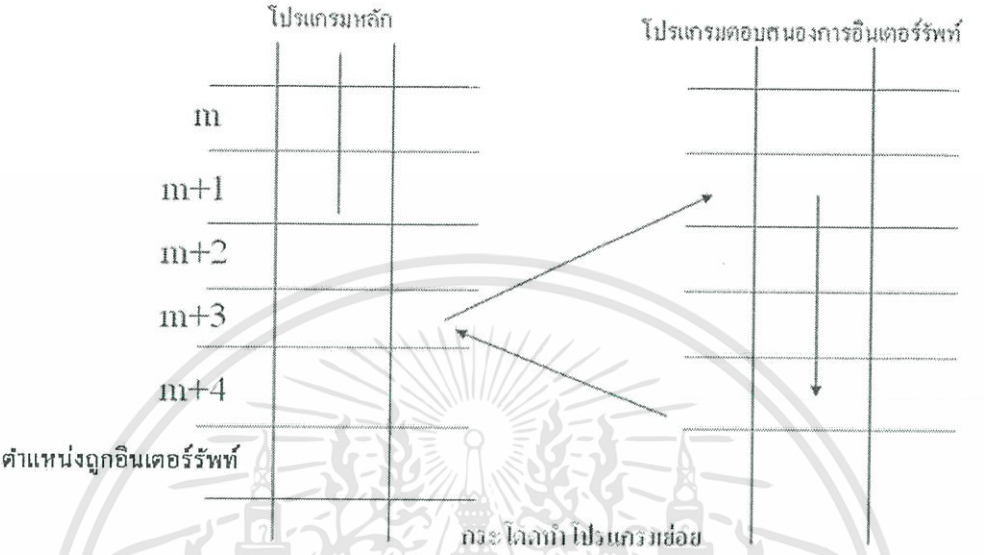
2.4 การอินเตอร์รัพท์

การอินเตอร์รัพท์ คือ การขัดการทำงานของโปรแกรมหลัก โดยให้โปรแกรมหลักหยุดชั่วคราวแล้วกระโดดไปทำงานในส่วนโปรแกรมบริการอินเตอร์รัพท์ เมื่อทำงานในโปรแกรมบริการเสร็จเรียบร้อยโปรแกรมจะกลับไปทำงานโปรแกรมหลักต่อไป เมื่อเกิดการอินเตอร์รัพท์เวลาใดๆ จะทำให้การทำงานของโปรแกรมหลักหยุดไป แล้วกระโดดไปทำงานของโปรแกรมบริการอินเตอร์รัพท์ (ISR)

2.4.1 ขบวนการเกิดอินเตอร์รัพท์

ถ้า CPU กำลังทำงานโปรแกรมหลักอยู่ เช่นกำลังทำคำสั่งในตำแหน่งของหน่วยความจำที่ m, m+1, m+2 ไปเรื่อยๆโดย PC จะชี้ที่ตำแหน่งที่อ่านค่าคำสั่งถัดมา เมื่อโปรแกรมทำงานมาถึงตำแหน่งที่ m+3 แล้วเกิดอินเตอร์รัพท์ขึ้น (ขณะนั้น PC อยู่ที่ m+4) โปรแกรมจะต้องทำงานโปรแกรมตอบสนองการอินเตอร์รัพท์ โดยย้าย PC ไปที่ตำแหน่งที่เก็บโปรแกรมตอบสนองอินเตอร์รัพท์ จากนั้นจะเก็บค่า PC เดิมลงในหน่วยความจำสแตก เมื่อคอมพิวเตอร์ทำงานโปรแกรม

ตอบสนองการอินเตอร์รัพท์เสร็จสิ้นลงจะคืนค่าในสแตก (m+4) ให้กับ PC เพื่อให้ PC ทำโปรแกรมหลักต่อไป



รูปที่ 2.4 ขั้นตอนการทำงานของโปรแกรมเมื่อถูกอินเตอร์รัพท์

2.4.2 สัญญาณอินเตอร์รัพท์

แหล่งกำเนิดสัญญาณอินเตอร์รัพท์ที่ใช้กับ MCS-51 มีสองชนิดคือ

1. อินเตอร์รัพท์ภายใน

อินเตอร์รัพท์ภายในจะเกิดขึ้นจากภายในตัว MCS-51 เอง ได้แก่ สัญญาณจาก

- ไทม์เมอร์แฟลก 0 (TIMER 0)
- ไทม์เมอร์แฟลก 1 (TIMER 1)
- พอร์ทอนุกรม (SERIAL)

2. อินเตอร์รัพท์ภายนอก

อินเตอร์รัพท์ภายนอกเกิดจากสัญญาณที่กระตุ้นเข้ามาทางขา

- INT0
- INT1

2.4.3 รีจีสเตอร์สำหรับโปรแกรมอินเตอร์รัพท์

การใช้งานโปรแกรมอินเตอร์รัพท์ในตัวไมโครคอนโทรลเลอร์ MSC-51 จะมีรีจีสเตอร์สำหรับการทำงานอินเตอร์รัพท์อยู่ 3 ตัว คือ

1. IE (Interrupt Enable)
2. IP (Interrupt Priority)
3. TCON (Timer Control)

2.4.3.1. IE (Interrupt Enable)

เป็นรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ ใช้สำหรับการกำหนดค่าว่าถ้าเกิดการอินเตอร์รัพท์จากแหล่งต่างๆ จะทำการอินเตอร์รัพท์เหล่านี้หรือไม่ โดยมีรายละเอียดต่างๆ ของบิตดังนี้

ตารางที่ 2.1 รายละเอียดของ IE (Interrupt Enable)

บิต	สัญลักษณ์	หน้าที่
IE.7	EA	บิต EA=1 ยอมให้การอินเตอร์รัพท์เกิดขึ้นกับบิตควบคุมของอินเตอร์รัพท์นั้นๆ
IE.6	-	ไม่ใช้
IE.5	ET2	บิต ET2 = 1 ยอมให้มีการอินเตอร์รัพท์จาก TIMER2
IE.4	ES	บิต ES = 1 ยอมให้มีการอินเตอร์รัพท์จากพอร์ตอนุกรม
IE.3	ET1	บิต ET1 = 1 ยอมให้มีการอินเตอร์รัพท์จาก TIMER1
IE.2	EX1	บิต EX1 = 1 ยอมให้มีการอินเตอร์รัพท์จากภายนอกผ่านขา INT1
IE.1	ET0	บิต ET0 = 1 ยอมให้มีการอินเตอร์รัพท์จาก TIMER0
IE.0	EX0	บิต EX0 = 1 ยอมให้มีการอินเตอร์รัพท์จากภายนอกผ่านขา INT0

2.4.3.2. IP (Interrupt Priority)

เป็นรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ ใช้ในการจัด ลำดับความสำคัญของการอินเตอร์รัพท์ ซึ่งสามารถจัดได้ 2 ลำดับคือ

1. ถ้าเป็น “1” หมายความว่ามีลำดับความสำคัญสูงสุด
2. ถ้าเป็น “0” หมายความว่ามีลำดับความสำคัญต่ำสุด

ตารางที่ 2.2 รายละเอียดของ IP (Interrupt Priority)

บิต	สัญลักษณ์	หน้าที่
IE.7	-	ไม่ใช้
IE.6	-	ไม่ใช้
IE.5	PT2	ลำดับความสำคัญของการอินเตอร์รัพท์จาก TIME2
IE.4	PS	ลำดับความสำคัญของการอินเตอร์รัพท์จากพอร์ทอนุกรม
IE.3	PT1	ลำดับความสำคัญของการอินเตอร์รัพท์จาก TIME1
IE.2	PX1	ลำดับความสำคัญของการอินเตอร์รัพท์จาก INT1
IE.1	PT0	ลำดับความสำคัญของการอินเตอร์รัพท์จาก TIME0
IE.0	PX0	ลำดับความสำคัญของการอินเตอร์รัพท์จาก INT0

2.4.3.3 TCON (Timer Control)

เป็นรีจิสเตอร์ขนาด 8 บิต สามารถอ้างตำแหน่งแบบบิตได้ ใช้ในการควบคุมการทำงานของไทม์เมอร์ และควบคุมการทำงานของอินเตอร์รัพท์

ตารางที่ 2.3 รายละเอียดของ TCON (Timer Control)

แหล่งการเกิดอินเตอร์รัพท์	ตำแหน่งแอดเดรส	หมายเลขอินเตอร์รัพท์
IE0	0003H	0
TF0	000BH	1
IE11	0013H	2
TF1	001BH	3
SERIAL	0023H	4

2.4.4 การทำงานของระบบหลังถูกอินเตอร์รัพท์

เมื่อ MCS-51 ถูกอินเตอร์รัพท์จะต้องกระโดดไปทำโปรแกรมตอบสนองการอินเตอร์รัพท์ โดยตำแหน่งที่กระโดดไปเรียกว่า"อินเตอร์รัพท์เวกเตอร์" (Interrupt Vectors) เมื่อทำการอินเตอร์รัพท์เสร็จเรียบร้อย MCS-51 จะกระโดดมาทำงานในตำแหน่งเดิม โดยก่อนที่จะกระโดดไปทำโปรแกรมตอบสนองการอินเตอร์รัพท์ จะต้องเก็บค่าในตำแหน่งเดิมไว้ โดยเก็บค่า PC ลงหน่วยความจำสแตก ซึ่งอยู่ที่หน่วยความจำที่ถูกชี้โดยรีจิสเตอร์ SP เมื่อทำการตอบสนอง

อินเตอร์รัพท์เสร็จแล้วจะคืนค่าในหน่วยความจำสแตกให้ PC ตามเดิม ค่า MCS-51 แสดงได้ดังตารางต่อไปนี้

ตารางที่ 2.4 อินเตอร์รัพท์เวกเตอร์ของอินเตอร์รัพท์ต่างๆ

อินเตอร์รัพท์	อินเตอร์รัพท์เวกเตอร์
System Reset	0000H
External 0	0003H
Timer 0	000BH
External 1	0013H
Timer 1	001BH
Serial Port	0023H
Timer 2	002BH

2.4.5 การออกแบบโปรแกรมอินเตอร์รัพท์

ในการเขียนโปรแกรมหลัก (Main Program) จะต้องกำหนดค่า MCS-51 ถูกอินเตอร์รัพท์ด้วยอะไร และจะให้ MCS-51 ถูกอินเตอร์รัพท์ได้หรือไม่ โดยการโปรแกรมค่าต่างๆ ใน IE รีจิสเตอร์ ถ้ามีการอินเตอร์รัพท์จากสองแหล่งขึ้นไปควรมีการจัดลำดับความสำคัญในรีจิสเตอร์ IP ดังนั้นโปรแกรมหลักจะต้องมีโปรแกรมต่อไปนี้

1. โปรแกรมค่ารีจิสเตอร์ IE
2. โปรแกรมค่ารีจิสเตอร์ IP

2.4.5.1. โปรแกรมตอบสนองการอินเตอร์รัพท์แบบสั้น

จากตารางอินเตอร์รัพท์เวกเตอร์ จะเห็นว่าที่เก็บโปรแกรมอินเตอร์รัพท์แต่ละแหล่งจะห่างกัน 8 ไบต์ ดังนั้นถ้ามีการอินเตอร์รัพท์จากแหล่งต่างๆ หลายๆ แหล่งและโปรแกรมตอบสนองการอินเตอร์รัพท์บางโปรแกรมมีขนาดยาวเกิน 8 ไบต์จะทำให้โปรแกรมทับกับตำแหน่งของโปรแกรมตอบสนองการอินเตอร์รัพท์ของอินเตอร์รัพท์ถัดไป แต่ถ้าโปรแกรมตอบสนองการอินเตอร์รัพท์ไม่ยาวเกินไปเราสามารถเขียนไปในตำแหน่งนั้นได้เลยดังโปรแกรมต่อไปนี้

```
        ORG     0000H
        LJUMP   MAIN   ; กระโดดไปโปรแกรมหลัก
        ORG     000BH ; ตำแหน่งเริ่มต้นของอินเตอร์รัพท์ Timer 0
TOISR : .....
        .....
        RETI           ; กลับโปรแกรมหลัก
MAIN :  .....          ; โปรแกรมหลัก
```

2.4.5.2. โปรแกรมตอบสนองการอินเตอร์รัพท์ขนาดใหญ่

ในกรณีที่มีการอินเตอร์รัพท์จากหลายแหล่ง และโปรแกรมตอบสนองการอินเตอร์รัพท์แต่ละโปรแกรมยาวเกิน 8 ไบต์ เราไม่สามารถเขียนโปรแกรมตอบสนองการอินเตอร์รัพท์ไว้ที่ตำแหน่งของอินเตอร์รัพท์เวกเตอร์ได้ ซึ่งจะแก้ปัญหานี้ได้โดยกำหนดให้ตำแหน่งของอินเตอร์รัพท์เวกเตอร์ให้ทำโปรแกรมกระโดด โดยกระโดดไปที่ตำแหน่งเก็บโปรแกรมตอบสนองการอินเตอร์รัพท์ที่เขียนไว้ที่ตำแหน่งอื่นดังตัวอย่างต่อไปนี้

```
        ORG    0000H  ; เริ่มโปรแกรมของระบบ
        LJMP   MAIN   ; กระโดดไปโปรแกรมหลัก
        ORG    000BH  ; ตำแหน่งของอินเตอร์รัพท์ Timer 0
        LJMP   LED1   ; กระโดดไปโปรแกรมตอบสนองอินเตอร์รัพท์ชื่อ LED1
        ORG    0030H  ; ตำแหน่งหลังอินเตอร์รัพท์เวกเตอร์
MAIN :  .....         ; โปรแกรมหลัก
        .....
LED :   .....         ; โปรแกรมตอบสนองการอินเตอร์รัพท์ Time 1
        .....
        RETI          ; กลับสู่โปแกรมหลัก
```

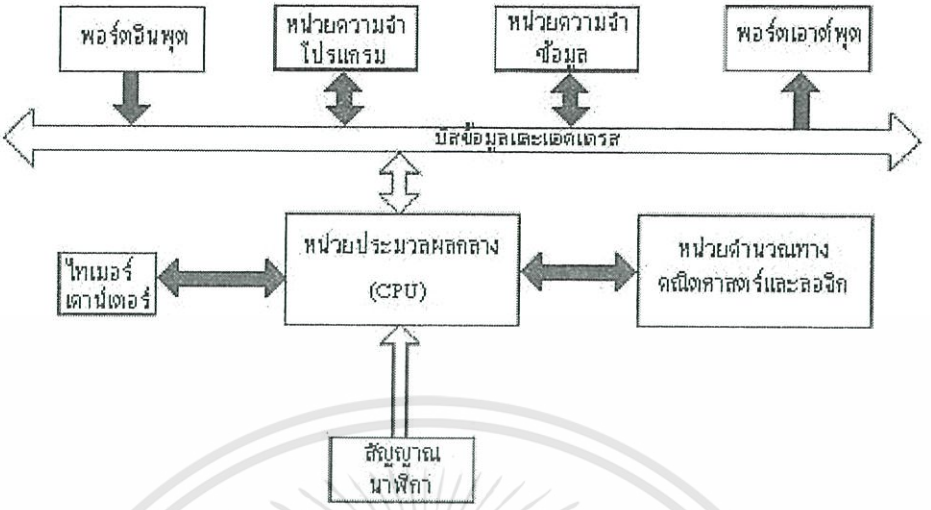
2.5 ไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิตที่มีอุปกรณ์สนับสนุนประกอบอยู่ภายในหลายอย่างได้แก่ หน่วยความจำสำหรับเก็บข้อมูลหน่วยความจำสำหรับเก็บโปรแกรม ตัวตั้งเวลา/ตัวนับ อุปกรณ์รับส่งข้อมูลแบบอนุกรม เนื่องจากโครงสร้างของไมโครคอนโทรลเลอร์มีอุปกรณ์สนับสนุนประกอบอยู่ภายในนี้เอง ทำให้การใช้งานง่ายขึ้นและมีประสิทธิภาพมากขึ้นโดยไม่ต้องมีการเชื่อมต่ออุปกรณ์ภายนอกเพิ่มเติมมากเหมือนกับ ตัวไมโครโปรเซสเซอร์ทั่วไปนอกจากนี้หากเราต้องการใช้งานไมโครคอนโทรลเลอร์ร่วมกับอุปกรณ์อื่นเพิ่มเติม เช่น ไอซี 8255 หรือหน่วยความจำภายนอกยังสามารถนำมาเชื่อมต่อเพิ่มเติมเข้ากับไมโครคอนโทรลเลอร์ได้อีกด้วย

2.5.1 โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

ประกอบด้วย

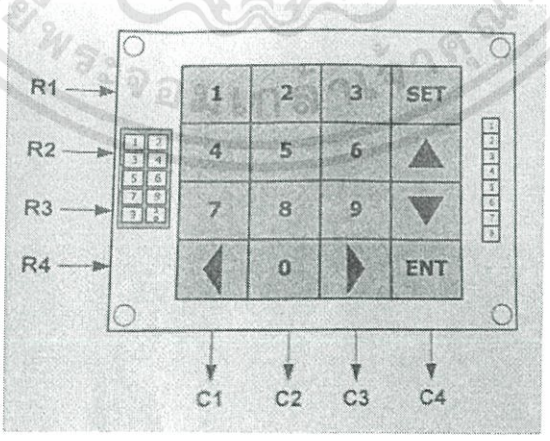
- หน่วยประมวลผลกลางขนาด 8 บิต
- หน่วยประมวลผลสำหรับข้อมูลแบบบิต (BOOLEAN PROCRSSOR)
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำโปรแกรม 64 กิโลไบต์
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำข้อมูล 64 กิโลไบต์
- หน่วยความจำโปรแกรมภายในขนาด 4 กิโลไบต์ แบบ อีพรอม (เบอร์ 8451)
- หน่วยความจำแบบ แรม ภายในจำนวน 128 ไบต์
- พอร์ตอินพุต/เอาต์พุตแบบขนานจำนวน 32 เส้น ซึ่งสามารถแยกทำงานได้อย่างอิสระ
- วงจรนับ/จับเวลาขนาด 16 บิต จำนวนสองวงจร
- วงจรสื่อสารแบบอนุกรมแบบดูเพล็กเต็ม (FULL DUPLEX)
- วงจรควบคุมการอินเตอร์รัพท์จากแหล่งกำเนิดสัญญาณ 6 ประเภท พร้อมการกำหนดลำดับ
- วงจรผลิตสัญญาณนาฬิกาภายในซึ่งโครงสร้างการทำงานทั้งหมดของไมโครคอนโทรลเลอร์จะอาศัยหลักการทำงานที่เกี่ยวข้องกัน โดยอาศัยหลักการทำงานที่เป็นไปตามโครงสร้างเสมอ



รูปที่ 2.5 โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

2.6 คีย์แพด (Keypad)

คีย์แพดเป็นอุปกรณ์สำหรับรับอินพุตจากผู้ใช้ มีลักษณะเป็นปุ่มกดหลายปุ่ม ถูกจัดเรียงกันในลักษณะเป็นอาร์เรย์ แบ่งเป็นแถวแนวนอน (Rows) และแถวแนวตั้ง (Columns) เช่น 3x4 (= 12 ปุ่ม) หรือ 4x4 (= 16 ปุ่ม) เป็นต้น แต่ละปุ่มก็จะมีสัญลักษณ์เขียนกำกับไว้ เช่น ตัวเลข 0-9, #, * เป็นต้น โดยปรกติถ้าต่อปุ่มกดแยกจำนวน 16 ตัว จะต้องใช้ขาสัญญาณทั้งหมด 16 ขา แต่ถ้าใช้การจัดเรียงแบบ 4x4 จะใช้ขาสัญญาณเพียง 8 ขา แต่ต้องมีเทคนิคในการตรวจดูว่าปุ่มกดใดถูกกดบ้างในขณะนั้น วิธีการนี้เรียกว่า การสแกนปุ่มกด (key scan)



รูปที่ 2.6 การแบ่งแถวแนวนอน (Rows) และแถวแนวตั้ง (Columns) ของคีย์แพด

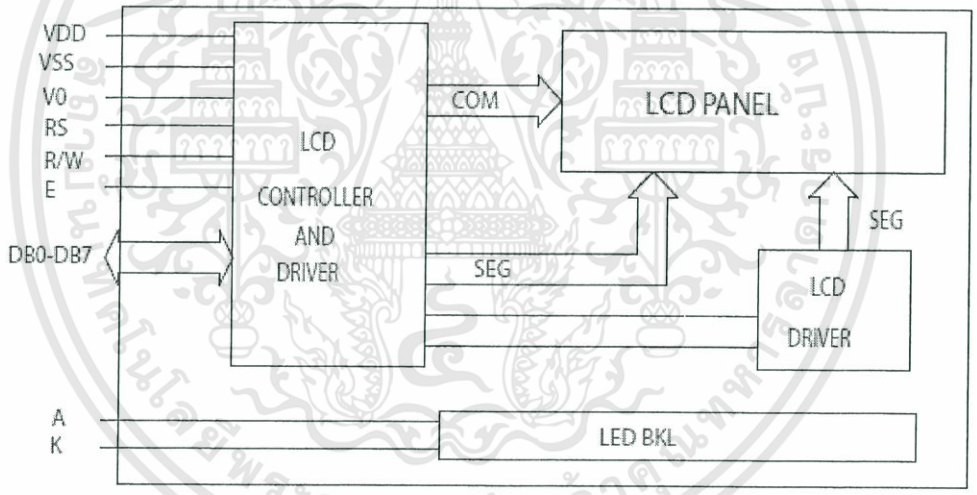
2.7 แอลซีดีโมดูล (LCD Module)

2.7.1 ส่วนประกอบของแอลซีดีโมดูล (LCD Module)

- ตัวแสดงผล (DISPLAY) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอแอลซีดี

- ตัวควบคุม (CONTROLLER) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูลแอลซีดีเช่น จอลบภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะชิปที่นิยมใช้ก็คือเบอร์ HD44780 และ HD61830 โดย HD44780 จะใช้ควบคุมแอลซีดีแบบอักขระ ส่วน HD61830ใช้ควบคุมแอลซีดีแบบกราฟิก

- ตัวขับ (DRIVER) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ HD4410H และ MSM5259 เป็นต้น



รูปที่ 2.7 บล็อกไดอะแกรมแสดงส่วนประกอบของแอลซีดีโมดูล (LCD Module)

2.8 ภาษา JAVA

ภาษา Java เป็นภาษาสำหรับเขียนโปรแกรมที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุ (OOP : Object-Oriented Programming) ซึ่งเป็นเทคนิคที่ใช้วัตถุ (Object) เป็นหลักในการพิจารณาสิ่งต่างๆ ที่สนใจ ดังนั้นโปรแกรมที่เขียนด้วยภาษาจาวาจึงประกอบไปด้วยกลุ่มของ Object โดยแต่ละ

Object จะถูกจัดกลุ่มอยู่ในรูปของ Class โดยที่แต่ละคลาสสามารถมีการถ่ายทอดลักษณะ (Inheritance) กันลงมาอย่างเป็นลำดับ (Hierarchy) ภาษา Java ได้รับความนิยมอย่างสูงเนื่องจากโปรแกรมคอมพิวเตอร์ที่เขียนด้วยภาษา Java สามารถทำงานบนเครื่องคอมพิวเตอร์ที่มีสภาพแวดล้อมต่างกันได้โดยไม่ต้อง Compile ใหม่ เป็นภาษาที่มีไวยากรณ์ที่สามารถเรียนรู้ได้ง่าย มีกลไกของภาษาที่ไม่ซับซ้อนเน้นความถูกต้องของชนิดข้อมูล (Data Type) ที่ใช้ในโปรแกรม นอกจากนี้ ด้วยเทคนิคการเขียนโปรแกรมเชิงวัตถุจึงสามารถนำบางส่วนของโปรแกรมที่ถูกเขียนไปแล้วไปใช้กับโปรแกรมที่จะถูกเขียนหรือพัฒนาขึ้นมาใหม่ได้

2.9 ระบบปฏิบัติการ Linux

ลีนุกซ์ (Linux) คือ โปรแกรมเคอร์เนล (kernel) ซึ่งทำหน้าที่เป็นศูนย์กลางของระบบปฏิบัติการ (Operating System) ซึ่งก็จะเหมือนกับระบบปฏิบัติการ อื่นๆ เช่น Windows , Unix โดยที่ Linux ได้รับการพัฒนามาจากระบบปฏิบัติการ Unix ลีนุกส์มีระบบปฏิบัติการแบบ 32 บิต มีระบบ X วินโดวส์ซึ่งเป็นระบบการติดต่อผู้ใช้แบบกราฟฟิก และเป็นระบบปฏิบัติการที่อยู่ภายใต้เงื่อนไขของGPL (GNU General Public Licence) หมายความว่า สามารถเปลี่ยนแปลงแก้ไขพัฒนาได้ และแจกจ่ายให้ใช้ฟรี

ส่วนประกอบที่สำคัญที่สุดภายในระบบปฏิบัติการก็คือตัวโปรแกรมเคอร์เนลนี้เอง โดยภายในเคอร์เนลจะมีส่วนโปรแกรมย่อยๆ เรียกว่า โมดูล รวมกันไว้ภายใน แต่ละโมดูลมีหน้าที่ และช่วยให้ระบบปฏิบัติการมีความสามารถต่างๆ จะมากหรือน้อยก็ขึ้นอยู่กับความสามารถ และจำนวนของโมดูลภายในเคอร์เนล เพราะฉะนั้นระบบปฏิบัติการทุกระบบที่นิยมใช้งานกันในปัจจุบัน จึงล้วนมีเคอร์เนลเป็นศูนย์กลางของระบบ แต่อาจมีความแตกต่างกันได้เนื่องจากมีผู้พัฒนาเคอร์เนลขึ้นหลายรายนั่นเอง ได้แก่ ระบบปฏิบัติการวินโดวส์ก็มีเคอร์เนลของตนเอง แตกต่างจากระบบปฏิบัติการอื่นๆ เช่น FreeBSD ซึ่งเป็นระบบปฏิบัติการยูนิกซ์ชนิดหนึ่งก็มีเคอร์เนลเป็นของตนเองเช่นกัน ดังนั้นหากพิจารณาภายในระบบปฏิบัติการลีนุกซ์ย่อมพบเคอร์เนลลีนุกซ์อย่างแน่นอน

2.10 Ubuntu

อูบุนตู (Ubuntu) คือ ระบบปฏิบัติการคอมพิวเตอร์ที่อยู่ใน ตระกูลหรือสายพันธุ์เดียวกันกับ Linux ที่เป็น (Open Source) ภายใต้สัญญาอนุญาตแบบ GNU/GPL สามารถนำ Linux ตัวนี้ไปใช้ ปรับปรุง เปลี่ยนแปลงได้อย่างเสรีโดยไม่มีค่าใช้จ่ายลิขสิทธิ์ Ubuntu มาจากคำในภาษาซูลู และ ภาษาโคซา ในแอฟริกาใต้ แปลว่า Humanity of Others มีความหมายในภาษาไทยคือ การช่วยเหลือกันของมวลมนุษย์ Ubuntu ถูกสร้างจาก Debian โดย Mark Shuttleworth Mark Shuttleworth เป็นคนแรกที่พัฒนา Ubuntu ขึ้นมาในปี 2004 โดย Mark Shuttleworth เกิดและโตที่แอฟริกาในปี 1995 ก่อตั้ง Thante และขายให้ VeriSign ในปี 1999 ต่อมาก็ก่อตั้ง Conoical เพื่อมาพัฒนา Ubuntu อย่างเต็มตัว ระบบปฏิบัติการตระกูล Ubuntu เป็นระบบปฏิบัติการที่ได้รับความนิยมมากที่สุดเป็นอันดับ1 ห่างจากที่ 2 คือ Open suse เป็นเท่าตัว เป็นเพราะมีการใช้งานที่ง่าย สวยขึ้นและการเข้าได้กับอุปกรณ์ต่อพวงต่างๆทำให้ Ubuntu ได้รับความนิยมเพิ่มขึ้นๆตลอดมา

2.11 NetBeans

เน็ตบีนส์ (NetBeans) เป็นเครื่องมือสำหรับนักโปรแกรมเมอร์ที่จะใช้พัฒนา Application ด้วยภาษาจาวา ในปี ค.ศ. 1998 ได้มีกลุ่มนักศึกษา "rock solid software" ได้พัฒนาซอฟต์แวร์ขึ้นมาตัวหนึ่ง ที่จะใช้ในการพัฒนา Application ด้วยภาษาจาวา เป็นโปรเจกต์นักศึกษา โดยตั้งชื่อว่า NetBeans และได้เผยแพร่ให้โปรแกรมเมอร์และบุคคลทั่วไปนำไปใช้งานได้ฟรีในรูปแบบ Opensource software ต่อมาในปี ค.ศ. 2000 บริษัทซัน ไมโครซิสเต็มส์ ผู้พัฒนาภาษาจาวา ได้เข้ามาเป็นผู้สนับสนุนหลักในการพัฒนา NetBeans และได้ทำออกมาในรูปของ Opensource software โดยผู้ใช้งานไม่จำเป็นที่จะต้องเสียเงิน เพื่อซื้อมาใช้งาน และยังได้เปิดเผย Soure code ให้ผู้สนใจและนักพัฒนานำไปดัดแปลง แก้ไข ตามกฎของ Opensource ปัจจุบันมีนักโปรแกรมเมอร์ทั่วโลกต่างช่วยกันพัฒนา NetBeans ให้มีความสามารถสูงยิ่งขึ้น

