การย้ายโหลดจากอุปกรณ์ไฟร์วอลล์แบบรักษาสถานะ โดยใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์

MIGRATING THE LOADS OF THE STATEFUL FIREWALL BY USING THE FIREWALL LOAD BALANCER

รัฐิพงษ์ พุทธะชารีย์

RATTIPONG PUTTHACHAROEN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2548

ISBN 974-15-1845-5

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การย้ายโหลดจากอุปกรณ์ไฟร์วอลล์แบบรักษาสถานะ โดยใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์

MIGRATING THE LOADS OF THE STATEFUL FIREWALL BY USING THE FIREWALL LOAD BALANCER

T060909

รัฐติ์พงษ์ พุทธเจริญ

RATTIPONG PUTTHACHAROEN

เลขหมู่.....
เลขทะเบียน 60909
วัน,เดือน,ปี - 6 ก.ค. 2549

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2548

ISBN 974-15-1845-5

MIGRATING THE LOADS OF THE STATEFUL FIREWALL BY USING THE FIREWALL LOAD BALANCER

RATTIPONG PUTTHACHAROEN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN COMPUTER ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2005

ISBN 974-15-1845-5

COPYRIGHT 2005

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

หัวข้อวิทยานิพนธ์	การย้ายโหลดจากอุปกรณ์ไฟร์วอลล์แบบรักษาสถานะโดยใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์
นักศึกษา	นายรัฐิติ์พงษ์ พุทธเจริญ
รหัสประจำตัว	43061613
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2548
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ประทีป บัญญัตินพรัตน์

บทคัดย่อ

ในปัจจุบันอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ (Firewall load balancer) ไม่สามารถทำการย้ายการเชื่อมต่อหรือโหลดจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งได้ ทั้งในกรณีที่มีไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งไม่สามารถทำงานต่อ และในกรณีที่มีไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งมีโหลดแตกต่างกับไฟร์วอลล์แบบรักษาสถานะตัวอื่น ๆ เป็นจำนวนมาก วิทยานิพนธ์ฉบับนี้เสนอวิธีการเพื่อเพิ่มความสามารถให้กับอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ให้สามารถทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งได้ โดยไม่คำนึงว่าไฟร์วอลล์แบบรักษาสถานะแต่ละตัวจะสามารถทำการแลกเปลี่ยนข้อมูลระหว่างกันหรือทำงานร่วมกันได้หรือไม่ วิธีที่นำเสนอใช้เทคนิคในการจำลองแพ็คเก็ตพิเศษ เพื่อหลอกให้ไฟร์วอลล์แบบรักษาสถานะตัวที่จะทำการย้ายการเชื่อมต่อไป ให้ทำการบันทึกข้อมูลของการเชื่อมต่อที่กำลังจะถูกย้ายมาลงในตารางสถานะหรือตารางเซสชั่นของมันเสียก่อน เมื่อมีข้อมูลของการเชื่อมต่อที่จะย้ายมาในตารางสถานะของมันแล้ว มันก็จะอนุญาตให้ทราฟฟิคของการเชื่อมต่อนั้นผ่านตัวมันได้ ดังนั้นอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ก็จะสามารถทำการย้ายการเชื่อมต่อที่ต้องการไปยังไฟร์วอลล์แบบรักษาสถานะตัวนั้นได้

Thesis Title	Migrating the loads of the stateful firewall by using the firewall load balancer
Student	Mr. Rattipong Putthacharoen.
Student ID.	43061613
Degree	M.Eng
Programme	Computer Engineering
Year	2005
Thesis Advisor	Assoc. Prof. Pratheep Bunyatnoparat

ABSTRACT

Nowadays the firewall load balancer can not migrate the connection (loads) from one stateful firewall to the other stateful firewall neither in case of one firewall fails nor in case of one firewall has the very peak load. This thesis proposed the method for increasing the efficiency of the firewall load balancer in order to enable it to migrate the connection from one stateful firewall to the other stateful firewall. By this method the connection would be migrated without concerning about those stateful firewalls come from the same vendor or concering about those stateful firewalls can share the state table (session table) between each others or not. In this thesis we used the special packet called "the simulated packet" in order to fake the stateful firewall that the new connection was coming to it. When the stateful firewall found that it would add the information of that connection into its state table and then the traffics of that connection would be allowed to pass through it. Upto this state the firewall load balacer would be able to migrate the connection to the new stateful firewall.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา รศ. ประทีป บัญญัตินพรัตน์ ที่ให้ความช่วยเหลือ ให้คำชี้แนะช่วยแก้ปัญหาตลอดจนให้ความรู้และประสบการณ์ที่ดีแก่ข้าพเจ้า ขอขอบพระคุณ รศ. บรรจง ปิยธำรง อดีตอาจารย์ที่ปรึกษาที่ให้คำแนะนำ และแนวทางในการทำวิทยานิพนธ์ฉบับนี้

ขอขอบพระคุณ คณะกรรมการสอบวิทยานิพนธ์ทุกท่าน ที่ได้ให้คำแนะนำในการแก้ไขวิทยานิพนธ์ฉบับนี้ให้มีความสมบูรณ์มากยิ่งขึ้น

สุดท้ายต้องขอขอบคุณ นายศักดิ์ชัย เส้งสุย เพื่อนที่ให้คำปรึกษา ให้คำแนะนำ และให้ความช่วยเหลือจนวิทยานิพนธ์นี้สำเร็จ

สำหรับคุณงามความดีใดที่เกิดจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้แก่บิดามารดา ซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนครูอาจารย์ที่เคารพทุกท่านที่ได้ประสิทประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีแก่ข้าพเจ้า

รัฐิติ์พงษ์ พุทธเจริญ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

สารบัญตาราง

สารบัญตาราง (ต่อ)

ตารางที่ หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันองค์กรต่าง ๆ ได้มีการใช้ไฟร์วอลล์แบบรักษาสถานะกันอย่างแพร่หลาย ในบางองค์กรมีการใช้ไฟร์วอลล์แบบรักษาสถานะถึงหลายตัวด้วยกัน ซึ่งไฟร์วอลล์แบบรักษาสถานะเหล่านั้นอาจจะมาจากหลาย ๆ ผู้ผลิต หรืออาจจะมาจากผู้ผลิตเดียวกันแต่มีหลากหลายรุ่น การที่จะทำให้ไฟร์วอลล์แบบรักษาสถานะจากต่างผู้ผลิตกัน หรือไฟร์วอลล์แบบรักษาสถานะจากผู้ผลิตเดียวกันแต่เป็นคนละรุ่นกัน สามารถทำงานรวมกัน เพื่อเพิ่มความเร็วในการส่งข้อมูล และเพิ่มเสถียรภาพให้กับระบบได้นั้น จะทำได้โดยการใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์มาช่วยควบคุมการส่งข้อมูล และทำการกระจายโหลดหรือกระจายการเชื่อมต่อไปยังไฟร์วอลล์แบบรักษาสถานะเหล่านั้น พร้อมกับคอยตรวจสอบสถานะภาพของไฟร์วอลล์แบบรักษาสถานะแต่ละตัว

เมื่อเป็นไฟร์วอลล์แบบรักษาสถานะที่มาจากต่างผู้ผลิตกัน หรือเป็นไฟร์วอลล์แบบรักษาสถานะจากผู้ผลิตเดียวกันแต่คนละรุ่นกันแล้ว โดยทั่วไปแล้วจะไม่สามารถทำการแลกเปลี่ยนข้อมูลในตารางสถานะหรือข้อมูลในตารางเซสชั่นระหว่างกันได้ ซึ่งจะทำให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ไม่สามารถทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งได้ การที่ไม่สามารถทำการย้ายการเชื่อมต่อได้นี้ จะทำให้เกิดปัญหาที่สำคัญ 2 อย่างตามมา คือ

1. เมื่อไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งหยุดทำงานไปไม่ว่ากรณีใดก็ตาม จะทำให้การเชื่อมต่อที่ถูกส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวนั้นต้องหยุดตามไปด้วย ซึ่งจะทำให้ต้องเริ่มต้นสร้างการเชื่อมต่อเพื่อส่งข้อมูลกันใหม่ทั้งหมด

2. เมื่อไฟร์วอลล์แบบรักษาสถานะมีโหลดแตกต่างกันมาก อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะไม่สามารถทำการย้ายโหลดจากไฟร์วอลล์แบบรักษาสถานะตัวที่มีโหลดมาก ไปยังไฟร์วอลล์แบบรักษาสถานะตัวที่มีโหลดน้อยกว่าได้

ซึ่งจากที่ได้กล่าวมาจะเห็นว่าถึงแม้ว่าเราจะใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์แล้วก็ตาม แต่ก็ยังไม่สามารถใช้ไฟร์วอลล์แบบรักษาสถานะได้อย่างเต็มประสิทธิภาพ และยังไม่สามารถทำให้ระบบมีเสถียรภาพสูงสุดได้

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

การศึกษาและการทำวิจัยนี้มีความมุ่งหมายและวัตถุประสงค์ดังต่อไปนี้

1. เพื่อทำการศึกษาการทำงานของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ และอุปกรณ์ไฟร์วอลล์แบบรักษาสถานะ
2. เพื่อทำการพัฒนาและเพิ่มความสามารถของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ ให้สามารถทำงานร่วมกับไฟร์วอลล์แบบรักษาสถานะได้อย่างมีประสิทธิภาพมากขึ้น
3. เพื่อทำการพัฒนาและเพิ่มความสามารถให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ ให้สามารถทำการรักษาสภาพของการเชื่อมต่อทุกการเชื่อมต่อให้สามารถดำเนินต่อไปได้ แม้ว่าจะมีไฟร์วอลล์แบบรักษาสถานะบางตัวไม่สามารถทำงานต่อได้ โดยอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ต้องสามารถส่งทราฟฟิคที่เหลือของการเชื่อมต่อ ผ่านไฟร์วอลล์แบบรักษาสถานะตัวอื่นที่ยังสามารถทำงานได้อยู่ ถึงแม้ว่าไฟร์วอลล์แบบรักษาสถานะตัวอื่น จะเป็นไฟร์วอลล์แบบรักษาสถานะที่มาจากต่างผู้ผลิตกัน หรือถึงแม้ว่าไฟร์วอลล์แบบรักษาสถานะตัวอื่น จะไม่สามารถทำการแลกเปลี่ยนข้อมูลในตารางสถานะกับไฟร์วอลล์ตัวที่หยุดทำงานได้ก็ตาม
4. เพื่อทำการพัฒนาและเพิ่มความสามารถให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ ให้สามารถทำการปรับโหลดระหว่างไฟร์วอลล์แบบรักษาสถานะในกลุ่มที่ตนเองดูแลให้สมดุลย์ หลังจากที่ได้มีการตัดสินใจกระจายโหลดไปให้ไฟร์วอลล์แบบรักษาสถานะแต่ละตัวแล้ว

1.3 สมมุติฐานของการศึกษา

สาเหตุที่อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ไม่สามารถทำการย้ายโหลดจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งได้นั้น เพราะไฟร์วอลล์แบบรักษาสถานะจะไม่อนุญาตให้ทราฟฟิคของการเชื่อมต่อใด ๆ ที่มันยังไม่มีข้อมูลของการเชื่อมต่อนั้นในตารางสถานะของมันผ่านได้ ดังนั้นถ้าอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวที่ไม่มีข้อมูลของการเชื่อมต่อที่กำลังจะถูกย้ายมาอยู่ในตารางสถานะของมันแล้ว ไฟร์วอลล์แบบรักษาสถานะตัวนั้นจะไม่ยอมให้ทราฟฟิคของการเชื่อมต่อนั้นผ่าน

ถ้าเกิดเรานำไฟร์วอลล์แบบรักษาสถานะจากต่างผู้ผลิตกัน หรือเป็นไฟร์วอลล์แบบรักษา-สถานะจากผู้ผลิตเดียวกัน แต่ไม่สามารถทำการแลกเปลี่ยนข้อมูลในตารางสถานะระหว่างกันมาทำงานร่วมกัน ในกรณีนี้ไฟร์วอลล์แบบรักษาสถานะแต่ละตัวจะไม่สามารถรู้ข้อมูลในตารางสถานะของไฟร์วอลล์แบบรักษาสถานะตัวอื่น ดังนั้นอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จึงไม่สามารถทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งได้

จากปัญหาที่กล่าวมาจะเห็นว่าการที่อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ไม่สามารถทำการย้ายการเชื่อมต่อได้นั้น เกิดจากการที่ไฟร์วอลล์แบบรักษาสถานะที่เราจะทำการย้ายการเชื่อมต่อไปนั้นไม่มีข้อมูลของการเชื่อมต่อที่กำลังจะถูกย้ายอยู่ในตารางสถานะ ดังนั้นถ้าเราสามารถทำให้ไฟร์วอลล์แบบรักษาสถานะตัวที่เราจะทำการย้ายการเชื่อมต่อไป มีข้อมูลของการเชื่อมต่อที่กำลังจะถูกย้ายอยู่ในตารางสถานะได้แล้ว เราก็ย่อมที่จะสามารถทำการย้ายการเชื่อมต่อนั้นไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่ได้

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการศึกษา

งานวิจัยนี้มีแนวคิดมาจากการที่พบว่า ในปัจจุบันมีการนำอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์มาช่วยเพิ่มประสิทธิภาพในการส่งข้อมูลผ่านไฟร์วอลล์ให้มีความรวดเร็วขึ้น และเพิ่มเสถียรภาพให้กับระบบให้ดียิ่งขึ้น เพราะเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์แล้ว จะทำให้เราสามารถนำอุปกรณ์ไฟร์วอลล์หลายตัว ๆ จากหลาย ๆ ผู้ผลิตมาทำงานร่วมกัน ซึ่งไฟร์วอลล์แต่ละตัวจะทำงานแบ่งเบาภาระซึ่งกันและกัน และทำงานทดแทนกันในกรณีที่มีไฟร์วอลล์ตัวใดตัวหนึ่งไม่สามารถทำงานต่อได้

อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะทำหน้าที่กระจายโหลดหรืออีกนัยหนึ่งคือ กระจายการเชื่อมต่อไปให้กับไฟร์วอลล์ที่ตนเองดูแล โดยใช้อัลกอริทึมที่ผู้ดูแลระบบกำหนด หลังจากอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ได้ทำการกระจายการเชื่อมต่อไปให้ไฟร์วอลล์แล้ว อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะทำการส่งทราฟฟิคของการเชื่อมต่อนั้นผ่านไฟร์วอลล์ตัวเดิมเสมอจนกระทั่งการเชื่อมต่อนั้นสิ้นสุด เมื่ออุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทำการตรวจพบว่ามีไฟร์วอลล์ตัวใดตัวหนึ่งไม่สามารถทำงานต่อได้แล้ว อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะทำการตัดไฟร์วอลล์ตัวนั้นออกจากระบบ โดยจะไม่ทำการกระจายการเชื่อมต่อใหม่ไปให้ไฟร์วอลล์ตัวนั้นอีก

แม้ว่าอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะทำให้เราสามารถส่งข้อมูลได้รวดเร็วขึ้น และทำให้ระบบมีเสถียรภาพมากขึ้น แต่อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ก็ยังมีข้อด้อย เหมือนที่ได้กล่าวมาในหัวข้อความเป็นมาและความสำคัญของปัญหา ซึ่งงานวิจัยนี้จะนำเสนอวิธีที่ช่วยพัฒนาและเพิ่มความสามารถให้กับอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ เพื่อแก้ไขปัญหาเหล่านั้น

1.5 ขอบเขตการวิจัย

ขอบเขตของการศึกษาและการทำวิจัยนี้มีดังต่อไปนี้

1. ศึกษาการทำงานของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ และทำการเขียนโปรแกรมเพื่อจำลองการทำงานของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์
2. นำเสนอวิธีที่ช่วยทำให้สามารถทำการย้ายการเชื่อมต่อที่เคยถูกส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวที่หยุดทำงาน ไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวอื่นได้
3. นำเสนอวิธีที่ช่วยทำให้สามารถทำการย้ายการเชื่อมต่อที่เคยถูกส่งผ่านไฟร์วอลล์แบบรักษาตัวที่มีโหลดมาก ไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวที่มีโหลดน้อยกว่าได้
4. ทำการทดสอบว่าอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่มีในท้องตลาด ไม่สามารถทำการย้ายโหลดระหว่างไฟร์วอลล์แบบรักษาสถานะได้
5. เขียนโปรแกรมเพื่อจำลองการทำงานของวิธีที่นำเสนอ
6. ทำการทดสอบว่าวิธีที่นำเสนอสามารถใช้งานได้จริง
7. ทำการทดสอบว่าวิธีที่นำเสนอสามารถช่วยเพิ่มประสิทธิภาพในการส่งข้อมูลผ่านไฟร์วอลล์แบบรักษาสถานะ และสามารถช่วยเพิ่มเสถียรภาพของระบบให้ดีขึ้น
8. สรุปผลการศึกษาและวิจัย

1.6 ขั้นตอนของการศึกษา

การศึกษานี้เริ่มจากการทำความเข้าใจพื้นฐานของการส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ที่อยู่ในระบบเครือข่ายเสียก่อน เพื่อให้รู้ว่ารูปแบบและโครงสร้างของข้อมูลที่ถูกส่งระหว่างเครื่องคอมพิวเตอร์มีลักษณะเป็นอย่างไร มีโปรโตคอลใดบางที่ถูกนำมาใช้ในการสื่อสารระหว่างเครื่องคอมพิวเตอร์ที่อยู่ในระบบเครือข่าย

หลังจากนั้นเราก็มาเริ่มทำการศึกษาการทำงานของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ว่าอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทำหน้าที่อะไร มีคุณสมบัติและกระบวนการทำงานเป็นอย่างไร เพื่อที่จะได้นำข้อมูลมาทำการเขียนโปรแกรมจำลองการทำงานของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ ก่อนที่จะทำการเพิ่มความสามารถให้กับโปรแกรมเพื่อทำการแก้ไขข้อด้อยของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ต่อไป

เมื่อรู้ถึงวิธีการทำงานของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์แล้ว เราก็มาทำการศึกษาการทำงานของอุปกรณ์ไฟร์วอลล์แบบรักษาสถานะ ว่าอุปกรณ์ไฟร์วอลล์แบบรักษาสถานะมีกระบวนการทำงานเป็นอย่างไร เราจะสามารถทำการย้ายทราฟฟิคของการเชื่อมต่อใด ๆ ที่เคยส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งได้อย่างไร

เมื่อรู้วิธีที่จะทำให้สามารถทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งแล้ว ก็เริ่มเขียนโปรแกรมเพื่อจำลองการทำงานของวิธีที่ได้ตั้งสมมุติฐานไว้

สุดท้ายก็นำโปรแกรมที่เขียนมาทำการทดสอบกับอุปกรณ์จริง เพื่อทำการทดสอบว่าวิธีที่ได้ตั้งสมมุติฐานไว้สามารถใช้ได้จริง และทำการแสดงผลการทดสอบ พร้อมกับสรุปผลการศึกษาและวิจัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทที่ 2
ทฤษฎีเบื้องต้น

บทนี้จะกล่าวถึงทฤษฎีเบื้องต้นของระบบเครือข่ายคอมพิวเตอร์ เพื่อให้ผู้อ่านมีพื้นฐานในการทำความเข้าใจเนื้อหาในส่วนต่างต่อ ๆ ไป โดยเราจะกล่าวถึง สถาปัตยกรรมเครือข่าย โปรโตคอลระบบเครือข่ายแบบไคลเอ็นต์เซิร์ฟเวอร์ โปรโตคอลไอพี โปรโตคอลยูดีพี และโปรโตคอลทีซีพี

2.1 สถาปัตยกรรมเครือข่าย

การที่มนุษย์สามารถสื่อสารกันได้อย่างมีประสิทธิภาพนั้น เนื่องจากใช้ภาษาเดียวกัน เช่น ภาษาไทย ภาษาอังกฤษ เป็นต้น ถ้าใช้คนละภาษาก็คงจะสื่อสารกันไม่ได้ความ คอมพิวเตอร์ก็เช่นเดียวกันกับมนุษย์ การที่คอมพิวเตอร์เครื่องหนึ่งจะสามารถสื่อสารกับคอมพิวเตอร์อีกเครื่องหนึ่งได้จำเป็นที่ต้องใช้ "ภาษา" เดียวกันภาษาที่ว่านี้ศัพท์ทางคอมพิวเตอร์จะเรียกว่า "โปรโตคอล (Protocol)" ดังนั้นคอมพิวเตอร์ที่สื่อสารกันได้ต้องใช้โปรโตคอลเดียวกัน เช่น คอมพิวเตอร์ที่เชื่อมต่อเข้ากับอินเตอร์เน็ตจะใช้ "ภาษา" หรือโปรโตคอลทีซีพี/ไอพี (TCP/IP) ส่วนคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการเครือข่ายเน็ตแวร์ (NetWare) ก็จะใช้ "ภาษา" หรือโปรโตคอลไอพีเอ็กซ์/เอสพีเอ็กซ์ (IPX/SPX) ในการสื่อสารกัน เป็นต้น

ปัจจุบันฮาร์ดแวร์และซอฟต์แวร์ที่ใช้กับระบบคอมพิวเตอร์มีหลายชนิด บางชนิดก็ใช้งานร่วมกันได้แต่บางชนิดก็ใช้ด้วยกันไม่ได้เลย ผู้ใช้บางคนอาจมีความจำเป็นต้องสื่อสารกับผู้ใช้ที่เชื่อมต่อกับเครือข่ายอื่น เครือข่ายคอมพิวเตอร์ในปัจจุบันส่วนใหญ่จะแตกต่างกันทั้งฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ บางครั้งก็จะสื่อสารกันไม่ได้เนื่องจากเหตุผลที่ว่า คอมพิวเตอร์ที่อยู่ในเครือข่ายต่างประเภทกันจะใช้คนละ "ภาษา" กัน ด้วยเหตุนี้จึงได้มีการพัฒนา "ภาษา"หรือโปรโตคอลขึ้นมาเพื่อให้คอมพิวเตอร์ที่กล่าวมานี้สามารถสื่อสารกันได้

2.2 โปรโตคอล (Protocol)

การเชื่อมต่อคอมพิวเตอร์ให้เป็นเครือข่ายด้วยสายสัญญาณนั้นเป็นขั้นตอนที่ง่าย แต่ส่วนที่ท้าทายคือ การพัฒนามาตรฐานเพื่อให้คอมพิวเตอร์และอุปกรณ์เครือข่ายที่ผลิตโดยบริษัทต่าง ๆ สามารถติดต่อสื่อสารกันได้ ซึ่งมาตรฐานนี้ก็คือ โปรโตคอล หรือสรุปสั้นๆ โปรโตคอลคือ กฎ และขั้นตอนการสื่อสาร และรูปแบบของข้อมูลที่ใช้ในการสื่อสารระหว่างคอมพิวเตอร์สองเครื่องใด ๆ ที่เชื่อมต่อกันเป็นเครือข่าย

ตัวอย่างที่เห็นได้ชัดของโปรโตคอล เช่น การสื่อสารโดยใช้โทรศัพท์ ซึ่งจะมีขั้นตอนต่าง ๆ ที่ต้องทำก่อนที่จะพูดคุยกันได้ เช่น โดยส่วนใหญ่คำแรกที่พูดเมื่อใช้โทรศัพท์คือ "ฮัลโหล" หรือคำทักทายของภาษาท้องถิ่นอื่น ๆ การทักทายกันนี้เป็นสัญญาณให้คู่สนทนาทราบว่าการเชื่อมต่อกันสำเร็จแล้ว ขั้นตอนต่อไปคือ อีกฝ่ายจะตอบด้วยคำว่า "ฮัลโหล" เช่นกัน ซึ่งจะเป็นสัญญาณบอกให้ทราบอีกว่าการติดต่อสื่อสารเป็นไปได้ทั้งสองทาง ถ้าทั้งสองฝ่ายที่สนทนากันรู้จักกันมาก่อนการสนทนาก็จะเข้าสู่เรื่องได้ทันที แต่ถ้าหากว่าทั้งสองฝ่ายยังไม่รู้จักกันก็จะมีขั้นตอนอื่นเพิ่มอีก เพื่อช่วยให้ทั้งสองฝ่ายรู้จักกันก่อนที่จะเริ่มเรื่องที่จะสนทนากันจริง ๆ

การสนทนากันของคอมพิวเตอร์ก็มีขั้นตอนไม่ได้แตกต่างจากตัวอย่างข้างต้นมากนัก โดยขั้นตอนและรูปแบบที่ใช้ในการสนทนาหรือว่าสื่อสารต้องเข้าใจตรงกัน โดยคอมพิวเตอร์ที่จะทำการสื่อสารกัน จึงจะทำให้คอมพิวเตอร์สามารถทำการสื่อสารกันได้สำเร็จ ซึ่งขั้นตอนในการสื่อสารที่เข้าใจกันระหว่างคอมพิวเตอร์นี้ก็คือ โปรโตคอลนั่นเอง

โปรโตคอลของเครือข่ายเมื่อมารวม ๆ กันแล้ว อาจเรียกว่า "สถาปัตยกรรมเครือข่าย (Network Architecture)" เนื่องจากระบบเครือข่ายคอมพิวเตอร์ในปัจจุบันเป็นระบบที่ซับซ้อนมาก ทำให้ยากต่อการออกแบบโดยคน ๆ เดียวหรือคนกลุ่มเดียว เพื่อให้การพัฒนาระบบเป็นไปอย่างมีประสิทธิภาพและง่ายขึ้น จึงมีการแบ่งโปรโตคอลออกเป็นชั้น ๆ หรือเลเยอร์ (Layer) การทำงานในแต่ละเลเยอร์จะไม่ซ้ำซ้อนกัน ซึ่งเลเยอร์ที่อยู่ต่ำกว่าจะทำหน้าที่ให้บริการ (Service) กับเลเยอร์ที่อยู่สูงกว่า โดยเลเยอร์ที่อยู่สูงกว่าไม่จำเป็นต้องทราบถึงรายละเอียดว่าเลเยอร์ที่อยู่ต่ำกว่ามีวิธีให้บริการอย่างไร เพียงแค่รู้ว่ามีบริการอะไรบ้าง และแต่ละบริการคืออะไรก็เพียงพอ ซึ่งแนวความคิดนี้จะเรียกว่า "เทคโนโลยีเลเยอร์ (Layer Technology)"

2.3 ระบบเครือข่ายแบบไคลเอ็นต์เซิร์ฟเวอร์

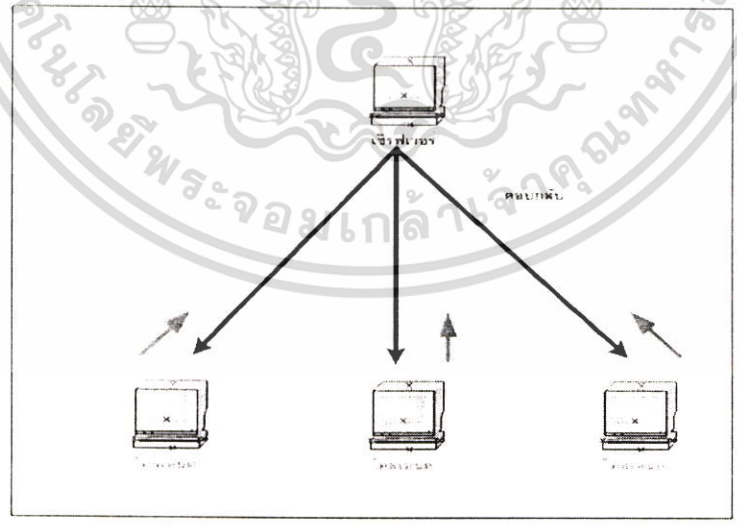
ระบบเครือข่ายประเภทนี้มีการใช้คอมพิวเตอร์เครื่องหนึ่งเป็นตัวหลัก ทำหน้าที่ให้บริการเกี่ยวกับข้อมูลข่าวสาร รวมทั้งจัดแบ่งปันข้อมูลแก่คอมพิวเตอร์เครื่องอื่น ๆ ที่ทำตนเป็นลูกข่าย ส่วนคอมพิวเตอร์อื่น ๆ ที่ไม่ใช่คอมพิวเตอร์หลักจะถูกเรียกว่าไคลเอนต์ ซึ่งมีหน้าที่ร้องขอได้หลาย ๆ อย่าง นับตั้งแต่การร้องขออนุญาตให้มีสิทธิ์เข้าสู่การใช้งานเครือข่าย จนถึงการร้องขอสิทธิ์ในการใช้งานทรัพยากร ไม่ว่าจะเป็นอุปกรณ์จัดเก็บข้อมูล แฟ้มข้อมูล หรือเครื่องพิมพ์ที่ติดตั้งบนเครือข่าย เป็นต้น

เครื่องคอมพิวเตอร์ที่เป็นตัวหลักจะต้องได้รับการติดตั้งระบบปฏิบัติการเครือข่าย (NOS หรือ Network Operating System) เป็นการเฉพาะเจาะจง เพื่อให้สามารถบริการแบ่งปันข้อมูลและทรัพยากรแก่ไคลเอนต์ทุกเครื่องบนเครือข่าย อีกทั้งยังสามารถดูแลระบบรักษาความปลอดภัยและบริหารจัดการทรัพยากรต่าง ๆ ได้

อย่างไรก็ดีระบบเครือข่ายประเภทนี้จะต้องมีผู้ดูแลคอมพิวเตอร์ เพื่อที่จะทำหน้าที่บริหารจัดการเกี่ยวกับแฟ้มข้อมูล รวมทั้งการกำหนดสิทธิ์การใช้งานของผู้ใช้งาน (User) ที่มีต่อคอมพิวเตอร์เครื่องนี้ เราเรียกคอมพิวเตอร์ที่ทำหน้าที่ให้บริการทรัพยากร ไม่ว่าจะเป็นแฟ้มข้อมูล อุปกรณ์จัดเก็บข้อมูล หรือเครื่องพิมพ์ว่าเซิร์ฟเวอร์ ซึ่งมีชื่อเรียกแตกต่างกันออกไปหลายชื่อ เช่น คอมพิวเตอร์หลักที่ให้บริการแบ่งปันการใช้งานไฟล์เพียงอย่างเดียวเราเรียกว่า "ไฟล์เซิร์ฟเวอร์ (File Server)" ส่วนคอมพิวเตอร์หลักที่ให้บริการข้อมูลข่าวสารเกี่ยวกับฐานข้อมูลเราเรียกว่า "เซิร์ฟเวอร์ฐานข้อมูล (Database Server)" และเรียกคอมพิวเตอร์หลักที่ให้บริการเกี่ยวกับข้อมูลของเว็บว่า "เว็บเซิร์ฟเวอร์ (Web Server)" เป็นต้น

คอมพิวเตอร์ที่เป็นเซิร์ฟเวอร์ทำหน้าที่เป็นผู้ให้บริการแบ่งปันข้อมูลหรือข่าวสาร สามารถทำหน้าที่ 2 แบบได้แก่ การให้บริการข้อมูล และให้บริการจัดเก็บข้อมูลตามที่ลูกข่ายส่งเข้ามาที่เซิร์ฟเวอร์ รูปแบบของทราฟฟิคของข้อมูลระหว่างเซิร์ฟเวอร์กับคอมพิวเตอร์ลูกข่ายหรือไคลเอนต์จึงมีอยู่ 2 แบบดังนี้

1. ทราฟฟิคที่มาจากเซิร์ฟเวอร์เป็นส่วนใหญ่ หมายถึงการร้องขอข้อมูลของไคลเอนต์ที่มีต่อเซิร์ฟเวอร์ โดยลักษณะเช่นนี้ทราฟฟิคที่ประกอบด้วยคำขอบริการจากเซิร์ฟเวอร์จึงมีเพียงนิดเดียว แต่ได้ข้อมูลกลับมาจากเซิร์ฟเวอร์ในปริมาณมากกว่า
2. ทราฟฟิคที่มาจากไคลเอนต์เป็นส่วนใหญ่ หมายถึงการป้อนข้อมูลจากไคลเอนต์มายังเซิร์ฟเวอร์เป็นส่วนใหญ่ ซึ่งสามารถเกิดขึ้นพร้อม ๆ กันหลายจุด ดังนั้นทราฟฟิคส่วนใหญ่จะมาจากไคลเอ็นต์เป็นหลัก



รูปที่ 2.1 ลักษณะการไหลเวียนของข้อมูลข่าวสารบนระบบเครือข่ายไคลเอ็นต์เซิร์ฟเวอร์

2.4 โปรโตคอลไอพี (IP protocol)

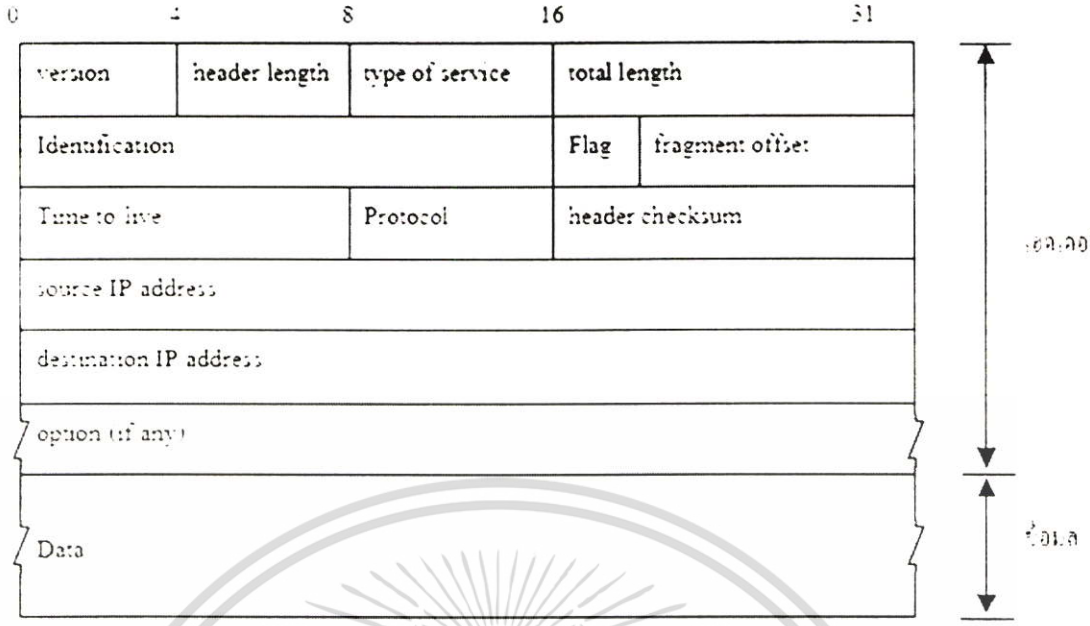
บทบาทหน้าที่ของชั้นเครือข่ายในระบบสื่อสารคอมพิวเตอร์คือ การนำส่งแพ็คเก็ตที่บรรจุข้อมูลของโปรโตคอลในชั้นที่สูงกว่า เช่น โปรแกรมแอพพลิเคชั่นจากโฮสต์ต้นทาง ผ่านสวิตชิ่งโนด (switching node) ของเครือข่ายให้ถึงมือของโฮสต์ปลายทางได้อย่างถูกต้อง ซึ่งโฮสต์ทั้งสองอาจจะอยู่ในบริเวณขอบเขตของเครือข่ายเดียวกัน หรืออาจจะอยู่ห่างไกลกันในอีกซีกหนึ่งของโลก การทำงานในส่วนนี้จึงจัดว่าเป็นองค์ประกอบหลักของระบบสื่อสารคอมพิวเตอร์

เราเตอร์จัดเป็นอุปกรณ์สื่อสารที่มีบทบาทสำคัญในการทำงานของชั้นโปรโตคอลไอพีคือ ทำหน้าที่ในการส่งผ่านข้อมูลของผู้ใช้ในรูปของไอพีดาต้าแกรมจากโฮสต์ต้นทางผ่านเราเตอร์อื่นในระบบและนำส่งไปให้โฮสต์ปลายทาง กระบวนการในการตัดสินใจเลือกเส้นทางในการส่งไอพีดาต้าแกรมแต่ละตัวจึงเป็นประเด็นหลัก ที่ต้องได้รับการพิจารณาและการออกแบบอย่างถูกต้องและมีประสิทธิภาพ เพื่อให้การรับส่งไอพีดาต้าแกรมมีความรวดเร็ว และถึงมือผู้ใช้ปลายทางโดยมีความผิดพลาดน้อยที่สุด

ในหัวข้อนี้จะกล่าวถึงรูปแบบโครงสร้างของไอพีดาต้าแกรม และการใช้งานขององค์ประกอบแต่ละส่วน เพื่อเป็นพื้นฐานในการทำความเข้าใจของเนื้อหาในส่วนต่อไป

2.4.1 โครงสร้างของไอพีดาต้าแกรม

พิจารณารูปแบบโครงสร้างของไอพีดาต้าแกรมในรูปที่ 2.2 จะเห็นว่าไอพีดาต้าแกรมมีองค์ประกอบหลัก 2 ส่วน คือ ส่วนที่เรียกว่าเฮดเดอร์ (header) และส่วนของข้อมูล (data) สำหรับในส่วนนี้จะขอกล่าวถึงองค์ประกอบ 4 ฟิลด์แรกก่อน และจึงจะอธิบายถึงฟิลด์อื่นในส่วนต่อไป



รูปที่ 2.2 โครงสร้างของไอพีดาต้าแกรม (IP datagram)

ฟิลด์ version ขนาด 4 บิตเป็นฟิลด์แรกของไอพีดาต้าแกรมที่ใช้ระบุถึงรุ่นหรือเวอร์ชันของโปรโตคอลไอพีที่ใช้ในการสร้างไอพีดาต้าแกรม ฟิลด์นี้มีไว้สำหรับแยกแยะระหว่างโปรโตคอลไอพีรุ่นเก่ากับโปรโตคอลไอพีรุ่นใหม่กว่า ในกรณีที่มีการพัฒนาปรับปรุงโปรโตคอลเวอร์ชันใหม่ขึ้น ทั้งนี้เพื่อให้แน่ใจว่าซอฟต์แวร์ของโปรโตคอลไอพีทั้งสองฝ่ายติดต่อสื่อสารโดยใช้เวอร์ชันเดียวกัน ปัจจุบันโปรโตคอลที่ใช้อยู่เป็นโปรโตคอลไอพีเวอร์ชัน 4

ฟิลด์ header length (HLEN) ขนาด 4 บิต เป็นฟิลด์ที่ระบุถึงขนาดเฮดเดอร์ของไอพีดาต้าแกรม โดยจะบอกในหน่วยของเวิร์ดขนาด 32 บิต เนื่องจาก header length มีขนาดเพียง 4 บิต เฮดเดอร์ของไอพีดาต้าแกรมจึงมีความยาวไม่เกิน 60 ไบต์ สังเกตว่าฟิลด์แทบทั้งหมดที่บรรจุอยู่ในเฮดเดอร์ของไอพีดาต้าแกรมมีขนาดที่ตายตัว ยกเว้นแต่ฟิลด์ option เท่านั้น ซึ่งมีขนาดที่แปรเปลี่ยนได้ ในสภาพการณ์โดยทั่วไปจะไม่มีการใช้งาน option บ่อยครั้งนัก ดังนั้น header length จึงมักกำหนดให้มีค่าเท่ากับ 5 ซึ่งเทียบเท่ากับ 20ไบต์

ฟิลด์ type of service (TOS) มีขนาด 8 บิต ใช้สำหรับบ่งถึงคุณลักษณะหรือรูปแบบการให้บริการที่ไอพีดาต้าแกรมต้องการ สำหรับรายละเอียดของฟิลด์นี้จะกล่าวถึงในส่วนถัดไป

ฟิลด์ total length ขนาด 16 บิต มีไว้เพื่อบ่งถึงขนาดของไอพีดาต้าแกรมทุกส่วนรวมกันในหน่วยของไบต์ ดังนั้นไอพีดาต้าแกรมจึงมีขนาดสูงสุดได้ไม่เกิน $2^{16}-1$ หรือ 65,535 ไบต์ เมื่อระบบได้อ่านค่าในฟิลด์ header length ก็จะทราบจุดเริ่มต้นของส่วนฟิลด์ data ได้ และเมื่อได้อ่านค่าฟิลด์ total length ด้วยก็จะทราบจุดสิ้นสุดหรือขนาดของฟิลด์ data ได้ทันที ฟิลด์นี้มีความสำคัญ

และจำเป็นต้องมีเมื่อใช้งานกับบางเครือข่ายที่ชั้นดาต้าลิงก์ (data link layer) กำหนดขนาดต่ำสุดของเฟรมไว้ เช่น อีเทอร์เน็ต (Ethernet) ซึ่งกำหนดให้ข้อมูลหรือไอพีดาต้าแกรมที่บรรจุลงในเฟรมต้องมีขนาดอย่างต่ำ 46 ไบต์ ถ้าข้อมูลที่ใส่มีขนาดเล็กกว่านี้ดาต้าลิงก์ก็จะเพิ่ม padding ต่อท้ายจนครบจำนวน เนื่องจากไอพีดาต้าแกรมอาจจะมีขนาดเล็กกว่า 46 ไบต์ ดังนั้นถ้าไม่มีฟิลด์ total length กำกับเราจะไม่สามารถรู้ได้ว่าส่วนข้อมูลที่ดึงออกมาจากเฟรมอีเทอร์เน็ตมีกี่ไบต์ที่เป็นไอพีดาต้าแกรมจริง ๆ

2.4.2 ชนิดการให้บริการ

จากโครงสร้างของไอพีดาต้าแกรมในรูปที่ 2.3 ฟิลด์ type of service (TOS) มีขนาด 8 บิต ใช้สำหรับบ่งถึงคุณลักษณะหรือรูปแบบการให้บริการที่ไอพีดาต้าแกรมต้องการ เพื่อให้อุปกรณ์เราเตอร์เลือกส่งไอพีดาต้าแกรมในเส้นทางที่เหมาะสมสอดคล้องกับความต้องการของไอพีดาต้าแกรมนั้นๆ สำหรับโครงสร้างของฟิลด์นี้ประกอบด้วยส่วนต่าง ๆ ในรูปที่ 2.3 ส่วนแรก คือ ส่วน precedence จำนวน 3 บิต ใช้สำหรับจัดลำดับความสำคัญของไอพีดาต้าแกรมซึ่งมีได้ 8 ระดับ ตั้งแต่ระดับ 0 ซึ่งมีความสำคัญน้อยสุดใช้สำหรับไอพีดาต้าแกรมข้อมูลปกติไปจนถึงระดับ 7 ที่มีความสำคัญสูงสุดสำหรับไอพีดาต้าแกรมที่ส่งคำสั่งควบคุมในเครือข่าย การจัดลำดับความสำคัญจะมีประโยชน์อย่างมากกับการแก้ปัญหาบางอย่างในระบบ เช่น ปัญหาความแออัดคับคั่ง (congestion) เพราะในสภาวะที่มีความคับคั่งเกิดขึ้น ไอพีดาต้าแกรมที่ส่งคำสั่งควบคุมจะยังคงสามารถส่งผ่านส่วนต่าง ๆ ของเครือข่ายได้ดีเพราะได้รับการจัดลำดับให้มีความสำคัญสูง จึงไม่ได้รับผลกระทบที่รุนแรงจากความแออัด การส่งข่าวสารคำสั่งควบคุมเพื่อแก้ปัญหาจึงกระทำได้ราบรื่น เมื่อเทียบกับในระบบที่ไม่มีการจัดลำดับความสำคัญ ไอพีดาต้าแกรมที่ส่งคำสั่งควบคุมจะติดขัดอยู่ในบริเวณที่มีความแออัด ทำให้การแก้ปัญหายากลำบากขึ้น แต่อย่างไรก็ตามในปัจจุบันส่วน precedence ยังมิได้มีการนำมาใช้งานในทางปฏิบัติมากนัก

0	1	2	3	4	5	6	7
Precedence			D	T	R	C	unused

รูปที่ 2.3 โครงสร้างของฟิลด์ type of service (TOS)

บิตอีก 4 บิตถัดมาคือ D , T , R และ C มีไว้เพื่อแสดงให้เราเตอร์ทราบถึงคุณภาพของการให้บริการที่ไอพีดาต้าแกรมต้องการซึ่งได้แก่ minimize delay , maximize throughput , maximize reliability และ minimize monetary cost ตามลำดับ สำหรับบิตสุดท้ายยังไม่มีการกำหนดหน้าที่การใช้งาน (unused) การใช้งานบิต D , T , R และ C มีคุณลักษณะดังนี้

1. เซต D = 1 เมื่อต้องการให้เราเตอร์ส่งผ่านไอพีดาต้าแกรมบนเส้นทางที่มีเวลาประวิงในการรับส่งต่ำสุด เช่น หากเราเตอร์หนึ่งมีทางเลือกในการส่งผ่านไอพีดาต้าแกรมได้สองทาง คือ ผ่านช่องสัญญาณเช่าความเร็วต่ำ (low speed leased line) หรือผ่านช่องสื่อสารดาวเทียม (satellite link) ที่มีอัตราการส่งสูงแต่มีเวลาประวิงเนื่องจากการแพร่ของสัญญาณมาก เมื่อได้รับไอพีดาต้าแกรมที่เซต D = 1 เราเตอร์ก็จะเลือกส่งต่อไอพีดาต้าแกรมดังกล่าวผ่านช่องสัญญาณเช่าความเร็วต่ำ
2. เซต T = 1 เมื่อต้องการให้เราเตอร์ส่งผ่านไอพีดาต้าแกรมบนเส้นทางที่มีความจุสูงสุด ยกตัวอย่าง เช่น ถ้าเราเตอร์ในตัวอย่างก่อนหน้าได้รับไอพีดาต้าแกรมที่เซต T = 1 เราเตอร์ก็จะเลือกส่งไอพีดาต้าแกรมผ่านช่องสื่อสารดาวเทียม เนื่องจากสามารถส่งข้อมูลได้ปริมาณมาก
3. เซต R = 1 เมื่อต้องการส่งผ่านเส้นทางที่มีความเชื่อถือสูง คือ มีอัตราความผิดพลาดต่ำ
4. เซต C = 1 เมื่อต้องการส่งผ่านเส้นทางที่มีค่าใช้จ่ายน้อยสุด

จากที่กล่าวมาจะเห็นว่าเวลาที่จะเซตบิตเหล่านี้จะเลือกเซตเพียงบิตใดบิตหนึ่งเท่านั้น เพราะหากเซตมากกว่าหนึ่งบิตจะไม่เกิดประโยชน์แต่อย่างใด เนื่องจากการตอบสนองความต้องการหลายอย่างพร้อมกันมักจะทำได้ยาก และจริง ๆ แล้วความต้องการที่เซตหรือระบุขึ้นนี้อาจจะไม่ได้รับการตอบสนองเลยก็เป็นได้ หากส่วนของเครือข่ายที่เกี่ยวข้องไม่มีโครงสร้างและคุณสมบัติที่พร้อมหรือเพียงพอที่จะปฏิบัติตามความต้องการของไอพีดาต้าแกรมนั้น ๆ ด้วยเหตุนี้การใช้งานของฟิลด์ TOS จึงเป็นเพียงแนวทางที่เราเตอร์อาจจะนำหรือไม่นำมาใช้ประกอบการพิจารณาเลือกเส้นทางการส่งไอพีดาต้าแกรม ที่ผ่านมาการใช้งานฟิลด์ TOS ยังไม่มีมากนัก แต่กระนั้นอุปกรณ์เราเตอร์รุ่นใหม่ได้เริ่มมีการใช้งานส่วนนี้มากขึ้น

สำหรับตัวอย่างการเซตบิต D , T , R และ C ที่ได้รับการแนะนำเพื่อใช้งานกับโปรแกรมแอพพลิเคชั่นในทางปฏิบัติที่น่าสนใจ ได้แก่ การเซต D = 1 ให้แก่โปรแกรมเทลเน็ต/อาร์ลอก์อิน (Telnet/Rlogin) เพราะการใช้งานโปรแกรมเหล่านี้ต้องการการโต้ตอบที่ฉับพลันทันเวลา การเซต T = 1 ให้กับการรับส่งข้อมูลของโปรแกรมเอฟทีพี เพื่อให้การส่งผ่านข้อมูลขนาดใหญ่เสร็จภายในเวลาอันสั้น การเซต R = 1 ให้กับโปรแกรมเอสเอ็นเอ็มพี เนื่องจากต้องการให้ข้อมูลที่ส่งผ่านเครือข่ายมีความถูกต้องเชื่อถือได้มากที่สุด เพราะเอสเอ็นเอ็มพีเป็นโปรแกรมที่มีหน้าที่ดูแลจัดการเครือข่ายดูตัวอย่างการเซตบิตดังกล่าวกับโปรแกรมอื่นๆ ได้ในตารางที่ 2.1

ตารางที่ 2.1 ตัวอย่างการเซตบิตดังกล่าวกับโปรแกรมอื่น ๆ

ICMP	0	0	0	0	ปกติ
BOOTP	0	0	0	0	ปกติ
NNTP	0	0	0	1	ค่าใช้จ่ายต่ำสุด
IGP	0	0	1	0	ความเชื่อถือสูงสุด
SNMP	0	0	1	0	ความเชื่อถือสูงสุด
Telnet	1	0	0	0	เวลาประวิงต่ำสุด
FTP (data)	0	1	0	0	อัตราส่งสูงสุด
FTP (Control)	1	0	0	0	เวลาประวิงต่ำสุด
TFTP	1	0	0	0	เวลาประวิงต่ำสุด
SMTP (command)	1	0	0	0	เวลาประวิงต่ำสุด
SMTP (Data)	0	1	0	0	อัตราส่งสูงสุด
DNS (UDP query)	1	0	0	0	เวลาประวิงต่ำสุด
DNS (TCP query)	0	0	0	0	ปกติ
DNS (Zone)	0	1	0	0	อัตราส่งสูงสุด

2.4.3 การแฟรกเมนต์

สำหรับฟิลด์อีก 3 ฟิลด์ตามโครงสร้างไอพีดาต้าแกรมในรูปที่ 2.2 ได้แก่ ฟิลด์ identification ฟิลด์ flag และฟิลด์ fragment offset มีหน้าที่และบทบาทร่วมกันสำหรับการส่งไอพีดาต้าแกรมผ่านโครงสร้างทางกายภาพของเครือข่ายคอมพิวเตอร์ที่แตกต่างกัน ก่อนที่จะกล่าวถึงหน้าที่ของแต่ละฟิลด์ จะขออธิบายถึงปัญหาหรือความจำเป็นที่ต้องมีฟิลด์เหล่านี้ก่อน โดยปกติข้อมูลของผู้ใช้ที่กำเนิดจากเครื่องคอมพิวเตอร์มักจะมีขนาดไม่แน่นอน การบรรจุข้อมูลเหล่านี้ลงในไอพีดาต้าแกรมแต่ละครั้งจะมีขนาดสูงสุดไม่เกิน 65,535 ไบต์ เพราะถูกจำกัดโดยฟิลด์ total length ที่มีขนาดเท่ากับ 16 บิต โดยทั่วไปขั้นตอนการสร้างและรับไอพีดาต้าแกรมจะทำโดยโปรแกรมซอฟต์แวร์ ทำให้การกำหนดขนาดของไอพีดาต้าแกรมสามารถทำได้ตามแต่จะเลือก ในขณะที่การส่งไอพีดาต้าแกรมผ่านเครือข่ายทางกายภาพจริงนั้นจะถูกรองรับโดยโปรโตคอลในชั้นดาต้าลิงก์ ซึ่งในชั้นนี้จะบรรจุไอพีดาต้าแกรมลงในรูปของเฟรมเพื่อส่งผ่านช่องสัญญาณ โดยปกติเพื่อให้ได้ประสิทธิภาพสูงสุดเรามักจะกำหนดขนาดไอพีดาต้าแกรมให้มีขนาดใหญ่ใกล้เคียง หรือเท่ากับขนาดของเฟรมที่กำหนดโดยชั้นดาต้าลิงก์

เนื่องจากระบบอินเทอร์เน็ตอนุญาตให้ใช้โปรโตคอลชั้นดาต้าลิงก์ได้หลากหลายรูปแบบ เช่น อีเทอร์เน็ต หรือโปรโตคอลเอฟดีดีไอ เป็นต้น โปรโตคอลเหล่านี้มักมีข้อจำกัดในเรื่องขนาดของเฟรมที่แตกต่างกันออกไป ยกตัวอย่างเช่น อีเทอร์เน็ตจำกัดขนาดของข้อมูลที่จะใส่ลงในเฟรมได้ไม่เกิน 1,500 ไบต์ ในขณะที่โปรโตคอลเอฟดีดีไออนุญาตให้ใส่ข้อมูลได้ไม่เกิน 4,470 ไบต์ในหนึ่งเฟรม ทั้งนี้เราเรียกข้อจำกัดขนาดของเฟรมนี้ว่าเอ็มทียู (maximum transfer unit) ด้วยเหตุนี้การกำหนดขนาดที่แน่นอนตายตัวของไอพีดาต้าแกรมค่าหนึ่งที่เหมาะสมและมีประสิทธิภาพ เพื่อบรรจุลงในแต่ละเฟรมจึงเป็นไปไม่ได้ เพราะไอพีดาต้าแกรมของผู้ใช้อาจจะได้รับการส่งผ่านเส้นทางที่แตกต่างกันผ่านเครือข่ายหลายประเภทและเปลี่ยนไปตามจุดหมายปลายทางที่ไม่เหมือนกัน ดังนั้นหากให้กำเนิดไอพีดาต้าแกรมที่มีขนาดใหญ่กว่าเอ็มทียูของโปรโตคอลในชั้นดาต้าลิงก์เพียงบางส่วนของเครือข่าย ระบบจำต้องแบ่งไอพีดาต้าแกรมนั้นออกเป็นส่วนย่อยเพื่อให้สามารถบรรจุลงในเฟรมซึ่งแน่นอนว่าต้องใช้มากกว่าหนึ่งเฟรม เราเรียกกระบวนการทำเช่นนี้ว่า การแฟรกเมนต์ (fragmentation) โดยหลักการแล้วหากต้องการหลีกเลี่ยงการแฟรกเมนต์ ก็สามารถทำได้โดยระบบจะต้องเลือกไอพีดาต้าแกรมที่มีขนาดเล็กกว่าหรือเท่ากับเอ็มทียูที่เล็กที่สุด การทำเช่นนี้ส่งผลให้การส่งผ่านไอพีดาต้าแกรมบนเครือข่ายส่วน อื่น ๆ ที่มีเอ็มทียูขนาดใหญ่กว่าด้อยประสิทธิภาพลงอย่างมาก ด้วยเหตุนี้โปรโตคอลไอพีจึงแยกการกำหนดขนาดของไอพีดาต้าแกรมกับขนาดของเฟรมให้อิสระจากกัน นั่นคือแหล่งกำเนิดข้อมูลสามารถกำหนดขนาดของไอพีดาต้าแกรมได้ตามต้องการ และปล่อยให้เป็นหน้าที่ของอุปกรณ์เราเตอร์แต่ละตัวระหว่างทางที่ต้องตรวจอินเทอร์เฟซของตัวมันกับชั้นดาต้าลิงก์เองว่าไอพีดาต้าแกรมมีขนาดใหญ่เกินกว่าเฟรมหนึ่งเฟรมจนต้องมีการทำแฟรกเมนต์หรือไม่ หากจำเป็นเราเตอร์จะแบ่งแยกไอพีดาต้าแกรมออกเป็นส่วน ๆ และตั้งค่า identification flag และ fragment offset ในส่วนของเฮดเดอร์ของแต่ละดาต้าแกรมให้ถูกต้องดังนี้

ฟิลด์ identification เริ่มแรกจะพิจารณาฟิลด์ identification ก่อนซึ่งมีขนาด 16 บิต ทุกครั้งที่โฮสต์ต้นทางมีการส่งไอพีดาต้าแกรมออกไปแต่ละดาต้าแกรม โฮสต์จะมีการกำหนดค่าในฟิลด์ identification ไม่ให้ซ้ำกับครั้งก่อนๆ การทำเช่นนี้จึงเหมือนเป็นการกำหนดหมายเลขประจำตัวของไอพีดาต้าแกรมแต่ละดาต้าแกรมนั่นเอง เมื่อไอพีดาต้าแกรมนี้เดินทางถึงเราเตอร์หนึ่งและพบว่าจำเป็นต้องมีการทำแฟรกเมนต์ เราเตอร์จะบรรจุค่า identification เดียวกันกับที่ได้รับให้กับแฟรกเมนต์ทุกตัวของไอพีดาต้าแกรมเหมือนกันหมด จะเห็นว่าฟิลด์นี้มีประโยชน์ต่อการทำแฟรกเมนต์โดยตรง เพราะมีหน้าที่บ่งบอกให้ปลายทางทราบว่าแฟรกเมนต์ที่ได้รับแต่ละส่วนเป็นองค์ประกอบหนึ่งของไอพีดาต้าแกรมเดียวกันหรือไม่ หากใช่ภาครับก็จะนำมาประกอบรวมกันใหม่ให้สมบูรณ์

