บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

การสำรวจบทความ

จากการสำรวจหาโปรแกรมที่มีการทำงานคล้ายคลึงกับโปรแกรมประสานระหว่างโปรแกรมประยุกต์ พบว่ามีโปรแกรมอยู่สองกลุ่มที่มีการทำงานที่ใกล้เคียงกับโปรแกรมประสานระหว่างโปรแกรมประยุกต์ คือ

1. โปรแกรมเชื่อมต่อโดยพิธีการเท็ลเน็ต เป็นโปรแกรมที่ทำการเชื่อมต่อไปยังเครื่องแม่ข่ายโดยพิธีการเท็ลเน็ต โปรแกรมประเภทนี้สามารถเชื่อมต่อไปยัง เครื่องแม่ข่ายแล้วจะทำการเลียนแบบเครื่องปลายทางชนิดต่างๆ ตามที่โปรแกรมสนับสนุน แต่โปรแกรมเหล่านี้มิได้คำนึงถึงการนำเอาอักขระต่างๆ บนส่วนของหน้าจอที่เกิดจากการเลียนแบบเครื่องปลายทางที่ตนเองสนับสนุนมาใช้ประโยชน์ ลักษณะการทำงานของโปรแกรมจะโต้ตอบกับผู้ใช้ได้โดยตรง ตัวอย่างของโปรแกรมประเภทนี้ เช่น โปรแกรมเท็ลเน็ตของไมโครซอร์ฟวินโดว์

2. ตัวประสานระหว่างโปรแกรมของฐานข้อมูล ลักษณะการทำงานของโปรแกรมจะไม่โต้ตอบกับผู้ใช้ได้โดยตรง ผู้ใช้ต้องเรียกใช้ตัวประสานนี้ผ่านชุดคำสั่งต่างๆ เรียงกันตามลำดับที่ถูกกำหนดไว้ การใช้งานจึงค่อนข้างใช้ความรู้ทางด้านการเขียนโปรแกรมอยู่พอสมควร จึงทำให้มีความสามารถในการดึงข้อมูลไม่ดีนัก แต่ตัวประสานเหล่านี้สามารถดึงข้อมูลระบียนใดก็ได้ไม่มีข้อจำกัด ตัวอย่างของตัวประสานประเภทนี้ ได้แก่ โอดีบีซี (ODBC) ของไมโครซอร์ฟ

พิธีการเท็ลเน็ต

พิธีการเท็ลเน็ต เป็นตัวกำหนดการสื่อสารเชิงโต้ตอบ ที่อนุญาตให้ผู้ใช้ สื่อสารกับ การให้บริการ (service) ของเครื่องคอมพิวเตอร์แม่ข่าย ส่วนใหญ่แล้วผู้ใช้จะใช้พิธีการเท็ลเน็ต ในการสื่อ

สารกับ การให้บริการ ลงบันทึกเข้า (login) ระยะไกล นอกจากนี้ พิธีการเท็ลเน็ตยังยอมให้ผู้ใช้ ทำการสื่อสารกับการให้บริการแบบอื่นๆ ได้

1. องค์ประกอบของพิธีการเท็ลเน็ต

1.1 เครื่องปลายทางของผู้ใช้

พิธีการเท็ลเน็ต เป็นตัวกำหนด การสื่อสารเชิงอักขระแบบโต้ตอบ พร้อมกันนั้น ยังเป็นตัวกำหนด เครื่องปลายทางข่ายงานเสมือน (Network Virtual Terminal) ซึ่งจะย่อว่า NVT ที่ประกอบไปด้วย แผงแป้นอักขระ และจอแสดงผล ตัวพิธีการเอง ยังได้กำหนด กลุ่มของอักขระสำหรับเครื่องปลายทางเสมือน โดยที่แป้นอักขระบางตัว มีความหมายเป็นการกระทำ แทนที่มีจะความหมายเป็นค่าอักขระ ตัวอย่างเช่น มีแป้นอักขระสำหรับ ยกเลิก หรือหยุดการทำงาน ข้อได้เปรียบที่เห็นได้เด่นชัดของการใช้เครื่องปลายทางข่ายงานเสมือน ก็คือ การยินยอมให้ ผู้ใช้บริการ (client) จากเครื่องคอมพิวเตอร์ ที่แตกต่างกันไป สามารถเชื่อมต่อเข้ามา ขอใช้บริการต่างๆ ได้ พิธีการเท็ลเน็ตใช้การแทนข้อมูลแบบสมมาตร คือ ในการส่งข้อมูล ผู้ใช้บริการต้องเปลี่ยนการแทนอักขระของเครื่องปลายทางระยะใกล้ของตน ไปเป็นการแทนอักขระแบบ เครื่องปลายทางข่ายงานเสมือน ส่วนในการรับข้อมูล ก็ต้องเปลี่ยนการแทนอักขระแบบ เครื่องปลายทางข่ายงานเสมือน มาเป็นการแทนอักขระแบบของตน กล่าวคือ พิธีการเท็ลเน็ต เป็นพิธีการเชิงอักขระ ที่มีมาตราฐานของการเข้ารหัสในการถ่ายโอนข้อมูล

1.2 สารสนเทศที่เป็นคำสั่งงาน และการควบคุม

นอกจากข้อมูลอักขระแล้ว พิธีการเท็ลเน็ต ยินยอมให้ผู้ใช้บริการ และผู้ให้บริการทำการแลกเปลี่ยนสารสนเทศที่เป็นคำสั่งงาน หรือเป็นการควบคุม เพราะว่าการสื่อสารทุกอย่างระหว่างผู้ใช้บริการ กับผู้ให้บริการ จะถูกส่งข้ามผ่านการเชื่อมต่อแบบ ทีซีพี (TCP) เพียงช่องเดียว ดังนั้นพิธีการเท็ลเน็ต จะเตรียมการสำหรับการเข้ารหัสสารสนเทศที่เป็นคำสั่งงาน หรือสารสนเทศที่เป็นการควบคุม ซึ่งจะทำให้ผู้รับสามารถแยกสารสนเทศเหล่านั้น ออกจากข้อมูลธรรมดาได้ ใจความสำคัญของพิธีการเน้นที่การกำหนดวิธีเข้ารหัสตัวคำสั่งงานของผู้ส่ง และวิธีที่ผู้รับจะรับรู้ถึงคำสั่งงานเหล่านั้น

1.3 เครื่องปลายทาง วินโดว์ และแฟ้มข้อมูล

พิธีการเท็ลเน็ต ได้กำหนดการสื่อสาร ระหว่างเครื่องปลายของผู้ใช้ กับการให้บริการระยะไกล ข้อกำหนดของพิธีการมีสมมุติฐานว่า เครื่องปลายทางประกอบไปด้วยแผงแป้นอักขระที่ผู้ใช้สามารถป้อนอักขระ และมีจอภาพแสดงผล ที่สามารถแสดงผลที่เป็นข้อความได้หลายบรรทัดพร้อมๆ กัน

1.4 ความต้องการ ภาวะพร้อมกัน

ในเชิงแนวคิดแล้ว ผู้ใช้บริการเท็ลเน็ต ถ่ายโอนอักขระ ระหว่างเครื่องปลายทางของผู้ใช้ กับการให้บริการระยะไกลดังแสดงในรูปที่ 2.1 ด้านหนึ่งของผู้ใช้บริการเท็ลเน็ตใช้ฟังก์ชันของระบบปฏิบัติการ ระยะใกล้ในการโต้ตอบกับเครื่องปลายทางของผู้ใช้ ส่วนอีกด้านหนึ่งของผู้ใช้บริการเท็ลเน็ต ก็ใช้การเชื่อมต่อ ทีซีพี ในการสื่อสารกับการให้บริการระยะไกล

รูปที่ 2.1 บทบาทของผู้ใช้บริการเท็ลเน็ต

ในการจัดทำการเชื่อมต่อแบบสองทางเต็มอัตรา (full duplex) ระหว่างเครื่องปลายทางของผู้ใช้ กับการให้บริการระยะไกล ผู้ใช้บริการเท็ลเน็ตจะต้องทำภารกิจสองอย่างในขณะเดียวกัน คือ

1.4.1 ผู้ใช้บริการ ต้องอ่านอักขระที่ผู้ใช้ พิมพ์ผ่านแผงแป้นอักขระ แล้วส่งอักขระเหล่านั้น ข้ามผ่านการเชื่อมต่อ ทีซีพี ไปยังการให้บริการระยะไกล

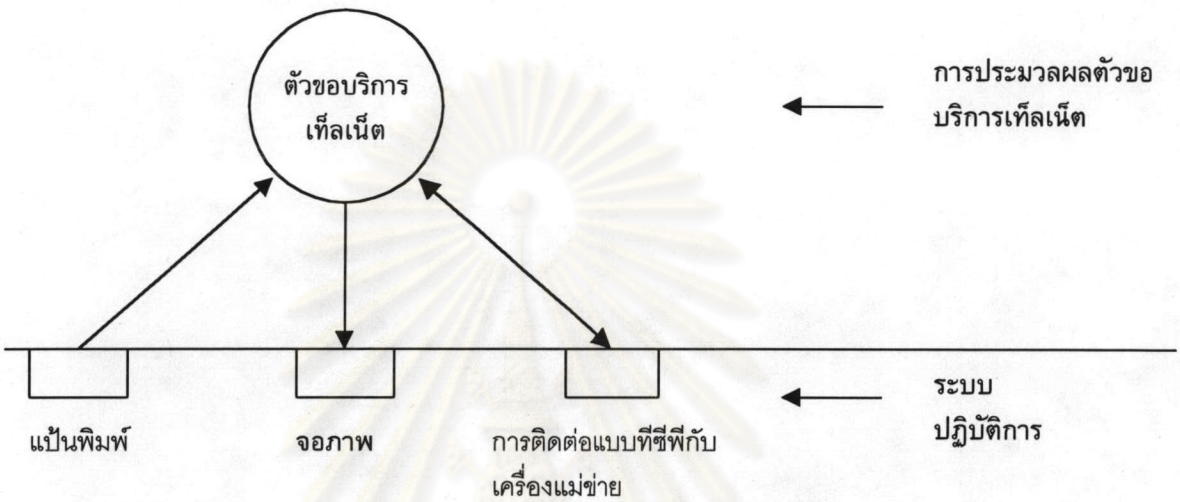
1.4.2 ผู้ให้บริการ ต้องอ่านอักขระที่เข้ามาทางการเชื่อมต่อ ทีซีพี แล้วแสดงอักขระเหล่านั้นไปบนจอภาพเครื่องปลายทางของผู้ใช้