ปัจจุบัน NetBeans ได้รับความนิยมมากยิ่งขึ้น และได้รับการพัฒนาให้มีความสามารถสูงยิ่งขึ้นเรื่อยๆ จนถึงเวอร์ชั่นล่าสุด คือ นอกจากจะใช้ในการพัฒนา Application ด้วยภาษาจาวาแล้ว ยังสามารถพัฒนาอื่นๆได้อีกหลากหลายโดยติดตั้งโปรแกรมเสริม (Add-on) ได้จาก เว็บไซต์ หรือ

ผ่านตัวอัปเดตเซนเตอร์ (Update Center) ของ NetBeans เช่น ภาษาซี/ซีพลัสพลัส (C/C++), Ruby, UML, SOA, Web Application, Java EE, Mobility(Java ME), Java FX, Java Script, PHP เป็นต้น ในเวอร์ชัน 6.0 เป็นต้นไปมีการรวมโปรแกรมเสริมต่างๆที่สำคัญเข้าในตัวติดตั้งของ NetBeans โดยสามารถเลือกติดตั้งได้ภายหลัง

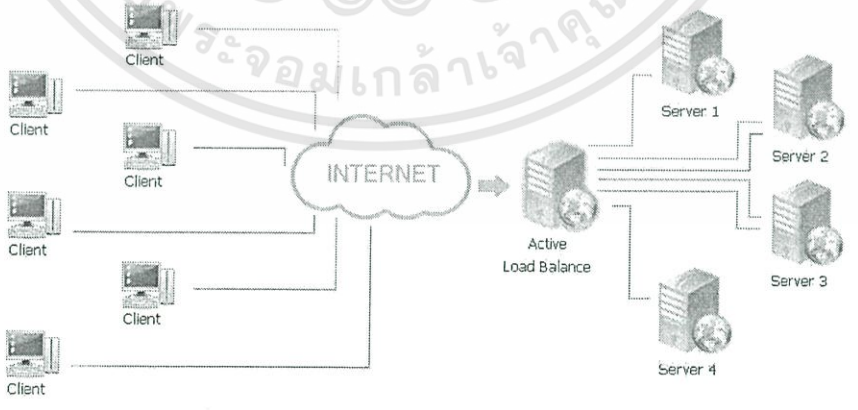
ข้อดีของโปรแกรมนี้ก็คือ โปรแกรม NetBeans นั้นทำงานแยกส่วนต่างๆ ออกจากกันเป็น Module จึงทำให้สามารถนำ Module ต่างๆที่มีผู้ที่ได้พัฒนาต่อเติมมาติดตั้งเพิ่มเติมในภายหลังได้ ใช้งานได้กับระบบปฏิบัติการ Windows , Linux, Mac OS X และ Solaris

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

3.1 การออกแบบการทำงานของระบบ

ปริญญานิพนธ์เรื่อง ระบบควบคุมการกระจายภาระงานของเครื่องแม่ข่าย เป็นปริญญานิพนธ์ที่เกี่ยวข้องกับการศึกษา พัฒนาและออกแบบระบบ เพื่อช่วยในการจัดการการใช้งานโหลดบาลานซ์เซิร์ฟเวอร์ ซึ่งผู้ศึกษาได้ทำการศึกษาและพัฒนาในส่วนของแอพพลิเคชั่นและฮาร์ดแวร์ เพื่อใช้ในการจัดการการใช้งานโหลดบาลานซ์เซิร์ฟเวอร์ ซึ่งสามารถใช้งานได้ทั้งสองส่วน โดยโหลดบาลานซ์เซิร์ฟเวอร์ คือ ระบบการกระจายภาระงานในเครือข่ายไปยังเครื่องแม่ข่าย (Server) ซึ่งได้แก่ ระบบเว็บเซิร์ฟเวอร์ (Web Server) โดยโหลดบาลานซ์จะเป็นตัวจัดการกระจายภาระงานจากผู้ใช้งาน (Users) ที่ Access เข้ามาใช้บริการเว็บไซต์ โดยจะกระจายงานไปยังกลุ่มของเซิร์ฟเวอร์ที่จัดเตรียมไว้หรือ Real Server เพื่อทำให้สภาพการจราจรภายในเครือข่ายมีความคล่องตัวและมีเสถียรภาพ และเว็บเซิร์ฟเวอร์ (Web Server) ก็สามารถรองรับการใช้งานของผู้ใช้งาน (Users) เป็นจำนวนเพิ่มขึ้นได้ ซึ่งในการกระจายภาระงานของโหลดบาลานซ์นั้นจะมีหลายวิธีการในการกระจายงานหลายวิธีแตกต่างกันออกไป ซึ่งในปริญญานิพนธ์นี้ได้แบ่งวิธีการการกระจายภาระงานออกเป็น 4 แบบคือ Round robin, Weighted Round Robin, Fast Response และ Least Connection ซึ่งเป็นวิธีที่มีการใช้งานได้ดีและครอบคลุมสภาพแวดล้อมในการใช้งานระบบเว็บเซิร์ฟเวอร์ (Web Server)

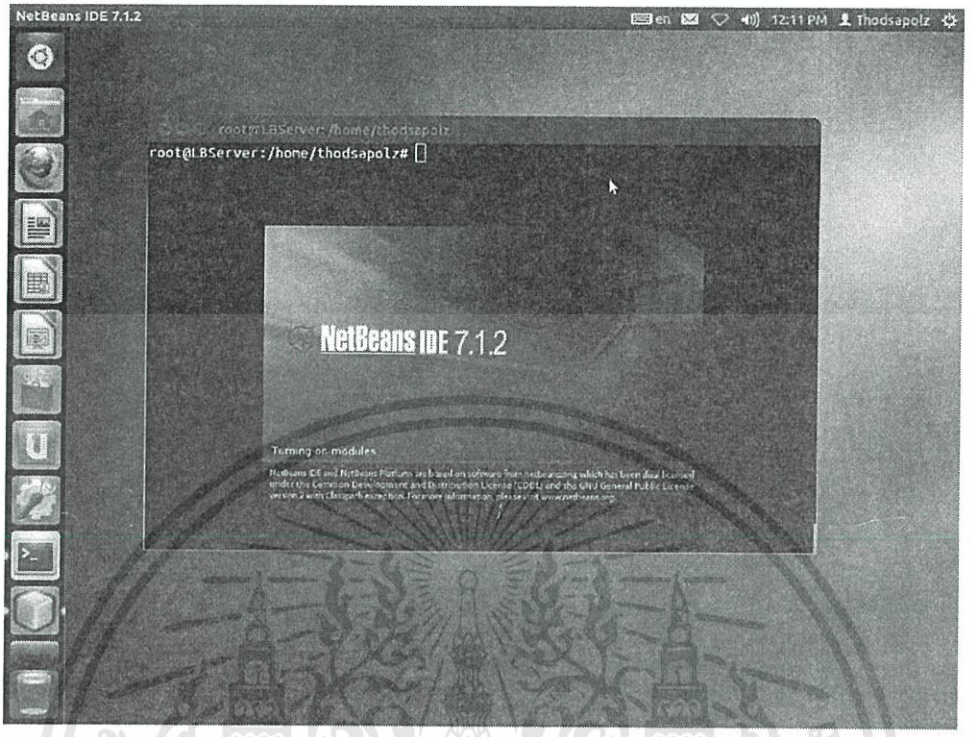


รูปที่ 3.1 ระบบการทำงานของโหลดบาลานซ์

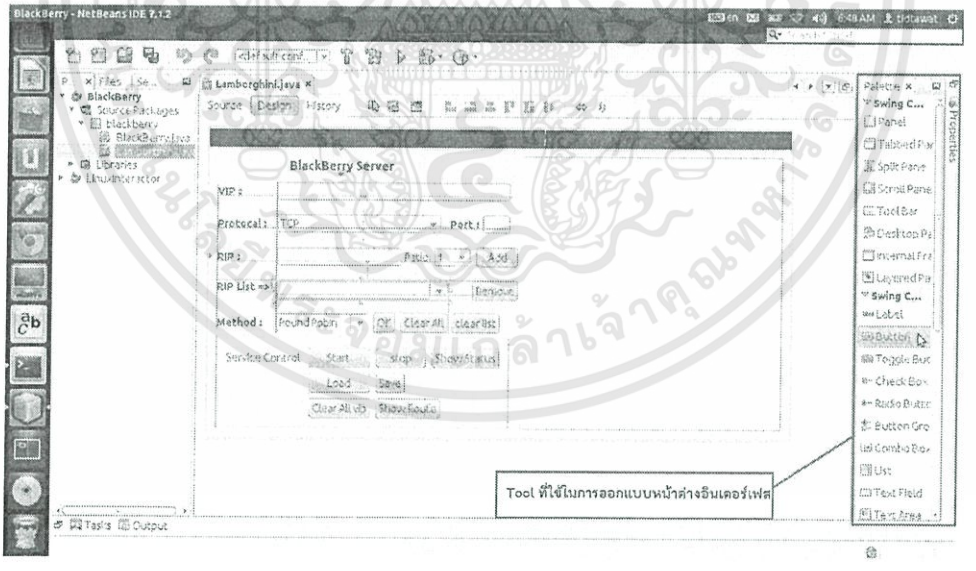
ปริญญานิพนธ์นี้ผู้ศึกษาได้ทำการศึกษาโหลดบาลานซ์ ที่ใช้งานอยู่บนระบบปฏิบัติการลีนุกซ์ ซึ่งผู้ใช้งานได้เลือกใช้ซอร์ฟแวร์ลีนุกซ์อูบุนตูในการใช้งาน เพราะเป็นซอร์ฟแวร์ที่ใช้ได้อย่างเสรีและสามารถนำไปพัฒนาและต่อยอดได้ ซึ่งในการทำโหลดบาลานซ์เซิร์ฟเวอร์นั้นจะต้องใช้งานโหลดบาลานซ์เซอร์วิส ในระบบปฏิบัติการลีนุกซ์ ซึ่งเซอร์วิสนั้นก็คือ IPVSADM โดยการทำปริญญานิพนธ์ผู้ศึกษาได้ทำการศึกษาวิธีการทำงานของเซอร์วิส IPVSADM เพื่อเรียนรู้คำสั่ง (command) เพื่อที่จะนำมาทำโหลดบาลานซ์เซิร์ฟเวอร์ ตามวิธีการที่เหมาะสมเพื่อใช้งานในระบบเครือข่ายให้มีประสิทธิภาพสูงสุด ดังนั้นผู้ศึกษาจึงได้มีการพัฒนาแอพพลิเคชั่นและฮาร์ดแวร์เพื่อที่จะนำมาใช้ในการจัดการเซอร์วิส IPVSADM เพื่อที่จะนำไปทำโหลดบาลานซ์เซิร์ฟเวอร์ในระบบเว็บเซิร์ฟเวอร์ทั่วไปได้ โดยในที่นี้ผู้ศึกษาจะแบ่งการอธิบายในส่วนของแอพพลิเคชั่นและฮาร์ดแวร์ ตามลำดับ

3.1.1 การออกแบบหน้าแอพพลิเคชั่น

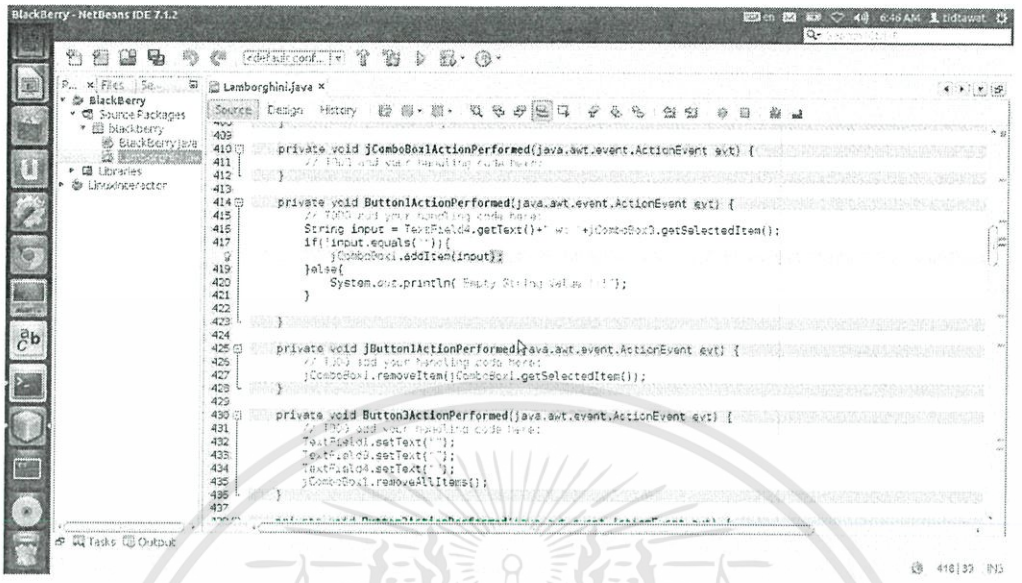
ผู้จัดทำได้ทำการพัฒนาแอพพลิเคชั่นเพื่อจัดการการใช้งานโหลดบาลานซ์เซิร์ฟเวอร์ขึ้นมา โดยได้สร้างเป็นรูปแบบของหน้าต่างอินเตอร์เฟส (Interface) ในการรับค่าพารามิเตอร์จากผู้ใช้หรือผู้ดูแลระบบ ซึ่งหน้าต่างอินเตอร์เฟส (Interface) นี้จะทำการป้อนค่าพารามิเตอร์ที่จำเป็นสำหรับโหลดบาลานซ์ ซึ่งสามารถทำโหลดบาลานซ์ได้โดยไม่ต้องไปพิมพ์คำสั่ง (command) ของ IPVASDM ทั้งหมดในลีนุกซ์โดยตรงผ่านหน้าต่างเทอร์มินอล (terminal) ซึ่งจุดประสงค์ที่ทำการพัฒนาแอพพลิเคชั่นนี้ขึ้นมาเพื่อที่จะให้ผู้ใช้งานและผู้ดูแลระบบใช้งานได้ง่าย โดยสามารถเลือกการทำโหลดบาลานซ์หรือวิธีการกระจายภาระงานได้ตามที่ต้องการโดยไม่ต้องเรียนรู้คำสั่งของเซอร์วิส IPVSADM ซึ่งมีความยุ่งยาก โดยในการพัฒนาแอพพลิเคชั่นจะพัฒนาโดยใช้ภาษาจาวา (Java) เพื่อทำการสร้างเงื่อนไขในการรับค่าพารามิเตอร์และทำการตัดสตริง (string) เพื่อนำไปรวมเป็นคำสั่งในรูปแบบของเซอร์วิส IPVSADM และสร้างหน้าอินเตอร์เฟสด้วยโปรแกรมเน็ตบีนไอดีอี (Netbean IDE) เพื่อสร้างหน้าอินเตอร์เฟสในการรับค่าพารามิเตอร์ที่สำคัญต่างๆจากผู้ใช้งาน



รูปที่ 3.2 หน้าต่างการทำงานของโปรแกรม NetBeans IDE 7.1.2

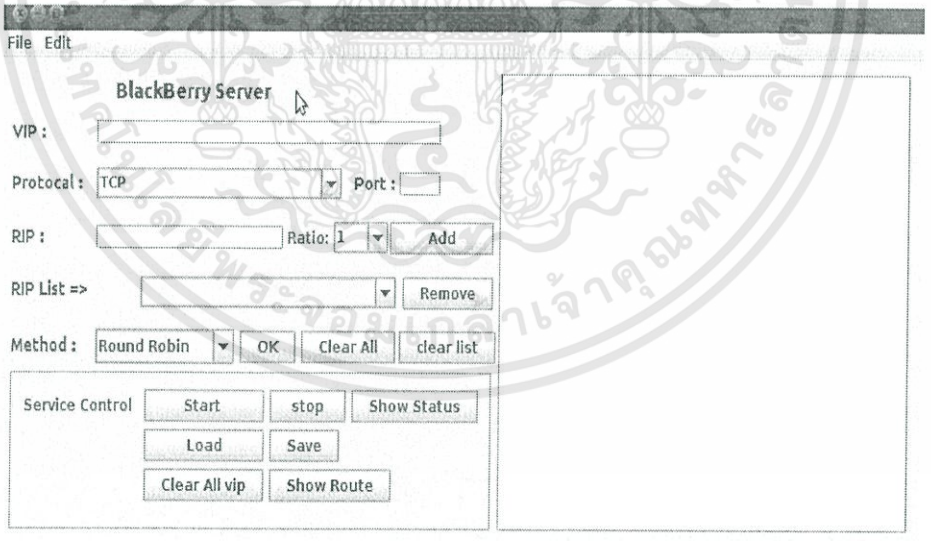


รูปที่ 3.3 การออกแบบหน้าต่างอินเตอร์เฟสโดยใช้บล็อกเครื่องมือ (Tool) ของแอพพลิเคชั่น NetBeans IDE 7.1.2



รูปที่ 3.4 การใช้ภาษาจาวาเขียนโปรแกรมรับค่าพารามิเตอร์และเขียน Process Builder เพื่อส่งค่าพารามิเตอร์ไปยังฐานคำสั่งของระบบปฏิบัติการลีนุกซ์บางส่วน

ซึ่งหน้าอินเตอร์เฟสของแอพพลิเคชั่นแสดงได้ดังรูปและอธิบายรายละเอียดได้ดังนี้



รูปที่ 3.5 หน้าอินเตอร์เฟสของแอพพลิเคชั่น

ในหน้าต่างอินเตอร์เฟสจะมีช่องในการรับค่าพารามิเตอร์จากผู้ใช้งานดังนี้

VIP (virtual Server)

RIP (Real Server)

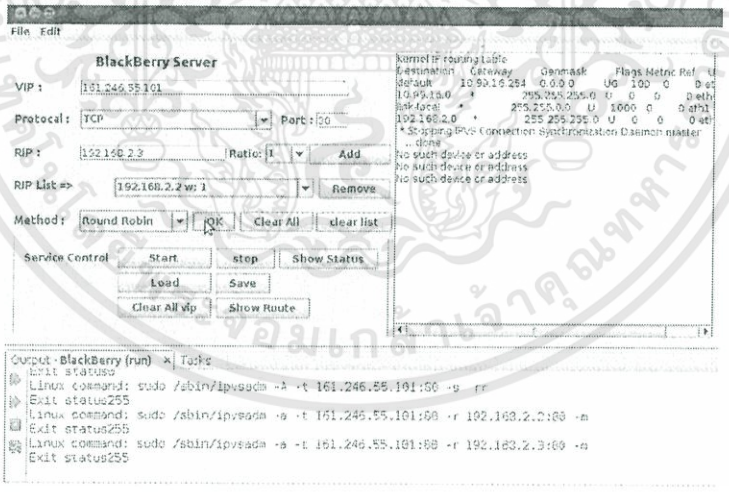
Port (ถ้าเว็บเซิร์ฟเวอร์คือ พอร์ต 80)

Protocol (TCP,UDP)

Method - Round Robin

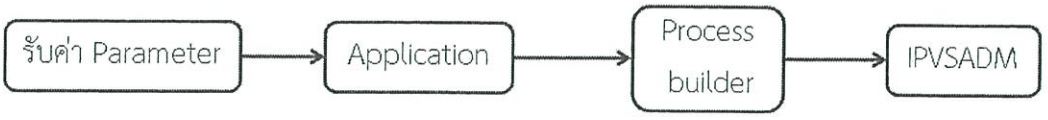
- Weighted Round Robin
- Fast Response
- Least Connection

ซึ่งค่าพารามิเตอร์ต่างๆที่ผู้ใช้งานหรือผู้ดูแลระบบได้ทำการป้อนค่าลงไปในหน้าอินเตอร์เฟส เมื่อสั่งให้โปรแกรมทำงานแอพพลิเคชั่นจามารถรวบรวมเป็นชุดคำสั่งของเซอร์วิส IPVSADM และรวบรวมชุดคำสั่งนี้ส่งไปยังลีนุกซ์เพื่อใช้งานเซอร์วิส IPVSADM เพื่อทำโหลดบาลานซ์ตามที่วิธีการที่ป้อนค่าไว้ โดยจะแสดงแผนผังการทำงาน เมื่อสั่งการแอพพลิเคชั่นให้ทำโหลดบาลานซ์ ได้ดังรูป



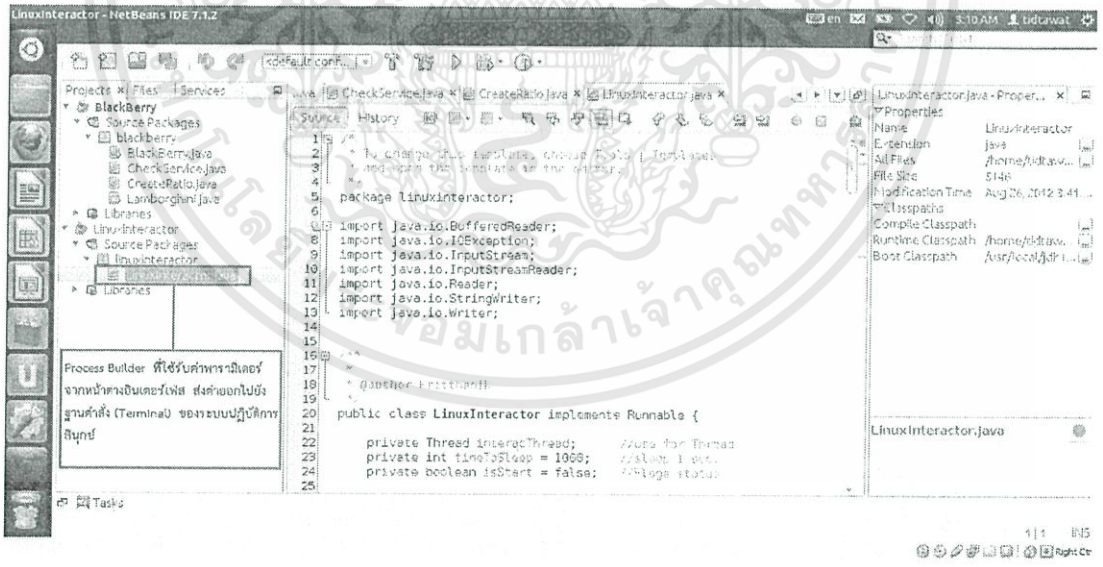
รูปที่ 3.6 หน้าอินเตอร์เฟสของแอพพลิเคชั่นขณะทำการกระจายโหลด

ในหลักการทำงานของแอพพลิเคชั่นจัดการการใช้งานโหลดบาลานซ์เซิร์ฟเวอร์ จะแสดงการทำงานของหน้าต่างอินเตอร์เฟสได้ดังบล็อกไดอะแกรมดังนี้



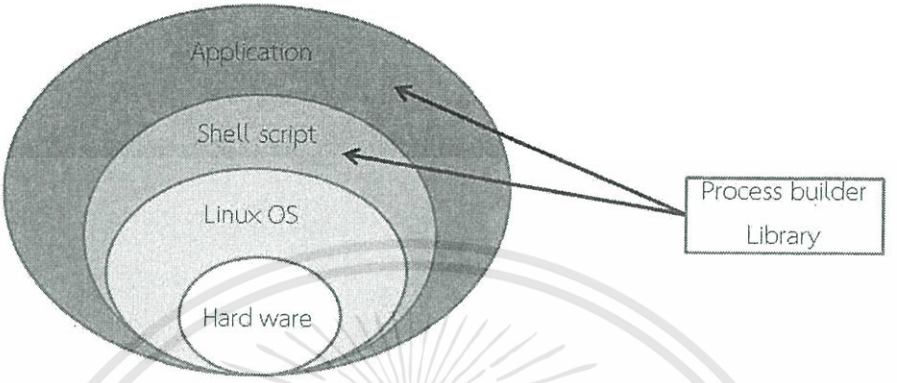
รูปที่ 3.7 แผนผังการติดต่อสื่อสารของหน้าต่างอินเตอร์เฟส

เมื่อผู้ใช้งานหรือผู้ดูแลระบบได้ทำการป้อนค่าพารามิเตอร์ที่หน้าอินเตอร์เฟสของแอพพลิเคชั่นแล้ว แอพพลิเคชั่นจะทำการเก็บค่าพารามิเตอร์และทำการตัดสตริง (string) เพื่อนำค่าพารามิเตอร์ไปรวมกันกับซินแทกซ์ (Syntax) ที่ได้เขียนด้วยภาษาจาวา (Java) ทำให้เป็นชุดคำสั่ง (command) ตามรูปแบบของเซอร์วิส IPVASDM จากนั่นเมื่อได้ชุดคำสั่งแล้วแอพพลิเคชั่นจะทำการส่งชุดคำสั่งไปยังลีนุกซ์เพื่อใช้งาน IPVSADM ในการทำโหลดบาลานซ์ ซึ่งในการติดต่อระหว่างแอพพลิเคชั่นกับลีนุกซ์จะอาศัยกระบวนการ Process Builder เป็นตัวกลางในการสื่อสาร ซึ่ง Process Builder Library ผู้ศึกษาได้พัฒนาขึ้นด้วยภาจาจาวา (Java)



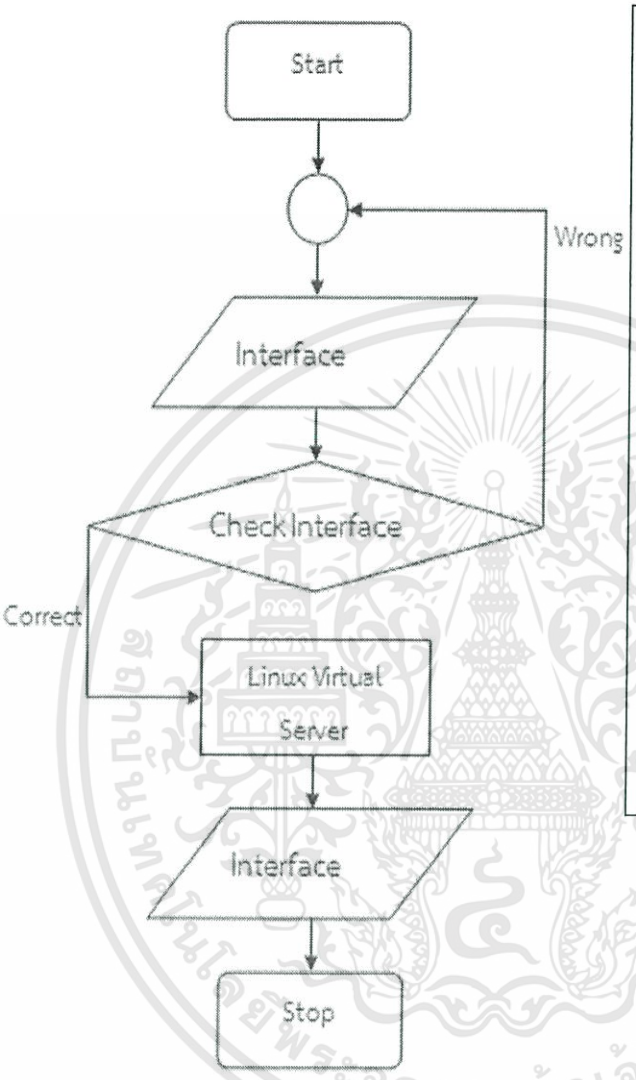
รูปที่ 3.8 LinuxInteractor เป็น Process ที่ใช้ในการรับค่าพารามิเตอร์จากหน้าต่างอินเตอร์เฟส เพื่อส่งคำสั่งออกไปยังฐานคำสั่ง (Terminal) ของระบบปฏิบัติการลีนุกซ์

ในภาพรวมของการติดต่อสื่อสารระหว่างแอพพลิเคชั่นกับเซอร์วิส IPVASDM เพื่อทำโหลดบาลานซ์ จะแสดงได้ดังแผนผังต่อไปนี้