ฟิลด์ flag ส่วนฟิลด์ flag เป็นฟิลด์ที่มีขนาด 3 บิต บิตแรกไม่มีการใช้งานและกำหนดให้เป็น 0 เสมอ บิตที่สองเรียกว่า บิต D มีไว้เพื่อกำหนดว่าไอพีดาต้าแกรมนี้อนุญาตให้ทำแฟรกเมนต์ได้หรือไม่ ถ้าโฮสต์ต้นทางกำหนดให้ D = 0 ก็หมายถึงอนุญาตให้อุปกรณ์เราเตอร์ระหว่างทางทำการ

แฟรกเมนต์ได้ถ้ามีความจำเป็น แต่หากเซต D = 1 หมายความว่าห้ามมิให้ทำการแฟรกเมนต์โดยเด็ดขาด ในกรณีหลังนี้ถ้าเราเตอร์ไม่สามารถส่งต่อไอพีดาต้าแกรมได้หากปราศจากการทำแฟรกเมนต์ เราเตอร์ก็จะทิ้งไอพีดาต้าแกรมนั้นไป พร้อมกับแจ้งความผิดพลาดที่เกิดขึ้นกลับไปยังโฮสต์ต้นทางโดยอาศัยโปรโตคอลไอซีเอ็มพี (ICMP) ส่วนบิตสุดท้ายบิตที่สามเรียกว่า บิต M เป็นบิตที่เซตค่าโดยเราเตอร์เมื่อมีการทำแฟรกเมนต์กับไอพีดาต้าแกรมนั้น โดยจะตั้งค่า M = 0 เพื่อหมายถึงว่าแฟรกเมนต์นั้นเป็นส่วนสุดท้ายของไอพีดาต้าแกรม และจะเซตให้ M = 1 เพื่อแสดงว่ายังมีแฟรกเมนต์อื่นตามมาอีก เพราะฉะนั้นฟิลด์ flag จึงเป็นตัวระบุให้โฮสต์ปลายทางทราบจุดสิ้นสุดของไอพีดาต้าแกรม มีข้อสังเกต ณ ตรงนี้เรื่องหนึ่ง คือ แฟรกเมนต์แต่ละส่วนอาจจะถูกส่งผ่านเครือข่ายด้วยเส้นทางที่แตกต่างกัน และแฟรกเมนต์เหล่านี้อาจเดินทางมาถึงจุดหมายปลายทางในลำดับที่ผิดไปจากเดิมได้ ดังนั้นหากโฮสต์ปลายทางได้รับแฟรกเมนต์หนึ่งที่บ่งว่าเป็นแฟรกเมนต์สุดท้าย ก็มิได้หมายความว่าโฮสต์ปลายทางได้รับแฟรกต์เมนต์ของไอพีดาต้าแกรมนั้นครบสมบูรณ์ทุกแฟรกเมนต์ แท้จริงแล้วการใช้ฟิลด์เพียงสองฟิลด์ คือ identificaiton และ flag จะไม่เพียงพอสำหรับโฮสต์ปลายทางที่จะนำแฟรกเมนต์มาประกอบกันได้อย่างถูกต้อง เพราะขาดข้อมูลที่บอกถึงลำดับการเรียงต่อของแฟรกเมนต์ ปัญหานี้แก้ไขได้โดยอาศัยฟิลด์ fragment offset ที่จะได้กล่างถึงต่อไป

ฟิลด์ fragment offset สำหรับฟิลด์ fragment offset มีหน้าที่ชี้หรือระบุตำแหน่งเริ่มต้นของส่วนย่อยแต่ละส่วนภายในไอพีดาต้าแกรม ฟิลด์นี้มีขนาด 13 บิต โดยค่าที่ใช้มีหน่วยเป็นจำนวนเท่าของ 8 ไบต์ เมื่อโฮสต์ปลายทางอ่านค่าฟิลด์นี้ประกอบกับฟิลด์ total length ของแฟรกเมนต์ที่ได้รับแต่ละตัว ก็จะสามารถตรวจสอบได้ว่าได้รับแฟรกเมนต์ของไอพีดาต้าแกรมครบถ้วนหรือไม่

ในกรณีที่แฟรกเมนต์เพียงส่วนเดียวของไอพีดาต้าแกรมสูญหายระหว่างทาง โฮสต์ปลายทางย่อมไม่สามารถนำมาประกอบกลับเป็นไอพีดาต้าแกรมที่สมบูรณ์ได้ ฉะนั้นแฟรกเมนต์อื่นที่ได้รับแล้วก็ไม่มีประโยชน์แต่อย่างใดอีกต่อไป จะเห็นว่าการสูญเสียแฟรกเมนต์เพียงส่วนเดียวส่งผลให้ต้องทิ้งไอพีดาต้าแกรมทั้งชุด ปัญหานี้จะส่งผลต่อสมรรถนะการส่งผ่านไอพีดาต้าแกรมได้มาก โดยเฉพาะอย่างยิ่งในสภาพที่มีการทำแฟรกเมนต์หลายครั้ง จากการที่โฮสต์ปลายทางไม่ทราบจำนวนแฟรกเมนต์ที่แน่นอนของไอพีดาต้าแกรม เพื่อป้องกันไม่ให้โฮสต์ปลายทางต้องรอการมาของแฟรกเมนต์อย่างไม่สิ้นสุดหากแฟรกเมนต์บางส่วนสูญหายระหว่างทาง โฮสต์ปลายทางจะอาศัยนาฬิกาจับเวลาเพื่อกำหนดเวลานานที่สุดในการรอแฟรกเมนต์ของไอพีดาต้าแกรมแต่ละชุด โดยจะเริ่มจับเวลาทันทีที่ได้รับแฟรกเมนต์แรก ถ้าเวลาหมดลงก่อนได้รับครบทุกแฟรกเมนต์โฮสต์ปลายทางจะทิ้งแฟรกเมนต์ที่มีอยู่ไปทั้งหมด ข้อสังเกตที่น่าสนใจอีกประการหนึ่ง คือ การรวมแฟรกเมนต์กลับคืนจะกระทำที่โฮสต์ปลายทางทีเดียวเลย ไม่ทำการรวมแฟรกเมนต์ทุกครั้งที่ผ่านเราเตอร์ระหว่างทางแต่ละตัว ขอให้ผู้อ่านลองวิเคราะห์ถึงข้อได้เปรียบและข้อเสียเปรียบระหว่างวิธีการทั้งสอง ในเชิงประสิทธิภาพการรับส่งไอพีดาต้าแกรมและความเป็นไปได้ในการนำมาใช้งาน

2.4.4 ฟิลด์อื่น ๆ

ฟิลด์ time to live (TTL) ขนาด 8 บิต มีหน้าที่กำหนดจำนวนเราเตอร์สูงสุดที่ไอพีดาต้าแกรมสามารถเดินทางผ่านได้ หรือกล่าวในอีกนัยหนึ่งว่าเป็นการกำหนดอายุของไอพีดาต้าแกรมที่อนุญาตให้อยู่ในเครือข่าย ขั้นตอนการใช้งานเป็นดังนี้คือ เมื่อโฮสต์ต้นทางส่งไอพีดาต้าแกรมออกไปจะตั้งค่าเริ่มต้นให้ฟิลด์ TTL ค่าหนึ่ง (โดยทั่วไปใช้ 32 หรือ 64) ทุกครั้งที่ไอพีดาต้าแกรมผ่านเราเตอร์หนึ่ง ค่า TTL จะถูกปรับลดค่าลงหนึ่งหน่วย หากเมื่อใดเราเตอร์พบไอพีดาต้าแกรมที่ค่า TTL ถูกลดลงจนเป็น 0 เราเตอร์จะทิ้งไอพีดาต้าแกรมดังกล่าวพร้อมกับรายงานเหตุการณ์กลับไปยังโฮสต์ต้นทางโดยใช้โปรโตคอลไอซีเอ็มพี การทำเช่นนี้สามารถป้องกันไอพีดาต้าแกรมวนเวียนอยู่ในเครือข่ายอย่างไม่สิ้นสุด

ฟิลด์ protocol เป็นค่าที่ระบุว่าข้อมูลในไอพีดาต้าแกรมได้รับการกำเนิดจากโปรโตคอลชั้นที่สูงกว่าประเภทหรือชนิดใด ซึ่งค่าเหล่านี้ได้รับการกำหนดไว้เป็นมาตรฐานร่วมกันสำหรับโปรโตคอลเหล่านี้เกือบครบทุกประเภทแล้ว ดังที่แสดงในตารางที่ 2.2 ตัวอย่างโปรโตคอลที่น่าสนใจมีการใช้งานและมีการกล่าวถึงบ่อยครั้ง ได้แก่ ไอซีเอ็มพี (Internet Control Message Protocol) ที ซี พี (Transmission Control Protocol) อี จี พี (Exterior Gateway Protocol) และ ยู ดี พี (User Datagram Protocol)

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ตารางที่ 2.2 มาตรฐานการกำหนดค่าในฟิลด์ protocol ของไอพีดาต้าแกรม

Decimal	Keyword	Protocol	Decimal	Keyword	Protocol
0	-	Reversed	49	BNA	BNA
1	ICMP	Internal Control Message Protocol	50	SIPP-ESP	SIPP Encap Security Payload
2	IGMP	Internal Group	51	SIPP-AH	SIPP Authentication Header
3	GGP	Gateway-to-Gateway	52	I-NLSP	Integrated Net Layer Security TUBA
4	IP	IP in IP (encasulation)	53	SWIPE	IP with Encryption
5	ST	Stream	54	NHRP	NBMA Next Hop Resolution Protocol
6	TCP	Transmission Control Protocol	55-60	-	Unassigned
7	UCL	UCL	61	-	any host internal protocol
8	EGP	Exterior Gateway Protocol	62	CFTP	CFTP
9	IGP	any private interior gateway	63	-	any local network
10	BBN-RCC-MON	BBN RCC Monitoring	64	SAT-EXPAK	SATNET and Backroom EXPAK
11	NVP-II	Network Voice Protocol	65	KRYPTOLAN	Kryptolan
12	PUP	PUP	66	RVD	MIT Remote Virtual Disk Protocol
13	ARGUS	ARGUS	67	IPPC	Internet Pluribus Packet Core
14	EMCON	EMCON	68	-	Any distributed file system
15	XNET	Cross Net Debugger	69	SAT-MON	SATNET Monitoring
16	CHAOS	Chaos	70	VISA	VISA Protocal
17	UDP	User Datagram Protocol	71	IPCV	Internet Packet Core Utility
18	MUX	Multiplexing	72	CPNX	Computer Protocol Network Executive
19	DCN-MEAS	DCN Measurement Subsystem	73	CPHB	Computer Protocol Heat Beat
20	HMP	Host Monitoring	74	WSN	Wang Span Network
21	PRM	Packet Radio Measurement	75	PVP	Packet Video Protocol
22	XNS-IDP	XEROX NS IDP	76	BR-SAT-MON	Backroom SATNET Monitoring
23	TRUNK-1	Trunk-1	77	SUN-ND	SUN ND PROTOCOL-Temporary
24	TRUNK-2	Trunk-2	78	WB-MON	WIDEBAND Monitoring
25	LEAF-1	Leaf-1	79	WB-EXPAK	WIDEBAND EXPAK
26	LEAF-2	Leaf-2	80	ISO-IP	ISO Internet Protocol
27	RDP	Reliable Data Protocol	81	VMTP	VMTP
28	IRTP	Internet Reliable Transaction	82	SECURE-VMTP	SECURE-VMTP
29	ISO-TP4	ISO Transport Protocol Class 4	83	VINES	VINES
30	NETBLT	Bulk Data Transfer Protocol	84	TTP	TTP
31	MFE-NSP	MFE Network Service Protocol	85	NSFNET-IGP	NSFNET-IGP
32	MERIT-INP	MERIT Intermodal Protocol	86	DGP	Dissimilar Gateway Protocol
33	SEP	Sequential Exchange Protocol	87	TCF	TCF
34	3PC	Third Party Connect Protocol	88	IGRP	IGRP
35	IDPR	Inter-Domain Policy Routing Protocol	89	OSPF IGP	OSPF IGP
36	XTP	XTP	90	Sprite-RPC	Sprite RPC Protocol
37	DDP	Datagram Delivery Protocol	91	LARP	Locus Address Resolution Protocol
38	IDPR-CMTP	IDPR Control Message Transport Protocol	92	MTP	Multicast Transport Protocol
39	TP++	TP++ Transport Protocol	93	AX.25	AX.25 Frames
40	IL	IL Transport Protocol	94	IPIP	IP-within-IP Encapsulation Protocol
41	SIP	Simple Internet Protocol	95	MICP	Mobile Internetworking Control Pro.
42	SDRP	Source Demand Routing Protocol	96	SCC-SP	Semaphore Communication Sec. Pro
43	SIP-SR	SIP Source Route	97	ETHERIP	Ethernet-within-IP Encapsulation
44	SIP-FRAG	SIP Fragment	98	ENCAP	Encapsulation Header
45	IDRP	Inter-Domain Routing Protocol	99	-	Any private encryption scheme
46	RSVP	Reservation Protocol	100	GMTP	GMTP
47	GRE	General Routing Encapsulation	101-254	-	Unassigned
48	MHRP	Mobile Host Routing Protocol	255	-	Reserved

60909

ฟิลด์ header checksum ขนาด 16 บิต มีไว้เพื่อตรวจสอบความถูกต้องของเฮดเดอร์ ลักษณะการใช้งานเป็นดังนี้คือ เมื่อโฮสต์ต้นทางให้กำเนิดไอพีดาต้าแกรมหนึ่งขึ้นจะคำนวณค่า header checksum โดยนำเฮดเดอร์ทีละ 16 บิตมาบวกกันแบบ 1 คอมพลีเมนต์ จากนั้นนำผลที่ได้มาทำ 1 คอมพลีเมนต์อีกครั้ง จึงจะได้เป็นค่าที่บรรจุลงใน header checksum ที่ภาครับปลายทางจะตรวจสอบความผิดพลาดของเฮดเดอร์ โดยนำเฮดเดอร์ทีละ 16 บิตมาบวกกับค่าในฟิลด์ header checksum แบบ 1 คอมพลีเมนต์หากผลลัพธ์ที่ได้มีค่าเป็นหนึ่งทั้ง 16 บิต แสดงว่าไม่มีความผิดพลาดเกิดขึ้นกับเฮดเดอร์ หากไม่ใช่ก็แสดงว่ามีความผิดพลาดบางอย่างเกิดขึ้นกับเฮดเดอร์ ในกรณีนี้ไอพีดาต้าแกรมดังกล่าวจะถูกทิ้งไปโดยที่ไม่มีการรายงานความผิดพลาดที่เกิดขึ้นแต่อย่างใด โปรโตคอลในชั้นที่สูงกว่าจะต้องตรวจรู้ปัญหาเหล่านี้ด้วยตนเอง

สังเกตว่าการคำนวณค่า header checksum จะกระทำเฉพาะกับส่วนเฮดเดอร์เท่านั้นแสดงว่าข้อมูลในไอพีดาต้าแกรมมิได้รับการตรวจสอบความผิดพลาดแต่อย่างใด การทำเช่นนี้มีข้อดีตรงที่อุปกรณ์สื่อสารระหว่างทาง เช่น เราเตอร์ สามารถตรวจสอบความถูกต้องของ header checksum ได้อย่างรวดเร็ว เพราะเฮดเดอร์มีจำนวนบิตน้อยกว่าส่วนของข้อมูลมาก เมื่อปริมาณงานที่ต้องประมวลผลลดลงเราเตอร์ก็มีการทำงานที่ซับซ้อนน้อยลง การตรวจสอบความผิดพลาดของส่วนข้อมูลจะเป็นภาระหน้าที่ของโฮสต์ต้นทางและปลายทางที่ต้องจัดการเอง โดยอาศัยโปรโตคอลชั้นที่สูงกว่า เช่น โปรโตคอลทีซีพีหรือยูดีพีเป็นต้น

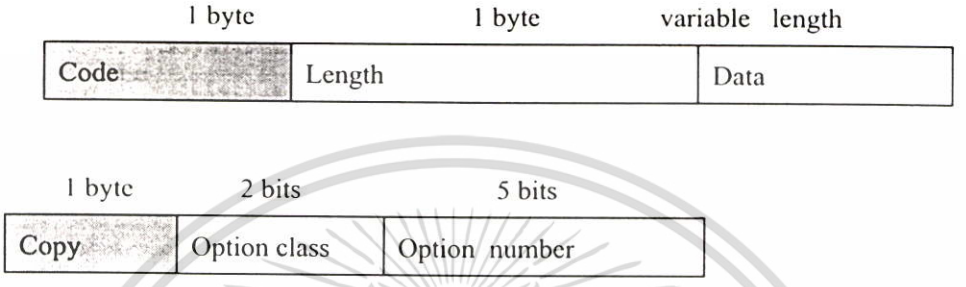
ฟิลด์ IP address ฟิลด์ถัดจากฟิลด์ header checksum คือ หมายเลขไอพีแอดเดรสต้นทาง (source IP address) และหมายเลขไอพีแอดเดรสปลายทาง (destination IP address) ฟิลด์ทั้งสองมีขนาดเท่ากันคือ 32 บิตใช้เป็นหมายเลขประจำตัวของโฮสต์ เมื่อโฮสต์ต้นทางกำเนิดไอพีดาต้าแกรมขึ้นจะระบุหมายเลขไอพีแอดเดรสต้นทางเป็นหมายเลขไอพีของตนเอง และกำหนดหมายเลขไอพีแอดเดรสปลายทางเป็นหมายเลขของโฮสต์ปลายทางที่ต้องการติดต่อด้วย ค่าในฟิลด์ทั้งสองนี้จะไม่มีการเปลี่ยนแปลงใด ๆ เลยในระหว่างที่ส่งผ่านเครือข่ายของระบบ

ฟิลด์ option และ padding ฟิลด์ option เป็นส่วนที่เพิ่มเติมเข้ามากับไอพีดาต้าแกรม มีการใช้เฉพาะกับงานบางอย่าง เช่น การทดสอบเครือข่าย และตรวจหาจุดผิดพลาดของระบบ ฟิลด์นี้มีขนาดที่ไม่ตายตัวขึ้นอยู่กับชนิดของ option ที่เลือกใช้ ในกรณีที่ฟิลด์ที่เลือกใช้มีขนาดที่ไม่ลงตัวเป็นจำนวนเท่าของ 32 บิต จะมีการเติมบิตที่มีค่าเป็นศูนย์ต่อท้ายจนครบ 32 บิต เราเรียกบิตเพิ่มเติมเหล่านี้ว่า padding

ฟิลด์ data คือส่วนที่ใช้บรรจุข้อมูลของไอพีดาต้าแกรม ฟิลด์นี้มีขนาดที่ไม่ตายตัว ความยาวของฟิลด์ data เมื่อรวมกับส่วนของเฮดเดอร์จะต้องมีค่าไม่เกิน 65,535 $\left(2^{16}-1\right)$

2.4.5 ฟิลด์ option

ในสภาพการใช้งานโดยทั่วไปไอพีดาต้าแกรมจะไม่มีการใช้งานฟิลด์ option แต่อย่างใด แต่หากมีการใช้งานฟิลด์นี้แล้ว เช่น ต้องการทดสอบเครือข่าย หรือตรวจหาจุดผิดพลาดของระบบ จะมีพื้นที่สำหรับใช้งานฟิลด์ option ขนาดความยาวสูงสุดไม่เกิน 40 ไบต์ เมื่อรวมกับเฮดเดอร์ส่วนที่มีความยาวคงที่ 20 ไบต์ จะได้เป็นเฮดเดอร์ของไอพีดาต้าแกรมที่มีขนาดสูงสุดไม่เกิน 60 ไบต์



รูปที่ 2.4 โครงสร้างพื้นฐานของ ฟิลด์ option

ก่อนอื่นจะขออธิบายถึงโครงสร้างพื้นฐานของฟิลด์ option จากรูปที่ 2.4 จะเห็นว่าฟิลด์ option ประกอบด้วย 3 ส่วนคือ

ฟิลด์ code ขนาด 1 ไบต์ ซึ่งแบ่งแยกออกได้เป็น 3 ส่วนคือ ฟิลด์ copy ขนาด 1 บิต ฟิลด์ option class ขนาด 2 บิต และฟิลด์ option number ขนาด 5 บิต

ฟิลด์ length ขนาด 1 ไบต์ ใช้บ่งบอกถึงขนาดความยาวของฟิลด์ option หมายเหตุ บางออปชั่นจะไม่มีการใช้งานฟิลด์ length กล่าวคือจะใช้เฉพาะฟิลด์ code เพียงไบต์เดียว

ฟิลด์ data มีขนาดที่แปรเปลี่ยนได้ ใช้บรรจุรายละเอียดของออปชั่นที่เลือกใช้ ในบางออปชั่นจะไม่มีการใช้งานฟิลด์ data แต่อย่างใด เช่นเดียวกับกรณีของฟิลด์ length

จะเห็นว่าหากมีการใช้งานฟิลด์ option ในไอพีดาต้าแกรมแล้ว อย่างน้อยจะต้องมีองค์ประกอบของฟิลด์ code ขนาดหนึ่งไบต์เสมอ สำหรับ option บางประเภทจะมีส่วนเพิ่มเติม ได้แก่ ฟิลด์ option length ขนาดหนึ่งไบต์ และฟิลด์ data ต่อท้าย ในเฮดเดอร์ของไอพีดาต้าแกรมยังอนุญาตให้บรรจุหลายออปชั่นพร้อมกันได้ โดยจะไม่มีฟิลด์พิเศษใด ๆ ขั้นระหว่างออปชั่นแต่ละชนิดที่เลือกใช้

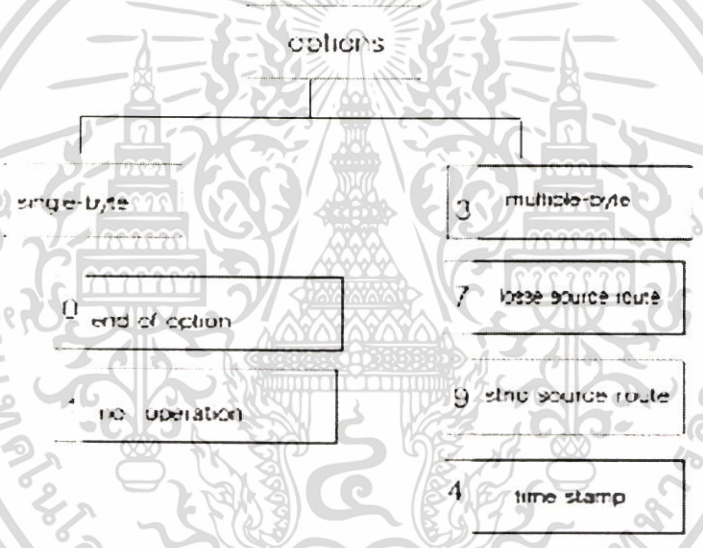
ทีนี้มาพิจารณาหน้าที่ของฟิลด์ code ก่อน จากที่แสดงในรูปที่ 2.4 ฟิลด์ code แบ่งย่อยออกเป็น 3 ส่วน โดยแต่ละส่วนมีหน้าที่ดังนี้

ฟิลด์ copy ขนาด 1 บิต ใช้ในการกำกับกระบวนการแฟรกเมนต์ของอุปกรณ์เราเตอร์ นั่นคือถ้าบิตนี้มีค่าเป็น 0 ให้มีความหมายว่า หากไอพีดาต้าแกรมต้องมีการแฟรกแมนต์แล้ว ให้บรรจุฟิลด์

option เฉพาะกับเฮดเดอร์ของแฟรกเมนต์แรกเท่านั้น แฟรกเมนต์ที่เหลือจะไม่มีข้อมูลของฟิลด์ option แต่ถ้ามีค่าเป็น 1 ให้ทำสำเนาฟิลด์ option ลงในเฮดเดอร์ของแฟรกเมนต์ทุกตัว

ฟิลด์ option class ขนาด 2 บิต ใช้ในการแบ่งแยกกลุ่มของออปชั่นที่ใช้งานอย่างกว้าง ๆ กล่าวคือ ถ้ามีค่าเท่ากับ 00 หมายความว่าออปชั่นที่ใช้เป็นชนิด datagram control หรือถ้ามีค่าเท่ากับ 10 หมายความว่าออปชั่นนี้ใช้สำหรับ debugging และ management ส่วนค่าอื่น ๆ อีก 2 ค่า คือ 01 และ 11 สงวนไว้สำหรับอนาคต

ฟิลด์ option number ขนาด 5 บิต ใช้ในการระบุชนิดของออปชั่น ซึ่งมีได้มากถึง 32 ชนิด แต่ในปัจจุบันได้มีการใช้งานอยู่เพียง 6 ชนิด โดย 2 ชนิดแรกเป็นออปชั่นที่ใช้เพียงไบต์เดียว นั่นคือไม่มีฟิลด์ length และ ฟิลด์ data ส่วนอีก 4 ชนิดที่เหลือมีส่วนของฟิลด์ length และ ฟิลด์ data ต่อท้าย ดูการแบ่งประเภทของออปชั่นตามฟิลด์ option number ได้ในรูปที่ 2.5

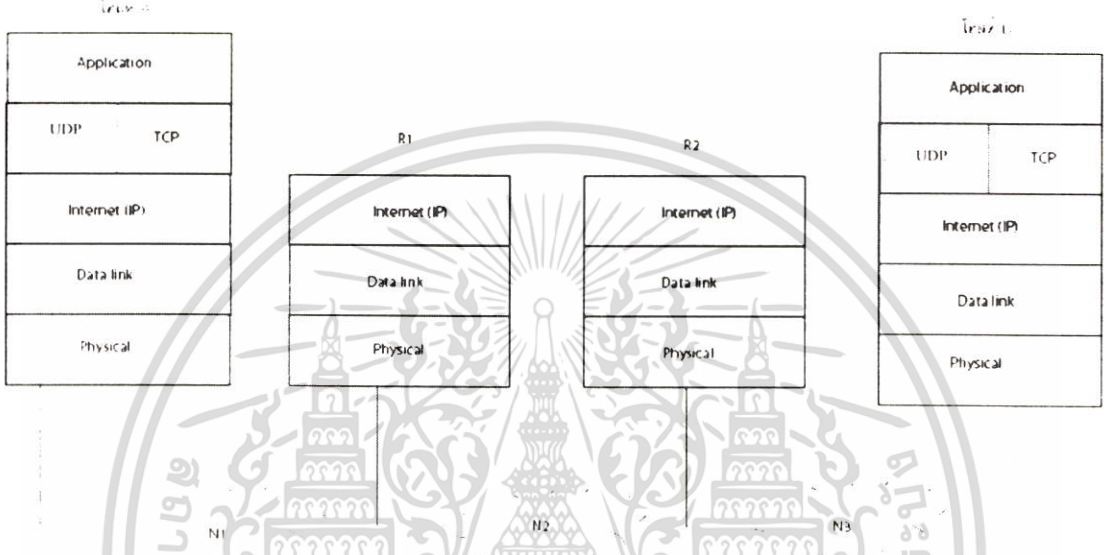


รูปที่ 2.5 ประเภทของออปชั่นที่แบ่งแยกตามค่าของ option number ทั้งนี้ออปชั่นเกือบทั้งหมดจัดอยู่ใน option class = 00 ยกเว้นออปชั่น time stamp เท่านั้นที่จัดเป็น option class = 10

2.5 โปรโตคอลยูดีพี (UDP protocol)

ในหัวข้อก่อน ๆ เราได้กล่าวถึงรายละเอียดของโปรโตคอลไอพี ซึ่งเทียบได้กับการทำงานในชั้นเครือข่าย (network layer) ตามสถาปัตยกรรมเครือข่ายโอเอสไอ หน้าที่ของโปรโตคอลไอพีคือการส่งผ่านข้อมูลข่าวสารในรูปของไอพีดาต้าแกรมจากโฮสต์ต้นทางผ่านเครือข่ายของระบบ โดยอาศัยอุปกรณ์เราเตอร์เป็นกลไกสำคัญในการส่งต่อจนถึงจุดหมายปลายทาง การให้บริการสื่อสารในชั้นโปรโตคอลไอพีเป็นแบบคอนเนกชั่นเลส (connectionless) คือ ไม่มีกลไกหรือมาตรการใดๆ ในการรับประกันว่าไอพีดาต้าแกรมจะได้รับการส่งผ่านอย่างถูกต้อง หากมีความผิดพลาดบางอย่าง

เกิดขึ้นระหว่างการรับส่งข้อมูล สิ่งที่เราเตอร์หรือโฮสต์ปลายทางจะกระทำคือ ทิ้งไอพีดาต้าแกรมนั้นไปพร้อมกับรายงานความผิดพลาดที่เกิดขึ้นโดยอาศัยโปรโตคอลไอซีเอ็มพีกลับไปให้กับโฮสต์ต้นทางเท่านั้น ด้วยเหตุนี้การแก้ไขปัญหาความผิดพลาดที่อาจเกิดขึ้นระหว่างการรับส่งไอพีดาต้าแกรมผ่านเครือข่าย จึงตกเป็นหน้าที่หรือภาระของโปรโตคอลชั้นที่สูงกว่าในส่วนของโฮสต์ต้นทางและปลายทางเอง



รูปที่ 2.6 ความสัมพันธ์ของโปรโตคอลยูดีพีและทีซีพีกับชั้นโปรโตคอลอื่น

2.5.1 โปรโตคอลยูดีพีและทีซีพี

ในหัวข้อนี้จะกล่าวถึงโปรโตคอลในอีกชั้นหนึ่งเมื่อเทียบตามสถาปัตยกรรมเครือข่ายโอเอสไอ คือ ชั้นทรานสปอร์ต (transport layer) โปรโตคอลในชั้นนี้เป็นชั้นที่อยู่ถัดขึ้นไปจากชั้นเครือข่าย มีหน้าที่เป็นตัวกลางเชื่อมต่อระหว่างโปรแกรมแอพพลิเคชั่นกับระบบเครือข่าย ชั้นทรานสปอร์ตจัดว่าเป็นชั้นโปรโตคอลชั้นแรกที่มีการทำงานเป็นแบบ โฮสต์ ถึง โฮสต์ (host-to-host) นั่นคืออุปกรณ์สื่อสารที่เชื่อมต่อโฮสต์ทั้งสองในระบบ เช่น เราเตอร์ จะไม่ทราบหรือสนใจข่าวสารหรือการทำงานที่เกิดขึ้นในชั้นโปรโตคอลนี้เลย (ดูรูป 2.6 ประกอบ) มีเพียงโฮสต์ต้นทางและโฮสต์ปลายทางเท่านั้นที่เกี่ยวข้องกับกระบวนการทำงานในชั้นโปรโตคอลนี้ ด้วยเหตุนี้เราจึงมักจะเรียกการสื่อสารแบบนี้ว่าเป็นการสื่อสารแบบ จุดปลาย ถึง จุดปลาย (end-to-end)

สำหรับมาตรฐานทีซีพี/ไอพี ชั้นทรานสปอร์ตประกอบด้วยชุดโปรโตคอลที่สำคัญอยู่ 2 ชุด คือ (ดูรูปที่ 2.6 ประกอบ)

1. ทีซีพี (Transmission control protocol)

2. ยูดีพี (User datagram protocol)

โปรโตคอลทีซีพีมีกระบวนการทำงานที่ซับซ้อนกว่าโปรโตคอลยูดีพีมาก เพราะมีกลไกหลายส่วนสำหรับแก้ปัญหาที่เกิดขึ้นจากความผิดพลาดในการรับส่งไอพีดาต้าแกรมของชั้นโปรโตคอลไอพี ทั้งนี้มีวัตถุประสงค์เพื่อให้การรับส่งข้อมูลข่าวสารระหว่างโฮสต์ต้นทางและปลายทางมีความถูกต้องแน่นอนและเชื่อถือได้ ฉะนั้นโปรแกรมแอพพลิเคชั่นที่ใช้งานโปรโตคอลทีซีพี จึงไม่ต้องกังวลและมั่นใจได้ว่าการรับส่งข่าวสารระหว่างโฮสต์ผ่านเครือข่ายจะมีความถูกต้องดีตลอดการใช้งาน ในทางตรงกันข้ามโปรโตคอลยูดีพีได้รับการออกแบบให้มีโครงสร้างที่เรียบง่ายไม่ซับซ้อนและมีบทบาทน้อยมาก หน้าที่ของหลักโปรโตคอลยูดีพีคือ การมัลติเพล็กซ์ดาต้าแกรมของโปรแกรมแอพพลิเคชั่นต่าง ๆ ที่ทำงานอยู่บนโฮสต์เดียวกัน ให้สามารถเชื่อมต่อและใช้งานระบบเครือข่ายพร้อมกันได้ และในทางกลับกันเมื่อโฮสต์ได้รับ ไอพีดาต้าแกรมจากเครือข่าย ก็จะทำการดีมัลติเพล็กซ์ดาต้าแกรมเหล่านั้นเพื่อส่งต่อไปให้โปรแกรมแอพพลิเคชั่นที่ถูกต้อง กรรมวิธีในการมัลติเพล็กซ์และดีมัลติเพล็กซ์นั้นอาศัยหลักการของพอร์ต (port) ซึ่งจะได้กล่าวถึงในส่วนถัดไป

เนื่องจากโปรโตคอลยูดีพีไม่มีมาตรการสำหรับรับประกันความถูกต้องของข้อมูลที่ส่งผ่านเครือข่าย ดังนั้นโปรแกรมแอพพลิเคชั่นที่ใช้ในงานโปรโตคอลยูดีพี จะต้องจัดการกับปัญหาต่าง ๆ ที่อาจเกิดขึ้น เช่น ปัญหาการสูญหายของข้อมูล ปัญหาการได้รับข้อมูลซ้ำซ้อน หรือปัญหาการสลับลำดับของข้อมูลเองทั้งหมด แต่ในทางปฏิบัติผู้พัฒนาโปรแกรมแอพพลิเคชั่นมักจะไม่สนใจและไม่นำปัญหาเหล่านี้มาพิจารณาร่วมในการเขียนซอฟต์แวร์ ดังนั้นโปรโตคอลยูดีพีจึงใช้งานได้ดีหรือเหมาะสมเฉพาะกับระบบที่มีอัตราความผิดพลาดต่ำเท่านั้น เช่น การสื่อสารภายในเครือข่ายท้องถิ่นแลน หากนำโปรโตคอลยูดีพีมารองรับโปรแกรมแอพพลิเคชั่นที่การเชื่อมต่อของโฮสต์อยู่ห่างไกลกัน และข้อมูลต้องส่งผ่านอุปกรณ์สื่อสาร เช่น เราเตอร์หลายตัวมาก คุณภาพการสื่อสารก็สามารถที่จะแย่ลงอย่างมาก ซึ่งจะส่งผลกระทบต่อการทำงานของโปรแกรมแอพพลิเคชั่น ถึงแม้โปรโตคอลยูดีพีจะดูไม่มีคุณสมบัติเพิ่มเติมหรือแตกต่างไปจากการใช้บริการโปรโตคอลไอพีเท่าใดนัก แต่โปรโตคอลยูดีพียังมีการใช้งานและประโยชน์กับการใช้งานบางประเภทอยู่

2.5.2 พอร์ตและซอกเกต

โดยทั่วไปเครื่องคอมพิวเตอร์หรือโฮสต์เครื่องหนึ่งสามารถรองรับการใช้งานโปรแกรมแอพพลิเคชั่นของผู้ใช้เครื่องได้หลายโปรแกรมพร้อมกัน เช่น ผู้ใช้อาจกำลังท่องอินเทอร์เน็ตด้วยโปรแกรมอินเตอร์เน็ตเอ็กซ์โปล์เรอร์ (Internet Explorer) และในขณะเดียวกันก็พูดคุยกับเพื่อนโดยใช้โปรแกรมไอซีคิว (ICQ) จะเห็นว่าโปรแกรมแอพพลิเคชั่นทั้งสองโปรแกรมต้องมีการเชื่อมต่อกับระบบอินเทอร์เน็ตในช่วงเวลาเดียวกัน จึงเกิดคำถามขึ้นว่าโฮสต์ที่กำลังรองรับการทำงานของโปรแกรมแอพพลิเคชั่นมากกว่าหนึ่งโปรแกรมพร้อมกัน จะสามารถแยกแยะได้อย่างไรว่าไอพีดา-

ด้าแกรมที่ได้รับจากระบบเครือข่ายอินเทอร์เน็ตบรรจุข้อมูลข่าวสารของโปรแกรมแอพพลิเคชั่นใดอยู่ จากที่ได้อธิบายไว้ในบทก่อนหน้าแล้วว่า การส่งไอพีดาต้าแกรมของระบบซึ่งกระทำในชั้นโปรโตคอลไอพี จะพิจารณาจากหมายเลขไอพีแอดเดรสของโฮสต์ปลายทางเท่านั้น ฉะนั้นไอพีดาต้าแกรมที่โฮสต์ได้รับแต่ละตัวจึงไม่มีส่วนประกอบใดที่บ่งถึงชนิดของโปรแกรมแอพพลิเคชั่น ดังนั้นจำเป็นที่ระบบจะต้องมีกลไกเพิ่มเติมสำหรับการแยกแยะไอพีดาต้าแกรมของโปรแกรมแอพพลิเคชั่นแต่ละตัว และปัญหานี้ได้ถูกจัดให้เป็นหน้าที่ของโปรโตคอลในชั้นทรานสปอร์ตคือ เป็นหน้าที่ของโปรโตคอลทีซีพีและยูดีพีนั่นเอง แนวคิดที่ใช้ในการจัดการกับปัญหาดังกล่าวของโปรโตคอลทั้งสองเหมือนกันคือ การใช้หมายเลขพอร์ตขนาด 16 บิต (16–bit port) เป็นหมายเลขแทนชนิดของโปรแกรมแอพพลิเคชั่น ซึ่งจะถูกกำหนดเป็นฟิลด์หนึ่งในโครงสร้างของยูดีพีดาต้าแกรมและทีซีพีดาต้าแกรม เมื่อดูจากขนาดของพอร์ตจะเห็นว่าสามารถแบ่งชนิดของโปรแกรมแอพพลิเคชั่นได้มากถึง 65,535 ประเภท

การกำหนดหมายเลขเฉพาะของพอร์ตสำหรับแทนโปรแกรมแอพพลิเคชั่นแต่ละชนิดเป็นเรื่องสำคัญ เพราะโฮสต์หรือคอมพิวเตอร์ทั้งหมดจะต้องรับทราบและเข้าใจตรงกัน ในปัจจุบันได้มีการกำหนดหมายเลขพอร์ตเป็นมาตรฐานที่ใช้ร่วมกันแล้วจำนวนหนึ่ง และเรียกพอร์ตเหล่านี้ว่าพอร์ตมาตรฐาน (well-know port) การกำหนดหมายเลขพอร์ตของโปรโตคอลทีซีพีและยูดีพีเป็นอิสระไม่ขึ้นแก่กัน แต่กระนั้นในอดีตพอร์ตมาตรฐานบางส่วนมีการกำหนดพอร์ตของโปรโตคอลทีซีพีและยูดีพี เป็นหมายเลขตรงกันสำหรับแอพพลิเคชั่นชนิดเดียวกัน ดูตัวอย่างการกำหนดพอร์ตมาตรฐานได้ใน ตารางที่ 2.3 สำหรับผู้ที่เคยใช้ระบบปฏิบัติการยูนิกซ์หรือลีนุกซ์สามารถดูการกำหนดพอร์ตสำหรับการให้บริการแบบต่างๆ ได้จากโฟล์ \etc\services

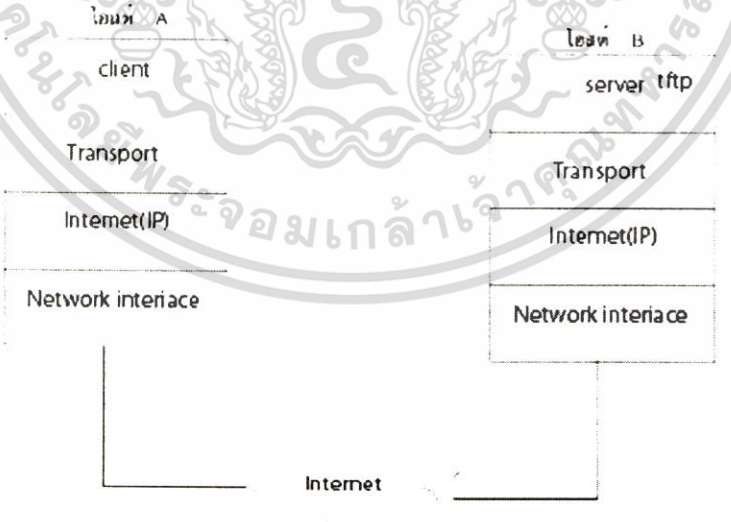
ตารางที่ 2.3 ตัวอย่างการกำหนดพอร์ตสำหรับการให้บริการของพอร์ตมาตรฐาน

ชนิดของการบริการ	หมายเลขพอร์ต / ชนิดโปรโตคอล	คำอธิบาย
Echo	7/tcp	
Echo	7/udp	
Discard	9/tcp	Sink null
Discard	9/udp	Sink null
Systat	11/tcp	users
Daytime	13/tcp	
Daytime	13/udp	
Netstat	15/tcp	
Quote	17/udp	text
Chargen	19/tcp	ttytst
Chargen	19/udp	ttytst
Ftb	21/tcp	
Ssh	22/tcp	# Secure shell login
Ssh	22/udp	# Secure shell login
Telnet	23/tcp	
Smtp	25/tcp	mail
Time	37/tcp	timeserver
Time	37/udp	timeserver
Name	42/tcp	nameserver
Whois	43/tcp	nickname
Domain	53/udp	nameserver # domain name server
Domain	53/tcp	nameserver
Mtb	57/tcp	
Bootps	67/udp	
Bootpc	68/udp	
Tftp	69/udp	
Finger	79/tcp	
www	80/tcp	http # WorldWideWeb Http
www	80/udp	# Hyper Text Tranfer protocol
Hostnames	101/tcp	hostname
Nntp	119/tcp	readnews
Ntp	123/udp	ntp
Snmp	161/udp	
print-srv	170/tcp	
Dnaeml	436/tcp	# DECnet /OSI
Biff	512/udp	comsat
Exec	512/ tcp	

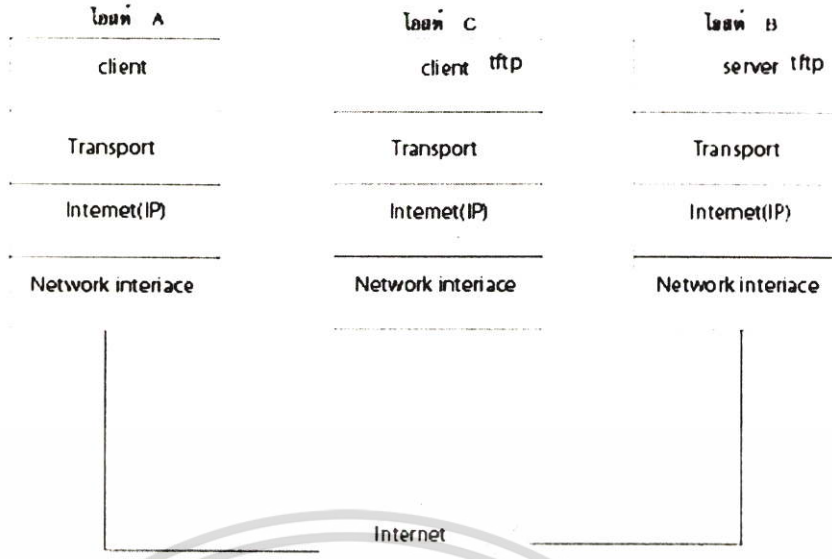
สำหรับผู้ที่เคยพัฒนาโปรแกรมแอพพลิเคชั่นบนโปรโตคอลทีซีพี/ไอพี จะคุ้นเคยกับคำว่า ซอกเกต (socket) เพราะจะเรียกใช้งานเมื่อต้องการสร้างการเชื่อมต่อหรือคอนเนกชั่นระหว่างโฮสต์ ซอกเกตแท้จริงแล้วก็คือ การอ้างถึงหมายเลขไอพีแอดเดรสกับหมายเลขพอร์ตพร้อมกันนั่นเอง แม้คำว่าซอกเกตมีนิยามค่อนข้างง่ายแต่ก็มีประโยชน์ เพราะเมื่อใดที่เราทราบค่าซอกเกตของโฮสต์ต้นทางและโฮสต์ปลายทาง ก็สามารถใช้อ้างถึงและแยกแยะการเชื่อมต่อสื่อสารหรือคอนเนกชั่นระหว่างโปรแกรมแอพพลิเคชั่น 2 โปรแกรมได้อย่างสมบูรณ์ เรามักเรียกซอกเกตของโฮสต์ต้นทางและซอกเกตของโฮสต์ปลายทางรวม ๆ กันว่าเป็น คู่ซอกเกต (socket pair)

2.5.3 การใช้งานพอร์ตและไอพีแอดเดรสคู่กัน

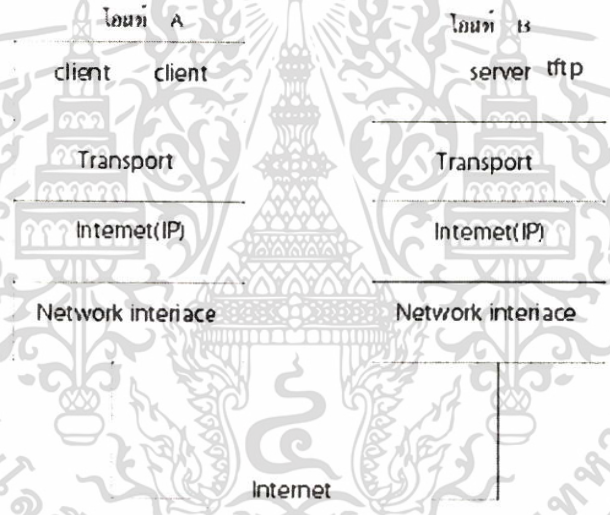
โดยทั่วไปพื้นฐานโครงสร้างการทำงานและการติดต่อสื่อสารของโปรแกรมแอพพลิเคชั่นในเครือข่ายอินเทอร์เน็ตมีลักษณะเป็นแบบที่เรียกว่า ไคลเอนต์ (Client) และเซิร์ฟเวอร์ (server) นั่นคือ เมื่อโปรแกรมแอพพลิเคชั่น 2 โปรแกรมมีการติดต่อสื่อสารระหว่างกัน โปรแกรมหนึ่งจะทำหน้าที่เป็นไคลเอนต์และอีกโปรแกรมหนึ่งจะทำหน้าที่เป็นที่เซิร์ฟเวอร์ โปรแกรมเซิร์ฟเวอร์คือโปรแกรมที่ทำหน้าที่เป็นผู้ให้บริการเฉพาะอย่างกับโปรแกรมไคลเอนต์ ฉะนั้นโดยปกติโปรแกรมเซิร์ฟเวอร์จะถูกรันอย่างต่อเนื่องและพร้อมรับการเรียกใช้บริการจากโปรแกรมไคลเอนต์อยู่ตลอดเวลา ในขณะที่โปรแกรมไคลเอ็นต์จะทำงานเฉพาะช่วงเวลาที่ผู้ใช้ต้องการเท่านั้น เมื่อใดที่โปรแกรมไคลเอ็นต์มีความประสงค์จะใช้บริการจากเซิร์ฟเวอร์ก็จะติดต่อขอเข้าใช้บริการ โดยการส่งดาต้าแกรมไปยังโฮสต์ที่มีการให้บริการของเซิร์ฟเวอร์ดังกล่าว



(ก) โปรแกรมแอพพลิเคชั่นจาก โฮสต์ A ใช้บริการ tftp ของโฮสต์ B



(ข) โปรแกรมแอพพลิเคชั่นจาก โฮสต์ A และ C ใช้บริการ tftp ของโฮสต์ B พร้อมกัน



(ค) โปรแกรมแอพพลิเคชั่นเหมือนกัน 2 โปรแกรม จากโฮสต์ A ใช้บริการ tftp ของโฮสต์ B พร้อมกัน

รูปที่ 2.7 ลักษณะการเชื่อมต่อสื่อสารไคลเอนต์เซิร์ฟเวอร์แบบต่างๆ

ยกตัวอย่างเช่น สมมุติโฮสต์ A (ไคลเอ็นต์) ต้องการติดต่อเพื่อเข้าใช้งานโปรแกรม tftp ของโฮสต์ B (เซิร์ฟเวอร์) ดูรูปที่ 2.7 (ก) ประกอบ คาต้าแกรมแต่ละตัวที่โฮสต์ A ส่งออกจะกำหนดหมายเลขไอพีแอดเดรสปลายทางเป็นไอพีแอดเดรสของโฮสต์ B และตั้งค่าพอร์ตปลายทางเป็นหมายเลขพอร์ตของแอพพลิเคชั่น tftp ซึ่งมีค่าเท่ากับ 69 ส่วนหมายเลขไอพีแอดเดรสต้นทางบรรจุ

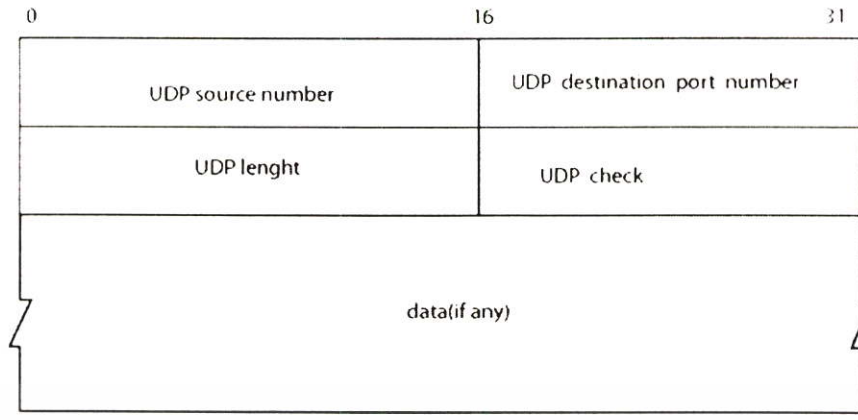
หมายเลขไอพีแอดเดรสของโฮสต์ A เอง และเลือกพอร์ตต้นทางค่าใดค่าหนึ่งที่ยังไม่ถูกใช้งาน เนื่องจากพอร์ตต้นทางเป็นค่าภายในเครื่องของโฮสต์ A เอง โฮสต์ A จึงสามารถควบคุมการจัดสรรหมายเลขพอร์ตไม่ให้มีการใช้งานซ้ำกับโปรแกรมแอพพลิเคชั่นอื่นของตนได้อย่างสมบูรณ์

ในกรณีที่มีโฮสต์อื่นต้องการใช้งานโปรแกรม tftp กับโฮสต์ B เช่นเดียวกับโฮสต์ A เพิ่มเข้ามา ดูรูปที่ 2.7 (ข) ประกอบ โฮสต์เหล่านี้จะกำหนดหมายเลขไอพีแอดเดรสปลายทางเป็นหมายเลขของโฮสต์ B และใช้หมายเลขพอร์ตปลายทางเป็นหมายเลขของแอพพลิเคชั่น tftp เหมือนกับโฮสต์ A ฉะนั้นส่วนที่ใช้ในการแยกแยะว่าเป็นการใช้งานจากโฮสต์เครื่องใดสามารถดูได้จากหมายเลขไอพีแอดเดรสต้นทาง

สำหรับกรณีที่มีโฮสต์ต้นทางเครื่องหนึ่งต้องการใช้งานโปรแกรม tftp กับโฮสต์ B มากกว่าหนึ่งโปรแกรมในเวลาเดียวกัน ดูรูปที่ 2.7 (ค) ประกอบ โปรแกรมเหล่านี้จะกำหนดหมายเลขไอพีแอดเดรสของโฮสต์ต้นทางและโฮสต์ปลายทาง และหมายเลขพอร์ตปลายทางตรงและเหมือนกันหมด มีเพียงส่วนของหมายเลขพอร์ตต้นทางเท่านั้นที่มีค่าแตกต่างกัน ดังนั้นจะเห็นว่าในสถานการณ์แบบนี้ หมายเลขพอร์ตต้นทางมีความสำคัญ เพราะเป็นส่วนที่ทำให้โฮสต์ B สามารถแยกแยะดาต้าแกรมของโปรแกรมแต่ละโปรแกรมออกจากกันได้ ซึ่งตัวอย่างนี้ก็เป็นการอธิบายถึงเหตุผลที่โฮสต์ต้นทางจะกำหนดหมายเลขพอร์ตต้นทางของโปรแกรมแต่ละโปรแกรมให้ต่างกันดังที่ได้กล่าวไว้ในส่วนก่อนหน้า จากที่กล่าวมาทั้งหมดจะเห็นว่าในดาต้าแกรมหนึ่งเมื่อมีการระบุค่าของคู่ซอกเกตครบ จะสามารถแยกแยะและบ่งถึงการเชื่อมต่อหรือคอนเนกชั่นของคู่โปรแกรมแอพพลิเคชั่นหนึ่งได้อย่างสมบูรณ์

2.5.4 โครงสร้างของยูดีพีดาต้าแกรม

โครงสร้างของยูดีพีดาต้าแกรมมีลักษณะดังที่แสดงในรูปที่ 2.8 พิจารณาดูสองฟิลด์แรกได้แก่ หมายเลขพอร์ตต้นทาง (UDP source port number) และหมายเลขพอร์ตปลายทาง (UDP destination port number) ลักษณะการใช้งานหมายเลขพอร์ตในโปรโตคอลยูดีพีเป็นดังนี้คือ เมื่อโปรแกรมแอพพลิเคชั่นหนึ่งมีการสื่อสารโดยใช้โปรโตคอลยูดีพี โปรแกรมดังกล่าวจะต้องกำหนดหมายเลขพอร์ตของโฮสต์ต้นทางและโฮสต์ปลายทาง หมายเลขพอร์ตของโฮสต์ปลายทางเป็นส่วนสำคัญเพราะเป็นค่าที่ระบุถึงชนิดของโปรแกรมแอพพลิเคชั่นที่โฮสต์ต้นทางต้องการติดต่อกับโฮสต์ปลายทาง ส่วนหมายเลขพอร์ตต้นทางระบุเพื่อให้โปรแกรมแอมพลิเคชั่นปลายทางสามารถตอบกลับได้ถูกต้องตรงกัน สังเกตว่าการเลือกหมายเลขพอร์ตต้นทางไม่สัมพันธ์กับชนิดของโปรแกรมแอพพลิเคชั่น



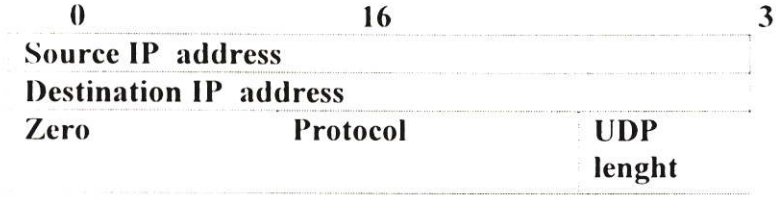
รูปที่ 2.8 โครงสร้างดาต้าแกรมของโปรโตคอลยูดีพี

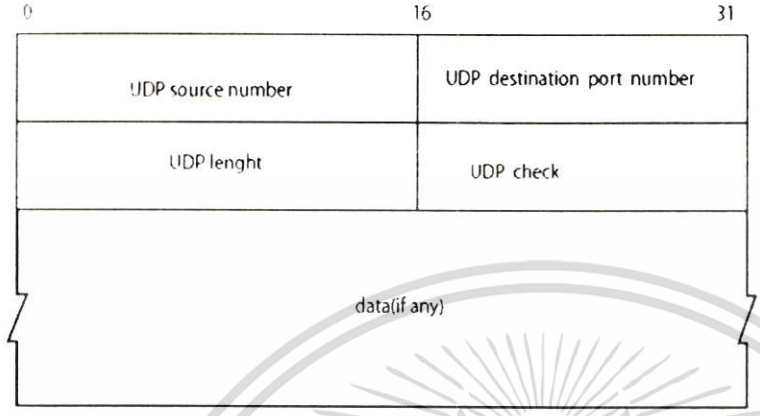
ฟิลด์ UDP length มีขนาด 8 บิต กำหนดเพื่อบ่งบอกจำนวนไบต์หรือออกเตทของยูดีพีดาต้า-แกรม ซึ่งนับรวมทั้งส่วนเฮดเดอร์และส่วนข้อมูลรวมกัน ฉะนั้นฟิลด์นี้ย่อมมีค่าอย่างต่ำเท่ากับ 8 ซึ่งคือขนาดของเฮดเดอร์นั่นเอง สังเกตว่าข่าวสารส่วนนี้มีความซ้ำซ้อนกับฟิลด์ length ที่กำกับอยู่ที่เฮดเดอร์ของไอพีดาต้าแกรม