เนื่องจาก การให้บริการระยะไกล สามารถส่งผลลัพธ์ได้ตลอดเวลา ในขณะที่ผู้ใช้ก็สามารถพิมพ์แผงแป้นอักขระขณะใดก็ได้ ดังนั้นผู้ใช้บริการเท็ลเน็ตจะไม่สามารถหยุดรอ สิ่งเข้า (input) อย่างไม่มีเวลาสิ้นสุดเพียงแหล่งกำเนิดเดียว จากทั้งหมดสองแหล่งกำเนิดได้โดยปราศจากการตรวจสอบว่า มีสิ่งเข้าจากแหล่งกำเนิดที่เหลือหรือไม่ โดยสรุปก็คือ ผู้ใช้บริการเท็ลเน็ตต้องโอนถ่ายข้อมูลไปพร้อมๆ กัน ทั้งสองทิศทาง

1.5 แบบจำลองการประมวลผลของผู้ใช้บริการเท็ลเน็ต

ในการถ่ายโอนข้อมูลแบบพร้อมกัน ผู้ใช้บริการต้องเลือกทำ ระหว่างการที่ผู้ใช้บริการต้องประกอบไปด้วยการประมวลผลหลายๆ อันที่กระทำการไปพร้อมๆ กัน กับการทำให้เกิด

การรับเข้า/ส่งออก เป็นไปพร้อมๆ กันในการประมวลผลเดียวกัน ในวิทยานิพนธ์นี้ใช้ยุทธวิธีแบบหลัง คือ ผู้ใช้บริการเท็ลเน็ตประกอบด้วยการประมวลผลเดียว ที่จะรอจนกระทั่ง แหล่งกำเนิดตัวใดตัวหนึ่งพร้อมที่จะส่งข้อมูล



รูปที่ 2.2 แบบจำลองการประมวลผลของผู้ใช้บริการเท็ลเน็ต

2. ขั้นตอนของ ผู้ใช้บริการเท็ลเน็ต

2.1 วิเคราะห์กระจาย (parse) อาร์กิวเมนต์ และกำหนดค่าเริ่มต้นของโครงสร้างข้อมูล

2.2 เปิด การเชื่อมต่อ ทีซีพี หนึ่งช่อง ไปยังทาง (port) ที่ถูกกำหนด สู่เครื่องระยะไกลที่ถูกกำหนด

2.3 หยุดรอจนกระทั่งผู้ใช้พิมพ์สิ่งเข้า หรือรอจนกระทั่งข้อมูลเข้ามาทางการเชื่อมต่อทีซีพี

2.4 ถ้าข้อมูลเข้ามาทางแป้นพิมพ์อักขระ ก็อ่านเข้ามาประมวลผล ก่อนนำมาแปลงให้เป็นไปตามการแทนอักขระแบบ เครื่องปลายทางข่ายงานเสมือน แล้วส่งออกไปทางการเชื่อมต่อ ทีซีพี ถ้าข้อมูลเข้ามาทางการเชื่อมต่อ ทีซีพี ก็อ่านเข้ามาประมวลผล ก่อนนำมาแปลงให้เป็นไปตามการแทนอักขระระยะใกล้ แล้วส่งออกไปยังที่แสดงผลของผู้ใช้

2.5 กลับไปที่ ขั้นตอนที่ 2.3

3. ข้อกำหนดคุณลักษณะของ เครื่องสถานะจำกัด (Finite State Machine)

พิธีการเท็ลเน็ต ได้กำหนดรูปแบบในการส่งอักขระของผู้ใช้บริการผ่านไปยังการให้บริการระยะไกล และกำหนดรูปแบบในการแสดงผลของข้อมูลของผู้ใช้บริการที่การบริการระยะไกลตอบกลับมา ส่วนใหญ่ของสิ่งที่ถูกส่งไปมาข้ามการเชื่อมต่อ ก็คือ อักขระข้อมูลที่เป็นตัวๆ อักขระข้อมูลแต่ละตัวเกิดขึ้นที่ผู้ใช้บริการ ในขณะที่ผู้ใช้พิมพ์แป้นอักขระ หรืออักขระข้อมูลเกิดขึ้นจากผู้ให้บริการเมื่อการประมวลผลระยะไกลสร้าง สิ่งออก (output) นอกจากอักขระข้อมูลแล้ว พิธีการเท็ลเน็ตยังยอมให้ผู้ใช้บริการและผู้ให้บริการ แลกเปลี่ยนสารสนเทศที่ใช้ในการควบคุม ในบางครั้งผู้ใช้บริการก็สามารถส่งลำดับของอักขระที่ประกอบกันเป็นคำสั่งหนึ่งคำสั่งให้กับผู้ให้บริการ ซึ่งเป็นผู้ควบคุมการกระทำของการให้บริการระยะไกล ตัวอย่างเช่น ผู้ใช้บริการสามารถส่งลำดับคำสั่งที่ขัดจังหวะโปรแกรมประยุกต์ระยะไกล ได้

ส่วนใหญ่ในการสร้างเท็ลเน็ตขึ้นจริง จะใช้เครื่องสถานะจำกัดที่กำหนด วากยสัมพันธ์ (syntax) ไว้อย่างแน่นอน และกำหนดการแปลคำสั่งของลำดับคำสั่ง เครื่องสถานะจำกัดถูกใช้เป็นเครื่องมือในการกำหนดคุณลักษณะอย่างแพร่หลาย เนื่องจากเครื่องสถานะจำกัดเป็นตัวอธิบายพิธีการได้อย่างถูกต้องและแม่นยำมาก เพราะสามารถแสดงรูปแบบของการฝังลำดับคำสั่งเข้าไปในกระแสข้อมูลได้อย่างชัดเจน และยังเป็นตัวกำหนดรูปแบบของการแปลลำดับคำสั่งเหล่านั้นของผู้รับได้อย่างชัดเจน จุดที่สำคัญก็คือ เครื่องสถานะจำกัด สามารถถูกแปลงเป็นโปรแกรมที่ทำงานตามพิธีการได้โดยตรง ดังนั้นก็เป็นไปได้ที่จะทำการตรวจสอบโปรแกรมที่ถูกสร้างขึ้นนี้ ว่าปฏิบัติตามข้อกำหนดคุณลักษณะของพิธีการหรือไม่

เนื่องจากพิธีการเท็ลเน็ต เป็นพิธีการเชิงอักขระที่ฝังลำดับคำสั่งเข้าไปในกระแสข้อมูลระหว่างผู้ใช้บริการ กับผู้ให้บริการ ดังนั้นในการสร้างจริงส่วนใหญ่จึงใช้เครื่องสถานะจำกัด เป็นเครื่องมือในการกำหนดพฤติกรรมของพิธีการได้อย่างถูกต้อง

<u>การถ่ายโอนข้อมูลบนระบบปฏิบัติการวินโดว์</u>

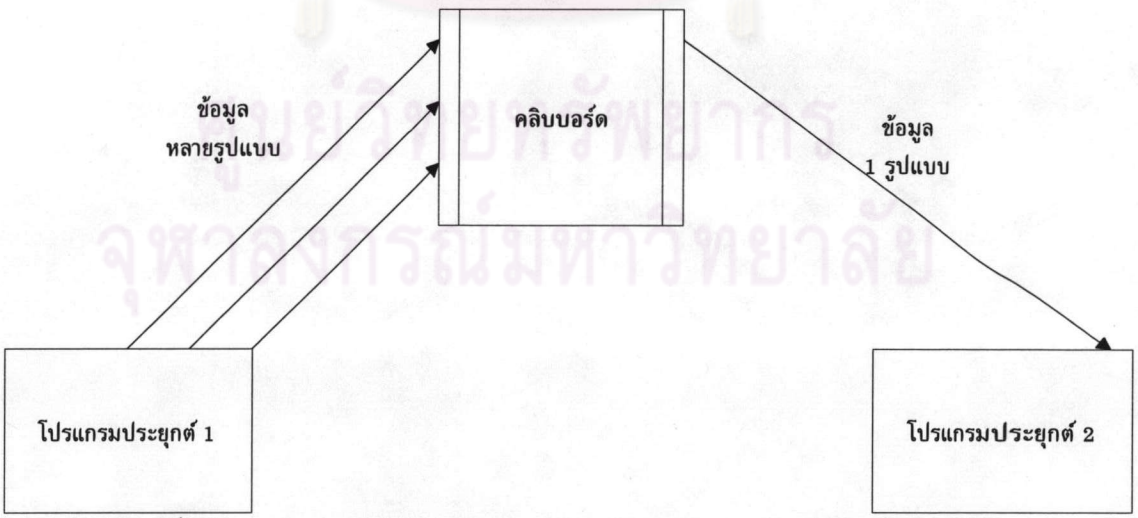
ระบบปฏิบัติการวินโดว์ได้ออกแบบให้โปรแกรมประยุกต์ต่างๆ สามารถถ่ายโอนข้อมูลไปมาระหว่างกันและกันได้ ซึ่งวิธีในการถ่ายโอนข้อมูลนั้นมีอยู่ด้วยกันหลายวิธีได้แก่ วิธีการใช้หน่วยความจำร่วมกัน , วิธีใช้แฟ้มข้อมูลร่วมกัน , วิธีการใช้คลิปบอร์ด , วิธีการแลกเปลี่ยนข้อมูลแบบดีดีอี และวิธการใช้ โอเอลอี (OLE)

ผู้พัฒนาโปรแกรมประยุกต์จำเป็นจะต้องเลือกใช้วิธีการถ่ายโอนข้อมูลที่กล่าวมาข้างต้นนี้ให้เหมาะสมกับระบบงานของตน

