สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

พีซีเทอร์มินัลเซิร์ฟเวอร์เพื่อเชื่อมอุปกรณ์ที่ต่อพอร์ตอนุกรม กับเครือข่ายอีเทอร์เน็ต

PC-BASED SERIAL TO ETHERNET TERMINAL SERVER

โดย

จิรายุ วิริยะอมรพันธุ์

ภวัต ชมภูแสง

อาจารย์ที่ปรึกษา

อาจารย์สุเมธ ประภาวัต

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เลขหมู่.....
เลขทะเบียน 73060
วัน,เดือน,ปี - 2 ก.ค. 2550

.b 11779895
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 2 ปีการศึกษา 2549

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PC-BASED SERIAL TO ETHERNET TERMINAL SERVER

JIRAYU VIRIYA-AMONPUN

PAWAT CHOMPHOOSANG

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2/2006

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

COPYRIGHT 2007

FACULTY ON INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใบรับรองปริญญานิพนธ์ ประจำปีการศึกษา 2549

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง พีซีเทอร์มินัลเซิร์ฟเวอร์เพื่อเชื่อมอุปกรณ์ที่ต่อพอร์ตอนุกรม
กับเครือข่ายอีเทอร์เน็ต

PC-based Serial to Ethernet Terminal Server

ผู้จัดทำ

1. นายจิรายุ วิริยะอมรพันธุ์ รหัสประจำตัว 46060008
2. นายภวัต ชมภูแสง รหัสประจำตัว 46060029

.....อาจารย์ที่ปรึกษา
(อาจารย์สุเมธ ประภาวัต)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อ	พีซีเทอร์มินัลเซิร์ฟเวอร์เพื่อเชื่อมอุปกรณ์ที่ต่อพอร์ตอนุกรมกับเครือข่ายอีเทอร์เน็ต	
นักศึกษา	นายจิรายุ วิริยะอมรพันธุ์	46060008
	นายภวัต ชมภูแสง	46060029
ปริญญา	วิทยาศาสตรบัณฑิต	
สาขาวิชา	เทคโนโลยีสารสนเทศ	
ปีการศึกษา	2549	
อาจารย์ที่ปรึกษา	สุเมธ ประภาวัต	

บทคัดย่อ

เนื่องจากการจัดการอุปกรณ์ที่ต้องอาศัยการเชื่อมต่อผ่านพอร์ตอนุกรม เพื่อการบริหารจัดการ หรือการฝึกปฏิบัติกับอุปกรณ์ที่เป็นส่วนหนึ่งในการเรียนการสอน มักต้องกระทำที่เครื่องคอมพิวเตอร์ที่อุปกรณ์นั้นต่ออยู่โดยตรง ดังนั้นโครงงานนี้จึงได้พัฒนาพีซีเทอร์มินัลเซิร์ฟเวอร์เพื่อช่วยให้สามารถติดต่อกับอุปกรณ์ดังกล่าวจากระยะไกลผ่านระบบเครือข่ายได้ ซึ่งเทอร์มินัลเซิร์ฟเวอร์จะช่วยเป็นตัวกลางเชื่อมการติดต่อระหว่างผู้ใช้ที่ต้นทางจากระยะไกลผ่านระบบเครือข่ายกับอุปกรณ์ที่เชื่อมต่ออยู่กับพอร์ตอนุกรม โดยผู้ใช้สามารถเลือกอุปกรณ์ที่ต้องการติดต่อได้ด้วยการระบุหมายเลขพอร์ตเครือข่ายในการติดต่อ

เทอร์มินัลเซิร์ฟเวอร์หนึ่งระบบประกอบด้วยหลายโมดูล โครงงานนี้ได้พัฒนาเทอร์มินัลเซิร์ฟเวอร์ให้มีการทำงานแยกส่วนกันอย่างชัดเจนในแต่ละโมดูล เพื่อให้การปรับปรุงพัฒนาโมดูลหนึ่งไม่มีผลกระทบต่อโมดูลอื่นๆ โดยแต่ละโมดูลมีหน้าที่การทำงานที่แตกต่างกันออกไป ได้แก่ โมดูลควบคุมการติดต่อกับระบบเครือข่าย โมดูลควบคุมการติดต่อกับอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรม โมดูลสนับสนุนให้สามารถติดต่อโดยโปรแกรมเว็บบราวเซอร์ได้ และโมดูลควบคุมประสานการติดต่อกันระหว่างทุกๆ โมดูลในระบบ ซึ่งในตอนท้ายของรายงานของโครงงานนี้ จะแสดงภาพสถาปัตยกรรมของระบบ รวมทั้งแสดงการทดสอบการทำงานของระบบที่มีการเชื่อมต่อกันเป็นกลุ่มของเทอร์มินัลเซิร์ฟเวอร์หลายๆระบบเข้าด้วยกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Title	PC-based Serial to Ethernet Terminal Server	
Student	Mr. Jirayu Viriya-amonpun	46060008
	Mr. Pawat Chomphoosang	46060029
Degree	Bachelor of Science	
Programme	Information Technology	
Academic Year	2006	
Advisor	Mr. Sumet Prabhavat	

ABSTRACT

Equipment configuration management and monitoring via its serial port or console port is usually required a directly connected computer and operation at the the device. This is not convenient for either system administration in regular work or workshop trainee in exercise practice. Our project proposes the PC-based Serial to Ethernet Terminal Server system in order to provide remote access to the serial device via network. Remote users are allowed to choose a device by a network port specifying.

Our proposed system composes of several independent modules; i.e. Network Interface Module, Serial Device Interface Module, web server module (HTTP Module), and Inter-Process Communication Module. The modifcation of a module will not cause the coupling impact on the others. Finally, we present the system architecture as well as the experimental results on real system so as to prove the functionality of the single server and also the server farm system.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ขอขอบคุณเพื่อนที่ร่วมทำงานด้วยกัน ทั้งที่อยู่ด้วยกันจนเสร็จโครงงานและแยกกันก่อนที่จะเสร็จโครงงาน ที่ช่วยกันทำงานกันอย่างเต็มที่ จนโครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี รวมถึงขอบคุณเพื่อนๆคนอื่นๆที่ช่วยเป็นกำลังใจ คอยให้คำแนะนำดีๆเท่าที่รู้ และเรื่องราวสนุกสนานเสียงหัวเราะที่มีให้กันตลอดระยะเวลาที่ทำงานอยู่ด้วยกัน

ขอขอบคุณพ่อ แม่ พี่ น้อง ญาติๆและคนรู้จักอื่นๆที่คอยเป็นกำลังใจให้กับพวกเราเสมอมา พร้อมทั้งยินดีไปกับความสำเร็จของพวกเราด้วย

ขอขอบคุณหนังสือและเว็บไซต์ต่างๆ ที่คอยเป็นแหล่งข้อมูลที่ดีให้กับพวกเราได้ค้นคว้าหาความรู้ เพื่อใช้ในการทำงาน

ขอขอบคุณสถาบันเทคโนโลยี พระจอมเกล้าเจ้าคุณทหารลาดกระบังและคณะเทคโนโลยีสารสนเทศที่มอบสิ่งดีๆมากมายให้กับพวกเรา คอยเอื้อเฟื้อทั้งสถานที่ทำงาน อุปกรณ์การทำงาน ตลอดจนสภาพแวดล้อมที่อบอุ่น จนสามารถทำงานสำเร็จลุล่วงไปได้ด้วยดี

ขอขอบคุณอาจารย์คณะเทคโนโลยีสารสนเทศที่มอบความรู้ทั้งในห้องเรียนและนอกห้องเรียนให้กับพวกเรามาตลอด 4 ปี ที่เป็นนักศึกษา

และสำคัญที่สุด โครงงานนี้จะประสบความสำเร็จไม่ได้ ถ้าขาดอาจารย์ที่ปรึกษา อาจารย์สุเมธ ประภาวัต ที่คอยช่วยเหลือและให้คำแนะนำที่ดีๆมากมายในยามที่เกิดปัญหา ขอบคุณสำหรับความจริงใจ ความทุ่มเททั้งแรงกายแรงใจ และความเป็นกันเองที่อาจารย์มอบให้กับพวกเรา ขอบคุณมากครับ

จิรายุ วิริยะอมรพันธุ์

ภวัต ชมภูแสง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

รูปที่ หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

รูปที่ หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

รูปที่ หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมา

เนื่องจากในปัจจุบันอุปกรณ์ต่างๆที่ติดต่อผ่านพอร์ตอนุกรม มีอยู่มากมายในปัจจุบัน ซึ่งการที่เราจะใช้อุปกรณ์ต่างๆเหล่านั้น จำเป็นต้องติดต่อกับอุปกรณ์นั้นๆโดยตรง ผ่านเครื่องคอมพิวเตอร์ที่อุปกรณ์นั้นต่ออยู่ ซึ่งหากผู้ควบคุมต้องการเปลี่ยนแปลงการทำงานของอุปกรณ์นั้น จำเป็นต้องมาเปลี่ยนแปลงที่เครื่องคอมพิวเตอร์ที่อุปกรณ์นั้นต่ออยู่โดยตรง จะเห็นว่าการทำการเปลี่ยนแปลงการทำงานของอุปกรณ์ในแต่ละครั้งต้องเสียเวลาและค่าใช้จ่ายในการเดินทางไปยังอุปกรณ์นั้น

จึงมีแนวคิดว่า หากเราสามารถควบคุมอุปกรณ์นั้นจากระยะทางไกลๆได้โดยผ่านเครือข่ายจะทำให้ผู้ควบคุมอุปกรณ์มีความสะดวกสบายยิ่งขึ้นในการทำงานและยังช่วยลดค่าใช้จ่ายในการเดินทางมายังอุปกรณ์นั้น จึงได้พัฒนาระบบพีซีเทอร์มินัลเซิร์ฟเวอร์เพื่อเชื่อมอุปกรณ์ที่ต่อพอร์ตอนุกรมกับเครือข่ายอีเทอร์เน็ตขึ้นมา เพื่อใช้ในการควบคุมอุปกรณ์ผ่านเครือข่าย

1.2 วัตถุประสงค์

1.2.1 ศึกษาหลักการและทฤษฎีเพื่อเชื่อมโยงการติดต่อสื่อสารระหว่างเครือข่ายอีเทอร์เน็ตและอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรมตามมาตรฐาน RS-232

1.2.2 ศึกษาการพัฒนาซอฟต์แวร์แบบคอมโพแนนท์ สำหรับเชื่อมโยงการรับส่งข้อมูลบนเครือข่ายอีเทอร์เน็ตกับอุปกรณ์ที่เชื่อมต่อกับพอร์ตอนุกรมตามมาตรฐาน RS-232

1.2.3 เพื่อพัฒนาระบบที่ช่วยให้เข้าถึงอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรมตามมาตรฐาน RS-232 ผ่านเครือข่ายอีเทอร์เน็ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.3 ขอบเขตโครงงาน

1.3.1 สามารถรับส่งข้อมูลระหว่างเครือข่ายอีเทอร์เน็ตและอุปกรณ์ที่เชื่อมต่อกับพอร์ตอนุกรมตามมาตรฐาน RS-232 (ซึ่งต่อไปนี้จะใช้คำว่าพอร์ตอนุกรม)

1.3.2 สามารถรับส่งข้อมูลกับอุปกรณ์ต่างๆที่ต้องการได้โดยใช้ไอพีแอดเดรส และหมายเลขพอร์ตในการระบุอุปกรณ์ปลายทาง

1.3.3 สามารถเชื่อมต่อการทำงานระหว่างโปรเซส เพื่อทำให้สามารถทำงานร่วมกันได้

1.3.4 สามารถรวมกลุ่มเทอร์มินัลเซิร์ฟเวอร์หลายๆระบบย่อยให้เป็นหนึ่งระบบใหญ่ได้

1.3.5 สามารถให้ผู้ใช้เรียกใช้บริการเทอร์มินัลเซิร์ฟเวอร์ผ่านเว็บบราวเซอร์ได้

1.4 ขั้นตอนการศึกษา

1.4.1 ศึกษาทฤษฎีหลักการทำงานและแนวทางการพัฒนา ส่วนประกอบต่างๆของระบบดังนี้

1.4.1.1 ในส่วนของการเชื่อมต่อเครือข่าย จะศึกษาการทำงานและการพัฒนาโปรแกรมสำหรับควบคุม LAN Card เพื่อให้รับส่งข้อมูลกับเครือข่ายอีเทอร์เน็ตได้

1.4.1.2 ในส่วนของการติดต่อกับอุปกรณ์ผ่านพอร์ตอนุกรม จะศึกษาการรับส่งข้อมูลกับอุปกรณ์ที่เชื่อมต่อผ่านพอร์ตดังกล่าว

1.4.1.3 การเชื่อมต่อการรับส่งข้อมูลระหว่างเครือข่ายอีเทอร์เน็ตกับอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรม

1.4.1.4 ศึกษาระบบปฏิบัติการและเครื่องมือที่จะใช้ในการพัฒนาซอฟต์แวร์ เช่น ระบบปฏิบัติการลินุกซ์และ gcc ตามลำดับ

1.4.2 ศึกษาการพัฒนาระบบในรูปแบบคอมโพแนนท์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.4.3 ออกแบบและพัฒนาระบบในรูปแบบคอมโพแนนท์ตามขั้นตอนต่างๆ ดังนี้

1.4.3.1 คอมโพแนนท์ควบคุมการทำงานของ LAN Card เพื่อให้สามารถติดต่อสื่อสารกับเครือข่ายอีเทอร์เน็ตได้

1.4.3.2 คอมโพแนนท์ควบคุมการทำงานของพอร์ตอนุกรม เพื่อให้สามารถติดต่อสื่อสารกับอุปกรณ์ที่เชื่อมต่ออยู่ได้

1.4.3.3 คอมโพแนนท์ควบคุมการติดต่อสื่อสารด้วยโพรโทคอลเอชทีทีพี เพื่อให้สามารถติดต่อผู้ใช้ผ่านเว็บบราวเซอร์ได้

1.4.3.4 คอมโพแนนท์ที่ประสานการทำงานระหว่างคอมโพแนนท์ควบคุมการทำงานของ LAN Card และคอมโพแนนท์ควบคุมการทำงานของพอร์ตอนุกรม

1.4.4 ทดสอบประสิทธิภาพการทำงานของระบบ

1.4.5 ปรับปรุงแก้ไขเพื่อเพิ่มประสิทธิภาพและคุณสมบัติของระบบให้มีความสมบูรณ์ยิ่งขึ้น

1.5 ประโยชน์ที่จะได้รับ

1.5.1 ระบบสนับสนุนที่ช่วยให้เข้าถึงอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรม ผ่านเครือข่ายอีเทอร์เน็ตโดยการใช้ไอพีแอดเดรสและหมายเลขพอร์ตในการระบุและอ้างอิงถึงอุปกรณ์ปลายทางได้

1.5.2 ได้ต้นแบบของคอมโพแนนท์ในการควบคุมการรับส่งข้อมูลบนเครือข่ายอีเทอร์เน็ตผ่าน LAN Card และการรับส่งข้อมูลกับอุปกรณ์ที่เชื่อมต่อผ่านพอร์ตอนุกรม

1.5.3 ได้แนวคิดเริ่มต้นในการพัฒนาระบบที่ใช้ในการเชื่อมโยงข้อมูลผ่านเครือข่ายอีเทอร์เน็ตหรือการรับส่งข้อมูลกับอุปกรณ์เชื่อมต่อผ่านพอร์ตอนุกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

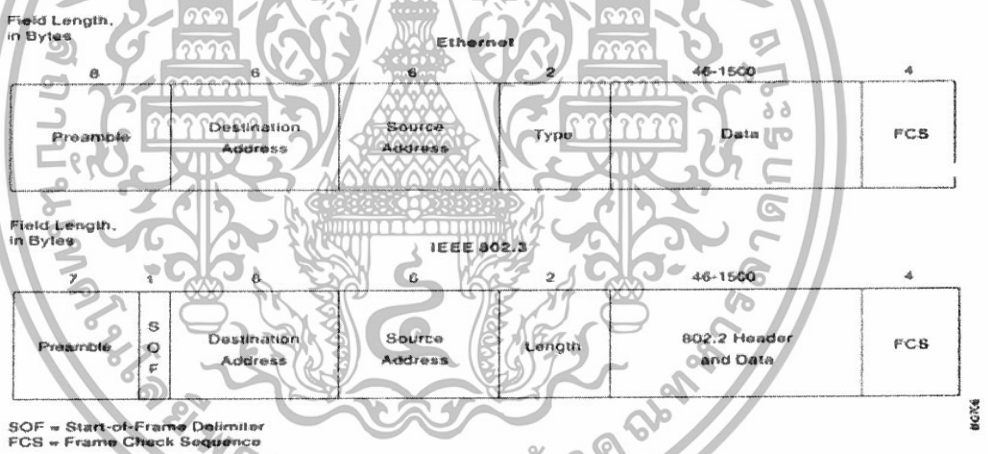
บทที่ 2
องค์ประกอบ เครื่องมือที่ใช้พัฒนา และแพลตฟอร์ม

2.1 ทฤษฎีและโพรโทคอลที่เกี่ยวข้องในองค์ประกอบต่างๆของระบบ

2.1.1 การติดต่อเครือข่าย

2.1.1.1 Ethernet Packet

Ethernet Packet เป็นข้อมูลในระดับดาต้าลิงค์เลเยอร์ ทำหน้าที่ในการตรวจสอบ ยืนยันสถานะของอุปกรณ์ที่ทำงานในระดับดาต้าลิงค์เลเยอร์ ซึ่งเฟรมของข้อมูลบนระบบเครือข่ายอีเทอร์เน็ตประกอบขึ้นด้วยกลุ่มของบิตที่เป็นข้อมูลและข่าวสารสำคัญแบ่งออกเป็นขนาดสัดส่วนที่แน่นอนดังนี้



รูปที่ 2.1 Ethernet and IEEE 802.3 Frame Formats

- **Preamble :** เป็นการส่งข้อมูลที่เป็น " 0 " และ" 1 " สลับกัน วัตถุประสงค์เพื่อบอกแก่ผู้รับว่ามีการส่งแพคเก็ตหรือมีการส่งเฟรมเข้ามาในเครือข่ายแล้ว ซึ่งจะมีใช้งานทั้งในอีเทอร์เน็ตและ IEEE 802.3 สำหรับ Preamble หากเป็นอีเทอร์เน็ตเฟรมแล้วมันจะมี 8 ไบต์ทั้งนี้เนื่องจากมันได้รวมเอาไบต์พิเศษคือ the Start-of-Frame field เข้าไปด้วย แต่ในกรณี n the IEEE 802.3 จะแยกฟิลด์นี้ออกมาต่างหาก
- **Destination และ Source Addresses :** ใช้สำหรับการบอกตำแหน่งของสถานีรับและส่งมีขนาด 6 ไบต์ โดยที่ 3 ไบต์แรกของ addresses field ถูกกำหนดโดย IEEE เพื่อใช้เป็น vendor-dependent basis ขณะที่ 3 ไบต์หลังถูกกำหนดโดยผู้ผลิต (vendor) ของอีเทอร์เน็ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หรือ IEEE 802.3 source address จะเป็น unicast address (คือต้องเป็นของผู้ใช้งานรายเดียวเท่านั้น) เสมอ ขณะที่ destination address สามารถที่จะเป็นแบบ unicast multicast (group) หรือ broadcast ก็ได้

- **Length (IEEE 802.3) :** แสดงจำนวนของไบต์ของข้อมูล ซึ่งเป็นฟิลด์ที่มาตามหลังฟิลด์นี้
- **Data (IEEE 802.3) :** หลังจากที่ฟิสิกส์คอลเลเยอร์และดาต้าลิงค์เลเยอร์ได้ประมวลผลจนสมบูรณ์แล้ว ข้อมูลที่ถูกบรรจุอยู่ในเฟรมนี้จะถูกส่งขึ้นไปยังโพรโทคอลระดับบนซึ่งจะต้องถูกตรวจสอบว่ามันเป็นส่วนข้อมูลของเฟรมถูกต้องหรือไม่ ถ้าข้อมูลในเฟรมนี้ไม่ครบถ้วนเพียงพอสำหรับการบรรจุลงในเฟรมที่มีขนาด 46 ไบต์ แล้วจะมีการเพิ่ม padding ไบต์เข้าไปเพื่อให้ข้อมูลในเฟรมมีอย่างน้อย 46 ไบต์
- **Frame Check Sequence (FCS) :** เป็น cyclic redundancy check (CRC) ขนาด 4 ไบต์ ซึ่งถูกคำนวณมาจากต้นทางและถูกส่งมารวมในเฟรม เพื่อให้ปลายทางทำการคำนวณเพื่อเปรียบเทียบกันอีกครั้ง ทั้งนี้เพื่อตรวจสอบว่า มีการผิดพลาดขึ้นในเฟรมหรือไม่

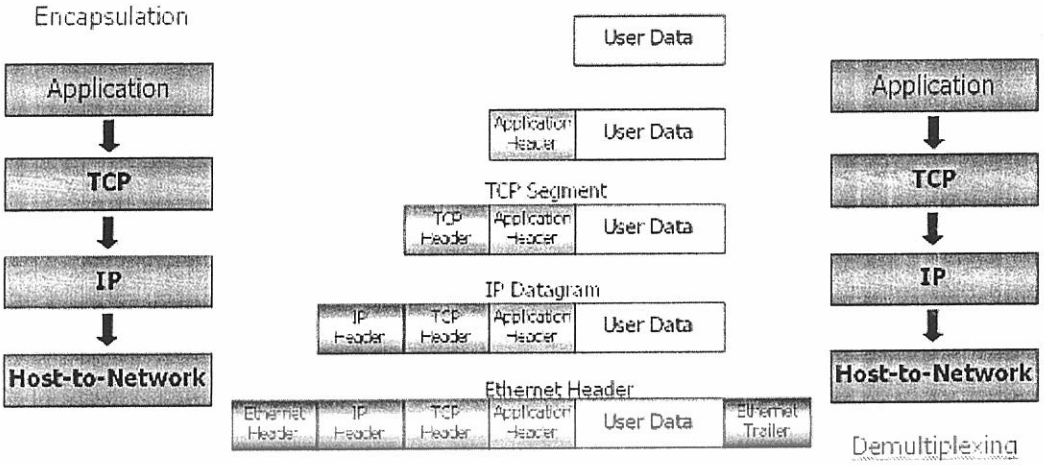
2.1.1.2 TCP/IP

เป็นชุดของโพรโทคอลที่ถูกใช้ในการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต โดยมีวัตถุประสงค์เพื่อให้สามารถใช้สื่อสารจากต้นทางข้ามเครือข่ายไปยังปลายทางได้ และสามารถหาเส้นทางที่จะส่งข้อมูลไปได้เองโดยอัตโนมัติ ถึงแม้ว่าในระหว่างทางอาจจะผ่านเครือข่ายที่มีปัญหา โพรโทคอลก็ยังคงหาเส้นทางอื่นในการส่งผ่านข้อมูลไปให้ถึงปลายทางได้

TCP/IP มีจุดประสงค์ของการสื่อสารตามมาตรฐาน สามประการคือ

1. เพื่อใช้ติดต่อสื่อสารระหว่างระบบที่มีความแตกต่างกัน
2. ความสามารถในการแก้ไขปัญหาที่เกิดขึ้นในระบบเครือข่าย เช่นในกรณีที่ผู้ส่งและผู้รับยังคงมีการติดต่อกันอยู่ แต่โหนดกลางที่ใช้เป็นผู้ช่วยรับ-ส่งเกิดเสียหายใช้การไม่ได้ หรือสายสื่อสารบางช่วงถูกตัดขาด กฎการสื่อสารนี้จะต้องสามารถจัดหาทางเลือกอื่นเพื่อทำให้การสื่อสารดำเนินต่อไปได้โดยอัตโนมัติ
3. มีความคล่องตัวต่อการสื่อสารข้อมูลได้หลายชนิดทั้งแบบที่ไม่มีความเร่งด่วน เช่น การจัดส่งแฟ้มข้อมูล และแบบที่ต้องการรับประกันความเร่งด่วนของข้อมูล เช่น การสื่อสารแบบ real-time และทั้งการสื่อสารแบบเสียง (Voice) และข้อมูล (Data)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.2 ขั้นตอนการ Encapsulation และ Demultiplexing

ข้อมูลที่ผ่านการ Encapsulate ในแต่ละเลเยอร์มีชื่อเรียกแตกต่างกัน ดังนี้

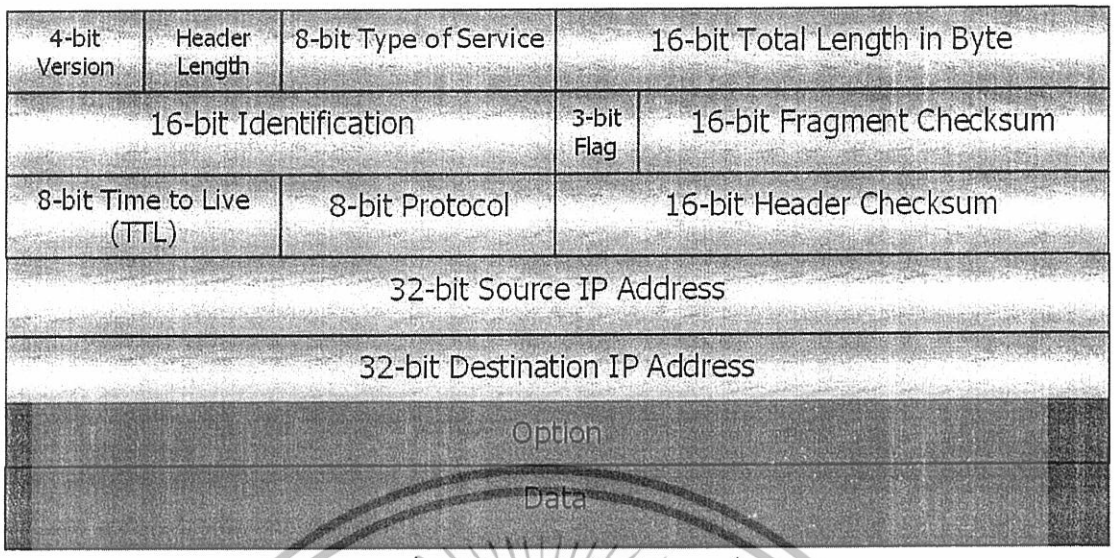
- ข้อมูลที่มาจากผู้ใช้หรือก็คือข้อมูลที่ผู้ใช้เป็นผู้ป้อนให้กับแอพพลิเคชั่นเรียกว่า ยูเซอร์ดาต้า (User Data)
- เมื่อแอพพลิเคชั่นได้รับข้อมูลจากผู้ใช้ก็จะนำมาประกอบกับส่วนหัวของแอพพลิเคชั่นเรียกว่า แอพพลิเคชั่น (Application Data) และส่งต่อไปยังทีซีพีโพรโทคอล
- เมื่อทีซีพีโพรโทคอลได้รับแอพพลิเคชั่นก็จะนำมารวมกับเฮดเดอร์ของทีซีพีโพรโทคอลเรียกว่า ทีซีพีเซกเมนต์ (TCP Segment) และส่งต่อไปยังไอพีโพรโทคอล
- เมื่อไอพีโพรโทคอลได้รับทีซีพีเซกเมนต์ก็จะนำมารวมกับเฮดเดอร์ของไอพีโพรโทคอลเรียกว่า ไอพีดาต้าแกรม (IP Datagram) และส่งต่อไปยังระดับ Host-to-Network

ในระดับ Host-to-Network จะนำไอพีดาต้าแกรมมาเพิ่มส่วน Error Correction และ Flag เรียกว่าอีเทอร์เน็ตเฟรม ก่อนจะแปลงข้อมูลเป็นสัญญาณไฟฟ้า ส่งผ่านสายสัญญาณที่เชื่อมโยงอยู่ต่อไป

IP (Internet Protocol)

ไอพีเป็นโพรโทคอลในระดับเน็ตเวิร์คเลเยอร์ ทำหน้าที่จัดการเกี่ยวกับแอดเดรสและข้อมูล และควบคุมการส่งข้อมูลบางอย่างที่ใช้ในการหาเส้นทางของแพ็กเก็ต ซึ่งกลไกในการหาเส้นทางของไอพีจะมีความสามารถในการหาเส้นทางที่ดีที่สุด และสามารถเปลี่ยนแปลงเส้นทางได้ในระหว่างการส่งข้อมูล และมีระบบการแยกและประกอบดาต้าแกรม (Datagram) เพื่อรองรับการส่งข้อมูลระดับดาต้าลิงค์เลเยอร์ที่มีขนาด MTU (Maximum Transmission Unit) ที่แตกต่างกัน ทำให้สามารถนำไอพีไปใช้บนโพรโทคอลอื่นได้หลากหลาย เช่น Ethernet Token ring หรือ Apple talk

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.3 โครงสร้างไอพีเฮดเดอร์

เฮดเดอร์ของไอพีโดยปกติจะมีขนาด 20 ไบต์ ยกเว้นในกรณีที่มีการเพิ่ม Option บางอย่าง ฟิลด์ของเฮดเดอร์ไอพีจะมีความหมายดังนี้

- **Version** : หมายเลขเวอร์ชั่นของโพรโทคอล ที่ใช้งานในปัจจุบันคือ เวอร์ชั่น 4 (IPv4 และเวอร์ชั่น 6 (IPv6)
- **Header Length** : ความยาวของเฮดเดอร์ โดยทั่วไปถ้าไม่มีส่วน Option จะมีค่าเป็น 5 (5*32 bit)
- **Type of Service (TOS)** : ใช้เป็นข้อมูลสำหรับเราเตอร์ในการตัดสินใจเลือกการเราต์ข้อมูลในแต่ละดาต้าแกรม แต่ในปัจจุบันไม่ได้มีการนำไปใช้งานแล้ว
- **Length :** ความยาวทั้งหมดเป็นจำนวนไบต์ของดาต้าแกรม ซึ่งด้วยขนาด 16 บิตของฟิลด์จะหมายถึงความยาวสูงสุดของดาต้าแกรม คือ 65535 ไบต์ (64k) แต่ในการส่งข้อมูลจริงข้อมูลจะถูกแยกเป็นส่วนๆตามขนาดของเอ็มทียูที่กำหนดในระดับดาต้าลิงค์เลเยอร์ และนำมารวมกันอีกครั้งเมื่อส่งถึง
- **Identification :** เป็นหมายเลขของดาต้าแกรมในกรณีที่มีการแยกดาต้าแกรมเมื่อข้อมูลส่งถึงปลายทางจะนำข้อมูลที่มี Identification เดียวกันมารวมกัน
- **Flag :** ใช้ในกรณีที่มีการแยกดาต้าแกรม
- **Time to live (TTL) :** กำหนดจำนวนครั้งที่มากที่สุดที่ดาต้าแกรมจะถูกส่งระหว่าง hop (การส่งผ่านข้อมูลระหว่างเครือข่าย) เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลโดยไม่สิ้นสุด โดยเมื่อข้อมูลถูกส่งไป 1 hop จะทำการลดค่า TTL ลง 1 เมื่อค่าของ TTL เป็น 0 และ

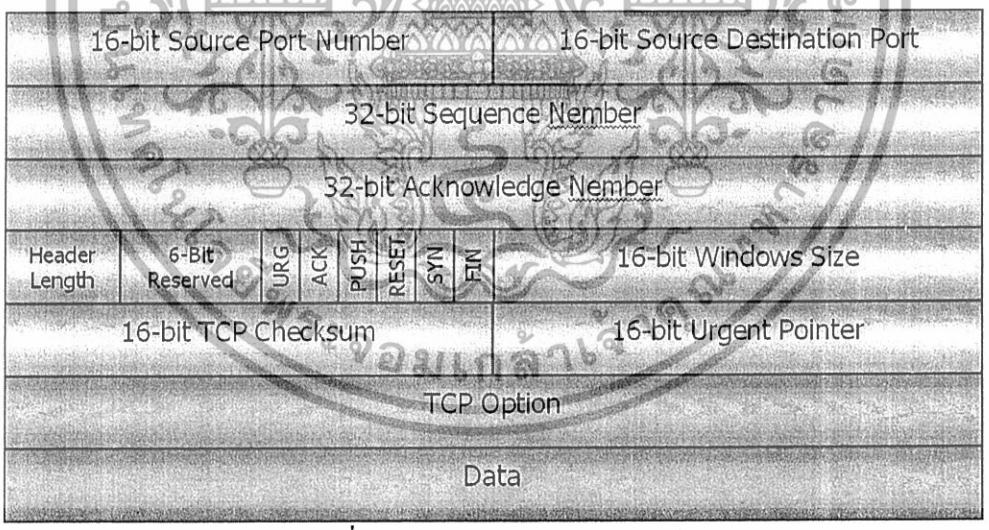
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูลยังไม่ถึงปลายทาง ข้อมูลนั้นจะถูกยกเลิก และเราเตอร์สุดท้ายจะส่งข้อมูลไอซีเอ็มพีแจ้งกลับมายังต้นทางว่าเกิดการหมดอายุ (time out) ในระหว่างการส่งข้อมูล

- **Fragment offset :** ใช้ในการกำหนดตำแหน่งข้อมูลในดาต้าแกรมที่มีการแยกส่วนเพื่อให้สามารถนำกลับมาเรียงต่อกันได้อย่างถูกต้อง
- **Protocol :** ระบุโพรโทคอลที่ส่งในดาต้าแกรม เช่น ทีซีพี ยูดีพี หรือไอซีเอ็มพี
- **Header checksum :** ใช้ในการตรวจสอบความถูกต้องของข้อมูลในเฮดเดอร์
- **Source IP address :** หมายเลขไอพีของผู้ส่งข้อมูล
- **Destination IP address :** หมายเลขไอพีของผู้รับข้อมูล
- **Data** : ข้อมูลจากโพรโทคอลระดับบน

TCP (Transmission Control Protocol)

อยู่ในระดับทรานสปอร์ตเลเยอร์เช่นเดียวกับยูดีพีทำหน้าที่จัดการและควบคุมการรับส่งข้อมูล ซึ่งมีความสามารถและรายละเอียดมากกว่ายูดีพี โดยดาต้าแกรมของทีซีพีจะมีความสัมพันธ์ต่อเนื่องกัน และมีกลไกควบคุมการรับส่งข้อมูลให้มีความถูกต้อง (Reliable) และมีการสื่อสารอย่างเป็นกระบวนการ (Connection-oriented)



