

การแบ่งภาระงานสำหรับเซิร์ฟเวอร์ให้บริการสื่อแบบต่อเนื่อง

LOAD BALANCING FOR CONTINUOUS MEDIA SERVER



ดุษิต นิยะโต  
DUSIT NIYATO

เลขหมู่.....  
เลขทะเบียน..... 40800  
วัน, เดือน, ปี 26 พ.ย. 2544

|        |
|--------|
| b..... |
| i..... |

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2544

ISBN 974-648-410-9

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

# **LOAD BALANCING FOR CONTINUOUS MEDIA SERVER**



**A THESIS SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENT FOR THE DEGREE OF  
MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING  
SCHOOL OF GRADUATE STUDIES  
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG**

**2001**

**ISBN 974-648-410-9**

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



**COPYRIGHT 2001**

**SCHOOL OF GRADUATE STUDIES**

**KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG**

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

|                             |                                                             |
|-----------------------------|-------------------------------------------------------------|
| หัวข้อวิทยานิพนธ์           | การแบ่งภาระงานสำหรับเซิร์ฟเวอร์ที่ให้บริการสื่อแบบต่อเนื่อง |
| นักศึกษา                    | นายคณิศ นิชะโต                                              |
| รหัสประจำตัว                | 42061036                                                    |
| ปริญญา                      | วิศวกรรมศาสตรมหาบัณฑิต                                      |
| สาขาวิชา                    | วิศวกรรมไฟฟ้า                                               |
| พ.ศ.                        | 2544                                                        |
| อาจารย์ผู้ควบคุมวิทยานิพนธ์ | ดร. ชุตินเมษฐ์ ศรีนิลทา                                     |

### บทคัดย่อ

ในวิทยานิพนธ์นี้ได้นำเสนอการศึกษาสถาปัตยกรรมและอัลกอริทึมในการแบ่งภาระงานสำหรับเซิร์ฟเวอร์ที่ให้บริการสื่อแบบต่อเนื่อง โดยได้นำสถาปัตยกรรมและอัลกอริทึมที่มีใช้งานอยู่ในเซิร์ฟเวอร์ทั่วไปมาประยุกต์และทดสอบในวิทยานิพนธ์นี้ สำหรับสถาปัตยกรรมที่วิทยานิพนธ์นี้ได้วิเคราะห์นั้นคือ สถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง และสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์สำหรับให้บริการสื่อแบบต่อเนื่องทุกตัว ส่วนอัลกอริทึมที่วิเคราะห์นั้นได้แก่ Random, Round-Robin, Bidding, Diffusion และ State Change Broadcast โดยในวิทยานิพนธ์นี้ได้พัฒนาโปรแกรมจำลองการทำงานเพื่อใช้ศึกษาพฤติกรรมของแต่ละสถาปัตยกรรมและอัลกอริทึมจากเวลาในการเริ่มต้นในการให้บริการ จำนวนการร้องขอที่ได้รับบริการ จำนวนการร้องขอที่ถูกปฏิเสธ และระดับการใช้งานดิสก์ของระบบ จากการศึกษาพบว่าเมื่อใช้อัลกอริทึม State Change Broadcast ร่วมกับสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง ระบบจะปฏิเสธการร้องขอน้อยที่สุด แต่มีโอกาสที่เวลาเริ่มต้นให้บริการจะเพิ่มขึ้นอย่างรวดเร็วหากอัตราการร้องขอสูงกว่าอัตราการผลิตผลตัดสินใจของตัวแบ่งภาระงาน นอกจากนี้ โปรแกรมจำลองที่พัฒนาขึ้น สามารถนำไปใช้ในการประเมินขนาดและประสิทธิภาพของระบบภายใต้สภาวะแวดล้อมต่างๆ ได้อีกด้วย

|                       |                                            |
|-----------------------|--------------------------------------------|
| <b>Thesis Title</b>   | Load Balancing for Continuous Media Server |
| <b>Student</b>        | Mr. Dusit Niyato                           |
| <b>Student ID.</b>    | 42061036                                   |
| <b>Degree</b>         | Master of Engineering                      |
| <b>Programme</b>      | Electrical Engineering                     |
| <b>Year</b>           | 2001                                       |
| <b>Thesis Advisor</b> | Dr. Chutimet Srinilta                      |

## ABSTRACT

This thesis presents a study of architectures and load balancing algorithms of a continuous media server. We applied two architectures and five load balancing algorithms usually used in other kinds of server on our continuous media server model. Such architectures were centralized load balancing architecture and distributed load balancing architecture; and the load balancing algorithms under consideration were Random, Round-Robin, Bidding, Diffusion and State Change Broadcast. We developed the simulator for the study. Performance was measured against the number of serviced requests, number of rejects, startup time and disk utilization. The study found that the centralized load balancing architecture with state change broadcast algorithm outperformed the rest in terms of number of rejects. However, it could give a very high startup time when the request arrival rate is higher than the decision process rate of the load balancer. In addition, the simulator can be used to estimate the system size and performance under any specific operating environment.

## กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยความช่วยเหลือของ ดร. ชุตติเมษฐ์ ศรีนิลทรา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งได้ให้คำแนะนำและคำปรึกษาในการทำวิทยานิพนธ์ฉบับนี้มาโดยตลอด รวมทั้งอาจารย์อีกหลายๆ ท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้คำปรึกษาและความรู้เพื่อใช้ในการทำวิทยานิพนธ์ฉบับนี้ ผู้วิจัยรู้สึกซาบซึ้งในความอนุเคราะห์ของท่านและขอกราบขอบพระคุณเป็นอย่างสูง

และสุดท้ายนี้ผู้วิจัยขอกราบขอบพระคุณ คุณแม่ผู้ให้กำเนิดและให้โอกาสทางการศึกษามาโดยตลอด รวมทั้งผู้ให้กำลังใจทุกๆ ท่านด้วย

คุณค่าและประโยชน์อันพึงมีจากวิทยานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

ชุตติ นิยะโต

# สารบัญ

|                                                                                                        | หน้า |
|--------------------------------------------------------------------------------------------------------|------|
| บทคัดย่อภาษาไทย.....                                                                                   | I    |
| บทคัดย่อภาษาอังกฤษ.....                                                                                | II   |
| กิตติกรรมประกาศ.....                                                                                   | III  |
| สารบัญ.....                                                                                            | IV   |
| สารบัญรูป.....                                                                                         | VII  |
| บทที่ 1 บทนำ.....                                                                                      | 1    |
| 1.1 ความเป็นมา และความสำคัญของปัญหา.....                                                               | 1    |
| 1.2 ความมุ่งหมาย และวัตถุประสงค์ของการศึกษา.....                                                       | 3    |
| 1.3 ขอบเขตการวิจัย.....                                                                                | 3    |
| 1.4 ขั้นตอนของการศึกษา.....                                                                            | 3    |
| บทที่ 2 ทฤษฎีพื้นฐาน.....                                                                              | 5    |
| 2.1 ข้อมูลสื่อผสม.....                                                                                 | 5    |
| 2.2 ระบบให้บริการข้อมูลสื่อผสม.....                                                                    | 6    |
| 2.3 สื่อแบบต่อเนื่อง.....                                                                              | 6    |
| 2.4 เซิร์ฟเวอร์ให้บริการสื่อแบบต่อเนื่อง.....                                                          | 7    |
| 2.4.1 ระบบให้บริการสื่อแบบต่อเนื่อง.....                                                               | 7    |
| 2.4.2 การประกันประสิทธิภาพ (Performance Guarantee).....                                                | 8    |
| 2.5 อัลกอริทึมการแบ่งภาระงาน.....                                                                      | 13   |
| 2.5.1 ประเภทของอัลกอริทึมในการแบ่งภาระงาน.....                                                         | 13   |
| 2.6 งานวิจัยที่เกี่ยวข้องกับระบบสื่อผสม.....                                                           | 14   |
| 2.7 งานวิจัยที่เกี่ยวข้องกับ QoS.....                                                                  | 16   |
| 2.8 งานวิจัยที่เกี่ยวข้องกับอัลกอริทึมการแบ่งภาระงาน.....                                              | 17   |
| 2.9 งานวิจัยที่เกี่ยวข้องกับอัลกอริทึมการแบ่งภาระงานสำหรับเซิร์ฟเวอร์ที่ให้บริการสื่อแบบต่อเนื่อง..... | 27   |

## สารบัญ(ต่อ)

|                                                                                                               | หน้า |
|---------------------------------------------------------------------------------------------------------------|------|
| บทที่ 3 เซิร์ฟเวอร์สำหรับให้บริการสื่อแบบต่อเนื่อง.....                                                       | 33   |
| 3.1 สถาปัตยกรรม.....                                                                                          | 33   |
| 3.2 โครงสร้างและการทำงาน.....                                                                                 | 34   |
| 3.2.1 ส่วนประกอบของสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง.....                                         | 34   |
| 3.2.2 ส่วนประกอบของสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์.....                               | 35   |
| 3.2.2 กระบวนการอนุญาตเพื่อเข้าใช้ (Admission Control) ที่ใช้ในเซิร์ฟเวอร์สำหรับให้บริการสื่อแบบต่อเนื่อง..... | 36   |
| 3.3 อัลกอริทึมการแบ่งภาระงาน.....                                                                             | 38   |
| 3.3.1 อัลกอริทึมการแบ่งภาระงานของสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง.....                              | 38   |
| 3.3.2 อัลกอริทึมการแบ่งภาระงานในสถาปัตยกรรมที่มีตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์.....                   | 44   |
| บทที่ 4 การจำลองการทำงาน.....                                                                                 | 60   |
| 4.1 โปรแกรมจำลองการทำงาน.....                                                                                 | 60   |
| 4.2 สมมติฐานในการจำลองการทำงาน.....                                                                           | 62   |
| 4.1.1 อัตราการร้องขอ.....                                                                                     | 62   |
| 4.1.2 สื่อ.....                                                                                               | 62   |
| 4.1.2.1 รูปแบบของสื่อ.....                                                                                    | 62   |
| 4.1.2.2 การเรียกใช้สื่อ.....                                                                                  | 63   |
| 4.1.3 เซิร์ฟเวอร์ให้บริการสื่อแบบต่อเนื่อง.....                                                               | 63   |
| 4.1.3.1 คิสก์.....                                                                                            | 64   |
| 4.1.4 ตัวแบ่งภาระงานที่ศูนย์กลาง.....                                                                         | 64   |
| 4.1.5 ระบบเครือข่าย.....                                                                                      | 64   |
| 4.3 มาตรวัดที่ใช้ในการจำลองการทำงาน.....                                                                      | 65   |
| 4.4 ผลการจำลองการทำงานที่ 1.....                                                                              | 65   |

## สารบัญ(ต่อ)

|                                       | หน้า |
|---------------------------------------|------|
| 4.5 ผลการจำลองการทำงานที่ 2.....      | 67   |
| 4.6 ผลการจำลองการทำงานที่ 3.....      | 68   |
| 4.7 ผลการจำลองการทำงานที่ 4.....      | 73   |
| 4.8 ผลการจำลองการทำงานที่ 5.....      | 76   |
| 4.9 ผลการจำลองการทำงานที่ 6.....      | 77   |
| 4.10 ผลการจำลองการทำงานที่ 7.....     | 80   |
| บทที่ 5 สรุป.....                     | 82   |
| 5.1 สรุปผลการจำลองการทำงาน.....       | 82   |
| 5.2 ข้อเสนอแนะ.....                   | 84   |
| 5.2.1 เกี่ยวกับงานวิจัย.....          | 84   |
| 5.2.2 เกี่ยวกับการนำไปใช้งานจริง..... | 84   |
| เอกสารอ้างอิง.....                    | 86   |
| ประวัติผู้เขียน.....                  | 91   |

# สารบัญรูป

| รูปที่                                                                                                                   | หน้า |
|--------------------------------------------------------------------------------------------------------------------------|------|
| 1.1 การเพิ่มความสามารถในการให้บริการ โดยการเพิ่มความสามารถของเซิร์ฟเวอร์.....                                            | 2    |
| 1.2 การเพิ่มความสามารถในการให้บริการด้วยการเพิ่มจำนวนเซิร์ฟเวอร์.....                                                    | 2    |
| 2.1 ข้อมูลสื่อผสมประกอบด้วยสื่อหลายๆ แบบ ดังเช่นข้อความ รูปภาพ เสียง และวิดีโอ .....                                     | 5    |
| 2.2 วิดีโอเป็นสื่อแบบต่อเนื่องที่ต้องแสดงผลภาพให้ตรงกับเวลาที่กำหนดไว้.....                                              | 6    |
| 2.3 โครงสร้างของระบบสื่อผสมที่ให้บริการแบบโต้ตอบ.....                                                                    | 7    |
| 2.4 โครงสร้างของระบบสื่อผสมแบบให้บริการสื่อแบบต่อเนื่อง.....                                                             | 8    |
| 2.5 โครงสร้างของบัฟเฟอร์เพื่อบรรเทา jitter.....                                                                          | 10   |
| 2.6 แสดงปริมาณข้อมูลในบัฟเฟอร์กรณีที่อัตราการรับข้อมูลใกล้เคียงกับอัตราการเล่นสื่อ.....                                  | 10   |
| 2.7 แสดงปริมาณข้อมูลในบัฟเฟอร์กรณีที่อัตราการรับข้อมูลมากกว่าอัตราการเล่นสื่อ.....                                       | 11   |
| 2.8 แสดงปริมาณข้อมูลในบัฟเฟอร์กรณีที่อัตราการรับข้อมูลน้อยกว่าอัตราการเล่นสื่อ.....                                      | 11   |
| 2.9 โครงสร้างของระบบที่ใช้อัลกอริทึมในการแบ่งภาระงาน MMPPacking.....                                                     | 28   |
| 2.10 โครงสร้างของวิดีโอเซิร์ฟเวอร์ Tiger.....                                                                            | 29   |
| 2.11 โครงสร้างของระบบที่มีการแบ่งภาระงานแบบอะแดปทีฟ.....                                                                 | 30   |
| 2.12 โครงสร้างสำหรับวิดีโอเซิร์ฟเวอร์ที่มีกระบวนการ Replication.....                                                     | 31   |
| 2.13 โครงสร้างของระบบ Scalable Access Platform (SAP).....                                                                | 32   |
| 3.1 สถาปัตยกรรมของระบบให้บริการสื่อแบบต่อเนื่องแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง.....                                   | 33   |
| 3.2 สถาปัตยกรรมของระบบให้บริการสื่อแบบต่อเนื่องแบบที่มีตัวแบ่งภาระงานประจำอยู่ที่เซิร์ฟเวอร์.....                        | 33   |
| 3.3 โครงสร้างของเซิร์ฟเวอร์สำหรับให้บริการสื่อแบบต่อเนื่องในสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง.....           | 34   |
| 3.4 โครงสร้างของเซิร์ฟเวอร์สำหรับให้บริการสื่อแบบต่อเนื่องในสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์..... | 36   |
| 3.5 เวลาในการเข้าใช้บริการเซิร์ฟเวอร์ของการร้องขอที่ 1 ถึง K (คาบเวลารอบของเซิร์ฟเวอร์คือ T วินาที).....                 | 37   |
| 3.6 การทำงานของอัลกอริทึม Random สำหรับสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง.....                                   | 38   |

## สารบัญรูป(ต่อ)

| รูปที่                                                                                                                                                                          | หน้า |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| 3.7 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Random ของสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง.....                                                                        | 39   |
| 3.8 การทำงานของอัลกอริทึม Round-Robin สำหรับสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง.....                                                                                     | 40   |
| 3.9 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Round-Robin ของสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง.....                                                                   | 41   |
| 3.10 การทำงานของอัลกอริทึม State Change Broadcast สำหรับสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง.....                                                                      | 42   |
| 3.11 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม State Change Broadcast ของสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง.....                                                       | 43   |
| 3.12 การทำงานของอัลกอริทึม Random สำหรับสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายที่เซิร์ฟเวอร์ทุกเครื่อง.....                                                                      | 44   |
| 3.13 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Random ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่ไม่มีการถ่ายโอนการร้องขอ.....                 | 45   |
| 3.14 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Random ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่มีการถ่ายโอนการร้องขอ.....                    | 46   |
| 3.15 การทำงานของอัลกอริทึม Round-Robin สำหรับสถาปัตยกรรมที่มีตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง.....                                                              | 47   |
| 3.16 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Round-Robin ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่ไม่มีการถ่ายโอนการร้องขอ.....            | 48   |
| 3.17 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Round-Robin ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่มีการถ่ายโอนการร้องขอ.....               | 49   |
| 3.18 การทำงานของอัลกอริทึม State Change Broadcast สำหรับสถาปัตยกรรมที่มีตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง.....                                                   | 50   |
| 3.19 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม State Change Broadcast ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่ไม่มีการถ่ายโอนการร้องขอ..... | 51   |

## สารบัญรูป(ต่อ)

| รูปที่                                                                                                                                                                           | หน้า |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| 3.20 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม State Change Broadcast ของสถาปัตยกรรมที่<br>ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่มีการถ่ายโอนการร้องขอ..... | 52   |
| 3.21 การทำงานของอัลกอริทึม Bidding สำหรับสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่<br>เซิร์ฟเวอร์ทุกเครื่อง.....                                                              | 53   |
| 3.22 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Bidding ของสถาปัตยกรรมที่ตัวแบ่งภาระงาน<br>กระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่ไม่มีถ่ายโอนการร้องขอ.....                | 54   |
| 3.23 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Bidding ของสถาปัตยกรรมที่ตัวแบ่งภาระงาน<br>กระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่มีการถ่ายโอนการร้องขอ.....                | 55   |
| 3.24 การทำงานของอัลกอริทึม Diffusion สำหรับสถาปัตยกรรมที่มีตัวแบ่งภาระงานกระจายอยู่ที่<br>เซิร์ฟเวอร์ทุกเครื่อง.....                                                             | 57   |
| 3.25 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Diffusion ของสถาปัตยกรรมที่ตัวแบ่งภาระ<br>งานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่ไม่มีถ่ายโอนการร้องขอ.....              | 57   |
| 3.26 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Diffusion ของสถาปัตยกรรมที่ตัวแบ่งภาระ<br>งานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่มีการถ่ายโอนการร้องขอ.....              | 58   |
| 4.1 โปรแกรมจำลองการทำงาน.....                                                                                                                                                    | 61   |
| 4.2 แสดงการร้องขอเข้าใช้ระบบ.....                                                                                                                                                | 62   |
| 4.3 Zipf Distribution ที่มีค่าพารามิเตอร์เท่ากับ 0.271.....                                                                                                                      | 63   |
| 4.4 กราฟแสดงแบนด์วิธของแต่ละเซิร์ฟเวอร์เมื่อใช้สวิตช์ความเร็ว 100 เมกะบิตต่อวินาที [47].....                                                                                     | 65   |
| 4.5 กราฟแสดงจำนวนการร้องขอที่ได้รับบริการในจำนวนเซิร์ฟเวอร์ที่แตกต่างกัน.....                                                                                                    | 66   |
| 4.6 กราฟแสดงจำนวนการร้องขอที่ได้รับบริการในระบบที่เซิร์ฟเวอร์มีคิสก์แบนด์วิธที่แตกต่างกัน.....                                                                                   | 67   |
| 4.7 กราฟแสดงจำนวนการร้องขอที่ถูกลบพิเศษและได้รับบริการในระบบที่มีเซิร์ฟเวอร์ 10 เครื่อง ด้วย<br>คาบเวลาเฉลี่ยระหว่างการร้องขอ 50 มิลลิวินาที.....                                | 70   |
| 4.8 กราฟแสดงจำนวนการร้องขอที่ถูกลบพิเศษและได้รับบริการในระบบที่มีเซิร์ฟเวอร์ 10 เครื่อง ด้วย<br>คาบเวลาเฉลี่ยระหว่างการร้องขอ 100 มิลลิวินาที.....                               | 71   |
| 4.9 กราฟแสดงจำนวนการร้องขอที่ถูกลบพิเศษในแต่ละอัลกอริทึมของแต่ละสถาปัตยกรรม.....                                                                                                 | 74   |
| 4.10 กราฟแสดงการร้องขอแรกที่ถูกปฏิเสธ.....                                                                                                                                       | 75   |

## สารบัญรูป(ต่อ)

| รูปที่                                                                                                                                                                                                                                                          | หน้า |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| 4.11 กราฟแสดงการร้องขอทั้งหมดที่ถูกปฏิเสธ.....                                                                                                                                                                                                                  | 75   |
| 4.12 กราฟแสดงเวลาตอบสนองของอัลกอริทึม State Change Broadcast ของสถาปัตยกรรมที่มีตัวแบ่ง<br>ภาระงานที่ศูนย์กลาง และสถาปัตยกรรมที่มีตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ ภายใต้อัตรา<br>การร้องขอที่มีคาบเวลาระหว่างการร้องขอเฉลี่ย 50 และ 100 มิลลิวินาที..... | 77   |
| 4.13 กราฟแสดงอัตราการใช้การร้องขอที่ถูกปฏิเสธ และ Disk Utilization เมื่อคาบเวลาเฉลี่ยระหว่างการร้อง<br>เป็น 100 มิลลิวินาที เมื่อแบนด์วิธของดิสก์เท่ากับ 10 เมกะไบต์ต่อวินาที.....                                                                              | 78   |
| 4.14 กราฟแสดงอัตราการใช้การร้องขอที่ถูกปฏิเสธ และ Disk Utilization เมื่อคาบเวลาเฉลี่ยระหว่างการร้อง<br>เป็น 50 มิลลิวินาที เมื่อแบนด์วิธของดิสก์เท่ากับ 10 เมกะไบต์ต่อวินาที.....                                                                               | 79   |
| 4.15 กราฟแสดงอัตราการใช้การร้องขอที่ถูกปฏิเสธ และ Disk Utilization เมื่อคาบเวลาเฉลี่ยระหว่างการร้อง<br>เป็น 50 มิลลิวินาที เมื่อแบนด์วิธของดิสก์เท่ากับ 40 เมกะไบต์ต่อวินาที.....                                                                               | 79   |
| 4.16 กราฟแสดงจำนวนการร้องขอที่ถูกปฏิเสธในกรณีที่สื่อมีความยาวประมาณ 30 นาที.....                                                                                                                                                                                | 81   |

# บทที่ 1

## บทนำ

### 1.1 ความเป็นมา และความสำคัญของปัญหา

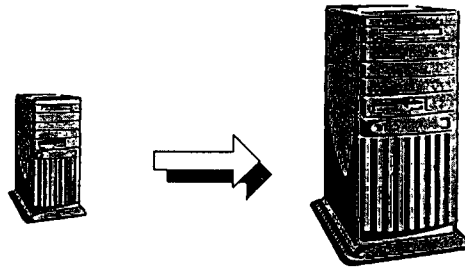
ในอดีตนั้นการใช้งานข้อมูลสื่อผสมยังไม่ได้ความนิยมแพร่หลายนัก ทั้งนี้เนื่องจากการขาดการสนับสนุนทางด้านฮาร์ดแวร์ ทั้งทางด้านประสิทธิภาพและคุณสมบัติ ในปัจจุบัน ความสามารถทางฮาร์ดแวร์ได้พัฒนาไปมาก เอื้อประโยชน์ต่อการใช้งานข้อมูลสื่อผสม และเนื่องจากข้อมูลสื่อผสมประกอบด้วยสื่อหลายๆ ชนิด ทำให้ผู้ใช้สามารถเข้าใจ และใช้งานข้อมูลได้สะดวกกว่าการใช้ข้อมูลจากสื่อชนิดใดชนิดหนึ่งเพียงชนิดเดียว นอกจากนั้นสื่อบางแบบก็สามารถดึงดูดความสนใจของผู้ใช้ได้มากกว่า ตัวอย่างเช่น สื่อประเภทรูปภาพ เสียง และวิดีโอ เหตุผลเหล่านี้ส่งผลให้ความต้องการการใช้งานข้อมูลสื่อผสมสูงขึ้นอย่างรวดเร็ว

แอปพลิเคชันการให้บริการข้อมูลสื่อผสมมีอยู่หลายแบบในปัจจุบัน ตัวอย่างเช่น การให้บริการข้อมูล (Information Retrieval) การให้บริการสื่อสารข้อมูลสื่อผสม (Multimedia Communication) ถึงแม้ว่าแอปพลิเคชันการใช้งานข้อมูลสื่อผสมจะแตกต่างกัน แต่การประยุกต์ใช้งานทางด้านสื่อผสมส่วนใหญ่ก็ยังใช้หลักการ และทฤษฎีพื้นฐานเหมือนกัน

ในวิทยานิพนธ์นี้จะเน้นที่การให้บริการข้อมูลสื่อผสม เฉพาะที่เป็นสื่อแบบต่อเนื่อง ตัวอย่างของการนำไปประยุกต์ใช้งานของสื่อดังกล่าวคือ บริการข้อมูลสื่อผสมแบบออนไลน์ บริการข้อมูลสื่อผสมตามคำขอ เป็นต้น ซึ่งแอปพลิเคชันทางด้านนี้ได้รับความนิยมอย่างมากในปัจจุบัน แต่ผู้ให้บริการประสบปัญหาอย่างหนึ่งคือ ความสามารถของระบบไม่เพียงพอต่อความต้องการของผู้ใช้ ซึ่งปัญหานี้สามารถแก้ไขได้โดยการเพิ่มความสามารถในการให้บริการ ซึ่งแบ่งออกเป็น 2 วิธี วิธีแรกทำโดยการเพิ่มความสามารถให้กับเซิร์ฟเวอร์ที่มีอยู่ ส่วนวิธีที่สองทำได้โดยการเพิ่มจำนวนเซิร์ฟเวอร์สำหรับให้บริการ

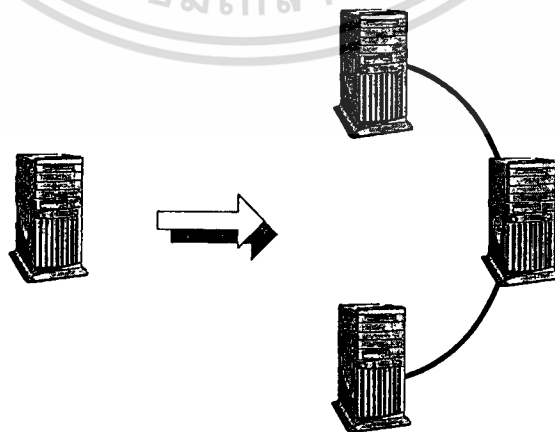
ในกรณีที่เซิร์ฟเวอร์เป็นคอมพิวเตอร์เครื่องเดียว วิธีแก้ปัญหาวีธีแรกนั้นเป็นการเพิ่มความสามารถให้กับเซิร์ฟเวอร์ที่ให้บริการข้อมูลสื่อผสมในระบบ วิธีนี้สามารถทำได้โดยการจัดซื้อฮาร์ดแวร์ที่มีความสามารถมากขึ้น ดังเช่น เปลี่ยนหน่วยประมวลผลให้มีความเร็วมากขึ้น เปลี่ยนไปใช้สถาปัตยกรรมของฮาร์ดแวร์ที่ให้ความเร็วในการทำงานสูงขึ้น เพิ่มขนาดของหน่วยความจำ หรือเพิ่มขนาดอุปกรณ์เก็บข้อมูล ซึ่งในส่วนของซอฟต์แวร์นั้น ผู้ให้บริการอาจจะไม่จำเป็นต้องมีการเปลี่ยนแปลงเลย แต่วิธีการเพิ่มความสามารถให้กับระบบนี้ก็ยังมีข้อเสีย เนื่องจากมีเซิร์ฟเวอร์ให้บริการอยู่เพียงเครื่องเดียวก็จะเป็นการให้บริการจากศูนย์กลาง ซึ่งมีจุดด้อยก็คือ ถ้าเซิร์ฟเวอร์ตัวเดียวตัวนั้นเกิดความผิดพลาด หรือหยุดให้บริการ ผู้เข้าใช้ข้อมูลทั้งหมดก็จะไม่สามารถเข้าใช้บริการได้ วิธีการเพิ่ม

ความสามารถในการให้บริการกับระบบโดยการเพิ่มความสามารถให้กับเซิร์ฟเวอร์แสดงอยู่ในรูปที่ 1.1



รูปที่ 1.1 การเพิ่มความสามารถในการให้บริการโดยการเพิ่มความสามารถของเซิร์ฟเวอร์

สำหรับวิธีแก้ปัญหาแบบที่สองนั้นก็คือการเพิ่มจำนวนเซิร์ฟเวอร์สำหรับให้บริการข้อมูลสื่อผสมให้กับระบบ ดังเช่น เดิมมีเซิร์ฟเวอร์สำหรับให้บริการข้อมูลสื่อผสมอยู่เพียง 1 เครื่อง ผู้ให้บริการก็จัดซื้อเซิร์ฟเวอร์ขึ้นมาให้บริการอีก 3 เครื่อง รวมเป็นทั้งหมด 4 เครื่องเป็นต้น วิธีการเพิ่มความสามารถในการให้บริการของระบบวิธีนี้ จะมีความซับซ้อนมากกว่าวิธีแรก ทั้งนี้เนื่องจากนอกจากผู้ให้บริการข้อมูลสื่อผสมจะต้องซื้อฮาร์ดแวร์มาเพิ่มแล้ว ก็อาจจะต้องมีการเปลี่ยนซอฟต์แวร์ที่ใช้เพื่อให้เหมาะสำหรับการให้บริการข้อมูลสื่อผสมจากหลายๆ เซิร์ฟเวอร์ด้วย นอกจากนี้ยังต้องกำหนดรูปแบบ และการแบ่งงานให้กับระบบอีกด้วย แต่การเพิ่มความสามารถในการให้บริการของระบบด้วยวิธีการเพิ่มจำนวนเซิร์ฟเวอร์นี้ก็มิใช่ว่าดีคือ ถ้าเซิร์ฟเวอร์เครื่องใดเครื่องหนึ่งเกิดความผิดพลาด หรือหยุดให้บริการ ก็ยังมีเซิร์ฟเวอร์เครื่องอื่นๆ ที่ยังสามารถให้บริการต่อไปได้ ทำให้ระบบไม่หยุดการให้บริการทั้งหมด วิธีการเพิ่มความสามารถในการให้บริการกับระบบโดยการเพิ่มจำนวนเซิร์ฟเวอร์แสดงอยู่ในรูปที่ 1.2



รูปที่ 1.2 การเพิ่มความสามารถในการให้บริการด้วยการเพิ่มจำนวนเซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## 1.2 ความมุ่งหมาย และวัตถุประสงค์ของการศึกษา

ในวิทยานิพนธ์นี้มีจุดประสงค์เพื่อศึกษาสถาปัตยกรรมและอัลกอริทึมสำหรับการแบ่งภาระงานบนเซิร์ฟเวอร์ที่ให้บริการสื่อแบบต่อเนื่อง โดยได้ทำการประเมินสถาปัตยกรรม และอัลกอริทึมในการแบ่งภาระงานที่มีอยู่ในปัจจุบันมาศึกษาเพื่อประเมินว่าสถาปัตยกรรมและอัลกอริทึมในการแบ่งภาระงานใดเหมาะสมสำหรับระบบที่ให้บริการสื่อแบบต่อเนื่องในสภาวะแวดล้อมใด และพัฒนาโปรแกรมจำลองการทำงานของระบบเพื่อใช้ในการประเมินดังกล่าวและวางแผนตั้งหรือปรับขนาดของระบบให้สอดคล้องกับสภาพแวดล้อมการใช้งานที่เป็นอยู่จริง

## 1.3 ขอบเขตการวิจัย

1. ศึกษาการให้บริการสื่อแบบต่อเนื่องจากระบบที่มีเซิร์ฟเวอร์หลายๆ เครื่อง ศึกษาสถาปัตยกรรมของเซิร์ฟเวอร์และอัลกอริทึมในการแบ่งภาระงาน
2. ศึกษาข้อดีข้อเสียของสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานอยู่ที่ศูนย์กลาง และแบบที่มีตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง
3. จำลองการทำงานเพื่อศึกษาพฤติกรรมของอัลกอริทึมคือ อัลกอริทึม Random, Round-Robin, State Change Broadcast, Bidding [1] และอัลกอริทึม Diffusion [2] สำหรับแต่ละสถาปัตยกรรม
4. วิเคราะห์ผลการจำลองการทำงาน และเสนอการนำโปรแกรมการจำลองการทำงานไปประยุกต์ในการวางแผนการตั้งหรือปรับขนาดของระบบให้บริการสื่อต่อเนื่อง

