การพัฒนาระบบแคชสำหรับการเชื่อมต่ออินเทอร์เน็ตผ่านดาวเทียม

Cache System Development for Internet Connection via Satellite

โดย

นายพงศธร มณีวัฒนา

รหัส 43067032

อาจารย์ที่ปรึกษา

ดร. จันทร์บูรณ์ สถิตวิริยวงศ์

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2544

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การพัฒนาระบบแคชสำหรับการเชื่อมต่ออินเทอร์เน็ตผ่านดาวเทียม
นักศึกษา	นายพงศธร มณีวัฒนา
อาจารย์ที่ปรึกษา	ดร. จันทร์บูรณ์ สถิตวิริยวงศ์
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2544

บทคัดย่อ

การเชื่อมต่ออินเทอร์เน็ตผ่านดาวเทียมสามารถช่วยเพิ่มความเร็วในการดึงข้อมูลจากแหล่งเก็บข้อมูลบนอินเทอร์เน็ตได้เป็นอย่างมาก เนื่องจากเทคโนโลยีการสื่อสารผ่านดาวเทียมจะให้แบนด์วิธขนาดใหญ่ ส่งข้อมูลได้โดยตรง แต่มีข้อจำกัดในเรื่องของการหน่วงเวลาของช่องสัญญาณดาวเทียม จึงมีการพัฒนาระบบแคชขึ้นมาเพื่อช่วยลดข้อจำกัดดังกล่าว โครงการพัฒนาระบบงานนี้ได้ทำการพัฒนาระบบแคชขึ้นเพื่อใช้งานในสิ่งแวดล้อมที่มีการเชื่อมต่ออินเทอร์เน็ตผ่านเครือข่ายสองระบบ ได้แก่ เครือข่ายที่มีอยู่เดิมของทางสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังและเครือข่ายที่เพิ่มเข้ามาของทางเอกชนซึ่งใช้การเชื่อมต่อผ่านดาวเทียม ระบบแคชที่พัฒนาขึ้นเป็นลักษณะแคชที่ทำงานร่วมกันหลายตัว โดยที่แต่ละตัวจะใช้สำหรับเครือข่ายหนึ่ง แล้วทำการแลกเปลี่ยนข้อมูลระหว่างกัน ทำให้สามารถใช้งานเครือข่ายทั้งสองระบบได้ร่วมกัน เป็นการช่วยเพิ่มแบนด์วิธและเพิ่มประสิทธิภาพสำหรับการเชื่อมต่ออินเตอร์เน็ทผ่านดาวเทียมมากยิ่งขึ้น

Title	Cache System Development for Internet Connection via Satellite
Student	Mr. Pongsatorn Maneewatana
Advisor	Dr. Chanboon Sathitwiriyawong
Level of Study	Marter of Science in Information Technology
Major	Information Science
Academic Year	2001

Abstract

the Internet connection via satellites is a method to increase the transmission speed for receiving large amount of data from Internet. This is because it provides a larger bandwidth, receives data directly. However, there is a major problem associated with the method, which is the delay of the satellite channels. Hence, the web cache technology was proposed to overcome this problem. In this Project, cache system is developed for using in two network system evironment. That is a Campus network and the Internet connection via satellite network. The cache system, that is developed, is cooperative cache. Each cache for each networks and exchanges data between them. That make two networks can work together. It can provides a larger bandwidth and increase more effective of the Internet connection via satellite system.

กิตติกรรมประกาศ

โครงการพัฒนาระบบแคชสำหรับการเชื่อมต่ออินเทอร์เน็ตผ่านดาวเทียมที่ได้จัดทำขึ้นนี้สำเร็จลุล่วงไปได้ด้วยดี โดยได้รับการสนับสนุนจากบุคคลหลายฝ่าย ข้าพเจ้าจึงขอขอบพระคุณบุคคลดังต่อไปนี้

- บิดามารดา ที่ให้ความช่วยเหลือ ดูแลและให้กำลังใจ
- อาจารย์จันทร์บูรณ์ สถิตวิริยวงศ์ อาจารย์ที่ปรึกษาโครงการ ที่ให้คำปรึกษา แนะนำ และชี้แนะแนวทางในการพัฒนาระบบเป็นอย่างดี
- อาจารย์โอฬาร วงศ์วิรัตน์ อาจารย์ที่ปรึกษาโครงการร่วม ที่ให้คำปรึกษา ชี้แนะแนวทาง และข้อบกพร่อง ร่วมถึงแนวทางแก้ไข และอำนวยความสะดวกเป็นอย่างดี
- MVLab ที่เอื้อเฟื้อสถานที่ในการพัฒนาระบบ และให้การสนับสนุนโครงการ
- คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง และคณาจารย์ทุกท่าน ที่ให้ความรู้ทางวิชาการ ทำให้สามารถใช้ความรู้ในการพัฒนาและแก้ไขระบบได้
- เพื่อนร่วมรุ่น IS9(ภาคปกติ) และ IS9(ภาคสมทบ) ที่ให้คำแนะนำในการพัฒนาระบบและให้กำลังใจตลอดมา

พงศธร มณีวัฒนา

ผู้จัดทำ

สารบัญ

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญภาพ

หน้า

รูปที่

บทที่ 1
บทนำ

1.1 ความเป็นมา

จากความนิยมที่เพิ่มมากขึ้นอย่างรวดเร็วในการใช้งาน World Wide Web ในปัจจุบัน ทำให้ปริมาณข้อมูลที่มีอยู่บนเครือข่ายมีเพิ่มสูงมากขึ้นเป็นทวีคูณ ส่งผลให้เกิดปัญหามากมาย ไม่ว่าจะเป็นปัญหาความคับคั่งของข้อมูลเกิดเป็นคอขวด (bottleneck) ขึ้น ทำให้ผลตอบสนองช้าลง จนบางครั้งไม่สามารถติดต่อเว็บไซต์ (website) นั้นได้เลย อีกทั้งยังมีปัญหาในการติดต่อไปยังเครือข่ายที่ช้าเอง ทำให้ต้องรอข้อมูลที่ได้รับเป็นเวลานาน จึงได้มีแนวคิดว่า น่าจะมีการนำข้อมูลที่ใช้บ่อยๆเก็บไว้ใกล้กับผู้ร้องขอข้อมูลให้มากที่สุด ด้วยเหตุผลที่ว่า ถ้าต้องการใช้งานข้อมูลนั้นซ้ำ ก็สามารถดึงข้อมูลจากแหล่งเก็บข้อมูลนั้น ซึ่งอยู่ใกล้ผู้ร้องขอมากกว่าแหล่งเก็บข้อมูลโดยตรง มาให้บริการผู้ร้องขอได้เลย ลักษณะนี้มีประโยชน์หลายประการ ที่เห็นได้ชัดคือ จะทำให้ได้รับผลตอบสนองที่เร็วขึ้น สามารถลดปริมาณการรับส่งข้อมูลออกไปยังภายนอกเครือข่าย ผลก็คือ ทำให้แบนด์วิธเพิ่มมากขึ้น และในกรณีที่เครือข่ายที่เราทำการติดต่อมีการทำงานที่ล่าช้า ทำให้เราไม่จำเป็นต้องติดต่อไปยังเครือข่ายนั้นบ่อยๆ แนวคิดดังกล่าวได้ถูกพัฒนาขึ้นมาจนเป็นระบบที่เรียกว่า แคช (Cache System)

ความต้องการความเร็วการเชื่อมต่ออินเทอร์เน็ตที่เพิ่มมากขึ้นเรื่อยๆ จึงมีการพัฒนาเทคโนโลยีมากมายเพื่อตอบสนองความต้องการดังกล่าว เทคโนโลยีการเชื่อมต่ออินเทอร์เน็ตผ่านดาวเทียมเป็นแนวทางหนึ่งที่ช่วยเพิ่มความเร็วในการรับส่งข้อมูลได้เป็นอย่างดี เนื่องจากการส่งข้อมูลผ่านทางดาวเทียมจะสามารถส่งข้อมูลได้โดยตรง โดยไม่ผ่านเครือข่ายภาคพื้นดิน อีกทั้งยังให้แบนด์วิธขนาดใหญ่ และไม่ต้องให้งานร่วมกับผู้อื่น ทำให้สามารถได้รับข้อมูลปริมาณที่มาก เมื่อเทียบกับระบบที่ใช้โมเด็ม (modem) เชื่อมต่อผ่านสายโทรศัพท์ตามบ้านทั่วไป แต่การสื่อสารผ่านดาวเทียมมีข้อจำกัดในเรื่องของ การหน่วงเวลาของช่องสัญญาณ เนื่องจากดาวเทียมอยู่ห่างจากโลกเป็นระยะทางที่ไกลมาก ฉะนั้นการส่งสัญญาณขาขึ้นและขาลงจึงต้องเสียเวลาช่วงหนึ่ง อีกทั้งเมื่อฝนตกยังทำให้สัญญาณดาวเทียมถูกรบกวนด้วย การนำระบบแคชเข้ามาประยุกต์ใช้สามารถช่วยลดผลกระทบดังกล่าวได้เป็นอย่างดี เนื่องจาก ข้อมูลที่ถูกใช้บ่อยๆสามารถถูกดึงจากในแคชโดยไม่ต้องติดต่อไปยังแหล่งเก็บข้อมูลบนอินเทอร์เน็ตโดยตรง

โครงการพัฒนาระบบงานนี้จัดทำขึ้นเนื่องจากว่า ในห้องปฏิบัติการมัลติมีเดีย (Multimedia Lab) ซึ่งแต่เดิมมีระบบเครือข่ายภายใน (LAN) ที่เชื่อมต่อกับทางเครือข่ายของสถาบันเทคโนโลยี

พระจอมเกล้าเจ้าคุณทหารลาดกระบังอยู่ ได้เพิ่มการเชื่อมต่อกับเครือข่ายของเอกชน ซึ่งเป็นลักษณะการเชื่อมต่ออินเทอร์เน็ตผ่านดาวเทียม เนื่องจากเราสามารถได้รับประโยชน์หลายประการจากการเชื่อมต่อนี้ และเพื่อเป็นการศึกษาถึงความเป็นไปได้ในการประยุกต์ใช้ระบบอินเทอร์เน็ตผ่านดาวเทียมไปใช้กับงานทางด้านมัลติมีเดียผ่านทางอินเทอร์เน็ต เพราะว่างานทางมัลติมีเดียมีการส่งข้อมูลทั้งภาพ เสียง และต้องการการรับส่งข้อมูลในปริมาณที่มาก จึงต้องการระบบเครือข่ายที่มีความเร็วสูง ซึ่งการใช้เครือข่ายผ่านดาวเทียมเข้ามาประยุกต์ใช้เป็นเหมือนทางเลือกหนึ่งที่น่าสนใจ รวมทั้งราคาก็ไม่สูงมากด้วย ซึ่งเราจะกล่าวถึงรายละเอียดในระบบนี้ในบทต่อไป

จะเห็นว่าในขณะนี้ในห้องปฏิบัติการมีเส้นทางที่ทำการเชื่อมต่ออินเทอร์เน็ตได้สองเส้นทาง เราสามารถเลือกการเชื่อมต่อได้เส้นทางใดเส้นทางหนึ่ง การพัฒนาระบบงานนี้จึงได้มีการพัฒนาระบบขึ้นเพื่อให้สามารถใช้งานอินเทอร์เน็ตโดยผ่านเส้นทางทั้งสองได้พร้อมกัน โดยผ่านระบบแคชที่สามารถทำงานร่วมกัน เพื่อเป็นทางผ่านของข้อมูลเป็นเสมือนเว็บเกตเวย์ (Web Gateway) และแลกเปลี่ยนข้อมูลระหว่างกัน ซึ่งเป็นการเพิ่มแบนด์วิธในการรับส่งข้อมูลอีกด้วย ทำให้การใช้งานอินเทอร์เน็ตมีประสิทธิภาพมากขึ้น

1.2 วัตถุประสงค์ในการพัฒนาระบบงาน

- เพื่อลดผลกระทบจากปัญหาการหน่วงเวลาของสัญญาณดาวเทียมในการรับข้อมูล โดยนำระบบแคชเข้ามาประยุกต์ใช้ เพื่อให้การใช้การเชื่อมต่ออินเทอร์เน็ตผ่านดาวเทียมเป็นไปอย่างมีประสิทธิภาพ
- ทำให้ระบบอินเทอร์เน็ตผ่านดาวเทียม สามารถใช้งานร่วมกับระบบเครือข่ายที่มีอยู่เดิมได้ โดยพัฒนาระบบแคชที่ทำงานร่วมกันเข้ามา เพื่อสร้างเส้นทางการเชื่อมต่อระหว่างระบบเก่ากับระบบใหม่ ทำให้ใช้ประโยชน์จากทรัพยากรที่มีอยู่ให้เกิดประโยชน์มากที่สุด
- เพื่อเป็นแนวทางสำหรับผู้ที่จะพัฒนาระบบแคชให้สามารถมองเห็นภาพรวม และสามารถนำไปประยุกต์ใช้กับงานอื่นๆได้ต่อไป

1.3 ขอบเขตการพัฒนาระบบงาน

โครงการพัฒนาระบบงานแคชสำหรับการเชื่อมต่ออินเทอร์เน็ตผ่านดาวเทียม จะช่วยให้สามารถใช้งานอินเทอร์เน็ตผ่านเส้นทางการเชื่อมต่อสองเส้นทางได้พร้อมกัน โดยเน้นไปที่การเชื่อมต่อระหว่างแคช เพื่อเป็นทางผ่านการร้องขอและแลกเปลี่ยนข้อมูลระหว่างกัน ระบบแคชดังกล่าวสามารถใช้งานได้ในระบบโพรโตคอล HTTP/1.0 ซึ่งมีวิธีการ (Method) พื้นฐานคือ GET, HEAD และ POST รวมทั้งเฮดเดอร์ (header) ที่ใช้ในเวอร์ชั่น HTTP/1.0 และ HTTP/1.1 ที่สำคัญๆ

บางส่วน เนื่องจากมีรายละเอียดมากมายสำหรับเฮดเดอร์ที่เพิ่มความสามารถให้กับโพรโตคอล HTTP และระบบแคชเพิ่มเข้ามามากมาย โครงการพัฒนานี้จึงเป็นเพียงการพัฒนาเค้าโครงหลักเพื่อให้สามารถใช้งานได้ตามเป้าหมายที่กำหนดไว้ การพัฒนาระบบให้สมบูรณ์จึงจำเป็นต้องใช้เวลาและต้องได้รับการพัฒนาต่อเนื่องไป

ในการพัฒนาโปรแกรมแบ่งเป็น 4 ส่วนหลัก คือ ส่วนที่เป็นระบบแคช, ส่วนที่ใช้ในการติดต่อระหว่างแคชด้วยกัน, ส่วนที่ติดต่อกับผู้ใช้งาน (User Interface) และส่วนที่ใช้ในการติดตามผลของข้อมูลเพื่อวัดประสิทธิภาพที่เกิดขึ้น

1.4 ขั้นตอนการศึกษาและพัฒนาระบบ

ในการพัฒนาโครงงานได้มีการศึกษาและพัฒนามาเป็นลำดับดังนี้

1.4.1 ศึกษากลไกการทำงานของระบบแคช (Cache) ในหลายๆรูปแบบ และระบบแคชที่มีใช้อยู่จริงในปัจจุบัน รวมทั้งโครงสร้างการจัดเก็บและจัดการกับแคชที่มีอยู่ด้วย

1.4.2 ศึกษาระบบอินเทอร์เน็ตผ่านดาวเทียมที่นำมาใช้กับเครือข่ายของเรา เพื่อให้ทราบถึงจุดเด่น ข้อจำกัด ลักษณะการใช้งาน เพื่อเป็นข้อมูลในการหารูปแบบของแคชที่เหมาะสมที่จะนำมาประยุกต์ใช้

1.4.3 ศึกษาระบบแคชแบบรวมศูนย์และแคชแบบกระจาย เพื่อเป็นแนวทางในการพัฒนาระบบแคชที่ทำงานร่วมกัน เพื่อที่จะรองรับความต้องการการใช้งานอินเทอร์เน็ตที่เพิ่มมากขึ้นอย่างรวดเร็ว

1.4.4 ศึกษาเครื่องมือ (Tool) ที่ใช้ เพื่อให้สามารถเลือกใช้เครื่องมือและพัฒนาโครงงานเป็นไปด้วยความรวดเร็วเมื่อมีการพัฒนาจริง

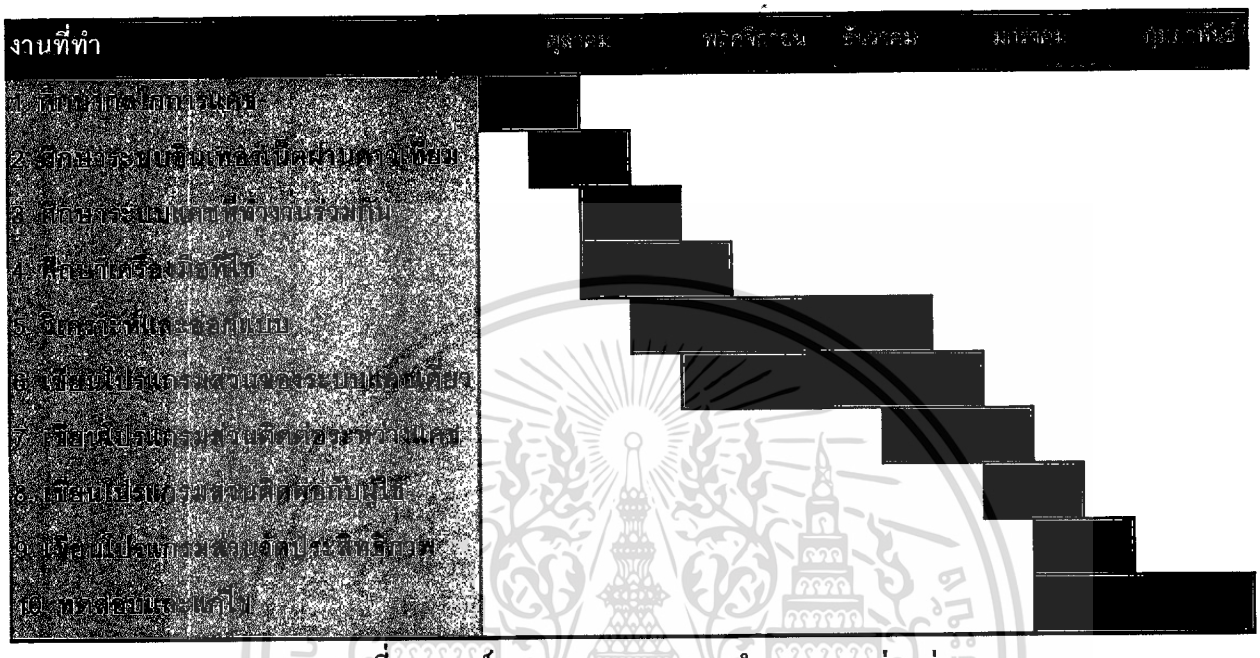
1.4.5 วิเคราะห์และออกแบบระบบแคชโดยใช้ข้อมูลจากที่ได้ศึกษามา เขียนเป็นลักษณะโฟลชาร์ต (Flow Chart) การทำงาน เพื่อแยกงานออกเป็นส่วนๆย่อย และพัฒนาในแต่ละส่วนทีละขั้นตอน

1.4.6 ทำการโปรแกรมในส่วนของระบบแคชเดียวขึ้นมาใช้งาน พร้อมทั้งทำการทดสอบและแก้ไข เพื่อให้ระบบแคชสามารถใช้งานได้จริงตามเป้าหมายที่กำหนด

1.4.7 ทำการโปรพแกรมระบบแคชที่ทำงานร่วมกัน (Co-orperative Cache) ในส่วนที่ใช้ติดต่อระหว่างแคช รวมถึงส่วนที่ใช้ติดต่อกับผู้ใช้ (User Interface)

1.4.8 ทำการโปรแกรมส่วนการติดตามผลของการประมวลผลและวัดประสิทธิภาพออกมาในเชิงตัวเลข

1.4.9 ทดสอบและแก้ไขข้อผิดพลาดที่เกิดขึ้นในการใช้งานกับสิ่งแวดล้อมจริง พร้อมทั้งติดตามผล



ตารางที่ 1.1: ชาร์ตแสดงระยะเวลาการทำงานของแต่ละช่วง

1.5 เนื้อหาในแต่ละบท

ภายใต้เอกสารประกอบการพัฒนาระบบนี้ ประกอบไปด้วยเนื้อหาหลักๆ 5 ส่วนด้วยกัน คือ

1.5.1 บทที่ 1 ซึ่งคือบทที่กำลังกล่าวถึงนี้ ซึ่งประกอบไปด้วยความเป็นมาในการพัฒนาโครงงานนี้ วัตถุประสงค์ ขอบเขตในการพัฒนา และขั้นตอนที่ได้ดำเนินงานในการพัฒนา

1.5.2 บทที่ 2 กล่าวถึงเนื้อหาและทฤษฎีที่เกี่ยวข้องทั้งหมด ทั้งระบบแคชและโพรโตคอลที่เกี่ยวข้อง ระบบอินเทอร์เน็ตผ่านดาวเทียม เพื่อเลือกระบบที่เหมาะสมกับระบบของเรามากที่สุด

1.5.3 บทที่ 3 กล่าวถึงเนื้อหาการออกแบบระบบแคช ให้ตรงตามความต้องการ โดยมีการกล่าวถึงรายละเอียดทั้งหมด ไม่ว่าจะเป็นโฟลชาร์ต (Flow Chart) แสดงการทำงาน การจัดเก็บข้อมูล การจัดการข้อมูล เทคนิคการติดต่อระหว่างแคช รวมทั้งกล่าวถึงรายละเอียดภายใต้สิ่งแวดล้อมที่ได้ทำการพัฒนาระบบด้วย

1.5.4 บทที่ 4 เป็นขั้นตอนการพัฒนาระบบงานตามที่ได้ออกแบบไว้ รวมถึงเครื่องมือที่ใช้และเทคนิคที่ใช้ในการเขียนโปรแกรม

1.5.5 บทที่ 5 เป็นการสรุปผลจากที่ได้พัฒนาระบบงานมา และทดลองใช้งานกับสภาพแวดล้อมจริง

1.5.6 บทที่ 6 บทสรุปของการพัฒนาระบบงาน ประโยชน์ที่ได้รับ อุปสรรค์ของการพัฒนา และข้อเสนอแนะเพื่อจะสามารถพัฒนาระบบได้ต่อไป

บทที่ 2

ระบบอินเทอร์เน็ตผ่านดาวเทียมและระบบแคช

2.1 ระบบการสื่อสารอินเทอร์เน็ตผ่านดาวเทียม (Internet Connection via Satellite)

การสื่อสารผ่านดาวเทียมได้เข้ามามีบทบาทสำคัญต่อการสื่อสารโทรคมนาคมตั้งแต่นักวิทยาศาสตร์ประสบความสำเร็จในการส่งดาวเทียมเข้าสู่วงโคจรเป็นผลสำเร็จตั้งแต่ปี ค.ศ. 1958 ได้มีการใช้งานจากการสื่อสารผ่านดาวเทียมอย่างมากมาย

ในปัจจุบันได้มีการนำช่องสัญญาณดาวเทียมในย่านความถี่ Ku-Band 14/12 GHz คือมีความถี่ในการส่งสัญญาณขึ้นไปยังดาวเทียม (Uplink) 14-14.5 GHz และความถี่ในการส่งสัญญาณลงมาจากดาวเทียม (Downlink) 11.7-12.2 GHz ใช้ในการเชื่อมต่ออินเทอร์เน็ต เนื่องจากว่าการให้บริการสื่อสารผ่านดาวเทียมในย่าน Ku-Band นั้นจะให้บริการในรูปแบบของสัญญาณดิจิตอล หมายความว่า ช่องสัญญาณที่แบ่งย่อยจากช่องสัญญาณดาวเทียมนั้นมีการทำงานแบบดิจิตอล ซึ่งต่างจากย่านความถี่ C-Band 6/4 GHz ที่ทำงานแบบอนาลอก ทำให้ย่านความถี่ Ku-Band สามารถเชื่อมต่อกับข้อมูลในระบบคอมพิวเตอร์ได้โดยตรง

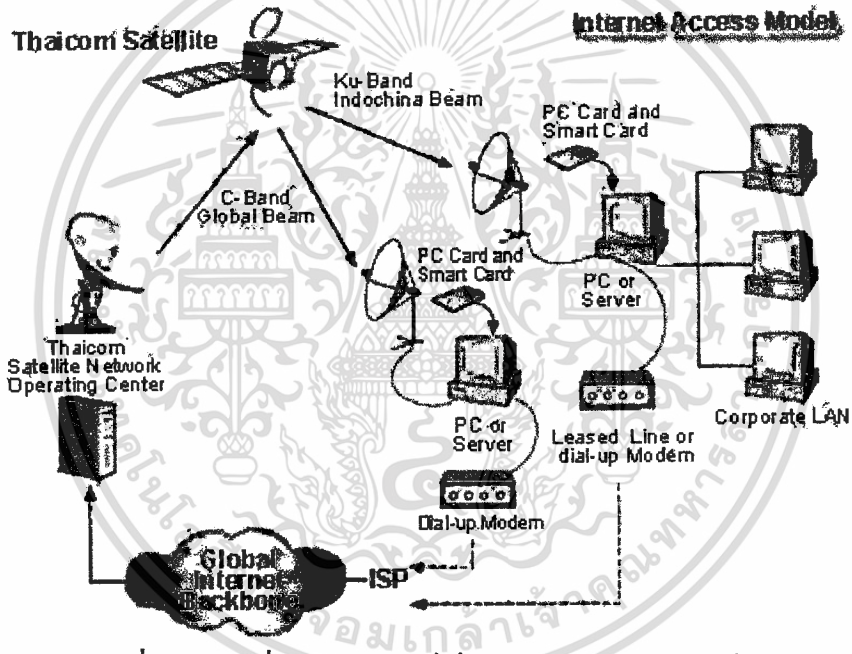
ดาวเทียมไทยคมซึ่งเป็นดาวเทียมสื่อสารดวงแรกของไทย ให้บริการหลักในการใช้วงจรดาวเทียมหรือทรานสพอนเดอร์ (Transponder) ทำหน้าที่ในการรับสัญญาณจากสถานีภาคพื้นดินผ่านจานสายอากาศและนำสัญญาณนั้นมาแปลงความถี่ให้ต่ำลงและขยายสัญญาณให้มีกำลังแรงขึ้นเพื่อส่งสัญญาณกลับมาสู่สถานีสัญญาณปลายทางบนพื้นโลก ณ ตำแหน่งใดๆ ภายใต้พื้นที่ครอบคลุม(Footprint)ของดาวเทียมไทยคม เพื่อให้ผู้ใช้บริการสามารถเชื่อมต่อการสื่อสารในระยะทางที่ห่างไกลหรือสำหรับการสื่อสารแบบกระจายสู่ผู้รับจำนวนมาก (Broadcasting)

ดาวเทียมไทยคมได้เปิดบริการเสริมในหลายรูปแบบ ซึ่งการให้บริการอินเทอร์เน็ทผ่านดาวเทียมก็เป็นบริการเสริมรูปแบบหนึ่งด้วย โดยมีการให้บริการสำหรับผู้ใช้ปลายทางใน 3 ลักษณะ คือ NetTurbo, Netcast, Thaicom Direct จากรูปแบบการให้บริการดังกล่าว เราเลือกใช้รูปแบบการเชื่อมต่อผ่านดาวเทียมทิศทางเดียวในด้าน downstream ที่เรียกว่า NetTurbo ซึ่งจะกล่าวถึงในรายละเอียดต่อไป

2.2 NetTurbo

เป็นการให้บริการอินเทอร์เน็ทความเร็วสูงแบบทิศทางเดียว ซึ่งทำให้ผู้ใช้ (ที่เป็นองค์กรหรือบุคคล) ดึงข้อมูลจากอินเทอร์เน็ตด้วยความเร็วสูงถึง 400 Kbps และสูงถึง 4 Mbps สำหรับข้อมูลที่อยู่ในแคช

การให้บริการนี้จะเชื่อมต่อข้อมูลออกด้วยโมเด็มผ่าน ISPs และดึงข้อมูลเข้ามาผ่านช่องสัญญาณดาวเทียมไทยคม ทั้งนี้ ISPs สามารถให้บริการนี้ได้โดยตรง โดยติดตั้งจานรับสัญญาณดาวเทียมและเพิ่ม PC Card ให้กับเครื่องคอมพิวเตอร์เพื่อใช้ในการรับสัญญาณอินเทอร์เน็ทผ่านดาวเทียม และผู้ใช้จะส่งข้อมูลโดยอาศัยการเชื่อมต่อโมเด็มผ่าน ISPs ดังแสดงในรูปที่ 2.1



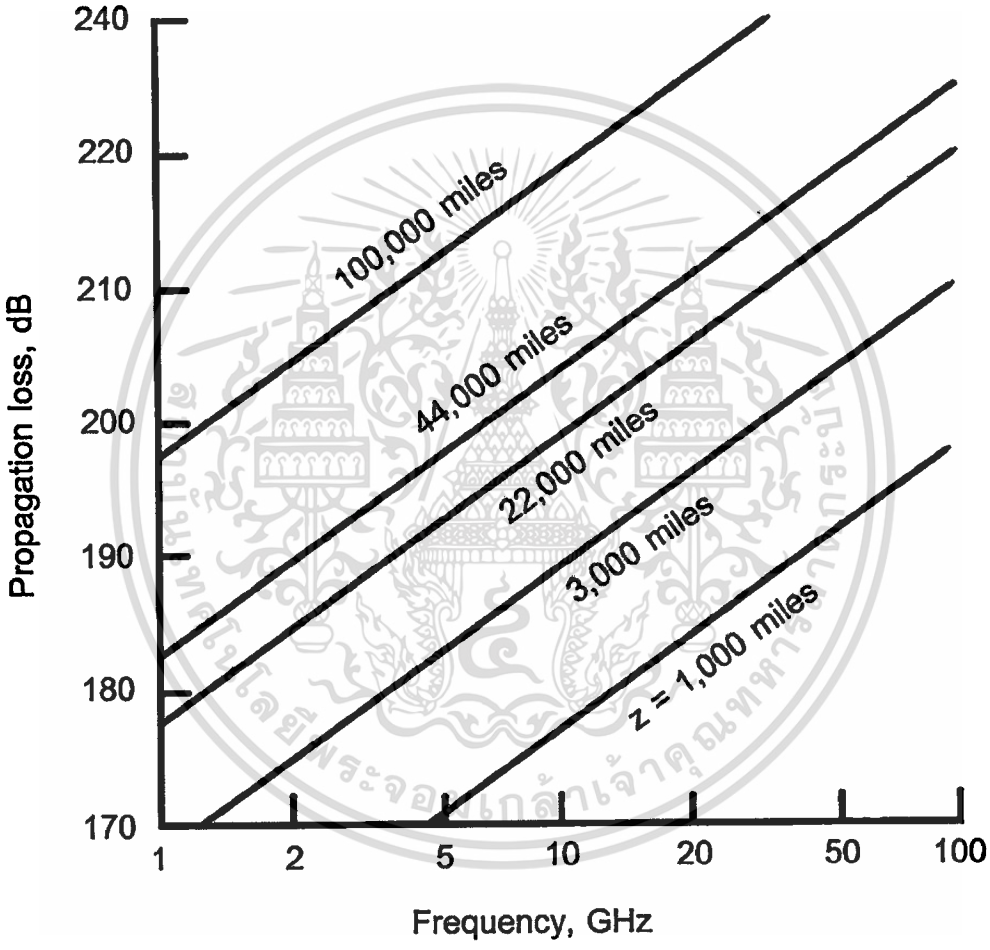
รูปที่ 2.1: การเชื่อมต่ออินเทอร์เน็ทผ่านดาวเทียมทิศทางเดียว

การให้บริการลักษณะนี้มีข้อดีคือ

- มีแบนด์วิธกว้าง ส่งข้อมูลได้เป็นจำนวนมาก
- สามารถรับข้อมูลได้โดยตรงผ่านจานรับสัญญาณดาวเทียม
- ช่วยลดการจราจรภาคพื้นดิน
- สามารถเพิ่มหรือลดขนาดของแบนด์วิธ (bandwidth) ได้
- ไม่ต้องใช้แบนด์วิธร่วมกับผู้ใช้คนอื่น
- อุปกรณ์มีราคาถูก ติดตั้งง่าย

แต่มีข้อเสียหลายประการในการใช้อินเทอร์เน็ทผ่านดาวเทียม ซึ่งเป็นผลมาจากปัญหาของระบบสื่อสารผ่านดาวเทียมเอง ดังนี้

- *Propagation Loss* เป็นการสูญเสียกำลังงานจากระยะทางที่ไกลมาก ในการส่งสัญญาณจากสถานีรับส่งบนพื้นโลกกับสถานีบนดาวเทียม จากกราฟรูปที่ 2.2 คลื่นความถี่ย่าน Ku-band มีอัตราการลดทอนสัญญาณที่มากกว่าคลื่นความถี่ย่าน C-band ซึ่งมีความถี่ต่ำกว่า

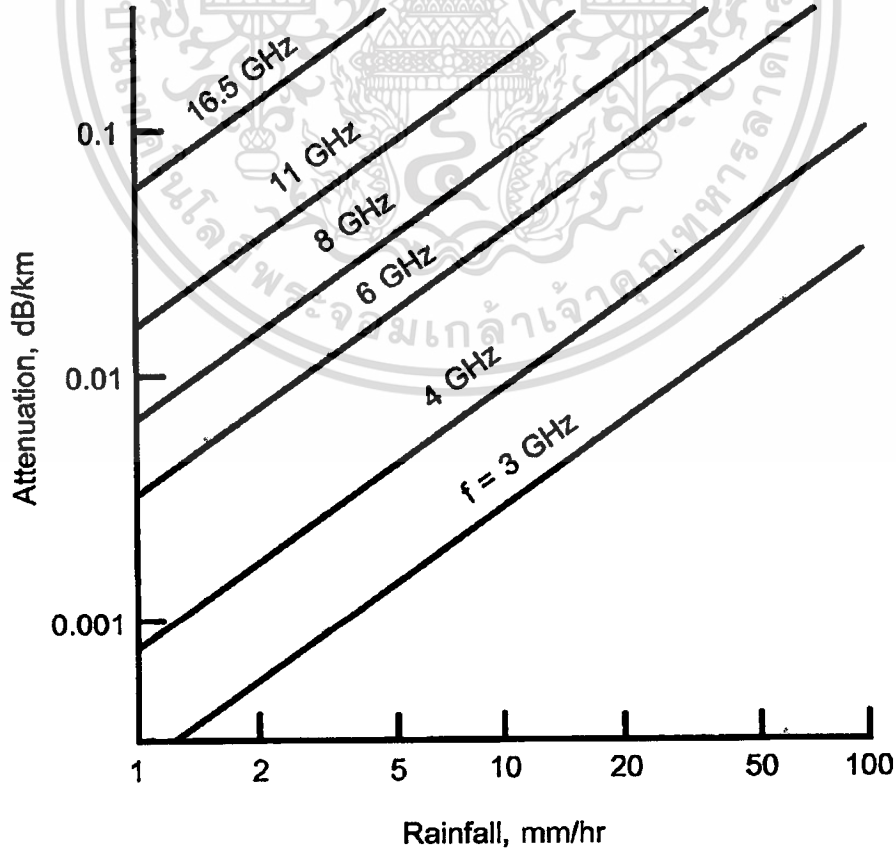


รูปที่ 2.2: Satellite propogation loss

ความถี่ย่าน Ku-band (14/12 GHz) มีอัตราลดทอนกำลังจากระยะทางที่คลื่นเดินทางในค่าต่างๆ ซึ่งอัตรลดทอนนี้จะมีค่ามากขึ้นตามระยะทางที่คลื่นเดินทาง โดยดาวเทียมที่โคจรในวงโคจรค้างฟ้า อยู่ห่างจากโลกตรงเส้นศูนย์สูตรระยะทางประมาณ 35,800-41,756 กิโลเมตร ซึ่งมีผลให้การเดินทางของสัญญาณขาขึ้นและลงใช้เวลาอยู่ระหว่าง 239.6-279.0 ms. นอกจากระยะทางที่

ห่างไกลจะทำให้สูญเสียกำลังงานในการส่งสัญญาณแล้ว ยังทำให้ใช้เวลาในการตอบสนองช้ากว่าระบบสื่อสารภาคพื้นดินอีกด้วย