รูปที่ 2.4 โครงสร้างทีซีพีเฮดเดอร์

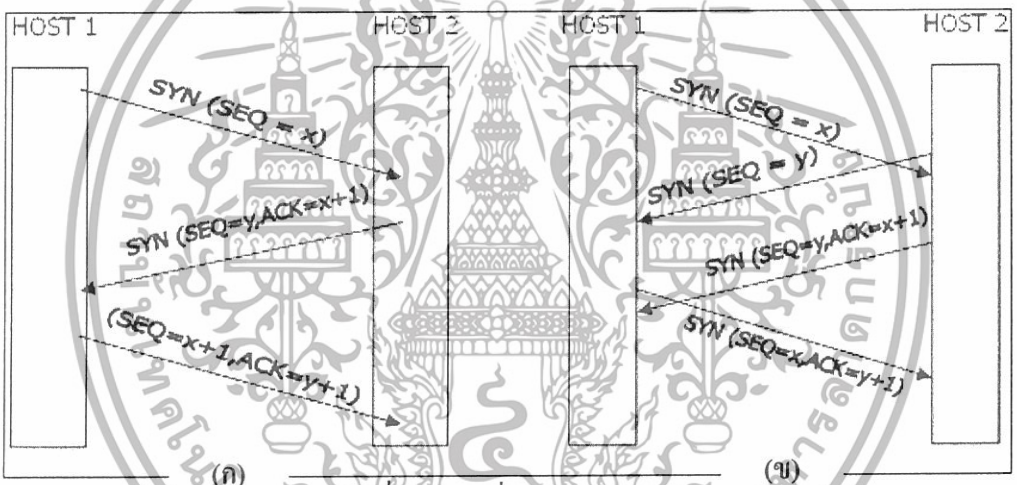
ซึ่งมีรายละเอียด ดังนี้

- **Source Port Number :** หมายเลขพอร์ตต้นทางที่ส่งดาต้าแกรมนี้
- **Destination Port Number :** หมายเลขพอร์ตปลายทางที่จะเป็นผู้รับดาต้าแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **Sequence Number :** ฟิลด์ที่ระบุหมายเลขลำดับอ้างอิงในการสื่อสารข้อมูลแต่ละครั้ง เพื่อใช้ในการแยกแยะว่าเป็นข้อมูลของชุดใด และนำมาจัดลำดับได้ถูกต้อง
- **Acknowledgment Number :** ทำหน้าที่เช่นเดียวกับ Sequence Number แต่จะใช้ในการตอบรับ
- **Header Length :** โดยปกติความยาวของทีซีพีเฮดเดอร์จะมีความยาว 20 ไบต์ แต่อาจจะมากกว่านั้น ถ้ามีข้อมูลในฟิลด์ option แต่ต้องไม่เกิน 60 ไบต์
- **Flag :** เป็นข้อมูลระดับบิตที่อยู่ในทีซีพีเฮดเดอร์ โดยใช้เป็นตัวบอกคุณสมบัติของทีซีพีแพ็กเก็ตขณะนั้นๆ และใช้เป็นตัวควบคุมจังหวะการรับส่งข้อมูลด้วย

การสื่อสารของทีซีพี



รูปที่ 2.5 การสื่อสารของทีซีพี

เมื่อเซกเมนต์ CONNECT (SYN = "1" และ ACK = "0") เดินทางมาถึงเอนทิตีทีซีพี (Entity TCP) ที่โฮสต์ปลายทางจะค้นหาโปรเซสตามหมายเลขพอร์ตที่กำหนดในฟิลด์ Destination port ซึ่งถ้าหากไม่พบก็จะตอบปฏิเสธด้วยเซกเมนต์ที่มี RST = "1" กลับไปยังผู้ส่งเซกเมนต์ CONNECT ของผู้ส่งจะถูกส่งต่อไปยังโปรเซส ตามพอร์ตที่ระบุซึ่งอาจจะตอบรับหรือตอบปฏิเสธก็ได้ ถ้าโปรเซสนั้นต้องการสื่อสารด้วยก็จะส่งเซกเมนต์ตอบรับกลับไป รูปที่ 2.5 แสดงลำดับขั้นตอนการส่งทีซีพีเซกเมนต์ในการสร้างการเชื่อมต่อในสภาวะปกติระหว่างผู้ส่งและผู้รับ

ในกรณีที่โฮสต์สองแห่งพยายามสร้างการเชื่อมต่อระหว่างซ็อคเก็ตคู่เดียวกันจะเกิดเป็นลำดับขั้นตอนแสดงในรูปที่ 2.5 (ข) ผลสุดท้ายจะมีการเชื่อมต่อเกิดขึ้นเพียงหนึ่งช่องทางเท่านั้น เนื่องจากการเชื่อมต่อในแต่ละช่องทางจะถูกกำหนดขึ้นโดยใช้หมายเลขซ็อคเก็ตผู้ส่งและผู้รับ ถ้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

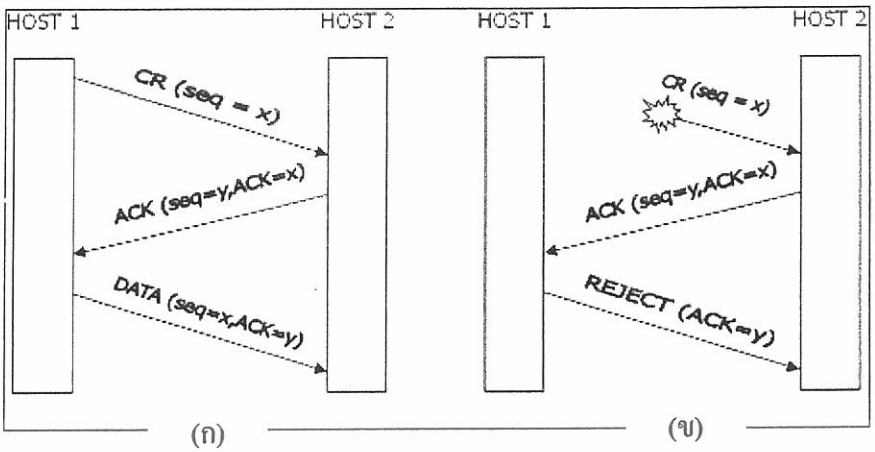
การเชื่อมต่อลำดับแรกสำเร็จก็จะถูกบันทึกไว้ในตารางการสื่อสาร เช่น (x, y) ถ้าการเชื่อมต่อลำดับที่สองสำเร็จในเวลาต่อมา ข้อมูลนี้ก็จะถูกบันทึกไว้ที่เดียวกันคือ (x, y)

การสื่อสารของทีซีพีโดยใช้การบันทึกเวลาแบบ Three-way handshake

Three-way Handshake เป็นวิธีการส่งแพ็กเก็ตที่สามารถช่วยแก้ปัญหาในเรื่องแพ็กเก็ตซ้ำซ้อนได้ดี แต่วิธีนี้จำเป็นจะต้องสร้างช่องสื่อสารให้ได้ก่อนที่จะเริ่มรับ-ส่งข้อมูล อย่างไรก็ตามแพ็กเก็ตควบคุมที่ใช้ในการต่อรองค่าตัวแปรสำหรับการสื่อสารต่างๆ อาจเกิดการตกค้างอยู่ในระบบได้ ทำให้การกำหนดค่าหมายเลขลำดับมีปัญหาไปด้วย เช่น การสร้างช่องสื่อสารระหว่างโฮสต์1 และ โฮสต์2 เริ่มจาก โฮสต์1 ขอเริ่มการเชื่อต่อด้วยการส่งแพ็กเก็ต CR (Connection Request) ไปยังโฮสต์2 ซึ่งจะมีค่าตัวแปรต่างๆสำหรับการสื่อสารรวมทั้งหมายเลขลำดับและหมายเลขช่องสื่อสารไปด้วย ผู้รับคือโฮสต์2 ก็จะส่ง ACK (Acknowledgement) กลับมายังโฮสต์1 แต่ถ้าแพ็กเก็ต จากผู้ส่งเกิดสูญหายระหว่างทางและสำเนาแพ็กเก็ตที่ยังตกค้างอยู่ระบบเกิดเดินทางไปถึงผู้รับในภายหลังก็จะทำให้การสร้างช่องสื่อสารใช้การไม่ได้เนื่องจากมีค่าตัวแปรต่างๆไม่ตรงกัน

การใช้ Three-way handshake เป็นการไม่บังคับให้ผู้ส่งและผู้รับข้อมูลจะต้องกำหนดค่าเริ่มต้นของหมายเลขลำดับเป็นเลขเดียวกัน ทำให้สามารถนำวิธีนี้มาใช้ร่วมกับวิธีการจัดจังหวะการทำงานให้พร้อมกัน (Synchronization) แบบต่างๆได้ แทนที่จะเป็นการใช้วิธีการบันทึกเวลา ดังรูปที่ 2.6 (ก) แสดงขั้นตอนการเริ่มต้นการทำงานจากโฮสต์1 ไปยังโฮสต์2 สมมุติให้โฮสต์1 เลือกหมายเลขลำดับเป็น “x” และส่งแพ็กเก็ต CONNECTION REQUEST ไปยังโฮสต์2 โฮสต์2 ตอบรับด้วยแพ็กเก็ต CONNECTION ACCEPTED ซึ่งจะยอมรับหมายเลขลำดับ “x” พร้อมกับประกาศหมายเลขลำดับ “y” ที่เป็นของตนเอง จากนั้นโฮสต์1 ก็จะตอบรับค่าตัวเลือกของโฮสต์ 2 ผ่านทางเขตข้อมูลสำหรับการควบคุมในแพ็กเก็ตข้อมูลแรกที่ส่งมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.6 การสื่อสารของทีซีพีโดยใช้ Three-Way Handshake

สมมติว่าได้เกิดปัญหาการสูญหายของแพ็กเก็ตในขณะที่สำเนาแพ็กเก็ตที่ค้างในระบบเดินทางไปถึงผู้รับแทน รูปที่ 2.6 (ข) แสดงเหตุการณ์ที่แพ็กเก็ต TPDU (Transport Protocol Data Unit) เป็นสำเนาแพ็กเก็ตเก่าที่พึ่งจะเดินทางไปถึงโฮสต์2 โดยที่โฮสต์1 ไม่ทราบ โฮสต์2 ก็จะทำงานตามปกติ คือจะตอบรับด้วยการส่งแพ็กเก็ต CONNECTION ACCEPTED TPDU กลับมา ที่โฮสต์1 ซึ่งโฮสต์1 จะสามารถตรวจสอบได้ว่า หมายเลขลำดับโฮสต์2 ตอบกลับมานั้นเป็นหมายเลขลำดับที่ได้เลิกใช้ไปแล้ว จึงมีการส่งแพ็กเก็ต REJECT กลับมายังโฮสต์2 เพื่อบอกยกเลิกการทำงาน จะเห็นว่าวิธีการนี้อาศัยการสื่อสารผ่านแพ็กเก็ต 3 ตัวซึ่งเป็นที่มาของคำว่า "การจับมือร่วมสามขั้นตอน" ผลสุดท้าย ทั้งโฮสต์1 และโฮสต์2 ก็จะไม่มีการสร้างช่องสื่อสารขึ้นมาจากข้อมูลในสำเนาแพกเก็ตเก่า

2.1.1.3 HTTP (Hyper text transfer protocol)

เอชทีทีพีเป็นกลไกหรือโพรโทคอลหลักที่ใช้ในการใช้บริการข้อมูลทางเวิลด์ไวด์เว็บโดยจะทำงานเป็นแบบไคลเอนต์เซิร์ฟเวอร์ ซึ่งจะเชื่อมต่อกันแบบ Connection Oriented ก็คือทั้งสองฝั่งต้องสร้างช่องทางเพื่อการสื่อสารกันก่อนที่จะส่งข้อมูลและเมื่อส่งข้อมูลกันแล้วก็จะปิดการเชื่อมต่อทันที จนกว่าจะมีการร้องขอมาเพื่อขอข้อมูลอีกก็จะเปิดการเชื่อมต่อให้ใหม่ การบริการข้อมูลเวิลด์ไวด์เว็บหรือที่เรียกกันว่า "บริการเว็บไซต์" นั้น จะต้องมีด้วยกัน 2 ฝั่ง นั้นคือ ฝั่งที่เป็นเซิร์ฟเวอร์และฝั่งที่เป็นไคลเอนต์ โดยฝั่งที่เป็นเซิร์ฟเวอร์จะมีโปรแกรมอยู่ประเภทหนึ่งทำงานอยู่นั้นคือเว็บเซิร์ฟเวอร์ โดยโปรแกรมนี้จะทำงานโดยเปิดพอร์ต 80 เอาไว้ และจะรอการเชื่อมต่อจากไคลเอนต์ โดยเราจะต้องใช้โปรแกรมที่เรียกว่า เอชทีทีพีไคลเอนต์ (HTTP Client) หรืออินเทอร์เน็ตเอ็กซ์โพเลอร์ (Internet Explorer) ในการติดต่อมายังเว็บเซิร์ฟเวอร์นั้นเอง ซึ่งเอชทีทีพีไคลเอนต์จะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใช้คำสั่งในการร้องขอไฟล์ก็คือ GET โดยเวลาส่งคำสั่งนี้ไป เราจะตามด้วยชื่อไฟล์ URI ที่ต้องการ เช่น

GET /	ขอไฟล์แรกของเว็บไซต์
GET /index.html	ขอไฟล์ index.html

โครงสร้างข้อมูลที่ใช้ในเอชทีทีพีโพรโทคอล จะแบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ

1. Header หรือเรียกว่า เมตาดาต้า (Metadata) จะเป็นส่วนที่เก็บรายละเอียดของเอกสารนั้นว่าเป็นเอกสารประเภทไหน ขนาดเท่าไร
2. Body ส่วนนี้จะเป็นส่วนของเนื้อหา ซึ่งจะอยู่ต่อจากส่วนของเฮดเดอร์เป็นส่วนเนื้อหาเอชทีเอ็มแอล (HTML)

ข้อมูลทั้งสองส่วนนี้จะถูกส่งกลับมาเมื่อไคลเอนต์ส่งคำสั่ง GET ไปขอไฟล์จากเว็บเซิร์ฟเวอร์ โดยคำสั่งที่ใช้ในเอชทีทีพีโพรโทคอลมีดังต่อไปนี้

ตารางที่ 2.1 ตารางคำสั่งในเอชทีทีพีโพรโทคอล

คำสั่ง	รายละเอียด
GET	ใช้อ่านข้อมูลจากเว็บเซิร์ฟและส่งไปยังไคลเอนต์โดยมีรูปแบบดังนี้ GET <URI> HTTP/1.0 ตัวอย่างเช่น ต้องการให้เว็บเซิร์ฟเวอร์ส่งไฟล์ sale.html จากโดเมน www.netcorp.com ไปยังไคลเอนต์จะใช้รูปแบบของคำสั่ง GET ดังนี้ GET /sale.html html/1.0 นอกจากนี้คำสั่ง GET ยังสามารถกำหนดเงื่อนไขให้อ่านข้อมูลจากเว็บเซิร์ฟเวอร์เฉพาะที่มีการเปลี่ยนแปลงแก้ไขได้ด้วย
HEAD	คำสั่งนี้จะทำงานคล้ายกับคำสั่ง GET แต่เว็บเซิร์ฟเวอร์จะส่งข้อมูลกลับมาให้เฉพาะในรายละเอียดของเมตาดาต้าหรือข้อมูลในเฮดเดอร์เท่านั้น ส่วนข้อมูลที่เป็นเอชทีเอ็มแอลจะไม่ถูกส่งมาด้วย ซึ่งคำสั่ง HEAD นี้จะใช้เพื่อทดสอบว่าข้อมูลตาม URI นั้นๆมีการเปลี่ยนแปลงหรือไม่เท่านั้น
POST	เป็นคำสั่งที่ตรงข้ามกับคำสั่ง GET และ HEAD โดยทำหน้าที่ส่งข้อมูลจากไคลเอนต์ไปยังเว็บเซิร์ฟเวอร์นั้นจะไม่ค่อยมีใช้งาน นอกจากในกรณีที่เอชทีเอ็มแอลทำงานในลักษณะที่ให้ผู้ใช้กรอกข้อมูลตามแบบฟอร์ม (เช่น รายละเอียดส่วนตัวของผู้ใช้งาน) และส่งข้อมูลนี้มาเก็บที่เว็บเซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PUT	เป็นคำสั่งที่ทำงานเหมือนกับคำสั่ง POST แต่ไม่เป็นที่นิยม
DELETE	เพื่อให้ไคลเอนต์สั่งเว็บเซิร์ฟเวอร์ลบ URI ที่กำหนดไว้ออกจากเว็บเซิร์ฟเวอร์ แต่เป็นคำสั่งที่ไม่นิยมใช้มากนัก เนื่องจากเว็บเซิร์ฟเวอร์ทั่วไปมักจะทำงานในแบบอ่านข้อมูลได้เท่านั้น (read-only)
LINK	เป็นคำสั่งที่เชื่อม URI ที่ต้องการไปยังเว็บเซิร์ฟเวอร์อื่น
UNLINK	UNLINK

สถานะการทำงานของเอชทีทีพี

เอชทีทีพีโพรโทคอลได้กำหนดรหัสแสดงสถานะการทำงานของโพรโทคอลไว้ โดยแบ่งกลุ่มของรหัสสถานะออกไว้เป็น 5 กลุ่ม คือ

ตารางที่ 2.2 ตารางรหัสแสดงสถานะการทำงานของเอชทีทีพีโพรโทคอล

รหัสสถานะ	**ประเภท**	**รายละเอียด**
100-199	Information	เป็นรหัสสถานะกลุ่มที่เปิดให้โปรแกรมประยุกต์ต่างๆ กำหนดใช้งานได้เอง
200-299	Successful	กลุ่มรหัสที่แสดงว่าการทำงานสำเร็จ
300-399	Redirection	กลุ่มรหัสนี้จะใช้ภายในเอชทีทีพีโพรโทคอลเอง โดยเป็นการทำงานที่ต่อเนื่องมาจากโปรเซสก่อนหน้านี้ ซึ่งไคลเอนต์เป็นผู้สั่งงาน
400-499	Client Error	ใช้แสดงการปัญหาที่เกิดขึ้นกับไคลเอนต์
500-599	Server Error	ใช้แสดงการปัญหาที่เกิดขึ้นกับเซิร์ฟเวอร์

2.1.2 การติดต่อกับอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรม

2.1.2.1 มาตรฐานพอร์ตอนุกรมแบบ RS- 232

มาตรฐานพอร์ตอนุกรมแบบ RS- 232 เป็นที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโคนัส 2 ทิศทาง โดยมาตรฐาน RS–232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้อมูล DTE (Data Terminal Equipment) กับวงจรข้อมูลปลายทาง DCE (Data Circuit Terminating) ไว้ว่า อุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์ หรือไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสอง

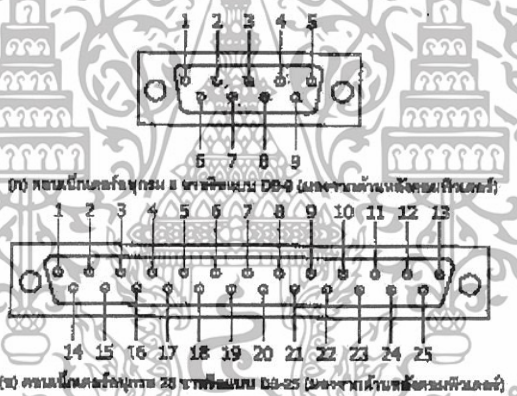
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะกระทำผ่านมาตรฐาน RS-232 ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ DCE

สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม RS-232 มักถูกใช้เชื่อมต่อกับโมเด็มโดยสามารถรับส่งข้อมูลได้ที่ความยาวของสายสัญญาณสูงสุดถึง 20 เมตร

คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้ หรือ DB-9 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้นเช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่นๆ ที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป โดยแสดงดังรูปที่ 2.7 ต่อไปนี้



รูปที่ 2.7 คอนเน็กเตอร์แบบ DB-25 และ DB-9

ตารางที่ 2.3 ตารางการจัดขาของคอนเน็กเตอร์พอร์ตอนุกรม ตามมาตรฐาน RS-232 ทั้งแบบ DB-9 และ DB-25

คอนเน็กเตอร์ DB-9	คอนเน็กเตอร์ DB-25	ชื่อของสายสัญญาณ	ชนิดของสายสัญญาณ
1	8	Data Carrier Detect : DCD	อินพุต
2	3	Received Data : RxD	อินพุต
3	2	Transmitted Data : TxD	เอาต์พุต
4	20	Data Terminal Ready : DTR	เอาต์พุต
5	7	Signal Ground : GND	-

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6	6	Data Set Ready : DSR	อินพุต
7	4	Request To Send : RTS	เอาต์พุต
8	5	Clear To Sent : CTS	อินพุต
9	22	Ring Indicator : RI	อินพุต

รูปที่ 2.8 การต่ออุปกรณ์ภายนอกกับพอร์ตอนุกรมของคอมพิวเตอร์ในลักษณะต่าง ๆ

สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกแสดงดังในรูปที่ 2.8 ลูกศรในรูปแสดงถึงทิศทางของข้อมูล ในรูปที่ 2.8 (ก) เป็นการเชื่อมต่อแบบ Null modem หรือการเชื่อมต่อโดยตรงโดยไม่ต้องผ่านโมเด็ม โดยมีการตรวจสอบหรือแฮนด์เช็กเต็มรูปแบบ ส่วนในรูปที่ 2.8 (ข) เป็นการเชื่อมต่อแบบ Null modem ในลักษณะที่ใช้สายสัญญาณเพียง 3 เส้น โดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นสำหรับรับข้อมูล และเส้นสุดท้ายเป็นกราวด์

รายละเอียดหน้าที่การทำงานในแต่ละขาของพอร์ตอนุกรม RS-232 มีดังนี้

- **Data Carrier Detect (DCD) :** ขานี้จะแอกตีฟเมื่อมีการ ส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งานมากนัก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **Receive Data (RD หรือ RxD)** : ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสเตอร์บัฟเฟอร์
- **Transmitted Data (TD หรือ TxD)** : ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป
- **Data Terminal Ready (DTR)** : เป็นขาสัญญาณที่ส่งออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่า ต้องการติดต่อด้วย โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง และขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกันและต้องต่อขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณ
- **Signal Ground (GND)** : ขากราวด์ของระบบ
- **Data Set Ready (DSR)** : ขานี้จะใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR
- **Request To Sent (RTS)** : เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ในกรณีที่ใช้การเชื่อมต่อแบบ Null model 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS ของตัวมันเองเข้าด้วยกัน เพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา
- **Clear To Sent (CTS)** : ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลหรือไม่
- **Ring Indicator (RI)** : ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น

2.1.3 การติดต่อระหว่างโปรเซส

2.1.3.1 Interprocess Communication

การทำงานของระบบแต่ละระบบนั้น อาศัยการทำงานของโปรเซสต่างๆหลายๆโปรเซสทำงานร่วมกัน ดังนั้นระหว่างโปรเซสจึงมีการติดต่อสื่อสารกันเพื่อให้สามารถทำงานร่วมกันได้ การติดต่อสื่อสารกันระหว่างโปรเซสนั้นเรียกว่า อินเทอร์โปรเซสคอมมูนิเคชั่น (Interprocess Communication) หรือเรียกย่อๆว่า ไอพีซี (IPC) เป็นเทคนิคและกลไกหลักที่จัดการกับการติดต่อสื่อสารระหว่างโปรเซส โปรเซสอาจถูกรันจากคอมพิวเตอร์เครื่องเดียวหรือหลายเครื่องโดยมี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การติดต่อกันทางเครือข่าย ซึ่งเทคนิคไอพีซีที่ทำการศึกษาได้แก่ ไฟล์ไปป์และ Message queue ซึ่งมีรายละเอียดดังนี้

ไฟล์ไปป์ (Pipe)

เป็นวิธีการติดต่อกันระหว่างโปรเซส 2 โปรเซส โดยที่ไฟล์ไปป์จะถูกใช้งานเป็นตัวกลางในการส่งผ่านข้อมูลระหว่างโปรเซส ซึ่งการส่งข้อมูลจะเป็นในลักษณะสื่อสารทางเดียว (One way flow) ทำงานผ่านเคอร์เนล เพื่อนำเอาต์พุตของโปรเซสเป็นอินพุตของอีกโปรเซสหนึ่ง

popen – pclose เป็นการสร้างไฟล์ไปป์โดยใช้คำสั่งในไลบรารี

- popen จะทำให้โปรแกรมสามารถเรียกใช้งานโปรแกรมใหม่ เสมือนเป็นโปรเซสใหม่ พร้อมกับส่งหรือรับข้อมูล จากโปรเซสใหม่นั้น การทำงานของฟังก์ชัน popen จะทำงานโดยเรียกเชลล์อีกต่อหนึ่ง โดยส่งข้อความคำสั่ง (command) เป็นอาร์กิวเมนต์ ซึ่งการทำงานในลักษณะนี้จะมีผลดีและผลเสีย ในยูนิกซ์ การหาค่าอาร์กิวเมนต์ของคำสั่ง (Shell Expansion) จะทำงานโดยเชลล์ และแน่นอน เมื่อมีการเรียกใช้เชลล์ จะต้องใช้รีซอร์สไปส่วนหนึ่งด้วยเช่นกัน
- pclose เมื่อโปรเซสรันฟังก์ชัน popen เสร็จ เราสามารถจะปิดไฟล์สตรีมของไปป์ด้วยฟังก์ชัน pclose แต่หากฟังก์ชันกำลังทำงานอยู่ ฟังก์ชัน pclose จะรอจนกว่าจะทำงานเสร็จ

คำสั่ง pipe

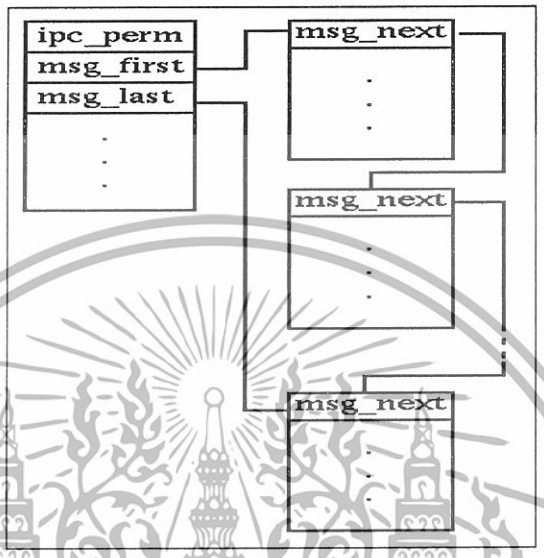
เป็นการสร้างไฟล์ไปป์ของโปรเซสโดยใช้ฟังก์ชันระดับต่ำ ใช้สำหรับส่งผ่านข้อมูลระหว่าง 2 โปรเซส โดยไม่ต้องผ่านเชลล์ ซึ่งจะทำให้เราสามารถควบคุมการอ่านและเขียนข้อมูลได้ดีขึ้น การติดต่อระหว่างโปรเซส โดยผ่านทาง file_descriptor ซึ่งเป็นอาร์เรย์ 2 มิติ file_descriptor[1] เป็นทางที่ไว้สำหรับเขียนข้อมูล ส่วน file_descriptor[0] เป็นทางที่ไว้สำหรับอ่านข้อมูล

Message Queue

เป็นวิธีการติดต่อสื่อสารระหว่างโปรเซสใน System V มีลักษณะคล้ายกับไฟล์ไปป์ แต่ละโปรเซสที่ต้องการติดต่อสื่อสารกันจะรับส่งข้อมูลผ่านคิวตัวเดียวกัน ข้อมูลที่ถูกส่งมาจะเก็บไว้ในลักษณะของบล็อกข้อมูลเรียงต่อๆกันเป็นคิว แต่ละคิวและบล็อกข้อมูลจะมีหมายเลขเพื่อใช้ในการอ้างอิง แต่สิ่งที่แตกต่างกับไฟล์ไปป์ก็คือไม่ต้องกำหนดจังหวะการทำงานร่วมกันของแต่ละโปรเซสที่ใช้งาน เป็นการติดต่อสื่อสารกันแบบอะซิงโครนัส (Asynchronous) ซึ่งก็หมายความว่าโปรเซส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
73060

สามารถจะเขียนข้อมูลไปยังคิวแล้วจบการรันโปรเซส โดยที่โปรเซสอื่นสามารถจะอ่านข้อมูลนั้นในภายหลังได้และข้อมูลที่ถูกเขียนลงไปไว้ที่คิวจะมีการจัดการกับข้อมูลในลักษณะเข้าก่อนออกก่อน (FIFO) แต่ข้อเสียคือ ขนาดของบล็อกข้อมูลมีขนาดที่จำกัด รวมทั้งจำนวนบล็อกข้อมูลทั้งหมดที่มีบนระบบก็ถูกจำกัดไว้เช่นกัน



รูปที่ 2.9 โครงสร้างของคิวในเคอร์เนล

พารามิเตอร์ต่างๆที่ใช้ในการกำหนดการทำงานของ Message queue มีดังนี้

- msgmax ขนาดที่ใหญ่ที่สุดของแต่ละบล็อกข้อมูล น้อยที่สุดเท่ากับ 0 ไบต์ มากที่สุด 65536 ไบต์ และถ้าไม่กำหนดจะเท่ากับ 8192 ไบต์
- msgmnb ขนาดที่มากที่สุดของบล็อกข้อมูลทั้งหมดที่สามารถบรรจุภายใน Message queue น้อยที่สุดเท่ากับ 0 ไบต์ มากที่สุด 65536 ไบต์ และถ้าไม่กำหนดจะเท่ากับ 16384 ไบต์
- msgmni จำนวน Message queue ที่มากที่สุดที่ยอมให้สร้างในระบบ น้อยที่สุดเท่ากับ 1 มากที่สุดตามขนาดของหน่วยความจำ และถ้าไม่กำหนดจะเท่ากับ 50
- msgseg จำนวนบล็อกข้อมูลทั้งหมดในทุก Message queue ที่มากที่สุด น้อยที่สุดเท่ากับ 1 มากที่สุด 32767 และถ้าไม่กำหนดจะเท่ากับ 2048

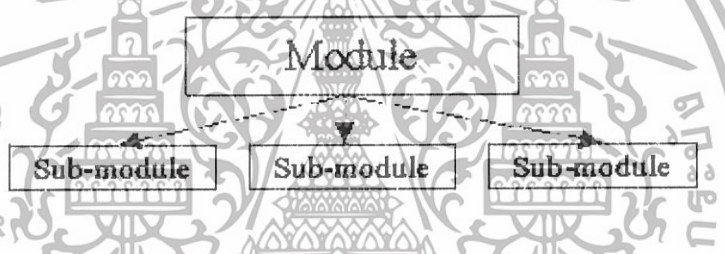
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 หลักการและภาษาที่ใช้ในการพัฒนาระบบ

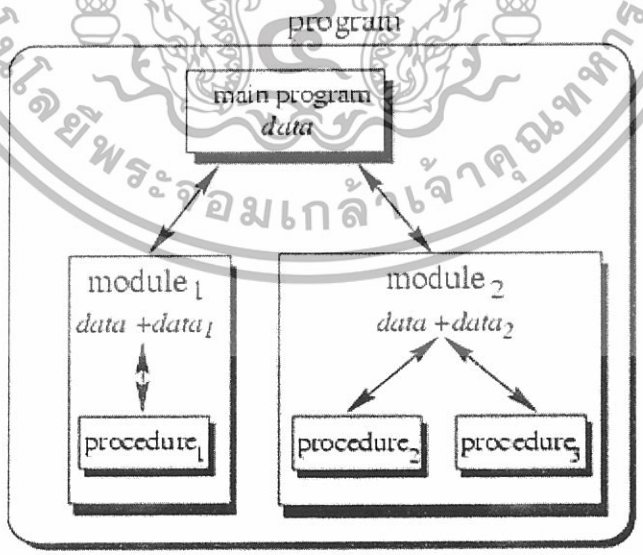
2.2.1 การเขียนโปรแกรมส่วนจำเพาะ (Modularity Programming)

การเขียนโปรแกรมส่วนจำเพาะคือ การเขียนโปรแกรมเป็นส่วนๆ ทำให้สามารถแก้ไขได้ง่าย และสามารถใช้หลายคนเขียนได้ ข้อดียิ่งไปกว่านั้นก็คือ สามารถจัดการเกี่ยวกับหน่อยความจำได้ดีขึ้นด้วย

- ออกแบบการทำงานให้สามารถทำงานได้หลายๆโปรแกรม
- เริ่มต้นด้วยการกำหนดปัญหา
- แตกปัญหาออกแก้ไขเป็นส่วนๆ
- การทำงานเฉพาะส่วนให้สามารถแก้ไขปัญหาในแต่ละส่วนได้
- การทำงานโดยรวมแก้ไขปัญหาทั้งหมดได้
- การทำงานแบบแบ่งส่วนสามารถแบ่งส่วนการทำงานย่อยๆลงไปได้อีก



รูปที่ 2.10 การแบ่งโมดูลการทำงานย่อย



รูปที่ 2.11 Modular programming. The main program coordinates calls to procedures in separate modules and hands over appropriate data as parameters

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.2 การเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming)

การเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming, OOP) เป็นแบบอย่างการเขียนโปรแกรมคอมพิวเตอร์ชนิดหนึ่ง

การเขียนโปรแกรมเชิงวัตถุใช้แนวคิดที่ให้ความสำคัญกับวัตถุ ซึ่งสามารถนำมาประกอบกันและนำมาทำงานรวมกันได้ โดยการแลกเปลี่ยนข่าวสารเพื่อนำมาประมวลผลและส่งข่าวสารที่ได้ไปให้วัตถุอื่นๆที่เกี่ยวข้องเพื่อให้ทำงานต่อไป

แนวคิดการเขียนโปรแกรมแบบดังเดิมมักนิยมใช้การเขียนโปรแกรมเชิงกระบวนการ (Procedural Programming) ซึ่งให้ความสำคัญกับขั้นตอนกระบวนการที่ทำ โดยแบ่งโปรแกรมออกเป็นส่วนๆตามลำดับขั้นตอนการทำงาน แต่แนวคิดการเขียนโปรแกรมเชิงวัตถุนั้นให้ความสำคัญกับข้อมูลและพฤติกรรม (Behavior) ของวัตถุและความสัมพันธ์กันระหว่างวัตถุกันมากกว่า