ฟิลด์ UDP checksum ขนาด 8 บิต มีไว้สำหรับตรวจสอบความผิดพลาดของยูดีพีดาต้าแกรม การคำนวณฟิลด์ UDP checksum มีการนำองค์ประกอบของข้อมูลถึง 3 ส่วน มาร่วมในการคำนวณเพื่อตรวจสอบความถูกต้องได้แก่

1. เฮดเดอร์ของยูดีพี (UDP header)
2. ข้อมูลของยูดีพี (UDP data)
3. เฮดเดอร์เทียมของยูดีพี (UDP pseudo header)

เมื่อเปรียบเทียบกับการคำนวณผลรวมตรวจสอบ (checksum) ของโปรโตคอลไอพี จะพบจุดแตกต่างที่น่าสนใจคือ ในกรณีโปรโตคอลไอพี การคำนวณผลรวมตรวจสอบจะกระทำเฉพาะกับส่วนเฮดเดอร์ของไอพีดาต้าแกรมเท่านั้น ส่วนข้อมูลของไอพีดาต้าแกรมจะถูกส่งผ่านเครือข่ายโดยไม่มีการตรวจสอบความผิดพลาดเลย ในขณะที่ยูดีพีดาต้าแกรมจะตรวจสอบผลรวมตรวจสอบกับทั้งเฮดเดอร์และข้อมูล สังเกตว่าเวลาใช้งานจริงยูดีพีดาต้าแกรมจะถูกบรรจุอยู่ในส่วนข้อมูลของไอพีดาต้าแกรม ฉะนั้นข่าวสารจากโปรแกรมแอพพลิเคชั่นจึงได้รับการตรวจสอบความถูกต้องเมื่อมีการใช้งานผ่านโปรโตคอลยูดีพี อย่างไรก็ตามโปรโตคอลยูดีพีไม่มีการแก้ปัญหาความผิดพลาดที่อาจเกิดขึ้นอีกทั้ง การคำนวณผลรวมตรวจสอบของโปรโตคอลยูดีพีก็มิได้กำหนดเป็นข้อบังคับว่าต้องทำ ผู้พัฒนาซอฟต์แวร์สามารถเลือกจะทำการคำนวณหรือไม่ก็ได้





รูปที่ 2.9 องค์ประกอบที่ใช้ในการคำนวณผลรวมตรวจสอบของโปรโตคอลยูดีพี

ในการคำนวณผลรวมตรวจสอบของโปรโตคอลยูดีพี นอกจากจะพิจารณาจากเฮดเดอร์และข้อมูลของยูดีพีดาต้าแกรมแล้ว ยังนำฟิลด์บางฟิลด์ในเฮดเดอร์ของไอพีดาต้าแกรมมาคำนวณด้วย และเรียกส่วนเพิ่มเติมนี้ว่า เฮดเดอร์เทียม (pseudo header) ดูรูปที่ 2.9 ประกอบ จากรูปจะเห็นว่ามีการนำหมายเลขไอพีแอดเดรสของอุปกรณ์ต้นทางและปลายทาง ฟิลด์ protocol และ ฟิลด์ UDP length มาร่วมในการคำนวณ สังเกตว่าฟิลด์ UDP length ในเฮดเดอร์เทียมเป็นข้อมูลที่ซ้ำซ้อนกับเฮดเดอร์ของยูดีพี การนำหมายเลขไอพีแอดเดรสมาร่วมในการตรวจสอบพร้อมกับหมายเลขพอร์ตมีวัตถุประสงค์เพื่อให้เกิดความแน่ใจว่า คู่ซอกเกต (socket pair) ของการเชื่อมต่อหรือคอนเนกชั่นของโปรแกรมแอพพลิเคชั่นมีความถูกต้องแน่นอน (หมายเหตุ ระบบจะไม่มีการส่งเฮดเดอร์เทียมออกไปกับยูดีพีดาต้าแกรมแต่อย่างใด เฮดเดอร์เทียมใช้เฉพาะกับการคำนวณค่าผลรวมตรวจสอบเท่านั้น)

วิธีการคำนวณค่าผลรวมตรวจสอบของยูดีพีดาต้าแกรมมีขั้นตอนดังนี้คือ นำฟิลด์ต่างๆ ที่เกี่ยวข้องทั้งหมดมาบวกกันครั้งละ 16 บิต แบบ 1 คอมพลีเมนต์ ในกรณีที่จำนวนบิตของส่วนท้ายสุดมีค่าไม่ลงตัวที่ 16 บิต ให้เติมศูนย์ (padding) ต่อท้ายจนครบ 16 บิต เพื่อให้คำนวณได้ ทั้งนี้ระบบจะไม่ส่งบิตศูนย์เหล่านี้ผ่านเครือข่ายไปกับยูดีพีดาต้าแกรมแต่อย่างใด ผลของการบวกที่ได้ให้นำมาทำ 1 คอมพลีเมนต์ ซึ่งคือการสลับค่าบิตจากศูนย์เป็นหนึ่งและจากบิตหนึ่งเป็นศูนย์นั่นเอง ผลลัพธ์ที่ได้เป็นค่าที่บรรจุลงในฟิลด์ checksum สำหรับกระบวนการตรวจสอบความถูกต้องของ

ยูดีพีดาต้าแกรมที่โฮสต์ปลายทางมีขั้นตอนการทำงานดังนี้ นำฟิลด์ต่าง ๆ ที่ใช้ในการคำนวณผลรวมตรวจสอบทั้งหมดมาบวกกับค่าในฟิลด์ checksum ครั้งละ 16 บิต แบบ 1 คอมพลีเมนต์ หากผลการบวกมีค่าเป็นหนึ่งทั้ง 16 บิต ก็แสดงว่าไม่มีความผิดพลาดเกิดขึ้นกับยูดีพีดาต้าแกรมดังกล่าว แต่ถ้าได้ค่าที่ต่างไปก็แสดงว่ามีความผิดพลาดขึ้นกับยูดีพีดาต้าแกรม

เนื่องจากการตรวจสอบความผิดพลาดของฟิลด์ checksum ในโปรโตคอลยูดีพี มิได้กำหนดเป็นข้อบังคับว่าต้องทำเสมอ ดังนั้นหากผู้พัฒนาโปรแกรมเลือกที่จะไม่ตรวจสอบความผิดพลาด ก็จะเซตค่าในฟิลด์ checksum ทั้ง 16 บิต ให้เป็นศูนย์ทั้งหมด ฉะนั้นเมื่อโฮสต์ปลายทางตรวจพบว่าฟิลด์ checksum มีค่าเป็นศูนย์จะทราบทันทีว่าไม่มีการใช้งานฟิลด์ดังกล่าว การตกลงกันระหว่างโฮสต์ต้นทางและปลายทางในลักษณะนี้มักทำให้เกิดคำถามขึ้นว่าหากระบบมีการคำนวณผลรวมตรวจสอบ และบังเอิญผลการคำนวณที่ได้มีค่าเป็นศูนย์ จะก่อให้เกิดความเข้าใจผิดหรือไม่ คำตอบคือไม่เนื่องจากในระบบการบวกแบบ 1 คอมพลีเมนต์ ตัวเลขศูนย์สามารถแสดงได้ 2 รูป แบบคือ แบบที่ทุกบิตมีค่าเป็นศูนย์ และแบบที่ทุกบิตมีค่าเป็นหนึ่ง ในกรณีที่ผลการคำนวณผลรวมตรวจสอบมีค่าเป็นศูนย์ระบบจะบรรจุค่าลงในฟิลด์ checksum ด้วยบิตที่มีค่าเป็นหนึ่งทั้งหมด

2.6 โปรโตคอลทีซีพี

จากที่ได้อธิบายไว้ในหัวข้อก่อน ๆ แล้วว่าการรับส่งข้อมูลผ่านเครือข่ายในรูปของไอพีดาต้าแกรมที่เกิดขึ้นในชั้นของโปรโตคอลไอพีเป็นแบบคอนเนกชั่นเลส (connectionless) นั่นคือ ไม่มีการรับประกันว่าการรับส่งข่าวสารจะถูกต้องตลอดเวลา เพราะไอพีดาต้าแกรมอาจเกิดการสูญหายระหว่างทาง ด้วยเหตุนี้การแก้ไขความผิดพลาดที่อาจเกิดขึ้น จึงเป็นหน้าที่ของโปรโตคอลในชั้นที่สูงกว่า ซึ่งตามมาตรฐานโปรโตคอลทีซีพี/ไอพี ได้กำหนดให้เป็นหน้าที่ของโปรโตคอลทีซีพี ในหัวข้อนี้จะขออธิบายถึงหลักการทำงาน รายละเอียดการใช้งาน และขีดความสามารถของโปรโตคอลทีซีพี สิ่งที่จะพิจารณาในหัวข้อนี้ประกอบด้วย ขั้นตอนการทำงานพื้นฐานของโปรโตคอลทีซีพี โครงสร้างของเซกเมนต์ทีซีพี ประกอบคำอธิบายหน้าที่ของแต่ละฟิลด์ ขั้นตอนการสร้างการเชื่อมต่อและการสิ้นสุดการเชื่อมต่อ

เนื่องจากแนวคิดหรือวัตถุประสงค์ในการพัฒนาโปรโตคอลทีซีพีแตกต่างไปจากโปรโตคอลยูดีพีมาก โดยเฉพาะความต้องการในการพัฒนามาตราการประกันความถูกต้องของการรับส่งข้อมูลระหว่างโฮสต์ต้นทางและปลายทาง ฉะนั้นโครงสร้างของโปรโตคอลทีซีพีจึงมีความซับซ้อนมากกว่าโปรโตคอลยูดีพีมาก ความแตกต่างที่สำคัญระหว่างโปรโตคอลทั้งสองพอจะสรุปเป็นเรื่อง ๆ ได้ดังนี้

โปรโตคอลทีซีพีอาศัยการเชื่อมต่อแบบคอนเนกชั่นออเรียนเต็ด (Conection oriented) คือ โฮสต์ต้นทางและปลายทางต้องมีการสร้างการเชื่อมต่อระหว่างกันก่อนที่จะส่งข้อมูล เพื่อให้แน่ใจว่าโฮสต์ทั้งสองฝ่ายได้เตรียมความพร้อมและรับทราบถึงความต้องการของกันและกันก่อน ซึ่งต่าง

จากโปรโตคอลยูดีพีที่โฮสต์สามารถส่งข้อมูลผ่านเครือข่ายได้ทันทีที่มีข้อมูลพร้อมส่ง โดยไม่คำนึงถึงว่าโฮสต์ปลายทางจะมีความพร้อมรับหรือไม่ ซึ่งเป็นการติดต่อแบบคอนเน็กชันเลส (Conectionless) นั่นเอง ดังนั้นถ้าโปรแกรมแอพพลิเคชั่นที่ใช้งานโปรโตคอลยูดีพีของโฮสต์ปลายทางได้ปิดตัวลงหรือเลิกใช้งานแล้ว โฮสต์ต้นทางก็ยังอาจส่งข้อมูลออกได้ แต่แน่นอนว่าจะไม่เกิดประโยชน์แต่อย่างใด

โปรโตคอลทีซีพีมีมาตรการหรือกลไกการทำงานที่ทำให้สามารถรับประกันความถูกต้องของข้อมูลระหว่างโฮสต์ได้ แม้ว่าข้อมูลเหล่านั้นจะถูกส่งผ่านเครือข่ายไอพีในรูปของไอพีดาต้าแกรม ในขณะที่โปรโตคอลยูดีพีมิได้มีบทบาทหน้าที่เพิ่มเติมหรือแตกต่างไปจากบริการที่ชั้นโปรโตคอลไอพีมีให้เท่าใดนัก ส่วนต่างจากโปรโตคอลไอพีที่สำคัญมีเพียงการเพิ่มหมายเลขพอร์ต เพื่อแยกแยะประเภทของโปรแกรมแอพพลิเคชั่นเท่านั้น

นอกจากนี้การตรวจสอบความถูกต้องของข้อมูลโดยวิธีผลรวมตรวจสอบของโปรโตคอลยูดีพีก็ไม่ได้กำหนดเป็นข้อบังคับ ผู้พัฒนาซอฟต์แวร์สามารถเลือกทำหรือไม่ทำก็ได้ ในขณะที่การตรวจสอบความถูกต้องของข้อมูลในโปรโตคอลทีซีพีเป็นข้อกำหนดที่ต้องทำเสมอ

2.6.1 การทำงานพื้นฐานของ โปรโตคอลทีซีพี

ขั้นแรกข้อมูลของโปรแกรมแอพพลิเคชั่นจะถูกแบ่งออกเป็นองค์ประกอบย่อย ๆ เรียกว่าเซกเมนต์ (segment) เพื่อส่งผ่านเครือข่ายโดยอาศัยไอพีดาต้าแกรม ขนาดของเซกเมนต์จะถูกกำหนดโดยชั้นโปรโตคอลทีซีพีเอง และไม่ขึ้นกับขนาดของข้อมูลที่ส่งมาจากโปรแกรมแอมพลิเคชั่น การทำงานในส่วนนี้แตกต่างจากโปรโตคอลยูดีพี เพราะในกรณีของโปรโตคอลยูดีพีข้อมูลที่กำเนิดจากโปรแกรมแอพพลิเคชั่นแต่ละครั้ง จะถูกบรรจุลงในยูดีพีดาต้าแกรมเลยโดยไม่ต้องนำมาแบ่งย่อยลง และสามารถส่งออกในทันทีที่พร้อม

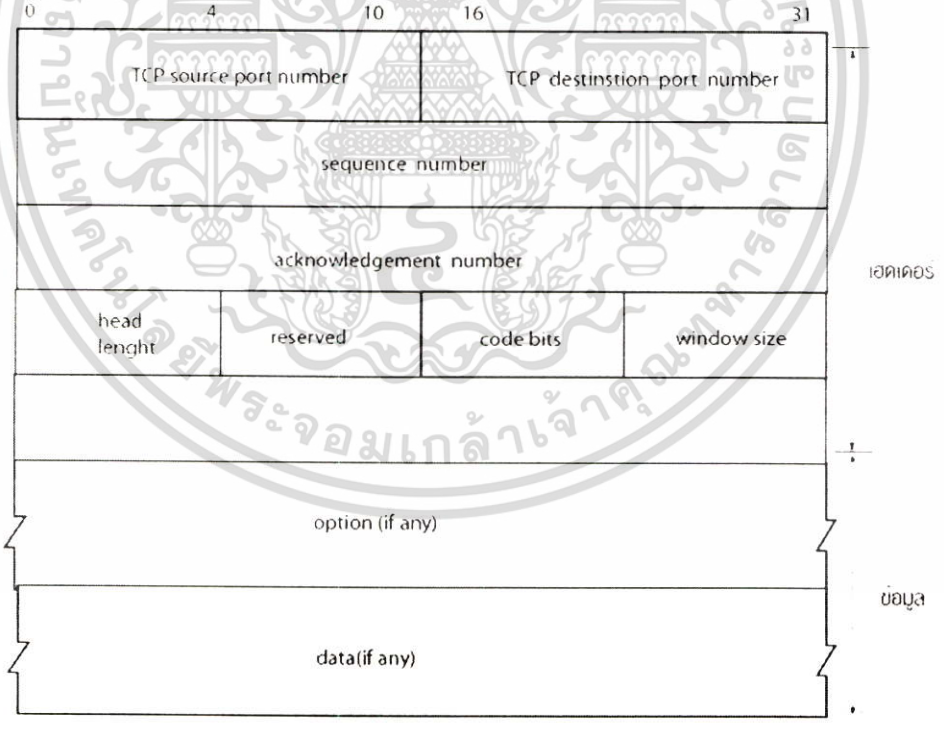
ทุกครั้งที่โฮสต์ต้นทางส่งเซกเมนต์ออกหนึ่งเซกเมนต์ โฮสต์ต้นทางจะตั้งนาฬิกาจับเวลา และรอการตอบรับจากโฮสต์ปลายทางว่าได้รับเซกเมนต์ที่ส่งออกเรียบร้อยหรือไม่ การตอบรับจะอยู่ในรูปของเซกเมนต์ตอบรับที่โฮสต์ปลายทางส่งกลับมา การทำเช่นนี้ช่วยให้โฮสต์ต้นทางมั่นใจว่าเซกเมนต์ที่ตนส่งออกถึงมือผู้รับถูกต้อง สำหรับกรณีที่โฮสต์ต้นทางยังมิได้รับเซกเมนต์ตอบรับจากโฮสต์ปลายทางและช่วงการจับเวลาได้สิ้นสุดลง โฮสต์ต้นทางก็จะอนุมานว่าการส่งเซกเมนต์ดังกล่าวล้มเหลว และจะทำการส่งเซกเมนต์เดิมออกไปซ้ำอีกครั้ง ความล้มเหลวอาจเกิดจากสาเหตุที่ต่างกันหลายแบบ เช่น ไอพีดาต้าแกรมที่บรรจุเซกเมนต์ดังกล่าวเกิดความผิดพลาด ไอพีดาต้าแกรมถูกประวิงเป็นเวลานานมาก จากความแออัดคับคั่งภายในเครือข่าย หรือถูกทิ้งไปโดยเราเตอร์เนื่องจากบัฟเฟอร์เต็มจนล้น ทั้งนี้เหตุการณ์ลักษณะเดียวกันก็อาจเกิดกับเซกเมนต์ตอบรับได้ด้วยเช่นกัน

ลักษณะของข้อมูลที่ส่งผ่านโปรโตคอลทีซีพี จะอยู่ในรูปสตรีมของไบต์ (Byte stream) คือไม่มีการแทรกอักขระหรือชุดบิตพิเศษเพื่อกำกับขอบเขตการเริ่มต้นหรือสิ้นสุดของข้อมูล อีกทั้ง

โปรโตคอลทีซีพีก็ไม่สนใจด้วยว่าข้อมูลที่ส่งผ่านมีรูปแบบเป็นรหัสไบนารี ASCCI EBCDIC หรือรูปแบบอื่น ๆ การตีความหมายของข้อมูลจึงเป็นส่วนที่โปรแกรมแอพพลิเคชั่นที่โฮสต์ต้นทางและปลายทางต้องตกลงกันเอง

2.6.2 โครงสร้างของเซกเมนต์ทีซีพี

เมื่อโปรโตคอลทีซีพีถูกออกแบบให้มีหน้าที่เพิ่มเติมมากมายกว่าโปรโตคอลยูดีพี โครงสร้างเซกเมนต์ของโปรโตคอลทีซีพี ก็ย่อมต้องมีความซับซ้อนและมีจำนวนฟิลด์มากกว่าด้วย พิจารณาดูรูปแบบโครงสร้างของเซกเมนต์ทีซีพี ในรูปที่ 2.10 สองฟิลด์แรกคือ หมายเลขพอร์ตทีซีพีของโฮสต์ต้นทาง (TCP source port number) และหมายเลขพอร์ตทีซีพีของโฮสต์ปลายทาง (TCP destination port number) ฟิลด์ทั้งสองเป็นส่วนที่ทำหน้าที่ระบุหมายเลขพอร์ตหรือชนิดของโปรแกรมแอพพลิเคชั่น ซึ่งมีหน้าที่เหมือนกันกับพอร์ตของโปรโตคอลยูดีพี เมื่อระบุหมายเลขไอพีแอดเดรสคู่กับหมายเลขพอร์ตร่วมกันก็คือ ซอกเกต (socket) นั่นเอง และเมื่อกำหนดซอกเกตของโฮสต์ทั้งสองฝ่าย คือ ไคลเอนต์และเซิร์ฟเวอร์คู่กันก็จะใช้บ่งบอกถึงการเชื่อมต่อหรือคอนแนกชันของโปรแกรมแอพพลิเคชั่นระหว่างโฮสต์ได้



รูปที่ 2.10 โครงสร้างของเซกเมนต์ทีซีพี

ฟิลด์ sequence number กับฟิลด์ acknowlegdement number มีขนาดเท่ากัน คือ 32 บิต กำหนดขึ้นเพื่อใช้งานคู่กัน สำหรับการตรวจสอบความถูกต้องของการส่งข้อมูลในชั้นโปรโตคอลทีซีพี ฟิลด์ sequence number ใช้ระบุหมายเลขไบต์ในสตรีมข้อมูลที่โฮสต์ต้นทางกำลังส่งอยู่ เนื่องจากไบต์ทุกไบต์ในสตรีมของโปรโตคอลทีซีพี จะมีการจัดสรรหมายเลขให้โดยเรียงลำดับจากน้อยไปหามาก หมายเลขที่ใช้มีค่าอยู่ระหว่าง 0 ถึง $2^{32} - 1$ เมื่อใดที่ได้ใช้ถึงตัวเลขที่มีค่าสูงสุดแล้ว ก็ให้วนกลับมาใช้ตัวเลขศูนย์ใหม่และเป็นเช่นนี้เรื่อยไป ส่วนฟิลด์ acknowlegement number จะถูกกำหนดโดยโฮสต์ปลายทาง ค่าที่บรรจุอยู่ในฟิลด์นี้จะถูกกำหนดให้สอดคล้องสัมพันธ์กับหมายเลขของฟิลด์ sequence number ในเซกเมนต์ข้อมูลที่ส่งมาจากโฮสต์ต้นทาง เพื่อแสดงความหมายว่าโฮสต์ปลายทางกำลังรอรับหมายเลขของไบต์ถัดไปในสตรีมข้อมูลหมายเลขใดอยู่ ทั้งนี้หมายเลขของไบต์สตรีมก่อนหน้าทั้งหมดนั้นให้เข้าใจว่าได้รับถูกต้องเป็นที่เรียบร้อยแล้ว

ฟิลด์ head length มีความยาว 4 บิต ใช้บอกถึงขนาดหรือจำนวนไบต์ในเฮดเดอร์ของเซกเมนต์ โดยตัวเลขที่ระบุเป็นตัวเลขที่เป็นจำนวนเท่าของ 4 ไบต์ เฮดเดอร์ปกติจะมีขนาดคงที่เท่ากับ 20 ไบต์ นั่นคือ header length = 5 เฮดเดอร์จะมีขนาดเพิ่มขึ้นเมื่อมีการใช้ฟิลด์ option ซึ่งมีขนาดไม่คงที่ขึ้นกับชนิดของ option ที่เลือกใช้ โดยเฮดเดอร์จะมีขนาดสูงสุดไม่เกิน 60 ไบต์

ฟิลด์ reserved สงวนไว้สำหรับใช้งานในอนาคต

ฟิลด์ code bits มีขนาด 6 บิต เป็นฟิลด์ที่เราจะกล่าวถึงค่อนข้างบ่อยในส่วนต่อไป เพราะเป็นส่วนที่บ่งถึงชนิดหรือประเภทของเซกเมนต์ที่ใช้งานอยู่ เนื่องจากโปรโตคอลทีซีพีกำหนดชนิดของเซกเมนต์ไว้หลายรูปแบบ เช่น เซกเมนต์ที่ใช้รับส่งข้อมูลของโปรแกรมแอพพลิเคชั่น เซกเมนต์ใช้ในการสร้างการเชื่อมต่อเพื่อเปิดการเชื่อมต่อ และเซกเมนต์สำหรับขอสิ้นสุดการเชื่อมต่อ ฟิลด์ code bits มีองค์ประกอบดังนี้

บิตเหล่านี้มีรายละเอียดและหน้าที่ดังต่อไปนี้

URG บิตนี้จะถูกเซตเพื่อแสดงว่าเซกเมนต์มีการใช้งานฟิลด์ urgent pointer อยู่ โดยจะใช้งานร่วมกันเพื่อบอกให้โปรแกรมของโฮสต์ปลายทางหยุดอ่านสตรีมข้อมูลที่อยู่ก่อนหน้าทั้งหมดชั่วคราว และให้อ่านข้อมูลเร่งด่วนที่อยู่ในเซกเมนต์ส่วนนี้ก่อนที่จะดำเนินกิจกรรมเดิมต่อ การใช้งานของบิต URG เกิดขึ้นเฉพาะในกรณี เช่น ผู้ใช้อาจต้องการยกเลิกการติดต่อสื่อสารกลางครันจึงกดปุ่มยกเลิก ข้อมูลการขอยกเลิกการสื่อสารจึงได้รับการส่งออกอย่างเร่งด่วนในเซกเมนต์ที่มีการเซตบิต URG ทั้งนี้ฟิลด์ urgent pointer มีหน้าที่ระบุตำแหน่งจุดสิ้นสุดของข้อมูลเร่งด่วนภายในเซกเมนต์

ACK ใช้เพื่อบอกว่ามีการใช้งานฟิลด์ acknowlegdement number สำหรับกระบวนการตอบรับเซกเมนต์อยู่หรือไม่ โดยทั่วไปฟิลด์นี้มีใช้งานในเซกเมนต์แทบทุกตัวยกเว้นก็เฉพาะแต่เซกเมนต์ที่ใช้ในการเปิดการเชื่อมต่อระหว่างโฮสต์ในครั้งแรกเท่านั้น

PSH บิตนี้จะถูกเซตในกรณีที่ต้องการให้ชั้นโปรโตคอลทีซีพีของโฮสต์ปลายทางส่งต่อข้อมูลในเซกเมนต์ไปให้โปรแกรมแอพพลิเคชั่นทันที ทั้งนี้เพราะโดยปกติโปรโตคอลทีซีพีจะสะสมและเก็บเซกเมนต์ไว้จนกว่าจะมีปริมาณมากพอ จึงค่อยส่งต่อให้โปรแกรมแอพพลิเคชั่น เพื่อลดปริมาณงานการติดต่อลง บิต PSH นี้มีความสำคัญต่อโปรแกรมแอพพลิเคชั่นบางประเภทที่มีการส่งข้อมูลทีละอักขระ และมีการโต้ตอบไปมาระหว่างสองฝ่าย

RST บิตนี้จะใช้งานเมื่อมีความผิดพลาดของการทำงานเกิดขึ้น และระบบไม่สามารถจัดการกับปัญหาเหล่านี้ได้อีกต่อไป การส่งเซกเมนต์ที่เซตบิต RST จึงเป็นกลไกในการปิดการเชื่อมต่อหรือสิ้นสุดการเชื่อมต่อที่มีอยู่ลง หรือยังอาจใช้ในการตอบปฏิเสธการขอเปิดการเชื่อมต่อได้ด้วยเมื่อมีการติดต่อขอสร้างการเชื่อมต่อเข้ามา

SYN บิตนี้ใช้งานเฉพาะสำหรับการแสดงความประสงค์ขอเปิดการเชื่อมต่อระหว่างโฮสต์ เริ่มแรกโฮสต์ด้านหนึ่งจะส่งเซกเมนต์ที่มีการเซตบิต SYN ออกไป หากโฮสต์ปลายทางยินดีจะตอบรับการเปิดการเชื่อมต่อก็จะส่งเซกเมนต์ที่เซตบิต SYN ตอบรับกลับไป สำหรับรายละเอียดในการเปิดการเชื่อมต่อจะได้กล่าวถึงในภายหลัง

FIN บิตนี้มีหน้าที่กลับกันกับบิต SYN คือมีไว้สำหรับโฮสต์ใช้ในการแสดงความจำนงขอปิดหรือสิ้นสุดการเชื่อมต่อ เนื่องจากไม่มีข้อมูลเหลือสำหรับส่งอีกต่อไป

ฟิลด์ window size ขนาด 16 บิต มีไว้สำหรับโฮสต์ปลายทางใช้ในการประกาศขนาดของวินโดว์ ที่ตนอนุญาตให้โฮสต์ต้นทางใช้งานได้ ขนาดของวินโดว์เป็นตัวกำหนดจำนวนไบต์สตรีมที่โฮสต์ต้นทางสามารถส่งออกอย่างต่อเนื่องโดยไม่ต้องรอการตอบรับจากโฮสต์ปลายทาง ฉะนั้นโฮสต์ปลายทางจึงสามารถควบคุมปริมาณหรืออัตราการส่งเซกเมนต์ของโฮสต์ต้นทางได้ตามที่ตนเห็นว่าเหมาะสม เช่น ถ้ากำหนดวินโดว์เท่ากับ 0 ก็หมายความว่าโฮสต์ปลายทางจะไม่ได้รับอนุญาตให้ส่งเซกเมนต์ข้อมูลมาอีก

ฟิลด์ checksum ขนาด 16 บิต ทำหน้าที่ตรวจสอบความถูกต้องขององค์ประกอบทุกส่วนในเซกเมนต์คือ ทั้งเฮดเดอร์และส่วนของข้อมูล วิธีการคำนวณค่าผลรวมตรวจสอบเหมือนกันกับกรณีของโปรโตคอลยูดีพีคือ นำองค์ประกอบทุกส่วนมาบวกรวมกันแบบ 1 คอมพลีเมนต์ทีละ 16 บิต โดยที่มีการนำเฮดเดอร์เทียมมาร่วมคำนวณด้วย ดูรูปที่ 2.11 หากจำนวนบิตของส่วนสุดท้ายที่นำมาคำนวณไม่ครบลงตัวที่ 16 บิต ให้เติมศูนย์เข้าไปเพื่อให้ครบ ต้องไม่ลืมว่าโปรโตคอลทีซีพีไม่มีการส่งเฮดเดอร์เทียมหรือศูนย์ที่เพิ่มเข้ามาไปกับเซกเมนต์แต่อย่างใด องค์ประกอบเหล่านี้ใช้เพื่อการคำนวณผลรวมตรวจสอบเท่านั้น ผลลัพธ์ที่ได้จากการบวกให้นำมาทำ 1 คอมพลีเมนต์อีกทอด

หนึ่งก่อนนำไปบรรจุลงในฟิลด์ผลรวมตรวจสอบ พิจารณาดูองค์ประกอบของเฮดเดอร์เทียมจะพบว่ามีองค์ประกอบหมายเลขไอพีแอดเดรสของโฮสต์ต้นทางและปลายทาง ชนิดของโปรโตรคอล และความยาวของเซกเมนต์ทีซีพี ข่าวสารเหล่านี้ได้มาจากโปรโตคอลในชั้นไอพีทั้งหมด การนำข่าวสารบางส่วนจากไอพีมาตรวจสอบในชั้นของโปรโตคอลทีซีพีด้วย ก็เพื่อให้แน่ใจว่าเซกเมนต์ทีซีพีส่งถึงปลายทางที่ถูกต้องตามต้องการจริงคือตรวจทั้งหมายเลขไอพีแอดเดรสควบคู่ไปกับหมายเลขพอร์ตทั้งโฮสต์ต้นทางและปลายทาง สำหรับโปรโตคอลทีซีพี ฟิลด์ protocal มีค่าเท่ากับ 6 โดยรวมแล้วการคำนวณผลรวมตรวจสอบของโปรโตคอลทีซีพีเป็นข้อกำหนดที่ต้องทำเสมอ ในขณะที่ผู้พัฒนาซอฟต์แวร์ของโปรโตคอลยูดีพีสามารถเลือกทำหรือไม่ทำก็ได้ สำหรับการตรวจสอบความถูกต้องที่โฮสต์ปลายทางก็ทำได้โดยนำ องค์ประกอบทั้งหมดมาบวกกันแบบ 1 คอมพลีเมนต์ หากพบว่าได้ผลลัพธ์เป็น 1 ทั้ง 16 บิต ก็แสดงว่าไม่มีความผิดพลาดกับเซกเมนต์นั้น

รูปที่ 2.11 โครงสร้างเฮดเดอร์เทียม (Pseudo header) ของโปรโตคอลทีซีพี

ฟิลด์ urgent pointer ขนาด 16 บิต จะมีความหมายก็เฉพาะเมื่อมีการเซตบิต URG = 1 เท่านั้น เมื่อโฮสต์ปลายทางได้รับเซกเมนต์ที่มีการเซตบิต URG ก็จะอ่านค่าในฟิลด์ urgent pointer เพื่อให้ทราบถึงตำแหน่งไบต์สุดท้ายของข้อมูลเร่งด่วนในเซกเมนต์นั้น

ฟิลด์ options ในเฮดเดอร์โปรโตคอลทีซีพีมีขนาดที่แปรเปลี่ยนได้ โดยขนาดที่แน่นอนของแต่ละเซกเมนต์สามารถดูได้จากฟิลด์ header length

2.6.3 การสร้างและสิ้นสุดการเชื่อมต่อทีซีพี

2.6.3.1 ขั้นตอนการสร้างและสิ้นสุดการเชื่อมต่อ

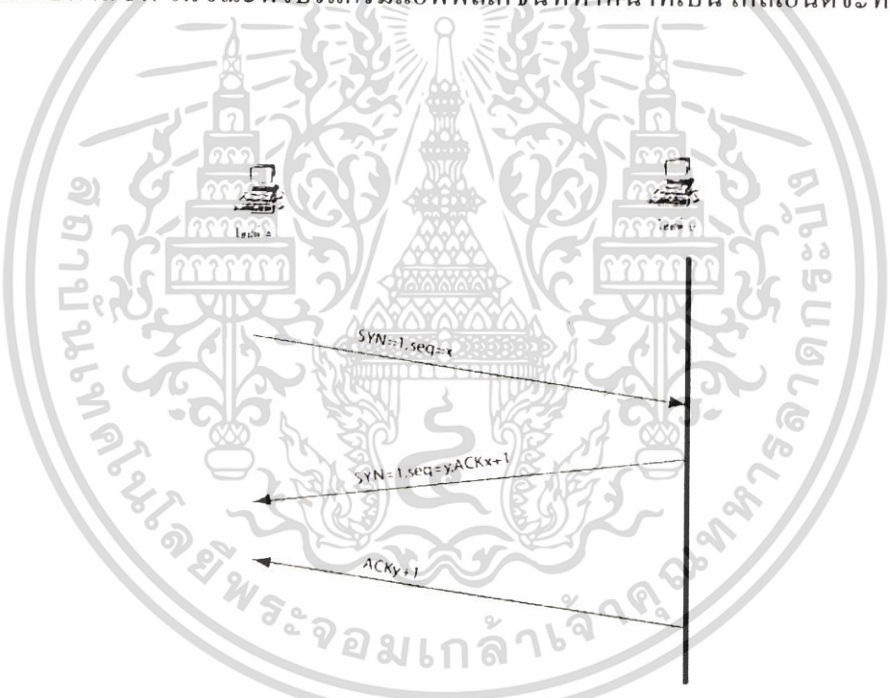
กระบวนการสร้างการเชื่อมต่อของโปรโตคอลทีซีพีระหว่างโฮสต์ 2 ตัว อาศัยการโต้ตอบไปมาทั้งหมด 3 ครั้งดังรูปที่ 2.12

1. ขั้นแรกโฮสต์ที่ต้องการเริ่มการติดต่อ (โฮสต์ A) ส่งเซกเมนต์ที่เซตบิต SYN ในฟิลด์ code ให้เป็น 1 พร้อมกับเลือกหมายเลข seq ขึ้นค่าหนึ่งเพื่อบรรจุลงในฟิลด์ squence number ในตัวอย่างนี้คือ x ค่า seq ที่เลือกนี้จะใช้เป็นค่าเริ่มต้นสำหรับการติดต่อสื่อสารที่จะเกิดขึ้นในลำดับต่อไป

2. ขั้นต่อมาโฮสต์อีกด้านหนึ่ง (โฮสต์ B) ตอบรับการขอเปิดการเชื่อมต่อด้วยเซกเมนต์ที่เซตบิต SYN = 1 และ ACK=1 ของฟิลด์ code พร้อมกันนั้นก็เลือกหมายเลข seq ของตนเพื่อบรรจุลงในฟิลด์ squence number สำหรับใช้เป็นค่าเริ่มต้นของหมายเลขที่ใช้กำกับให้กับเซกเมนต์ที่จะส่งในลำดับต่อไป ในการตอบรับด้วยบิต ACK จะใช้ควบคู่กับฟิลด์ acknowledge number ซึ่งจะบรรจุเป็นค่า x+1 สังเกตว่าเซกเมนต์ SYN จะใช้หมายเลข squence number ไปหนึ่ง

3. ขั้นสุดท้ายโฮสต์ A ยืนยันการเปิดการเชื่อมต่อกับโฮสต์ B โดยส่งเป็นเซกเมนต์ที่เซตบิต ACK เท่านั้น ไม่ใช้บิต SYN อีกต่อไป ส่วนค่าในฟิลด์ acknowledge number ตั้งให้มีค่าเท่ากับ y+1

การเปิดการเชื่อมต่อแบบนี้เป็นแบบที่โฮสต์ด้านหนึ่งแอกทีฟ คือ ริเริ่มและร้องขอการเปิด (active open) ในขณะที่โฮสต์อีกด้านหนึ่งจะทำงานเป็นแบบพาสซีฟ คือจะไม่เริ่มขอเองแต่จะรอโฮสต์อื่นมาขอต่อเชื่อม (passive open) โดยทั่วไปโปรแกรมแอพพลิเคชั่นที่ทำหน้าที่เป็นเซิร์ฟเวอร์จะทำงานแบบพาสซีฟ ในขณะที่โปรแกรมแอพพลิเคชั่นที่ทำหน้าที่เป็นไคลเอนต์จะทำงานแบบแอกทีฟ

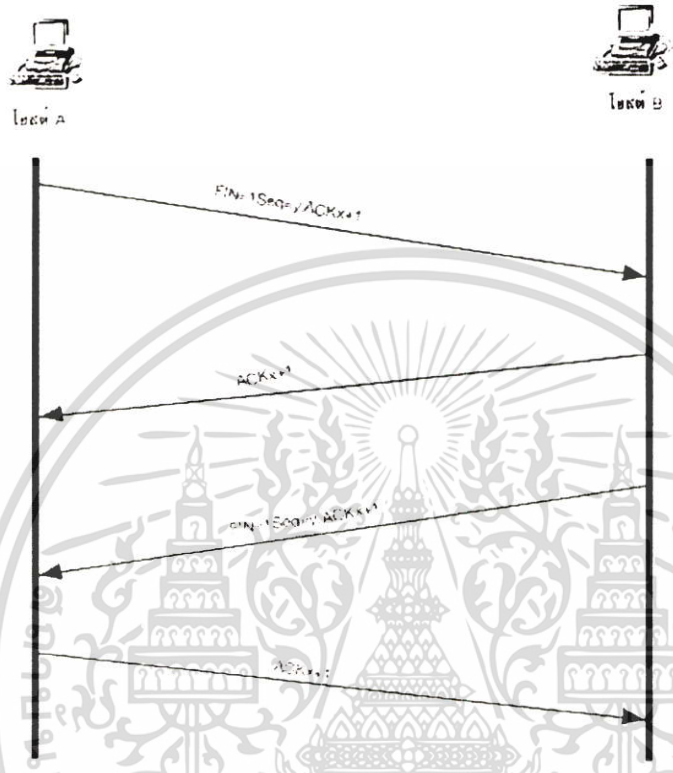


รูปที่ 2.12 ขั้นตอนการสร้างการเชื่อมต่อเพื่อขอเปิดการเชื่อมของโปรโตคอลทีซีพี

2.6.3.2 ขั้นตอนการสิ้นสุดการเชื่อมต่อทีซีพี

เนื่องจากการเชื่อมต่อสื่อสารตามโปรโตคอลทีซีพีเป็นแบบฟูลดูเพลกซ์ (full duplex) คือสามารถรับส่งข้อมูลทั้งสองทิศทางได้พร้อมกันและไม่ขึ้นแก่กันและกัน กระบวนการปิดการเชื่อมต่อเพื่อสิ้นสุดการเชื่อมต่อระหว่างโฮสต์ในโปรโตคอลทีซีพี ก็สามารถแยกทำแต่ละทิศทางอิสระจากกันได้ คือ เมื่อใดที่โฮสต์ด้านใดด้านหนึ่งไม่มีข้อมูลที่จะส่งอีกต่อไป โฮสต์ดังกล่าวสามารถจะ

ขอปิดการเชื่อมต่อสำหรับทิศทางการส่งของตัวเองเพียงด้านเดียวได้โดยส่งเซกเมนต์ที่ได้เซตบิต FIN = 1 ออกเพื่อแสดงความจำนงเพื่อขอปิดการส่งข้อมูลลง โฮสต์อีกด้านหนึ่งก็จะตอบรับความประสงค์โดยส่งเซกเมนต์ที่เซตบิต ACK = x+1 ในขณะเดียวกันก็แจ้งให้โปรแกรมแอพพลิเคชั่นของตนรับทราบว่าจะไม่มีข้อมูลใหม่เข้ามาจากการเชื่อมต่อดังกล่าวอีกแล้ว



รูปที่ 2.13 ขั้นตอนการสิ้นสุดการเชื่อมต่อเพื่อขอปิดการเชื่อมต่อของโปรโตคอลทีซีพี

บทที่ 3
เทคโนโลยีที่เกี่ยวข้อง

เทคโนโลยีที่เกี่ยวข้องกับงานวิจัยนี้คือ เทคโนโลยีของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ งานวิจัยนี้จะทำการปรับปรุงความสามารถของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ให้มีประสิทธิภาพมากขึ้น ดังนั้นก่อนที่เราจะสามารถเพิ่มความสามารถให้กับอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ได้นั้น เราต้องทำความเข้าใจหน้าที่และการทำงานของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์เสียก่อน

3.1 การกระจายโหลดหรือแบ่งโหลดให้กับไฟร์วอลล์

หน้าที่หลักของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ คือ ทำการแบ่งโหลดหรือกระจายโหลดให้กับไฟร์วอลล์ในกลุ่ม (Farm) ที่ผู้ดูแลระบบกำหนด โดยอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะใช้อัลกอริทึมที่ผู้ดูแลระบบเลือก สำหรับกระจายโหลดให้กับไฟร์วอลล์ที่อยู่ในกลุ่มที่ตนดูแลอัลกอริทึมที่ผู้ดูแลระบบสามารถเลือกใช้ โดยทั่วไปแล้วมีตัวอย่างดังต่อไปนี้

การกระจายโหลดแบบวนรอบ (Cyclic) การกระจายโหลดแบบนี้เมื่อมีการเชื่อมต่อระหว่างเครื่องไคลเอนต์กับเครื่องเซิร์ฟเวอร์ผ่านอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะทำการกระจายโหลดของการเชื่อมต่อไปยังไฟร์วอลล์ที่ตนดูแลแบบตามลำดับเป็นวงรอบ ไฟร์วอลล์ตัวละตัวจะได้รับการกระจายการเชื่อมต่อให้รอบละหนึ่งการเชื่อมต่อ ทราฟฟิคของแต่ละการเชื่อมต่อ จะถูกส่งผ่านไฟร์วอลล์ตัวเดิมเสมอจนกระทั่งการเชื่อมต่อนั้นสิ้นสุด ยกตัวอย่าง เมื่อมีการเชื่อมต่อแรกเกิดขึ้นอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะทำการส่งทราฟฟิคของการเชื่อมต่อแรกนี้ผ่านไฟร์วอลล์ตัวที่ 1 เมื่อมีการเชื่อมต่อที่สองเกิดขึ้นอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ก็จะทำการส่งทราฟฟิคของการเชื่อมต่อที่สองผ่านไฟร์วอลล์ตัวที่ 2 อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะทำเช่นนี้ไปเรื่อย ๆ จนกระทั่งถึงไฟร์วอลล์ตัวสุดท้าย หลังจากทำการกระจายการเชื่อมต่อไปยังไฟร์วอลล์ตัวสุดท้ายแล้ว อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ก็จะเริ่มกลับไปกระจายการเชื่อมต่อที่เกิดขึ้นใหม่ให้กับไฟร์วอลล์ตัวที่ 1 ใหม่ ซึ่งจะเห็นว่าการกระจายโหลดแบบนี้จะมีลักษณะเป็นวงรอบ จึงถูกเรียกว่าการกระจายโหลดแบบวนรอบ

แบบกำหนดค่าน้ำหนัก (Weight Round Robin) การกระจายโหลดแบบนี้เป็นการกระจายโหลดที่นำเอาการกระจายโหลดแบบวนรอบมาปรับปรุงให้ดีขึ้น โดยการกระจายโหลดแบบกำหนดค่าน้ำหนักนี้จะทำการกระจายโหลดในลักษณะเป็นวงรอบ เช่นเดียวกับการกระจายโหลดแบบวนรอบ แต่ผู้ดูแลระบบสามารถที่จะกำหนดได้ว่าจะทำการกระจายโหลดให้กับไฟร์วอลล์แต่ละตัวเป็นสัดส่วนเท่าไร โดยค่าที่กำหนดให้กับไฟร์วอลล์ตัวละตัวนี้เราเรียกว่า ค่าน้ำหนัก (weight) บาง

ไฟร์วอลล์อาจจะถูกกำหนดให้ได้รับการกระจายโหลดของการเชื่อมต่อใหม่เพียงรอบละ 1 การเชื่อมต่อ บางไฟร์วอลล์อาจถูกกำหนดให้ได้รับการกระจายโหลดของการเชื่อมต่อใหม่รอบละ 2 หรือ 3 หรือ 4 การเชื่อมต่อ การกระจายโหลดด้วยวิธีนี้จะทำให้ผู้ดูแลระบบสามารถกำหนดได้ว่าควรจะกระจายโหลดให้กับไฟร์วอลล์แต่ละตัวเป็นสัดส่วนเท่าไรในหนึ่งรอบ ซึ่งจะทำให้สามารถทำการกระจายโหลดหรือแบ่งโหลดให้เหมาะสมกับความสามารถในการประมวลผลของไฟร์วอลล์แต่ละตัว การกระจายโหลดแบบนี้จะนิยมใช้ในกรณีที่ไฟร์วอลล์แต่ละตัวมีความสามารถในการประมวลผลไม่เท่ากัน

แบบจำนวนการเชื่อมต่อน้อยที่สุด (Least number of connections) การกระจายโหลดแบบนี้จะเป็นการกระจายโหลดที่พิจารณาจำนวนของการเชื่อมต่อที่ถูกกระจายหรือแบ่งไปให้ไฟร์วอลล์ตัวละตัว โดยจะทำการกระจายการเชื่อมต่อใหม่ไปให้ไฟร์วอลล์ตัวที่มีจำนวนของการเชื่อมต่อถูกแบ่งไปให้น้อยที่สุดในเวลานั้น ยกตัวอย่างเช่น ถ้าไฟร์วอลล์ตัวที่ 1 มีการเชื่อมต่อถูกแบ่งไปให้ในขณะนั้น 1 การเชื่อมต่อ ไฟร์วอลล์ตัวที่ 2 มีการเชื่อมต่อถูกแบ่งไปให้ในขณะนั้น 2 การเชื่อมต่อ และไฟร์วอลล์ตัวที่ 3 มีการเชื่อมต่อถูกแบ่งไปให้ในขณะนั้น 3 การเชื่อมต่อ เมื่อมีการเชื่อมต่อใหม่เกิดขึ้นการเชื่อมต่อใหม่นี้จะถูกแบ่งหรือการะจายไปให้ไฟร์วอลล์ตัวที่ 1 เนื่องจากไฟร์วอลล์ตัวที่ 1 มีจำนวนของการเชื่อมต่อถูกแบ่งไปให้น้อยที่สุดในขณะนั้น

แบบจำนวนแพ็คเก็ตน้อยที่สุด (Least amount of packets) การกระจายโหลดแบบนี้จะคล้ายกับการกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุด แต่จะเปลี่ยนจากการพิจารณาจำนวนของการเชื่อมต่อที่ถูกกระจายไปให้ไฟร์วอลล์แต่ละตัวในขณะนั้น มาทำการพิจารณาจำนวนแพ็คเก็ตที่ผ่านไฟร์วอลล์แต่ละตัวในขณะนั้นแทน โดยการกระจายโหลดด้วยวิธีนี้จะทำการกระจายโหลดของการเชื่อมต่อที่เกิดขึ้นใหม่ไปยังไฟร์วอลล์ตัวที่มีจำนวนแพ็คเก็ตผ่านน้อยที่สุดในเวลานั้น

แบบจำนวนไบต์น้อยที่สุด (Least amount of bytes) การกระจายโหลดแบบนี้จะคล้ายกับการกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุด และแบบจำนวนแพ็คเก็ตน้อยที่สุด แต่สิ่งที่แตกต่างจากการกระจายโหลดทั้ง 2 แบบที่กล่าวมา คือ การกระจายโหลดแบบนี้จะกระจายโหลดของการเชื่อมต่อที่เกิดขึ้นใหม่ไปยังไฟร์วอลล์ตัวที่มีจำนวนข้อมูลคิดเป็นไบต์ผ่านน้อยที่สุดในเวลานั้น โดยการกระจายโหลดแบบนี้จะพิจารณาจำนวนข้อมูลเป็นไบต์ แทนที่จะพิจารณาจำนวนการเชื่อมต่อ หรือจำนวนแพ็คเก็ต

แบบวิธีการแฮ็ช (Hashing dispatch method) การกระจายโหลดวิธีนี้จะใช้การคำนวณทางคณิตศาสตร์มาช่วยในการคำนวณหาหมายเลขประจำตัวไฟร์วอลล์ ที่จะทำการกระจายการเชื่อมต่อใหม่ไปให้ โดยการคำนวณหาหมายเลขประจำตัวไฟร์วอลล์นี้ จะนิยมใช้หมายเลขไอพีแอดเดรสต้นทางร่วมกับหมายเลขไอพีแอดเดรสปลายทางของแพ็คเก็ตของการเชื่อมต่อที่กำลังถูกพิจารณามาใช้ในการคำนวณหาหมายเลขประจำตัวไฟร์วอลล์ หรืออาจจะเลือกใช้เฉพาะหมายเลขไอพีแอดเดรสต้นทาง หรือหมายเลขไอพีแอดเดรสปลายทางอย่างใดอย่างหนึ่งมาใช้ในการคำนวณก็ได้

นอกจากนั้นยังสามารถนำหมายเลขพอร์ตมาคำนวณร่วมด้วยได้เช่นเดียวกัน หลังจากทำการคำนวณแล้วจะได้หมายเลขประจำตัวไฟร์วอลล์ที่จะทำการกระจายการเชื่อมต่อนั้นไปให้นั่นเอง

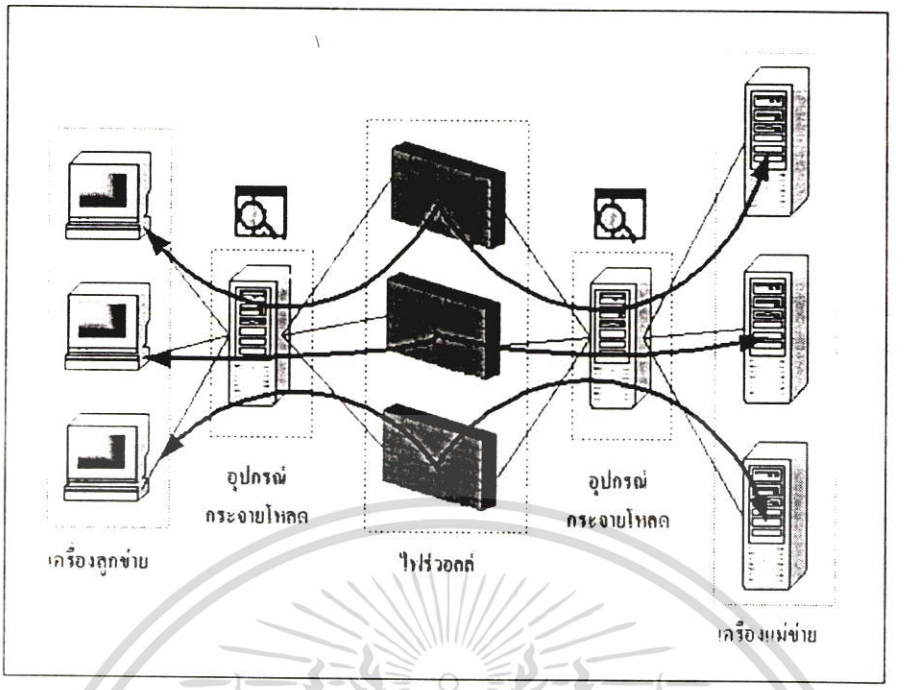
3.2 การตรวจสอบสถานะของไฟร์วอลล์

นอกจากทำหน้าที่กระจายโหลดให้กับไฟร์วอลล์แล้ว อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ยังทำหน้าที่ตรวจสอบสถานะของไฟร์วอลล์แต่ละตัวที่อยู่ในกลุ่มที่ตนเองดูแลอยู่เสมออีกด้วย โดยถ้าไฟร์วอลล์ตัวใดตัวหนึ่งไม่สามารถทำงานต่อด้วยสาเหตุใดก็ตาม อุปกรณ์กระจาย-โหลดระหว่างไฟร์วอลล์จะทำการตัดไฟร์วอลล์ตัวนั้นออกจากระบบ และจะไม่ทำการแบ่งโหลดหรือกระจายการเชื่อมต่อที่เกิดขึ้นใหม่ไปให้ไฟร์วอลล์ตัวนั้นอีกต่อไป ส่วนการเชื่อมต่อที่ถูกกระจายไปให้ไฟร์วอลล์ตัวที่ไม่สามารถทำงานต่อนั้นก็จะต้องหยุดตามไฟร์วอลล์ไปโดยปริยาย

3.3 ทำการส่งทราฟฟิคของแต่ละการเชื่อมต่อผ่านไฟร์วอลล์ตัวเดิมเสมอ

อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ต้องทำการส่งทุกแพ็คเก็ตของการเชื่อมต่อใด ๆ ผ่านไฟร์วอลล์ตัวเดิมเสมอ สาเหตุที่ต้องส่งทุกแพ็คเก็ตผ่านไฟร์วอลล์ตัวเดิม เพราะไฟร์วอลล์ในปัจจุบันโดยทั่วไปแล้วจะเป็นไฟร์วอลล์แบบรักษาสถานะ ไฟร์วอลล์แบบรักษาสถานะจะมีการเก็บข้อมูลของการเชื่อมต่อในตารางสถานะในช่วงที่มีการสร้างการเชื่อมต่อจากเครื่องไคลเอนต์ไปยังเครื่องเซิร์ฟเวอร์ผ่านตัวมัน และไฟร์วอลล์แบบรักษาสถานะจะอนุญาตให้เฉพาะทราฟฟิคของการเชื่อมต่อที่มีข้อมูลอยู่ในตารางสถานะของมันผ่าน ดังนั้นอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ต้องคอยส่งแพ็คเก็ตของแต่ละการเชื่อมต่อผ่านไฟร์วอลล์ตัวเดิมเสมอ สามารถดูรายละเอียดของการทำงานของไฟร์วอลล์แบบรักษาสถานะเพิ่มเติมได้ในบทต่อไป

การติดตั้งอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ โดยทั่วไปแล้วจะแตกต่างจากการติดตั้งอุปกรณ์กระจายโหลดแบบอื่น ๆ เช่น อุปกรณ์กระจายโหลดสำหรับเครื่องแม่ข่าย (Server Load balancer) หรืออุปกรณ์กระจายโหลดสำหรับสายสื่อสาร (Link Load balancer) โดยการติดตั้งอุปกรณ์กระจายโหลดระหว่างไฟล์วอลล์ ต้องทำการติดตั้งอุปกรณ์กระจายโหลดสองตัวที่ด้านหน้าและหลังไฟร์วอลล์ ดังรูปที่ 3.1



รูปที่ 3.1 แสดงการติดตั้งอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์

สาเหตุที่ต้องใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์สองตัวก็เพราะ อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ต้องคอยทำการเก็บข้อมูลของแต่ละการเชื่อมต่อว่าผ่านมาจากไฟร์วอลล์ตัวใด เพื่อที่อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะได้ทำการส่งทราฟฟิคของแต่ละการเชื่อมต่อกลับได้ถูกไฟร์วอลล์นั่นเอง

บทที่ 4
การทำงานของไฟร์วอลล์แบบรักษาสถานะ

ก่อนที่เราจะสามารถทำการย้ายการเชื่อมต่อระหว่างไฟร์วอลล์แบบรักษาสถานะได้นั้น เราจะต้องเข้าใจคุณสมบัติและกระบวนการทำงานของไฟร์วอลล์แบบรักษาสถานะเสียก่อน เราถึงจะสามารถหาวิธีมาทำการย้ายการเชื่อมต่อระหว่างไฟร์วอลล์แบบรักษาสถานะได้