- *Atmosphere Loss* เป็นการสูญเสียที่เกิดสัญญาณดาวเทียมเดินทางผ่านชั้นบรรยากาศที่ปกคลุมโลก ซึ่งประกอบไปด้วย ก๊าซต่างๆ อะตอม ไอน้ำ มลภาวะ และสิ่งเจือปนอื่นๆที่อยู่ในชั้นบรรยากาศโลกที่ปกคลุมในรัศมีประมาณ 400 ไมล์ห่างจากพื้นโลก เมื่อคลื่นความถี่ย่าน Ku-band (14/12 GHz) เดินทางผ่านชั้นบรรยากาศโลก สัญญาณจะปะทะเข้ากับสิ่งต่างๆ ที่อยู่ในชั้นบรรยากาศ โดยเฉพาะอย่างยิ่งในชั้นไอโอโนสเฟียร์ (ionosphere) ส่งผลให้สัญญาณที่มีความถี่สูงเกิดการดูดกลืน สะท้อน หักเห ทำให้สัญญาณเดินทางไม่สะดวก ดังนั้นกำลังงานของสัญญาณจึงลดลงได้ ซึ่งถือว่าเป็นอุปสรรค์อีกอย่างหนึ่งในการสื่อสารผ่านดาวเทียม
- *Rainfall Effect* เป็นปัญหาสำคัญในการใช้ความถี่ของสัญญาณที่สูงในย่าน Ku-Band (14/12 GHz) ซึ่งมีความยาวคลื่นสั้นมาก จนทำให้เมื่อเกิดฝนตกในชั้นบรรยากาศ เม็ดฝนที่หนาแน่นสามารถที่จะดูดซับคลื่นความถี่ในย่านนี้ได้ ทำให้การสื่อสารถูกตัดขาดได้ เป็นการเกิด Loss ในกรณีที่มีฝนตกในชั้นบรรยากาศ



รูปที่ 2.3: Attenuation per length vs rainfall rate

2.3 การแคช (Cache)

การแคชเป็นเทคนิคที่มีจุดมุ่งหมายในการนำบางส่วนของข้อมูลทั้งหมดเก็บไว้ใกล้ส่วนที่ใช้ประมวลผล อย่างเช่น ในการเก็บข้อมูลแบบลำดับชั้น (memory hierarchies) ซึ่งเป็นแคชขนาดเล็กบนหน่วยประมวลผล จะใช้เวลาเข้าถึงข้อมูลได้เร็วกว่าวงจรหน่วยความจำ (memory-chip) ด้วยหลักการเดียวกัน มีการประยุกต์ใช้กับหน่วยที่ใหญ่กว่า เช่น ในระบบไฟล์ โดยเอกสารที่ถูกใช้บ่อยจะเก็บไว้ในหน่วยความจำหลักแทนการเก็บในดิสก์ซึ่งช้ากว่า ตามหลักการแล้ว การเก็บเอกสารที่ต้องการใช้บ่อยๆใกล้กับบริเวณที่ต้องการเอกสารนั้น จะช่วยลดเวลาในการติดต่อและเพิ่มแบนด์วิธได้

การแคชบนเว็บมีหลักการทำงานเหมือนกัน โดยจะแนะนำให้เก็บเอกสารบนเว็บที่เข้าถึงไว้ใกล้กับผู้ใช้ในหน่วยความจำหรือในดิสก์ เพื่อช่วยลดเวลาในการติดต่อไปที่เครื่องให้บริการเว็บต้นกำเนิด (origin web server) ในครั้งถัดไปที่มีการร้องขอ ตามธรรมชาติของเว็บจะใช้แคชร่วมกันจากคนหลายคนซึ่งมีความสนใจเหมือนกัน จากการสังเกตนี้ รวมถึงความต้องการที่จะควบคุมการเข้าถึงข้อมูลอันมหาศาลบนอินเทอร์เน็ตผ่านไฟล์วอลล์ (firewall) ทำให้มีการพัฒนาแคชชิ่งพร็อกซี่ (caching proxy) ขึ้น โดยทั่วไปพร็อกซี่เป็นโปรแกรมที่ทำหน้าควบคุมการเชื่อมต่อของอินทราเน็ทกับอินเทอร์เน็ตเพื่อเหตุผลทางด้านความปลอดภัย และการทำแคชชิ่งเป็นหลัก

มีเหตุผลหลายประการที่นำการแคชมาใช้ดังนี้

- ลดความล่าช้าในการติดต่อ (reduce latency) เนื่องจาก เมื่อผู้ขอใช้บริการร้องขอข้อมูล จะทำการเรียกข้อมูลจากหน่วยความจำหรือดิสก์ซึ่งอยู่ใกล้ผู้ร้องขอมากกว่า แทนที่จะเรียกจากแหล่งเก็บข้อมูลบนอินเทอร์เน็ตโดยตรง ทำให้สามารถตอบสนองความต้องการผู้ร้องขอได้รวดเร็วกว่า
- ลดปริมาณของข้อมูล(reduce traffic) เนื่องจาก การที่ไม่ต้องติดต่อกับแหล่งเก็บข้อมูลภายนอกเครือข่าย(internet)โดยตรง ทำให้ช่วยรักษาแบนด์วิธไม่ให้ถูกใช้โดยไม่จำเป็นและสามารถควบคุมได้ง่ายกว่าอีกด้วย

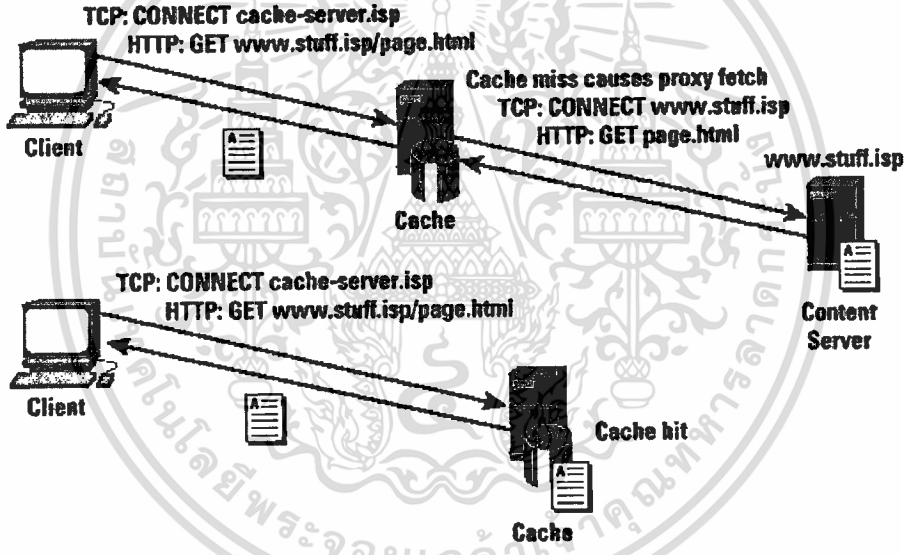
การใช้แคชอาจทำให้เกิดข้อเสียสำคัญ คือ ความเสี่ยงที่จะได้รับข้อมูลที่ล้าสมัย (stale) เนื่องจากข้อมูลที่ได้อาจไม่ได้รับการปรับปรุง (update) จากแหล่งข้อมูลจริงในอินเทอร์เน็ต แต่อย่างไรก็ตาม HTTP โพรโตคอลมีกลไกที่ชัดเจนในการแก้ปัญหาดังกล่าวนี้ ซึ่งจะกล่าวถึงในภายหลัง

2.3.1 เว็บแคช (Web Cache)

แบ่งออกเป็น 2 ชนิด

1. บราวเซอร์แคช (browser caches) ในบราวเซอร์สมัยใหม่ เช่น Internet Explorer หรือ Netscape จะมีการติดตั้งแคชทางด้านคอมพิวเตอร์ของผู้ใช้บนฮาร์ดดิสก์เพื่อเก็บข้อมูลหรือเอกสารไว้ การแคชทางด้านบราวเซอร์จะมีทำงานสอดคล้องกับกฎการแคชทั่วไป มีการตรวจสอบความใหม่ของข้อมูล การแคชนี้ใช้ประโยชน์เมื่อผู้ใช้กดปุ่มย้อนกลับ (back botton) เพื่อกลับไปยังหน้าที่เคยไปแล้ว

2. พร็อกซี่แคช (proxy caches) ทำงานในหลักการเดียวกัน เพียงแต่มีสัดส่วนที่ใหญ่กว่า และทำงานทางด้านเครื่องผู้ให้บริการ พร็อกซี่จะให้บริการกับผู้ใช้จำนวนมากในแนวทางเดียวกัน โดยเอกสารถูกร้องไปยังเครื่องให้บริการจริงในอินเทอร์เน็ทเพียงครั้งเดียว จากนั้นเมื่อมีผู้ใช้รายอื่นร้องขอเอกสารเดิม พร็อกซี่จะเป็นผู้ให้บริการแทน ดังจะเห็นได้จากรูปที่ 2.4



รูปที่ 2.4: การทำงานของพร็อกซี่แคช

2.4 แคชรวมศูนย์ (Centralized Cache) และแคชแบบกระจาย (Distributed Cache)

รูปแบบของแคชแรกเริ่มจะเป็นแคชแบบรวมศูนย์ (centralized cache) ซึ่งจะมีแคชเป็นศูนย์กลางตัวเดียวใช้ทรัพยากรร่วมกัน แต่เมื่อมีจำนวนผู้ใช้เพิ่มมากขึ้น จะเกิดความคับคั่งของข้อมูลขึ้น เนื่องจากการร้องขอเนื้อหาของเว็บ (web object) จะต้องผ่านแคชศูนย์กลางเพียงตัวเดียว จึงได้มีรูปแบบแคชแบบกระจาย (distributed cache) ขึ้น โดยจะมีเครื่องให้บริการแคช (cache server) หลายๆ ตัวทำงานร่วมกัน เป็นการช่วยกระจายการร้องขอออกไปยังเครื่องให้บริการแคชหลายๆตัว เพื่อทำให้การแคชมีประสิทธิภาพมากขึ้น

2.4.1 แคชรวมศูนย์

สร้างในเครื่องจักรที่มีหน่วยความจำ และฮาร์ดดิสก์ขนาดใหญ่ ดูเหมือนการใช้แคชรวมศูนย์ขนาดใหญ่ จะมีข้อดีเหนือกว่าแคชแบบกระจาย อีกทั้งแนวโน้มของราคาหน่วยความจำและฮาร์ดดิสก์จะมีราคาถูกลง ข้อดีหลักของแคชรวมศูนย์ คือ การดำเนินการของซอฟแวร์ง่าย มีอยู่ทั่วไป และการบริหารงานยังง่ายอีกด้วย

แม้กระนั้นก็ตาม ระบบแคชรวมศูนย์มีข้อจำกัดหลักๆทางโครงสร้างพื้นฐาน 4 ประการ คือ อย่างแรก การปรับเปลี่ยนขนาดของเครื่องให้บริการมีข้อจำกัด ถ้ามีมากกว่า 1 ตัว มันจะกลายเป็นระบบแคชแบบกระจายไปในทันที อย่างที่สอง ระบบแคชรวมศูนย์ ต้องเพิ่มค่าใช้จ่ายในส่วนของความอดทนต่อความผิดพลาดของเครื่องให้บริการและเครือข่าย อย่างที่สาม การใช้เครื่องให้บริการแคชร่วมกันโดย หลายๆเว็บไซต์ (websites) บังคับให้บางเว็บไซต์ส่งข้อความ สำหรับทุกๆการร้องขอของผู้ใช้ (user) สุดท้าย คือ เป็นธรรมชาติที่ว่าไม่สามารถบรรเทาการจราจรระหว่างเว็บไซต์ได้ ระบบแคชรวมศูนย์ทำให้เพิ่มเวลาการตอบสนองของผู้ใช้ในการส่งเว็บเพจขนาดใหญ่

2.4.2 แคชแบบกระจาย

มีข้อดีดังนี้ อย่างแรก การลงทุนเบื้องต้นมีราคาถูกกว่า ความจุของแคชเพิ่มขึ้นได้ง่าย โดยการเพิ่มเครื่องให้บริการ อย่างที่สอง การออกแบบแคชแบบกระจายที่เหมาะสมจะทำให้เพิ่มความอดทนต่อความผิดพลาดของเครื่องให้บริการและเครือข่าย ซึ่งแคชแบบกระจายสามารถก้าวข้ามหลายๆเว็บไซต์ และบรรเทาการจราจรระหว่างเว็บไซต์ได้ ในทางกลับกัน มีข้อเสียอยู่สองประการ คือ อัตราการ cache-hit จะมีค่าต่ำกว่าแคชรวมศูนย์ และราคาในการบริการงานจะสูงได้ ถ้าไม่มีการรองรับที่เหมาะสมโดยเครื่องให้บริการของมันเอง

เราพิจารณาปัจจัยหลายอย่างในรูปของการแคชแบบมาตราส่วนขนาดใหญ่ (large-scale) สำหรับ www และสรุปว่าแคชแบบกระจายเหนือกว่า โดยที่ความสามารถในการปรับเปลี่ยนได้เป็นปัจจัยสำคัญในการพิจารณามาตราส่วนของการให้บริการ www ที่กำลังเติบโตอย่างรวดเร็วมาก ซึ่งเป็นการยากที่จะรองรับโดยแคชรวมศูนย์ ดังนั้นเราจึงใช้แคชแบบกระจาย และให้ความสนใจกับการจัดการด้วยตัวเอง (self-organizing) ซึ่งเป็นธรรมชาติของแคชแบบกระจายในการแก้ปัญหาความผิดพลาดของเครื่องให้บริการและเครือข่ายรับมือกับสิ่งแวดล้อมของระบบกระจายขนาดใหญ่ และบรรเทาราคาในการบริหารงานอีกด้วย

2.4.3 รูปแบบของแคชแบบกระจาย

รูปแบบของแคชแบบกระจายสามารถแบ่งออกได้เป็น 3 กลุ่ม คือ

- แคชแบบกระจายด้วยรูปแบบลำดับชั้น (Hierarchical distributed cache) มีลักษณะเชื่อมต่อกันแบบลำดับชั้น ซึ่งปกติจะมีการปรับแต่งแบบใช้มือ (manual) แต่ละเครื่องให้บริการจะเปลี่ยน

ทิศทางการร้องขอไปตามลำดับชั้นที่สูงกว่า ระบบแบบนี้เช่น CERN, Squid และ Netscape แม้ว่าเครื่องให้บริการหลายตัวจะทำงานร่วมกันแคชของแต่ละเครื่องให้บริการก็ยังคงแบ่งแยกกัน บางส่วนของพื้นที่แคชเสียไปโดยเปล่าประโยชน์ ทำให้อัตราการ cache-hit มีค่าต่ำ

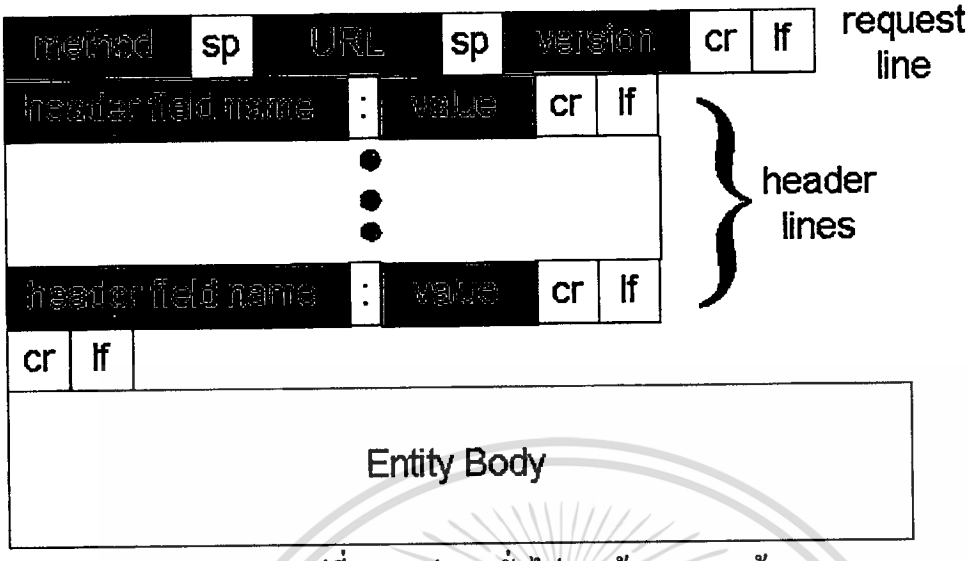
- แคชแบบกระจายด้วยตัวจัดการที่ศูนย์กลาง (centralized manager) จะมีเครื่องให้บริการหนึ่งตัว (หรือมากกว่า) ที่เรียกว่า ตัวจัดการ (manager) อยู่ท่ามกลางเครื่องให้บริการ โดยที่ตัวจัดการจะรักษา cache directory ของเครื่องให้บริการ ข้อมูลแคชจะถูกเคลื่อนย้ายซ้ำไปซ้ำมาไปตามเครื่องให้บริการต่างๆ ในแคชแบบกระจายที่มีอยู่ในแบบนี้ จะมุ่งหมายให้ประยุกต์ใช้กับสถานการณ์ในมาตราส่วนขนาดใหญ่ เช่น World Wide Web เพื่อที่จะฝ่าฟันอุปสรรคของการที่มีตัวจัดการมากเกินไป จัดการกับความผิดพลาด และบรรเทาราคาในการบริหารงานอีกด้วย
- แคชแบบกระจายด้วยรูปแบบสมมาตร (Symmetric distributed cache) ประกอบด้วยหน้าที่ของเครื่องให้บริการแคชที่เท่าเทียมกัน มีความเป็นไปได้ที่จะทำให้เครื่องให้บริการแคชส่ง cache directory ออกไปยังที่อื่น และทำงานร่วมกันอย่างเป็นอิสระ ในระบบนี้จะไม่เกิดความคับคั่งของข้อมูล (bottleneck) ขึ้นในโครงสร้างของมัน ดังนั้นจึงน่าจะเป็นโครงสร้างของแคชที่ดีสำหรับเว็บแคชในยุคถัดไป

2.5 โพรโตคอล HTTP (Hypertext Transfer Protocol)

HTTP โพรโตคอลมีความสัมพันธ์อย่างมากกับเครื่องให้บริการพร็อกซี่หรือเครื่องให้บริการแคช เนื่องจากเป็นกลไกสำคัญในการติดต่อระหว่างเครื่องให้บริการเว็บ (Web Server) กับเว็บบราวเซอร์ (Web Browser) เพื่อทำการส่งเอกสารตามที่ผู้ร้องขอต้องการ ซึ่งการสร้างระบบแคชจำเป็นจะต้องเรียนรู้กลไกการติดต่อด้วยรูปแบบของโพรโตคอลนี้

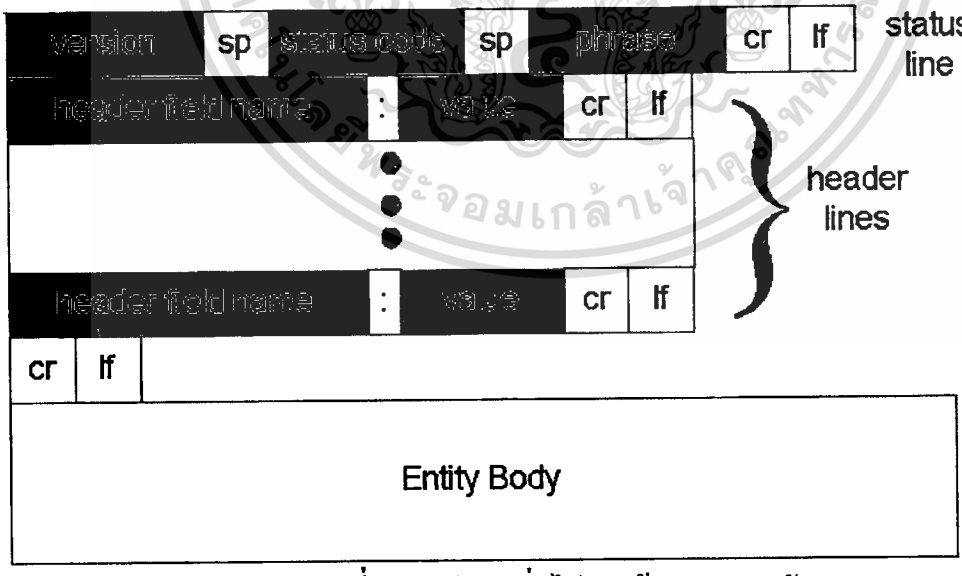
HTTP เป็นโพรโตลอลที่ใช้ร้องขอและตอบรับ ผู้ใช้จะส่งการร้องขอไปยังเครื่องให้บริการเว็บ และเครื่องให้บริการจะส่งข้อความตอบรับกลับมา ไม่มีการ Handshake หลายระดับเหมือนดังโพรโตคอลอื่น เช่น FTP

ข้อความร้องขอของ HTTP ประกอบด้วยวิธีการ (method) และเฮดเดอร์ (header) ต่างๆ รวมไปถึงส่วนที่เป็นข้อมูลที่ต้องการส่งด้วย วิธีการในการติดต่อมีหลายชนิด ที่ใช้มากที่สุดคือ GET ซึ่งใช้ในการดึงเอกสาร ส่วน POST ใช้ในการส่งข้อมูลตัวผู้ใช้ไปยังเครื่องให้บริการ และ HEAD ใช้ในการดึงส่วนเฉพาะที่เป็นเฮดเดอร์มาเท่านั้น รูปแบบของวิธีการและเฮดเดอร์ในโพรโตคอล HTTP เป็นดังรูปที่ 2.5



รูปที่ 2.5: รูปแบบทั่วไปของข้อความการร้องขอ

จากรูปส่วนที่อยู่หลังเฮดเดอร์จะเป็นส่วนของข้อมูล (ถ้ามี) ที่จะส่งไปให้เครื่องให้บริการข้อมูลนั้นอาจจะเป็นรูปแบบของฟอร์มแล้วให้ผู้ใช้ทำการกรอกเพื่อทำการส่งไปให้เว็บนั้นๆ เมื่อเครื่องให้บริการได้รับข้อความการร้องขอแล้ว จะทำการตอบกลับมาได้ข้อความตอบรับ ซึ่งมีรูปแบบคล้ายกับข้อความการร้องขอดังรูปที่ 2.6



รูปที่ 2.6: รูปแบบทั่วไปของข้อความตอบรับ

รูปแบบของรหัสสถานะ (status code) ที่ตอบกลับมาโดยทั่วไปจะมีรูปแบบดังตารางที่ 2.1 ดังนี้คือ

Code	Category	Description
1xx	Informational	รูปแบบนี้บอกว่าการร้องขอไม่ถูกปฏิเสธ แต่ยังไม่ทราบผลที่แท้จริง
2xx	Sucessful	การร้องขอได้รับและการประมวลผลสำเร็จ
3xx	Redirection	ใช้ในการเปลี่ยนทิศทางของผู้ร้องขอไปยัง URL อื่น
4xx	Client Error	ระบุความผิดพลาดในการร้องขอเนื่องจากผู้ร้องขอ
5xx	Server Error	ระบุความผิดพลาดบนเครื่องให้บริการเอง

ตารางที่ 2.1: แบ่งประเภทของรหัสสถานะของ HTTP

รูปแบบที่เห็นทั่วไปที่ระบุในตาราง เช่น

- 200 OK: แสดงว่าการร้องขอประสบผลสำเร็จและข้อมูลได้ถูกส่งกลับมาด้วย
- 301 Moved Permanently: ข้อมูลที่ร้องขอถูกเครื่องย้ายไปยังที่อื่นแล้ว พร้อมทั้งระบุ URL ใหม่ใน Location: เฮดเดอร์ที่ตอบรับกลับมา
- 400 Bad Request: ระบุว่าการร้องขอนั้นเครื่องให้บริการไม่สามารถเข้าใจได้
- 404 Not Found: ข้อมูลที่ร้องขอไม่อยู่ในเครื่องให้บริการนี้
- 505 HTTP Version Not Supported: เครื่องให้บริการไม่รองรับพรโตคอลที่ร้องขอในเวอร์ชั่นนี้

ส่วนเฮดเดอร์ที่ระบุในข้อความตอบรับมีการระบุถึงข้อมูลของเครื่องให้บริการเอง รูปแบบของเอกสารที่ส่งกลับมา รวมทั้งกลไกในการยืนยันความใหม่ของเอกสารซึ่งจะกล่าวถึงในภายหลัง ซึ่งมีอีกหลากหลายเฮดเดอร์ที่ใช้จัดการกับการเชื่อมต่อที่เรียกว่า Persistent Connection ที่ทำให้การร้องขอที่ต่อมาภายหลังใช้เส้นทางการเชื่อมต่อเดิมกับเครื่องให้บริการ เพื่อลดเวลาในการสร้างเส้นทางการเชื่อมต่อ (Round Trip Time:RTT) และการ slow start ได้ โดยใช้รูปแบบเฮดเดอร์แตกต่างกันในแต่ละเวอร์ชั่นของ HTTP ในเวอร์ชั่น HTTP/1.0 จะใช้ *Connection: keep-alive* ในการบอกว่าเครื่องให้บริการจะยังคงเส้นทางเดิม (keep-alive) และยังไม่ตัดการเชื่อมต่อในเส้นทางนั้น ส่วนในเวอร์ชั่น HTTP/1.1 นั้นรูปแบบของ Persistent Connection หรือ Keep Alive นั้นเป็นรูปแบบพื้นฐานอยู่แล้วถ้าไม่มีเฮดเดอร์ที่เกี่ยวข้องระบุ ซึ่งถ้าต้องการไม่ใช้รูปแบบดังกล่าวจะต้องระบุเฮดเดอร์ว่า *Connection: close*

2.6 การควบคุมการแคช

HTTP โพรโตคอลกำหนดทิศทางชัดเจนในการทำการแคช โดยใน HTTP/1.0 ได้กำหนดรูปแบบของ “conditional GET” ขึ้น เพื่อให้ผู้ใช้ได้รับเอกสารอย่างมีเงื่อนไข โดยที่เครื่องให้บริการพร็อกซี่จะทำการตรวจสอบ“up-to-date” ถ้าเอกสารไม่ได้มีการแก้ไขเครื่องให้บริการจะส่งเฮดเดอร์

"Not modified" กลับมา เพื่อให้พร็อกซี่ใช้แคชของตัวเอง มิฉะนั้นเอกสารใหม่จะถูกส่งกลับมาพร้อมกับเฮดเดอร์ "OK response"

conditional GET ใน HTTP/1.0 ใช้ค่าของเฮดเดอร์ตอบรับ Last-Modified ซึ่งได้รับพร้อมกับเอกสารเมื่อถูกดึงมาเก็บไว้ในแคช และส่งค่านี้ไปกับเฮดเดอร์ร้องขอ If-Modified-Since ดังตัวอย่างต่อไปนี้ :

การร้องขอครั้งแรกจะได้รับเฮดเดอร์ตอบรับจากเครื่องให้บริการ คือ

HTTP/1.0 200 OK

Server: Netscape-Enterprise/2.0

Date: Sat, 19 Apr 1997 10:22:00 GMT

Last-modified: Fri, 18 Apr 1997 15:12:05 GMT

Content-type: text/html

Content-length: 6510

เมื่อทำการร้องขออีกครั้งแบบ condition GET เวลาที่ได้รับจากเฮดเดอร์ Last-Modified จะถูกส่งไปกับการร้องขอในเฮดเดอร์ If-Modified-Since

GET /people/ari/ HTTP/1.0

User-agent: Mozilla/3.0

*Accept: text/html, image/gif, image/jpeg, */**

If-modified-since: Fri, 18 Apr 1997 15:12:05 GMT

ถ้าเอกสารไม่มีการเปลี่ยนแปลง เครื่องให้บริการจะตอบกลับด้วยเฮดเดอร์ 304 Not modified ดังนี้

HTTP/1.0 304 Not modified

Server: netscape-Enterprise/2.0

Date: Sun, 20 Apr 1997 15:45:12 GMT

ด้วย conditional GET การส่งเอกสารจริงจะเกิดขึ้นกรณีเดียวเมื่อเอกสารมีการเปลี่ยนแปลง วิธีการนี้จะช่วยรักษาแบนด์วิธและบรรเทาการติดต่อไปยังเครือข่ายภายนอกได้

HTTP/1.1 ได้กำหนดเฮดเดอร์มากขึ้นสำหรับการร้องขอในเงื่อนไขอื่น เช่น การให้ความสำคัญกับการป้องกันข้อมูลสูญหายเมื่อข้อมูลถูกแก้ไขบนเครื่องให้บริการเว็บโดยใช้วิธี HTTP PUT มีเฮดเดอร์หลายอย่างซึ่งใช้ในการแก้ไขอย่างมีเงื่อนไขดังนี้:

- *It-Modified-Since*: จะคืนเอกสารถ้ามีการเปลี่ยนแปลงตั้งแต่เวลาที่ระบุ ใช้ตรวจสอบแคชว่า up-to-date หรือไม่

- *If-Unmodified-Since*: จะคืนเอกสารถ้าไม่มีการเปลี่ยนแปลงตั้งแต่เวลาที่ระบุ ใช้ประโยชน์เมื่อผู้ขอใช้บริการเก็บข้อมูลบางส่วนของเอกสารอยู่แล้ว และต้องการดึงเพียงส่วนที่เหลือ เพื่อให้แน่ใจว่าได้รับเอกสารชุดเดิม
- *If-Match*: มุ่งหมายเพื่อให้การตรวจสอบ up-to-date มีประสิทธิภาพ ซึ่งสามารถใช้รับประกันว่าเอกสารไม่ได้มีการเปลี่ยนแปลงตั้งแต่เอกสารถูกดึงก่อนทำการแก้ไข
- *If-None-Match*: ใช้คล้ายกับการใช้ If-Match แต่ด้วยเงื่อนไขที่ตรงกันข้ามกัน
- *If-Range*: เป็นแนวทางที่ปรับปรุงของ If-Unmodified-Since สำหรับการร้องขอเป็นลำดับ ถ้าเอกสารมีการเปลี่ยนแปลง อันดับที่มีอยู่ของผู้ขอใช้บริการจะเป็นout-of-dateและถูกดึงออกมาด้วยเฮดเดอร์ If-Range เป็นไปได้ที่จะระบุการร้องขอเป็นคู่ นั่นคือถ้าเอกสารไม่มีการเปลี่ยนแปลง ลำดับที่ระบุจะถูกคืนกลับมา ถ้าเอกสารมีการเปลี่ยนแปลง เอกสารทั้งหมดจะถูกคืนมาแทน

2.7 การยืนยันความใหม่ของเอกสารที่ถูกแคช

แม้ว่าโพรโตคอล HTTP/1.0 จะมีการรวมข้อกำหนดบางอย่างเกี่ยวกับพร็อกซี่และการแคช แต่ก็มีข้อมูลที่ระบุถึงการควบคุมพวกมันเพียงเล็กน้อยเท่านั้น ข้อความตอบรับจากเครื่องให้บริการที่เกี่ยวกับการแคชมีดังนี้

- *Last-Modified*: ระบุถึงเวลาที่สร้างหรือเปลี่ยนแปลงเอกสารครั้งสุดท้าย เป็นค่าที่ใช้ในเฮดเดอร์ If-Modified-Since ด้วยการร้องขอแบบ conditional GET
- *Expires*: ระบุถึงเวลาหมดอายุ ซึ่งทำให้กลายเป็นเอกสารล้าสมัยที่ไม่ควรส่งให้ผู้ขอใช้บริการโดยปราศจากการ up-to-date
- Pragma: no-cache ระบุในการตอบรับว่าเอกสารไม่ควรถูกแคช ในHTTP/1.0ไม่ได้มีการระบุอย่างเป็นทางการ แต่ในทางปฏิบัติมีผู้ขอใช้บริการและเครื่องให้บริการพร็อกซี่รองรับรูปแบบนี้ ในHTTP/1.1 แทนการทำงานนี้ด้วยเฮดเดอร์ cache-control

จะเห็นได้ว่าการใช้ conditional GET อย่างง่ายๆสำหรับทุกการร้องขอในการตรวจสอบว่าสำเนาในแคชยังคง up-to-date จะเป็นการรับประกันว่าข้อมูลจะไม่ล้าสมัย อย่างไรก็ตามในทางปฏิบัติเป็นการสิ้นเปลือง เนื่องจากการร้องขอจะได้รับการตอบรับ"not-modified"เป็นส่วนใหญ่ และสำเนาในแคชจะถูกใช้ ทำให้ประสิทธิภาพไม่เป็นที่น่าพอใจ

การแคชด้วยการตรวจสอบ up-to-date ทุกครั้งจะยังคงช่วยรักษาแบนด์วิธและช่วยด้านความเร็วโดยรวม แต่อย่างไรก็ตามเวลาในการติดต่อไปยังผู้ให้บริการจะไม่ลดลง ความจริงแล้วอาจจะเพิ่มขึ้นด้วยซ้ำ เนื่องจากความจริงที่ว่าการเชื่อมต่อไปที่ เครื่องให้บริการต้นกำเนิดต้องการการ

สร้างเส้นทางเชื่อมต่อทุกครั้ง การได้เปรียบจากความเร็วจากการแคชจะมีผลอย่างเดียวเมื่อมีการตอบรับ "not-modified"จากเครื่องให้บริการต้นกำเนิด

สำหรับเหตุผลนี้ จะมีการหลีกเลี่ยงการตรวจสอบ up-to-date ถ้าเอกสารนั้นยังไม่น่าจะมีการเปลี่ยนแปลง โดยเครื่องให้บริการพร็อกซี่จะทำการประเมินว่าเมื่อการตรวจสอบ up-to-date ไม่จำเป็น ก็จะส่งเอกสารจากแคชโดยไม่ทำการติดต่อกับเครื่องให้บริการต้นกำเนิดเลย

2.8 การหมดอายุของเอกสาร

จริงๆแล้วใน HTTP/1.0 ใช้เพียงเฮดเดอร์ Last-Modified ในการทำการตรวจสอบ up-to-date ส่วน Expires และ Pragma ใช้น้อยมากและบ่อยครั้งใช้ในการป้องกันการแคชบนพร็อกซี่

ปกติแล้วใช้อายุของเอกสารบอกเวลาในการตรวจสอบ up-to-date เป็นการประมาณว่าเมื่อใดเอกสารจะยังคงไม่ถูกเปลี่ยนแปลง บ่อยครั้งที่มีเอกสารที่ถูกสร้างขึ้นครั้งเดียวโดยไม่มีการเปลี่ยนแปลงอีกเลย เช่น รูปภาพ หนังสือแบบonlineหรือพวกคู่มือติดตั้ง เป็นต้น การตรวจสอบup-to-date ไม่ต้องทำบ่อยสำหรับเอกสารเหล่านี้

ในทางตรงข้าม ถ้าเอกสารต้องการความใหม่เสมอ อาจระบุให้มีการแก้ไขในช่วงเวลาที่กำหนด เช่น online schedules , home pages เอกสารชนิดนี้ควรมีการตรวจสอบบ่อยๆ

อย่างไรก็ตาม เอกสารที่มีการเปลี่ยนแปลงบ่อยๆ ควรมีการกำหนดเฮดเดอร์ Expires บอกเวลาการหมดอายุของเอกสาร อย่างเช่น

Expires: 0

Expires: now

Expires: Thu, 01 Jan 1970 00:00:00 GMT

Expires: current time

อายุของเอกสารจากเวลาของการดึงข้อมูลครั้งสุดท้ายหรือเวลาการทำตรวจสอบ up-to-date คำนวณอย่างง่ายดังนี้

$$age = last\_check\_time - last\_modified\_time$$

การประมาณค่า freshness time จะถูกคำนวณโดยใช้ปัจจัยที่เรียกว่า "Last-Modified factor" หรือ "LM factor" โดยการคูณด้วยอายุ

$$freshness\_time = age \times lm\_factor$$

การประมาณค่า expiration time จะถูกคำนวณโดยบวกค่า freshness time กับ เวลาในการดึงข้อมูลครั้งสุดท้าย หรือ เวลาในการตรวจสอบ up-to-date

$$expiration\_time = last\_check\_time + freshness\_time$$

2.9 CACHE HIT RATIO

วิธีหนึ่งการวัดประสิทธิภาพของการแคช จะวัดจาก cache hit ratio :

- cache hit ratio เป็นจำนวนของ cache hit หารด้วย การร้องขอทั้งหมด
- cache hit หมายถึง เอกสารซึ่งถูกเก็บในแคชและไม่ล้าสมัย(not stale)
- เอกสารที่ไม่ล้าสมัย คือ เอกสารนั้นยังคงใหม่ หรือเป็นเอกสารเก่าแต่เมื่อตรวจสอบด้วย up-to-date แล้ว ให้ผล "not modified" ซึ่งจะทำให้เป็นข้อมูลแคชที่ใหม่อีกครั้ง

มีcache hit แบ่งเป็นสองชนิดใหญ่ๆ จะเห็นได้ว่าcache hit ที่ไม่มีการตรวจสอบ up-to-date จะเร็วกว่า โดยหลีกเลี่ยงความล้าช้าจากการเชื่อมต่อไปยังเครื่องให้บริการที่อยู่ไกล

