ไฟร์วอลและเอเจนต์ฉลาด

Firewall with Intelligent Agent

นางสาววันลักษณ์ โกเมศ รหัส 41014381

นาย ไวโรจน์ อารัณยอังกูร รหัส 41014407

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ไฟร์วอลและเอเจนต์ฉลาด

Firewall with Intelligent Agent

โดย

นางสาววันลักษณ์ โกเมศ รหัส 41014381

นาย ไวโรจน์ อารัณยอังกูร รหัส 41014407

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

อาจารย์อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง ไฟร์วอลและเอเจนต์ฉลาด

Firewall with Intelligence Agent

ผู้จัดทำ

1. น.ส.วันลักษณ์ โกเมศ รหัสประจำตัว 41014381
2. นายไวโรจน์ อารัณยอังกูร รหัสประจำตัว 41014407

_____อาจารย์ที่ปรึกษา

(อาจารย์ ธนา หงษ์สุวรรณ)

_____อาจารย์ที่ปรึกษา

(อาจารย์ อัครเดช วัชระภูพงษ์)

ไฟร์วอลและเอเจนต์ฉลาด

น.ส.วันลักษณ์ โกเมศ 41014381
นายไวโรจน์ อารัณยอังกูร 41014407
อาจารย์ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ในปัจจุบันเรื่องของความปลอดภัยของเครือข่ายเป็นเรื่องที่ผู้คนให้ความสนใจกันมากขึ้น เนื่องจากการเจริญเติบโตของอินเทอร์เน็ต ความเสี่ยงต่อเรื่องของความปลอดภัยจึงสูงขึ้น ไฟร์วอลเป็นสิ่งหนึ่งซึ่งช่วยในการรักษาความปลอดภัย แต่ด้วยไฟร์วอลที่มีอยู่ในปัจจุบันยังมีข้อจำกัดในการนำมาใช้ในเครือข่าย จึงได้มีแนวคิดในการพัฒนาไฟร์วอลโดยใช้สถาปัตยกรรมใหม่ขึ้น

โครงงานนี้ เป็นการพัฒนาไฟร์วอลโดยใช้สถาปัตยกรรมในรูปแบบใหม่ ซึ่งเป็นการกำจัดข้อด้อยของเพอร์ซันนอลไฟร์วอลและเกตเวย์ไฟร์วอล อีกทั้งรวมเอาข้อดีของไฟร์วอลทั้งสองประเภทมารวมเข้าด้วยกัน คือสามารถควบคุมดูแลกฎได้จากส่วนกลางเหมือนเช่นเกตเวย์ไฟร์วอล แต่ส่วนไฟร์วอลจะทำงานเป็นเอเจนต์ที่ฝังตัวไว้ในเครื่องแต่ละเครื่องบนเครือข่าย ซึ่งทำให้สามารถลบจุดอ่อนของการถูกโจมตีจากภายในเครือข่ายเองซึ่งเป็นจุดอ่อนของเกตเวย์ไฟร์วอล ในขณะเดียวกันก็มีความสามารถในเรื่องการบริหารควบคุม เหมาะแก่การใช้ในองค์กร

Firewall with Intelligent Agent

Miss Wanluck Komes	41014381
Mr. Vairoj Aranyaangul	41014407
Thana Hongsuwan	Advisor
Akkradach Watcharapupong	Advisor

Abstract

Nowadays, the risk of network security is more concerned because internet has started showing up in all fact of everyday life . The connection to internet becomes as common as a telephone for most of the population. Firewall is a method used for securing data and resources on a network by limiting network traffic between user and external threats or the Internet, however, current firewall architectures has some limitation of using.

This project is concerned with the design and development of new firewall architecture not only by eliminating the inferior of Personal Firewall and Gateway Firewall but also including the advantage of both architecture by using agents. Server can control and distribute rules throughout the personal firewalls that reside as agents on every clients. The personal firewalls can protect hosts from internal attack and also have ability to manage and control like Gateway Firewall.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจสำเร็จลงด้วยดี ถ้าไม่ได้รับคำแนะนำจากอาจารย์ที่ปรึกษาทั้งสองท่าน คือ อาจารย์ธนา หงษ์สุวรรณ และอาจารย์อัครเดช วัชระภูพงษ์ ซึ่งต้องขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณบุคคลที่สำคัญที่สุด ที่ทำให้คณะผู้จัดทำมีวันนี้ คือ บิดา มารดา ที่ให้โอกาสในการศึกษาอย่างเต็มที่ และคอยเป็นกำลังใจให้เสมอมา

ขอขอบคุณเพื่อนๆ ทุกคนที่ได้ให้ความช่วยเหลือ และคำแนะนำต่างๆ

สุดท้ายนี้ ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่อำนวยความสะดวกในการใช้งานเครือข่ายและอุปกรณ์ต่างๆ

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ไฟร์วอลสามารถแบ่งออกได้เป็น 2 ประเภทใหญ่ๆ ตามลักษณะตำแหน่งที่ไฟร์วอลทำงาน คือเกทเวย์ไฟร์วอล หรือเรียกสั้นๆว่า ไฟร์วอล และอีกประเภทหนึ่งคือเพอร์ซันนอลไฟร์วอล

1. เกทเวย์ไฟร์วอล (Gateway Firewall) หรือไฟร์วอล(Firewall) จะทำงานในระดับเครือข่าย โดยอาจจะเป็นฮาร์ดแวร์หรือซอฟต์แวร์ก็ได้ ติดตั้งบนจุดที่เป็นเกทเวย์ของเครือข่ายนั้นๆเพื่อทำหน้าที่คอยตรวจสอบข้อมูลที่เข้า-ออกเครือข่าย

   ข้อดี คือ สำหรับเครือข่าย สามารถที่จะควบคุมดูแล การตั้งค่า และกฎต่างๆ สำหรับทั้งเครือข่ายให้เป็นหนึ่งเดียวได้

   ข้อเสีย คือ ไม่สามารถป้องกันการโจมตีจากภายในเครือข่ายเองได้

2. เพอร์ซันนอลไฟร์วอล (Personal Firewall) เป็นซอฟต์แวร์ซึ่งทำงานในระดับโฮสต์ คือติดตั้งที่เครื่องใดก็จะทำหน้าที่เป็น ไฟร์วอลให้เครื่องๆนั้นเท่านั้น

   ข้อดี คือสามารถที่จะป้องกันการโจมตีจากภายในเครือข่ายเดียวกันได้

   ข้อเสีย คือ ไม่สามารถที่จะควบคุม ดูแล และจัดการจากส่วนกลางได้ จึงเหมาะสำหรับการใช้งานบนเครื่องที่ไม่ได้อยู่บนเครือข่าย เช่นเครื่องตามบ้านที่ต่อเชื่อมอินเตอร์เนตผ่านโมเด็ม เนื่องจากแต่ละเครื่องสามารถกำหนดกฎเกณฑ์ได้เอง

จากข้อดีและข้อเสียของไฟร์วอลทั้ง 2 ประเภทที่ได้กล่าวมา จึงได้มีแนวคิดที่จะประยุกต์รูปแบบของไฟร์วอลโดยได้มีการนำเอาข้อดีจากไฟร์วอลทั้งสองประเภทนี้มาใช้ เพื่อกำจัดข้อด้อยที่เกิดกับไฟร์วอลทั้งสองประเภท

สำหรับแนวคิดของโครงงานนี้ เครื่องทุกๆ เครื่องในเครือข่ายจะมีเอเจนต์ ฝังตัวทำงานอยู่ ซึ่งเอเจนต์ทั้งหลายนั้นจะทำหน้าที่คล้ายกับเพอร์ซันนอลไฟร์วอล คือจะทำหน้าที่ตรวจสอบแพ็คเก็ตที่เข้า-ออกเครื่องนั้นๆ แต่จะต่างกับเพอร์ซันนอลไฟร์วอลธรรมดาตรงที่สามารถควบคุม ดูแล และบริหารเอเจนต์ทั้งหมด โดยการกำหนดกฎเกณฑ์ หรือการจัดการใดๆ จากส่วนกลาง ซึ่งไม่จำเป็นว่าแต่ละเครื่องในเครือข่ายจะต้องใช้กฎเกณฑ์เดียวกันทั้งหมด การกำหนดกฎนี้ ผู้ดูแลสามารถกำหนดให้กับผู้ใช้แต่ละคน ว่าบุคคลนั้น มีสิทธิ์ในการใช้บริการมากน้อยเพียงใด ซึ่งเป็นข้อด้อยที่อินเทอร์เน็ตไฟร์วอลไม่สามารถทำได้ จากสถาปัตยกรรมการทำงานแบบนี้ทำให้สามารถที่จะรวมเอาข้อดีของไฟร์วอลทั้งสองประเภทเอาไว้เข้าด้วยกันได้

1.2 วัตถุประสงค์ของโครงงาน

1. ศึกษาไฟร์วอล ประเภทของไฟร์วอล รูปแบบการทำงานต่างๆของไฟร์วอล รวมถึงข้อดีข้อเสียของแต่ละแบบ เพื่อใช้พิจารณาในการออกแบบไฟร์วอลให้เหมาะสม

2. ศึกษาถึงรูปแบบ วิธีการทำงาน และการพัฒนาเอเจนต์ เพื่อที่จะประยุกต์ใช้กับโครงงานได้อย่างเหมาะสม

3. ศึกษาการพัฒนาโปรแกรมควบคุมติดต่อผ่านเครือข่ายบนระบบปฏิบัติการไมโครซอฟท์วินโดวส์

4. เพื่อเป็นการเสนอแนวคิด และศึกษาความเป็นไปได้ในการสร้างไฟร์วอลโดยใช้สถาปัตยกรรมรูปแบบใหม่ ซึ่งสามารถแก้ปัญหาและจุดอ่อนจากไฟร์วอลในรูปแบบเดิมได้

ขอบเขตของงานวิจัย

1.3 ขอบเขตของโครงงาน

โครงงานนี้เป็นการพัฒนาไฟร์วอลโดยใช้สถาปัตยกรรมแบบใหม่ขึ้นมาบนระบบปฏิบัติการวินโดวส์ 2000 ซึ่งจะประกอบด้วยสองส่วนใหญ่ๆคือส่วนโปรแกรมฝั่งเซิร์ฟเวอร์ และโปรแกรมฝั่งไคลเอนต์ โดยส่วนโปรแกรมฝั่งเซิร์ฟเวอร์นั้นจะรับผิดชอบการบริหารกฎ การจัดเก็บกฎของเครื่องไคลเอนต์แต่ละเครื่องโดยใช้ระบบฐานข้อมูล ซึ่งผู้ใช้สามารถที่จำควบคุมการทำงานและรายละเอียดผ่านทาง User Interface ได้ ส่วนโปรแกรมฝั่งไคลเอนต์นั้นจะมีหน้าที่เป็นไฟร์วอลโดยทั้งสองส่วนนี้จะสามารถสื่อสารกันได้

เนื่องจากโครงงานนี้เป็นโครงงานที่สนใจศึกษาการพัฒนาสถาปัตยกรรมของไฟร์วอลมากกว่าการพัฒนารูปแบบการทำงานของไฟร์วอล ดังนั้นในโครงงานนี้จึงใช้วิธีการแพ็คเก็ตฟิลเตอร์เป็นรูปแบบการทำงานของไฟร์วอล ซึ่งง่ายที่สุดในการประยุกต์ แต่อาจจะมีความสามารถน้อยกว่ารูปแบบการทำงานของไฟร์วอลรูปแบบอื่น ซึ่งในอนาคตก็สามารถที่จะปรับปรุงแก้ไขในส่วนนี้ได้

นอกจากนี้ในโครงงานนี้ยังถือว่าเป็นโครงงานที่ทดลองสร้าง เพื่อศึกษาความเป็นไปได้ในการใช้งาน ดังนั้นจึงอาจจะมีข้อจำกัดในการใช้งาน เช่น การปรับแต่งค่าที่ไม่ยืดหยุ่นมากนัก อินเทอร์เฟสซึ่งอาจจะยังไม่สวยงามเท่าที่ควร

1.4 วิธีการดำเนินงาน

โครงงานนี้จะเริ่มต้นด้วยการศึกษาทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้องกับโครงงาน ซึ่งก็มีเรื่องหลักๆ คือ ทีซีพี/ไอพี ไฟร์วอล วินโดวส์เน็ตเวิร์กโปรแกรมมิ่ง และ เอเจนต์ จากนั้นก็จะนำเอาความรู้ที่ได้ศึกษาทั้งหมดมาออกแบบสถาปัตยกรรมของระบบ รวมทั้งส่วนต่างๆที่เกี่ยวข้อง เช่นรูปแบบของกฎ ระบบฐานข้อมูล

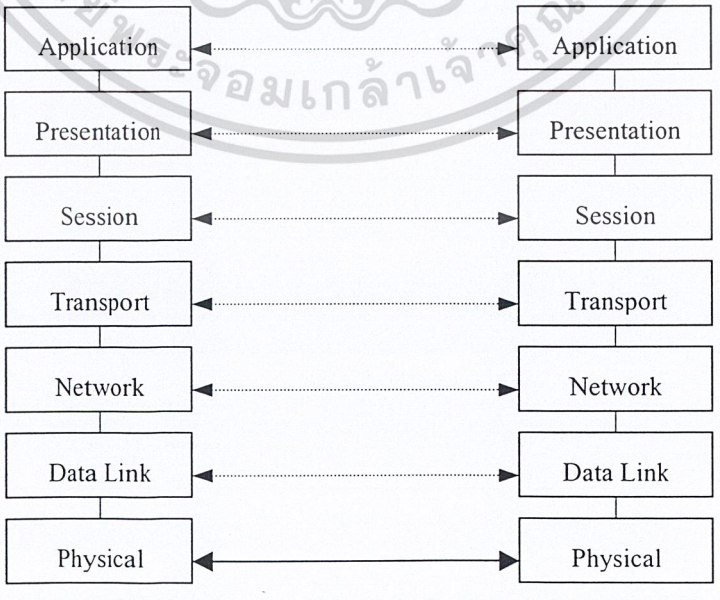
บทที่ 2
ทีซีพี/ไอพี

การศึกษาหลักการทำงานของเน็ตเวิร์คโพรโตคอล จะเริ่มต้นด้วยการมองการทำงานที่แบ่งออกเป็นชั้นๆหรือที่เรียกว่าเลเยอร์ (Layer) โดยที่แต่ละชั้นมีหน้าที่การทำงานที่ชัดเจน และไม่เกี่ยวข้องกัน แต่ละชั้นจะรู้เพียงวิธีการส่งข้อมูลไปยังชั้นอื่นๆ แต่ไม่รู้ถึงการทำงานภายในของแต่ละชั้น แต่ละโพรโตคอลจะมีการแบ่งการทำงานออกเป็นจำนวนชั้นไม่เท่ากัน ทำให้เป็นการยากที่จะระบุว่าเน็ตเวิร์คโพรโตคอลโดยรวมแล้วมีการทำงานกี่ชั้น แต่ก็มีมาตรฐานที่เป็นที่ยอมรับกันโดยทั่วไป คือ แบบจำลองสำหรับอ้างอิงแบบโอเอสไอ (OSI Reference Model) ดังนั้นจึงจะขอกล่าวถึงมาตรฐานของโอเอสไอก่อน

2.1 แบบจำลองสำหรับอ้างอิงแบบโอเอสไอ

การเชื่อมต่อคอมพิวเตอร์หลายๆ เครื่องเข้าด้วยกัน จนเกิดเป็นระบบเครือข่ายคอมพิวเตอร์ ซึ่งสามารถรับส่งข้อมูลระหว่างคอมพิวเตอร์เครื่องหนึ่งไปสู่คอมพิวเตอร์อีกเครื่องหนึ่งที่ต่างระบบต่างยี่ห้อกันได้ จำเป็นจะต้องมีมาตรฐานในการสื่อสาร หน่วยงานกำหนดมาตรฐานสากล คือ International Standards Organization หรือ ISO จึงกำหนดโครงสร้างทั้งหมดที่จำเป็นต้องใช้ในการรับส่งข้อมูลขึ้น ซึ่งเป็นสถาปัตยกรรมแบบระบบเปิด (Open System) เรียกว่า Open Systems Interconnection หรือเรียกสั้นๆ ว่า โอเอสไอ โมเด็ล (OSI Model)

โอเอสไอ โมเด็ล กำหนดมาตรฐานการสื่อสารข้อมูลจากระบบคอมพิวเตอร์หนึ่งไปยังอีกระบบหนึ่ง โดยแบ่งออกเป็น 7 ชั้น ซึ่งคอมพิวเตอร์ทั้ง 2 ฝั่งจะมีชั้นการสื่อสาร 7 ชั้นเหมือนกัน



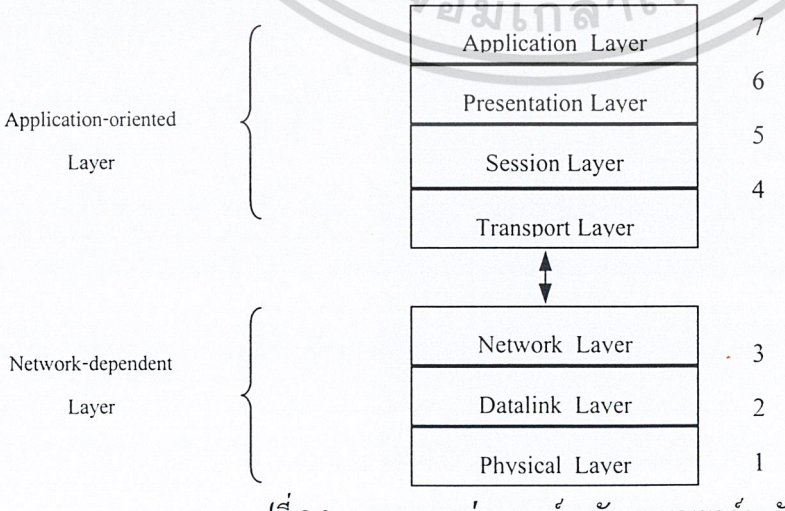
รูปที่ 2-1 การรับส่งข้อมูลของแบบจำลองอ้างอิงโอเอสไอทั้ง 7 ชั้น

ในแต่ละเลเยอร์จะเสมือนเชื่อมต่ออยู่กับชั้นที่เทียบเท่ากันของคอมพิวเตอร์อีกด้านหนึ่ง เช่น ชั้นทรานสปอร์ต จะติดต่อสื่อสารกับชั้นทรานสปอร์ต และ ชั้นแอปพลิเคชั่นจะติดต่อกับชั้นแอปพลิเคชั่นของคอมพิวเตอร์อีกฝั่งหนึ่งเท่านั้น แต่จริงๆแล้วมีสายส่งข้อมูลที่เชื่อมต่ออยู่ที่ชั้นที่ 1 หรือชั้นล่างสุดเพียงชั้นเดียว

ในการติดต่อรับส่งข้อมูล ผู้ใช้จะทำการรับส่งข้อมูลผ่านทางชั้นที่ 7 คือ ชั้นแอปพลิเคชั่นซึ่งอยู่ด้านบนสุดของ โอเอสไอ โมเด็ลเท่านั้น แล้วส่งข้อมูลให้ต่อกับชั้นที่ 6 เรื่อยไปจนถึงชั้นล่างสุดแล้วจึงส่งต่อให้กับชั้นล่างสุดของคอมพิวเตอร์ผู้รับไล่ขึ้นไปจนถึงชั้นที่ 7 ตามลำดับ เนื่องจากการติดต่อรับส่งข้อมูลจะกระโดดข้ามไปส่งให้ชั้นอื่นที่ไม่อยู่ติดกันไม่ได้ แต่ละชั้นที่ทำหน้าที่รับส่งข้อมูลจะติดต่อรับส่งข้อมูลกับชั้นที่ติดอยู่กับตัวเองเท่านั้น ชั้นแอปพลิเคชั่นจะส่งให้ชั้นเซสชั่น โดยไม่ผ่านชั้นพรีเซนเทชั่นไม่ได้ ประโยชน์ในการแบ่งเป็นชั้น คือ การกำหนดการติดต่อระหว่างชั้น ทำได้โดยไม่ต้องคำนึงถึงการเปลี่ยนแปลงในชั้นอื่นๆ

ในทางทฤษฎี แต่ละชั้นของการรับส่งข้อมูลจะมีฟังก์ชันการทำงานที่แน่นอน และแยกออกจากกัน สามารถที่จะนำแต่ละชั้นของแต่ละบริษัทมาเชื่อมต่อกันได้อย่างไม่มีขีดจำกัด แต่ในทางปฏิบัตินั้น โอเอสไอ โมเด็ล จะแบ่งออกเป็น 2 กลุ่มใหญ่ๆ คือ กลุ่มแรกได้แก่ 4 ชั้นด้านบน คือ ชั้น 7,6,5 และ 4 ทำหน้าที่เชื่อมต่อกับส่งข้อมูลระหว่างผู้ใช้กับแอปพลิเคชั่น ให้รับส่งข้อมูลกับฮาร์ดแวร์ที่อยู่ชั้นล่างได้อย่างถูกต้อง เรียกว่า ชั้นแอปพลิเคชั่นโอเรียนเต็ด โดย 4 ชั้นบนนี้มักจะเป็นซอฟแวร์ของบริษัทใดบริษัทหนึ่งรวมอยู่เป็นโปรแกรมเดียว จะแยกออกจากกันเป็นชั้นๆ เพื่อใช้โปรแกรมของบริษัทอื่นได้ลำบาก หรือในบางกรณีก็อาจทำไม่ได้เลย

กลุ่มที่สองจะเป็นชั้นล่าง ได้แก่ชั้น 3,2 และ 1 ทำหน้าที่เกี่ยวกับการรับส่งข้อมูลผ่านสายส่ง และควบคุมการรับส่งข้อมูล ตรวจสอบข้อผิดพลาด รวมทั้งเลือกเส้นทางที่ใช้ในการรับส่งข้อมูล ซึ่งจะเกี่ยวกับฮาร์ดแวร์เป็นหลัก เรียกว่า ชั้นเน็ตเวิร์กดีเพนเดนท์ ซึ่งในส่วนนี้เกี่ยวข้องกับฮาร์ดแวร์และโปรแกรมควบคุมฮาร์ดแวร์เป็นหลัก ทำให้สามารถแยกแต่ละชั้นออกจากกันได้ง่าย และใช้ผลิตภัณฑ์ของต่างบริษัทกันในแต่ละชั้นได้อย่างไม่มีปัญหา



รูปที่ 2-2 แสดงการแบ่งเลเยอร์ระดับบน-เลเยอร์ระดับล่าง

ชื่อและหน้าที่การทำงานของแต่ละชั้น

ชั้นที่ 1: ชั้นฟิสิคัล (Physical Layer)

ทำการรับส่งข้อมูลในรูปแบบ "บิต" (bit) โดยไม่พิจารณาเรื่องความหมายข้อมูล การรับส่งจะส่งข้อมูล "0" หรือ "1" ผ่านสายส่งข้อมูลจริง โดยสามารถกำหนดคุณสมบัติทางกายภายของฮาร์ดแวร์ที่ใช้เชื่อมต่อระหว่างคอมพิวเตอร์ทั้ง 2 ระบบได้ว่าจะใช้สายส่งข้อมูลแบบไหน ข้อต่อหรือปลั๊กที่ใช้ในการรับส่งข้อมูล ใช้ความต่างศักย์ไฟฟ้าเท่าใด ความเร็วในการส่งข้อมูลเป็นเท่าใด สัญญาณที่ใช้รับส่งข้อมูลในสายมีรูปร่างอย่างไร หากการรับส่งข้อมูลมีปัญหาเนื่องจากฮาร์ดแวร์ เช่น สายสัญญาณที่ใช้รับส่งข้อมูลขาด , อุปกรณ์เสียหาย ก็จะเป็นหน้าที่ของชั้นที่ 1 ที่จะตรวจสอบและแจ้งข้อผิดพลาดนั้นให้ชั้นอื่นๆที่อยู่เหนือขึ้นไปทราบ

ชั้นที่ 2: ชั้นดาต้าลิงก์ (Datalink Layer)

ทำหน้าที่จัดการและรับส่งข้อมูลในระดับฮาร์ดแวร์ โดยแปลคำสั่งที่ได้รับจากชั้นที่ 3 ให้เป็นคำสั่งควบคุมฮาร์ดแวร์ ตรวจสอบและแก้ไขข้อผิดพลาดในการรับส่งข้อมูล

ชั้นที่ 3: ชั้นเน็ตเวิร์ก (Network Layer)

ทำหน้าที่เลือกหรือกำหนดเส้นทางที่จะใช้ในการรับส่งข้อมูลระหว่างเครือข่าย และส่งผ่านข้อมูลที่ได้รับไปสู่ปลายทาง ในชั้นนี้จะมองเห็นข้อมูลทั้งหมดเป็นแพ็คเก็ตหรือเฟรม ซึ่งเป็นการผนึกคำสั่งและไดอะล็อกต่างๆ ไว้ภายใน มีเพียงแอดเดรสของผู้รับ , ผู้ส่ง , ลำดับการรับส่งและส่วนของข้อมูลเท่านั้นที่ในชั้นนี้จะมองเห็น ดังนั้นตัวเนื้อหาของข้อมูลจะไม่มีผลใดๆในการรับส่งข้อมูลเลย ไม่ว่าข้อมูลในระดับสูงจะเป็นวีดีโอ , ภาพ , เสียง หรือข้อมูลอื่นใดก็ตาม นอกจากนี้ยังทำการ Call Setup หรือเรียกติดต่อคอมพิวเตอร์ปลายทางก่อนการรับส่งข้อมูล และทำการ Call Clearing หรือยกเลิกการติดต่อเมื่อการรับส่งข้อมูลจบลงแล้ว ในกรณีที่การรับส่งข้อมูลนั้นต้องมีการติดต่อกันก่อน(Hand shaking) [1]

ชั้นที่ 4: ชั้นทรานสปอร์ต (Transport Layer)

เป็นรอยต่อระหว่างการรับส่งข้อมูลของซอฟต์แวร์กับฮาร์ดแวร์ นำข้อมูลในระดับสูงมาแปลงให้รับส่งได้ในระดับฮาร์ดแวร์ เช่น แปลงค่าหรือชื่อของคอมพิวเตอร์ในเครือข่าย (Mac address) ให้เป็นหมายเลขเครือข่าย พร้อมทั้งเป็นชั้นที่ควบคุมการรับส่งข้อมูลจากปลายด้านส่งถึงปลายด้านรับตลอดเส้นทางตามจังหวะที่กำหนดไว้ในชั้นที่ 5 เช่น ไม่ส่งข้อมูลเร็วเกินไปจนฝั่งผู้รับไม่สามารถรับข้อมูลได้ทัน เป็นการควบคุมคุณภาพของการรับส่งข้อมูลให้มีมาตรฐานในระดับที่ตกลงกันของทั้งสองฝ่าย ทำหน้าที่ตัดข้อมูลออกเป็นส่วนย่อยๆ ให้เหมาะกับลักษณะการทำงานของฮาร์ดแวร์ที่ใช้ในเครือข่าย เช่น หากข้อมูลที่ได้รับจากชั้นที่ 5 มีความยาวเกินกว่าที่เครือข่ายจะส่งได้ ชั้นที่ 4 จะตัดข้อมูลออกเป็นส่วนย่อยๆ แล้วส่งไปให้ผู้รับ ข้อมูลที่ได้รับปลายทางก็จะถูกนำมาต่อกันที่ชั้นที่ 4 ของด้านผู้รับ แล้วจึงส่งต่อให้ชั้นที่ 5 นำไปใช้ต่อไป

---

[1] การที่ผู้ส่งร้องขอการติดต่อไปสู่ผู้รับว่าจะขอทำการติดต่อรับส่งข้อมูลได้หรือไม่

ชั้นที่ 5: ชั้นเซสชั่น (Session Layer)

ทำหน้าที่ควบคุม "จังหวะ" ในการรับส่งข้อมูลของคอมพิวเตอร์ทั้งสองด้าน ที่รับส่งแลกเปลี่ยนข้อมูลกันให้มีความสอดคล้อง (Synchronization) และกำหนดวิธีที่ใช้รับส่งข้อมูล เช่น อาจจะเป็นในลักษณะสลับกันส่งฝ่ายหนึ่งรับ อีกฝ่ายหนึ่งส่ง (Half Duplex) หรือรับส่งข้อมูลพร้อมกันทั้งสองฝ่าย (Full Duplex) โดยจะมองข้อมูลเสมือนเป็นประโยคที่สนทนาโต้ตอบกัน เช่น เมื่อผู้รับได้รับข้อมูลส่วนแรกจากผู้ส่ง ก็จะโต้ตอบกลับไปให้ผู้ส่งรู้ว่าได้รับข้อมูลส่วนแรกเรียบร้อยแล้ว และพร้อมที่จะรับข้อมูลส่วนที่สองต่อไป

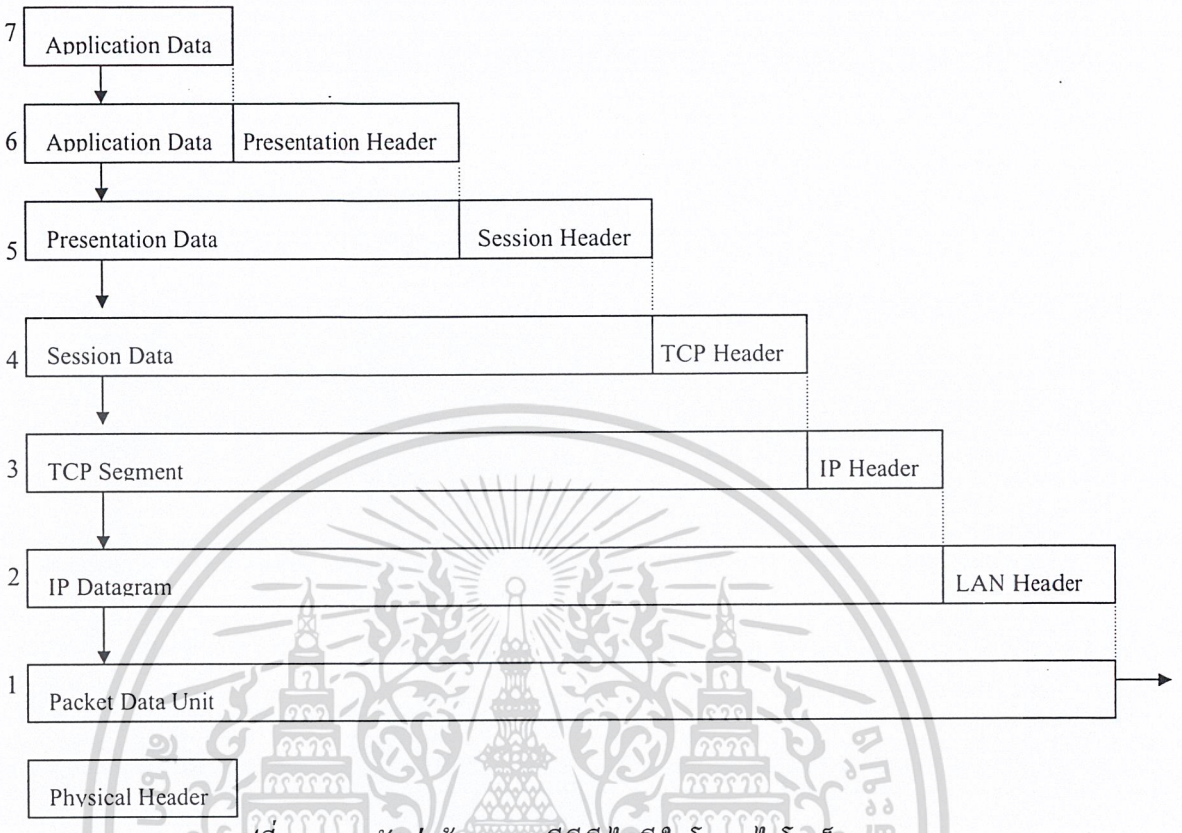
ชั้นที่ 6: ชั้นพรีเซนเทชั่น (Presentation Layer)

เป็นชั้นที่ทำหน้าที่ตกลงกับคอมพิวเตอร์อีกฝั่งหนึ่งว่ามีขั้นตอนและข้อบังคับอย่างไรในการรับส่งข้อมูล เนื่องจากรูปแบบของข้อมูลจะเป็นคำสั่งที่มีกฎ(Syntax)บังคับอย่างแน่นอน เช่น ในการคัดลอกไฟล์ จะต้องสร้างไฟล์ใหม่ขึ้นมาจากนั้นจึงเปิดไฟล์ แล้วจึงรับข้อมูลจากปลายทางมาเก็บลงในไฟล์ที่สร้างใหม่นี้ นอกจากนี้ยังทำหน้าที่แปลความหมายของคำสั่งที่ได้รับจากชั้นที่ 7 ให้เป็นคำสั่งระดับปฏิบัติการส่งให้ชั้นที่ 5 ต่อไปอีกด้วย

ชั้นที่ 7: ชั้นแอปพลิเคชั่น (Application Layer)