โดยหากจะทำการจำแนกไฟล์วอลล์โดยใช้ลักษณะการทำงานเป็นเกณฑ์แล้ว เราจะสามารถทำการจำแนกไฟร์วอลล์ได้ดังต่อไปนี้

1. ไฟร์วอลล์แบบแพ็คเก็ตฟิลเตอริ่ง (Packet Filtering Firewall)
2. ไฟร์วอลล์แบบรักษาสถานะ (Stateful Firewall) หรือไฟร์วอลล์แบบตรวจสอบสถานะ (Stateful Inspection Firewall) ในวิทยานิพนธ์นี้จะใช้คำว่าไฟร์วอลล์แบบรักษาสถานะ ซึ่งจะมีความหมายเดียวกับไฟร์วอลล์แบบตรวจสอบสถานะนั่นเอง

4.1 ไฟร์วอลล์แบบแพ็คเก็ตฟิลเตอริ่ง (Packet Filtering Firewall)

เป็นไฟร์วอลล์แบบพื้นฐานที่มีความสามารถในการควบคุมทราฟฟิค โดยอาศัยการตรวจสอบข้อมูลที่ปรากฏอยู่ในส่วนหัวของแพ็คเก็ต ไฟร์วอลล์ประเภทนี้อาจจะเป็นความสามารถที่ฝังเพิ่มมาในเราเตอร์ (Router) โดยแทนที่เราเตอร์จะสามารถทำการเราต์ (การจัดเส้นทาง) แพ็คเก็ตไปตามทิศทางที่เหมาะสมเพียงอย่างเดียว จะสามารถทำการตรวจสอบเปรียบเทียบกับเงื่อนไขที่กำหนดไว้ก่อนที่จะทำการเราต์แพ็คเก็ตออกไปได้ด้วย

เงื่อนไขสำหรับการควบคุมการผ่านเข้าออกของข้อมูล เราเรียกว่า กฎการแอคเซส (Access Rules) หรือกฎในการควบคุมการผ่านเข้าออกของแพ็คเกต โดยทั่วไปรูปแบบของกฎการแอคเซสเบื้องต้นจะเป็นดังตารางที่ 4.1

ตารางที่ 4.1 แสดงองค์ประกอบของกฎการแอคเซส

แอดเดรสต้นทาง	แอดเดรสปลายทาง	โปรโตคอล	พอร์ตต้นทาง	พอร์ตปลายทาง	แอคชั่น

โดยที่ข้อมูลในส่วนหัวของแพ็คเก็ตจะถูกนำมาเปรียบเทียบกับค่าที่ระบุไว้ในกฎการแอคเซสทุกฟิลด์ยกเว้นฟิลด์แอคชั่น โดยฟิลด์แอคชั่นจะถูกใช้สำหรับกำหนดการกระทำเมื่อข้อมูลในส่วนหัวของแพ็คเก็ตนั้นตรงกับเงื่อนไขในกฎการแอคเซส

ตารางที่ 4.2 แสดงกฎการแอคเซส เพื่ออนุญาตให้สามารถใช้บริการเอชทีทีพี

แอดเดรสต้นทาง	แอดเดรสปลายทาง	โปรโตคอล	พอร์ตต้นทาง	พอร์ตปลายทาง	แอคชั่น
192.1.0.5	ANY	TCP	ANY	80	Accept

กฎการแอคเซสในตารางที่ 4.2 ระบุไว้ว่าจะอนุญาตให้แพ็คเก็ตที่มีไอพีแอดเดรสต้นทาง 192.1.0.5 ปลายทางใด ๆ ก็ได้ ใช้โปรโตคอลทีซีพี และหมายเลขพอร์ตปลายทางเท่ากับ 80 ผ่านไฟร์วอลล์ได้ หากไฟร์วอลล์มีกฎการแอคเซสนี้เพียงข้อเดียว ก็เท่ากับอนุญาตให้โฮสต์เพียงโฮสต์เดียวคือ โฮสต์ที่มีไอพีแอดเดรส 192.1.0.5 เท่านั้นที่สามารถใช้บริการเอชทีทีพี (ทีซีพีพอร์ต 80) ไปยังโฮสต์อื่นที่อยู่อีกฝั่งหนึ่งของไฟร์วอลล์ได้

จริง ๆ แล้วการกำหนดกฎการแอคเซสตามตารางที่ 4.2 จะยังไม่ครบถ้วน เป็นเพียงการกำหนดกฎการแอคเซสแค่ด้านเดียวเท่านั้น สำหรับแพ็คเก็ตฟิลเตอร์ริ่งไฟล์วอลล์เราต้องทำการกำหนดกฎการแอคเซส เพื่ออนุญาตให้ข้อมูลที่ตอบกลับจากเซิร์ฟเวอร์สามารถผ่านแพ็คเก็ตฟิวเตอริ่งไฟร์วอลล์ได้ด้วย ตารางที่ 4.3 แสดงกฎการแอคเซสที่ต้องทำการเพิ่มเติมสำหรับการส่งข้อมูลกลับ

สำหรับแพ็คเก็ตฟิลเตอริ่งไฟร์วอลล์นี้ ผู้ดูแลระบบต้องคอยตรวจสอบกฎการแอคเซสให้ถูกต้องครบถ้วนทั้งขาไปและขากลับด้วยตนเอง

ตารางที่ 4.3 แสดงกฎการแอคเซสสำหรับขากลับ

แอดเดรสต้นทาง	แอดเดรสปลายทาง	โปรโตคอล	พอร์ตต้นทาง	พอร์ตปลายทาง	แอคชั่น
ANY	192.1.0.5	TCP	80	ANY	Accept

นี้เป็นตัวอย่างเบื้องต้นซึ่งยังไม่ลงไปในรายละเอียดของกฎการแอคเซส เป็นเพียงแค่ตัวอย่างเพื่อให้ได้เห็นภาพและหลักการพื้นฐานในการควบคุมทราฟฟิคของแพ็คเก็ตฟิลเตอริ่งไฟร์วอลล์ สำหรับรายละเอียดและวิธีการของการกำหนดกฎการแอคเซสโดยละเอียดนั้นจะกล่าวถึงต่อไป

ส่วนใหญ่การควบคุมทราฟฟิคโดยใช้เราเตอร์มักจะไม่เรียกว่าไฟร์วอลล์ เนื่องจากขีดจำกัดของความสามารถในการควบคุมและความคล่องตัวในการบริหารหรือกำหนดกฎต่าง ๆ การกำหนดกฎการแอคเซสบนแพ็คเก็ตฟิงเตอริ่งไฟร์วอลล์หรือเราเตอร์นั้นค่อนข้างยาก และเหมาะกับวิศวกรเครือข่ายเท่านั้น การเข้าไปตั้งค่าในเราเตอร์จะต้องอาศัยความชำนาญทางเทคนิคพอสมควร และเนื่องจากเราเตอร์ถูกออกแบบมาให้ทำหน้าที่หลักเป็นเราเตอร์ ส่วนความสามารถในเรื่องแพ็คเก็ต-ฟิลเตอร์นั้นเป็นเพียงส่วนประกอบ ดังนั้นฟังก์ชันที่เกี่ยวข้องกับกฎการแอคเซสไม่ว่าจะเป็น การตรวจสอบ การแก้ไขกฎการแอคเซส รวมไปถึงการทดสอบผลในการป้องกันก็ทำได้ยากยิ่ง นอกจากนี้หากเราเตอร์ทำการประมวลผลกฎการแอคเซสที่ซับซ้อนหรือมากเกินไป ประสิทธิภาพในการประมวลผลงานหลักของเราเตอร์ก็จะต่ำลงได้

4.2 ไฟร์วอลล์แบบรักษาสถานะ (Stateful Firewall) หรือไฟร์วอลล์แบบตรวจสอบสถานะ (Stateful Inspection Firewall)

การเชื่อมต่อโดยทั่วไปจะเป็นการสื่อสารแบบต่อเนื่อง โต้ตอบกันไปมาระหว่างผู้รับและผู้ส่งอยู่เสมอ โปรโตคอลที่อยู่ในระดับชั้นที่สูงกว่าระดับชั้นเครือข่าย ไม่ว่าจะเป็นระดับชั้นทรานสปอร์ตอย่าง เช่น ทีซีพี และยูดีพี หรือเลยไปถึงระดับชั้นแอพพลิเคชั่น เช่น เอฟทีพี, เอชทีทีพี หรือเอสเอ็มทีพี ล้วนแล้วแต่จะต้องมีสถานะของการสื่อสาร (State) เสมอ สถานะนี้จะทำให้ทั้งสองฝั่งสามารถสื่อสารกันได้อย่างต่อเนื่อง คือ ทราบว่าตอนนี้กำลังอยู่ ณ จุดใด และจะต้องส่งหรือรับข้อมูลใดเป็นลำดับต่อไป

4.2.1 ความแตกต่างของการพิจารณาข้อมูลแบบแพ็คเก็ตกับแบบรักษาสถานะ (Stateful)

อันที่จริงสองเรื่องนี้มิได้ขัดแย้งกันแต่ประการใด แพ็คเก็ตนั้นเป็นการสื่อสารที่เป็นส่วนย่อยของการสื่อสารทั้งหมด ผลของการเชื่อมต่อก็คือ ผลรวมของหลาย ๆ แพ็คเก็ตนั่นเอง แต่อย่างไรก็ตามการฟิลเตอร์หรือกรอง โดยพิจารณาที่ละแพ็คเก็ตของทุกแพ็คเก็ตที่ผ่านเข้าออกนั้น อาจจะมีผลลัพธ์แตกต่างจากการฟิลเตอร์ในแบบที่มองสถานะและภาพรวม หรือที่เรียกว่าแบบรักษาสถานะ (Stateful) หากเปรียบเทียบการพิจารณาข้อมูลครั้งละแพ็คเก็ตกับพิจารณาแบบรักษาสถานะแล้ว ตัวอย่างที่น่าจะช่วยทำให้เข้าใจได้ง่ายขึ้นคือ

เปรียบเทียบการสื่อสารข้อมูลทั้งหมดเสมือนภาพยนตร์ แพ็คเก็ตก็จะหมายถึงภาพนิ่งแต่ละภาพที่เมื่อนำมาต่อรวมกันแล้วเปิดดูอย่างรวดเร็วภาพนิ่งเหล่านั้นก็จะกลายเป็นภาพเคลื่อนไหว ดังนั้นการพิจารณาแพ็คเก็ตก็เป็นเสมือนหนึ่งการดูภาพนิ่งทีละภาพ โดยที่แต่ละภาพไม่มีส่วนเกี่ยวข้องกัน ดังนั้นข้อมูลที่จะรับรู้ได้ก็คือ ข้อมูลที่ปรากฏบนภาพนั้น ๆ เท่านั้น หากจะมีการเซ็นเซอร์ก็จะอยู่บนพื้นฐานของสิ่งที่ปรากฏอยู่บนแต่ละภาพ แต่จะไม่สามารถเซ็นเซอร์เนื้อเรื่องซึ่งเป็นสิ่งที่เกิดขึ้นจากความสัมพันธ์ของภาพหลาย ๆ ภาพได้ มีโอกาสที่เป็นไปได้ว่ากิจกรรมบางชนิดที่หากดูเป็นภาพนิ่งแล้ว จะไม่รู้สึกว่าเป็นสิ่งที่ไม่เหมาะสม แต่หากนำภาพนิ่งมาดูอย่างต่อเนื่องเป็นภาพเคลื่อนไหวแล้วก็อาจจะเป็นสิ่งที่ไม่พึงปรารถนาที่จะให้ปรากฏบนภาพยนตร์ก็เป็นได้

ไฟร์วอลล์แบบรักษาสถานะเป็นไฟร์วอลล์ที่สามารถเข้าใจสถานะการสื่อสารทั้งกระบวนการการเชื่อมต่อจะสมบูรณ์ได้นั้นจะต้องมีทั้งการส่งและการรับอย่างสอดคล้องสัมพันธ์กัน หากไฟร์วอลล์จะสามารถควบคุมการสื่อสารได้จริงก็จะต้องสามารถเข้าใจกระบวนการของการสื่อสารตั้งแต่ต้นจนจบ ไฟร์วอลล์แบบรักษาสถานะเป็นไฟร์วอลล์ที่ทำการควบคุมทราฟฟิค โดยใช้หลักการคล้ายกับของแพ็คเก็ตฟิลเตอริ่ง แต่ไฟร์วอลล์แบบรักษาสถานะจะมีความสามารถในการวิเคราะห์และรับรู้ความต่อเนื่องของแพ็คเก็ตในโปรโตคอลในระดับสูงขึ้นไปมากกว่า ไม่ว่าจะเป็นทีซีพี,เอฟทีพี และเอชทีทีพี หรือแม้กระทั่งโปรโตคอลในระดับชั้นแอพพลิเคชั่นที่มีลักษณะ

เฉพาะตัวของแอพพลิเคชั่นที่ไฟร์วอลล์แบบแพ็คเก็ตฟิลเตอริ่งไม่สามารถแยกแยะได้ เช่น โอราเคิล (Oracle) และ เอช323 (H.323)

ไฟร์วอลล์แบบรักษาสถานะเป็นเครื่องมือที่ถูกออกแบบมาเพื่อทำหน้าที่ในการควบคุมทราฟฟิคโดยเฉพาะ ไม่ได้เป็นการดัดแปลงการทำงานมาจากเราเตอร์ จึงมีความสามารถในการควบคุมทราฟฟิคการกำหนดกฎการแอคเซส การบริหารรวมไปถึงความยืดหยุ่นของการควบคุมทราฟฟิค และประสิทธิภาพในการทำงานที่สูงกว่าไฟร์วอลล์แบบแพ็คเก็ตฟิลเตอริ่งเป็นอย่างมาก โดยทั่วไปหากพูดถึงไฟร์วอลล์จะหมายถึงไฟร์วอลล์ประเภทนี้นี่เอง

ตามที่ได้กล่าวไว้ข้างต้นว่า ความแตกต่างที่สำคัญของไฟร์วอลล์ทั้งสองชนิดในแง่ของการตรวจสอบทราฟฟิค คือ ไฟร์วอลล์แบบรักษาสถานะ (Stateful Firewall) มีความสามารถในการวิเคราะห์ทราฟฟิคที่ผ่านไปมาในระดับชั้นสูงขึ้นไปไม่ว่าจะเป็น ทีซีพี, ยูดีพี และไอซีเอ็มพี ได้อย่างสมบูรณ์ต่างจากไฟร์วอลล์แบบแพ็คเก็ตฟิลเตอริ่ง ที่สามารถวิเคราะห์ได้เฉพาะเท่าที่จะมีข้อมูลใน 1 แพ็คเก็ตเท่านั้น เพราะบางครั้งทราฟฟิคที่ผ่านไปมานั้นมีการเชื่อมโยงกันหลายแพ็คเก็ตโดยเฉพาะทีซีพี ซึ่งจะมีลำดับของการติดต่อสื่อสารที่สัมพันธ์กันในแต่ละแพ็คเก็ต การพิจารณาแพ็คเก็ตใดแพ็คเก็ตหนึ่งโดยไม่พิจารณาแพ็คเก็ตอื่นที่เกี่ยวข้องกันอาจจะไม่สามารถควบคุมทราฟฟิคของทีซีพีได้ นอกจากนี้ยังรวมไปถึงการที่ไฟร์วอลล์แบบรักษาสถานะมีความสามารถในการประกอบรวมแฟรกเมนต์เข้าด้วยกันให้เป็นดาต้าแกรมที่สมบูรณ์ก่อน หลังจากนั้นจึงนำดาต้าแกรมนั้นมาทำการตรวจสอบเปรียบเทียบกับกฎการแอคเซส

นอกจากการเชื่อมโยงกันของหลายแพ็คเก็ตสำหรับโปรโตคอลทีซีพีในระดับชั้นทรานสปอร์ตแล้ว ในระดับชั้นแอพพลิเคชั่นก็มีแอพพลิเคชั่นบางชนิดที่จะต้องอาศัยการพิจารณาทราฟฟิคอย่างต่อเนื่อง เพื่อที่จะนำมากำหนดเป็นกฎการแอคเซสได้ ยกตัวอย่างเช่น การทำงานของโปรโตคอลเอฟทีพี ซึ่งในระหว่างการทำงานของแอพพลิเคชั่นนี้ โฮสต์ที่เป็นไคลเอ็นต์จะสามารถกำหนดพอร์ตชั่วคราวขึ้นมาทำหน้าที่เป็นเซิร์ฟเวอร์พอร์ตใช้สำหรับรับและส่งไฟล์ได้ โดยที่พอร์ตเหล่านี้จะปิดลงเมื่อการรับและส่งข้อมูลเสร็จสิ้นสมบูรณ์ ซึ่งในกรณีเช่นนี้หากไม่มีการพิจารณาทราฟฟิคที่มีมาก่อนหน้าแล้ว ไฟร์วอลล์อาจจะถือได้ว่าการเปิดเซิร์ฟเวอร์พอร์ตชั่วคราวของเอฟทีพีไคลเอ็นต์นั้นเป็นการเปิดให้บริการใหม่ขึ้นมาได้ ดังนั้นไฟร์วอลล์แบบรักษาสถานะจึงมีการทำงานที่ค่อนข้างใกล้ชิดกับแอพพลิเคชั่นเป็นอย่างมาก จะต้องสามารถเข้าใจคุณสมบัติของการเชื่อมต่อของแต่ละแอพพลิเคชั่นได้ค่อนข้างดี เพราะแอพพลิเคชั่นที่ใช้งานอยู่ในเครือข่ายไม่ได้มีเฉพาะแอพพลิเคชั่นพื้นฐานเท่านั้น มีแอพพลิเคชั่นอื่น ๆ อีกมาก แต่หากแอพพลิเคชั่นใดมีการใช้งานอย่างแพร่หลายและเป็นที่นิยมของผู้ใช้ โดยส่วนใหญ่ผู้ผลิตจะกำหนดวิธีการควบคุมทราฟฟิคสำเร็จรูปมาให้อยู่ในไฟร์วอลล์เลย อย่างไรก็ตามถึงแม้ว่าอาจจะมีแอพพลิเคชั่นที่ไม่ได้กำหนดเป็นสำเร็จรูปไว้ไฟร์วอลล์แบบรักษาสถานะก็เปิดโอกาสให้ผู้ใช้สามารถปรับแต่ง (Customize) ให้สามารถรู้จัก เข้าใจ และควบคุมทราฟฟิคของแอพพลิเคชั่นเหล่านั้นได้

การกำหนดกฎการแอคเซสของไฟร์วอลล์แบบรักษาสถานะนั้น ในเงื่อนไขส่วนที่เป็นเซอร์วิส (Service) ซึ่งโดยทั่วไปจะคิดว่าหมายถึงพอร์ตปลายทางนั้น จริง ๆ แล้วจะหมายถึงการบริการจริง ไม่ใช่หมายถึงเฉพาะพอร์ตปลายทาง ถึงแม้ว่าโดยส่วนใหญ่เกือบจะเป็นสิ่งเดียวกัน เช่น เอชทีทีพี ก็หมายถึง พอร์ต 80 ของ ทีซีพี ,ป๊อป-3 (POP-3) หมายถึงพอร์ต 110 ของทีซีพี เป็นต้น แต่ในบางกรณีการบริการจริงไม่ใช้หมายถึงเฉพาะพอร์ตตายตัวเช่น เอฟทีพี ที่กล่าวถึงในตอนต้น โดยปกติเอฟทีพีจะใช้พอร์ตหมายเลข 20 และ 21 เป็นหลัก แต่ก็มีในบางโหมดที่จะใช้พอร์ตหมายเลขอื่นเพิ่มเติมขึ้นมาเป็นเซิร์ฟเวอร์พอร์ตได้ ดังนั้นการกำหนดเซอร์วิส จึงไม่ใช่เป็นเพียงแค่การกำหนดพอร์ตปลายทางเสมอไป การบริการใด ๆ อาจจะมีการใช้พอร์ตที่เปลี่ยนแปลงไปได้ขึ้นอยู่กับแอพพลิเคชั่นที่ทำงาน แต่ไฟร์วอลล์แบบรักษาสถานะก็มีความสามารถมากพอที่จะติดตามไปควบคุมทราฟฟิคและบริการต่าง ๆ ได้ ซึ่งในส่วนนี้จะเห็นความแตกต่างกับไฟร์วอลล์แบบแพ็คเก็ตฟิลเตอริ่งได้อย่างชัดเจน เพราะไฟร์วอลล์แบบแพ็คเก็ตฟิลเตอริ่งนั้นจะควบคุมทราฟฟิคโดยการระบุหมายเลขพอร์ตที่ตายตัว และไม่สามารถยืดหยุ่นไปกับแอพพลิเคชั่นได้

วิธีการกำหนดกฎการแอสเซสของไฟร์วอลล์แบบรักษาสถานะก็เป็นอีกสิ่งหนึ่งที่แตกต่างกับไฟร์วอลล์แบบแพ็คเก็ตฟิงเตอริ่ง ไฟร์วอลล์แบบรักษาสถานะจะใช้การกำหนดกฎการแอคเซสแบบทิศทางเดียว หมายความว่าเมื่อจะทำการอนุญาตให้เครื่องไคลเอ็นต์ในเครือข่ายภายในสามารถใช้งานเอชทีทีพีในอินเตอร์เน็ตได้นั้น จะทำการกำหนดกฎการแอคเซสเพื่อให้ไคลเอ็นต์ผ่านออกไปภายนอกเท่านั้น ไม่จำเป็นต้องทำการกำหนดกฎการแอคเซส เพื่อให้ข้อมูลที่ตอบกลับมาจากอินเตอร์เน็ตสามารถผ่านเข้ามาภายใน ดังนั้นการกำหนดกฎการแอคเซสของไฟร์วอลล์แบบรักษาสถานะจะใช้เพียงกฎการแอคเซสที่แสดงในตารางที่ 4.2 ก็เพียงพอ

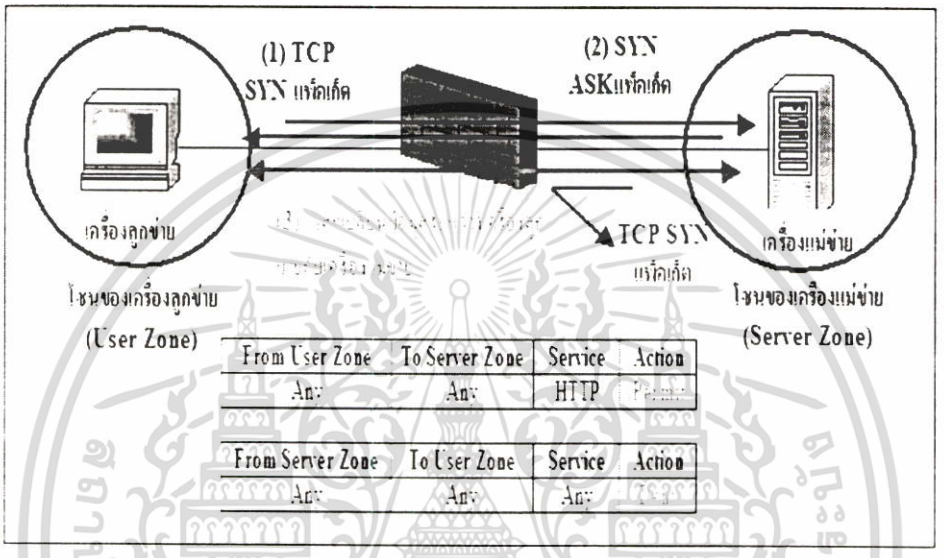
4.2.2 กระบวนการทำงานของไฟร์วอลล์แบบรักษาสถานะ

ไฟร์วอลล์แบบรักษาสถานะจะบันทึกข้อมูล เช่น ไอพีแอดเดรสต้นทาง ไอพีแอดเดรสปลายทาง หมายเลขพอร์ตต้นทาง และหมายเลขพอร์ตปลายทาง เป็นต้น ของการเชื่อมต่อที่ได้รับอนุญาตให้ผ่านตัวมันลงในตารางสถานะ (ไฟร์วอลล์แบบรักษาสถานะจากบางผู้ผลิตอาจเก็บข้อมูลมากกว่านี้) และจะอนุญาตให้เฉพาะทราฟฟิคของการเชื่อมต่อที่ได้มีการบันทึกข้อมูลข้างต้นไว้ผ่านเท่านั้น

การกำหนดกฎการแอคเซส (access rule) ให้ไฟร์วอลล์แบบรักษาสถานะ เพื่ออนุญาตให้ทราฟฟิคของการเชื่อมต่อใด ๆ ผ่านเข้าและออกระหว่างโซนนั้น จะใช้การกำหนดกฎการแอคเซสเพียงทิศทางเดียว เพื่ออนุญาตให้ทราฟฟิคสามารถข้ามระหว่างโซนได้ทั้งขาไปและขากลับ โดยไม่จำเป็นต้องมีการกำหนดกฎการแอคเซสสำหรับขากลับแต่อย่างไร

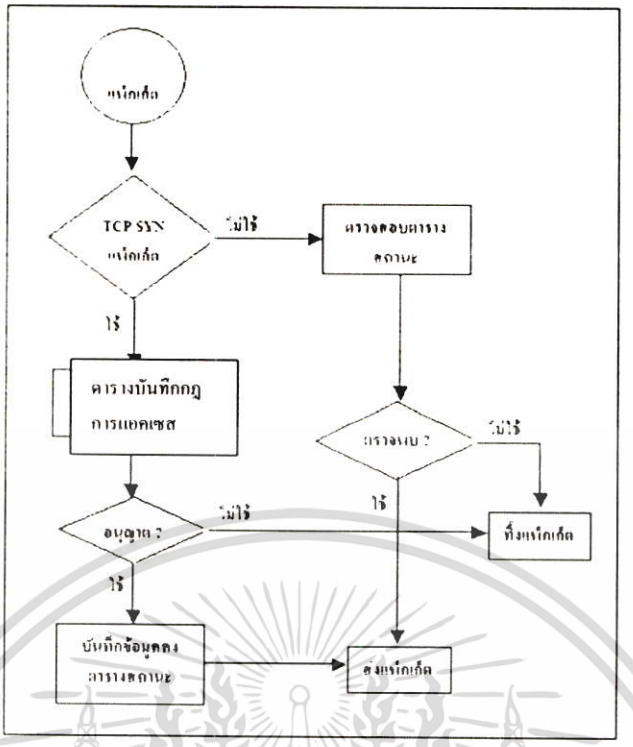
เมื่อมีการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์ที่อยู่ต่างโซนกันผ่านไฟร์วอลล์แบบรักษาสถานะ ไฟร์วอลล์แบบรักษาสถานะจะทำการตรวจสอบกฎการแอคเซสในตารางบันทึกกฎการแอคเซส (access table) ว่าจะอนุญาตให้การเชื่อมต่อนี้สามารถเกิดขึ้นได้หรือไม่ ถ้ามีกฎการแอคเซสที่

อนุญาตให้การเชื่อมต่อนี้สามารถเกิดขึ้นได้แล้ว ไฟร์วอลล์แบบรักษาสถานะจะทำการบันทึกข้อมูลของการเชื่อมต่อนี้ในตารางสถานะ โดยข้อมูลนี้เองจะทำให้ทราฟฟิคของการเชื่อมต่อนี้สามารถผ่านไฟร์วอลล์แบบรักษาสถานะได้ทั้งขาไปและขากลับ ยกตัวอย่างเช่น ถ้าต้องการอนุญาตให้เครื่องคอมพิวเตอร์จากโซนของเครื่องลูกข่าย (User zone) สามารถใช้งานเว็บแอพพลิเคชั่นบนเครื่องคอมพิวเตอร์ที่อยู่ในโซนของเครื่องแม่ข่าย (Server zone) จะทำการกำหนดกฎการแอคเซส ดังรูปที่ 4.1



รูปที่ 4.1 แสดงการทำงานของไฟร์วอลล์แบบรักษาสถานะ

จะเห็นว่าการกำหนดกฎการแอคเซสให้ไฟร์วอลล์แบบรักษาสถานะดังรูปที่ 4.1 จะทำการกำหนดกฎการแอคเซส เพื่ออนุญาตให้มีการเชื่อมต่อจากโซนของเครื่องลูกข่ายไปยังโซนของเครื่องแม่ข่ายเท่านั้น ทำให้ทราฟฟิคจากโซนของเครื่องแม่ข่ายไม่สามารถข้ามมายังโซนของเครื่องลูกข่ายได้ จนกว่าจะมีการสร้างการเชื่อมต่อจากเครื่องคอมพิวเตอร์ที่อยู่ในโซนของเครื่องลูกข่ายไปใช้งานเว็บแอพพลิเคชั่นที่อยู่ในในโซนของเครื่องแม่ข่ายเสียก่อน หลังจากนั้นทราฟฟิคของการเชื่อมต่อนั้นถึงจะสามารถข้ามจากโซนของเครื่องแม่ข่ายมายังโซนของเครื่องลูกข่ายได้ สำหรับเครื่องคอมพิวเตอร์ที่อยู่ในโซนของเครื่องแม่ข่ายจะไม่สามารถสร้างการเชื่อมต่อไปยังโซนของเครื่องลูกข่ายได้ เพราะไม่มีการกำหนดกฎการแอคเซสเพื่อทำการอนุญาตไว้ รูปที่ 4.2 แสดงขั้นตอนในการทำงานของไฟร์วอลล์แบบรักษาสถานะ



รูปที่ 4.2 แสดงขั้นตอนการทำงานของไฟร์วอลล์แบบรักษาสถานะ

เมื่อมีแพ็คเก็ตถูกส่งมาถึงไฟร์วอลล์แบบรักษาสถานะ ลำดับแรกไฟร์วอลล์แบบรักษาสถานะจะทำการตรวจสอบว่าเป็น SYN แพ็คเก็ตของโปรโตคอลทีซีพีหรือเปล่า ถ้าใช้มันจะทำการตรวจสอบในตารางบันทึกกฎการแอคเซสว่ามีกฎการแอคเซสสำหรับอนุญาตให้การเชื่อมต่อนี้สามารถเกิดขึ้นได้หรือไม่ ถ้าไม่มีกฎการแอคเซสอนุญาตไว้มันจะทำการทิ้ง SYN แพ็คเก็ตนี้ไป แต่ถ้ามีกฎการแอคเซสอนุญาตไว้มันจะทำการบันทึกข้อมูลของการเชื่อมต่อนี้ไว้ในตารางสถานะ เพื่อไว้ใช้สำหรับตรวจสอบกับแพ็คเก็ตต่อ ๆ ไปของการเชื่อมต่อนี้ แพ็คเก็ตของโปรโตคอลทีซีพีที่ไม่ใช้ SYN แพ็คเก็ตจะถูกนำมาตรวจสอบกับข้อมูลที่ถูกเก็บไว้ในตารางสถานะเท่านั้น ถ้าเกิดไม่พบข้อมูลของการเชื่อมต่อของแพ็คเก็ตนั้นอยู่ในตารางสถานะแล้วแพ็คเก็ตนั้นจะถูกทิ้งไป แต่ถ้าพบข้อมูลในตารางสถานะแพ็คเก็ตนั้นจะสามารถผ่านไฟร์วอลล์แบบรักษาสถานะไปได้เลยโดยไม่ต้องถูกนำไปตรวจสอบกับกฎการแอคเซสอีก สำหรับการเชื่อมต่อที่ไม่ได้ใช้โปรโตคอลทีซีพี เช่น การเชื่อมต่อที่ใช้โปรโตคอลยูดีพี จะเป็นการเชื่อมต่อที่ไม่ต้องมีการสร้างการเชื่อมต่อก่อนการส่งข้อมูลเหมือนโปรโตคอลทีซีพี การเชื่อมต่อที่ใช้โปรโตคอลยูดีพีจึงไม่มี SYN แพ็คเก็ต ดังนั้นแพ็คเก็ตของการเชื่อมต่อประเภทนี้ จะต้องถูกนำไปตรวจสอบกับกฎการแอคเซสที่อยู่ในตารางบันทึกกฎการแอคเซสทุกครั้ง ถ้าเกิดไม่พบข้อมูลของการเชื่อมต่อของแพ็คเก็ตนั้นอยู่ในตารางสถานะ เมื่อมีกฎการแอคเซสอนุญาตไว้ไฟร์วอลล์แบบรักษาสถานะก็จะทำการบันทึกข้อมูลของ

การเชื่อมต่อนั้นลงในตารางสถานะ เพื่อไว้ใช้กับแพ็คเก็ตต่อ ๆ ไปของการเชื่อมต่อนั้นเช่นเดียวกับการเชื่อมต่อของโปรโตคอลทีซีพี

เนื่องจากการเชื่อมต่อที่ไม่ใช้โปรโตคอลทีซีพีจะไม่มี SYN แพ็คเก็ต ดังนั้นไฟร์วอลล์แบบรักษาสถานะจะไม่สามารถแยกแยะได้ว่าแพ็คเก็ตใดเป็นแพ็คเก็ตแรกของการเชื่อมต่อ ไฟร์วอลล์แบบรักษาสถานะจึงต้องถือเอาว่าแพ็คเก็ตที่ยังไม่มีข้อมูลของการเชื่อมต่อของมันอยู่ในตารางสถานะแล้วจะเป็นแพ็คเก็ตแรกของการเชื่อมต่อ จึงต้องนำแพ็คเก็ตนั้นไปตรวจสอบกับกฎการแอคเซสที่อยู่ในตารางบันทึกกฎการแอคเซสด้วย สำหรับแพ็คเก็ตที่มีข้อมูลของการเชื่อมต่อของมันอยู่ในตารางสถานะแล้ว ไฟร์วอลล์แบบรักษาสถานะจะอนุญาตให้แพ็คเก็ตนั้นผ่านไปได้โดยไม่ต้องตรวจสอบกฎการแอคเซสอีกเช่นเดียวกับโปรโตคอลทีซีพี

4.3 กฎการแอคเซส (Access Rule)

กฎการแอคเซส คือ กฎที่ไฟร์วอลล์ใช้ในการพิจารณาทราฟฟิคที่จะผ่านไฟร์วอลล์ไปยังโซนต่าง ๆ กฎการแอคเซสเป็นหัวใจสำคัญที่สุดของไฟร์วอลล์ทั้งในด้านความปลอดภัย และในด้านความสามารถในการใช้งานระบบเครือข่าย กฎการแอคเซสเป็นหัวใจสำคัญในการทำให้ระบบเครือข่ายทั้งหมดนั้นปลอดภัย โดยที่การกำหนดกฎการแอคเซสนั้นจะต้องพิจารณาถึงนโยบายสภาพการใช้งานและองค์ประกอบต่าง ๆ ในแต่ละระบบเครือข่ายเป็นสำคัญ กฎการแอคเซสที่ดีนั้นจะต้องสามารถทำให้ระบบเครือข่ายทั้งหมดปลอดภัยและสามารถใช้งานได้ดี ในขณะเดียวกันก็ต้องป้องกันปัญหาความปลอดภัยอื่นที่อาจจะเกิดขึ้นในอนาคตได้ด้วย

4.4 องค์ประกอบของกฎการแอคเซส

กฎการแอคเซสโดยทั่วไปจะประกอบด้วยองค์ประกอบที่ใช้ในการพิจารณาดังนี้

4.4.1 ต้นทาง (Source)

ต้นทาง หมายถึง ต้นทางของการเชื่อมต่อ อันจะแสดงให้เห็นว่าข้อมูลที่กำลังจะผ่านไฟร์วอลล์นั้นมีต้นกำเนิดมาจากที่ใด ซึ่งจะทำให้สามารถคาดหมายได้ว่าผู้ที่ใช้งานโฮสต์นั้นเป็นใคร มีสิทธิ์ในการใช้งานที่อนุญาตมากน้อยเพียงใด ด้วยการพิจารณาเพียงข้อมูลอิเล็กทรอนิคส์ที่อยู่บนระบบเครือข่ายนั้นเราไม่สามารถที่จะทราบตัวผู้ใช้ที่แท้จริงได้ สิ่งที่เป็นตัวแทนของผู้ใช้ก็คือโฮสต์นั่นเอง ข้อมูลฟิลด์นี้จะได้มาจากการอ่านค่าไอพีแอดเดรสต้นทาง (Source IP Address) ในไอพีแพ็คเก็ต ซึ่งในการกำหนดค่าต้นทางบนไฟร์วอลล์นั้นสามารถกำหนดได้หลายประเภทเช่น

1. โฮสต์เดียว (Individual Host) คือโฮสต์ใดโฮสต์หนึ่งเพียงโฮสต์เดียว โดยการระบุจะระบุไอพีแอดเดรสเฉพาะเจาะจง เช่น 200.1.1.100 หรือ 200.1.1.101 เป็นต้น หรือระบุโดยใช้ชื่อ

ของโฮสต์นั้น เช่น www.kmitl.ac.th เป็นต้น แต่โดยทั่วไปมักจะไม่นิยมระบุต้นทาง ด้วยวิธีนี้เพราะต้นทางมักจะมีจำนวนมาก โดยเฉพาะต้นทางซึ่งมีที่มาจากอินเตอร์เน็ตจะมีเป็นหลายพันหรือล้านโฮสต์ การระบุต้นทางด้วยวิธีนี้จะทำให้กฎการแอคเซสที่ได้มีจำนวนมาก และส่งผลให้ประสิทธิภาพในการทำงานของไฟร์วอลล์ต่ำลง ควรใช้เฉพาะกับโฮสต์ที่เฉพาะเจาะจงจริง ๆ เท่านั้น

2. กลุ่มของโฮสต์ (Group of Hosts) คือกลุ่มของโฮสต์หลาย ๆ ตัวมาจัดกลุ่มรวมกันเพื่อให้ง่ายต่อการกำหนดกฎการแอคเซส แทนที่จะกำหนดแต่ละกฎโดยอ้างถึงโฮสต์แต่ละตัว หากโฮสต์เหล่านั้นสามารถใช้กฎร่วมกันได้ ก็ให้นำโฮสต์เหล่านั้นรวมกลุ่มเข้าด้วยกันเสีย การกำหนดกฎการแอคเซสจะได้ง่ายขึ้นและไม่ต้องกำหนดกฎจำนวนมาก
3. เครือข่าย (Network) เป็นการระบุโฮสต์โดยการระบุทั้งเครือข่าย เพื่อให้ครอบคลุมโฮสต์ทั้งหมดที่อยู่ในเครือข่ายนั้น เช่น เครือข่ายภายใน (Internal Network), ดีเอ็มซี (DMZ) หรือเครือข่ายภายนอก (External network) หากต้องการระบุถึงโฮสต์ทั้งหมดจะระบุว่า "ANY"

4.4.2 ปลายทาง (Destination)

ปลายทาง หมายถึง ปลายทางของการเชื่อมต่อ อันจะแสดงให้เห็นว่าข้อมูลที่กำลังจะผ่านไฟร์วอลล์นั้นมีเป้าหมายสุดท้ายปลายทางอยู่ที่ใด ซึ่งจะทำให้สามารถคาดการณ์ได้ว่าเจ้าของข้อมูลชุดนั้นต้องการสื่อสารกับใคร วิธีการระบุลงในกฎการแอคเซสก็จะเป็นเช่นเดียวกับการระบุต้นทาง คือมีทั้งโฮสต์เดียว , กลุ่มของโฮสต์ และเครือข่าย ซึ่งไฟร์วอลล์จะตรวจสอบปลายทางได้จากการอ่านข้อมูลจากฟิลด์ ไอพีแอดเดรสปลายทาง (Destination IP Address) ของแพ็คเก็ต

ความยากง่ายในการกำหนดต้นทาง และปลายทางอาจจะแตกต่างกันไปในแต่ละผู้ผลิต แต่การกำหนดทั้ง 3 ประเภทข้างต้นเป็นวิธีการพื้นฐานที่จะต้องมีอยู่กับไฟร์วอลล์เสมอ วัตถุประสงค์ของการกำหนดต้นทาง และปลายทางคือ เพื่อให้ตัวไฟร์วอลล์สามารถเข้าใจและรู้จักกลุ่มของผู้ใช้ที่จะต้องควบคุม โดยทำการพิจารณาจากต้นทางและปลายทางของแพ็คเก็ตนั่นเอง ไฟร์วอลล์ที่มีวิธีการกำหนดต้นทาง และปลายทางที่กระชับและครอบคลุมโฮสต์จำนวนมาก จะช่วยทำให้การกำหนดกฎการแอคเซสทำได้ง่ายขึ้น

4.4.3 เซอร์วิส (Service)

เซอร์วิส หมายถึง บริการที่การเชื่อมต่อนั้นกำลังใช้งานอยู่ การรู้จักต้นทาง และปลายทาง จะทำให้ไฟร์วอลล์รู้ว่า "ใคร" กำลังสื่อสารกับ "ใคร" ส่วนของเซอร์วิสนั้นจะบอกให้ทราบว่าการเชื่อมต่อนั้นกำลัง "ทำอะไร" โดยการอ่านข้อมูลจากฟิลด์ "พอร์ตต้นทาง (Source Port) หรือพอร์ตปลายทาง (Destination Port)" ของแพ็คเก็ต ซึ่งขึ้นอยู่กับทิศทางของการเชื่อมต่อในขณะนั้น พอร์ตนั้นจะเป็นตัวที่บ่งบอกได้เป็นอย่างดีว่า ระหว่างโฮสต์ต้นทางและปลายทางนั้นกำลังใช้

แอพพลิเคชั่นอะไรในการเชื่อมต่อกัน และเราจึงสามารถสันนิษฐานได้ว่าผู้ใช้กำลัง "ทำอะไร" จากการดูแอพพลิเคชั่นที่ใช้งานนั่นเอง

ถึงแม้ว่าข้อมูลของเซอร์วิส โดยส่วนใหญ่แล้วก็คือข้อมูลที่ได้มาจากการอ่านค่าพอร์ต (ซึ่งอาจจะเป็นพอร์ตของโปรโตคอลทีซีพี หรือยูดีพีขึ้นอยู่กับแอพพลิเคชั่นที่ใช้งาน) แต่ก็มิได้หมายความว่าเซอร์วิส คือ พอร์ตเสมอไป และพอร์ตกับเซอร์วิส อาจจะไม่ได้หมายถึงสิ่งเดียวกันด้วยเหตุผลดังนี้คือ

1. การพิจารณาว่าแพ็คเก็ตนั้นใช้บริการอะไรอยู่ไม่สามารถพิจารณาได้จากข้อมูลในฟิลด์ของพอร์ตได้เพียงลำพัง จะต้องพิจารณาถึงทิศทางของการเชื่อมต่อด้วย เนื่องจากพอร์ตที่ใช้งานนั้นมีทั้งไคลเอ็นต์พอร์ตและเซิร์ฟเวอร์พอร์ตที่ปรากฏอยู่บนแพ็คเก็ต ซึ่งเราจะสามารถทราบได้ว่าการเชื่อมต่อนั้นใช้บริการอะไรก็จากการพิจารณาจากเซิร์ฟเวอร์พอร์ตเป็นหลัก
2. แอพพลิเคชั่นบางชนิดมีการใช้งานพอร์ตมากกว่า 1 พอร์ต เช่น เอฟทีพี ซึ่งใช้พอร์ต 21 เป็นพอร์ตหลักของคำสั่ง แต่ก็ใช้พอร์ต 20 สำหรับรับและส่งข้อมูล ดังนั้นการระบุว่าเซอร์วิสคือ เอฟทีพี จึงไม่ได้มีผลเช่นเดียวกับการระบุพอร์ต 21 จะเห็นได้ว่าเซอร์วิสจะครอบคลุมการทำงานมากกว่าพอร์ต หรืออาจเรียกได้ว่าเป็นคำนิยามของระดับชั้นที่สูงกว่าพอร์ต
3. สำหรับการสื่อสารของโปรโตคอลที่ไม่มีสถานะ (state) การรับส่งข้อมูล ดังเช่นโปรโตคอลยูดีพีนั้น เป็นการยากที่จะระบุว่าพอร์ตใดคือเซิร์ฟเวอร์พอร์ตอันหมายถึงเซอร์วิสหรือพอร์ตใดเป็นไคลเอ็นต์พอร์ต ซึ่งต่างจากโปรโตคอลทีซีพีที่สามารถตรวจสอบได้จาก TCP Flag ดังนั้นไฟร์วอลล์จึงต้องมีวิธีที่จะทำให้รู้ได้ว่าแพ็คเก็ตที่กำลังจะผ่านไปนั้นพอร์ตใดกันแน่ที่เป็นเซิร์ฟเวอร์พอร์ต โดยส่วนใหญ่จึงต้องมีการจัดเก็บตารางเซสชั่นของการเชื่อมต่อในขณะนั้นไว้เพื่อจะได้ทราบได้ว่าเซสชั่นเริ่มขึ้นเมื่อใด และโฮสต์ใดเป็นเซิร์ฟเวอร์และโฮสต์ใดเป็นไคลเอ็นต์ ดังนั้นจึงเห็นได้ว่ากระบวนการที่ทำให้ทราบเซอร์วิสที่ใช้งานนั้น มิใช่เพียงการดูที่ข้อมูลในฟิลด์ของพอร์ตเท่านั้น
4. ถึงแม้ว่าโปรโตคอลหลักที่ใช้โดยแอพพลิเคชั่น นั้นคือ โปรโตคอลทีซีพีและยูดีพี แต่เซอร์วิสที่ให้บริการมิได้มีใช้เฉพาะโปรโตคอลทีซีพีและยูดีพีเท่านั้น มีเซอร์วิสบางประเภทที่ใช้โปรโตคอลไอซีเอ็มพีในการทำงาน ซึ่งแน่นอนว่าไอซีเอ็มพีนั้นย่อมไม่มีพอร์ตที่ระบุแอพพลิเคชั่น เช่น ที่มีอยู่ในโปรโตคอลทีซีพีและยูดีพี การกำหนดเซอร์วิสจึงต้องกำหนดตามลักษณะการเชื่อมต่อของโปรโตคอลนั้นจริง ๆ ดังนั้นจึงอาจจะพบเซอร์วิสบางประเภท เช่น Ping, Trace Route ซึ่งใช้โปรโตคอลไอซีเอ็มพีในการทำงาน ไม่ได้ใช้ข้อมูลของพอร์ตมาเกี่ยวข้องแม้แต่น้อย

สำหรับโปรโตคอลทีซีพี และยูดีพีนั้นเซอร์วิสก็เป็นข้อมูลที่มีรากฐานมาจากข้อมูลในฟิลด์พอร์ตที่ผ่านกระบวนการวิเคราะห์มาแล้วนั่นเอง เมื่อประกอบกับปัจจัยอื่น ๆ แล้วจึงค่อยสรุปได้ว่าเป็นเซอร์วิสอะไรประเภทไหน การกำหนดกฎการแอคเซสของไฟร์วอลล์โดยการระบุเซอร์วิสนั้น

จะเป็นการง่ายต่อการใช้งานมากขึ้น เช่นจะทราบดีว่ามีการเรียกดูเว็บเพจโดยผ่านบราวเซอร์ แต่อาจจะไม่ทราบว่าบราวเซอร์ใช้พอร์ต 80 เป็นต้น ดังนั้นการกำหนดกฎการแอคเซสด้วยสิ่งที่ผู้ใช้เข้าใจจึงสะดวก ทั้งในแง่ของการใช้งานและการตรวจสอบกฎการแอคเซสที่กำหนดไว้ในภายหลัง

เซอร์วิสที่กำหนดมากับไฟร์วอลล์โดยดีฟอลต์นั้นจะเป็นเซอร์วิสพื้นฐาน และเซอร์วิสที่เป็นที่รู้จักและใช้งานกันทั่วไป ทั้งนี้เพื่อให้ผู้ใช้ไฟร์วอลล์สามารถใช้งานได้ง่ายที่สุด ผู้ใช้อาจจะไม่จำเป็นต้องเข้าใจและศึกษาการเชื่อมต่อของแต่ละแอพพลิเคชั่นมากนัก ขอเพียงให้รู้จักชื่อก็เพียงพอที่จะนำมากำหนดกฎการแอคเซสได้ อย่างไรก็ตามเซอร์วิสใดที่ไม่ได้กำหนดเป็นดีฟอลต์มาโดยไฟร์วอลล์ ผู้ใช้ก็สามารถกำหนดขึ้นมาใหม่ได้ โดยการกำหนดโปรโตคอลและหมายเลขพอร์ตที่เซอร์วิสนั้นใช้งาน ดังนั้นถึงแม้ว่ามีแอพพลิเคชั่นใดเกิดขึ้นมาใหม่ก็สามารถจะกำหนดให้ไฟร์วอลล์ควบคุมการเชื่อมต่อของแอพพลิเคชั่นเหล่านั้นได้ เพราะไม่ว่าแอพพลิเคชั่นใดที่ทำงานบน "ทีซีพี/ไอพี" ก็ย่อมมีรากฐานการทำงานที่ใกล้เคียงกัน สิ่งที่ผู้ดูแลไฟร์วอลล์ควรจะเรียนรู้คือหลักการสำคัญที่ไฟร์วอลล์ใช้ในการควบคุมการเชื่อมต่อ ซึ่งแม้ว่าไฟร์วอลล์ของผู้ผลิตบางรายอาจจะไม่มีเครื่องมือที่ช่วยในการอำนวยความสะดวกมาให้ด้วยมากนัก แต่ก็ด้วยพื้นฐานที่ดีจะทำให้ยังคงสามารถกำหนดกฎการแอคเซสเพื่อควบคุมการเชื่อมต่อได้อย่างมีประสิทธิภาพ

4.4.4 แอคชั่น (Action)

แอคชั่น คือ สิ่งที่ไฟร์วอลล์จะกระทำกับแพ็คเก็ตที่มีข้อมูลในส่วนหัวตรงกับเงื่อนไขในส่วนต้นทาง (Source), ปลายทาง (Destination) และเซอร์วิส (Service) ของกฎการแอคเซสที่กำหนดไว้ใน ไฟร์วอลล์แอคชั่นที่สามารถกำหนดให้ไฟร์วอลล์ดำเนินการได้นั้นมีดังนี้

1. อนุญาต (Accept) หมายถึง อนุญาตให้แพ็คเก็ตที่ตรงตามเงื่อนไขนั้นสามารถผ่านไฟร์วอลล์ไปยังปลายทางได้ตามปกติเหมือนไม่มีอะไรเกิดขึ้น
2. ไม่อนุญาตแบบตอบกลับ (Reject) หมายถึง การปฏิเสธไม่ให้แพ็คเก็ตนั้นผ่านไฟร์วอลล์ไปยังปลายทางได้ โดยจะส่งสัญญาณปฏิเสธกลับไปยังผู้ส่งต้นทาง เพื่อให้ทราบว่าแพ็คเก็ตที่ส่งมานั้นไม่ได้รับอนุญาตให้ผ่านเข้าไป แอพพลิเคชั่นเมื่อได้รับแพ็คเก็ตตอบกลับเหล่านั้นก็จะหยุดการสื่อสารในทันที พร้อมกับอาจจะแจ้งให้ผู้ใช้ทราบว่าไม่สามารถติดต่อโฮสต์ปลายทางได้ด้วยสาเหตุใด โดยการนำแพ็คเก็ตที่ตอบกลับมาวิเคราะห์ก็จะสามารถทราบได้ทันทีว่าไม่สามารถติดต่อกับปลายทางด้วยสาเหตุใด
3. ไม่อนุญาตแบบไม่ตอบกลับ (Drop) หมายถึง การปฏิเสธไม่ให้แพ็คเก็ตนั้นผ่านไฟร์วอลล์ไปยังปลายทางเช่นเดียวกับแบบไม่อนุญาตแบบตอบกลับ แต่ในการไม่อนุญาตแบบไม่ตอบกลับ นั้นไฟร์วอลล์จะไม่ส่งสัญญาณใด ๆ กลับไปยังต้นทางของแพ็คเก็ต กล่าวคือไฟร์วอลล์จะทิ้ง แพ็คเก็ตที่เข้ามานั้นไปเฉย ๆ ซึ่งเป็นสิ่งที่จะไม่มีโอกาสเกิดขึ้นได้ในการสื่อสารตามปกติของ ทีซีพี/ไอพี ที่มีกฎว่าหากไม่สามารถติดต่อโฮสต์ปลายทางได้จะต้องมี

การแจ้งความผิดพลาดอย่างใดอย่างหนึ่งกลับมา ไม่ว่าจะเป็น Destination Unreachable, Port Unreachable หรือ Reset การไม่ตอบอะไรกลับไปนั้นอาจทำให้แอพพลิเคชั่นที่ส่งแพ็คเก็ตมาทำงานผิดปกติได้ เพราะหากเป็นกรณีของแบบไม่อนุญาตแบบตอบกลับ หรือการสื่อสารตามปกติ เมื่อโฮสต์ได้รับการแจ้งกลับมาว่าไม่สามารถติดต่อโฮสต์ปลายทางอีกฝั่งหนึ่งได้นั้น แอพพลิเคชั่นก็จะรู้ทันทีว่าไม่ต้องพยายามติดต่ออีกต่อไปและยุติการทำงานได้ทันที แต่สำหรับในกรณีการไม่อนุญาตแบบไม่ตอบกลับนั้น จะไม่มีสัญญาณใด ๆ ตอบกลับจากไฟร์วอลล์ ซึ่งจะทำให้แอพพลิเคชั่นที่ไม่ได้มีการกำหนดเวลาสิ้นสุดของการที่จะคอยการตอบรับ จะต้องรอไปเรื่อย ๆ โดยไม่มีการรับการตอบรับใด ๆ เลย ดังนั้น การไม่อนุญาตแบบไม่ตอบกลับ จึงอาจส่งผลกระทบต่อแอพพลิเคชั่นของผู้ใช้ได้

ตัวอย่างกฎการแอคเซสที่บังคับใช้บนไฟร์วอลล์ที่ควบคุมให้ (1) ผู้ใช้ภายในสามารถใช้บริการเอชทีทีพีเพื่อเรียกดูเว็บเพจ และใช้บริการป็อบและเอสเอ็มทีพี เพื่อรับส่งอีเมลล์กับโฮสต์ใดก็ได้ (2) ห้ามไม่ให้ใช้บริการอื่น ๆ นอกจากนี้ (3) ไม่ให้โฮสต์จากที่อื่นที่อยู่ภายนอกสามารถติดต่อเพื่อขอใช้บริการจากโฮสต์ภายใน

ตารางที่ 4.4 ตัวอย่างการกำหนดกฎการแอคเซส

No	Source	Destination	Service	Action	Track	Time
1.	Internal	Any	HTTP,POP,SMTP	Accept	None	Any
2.	Internal	Any	Any	Reject	None	Any
3.	Any	Internal	Any	Drop	None	Any

แอคชั่นต่อไปนี้อาจจะไม่ได้เป็นแอคชั่นมาตรฐานสำหรับไฟร์วอลล์ทุกยี่ห้อ ไฟร์วอลล์บางยี่ห้ออาจไม่มีแอคชั่นครบถ้วนเหมือนดังที่กล่าวมา

บทที่ 5
วิธีที่นำเสนอ

อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่กล่าวผ่านมาในบทที่ 3 นั้น จะไม่ทำการย้ายโหลดจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่ง ทั้งในกรณีที่พบว่ามีไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งไม่สามารถทำงานต่อ และในกรณีที่มีไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งมีโหลดมากเกินไป

ในกรณีที่มีไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งไม่สามารถทำงานต่อได้นั้น อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์โดยทั่วไป จะทำแค่เพียงตัดไฟร์วอลล์แบบรักษาสถานะตัวนั้นออกจากระบบ โดยที่มันจะไม่ทำการกระจายการเชื่อมต่อใหม่ไปให้ไฟร์วอลล์แบบรักษาสถานะตัวนั้นอีก การเชื่อมต่อที่เคยถูกส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวที่หยุดทำงานนั้นก็จะสิ้นสุดตามไฟร์วอลล์ไป การที่อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ไม่ทำการย้ายการเชื่อมต่อที่เคยถูกส่งผ่านไฟร์วอลล์ตัวที่หยุดทำงานไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวอื่น ก็เพราะไฟร์วอลล์แบบรักษาสถานะตัวอื่นจะไม่อนุญาตให้ทราฟฟิคของการเชื่อมต่อที่จะถูกย้ายมานั้นผ่าน