อัตราการ cache hit ใน HTTP/1.0 มีแปรผันอยู่ระหว่าง 20 และ 70 เปอร์เซ็นต์ โดยปกติจะอยู่ระหว่าง 30-60 เปอร์เซ็นต์ อัตราการcache hitจะต่ำสุดบนพร็อกซี่เมื่อมีผู้ใช้และการร้องขอน้อยที่สุด นั่นคือจำนวนผู้ใช้และการร้องจะเพิ่มอัตรา cache hit

การเข้าถึงจุดcritical mass ของผู้ใช้และการร้องขอ จะทำให้อัตราcache hitที่ต่ำ เพิ่มขึ้นอย่างรวดเร็ว อัตราcache hit ต่ำเมื่อมีเพียงผู้ใช้สองสามคน เนื่องจากความจริงที่ว่าเป็นไปได้ยากที่ผู้ใช้สองสามคนนั้นจะเข้าไปในเว็บไซค์เดียวกัน เมื่อจำนวนผู้ใช้มากขึ้นก็ทำให้มีผู้ใช้คนอื่นเข้าเยี่ยมชมเว็บเดียวกันมากขึ้น จำนวนที่เพิ่มขึ้นเหนือ critical mass จะทำให้ไม่น่าเป็นไปได้ที่จะมีเว็บไม่เคยมีผู้ใช้คนอื่นเยี่ยมชมก่อน

เมื่อถึงจุดอิ่มตัว การเพิ่มจำนวนผู้ใช้จะไม่เพิ่มอัตรา cache hit อีก เหตุผลส่วนหนึ่งเนื่องจากความจริงที่ว่าบางส่วนของเอกสารถูกสร้างแบบ dynamic และไม่สามารถแคชได้ เหตุผลอื่นๆก็คือแม้ว่าจะมีจำนวนผู้เข้าชมเว็บเป็นจำนวนมาก แต่ก็มีข้อจำกัดของจำนวนสูงสุดที่เข้าชมได้ ซึ่งทำให้อัตราhit จะเข้าใกล้ 100 เปอร์เซ็นต์ มีบางที่จำกัดให้เข้าชมได้เพียงครั้งเดียวโดยผู้ใช้คนเดียว และส่วนใหญ่จะไม่สามารถเข้าถึงได้

2.10 การควบคุมการแคชด้วย HTTP/1.1

HTTP/1.1 ได้กล่าวถึงการแคชซึ่งเพิ่มจาก HTTP/1.0 ที่มีการควบคุมการแคชแบบง่ายๆ ให้มีระดับควบคุมการแคชใหม่ๆที่สมบูรณ์ขึ้น

HTTP/1.1 แบ่งจุดมุ่งหมายการแคชเป็นสองประเภท คือ การขจัดการส่งการร้องขอในบางกรณี และขจัดการส่งการตอบสนองแบบเต็มรูปแบบในอีกบางกรณี อย่างแรกจะช่วยลดความล่าช้าโดยการขจัดการเชื่อมต่อไปสู่ภายนอก และอย่างหลังเป็นการบรรเทาการใช้แบนด์วิธ

2.10.1 ความใหม่ของเอกสาร

HTTP/1.1 แนะนำ"ความใหม่"สำหรับเอกสารที่ถูกแคชไว้ดังนี้

- เอกสารใหม่ เป็นเอกสารที่ไม่ล้าสมัย(stale)
- เอกสารล้าสมัย เป็นเอกสารที่ไม่สามารถใช้อะไรได้ถ้าปราศจากการยืนยันว่ายังคง up-to-dateอยู่

เอกสารจะใหม่เมื่อ

- เพิ่งถูกส่งมาจากเครื่องให้บริการต้นกำเนิดหรือ
- เมื่อเครื่องให้บริการต้นกำเนิดถูกติดต่อเพื่อทำการตรวจสอบ up-to-date หรือ
- เมื่ออายุ(age)ของมันมีมากกว่า freshness lifetime ของมัน

2.10.2 อายุของเอกสาร(Age of Objects)

HTTP/1.1 เสนอแนะแนวคิดของอายุของการแคชเอกสารว่าเป็นเวลาที่ผ่านไปตั้งแต่เวลาที่เอกสารต้นฉบับถูกส่งมา หรือเป็นเวลาตั้งแต่มีการทำ up-to-date ครั้งสุดท้าย HTTP ไม่ต้องการการเทียบสัญญาณนาฬิกา ดังนั้นการคำนวณอายุไม่ต้องการเวลาจริงแต่เพียงแค่ประมาณเท่านั้น

HTTP โพรโตคอลต้องการให้เครื่องให้บริการต้นกำเนิด ส่งเฮดเดอร์ Date เพื่อบอกวันและเวลาในการสร้างผลตอบสนอง โดยค่านี้จะใช้ในการคำนวณอายุ อย่างไรก็ตาม เวลาในการนำผลตอบสนองส่งผ่านเครือข่ายและได้รับโดยผู้ขอใช้บริการจะต้องถูกนำมาพิจารณาด้วย

ต่อไปจะอธิบายขั้นตอนทีละขั้น ในการได้อายุของเอกสาร อายุที่เห็นได้ชัดของเอกสารเป็นความแตกต่างระหว่างเวลาปัจจุบัน(เวลาของการได้รับผลตอบสนอง) และเวลาที่ปรากฏในเฮดเดอร์ Date ซึ่งบอกถึงเวลาในการสร้างผลตอบสนองโดยเครื่องให้บริการต้นกำเนิด

apparent_age = response_time – date_value

อย่างไรก็ตาม เนื่องจากเวลาคลาดเคลื่อนไป มีผลให้เกิดค่าลบได้HTTP/1.1จึงระบุให้ค่าเป็นศูนย์เมื่อได้ผลเป็นลบ

apparent_age = max(0,response_time – date_value)

มีทางอื่นที่จะหาค่าอายุของผลตอบสนองเมื่อค่าเวลาปัจจุบันสามารถใช้ได้โดยตรง อย่างไรก็ตาม จะเกิดความเสี่ยงขึ้นถ้ามี HTTP/1.0พร็อกซี่ ซึ่งไม่รู้จักเฮดเดอร์ Age แทรกระหว่างทาง ทำให้ได้ค่าเฮดเดอร์ Age ที่ไม่ได้ผ่านการแก้ไขจาก HTTP/1.1 ในการพิจารณาที่ถูกต้องจะใช้ค่าที่เหนือกว่าค่า apparent_age และค่าที่มากกว่าจะถูกใช้

corrected_received_age=max(apparent_age,age_value)

การคำนวณ apparent_age ก่อนหน้านี้ไม่ได้นำเวลาที่ใช้ในการเดินทางผ่านเครือข่ายของการตอบรับไปถึงผู้รับ เป็นไปได้ที่จะเดินทางผ่านเครื่องให้บริการพร็อกซี่หนึ่งตัวหรือมากกว่านั้น ในทางปฏิบัติแล้ว การตอบรับบางครั้งสร้างขึ้นระหว่าง request_time และ response_time แต่ค่าที่ถูกต้องแน่นอนไม่สามารถคำนวณได้ เวลาในการติดต่อผ่านเครือข่ายอาจเกิดความล่าช้าที่ไม่อาจ

คาดเดาได้ ดังนั้นการสมมุติว่าเวลาที่ได้รับการตอบรับเป็นค่าเฉลี่ยของ request_time และ response_time จึงไม่ถูกต้อง

เฮดเดอร์ Date ไม่สามารถถูกใช้เนื่องจากความคลาดเคลื่อนของเวลา ดันนั้นจะต้องสมมุติว่าความล่าช้าจากเครื่องให้บริการในการสร้างการตอบรับ ถึงเวลาที่ได้รับจริง มีค่าใกล้เคียงความล่าช้าทั้งหมดระหว่างการร้องขอที่ส่งไปและผลตอบรับที่ได้ ในทางปฏิบัติ ความล่าช้าทั้งหมดจะใช้ค่า

response_delay = response_time – request_time

ความล่าช้านี้ถูกเพิ่มเข้าไปในค่าของอายุที่ได้รับอย่างถูกต้องดังนี้

corrented_initial_age = corrected_received_age + response_delay

อายุตั้งต้นที่ถูกต้องเป็นการประมาณของอายุมากที่สุดของเอกสารที่เวลาของผู้รับ

ขณะนี้ เมื่อพร็อกซี่สร้างการตอบรับจากแคชของตัวเอง จะส่งเฮดเดอร์ Age ซึ่งเป็นผลรวมของอายุตั้งต้นที่ถูกต้อง และเวลาที่การตอบรับถูกเก็บบนแคชท้องถิ่น การคำนวณ resident_time จะได้

resident_time = now – response_time

และเฮดเดอร์ Age ซึ่งเครื่องให้บริการพร็อกซี่ส่งให้ เมื่อส่งการตอบรับจากแคชของตัวเองมีค่า

current_age = corrected_initial_age + resident_time

2.10.3 Freshness Lifetimeของเอกสาร

Freshness lifetime ของเอกสารเป็นเวลาซึ่งเอกสารถูกแคชและใช้โดยปราศจากการตรวจสอบ up-to-date ซึ่งก็คือ ช่วงเวลาที่เอกสารถูกพิจารณาว่ายังใหม่อยู่นั่นเอง หลังจากหมดช่วง freshness lifetime เอกสารจะล้าสมัยและต้องการการตรวจสอบ up-to-date

มีทางเลือกหลายทางในการได้มาซึ่งค่า freshness lifetime เช่น กำหนดเป็นค่า max-ageที่ระบุในเฮดเดอร์ Cache-Control

freshness_lifetime = max_age_value

มิฉะนั้นถ้าเฮดเดอร์ Expires ระบุวันและเวลาของการหมดอายุอย่างชัดเจน ค่านี้อาจจะถูกใช้ได้ โดยค่า freshness lifetime จะเป็นดังนี้

freshness_lifetime = expires_value – date_value

ในการหาว่าเอกสารใหม่หรือล้าสมัย ค่าfreshness lifetime จะถูกเปรียบเทียบกับอายุปัจจุบัน

response_in_fresh = (freshness_lifetime > current_age)

2.11 การจัดการระบบแคช (Cache Management)

ข้อมูลที่ถูกแคชในระบบการจัดเก็บแบบแฟ้ม (file system) นั้น เมื่อจัดเก็บไประยะหนึ่ง จะมีขนาดเพิ่มมากขึ้นอย่างรวดเร็ว เราจึงจำเป็นต้องทำการจำกัดขนาดของแคชไว้ว่า ต้องการที่จะจัดเก็บข้อมูลในแคชเป็นจำนวนเท่าใด การจำกัดขนาดของแคชนี่เอง จะส่งผลให้เมื่อจัดเก็บไปเรื่อยๆ อาจทำให้แคชเต็มได้ จึงต้องมีการลบแคชบางส่วนออกไป ทั้งนี้ทั้งนั้นในการพิจารณาว่าจะลบแคชส่วนใดออกไปนั้น จำเป็นต้องมีวิธีการในการพิจารณาแทนที่แคช (cache replacement algorithm) เพราะว่าถ้าลบแคชส่วนที่มีการใช้งานบ่อยๆออกไป อาจทำให้ประสิทธิภาพของระบบลดลง อัตราการแคช (Cache Hit) ลดลง ส่งผลให้เวลาการตอบสนองลดลงด้วย

มีการศึกษาถึงวิธีการแทนที่แคชไว้อย่างมากมายด้วยกัน ยกตัวอย่างเช่น first-in first-out (FIFO), least frequently used (LFU), least recently used (LRU) และยังมีการแตกแขนงวิธีการ LRU ซึ่งพิจารณาขนาดของเอกสาร (LRU-MIN และ LRU-SIZE) วิธีการต่างๆเหล่านี้ ได้มีการศึกษาเพิ่มเติมและแตกแขนงวิธีการออกไปมากมาย ซึ่งจะไม่กล่าวถึงในที่นี้

วิธีการในการแทนที่แคชที่ใช้ในโครงการนี้ ใช่วิธีที่เรียกว่า Least-Frequently-Used (LFU) ซึ่งวิธีการดังกล่าว จะพิจารณาลบแคชที่มีถูกเรียกใช้งานน้อยที่สุดออกไป เนื่องจากว่า การจัดเก็บข้อมูลที่ถูกใช้บ่อยๆไว้นั้น จะทำให้อัตราการแคช (Cache Hit Ratio) สูง วิธีการที่ใช้ในโครงการนี้ จะทำการเพิ่มเฮดเดอร์พิเศษเข้าที่ดังนี้

LFU = 0

โดยเพิ่มเข้าไปที่ส่วนแรกสุดของข้อมูลที่เราทำการจัดเก็บในแคช เมื่อได้รับข้อมูลนั้นมาครั้งแรกจะกำหนดให้ LFU = 0 โดยที่ เมื่อข้อมูลนั้นถูกเรียกใช้ครั้งถัดไป จะเกิด cache hit ขึ้น แล้วจะทำการเพิ่ม LFU ขึ้น 1 ค่า ได้ LFU = 1 ทำการเพิ่มค่าเช่นนี้ไปเรื่อยๆ

เมื่อแคชเต็มจะมีการพิจารณาลบข้อมูลโดยอ้างอิง LFU เฮดเดอร์ที่ได้เพิ่มเข้าไป จะทำการตรวจสอบข้อมูลที่มี LFU = 0 ก่อน แล้วทำการลบข้อมูลที่มี LFU = 0 ทิ้งไป ทำการตรวจสอบอีกครั้งว่าความจุของข้อมูลน้อยกว่าขนาดของแคชที่กำหนดหรือไม่ ถ้าไม่ จะพิจารณาข้อมูลที่มี LFU = 1 ต่อไป และทำการลบข้อมูลนั้นทิ้ง พิจารณาความจุแคชอีกครั้ง แล้วทำตามขั้นตอนเดิมจนกระทั่งได้ความจุของข้อมูลที่น้อยกว่าความจุของแคชที่กำหนด

เมื่อทำการลบข้อมูลที่มี LFU ต่ำๆออกไปแล้ว ส่งผลให้ LFU ที่เหลือ อาจมีค่าสูงและห่างจากค่า LFU ของข้อมูลที่ถูกแคชเข้ามาใหม่ ทำให้เสียเวลาประมวลผลส่วนหนึ่ง จึงได้ทำการพิจารณาลบค่า LFU ที่น้อยที่สุดที่ไม่ถูกลบ ให้ลดลงมาจนเหลือค่า LFU = 1 ส่วนค่า LFU อื่นๆ ก็ถูกลดลงมาในปริมาณที่เท่ากัน

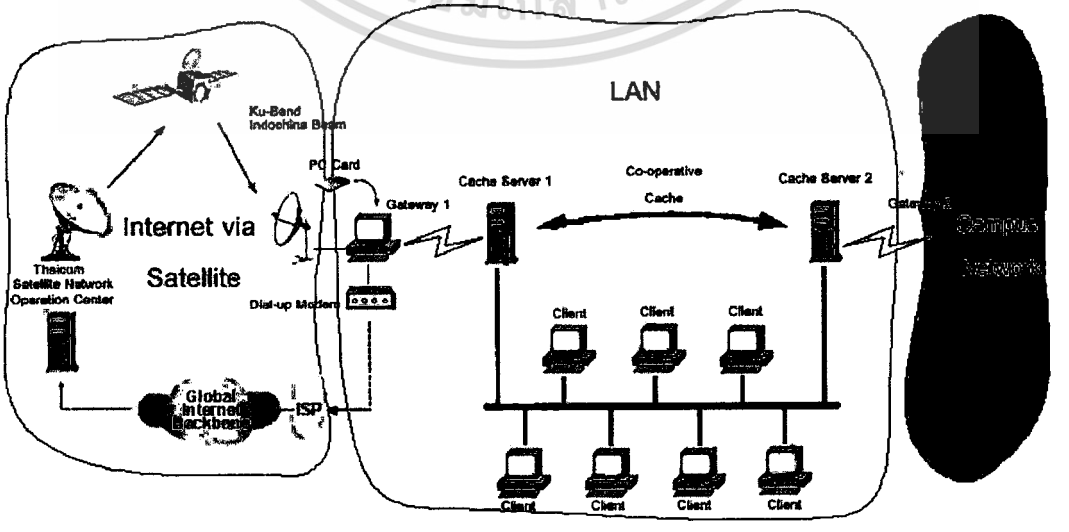
วิธีการดังกล่าว จะสามารถทำให้การจัดเก็บแคชและการแทนที่แคชเป็นไปอย่างมีประสิทธิภาพมากขึ้น ส่งผลให้ประสิทธิภาพโดยรวมของแคชมีเพิ่มมากขึ้นด้วย ทั้งนี้แล้ว ในโครงงานนี้ยังได้พิจารณาลบส่วนของข้อมูลที่ไม่จำเป็นในแคชทิ้งออกไป เช่น ส่วนของข้อมูลที่ล้าสมัยไปแล้ว โดยพิจารณาจากเฮดเดอร์ Expires: ในโพรโตคอล HTTP ในช่วงเวลาที่กำหนดไว้ระยะหนึ่ง เพื่อเพิ่มเนื้อที่ของแคชให้มีมากขึ้น

บทที่ 3
การออกแบบระบบแคชที่ทำงานร่วมกัน

3.1 ปัญหาของระบบ

จากที่ได้กล่างถึงไปข้างต้นเกี่ยวกับรูปแบบของการสื่อสารผ่านดาวเทียมซึ่งมีข้อจำกัดหลักในเรื่องของการหน่วงเวลาของช่องสัญญาณ จึงต้องการระบบแคชเข้ามาช่วยลดผลกระทบเนื่องจากการร้องขอข้อมูลผ่านแคชนั้น อาจจะไม่มีการร้องขอออกไปยังเครือข่ายภายนอกจริงๆ เพราะว่าข้อมูลที่เคยถูกร้องขอ จะถูกเก็บไว้ในแคช ทั้งนี้จะต้องมีกระบวนการยืนยันความใหม่ของเอกสารด้วยเพื่อรับรองว่าเอกสารที่ได้รับจากแคชมีความทันสมัยอยู่เสมอ

เมื่อความต้องการระบบข้างต้นได้ถูกพิจารณาแล้ว การเพิ่มระบบอินเทอร์เน็ตผ่านดาวเทียมเข้ามา ทำให้ระบบเก่าไม่สามารถใช้งานร่วมกันได้ ต้องทำการปรับแต่งรูปแบบการเชื่อมต่อใหม่ จึงได้ทำการออกแบบระบบแคชซึ่งไม่เพียงแต่ให้บริการแคชข้อมูลเท่านั้น แต่ให้มีการติดต่อระหว่างแคชด้วยกัน เป็นรูปแบบของแคชแบบกระจาย (Distributed Cache) ซึ่งเป็นไปในลักษณะของเก็ตเวย์ของแคช (Cache Gateway) เข้ามาช่วยให้ระบบของเราสามารถใช้เส้นทางการร้องขอข้อมูลได้ทั้งสองเส้นทาง ได้แก่ เส้นทางที่เชื่อมต่อกับอินเทอร์เน็ตผ่านดาวเทียมเป็นเกตเวย์หนึ่ง และเชื่อมต่อกับทางสถาบันฯเป็นอีกเกตเวย์หนึ่ง เป็นการช่วยเพิ่มแบนด์วิธและช่วยเพิ่มประสิทธิภาพในการแคชได้อีกด้วย ดังแสดงในรูปที่ 3.1



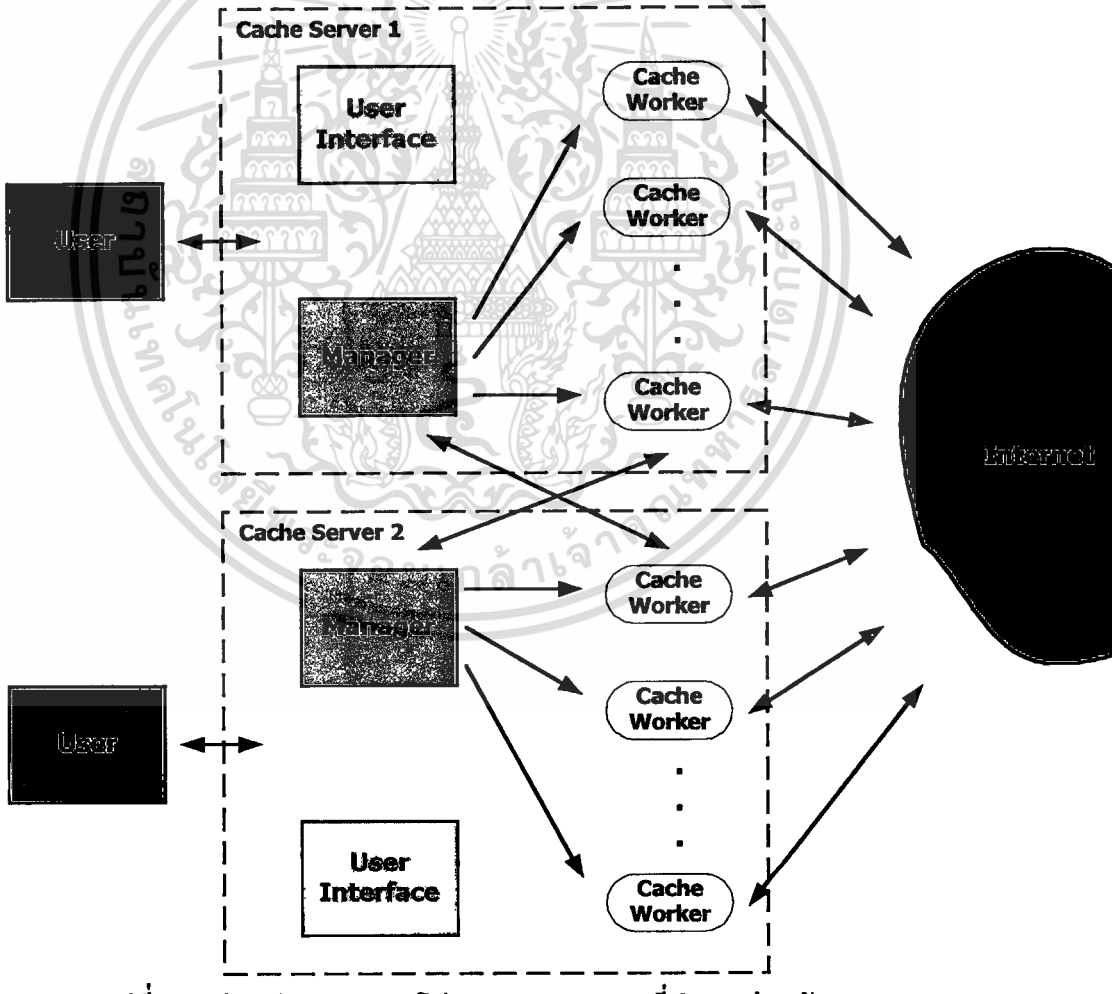
รูปที่ 3.1: โครงสร้างของระบบเครือข่ายที่ทำการออกแบบ

3.2 การออกแบบการทำงาน

รูปแบบการทำงานแบ่งออกเป็น 3 ส่วนหลักๆ ดังนี้

- ส่วนที่ทำการติดต่อกับผู้ใช้ (User Interface)
- ส่วนของตัวจัดการ (Manager)
- ส่วนของระบบแคช

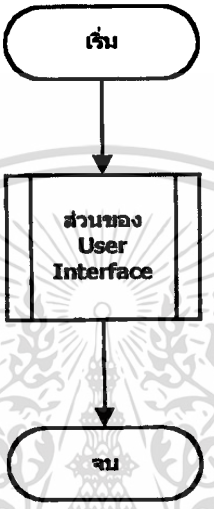
ส่วนที่ใช้ติดต่อกับผู้ใช้จะเป็นส่วนแรกที่เห็นเมื่อโปรแกรมเริ่มทำงาน แต่ในส่วนของเครื่องให้บริการแคชนั้น แต่ละส่วนจะทำงานประสานกันโดยมีตัวจัดการส่วนกลาง (Manager) อยู่เพื่อเป็นตัวกลางในการพิจารณาส่งต่อการร้องขอไปยังส่วนของระบบแคช ให้สามารถทำงานร่วมกันดังรูปที่ 3.2



รูปที่ 3.2: ส่วนประกอบของโปรแกรมระบบแคชที่ทำงานร่วมกัน

3.3 ส่วนของโปรแกรมหลัก (Main Program)

เมื่อเริ่มต้นทำงานของโปรแกรม ในส่วนโปรแกรมหลักจะเป็นเพียงการสร้างส่วนที่ใช้ติดต่อกับผู้ใช้ (User Interface) ขึ้นมาดังรูปที่ 3.3 แล้วส่วนของการติดต่อกับผู้ใช้จะเป็นส่วนที่ใช้กำหนดให้ส่วนอื่นๆทั้งหมดทำงาน ซึ่งจะกล่าวถึงต่อไป



รูปที่ 3.3: ส่วนของโปรแกรมหลัก

3.4 ส่วนที่ทำการติดต่อกับผู้ใช้ (User Interface)

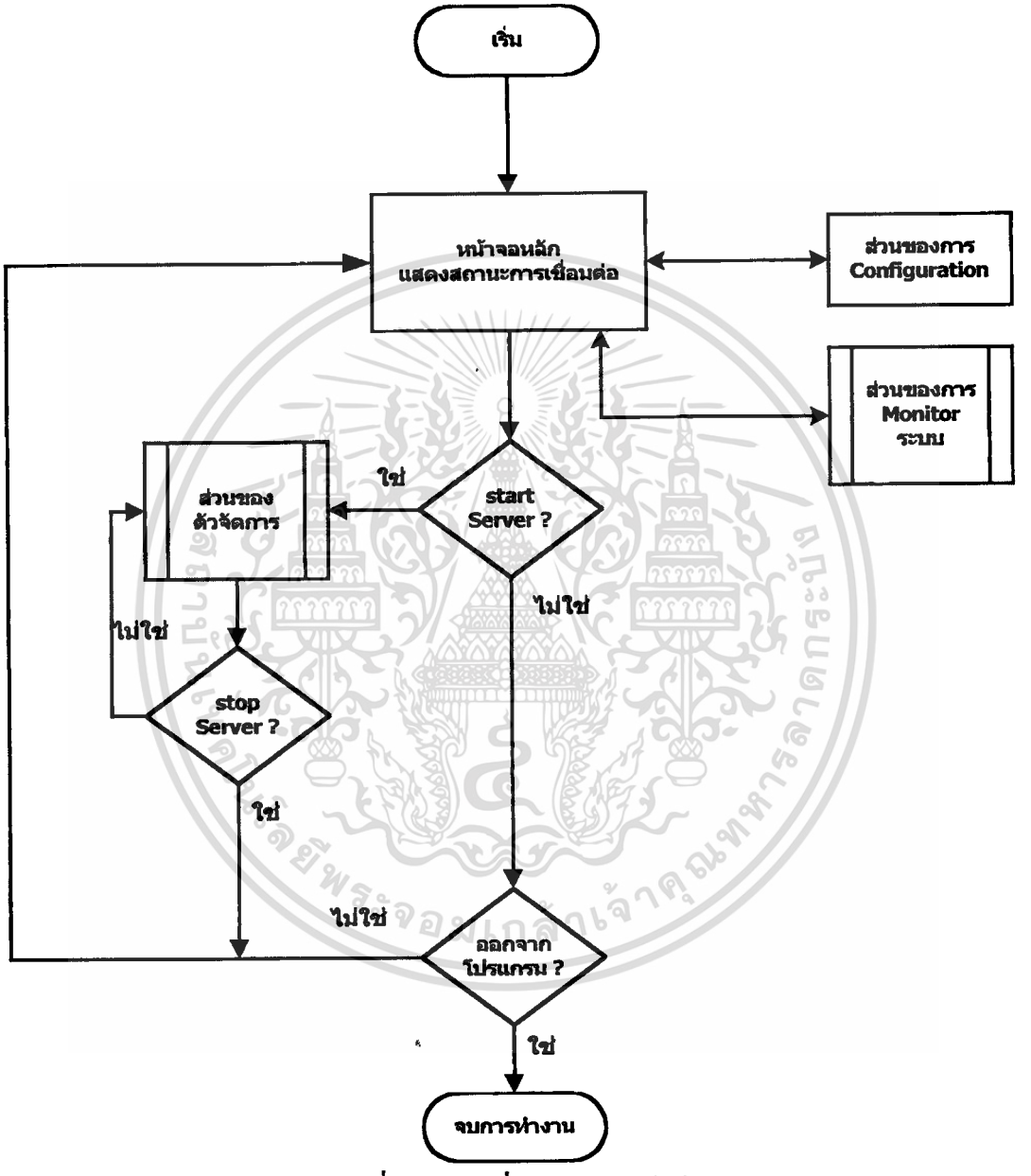
ในส่วนนี้จะใช้ทำการแสดงสถานะของแคชที่เชื่อมต่ออยู่ รวมทั้งผู้ใช้ด้วย โดยจะแยกออกเป็นส่วนที่ใช้ทำการปรับแต่งค่าตัวแปร (Configuration) และส่วนที่ใช้ตรวจสอบระบบแคชเอง เช่น ตรวจสอบ cache hit ratio, ตรวจสอบผลตอบสนอง เป็นต้น ดังรูปที่ 3.4

ส่วนของหน้าจอหลัก จะแสดงสถานะว่าในขณะนั้นว่ามีผู้ใช้ใดทำการเชื่อมต่ออยู่ และมีแคชใดบ้างในเครือข่ายที่ใช้งานอยู่ ฉะนั้น เราจะทราบว่ามีเส้นทางการเชื่อมต่อกี่เส้นทางที่สามารถส่งต่อการร้องขอต่อไปได้ โดยจะมีส่วนที่ใช้ในการตรวจสอบสถานะตลอดเวลาว่ามีแคชใดเชื่อมต่อหรือออกจากระบบไปบ้าง

ส่วนสำคัญของส่วนนี้คือ การกำหนดให้เครื่องให้บริการทำงานหรือหยุดทำงาน เมื่อให้เครื่องให้บริการทำงาน จะเป็นการสร้างส่วนที่เป็นตัวจัดการ (Manager) ขึ้นมา ให้ใช้จัดการการส่งต่อการร้องของไปยังส่วนของระบบแคชอีกที

เมื่อเข้าส่วนของหน้าจอหลักแล้ว สามารถเข้าไปในส่วนของการปรับแต่งค่าตัวแปรที่ใช้ในการกำหนดการทำงานของแคชเพื่อให้แคชทำตามเป้าหมายที่เราต้องการ รวมทั้งจะมีส่วนของการ

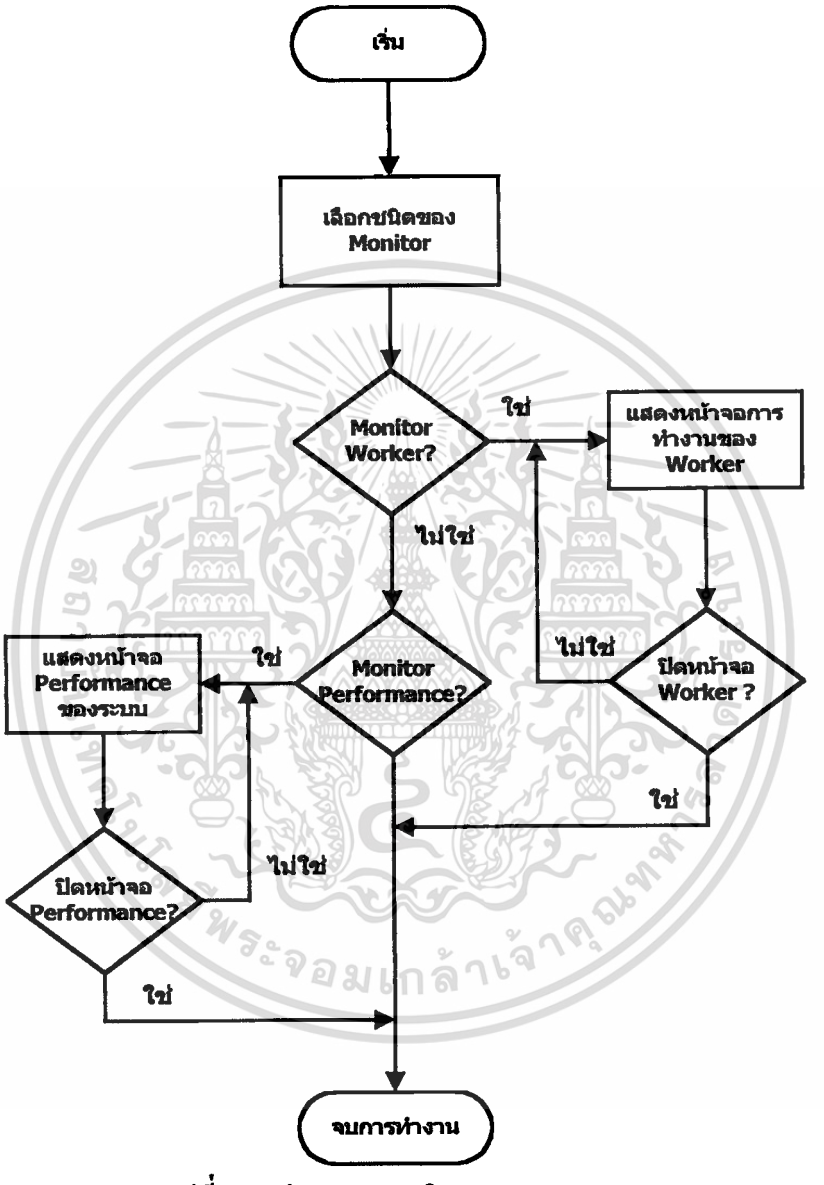
ติดตามผลตอบสนองเกี่ยวกับแคชตัวนั้นว่า มีประสิทธิภาพเช่นไร โดยวัดจาก cache hit ratio, response time เป็นต้น



รูปที่ 3.4: ส่วนที่ใช้ติดต่อกับผู้ใช้

สำหรับส่วนของการติดตามผลการทำงานของระบบนั้นแสดงดังรูปที่ 3.5 มีฟังก์ชั่นการทำงานอยู่ 2 ลักษณะคือ ติดตามผลของการโพรเซสระหว่างที่แต่ละผู้ทำงานแคช (cache worker) ทำงาน และติดตามผลการทำงานว่าในการทำงานนั้นให้ประสิทธิภาพเท่าใดบ้าง โดยมีการบอกถึง

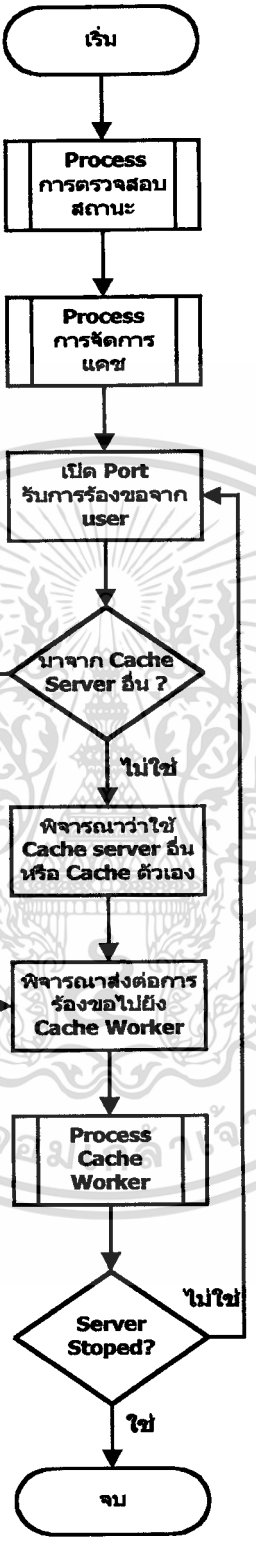
เวลาการตอบสนอง , อัตราที่ใช้แคช และส่วนของข้อมูลที่ส่งระหว่างเครื่องให้บริการแคชด้วยกัน เพื่อติดตามผล และประเมินประสิทธิภาพได้



รูปที่ 3.5: ส่วนของการติดตามผลของระบบ

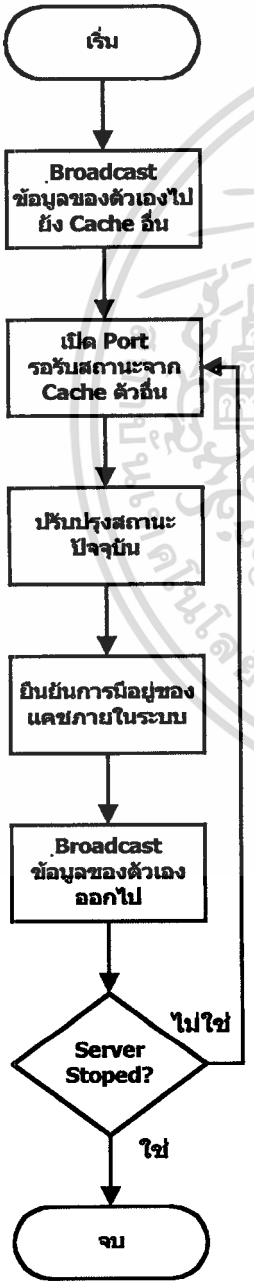
3.5 ส่วนของตัวจัดการ (Manager Process)