เป็นชั้นสูงสุด ที่รับคำสั่งต่างๆ จากผู้ใช้ นำมาแปลความหมายและทำงานตามคำสั่งที่ได้รับในระดับโปรแกรมประยุกต์ เช่น แปลความหมายของการกดปุ่มเมาส์ให้เป็นคำสั่งในการคัดลอกไฟล์ หรือดึงข้อมูลมาแสดงผลบนจอภาพ ซึ่งการแปลคำสั่งจะต้องแปลออกมาให้ถูกกฎ(Syntax)ที่ใช้ในระบบคอมพิวเตอร์นั้นๆ ตัวอย่างเช่น ถ้ามีการคัดลอกไฟล์ ชื่อไฟล์จะต้องยาวไม่เกินจำนวนที่ระบบปฏิบัติการใช้อยู่ และต้องประกอบด้วยตัวอักษรตามที่กำหนด รวมไปถึงฟังก์ชันที่ใช้ในการรับส่งข้อมูลระหว่างชั้นที่ 7 กับชั้นที่ 6 ด้วย

โดยในการรับส่งข้อมูลใน โอเอสไอ โมเด็ลนี้ ข้อมูลจากชั้นบนสุด คือชั้นที่ 7 จะถูกส่งลงไปในชั้นถัดไปจนกระทั่งถึงชั้นที่ 1 โดยข้อมูลเดิมจะถูกผนึกรวมกับข้อมูลที่ใช้ควบคุมของแต่ละชั้นซ้อนๆ กันเป็นลำดับ เช่น ข้อมูลจากชั้นแอปพลิเคชั่น คือ แอปพลิเคชั่นดาต้า เมื่อถูกส่งลงไปยังชั้นถัดไป ก็จะถูกผนึกด้วยแอปพลิเคชั่นเฮดเดอร์ และทั้ง แอปพลิเคชั่นเฮดเดอร์ และแอปพลิเคชั่นดาต้าจะถูกนำรวมกันเป็นข้อมูลของชั้นพรีเซนเทชั่น ซึ่งก็จะถูกผนึกด้วยชั้นพรีเซนเทชั่นอีกครั้ง ก่อนที่จะส่งต่อเป็นข้อมูลให้กับชั้นถัดไป แต่ละชั้นจะผนึกเฮดเดอร์ของตัวเองก่อนส่งต่อชั้นถัดไป จนกระทั่งถึงชั้นล่างสุดซึ่งเป็น ชั้นฟิสิคัล ซึ่งจะทำการส่งข้อมูลต่อไปให้ถึงปลายทาง และข้อมูลที่ปลายทางได้รับ จะถูกนำมาแยกเฮดเดอร์ที่เพิ่มเข้ามานี้ออกทีละชั้นจนกระทั่งถึงชั้นบนสุด ก็จะได้แอปพลิเคชั่นดาต้าที่ผู้ส่งส่งให้แก่ผู้รับ



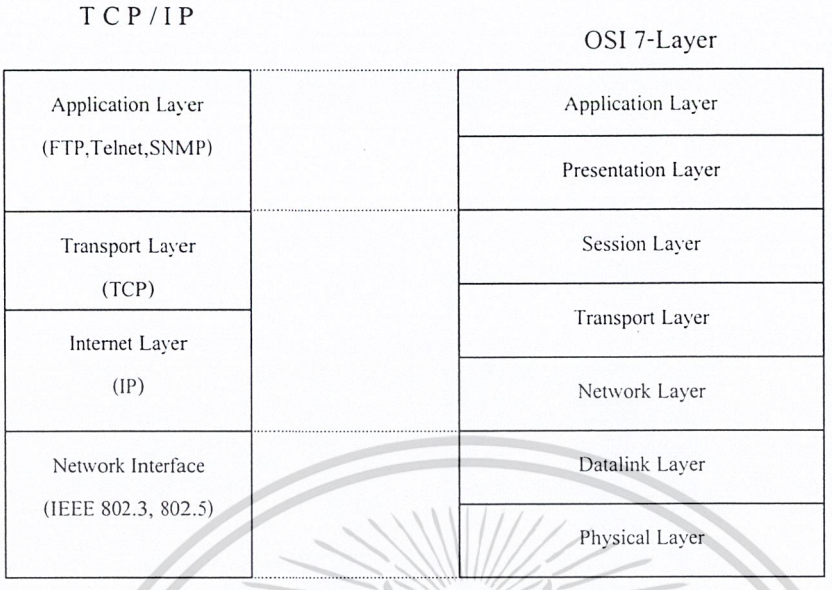
รูปที่ 2-3 การรับส่งข้อมูลของทีซีพี/ไอพี ในโอเอสไอโมเด็ล

2.2 ทีซีพี/ไอพี

TCP/IP (Transmission Control Protocol / Internet Protocol) เป็นมาตรฐานที่เกิดขึ้นก่อนโอเอสไอโมเด็ล ทีซีพี/ไอพีได้ถือกำเนิดขึ้นบนเครือข่ายอาร์พาเน็ต (ARPANET) ซึ่งต่อมาได้ขยายการเชื่อมต่อทั่วโลกเป็นเครือข่ายอินเทอร์เน็ต ทำให้มาตรฐานของทีซีพี/ไอพี เป็นที่ยอมรับกว้างขวางและเนื่องจากเป็นโพรโตคอลที่ใช้ได้ฟรีโดยไม่ต้องเสียค่าลิขสิทธิ์ จึงทำให้มีจำนวนผู้ใช้งานเพิ่มขึ้นจำนวนมากจนถือเป็นมาตรฐานที่มีผู้ใช้รับส่งข้อมูลมากที่สุดในปัจจุบัน

มาตรฐาน ทีซีพี/ไอพี นี้ไม่ได้เป็นไปตามรูปแบบของโอเอสไอโมเด็ลเนื่องจากโอเอสไอโมเด็ลถูกออกแบบโดยองค์กรขนาดใหญ่ซึ่งใช้เวลานานในการออกแบบตลอดจนการรับรองมาตรฐานต่างกับทีซีพี/ไอพี ซึ่งถูกออกแบบด้วยความต้องการอันเร่งด่วนของรัฐบาลสหรัฐฯ จึงทำให้การพัฒนาทีซีพี/ไอพีมีเงื่อนไขในด้านความต้องการที่ต่างจากโอเอสไอโมเด็ล

ทีซีพี/ไอพี มีการแบ่งจำนวนชั้นที่ใช้รับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ออกเป็นเพียง 4 ชั้น เรียกว่าทีซีพี/ไอพีสแตก



รูปที่ 2-4 ทีซีพี/ไอพี โพรโตคอลเมื่อเทียบกับโอเอสไอโมเด็ล

2.2.1 ทีซีพี/ไอพีสแตก

ชั้นที่ 1: เน็ตเวิร์กอินเตอร์เฟซ (Network Interface)

เป็นชั้นที่ควบคุมฮาร์ดแวร์การรับส่งข้อมูลผ่านเครือข่าย ซึ่งเทียบได้กับชั้นที่ 1 และชั้นที่ 2 ของโอเอสไอโมเด็ล ในชั้นนี้จะทำหน้าที่เชื่อมต่อกับฮาร์ดแวร์และควบคุมการรับส่งข้อมูลในระดับฮาร์ดแวร์ของเครือข่ายทีซีพี/ไอพี ไม่ได้กำหนดรูปแบบของการเชื่อมต่อในระดับนี้ไว้ใหม่แต่ได้ใช้มาตรฐานที่มีอยู่เดิมที่กำหนดไว้ก่อน ซึ่งที่ใช้กันอยู่จะเป็นตามมาตรฐานของ IEEE เช่น IEEE 802.3 จะเป็นการเชื่อมต่อผ่าน LAN แบบ Ethernet LAN หรือ IEEE 802.5 จะเป็นการเชื่อมต่อผ่าน LAN แบบ Token Ring เป็นต้น ตัวอย่างการทำงานได้แก่ การเพิ่มเฮดเดอร์เข้าไปในดาต้าแกรมเพื่อให้กลายเป็นเฟรมแล้วส่งไปตามเครือข่าย

ชั้นที่ 2: ชั้นอินเทอร์เน็ต (Internet Layer)

ในชั้นนี้มีหน้าที่ส่งผ่านข้อมูลระหว่างเครือข่าย ทำการหาเส้นทางการส่ง หน้าที่ของชั้นนี้เทียบเท่ากับชั้นเน็ตเวิร์กและชั้นดาต้าลิงก์ของโอเอสไอโมเด็ล โดยมีโพรโตคอลที่ทำงานเป็นกลไกสำคัญในชั้นนี้คือไอพี นอกจากนี้ยังมีโพรโตคอลที่ทำงานอยู่ในชั้นนี้อีก 2 ชนิด คือ ไอซีเอ็มพี (Internet Control Message Protocol , ICMP) และเออาร์พี (Address Resolution Protocol,ARP)

ชั้นที่ 3: ชั้นทรานสปอร์ต (Transport Layer)

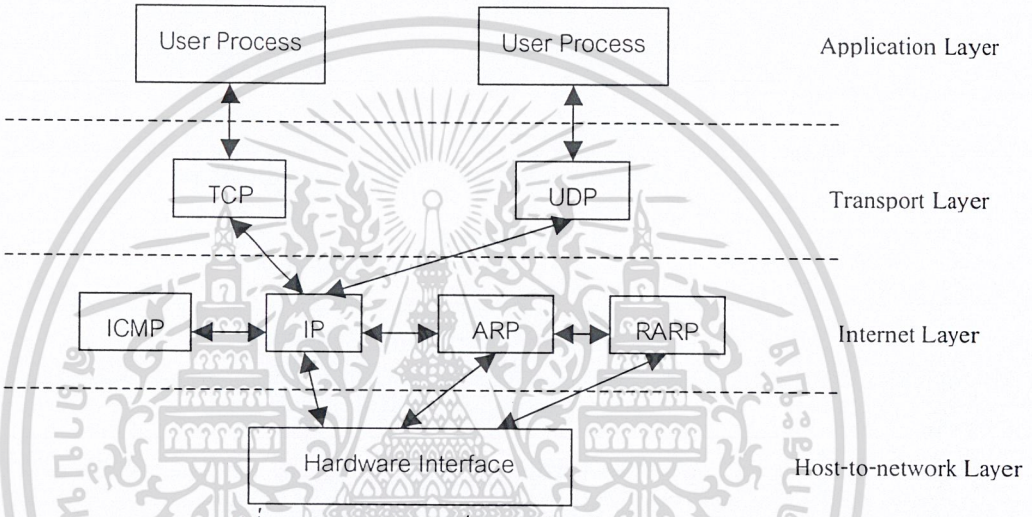
รับผิดชอบการส่งข้อมูลจากจุดต้นทางไปจนถึงปลายทาง หากเปรียบเทียบกับโอเอสไอโมเด็ลก็สามารถเทียบได้กับชั้นเซสชั่นรวมกับชั้นทรานสปอร์ตนั่นเอง โดยมีซ็อคเก็ตเป็นจุดปลายในการสื่อสาร ซึ่งซ็อคเก็ตนี้ประกอบไปด้วยหมายเลขของคอมพิวเตอร์และหมายเลขพอร์ตของเครื่องที่ต้องการส่งข้อมูลไปถึง บริการหลักในชั้นนี้มีอยู่ 2 แบบ คือ คอนเน็คชั่นโอเรียนเต็ด โดยผ่านทีซีพี และ คอนเน็คชั่นเลส ซึ่งผ่านยูดีพี

ชั้นที่ 4 : ชั้นแอปพลิเคชั่น (Application Layer)

ชั้นนี้รองรับการทำงานของแอปพลิเคชั่นต่างๆ ที่ทำงานอยู่ในเครื่องต้นทางและปลายทาง การทำงานของแอปพลิเคชั่นต่างๆมีการติดต่อกันตามแต่ละโพรโตคอลเฉพาะแล้วแต่แอปพลิเคชันที่ใช้งาน การทำงานในชั้นนี้เทียบได้กับชั้นแอปพลิเคชั่นรวมกับชั้นพรีเซนเทชั่นของโอเอสไอโมเด็ล

2.2.2 ชุดของทีซีพี/ไอพี โพรโตคอล

ชุดของทีซีพี/ไอพีโพรโตคอล นอกจากมีโพรโตคอลทีซีพีและไอพีแล้ว ยังมีโพรโตคอลอื่นๆอีก ดังภาพแสดงความสัมพันธ์ของชุดโพรโตคอลโดยแบ่งตามชั้น



รูปที่ 2-5 ความสัมพันธ์ของชุดโพรโตคอล

2.3 โพรโตคอล

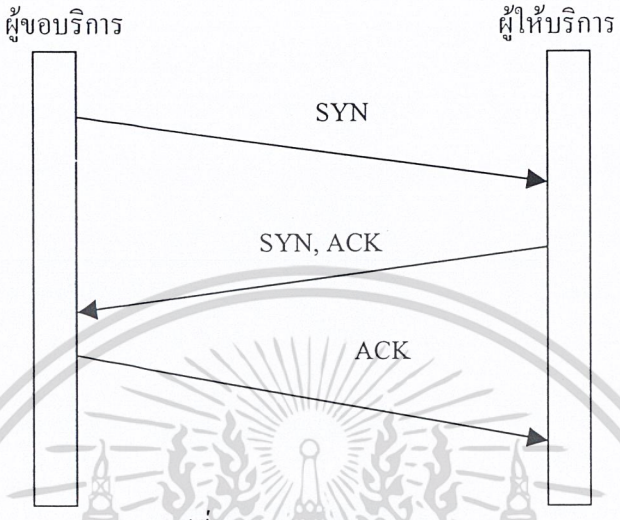
2.3.1 โพรโตคอลทีซีพี

เป็นโพรโตคอลแบบคอนเน็คชั่นโอเรียนเต็ดซึ่งจะรับประกันการรับ-ส่งข้อมูล มีความน่าเชื่อถือ หน้าที่หลักของการทำงานของทีซีพีในการรับส่งข้อมูลมี 6 อย่าง ดังนี้

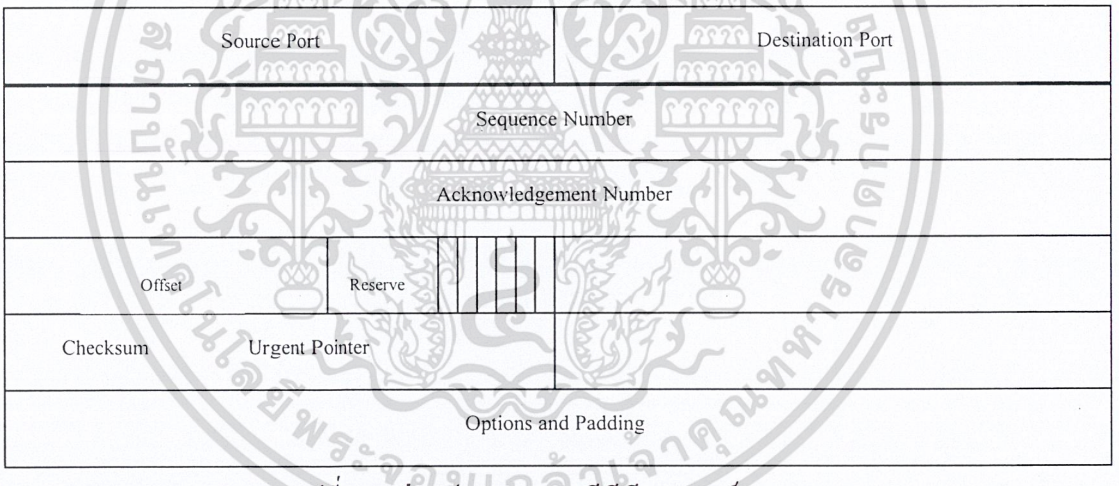
1. ควบคุมการรับส่งข้อมูล (Basic Data Transfer)
2. ความน่าเชื่อถือในการรับส่งข้อมูล (Reliability)
3. ควบคุมการไหลของข้อมูล (Flow Control)
4. การทำมัลติเพล็ก (Multiplexing)
5. ควบคุมการเชื่อต่อ (Connection)
6. ความปลอดภัยในการรับส่งข้อมูล (Security)

การทำงานที่สำคัญอย่างหนึ่งของทีซีพี คือการทำ “3-way Handshake” ซึ่งเป็นการะบวนการเริ่มต้นในการสร้างการเชื่อมต่อในชั้นทรานสปอร์ต กล่าวคือ ในการติดต่อกันระหว่างระบบในเครือข่ายต้องมีการสร้างการเชื่อมต่อไปยังระบบที่ให้บริการก่อน โดยผู้ขอบริการส่งสัญญาณ SYN เพื่อขอบริการ จากนั้นผู้ให้บริการจะส่งสัญญาณ ACK เพื่อตอบรับการเชื่อต่อที่ร้องขอมา จึงจะเริ่มต้นส่งข้อมูลได้

การเชื่อมต่อแบบ 3-way handshake นี้ เป็นการตรวจสอบความพร้อมของทั้งฝ่ายส่งและฝ่ายรับ และการกำหนดค่าเริ่มต้นของพารามิเตอร์ต่างๆของทั้งสองฝ่ายให้ตรงกัน หลังจากกระบวนการทำ 3-way handshake สิ้นสุด ทั้งสองฝ่ายจึงสามารถรับและส่งข้อมูลระหว่างกันได้



รูปที่ 2-6 3-way handshake



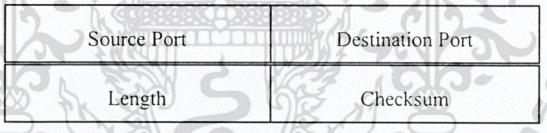
รูปที่ 2-7 ส่วนประกอบของทีซีพี เฮดเดอร์

เฮดเดอร์ของทีซีพี

- Source port (16-bit) พอร์ตที่โปรแกรมผู้ส่งใช้ส่งข้อมูล
- Destination port (16-bit) พอร์ตทางฝั่งผู้รับที่ข้อมูลถูกส่งไป
- Sequence number (32-bit) บอกตำแหน่งของข้อความที่ถูกแบ่ง เมื่อข้อมูลมีขนาดใหญ่เกินกว่าที่ระบบกำหนด ใช้ในการเรียงลำดับข้อมูลเมื่อถึงผู้รับ
- Acknowledgement number (32-bit) บอก Sequence number ที่ควรจะเป็น
- Data offset (4-bit) นำขนาดของ Header คำนวณหาตำแหน่งเริ่มต้นของข้อมูล
- Reserved (6-bit) สงวนไว้สำหรับใช้ในอนาคต แต่เนื่องจากไม่เคยถูกใช้ จึงเซ็ทค่าเป็นศูนย์
- URG flag ถ้ามีค่าเป็น “1” Urgent Pointer จะชี้อยู่ที่ข้อมูลที่มี flag บอกว่า “urgent”

- ACK flag ถ้ามีค่าเป็น “1” แสดงว่าแพ็คเก็ต[2]เป็นแพ็คเก็ตตอบรับ เมื่อมีผู้ส่งส่งการร้องขอการติดต่อข้อมูลออกไป
- PSH flag ถ้ามีค่าเป็น “1” มีการใช้ push function
- RST flag ถ้ามีค่าเป็น “1” เป็นการเริ่มต้นการติดต่อใหม่
- SYN flag ถ้ามีค่าเป็น “1” แสดงถึง Sequence number เกิดขึ้นในจังหวะเดียวกัน
- FIN flag ถ้ามีค่าเป็น “1” ผู้ส่ง ส่งข้อมูลเสร็จสิ้น
- Window (16-bit) จำนวนบล๊อคของข้อมูลที่ผู้รับสามารถรับได้ในขณะหนึ่ง
- Checksum (16-bit)เป็นการคำนวณเพื่อตรวจสอบความถูกต้องของข้อมูลทั้งในส่วนเฮดเดอร์และเนื้อหา
- Urgent Pointer (16-bit) ถ้า URG flag มีค่าเป็น “1” พอยเตอร์จะชี้ไปยังที่ซึ่งเก็บ Urgent Data โดยฟิลด์นี้จะถูกเรียกใช้จากโปรแกรมที่อยู่เหนือทีซีพีในสแตก
- ออปชัน (Option) ความยาวไม่คงที่เช่นเดียวกับออปชันในเฮดเดอร์ของไอพี หน้าที่หนึ่งของออปชันคือการกำหนดขนาดใหญ่สุดของเซกเมนต์ (Segment) และเนื่องจากความยาวที่ไม่แน่นอนของออปชันทำให้ต้องมี Padding เพื่อเติมขนาดให้เต็ม 32-bit

2.3.2 โพรโตคอลยูดีพี



รูปที่ 2-8 ส่วนประกอบของยูดีพี เฮดเดอร์ ทุกฟิลด์มีขนาด 16-bit

มีการทำงานคล้ายกับทีซีพี คือ จัดการเกี่ยวกับการสื่อสารระหว่างเครื่อง แต่เป็นแบบคอนเน็คชั่นเลส คือ ทั้งฝ่ายส่งและฝ่ายรับไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกัน และไม่รับรองว่าข้อมูลที่ส่งไปจะไปถึงปลายทางหรือไม่ และอาจซ้ำซ้อนหรือผิดลำดับได้ แต่ข้อดีของโพรโตคอลนี้คือโอเวอร์เฮดต่ำ

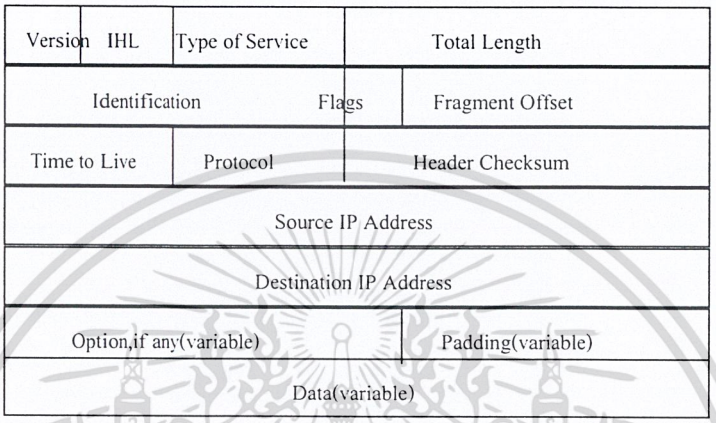
2.3.3 โพรโตคอลไอพี

ไอพีเป็นโพรโตคอลที่จัดการเกี่ยวกับแอดเดรสของแต่ละแพ็คเก็ต เพื่อให้การส่งแพ็คเก็ตต่างๆ ไยังเป้าหมายได้ถูกต้อง การทำงานของไอพีเป็นเพียงการส่งข้อมูลไปยังเครื่องเป้าหมายเท่านั้น ไม่มีการส่งสัญญาณของบริการ หรือสัญญาณให้บริการระหว่างกันเหมือนทีซีพี กล่าวคือเป็นการเชื่อมต่อแบบคอนเน็คชั่นเลส โดยปล่อยให้โพรโตคอลในชั้นที่เหนือขึ้นไป มีหน้าที่จัดการเรื่องความถูกต้อง การหาเส้น

---

[2] แพ็คเก็ต(Packet) หรือดาต้าแกรม (Datagram) คือข้อมูลที่ประกอบไปด้วยส่วนหัวและส่วนของเนื้อหาของแต่ละชั้นใน ทีซีพี/ไอพี สแตก โดยชั้นอื่น จะไม่สนใจและอ่านข้อมูลภายในแพ็คเก็ตได้ เรียกว่าการ เอ็นแคปซูเลชั่น (Encapsulation)

ทางของข้อมูลจะทำในระดับของไอพีนี้ โดยพิจารณาแต่ละแพคเก็ตแยกออกจากกัน ในข้อมูลของโพรโตคอลไอพีจะมีข้อมูลของหมายเลขไอพีปลายทางที่จะส่งข้อมูลไปและเมื่อถึงเครือข่ายปลายทางแล้วจะมีกลไกแปลงหมายเลขไอพีให้เป็นหมายเลขฮาร์ดแวร์ประจำเครื่องที่ถูกต้องอีกทีหนึ่งด้วยโพรโตคอลเออาร์พี



รูปที่ 2-9 ส่วนประกอบของไอพี เฮดเดอร์

- Version (4-bit) เวอร์ชันของไอพี ถ้าเวอร์ชันต่างกัน รูปแบบของข้อมูลในเฮดเดอร์จะต่างกันและไม่สามารถใช้ร่วมกันได้ ปัจจุบันมีเพียงถึงเวอร์ชัน 4 สำหรับไอพีเวอร์ชันถัดไป เรียกว่า "IPV6"
- Internet Header Length , IHL (4-bit) บอกความยาวในส่วนของเฮดเดอร์ ทำให้โปรแกรมสามารถคำนวณและทราบได้ว่าส่วนของข้อมูลเริ่มต้นที่บิตใด
- Type of Service (8-bit) บอกลำดับความสำคัญก่อนหลังของแพ็คเก็ต แต่เนื่องจากยังไม่มีการใช้ในไอพีเวอร์ชัน 4 นี้ จึงมีค่าเป็น 0s
- Datagram Length (8-16bit) บอกความยาวของทั้งดาต้าแกรมมีขนาด 8-bit หรือมากถึง 65,535 ไบต์(16-bit) หากลบออกด้วย IHL ก็จะสามารถทราบความยาวในส่วนของเนื้อหาได้
- Identification(16-bit) เป็นฟิลด์ที่บ่งชี้ว่าดาต้าแกรมนี้มาจากข้อความใด ในกรณีที่ข้อความมีขนาดใหญ่เกินจากที่กำหนด ต้องแบ่งออกให้เป็นดาต้าแกรมย่อย แต่ไม่ได้บอกลำดับของดาต้าแกรม
- Flags(3-bit) เป็นตัวที่บอกว่าดาต้าแกรมนี้ได้มาจากการแฟรกเมนเทชั่นหรือไม่
- Time to Live , TTL (8-bit) ตัวกำหนดระยะเวลายาวนานที่สุด ที่ดาต้าแกรมจะอยู่ในเครือข่าย
- Protocol (8-bit) ชนิดของโพรโตคอลของข้อมูลที่อยู่ในดาต้าแกรม
- Header Checksum (16-bit) เช็คความถูกต้องของข้อมูลส่วนเฮดเดอร์

- Source Address (32-bit) หมายเลขไอพีของผู้ส่งดาต้าแกรม
- Destination Address (32-bit) หมายเลขไอพีของดาต้าแกรมที่ถูกส่งไป
- ออปชัน เช่น Control, Reserved,Debugging หรือ Measurement
- Padding เนื่องจากออปชันมีขนาดไม่แน่นอน จึงนำแพดดิงมาทำให้ดาต้าแกรมมีขนาด32บิต

2.3.4 โพรโตคอลไอซีเอ็มพี(Internet Control Message Protocol)

การแจ้งเตือนหรือแสดงข้อความจากระบบ เพื่อบอกให้ผู้ใช้ทราบว่าเกิดอะไรขึ้นในการส่งผ่านข้อมูลนั้น ซึ่งปัญหาส่วนมากคือส่งไปไม่ได้ หรือปลายทางรับข้อมูลไม่ได้ ข้อความที่ไอซีเอ็มพี ส่งนั้นแบ่งออกได้ 2 แบบคือ ข้อความแจ้งข้อผิดพลาด และ ข้อความเรียกขอข้อมูลเพิ่ม คำสั่ง ping หรือ traceroute ก็เป็นคำสั่งหนึ่งของ ไอซีเอ็มพี

คำสั่งที่สำคัญของ ไอซีเอ็มพี

- ping จะใช้ echo request และ echo reply message เพื่อตัดสินว่ามีการติดต่อกันจริงระหว่างสองเครือข่าย
- source quench message ใช้บอกผู้ส่ง เมื่อส่งข้อมูลในอัตราเร็วเกินกว่าที่ผู้รับ สามารถรับได้ทัน
- redirect message เราเตอร์ใช้ในการบอกเราเตอร์ตัวอื่นๆ เมื่อพบเส้นทางไปสู่ปลายทางที่ดีกว่า
- time exceeded message เราเตอร์ใช้ในการบอกอุปกรณ์อื่นๆ หากแพ็คเก็ตถูกทิ้งไป

เช่นเดียวกับทีซีพีและยูดีพี ไอซีเอ็มพี ใช้ไอพีโพรโตคอลในการส่งผ่านข้อมูลออกสู่เครือข่าย

2.3.5 โพรโตคอลเออาร์พี (Address Resolution Protocol)

ถูกเรียกใช้งานโดยโพรโตคอลไอพีเพื่อช่วยแปลงหมายเลขไอพีไปเป็นหมายเลขไอพีปลายทาง ตัวอย่างเช่น เว็บเซิร์ฟเวอร์เครื่องหนึ่งเชื่อมต่ออยู่ในเครือข่ายอินเทอร์เน็ต และในการเชื่อมต่อนี้ต้องอาศัย Network Interfece Card (NIC) หรือ LAN card ติดตั้งอยู่ LAN card นี้จะมีหมายเลขเฉพาะประจำฮาร์ดแวร์ที่ไม่ซ้ำใคร เพื่อใช้อ้างอิงการส่งข้อมูลในเครือข่าย แต่เมื่อมาใช้งานโพรโตคอล ทีซีพี/ไอพี ก็จะต้องมีการกำหนดหมายเลขไอพีหมายเลขประจำตัวเพื่อใช้อ้างอิงกัน และโพรโตคอลเออาร์พีจะทำหน้าที่แปลงค่าหมายเลขไอพีให้เป็นหมายเลขฮาร์ดแวร์จริงให้ในระดับการทำงานที่ชั้นอินเทอร์เน็ตเวิร์ก ซึ่งกลไกในการแปลงนี้เรียกว่า address resolution

2.4 บริการของ ทีซีพี/ไอพี

2.4.1 เทลเน็ต

เป็นโปรแกรมบน UNIX ที่อนุญาตให้ผู้ใช้เข้าสู่ระบบได้ แม้จะอยู่นอกพื้นที่ของเครือข่าย Te เทลเน็ตถูกนำมาพัฒนาให้สามารถใช้ได้บนหลายระบบปฏิบัติการ เช่น ผู้ใช้สามารถเข้าสู่ระบบได้ ถึงแม้จะเรียกใช้จาก Window NT แต่ยังคงใช้คำสั่งของ UNIX ในการดำเนินการต่างๆ การใช้งานเทลเน็ตถูกนำมาใช้อยู่บนพอร์ต 23

2.4.2 เอฟทีพี

เช่นเดียวกับเทลเน็ตเป็นบริการ (Service) ที่ใช้สำหรับรับ-ส่งไฟล์ สามารถเข้าสู่ระบบได้แม้จะอยู่ในพื้นที่อื่นภายนอกเครือข่าย ด้วยการใช้ Username และ Password รวมถึง Anonymous ซึ่งอนุญาตให้ผู้ใดก็ตามสามารถใช้บริการนี้ได้ FTP ถูกนำมาใช้บริการอยู่บนพอร์ต 21

2.4.3 ดีเอ็นเอส

ทำหน้าที่แปลงข้อมูลชื่อโดเมนเนม หรือชื่อเว็บไซต์ทั้งหลายให้เป็นหมายเลขไอพี

2.4.4 เอสเอ็มทีพี

เป็นบริการสำหรับการส่งและรับอีเมล์ โดยจะถูกเรียกใช้ขณะที่โปรแกรมคอมพิวเตอร์ของด้านผู้ใช้งาน ส่งเมล์มาที่ MTA[3] (Mail Transfer Agent) และใช้รับส่งอีเมล์ระหว่าง MTA ด้วยกัน สำหรับการใช้เมล์บนเครื่องที่ผู้ใช้ใช้อ่านเมล์ไม่ได้ต่อกับเครื่องที่มีเมล์บ๊อกซ์ตลอดเวลา อาจจะดาวน์โหลดเมล์มาเก็บไว้ที่เครื่องของตัวเอง โดยใช้โพรโตคอล POP (Post Office Protocol) และ IMAP (Internet Message Access Protocol) ได้

2.5 รูปแบบและการกำหนดแอดเดรสของ ทีซีพี/ไอพี

ในหัวข้อนี้จะได้อธิบายรูปแบบการกำหนดแอดเดรสของโพรโตคอลทีซีพี/ไอพี ซึ่งลักษณะแอดเดรสของโพรโตคอล นี้ค่าของแอดเดรสของเครื่องในระบบเครือข่ายจะไม่ซ้ำกันเลย โดยเรียกเลขนี้ว่าหมายเลขไอพี เป็นเลข 32 บิตซึ่งแบ่งเป็นคลาส ตามหลักในการพิจารณาที่จะได้กล่าวต่อไปนี้

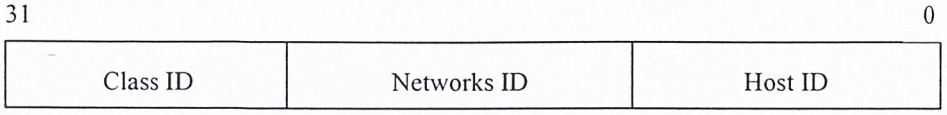
2.5.1 การแบ่งคลาสเน็ตเวิร์ก

เนื่องจากหมายเลขแอดเดรสของคอมพิวเตอร์เครื่องใดๆนั้น จะต้องสามารถบอกถึงความแตกต่างระหว่างตัวเรื่องเอง ตลอดจนเครือข่ายที่คอมพิวเตอร์นั้นเชื่อมต่ออยู่ด้วย หมายเลขไอพีจึงแบ่งแยกออกเป็น 2 ส่วน ได้แก่ ส่วนที่แสดงหมายเลขของโฮสต์ และส่วนที่เป็นหมายเลขของเครือข่าย