## 1.4 ขั้นตอนของการศึกษา

1. ศึกษากระบวนการให้บริการของเซิร์ฟเวอร์สำหรับสื่อแบบต่อเนื่อง ทั้งในส่วนของสถาปัตยกรรม และการทำงาน
2. ศึกษาอัลกอริทึมสำหรับแบ่งภาระงานที่มีอยู่สำหรับงานทางด้านการประมวลผลทั่วไป พร้อมทั้งทำการเลือกอัลกอริทึมที่สามารถนำมาประยุกต์ใช้กับการให้บริการสื่อแบบต่อเนื่องได้
3. ศึกษาและกำหนดเงื่อนไข สภาพแวดล้อมในการประยุกต์ใช้สถาปัตยกรรมและอัลกอริทึมในการแบ่งภาระงานกับเซิร์ฟเวอร์ให้บริการสื่อแบบต่อเนื่อง
4. ออกแบบและพัฒนาโปรแกรมจำลองการทำงานเพื่อศึกษาคุณสมบัติของสถาปัตยกรรมและ อัลกอริทึมในการแบ่งภาระงาน
5. จำลองการทำงานพร้อมทั้งวิเคราะห์ผลที่ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## 6. สรุปผลการศึกษาและวิจัย

วิทยานิพนธ์นี้แบ่งเป็น 5 บท ได้แก่ บทที่ 1 เป็นบทนำของวิทยานิพนธ์ที่ประกอบด้วยความเป็นมา หลักการและเหตุผล ความมุ่งหมายและวัตถุประสงค์ ขอบเขต และขั้นตอนของการวิจัย บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานและงานวิจัยที่เกี่ยวข้องโดยแบ่งเป็น หลักการพื้นฐานของข้อมูลสื่อสาร ระบบสำหรับให้บริการข้อมูลสื่อสาร สื่อแบบต่อเนื่อง รวมทั้งทฤษฎีพื้นฐานของเซิร์ฟเวอร์ที่ให้บริการสื่อแบบต่อเนื่อง และอัลกอริทึมที่ใช้ในการแบ่งภาระงาน

ในบทที่ 3 ได้กล่าวถึงโมเดลของเซิร์ฟเวอร์สำหรับให้บริการสื่อแบบต่อเนื่องที่ใช้ในงานวิจัยนี้ โดยอธิบายถึงสถาปัตยกรรมของระบบและการทำงานของอัลกอริทึมแบ่งภาระงาน ในบทที่ 4 ได้กล่าวถึงการจำลองการทำงาน โดยได้อธิบายภาพรวมของการจำลองการทำงาน สมมติฐานที่กำหนดขึ้นในการจำลองการทำงาน และสถานะแวดล้อมในการจำลองการทำงาน

ในบทที่ 5 จะเป็นส่วนของสรุปผลงานวิจัย ซึ่งประกอบด้วย 2 ส่วนคือ สรุปผลที่ได้จากการจำลองการทำงาน และสรุปผลจากการให้บริการสื่อแบบต่อเนื่องในสถานะแวดล้อมการทำงานต่างๆ

## บทที่ 2

### ทฤษฎีพื้นฐาน

#### 2.1 ข้อมูลสื่อผสม

ข้อมูลสื่อผสมเป็นข้อมูลที่ประกอบด้วยสื่อหลายๆ ชนิด โดยความหมายของสื่อคือ ชนิดของข้อมูลสารสนเทศ (Information) หรือชนิดของผู้ให้บริการข้อมูล ดังเช่น ข้อมูลตัวอักษร ภาพ เสียง และวิดีโอ ดังที่แสดงอยู่ในรูปที่ 2.1 โดยเราสามารถแบ่งชนิดของสื่อตามพื้นฐานของรูปแบบทางกายภาพ และความสัมพันธ์ของสื่อกับเวลา สำหรับการแบ่งชนิดของสื่อตามความสัมพันธ์กับเวลานั้นแบ่งออกเป็น 2 แบบคือ สื่อแบบสถิต (Static Media) และสื่อแบบไดนามิก (Dynamic Media)



รูปที่ 2.1 ข้อมูลสื่อผสมประกอบด้วยสื่อหลายๆ แบบ ดังเช่นข้อความ รูปภาพ เสียง และวิดีโอ

สื่อแบบสถิตนั้นเป็นสื่อที่ไม่ขึ้นกับกับเวลา ข้อมูลภายในสื่อและความหมายของสื่อ นั้นไม่ขึ้นกับเวลาในการนำเสนอ ตัวอย่างของสื่อแบบสถิตได้แก่ ข้อมูลตัวอักษร ภาพกราฟิก และรูปภาพ เป็นต้น

สำหรับสื่อแบบไดนามิกนั้นเป็นสื่อที่ขึ้นกับเวลา ความหมาย และความถูกต้องของสื่อประเภทนี้จะขึ้นกับอัตราการในการนำเสนอ ตัวอย่างของสื่อแบบไดนามิกเช่น แอนิเมชัน เสียง และวิดีโอ เป็นต้น สื่อแบบไดนามิกนี้จะมีอัตราการนำเสนอเป็นค่าเฉพาะของตนเอง ตัวอย่างเช่น ข้อมูลวิดีโอมีอัตราการนำเสนอ 25 เฟรมต่อวินาที เป็นต้น

2.2 ระบบให้บริการข้อมูลสื่อผสม

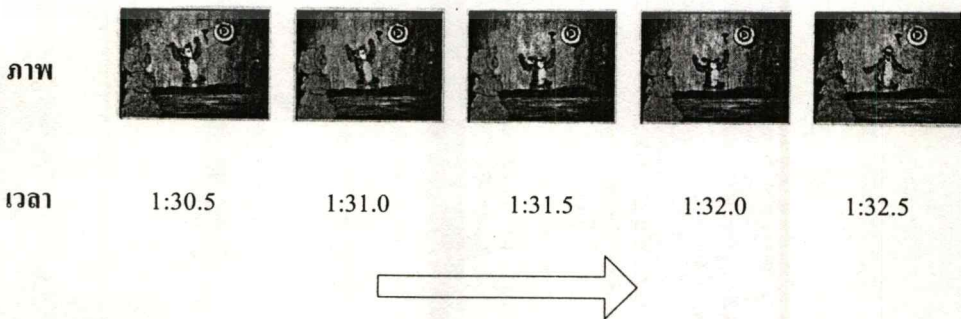
ระบบที่ให้บริการข้อมูลสื่อผสมนั้นจะต้องเป็นระบบที่มีความสามารถในการรองรับ และให้บริการชนิดของสื่อได้มากกว่า 1 ชนิด ซึ่งอาจจะเป็นได้ทั้งสื่อแบบสแตติกและไดนามิก

สำหรับหน้าที่ของระบบให้บริการข้อมูลสื่อผสมนั้นประกอบด้วย การบันทึก สร้าง เก็บ คัด ประมวลผล ส่ง และนำเสนอข้อมูลสื่อผสม เราสามารถแบ่งประเภทหน้าที่ของระบบที่ให้บริการข้อมูลสื่อผสมออกได้เป็น 2 ประเภท คือ หน้าที่ที่เกี่ยวข้องกับการคำนวณ และหน้าที่ที่เกี่ยวข้องกับการสื่อสาร ซึ่งหน้าที่ทั้งสองประเภทนี้จะต้องทำงานร่วมกันเพื่อให้ระบบสามารถให้บริการข้อมูลสื่อผสมได้ โดยหน้าที่ในส่วนของการสื่อสารข้อมูลสื่อผสมนั้นจะเน้นที่การบันทึก ส่ง และการนำเสนอข้อมูลสื่อผสม ในขณะที่ส่วนของการคำนวณจะเป็นหน้าที่ที่เกี่ยวข้องกับการประมวลผลข้อมูลสื่อผสม ดังเช่น การค้นหา การดึง และจัดจำและการปรับปรุงข้อมูลสื่อผสม

จากประเภทของหน้าที่ของระบบให้บริการข้อมูลสื่อผสม ทำให้สามารถแบ่งประเภทของระบบให้บริการข้อมูลสื่อผสมได้ โดยระบบที่ให้บริการสื่อผสมประเภทแรกคือระบบที่มีความสามารถในการบันทึก เก็บ และดึงข้อมูลสื่อผสม ตัวอย่างของระบบประเภทนี้ได้แก่ ระบบให้บริการวิดีโอตามคำขอเป็นต้น ส่วนระบบที่ให้บริการสื่อผสมอีกประเภทจะเป็นระบบให้บริการสื่อผสมที่มีความสามารถในการประมวลผลด้วย เช่น มีความสามารถในการค้นหา และจัดจำรูปภาพ เป็นต้น

2.3 สื่อแบบต่อเนื่อง

สื่อแบบต่อเนื่อง (Continuous Media) เป็นสื่อแบบไดนามิก โดยสื่อแบบต่อเนื่องนั้นเป็นสื่อที่ต้องมีการนำเสนออย่างต่อเนื่องด้วยอัตราคงที่ สื่อแบบต่อเนื่องสามารถเรียกอีกอย่างหนึ่งได้ว่า สื่อแบบไอโซโครนัส (Isochronous Media) เนื่องจากสื่อแบบนี้จะมีความสัมพันธ์ระหว่างสื่อกับเวลาที่คงที่ ตัวอย่างของสื่อแบบต่อเนื่องก็ได้แก่ ข้อมูลเสียง และวิดีโอ ดังที่แสดงอยู่ในรูปที่ 2.2



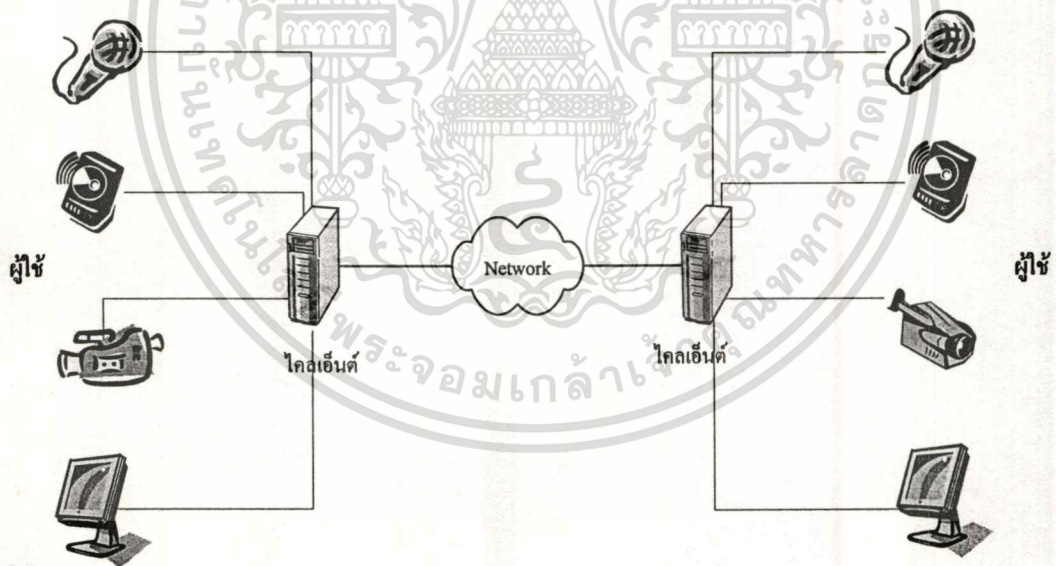
รูปที่ 2.2 วิดีโอเป็นสื่อแบบต่อเนื่องที่ต้องแสดงผลภาพให้ตรงกับเวลาที่กำหนดไว้

## 2.4 เซิร์ฟเวอร์ให้บริการสื่อแบบต่อเนื่อง

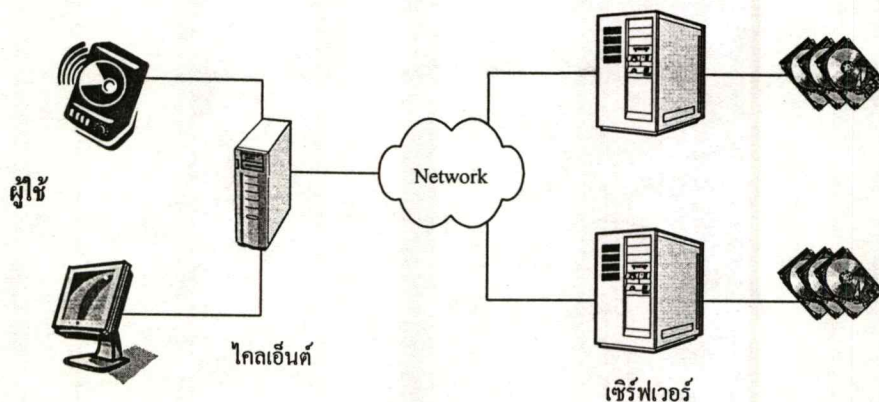
### 2.4.1 ระบบให้บริการสื่อแบบต่อเนื่อง

สำหรับระบบที่ให้บริการข้อมูลสื่อผสมนั้นเราสามารถแบ่งออกได้เป็น 2 ประเภทก็คือ ระบบที่มีการโต้ตอบระหว่างผู้ใช้ (รูปที่ 2.3) ดังเช่นแอปพลิเคชันการประชุมทางไกลผ่านระบบเครือข่าย (Video Conference) หรือโทรศัพท์ผ่านอินเทอร์เน็ต (Internet Telephone) เป็นต้น ส่วนอีกประเภทจะเป็นระบบที่ผู้ใช้ไปขอเข้าใช้บริการจากเซิร์ฟเวอร์ (รูปที่ 2.4) ตัวอย่างเช่น บริการวิดีโอตามคำขอ (Video On Demand) และการออกอากาศวิทยุผ่านทางอินเทอร์เน็ต เป็นต้น ในระบบประเภทที่สอง ผู้ใช้จะร้องขอไปยังเซิร์ฟเวอร์เพื่อขอใช้สื่อ เมื่อเซิร์ฟเวอร์รับการร้องขอก็จะส่งข้อมูลที่เกี่ยวข้องและข้อมูลสำหรับการแสดงผลไปให้ผู้ใช้

ส่วนประกอบหลักของระบบให้บริการข้อมูลสื่อผสม โดยทั่วไปแอปพลิเคชันของสื่อแบบต่อเนื่องนั้นจะทำการบันทึกข้อมูลเสียงและวิดีโอด้วยไมโครโฟนและกล้องวิดีโอ จากนั้นก็นำมาทำการแปลงให้เป็นข้อมูลดิจิทัลและทำการเข้ารหัส จากนั้นก็จะส่งไปให้กับผู้ใช้ปลายทางผ่านระบบเครือข่าย เมื่อไคลเอนต์ปลายทางได้รับข้อมูลก็จะทำการถอดรหัสและแสดงผลสื่อให้กับผู้ใช้



รูปที่ 2.3 โครงสร้างของระบบสำหรับสื่อแบบต่อเนื่องที่ให้บริการแบบโต้ตอบ



รูปที่ 2.4 โครงสร้างของระบบสื่อสารแบบที่ผู้ใช้ขอบริการจากเซิร์ฟเวอร์

ในวิทยานิพนธ์นี้จะครอบคลุมที่ระบบสำหรับให้บริการข้อมูลจากเซิร์ฟเวอร์ ซึ่งเป็นระบบที่ให้บริการสื่อสารประเภทที่สองที่เราได้กล่าวมา โดยส่วนประกอบสำหรับระบบที่ให้บริการสื่อสารแบบต่อเนื่องนั้นมีส่วนประกอบคือ ข้อมูลที่อยู่ในรูปของสื่อแบบต่อเนื่อง ซึ่งอาจจะเป็นข้อมูลเสียงหรือวิดีโอในรูปแบบดิจิทัล หน่วยเก็บข้อมูลทำหน้าที่เก็บข้อมูลที่อยู่ในรูปของสื่อแบบต่อเนื่อง เซิร์ฟเวอร์ที่ทำหน้าที่รับการร้องขอจากไคลเอ็นต์ ดึงข้อมูลมาจากหน่วยเก็บข้อมูลและส่งไปให้ไคลเอ็นต์ ระบบเครือข่ายซึ่งเป็นตัวกลางในการรับส่งข้อมูลระหว่างไคลเอ็นต์และเซิร์ฟเวอร์ และไคลเอ็นต์ที่ทำหน้าที่ส่งคำร้องขอไปยังเซิร์ฟเวอร์ รับข้อมูลจากระบบเครือข่ายที่ส่งมาจากเซิร์ฟเวอร์ จากนั้นทำการถอดรหัสข้อมูลดิจิทัลให้อยู่ในรูปที่เหมาะสมสำหรับการแสดงผล

สำหรับเงื่อนไขที่ทำให้ระบบสำหรับให้บริการสื่อสารแบบต่อเนื่องแตกต่างจากระบบที่ให้บริการสื่อสารแบบได้ตอบได้ทั่วไปก็คือ

- ข้อมูลที่อยู่ในรูปของสื่อแบบต่อเนื่องจะได้รับการบีบอัดและเข้ารหัสข้อมูลไว้ก่อนแล้ว และข้อมูลนี้จะเก็บอยู่ในอุปกรณ์สำหรับเก็บข้อมูล
- ข้อมูลที่เก็บอยู่ในสื่อแบบต่อเนื่องจะถูกดึงมาจากอุปกรณ์เก็บข้อมูล มากกว่าที่จะสร้างและเข้ารหัสแบบทันทีทันใด
- ข้อมูลที่เก็บอยู่ในอุปกรณ์เก็บข้อมูลมีขนาดใหญ่
- ค่าหน่วยเวลาที่เกิดขึ้นมีผลกระทบต่อให้บริการไม่มาก แต่ทั้งนี้ก็ต้องไม่มากเกินไปกว่าที่ผู้ใช้จะยอมรับได้เช่นกัน

#### 2.4.2 การประกันประสิทธิภาพ (Performance Guarantee)

ระบบให้บริการข้อมูลสื่อสารนั้นเป็นระบบที่สนับสนุนการดึง ประมวลผล และการส่งทั้งสื่อต่อเนื่องทั้งที่ขึ้นและไม่ขึ้นกับเวลา สำหรับสื่อแบบต่อเนื่องที่มีเงื่อนไขของเวลาอย่างเข้มงวด ทั้งในแง่ของแบนด์วิดท์ (Bandwidth) ค่าหน่วยเวลา (Delay) และจitter (Jitter) โดยทั่วไปนั้นสื่อแบบ

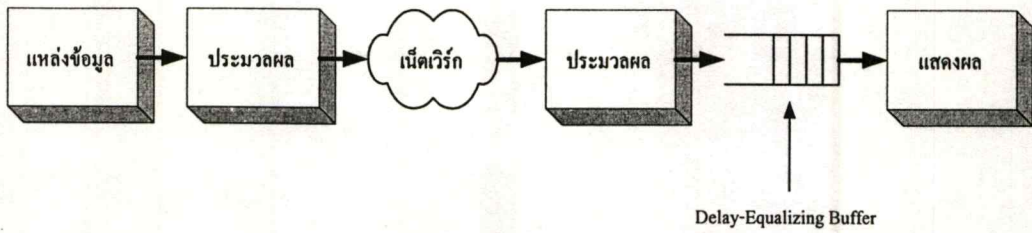
เอกภพออนไลน์สำหรับทั้งระบบที่มีประสิทธิภาพสูงและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต่อเนื่องจะได้รับการบีบอัดข้อมูล เพื่อให้มีขนาดที่เล็กลง และทำให้การส่งข้อมูลต้องการแบนด์วิดท์ น้อยลง แต่ถึงแม้ว่าจะมีการบีบอัดข้อมูลแล้ว ข้อมูลของสื่อแบบต่อเนื่องก็ยังมีขนาดใหญ่ และยัง ต้องการแบนด์วิดท์ที่สูงกว่าสื่อประเภทอื่น ตัวอย่างเช่น การส่งข้อมูลวิดีโอในมาตรฐาน MPEG (Motion Pictures Expert Group) คุณภาพปานกลางถึงคุณภาพสูงนั้นจำเป็นต้องใช้แบนด์วิดท์ของ ระบบเครือข่ายสูงมาก (1.5 เมกะบิตต่อวินาทีสำหรับ MPEG-1)

ข้อมูลที่อยู่ในรูปของสื่อแบบต่อเนื่องนั้นจะมีเงื่อนไขเกี่ยวกับเวลา สื่อเหล่านี้จะได้รับการ บันทึกภายใต้อัตราที่กำหนดไว้ พร้อมทั้งมีการกำหนดอัตราการเล่น (Playback Rate) ไว้ด้วย ซึ่งถ้า หากเล่นสื่อแบบต่อเนื่องในอัตราที่ช้าหรือเร็วกว่าที่กำหนดไว้ก็จะทำให้สื่อแบบต่อเนื่องนั้นกระตุก หรือกระโดด ด้วยเหตุผลนี้การให้บริการสื่อแบบต่อเนื่องจึงจำเป็นต้องมีการประกันประสิทธิภาพ การให้บริการ

ถ้าการให้บริการข้อมูลสื่อแบบต่อเนื่องบนกระทำบนคอมพิวเตอร์และระบบเครือข่ายที่ได้รับการ จัดเตรียมไว้โดยเฉพาะ การประกันประสิทธิภาพการให้บริการนั้นก็ทำได้ไม่ยากนัก แต่ด้วย เหตุผลทางด้านเศรษฐกิจ ระบบสำหรับให้บริการสื่อแบบต่อเนื่องส่วนใหญ่จึงมักให้บริการโดยใช้ คอมพิวเตอร์และระบบเครือข่ายที่มีการใช้งานร่วมกับแอปพลิเคชันอื่น โดยทั่วไประบบเครือข่ายที่ใช้ จะเป็นแบบแพคเกจสวิตช์ ซึ่งในสถานะแวดล้อมของระบบเครือข่ายแบบนี้ แพคเกจข้อมูลจะถูกส่ง ไปให้กับหน่วยจัดเส้นทางของข้อมูลต่างๆ จนกระทั่งข้อมูลไปถึงยังผู้รับ ในการเดินทางนี้จะใช้ เวลามากหรือน้อยขึ้นอยู่กับความหนาแน่นของข้อมูลของเครือข่ายในขณะนั้น สถานะแวดล้อมเช่น นี้ทำให้ข้อมูลเกิดจิทเทอร์ (Jitter) ซึ่งหมายถึงแต่ละแพคเกจข้อมูลเดินทางมาถึงด้วยอัตราที่ไม่เท่า กันส่งผลให้การเล่นสื่อไม่สม่ำเสมอ ซึ่งปัญหาจิทเทอร์นี้สามารถบรรเทาได้ด้วยการใช้บัฟเฟอร์ (Buffer)

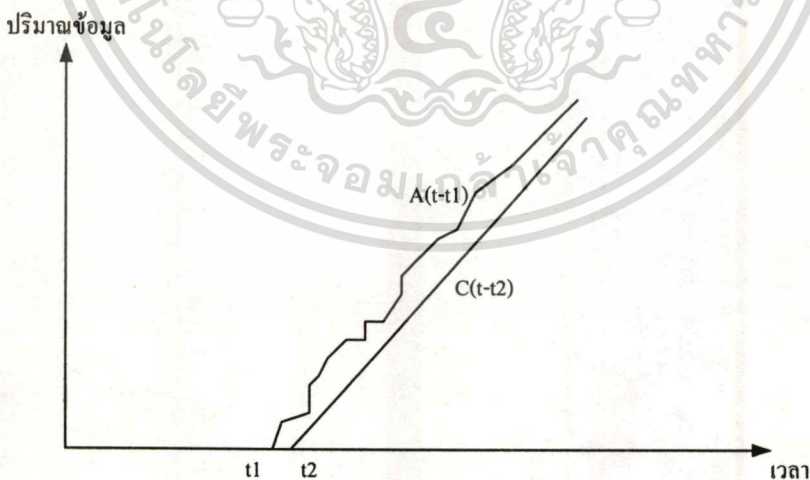
จิทเทอร์นั้นเกิดขึ้นได้จากหลายสาเหตุ ดังเช่น เวลาที่ใช้ในการประมวลผลแพคเกจที่แตก ต่างกัน ความแตกต่างในการเข้าใช้ระบบเครือข่าย และความแตกต่างในการเข้าคิวของแพคเกจ ระบบที่ให้บริการสื่อผสมสามารถลดจิทเทอร์ได้ด้วยการใช้บัฟเฟอร์แบบ First-In First-Out (FIFO) ที่โคลเ็นด์ปลายทางก่อนที่จะเริ่มเล่นสื่อ ดังที่แสดงอยู่ในรูปที่ 2.5 ถึงแม้ว่าแพคเกจที่ได้รับเข้ามา จะมีอัตราที่แตกต่างกัน แต่โคลเ็นด์สามารถแสดงผลข้อมูลด้วยอัตราที่คงที่จากบัฟเฟอร์ หลักการ พื้นฐานของบัฟเฟอร์ก็คือ ตัวบัฟเฟอร์จะเพิ่มค่าหน่วงเวลาให้กับทุกๆ แพคเกจที่มีเข้ามาทั้งนี้เพื่อให้ อัตราการเล่นสื่อคงที่เช่นเดียวกับแพคเกจข้อมูลในบัฟเฟอร์ ด้วยเหตุผลนี้ทำให้บางครั้งมีการเรียก บัฟเฟอร์ว่า Delay-Equalizing Buffer โครงสร้างของระบบที่มีการใช้บัฟเฟอร์แสดงอยู่ในรูปที่ 2.5



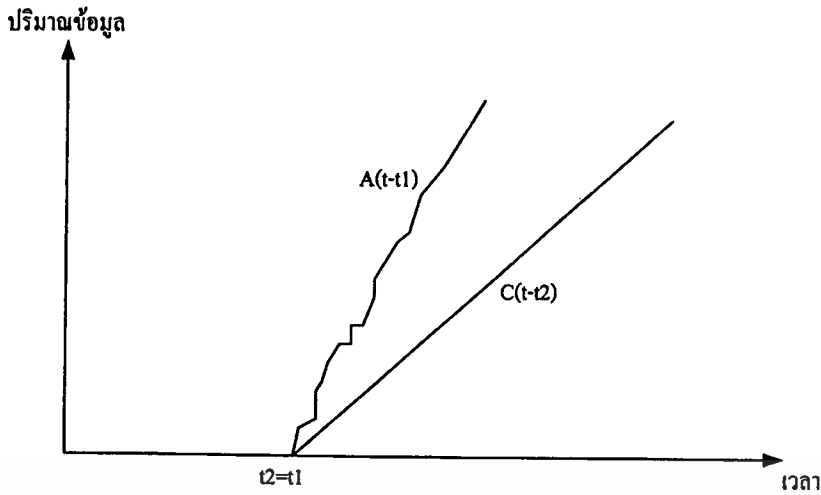
รูปที่ 2.5 โครงสร้างของระบบที่มีการใช้บัฟเฟอร์เพื่อบรรเทาจอเทอร์

แต่ทั้งนี้การใช้บัฟเฟอร์นั้นระบบจะต้องคอยตรวจสอบไม่ให้บัฟเฟอร์เกิดการโอเวอร์โฟลว์ (Overflow) คือข้อมูลเข้ามากกว่าเนื้อที่ว่างในบัฟเฟอร์ หรือเกิดอันเดอร์โฟลว์ (Underflow) คือข้อมูลในบัฟเฟอร์ไม่พอใช้ และขนาดของบัฟเฟอร์จะต้องมีขนาดไม่ใหญ่มาเกินไปเพื่อเป็นการประหยัดหน่วยความจำ

สำหรับแบบจำลองที่ระบบให้บริการสื่อแบบต่อเนื่องต้องการนั้น ค่าเฉลี่ยของอัตราการรับข้อมูลจะต้องเท่ากับค่าเฉลี่ยของอัตราการเล่นสื่อแบบต่อเนื่อง ดังที่แสดงอยู่ในรูปที่ 2.6 โดย A เป็นอัตราการรับข้อมูล ส่วน C เป็นอัตราการเล่นสื่อ ถ้าค่าเฉลี่ยของอัตราการรับข้อมูลมากกว่าอัตราการเล่นสื่อแล้วความแตกต่างระหว่าง  $A(t-t_1)$  และ  $C(t-t_2)$  จะมากขึ้นเรื่อยๆ ซึ่งในกรณีนี้ขนาดของบัฟเฟอร์จะต้องมีไม่จำกัด เพื่อให้บัฟเฟอร์สามารถรองรับข้อมูลที่มีเข้ามาได้อย่างเพียงพอ ดังแสดงอยู่ในรูปที่ 2.7 ในกรณีนี้ระบบจะต้องมีความสามารถในการตรวจสอบอัตราการเข้ามาของข้อมูลกับขนาดของบัฟเฟอร์ เพื่อควบคุมการส่งข้อมูลจากแหล่งข้อมูล

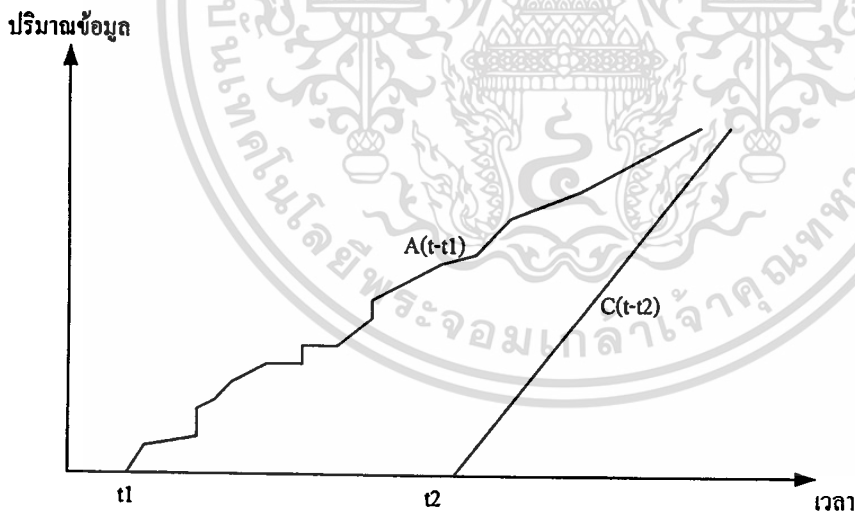


รูปที่ 2.6 แสดงปริมาณข้อมูลในบัฟเฟอร์กรณีที่อัตราการรับข้อมูลใกล้เคียงกับอัตราการเล่นสื่อ [3]



รูปที่ 2.7 แสดงปริมาณข้อมูลในบัฟเฟอร์กรณีที่อัตราการรับข้อมูลมากกว่าอัตราการเล่นสื่อ [3]

ในอีกกรณีหนึ่งคืออัตราการเข้ามาของข้อมูลต่ำกว่าอัตราการเล่นข้อมูล ในกรณีนี้ถ้าต้องการให้การเล่นสื่อไม่เสียคุณภาพ ค่า  $A(t-t1)-C(t-t2)$  จะต้องไม่น้อยกว่า 0 นั่นหมายความว่า  $t2$  จะต้องมากกว่า  $t1$  มาก ดังที่แสดงอยู่ในรูปที่ 2.8 กล่าวง่ายๆ ก็คือผู้แสดงผลข้อมูลจะต้องคอยจนกว่าจะมีข้อมูลเพียงพอจึงจะเริ่มเล่นสื่อแบบต่อเนื่อง



รูปที่ 2.8 แสดงปริมาณข้อมูลในบัฟเฟอร์กรณีที่อัตราการรับข้อมูลน้อยกว่าอัตราการเล่นสื่อ [3]