แนวคิดที่สำคัญของการเขียนโปรแกรมเชิงวัตถุ

คลาส (Class) – เป็นชนิดของวัตถุ โดยกำหนดว่า วัตถุจะประกอบไปด้วยข้อมูลหรือคุณสมบัติ (Property) และพฤติกรรมหรือการกระทำ (Method) อะไรบ้าง ซึ่ง คลาส (เช่น มนุษย์) เป็นโครงสร้างพื้นฐานของการเขียนโปรแกรมเชิงวัตถุ

วัตถุ (Object) - โดยมากจะเรียกว่า ออปเจ็ค คือ ตัวตน (Instance) ของคลาส (เช่น นายทักษิณ, นายสนธิ) ซึ่งจะเกิดขึ้นระหว่าง run-time โดยแต่ละออปเจ็คจะมีข้อมูลเฉพาะของตัวเอง ทำให้ออปเจ็คแต่ละออปเจ็คของคลาสซึ่งใช้ซอสโค้ดเดียวกันมีคุณลักษณะและคุณสมบัติที่แตกต่างกัน

Encapsulation - วิธีการกำหนดสิทธิในการเข้าถึงข้อมูล หรือการกระทำกับออปเจ็คของคลาสนั้นๆ ทำให้แน่ใจได้ว่าข้อมูลของออปเจ็คนั้นจะถูกเปลี่ยนแปลงแก้ไขผ่านทางการกระทำที่อนุญาตเท่านั้น (เช่น การกำหนดตำแหน่งทางการเมือง เป็น public method ที่ผู้อื่นสามารถกระทำได้ ส่วนการลาออกจากตำแหน่ง เป็น private method ที่มีแต่ออปเจ็คของคลาสเท่านั้นที่จะสามารถทำได้ แต่การกดดันและการขับไล่สามารถสร้างข้อมูลที่อาจจะส่งผลเกิดการลาออกได้เช่นกัน)

Inheritance - การสืบทอดคุณสมบัติ เป็นวิธีการสร้างคลาสย่อย (Subclass) ซึ่งจะเป็นกำหนดประเภทของวัตถุให้จำเพาะเจาะจงขึ้น ซึ่งซับคลาสจะได้รับถ่ายทอดคุณสมบัติต่างๆ (Attribute Behavior และ Relationship) มาจากคลาสหลักด้วย (เช่น คลาส มนุษย์ สืบทอดมากจากคลาส สิ่งมีชีวิต)

Abstraction - การแสดงถึงคุณลักษณะและพฤติกรรมของออปเจ็คเท่าที่จำเป็นต้องรับรู้และใช้งาน โดยซ่อนส่วนที่เหลือเอาไว้เพื่อไม่ให้เกิดความสับสน เช่น ตามปกติแล้ว นายทักษิณ จัดเป็นตัวตนของ คลาส มนุษย์ ซึ่งจะมีพฤติกรรม การกระทำทุกอย่างที่ตามที่กำหนดไว้ตามโครงสร้างของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คลาสมนุษย์ แต่ในบางกรณีที่นำไปใช้งาน เราไม่ต้องการให้เกิดการสับสนต่อการใช้งานหรือการจัดประเภทมากเราสามารถจัดการหรือใช้งาน ออปเจ็ค นายทักษิณ ให้อยู่ในรูปของสิ่งมีชีวิตก็ได้

Polymorphism - เป็นวิธีการกำหนดรูปแบบการกระทำที่เหมือนกันแต่มีกระบวนการทำที่ต่างกัน เช่น การเคลื่อนที่เป็นการกระทำหลักของคลาสสิ่งมีชีวิต ซึ่งมีคลาสมนุษย์และคลาสปลาเป็นซับคลาสแต่การเคลื่อนที่ของออปเจ็คจากคลาสทั้งสองจะทำไม่เหมือนกัน

2.2.3 ภาษาซี

ภาษาซี (C programming language) เป็นภาษาโปรแกรมเชิงโครงสร้างระดับสูงที่ได้รับการพัฒนาขึ้นในช่วงทศวรรษ 1970 โดย เคน ธอมป์สัน (Ken Thompson) และ เดนนิส ริทชี่ (Dennis Ritchie) สำหรับใช้ในระบบปฏิบัติการยูนิกซ์ ต่อมาภายหลังได้ถูกนำไปใช้กับระบบปฏิบัติการอื่นๆ และกลายเป็นภาษาโปรแกรมหนึ่งที่ใช้กันแพร่หลายมากที่สุด ภาษาซีมีจุดเด่นที่ประสิทธิภาพในการทำงาน เนื่องจากมีความสามารถใกล้เคียงกับภาษาระดับต่ำ แต่เขียนแบบภาษาระดับสูง โปรแกรมคอมพิวเตอร์ที่เขียนด้วยเป็นภาษาซีจึงทำงานได้รวดเร็ว ภาษาซีเป็นภาษาโปรแกรมที่นิยมใช้กันมากสำหรับพัฒนาระบบปฏิบัติการ ซอฟต์แวร์ระบบ ควบคุมไมโครคอนโทรลเลอร์และเป็นภาษาที่ใช้กันทั่วไปในหลักสูตรวิทยาการคอมพิวเตอร์

2.2.4 ภาษาซีพลัสพลัส

ภาษาซีพลัสพลัส (C++ programming language) เป็นภาษาโปรแกรมคอมพิวเตอร์อเนกประสงค์ มีโครงสร้างภาษาที่มีการจัดชนิดข้อมูลแบบสแตติก (statically typed) และสนับสนุนรูปแบบการเขียนโปรแกรมที่หลากหลาย (multi-paradigm language) ได้แก่ การเขียนโปรแกรมเชิงกระบวนคำสั่ง การนิยามข้อมูล การเขียนโปรแกรมเชิงวัตถุ และการเขียนโปรแกรมแบบเจเนริก (generic programming) ภาษาซีพลัสพลัสเป็นภาษาโปรแกรมเชิงพาณิชย์ที่นิยมมากภาษาหนึ่งนับตั้งแต่ช่วงทศวรรษ 1990

Bjarne Stroustrup จากห้องวิจัยเบลล์ (Bell Labs) เป็นผู้พัฒนาภาษา C++ ขึ้น (เดิมใช้ชื่อ "C with classes") ในปี ค.ศ. 1983 เพื่อพัฒนาภาษาซีดั้งเดิม สิ่งที่พัฒนาขึ้นเพิ่มเติมนั้นเริ่มจากการเพิ่มเติมการสร้างคลาสจากนั้นก็เพิ่มคุณสมบัติต่างๆ ตามมา ได้แก่ เวอร์ชวลฟังก์ชัน การโอเวอร์โหลดโอเปอเรเตอร์ การสืบทอดหลายสาย เท็มเพลตและการจัดการเอ็กเซ็พชัน มาตรฐานของภาษาซีพลัสพลัสได้รับการรับรองในปี ค.ศ. 1998 เป็นมาตรฐาน ISO/IEC 14882:1998 เวอร์ชันล่าสุดคือเวอร์ชันในปี ค.ศ. 2003 ซึ่งเป็นมาตรฐาน ISO/IEC 14882:2003 ในปัจจุบันมาตรฐานของภาษาในเวอร์ชันใหม่ (รู้จักกันในชื่อ C++0x) กำลังอยู่ในขั้นพัฒนา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.5 HTML (HyperText Markup Language)

เอชทีเอ็มแอลเป็นสิ่งที่ผู้พัฒนาโฮมเพจคุ้นเคยกันดี และเป็นภาษาที่ออกแบบมาเพื่อให้โปรแกรมบราวเซอร์สามารถเข้าใจ และทำงานได้ในแบบของไฮเปอร์เท็กซ์ ซึ่งผู้สร้างเว็บเพจจะใช้ภาษาเอชทีเอ็มแอลนี้ในการสร้างเว็บเพจและเก็บไว้ในเว็บเซิร์ฟเวอร์ เมื่อมีผู้ใช้งานติดต่อผ่านโปรแกรมบราวเซอร์ที่เครื่องไคลเอนต์ โดยระบุ URL ของเว็บเซิร์ฟเวอร์นั้นๆ ไฟล์เอชทีเอ็มแอลที่เก็บไว้ในเว็บเซิร์ฟเวอร์ก็จะถูกส่งไปยังไคลเอนต์โดยใช้โพรโทคอลเอชทีทีพีและแสดงผลให้ผู้ใช้งานเห็นโดยผ่านโปรแกรมบราวเซอร์ เอชทีเอ็มแอลมีพื้นฐานมาจากภาษาที่เรียกว่า SGML (Standard Generalize Markup Language) และเอชทีเอ็มแอลได้รับการออกแบบมาให้ใช้งานกับเว็บเพจที่ไม่มีการเปลี่ยนแปลงหรือเคลื่อนไหวในเพจนั้นๆ ซึ่งในกรณีที่ต้องการพัฒนาให้เว็บเพจสามารถเปลี่ยนแปลงหรือเคลื่อนไหว (Dynamic html) ได้นั้น จะต้องใช้ภาษาอื่นเข้ามาร่วมด้วย เช่น จาวาหรือภาษาที่เป็นสคริปต์ต่างๆ

แท็กเอชทีเอ็มแอล (Tag HTML)

ความเข้าใจในเรื่องของแท็กเป็นส่วนสำคัญของการเรียนรู้เอชทีเอ็มแอล โดยแท็กจะทำหน้าที่กำหนดขอบเขตหรือแบ่งแยกการสั่งงานต่างๆ ในไฟล์เอชทีเอ็มแอล ให้ตัวแปลภาษาสามารถเข้าใจได้ หน้าที่แท็กคือใช้ในการสร้างเฮดดิ้ง กำหนดย่อหน้า สร้างลิสต์ กำหนดรูปแบบ (Formatting) และการเชื่อมไปยังเว็บอื่นๆ โดยการเริ่มต้นแท็กจะใช้เป็นตัวอักษรอยู่ภายในสัญลักษณ์ "<" และ ">" และปิดท้ายด้วยตัวอักษรที่อยู่ระหว่างสัญลักษณ์ "</" และ ">" ตัวอย่างเช่น ถ้าต้องการให้แสดงคำว่า "Welcome" เป็นอักษรตัวหนา (boldface) ก็จะต้องกำหนดแท็กเป็น <B>Welcome</B> เป็นต้น ซึ่งแท็กบางประเภทก็ไม่จำเป็นต้องปิดท้าย เช่น
 ซึ่งหมายถึง Line Break ตัวอย่างเช่น ให้แสดงว่า Good Morning และต่อท้ายด้วย Line Break จะทำได้โดย Good Morning
 เป็นต้น ส่วนแท็กที่เป็นคำอธิบายโปรแกรม (Comments) จะเริ่มต้นด้วย "<!--" และปิดท้ายด้วย "-->" โครงสร้างโดยทั่วไปของไฟล์เอชทีเอ็มแอล

```
<HTML>
<HEAD>
<- - - - - - - - - - - รายละเอียดของ Title - - - - - - >
</TITLE>
<- - - - - - - - - - - Header อื่นๆ - - - - - - - - >
</HEAD>
<BODY>
<- - - - - - - - - - - ส่วนที่เป็นเนื้อหาของเอกสาร - - - - - - - - - >
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
</BODY>
</HTML>
```

ในบางครั้งคำสั่งในเอชทีเอ็มแอลก็ไม่สามารถครอบคลุมการทำงานได้เพียงพอ จึงจำเป็นต้องใช้ภาษาอื่นๆเข้าทำงานร่วมด้วย ซึ่งเรียกว่า ภาษาสคริปต์ (Script Language) ตัวอย่างเช่น จาวาสคริปต์ (JavaScript) วีบีสคริปต์ (VBScript) หรือจาวาแอพเพลต (Java Applet) เป็นต้น

แท็กที่ใช้กำหนดสคริปต์นี้ก็คือ <SCRIPT LANGUAGE> ตัวอย่างเช่น

```
< SCRIPT LANGUAGE=JAVASCRIPT>
```

หมายถึง กำหนดเนื้อที่ในส่วนที่จะเป็นสคริปต์ของจาวาสคริปต์หรือถ้าจาวาแอพเพลตแท็กที่ใช้ในการกำหนด มีตัวอย่างดังนี้

```
<APPLET CODE = "Remote_Access.class" HEIGHT=300 WIDTH=300></APPLET>
```

หมายถึง การกำหนดให้มีการเรียกไฟล์ Remote_Access.class จากเว็บเซิร์ฟเวอร์

2.2.6 จาวาแอพเพลต (Java Applet)

จาวาแอพเพลต คือโปรแกรมประเภทหนึ่งที่เขียนขึ้นมาด้วยภาษาจาวา แต่ต้องทำงานอยู่บนโปรแกรมเว็บบราวเซอร์ ซึ่งต่างจากจาวาแอพพลิเคชั่น (Java Application) ที่สามารถทำงานได้โดยไม่ต้องอาศัยเว็บบราวเซอร์แต่อย่างใด ปกติแล้วแอพเพลตจะถูกจะถูกเก็บไว้ในเครื่องที่ทำหน้าที่เป็นเว็บเซิร์ฟเวอร์ และจะถูกดาวน์โหลดลงสู่เว็บบราวเซอร์ในเครื่องของผู้ใช้ เมื่อผู้ใช้เปิดดูเว็บเพจที่มีการเรียกไปยังแอพเพลตนั้น โดยการที่โปรแกรมจาวาแอพเพลตจะทำงานได้ ในเครื่องของไคลเอนต์ต้องทำการติดตั้ง Java SE Development Kit (JDK) โปรแกรมแอพเพลตจึงจะสารมารถทำงานได้

ข้อดีของแอพเพลต

ข้อดีของแอพเพลตเมื่อเทียบกับจาวาแอพพลิเคชั่น คือหากเราต้องการเปลี่ยนแปลงแก้ไขการทำงานของโปรแกรม จะสามารถทำได้สะดวกกว่า เพราะเพียงแค่แก้ไขตัวแอพเพลตที่เก็บไว้บนเว็บเซิร์ฟเวอร์เพียงที่เดียวเท่านั้น ผู้ใช้งานก็จะได้ใช้โปรแกรมที่ปรับปรุง (Update) แล้ว ไม่เหมือนจาวาแอพพลิเคชั่น ที่หลังจากแก้ไขการทำงานแล้ว เราจะต้องนำโปรแกรมไปติดตั้งลงในเครื่องผู้ใช้งานแต่ละเครื่องเอง เพื่อที่จะให้ผู้ใช้งานทุกคนได้ใช้โปรแกรมที่ปรับปรุงแล้วเหมือนกันหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อเสียของแอพเพลต

เนื่องจากแอพเพลตถูกเก็บอยู่บนเว็บเซิร์ฟเวอร์ จึงต้องใช้เวลาพอสมควรในการรอให้แอพเพลตถูกดาวน์โหลดลงสู่เว็บบราวเซอร์ ซึ่งหากเป็นการเชื่อมต่อเข้าสู่เครือข่ายด้วยความเร็วในการรับส่งข้อมูลต่ำ อย่างเช่น การเชื่อมต่ออินเทอร์เน็ตด้วยความเร็วต่ำๆแล้วอาจต้องรอนานกว่าแอพเพลตจะถูกดาวน์โหลดจนเสร็จสมบูรณ์

2.3 แพลตฟอร์มของระบบ

2.3.1 ระบบปฏิบัติการลินุกซ์

ลินุกซ์ (Linux) คือชื่อของระบบปฏิบัติการและชื่อเคอร์เนลที่นิยมตัวหนึ่งในฐานะของซอฟต์แวร์โอเพนซอร์ส ระบบปฏิบัติการลินุกซ์หรือที่เรียกเต็มๆว่า "กนู/ลินุกซ์" มีลักษณะคล้ายกับระบบปฏิบัติการยูนิกซ์ โดยมีลินุกซ์ เคอร์เนล เป็นศูนย์กลางทำงานร่วมกับไลบรารีและเครื่องมือต่างๆ ลินุกซ์นิยมจำหน่ายหรือแจกฟรีในลักษณะเป็นแพคเกจ โดยผู้จัดทำจะรวมซอฟต์แวร์สำหรับใช้งานในด้านต่างๆ เป็นชุดเข้าด้วยกัน

เริ่มแรกลินุกซ์พัฒนาและใช้งานเฉพาะกลุ่มผู้ที่สนใจ ซึ่งในปัจจุบันลินุกซ์ได้รับความนิยมเนื่องมาจากระบบการทำงานที่เป็นอิสระ ปลอดภัย เชื่อถือได้ และราคาต่ำ จึงได้มีการพัฒนาจากองค์กรต่างๆ เช่น ไอบีเอ็ม ฮิวเลตต์-แพกการ์ด และโนเวลล์ใช้สำหรับในระบบเซิร์ฟเวอร์และพีซี เริ่มแรกลินุกซ์พัฒนาสำหรับใช้กับเครื่อง อินเทล 386 ไมโครโปรเซสเซอร์ หลังจากที่ได้รับความนิยมปัจจุบัน ลินุกซ์ได้พัฒนาให้รับรองการใช้งานของระบบสถาปัตยกรรมคอมพิวเตอร์ในระบบต่างๆ รวมถึงในโทรศัพท์มือถือ และกล้องวีดีโอ

ลิขสิทธิ์ของหลายๆส่วนของ ลินุกซ์จดภายใต้ ลิขสิทธิ์ GPL ซึ่งเป็นลิขสิทธิ์ที่กำหนดให้ผู้ที่นำโค้ดไปใช้ต้องใช้ลิขสิทธิ์แบบเดิมต่อคือใช้ลิขสิทธิ์ GPL เช่นเดียวกัน (ในบางครั้งจะเรียกว่า copyleft)

เหตุที่เลือกใช้ลินุกซ์เนื่องด้วยระบบปฏิบัติการลินุกซ์เป็นโอเพ็นซอร์ส ซึ่งโอเพนซอร์สมีบรรทัดฐานดังนี้

- เผยแพร่ได้อย่างเสรี
- ซอร์สโค้ด จะต้องอนุญาตให้เผยแพร่หรือดาวน์โหลดได้
- งานดัดแปลงได้ แต่มีเงื่อนไขเดียวกับต้นฉบับ
- การคงความสมบูรณ์ในซอร์สโค้ดของผู้เขียน
- ไม่เลือกปฏิบัติเพื่อกีดกันบุคคลหรือกลุ่มใด ๆ
- ไม่เลือกปฏิบัติเพื่อกีดกันกิจการในสาขาใด ๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- การเผยแพร่ของสัญญา
- สัญญาต้องไม่เจาะจงจำเพาะผลิตภัณฑ์อันใดอันหนึ่ง
- สัญญาจะต้องไม่ผูกพันไปถึงซอฟต์แวร์อื่นในสื่อเดียวกัน

อีกทั้งลินุกซ์สามารถนำไปพัฒนาต่อเป็นระบบลินุกซ์ฝังตัวได้ (Linux Embedded System) เพื่อรองรับการทำงานเฉพาะอย่างและลดค่าใช้จ่ายในการพัฒนาอุปกรณ์ทางด้านฮาร์ดแวร์อีกด้วย ซึ่งในอนาคตจะสามารถพัฒนาโครงงานต่อเนื่องเป็นระบบฝังตัวได้อีกด้วย

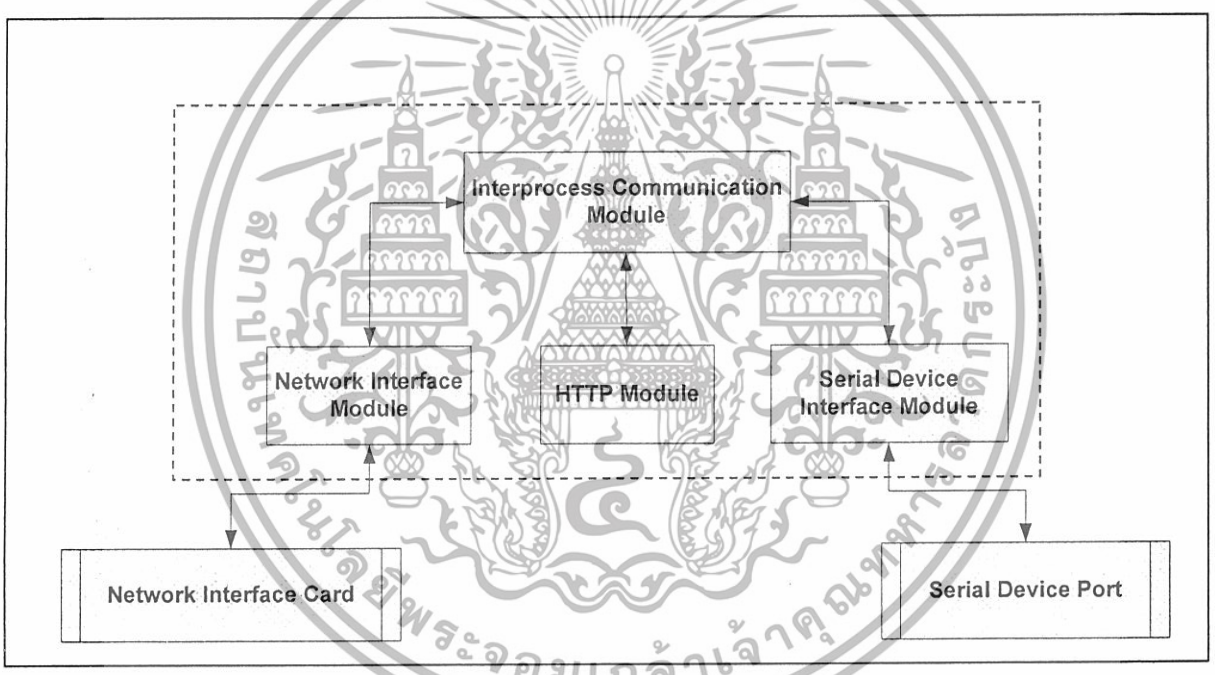
ในการใช้การติดต่อสื่อสารระหว่างโปรเซสแบบ Message queue บนระบบปฏิบัติการลินุกซ์ มันมีข้อจำกัดหลายเรื่อง ได้แก่ ขนาดที่มากที่สุดของบล็อกข้อมูล จำนวนคิวที่มาที่สุด เป็นต้น ซึ่งเนื่องจากการทำงานบางอย่างต้องการลักษณะการใช้งานที่แตกต่างกัน เช่น บางระบบต้องการใช้คิวจำนวนมากกว่าที่เคอร์เนลตั้งไว้ จึงยอมให้ผู้พัฒนาสามารถปรับเปลี่ยนค่าพารามิเตอร์ต่างๆนั้นได้ ใน Configuration file ตามที่อยู่นี้ `/etc/sysctl.conf` โดยทำการเพื่อข้อมูลเข้าไป เช่น `kernel.msgmni = 32767` แล้วจึงทำการเริ่มการทำงานของระบบปฏิบัติการใหม่อีกครั้ง จะช่วยให้สามารถใช้คิวได้จำนวนมากถึง 32767 คิว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
การวิเคราะห์และออกแบบระบบ

3.1 การวิเคราะห์ระบบ

งานทั้งหมดในภาพรวมของโครงงานจำแนกออกเป็น 4 โมดูล เพื่อที่จะเชื่อมโยงการส่งข้อมูลจากเครือข่ายอีเทอร์เน็ตสู่พอร์ตอนุกรม โดยจะแบ่งการทำงานของแต่ละส่วนอย่างชัดเจนดังนี้ Interprocess Communication Module , Network Interface Module , Serial Device Interface Module และ HTTP Module ดังรูปที่ 3.1 ซึ่งมีรายละเอียดดังนี้



รูปที่ 3.1 แสดงโครงสร้างการทำงานของระบบและส่วนที่เกี่ยวข้อง

Interprocess Communication Module

สนับสนุนการทำงานที่ให้สามารถเลือกติดต่อกับอุปกรณ์ปลายทางได้ โดยใช้หมายเลขพอร์ตปลายทางเป็นเงื่อนไขในการเลือก Interprocess Communication Module จะเชื่อมโยงการทำงานระหว่าง Network Interface Module กับ Serial Device Interface Module โดยอาศัยกลไกของ Interprocess Communication เพื่อใช้ในการติดต่อสื่อสารกันระหว่างโปรเซสของ Interprocess Communication Module กับโปรเซสของโมดูลอื่น โดยที่ Interprocess Communication Module จะรอรับข้อมูลที่ส่งจาก Network Interface Module หรือ Serial Device Interface Module เมื่อมีข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถูกส่งเข้ามาจะทำการตรวจสอบว่าข้อมูลนั้นควรจะส่งต่อไปอย่างไร โดยพิจารณาจากหมายเลขพอร์ต และสามารถรองรับการร้องขอการใช้บริการจากหลายไคลเอนต์ได้

Network Interface Module

ภาพรวมของการทำงานทั้งหมดในโมดูลนี้จะทำการพัฒนาโปรแกรมเพื่อทำการเชื่อมต่อการทำงานกับ Network Interface Card ซึ่งจะมีการแบ่งการทำงานย่อยตั้งแต่การสร้างเส้นทางการบริการเพื่อรับส่งข้อมูล การผูกมัดเพื่อเลือกพอร์ตเพื่อที่จะใช้เป็นช่องทางในการรับส่งข้อมูล การทำตัวเป็นเซิร์ฟเวอร์เพื่อรอรับการเชื่อมต่อจะผู้ใช้งาน การยอมรับการเชื่อมต่อให้ผู้ใช้งานสามารถใช้งานได้ ไปจนถึงการรับส่งข้อมูลต่างๆ เพื่อที่จะทำการส่งต่อข้อมูลและการทำงานไปยัง Interprocess Communication Module

Serial Device Interface Module

งานทั้งหมดในภาพรวมของโมดูลนี้จะทำการเขียนโปรแกรมในการติดต่อกับพอร์ตอนุกรม โดยทำการรับข้อมูลจาก Interprocess Communication Module และทำการส่งข้อมูลออกทางพอร์ตอนุกรมไปยังอุปกรณ์ที่เชื่อมต่ออยู่ได้ และทำการรับข้อมูลจากอุปกรณ์ผ่านทางพอร์ตอนุกรมแล้วทำการส่งไปยัง Interprocess Communication Module ได้

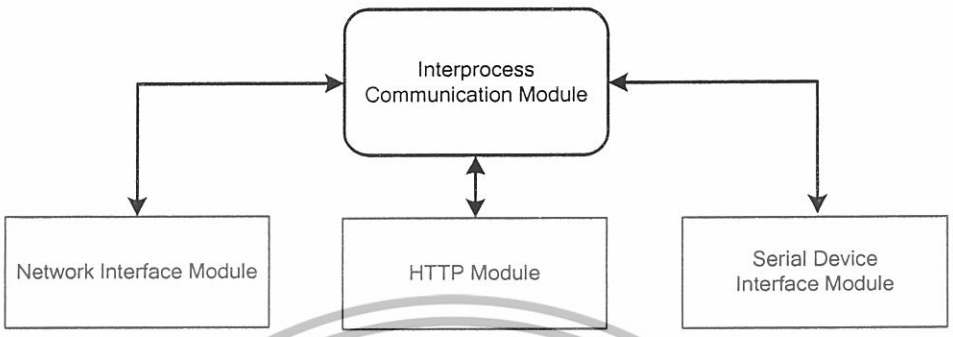
HTTP Module

เป็นส่วนที่ทำหน้าที่สนับสนุนการติดต่อที่มาจากเครื่องปลายทางโดยใช้เว็บบราวเซอร์ โดยลักษณะการทำงานของโมดูลนี้จะทำงานคล้ายกับเว็บเซิร์ฟเวอร์ ติดต่อกันโดยใช้เอชทีทีพีโพรโทคอล HTTP Module ซึ่งจะรอรับข้อมูลที่ Interprocess Communication Module พิจารณาแล้วว่าเป็น HTTP Request จากนั้นทำการตอบ HTTP response ที่ประกอบด้วยข้อมูลที่เว็บบราวเซอร์ต้องการกลับไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 การออกแบบระบบงานโดยใช้วิธี Processing Modeling

3.2.1 แผนภาพบริบท (Context Diagram)



รูปที่ 3.2 Context Diagram ของระบบ

จากรูปที่ 3.2 จะเห็นว่าส่วนต่างๆจะทำงานดังนี้

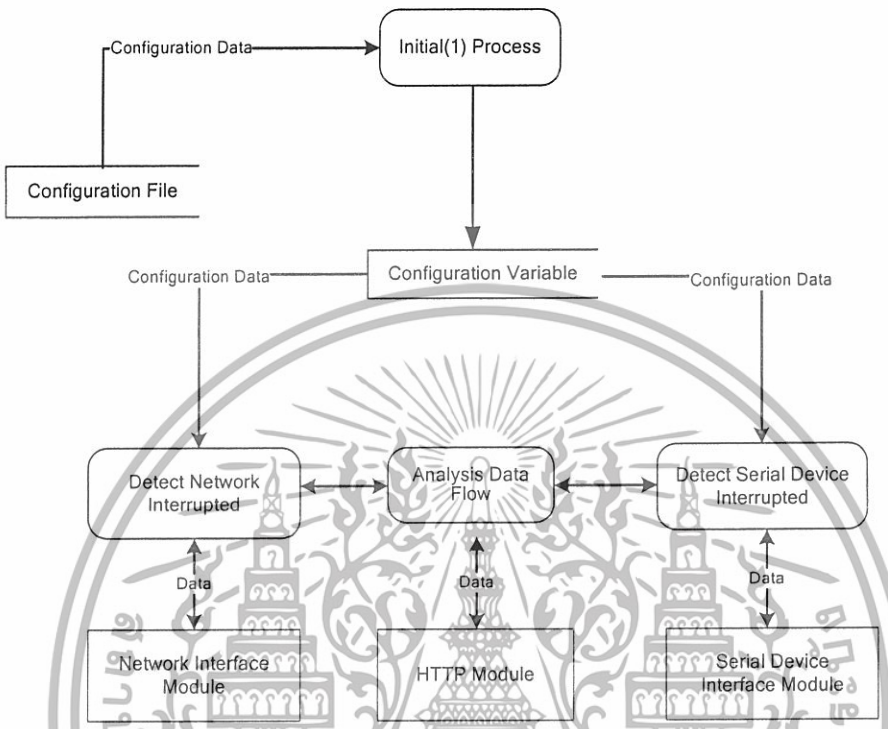
การทำงานของ Interprocess Communication Module จะทำงานร่วมกัน 3 โมดูลคือ

- Network Interface Module จะทำการรองรับการติดต่อสื่อสารเพื่อรับส่งข้อมูลจากเครือข่าย
- HTTP Module จะทำการรองรับการติดต่อสื่อสารที่มาจากเว็บบราวเซอร์
- Serial Device Interface Module จะทำการรองรับการติดต่อสื่อสารเพื่อรับส่งข้อมูลกับอุปกรณ์มาตรฐาน RS 232

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.2 แผนภาพกระแสข้อมูล (Data Flow Diagram)

การไหลของข้อมูลในภาพรวมของระบบ



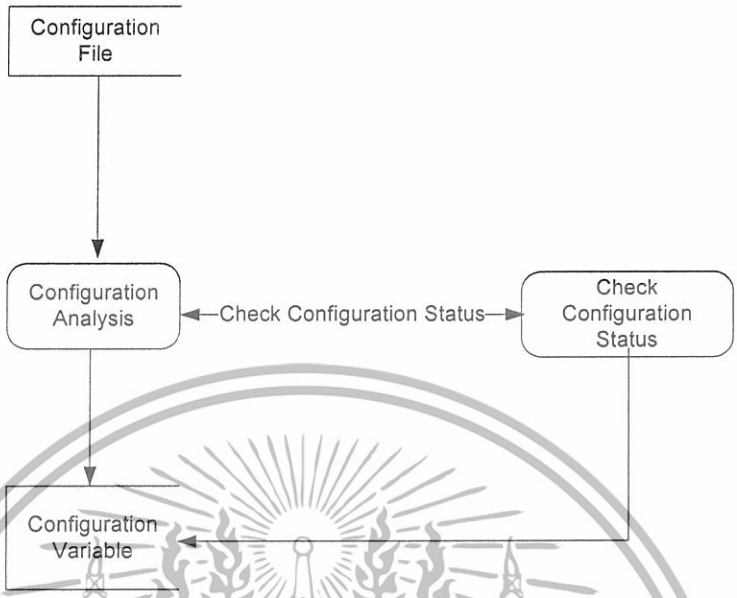
รูปที่ 3.3 แผนผังการไหลของข้อมูลในภาพรวมของระบบ

จากรูปที่ 3.3 แสดงถึงการไหลของข้อมูลในภาพรวมของระบบ จะเห็นว่าส่วนต่างๆจะทำงานดังนี้

- Initial Process (1) เป็นส่วนที่กำหนดค่าเริ่มต้นของระบบ โดยอ่านค่าเริ่มต้นการทำงานจาก Configuration file และทำการตรวจสอบสถานะของอุปกรณ์ที่เชื่อมต่อว่ารองรับกับการกำหนดค่าเริ่มต้นของระบบอยู่
- Detect Network Interrupted เป็นส่วนที่รองรับการติดต่อขอใช้งานผ่าน Network Interface Module
- Detect Serial Device Interrupted เป็นส่วนที่รองรับการติดต่อขอใช้งานผ่านพอร์ตอนุกรม
- Analysis Data Flow เป็นส่วนวิเคราะห์ การรับการส่งข้อมูลจากจุดหนึ่งไปยังอีกจุดหนึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การกำหนดค่าเริ่มต้นและวิเคราะห์ค่าการทำงาน



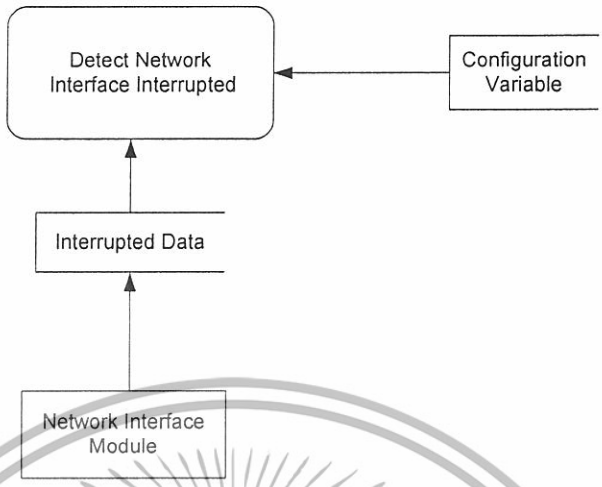
รูปที่ 3.4 แผนผังการกำหนดค่าเริ่มต้นและวิเคราะห์ค่าการทำงาน