1. การถ่ายโอนข้อมูลโดยวิธีการใช้คลิบบอร์ด

การถ่ายโอนโดยวิธีการใช้คลิบบอร์ดเป็นวิธีที่ใช้กันอย่างแพร่หลาย คลิบบอร์ดเป็นหน่วยความจำว่างที่ใช้เป็นตัวกลางในการถ่ายโอนข้อมูลระหว่างโปรแกรมประยุกต์ต่างๆ การใช้คลิบบอร์ดใช้วิธีการ ตัด (cut) หรือ คัดลอก (copy) บริเวณส่วนของข้อมูลที่ต้องการบนจอภาพ จากนั้นระบบปฏิบัติการวินโดว์จะทำให้เนื้อที่ของคลิบบอร์ดว่างลงและถ่ายโอนข้อมูลจากส่วนที่ทำการ ตัด หรือ คัดลอก ลงในคลิบบอร์ด และเมื่อผู้ใช้ทำงานอยู่บนโปรแกรมประยุกต์อีกโปรแกรมหนึ่งผู้ใช้ก็สามารถถ่ายโอนข้อมูลจากคลิบบอร์ดมายังหน้าจอของโปรแกรมประยุกต์นั้นๆ ได้ โดยวิธีการจัดวาง (paste) ข้อมูลลงบนหน้าจอของโปรแกรมประยุกต์ จากนั้นข้อมูลจากคลิบบอร์ดจะถูกคัดลอกลงบนหน้าจอของโปรแกรมประยุกต์ ซึ่งตัวข้อมูลต้นแบบเองยังคงอยู่ในคลิบบอร์ด โปรแกรมประยุกต์สามารถทำการแก้ไขตัวข้อมูลโดยการแก้ไขจะไม่เกี่ยวข้องกับข้อมูลต้นแบบ ข้อมูลต้นแบบจะไม่ถูกแก้ตามไปด้วย ข้อมูลต้นแบบในคลิบบอร์ดจะถูกลบทิ้งไปเมื่อมีการถ่ายโอนข้อมูลลงในคลิบบอร์ดครั้งใหม่ ซึ่งทำโดยการ ตัด หรือ คัดลอก

คลิบบอร์ดจะจัดเก็บข้อมูลที่รับมาให้อยู่ในหลายๆ รูปแบบเพื่อพร้อมที่จะอำนวยความสะดวกให้โปรแกรมประยุกต์ต่างๆ ถ่ายโอนรูปแบบข้อมูลที่เหมาะสมไปใช้งาน การทำงานของคลิบบอร์ดสามารถแสดงได้ดังรูปที่ 2.3



รูปที่ 2.3 แสดงการถ่ายโอนข้อมูลระหว่างโปรแกรมประยุกต์กับคลิบบอร์ด

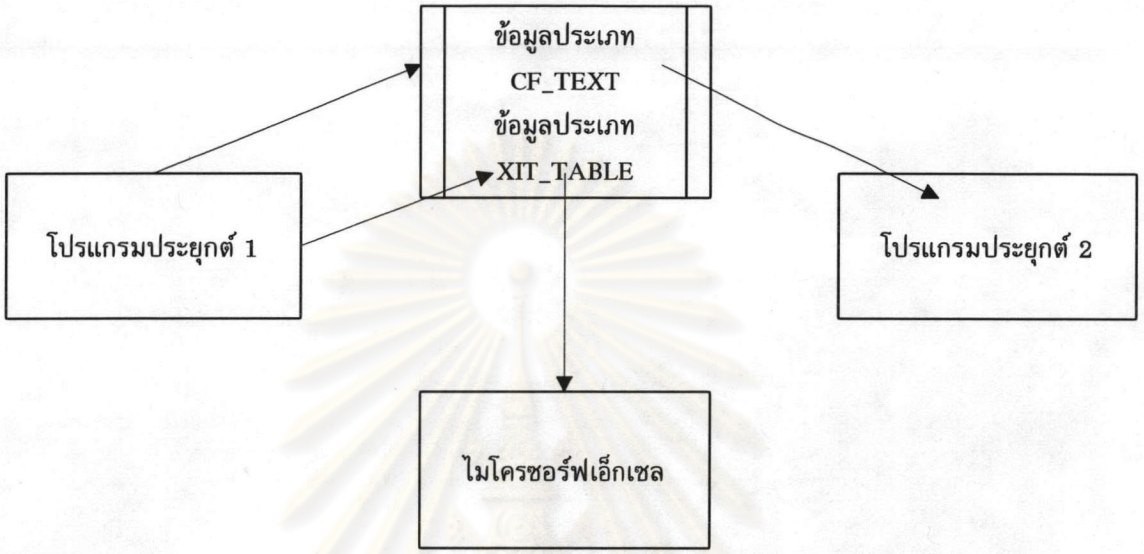
รูปแบบของข้อมูลที่คลิบบอร์ดทำการจัดเก็บสามารถแสดงได้ดังตารางที่ 2.1

ชื่อรูปแบบข้อมูล	ความหมายของรูปแบบ
CF_TEXT	ปิดท้ายด้วย NULL เป็นอักขระ แอนซีที่เก็บอยู่บนหน่วยความจำที่เรียกใช้ได้จากทุกโปรแกรม
CF_BITMAP	เป็นข้อมูลประเภทแผนที่บิต
CF_METAFILEPICT	เป็นเมทาไฟล์ (metafile) และข้อมูลเพิ่มเติมบางส่วนเกี่ยวกับวิธีการแสดงผลเมทาไฟล์
CF_SYLK	รูปแบบอักขระแอสกีที่ใช้โดยผลิตภัณฑ์เก่าๆ บางชิ้นของไมโครซอร์ฟ
CF_DIF	ข้อมูลประเภทกราฟฟิก รูปแบบ DIF ซึ่งย่อมาจาก Data Interchange Format
CF_TIFF	ข้อมูลประเภทกราฟฟิก รูปแบบ TIFF ซึ่งย่อมาจาก Tag Image File Format
CF_OEMTEXT	ข้อมูลอักขระที่ใช้รหัส OEM
CF_DIB	เป็นข้อมูลประเภท DIB ที่มี BITMAPINFO อยู่นำหน้าข้อมูล
CF_PALETTE	เป็นข้อมูลที่เกี่ยวกับรายละเอียดของสี ใช้ร่วมกับข้อมูลประเภท DIB
CF_PENDATA	ข้อมูลประเภทเกี่ยวกับปากกาแสง
CF_RIFF	ข้อมูลประเภทกราฟฟิก รูปแบบ RIFF ซึ่งย่อมาจาก Resource Interchange File Format
CD_WAVE	ข้อมูลประเภทกราฟฟิก รูปแบบคลื่น

ตารางที่ 2.1 แสดงรูปแบบของข้อมูลที่คลิปบอร์ดจัดเก็บเป็นมาตรฐาน

ระบบปฏิบัติการวินโดว์เปิดโอกาสให้ผู้ใช้สามารถกำหนดประเภทของข้อมูลที่ต้องการจะจัดเก็บบนคลิบบอร์ดได้ เช่นโปรแกรม เอ็กเซล (Excel) สามารถระบุประเภทข้อมูลที่ตนเองต้องการ คือข้อมูลรูปแบบ เอ็กไอที (XIT) เพื่อจัดเก็บในคลิบบอร์ด และนำไปใช้ในตัวโปรแกรมเองได้ แต่โดยทั่วไปแล้วโปรแกรมประยุกต์ที่ไม่ได้ระบุประเภทของข้อมูลที่ต้องการจะจัดเก็บในคลิบบอร์ด ก็สามารถใช้ข้อมูลประเภท CF_TEXT เป็นตัวกลางในการถ่ายโอนข้อมูลกับ โปรแกรมประยุกต์อื่นได้ จากรูปที่ 2.4 โปรแกรม เอ็กเซล ได้ระบุให้คลิบบอร์ดจัดเก็บข้อมูลรูปแบบเอ็กไอที

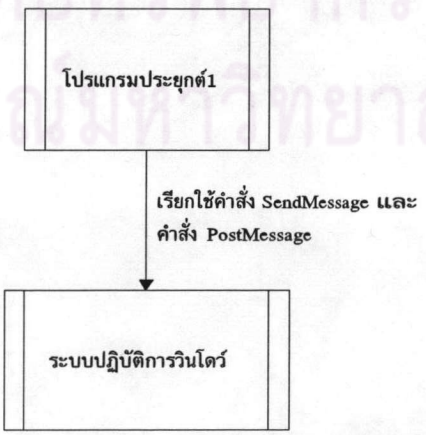
ดังนั้น เอ็กเซล จึงถ่ายโอนข้อมูลระหว่างตัวโปรแกรมเองกับคลิบบอร์ดโดยใช้รูปแบบเอ็กไอที แต่โปรแกรมประยุกต์2 ถ่ายโอนข้อมูลจากคลิบบอร์ดโดยใช้รูปแบบ CF_TEXT



รูปที่ 2.4 แสดงถึงประเภทข้อมูลที่คลิบบอร์ดถ่ายโอนให้กับโปรแกรมประยุกต์

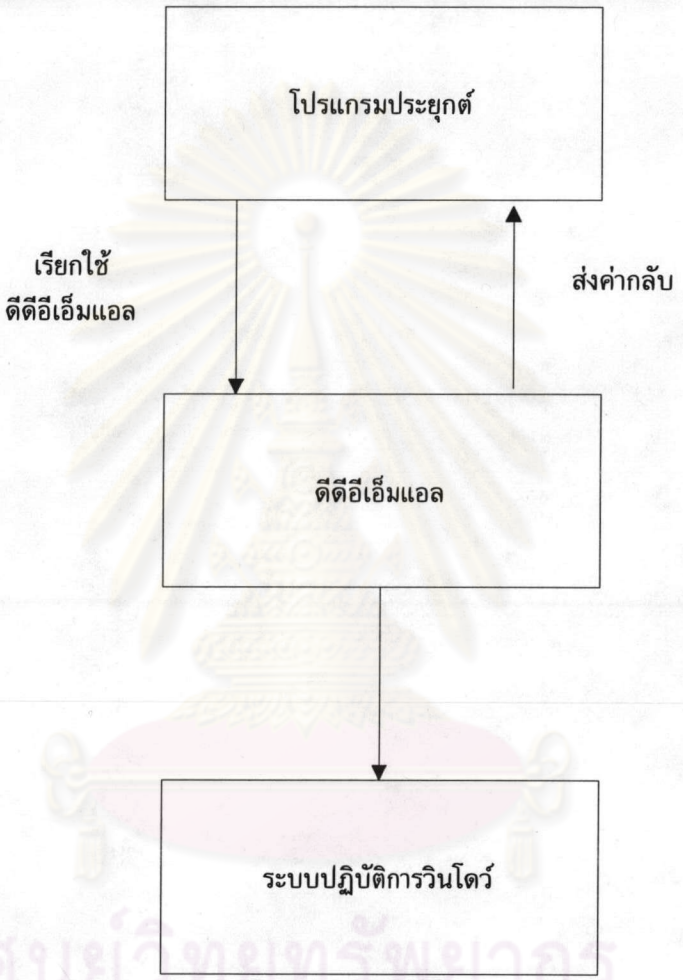
2. การถ่ายโอนข้อมูลโดย ดีดีอี

ดีดีอี เป็นการถ่ายโอนข้อมูลระหว่างโปรแกรมประยุกต์ 2 โปรแกรมในรูปแบบที่เป็นที่ตกลงกันระหว่างโปรแกรมประยุกต์ทั้งคู่นั้น โดยโปรแกรมประยุกต์ทั้งสองจะนำข้อมูลที่ได้รับการถ่ายโอนไปจัดการประมวลผลเอาเอง ในตอนเริ่มแรก ดีดีอี ถูกพัฒนาขึ้นโดยการใช้ ข่าวสารของวินโดว์ และใช้ประเภทข้อมูลกลุ่มเดียวกันกับคลิบบอร์ดดังแสดงในรูปที่ 2.5