#### 2.4.3 การจัดการการรับรองคุณภาพของการให้บริการ (QoS Management)

##### นิยาม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การสื่อสารและให้บริการข้อมูลสื่อผสมนั้นจะต้องมีการประกันประสิทธิภาพ โดยประสิทธิภาพในที่นี้จะหมายถึงคุณสมบัติของการให้บริการสื่อแบบต่อเนื่องซึ่งโดยทั่วไปจะพิจารณาในแง่ของ อัตราการเล่นสื่อ เป็นหลัก ทั้งนี้การประกันประสิทธิภาพจำเป็นต้องมีการกำหนดเฟรมเวิร์กกลางสำหรับแอปพลิเคชันที่แตกต่างกันเพื่อกำหนดความต้องการในการประกันประสิทธิภาพ รวมถึงคุณสมบัติของระบบที่ต้องรับประกันด้วย ได้มีผู้นำเสนอแนวความคิดของการรับรองคุณภาพในการให้บริการหรือ Quality of Service (QoS) [4]-[6] แต่จนถึงในปัจจุบันก็ยังไม่มีการนิยามสำหรับ QoS ใดที่เป็นที่ยอมรับอย่างแท้จริง อย่างไรก็ตามถ้ากล่าวถึง QoS แล้วจะหมายถึงการกำหนดคุณลักษณะเพื่อให้แอปพลิเคชันสามารถทำงานและให้เอาท์พุทได้ในระดับที่ยอมรับได้ ใน [4] ได้กล่าวถึงนิยามของ QoS ไว้ว่า

“การรับรองคุณภาพในการให้บริการจะนำเสนอชุดของคุณลักษณะทั้งทางด้านปริมาณและคุณภาพของระบบที่ให้บริการสื่อผสมแบบกระจายเพื่อให้แอปพลิเคชันสามารถทำงานได้ตามที่ต้องการ”

อีกนิยามหนึ่งคือ “การรับรองคุณภาพในการให้บริการเป็นข้อกำหนดทั้งด้านคุณภาพและปริมาณสำหรับความต้องการของแอปพลิเคชัน ซึ่งระบบที่ให้บริการสื่อผสมจะต้องมีคุณสมบัติให้ตรงกับข้อกำหนดเพื่อให้แอปพลิเคชันสามารถทำงานได้ตามคุณภาพที่กำหนดไว้”

จากนิยามเราจะเห็นได้ว่าระบบการรับรองคุณภาพในการให้บริการนั้นจะประกอบด้วย 2 ส่วนใหญ่ๆ ส่วนแรกคือแอปพลิเคชันที่ทำหน้าที่กำหนด ความต้องการทางด้านคุณภาพ และอีกส่วนคือระบบที่มีรับรองคุณสมบัติของคุณภาพตามที่ได้กำหนดไว้ สำหรับการรับรองคุณภาพในการให้บริการนั้นโดยทั่วไปจะมีการกำหนดค่าของพารามิเตอร์ต่างๆ ดังเช่น บิตเรต อัตราความผิดพลาด ค่าหน่วงเวลา และจิทเทอร์เป็นต้น ในระบบหนึ่งๆ พารามิเตอร์เหล่านี้อาจจะมีค่ามากกว่าหนึ่งค่า แล้วแต่คุณลักษณะและคุณภาพของระบบกำหนดไว้

สำหรับการรับรองคุณภาพในการให้บริการของระบบเครือข่ายนั้นก็ได้แก่การกำหนดประสิทธิภาพของการถ่ายโอนข้อมูลในแง่ของความเชื่อถือได้ ค่าหน่วงเวลา และทราฟฟิค (Throughput) ตัวอย่างเช่นในโมเดลอ้างอิง Open System Interconnection (OSI) นั้นมีพารามิเตอร์สำหรับการรับรองคุณภาพที่แบ่งเป็น ความเร็ว และความเชื่อถือได้ของการถ่ายโอนข้อมูล ดังเช่น ทราฟฟิค ค่าหน่วงเวลาของการถ่ายโอนข้อมูล อัตราความผิดพลาดของข้อมูล และความน่าจะเป็นของการเชื่อมต่อที่ล้มเหลว พารามิเตอร์เหล่านี้จะมีการรับรองอยู่ในระดับต่างๆ ของโมเดล ซึ่งก็จะแตกต่างกันไป สำหรับการรับรองคุณภาพในการให้บริการของระบบที่ให้บริการสื่อผสมนั้นก็就会有ความแตกต่างจากของระบบเครือข่าย เพราะว่าในระบบที่ให้บริการสื่อผสมจะต้องมีการรับรองคุณภาพในทุกๆ ระดับของการให้บริการ ซึ่งการรับรองคุณภาพในระบบเครือข่ายเพียงส่วนหนึ่งเท่านั้น

#### 2.4.4 การอนุญาตเพื่อเข้าใช้ (Admission Control)

การทำงานของระบบที่ให้บริการสื่อผสมที่มีคุณสมบัติในการรับรองคุณภาพการให้บริการ ระบบจะต้องรักษาคุณภาพของการบริการให้ไม่ต่ำกว่าอัตราที่ผู้ใช้ต้องการตลอดช่วงของการบริการ ก่อนที่ระบบจะตัดสินใจให้บริการใดๆ ระบบต้องพิจารณาก่อนว่าระบบสามารถรักษาคุณภาพได้ตลอดช่วงการบริการหรือไม่ โดยพิจารณาจากทรัพยากรที่เหลืออยู่ในระบบ ถ้าระบบไม่สามารถทำได้ ระบบก็จะปฏิเสธการให้บริการ การตรวจสอบในลักษณะนี้เรียกว่าการทำ Admission Control

### 2.5 อัลกอริทึมการแบ่งภาระงาน

#### 2.5.1 ประเภทของอัลกอริทึมในการแบ่งภาระงาน

อัลกอริทึมในการแบ่งภาระงานเป็นอัลกอริทึมที่เกี่ยวข้องกับการตัดสินใจแบ่งงานที่มีอยู่บนคอมพิวเตอร์หรือหน่วยประมวลหนึ่งไปให้กับคอมพิวเตอร์หรือหน่วยประมวลผลอื่นทั้งนี้เพื่อให้งานสามารถประมวลผลได้เสร็จเร็วขึ้น หรือมีประสิทธิภาพมากยิ่งขึ้น โดยทั่วไปเราสามารถแบ่งอัลกอริทึมในการแบ่งภาระงานออกได้ 3 ประเภทตามกระบวนการคือ อัลกอริทึมในการแบ่งภาระงานแบบสถิติก (Static Load Balancing Algorithms) อัลกอริทึมในการแบ่งภาระงานแบบไดนามิก (Dynamic Load Balancing Algorithms) และ อัลกอริทึมในการแบ่งภาระงานแบบอะแดปทีฟ (Adaptive Load Balancing Algorithms)

สำหรับอัลกอริทึมในการแบ่งภาระงานแบบสถิติกเป็นรูปแบบการแบ่งภาระงานที่มีการกำหนดการถ่ายโอนภาระงานไว้ตั้งแต่ก่อนที่ระบบจะทำงานแล้ว ตัวอย่างเช่นเซิร์ฟเวอร์ฐานข้อมูลอาจจะแบ่งการเก็บข้อมูลตามเซิร์ฟเวอร์ต่างๆ ที่มี คือเซิร์ฟเวอร์ตัวที่ 1 อาจจะเก็บข้อมูลของสินค้าที่มีหมายเลขประจำตัวตั้งแต่ 1-1,000 เซิร์ฟเวอร์ตัวที่ 2 เก็บข้อมูลของสินค้าที่มีหมายเลขประจำตัว 1,000-2,000 เป็นต้น

อัลกอริทึมในการแบ่งภาระงานแบบไดนามิกเป็นรูปแบบการแบ่งภาระงานที่มีการตัดสินใจแบ่งภาระงานเมื่อระบบได้ทำงานไปแล้ว อัลกอริทึมแบบนี้ผู้พัฒนาจะกำหนดว่าระบบจะใช้เงื่อนไขใดในการตัดสินใจแบ่งภาระงาน ตัวอย่างเช่น เซิร์ฟเวอร์ฐานข้อมูลอาจจะพิจารณาว่าในขณะที่เซิร์ฟเวอร์ใดเก็บข้อมูลน้อยที่สุด แล้วเลือกให้เซิร์ฟเวอร์ที่มีข้อมูลเก็บอยู่น้อยที่สุดเพื่อนำข้อมูลที่มีเข้ามาใหม่ไปเก็บ เป็นต้น

อัลกอริทึมในการแบ่งภาระงานแบบอะแดปทีฟ เป็นรูปแบบในการแบ่งภาระงานที่ระบบมีความสามารถในการปรับเปลี่ยนรูปแบบการกระจายภาระงานตามสถานะของระบบได้ ในกรณีนี้ระบบจะมีรูปแบบในการกระจายงานอยู่หลายรูปแบบ แล้วในสถานะหนึ่งๆ ระบบก็จะเลือกใช้วิธี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โควิธีหนึ่งเท่านั้น ตัวอย่างเช่น ในสถานะที่มีการเข้าใช้ข้อมูลจากเซิร์ฟเวอร์ไม่มาก เซิร์ฟเวอร์อาจจะใช้วิธีเลือกเซิร์ฟเวอร์สำหรับเพิ่มข้อมูลจากจำนวนเซิร์ฟเวอร์ที่เก็บอยู่ แต่ในสถานะที่มีการเข้าใช้ข้อมูลมาก เซิร์ฟเวอร์อาจจะเลือกเก็บข้อมูลโดยแบ่งตามหมายเลขประจำตัวเป็นต้น

จะเห็นว่าอัลกอริทึมแบบสแตติกไม่มีการใช้ข้อมูลสถานะของภาระงานปัจจุบันในการตัดสินใจกระจายภาระงาน ในขณะที่อัลกอริทึมแบบไดนามิกและอะแดปทีฟมีการใช้ข้อมูลสถานะภาระงานปัจจุบันประกอบการตัดสินใจ ด้วยเหตุนี้โดยทั่วไปแล้ว อัลกอริทึมแบบไดนามิกและอะแดปทีฟจะมีความซับซ้อนมากกว่าอัลกอริทึมแบบสแตติก

## 2.6 งานวิจัยที่เกี่ยวข้องกับระบบที่ให้บริการสื่อผสม

[7] กล่าวถึงการสร้างแอปพลิเคชันในการแสดงภาพวิดีโอผ่านบน WWW ตัวโปรแกรม cmplayer นั้นพัฒนาขึ้นโดยใช้ Berkeley Continuous Media Toolkit (CMT) (<http://bmrc.berkeley.edu/projects/cmt>) โปรแกรมสนับสนุนฟังก์ชัน VCR โดยมีการใช้ออบเจกต์ Logical Time Systems ในการควบคุมการแสดงผลและฟังก์ชันของ VCR โปรแกรม cmplayer นั้นสนับสนุน Motion-JPEG และ MPEG-I สำหรับวิดีโอ และ 8-bit ulaw และ 16-bit linear audio สำหรับเสียง

[8] ได้กล่าวถึง Oracle Media Server สำหรับการให้บริการสื่อต่างๆ โดยเฉพาะข้อมูลวิดีโอสำหรับระบบ Video On Demand (VOD) ในบทความได้อธิบายว่าระบบโครงสร้างการทำงานของซูเปอร์คอมพิวเตอร์นั้นไม่เหมาะสำหรับงานทางด้านการให้บริการสื่อผสมเนื่องจากเป็นสถาปัตยกรรมที่มีการใช้งานหน่วยความจำร่วมกันระหว่างโนด ส่งผลให้เส้นทางในการรับส่งข้อมูลระหว่างโนดกับหน่วยความจำอาจจะกลายเป็นคอขวดขึ้นมาได้เมื่อจำเป็นต้องใช้ในการส่งข้อมูลจำนวนมาก ซึ่งสถาปัตยกรรมแบบนี้จะเหมาะสำหรับการประมวลผลมากกว่า ส่วน Oracle Media Server นั้นทำงานบนคอมพิวเตอร์ที่มีสถาปัตยกรรมแบบ Massively Parallel Processing (MPP) ที่โนดแต่ละโนดจะมีหน่วยความจำเป็นของตัวเอง โดยการเชื่อมต่อระหว่างโนดใน Oracle Media Server นั้นเป็นแบบ nCube คือเชื่อมต่อกับแบบ Hypercube โดยข้อดีของสถาปัตยกรรมต่อการนำมาประยุกต์ใช้เป็นเซิร์ฟเวอร์ให้บริการสื่อก็คือ มีความยืดหยุ่นสูง สามารถแก้ไขการติดตั้งอุปกรณ์ I/O ได้ง่าย มีการสลับการทำงานระหว่างโนดที่เร็วมาก มีแบนด์วิดธ์ในการสื่อสารระหว่างโนดที่สูง และสามารถทำงานร่วมกับเครือข่ายได้ง่าย

[9] ได้นำเสนอวิธีการ Admission Control สำหรับเซิร์ฟเวอร์ให้บริการสื่อผสม ซึ่งเป็นแบบการประมาณความน่าจะเป็นในกรณีทั่วไป ซึ่งงานวิจัยส่วนใหญ่จะใช้กรณี Worst-Case ในการตัดสินใจว่าจะตอบรับการร้องขอของไคลเอนต์หรือไม่ การให้บริการต่อไคลเอนต์ของเซิร์ฟเวอร์นั้นจะเป็นรอบๆ ไป โดยแต่ละรอบเซิร์ฟเวอร์จะดึงข้อมูลจากดิสก์และส่งไปให้กับไคลเอนต์ การตรวจเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สอบว่าจะยอมให้บริการต่อการร้องขอจากไคลเอ็นต์ได้หรือไม่ก็จะพิจารณาว่าเวลาที่ใช้ในการดึงข้อมูลต่อไคลเอ็นต์ทั้งหมดนั้นมากกว่าคาบเวลาของรอบหรือไม่ โดยจะในการประมาณจำนวนข้อมูลที่ให้บริการสำหรับไคลเอ็นต์ที่เชื่อมต่ออยู่ กับจำนวนข้อมูลทั้งหมดที่สามารถให้บริการได้ทั้งหมด การประมาณจำนวนข้อมูลที่ให้บริการสำหรับไคลเอ็นต์ที่เชื่อมต่ออยู่นั้นจะใช้ความน่าจะเป็นที่จะทำให้ระบบเกิดสถานะโอเวอร์โฟลว์ โดยความน่าจะเป็นที่จะทำให้ระบบโอเวอร์โฟลว์นั้นจะขึ้นอยู่กับคุณลักษณะในการดึงข้อมูลจากดิสก์ และคุณลักษณะในการให้บริการต่อไคลเอ็นต์ สำหรับการพิจารณาไคลเอ็นต์ที่ร้องขอเข้ามานั้นจะพิจารณาจากการประมาณ นอกจากนี้ในการให้บริการของไคลเอ็นต์ใหม่นั้นเซิร์ฟเวอร์อาจจะมีการละเว้นข้อมูลบางส่วนของสื่อได้ ทั้งนี้เนื่องจากมนุษย์นั้นสามารถยอมรับความผิดพลาดของข้อมูลสื่อผสมบางส่วนได้ โดยในระบบจะมีการกำหนดไว้ว่าส่วนใดของสื่อที่สามารถละเว้นได้ จากการทดลองแสดงให้เห็นว่าการประมาณค่าความน่าจะเป็นช่วยให้เซิร์ฟเวอร์สามารถให้บริการต่อไคลเอ็นต์ได้มากขึ้น และการที่เซิร์ฟเวอร์ให้บริการโดยยอมให้มีการละเว้นข้อมูลบางส่วนนั้นทำให้บริการไคลเอ็นต์ได้เพิ่มขึ้นมากกว่า 200%

[10] ได้นำเสนอกระบวนการ Admission Control สำหรับเซิร์ฟเวอร์ที่ให้บริการข้อมูลสื่อผสมแบบใหม่ โดยไม่ได้คำนึงถึงเฉพาะแต่ความสามารถของเซิร์ฟเวอร์ในการรองรับการร้องขอจากไคลเอ็นต์เท่านั้น แต่ยังพิจารณาถึงคุณลักษณะของการร้องขอจากไคลเอ็นต์เอง ดังเช่นค่าตอบแทน (Reward) ซึ่งจะเป็นรายได้ที่ได้จากการให้บริการ และค่าสูญเสีย (Penalty) ซึ่งจะเป็นรายได้ที่เสียไปเนื่องจากไม่ได้ให้บริการ ในงานวิจัยได้แบ่งความจุในการให้บริการของเซิร์ฟเวอร์ออกเป็น 3 ประเภทตามค่าเรสโธลด์โดยแบบแรกเป็นแบบ Free-Threshold คือระบบจะรองรับการร้องขอที่มีเข้ามาทั้งหมดโดยไม่สนใจคุณลักษณะของไคลเอ็นต์ แบบ Fix-Threshold จะมีการกำหนดค่าเรสโธลด์สำหรับความจุของเซิร์ฟเวอร์ในการให้บริการไคลเอ็นต์ที่มีความสำคัญมาก (High-Priority) และความสำคัญน้อย (Low-Priority) และแบบ Dynamic-Threshold ซึ่งเซิร์ฟเวอร์มีความสามารถในการปรับค่าเรสโธลด์ให้ได้ค่าตอบแทนสูงสุดได้เอง จากการทดลองพบว่าที่สถานะภาระงานน้อยการใช้ Admission Control แบบ Free-Threshold จะดีที่สุด เพราะเซิร์ฟเวอร์ไม่จำเป็นต้องคำนวณค่าตอบแทนจากการร้องขอ ส่วนในสถานะที่มีภาระงานปานกลางถึงมาก การเลือกใช้ Admission Control แบบ Fix-Threshold จะให้ผลดีที่สุดที่ค่า Optimal Threshold และที่สถานะที่มีภาระงานมาก การใช้ Admission Control แบบ Dynamic-Threshold จะให้ผลดีที่สุด เพราะเซิร์ฟเวอร์สามารถกำหนดค่าเรสโธลด์ที่ทำให้ได้ค่าตอบแทนสูงสุดได้เอง

[11] ได้กล่าวถึงการพัฒนา ระบบ Berkeley VOD สำหรับให้บริการวิดีโอตามต้องการ (Video On Demand, VOD) โดยมีการจัดลำดับชั้นของหน่วยเก็บข้อมูล ในระบบประกอบด้วย Archive Server (AS) ที่ทำหน้าที่เก็บมัลติมีเดียออบเจกต์ (Multimedia Object, MO) ทั้งหมด, Video File Server (VFS) สำหรับให้บริการ VOD กับไคลเอ็นต์ ในงานวิจัยได้นำเสนอระบบเครือข่ายสำหรับการให้บริการ VOD โดยแคชนั้นจะอยู่ที่ตัว VFS ในการจัดการกับแคชนั้นตัว VFS จะพิจารณาจากเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ลำดับความสำคัญ (Priority) ของตัว MO พร้อมทั้งมีการสับเปลี่ยนให้ MO ใน CFS มีความเป็น Equilibrium ซึ่งหมายความว่าระดับความสำคัญของ MO ในแคชนั้นจะต้องไม่แตกต่างกันมากกว่าค่าเรสโสลต์ที่กำหนดไว้ ซึ่งโดยปกติแล้ว MO ที่อยู่ในแคชก็จะมีค่า Priority สูงกว่าอยู่แล้ว และงานวิจัยยังได้นำเสนออัลกอริทึมในการวาง MO บน VFS ด้วยการพิจารณาจากภาระงานบน VFS เอง ในงานวิจัยได้แบ่งออกเป็น 2 อัลกอริทึม โดยอัลกอริทึมแรกนั้นจะพิจารณาเฉพาะที่ตัว VFS ซึ่งประกอบด้วย ภาระงานของแคช ภาระงานของตัวสตรีม และภาระงานของระบบเครือข่าย สำหรับการตัดสินใจนั้นจะใช้อัลกอริทึม Greedy

## 2.7 งานวิจัยที่เกี่ยวข้องกับ QoS

[12] กล่าวถึงโพรโตคอล Continuous Media TCP (CMTCP) ซึ่งใช้อัลกอริทึม “TCP-friendly” สำหรับควบคุมความคับคั่ง (Congestion Control) บนโพรโตคอล TCP เพื่อการสื่อสารข้อมูลสื่อผสมแบบต่อเนื่อง โดยผู้วิจัยได้วิเคราะห์คุณลักษณะของการสื่อสารผ่าน TCP จากนั้นจึงได้โมเดล TCP Throughput โดยพิจารณาจาก Round Trip Time และ Loss Rates ในการสื่อสารผู้ส่งจะกำหนด Rounds ตามช่วงเวลา  $M$  (เราเรียก  $M$  นี้ว่า Recomputation Interval) ผู้ส่งจะส่งแพ็กเกตออกไปตาม Rate ใน Round ก่อนหน้า โดยแพ็กเกตจะมีการเพิ่มลำดับของแพ็กเกต และ Time Stamp เข้าไป เมื่อผู้รับได้รับแพ็กเกตก็จะส่ง ACK กลับไปพร้อมทั้งบอกลำดับของแพ็กเกตที่ได้รับ ถ้าแพ็กเกตเกิดการเสียหายตัวผู้ส่งจะไม่ทำการส่งใหม่ แต่จะเก็บข้อมูลไว้ในการคำนวณหาค่า Round Trip Time และ Loss Rate ได้ ในงานวิจัยนี้ใช้การจำลองการทำงานเป็นส่วนใหญ่ และมีการทดลองใช้งานกับอินเทอร์เน็ตโดยให้ CMTCP ทำงานบนโพรโตคอล UDP การทดลองนั้นได้เชื่อมต่อระหว่าง University of Massachusetts กับ University of Washington

[13] แสดงถึงการประยุกต์ใช้เอเจนต์ในการควบคุมและปรับระบบให้ได้ Quality-of-Service (QoS) ในงานวิจัยได้พัฒนา QoS-Agent (QoSA) ที่ติดตั้งอยู่ในอุปกรณ์ในระบบเครือข่าย โดยถ้าเป็นอุปกรณ์ตัวเอเจนต์จะอยู่ในชั้นของโพรโตคอลในเครือข่าย ถ้าอยู่ในเครื่องคอมพิวเตอร์เอเจนต์จะเป็นส่วนหนึ่งของระบบปฏิบัติการ เอเจนต์ในระบบจะทำงานร่วมกันเพื่อให้การเชื่อมต่อรองรับ QoS ตามที่กำหนดไว้ โดยถ้าเอเจนต์ตัวหนึ่งไม่สามารถให้บริการ QoS ได้ เอเจนต์จะสอบถามเอเจนต์ที่อยู่ข้างเคียงด้วยเมสเสจ QoS\_Request เพื่อค้นหาว่าเอเจนต์ข้างเคียงใดที่สามารถให้บริการได้ เอเจนต์ข้างเคียงจะตอบรับด้วยเมสเสจ Available\_QoS เพื่อแจ้งทรัพยากรที่เหลืออยู่ ส่วนเอเจนต์ตัวที่ร้องขอไปเมื่อได้รับเมสเสจนี้มาก็จะตัดสินใจและเลือกเอเจนต์ในอุปกรณ์ที่จะให้มาทำหน้าที่แทนโดยใช้เมสเสจ Allocate\_QoS และในกรณีที่เอเจนต์ทุกตัวในระบบไม่มีทรัพยากรเหลือพอ เอเจนต์ก็จะมีการแจ้งให้ระบบมีการปรับเปลี่ยน QoS ถ้ายังต้องการให้ระบบรับบริการใหม่ที่เข้ามาได้ จากการจำลองการทำงานพบว่าระบบที่มี QoSA จะทำให้ค่าหน่วยเวลาของระบบลดลง และเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำให้ความถี่ของสถานการณ์ที่ระบบไม่สามารถให้บริการได้ตาม QoS น้อยลง ซึ่งเป็นผลดีต่อการให้บริการข้อมูลสื่อผสม

[14] ได้นำเสนอการนำเทคนิคของเอเจนต์มาประยุกต์ใช้ในการดูแล QoS ของระบบสื่อสารภาพวิดีโอ ทั้งนี้เนื่องจากระบบที่ให้บริการสื่อผสมแบบกระจายที่มีเอเจนต์ทำให้ระบบมีความยืดหยุ่น ทำงานแบบกระจายตามพื้นที่ต่างๆ ได้ดี ช่วยปรับปรุงประสิทธิภาพเมื่อมีปัญหาเกิดขึ้น และช่วยทำให้เกิดสถานะแวดล้อมที่เป็นอันหนึ่งอันเดียวกัน ในงานวิจัยได้แบ่งเอเจนต์ออกเป็น Quality Evaluator ซึ่งทำหน้าที่ตรวจสอบคุณภาพของ QoS โดยพิจารณาจากปัจจัยต่างๆ คือ แบนด์วิดท์ ค่าหน่วงเวลา ค่าจitter ความเชื่อถือได้ของระบบ ทรูกุญแจ และเสถียรภาพของระบบ โดยเอเจนต์นี้จะแบ่งเป็น 2 เอเจนต์ย่อยคือ Comparator ที่ทำหน้าที่วัดคุณภาพของ QoS ส่วน Analyzer นั้นทำหน้าที่ในการวิเคราะห์ปัญหาที่อาจจะทำให้ประสิทธิภาพของระบบตกลงโดยใช้วิธีทาง Artificial Intelligent ช่วย ส่วนเอเจนต์ที่สองคือ Performance Enhancers ซึ่งทำหน้าที่แก้ไขเมื่อคุณภาพ QoS ของระบบไม่ได้ตามที่กำหนดไว้ ซึ่งแก้ไขได้ 2 วิธีคือ ทำการลดระดับคุณภาพ QoS ของระบบลง และการกระจายภาระงานไปให้กับทรัพยากรที่เหลืออยู่ ทั้งนี้การทำงานของ Performance Enhancer จะขึ้นอยู่กับข้อมูลที่เอเจนต์ Quality Evaluator ส่งมาให้ หรือถ้าจำเป็นเอเจนต์ Performance Enhancer ก็อาจจะสร้างเอเจนต์ Analyzer ขึ้นมาวิเคราะห์ข้อมูลที่ได้จากเอเจนต์ Quality Evaluator เพื่อหาทางออกที่ดีที่สุดให้กับระบบ

## 2.8 งานวิจัยที่เกี่ยวข้องกับอัลกอริทึมการแบ่งภาระงาน

[15] ได้กล่าวถึงกระบวนการในการแบ่งภาระงานให้กับโพรเซสเซอร์พื้นฐาน โดยมีหลักการง่ายๆ ว่าถ้าโพรเซสเซอร์หนึ่งไม่สามารถประมวลผลภาระงานได้ทันตามเวลาที่ต้องการก็จะมีการแบ่งภาระงานนั้นไปให้กับโพรเซสเซอร์อื่น ในงานวิจัยได้นำเสนอตัวแบ่งภาระงาน (Load Balancer) ใน 3 รูปแบบคือ ตัวแบ่งภาระงานแบบกระจาย (Distributed Load Balancer) ตัวแบ่งภาระงานกลาง (Central Load Balancer) และตัวกระจายภาระงานแบบสุ่ม (Random Load Balancer) โดยตัวกระจายภาระงานแบบกระจายนั้นจะมีอยู่ในโพรเซสเซอร์ทุกตัวในระบบ โดยตัวกระจายภาระงานจะคอยกระจายข้อมูลภาระงานของตัวให้กับตัวกระจายภาระงานอื่นๆ อยู่เป็นระยะเมื่อต้องการแบ่งภาระงานก็จะมีการดึงข้อมูลภาระงานของโพรเซสเซอร์อื่นมาเลือกว่าโพรเซสเซอร์ใดมีภาระงานน้อยที่สุด ส่วนตัวกระจายภาระงานกลางนั้นจะอยู่บนโพรเซสเซอร์ตัวเดียว ทำหน้าที่ตรวจสอบภาระงานบนโพรเซสเซอร์ทั้งหมด และกำหนดภาระงานให้กับโพรเซสเซอร์ที่มีภาระงานน้อยที่สุด และสำหรับตัวแบ่งภาระงานแบบสุ่ม โพรเซสเซอร์แต่ละตัวจะทำหน้าที่แบ่งภาระงานให้กับโพรเซสเซอร์อื่น โดยจะมีโพรเซสเซอร์ที่ทำหน้าที่ตรวจสอบภาระงานคอย

ตรวจสอบภาระงานของโพรเซสเซอร์เป็นระยะ โดยถ้าโพรเซสเซอร์ไม่สามารถประมวลผลภาระงานนั้นได้ โพรเซสเซอร์ก็จะส่งภาระงานไปให้กับโพรเซสเซอร์อื่นแบบสุ่ม

[16] เป็นงานวิจัยที่เกี่ยวข้องกับการพัฒนาระบบ Amoeba ซึ่งเป็นระบบที่ทำงานแบบกระจาย ในการทดสอบประสิทธิภาพของอัลกอริทึมในการแบ่งภาระงานพื้นฐานแบบต่างๆ โดยมาตรวัดหลักที่ใช้ของการทดลองคือเวลาในการตอบสนองเฉลี่ย และจำนวนโพรเซสในโพรเซสเซอร์ ในระบบได้มีการแยกตัวแบ่งภาระงาน (Load Balancer) และเซิร์ฟเวอร์สำหรับย้ายโพรเซส (Process Migration Server) ออกจากกันทั้งนี้เพื่อให้สามารถเปลี่ยนอัลกอริทึมในการแบ่งภาระงานได้ง่าย โดยอัลกอริทึมในการแบ่งภาระงานที่ได้ทดสอบนั้นแบ่งออกเป็น 2 แบบ โดยแบบแรกคือการแบ่งภาระงานในตอนเริ่มต้นประกอบด้วยภาระงานจากส่วนกลาง การแบ่งภาระงานแบบกระจาย และการแบ่งภาระงานแบบสุ่ม แบบที่สองคือการแบ่งภาระงานแบบที่มีการย้ายภาระงานได้ ประกอบด้วยภาระงานจากส่วนกลาง การแบ่งภาระงานแบบสุ่ม การแบ่งภาระงานให้กับคู่ข้างเคียง (Neighbor) [17] และการกระจายข้อมูลของการแบ่งภาระงาน (Broadcast) จากการทดลองพบว่าภาระงานจากส่วนกลางนั้นให้ผลของประสิทธิภาพดีที่สุด และการแบ่งภาระงานเฉพาะในตอนเริ่มต้นมีแนวโน้มที่จะให้ประสิทธิภาพดีกว่าการแบ่งภาระงานแบบที่มีการย้ายภาระงานได้ ทั้งนี้เนื่องจากไม่มี Overhead เกิดขึ้น

[18] ได้กล่าวถึงวิธีในการแบ่งภาระงานโดยเลือกเวลาในการแบ่งภาระงานให้เท่าเทียม โดยใช้เวลาวางของโพรเซสเซอร์ในการบ่งบอกความเท่าเทียมของภาระงาน โดยมีค่า  $C$  เป็นเรสโสดเพื่อเลือกเวลาในการแบ่งภาระงาน ในงานวิจัยได้แบ่งสถานะของภาระงานออกเป็น 2 สถานะคือ Stable State คือสถานะที่ภาระงานคงที่ และมีภาระงานใหม่เข้ามาไม่บ่อยนัก และ Dynamic State คือสถานะที่ภาระงานเข้ามาบ่อย และภาระงานมีระยะเวลาในการทำงานไม่นานมาก ในงานวิจัยก่อนหน้านี้ได้มีการกำหนดค่า  $C$  เป็นค่าคงที่ซึ่งมีข้อเสียคือถ้าภาระงานมีความไม่เท่าเทียมกันน้อยกว่าค่า  $C$  แต่คงอยู่ในสถานะ Stable State เป็นเวลานานก็จะทำให้ประสิทธิภาพของระบบไม่ดี ถ้าเปลี่ยนค่า  $C$  ให้ลดลงก็จะทำให้ระบบมีการแบ่งภาระงานบ่อยเกินไปและทำให้ค่า Overhead สูง ในงานวิจัยนี้ได้นำเสนอวิธีในการกำหนดค่า  $C$  ใหม่โดยแบ่งออกเป็น 2 ค่า ค่า  $C_s$  เป็นเรสโสดปกติ และค่า  $C_d$  ซึ่งเป็นค่าเรสโสดที่ลดลงตามเวลาโดยมีค่าเริ่มต้นเท่ากับ  $C_s$  และเมื่อมีการแบ่งภาระงานเป็นจำนวน  $n$  ครั้งแล้วก็จะรีเซ็ตค่า  $C_d$  เป็นค่า  $C_s$  อีกครั้ง ส่วนในสถานะของภาระงานเป็น Dynamic State นั้นจะใช้วิธีพิจารณาจากคุณลักษณะของภาระงานที่ผ่านมา จากนั้นจึงทำการเลือกค่า  $C$  ตามที่เหมาะสมกับภาระงานในรูปแบบต่างๆ