การแบ่งคลาสของแอดเดรสทำได้โดยพิจารณาจำนวนบิตของ 2 ส่วนประกอบข้างต้น ซึ่งมีการแบ่งออกเป็น 5 คลาส แต่มีการใช้เพียง 3 คลาสแรก คือ คลาส A, คลาส B และ คลาส C ส่วนคลาส D และคลาส E ถูกสงวนไว้สำหรับจุดประสงค์พิเศษ

---

[3] โปรแกรมคอมพิวเตอร์ที่จะส่งอีเมล์จากต้นทางไปยังปลายทาง ซึ่งต้องส่งผ่านเครื่องจำนวนมากที่เชื่อมต่อกันในเครือข่าย โดยโปรแกรมเหล่านี้จะช่วยส่งต่ออีเมล์เป็นทอดๆจนไปถึงเครื่องที่มี account หรือเมล์บ๊อกซ์ของผู้รับ และหากไม่สามารถส่งเมล์ถึงผู้รับได้ อาจเป็นเพราะชื่อผู้รับผิด จะส่ง mail error กลับมาส่งผู้ส่ง โดยเครื่องที่มี MTA ทำงานอยู่มักจะมีเมล์บ๊อกซ์ของผู้ใช้อยู่ด้วย ซึ่งเป็น primary mailbox และเรียกเครื่องนั้นว่า Mail Server



IP Address Format

Network Class	Networks	Hosts per Network
A	124	16,777,214
B	16,382	65,534
C	2,097,150	254

รูปที่ 2-10 แสดงคลาส, จำนวนเครือข่าย และจำนวนโฮสต์ของแต่ละคลาส

2.5.2 การทำซับน็ต (Subnetting)

การทำซับเน็ตเป็นการเปลี่ยนแปลงการใช้หมายเลขของเครื่องโฮสต์และหมายเลขของเครือข่ายในระดับท้องถิ่น โดยในทางตรรก คือ การเลื่อนเส้นแบ่งที่แยกหมายเลขเครื่อง และหมายเลขของเครือข่ายที่อยู่ในหมายเลขไอพีโดยที่ปริมาณของหมายเลขโฮสต์และหมายเลขเครือข่ายจะแปรผกผันกัน ยกตัวอย่างเช่น หากมีปริมาณของเครือข่ายมาก ก็จะทำให้เครื่องใดๆที่จะต่อกับระบบเครือข่ายหนึ่งๆน้อยลง เป็นต้น ในการปฏิบัติการทำซับเน็ตทำโดยการทำซับเน็ตมาสก์ (Subnet Mask) คือตัวเลขจำนวน 32 บิต มาทำการ AND กับหมายเลขไอพีตัวอย่างเช่นกำหนดหมายเลขไอพีเป็น 161.246.5.13 และซับเน็ตมาสก์คือ 255.255.255.0 จะได้เป็น 161.246.5.0

จะเห็นว่าหากพิจารณาโดยไม่มีการทำซับเน็ตแล้วจะได้หมายเลขของเน็ตเวิร์คคือ 161.246 และหมายเลขประจำเครื่องคือ 5.13 แต่ผลที่ได้จากการทำซับเน็ตจะได้หมายเลขเน็ตเวิร์คเป็น 161.246.5.0 และหมายเลขเครื่องคือ 13 หรืออาจกล่าวได้ว่า คอมพิวเตอร์เครื่องนี้มีหมายเลขเครื่องเท่ากับ 13 และอยู่บนเครือข่ายย่อยหมายเลข 161.246.5

2.6 หมายเลขพอร์ต

เนื่องจากในเวลาใดๆสามารถมีโพรเซสของผู้ใช้สามารถใช้ยูดีพีหรือทีซีพี ได้พร้อมกันหลายๆโพรเซส ดังนั้นจึงต้องมีวิธีแยกแยะว่าข้อมูลเป็นของโพรเซสใด ซึ่งวิธีที่ทีซีพีและยูดีพีใช้ คือการใช้หมายเลขพอร์ต เมื่อโพรเซสของเครื่องไคลเอนต์ต้องการที่จะติดต่อกับเซิร์ฟเวอร์ ไคลเอนต์จะต้องติดต่อแต่ลำพังรู้หมายเลขอินเตอร์เนต 32 บิต เพียงอย่างเดียวนั้นไม่เพียงพอ เพราะว่าสามารถติต่อกับโฮสต์ได้เพียงอย่างเดียวแต่ไม่สามารถเจาะจงโพรเซสที่จะทำการติดต่อได้ ดังนั้นเพื่อแก้ปัญหานี้ทั้งทีซีพีและยูดีพี ได้มีการกำหนดหมายเลขพอร์ตมาตรฐาน (well-known ports) ซึ่งเป็นที่รู้จักกัน เช่น ทุกๆระบบทีซีพี/ไอพี จะกำหนดพอร์ต 23 เป็นพอร์ตบริการของเทลเน็ต เป็นต้น

เมื่อทีซีพีหรือยูดีพี กำหนดหมายเลขพอร์ตที่ไม่ซ้ำกันให้โพรเซสของผู้ใช้ เราเรียกหมายเลขพอร์ตนี้ว่าหมายเลขพอร์ตชั่วคราว (ephemeral port numbers) เมื่อไคลเอนต์เลิกใช้หมายเลขพอร์ตนี้แล้ว สามารถกำหนดหมายเลขพอร์ตนี้ให้ไคลเอนต์อื่นได้ โพรเซสที่ได้รับหมายเลขพอร์ตชั่วคราวนี้จะไม่สนใจว่ามีค่าเท่าไร แต่เป็นหน้าที่ของอีกโพรเซสหนึ่งที่ต่อกันที่ต้องสนใจ เพราะต้องส่งข้อมูลกลับมาที่พอร์ตนี้ในทีซีพี และ ยูดีพี นั้น หมายเลขพอร์ตตั้งแต่ 1-1023 เป็นพอร์ตที่สงวนไว้สำหรับหมายเลขพอร์ตมาตรฐาน

บทที่ 3
ไฟร์วอล

3.1 ประเภทของไฟร์วอล

3.1.1 เกตเวย์ไฟร์วอล (Gateway Firewall) หรือไฟร์วอล (Firewall)

ไฟร์วอลเป็นระบบหรือกลุ่มของระบบป้องกันที่บังคับใช้นโยบายการรักษาความปลอดภัยระหว่างเครือข่ายกับเครือข่าย หรือเครือข่ายกับอินเทอร์เน็ต ไฟร์วอลเป็นตัวกำหนดว่าบริการของเครือข่ายภายในชนิดใดบ้างที่เข้าถึงได้จากภายนอก และบริการภายนอกใดที่เข้าถึงได้จากผู้ใช้ภายใน ไฟร์วอลสามารถที่จะป้องกันการโจมตีจากภายนอกเครือข่ายได้ โดยจะกรองข้อมูล (หมายเลขไอพี สับเน็ต พอร์ต ฯลฯ) และอนุญาตให้ผู้ใช้ที่มีข้อมูลที่น่าไว้ใจเท่านั้น ผ่านเข้ามาในระบบเครือข่ายของเรา

วิธีการของไฟร์วอล จะจำกัดให้มีการผ่านเข้าออกได้ ที่จุดเดียว (Controlled point) และป้องกันผู้บุกรุกที่พยายามจะเข้ามาในเครือข่าย ดังนั้นการรับส่งข้อมูลทั้งหมดต้องผ่านไฟร์วอลซึ่งเป็นด่านสำหรับตรวจสอบแพ็คเก็ต[1] ไฟร์วอลมีหน้าที่ตัดสินใจว่าจะอนุญาตให้แพ็คเก็ตนั้นผ่านไปหรือไม่ ซึ่งการตัดสินใจนี้ ขึ้นอยู่กับกฎเกณฑ์ที่กำหนดไว้ สอดคล้องกับนโยบายขององค์การ

สิ่งที่ไฟร์วอลสามารถทำได้

- เป็นจุดสำคัญของการตัดสินใจเพื่อรักษาความปลอดภัย เนื่องจากเป็นจุดเดียวที่เครือข่ายติดต่อกับเครื่องภายนอกเครือข่าย
- สามารถตรวจสอบ และเก็บรายละเอียดกิจกรรมต่างๆ ระหว่างเครือข่ายภายใน และเครือข่ายภายนอก เพราะในการติดต่อทุกครั้งต้องผ่านไฟร์วอล
- สามารถกำหนดกฎเกณฑ์ นโยบายในการอนุญาต หรือไม่อนุญาตในการใช้บริการต่างๆ ภายในเครือข่าย
- มีการตรวจตราบริการต่างๆ ทำงานได้อย่างถูกต้อง

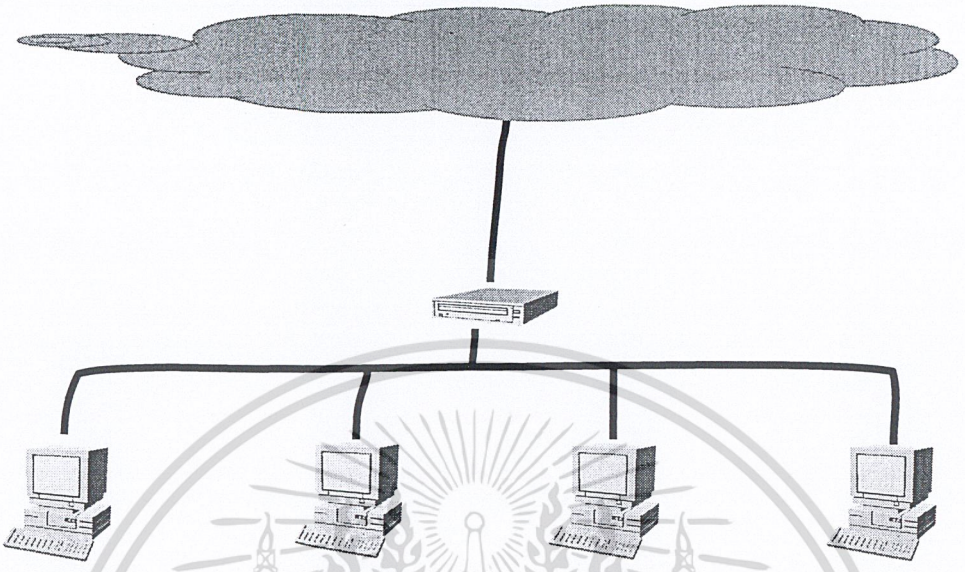
สิ่งที่ไฟร์วอลไม่สามารถทำได้

- ไฟร์วอลไม่สามารถป้องกันผู้บุกรุกที่อยู่ภายในเครือข่าย(Internal Network) เปรียบเสมือนการล็อคประตูบ้าน โดยที่โจรเข้ามาในบ้านแล้ว
- ไฟร์วอลไม่สามารถป้องกันการโจมตีที่ไม่ได้ผ่านไฟร์วอล เช่น ผู้ใช้ภายในเครือข่ายมีการเชื่อมต่อกับอินเทอร์เน็ตในทางอื่น ซึ่งไม่ผ่านไฟร์วอล โดยที่ผู้ดูแลระบบไม่รับทราบ เช่น การ Dial-up ไปยังอินเทอร์เน็ตจากเครื่องคอมพิวเตอร์ส่วนตัวที่อยู่ภายในเครือข่าย เปรียบได้กับเราล็อคประตูบ้านเรียบร้อยแล้ว แต่มีคนในบ้านเปิดหน้าต่างทิ้งไว้

---

[1] รูปแบบของข้อมูลที่ใช้ในการรับ-ส่งในเครือข่าย คือ มีส่วนของที่อยู่ผู้ส่ง-ผู้รับ พอร์ตผู้ส่ง-ผู้รับ และโพรโตคอลที่ใช้ รวมทั้งเนื้อหาของข้อมูล

- ไม่สามารถป้องกันไวรัสหรือ Trojan Horse[2] ได้ เพราะไฟร์วอลไม่สามารถตรวจสอบรายละเอียดข้อมูลที่อยู่ภายในแพ็คเก็ตว่ามีไวรัสหรือไม่



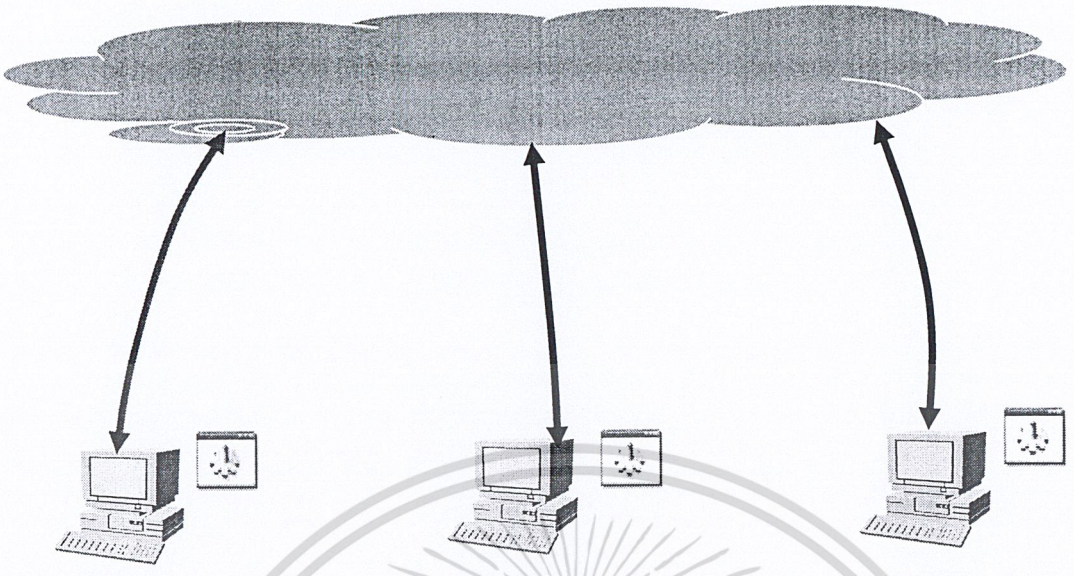
รูปที่ 3-1 เกตเวย์ไฟร์วอล

3.1.2 เพอร์ซันนอลไฟร์วอล

เพอร์ซันนอลไฟร์วอล (Personal Firewall) เป็นซอฟต์แวร์แอปพลิเคชั่นออกแบบมาเพื่อผู้ใช้ทั่วไปที่มีการเชื่อมต่อแบบ "always-on" อย่างเช่น ดีเอสแอล หรือ เคเบิลโมเด็ม โดยพื้นฐานการทำงานเป็นลักษณะเดียวกับเกตเวย์ไฟร์วอล คือ ตรวจสอบข้อมูลเข้าออก ต่างกันที่เพอร์ซันนอลไฟร์วอล จะทำงานบนเครื่องคอมพิวเตอร์เครื่องหนึ่ง ไม่ได้ทำงานในระดับเครือข่าย มีลักษณะการทำงานคล้ายโปรแกรมประเภทป้องกันไวรัส เพอร์ซันนอลไฟร์วอลจึงได้รับความสนใจ เนื่องจากความเสี่ยงของการถูกบุกรุกในปัจจุบันนั้นเพิ่มมากขึ้น

---

[2] เป็นโปรแกรมที่แอบแฝงมาในคราบของโปรแกรมปกติ แต่โปรแกรมประเภทนี้ยังทำหน้าที่อื่นแอบแฝงโดยที่เราไม่รู้ตัว เช่น ดักจับรหัสผ่าน เก็บข้อมูลการกดแป้นพิมพ์ เป็นต้น รวมไปถึงที่เกี่ยวข้องกับพอร์ตคือ เมื่อถึงเวลาที่กำหนดไว้ โทรจันเหล่านี้ ก็จะอาศัยพอร์ตใดพอร์ตหนึ่งในการแอบส่งข้อมูลออกไปยังจุดหมายปลายทางที่ใดที่หนึ่ง



รูปที่ 3-2 เพอร์ซันนอลไฟร์วอล

3.2 รูปแบบการทำงานของไฟร์วอล

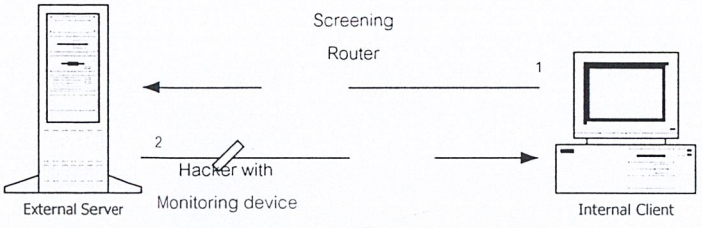
เราสามารถแบ่งรูปแบบการทำงานของไฟร์วอลออกได้เป็น 3 ประเภท ตามกลวิธีในการป้องกันเครือข่ายออกจากเครือข่ายอื่นๆ สำหรับการทำงานบนเราเตอร์ในเลเยอร์ระดับล่าง จะใช้วิธีการกรองแพ็คเก็ต โดยอาศัยข้อมูลในส่วนเฮดเดอร์ เรียกว่า "แพ็คเก็ตฟิลเตอร์ริ่ง" (Packet Filtering) ส่วนการทำงานในเลเยอร์ระดับสูง ซึ่งจะมีพร็อกซีเซิร์ฟเวอร์คอยตรวจสอบเนื้อหาภายในแพ็คเก็ตและแสดงผลการตรวจสอบ เรียกว่า "พร็อกซีเซิร์ฟเวอร์เกตเวย์" และประเภทสุดท้าย จะเก็บสถานะการทำงานไว้เป็นลำดับขั้น คือ "สเตทฟูลอินสเป็คชัน"

3.2.1 แพ็คเก็ตฟิลเตอร์ริ่ง

เป็นวิธีที่ใช้กันอย่างแพร่หลาย หลักการทำงานคือ ไฟร์วอลจะกรองแพ็คเก็ตโดยพิจารณาแพ็คเก็ตที่เข้าออกตามกฎที่ตั้งไว้ การตัดสินใจกรองแพ็คเก็ตเหล่านี้จะยึดข้อมูลที่อยู่ในเฮดเดอร์ของแพ็คเก็ตตัวนั้นๆ เช่น แอดเดรสต้นทาง แอดเดรสปลายทาง พอร์ตหรือโพรโตคอล เมื่อแพ็คเก็ตแต่ละแพ็คเก็ตเข้ามาสู่ไฟร์วอล จะนำมาเทียบกับกฎ หากเข้ากับกฎใดกฎหนึ่ง ก็จะดูว่ากฎนั้นสั่งให้ส่งแพ็คเก็ตต่อไปหรือดร็อปแพ็คเก็ตนั้นทิ้งไป และหากไม่เข้ากับกฎใดเลย ก็จะพิจารณาว่าค่าดีฟอลต์เป็นอะไร ให้ส่งหรือดร็อป

แพ็คเก็ตฟิลเตอร์ริ่ง เป็นการทำงานในชั้นเน็ตเวิร์ก วิธีการนี้เป็นวิธีที่ง่ายและทำงานได้รวดเร็วที่สุด แต่จุดอ่อนคือ ความยากในการกำหนดกฎต่างๆ ให้รัดกุมและการติดต่อกันระหว่างโฮสต์ต้นทางและโฮสต์ปลายทางได้โดยตรง อาจส่งผลให้ข้อมูลต่างๆ ของโฮสต์ปลายทาง รวมทั้งโฮสต์อื่นๆ ที่ติดต่ออยู่กับโฮสต์ปลายทางถูกโจมตีได้

ไฟร์วอลแบบนี้สามารถต่อต้านการโจมตีได้หลายชนิด เช่น IP spoofing , Source Routing Attack , Tiny Fragmentation Attack

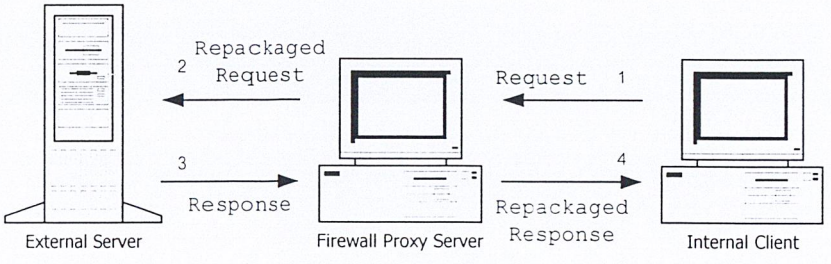


รูปที่ 3-3 รูปแบบการทำงานของแพ็คเก็ตฟิลเตอร์ริ่ง

3.2.2 พร็อกซีเซิร์ฟเวอร์เกตเวย์ (Proxy-Server Gateway)

พร็อกซีทำงานอยู่ในเลเยอร์ระดับสูง เป็นโปรแกรมแอปพลิเคชั่นที่ทำงานอยู่ระหว่างสองเครือข่าย โดยจะทำงานในลักษณะของการส่งข้อมูลต่อให้ เมื่อแพ็คเก็ตมาถึง พร็อกซีจะแยกข้อมูลส่วนที่เป็นเฮดเดอร์ของแพ็คเก็ตออก เหลือแต่ส่วนของข้อมูลในส่วนของชั้นแอปพลิเคชั่น เช่น หากเป็นข้อมูลเว็ป พร็อกซีจะแยกเฮดเดอร์ออกเหลือแต่ส่วนโพรโตคอล HTTP จากนั้นก็จะนำเฮดเดอร์มาพิจารณากับกฎต่างๆ ที่กำหนดเอาไว้ หากเป็นไปตามกฎที่ให้ส่งต่อ ก็จะนำข้อมูล HTTP นั้นมาประกอบเป็นแพ็คเก็ตขึ้นมาใหม่ แล้วส่งต่อตามฟังก์ชันการหาเส้นทาง

จากการที่พร็อกซี เพิ่มความสามารถในการติดตามและควบคุมการผ่านเข้าออกของแพ็คเก็ตระหว่างเครือข่าย คือ เมื่อผู้ใช้ภายในต้องการส่งข้อมูลไปยังเครื่องใดเครื่องหนึ่งในอินเทอร์เน็ต จะต้องส่งผ่านมาให้กับพร็อกซี เมื่อพร็อกซีได้รับและ ตรวจสอบว่าเป็นไปตามกฎแล้ว พร็อกซีจึงจะส่งข้อมูลต่อให้อินเทอร์เน็ตอีกทีหนึ่ง เสมือนกับเป็นการส่งมาจากผู้ใช้ภายในโดยตรง การทำงานแบบนี้ ทำให้การติดต่อระหว่างผู้ใช้ภายในกับภายนอก ไม่ต้องติดต่อกันโดยตรง ผู้ดูแลระบบสามารถมองเห็นเหตุการณ์ที่เกิดขึ้นบริเวณเกตเวย์ แต่พร็อกซีจะต้องรับภาระอย่างหนัก ทำให้ประสิทธิภาพในการติดต่อระหว่างเครือข่ายลดลง ทำงานช้า แต่มีความปลอดภัยมากกว่าเนื่องจากมีการพิจารณาข้อมูลถึงระดับแอปพลิเคชั่น ทั้งนี้ไฟร์วอลจะส่งต่อได้เฉพาะโพรโตคอลที่ไฟร์วอลรู้จักเท่านั้น



รูปที่ 3-4 รูปแบบการทำงานของพร็อกซีเซิร์ฟเวอร์เกตเวย์

พร็อกซีเซิร์ฟเวอร์เกตเวย์แบ่งออกได้เป็น 2 ประเภท

เซอร์กิต-เลเวล เกตเวย์ (Circuit – Level Gateway)

เป็นพร็อกซีที่ทำการควบคุมการติดต่อกันระหว่างเครือข่ายภายในและภายนอกโดยไม่มีช่องว่าง กล่าวคือ จะมีวงจรเสมือน (Virtual Circuit) อยู่ระหว่างโฮสต์ภายในเครือข่ายกับพร็อกซีเซิร์ฟเวอร์ เมื่อมีการร้องขอการติดต่อ (Request) จากเครือข่ายภายใน แพ็คเก็ตจะถูกส่งให้วงจรเสมือน ผ่านไปยังพร็อกซีเซิร์ฟเวอร์ซึ่งจะส่งการร้องขอการติดต่อไปยังอินเทอร์เน็ต หลังจากทำการแปลงหมายเลขไอพีเรียบร้อยแล้ว ในทำนองเดียวกัน การตอบรับ(Response)จากอินเทอร์เน็ต จะถูกส่งมายังพร็อกซีเซิร์ฟเวอร์ผ่านวงจรเสมือน ก่อนส่งกลับให้โฮสต์ต้นทาง โดยในการติดต่อนี้เครือข่ายภายนอกจะไม่สามารถมองเห็นโฮสต์ใดๆ ภายในเครือข่ายได้เลย การติดต่อแบบนี้จะใช้ในกรณีที่ผู้ใช้ภายในเครือข่ายกับอินเทอร์เน็ตไว้ใจได้เท่านั้น

แอปพลิเคชั่น-เลเวล เกตเวย์ (Application – Level Gateway)

สำหรับแอปพลิเคชั่นเกตเวย์ นอกจากจะทำงานเช่นเดียวกับวงจรเสมือนแล้ว ยังเพิ่มความสามารถในการตรวจสอบแพ็คเก็ตด้วย ไม่ว่าจะเป็นส่วนของเฮดเดอร์หรือเนื้อหาข้อมูลภายในแพ็คเก็ต เพื่อหยุดยั้งการส่งข้อมูลจากภายนอก หากมีข้อมูลจากแฮ็กเกอร์แอบซ่อนอยู่ภายในแพ็คเก็ต นอกจากนี้แอปพลิเคชั่นเกตเวย์ยังสนับสนุนการให้บริการ สำหรับโพรโตคอลแบบต่างๆ คือ เทลเน็ต , เอฟทีพี , เอชทีทีพี และ เอสเอ็มทีพี โดยที่ผู้ดูแลระบบจะต้องทำการติดตั้งในพร็อกซีแยกสำหรับแต่ละการให้บริการ

3.2.3 สเตทฟูลอินสเป็คชั่นหรือไดนามิกแพ็คเก็ตฟิลเตอร์ริ่ง

เป็นรูปแบบการทำงานแบบใหม่ ซึ่งปรับปรุงมาจากแพ็คเก็ตฟิลเตอร์ริ่งอย่างเดิมซึ่งจะตัดสินใจโดยดูข้อมูลในส่วนของเฮดเดอร์เท่านั้น และจะพิจารณาเฉพาะแพ็คเก็ตนั้นๆ โดยไม่ได้คำนึงถึงแพ็คเก็ตก่อนหน้านี้ ส่วนสเตทฟูลอินสเป็คชั่นจะเอาข้อจำกัดนี้ ทั้งในส่วนของข้อมูลที่ใช้ในการตัดสินใจ โดยจะดูถึงข้อมูลในส่วนของ Payload ด้วย และจะคำนึงถึงส่วนของแพ็คเก็ตอื่นๆ ที่อยู่ก่อนหน้านี้ประกอบในการตัดสินใจ ทำให้ความสามารถในการตัดสินใจมากขึ้น เพราะรู้ข้อมูลมากขึ้น โดยเพิ่มตารางเก็บสถานะ เช่น เมื่อส่งข้อมูลให้อินเทอร์เน็ต ตารางสถานะจะเก็บหมายเลขพอร์ตต้นทาง หมายเลขพอร์ตปลายทาง การเก็บนี้เรียกว่า “Saving the state” เมื่อมีแพ็คเก็ตที่เป็นการตอบรับ ก็จะนำแพ็คเก็ตที่รับมานั้น เปรียบเทียบกับ “Saved state” ที่เก็บไว้ว่ามีข้อมูลตรงกัน

การทำงานของสเตทฟูลอินสเป็คชั่น จะมีการติดตามสถานะ(State) การทำงานของการเชื่อมต่อแบบ ทีซีพี ซึ่งเป็นผลให้สามารถดูรูปแบบการทำงานได้ทั้งกระบวนการ ไม่ได้ดูเพียงข้อมูลในแต่ละแพ็คเก็ต โดยสามารถดูลักษณะการเชื่อมต่อ การโต้ตอบของแต่ละโพรโตคอลที่มีลักษณะที่แตกต่างกัน โดยสามารถแยกแยะโพรโตคอลที่ถูกต้องกับโพรโตคอลที่ไม่ถูกต้องออกจากกันได้ นอกจากนี้สเตทฟูลอินสเป็คชั่นยังมีความปลอดภัยมากกว่า เพราะสามารถจะปิดพอร์ตที่มีหมายเลขมากกว่า 1,024 ได้ เนื่องจากการเชื่อมต่อแบบ ทีซีพี อย่างเช่นเว็บนั้น แม้เมื่อเริ่มแรกจะติดต่อกันโดยผ่านพอร์ต 80 แต่หลังจากที่ติดต่อกันแล้ว จะมีการใช้หมายเลขพอร์ตแบบสุ่ม โดยมีหมายเลขมากกว่า 1,024 ซึ่งทำให้ไฟร์วอล

แบบแพ็คเก็ตฟิลเตอร์ริ่งจำเป็นต้องเปิดพอร์ตที่มีหมายเลขมากกว่า 1,024 ไว้ตลอดเวลา แต่สเตทฟูลจะเปิดพอร์ตเฉพาะเวลาที่มีการเชื่อมต่อผ่านพอร์ตนั้นๆ เท่านั้น หากแพ็คเก็ตถูกตรวจสอบแล้วว่าไม่เป็นไปตามกฎก็จะปิดพอร์ตนั้นทันที สำหรับข้อจำกัดของสเตทฟูลอินสเป็คชั่น คือ ไม่รู้จักการทำงานในระดับแอปพลิเคชั่น และยังคงเป็นการติดต่อระหว่างผู้ใช้ภายในกับภายนอกโดยตรง

ดังนั้นไฟร์วอลที่ดี จึงมักจะนำวิธีการหลายๆวิธีเข้ามาใช้ด้วยกัน เช่น ใช้พร็อกซีเซิร์ฟเวอร์และและสเตทฟูลอินสเป็คชั่นทำงานร่วมกัน

	Packet Filter	**Stateful Inspection**	**พร็อกซีเซิร์ฟเวอร์Gateways**
ข้อดี	• ประสิทธิภาพดี • ง่ายในการ implement • ไม่ขึ้นกับแอปพลิเคชั่น (Application Independent)	• ประสิทธิภาพดี • เปิดพอร์ตเฉพาะเมื่อมีการติดต่อ • สนับสนุนเกือบทุกบริการ	• ไม่เปิดเผยหมายเลขไอพีภายใน • พิจารณาเนื้อหาของข้อมูลด้วย • มี User Authentication • เก็บรายละเอียด log ได้มาก
ข้อเสีย	• เปิดเผยหมายเลขไอพีภายใน • มีการเปิดช่องว่างทิ้งไว้ถาวร • No User Authentication • ใช้การเชื่อมต่อโดยตรงกับภายนอก	• No User Authentication • ใช้การเชื่อต่อโดยตรงกับภายนอก • เปิดเผยหมายเลขไอพีภายใน	• ประสิทธิภาพต่ำกว่า • ต้องมีพร็อกซี สำหรับทุกๆแอปพลิเคชั่นที่ใช้ • ไม่มีการป้องกันในระดับชั้นที่ต่ำกว่าชั้นแอปพลิเคชั่น • เปิดเผยระบบปฏิบัติการ

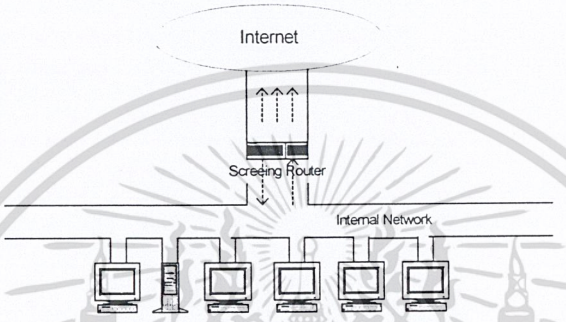
ตารางที่ 3-1 เปรียบเทียบรูปแบบการทำงานของไฟร์วอลทั้ง 3 ประเภท

3.3 สถาปัตยกรรมของไฟร์วอล

3.3.1 ซิงเกิลบ็อกซ์ (Single Box)

3.3.1.1 สกรีนนิ่ง เราเตอร์ (Screening Router)

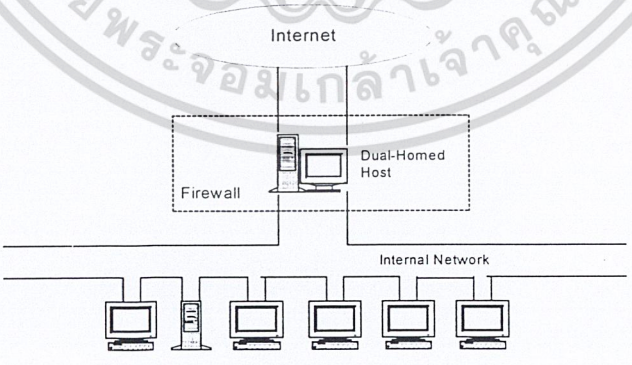
สามารถฟิลเตอร์แพ็คเก็ตได้ตามกฎที่ตั้งไว้ โดยดูจากข้อมูลในส่วนของเฮดเดอร์ของไอพีเท่านั้น เช่น จะอนุญาตหรือไม่อนุญาต พอร์ตหรือไอพีแอดเดรสใด ไฟร์วอลแบบนี้เป็นการเชื่อมต่อระหว่างเครือข่ายภายในและเครือข่ายภายนอก ดังนั้นหากมีการโจมตีที่ไฟร์วอล ซึ่งเป็นตัวหน้าด่านเพียงตัวเดียว หากการโจมตีทะลุผ่านไฟร์วอลก็จะสามารถเข้ามาในเครือข่ายภายในได้