รูปที่ 3.9 แผนผังของการติดต่อสื่อสารระหว่างแอพพลิเคชั่นกับเซอร์วิส IPVASDM

จากแผนภาพจะอธิบายการทำงานระหว่างแอพพลิเคชั่นกับเซอร์วิส IPVASDM โดยในการที่จะติดต่อสื่อสารและจัดการการใช้งานได้นั้น จะต้องมี process builder library เป็นสื่อกลางซึ่ง process builder จะทำการรวมค่าพารามิเตอร์จากชั้นของแอพพลิเคชั่น ซึ่งก็คือภาษาจาวา (Java) ส่งลงไปยังชั้นของ shell script ซึ่ง shell script ก็คือการรวมซินแท็กซ์ (syntax) เข้าด้วยกันเป็นชุดคำสั่งในรูปแบบของเซอร์วิส IPVSADM ซึ่งถ้าไม่มีชั้น shell script จะส่งได้แค่ทีละคำสั่งหรือ command line หลังจากนั้น shell script จะทำการติดต่อกับเซอร์วิส IPVSADM เพื่อส่งชุดคำสั่งเข้าไปทำงานเพื่อทำโหลดบาลานซ์ตามวิธีการที่ผู้ใช้งานหรือผู้ดูและระบบป้อนที่แอพพลิเคชั่น



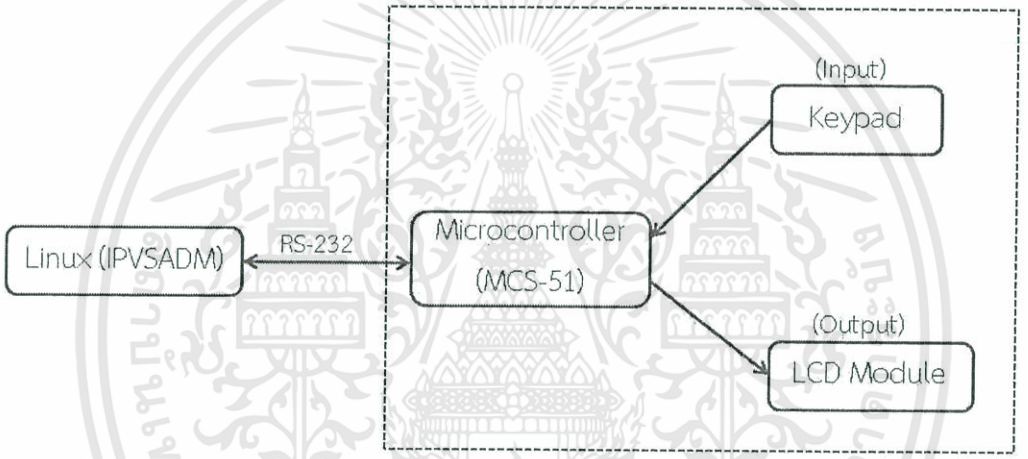
คำอธิบายการทำงาน(โดยย่อ)

แอพพลิเคชันที่ออกแบบจะรับคำสั่งจากผู้ดูแลระบบผ่านหน้าต่าง Interface ที่ได้ออกแบบขึ้นจากนั้นแปลงค่าอินพุทให้เป็นค่าพารามิเตอร์ไปยัง Linux Virtual Server ซึ่งเป็นตัวควบคุมระบบ หากคำสั่งที่ป้อนเข้ามาถูกต้อง Linux Virtual Server ก็จะทำการควบคุมตัวกระจายภาระงาน (Load Balance) ให้แบ่งงานแก่เครื่องแม่ข่าย (Server) หลายๆเครื่องให้ทำงานพร้อมกันได้อย่างสมดุล ตามวิธีการและคำสั่งที่ผู้ดูแลระบบได้

รูปที่ 3.10 Flow Chart การทำงานหน้าต่างอินเตอร์เฟสของแอพพลิเคชั่น

3.1.2 การออกแบบ Panel Controller

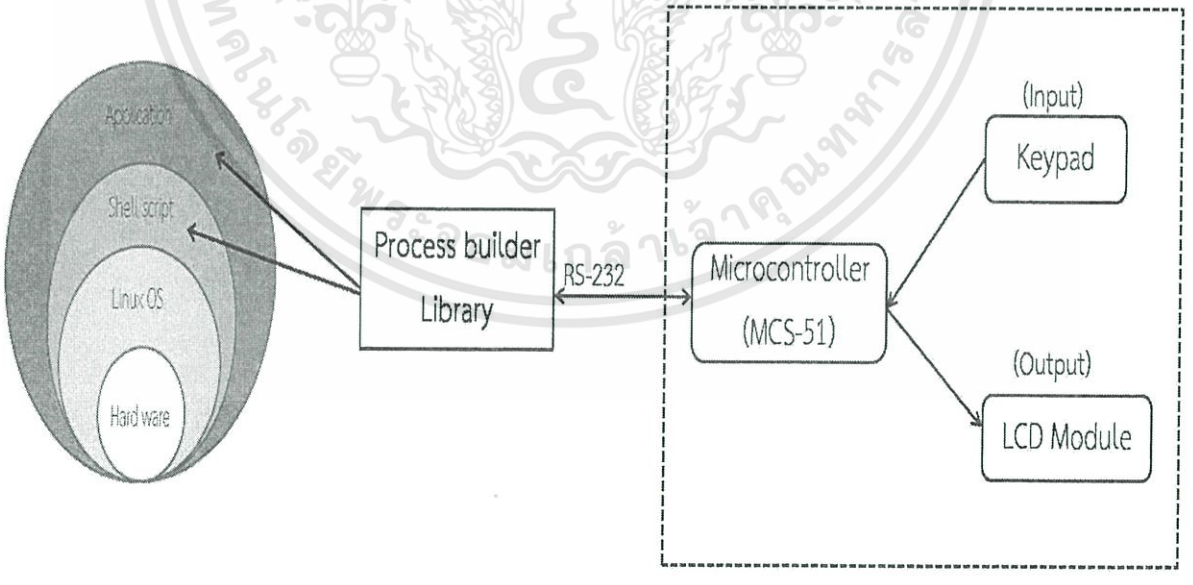
ปริญญานิพนธ์นี้ผู้ศึกษาได้ทำการศึกษาและสร้างฮาร์ดแวร์หรือ Panel controller เพื่อนำมาใช้ในการจัดการการใช้งานโหลดบาลานซ์เซิร์ฟเวอร์ เพื่อให้การใช้งานทำได้ง่ายโดยไม่ต้องป้อนค่าพารามิเตอร์ผ่านหน้าอินเตอร์เฟส ซึ่งฮาร์ดแวร์ นี้จะพัฒนาด้วยไมโครคอนโทรเลอร์ MCS-51 ซึ่งพัฒนาด้วยภาษา C และทำการป้อนคำสั่ง/เลือกฟังก์ชั่นการทำงานด้วย Metrix switch หรือคีย์แพด (Keypad) และแสดงผลด้วยแอลซีดีโมดูล (LCD Module) ขนาด 40x4 ซึ่งจะแสดงแผนผังการใช้งานได้ดังรูป



รูปที่ 3.11 แผนผังแสดงการติดต่อสื่อสารระหว่างฮาร์ดแวร์กับลีนุกซ์

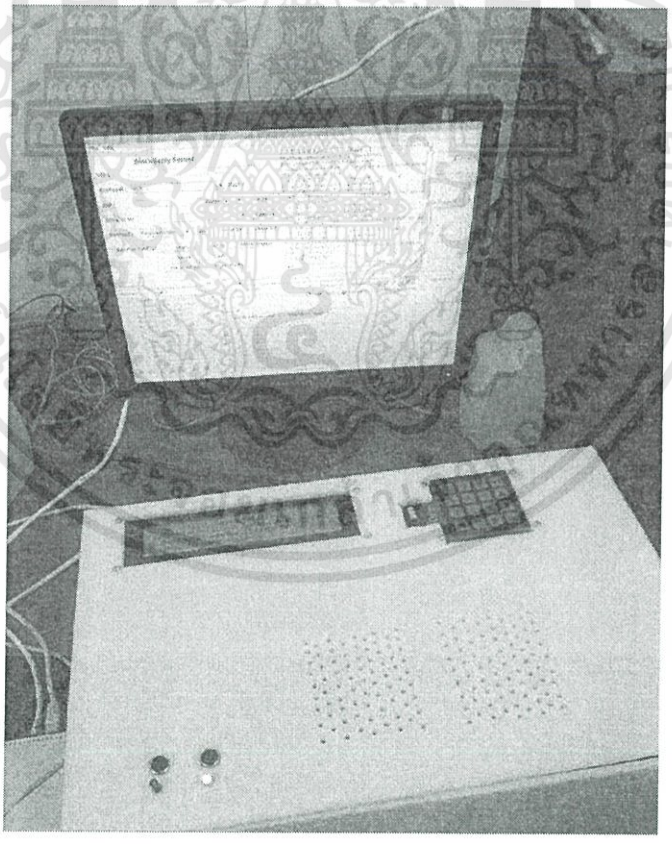
จากรูป ฮาร์ดแวร์ ที่สร้างขึ้นมาจะสามารถป้อนค่าอินพุทในการเลือกฟังก์ชั่นการทำงานและป้อนค่าพารามิเตอร์ (ค่า IP address) ได้และเก็บค่าอินพุตพารามิเตอร์ลงบนไมโครคอนโทรเลอร์ ซึ่งไมโครคอนโทรเลอร์นี้ได้ทำการพัฒนาเฟิร์มแวร์ (Firmware) ขึ้นมาด้วยภาษา C ซึ่งเป็นไฟล์ .HEX เพื่อนำไปอัดโปรแกรมลงบนไมโครคอนโทรลเลอร์ให้ทำงานร่วมกับระบบปฏิบัติการลีนุกซ์ โดยไมโครคอนโทรเลอร์ จะทำการส่งข้อมูล (data) ในรูปแบบของสัญญาณไฟฟ้าส่งผ่านพอร์ตอนุกรม RS-232 ไปยังลีนุกซ์ โดยมี process builder ทำการแปลงสัญญาณข้อมูลไปเป็นรูปแบบของ message เพื่อแปลงเป็นคำสั่ง (command) ในรูปแบบของเซอร์วิส IPVASDM เพื่อทำโหลดบาลานซ์ โดยในการติดต่อสื่อสารเน้นระหว่างลีนุกซ์กับฮาร์ดแวร์ จะใช้พอร์ตอนุกรม RS-232 เป็นสื่อกลางในการติดต่อกัน ซึ่งมี Process builder library เป็นข้อตกลงหรือเป็น Protocol ในการ

รับส่งข้อมูลของทั้งสองฝั่งหรือพูดง่ายๆเป็นการโต้ตอบกันระหว่างฮาร์ดแวร์กับลีนุกซ์ซึ่งในการทำงานนั้น Process builder library จะทำงานเสมือนเป็นไดร์เวอร์ของฝั่งลีนุกซ์เพื่อทำการรับข้อมูลที่เข้ามาจากไมโครคอนโทรลเลอร์ผ่านพอร์ตอนุกรม RS232 เมื่อป้อนค่าอินพุทผ่านคีย์แพด (keypad) และส่งข้อมูล (data) ไปยังลีนุกซ์ เราจะอาศัยกระบวนการในการทำอินเตอร์รัพท์เพื่อให้การติดต่อสื่อสารกันระหว่างเฟิร์มแวร์ (Firmware) และลีนุกซ์สามารถรับส่งข้อมูลและมีผลตอบสนองกันได้ เช่น เมื่อเลือกค่าฟังก์ชั่นใดฟังก์ชั่นหนึ่งจากไมโครคอนโทรเลอร์เข้าไปยังลีนุกซ์ ดังนั้นลีนุกซ์ก็จะส่งค่าอินพุตกลับมาแสดงผลยังแอลซีดีโมดูล (LCD Module) ของไมโครคอนโทรลเลอร์ซึ่งถือว่าเสร็จสิ้นกระบวนการทำอินเตอร์รัพท์และในการส่งค่ารับค่าต่างๆ ลีนุกซ์ก็จะทำการรับทีละคำสั่งซึ่งไมโครคอนโทรลเลอร์ก็จะมีฟังก์ชั่นในการป้อนค่าพารามิเตอร์ไปเรื่อยๆ จนครบทุกฟังก์ชั่น ซึ่งจะส่งคำสั่งไปยัง ลีนุกซ์ตามลำดับ ซึ่งไมโครคอนโทรลเลอร์ก็จะทำการรวมค่าพารามิเตอร์ที่ป้อนทั้งหมดเก็บไว้ เมื่อป้อนค่าอินพุทจากไมโครคอนโทรเลอร์ครบทุกฟังก์ชันแล้วและทำการส่งค่าพารามิเตอร์โดยการ Commit ที่อุปกรณ์ฮาร์ดแวร์ Process Builder ก็จะทำการรวมแปลงข้อมูลที่ส่งมาจากไมโครคอนโทรลเลอร์ผ่านพอร์ตอนุกรม RS-232 เป็นชุดคำสั่งในรูปแบบของเซอร์วิส IPVSADM และส่งคำสั่งไปยัง IPVSADM เพื่อทำโหลดบาลานซ์ซึ่งในการทำงานทั้งหมดจะแสดงได้ดังแผนภาพ



รูปที่ 3.12 แผนภาพแสดงการทำงานของระบบ

จากแผนภาพเราจะอธิบายถึงฟังก์ชั่นในการทำงานของไมโครคอนโทรเลอร์ได้ดังนี้เนื่องจากในการทำโหลดบาลานซ์ผ่านฮาร์ดแวร์นั้นจะไม่มีการป้อนค่าพารามิเตอร์และเก็บคำสั่งข้อมูลไว้ในไมโครคอนโทรเลอร์เนื่องจากหน่วยความจำ (memory) มีไม่เพียงพอในการเก็บค่า syntax เพื่อรวมเป็นคำสั่ง ดังนั้น เราจึงใช้ process builder ที่สร้างขึ้นเพื่อใช้ในการเก็บค่า syntax และรวมกันเป็น command เพื่อส่งค่าคำสั่งไปยัง IPVSADM ชุดเดียว ดังนั้นไมโครคอนโทรเลอร์จึงใช้หลักการสร้างฟังก์ชันในการป้อนค่าพารามิเตอร์ ซึ่งในการป้อนค่าพารามิเตอร์แต่ละแบบเช่น VIP, RIP, Ratio Method ไมโครคอนโทรเลอร์จะทำการโต้ตอบกับลีนุกซ์เป็นครั้งๆ และส่งค่าพารามิเตอร์ไปเป็นรอบๆ จนครบทุกค่า ซึ่งกระบวนการนี้คือกระบวนการสร้างฟังก์ชันการป้อนค่าพารามิเตอร์ซึ่งฟังก์ชันต่างๆที่แสดงผลบนแอลซีดีโมดูล (LCD Module) คือการดึงข้อมูลจากลีนุกซ์เข้ามายังไมโครคอนโทรเลอร์เพื่อทำการแสดง และป้อนค่าพารามิเตอร์ผ่านคีย์แพด (Keypad) และส่งข้อมูล (Data) ออกไปด้วยพอร์ตอนุกรม RS-232

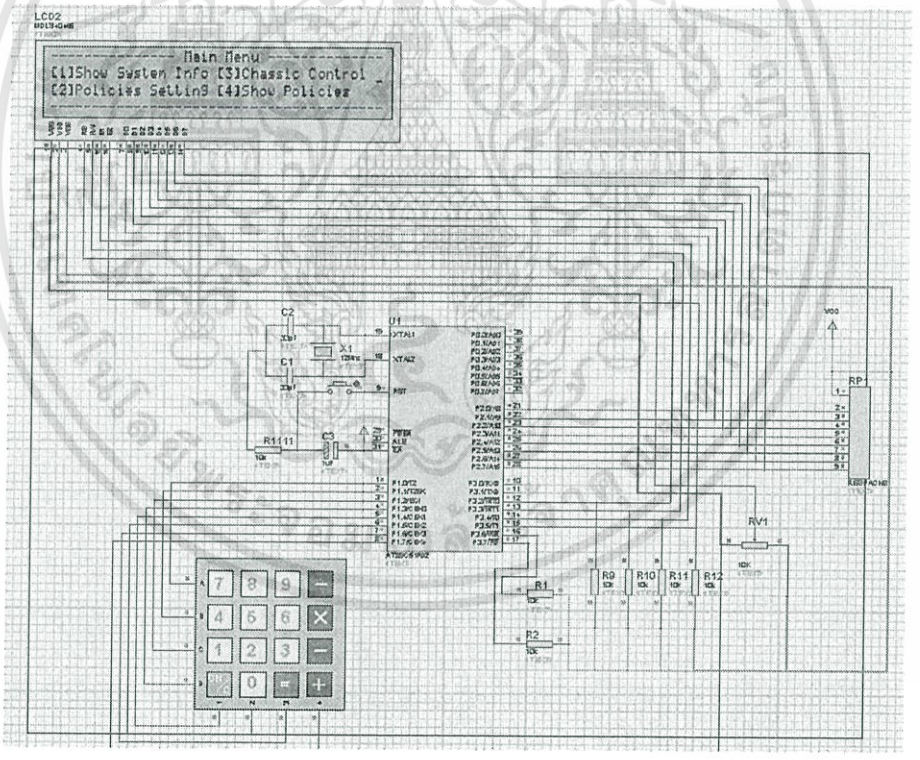


รูปที่ 3.13 ฮาร์ดแวร์โหลดบาลานซ์เซิร์ฟเวอร์

ในการสร้างฮาร์ดแวร์ในการทำโหลดบาลานซ์เซิร์ฟเวอร์เราได้มีการใช้ซอฟแวร์เข้ามาช่วยในการออกแบบ ซึ่งประกอบไปด้วยซอฟแวร์ดังนี้

3.1.2.1 โปรแกรม ISIS 7 Professional (Proteus ver.7)

เป็นโปรแกรมเพื่อใช้ในการจำลองการสร้างฮาร์ดแวร์โดยการต่อวงจรด้วยไมโครคอนโทรลเลอร์ MCS-51 ซึ่งป้อนค่าอินพุทพารามิเตอร์ด้วย Metrix Switch และแสดงผลด้วยแอลซีดีโมดูล (LCD Module) ขนาด 40x4 และในโปรแกรมได้ทำการอัดโปรแกรมเฟิร์มแวร์ (Firmware) ในไมโครคอนโทรลเลอร์ที่พัฒนาด้วยภาษาซี เพื่อทำการทดลองการรับค่าอินพุตพารามิเตอร์จากคีย์แพด (Keypad) เพื่อไปแสดงผลยังแอลซีดีโมดูล (LCD Module) ได้ตามฟังก์ชั่นที่เขียนบนเฟิร์มแวร์ (Firmware)



รูปที่ 3.14 การจำลองการทำงานวงจรด้วยโปรแกรม ISIS 7 Professional (Proteus ver.7)

3.1.2.2 โปรแกรม Keil uVision3

ปริญญานิพนธ์นี้ผู้ศึกษาได้ทำการศึกษาการทำงานของแอลซีดีโมดูล (LCD Module) และคีย์แพด (Keypad) เพื่อใช้ในการแสดงผลและป้อนอินพุตคำสั่งผ่านไมโครคอนโทรลเลอร์ MCS-51 และได้เขียนโค้ดภาษาซีลงบนไมโครคอนโทรลเลอร์เพื่อทำการรับค่าอินพุตจากคีย์แพด (Keypad) โดยใช้พอร์ต P1.0-P1.7 ของไมโครคอนโทรเลอร์ในการรับค่าอินพุตจากคีย์แพด (Keypad) เพื่อส่งค่าอินพุตไปยังแอลซีดีโมดูล (LCD Module) เพื่อให้แสดงผลตามโปรแกรมที่เขียน โดยแอลซีดีโมดูล (LCD Module) กำหนดให้ใช้พอร์ต P2.0-P2.7 ของไมโครคอนโทรเลอร์ ซึ่งผู้ศึกษาได้ทำการเขียนโค้ดภาษาซีด้วยโปรแกรม Keil uVision3 เพื่อสร้างเฟิร์มแวร์ (Firmware) ไฟล์ .hex อัดโปรแกรมลงบนบอร์ดไมโครคอนโทรลเลอร์และได้ทำการทดลอง Simulate วงจรทั้งหมดด้วยโปรแกรม ISIS 7 Professional (Proteus ver.7) ตลอดจนถึงการทดลองต่อวงจรไมโครคอนโทรลเลอร์กับคีย์แพด (Keypad) และแอลซีดีโมดูล (LCD Module) และทำการทดลองซึ่งผลที่ได้เป็นไปตามโค้ดที่เขียนและการทดลอง Simulate ซึ่งคีย์แพด (Keypad) สามารถส่งค่าอินพุตพารามิเตอร์ผ่านไมโครคอนโทรลเลอร์ MCS-51 และสามารถแสดงผลบนแอลซีดีโมดูล (LCD Module) ตามโปรแกรมที่เขียนขึ้นได้

```
001 // LCD 40x4
002 // Refer code http://www.npeducations.com/2012/08/8051-based-16x2-lcd-interfacing-in-8.html
003 #include<reg52.h>
004 #include<stdio.h>
005 #include<intrins.h>
006 #include<absacc.h>
007 #define LCD_PORT P2      // define port
008
009 typedef int bool;
010 #define true 1
011 #define false 0
012
013 void send_cmd(unsigned char, int);  void send_data(unsigned char* ,int);
014 void send_data_tx(unsigned char *); void serial_data(unsigned char);
015 void send_char(unsigned char, int); void LCD_init();
016 void delayms(unsigned int);         unsigned char READ_SWITCHES();
017 unsigned char get_key();            void setPolicy();
018 void systemInfo();                  void control();
019 void showPolicy();                  void sconfig();
020 void clear();
021
022 sbit RS = P3^2; sbit En1 = P3^4; sbit swb1 = P3^6;
023 sbit RW = P3^3; sbit En2 = P3^5; sbit swb2 = P3^7;
024
025 sbit RowA = P1^0;       sbit  C1 = P1^4;
026 sbit RowB = P1^1;       sbit  C2 = P1^5;
027 sbit RowC = P1^2;       sbit  C3 = P1^6;
028 sbit RowD = P1^3;       sbit  C4 = P1^7;
029
```

รูปที่ 3.15 เขียนโค้ดภาษาซีด้วยโปรแกรม Keil uVision3 เพื่อสร้างเงื่อนไขในการรับค่าอินพุตจากคีย์แพด (Keypad) เพื่อส่งคำสั่งไปยังไมโครคอนโทรลเลอร์ MCS-51 เพื่อให้แอลซีดีโมดูล (LCD Module) แสดงผลตามโปรแกรม

```
051  void main(){
052      main_menu:
053
054      LCD_PORT = 0x00;     // Make the port as output
055      LCD_init();          // LCD initialization
056      sconfig();
057
058      send_cmd(0x80,1);    // Beginning of 1st line
059      delayms(100);
060      send_data("---------------- Main Menu --------------",1);
061      send_cmd(0xC0,1);    // Beginning of 2nd
062      delayms(100);
063      send_data("[1]Show System Info [3]Chassic Control",1);
064      send_cmd(0x80,2);    // Beginning of 2nd line c0
065      delayms(100);
066      send_data("[2]Policies Setting [4]Show Policies",2);
067      send_cmd(0xC0,2);    // Beginning of 2nd line
068      delayms(100);
069      send_data("----------------------------------------",2);
070
071      send_char(' ',1);
072      //send_cmd(0xC1,2);
073      //send_cmd(0x01,3);      // Clear display
074      delayms(100);
075
```

รูปที่ 3.16 ตัวอย่างโค้ดภาษาซีที่ต้องการอินพุตคำสั่งไปยังแอลซีดีโมดูล (LCD Module) เพื่อให้แสดงผลตามโปรแกรมตามฟังก์ชั่น main_menu

3.1.3 คำสั่งที่ใช้ในการควบคุมการกระจายภาระงาน (Load Balance) ของระบบปฏิบัติการลีนุกซ์

คำสั่งที่ใช้อ้างอิงได้จากภาคผนวก ข. MAN IPVSADM ซึ่งเป็นซอฟแวร์ที่ใช้ในการกระจายภาระงานของโหลดบาลานซ์ ซึ่งเป็นเครื่องมือ (Tool) ที่มีอยู่ในระบบปฏิบัติการลีนุกซ์

- ชุดคำสั่ง Service Control

/etc/init.d/ipvsadm start

/etc/init.d/ipvsadm stop

/etc/init.d/ipvsadm status

/etc/init.d/ipvsadm load

/etc/init.d/ipvsadm save

- ชุดคำสั่ง Load Balance

แบบ Round Robin

ipvsadm -A –t 192.168.18.249:80 –s rr

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.2:80 –m

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.3:80 –m

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.4:80 –m

หมายเหตุ : การกระจายภาระงานแบบ Round Robin ไม่จำเป็นต้องกำหนด Ratio (-w) เนื่องจากในการกระจายภาระงานแบบนี้ Ratio มีค่าเท่ากันหมดคือ “Ratio = 1”

แบบ Ratio

ipvsadm -A –t 192.168.18.249:80 –s wrr

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.2:80 –w (Ratio) –m

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.3:80 –w (Ratio) –m

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.4:80 –w (Ratio) –m

หมายเหตุ : การกระจายภาระงานแบบ Ratio ต้องทำการกำหนดค่าอัตราส่วน (Ratio) ให้แก่ Real Server ของแต่ละตัว โดยกำหนดเป็นตัวเลขหลัง “-w” ซึ่ง -w ในที่นี้คือ Weight

แบบ Fast Response

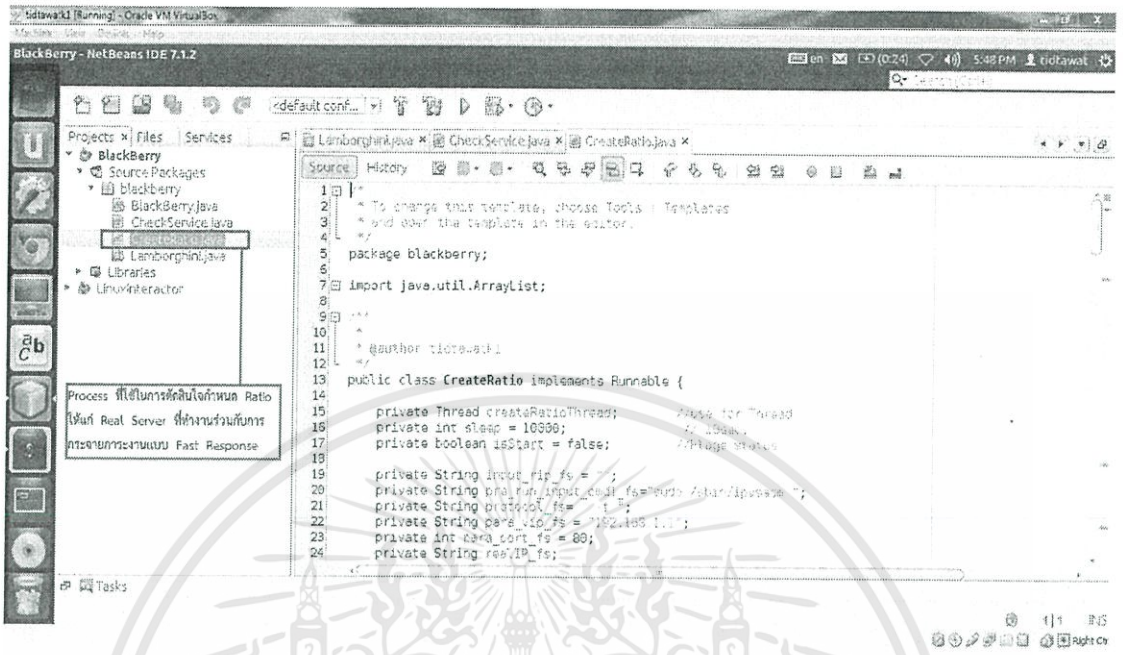
ipvsadm -A –t 192.168.19.249:80 –s wrr

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.2:80 –w (Ratio) –m

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.3:80 –w (Ratio) –m

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.4:80 –w (Ratio) –m

หมายเหตุ : การกระจายภาระงานแบบ Fast Response จะใช้คำสั่งเหมือนกันกับแบบ Ratio แต่จะใช้ Process เข้ามาช่วย ซึ่งการกระจายภาระงานแบบ Fast Response จะพิจารณาที่ผลการตอบสนองของ Real Server (time) ซึ่ง Process จะเป็นตัวกำหนดว่า Real Server ตัวไหนมีผลตอบสนองไวที่สุด ก็จะทำการกำหนด Ratio ให้เยอะที่สุด (รับงานมากที่สุด) ซึ่ง Process ที่เราได้สร้างขึ้นคือ “CreateRatio”



รูปที่ 3.17 CreateRatio เป็น Process ที่ใช้ในการตัดสินใจกำหนด Ratio ให้แก่ Real Server ที่ทำงานร่วมกับการกระจายภาระงานแบบ Fast Response

แบบ Least Connection

ipvsadm -A –t 192.168.18.249:80 –s lc

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.2:80 –m

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.3:80 –m

ipvsadm -a –t 192.168.18.249:80 –r 192.168.2.4:80 –m

หมายเหตุ : การกระจายภาระงานแบบ Least Connection จะใช้คำสั่งเหมือนกันกับแบบ Round Robin แต่ในการกระจายโหลดแบบ Least Connection จะทำการพิจารณางานที่ Real Server แต่ละตัวได้รับ โดยจะเลือกกระจายโหลดให้แก่ Real Server ที่มีการรับงานเข้ามาน้อยที่สุด ซึ่งอ้างอิงได้จากภาคผนวก ข. MAN IPVSADM

อ้างอิง : lc - Least-Connection: assigns more jobs to real servers with fewer active jobs.