จากรูปที่ 3.4 แสดงถึงการกำหนดค่าเริ่มต้นและวิเคราะห์ค่าการทำงาน จะเห็นว่าส่วนต่างๆ ทำงานดังนี้

- Configuration Analysis เป็นส่วนที่อ่านข้อมูลจาก Configuration file แล้วกำหนดค่าเริ่มต้นให้กับระบบและทำการส่งค่าไปเพื่อทำการตรวจสอบสถานะของอุปกรณ์
- Check Configuration status เป็นส่วนที่ใช้ในการตรวจสอบสถานะและความพร้อมในการทำงานของอุปกรณ์ต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การไหลของข้อมูลขณะที่ Network Interface Module ถูกขัดจังหวะ

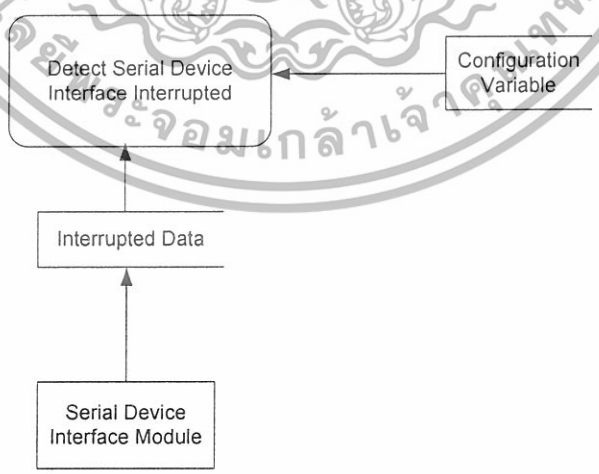


รูปที่ 3.5 แผนผังการไหลของข้อมูลขณะที่ Network Interface Module ถูกขัดจังหวะ

จากรูปที่ 3.5 แสดงถึงการไหลของข้อมูลขณะที่ Network Interface Module ถูกขัดจังหวะ จะเห็นว่าส่วนต่างๆทำงานดังนี้

- Detect Network Interface Interrupted ทำหน้าที่ในการรอรับการขัดจังหวะจาก Network Interface Module โดยมีการกำหนดการทำงาน

การไหลของข้อมูลขณะที่ Serial Device Interface Module ถูกขัดจังหวะ



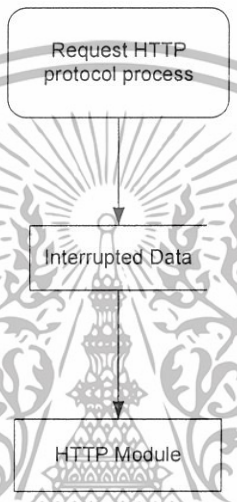
รูปที่ 3.6 แผนผังการไหลของข้อมูลขณะที่ Serial Device Interface Module ถูกขัดจังหวะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 3.6 แสดงถึงการไหลของข้อมูลขณะที่ Serial Device Interface Module ถูกขัดจังหวะ จะเห็นว่าส่วนต่างๆทำงานดังนี้

- Detect Serial Device Interface Interrupted ทำหน้าที่ในการรอรับการขัดจังหวะจาก Serial Device Network Interface Module โดยมีการกำหนดการทำงาน

การไหลของข้อมูลขณะร้องขอบริการจาก HTTP Module



รูปที่ 3.7 แผนผังการไหลของข้อมูลขณะร้องขอบริการจาก HTTP Module

จากรูปที่ 3.7 แสดงถึงการไหลของข้อมูลขณะร้องขอบริการจาก HTTP Module จะเห็นว่าส่วนต่างๆทำงานดังนี้

- Request HTTP protocol process ทำการร้องขอการประมวลผล HTTP request ที่ได้รับมาจากเว็บบราวเซอร์ เพื่อต้องการ HTTP response ตอบไปยังเว็บบราวเซอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การพัฒนาระบบ

จากการออกแบบระบบ การทำงานของระบบ ถูกแบ่งออกเป็น 4 ส่วน โดยแต่ละส่วนมีการแยกส่วนการทำงานอย่างชัดเจน ดังต่อไปนี้

4.1 Interprocess Communication Module

สนับสนุนการทำงานที่ให้สามารถเลือกติดต่อกับอุปกรณ์ปลายทางได้ โดยใช้หมายเลขพอร์ตปลายทางเป็นเงื่อนไขในการเลือก ผู้ดูแลระบบสามารถกำหนดลักษณะการทำงานของระบบได้ โดยการสร้างตารางกำหนดเส้นทางผ่านทาง Configuration file

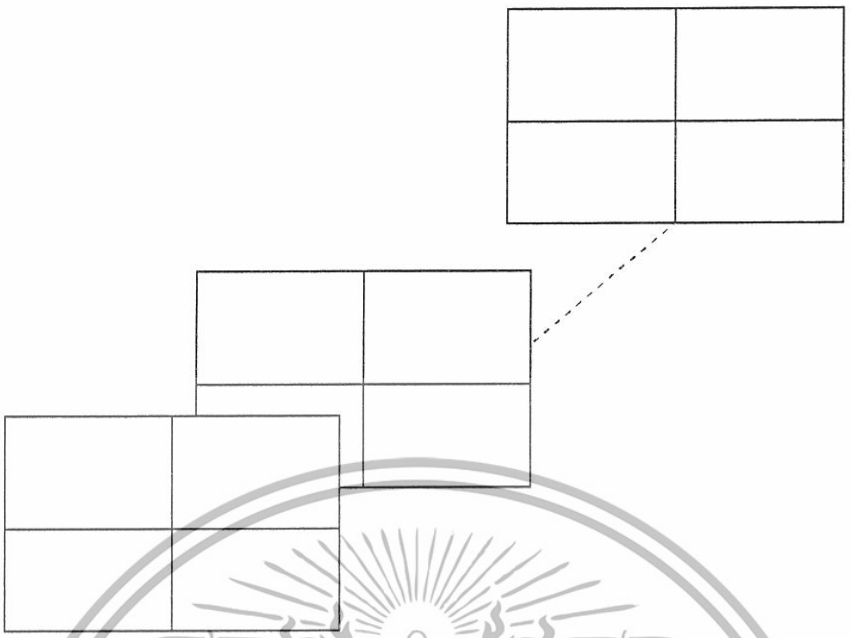
Port Timeout - Destination(Serial port | IpAddress:port)

- Port หมายเลขพอร์ตที่ต้องการเปิดเพื่อรอรับการติดต่อจากไคลเอนต์
- Timeout เมื่อช่องทางการติดต่อสื่อสาร ไม่มีการรับส่งข้อมูลจากผู้ใช้เป็นเวลา Timeout วินาที Network Interface Module จะทำการตัดการติดต่อสื่อสารนั้น
- Destination เมื่อมีข้อมูลถูกส่งมาทาง Port จะถูกส่งต่อมายัง Destination ซึ่งอาจจะเป็น Serial Device Interface Module หรือ Network Interface Module ก็ได้

Interprocess Communication Module จะสร้างคิวเพื่อใช้ในการติดต่อสื่อสารกับโมดูลอื่นตามตารางกำหนดเส้นทางที่ผู้ดูแลระบบได้กำหนดไว้ใน Configuration file พร้อมทั้งสร้างตารางการเชื่อมโยงกันระหว่างคิว (Queue Mapping Table) ขึ้น เพื่อใช้ในการพิจารณาว่าข้อมูลที่ถูกส่งมาจากแต่ละช่องทางควรส่งต่อไปยังช่องทางใด

ตารางการเชื่อมโยงกันระหว่างคิวมีลักษณะเป็นตาราง 3 มิติ เก็บหมายเลขอ้างอิงคิว (Qid) ที่ได้หลังจากการสร้างคิวเพื่อใช้ในการติดต่อสื่อสารกับโมดูลอื่นๆ เพื่อสร้างความสัมพันธ์ขึ้นระหว่างคู่โมดูลที่ติดต่อกันผ่านทาง Interprocess Communication Module โดยทำหน้าที่บอกว่า เมื่อมีโมดูลหนึ่งส่งข้อมูลเข้ามาทางคิวหนึ่งให้ส่งข้อมูลนั้นไปยังอีกคิวหนึ่งที่ใช้ติดต่อกับอีกโมดูลหนึ่ง โดยในการรับส่งข้อมูลกันระหว่างโมดูลผ่านคิวแต่ละครั้งนั้น จะใช้หมายเลขอ้างอิงคิวเพื่อระบุคิวตัวที่จะใช้รับส่งข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.1 แสดงลักษณะตารางการเชื่อมโยงกันระหว่างคิว

- แต่ละระนาบ หมายถึง เส้นทางทั้งขาไปและขากลับที่แต่ละคู่โมดูลใช้ติดต่อสื่อสารกันผ่านทาง Interprocess Communication Module
- แต่ละหลัก หมายถึง คู่ของช่องทางอินพุตและเอาต์พุตที่ Interprocess Communication Module ใช้ในการติดต่อกับโมดูลข้างใดข้างหนึ่ง
- แต่ละแถว หมายถึง เส้นทางขาไปหรือขากลับที่ใช้ติดต่อกันระหว่างคู่โมดูลผ่านทาง Interprocess Communication Module เพียงขาเดียว

เมื่อสร้างคิวที่ใช้ติดต่อกับทุกโมดูลเรียบร้อยแล้วแล้ว Interprocess Communication Module จะรอรับข้อมูลจากคิวที่เป็นช่องทางอินพุตของ Interprocess Communication Module เมื่อมีข้อมูลถูกส่งมาจะส่งต่อไปยังคิวที่กำหนดตามตารางการเชื่อมโยงกันระหว่างคิว แต่หาก Interprocess Communication Module พิจารณาแล้วว่าข้อมูลที่รับมาเป็น HTTP request จะส่งต่อไปยัง HTTP Module โดยก่อนที่จะส่งไปนั้น Interprocess Communication Module จะพิจารณาจากตารางการเชื่อมโยงกันระหว่างคิวก่อนว่าช่องทางที่ส่งเข้ามานั้น ช่องทางขากลับที่เป็นคู่กันคือช่องทางใด แล้วจะทำการแนบหมายเลขอ้างอิงคิวที่ใช้อ้างถึงคิวนั้นส่งไปพร้อมข้อมูลให้กับ HTTP Module ไปประมวลผลต่อ เมื่อ HTTP Module ตอบกลับมาจะทำการดูว่าเป็นข้อมูล HTTP response นี้ต้องส่งไปยังช่องทางขากลับช่องทางใด โดยดูจากหมายเลขอ้างอิงคิวที่แนบมาพร้อมกับข้อมูลแล้วจึงใช้อ้างถึงคิวในการส่ง HTTP response ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เหตุผลที่สร้างตารางการเชื่อมโยงกันระหว่างคิวเป็นตาราง 3 มิติ

1. เมื่อข้อมูลที่ส่งมายัง Interprocess Communication Module เป็น HTTP request จะส่งไปให้ HTTP Module ประมวลผล เมื่อประมวลผลเสร็จจะส่ง HTTP response กลับมาให้ Interprocess Communication Module จำเป็นต้องตอบ HTTP response นั้นกลับไปยังต้นทางที่ส่งมา ดังนั้นเราจึงต้องรู้ว่าช่องทางอินพุตแต่ละช่องทางเป็นคู่กับช่องทางเอาต์พุตช่องทางใด เนื่องจากช่องทางที่ HTTP Module ตอบกลับ HTTP response นั้น เป็นช่องทางเดิมตลอด เราจึงไม่สามารถใช้ตารางการเชื่อมโยงกันระหว่างคิวเทียบได้ว่า ถ้ามาทางช่องหนึ่งให้ส่งไปที่อีกช่องทางหนึ่ง ดังนั้นเราจึงต้องสร้างความสัมพันธ์ระหว่างช่องทางอินพุตกับเอาต์พุตของแต่ละโมดูล
2. เพื่อสร้างความสัมพันธ์ระหว่างคู่โมดูลที่ติดต่อสื่อสารกันผ่าน Interprocess Communication Module โดยการเชื่อมช่องทางการติดต่อเข้าด้วยกัน เพื่อให้สามารถตรวจสอบได้ว่าเมื่อ Network Interface Module ส่งข้อมูลมาจะส่งต่อไปไปยังช่องทางใด และในขณะเดียวกันเมื่อมีข้อมูลถูกตอบกลับจาก Serial Device Interface Module หรือ Network Interface Module จะส่งต่อไปยังช่องทางใด

ตัวอย่างการทำงานของ Interprocess Communication Module

ตารางกำหนดเส้นทางที่ผู้ดูแลกำหนดใน Configuration file

8000 60	-	COM1
8001 60	-	COM2
9000 200	-	10.1.1.2:8000

ความหมายคือ

1. เมื่อมีข้อมูลถูกส่งมาทางพอร์ต 8000 ให้ส่งต่อไปที่พอร์ตอนุกรม COM1 โดยการติดต่อสื่อสารที่เชื่อมต่ออยู่ เมื่อไม่มีการรับส่งข้อมูลเป็นเวลา 60 วินาที จะตัดการติดต่อสื่อสารไป
2. เมื่อมีข้อมูลถูกส่งมาทางพอร์ต 8001 ให้ส่งต่อไปที่พอร์ตอนุกรม COM2 โดยการติดต่อสื่อสารที่เชื่อมต่ออยู่ เมื่อไม่มีการรับส่งข้อมูลเป็นเวลา 60 วินาที จะตัดการติดต่อสื่อสารไป
3. เมื่อมีข้อมูลถูกส่งมาทางพอร์ต 9000 ให้ส่งต่อไปที่ Network Interface Module ที่ต่อกับเทอร์มินัลเซิร์ฟเวอร์ที่มีไอพี 10.1.1.2 ทางพอร์ต 8000 โดยการติดต่อสื่อสารที่เชื่อมต่ออยู่ เมื่อไม่มีการรับส่งข้อมูลเป็นเวลา 200 วินาที จะตัดการติดต่อสื่อสารไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตารางกำหนดเส้นทาง การสร้างคิวเพื่อติดต่อสื่อสารกับโมดูลอื่นของ Interprocess Communication Module มีขั้นตอนดังนี้

1. Interprocess Communication Module สร้างคิวที่ใช้ติดต่อกับ Network Interface Module ด้วยคีย์ที่เท่ากับ 110 และ 120 (สร้างขึ้นเอง) จะได้หมายเลขอ้างอิงคิวเท่ากับ 100110 และ 100120 (เลขที่สมมติขึ้น) ที่เคอร์เนลสร้างให้ จึงนำมาเก็บไว้ในตารางการเชื่อมโยงกันระหว่างคิวในแนวหลัก (2 คิวที่ Interprocess Communication Module ใช้ติดต่อกับโมดูลข้างใดข้างหนึ่ง) จากนั้นเรียก Network Interface Module ให้ทำงานรอรับข้อมูลที่พอร์ต 8000 พร้อมทั้งส่งค่าคีย์ที่ Interprocess Communication Module ใช้ในการสร้างคิวไปให้กับ Network Interface Module เพื่อใช้ในการหาหมายเลขอ้างอิงคิวที่ใช้ในการติดต่อสื่อสารกับ Interprocess Communication Module และกำหนดให้ Network Interface Module ทำการตัดการติดต่อสื่อสารที่ไม่มีการรับส่งข้อมูลนานกว่า 60 วินาที

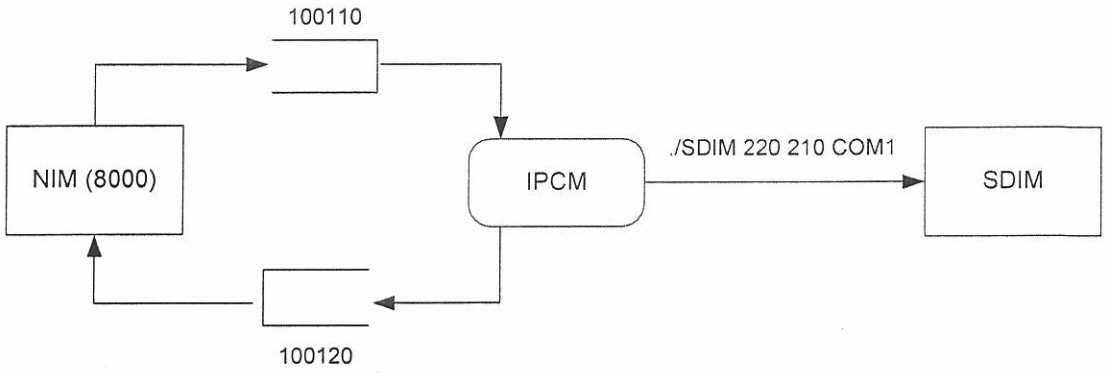
รูปที่ 4.2 ตัวอย่าง Interprocess Communication Module เรียก Network Interface Module ทำงานที่พอร์ต 8000

ตารางที่ 4.1 ตัวอย่างตารางการเชื่อมโยงกันระหว่างคิวหลังจากสร้างคิวที่ใช้ติดต่อสื่อสารกับ Network Interface Module ที่รอรับข้อมูลที่พอร์ต 8000

100110	
100120	

2. Interprocess Communication Module สร้างคิวที่ใช้ติดต่อกับ Serial Device Interface Module ด้วยคีย์ที่เท่ากับ 210 และ 220 จะได้หมายเลขอ้างอิงคิวเท่ากับ 100210 และ 100220 ที่เคอร์เนลสร้างให้ จึงนำมาเก็บไว้ในตารางการเชื่อมโยงกันระหว่างคิวในแนวหลัก จากนั้นเรียก Serial Device Interface Module ให้ทำงานควบคุมพอร์ตอนุกรม COM1 พร้อมทั้งส่งค่าคีย์ที่ Interprocess Communication Module ใช้ในการสร้างคิวไปให้กับ Serial Device Interface Module เพื่อใช้ในการหาหมายเลขอ้างอิงคิวที่ใช้ในการติดต่อสื่อสารกับ Interprocess Communication Module ด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



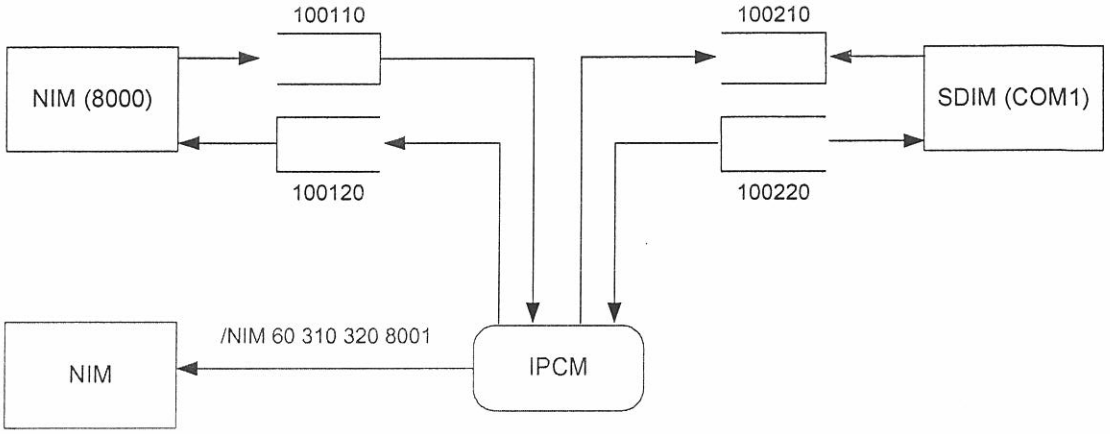
รูปที่ 4.3 ตัวอย่าง Interprocess Communication Module เรียก Serial Device Interface Module ทำงานที่พอร์ตอนุกรม COM1

ตารางที่ 4.2 ตัวอย่างตารางการเชื่อมโยงกันระหว่างคิวหลังจากสร้างคิวที่ใช้ติดต่อสื่อสารกับ Serial Device Interface Module ที่ควบคุมพอร์ตอนุกรม COM1

100110	**100210**
100120	**100220**

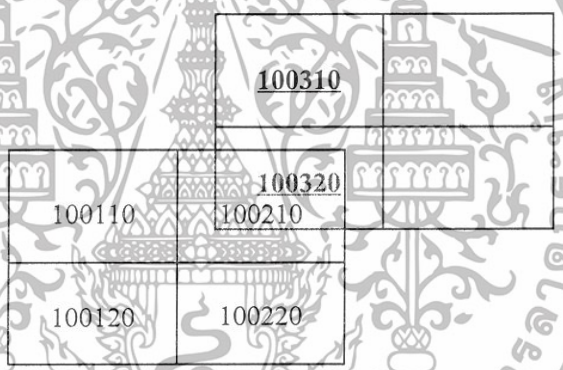
3. Interprocess Communication Module สร้างคิวที่ใช้ติดต่อกับ Network Interface Module ด้วยคีย์ที่เท่ากับ 310 และ 320 จะได้หมายเลขอ้างอิงคิวเท่ากับ 100310 และ 100320 ที่เคอร์เนลสร้างให้ จึงนำมาเก็บไว้ในตารางการเชื่อมโยงกันระหว่างคิวในแนวหลัก จากนั้นเรียก Network Interface Module ให้ทำงานรอรับข้อมูลที่พอร์ต 8001 พร้อมทั้งส่งค่าคีย์ที่ Interprocess Communication Module ใช้ในการสร้างคิวไปให้กับ Network Interface Module เพื่อใช้ในการหาหมายเลขอ้างอิงคิวที่ใช้ในการติดต่อสื่อสารกับ Interprocess Communication Module และและกำหนดให้ Network Interface Module ทำการตัดการติดต่อสื่อสารที่ไม่มีการรับส่งข้อมูลนานกว่า 60 วินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



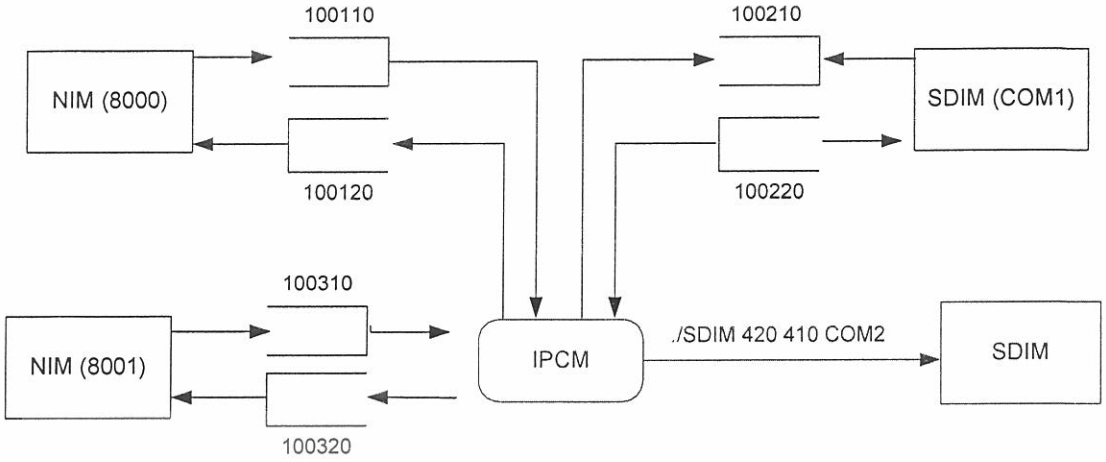
รูปที่ 4.4 ตัวอย่าง Interprocess Communication Module เรียก Network Interface Module ทำงานที่พอร์ต 8001

ตารางที่ 4.3 ตัวอย่างตารางการเชื่อมโยงกันระหว่างคิวหลังจากที่สร้างคิวที่ใช้ติดต่อสื่อสารกับ Network Interface Module ที่รอรับข้อมูลที่พอร์ต 8001



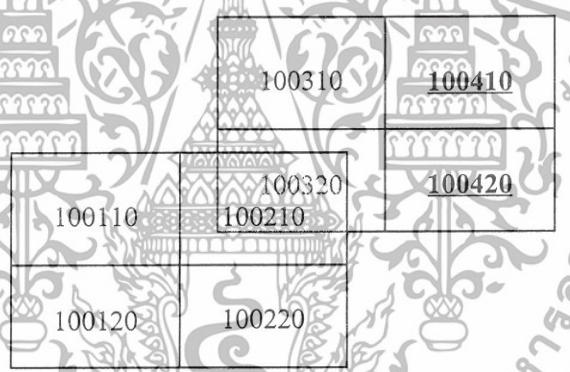
4. Interprocess Communication Module สร้างคิวที่ใช้ติดต่อกับ Serial Device Interface Module ด้วยคีย์ที่เท่ากับ 410 และ 420 จะได้หมายเลขอ้างอิงคิวเท่ากับ 100410 และ 100420 ที่เคอร์เนลสร้างให้ จึงนำมาเก็บไว้ในตารางการเชื่อมโยงกันระหว่างคิวในแนวหลัก จากนั้นเรียก Serial Device Interface Module ให้ทำงานควบคุมพอร์ตอนุกรม COM2พร้อมทั้งส่งค่าคีย์ที่ Interprocess Communication Module ใช้ในการสร้างคิวไปให้กับ Serial Device Interface Module เพื่อใช้ในการหาหมายเลขอ้างอิงคิวที่ใช้ในการติดต่อสื่อสารกับ Interprocess Communication Module ด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



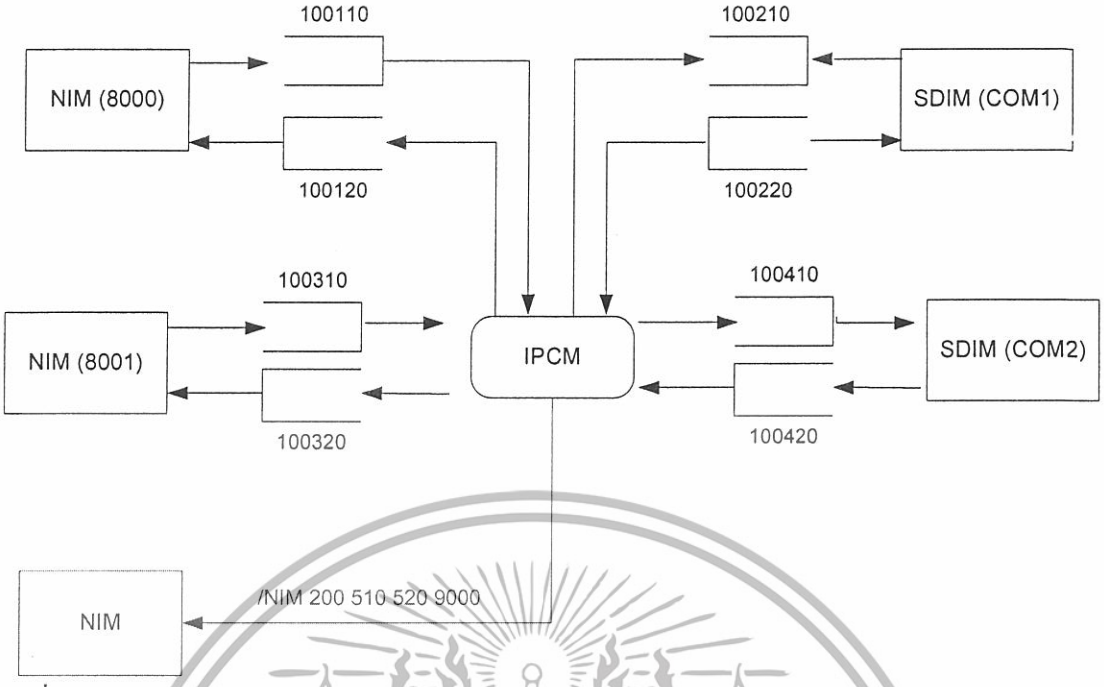
รูปที่ 4.5 ตัวอย่าง Interprocess Communication Module เรียก Serial Device Interface Module ทำงานที่พอร์ตอนุกรม COM2

ตารางที่ 4.4 ตัวอย่างตารางการเชื่อมโยงกันระหว่างคิวหลังจากสร้างคิวที่ใช้ติดต่อสื่อสารกับ Serial Device Interface Module ที่ควบคุมพอร์ตอนุกรม COM2



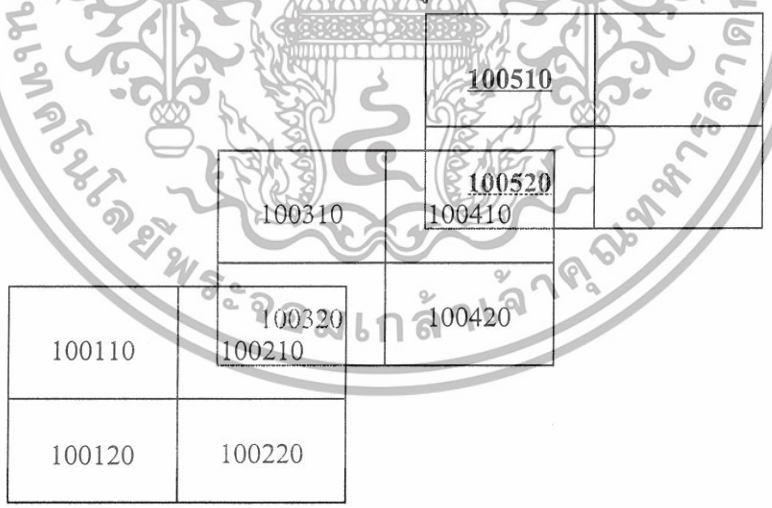
5. Interprocess Communication Module สร้างคิวที่ใช้ติดต่อกับ Network Interface Module ด้วยคีย์ที่เท่ากับ 510 และ 520 จะได้หมายเลขอ้างอิงคิวเท่ากับ 100510 และ 100520 ที่เคอร์เนลสร้างให้ จึงนำมาเก็บไว้ในตารางการเชื่อมโยงกันระหว่างคิวในแนวหลัก จากนั้นเรียก Network Interface Module ให้ทำงานรอรับข้อมูลที่พอร์ต 9000 พร้อมทั้งส่งค่าคีย์ที่ Interprocess Communication Module ใช้ในการสร้างคิวไปให้กับ Network Interface Module เพื่อใช้ในการหาหมายเลขอ้างอิงคิวที่ใช้ในการติดต่อสื่อสารกับ Interprocess Communication Module และและกำหนดให้ Network Interface Module ทำการตัดการติดต่อสื่อสารที่ไม่มีการรับส่งข้อมูลนานกว่า 200 วินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



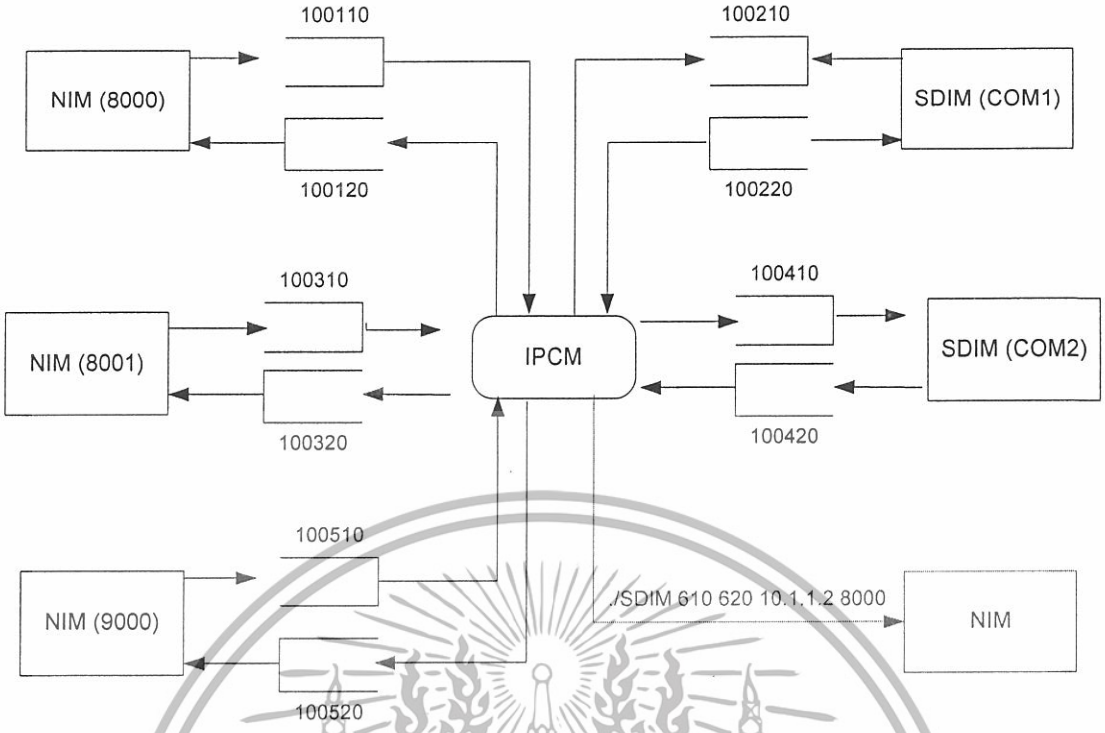
รูปที่ 4.6 ตัวอย่าง Interprocess Communication Module เรียก Network Interface Module ทำงานที่พอร์ต 9000

ตารางที่ 4.5 ตัวอย่างตารางการเชื่อมโยงกันระหว่างคิวหลังจากสร้างคิวที่ใช้ติดต่อสื่อสารกับ Network Interface Module ที่รอรับข้อมูลที่พอร์ต 9000