รูปที่ 2.5 แสดงการพัฒนา ดีดีอี โดยใช้ข่าวสารของวินโดว์

ต่อมาได้มีการพัฒนา ดีดีอีเอมแอล (DDEML) ขึ้น โดยผู้พัฒนา ดีดีอี ทำการเรียกใช้ชุดคำสั่งในดีดีอีเอมแอลดังแสดงในรูปที่ 2.6 เพื่อเพิ่มขีดความสามารถของ ดีดีอี และทำให้พัฒนา ดีดีอี ได้สะดวกรวดเร็วขึ้น



รูปที่ 2.6 แสดงการพัฒนา ดีดีอี ผ่านดีดีอีเอมแอล

การพัฒนา ดีดีอี ได้ออกแบบให้มี ตัวให้บริการ ดีดีอี (DDE server) เพื่อทำการให้ข้อมูลแก่ ตัวขอบริการ ดีดีอี (DDE client) โดย ตัวขอบริการ ดีดีอี จะส่งข้อความไปเพื่อขอข้อมูลจากตัวให้บ ริการ ดีดีอี

ข้อมูลของ ดีดีอี แบ่งออกเป็น หัวเรื่องดีดีอี (DDE TOPIC) และ หัวข้อดีดีอี (DDE ITEM) โดย หัวเรื่องดีดีอี หมายถึงประเภทข้อมูลต่างๆ ที่ตัวให้บริการ ดีดีอี จัดส่งให้แก่ ตัวขอ

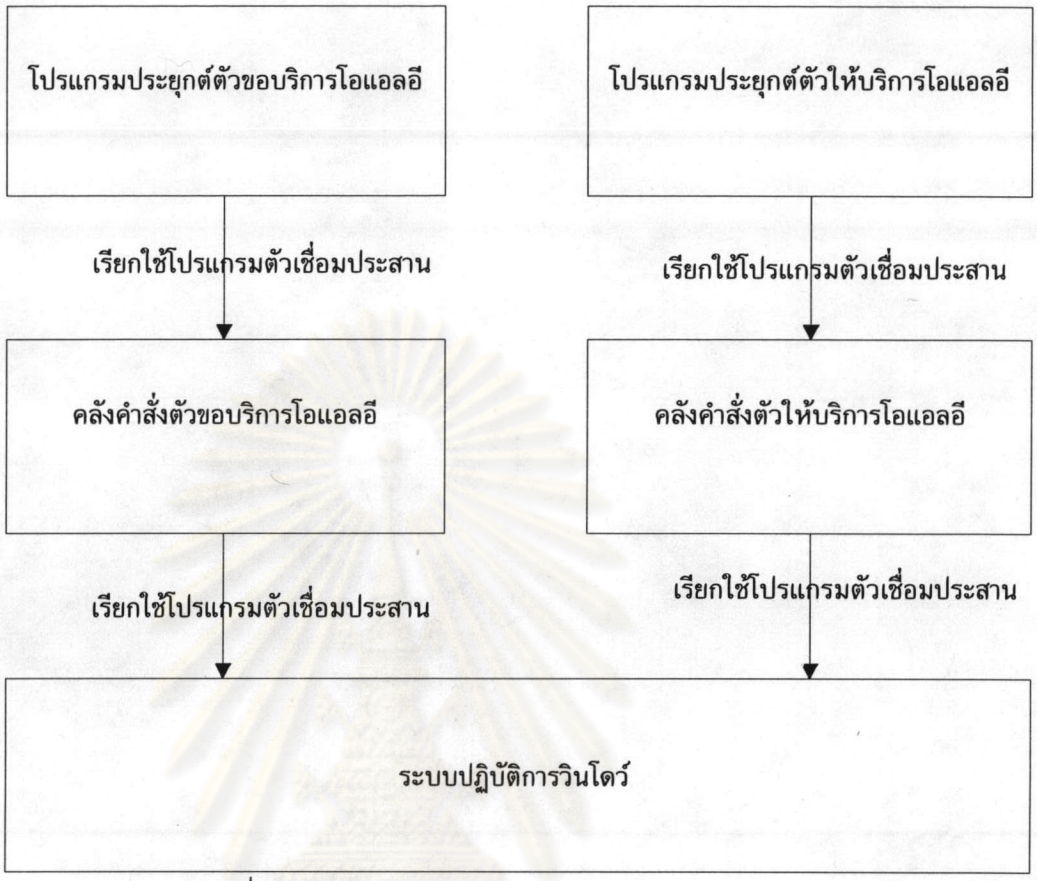
บริการ ดีดีอี เช่น ตัวให้บริการ ดีดีอี ส่งข้อมูลประเภทแฟ้มข้อมูลไปให้แก่ ตัวขอบริการ ดีดีอี ในที่นี้ หัวเรื่องดีดีอี คือ แฟ้มข้อมูลนั่นเอง ส่วน หัวข้อดีดีอี หมายถึงรายละเอียดเกี่ยวกับตัวข้อมูล ยกตัวอย่างเช่น หัวเรื่องดีดีอี ประเภทแฟ้มข้อมูล อาจจะมี หัวข้อดีดีอี ที่เป็น ขนาด ของแฟ้ม และ วัน เดือน ปี ของแฟ้มข้อมูลนั้นๆ ซึ่ง หัวข้อดีดีอี อาจจะประกอบอยู่ใน หัวเรื่องดีดีอี หลายๆ ประเภทก็ได้

ข้อมูลของ ดีดีอี ประเภทคำสั่ง คือ ข้อมูลที่ ตัวให้บริการ ดีดีอี ส่งไปให้ ตัวขอบริการ ดีดีอี เพื่อสั่งให้ ตัวขอบริการ ดีดีอี ทำงานตามคำสั่งที่ได้รับ

3. การถ่ายโอนข้อมูลโดยวิธีการใช้ โอเอลอี

โอเอลอี เป็นการถ่ายโอนข้อมูลระหว่างโปรแกรมประยุกต์ 2 โปรแกรมในลักษณะแตกต่างไปจาก ดีดีอี โดยในกรณีของ ดีดีอี นั้นโปรแกรมประยุกต์ 2 โปรแกรมจะรับข้อมูลที่ถ่ายโอนระหว่างกันและกันไปประมวลผลเอง แต่สำหรับ โอเอลอี นั้นการจัดการกับข้อมูลจะขึ้นอยู่กับโปรแกรมประยุกต์โปรแกรมใดโปรแกรมหนึ่งเพียงโปรแกรมเดียว โปรแกรมนี้จะทำหน้าที่เป็นโปรแกรมประยุกต์ให้บริการ โอเอลอี จะทำการฝัง ข้อมูล หรือ ผูก ข้อมูลที่ได้รับจากโปรแกรมให้บริการกับโปรแกรมขอบริการ เช่นการฝังหรือผูก ข้อมูลของโปรแกรม วีซีโอ (VISIO) ไว้ในโปรแกรมไมโครซอร์ฟเวิร์ด เมื่อผู้ใช้ทำงานอยู่ในโปรแกรมไมโครซอร์ฟเวิร์ดผู้ใช้ก็สามารถทำการแก้ไขข้อมูลของโปรแกรม วีซีโอ ได้ โดยการ ดับเบิลคลิกเมาส์ที่รูปภาพของโปรแกรม วีซีโอ จากนั้นโปรแกรมไมโครซอร์ฟเวิร์ด จะถ่ายโอนการทำงานไปให้โปรแกรม วีซีโอ เพื่อทำหน้าที่จัดการกับข้อมูลรูปภาพนั้น โดยโปรแกรมไมโครซอร์ฟเวิร์ดไม่ได้เข้ามาทำหน้าที่เกี่ยวข้องแต่อย่างใด ซึ่งในที่นี้โปรแกรมให้บริการคือ โปรแกรม วีซีโอ และโปรแกรมขอบริการคือโปรแกรมไมโครซอร์ฟเวิร์ด

การฝังข้อมูลของ โอเอลอี นั้น โอเอลอีจะทำการแทรกตัวข้อมูลไว้ในข้อมูลเดิมของโปรแกรมขอบริการเลย แต่การผูกข้อมูลนั้น โอเอลอี จะทำการแทรกเพียงแค่ตำแหน่ง ของข้อมูลไว้ในข้อมูลเดิมของโปรแกรมขอบริการเท่านั้นทำให้ไม่เปลืองเนื้อที่ในการเก็บข้อมูล การทำงานของ โอเอลอี สามารถแสดงได้ดังรูปที่ 2.7



รูปที่ 2.7 แสดงการทำงานของ โอเอลอี

วินโดว์ ซ็อกเก็ต

ในการเขียนโปรแกรมทางด้านเครือข่ายบนวินโดว์ จำเป็นจะต้องเขียนโปรแกรมที่เรียกผ่านตัวประสานสำหรับติดต่อกับระบบ ซึ่งได้กำหนดพื้นฐานเป็นแนวทางซึ่งเรียกว่า ซ็อกเก็ต โดยข้อกำหนดของวินโดว์ ซ็อกเก็ต หมายถึง ข้อกำหนดในการติดต่อที่เกี่ยวข้องกับการเขียนโปรแกรมทางด้านเครือข่ายบนระบบไมโครซอฟต์ วินโดว์ ซึ่งโดยมากการเขียนโปรแกรมทางด้านเครือข่ายบนวินโดว์นี้ จะต้องอยู่บนพื้นฐานของซ็อกเก็ตของ บีเอสดี (BSD) ซึ่งย่อมาจาก Berkeley Software Distribution จากมหาวิทยาลัยเบิร์กเลย์ ซึ่งได้รวมเอาส่วนสำคัญของทั้งการเขียน ชุดคำสั่งแบบในแบบซ็อกเก็ต จากเบิร์กเลย์ และส่วนขยายเพิ่มเติมของข้อกำหนดของวินโดว์ ที่ออกแบบมาเพื่อให้ผู้เขียนโปรแกรมสามารถที่จะใช้ประโยชน์ของธรรมชาติในการขับเคลื่อนด้วยข่าวสารของวินโดว์ได้