ชุดคำสั่งในการตั้งค่า iptable

vi /etc/runIPTABLE

iptables –t nat –A POSTROUTING –s 192.168.2.0/24 –j MASQUERADE

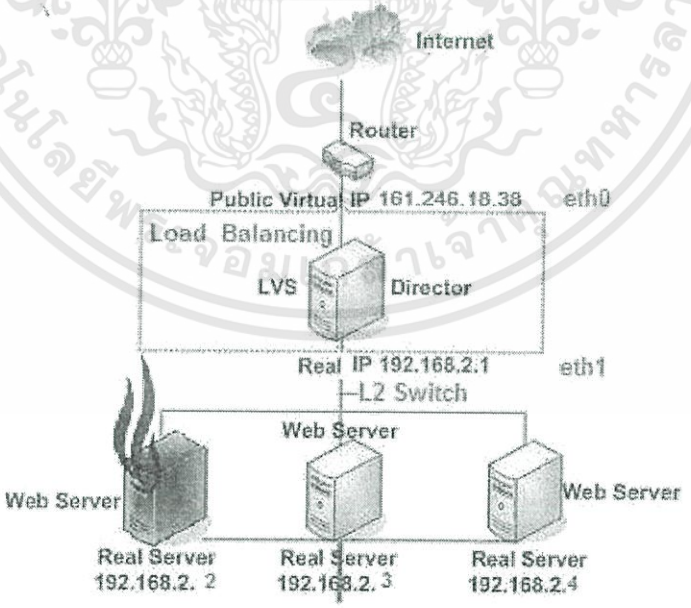
vi /etc/rc.local

more etc/runIPTABL

หมายเหตุ : เป็นการตั้งค่า iptable เพื่อทำการสร้างเส้นทางในการวิ่งออกไปข้างนอกวงของ Real Server

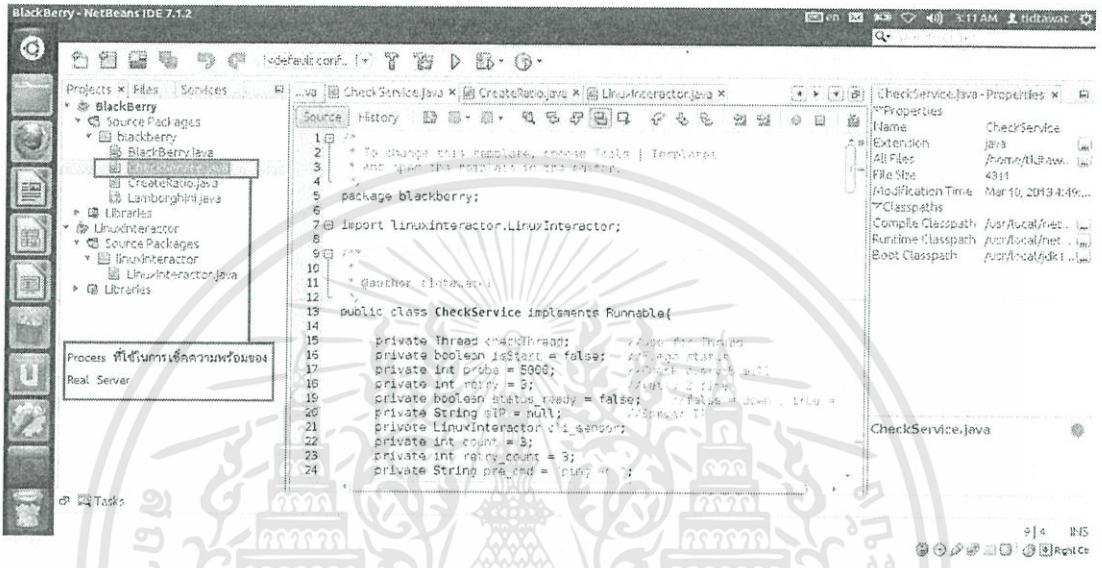
- การติดต่อแบบ Polling

การติดต่อแบบ Polling จะเป็นการตรวจสอบความพร้อมของเครื่องแม่ข่ายก่อนทำการกระจายโหลดซึ่งการทำ Polling จะมีเงื่อนไขคือ ถ้าเครื่องแม่ข่าย Real Server เกิด Fail Over ขึ้นมาหรือกำลังดาวน์อยู่ไม่สามารถทำงานได้ ระบบก็จะทำการเช็คเพื่อให้แน่ใจว่าเครื่องแม่ข่าย Real Server จะทำงานได้หรือไม่ ถ้าเครื่องแม่ข่ายไม่สามารถทำงานต่อไปได้หลังจากที่ทำการเช็คซ้ำภายในระยะเวลาและรอบการเช็คที่กำหนดไว้โหลดบาลานซ์เซิร์ฟเวอร์ ก็จะทำการหยุดกระจายโหลดไปยังเครื่องแม่ข่าย Real Server ที่ไม่สามารถทำงานได้จึงเป็นการประหยัดเวลาและทรัพยากรขึ้นในระบบดังรูปที่ 3.18



รูปที่ 3.18 การติดต่อแบบ Polling ในกรณีเครื่องแม่ข่าย Real Server เกิด Fail Over

ในแอพพลิเคชั่นควบคุมการกระจายภาระงานให้กับเครื่องแม่ข่ายหรือโหลดบาลานซ์เซิร์ฟเวอร์ (Load Balance Server) เราได้สร้าง Process ในการเช็คความพร้อมของ Real Server หรือการติดต่อแบบ Polling โดยเราได้สร้าง Process คือ CheckService

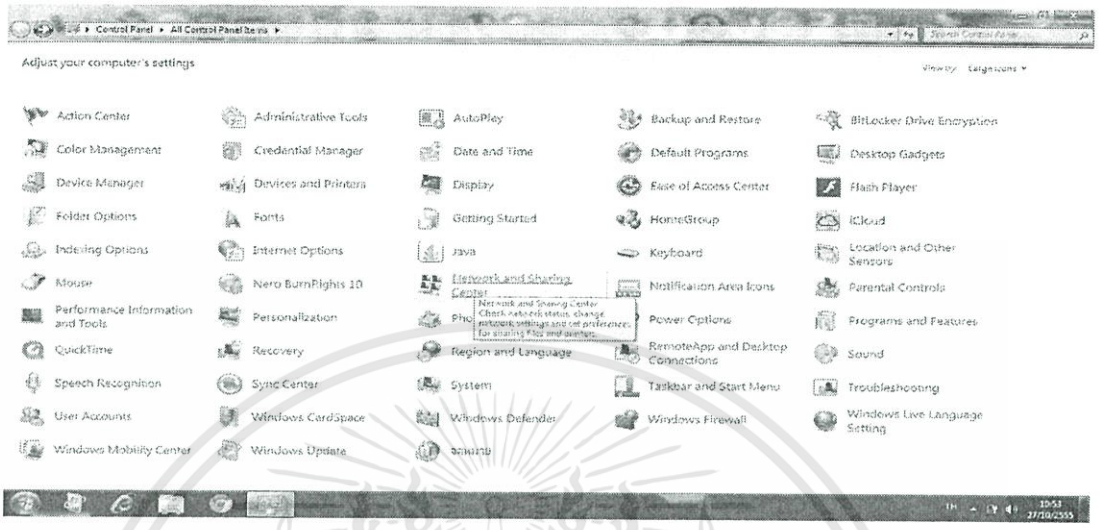


รูปที่ 3.19 CheckService เป็น Process ที่ใช้ในการเช็คความพร้อมของ Real Server

3.1.4 การจำลอง Web Server โดยใช้โน๊คบุค

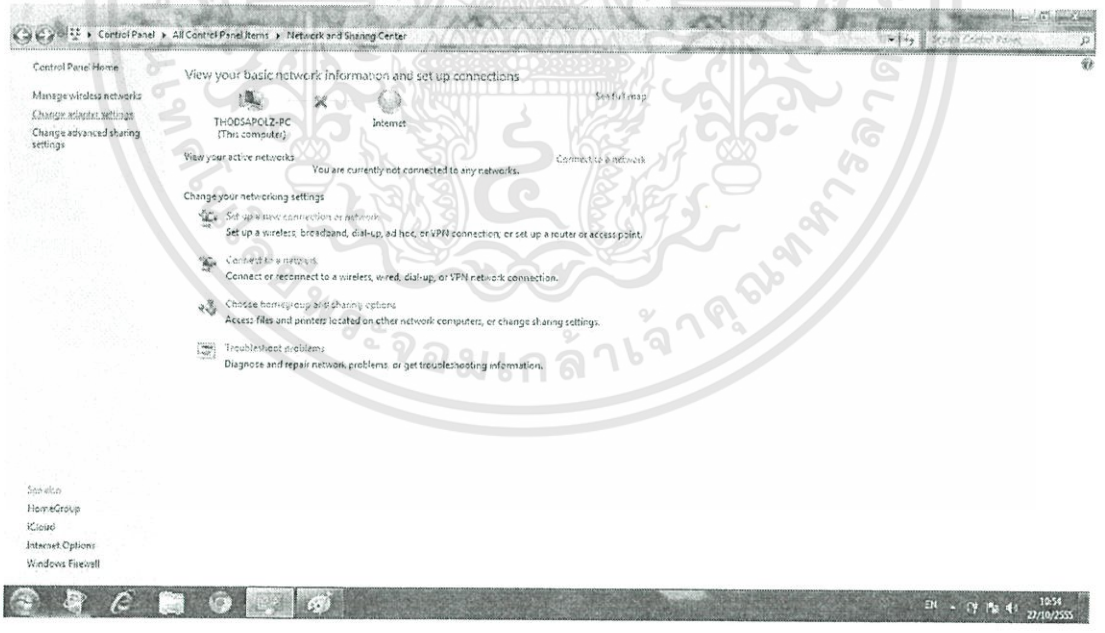
ในการจำลอง Web Server เพื่อใช้ในเป็นชุด Server ที่ใช้รองรับบริการของเครื่อง Client ที่จะทำการ Access เข้ามาใช้งานเว็บโดยการติดตั้งโปรแกรม Appserv 2.5.10 เพื่อจำลองเป็นเซิร์ฟเวอร์ ซึ่งหลังจากติดตั้งโปรแกรมแล้วก็ทำการตั้งค่า Real IP ตัวอย่างการตั้งค่าแสดงได้ดังรูป

1. ไปที่ Control Panel เลือกที่ Network and Sharing Center



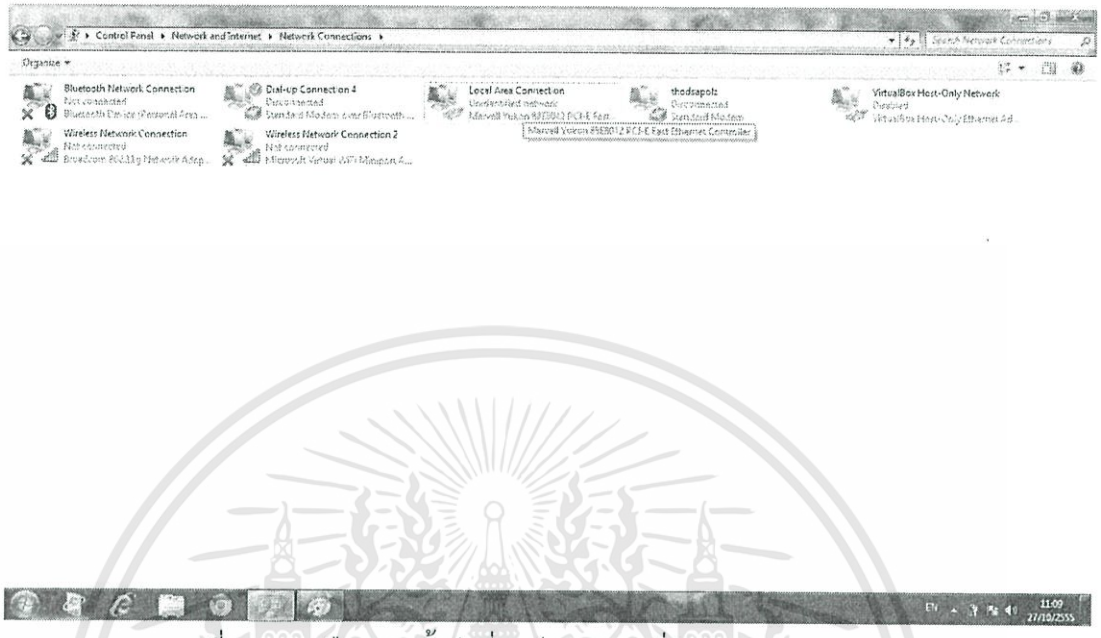
รูปที่ 3.20 การตั้งค่าการจำลอง Web Server โดยเลือก Network and Sharing Center

2. เลือกที่ Change adapter setting



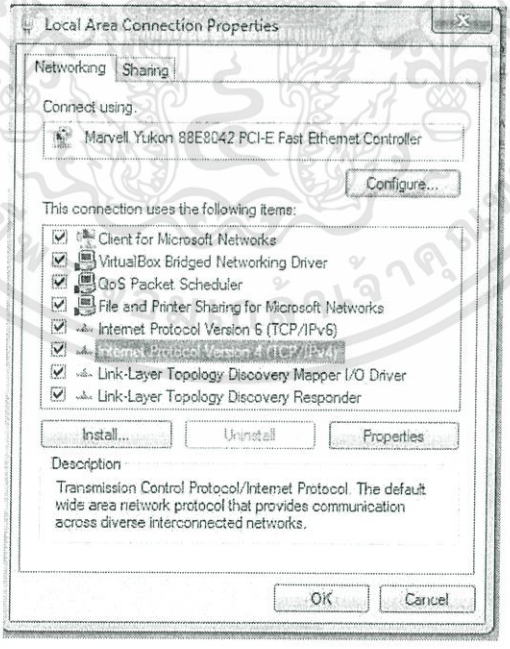
รูปที่ 3.21 การตั้งค่าการจำลอง Web Server โดยเลือก Change adapter setting

3. เลือกที่ Local Area Connection คลิกขวา เลือกที่ Properties



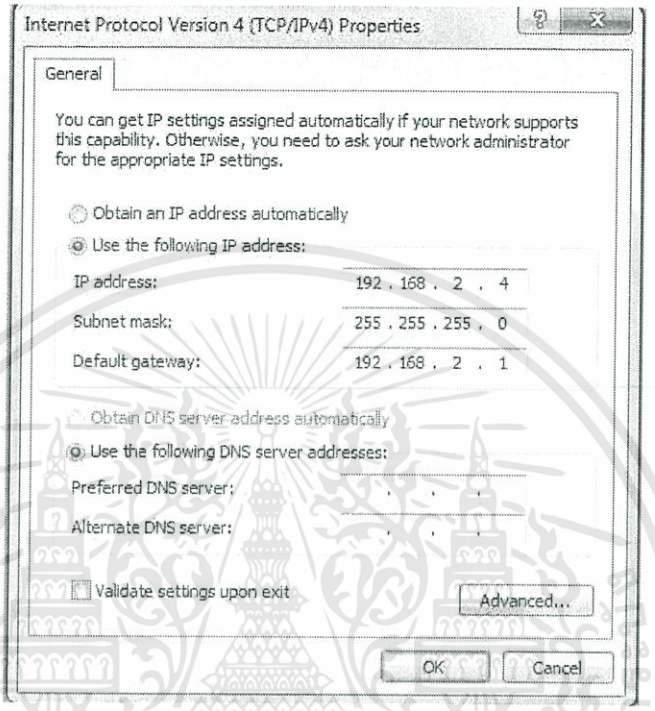
รูปที่ 3.22 เลือกการตั้งค่าที่พอร์ต LAN ที่ต่อกับ L2 Switch

4. เลือกที่ Internet Protocol Version 4 (TCP/IPv4) แล้วเลือกที่ Properties



รูปที่ 3.23 การตั้งค่า Real IP ใน Internet Protocol Version 4(TCP/IPv4)

5. เลือกที่ Use the following IP address โดยกำหนดค่า IP address เป็น 192.168.2.4, Subnet Mask เป็น 255.255.255.0 และ Default gateway เป็น 192.168.2.1



รูปที่ 3.24 การตั้งค่า Real IP

3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 ฮาร์ดแวร์

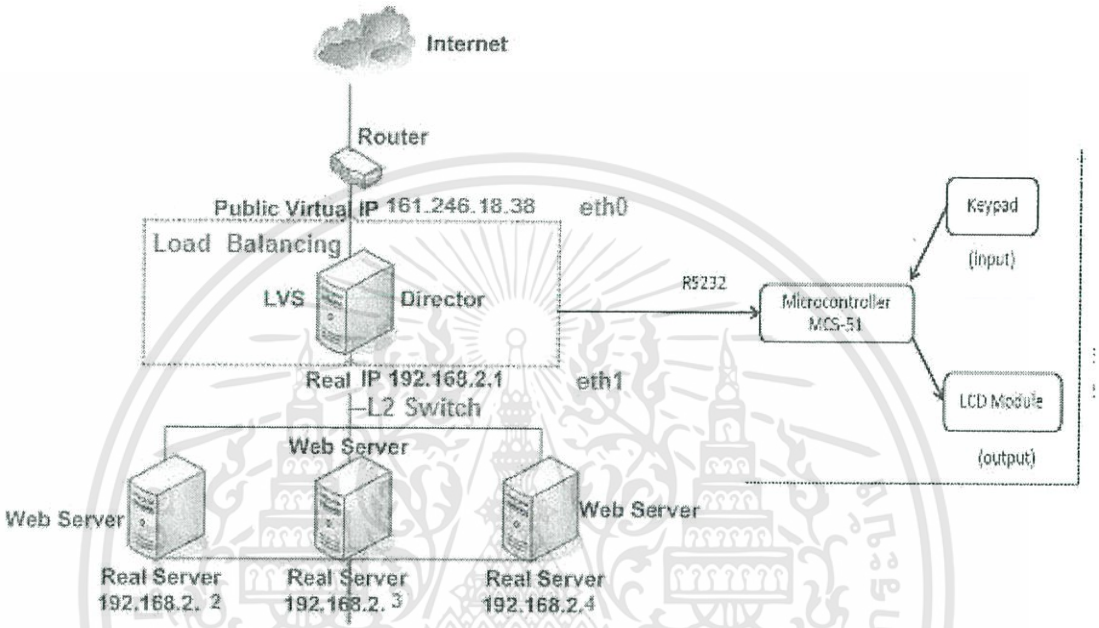
1) โหลดบาลานเซิร์ฟเวอร์ จำนวน 1 เครื่อง
2) คอมพิวเตอร์เว็บเซิร์ฟเวอร์ จำนวน 3 เครื่อง
3) คอมพิวเตอร์ไคลแอนต์ จำนวน 3 เครื่อง
4) ไมโครคอนโทรลเลอร์ MCS-51 จำนวน 1 เครื่อง
5) แอลซีดีโมดูล (LCD Module) จำนวน 1 จอ
6) คีย์แพด (Keypad) จำนวน 1 อัน
7) Switch L2 จำนวน 1 ตัว

3.2.2 ซอร์ฟแวร์

1) โปรแกรม Virtual Box
2) ระบบปฏิบัติการ Linux Ubuntu Server
3) โปรแกรม AppServ 2.5.10
4) ระบบปฏิบัติการ Windows 7

3.3 การจัดเก็บผลการทดลอง

การต่ออุปกรณ์ในการทดลองจะเป็นดังแผนภาพ ดังนี้

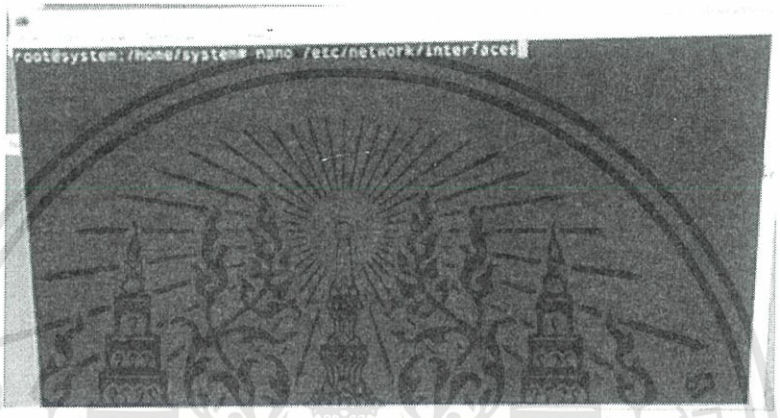


รูปที่ 3.25 แผนภาพการทดลองของระบบโหลดบาลานซ์

ในการทดลอง จะพิจารณาได้จากรูป 3.25 ซึ่งเป็นแผนผังการทำงานของระบบทั้งหมดที่ได้ออกแบบไว้ โดยในการทดลองจะกำหนดให้ใช้เครื่องแม่ข่าย (Server) จำนวน 3 เครื่องไว้รองรับผู้ใช้งาน (Users) และเครื่อง Client จำนวน 3 เครื่อง โดยเครื่องแม่ข่ายจะใช้คอมพิวเตอร์โน๊ตบุคในการจำลองโดยใช้โปรแกรม AppServ 2.5.10 ส่วนเครื่อง Client จะใช้ระบบปฏิบัติการ Windows 7 สำหรับโหลดบาลานซ์จะทำการควบคุมโดยแอพพลิเคชั่นที่ได้ออกแบบไว้ ซึ่งจะรันบนเครื่องแม่ข่ายจริงภายใต้ระบบปฏิบัติการลินุกซ์ ซึ่งจะใช้แอพพลิเคชั่นที่ได้ทำการออกแบบไว้ใช้ในการควบคุมการทำงานของโหลดบาลานซ์และส่วนที่สองได้ทดลองใช้ฮาร์ดแวร์ในการจัดการการใช้งานโหลดบาลานซ์เซิร์ฟเวอร์

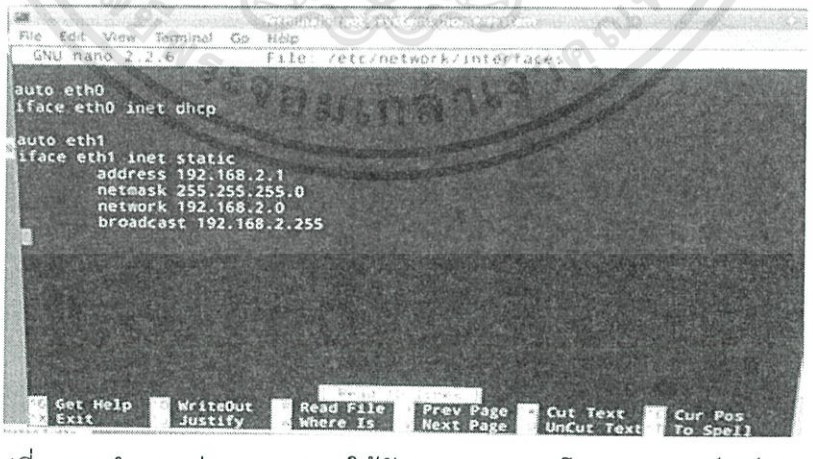
3.3.1 การทดสอบสถานะของโหลดบาลานซ์เซิร์ฟเวอร์

ก่อนทำการทดลองเราจะตั้งค่าพอร์ต eth0 และ eth1 ของโหลดบาลานซ์เซิร์ฟเวอร์ เพื่อทำการกำหนด IP Address ของเครื่องแม่ข่ายโหลดบาลานซ์ (Virtual IP) และ IP address ของ web server (Real IP) โดยป้อนคำสั่งผ่านหน้าต่าง Terminal ด้วยคำสั่ง "nano/etc/network/interfaces"



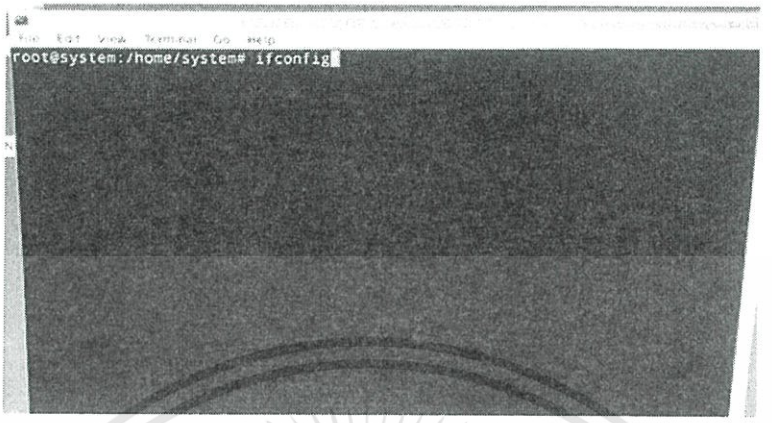
รูปที่ 3.26 การใช้คำสั่งเพื่อตั้งค่าขา eth0 และ eth1 ของโหลดบาลานซ์เซิร์ฟเวอร์

ทำการตั้งค่าขา eth0 และ eth1 ซึ่งเราจะกำหนด IP Address ในการทำการ Access เข้ามาของเครื่อง Client ซึ่งจะเป็น Virtual IP เป็นแบบ DHCP ซึ่งเป็นการรับค่า IP Address ที่สุ่มมา คือ 161.246.18.38 และขา eth1 เราตั้งในวงของ 192.168.2.1 ดังรูป



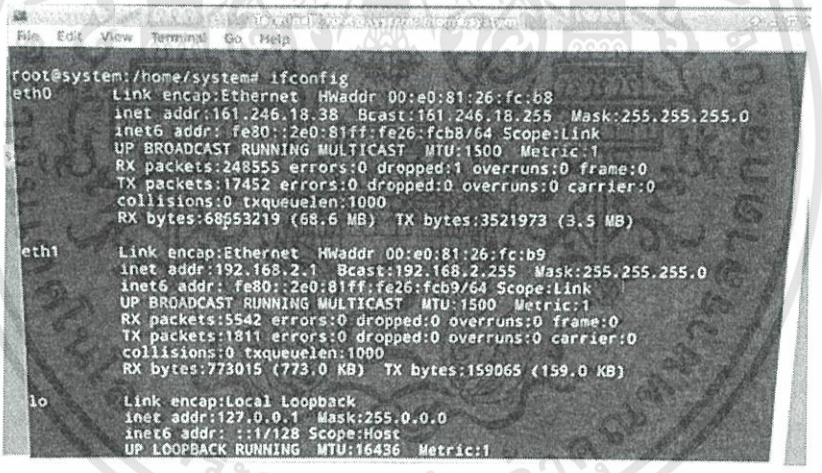
รูปที่ 3.27 กำหนดค่า Virtual IP ให้กับขา eth1 ของโหลดบาลานซ์เซิร์ฟเวอร์

เมื่อทำการตั้งค่าเสร็จเราจะทำการเช็คได้ด้วยคำสั่ง ipconfig



รูปที่ 3.28 การใช้คำสั่งในการเช็คขา eth0 และ eth1 ของโหลดบาลานซ์

จากรูปเราจะเห็นว่า ขา eth0 ถูกตั้งค่าให้เป็น

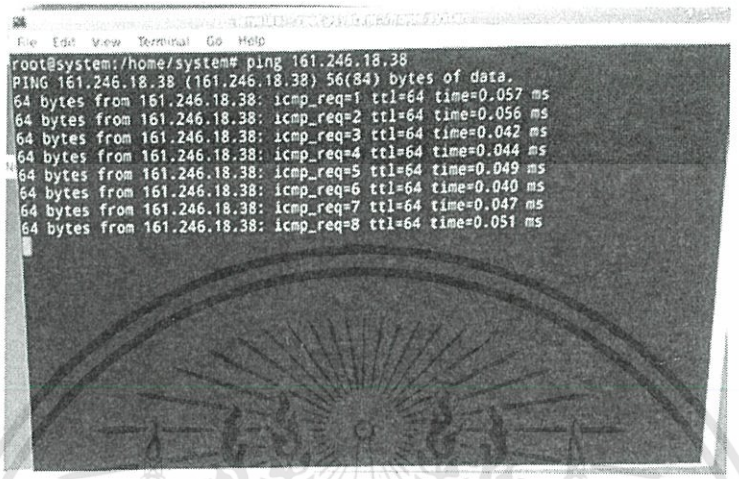


รูปที่ 3.29 แสดงผลของการ config ขา eth0 และ eth1

ขา eth0 มี IP คือ 161.246.18.38

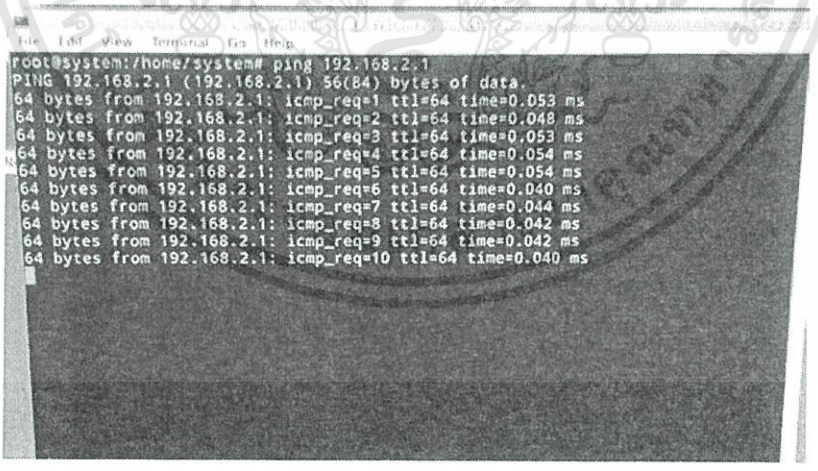
ขา eth1 มี IP คือ 192.168.2.1

หลังจากทำการตั้งค่าขา eth0 และ eth1 เสร็จแล้ว เราจะทำการทดสอบการทำงานการส่งข้อมูลของขา eth0 โดยการ ping 161.246.18.38