โดยส่วนของโปรแกรมตัวจัดการเองนั้น แสดงดังรูปที่ 3.6 ซึ่งเป็น Flowchart แสดงการทำงานโดยรวม ในส่วนของตัวจัดการนี้จะเริ่มจากการสร้างโพรเซสการทำงาน 2 ส่วน ซึ่งทำงานเป็นอิสระต่อกันในลักษณะมัลติโปรเซส (multiprocess หรือ multithread) คือ ส่วนของโพรเซสในการตรวจสอบสถานะและ ส่วนของโพรเซสในการบริหารแคช

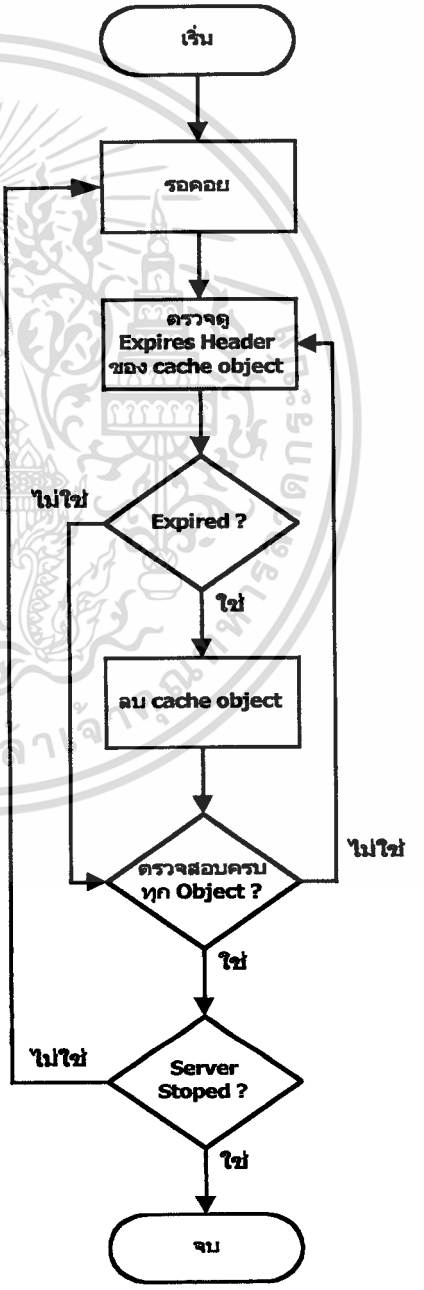


รูปที่ 3.6: ส่วนของตัวจัดการ

จากนั้น ในจะทำการเปิดเส้นทางการรับข้อมูลหรือพอร์ต (port) เพื่อรอรับการร้องขอจากผู้ใช้หรือแคชตัวอื่น เมื่อได้รับการร้องขอแล้วจะมีการพิจารณาว่าการร้องขอนั้นมาจากเครื่องให้บริการแคชตัวอื่นหรือมาจากผู้ใช้เอง เพื่อประโยชน์ในการพิจารณาส่งต่อการร้องขอในส่วนของระบบแคชไม่ให้มีการส่งการร้องขอกลับไปยังเครื่องให้บริการแคชอีก เพราะอาจทำให้การร้องขอวนอยู่เพียงในเครือข่ายเท่านั้น จากนั้นพิจารณาว่าควรจะส่งต่อการร้องขอไปยังโพรเซสใดของส่วนที่ทำการแคช (cache worker) เมื่อส่งต่อการร้องขอไปยังส่วนที่ทำการแคชแล้ว จะทำการตรวจสอบว่าเครื่องให้บริการหยุดทำงานหรือไม่ ถ้าไม่หยุดจะกลับไปรอรับการร้องขอจากผู้ใช้คนอื่นต่อไป



รูปที่ 3.7: โพรเซสในการตรวจสอบสถานะ



รูปที่ 3.8: โพรเซสในการบริหารแคช

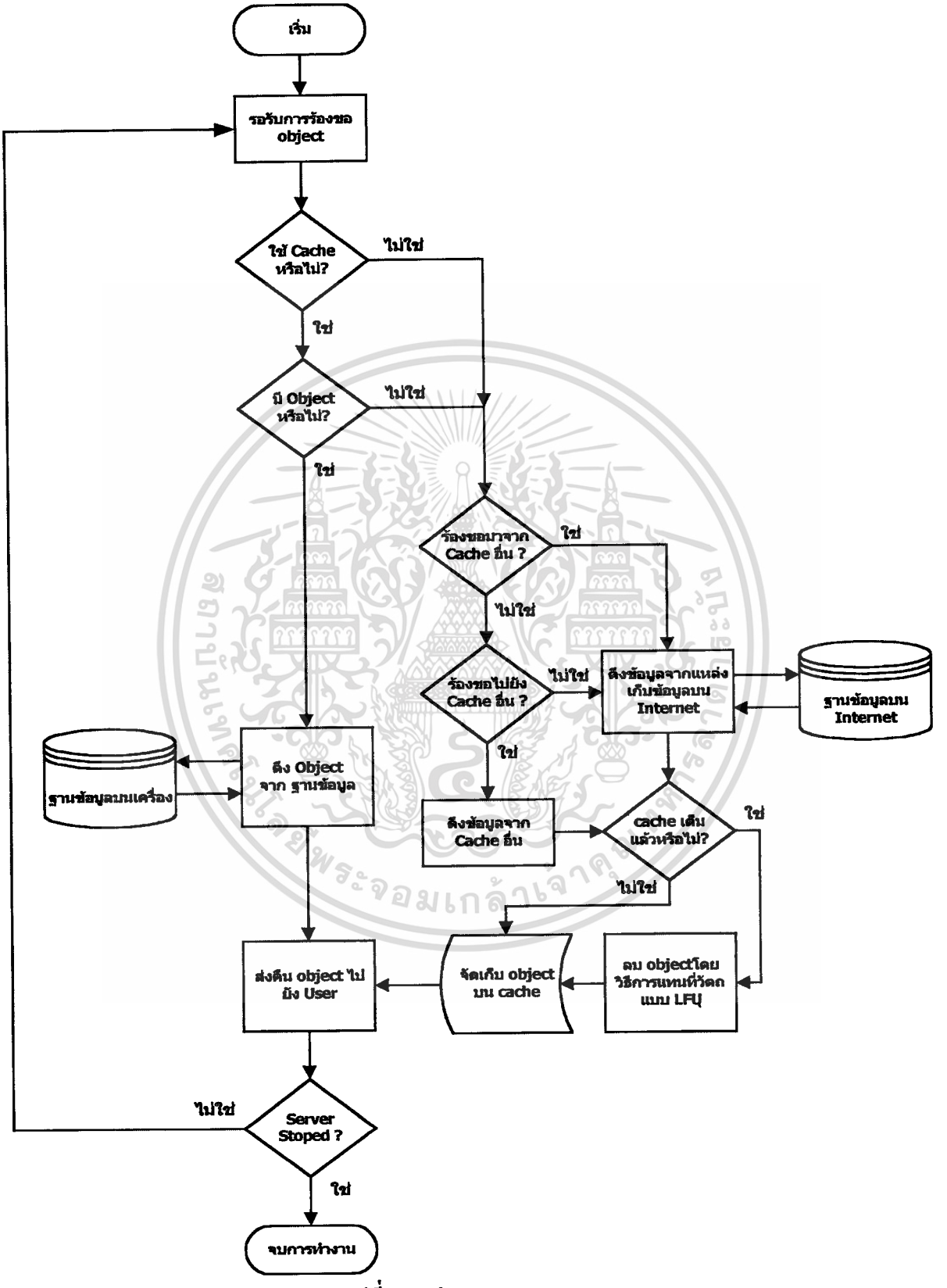
สำหรับส่วนที่ใช้ตรวจสอบสถานะการเชื่อมต่อดังรูปที่ 3.7 นั้น เมื่อเรียกใช้ครั้งแรก จะทำการกระจายข้อมูล (broadcast) ของตัวเองไปยังแคชตัวอื่น เพื่อให้ตัวอื่นรับรู้ว่ามีแคชเพิ่มเข้ามาในระบบแล้ว จากนั้นทำการรอรับข้อมูลที่แคชเครื่องอื่นกระจายออกมาเช่นกัน แล้วทำการปรับปรุงสถานะปัจจุบัน โดยจะมีการยืนยันการเชื่อมต่อกับแคชตัวอื่นตลอดเวลา และตรวจสอบว่าเครื่องให้บริการหยุดทำงานหรือไม่ ถ้ายังก็กลับไปรอรับสถานะต่อไป ส่วนโพรเซสในการให้บริการแคชนั้นจะทำหน้าที่เพียงตรวจสอบว่าข้อมูลในแคชนั้นหมดอายุหรือไม่ ถ้าหมดอายุก็จะลบทิ้ง โดยกำหนดช่วงระยะเวลาตรวจสอบระยะหนึ่ง ดังรูปที่ 3.8

3.6 ส่วนของระบบแคช

โดยส่วนของระบบแคชจะมีส่วนผู้ทำงาน (cache workers) หลายๆตัวทำงานพร้อมกันในลักษณะมัลติโพรเซส ซึ่งขั้นตอนการทำงานของแต่ผู้ทำงานแต่ละตัว จะเริ่มจากการได้รับการส่งต่อการร้องขอจากผู้ร้องขอจนกระทั่งส่งข้อมูลที่ผู้ร้องขอต้องการกลับไปดังรูปที่ 3.9

เมื่อผู้ทำงาน (worker) ได้รับการร้องขอจากส่วนของตัวจัดการ (manager) แล้ว จะทำการตรวจสอบเฮดเดอร์ว่าต้องการไม่ใช้แคชหรือไม่ ถ้าใช้จะทำการตรวจดูว่าภายในฐานข้อมูลในแคชมีข้อมูล (object) ที่เราต้องการอยู่หรือไม่ ถ้ามีจะทำการดึงข้อมูลส่งกลับไปให้ผู้ใช้ต่อไป แต่ถ้าข้อมูลนั้นไม่มีอยู่ จะทำการร้องขอข้อมูลจากแหล่งเก็บข้อมูลบนอินเทอร์เน็ตโดยตรงในกรณีที่เราพิจารณาข้างต้นแล้วว่าเป็นการร้องขอที่มาจากแคชตัวอื่น แต่ถ้าไม่ใช่ จะมีกระบวนการพิจารณาว่าควรจะร้องขอข้อมูลจากอินเทอร์เน็ตโดยตรง หรือส่งการร้องไปอีกเส้นทางหนึ่งโดยผ่านแคชตัวอื่น ถ้าต้องการร้องขอข้อมูลผ่านแคชตัวอื่น จะส่งไปยังส่วนที่ทำการติดต่อกับแคชตัวอื่น ซึ่งส่วนนี้จะติดต่อไปยังแคชอื่นโดยพิจารณาเหมือนตัวเองเป็นผู้ใช้คนหนึ่ง ส่วนแคชที่ได้รับการร้องขอก็จะทำกระบวนที่ได้กล่าวมาแล้วนี้เช่นเดิม เมื่อข้อมูลถูกดึงกลับมาแล้ว จะต้องทำการพิจารณาแหล่งเก็บข้อมูลว่ามีเนื้อที่พอที่จะเก็บหรือไม่ ถ้าไม่เพียงพอต้องทำการลบข้อมูลส่วนที่ไม่จำเป็นทิ้ง ด้วยกระบวนการแทนที่วัตถุ ในโครงการนี้จะใช้วิธี Least-Frequently-Used (LFU) เมื่อทำการจับเก็บข้อมูล ก็จะส่งต่อไปให้บริการผู้ใช้ต่อไป

เมื่อทำการเลือกเส้นทางว่าจะส่งการร้องขอต่อไปยังเครื่องให้บริการอื่นนั้น ตัวเครื่องให้บริการแคชเองจะเพิ่มเฮดเดอร์พิเศษอันหนึ่งเข้าไป ได้แก่ เฮดเดอร์ *CacheServerApp/1.0* ซึ่งเฮดเดอร์นี้จะถูกเพิ่มเข้าไปที่ต้นการร้องขอที่จะถูกส่งไป การเพิ่มเฮดเดอร์นี้เข้าไป เพื่อให้เครื่องให้บริการแคชเครื่องอื่นสามารถเฮอะแยะได้ว่าเป็นการร้องขอมาจากแคชหรือมาจากผู้ใช้โดยตรง เนื่องจากถ้าเครื่องให้บริการไม่รับรู้ อาจทำให้การร้องขอถูกส่งวนไปวนมาภายในเครือข่ายก็เป็นได้



รูปที่ 3.9: ส่วนของระบบแคช

3.7 การจัดเก็บแฟ้มข้อมูลและส่วนที่เกี่ยวข้อง

ในระบบแคชที่ได้พัฒนาขึ้น จะมีส่วนที่เกี่ยวข้องกับการจัดเก็บแฟ้มข้อมูลอยู่เพียงสองส่วนเท่านั้น ส่วนแรกเป็นส่วนที่สำคัญมากในระบบของเรา เนื่องจากว่าเป็นเป้าหมายของระบบแคชที่จะต้องมีการจัดเก็บข้อมูลไว้ เพื่อให้บริการผู้ใช้เมื่อผู้ใช้ร้องขอข้อมูลเดิมนั้นอีกซึ่งเราจะจัดเก็บข้อมูลดังกล่าวในลักษณะของไฟล์ โดยที่ไฟล์ที่จัดเก็บจะเก็บในลักษณะไม่มีโครงสร้าง หรือเป็นรูปแบบแบน (flat file) ที่จัดเก็บลักษณะนี้ เนื่องจากยังอยู่ในช่วงเริ่มของการพัฒนาโครงการอีกทั้งระบบแคชให้บริการกับผู้ใช้จำนวนน้อย จึงยังไม่จำเป็นต้องมีโครงสร้างในการจัดเก็บฐานข้อมูลที่ซับซ้อน เราจะจัดเก็บข้อมูลแบบตรงไปตรงมา คือ เมื่อร้องขอ URL ใด ก็จัดเก็บในลักษณะ URL นั้นเพียงแต่ว่าจะต้องมีการเปลี่ยนรูปแบบให้สามารถสร้างเป็นชื่อไฟล์ที่สามารถจัดเก็บได้ โดยทำดังนี้ แปลง " / " เป็น " ` ", แปลง " : " เป็น " ! ", แปลง " ? " เป็น " ^ " และแปลง " * " เป็น " # " ยกตัวอย่างเช่น

http://www.it.kmitl.ac.th/ $\Rightarrow$ http!``www.it.kmitl.ac.th`

จากนั้นทำการจัดเก็บไฟล์ในชื่อดังกล่าว ซึ่งเมื่อผู้ใช้ทำการร้องขอ URL เดิม จะถูกแปลงในรูปแบบเช่นนี้ และทำการเปรียบเทียบว่ามีชื่อตรงกันหรือไม่ ถ้าตรงก็จะดึงข้อมูลที่จัดเก็บภายในชื่อดังกล่าวนี้ ส่งไปให้ผู้ร้องขอได้เลย

การจัดเก็บแฟ้มข้อมูลอีกส่วนหนึ่ง เป็นการจัดเก็บค่าปรับแต่ง ซึ่งจัดเก็บเป็นไฟล์เช่นเดียวกัน โดยค่าการปรับแต่งนี้จะถูกบันทึกไว้และเมื่อมีการเริ่มต้นโปรแกรมใหม่ ค่านี้ก็จะถูกดึงออกมาจากแฟ้มข้อมูล โดยจับเก็บไว้ในชื่อ status.txt ซึ่งโปรแกรมจะทำการตรวจสอบว่ามีชื่อดังกล่าวอยู่หรือไม่ ถ้าไม่มีจะใช้ค่า default ที่โปรแกรมได้ตั้งไว้

บทที่ 4
การพัฒนาระบบงาน

ที่ผ่านได้ เราได้ออกแบบระบบแคชโดยมีส่วนประกอบต่างๆดังที่ได้แสดงไว้ก่อนหน้านี้แล้ว ในบทที่จะนำส่วนที่ได้ออกแบบไว้มาทำการสร้างระบบแคชจริง ซึ่งในขั้นต้นจะกล่าวถึงเครื่องมือที่ใช้ และได้กล่าวถึงระบบที่ได้พัฒนาขึ้นภายหลัง ดังมีรายละเอียดดังนี้

4.1 เครื่องมือที่ใช้ (Tools)

เราจะทำการเขียนโปรแกรมเครื่องให้บริการแคชขึ้นมาโดยใช้โครงสร้างทางภาษาที่เป็นแบบ object-oriented program ได้แก่ ภาษาจาวา ซึ่งการเขียนโค้ดด้วยภาษาจาวาของเราใช้โปรแกรม Jbuilder ในการคอมไพล์และดีบัก ที่เลือกใช้ Jbuilder ในการออกแบบมีเหตุผลหลายประการ ดังจะได้กล่าวถึงในรายละเอียดต่อไปนี้

- สามารถออกแบบส่วนที่ติดต่อกับผู้ใช้ (User Interface) ได้ง่าย เนื่องจากมีเราไม่จำเป็นต้องเขียนโค้ดในส่วนนี้เอง ทางโปรแกรมมีรูปแบบซึ่งสามารถดึงมาใช้ได้เลย
- มี library ที่เกี่ยวข้องกับส่วนที่เราต้องการให้เลือกมากมาย สะดวกแก่การใช้ไม่ต้องสร้างฟังก์ชันเอง ซึ่งจะทำให้เสียเวลาในการพัฒนาเป็นอย่างมาก
- สามารถสร้างการทำงานแบบมัลติเธรด (multithread) ได้ง่ายเพราะมี class รองรับอยู่แล้ว ซึ่งส่วนนี้จำเป็นสำหรับการสร้างการร้องขอในหลายช่องทางเพื่อสามารถรับข้อมูลได้พร้อมๆกัน จึงเกิดความรวดเร็วในการร้องขอข้อมูลมากยิ่งขึ้น
- โครงสร้างทางภาษาเป็นแบบ object-oriented ทำให้การออกแบบซ่อนความซับซ้อนไว้ภายใน จึงสามารถตรวจสอบโปรแกรมและหาข้อผิดพลาดได้ง่าย
- ตัว Jbuilder สามารถตรวจสอบความผิดพลาดทางไวยากรณ์ พร้อมทั้งแสดงผลและส่วนที่เกิดความผิดพลาด ทำให้เกิดความรวดเร็วในการแก้ไขส่วนที่ผิดพลาด

4.2 อุปกรณ์ที่ใช้ในการพัฒนาระบบงาน

จากที่กล่าวไปในบทก่อนหน้านี้แล้วว่าระบบเครือข่ายที่ใช้นั้น เป็นระบบอินเทอร์เน็ตผ่านดาวเทียม ซึ่งเราได้ออกแบบระบบแคชในลักษณะที่ทำงานร่วมกัน เพื่อรองรับระบบอินเทอร์เน็ตผ่านดาวเทียม และเพื่อให้สื่อสารกับเครือข่ายที่มีอยู่ได้ด้วย จึงมีรายละเอียดของอุปกรณ์ดังนี้

4.2.1 เครื่องที่ให้บริการระบบแคชเครื่องที่ 1

- CPU: Pentium III 700 MHz
- RAM: 128 Mbyte
- Harddisk: 20 GB
- NIC: Realtek RTL8139(A) PCI Fast Ethernet Adapter
- อุปกรณ์เชื่อมต่อกับ NetTurbo
- OS: Windows2000 Advance Server
- Host: Ligay.mvlab.reccit.kmitl.ac.th

4.2.2 เครื่องที่ให้บริการระบบแคชเครื่องที่ 2

- CPU: Pentium III 700 MHz
- RAM: 256 Mbyte
- Harddisk: 20 GB
- NIC: Realtek RTL8139(A) PCI Fast Ethernet Adapter
- OS: Windows2000 Advance Server
- Host: Khon.mvlab.reccit.kmitl.ac.th

4.2.3 อุปกรณ์ที่เกี่ยวกับอินเทอร์เน็ตผ่านดาวเทียมทิศทางเดียว (NetTurbo)

- จานรับสัญญาณดาวเทียม (only antenna (TVRO) Ku-band) ขนาด 60-120cm
- BroadLogic ABA2030 Adaper (NetTurbo PC card)
- Diamond Supra Express Modem 56 Kbps

4.2.4 ระบบเครือข่าย

- LAN ของสำนักวิจัยการสื่อสารและเทคโนโลยี (ReCCIT) ภายในห้องปฏิบัติการมัลติมีเดีย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
- NetTurbo อินเทอร์เน็ตผ่านดาวเทียมความเร็ว 128 MB ให้บริการโดย CS Internet (ISP)

4.2.5 เครื่องมือที่ใช้ในการพัฒนาระบบ (Development Tools)

- Borland Jbuilder 4 Enterprise

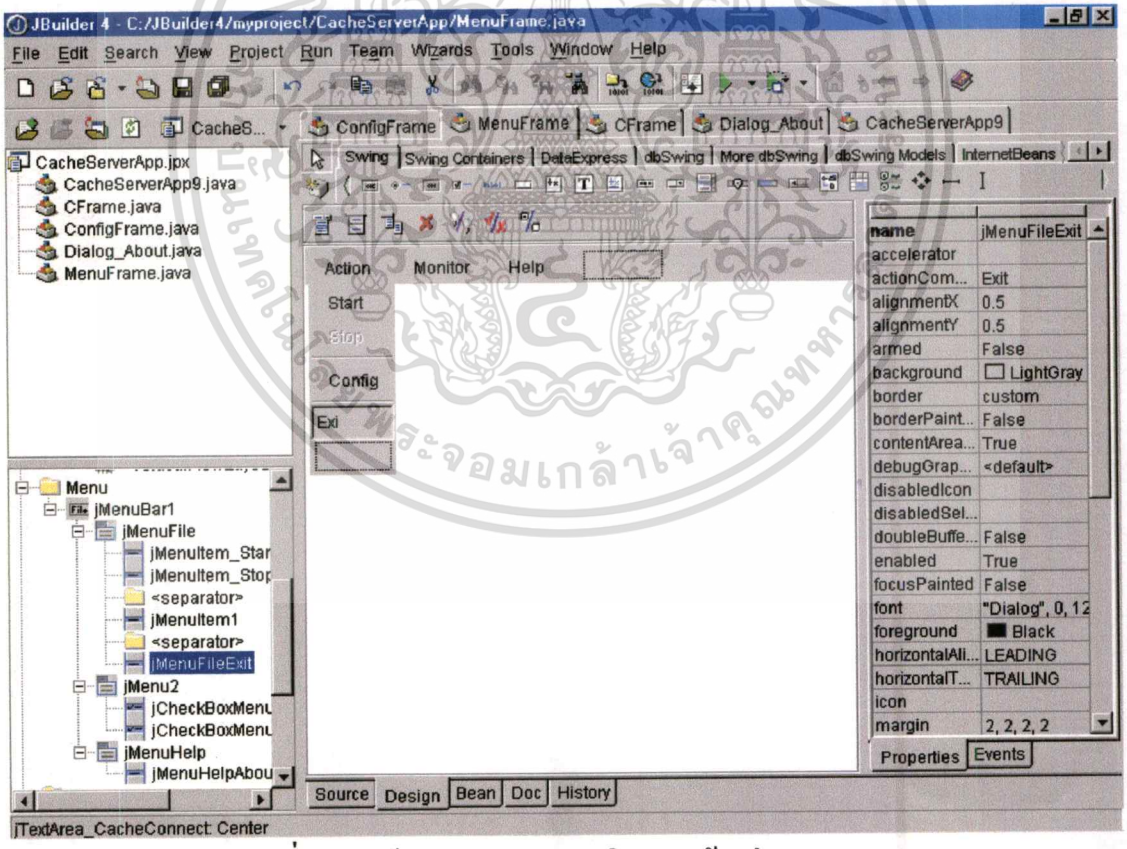
4.3 การสร้างส่วนติดต่อกับผู้ใช้ (user interface)

ในการสร้างส่วนที่ใช้ติดต่อกับผู้ใช้โดยใช้ Jbuilder นั้นสามารถทำได้ง่าย เนื่องจากเครื่องมือดังกล่าว ได้อำนวยความสะดวกในการสร้างรูปแบบของหน้าจอ ปุ่ม และส่วนต่างๆ โดยพิจารณาเป็นวัตถุ (object) หนึ่งที่ Jbuilder สร้างไว้ให้ มาวาง ณ ตำแหน่งที่เรากำหนดและสามารถปรับเปลี่ยนขนาดได้ตามต้องการ รวมถึงการกำหนดรูปแบบของการจัดวางและเหตุการณ์ที่เชื่อมโยงกับวัตถุนั้น

ในการสร้างส่วนการติดต่อกับผู้ใช้นั้นแบ่งออกเป็น 2 ส่วนหลัก คือ

4.3.1 ส่วนของเมนู

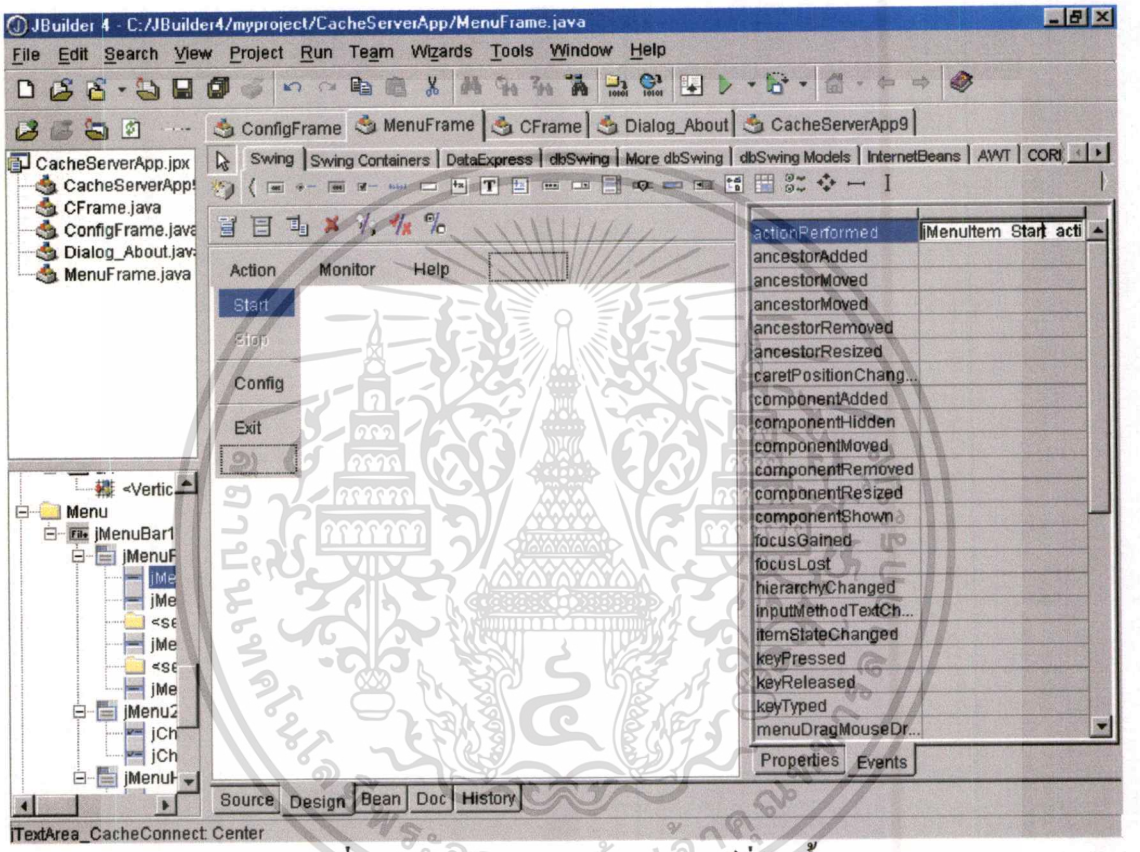
โปรแกรม Jbuilder ได้อำนวยความสะดวกในการสร้างเมนูเช่นกัน โดยในขั้นแรกจะต้องเลือกแถบข้างเครื่องมือให้อยู่ในโหมดของการออกแบบ (design) แล้วทำการเลือกที่เมนู (menu) ในส่วนของหน้าต่าง structor ทางด้านซ้ายล่างของหน้าจอดังรูปที่ 4.1



รูปที่ 4.1: หน้าต่างของ Jbuilder ในการสร้างส่วนของเมนู

ในการเพิ่มข้อมูลเข้าไปในเมนูนั้น สามารถทำได้โดยตรงโดยทำการคลิกเมาท์ไปยังส่วนของเมนูหลักที่ต้องการเพิ่ม แล้วพิมพ์ชื่อของเมนูย่อยเข้าไปโดยตรง Jbuilder จะทำการสร้างส่วนที่

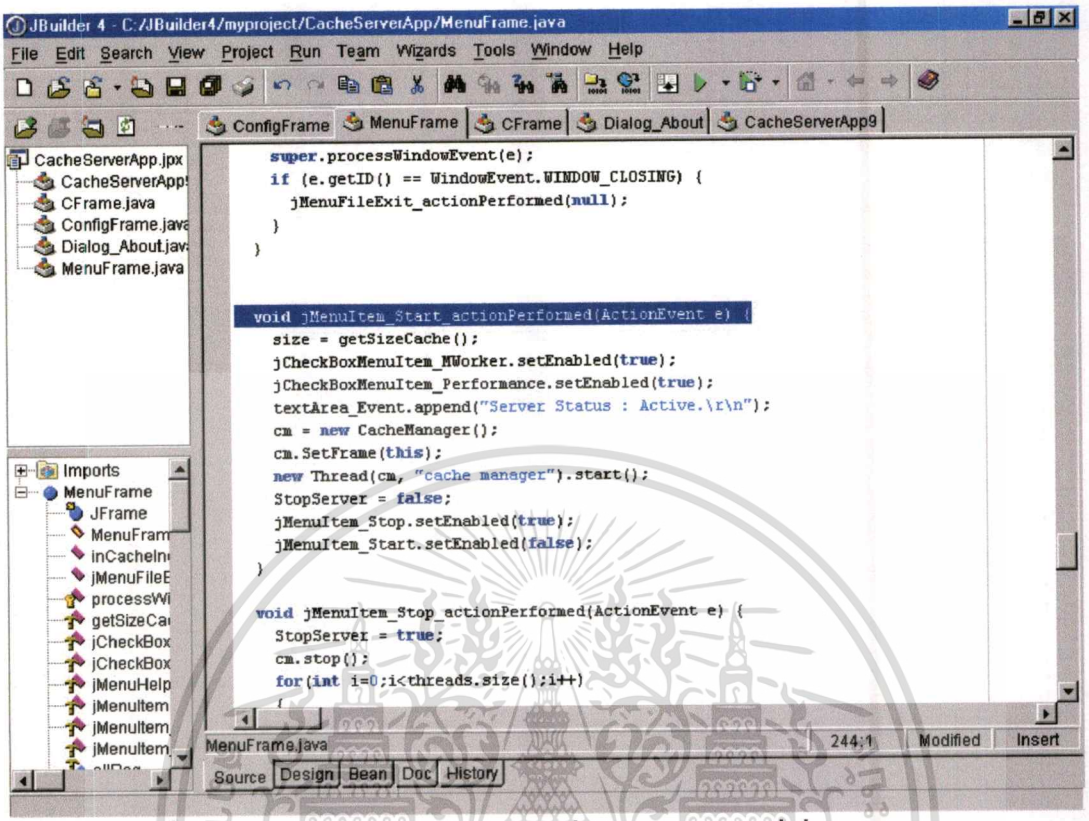
เกี่ยวข้องในการสร้างเมนูนั้นให้โดยอัตโนมัติ จากนั้นเมื่อต้องการเชื่อมโยงเมนูย่อยกับเหตุการณ์ที่กระทำกับเมนูนั้น วิธีการหนึ่งคือเลือกหน้าต่างด้านขวามือสุด ซึ่งสามารถเลือก Properties กับ Events ได้ให้ทำการเลือก Event ที่ต้องการ เช่น รูปที่ 4.2 ทำการเลือก actionPerformed event โดยเมื่อเกิดเหตุการณ์ที่มีการเลือกเมนูย่อยนั้น จะกระโดดไปส่วนของโปรแกรมนั้นๆ



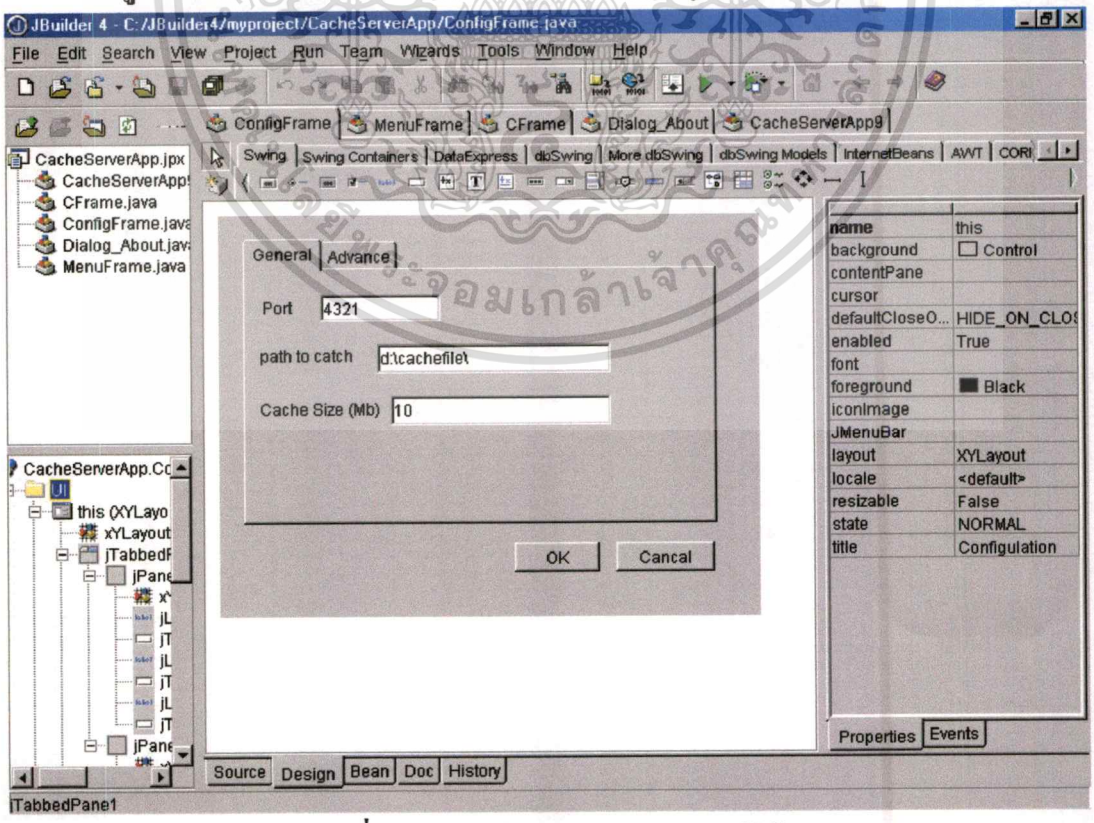
รูปที่ 4.2: การเลือกจัดการกับเหตุการณ์ที่เกิดขึ้น

เมื่อทำการเลือกเหตุการณ์ดังรูปแล้ว Jbuilder จะเพิ่มฟังก์ชั่นที่เชื่อมโยงกับเหตุการณ์นั้นให้แล้วกระโดดไปยังส่วนของโหมด Source ให้เพื่อเพิ่มโคดที่เราต้องการเมื่อเกิดเหตุการณ์ดังกล่างดังรูปที่ 4.3

สำหรับส่วนโคดในการเชื่อมโยงเหตุการณ์ต่างๆกับเมนูในแต่ละส่วนที่เราออกแบบไว้นั้นจะแสดงไว้โดยละเอียดในส่วนภาคผนวกต่อไป