ตัวข้อกำหนดของวินโดว์ซ็อกเก็ตนี้ จะมีส่วนของตัวประสานระหว่างโปรแกรมประยุกต์รูปแบบเดียว ที่จะช่วยให้ผู้พัฒนาโปรแกรมเขียนโปรแกรมให้สามารถใช้งานได้กับเครือข่ายหลายประเภท การเรียกใช้คลังชุดคำสั่งเหล่านี้ ทำให้โปรแกรมที่ถูกเขียนให้เข้ากับด้วยตัวประสานงานโปรแกรมประยุกต์ วินโดว์ ซ็อกเก็ต สามารถทำงานร่วมกับพิธีการที่หลากหลายได้ด้วยเช่นกัน

โปรแกรมเครือข่ายที่ เป็นไปตามข้อกำหนดของวินโดว์ซ็อกเก็ต และโปรแกรมที่สามารถทำงานร่วมกับตัวประสานวินโดว์ ซ็อกเก็ต จะเรียกว่าโปรแกรมประยุกต์วินโดว์ซ็อกเก็ต

1. เบิร์กเลย์ ซ็อกเก็ต

ข้อกำหนดของ วินโดว์ซ็อกเก็ต ถูกพัฒนาขึ้นมาจากแบบการเขียนโปรแกรมของทางเบิร์กเลย์ ซ็อกเก็ต ซึ่งใช้เป็นมาตรฐานที่เป็นเครือข่ายแบบทีซีพีไอพีและบางส่วนได้ถูกออกแบบมาจากมหาวิทยาลัยแคลิฟอร์เนีย

2. ไมโครซอฟท์วินโดว์และส่วนขยายเพิ่มเติมของข้อกำหนดของวินโดว์

ตัวประสานระหว่างโปรแกรมประยุกต์ของข้อกำหนดของวินโดว์ซ็อกเก็ต นั้นมีการทำงานที่เป็นแบบทั้ง 16 บิตและ 32 บิต ซึ่งส่วนขยายเพิ่มเติมของข้อกำหนดของวินโดว์ที่ถูกรวมไว้ใน วินโดว์ ซ็อกเก็ต จะช่วยให้ผู้พัฒนาโปรแกรมประยุกต์สามารถสร้างซอฟต์แวร์ที่อยู่ในรูปแบบการเขียนโปรแกรมบนวินโดว์ ได้ซึ่งจะทำให้โปรแกรมมีประสิทธิภาพสูงสุด และการทำงานร่วมกันแบบหลายงานได้ด้วย

3. การเขียนโปรแกรมที่เกี่ยวกับซ็อกเก็ต

3.1 การตรวจสอบการติดตั้งชั้นวินโดว์ ซ็อกเก็ต

การตรวจสอบการใช้งานวินโดว์ซ็อกเก็ตในระบบ โปรแกรมประยุกต์ที่ถูกเชื่อมโยงกับชุดคำสั่งนำเข้าของวินโดว์ซ็อกเก็ต (Windows Sockets Import Library) จะมีการเรียกชุดคำสั่ง WSAStartup() หรืออีกทางหนึ่งให้พิจารณาที่ตัวแปร $PATH ซึ่งจะกำหนดไว้ให้ค้นหาและเรียกใช้ วินโดว์ ซ็อกเก็ต

3.2 หลักการพื้นฐาน

แบบที่สร้างขึ้นมาเป็นพื้นฐานสำหรับใช้เพื่อการสื่อสารเรียกว่าซ็อกเก็ต โดยตัวซ็อกเก็ตนี้จะเป็นเหมือนจุดปลายของการสื่อสารที่ถูกตั้งชื่อและจะถูกเชื่อมโยงถึงกันได้ โดยที่ซ็อกเก็ตแต่ละตัวนั้นจะมีชนิดและการประมวลผลที่เกี่ยวข้องกัน โดยทั่วไปแล้วซ็อกเก็ตนั้นจะมีอยู่ภายในส่วนของการติดต่อสื่อสาร ซึ่งส่วนการติดต่อสื่อสารนี้ จะเป็นที่รวบรวมคุณลักษณะพื้นฐานของการประมวลผลย่อยที่จะใช้สื่อสารผ่านไปยังซ็อกเก็ต

โดยปกติแล้วซ็อกเก็ตจะแลกเปลี่ยนข้อมูลเฉพาะกับซ็อกเก็ตที่อยู่ภายในขอบเขตเดียวกัน แต่อาจจะแลกเปลี่ยนข้ามขอบเขตได้ ถ้ามีการแปลงการประมวลผลบางอย่าง โดยปกติแล้วส่วนใช้สอยในวินโดว์ซ็อกเก็ตจะรองรับการทำงานที่เป็นแบบการสื่อสารเดียว นั่นก็คือโดเมนของอินเตอร์เนต ซึ่งถูกใช้งานโดยการประมวลผลที่มีการใช้ชุดพิธีการอินเตอร์เนต ในการสื่อสาร

ซ็อกเก็ตถูกแบ่งชนิดตามคุณสมบัติการสื่อสารที่ผู้ใช้สามารถเห็นได้ นั่นก็คือโปรแกรมประยุกต์จะถูกกำหนดให้สื่อสารกับซ็อกเก็ตที่อยู่ในประเภทเดียวกันเท่านั้น ถึงแม้ว่าจะไม่มีการป้องกันระบบการสื่อสารระหว่างซ็อกเก็ตที่ต่างชนิดกันก็ตาม

ซ็อกเก็ตแบ่งเป็น 2 ประเภท

3.2.1 ซ็อกเก็ตแบบสายธาร (Stream socket) มีไว้สำหรับการสื่อสารสองทาง (bi-directional) ที่เป็นลำดับโดยไม่มีข้อมูลไหลซ้ำซ้อน ที่เชื่อถือได้ โดยไม่มีขอบเขตของระเบียน

3.2.2 ซ็อกเก็ตแบบดาตาแกรม (datagram socket) จะสนับสนุนการทำงานที่เป็นแบบการสื่อสารสองทางของข้อมูล แต่ไม่รับรองเรื่อง ความน่าเชื่อถือ , ลำดับของข้อมูล , ความซ้ำซ้อนของข้อมูล นั้นคือว่าข้อความที่รับเข้ามาจากซ็อกเก็ตแบบดาตาแกรมอาจพบว่ามีข้อความที่ซ้ำซ้อน และเป็นไปได้ว่าอาจมีลำดับของข้อมูลที่แตกต่างจากต้นทางที่ส่งมา แต่ลักษณะที่สำคัญของซ็อกเก็ตแบบดาตาแกรมคือระเบียนจะมีขอบเขต การใช้งานแบบซ็อกเก็ตแบบดาตาแกรมนี้ จะมีลักษณะเหมือนการทำงานแบบเครือข่ายแบบสับหลีกกลุ่มข้อมูล

3.3 แบบจำลองผู้ใช้บริการผู้ให้บริการ

เป็นลักษณะการทำงานแบบกระจาย ซึ่งโปรแกรมประยุกต์ผู้ใช้บริการ จะต้องขอบริการจากโปรแกรมประยุกต์ผู้ให้บริการ ทั้งผู้ใช้บริการ และผู้ให้บริการจะต้องมีข้อตกลงร่วมกันระหว่างจุดปลายของการสื่อสารก่อนจึงจะสามารถทำงานร่วมกันได้ ซึ่งข้อตกลงนั้นอาจเป็นแบบสมมาตรหรือแบบไม่สมมาตรก็ได้ ถ้าเป็นพิธีการแบบสมมาตร ข้างใดข้างหนึ่งจะต้องทำหน้าที่เป็นส่วนหลักหรือไม่ก็ส่วนรอง ตัวอย่างของพิธีการแบบสมมาตรคือ พิธีการ เท็ลเน็ต ซึ่งใช้ในระบบเครือข่ายอินเตอร์เน็ตเพื่อใช้งานเครื่องปลายทางสมมติระยะไกล ตัวอย่างพิธีการที่ไม่สมมาตร คือ พิธีการถ่ายโอนแฟ้มข้อมูลบนอินเตอร์เนต อย่างไรก็ตามไม่สำคัญว่าพิธีการที่ใช้จะเป็นแบบใด แต่เมื่อมีการเข้ามาใช้งานการบริการที่มีให้ จะมีกระบวนการที่เกิดขึ้น 2 อย่างคือ การประมวลผลของผู้ใช้บริการ และการประมวลผลของผู้ให้บริการ

โปรแกรมประยุกต์ที่ทำหน้าที่เป็นผู้ให้บริการส่วนใหญ่จะรอคำขอบริการจาก ผู้ใช้บริการที่ตัวเองรู้จัก และเมื่อมีการขอบริการเข้ามาผู้ให้บริการก็จะทำงานตามที่ผู้ใช้บริการนั้นขอเข้ามา โดยที่การทำงานบางอย่างอาจมีการใช้งานซ็อกเก็ตแบบดาตาแกรมด้วย

3.4 การกระจาย

โดยการใช้งานซ็อกเก็ตแบบดาตาแกรมจะทำให้สามารถส่งกลุ่มข้อมูลที่ถูกกระจาย ไปยังหลายๆ เครือข่ายได้ ซึ่งเครือข่ายนั้นจะต้องรองรับการทำงานแบบกระจายด้วย โดยที่ข้อความที่ถูกกระจาย นั้นจะบังคับให้ ทุกๆ แม่ข่ายบนเครือข่ายให้บริการ ถ้าไม่มีการจำลองการกระจายด้วยซอฟท์แวร์จะทำให้ข่ายงานต้องรับภาระงานหนัก ดังนั้นความสามารถของการส่งกลุ่มข้อมูลที่ถูกกระจายจะถูกจำกัดด้วยซ็อกเก็ตที่จะสามารถทำงานแบบการกระจายได้ โดยปกติแล้วการกระจายจะถูกใช้ด้วยวัตถุประสงค์อย่างใดอย่างหนึ่งคือ ต้องการที่จะหาทรัพยากรในเครือข่ายท้องถิ่น โดยที่ไม่รู้ตำแหน่งของทรัพยากรนั้นมาก่อน หรือไม่ก็ทำหน้าที่การหาเส้นทางที่ต้องการ โดยที่สารสนเทศนั้นจะถูกส่งไปยังตำแหน่งทั้งหมดที่สามารถติดต่อได้ ตำแหน่งปลายทางของข้อความที่ถูกกระจายไปนั้น ขึ้นกับเครือข่ายที่ข้อความนั้นถูกกระจายออกไป โดเมนของอินเตอร์เนตจะรองรับการทำงานที่เป็นรูปแบบระยะสั้น ที่ใช้สื่อสารติดต่อกันในเครือข่ายท้องถิ่น โดยอ้างอิงตำแหน่งผ่านค่าตัวแปร INADDR_BROADCAST ข้อความกระจายที่ได้รับนั้นจะประกอบด้วย ตำแหน่งของผู้ส่ง และพอร์ท (port) ในขณะที่ซ็อกเก็ตแบบดาตาแกรมต้องถูก เชื่อมโยง ก่อนจะใช้งาน