รูปที่ 3-5 สกรีนนิ่ง เราเตอร์

3.3.1.2 ดูอัล – โฮม โฮสต์ (Dual-Home Host)

เป็นคอมพิวเตอร์ที่เชื่อมต่ออยู่กับเครือข่ายอย่างน้อย 2 เครือข่าย สามารถทำหน้าที่เป็นเราเตอร์ได้ แต่จะไม่อนุญาตให้มีการติดต่อผ่านกันโดยตรงโดยไม่ผ่านดูอัล-โฮมโฮสต์ ทำให้คอมพิวเตอร์ต้องรับภาระหนัก จึงเหมาะกับเครือข่ายที่มีข้อมูลเข้าออกน้อยๆ เนื่องจากสามารถรักษาความปลอดภัยได้มากกว่าสกรีนนิ่งเราเตอร์ธรรมดา

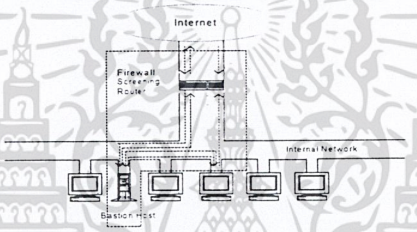


รูปที่ 3-6 ดูอัล-โฮม โฮสต์

3.3.2 มัลติเพิล-เพอโพส บ็อกซ์

3.3.2.1 สกรีนนิ่งโฮสต์ อาร์คิเทคเจอร์ (Screen Host Architecture)

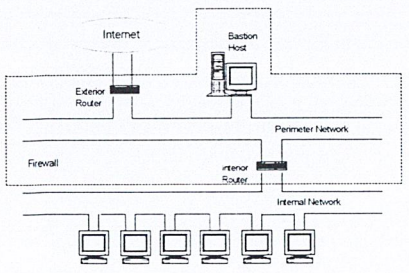
ประกอบไปด้วยไฟร์วอลประเภทแพ็คเก็ตฟิลเตอร์ริ่งและเบสชันโฮสต์(BastionHost) โดยเบสชันโฮสต์เป็นคอมพิวเตอร์พิเศษ ที่อนุญาตให้มีการเชื่อมต่อกับเครือข่ายภายนอก ซึ่งจะทำหน้าที่เป็นพร็อกซีที่จะส่งแพ็คเก็ตต่อไปยังเครื่องอื่นๆในเครือข่าย ดังนั้นหากมีการบริการใดๆในระบบ เบสชันโฮสต์จะต้องรู้จักโพรโตคอลนั้นๆ เบสชันโฮสต์จะต้องมีการพิสูจน์สิทธิ์(Authentication) ที่เหมาะสม โดยในการเริ่มใช้งาน แพ็คเก็ตจะถูกฟิลเตอร์โดยแพ็คเก็ตฟิลเตอร์ริ่งเราเตอร์ก่อนตามกฎที่กำหนดว่าควรจะอนุญาตหรือไม่ หากอนุญาตจะต้องขอพิสูจน์สิทธิ์กับเบสชันโฮสต์ แล้วให้เบสชันโฮสต์เป็นตัวกลางจัดการเชื่อมต่ออีกครั้งหนึ่ง ข้อเสียของโครงสร้างแบบนี้ขึ้นอยู่กับการทำงานของฟิลเตอร์ริ่งเราเตอร์ หากมีการทำงานผิดพลาด เบสชันโฮสต์จะให้บริการทุกการเชื่อมต่อ ส่งผลให้เครือข่ายภายในถูกโจมตีได้ จึงไม่ควรให้บริการที่มีความเสี่ยงสูง เช่น เว็บเซิร์ฟเวอร์



รูปที่ 3-7 สกรีนนิ่งโฮสต์ อาร์คิเทคเจอร์

3.3.2.2 สกรีนสับเน็ตอาร์คิเทคเจอร์- (Screened Subnet Architecture)

ต่างจากสถาปัตยกรรมสกรีนนิ่งโฮสต์ ตรงที่ส่วนของเบสชันโฮสต์จะมีการแบ่งเครือข่ายภายในออกจากเครือข่ายที่ให้บริการซึ่งให้ความปลอดภัยมากกว่า ในกรณีนี้จะมีไฟร์วอลถึงสองตัว คือ เราเตอร์ภายนอกและเราเตอร์ภายใน เราเตอร์ภายในจะทำหน้าที่ป้องกันไม่ให้เครือข่ายภายในถูกโจมตี เมื่อภายนอกถูกเจาะทะลุ นอกจากนั้นยังป้องกันไม่ให้โจมตีจากเครือข่ายภายในได้อีกด้วย นั่นคือภายในจะกรองทั้งแพ็คเก็ตที่มาจากอินเทอร์เน็ตและเบสชันโฮสต์ นอกจากนี้ยังมีการเพิ่มชั้นของเพอริมิเตอร์ขึ้นอีกระหว่างเราเตอร์ภายนอกและภายใน เพื่อป้องกันผู้ประสงค์ร้ายไม่ให้เข้าถึงเครือข่ายภายใน



รูปที่ 3-8 สกรีนสับเน็ตอาร์คิเทคเจอร์

นอกจากนี้ยังมีเครือข่าย DMZ (De-militarize zone) คือการวางโฮสต์ที่ไม่น่าเชื่อถือหรือมีความปลอดภัยต่ำ (Untrusted Host)ไว้ภายในไฟร์วอล แต่อยู่นอกเครือข่ายภายใน ซึ่งการทำเช่นนี้ไฟร์วอลจะทำการเชื่อมต่อกับสามเครือข่ายและการทำเช่นนี้เป็นการเพิ่มประสิทธิภาพ ความปลอดภัย ความน่าเชื่อถือและความสามารถในการนำโฮสต์ที่ไม่น่าเชื่อถือ แต่ไม่ได้เพิ่มระดับของความปลอดภัยในการเข้าถึงจากโฮสต์ที่อยู่ในเครือข่ายภายใน โฮสต์ที่ไม่น่าเชื่อถือมีจุดประสงค์ต่างๆกัน เช่น FTP Server และเว็บไซต์สาธารณะ จะถูกนำมาไว้ในเครือข่าย DMZ เป็นการสร้างเครือข่ายบริการสาธารณะ

3.4 การกำหนดกฎของไฟร์วอล

สำหรับการกำหนดกฎโดยทั่วไป จะมี 2 แบบ คือ

- Default Deny Stance : อะไรที่ไม่ได้ระบุว่าอนุญาตถือว่าไม่อนุญาต
- Default Permit Stance : อะไรที่ไม่ได้ระบุว่าไม่อนุญาตถือว่าอนุญาต

โดยส่วนใหญ่แล้วจะเลือกวิธีการแบบที่ 1 เพราะในแง่ของความปลอดภัยถือว่าวิธีการที่ 1 มีความปลอดภัยสูงกว่า เนื่องจากวิธีการแบบที่ 2 เป็นการยากที่จะสามารถระบุกฎให้ครอบคลุมจุดอ่อนทั้งหมดได้
ฟิลด์ในเฮดเดอร์ที่สามารถนำมาใช้พิจารณาได้ มีดังนี้

- แอดเดรสต้นทางและแอดเดรสปลายทาง
- ส่วนขยายของไอพี
- โพรโตคอล เช่น ทีซีพี, ยูดีพีหรือไอซีเอ็มพี
- หมายเลขพอร์ตต้นทางและหมายเลขพอร์ตปลายทาง
- ชนิดของข้อความแจ้งเตือน
- ACK bit ใน ทีซีพีเซกเมนต์

ในที่นี้จะขอยกตัวอย่างในการกำหนดกฎเกณฑ์เพื่อง่ายต่อการเข้าใจ

Service Destination	Packet Direction	Source Address	Dest. Address	Packet Type	Source Port	Dest. Port	ACK Set
Outbound	Outgoing	Internal	External	TCP	Y	23	
Outbound	Incoming	External	Internal	TCP	23	Y	Yes
Inbound	Incoming	External	Internal	TCP	Z	23	
Inbound	Outgoing	Internal	External	TCP	23	Z	Yes

ตารางที่ 3-2 ประเภทของแพ็คเก็ตที่ให้บริการ Telnet

Y, Z เป็นพอร์ตที่ได้มาจากการสุ่ม ซึ่งมีค่ามากกว่า 1023

หากไม่ต้องการให้อนุญาตทุกแพ็คเก็ต ยกเว้น outgoing เทลเน็ตก็จะสามารถกำหนดกฎเกณฑ์ได้ดังนี้

Rule	Direction	Source Address	Dest. Address	Protocol	Source Port	Dest. Port	ACK Set	Action
A	Out	Internal	Any	TCP	>1023	23	Either	Permit
B	In	Any	Internal	TCP	23	>1023	Yes	Permit
C	Either	Any	Any	Any	Any	Any	Either	Deny

ตารางที่ 3-3 ตัวอย่างการกำหนดกฎที่อนุญาตเฉพาะการให้บริการเทลเน็ต

- A อนุญาตทุกแพ็คเก็ตเพื่อขอใช้บริการเทลเน็ต
- B อนุญาตแพ็คเก็ตตอบรับ เนื่องจากสามารถพิสูจน์ได้ว่า ACK บิต ถูกเซ็ท
- C เป็นดีฟอลต์ ถ้าข้อมูลของเฮดเดอร์ ไม่ตรงกับทั้ง A และ B แพ็คเก็ตจะไม่ได้รับอนุญาตให้ผ่านเข้ามาตามกฎ C

บางครั้งการกำหนดกฎที่คลุมเครือ อาจทำให้มีความขัดแย้งกันของกฎ

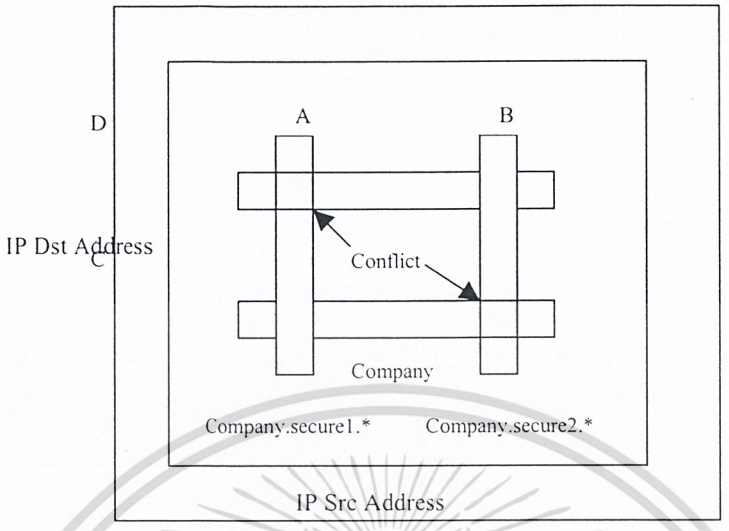
การตรวจสอบและแก้ไขการตั้งกฎที่ขัดแย้งกันของแพ็คเก็ตฟิลเตอร์ริ่ง

สมมติให้หมายเลขไอพีของบริษัทหนึ่งขึ้นด้วย Z โดยภายในบริษัทแบ่งเป็นเครือข่ายย่อย 2 เครือข่ายคือ Z.Secure1 และ Z.Secure2 มีการกำหนดกฎดังนี้

Rule	SrcAddr	DstAddr	Protocol	Action
A	Z.Secure1.*	Z.*	TCP	Permit
B	Z.Secure2.*	Z.*	TCP	Permit
C	Z.*	Z.secure1.*	TCP	Deny
D	Z.*	Z.secure2.*	TCP	Deny

ตารางที่ 3-4 แสดงการกำหนดกฎของบริษัท Z

จากกฎข้างต้นนี้จะได้ว่า



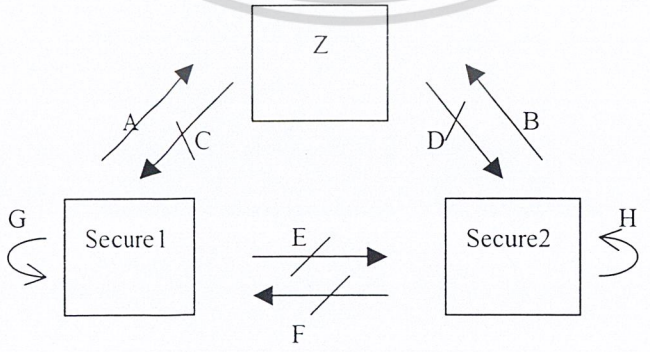
รูปที่ 3-9 แสดงความขัดแย้งของกฎ

หาก Secure1 จะติดต่อไปยัง Secure 2 หรือ Secure2 จะติดต่อไปยัง Secure1 ไม่มีการกำหนดอย่างชัดเจนว่าอนุญาตหรือไม่

หากเราต้องการกำหนดว่าเครือข่ายย่อยภายใน ไม่อนุญาตให้มีการติดต่อระหว่างกัน เราจึงเพิ่มกฎเข้าไปดังนี้

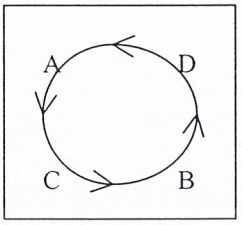
Rule	SrcAddr	DstAddr	Protocol	Action
E	Z.secure1.*	Z.secure2.*	TCP	Deny
F	Z.secure2.*	Z.secure1.*	TCP	Deny
G	Z.secure1.*	Z.secure1.*	TCP	Permit
H	Z.Secure2.*	Z.secure2.*	TCP	Permit

ตารางที่ 3-5 แก้ไขกฎเพื่อไม่ให้กฎมีความคลุมเครือ



รูปที่ 3-10 รูปแสดงผลของกฎของบริษัท Z

ผลจากการเพิ่มกฎ เพื่อที่จะให้กฎ E ทำงานได้อย่างถูกต้อง กฎ D จะต้องมาก่อนกฎ A กฎ C ต้องมาก่อนกฎ B เพื่อให้ได้กฎ F กฎ A มาก่อนกฎ C เพื่อให้ได้กฎ G และกฎ B จะต้องมาก่อนกฎ D จึงจะได้กฎ H ซึ่งทำให้เกิดเป็น ลูปขึ้น



รูปที่ 3-11 แสดงความสัมพันธ์ของกฎ

ถ้าเกิดเหตุการณ์ดังรูป คือ A -> C -> B -> D -> A ให้เราสมมติกฎใหม่ขึ้นมาเพื่อแก้การขัดแย้งกันระหว่าง A, D คือ E และให้ E มีความสำคัญมากกว่าทั้ง A และ D ได้ว่า A -> E และ D -> E จะทำให้ลูป A -> C -> B -> D -> A นั้นหายไป โดยมีอัลกอริทึมต่างๆ ดังนี้

1. อัลกอริทึมเพื่อตรวจสอบว่ามีการขัดแย้งของกฎเกิดขึ้นหรือไม่

```
Algorithm 2FilterConflict(F,G)
(* Determines if F and G conflict *)
for i = 1 to k do
    if F[i] and G[i] are disjoint
       return "No Conflict"
SetFlag = 1;
for i = 1 to k do
    if F[i] is not a prefix of G[i]
       Flag = 0;
if(Flag = 1) return "No Conflict";
SetFlag = 1;
for i = 1 to k do
    if G[i] is not a prefix of F[i]
       Flag = 0;
If(Flag = 1) return "No Conflict";
else return "Conflict";
end Algorithm
```

2. อัลกอริทึมในการคำนวณหากฎเพื่อแก้ไขการขัดแย้งระหว่างกฎ

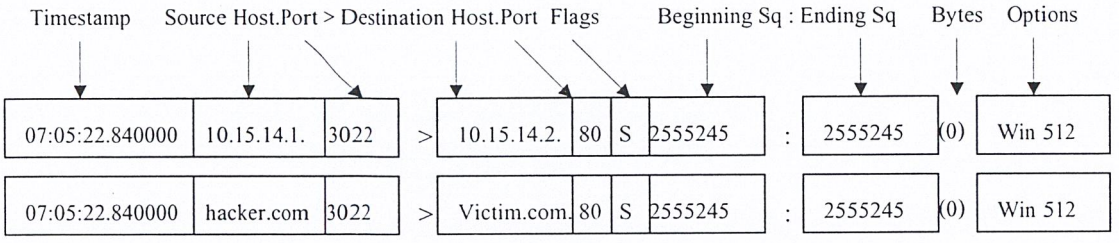
```
Algorithm ResolveFilter(F,G)
(* Computes the filter resolving the conflict of F,G *)
for i = 1 to k do
    let Xi be the longer of the two prefixes F[i] and G[i];
    return ( X1 , X2 , … , Xk);
end Algorithm
```

3. อัลกอริทึมในการเพิ่มกฎใหม่เพื่อแก้การขัดแย้งของกฎ

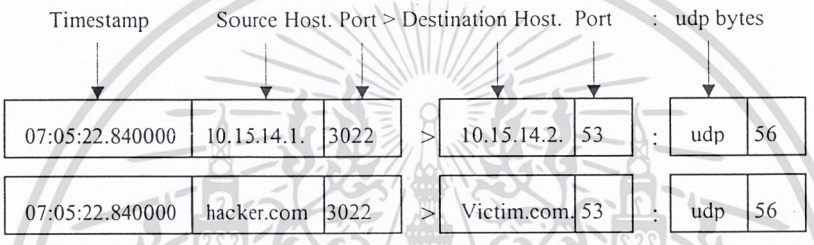
```
Algorithm AddNewFilter(F,B)
(* Insert a new filter into B *)
Initialize C(F) = {F};
for i = 1 to |B| do
     if F and Fi have a conflict then
        Add Fi to C(F);
for each filter F' ε C(F) do
     Add ResolveFilter(F,F') to B;
end  algorithm
```

3.5 วิธีการอ่านแพ็คเก็ตเพื่อนำมาใช้ฟิลเตอร์

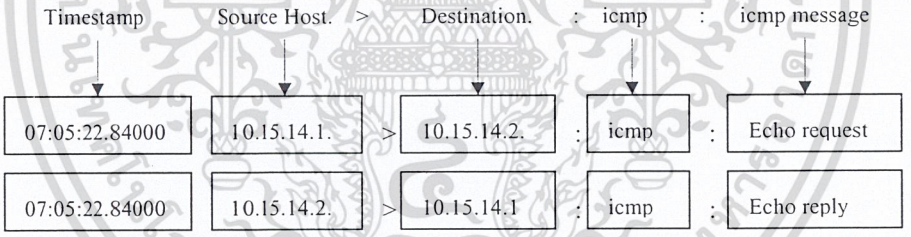
1. โพรโตคอล ทีซีพี/ไอพี



2. โพรโตคอล UDP



2. โพรโตคอล ICMP



3.6 ฟิลด์ที่นำมาใช้ในการฟิลเตอร์แพ็คเก็ต

Version	IHL	Type of Service	Total Length
Identification		Flags	Fragment Offset
TTL	**Protocol**		Header Checksum
Source IP Address			
Destination IP Address			
Option			Padding
Data			

โดยหากโพรโตคอลคือ ทีซีพี ก็จะนำส่วนเฮดเดอร์ของทีซีพีมาพิจารณาด้วย

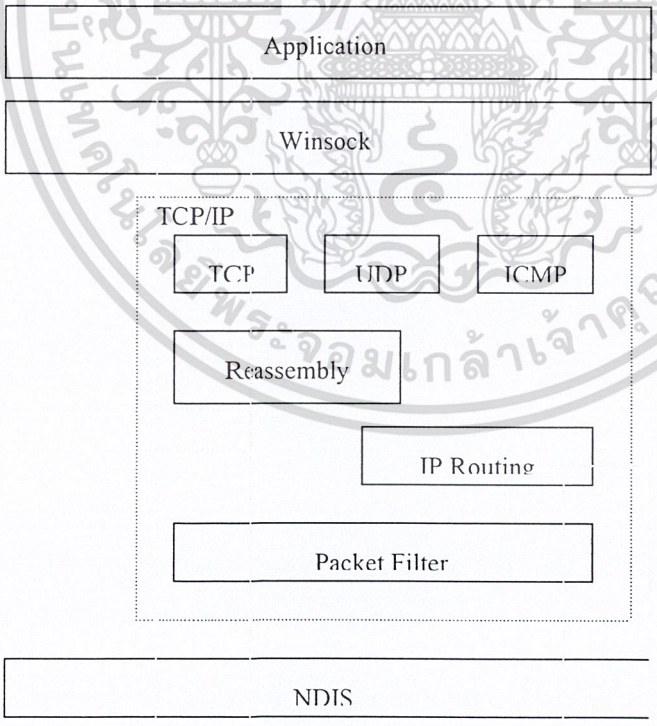
Source Port	**Destination Port**
Sequence No.	
Acknowledgement No.	
	Window
Checksum	Urgent Pointer
Options and Padding	

โพรโตคอลยูดีพี

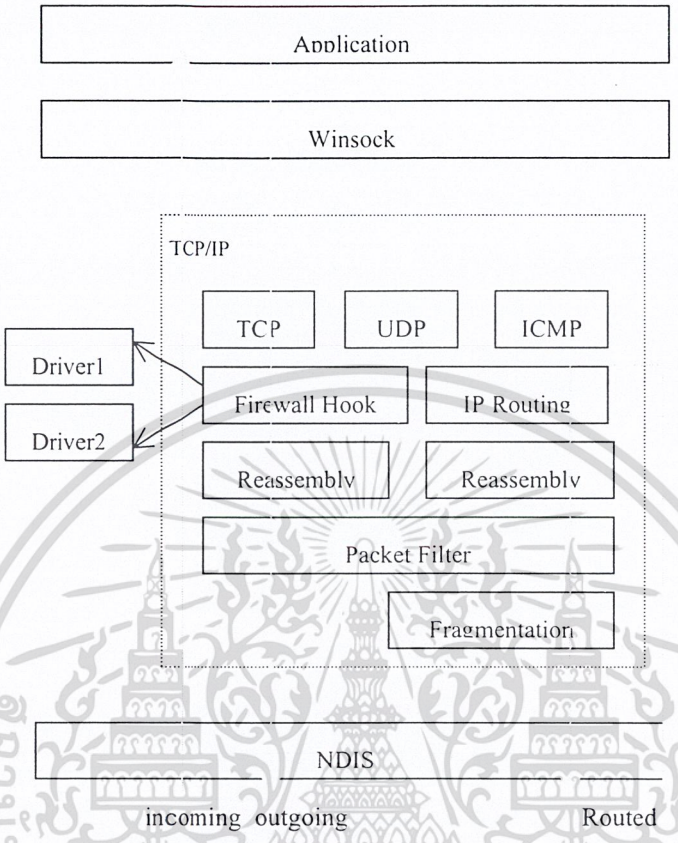
Source Port	**Destination Port**
Length	Checksum

3.7 เส้นทางเดินของแพ็คเก็ต

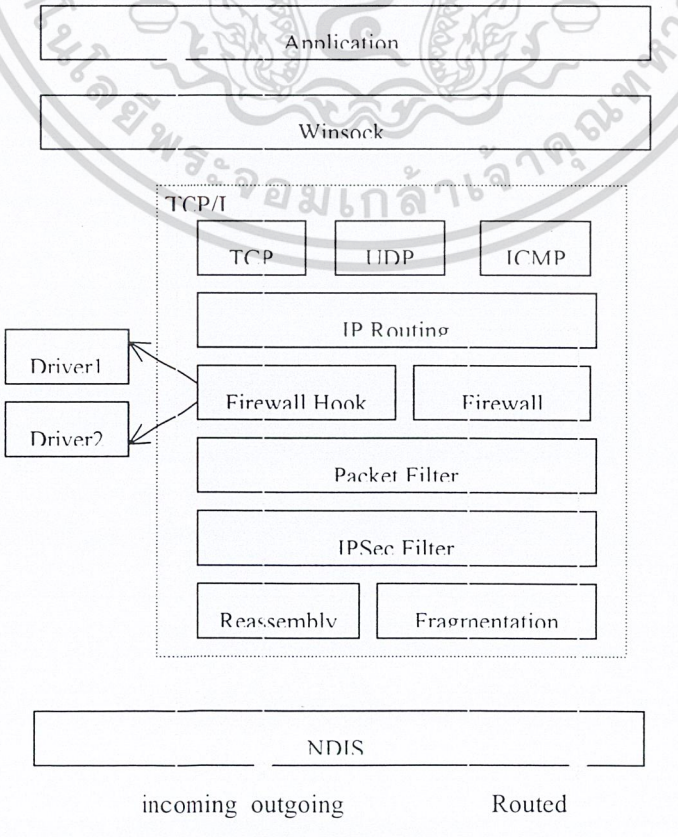
3.7.1 เส้นทางเดินของแพ็คเก็ตเมื่อไม่มีไฟร์วอลล์และIPSec



3.7.2 เส้นทางเดินของแพ็คเก็ตที่ผ่านไฟร์วอลฮุก แต่ไม่ผ่าน IPSec



3.7.3 เส้นทางของแพ็คเก็ตที่ผ่านไฟร์วอลฮุกและ IPSec



3.8 ภัยจากการโจมตีและความสามารถของไฟร์วอล

3.8.1 SYN flooding

เครื่องผู้บุกรุก ส่ง SYN Flag มายังเครื่องเป้าหมาย เพื่อทำ 3way-handshaking เมื่อเครื่องเป้าหมายได้รับ SYN Flag จะส่ง ACK และ SYN กลับไปยังเครื่องผู้บุกรุก ในขณะเดียวกันเครื่องเป้าหมายจะจองพื้นที่ในหน่วยความจำเพื่อรอรับ ACK ซึ่งหากผู้บุกรุกปลอมหมายเลขไอพีและส่ง SYN จำนวนมาก หน่วยความจำของเครื่องเป้าหมายจะถูกใช้จนหมด จนเครื่องเป้าหมายไม่สามารถให้บริการได้หรือใช้งานได้ชั่วขณะ

การป้องกันโดยไฟร์วอล

SYN flooding เป็นการโจมตีในเชิงปริมาณและความเร็ว ซึ่งกฎของไฟร์วอลไม่ครอบคลุมถึงปริมาณและความเร็วของข้อมูล การมีไฟร์วอลเป็นด่านหน้าจะช่วยบรรเทาความรุนแรงเนื่องจากการ SYN flood ได้ เพราะไฟร์วอลจะเป็นผู้รักษาการติดต่อไว้ทั้งหมดตั้งแต่เริ่ม SYN จนสิ้นสุด เท่านั้น

3.8.2 PING Attack

ไอซีเอ็มพีมีหน้าที่ส่งข่าวสารและคำสั่งควบคุมของไอพี โดยเฉพาะการรายงานข้อผิดพลาดในการรับส่ง PING เป็นข้อความไอซีเอ็มพีประเภทหนึ่งใช้ในการหาข้อมูลเกี่ยวกับเครือข่าย

การป้องกันโดยไฟร์วอล

ไม่ควรให้เครื่องภายนอกไฟร์วอลหาข้อมูลของเครื่องภายในเครือข่ายได้ โดยการกำหนดกฎ

Rule	Src Addr	Dst Addr	Protocol	Action
A	External	Internal	echo reply	permit
B	External	Internal	echo request	deny

3.8.3 Tiny Fragmentation

หากแพ็คเก็ตมีขนาดใหญ่เกินกว่าที่ชั้นดาต้าลิงก์อนุญาต โพรโตคอลไอพี มีความสามารถในการแบ่งแพ็คเก็ตออกเป็นแพ็คเก็ตย่อยๆ แล้วรวมกลับให้เหมือนเดิมเมื่อแพ็คเก็ตถึงปลายทาง เนื่องจากโพรโตคอล ทีซีพี จะมีเฮดเดอร์แพ็คเก็ตของตัวเอง และถูกเติมเข้ากับเฮดเดอร์ของโพรโตคอลไอพีก่อนส่งไปยังปลายทาง หากแพ็คเก็ตมีขนาดใหญ่ แพ็คเก็ตก็จะถูกแบ่งออกเป็นแพ็คเก็ตย่อย ซึ่งเพียงแพ็คเก็ตแรกเท่านั้นที่บอกว่าแพ็คเก็ตนี้ถูกส่งมาจากหมายเลขไอพีใด หากเป็นหมายเลขไอพีที่ไม่ได้รับอนุญาตให้ผ่านไฟร์วอล ก็จะมีเพียงแพ็คเก็ตแรกแพ็คเก็ตเดียวเท่านั้นที่ถูกดร็อป หรืออาจจะไม่มีการดร็อปแพ็คเก็ตใดๆเลย หาก Sequence No มีการแก้ไขเป็น1 (ซึ่งปกติเป็น 0) เสมือนว่าไม่ใช่แพ็คเก็ตแรก

3.8.4 Port Scanning

การสแกนพอร์ตทุกครั้ง ผลลัพธ์ที่ได้คือรายละเอียดที่จะบอกได้ว่าโฮสต์ที่ถูกสแกนใช้ระบบปฏิบัติการอะไร มีแอปพลิเคชั่นใดทำงานอยู่บ้าง เปรียบเสมือนบอกผู้บุกรุกว่ามีช่องทางใดที่จะเจาะระบบเข้ามาได้

ผู้บุกรุกสามารถเจาะเข้าสู่ระบบได้ โดยอาศัยข้อบกพร่องของแอปพลิเคชั่น โดยข้อมูลเหล่านี้สามารถหาได้ตามเว็บไซท์ที่เป็นแหล่งชุมชนของผู้บุกรุก

การสแกนพอร์ตจะทำให้ผู้บุกรุกทราบได้ว่ามีแอปพลิเคชั่นใดทำงานอยู่ หลังจากนั้นจะทำการสแกนแบบเจาะลึก ก็จะทราบว่าแอปพลิเคชั่นนั้นเป็นโปรแกรมอะไร เวอร์ชั่นใด มีข้อบกพร่องอย่างไร และควรนำเครื่องมือ หรือเทคนิคใดในการเจาะระบบ

พอร์ตที่เปิดให้บริการ จึงเป็นสิ่งที่ดึงดูดใจผู้บุกรุกเสมอ เพราะบอกว่าได้เป้าหมายมีส่วนประกอบอะไร ทำหน้าที่อะไร มีความสำคัญแค่ไหน และมีจุดอ่อนจุดแข็งอย่างไร

การเปิดพอร์ต

เราสามารถกำหนดได้เพียงส่วนที่อยู่ในระดับไอพี คือ หมายเลขไอพี สับเน็ตมาสก์ และ เกตเวย์เท่านั้น จะไม่สามารถกำหนดได้ว่าจะเปิดปิดพอร์ตใดบ้าง การที่พอร์ตใดจะเปิดให้บริการเป็นเซิร์ฟเวอร์พอร์ตนั้นจะต้องมีแอปพลิเคชั่นทำงานอยู่บนพอร์ตนั้นเสมอ คือมีโปรแกรมที่รับหน้าที่โต้ตอบและจัดการการสื่อสารที่มายังพอร์ตนั้น จึงอาจเปรียบได้ว่าพอร์ตก็คือแอปพลิเคชั่น การที่มีพอร์ตเปิดอยู่ก็หมายถึงการมีแอปพลิเคชั่นทำงานอยู่

นอกจากแอปพลิเคชั่นจะเปิดพอร์ตเพื่อใช้งานแล้ว ระบบปฏิบัติการที่อาศัย ทีซีพี/ไอพี ก็จะต้องเปิดพอร์ตเพื่อใช้ในกิจการของระบบปฏิบัติการด้วย โดยที่ผู้ใช้ไม่รู้ตัว เพราะเป็นการใช้งานภานในของระบบปฏิบัติการและผู้ผลิตคิดว่าผู้ใช้ไม่จำเป็นต้องรู้ จึงทำให้ผู้ใช้ถูกบุกรุกจากพอร์ตเหล่านี้ด้วย ซึ่งเมื่อเริ่มใช้แอปพลิเคชั่นมาก เครื่องคอมพิวเตอร์ของเราก็จะเริ่มเปิดพอร์ตมากขึ้น ซึ่งเป็นการเปิดช่องทางให้ผู้อื่นติดต่อเข้ามาได้มากขึ้นตามไปด้วย

การปิดพอร์ต

การปิดพอร์ต คือ การไม่ยอมรับการติดต่อเข้ามายังพอร์ตนั้นๆ เช่นเดียวกับการเปิดพอร์ต เราไม่สามารถปิดพอร์ตนั้นโดยตรงได้ด้วยโอสต์ทั่วไป หากจะปิดพอร์ต จะต้องหยุดการทำงานของแอปพลิเคชั่นก่อนแล้วพอร์ตจะถูกปิดไปเอง หรือสามารถทำได้โดยผ่านไฟร์วอล เราเตอร์ หรืออุปกรณ์ Layer 4 Switch