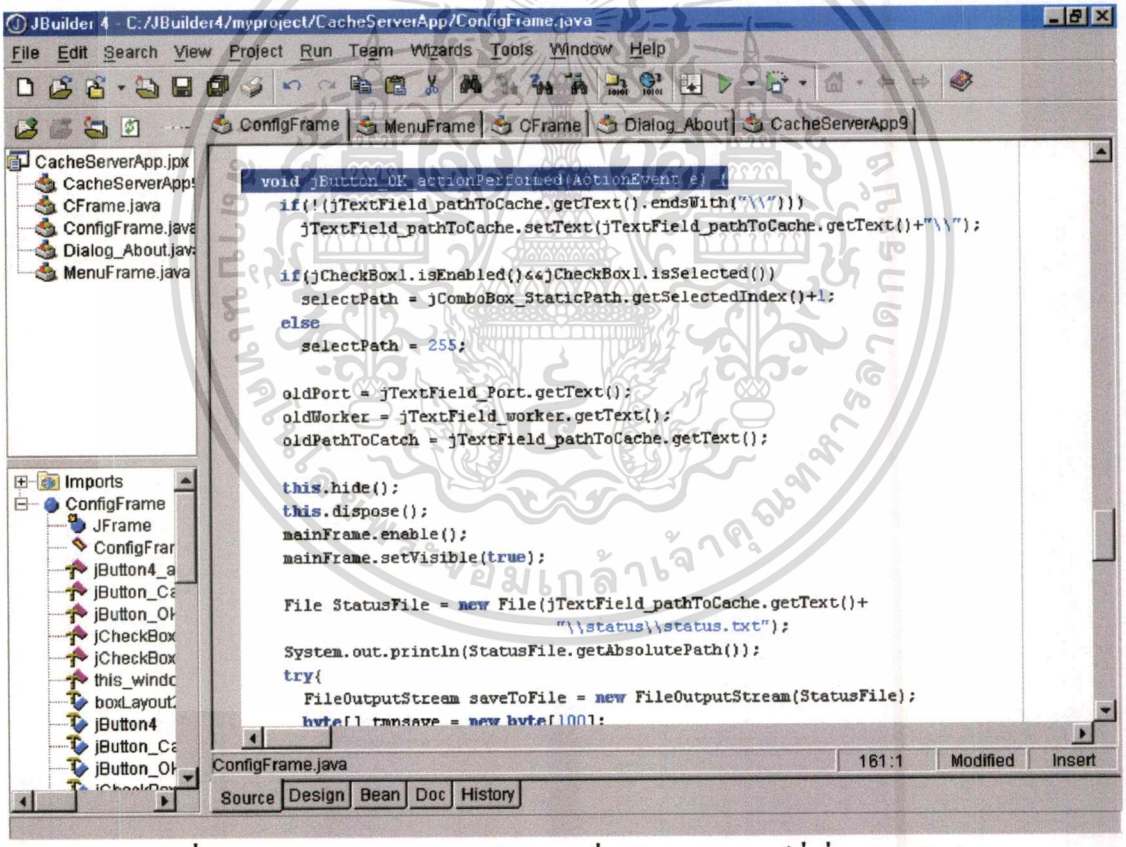
รูปที่ 4.3: แสดงส่วนของการสร้างโคดเมื่อเลือกเหตุการณ์ที่เชื่อมโยงกับเมนู



รูปที่ 4.4: การสร้างส่วนการติดต่อกับผู้ใช้

4.3.2 ส่วนของปุ่ม(button), พื้นที่ตัวอักษร(textArea), ป้าย(label) และวัตถุอื่นๆ

ในการสร้างส่วนของหน้าจอหลักที่ไม่ใช่เมนูนั้น จะทำการเลือกโหมด Design เช่นกัน ส่วนหน้าต่าง structor ให้เลือกที่ UI จะปรากฏเป็นกรอบสี่เหลี่ยมว่างๆไว้ จากนั้นทำการเลือกวัตถุที่ต้องการนำมาวางไว้บนหน้าต่างจากแถบวัตถุเหนือหน้าต่างที่ใช้สร้างส่วนติดต่อกับผู้ใช้ ไม่ว่าจะเป็นปุ่ม พื้นที่ตัวอักษร เพลน ป้าย หรือวัตถุอื่น จัดวางตามต้องการ โดยสามารถเลือกรูปแบบที่จัดวางได้จากส่วนของ Properties หน้าต่างด้านขวามือ ให้เลือกที่ layout ดังรูปที่ 4.4 ซึ่งเมื่อทำการวางวัตถุใดบนเพลนแล้ว Jbuilder จะทำการสร้างโคดไว้ในส่วนของ Source ไว้เช่นกัน โดยการเชื่อมโยงกับเหตุการณ์ต่างๆสามารถเลือกได้กรอบด้านขวามือในส่วนของ Events เช่นกัน เมื่อเลือกเหตุการณ์แล้วจะเข้าส่วนของ Source ดังรูปที่ 4.5



รูปที่ 4.5: แสดงส่วนของการสร้างโคดเมื่อเลือกเหตุการณ์ที่เชื่อมโยงกับปุ่ม

จะเห็นว่าการสร้างส่วนของการติดต่อกับผู้ใช้โดยใช้ Jbuilder สามารถทำได้ง่ายมาก โดยไม่จำเป็นต้องเขียนโคดในส่วนของการสร้างหน้าจอต่างๆเลย โดยเขียนเพียงส่วนที่ต้องการกระทำเมื่อเกิดเหตุการณ์ใดๆกับวัตถุนั้นๆ

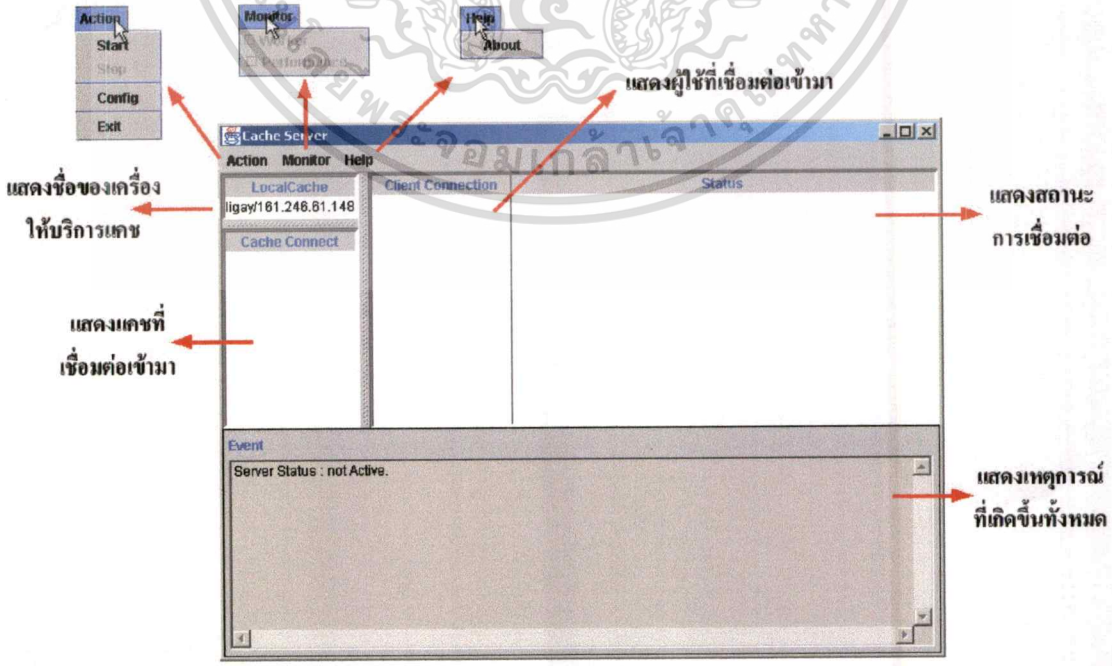
4.4 การพัฒนาระบบแคช (Development to Cache System)

บทที่แล้วเราได้ออกแบบระบบแคชที่ทำงานร่วมกัน (cooperative cache) แล้ว ในบทที่จะนำส่วนที่ได้ออกแบบไว้มาสร้างเป็นระบบงานจริง เราได้ใช้เครื่องมือในการพัฒนาระบบคือ Java Jbuilder4 ซึ่งใช้โครงสร้างของภาษา Java แต่เนื่องจากว่าเราไม่สามารถจะคอมไฟล์ภาษา Java เป็นไฟล์ที่สามารถทำงานได้เลย (Execute File) ในที่เราจึงทำการ run โปรแกรมผ่านตัว Jbuilder โดยตรง เพราะโปรแกรมยังอยู่ในขั้นตอนการพัฒนา เราสามารถแก้ไข้ และปรับเปลี่ยนโปรแกรมได้ง่ายเมื่อเกิดข้อผิดพลาดขึ้น

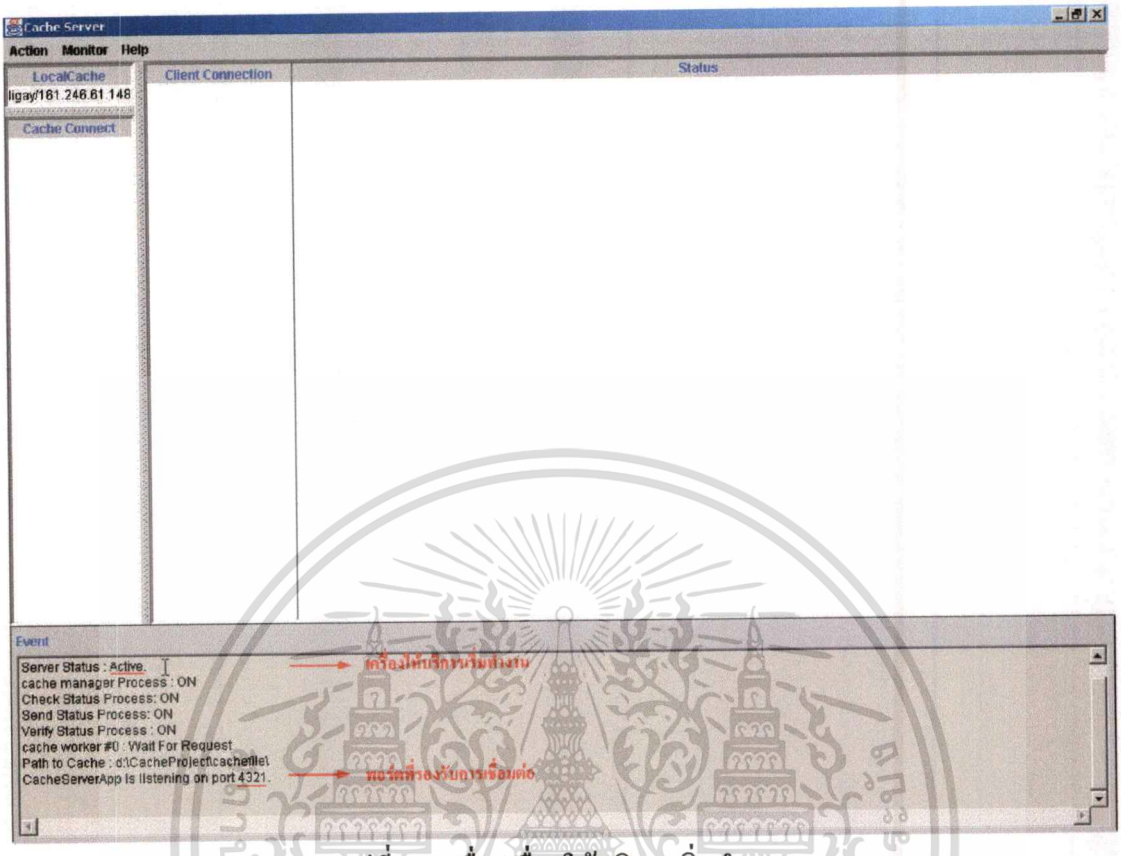
เราสามารถติดต่อกับระบบแคชโดยผ่านที่ส่วนที่ติดต่อกับผู้ใช้ (User Interface) โดยโปรแกรมของเราจะมีหน้าจอหลักดังรูปที่ 4.6 จากรูปจะแบ่งหน้าจอออกเป็น 5 ส่วนหลักๆ คือ ส่วนของเมนูบาร์, หน้าต่าง Locol Cache, หน้าต่าง Client Connection, หน้าต่าง Cache Conneciton และส่วนล่างของกรอบคือ หน้าต่าง Event โดยหน้าต่าง Event นั้นจะแสดงเหตุการณ์ที่เกิดขึ้นทั้งหมดกับระบบแคช เพื่อใช้ดูว่าขณะนั้นเครื่องให้บริการมีสถานะการทำงานอย่างไรบ้าง

ส่วนของเมนูบาร์นั้นจะแบ่งย่อยเป็น 3 เมนูหลัก คือ

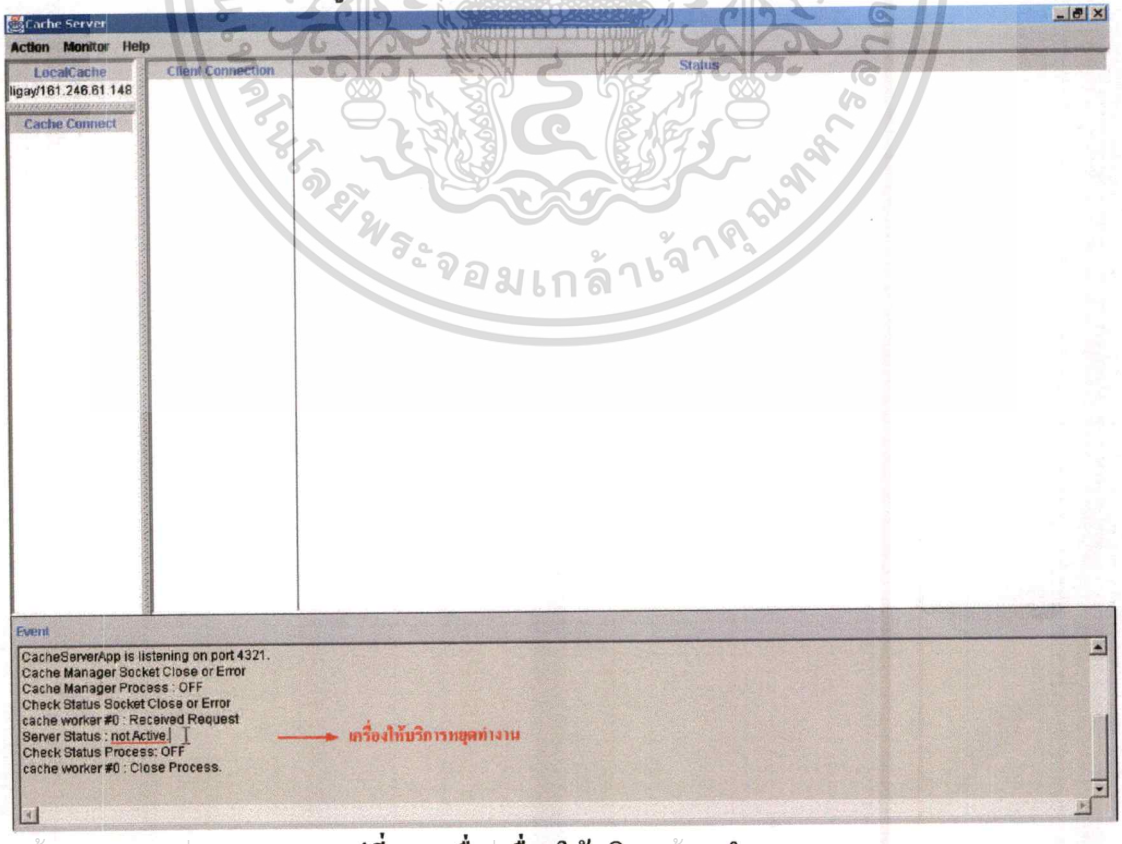
1. Action
2. Monitor
3. Help



รูปที่ 4.6: หน้าจอหลักของโปรแกรม



รูปที่ 4.7: เมื่อเครื่องให้บริการเริ่มทำงาน



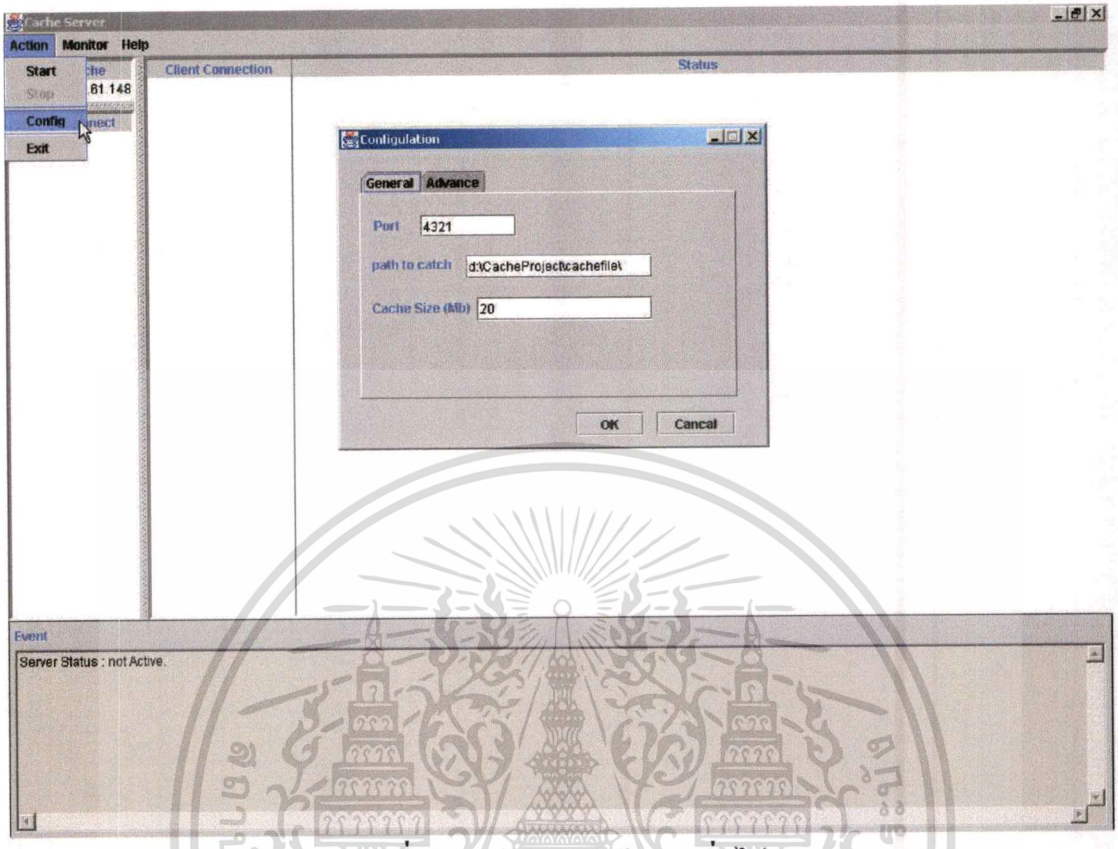
รูปที่ 4.8: เมื่อเครื่องให้บริการหยุดทำงาน

โดยในส่วนของ Action เมนูนั้นยังแบ่งออกเป็นส่วนย่อยๆ คือ

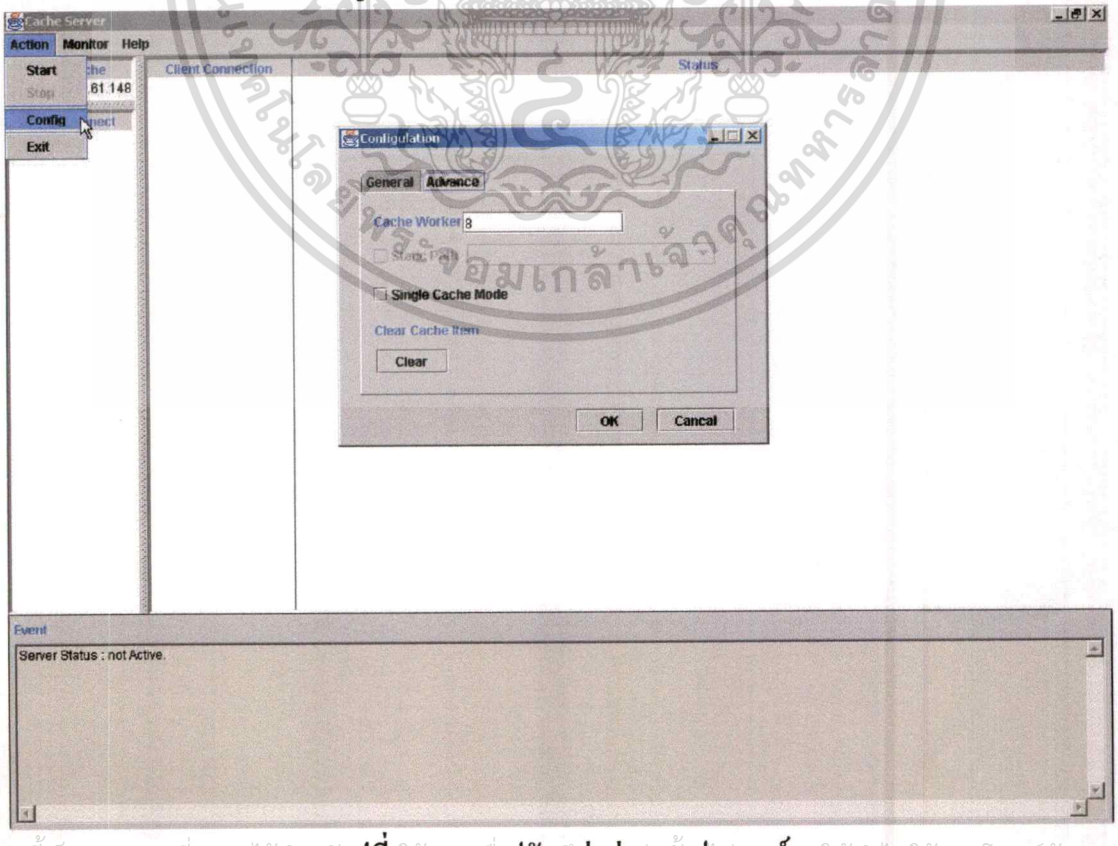
- Start Server
- Stop Server
- Configuration
- Exit

สองตัวแรกเป็นส่วนที่ใช้ในการสั่งงานให้เครื่องให้บริการทำงานและหยุดทำงานดังรูปที่ 4.7 และ 4.8 โดยที่เมื่อมีการสั่งให้เครื่องให้บริการทำงานนั้น ตรงหน้าต่าง Event จะแสดงสถานะในขณะนั้นว่าเครื่องให้บริการทำงาน มีโพรเซสใดเริ่มทำงานบ้าง ใช้พอร์ตใดในการเปิดรอรับการร้องขอ ใช้เส้นทางใดเก็บข้อมูลที่ถูกแคช รวมถึงมีการสร้างผู้ทำงานจำนวนเท่าใด เมื่อสั่งให้เครื่องให้บริการหยุดทำงานก็เช่นกัน ที่หน้าต่าง Event ก็จำแสดงสถานะว่าเครื่องให้บริการหยุดทำงาน มีโพรเซสใดบ้างที่หยุดทำงาน รวมทั้งถ้ามีความผิดพลาดเกิดขึ้น ก็จะแสดงไว้ที่หน้าจอ Event ด้วยเช่นกัน

ส่วนถัดไปเป็นส่วนที่ใช้ในการปรับแต่งค่าของระบบแคชเอง โดยส่วนนี้จะมีหน้าต่างย่อยขึ้นมาเพื่อให้เราสามารถปรับแต่งค่าได้อย่างสะดวก ซึ่งจะมีการปรับแต่งค่าเป็นสองส่วนคือ ส่วนทั่วไปกับส่วนประยุกต์ การปรับแต่งค่าแบบทั่วไปนั้น แสดงดังรูปที่ 4.9 โดยมีการปรับแต่งค่า 3 ค่า คือ พอร์ต (port) สำหรับรอรับการเชื่อมต่อจากผู้ใช้, เส้นทางที่ใช้ในการจัดเก็บข้อมูล (path to cache) และความจุของแคช (cache size) อีกส่วนหนึ่งคือการปรับแต่งค่าแบบประยุกต์ จะประกอบด้วยการปรับค่า ผู้ทำงาน (worker) ซึ่งจะเป็นการระบุว่าต้องการให้มีผู้ทำงานหรือจำนวนโพรเซสที่รองรับการร้องขอกี่เส้นทาง จำนวนที่ปรับแต่งนี้ขึ้นอยู่กับประสิทธิภาพของเครื่องและเครือข่าย, การระบุเส้นทางในการร้องขอ (static path) เมื่อเครื่องให้บริการแคชทำงานร่วมกันหลายตัว จะสามารถระบุได้ว่าต้องการให้การร้องขอถูกส่งต่อไปยังโหนดใด, การกำหนดให้เป็นระบบแคชเดี่ยว (single cache mode) ซึ่งจะระบุให้เครื่องให้บริการไม่ติดต่อกับแคชอื่น และส่วนสุดท้ายใช้ในการลบข้อมูลที่ถูกแคชทั้งหมด ดังรูปที่ 4.10



รูปที่ 4.9: การปรับแต่งค่าแบบทั่วไป

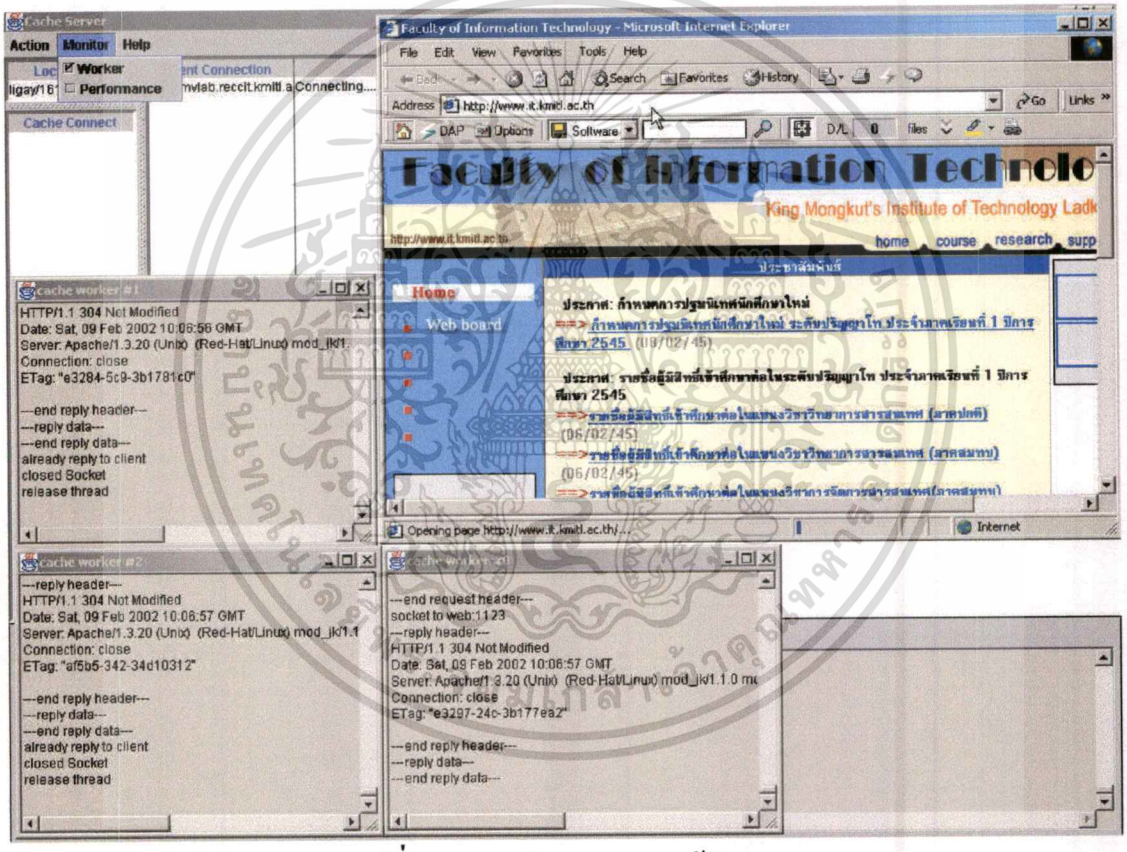


รูปที่ 4.10: การปรับแต่งค่าแบบประยุกต์

ส่วนต่อไปที่จะกล่าวถึงคือเมนู Monitor ซึ่งส่วนนี้ยังแบ่งออกเป็น 2 ส่วนย่อย คือ

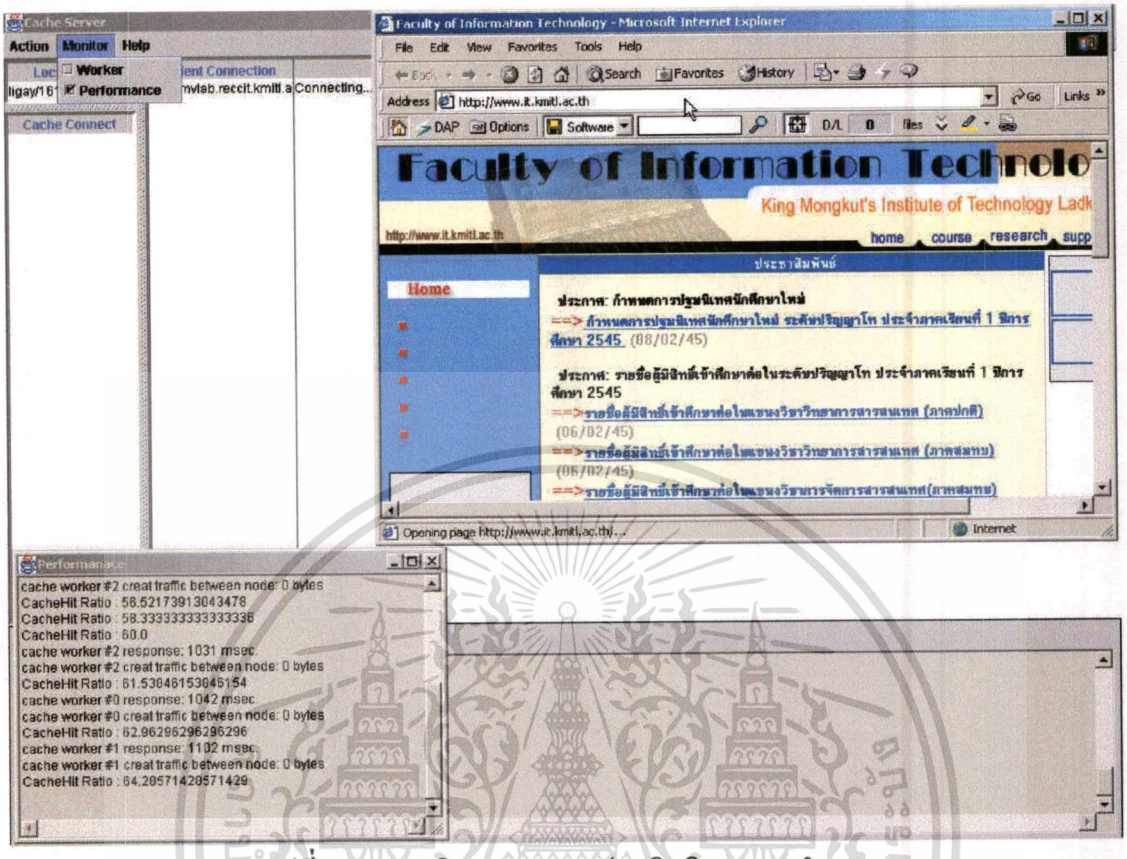
- Monitor worker
- Monitor performance

ส่วนการติดตามผลของผู้ทำงานนั้นจะแสดงโพรเซสการทำงานในขณะนั้นของผู้ทำงานว่ามีข้อมูลใดผ่านตัวผู้ทำงานอยู่บ้างดังรูปที่ 4.11 เมื่อเลือกที่เมนูย่อยผู้ทำงานแล้ว จะปรากฏหน้าต่างเล็กตามจำนวนของผู้ทำงานที่ได้เลือกไว้ แต่ละหน้าต่างจะเป็นอิสระต่อกัน สามารถเปิด-ปิด ปรับเปลี่ยนขนาด และย้ายไปมาได้



รูปที่ 4.11: การติดตามผลของผู้ทำงาน

การปรับแต่งส่วนของประสิทธิภาพนั้นมีลักษณะคล้ายส่วนของผู้ทำงาน คือ จะปรากฏหน้าต่างเล็กๆขึ้นมา แสดงสถานะการทำงานขณะนั้นว่ามีประสิทธิภาพเท่าใด โดยเราจะวัดประสิทธิภาพจากเวลาการตอบสนอง (response time) , อัตราการแคช (cache hit ratio) ซึ่งเป็นสัดส่วนการร้องขอที่เจอข้อมูลบนแคชกับการร้องขอทั้งหมด และแสดงถึงข้อมูลที่ส่งระหว่างแคชโหนดเอง ดังแสดงในรูปที่ 4.12 ส่วนอีกเมนูหลัก (Help) จะแสดงเพียงรายละเอียดของโครงการและส่วนที่เกี่ยวข้อง



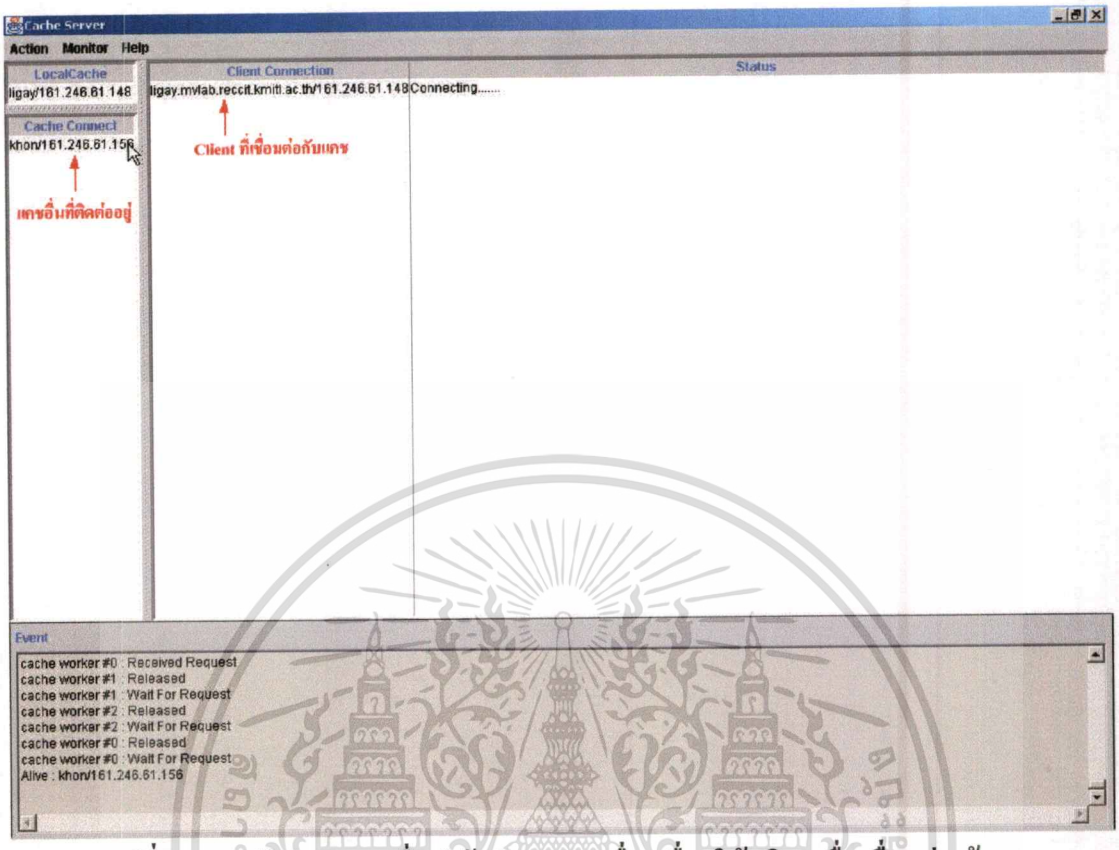
รูปที่ 4.12: การติดตามผลของประสิทธิภาพการทำงาน