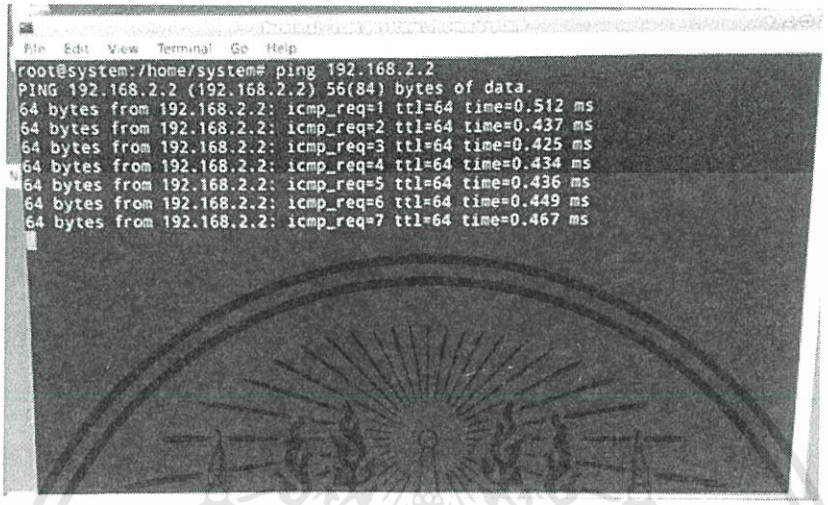
รูปที่ 3.30 การทดสอบการทำงานของเครื่องโหลดบาลานซ์เซิร์ฟเวอร์

และทำการทดสอบการทำงานของขา eth 1 และเครื่อง Real Server โดยการ Ping 192.168.2.1 eth1 real IP



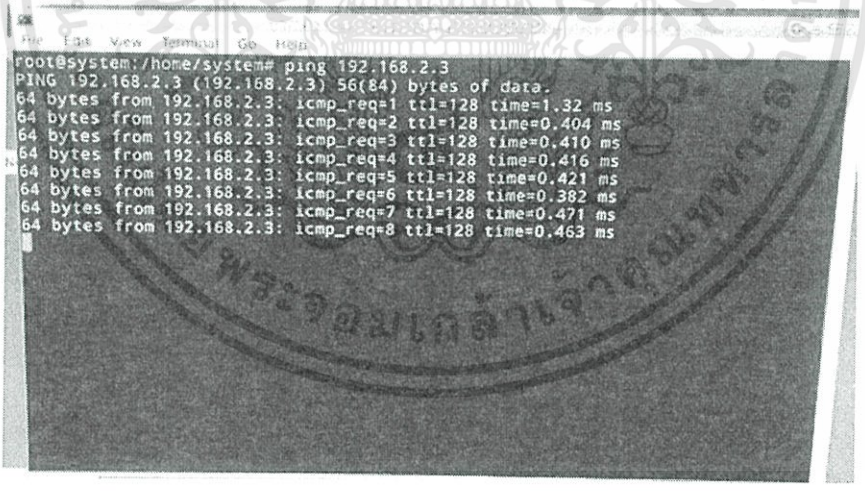
รูปที่ 3.31 การทดสอบการทำงานของเครื่องโหลดบาลานซ์เซิร์ฟเวอร์

การทดสอบการทำงานของเครื่อง web server1 โดยการ Ping 192.168.2.2 eth1ของ Real Server1



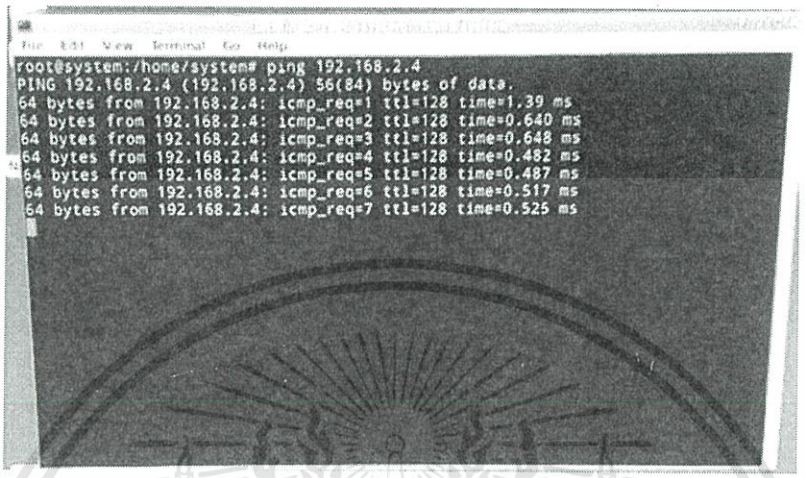
รูปที่ 3.32 การทดสอบการทำงานของเครื่อง web server1

การทดสอบการทำงาน web server 2 โดยการ Ping 192.168.2.3 eth1ของ Real Server2



รูปที่ 3.33 การทดสอบการทำงานของเครื่อง web server 2

การทดสอบการทำงาน web server 3 โดยการ Ping 192.168.2.4 eth1ของ Real Server3



รูปที่ 3.34 การทดสอบการทำงานของเครื่อง web server 3

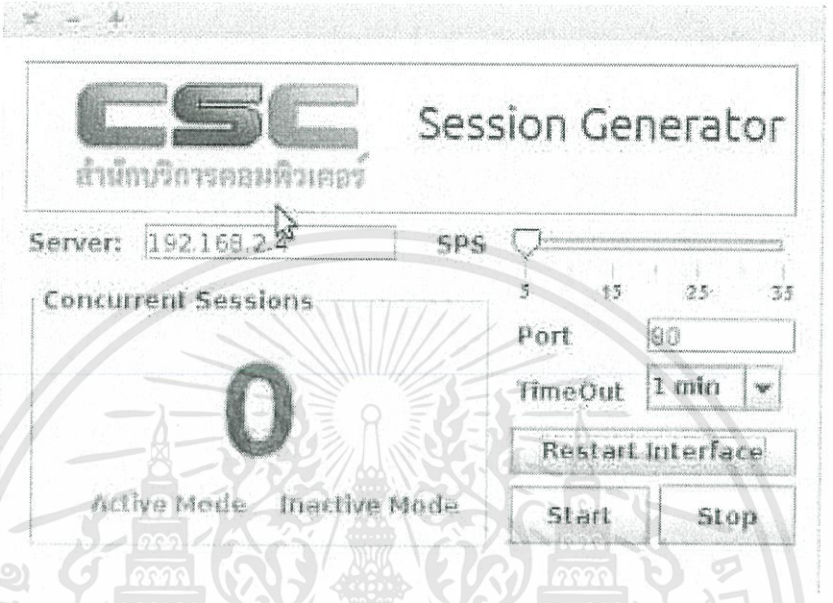
3.3.2 การจัดเก็บผลการทำงานของ Panel Controller

Panel Controller ถูกใช้เพื่อป้อนค่าพารามิเตอร์ต่างๆที่จำเป็นให้แก่โหลดบาลานซ์เซิร์ฟเวอร์ซึ่งมีหลักการในการทำงานเหมือนกันกับการทำงานในส่วนของแอพพลิเคชั่น เพื่อควบคุมการกระจายภาระงานของโหลดบาลานซ์ เพียงแต่ในส่วนของ Panel Controller นี้จะเป็นการป้อนค่าพารา มิเตอร์ต่างๆลงใน Panel Controller โดยใช้ไมโครคอนโทรลเลอร์โดยรับค่าอินพุตจากคีย์แพด (Keypad) และแสดงผลผ่าน LCD Module

3.3.3 การจัดเก็บผลการกระจายภาระงานของโหลดบาลานซ์เซิร์ฟเวอร์

การจัดเก็บผลการกระจายภาระงานของโหลดบาลานซ์เซิร์ฟเวอร์ ซึ่งในการทดลองได้ทำการจำลอง Web Server ขึ้นมา 3 เครื่อง เพื่อรองรับภาระงานจากผู้ใช้บริการ (Users) โดยการทดลองได้ทำการจำลองการใช้งานของ (Users) และเครื่อง Web Server โดยใช้โปรแกรม Session Generator เพื่อจำลองการเข้าใช้งาน (Access) ของผู้ใช้บริการ (Users) และแสดงปริมาณภาระ

งานที่ Web Server แต่ละเครื่องได้รับเข้ามาตามวิธีการกระจายภาระงานทั้ง 4 วิธีและแบบวิธีที่ไม่มีโหลดบาลานซ์ ซึ่งวิธีการใช้งานโปรแกรม Session Generator มีดังนี้



รูปที่ 3.35 หน้าเริ่มต้นของโปรแกรม Session Generator ของสำนักบริการคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

- Server : เป็นช่องที่ใช้ป้อนหมายเลข IP Address
- สไลด์บาร์ (SPS) : ใช้สำหรับกำหนดอัตราการ Generate มีหน่วยเป็น Session per Second
- Port : ใช้สำหรับกำหนดพอร์ตที่ต้องการจะ Generate ไปให้
- TimeOut : ใช้กำหนดเวลาไทม์เอาท์ของ Session (โดย Session สามารถไทม์เอาท์ก่อนเวลาที่กำหนดได้ แต่ถ้ายังไม่มีการไทม์เอาท์ระบบก็จะทำการไทม์เอาตามเวลาที่กำหนดได้)
- Restart Interface : ใช้ในการ Start และ Stop Interface
- Start : เริ่มการ Generate
- Stop : หยุดการ Generate
- Concurrent Session : แสดงปริมาณของ Session ที่ถูก Generate ออกไป โดยโปรแกรม Session Generator จะทำการนับปริมาณของ Session ที่อยู่ในระบบโดยนับ Session ที่อยู่ในสถานะ ESTABLISHED เท่านั้น และปริมาณที่แสดงจะมีค่าไม่คงที่ขึ้นอยู่กับปริมาณ Session ที่เข้า

มาและไหลออกไปจากระบบ เปรียบเทียบได้กับปริมาณของน้ำในถังน้ำที่มีปริมาณไม่คงที่เนื่องจากจะมีน้ำไหลเข้าและไหลออกตลอดเวลา ซึ่งตัวเลขที่แสดงมี 2 สถานะคือ

- Active Mode (จะปรากฏเป็นตัวเลขสีน้ำเงิน)
- inactive Mode (จะปรากฏเป็นตัวเลขสีแดง)

บทที่ 4

ผลการทดลอง

4.1 ผลการทดลอง

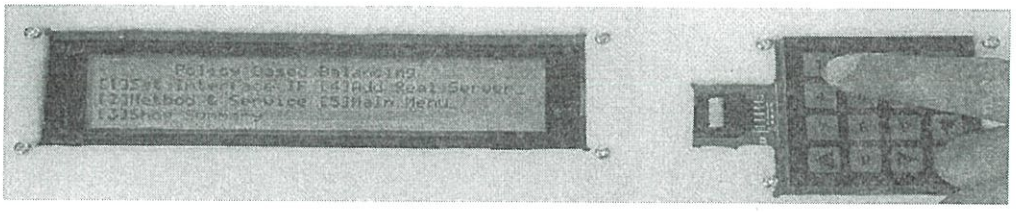
4.1.1 การทดลองในส่วนของ Panel Controller

การทดลองในส่วนของ Panel Controller นี้ มีหลักการในการทดลองเหมือนกันกับการทดลองในส่วนของแอพพลิเคชั่น เพียงแต่ว่าในส่วนของ Panel Controller นี้จะเป็นการป้อนค่าพารามิเตอร์ต่างๆลงใน Panel Controller โดยใช้ไมโครคอนโทรลเลอร์ โดยรับค่าอินพุตจากคีย์แพด (Keypad) และแสดงผลผ่าน LCD Module ซึ่งผู้ทดลองได้อธิบายเมนูบนหน้าจอ LCD Module และการป้อนค่าพารามิเตอร์ต่างๆรวมถึงผลการทดลองดังรูปต่อไปนี้

รูปที่ 4.1 Main Menu ของ Panel Controller

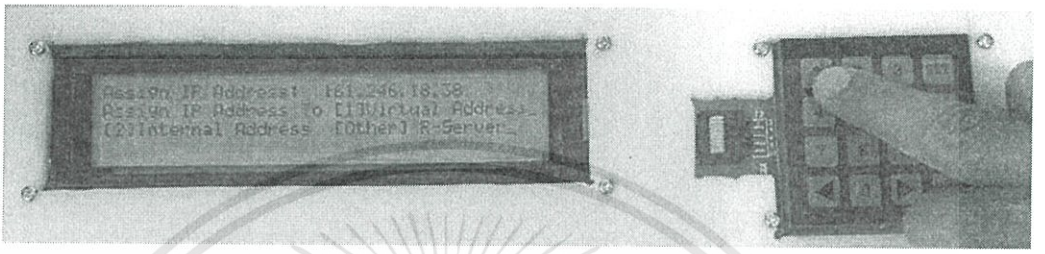
4.1.1.1 ขั้นตอนการกำหนดค่าพารามิเตอร์

ซึ่งในหน้า Main Menu ประกอบไปด้วย [1] Show System Info, [2] Balancing Policy, [3] Chassic Control, [4] Show policies หลังจากนั้นเลือกกด [2] Balancing Policy ก็จะเข้าไปหน้า Policy Based Balancing เพื่อเป็นการเข้าไปป้อนค่าพารามิเตอร์ต่างๆที่จำเป็นในการทำการกระจายโหลด โดยเมื่อกดเข้าไปแล้วจะได้หน้าเมนูดังรูป



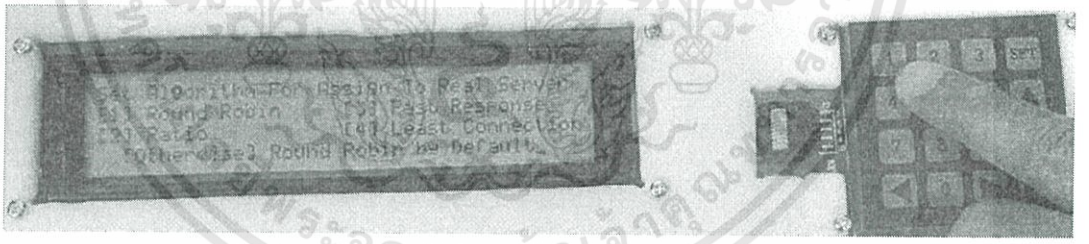
รูปที่ 4.2 Menu Policy Based Balancing ของ Panel Controller

ซึ่งในหน้า Policy Based Balancing ประกอบไปด้วย [1] Set Interface IP, [2] Method&Service, [3] Show Summary, [4] Add Real Server, [5] Main Menu หลังจากนั้นกดเลือกที่[1] Set Interface IP เพื่อเข้าไปกำหนดค่า Virtual IP เป็น 161.246.18.38 เสร็จแล้วกดเลือกที่ [1] Virtual Address ดังรูป



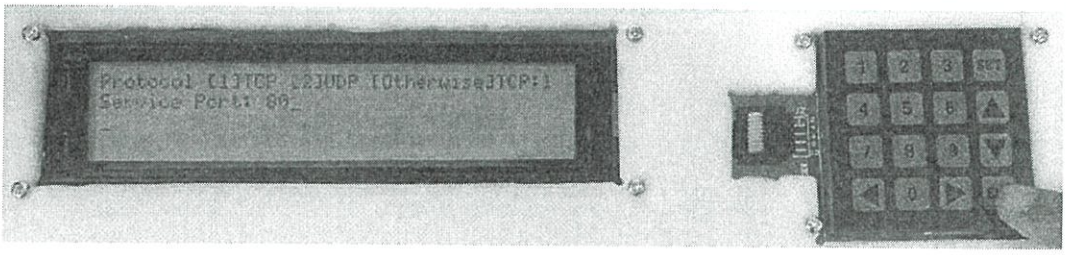
รูปที่ 4.3 การกำหนดค่า Virtual IP ของ Panel Controller

เมื่อเสร็จแล้วระบบจะกลับไปที่หน้า Policy Based Balancing และเลือกกดที่ [2] Method&Service ซึ่งเมื่อกดเข้าไปจะได้เมนู Set Algorithm For Assign to Real Server ซึ่งจะเป็นเมนูที่แสดงวิธีการการกระจายโหลดแบบต่างๆคือ [1] Round Robin, [2] Ratio, [3] Fast Response, [4] Least Connection, [Otherwise] Round Robin by Default ซึ่งในที่นี้ผู้ทดลองได้เลือกวิธีการการกระจายโหลดเป็นแบบ Round Robin ดังรูป



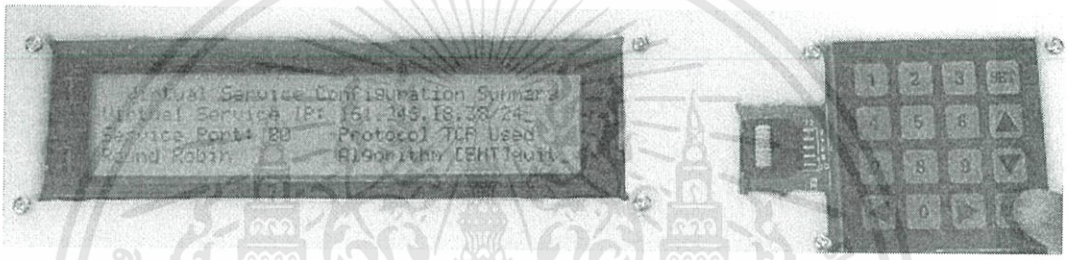
รูปที่ 4.4 เมนูในการเลือกวิธีการการกระจายโหลด ของ Panel Controller

เมื่อเสร็จแล้วจะเข้าไปในหน้าที่เลือก Protocol โดยมี Protocol ให้เลือกคือ [1] TCP, [2] UDP, [Otherwise] TCP ซึ่งการทดลองผู้ทดลองได้เลือก Protocol TCP โดยเลือก [1] TCP และเลือก Service Port เป็น Port 80 ดังรูป



รูปที่ 4.5 เมนูกำหนด Protocol และ Service Port ของ Panel Controller

หลังจากนั้นจะได้หน้าเมนูแสดงผลการกำหนดค่า Virtual Service IP ที่ได้กำหนดไปก่อนหน้านี้ดังรูป



รูปที่ 4.6 ผลการกำหนดค่า Virtual Service IP ของ Panel Controller

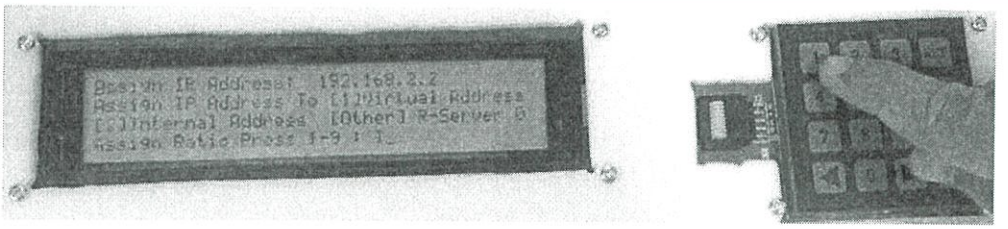
เมื่อทำการกำหนดค่าของ Virtual Service IP เสร็จเรียบร้อยแล้วระบบจะกลับไปสู่หน้า Policy Based Balancing เพื่อไปทำการกำหนดค่า Real Server โดยเลือกที่ [4] Add Real Server ซึ่งผู้ทำการทดลอง ได้กำหนดค่า Real Server IP เป็น

192.168.2.2 (Server1) ; Ratio : 1

192.168.2.3 (Server2) ; Ratio : 1

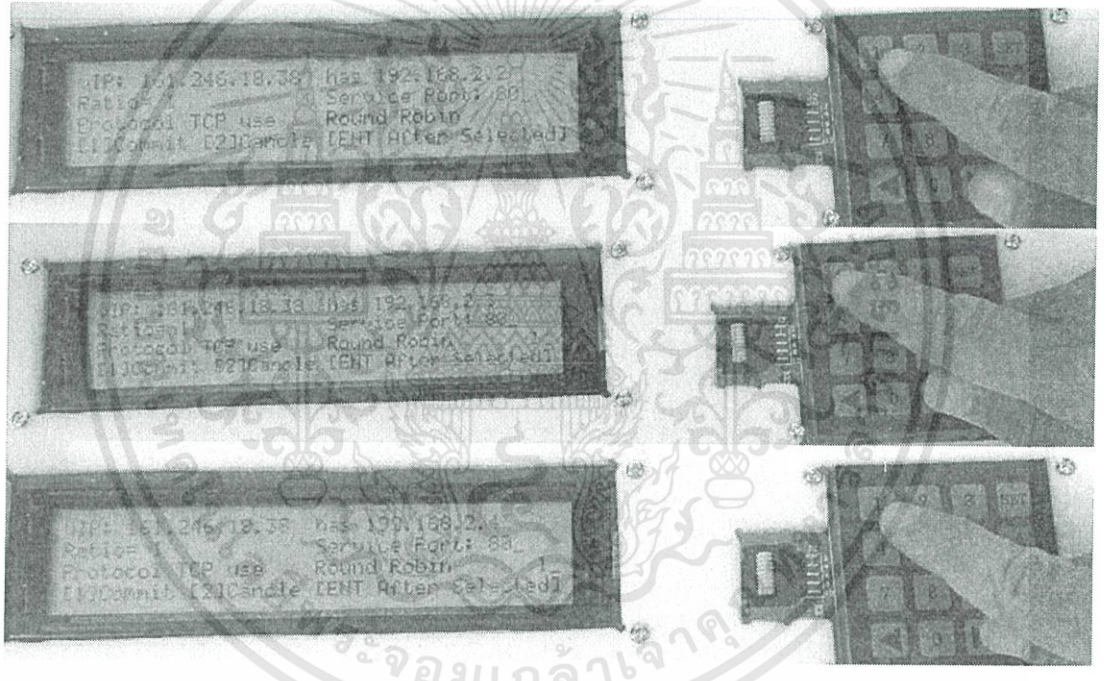
192.168.2.4 (Server3) ; Ratio : 1

โดยทำการกำหนด Real Server IP ที่ล่ะค่าเมื่อกำหนดเสร็จแล้วเลือกที่ [Other] R-Server เพื่อไปกำหนดค่า Real Server IP ตัวต่อไป หลังจากนั้นก็ทำการกำหนด Ratio ที่ Assign Ratio Press ให้เป็น 1 ดังรูป



รูปที่ 4.7 การกำหนดค่า Real Service IP ของ Panel Controller

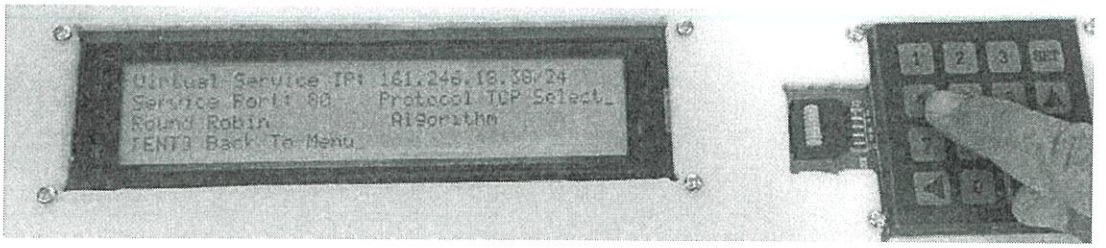
เมื่อกำหนดค่า Real Service IP เสร็จทั้งสามแล้ว ระบบก็จะแสดงผลการกำหนดค่าของ Real Service IP ดังรูป



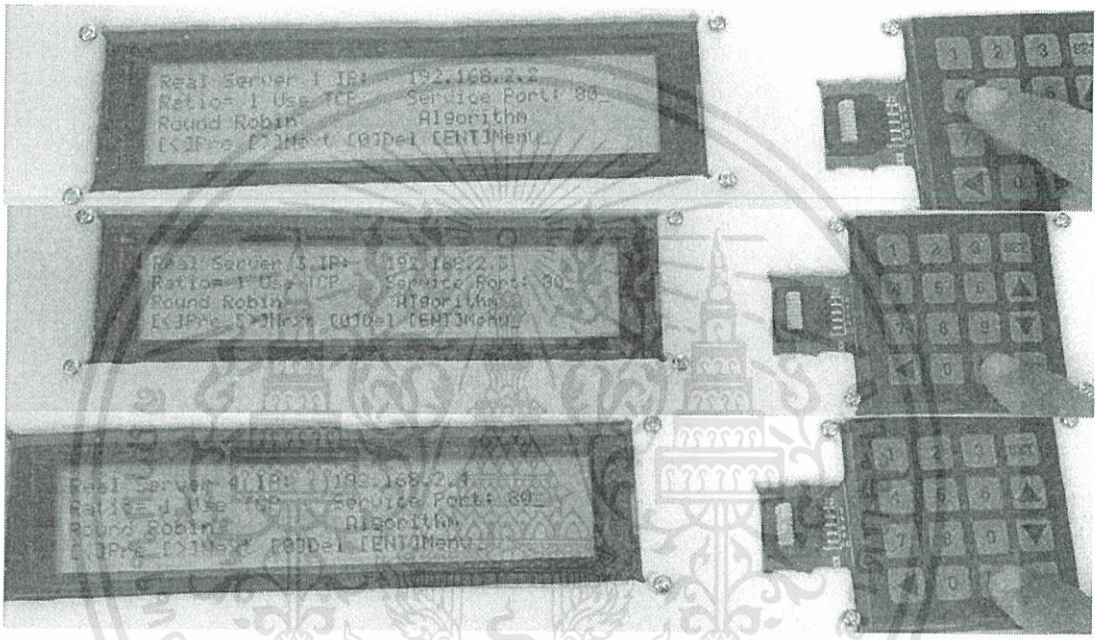
รูปที่ 4.8 ผลการกำหนดค่า Real Service IP ของ Panel Controller

4.1.1.2 ผลการทำงานของ Panel Controller

เมื่อเสร็จแล้วทำการเลือกที่ [1] Commit และ [ENT] After Selected เพื่อยืนยันให้ระบบทำการกระจายโหลด ซึ่งผลของการกระจายโหลดสามารถตรวจสอบได้โดย กลับไปที่หน้า Main Menu และเลือกที่[4] Show policies ระบบจะทำการแสดงสถานะการทำงานของ Virtual Service และ Real Server ดังรูป



รูปที่ 4.9 สถานะการทำงานของ Virtual Service



รูปที่ 4.10 สถานะการทำงานของ Real Server

4.1.2 ผลการกระจายภาระงานของโหลดบาลานซ์เซิร์ฟเวอร์

ในการทดลองผู้ทดลองได้กำหนดอัตราการ Generate ที่ 5 Sessions per Second ทั้งนี้เพื่อให้การแสดงผลเป็นที่เข้าใจง่ายยิ่งขึ้น เนื่องจากจะสามารถเปรียบเทียบปริมาณของ Sessions ที่ Web Server แต่ละตัวได้รับได้ชัดเจน และกำหนดเวลา Time-out ที่ 1 นาที

4.1.2.1 การกระจายงานแบบ Round Robin

เราจะทำการทดสอบการทำงานของแอพพลิเคชั่นโดยทำการกระจายโหลดแบบ Round Robin ซึ่งเราจะป้อนค่าพารามิเตอร์ได้ดังนี้

Virtual IP : 161.246.18.249

Protocol : TCP

Port : 80

Real IP : 192.168.2.2 (Server1)

192.168.2.3 (Server2)

192.168.2.4 (Server3)

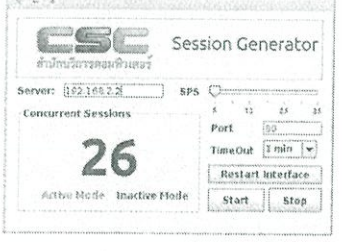
Ratio : 1 (กำหนดให้เป็น 1 เนื่องจากเป็นการกระจายโหลด แบบ Round Robin)