3.5 การเรียงลำดับไบท์

การอ้างอิงถึงที่อยู่แบบไอพี และเลขพอร์ท ถูกส่งออกไปหรือรับเข้ามาจากชุดคำสั่งวินโดว์ซ็อกเก็ต ซึ่งจะต้องถูกเรียกตามลำดับของเครือข่าย ซึ่งได้รวมที่อยู่แบบไอพีและเขตของพอร์ทของตัวแปรชื่อ socketaddr_in เข้าไปด้วย

เมื่อพิจารณาถึงโปรแกรมประยุกต์ที่ทำการติดต่อไปยังผู้ให้บริการบนพอร์ททีซีพีที่ทำงานร่วมกันกับการให้บริการเวลา ในการทำงานนี้จะมีกลไกสำหรับผู้ใช้ให้สามารถกำหนดพอร์ทที่จะถูกเลือกใช้งานได้ ซึ่งจะเห็นว่าเลขพอร์ทที่ถูกคืนค่ามาจาก ชุดคำสั่ง getservbyname() นั้นจะถูกจัดเรียงให้เป็นไปตามลำดับของเครือข่าย ซึ่งรูปแบบที่ต้องการจะต้องอยู่ในรูปของที่อยู่ซึ่งจะทำให้ไม่ต้องมาแปลค่าอีก อย่างไรก็ตามถ้าผู้ใช้เลือกที่จะใช้พอร์ทที่แตกต่างออกไป โดยการใส่เลขจำนวนเต็ม โปรแกรมประยุกต์ที่ใช้งานจะต้องทำการเปลี่ยนค่าพอร์ทนั้นจากแม่ข่ายไปยังลำดับของเครือข่าย โดยการใช้ชุดคำสั่ง htons() ก่อนที่จะนำค่านั้นไปสร้างเป็นที่อยู่ในทางกลับกันถ้าโปรแกรมประยุกต์ต้องการที่จะแสดงค่าจำนวนของพอร์ทภายในที่อยู่ ซึ่งเป็นค่าที่คืนมาจากชุดคำสั่ง getpeername() เลขพอร์ทนั้นจะต้องถูกเปลี่ยนค่าจากเครือข่ายไปเป็นลำดับของแม่ข่ายโดยการใช้ชุดคำสั่ง ntohs() ก่อนที่จะสามารถนำค่านั้นไปแสดงได้

3.6 ตัวเลือกของซ็อกเก็ต

ค่าโดยปริยายที่เป็นทางเลือกเมื่อมีการเรียกใช้ชุดคำสั่ง setsockopt() แสดงได้ดังนี้

ชื่อตัวแปร	ประเภท	ความหมาย	ค่ามาตรฐาน
SO_ACCEPTCONN	BOOL	ซ็อกเก็ตกำลังใช้ชุดคำสั่ง listen()	FALSE unless a listen() has been performed
SO_BROADCAST	BOOL	ซ็อกเก็ตถูกกำหนดให้ใช้สำหรับส่งข้อมูลที่ถูกกระจาย	FALSE
SO_DEBUG	BOOL	สามารถทำการการตรวจสอบได้	FALSE
SO_DONTLINGER	BOOL	ถ้ามีค่าเป็น TRUE ตัวเลือกที่ชื่อ SO_LINGER จะถูกยกเลิก	TRUE
SO_DONTROUTE	BOOL	การหาเส้นทางถูกยกเลิก	FALSE
SO_ERROR	int	อ่านสถานะของความผิดพลาดและทำให้เป็นค่า 0	0
SO_KEEPALIVE	BOOL	กำลังส่งสัญาณเพื่อบอกว่ายังทำงานอยู่	FALSE
SO_LINGER	struct linger FAR	ส่งค่าของตัวเลือกที่ชื่อ linger	*l_onoff* is 0
SO_OOBINLINE	BOOL	ข้อมูลอื่นกำลังถูกพบในสายธารข้อมูลปกติ	FALSE
SO_RCVBUF	int	ขนาดหน่วยความจำชั่วคราวสำหรับรับข้อมูล	Implementation dependent

ตารางที่ 2.2 ตัวเลือกของซ็อกเก็ต

ชื่อตัวแปร	ประเภท	ความหมาย	ค่ามาตรฐาน
SO_REUSEADDR	BOOL	ตำแหน่งที่ซ็อกเก็ตใช้นั้นส่วนอื่นๆ สามารถนำไปใช้ได้	FALSE
SO_SNDBUF	int	ขนาดของหน่วยความจำชั่วคราวเพื่อส่งข้อมูล	Implementation dependent
SO_TYPE	int	ประเภทของซ็อกเก็ต เช่น SOCK_STREAM	As created via socket()
TCP_NODELAY	BOOL	ยกเลิกการใช้ วิธีการนาเกิล (Nagle)	Implementation dependent

ตารางที่ 2.2 ตัวเลือกของซ็อกเก็ต (ต่อ)

3.7 แฟ้มฐานข้อมูล

ชุดคำสั่ง getXbyY() และ ชุดคำสั่ง WSAAsyncGetXByY() มีไว้สำหรับค้นหาเครือข่ายที่ระบุไว้ในสารสนเทศ ชุดของชุดคำสั่ง getXbyY() โดยเริ่มแรกที่มากับยูนิกซ์ของเบิร์กเลย์นั้นถูกออกแบบมาเพื่อเป็นเครื่องมือที่ใช้ค้นหาสารสนเทศในฐานข้อมูลตัวอักขระ (text databases) ซึ่งจริงๆ แล้วสารสนเทศเหล่านั้นสามารถใช้วินโดว์ซ็อกเก็ต ที่สร้างขึ้น ดึงสารสนเทศออกมาได้ในหลายวิธี แต่กระนั้นก็ตามโปรแกรมประยุกต์ที่ใช้วินโดว์ซ็อกเก็ตก็ยังจำเป็นที่จะต้องเรียกใช้สารสนเทศเหล่านั้นผ่านทางชุดคำสั่ง getXbyY() หรือ ชุดคำสั่ง WSAAsyncGetXByY()

3.7.1 ชนิดข้อมูลของซ็อกเก็ต และค่าความผิดพลาด

ได้มีการกำหนดชนิดข้อมูลตัวใหม่ขึ้นมาคือ ซ็อกเก็ต ซึ่งการกำหนดชนิดข้อมูลนี้จะเป็นส่วนจำเป็นในการปรับปรุงประสิทธิภาพในอนาคตของข้อกำหนดวินโดว์ ซ็อกเก็ต เป็นต้นว่า สามารถใช้วินโดว์ ซ็อกเก็ตเป็นตัวจัดการแฟ้มข้องมูล ได้ในวินโดว์ เอ็นที และยังจะเป็นประโยชน์ในการนำไปใช้งานกับโปรแกรมประยุกต์ที่เป็นแบบวิน/32 ได้อีกด้วย

บนระบบยูนิกซ์การจัดการทั้งหมด รวมถึงการจัดการทางด้านซ็อกเก็ตจะใช้ค่าจำนวนเต็มบวกน้อยๆ ซึ่งจะให้ค่าเป็นจริง แต่การจัดการโดยใช้ วินโดว์ ซ็อกเก็ต นั้นไม่มีข้อจำกัด ถึงแม้ว่าค่าที่คืนกลับมาจากตัวแปร INVALID_SOCKET จะเป็นค่าที่ใช้ไม่ได้ใน ซ็อกเก็ตก็ตามเพราะว่าค่าที่ใช้ในการจัดการซ็อกเก็ตนั้นสามารถที่จะใช้ ที่อยู่ ในช่วง0ถึง INVALID_SOCKET-1 ได้

เนื่องจากว่าชนิดข้อมูลของซ็อกเก็ตนั้นเป็นแบบไม่คิดเครื่องหมาย การที่จะแปลชุดคำสั่งรหัสต้นฉบับที่มีอยู่บนสภาพแวดล้อมแบบยูนิกซ์อาจจะมีข้อความเตือนจากตัวแปลชุดคำสั่งเกี่ยวกับเรื่องชนิดข้อมูลแบบคิดเครื่องหมายและไม่คิดเครื่องหมายที่ไม่เข้ากัน

นั่นหมายความว่าการที่จะตรวจสอบค่าที่คืนมาจากการเรียกใช้ชุดคำสั่ง socket() และ ชุดคำสั่ง accept() ไม่ควรที่จะใช้วิธีการเปรียบเทียบค่าที่คืนมากับค่า -1 หรือค่าจำนวนลบตัวอื่นๆ แต่ให้ทำโดยใช้ค่าของตัวแปร INVALID_SOCKET ที่กำหนดไว้ในแฟ้ม winsock.h ดังตัวอย่างต่อไปนี้

```
TYPICAL BSD STYLE:
        s = socket(...);
        if (s == -1)        /* or s < 0 */
          {...}

PREFERRED STYLE:
        s = socket(...);
        if (s == INVALID_SOCKET)
          {...}
```

3.7.2 ชุดคำสั่ง select()