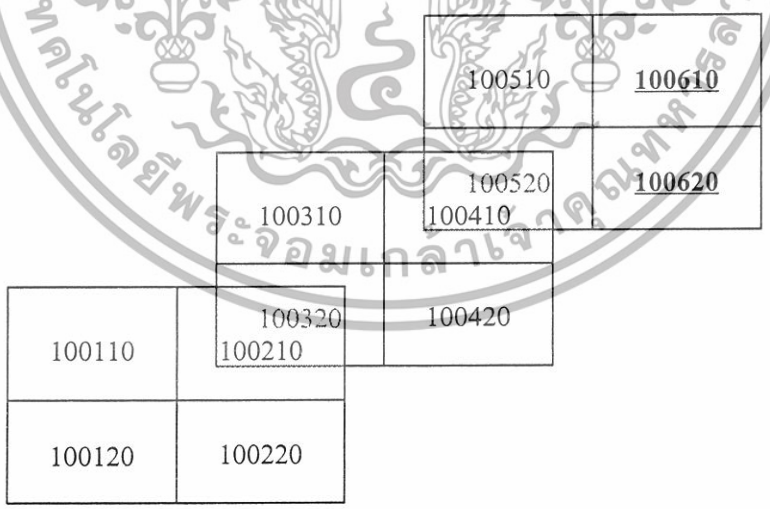
6. Interprocess Communication Module สร้างคิวที่ใช้ในการติดต่อสื่อสารกับ Network Interface Module โดยใช้คีย์เท่ากับ 610 และ 620 จะได้หมายเลขอ้างอิงคิวที่เคอร์เนลสร้างให้ แล้วจึงบันทึกลงตารางการเชื่อมโยงกันระหว่างคิว แล้วจึงไปเรียก Network Interface Module ทำงานเชื่อมต่อกับอีกเทอร์มินัลเซิร์เวอร์หนึ่งที่ไอพี 10.1.1.2 ทางพอร์ต 8500 พร้อมทั้งส่งค่าคีย์เพื่อใช้ในการหาหมายเลขอ้างอิงด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

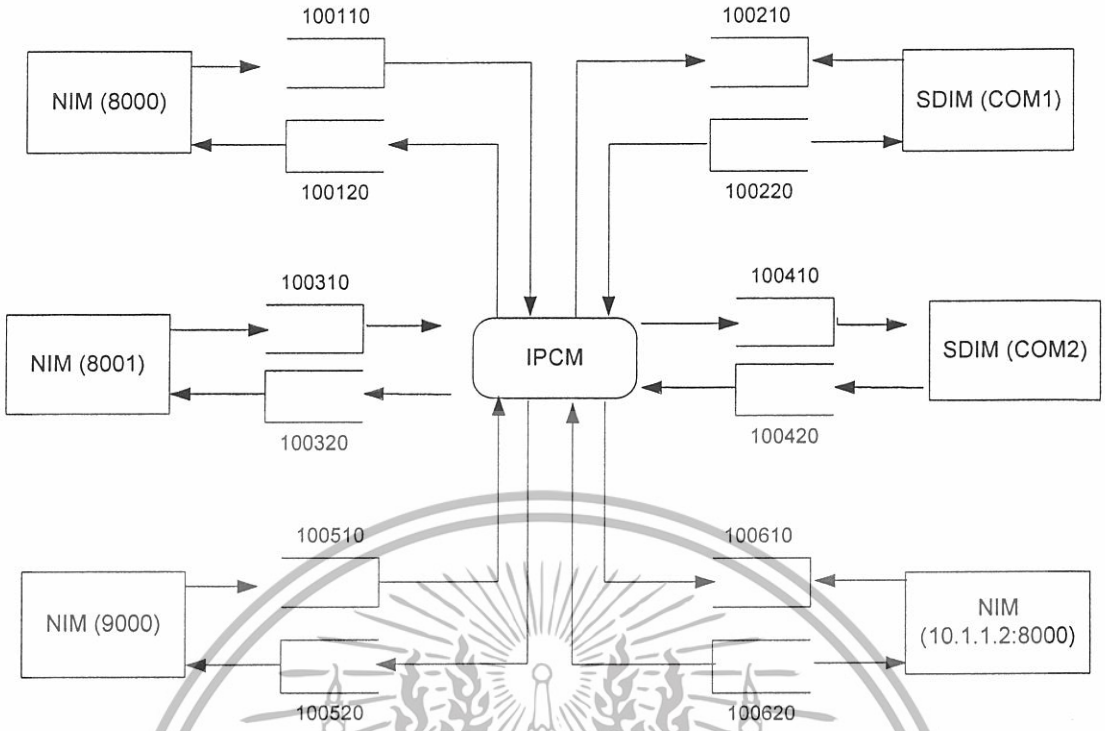


รูปที่ 4.7 ตัวอย่าง Interprocess Communication Module เรียก Network Interface Module ทำงานโดยทำการเชื่อมต่อกับเทอร์มินัลเซิร์ฟเวอร์อื่น

ตารางที่ 4.6 ตัวอย่างตารางการเชื่อมโยงกันระหว่างคิวหลังจากสร้างคิวที่ใช้ติดต่อกับ Network Interface Module ที่เชื่อมต่อกับเทอร์มินัลเซิร์ฟเวอร์อื่น



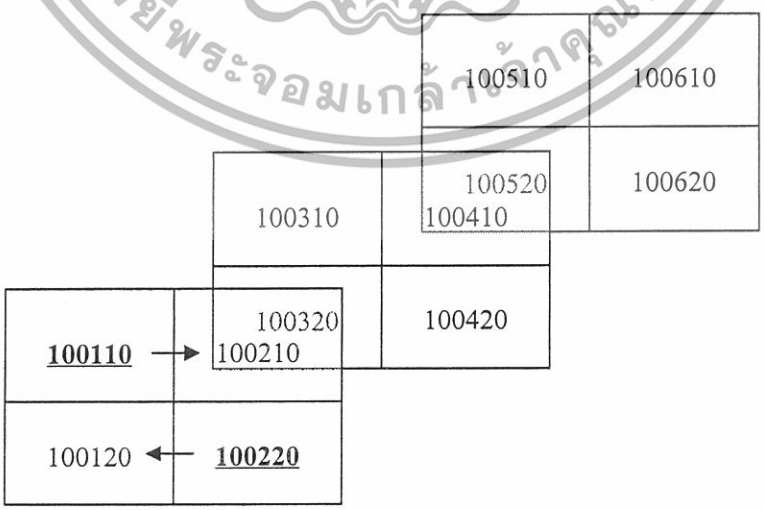
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.8 ตัวอย่างโครงสร้างของเทอร์มินัลเซิร์ฟเวอร์หลังจากสร้างเส้นทางการติดต่อเรียบร้อยแล้ว

หลังจากสร้างเส้นทางการติดต่อเรียบร้อยแล้ว Interprocess Communication Module จะทำการรอรับข้อมูลที่ถูกส่งมาทางคิว โดยตรวจสอบว่าในคิวที่เป็นช่องทางอินพุตของ Interprocess Communication Module มีข้อมูลถูกส่งมาหรือไม่

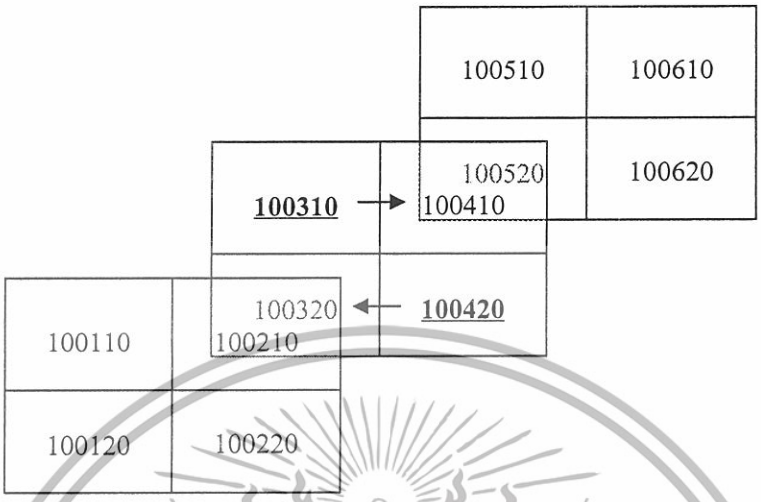
เริ่มที่เส้นทางที่ 1 ตรวจสอบว่าในคิวที่มีหมายเลขอ้างอิงคิวเท่ากับ 100110 และ 100220 มีข้อมูลถูกส่งเข้ามาหรือไม่ ถ้ามีจะส่งข้อมูลต่อไปยังคิวที่มีหมายเลขอ้างอิงคิวเท่ากับ 100210 และ 100120 ตามลำดับ



รูปที่ 4.9 ตัวอย่างลักษณะการตรวจสอบข้อมูลในเส้นทางที่ 1

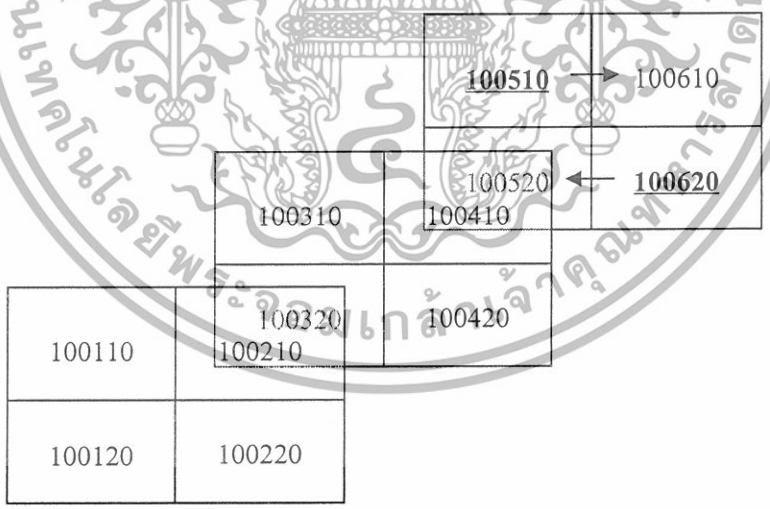
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากนั้นดูที่เส้นทางที่ 2 ตรวจสอบว่าในคิวที่มีหมายเลขอ้างอิงคิวเท่ากับ 100310 และ 100420 มีข้อมูลถูกส่งเข้ามาหรือไม่ ถ้ามีจะส่งข้อมูลต่อไปยังคิวที่หมายเลขอ้างอิงคิวเท่ากับ 100410 และ 100320 ตามลำดับ



รูปที่ 4.10 ตัวอย่างลักษณะการตรวจสอบข้อมูลในเส้นทางที่ 2

จากนั้นดูที่เส้นทางที่ 3 ตรวจสอบว่าในคิวที่มีหมายเลขอ้างอิงคิวเท่ากับ 100510 และ 100620 มีข้อมูลถูกส่งเข้ามาหรือไม่ ถ้ามีจะส่งข้อมูลต่อไปยังคิวที่มีหมายเลขอ้างอิงคิวเท่ากับ 100610 และ 100520 ตามลำดับ



รูปที่ 4.11 ตัวอย่างลักษณะการตรวจสอบข้อมูลในเส้นทางที่ 3

เมื่อครบทุกเส้นทางแล้วจะทำการกลับมาตรวจสอบที่ช่องทางที่ 1 ใหม่เป็นเช่นนี้ไปเรื่อยๆ แต่หาก Interprocess Communication Module พิจารณาแล้วว่า ข้อมูลที่ส่งมาจากช่องทางที่มีหมายเลขอ้างอิงคิวเท่ากับ 100310 เป็น HTTP request ก่อนที่จะส่งต่อให้ HTTP Module Interprocess Communication Module จะพิจารณาจากตารางการเชื่อมโยงกันระหว่างคิวว่าช่องทาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ที่ส่งเข้ามาช่องทางขากลับที่เป็นคู่กันคือช่องทางใด ในที่นี้ถ้าข้อมูลเข้ามาทาง 100310 ช่องทางขากลับที่เป็นคู่กัน คือ 100320 ก็จะแนบไปพร้อมกับข้อมูลที่จะส่งไปให้ HTTP Module เมื่อ HTTP Module ส่ง HTTP response มา Interprocess Communication Module จะดูว่าควรจะส่งต่อไปยังช่องทางใด โดยดูจากหมายเลขอ้างอิงคิวที่แนบมากับ HTTP response ที่ถูกตอบกลับมา นั่นคือ 100320 ก็จะส่งออกทางช่องทางนั้นต่อไป

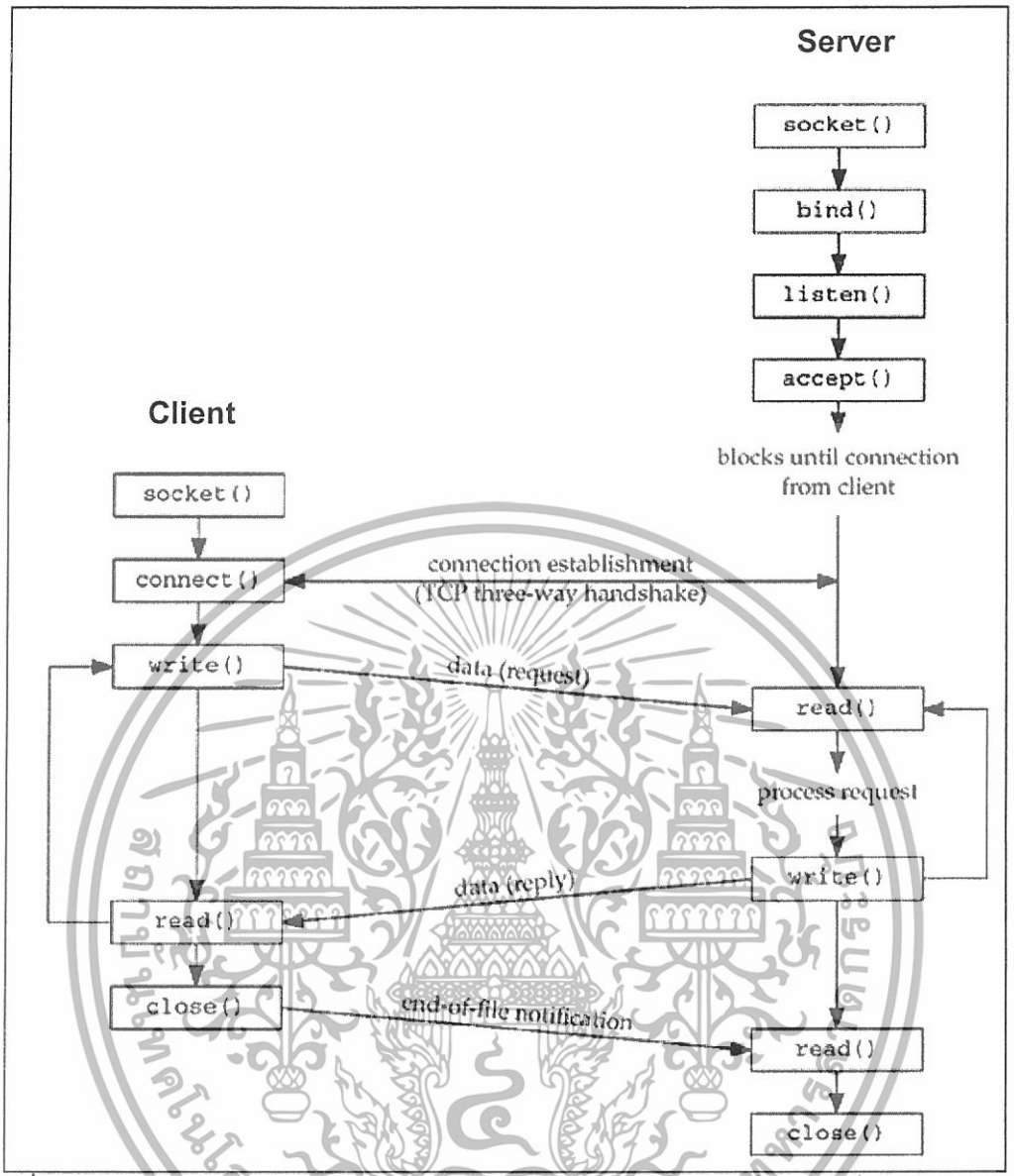
4.2 Network Interface Module

เป็นการเขียนโปรแกรมเครือข่าย เพื่อรองรับการติดต่อสื่อสารผ่านทางระบบเครือข่าย โดยทำหน้าที่เป็นเครื่องเซิร์ฟเวอร์เพื่อรอการติดต่อจากไคลเอนต์ เมื่อไคลเอนต์ส่งข้อมูลเข้ามาจะส่งต่อให้ Interprocess Communication Module จึงมีการสร้างช่องทางในการติดต่อด้วย

กระบวนการทำงานของ Network Interface Module เริ่มต้นจะสร้างซ็อคเก็ตขึ้นมา แล้วทำการกำหนดชื่อ (Bind) ให้กับซ็อคเก็ตเพื่อกำหนดจุดเชื่อมต่อซึ่งประกอบด้วย หมายเลขพอร์ตและจุดเชื่อมต่อของโปรเซส ซึ่งเป็นค่าเฉพาะสำหรับเครือข่ายนั้น จากนั้นจะรอรับการเชื่อมต่อจากไคลเอนต์ (Listen) เมื่อมีไคลเอนต์ติดต่อเข้ามาจะตอบรับ (Accept) โดยจะสร้างซ็อคเก็ตขึ้นมาเพื่อใช้ติดต่อกับไคลเอนต์เฉพาะ เมื่อมีการส่งข้อมูลมา จะส่งผ่านทางซ็อคเก็ตนี้ แล้ว Network Interface Module จะส่งข้อมูลนั้นต่อไปยัง Interprocess Communication Module และในขณะเดียวกัน ถ้า Interprocess Communication Module ส่งข้อมูลมาผ่านทางช่องทางที่ได้สร้างไว้แล้ว จะส่งต่อไปยังไคลเอนต์ที่เชื่อมต่ออยู่

การทำงานของ Network Interface Module จะมีการตรวจสอบด้วยว่าไคลเอนต์ที่เชื่อมต่ออยู่ยังคงมีการติดต่อสื่อสารกันอยู่หรือไม่ ถ้าไม่มีการติดต่อสื่อสารกันเป็นระยะเวลาหนึ่งจะมีการตัดช่องทางการเชื่อมต่อ โดย Network Interface Module จะทำการปิดไฟล์ (File descriptor) ที่ใช้ในการอ้างอิงซ็อกเก็ตซึ่งเป็นช่องทางในการติดต่อสื่อสารกับไคลเอนต์ และรอรับการเชื่อมต่อจากไคลเอนต์ใหม่ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.12 แสดงกระบวนการทำงานของ Network Interface Module ที่ทำหน้าที่เป็นเซิร์ฟเวอร์

4.3 Serial Device Interface Module

ส่วนที่ใช้ในการติดต่อพอร์ตอนุกรมโดยประเด็นหลักของโมดูลนี้ คือ จะทำการเขียนโปรแกรมติดต่ออุปกรณ์ภายนอกผ่านพอร์ตอนุกรม โดยโมดูลนี้จะทำการรับข้อมูลจาก Interprocess Communication Module และทำการส่งข้อมูลต่อไปยังอุปกรณ์ภายนอกโดยผ่านพอร์ตอนุกรม ในขณะเดียวกันก็จะทำการรับข้อมูลจากอุปกรณ์ภายนอกและส่งกลับให้ Interprocess Communication Module เช่นกัน จึงได้สร้างช่องทางที่ใช้ติดต่อกับ Interprocess Communication Module ขึ้นด้วย

ซึ่งกระบวนการทำงานของ Serial Device Interface Module เริ่มแรกจะมีการกำหนดค่าเริ่มต้นในการติดต่อสื่อสาร เช่น ความเร็วและลักษณะของข้อมูลที่ใช้ในการสื่อสาร จากนั้นจะทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การรอรับข้อมูลที่ Interprocess Communication Module ส่งมา แล้วส่งข้อมูลนั้นต่อไปยังพอร์ตอนุกรมที่อุปกรณ์ต่ออยู่ และในขณะเดียวกันก็รอรับข้อมูลที่อุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรมส่งมาแล้วจะส่งข้อมูลนั้นต่อไปยัง Interprocess Communication Module ต่อไป โดยในการรับส่งข้อมูลแต่ละครั้งของ Serial Device Interface Module จะมีการตรวจสอบสถานะของพอร์ตอนุกรมว่าพร้อมในการทำงานหรือไม่ ซึ่งถ้าพอร์ตอนุกรมไม่พร้อมทำงานก็จะมีการส่งข้อความกลับไปแจ้งยัง Interprocess Communication Module ต่อไป โดยการตรวจความพร้อมของอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรม จะทำโดยการตรวจสอบขาสัญญาณ DTR CTS และ RTS ถ้าขาดังกล่าวมีสัญญาณเป็น 1 แสดงว่าพร้อมที่จะทำงาน แต่ถ้ามีบางค่าเป็น 0 แสดงว่าพอร์ตอนุกรมไม่พร้อมที่จะทำงาน

4.4 HTTP Module

HTTP Module เป็นส่วนที่ทำหน้าที่สนับสนุนการติดต่อจากไคลเอนต์ที่ใช้เว็บบราวเซอร์ โดย HTTP Module จะรอรับข้อมูลที่ Interprocess Communication Module พิจารณาแล้วว่าเป็น HTTP request ที่ถูกส่งมาจากเว็บบราวเซอร์ โดย HTTP Module จะตรวจสอบ HTTP request ดังกล่าวว่าไคลเอนต์ต้องการข้อมูล (ไฟล์) ใด จากนั้นจึงทำการอ่านข้อมูลดังกล่าวขึ้นมา แล้วสร้าง HTTP response ที่ประกอบด้วยข้อมูลดังกล่าวตอบกลับไปด้วย เพื่อให้ Interprocess Communication Module ส่งต่อให้ Network Interface Module ส่งไปให้เว็บบราวเซอร์ที่ไคลเอนต์ต่อไป

ในข้อมูลที่ Interprocess Communication Module ส่งมาให้ จะเป็นข้อมูลของ HTTP request พร้อมทั้งแนบหมายเลขอ้างอิงคิวที่ใช้อ้างอิงช่องทางสำหรับส่ง HTTP response กลับมาด้วย โดยในการประมวลผลจะตรวจสอบ HTTP request ว่าไคลเอนต์ต้องการข้อมูลใด

HTTP request ที่ไคลเอนต์ส่งมาในครั้งแรกจะมีลักษณะ ดังนี้

```
GET / HTTP/1.1
```

HTTP Module เมื่อรับข้อมูลดังกล่าว ก็จะทำการอ่านไฟล์ index.html (ไฟล์เอชทีเอ็มแอลที่เก็บอยู่ในเครื่องเซิร์ฟเวอร์) ขึ้นมาเพื่อนำไปสร้าง HTTP response ต่อไป ซึ่งใน index.html จะแทรกแท็กจาวาแอพเพลตในการสั่งให้จาวาแอพเพลตทำงาน ดังนี้

```
<APPLET        CODE =”Remote_Access.class” WIDTH=”900” HEIGHT=”500”>
        <PARAM        NAME=”HOST” VALUE=”10.1.1.2”>
        <PARAM        NAME=”IP” VALUE=”8000”>
</APPLET>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังนั้นเว็บบราวเซอร์จะส่ง HTTP request มาอีกครั้ง เพื่อร้องขอไฟล์ Remote_Access.class ซึ่ง HTTP request มีลักษณะ ดังนี้