Method : Round Robin

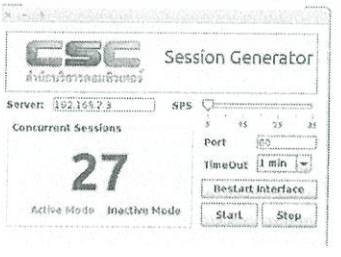
หลังจากป้อนคำสั่งแล้ว เราจะทำการ Start Service ก่อนเพื่อเริ่มการทำงานของระบบ โดยคลิกที่ปุ่ม Start หลังจากนั้นเราจะทำการป้อนคำสั่งโดยการเลือกปุ่ม OK แอพพลิเคชั่นก็จะทำการป้อนคำสั่งเข้าไปในระบบ โดยผลการทดลองเป็นดังนี้

รูปที่ 4.11 ปริมาณ Session ที่ถูก Generate โดยผู้ใช้บริการ

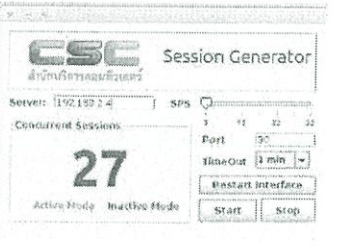
Server 1 (192.168.2.2)



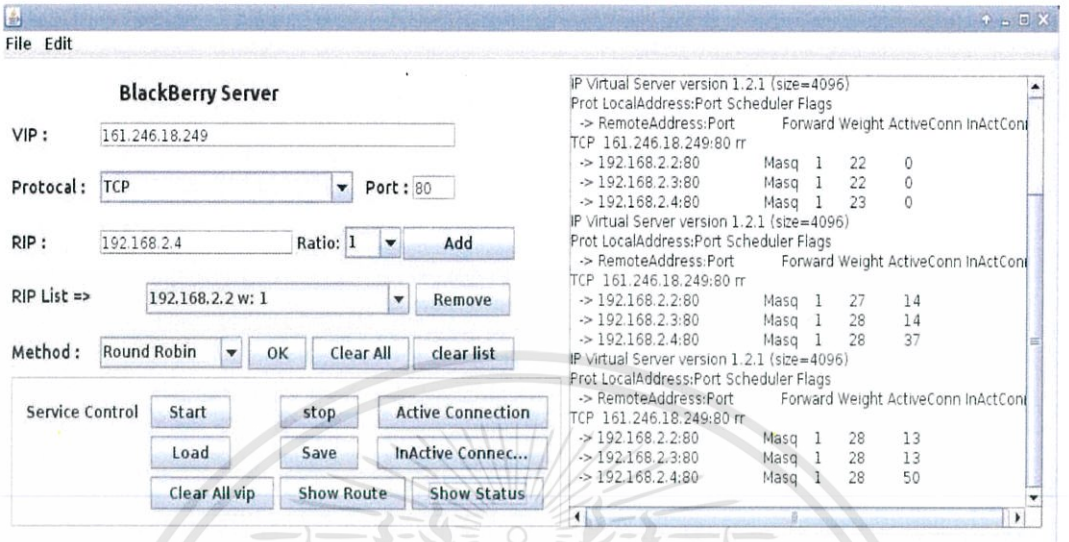
Server 2 (192.168.2.3)



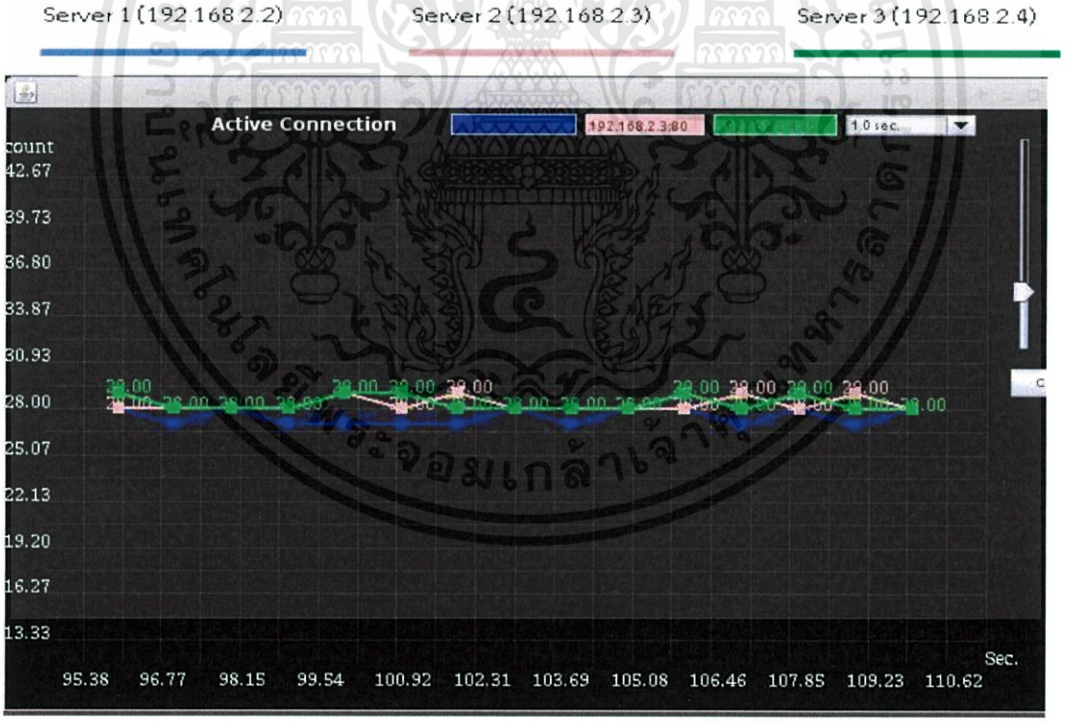
Server 3 (192.168.2.4)



รูปที่ 4.12 ปริมาณ Session ที่ Web server แต่ละตัวได้รับด้วยวิธี Round Robin



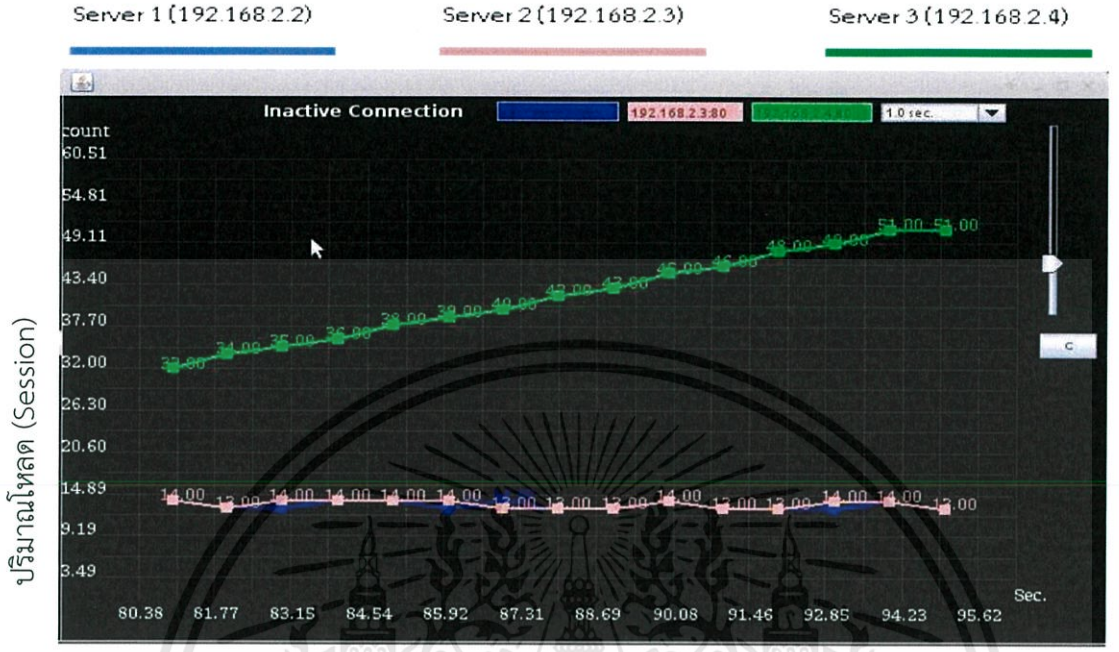
รูปที่ 4.13 กรอบแสดงสถานะปริมาณ Session ที่ Web server แต่ละตัวได้รับผ่านหน้าแอพพลิเคชั่น



ปริมาณโหลด (Session)

เวลา (วินาที)

รูปที่ 4.14 กราฟแสดงปริมาณ Session แบบ Real time ของสถานะ Active connection



รูปที่ 4.15 กราฟแสดงปริมาณ Session แบบ Real time ของสถานะ Inactive connection

จากหน้าโปรแกรม Session Generator และกรอบแสดงสถานะเราจะเห็นว่าปริมาณโหลดหรือ Session ที่ Web Server ทั้ง 3 ตัวได้รับมีปริมาณเท่ากันหรือใกล้เคียงกันตามวิธีการกระจายภาระงานแบบ Round Robin ทั้งนี้จากรูปในการทดลองมีปริมาณของโหลดไม่เท่ากันเนื่องจากระบบมีการกระจายโหลดอยู่ตลอดเวลา และจากกราฟ Active Connection จะแสดงปริมาณโหลดหรือ Session ที่มีสถานะการทำงานเสร็จสิ้นกระบวนการคือมีการรับโหลดเข้ามาแล้วส่งออกไปยังผู้ใช้บริการโดยสมบูรณ์ ซึ่งกราฟจะแสดงปริมาณของโหลดในปริมาณที่เท่ากันหรือใกล้เคียงกันซึ่งเป็นไปตามวิธีการกระจายภาระงานแบบ Round Robin และกราฟ Inactive Connection จะแสดงปริมาณโหลดหรือ Session ที่มีสถานะอยู่ในระหว่างรอส่งออกไปยังผู้ใช้บริการและรอเวลา Time-out ซึ่งจากการทดลองสรุปได้ว่า เมื่อผู้ใช้บริการทำการ Access เข้ามาใช้งาน Service แล้ว ระบบจะทำการกระจายโหลดไปยัง Web Server ในอัตราส่วนที่เท่าๆกัน เป็นไปตามวิธีการกระจายภาระงานแบบ Round Robin

4.1.2.2 การกระจายงานแบบ Weighted Round Robin

เราจะทำการทดสอบการทำงานของแอพพลิเคชั่นโดยทำการกระจายโหลดแบบ Weighted Round Robin ซึ่งเราจะป้อนค่าพารามิเตอร์ได้ดังนี้

Virtual IP : 161.246.18.249

Protocol : TCP

Port : 80

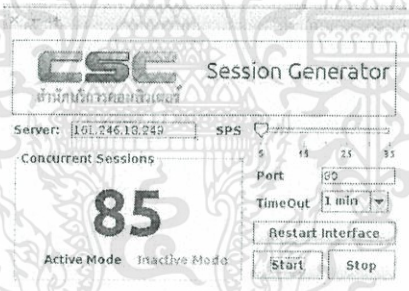
Real IP : 192.168.2.2 (Server1) ; Ratio : 1

192.168.2.3 (Server2) ; Ratio : 2

192.168.2.4 (Server3) ; Ratio : 3

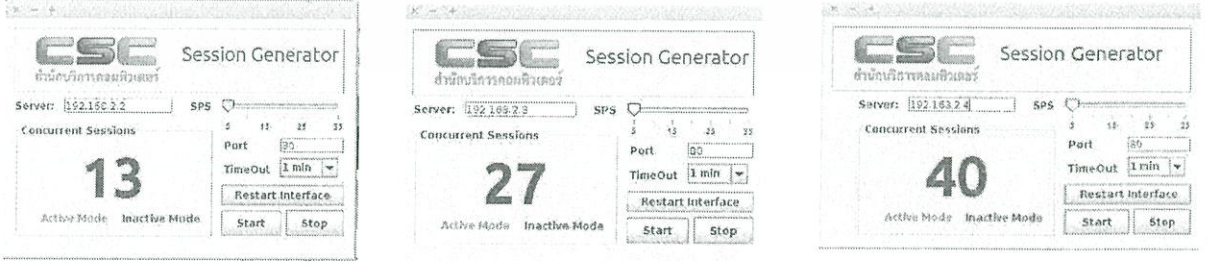
Method : Ratio

หลังจากป้อนคำสั่งแล้ว เราจะทำการ Start Service ก่อนเพื่อเริ่มการทำงานของระบบ โดยคลิกที่ปุ่ม Start หลังจากนั้นเราจะทำการป้อนคำสั่งโดยการเลือกปุ่ม OK แอพพลิเคชั่นก็จะทำการป้อนคำสั่งเข้าไปในระบบ โดยผลการทดลองเป็นดังนี้

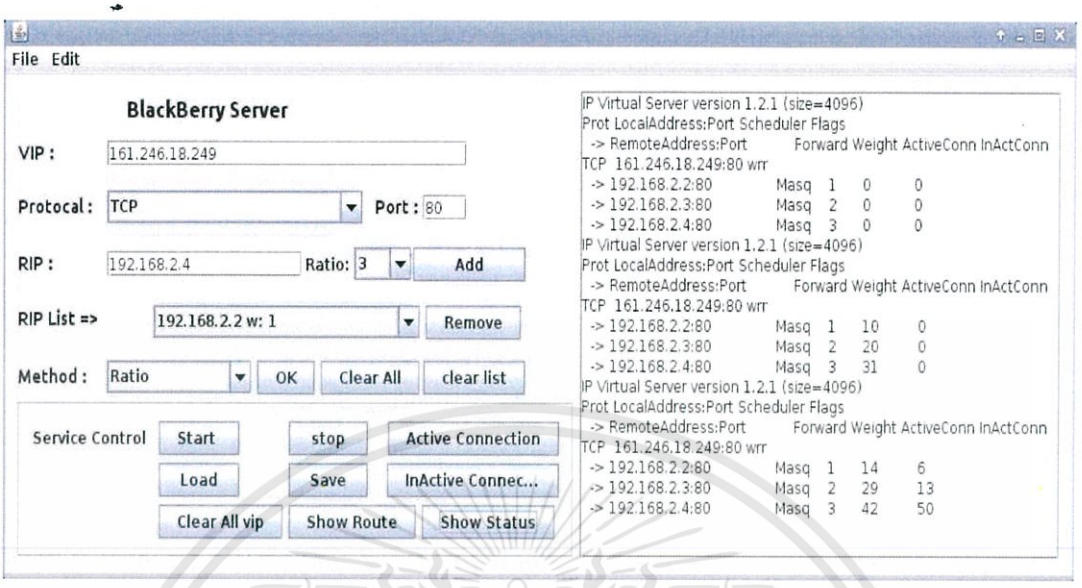


รูปที่ 4.16 ปริมาณ Session ที่ถูก Generate โดยผู้ใช้บริการ

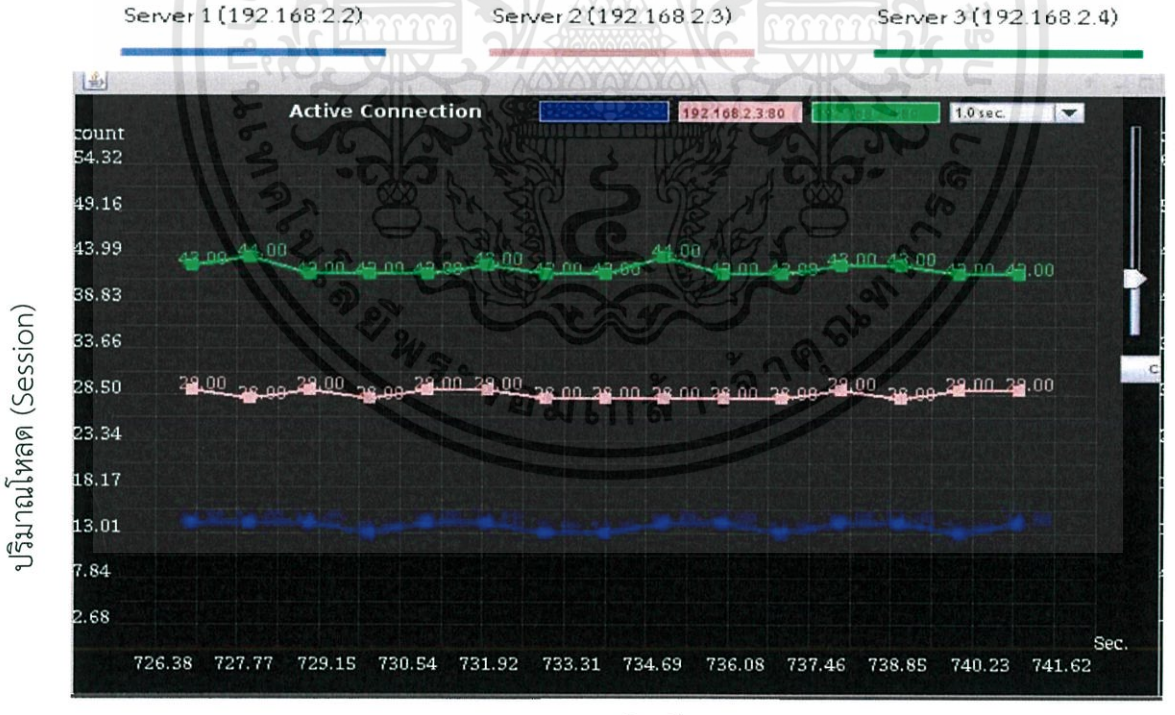
Server 1 (192.168.2.2) Server 2 (192.168.2.3) Server 3 (192.168.2.4)



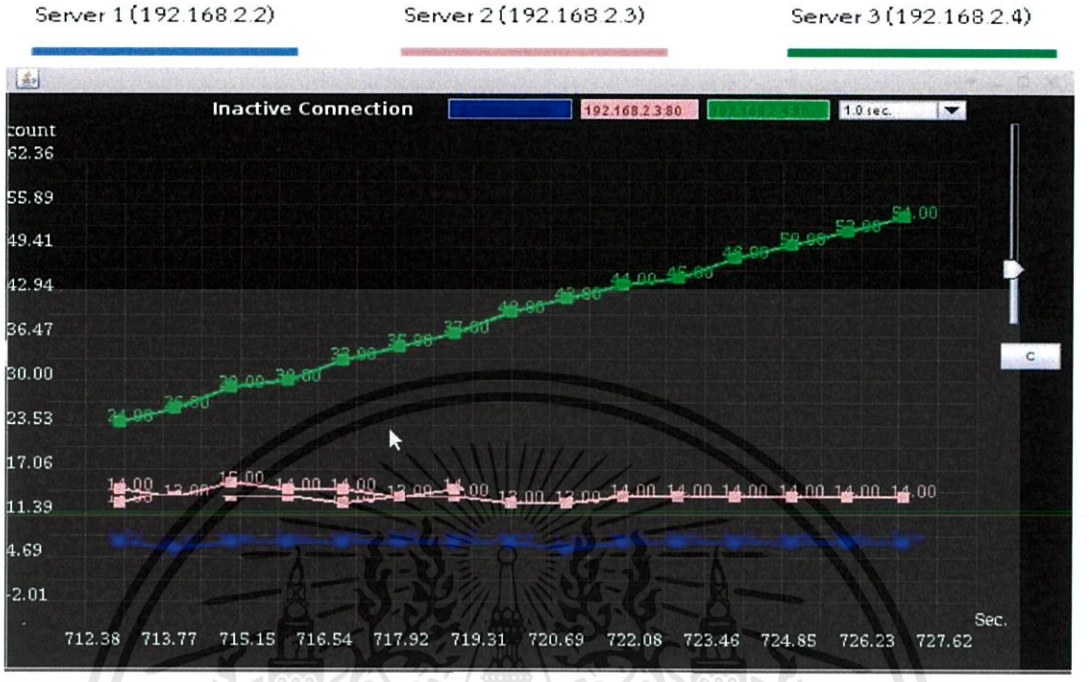
รูปที่ 4.17 ปริมาณ Session ที่ Web server แต่ละตัวได้รับด้วยวิธี Ratio



รูปที่ 4.18 กรอบแสดงสถานะปริมาณ Session ที่ Web server แต่ละตัวได้รับผ่านหน้าแอพพลิเคชั่น



รูปที่ 4.19 กราฟแสดงปริมาณ Session แบบ Real time ของสถานะ Active connection



ปริมาณโหลด (Session)

เวลา (วินาที)

รูปที่ 4.20 กราฟแสดงปริมาณ Session แบบ Real time ของสถานะ Inactive connection

จากหน้าโปรแกรม Session Generator และกรอบแสดงสถานะเราจะเห็นว่าปริมาณโหลดหรือ Session ที่ Web Server ทั้ง 3 ตัวได้รับมีอัตราส่วนเป็นไปตามที่กำหนด คือ 1:2:3 ตามวิธีการกระจายภาระงานแบบ Ratio ทั้งนี้จากรูปในการทดลองมีปริมาณของโหลดที่ Web Server แต่ละตัวได้รับ เมื่อบวกกันแล้วอาจไม่สัมพันธ์กับปริมาณของโหลดที่ผู้ใช้บริการ Generate เข้ามา เนื่องจากระบบมีการกระจายโหลดอยู่ตลอดเวลา และจากกราฟ Active Connection จะแสดงปริมาณโหลดหรือ Session ที่มีสถานะการทำงานเสร็จสิ้นกระบวนการคือมีการรับโหลดเข้ามาแล้วส่งออกไปยังผู้ใช้บริการโดยสมบูรณ์ ซึ่งกราฟจะแสดงปริมาณของโหลดในปริมาณที่แตกต่างกันเป็นไปตามอัตราส่วนที่กำหนดซึ่งเป็นไปตามวิธีการกระจายภาระงานแบบ Ratio และกราฟ Inactive Connection จะแสดงปริมาณโหลดหรือ Session ที่มีสถานะอยู่ในระหว่างรอส่งออกไปยังผู้ใช้บริการและรอเวลา Time-out ซึ่งจากการทดลองสรุปได้ว่า เมื่อผู้ใช้บริการทำการ Access เข้ามาใช้งาน Service แล้ว ระบบจะทำการกระจายโหลดไปยัง Web Server ตามอัตราส่วนที่กำหนด ซึ่งเป็นไปตามวิธีการกระจายภาระงานแบบ Ratio

4.1.2.3 การกระจายงานแบบ Fast Response

เราจะทำการทดสอบการทำงานของแอพพลิเคชั่นโดยทำการกระจายโหลดแบบ Fast Response ซึ่งเราจะป้อนค่าพารามิเตอร์ได้ดังนี้

Virtual IP : 161.246.18.249

Protocol : TCP

Port : 80

Real IP : 192.168.2.2 (Server1) ; Ratio : 1

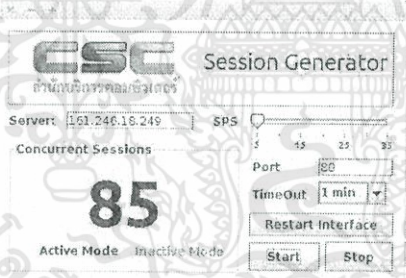
192.168.2.3 (Server2) ; Ratio : 1

192.168.2.4 (Server3) ; Ratio : 1

Ratio : จะถูกกำหนดโดย Process Crate Ratio ซึ่งจะทำการพิจารณาที่ Response time โดยในการใส่ค่าพารามิเตอร์จะตั้งค่า Ratio ให้เป็นอะไรก็ได้ไม่มีผล

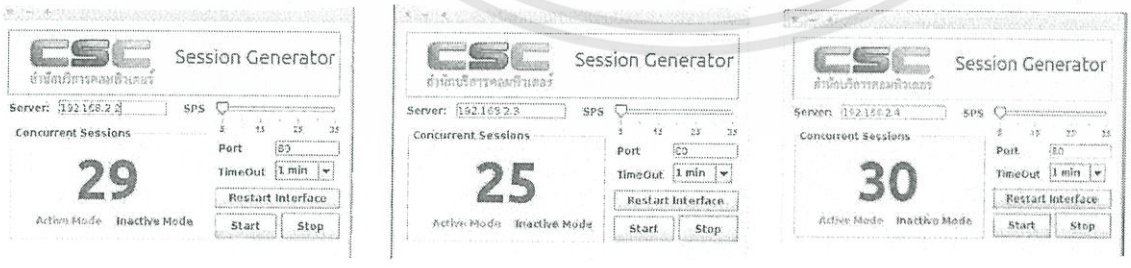
Method : Fast Response

หลังจากป้อนคำสั่งแล้ว เราจะทำการ Start Service ก่อนเพื่อเริ่มการทำงานของระบบ โดยคลิกที่ปุ่ม Start หลังจากนั้นเราจะทำการป้อนคำสั่งโดยการเลือกปุ่ม OK แอพพลิเคชั่นก็จะทำการป้อนคำสั่งเข้าไปในระบบ

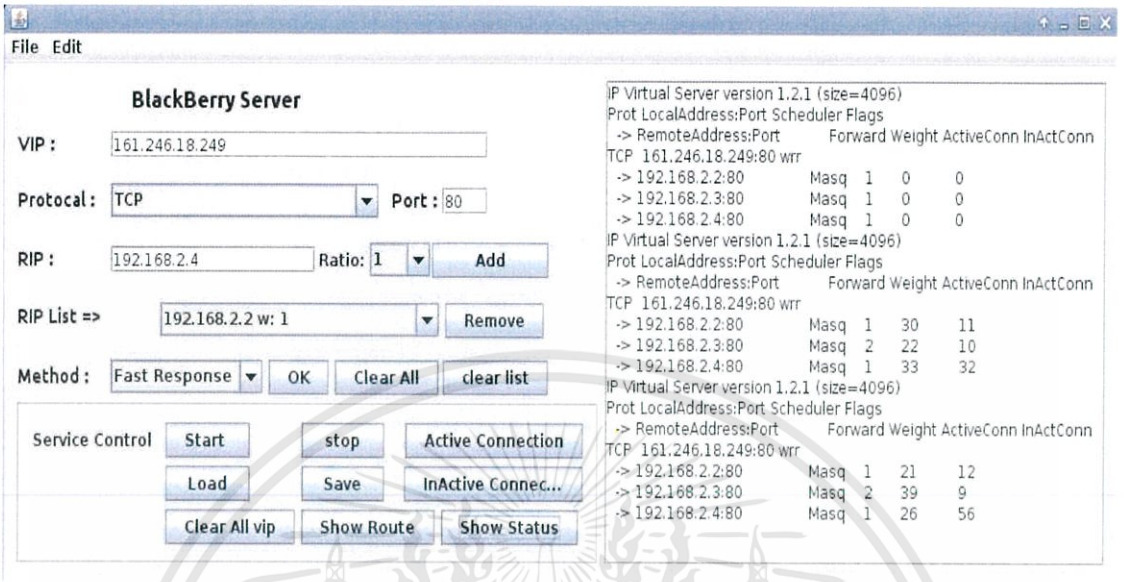


รูปที่ 4.21 ปริมาณ Session ที่ถูก Generate โดยผู้ใช้บริการ

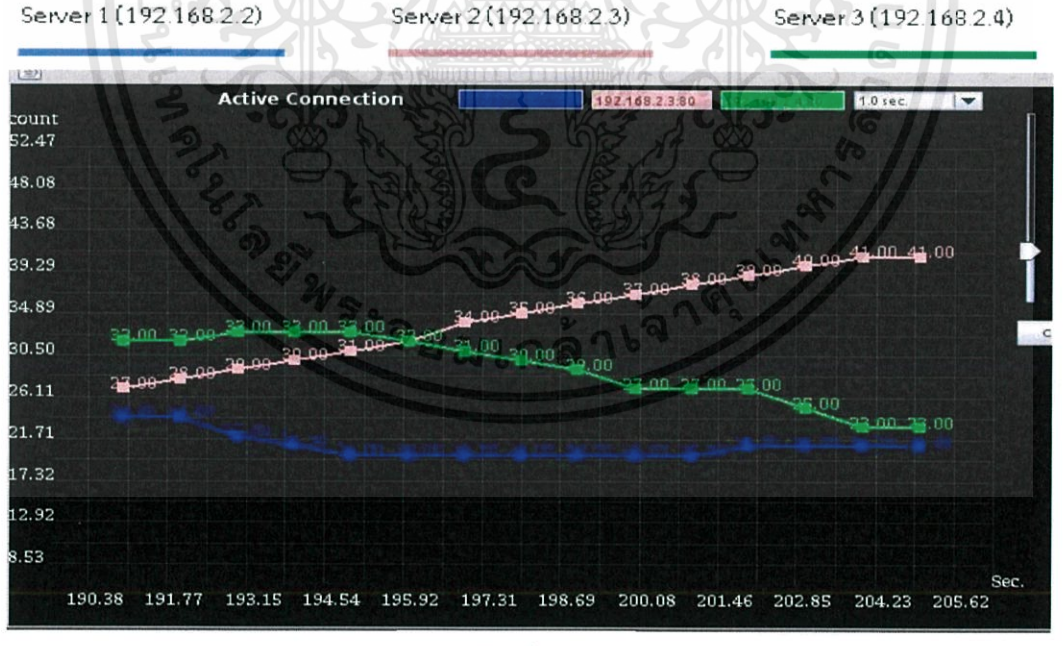
Server 1 (192.168.2.2) Server 2 (192.168.2.3) Server 3 (192.168.2.4)