เนื่องจากว่าซ็อกเก็ตไม่ได้ถูกใช้แทนค่าด้วยจำนวนเต็นบวกน้อยๆ ในยูนิกซ์ การสร้างชุดคำสั่ง socket() ในตัวประสานงานระหว่างโปรแกรมประยุกต์บนวินโดว์ ซ็อกเก็ตได้เปลี่ยนรูปแบบไป คือ แต่ละชุดของซ็อกเก็ตจะถูกแทนค่าด้วยชุดของ ตัวแปร fd_set แต่การแทนค่านี้จะถูกเก็บไว้ ในรูปของหน้ากากบิต ซึ่งถูกกำหนดให้เป็นเหมือนแถวลำดับของซ็อกเก็ต เพื่อหลีกเหลี่ยงปัญญหาที่อาจเกิดขึ้น โปรแกรมประยุกต์ที่เขียนขึ้นจะต้องใช้ชุดคำสั่งแมคโคร เพื่อที่จะตั้งค่า กำหนดค่าเริ่มต้น ลบค่าและตรวจสอบค่าของตัวแปร fd_set

3.7.3 รหัสระบุความผิดพลาด

ชุดของรหัสระบุความผิดพลาดที่ถูกกำหนดโดย วินโดว์ ซ็อกเก็ต จะไม่มีอยู่ใน ตัวแปรความผิดพลาด และสำหรับชุดของชุดคำสั่ง getXbyY() ชุดของรหัสระบุความผิดพลาดก็ไม่มีอยู่ในตัวแปร h_errno แต่รหัสระบุความผิดพลาดที่เกิดขึ้นสามารถเข้าถึงได้ด้วยการใช้ตัวประสานงานระหว่างโปรแกรมประยุต์ ผ่านชุดคำสั่ง WSAGetLastError() ซึ่งชุดคำสั่งนี้ที่มีอยู่

ใน วินโดว์ ซ็อกเก็ต จะเหมือนกับชุดคำสั่ง GetLastError() ซึ่งชุดคำสั่งนี้มีไว้สำหรับการประมวลผลย่อยในการทำงานที่เป็นแบบหลายการประมวลผลย่อย ใช้เพื่อสามารถจะหาข้อมูลค่าข้อผิดพลาดที่เกิดขึ้นในแต่ละการประมวลผลย่อยออกมาได้ เพื่อที่ จะสามารถใช้งานร่วมกับบีเอสดีได้ โปรแกรมที่เขียนขึ้นจะต้องมีบรรทัดต่อไปนี้

```
#define errno WSAGetLastError()
```

ซึ่งจะทำให้รหัสเครือข่ายที่เขียนขึ้นสามารถใช้งานกับตัวแปร errno ที่ใช้ระบุรหัสความผิดพลาดได้อย่างถูกต้องเมื่อทำงานอยู่ในการประมวลผลย่อยแบบเชิงเดี่ยว แต่กระนั้นก็ตามก็มีข้อเสียที่เห็นได้ชัดคือ ถ้าแฟ้มข้อมูลต้นฉบับ รวมเอารหัส ที่ใช้ตรวจสอบตัวแปร errno โดยใช้ชุดคำสั่งที่เป็นทั้งแบบใช้ซ็อกเก็ตและไม่ใช้ซ็อกเก็ตร่วมกัน การทำงานแบบข้างต้นจะไม่สามารถใช้ได้ นอกจากนี้ การกำหนดค่าใหม่ให้กับตัวแปร errno นั้นไม่สามารถทำได้โดยการกำหนดไว้ในโปรแกรมประยุกต์ที่เขียนขึ้น

```
   TYPICAL BSD STYLE:
r = recv(...);
if (r == -1
   && errno == EWOULDBLOCK)
   {...}

   PREFERRED STYLE:
r = recv(...);
if (r == -1     /* (but see below) */
   && WSAGetLastError() == EWOULDBLOCK)
   {...}
```

ถึงแม้ว่าค่าคงที่ของข้อผิดพลาดที่มีมาให้กับซ็อกเก็ตของมหาวิทยาลัยเบิร์กเลย์เวอร์ชั่น 4.3 จะสามารถทำงานเข้ากันได้แต่โปรแกรมประยุกต์ที่พัฒนาขึ้นควรใช้ค่ารหัสระบุความผิดพลาด ที่มีมาให้ โดยควรเขียนรหัสต้นฉบับที่ถูกต้องดังนี้

```
r = recv(...);
if (r == -1      /* (but see below) */
   && WSAGetLastError() == WSAEWOULDBLOCK)
        {...}
```

3.7.4 ตัวชี้

ตัวชี้ที่ถูกใช้โดยโปรแกรมที่เขียนด้วย วินโดว์ ซ็อกเก็ต ควรจะเป็น FAR ซึ่งสามารถทำงานได้อย่างดีเมื่อใช้ชนิดข้อมูลแบบ LPHOSTENT

3.7.5 ชุดคำสั่งที่ถูกเปลี่ยนชื่อ

มีการเปลี่ยนชื่อชุดคำสั่งที่ถูกใช้งานในเบิร์กเลย์ ซ็อกเก็ต เพื่อที่จะหลีกเหลี่ยง การชนกันกับตัวประสานงานโปรแกรมตัวอื่นๆ แบ่งได้เป็น 2 กลุ่มคือ

3.7.5.1 ชุดคำสั่ง close() และชุดคำสั่ง closesocket()

การทำงานกับซ็อกเก็ตไม่สามารถที่จะใช้ชุดคำสั่งที่ใช้กับแฟ้มข้อมูลปกติ เช่นการอ่าน การเขียน และการปิดแฟ้ม ได้ เพราะอาจทำให้การทำงานกับ ซ็อกเก็ต เกิดข้อผิดพลาดได้ ดังนั้นซ็อกเก็ตจึงจำเป็นต้องถูกปิดโดยการใช้ชุดคำสั่ง closesocket()

3.7.5.2 ชุดคำสั่ง ioctl() และ ชุดคำสั่ง ioctlsocket()

ชุดคำสั่ง ioctlsocket() ถูกใช้เพื่อจัดการการทำงานของซ็อกเก็ต แทนที่จะใช้ชุดคำสั่ง ioctl() และ fcnntl()

3.7.6 จำนวนค่าสูงสุดของซ็อกเก็ตที่สามารถใช้งานได้

จำนวนค่าสูงสุดของซ็อกเก็ตที่สามารถใช้งานได้นั้นขึ้นอยู่กับว่าเลือกใช้วินโดว์ ซ็อกเก็ตของผู้ผลิตรายไหน ซึ่งจำนวนซ็อกเก็ตสูงสุดที่โปรแกรมประยุกต์สามารถใช้งานได้นั้นจะถูกกำหนดตอนช่วงเวลาแปลชุดคำสั่ง

ค่ามาตรฐานในแฟ้ม winsock.h คือ 64 แต่ถ้าโปรแกรมถูกเขียนขึ้นมานั้นสามารถทำงานกับซ็อกเก็ตได้มากกว่า 64 ตัวผู้เขียนโปรแกรมควรกำหนดค่าที่มากกว่านั้นด้วยชุดคำสั่งย่อย FD_SETSIZE ในทุกๆ แฟ้มต้นฉบับ ก่อนที่จะมีการเรียกใช้งาน แฟ้ม winsock.h หรือกำหนดโดยเพิ่มตัวแปรกำหนดค่า DFD_SETSIZE=128 ในเม็กไฟล์ (makefile) สำหรับไมโครซอฟท์ ซี ซึ่งจากตรงนี้สามารถสรุปได้ว่าการกำหนดค่าโดยชุดคำสั่งย่อย FD_SETSIZE นั้นจะไม่มีผลกระทบต่อค่าของซ็อกเก็ตที่มีมาให้ในชุดของวินโดว์

3.7.7 แฟ้มข้อมูลที่ใช้

เพื่อความสะดวกในการที่จะสามารถใช้งานกับรหัสต้นฉบับบนซ็อกเก็ตของมหาวิทยาลัยเบิร์กเลย์ได้โปรแกรมประยุกต์ที่ทำงานกับวินโดว์ ซ็อกเก็ต จำเป็นที่จะต้องเรียกใช้งาน แฟ้ม winsock.h ไว้ในรหัสต้นฉบับ

3.7.8 ค่าที่คืนกลับมาเมื่อเกิดข้อผิดพลาดจากตัวประสานงานโปรแกรมประยุกต์

ค่าคงที่ที่ใช้ในการตรวจสอบการทำงานที่ผิดพลาดของตัวประสานงานโปรแกรมประยุกต์ คือ SOCKET_ERROR ตัวอย่างการใช้ค่าคงที่นี้

```
TYPICAL BSD STYLE:
        r = recv(...);
        if (r == -1     /* or r < 0 */
            && errno == EWOULDBLOCK)
            {...}
PREFERRED STYLE:
        r = recv(...);
        if (r == SOCKET_ERROR
            && WSAGetLastError() ==
WSAEWOULDBLOCK)
            {...}
```

3.7.9 ซ็อกเก็ตดิบ

ข้อกำหนดของวินโดว์ ซ็อกเก็ตไม่ได้กำหนดให้ วินโดว์ ซ็อกเก็ต ดีแอลแอล (DLL) รองรับการทำงานของซ็อกเก็ตดิบ นั้นคือว่าซ็อกเก็ตจะต้องใช้งานกับค่า SOCK_RAW และการใช้งานกับซ็อกเก็ตดิบนั้นจะต้องทำการเปิดซ็อกเก็ตด้วยชุดคำสั่ง socket()

3.8 วินโดว์ ซ็อกเก็ตในการทำงานของวินโดว์แบบหลายการประมวลผลย่อย

ตัวประสานงานของวินโดว์ ซ็อกเก็ตได้ถูกออกแบบขึ้นให้ทำงานได้ทั้งบนวินโดว์ที่ทำงานแบบการประมวลผลย่อยเดี่ยว เช่น วินโดว์ 3.1 และวินโดว์ที่ทำงานแบบหลายการประมวลผลย่อย เช่น วินโดว์เอ็นที ในการทำงานที่เป็นแบบหลายการประมวลผลย่อยการที่จะทำให้ซ็อกเก็ตที่อยู่ระหว่างการประมวลผลย่อยสามารถเรียกใช้งานให้เข้าจังหวะกันได้ นั้นจะต้อง

เป็นหน้าที่ของโปรแกรมประยุกต์ ไม่ใช่ของวินโดว์ ซ็อกเก็ต ดังนั้นความผิดพลาดที่เกิดจากการการเรียกใช้ซ็อกเก็ตให้ทำงานเข้าจังหวะกันได้อาจเกิดผลที่ไม่คาดคิดได้ เช่นมีการเรียกใช้ชุดคำสั่ง send() พร้อมกัน 2 จุด จะไม่มีการรับประกันลำดับของข้อมูลที่ถูกส่งออกไปว่าถูกต้องหรือไม่