การปิดพอร์ตไม่ใช่เรื่องยาก ปัญหาที่เกิดขึ้นเนื่องมาจากพอร์ตไม่ได้ปิด ด้วยสาเหตุต่างๆ

1. พอร์ตที่เปิดไว้โดยไม่ได้ตั้งใจ

การเปิดพอร์ตเป็นการเปิดแบบลอจิคัลและมองไม่เห็น ดังนั้นหากไม่ทำการตรวจสอบโอสต์ของเราให้ดี จะไม่ทราบว่ามีพอร์ตใดเปิดอยู่ ส่วนใหญ่เกิดจากแอปพลิเคชั่นอื่นๆ มาเปิดพอร์ตบนโฮสต์เราโดยที่เราไม่เคยติดตั้งเข้าไปด้วยเลย โดยแอปพลิเคชั่นเหล่านี้ได้มาตั้งแต่ขั้นตอนการติดตั้งระบบปฏิบัติการซึ่งผู้ผลิตคาดว่าผู้ใช้ต้องการใช้แอปพลิเคชั่นเหล่านั้น ผู้ใช้สามารถตรวจสอบว่ามีแอปพลิเคชั่นใดบ้างที่ทำงานอยู่โดยที่เราไม่ต้องการ ให้หยุดการทำงานของแอปพลิเคชั่นเหล่านั้น พอร์ตก็จะถูกปิดไปเอง

2. พอร์ตของระบบปฏิบัติการ

เป็นพอร์ตที่จำเป็นสำหรับระบบปฏิบัติการนั้นๆ หากไม่เปิดพอร์ตเปล่านี้ ระบบปฏิบัติการก็จะไม่สามารถทำงานได้อย่างสมบูรณ์ เช่น ไมโครซอฟท์ วินโดวส์ เอ็นที จะต้องใช้พอร์ต 135-139 ของ ทีซีพี ในการทำงาน พอร์ตประเภทนี้จะไม่สามารถปิดลงได้ เนื่องจากแอปพลิเคชั่นที่ใช้งานพอร์ตนั้นเป็นส่วนหนึ่งของระบบปฏิบัติการ ข้อเสียอย่างมากก็คือ นอกจากผู้ใช้จะไม่สามารถปิดพอร์ตเหล่านี้ได้ ยังเป็นการบอกผู้บุกรุกอีกด้วยว่าใช้ระบบปฏิบัติการอะไร และทำให้ผู้บุกรุกสามารถโจมตีได้ง่ายขึ้น

3. พอร์ตที่เปิดแบบสุ่ม

เกิดจากแอปพลิเคชั่นบางประเภทที่มีการใช้งานพอร์ตมากกว่า 1 พอร์ต โดยมีหมายเลขพอร์ตที่คงที่ไว้เป็นหลัก 1 พอร์ต ส่วนพอร์ตที่จะเปิดเป็นการชั่วคราวนี้ ไคลเอนต์และเซิร์ฟเวอร์ จะมีการตกลงกันเพื่อเปลี่ยนไปสื่อสารกันที่พอร์ตอื่นๆ ซึ่งการเปิดพอร์ตประเภทนี้มีปัญหาคือ

- พอร์ตจะปิดลงเมื่อการใช้งานสิ้นสุดลง แต่หากแอปพลิเคชั่นทำงานผิดพลาดหรือหยุดลงกลางคัน พอร์ตก็อาจจะถูกเปิดค้างทิ้งไว้
- การไม่มีหมายเลขพอร์ตแน่นอน ทำให้ควบคุมและตรวจสอบได้ยาก หากพอร์ตที่ใช้ทับซ้อนกับพอร์ตที่อันตรายซึ่งใช้โดยโปรแกรมประเภทโทรจัน
- หากมีการนำไฟร์วอลมาใช้งาน การกำหนดกฎสำหรับไฟร์วอลจะทำได้ยาก เพราะกฎของไฟร์วอลจะตั้งอยู่บนพื้นฐานของการใช้พอร์ตเป็นหลัก

การป้องกันโดยไฟร์วอล

สเตทฟูลอินสเปคชั่นจะเปิดพอร์ตเฉพาะเวลาที่มีการเชื่อมต่อผ่านพอร์ตนั้นๆเท่านั้น หากแพ็คเก็ตถูกตรวจสอบแล้วว่าไม่เป็นไปตามกฎก็จะปิดพอร์ตนั้นทันที

บทที่ 4

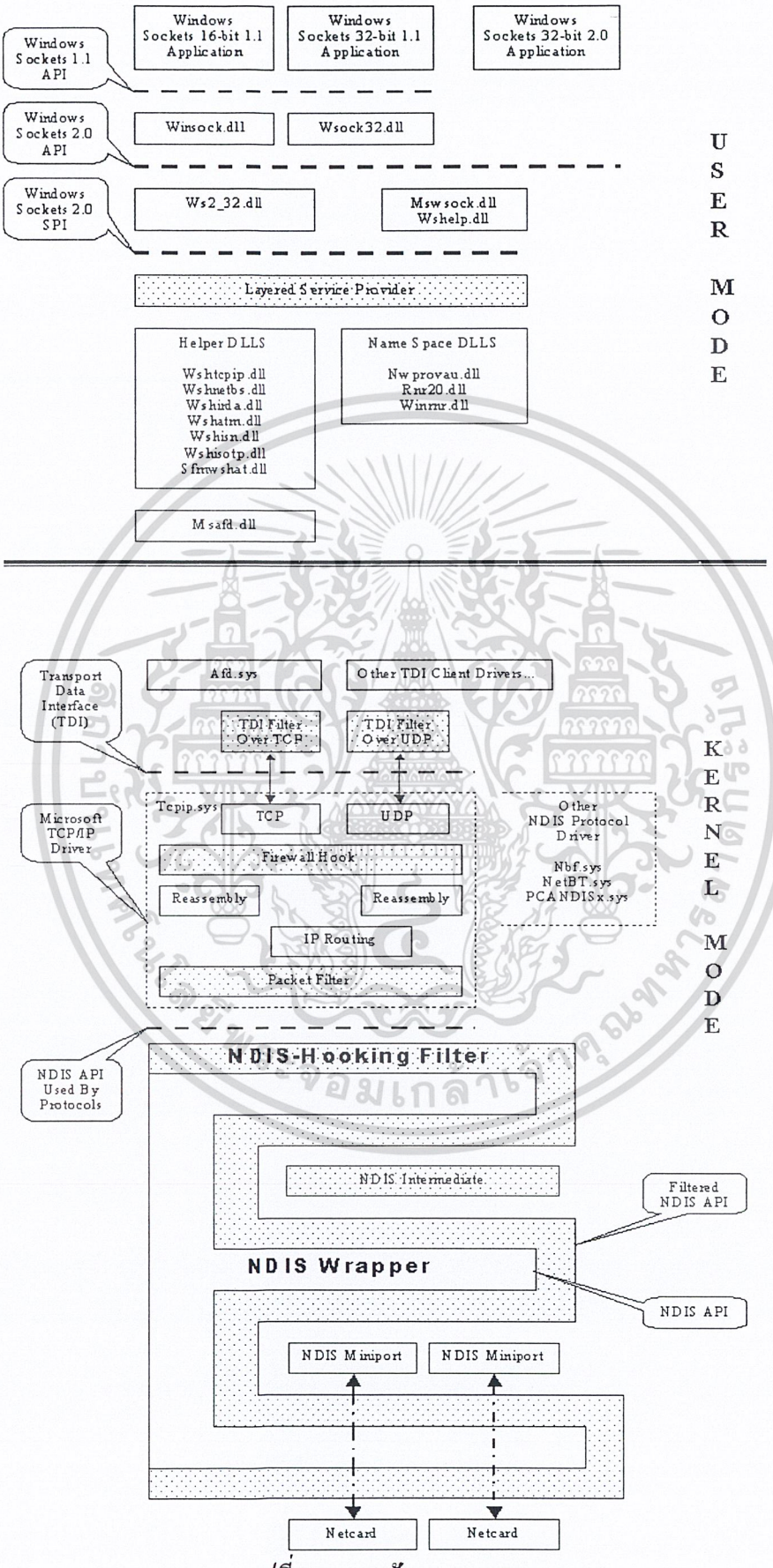
Windows Network Programming

ก่อนนี้การเขียนโปรแกรมที่เกี่ยวกับเน็ตเวิร์คบนแพลตฟอร์มของวินโดวส์เป็นเรื่องที่ไม่ง่ายนัก เนื่องจากจำเป็นที่จะต้องเขียนไดรเวอร์ เพื่อติดต่อและควบคุมลงไปถึง NIC (Network Interface Card) ซึ่ง NIC แต่ละอันก็จะแตกต่างกันไป ทำให้เป็นการยุ่งยากมากในการเขียนโปรแกรม

ไมโครซอฟต์ได้สร้างมาตรฐาน NDIS (Network Driver Interface Specification) ขึ้นมาซึ่งช่วยทำให้การเขียนโปรแกรมที่เกี่ยวกับเน็ตเวิร์คทำได้อย่างง่ายขึ้น

4.1 Network Driver Interface Specification

NDIS เป็นตัวกำหนดลักษณะการเชื่อมต่อของ NIC Driver , Protocol Driver ซึ่งอาจมีจำนวนมากกว่าหนึ่งไดรเวอร์ต่อหนึ่งระบบปฏิบัติการ โดย NDIS จะทำให้การพัฒนา Network driver เป็นไปได้โดยไม่จำเป็นต้องสนใจในฮาร์ดแวร์ข้างล่างอย่าง NIC ว่าจะเป็นอะไร เปรียบเสมือนเป็นการกำหนด API (Application Programming Interface) มาตรฐานสำหรับการเขียนโปรแกรมติดต่อกับ NIC เนื่องจาก NDIS จะเป็นผู้ที่เป็นตัวเชื่อมระหว่าง Network Driver กับ NIC เอง



รูปที่ 4-1 สถาปัตยกรรมของ NDIS

สำหรับส่วนประกอบหลักๆของ NDIS ก็มีดังต่อไปนี้

4.1.1 NIC Driver คือไดรเวอร์ที่ทำหน้าที่ติดต่อควบคุม NIC ซึ่งเป็นการติดต่อโดยตรงกับฮาร์ดแวร์ ในขณะเดียวกันก็ทำหน้าที่ให้บริการไดรเวอร์ที่อยู่ระดับสูงขึ้นไปในการรับ-ส่งแพ็คเก็ตหรือกระทำการต่างๆ กับ NIC โดย NICไดรเวอร์ มีอยู่ 2 ประเภท

- **Full NIC Driver** จะจัดการส่วนของงานหรือฟังก์ชันที่เจาะจงฮาร์ดแวร์และระบบปฏิบัติการซึ่งปกติเป็นหน้าที่ของ NDIS ซึ่งทำให้ Legacy Driver สามารถเรียกใช้ฟังก์ชันที่เป็นเฉพาะของฮาร์ดแวร์หรือระบบปฏิบัติการนั้นๆได้
- **Miniport Driver** ใช้ฟังก์ชันซึ่งไม่ขึ้นกับแพลตฟอร์มในการจัดการ จะไม่มีการเรียกใช้บริการจากระบบปฏิบัติการโดยตรง

4.1.2 Intermediate Protocol Driver จะอยู่ระหว่าง Legacy Protocol Driver กับ Miniport Driver โดยถ้ามองจาก Transport Driver ด้านบนลงมาจะดูเหมือนเป็น Miniport Driver แต่ถ้ามองจาก Miniport Driver จะดูเหมือน Protocol Driver จุดประสงค์ที่มี Intermediate Protocol Driver นี้สำหรับการแปลง Media Type ระหว่าง Transport Driver กับ Miniport Driver ซึ่งจัดการ Media type แบบใหม่ซึ่งไม่เป็นที่รู้จักของ Transport Driver

4.1.3 Upper Level Protocol Driver ประยุกต์ TDI Interface หรือ Interface เฉพาะอื่นๆเพื่อให้บริการแก่ผู้ใช้ อย่างเช่น การสร้างแพ็คเก็ต การเอาข้อมูลใส่เข้าไปในแพ็คเก็ตและส่งแพ็คเก็ตลงไปสู่ Driver ระดับล่างกว่าโดยเรียกผ่าน NDIS

4.2 การฟิลเตอร์แพ็คเก็ตบนวินโดวส์

สำหรับสถาปัตยกรรมของ NDIS นั้น ก็ได้อนุญาตให้เราสามารถที่จะเขียนโปรแกรมเพื่อเข้าไปฟิลเตอร์แพ็คเก็ตได้หลายวิธีโดยการเรียกใช้ฟังก์ชันการทำงานจากชั้นต่างๆของ NDIS ซึ่งแต่ละวิธีก็มีวิธีการ รูปแบบการใช้งาน และความสามารถในการทำงานที่แตกต่างกันไป มีทั้งในแบบ User-Mode และ Kernel-Mode

4.2.1 User-Mode Network Data Filtering

- Winsock Layered Service Provider (LSP) เป็นไปได้ที่เราสามารถที่จะเรียก Kernel-Mode TCP/IP Driver ผ่าน Transport Data Interface (TDI) โดยไม่จำเป็นต้องผ่าน Winsock แต่ในงานที่จำเป็นจะต้องตรวจสอบหรือกระทำการกับทุกๆแพ็คเก็ตนั้นไม่ควรที่จะอิงการทำงานของ Winsock LSP แต่ควรจะประยุกต์ใช้วิธีการ Kernel-mode มากกว่า

- Windows 2000 Packet Filtering Interface สำหรับวินโดว์ 2000 ได้มี API ที่อนุญาตให้แอปพลิเคชั่นหรือเซอร์วิสที่ทำงานใน User-mode สามารถที่จะกำหนดชุดของ Filter Descriptor ซึ่งจะถูกนำไปใช้โดยคอมโพเนนท์ทีซีพี/ไอพีในชั้นล่าง การฟิลเตอร์นี้จะดูที่หมายเลขไปพีต้นทาง ปลายทาง และหมายเลขพอร์ต โดยสามารถกำหนดได้เพียงแค่อนุญาตให้ผ่านหรือไม่ เท่านั้น

- Winsock Replacement DLL ก่อนที่จะมี LSP วิธีการเดียวที่จะเพิ่มความฟังก์ชันในการทำงานของ Winsock คือการสร้าง DLL ชุดใหม่มาทับ DLL ของ Winsock ซึ่งก็สามารถประยุกต์ใช้ในการฟิลเตอร์แพ็คเก็ตได้ แต่ไม่เป็นที่นิยมเนื่องจากความยากในการทำ อีกทั้งยังต้องยุ่งเกี่ยวกับฟังก์ชันการทำงานที่ไม่มีคู่มืออธิบายอีกด้วย

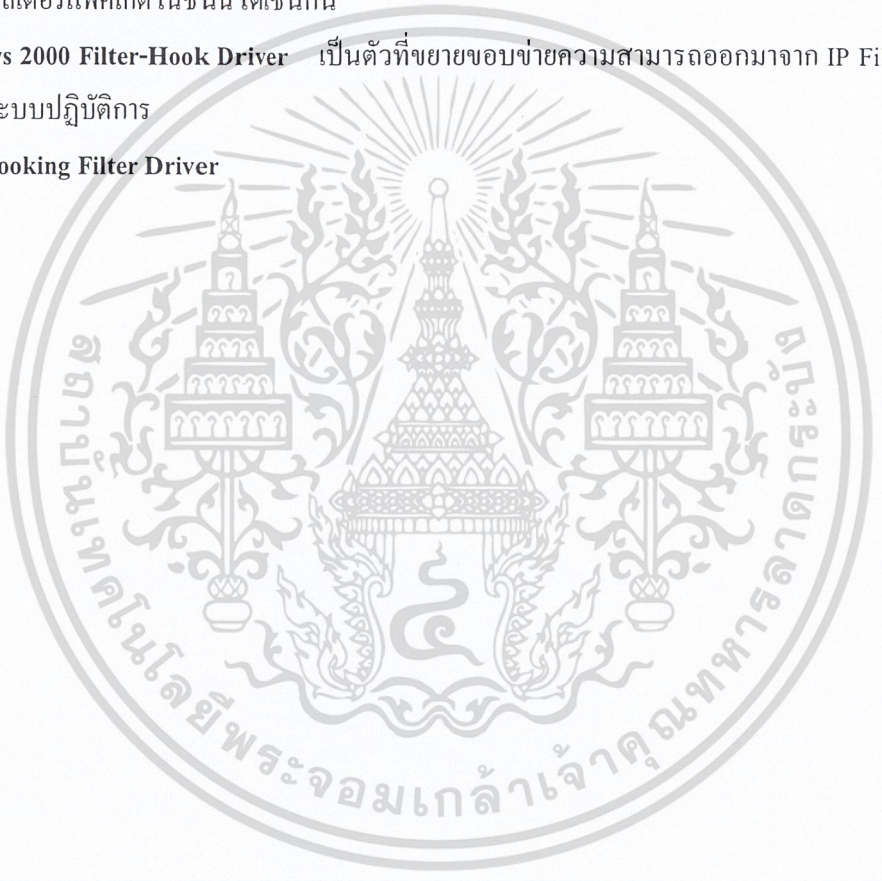
4.2.2 Kernel-Mode Network Data Filtering

- Transport Data Interface (TDI) Filter Driver ทำงานในชั้นของ TCP/IP Driver

- NDIS Intermediate Driver ทำงานอยู่ระหว่าง Miniport Driver กับ Transport Driver ซึ่งสามารถที่จะประยุกต์ฟิลเตอร์แพ็คเก็ตในชั้นนี้ได้เช่นกัน

- Windows 2000 Filter-Hook Driver เป็นตัวที่ขยายขอบข่ายความสามารถออกมาจาก IP Filter Driver ซึ่งมากับระบบปฏิบัติการ

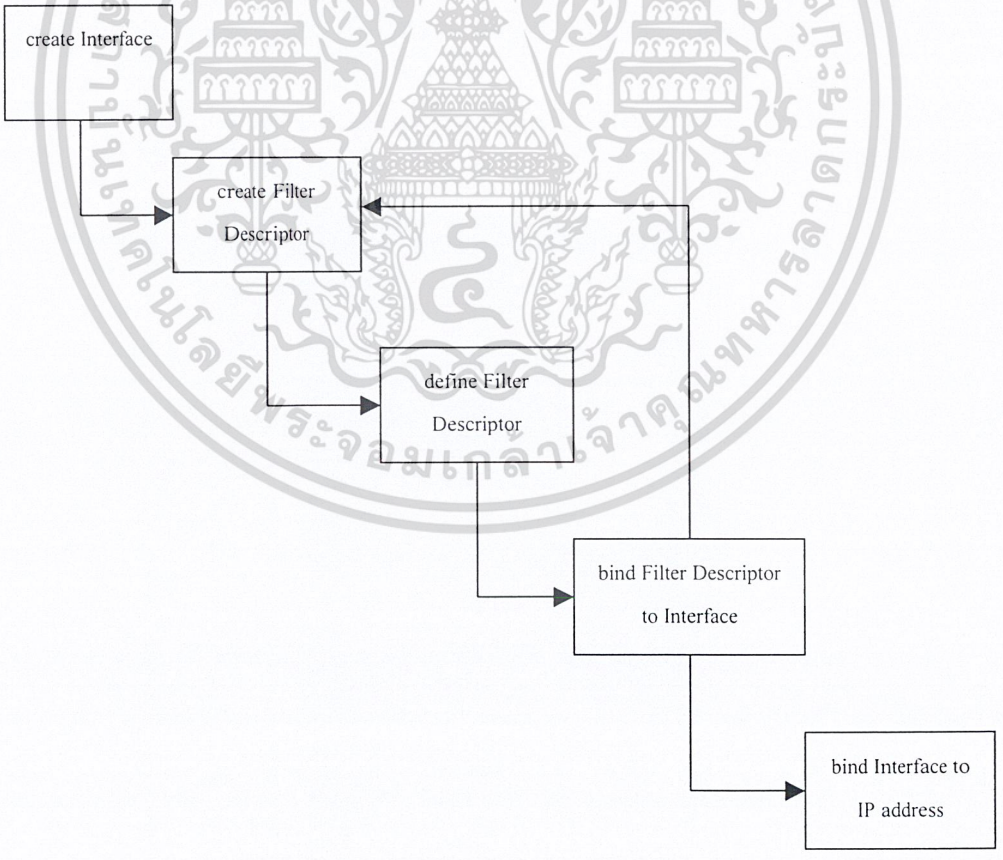
- NDIS Hooking Filter Driver



4.3 Windows 2000 Packet Filter API

จากการที่มีไลบรารีชุด iphlpapi.dll เข้ามาในวินโดวส์ 98 และวินโดวส์ 2000 นั้นได้ช่วยการทำงานที่เกี่ยวกับเน็ตเวิร์คให้ง่ายขึ้นหลายอย่าง ในอดีต แม้แต่การเรียกค่าหมายเลขไอพีของการ์ดเน็ตเวิร์คของเครื่องยังเป็นเรื่องที่ยุ่งยาก iphlpapi.dll (IP Helper API) จึงถูกสร้างขึ้นเพื่อแก้ไขข้อจำกัดเหล่านี้ เราสามารถใช้ API ตัวนี้ในการเรียกดูหมายเลขไอพี, routing table, ARP table entries, และสถิติทางเน็ตเวิร์ค โดยที่ในวินโดวส์ 98 จะสนับสนุนเพียงแค่บางส่วนของ iphlpapi.dll ส่วนวินโดวส์ 2000 นั้นสนับสนุนทั้งหมด โดย iphlpapi.dll นั้นเป็นส่วนหนึ่งของ Microsoft Platform SDK

หนึ่งใน interface ของ iphlpapi.dll นี้คือ Packet Filtering interface ซึ่งอยู่ภายใต้ Routing and Remote Access services ซึ่งมีในทุกเวอร์ชันของวินโดวส์ 2000 ซึ่ง interface นี้มีความสามารถในการฟิลเตอร์แพ็คเก็ต TCP/IP ซึ่งทำให้เราสามารถที่จะระงับการติดต่อกับคอมพิวเตอร์บางเครื่องหรือบริการบางอย่างได้ ซึ่งก่อนหน้านี้ การจะทำเช่นนี้ได้อย่างน้อยจะต้องเขียนไดรเวอร์ในชั้นของ NDIS ขึ้นมา แต่เมื่อมี API ตัวนี้แล้ว สามารถทำได้เพียงแค่เรียกฟังก์ชันเท่านั้น



รูปที่ 4-2 ขั้นตอนการทำงานของ Packet Filter API

ในการกำหนดกฎการฟิลเตอร์และผูกเข้ากับการ์ดเน็ตเวิร์ค จะต้องมี INTERFACE_HANDLE ก่อน ซึ่งสามารถได้จากฟังก์ชัน PfCreateInterface() ซึ่งฟังก์ชันนี้ยังใช้ในการกำหนดค่าดีฟอลต์ในการฟิลเตอร์แพ็คเก็ตสำหรับแพ็คเก็ต TCP/IP ขาเข้าและขาออกอีกด้วย เราสามารถกำหนดให้ interface ที่เราสร้างขึ้นมาบล็อคแพ็คเก็ตทั้งหมดยกเว้นเฉพาะแพ็คเก็ตที่ตรงกับข้อกำหนดของกฎการฟิลเตอร์ (PF_ACTION_DROP) หรือกำหนดให้แพ็คเก็ตทั้งหมดผ่านไปได้ยกเว้นเฉพาะแพ็คเก็ตที่ตรงกับข้อกำหนดของกฎการฟิลเตอร์ (PF_ACTION_FORWARD) เรายังสามารถกำหนดได้ว่า interface ที่สร้างขึ้นนั้นจะสามารถแชร์ได้หรือไม่ ซึ่งการแชร์ได้หมายถึงการที่อนุญาตให้โพรเซสอื่นๆสามารถเพิ่มหรือลบกฎการฟิลเตอร์ใน interface นี้ได้

หลังจากที่เราได้ handle จากฟังก์ชัน PfCreateInterface() แล้ว เราสามารถที่จะเพิ่มกฎการฟิลเตอร์เข้าไปได้โดยผ่านฟังก์ชัน PfAddFiltersToInterface() ซึ่งสามารถกำหนดได้ทั้งการฟิลเตอร์ทั้งขาเข้าและขาออก พารามิเตอร์หลักสำหรับฟังก์ชันนี้คือ PF_FILTER_DESCRIPTOR ซึ่งจะได้กล่าวถึงต่อไป หลังจากที่ได้กำหนดกฎการฟิลเตอร์ครบทั้งหมดแล้ว ขั้นตอนต่อไปคือการผูก interface handle เข้ากับการ์ดเน็ตเวิร์คโดยใช้หมายเลขไอพี การฟิลเตอร์จะทำงานและมีผลจนกว่าจะมีการเรียกฟังก์ชัน PfDeleteInterface()

PF_FILTER_DESCRIPTOR

Filter Descriptor นั้นเป็นรูปแบบโครงสร้างของกฎการฟิลเตอร์ ในการที่เราจะกำหนดกฎแต่ละข้อเราจะต้องสร้าง Filter Descriptor ขึ้นมาและกำหนดค่าต่างๆ โดยโครงสร้างของ PF_FILTER_DESCRIPTOR นั้นเป็นดังต่อไปนี้

```
typedef struct PF_FILTER_DESCRIPTOR {
  DWORD          dwFilterFlags;
  DWORD          dwRule;
  PFADDRESSTYPE  pfatType;
  PBYTE          SrcAddr;
  PBYTE          SrcMask;
  PBYTE          DstAddr;
  PBYTE          DstMask;
  DWORD          dwProtocol;
  DWORD          fLateBound;
  WORD           wSrcPort;
  WORD           wDstPort;
  WORD           wSrcPortHighRange;
  WORD           wDstPortHighRange;
} PF_FILTER_DESCRIPTOR, *PPF_FILTER_DESCIPTOR;
```

โดยมีรายละเอียดของแต่ละฟิลด์ดังนี้

dwFilterFlags ณ ขณะนี้มีเพียงแค่ค่าเดียวที่ถูกกำหนดขึ้นสำหรับฟิลด์นี้ คือ FD_FLAGS_NOSYN โดยเมื่อแอปพลิเคชั่นต้องการที่จะสร้างการเชื่อมต่อแบบ TCP จะต้องมีการส่ง synchronization request ซึ่ง flag นี้เป็นการห้ามไม่ให้ทำเช่นนั้น แต่จากการทดลองพบว่าไม่มีความแตกต่างในการกำหนดหรือไม่กำหนดค่าให้สำหรับ flag นี้

dwRule เป็นค่าที่ผู้ใช้กำหนดขึ้นซึ่งจะถูกสำเนาไปสู่ log (ถ้ามีการใช้งาน log) เพื่อแสดงว่ากฎการฟิลเตอร์ข้อไหนที่รับผิดชอบสำหรับแต่ละข้อของ log

pfatType กำหนดเวอร์ชันของไอพี ซึ่งสามารถกำหนดได้เป็น PF_IPV4 (IP เวอร์ชัน 4) หรือ PF_IPV6 (IP เวอร์ชัน 6)

SrcAddr หมายเลขไอพีต้นทาง

SrcMask เน็ตมาสก์ต้นทาง

DstAddr หมายเลขไอพีปลายทาง

DstMask เน็ตมาสก์ปลายทาง

หมายเลขไอพีต้นทาง ปลายทาง รวมถึงมาสก์ เป็นการกำหนดช่วงของหมายเลขไอพีที่กฎการฟิลเตอร์มีผล โดยมาสก์เป็นตัวกำหนดขอบเขตของแอดเดรสที่จะถูกทดสอบ และเมื่อแอดเดรสถูกกำหนดเป็นช่วง (โดยการกำหนดส่วนของมาสก์เป็นศูนย์) ส่วนของแอดเดรสที่สัมพันธ์กันก็ต้องถูกกำหนดให้เป็นศูนย์ด้วยเช่นกัน ตัวอย่างเช่น ถ้าต้องการกำหนดหมายเลขไอพีหมายเลขเดียว แอดเดรสจะต้องสมบูรณ์ และค่ามาสก์ก็จะต้องเป็น FF.FF.FF.FF ในทางกลับกัน ถ้าต้องการจะกำหนดฟิลเตอร์สำหรับช่วงของหมายเลขไอพี ก็ต้องกำหนดส่วนที่เป็นช่วงของแอดเดรสให้เป็นศูนย์เช่นเดียวกันกับส่วนเดียวกันสำหรับมาสก์ (แอดเดรส a.b.c.0 และมาสก์ FF.FF.FF.00 จะฟิลเตอร์หมายเลขไอพีทั้งหมดที่ขึ้นต้นด้วย a.b.c)

ในการกำหนดฟิลเตอร์ขาออก จะต้องกำหนดให้แอดเดรสของตัวเองเป็นแอดเดรสต้นทาง ส่วนการฟิลเตอร์ขาเข้าจะต้องกำหนดแอดเดรสของฝั่งรีโมทเป็นต้นทาง

dwProtocol กำหนดโพรโตคอลไอพีที่จะต้องฟิลเตอร์ ซึ่งมี ICMP, TCP, UDP หรือทั้งหมด

fLateBound ใช้ในการกำหนดข้อมูลเกี่ยวกับแอดเดรสที่ต้องการถูกปรับปรุงเมื่อมีการ rebound ฟิลเตอร์ (เช่นเมื่อมีการเชื่อมต่อใหม่ของ dial-up adapter) สำหรับการใช้งานของฟิลด์นี้ยังเป็นที่สงสัยอยู่

wSrcPort หมายเลขพอร์ตต้นทาง

wSrcPortHighRange ขอบบนของหมายเลขพอร์ตต้นทาง

wDstPort หมายเลขพอร์ตปลายทาง

wDstPortHighRange ขอบบนของหมายเลขพอร์ตปลายทาง

การกำหนดพอร์ตในการฟิลเตอร์นั้น ถ้ากำหนดหมายเลขพอร์ตเป็นศูนย์จะหมายถึงทุกพอร์ต แต่ถ้ากำหนดหมายเลขพอร์ตไม่เลขที่ไม่ใช่ศูนย์แล้วจะต้องกำหนด PortHighRange ด้วย โดยถ้าต้องการระบุพอร์ตใดพอร์ตหนึ่งก็ให้กำหนดทั้งสองฟิลด์ให้เป็นค่าเดียวกัน แต่ถ้าต้องการกำหนดเป็นช่วงของพอร์ตก็ให้กำหนด PortHighRange ให้เป็นค่าช่วงของหมายเลขพอร์ต กล่าวคือ กำหนดค่าพอร์ตให้เป็นหมายเลขพอร์ตต่ำสุดที่ต้องการฟิลเตอร์ และ PortHighRange เป็นค่าสูงสุดของพอร์ตที่ต้องการฟิลเตอร์ เช่น ถ้าต้องการให้มีการฟิลเตอร์ตั้งแต่พอร์ต 21 ถึงพอร์ต 48 ก็ให้กำหนด Port เป็น 21 และ PortHighRange เป็น 48 เป็นต้น

บทที่ 5

เอเจนต์

ไม่ใช่เรื่องใหม่สำหรับมนุษย์ที่พยายามจะสร้างสิ่งที่จะมาผ่อนภาระในการทำงานของมนุษย์ลง โดยคอมพิวเตอร์ก็เป็นแนวคิดหนึ่งสำหรับความพยายามนี้ แต่สำหรับคอมพิวเตอร์โปรแกรมทั่วไป เราจำเป็นจะต้องให้คำสั่งเป็นขั้นๆสำหรับการจะทำอะไรสักอย่าง ซึ่งหมายถึงว่าผู้ใช้จะต้องรู้เป้าหมายและรู้วิธีการที่จะไปสู่เป้าหมายและเป็นผู้รับผิดชอบในขั้นตอนต่างๆ ส่วนคอมพิวเตอร์โปรแกรมก็จะรับหน้าที่การทำงานในขั้นตอนต่างๆที่ได้รับคำสั่งไปนั่นเอง

แต่สำหรับเอเจนต์ได้ก้าวไปมากกว่านั้นอีกหนึ่งก้าว แนวคิดก็คือเรามอบงานที่ต้องการให้เอเจนต์โดยที่เราไม่จำเป็นต้องรู้ว่าวิธีการนำไปสู่เป้าหมายนั้นเป็นขั้นตอนอย่างไร ซึ่งเอเจนต์จะเป็นผู้รับผิดชอบในส่วนนี้ และเมื่อเอเจนต์ทำงานจนถึงเป้าหมายที่ต้องการแล้วค่อยนำผลลัพธ์กลับมาให้ผู้ใช้ ซึ่งหมายถึงว่าเอเจนต์จะต้องมีหน้าที่ในการรับผิดชอบวิธีการที่จะนำไปสู่เป้าหมายด้วย

จริงๆแล้วเอเจนต์ หรือเอเจนต์ฉลาด (Intelligent Agent) ไม่ใช่เรื่องใหม่ ได้มีการค้นคว้าวิจัยมานานพอสมควรแล้ว แต่อาจจะมีช่วงหนึ่งที่คำว่าเอเจนต์ฉลาดได้ถูกใช้อย่างฟุ่มเฟือย เนื่องจากผู้คนได้คาดหวังกับคำว่า Intelligent ว่าจะต้องสามารถทำสิ่งที่เขาต้องการได้ ในขณะที่ผู้ที่นำเอาคำว่าเอเจนต์ฉลาดไปใช้กับโปรแกรมก็เป็นเพียงแค่ซอฟต์แวร์ธรรมดาแต่ต้องการหวังผลทางด้านการตลาด หรือทั้งการใช้คำเอเจนต์ฉลาดอย่างผิดๆ จึงทำให้ผู้คนรู้สึกเฉยชากับเรื่องนี้ไป

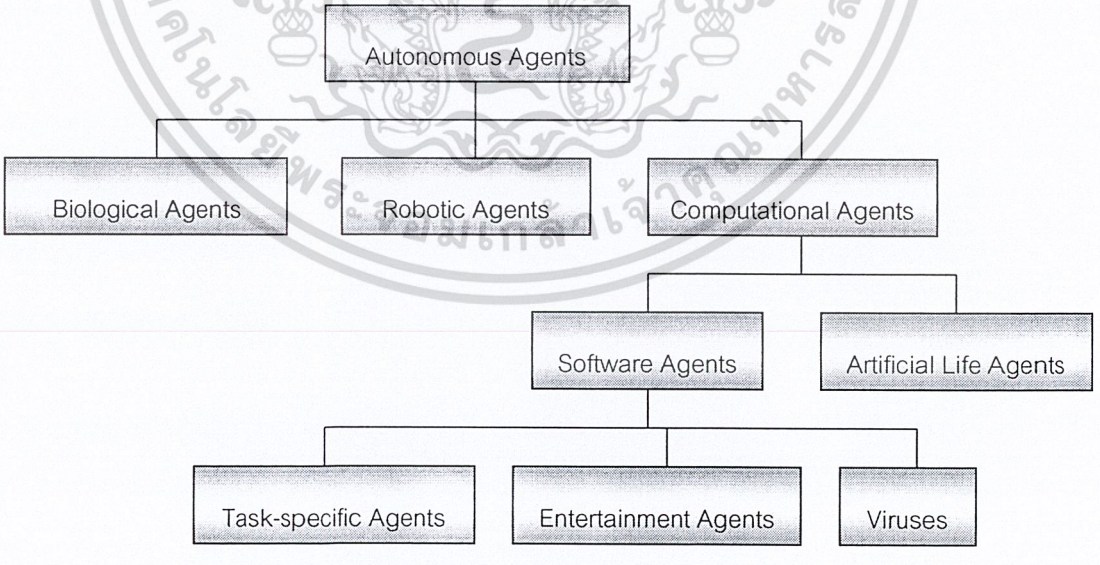
จริงๆแล้วในปัจจุบันก็ได้มีการใช้งานเอเจนต์ในงานต่างๆมากมายโดยที่เราก็อาจจะไม่รู้ว่านั่นก็เป็นเอเจนต์เช่นกัน โดยเอเจนต์มีอยู่ตั้งแต่ระบบเล็กๆอย่าง mail filtering ตามโปรแกรมอีเมล์ต่างๆ จนถึงระบบขนาดใหญ่ที่ซับซ้อน เช่น ระบบจัดการการจราจรทางอากาศ

5.1 เอเจนต์คืออะไร?