4.5 การเชื่อมต่อระหว่างเครื่องให้บริการแคช

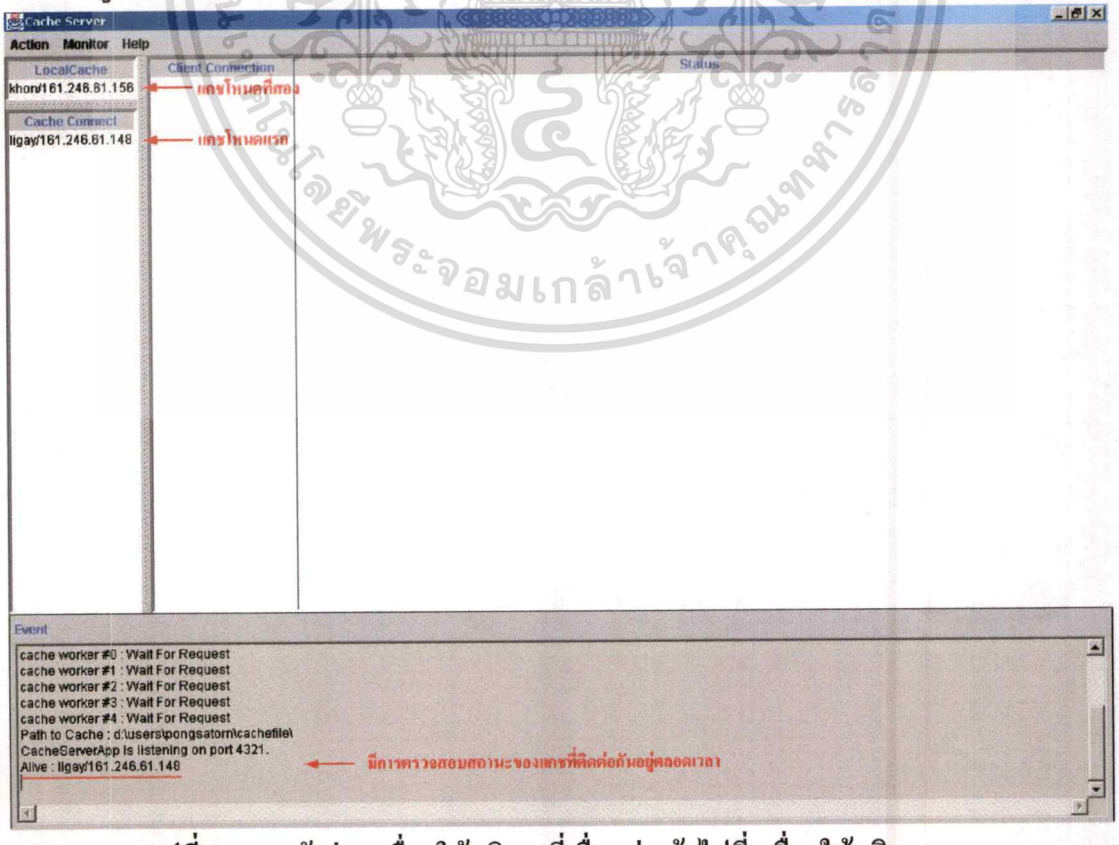
เมื่อมีผู้ใช้ร้องขอข้อมูลผ่านเครื่องให้บริการแคชหน้าต่าง Client Connection จะปรากฏชื่อของผู้ใช้ที่ติดต่อเข้ามาขณะนั้น รวมทั้งสถานะของผู้ใช้นั้นด้วย

เมื่อเครื่องให้บริการแคชทำงานพร้อมกันหลายตัว แต่ละตัวจะมีการส่งข้อมูลถึงกัน เพื่อใช้ตรวจสอบว่ามีแคชใดอยู่ภายในเครือข่ายบ้าง ทำให้สามารถส่งต่อการร้องขอไปยังเครื่องให้บริการแคชอื่นได้ โดยจะมีการทำงานเป็นโพรเซสย่อยๆ 3 ส่วน แยกจากกันและเป็นอิสระต่อกัน ส่วนแรกจะทำการ Broadcast ข้อมูลออกไปทุกๆช่วงเวลาที่กำหนด เพื่อให้เครื่องให้บริการอื่นรับรู้สถานะ โดยมีส่วนที่ใช้รับสถานะแยกออกมาอีกส่วน ซึ่งมีการเปิดพอร์ตรอรับการติดต่อไว้ ส่วนสุดท้ายจะเป็นการตรวจสอบสถานะว่าเครื่องให้บริการที่เชื่อมต่ออยู่นั้น ยังอีกหรือไม่ ถ้าไม่อยู่แล้วจะตัดการติดต่อกับเครื่องให้บริการนั้น

รูปที่ 4.13 แสดงการติดต่อกับเครื่องให้บริการเมื่อมีผู้ใช้ร้องขอข้อมูลเข้ามาจะแสดงที่หน้าต่าง Client Connection ส่วนถ้ามีเครื่องให้บริการอื่นเชื่อมต่อเข้ามาจะแสดงที่หน้าตา Cache Connection ส่วนรูป 4.14 เป็นหน้าจอที่เครื่องให้บริการอีกเครื่องหนึ่งที่เชื่อมต่อเข้ามา ก็จะแสดงรายละเอียดเช่นเดียวกัน

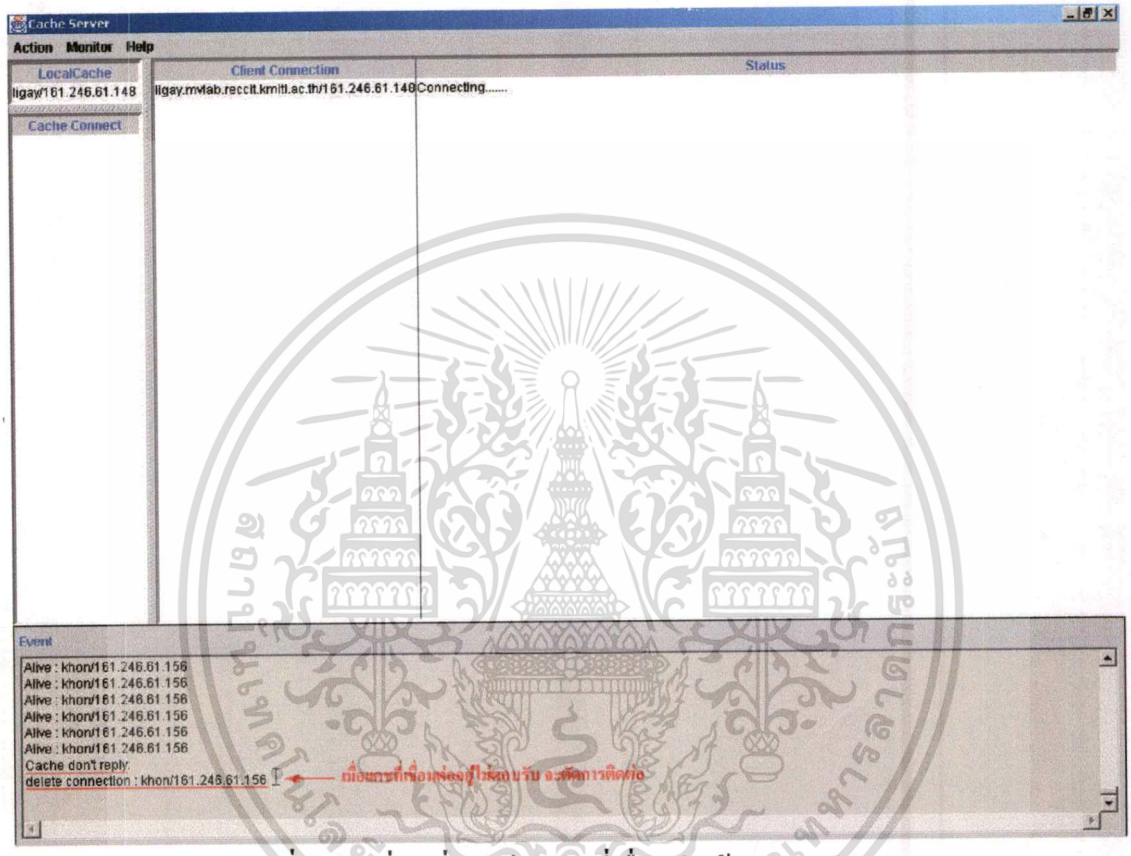


รูปที่ 4.13: หน้าต่างของเครื่องให้บริการแรกเมื่อเครื่องให้บริการอื่นเชื่อมต่อเข้ามา



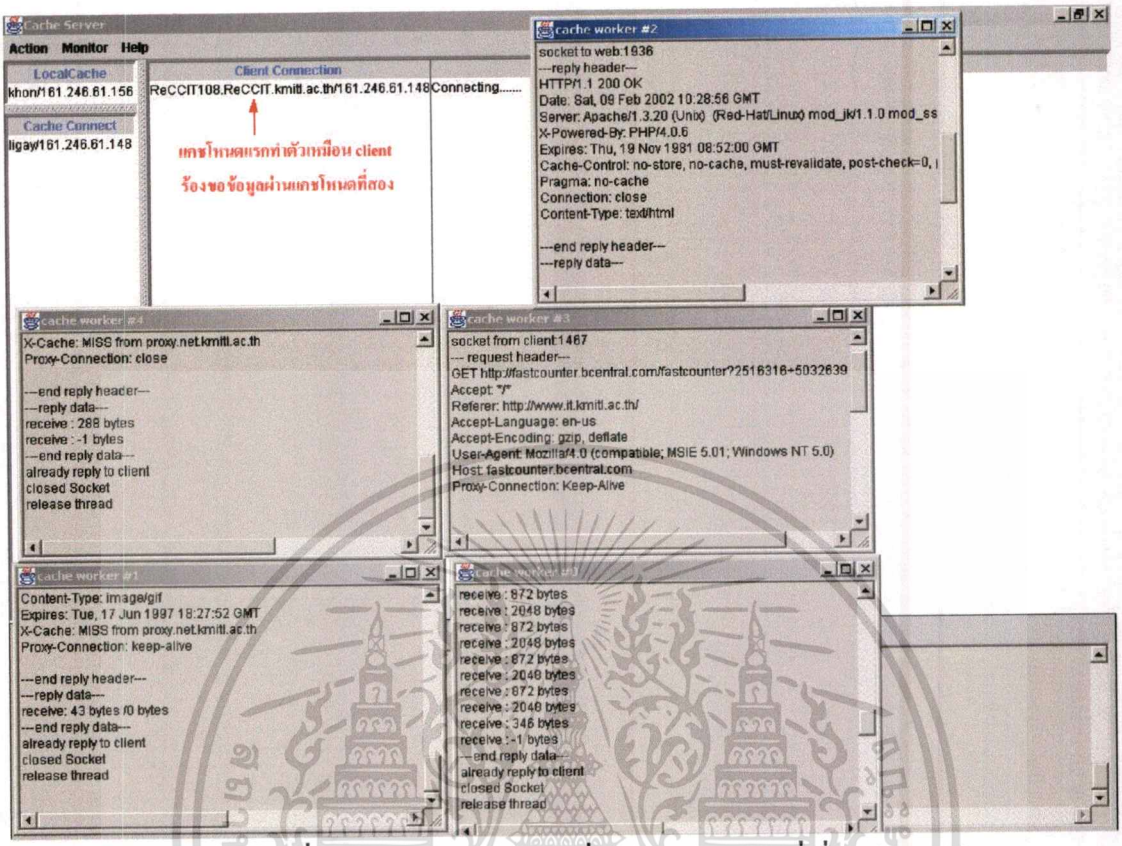
รูปที่ 4.14: หน้าต่างเครื่องให้บริการที่เชื่อมต่อเข้าไปที่เครื่องให้บริการแรก

เมื่อเครื่องให้บริการที่เชื่อมต่อเข้าไปหยุดทำงานไปส่วนที่ใช้ตรวจสอบการมีอยู่ของแคชจะส่งข้อมูลออกไป เมื่อไม่ได้รับการตอบรับ จะทำการตัดการเชื่อมต่อกับเครื่องให้บริการแคชอื่นและลบชื่อเครื่องนั้นออกจากรายชื่อดังรูปที่ 4.15

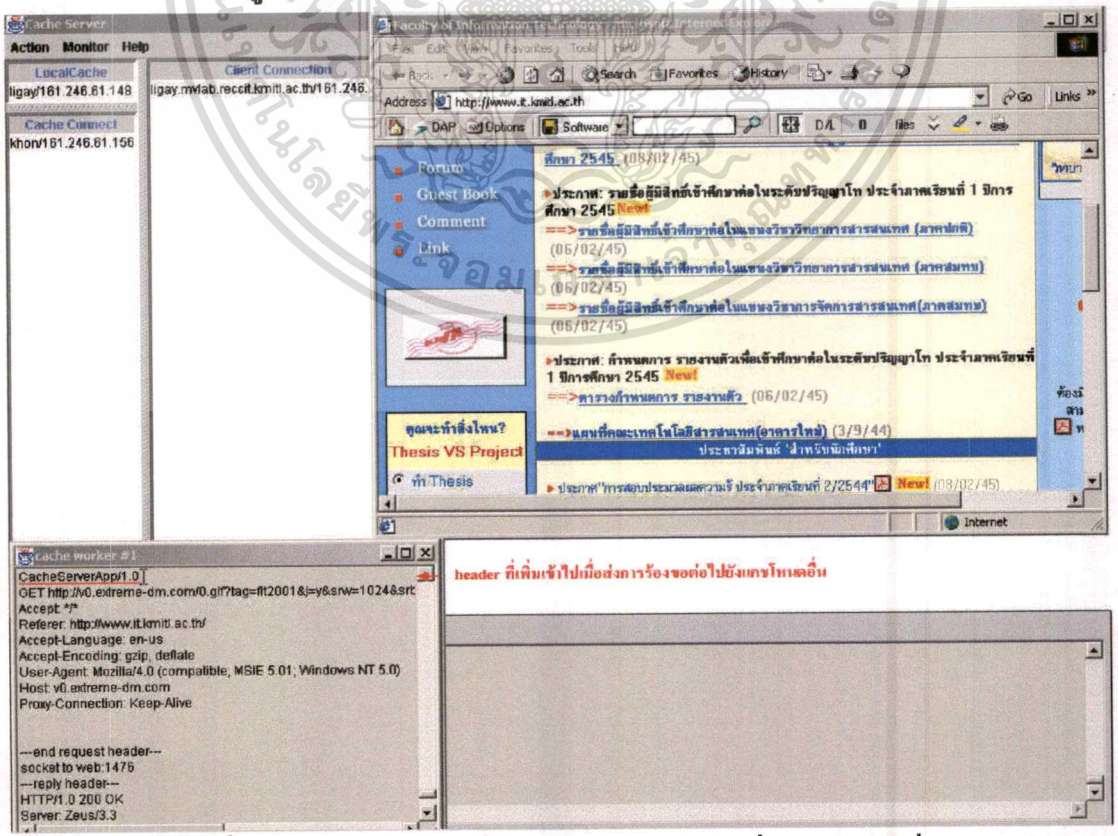


รูปที่ 4.15: เมื่อเครื่องให้บริการที่เชื่อมต่อเข้ามาหยุดทำงาน

ในขณะที่เครื่องให้บริการแคชเชื่อมต่อกันอยู่นั้น ที่หน้าต่างติดตามผลการทำงานของผู้ทำงานในเครื่องนั้นจะแสดงให้เห็นว่ามีข้อมูลการร้องขอส่งมา แสดงว่าในขณะนี้ ได้มีการส่งต่อการร้องขอผ่านเครื่องให้บริการแคชที่เชื่อมต่อกันแล้วดังรูปที่ 4.16 ส่วนรูปที่ 4.17 แสดงให้เห็นว่าเมื่อมีการส่งต่อการร้องขอไปยังเครื่องให้บริการแคชอื่น จะเพิ่มเฮดเดอร์พิเศษเข้าไปเพื่อให้เครื่องให้บริการอีกตัวรับรู้ว่าเป็นการร้องขอที่ถูกส่งมาจากเครื่องให้บริการอื่น ทำให้สามารถแยกแยะและตัดสินใจว่าการร้องขอนั้นควรจะจัดการอย่างไร และส่งต่อไปอย่างไรบ้าง เป็นการจำกัดไม่ให้เครื่องให้บริการที่ได้รับการร้องขอจากเครื่องให้บริการอื่น ส่งการร้องขอต่อไปยังเครื่องให้บริการแคชถัดไปอีก



รูปที่ 4.16: การร้องขอผ่านเครื่องให้บริการเคชที่เชื่อมต่อกันอยู่



รูปที่ 4.17: เฮดเดอร์พิเศษเมื่อส่งการร้องขอไปยังเครื่องให้บริการอื่น

บทที่ 5
วิเคราะห์และสรุปผล

5.1 การใช้งานในสิ่งแวดล้อมจริง

เมื่อได้พัฒนาระบบแคชที่ทำงานร่วมกันขึ้นมาแล้ว ได้ทำงานติดตั้งระบบแคชไปยังสิ่งแวดล้อมจริงภายในเครือข่ายของห้องปฏิบัติการมัลติมีเดีย โดยทำการติดตั้งที่เครื่องคอมพ์พิวเตอร์ที่ทำหน้าที่เป็นเครื่องให้บริการจำนวนสองโหนด โดยที่เครื่องให้บริการแคชโหนดหนึ่งจะทำการเชื่อมโยงกับเครือข่ายของทางสถาบันฯ และอีกโหนดหนึ่งจะทำการเชื่อมโยงกับเครือข่ายผ่านดาวเทียม โดยจะมีการติดต่อระบบแคชสองโหนดด้วยวิธีการตามที่ได้ออกแบบไว้ในบทที่ 3 ดังรูปที่ 3.1

จากนั้นได้ทำการทดลองใช้งานจริงโดยให้ผู้ใช้ซึ่งเป็นผู้ร้องขอข้อมูลทางอินเทอร์เน็ตทำการร้องขอโดยผ่านเครื่องคอมพิวเตอร์ (client) อีกโหนดหนึ่ง เพื่อวัดประสิทธิภาพของระบบโดยพิจารณาได้จากส่วนของการติดตามผล (monitor) ของระบบ ซึ่งได้อธิบายไปแล้วในบทก่อนหน้านี้ โดยจะทำการพิจารณาในเรื่องของ

- *Response Time* : เป็นเวลาในการตอบสนองต่อการร้องขอ โดยจะเริ่มวัดตั้งแต่เมื่อเครื่องให้บริการได้รับการร้องขอจากผู้ใช้ จนกระทั่งเครื่องให้บริการส่งข้อมูลที่ถูกร้องขอจนหมด หน่วยเป็นมิลลิวินาที (millisecond)
- *Cache Hit Ratio* : เป็นอัตราการร้องขอที่ได้รับข้อมูลจากในแคช (หรือที่เรียกว่า cache hit) ต่อการร้องขอทั้งหมด คิดเป็นเปอร์เซ็น (%)
- *Traffic Between Cache Node* : เป็นปริมาณข้อมูลที่ส่งระหว่างแคชโหนดด้วยกัน จะมีข้อมูลในกรณีที่ส่งข้อมูลไปยังเครื่องให้บริการแคชอีกโหนดเท่านั้น

5.1.1 การวัด Response Time

ใช้ในการบอกประสิทธิภาพในเชิงของเวลาที่ใช้เมื่อมีการร้องขอ โดยระบบจะมีประสิทธิภาพดี เมื่อเวลาตอบสนองน้อยลง ในการทดลองวัดผลในการทดลองของเรา จะใช้แคชทั้งสองโหนดทำงานร่วมกัน โดยแบ่งออกเป็น 4 ส่วนการทดลอง คือ

- *ไม่ใช้แคช* : ทำการวัดเวลาการตอบสนองโดยไม่ผ่านแคช แล้วดูเวลาการตอบสนอง
- *เกิด cache hit ที่โหนดแรก* : ในกรณีนี้คือ จะใช้แคชทั้งสองโหนด แต่เกิด hit ที่โหนดแรก แล้วดูเวลาการตอบสนอง

- *เกิด cache hit ที่โหนดที่สอง* : ในกรณีนี้คือ จะใช้แคชทั้งสองโหนด แต่เกิด miss ที่โหนดแรก แล้วทำการติดต่อไปยังโหนดที่สอง แล้วเกิด hit ขึ้น ดูเวลาการตอบสอง
- *เกิด cache miss ทั้งสองโหนด* : ในกรณีนี้คือใช้แคชทั้งสองโหนดเช่นกัน แต่เกิด miss ที่แคชโหนดแรก แล้วติดต่อไปยังโหนดที่สอง แล้วยังเกิด miss ขึ้นอีก จากนั้นทำการส่งการร้องขอไปยังเครื่องให้บริการจริงบนอินเทอร์เน็ต ดูเวลาการตอบสนอง

อีกส่วนหนึ่งในการวัดเวลาการตอบสนองนั้น จะทำการใช้แคชเพียงโหนดเดียว แต่ปรับเปลี่ยนส่วนของขนาดของผู้ทำงาน (worker) ตั้งแต่ 1, 2, 4, 8, 16, 24 และ 32 โพรเซส เพื่อดูว่าขนาดของผู้ทำการมีปัจจัยต่อเวลาการตอบสนองอย่างไรบ้าง เพื่อให้สามารถปรับเปลี่ยนขนาดของผู้ทำงานให้เหมาะสมกับเครือข่าย

5.1.2 การวัด Cache Hit Ratio

ใช้ในการบอกถึงประสิทธิภาพในเรื่องของอัตราในการใช้ข้อมูลในแคชว่ามีมากน้อยเพียงใด โดยอัตราการใช้ข้อมูลในแคชนั้นขึ้นอยู่กับพฤติกรรมของผู้ใช้งานในระบบว่ามีการเรียกใช้ข้อมูลเดิมมากน้อยเพียงใด ซึ่งระบบแคชเกิดประโยชน์เมื่อมีการเรียกใช้ข้อมูลเดิมซ้ำบ่อยๆ ซึ่งก็หมายถึงอัตราการใช้ข้อมูลในแคชมีค่าสูง โดยปกติอยู่ประมาณ 40 – 60 % ซึ่งในการทดลองใช้งานกับสิ่งแวดลอมจริงของเราจะทำการพิจารณาว่าพฤติกรรมของผู้ใช้ในระบบของเรา จะให้อัตราการใช้ข้อมูลในแคชต่อการร้องขอทั้งหมดเท่ากับ 60% โดยทำการร้องขอข้อมูลเดิม โดยมีข้อมูลบางส่วนอยู่ในแคชอยู่แล้ว แต่ทำการปรับเปลี่ยนปัจจัยดังต่อไปนี้

- *ใช้แคชโหนดเดียว* : ในการทดลองนี้จะใช้แคชโหนดเดียว เพื่อใช้เป็นมาตรฐานในการเปรียบเทียบกับปัจจัยอื่น
- *ใช้แคชสองโหนด* : ทำการทดลองโดยใช้แคชทั้งสองโหนด เพื่อดูว่า ในกรณีเกิด miss ที่โหนดแรกแล้ว และมีโหนดที่สองรองรับ มีผลต่อเวลาการตอบสนองว่าลดลงหรือเพิ่มมากขึ้นเพียงใด

ในอีกส่วนของการทดลองจะทำการใช้แคชเพียงโหนดเดียว แต่ทำการปรับเปลี่ยนขนาดของความจุของแคชที่สามารถเก็บได้ โดยปรับเปลี่ยนขนาดตั้งแต่ 0.1 MB, 0.2 MB, 0.4 MB, 0.8 MB, 1.6 MB และ 3.2 MB เพื่อดูว่าขนาดความจุของแคชมีผลต่ออัตราการแคชอย่างไรบ้าง เพื่อให้สามารถหาความจุของแคชที่เหมาะสมได้

5.1.3 การวัด Traffic Between Cache Node

ในการวัดข้อมูลที่ส่งระหว่างแคชโหนดทั้งสองนั้น จะบอกถึงว่าในกรณีใช้แคชเพิ่มขึ้นมาอีกโหนดนั้น จะทำให้มีปริมาณข้อมูลเพิ่มขึ้นภายในเครือข่ายมากน้อยเพียงใด โดยทำการตรวจสอบ

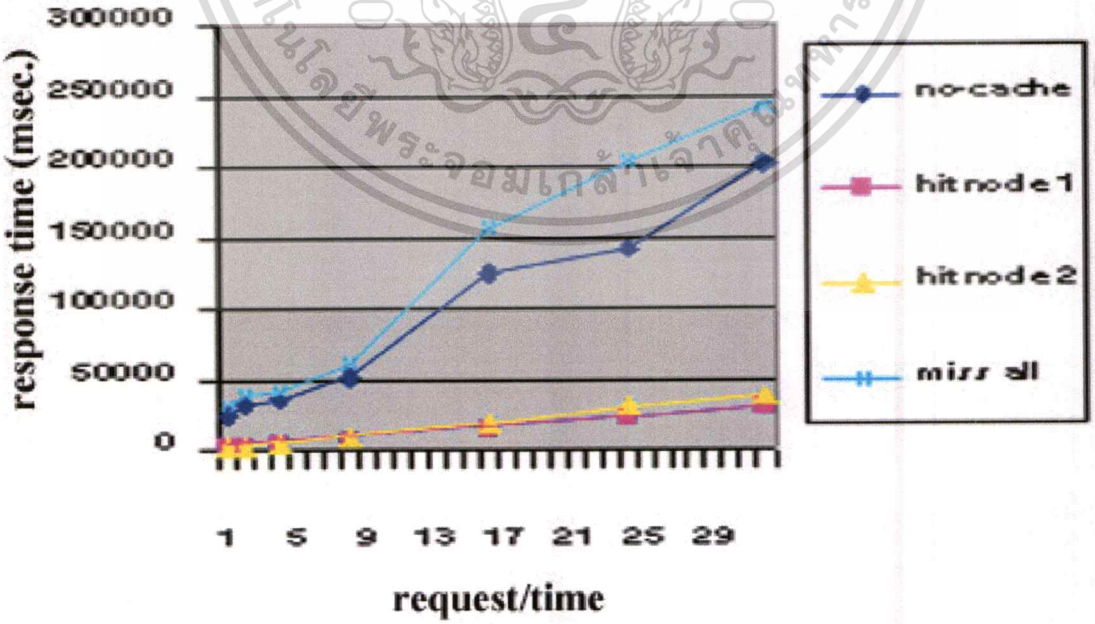
ว่า ถ้ามีการส่งข้อมูลไปยังแคชอีกโหนดหนึ่ง จะทำการบันทึกขนาดของข้อมูลและแสดงผลออกมาผ่านทางหน้าต่างของการติดตามผลของระบบ โดยบอกเป็นขนาดของข้อมูลเป็นไบต์ (byte)

5.2 ผลจากการพัฒนาระบบงาน

จากการพัฒนาระบบแคชที่ทำงานร่วมกันและการทดลองใช้งานในสิ่งแวดล้อมจริง สามารถสรุปผลในลักษณะของกราฟความสัมพันธ์ โดยทำการบันทึกผลจากส่วนของการติดตามผลของระบบ (monitor system) ทำการรวบรวมข้อมูล และใช้โปรแกรมช่วยคือ microsoft excel ในการสร้างกราฟ ได้ผลออกมาดังนี้

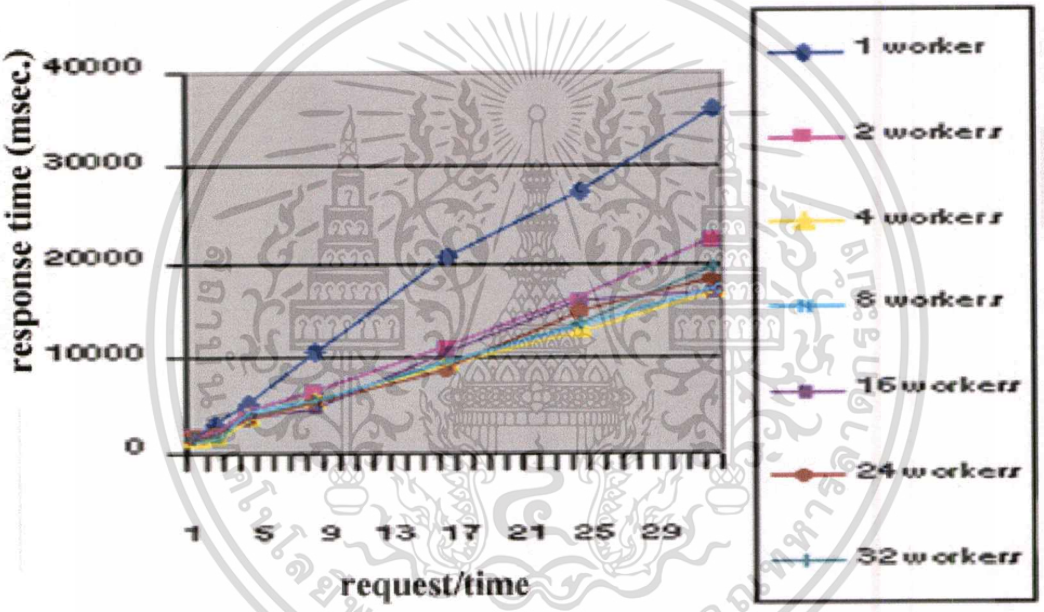
5.2.1 เวลาการตอบสนองของระบบ (response time)

จากรูปที่ 5.1 จะเห็นว่าเวลาการตอบสนองต่อการร้องขอข้อมูลเมื่อผ่านแคชแล้วจะให้ผลตอบสนองที่ช้ากว่าเมื่อผ่านแคชในกรณีที่ไม่มีข้อมูลที่ร้องขออยู่ในแคชเลยซึ่งก็คือกรณีเกิด miss ทั้งสองโหนด จากกราฟจะเห็นว่ากรณีเกิด hit ที่โหนดแรกเลย จะให้เวลาตอบสนองที่เร็วกว่าในกรณีเกิด hit ที่โหนดที่สองเล็กน้อย เนื่องจากต้องเสียเวลาส่วนหนึ่งในการสร้างการเชื่อมต่อและส่งข้อมูลระหว่างโหนดของแคช แต่จะสังเกตุเห็นได้อย่างชัดเจนว่ากรณีที่เกิด hit ที่โหนดใดโหนดหนึ่งนั้น จะให้ผลตอบสนองที่เร็วกว่าการร้องขอไปยังเครื่องให้บริการต้นกำเนิดอย่างมาก แต่ในกรณีร้องขอข้อมูลมูลขนาดเล็กและเครือข่ายมีความเร็วสูง อาจไม่เห็นความแตกต่างมากนัก



รูปที่ 5.1: เวลาการตอบสนองของระบบแคชเมื่อ:ไม่ใช่แคช, เกิด hit ที่โหนดแรก, miss ที่โหนดแรกและ hit ที่โหนดสอง, miss ทั้งสองโหนด

ส่วนในรูปที่ 5.2 เป็นการแสดงเวลาการตอบสนองเมื่อเปลี่ยนจำนวนผู้ทำงาน (worker) โดยใช้เพียงโหนดเดียวในการทดลอง จะเห็นว่าเมื่อผู้ทำงานมีจำนวนน้อยกว่าโพรเซสที่ร้องขอมากๆ ส่งผลให้เวลาตอบสนองช้าลงอย่างเห็นได้ชัด แต่เมื่อเพิ่มผู้ทำงานจนถึงประมาณ 8 ผู้ทำงานขึ้นไปแล้วจะให้ผลตอบสนองใกล้เคียงกัน เนื่องจากว่า ในความเป็นจริงแล้ว เมื่อมีการร้องขอเข้ามาพร้อมกันก็ตาม แต่การประมวลผลข้อมูลแต่ละชุดมีขนาดไม่เท่ากัน ดังนั้นผู้ทำงานหนึ่งอาจทำงานเสร็จก่อน แล้วสามารถให้บริการการร้องขออื่นๆต่อไปได้ ฉะนั้นการเพิ่มจำนวนผู้ทำงานมากจนเกินไปจะไม่ส่งผลให้เวลาตอบสนองเร็วขึ้น และทำให้เสียหน่วยความจำโดยเปล่าประโยชน์ด้วย

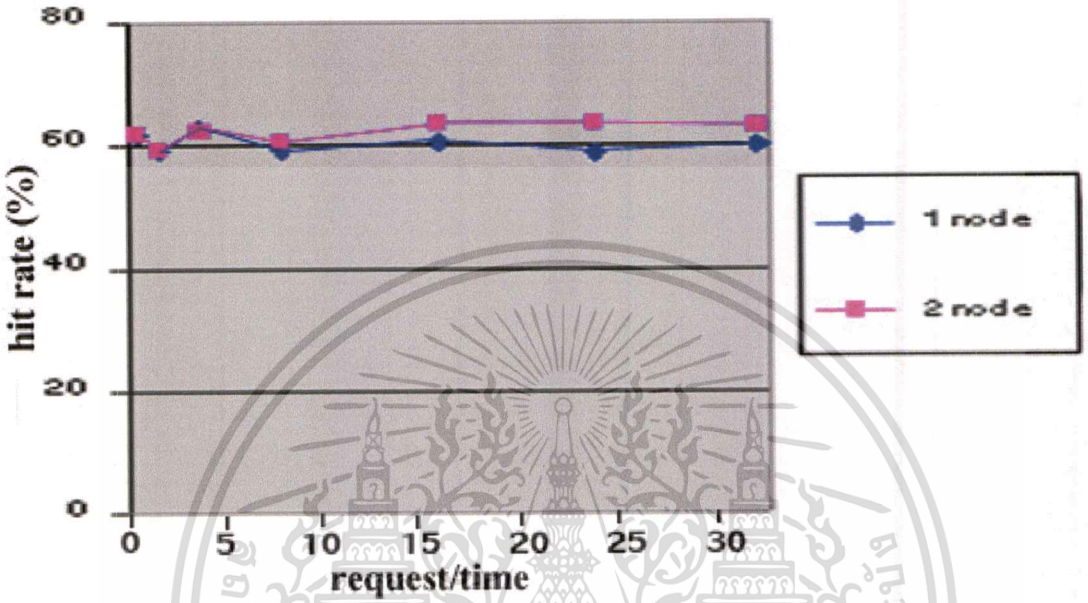


รูปที่ 5.2: เวลาการตอบสนองเมื่อเปลี่ยนจำนวนผู้ทำงาน (ในกรณีเกิด hit ทุกครั้งที่โหนดแรก)

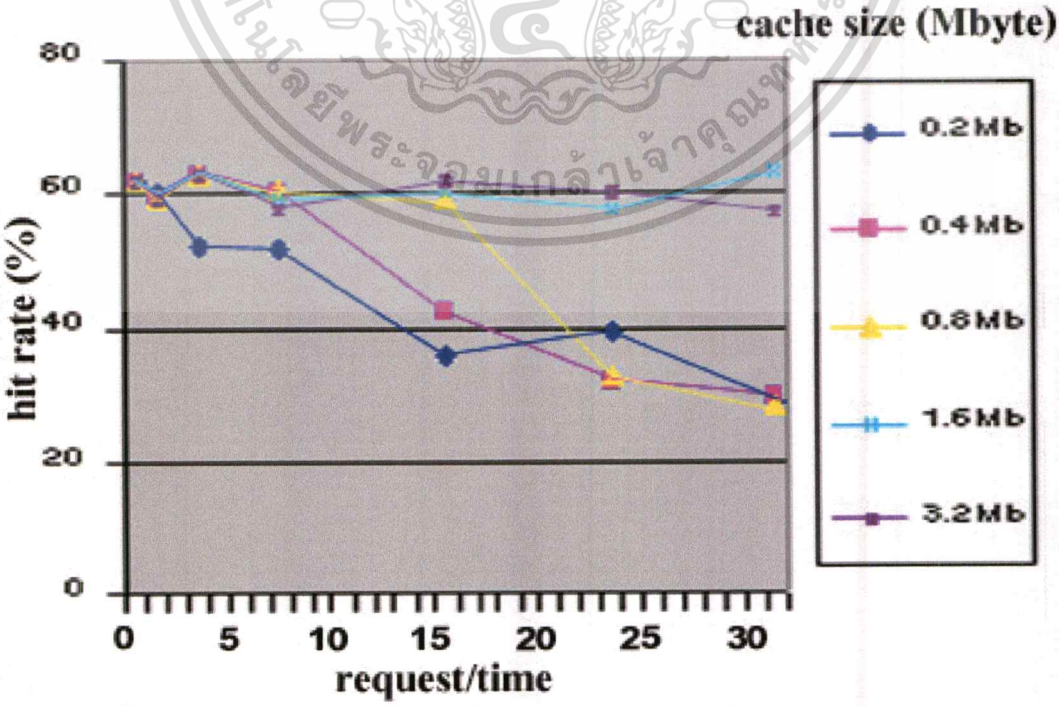
5.2.2 อัตราการแคช (cache hit ratio)

อัตราการแคชขึ้นอยู่กับพฤติกรรมของผู้ใช้ แต่โดยทั่วไปอยู่ประมาณ 40-60 % ฉะนั้นในการทดลองเราจะพิจารณาโดยใช้หลักว่าจะเกิด hit ที่ 60 % โดยรูปที่ 5.3 จะเห็นว่าการใช้แคช 2 โหนดจะให้อัตราการแคชมากกว่าเล็กน้อย หรืออาจใกล้เคียงกัน เนื่องจากว่าเมื่อเกิด miss ที่โหนดแรกแล้ว ถ้าใช้ 2 โหนดอาจเกิด hit ที่โหนดที่สองแทน ทำให้อัตราการแคชไม่ลดลงมากนัก หรืออาจมากกว่าเดิมด้วยซ้ำ อีกทั้งการใช้ 2 โหนดทำงานร่วมกัน จะเห็นเหมือนการเพิ่มพื้นที่ในการจัดเก็บแคช ซึ่งมีผลต่ออัตราการ hit โดยตรง สังเกตุได้จากรูป 5.4 เมื่อขนาดความจุของแคชไม่เพียงพอทำให้จำเป็นต้องผ่านกระบวนการแทนที่แคช ซึ่งแน่นอนว่าข้อมูลที่แคชบางส่วนจะถูกลบออกไป ถ้าทำกระบวนการนี้บ่อยๆ ทำให้อัตราการ hit ต่ำ เมื่อเพิ่มความจุจนถึงระดับหนึ่งที่เหมาะสมแล้ว