[19] ได้กล่าวถึงการแบ่งภาระงานในระบบเกมออนไลน์ขนาดใหญ่ โดยสถาปัตยกรรมของระบบนั้นจะเป็นกลุ่มของเซิร์ฟเวอร์ที่เชื่อมต่อระหว่างกันด้วยระบบเครือข่ายความเร็วสูงในการถ่ายโอนภาระงานซึ่งกันและกัน และมีระบบเครือข่ายที่เชื่อมต่อกับไคลเอนต์เฉพาะ ในการออกแบบระบบนั้นมีจุดสำคัญที่การแบ่งแผนที่ของเกมส์ตามแกน  $X$  ในแผนที่ โดยพื้นที่ของการแบ่งนั้นจะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขึ้นอยู่กับตัวละครของเกมสไลซ์นั้น สำหรับอัลกอริทึมในการแบ่งภาระงานนั้นเซิร์ฟเวอร์จะมี  $Overload\_Threshold$  ซึ่งทำหน้าที่เป็นขีดจำกัดของภาระงานที่มากเกินไป และ  $Underload\_Threshold$  ซึ่งเป็นขีดจำกัดของภาระงานที่น้อยเกินไป ถ้าภาระงานของเซิร์ฟเวอร์มากกว่าหรือน้อยกว่าเรสโสดทั้งสองนี้เซิร์ฟเวอร์จะส่งเมสเสจออกไปเพื่อกระจายภาระงานไปให้หรือขอรับภาระงานกับเซิร์ฟเวอร์เครื่องอื่น ในการทดลองได้มีโมเดลการทำงานของระบบและจำลองการทำงานด้วยภาษา Java บนคอมพิวเตอร์พีซีที่เชื่อมต่อระหว่างกันด้วยระบบ LAN ผลจากการทดลองพบว่าการใช้รูปแบบการแบ่งภาระงานแบบไดนามิกนั้นทำให้ Turn Around Time น้อยกว่าการแบ่งภาระงานแบบสแตติกอย่างเห็นได้ชัด

[20] กล่าวถึงการประยุกต์ใช้เทคนิคสำหรับแบ่งภาระงานในระบบ Real-Time ที่ติดตั้งโนดแบบกระจาย งานวิจัยได้เปรียบเทียบและรวมวิธีในการแบ่งภาระงาน Gradient Model [20], Focussed Addressing และ Bidding Methods [22] ในงานวิจัยนั้นแบ่งภาระงานออกเป็น 2 ประเภทคือ Hard Task เป็นภาระงานที่มี Deadline จำกัดและไม่สามารถย้ายภาระงานได้ กับ Soft Task เป็นภาระงานที่ไม่มี Deadline แต่จะมีการกำหนด Response Time ไว้ สำหรับ Gradient Model นั้นจะพิจารณาระดับภาระงานของตน ถ้าภาระงานมากกว่าค่าที่กำหนดก็จะมีการเคลื่อนย้ายภาระงานไปให้กับโนดข้างเคียง โดยพิจารณาจากระยะทางระหว่างโนด และเลือกโนดที่มีภาระงานน้อยที่สุด วิธี Focussed Address จะพิจารณาภาระงานใหม่ที่เข้ามาว่าสามารถจัดการได้หรือไม่ ถ้าไม่ก็จะมีการเลือกโนดปลายทาง วิธีเลือกนั้นจะใช้ข้อมูลทรัพยากรที่เหลือของโนดอื่น โดยโนดที่มีทรัพยากรเหลือมากกว่า  $Threshold * (task\ computation\ times)$  แล้วโนดนั้นสามารถเป็นโนดปลายทางได้ ถ้ามีหลายโนด โหนดต้นทางจะสุ่มเลือกโนดปลายทางขึ้นในกรณีที่ไม่สามารถใช้วิธี Focussed Address ได้ระบบจะใช้วิธี Bidding และสำหรับวิธี Bidding นั้นโนดที่มีภาระงานมากโดยพิจารณาจาก  $earliest-possible-response-time \geq deadline-for-response$  ถ้าสมการเป็นจริงก็จะมีการเมสเสจไปยังโนดอื่น โหนดอื่นก็จะพิจารณาภาระงานของตน คือถ้าภาระงานมากกว่า  $(threshold * the\ task\ computation\ time)$  แล้วก็จะไม่รับภาระงาน แต่ถ้าน้อยกว่าก็จะรับภาระงาน จากการทดลองพบว่าวิธี Gradient Model นั้นให้ผลดีกับช่วงเวลาการเข้ามาของภาระงานที่ไม่ถี่นัก

[23] กล่าวถึงกระบวนการในการวิเคราะห์ความสามารถในการแบ่งภาระงานด้วยอัลกอริทึมต่างๆ ในงานวิจัยมีการกำหนดระดับของภาระงานที่โนดให้เป็น Imbalance คือภาระงานตกที่โนดเพียงโนดเดียว, Moderately Balance คือภาระงานทั้งหมดตกที่โนดจำนวนครึ่งหนึ่งเท่าๆ กัน และ Balanced โหนดทุกๆ โหนดได้รับภาระงานเท่าเทียมกัน โดยพารามิเตอร์ที่ใช้เป็นมาตรวัดในการตัดสินใจการแบ่งภาระงานคือเวลาในการตอบสนอง ในงานวิจัยได้นำเสนอวิธีในการพิจารณาความเท่าเทียมของภาระงานในระบบ คือ System Distance เป็น  $\Delta_t = \sum_{i=1}^n |N_i^t - N_{ave}|$  โดยที่  $N_i^t$  เป็นจำนวน

ของ Task ที่โพรเซสเซอร์ I และ  $N_{ave} = \frac{1}{n} \sum_{i=1}^n N_i^t$  และได้พิสูจน์ว่า ถ้าระบบ Imbalance ค่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Distance จะเท่ากับ 1, ระบบที่ Moderately Balance จะมีค่า Distance อยู่ระหว่าง 0-1 และระบบที่ Balanced จะมี Distance เป็น 0 ในงานวิจัยได้ยกตัวอย่างอัลกอริทึมในการแบ่งภาระงานให้เท่าเทียมแบบสุ่ม โดยสรุปได้ว่าความน่าจะเป็นของภาระงานที่แต่ละโพรเซสเซอร์นั้นจะต้องเท่ากับ  $1/n$  สำหรับการย้ายภาระงานนั้นงานวิจัยกำหนดให้ เวลาของการย้ายภาระงานเท่ากับเวลาที่ใช้ในการเตรียมการบวกด้วยเวลาที่ใช้ในการถ่ายโอนภาระงาน ( $T_c = T_s + T_t$ )

[24] ได้กล่าวถึงการแบ่งภาระงานให้เท่าเทียมสำหรับเว็บเซิร์ฟเวอร์ โดย Domain Name System นั้นจะทำหน้าที่เป็นตัวแบ่งภาระงานจากศูนย์กลางโดยการส่งไอพีแอดเดรสของเซิร์ฟเวอร์ที่ต้องการแบ่งภาระงานไปให้กับไคลเอนต์ที่ร้องขอ URL มา แต่ก็มีปัญหาเนื่องจาก Name Server จะเก็บไอพีแอดเดรสไว้ ในงานวิจัยจึงใช้ค่า Time To Live (TTL) ในการกำหนดว่าไอพีแอดเดรสนั้นจะใช้ได้ถึงเวลาใด งานวิจัยได้นำเสนอการกำหนดค่า TTL แบบ Adaptive TTL ซึ่งค่า TTL จะเปลี่ยนไปตามการร้องขอของไคลเอนต์จากโดเมน ในการพิจารณาการร้องขอนั้นตัวแบ่งภาระงานจะพิจารณาจากโดเมน ในงานวิจัยได้นำเสนอการใช้อัลกอริทึม Probabilistic และ Deterministic ร่วมกับการกระจายภาระงานแบบ Round-Robin (RR) และ two-tier Round-Robin (RR2) โดย RR2 นั้นจะมีการแบ่งระดับของโดเมนออกเป็น 2 ระดับคือ Hot Domain และ Normal Domain โดยจะมีการวนจ่ายภาระงานจาก Hot Domain ให้กับเซิร์ฟเวอร์ และจาก Normal Domain แยกกัน ทั้งนี้เพื่อป้องกันไม่ให้ภาระงานจาก Hot Domain ตกอยู่ที่เซิร์ฟเวอร์ใดเซิร์ฟเวอร์หนึ่งมากเกินไป ส่วนวิธี Probabilistic RR2 (PRR2) นั้นจะมีการกำหนดค่า TTL จากภาระงานของโดเมนตามสมการ  $TTL_j = \frac{\lambda_{\max} \overline{TTL}}{\lambda_j}$  โดยที่  $\overline{TTL}$  คือค่า TTL น้อยที่สุด,  $\lambda_{\max}$  และ  $\lambda_j$  คือค่า Hidden Load Weights ที่มากที่สุด และของโดเมน  $j$  สำหรับ Hidden Load Weights นั้นเป็นจำนวนการร้องขอเฉลี่ยของโดเมนนั้นๆ ในช่วงเวลา TTL และวิธี Deterministic RR2 (DRR2) นั้นจะนำค่าภาระงานของเซิร์ฟเวอร์เข้ามาพิจารณาด้วยตามสมการ  $TTL_{ij} = \frac{\lambda_{\max} \overline{TTL}}{\lambda_j} \xi_i \eta$  โดยที่  $TTL_{ij}$  คือค่า TTL สำหรับโดเมน  $j$  และเซิร์ฟเวอร์  $i$  ส่วน  $\xi_i$  ค่าอัตราส่วนความจุของเซิร์ฟเวอร์  $i$  กับความจุของเซิร์ฟเวอร์ที่มากที่สุด ในระบบ ( $\xi_i = \frac{C_i}{C_1}$ ) และ  $\eta = \frac{C_1}{C_N}$  โดยในระบบเซิร์ฟเวอร์ที่ 1 จะเป็นเซิร์ฟเวอร์ที่มีความจุสูงสุด ไล่ลงมาจนถึงเซิร์ฟเวอร์ที่  $N$

[25] ในงานวิจัยได้มีการนำเสนอรูปแบบในการแลกเปลี่ยนข้อมูลสถานะของโหนด และโหนดปลายทางของงานที่จะย้ายไป โดยในบทความได้แบ่งประเด็นต่างๆ ในการแบ่งภาระงานให้เท่าเทียมออกเป็น 4 แบบคือ 1. การพิจารณาโหนดปลายทางสามารถแบ่งออกได้เป็น 3 แบบคือ การเลือกแบบสุ่ม การเลือกแบบไดนามิก (เลือกโหนดตามข้อมูลที่มีในตัวเอง) และการเลือกตามข้อมูลสถานะของโหนดทั้งหมดที่ประกอบด้วย State Probing [26], Bidding [27] และ Drafting 2. รูปแบบการแลกเปลี่ยนข้อมูล แบ่งออกเป็น ไม่มีการแลกเปลี่ยนข้อมูล มีการส่งสัญญาณขอสถานะ (state probing)

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มีการแลกเปลี่ยนข้อมูลตามระยะเวลาที่กำหนด และแลกเปลี่ยนข้อมูลเมื่อสถานะเปลี่ยน 3. การถ่ายโอนงาน ซึ่งส่วนใหญ่จะใช้ค่าเรสโสดในการกำหนดว่าจะถ่ายโอนหรือไม่ ซึ่งค่าเรสโสดนี้มี 2 แบบคือค่าคงที่กับค่าที่เปลี่ยนตามสถานะของระบบ และ 4.การเลือก Task ในการถ่ายโอน แบ่งออกเป็น 2 แบบคือการถ่ายโอนเฉพาะในตอนเริ่มต้น (Non-Preempt) และการถ่ายโอนเมื่อทำงานไปแล้ว (Preempt) ในงานวิจัยได้นำเสนอความสัมพันธ์ระหว่างการเลือกโหนดปลายทางกับวิธีการแลกเปลี่ยนข้อมูล และชี้ให้เห็นข้อเสียของวิธีการที่มีอยู่ในปัจจุบัน จากนั้นก็ได้นำเสนอโครงสร้างข้อมูลเกี่ยวกับโหนดต่างๆ โหนดทุกโหนดจะเก็บรายการของ ProbedNode ที่บอกว่าโหนดใดมีภาระงานน้อยและ NotifiedNode คอยเก็บว่าโหนดใดที่จะต้องแจ้งสถานะเมื่อสถานะมีการเปลี่ยนแปลงจากภาระงานมากเป็นภาระงานน้อย

[28] ได้นำเสนอรูปแบบในการดึงข้อมูลแบบใหม่ Periodic Symmetrically-Initiated (PSI) สำหรับระบบแบ่งภาระงาน ในอดีตนั้นมี วิธี Random Algorithm เมื่อภาระงานมากกว่าค่าเรสโสดก็จะส่งภาระงานไปให้กับโหนดอื่นแบบสุ่ม วิธี Sender Algorithm เมื่อโหนดมีภาระงานเกินค่าเรสโสดก็จะส่งเมสเสจไปถามโหนดอื่นทุกโหนด ถ้าโหนดใดมีภาระงานน้อยกว่าเรสโสดก็จะตอบรับภาระงาน วิธี Receiver [29] เมื่อโหนดมีภาระงานน้อยกว่าเรสโสด จะส่งเมสเสจไปถามโหนดอื่นทุกโหนด ว่ามีภาระงานมากเกินกว่าเรสโสดหรือไม่ถ้ามีก็จะส่งภาระงานมาให้ และแบบ Symmetric Algorithm [30] เป็นแบบ Sender และ Receiver ผสมกัน แต่มีข้อเสียคือจำเป็นต้องสื่อสารข้อมูลมาก ส่วนวิธี PSI นั้นเมื่อภาระงานมากหรือน้อยกว่าค่าเรสโสดก็จะส่งเมสเสจไปบอกโหนดใดโหนดหนึ่งแบบสุ่มว่าต้องการรับ หรือส่งภาระงานมาให้หรือไม่ และโหนดแต่ละโหนดจะมีการแลกเปลี่ยนข้อมูลเป็นระยะๆ โดยมีการทดลองว่าคาบเวลาเท่าใดจะมีผลอย่างไรต่อระบบด้วย

[31] กล่าวถึงการกำหนดงานให้กับเซิร์ฟเวอร์โดยใช้วิธี Artificial Neural Network (ANN) ซึ่งมีพื้นฐานมาจากทฤษฎี Neural Network โดยระบบจะมีการคาดเดาภาระงานที่จะเข้ามาในอนาคต พร้อมทั้งทำการคำนวณว่าจะกำหนดภาระงานให้กับโหนดใดในระบบจึงจะทำให้ภาระงานในระบบมีความเท่าเทียมมากที่สุด โดยระบบจะมีการเรียนรู้จากตัวอย่างที่ผู้วิจัยสร้างขึ้น ระบบจะเรียนรู้โดยพิจารณาจากพารามิเตอร์คือ เวลาว่างของซีพียู หน่วยความจำที่เหลืออยู่ ความเร็วของระบบเครือข่าย ความยาวของคิวงาน และความแตกต่างระหว่างอัตรางานที่ทำเสร็จกับอัตราที่งานเข้ามา

[32] กล่าวถึงการแบ่งภาระงานสำหรับเซิร์ฟเวอร์ให้บริการสื่อแบบต่อเนื่องโดยมีการแชร์ภาระงานแบบอะแดปทีฟและมีการจัดตารางเวลาของการใช้งานสื่อ ระบบมีส่วนประกอบคือ Archive Server ทำหน้าที่เก็บสื่อทั้งหมดของระบบ Global Scheduler ทำหน้าที่จัดตารางเวลาให้กับบริการของไคลเอนต์ต่างๆ Media Server ทำหน้าที่ให้บริการสื่อกับไคลเอนต์ และไคลเอนต์ซึ่งทำหน้าที่ร้องขอบริการจากเซิร์ฟเวอร์ ในงานวิจัยนี้ได้นำเสนอ Two-Phase List Scheduling scheme (TLS) ซึ่งทำหน้าที่กำหนด Task และตอบรับการร้องขอของไคลเอนต์ โดยแบ่งการทำงานออกเป็น 2 ระดับในระดับแรกนั้นจะมีการกำหนด Upper Threshold และ Lower Threshold ตามความจุของเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เซิร์ฟเวอร์ เมื่อภาระงานของเซิร์ฟเวอร์เกิด Upper Threshold เซิร์ฟเวอร์ก็จะทำหน้าที่เป็น Sender-Initiate และเมื่อภาระงานต่ำกว่า Lower Threshold ก็จะทำหน้าที่เป็น Receiver-Initiate ในระดับที่สองจะมีการเลือกลำดับการร้องขอใช้สื่อของผู้ใช้ตามกระบวนการ 6 รูปแบบคือ 1.เซิร์ฟเวอร์จะทำหน้าที่เป็น Receiver เมื่อภาระงานต่ำกว่า Lower Threshold, 2. พิจารณา 1 ก่อนแล้วดูเซิร์ฟเวอร์ที่มีคิวของ Task น้อยที่สุด, 3. พิจารณา 2 จากนั้นเลือกเซิร์ฟเวอร์ที่มีจำนวนสื่อที่โคลเอ็นต์ต้องการมากที่สุด, 4.พิจารณา 2 ถึง 3 แล้วเลือกเซิร์ฟเวอร์ที่มีภาระงานน้อยที่สุด, 5. พิจารณา 2 ถึง 4 แล้วเลือกเซิร์ฟเวอร์ที่ติดต่อกับโคลเอ็นต์ได้เร็วที่สุด และ 6. พิจารณา 2 ถึง 6 แล้วเลือกเซิร์ฟเวอร์ที่สามารถจัดตารางเวลาการให้บริการต่อโคลเอ็นต์ได้ดีที่สุด

[1] ได้นำเสนอวิธีในการกระจายภาระงานที่ได้ปรับปรุงจากวิธี Bidding สำหรับวิธี Bidding นั้นคือเมื่อโนดต้องการส่งภาระงานไปให้กับโนดอื่นก็จะมีการส่งแมสเซจออกไป โนดที่ได้รับแมสเซจก็จะตอบกลับพร้อมทั้งภาระงานของตน โนดต้นทางก็จะพิจารณาเลือกโนดพร้อมทั้งส่งภาระงานไปให้ ในวิธีการนี้ เมื่อมีภาระงานเข้ามา ตัวระบบจะตัดสินใจว่าจะปฏิบัติภาระงานเองหรือจะส่งให้โนดอื่น (Transfer Policy) โดยพิจารณาจากค่าเรสโสลดโดยค่าเรสโสลดของโนด  $j$  คือ  $T_j = (\frac{S_j}{S_i})(T_i + 1) - 1$  โดยที่  $s$  คือ Service Time และ  $T_i$  คือค่าเรสโสลดของโนด  $i$  จากนั้นก็จะมีการแลกเปลี่ยนข้อมูล (Information Policy) กัน ในงานวิจัยนี้แบ่งการแลกเปลี่ยนข้อมูลแบบ Bidding เป็น 3 แบบคือ BR (Broadcast) จะมีการกระจายแมสเซจไปให้กับทุกโนด แบบ LB (Limited Broadcast) จะมีการส่งแมสเซจเฉพาะบางโนด โดยเลือกจากการสุ่ม และแบบ LBT (Limited Broadcast with Time windows) จะเป็นเหมือนวิธี LB โดยมีการเก็บข้อมูลเวลาไว้ว่าถ้าโนดใดได้รับการงานไปแล้วในช่วงเวลาที่กำหนดไว้จะไม่ส่งภาระงานไปให้กับโนดนั้นอีก เมื่อได้ข้อมูลแล้วโนดก็จะทำการเลือกโนดปลายทางโดยเลือกโนดที่ตอบกลับมามีภาระงานน้อยที่สุด ผลจากการทดลองในงานวิจัยแสดงให้เห็นว่าวิธีการแลกเปลี่ยนข้อมูลแบบ LB จะสามารถแบ่งภาระงานได้ดีกว่าวิธี LBT ในกรณีที่มีการสุ่มเลือกโนดไม่มาก (3-4 โนด) และทั้งวิธี LB และ LBT นั้นให้แบ่งภาระงานได้ดีกว่าวิธี Eager's Shortest ซึ่งสามารถสรุปได้ว่าการใช้วิธีการที่ง่ายกว่าอาจจะช่วยให้การแบ่งภาระงานในระบบดีขึ้น

[33] ได้กล่าวถึงการออกแบบและประเมินผลอัลกอริทึมในการแชร์ภาระงานในระบบที่มีการทำงานแบบเรียลไทม์กระจาย ในงานวิจัยทดลองใช้รูปแบบในการแลกเปลี่ยนข้อมูลภายใน Buddy Set เพื่อลด Overhead ที่เกิดขึ้นเนื่องจากการสื่อสารโดยได้ลองใช้วิธี State Probing ที่โนดจะสุ่มเก็บสถานะของโนดอื่น, Focused Addressing จะมีการกระจายสถานะเป็นระยะและเมื่อต้องการส่ง Task ไปให้ก็จะทำการหา Focused Node โดยสุ่มเลือกจากกลุ่มของโนดที่สามารถรองรับภาระงานได้ ในงานวิจัยใช้กระบวนการตัดสินใจในการส่งภาระงานไปให้บนวิธี Minimum-Laxity First-Served (MLFS) ซึ่งจะพิจารณาว่าถ้าภาระงานที่เข้ามาใหม่สามารถปฏิบัติให้เสร็จตามค่า Lexity ซึ่งเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นค่า Deadline ที่ประมาณไว้ได้หรือไม่ ถ้าไม่ได้ก็จะทำการส่งภาระงานนั้นไปให้กับโนคอื่น สำหรับการตัดสินใจส่งภาระงานไปให้กับโนคใดนั้นใช้กระบวนการตัดสินใจ Bayesian Decision Model ซึ่งมีการประมาณข้อมูล Prior ก่อนการได้รับภาระงาน และ Posterior หลังจากได้รับภาระงานไปแล้ว โนคจะทำการส่งแพกเกต Time Stamp ออกไป โนคที่ได้รับก็จะนำแพกเกต Time Stamp นี้มาประมาณข้อมูล Prior และ Posterior และให้ประกอบการตัดสินใจในการส่งภาระงานไปให้

[2] ได้นำเสนออัลกอริทึมในการแบ่งภาระงานแบบไดนามิก คือ Diffusion Method อัลกอริทึมนี้ไม่จำกัดสถาปัตยกรรมในการเชื่อมต่อของโนค เช่นอาจจะเป็นแบบ Mesh, Linear หรือ Hypercube ก็ได้ ในบทนำได้กล่าวไว้ว่าการแบ่งภาระงานจากศูนย์กลางนั้นมีความถูกต้องของข้อมูลมากกว่าแต่ก็จะทำให้เกิด Overhead มากกว่า แต่การแบ่งภาระงานแบบกระจายนั้นถึงแม้ว่าจะมีความถูกต้องของข้อมูลน้อยกว่าแต่ก็มี Overhead ที่น้อยกว่าด้วย ในการแบ่งภาระงานด้วยวิธี Diffusion ปกติ (Simple Diffusion Method, SDM) นั้น โนคจะติดต่อกับโนคที่อยู่ข้างเคียง (Neighbor Node) โดยถ้ามีการตรวจพบว่าโนคกำลังจะ Overload ก็จะมีการส่งภาระงานไปให้กับโนคข้างเคียง โดยพิจารณาจากโนคข้างเคียงที่มีภาระงานน้อยกว่า และในกรณีที่โนคข้างเคียงที่ได้รับภาระงานไปมีโนคข้างเคียงที่มีภาระงานน้อยกว่า โนคที่ได้รับภาระงานไปก็จะทำการส่งภาระงานไปให้กับโนคข้างเคียงอีกทีหนึ่ง การถ่ายโอนภาระงานจะทำเช่นนี้ไปเรื่อยๆ จนกว่าภาระงานของแต่ละโนคมีผลต่างจากภาระงานเฉลี่ยของระบบน้อยที่สุด หลักการที่งานวิจัยได้นำเสนอนั้นเป็น Diffusion Algorithm Searching Unbalanced Domains, DASUD) ที่ได้มีการแบ่งกลุ่มของโนค และแต่ละโนคจะมีส่วนที่ Overlapping กัน โนคที่ Overlap นี้จะทำหน้าที่ในการแลกเปลี่ยนข้อมูลและภาระงานระหว่างกลุ่มของโนคใกล้เคียง ในการถ่ายโอนภาระงานนั้น โนคที่พบว่ามีการะงานมากกว่าค่าภาระงานเฉลี่ยของระบบจะทยอยส่งภาระงานของตนเองไปให้กับโนคข้างเคียง โนคข้างเคียงที่ได้รับภาระงานก็จะตรวจสอบและถ่ายโอนภาระงานเช่นเดียวกันไปเรื่อยๆ จนกว่าจะไม่พบโนคที่มีการะงานไม่เท่าเทียมกับโนคอื่นๆ ในระบบ ข้อดีของวิธีนี้ก็คือทำให้ช่วยลดการสื่อสารข้อมูลในระบบ เพราะโนคจะสื่อสารข้อมูลกับโนคข้างเคียงเท่านั้น และยังสามารถเพิ่มโนคเข้าไปในระบบได้ง่ายกว่าการแบ่งภาระงานจากศูนย์กลางอีกด้วย

[34] เป็นวิธีในการหาโนคสำหรับรับภาระงาน (Location Policy) โดยการทำงานของ การแบ่งภาระงานนั้นจะเป็นแบบอะแดปทีฟวิธีการนี้จะแบ่งโนคออกเป็น 2 แบบคือ Sender Receiver และ OK ในโนคจะมีคิวสำหรับเก็บว่าโนคใดในระบบเป็นชนิดใด โนคที่เปลี่ยนสถานะใหม่จะใส่เข้าไปทางหัวคิว วิธีการในการแบ่งภาระงานนั้นประกอบไปด้วย วิธี Sender-Initial โดยโนคที่มีการะงานเกินจะทำการสุ่ม Probe ว่าโนคใดมีการะงานน้อยที่สุดเพื่อถ่ายโอนภาระงานไปให้ เมื่อได้รับเมสเสจมาโนคนั้นก็จะอัปเดตข้อมูลที่ได้รับเข้ามา วิธี Receiver-Initial โนคที่มีการะงานน้อยจะ

ทำการสุ่มว่าโนคใดมีภาระงานมากที่สุดเพื่อขอรับภาระงานมาทำ โดยจะเริ่มตรวจสอบจากโนคแรกของคิว Sender ก่อน ตามด้วยโนคสุดท้ายของคิว OK ตามด้วยโนคสุดท้ายของ Receiver

[35] ได้พัฒนาระบบสำหรับให้บริการอินเทอร์เน็ตที่ประกอบด้วย HTTP, DNS และ Gopher โดยใช้กลไกของเครื่องพีซี ในระบบจะแบ่งออกเป็น Internet Request Distributor (IRD) ทำหน้าที่รับการร้องขอใช้บริการจากอินเทอร์เน็ตและแจกจ่ายไปให้กับเซิร์ฟเวอร์ในระบบ โดยพิจารณาตามบริการ และภาระงานของเซิร์ฟเวอร์ ในงานวิจัยได้นำเสนอ Dynamic Port Forwarding (DPF) ซึ่งเป็นการกำหนดการร้องขอให้กับเซิร์ฟเวอร์ในระบบโดยพิจารณาจากภาระงาน โดยภาระงานในระบบนั้นจะวัดจาก Relative Performance Rating ที่ตัว IRD สำหรับเซิร์ฟเวอร์ในระบบนั้นจะติดต่อกับอินเทอร์เน็ตผ่านตัว IRD เท่านั้นส่งผลให้ระบบสามารถขยายขนาดได้ง่ายเมื่อมีเซิร์ฟเวอร์เพิ่มขึ้นมาเซิร์ฟเวอร์นั้นก็จะส่งเมสเสจไปบอก IRD ตัว IRD ก็จะตอบกลับ จากนั้นเซิร์ฟเวอร์ก็จะบอกว่าตัวเองให้บริการอะไรบ้าง ตัว IRD ก็จะเก็บข้อมูลไว้เพื่อใช้ในการกำหนดการร้องขอใช้บริการให้กับเซิร์ฟเวอร์ต่อไป นอกจากการที่มีเฉพาะ IRD เชื่อมต่อกับอินเทอร์เน็ตเท่านั้น ยังช่วยให้ระบบมีความปลอดภัยมากขึ้นเพราะผู้ใช้สามารถให้ IRD ทำหน้าที่เป็นไฟร์วอลล์ (Firewall) ได้ ในงานวิจัยได้นำเสนอว่าระบบมีความเชื่อถือได้สูงกว่าเพราะมีการสำรองอากรให้บริการ ตัว IRD จะคอยตรวจสอบเซิร์ฟเวอร์อยู่เป็นระยะโดยใช้วิธี Automatic Failure Notification (AFN) ถ้าเซิร์ฟเวอร์ไม่ตอบกลับก็แสดงว่าเซิร์ฟเวอร์ไม่สามารถให้บริการ และเมื่อกลับมาให้บริการได้ตัว IRD ก็จะตรวจสอบได้ด้วยวิธีเดียวกัน ในอนาคตผู้วิจัยจะทำการพัฒนาให้ระบบสามารถรองรับการให้บริการที่กระจายตามสถานที่ซึ่งสามารถช่วยแบ่งเบาภาระในด้านของระบบเครือข่ายลงได้

[36] ได้นำเสนอรูปแบบในการแบ่งอัลกอริทึมสำหรับการกระจายภาระงานแบบใหม่ โดยแบ่งเป็น แบ่งตามวัตถุประสงค์ (Objective) แบ่งตามข้อมูลที่ใช้ แบ่งตามแหล่งในการกระจายภาระงาน แบ่งตามเวลาในการกระจายภาระงาน และแบ่งตามผู้เริ่มต้นทำการแบ่งภาระงาน ในบทความได้นำเสนอว่าอัลกอริทึมในการกระจายภาระงานนั้นแบ่งออกได้เป็น 3 ส่วนคือ Load Model, Action Model และ Solution Model โดย Load Model นั้นจะอธิบาย Workload ของระบบ รวมทั้งข้อมูลที่เกี่ยวข้องกับภาระงานเช่น ค่าจำกัดความของภาระงาน ความสัมพันธ์ระหว่างภาระงาน และการทำงานของภาระงานเป็นต้น สำหรับ Action Model นั้นจะอธิบายถึงการกระทำเพื่อการกระจายภาระงานดังเช่น วิธีการแลกเปลี่ยนข้อมูล และการถ่ายโอนงานไปให้กับโนคอื่นๆ เป็นต้น และ Solution Model จะอธิบายว่าการกระจายภาระงานจะทำได้อย่างไร หรือมีการตัดสินใจในการกระจายภาระงานอย่างไร ดังเช่นการกำหนดปลายทางของงาน เป็นต้น นอกจาก Solution Model ก็ยังมีอัลกอริทึม Solution Making Procedure ซึ่งเป็นกระบวนการในการบ่งบอกว่าจาก Solution Model นั้นจะมีวิธีการตัดสินใจในการกระจายภาระงานเป็นอย่างไร สำหรับการแบ่งประเภทของการกระจายภาระงานโดยแบ่งตามวัตถุประสงค์นั้นสามารถแบ่งได้ ประสิทธิภาพ ระยะทางระหว่างเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

งานกับข้อมูลที่ต้องใช้ ประสิทธิภาพในการให้บริการ ความเท่าเทียมของภาระงาน และความคุ้มค่าในการใช้ทรัพยากรของระบบ (Resource Utilization) สำหรับการแบ่งตามข้อมูลนั้นสามารถแบ่งได้เป็นแหล่งที่ให้ข้อมูล และชนิดของข้อมูลที่ใช้ในการกระจายภาระงาน ส่วนการแบ่งตามแหล่งในการกระจายภาระงาน สามารถแบ่งได้เป็นประเภทย่อยๆ คือการกระจายภาระงานจากศูนย์กลาง หรือการกระจายภาระงานแบบกระจาย ทั้งนี้จะต้องพิจารณาถึงการกระจายในทางกายภาพและทางตรรกด้วย ประเภทในการกระจายภาระงานตามผู้เริ่มต้นสามารถแบ่งได้เป็น การส่งภาระงานเริ่มต้นจากศูนย์กลาง มีการกระจายภาระงานแบบกระจาย หรือมีการร้องขอการกระจายภาระงาน และการร้องขอการกระจายภาระงานจากศูนย์กลาง ในบทความยังได้นำเสนอการนำทฤษฎีความน่าจะเป็นมาใช้ในการกระจายภาระงานอีกด้วย