```
GET /Remote_Access.class HTTP/1.1
```

HTTP Module เมื่อรับข้อมูลดังกล่าว ก็จะทำการอ่านไฟล์ Remote_Access.class (ไฟลจาวาแอพเพลตที่เก็บอยู่ในเครื่องเซิร์ฟเวอร์) ขึ้นมาเพื่อนำไปสร้าง HTTP response ต่อไป

ในการสร้าง HTTP response ตัว HTTP Module จะสร้างเฮดเดอร์ของเอชทีทีพีโพรโทคอล แล้วต่อท้ายด้วยข้อมูลที่จะส่ง (อาจเป็นไฟล์เอชทีเอ็มแอลหรือไฟล์จาวาแอพเพลตก็ได้) เมื่อสร้าง HTTP response เรียบร้อยแล้ว จะส่งต่อให้ Interprocess Communication Module โดยจะแนบหมายเลขอ้างอิงคิวที่ใช้อ้างอิงช่องทางสำหรับส่ง HTTP response กลับที่ได้จากตอนที่ Interprocess Communication Module ส่ง HTTP request มาไปด้วย

การทำงานของจาวาแอพเพลต (ไฟล์ Remote_Access.class)

จาวาแอพเพลตจะสร้างอินเทอร์เฟสเป็นเท็คบล็อก (TextBox) แล้วเชื่อมการติดต่อกับเทอร์มินัลเซิร์ฟเวอร์ด้วยโพรโทคอลทีซีพีผ่านซ็อคเก็ต โดยดูค่าไอพีแอดเดรสและหมายเลขพอร์ต ซึ่งเป็นค่าพารามิเตอร์ที่ส่งมาพร้อมกับไฟล์เอชทีเอ็มแอลจาวาแอพเพลตจะทำงานเป็นเทรด (Thread) การรับค่าที่จะส่งเข้ามาและคอยฟังการกดคีย์ (KeyListenner) จากผู้ใช้ เมื่อผู้ใช้กดคีย์ ค่าของคีย์ดังกล่าวจะถูกส่งไปเทอร์มินัลเซิร์ฟเวอร์ทันที ในขณะเดียวกันเมื่อมีการส่งค่ามาจะนำมาแสดงผลบนเท็คบล็อก

4.5 การประสานงานการทำงานของแต่ละโมดูล

ในการพัฒนาในแต่ละโมดูลมีการทำงานที่แยกกัน เมื่อแต่ละโมดูลพัฒนาสมบูรณ์แล้วจึงมีการรวมแต่ละโมดูลเข้าด้วยกัน ซึ่งการเชื่อมต่อแต่ละโมดูลเข้าด้วยกัน แบ่งเป็น 3 ส่วน ดังนี้

4.5.1 การเชื่อมต่อระหว่าง Network Interface Module และ Interprocess Communication Module ใช้การติดต่อระหว่างโปรเซสแบบ Message queue ซึ่งการรับส่งข้อมูลของทั้ง 2 โมดูล จะรับส่งผ่านทางคิวที่สร้างมา 2 คิว อันหนึ่งไว้สำหรับรับข้อมูลเข้า อีกอันหนึ่งไว้สำหรับส่งข้อมูลออก เพื่อป้องกันไม่ให้ข้อมูลรับและส่งปะปนกัน การอ้างถึงคิวของทั้ง 2 โมดูลจะต้องมีหมายเลขอ้างอิงคิวในการอ้างถึง (เช่นคิวสำหรับส่งข้อมูลของ Network Interface Module จะมีหมายเลขอ้างอิงคิวตรงกับหมายเลขอ้างอิงคิวของคิวสำหรับรับข้อมูลของ Interprocess Communication Module) Network

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Interface Module จะคอยตรวจดูในคิวว่า Interprocess Communication Module ส่งข้อมูลมาหรือไม่ เช่นเดียวกัน Interprocess Communication Module ก็จะคอยตรวจดูว่า Network Interface Module ส่งข้อมูลมาหรือไม่

4.5.2 การเชื่อมต่อระหว่าง Interprocess Communication Module และ Serial Device Interface Module ก็ใช้หลักการเดียวกับ การเชื่อม Network Interface Module และ Interprocess Communication Module เข้าด้วยกันทุกประการ

4.5.3 การเชื่อมต่อระหว่าง Interprocess Communication Module และ HTTP Module ก็ใช้หลักการเดียวกับ การเชื่อม Network Interface Module และ Interprocess Communication Module เข้าด้วยกันทุกประการ

4.6 ฟังก์ชันเสริมของ Interprocess Communication Module

ขณะที่ไคลเอนต์เชื่อมต่อกับเทอร์มินัลเซิร์ฟเวอร์สามารถเข้าสู่ Off-Line mode (Command mode) เพื่อติดต่อสื่อสารกับเทอร์มินัลเซิร์ฟเวอร์ที่เชื่อมต่อโดยตรง โดยคำสั่งที่ถูกส่งมานั้น Interprocess Communication Module จะไม่ส่งต่อไปยัง Serial Device Interface Module ที่ควบคุมพอร์ตอนุกรมอยู่ หรือ Network Interface Module ที่เชื่อมต่อกับเทอร์มินัลเซิร์ฟเวอร์อื่น

เมื่อไคลเอนต์ต้องการเข้าสู่ Off-Line mode ต้องพิมพ์คีย์ '^' จะเข้าสู่ Off-Line mode ไคลเอนต์สามารถติดต่อกับเทอร์มินัลเซิร์ฟเวอร์โดยตรงได้ โดยมีคำสั่งต่อไปนี้

- q (quit) คำสั่งให้ตัดการติดต่อสื่อสาร
- b (back) คำสั่งให้กลับสู่ On-Line mode
- n (end) คำสั่งให้เทอร์มินัลเซิร์ฟเวอร์หยุดกระบวนการทำงาน

หลักการทำงานในฟังก์ชันเสริม

เมื่อไคลเอนต์ส่งคำสั่งเข้ามายังเทอร์มินัลเซิร์ฟเวอร์ทางพอร์ตหนึ่ง Network Interface Module ที่รอรับข้อมูลที่พอร์ตนั้น จะทำการรับคำสั่งและส่งไปให้ Interprocess Communication Module เมื่อ Interprocess Communication Module ตรวจสอบคำสั่งนั้นแล้วพบว่าเป็น '^' จะกำหนดให้การติดต่อสื่อสารที่มาทาง Network Interface Module ที่รอรับข้อมูลทางพอร์ตดังกล่าวนั้นอยู่ใน Off-Line mode จากนั้นเมื่อมีคำสั่งเข้ามาทาง Network Interface Module ดังกล่าว Interprocess Communication Module จะไม่ส่งต่อไปยัง Serial Device Interface Module ควบคุมพอร์ตอนุกรมหรือ Network Interface Module ที่เชื่อมต่อกับเทอร์มินัลเซิร์ฟเวอร์อื่น แต่จะนำมาตรวจสอบว่าคำสั่งที่ส่งมานั้นตรงกับคำสั่งใดใน Off-Line mode หากตรงกับคำสั่งใด Interprocess Communication Module จะส่งรหัสคำสั่งไปยังโมดูลต่างๆที่เกี่ยวข้องเพื่อจัดการต่อไป เช่น เมื่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตรวจพบว่าเป็นคำสั่ง q จะส่งรหัสคำสั่ง 46060008 ไปยัง Network Interface Module ที่เชื่อมต่ออยู่กับการติดต่อสื่อสารที่อยู่ใน Off-Line mode เมื่อ Network Interface Module รับข้อมูลจาก Interprocess Communication Module พบว่าเป็นรหัสคำสั่ง 46060008 จะตัดการติดต่อสื่อสารนั้นทันที หากเป็นคำสั่ง b จะไม่ส่งรหัสต่อไปยังโมดูลใด แต่ Interprocess Communication Module จะกำหนดให้การติดต่อสื่อสารที่อยู่ใน Off-Line mode นั้นกลับส่ง On-Line mode แต่หากเป็นคำสั่ง n Interprocess Communication Module จะส่งรหัสคำสั่งไปยังทุกๆโมดูลเพื่อให้หยุดการทำงาน โดยการหยุดการรอรับข้อมูลและจบการทำงานไป

4.7 ข้อจำกัดของระบบ

4.7.1 ข้อจำกัดในการสั่งให้ระบบหยุดทำงานแบบอัตโนมัติ

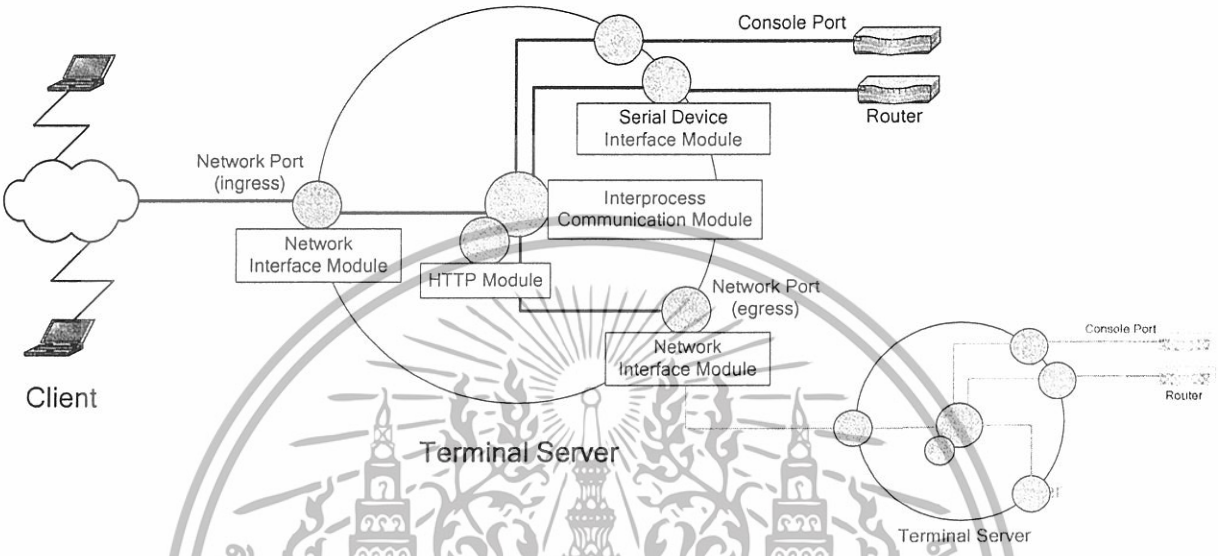
เนื่องจากการทำงานของระบบมีหลายโมดูลทำงานแยกกันอย่างชัดเจน จึงทำให้เมื่อต้องการสั่งหยุดการทำงานของระบบจึงต้องหยุดการทำงานทีละโมดูล ซึ่งจะต้องทราบก่อนว่าโปรเซสของโมดูลที่ทำงานอยู่นั้นมีหมายเลขโปรเซส (PID) อะไร โดยการใช้คำสั่ง ps –ax ในการดูว่าโปรเซสที่ทำงานอยู่มีอะไรบ้าง เราจึงทำการเลือกโปรเซสที่เกี่ยวข้องกับระบบมาเพื่อสั่งให้หยุดทำงาน โดยใช้คำสั่ง kill ตามด้วยหมายเลขโปรเซสที่เราต้องการสั่งให้หยุดทำงาน

4.7.2 ข้อจำกัดทางด้านความปลอดภัยของระบบ

ในส่วนของ HTTP Module ทำหน้าที่ตอบสนองต่อ HTTP request ที่เว็บบราวเซอร์ส่งมาร้องขอข้อมูล โดย HTTP Module จะทำการส่งข้อมูลที่ต้องการไปในรูปของ HTTP response ตามคำขอ แต่หากมีผู้ไม่ประสงค์ดีทำการปลอม HTTP request แล้วส่งมายังเทอร์มินัลเซิร์ฟเวอร์เพื่อร้องขอข้อมูลอื่นที่ไม่เกี่ยวข้องกับระบบภายในเทอร์มินัลเซิร์ฟเวอร์ เช่น พาสเวิร์ด HTTP Module ก็สามารถตอบสนองคำร้องขอได้ด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
การทดสอบระบบ



รูปที่ 5.1 ภาพสถาปัตยกรรมของระบบ

เทอร์มินัลเซิร์ฟเวอร์เป็นระบบการทำงานที่ใช้ในการติดต่อกับอุปกรณ์ภายนอกที่เชื่อมต่อกับพอร์ตอนุกรมโดยให้ผู้ใช้ต้องระบุพอร์ตในการเชื่อมต่อผ่านระบบเครือข่าย เพื่อเลือกใช้อุปกรณ์ปลายทางที่ติดต่อกับพอร์ตอนุกรม

5.1 ลักษณะการทดสอบระบบ

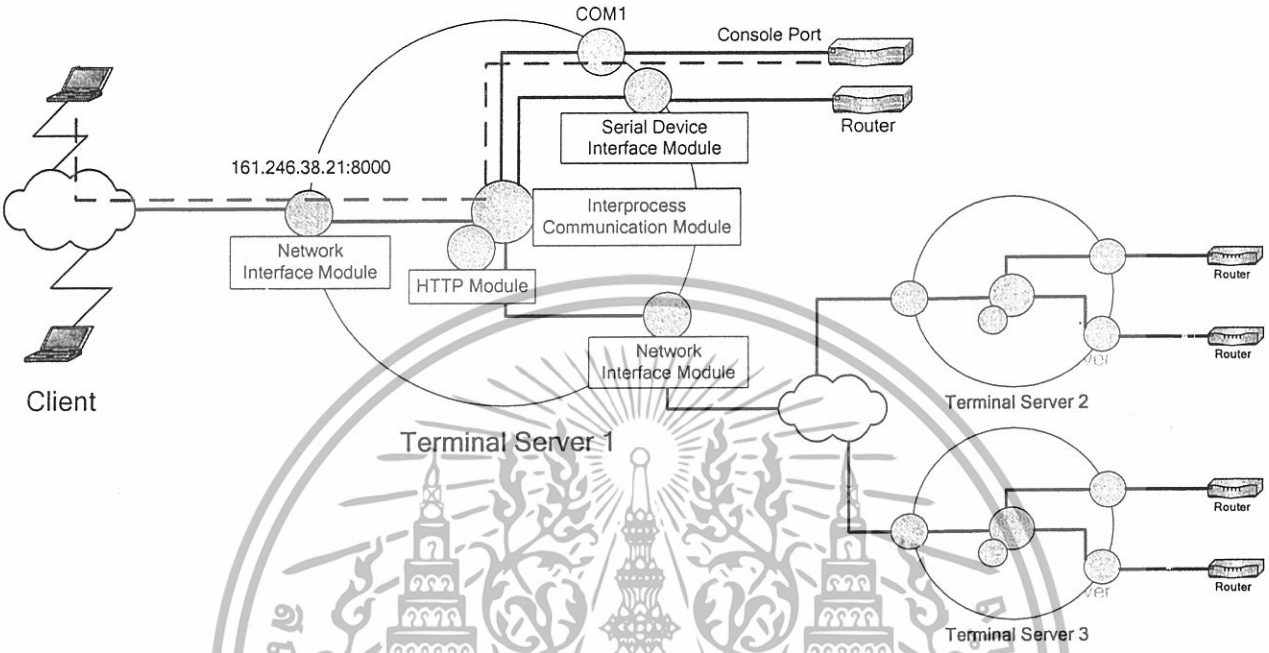
เมื่อไคลเอนต์ทำการติดต่อเข้ามายังเทอร์มินัลเซิร์ฟเวอร์แล้วต้องการควบคุมการทำงานของอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรม ซึ่งจะต้องสามารถควบคุมการทำงานของอุปกรณ์ได้เช่นเดียวกับการติดต่ออุปกรณ์โดยตรงโดยมีลักษณะการทดสอบระบบมีดังนี้

- ทดสอบว่าระบบสามารถเชื่อมโยงการติดต่อจากเครือข่ายอีเทอร์เน็ตมายังอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรมได้จริง
- ทดสอบว่าระบบสามารถรวมกลุ่มของเทอร์มินัลเซิร์ฟเวอร์หลายๆระบบย่อยเป็นระบบใหญ่และสามารถเชื่อมโยงการทำงานร่วมกันได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.1 การทดสอบแบบที่ 1

การทดสอบเพื่อยืนยันว่าระบบสามารถเชื่อมโยงการติดต่อสื่อสารจากเครือข่ายอีเทอร์เน็ต ไปยังอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรมได้จริง



รูปที่ 5.2 ภาพรวมของการทดสอบแบบที่ 1

ตารางที่ 5.1 ตารางกำหนดเส้นทางที่ใช้ในการตัดสินใจของเทอร์มินัลเซิร์ฟเวอร์ 1

IP	PORT	IP	PORT
161.246.38.21	8000	-	COM1
161.246.38.21	8001	-	COM2
161.246.38.21	9000	10.20.7.10	8500
161.246.38.21	9001	10.20.7.10	8501

รายละเอียดการทดสอบแบบที่ 1

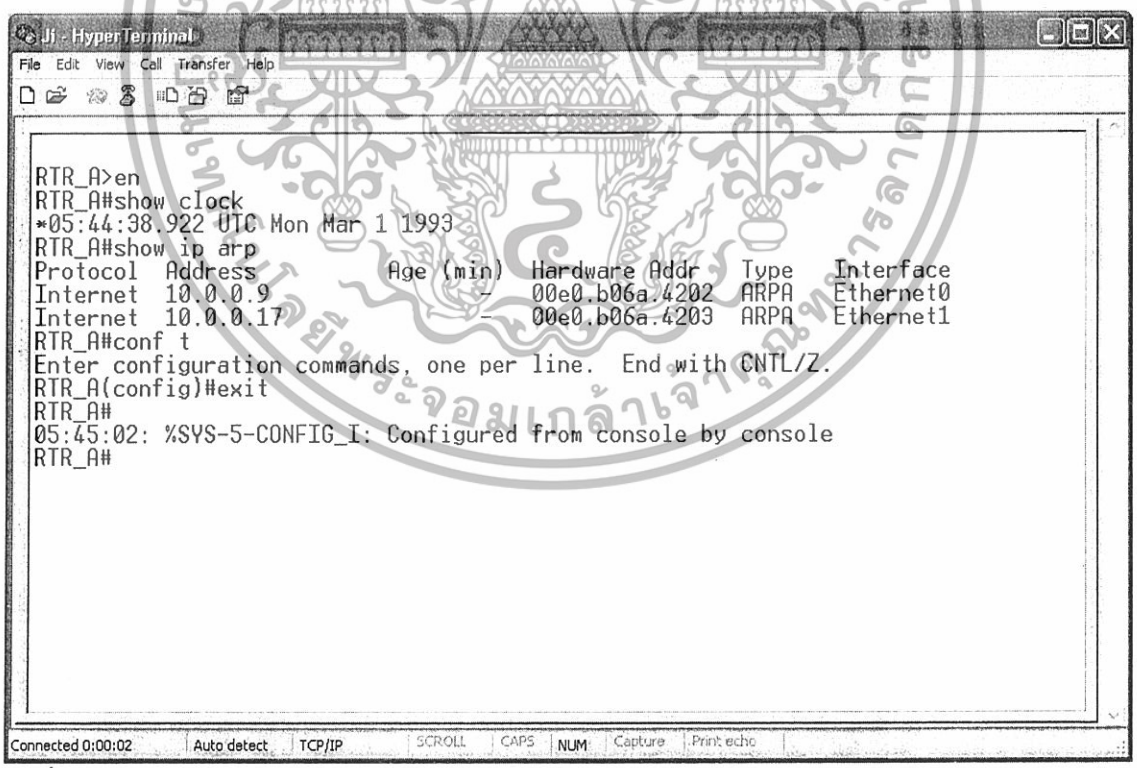
จากรูปที่ 5.2 แสดงถึงการทดสอบการทำงานของระบบซึ่ง ในการทดลองนี้จะให้ไคลเอนต์ทำการติดต่อจากระยะไกล (Remote access) เข้าไปยังเทอร์มินัลเซิร์ฟเวอร์ที่ไอพี 161.246.38.21 ทางพอร์ต 8000 ด้วยโพรโทคอลทีซีพี ผ่านโปรแกรม Terminal Emulator ด้วยไฮเพอร์ เทอร์มินัล โดยการทำงานของระบบ Network Interface Module จะทำหน้าที่ในการติดต่อสื่อสารกับเครือข่ายอีเทอร์เน็ต เมื่อไคลเอนต์ส่งข้อมูลเข้ามา Network Interface Module จะทำการส่งข้อมูลไปยัง Interprocess Communication Module จากนั้น Interprocess Communication Module จะทำการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พิจารณาว่าข้อมูลที่เข้ามาทางพอร์ต 8000 ควรจะส่งต่อไปยังช่องทางใด ตามตารางกำหนดเส้นทาง (ตารางที่ 5.1) พบว่าต้องส่งไปที่พอร์ตอนุกรม COM1 ก็จะทำการส่งข้อมูลไปยัง Serial Device Interface Module ที่ควบคุมพอร์ตอนุกรม COM1 เมื่อ Serial Device Interface Module รับข้อมูลมาจาก Interprocess Communication Module แล้ว จะส่งข้อมูลนั้นไปยังอุปกรณ์ปลายทางที่เชื่อมต่ออยู่กับพอร์ตอนุกรม COM1 (เราเตอร์) หากอุปกรณ์ที่เชื่อมต่ออยู่กับพอร์ตอนุกรม COM1 มีการส่งข้อมูลตอบสนองกลับมา Serial Device Interface Module จะรับและส่งข้อมูลนั้นกลับไปยัง Interprocess Communication Module เมื่อ Interprocess Communication Module ได้รับข้อมูลจะพิจารณาว่าข้อมูลที่ส่งมาจากพอร์ตอนุกรม COM1 ควรจะส่งต่อไปยังช่องทางใด ตามตารางกำหนดเส้นทาง พบว่าต้องส่งไปที่พอร์ต 8000 ก็จะส่งข้อมูลนั้นไปให้ Network Interface Module เพื่อส่งไปแสดงผลยังไคลเอนต์ต่อไป

ผลการทดสอบแบบที่ 1

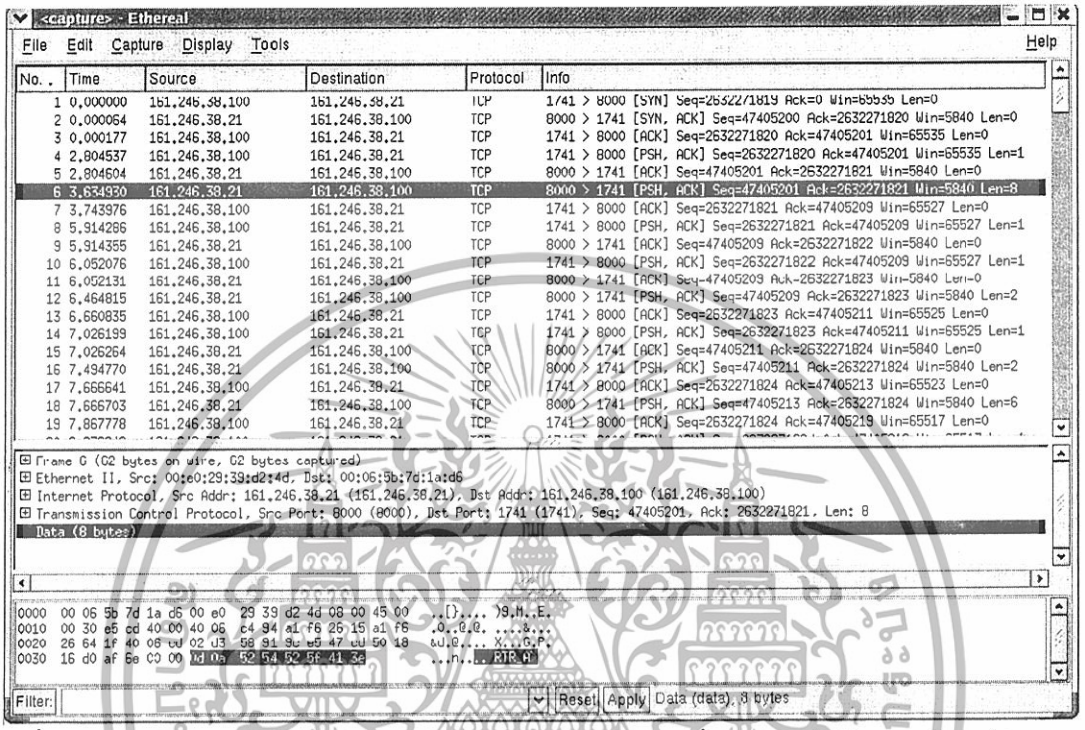
เมื่อทำการติดต่อจากระยะไกลด้วยโปรแกรมไฮเพอร์เทอร์มินัลไปยังเราเตอร์ โดยติดต่อผ่านเทอร์มินัลเซิร์ฟเวอร์ที่ไอพี 161.246.38.21 ทางพอร์ต 8000 ผลจากการติดต่อสื่อสาร เป็นดังรูปที่ 5.3



รูปที่ 5.3 แสดงการติดต่อสื่อสารกับเราเตอร์ผ่านเทอร์มินัลเซิร์ฟเวอร์โดยใช้โปรแกรมไฮเพอร์เทอร์มินัล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

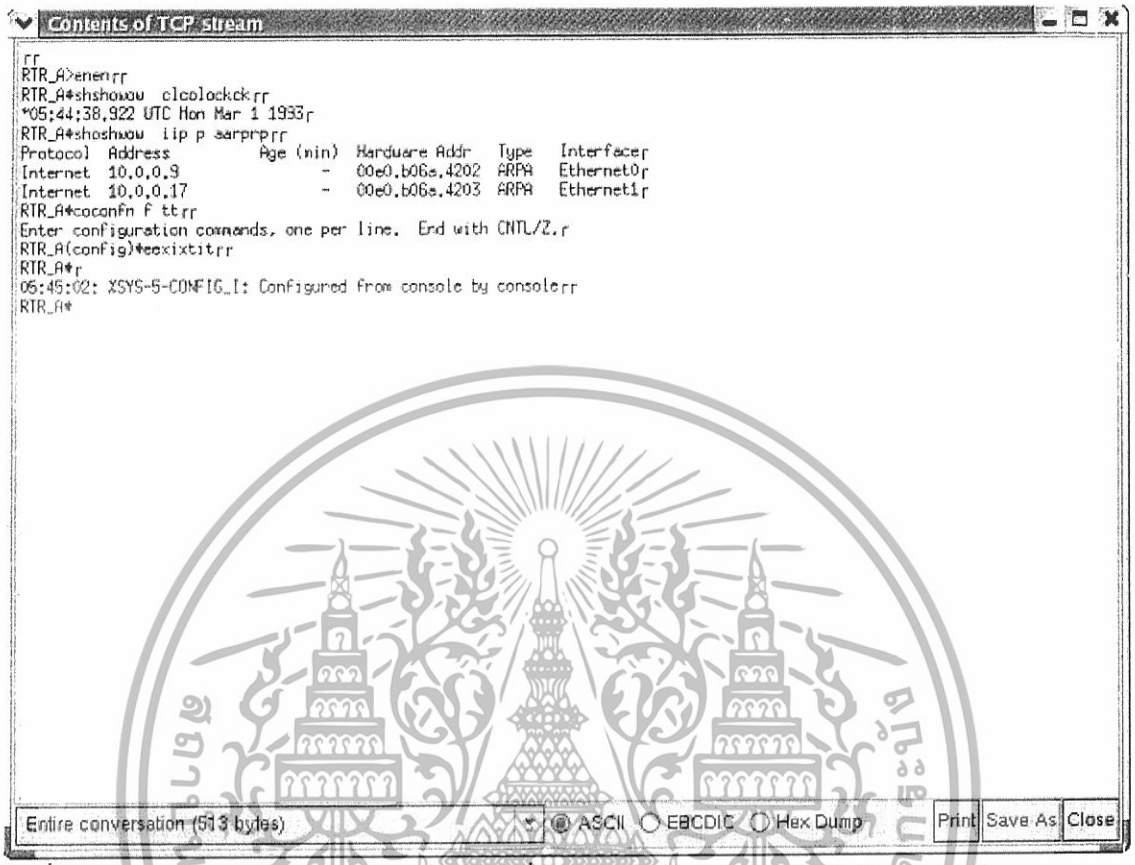
เมื่อเราได้ทำการดักจับข้อมูลด้วยโปรแกรมอีเทอร์เรียล (Ethereal) ที่เน็ตเวิร์คพอร์ตอินเกรส (Network port ingress) เพื่อทำการยืนยันว่ามีการรับส่งข้อมูลระหว่างไคลเอนต์ (ไอพี 161.246.38.100) กับเทอร์มินัลเซิร์ฟเวอร์ (ไอพี 161.246.38.21) จริง โดยได้ผลการดักจับข้อมูล ดังรูปที่ 5.4



รูปที่ 5.4 ผลการดักจับข้อมูลโดยใช้โปรแกรมอีเทอร์เรียล ขณะที่ไคลเอนต์ใช้ไฮเพอร์เทอร์มินัลติดต่อกับเราเตอร์ผ่านทางเทอร์มินัลเซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยหลังจากทำการดักจับข้อมูลแล้ว ได้ทำการติดตามการรับส่งข้อมูลของทีซีพีสตรีม (TCP stream) มีลำดับการรับส่งข้อมูล ดังรูป 5.5



รูปที่ 5.5 หลังจากการทำ Follow TCP stream เพื่อแสดงการติดตามการรับส่งข้อมูลระหว่างไคลเอนต์กับเทอร์มินัลเซิร์ฟเวอร์

การใช้เว็บบราวเซอร์ในการติดต่อไปยังเราเตอร์โดยเรียกผ่านเทอร์มินัลเซิร์ฟเวอร์

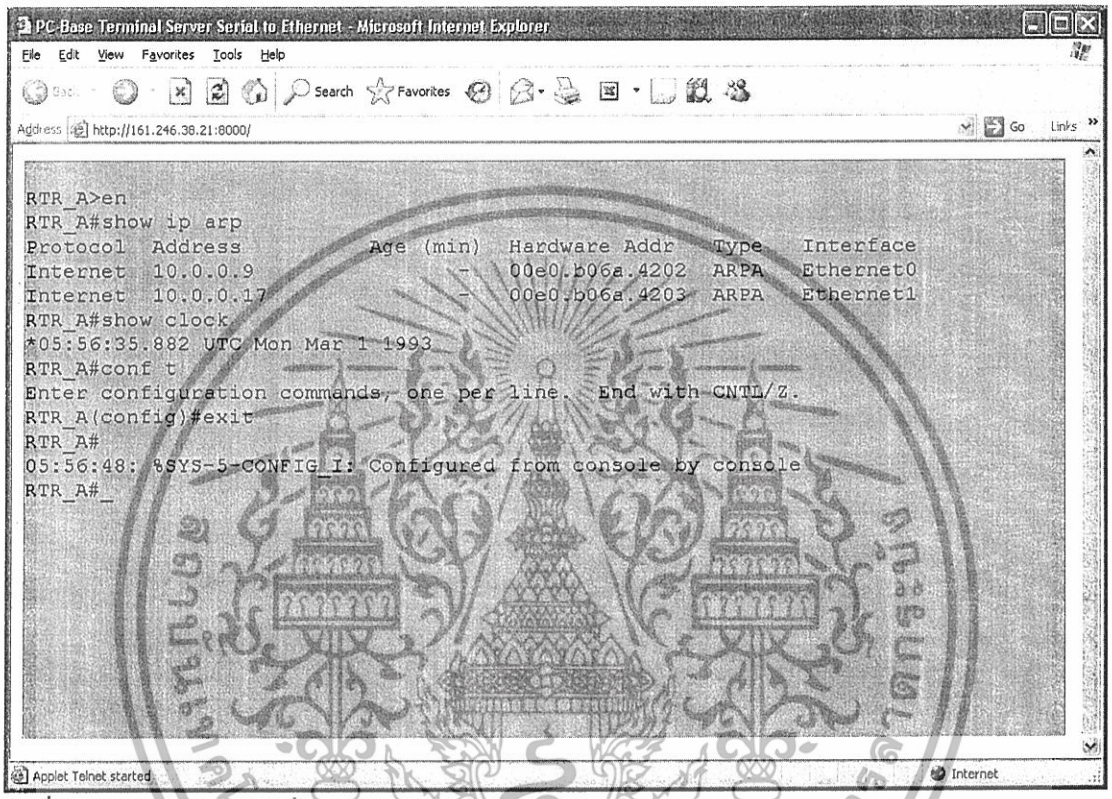
เมื่อใช้เว็บบราวเซอร์ (Internet Explorer) ติดต่อมาที่เทอร์มินัลเซิร์ฟเวอร์ เว็บบราวเซอร์จะส่ง HTTP request เพื่อร้องขอไฟล์เอชทีเอ็มแอล (index.html) ไปแสดงผล ซึ่ง Network Interface Module จะรับข้อมูลที่เป็น HTTP request นั้นและส่งต่อไปยัง Interprocess Communication Module เมื่อ Interprocess Communication Module ได้รับข้อมูลแล้วพิจารณาว่าเป็น HTTP request ก็จะส่งข้อมูลนั้นต่อยัง HTTP Module จัดการ โดย HTTP Module จะสร้าง HTTP response ซึ่งประกอบด้วยไฟล์เอชทีเอ็มแอลส่งกลับไปยัง Interprocess Communication Module จากนั้น Interprocess Communication Module จะส่งต่อไปยัง Network Interface Module และ Network Interface Module ก็จะส่งกลับไปประมวลผลยังเว็บบราวเซอร์

เมื่อเว็บบราวเซอร์ประมวลผลไฟล์เอชทีเอ็มแอลแล้วต้องการไฟล์แอพเพลต (Remote_Access.class) จึงส่ง HTTP request ไปร้องขอไฟล์แอพเพลตอีกครั้งหนึ่ง หลังจากนั้นเมื่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เว็บบราวเซอร์ได้รับไฟล์แอพเพลตแล้ว ตัวจาวาแอพเพลตจะทำหน้าที่เป็นโปรแกรมไคลเอนต์เพื่อติดต่อกลับไปยังเทอร์มินัลเซิร์ฟเวอร์ต่อไป

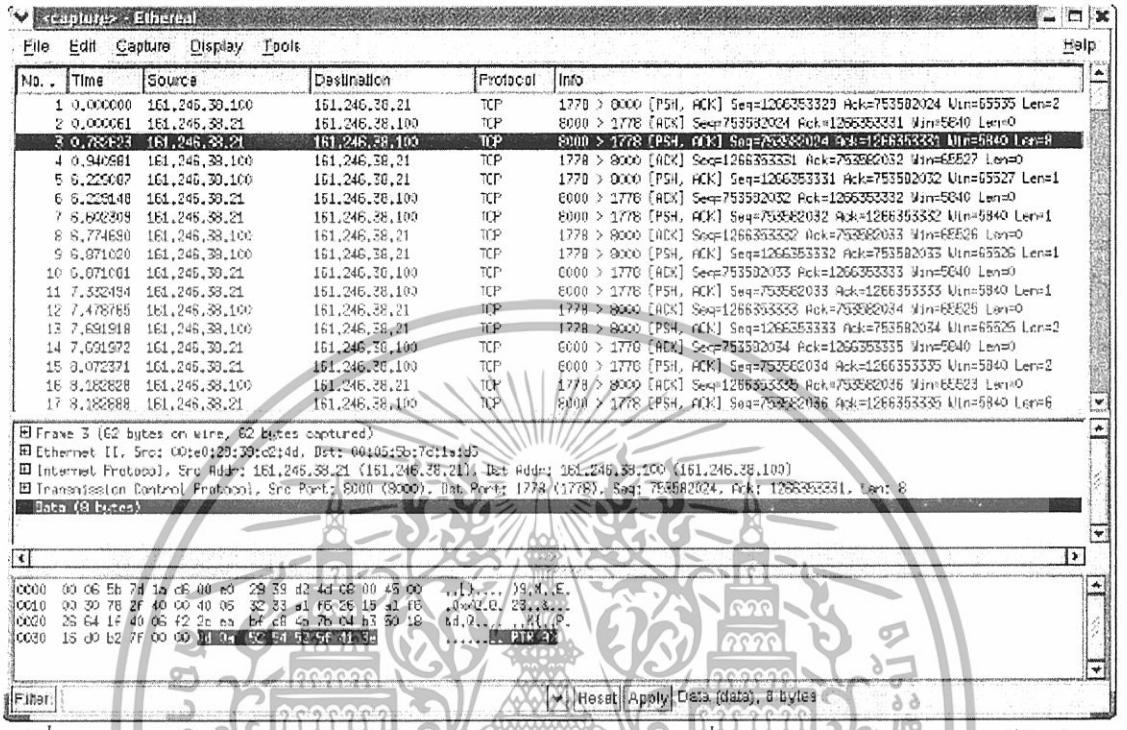
ในลักษณะเดียวกัน เมื่อทำการติดต่อจากระยะไกลด้วยเว็บบราวเซอร์ไปยังเราเตอร์ผ่านเทอร์มินัลเซิร์ฟเวอร์ที่ไอพี 161.246.38.21 ทางพอร์ต 8000 ผลการติดต่อสื่อสารเป็นดังรูปที่ 5.6



รูปที่ 5.6 แสดงการติดต่อสื่อสารกับเราเตอร์ผ่านเทอร์มินัลเซิร์ฟเวอร์โดยใช้โปรแกรมเว็บบราวเซอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

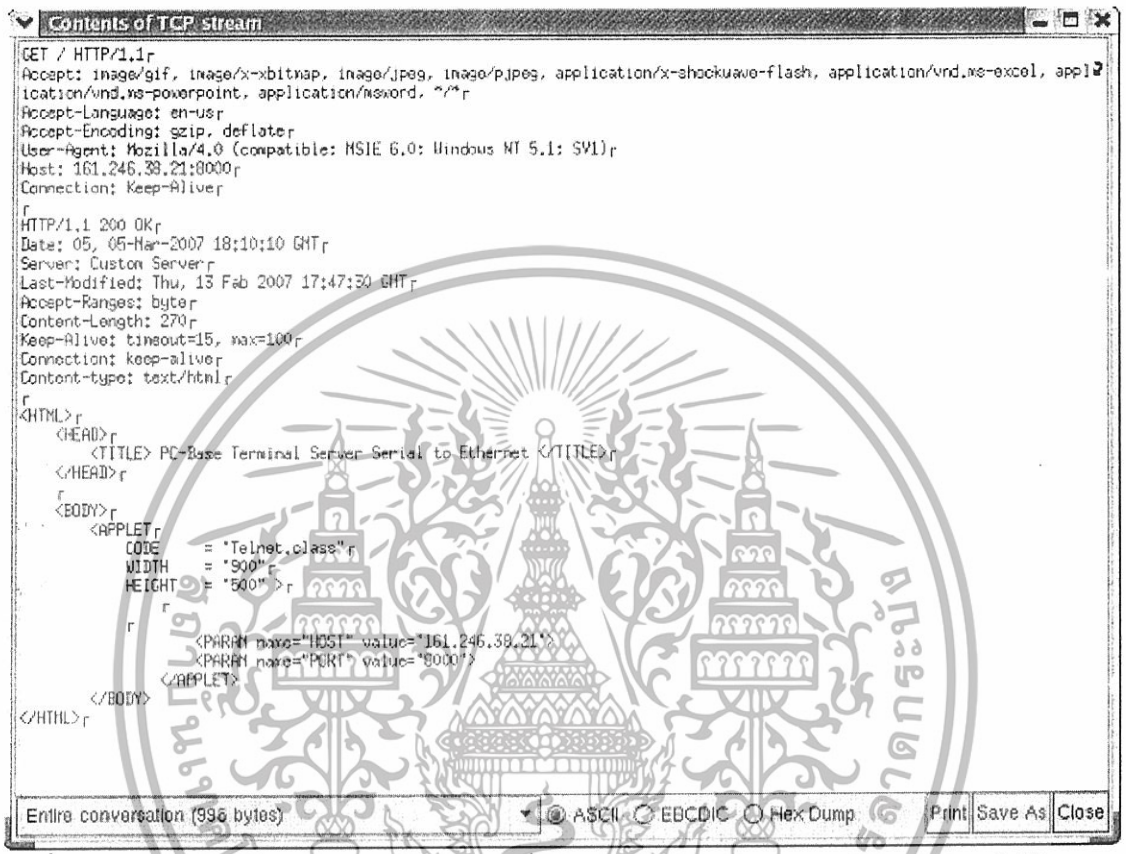
เมื่อเราได้ทำการดักจับข้อมูลด้วยโปรแกรมอีเทอร์เรียลที่เน็ตเวิร์คพอร์ตอินเกรส (Network port ingress) เพื่อทำการยืนยันว่ามีการรับส่งข้อมูลระหว่างไคลเอนต์ (ไอพี 161.246.38.100) กับเทอร์มินัลเซิร์ฟเวอร์ (ไอพี 161.246.38.21) จริง โดยได้ผลการดักจับข้อมูล ดังรูปที่ 5.7



รูปที่ 5.7 ผลการดักจับข้อมูลโดยใช้โปรแกรมอีเทอร์เรียล ขณะที่ไคลเอนต์ใช้เว็บบราวเซอร์ติดต่อกับเทอร์มินัลเซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

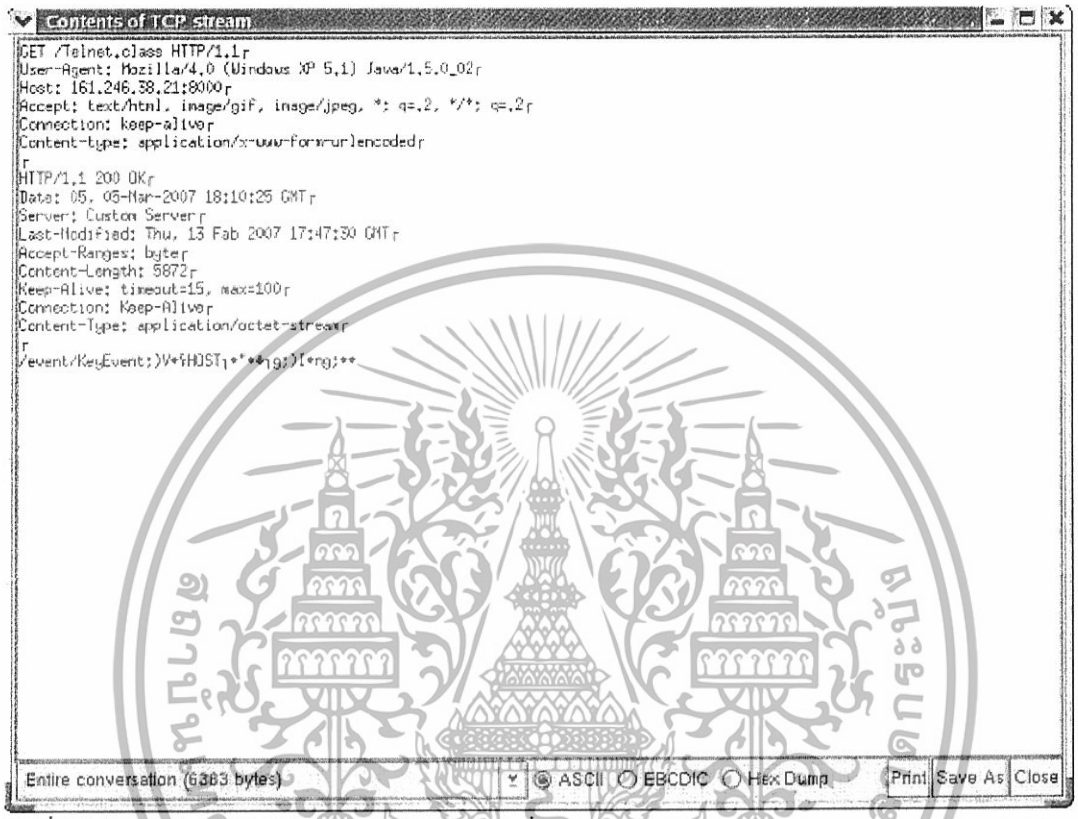
แสดงการติดตามการรับส่งข้อมูลของทีซีพีสตรีม ซึ่งแสดงให้เห็นกระบวนการส่ง HTTP request ของเว็บบราวเซอร์เพื่อไปขอไฟล์เอชทีเอ็มแอลจากเทอร์มินัลเซิร์ฟเวอร์และการส่ง HTTP response ที่ประกอบด้วยไฟล์เอชทีเอ็มแอลของเทอร์มินัลเซิร์ฟเวอร์กลับไปยังเว็บบราวเซอร์ซึ่งแสดงผล ดังรูป 5.8



รูปที่ 5.8 หลังจากการทำ Follow TCP stream เพื่อแสดงการติดตามการส่ง HTTP request ไปร้องขอไฟล์เอชทีเอ็มแอลและการส่ง HTTP response ตอบกลับมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

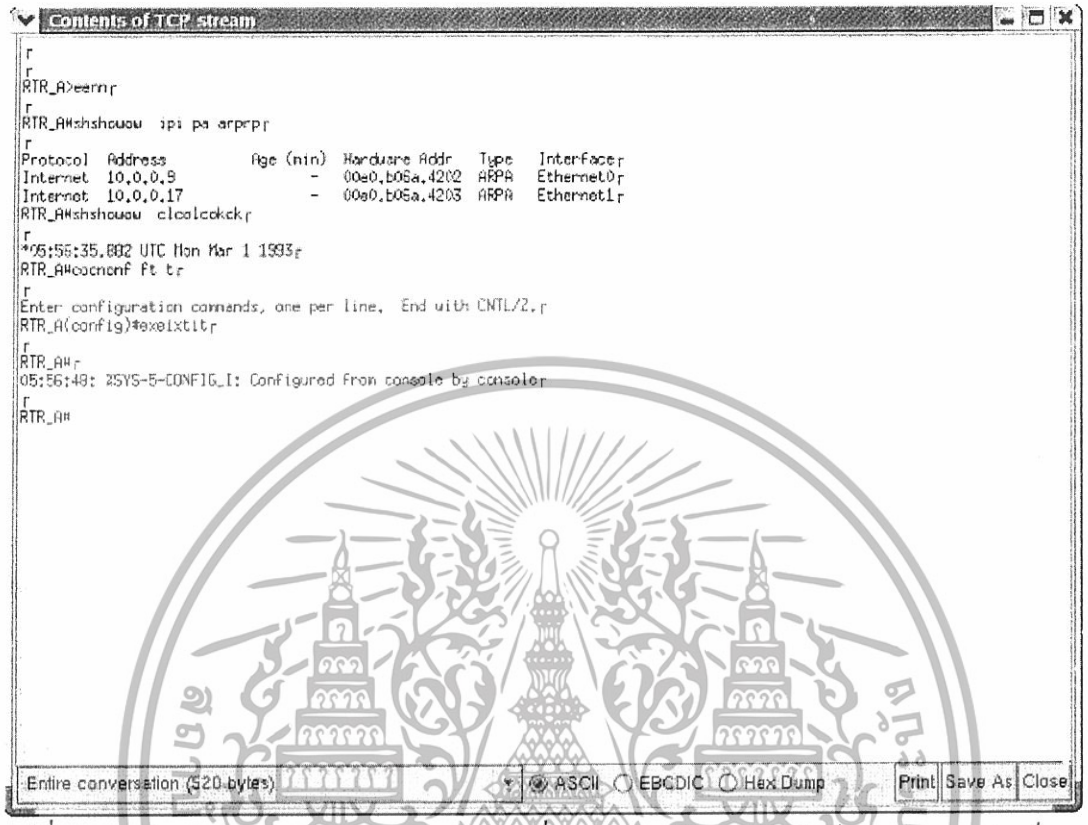
แสดงการติดตามการรับส่งข้อมูลของทีซีพีสตรีม ซึ่งแสดงให้เห็นว่าเมื่อเว็บบราวเซอร์ได้รับไฟล์เอชทีเอ็มแอลที่กล่าวมาก่อนหน้านี้แล้ว เว็บบราวเซอร์จะทำการส่ง HTTP request เพื่อร้องขอไฟล์แอพเพลตไปยังเทอร์มินัลเซิร์ฟเวอร์และเทอร์มินัลเซิร์ฟเวอร์ก็จะส่ง HTTP response โดยแนบไฟล์แอพเพลตกลับยังเว็บบราวเซอร์ ดังรูป 5.9



รูปที่ 5.9 หลังจากการทำ Follow TCP stream เพื่อแสดงการติดตามการส่ง HTTP request ไปร้องขอไฟล์แอพเพลตและการส่ง HTTP response ตอบกลับมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แสดงการติดตามการรับส่งข้อมูลของทีซีพีสตรีม ซึ่งแสดงให้เห็นการติดต่อสื่อสาร กันระหว่างเว็บบราวเซอร์และเราเตอร์โดยติดต่อผ่านเทอร์มินัลเซิร์ฟเวอร์ โดยแสดงดังรูปที่ 5.10

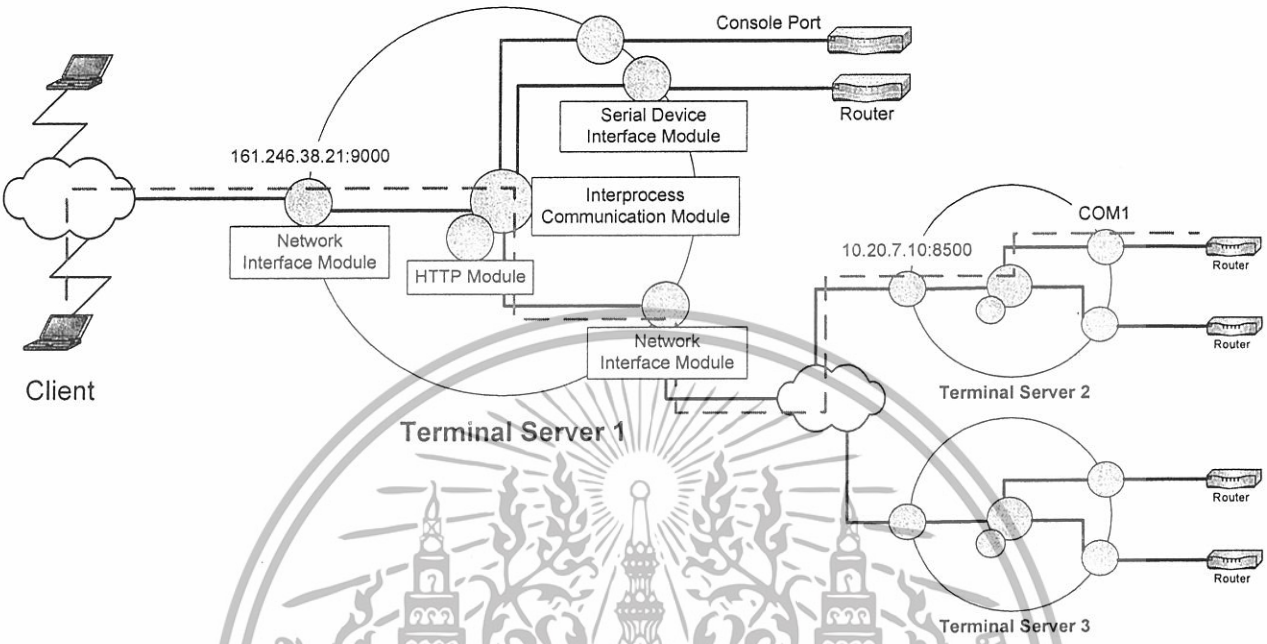


รูปที่ 5.10 หลังจากการทำ Follow TCP stream เพื่อแสดงการติดตามรับส่งข้อมูลกันระหว่างเว็บบราวเซอร์กับเราเตอร์ผ่านเทอร์มินัลเซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.2 การทดสอบแบบที่ 2

ทดสอบเพื่อยืนยันว่าสามารถส่งต่อการทำงานไปยังเทอร์มินัลเซิร์ฟเวอร์อื่นได้จริง



รูปที่ 5.11 ภาพรวมของการทดสอบแบบที่ 2

ตารางที่ 5.2 ตารางกำหนดเส้นทางที่ใช้ในการตัดสินใจของเทอร์มินัลเซิร์ฟเวอร์ 1

IP	PORT	IP	PORT
161.246.38.21	8000	-	COM1
161.246.38.21	8001	-	COM2
161.246.38.21	9000	10.20.7.10	8500
161.246.38.21	9001	10.20.7.10	8501

ตารางที่ 5.3 ตารางกำหนดเส้นทางที่ใช้ในการตัดสินใจของเทอร์มินัลเซิร์ฟเวอร์ 2

IP	PORT	IP	PORT
10.20.7.10	8500	-	COM1
10.20.7.10	8501	-	COM2

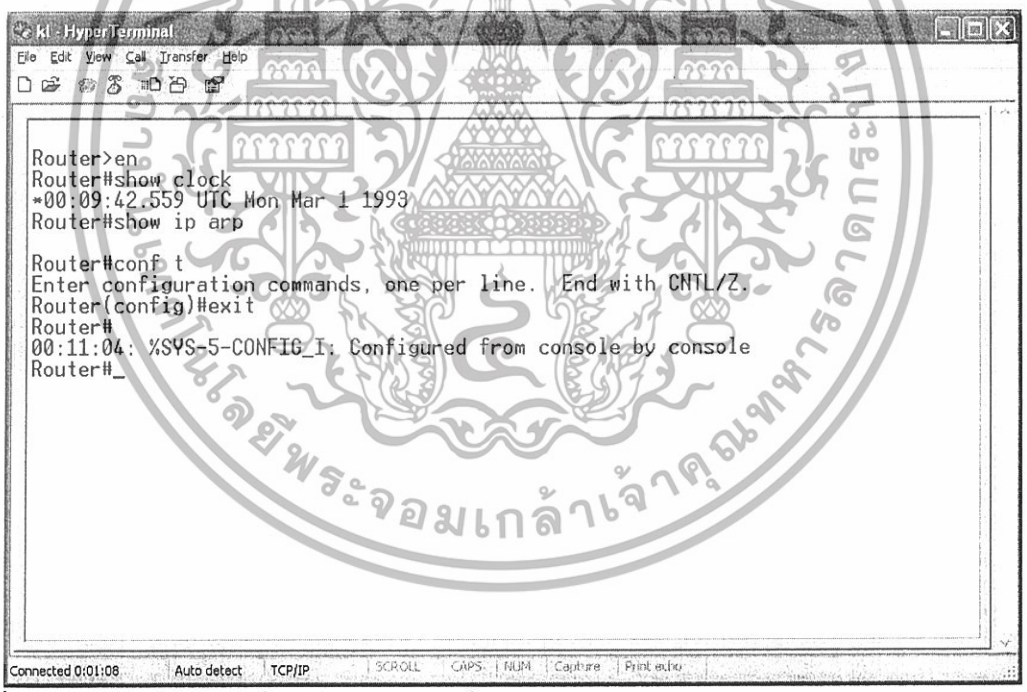
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รายละเอียดการทดสอบแบบที่ 2

การทดลองนี้จะแสดงให้เห็นถึงการทำงานที่สามารถติดต่อกับเทอร์มินัลเซิร์ฟเวอร์ตัวอื่นได้ โดยจะให้ไคลเอนต์ทำการติดต่อจากระยะไกล เพื่อติดต่อกับเราเตอร์ที่ต่ออยู่กับเทอร์มินัลเซิร์ฟเวอร์ 2 (เทอร์มินัลเซิร์ฟเวอร์ไอพี 10.20.7.10) โดยการติดต่อผ่านเทอร์มินัลเซิร์ฟเวอร์ 1 (เทอร์มินัลเซิร์ฟเวอร์ไอพี 161.246.38.21) ทางพอร์ต 9000 เมื่อไคลเอนต์ส่งข้อมูลเข้ามายังเทอร์มินัลเซิร์ฟเวอร์ 1 ข้อมูลจะถูกส่งไปยังเทอร์มินัลเซิร์ฟเวอร์ 2 ทางพอร์ต 8500 จากนั้นเทอร์มินัลเซิร์ฟเวอร์ 2 จะส่งต่อไปยังเราเตอร์ต่อไป ในขณะเดียวกัน เมื่อเราเตอร์ส่งข้อมูลกลับมาเทอร์มินัลเซิร์ฟเวอร์ 2 จะส่งออกทางพอร์ต 8500 ไปยังเทอร์มินัลเซิร์ฟเวอร์ 1 เพื่อส่งให้ไคลเอนต์ทางพอร์ต 9000 ต่อไป

ผลการทดสอบแบบที่ 2

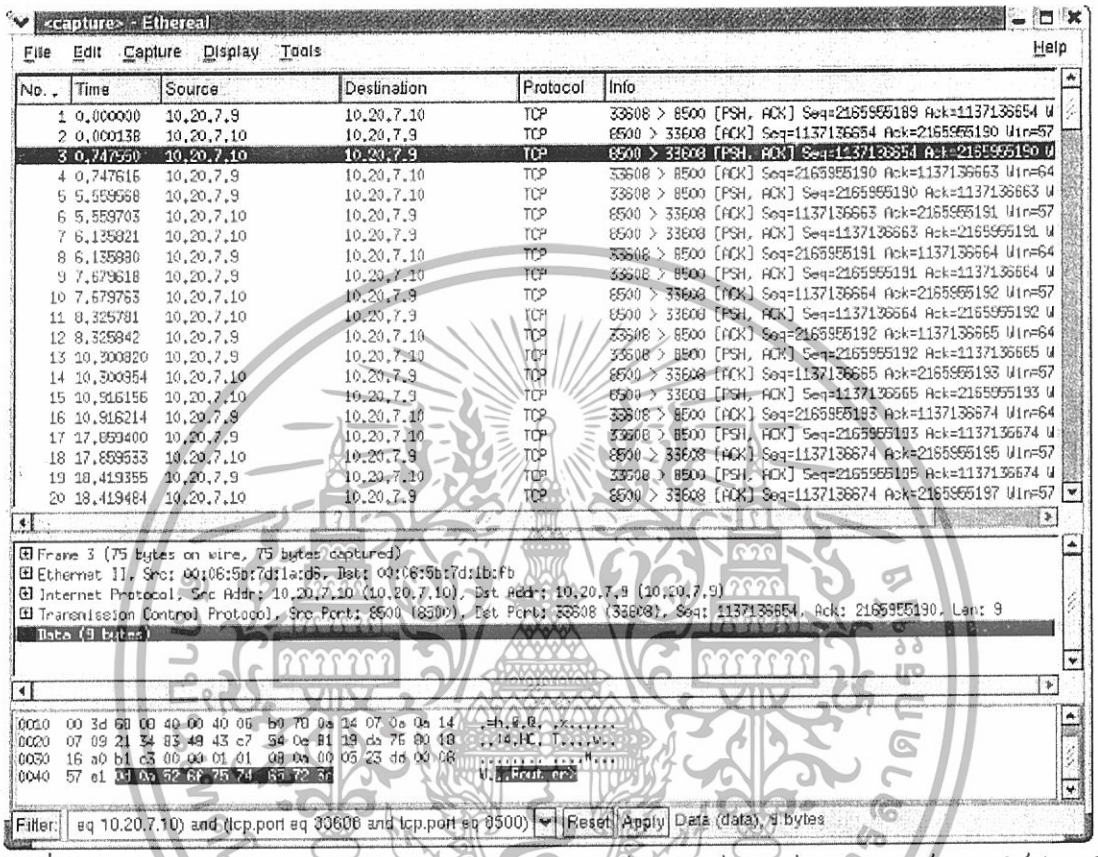
เมื่อทำการติดต่อจากระยะไกลด้วยโปรแกรมไฮเพอร์ เทอร์มินัลไปยังเราเตอร์ที่ต่ออยู่กับเทอร์มินัลเซิร์ฟเวอร์ 2 โดยผ่านทางเทอร์มินัลเซิร์ฟเวอร์ 1 ได้ผลดังรูปที่ 5.12



รูปที่ 5.12 แสดงการติดต่อสื่อสารกับเราเตอร์ที่ต่อกับเทอร์มินัลเซิร์ฟเวอร์ 2 โดยใช้โปรแกรมไฮเพอร์เทอร์มินัล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยเราได้ทำการดักจับข้อมูลที่เน็ตเวิร์คพอร์ตอีเกรส (Network port engress) เพื่อทำการยืนยันว่ามีการรับส่งข้อมูลระหว่างเทอร์มินัลเซิร์ฟเวอร์ 1 (Network Interface Module ไอพี 10.20.7.9) และเทอร์มินัลเซิร์ฟเวอร์ 2 (Network Interface Module ไอพี 10.20.7.10) เกิดขึ้นจริง ได้ผลดังรูปที่ 5.13



รูปที่ 5.13 ผลการดักจับข้อมูลการติดต่อกันระหว่างเทอร์มินัลเซิร์ฟเวอร์ 1 และเทอร์มินัลเซิร์ฟเวอร์ 2 โดยใช้โปรแกรมอีเทอร์เรียล

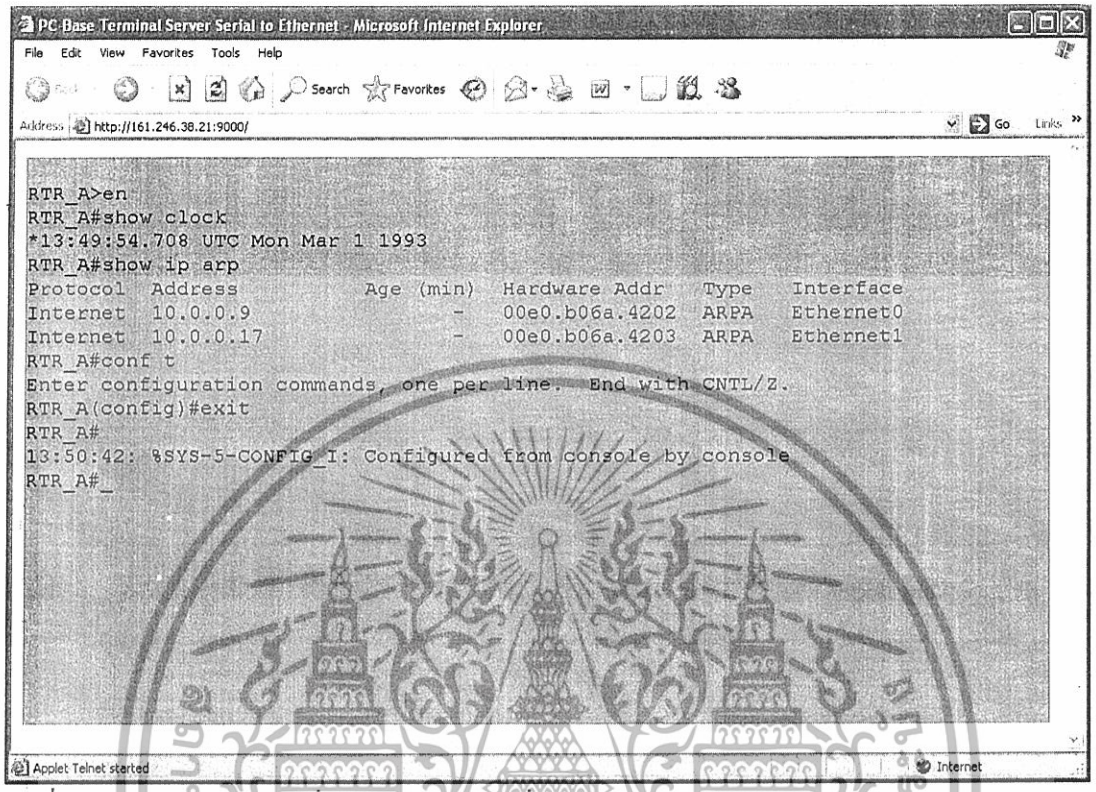
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลังจากดักจับข้อมูลแล้วได้ทำการติดตามการรับส่งข้อมูลของ ทีซีพีสตรีมจะแสดงให้เห็นการติดต่อสื่อสารกันระหว่างไฮเพอร์เทอร์มินัลและเราเตอร์ที่ต่ออยู่กับเทอร์นินัลเซิร์ฟเวอร์ 2 โดยติดต่อผ่านทางเทอร์นินัลเซิร์ฟเวอร์ 1 ผลดังรูปที่ 5.14

รูปที่ 5.14 หลังจากการทำ Follow TCP stream เพื่อแสดงการติดตามการติดต่อสื่อสารกันระหว่างไฮเพอร์เทอร์มินัลกับเราเตอร์ที่ต่ออยู่กับเทอร์นินัลเซิร์ฟเวอร์ 2 โดยติดต่อผ่านทางเทอร์นินัลเซิร์ฟเวอร์ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

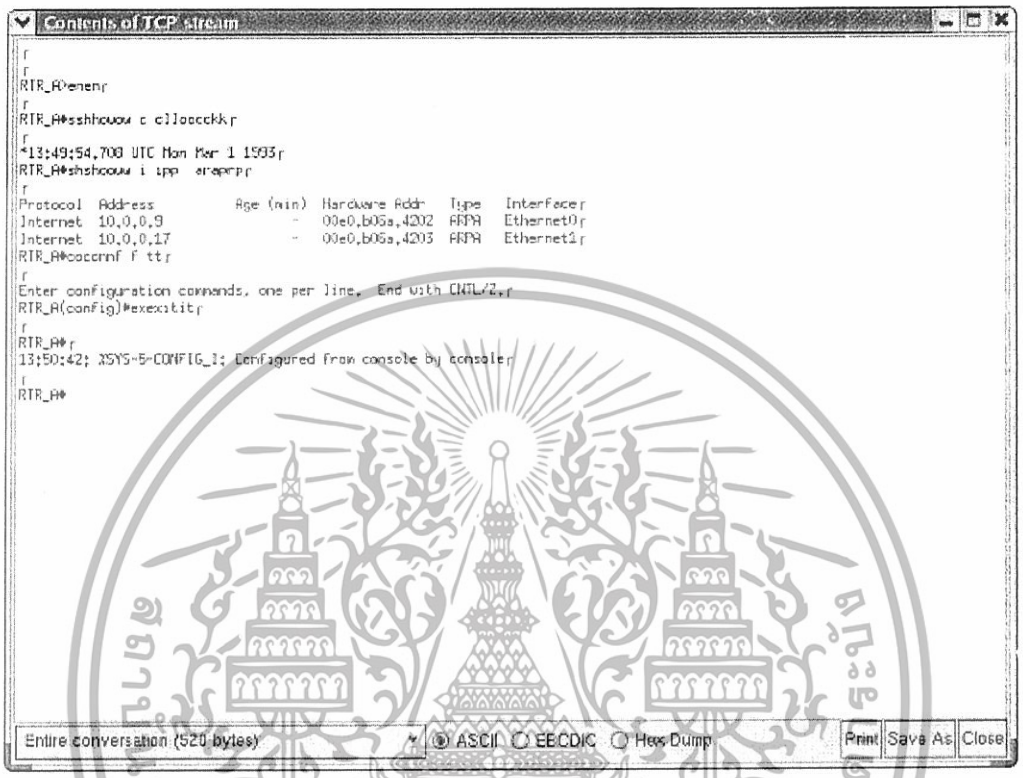
ในลักษณะเดียวกัน เมื่อทำการติดต่อจากระยะไกลด้วยเว็บบราวเซอร์ไปยังเราเตอร์ที่ต่ออยู่กับเทอร์นินัลเซิร์ฟเวอร์ 2 โดยติดต่อผ่านทางเทอร์นินัลเซิร์ฟเวอร์ 1 ได้ผลดังรูปที่ 5.15



รูปที่ 5.15 แสดงการติดต่อสื่อสารกับเราเตอร์ที่ต่ออยู่กับเทอร์นินัลเซิร์ฟเวอร์ 2 โดยติดต่อผ่านทางเทอร์นินัลเซิร์ฟเวอร์ 1 ด้วยโปรแกรมเว็บบราวเซอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แสดงการติดตามการรับส่งข้อมูลของทีซีพีสตรีมจะแสดงให้เห็นการติดต่อสื่อสาร กันระหว่างเว็บบราวเซอร์และเราเตอร์ที่ต่ออยู่กับเทอร์นินัลเซิร์ฟเวอร์ 2 โดยติดต่อผ่านทางเทอร์นินัลเซิร์ฟเวอร์ 1 แสดงผลดังรูปที่ 5.16