ในกรณีที่ไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งมีโหลดมากเกินไป อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์โดยทั่วไปก็ไม่สามารถทำการย้ายการเชื่อมต่อ จากที่ส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวที่มีโหลดมากให้ไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวอื่น ๆ ที่มีโหลดในขณะนั้นน้อยกว่าได้เช่นเดียวกัน ทั้งที่ไฟร์วอลล์แบบรักษาสถานะตัวอื่นสามารถประมวลผลได้เร็วกว่าเนื่องจากการที่มีโหลดน้อยกว่ามาก จะเห็นว่าการทำงานของอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์โดยทั่วไปจะไม่ค่อยยืดหยุ่น เมื่อทำการตัดสินใจกระจายการเชื่อมต่อไปให้ไฟร์วอลล์แบบรักษาสถานะตัวใดแล้ว ก็จะส่งทราฟฟิคของการเชื่อมต่อนั้นผ่านไฟร์วอลล์แบบรักษาสถานะตัวเดิมเสมอ จะไม่สามารถปรับโหลดให้สมดุลย์ภายหลังได้

5.1 สาเหตุที่อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ไม่ทำการย้ายหรือปรับโหลด

สาเหตุที่อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ไม่ทำการย้ายโหลดจากไฟร์วอลล์หนึ่งไปยังไฟร์วอลล์อีกตัวหนึ่งหลังจากเกิดเหตุการณ์ทั้งสองเหตุการณ์ที่ได้กล่าวมาก็เพราะ เมื่อทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งแล้ว ถ้าเกิดไฟร์วอลล์แบบรักษาสถานะทั้งสองตัวมาจากต่างผู้ผลิตกัน ซึ่งจะไม่สามารถทำการแลกเปลี่ยนข้อมูลในตารางสถานะระหว่างกันได้ หรือเป็นไฟร์วอลล์แบบรักษาสถานะที่มาจากผู้ผลิตเดียวกันแต่ไม่สามารถทำการแลกเปลี่ยนข้อมูลในตารางสถานะระหว่างกัน เนื่องจากเป็นไฟร์วอลล์แบบรักษาสถานะคนละรุ่น หรือเป็นไฟร์วอลล์แบบรักษาสถานะรุ่นเดียวกันแต่ไม่รองรับ

การแลกเปลี่ยนข้อมูลในตารางสถานะระหว่างกันแล้ว ไฟร์วอลล์แบบรักษาสถานะอีกตัวจะไม่อนุญาตให้ทราฟฟิคของการเชื่อมต่อที่กำลังจะย้ายมาผ่าน

5.2 สาเหตุที่ไฟร์วอลล์แบบรักษาสถานะไม่ยอมให้การเชื่อมต่อที่ย้ายมาผ่าน

ดังที่ได้กล่าวไปในบทที่ 4 อุปกรณ์ไฟร์วอลล์แบบรักษาสถานะจะทำการเก็บข้อมูลของแต่ละการเชื่อมต่อที่ถูกสร้างผ่านตัวมัน ในขณะที่มีการสร้างการเชื่อมต่อจากเครื่องไคลเอนต์ไปยังเครื่องเซิร์ฟเวอร์ผ่านตัวมัน โดยไฟร์วอลล์แบบรักษาสถานะจะเก็บเฉพาะข้อมูลของการเชื่อมต่อที่ได้รับอนุญาตให้ผ่านไฟร์วอลล์ไว้ในตารางสถานะเท่านั้น ดังนั้นก็หมายความว่าการเชื่อมต่อที่ไม่มีข้อมูลอยู่ในตารางสถานะของไฟร์วอลล์แบบรักษาสถานะก็คือ การเชื่อมต่อที่ไม่ได้รับอนุญาตให้ผ่านไฟร์วอลล์แบบรักษาสถานะนั้นเอง ดังนั้นไฟร์วอลล์แบบรักษาสถานะจะไม่อนุญาตให้ทราฟฟิคของการเชื่อมต่อเหล่านั้นผ่านได้เลย

5.3 สาเหตุที่ไฟร์วอลล์มีโหลดแตกต่างกัน

อาจมีคำถามว่าอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ได้พยายามทำการกระจายหรือแบ่งโหลดให้แต่ละไฟร์วอลล์แล้ว ทำไมโหลดระหว่างไฟร์วอลล์ถึงยังแตกต่างกันได้ สาเหตุที่โหลดระหว่างไฟร์วอลล์เกิดความแตกต่างกัน หลังจากที่ได้มีการกระจายโหลดหรือแบ่งโหลดอย่างเหมาะสมตามอัลกอริทึมที่ผู้ดูแลระบบเลือกใช้แล้วนั้น เกิดจากการที่การเชื่อมต่อแต่ละการเชื่อมต่อใช้ระยะเวลาในการเชื่อมต่อไม่เท่ากัน บางการเชื่อมต่อใช้ระยะเวลาในการเชื่อมต่อน้อย บางการเชื่อมต่อใช้ระยะเวลาในการเชื่อมต่อมาก ถ้าเกิดเหตุการณ์ที่บังเอิญว่าไฟร์วอลล์ตัวที่หนึ่งได้รับเฉพาะการเชื่อมต่อที่ใช้ระยะเวลาในการเชื่อมต่อมากเป็นส่วนใหญ่ และไฟร์วอลล์ตัวที่สองบังเอิญได้รับเฉพาะการเชื่อมต่อที่ใช้ระยะเวลาในการเชื่อมต่อน้อยเป็นใหญ่ พอเวลาผ่านไปสักช่วงเวลาหนึ่ง การเชื่อมต่อที่ผ่านไฟร์วอลล์ตัวที่สองก็จะค่อย ๆ สิ้นสุดลงจนเกือบหมด จะเหลือเฉพาะการเชื่อมต่อที่ใช้ระยะเวลาในการเชื่อมต่อมากไม่กี่การเชื่อมต่อเท่านั้น แต่การเชื่อมต่อที่ผ่านไฟร์วอลล์ตัวแรกยังคงเหลืออยู่เป็นจำนวนมาก จะมีเฉพาะการเชื่อมต่อที่ใช้ระยะเวลาในการเชื่อมต่อน้อยสิ้นสุดลงไปบ้างเท่านั้น ซึ่งจะเห็นว่าเมื่อมีเหตุการณ์เช่นนี้ไฟร์วอลล์แบบรักษาสถานะตัวที่หนึ่งจะมีโหลดแตกต่างจากไฟร์วอลล์แบบรักษาสถานะตัวที่สองมาก

เหตุการณ์ที่ไฟร์วอลล์มีโหลดแตกต่างกันมากสามารถเกิดขึ้นได้เสมอ ยิ่งเป็นระบบที่ใหญ่มีการเชื่อมต่อเป็นจำนวนมาก ยิ่งมีความเป็นได้สูงว่าจะเกิดเหตุการณ์เช่นนี้ได้บ่อย ๆ เหตุการณ์เช่นนี้สามารถที่จะเกิดขึ้นได้ แม้ว่าเราจะเลือกใช้อัลกอริทึมในการกระจายโหลดแบบใด หรือดีแค่ไหนก็ตาม

5.4 วิธีที่นำเสนอ

เพื่อทำการแก้ปัญหาที่กล่าวมาเราได้ทำการเสนอวิธีที่ช่วยให้สามารถทำการย้ายโหลดจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งได้ โดยวิธีที่นำเสนอ [29] จะทำการหลอกให้ไฟร์วอลล์แบบรักษาสถานะตัวที่เราจะทำการย้ายการเชื่อมต่อไป ให้ทำการบันทึกข้อมูลของการเชื่อมต่อที่กำลังจะถูกย้ายไปลงในตารางสถานะของตนเสียก่อน เมื่อไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลของการเชื่อมต่อลงในตารางสถานะแล้ว มันก็จะอนุญาตให้ทราฟฟิคของการเชื่อมต่อนั้นผ่านได้ หลังจากนั้นอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ก็สามารถส่งทราฟฟิคของการเชื่อมต่อที่ต้องการจะย้ายผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่ได้

ไฟร์วอลล์แบบรักษาสถานะจะทำการบันทึกข้อมูลของการเชื่อมต่อที่ใช้โปรโตคอลทีซีพีลงในตารางสถานะก็ต่อเมื่อ มันได้รับแพ็คเก็ตประเภท SYN ของการเชื่อมต่อที่ใช้โปรโตคอลทีซีพีที่มีกฎการแอคเซสอนุญาตให้การเชื่อมต่อนี้สามารถเกิดขึ้นได้ ซึ่งแพ็คเก็ตประเภท SYN นี้จะเป็นแพ็คเก็ตที่ใช้สำหรับร้องขอสร้างการเชื่อมต่อจากเครื่องไคลเอ็นต์ไปยังเครื่องเซิร์ฟเวอร์ เมื่อไฟร์วอลล์แบบรักษาสถานะได้รับแพ็คเก็ตประเภท SYN มันจะยังไม่ทำการบันทึกข้อมูลของการเชื่อมต่อนี้ลงในตารางสถานะโดยทันที มันจะทำการตรวจสอบเสียก่อนว่าจะอนุญาตให้การเชื่อมต่อนี้สามารถถูกสร้างขึ้นผ่านตัวมันได้หรือไม่ ถ้าอนุญาตให้ผ่านได้มันก็จะทำการเก็บข้อมูลของการเชื่อมต่อนี้ลงในตารางสถานะ ถ้าไม่อนุญาตให้ผ่านมันก็จะทำการทิ้งแพ็คเก็ต SYN ไป และจะไม่บันทึกข้อมูลใด ๆ ลงในตารางสถานะ ข้อมูลในตารางสถานะที่ถูกบันทึกลงไปนี้จะทำให้แพ็คเก็ตที่เหลือของการเชื่อมต่อนี้สามารถผ่านไฟร์วอลล์แบบรักษาสถานะได้ทั้งสองทาง โดยแพ็คเก็ตที่เหลือจะหมายถึง แพ็คเก็ตธรรมดาที่จะไม่ใช้แพ็คเก็ตประเภท SYN ที่ใช้ในการขอสร้างการเชื่อมต่อ

ถ้าเป็นแพ็คเก็ตของโปรโตคอลทีซีพี แต่ไม่ใช้แพ็คเก็ตประเภท SYN ไฟร์วอลล์แบบรักษาสถานะจะนำแพ็คเก็ตประเภทนี้ไปตรวจสอบกับข้อมูลที่อยู่ในตารางสถานะเท่านั้น จะไม่นำไปตรวจสอบกับกฎการแอคเซสที่อยู่ในตารางบันทึกกฎการแอคเซสอีก สรุปว่าถ้าแพ็คเก็ตที่ไม่ใช้แพ็คเก็ตประเภท SYN จะผ่านไฟร์วอลล์แบบรักษาสถานะได้นั้น ต้องมีข้อมูลของการเชื่อมต่อของแพ็คเก็ตนั้นอยู่ในตารางสถานะของไฟร์วอลล์แบบรักษาสถานะเสียก่อน ไฟร์วอลล์แบบรักษาสถานะถึงจะอนุญาตให้ผ่านได้

จากที่กล่าวมาจะเห็นว่าไฟร์วอลล์แบบรักษาสถานะจะทำการบันทึกข้อมูลของการเชื่อมต่อลงในตารางสถานะ เมื่อได้รับแพ็คเก็ตประเภท SYN ของการเชื่อมต่อที่มีกฎการแอคเซสสำหรับอนุญาตให้การเชื่อมต่อนั้นสามารถเกิดขึ้นได้ ดังนั้นถ้าเกิดเราจำลองแพ็คเก็ตประเภท SYN ของการเชื่อมต่อที่ต้องการจะย้าย แล้วนำไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวที่ต้องการจะย้ายการเชื่อมต่อไป จะทำให้ไฟร์วอลล์แบบรักษาสถานะตัวนั้นบันทึกข้อมูลของการเชื่อมต่อที่กำลังจะ

ทำการย้ายลงในตารางสถานะของตน และจะอนุญาตให้ทราฟฟิคที่เหลือของการเชื่อมต่อนั้นผ่านได้ทั้งสองทาง ดังนั้นเราก็จะสามารถส่งทราฟฟิคที่เหลือที่ไม่สามารถส่งผ่านไฟร์วอลล์ตัวที่หยุดทำงานให้ผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่ได้แล้ว

สำหรับการเชื่อมต่อที่ใช้โปรโตคอลยูดีพี ซึ่งเป็นการเชื่อมต่อที่ไม่ต้องมีการสร้างการเชื่อมต่อ จะดูเหมือนว่าไม่จำเป็นต้องสร้างแพ็คเก็ต เพื่อหลอกให้ไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลของการเชื่อมต่อลงในตารางสถานะ เพราะถ้าเป็นการเชื่อมต่อที่ใช้โปรโตคอลยูดีพีแล้ว เมื่อมีแพ็คเก็ตเข้ามายังไฟร์วอลล์แบบรักษาสถานะ และไฟร์วอลล์แบบรักษาสถานะไม่พบข้อมูลของการเชื่อมต่อของแพ็คเก็ตนี้ในตารางสถานะแล้ว ไฟร์วอลล์แบบรักษาสถานะจะทำการตรวจสอบต่อไปอีกว่ามีกฎการแอคเซสในตารางบันทึกกฎการแอคเซสอนุญาตให้แพ็คเก็ตของการเชื่อมต่อนี้ผ่านได้หรือไม่ ถ้ามีกฎอนุญาตไว้ไฟร์วอลล์แบบรักษาสถานะก็จะอนุญาตให้แพ็คเก็ตนี้ผ่านได้เช่นเดียวกัน พร้อมกับทำการบันทึกข้อมูลของการเชื่อมต่อนี้ลงในตารางสถานะ เพื่อไว้ใช้สำหรับตรวจสอบแพ็คเก็ตต่อไปของการเชื่อมต่อนี้ ซึ่งแพ็คเก็ตต่อไปของการเชื่อมต่อนี้ก็จะมีข้อมูลอยู่ในตารางสถานะเรียบร้อยแล้ว ไฟร์วอลล์แบบรักษาสถานะจึงไม่ต้องไปตรวจสอบกฎการแอคเซสในตารางบันทึกกฎการแอคเซสอีก จากที่กล่าวมานี้อาจจะดูเหมือนว่า ถ้ามีกฎการแอคเซสที่อนุญาตให้การเชื่อมต่อที่ใช้โปรโตคอลยูดีพีสามารถผ่านไฟร์วอลล์แบบรักษาสถานะได้ อยู่ในตารางบันทึกกฎการแอคเซสของไฟร์วอลล์แบบรักษาสถานะตัวที่เราจะทำการย้ายการเชื่อมต่อไปแล้ว อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ก็สามารถที่ย้ายการเชื่อมต่อไปยังไฟร์วอลล์แบบรักษาสถานะตัวใหม่ได้ทันที

แต่จริง ๆ แล้วยังต้องทำการสร้างแพ็คเก็ตจำลองสำหรับการเชื่อมต่อที่ใช้โปรโตคอลยูดีพี เพื่อใช้สำหรับหลอกไฟร์วอลล์แบบรักษาสถานะเช่นเดียวกันกับการเชื่อมต่อที่ใช้โปรโตคอลทีซีพี สาเหตุที่ต้องสร้างแพ็คเก็ตหลอกเช่นเดียวกับโปรโตคอลทีซีพี เพราะการกำหนดกฎการแอคเซสของไฟร์วอลล์แบบสถานะทั้งของโปรโตคอลยูดีพีและทีซีพี จะเป็นการกำหนดแบบทิศทางเดียวเหมือนกัน ดังนั้นเมื่อเป็นการกำหนดกฎการแอคเซสแบบทิศทางเดียว การเชื่อมต่อจะสามารถเกิดขึ้นได้ก็ต่อเมื่อผู้เริ่มการเชื่อมต่อเป็นเครื่องที่อยู่ในโซนที่อนุญาตให้เริ่มการเชื่อมต่อได้เท่านั้น ไฟร์วอลล์แบบรักษาสถานะถึงจะให้การเชื่อมต่อนี้เกิดขึ้นได้ และบันทึกข้อมูลของการเชื่อมต่อนี้ลงในตารางสถานะ

อาจจะมีข้อสังเกตว่าโปรแกรมประยุกต์ที่ใช้โปรโตคอลยูดีพีที่อยู่ทั้งสองฝั่ง น่าจะมีการส่งข้อมูลตอบรับการส่งข้อมูลระหว่างกันเป็นช่วง ๆ อยู่ตลอดเวลา ดังนั้นน่าที่จะมีทราฟฟิคจากทั้งสองฝั่งมายังไฟร์วอลล์อยู่ตลอดเวลาเช่นเดียวกัน ซึ่งถ้ามีทราฟฟิคที่เป็นของโปรโตคอลยูดีพีผ่านไฟร์วอลล์แบบรักษาสถานะทั้งสองทางยอมแน่นอนว่าต้องมีทิศทางหนึ่ง ซึ่งเป็นทิศทางเดียวกับที่ได้มีการกำหนดกฎการแอคเซสอนุญาตให้การเชื่อมต่อนี้ผ่านได้ เมื่อมีกฎการแอคเซสอนุญาตไว้ในทิศทางเดียวกับที่แพ็คเก็ตที่ใช้โปรโตคอลยูดีพีนี้ผ่าน ก็จะมีการบันทึกข้อมูลของการเชื่อมต่อนี้ลง

ในตารางสถานะแล้ว ทราฟฟิคของการเชื่อมต่อนี้ก็จะสามารถผ่านไฟร์วอลล์แบบรักษาสถานะได้ทั้งสองทาง ซึ่งดูเหมือนว่าเราไม่จำเป็นที่จะต้องสร้างแพ็คเก็ตหลอกแต่อย่างใด เพียงแต่ให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทำการย้ายแพ็คเก็ตของการเชื่อมต่อที่ใช้โปรโตคอลยูดีพีมาส่งผ่านไฟร์วอลล์แบบรักษาสถานะอีกตัวเท่านั้นก็น่าจะเพียงพอแล้ว

แต่อย่าลืมว่าการควบคุมการส่งข้อมูลของโปรแกรมประยุกต์จะทำโดยโปรแกรมประยุกต์เอง ดังนั้นก็มีความเป็นไปได้ว่าผู้เขียนโปรแกรมอาจจะไม่ต้องการให้มีการตอบรับกันระหว่างโปรแกรมประยุกต์ทั้งสองฝั่ง ต้องการให้โปรแกรมประยุกต์ฝั่งหนึ่งทำหน้าที่กระจายข้อมูลออกไป อีกฝั่งหนึ่งทำหน้าที่แค่เพียงรอรับข้อมูลเพียงอย่างเดียว โดยไม่ต้องมีการตอบรับใด ๆ ทั้งสิ้น โดยฝั่งรับข้อมูลทำแค่เพียงร้องขอรับข้อมูลในตอนแรกเท่านั้นหลังจากนั้นจะทำการรอรับข้อมูลเพียงอย่างเดียว

ซึ่งจะเห็นว่าถ้าเกิดการส่งข้อมูลประเภทยูดีพีโดยโปรแกรมประยุกต์ เป็นการส่งจากเซิร์ฟเวอร์ไปยังไคลเอ็นต์เกือบจะเพียงทิศทางด้านเดียวแล้ว เมื่อเกิดเหตุการณ์ที่ไฟร์วอลล์แบบรักษาสถานะตัวที่การเชื่อมต่อนี้ผ่านหยุดทำงาน อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะไม่สามารถทำการส่งทราฟฟิคของการเชื่อมต่อนี้ผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่ได้ เพราะไม่มีกฎการแอคเซสสำหรับอนุญาตให้ทราฟฟิคจากโซนของเซิร์ฟเวอร์สามารถผ่านไปยังโซนของไคลเอ็นต์ และไม่มีแพ็คเก็ตตอบรับจากไคลเอ็นต์ไปกระตุ้นให้ไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลของการเชื่อมต่อลงในตารางสถานะไฟร์วอลล์แบบรักษาสถานะก็จะไม่ยอมให้ทราฟฟิคที่ส่งมาจากเซิร์ฟเวอร์ผ่าน ดังนั้นเราจึงจำเป็นที่จะต้องสร้างแพ็คเก็ตจำลอง เพื่อทำการหลอกให้ไฟร์วอลล์แบบรักษาสถานะตัวที่เราจะทำการย้ายการเชื่อมต่อที่ทำงานในลักษณะนี้ไป ให้ทำการบันทึกข้อมูลของการเชื่อมต่อนี้ลงในตารางสถานะก่อนที่จะทำการย้ายทราฟฟิคที่เหลือของการเชื่อมต่อนี้ไปส่งผ่าน เมื่อทำการส่งแพ็คเก็ตจำลองผ่านไฟร์วอลล์แบบรักษาสถานะแล้ว ไฟร์วอลล์แบบรักษาสถานะก็จะบันทึกข้อมูลของการเชื่อมต่อนี้ลงในตารางสถานะ และหลังจากนั้นข้อมูลจากเซิร์ฟเวอร์ก็จะสามารถถูกส่งผ่านไฟร์วอลล์แบบรักษาสถานะไปยังไคลเอ็นต์ได้

ถึงแม้ว่าในกรณีที่โปรแกรมประยุกต์ทั้งสองฝั่งมีการตอบรับการส่งข้อมูลกันอยู่เป็นช่วง ๆ แต่ถ้าเป็นการตอบรับแบบนาน ๆ ตอบที่ เมื่อไฟร์วอลล์แบบรักษาสถานะตัวที่การเชื่อมต่อที่ทำงานในลักษณะนี้ผ่านหยุดทำงาน ก็อาจทำให้มีข้อมูลสูญหายเป็นจำนวนมาก หรือแม้กระทั่งอาจจะทำให้การเชื่อมต่อไม่สามารถดำเนินต่อไปจนจบก็เป็นได้

5.5 กระบวนการทำงานของวิธีที่เสนอ

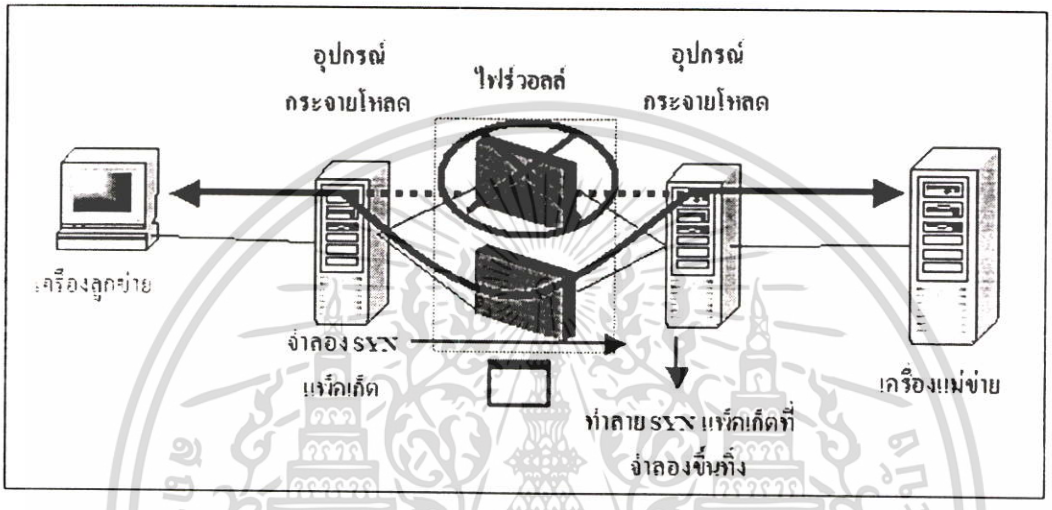
จากที่ได้กล่าวมาเราสามารถนำมาเขียนเป็นขั้นตอนของวิธีที่นำเสนอได้ดังต่อไปนี้ โดยเราจะทำการจำแนกเหตุการณ์ที่จะทำการย้ายการเชื่อมต่อเป็น 2 เหตุการณ์ คือ

1. ทำการย้ายการเชื่อมต่อเมื่อมีไฟร์วอลล์หยุดทำงาน

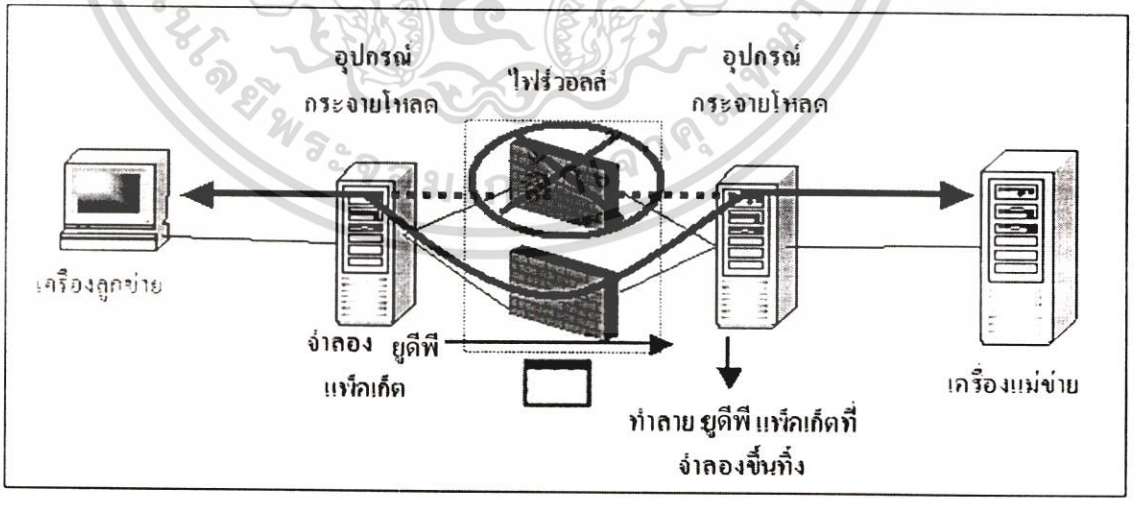
2. ทำการย้ายการเชื่อมต่อเมื่อไฟร์วอลล์มีโหลดต่างกันมาก

5.5.1 การย้ายการเชื่อมต่อเมื่อมีไฟร์วอลล์หยุดทำงาน

เมื่อมีไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งหยุดทำงาน เราจะทำการย้ายการเชื่อมต่อทั้งหมดที่ผ่านไฟร์วอลล์แบบรักษาสถานะตัวนั้น ไปยังไฟร์วอลล์แบบรักษาสถานะตัวอื่น โดยขั้นตอนสำหรับการย้ายการเชื่อมต่อมีดังต่อไปนี้ (ดูรูปที่ 5.1 และ 5.2 ประกอบ)



รูปที่ 5.1 แสดงขั้นตอนที่ทำให้สามารถส่งทราฟฟิคของการเชื่อมต่อที่ใช้โปรโตคอลทีซีพีผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่



รูปที่ 5.2 แสดงขั้นตอนที่ทำให้สามารถส่งทราฟฟิคของการเชื่อมต่อที่ใช้โปรโตคอลยูดีพีผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่

1. เมื่ออุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ตรวจพบว่ามีไฟร์วอลล์แบบรักษาสถานะตัวใดหยุดทำงาน อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั้งสองตัว จะทำการตรวจสอบว่ามันได้ทำการกระจายหรือแบ่งการเชื่อมต่อใดบ้างไปให้ไฟร์วอลล์แบบรักษาสถานะตัวนั้น

2. หลังจากนั้นอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั้งสองตัวจะทำการจำลองแพ็คเก็ตพิเศษ สำหรับแต่ละการเชื่อมต่อที่ตรวจพบจากขั้นตอนที่ผ่านมา เพื่อใช้สำหรับหลอกให้ไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลของการเชื่อมต่อเหล่านั้นลงในตารางสถานะ โดยสำหรับการเชื่อมต่อที่ใช้โปรโตคอลทีซีพีจะทำการจำลองแพ็คเก็ตประเภท SYN พร้อมกับระบุว่าแพ็คเก็ตจำลองนี้เป็นแพ็คเก็ตของโปรโตคอลทีซีพี และสำหรับการเชื่อมต่อที่ใช้โปรโตคอลยูดีพีจะทำการจำลองแพ็คเก็ตธรรมดา (โปรโตคอลยูดีพีจะไม่มีแพ็คเก็ตประเภท SYN) พร้อมกับระบุว่าแพ็คเก็ตจำลองนี้เป็นแพ็คเก็ตของโปรโตคอลยูดีพี เฮดเดอร์ของแพ็คเก็ตที่จำลองขึ้นจะต้องมีไอพีแอดเดรสต้นทาง , ไอพีแอดเดรสปลายทาง , พอร์ตต้นทาง และพอร์ตปลายทาง เหมือนกับแพ็คเก็ตของการเชื่อมต่อที่กำลังจะถูกย้าย นอกจากนี้ยังต้องทำการใส่ข้อมูลพิเศษไว้ในแพ็คเก็ตจำลอง เพื่อเป็นการบอกให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่อยู่อีกฝั่งรู้ว่าแพ็คเก็ตนี้เป็นแพ็คเก็ตที่ถูกจำลองขึ้น

3. อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ใช้อัลกอริทึมสำหรับการกระจายโหลดที่ผู้ดูแลระบบกำหนด ในการเลือกไฟร์วอลล์แบบรักษาสถานะตัวใหม่ที่จะทำการย้ายแต่ละการเชื่อมต่อไปส่งผ่าน โดยไม่จำเป็นว่าทุกการเชื่อมต่อจะถูกย้ายไปยังไฟร์วอลล์ตัวเดียวกัน

4. อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ส่งแพ็คเก็ตที่จำลองขึ้นจากขั้นตอนที่ 2 ไปยังไฟร์วอลล์แบบรักษาสถานะที่ถูกเลือกจากขั้นตอนที่ 3 เพื่อหลอกให้ไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลของการเชื่อมต่อที่กำลังจะถูกย้ายมาลงในตารางสถานะของตน

5. เมื่อไฟร์วอลล์แบบรักษาสถานะพบแพ็คเก็ตประเภท SYN ของโปรโตคอลทีซีพี หรือแพ็คเก็ตธรรมดาของโปรโตคอลยูดีพี (แต่เป็นแพ็คเก็ตของโปรโตคอลยูดีพีที่ยังไม่มีข้อมูลของการเชื่อมต่ออยู่ในตารางสถานะ) มันจะทำการตรวจสอบกฎการแอคเซสในตารางบันทึกกฎการแอคเซสว่าการเชื่อมต่อนี้สามารถอนุญาตให้เกิดขึ้นได้หรือไม่ ถ้ามีกฎการแอคเซสอนุญาตไว้มันจะทำการบันทึกข้อมูลของการเชื่อมต่อนี้ลงในตารางสถานะ แล้วส่งแพ็คเก็ตจำลองต่อไปยังอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่อยู่ถัดไป

6. เมื่อแพ็คเก็ตจำลองถูกส่งมาถึงอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ลำดับถัดไปแล้ว อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ลำดับถัดไปจะทำลายแพ็คเก็ตจำลองทิ้ง และทำการปรับเปลี่ยนข้อมูลในตารางข้อมูลของตน เพราะต่อไปจะต้องส่งทราฟฟิคของการเชื่อมต่อนี้ผ่านไฟล์วอลล์แบบรักษาสถานะตัวใหม่

7. อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั้งสองฝั่ง จะทำการส่งทราฟฟิคที่เหลือ ผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่จนกระทั่งการเชื่อมต่อสิ้นสุด

5.5.2 การย้ายการเชื่อมต่อ เมื่อไฟร์วอลล์มีโหลดต่างกันมาก

เมื่อไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งมีโหลดมากกว่าไฟร์วอลล์แบบรักษาสถานะตัวอื่นมาก ๆ วิธีที่นำเสนอจะทำการย้ายการเชื่อมต่อบางการเชื่อมต่อจากไฟร์วอลล์ตัวที่มีโหลดมากไปยังไฟร์วอลล์แบบรักษาสถานะตัวที่มีโหลดน้อยกว่า ขั้นตอนสำหรับการย้ายการเชื่อมต่อมีดังต่อไปนี้

1. เมื่ออุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ตรวจพบว่ามีไฟร์วอลล์ตัวใดตัวหนึ่งมีโหลดมากกว่าไฟร์วอลล์แบบรักษาสถานะตัวอื่นมาก อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ตัวนั้น จะทำการตัดสินใจว่าควรจะทำการย้ายการเชื่อมต่อใดบ้างไปยังไฟร์วอลล์แบบรักษาสถานะตัวอื่น เพื่อทำให้โหลดที่ผ่านไฟร์วอลล์แบบรักษาสถานะแต่ละตัวมีความสมดุลย์

2. หลังจากนั้นอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะทำการจำลองแพ็คเก็ตพิเศษสำหรับการเชื่อมต่อที่จะทำการย้ายไปยังไฟร์วอลล์แบบรักษาสถานะตัวใหม่ เพื่อใช้สำหรับหลอกให้ไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลของการเชื่อมต่อเหล่านั้นลงในตารางสถานะ โดยสำหรับการเชื่อมต่อที่ใช้โปรโตคอลทีซีพีจะทำการจำลองแพ็คเก็ตประเภท SYN พร้อมกับระบุว่าแพ็คเก็ตจำลองนี้เป็นแพ็คเก็ตของโปรโตคอลทีซีพี และสำหรับการเชื่อมต่อที่ใช้โปรโตคอลยูดีพีจะทำการจำลองแพ็คเก็ตธรรมดา (โปรดตคอลยูดีพีจะไม่มีแพ็คเก็ตประเภท SYN) พร้อมกับระบุว่าแพ็คเก็ตจำลองนี้เป็นแพ็คเก็ตของโปรโตคอลยูดีพี เฮดเดอร์ของแพ็คเก็ตที่จำลองขึ้นจะต้องมีไอพีแอดเดรสต้นทาง , ไอพีแอดเดรสปลายทาง , พอร์ตต้นทาง และพอร์ตปลายทาง เหมือนกับแพ็คเก็ตของการชื่อมต่อที่เราจะทำการย้าย นอกจากนี้ยังต้องทำการใส่ข้อมูลพิเศษไว้ในแพ็คเก็ตจำลอง เพื่อเป็นการบอกให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่อยู่อีกฝั่งรู้ว่าแพ็คเก็ตนี้เป็นแพ็คเก็ตที่ถูกจำลองขึ้น

3. อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทำการเลือกไฟร์วอลล์แบบรักษาสถานะตัวใหม่ที่จะทำการย้ายการเชื่อมต่อไป

4. อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ส่งแพ็คเก็ตที่จำลองขึ้นจากขั้นตอนที่ 2 ไปยังไฟร์วอลล์แบบรักษาสถานะที่ถูกเลือกจากขั้นตอนที่ 3 เพื่อหลอกให้ไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลของการเชื่อมต่อที่กำลังจะถูกย้ายมาลงในตารางสถานะของตน

5. เมื่อไฟร์วอลล์แบบรักษาสถานะพบแพ็คเก็ตประเภท SYN ของโปรดตคอลทีซีพี หรือแพ็คเก็ตธรรมดาของโปรโตรคอลยูดีพี (แต่เป็นแพ็คเก็ตของโปรโตคอลยูดีพีที่ยังไม่มีข้อมูลของการเชื่อมต่ออยู่ในตารางสถานะ) มันจะทำการตรวจสอบกฎการแอคเซสในตารางบันทึกกฎการแอคเซสว่าการเชื่อมต่อนี้สามารถอนุญาตให้เกิดขึ้นได้หรือไม่ ถ้ามีกฎการแอคเซสอนุญาตไว้มันจะทำการบันทึกข้อมูลของการเชื่อมต่อนี้ลงในตารางสถานะ แล้วส่งแพ็คเก็ตจำลองต่อไปยังอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่อยู่ถัดไป

6. เมื่อแพ็คเก็ตจำลองถูกส่งมาถึงอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ลำดับถัดไปแล้ว อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ลำดับถัดไปจะทำลายแพ็คเก็ตจำลองทิ้ง และทำการปรับเปลี่ยนข้อมูลในตารางข้อมูลของตน เพราะต่อไปจะต้องส่งทราฟฟิคของการเชื่อมต่อนี้ผ่านไฟล์วอลล์แบบรักษาสถานะตัวใหม่

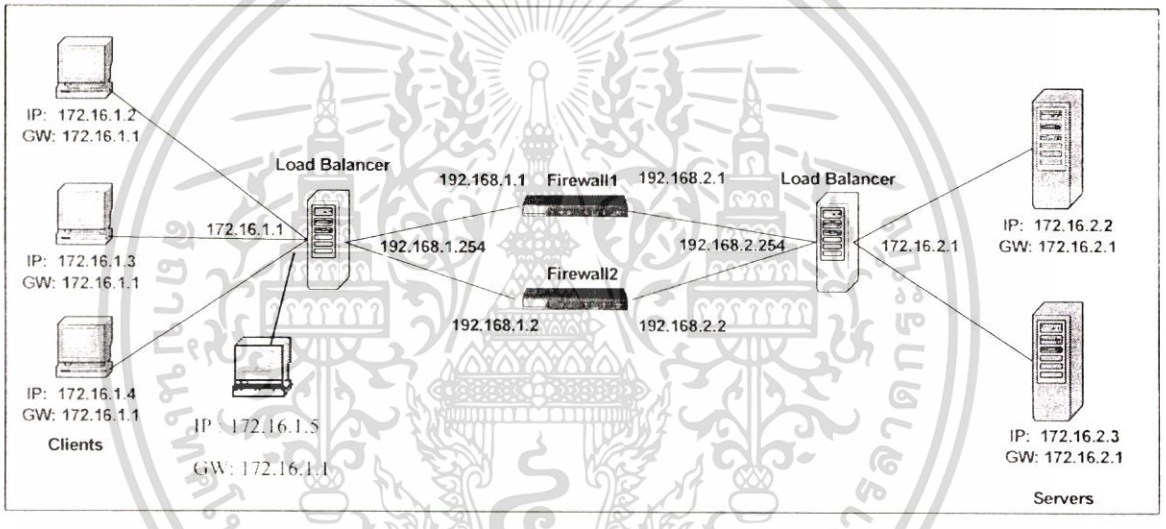
7. อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั้งสองฝั่งจะทำการส่งทราฟฟิคที่เหลือ ผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่จนกระทั่งการเชื่อมต่อสิ้นสุด

สำหรับแอพพลิเคชั่นที่ใช้หลาย ๆ พอร์ต เช่น เอฟทีพี ที่ใช้ทั้งพอร์ต 20 และ 21 ต้องระวังว่าเราจะต้องทำการส่งแพ็คเก็ตจำลองจากอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จากฝั่งที่ถูกต้อง จึงจะทำให้ไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลทั้งของการเชื่อมต่อที่ใช้พอร์ต 20 และการเชื่อมต่อที่ใช้พอร์ต 21 ลงในตารางสถานะ บางครั้งทิศทางที่เราจะต้องทำการส่งแพ็คเก็ตจำลองจะขัดแย้งกับการทำงานของแอพพลิเคชั่น ซึ่งมีสาเหตุมาจากการที่ไฟร์วอลล์แบบรักษาสถานะใช้วิธีในการกำหนดกฎการแอคเซสแบบทิศทางเดียว ตัวอย่างเช่น เมื่อเราใช้โปรโตคอลเอฟทีพี ไคลเอ็นต์จะทำหน้าที่สร้างการเชื่อมต่อโดยใช้พอร์ต 21 ไปยังเซิร์ฟเวอร์ และจะใช้พอร์ต 21 ในการส่งคำสั่งสำหรับการส่งข้อมูลธรรมดา (data) เช่น การส่งไฟล์ข้อมูลจากเซิร์ฟเวอร์ไปยังไคลเอ็นต์ เซิร์ฟเวอร์จะเป็นผู้ทำหน้าที่สร้างการเชื่อมต่อโดยใช้พอร์ต 20 ไปยังไคลเอ็นต์ ซึ่งจากตัวอย่างนี้จะเห็นว่าอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่อยู่ฝั่งไคลเอ็นต์ควรจะเป็นผู้ส่งแพ็คเก็ตจำลอง (แพ็คเก็ต SYN) สำหรับการเชื่อมต่อที่ใช้พอร์ต 21 ไปยังไฟร์วอลล์แบบรักษาสถานะ และอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่อยู่ฝั่งเซิร์ฟเวอร์ควรจะเป็นผู้ส่งแพ็คเก็ตจำลองสำหรับการเชื่อมต่อที่ใช้พอร์ต 20 ไปยังไฟร์วอลล์ แต่อันที่จริงแล้วถ้าต้องการให้ทั้งการเชื่อมต่อที่ใช้พอร์ต 20 และการเชื่อมต่อที่ใช้พอร์ต 21 สามารถผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่ได้นั้น อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่อยู่ฝั่งไคลเอนต์ต้องเป็นผู้ส่งแพ็คเก็ตจำลองทั้งสำหรับการเชื่อมต่อที่ใช้พอร์ต 20 และการเชื่อมต่อที่ใช้พอร์ต 21 เพราะว่ากฎการแอคเซสสำหรับโปรโตคอลเอฟทีพีในไฟร์วอลล์แบบรักษาสถานะมีทิศทางจากฝั่งไคลเอ็นต์ไปฝั่งเซิร์ฟเวอร์เท่านั้น

บทที่ 6

การทดลอง

ในบทนี้เราจะทำการทดลองเพื่อพิสูจน์ว่าวิธีที่เรานำเสนอสามารถนำมาใช้ได้จริง โดยเราจะนำโปรแกรมที่เขียนเพื่อทำงานตามวิธีที่นำเสนอ มาทำการทดสอบกับไฟร์วอลล์แบบรักษาสถานะยี่ห้อ Juniper (NetScreen เดิม) รุ่น NS-204 เราทำการเชื่อมต่ออุปกรณ์ที่จะใช้ในการทดสอบเหมือนดังรูปที่ 6.1 เราใช้เครื่องคอมพิวเตอร์ยี่ห้อ HP 4 เครื่อง สำหรับทำเป็นเครื่องไคลเอ็นต์ และใช้เครื่องคอมพิวเตอร์ยี่ห้อ IBM จำนวน 4 เครื่อง ใช้สำหรับทำเป็นเครื่องเซิร์ฟเวอร์ 2 เครื่อง และใช้สำหรับทำเป็นอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ 2 เครื่อง



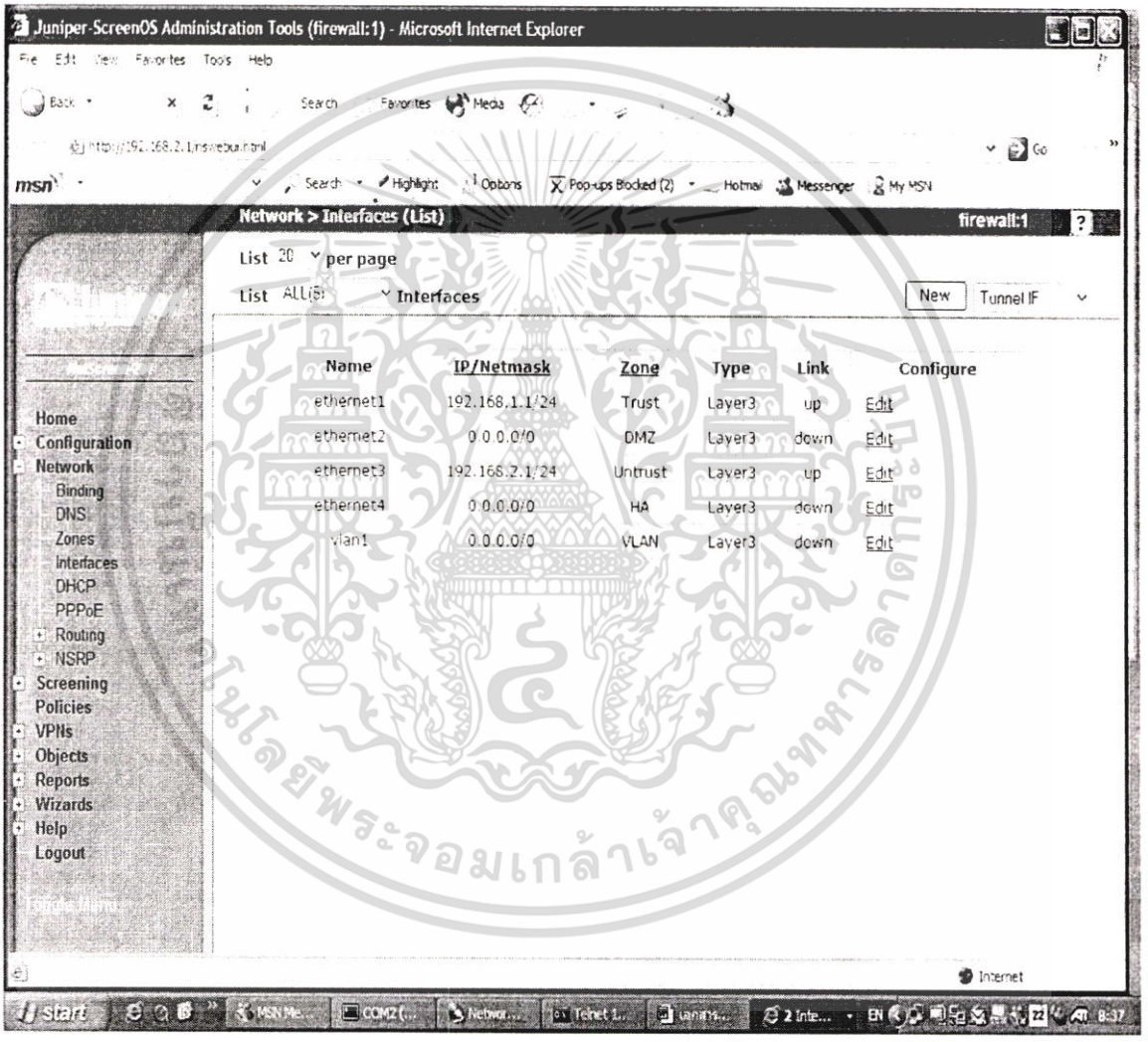
รูปที่ 6.1 แสดงรูปภาพเครือข่ายที่ใช้ในการทดสอบ

6.1 Configuration ของไฟร์วอลล์แบบรักษาสถานะ

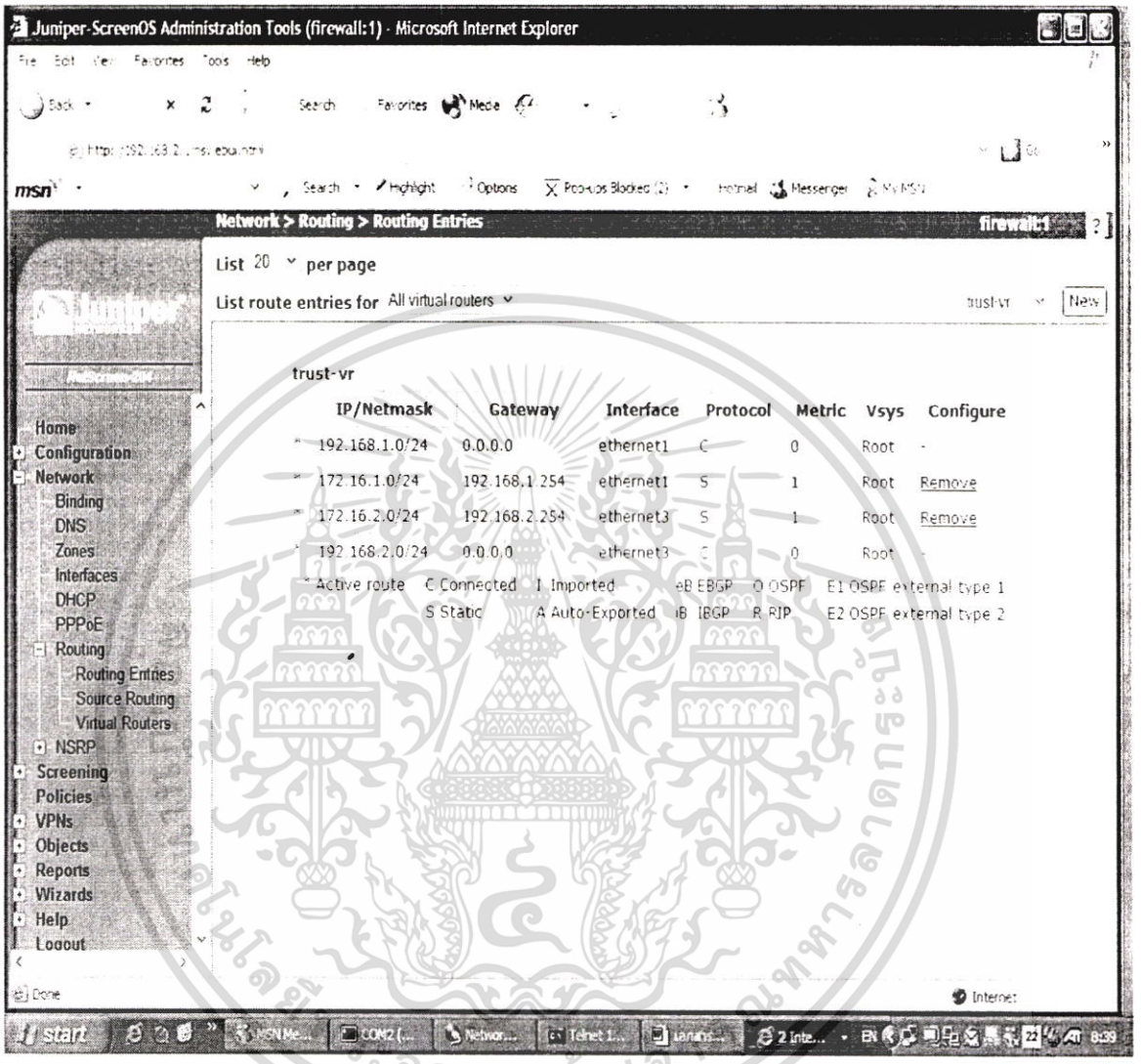
เราทำการกำหนดค่าให้กับไฟร์วอลล์แบบรักษาสถานะเป็นดังต่อไปนี้

6.1.1 การกำหนดค่าให้กับไฟร์วอลล์แบบรักษาสถานะตัวที่หนึ่ง

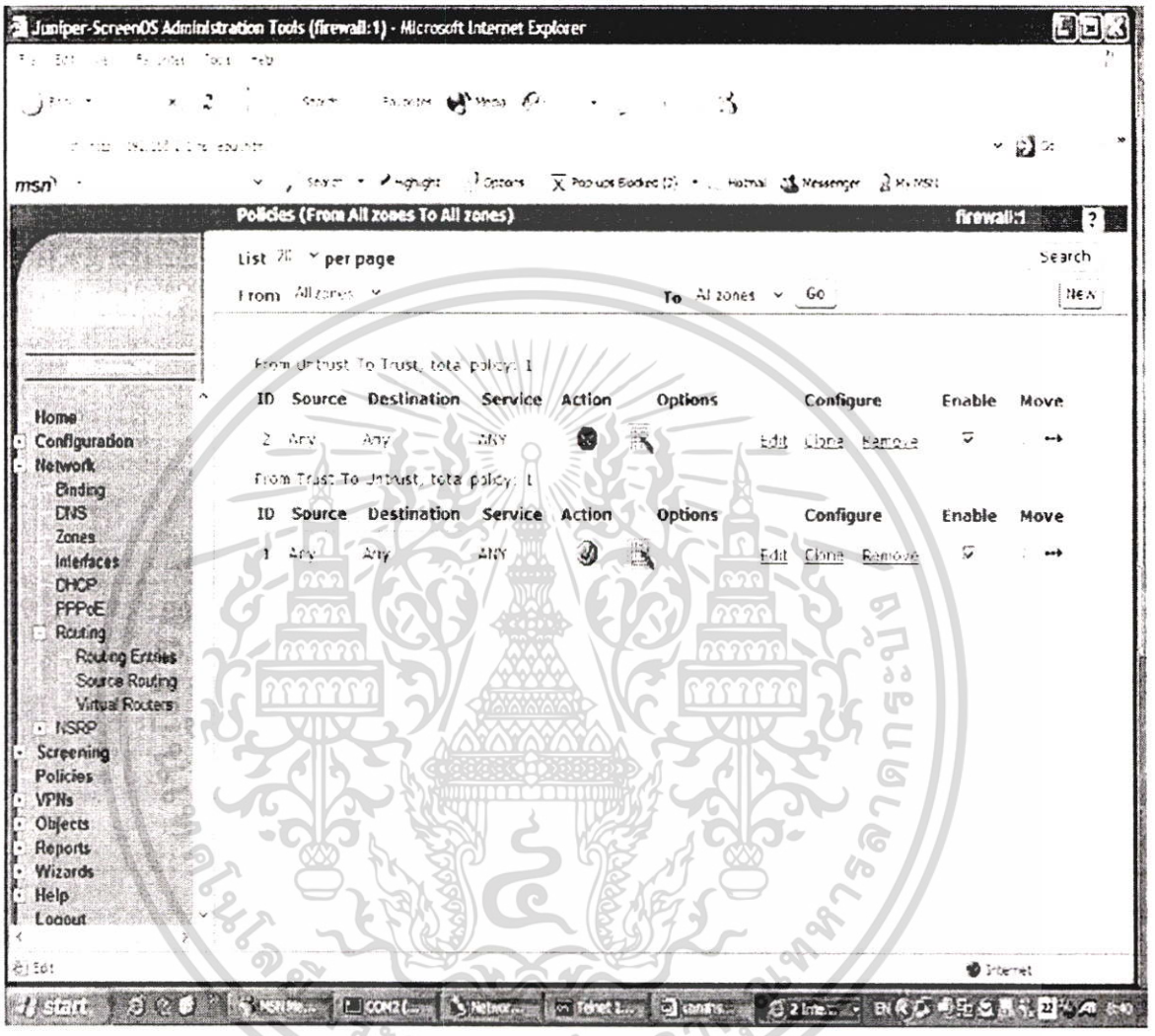
เราทำการกำหนดค่าไอพีแอดเดรสให้กับไฟร์วอลล์แบบรักษาสถานะตัวที่หนึ่งดังรูปที่ 6.2 กำหนดค่าเราต์ให้กับไฟร์วอลล์แบบรักษาสถานะตัวที่หนึ่งดังรูปที่ 6.3 และกำหนดกฎการแอคเซสให้กับไฟร์วอลล์แบบรักษาสถานะตัวที่หนึ่งดังรูปที่ 6.4



รูปที่ 6.2 ค่าไอพีแอดเดรสของแต่ละอินเตอร์เฟซ



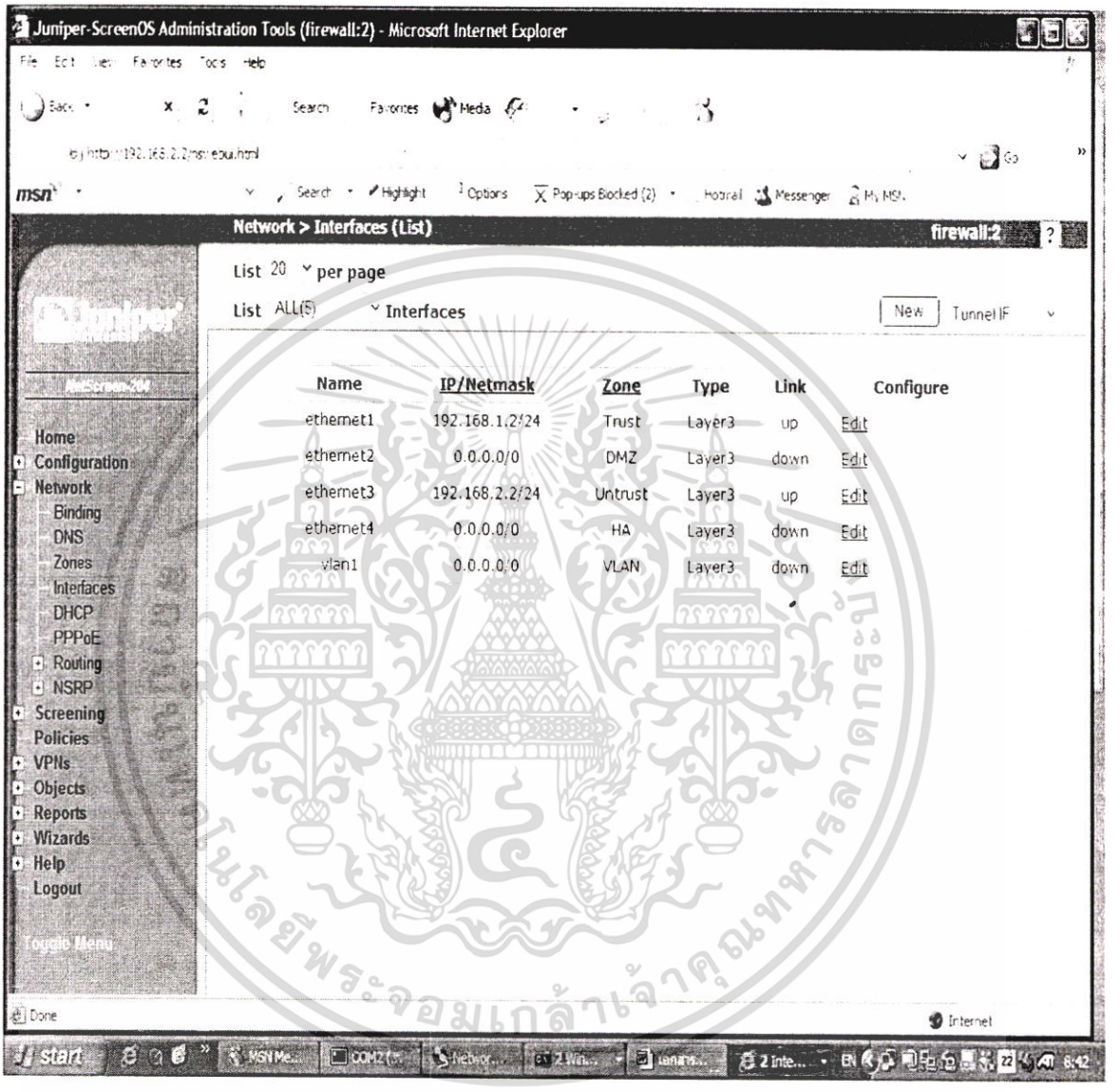
รูปที่ 6.3 ค่าเราต์ของไฟร์วอลล์แบบรักษาสถานะตัวที่หนึ่ง