[37] ได้นำเสนอโมเดลและวิธีการประเมินประสิทธิภาพของอัลกอริทึมของ Load Balancing แบบ Dynamic, Distributed และ Sender-Initiate ที่มีการแบ่งภาระงานเป็นแบบ Iteration คือจะมีวงรอบในการแบ่งภาระงานในระหว่างวงรอบจะเป็น Synchronized Point สำหรับแลกเปลี่ยนข้อมูลระหว่างโนด โมเดลประกอบด้วยเมตริกที่ใช้ในการอธิบายว่าโนดสามารถส่งภาระงานไปให้กับโนดใดได้บ้าง พร้อมทั้งมีการกำหนดจำนวนภาระงานที่จะส่งและค่าเรสโสดด้วย ในการประเมินประสิทธิภาพใช้การวัดจากเวลาในการทำงานร่วมของระบบ ( $T_{Total}$ ) และความเร็วที่เพิ่มขึ้น ( $SpeedUp(n) = \frac{T_{Total\_on\_one\_processor}}{T_{Total\_on\_n\_processor}}$ ) ในงานวิจัยได้ทดลองใช้อัลกอริทึม Load Balancing แบบ Random และ Redistribution โดยอัลกอริทึมแบบ Random นั้นจะสุ่มเลือกโนดที่สามารถส่งภาระงานไปให้ได้ ส่วนวิธี Redistribution นั้นจะมีการแบ่งภาระงานในแต่ละรอบของ Iteration สำหรับโนดที่มีภาระงานเกินค่าเรสโสดโดยจะเลือกโนดปลายทางที่ส่งภาระงานไปให้แล้วโนดนั้นมีภาระงานใกล้เคียงกับค่าภาระงานเฉลี่ยมากที่สุด จากการทดลองพบว่า ประสิทธิภาพที่ได้จากโมเดล และการทดลองนั้นมีความแตกต่างกันอยู่ไม่เกิน 20 % โดยความผิดพลาดที่เกิดขึ้นเนื่องจากความไม่แน่นอนของระบบเครือข่าย ส่วนพารามิเตอร์ที่เกี่ยวข้องกับอัลกอริทึมนั้นคือค่า  $p$  เป็นค่าที่ใช้บอกว่าจะย้ายภาระงานจำนวนเท่าใดและ  $w^*$  ที่เป็นค่าเรสโสดโดยถ้าค่า  $p$  น้อยจะทำให้การแบ่งภาระงานไม่ค่อย Balance แต่ถ้าค่า  $p$  มากก็จะทำให้เกิด Overhead เนื่องจากการสื่อสารมาก ส่วน  $w^*$  ถ้ามีค่ามากจะทำให้มีการถ่ายโอนภาระงานน้อยจะไม่ค่อย Balance แต่ถ้า  $w^*$  น้อยก็จะทำให้มีการแบ่งภาระงานมากเกินไป

[38] ได้นำเสนอการจำแนกประเภทของระบบที่ประมวลผลแบบกระจายโดยศึกษาจากงานวิจัยที่ผ่านมา ในการแบ่งนั้นผู้แต่งได้แบ่งการจำแนกออกเป็น 2 แบบคือ Hierarchical และ Flat สำหรับการจำแนกแบบ Hierarchical นั้นจะมีการจำแนกประเภทของระบบประมวลผลแบบกระจายตามลำดับชั้นที่ชัดเจน ส่วนแบบ Flat นั้นจะเป็นการแบ่งตามคุณสมบัติของระบบที่เกี่ยวข้องกับการประมวลผลแบบกระจาย ในบทความได้ให้ความหมายของการวัดระบบไว้ 2 แบบคือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Performance ซึ่งเป็นมุมมองจากทรัพยากรว่าระบบทำงานได้ดีเพียงใด และ Efficiency ซึ่งเป็นมุมมองของผู้ใช้ระบบว่าระบบมี Cost เท่าใดในการทำงาน โดยปกติ Performance และ Efficiency นั้นจะเป็น Trade-Off กัน ความแตกต่างระหว่าง Scheduling และ Allocation โดย Scheduling เป็นมุมมองของการจัดสรรทรัพยากรจากผู้ใช้ ส่วน Allocation เป็นมุมมองของทรัพยากรเอง ในการจำแนกแบบ Hierarchical นั้นจะแบ่งได้เป็นลำดับชั้นที่สำคัญคือ Optimal และ Suboptimal โดยแบบ Optimal นั้นระบบจะรู้ข้อมูลเกี่ยวกับสถานะของระบบรวมทั้งทรัพยากรที่เกี่ยวข้องกับการประมวลผลทั้งหมด และจะมีการตัดสินใจโดยเลือกจาก Criterion Function ที่เหมาะสมด้วย Approximate และ Heuristic โดยแบ่ง Approximate นั้นจะมีโมเดลในการประมวลผลแบบ Formal ในการประมาณการทำงานเพื่อหาการแบ่งงานที่ดีที่สุด ส่วนแบบ Heuristic จะมีการคาดเดาที่ใช้ในการประมวลผลงานของระบบ และแบบ Cooperative ซึ่งการแบ่งงานจะมีการติดต่อระหว่างโนด ส่วนแบบ Noncooperative นั้นจะไม่มี การติดต่อกัน สำหรับการจำแนกด้วยวิธี Flat ที่สำคัญคือ Probabilistic จะมีการจำลองวิธีที่ดีที่สุดในการกระจายงาน ในงานวิจัยได้นำตัวอย่างอัลกอริทึมในการแบ่งงานมาจำแนกตามวิธีที่ได้นำเสนอเป็นจำนวนมาก ในส่วนสุดท้ายของบทความได้นำเสนอการจำแนกอื่นๆ ที่เกี่ยวข้องสำหรับในอนาคตอีกด้วย สิ่งที่น่าสนใจก็คือการกำหนดเมตริกของอัลกอริทึมโดยถ้าเป็นเมตริกในมุมมองของผู้ใช้ส่วนใหญ่จะวัดจาก Response Time แต่ถ้าเป็นมุมมองของระบบส่วนใหญ่จะวัดจาก Throughput ซึ่งเกี่ยวข้องกับความเสมอภาคในการให้บริการต่อผู้ใช้และ Utilization ของระบบ

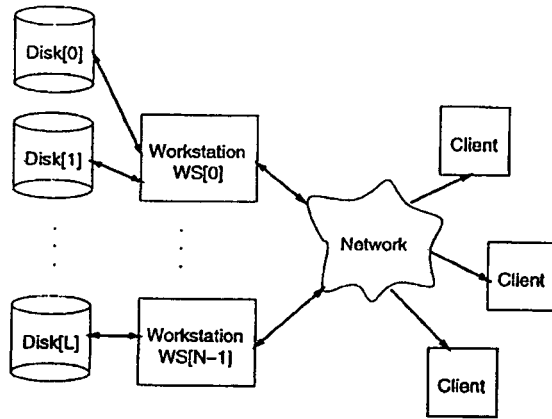
[39] เป็นงานวิจัยที่เกี่ยวข้องกับการวิเคราะห์ Overhead ที่เกิดขึ้นเนื่องจากการทำ Load Distribution ในงานวิจัยได้แบ่งประเภทของการทำ Load Distribution ออกเป็น Placement หรือ Nonpreemptive คือโพรเซสจะถูกย้ายไปทำงานยังเครื่องอื่นๆ ตั้งแต่โพรเซสยังไม่ได้เริ่มต้นทำงาน และ Migration หรือ Preemptive ซึ่งโพรเซสสามารถถูกส่งไปให้กับเครื่องอื่นๆ ได้เมื่อได้ทำงานไปแล้ว ทั้งนี้กระบวนการ Placement นั้นจำเป็นต้องใช้ทรัพยากรมากกว่า แต่ไม่ต้องเสียเวลาในการกำหนดสถานะการทำงานให้กับโพรเซสที่ทำงานไปแล้ว แต่ Migration นั้นจะต้องใช้ทรัพยากรในการดึงสถานะของโพรเซสให้เป็นเหมือนเดิม สำหรับการวัดนั้นงานวิจัยพิจารณาประสิทธิภาพจาก Response Time และ Response Ratio ซึ่งเป็นอัตราส่วนระหว่าง Response Time กับ Service Time ที่บ่งบอกความคุ้มค่าที่ระบบเสียเวลาไปกับกระบวนการ Load Distribution มากน้อยเท่าใด สำหรับอัลกอริทึมที่งานวิจัยนี้ใช้นั้นเป็นแบบ Sender-Initiate, Receiver-Initiate และ Symmetric-Initiate โดยพิจารณาว่าอัลกอริทึมใดมีประสิทธิภาพเป็นอย่างไร นอกจากนั้นยังพิจารณาถึงพื้นที่ในการเก็บข้อมูลที่ใช้สำหรับเก็บโพรเซสที่จำเป็นต้องทำงานด้วย จากการทดลองพบว่า กระบวนการ Placement นั้นจะเพิ่มประสิทธิภาพสำหรับระบบที่มีสถานะของภาระงานค่อนข้างมาก ส่วนกระบวนการ Migration นั้นช่วยเพิ่มประสิทธิภาพสำหรับระบบได้สำหรับระดับของภาระงานที่หลากหลาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

[40] ได้นำเสนอสถานะแวดล้อมในการเขียนโปรแกรมแบบ Object-Based ชื่อว่า MOM (Mutual Object Machine) พร้อมทั้งได้กำหนด Load Balancing Algorithm ที่มีชื่อว่า Gating ใหม่ ขึ้นมาพร้อมเปรียบเทียบกับอัลกอริทึมที่มีอยู่โดยการจำลองการทำงานบนระบบ MOM ในระบบ MOM นั้นจะมีออบเจกต์ Office ที่ทำหน้าที่เป็นตัวกลางในการส่งเมสเสจระหว่างออบเจกต์ในระบบ โดยออบเจกต์จะเรียกเมธอด Send ที่ออบเจกต์ Office พร้อมด้วยปลายทาง และเมื่อพร้อมแล้วออบเจกต์ Office ก็จะส่งเมสเสจไปให้กับปลายทาง สำหรับอัลกอริทึม Gating นั้นงานวิจัยได้แบ่งส่วนประกอบของอัลกอริทึมออกเป็น Transmission Strategy จะพิจารณาจากค่าเรสโสดกับคิวของโนด Target Strategy จะพิจารณาจากโนดที่มีจำนวน Task น้อยกว่าค่าเรสโสด Information Strategy จะมีการส่งข้อมูลภาระงานไปพร้อมกับภาระงานที่ส่งไป และมีการส่งเมื่อข้อมูลมีการเปลี่ยนแปลงมากกว่าที่กำหนดไว้และ Local Service Strategy จะเป็นแบบ First-Come First-Serve จากการเปรียบเทียบกับ Load Balancing Algorithm อื่นนั้น แสดงให้เห็นว่าอัลกอริทึม Gating จะดีกว่าอัลกอริทึมอื่น แต่ก็มีข้อเสียคือจะทำให้เกิด Overhead เนื่องจากการสื่อสารมาก

## 2.9 งานวิจัยที่เกี่ยวข้องกับอัลกอริทึมการแบ่งภาระงานสำหรับเซิร์ฟเวอร์ที่ให้บริการสื่อแบบต่อเนื่อง

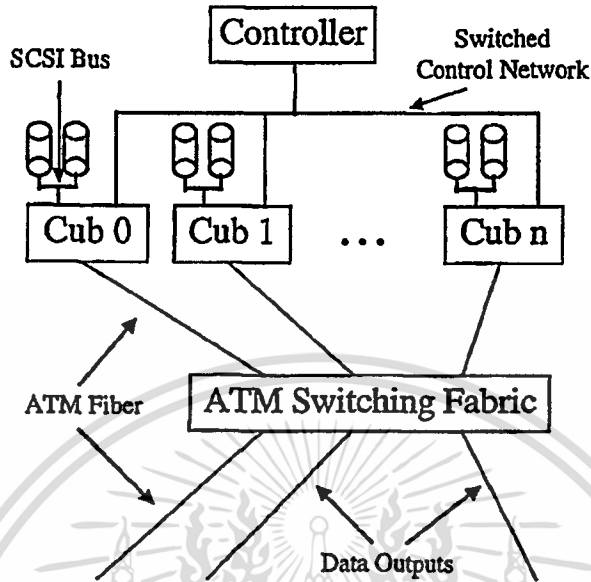
ใน [41] ได้นำเสนออัลกอริทึมในการแบ่งภาระงานสำหรับเซิร์ฟเวอร์ให้บริการสื่อผสมโดยอาศัยเทคนิคในการ Replication ข้อมูลที่มีชื่อว่า MMPacking และการกำหนดภาระงานโดยใช้ อัลกอริทึมในการจัดตารางเวลาแบบถ่วงน้ำหนัก (Weighted Scheduling Algorithm) วิธี MMPacking นั้นจะมีการวางข้อมูลของวิดีโอเรียงกันไปตามเซิร์ฟเวอร์โดยมีเงื่อนไขว่าจะต้องไม่มีข้อมูลซ้ำบนเซิร์ฟเวอร์เดียวกัน และจำนวนของข้อมูลบนเซิร์ฟเวอร์แต่ละตัวจะต่างกันไม่เกิน 2 ชิ้น ส่วนการกำหนดภาระงานให้แก่เซิร์ฟเวอร์นั้นจะพิจารณาจากความน่าจะเป็นที่ข้อมูลบนเซิร์ฟเวอร์ทั้งหมดจะถูกเรียกใช้ ในกรณีที่เซิร์ฟเวอร์มากกว่า 2 ตัวมีความน่าจะเป็นเท่ากัน ระบบจะใช้วิธี Round-Robin ในการกำหนดภาระงาน งานวิจัยนี้เกี่ยวข้องกับการแบ่งภาระงานแบบสแตติกโดยพิจารณาตามข้อมูลที่อยู่บนเซิร์ฟเวอร์โดยไม่ได้พิจารณาถึงปัจจัยอื่นๆ ในขณะรันไทม์ ดังเช่น Utilization ทรัพยากรของเซิร์ฟเวอร์เป็นต้น สำหรับโครงสร้างของ MMPacking นั้นแสดงอยู่ในรูปที่ 2.9



รูปที่ 2.9 โครงสร้างของระบบที่ใช้อัลกอริทึมในการแบ่งภาระงาน MM Packing [41]

ใน [42] ได้กล่าวถึงการทำงานของวิดีโอเซิร์ฟเวอร์ Tiger ของบริษัทไมโครซอฟท์ซึ่งมีโครงสร้างแสดงอยู่ในรูปที่ 2.10 วิดีโอเซิร์ฟเวอร์ Tiger ได้รับการออกแบบมาให้มีความสามารถหลัก 3 ประการ คือความสามารถในการกระจายการให้บริการข้อมูลต่อผสม ความสามารถในการขยายขนาด (Scalability) และความสามารถในการรองรับความเสียหาย (Fault Tolerance) ระบบจะมี Cub ซึ่งเป็นคอมพิวเตอร์ที่เก็บและให้บริการสื่อ ในการเก็บสื่อนั้นจะเก็บแบบ Stripping คือสื่อจะถูกแบ่งออกเป็นส่วนๆ และกระจายเก็บไว้บน Cub ทั้งหมดในระบบ มีตัว Controller ที่ทำหน้าที่ดูแลการร้องขอจากผู้มี Control Network ที่เป็นระบบเครือข่ายภายในสำหรับเชื่อมต่อและสั่งงาน Cub แต่ละตัว และ เครื่องข่ายที่ให้บริการกับไคลเอนต์ภายนอก ในส่วนของการรองรับความเสียหายนั้นตัวระบบใช้โปรแกรม Deadman เพื่อตรวจสอบ Cub ที่อยู่ข้างๆ ว่ายังสามารถทำงานได้หรือไม่ และระบบจะมีการทำสื่อสำรอง โดยจะมีการแบ่งสื่อที่เก็บอยู่ใน Cub (Decuster) โดยแบ่งตามจำนวน Decuster Factor จากนั้นก็จะกระจายเก็บส่วนของสื่อที่ได้แบ่งไว้บนดิสก์ต่างๆ ในการจัดการเวลาของระบบนั้น ระบบจะให้ Cub แต่ละตัวมีข้อมูลในการจัดการเวลาสำหรับ Cub อื่นได้ด้วย (เป็นการจัดการเวลาแบบกระจาย) โดยจุดประสงค์ก็คือจะต้องมีการจัดการเวลาในการให้บริการสื่อของแต่ละ Cub ไม่ให้เกิด Hotspot หรือ Cub ที่มีสถานะ Overload นั้นเอง ซึ่งสามารถทำได้โดยการกำหนดเวลาในการให้บริการสื่อเริ่มต้น ในการแลกเปลี่ยนข้อมูลนั้น Cub จะส่งข้อมูลตารางเวลาของตัวเองและ Cub อื่นๆ ที่ได้รับมาให้กับ Cub ที่อยู่ข้างเคียง 2 Cub เพื่อป้องกัน Cub ที่รับข้อมูลไปเกิดความเสียหาย ในการจัดการเวลานั้นสามารถแบ่งได้สองแบบคือ Disk Schedule สำหรับการให้บริการแบบ Single Bitrate และ Network Schedule Bitrate โดยจะเกี่ยวข้องกับการเพิ่มและลบการให้บริการของ Viewer ออก ในการลบ (Deschedule) การให้บริการนั้นสามารถทำได้ทันทีจากนั้น Cub ก็จะส่งเมสเสจไปบอก Cub อื่นๆ ด้วย ส่วนการเพิ่มการให้บริการนั้นจะทำให้ยากกว่า เพราะ Cub ที่มีส่วนแรกของสื่อจะต้องรอข้อมูลการจัดการเวลามาให้ครบก่อนซึ่งเวลาใน

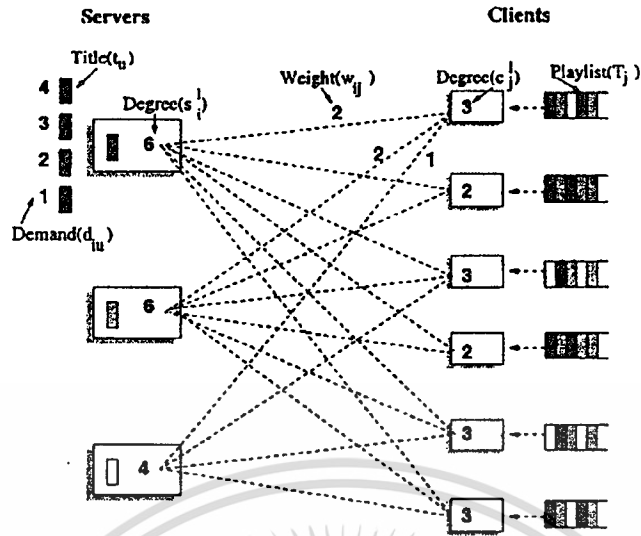
ช่วงนี้เราจะกำหนดให้ Cub เป็นเจ้าของ Slot ถ้า Slot ว่าง (ทุก Cub สามารถรับภาระงานได้)ตัว Cub ก็จะเริ่มให้บริการ



รูปที่ 2.10 โครงสร้างของวิดิโอเซิร์ฟเวอร์ Tiger [42]

ใน [43] กล่าวถึงการแบ่งภาระงานสำหรับเซิร์ฟเวอร์ให้บริการสื่อแบบต่อเนื่องโดยมีการแชร์ภาระงานแบบอะแดปทีฟและมีการจัดตารางเวลาของการใช้งานสื่อ ระบบมีส่วนประกอบคือ Archive Server ทำหน้าที่เก็บสื่อทั้งหมดของระบบ Global Scheduler ทำหน้าที่จัดตารางเวลาให้กับบริการของไคลเอนต์ต่างๆ Media Server ทำหน้าที่ให้บริการสื่อกับไคลเอนต์ และไคลเอนต์ซึ่งทำหน้าที่ร้องขอบริการจากเซิร์ฟเวอร์ ในงานวิจัยนี้ได้นำเสนอ Two-Phase List Scheduling scheme (TLS) ซึ่งทำหน้าที่กำหนด Task และตอบรับการร้องขอของไคลเอนต์ โดยแบ่งการทำงานออกเป็น 2 ระดับในระดับแรกนั้นจะมีการกำหนด Upper Threshold และ Lower Threshold ตามความจุของเซิร์ฟเวอร์ เมื่อภาระงานของเซิร์ฟเวอร์เกิด Upper Threshold เซิร์ฟเวอร์ก็จะทำหน้าที่เป็น Sender-Initiate และเมื่อภาระงานต่ำกว่า Lower Threshold ก็จะทำหน้าที่เป็น Receiver-Initiate ในระดับที่สองจะมีการเลือกลำดับการร้องขอใช้สื่อของผู้ใช้ตามกระบวนการ 6 รูปแบบคือ 1.เซิร์ฟเวอร์จะทำหน้าที่เป็น Receiver เมื่อภาระงานต่ำกว่า Lower Threshold, 2. พิจารณา 1 ก่อนแล้วดูเซิร์ฟเวอร์ที่มีคิวของ Task น้อยที่สุด, 3. พิจารณา 2 จากนั้นเลือกเซิร์ฟเวอร์ที่มีจำนวนสื่อที่ไคลเอนต์ต้องการมากที่สุด, 4.พิจารณา 2 ถึง 3 แล้วเลือกเซิร์ฟเวอร์ที่มีภาระงานน้อยที่สุด, 5. พิจารณา 2 ถึง 4 แล้วเลือกเซิร์ฟเวอร์ที่ติดต่อกับไคลเอนต์ได้เร็วที่สุด และ 6. พิจารณา 2 ถึง 6 แล้วเลือกเซิร์ฟเวอร์ที่สามารถจัดตารางเวลาการให้บริการต่อไคลเอนต์ได้ดีที่สุด โดยโครงสร้างของการจัดการตารางเวลาแสดงอยู่ในรูปที่ 2.11

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

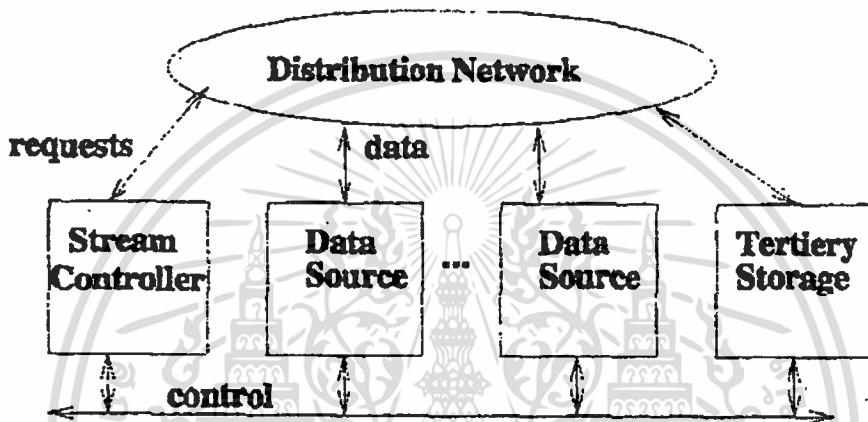


รูปที่ 2.11 โครงสร้างของระบบที่มีการแบ่งภาระงานแบบอะแคพทีฟ [43]

ใน [44] ได้นำเสนอการจัดการกับภาระงานของวิดีโอเซิร์ฟเวอร์ที่มีโครงสร้างดังแสดงอยู่ในรูปที่ 2.12 ในงานวิจัยได้นำเสนอการคาดการณ์ในการวางวิดีโอบนเซิร์ฟเวอร์โดยพิจารณาจากคำตอบแทนของวิดีโอแต่ละเรื่องเป็นหลัก สำหรับสถาปัตยกรรมของระบบนั้นในระบบจะประกอบด้วย Central Distribution Controller (DC) ที่ทำหน้าที่ควบคุมการทำงานของ Data Source พร้อมทั้งอาจจะมี Tertiary Storage ที่ทำหน้าที่เก็บสื่อทั้งหมดที่มีในระบบ สำหรับกระบวนการจัดการภาระงานของระบบนั้นประกอบด้วย การทำ Replication ซึ่งเป็นกระบวนการในการเพิ่มจำนวนสื่อที่ได้รับคามนิยมสูงในระบบ การทำ Request Migration เป็นกระบวนการเปลี่ยนการร้องขอที่มีเข้ามาไปยังเซิร์ฟเวอร์ที่มีขนาดของภาระงานน้อยกว่า และการทำ Dereplication เป็นกระบวนการลดจำนวนสื่อที่ซ้ำในระบบเพื่อให้ระบบมีเนื้อที่ในการเก็บสื่อที่ได้รับความนิยม ในงานวิจัยได้กำหนดโครงสร้างของภาระงานเป็น Load Factor ซึ่ง  $LF(R_i, S_i) = \max(\frac{DB_i}{DB_j}, \frac{M_i}{M_j}, \frac{CPU_i}{CPU_j}, \frac{X_i}{X_j})$

เป็นอัตราส่วนระหว่างทรัพยากรของระบบที่ถูกใช้ไปกับที่เหลืออยู่ ใช้บ่งบอกระดับของภาระงานและสถานะคอขวดที่เกิดขึ้นกับเซิร์ฟเวอร์แต่ละตัว สำหรับกระบวนการจัด Request นั้นตัว DC จะพิจารณาจากเซิร์ฟเวอร์ที่มีวิดีโอที่ Request ต้องการเก็บอยู่โดยพิจารณาจาก  $1/LF$  นำมาคูณกับคำตอบแทนของสื่อที่เก็บอยู่บนเซิร์ฟเวอร์แต่ละตัว คว้าเซิร์ฟเวอร์ใดให้ค่าที่สูงที่สุด กระบวนการนี้เรียกว่า Pseudo-Allocation Procedure สำหรับกระบวนการ Replication นั้นจะเกิดขึ้นเมื่อระบบได้พิจารณาแล้วว่าจำนวนวิดีโอที่เก็บอยู่มีไม่เพียงพอกับความต้องการในอนาคต ระบบก็จะทำซ้ำวิดีอนั้นโดยใช้ทรัพยากรของระบบที่มีอยู่ทั้งหมด เพื่อให้กระบวนการ Replication ใช้เวลาน้อยที่สุด ในระบบได้แบ่งประเภทของวิดีโอออกเป็น 4 คลาส ที่ประกอบด้วย 1.วิดีโอที่ไม่มีอยู่ในเซิร์ฟเวอร์ 2.วิ

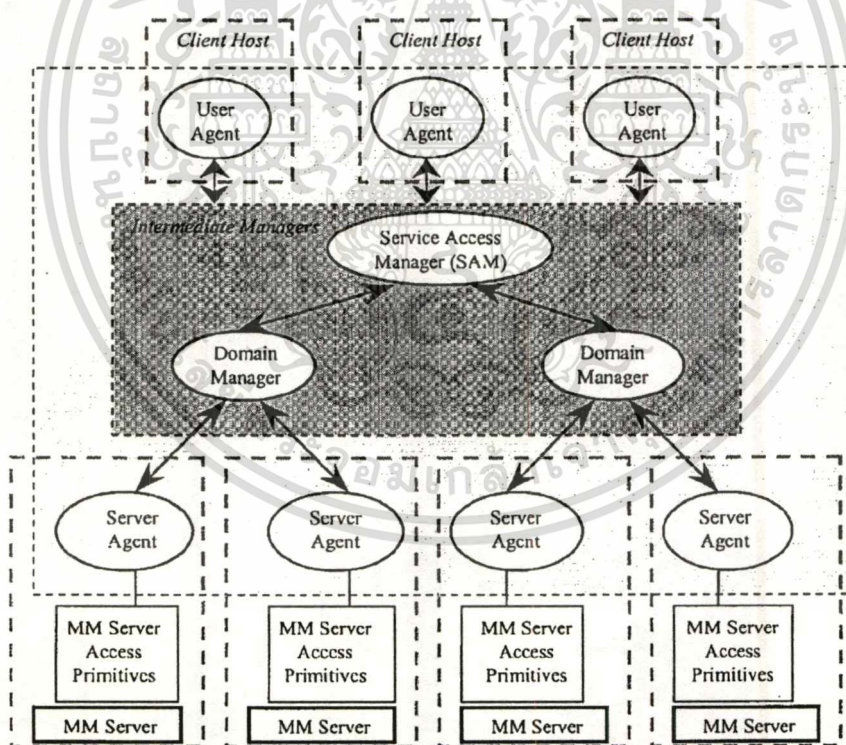
ดีโอที่กำลังถูกใช้งานอยู่ 3.วิดีโอที่ถูกมาร์กว่าลบไปแล้ว (Lazy Dereplication) และ 4.วิดีโอเก็บอยู่แต่ไม่ได้ถูกใช้งาน ในการทดลองงานวิจัยได้กำหนด Policies ขึ้น 4 Policies ที่ประกอบด้วย P1 ที่มีความสามารถในการกำหนด Request และมีการทำ Replication ตามต้องการเมื่อระบบมีจำนวนสื่อที่ให้บริการไม่พอ P2 มีกระบวนการคัดลอกการวางสื่อสำหรับอนาคตและมีการกำหนด Request แบบ Pseudo-allocation P3 มีกระบวนการคัดลอกการวางสื่อและมีการกำหนด Request ปกติ และ P4 มีการคัดลอกการวางสื่อและมีการทำ Eager Replication ที่มีการทำ Optimization สำหรับระบบในอนาคต จากการทดสอบพบว่า Policy ที่ให้ผลดีต่อระบบที่สุดคือ P3 และ P4



รูปที่ 2.12 โครงสร้างสำหรับวิดีโอเซิร์ฟเวอร์ที่มีกระบวนการ Replication [44]