ถ้าจะให้กำหนดนิยามที่แน่ชัดและเจาะจงไปสำหรับคำว่า "เอเจนต์" นั้นคงจะเป็นการยาก เนื่องจากในปัจจุบันนักวิจัยหรือสถาบันวิจัยแต่ละที่ก็ยังให้ความหมายของคำว่าเอเจนต์ต่างกันไป ซึ่งอาจจะเนื่องมาจากพื้นฐานทางด้านแนวทางการศึกษาของแต่ละที่ ยกตัวอย่างเช่นสำหรับนักวิจัยที่ศึกษาหรือมาจากทางด้าน AI ก็จะเน้นให้เอเจนต์ต้องมีความสามารถในการเรียนรู้ได้ ในขณะที่ถ้าเป็นฝั่งที่ไม่ได้มาจากด้าน AI ก็อาจจะไม่ได้ให้ความสำคัญในจุดนั้น ซึ่งอย่างไรก็ตามก็ขึ้นอยู่กับเป้าหมายและแนวทางว่าจะต้องการพัฒนาเอเจนต์เพื่องานประเภทใด

แต่สำหรับพื้นฐานที่เอเจนต์จะต้องมีเป็นจุดร่วมนั้นก็คือการทำงานแบบ "autonomous" เพื่อบรรลุเป้าหมาย ถึงแม้ว่าคำว่า autonomous จะเป็นคำที่ยากในการที่จะอธิบายความหมายอย่างถูกต้อง แต่อาจจะพูดได้ว่าระบบใดๆจะเป็น autonomous ก็ต่อเมื่อระบบนั้นๆสามารถที่จะกระทำการโดยไม่จำเป็นต้องมีการแทรกแซงหรือควบคุมจากผู้ใช้หรือสิ่งอื่นจากภายนอก และมีความสามารถในการควบคุมการกระทำและสถานะต่างๆของตัวเอง

แน่นอนว่าเอเจนต์ก็เหมือนกับเรื่องอื่นที่สามารถแตกแขนงแยกย่อยลงไปได้อีกหลายขั้น ซึ่งสำหรับเอเจนต์แล้วก็สามารถจะเป็นได้ทั้งซอฟต์แวร์หรือฮาร์ดแวร์ เช่น robotic agents แต่สำหรับในที่นี้คำว่าเอเจนต์จะหมายถึงซอฟต์แวร์เอเจนต์เท่านั้น



รูปที่ 5-1 เอเจนต์แบบต่างๆ

สำหรับคุณสมบัติอื่นๆที่ เอเจนต์มีได้มีดังนี้:

- **Interactive** – สื่อสารกับสิ่งแวดล้อมหรือเอเจนต์อื่นๆ
- **Adaptive** – สามารถที่จะโต้ตอบกับสิ่งแวดล้อมหรือเอเจนต์อื่นๆได้ในระดับหนึ่ง ถ้าเป็นรูปแบบที่สูงกว่านั้นก็คือเอเจนต์จะสามารถปรับปรุงแก้ไขพฤติกรรมของตัวเองตามประสพการณ์ได้
- **Sociable** – ลักษณะของเอเจนต์ที่มีการสังคม การสื่อสารอย่างเป็นมิตร
- **Mobile** – สามารถที่จะเคลื่อนย้ายตัวเองจากสิ่งแวดล้อมหนึ่งไปยังอีกที่หนึ่งได้
- **Proxy** – แสดงตัวเป็นตัวแทนของบางสิ่งได้ เพื่อประโยชน์บางอย่าง
- **Proactive** – มีเป้าหมายในตัวเอง ไม่ใช่เพียงแค่ปฏิบัติเป็นการตอบรับกับ environment
- **Intelligent** – สถานะในตัวเองประกอบด้วยความรู้ (beliefs, goals, plans, assumptions) และโต้ตอบกับเอเจนต์อื่นโดยใช้ภาษาทางสัญลักษณ์
- **Rational** – สามารถที่จะเลือกสิ่งที่จะทำโดยขึ้นอยู่กับ internal goals และ knowledge ซึ่ง action ใดที่จะทำให้ตัวเองเข้าไปสู่ goal ได้
- **Unpredictable** – การกระทำไม่สามารถที่จะทำนายได้แน่นอน แม้ว่าจะรู้ condition ต่างๆ (nondeterministic)
- **Temporally continuous** – เป็น process ที่มีการทำงานอย่างต่อเนื่อง
- **Character** – มีความเป็นตัวตนและสถานะทางอารมณ์
- **Transparent และ accountable** – สามารถที่จะ transparent ได้เมื่อจำเป็น แต่ก็ต้องสามารถแสดง log ของ activity ที่ทำไปเมื่อต้องการได้
- **Coordinative** – สามารถที่จะทำกิจกรรมในสิ่งแวดล้อมหนึ่งๆร่วมกับ เอเจนต์อื่นๆได้ กิจกรรมมักจะเกิดจากการร่วมงานผ่าน plan, workflow หรือ process management mechanism อื่นๆ
- **Cooperative** – สามารถที่จะร่วมมือกับเอเจนต์อื่นๆเพื่อบรรลุจุดประสงค์ร่วมกัน
- **Competitive** – สามารถที่จะร่วมมือกับเอเจนต์อื่นๆได้ยกเว้นเมื่อความสำเร็จของเอเจนต์ตัวหนึ่งนำมาซึ่งความล้มเหลวของเอเจนต์ตัวอื่น (ตรงกันข้ามกับ cooperative)
- **Rugged** – สามารถที่จะจัดการกับ errors หรือข้อมูลที่ไม่ถูกต้องได้อย่างดี
- **Trustworthy** - สามารถเชื่อถือได้

5.2 เอเจนต์และออปเจกต์

หลายๆคนอาจจะมองว่าเอเจนต์ก็เหมือนกับออปเจกต์นั่นเอง บางคนบอกว่าเอเจนต์เป็นออปเจกต์ชนิดหนึ่ง ในขณะที่บางคนบอกว่าออปเจกต์เป็นเอเจนต์ชนิดหนึ่ง จึงอยากจะเปรียบเทียบระหว่างเอเจนต์และออปเจกต์ว่ามีความเหมือนและความแตกต่างอย่างไร

เอเจนต์เหมือนกับออปเจกต์ในหลายๆอย่าง เราสามารถมองว่าเอเจนต์เหมือนกับออปเจกต์ตรงที่ว่าเอเจนต์มี identity (เราสามารถแบ่งแยกระหว่างเอเจนต์แต่ละตัวได้) และเอเจนต์มีสถานะและ พฤติกรรมเป็นของตัวเอง และเอเจนต์ก็มีอินเตอร์เฟส (Interface) ซึ่งใช้ในการสื่อสารกับสิ่งอื่น

แต่เอเจนต์ก็มีความแตกต่างจากออปเจกต์ในหลายอย่างเช่นกัน เรามองว่าเอเจนต์จะมีการทำงานที่ "ฉลาด" กว่าออปเจกต์ซึ่งมักจะประกอบด้วยระบบการตัดสินใจภายในเอเจนต์ยังมีลักษณะของ autonomous และ reactive ซึ่งหมายความว่าเอเจนต์สามารถที่จะตัดสินใจได้ว่าจะตอบสนองกับ message ที่เข้ามาหรือไม่อย่างไร ในขณะที่ออปเจกต์จะทำในสิ่งที่ถูกขอให้ทำเสมอ (มีลักษณะเป็น passive) นี่เป็นข้อตกต่างหลักๆระหว่างเอเจนต์และออปเจกต์

5.3 ประเภทของเอเจนต์

เอเจนต์สามารถแบ่งประเภทได้จากหลายๆอย่าง เราอาจจะแบ่งแยกเอเจนต์จากเรื่องของ mobility ว่าเอเจนต์นั้นสามารถที่จะเคลื่อนที่ไปยังที่ต่างๆในเครือข่ายได้หรือไม่ หรืออาจจะแบ่งว่าเอเจนต์นั้นมีระบบการคิดภายในหรือว่าเพียงแค่ตอบสนองจากสิ่งภายนอกเพียงเท่านั้น หรือเราอาจจะแบ่งตามคุณสมบัติพื้นฐานของเอเจนต์เช่น ความเป็น autonomous, ความสามารถในการเรียนรู้ หรือความสามารถในการร่วมมือกันทำงาน แต่จะอย่างไรก็ตาม เราก็ยังสามารถแบ่งเอเจนต์ออกเป็น 7 ประเภทใหญ่ๆได้ดังนี้

- **Collaborative Agents** เป็นเอเจนต์ที่มีการทำงานร่วมกันระหว่างเอเจนต์ในการแก้ปัญหาซึ่งใหญ่เกินไปสำหรับการใช้เอเจนต์ตัวเดียวซึ่งอาจจะมาจากข้อจำกัดทางด้าน resource หรือเพื่อลดความเสี่ยง เพื่อเพิ่มความเร็วในการทำงาน ความน่าเชื่อถือ และลดความซับซ้อนของงาน เป็นต้น
- **Interface Agents** หมายถึงเอเจนต์ซึ่งมีการติดต่อสื่อสารกับคนในการที่จะทำงานอย่างใดอย่างหนึ่ง ซึ่งแนวคิดมาจากการที่จะให้เอเจนต์รับงานที่ซับซ้อนไปจากคนโดยคนไม่จำเป็นที่จะต้องให้คำสั่งโดยตรงแก่ระบบในการทำงานอย่างใดอย่างหนึ่ง ซึ่งเอเจนต์จะต้องมีความสามารถในการเรียนรู้จากผู้ใช้ถึงความต้องการและวิธีที่จะทำความต้องการนั้นให้สำเร็จได้ ตัวอย่างของ Interface Agents ก็เช่น ระบบ personalize ต่างๆ เช่น ใน amazon.com
- **Mobile Agents** หมายถึงเอเจนต์ที่มีความสามารถในการเคลื่อนย้ายตัวไปยังที่ต่างๆได้ ซึ่งใช้ประโยชน์ในงานทางด้าน network หลายๆอย่าง
- **Information/Internet Agents** คือเอเจนต์ที่ทำหน้าที่จัดการในเรื่องข้อมูลต่างๆที่มีขนาดใหญ่ อย่างเช่น search engine ซึ่งจะใช้ Information หรือ Internetเอเจนต์ในการค้นหาและรวบรวมข้อมูล
- **Reactive Agents** คือเอเจนต์ที่ไม่มีระบบในการตัดสินใจหรือการประมวลผลภายในตัวเอง แต่จะกระทำโดยการตอบสนองกับภายนอกเพียงเท่านั้น
- **Hybrid Agents** เป็นเอเจนต์ที่รวมเอาคุณสมบัติของเอเจนต์ประเภทต่างๆเข้ามาด้วยกัน เนื่องจากสำหรับในงานบางอย่างอาจจะเป็นการเหมาะสมกว่าที่เอเจนต์จะมีคุณสมบัติของเอเจนต์มากกว่าหนึ่งประเภท
- **Heterogeneous Agents** หมายถึงเอเจนต์ที่ทำงานในระบบซึ่งมีการร่วมมือระหว่างเอเจนต์หลายๆตัว แต่ต่างจาก cooperativeเอเจนต์ตรงที่ว่าเป็นระบบที่มีเอเจนต์ ที่มีหน้าที่ต่างๆกันมาทำงานโดยช่วยเหลือกันในแต่ละส่วนเพื่อให้บรรลุเป้าหมายร่วมกัน

บทที่ 6
การดำเนินงาน

ในโครงงานนี้ จะประกอบไปด้วยโปรแกรม

1. เพอร์ซันนอลไฟร์วอล
2. ส่วนการติดต่อระหว่างไคลเอนต์และเซิร์ฟเวอร์
3. ส่วนควบคุมการทำงาน

6.1 เพอร์ซันนอลไฟร์วอล

6.1.1 หลักการ

เนื่องจากไฟร์วอลที่ทำงานอยู่บนระดับเกทเวย์มีราคาสูง ทำให้สำนักงานขนาดย่อมที่มีจำนวนเครื่องคอมพิวเตอร์ไม่มาก ไม่คุ้มค่าแก่การลงทุน จึงนำเพอร์ซันนอลไฟร์วอลมาใช้งาน ซึ่งข้อเสียของการนำเพอร์ซันนอลไฟร์วอลมาใช้งานนี้ ทำให้การควบคุม การกระจายกฎ เป็นไปอย่างไม่เหมาะสม กล่าวคือ ผู้ใช้เครื่องสามารถที่จะกำหนดกฎ สิทธิในการใช้งานได้เอง อีกทั้งเมื่อมีการเพิ่มคอมพิวเตอร์ใหม่ ก็จะต้องมีการติดตั้งเพอร์ซันนอลไฟร์วอล และกำหนดกฎเกณฑ์ใหม่อีกครั้ง ดังนั้น เราจึงพัฒนาสถาปัตยกรรมของไฟร์วอล โดยการกระจายกฎจากส่วนกลางผ่านเอเจนต์เพื่อความสะดวกและเพิ่มประสิทธิภาพของการใช้งานมากยิ่งขึ้น

6.1.2 ขอบเขตความสามารถ

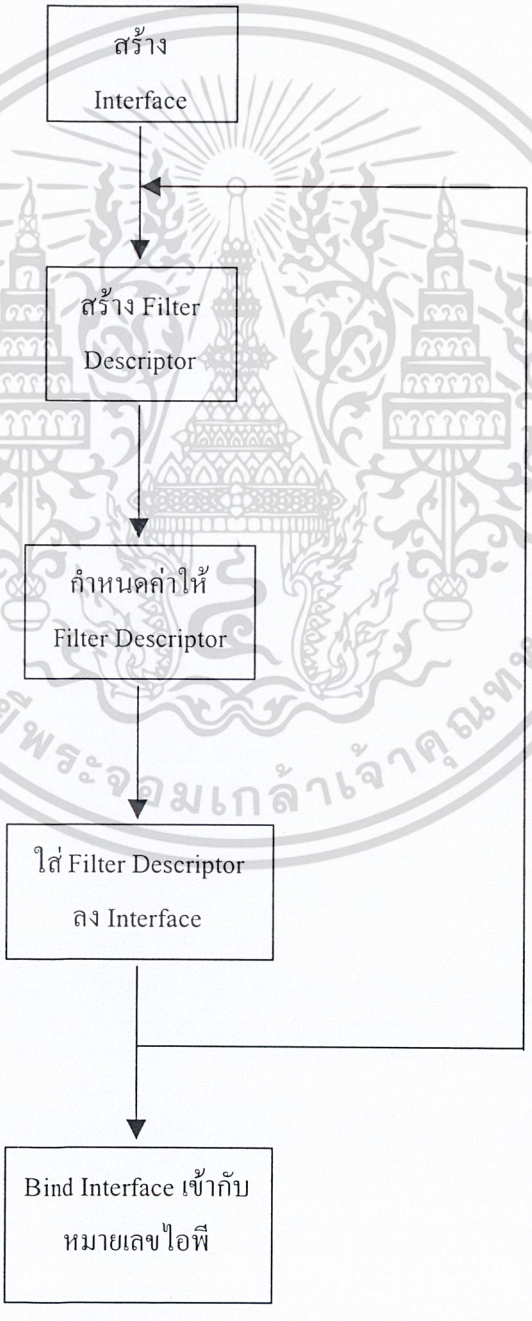
เครื่องทุกๆ เครื่องในเครือข่ายจะต้องมีเอเจนต์ฝังตัวทำงานอยู่ ซึ่งเอเจนต์ทั้งหลายนั้นจะทำหน้าที่คล้ายกับเพอร์ซันนอลไฟร์วอล คือจะทำหน้าที่ตรวจสอบแพ็คเก็ตที่เข้า - ออกเครื่องนั้นๆ แต่จะต่างกับเพอร์ซันนอลไฟร์วอลธรรมดาตรงที่สามารถควบคุม ดูแล และบริหารเอเจนต์ทั้งหมดจากส่วนกลาง โดยผู้ใช้จะไม่รับรู้ถึงการทำงานของเอเจนต์ อีกทั้งผู้ดูแลระบบ สามารถกำหนดกฎให้กับผู้ใช้แต่ละคนตามความเหมาะสมว่าบุคคลนั้น มีสิทธิ์ในการใช้บริการมากน้อยเพียงใด จากสถาปัตยกรรมการทำงานแบบนี้ทำให้สามารถที่จะรวมเอาข้อดีของไฟร์วอลทั้งสองประเภทเอาไว้เข้าด้วยกันได้

6.1.3 หน้าที่การทำงาน

การตรวจสอบและกรองแพ็คเก็ต

การตรวจสอบและกรองแพ็คเก็ตสามารถทำได้ผ่าน Windows 2000 Packet Filtering API ใน วินโดวส์ 2000 นั้นได้มี API (Application Programming Interface) ที่ใช้สำหรับทำแพ็คเก็ตฟิลเตอร์โดย เฉพาะมาให้ โดยไลบรารีของ API นี้จะอยู่ใน Microsoft Platform SDK ในไฟล์ IphIpApi.lib โดยจะต้อง include เฮดเดอร์ไฟล์ ชื่อ Fltdefs.h เข้าไปด้วย ซึ่งสำหรับรายละเอียดได้มีการกล่าวไปแล้ว

สำหรับขั้นตอนการทำงานของ Packet Filtering API เป็นดังนี้



รูปที่ 6-1 ขั้นตอนการทำงานของ Packet Filter API

รูปแบบกฎของไฟร์วอล

สำหรับโครงสร้างกฎของไฟร์วอลนั้นได้ยึดตามรูปแบบของ Windows 2000 Packet Filter API ซึ่งเป็นดังต่อไปนี้

Src Address	Src Mask	Src Port	Dst Address	Dst Mask	Dst Port	Protocol

6.2 ส่วนการติดต่อระหว่างไคลเอนต์และเซิร์ฟเวอร์

สำหรับการติดต่อกับระหว่างไคลเอนต์และเซิร์ฟเวอร์นั้น จะกระทำผ่านโพรโตคอล TCP ซึ่งจะมีส่วนในการรับผิดชอบความถูกต้องของข้อมูลอยู่แล้ว เมื่อมีการส่งข้อมูลการเชื่อมต่อแบบ TCP ก็จะถูกสร้างขึ้น และหลังจากส่งข้อมูลเสร็จการเชื่อมต่อก็จะถูกทำลายทันที โดยข้อมูลที่มีการส่งระหว่างกันนั้นคือคำสั่งต่างๆนั่นเอง โดยรูปแบบของคำสั่งเป็นดังนี้

COMMAND	DATA (Optional)

ส่วนของ COMMAND เป็นส่วนที่ระบุคำสั่ง และส่วนของ DATA นั้นอาจจะมีหรือไม่มีก็ได้ โดยถ้ามีก็จะเป็นเนื้อข้อมูลที่เกี่ยวข้องกับคำสั่งนั้นๆ เช่นกฎของการฟิลเตอร์

6.2.1 ลักษณะทั่วไปของการติดต่อ

ทั้งฝั่งไคลเอนต์และเซิร์ฟเวอร์จะมีการสร้างซ็อคเก็ต ขึ้นมาฝั่งละ 2 ซ็อคเก็ต โดยซ็อคเก็ตหนึ่งทำหน้าที่รอรับการเชื่อมต่อ ส่วนอีกซ็อคเก็ตหนึ่งจะใช้เมื่อต้องการจะเชื่อมต่อไปยังเครื่องอื่นๆ กล่าวอีกนัยหนึ่งคือซ็อคเก็ตหนึ่งไว้สำหรับรับข้อมูล อีกซ็อคเก็ตหนึ่งใช้ส่งข้อมูล โดยผู้ส่งข้อมูลจะเป็นผู้สร้างการเชื่อมต่อเท่านั้น

6.2.2 ส่วนประกอบของการติดต่อ

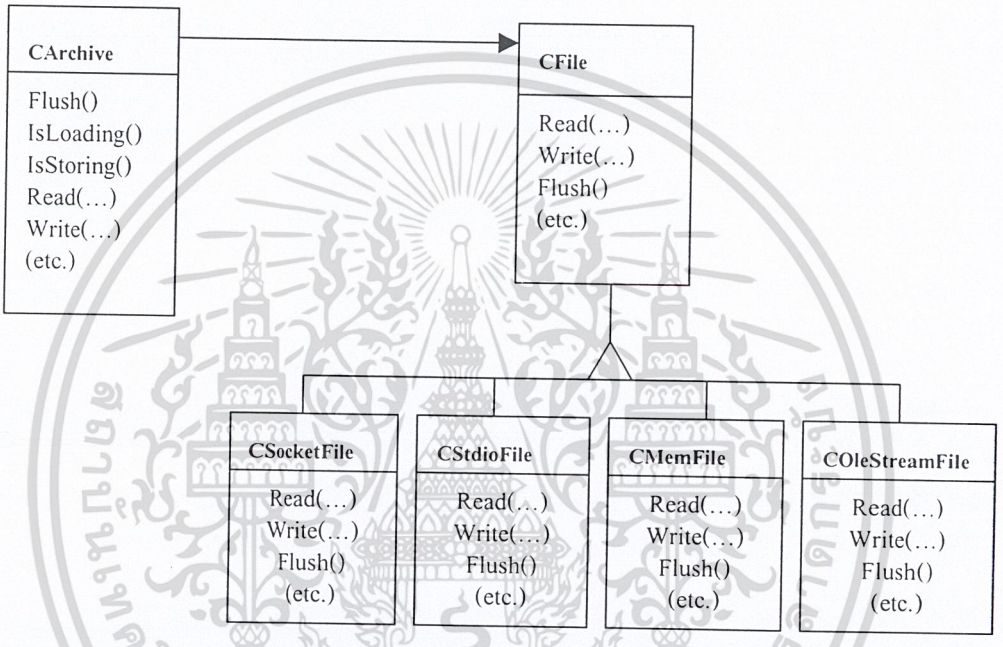
สำหรับส่วนของการส่งข้อมูลนั้น จะกระทำผ่านคลาส CSocket ซึ่งเป็นคลาส MFC ที่ครอบการทำงานของ Winsock ไว้อีกทีหนึ่ง ซึ่งข้อดีของ CSocket คือ เราจะมองการรับ/ส่งข้อมูลเป็นการทำ Serialization เหมือนกับการบันทึกข้อมูลลงดิสก์หรืออื่นๆ ซึ่งทำให้ง่ายและเกิดความยืนหยุ่นในการออกแบบขึ้น

Serialization

Serialization เป็นวิธีการในการจัดเก็บ/ดึงออปเจกต์จากแหล่งข้อมูลต่างๆเช่น แฟ้มข้อมูล หน่วยความจำ รวมทั้งซ็อคเก็ตให้เป็นไปในรูปแบบวิธีการเดียวกัน แนวคิดของ Serialization คือออปเจกต์ควรที่จะสามารถบันทึกสถานะปัจจุบัน ซึ่งมักจะหมายถึงค่าของสมาชิกต่างๆไปสู่ตัวกลางต่างๆ เช่น ไฟล์ หน่วย

ความจำ ซ็อคเก็ต เป็นต้น ได้ ซึ่งในเวลาถัดไป ออปเจกต์ก็สามารถที่จะถูกสร้างขึ้นโดยการอ่านค่ากลับมาจากตัวกลางต่างๆเหล่านั้นได้อีกทีหนึ่ง ซึ่งออปเจกต์แต่ละตัวจะรับผิดชอบการบันทึกและอ่านสถานะและค่าต่างๆของตัวเอง

สำหรับ MFC นั้น ในการทำ Serialization จะใช้คลาส CArhive ซึ่งมีหน้าที่เป็นตัวกลางระหว่างออปเจกต์ที่ต้องการ Serialize กับออปเจกต์ของสื่อที่ต้องการจะ Serialize ลงไป โดยรูปแบบโครงสร้างของการ Serialization ใน MFC นั้นเป็นไปตามรูปแบบ Design Patterns แบบ Bridge Pattern



รูปที่ 6-2 รูปแบบของ MFC Serialization

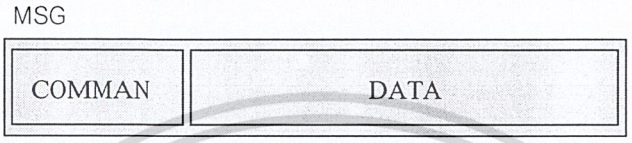
ออปเจกต์ที่ต้องการจะทำ Serialize นั้นจะติดต่อกับออปเจกต์ของคลาส CArchive ซึ่งจะทำหน้าที่ติดต่อกับออปเจกต์จากคลาสที่สืบทอดมาจาก CFile อีกทีหนึ่งหมายความว่าออปเจกต์ที่ต้องการจะ Serialize นั้นไม่จำเป็นจะต้องคำนึงว่าสื่อที่จะทำการ Serialize ลงไปนั้นจะเป็นซ็อคเก็ตหรือว่าแฟ้มข้อมูล โดยทุกอย่างจะมีการทำงานที่เหมือนกันหมดเนื่องจากกระทำผ่าน CArchive นั่นเอง

สำหรับการที่จะทำให้คลาสใดนั้นสามารถ Serialize ได้จะต้องทำขั้นตอนดังต่อไปนี้

1. สืบทอดคลาสจาก CObject
2. โอเวอร์ไรด์ฟังก์ชัน Serialize()
3. ประกาศมาโครDECLARE_SERIAL
4. สร้างคอนสตรัคเตอร์ที่ไม่รับอาร์กูเมนต์ใดๆ
5. ประกาศมาโคร IMPLEMENT_SERIAL ในไฟล์

การใช้วิธีการนี้สามารถทำให้เราออกแบบโปรแกรมในระดับที่สูงขึ้นมาได้ เนื่องจากข้อมูลที่รับ/ส่งโดยการ Serialize นั้นเป็นออปเจกต์จะถูกแปลงเป็นสตรีมผ่านช่องทางการส่งไปยังสู่อีกฝั่งหนึ่ง ซึ่งในกระบวนการ Serialize และ De-Serialize นั้นจะรับผิดชอบในรายละเอียดปลีกย่อยต่างๆเองทั้งหมด เช่น การสร้างออปเจกต์ที่ปลายทางขึ้นมา

จากวิธีดังกล่าวทำให้สามารถออกแบบโพรโตคอลให้อยู่ในรูปแบบของออปเจกต์ได้เป็นดังนี้



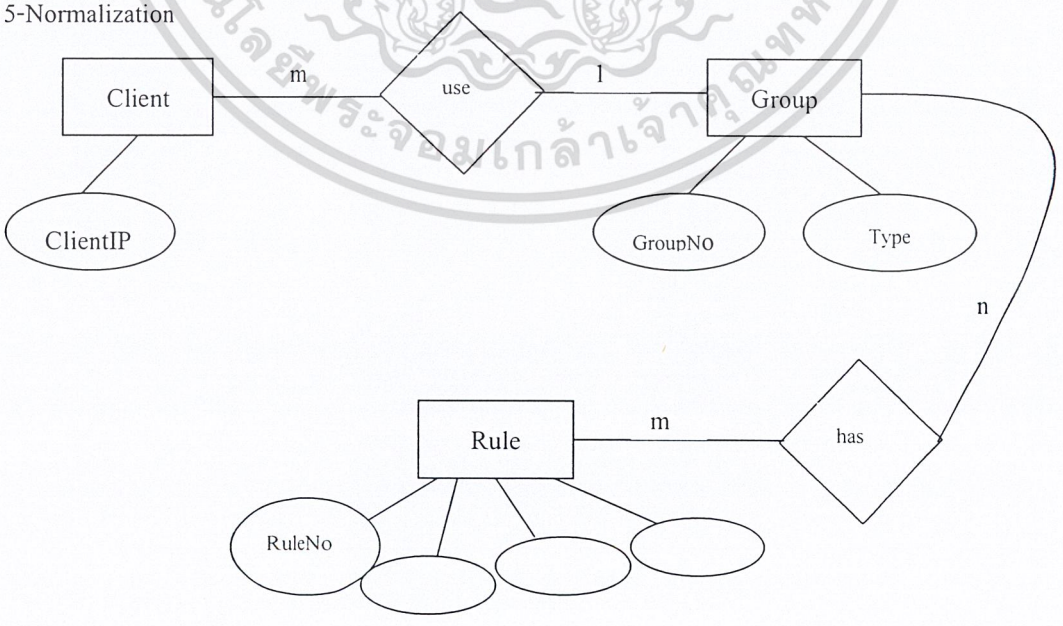
ทุกครั้งที่ส่งข้อมูลจะส่งเป็นออปเจกต์ของ Msg ไป ซึ่งในออปเจกต์ Msg นั้นจะประกอบไปด้วยส่วนของ Command ซึ่งเป็นส่วนที่ระบุคำสั่ง และส่วน Data ซึ่งเป็นส่วนที่ใส่ข้อมูลถ้ามี เช่นกฎสำหรับไคลเอนต์ และในคำสั่งที่ไม่ต้องมีข้อมูลส่วนนี้ก็สามารถที่จะว่างเปล่าได้

6.3 ส่วนควบคุมการทำงาน

6.3.1 ฐานข้อมูล

6.3.1.1 การออกแบบฐานข้อมูล

เนื่องจากตารางที่ใช้ในการเก็บข้อมูล จะต้องมีข้อมูลที่สัมพันธ์กัน เพื่อให้ข้อมูลต่างๆ อยู่รวมกันเป็นกลุ่ม และจะต้องมีความถูกต้องตาม Integrity Rule เพื่อลดปัญหาความซ้ำซ้อน และปัญหาต่างๆ ที่เกิดขึ้น เมื่อมีการเพิ่ม , ลบ หรือแก้ไขข้อมูลในตาราง จึงได้นำเอา ER Diagram มาใช้ในการออกแบบ และทำ 5-Normalization



รูปที่ 6-3 ER Diagram ของฐานข้อมูลกฎ

กลุ่มของกฎ 1 กลุ่ม สามารถมีได้หลายกฎเกณฑ์

กฎเกณฑ์ 1 กฎ สามารถมีอยู่ได้ในหลายกลุ่ม

เป็นความสัมพันธ์แบบ m : n

m : n สร้างตารางใหม่ของแต่ละ Relationship

Rule Table ใช้เก็บฟิลด์ต่างๆของกฎเกณฑ์ ที่นำมาใช้การในฟิลเตอร์

RuleNo	SrcAddr	SrcMask	SrcPort	PortRange	DstAddr	DstMask	DstPort	PortRange	Ptcol

SetType ใช้เก็บประเภทของกลุ่มว่าเป็นกลุ่มที่เก็บกฎเกณฑ์ที่เป็นมาตรฐาน

GroupNo	Type

RuleSet Table ใช้เก็บกฎเกณฑ์เป็นกลุ่ม เพื่อสะดวกต่อผู้ดูแลในการแจกจ่ายกฎให้กับไคลเอนต์ นำไปใช้