อัตราการ hit จะมีค่าใกล้เคียงกัน แต่อย่างไรก็ตาม ถ้ามีเนื้อที่เพียงพอแล้ว การเพิ่มความจุแคชให้มากเข้าไว้จะส่งผลดีมากกว่า โดยไม่มีผลเสียใดๆต่อระบบ



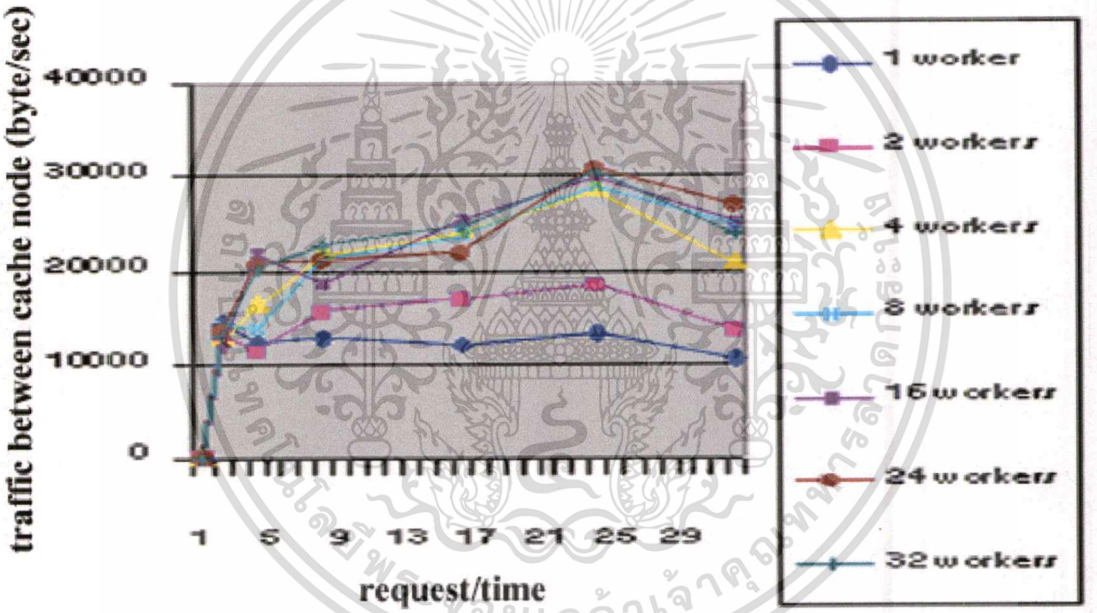
รูปที่ 5.3: อัตราการเกิด cache hit เมื่อมีโหนดเดียวและสองโหนด(ในกรณี hit ที่โหนดที่สอง)



รูปที่ 5.4: อัตราการเกิด cache hit เมื่อขนาดความจุแคชเปลี่ยนไป (ใช้โหนดเดียว)

5.2.3 ปริมาณการจราจรที่เพิ่มขึ้นภายในเครือข่าย

ปริมาณการจราจรที่เพิ่มขึ้นเนื่องจากการติดต่อระหว่างแคชโหนดทั้งสอง จะเพิ่มขึ้นตามปริมาณข้อมูลที่สามารถส่งได้ต่อหน่วยเวลา จากรูปที่ 5.5 ข้อมูลที่ส่งจะถูกจำกัดด้วยจำนวนผู้ทำงาน จะเห็นได้ชัดว่าที่ผู้ทำงานจำนวนน้อย หมายความว่า มีเส้นทางที่เปิดรับการร้องขอน้อย ฉะนั้น ปริมาณข้อมูลที่ส่งระหว่างโหนดจึงมีน้อยด้วย เมื่อจำนวนผู้ทำงานเพิ่มขึ้นถึงระดับหนึ่งแล้ว ปริมาณข้อมูลที่ส่งระหว่างโหนดจะมีค่าใกล้เคียงกัน ซึ่งให้ผลสอดคล้องกับรูปที่ 5.2 เนื่องจากเวลาการตอบสนองเมื่อจำนวนผู้ทำงานประมาณ 8 ผู้ทำงาน จะให้เวลาการตอบสนองที่ใกล้เคียงกัน แต่ปริมาณข้อมูลที่ส่งมีจำนวนเท่าเดิม ฉะนั้นการจราจรระหว่างโหนดคือ ปริมาณข้อมูล/เวลาการตอบสนอง เมื่อเวลาตอบสนองใกล้เคียงกัน ส่งผลให้การจราจรที่ส่งระหว่างโหนดใกล้เคียงกันด้วย



รูปที่ 5.5: การจราจรระหว่างโหนดของแคชเมื่อจำนวนผู้ทำงานเปลี่ยนไป
(กรณี miss ที่โหนดแรกและ hit ที่โหนดสอง)

บทที่ 6
บทสรุป

6.1 ประโยชน์ที่ได้รับ

เครื่องให้บริการแคชที่ทำงานร่วมกัน เอื้อประโยชน์ต่อระบบเครือข่ายของเรา เนื่องจากว่าสามารถทำให้เส้นทางการเชื่อมต่อไปยังอินเทอร์เน็ตทั้งสองสามารถใช้ร่วมกันได้ เมื่อผู้ใช้ทำการร้องขอข้อมูลบนเว็บไซต์ เครื่องให้บริการแคชสามารถเลือกเส้นทางการส่งต่อการร้องขอไปยังช่องทางใดก็ได้ โดยที่มีแคชอีกตัวเป็นเสมือนทางผ่านของการร้องขอ แต่ถ้าเกิดเจอข้อมูลที่แคชโหนดที่สอง จะทำให้อัตราการแคชเพิ่มมากขึ้น เวลาการตอบสนองก็น้อยลงด้วย โดยที่เครื่องให้บริการสามารถทำงานอย่างอัตโนมัติในการรับรู้การเชื่อมต่อกับโหนดอื่น ไม่ต้องยุ่งยากในการปรับแต่งค่าใดๆ

สรุปว่า โครงการพัฒนาระบบงานนี้ สามารถทำงานได้ตามเป้าหมายที่ได้กำหนดไว้ สามารถใช้งานได้ในสิ่งแวดล้อมจริง โดยให้ผลเป็นที่น่าพอใจ และเกิดความผิดพลาดเล็กน้อยในระดับที่สามารถยอมรับได้

6.2 ปัญหาที่เกิดขึ้น

ระบบแคชที่ทำงานร่วมกันนี้ มีปัญหาเกิดขึ้นหลายประการเนื่องจาก

6.2.1 ใน HTTP/1.1 มีเฮดเดอร์เพิ่มเข้ามาจาก HTTP/1.0 เป็นจำนวนมาก ซึ่งจะช่วยอำนวยความสะดวกในส่วนของการติดต่อระหว่างแคชมากขึ้น แต่ในโครงงานนี้ไม่ได้พัฒนาให้ครอบคลุมทุกเฮดเดอร์ จึงอาจจะไม่สามารถเข้าถึงบางเว็บไซต์อย่างสมบูรณ์ได้

6.2.2 เนื่องจากในระบบแคชนั้นมีส่วนที่เกี่ยวข้องมากมาย เช่น ด้านการจัดการแคชอย่างมีประสิทธิภาพ ด้านฐานข้อมูลการจัดเก็บแคช ด้านการเชื่อมต่อระหว่างเครือข่าย เป็นต้น แต่ด้วยเวลาที่มีจำกัด ฉะนั้นบางส่วนของระบบจึงใช้รูปแบบพื้นฐานที่สุดที่มีอยู่ เช่น การจัดเก็บในโครงงานใช้แบบแบน (flat file) หรือจัดเก็บในระดับเดียวกันทั้งหมด การจัดการแคชนั้นใช้รูปแบบของ least-frequently-used ซึ่งเป็นรูปแบบที่ง่าย

6.2.3 ในการเชื่อมต่อที่จำนวนผู้ทำงาน (worker) มากๆ และมีการร้องขอข้อมูลพร้อมกันหลายๆเส้นทาง บางครั้งอาจทำให้ข้อมูลสูญหายไปบ้าง อาจเนื่องจากโพรเซสไม่สามารถตอบสนองได้ทัน และการทำงานที่ไม่สัมพันธ์กันของแต่ละโพรเซสที่แยกการทำงานกันอย่างอิสระ

6.2.4 สำหรับระบบเครือข่ายที่ใช้งานอยู่ทั้งสองระบบ มีความเร็วค่อนข้างสูงอยู่แล้ว การร้องขอข้อมูล (object) ที่มีขนาดเล็ก เมื่อผ่านระบบแคชอาจได้ผลตอบสนองที่ช้ากว่าการร้องขอโดยตรงได้ เนื่องจากต้องเสียเวลาโพรเซสภายในตัวแคชเอง

6.2.5 บางครั้งข้อมูลอาจถูกแคชไว้ที่เครื่องให้บริการแคชที่อยู่เหนือขึ้นไป ซึ่งเป็นของผู้ให้บริการ จึงอาจไม่จำเป็นต้องแคชไว้ที่เครื่องให้บริการแคชของเราเอง เนื่องจากผลตอบสนองที่ได้จะเร็วอยู่แล้ว

6.3 ข้อเสนอแนะ

ระบบแคชที่ทำงานร่วมกันสำหรับเครือข่ายอินเทอร์เน็ตผ่านดาวเทียมที่ได้พัฒนาขึ้นมาเป็นเพียงการพัฒนาขั้นต้น ยังมีอีกหลายส่วนที่ควรพัฒนาต่อไป เพื่อให้ระบบมีประสิทธิภาพมากขึ้น ไม่ว่าจะเป็น ระบบการจัดเก็บข้อมูลบนแคช การแทนที่แคชเมื่อความจุของแคชเต็ม การจัดการกับแคชให้บริหารพื้นที่อยู่มีประสิทธิภาพและลบส่วนของข้อมูลที่ไม่จำเป็นออกไป การเลือกเส้นทางการส่งต่อการร้องขอที่เหมาะสม การพัฒนาให้เครื่องให้บริการสามารถครอบคลุมเฮดเดอร์ทั้งหมด เรื่องของเสถียรภาพของระบบแคช รวมถึงเรื่องของการรักษาความปลอดภัยของข้อมูลด้วย

บรรณานุกรม

จิตกุศล ไม้เกต. 2542. **"การศึกษาวิธีการสื่อสารข้อมูลผ่านดาวเทียมโดยใช้ความถี่ย่าน Ku-Band."** บทความสัมมนา สาขาวิชาวิทยาการสารสนเทศ คณะเทคโนโลยีสารสนเทศ, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

เจนวิทย์ เหลืองอร่าม, พ.ต. 2542. **การเขียนโปรแกรม สำหรับ Application และ Applets ด้วย Java.** กรุงเทพฯ: ซีเอ็ดยูเคชั่น.

Baentsch,M. et. Al. 1997. **"World Wide Web Caching: The Application-Level View of the Internet".** IEEE Communications Magazine:170-178.

Barish,G. and Obraczka,K. 2000. **"World Wide Web Caching: Trends and Techniques"**, IEEE Communication Magazine:178-186.

Doherty,D. and Manning,M.M. 1998. **"Teach Yourself Jbuilder 2 in 21 Days".** Indiana: Sams Publishing.

Inohara,S. et. Al. 1998. **"Seld-Organizing Cooperative WWW Caching".** 0-8186-8292-2/98. IEEE:74-83.

Luotonen,A. 1998. **"Web Proxy Sever".** New York:Prentice Hall PTR.

Mark Nottingham. 2000. **Caching Tutorial for Web Authors and Webmasters.** [online] Available:http://www.mnot.net/cache_docs/, June 2000.

Steflik,D. and Sridharan,P. 2000. **Advanced JAVA networking.** Second edition. New York: Prentice Hall PTR.

Thaicom Satellite Network. **"Internet via Satellite".** [online]. Available: http://www.thaicom.net/internet-via/platform.html.

ภาคผนวก

โปรแกรมระบบแคชที่ทำงานร่วมกัน

Projectfile: CacheServerApp.jpx ประกอบด้วย File หลักๆ ดังนี้

- *CacheServerApp.java* : รวมส่วนของ Main Program, Cache Manager, Cache Worker และส่วนที่เกี่ยวข้องไว้
- *MenuFrame.java* : หน้าจอหลักของ User Interface
- *ConfigFrame.java* : หน้าจอ Configulation
- *Cframe.java* : หน้าจอ Monitor

CacheServerApp.java

```
//CacheServerApp.java

package CacheServerApp;

import java.awt.*;
import java.util.*;
import java.net.*;
import java.io.*;
import java.lang.Number;

class CacheServerApp implements HttpConstants
{
 // Where worker threads stand idle
 static Vector threads = new Vector();
 static Vector StatusTab = new Vector();
 static Vector ClientTab = new Vector();

 //Main Program
 public static void main(String args[])throws Exception
 {
  MenuFrame FPage = new MenuFrame();
  FPage.inCacheIndex(threads);
  setInitMenu(FPage);
  FPage.show();
  FPage.jTextArea_LocalCache.setText(InetAddress.getLocalH
ost().toString());
  }
  //end Main Program

 static void setInitMenu(MenuFrame frame)
 {
  //Center the window
  Dimension screenSize = Toolkit.getDefaultToolkit
().getScreenSize();
  Dimension frameSize = frame.getSize();
  if (frameSize.height > screenSize.height) {
   frameSize.height = screenSize.height;
  }
  if (frameSize.width > screenSize.width) {
   frameSize.width = screenSize.width;
  }
  frame.setLocation((screenSize.width - frameSize.width) / 2,
(screenSize.height - frameSize.height) / 2);


  frame.setVisible(true);
 }
```

```
}

//Start Cache Manager Class Process
class CacheManager extends CacheServerApp implements
HttpConstants,Runnable
{
 // max # worker threads
 static int workers = 5;
 // Set Additional Worker Active
 static boolean AddWorker = false;
 String status="";
 MenuFrame FPage;
 ServerSocket ss;
 Socket s;
 CheckStatus cStatus;
 boolean error = false;

 void SetFrame(MenuFrame FPage)
 {
  this.FPage = FPage;
 }

 void stop()
 {
  try{ss.close();}catch(IOException e){}
 }

 public void run()
 {
  FPage.textArea_Event.append(Thread.currentThread
().getName()+
                    " Process : ON\r\n");
  workers =
Integer.decode(FPage.ConF.jTextField_worker.getText().trim
()
        ).intValue();
  int port =
Integer.decode(FPage.ConF.jTextField_Port.getText().trim()
        ).intValue();
  int index = 0;
  String[] HostTable = new String[workers];
  int selectPath = 0;
  if(FPage.ConF.State_CB_Single == false)
  {
   cStatus = new CheckStatus(FPage);
   (new Thread(cStatus, "Check Status Process")).start();
   SendStatus info = new SendStatus(FPage);
   (new Thread(info, "send imformation Process")).start();
   VerifyStatus verify = new VerifyStatus(FPage);
   (new Thread(verify, "verify status Process")).start();
  }
  cacheManagement cmm = new cacheManagement(FPage);
  (new Thread(cmm, "Cache Management")).start();
  /* start cache worker threads */
  threads.clear();
  for (int i = 0; i < workers; ++i)
  {
   CacheWorker cw = new CacheWorker(FPage);
   (new Thread(cw, "cache worker #"+i)).start();
   threads.addElement(cw);
   HostTable[i]="";
  }
  try
  {
   ss = new ServerSocket(port);
  }catch(IOException e)
  {
   FPage.textArea_Event.append("Create Sever Socket
error.");
   System.out.println("Create Sever Socket error.");
   error=true;
  }
  //if serversocket if not error
  if(!error)
  {
   FPage.textArea_Event.append("Path to Cache : "+
    FPage.ConF.jTextField_pathToCache.getText()+"\r\n");
   FPage.textArea_Event.append("CacheServerApp is
listening on port " +port +
                    ".\r\n");
```

```
while (true&&!FPage.StopServer)//test:stop server
{
 try
 {
  try{Thread.sleep(200);}catch(InterruptedException e){}
  s = ss.accept();
  Date now = new Date();
  System.out.println(now.getTime());
  boolean insertClient=true;
  //update table for client connection
  for(int i=0;i<ClientTab.size();i++)
  {
   if(ClientTab.elementAt(i).toString().startsWith(
     s.getInetAddress().toString()))
      insertClient=false;
  }
  if(insertClient)
  {
   synchronized(ClientTab)
   {
    ClientTab.addElement(s.getInetAddress());
   }
  }
  FPage.jTextArea_Connection.setText("");
  FPage.jTextArea_Status.setText("");
  //show client connection to UI
  for(int i=0;i<ClientTab.size();i++)
  {

FPage.jTextArea_Connection.append(ClientTab.elementAt
(i)+"\r\n");

FPage.jTextArea_Status.append("Connecting.....\r\n");
  }
  boolean newAccept = false;
  requestHeader reqH = new requestHeader();
  PrintStream outToClient = new PrintStream
(s.getOutputStream());
  DataInputStream inFromClient = new DataInputStream
(s.getInputStream());
  String readlineH,Host,Method="";
  StringBuffer requestHeader = new StringBuffer();
  //test check request from other cache
  if((readlineH=inFromClient.readLine())!=null)
  {//Check Null Pointer
   do
   {//receive header from client
    if(readlineH.startsWith("CacheServerApp/1.0"))
    {//check request from other cache
     reqH.ReqFromCache = true;
     System.out.println("Request From Other Cache");
    }else
    {//request from user
     requestHeader.append(readlineH+"\r\n");
     if((readlineH.startsWith("GET
"))||(readlineH.startsWith("HEAD "))||
      (readlineH.startsWith("POST ")))
       reqH.Method = readlineH;
     if(readlineH.startsWith("Host:"))
      reqH.Host = readlineH.substring(6);
     if(readlineH.endsWith("no-
cache")||readlineH.endsWith("no-store"))
      reqH.noUseCache=true;
     if(readlineH.startsWith("Range:"))
      reqH.noUseCache=true;
     if(readlineH.startsWith("Content-Length:"))
      reqH.Content_Length = Integer.decode(
       readlineH.substring(16).trim()).intValue();
     if(readlineH.startsWith("Referer:"))
     {
      reqH.Ref = readlineH.substring(9);
      reqH.Ref = reqH.Ref.substring(reqH.Ref.indexOf
('/')+2);
      reqH.Ref = reqH.Ref.substring(0,reqH.Ref.indexOf
('/'));
     }
    }
   }while((readlineH=inFromClient.readLine()).length
()!=0);
   requestHeader.append("\r\n");
```

```
        System.out.println(requestHeader.toString());
        // test include extra header to request to other server
        if(reqH.ReqFromCache == false)
        {
         if(FPage.ConF.jCheckBox1.isEnabled
()&&FPage.ConF.jCheckBox1.isSelected())
          //if select static path
          selectPath = FPage.ConF.selectPath;
         if(selectPath == 0)
         {
         }else
         {//request to other cache
          String cacheHost = StatusTab.elementAt(selectPath-
1).toString();
          cacheHost =
cacheHost.substring(cacheHost.indexOf("/")+1);
          reqH.Host = cacheHost;
          reqH.Port = port;
          requestHeader.insert(0,"CacheServerApp/1.0\r\n");
         }
         selectPath++;
         if(selectPath>StatusTab.size())selectPath=0;
        }
        synchronized (threads)
        {
         CacheWorker cw = null;
         for(int i=0;i<workers;i++)//set index for use or wait
process
         {
          cw = (CacheWorker) threads.elementAt(i);
          //not process and not keepalive,that host in table is
deleted
          if((!cw.process())&&(!cw.keepalive()))
           //update URL that request table
           synchronized (HostTable)
           {
            HostTable[i]="";
           }
           System.out.println((i+1)+"Table:"+HostTable[i]);
           //same host and keepalive,
           if((HostTable[i].startsWith(reqH.Host))&&
(cw.keepalive()))
           {
            index = i;
            System.out.println("testKeepAlive");
           }
          }
          System.out.println("index "+index);
          cw = (CacheWorker) threads.elementAt(index);

cw.setSocket(s,requestHeader,outToClient,inFromClient,reqH)
;
          HostTable[index]=cw.getName();
          index++;
          if(index>workers-1)index=0;
          }
         }
        }else
        {
         FPage.textArea_Event.append("Cache Manager Socket
Null Pointer\r\n");
         System.out.println("Cache Manager Socket Null
Pointer");
         outToClient.println("HTTP/1.0 404 Not Found");
         s.shutdownOutput();
        }
       }catch(IOException e)
       {
        FPage.textArea_Event.append("Cache Manager Socket
Error\r\n");
        System.out.println("Cache Manager Socket Error");
       }
      }
      try{
       ss.close();
      }catch(IOException e){}
     }
     FPage.textArea_Event.append("Cache Manager Close\r\n");
     System.out.println("Cache Manager Close");
```

```
   FPage.textArea_Event.append("Cache Manager Process :
OFF\r\n");
   if(!FPage.ConF.State_CB_Single)
cStatus.StatusSocket.close();
   System.out.println("exit manager");
  }

  static String ConvertURL(String s)
  {
   String TempString = s.replace('/','`');
   TempString = TempString.replace(':','!');
   TempString = TempString.replace('?','^');
   TempString = TempString.replace('*','#');
   return TempString;
  }

  static String getFileNameURL(String s)
  {
   String fileName = s.substring(s.indexOf(' ') + 1);
   fileName = fileName.substring(0,fileName.indexOf(' '));
   try
   {
    if(fileName.charAt(0) == '/')
        fileName =  fileName.substring(1);
   }
   catch(StringIndexOutOfBoundsException e)
{System.out.println("error file");}

   if(fileName.equals(""))fileName = "default.html";
   return fileName;
  }

  boolean testDate(String D)
  {
   String Value;
   String TempDate;
   if(D.indexOf("-")>0)
   {
    Value = D.substring(0,D.indexOf("-"));
    TempDate = D.substring(D.indexOf("-")+1);
   }
   else
   {
    Value = D.substring(0,D.indexOf(" "));
    TempDate = D.substring(D.indexOf(" ")+1);
   }
   if(Value.startsWith("0"))Value=Value.substring(1);
   int Day = Integer.decode(Value).intValue();
   int Month=0;
   if(D.indexOf("-")>0)
   {
    Value = TempDate.substring(0,D.indexOf("-"));
    TempDate = TempDate.substring(D.indexOf("-")+1);
   }
   else
   {
    Value = TempDate.substring(0,TempDate.indexOf(" "));
    TempDate = TempDate.substring(TempDate.indexOf("
")+1);
   }
   if(Value.startsWith("Jan"))Month=0;
   if(Value.startsWith("Feb"))Month=1;
   if(Value.startsWith("Mar"))Month=2;
   if(Value.startsWith("Apr"))Month=3;
   if(Value.startsWith("May"))Month=4;
   if(Value.startsWith("Jun"))Month=5;
   if(Value.startsWith("Jul"))Month=6;
   if(Value.startsWith("Aug"))Month=7;
   if(Value.startsWith("Sep"))Month=8;
   if(Value.startsWith("Oct"))Month=9;
   if(Value.startsWith("Nov"))Month=10;
   if(Value.startsWith("Dec"))Month=11;

   Value = TempDate.substring(0,TempDate.indexOf(" "));
   TempDate = TempDate.substring(TempDate.indexOf("
")+1);
   int Year = Integer.decode(Value).intValue()-1900;

   Date date = new Date(Year,Month,Day);
```

```
    Value = TempDate.substring(0,TempDate.indexOf(":"));
    if(Value.startsWith("0"))Value=Value.substring(1);
    TempDate = TempDate.substring(TempDate.indexOf
(":")+1);
    int Hour = Integer.decode(Value).intValue();
    date.setHours(Hour);

    Value = TempDate.substring(0,TempDate.indexOf(":"));
    if(Value.startsWith("0"))Value=Value.substring(1);
    TempDate = TempDate.substring(TempDate.indexOf
(":")+1);
    int Minute = Integer.decode(Value).intValue();
    date.setMinutes(Minute);

    Value = TempDate.substring(0,TempDate.indexOf(" "));
    if(Value.startsWith("0"))Value=Value.substring(1);
    int Second = Integer.decode(Value).intValue();
    date.setSeconds(Second);

    Date nowDate = new Date();

    return nowDate.after(date);
   }

}
// End Cache Manager Process

// Start Cache Worker Class Process
class CacheWorker extends CacheManager implements
HttpConstants,Runnable
{
  static final byte[] EOL = {(byte)'\r', (byte)'\n' };
  Vector WaitToConnect = new Vector();
  String pathToCache;
  int traffic=0;
  long restime=0;
  String threadName="";
  boolean KeepAlive = false; // Check KeepAlive is enable ??
  boolean isProcess = false; // Check Process is working ??
  byte[] buff;
  private Socket s;
  Socket LastSocket;
  DataOutputStream LastDOS;
  DataInputStream LastDIS;
  Socket socket;
  DataOutputStream dos = null;
  DataInputStream dis = null;
  String LastHost;

  MenuFrame FPage;// = new MenuFrame();
  CFrame CachePage = new CFrame();

  requestHeader requestH = new requestHeader();
  StringBuffer requestHeader;// = new StringBuffer();
  String Method;
  String DefaultHost ="www.it.kmitl.ac.th";
  PrintStream outToClient;DataInputStream inFromClient;

  CacheWorker(MenuFrame FPage)
  {
   buff = new byte[2048];
   s = null;
   this.FPage = FPage;
  }

  synchronized void setSocket(Socket s,StringBuffer
requestHeader,
   PrintStream outToClient,DataInputStream
inFromClient,requestHeader reqH)
  {
   this.s = s;
   this.requestHeader = requestHeader;
   this.outToClient = outToClient;
   this.inFromClient = inFromClient;
   requestH = reqH;
   notify();
  }

  public void ShowMoniter()
  {
```

```
 CachePage.show();
}

String getName()
{
 return requestH.Host;
}

boolean keepalive()
{
 return KeepAlive;
}

boolean process()
{
 return isProcess;
}

synchronized void stop()
{
 notify();
}
public synchronized void run()
{
 int hit=0;
 pathToCache =
FPage.ConF.jTextField_pathToCache.getText();
 String Host= DefaultHost;
 int cachesize = (int)(Double.valueOf(
          FPage.ConF.jTextField_CacheSize.getText
()).doubleValue()
          *1000000);
 threadName = Thread.currentThread().getName();
 CachePage.setTitle(threadName);
 do
 {
  if ((s == null))
  { /* nothing to do */
   try
   {
    isProcess = false;
    FPage.textArea_Event.append(threadName+
     " : Wait For Request\r\n");
    wait();
    FPage.textArea_Event.append(threadName+
     " : Received Request\r\n");
    if(!requestH.Host.equalsIgnoreCase(LastHost))
KeepAlive = false;
    isProcess = true;
   }catch (InterruptedException e)
   { /* should not happen */
    continue;
   }
  }else
  {
   restime = 0;
   Date now = new Date();
   restime = now.getTime();
   CachePage.textArea1.append("new connection\r\n");
   CachePage.textArea1.append("socket from
client:"+s.getPort()+"\r\n"+
                "--- request header---\r\n");
   boolean finished = false;
   CachePage.textArea1.append(requestHeader.toString
()+"\r\n");
   CachePage.textArea1.append("---end request header---
\r\n");
   String URLname = getFileNameURL(requestH.Method);
   String conURL = ConvertURL(URLname);
   File StoreFile = new File(pathToCache+conURL);

   //work 1
   hit=0;
   if(StoreFile.exists()&&!requestH.noUseCache)
   {
    sendCacheFile(outToClient,StoreFile);
    CachePage.textArea1.append("Object exist,Cache
Hit\r\n");
    System.out.println("cache hit");
    hit=1;
```

```
}else
{
 System.out.println("cache miss");
 if(StoreFile.exists()){
  FPage.size = FPage.size - StoreFile.length();
  System.out.println("all size:"+FPage.size);
 }
 try
 {
  if(KeepAlive)
  {
   socket = LastSocket;
   dos = LastDOS;
   dis = LastDIS;
   CachePage.textArea1.append("Keep Alive
Loop\r\n");
  }else
  {
   socket = new Socket(requestH.Host,requestH.Port);
   dos = new
DataOutputStream(socket.getOutputStream());
   dis = new DataInputStream(socket.getInputStream());
  }
  CachePage.textArea1.append("socket to
web:"+socket.getLocalPort()+
                    "\r\n");
  // send request to www server
  traffic=0;
  if(FPage.ConF.jTextField_Port.getText().startsWith(
                Integer.toString(requestH.Port)))
  {
   traffic = traffic + requestHeader.length();
  }
  dos.writeBytes(requestHeader.toString());
  dos.flush();

  // if(HaveDataToSent)
  if(requestH.Content_Length >0 )//if have data to sent
  {
   CachePage.textArea1.append("---sent request data---
\r\n");
   while (requestH.Content_Length>0)
   {
    //read data from client and decrease data length until
0

requestH.Content_Length=requestH.Content_Length-
                    inFromClient.read(buff);

CachePage.textArea1.append(requestH.Content_Length+"\r\n"
);
    if(FPage.ConF.jTextField_Port.getText().startsWith(
                    Integer.toString(requestH.Port)))
    {
     traffic = traffic + buff.length + EOL.length;
    }
    dos.write(buff);
    dos.flush();
   }
   dos.write(EOL);
   CachePage.textArea1.append("---end request data---
\r\n");
  }
  // read object from www server
  int num;
  replyHeader replyH = new replyHeader();
  ByteArrayOutputStream tempBuffer = new
                ByteArrayOutputStream();
  StringBuffer replyHeader = new StringBuffer();
  CachePage.textArea1.append("---reply header---\r\n");
  replyH.Content_Length = -1;//clear value of
Content_Length
  int ct;
  String readlineH;
  if((conURL.endsWith(".php"))||(conURL.endsWith
(".css"))
    ||(conURL.endsWith(".pl"))||(conURL.endsWith
(".js"))
    ||(conURL.endsWith(".ashx")))
```

```
        replyH.nocache = true;
      if((readlineH=dis.readLine())!=null)
      {
       do
       {//receive header from Webserver
        replyHeader.append(readlineH+"\r\n");
        if(readlineH.startsWith("HTTP/"))
        {
         replyH.Status = readlineH;
         String stTmp = readlineH.substring
(readlineH.indexOf(" ")+1);
         //if reply status error code 3xx,4xx,5xx
         if(stTmp.startsWith("3")||stTmp.startsWith("4")||
           stTmp.startsWith("5"))
           replyH.nocache = true;
        }
        if(readlineH.startsWith("Expires:"))
        {
         String stTmp = readlineH.substring(9);
         if(stTmp.startsWith("0")||stTmp.startsWith("now"))
         {
          replyH.nocache = true;
         }else
         {//test Expired Date with Now Date for Caching
          stTmp = stTmp.substring(stTmp.indexOf(",")+2);
          if(testDate(stTmp)==true)replyH.nocache=true;
         }
        }
        if((readlineH.startsWith("Pragma:"))||
          (readlineH.startsWith("Cache-Control:")))
        {
         readlineH = readlineH.substring(readlineH.indexOf
(" ")+1);
         if(readlineH.startsWith("no-cache")||
           readlineH.startsWith("no-store")||
           readlineH.endsWith("no-cache"))
            replyH.nocache=true;
        }
        if(readlineH.startsWith("Content-Length:"))
        {
         replyH.Content_Length = Integer.decode(
            readlineH.substring(16).trim()).intValue
();//test
         if(requestH.Method.startsWith("HEAD "))
          replyH.Content_Length = 0;
        }
        if(readlineH.startsWith("Proxy-Connection:"))
        {
         readlineH = readlineH.substring(18).trim();
         if(readlineH.startsWith("keep-alive"))
         {
          KeepAlive = false;
         }else
         {
          KeepAlive = false;
         }
        }while((readlineH=dis.readLine()).length()!=0);
        CachePage.textArea1.append(replyHeader+"\r\n");
        if(FPage.ConF.jTextField_Port.getText().startsWith(
                    Integer.toString(requestH.Port)))
        {
         traffic = traffic + replyHeader.length()+EOL.length;
        }
        outToClient.print(replyHeader.toString()+"\r\n");
        outToClient.flush();
        CachePage.textArea1.append("---end reply header---
\r\n");
        CachePage.textArea1.append("---reply data---\r\n");
        if(!replyH.nocache&&!requestH.ReqFromCache)
        {//save to file
         if(FPage.size>cachesize)deleteCache();
         CachePage.textArea1.append("Store Object To:"+
                    StoreFile.getAbsoluteFile
()+"\r\n");
         FileOutputStream saveToFile = new
FileOutputStream(StoreFile);
         //save reply header to file
         saveToFile.write(("LFU=0\r\n").getBytes());
         saveToFile.write(replyHeader.toString().getBytes());
```

```
        saveToFile.write(EOL);
        num=0;
        if((replyH.Content_Length == -1)
         &&(!replyH.Status.substring(9).startsWith
(HTTP_NOT_MODIFIED)))
        //if don't have Content-Length:Header
        { //but have data to sent
         while(((num=dis.read(buff))>0)&&
(!outToClient.checkError()))//
          {
           if(FPage.ConF.jTextField_Port.getText
().startsWith(
                        Integer.toString(requestH.Port)))
           {
            traffic = traffic + num;
           }
           outToClient.write(buff,0,num);
           CachePage.textArea1.append("receive and cache:
"+num+
                        " bytes\r\n");
           outToClient.flush();
           saveToFile.write(buff,0,num);
          }
          CachePage.textArea1.append("receive and cache:
"+num+
                        " bytes\r\n");
        }
        //have Content-Length:Header
        while ((replyH.Content_Length>0)&&
(!outToClient.checkError()))
        {
         num = dis.read(buff);
         replyH.Content_Length = replyH.Content_Length -
num;
         CachePage.textArea1.append("receive and cache:
"+num+
                " bytes /"+replyH.Content_Length+"
bytes\r\n");
         saveToFile.write(buff,0,num);
         if(FPage.ConF.jTextField_Port.getText
().startsWith(
                        Integer.toString(requestH.Port)))
         {
          traffic = traffic + num;
         }
         outToClient.write(buff,0,num);
         outToClient.flush();
        }
        outToClient.write(EOL);
        saveToFile.close();
        if(outToClient.checkError())
        {
         StoreFile.delete();
        }
        }else
        {//not save to file
        if((replyH.Content_Length == -1)
         &&(!replyH.Status.substring(9).startsWith
(HTTP_NOT_MODIFIED)))
         //if don't have Content-Length:Header
         { //but have data to sent
          while(((num=dis.read(buff))>0)&&
(!outToClient.checkError()))//
          {
           if(FPage.ConF.jTextField_Port.getText
().startsWith(
                        Integer.toString(requestH.Port)))
           {
            traffic = traffic + num;
           }
           outToClient.write(buff,0,num);          //
           CachePage.textArea1.append("receive : "+num+"
bytes\r\n");
           outToClient.flush();
          }                                        //
          CachePage.textArea1.append("receive : "+num+"
bytes\r\n");
          outToClient.print("\r\n");
         }
```

```
        //have Content-Length:Header
        while ((replyH.Content_Length>0)&&
(!outToClient.checkError()))
        {
         num = dis.read(buff);
         replyH.Content_Length = replyH.Content_Length -
num;
         CachePage.textArea1.append("receive: "+num+"
bytes /"+
                        replyH.Content_Length+"
bytes\r\n");
         if(FPage.ConF.jTextField_Port.getText
().startsWith(
                        Integer.toString(requestH.Port)))
         {
          traffic = traffic + num;
         }
         outToClient.write(buff,0,num);
         outToClient.flush();
        }
       }
       outToClient.write(EOL);outToClient.write
(EOL);outToClient.write(EOL);
       CachePage.textArea1.append("---end reply data---
\r\n");
       CachePage.textArea1.append("already reply to
client\r\n");
      }else
      {
       System.out.println("Null Pointer From Web");
       FPage.textArea_Event.append("Null Pointer From
Web!!!");
       outToClient.println("HTTP/1.0 404 Not Fount");
      }
      System.out.println("traffic ="+traffic);
      LastSocket = socket;
      LastDOS = dos;
      LastDIS = dis;
      LastHost = requestH.Host;
      if(!KeepAlive)socket.close();
       if(StoreFile.exists()){
        System.out.println("length:"+StoreFile.length());
        FPage.size = FPage.size + StoreFile.length();
        System.out.println("all size:"+FPage.size);
       }
      }catch(IOException e)
      {
       CachePage.textArea1.append("Connection to Webpage
Error or "+
                        "Save to File Error\r\n");
       }
      }
      try{s.close();}catch(IOException e){}
      synchronized(ClientTab)
      {
       ClientTab.removeElement(s.getInetAddress());
      }
      CachePage.textArea1.append("closed Socket\r\n");
      now = new Date();
      FPage.CacheHitF.textArea1.append(threadName+"
response: "+
        (now.getTime()-restime)+ " msec.\r\n");
      FPage.CacheHitF.textArea1.append(threadName+
        " creat traffic between node: "+traffic+" bytes\r\n");
      System.out.println(now.getTime());
      FPage.hitRatio = ((FPage.hitRatio*FPage.allReq)+
(hit*100))/
            (FPage.allReq+1);
      FPage.allReq++;
      FPage.CacheHitF.textArea1.append(
            "CacheHit Ratio : "+FPage.hitRatio+"\r\n");
      s = null;
      Vector pool = CacheServerApp.threads;
      CachePage.textArea1.append("release thread\r\n");
      FPage.textArea_Event.append(threadName+" :
Released\r\n\r\n");
     }//end check s=null for stop server
    }while(true&&!FPage.StopServer);
    CachePage.hide();
```

```
  CachePage.dispose();
  FPage.textArea_Event.append(threadName+" : Close
Process.\r\n");
 }//End run CacheWorker


 void sendCacheFile(PrintStream out,File file)
 {
  try
  {
   DataInputStream in = new DataInputStream(new
FileInputStream(file));
   byte buffer[] = new byte[9192];
   int n;
   Thread.sleep(100);
   String LFU = in.readLine();
   while ((n = in.read(buffer)) > 0) {
    out.write(buffer, 0, n);
    System.out.println(n);
   }
   out.flush();
   out.print("\r\n\r\n");
   out.close();
   in.close();

   DataInputStream inFile = new DataInputStream(new
FileInputStream(file));
   int len = (int) file.length();
   byte[] buffFull = new byte[len-LFU.length()-2];
   int iLFU = Integer.decode(LFU.substring(4)).intValue();
   inFile.readLine();
   inFile.readFully(buffFull,0,len-LFU.length()-2);
   LFU = "LFU="+(iLFU+1)+"\r\n";
   byte[] bLFU = LFU.getBytes();
   FileOutputStream saveToFile = new FileOutputStream
(file);
   saveToFile.write(bLFU);
   saveToFile.write(buffFull);
   saveToFile.close();
   inFile.close();
  }
  catch(Exception e)
  {
   CachePage.textArea1.append("Error to retrieve Cache
file.\r\n");
   System.out.println("Error retrieving file.");
   System.exit(1);
  }
 }


 public void deleteCache()
 {
  File path = new
File(FPage.ConF.jTextField_pathToCache.getText());
  File[] list = path.listFiles();
  int cachesize = (int)(Double.valueOf(
          FPage.ConF.jTextField_CacheSize.getText
()).doubleValue()
          *1000000);
  int x;
  for(x=0;FPage.size>cachesize;x++)
  {
   for(int y=0;y<list.length;y++)
   {
    if(list[y].isFile())
    {
     try
     {
     DataInputStream in = new DataInputStream(
                new FileInputStream(list[y]));
     String tmpread = in.readLine().substring(4);
     in.close();
     int itmread = Integer.decode(tmpread).intValue();
     if(itmread == x)
     {
      System.out.println(x);
      FPage.size = FPage.size - list[y].length();
      System.out.println(FPage.size);
      System.out.println(list[y].delete());
     }
    }catch(IOException e){}
```

```
    }
   }
  }
  list = path.listFiles();
  for(int i=0;i<list.length;i++)
  {
   try
   {
    DataInputStream inFile = new DataInputStream(
                    new FileInputStream(list[i]));

    String LFU = inFile.readLine();
    int len = (int) list[i].length();
    byte[] buffFull = new byte[len-LFU.length()-2];
    int iLFU = Integer.decode(LFU.substring(4)).intValue();
    inFile.readFully(buffFull,0,len-LFU.length()-2);
    LFU = "LFU="+(iLFU-x+1)+"\r\n";
    byte[] bLFU = LFU.getBytes();
    FileOutputStream saveToFile = new FileOutputStream
(list[i]);
    saveToFile.write(bLFU);
    saveToFile.write(buffFull);
    saveToFile.close();
    inFile.close();
   }catch(IOException e){}
  }
 }
}

class CheckStatus implements Runnable
{
 Vector Tab = new Vector();
 MenuFrame FPage;
 DatagramSocket StatusSocket;

 public CheckStatus(MenuFrame FPage)
 {
  this.FPage = FPage;
 }

 public synchronized void run()
 {
  String status;
  byte[] receiveData = new byte[50];
  byte[] sendData = new byte[50];
  FPage.textArea_Event.append("Check Status Process:
ON\r\n");
  try
  {
   StatusSocket = new DatagramSocket(9876);

   while(true&&!FPage.StopServer)
   {
    DatagramPacket receiveStatus = new DatagramPacket
(receiveData,
                    receiveData.length);
    StatusSocket.receive(receiveStatus);
    status = new String(receiveStatus.getData());
    status = status.trim();
    if(status.startsWith("Status CacheServerApp/1.0"))
    {
     status = status.substring(27);
     if(!status.startsWith(InetAddress.getLocalHost
().toString()))
     {//if status not reply by itself
      if(Tab.size()==0)
      {
       Tab.addElement(status);
      }else
      {
       boolean insert=true;
       for(int i=0;i<Tab.size();i++)
       {
        if(status.startsWith(Tab.elementAt(i).toString()))
        {
         insert = false;
        }
       }
       if(insert)
       {
```

```
            Tab.addElement(status);
          }
        }
        CacheServerApp.StatusTab = Tab;
      }
    }else if(status.startsWith("requestStatus
CacheServerApp/1.0"))
    {
      String replyStatus = "Alive";
      sendData = replyStatus.getBytes();
      InetAddress IPAddress = receiveStatus.getAddress();
      int Port = receiveStatus.getPort();

      DatagramPacket sendStatus = new DatagramPacket
(sendData,sendData.length,
                          IPAddress,Port);
      StatusSocket.send(sendStatus);
    }
   }//end while loop
  }catch(IOException e)
  {
   FPage.textArea_Event.append("Check Status Socket Close
or Error\r\n");
  }
  StatusSocket.close();
  FPage.textArea_Event.append("Check Status Process:
OFF\r\n");
 }
}

class SendStatus implements Runnable
{
 MenuFrame FPage;

 public SendStatus(MenuFrame FPage)
 {
  this.FPage = FPage;
 }

 public void run()
 {
  FPage.textArea_Event.append("Send Status Process:
ON\r\n");
  byte[] sendData = new byte[50];
  try
  {
   DatagramSocket StatusSocket = new DatagramSocket();
   InetAddress BroadCastAddress = InetAddress.getByName
("255.255.255.255");

   while(true&&!FPage.StopServer)
   {
    String status="";
    status = "Status CacheServerApp/1.0\r\n"+
             InetAddress.getLocalHost().toString()+"\r\n";
    sendData = status.getBytes();
    DatagramPacket sendStatus = new DatagramPacket
(sendData,sendData.length,
                    BroadCastAddress,9876);
    StatusSocket.send(sendStatus);
    try{Thread.sleep(25000);}catch(InterruptedException e)
{}
   }
   }catch(IOException e)
   {
    FPage.textArea_Event.append("Send Status Socket
Error\r\n");
   }
  }
 }

class VerifyStatus implements Runnable
{
 MenuFrame FPage;
 public VerifyStatus(MenuFrame FPage)
 {
  this.FPage = FPage;
 }

 public void run()
```

```
{
  byte[] receiveData = new byte[50];
  byte[] sendData = new byte[50];

  FPage.textArea_Event.append("Verify Status Process :
ON\r\n");
  while(true)
  {
   Vector Tab = CacheServerApp.StatusTab;
   //clear textArea of cache connection
   FPage.jTextArea_CacheConnect.setText("");
   FPage.ConF.jComboBox_StaticPath.removeAllItems();
   if(Tab.size()!=0)
    FPage.ConF.jCheckBox1.setEnabled(true);
   else
    FPage.ConF.jCheckBox1.setEnabled(false);

   for(int i=0;i<Tab.size();i++)
   {
    try
    {
     DatagramSocket StatusSocket = new DatagramSocket();
     String sendToCheck = "requestStatus
CacheServerApp/1.0";
     sendData = sendToCheck.getBytes();
     FPage.textArea_Event.append("Alive :
"+Tab.elementAt(i)+"\r\n");
     FPage.jTextArea_CacheConnect.append(Tab.elementAt
(i)+"\r\n");
     String tempElement = Tab.elementAt(i).toString();
     tempElement = tempElement.substring
(tempElement.indexOf("/")+1);
     InetAddress IPAddress = InetAddress.getByName
(tempElement);

     FPage.ConF.jComboBox_StaticPath.addItem
(Tab.elementAt(i));

     DatagramPacket checkStatus = new DatagramPacket
(sendData,
                    sendData.length,IPAddress,9876);
     StatusSocket.send(checkStatus);
     DatagramPacket receiveStatus = new DatagramPacket
(receiveData,
                    receiveData.length);
     StatusSocket.receive(receiveStatus);
     StatusSocket.close();
    }catch(IOException e)
    {
     FPage.textArea_Event.append("Cache don't reply:\r\n"+
      "delete connection : "+Tab.elementAt(i)+"\r\n");
     synchronized(Tab)
     {
      Tab.removeElementAt(i);
     }
     FPage.jTextArea_CacheConnect.setText("");
     for(i=0;i<Tab.size();i++)
     {
      FPage.jTextArea_CacheConnect.append
(Tab.elementAt(i)+"\r\n");
     }
    }
    try{Thread.sleep(50000);}catch(InterruptedException e)
    {
    FPage.textArea_Event.append("Verify Process Socket
Error");
    }
   }
  }
}

class cacheManagement extends CacheManager implements
Runnable
{
 MenuFrame FPage;
 cacheManagement(MenuFrame FPage)
 {
  this.FPage = FPage;
```

```
 }

 public void run()
 {
  while(true&&!FPage.StopServer)
  {
   try{Thread.sleep(3600000);}catch(InterruptedException e)
{}
   System.out.println("management loop");
   byte[] tempread = new byte[400];
   File path = new
File(FPage.ConF.jTextField_pathToCache.getText());
   File[] list = path.listFiles();

   for(int i = 0;i<list.length;i++)
   {
    String readin;
    try
    {
     DataInputStream in = new DataInputStream(new
FileInputStream(list[i]));

     while((readin=in.readLine()).length()!=0)
     {
      if(readin.startsWith("Expires:"))
      {
       readin = readin.substring(14);
       if(testDate(readin))
       {
        FPage.size = FPage.size-list[i].length();
        in.close();
        list[i].delete();
        System.out.println("delete");
       }
      }
     }
     in.close();
    }catch(IOException e){}
   }
  }
 }
}

class requestHeader
{
 String Host="",Method="",Ref="";
 int Content_Length=-1,Port=80;
 boolean ReqFromCache = false,noUseCache = false;
}

class replyHeader
{
 String Status="";
 int Content_Length=-1;
 boolean nocache=false;
}

interface HttpConstants {
/* 2XX: generally "OK" */
 public static final String HTTP_OK = "200";
 public static final int HTTP_CREATED = 201;
 public static final int HTTP_ACCEPTED = 202;
 public static final int HTTP_NOT_AUTHORITATIVE =
203;
 public static final int HTTP_NO_CONTENT = 204;
 public static final int HTTP_RESET = 205;
 public static final String HTTP_PARTIAL = "206";

/** 3XX: relocation/redirect */
 public static final int HTTP_MULT_CHOICE = 300;
 public static final int HTTP_MOVED_PERM = 301;
 public static final String HTTP_MOVED_TEMP = "302";
 public static final String HTTP_SEE_OTHER = "303";
 public static final String HTTP_NOT_MODIFIED = "304";
 public static final int HTTP_USE_PROXY = 305;

/** 4XX: client error */
 public static final String HTTP_BAD_REQUEST = "400";
 public static final int HTTP_UNAUTHORIZED = 401;
```

```
public static final int HTTP_PAYMENT_REQUIRED = 402;
public static final int HTTP_FORBIDDEN = 403;
public static final String HTTP_NOT_FOUND = "404";
public static final int HTTP_BAD_METHOD = 405;
public static final int HTTP_NOT_ACCEPTABLE = 406;
public static final int HTTP_PROXY_AUTH = 407;
public static final int HTTP_CLIENT_TIMEOUT = 408;
public static final int HTTP_CONFLICT = 409;
public static final int HTTP_GONE = 410;
public static final int HTTP_LENGTH_REQUIRED = 411;
public static final int HTTP_PRECON_FAILED = 412;
public static final int HTTP_ENTITY_TOO_LARGE = 413;
public static final int HTTP_REQ_TOO_LONG = 414;
public static final int HTTP_UNSUPPORTED_TYPE = 415;

/** 5XX: server error */
public static final int HTTP_SERVER_ERROR = 500;
public static final int HTTP_INTERNAL_ERROR = 501;
public static final int HTTP_BAD_GATEWAY = 502;
public static final int HTTP_UNAVAILABLE = 503;
public static final int HTTP_GATEWAY_TIMEOUT = 504;
public static final int HTTP_VERSION = 505;
}
```