ใน [45] ได้กล่าวถึงระบบสำหรับให้บริการข้อมูลสื่อผสมตามต้องการ (Multimedia-on-Demand) ที่มีชื่อว่า Scalable Access Platform (SAP) ตามโครงสร้างที่แสดงอยู่ในรูปที่ 2.13 ระบบเน้นที่ความสามารถในสร้าง ปรับเปลี่ยน และจัดการบริการและแอปพลิเคชันทางด้านสื่อผสม โดยการเพิ่มเซิร์ฟเวอร์ใหม่เข้าไปในระบบนั้นก็ทำตามข้อกำหนดระหว่างโพรโตคอลของ SAP กับโพรโตคอลของเซิร์ฟเวอร์เท่านั้น ระบบ SAP นี้มีการแบ่งกลุ่มของเซิร์ฟเวอร์ออกเป็นโดเมน และแต่ละกลุ่มก็จะได้รับการจัดโครงสร้างเป็นลำดับชั้น สำหรับตำแหน่งการเก็บมีลติมีเดียออบเจกต์นั้นจะเป็นแบบสเตติก คือมีการกำหนดไว้ก่อนหน้าแล้ว ที่ไคลเอ็นต์จะมี User Agent ที่ทำหน้าที่ติดต่อกับผู้ใช้และสื่อสารกับส่วนบนสุดของลำดับชั้นของระบบ SAP คือ Service Access Manager (SAM) ตัว SAM นี้จะเป็นศูนย์กลางในการกระจายภาระงานไปให้กับโดเมนต่างๆ ตามตำแหน่งของมีลติมีเดียออบเจกต์ และระดับของภาระงาน ในแต่ละโดเมนก็จะมี Domain Manager (DM) ที่ทำหน้าที่เป็นศูนย์กลางของโดเมนในการกระจายการร้องขอที่ถูกส่งมาจาก SAM ถ้าเซิร์ฟเวอร์ไม่สามารถตอบสนองต่อการร้องขอนั้นได้ ตัว DM ก็จะจัดหาเซิร์ฟเวอร์เครื่องอื่นแทนไปเรื่อยๆ โดยที่ตัวเซิร์ฟเวอร์นั้นจะมี Server Agent ฝังตัวอยู่เพื่อทำหน้าที่เป็นตัวกลางระหว่าง DM กับเซิร์ฟเวอร์ให้บริการสื่อผสมซึ่งเป็นเซิร์ฟเวอร์ที่มีความสามารถในการให้บริการข้อมูลสื่อผสมใดๆ ในการการคำนวณภาระงานของเซิร์ฟเวอร์นั้นอีกทั้งยังมีให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กระจายภาระงานนั้นจะเป็นหน้าที่ของ DM ในโดเมน เซิร์ฟเวอร์จะแบ่งออกเป็น 3 แบบคือ Blocking คือไม่สามารถให้บริการข้อมูลสื่อผสมได้เพราะไม่เหลือทรัพยากร SemiBlocking ที่สามารถให้บริการได้บางระดับ และ NonBlocking ที่สามารถให้บริการได้ทุกระดับ เมื่อเซิร์ฟเวอร์มีการเปลี่ยนระดับก็จะมี การส่งข้อมูลไปบอก DM และเช่นเดียวกันตัว DM ก็จะแบ่งระดับของภาระงานออกเป็นระดับ เมื่อมีการเปลี่ยนระดับตัว DM ก็จะส่งข้อมูลไปบอก SAM เพื่อเป็นข้อมูลในการกระจายภาระงานต่อไป โมเดลภาระงานของเซิร์ฟเวอร์ที่ใช้ นั้นคือ  $LoadLevel(LL) = \sum \{w_i \times b_i\}$  โดยที่  $b_i$  มีค่าเป็น 0 ถ้าไม่มีมัลติมีเดียออบเจกต์และเป็น 1 ถ้ามี  $w_i$  เป็นค่าถ่วงน้ำหนักของเซิร์ฟเวอร์ซึ่งได้จากการเก็บสถิติคุณสมบัติต่างๆ ของเซิร์ฟเวอร์เช่นระดับที่สามารถให้บริการได้ อัตราการล้มเหลวในการทำงานเป็นต้น ในการทดลองผู้วิจัยได้สร้างระบบให้บริการสื่อผสมตามต้องการขึ้นมา จากผลการทดลองพบว่าถึงแม้ว่าโมเดลที่พัฒนาขึ้นจะทำให้ระบบมีเวลาในการตอบสนองต่อการร้องขอของผู้ใช้มากขึ้นเล็กน้อย แต่เมื่อเทียบกับระบบให้บริการข้อมูลสื่อผสมแล้วพบว่าระบบสามารถกระจายภาระงานได้อย่างเท่าเทียมและลดอัตราปฏิเสธลงได้มาก



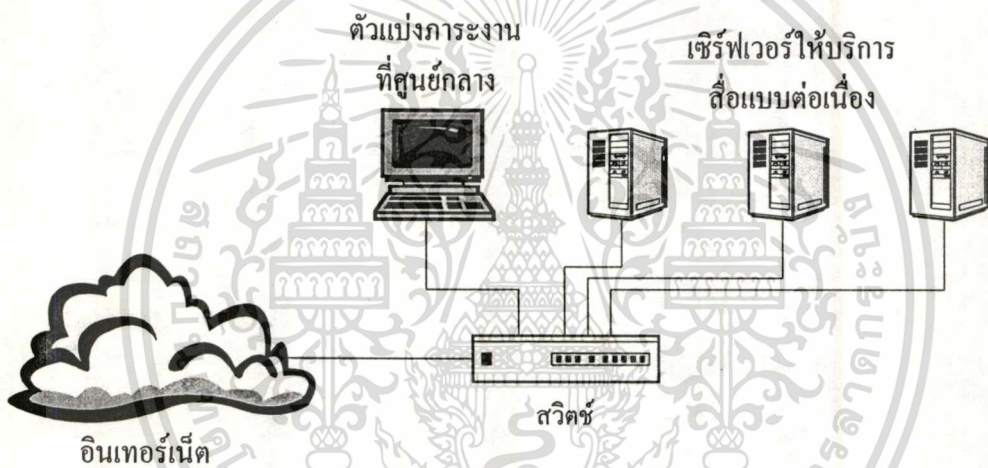
รูปที่ 2.13 โครงสร้างของระบบ Scalable Access Platform (SAP) [45]

### บทที่ 3

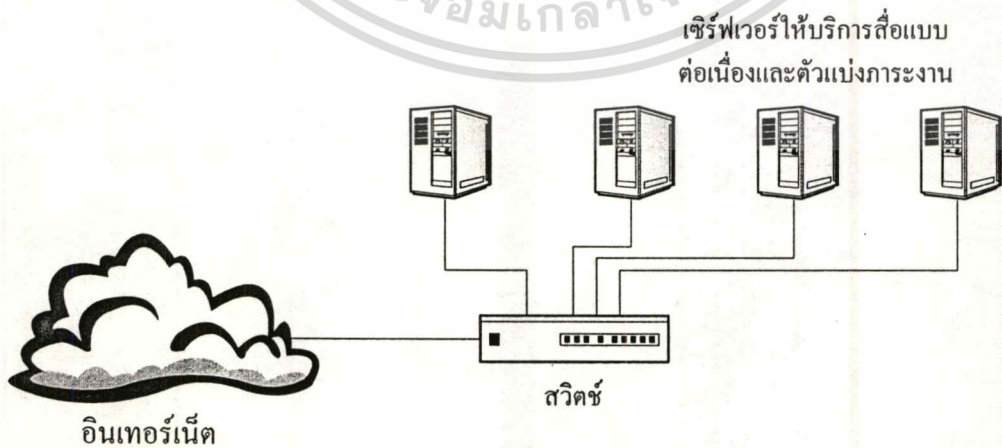
## เซิร์ฟเวอร์สำหรับให้บริการสื่อแบบต่อเนื่อง

### 3.1 สถาปัตยกรรม

ในงานวิจัยนี้ได้ประเมินสถาปัตยกรรมของระบบที่ให้บริการสื่อแบบต่อเนื่อง 2 สถาปัตยกรรม คือ สถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง (Centralized Load Balancing Architecture) ดังที่แสดงอยู่ในรูปที่ 3.1 และสถาปัตยกรรมที่มีตัวแบ่งภาระงานประจำอยู่ที่เซิร์ฟเวอร์ (Distributed Load Balancing Architecture) ดังที่แสดงอยู่ในรูปที่ 3.2



รูปที่ 3.1 สถาปัตยกรรมของระบบให้บริการสื่อแบบต่อเนื่องแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง



รูปที่ 3.2 สถาปัตยกรรมของระบบให้บริการสื่อแบบต่อเนื่องแบบที่มีตัวแบ่งภาระงานประจำอยู่ที่เซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

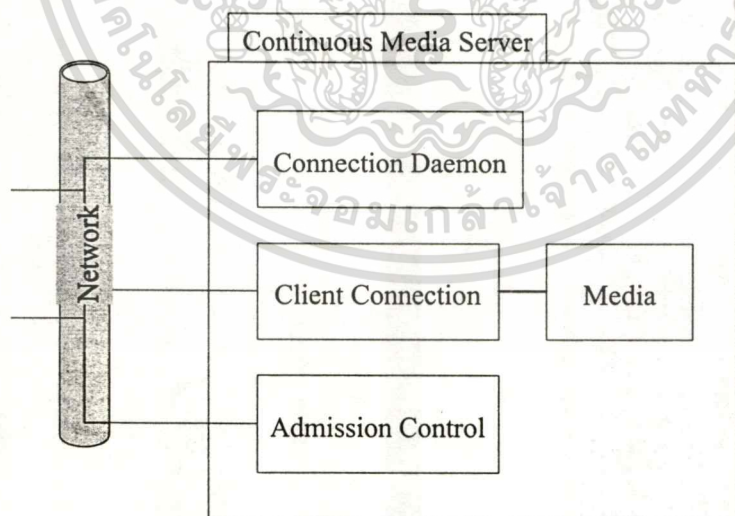
สำหรับการทำงานของระบบที่มีสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง การร้องขอจากไคลเอ็นต์จะถูกส่งมาที่ตัวแบ่งภาระงานที่ศูนย์กลาง ตัวแบ่งภาระงานที่ศูนย์กลางก็จะทำหน้าที่ตัดสินใจส่งการร้องขอไปให้กับเซิร์ฟเวอร์ที่มีอยู่ในระบบ ส่วนสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์นั้นเซิร์ฟเวอร์แต่ละเครื่องจะทำหน้าที่ตัดสินใจรับ ปฏิเสธ หรือถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์อื่นในระบบได้ด้วยตัวเอง

## 3.2 โครงสร้างและการทำงาน

### 3.2.1 ส่วนประกอบของสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

สำหรับโครงสร้างการทำงานของเซิร์ฟเวอร์ที่ให้บริการสื่อแบบต่อเนื่อง (Continuous Media Server) สถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลางแสดงอยู่ในรูปที่ 3.3 โดยมีส่วนประกอบดังนี้คือ

- ส่วนที่ทำหน้าที่ร้องขอเชื่อมต่อจากไคลเอ็นต์ (Connection Daemon)
- ส่วนที่ทำหน้าที่ดูแลการเชื่อมต่อกับไคลเอ็นต์ (Client Connection)
- ส่วนที่ทำหน้าที่อ่านข้อมูลจากดิสก์ (Media)
- ส่วนที่ทำหน้าที่สำหรับกระบวนการอนุญาตเพื่อเข้าใช้ (Admission Control)



รูปที่ 3.3 โครงสร้างของเซิร์ฟเวอร์สำหรับให้บริการสื่อแบบต่อเนื่องในสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

การทำงานของเซิร์ฟเวอร์ให้บริการสื่อแบบต่อเนื่องนั้น เมื่อเซิร์ฟเวอร์เริ่มทำงานส่วน Connection Daemon ก็จะคอยการเชื่อมต่อจากไคลเอนต์ เมื่อไคลเอนต์เชื่อมต่อผ่านโพรโตคอล TCP มาทางพอร์ต 80 ส่วน Connection Daemon ก็จะทำการสร้าง Client Connection ขึ้นมาเพื่อให้ทำหน้าที่รับผิดชอบดูแลการเชื่อมต่อระหว่างไคลเอนต์นั้นๆ ตลอดจนกว่าจะจบการให้บริการ

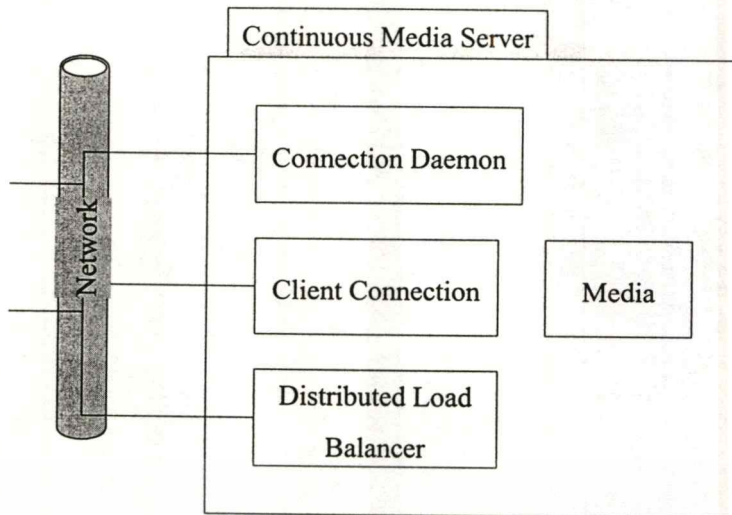
เมื่อไคลเอนต์ส่งการร้องขอเข้ามาที่ Client Daemon แล้ว ส่วน Client Daemon ก็จะทำการตรวจสอบจากส่วน Admission Control ว่าเซิร์ฟเวอร์มีทรัพยากรเหลือเพียงพอที่จะให้บริการหรือไม่ ถ้าเซิร์ฟเวอร์มีทรัพยากรเหลือไม่พอก็จะทำการปฏิเสธการร้องขอ ส่วนในกรณีที่ Admission Control ตอบกลับว่ามีทรัพยากรเพียงพอและยอมให้บริการ ส่วน Client Connection ที่สัมพันธ์กับการร้องขอนั้นก็จะทำเรียกใช้บริการจาก Media เพื่อทำการเปิดไฟล์เพื่ออ่านข้อมูล ซึ่งส่วน Media นั้นนอกจากจะทำหน้าที่ติดต่อกับไฟล์แล้วยังทำหน้าที่เป็นแคชในตัวเพื่อเพิ่มประสิทธิภาพในการอ่านสื่อแบบต่อเนื่องจากดิสก์อีกด้วย

ในการทำงานของส่วน Client Connection นั้น จะมีการกำหนดคาบเวลาในการอ่านข้อมูลจาก Media ขึ้นมาแล้วทำการส่งต่อไปให้กับไคลเอนต์ที่รับผิดชอบอยู่ ถ้าการร้องขอต้องการสื่อที่มีอัตราเล่นสูง ส่วน ClientConnection ก็จะมีการอ่านข้อมูลจาก Media ในปริมาณที่มากกว่าสื่อที่ต้องการอัตราการเล่นต่ำกว่า

### 3.2.2 ส่วนประกอบของสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์

สำหรับโครงสร้างการทำงานของเซิร์ฟเวอร์ที่ให้บริการสื่อแบบต่อเนื่อง (Continuous Media Server) ในสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์แสดงอยู่ในรูปที่ 3.4 โดยมีส่วนประกอบดังนี้คือ

- ส่วนที่ทำหน้าที่รอฟังการขอเชื่อมต่อจากไคลเอนต์ (Connection Daemon)
- ส่วนที่ทำหน้าที่ดูแลการเชื่อมต่อกับไคลเอนต์ (Client Connection)
- ส่วนที่ทำหน้าที่อ่านข้อมูลจากดิสก์ (Media)
- ส่วนที่ทำหน้าที่สำหรับกระบวนการอนุญาตเพื่อเข้าใช้และแบ่งภาระงาน (Distributed Load Balancer)



รูปที่ 3.4 โครงสร้างของเซิร์ฟเวอร์สำหรับให้บริการสื่อแบบต่อเนื่องในสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์

การทำงานของเซิร์ฟเวอร์ให้บริการสื่อแบบต่อเนื่องในสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์นั้น จะคล้ายกับการทำงานของสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลางแต่จะแตกต่างกันในกระบวนการของการตรวจสอบทรัพยากรของเซิร์ฟเวอร์ซึ่งในสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์นั้นจะทำโดย Distributed Load Balancer นอกจากนั้นส่วน Distributed Load Balancer ยังทำหน้าที่ตัดสินใจว่าจะทำการการถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์เครื่องอื่นหรือจะปฏิเสธในกรณีที่เซิร์ฟเวอร์มีทรัพยากรเหลือไม่เพียงพออีกด้วย และถ้าเซิร์ฟเวอร์ใช้อัลกอริทึมที่จำเป็นต้องมีการแลกเปลี่ยนข้อมูลภาระงานและการให้บริการ ส่วน Distributed Load Balancer ก็จะทำหน้าที่นี้ด้วย

### 3.2.3 กระบวนการอนุญาตเพื่อเข้าใช้ (Admission Control) ที่ใช้ในเซิร์ฟเวอร์สำหรับให้บริการสื่อแบบต่อเนื่อง

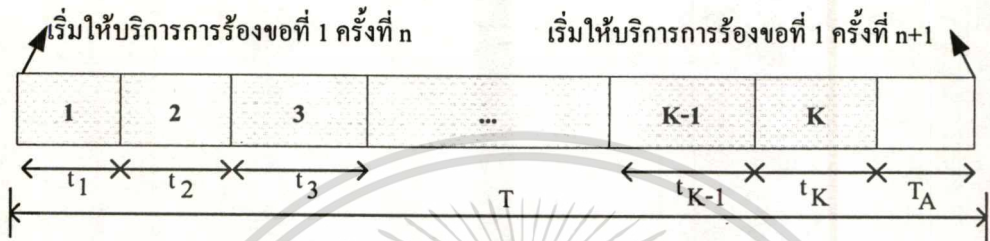
สำหรับกระบวนการอนุญาตเพื่อเข้าใช้ เป็นกระบวนการสำหรับการตรวจสอบว่าเซิร์ฟเวอร์มีทรัพยากรเพียงพอต่อการให้บริการกับการร้องขอจากไคลเอนต์ที่มีเข้ามาหรือไม่ ซึ่งกระบวนการนี้ทำเพื่อเป็นการประกันว่าการร้องขอที่ทุกการร้องขอที่เซิร์ฟเวอร์ได้รับบริการ โดยมีคุณภาพในระดับที่การร้องขอต้องการตลอดช่วงเวลาของการบริการ

กระบวนการอนุญาตเพื่อเข้าใช้งานนี้จะทำการตรวจสอบทรัพยากรของเซิร์ฟเวอร์ที่มีเหลืออยู่ ต่อการร้องขอที่มีเข้ามาใหม่ ซึ่งถ้าเพียงพอก็จะยอมให้บริการ แต่ถ้าไม่พอเซิร์ฟเวอร์ก็เลือกที่จะปฏิเสธการให้บริการ ทั้งนี้ถ้าเซิร์ฟเวอร์ยอมให้บริการไป ก็จะส่งผลให้ทรัพยากรของระบบไม่เพียงพอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พอต่อความต้องการของการร้องขอที่กำลังให้บริการอยู่ ซึ่งจะส่งผลให้คุณภาพของการให้บริการการร้องขอทั้งหลายลดลง

เนื่องจากการทำงานของเซิร์ฟเวอร์ในการให้บริการสื่อบันทึกต่อเนื่องนั้นมีลักษณะเป็นการวนรอบ เซิร์ฟเวอร์จะให้บริการการร้องขอหลายๆ การร้องขอในช่วงเวลาเดียวกัน โดยจะเวียนให้บริการทีละหนึ่งการร้องขอ โดยมีคาบเวลาของการวนรอบที่แน่นอน สมมติเป็น  $T$  วินาทีดังแสดงในรูปที่ 3.5



รูปที่ 3.5 เวลาในการเข้าใช้บริการเซิร์ฟเวอร์ของการร้องขอที่ 1 ถึง K (คาบเวลาวนรอบของเซิร์ฟเวอร์คือ  $T$  วินาที)

กำหนดให้  $t_i$  คือเวลาที่การร้องขอที่  $i$  ใช้บริการใน 1 การวนรอบ

$$t_i = \frac{X_i T}{Y} + \frac{X_i T}{Z} \quad (1)$$

$X_i$  คืออัตราการเล่น (Playback Rate) ของสื่อบันทึกต่อเนื่องของการร้องขอที่  $i$

$Y$  คือแบนด์วิดท์ของดิสก์ในเซิร์ฟเวอร์

$Z$  คือแบนด์วิดท์ของเน็ตเวิร์กที่เซิร์ฟเวอร์เชื่อมต่ออยู่

$T$  คือ คาบเวลาของการวนรอบของเซิร์ฟเวอร์

และช่วงเวลาที่ยังว่างอยู่ ( $T_A$ ) คำนวณจาก

$$T_A = T - \sum_{i=1}^K t_i \quad (2)$$

โดย  $K$  คือจำนวนการร้องขอที่เซิร์ฟเวอร์ให้บริการอยู่ในขณะนั้น

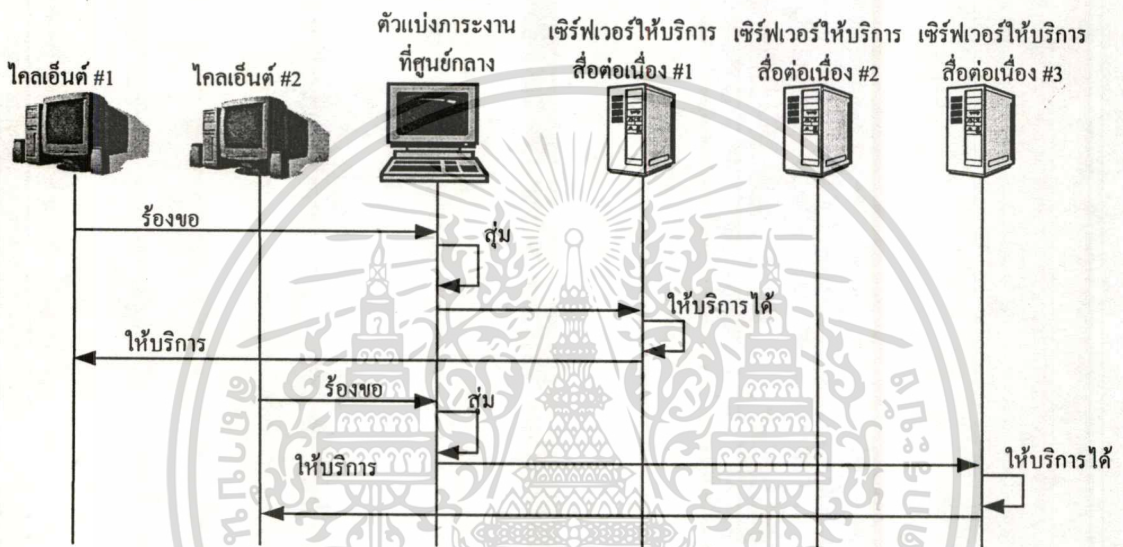
ทุกครั้งที่มีการร้องขอใหม่เข้ามาที่เซิร์ฟเวอร์ ตัวเซิร์ฟเวอร์จะประเมินเวลาที่ใช้การร้องขอนั้นๆ ต้องการ และเวลาที่ยังว่างอยู่ ตามสมการที่ (1) และ (2) ตามลำดับ หากพบว่าเวลาจาก (1) มากกว่าเวลาจาก (2) เซิร์ฟเวอร์ก็จะปฏิเสธการร้องขอนั้น

### 3.3 อัลกอริทึมการแบ่งภาระงาน

#### 3.3.1 อัลกอริทึมการแบ่งภาระงานของสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

Random

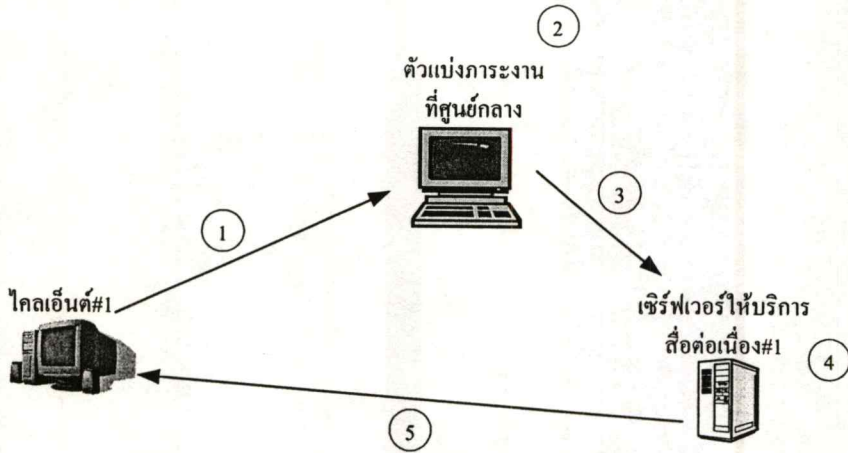
เป็นอัลกอริทึมสำหรับการแบ่งภาระงานแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลางทำหน้าที่รับการร้องขอที่มีมาจากไคลเอ็นต์ จากนั้นก็จะเลือกเซิร์ฟเวอร์ในระบบที่จะให้บริการโดยการสุ่ม โดยแสดงอยู่ในรูปที่ 3.6



รูปที่ 3.6 การทำงานของอัลกอริทึม Random สำหรับสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

เวลาเริ่มต้นในการให้บริการ

เวลาทั้งหมดสามารถแบ่งได้ดังแสดงอยู่ในรูปที่ 3.7



รูปที่ 3.7 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Random ของสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

1. เวลาที่การร้องขอจากไคลเอนต์เดินทางมายังตัวแบ่งภาระงานที่ศูนย์กลาง เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่ตัวแบ่งภาระงานที่ศูนย์กลางใช้ในการตัดสินใจเลือกเซิร์ฟเวอร์สำหรับให้บริการต่อการร้องขอจากไคลเอนต์ โดยการสุ่มเลือกเซิร์ฟเวอร์
3. เวลาที่การร้องขอจากตัวแบ่งภาระงานที่ศูนย์กลางเดินทางไปยังเซิร์ฟเวอร์ให้บริการสื่อต่อเนื่องที่ได้รับเลือกเพื่อให้บริการกับไคลเอนต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
4. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องให้บริการต่อไคลเอนต์
5. เวลาที่ข้อมูลชิ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องไปยังไคลเอนต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + (2) + (3) + (4) + (5)$$

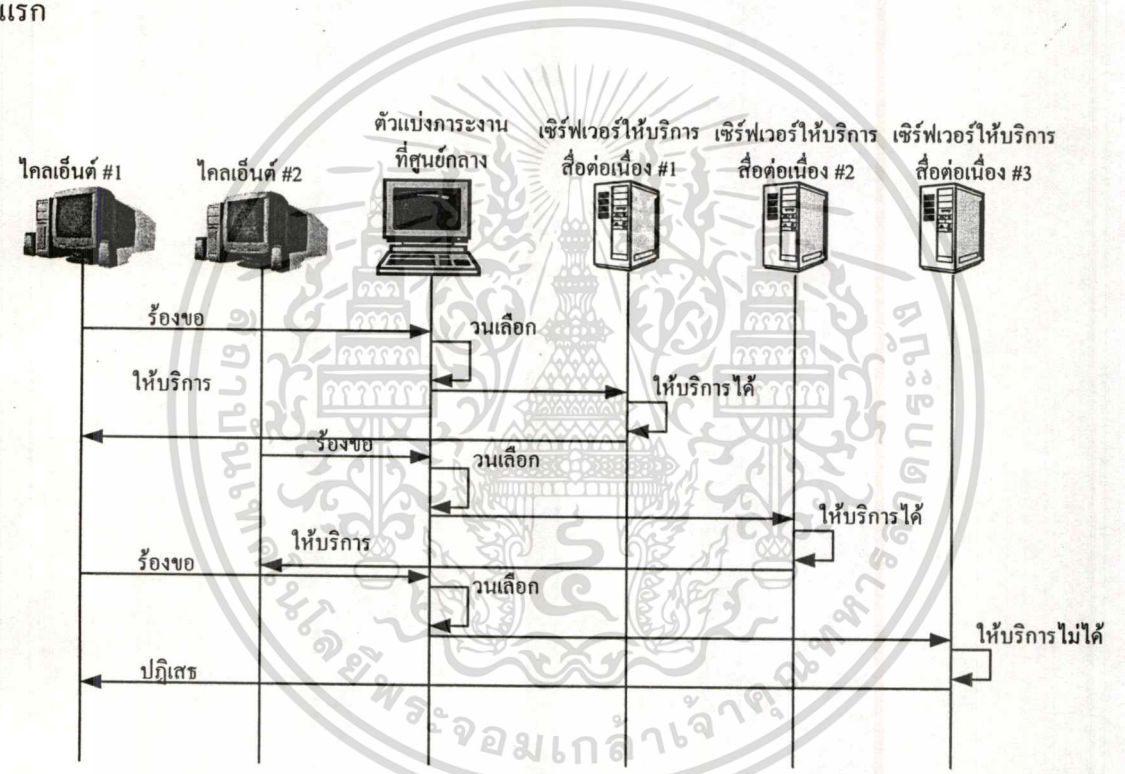
- โดยที่เวลา (1) (3) และ (5) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอด้วย ซึ่งจะขึ้นอยู่กับความสามารถของตัวแบ่งภาระงานที่ศูนย์กลางเป็นหลัก (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.4)
- เวลา (4) จะขึ้นอยู่กับความสามารถของคิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Round-Robin

เป็นอัลกอริทึมสำหรับการแบ่งภาระงานแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลางทำหน้าที่รับ การร้องขอที่มีมาจากไคลเอ็นต์ จากนั้นก็จะเลือกเซิร์ฟเวอร์ในระบบที่จะให้บริการโดยการวนเลือก โดยการภาพรวมของการทำงานแสดงอยู่ในรูปที่ 3.8

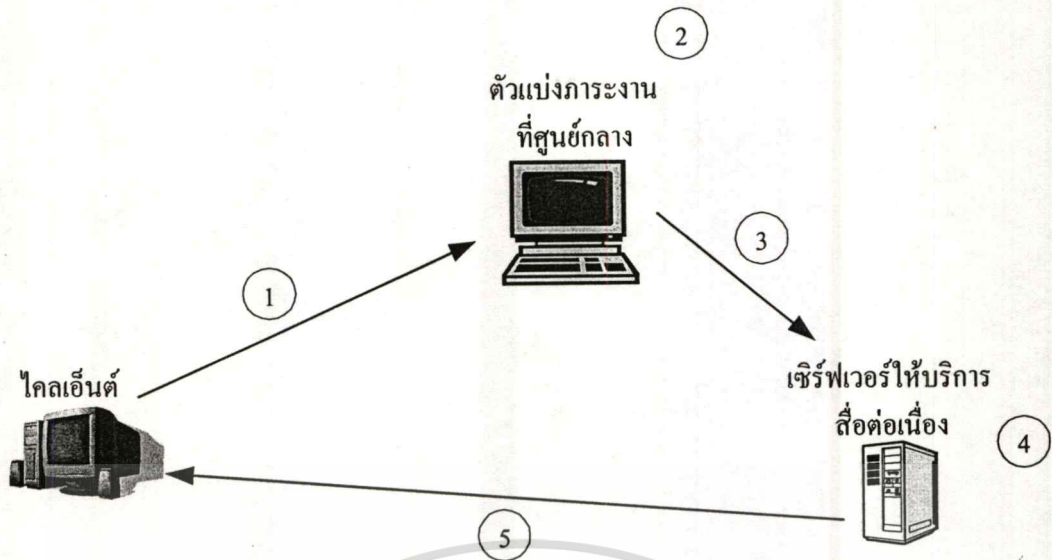
สำหรับอัลกอริทึมในการวนเลือกนั้น ที่ตัวแบ่งภาระงานที่ศูนย์กลางจะเก็บมีตัวแปรไว้ สำหรับเก็บว่าในการร้องขอก่อนหน้านี้เซิร์ฟเวอร์ใดเป็นผู้ให้บริการ เมื่อมีการร้องขอเข้ามา ตัวแบ่ง ภาระงานที่ศูนย์กลางก็จะเพิ่มค่าในตัวแปรเพื่อเลือกเซิร์ฟเวอร์ตัวต่อไปสำหรับให้บริการ และเมื่อ เซิร์ฟเวอร์ทุกตัวได้ให้บริการครบแล้ว ตัวแบ่งภาระงานที่ศูนย์กลางก็จะกลับมาเลือกเซิร์ฟเวอร์ตัว แรก



รูปที่ 3.8 การทำงานของอัลกอริทึม Round-Robin สำหรับสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

เวลาเริ่มต้นในการให้บริการ

คล้ายกับอัลกอริทึม Random โดยแสดงอยู่ในรูปที่ 3.9



รูปที่ 3.9 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Round-Robin ของสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

1. เวลาที่การร้องขอจากไคลเอ็นต์เดินทางมายังตัวแบ่งภาระงานที่ศูนย์กลาง เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่ตัวแบ่งภาระงานที่ศูนย์กลางใช้ในการตัดสินใจเลือกเซิร์ฟเวอร์สำหรับให้บริการต่อการร้องขอจากไคลเอ็นต์โดยการวนเลือกเซิร์ฟเวอร์
3. เวลาที่การร้องขอจากตัวแบ่งภาระงานที่ศูนย์กลางเดินทางไปยังเซิร์ฟเวอร์ให้บริการสื่อต่อเนื่องที่ได้รับเลือกเพื่อให้บริการกับไคลเอ็นต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
4. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องเริ่มต้นการให้บริการต่อไคลเอ็นต์
5. เวลาที่ข้อมูลชิ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องไปยังไคลเอ็นต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + (2) + (3) + (4) + (5)$$