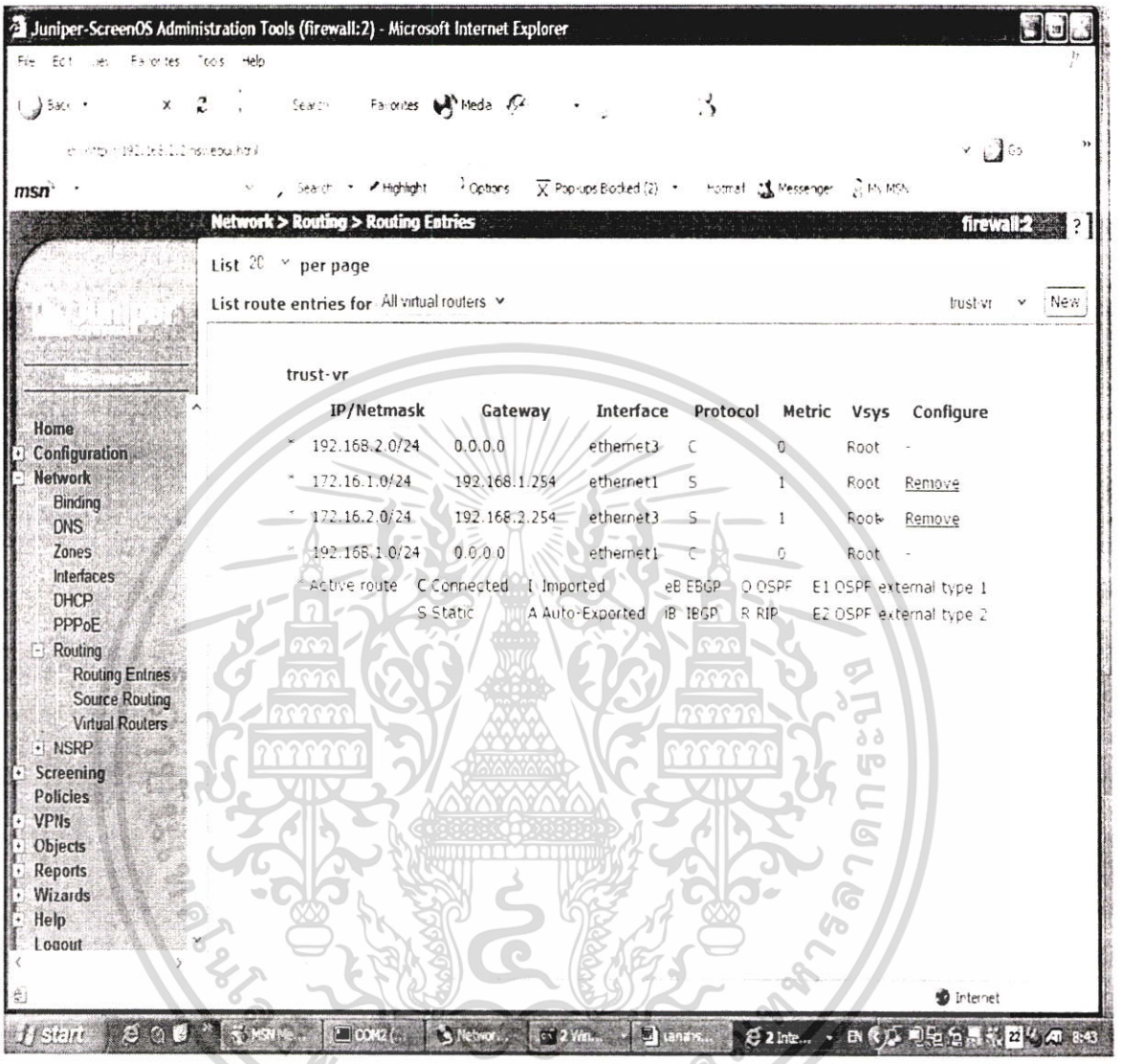
รูปที่ 6.4 กฎการแอคเซสของไฟร์วอลล์แบบรักษาสถานะตัวที่หนึ่ง

6.1.2 การกำหนดค่าให้กับไฟร์วอลล์แบบรักษาสถานะตัวที่สอง

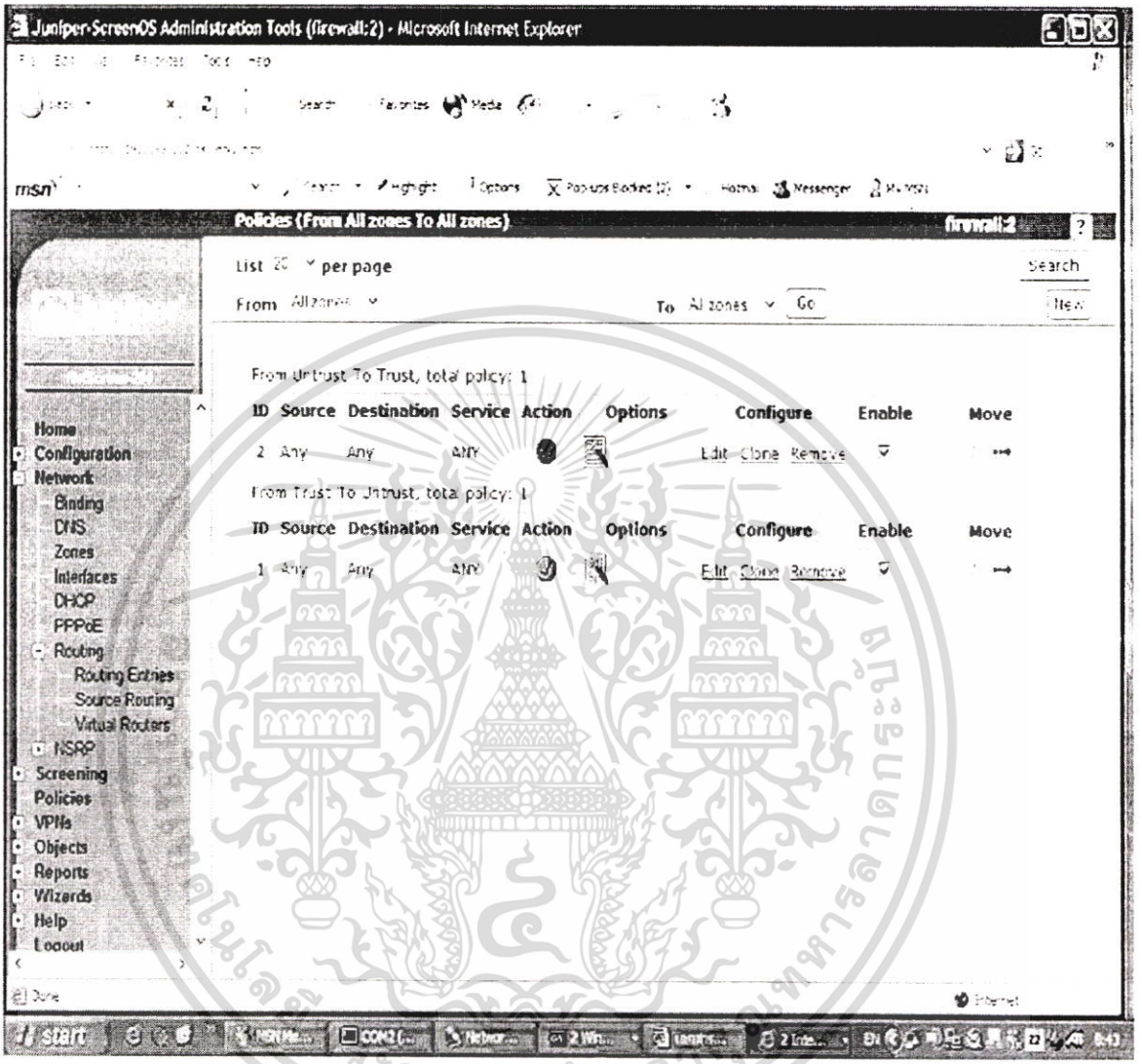
เราทำการกำหนดค่าไอพีแอดเดรสให้กับไฟร์วอลล์แบบรักษาสถานะตัวที่สองดังรูปที่ 6.5 กำหนดค่าเราต์ให้กับไฟร์วอลล์แบบรักษาสถานะตัวที่สองดังรูปที่ 6.6 และกำหนดกฎการแอคเซสให้กับไฟร์วอลล์แบบรักษาสถานะตัวที่สองดังรูปที่ 6.7



รูปที่ 6.5 ค่าไอพีแอดเดรสของแต่ละอินเตอร์เฟซ



รูปที่ 6.6 ค่าเราต์ของไฟร์วอลล์แบบรักษาสถานะตัวที่สอง



รูปที่ 6.7 กฎการแอคเซสของไฟร์วอลล์แบบรักษาสถานะตัวที่สอง

6.2 การทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี

การทดสอบนี้มีจุดประสงค์เพื่อแสดงให้เห็นว่าอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่มีอยู่ทั่วไปในท้องตลอด ไม่สามารถทำการรักษาสถานะภาพของการเชื่อมต่อให้ดำเนินจนจบหรือดำเนินต่อไปได้ เมื่อไฟร์วอลล์ตัวใดตัวหนึ่งไม่สามารถทำงานต่อ

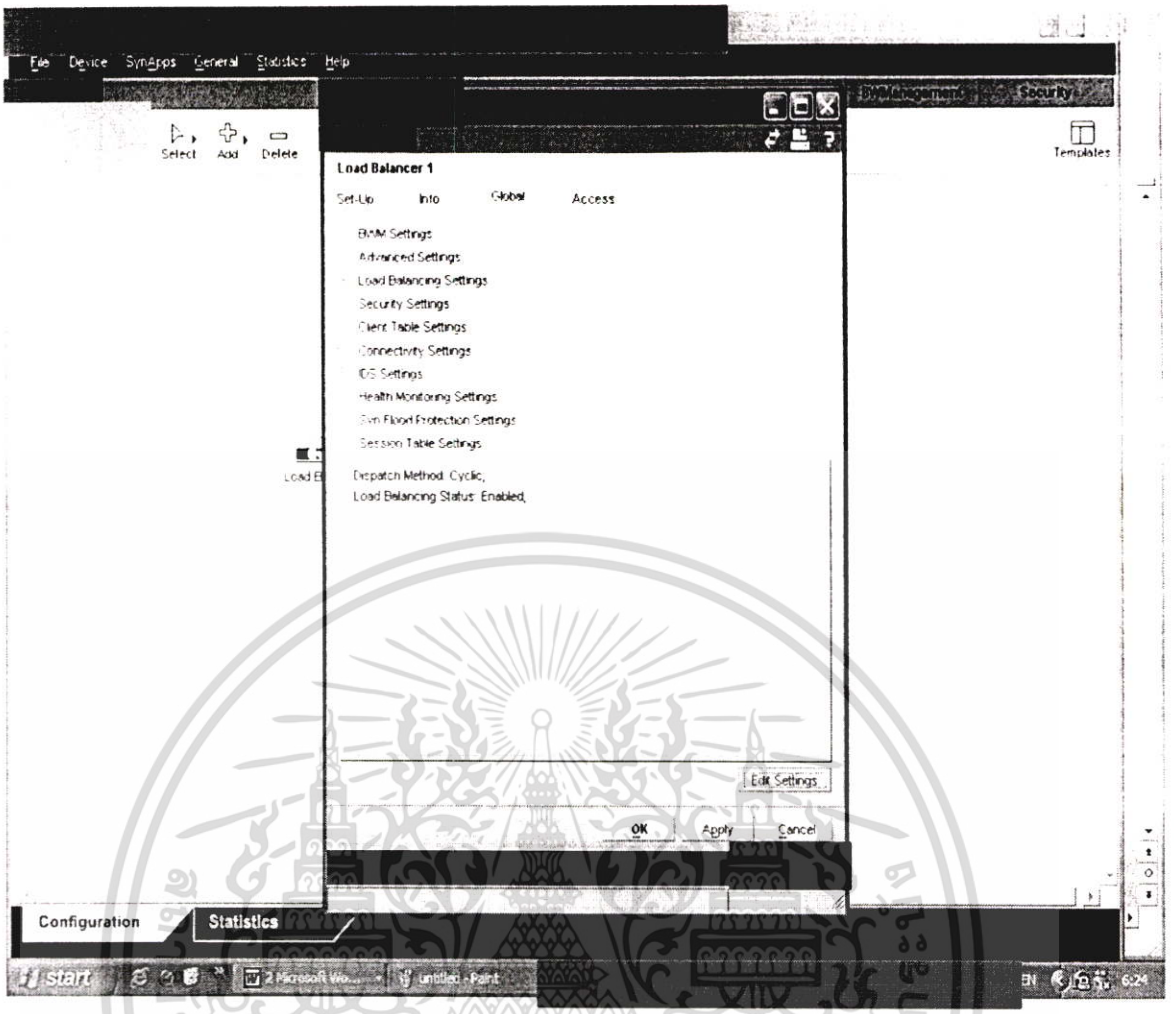
ในการทดสอบนี้เราจะทำการทดสอบโดยใช้วิธีที่นำเสนอ เปรียบเทียบกับเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปยี่ห้อหนึ่ง ในการทดสอบนี้เราจะทำการทดสอบโดยใช้อัลกอริทึมดังต่อไปนี้ในการกระจายโหลดให้กับไฟร์วอลล์ตามลำดับ

1. แบบวนรอบ
2. แบบจำนวนการเชื่อมต่อน้อยที่สุด
3. แบบจำนวนแพ็คเก็ตน้อยที่สุด
4. แบบวิธีการแฮ็ช

เราจะทำการทดสอบโดยใช้โปรโตคอลเอฟทีพีซึ่งทำงานบนโปรโตคอลทีซีพี ในการโอนย้ายไฟล์จากเครื่องเซิร์ฟเวอร์มายังเครื่องไคลเอ็นต์ แล้วทำการสังเกตว่าเมื่อทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งแล้ว จะมีจำนวนการเชื่อมต่อที่สามารถทำงานจนจบหรือทำงานต่อไปได้เป็นจำนวนกี่การเชื่อมต่อ โดยจะทำการทดสอบโดยใช้วิธีที่นำเสนอเปรียบเทียบกับการใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป

6.2.1 ทดสอบโดยใช้การกระจายโหลดแบบวนรอบ

การทดสอบนี้จะใช้การกระจายโหลดแบบวนรอบสำหรับกระจายโหลดให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัว โดยเราจะทำการกำหนดให้วิธีที่นำเสนอทำการย้ายการเชื่อมต่อเมื่อตรวจพบว่ามีไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งหยุดทำงาน เราจะทำการสร้างการเชื่อมต่อจากเครื่องไคลเอ็นต์ไปยังเครื่องเซิร์ฟเวอร์โดยใช้โปรโตคอลเอฟทีพี เพื่อทำการโอนย้ายไฟล์ข้อมูลขนาด 500 เมกะไบต์ จากเครื่องเซิร์ฟเวอร์ไปยังเครื่องไคลเอ็นต์ โดยเครื่องไคลเอ็นต์ 172.16.1.2 กับ 172.16.1.3 จะทำการสร้างการเชื่อมต่อไปยังเครื่องเซิร์ฟเวอร์ 172.16.2.2 และเครื่องไคลเอ็นต์ 172.16.1.4 กับ 172.16.1.5 จะทำการสร้างการเชื่อมต่อไปยังเซิร์ฟเวอร์ 172.16.2.3 เราจะทำการทดสอบเป็นจำนวน 10 ครั้ง โดยการทดสอบแต่ละครั้งเราจะสร้างการเชื่อมต่อเป็นจำนวนไม่เท่ากัน เมื่อทำการสร้างการเชื่อมต่อจนครบตามจำนวนที่ต้องการในแต่ละการทดสอบแล้ว เราจะทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวแรก แล้วทำการนับจำนวนการเชื่อมต่อที่ทำงานจนจบหรือการเชื่อมต่อที่สามารถดำเนินต่อไป เมื่อทดสอบโดยใช้วิธีที่นำเสนอและเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป เรากำหนดให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปใช้การกระจายโหลดแบบวนรอบ ดังรูปที่ 6.8



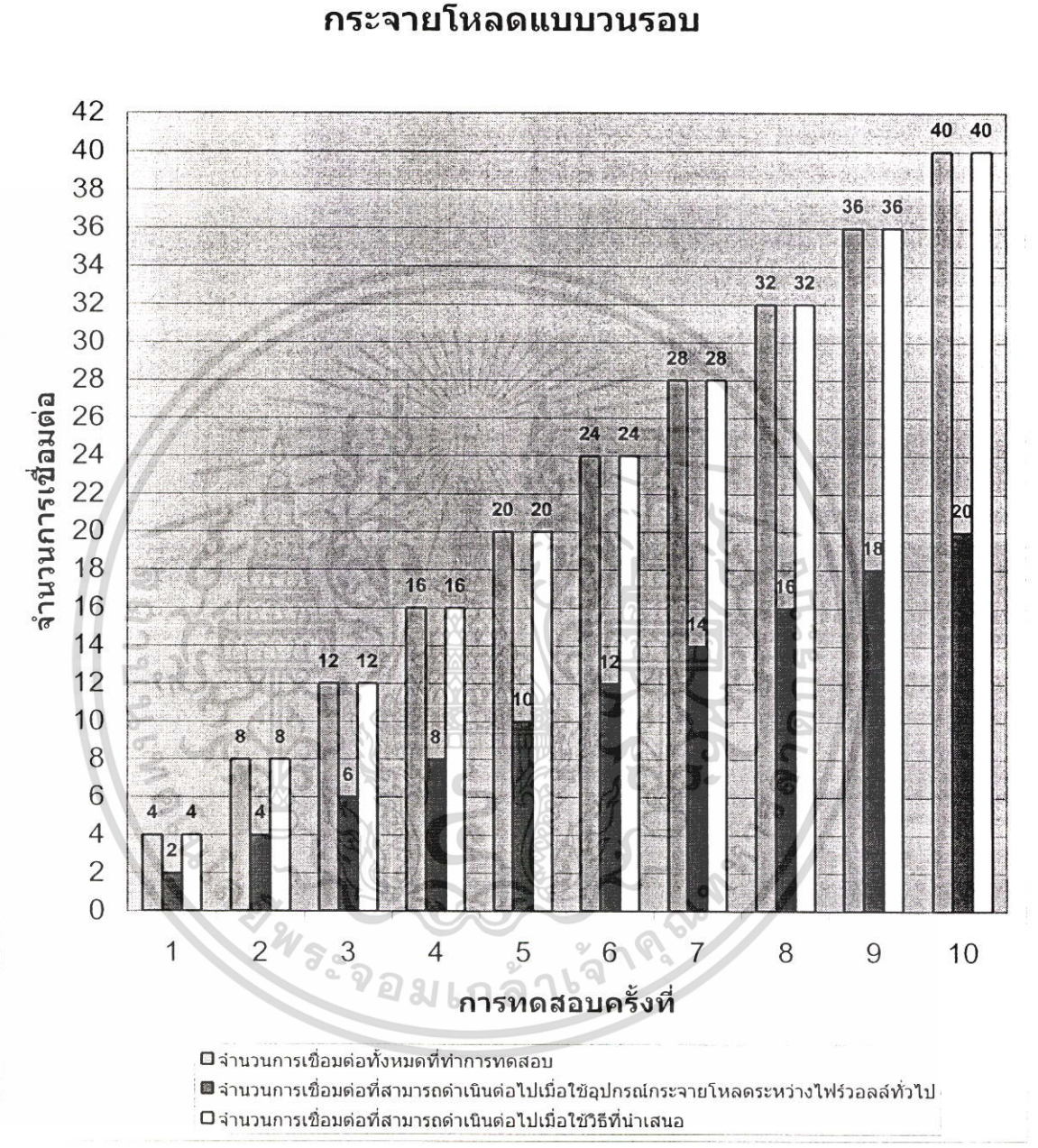
รูปที่ 6.8 แสดงการกำหนดค่าให้กับอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปใช้วิธีกระจายโหลดแบบวนรอบ

หลังจากทำการทดสอบแล้ว เราได้ผลลัพธ์ของการทดสอบดังตารางที่ 6.1 โดยคอลัมน์แรกแสดงจำนวนการเชื่อมต่อทั้งหมดที่ถูกสร้างระหว่างเครื่องไคลเอ็นต์กับเครื่องเซิร์ฟเวอร์ โดยค่าที่อยู่ในวงเล็บจะเป็นจำนวนการเชื่อมต่อที่สร้างจากเครื่องไคลเอ็นต์แต่ละตัวในการทดสอบนั้น คอลัมน์ที่สองแสดงจำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป คอลัมน์ที่สามแสดงจำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้วิธีที่นำเสนอ การนับจำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้จะทำหลังจากได้ทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวแรกเรียบร้อยแล้ว

ตารางที่ 6.1 แสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบวนรอบ

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้วิธีที่นำเสนอ
4 (1)	2	4
8 (2)	4	8
12 (3)	6	12
16 (4)	8	16
20 (5)	10	20
24 (6)	12	24
28 (7)	14	28
32 (8)	16	32
36 (9)	18	36
40 (10)	20	40

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง



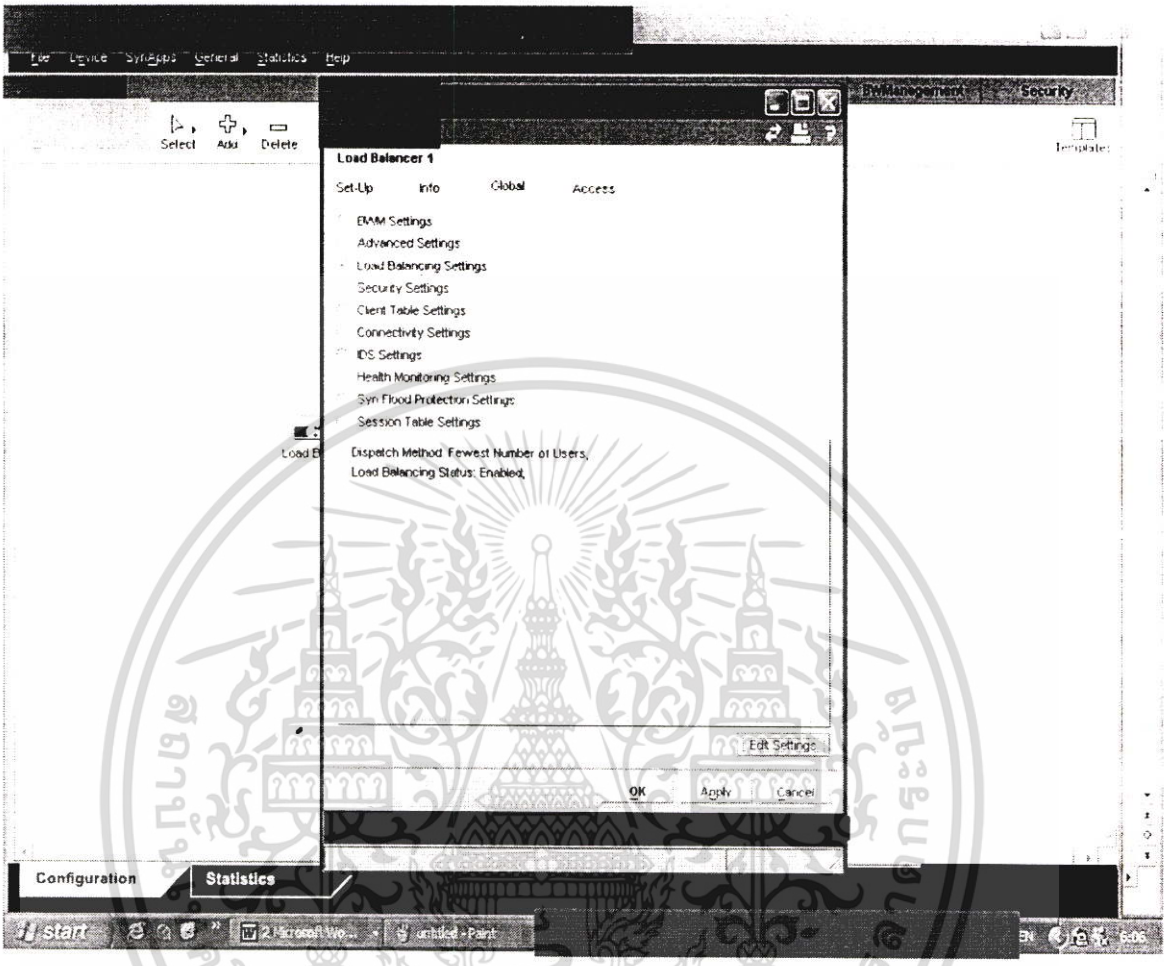
รูปที่ 6.9 กราฟแสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบวนรอบ

6.2.1.1 การวิเคราะห์ผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบวนรอบ

การกระจายโหลดแบบวนรอบจะทำการกระจายการเชื่อมต่อให้กับไฟร์วอลล์แบบรักษา-สถานะแต่ละตัวเป็นจำนวนที่เท่ากัน ดังนั้นเมื่อทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวแรกแล้ว ถ้าเป็นการทดสอบที่ใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป การเชื่อมต่อที่ถูกกระจายไปยังไฟร์วอลล์แบบรักษาสถานะตัวแรกหรือตัวที่ถูกปิดนั้นจะไม่สามารถดำเนินต่อไปได้ จากตารางที่ 6.1 จะเห็นว่า การเชื่อมต่อที่สามารถดำเนินต่อไปได้จะเหลือเพียงครึ่งเดียว ซึ่งการเชื่อมต่อเหล่านั้นเป็นการเชื่อมต่อที่ถูกกระจายไปยังไฟร์วอลล์ตัวที่ไม่ถูกปิดนั้นเอง ถ้าเป็นการทดสอบที่ใช้วิธีที่นำเสนอการเชื่อมต่อทั้งหมดทุกการเชื่อมต่อจะสามารถดำเนินต่อไปและสามารถทำงานจนจบ สาเหตุที่การเชื่อมต่อสามารถดำเนินต่อไปได้นั้นก็เพราะ วิธีที่นำเสนอจะทำการย้ายการเชื่อมต่อที่ถูกส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวที่ถูกปิดไปยังไฟร์วอลล์แบบรักษาสถานะตัวที่เหลืออยู่ ดังนั้นการเชื่อมต่อทั้งหมดจึงสามารถดำเนินต่อไปได้

6.2.2 ทดสอบโดยใช้การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุด

การทดสอบนี้จะใช้การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุดสำหรับกระจายโหลดให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัว โดยการทดสอบนี้จะทำการทดสอบเช่นเดียวกับการทดสอบที่ 6.2.1 แต่เราจะทำการกำหนดให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปใช้การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุดแทนการกระจายโหลดแบบวนรอบ ดังรูปที่ 6.10 และทำการกำหนดให้วิธีที่นำเสนอใช้วิธีกระจายโหลดวิธีเดียวกัน แต่ให้ทำการย้ายการเชื่อมต่อด้วยเมื่อตรวจพบว่ามีไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งหยุดทำงาน



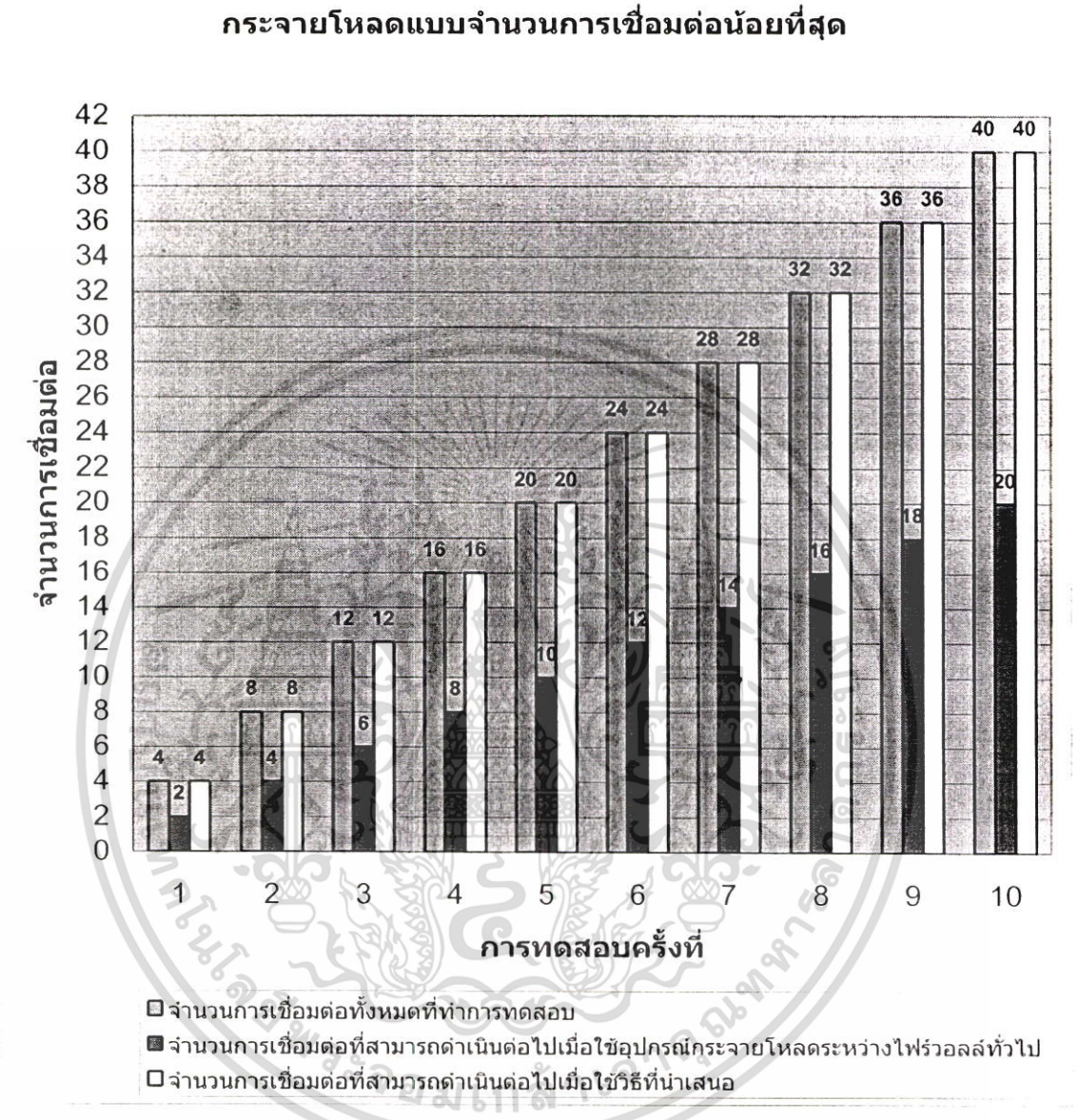
รูปที่ 6.10 แสดงการกำหนดค่าให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปใช้วิธีการกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุด

หลังจากทำการทดสอบที่ 6.2.2 แล้ว เราได้ผลลัพธ์ของการทดสอบดังตารางที่ 6.2 และรูปที่ 6.11 ซึ่งจะแสดงต่อไปนี้

ตารางที่ 6.2 แสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุด

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้วิธีที่นำเสนอ
4 (1)	2	4
8 (2)	4	8
12 (3)	6	12
16 (4)	8	16
20 (5)	10	20
24 (6)	12	24
28 (7)	14	28
32 (8)	16	32
36 (9)	18	36
40 (10)	20	40

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง



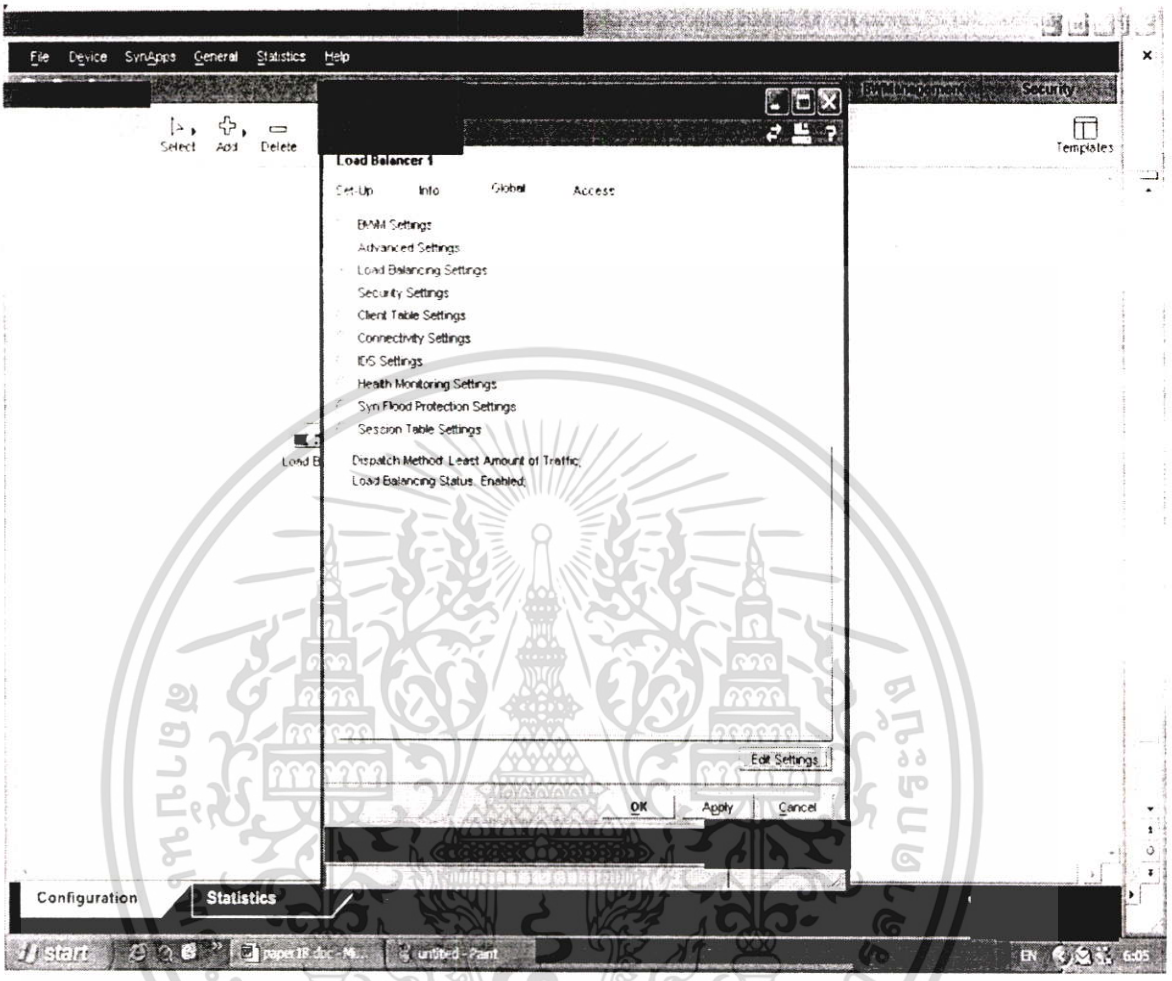
รูปที่ 6.11 กราฟแสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุด

6.2.2.1 การวิเคราะห์ผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุด

จากการทดสอบที่ 6.2.2 การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุดจะทำการกระจายการเชื่อมต่อไปยังไฟร์วอลล์แบบรักษาสถานะทั้งสองตัวเป็นจำนวนที่เท่ากันเช่นเดียวกับการกระจายโหลดบนวนรอบ ดังนั้นเมื่อทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวแรกแล้ว ถ้าเกิดเป็นการทดสอบที่ใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป จะมีการเชื่อมต่อที่สามารถทำงานต่อไปได้เพียงครึ่งเดียว ซึ่งเป็นการเชื่อมต่อที่ถูกกระจายไปให้ไฟร์วอลล์แบบรักษาสถานะตัวที่ไม่ถูกปิด แต่ถ้าทำการทดสอบโดยใช้วิธีที่นำเสนอแล้ว การเชื่อมต่อทั้งหมดทุกการเชื่อมต่อจะสามารถดำเนินต่อไปได้เหมือนเป็นปกติ ทำให้สามารถทำการโอนย้ายไฟล์ได้จนครบ

6.2.3 ทดสอบโดยใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุด

การทดสอบนี้จะใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุดสำหรับกระจายโหลดให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัว โดยการทดสอบนี้จะทำการทดสอบเช่นเดียวกับการทดสอบที่ 6.2.1 และ 6.2.2 แต่เราจะทำการกำหนดให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุดแทน ดังรูปที่ 6.12 และทำการกำหนดให้วิธีที่นำเสนอใช้วิธีในการกระจายโหลดวิธีเดียวกัน แต่ให้ทำการย้ายการเชื่อมต่อด้วยเมื่อตรวจพบว่ามีไฟร์วอลล์ตัวใดตัวหนึ่งหยุดทำงาน



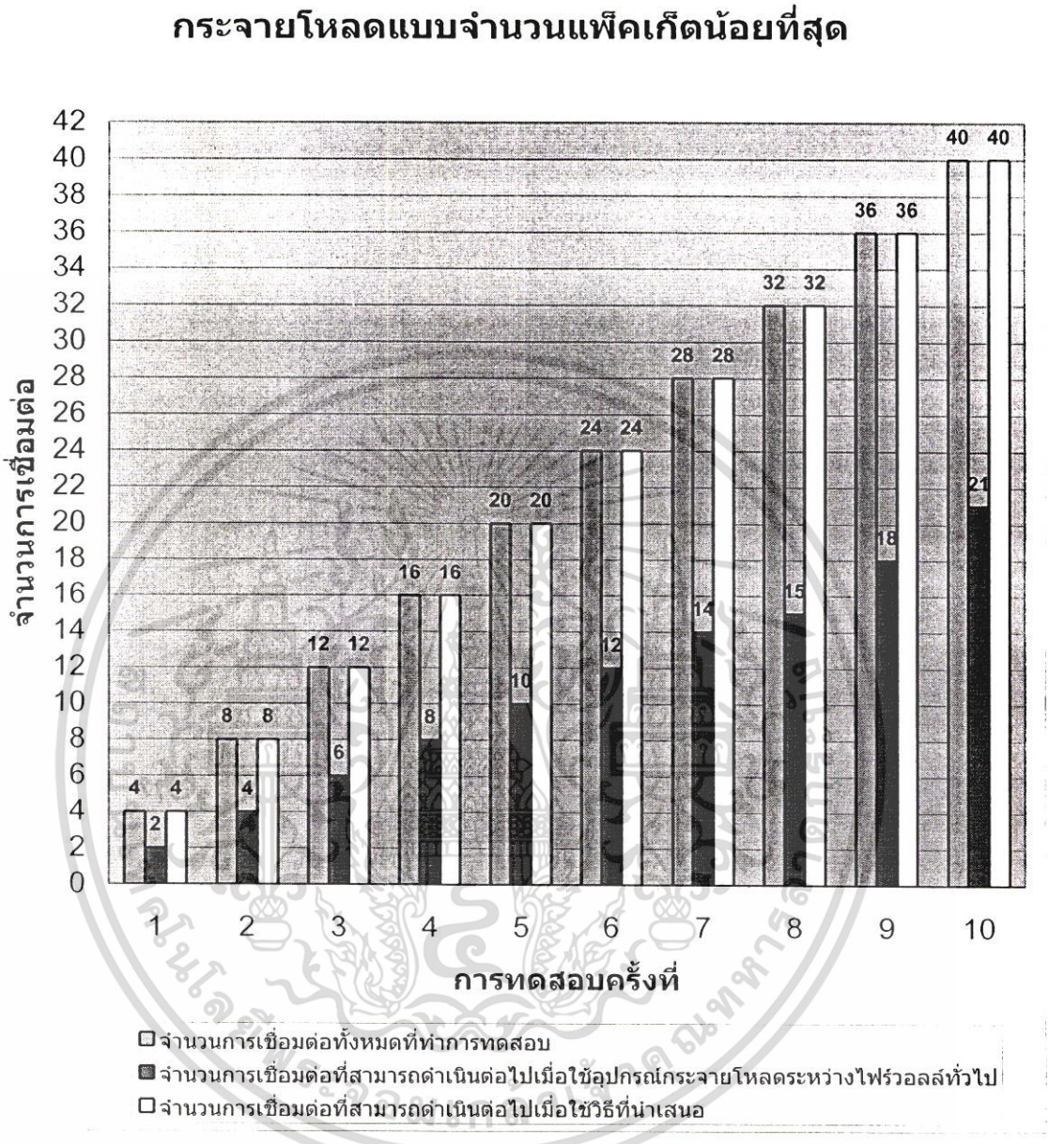
รูปที่ 6.12 แสดงการกำหนดค่าให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปใช้วิธีการกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุด

หลังจากทำการทดสอบที่ 6.2.3 แล้ว เราได้ผลลัพธ์ของการทดสอบดังตารางที่ 6.3 และรูปที่ 6.13 ซึ่งจะแสดงต่อไปนี้

ตารางที่ 6.3 แสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุด

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้วิธีที่นำเสนอ
4 (1)	2	4
8 (2)	4	8
12 (3)	6	12
16 (4)	8	16
20 (5)	10	20
24 (6)	12	24
28 (7)	14	28
32 (8)	15	32
36 (9)	18	36
40 (10)	21	40

เทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง



รูปที่ 6.13 กราฟแสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุด

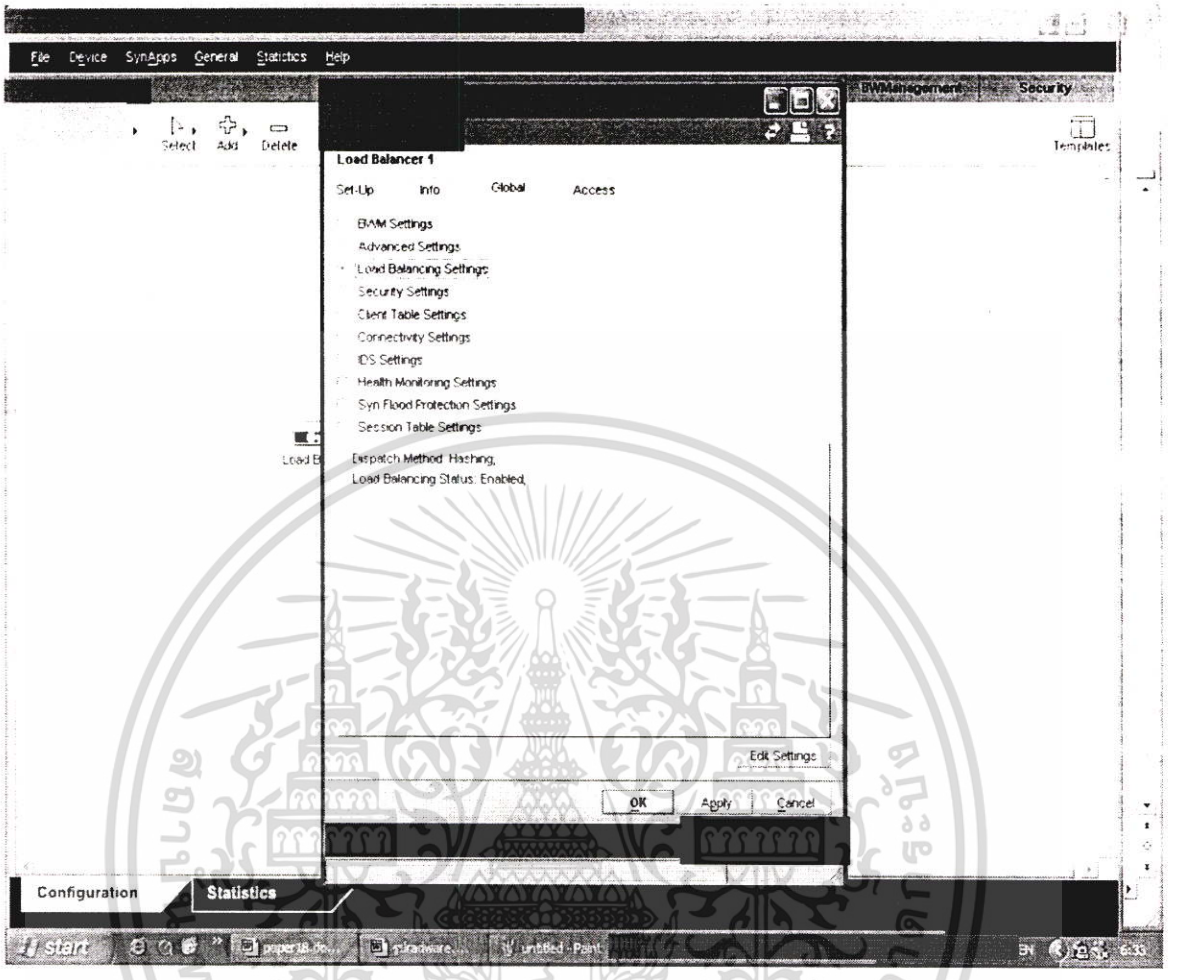
6.2.3.1 การวิเคราะห์ผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุด

ผลลัพธ์ของการทดสอบเมื่อใช้การกระจายโหลดแบบแพ็คเก็ตน้อยที่สุด จะมีผลลัพธ์ไปในแนวเดียวกับการทดสอบที่ 6.2.1 และ 6.2.2 คือ เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปแล้ว การเชื่อมต่อที่ถูกกระจายไปยังไฟร์วอลล์แบบรักษาสถานะตัวที่ถูกปิดจะหยุดตามไฟล์วอลล์ไปด้วย แต่เมื่อใช้วิธีที่นำเสนอการเชื่อมต่อทุกการเชื่อมต่อจะสามารถดำเนินต่อไปได้เหมือนเป็นปกติ

การกระจายโหลดโดยใช้การกระจายโหลดแบบแพ็คเก็ตน้อยที่สุด จะทำการกระจายการเชื่อมต่อโดยพิจารณาจำนวนแพ็คเก็ตต่อวินาทีที่ผ่านไฟร์วอลล์แบบรักษาสถานะแต่ละตัว ณ เวลาที่กำลังจะทำการกระจายการเชื่อมต่อ ดังนั้นจำนวนการเชื่อมต่อที่ถูกกระจายไปให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัวอาจจะไม่เท่ากัน ดังจะเห็นได้จากการทดสอบที่ทำการสร้างการเชื่อมต่อเป็นจำนวน 32 และ 40 การเชื่อมต่อ จะมีจำนวนการเชื่อมต่อที่ทำงานจนจบเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป ไม่ได้เป็นจำนวนครึ่งหนึ่งของจำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ

6.2.4 ทดสอบโดยใช้การกระจายโหลดแบบวิธีการแฮ็ช

การทดสอบนี้จะใช้การกระจายโหลดแบบวิธีการแฮ็ชสำหรับกระจายโหลดให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัว โดยการแฮ็ชนี้จะใช้ทั้งหมายเลขไอพีแอดเดรสต้นทาง หมายเลขไอพีแอดเดรสปลายทาง และหมายเลขพอร์ตของแพ็คเก็ต มาใช้ในการคำนวณหาไฟร์วอลล์ที่จะทำการกระจายการเชื่อมต่อไปให้ โดยการทดสอบนี้จะทำการทดสอบเช่นเดียวกับการทดสอบที่ผ่านแต่เราจะทำการกำหนดให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปใช้การกระจายโหลดแบบวิธีการแฮ็ชแทน ดังรูปที่ 6.14 และทำการกำหนดให้วิธีที่นำเสนอใช้วิธีในการกระจายโหลดวิธีเดียวกัน แต่ให้ทำการย้ายการเชื่อมต่อด้วยเมื่อตรวจพบว่ามีไฟร์วอลล์ตัวใดตัวหนึ่งหยุดทำงาน

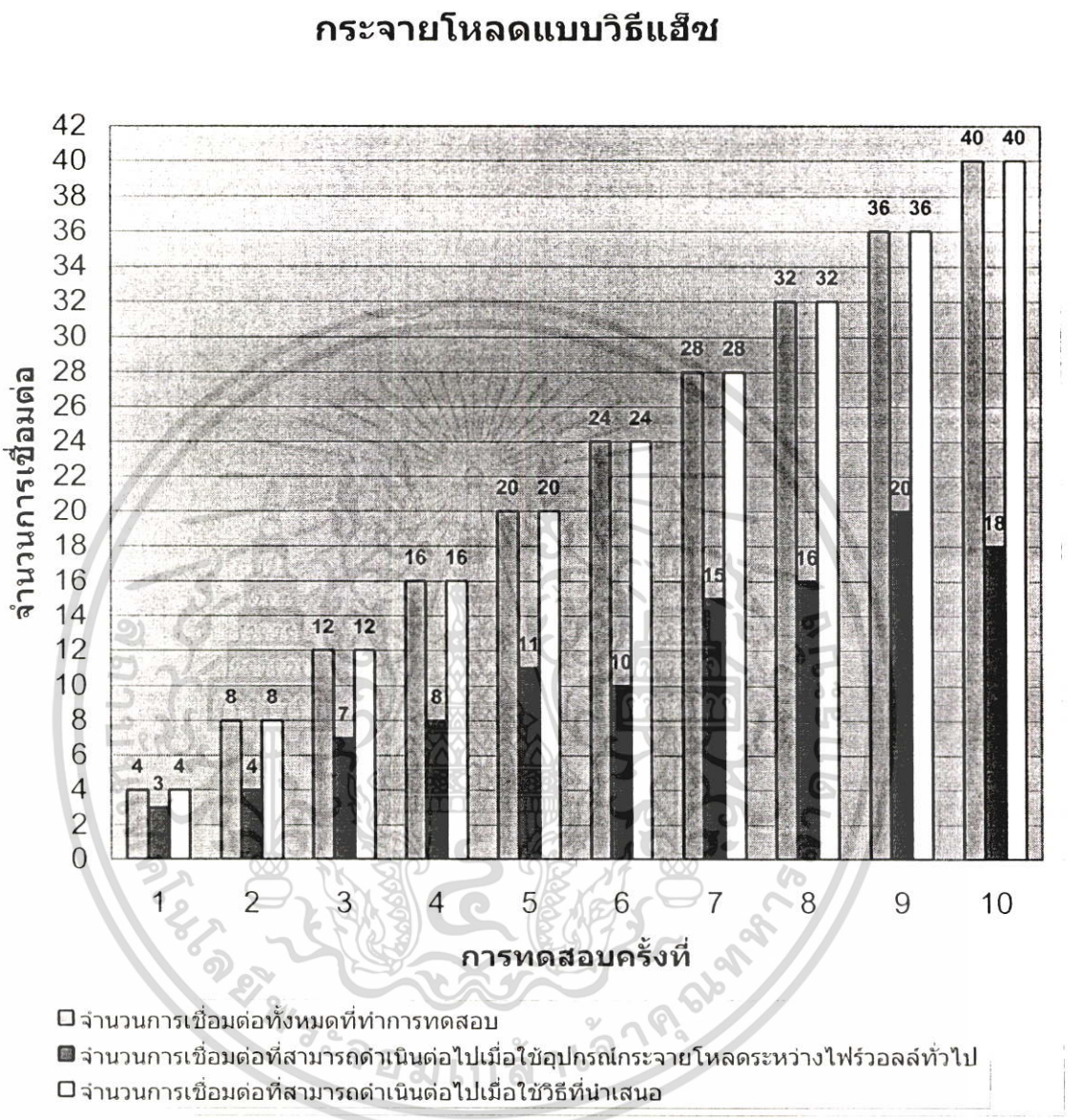


รูปที่ 6.14 แสดงการคำหนดค่าให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปใช้วิธีการกระจายโหลดแบบวิธีการแฮ็ช

หลังจากทำการทดสอบที่ 6.2.4 แล้ว เราได้ผลลัพธ์ของการทดสอบดังตารางที่ 6.4 และรูปที่ 6.15 ซึ่งจะแสดงต่อไปนี้

ตารางที่ 6.4 แสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบวิธีการแฮ็ช

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้วิธีที่นำเสนอ
4 (1)	3	4
8 (2)	4	8
12 (3)	7	12
16 (4)	8	16
20 (5)	11	20
24 (6)	10	24
28 (7)	15	28
32 (8)	16	32
36 (9)	20	36
40 (10)	18	40



รูปที่ 6.15 กราฟแสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบวิธีการแฮ็ซ

6.2.4.1 การวิเคราะห์ผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจาย โหลดแบบวิธีการแฮ็ช

ผลลัพธ์ของการทดสอบเมื่อใช้การกระจายโหลดแบบวิธีแฮ็ชนี้ จะมีผลลัพธ์ไปในแนวเดียวกับการทดสอบที่ผ่านมา คือ เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปแล้ว การเชื่อมต่อที่ถูกกระจายไปยังไฟร์วอลล์แบบรักษาสถานะตัวที่ถูกปิดจะหยุดตามไฟล์วอลล์ไปด้วย แต่เมื่อใช้วิธีที่นำเสนอแล้วการเชื่อมต่อทั้งหมดทุกการเชื่อมต่อจะสามารถดำเนินต่อไปได้เหมือนเป็นปกติ ถึงแม้จะมีไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไม่สามารถทำงานต่อได้

การกระจายโหลดโดยใช้วิธีแฮ็ชนี้ จะทำการกระจายการเชื่อมต่อโดยใช้การคำนวณทางคณิตศาสตร์กับหมายเลขไอพีแอดเดรสต้นทาง หมายเลขไอพีแอดเดรสปลายทาง และหมายเลขพอร์ตของแพ็คเก็ต ดังนั้นจึงทำให้จำนวนการเชื่อมต่อที่ถูกกระจายไปยังไฟร์วอลล์แบบรักษาสถานะแต่ละตัวอาจจะไม่ได้เป็นจำนวนครึ่งหนึ่งของจำนวนการเชื่อมต่อทั้งหมด ดังนั้นจากตารางที่ 6.4 จะเห็นว่าเมื่อทำการทดสอบโดยใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปแล้ว จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้ไม่ได้เป็นจำนวนครึ่งหนึ่งของจำนวนการเชื่อมต่อทั้งหมดที่ใช้ในการทดสอบ

6.3 การทดสอบด้านเสถียรภาพโดยใช้โปรโตคอลเทลเน็ต

การทดสอบนี้มีจุดประสงค์เพื่อแสดงให้เห็นว่าอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปในท้องตลอด ไม่สามารถทำการรักษาสถานะภาพของการเชื่อมต่อของโปรโตคอลอื่น ๆ ที่ทำงานบนโปรโตคอลทีซีพี ให้สามารถทำงานหรือดำเนินต่อไปได้บนไฟร์วอลล์แบบรักษสถานะตัวอื่นได้ ถ้าเกิดว่าไฟร์วอลล์แบบรักษาสถานะตัวที่การเชื่อมต่อเหล่านั้นถูกส่งผ่านไม่สามารถทำงานต่อ โดยการทดสอบนี้เราจะใช้โปรโตคอลเทลเน็ตในการทดสอบ ซึ่งโปรโตคอลเทลเน็ตเป็นโปรโตรคอลที่ทำงานบนโปรโตคอลทีซีพีเช่นเดียวกับโปรโตคอลเอฟทีพี