รูปที่ 4.22 ปริมาณ Session ที่ Web server แต่ละตัวได้รับด้วยวิธี Fast Response



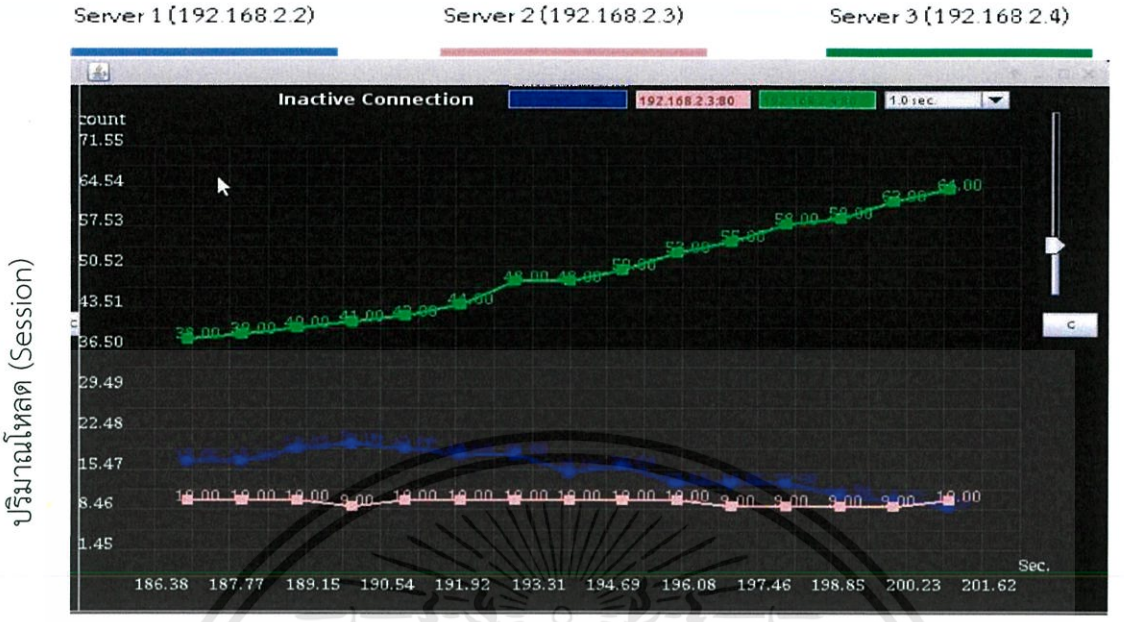
รูปที่ 4.23 กรอบแสดงสถานะปริมาณ Session ที่ Web server แต่ละตัวได้รับผ่านหน้าแอพพลิเคชั่น



ปริมาณโหลด (Session)

เวลา (วินาที)

รูปที่ 4.24 กราฟแสดงปริมาณ Session แบบ Real time ของสถานะ Active connection



รูปที่ 4.25 กราฟแสดงปริมาณ Session แบบ Real time ของสถานะ Inactive connection

จากหน้าโปรแกรม Session Generator และกรอบแสดงสถานะเราจะเห็นว่าปริมาณโหลดหรือ Session ที่ Web Server ทั้ง 3 ตัวได้รับมีปริมาณแตกต่างกันโดยขึ้นอยู่กับค่า weight ที่เปลี่ยนแปลงสลับไปมา เพราะว่าการกระจายภาระงานแบบ Fast Response จะพิจารณา Response time ของ Real Server ที่มีผลตอบสนองเร็วที่สุด ซึ่งเราได้สร้าง Process "CreateRatio" เพื่อใช้ในการตัดสินใจกำหนด Ratio ให้กับ Real Server ที่มี Response time น้อยที่สุด (มีการตอบสนองต่อระบบเร็วที่สุด) ทั้งนี้จากรูปในการทดลองมีปริมาณของโหลดที่ Web Server แต่ละตัวได้รับ เมื่อบวกกันแล้วอาจไม่สัมพันธ์กับปริมาณของโหลดที่ผู้ใช้บริการ Generate เข้ามาเนื่องจากระบบมีการกระจายโหลดอยู่ตลอดเวลา และจากกราฟ Active Connection จะแสดงปริมาณโหลดหรือ Session ที่มีสถานะการทำงานเสร็จสิ้นกระบวนการคือมีการรับโหลดเข้ามาแล้วส่งออกไปยังผู้ใช้บริการโดยสมบูรณ์ ซึ่งกราฟจะแสดงปริมาณของโหลดในปริมาณที่แตกต่างกันโดยมีการมีการเพิ่ม-ลด ของปริมาณโหลดสลับไปมา และกราฟ Inactive Connection จะแสดงปริมาณโหลดหรือ Session ที่มีสถานะอยู่ในระหว่างรอส่งออกไปยังผู้ใช้บริการและรอเวลา Time-out ซึ่งจากการทดลองสรุปได้ว่าระบบโหลดบาลานซ์ได้กระจายภาระงานเป็นไปตามวิธีการกระจายโหลดแบบ Fast Response

4.1.2.4 การกระจายงานแบบ Least Connection

เราจะทำการทดสอบการทำงานของแอพพลิเคชั่นโดยทำการกระจายโหลดแบบ Least Connection ซึ่งเราจะป้อนค่าพารามิเตอร์ได้ดังนี้

Virtual IP : 161.246.18.249

Protocol : TCP

Port : 80

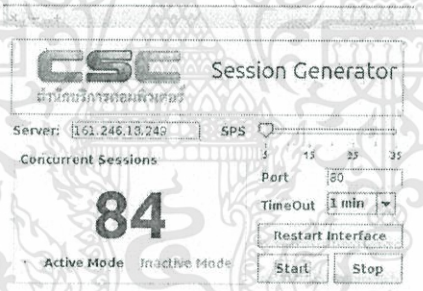
Real IP : 192.168.2.2 (Server1) ; Ratio : 1

192.168.2.3 (Server2) ; Ratio : 2

192.168.2.4 (Server3) ; Ratio : 1

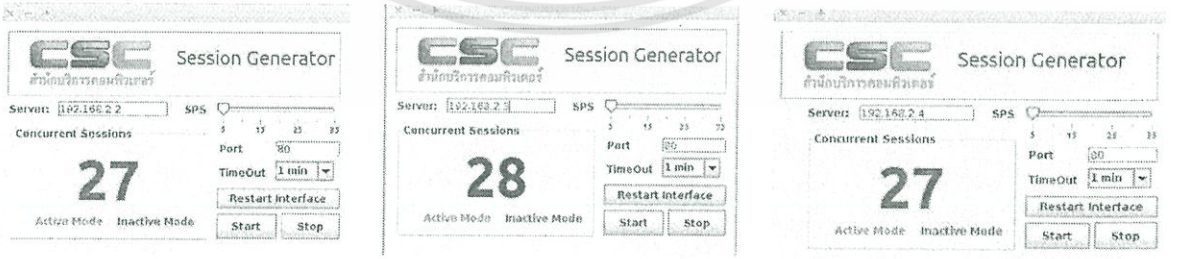
Method : Least Connection

หลังจากป้อนคำสั่งแล้ว เราจะทำการ Start Service ก่อนเพื่อเริ่มการทำงานของระบบ โดยคลิกที่ปุ่ม Start หลังจากนั้นเราจะทำการป้อนคำสั่งโดยการเลือกปุ่ม OK แอพพลิเคชั่นก็จะทำการป้อนคำสั่งเข้าไปในระบบ

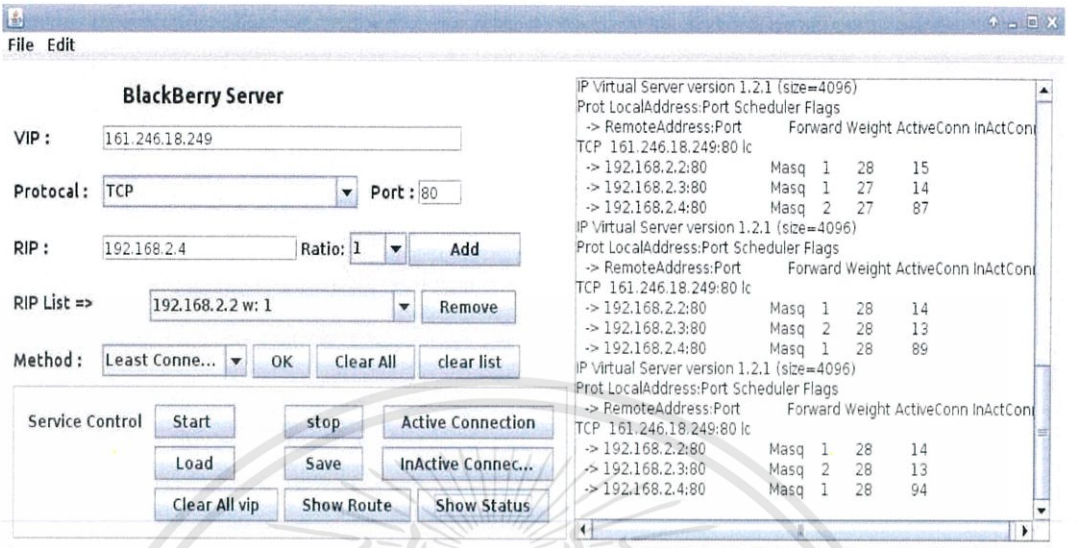


รูปที่ 4.26 ปริมาณ Session ที่ถูก Generate โดยผู้ใช้บริการ

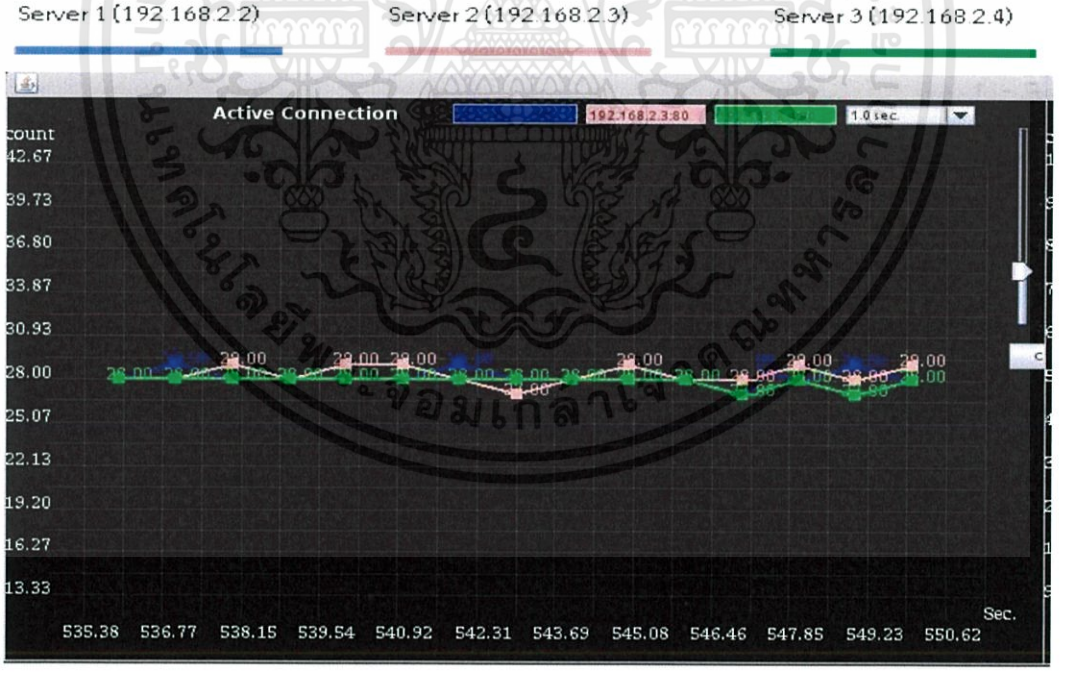
Server 1 (192.168.2.2) Server 2 (192.168.2.3) Server 3 (192.168.2.4)



รูปที่ 4.27 ปริมาณ Session ที่ Web server แต่ละตัวได้รับด้วยวิธี Least Connection



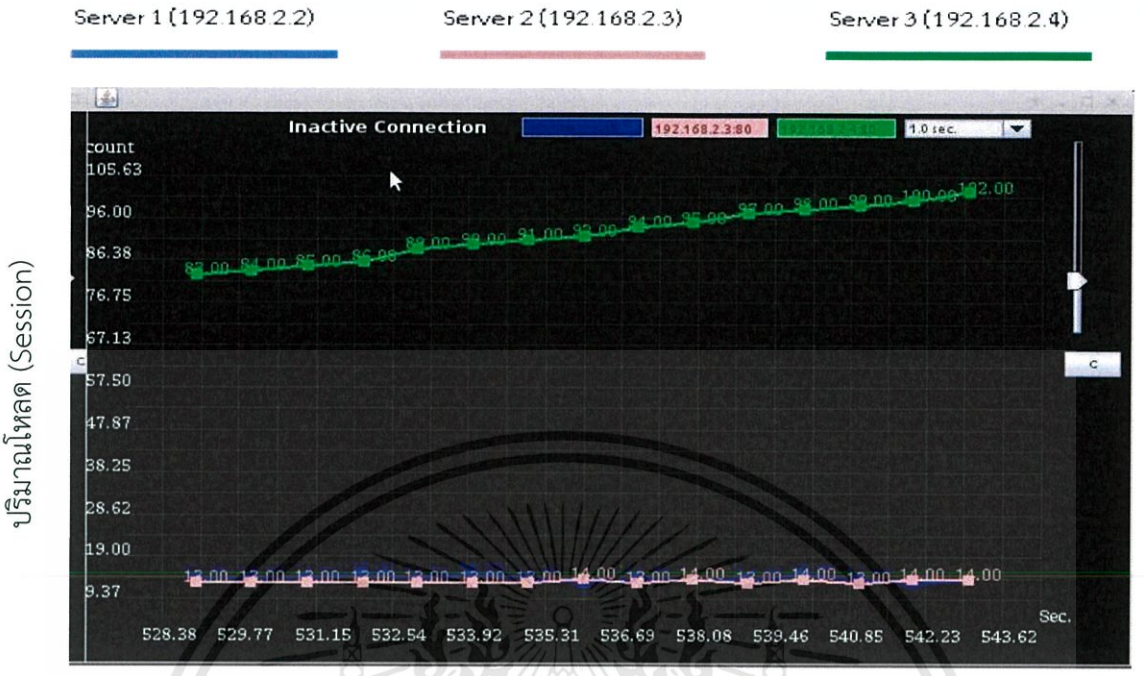
รูปที่ 4.28 กรอบแสดงสถานะปริมาณ Session ที่ Web server แต่ละตัวได้รับผ่านหน้าแอพพลิเคชั่น



ปริมาณโหลด (Session)

เวลา (วินาที)

รูปที่ 4.29 กราฟแสดงปริมาณ Session แบบ Real time ของสถานะ Active connection



รูปที่ 4.30 กราฟแสดงปริมาณ Session แบบ Real time ของสถานะ Inactive connection

จากหน้าโปรแกรม Session Generator และกรอบแสดงสถานะเราจะเห็นว่าปริมาณโหลดหรือ Session ที่ Web Server ทั้ง 3 ตัวได้รับมีปริมาณแตกต่างกันโดยขึ้นอยู่กับค่า weight ที่เปลี่ยนแปลงสลับไปมา เพราะว่าการกระจายภาระงานแบบ Least Connection พิจารณาถึงจำนวนภาระงานที่ Web Server แต่ละตัวได้รับ ซึ่งในการกระจายโหลดของโหลดบาลานซ์เซิร์ฟเวอร์ จะกระจายโหลดไปให้ Web Server ตัวที่มีภาระงานน้อยที่สุดก่อน ส่วนตัวที่รับภาระงานเยอะก็หยุดกระจายโหลดชั่วคราวซึ่งโหลดบาลานซ์เซิร์ฟเวอร์ จะพยายามกระจายโหลดไปให้แก่ Real Server ในอัตราส่วนที่เหมาะสมตามประสิทธิภาพเครื่อง Web Server แต่ละตัว และจากกราฟ Active Connection จะแสดงปริมาณโหลดหรือ Session ที่มีสถานะการทำงานเสร็จสิ้นกระบวนการคือมีการรับโหลดเข้ามาแล้วส่งออกไปยังผู้ใช้บริการโดยสมบูรณ์ ซึ่งกราฟจะแสดงปริมาณของโหลดในปริมาณที่แตกต่างกันโดยมีการมีการเพิ่ม-ลด ของปริมาณโหลดสลับไปมา และกราฟ Inactive Connection จะแสดงปริมาณโหลดหรือ Session ที่มีสถานะอยู่ในระหว่างรอส่งออกไปยังผู้ใช้บริการและรอเวลา Time-out

4.1.2.5 แบบวิธีที่ไม่มีโหลดบาลานซ์

เราจะทำการทดสอบการทำงานของแอพพลิเคชั่นโดยทำการกระจายโหลดแบบไม่มีโหลดบาลานซ์ ซึ่งเราจะป้อนค่าพารามิเตอร์ได้ดังนี้

Virtual IP : 161.246.18.249

Protacol : TCP

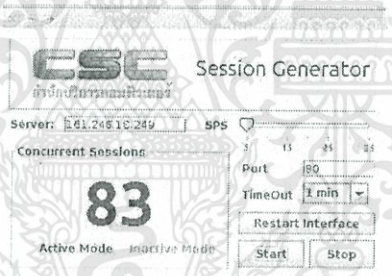
Port : 80

Real IP : 192.168.2.2 (Server1)

Ratio : 1

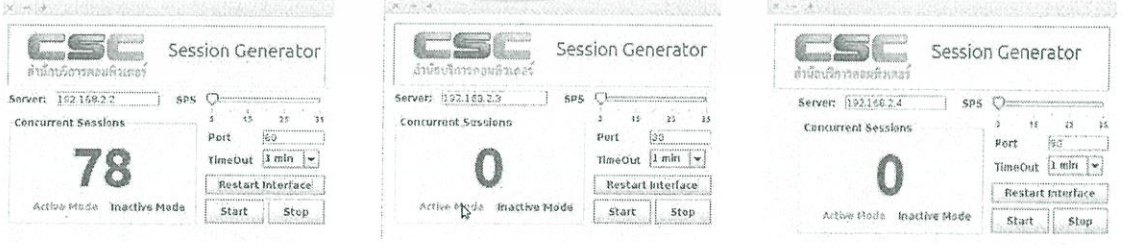
Method : Round Robin (สามารถเลือกวิธีการกระจายภาระงานแบบใดก็ได้ เนื่องจากแบบไม่มีโหลดบาลานซ์เปรียบเสมือนการกระจายโหลดไปให้ Web Server เพียงแค่เครื่องเดียวจึงไม่มีผล)

หลังจากป้อนคำสั่งแล้ว เราจะทำการ Start Service ก่อนเพื่อเริ่มการทำงานของระบบ โดยคลิกที่ปุ่ม Start หลังจากนั้นเราจะทำการป้อนคำสั่งโดยการเลือกปุ่ม OK แอพพลิเคชั่นก็จะทำการป้อนคำสั่งเข้าไปในระบบ โดยผลการทดลองเป็นดังนี้

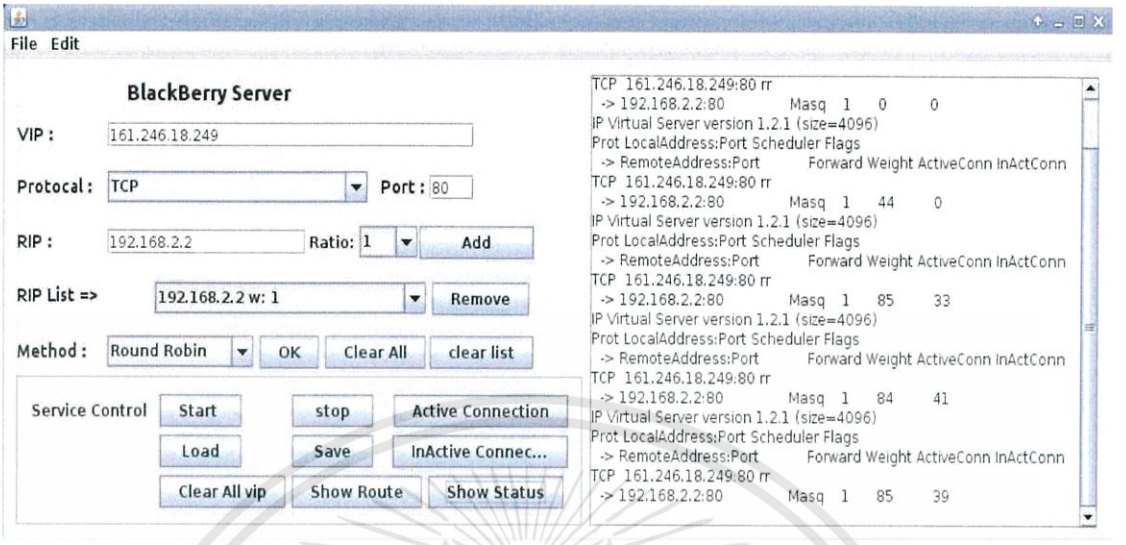


รูปที่ 4.31 ปริมาณ Session ที่ถูก Generate โดยผู้ใช้บริการ

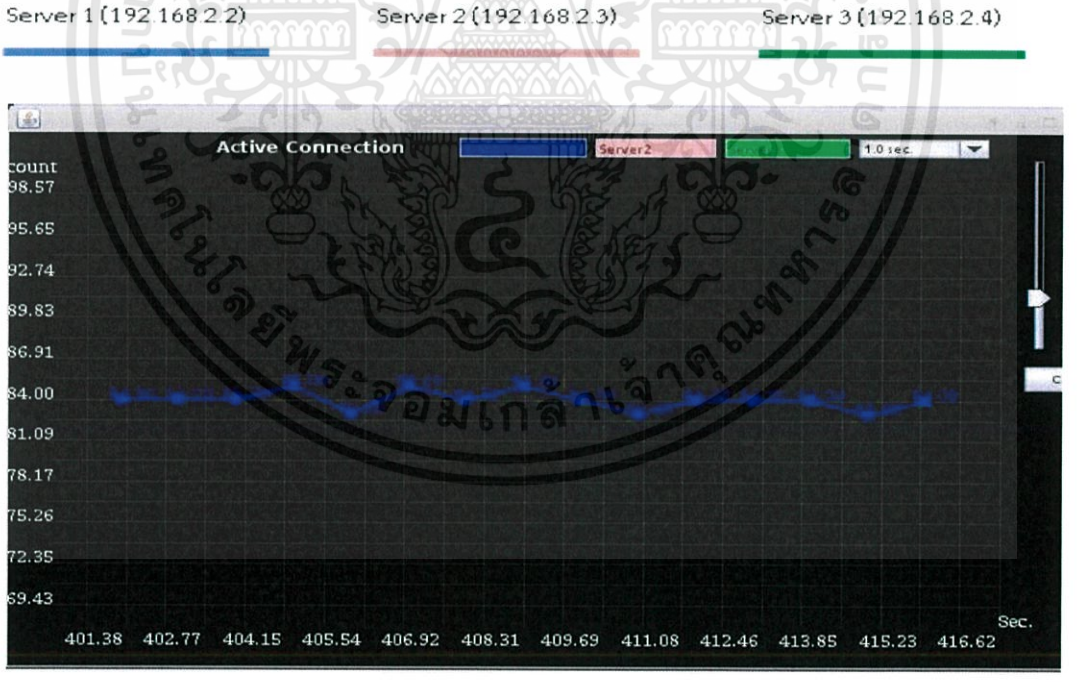
Server 1 (192.168.2.2) Server 2 (192.168.2.3) Server 3 (192.168.2.4)



รูปที่ 4.32 ปริมาณ Session ที่Web server แต่ละตัวได้รับแบบไม่มีโหลดบาลานซ์



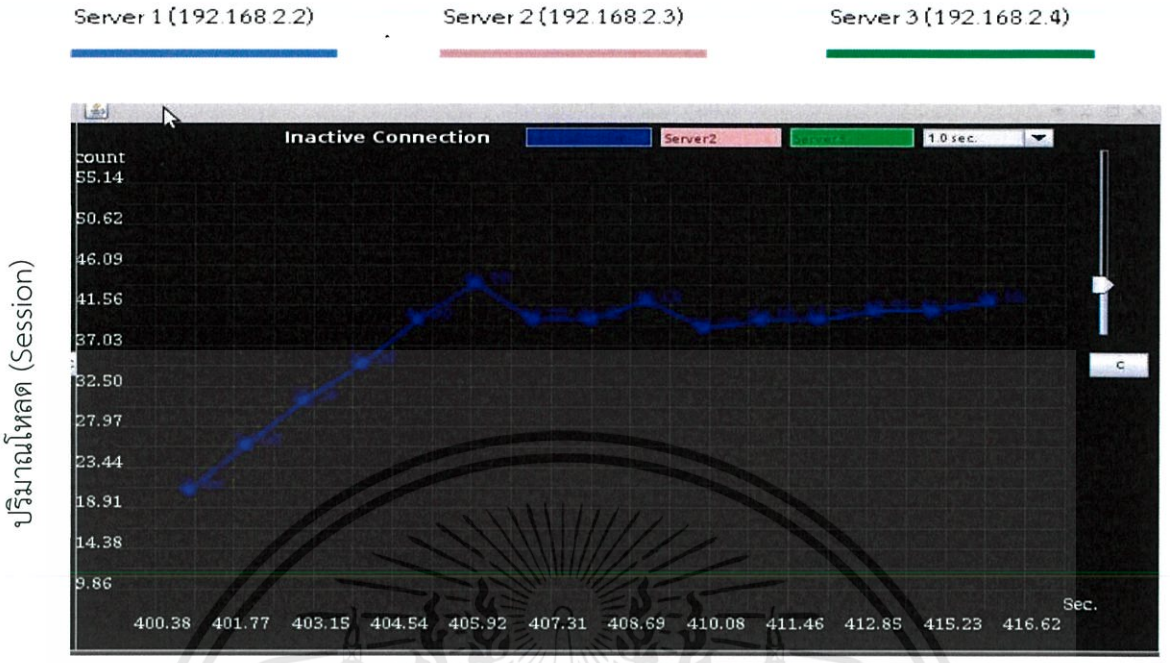
รูปที่ 4.33 กรอบแสดงสถานะปริมาณ Session ที่Web server แต่ละตัวได้รับผ่านหน้าแอพพลิเคชั่น



ปริมาณโหลด (Session)

เวลา (วินาที)

รูปที่ 4.34 กราฟแสดงปริมาณ Session แบบ Real time ของสถานะ Active connection



รูปที่ 4.35 กราฟแสดงปริมาณ Session แบบ Real time ของสถานะ Inactive connection

จากหน้าโปรแกรม Session Generator และกรอบแสดงสถานะเราจะเห็นว่า Server 1 รองรับปริมาณโหลดหรือ Session ทั้งหมดไว้ ทั้งนี้จากรูปในการทดลองมีปริมาณของโหลดไม่เท่ากันเนื่องจากระบบมีการกระจายโหลดอยู่ตลอดเวลา และจากกราฟ Active Connection จะแสดงปริมาณโหลดหรือ Session ที่มีสถานะการทำงานเสร็จสิ้นกระบวนการคือมีการรับโหลดเข้ามาแล้วส่งออกไปยังผู้ใช้บริการโดยสมบูรณ์ และกราฟ Inactive Connection จะแสดงปริมาณโหลดหรือ Session ที่มีสถานะอยู่ในระหว่างรอส่งออกไปยังผู้ใช้บริการและรอเวลา Time-out ซึ่งจากการทดลองสรุปได้ว่า เมื่อไม่มีโหลดบาลานซ์จะทำให้ Web Server รองรับภาระงานในปริมาณที่สูงเกินไปเมื่อผู้ใช้บริการทำการ Access เข้ามาใช้งาน Service ของระบบ

4.2 สรุปผลการทดลอง

จากการทดลองสรุปได้ว่า Panel Controller สามารถที่จะป้อนค่าพารามิเตอร์และแสดงผลการป้อนค่าในการสั่งการกระจายโหลดบาลานซ์ร่วมกับหน้าต่างแอพพลิเคชั่นและเครื่องโหลดบาลานซ์เซิร์ฟเวอร์ได้ และเมื่อทำการกระจายภาระงานด้วยแอพพลิเคชั่นและ Panel controller โดยได้จำลองการเข้าใช้งาน (Access) ของผู้ใช้บริการโดยใช้โปรแกรม Session Generator ด้วยวิธีการกระจายภาระงานทั้ง 4 วิธี ได้แก่ Round robin, Weighted Round Robin(Ratio), Fast Response และ Least Connection โหลดบาลานซ์เซิร์ฟเวอร์สามารถทำการกระจายภาระงานตามวิธีการที่กำหนดได้และสามารถแสดงผลการทำงานที่ชัดเจน