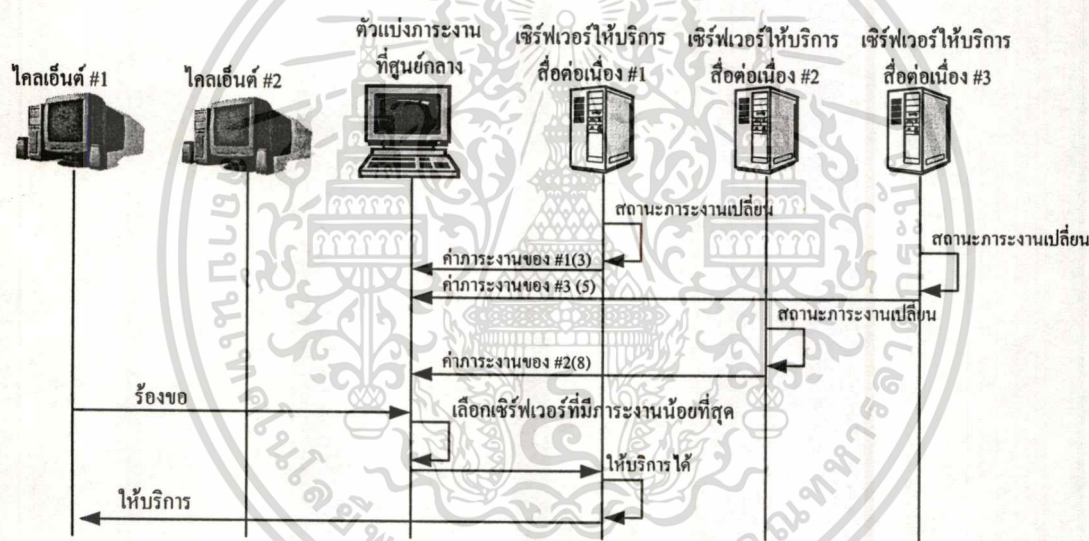
- โดยที่เวลา (1) (3) และ (5) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอด้วย ซึ่งจะขึ้นอยู่กับความสามารถของตัวแบ่งภาระงานที่ศูนย์กลางเป็นหลัก (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เวลา (4) จะขึ้นอยู่กับความสามารถของดิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1)

State Change Broadcast

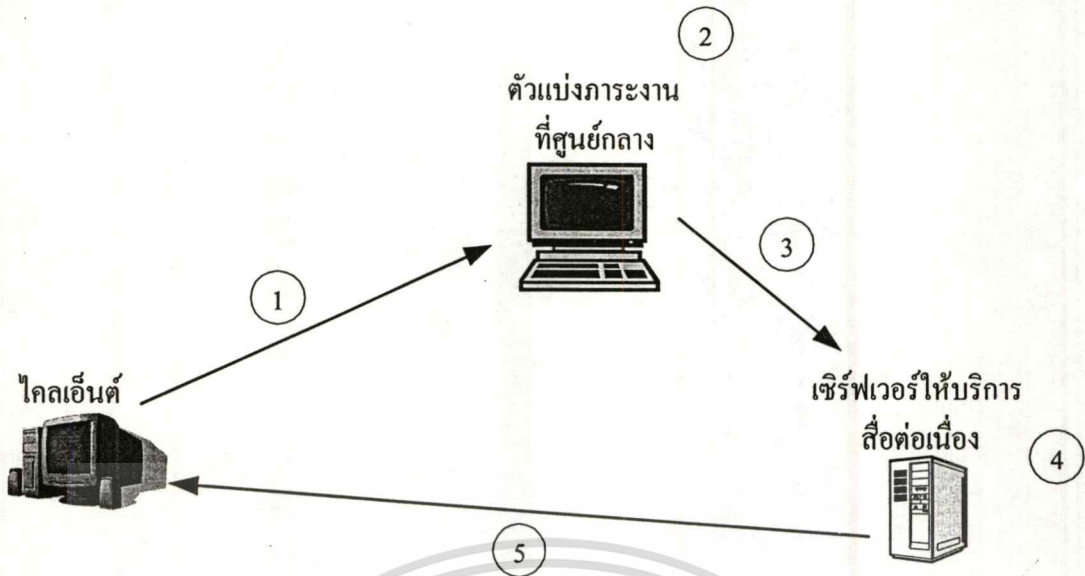
เป็นอัลกอริทึมสำหรับการแบ่งภาระงานแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลางทำหน้าที่รับการร้องขอที่มีมาจากไคลเอนต์ ซึ่งนอกจากตัวแบ่งภาระงานที่ศูนย์กลางจะทำหน้าที่รับการร้องขอจากไคลเอนต์แล้วยังทำหน้าที่คอยรับสถานะของภาระงานจากเซิร์ฟเวอร์ทั้งหมดในระบบอีกด้วย สำหรับเซิร์ฟเวอร์ให้บริการสื่อต่อเนื่องในระบบจะส่งค่าภาระงานมาให้กับตัวแบ่งภาระงานที่ศูนย์กลางก็ต่อเมื่อสถานะภาระงานของตนมีการเปลี่ยนแปลง ซึ่งตัวแบ่งภาระงานก็จะใช้ข้อมูลสถานะของภาระงานที่ได้รับมานี้ในการตัดสินใจเลือกเซิร์ฟเวอร์ที่จะให้บริการต่อการร้องขอจากไคลเอนต์ โดยแสดงอยู่ในรูปที่ 3.10



รูปที่ 3.10 การทำงานของอัลกอริทึม State Change Broadcast สำหรับสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

เวลาเริ่มต้นในการให้บริการ

คล้ายกับอัลกอริทึม Random โดยแสดงอยู่ในรูปที่ 3.11



รูปที่ 3.11 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม State Change Broadcast ของสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

1. เวลาที่การร้องขอจากไคลเอ็นต์เดินทางมายังตัวแบ่งภาระงานที่ศูนย์กลาง เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่ตัวแบ่งภาระงานที่ศูนย์กลางใช้ในการตัดสินใจเลือกเซิร์ฟเวอร์สำหรับให้บริการต่อการร้องขอจากไคลเอ็นต์ โดยเลือกจากเซิร์ฟเวอร์ที่มีทรัพยากรเหลืออยู่มากที่สุด
3. เวลาที่การร้องขอจากตัวแบ่งภาระงานที่ศูนย์กลางเดินทางไปยังเซิร์ฟเวอร์ให้บริการสื่อต่อเนื่องที่ได้รับเลือกเพื่อให้บริการกับไคลเอ็นต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
4. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องเริ่มต้นการให้บริการต่อไคลเอ็นต์
5. เวลาที่ข้อมูลชิ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องไปยังไคลเอ็นต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + (2) + (3) + (4) + (5)$$

- โดยที่เวลา (1) (3) และ (5) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอด้วย ซึ่งจะขึ้นอยู่กับความสามารถของตัวแบ่งภาระงานที่ศูนย์กลางเป็นหลัก (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.4) แต่ทั้งนี้ตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แบ่งภาระงานที่ศูนย์กลางก็ต้องแบ่งเวลาในการทำงานไปให้กับการเก็บข้อมูลที่ได้รับบริการส่งมาจากเซิร์ฟเวอร์ต่างๆ ในระบบด้วย ซึ่งอาจจะทำให้กระบวนการในการตัดสินใจใช้เวลามากกว่าปกติเพราะต้องคอยให้ทรัพยากรว่างเสียก่อน

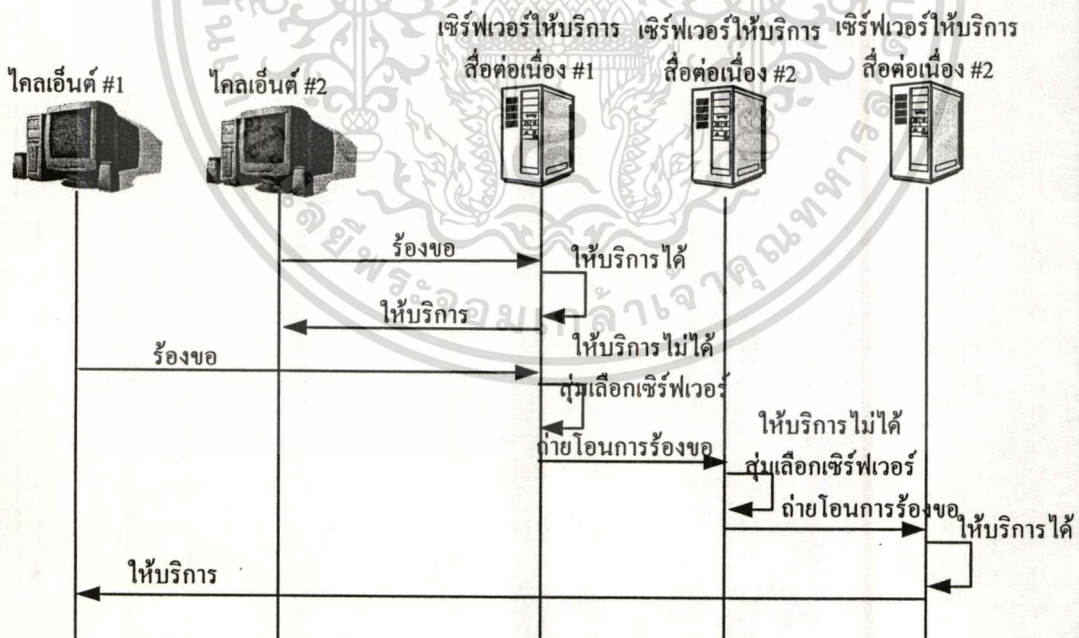
- เวลา (4) จะขึ้นอยู่กับความสามารถของคิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1)

### 3.3.2 อัลกอริทึมการแบ่งภาระงานในสถาปัตยกรรมที่มีตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์

#### Random

เนื่องจากเป็นสถาปัตยกรรมที่เซิร์ฟเวอร์ทุกตัวมีหน่วยแบ่งภาระงาน (Load Balancer) ดังนั้นเซิร์ฟเวอร์ที่ให้บริการต่อเนื่องแต่ละตัวจะสามารถตัดสินใจให้บริการต่อการร้องขอเอง หรือจะถ่ายโอนไปให้กับเซิร์ฟเวอร์อื่นหรือปฏิเสธการร้องขอถ้าเซิร์ฟเวอร์มีทรัพยากรเหลือไม่เพียงพอ โดยเงื่อนไขการตัดสินใจเป็นไปตามกระบวนการอนุญาตเพื่อเข้าใช้ที่ได้กล่าวไปแล้ว

สำหรับรายละเอียดการทำงานของอัลกอริทึม Random ในสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์แสดงอยู่ในรูปที่ 3.12



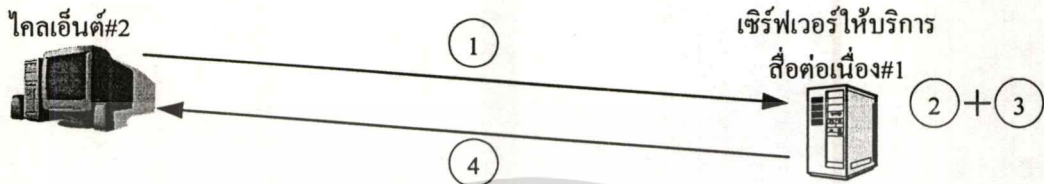
รูปที่ 3.12 การทำงานของอัลกอริทึม Random สำหรับสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายที่เซิร์ฟเวอร์ทุกเครื่อง

## เวลาเริ่มต้นให้บริการ

เราสามารถแบ่งได้ออกเป็น 2 กรณีคือ กรณีที่ไม่มีการถ่ายโอนการร้องขอ และกรณีที่มีการถ่ายโอนการร้องขอ

กรณีที่ไม่มีถ่ายโอนการร้องขอ

แสดงอยู่ในรูปที่ 3.13



รูปที่ 3.13 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Random ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่ไม่มีถ่ายโอนการร้องขอ

1. เวลาที่การร้องขอจากไคลเอนต์เดินทางมายังเซิร์ฟเวอร์ให้บริการเชื่อมต่อเนื่อง เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่เซิร์ฟเวอร์ใช้ในการตัดสินใจว่าจะให้บริการการร้องขอจากไคลเอนต์ที่มีเข้ามาเองเลยหรือไม่
3. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการเชื่อมต่อเนื่องเริ่มต้นการให้บริการต่อไคลเอนต์
4. เวลาที่ข้อมูลขึ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการเชื่อมต่อเนื่องไปยังไคลเอนต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

เวลาในการเริ่มต้นให้บริการจะเท่ากับ

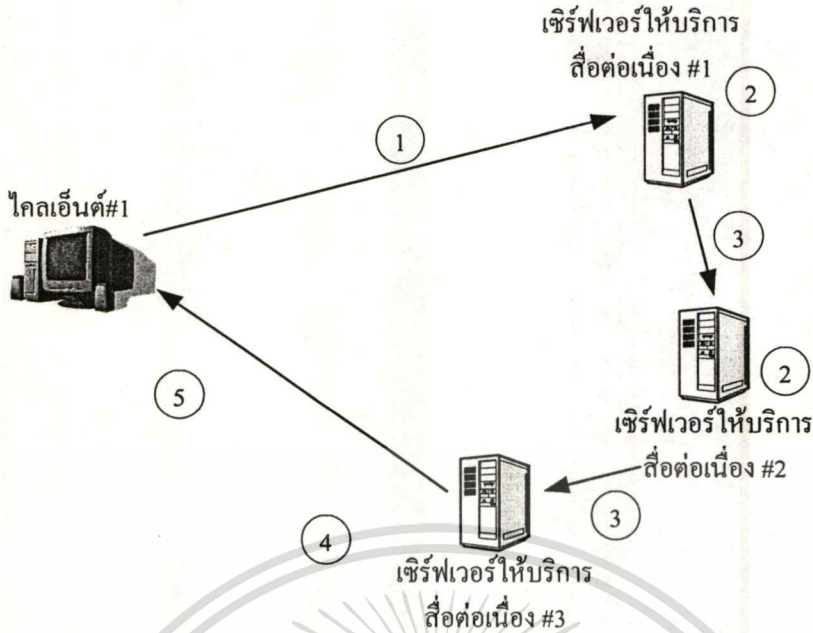
$$\text{StartupTime} = (1) + (2) + (3) + (4)$$

- โดยที่เวลา (1) และ (4) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอและเวลาที่อยู่ในคิวเพื่อคอยการประมวลผลด้วย
- เวลา (3) จะขึ้นอยู่กับความสามารถของดิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1)

## กรณีที่มีการถ่ายโอนการร้องขอ

แสดงอยู่ในรูปที่ 3.14

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.14 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Random ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่มีการถ่ายโอนการร้องขอ

1. เวลาที่การร้องขอจากไคลเอนต์เดินทางมายังเซิร์ฟเวอร์ให้บริการสื่อบทเรียน เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่เซิร์ฟเวอร์ใช้ในการตัดสินใจว่าจะให้บริการการร้องขอจากไคลเอนต์ที่มีเข้ามาเองเลยหรือไม่ ในกรณีนี้จะต้องมีการถ่ายโอนภาระงานไปให้กับเซิร์ฟเวอร์เครื่องอื่น โดยการสุ่มเลือกเซิร์ฟเวอร์ที่มีอยู่ในระบบ
3. เป็นเวลาที่ใช้ในการถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์เครื่องอื่น
4. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการสื่อบทเรียนเริ่มต้นการให้บริการต่อไคลเอนต์
5. เวลาที่ข้อมูลชิ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการสื่อบทเรียนไปยังไคลเอนต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + \sum(2) + \sum(3) + (4) + (5)$$

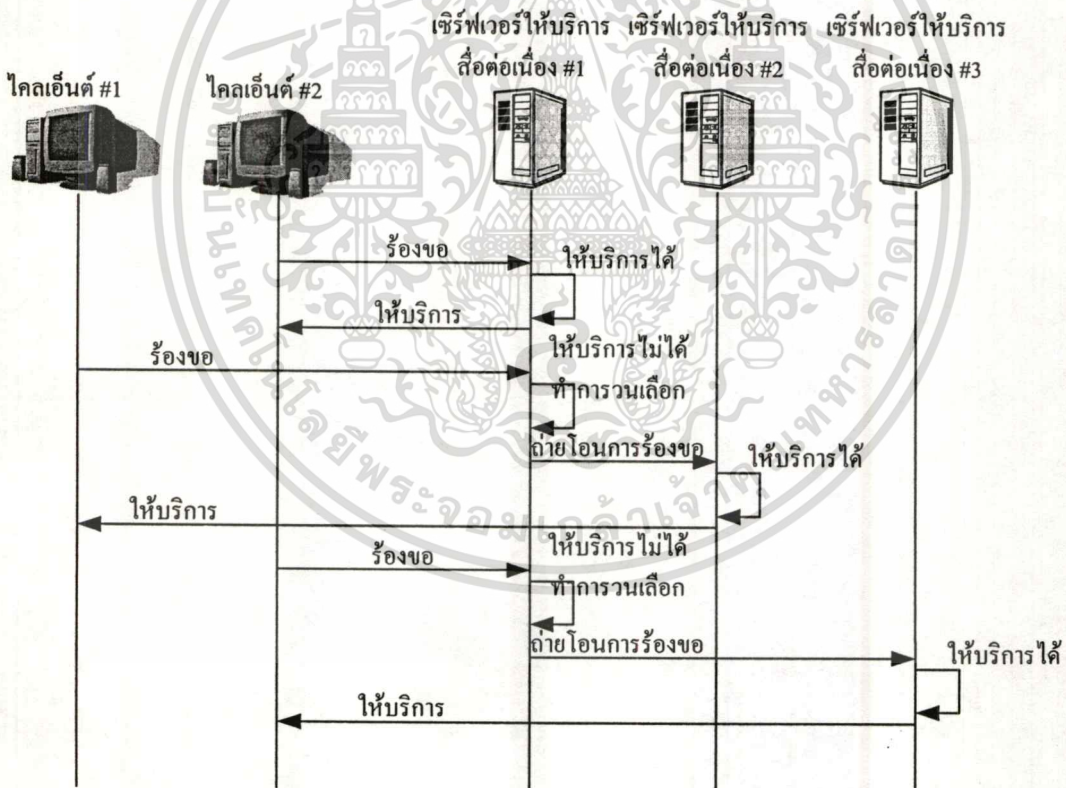
- โดยที่เวลา (1) (3) และ (5) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (3) นั้นนอกจากจะขึ้นกับระบบเครือข่ายแล้วยังขึ้นกับจำนวนครั้งของการถ่ายโอนการร้องขอด้วย ดังนั้น เวลาในส่วนที่ (3) นั้นจะเท่ากับเวลาทั้งหมดที่ใช้ในการถ่ายโอนการร้องขอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอและเวลาที่อยู่ในคิวคอยการประมวลผลด้วย
- เวลา (4) จะขึ้นอยู่กับความสามารถของดิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1) ดังนั้นเวลาเริ่มต้นให้บริการจึงเท่ากับ  $StartupTime = (1) + (n \times (2)) + (n \times (3)) + (4) + (5)$

### Round-Robin

ถ้าเซิร์ฟเวอร์สามารถให้บริการต่อการร้องขอได้ก็จะทำการให้บริการเอง แต่ถ้าไม่สามารถให้บริการได้ เนื่องจากมีทรัพยากรเหลือไม่เพียงพอก็จะทำการถ่ายโอนการร้องขอไปยังเซิร์ฟเวอร์อื่นๆ ที่อยู่ในระบบ โดยวนเลือกเซิร์ฟเวอร์ที่มีอยู่ในระบบ เซิร์ฟเวอร์แต่ละตัวในระบบจะเก็บข้อมูลไว้ว่าในครั้งก่อนได้ถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์ใด เพื่อใช้ในการตัดสินใจถ่ายโอนการร้องขอใหม่ ดังแสดงในรูปที่ 3.15



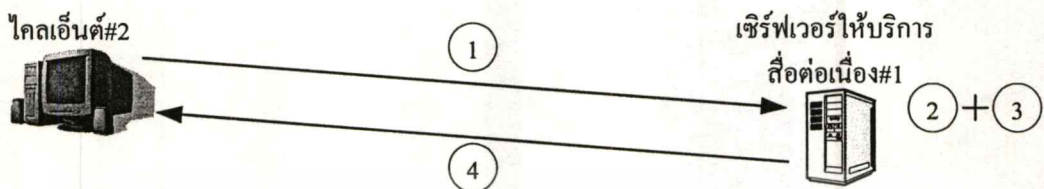
รูปที่ 3.15 การทำงานของอัลกอริทึม Round-Robin สำหรับสถาปัตยกรรมที่มีตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง

### เวลาเริ่มต้นให้บริการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เช่นเดียวกับอัลกอริทึม Random เราสามารถแบ่งได้ออกเป็น 2 กรณีคือ กรณีที่ไม่มีการถ่ายโอนการร้องขอ และกรณีที่มีการถ่ายโอนการร้องขอ

แสดงอยู่ในรูปที่ 3.16



รูปที่ 3.16 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Round-Robin ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่ไม่มีถ่ายโอนการร้องขอ

1. เวลาที่การร้องขอจากไคลเอนต์เดินทางมายังเซิร์ฟเวอร์ให้บริการต่อเนื่อง เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่เซิร์ฟเวอร์ใช้ในการตัดสินใจว่าจะให้บริการการร้องขอจากไคลเอนต์ที่มีเข้ามาเองเลยหรือไม่
3. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการต่อเนื่องเริ่มต้นการให้บริการต่อไคลเอนต์
4. เวลาที่ข้อมูลขึ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการต่อเนื่องไปยังไคลเอนต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

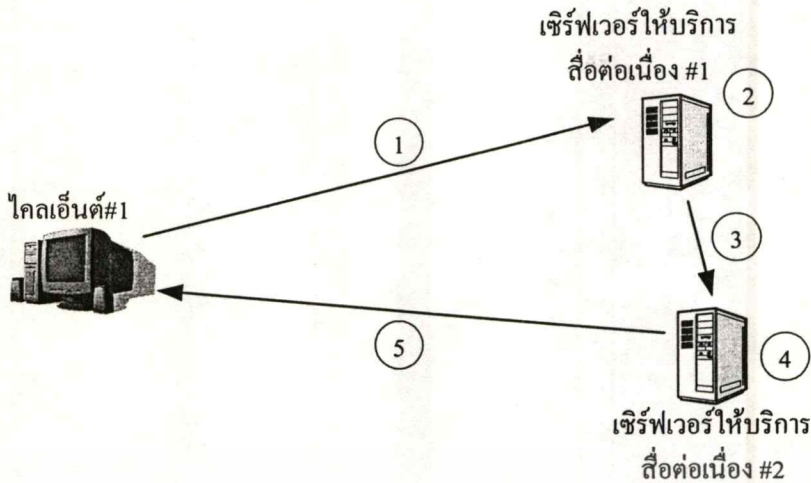
เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + (2) + (3) + (4)$$

- โดยที่เวลา (1) และ (4) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณหาค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอและเวลาที่อยู่ในคิวเพื่อคอยการประมวลผลด้วย
- เวลา (3) จะขึ้นอยู่กับความสามารถของดิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1)

กรณีที่มีการถ่ายโอนการร้องขอ

แสดงอยู่ในรูปที่ 3.17



รูปที่ 3.17 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Round-Robin ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่มีการถ่ายโอนการร้องขอ

1. เวลาที่การร้องขอจากไคลเอ็นต์เดินทางมายังเซิร์ฟเวอร์ให้บริการสัปดาห์ที่ 1 เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่เซิร์ฟเวอร์ใช้ในการตัดสินใจว่าจะให้บริการการร้องขอจากไคลเอ็นต์ที่มีเข้ามาเองเลยหรือไม่ ในกรณีนี้จะต้องมีการถ่ายโอนภาระงานไปให้กับเซิร์ฟเวอร์เครื่องอื่น โดยการวนเลือกเซิร์ฟเวอร์ที่มีอยู่ในระบบ
3. เป็นเวลาที่ใช้ในการถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์เครื่องอื่น
4. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการสัปดาห์ที่ 2 เริ่มต้นการให้บริการต่อไคลเอ็นต์
5. เวลาที่ข้อมูลขึ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการสัปดาห์ที่ 2 ไปยังไคลเอ็นต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + \sum (2) + \sum (3) + (4) + (5)$$

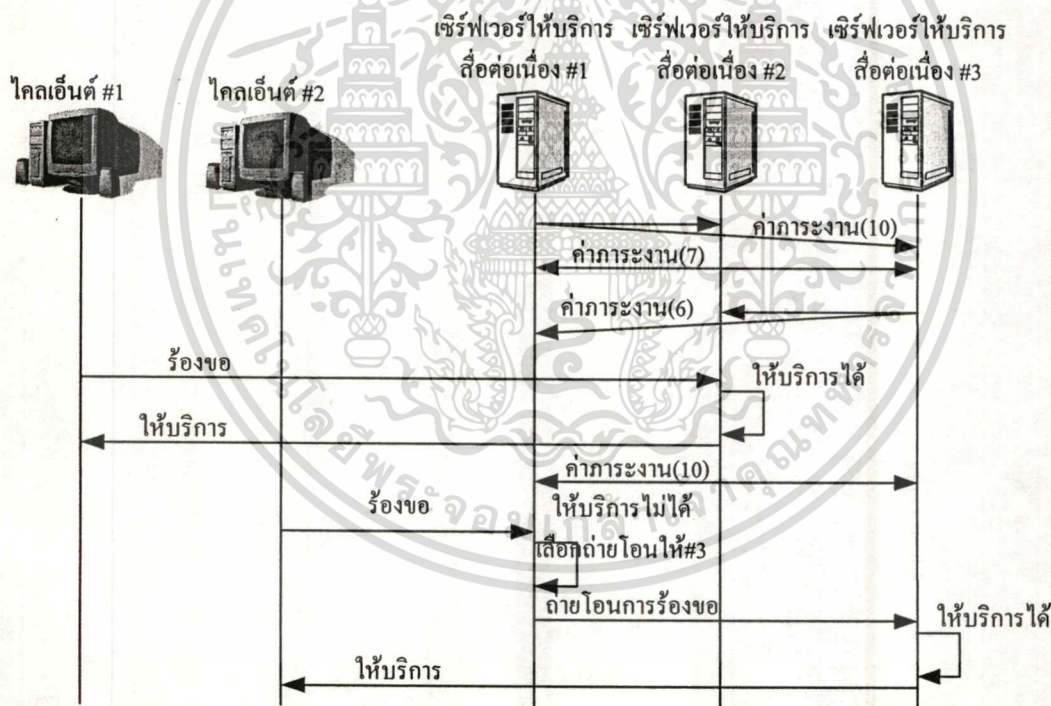
- โดยที่เวลา (1) (3) และ (5) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (3) นั้นนอกจากจะขึ้นกับระบบเครือข่ายแล้วยังขึ้นกับจำนวนครั้งของการถ่ายโอนการร้องขอด้วย เวลาในส่วนที่ (3) นั้นจะเท่ากับเวลาทั้งหมดที่ใช้ในการถ่ายโอนการร้องขอ
- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอและเวลาที่อยู่ในคิวเพื่อคอยการประมวลผลด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เวลา (4) จะขึ้นอยู่กับความสามารถของดิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1) ดังนั้นเวลาเริ่มต้นให้บริการจึงเท่ากับ  $StartupTime = (1) + (n \times (2)) + (n \times (3)) + (4) + (5)$

State Change Broadcast

เป็นอัลกอริทึมสำหรับการแบ่งภาระงานแบบที่เซิร์ฟเวอร์ทุกตัวจะมีหน่วยสำหรับแบ่งภาระงานอยู่ในตัว ดังนั้น เซิร์ฟเวอร์ที่ให้บริการต่อเนื่องจะรับการร้องขอจากไคลเอนต์โดยตรง โดยถ้าเซิร์ฟเวอร์สามารถให้บริการต่อการร้องขอได้ก็จะทำการให้บริการเอง แต่ถ้าไม่สามารถให้บริการได้ เนื่องจากมีทรัพยากรเหลือไม่เพียงพอก็จะทำการถ่ายโอนการร้องขอไปยังเซิร์ฟเวอร์อื่นๆ ที่อยู่ในระบบ โดยเลือกจากเซิร์ฟเวอร์ที่มีทรัพยากรเหลืออยู่มากที่สุด โดยเมื่อสถานะภาระงานของเซิร์ฟเวอร์ในระบบเปลี่ยนแปลง เซิร์ฟเวอร์ก็จะส่งเมสเสจออกไปบอกกับเซิร์ฟเวอร์เครื่องอื่นๆ เพื่อให้เซิร์ฟเวอร์ในระบบใช้ข้อมูลนี้ในการตัดสินใจถ่ายโอนภาระงาน ดังที่แสดงอยู่ในรูปที่ 3.18



รูปที่ 3.18 การทำงานของอัลกอริทึม State Change Broadcast สำหรับสถาปัตยกรรมที่มีตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง

เวลาเริ่มต้นให้บริการ

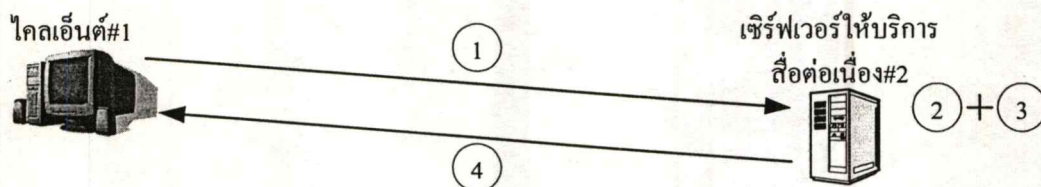
เช่นเดียวกับอัลกอริทึมที่ผ่านมาเราสามารถแบ่งได้ออกเป็น 2 กรณีคือ กรณีที่ไม่มีการถ่ายโอนการร้องขอ และกรณีที่มีการถ่ายโอนการร้องขอ

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้ในเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กรณีที่ไม่มีการถ่ายโอนการร้องขอ

แสดงอยู่ในรูปที่ 3.19



รูปที่ 3.19 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม State Change Broadcast ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่ไม่มีการถ่ายโอนการร้องขอ

1. เวลาที่การร้องขอจากไคลเอนต์เดินทางมายังเซิร์ฟเวอร์ให้บริการสื่อต่อเนื่อง เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่เซิร์ฟเวอร์ใช้ในการตัดสินใจว่าจะให้บริการการร้องขอจากไคลเอนต์ที่มีเข้ามาเองเลยหรือไม่
3. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องเริ่มต้นการให้บริการต่อไคลเอนต์
4. เวลาที่ข้อมูลชิ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องไปยังไคลเอนต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

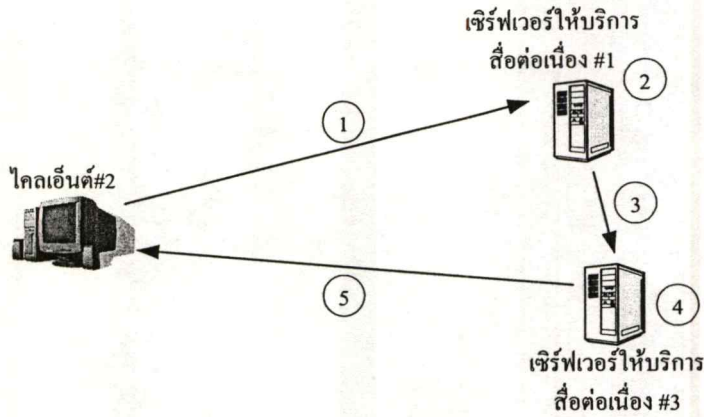
เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + (2) + (3) + (4)$$

- โดยที่เวลา (1) และ (4) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอและเวลาที่อยู่ในคิวเพื่อคอยการประมวลผลด้วย
- เวลา (3) จะขึ้นอยู่กับความสามารถของดิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1)

กรณีที่มีการถ่ายโอนการร้องขอ

แสดงอยู่ในรูปที่ 3.20



รูปที่ 3.20 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม State Change Broadcast ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่มีการถ่ายโอนการร้องขอ