GroupNo	RuleNo

ไคลเอนต์หนึ่ง เรียกใช้กฎเกณฑ์ได้เพียงกลุ่มเดียว

กลุ่มของกฎ 1 กุลุ่ม ถูกเรียกใช้ได้จากหลายไคลเอนต์

เป็นความสัมพันธ์แบบ 1 : m

1 : m นำ Primary Key ของ many มาเกาะ

Client Table ใช้เก็บข้อมูลของผู้ใช้และกลุ่มของกฎเกณฑ์ที่ใช้

ClientIP	GroupNo

6.3.1.2 การติดต่อกับฐานข้อมูล

การติดต่อกับฐานข้อมูลโดยผ่านไดรเวอร์ ODBC โดยจะต้องสร้าง Data Source ผ่าน ODBC Administrator

ODBC สนับสนุนคลาสต่างๆ ดังนี้

คลาส CDatabase เป็นคลาสที่แสดงการติดต่อกับแหล่งข้อมูลต่างๆ เช่น SQL , dBase , Microsoft Access ในการติดต่อกับ Database ทุกครั้ง เราจะต้องสร้างอ๊อบเจ็กต์ของ CDatabase ด้วยการเรียกฟังก์ชัน Open ขึ้นมาก่อน เพื่อกำหนดว่าต้องการเรียกใช้ Data Source ใด การใช้ ExecuteSQL เพื่อรัน SQL สเตทเม็นท์ และ Close เพื่อปิดการติดต่อข้อมูล

คลาส CRecordset เป็นคลาสที่แสดงเซตของเรคคอร์ด โดยคลาสนี้ถูกสร้างจาก MFC AppWizard อ๊อบเจ็กของคลาสนี้ แบ่งออกได้เป็น 2 ชนิด คือ ไดนาเซต (Dynaset) และ Snapshot ซึ่งไดนาเซ็ตเป็นแบบไดนามิก คือ ถ้าเราอัปเดทข้อมูลในเรคคอร์ดแล้ว ก็จะเห็นข้อมูลเปลี่ยนแปลงทันที แต่

Snapshot จะเป็นแบบสเตติก เมื่อทำการอัปเดทข้อมูลจะไม่มีผลต่อข้อมูลจนกว่าจะเรียกคำสั่ง Requery อีกครั้ง

การเรียกใช้ Store Procedure เป็นโค้ดที่ถูกเขียนขึ้นมาจาก SQL สเตทเม็นท์และถูกเก็บไว้ในฐานข้อมูลของเซิร์ฟเวอร์ ซึ่งโปรแกรมอื่นๆ สามารถเรียกใช้ได้ ข้อดีของการใช้ Store Procedure คือ Store Procedure ถูกคอมไพล์บนเซิร์ฟเวอร์ในขณะที่สร้าง ดังนั้น จึงทำงานได้เร็วกว่าคำสั่ง SQL สเตทเม็นต์ สามารถเรียก Store Procedure อื่นๆ ได้จาก Store Procedure ของเรา ทำให้ไม่ต้องใช้สเตทเม็นต์ที่ซับซ้อนมาก นอกจากนี้ Store Procedure ยังยอมรับตัวแปรจากแอปพลิเคชั่นที่เรียกใช้ และสามารถให้ผลลัพธ์เป็นเซตเดี่ยวหรือหลายๆเซตด้วย

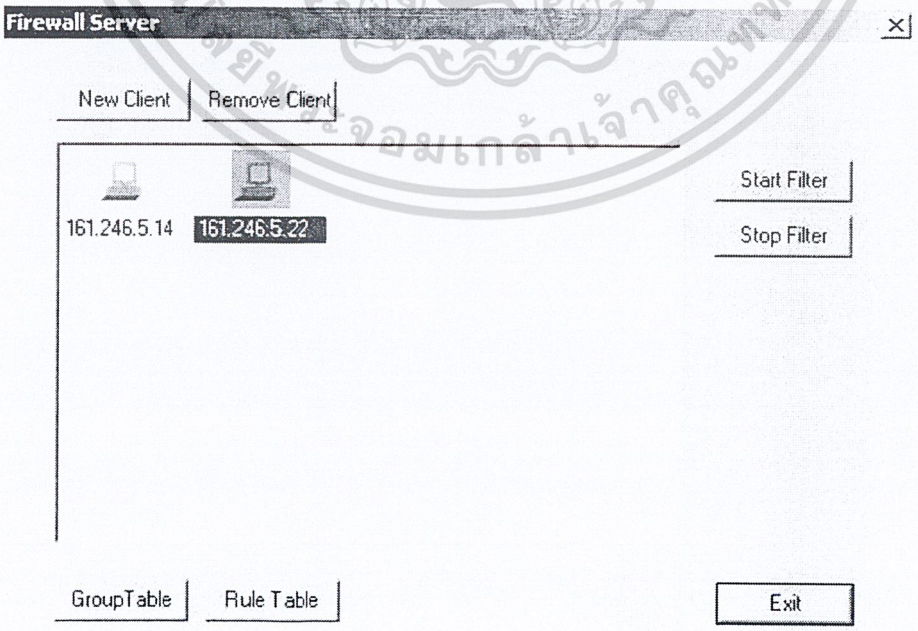
6.3.1.3 ส่วนควบคุมการทำงาน

มีฟังก์ชันต่างๆ ดังนี้

1. การควบคุมการส่งกฎเกณฑ์ให้ไคลเอนต์
2. การเข้าถึงฐานระบบฐานข้อมูล
3. การกำหนดกฎเกณฑ์
4. การกำหนดกลุ่มของกฎเกณฑ์
5. การเพิ่มไคลเอนต์
6. การค้นหา

6.3.1.4 การควบคุมการส่งกฎเกณฑ์ให้ไคลเอนต์

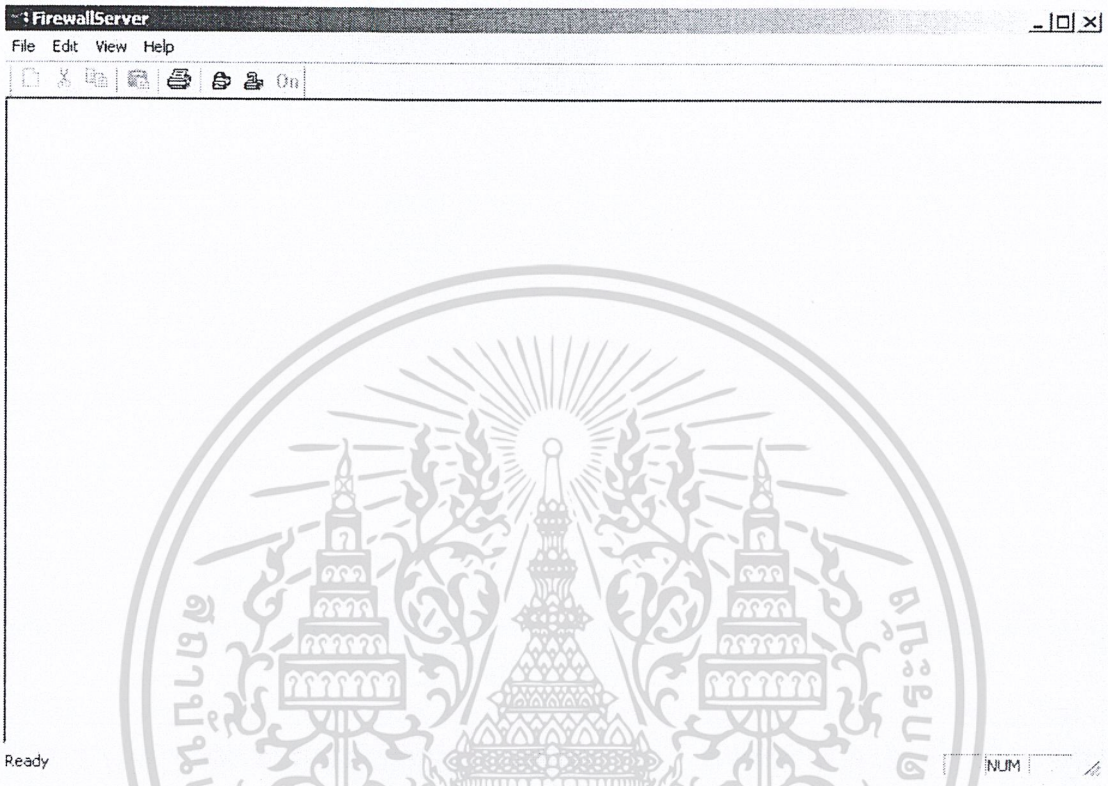
เมื่อมีไคลเอนต์ Login เข้าสู่ระบบ เซิร์ฟเวอร์จะส่งกฎเกณฑ์ของไคลเอนต์นั้นผ่านทางไฟร์วอลเอเจนต์ เพื่อใช้ในการกรองแพ็คเก็ต กดปุ่ม Start Filter เพื่อเริ่มการทำงาน และ Stop filter เพื่อยกเลิกการทำงานของไฟร์วอลเอเจนต์



รูปที่ 6-4 แสดงการส่งกฎให้ไคลเอนต์

กดปุ่มแม่กุญแจเพื่อเริ่มติดต่อและส่งข้อมูล

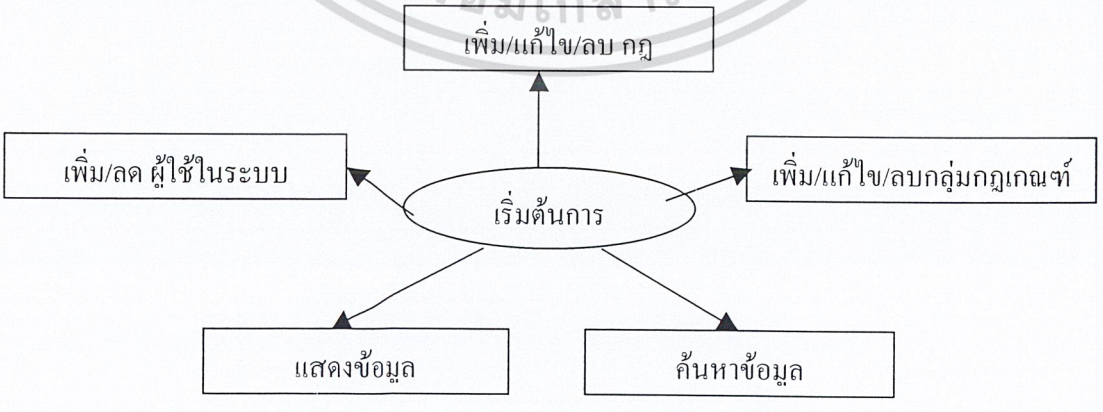
เมื่อฝั่งเอเจนต์ได้รับ ก็จะทำการ Bind Interface เพื่อนำไปใช้งาน



รูปที่ 6-5 แสดงการเริ่มติดต่อของฝั่งเซิร์ฟเวอร์

6.3.1.5 การเข้าถึงระบบฐานข้อมูล

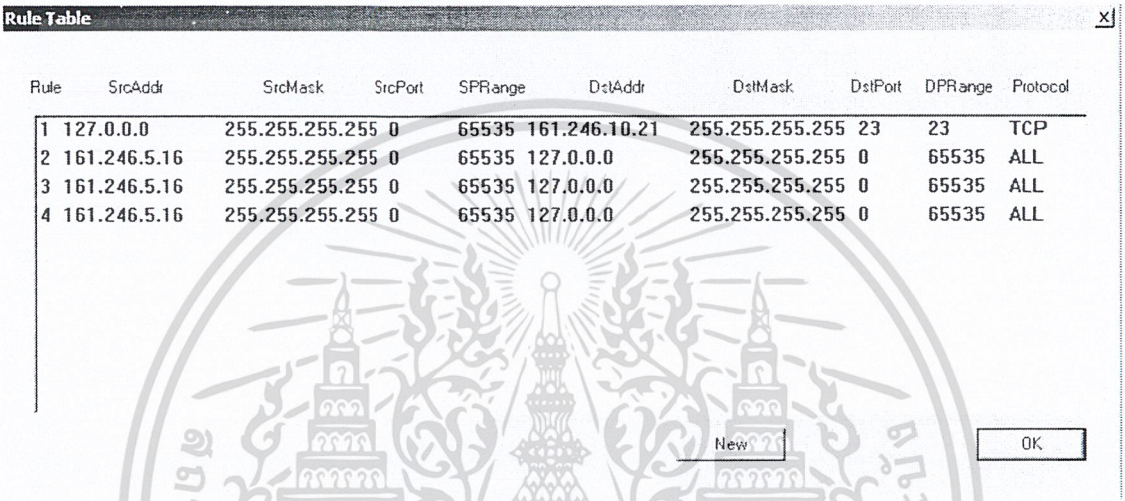
ผู้ดูแลระบบ สามารถเปลี่ยนแปลง แก้ไขข้อมูลต่างๆ จากงฐานข้อมูลที่มีอยู่ในเซิร์ฟเวอร์ ได้ดังนี้



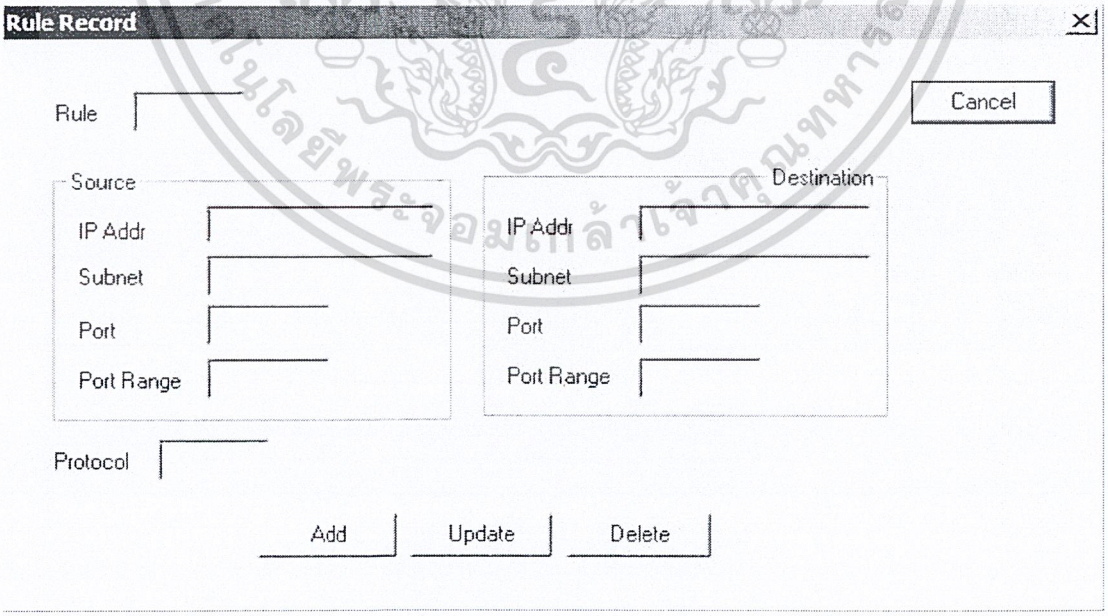
รูปที่ 6-6 แสดงแผนภาพการเข้าถึงฐานข้อมูล

6.3.1.6 การกำหนดกฎเกณฑ์

ผู้ดูแลระบบสามารถเลือกดูกฎทั้งหมดที่มีอยู่ได้ หรือเพิ่มกฎใหม่โดยการกดปุ่ม New เมื่อกรอกข้อมูลเสร็จ กดปุ่ม Add เพื่อส่งข้อมูลให้กับฐานข้อมูล สามารถทำการแก้ไขข้อมูลด้วยการกดปุ่ม Update และลบข้อมูลด้วยปุ่ม Delete



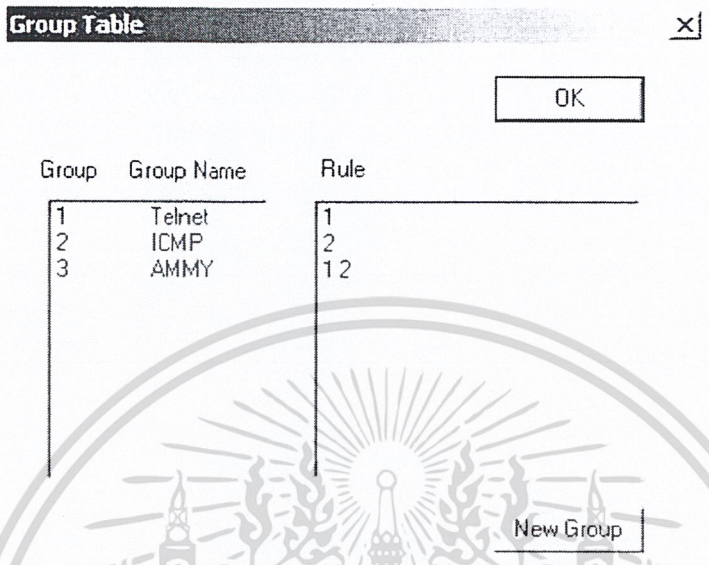
รูปที่ 6-7 แสดงกฎทั้งหมดที่มีในฐานข้อมูล



รูปที่ 6-8 แสดงตารางกฎเกณฑ์

6.3.1.7 การกำหนดกลุ่มของกฎเกณฑ์

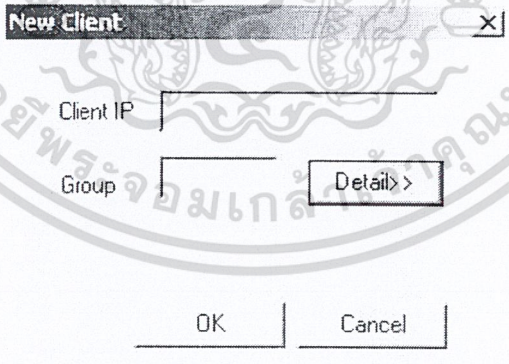
ผู้ดูแลระบบสามารถกำหนดกลุ่มของกฎ รวมถึงประเภทของกลุ่มได้ เพื่อสามารถนำมาใช้ได้สำหรับไคลเอนต์ทั่วไป โดนการกดปุ่ม New Group



รูปที่ 6-9 แสดงตารางกลุ่มของกฎเกณฑ์

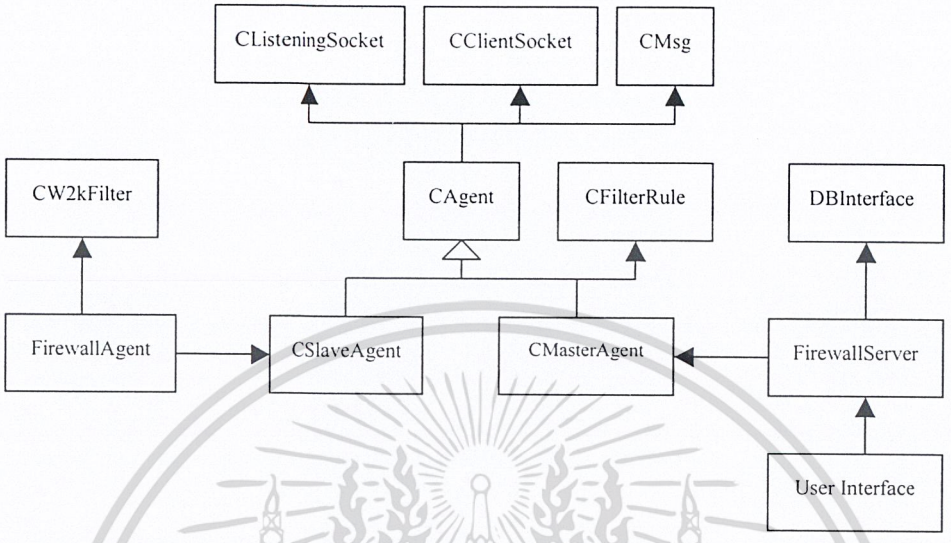
6.3.1.8 การเพิ่มไคลเอนต์

เมื่อมีเครื่องไคลเอนต์เพิ่มขึ้นในระบบ สามารถเพิ่มและกำหนดกลุ่มของกฎได้ โดยการเพิ่มกฎนี้จะสามารถเพิ่มได้ ก็ต่อเมื่อกลุ่มของกฎดังกล่าว ได้ถูกตั้งไว้เรียบร้อยแล้ว



รูปที่ 6-10 แสดงตารางไคลเอนต์

6.4 โครงสร้างของโปรแกรม



รูปที่ 6-11 ผังโครงสร้างโปรแกรม

CListeningSocket รอรับการเชื่อมต่อจากเครื่องอื่นๆ

CClientSocket ใช้ในการสร้างการเชื่อมต่อไปยังเครื่องอื่นๆ

CMsg เป็นรูปแบบโพรโตคอลที่ใช้สื่อสารกันระหว่างเครื่อง

CAgent เป็นเบสคลาสซึ่งมีการกำหนดความสามารถพื้นฐานในการติดต่อสื่อสารระหว่างเอเจนต์

CMasterAgent เอเจนต์ฝั่งเซิร์ฟเวอร์ ทำหน้าที่รับผิดชอบการติดต่อต่างๆ กับไคลเอนต์

CSlaveAgent เอเจนต์ฝั่งไคลเอนต์ ทำหน้าที่รับผิดชอบการติดต่อต่างๆ กับฝั่งเซิร์ฟเวอร์

CFilterRule เป็นโครงสร้างที่ใช้เก็บกฎการฟิลเตอร์ของไฟร์วอล

CW2kFilter ส่วนที่ทำหน้าที่ฟิลเตอร์แพ็คเก็ตโดยใช้ Windows 2000 Packet Filter API

DBInterface ติดต่อกับฐานข้อมูล

FirewallAgent ส่วนโปรแกรมหลักฝั่งไคลเอนต์

FirewallServer ส่วนโปรแกรมหลักฝั่งเซิร์ฟเวอร์

บทที่ 7
เครื่องมือที่ใช้ในการทำโครงงาน

โครงงานนี้ ต้องใช้เครื่องมือต่างๆ และมีวิธีการใช้ ดังต่อไปนี้

7.1 ระบบปฏิบัติการไมโครซอฟท์ วินโดวส์ 2000 เซิร์ฟเวอร์ สำหรับเครื่องแม่ข่าย
ระบบปฏิบัติการไมโครซอฟท์ วินโดวส์ 2000 โปรเฟสชันแนล สำหรับเครื่องลูกข่าย

7.2 แพลตฟอร์ม SDK

รวบรวมข้อมูลในการพัฒนาแอปพลิเคชั่นสำหรับระบบปฏิบัติการวินโดว์ ประกอบไปด้วย เอกสารประกอบการเขียนโปรแกรม Header Files , Library และ Tools ต่างๆ

7.2.1 ระบบปฏิการที่แพลตฟอร์ม SDK สนับสนุนการพัฒนา

- ระบบปฏิบัติการวินโดวส์ XP และวินโดวส์ .NET เซิร์ฟเวอร์
- ระบบปฏิบัติการแอดวานซ์เซิร์ฟเวอร์ Limitied Edition
- ระบบปฏิบัติการวินโดวส์ 2000
- ระบบปฏิบัติการวินโดวส์ NT
- ระบบปฏิบัติการวินโดวส์ 95 , วินโดวส์ 98 และวินโดวส์ ME
- โปรแกรมอินเตอร์เน็ตเอ็กซ์โพลเรอร์ 4.0 และเวอร์ชันสูงกว่า
- .NET Enterprise เซิร์ฟเวอร์

7.2.2 ความต้องการขั้นต่ำเพื่อติดตั้งแพลตฟอร์ม SDK

- ที่ว่างบนฮาร์ดดิสก์อย่างน้อย 1 GB
- หากต้องการติดตั้งโปรแกรม Visual Studio ควรติดตั้งก่อน การติดตั้งแพลตฟอร์ม SDK

7.2.3 การติดตั้งแพลตฟอร์ม SDK

- extract bat file และกำหนดโฟลเดอร์ที่ต้องการเก็บ เช่น
  extract /A /E /Y PSDK-FULL1.cab /L C:\SDK
- เรียกไฟล์ SETUP จาก โฟลเดอร์ที่ Extract ไฟล์เก็บไว้
- เลือก SDK Update และกำหนด Component ที่ต้องการใช้

7.3 ระบบฐานข้อมูล ไมโครซอฟท์ SQL เซิร์ฟเวอร์ 2000

7.3.1 ความต้องการขั้นต่ำของโปรแกรม

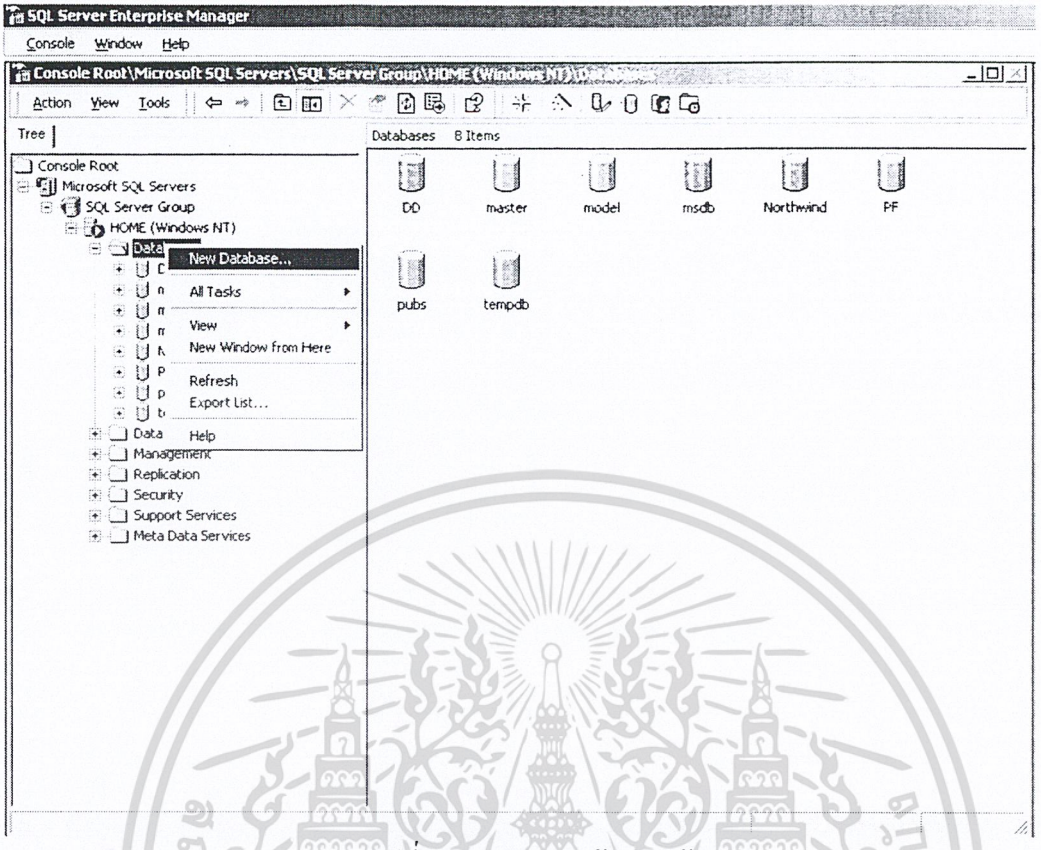
- ซีพียูตระกูลอินเทล หรือตระกูลอื่นที่คอมแพททิเบิ้ล โดยรุ่นต่ำสุดที่ใช้งานได้คือ เพนเทียม 166 MHz
- หน่วยความจำ 128 MB
- ฮาร์ดดิสก์ระหว่าง 95 ถึง 270 MB
- ระบบปฏิบัติการวินโดวส์เอ็นทีเซิร์ฟเวอร์ 4.0 , วินโดวส์ 2000 เซิร์ฟเวอร์

7.3.2 การติดตั้งมีอยู่ด้วยกัน 3 รูปแบบ คือ

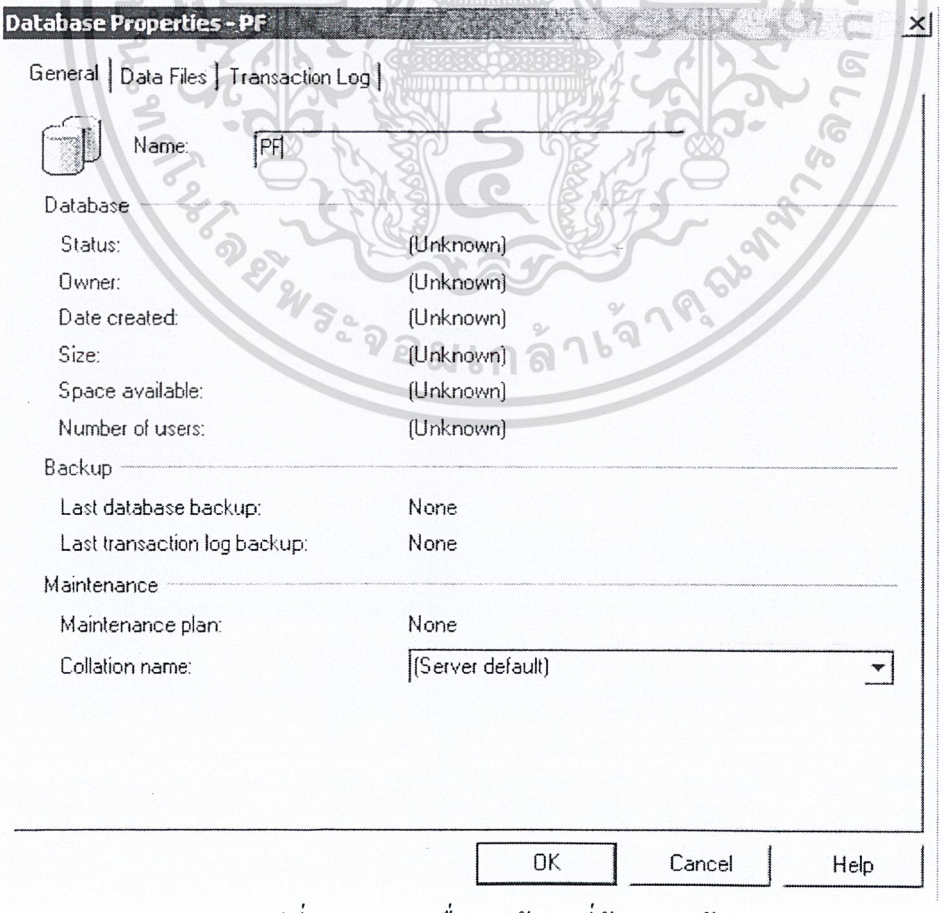
- การติดตั้ง SQL เซิร์ฟเวอร์ 2000 ทั้งในส่วนเซิร์ฟเวอร์คอมโพเน้นท์ และไคลเอนต์คอมโพเน้นท์ ซึ่งจะเป็นการติดตั้งทั้ง DBMS และเครื่องมือที่ใช้ในการจัดการระบบฐานข้อมูลให้อยู่บนเครื่องเดียวกัน ทำให้ผู้ดูแลระบบสามารถจัดการระบบฐานข้อมูล โดยส่งคำสั่งไปให้ DBMS ผ่านทางเครื่องมือต่างๆ ที่มีอยู่ในไคลเอนต์คอมโพเน้นท์บนเครื่องเดียวกัน
- การติดตั้ง SQL เซิร์ฟเวอร์ 2000 เฉพาะไคลเอนต์ การติดตั้งแบบนี้จะติดตั้งเฉพาะส่วนไคลเอนต์คอมโพเน้นท์ ซึ่งทำให้เราสามารถจัดการกับระบบฐานข้อมูลที่อยู่บนเครื่องเซิร์ฟเวอร์อื่นๆ ที่มี SQL เซิร์ฟเวอร์ 2000 เป็น DBMS ได้ โดยผ่านเครื่องมือต่างๆ ที่มีอยู่ในไคลเอนต์คอมโพเน้นท์
- การติดตั้ง SQL เซิร์ฟเวอร์ 2000 เฉพาะในส่วน Connectivity Component จะเป็นการติดตั้งเฉพาะส่วน Microsoft Data Access Components ซึ่งทำให้เครื่องไคลเอนต์สามารถติดตั้งกับเซิร์ฟเวอร์ที่ใช้ SQL เซิร์ฟเวอร์ 2000 เป็นฐานข้อมูลได้

7.3.3 การสร้างฐานข้อมูลโดยใช้ Enterprise Manager

- เรียกเครื่องมือ Enterprise Manager โดยเลือก Start -> Programs -> Microsoft SQL Server -> Enterprise Manager
- เลือกเซิร์ฟเวอร์ที่ต้องการสร้างฐานข้อมูล กดเมาส์ปุ่มขวา เลือก New Database หรือเลือกเมนู Action -> New Database ได้
- จากนั้นจะปรากฏไดอะล็อก Database Properties ที่แท็ป General ให้ทำการตั้งชื่อฐานข้อมูลที่เราสร้างลงไป แล้วกดเมาส์ที่แท็ป Data File เพื่อนิยามคุณสมบัติต่างๆ ในฐานข้อมูลเรา