รูปที่ 5.16 หลังจากการทำ Follow TCP stream เพื่อแสดงการติดตามการติดต่อสื่อสารระหว่าง เว็บบราวเซอร์กับเราเตอร์ที่ต่อกับเทอร์มินัลเซิร์ฟเวอร์ 2 โดยผ่านเทอร์มินัลเซิร์ฟเวอร์ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6
บทสรุป

6.1 ผลจากการดำเนินการ

จากการพัฒนาพีซีเทอร์มินัลเซิร์ฟเวอร์เพื่อช่วยสนับสนุนการติดต่อจากระยะไกลผ่านระบบเครือข่ายอีเทอร์เน็ตมายังอุปกรณ์ที่ต่อพอร์ตอนุกรม ซึ่งพีซีเทอร์มินัลเซิร์ฟเวอร์นี้จะทำหน้าที่เป็นตัวกลางเชื่อมการติดต่อระหว่างผู้ใช้ที่ต้นทางจากระยะไกลกับอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรม โดยผู้ใช้สามารถเลือกอุปกรณ์ที่ต้องการติดต่อได้ด้วยการระบุหมายเลขพอร์ตในการติดต่อ

นอกจากนี้การพัฒนาระบบในลักษณะที่เป็นโมดูล ยังช่วยทำให้มีความยืดหยุ่นในการขยับขยายการทำงานของระบบได้ง่าย ซึ่งจะทำให้รองรับจำนวนอุปกรณ์ปลายทางได้โดยไม่มีขีดจำกัด

6.2 ประโยชน์ที่ได้รับ

6.2.1 ระบบสนับสนุนที่ช่วยให้เข้าถึงอุปกรณ์ที่เชื่อมต่อพอร์ตอนุกรม ผ่านเครือข่ายอีเทอร์เน็ตโดยการใช้ไอพีแอดเดรสและหมายเลขพอร์ตในการระบุ อ้างอิงถึงอุปกรณ์ปลายทางได้ ซึ่งช่วยให้ผู้ดูแลมีความสะดวกในดูแลและควบคุมอุปกรณ์ที่ต่อกับพอร์ตอนุกรม

6.2.2 ได้ต้นแบบของคอมโพแนนท์ในการควบคุมการรับส่งข้อมูลบนเครือข่ายอีเทอร์เน็ตผ่าน LAN Card และการรับส่งข้อมูลกับอุปกรณ์เชื่อมต่อผ่านพอร์ตอนุกรมทำให้สะดวกในการนำไปพัฒนาต่อไป

6.2.3 ได้แนวคิดเริ่มต้นในการพัฒนาระบบที่ใช้ในการเชื่อมโยงข้อมูลผ่านเครือข่ายอีเทอร์เน็ตหรือการรับส่งข้อมูลกับอุปกรณ์เชื่อมต่อผ่านพอร์ตอนุกรม

6.3 แผนการดำเนินงานในอนาคต

เมื่อเราพัฒนาต้นแบบสมบูรณ์ก็จะทำการสร้างระบบจัดการเพื่อให้ผู้ใช้จำนวนมากสามารถเข้าใช้อุปกรณ์อย่างเป็นระบบ รวมถึงระบบจัดการความปลอดภัยในการเข้าใช้อุปกรณ์ หรือนำคอมโพแนนท์ต้นแบบแต่ละโมดูลไปพัฒนาต่อในรูปแบบต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

InterprocessModule.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/msg.h>

#define NUM 20

struct my_q_st {
        long int mesg_ident;
        char some_text[BUFSIZ];
};


int way[NUM][2][2];             // Table [1][2][3] 1.way  2.Src. or Dest. Module 3.Go or
Back
int num_way = 0;

char *Cmd(char *cmd) {                                  // check cmd
        int i;
        int siz = strlen(cmd);

        *(cmd+(siz-2)) = '\0';


        if(strcmp(cmd, "q") == 0) {             // disconnect cmd
                return "46060008";
        } else if(strcmp(cmd, "b") == 0) {      // back from cmd mode cmd
                return "46060029";
        } else if(strcmp(cmd, "n") == 0) {      // end system cmd
                return "999999999";
        } else {
                return "Not Command\n\n";       // miss match cmd
        }


}

int chkCmd_Mode(char *cmd) {
        int cmd_mode = 0, i;



        if(*cmd == 94) {                        // check active cmd mode
                cmd_mode = 1;
        } else {
                cmd_mode = 0;
        }

        return cmd_mode;
}

int chkHttp(char *type, char *uri, char *ptc) {
        int http = 0;
        int i = 0;

        printf("IPC : type : %s\n", type);
        printf("IPC : uri : %s\n", uri);
        printf("IPC : ptc : %s\n", ptc);


        if(((strcmp(type, "GET"))||(strcmp(ptc, "HTTP/1.1"))) == 0) {       // check
Http request
                http = 1;
        } else {
                http = 0;
        }

        return http;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
int main() {
        char *table[NUM][2];                    // Declare Variable
        char *sys_cmd;
        char *IpAddr, *detect, *chk;
        char *type, *uri, *ptc;
        char *code;

        int end = 1, http = 0;
        int i, j, k;
        int go, back;                   // Key Msg. Queue
        int time[NUM];


        int cmd_mode[NUM];
        int port, ip[4];
        int rc_in, rc_out, rc_http, rc;
        int toHttp, fromHttp;
        long int msg_to_receive = 0;


        struct my_q_st data_fromq, data_toq;

        FILE *config;



        for(i=0; i<NUM; i++) {                  // Allocate String Variable
                for(j=0; j<2; j++) {
                        table[i][j] = (char *)malloc(BUFSIZ);
                }
                cmd_mode[i] = 0;
        }
        sys_cmd = (char *)malloc(BUFSIZ);
        IpAddr = (char *)malloc(BUFSIZ);
        chk = (char *)malloc(BUFSIZ);
        type = (char *)malloc(BUFSIZ);
        uri = (char *)malloc(BUFSIZ);
        ptc = (char *)malloc(BUFSIZ);
        code = (char *)malloc(BUFSIZ);



        if((config = fopen("Terminal_Config.txt", "r")) == NULL) {          // Open
Config File
                printf("Couldn't open Congig file");
                exit(1);
        }

        for(i=0; i<NUM; i++) {
                if(!feof(config)) {
                        fscanf(config, "%s %d\t-\t%s", table[i][0], &time[i],
table[i][1]);  // Read Config File
                        num_way++;
                }
        }

        num_way--;

        printf("IPC : num_way : %d\n" ,num_way);

        go = 10;                                // Mapp Msg. Queue for connect other
Module
        back = 20;

        for(i=0; i<num_way; i++) {
                for(j=0; j<2; j++) {

                        go += 100;
                        back += 100;

                        way[i][j][0] = msgget((key_t)go, 0666|IPC_CREAT);          //
Create Msg. Queue
                        way[i][j][1] = msgget((key_t)back, 0666|IPC_CREAT);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                        if(j == 0) {            // Src. module process

                                sprintf(sys_cmd, "./NetworkModule_3 %d %d %d %s
&",time[i] , go, back, table[i][j]);          // args - Key Output , Key Input , Port

                                /*DEBUG*/
                                //printf("%s\n", sys_cmd);     // Cmd to call NIM
                                system(sys_cmd);               // Call Src. module


                        } else {                        // Dest. module process

                                if(strcmp("COM1", table[i][j]) && strcmp("COM2",
table[i][j]) == 1) {   // Dest. = NIM

                                        sscanf(table[i][j], "%d.%d.%d.%d:%d", &ip[0],
&ip[1], &ip[2], &ip[3], &port);       // get ip address and port number
                                        sprintf(IpAddr, "%d.%d.%d.%d", ip[0], ip[1],
ip[2], ip[3]);


                                        sprintf(sys_cmd, "./ClientNetwork_3 %d %d %s %d
&", back, go, IpAddr, port);  // args - Key Output , Key Input , IP, Port

                                        system(sys_cmd);               // Call NIM

                                        /*DEBUG*/
                                        //printf("%s\n", sys_cmd);     // Cmd to call NIM


                                } else {                                // Dest. = SD.M

                                        sprintf(sys_cmd, "./SerialModule_3 %d %d %s &",
back, go, table[i][j]);       // args - Key Output , Key Input , Detect);

                                        system(sys_cmd);               // Call SDIM

                                        /*DEBUG*/
                                        //printf("%s\n", sys_cmd);     // Cmd to call SDIM
                                }
                        }
                }
        }


        go += 100;
        back += 100;

        toHttp = msgget((key_t)go, 0666|IPC_CREAT);  // create Message queue for HTTP
Module
        fromHttp = msgget((key_t)back, 0666|IPC_CREAT);


        sprintf(sys_cmd, "./HttpModule_3 %d %d &", back, go);
        system(sys_cmd);


        while(end) {            // Detect that Have Msg. in Queue?

                for(i=0; i<num_way; i++) {
                        rc_in = 0;
                        rc_out = 0;


                        rc_in = msgrcv(way[i][0][0], (void *)&data_fromq, BUFSIZ,
msg_to_receive, IPC_NOWAIT);  // Receive Info from Src Module
                        if(rc_in>0) {

                                strcpy(chk, data_fromq.some_text);

                                if(cmd_mode[i] == 0) {          // Check in cmd mode?

                                        cmd_mode[i] = chkCmd_Mode(chk);       // check
will active cmd mode?
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                                        /*DEBUG*/
                                        //printf("IPC : cmd mode : %d\n", cmd_mode[i]);
        // in Cmd mode ?
                                        if(cmd_mode[i] == 0) {

                                                sscanf(chk, "%s %s %s", type, uri, ptc);
                                                http = chkHttp(type, uri, ptc);       //
check Http request?

                                                /*DEBUG*/
                                                //printf("IPC : http %d\n", http);    //
is Http request ?


                                                if(http == 1) {                 // HTTP
request

                                                        data_toq.mesg_ident =
way[i][0][1];
                                                        strcpy(data_toq.some_text,
data_fromq.some_text);

                                                        rc = msgsnd(toHttp, (void
*)&data_toq, rc_in, 0);         // send to HTTP Module

                                                        /*DEBUG*/
                                                        //printf("IPC : send to Http
:\n%s", data_fromq.some_text);      // Info send to Http Module

                                                } else {


                                                        data_toq.mesg_ident = 1;
                                                        strcpy(data_toq.some_text,
data_fromq.some_text);


                                                        rc = msgsnd(way[i][1][0], (void
*)&data_toq, rc_in, 0);     // Send Info to Dest. Module

                                                        /*DEBUG*/
                                                        //printf("IPC : send to COM1
:\n%s\n", data_toq.some_text);      // Info send to SDIM
                                                }

                                        } else {        // active cmd mode

                                                data_toq.mesg_ident = 1;
                                                strcpy(data_toq.some_text, "\nCommand
mode >\n");

                                                rc = msgsnd(way[i][0][1], (void
*)&data_toq, strlen(data_toq.some_text), 0);

                                                /*DEBUG*/
                                                //printf("IPC : send to NIC :\n%s\n",
data_toq.some_text);   // Info send to NIM

                                        }

                                } else {                // in cmd mode

                                        code = Cmd(chk);                // cmd in cmd mode

                                        if(strcmp("999999999", code) == 0) {          //
end System

                                                data_toq.mesg_ident = 1;
                                                strcpy(data_toq.some_text, "999999999");

                                                for(i=0; i<num_way; i++) {

                                                        // send end code to dest, src
module and HTTP Module

                                                        rc = msgsnd(way[i][0][1], (void
*)&data_toq, strlen(data_toq.some_text), 0);

                                                        rc = msgsnd(way[i][1][0], (void
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
*)&data_toq, strlen(data_toq.some_text), 0);
                                                }

                                                rc = msgsnd(toHttp, (void *)&data_toq,
strlen(data_toq.some_text), 0);
                                                end = 0;
                                                break;


                                        } else if(strcmp("46060029", code) == 0) {    //
send back from cmd mode to src module

                                                data_toq.mesg_ident = 1;
                                                strcpy(data_toq.some_text, "\nActive mode
>\n");

                                                rc = msgsnd(way[i][0][1], (void
*)&data_toq, strlen(data_toq.some_text), 0);

                                                cmd_mode[i] = 0;

                                        } else if(strcmp("46060008", code) == 0) {    //
send disconnect code to NIM

                                                data_toq.mesg_ident = 1;
                                                strcpy(data_toq.some_text, "Connection
Close!!\n");

                                                rc = msgsnd(way[i][0][1], (void
*)&data_toq, strlen(data_toq.some_text), 0); // Send Info into Msg. Queue


                                                data_toq.mesg_ident = 1;
                                                strcpy(data_toq.some_text, code);

                                                rc = msgsnd(way[i][0][1], (void
*)&data_toq, strlen(data_toq.some_text), 0); // Send Info into Msg. Queue

                                                cmd_mode[i] = 0;


                                        } else {

                                                data_toq.mesg_ident = 1;
                                                strcpy(data_toq.some_text, code);

                                                rc = msgsnd(way[i][0][1], (void
*)&data_toq, strlen(code), 0);

                                        }
                                }

                        }


                        rc_out = msgrcv(way[i][1][1], (void *)&data_fromq, BUFSIZ,
msg_to_receive, IPC_NOWAIT);  // Receive Info from Dest Module
                        if(rc_out>0) {

                                data_toq.mesg_ident = 1;
                                strcpy(data_toq.some_text, data_fromq.some_text);

                                rc = msgsnd(way[i][0][1], (void *)&data_toq, rc_out, 0);
        // Send Info into Msg. Queue
                                //printf("IPC : send to NIC :\n%s\n",
data_toq.some_text);

                        }


                        rc_http = msgrcv(fromHttp, (void *)&data_fromq, BUFSIZ,
msg_to_receive, IPC_NOWAIT);  // Receive Info from HTTP Module

                        if(rc_http>0) {

                                rc = msgsnd(data_fromq.mesg_ident, (void *)&data_fromq,
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
rc_http, 0);

                                //printf("IPC : receive from http :\n%d\n%s\n",
data_fromq.mesg_ident, data_fromq.some_text);

                        }


                        bzero((void *)data_fromq.some_text, BUFSIZ);
                        bzero((void *)data_toq.some_text, BUFSIZ);
                        bzero((void *)type, BUFSIZ);
                        bzero((void *)uri, BUFSIZ);
                        bzero((void *)ptc, BUFSIZ);
                        bzero((void *)chk, BUFSIZ);

                }
        }



        printf("Ending...\n");
        sleep(5);

        for(i=0; i<NUM; i++) {                  // Free String Variable
                for(j=0; j<2; j++) {
                        free(table[i][j]);
                }
        }

        for(i=0; i<num_way; i++) {
                for(j=0; j<2; j++) {
                        msgctl(way[i][j][0], IPC_RMID, 0);
                        msgctl(way[i][j][1], IPC_RMID, 0);
                }
        }
        msgctl(toHttp, IPC_RMID, 0);            // delete message queue
        msgctl(fromHttp, IPC_RMID, 0);

        free(sys_cmd);
        free(IpAddr);
        free(chk);
        free(type);
        free(uri);
        free(ptc);

        printf("Finished\n\n");

        exit(0);
}
```