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

จากการทดลองเราจะสรุปได้ว่า เมื่อทำการกระจายภาระงานของโหลดบาลานซ์ แบบ Round Robin แล้ว งานที่ได้รับของ Web Server แต่ละตัวจะมีอัตราส่วนที่เท่าๆกัน ส่วนการกระจายภาระงานของโหลดบาลานซ์ แบบ Ratio งานที่ได้รับของ Web Server แต่ละตัวจะมีอัตราส่วนขึ้นอยู่กับค่าพารามิเตอร์ที่ผู้ดูแลได้ป้อนคำสั่งไว้ในแอพพลิเคชั่น ซึ่งในการกระจายภาระงานทั้ง 2 แบบนั้นจะมีความแตกต่างกันโดยการกระจายภาระงานแบบ Round Robin จะเป็นการกระจายโหลดแบบอัตราส่วนที่เท่าๆกันเหมาะสำหรับชุด Server ที่มีประสิทธิภาพเท่ากันหรือใกล้เคียงกัน ส่วนการกระจายภาระงานแบบ Ratio จะเป็นการกระจายโหลดแบบอัตราส่วนต่างกันซึ่งจะประเมินประสิทธิภาพของชุด Server แต่ละเครื่อง ซึ่งจะเหมาะสำหรับชุด Server ที่มีประสิทธิภาพแตกต่างกัน ถ้า Server เครื่องใดมีประสิทธิภาพสูงเราก็จะกำหนดอัตราส่วนในการรับงานมากกว่าเครื่องอื่นการกระจายโหลดในระบบก็จะเกิดความสมดุลและมีประสิทธิภาพซึ่งในการกระจายภาระงานแบบ Round Robin และ Ratio นั้นจะเป็นการกระจายภาระงานแบบ Static

สำหรับการทดลองการกระจายภาระงานแบบ Dynamic ซึ่งมีสองแบบด้วยกันคือการกระจายภาระงานแบบ Fast Response และ Least Connection นั้นจะต่างออกไปจากการกระจายภาระงานแบบ Static โดยการกระจายภาระงานแบบ Fast Response จะพิจารณาเวลา ที่ใช้ในการตอบสนองต่อระบบ (time response) โดยจะเลือกกระจายโหลดให้แก่ Real Server ที่มีการตอบสนองเร็วที่สุดก่อนโดยการกำหนด Ratio ให้เยอะสุดซึ่งต้องใช้ Process ที่สร้างขึ้นมาทำงานร่วมกัน และการกระจายภาระงานแบบ Least Connection จะพิจารณา Real Server ที่มีการรับงานน้อยที่สุด และจะทำการกระจายโหลดให้แก่ Real Server ที่มีการรับงานน้อยที่สุดก่อน Real Server เครื่องอื่นๆ

สำหรับการทดลองเราจะสรุปได้ว่าในการทำโหลดบาลานซ์มีข้อดีคือ จะทำให้ระบบสามารถรองรับการเข้ามาใช้งานของผู้ใช้บริการได้มากขึ้นทำให้ระบบมีความเสถียรมากขึ้น และในการศึกษาแล้วสร้าง Process เพิ่มขึ้นมาสำหรับการกระจายภาระงานแบบ Dynamic นั้นจะช่วยเพิ่ม

ประสิทธิภาพให้แก่แอพพลิเคชั่น ซึ่งทำให้เกิดความหลากหลายในการเลือกใช้งานและได้ประโยชน์สูงสุดในการใช้งานระบบเครือข่ายและการสร้าง Process สำหรับทำการติดต่อแบบ Polling ที่ช่วยเช็คความพร้อมของ Real Server จะช่วยทำให้เกิดการประหยัดทรัพยากรในการใช้งานของระบบเครือข่ายได้มากขึ้น ส่วนปัญหาในการทำการทดลองคือ ในการทดลองเครื่อง Web Server เราไม่สามารถประเมินประสิทธิภาพของเครื่องได้เนื่องจากเราใช้คอมพิวเตอร์โน๊ตบุคในการจำลองเป็นเครื่อง Web Server เนื่องจากงบประมาณในการทดสอบเครื่อง Real Server ไม่พอ จึงทำให้การทดลองไม่ได้ผลดีเท่าที่ควร เช่นในเรื่องของการกระจายภาระงานแบบ Ratio และการกระจายภาระงานแบบ Dynamic (Fast Response, Least Connection)

และสุดท้ายในการทดสอบการจัดการการใช้งานโหลดบาลานซ์เซิร์ฟเวอร์ด้วยฮาร์ดแวร์ที่สร้างขึ้นเราพบว่าในการป้อนอินพุทคำสั่งมีความสะดวกและรวดเร็วในการใช้งานโหลดบาลานซ์เซิร์ฟเวอร์ เพราะไม่ต้องพิมพ์คำสั่งเพียงแค่ป้อนค่าอินพุทพารามิเตอร์ที่สำคัญลงไปในฮาร์ดแวร์ตามฟังก์ชั่นและสั่งการให้ทำงานก็สามารถทำการกระจายภาระงานได้อย่างมีประสิทธิภาพ

5.2 ข้อเสนอแนะ

5.2.1 แอพพลิเคชั่นควบคุมโหลดบาลานซ์ที่ได้สร้างขึ้น สามารถพัฒนาเพื่อต่อยอดและนำไปใช้ได้จริง และสามารถนำทฤษฎีอื่นๆมาเพิ่มเติมได้ เช่น การประเมินประสิทธิภาพของ Server คือ CPU, RAM ใน Server เป็นต้น

5.2.2 สามารถนำแอพพลิเคชั่นมาประยุกต์ใช้ร่วมกับอุปกรณ์โหลดบาลานซ์ แต่อาจจะแยกการทำงานคนละส่วนได้อย่างมีประสิทธิภาพ เช่น การรีโมตจากผู้ดูแลระบบเข้ามาควบคุมระบบโดยที่ไม่ได้อยู่หน้าเครื่อง Server

5.2.3 โหลดบาลานซ์ที่สร้างขึ้นเป็นเพียงตัวอย่างในการศึกษาทดลอง ซึ่ง specifications ของระบบยังมีข้อด้อยกว่าโหลดบาลานซ์ที่มีในท้องตลาด ซึ่งอ้างอิงได้จาก ภาคผนวก ก. ตารางแสดงการเปรียบเทียบ specifications โดยโหลดบาลานซ์ที่สร้างขึ้นสามารถพัฒนาให้มีวิธีการกระจายภาระงานที่หลากหลายขึ้นได้ เช่น แบบ Policy Routing แบบ Link Backup เป็นต้น

บรรณานุกรม

[1] พนิดา พานิชกุล. การเขียนโปรแกรมคอมพิวเตอร์ เบื้องต้น ด้วยภาษา Java. พิมพ์ครั้งที่ 5. กรุงเทพฯ : เคทีพี, 2554.

[2] อนรรฆนงค์ คุณมนี. คู่มือเขียนโปรแกรมภาษา JAVA ฉบับผู้เริ่มต้น. พิมพ์ครั้งที่ 1. นนบุรี : ไอดีซีฯ, 2551.

[3] พิศาล พิทยาธุรวิวัฒน์. ติดตั้งระบบเครือข่าย Linux Server ภาคปฏิบัติ. กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2555.

[4] ศูนย์เทคโนโลยีสารสนเทศและการสื่อสาร สป.ศธ. "การทำ Load Balance รองรับการการใช้งานของ User จำนวนมาก." http://www.bict.moe.go.th/event/index.php.

[5] Java คืออะไร. http://www.mindphp.com.

[6] ระบบปฏิบัติการ Linux. http://www.thaiall.com/os/os11.htm.

[7] อูบุนตู. http://th.wikipedia.org/wiki/อูบุนตู.

[8] load balancing คืออะไร. http://www.mindphp.com/คู่มือ/73-คืออะไร/2060-load-balancing คืออะไร

[9] Load Balance System. http://www.rackjumper.com/loadbalance/ http://www.reocities.com/SiliconValley/station/3169/lcd.htm แอลซีดี

[10] การสื่อสารผ่านพอร์ทอนุกรม. http://adisak-diy.com/page06_1.html

[11] ไมโครคอนโทรลเลอร์ MCS-51. http://eng.sut.ac.th/tce/SeniorProjects /old/student2/ST51.htm

[12] การอินเตอร์รัพท์ [ออนไลน์]. เข้าถึงได้จาก : http://www.commuatts.com /ckfinder/userfiles/files/วิชาไมโครคอนโทรลเลอร์/interrupt.ppt

[13] การใช้งานอินเตอร์รัพท์ [ออนไลน์]. เข้าถึงได้จาก : http://eerg.eng.rmutp.ac.th /E_Learning/Microprocessor/chapter/6/

[14] การใช้งานแป้นปุ่มกด (Keypad) แบบ 4x4 ปุ่ม [ออนไลน์]. เข้าถึงได้จาก : http://www.ee.kmutnb.ac.th/eerobot/esl/learning/index.php?article=4x4-keypad

[15] การใช้งาน LCD Module *[ออนไลน์]. เข้าถึงได้จาก* : http://eng.vu.ac.th/mn/2009/~arnon/thecheing/fms_cim/lab5.pdf

[16] เรียนรู้การทำงานของโมดูล LCD [ออนไลน์]. เข้าถึงได้จาก : http://www.tpemagazine.com/download/issue03/LCD_tutorial.pdf

ภาคผนวก ก. ตารางแสดงการเปรียบเทียบ Specifications

ตารางแสดงการเปรียบเทียบ Specifications

คุณสมบัติ	ของผู้ศึกษา	ของบริษัท XXX
Standards and Protocols	IEEE 802.3, TCP/IP, DHCP, , NAT, HTTP,	IEEE 802.3, 802.3u, 802.3x, TCP/IP, DHCP, ICMP, NAT, PPPoE, SNTP, HTTP, DDNS
Interface	2 Fixed Ethernet LAN Port	1 Fixed Ethernet WAN Port 1 Fixed Ethernet LAN Port 3 Changeable Ethernet WAN/LAN Ports
Concurrent Session	600	30000
DHCP	DHCP Server/Client	DHCP Server/Client, DHCP Address Reservation
Load Balance	Round robin, Weighted Round Robin, Fast Response , Least Connection	Policy Routing, Link Backup
NAT	One-to-One NAT	One-to-One NAT, Multi-Nets NAT
Routing	Static Routing	Static Routing
Hard disk	31 GB	-
Ram	1122 MB	-

ภาคผนวก ข. MAN IPVSADM

ipvsadm(8) - Linux man page

Name

ipvsadm - Linux Virtual Server administration

Synopsis

ipvsadm -A|E -t|u|f service-address **[-s** scheduler]

[-p [timeout]] [-O] [-M netmask]

ipvsadm -D -t|u|f service-address

ipvsadm -C

ipvsadm -R

ipvsadm -S [-n]

ipvsadm -a|e -t|u|f service-address **-r** server-address

[-g|i|m] [-w weight] [-x upper] [-y lower]

ipvsadm -d -t|u|f service-address **-r** server-address

ipvsadm -L|l [options]

ipvsadm -Z [-t|u|f service-address]

ipvsadm --set tcp tcpfin udp

ipvsadm --start-daemon state **[--mcast-interface** interface]

[**--syncid** syncid]

ipvsadm --stop-daemon state

ipvsadm -h

1. Description

Ipvsadm(8) is used to set up, maintain or inspect the virtual server table in the Linux kernel. The Linux Virtual Server can be used to build scalable network services based on a cluster of two or more nodes. The active node of the cluster redirects service requests to a collection of server hosts that will actually perform the services. Supported features include two protocols (TCP and UDP), three packet-forwarding methods (NAT, tunneling, and direct routing), and eight load balancing algorithms (round robin, weighted round robin, least-connection, weighted least-connection, locality-based least-connection, locality-based least-connection with replication, destination-hashing, and source-hashing).

The command has two basic formats for execution:

ipvsadm COMMAND [protocol] service-address

[scheduling-method] [persistence options]

ipvsadm command [protocol] service-address

server-address [packet-forwarding-method] [weight options]

The first format manipulates a virtual service and the algorithm for assigning service requests to real servers. Optionally, a persistent timeout and network mask for the granularity of a persistent service may be specified. The second format manipulates a real server that is associated with an existing virtual service. When specifying a real server, the packet-forwarding method and the weight of the real server, relative to other real servers for the virtual service, may be specified, otherwise defaults will be used.

2. COMMANDS

ipvsadm(8) recognises the commands described below. Upper-case commands maintain virtual services. Lower-case commands maintain real servers that are associated with a virtual service.

-A, --add-service

Add a virtual service. A service address is uniquely defined by a triplet: IP address, port number, and protocol. Alternatively, a virtual service may be defined by a firewall-mark.

-E, --edit-service

Edit a virtual service.

-D, --delete-service

Delete a virtual service, along with any associated real servers.

-C, --clear

Clear the virtual server table.

-R, --restore

Restore Linux Virtual Server rules from stdin. Each line read from stdin will be treated as the command line options to a separate invocation of ipvsadm. Lines read from stdin can optionally begin with "ipvsadm". This option is useful to avoid executing a large number or ipvsadm commands when constructing an extensive routing table.

-S, --save

Dump the Linux Virtual Server rules to stdout in a format that can be read by -R|--restore.

-a, --add-server

Add a real server to a virtual service.

-e, --edit-server

Edit a real server in a virtual service.

-d, --delete-server

Remove a real server from a virtual service.

-L, -l, --list

List the virtual server table if no argument is specified. If a service-address is selected, list this service only. If the -c option is selected, then display the connection table. The exact output is affected by the other arguments given.

-Z, --zero

Zero the packet, byte and rate counters in a service or all services.

--set tcp tcpfin udp

Change the timeout values used for IPVS connections. This command always takes 3 parameters, representing the timeout values (in seconds) for TCP sessions, TCP sessions after receiving a FIN packet, and UDP packets, respectively. A timeout value 0 means that the current timeout value of the corresponding entry is preserved.

--start-daemon state

Start the connection synchronization daemon. The state is to indicate that the daemon is started as master or backup. The connection synchronization daemon is implemented inside the Linux kernel. The master daemon running at the primary load balancer multicasts changes of connections periodically, and the backup daemon running at the backup load balancers receives multicast message and creates corresponding connections. Then, in case the primary load balancer fails, a backup load balancer will takeover, and it has state of almost all connections, so that almost all established connections can continue to access the service.

The sync daemon currently only supports IPv4 connections.

--stop-daemon

Stop the connection synchronization daemon.

-h, --help

Display a description of the command syntax.

3. PARAMETERS

The commands above accept or require zero or more of the following parameters.

-t, --tcp-service service-address

Use TCP service. The service-address is of the form host[:port]. Host may be one of a plain IP address or a hostname. Port may be either a plain port number or the service name of port. The Port may be omitted, in which case zero will be used. A Port of zero is only valid if the service is persistent as the -p|--persistent option, in which case it is a wild-card port, that is connections will be accepted to any port.

-u, --udp-service service-address

Use UDP service. See the -t|--tcp-service for the description of the service-address.

-f, --fwmark-service integer

Use a firewall-mark, an integer value greater than zero, to denote a virtual service instead of an address, port and protocol (UDP or TCP). The marking of packets with a firewall-mark is configured using the -m|--mark option to iptables(8). It can be used to build a virtual service assoicated with the same real servers, covering multiple IP address, port and protocol tripplets. If IPv6 addresses are used, the -6 option must be used.

Using firewall-mark virtual services provides a convenient method of grouping together different IP addresses, ports and protocols into a single virtual service. This is useful for both simplifying configuration if a large number of virtual services are required and grouping persistence across what would otherwise be multiple virtual services.

-s, --scheduler scheduling-method

scheduling-method Algorithm for allocating TCP connections and UDP datagrams to real servers. Scheduling algorithms are implemented as kernel modules. Ten are shipped with the Linux Virtual Server:

rr - Robin Robin: distributes jobs equally amongst the available real servers.

wrr - Weighted Round Robin: assigns jobs to real servers proportionally to there real servers' weight. Servers with higher weights receive new jobs first and get more jobs than servers with lower weights. Servers with equal weights get an equal distribution of new jobs.

lc - Least-Connection: assigns more jobs to real servers with fewer active jobs.

wlc - Weighted Least-Connection: assigns more jobs to servers with fewer jobs and relative to the real servers' weight (Ci/Wi). This is the default.

lblc - Locality-Based Least-Connection: assigns jobs destined for the same IP address to the same server if the server is not overloaded and available; otherwise assign jobs to servers with fewer jobs, and keep it for future assignment.

lblcr - Locality-Based Least-Connection with Replication: assigns jobs destined for the same IP address to the least-connection node in the server set for the IP address. If all the node in the server set are over loaded, it picks up a node with fewer jobs in the cluster and adds it in the sever set for the target. If the server set has not been modified for the specified time, the most loaded node is removed from the server set, in order to avoid high degree of replication.

dh - Destination Hashing: assigns jobs to servers through looking up a statically assigned hash table by their destination IP addresses.

sh - Source Hashing: assigns jobs to servers through looking up a statically assigned hash table by their source IP addresses.

sed - Shortest Expected Delay: assigns an incoming job to the server with the shortest expected delay. The expected delay that the job will experience is (Ci + 1) / Ui if sent to the ith server, in which Ci is the number of jobs on the the ith server and Ui is the fixed service rate (weight) of the ith server.

nq - Never Queue: assigns an incoming job to an idle server if there is, instead of waiting for a fast one; if all the servers are busy, it adopts the Shortest Expected Delay policy to assign the job.

-p, --persistent [timeout]

Specify that a virtual service is persistent. If this option is specified, multiple requests from a client are redirected to the same real server selected for the first request. Optionally, the timeout of persistent sessions may be specified given in seconds, otherwise the default of 300 seconds will be used. This option may be used in conjunction with protocols such as SSL or FTP where it is important that clients consistently connect with the same real server.

Note: If a virtual service is to handle FTP connections then persistence must be set for the virtual service if Direct Routing or Tunnelling is used as the forwarding mechanism. If Masquerading is used in conjunction with an FTP service than persistence is not necessary,

but the ip_vs_ftp kernel module must be used. This module may be manually inserted into the kernel using insmod(8).

-M, --netmask netmask

Specify the granularity with which clients are grouped for persistent virtual services. The source address of the request is masked with this netmask to direct all clients from a network to the same real server. The default is 255.255.255.255, that is, the persistence granularity is per client host. Less specific netmasks may be used to resolve problems with non-persistent cache clusters on the client side. IPv6 netmasks should be specified as a prefix length between 1 and 128. The default prefix length is 128.

-r, --real-server server-address

Real server that an associated request for service may be assigned to. The server-address is the host address of a real server, and may plus port. Host can be either a plain IP address or a hostname. Port can be either a plain port number or the service name of port. In the case of the masquerading method, the host address is usually an RFC 1918 private IP address, and the port can be different from that of the associated service. With the tunneling and direct routing methods, port must be equal to that of the service address. For normal services, the port specified in the service address will be used if port is not specified. For fwmark services, port may be omitted, in which case the destination port on the real server will be the destination port of the request sent to the virtual service.

[packet-forwarding-method]

-g, --gatewaying Use gatewaying (direct routing). This is the default.

-i, --ipip Use ipip encapsulation (tunneling).

-m, --masquerading Use masquerading (network access translation, or NAT).

Note: Regardless of the packet-forwarding mechanism specified, real servers for addresses for which there are interfaces on the local node will be use the local forwarding method, then packets for the servers will be passed to upper layer on the local node. This cannot be specified by ipvsadm, rather it set by the kernel as real servers are added or modified.

-w, --weight weight

Weight is an integer specifying the capacity of a server relative to the others in the pool. The valid values of weight are 0 through to 65535. The default is 1. Quiescent servers

are specified with a weight of zero. A quiescent server will receive no new jobs but still serve the existing jobs, for all scheduling algorithms distributed with the Linux Virtual Server. Setting a quiescent server may be useful if the server is overloaded or needs to be taken out of service for maintenance.

-x, --u-threshold uthreshold

uthreshold is an integer specifying the upper connection threshold of a server. The valid values of uthreshold are 0 through to 65535. The default is 0, which means the upper connection threshold is not set. If uthreshold is set with other values, no new connections will be sent to the server when the number of its connections exceeds its upper connection threshold.

-y, --l-threshold lthreshold

lthreshold is an integer specifying the lower connection threshold of a server. The valid values of lthreshold are 0 through to 65535. The default is 0, which means the lower connection threshold is not set. If lthreshold is set with other values, the server will receive new connections when the number of its connections drops below its lower connection threshold. If lthreshold is not set but uthreshold is set, the server will receive new connections when the number of its connections drops below three forth of its upper connection threshold.

--mcast-interface interface

Specify the multicast interface that the sync master daemon sends outgoing multicasts through, or the sync backup daemon listens to for multicasts.

--syncid syncid

Specify the syncid that the sync master daemon fills in the SyncID header while sending multicast messages, or the sync backup daemon uses to filter out multicast messages not matched with the SyncID value. The valid values of syncid are 0 through to 255. The default is 0, which means no filtering at all.

-c, --connection

Connection output. The list command with this option will list current IPVS connections.

--timeout

Timeout output. The list command with this option will display the timeout values (in seconds) for TCP sessions, TCP sessions after receiving a FIN packet, and UDP packets.

--daemon

Daemon information output. The list command with this option will display the daemon status and its multicast interface.

--stats

Output of statistics information. The list command with this option will display the statistics information of services and their servers.

--rate

Output of rate information. The list command with this option will display the rate information (such as connections/second, bytes/second and packets/second) of services and their servers.

--thresholds

Output of thresholds information. The list command with this option will display the upper/lower connection threshold information of each server in service listing.

--persistent-conn

Output of persistent connection information. The list command with this option will display the persistent connection counter information of each server in service listing. The persistent connection is used to forward the actual connections from the same client/network to the same server.

--sort

Sort the list of virtual services and real servers. The virtual service entries are sorted in ascending order by <protocol, address, port>. The real server entries are sorted in ascending order by <address, port>. (default)

--nosort

Do not sort the list of virtual services and real servers.

-O, --ops

Specify that a virtual service uses one-packet scheduling. This option can be used only for UDP services. If this option is specified, all connections are created only to schedule

one packet. Option is useful to schedule UDP packets from same client port to different real servers.

-n, --numeric

Numeric output. IP addresses and port numbers will be printed in numeric format rather than as as host names and services respectively, which is the default.

--exact

Expand numbers. Display the exact value of the packet and byte counters, instead of only the rounded number in K's (multiples of 1000) M's (multiples of 1000K) or G's (multiples of 1000M). This option is only relevant for the -L command.

-6

Use with -f to signify fwmark rule uses IPv6 addresses.

EXAMPLE 1 - Simple Virtual Service

The following commands configure a Linux Director to distribute incoming requests addressed to port 80 on 207.175.44.110 equally to port 80 on five real servers. The forwarding method used in this example is NAT, with each of the real servers being masqueraded by the Linux Director.

```
ipvsadm -A -t 207.175.44.110:80 -s rr
ipvsadm -a -t 207.175.44.110:80 -r 192.168.10.1:80 -m
ipvsadm -a -t 207.175.44.110:80 -r 192.168.10.2:80 -m
ipvsadm -a -t 207.175.44.110:80 -r 192.168.10.3:80 -m
ipvsadm -a -t 207.175.44.110:80 -r 192.168.10.4:80 -m
ipvsadm -a -t 207.175.44.110:80 -r 192.168.10.5:80 -m
```

Alternatively, this could be achieved in a single ipvsadm command.

```
echo "
-A -t 207.175.44.110:80 -s rr
```

-a -t 207.175.44.110:80 -r 192.168.10.1:80 -m

-a -t 207.175.44.110:80 -r 192.168.10.2:80 -m

-a -t 207.175.44.110:80 -r 192.168.10.3:80 -m

-a -t 207.175.44.110:80 -r 192.168.10.4:80 -m

-a -t 207.175.44.110:80 -r 192.168.10.5:80 -m

" | ipvsadm -R

As masquerading is used as the forwarding mechanism in this example, the default route of the real servers must be set to the linux director, which will need to be configured to forward and masquerade packets. This can be achieved using the following commands:

echo "1" > /proc/sys/net/ipv4/ip_forward

EXAMPLE 2 - Firewall-Mark Virtual Service

The following commands configure a Linux Director to distribute incoming requests addressed to any port on 207.175.44.110 or 207.175.44.111 equally to the corresponding port on five real servers. As per the previous example, the forwarding method used in this example is NAT, with each of the real servers being masqueraded by the Linux Director.

ipvsadm -A -f 1 -s rr

ipvsadm -a -f 1 -r 192.168.10.1:0 -m

ipvsadm -a -f 1 -r 192.168.10.2:0 -m

ipvsadm -a -f 1 -r 192.168.10.3:0 -m

ipvsadm -a -f 1 -r 192.168.10.4:0 -m

ipvsadm -a -f 1 -r 192.168.10.5:0 -m

As masquerading is used as the forwarding mechanism in this example, the default route of the real servers must be set to the linux director, which will need to be configured to forward and masquerade packets. The real server should also be configured to mark incoming packets addressed to any port on 207.175.44.110 and 207.175.44.111 with firewall-mark 1. If FTP traffic is to be handled by this virtual service, then the ip_vs_ftp kernel module needs to be inserted into the kernel. These operations can be achieved using the following commands:

echo "1" > /proc/sys/net/ipv4/ip_forward

modprobe ip_tables

iptables -A PREROUTING -t mangle -d 207.175.44.110/31 -j MARK --set-mark 1

modprobe ip_vs_ftp

4. IPv6

IPv6 addresses should be surrounded by square brackets ([and]).

ipvsadm -A -t [2001:db8::80]:80 -s rr

ipvsadm -a -t [2001:db8::80]:80 -r [2001:db8::a0a0]:80 -m

fwmark IPv6 services require the -6 option.

Notes

The Linux Virtual Server implements three defense strategies against some types of denial of service (DoS) attacks. The Linux Director creates an entry for each connection in order to keep its state, and each entry occupies 128 bytes effective memory. LVS's vulnerability to a DoS attack lies in the potential to increase the number entries as much as possible until the linux director runs out of memory. The three defense strategies against the attack are: Randomly drop some entries in the table. Drop 1/rate packets before forwarding them. And use secure tcp state transition table and short timeouts. The strategies are controlled by sysctl variables and corresponding entries in the /proc filesystem:

/proc/sys/net/ipv4/vs/drop_entry /proc/sys/net/ipv4/vs/drop_packet

/proc/sys/net/ipv4/vs/secure_tcp

Valid values for each variable are 0 through to 3. The default value is 0, which disables the respective defense strategy. 1 and 2 are automatic modes - when there is no enough available memory, the respective strategy will be enabled and the variable is automatically set to 2, otherwise the strategy is disabled and the variable is set to 1. A value of 3 denotes that the respective strategy is always enabled. The available memory threshold and secure TCP timeouts can be tuned using the sysctl variables and corresponding entries in the /proc filesystem:

/proc/sys/net/ipv4/vs/amemthresh /proc/sys/net/ipv4/vs/timeout_*

Files

/proc/net/ip_vs

/proc/net/ip_vs_app

/proc/net/ip_vs_conn

/proc/net/ip_vs_stats

/proc/sys/net/ipv4/vs/am_droprate

/proc/sys/net/ipv4/vs/amemthresh

/proc/sys/net/ipv4/vs/drop_entry

/proc/sys/net/ipv4/vs/drop_packet

/proc/sys/net/ipv4/vs/secure_tcp

/proc/sys/net/ipv4/vs/timeout_close

/proc/sys/net/ipv4/vs/timeout_closewait

/proc/sys/net/ipv4/vs/timeout_established

/proc/sys/net/ipv4/vs/timeout_finwait

/proc/sys/net/ipv4/vs/timeout_icmp

/proc/sys/net/ipv4/vs/timeout_lastack

/proc/sys/net/ipv4/vs/timeout_listen

/proc/sys/net/ipv4/vs/timeout_synack

/proc/sys/net/ipv4/vs/timeout_synrecv

/proc/sys/net/ipv4/vs/timeout_synsent

/proc/sys/net/ipv4/vs/timeout_timewait

/proc/sys/net/ipv4/vs/timeout_udp

See Also

The LVS web site (http://www.linuxvirtualserver.org/) for more documentation about LVS.

ipvsadm-save(8), ipvsadm-restore(8), iptables(8),

insmod(8), modprobe(8)

Authors

ipvsadm - Wensong Zhang <wensong@linuxvirtualserver.org>

Peter Kese <peter.kese@ijs.si>

man page - Mike Wangsmo <wanger@redhat.com>

Wensong Zhang <wensong@linuxvirtualserver.org>

Horms <horms@verge.net.au>