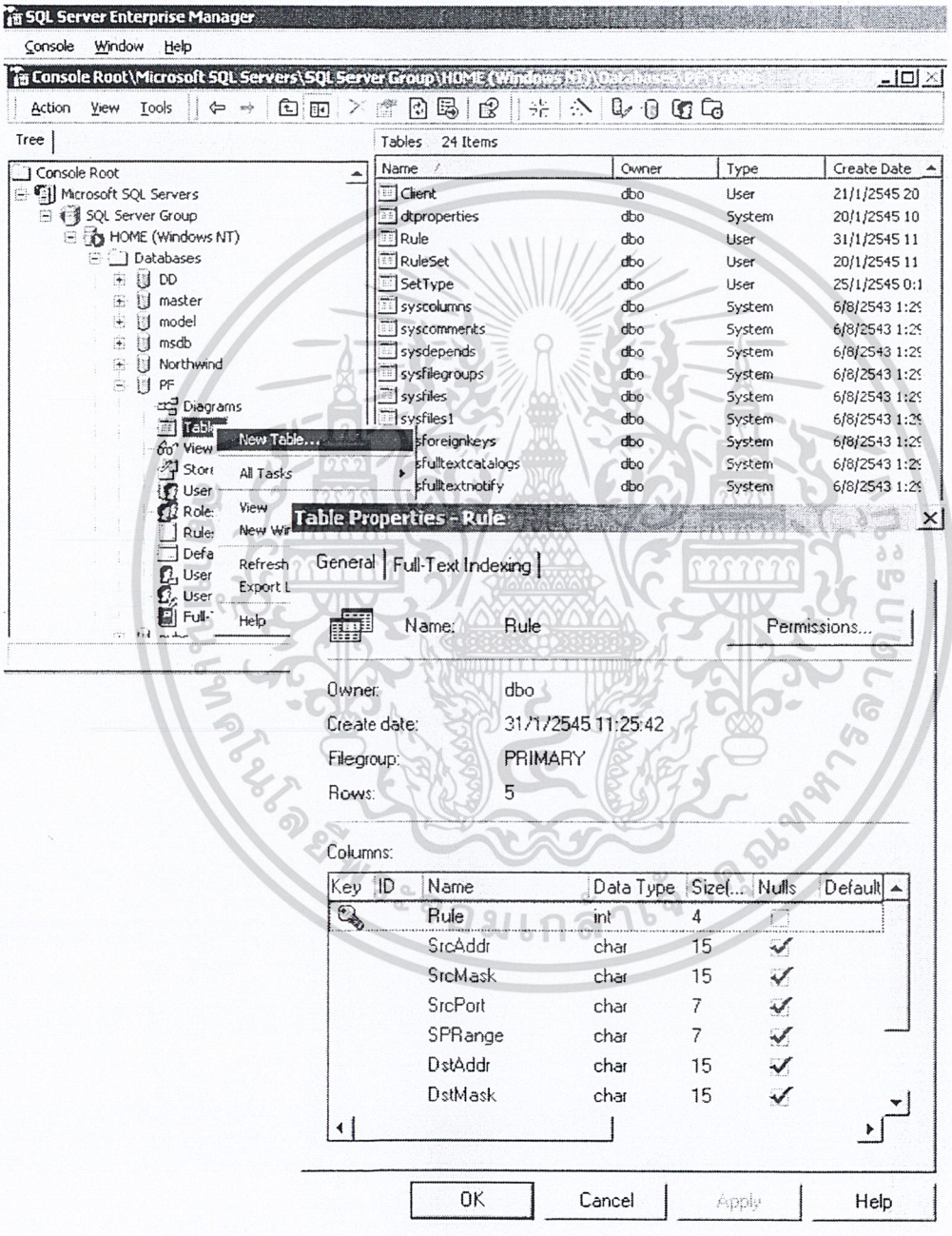
รูปที่ 7-1 แสดงการสร้างฐานข้อมูล



รูปที่ 7-2 กำหนดชื่อฐานข้อมูลที่ต้องการสร้าง

7.3.4 การสร้างตาราง

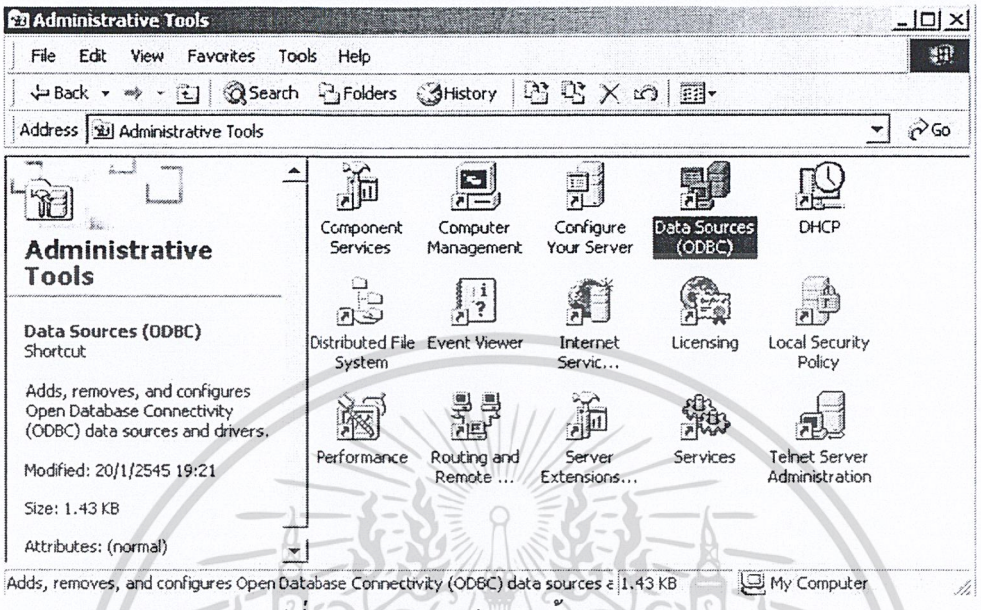
- เลือกเซิร์ฟเวอร์ และเลือกฐานข้อมูลที่เราต้องการสร้างตาราง
- กดเมาส์ปุ่มขวาที่ Table และเลือกเมนู New Table
- จะปรากฏหน้าต่าง Table Design ให้เรากำหนดฟิลด์แต่ละฟิลด์ , กำหนดชนิดข้อมูล , ขนาดข้อมูล และอื่นๆ



รูปที่ 7-3 การสร้างตาราง

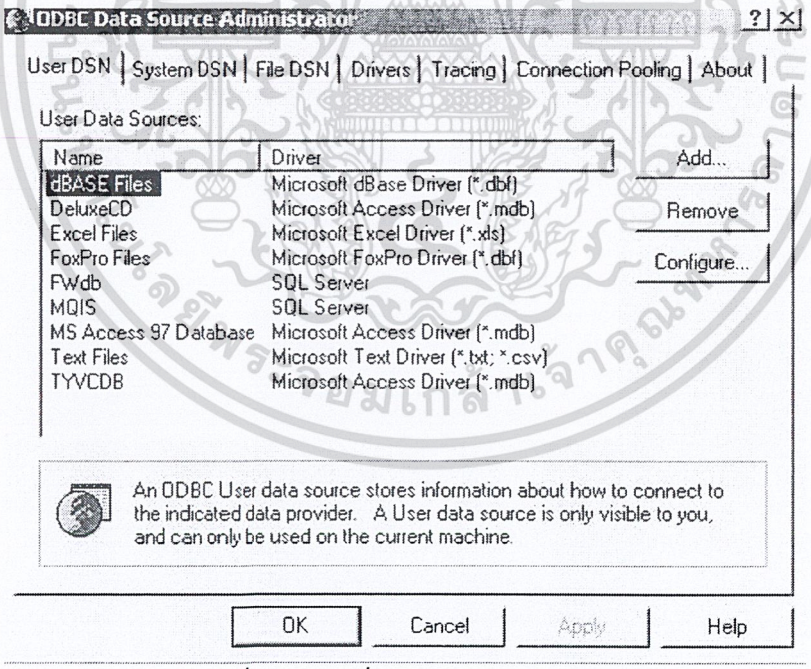
7.3.5 การติดตั้ง ODBC Driver

- เรียก Start -> Programs -> Administrator Tools -> Data Sources (ODBC)



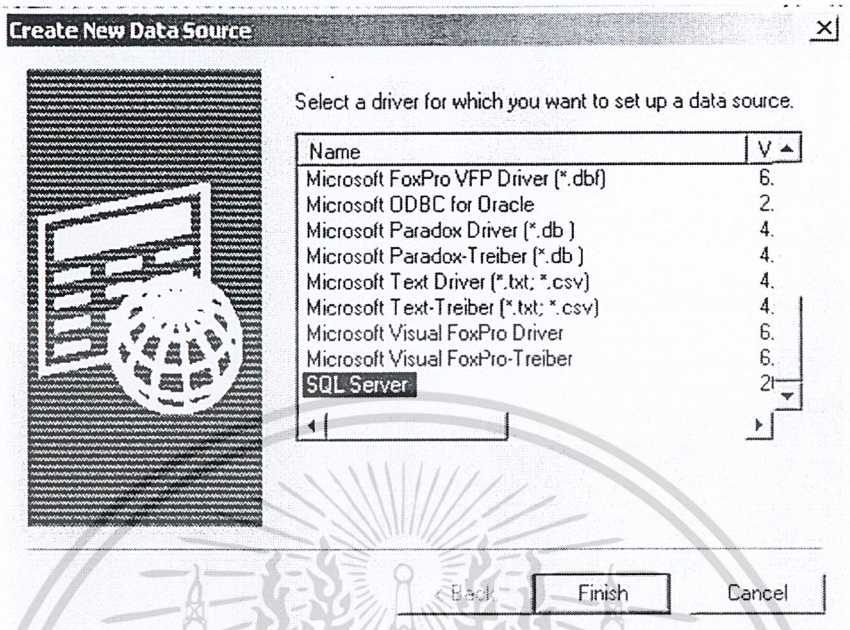
รูปที่ 7-4 แสดงการเรียกติดตั้ง ODBC Driver

- เลือก Add



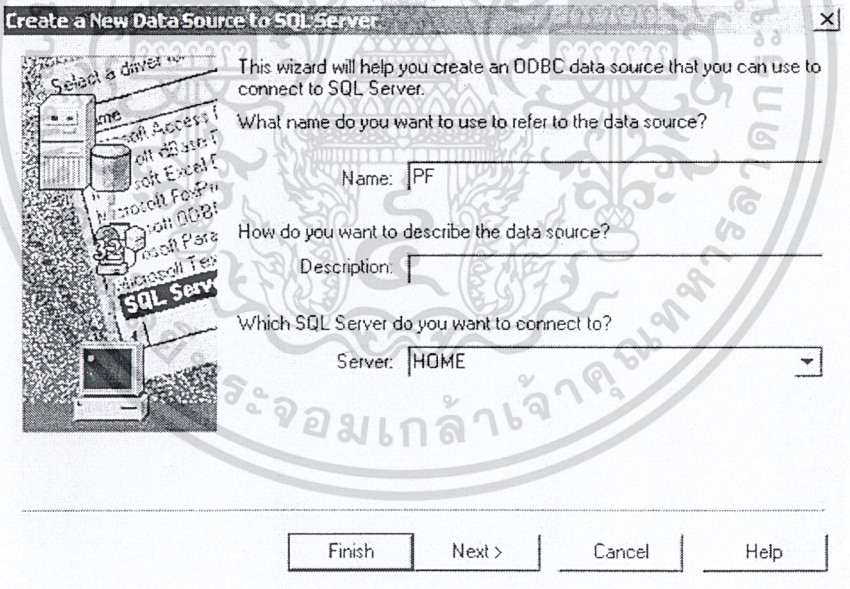
รูปที่ 7-5 การเพิ่มฐานข้อมูลสำหรับใช้งาน

- เลือก SQL Server -> Finish



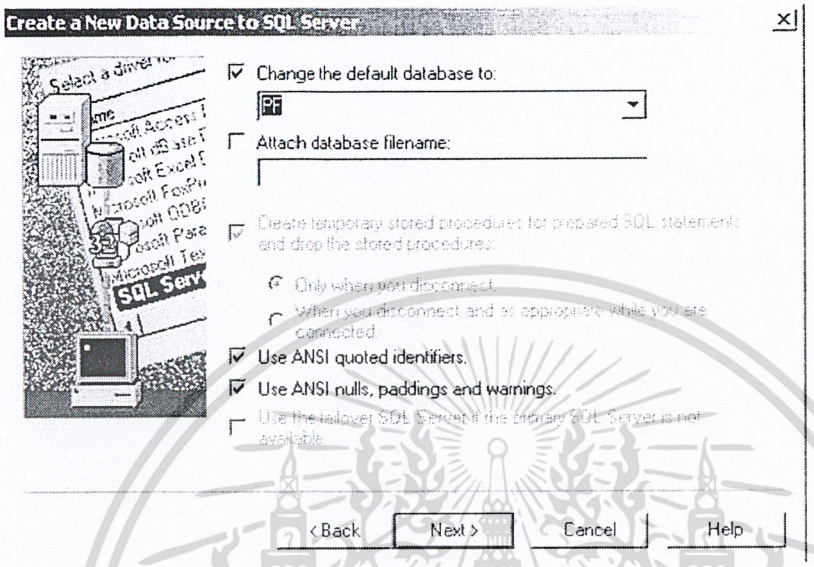
รูปที่ 7-6 เลือกสร้าง SQL เซิร์ฟเวอร์ สำหรับสร้างฐานข้อมูลใหม่

- กำหนด Data Source Name และ Server



รูปที่ 7-7 ตั้งชื่อฐานข้อมูลใหม่

- เลือกไฟล์ฐานข้อมูลที่ต้องการ



รูปที่ 7-8 เลือกชื่อไฟล์ฐานข้อมูล

บทที่ 8
สรุปและวิจารณ์ผลการทดลอง

จากการพัฒนาสถาปัตยกรรมของโปรแกรม ที่ได้ออกแบบนั้น ระบบสามารถทำงานได้บนระบบปฏิบัติการวินโดวส์ 2000 ไฟร์วอลเซิร์ฟเวอร์สามารถติดต่อกับไฟร์วอลเอเจนต์ในเครื่องไคลเอนต์ กรองแพ็คเก็ตตามกฎเกณฑ์ และติดต่อกับระบบฐานข้อมูลได้ อย่างถูกต้อง แม้ว่าบางเครื่องซึ่งใช้ ระบบปฏิบัติการวินโดว์ 2000 ที่สนับสนุนการทำงานของวินโดวส์แพ็คเก็ตฟิลเตอร์ API ไม่สามารถทำงานได้

ในขั้นตอนการพัฒนาโปรแกรม ผู้จัดทำประสบปัญหาดังนี้

- การหาข้อมูลเกี่ยวกับ NDIS ของวินโดวส์ทำได้ลำบาก ข้อมูลที่หาได้มีค่อนข้างน้อย
- ไลบรารีที่ใช้ในการฟิลเตอร์แพ็คเก็ตหาได้ยาก ไลบรารีที่มีความยืดหยุ่นในการใช้งานมีราคาสูง
- เนื่องจากมีการศึกษาและพัฒนาโปรแกรมด้วย Visual C++ ไปในขณะเดียวกัน จึงทำให้เกิดความติดขัดในการพัฒนา การเลือกวิธีในการประยุกต์บางส่วนอาจจะไม่เหมาะสม
- การออกแบบโปรแกรมในช่วงแรกยังไม่สามารถออกแบบได้ครบถ้วนดีนัก เนื่องจากขาดความรู้ทางด้านเทคนิคในการประยุกต์ จึงทำให้ยังไม่เข้าใจว่าแต่ละส่วนนั้นจะต้องทำด้วยวิธีอะไร
- เนื่องจากการพัฒนาโปรแกรมเป็นไปโดยที่การออกแบบโปรแกรมยังไม่สมบูรณ์ดีนัก จึงทำให้มีปัญหาในขั้นตอนการเอาโปรแกรมแต่ละส่วนมารวมกัน

จากที่ได้กล่าวมาแล้วนั้น โครงงานนี้เป็นการพัฒนาสถาปัตยกรรมของไฟร์วอล เพื่อให้พอร์ซันนอลไฟร์วอลสามาถนำมาใช้งานได้อย่างมีประสิทธิภาพ สามารถควบคุม กระจายกฎต่างๆ ได้จากส่วนกลาง จึงใช้แพ็คเก็ตฟิลเตอร์ริ่ง ซึ่งเป็นวิธีในการฟิลเตอร์แพ็คเก็ตแบบพื้นฐาน และไม่สามารถป้องกันการโจมตีบางประเภทได้ เนื่องจากไม่มีการตรวจสอบในส่วนของเนื้อหาภายในแพ็คเก็ต อีกทั้งข้อจำกัดของวินโดวส์ 2000 แพ็คเก็ต ฟิลเตอร์ริ่ง API ซึ่งไม่สามารถนำฟิลด์อ๊อบชั่น และแพดดิ้งต่างๆ มาใช้ในการฟิลเตอร์ได้ จึงควรพัฒนาวิธีการในการฟิลเตอร์แพ็คเก็ตแบบสเตทฟูล อินสเป็คชั่น หรือแบบแอปพลิเคชั่นเกทเวย์ซึ่งมีความปลอดภัยสูงกว่าแทน

ในเรื่องของการพิสูจน์สิทธิ์ เนื่องจากเราจำเป็นต้องมีวิธีการที่จะพิสูจน์ว่าเครื่องไคลเอนท์ที่ทำการติดต่อด้วยเป็นเครื่องไคลเอนท์เครื่องนั้นจริงๆ จึงจำเป็นจะต้องมีวิธีการพิสูจน์สิทธิ์ที่เหมาะสม ซึ่งอาจจะพัฒนาให้ระบบไฟร์วอลทำงานร่วมกับระบบการพิสูจน์สิทธิ์ที่ใช้กันอยู่แล้วในองค์กร

บรรณานุกรม

[1] Elizabeth D. Zwicky, Simon Cooper & D. Brent Chapman (2000) : "Building Internet Firewalls 2nd Edition", O'Reilly, California.

[2] Joel Scambray, Stuart McClure, George Kurtz (2001) : "Hacking Exposed : Network Security Secrets & Solutions", McGraw-Hill.

[3] Terry Wiliam Ogletree (2000) : "Practical Firewalls", Que, Indiana.

[4] Ton Plooy (2000) : "Packet Filtering with iphlpapi.dll", Windows developer's journal, October 2000

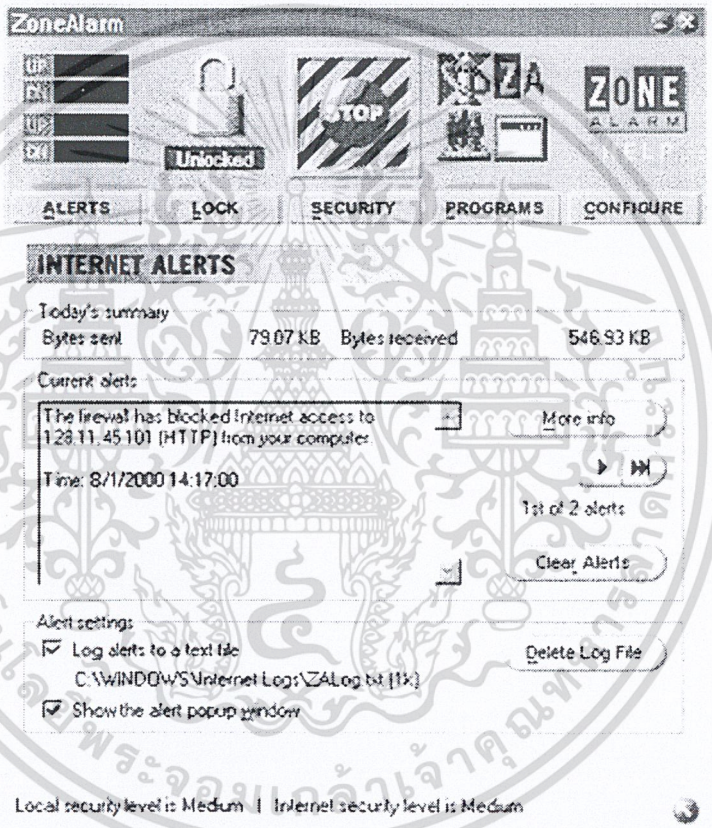
[5] เรืองไกร รังสิพล (2544) : "เจาะระบบ TCP/IP : จุดอ่อนของโปรโตคอลและวิธีป้องกัน", Provision, กรุงเทพฯ

ภาคผนวก

เพอร์ซันนอลไฟร์วอลที่น่าสนใจ

เนื่องจากในปัจจุบันนี้มีเพอร์ซันนอลไฟร์วอล ออกมาเป็นจำนวนมาก จึงทำการเลือกขึ้นมาบางส่วน เพื่อมาทดสอบการทำงาน โดยได้เลือกโปรแกรมที่มีชื่อเสียงและคุณภาพเป็นเกณฑ์ในการเลือก

1. ZoneAlarm 2.1.44



โซนอลาร์ม (ZoneAlarm) นำวิธีการ ของแอปพลิเคชัน-เลเวล เกตเวย์ มาใช้ในการรักษาความปลอดภัย ออกแบบง่ายต่อการใช้ สามารถกำหนดค่าต่างๆได้เอง ทำการเช็คทุกๆ แอปพลิเคชันที่ไม่มีกฎเกณฑ์ตามที่กำหนดไว้ การทำงานพื้นฐานของโซนอลาร์มจะทำการซ่อนพอร์ตทุกพอร์ตเข้าสู่สตีลท์ โมด (stealth mode) ทั้งหมด ซึ่งเป็นการทำให้เหมือนว่าเครื่องของเราไม่มีพอร์ตนั้นๆอยู่ โดยการ ignore request ที่เข้ามาทิ้งไปเฉยๆ

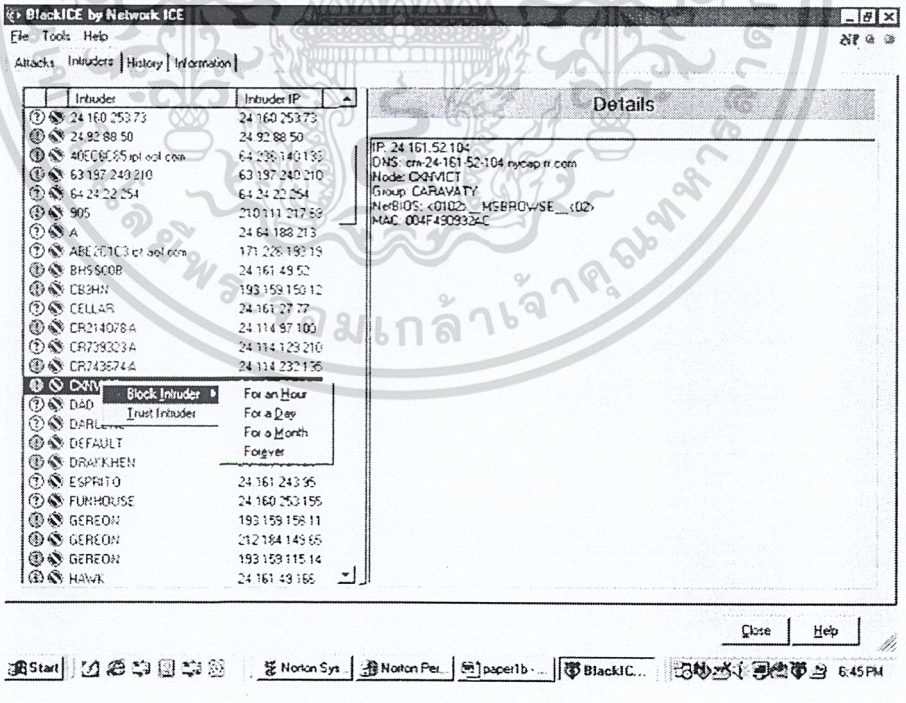
วิธีการของโซนอลาร์มที่นอกเหนือจากทั่วไปก็คือสำหรับโปรแกรมภายในที่ต้องการจะติดต่อภายนอกจะมีการแจ้งขึ้นมาว่าจะอนุญาตหรือไม่ รวมถึงโปรแกรมที่ต้องการจะทำตัวเป็นเซิร์ฟเวอร์ก็จะมีการถามขึ้นมาเช่นกัน ซึ่งวิธีนี้จะทำให้เราสามารถจะป้องกันโปรแกรมที่เราไม่ต้องการไม่ให้ติดต่อสู่ภายนอกได้ เช่น ม้าโทรจัน (Trojan Horse) หรือพวกแอด-แวร์ (Ad-ware) ต่างๆ

โซนอลาร์มยังมีระบบป้องกันผู้บุกรุก ซึ่งจะมีการแจ้งเตือนเมื่อมีคนพยายามจะบุกรุกเข้ามาในเครื่องของเราด้วยวิธีการสแกนพอร์ตหรือวิธีอื่นๆ และโซนอลาร์มยังสามารถกำหนด trusted zone ซึ่งเป็นการกำหนดหมายเลขไอพี ที่เรายอมให้ผ่านเข้ามาได้ ซึ่งเหมาะสำหรับเครื่องที่อยู่ในเครือข่าย

อีกความสามารถหนึ่งของโซนอลาร์มก็คืออินเทอร์เน็ตล็อค (Internet Lock) ซึ่งเมื่อสั่งให้ทำงาน โซนอลาร์มจะทำการ บล็อคทุกๆอย่างที่เข้ามาที่เครื่องของเรา ซึ่งเราสามารถที่จะสั่งให้ฟังชั่นนี้ทำงานด้วยตัวเองหรือจะตั้งให้ทำงานตามกำหนดก็ได้ และเรายังสามารถตั้งให้บางโปรแกรมเช่น โปรแกรมรับส่งเมล์สามารถผ่านอินเทอร์เน็ตล็อค เพื่อใช้งานตามปกติได้อีกด้วย

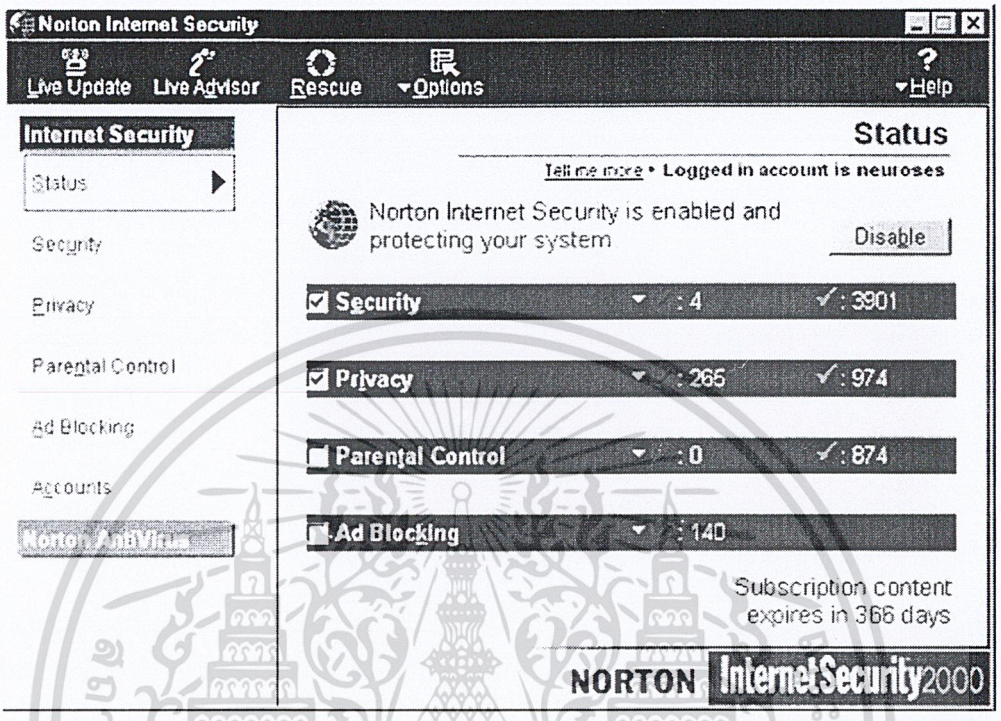
2. BlackICE Defender

เป็นหนึ่งในเพอร์ซันนอลไฟร์วอลที่มีชื่อเสียงมากที่สุด เนื่องจากเป็นโปรแกรมที่ใช้งานง่าย ไม่จำเป็นจะต้องมีความรู้มากมาย และมีประสิทธิภาพที่ดี โดยมีการนำวิธีการของแพ็คเก็ตฟิลเตอร์ริ่ง มาใช้ร่วมกับระบบตรวจจับผู้บุกรุก(intrusion detection) ซึ่งถ้ามีการตรวจพบการโจมตีแบล็คไอซ์ จะมีการแจ้งเตือนขึ้นมาและเก็บรายละเอียดของผู้โจมตีเอาไว้ด้วยความสามารถที่เรียกว่า Back Trace ซึ่งก็ขึ้นอยู่กับการตั้งค่าของผู้ใช้ ซึ่งจุดนี้ก็เป็นจุดเด่นจุดหนึ่งของแบล็คไอซ์เนื่องจากความยืดหยุ่นในการตั้งค่า สำหรับการทำงานพื้นฐานแบล็คไอซ์ก็จะทำการซ่อนพอร์ตทั้งหมดเข้าสู่สตีลท์ โมด เช่นเดียวกันกับโซนอลาร์ม



แต่แบล็คไอซ์ไม่มีความสามารถในการที่จะตรวจสอบข้อมูลขาออก ดังนั้นจึงไม่สามารถป้องกันม้าโทรจันหรือโปรแกรมประเภทแอด-แวร์ได้

3. Norton Personal Firewall



นอร์ตันเพอร์ซันนอลไฟร์วอล(Norton Personal Firewall) เป็นไฟร์วอล ที่ใช้วิธีการแบบดั้งเดิมคือ ตรวจสอบทุกๆ แพ็คเก็ต ที่เข้ามาหรือออกไปจากเครื่อง และนำไปเปรียบเทียบกันกฎที่ตั้งไว้ว่าจะอนุญาตให้ผ่านหรือไม่ ซึ่งโดยปกติจะเป็นการยากในการใช้สำหรับคนที่ไม่มีความรู้เรื่องเครือข่าย แต่นอร์ตันเพอร์ซันนอลไฟร์วอล ได้ช่วยในส่วนนี้โดยมีกฎที่เขียนขึ้นมาไว้แล้วในรูปแบบต่างๆที่มีในสถานการณ์ต่างๆ และยังมีวิซาร์ด(wizard) ซึ่งช่วยในการกำหนดกฎเองด้วย แต่สำหรับผู้ใช้ระดับสูงก็สามารถที่จะเข้าไปแก้ในส่วนของรายละเอียดต่างๆได้ทั้งหมด

และก็เหมือนกับทั้งสองโปรแกรมก่อนหน้านี้นอร์ตันเพอร์ซันนอลไฟร์วอล ก็จัดการให้เครื่องอยู่ในสตีลท์ โมด(stealth mode) เช่นกัน ซึ่งหมายความว่าผู้บุกรุกจะมองไม่เห็นว่ามีพอร์ตต่างๆเหล่านี้อยู่

สิ่งที่มีประโยชน์ข้อหนึ่งสำหรับนอร์ตันเพอร์ซันนอลไฟร์วอลก็คือ ในขณะที่โปรแกรมบางโปรแกรมที่มีการตรวจสอบ ข้อมูลขาออกจากโปรแกรมต่างจะต้องมีการถามผู้ใช้ทุกครั้ง สำหรับโปรแกรมที่ต้องการติดต่อกับภายนอกนอร์ตันเพอร์ซันนอลไฟร์วอล สามารถที่จะสแกนฮาร์ดดิสก์ ทั้งหมดสำหรับโปรแกรมที่ต้องการติดต่ออินเทอร์เน็ตตั้งแต่ขั้นตอนการ ติดตั้ง

นอร์ตันเพอร์ซันนอลไฟร์วอลเหมือนกับแบล็คไอซ์ ตรงที่สามารถที่จะตรวจจับผู้ที่พยายามจะสแกนพอร์ตของเครื่องและทำการบล็อคทุกๆการเข้าถึงของผู้บุกรุกและก็เหมือนกับโซนอลาร์มตรงที่สามารถกำหนดทรัสต์ โซน(trusted zone) ซึ่งทำให้ง่ายในการตั้งค่าโปรแกรมสำหรับเครือข่าย แต่นอร์ตันเพอร์ซันนอลไฟร์วอล ไม่สามารถที่จะแยกแยะระหว่างการติดต่อภายในกับภายนอกเครือข่ายได้

นอกเหนือจากส่วนของไฟร์วอลแล้วนอร์ตันเพอร์ซันนอลไฟร์วอล ก็ยังมีส่วนของ Privacy Protection ซึ่งจะคอยดูแลในเรื่องของการส่งข้อมูลส่วนตัวไปในเว็บไซท์ต่างๆอีกด้วย