การปิดซ็อกเก็ตในการประมวลผลย่อยตัวใดตัวหนึ่งที่มีการเรียกใช้แบบหยุดรอไปยังซ็อกเก็ตเดียวกันในอีกการประมวลผลย่อยจะส่งผลให้การเรียกแบบหยุดรอนั้นเกิดข้อผิดพลาดด้วยค่าตัวแปร WSAEINTR ทำให้การทำงานนั้นถูกยกเลิก ซึ่งข้อผิดพลาดนี้จะเหมือนกับการเรียกใช้ชุดคำสั่ง select() อยู่แล้วมีการปิดซ็อกเก็ตใดซ็อกเก็ต หนึ่งที่เรียกใช้งาน ชุดคำสั่ง select()

4. ชุดคำสั่งที่เกี่ยวกับซ็อคเก็ต

4.1 ชุดคำสั่งที่ทำงานเกี่ยวกับซ็อกเก็ต

ข้อกำหนดของวินโดว์ ซ็อกเก็ตได้รวมเอาชุดคำสั่งของซ็อกเก็ต ที่เหมือนกับของทางมหาวิทยาลัยเบิร์กเลย์เข้ามาไว้ด้วย ซึ่งสามารถแสดงได้ดังตารางที่ 2.3

accept()	การติดต่อที่กำลังจะเกิดขึ้นถูกรับรู้และเกี่ยวเนื่องกับซ็อกเก็ตที่ถูกสร้างขึ้น ซ็อกเก็ตเดิมจะส่งค่าสถานะว่ากำลังรอสัญาณ
bind()	กำหนดชื่อให้กับซ็อกเก็ต
closesocket()	ยกเลิกซ็อกเก็ตจากตาราง
connect()	เริ่มต้นการติดต่อกับซ็อกเก็ตที่ระบุ
getpeername()	อ่านชื่อของฝั่งที่ติดต่อกับซ็อกเก็ตที่ระบุ
getsockname()	อ่านชื่อจากซ็อกเก็ตที่ระบุ
getsockopt()	อ่านค่าตัวเลือกจากซ็อกเก็ตที่ระบุ
htonl()	แปลงข้อมูลประเภท 32 บิตจากการเรียงลำดับของเครื่องคอมพิวเตอร์หลักเป็นการเรียงลำดับของระบบเครือข่าย

ตารางที่ 2.3 ชุดคำสั่งของซ็อกเก็ต

htons()	แปลงข้อมูลประเภท 12 บิตจากการเรียงลำดับของเครื่องคอมพิวเตอร์หลักเป็นการเรียงลำดับของระบบเครือข่าย
inet_addr()	แปลงที่อยู่ของระบบเครือข่ายจากประเภทอักขระให้เป็นตำแหน่งจริง
inet_ntoa()	แปลงที่อยู่ของระบบเครือข่ายจากตำแหน่งจริงให้เป็นที่อยู่ประเภทอักขระ
ioctlsocket()	กำหนดการควบคุมซ็อกเก็ต
listen()	รอรับสัญญาณการติดต่อที่กำลังจะเข้ามาจากซ็อกเก็ตที่ระบุ
ntohl()	แปลงข้อมูลประเภท 32 บิตจากประเภทการเรียงลำดับแบบระบบเครือข่ายเป็นการเรียงลำดับแบบเครื่องคอมพิวเตอร์หลัก
ntohs()	แปลงข้อมูลประเภท 16 บิตจากประเภทการเรียงลำดับแบบระบบเครือข่ายเป็นการเรียงลำดับแบบเครื่องคอมพิวเตอร์หลัก
recv()	รับข้อมูลจากซ็อกเก็ตที่ติดต่ออยู่
recvfrom()	รับข้อมูลจากซ็อกเก็ตใดๆ
select()	จัดสร้างการทำงานของการรับและการส่งแบบหลายทางให้สอดคล้องกัน
send()	ส่งข้อมูลไปยังซ็อกเก็ตที่ติดต่ออยู่
sendto()	ส่งข้อมูลไปยังซ็อกเก็ตใดๆ
setsockopt()	เก็บค่าตัวเลือกที่เกี่ยวกับซ็อกเก็ตที่ระบุ
shutdown()	ยกเลิกส่วนของการติดต่อแบบไปและกลับ
socket()	สร้างจุดปลายของการเชื่อมต่อและส่งซ็อกเก็ตกลับมา

ตารางที่ 2.3 ชุดคำสั่งของซ็อกเก็ต (ต่อ)

4.1.1 การหยุดรอของซ็อกเก็ต

หัวข้อหลักในการนำโปรแกรมที่พัฒนาขึ้นมาโดยใช้สภาพแวดล้อมของซ็อกเก็ตไปใช้งานในสภาพแวดล้อมของระบบปฏิบัติการวินโดว์ คือ การหยุดรอ ซึ่งเป็นการทำงานที่จะไม่มีการคืนค่าจนกว่าการทำงานนั้นจะเสร็จสมบูรณ์ ซึ่งปัญหานี้จะเกิดขึ้นเมื่อการทำงานใช้เวลานานมากกว่าจะเสร็จ เช่น ชุดคำสั่ง recv() จะหยุดรอจนกว่าจะมีการรับข้อมูลที่ส่งมาจากอีกฝั่งหนึ่ง ลักษณะแบบนี้จะเป็นลักษณะที่มีมากับ ซอกเก็ตของเบอร์เลย์ เพื่อให้ ซ็อกเก็ตทำงานในสภาวะหยุดรอ แต่ถ้าผู้เขียนโปรแกรมไม่กำหนดไว้โดยตรงว่าให้ทำงานเป็นแบบในสภาวะไม่หยุดรอ

แต่ทั้งนี้มีการทำงานบนซ็อกเก็ตที่กำลังหยุดรอ บางตัว เช่น ชุดคำสั่ง bind(), ชุดคำสั่ง getsockopt() และชุดคำสั่ง getpeername() สามารถเสร็จสมบูรณ์ได้ทันที ซึ่งลักษณะการทำงานจะไม่มีความแตกต่างกันระหว่างการทำงาน แบบหยุดรอ กับ แบบไม่หยุดรอ

ดังนั้นจะมีชุดคำสั่งอยู่ 2 ตัวที่จะช่วยผู้เขียนโปรแกรมจัดการเกี่ยวกับเรื่อง การหยุดรอ และ การไม่หยุดรอ คือชุดคำสั่ง WSAIsBlocking() ใช้เพื่อกำหนดว่า วินโดว์ซ็อกเก็ต กำลังทำงานอยู่หรือไม่ และ ชุดคำสั่ง WSACancelBlockingCall() ใช้เพื่อที่จะยกเลิกการทำงานของ การหยุดรอ ที่กำลังดำเนินการอยู่ และถ้าเกิดค่าตัวแปร WSAEINPROGRESS ไม่เท่ากับ 0 เมื่อเรียกใช้งานชุดคำสั่งทั้งสองนี้ แสดงว่ามีข้อผิดพลาดเกิดขึ้น แต่ถ้าเป็นการทำงานที่ซับซ้อนกว่านี้เช่นมีการใช้งาน รูปแบบเอ็มดีไอ (MDI) โปรแกรมที่เขียนขึ้นจะต้องเรียกใช้ชุดคำสั่ง WSASetBlockingHook() ในการทำงานกับ ตัวเชื่อมประสานวินโดว์ ซ็อกเก็ต

4.2 ชุดคำสั่งจัดการฐานข้อมูล

ข้อกำหนดของวินโดว์ ซ็อกเก็ต ระบุชุดคำสั่ง ไว้ดังตารางที่ 2.4

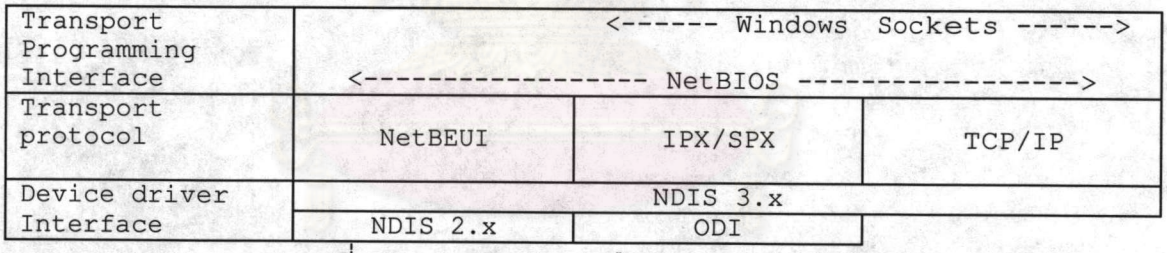
gethostbyaddr()	อ่านชื่อและตำแหน่งที่ตรงกับตำแหน่งของระบบเครือข่าย
gethostbyname()	อ่านชื่อและตำแหน่งที่ตรงกับตำแหน่งของเครื่องคอมพิวเตอร์แม่ข่าย
gethostname()	อ่านชื่อของเครื่องคอมพิวเตอร์แม่ข่าย

ตารางที่ 2.4 ชุดคำสั่งของซ็อกเก็ตประเภทจัดการฐานข้อมูล

getprotobyname()	อ่านชื่อของพิธีการและหมายเลขของพิธีการนั้นๆ จากชื่อพิธีการที่ระบุ
getprotobynumber()	อ่านชื่อของพิธีการและหมายเลขของพิธีการนั้นๆ จากหมายเลขพิธีการที่ระบุ
getservbyname()	อ่านชื่อของการให้บริการและช่องส่งสัญาณจากชื่อของการให้บริการที่กำหนด
getservbyport()	อ่านชื่อของการให้บริการและช่องส่งสัญาณจากช่องส่งสัญญาณที่กำหนด

ตารางที่ 2.4 ชุดคำสั่งของซ็อกเก็ตประเภทจัดการฐานข้อมูล (ต่อ)

สำหรับในระบบปฏิบัติการวินโดว์95 ก็มี คลังคำสั่งวินโดว์ซ็อกเก็ตเพื่ออำนวยความสะดวกให้ผู้เขียนโปรแกรมได้นำไปใช้พัฒนาโปรแกรมเกี่ยวกับเครือข่าย จากรูปที่ 2.8 แสดงให้เห็นถึงวินโดว์ซ็อกเก็ตภายใต้สถาปัตยกรรมเครือข่ายของวินโดว์ 95



รูปที่ 2.8 สถาปัตยกรรมเครือข่ายของวินโดว์ 95