MenuFrame.java

```
package CacheServerApp;

import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import com.borland.jbcl.layout.*;
import javax.swing.border.*;
import java.beans.*;
import javax.swing.event.*;
import java.io.*;

public class MenuFrame extends JFrame {
  int menu_w,menu_h;
  boolean StopServer = false;
  long size = 0;
  long restime=0;
  long allReq=100;
  double hitRatio=60.0;

  // int MaxLRU = 0;
  String WorkerName;
  CFrame CachePage;
  Vector threads;
  CacheManager cm;

  JPanel contentPane;
  JMenuBar jMenuBar1 = new JMenuBar();
  JMenu jMenuFile = new JMenu();
  JMenuItem jMenuFileExit = new JMenuItem();
  JMenu jMenuHelp = new JMenu();
  JMenuItem jMenuHelpAbout = new JMenuItem();

  ConfigFrame ConF = new ConfigFrame(this);
  CFrame CacheHitF = new CFrame();

  TitledBorder titledBorder1;
  TitledBorder titledBorder2;
  TitledBorder titledBorder3;
  JPanel jPanel1 = new JPanel();
  TextArea textArea_Event = new TextArea();
  VerticalFlowLayout verticalFlowLayout1 = new
VerticalFlowLayout();
  JLabel jLabel1 = new JLabel();
  JPanel jPanel2 = new JPanel();
  BorderLayout borderLayout2 = new BorderLayout();

JSplitPane jSplitPane1 = new JSplitPane();
  GridLayout gridLayout1 = new GridLayout();
  JPanel jPanel3 = new JPanel();
  JMenuItem jMenuItem_Start = new JMenuItem();
  JMenuItem jMenuItem_Stop = new JMenuItem();
  JSplitPane jSplitPane2 = new JSplitPane();
  TitledBorder titledBorder4;
  BorderLayout borderLayout1 = new BorderLayout();
  JPanel jPanel4 = new JPanel();
  JLabel jLabel_Connection = new JLabel();
  JTextArea jTextArea_Connection = new JTextArea();
  BorderLayout borderLayout3 = new BorderLayout();
  JPanel jPanel5 = new JPanel();
  BorderLayout borderLayout4 = new BorderLayout();
  JLabel jLabel_Status = new JLabel();
  JTextArea jTextArea_Status = new JTextArea();
  JSplitPane jSplitPane3 = new JSplitPane();
  JPanel jPanel6 = new JPanel();
  JLabel jLabel4 = new JLabel();
  BorderLayout borderLayout5 = new BorderLayout();
  JTextArea jTextArea_LocalCache = new JTextArea();
  JPanel jPanel7 = new JPanel();
  JLabel jLabel5 = new JLabel();
  BorderLayout borderLayout6 = new BorderLayout();
  JTextArea jTextArea_CacheConnect = new JTextArea();
  JMenu jMenu2 = new JMenu();
  JCheckBoxMenuItem jCheckBoxMenuItem_MWorker =
new JCheckBoxMenuItem();
  JMenuItem jMenuItem1 = new JMenuItem();
```

```
 JCheckBoxMenuItem jCheckBoxMenuItem_Performance =
new JCheckBoxMenuItem();
 Dialog_About dA = new Dialog_About(this,"About",true);

 /**Construct the frame*/
 public MenuFrame() {
  enableEvents(AWTEvent.WINDOW_EVENT_MASK);
  try {
   jbInit();
  }
  catch(Exception e) {
   e.printStackTrace();
  }
 }

 public void inCacheIndex(Vector threads)
 {
  this.threads = threads;
 }
 /**Component initialization*/
 private void jbInit() throws Exception  {
  //setIconImage(Toolkit.getDefaultToolkit().createImage
(Frame1.class.getResource("[Your Icon]")));
  contentPane = (JPanel) this.getContentPane();
  titledBorder1 = new TitledBorder("");
  titledBorder2 = new TitledBorder("");
  titledBorder3 = new TitledBorder(new MatteBorder
(null),"");
  titledBorder4 = new TitledBorder("");
  this.setSize(new Dimension(640, 480));

  menu_w = (int)getSize().getWidth();
  menu_h = (int)getSize().getHeight();
  CacheHitF.setTitle("Performanace");

  this.setTitle("Cache Server");
  jMenuFile.setText("Action");
  jMenuFileExit.setText("Exit");
  jMenuFileExit.addActionListener(new ActionListener()  {
   public void actionPerformed(ActionEvent e) {
    jMenuFileExit_actionPerformed(e);
   }
  });
  jMenuHelp.setText("Help");
  jMenuHelpAbout.setText("About");
  jMenuHelpAbout.addActionListener(new
java.awt.event.ActionListener() {
   public void actionPerformed(ActionEvent e) {
    jMenuHelpAbout_actionPerformed(e);
   }
  });
  contentPane.setLayout(borderLayout2);
  jPanel1.setLayout(verticalFlowLayout1);
  textArea_Event.setEditable(false);
  jLabel1.setText("Event ");
  jPanel2.setLayout(gridLayout1);
  jPanel1.setBorder(BorderFactory.createLineBorder
(Color.black));
  jSplitPane1.setContinuousLayout(true);
  jPanel3.setLayout(borderLayout1);
  jMenuItem_Start.setText("Start");
  jMenuItem_Start.addActionListener(new
java.awt.event.ActionListener() {
   public void actionPerformed(ActionEvent e) {
    jMenuItem_Start_actionPerformed(e);
   }
  });
  jMenuItem_Stop.setEnabled(false);
  jMenuItem_Stop.setText("Stop");
  jMenuItem_Stop.addActionListener(new
java.awt.event.ActionListener() {
   public void actionPerformed(ActionEvent e) {
    jMenuItem_Stop_actionPerformed(e);
   }
  });
  jSplitPane2.setBackground(Color.white);
  jSplitPane2.setForeground(Color.white);
  jSplitPane2.setBorder(null);
  jSplitPane2.setContinuousLayout(true);
  jSplitPane2.setDividerSize(1);
```

```
    jLabel_Connection.setHorizontalAlignment
(SwingConstants.CENTER);
    jLabel_Connection.setText("Client Connection");
    jTextArea_Connection.setEditable(false);
    jPanel4.setLayout(borderLayout3);
    jPanel5.setLayout(borderLayout4);
    jLabel_Status.setToolTipText("");

jLabel_Status.setHorizontalAlignment(SwingConstants.CENT
ER);
    jLabel_Status.setText("Status");
    jTextArea_Status.setEditable(false);
    jSplitPane3.setOrientation(JSplitPane.VERTICAL_SPLIT);

jLabel4.setHorizontalAlignment(SwingConstants.CENTER);
    jLabel4.setText("LocalCache");
    jPanel6.setLayout(borderLayout5);
    jTextArea_LocalCache.setNextFocusableComponent(this);
    jTextArea_LocalCache.setToolTipText("");
    jTextArea_LocalCache.setEditable(false);

jLabel5.setHorizontalAlignment(SwingConstants.CENTER);
    jLabel5.setText("Cache Connect");
    jPanel7.setLayout(borderLayout6);
    jTextArea_CacheConnect.setEditable(false);

jPanel6.setBorder(BorderFactory.createLoweredBevelBorder
());

jPanel7.setBorder(BorderFactory.createLoweredBevelBorder
());

jPanel3.setBorder(BorderFactory.createLoweredBevelBorder
());
    jMenu2.setText("Monitor");
    jCheckBoxMenuItem_MWorker.setEnabled(false);
    jCheckBoxMenuItem_MWorker.setBorder(null);
    jCheckBoxMenuItem_MWorker.setText("Worker");
    jCheckBoxMenuItem_MWorker.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
       jCheckBoxMenuItem_MWorker_actionPerformed(e);
      }
    });
    jMenuItem1.setText("Config");
    jMenuItem1.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
       jMenuItem1_actionPerformed(e);
      }
    });
    jCheckBoxMenuItem_Performance.setEnabled(false);
    jCheckBoxMenuItem_Performance.setText
("Performance");
    jCheckBoxMenuItem_Performance.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
       jCheckBoxMenuItem_Performance_actionPerformed(e);
      }
    });
    jMenuFile.add(jMenuItem_Start);
    jMenuFile.add(jMenuItem_Stop);
    jMenuFile.addSeparator();
    jMenuFile.add(jMenuItem1);
    jMenuFile.addSeparator();
    jMenuFile.add(jMenuFileExit);
    jMenuHelp.add(jMenuHelpAbout);
    jMenuBar1.add(jMenuFile);
    jMenuBar1.add(jMenu2);
    jMenuBar1.add(jMenuHelp);
    contentPane.add(jPanel1, BorderLayout.SOUTH);
    jPanel1.add(jLabel1, null);
    jPanel1.add(textArea_Event, null);
    contentPane.add(jPanel2, BorderLayout.CENTER);
    jPanel2.add(jSplitPane1, null);
    jSplitPane1.add(jPanel3, JSplitPane.RIGHT);
    jPanel3.add(jSplitPane2, BorderLayout.CENTER);
    jSplitPane2.add(jPanel4, JSplitPane.LEFT);
    jPanel4.add(jTextArea_Connection,
BorderLayout.CENTER);
```

```
    jPanel4.add(jLabel_Connection, BorderLayout.NORTH);
    jSplitPane2.add(jPanel5, JSplitPane.RIGHT);
    jPanel5.add(jLabel_Status, BorderLayout.NORTH);
    jPanel5.add(jTextArea_Status, BorderLayout.CENTER);
    jSplitPane1.add(jSplitPane3, JSplitPane.LEFT);
    jSplitPane3.add(jPanel6, JSplitPane.TOP);
    jPanel6.add(jLabel4, BorderLayout.NORTH);
    jPanel6.add(jTextArea_LocalCache,
BorderLayout.CENTER);
    jSplitPane3.add(jPanel7, JSplitPane.BOTTOM);
    jPanel7.add(jLabel5, BorderLayout.NORTH);
    jPanel7.add(jTextArea_CacheConnect,
BorderLayout.CENTER);
    jMenu2.add(jCheckBoxMenuItem_MWorker);
    jMenu2.add(jCheckBoxMenuItem_Performance);
    this.setJMenuBar(jMenuBar1);



    if(StopServer)textArea_Event.append("Server Status :
Active.\r\n");
    else textArea_Event.append("Server Status : not
Active.\r\n");
  }
  /**File | Exit action performed*/
  public void jMenuFileExit_actionPerformed(ActionEvent e)
{
    System.exit(0);
  }
  /**Help | About action performed*/
  /**Overridden so we can exit when window is closed*/
  protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      jMenuFileExit_actionPerformed(null);
    }
  }



  void jMenuItem_Start_actionPerformed(ActionEvent e) {
    size = getSizeCache();
    jCheckBoxMenuItem_MWorker.setEnabled(true);
    jCheckBoxMenuItem_Performance.setEnabled(true);
    textArea_Event.append("Server Status : Active.\r\n");
    cm = new CacheManager();
    cm.SetFrame(this);
    new Thread(cm, "cache manager").start();
    StopServer = false;
    jMenuItem_Stop.setEnabled(true);
    jMenuItem_Start.setEnabled(false);
  }

  void jMenuItem_Stop_actionPerformed(ActionEvent e) {
    StopServer = true;
    cm.stop();
    for(int i=0;i<threads.size();i++)
    {
      CacheWorker cw = (CacheWorker)threads.elementAt(i);
      if(!cw.isProcess)cw.stop();
    }
    jMenuItem_Start.setEnabled(true);
    jMenuItem_Stop.setEnabled(false);
    jCheckBoxMenuItem_MWorker.setEnabled(false);
    jCheckBoxMenuItem_MWorker.setSelected(false);
    jCheckBoxMenuItem_Performance.setEnabled(false);
    jCheckBoxMenuItem_Performance.setSelected(false);
    CacheHitF.hide();
    CacheHitF.dispose();
    textArea_Event.append("Server Status : not Active.\r\n");
  }

  void jCheckBoxMenuItem_MWorker_actionPerformed
(ActionEvent e) {
    if(jCheckBoxMenuItem_MWorker.isSelected())
    {
      for(int i=0;i<threads.size();i++)
      {
        CacheWorker cw = (CacheWorker)threads.elementAt(i);
        cw.CachePage.show();
      }
    }else
```

```
  {
   for(int i=0;i<threads.size();i++)
   {
    CacheWorker cw = (CacheWorker)threads.elementAt(i);
    cw.CachePage.hide();
    cw.CachePage.dispose();
   }
  }
 }

 long getSizeCache()
 {
  long size=0;
  File path = new File(ConF.jTextField_pathToCache.getText
());
  File[] list = path.listFiles();
  for(int y=0;y<list.length;y++)
  {
   if(list[y].isFile())
   {
    size = size + list[y].length();
   }
  }
  return size;
 }

 void jMenuItem1_actionPerformed(ActionEvent e) {
  this.disable();
  ConF.show();
 }

 void jCheckBoxMenuItem_Performance_actionPerformed
(ActionEvent e) {
  if(jCheckBoxMenuItem_Performance.isSelected())
  {
   CacheHitF.show();
  }else
  {
   CacheHitF.hide();
   CacheHitF.dispose();
  }
 }

 void jMenuHelpAbout_actionPerformed(ActionEvent e) {
  dA.setResizable(false);
  dA.show();
 }
}
```

ConfigFrame.java

```
package CacheServerApp;

import java.awt.*;
import com.borland.jbcl.layout.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.border.*;
import javax.swing.event.*;
import java.io.*;

public class ConfigFrame extends JFrame{
 MenuFrame mainFrame;
 boolean State_CB_Static_Path = false;
 boolean State_CB_Single = false;
 int selectPath=0;
 String oldPathToCatch,oldStaticPath,oldPort,oldWorker;


 XYLayout xYLayout1 = new XYLayout();
 JButton jButton_OK = new JButton();
 JButton jButton_Cancal = new JButton();
 JTabbedPane jTabbedPane1 = new JTabbedPane();
 JPanel jPanel1 = new JPanel();
 JPanel jPanel2 = new JPanel();
 JLabel jLabel1 = new JLabel();
 XYLayout xYLayout2 = new XYLayout();
 JTextField jTextField_Port = new JTextField();
 JLabel jLabel2 = new JLabel();
 JTextField jTextField_pathToCache = new JTextField();
 JLabel jLabel3 = new JLabel();
 XYLayout xYLayout3 = new XYLayout();


 TitledBorder titledBorder1;
 JCheckBox jCheckBox1 = new JCheckBox();
 JPanel jPanel3 = new JPanel();
 BoxLayout2 boxLayout21 = new BoxLayout2();
 JCheckBox jCheckBox2 = new JCheckBox();
 JLabel jLabel4 = new JLabel();
 JButton jButton4 = new JButton();
 JTextField jTextField_worker = new JTextField();
 JLabel jLabel_CacheSize = new JLabel();
 JTextField jTextField_CacheSize = new JTextField();
 JComboBox jComboBox_StaticPath = new JComboBox();

 public ConfigFrame(MenuFrame mainFrame) {
 this.mainFrame = mainFrame;
  try {
   jbInit();
  }
  catch(Exception e) {
   e.printStackTrace();
  }
 }

  private void jbInit() throws Exception {
  titledBorder1 = new TitledBorder("");
  jButton_OK.setText("OK");
  jButton_OK.addActionListener(new
 java.awt.event.ActionListener() {
   public void actionPerformed(ActionEvent e) {
    jButton_OK_actionPerformed(e);
   }
  });
  this.setResizable(false);
  this.setSize(new Dimension(400, 300));
  this.setTitle("Configulation");
  this.addWindowListener(new
 java.awt.event.WindowAdapter() {
    public void windowClosing(WindowEvent e) {
     this_windowClosing(e);
    }
  });
  this.getContentPane().setLayout(xYLayout1);
  jButton_Cancal.setText("Cancal");
  jButton_Cancal.addActionListener(new
 java.awt.event.ActionListener() {
```

```
    public void actionPerformed(ActionEvent e) {
      jButton_Cancal_actionPerformed(e);
    }
  });
  jLabel1.setText("Port");
  jPanel1.setLayout(xYLayout2);
  jTextField_Port.setText("4321");
  jLabel2.setText("path to catch");
  jTextField_pathToCache.setText("d:\\cachefile\\");
  jLabel3.setText("Cache Worker");
  jPanel2.setLayout(xYLayout3);
  jCheckBox1.setEnabled(false);
  jCheckBox1.setText("Static Path");
  jCheckBox1.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
      jCheckBox1_actionPerformed(e);
    }
  });
  jPanel3.setLayout(boxLayout21);
  jCheckBox2.setText("Single Cache Mode");
  jCheckBox2.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
      jCheckBox2_actionPerformed(e);
    }
  });
  jLabel4.setText("Clear Cache Item");
  jButton4.setText("Clear");
  jButton4.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
      jButton4_actionPerformed(e);
    }
  });
  jTextField_worker.setText("5");
  jLabel_CacheSize.setText("Cache Size (Mb)");
  jTextField_CacheSize.setText("10");
  jComboBox_StaticPath.setEnabled(false);
    this.getContentPane().add(jTabbedPane1, new
XYConstraints(18, 16, 350, 214));
    jTabbedPane1.add(jPanel1, "General");
    jPanel1.add(jLabel1, new XYConstraints(12, 19, -1, -1));
    jPanel1.add(jTextField_Port, new XYConstraints(55, 18,
89, 20));
    jPanel1.add(jLabel2, new XYConstraints(10, 54, -1, 18));
    jPanel1.add(jTextField_pathToCache, new XYConstraints
(97, 55, 173, 22));
    jPanel1.add(jLabel_CacheSize, new XYConstraints(10, 92,
91, 22));
    jPanel1.add(jTextField_CacheSize, new XYConstraints
(107, 94, 163, 22));
    jTabbedPane1.add(jPanel2, "Advance");
    jPanel2.add(jLabel3, new XYConstraints(12, 17, -1, -1));
    jPanel2.add(jPanel3, new XYConstraints(11, 47, 318, 21));
    jPanel3.add(jCheckBox1, null);
    jPanel3.add(jComboBox_StaticPath, null);
    jPanel2.add(jCheckBox2, new XYConstraints(11, 80, 171,
27));
    jPanel2.add(jLabel4, new XYConstraints(12, 116, 113, 23));
    jPanel2.add(jButton4, new XYConstraints(13, 145, 67, 23));
    jPanel2.add(jTextField_worker, new XYConstraints(94, 17,
150, 20));
    this.getContentPane().add(jButton_OK, new XYConstraints
(218, 244, 63, 22));
    this.getContentPane().add(jButton_Cancal, new
XYConstraints(293, 244, -1, 22));

    File initfile = new File(jTextField_pathToCache.getText()+
                "\\status\\status.txt");
    if(initfile.exists())
    {
      byte[] tmpbyte = new byte[100];
      FileInputStream retrieveFile = new FileInputStream
(initfile);
      retrieveFile.read(tmpbyte);
      String rd = new String(tmpbyte);
      rd = rd.trim();
      jTextField_Port.setText(rd.substring(rd.indexOf("=")+1,
```

```
                    rd.indexOf("\r")));
    rd = rd.substring(rd.indexOf("\r")+2);
    jTextField_pathToCache.setText(rd.substring(rd.indexOf
("=")+1,
                    rd.indexOf("\r")));
    rd = rd.substring(rd.indexOf("\r")+2);
    jTextField_worker.setText(rd.substring(rd.indexOf
("=")+1,
                    rd.indexOf("\r")));
    rd = rd.substring(rd.indexOf("\r")+2);
    jTextField_CacheSize.setText(rd.substring(rd.indexOf
("=")+1));
   }

   oldPort = jTextField_Port.getText().trim();
   oldWorker = jTextField_worker.getText().trim();
   oldPathToCatch = jTextField_pathToCache.getText().trim
();
  }

  void jButton_OK_actionPerformed(ActionEvent e) {
   if(!(jTextField_pathToCache.getText().endsWith("\\")))

jTextField_pathToCache.setText(jTextField_pathToCache.get
Text()+"\\");

   if(jCheckBox1.isEnabled()&&jCheckBox1.isSelected())
    selectPath = jComboBox_StaticPath.getSelectedIndex()+1;
   else
    selectPath = 255;

   oldPort = jTextField_Port.getText();
   oldWorker = jTextField_worker.getText();
   oldPathToCatch = jTextField_pathToCache.getText();
//   oldStaticPath = jTextField_StaticPath.getText();

   this.hide();
   this.dispose();
   mainFrame.enable();
   mainFrame.setVisible(true);
   File StatusFile = new File(jTextField_pathToCache.getText
()+
                    "\\status\\status.txt");
   System.out.println(StatusFile.getAbsolutePath());
   try{
    FileOutputStream saveToFile = new FileOutputStream
(StatusFile);
    byte[] tmpsave = new byte[100];
    tmpsave = ("Port="+jTextField_Port.getText()+"\r\n"+
                "pathtocache="+jTextField_pathToCache.getText
()+"\r\n"+
                "worker="+jTextField_worker.getText()+"\r\n"+
                "cachesize="+jTextField_CacheSize.getText
()+"\r\n")
                .getBytes();
     saveToFile.write(tmpsave);
     saveToFile.close();
    }catch(IOException f){}
   }

   void jButton_Cancal_actionPerformed(ActionEvent e) {
    jTextField_Port.setText(oldPort);
    jTextField_worker.setText(oldWorker);
    jTextField_pathToCache.setText(oldPathToCatch);

    this.hide();
    this.dispose();
    mainFrame.enable();
    mainFrame.setVisible(true);
   }

   void this_windowClosing(WindowEvent e) {
    jTextField_Port.setText(oldPort);
    jTextField_worker.setText(oldWorker);
    jTextField_pathToCache.setText(oldPathToCatch);

    this.dispose();
    mainFrame.enable();
    mainFrame.setVisible(true);
```

```
}

 void jCheckBox1_actionPerformed(ActionEvent e) {
  if (State_CB_Static_Path == false)
{State_CB_Static_Path=true;}
  else {State_CB_Static_Path=false;}

  if (State_CB_Static_Path == true)
  {
   this.jComboBox_StaticPath.setBackground(Color.white);
   this.jComboBox_StaticPath.enable();
  }else
  {
   this.jComboBox_StaticPath.setBackground
(jPanel2.getBackground());
   this.jComboBox_StaticPath.disable();
  }
 }

 void jCheckBox2_actionPerformed(ActionEvent e) {
  if(State_CB_Single == false ) State_CB_Single = true;
  else State_CB_Single = false;
 }

 void jButton4_actionPerformed(ActionEvent e) {
  File path = new File(jTextField_pathToCache.getText());
  File[] list = path.listFiles();
  for(int i = 0;i<list.length;i++)
   if(list[i].isFile())list[i].delete();
  mainFrame.size = 0;
  System.out.println(mainFrame.size);
 }

}
```

Cframe.java

```
package CacheServerApp;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import com.borland.jbcl.layout.*;
import javax.swing.border.*;

public class CFrame extends JFrame {
 JPanel contentPane;
 TitledBorder titledBorder1;
 BorderLayout borderLayout2 = new BorderLayout();
 TextArea textArea1 = new TextArea();

 /**Construct the frame*/
 public CFrame() {
  enableEvents(AWTEvent.WINDOW_EVENT_MASK);
  try {
   jbInit();
  }
  catch(Exception e) {
   e.printStackTrace();
  }
 }
 /**Component initialization*/
 private void jbInit() throws Exception  {
  this.show(false);
  titledBorder1 = new TitledBorder("");
  //setIconImage(Toolkit.getDefaultToolkit().createImage
(Frame1.class.getResource("[Your Icon]")));
  contentPane = (JPanel) this.getContentPane();
  contentPane.setLayout(borderLayout2);
  this.setSize(new Dimension(400, 289));
  this.setTitle("Frame Title");
  textArea1.setEditable(false);
  this.hide();
  contentPane.add(textArea1, BorderLayout.CENTER);
 }
 /**Overridden so we can exit when window is closed*/
 protected void processWindowEvent(WindowEvent e) {
  super.processWindowEvent(e);
  if (e.getID() == WindowEvent.WINDOW_CLOSING) {
   this.dispose();
  }
 }
}
```