ในการทดสอบนี้เราจะทำการทดสอบโดยใช้วิธีที่เรานำเสนอเปรียบเทียบกับ เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปเหมือนกับการทดสอบที่ใช้โปรโตคอลเอฟทีพี การทดสอบนี้จะกำหนดให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปและวิธีที่นำเสนอ ใช้การกระจายโหลดแบบวนรอบสำหรับกระจายโหลดให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัว พร้อมกับทำการกำหนดให้วิธีที่นำเสนอทำการย้ายการเชื่อมต่อด้วย เมื่อตรวจพบว่ามีไฟร์วอลล์ตัวใดตัวหนึ่งหยุดทำงาน รูปที่ 6.8 แสดงการกำหนดค่าให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปใช้การกระจายโหลดแบบวนรอบ

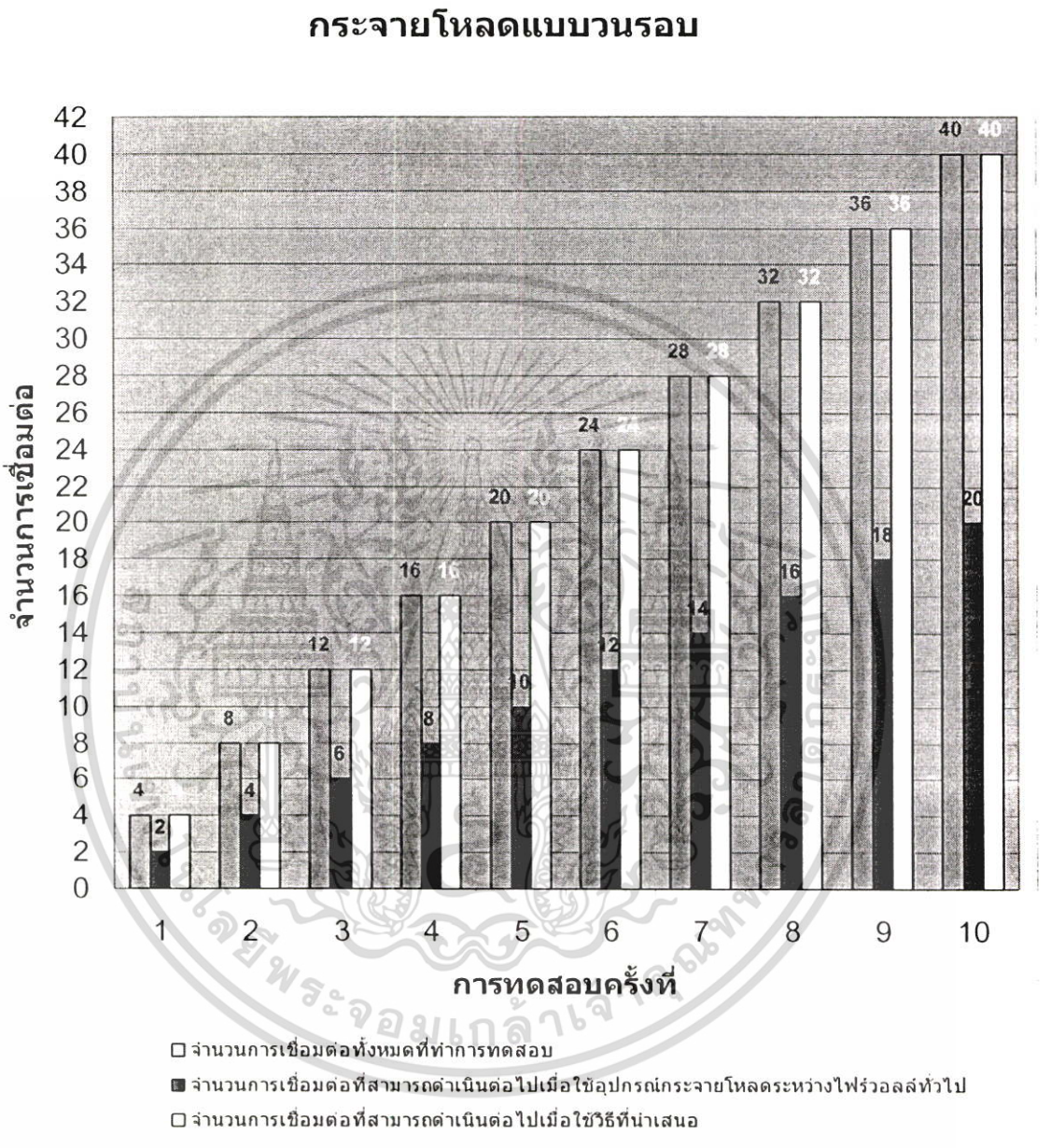
ในการทดสอบนี้เราจะทำการเชื่อมต่ออุปกรณ์ต่าง ๆ ที่ใช้ในการทดสอบ เหมือนกับการทดสอบที่ใช้โปรโตคอลเอฟทีพีดังรูปที่ 6.1 แต่เราจะทำการทดสอบโดยการสร้างการเชื่อมต่อโดย

ใช้โปรโตคอลเทลเน็ตจากเครื่องไคลเอ็นต์ไปยังเครื่องเซิร์ฟเวอร์แทน แล้วทำการสังเกตว่าเมื่อทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งแล้ว จะมีจำนวนการเชื่อมต่อที่ยังสามารถทำงานต่อได้เป็นจำนวนกี่การเชื่อมต่อ เรากำหนดให้เครื่องไคลเอ็นต์ 172.16.1.2 กับ 172.16.1.3 ทำการสร้างการเชื่อมต่อโดยใช้ โปรโตคอลเทลเน็ตไปยังเครื่องเซิร์ฟเวอร์ 172.16.2.2 และเครื่องไคลเอ็นต์ 172.16.1.4 กับ 172.16.1.5 ทำการสร้างการเชื่อมต่อโดยใช้โปรโตคอลเทลเน็ตไปยังเซิร์ฟเวอร์ 172.16.2.3 เราจะทำการทดสอบเป็นจำนวน 10 ครั้ง โดยการทดสอบแต่ละครั้งเราจะสร้างการเชื่อมต่อเป็นจำนวนไม่เท่ากัน เมื่อทำการสร้างการเชื่อมต่อจนครบตามจำนวนที่ต้องการในแต่ละการทดสอบแล้ว เราจะทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวแรก แล้วทำการนับจำนวนการเชื่อมต่อที่ยังสามารถทำงานต่อได้เมื่อทดสอบโดยใช้วิธีที่นำเสนอเปรียบเทียบกับ เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป

หลังจากทำการทดสอบตามที่ได้กล่าวมาแล้ว เราได้ผลลัพธ์ของการทดสอบดังตารางที่ 6.5 และรูปที่ 6.16 ซึ่งจะแสดงต่อไปนี้

ตารางที่ 6.5 แสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเทลเน็ต เมื่อใช้การกระจายโหลดแบบวนรอบ

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้วิธีที่นำเสนอ
4 (1)	2	4
8 (2)	4	8
12 (3)	6	12
16 (4)	8	16
20 (5)	10	20
24 (6)	12	24
28 (7)	14	28
32 (8)	16	32
36 (9)	18	36
40 (10)	20	40



รูปที่ 6.16 แสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเทลเน็ต เมื่อใช้การกระจายโหลดแบบวนรอบ

6.3.1 การวิเคราะห์ผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลเทลเน็ต เมื่อใช้การกระจายโหลดแบบวนรอบ

เมื่อทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวแรกแล้วพบว่า ถ้าเกิดเป็นการทดสอบที่ใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป การเชื่อมต่อของโปรโตคอลเทลเน็ตที่ถูกกระจายไปยังไฟร์วอลล์แบบรักษาสถานะตัวแรกหรือตัวที่ถูกปิดจะไม่สามารถดำเนินต่อไปได้ แต่ถ้าทำการทดสอบโดยใช้วิธีที่นำเสนอแล้วการเชื่อมต่อทั้งหมดทุกการเชื่อมต่อจะสามารถดำเนินต่อไปเหมือนเป็นปกติ เพราะวิธีที่นำเสนอจะสามารถทำการย้ายการเชื่อมต่อของโปรโตคอลเทลเน็ตที่ถูกส่งผ่านไฟร์วอลล์ตัวที่ถูกปิด ให้ไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวที่เหลืออยู่ได้เช่นเดียวกับการเชื่อมต่อของโปรโตคอลเอฟทีพี ดังนั้นการเชื่อมต่อของโปรโตคอลเทลเน็ตทั้งหมดจึงสามารถดำเนินต่อไปได้ การเชื่อมต่อที่หายไปเมื่อทำการทดสอบโดยใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปก็คือ การเชื่อมต่อที่ถูกกระจายไปให้ผ่านไฟร์วอลล์แบบรักษาสถานะตัวที่ถูกปิดนั้นเอง

6.4 การทดสอบด้านเสถียรภาพโดยใช้โปรโตคอลทีเอฟทีพี

การทดสอบนี้มีจุดประสงค์เพื่อทำการทดสอบว่าอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปที่มีอยู่ในท้องตลาด สามารถทำการรักษาสถานะภาพของการเชื่อมต่อของโปรโตคอลที่ทำงานบนโปรโตคอลยูดีพีได้หรือไม่ โดยในที่นี้เราจะทำการทดสอบโดยใช้โปรโตคอลทีเอฟทีพี ซึ่งเป็นโปรโตคอลหนึ่งที่ทำงานบนโปรโตคอลยูดีพี

ในการทดสอบนี้เราจะทำการทดสอบโดยใช้วิธีที่นำเสนอเปรียบเทียบกับ เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปเหมือนกับการทดสอบที่ผ่านมา การทดสอบนี้จะกำหนดให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปและวิธีที่เรานำเสนอ ใช้การกระจายโหลดแบบวนรอบสำหรับกระจายโหลดให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัว พร้อมกับทำการกำหนดให้วิธีที่นำเสนอทำการย้ายการเชื่อมต่อด้วย เมื่อตรวจพบว่ามีไฟร์วอลล์ตัวใดตัวหนึ่งหยุดทำงาน รูปที่ 6.8 แสดงการกำหนดค่าให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปให้ใช้การกระจายโหลดแบบวนรอบ

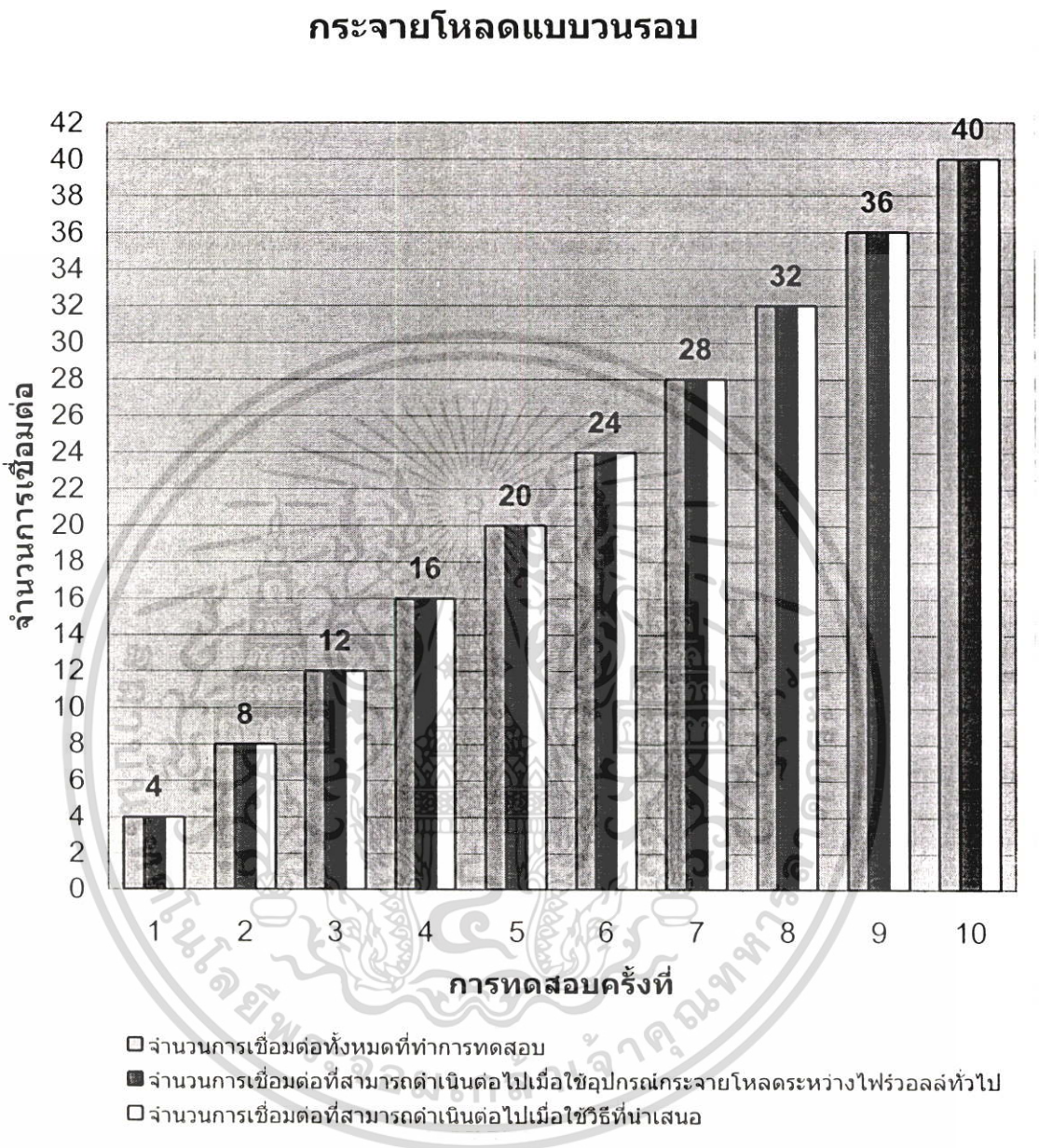
ในการทดสอบนี้เราทำการทดสอบโดยการสร้างการเชื่อมต่อโดยใช้โปรโตคอลทีเอฟทีพีจากเครื่องไคลเอ็นต์ไปยังเครื่องเซิร์ฟเวอร์ เพื่อทำการโอนย้ายไฟล์ข้อมูลขนาด 500 เมกะไบต์จากเครื่องเซิร์ฟเวอร์ไปยังเครื่องไคลเอ็นต์ แล้วทำการสังเกตว่าเมื่อทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งแล้ว จะมีจำนวนการเชื่อมต่อที่ยังสามารถทำงานต่อได้เป็นจำนวนกี่การเชื่อมต่อ เรากำหนดให้เครื่องไคลเอ็นต์ 172.16.1.2 กับ 172.16.1.3 ทำการสร้างการเชื่อมต่อโดยใช้โปรโตคอลทีเอฟทีพี เพื่อทำการโอนย้ายไฟล์ไปยังเครื่องเซิร์ฟเวอร์ 172.16.2.2 และเครื่องไคลเอ็นต์ 172.16.1.4

กับ 172.16.1.5 ทำการสร้างการเชื่อมต่อโดยใช้โปรโตคอลทีเอฟทีพี เพื่อทำการโอนย้ายไฟล์ไปยังเซิร์ฟเวอร์ 172.16.2.3 เราจะทำการทดสอบเป็นจำนวน 10 ครั้ง โดยการทดสอบแต่ละครั้งเราจะสร้างการเชื่อมต่อเป็นจำนวนไม่เท่ากัน เมื่อทำการสร้างการเชื่อมต่อจนครบตามจำนวนที่ต้องการในแต่ละการทดสอบแล้ว เราจะทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวแรก แล้วทำการนับจำนวนการเชื่อมต่อที่ยังสามารถดำเนินต่อไปได้ เมื่อทดสอบโดยใช้วิธีที่นำเสนอเปรียบเทียบกับเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป

หลังจากทำการทดสอบตามที่ได้กล่าวมาแล้ว เราได้ผลลัพธ์ของการทดสอบดังตารางที่ 6.6 และรูปที่ 6.17 ซึ่งจะแสดงต่อไปนี้

ตารางที่ 6.6 แสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลทีเอฟทีพี เมื่อใช้การกระจายโหลดแบบวนรอบ

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้วิธีที่นำเสนอ
4 (1)	4	4
8 (2)	8	8
12 (3)	12	12
16 (4)	16	16
20 (5)	20	20
24 (6)	24	24
28 (7)	28	28
32 (8)	32	32
36 (9)	36	36
40 (10)	40	40



รูปที่ 6.17 กราฟแสดงผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลทีเอฟทีพี เมื่อใช้การกระจายโหลดแบบวนรอบ

6.4.1 การวิเคราะห์ผลการทดสอบด้านเสถียรภาพด้วยโปรโตคอลทีเอฟทีพี เมื่อใช้การกระจายโหลดแบบวนรอบ

จากการทดสอบนี้จะเห็นว่าอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปที่มีอยู่ในท้องตลาด สามารถทำการรักษาสถานะภาพของการเชื่อมต่อของโปรโตคอลทีเอฟทีพีให้ดำเนินต่อไปได้ เช่นเดียวกับวิธีที่นำเสนอ สาเหตุแรกที่ทำให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปสามารถทำการรักษาสถานะภาพของการเชื่อมต่อของโปรโตคอลทีเอฟทีพีไว้ได้ก็คือ ไฟร์วอลล์แบบรักษาสถานะจะทำการตรวจสอบแพ็คเก็ตของโปรโตคอลยูดีพี (โปรโตคอลทีเอฟทีพีทำงานอยู่บนโปรโตคอลยูดีพี) กับตารางบันทึกกฎการแอคเซสทุกครั้งที่พบว่าแพ็คเก็ตนี้เป็นแพ็คเก็ตของการเชื่อมต่อที่ยังไม่มีข้อมูลอยู่ในตารางสถานะของมัน นั้นก็คือ ไฟร์วอลล์แบบรักษาสถานะจะไม่ทำการทิ้งแพ็คเก็ตของโปรโตคอลยูดีพีที่ยังไม่มีข้อมูลของการเชื่อมต่อที่แพ็คเก็ตนั้นอยู่ไปทันที แต่จะทำการตรวจสอบกับตารางบันทึกกฎการแอคเซสก่อน ถ้าเกิดมีกฎการแอคเซสที่อนุญาตการเชื่อมต่อประเภทนี้ไว้ มันก็จะทำการบันทึกข้อมูลของการเชื่อมต่อนี้ไว้ในตารางสถานะ และอนุญาตให้ทุก ๆ แพ็คเก็ตของการเชื่อมต่อนี้ผ่านตัวมันได้ทั้งสองทาง สาเหตุที่สอง คือ โปรโตคอลทีเอฟทีพีจะมีการตอบรับการได้รับข้อมูลจากเครื่องไคลเอ็นต์ไปยังเครื่องเซิร์ฟเวอร์อยู่ตลอดเวลา จึงทำให้มีแพ็คเก็ตที่มีทิศทางเดียวกับกฎการแอคเซสที่ได้อนุญาตการเชื่อมต่อประเภทนี้ไว้ (ในที่นี้เป็นแพ็คเก็ตตอบรับจากเครื่องไคลเอ็นต์ไปยังเครื่องเซิร์ฟเวอร์) ไปกระตุ้นให้ ไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลของการเชื่อมต่อนี้ลงในตารางสถานะ ซึ่งจะทำให้ไฟร์วอลล์แบบรักษาสถานะยอมอนุญาตให้แพ็คเก็ตที่ส่งจากเครื่องเซิร์ฟเวอร์ผ่านมาถึงเครื่องไคลเอ็นต์ได้ เพราะว่ามีข้อมูลของการเชื่อมต่อนี้อยู่ในตารางสถานะเรียบร้อยแล้ว ด้วยสองสาเหตุที่กล่าวมานี้รวมกันทำให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปที่มีอยู่ในท้องตลาด สามารถทำการรักษาสถานะภาพของการเชื่อมต่อที่ใช้โปรโตคอลทีเอฟทีพีให้สามารถดำเนินต่อไปได้เช่นเดียวกันกับเมื่อใช้วิธีที่นำเสนอ

6.5 การทดสอบด้านเสถียรภาพกับโปรแกรมที่ใช้โปรโตคอลยูดีพีที่สร้างขึ้นเอง

การทดสอบนี้มีจุดประสงค์เพื่อทดสอบว่าถ้าเกิดผู้พัฒนาโปรแกรมที่ใช้โปรโตคอลยูดีพี ไม่ได้กำหนดให้เครื่องไคลเอ็นต์ที่ทำการร้องขอรับข้อมูล มีการตอบรับการได้รับข้อมูลกลับไปยังเครื่องเซิร์ฟเวอร์ที่ให้บริการข้อมูลอยู่ตลอดเวลาแล้ว อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปยังจะสามารถรักษาสถานะภาพของการเชื่อมต่อของโปรแกรมหรือแอพพลิเคชั่นที่ทำงานในลักษณะนี้ให้สามารถดำเนินต่อไปได้หรือไม่

การทดสอบนี้เราได้ทำการเขียนโปรแกรมขึ้นเองโดยใช้ภาษาซี โดยโปรแกรมนี้เป็นโปรแกรมที่ใช้โปรโตคอลยูดีพีในการส่งข้อมูลระหว่างโปรแกรมในฝั่งไคลเอ็นต์กับโปรแกรมบนฝั่งเซิร์ฟ เวอร์ การทำงานของโปรแกรมนี้คือ โปรแกรมในฝั่งไคลเอ็นต์จะร้องขอเพื่อขอรับข้อมูลจาก

โปรแกรมบนฝั่งเซิร์ฟเวอร์เฉพาะตอนเริ่มต้นเท่านั้น หลังจากนั้นโปรแกรมในฝั่งไคลเอ็นต์จะรอรับข้อมูลจากเซิร์ฟเวอร์เพียงอย่างเดียว โดยไม่มีการตอบกลับใด ๆ ไปยังเซิร์ฟเวอร์ จนกว่าต้องการจะหยุดรับข้อมูล

เราทำการทดสอบโดยใช้วิธีที่นำเสนอเปรียบเทียบกับ เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปเหมือนกับการทดสอบที่ผ่านมา การทดสอบนี้จะกำหนดให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปและวิธีที่นำเสนอ ใช้การกระจายโหลดแบบวนรอบสำหรับกระจายโหลดให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัว พร้อมกับทำการกำหนดให้วิธีที่นำเสนอทำการย้ายการเชื่อมต่อด้วย เมื่อตรวจพบว่ามีไฟร์วอลล์ตัวใดตัวหนึ่งหยุดทำงาน รูปที่ 6.8 แสดงการกำหนดค่าให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ให้ใช้การกระจายโหลดแบบวนรอบ

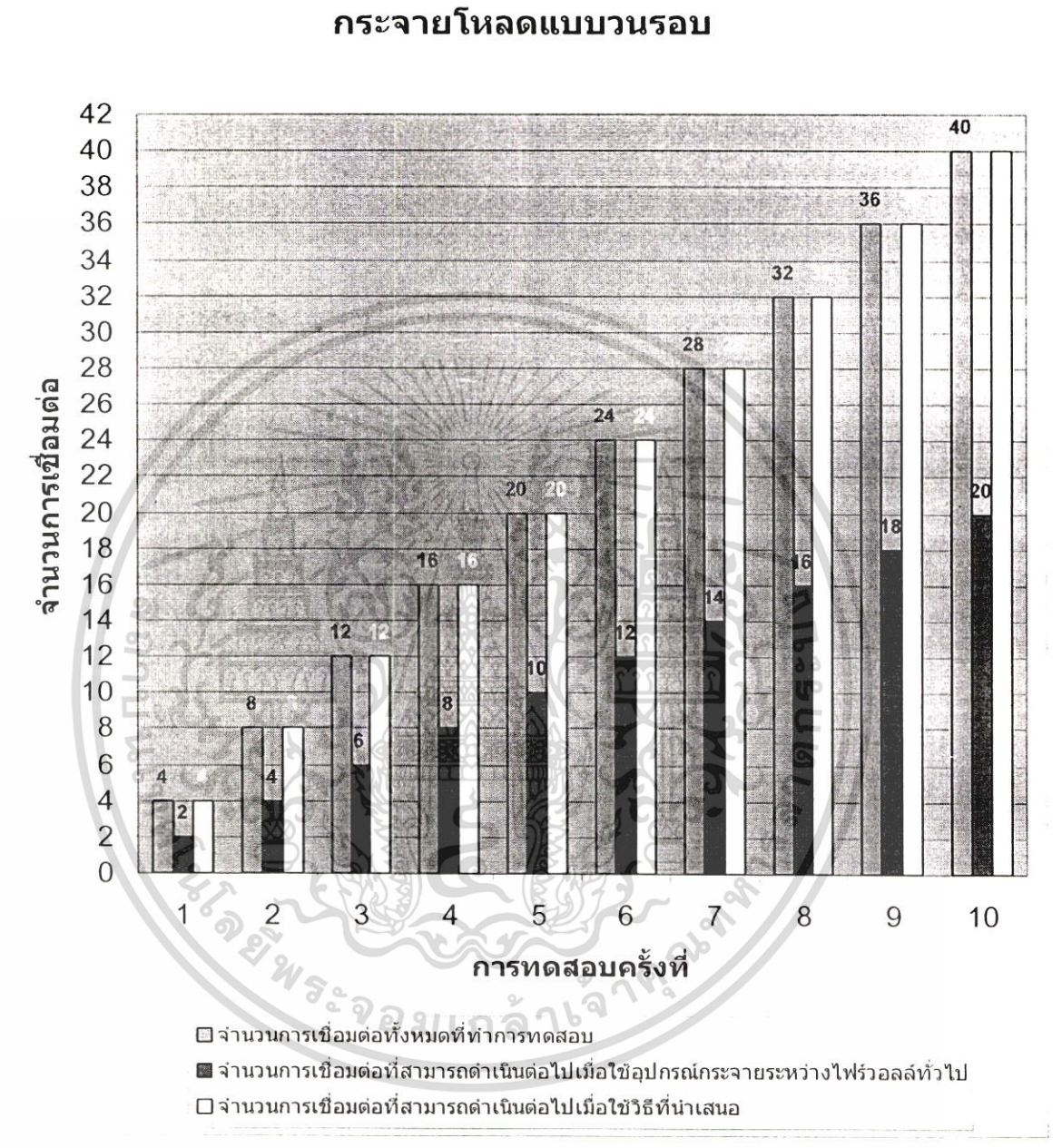
ในการทดสอบนี้เราทำการทดสอบโดยสร้างการเชื่อมต่อโดยใช้โปรแกรมที่สร้างขึ้น แล้วทำการสังเกตว่าเมื่อทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งแล้ว จะมีจำนวนการเชื่อมต่อที่ยังสามารถดำเนินต่อไปได้เป็นจำนวนกี่การเชื่อมต่อ เรากำหนดให้เครื่องไคลเอ็นต์ 172.16.1.2 กับ 172.16.1.3 ทำการสร้างการเชื่อมต่อโดยใช้โปรแกรมที่สร้างขึ้นไปยังเครื่องเซิร์ฟเวอร์ 172.16.2.2 และเครื่องไคลเอ็นต์ 172.16.1.4 กับ 172.16.1.5 ทำการสร้างการเชื่อมต่อโดยใช้โปรแกรมที่สร้างขึ้นไปยังเซิร์ฟเวอร์ 172.16.2.3 เราจะทำการทดสอบเป็นจำนวน 10 ครั้ง โดยการทดสอบแต่ละครั้งเราจะสร้างการเชื่อมต่อเป็นจำนวนไม่เท่ากัน เมื่อทำการสร้างการเชื่อมต่อจนครบตามจำนวนที่ต้องการในแต่ละการทดสอบแล้ว เราจะทำการปิดไฟร์วอลล์แบบรักษาสถานะตัวแรก แล้วทำการนับจำนวนการเชื่อมต่อที่ยังสามารถดำเนินต่อไปได้ เมื่อทดสอบโดยใช้วิธีที่นำเสนอเปรียบเทียบกับเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป

หลังจากทำการทดสอบตามที่ได้กล่าวมาแล้ว เราได้ผลลัพธ์ของการทดสอบดังตารางที่ 6.7 และรูปที่ 6.18 ซึ่งจะแสดงต่อไปนี้

ตารางที่ 6.7 แสดงผลการทดสอบด้านเสถียรภาพด้วยโปรแกรมที่ใช้โปรโตคอลยูดีพีที่สร้างขึ้นเอง เมื่อใช้การกระจายโหลดแบบวนรอบ

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป	จำนวนการเชื่อมต่อที่สามารถดำเนินต่อไปได้เมื่อใช้วิธีที่นำเสนอ
4 (1)	2	4
8 (2)	4	8
12 (3)	6	12
16 (4)	8	16
20 (5)	10	20
24 (6)	12	24
28 (7)	14	28
32 (8)	16	32
36 (9)	18	36
40 (10)	20	40

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง



รูปที่ 6.18 แสดงผลการทดสอบด้านเสถียรภาพด้วยโปรแกรมที่ใช้โปรโตคอลยูดีพีที่สร้างขึ้นเอง เมื่อใช้การกระจายโหลดแบบวนรอบ

6.5.1 การวิเคราะห์ผลการทดสอบด้านเสถียรภาพด้วยโปรแกรมที่ใช้โปรโตคอลยูดีพีที่สร้างขึ้นเอง เมื่อใช้การกระจายโหลดแบบวนรอบ

จากการทดสอบนี้จะเห็นว่าถ้าเกิดโปรแกรมที่ใช้โปรโตคอลยูดีพีไม่มีการตอบรับการได้รับข้อมูลจากโปรแกรมบนฝั่งไคลเอ็นต์ไปยังโปรแกรมที่อยู่บนฝั่งเซิร์ฟเวอร์แล้ว จะทำให้ไม่มีแพ็คเก็ตจากฝั่งไคลเอ็นต์ไปกระตุ้นให้ไฟร์วอลล์แบบรักษาสถานะตัวใหม่ที่จะทำการย้ายการเชื่อมต่อไป ให้ทำการบันทึกข้อมูลของการเชื่อมต่อเหล่านั้นลงในตารางสถานะ ดังนั้นในกรณีนี้ถ้าอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปทำการย้ายการเชื่อมต่อเหล่านั้นไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่ ทราฟฟิคของการเชื่อมต่อเหล่านั้นจะไม่สามารถผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่ได้ แต่เมื่อใช้วิธีที่นำเสนอแล้วจะสามารถทำการย้ายการเชื่อมต่อที่ใช้โปรโตคอลยูดีพีไปส่งผ่านไฟร์วอลล์แบบรักษาสถานะตัวใหม่ได้ ถึงแม้ว่าผู้พัฒนาโปรแกรมไม่ได้มีการกำหนดให้โปรแกรมบนฝั่งไคลเอ็นต์ทำการตอบรับการได้รับข้อมูลกลับไปยังโปรแกรมบนฝั่งเซิร์ฟเวอร์อยู่ตลอดเวลาก็ตาม ดังนั้นเมื่อไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งหยุดทำงาน วิธีที่นำเสนอจะสามารถรักษาสถานะภาพของการเชื่อมต่อของโปรโตคอลยูดีพีให้สามารถดำเนินต่อไปได้ ไม่ว่าแอพพลิเคชั่นหรือโปรแกรมที่ใช้โปรโตคอลยูดีพีจะทำงานในลักษณะใด

6.6 สรุปผลการทดสอบด้านเสถียรภาพ

จากผลลัพธ์ของการทดสอบที่ 6.2 จนถึง 6.5 จะเห็นว่าวิธีที่นำเสนอจะสามารถทำการย้ายการเชื่อมต่อที่ใช้โปรโตคอลทีซีพีและยูดีพีจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะตัวอื่นได้ ทำให้การเชื่อมต่อทุกการเชื่อมต่อยังสามารถดำเนินต่อไปหรือทำงานต่อไปได้เป็นปกติ ถึงแม้ว่าจะมีไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งหยุดทำงาน สำหรับอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปจะไม่สามารถทำการรักษาสถานะภาพของการเชื่อมต่อที่ใช้โปรโตคอลทีซีพี ที่ถูกกระจายไปให้ไฟร์วอลล์แบบรักษาสถานะตัวที่หยุดทำงาน ให้สามารถดำเนินต่อไปได้บนไฟร์วอลล์แบบรักษาสถานะตัวอื่นแม้แต่การเชื่อมต่อเดียว อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปยังไม่สามารถทำการรักษาสถานะภาพของการเชื่อมต่อที่ใช้โปรโตคอลยูดีพี ที่ผู้พัฒนาโปรแกรมไม่ได้มีการกำหนดให้มีการตอบรับการได้รับข้อมูลจากโปรแกรมที่อยู่บนฝั่งไคลเอ็นต์กลับไปยังโปรแกรมที่อยู่บนฝั่งเซิร์ฟเวอร์อยู่เสมอได้ โดยจะสามารถทำการรักษาสถานะภาพของการเชื่อมต่อที่ใช้โปรโตคอลยูดีพีได้เฉพาะ ในกรณีที่โปรแกรมหรือแอพพลิเคชั่นบนฝั่งไคลเอนต์มีการตอบรับการได้รับข้อมูลกลับไปยังโปรแกรมบนฝั่งเซิร์ฟเวอร์อยู่เสมอเท่านั้น

การกำหนดให้โปรแกรมหรือแอพพลิเคชั่นบนฝั่งไคลเอ็นต์ทำการตอบรับการได้รับข้อมูลกลับไปยังโปรแกรมบนฝั่งเซิร์ฟเวอร์ สำหรับโปรแกรมหรือแอพพลิเคชั่นที่ใช้โปรโตคอลยูดีพีแล้ว

จะเป็นหน้าที่ของผู้พัฒนาโปรแกรมเองว่าจะทำการเขียนโปรแกรมเพื่อทำการตอบรับการได้รับข้อมูลหรือไม่ เพราะฉะนั้นถ้าผู้พัฒนาโปรแกรมไม่ได้ทำการกำหนดให้มีการตอบรับการได้รับข้อมูลจากโปรแกรมที่อยู่บนฝั่งไคลเอ็นต์กลับไปยังโปรแกรมที่อยู่บนฝั่งเซิร์ฟเวอร์แล้ว ทำให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป จะไม่สามารถทำการย้ายการเชื่อมต่อของโปรแกรมเหล่านั้นไปยังไฟร์วอลล์แบบรักษาสถานะตัวอื่นได้ เพราะว่าจะไม่มีแพ็คเก็ตจากฝั่งไคลเอ็นต์ไปกระตุ้นให้ไฟร์วอลล์แบบรักษาสถานะตัวที่จะทำการย้ายการเชื่อมต่อไป ให้ทำการบันทึกข้อมูลของการเชื่อมต่อเหล่านั้นลงในตารางสถานะ เมื่อไม่ข้อมูลของการเชื่อมต่อเหล่านั้นอยู่ในตารางสถานะ ทราฟฟิคของการเชื่อมต่อเหล่านั้นจากโปรแกรมบนฝั่งเซิร์ฟเวอร์ก็จะไม่สามารถผ่านไฟร์วอลล์แบบรักษาสถานะไปยังโปรแกรมบนฝั่งไคลเอ็นต์ได้ เพราะว่าไม่มีกฎการแอคเซสที่อนุญาตให้ทราฟฟิคจากฝั่งเซิร์ฟเวอร์สามารถข้ามไปยังฝั่งไคลเอ็นต์ได้

6.7 การทดสอบด้านประสิทธิภาพ

การทดสอบนี้มีจุดประสงค์เพื่อแสดงให้เห็นว่า วิธีที่นำเสนอช่วยให้สามารถทำการส่งข้อมูลผ่านกลุ่มของไฟร์วอลล์แบบรักษาสถานะได้รวดเร็วขึ้น เมื่อเปรียบเทียบกับการใช้อุปกรณ์กระจาย-โหลดระหว่างไฟร์วอลล์ทั่วไปที่มีอยู่ในท้องตลาด

ในการทดสอบนี้เราจะทำการทดสอบโดยใช้โปรโตคอลเอฟทีพีซึ่งทำงานบนโปรโตคอลทีซีพีในการโอนย้ายไฟล์จากเครื่องเซิร์ฟเวอร์ไปยังเครื่องไคลเอ็นต์ เราทำการเชื่อมต่ออุปกรณ์ต่าง ๆ ที่ใช้ในการทดสอบเข้าด้วยกันดังรูปที่ 6.1 โดยเราจะกำหนดให้เครื่องไคลเอ็นต์ 172.16.1.2 ทำการโอนย้ายไฟล์ขนาด 150 เมกะไบต์ จากเครื่องเซิร์ฟเวอร์ 172.16.2.2 เครื่องไคลเอ็นต์ 172.16.1.3 ทำการโอนย้ายไฟล์ขนาด 120 เมกะไบต์ จากเครื่องเซิร์ฟเวอร์ 172.16.2.3 เครื่องไคลเอ็นต์ 172.16.1.4 ทำการโอนย้ายไฟล์ขนาด 60 เมกะไบต์ จากเครื่องเซิร์ฟเวอร์ 172.16.2.3 และเครื่องไคลเอ็นต์ 172.16.1.5 ทำการโอนย้ายไฟล์ขนาด 30 เมกะไบต์ จากเครื่องเซิร์ฟเวอร์ 172.16.2.2 เราจะทำการหาผลการทดสอบโดยการจับเวลาที่ใช้ในการโอนย้ายไฟล์ ตั้งแต่เริ่มต้นทำการโอนย้ายไฟล์แรกจนกระทั่งทำการโอนย้ายไฟล์สุดท้ายเสร็จ โดยเราจะทำการทดสอบกับอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปเปรียบเทียบกับเมื่อใช้วิธีที่นำเสนอ

6.7.1 ทดสอบโดยใช้การกระจายโหลดแบบวนรอบ

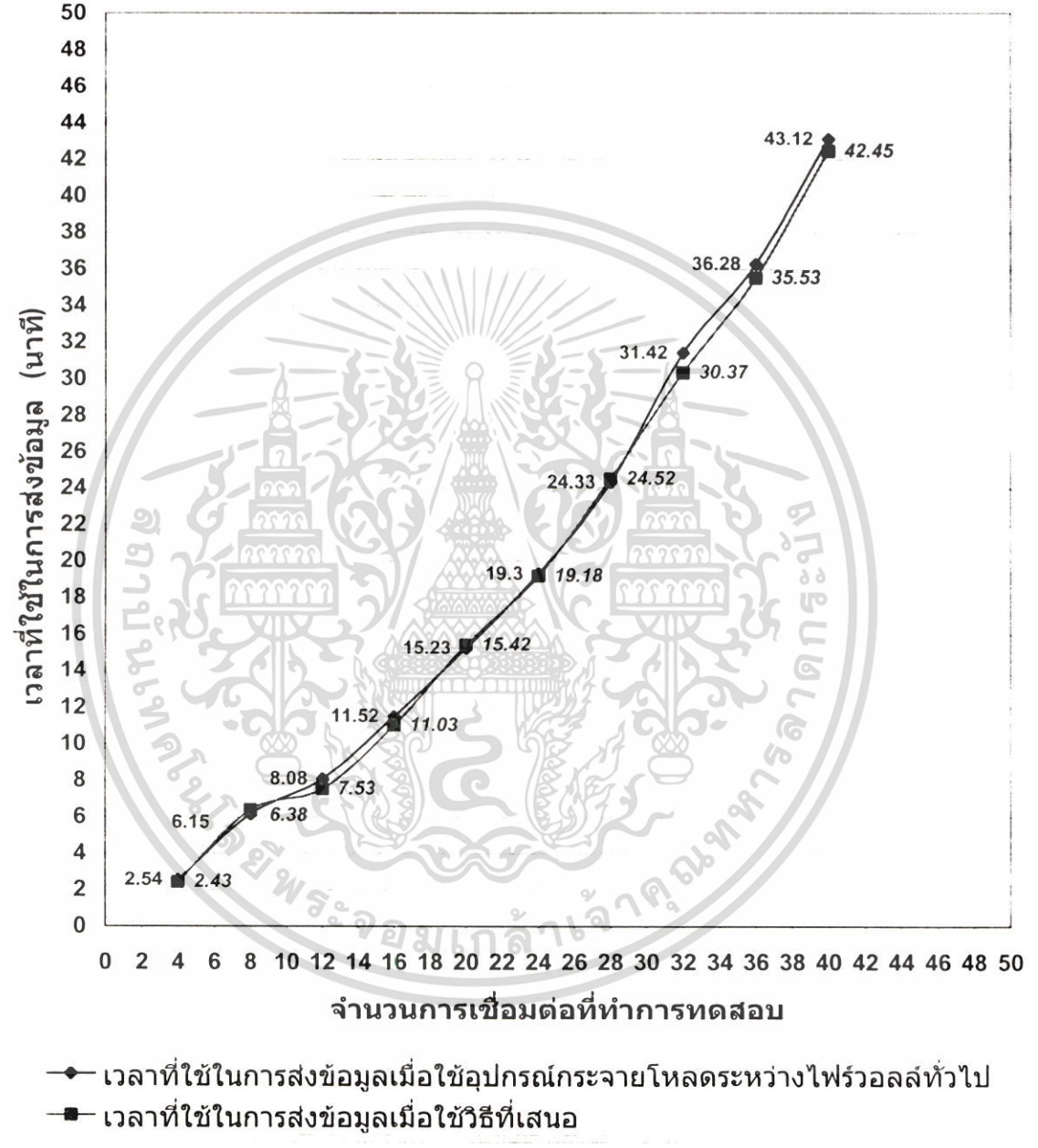
การทดสอบนี้จะใช้การกระจายโหลดแบบวนรอบสำหรับกระจายโหลดให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัว โดยเราจะทำการกำหนดให้วิธีที่นำเสนอทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งด้วย เมื่อตรวจพบว่ามีไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งมีจำนวนการเชื่อมต่อ แตกต่างกับไฟร์วอลล์แบบ

รักษาสถานะอีกตัวหนึ่งมากกว่า 1 การเชื่อมต่อ เราจะทำการเลือกการเชื่อมต่อที่จะทำการย้ายไปยังไฟร์วอลล์แบบรักษาสถานะตัวที่มีการเชื่อมต่อน้อยกว่าโดยใช้วิธีสุ่มเลือก (Random) เราทำการหาผลการทดสอบโดยการจับเวลาการโอนย้ายข้อมูลจากเครื่องเซิร์ฟเวอร์ไปยังเครื่องไคลเอ็นต์ ตั้งแต่เริ่มต้นโอนย้ายไฟล์แรกจนกระทั่งทำการโอนย้ายไฟล์สุดท้ายเสร็จ เราจะทำการทดสอบเป็นจำนวน 10 ครั้ง โดยการทดสอบแต่ละครั้งเราจะสร้างการเชื่อมต่อเป็นจำนวนไม่เท่ากัน แล้วทำการแสดงผลการทดสอบเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปเปรียบเทียบกับเมื่อใช้วิธีที่นำเสนอ

หลังจากทำการทดสอบที่ 6.7.1 แล้ว เราได้ผลลัพธ์ของการทดสอบดังตารางที่ 6.8 และรูปที่ 6.19 ซึ่งจะแสดงต่อไปนี้

ตารางที่ 6.8 แสดงผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบวนรอบ

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	เวลาที่ใช้ในการส่งข้อมูลเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป	เวลาที่ใช้ในการส่งข้อมูลเมื่อใช้วิธีที่เสนอ
4 (1)	2.54 นาที	2.43 นาที
8 (2)	6.15 นาที	6.38 นาที
12 (3)	8.08 นาที	7.53 นาที
16 (4)	11.52 นาที	11.03 นาที
20 (5)	15.23 นาที	15.42 นาที
24 (6)	19.30 นาที	19.18 นาที
28 (7)	24.33 นาที	24.52 นาที
32 (8)	31.42 นาที	30.37 นาที
36 (9)	36.28 นาที	35.53 นาที
40 (10)	43.12 นาที	42.45 นาที



รูปที่ 6.19 กราฟแสดงผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบวนรอบ

6.7.1.1 วิเคราะห์ผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบวนรอบ

จากการทดสอบนี้จะเห็นว่าวิธีที่นำเสนอทำให้สามารถส่งข้อมูลผ่านกลุ่มของไฟร์วอลล์แบบรักษาสถานะโดยเฉลี่ยได้รวดเร็วขึ้นกว่าเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป สาเหตุที่วิธีที่นำเสนอทำให้สามารถส่งข้อมูลได้รวดเร็วขึ้นก็เพราะ วิธีที่นำเสนอจะทำการปรับโหลดระหว่างไฟร์วอลล์แบบรักษาสถานะให้มีความสมดุลย์อยู่ตลอดเวลา โดยจะไม่ยอมให้มีจำนวนการเชื่อมต่อระหว่างไฟร์วอลล์แบบรักษาสถานะทั้งสองตัวแตกต่างกันเกินกว่า 1 การเชื่อมต่อ

6.7.2 ทดสอบโดยใช้การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุด

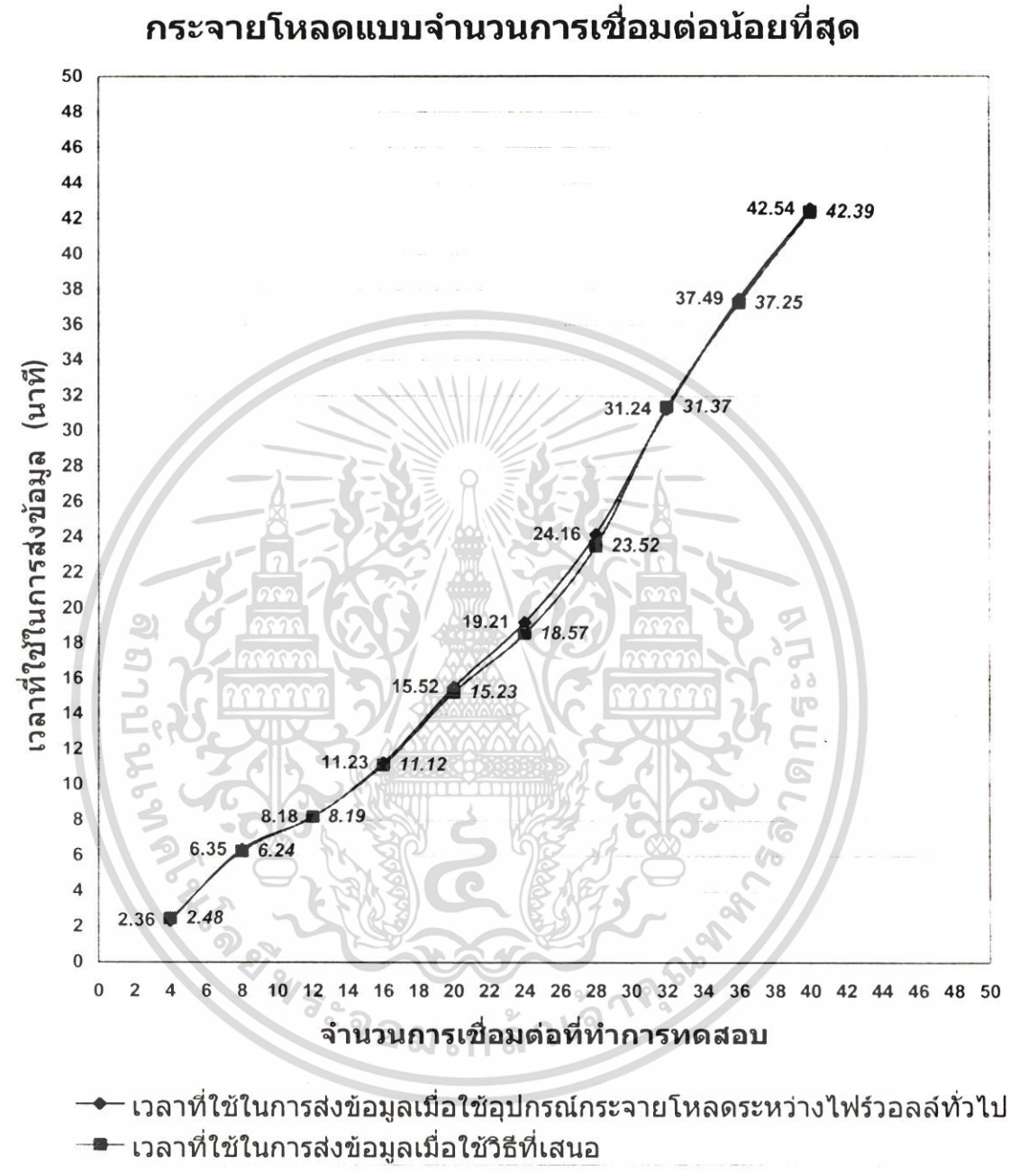
การทดสอบนี้จะใช้การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุดสำหรับกระจายโหลดให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัวแทนการกระจายโหลดแบบวน เราจะทำการกำหนดให้วิธีที่นำเสนอทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งแบบเดียวกับที่กำหนดในการทดสอบที่ 6.7.1 การทดสอบนี้จะมีขั้นตอนในการทดสอบเช่นเดียวกับการทดสอบที่ 6.7.1 จะแตกต่างกันก็ตรงที่เปลี่ยนมาใช้วิธีกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุดแทน

หลังจากทำการทดสอบที่ 6.7.2 แล้ว เราได้ผลลัพธ์ของการทดสอบดังตารางที่ 6.9 และรูปที่ 6.20 ซึ่งจะแสดงต่อไปนี้

ตารางที่ 6.9 แสดงผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุด

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	เวลาที่ใช้ในการส่งข้อมูลเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป	เวลาที่ใช้ในการส่งข้อมูลเมื่อใช้วิธีที่เสนอ
4 (1)	2.36 นาที	2. 48 นาที
8 (2)	6.35 นาที	6.24 นาที
12 (3)	8.18 นาที	8.19 นาที
16 (4)	11.23 นาที	11.12 นาที
20 (5)	15.52 นาที	15.23 นาที
24 (6)	19.21 นาที	18.57 นาที
28 (7)	24.16 นาที	23.52 นาที
32 (8)	31.24 นาที	31.37 นาที
36 (9)	37.49 นาที	37.25 นาที
40 (10)	42.54 นาที	42.39 นาที

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง



รูปที่ 6.20 กราฟแสดงผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุด

6.7.2.1 วิเคราะห์ผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุด

การทดสอบนี้ได้ผลลัพธ์เป็นทำนองเดียวกับผลลัพธ์ที่ได้จากการทดสอบที่ 6.7.1 คือ วิธีที่นำเสนอทำให้สามารถทำการส่งข้อมูลผ่านกลุ่มของไฟร์วอลล์แบบรักษาสถานะโดยเฉลี่ยแล้วได้รวดเร็วกว่าที่ใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป จากตารางที่ 6.9 และรูปที่ 6.20 จะเห็นว่ามีไม่กี่การทดสอบที่อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป สามารถทำการส่งข้อมูลผ่านกลุ่มของไฟร์วอลล์แบบรักษาสถานะได้เร็วกว่าวิธีที่นำเสนอ

6.7.3 ทดสอบโดยใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุด

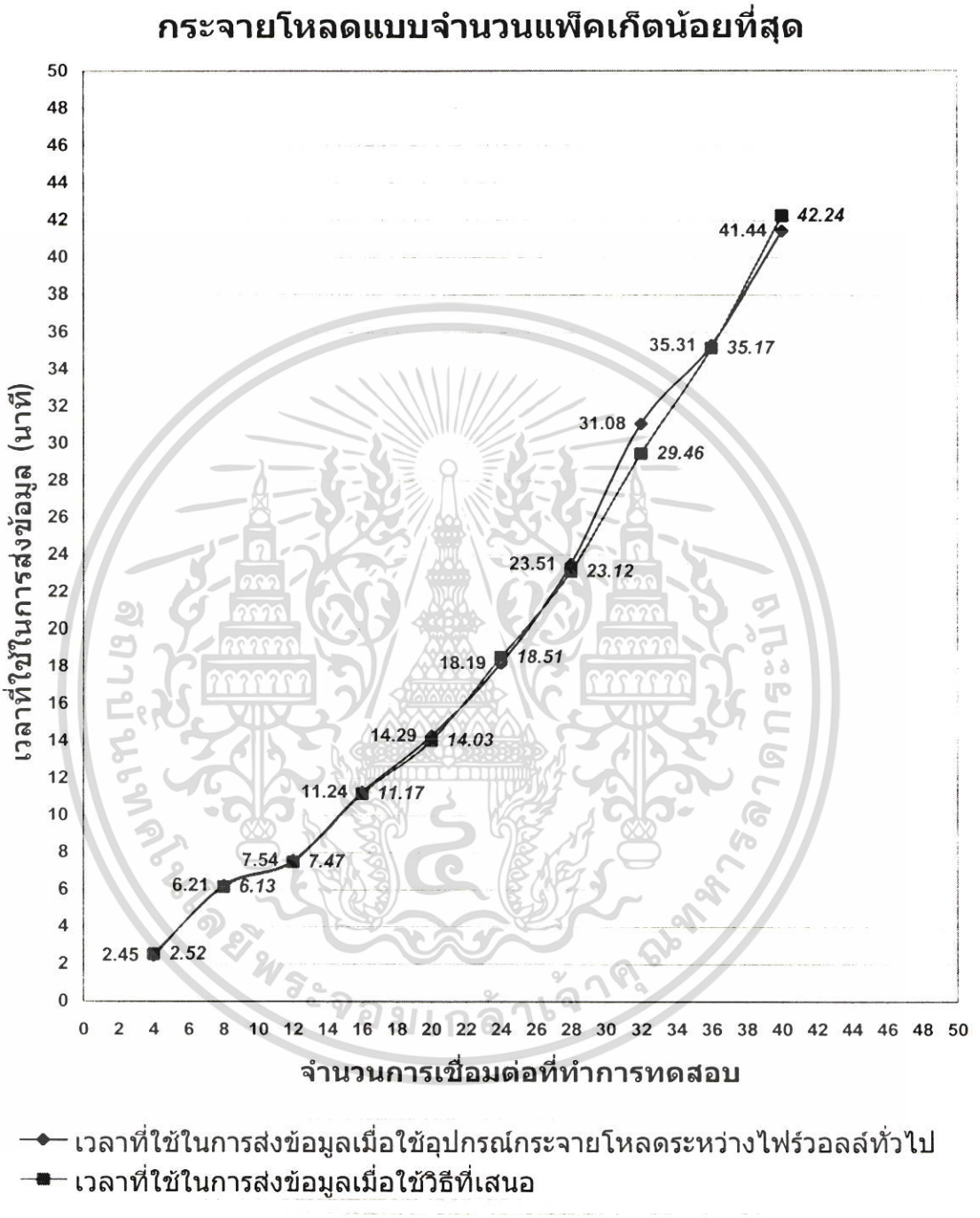
การทดสอบนี้จะใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุดสำหรับกระจายโหลดให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัว แทนการกระจายโหลดแบบวนและการกระจายโหลดแบบจำนวนการเชื่อมต่อน้อยที่สุดที่ใช้ในการทดสอบที่ 6.7.1 และ 6.7.2 ตามลำดับ ในการทดสอบนี้เราจะทำการกำหนดให้วิธีที่นำเสนอทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งเหมือนกับการทดสอบที่ผ่านมา ขั้นตอนที่ใช้ในการทดสอบนี้จะมีขั้นตอนเช่นเดียวกับการทดสอบที่ 6.7.1 และ 6.7.2 จะแตกต่างกันเพียงเปลี่ยนมาใช้วิธีกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุดแทน

หลังจากทำการทดสอบที่ 6.7.3 แล้ว เราได้ผลลัพธ์ของการทดสอบดังตารางที่ 6.10 และรูปที่ 6.21 ซึ่งจะแสดงต่อไปนี้

ตารางที่ 6.10 แสดงผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุด

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	เวลาที่ใช้ในการส่งข้อมูลเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป	เวลาที่ใช้ในการส่งข้อมูลเมื่อใช้วิธีที่เสนอ
4 (1)	2.45นาที	2. 52 นาที
8 (2)	6.21 นาที	6.13 นาที
12 (3)	7.54 นาที	7.47 นาที
16 (4)	11.24 นาที	11.17 นาที
20 (5)	14.29 นาที	14.03 นาที
24 (6)	18.19 นาที	18.51 นาที
28 (7)	23.51 นาที	23.12 นาที
32 (8)	31.08 นาที	29.46 นาที
36 (9)	35.31 นาที	35.17 นาที
40 (10)	41.44 นาที	42.24 นาที