NetworkModule.c

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include<unistd.h>
#include<time.h>

#include<sys/ipc.h>
#include<sys/msg.h>

#define BACKLOG 0       //Backlog Queue
#define TIME_OUT 30

struct my_q_st {
        long int mesg_ident;
        char some_text[BUFSIZ];
};

int main(int argc, char **argv) {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        int sockfd, new_fd;                     //Socket Description | sock -> Parent |
newfd -> child
        struct sockaddr_in server_addr;         //My Server Address
        struct sockaddr_in client_addr;         //Client Address
        int sin_size;
        int out, in, port;

        int output, input, rq, rn, i,j;
        struct my_q_st data_toq, data_fromq;
        long int msg_to_receive = 0;

        int disc = 1;
        int end = 1;

        time_t  lastTime;
        time_t  currentTime;

        char *temp = (char *)malloc(BUFSIZ);
        char *buffer = (char *)malloc(BUFSIZ);
        char *chk = (char *)malloc(BUFSIZ);

        int wait;

        wait = atoi(argv[1]);
        /// default time

        out = atoi(argv[2]);
        in = atoi(argv[3]);
        port = atoi(argv[4]);


        /* create Message queue */
        output = msgget((key_t)out, 0666 | IPC_CREAT);
        input = msgget((key_t)in, 0666 | IPC_CREAT);

        /* create Socket */
        if((sockfd = socket(PF_INET, SOCK_STREAM, 0)) == -1) {
                perror("Socket Error");
                exit(1);
        }

        /* Bind Socket */
        server_addr.sin_family = AF_INET;                       //Host Byte
        server_addr.sin_port = htons(port);                     //Short Network
        server_addr.sin_addr.s_addr = INADDR_ANY;               //Auto-Fill My IP
        bzero(&(server_addr.sin_zero), 8);                      //Fill all with Zero

        if(bind(sockfd, (struct sockaddr *)&server_addr, sizeof(struct sockaddr)) == -
1) {
                perror("Bind Error");
                exit(1);
        }

        /* Listen -> create queue connect */
        if(listen(sockfd, 0) == -1) {
                perror("Listen Error");
                exit(1);
        }



      while(end) {

                disc = 1;

                printf("Waiting at Port  %d ...\n", port);


                /* Wait client connect will accept */
                sin_size = sizeof(struct sockaddr_in);
                if ((new_fd = accept(sockfd, (struct sockaddr *)&client_addr,
&sin_size)) == -1) {
                        perror("Accept Error");
                        continue;
                }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            printf("Server: Accept connection from Client IP: %s\n\n",
inet_ntoa(client_addr.sin_addr));


            lastTime = time((time_t *)0);
            currentTime = time((time_t *)0);

            while(disc) {  //


                  if(wait != 0) {

                        if((currentTime - lastTime) < wait) {
      // Check : Client don't Active long time?


                              //printf("Wait  %d\n", time);

                              rn = 0;

                              rn = recv(new_fd, buffer, BUFSIZ, MSG_DONTWAIT);
      // recv info from client
                              if(rn > 0) {
            // when recv have info

                                    data_toq.mesg_ident = 1;
            // create block info
                                    strcpy(data_toq.some_text, buffer);

                                    /*DEBUG*/
                                    //printf("NIC : send to IPC : %s\n",
data_toq.some_text);    // info send to ipcm


                                    rq = msgsnd(output, (void *)&data_toq,
rn, 0); // send info to IPC by Message queue

                                    lastTime = time((time_t *)0);

                              }


                              rq = msgrcv(input, (void *)&data_fromq, BUFSIZ,
msg_to_receive, IPC_NOWAIT);  // recv info from IPC by Message queue
                              if(rq > 0) {    // when recv have info

                                    /*DEBUG*/
                                    //printf("NIM : receive from IPC : %s\n",
data_fromq.some_text);        // info info send to IPCM


                                    strcpy(chk, data_fromq.some_text);

                                    if(strcmp(chk, "46060008") == 0) {
            // Check : info is Disconnect code?
                                          disc = 0;
                                          break;
                                    } else if(strcmp(chk, "999999999") == 0)
{       // Check : info is End code?
                                          end = 0;
                                          break;
                                    }


                                    send(new_fd, data_fromq.some_text , rq,
0);     // send info to client

                                    lastTime = time((time_t *)0);
                              }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                                currentTime = time((time_t *)0);
                        } else {

                                printf("Close Connection\n");
                                break;

                        }
                } else {


                        //printf("no wait\n");


                        rn = 0;

                        rn = recv(new_fd, buffer, BUFSIZ, MSG_DONTWAIT);     //
recv info from client
                        if(rn > 0) {
        // when recv have info

                                data_toq.mesg_ident = 1;
        // create block info
                                strcpy(data_toq.some_text, buffer);

                                //printf("NIC : send to IPC : %s\n",
data_toq.some_text);


                                rq = msgsnd(output, (void *)&data_toq, rn, 0);
        // send info to IPC by Message queue

                        }



                        rq = msgrcv(input, (void *)&data_fromq, BUFSIZ,
msg_to_receive, IPC_NOWAIT);  // recv info from IPC by Message queue
                        if(rq > 0) {    // when recv have info

                                //printf("NIM : receive from IPC : %s\n",
data_fromq.some_text);

                                /*for(i=0; i<strlen(data_fromq.some_text); i++) {
                                        printf("NIC : NW code : %d \n",
*(data_fromq.some_text+i));
                                }*/

                                strcpy(chk, data_fromq.some_text);

                                if(strcmp(chk, "46060008") == 0) {
        // Check : info is Disconnect code?
                                        disc = 0;
                                        break;
                                } else if(strcmp(chk, "999999999") == 0) {   //
Check : info is End code?
                                        end = 0;
                                        break;
                                }

                                send(new_fd, data_fromq.some_text , rq, 0);  //
send info to client
                        }
                }

                bzero((void *)data_fromq.some_text, BUFSIZ);
                bzero((void *)buffer, BUFSIZ);
                bzero((void *)chk, BUFSIZ);
        }
        close(new_fd);
    }

    close(sockfd);
    free(temp);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        free(buffer);
        free(chk);
}
```

SerialModule.c

```
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<fcntl.h>
#include<errno.h>
#include<termios.h>
#include<stdlib.h>

#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<sys/ioctl.h>


struct my_q_st {
        long int mesg_ident;
        char some_text[BUFSIZ];
};


void writeport(int fd, char *data) {  // write info to Serial port
        int n, i;
        int len = strlen(data);


        for(i=0; i<strlen(data); i++) {
                if(*(data+i) == '\n') {
                        *(data+i) = '\0';
                }
        }

        for(i=0; i<strlen(data); i++) {
                printf("code : %d \n", *(data+i));
        }


        n = write(fd, data, strlen(data));
      if(n<0){
        printf("\nwrite failed\n");
      }
}

int print_status(int fd) {                          // check Serial device active
        int sta;
        int status;

      ioctl(fd, TIOCMGET, &status);

        if(status&TIOCM_RTS)    printf("RTS (request to send)\n");
        if(status&TIOCM_ST)     printf("Secondary TXD (transmit)\n");
        if(status&TIOCM_SR)     printf("Secondary RXD (receive)\n");
        if(status&TIOCM_CTS)    printf("CTS (clear to send)\n");
        if(status&TIOCM_CAR)    printf("DCD (data carrier detect)\n");
        if(status&TIOCM_CD)     printf("Synonym for TIOCM_CAR\n");
        if(status&TIOCM_RNG)    printf("RNG (ring)\n");
        if(status&TIOCM_RI)     printf("Synonym for TIOCM_RNG\n");
        if(status&TIOCM_DSR)    printf("DSR (data set ready)\n");
        if(status&TIOCM_DTR)    printf("DTR (data terminal ready)\n");


        if((status&TIOCM_DTR)&&(status&TIOCM_CTS)&&(status&TIOCM_RTS)) {
                sta = 1;
        } else {
                sta = 0;
        }

        return(sta);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
int main(int argc, char *argv[]) {

        int fd, i;
        int parity;
        int baud;
        int status;

        int input, output;
        int rq, rs;
        int in, out;
        int end = 1;

        long int msg_to_receive = 0;

        struct my_q_st data_fromq, data_toq;
        struct termios options;

        char *data = (char*)malloc(BUFSIZ);
        char *detect = (char*)malloc(BUFSIZ);
        char *path = (char*)malloc(BUFSIZ);
        char *chk = (char*)malloc(BUFSIZ);



        out = atoi(argv[1]);
        in = atoi(argv[2]);
        strcpy(detect, argv[3]);

        output = msgget((key_t)out, 0666 | IPC_CREAT);
        input = msgget((key_t)in, 0666 | IPC_CREAT);

        if(strcmp(detect, "COM1") == 0) {
                strcpy(path, "/dev/ttyS0");
        } else if(strcmp(detect, "COM2") == 0) {
                strcpy(path, "/dev/ttyS1");
        }


        fd = open(path, O_RDWR|O_NOCTTY|O_NDELAY );  // initial info Serial device
        if(fd == -1) {
                perror("open_port:Unble to open /dev/ttyS0 -");
        }

        fcntl(fd,F_SETFL,0);
        tcgetattr(fd,&options);

        baud = 9600;

        switch(baud) {
                case 1200:
                        cfsetispeed(&options,B1200);
                        cfsetospeed(&options,B1200);
                        break;
                case 2400:
                        cfsetispeed(&options,B2400);
                        cfsetospeed(&options,B2400);
                        break;
                case 4800:
                        cfsetispeed(&options,B4800);
                        cfsetospeed(&options,B4800);
                        break;
                case 9600:
                        cfsetispeed(&options,B9600);
                        cfsetospeed(&options,B9600);
                        break;
                case 19200:
                        cfsetispeed(&options,B19200);
                        cfsetospeed(&options,B19200);
                        break;
                default:
                        perror("Unsupported baud rate\n");
                        //return(false);
        }

        parity = 1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        switch(parity) {
                case 1:
                                options.c_cflag |=(CLOCAL|CREAD);
                                options.c_cflag &= ~PARENB;
                                options.c_cflag &= ~CSTOPB;
                                options.c_cflag &= ~CSIZE;
                                options.c_cflag |= CS8;

                                options.c_cflag |=CRTSCTS;
                                options.c_iflag &=~(IXON |IXOFF);
                                options.c_iflag =IGNPAR;
                                options.c_lflag &= ~(ICANON|ECHO|ECHOE|ISIG);
                                options.c_oflag &= ~OPOST;
                                options.c_cc[VMIN]=0;
                                options.c_cc[VTIME]=1;
                                break;
                case 2:
                                options.c_cflag |=(CLOCAL|CREAD);
                                //ENABLE THE receiver and set local made

                                options.c_cflag |=PARENB;
                                options.c_cflag &=~PARODD;
                                options.c_cflag &=~CSTOPB;
                                options.c_cflag &=~CSIZE;
                                //printf("\nparity you select Even parity (7E1)\n");

                                break;

                default:
                                perror("Unsupport parity \n");
                                //return(false);
        }

        if(tcsetattr(fd,TCSANOW,&options) != 0) {
                perror("set the new option for the port");
        //return(false);
        }


        while(end) {
                rq = 0;
                rs = 0;


                rq = msgrcv(input, (void *)&data_fromq, BUFSIZ, msg_to_receive,
IPC_NOWAIT );
                if(rq > 0) {

                strcpy(chk, data_fromq.some_text);
                if(strcmp(chk, "999999999") == 0) {             // check end system
code?
                        end = 0;
                        break;
                }

                for(i=0; i<strlen(data_fromq.some_text); i++) {
                        printf("SDIM : recv code : %d \n",
*(data_fromq.some_text+i));
                }


                status = print_status(fd);      // check status Serial device
                if(status == 0) {                       // not active

                        data_toq.mesg_ident = 1;
                        strcpy(data_toq.some_text, "Serial Port Status not
Active!!\n");

                        rq = msgsnd(output, (void *)&data_toq,
strlen(data_toq.some_text), 0);

                } else {                                // active

                        writeport(fd, data_fromq.some_text);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                }
        }


        rs = read(fd, data, 1024);              // read from serial device
        if(rs > 0) {

                data_toq.mesg_ident = 1;
                strcpy(data_toq.some_text, data);


                printf("SDIM : read from serial : %s", data_toq.some_text);

                rs = msgsnd(output, (void *)&data_toq, rs, 0);

        }

        bzero((void *)data_fromq.some_text, BUFSIZ);
        bzero((void *)data_toq.some_text, BUFSIZ);
        bzero((void *)data, BUFSIZ);
        bzero((void *)chk, BUFSIZ);
    }

    free(data);
    free(detect);
    free(path);
    free(chk);
}
```

ClientNetwork.c

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include<unistd.h>

#include<sys/ipc.h>
#include<sys/msg.h>


struct my_q_st {
        long int mesg_ident;
        char some_text[BUFSIZ];
};

int main(int argc, char *argv[]) {

        int input, output, rq, rn;
        int in, out, port;
        int end = 1;

        int sockfd, new_fd;  //Socket Description | sock -> Parent | newfd -> child

        struct sockaddr_in server_addr;       //My Server Address
        struct sockaddr_in client_addr;       //Client Address
        struct my_q_st data_fromq, data_toq;

        long int msg_to_receive = 0;



        char *IpAddr = (char *)malloc(BUFSIZ);
        char *buffer = (char *)malloc(BUFSIZ);
        char *chk = (char *)malloc(BUFSIZ);

        out = atoi(argv[1]);
        in = atoi(argv[2]);
        strcpy(IpAddr, argv[3]);
        port = atoi(argv[4]);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        output = msgget((key_t)out, 0666 | IPC_CREAT);       // Create message queue
        input = msgget((key_t)in, 0666 | IPC_CREAT);


        if ((sockfd = socket(PF_INET, SOCK_STREAM, 0)) == -1) {
             perror("Socket Error");
             exit(1);
       }


       server_addr.sin_family = AF_INET;                                    //Host Byte
       server_addr.sin_port = htons(port);                                  //Short Network
       server_addr.sin_addr.s_addr = inet_addr(IpAddr);      //Auto-Fill My IP

       if (connect(sockfd, (struct sockaddr *)&server_addr, sizeof(struct sockaddr)) ==
-1) {  // connect to NIM
          perror("Connect Error");
          exit(1);
       }


        while(end) {
                rq = 0;
                rn = 0;



                rq = msgrcv(input, (void *)&data_fromq, BUFSIZ, msg_to_receive,
IPC_NOWAIT );          // recv form IPC
                if(rq>0)
                {
                     printf("Plus receive : %s \n",data_fromq.some_text);

                    strcpy(chk, data_fromq.some_text);
                    if(strcmp(chk, "999999999") == 0) {
                       end = 0;

                       send(sockfd, "^", strlen("^"), 0);
                       sleep(3);
                       send(sockfd, "q", strlen("q"), 0);
                       sleep(3);
                       break;
                    }

                    send(sockfd, data_fromq.some_text, strlen(data_fromq.some_text),
0);
                }



                rn = recv(sockfd, buffer, BUFSIZ, MSG_DONTWAIT);             // recv
form NIM
                if(rn > 0) {

                        data_toq.mesg_ident = 1;
                        sprintf(data_toq.some_text, "%s", buffer);

                        printf("Plus send : %s \n", data_toq.some_text);

                        rn = msgsnd(output, (void *)&data_toq, BUFSIZ, 0);
                }



                bzero((void *)data_fromq.some_text, strlen(data_fromq.some_text));
                bzero((void *)buffer, BUFSIZ);
                bzero((void *)chk, BUFSIZ);

        }

       free(IpAddr);
       free(buffer);
       free(chk);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

HttpModule.c

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>

#include<sys/ipc.h>
#include<sys/msg.h>


#include<sys/stat.h>
#include<sys/types.h>

#include<fcntl.h>
#include<errno.h>
#include<termios.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/msg.h>
#include <time.h>

struct my_q_st {
        long int mesg_ident;
        char some_text[BUFSIZ];
};

#define MAX_SIZE 1024


int main(int argc, char **argv){

        int input, output;
        int a, s, i, j;
        int rc_in, rc_out;
        int httpok = 0;
        int size;
        int out, in, port, portcpy;
        int sub[4];
        int num_read;
        int end = 1;
        long int msg_to_receive = 0;
        long int dest;

        struct my_q_st data_toq, data_fromq;
        struct stat f_s, f_sa;

        char* temp = (char *)malloc(MAX_SIZE);
        char* html = (char*)malloc(MAX_SIZE);
        char* ip = (char*)malloc(MAX_SIZE);
        char* host = (char*)malloc(MAX_SIZE);
        char* response = (char*)malloc(MAX_SIZE);
        char *buffer = (char *)malloc(BUFSIZ);

        char cmd[10];
        char param1[20];
        char param2[20];

        time_t rawtime;
        struct tm *timeinfo;
        char timef[80];

        FILE *fp, *fclass;



        out = atoi(argv[1]);
        in = atoi(argv[2]);

        output = msgget((key_t)out, 0666 | IPC_CREAT);
        input = msgget((key_t)in, 0666 | IPC_CREAT);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        while(end){
                bzero((void *)data_fromq.some_text, BUFSIZ);
                bzero((void *)data_toq.some_text, BUFSIZ);
                bzero((void *)response, MAX_SIZE);
                bzero((void *)host, MAX_SIZE);

                rc_in = msgrcv(input, (void *)&data_fromq, BUFSIZ, msg_to_receive,
IPC_NOWAIT);// Receive Info from Msg (NIC)
                if(rc_in > 0){
                        printf("HTTP : receive from IPC :\n%s\n\n",
data_fromq.some_text);

                        dest = data_fromq.mesg_ident;
                        strcpy(temp, data_fromq.some_text);

                        if(strcmp(temp, "999999999") == 0) {
                                end = 0;
                                break;
                        }


                        sscanf(temp, "%s %s", cmd, param1);


        if(strcmp(cmd, "GET") == 0) {
                if(strcmp(param1, "/") == 0) {
                        if((fp = fopen("index.html", "rt")) == NULL) {
                                printf("cannot open fileindex.html\n");
                                httpok = 0;
                        }else {
                                httpok = 1;

                                stat("index.html", &f_s);
                                size = f_s.st_size;

                                bzero((void *)html, strlen(html));

                                while(!feof (fp)) {
                                        fread(html, 1, size, fp);       //Ji
                                }

                                for(i=0; i<strlen(temp); i++) {
                                        if((*(temp+i) == 'H')&&
                                        (*(temp+(i+1)) == 'o')&&
                                        (*(temp+(i+2)) == 's')&&
                                        (*(temp+(i+3)) == 't')&&
                                        (*(temp+(i+4)) == ':')&&
                                        (*(temp+(i+5)) == ' ')) {

                                                a = i+6;
                                                for(s=0; s<20; s++) {
                                                        if(*(temp+(s+a)) != '\r')
{
                                                                sprintf(host,
"%s%c", host, *(temp+(s+a)));
                                                        } else {
                                                                break;
                                                        }
                                                }

                                                //printf("HOST : %s\n\n", host);

                                                sscanf(host, "%d.%d.%d.%d:%d",
&sub[0], &sub[1], &sub[2], &sub[3], &portcpy);
                                                sprintf(ip, "%d.%d.%d.%d", sub[0],
sub[1], sub[2], sub[3]);

                                        }
                                }//for host

                                /*printf("IP : %s\n", ip);
                                printf("port : %d\n", portcpy);*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                                        sprintf(html, "%s\t\t\t<PARAM name=\"HOST\"
value=\"%s\">", html, ip);
                                        sprintf(html, "%s\n\t\t\t<PARAM name=\"PORT\"
value=\"%d\">", html, portcpy);
                                        sprintf(html,
"%s\n\t\t</APPLET>\n\t</BODY>\n</HTML>", html);

                                        //printf("HTML : %s\n", html);

                                        fclose(fp);
                                }
                } else {

                        strcpy(param2, ".");
                        sprintf(param2, "%s%s", param2, param1);

                        /*DEBUG*/
                        //printf("URI : %s\n", param2);

                        if((fclass = fopen(param2, "rb")) == NULL)    {

                                httpok = 0;
                                printf("cannot open file\n");
                                //exit(0);
                        } else {

                                httpok = 2;

                                stat(param2, &f_s);
                                size = f_s.st_size;
                        }

                }//else next to sw


        printf("httpok : %d\n",httpok);

        switch(httpok) {
                case 1:
                        printf("Send File HTML\n");

                        strcpy(response, "HTTP/1.1 200 OK\r\n");

                        time (&rawtime);
                        timeinfo = localtime(&rawtime);
                        strftime(timef,80,"Date: %d, %d-%b-%Y %H:%M:%S
GMT\r\n",timeinfo);

                        strcat(response, timef);
                        strcat(response, "Server: Custom Server\r\n");
                        strcat(response, "Last-Modified: Thu, 13 Fab 2007 17:47:30
GMT\r\n");
                        //strcat(response, "ETag: "20df40c-1ab-b4474f80"\r\n");
                        strcat(response, "Accept-Ranges: byte\r\n");
                        strcat(response,"Content-Length:");
                        sprintf(response, "%s %d\r\n", response, size);
                        strcat(response, "Keep-Alive: timeout=15, max=100\r\n");
                         strcat(response, "Connection: keep-alive\r\n");
                        strcat(response, "Content-type: text/html\r\n");
                        strcat(response, "\r\n");

                        strcat(response, html);
                         strcat(response,"\r\n");

                        data_toq.mesg_ident = dest;
                        strcpy(data_toq.some_text, response);

                        rc_out = msgsnd(output, (void *)&data_toq, strlen(response), 0);

                        /*DEBUG*/
                        //printf("HTTP : RESPONSE :\n %s", data_toq.some_text);      //
reponse

                        data_toq.mesg_ident = dest;
                        strcpy(data_toq.some_text,"46060008");
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                        rc_out = msgsnd(output, (void *)&data_toq,
strlen(data_toq.some_text), 0);

                        break;

                case 2:
                        printf("Send File .class\n");

                        //send header
                        strcpy(response,"HTTP/1.1 200 OK\r\n");

                        time (&rawtime);
                        timeinfo = localtime(&rawtime);
                        strftime(timef,80,"Date: %d, %d-%b-%Y %H:%M:%S
GMT\r\n",timeinfo);

                        strcat(response, timef);
                        strcat(response,"Server: Custom Server\r\n");
                        strcat(response, "Last-Modified: Thu, 13 Fab 2007 17:47:30
GMT\r\n");
                        //strcat(response,"ETag: w/"20dcbf8-d41-68f0b340"\r\n");
                        strcat(response,"Accept-Ranges: byte\r\n");
                        strcat(response,"Content-Length:");
                        sprintf(response, "%s %d\r\n", response, size);
                        strcat(response,"Keep-Alive: timeout=15, max=100\r\n");
                         strcat(response,"Connection: Keep-Alive\r\n");
                        strcat(response,"Content-Type: application/octet-stream\r\n");

                        strcat(response,"\r\n");

                        data_toq.mesg_ident = dest;
                        strcpy(data_toq.some_text, response);

                        rc_out = msgsnd(output, (void *)&data_toq, strlen(response), 0);

                        /*DEBUG*/
                        //printf("HTTP : RESPONSE :\n%s\n", response);


                        while((num_read = fread(data_toq.some_text, 1, BUFSIZ, fclass))
> 0) {
                                data_toq.mesg_ident = dest;

                                rc_out = msgsnd(output, (void *)&data_toq, num_read, 0);
                        }

                        //strcat(response,"\r\n");
                        //send(new_fd, response, strlen(response), 0);



                        data_toq.mesg_ident = dest;
                        strcpy(data_toq.some_text, "46060008");

                        rc_out = msgsnd(output, (void *)&data_toq,
strlen(data_toq.some_text), 0);

                        break;

                default:
                        printf("Send ERROR PAGE\n");

                        strcpy(response,"<b>404 Not Found</b>");
                        strcat(response,"\r\n");

                        data_toq.mesg_ident = dest;
                        strcpy(data_toq.some_text, response);

                        rc_out = msgsnd(output, (void *)&data_toq, BUFSIZ, 0);
                        printf("HTTP : HEADER :\n%s", data_toq.some_text);

                        data_toq.mesg_ident = dest;
                        strcpy(data_toq.some_text,"2546-2550");
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                        rc_out = msgsnd(output, (void *)&data_toq, BUFSIZ, 0);
                        break;

        }////switch
        }//GET

        }
        }
}
```

index.html

```
<HTML>
    <HEAD>
        <TITLE> PC-Base Terminal Server Serial to Ethernet </TITLE>
    </HEAD>

    <BODY>
        <APPLET
            CODE     = "Remote_Access.class"
            WIDTH    = "900"
            HEIGHT   = "600" >
```

Remote_Access.java

```
import com.sun.org.apache.bcel.internal.classfile.Code;
import com.sun.org.omg.SendingContext.CodeBase;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.*;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;

public class Remote_Access extends JApplet implements KeyListener, Runnable {

    private int port;
    private String host;

    private int width, height;

    private int ne = 1, no = 0;

    private OutputStream out;
    private DataOutputStream dout;
    private InputStream in;
    private DataInputStream din;
    private Socket s;

    private JScrollPane sp;
    private JScrollBar sb;
    private JTextArea ar;

    Remote_Access teln;


    public void init() {

        teln = new Remote_Access();

        ar = new JTextArea(width, height);

        ar.setBackground(Color.lightGray);
        ar.setForeground(Color.BLACK);
        ar.setFont(new Font("Courier New", Font.PLAIN, 18));

        ar.setEditable(false);
        //ar.getDocument().addDocumentListener(this);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        sp = new JScrollPane(ar);

        getContentPane().add(sp, BorderLayout.CENTER);
        ar.addKeyListener(this);

        host = getParameter("HOST");
        port = Integer.parseInt(getParameter("PORT"));

        connect(host, port);

        width = getWidth();
        height = getHeight();


        Thread t = new Thread(this);
        t.setPriority(Thread.MIN_PRIORITY);
        t.start();      // call run()
    }

    public void connect(String host, int prt) {

        try {
            s = new Socket(host, port);

            out = s.getOutputStream();
            dout = new DataOutputStream(out);

            in = s.getInputStream();
            din = new DataInputStream(in);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void run() {         // Receive data from server
        String str, recv;
        int n, p;

        try{
            while(true) {
                str = "";
                recv = "";


                byte[] b = new byte[65536];

                n = in.read(b);
                if(n != -1) {
                    str = new String(b);

                    for(int i=0; i<str.length(); i++) {
                        char chr = str.charAt(i);

                        if(chr != '\0') {
                            recv = recv + chr;
                        } else {
                            break;
                        }
                    }

                    ne = 1;

                    display(recv);
                }

                Thread.sleep(100);
            }
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    public void keyPressed(KeyEvent e){
        String send;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        char key = e.getKeyChar();
        int keyCode = e.getKeyCode();

        if((keyCode != e.VK_CONTROL)&&
                (keyCode != e.VK_SHIFT)&&
                (keyCode != e.VK_CAPS_LOCK)&&
                (keyCode != e.VK_F1)&&
                (keyCode != e.VK_F2)&&
                (keyCode != e.VK_F3)&&
                (keyCode != e.VK_F4)&&
                (keyCode != e.VK_F5)&&
                (keyCode != e.VK_F6)&&
                (keyCode != e.VK_F7)&&
                (keyCode != e.VK_F8)&&
                (keyCode != e.VK_F9)&&
                (keyCode != e.VK_F10)&&
                (keyCode != e.VK_F11)&&
                (keyCode != e.VK_F12)&&
                (keyCode != e.VK_HOME)&&
                (keyCode != e.VK_END)&&
                (keyCode != e.VK_PRINTSCREEN)&&
                (keyCode != e.VK_SCROLL_LOCK)&&
                (keyCode != e.VK_PAUSE)&&
                (keyCode != e.VK_INSERT)&&
                (keyCode != e.VK_DELETE)&&
                (keyCode != e.VK_PAGE_UP)&&
                (keyCode != e.VK_PAGE_DOWN)&&
                (keyCode != e.VK_NUM_LOCK)&&
                (keyCode != e.VK_WINDOWS)&&
                (keyCode != e.VK_ALT)) {

            if(keyCode == e.VK_ENTER) {
                send = "\r\n";
            } else if(keyCode == e.VK_UP) {
                send = "\u001b[A";
            } else if(keyCode == e.VK_DOWN) {
                send = "\u001b[B";
            } else if(keyCode == e.VK_RIGHT) {
                send = "\u001b[C";
            } else if(keyCode == e.VK_LEFT) {
                send = "\u001b[D";
            } else {
                send = key + "";
            }

            byte b[] = send.getBytes();
            try {
                dout.write(b);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    }

    public void display(String recv) {
        String temp = "";
        int rob = 1;


        for(int i=0; i<recv.length(); i++) {
            temp = ar.getText();

            if(recv.charAt(i) == '\b') {          // Backspace

                if(ne == 0) {
                    if(temp.length() > 1) {
                        temp = temp.substring(0, temp.length()-1);
                    } else {
                        temp = temp;
                    }
                } else {
                    if(temp.length() > 1) {
                        temp = temp.substring(0, temp.length()-2);
                    } else {
                        temp = temp.substring(0, temp.length()-1);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    }

                    ne = 0;
                }

                ar.setText(temp);


            } else if(recv.codePointAt(i) == 7) {       // Serial Device return can't
Backspace
               rob = 0;
            } else {
                if(ne == 0) {
                    if(temp.length() > 1) {
                        temp = temp;
                    } else {
                        temp = temp;
                    }
                } else {
                    if(temp.length() > 1) {
                        temp = temp.substring(0, temp.length()-1);
                    } else {
                        temp = temp;
                    }

                    ne = 0;
                }

                ar.setText(temp);
                ar.append(recv.charAt(i) + "");

                /*if(recv.charAt(i) == '\n') {
                    updateScroll();
                }*/
            }


            updateScroll();

        }

        if(rob == 1) {
            putCursor();
        }
    }

    public void putCursor() {
        String temp;


        /*try {
            Thread.sleep(100);
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }*/

        ar.append("_");

        updateScroll();

        ne = 1;

    }

    public void updateScroll() {
        /*JScrollBar vbar = sp.getVerticalScrollBar();
        boolean autoScroll = ((vbar.getValue() + vbar.getVisibleAmount()) ==
vbar.getMaximum());

        if( autoScroll ) {
            ar.setCaretPosition( ar.getDocument().getLength() );
        }*/
         ar.setCaretPosition( ar.getDocument().getLength() );
        //ar.setCaretPosition(vbar.getMaximum());
    }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    public void keyReleased(KeyEvent evt) {}
    public void keyTyped(KeyEvent evt) {}

}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

นิรุธ อำนายศิลป์. 2548. **Network and Protocols Programming using C/C++.** กรุงเทพ: บริษัทดวงกมลสมัย จำกัด.

สันติ ศรีลาศักดิ์. 2542. **เจาะประเด็นงานเขียนโปรแกรมบนลินุกซ์.** กรุงเทพ: บริษัท ออฟเซ็ท เพรส จำกัด.

Elliotte R. Harold. 2004. **Java Network Programming 1.** Third Edition. New York: O'Reilly Publishing.

Kurt Wall, Mark Watson, and Mark Whitis etal. 1999. **Linux Programming Unleashed .** London: Sams Publishing.

Sean Walton. 2001. **Linux Socket Programming.** Chicago: Sams Publishing.

Warren W. Gay. 2000. **Linux Socket Programming by Example.** Manchester: Willame Publishing.

W. Richard Stevens, Bill Fenner, Andrew M. Rudoff. 2003. **UNIX® Network Programming Volume 1.** Third Edition. New York: Bill Publishing.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้