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง



รูปที่ 6.21 กราฟแสดงผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุด

6.7.3.1 วิเคราะห์ผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุด

การกระจายโหลดโดยใช้วิธีกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุด จะเป็นวิธีที่กระจายโหลดที่ค่อนข้างมีประสิทธิภาพ เพราะการกระจายโหลดวิธีนี้จะนำโหลดที่ผ่านไฟร์วอลล์แบบรักษาสถานะแต่ละตัว ณ เวลาปัจจุบันมาใช้ในการพิจารณาหาไฟร์วอลล์แบบรักษาสถานะ ที่จะทำการกระจายการเชื่อมต่อใหม่ไปให้ ทำให้เมื่อใช้การกระจายโหลดวิธีนี้แล้ว โหลดระหว่างไฟร์วอลล์แบบรักษาสถานะแต่ละตัวค่อนข้างที่จะสมดุลย์อยู่ตลอดเวลา ดังนั้นจะเห็นว่าระยะเวลาที่ใช้ในการส่งข้อมูลผ่านกลุ่มของไฟร์วอลล์แบบรักษาสถานะ เมื่อใช้การกระจายโหลดวิธีนี้ในกรณีที่ทดสอบโดยใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป จะค่อนข้างรวดเร็วกว่าการทดสอบโดยใช้การกระจายโหลดแบบอื่น ๆ ที่ได้ทดสอบผ่านมา แต่อย่างไรก็ตามถึงแม้ว่าการกระจายโหลดด้วยวิธีนี้จะเป็นการกระจายโหลดที่ค่อนข้างมีประสิทธิภาพ แต่ก็ยังมีโอกาสที่จะเกิดเหตุการณ์ที่ไฟร์วอลล์แบบรักษาสถานะแต่ละตัวมีโหลดไม่สมดุลย์กันได้เช่นเดียวกัน โดยเหตุการณ์นี้จะเกิดขึ้นเมื่อมีการเชื่อมต่อใหม่เป็นจำนวนมากเกิดขึ้นมาในช่วงเวลาเดียวกันพร้อม ๆ กัน และบังเอิญว่าการเชื่อมต่อที่ใช้ระยะเวลาในการสื่อสารน้อยถูกกระจายไปให้ไฟร์วอลล์แบบรักษาสถานะแต่ละตัวไม่เท่ากัน โดยไฟร์วอลล์แบบรักษาสถานะบางตัวอาจได้รับการเชื่อมต่อที่ใช้ระยะเวลาในการสื่อสารน้อยเป็นจำนวนมาก และไฟร์วอลล์แบบรักษาสถานะบางตัวอาจได้รับการเชื่อมต่อที่ใช้ระยะเวลาในการสื่อสารน้อยเป็นจำนวนน้อย ดังนั้นเมื่อการเชื่อมต่อที่ใช้ระยะเวลาในการสื่อสารน้อยจบลง จะทำให้โหลดระหว่างไฟร์วอลล์แบบรักษาสถานะแต่ละตัวไม่สมดุลย์กันได้ ซึ่งเมื่อเกิดเหตุการณ์ในลักษณะนี้แล้ว วิธีที่นำเสนอจะสามารถทำการปรับโหลดระหว่างไฟร์วอลล์แบบรักษาสถานะให้สมดุลย์กันได้ ซึ่งจะทำให้สามารถส่งข้อมูลได้รวดเร็วขึ้น ดังผลลัพธ์ของการทดสอบที่แสดงในตารางที่ 6.10 และรูปที่ 6.21 จะเห็นว่าเมื่อใช้การกระจายโหลดแบบจำนวนแพ็คเก็ตน้อยที่สุดแล้ว วิธีที่นำเสนอทำให้สามารถส่งข้อมูลโดยเฉลี่ยได้เร็วกว่าเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป

6.7.4 ทดสอบโดยใช้การกระจายโหลดแบบวิธีแฮ็ช

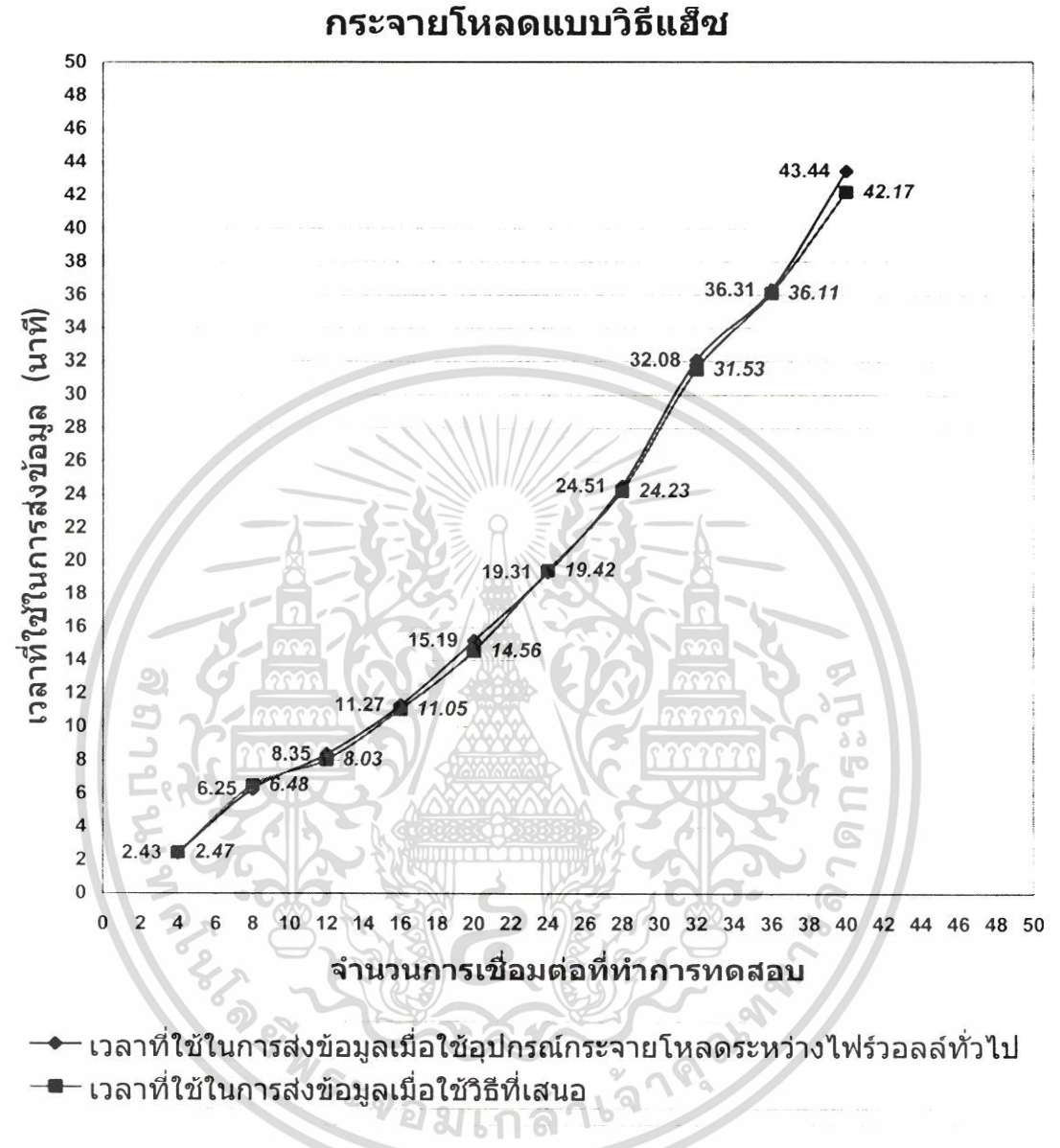
การทดสอบนี้จะใช้การกระจายโหลดแบบวิธีแฮ็ชสำหรับกระจายโหลดให้กับไฟร์วอลล์แบบรักษาสถานะแต่ละตัว โดยการแฮ็ชนี้จะใช้ทั้งหมายเลขไอพีแอดเดรสต้นทาง หมายเลขไอพีแอดเดรสปลายทาง และหมายเลขพอร์ตของแพ็คเก็ต มาใช้ในการคำนวณหาไฟร์วอลล์แบบรักษาสถานะที่จะทำการกระจายการเชื่อมต่อใหม่ไปให้ เราจะทำการกำหนดให้วิธีที่นำเสนอทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งแบบเดียวกับที่กำหนดในการทดสอบที่ผ่านมา ขั้นตอนที่ใช้ในการทดสอบนี้จะมีขั้นตอนเช่นเดียวกับการทดสอบที่ 6.7.1 , 6.7.2 และ 6.7.3 จะแตกต่างกันเพียงเปลี่ยนมาใช้วิธีกระจายโหลดแบบวิธี

แฮ็ซแทน หลังจากทำการทดสอบที่ 6.7.4 แล้ว เราได้ผลลัพธ์ของการทดสอบดังตารางที่ 6.11 และรูปที่ 6.22 ซึ่งจะแสดงต่อไปนี้

ตารางที่ 6.11 แสดงผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบวิธีแฮ็ซ

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	เวลาที่ใช้ในการส่งข้อมูลเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป	เวลาที่ใช้ในการส่งข้อมูลเมื่อใช้วิธีที่เสนอ
4 (1)	2.43นาที	2. 47 นาที
8 (2)	6.25 นาที	6.48 นาที
12 (3)	8.35 นาที	8.03 นาที
16 (4)	11.27 นาที	11.05 นาที
20 (5)	15.19 นาที	14.56 นาที
24 (6)	19.31 นาที	19.42 นาที
28 (7)	24.51 นาที	24.23 นาที
32 (8)	32.08 นาที	31.53 นาที
36 (9)	36.31 นาที	36.11 นาที
40 (10)	43.44 นาที	42.17 นาที

เทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง



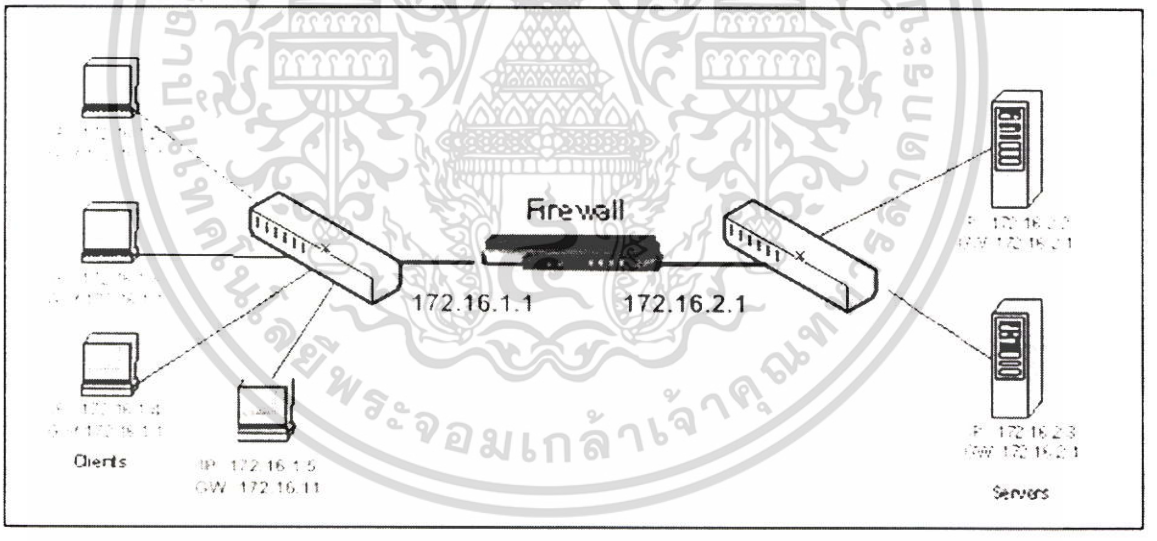
รูปที่ 6.22 กราฟแสดงผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบวิธีแฮ็ช

6.7.4.1 วิเคราะห์ผลการทดสอบด้านประสิทธิภาพด้วยโปรโตคอลเอฟทีพี เมื่อใช้การกระจายโหลดแบบวิธีแฮ็ช

การกระจายโหลดโดยวิธีแฮ็ชจะเป็นการกระจายโหลดที่ไม่ค่อยมีประสิทธิภาพมากนัก จำนวนการเชื่อมต่อที่ถูกกระจายไปให้ไฟร์วอลล์แบบรักษาสถานะแต่ละตัวเมื่อใช้การกระจายโหลดวิธีนี้ จะไม่ค่อยสม่ำเสมอและสมดุลย์ ซึ่งจากการทดสอบนี้ทำให้เห็นได้ชัดเจนว่า เมื่อทำการกระจายโหลดไม่สมดุลย์แล้ว วิธีที่นำเสนอจะช่วยทำการปรับโหลดระหว่างไฟร์วอลล์แบบรักษาสถานะให้สมดุลย์ จึงทำให้เมื่อใช้วิธีที่นำเสนอแล้วสามารถส่งข้อมูลโดยเฉลี่ยผ่านกลุ่มของไฟร์วอลล์แบบรักษาสถานะได้เร็วกว่าเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป

6.8 การวัดความเร็วในการส่งข้อมูลเมื่อไม่มีอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์

การทดสอบนี้เป็นการทดสอบเพื่อทำการวัดเวลาที่ใช้ในการโอนย้ายไฟล์จากเครื่องเซิร์ฟเวอร์ไปยังเครื่องไคลเอ็นต์ ในกรณีที่ไม่มีอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ เปรียบเทียบกับเวลาในการโอนย้ายไฟล์ เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่สามารถทำงานตามวิธีที่นำเสนอ



รูปที่ 6.23 แสดงแผนภาพเครือข่ายที่ใช้ในการทดสอบการวัดความเร็วในการส่งข้อมูล เมื่อไม่มีอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์

การทดสอบนี้เราทำการเชื่อมต่ออุปกรณ์ต่าง ๆ ที่ต้องใช้ในการทดสอบเข้าด้วยกัน ดังรูปที่ 6.23 และทำการทดสอบโอนย้ายไฟล์เหมือนกับในการทดสอบที่ 6.7 คือ จะให้มีการโอนย้ายไฟล์ 4 ขนาด จากเครื่องเซิร์ฟเวอร์ไปยังเครื่องไคลเอ็นต์ และทำการจับเวลาตั้งแต่เริ่มต้นทำการโอนย้าย

ไฟล์แรกจนกระทั่งทำการโอนย้ายไฟล์สุดท้ายเสร็จสมบูรณ์ ผลลัพธ์ของการทดสอบนี้ถูกแสดงดังตารางที่ 6.12 , 6.13 และรูปที่ 6.24 ด้านล่างนี้

ตารางที่ 6.12 แสดงระยะเวลาที่ใช้ในการโอนย้ายไฟล์เมื่อไม่มีอุปกรณ์กระจายโหลด

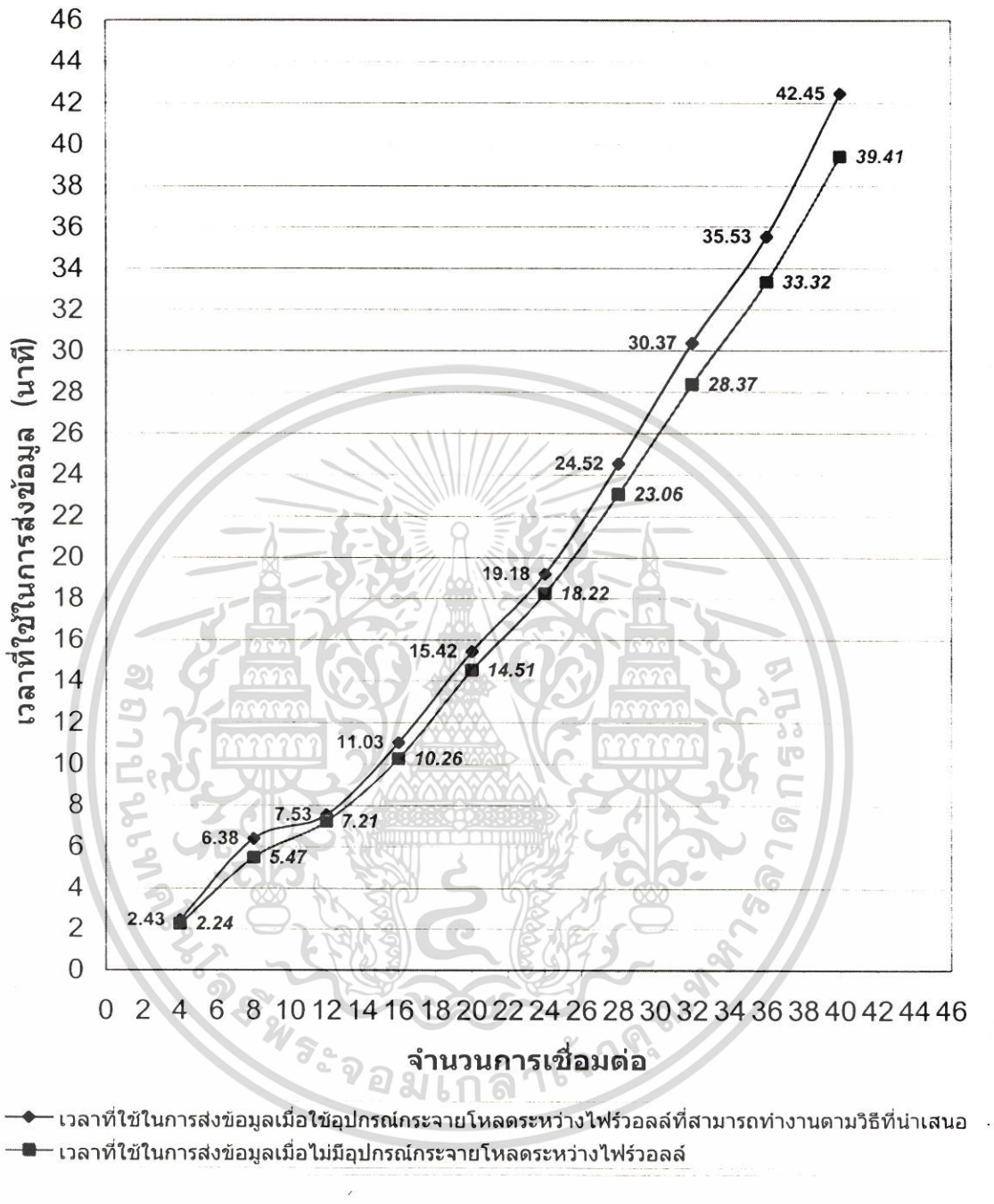
จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	เวลาที่ใช้ในการส่งข้อมูลเมื่อไม่มีอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์
4 (1)	2.24 นาที
8 (2)	5.47 นาที
12 (3)	7.21 นาที
16 (4)	10.26 นาที
20 (5)	14.51 นาที
24 (6)	18.22 นาที
28 (7)	23.06 นาที
32 (8)	28.37 นาที
36 (9)	33.32 นาที
40 (10)	39.41 นาที

เราทำการเปรียบเทียบเวลาที่ใช้ในการโอนย้ายไฟล์ ระหว่างผลลัพธ์เมื่อใช้วิธีที่นำเสนอที่ได้ทำการทดสอบไปแล้วในการทดสอบที่ 6.7.1 กับเวลาที่ใช้ในการโอนย้ายไฟล์ เมื่อไม่มีอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ดังตารางที่ 6.13 และรูปที่ 6.24 ด้านล่างนี้

ตารางที่ 6.13 แสดงการเปรียบเทียบระยะเวลาที่ใช้ในการโอนย้ายไฟล์

จำนวนการเชื่อมต่อทั้งหมดที่ทำการทดสอบ (จำนวนการเชื่อมต่อจากแต่ละไคลเอ็นต์)	เวลาที่ใช้ในการส่งข้อมูลเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ที่สามารถทำงานตามวิธีที่นำเสนอ	เวลาที่ใช้ในการส่งข้อมูลเมื่อไม่มีอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์
4 (1)	2.43 นาที	2.24 นาที
8 (2)	6.38 นาที	5.47 นาที
12 (3)	7.53 นาที	7.21 นาที
16 (4)	11.03 นาที	10.26 นาที
20 (5)	15.42 นาที	14.51 นาที
24 (6)	19.18 นาที	18.22 นาที
28 (7)	24.52 นาที	23.06 นาที
32 (8)	30.37 นาที	28.37 นาที
36 (9)	35.53 นาที	33.32 นาที
40 (10)	42.45 นาที	39.41 นาที

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง



รูปที่ 6.24 กราฟแสดงการเปรียบเทียบระยะเวลาที่ใช้ในการโอนย้ายไฟล์

6.8.1 การวิเคราะห์ผลการทดสอบการวัดความเร็วในการส่งข้อมูล เมื่อไม่มีอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์

จากการทดสอบที่ 6.8 จะเห็นว่าเวลาที่ใช้ในการโอนย้ายไฟล์ เมื่อไม่มีอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์จะน้อยกว่าเวลาที่ใช้ในการโอนย้ายไฟล์เมื่อใช้วิธีที่นำเสนอ (เปรียบเทียบกับผลลัพธ์ของการทดสอบที่ 6.7.1) สาเหตุที่เมื่อไม่มีอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์แล้วใช้เวลาในการโอนย้ายไฟล์น้อยกว่าก็เพราะว่า เนื่องจากเครื่องคอมพิวเตอร์ที่นำมาใช้ในการลงโปรแกรมที่ทำงานตามวิธีที่นำเสนอในการทดสอบที่ 6.7.1 นั้น เป็นเครื่องคอมพิวเตอร์ธรรมดาที่ใช้สำหรับทำงานทั่วไป ที่ไม่ได้ถูกออกแบบมาให้ทำงานในลักษณะนี้โดยเฉพาะ จึงทำให้เกิดการประวิงเวลาอันเนื่องมาจากการที่เครื่องคอมพิวเตอร์ต้องทำงานหลาย ๆ อย่างในเวลาเดียวกันขึ้น แต่อย่างไรก็ดีวิธีที่นำเสนอนี้เป็นวิธีที่คิดขึ้น เพื่อนำไปใช้กับอุปกรณ์ที่ถูกออกแบบให้เป็นอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์โดยเฉพาะ ซึ่งอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์แต่ละยี่ห้อจะถูกออกแบบมาให้ทำงานในลักษณะนี้โดยตรง และมีรุ่นให้เลือกใช้หลากหลายรุ่น โดยเริ่มตั้งแต่รุ่นที่สามารถรองรับทราฟฟิคได้รับเมกะบิตต่อวินาที จนถึงรุ่นที่สามารถรองรับทราฟฟิคได้ในรับหลาย ๆ กิกะบิตต่อวินาที จึงทำให้เราสามารถที่จะเลือกใช้รุ่นที่เหมาะสมกับปริมาณทราฟฟิคในระบบของเราได้ ซึ่งจะทำให้การประวิงต่ำลงจนเหมือนว่าไม่มีอุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์อยู่เลย

6.9 สรุปผลการทดสอบด้านประสิทธิภาพ

จากการทดสอบด้านประสิทธิภาพที่ผ่านมา จะเห็นว่าเมื่อใช้วิธีที่นำเสนอมาช่วยทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวที่มีจำนวนการเชื่อมต่อมาก ไปยังไฟร์วอลล์แบบรักษาสถานะตัวที่มีจำนวนการเชื่อมต่อน้อยกว่าแล้ว ทำให้เราสามารถทำการโอนย้ายข้อมูลโดยเฉลี่ยแล้วได้รวดเร็วขึ้นกว่าเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป โดยวิธีที่นำเสนอจะไม่ยอมให้จำนวนการเชื่อมต่อที่ถูกส่งผ่านไฟร์วอลล์แบบรักษาสถานะแต่ละตัว มีจำนวนแตกต่างกันมากกว่า 1 การเชื่อมต่อ โดยถ้ามีจำนวนการเชื่อมต่อแตกต่างกันมากกว่า 1 การเชื่อมต่อเมื่อไรแล้ว วิธีที่นำเสนอจะทำการย้ายการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวที่มีจำนวนของการเชื่อมต่อมากกว่า ไปยังไฟร์วอลล์แบบรักษาสถานะตัวที่มีจำนวนของการเชื่อมต่อน้อยกว่าให้ ทำให้โหลดหรือการเชื่อมต่อระหว่างไฟร์วอลล์แบบรักษาสถานะมีความสมดุลย์กันอยู่ตลอดเวลา

เนื่องจากวิธีที่นำเสนอสามารถทำการย้ายโหลดระหว่างไฟร์วอลล์แบบรักษาสถานะ เมื่อไฟร์วอลล์แบบรักษาสถานะมีโหลดแตกต่างกันมากกว่า 1 การเชื่อมต่อได้ ทำให้วิธีที่นำเสนอมีความสามารถในการจัดการและควบคุมการเชื่อมต่อที่ผ่านกลุ่มของไฟร์วอลล์แบบรักษาสถานะได้มีประสิทธิภาพมากกว่า เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไป ที่ไม่สามารถทำ

การย้ายโหลดระหว่างไฟร์วอลล์แบบรักษาสถานะ หลังจากได้ทำการตัดสินใจกระจายโหลดไปให้ไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งแล้ว

บทที่ 7
สรุปผลการวิจัยและข้อเสนอแนะ

7.1 สรุปผลการวิจัย

วิทยานิพนธ์ฉบับนี้ได้ทำการเพิ่มความสามารถให้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ ให้สามารถทำการย้ายโหลดหรือการเชื่อมต่อจากไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งไปยังไฟร์วอลล์แบบรักษาสถานะอีกตัวหนึ่งได้ ทั้งในกรณีที่พบว่ามีไฟร์วอลล์แบบรักษาสถานะตัวใดตัวหนึ่งไม่สามารถทำงานต่อได้ และในกรณีที่มีไฟร์วอลล์แบบรักษาสถานะตัวหนึ่งมีโหลดมากกว่าไฟร์วอลล์แบบรักษาสถานะตัวอื่น ๆ เป็นจำนวนมาก วิธีการที่นำเสนอนี้จะสามารถทำการย้ายโหลดระหว่างไฟร์วอลล์แบบรักษาสถานะได้ โดยไม่คำนึงว่าไฟร์วอลล์แบบรักษาสถานะแต่ละตัวจะมาจากผู้ผลิตเดียวกันหรือไม่ และไม่คำนึงว่าไฟร์วอลล์แบบรักษาสถานะแต่ละตัวสามารถทำการแลกเปลี่ยนข้อมูลในตารางสถานะระหว่างกันได้หรือไม่

วิธีการที่นำเสนอจะสามารถทำการย้ายโหลดหรือย้ายการเชื่อมต่อระหว่างไฟร์วอลล์แบบรักษาสถานะที่มาจากต่างผู้ผลิตกัน และไฟร์วอลล์แบบรักษาสถานะที่ไม่สามารถทำการแลกเปลี่ยนข้อมูลในตารางสถานะระหว่างกันได้ โดยใช้วิธีการสร้างแพ็คเก็ตจำลองเพื่อหลอกให้ไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลของการเชื่อมต่อที่กำลังจะถูกย้ายมาลงในตารางสถานะของตนเสียก่อน เมื่อไฟร์วอลล์แบบรักษาสถานะมีข้อมูลของการเชื่อมต่อที่กำลังจะถูกย้ายมาในตารางสถานะเรียบร้อยแล้ว มันก็จะอนุญาตให้ทราฟฟิคของการเชื่อมต่อนั้นผ่านตัวมันได้ แต่ถ้าไม่ทำการหลอกให้ไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลลงในตารางสถานะเสียก่อนแล้ว ไฟร์วอลล์แบบรักษาสถานะก็จะไม่มีข้อมูลของการเชื่อมต่อนั้น และจะไม่อนุญาตให้ทราฟฟิคของการเชื่อมต่อนั้นผ่าน

แพ็คเก็ตจำลองที่ใช้สำหรับหลอกให้ไฟร์วอลล์แบบรักษาสถานะทำการบันทึกข้อมูลของการเชื่อมต่อลงในตารางสถานะนั้น ถ้าเป็นการเชื่อมต่อที่ใช้โปรโตคอลทีซีพีแล้วเราจะจำลองแพ็คเก็ตประเภท SYN แต่ถ้าเป็นการเชื่อมต่อที่ใช้โปรโตคอลยูดีพีเราจะจำลองแพ็คเก็ตธรรมดา แต่ต้องระบุว่าแพ็คเก็ตที่จำลองขึ้นนี้เป็นแพ็คเก็ตของโปรโตคอลยูดีพี เฮดเดอร์ของแพ็คเก็ตที่จำลองขึ้นในส่วนของ ไอพีแอดเดรสต้นทาง , ไอพีแอดเดรสปลายทาง , พอร์ตต้นทาง และพอร์ตปลายทาง จะต้องตรงกันกับข้อมูลที่อยู่ในแพ็คเก็ตของการเชื่อมต่อที่กำลังจะทำการย้าย

เราได้นำวิธีที่นำเสนอมาทำการทดสอบในด้านเสถียรภาพ เปรียบเทียบกับการทดสอบเมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปที่มีอยู่ในท้องตลาดยี่ห้อหนึ่ง โดยการทดสอบนี้จะเราทำการทดสอบกับการเชื่อมต่อประเภทต่าง ๆ ดังนี้ การเชื่อมต่อของโปรโตคอลเอฟทีพี การเชื่อมต่อของโปรโตคอลเทลเน็ต การเชื่อมต่อของโปรโตคอลทีเอฟทีพี และการเชื่อมต่อที่สร้างจาก

ปัญหาที่สอง คือ การทดสอบเพื่อทำการเปรียบเทียบวิธีที่นำเสนอกับ เมื่อใช้อุปกรณ์กระจาย-โหลดระหว่างไฟล์วอลล์ทั่วไปที่มีในท้องตลาด ต้องทำการทดสอบคนละครั้งกันทำให้ไม่สามารถนำผลลัพธ์ของการทดสอบมาทำการเปรียบเทียบกันได้แม่นยำร้อยเปอร์เซ็นต์

ปัญหาที่พบปัญหาสุดท้าย คือ ความสามารถในการประมวลผลของไฟร์วอลล์แบบรักษาสถานะที่นำมาทดสอบค่อนข้างสูง และจำนวนการเชื่อมต่อที่สามารถทำการสร้างเพื่อทำการทดสอบมีจำนวนไม่มากนัก จึงทำให้ไม่สามารถแสดงความแตกต่างระหว่างเมื่อใช้วิธีที่เสนอกับ เมื่อใช้อุปกรณ์กระจายโหลดระหว่างไฟร์วอลล์ทั่วไปได้อย่างชัดเจนตามที่คาดหวังไว้

7.3 ข้อเสนอแนะ

การเลือกการเชื่อมต่อที่จะทำการย้ายจากไฟร์วอลล์แบบรักษาสถานะตัวที่มีโหลดมากไปยังไฟร์วอลล์แบบรักษาสถานะตัวที่มีโหลดน้อยกว่า ในวิทยานิพนธ์นี้จะใช้วิธีการเลือกแบบสุ่มเลือก (Random) ซึ่งวิธีนี้ยังไม่ได้เป็นวิธีที่ดีที่สุด วิทยานิพนธ์ฉบับนี้มีจุดประสงค์เพียงแค่ต้องการแสดงให้เห็นว่า สามารถทำการย้ายโหลดระหว่างไฟร์วอลล์แบบรักษาสถานะได้ แต่ยังไม่ได้คิดหาวิธีที่ดีที่สุดสำหรับเลือกการเชื่อมต่อที่จะทำการย้าย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารอ้างอิง

[1] Held, G. **Assembly Instructions Included (Cisco Routers).** *Network Magazine*. January 2001.

[2] Meyer, A. **Building A Floppy Firewall.** January 2001.

[3] Chapman, D. B. and Zwicky, E. D. **Building Internet Firewalls – 2nd Edition.** *O'Reilly*. 2000.

[4] Sonnenreich, W. and Yates, T. **Building Linux and OpenBSD Firewalls**. *Wiley*. 2000.

[5] Shipley, G. **Cisco IOS: It's Not Just for Routing Anymore.** *Network Computing*. May 31. 1999.

[6] **Cisco IOS 12 Network Security.** *Cisco Press/Macmillan Technical Publishing*.1999.

[7] Held, G. and Hundley, K. **Cisco Security Architectures**. *McGraw-Hill*. 1999.

[8] Graham, R. **Decipher Your Firewall Logs**. Mar/Apr 2000.

[9] Farrow, R. **Firewall Configuration Done Right.** *Network Magazine*. December 1998.

[10] Farrow, R. **Firewall Vulnerabilities.** *Network Magazine*. August 1999

[11] Strebe, M. and Perkins, C. **Firewalls 24Seven.** *Sybex Network Press*. 1999

[12] Goncalves, M. **Firewalls Complete.** *McGraw-Hill*. 1998

[13] B. Cheswick & S. Bellovin. "Firewalls & Internet Security - Repelling the Wiley Hacker". 1998

[14] M. Lucas . "FreeBSD Firewall Tools & Techniques". June 2000.

[15] L. Boyer . "Great Walls of Fire (Firewall Security)" *NetWare Connection*.January 1997.

[16] M. Fratto. "The 'Ins' and 'Outs; of Firewall Security" *Network Computing Guide to Firewall Selection and Policy – Draft - 10/11/2001*. September 6, 1999.

[17] K. Siyan . "Internet Firewalls & Network Security - Second Edition" *New Riders Publishing*. 1996

[18] NIST. "Keeping Your Site Comfortably Secure: An Introduction to Internet Firewalls" *Special Publication 800-10.*

[19] C. Sample. "Kicking Firewall Tires" *Network Magazine*. March 1998

[20] M. Gagne. "A Linux Internet Gateway. Sys Admin". June 2000

[21] R. McCarty. " NAT: Network Address Translator. Sys Admin". March 2000

[22] R. McCarty. "Packet Filtering and Cisco's Way. Sys Admin". May 1999

[23] G. Held. "Router-Based Network Defense. Sys Admin". March 2000

[24] M. Sustic. "The Use of Routers in Firewall Setup. Sys Admin". May 2000

[25] Radware. "FireProof Application Switch I & Application Switch II Software Version: 2.61". 2004.

[26] Radware. "FireProof Sales information guide". 2002.

[27] D. E. Comer. "Internetworking with TCP/IP Volume I: Principles, Protocol, and Architecture. Thied Edition" *Prentice Hall*, Inc. 1995.

[28] K. Washburn and J. T. Evans. "TCP/IP: Running a Successful Network" *Addison Wesley Publishing Company*. 1993.

[29] R. Putthacharoen , P. Bunyatnopparat . "Lossless high availability for multi-vendor stateful firewalls" *International Conference on Integration of Science & Technology for Sustainable Developmen (STIC)*, KMITL, Bangkok, Thailand, August 2004. pp.283-286

[30] ลัญฉกร วุฒิสิทธิกุลกิจ. **โครงข่ายอินเทอร์เน็ต และโพรโตคอลทีซีพี/ไอพี.** สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย. 2545.

[31] เรืองไกร รังสิพล. **เจาะระบบ TCP/IP จุดอ่อนของโปรโตคอลและวิธีป้องกัน.** บริษัท โปรวิชั่น จำกัด 2544.

[32] พิพัฒน์ หิรัณย์วณิชชากร. **ระบบการสื่อสารข้อมูลและเครือข่ายคอมพิวเตอร์.** บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) 2542.

[33] สุรศักดิ์ สงวนพงษ์. **สถาปัตยกรรมและโปรโตคอลทีซีพี/ไอพี.** บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) 2543.

ภาคผนวก

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคผนวก ก.
ตัวอย่างซอร์สโค้ดของวิธีที่นำเสนอ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ก.1 โปรแกรมหลัก ทำหน้าที่ในการ initialize โปรแกรมและสร้าง thread ย่อย ๆ ขึ้นมา

```
int main(void)
{
  init_session(); //สร้างบัฟเฟอร์เพื่อเก็บเซสชัน
  pthread_t id1,id2,id3,id4,id5,id6,id7,id8; //ตัวแปรสำหรับ thread

  // ตัวแปร mutex สำหรับ synchronize thread
  pthread_mutex_init(&fwsession_mutex,NULL);
  pthread_mutex_init(&fwbyte_mutex,NULL);
  pthread_mutex_init(&fwpacket_mutex,NULL);
  pthread_mutex_init(&sessioninfo_mutex,NULL);

  local_ip=getIPAddress(local_dev); //อ่าน IP address ของ local interface
  remote_ip=getIPAddress(remote_dev); //อ่าน IP address ของ remote interface

  local_mac=getMacAddressStr(local_dev); //อ่าน Mac address ของ local interface
  remote_mac=getMacAddressStr(remote_dev); //อ่าน Mac address ของ remote interface

  pthread_create(&id1,NULL,&local,NULL); //สร้าง thread สำหรับ packet ขา local interface
  pthread_create(&id2,NULL,&remote,NULL); //สร้าง thread สำหรับ packet ขา remote interface

 //สร้าง thread สำหรับตรวจสอบ firewall
  pthread_create(&id3,NULL,&sendPing,NULL);
  pthread_create(&id4,NULL,&receivePing,NULL);

  pthread_create(&id6,NULL,&calculatePacketRate,NULL); //thread สำหรับคำนวณ packet rate
  pthread_create(&id7,NULL,&calculateByteRate,NULL); //thread สำหรับคำนวณ bit rate
  pthread_create(&id8,NULL,&loadBalance,NULL); //thread สำหรับ load balance

  printf("Load Balance Server Start.\n");
  while(1) ;
  return 0;
}
```

ก.2 ฟังก์ชันสำหรับสร้าง SYN แพ็คเก็ต

```
void createSync(int sid,int fwid)
{
   u_char *cp; libnet_t *l;  libnet_ptag_t t;            //สร้างตัวแปลสำหรับ libnet
   char *payload="AAAA";  u_short payload_s=4;    //สร้าง payload และขนาด payload
   char errbuf[LIBNET_ERRBUF_SIZE];              //สร้าง buffer สำหรับเก็บ error
   l = libnet_init(LIBNET_LINK,remote_dev,errbuf); //สร้าง libnet descriptor สำหรับส่งข้อมูล
   unsigned short sport=sessioninfo[sid].sport ; unsigned short dport=sessioninfo[sid].dport;
   unsigned int sip=htonl(sessioninfo[sid].sip); unsigned int dip=htonl(sessioninfo[sid].dip);
   //ใส่ TCP header
   t = libnet_build_tcp( sport,dport,0x01010101,0x02020202,TH_SYN,32767,
      0, 0,LIBNET_TCP_H + payload_s, payload, payload_s,l,0);
   t = libnet_build_ipv4(LIBNET_IPV4_H + LIBNET_TCP_H +  payload_s,0, 242, 0, 64,
                IPPROTO_TCP, 0, sip, dip, NULL, 0, l, 0);
   char *smc=sessioninfo.smac[sid]; char *dmc= sessioninfo.dmac[sid];
   //ใส่ Ethernet header
   t = libnet_build_ethernet( dmc, smc, ETHERTYPE_IP, NULL, 0, l, 0);
   int c = libnet_write(l); //ส่ง packet
   libnet_destroy(l);        //เคลียร์ descriptor
}
```

ก.3 ฟังก์ชั่นสำหรับทำการกระจายการเชื่อมต่อ

```
void *loadBalance(void *arg)
{
    int i,j,k;
    while(1)
    {
        pthread_mutex_lock(&fwsession_mutex);
        for(j=0;j<fwnum;j++)
        {
            if( fwstatus[j]<=0)  // กรณี firewall j down
            {
                for(i=0;i<MAX_SESSION;i++)
                {
                   if( (sessioninfo[i].flag==1)&&(sessioninfo[i].fwid==j)) //กรณี session i ใช้ firewall j
                   {
                       k=getBestFirewall(); //หา firewall ที่จะย้าย session ไป
                       moveSession(i,k);//ย้าย session i ไปยัง firewall k
                   }
                }
            }
        }
        balance(); //ทำ load balance firewall
        pthread_mutex_unlock(&fwsession_mutex);
        usleep(100000);
}
```

ก.4 ฟังก์ชั่นในการสร้างการเชื่อมต่อใหม่

```
Int createSession(u_char *pk,u_char *ip_pk,u_char *tcp_pk,int stype)
{
   int i;
   u_int sip=getSourceIP(ip_pk);
   u_int dip=getDestinationIP(ip_pk);
   u_short sport=getTCPSourcePort(tcp_pk);
   u_short dport=getTCPDestinationPort(tcp_pk);
   int sid=getSession(sip,dip,sport,dport);
   if(sid==-1)
   {
    for(i=0;i<MAX_SESSION;i++)
    {
      if(sessioninfo[i].flag==0) break;
    }
    pthread_mutex_lock(&fwsession_mutex);
    pthread_mutex_lock(&fwbyte_mutex);
    pthread_mutex_lock(&fwpacket_mutex);
    switch(algorithm) // session ใหม่จะถูกสร้างโดย firewall จะถูกเลือกตาม algorithm ที่ตั้งไว้
    {
          case 1:sessioninfo[i].fwid=fwHashIndex(sip,dip);break;
          case 2:sessioninfo[i].fwid=fwPacketCountIndex(sip,dip);break;
          case 3:sessioninfo[i].fwid=fwByteCountIndex(sip,dip);break;
          case 4:sessioninfo[i].fwid=fwSessionCountIndex(sip,dip);break;
          default:sessioninfo[i].fwid=fwRoundRobinIndex();break;
    }

     sessioninfo[i].stype=stype;
     sessioninfo[i].sip=sip;
     sessioninfo[i].dip=dip;
     sessioninfo[i].sport=sport;
     sessioninfo[i].dport=dport;
     strncpy(sessioninfo[i].smac,getSourceMac(pk),17);
     strncpy(sessioninfo[i].dmac,getDestMac(pk),17);
     sessioninfo[i].flag=1;
     fwsession[sessioninfo[i].fwid]++;
     pthread_mutex_unlock(&fwpacket_mutex);
     pthread_mutex_unlock(&fwbyte_mutex);
     pthread_mutex_unlock(&fwsession_mutex);

     return i;
   }
   return -1;

}
```

ก.5 ฟังก์ชั่นสำหรับคำนวณอัตราการส่งแพ็คเก็ต

```
void *calculatePacketRate(void *arg)
{
        int pk,i;
        while(1)
        {
                pthread_mutex_lock(&fwpacket_mutex);
                for(i=0;i<fwnum;i+)
                {
                    pk=fwpacket[i]-oldfwpacket[i];
                    oldfwpacket[i]=fwpacket[i];
                }
                 pthread_mutex_unlock(&fwpacket_mutex);
                sleep(1);
        }
}
```

ภาคผนวก ข.

ผลงานวิจัยในระหว่างการศึกษาที่ได้รับการตีพิมพ์เผยแพร่

Pro. of the 1st KMITL International Conference on Integration of Science & Technology
for Sustainable Development Bangkok. Thailand 25-26 August 2004. Vol.1: 283-286.

Lossless high availability for multi-vendor stateful firewalls

Rattipong Putthacharoen[1] and Assc. Prof. Pratheep Bunyatnopparat[2]

[1]King Mongkut's Institute of Technology Ladkrabang
Chalongkrung Road, Bangkok 10520, Thailand
E-mail : rattipong@yahoo.com,rattipong@wit.co.th

[2]King Mongkut's Institute of Technology Ladkrabang
Chalongkrung Road, Bangkok 10520, Thailand
E-mail : pratheep@ce.kmitl.ac.th

ABSTRACT

This article proposes the new algorithm for the Firewall Load balancer in order to provide an excellent high availability for firewalls. If a Firewall on the network fails , the proposed algorithm will allow the Firewall Load balancer to send remaining traffics of running connections via the other stateful firewalls even though those stateful firewalls come from multi-vendors. Without this algorithm , the running connections will die and clients have to make new connections in order to retransmit every packets again.

Keywords

Firewall Load balancer , Stateful firewall , High availability.

1. INTRODUCTION

Nowadays stateful firewalls are the major security device that network administrators deployed in their network in order to protect hackers looking to cause damage to or misuse resources of a particular area.

Because stateful firewalls sit on the data path, they can limit network performance and scalability. They also represent single points-of-failure that degrade network resource availability. While most firewalls can be deployed with commercially-available high-availability but none firewalls from any vendor can synchronize its session table with the other vendor firewalls.

The Firewall Load balancer provides scalability and fault tolerance for firewalls by automatically directing user traffic to only those firewalls that are functioning properly. It can mange multiple firewalls in active/active configuration but it's still not excellent yet. The Firewall load balancer can not provide the lossless high availability while a firewall fails.

In order to provide the lossless high availability, we propose the new algorithm for the Firewall load balancer which extremely leverages the efficiency of a high availability.

2. FIREWALL LOAD BALANCER

The Firewall Load balancer is a device that provides load balancing, optimization, and high availability for firewalls, within your network. Using an advanced array of built-in load balancing algorithms that monitor the number of clients and the traffic load, the Firewall Load balancer dynamically distributes traffic evenly across the various firewalls, while taking into account both inbound and outbound traffic. The Firewall Load balancer directly interfaces with multi-vendor firewalls in order to provide comprehensive load balancing and traffic redirection while achieving high availability and instantaneous failover if a failure should occur. Because the Firewall Load balancer performs load balancing and traffic management to the most available resource, security resources are always optimized and maximum performance is achieved. Some of populated Load balancing algorithms are shown below:

Cyclic - Traffic is distributed in round robin fashion.

Weight - Traffic is distributed based on the weight of firewalls.

Pro. of the 1st KMITL International Conference on Integration of Science & Technology
for Sustainable Development Bangkok. Thailand 25-26 August 2004. Vol.1: 283-286.

Least number of users - A new session is sent to the firewall with the least number of users.

Least amount of packets - A load balancing decision is made based on the least number of packets.

Least amount of bytes – Optimization decision is made based on the least number of bytes.

Hashing dispatch method - Optimization of each security device is achieved using a hash function to select the appropriate resource for each new session. The input for the hash function is determined by the Client Table mode according to source and destination IP. This algorithm is particularly useful for high traffic environments where traffic management not load balancing is the primary concern.

In additional to provide as above, the Firewall Load balancer is also to keep track of all sessions on each of firewalls. This is called session persistence. It is critical that the Firewall Load balancer keeps track of all sessions because the reply packets must return via the same firewall, otherwise they will be discarded by the other firewalls. Maintaining session persistence , it is necessary to implement two Firewall Load balancers as in Figure 1.

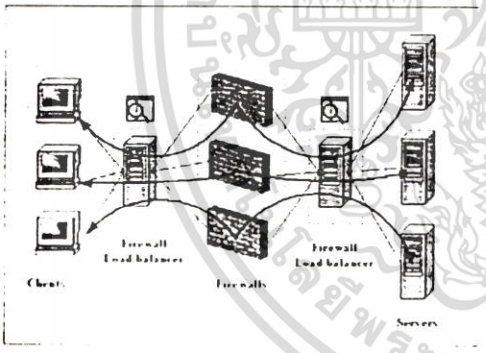


Figure 1 Implementing Firewall Load balancers

3. STATEFUL FIREWALL

Stateful packet inspection firewalls are based on the filtering of packets at the network level examining packet header fields: source IP address, destination IP address, source ports, and destination ports. This type of firewall tracks outgoing requests and only allows responses to those requests back through the firewall. This way, attempts to access the internal network that have not been requested by the internal network will be denied. The stateful firewall keeps track of the outgoing packets that it allows to pass and allows only corresponding response packets to return. In addition to the static rule checking used by stateless firewalls, it will create a temporary (time-limited) inbound filter for the connection. This temporary filter will only allow packets from the host and port to which the outbound packet was sent, otherwise the firewall will drop the packet.

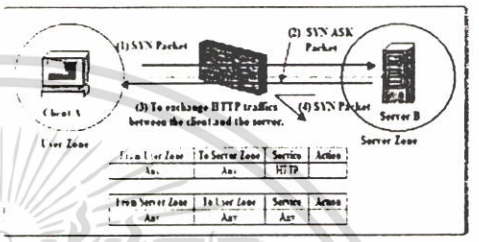


Figure 2 Steps of a stateful firewall

Let us see how different packets are accepted or rejected by the firewall according to the figure 2. Consider the following sequence of packets.

- A SYN packet sent from the client A to the server B. Since this is a SYN packet, the firewall checks the policy table to see if the packet is permitted to go through. Considering the rules in figure 2, assume the packet is permitted. An entry for this connection will be created in the session table, and the packet will be transmitted.
- A SYNACK packet is sent from the server B to the client A. Since this is not a SYN packet, the firewall will consult the session table. It will find an entry for the connection which the packet is belongs. The entry will suggest that an SYNACK packet is valid in this stage of the connection. So, the session table will be updated and the packet will be forwarded to the client A.
- Up to this state, a HTTP request packet can be sent from the client A to the server B and a HTTP response packet can also be sent from the server B to the client A even though there is no policy to allow from Server zone to User zone. Same as previous scenario.

- The server B sends a SYN packet to the client A with source port 80. Since there is no policy to allow the traffic from Server zone to User zone so the packet will be dropped

What if the packet is a UDP packet? UDP is a connectionless protocol, it's stateless by nature. There is no handshake to validate the communication, there is no sequence in which the packets must arrive. However, there is a source and destination along with a set of port numbers. An entry is still created in the session table for a UDP connection, based on source and destination IP and ports. Since there are no FIN or RST flags to close the connection, a connection must timeout to be cleared from the connection table.

4. LEGACY WAY OF HIGH AVAILABILITY

The Firewall Load balancer ensures the full availability and fault tolerance of firewalls, load balancing and dynamically distributing traffic among firewalls to guarantee continuous and uncompromised access control. The Firewall Load balancer offers full scalability and effective service growth across any firewall OS and vendor.

The Firewall Load balancer performs health checking and can determine if a firewall has failed. If a failure is detected, the firewall is taken out of service and no traffic is sent to it.

When a firewall fails, the connections on it will die. The remaining traffics of those connections will be unable to reach their final destination any more. So, clients have to retransmit all information for dead connections again.

Why will not we send the remaining traffic via the other firewalls?. Because firewalls in the farm normally are stateful firewalls and they also may come from several vendors. When the firewalls did not come from a same vendor then they will be unable to share their session table to the other vendor firewalls. If the session table is not shared , each firewall will not know the state of running connections in the other vendor firewalls. So, it will only allow the packet which it is part of existing connections it its session table , otherwise the packet will be dropped (see section 3).

5. PROPOSED ALGORITHM

As we ever mentioned, the running connections will die when a firewall fails. We propose a new algorithm to the Firewall load balancer. This algorithm will fake the stateful firewall that it is having a new connection coming to. When the stateful firewall sees the fake SYN packet , it will add information for this connection into the session table so that subsequent packets for this connection will be able to pass the stateful firewall. The steps of the proposed algorithm are shown in Figure 3.

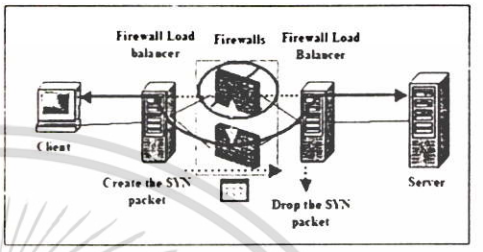


Figure 3 Steps of the proposed algorithm.

- When a firewall fails and a failure is detected, the Firewall Load balancer will create the fake SYN packets for the dead connections (The dead connection are running connections in a failing firewall). This fake SYN packet will use a source address , a destination address , a source port and a destination port of the dead connection by using information in the client table.
- The Firewall Load balancer chooses the new stateful Firewall for distributing the remaining traffic to, based on the selected load balance algorithm which may be cyclic , weight , hashing and so on.
- The Firewall Load balancer then sends the fake SYN packets to the selected stateful firewall.
- When the SYN packet come to the Stateful Firewall , the stateful firewall will keep track of the outgoing packet in its session table and allows corresponding packets to return
- Up to this state , the temporary inbound filter is created (see section 3). Both outbound and inbound packets of this connection will be allowed to pass the new stateful firewall.
- The stateful firewall then forwards the SYN packet to the next Firewall load balancer.

Pro. of the 1st KMITL International Conference on Integration of Science & Technology for Sustainable Development Bangkok. Thailand 25-26 August 2004. Vol.1: 283-286.

- When the SYN packet come to the next Firewall load balancer , because this is the fake SYN packet so the next Firewall load balancer will drop it.
- Up to this state , the Firewall load balancers at both sides of stateful firewalls will be able to send the packets to pass the new stateful firewall. In order to maintain session persistence, both Firewall load balancers will always send every remaining packets of this connection through this stateful firewall until this connection is finish.

6. EXPERIMENTS

We did two experiments , each experiment used FTP (TCP) and TFTP (UDP) for transferring files between five clients and two servers via two Juniper/NetScreen Firewalls. We started with 20 connections and expanded to 40, 60, 80 and 100 connections respectively. The first experiment used the legacy cyclic algorithm and the second experiment used our proposed algorithm. After we switched off one firewall , the results of each experiment are shown in Figure 4,5,6 and 7.

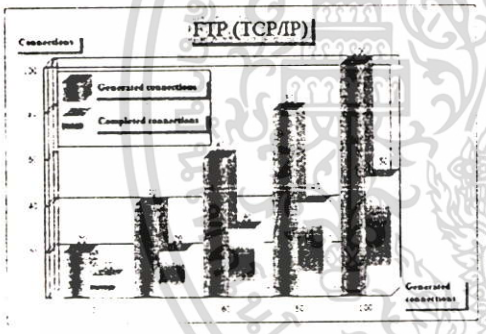


Figure 4 Legacy algorithm with FTP

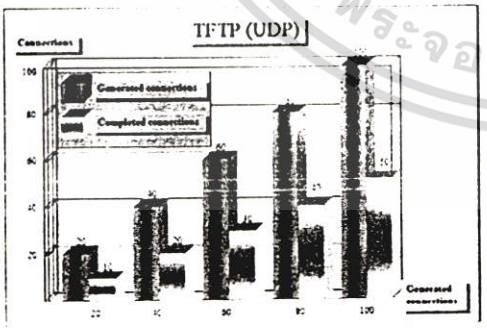


Figure 5 Legacy algorithm with TFTP

Figure 6 Proposed algorithm with FTP

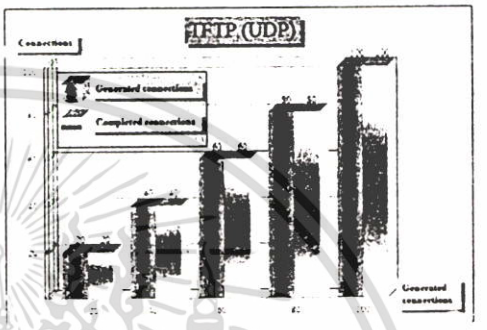


Figure 7 Proposed algorithm with TFTP

With our proposed algorithm , after we switched off one firewall , all connections were still progress until they finished but with legacy algorithm the connection failed 50%.

7. CONCLUSION

The proposed algorithm will allow the Firewall Load balancer to provide the lossless high availability to the multi-vendor stateful firewalls. With the proposed algorithm, dead connections in a failing firewall will be allowed to run in the other vendor stateful firewall by no need to recreate the connection for retransmitting the packets again.

8. REFERENCES

1. Avolio and Blask (Jan 1998), "Application Gateways and Stateful Inspection".

2. Cisco, White Paper (June 2000). "Cisco's PIX Firewall and Stateful Firewall Security".

3. Radware (Jan 2000), " FireProof sale information guide (Jan 2000)"

4. Juniper/NetScren ,White Paper (2001): "Stateful Inspection firewall" .

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายรัฐิติ์พงษ์ พุทธเจริญ
วัน เดือน ปีเกิด	10 ธันวาคม 2520
ที่อยู่	115/91 ถนน โชคชัย 4 ซอย 14 แขวง ลาดพร้าว เขต ลาดพร้าว กรุงเทพมหานคร
ประวัติการศึกษา	2542 วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ความชำนาญเฉพาะทาง	1) ระบบเครือข่าย LAN 2) ระบบเครือข่าย MAN และ WAN 3) ระบบความปลอดภัยในเครือข่าย 4) ระบบประชุมทางไกล 5) อุปกรณ์ในเครือข่ายต่าง ๆ

ประสบการณ์การทำงานและผลงานวิจัย

2542-2543	ตำแหน่งวิศวกรเครือข่าย บริษัท ไอบีเอ็มประเทศไทย จำกัด
2543-2544	ตำแหน่งวิศวกรเครือข่าย บริษัท เอทีแอนด์ทีประเทศไทย จำกัด
2544-2545	ตำแหน่งอาจารย์ประจำมหาวิทยาลัยเกษมบัณฑิต
2545-2547	ตำแหน่งวิศวกรอาวุโส บริษัท เวิลด์อินฟอร์เมชั่นเทคโนโลยี่ จำกัด
2548-ปัจจุบัน	ตำแหน่งผู้ช่วยผู้จัดการฝ่ายระบบ บริษัท เวิลด์อินฟอร์เมชั่นเทคโนโลยี่ จำกัด