1. เวลาที่การร้องขอจากไคลเอนต์เดินทางมายังเซิร์ฟเวอร์ให้บริการสื่อก่อน เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่เซิร์ฟเวอร์ใช้ในการตัดสินใจว่าจะให้บริการการร้องขอจากไคลเอนต์ที่มีเข้ามาเองเลยหรือไม่ ในกรณีนี้จะต้องมีการถ่ายโอนภาระงานไปให้กับเซิร์ฟเวอร์เครื่องอื่น โดยการตัดสินใจเลือกเซิร์ฟเวอร์ที่มีอยู่ในระบบ โดยอาศัยข้อมูลของทรัพยากรที่มีอยู่ในเซิร์ฟเวอร์อยู่แล้ว
3. เป็นเวลาที่ใช้ในการถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์เครื่องอื่น
4. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการสื่อก่อนเริ่มต้นการให้บริการต่อไคลเอนต์
5. เวลาที่ข้อมูลชิ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการสื่อก่อนไปยังไคลเอนต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + \sum (2) + \sum (3) + (4) + (5)$$

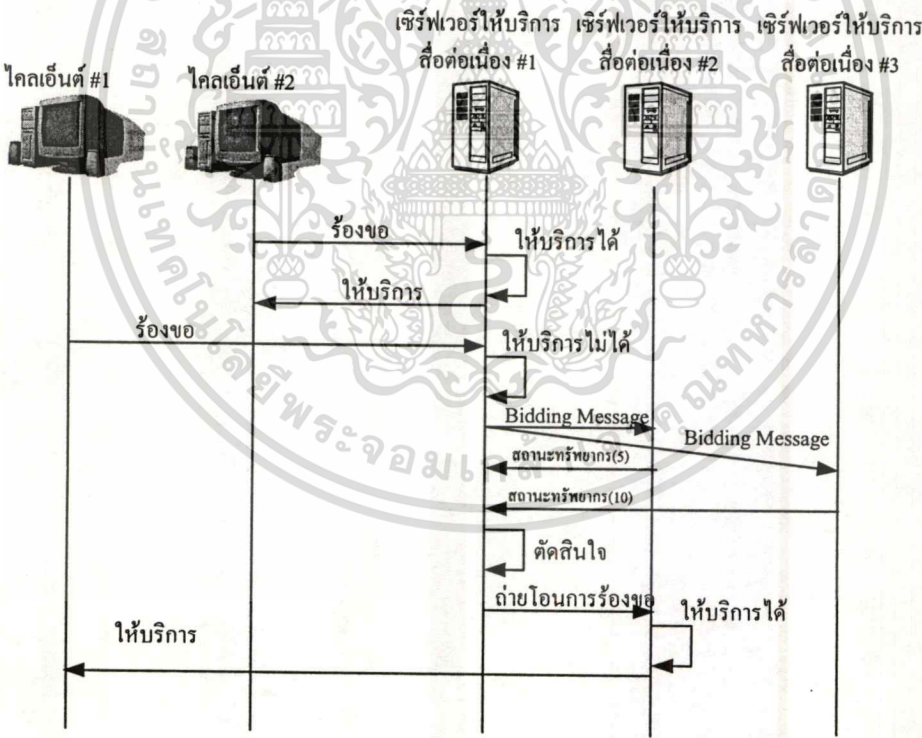
- โดยที่เวลา (1) (3) และ (5) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (3) นั้นนอกจากจะขึ้นกับระบบเครือข่ายแล้วยังขึ้นกับจำนวนครั้งของการถ่ายโอนการร้องขอด้วย ดังนั้น เวลาในส่วนที่ (3) นั้นจะเท่ากับเวลาทั้งหมดที่ใช้ในการถ่ายโอนการร้องขอ
- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอและเวลาที่อยู่ในคิวเพื่อคอยการประมวลผลด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เวลา (4) จะขึ้นอยู่กับความสามารถของคิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1) ดังนั้นเวลาเริ่มต้นให้บริการจึงเท่ากับ  $StartupTime = (1) + (n \times (2)) + (n \times (3)) + (4) + (5)$

Bidding

ถ้าเซิร์ฟเวอร์สามารถให้บริการต่อการร้องขอได้ก็จะทำการให้บริการเอง แต่ถ้าไม่สามารถให้บริการได้ เนื่องจากมีทรัพยากรเหลือไม่เพียงพอก็จะทำการถ่ายโอนการร้องขอไปยังเซิร์ฟเวอร์อื่นๆ ที่อยู่ในระบบ โดยเมื่อต้องการถ่ายโอนภาระงานไปให้กับเซิร์ฟเวอร์อื่นในระบบ เซิร์ฟเวอร์ที่มีภาระงานมากเกินไปจะส่งแมสเซจออกไปเพื่อถามสถานะทรัพยากรกับเซิร์ฟเวอร์อื่นๆ เราเรียกแมสเซจนี้ว่า Bidding Message เมื่อเซิร์ฟเวอร์อื่นได้รับ Bidding Message นี้ก็จะตอบกลับด้วยสถานะทรัพยากรของตนในขณะนั้น เมื่อเซิร์ฟเวอร์ที่มีภาระงานมากเกินไปได้รับสถานะของทรัพยากรจากเซิร์ฟเวอร์อื่นในระบบจนครบแล้วก็จะทำการตัดสินใจโดยเลือกถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์ที่มีทรัพยากรเหลืออยู่มากที่สุด ดังที่แสดงอยู่ในรูปที่ 3.21

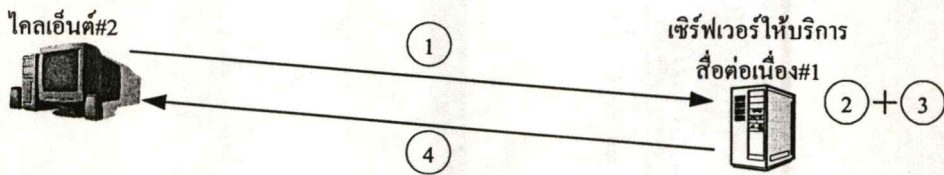


รูปที่ 3.21 การทำงานของอัลกอริทึม Bidding สำหรับสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง

เวลาเริ่มต้นให้บริการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เช่นเดียวกับอัลกอริทึมที่ผ่านมาเราสามารถแบ่งได้ออกเป็น 2 กรณีคือ กรณีที่ไม่มีการถ่ายโอนการร้องขอ และกรณีที่มีการถ่ายโอนการร้องขอ  
กรณีที่ไม่มีการถ่ายโอนการร้องขอ  
แสดงอยู่ในรูปที่ 3.22



รูปที่ 3.22 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Bidding ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่ไม่มีถ่ายโอนการร้องขอ

1. เวลาที่การร้องขอจากไคลเอนต์เดินทางมายังเซิร์ฟเวอร์ให้บริการสื่อต่อเนื่อง เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่เซิร์ฟเวอร์ใช้ในการตัดสินใจว่าจะให้บริการการร้องขอจากไคลเอนต์ที่มีเข้ามาเองเลยหรือไม่
3. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องเริ่มต้นการให้บริการต่อไคลเอนต์
4. เวลาที่ข้อมูลชิ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องไปยังไคลเอนต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

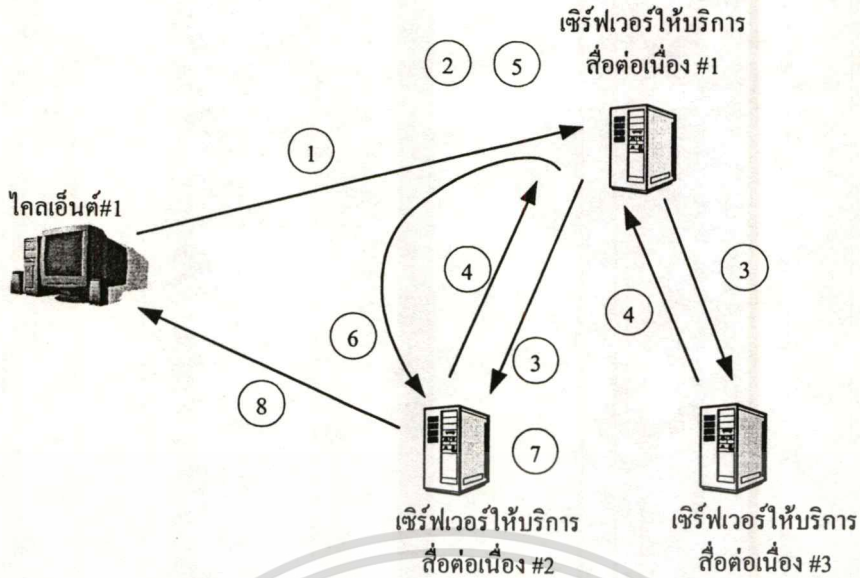
เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + (2) + (3) + (4)$$

- โดยที่เวลา (1) และ (4) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอและเวลาที่อยู่ในคิวเพื่อคอยการประมวลผลด้วย
- เวลา (3) จะขึ้นอยู่กับความสามารถของดิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1)

กรณีที่มีการถ่ายโอนการร้องขอ

แสดงอยู่ในรูปที่ 3.23



รูปที่ 3.23 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Bidding ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่มีการถ่ายโอนการร้องขอ

1. เวลาที่การร้องขอจากไคลเอ็นต์เดินทางมายังเซิร์ฟเวอร์ให้บริการสื่อก่อน เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่เซิร์ฟเวอร์ใช้ในการตัดสินใจว่าจะให้บริการการร้องขอจากไคลเอ็นต์ที่มีเข้ามาเองเลยหรือไม่ ในกรณีนี้จะต้องมีการถ่ายโอนภาระงานไปให้กับเซิร์ฟเวอร์เครื่องอื่น ก็จะมีการส่ง Bidding Message ออกไปเพื่อถามค่าทรัพยากรที่เหลืออยู่ของเซิร์ฟเวอร์ในระบบ
3. เป็นเวลาที่ Bidding Message เดินทางจากเซิร์ฟเวอร์ที่มีภาระงานมากเกินไปไปยังเซิร์ฟเวอร์เครื่องอื่นในระบบ
4. เป็นเวลาที่เซิร์ฟเวอร์อื่นในระบบตอบกลับ Bidding Message ด้วยค่าทรัพยากรที่ตนเองเหลืออยู่
5. เวลาที่ใช้ในการตัดสินใจเลือกเซิร์ฟเวอร์อื่นในระบบที่มีทรัพยากรเหลืออยู่มากที่สุด
6. เวลาที่ใช้ในการถ่ายโอนภาระงานจากเซิร์ฟเวอร์ที่มีภาระงานมากเกินไป ไปยังเซิร์ฟเวอร์ที่มีทรัพยากรเหลืออยู่มากที่สุด ผ่านระบบเครือข่าย
7. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการสื่อก่อนเริ่มต้นการให้บริการต่อไคลเอ็นต์
8. เวลาที่ข้อมูลชิ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการสื่อก่อนไปยังไคลเอ็นต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + (2) + \text{Max}((3) + (4)) + (5) + (6) + (7) + (8)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- โดยที่เวลา (1) (3) (4) (6) และ (8) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณหาค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5) ซึ่งเวลาในส่วน (3) และ (4) นั้นจะขึ้นอยู่กับเวลาที่มากที่สุดที่ใช้สำหรับการส่งและตอบรับกับ Bidding Message เท่ากับ

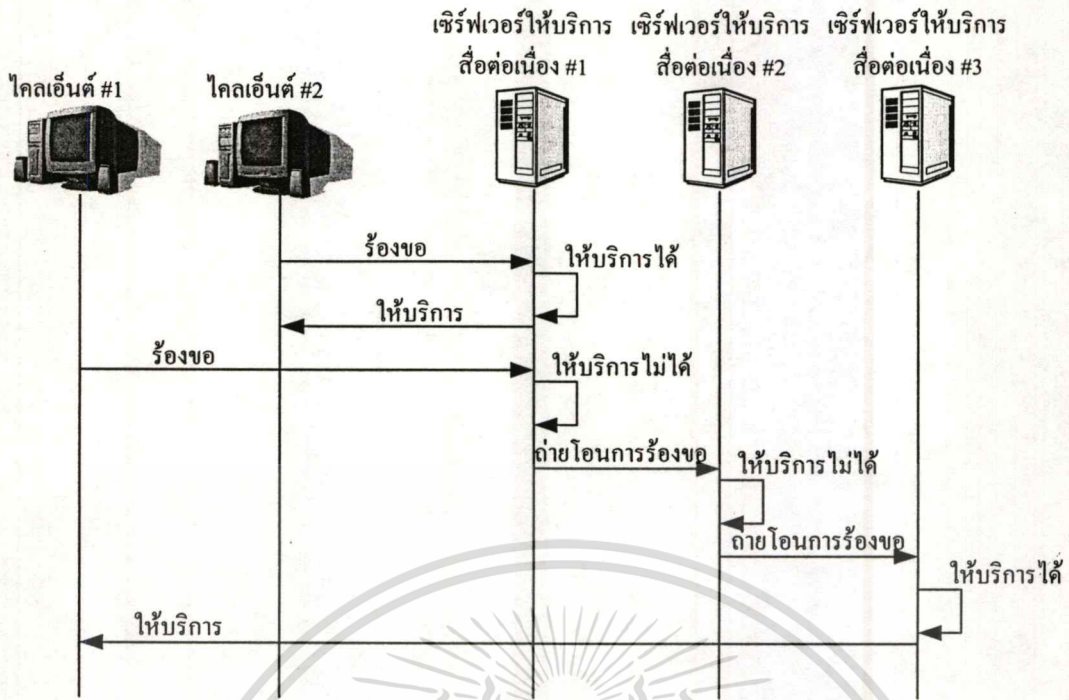
$$(3) = \text{Max}(\text{DelayBiddingMessage}(\text{Server}_1), \dots, \text{DelayBiddingMessage}(\text{Server}_n))$$

$$(4) = \text{Max}(\text{DelayBiddingResponse}(\text{Server}_1), \dots, \text{DelayBiddingResponse}(\text{Server}_n))$$

- สำหรับเวลา (2) และ (5) ที่ใช้ในการพิจารณาว่ามีภาระงานมากเกินไปหรือไม่ และเวลาที่ใช้ในการตัดสินใจเลือกเซิร์ฟเวอร์ในระบบที่มีทรัพยากรเหลือมากที่สุดสำหรับถ่ายโอนการร้องขอไปให้ และยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอและเวลาที่อยู่ในคิวเพื่อคอยการประมวลผลด้วย
- เวลา (7) จะขึ้นอยู่กับความสามารถของดิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1) ดังนั้นเวลาเริ่มต้นให้บริการจึงเท่ากับ  $\text{StartupTime} = (1) + (2) + (3) + (4) + (5) + (6) + (7) + (8)$
- นอกจากเวลาที่ใช้ดังที่ได้กล่าวไปแล้ว ถ้าระบบยอมให้มีการถ่ายโอนการร้องขอได้มากกว่า 1 ครั้ง เวลาที่ใช้ก็จะเพิ่มขึ้นตามไปด้วย โดยเวลาในส่วน (3) (4) และ (6) จะเพิ่มขึ้นตามจำนวนครั้งของการถ่ายโอนการร้องขอ

## Diffusion

ถ้าเซิร์ฟเวอร์สามารถให้บริการต่อการร้องขอได้ก็จะทำการให้บริการเอง แต่ถ้าไม่สามารถให้บริการได้ เนื่องจากมีทรัพยากรเหลือไม่เพียงพอก็จะทำการถ่ายโอนการร้องขอไปยังเซิร์ฟเวอร์อื่นๆ ที่อยู่ในระบบ สำหรับอัลกอริทึมในการถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์เครื่องอื่นนั้น เซิร์ฟเวอร์ที่มีภาระงานมากเกินไปจะทำการถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์ข้างเคียง (Neighborhood Server) ซึ่งได้มีการกำหนดไว้ก่อนแล้วซึ่งต่างจากอัลกอริทึม Round-Robin ที่จะมีการวนเลือกเซิร์ฟเวอร์ และถ้าเซิร์ฟเวอร์ที่ได้รับการถ่ายโอนการร้องขอไม่สามารถให้บริการได้ก็จะถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์ข้างเคียงไปเรื่อยๆ จนกว่าเซิร์ฟเวอร์ที่ได้รับการถ่ายโอนการร้องขอจะให้บริการต่อไคลเอนต์ได้ ดังแสดงในรูปที่ 3.24

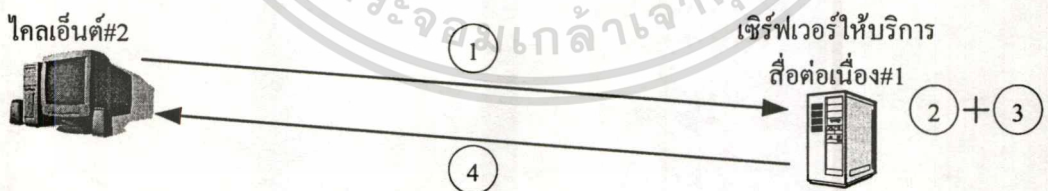


รูปที่ 3.24 การทำงานของอัลกอริทึม Diffusion สำหรับสถาปัตยกรรมที่มีตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง

เวลาเริ่มต้นให้บริการ

เช่นเดียวกับอัลกอริทึมอื่นคือ เราสามารถแบ่งได้ออกเป็น 2 กรณีคือ กรณีที่ไม่มีการถ่ายโอนการร้องขอ และกรณีที่มีการถ่ายโอนการร้องขอ

แสดงอยู่ในรูปที่ 3.25



รูปที่ 3.25 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Diffusion ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่ไม่มีถ่ายโอนการร้องขอ

1. เวลาที่การร้องขอจากไคลเอนต์เดินทางมายังเซิร์ฟเวอร์ให้บริการสื่อต่อเนื่อง เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

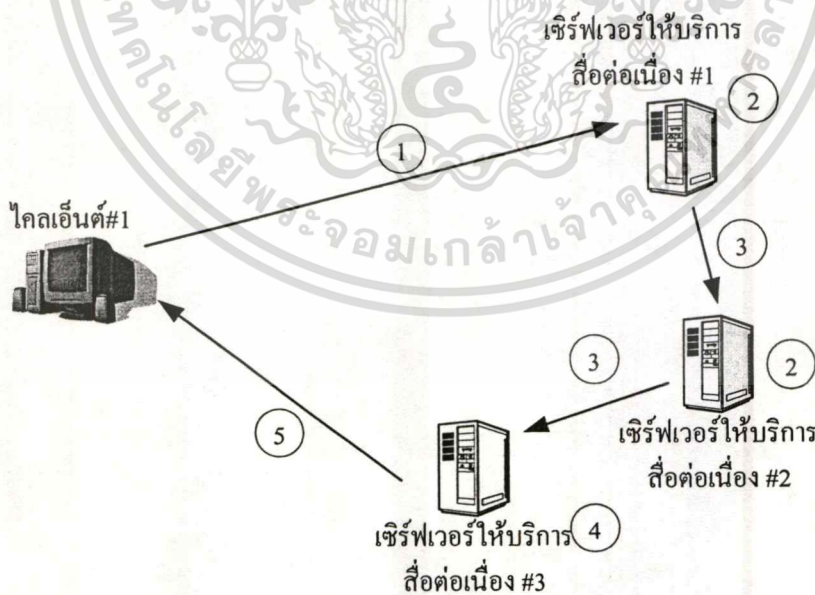
2. เวลาที่เซิร์ฟเวอร์ใช้ในการตัดสินใจว่าจะให้บริการการร้องขอจากไคลเอนต์ที่มีเข้ามาเองเลขหรือไม่
3. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องเริ่มต้นการให้บริการต่อไคลเอนต์
4. เวลาที่ข้อมูลชิ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการสื่อต่อเนื่องไปยังไคลเอนต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + (2) + (3) + (4)$$

- โดยที่เวลา (1) และ (4) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอและเวลาที่อยู่ในคิวเพื่อคอยการประมวลผลด้วย
- เวลา (3) จะขึ้นอยู่กับความสามารถของดิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1)

กรณีที่มีการถ่ายโอนการร้องขอ  
แสดงอยู่ในรูปที่ 3.26



รูปที่ 3.26 แสดงเวลาที่ใช้ในการตอบสนองสำหรับอัลกอริทึม Diffusion ของสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ในกรณีที่มีการถ่ายโอนการร้องขอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. เวลาที่การร้องขอจากไคลเอนต์เดินทางมายังเซิร์ฟเวอร์ให้บริการต่อเนื่อง เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย
2. เวลาที่เซิร์ฟเวอร์ใช้ในการตัดสินใจว่าจะให้บริการการร้องขอจากไคลเอนต์ที่มีเข้ามาเองเลยหรือไม่ ในกรณีนี้จะต้องมีการถ่ายโอนภาระงานไปให้กับเซิร์ฟเวอร์เครื่องอื่น โดยเลือกเซิร์ฟเวอร์ที่อยู่ข้างเคียง
3. เป็นเวลาที่ใช้ในการถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์เครื่องอื่น
4. เวลาที่เซิร์ฟเวอร์สำหรับให้บริการต่อเนื่องเริ่มต้นการให้บริการต่อไคลเอนต์
5. เวลาที่ข้อมูลชิ้นแรกเดินทางจากเซิร์ฟเวอร์สำหรับให้บริการต่อเนื่องไปยังไคลเอนต์ เป็นเวลาที่เกิดขึ้นระหว่างการเดินทางของข้อมูลบนระบบเครือข่าย

เวลาในการเริ่มต้นให้บริการจะเท่ากับ

$$\text{StartupTime} = (1) + \sum (2) + \sum (3) + (4) + (5)$$

- โดยที่เวลา (1) (3) และ (5) นั้นเป็นเวลาที่เกิดขึ้นเนื่องจากระบบเครือข่ายซึ่งสามารถใช้โมเดลทางคณิตศาสตร์คำนวณค่าโดยประมาณได้ ซึ่งค่าเวลานี้ก็จะแตกต่างกันไปตามระบบเครือข่ายที่ใช้ (ที่ใช้ในการจำลองการทำงานอยู่ในหัวข้อ 4.2.5)
- สำหรับเวลา (3) นั้นนอกจากจะขึ้นกับระบบเครือข่ายแล้วยังขึ้นกับจำนวนครั้งของการถ่ายโอนการร้องขอด้วย ดังนั้น เวลาในส่วนที่ (3) นั้นจะเท่ากับเวลาทั้งหมดที่ใช้ในการถ่ายโอนการร้องขอ
- สำหรับเวลา (2) นั้นนอกจากจะเป็นเวลาที่ใช้ในการตัดสินใจแล้วยังรวมถึงเวลาที่ใช้ในการประมวลผลการร้องขอและเวลาที่อยู่ในคิวเพื่อคอยการประมวลผลด้วย
- เวลา (4) จะขึ้นอยู่กับความสามารถของดิสก์ในเซิร์ฟเวอร์ (ที่ใช้ในการจำลองการทำงานแสดงอยู่ในหัวข้อ 4.2.3.1) ดังนั้นเวลาเริ่มต้นให้บริการจึงเท่ากับ  $\text{StartupTime} = (1) + (n \times (2)) + (n \times (3)) + (4) + (5)$

## บทที่ 4

### การจำลองการทำงาน

#### 4.1 โปรแกรมจำลองการทำงาน

เป็นโปรแกรมที่พัฒนาขึ้นโดยใช้โมเดลทั้งจากสถาปัตยกรรมและอัลกอริทึมในการแบ่งภาระงานที่ได้กล่าวไปแล้ว มีวัตถุประสงค์เพื่อจำลองการทำงานในหลายๆ สถานะซึ่งไม่เหมาะที่จะทำการให้บริการจริง ตัวโปรแกรมพัฒนาขึ้นโดยใช้เครื่องมือ Delphi เวอร์ชัน 5.0

สำหรับโครงสร้างภายในของโปรแกรมจำลองการทำงานนั้นประกอบด้วย ส่วนที่ทำหน้าที่สร้างการร้องขอ ส่วนที่ทำหน้าที่เป็นตัวแบ่งภาระงานที่ศูนย์กลาง ส่วนที่ทำหน้าที่เป็นเซิร์ฟเวอร์ในการให้บริการสื่อแบบต่อเนื่อง และส่วนที่ทำหน้าที่บันทึกข้อมูลผลการจำลองการทำงานลงในไฟล์ ส่วนประกอบเหล่านี้จะทำงานเป็นเธรด (Thread) เพื่อให้สามารถทำงานไปได้พร้อมๆ กันเหมือนกับเป็นส่วนประกอบที่ทำงานอยู่ในระบบจริง

ส่วนที่ทำหน้าที่สร้างการร้องขอ เป็นส่วนที่ทำหน้าที่สร้างการร้องขอสื่อที่ต้องการใช้งานขึ้นตามคาบเวลาที่ได้กำหนดไว้ โดยหมายเลขของสื่อที่เข้าใช้นั้นจะอ่านจากไฟล์ที่บรรจุหมายเลขของสื่อที่ต้องการเข้าใช้ไว้ โดยในไฟล์นี้มีการกำหนดหมายเลขสื่อตามที่อยู่ใน 4.2.2.2 ส่วนคาบเวลาก็จะอ่านจากไฟล์ที่บรรจุคาบเวลาระหว่างการร้องขอ โดยในไฟล์นี้มีการกำหนดคาบเวลาตามที่อยู่ใน 4.2.1 สำหรับการร้องขอที่ถูกสร้างขึ้นจะส่งไปให้กับส่วนที่ทำหน้าที่เป็นตัวแบ่งภาระงานที่ศูนย์กลางในกรณีที่เป็นสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง และจะถูกส่งต่อไปให้กับส่วนที่เป็นเซิร์ฟเวอร์ในการให้บริการสื่อแบบต่อเนื่องในกรณีที่เป็นสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์

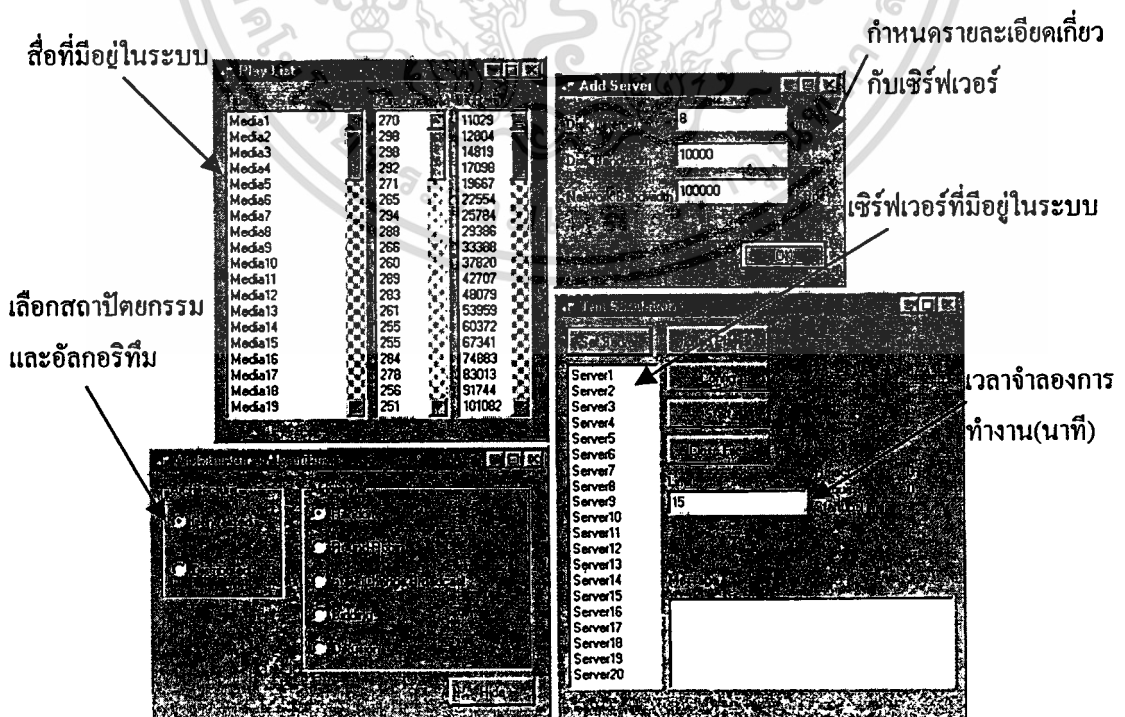
ส่วนที่ทำหน้าที่เป็นตัวแบ่งภาระงานที่ศูนย์กลางจะทำการรับการร้องขอที่ถูกส่งมาจากส่วนที่ทำหน้าที่สร้างการร้องขอ ในกรณีที่เป็นสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง โดยในส่วนนี้จะมีคิวของการร้องขอสำหรับประมวลผลเพื่อตัดสินใจส่งการร้องไปให้กับเซิร์ฟเวอร์ตามอัลกอริทึมที่ได้กำหนดไว้

ส่วนที่ทำหน้าที่เป็นเซิร์ฟเวอร์ในการให้บริการสื่อแบบต่อเนื่อง เมื่อได้รับการร้องขอเข้ามาแล้วก็จะทำการเก็บไว้ในคิวก่อน เพื่อรอเวลาที่จัดสรรไว้ในการประมวลผล เมื่อถึงคิวของการร้องขอที่มีเข้ามาใหม่ ส่วนที่ทำหน้าที่เป็นเซิร์ฟเวอร์ก็จะทำการตัดสินใจว่ามีทรัพยากรเพียงพอหรือไม่ โดยพิจารณาจากแบนด์วิดธ์ของดิสก์และเน็ตเวิร์กที่ถูกใช้งานไปแล้ว ถ้าเซิร์ฟเวอร์มีทรัพยากรไม่เพียงพอก็จะปฏิเสธการร้องขอเลยในกรณีที่เป็นสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง ส่วนเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในกรณีที่เป็นการสาปฎิการแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ก็จะตัดสินใจว่าจะปฏิเษในกรณีที่การร้องขอได้ถูกถ่ายโอนมาก่อนหน้าแล้วตามจำนวนที่กำหนดไว้ (ในการจำลองยอมให้มีการถ่ายโอนการร้องขอ 1 ครั้ง และเท่ากับจำนวนเซิร์ฟเวอร์ที่มีในระบบสำหรับอัลกอริทึม Diffusion) หรือถ่ายโอนภาระงานไปให้กับเซิร์ฟเวอร์อื่น ถ้าเซิร์ฟเวอร์ตัดสินใจว่าการร้องจะได้รับบริการ การร้องขอก็จะถูกใส่ลงในคิวสำหรับการอ่านดิสก์ พร้อมทั้งมีการตรวจสอบเวลาที่ให้บริการว่าการร้องขอนั้นได้ให้บริการจนสิ้นสุดความยาวของสื่อหรือยังด้วย

และส่วนที่ทำหน้าที่บันทึกข้อมูลผลการจำลองการทำงานลงในไฟล์ เป็นส่วนที่ทำหน้าที่อ่านข้อมูลที่มีความสำคัญจากส่วนประกอบอื่นๆ จากนั้นก็จะทำการรวบรวมและบันทึกลงในไฟล์ตามช่วงเวลาที่กำหนด (ทุกๆ 10 วินาที) โดยข้อมูลที่ทำกรบันทึกนั้นประกอบด้วย เวลาในการเริ่มต้นให้บริการ จำนวนการร้องขอที่ถูกปฏิเสธ และแบนด์วิดธ์ของดิสก์ที่ถูกใช้งานไปแล้วสำหรับการคำนวณค่า Disk Utilization

สำหรับทำงานของโปรแกรมจำลองการทำงาน (รูปที่ 4.1) ขั้นแรกก็จะต้องอ่านไฟล์ข้อมูลที่บันทึกหมายเลขของสื่อ ความยาว และอัตราการเล่นของสื่อขึ้นมา จากนั้นก็อ่านไฟล์ที่บรรจุหมายเลขของสื่อที่จะเรียกใช้ อ่านไฟล์คาบเวลา ทำการเพิ่มเซิร์ฟเวอร์เข้าไปในโปรแกรมจำลองการทำงาน โดยจะต้องกำหนดค่าหน่วยเวลาและแบนด์วิดธ์ของดิสก์ และกำหนดแบนด์วิดธ์ของระบบเครือข่าย จากนั้นเลือกสถาปัตยกรรมและอัลกอริทึมในการทำงาน กำหนดระยะเวลาที่ต้องการจำลองการทำงาน จากนั้นก็เริ่มจำลอง เมื่อโปรแกรมจำลองการทำงานดำเนินไปจนเสร็จแล้ว ข้อมูลต่างๆ ที่มีความสำคัญก็จะบรรจุอยู่ในไฟล์ซึ่งสามารถนำมาใช้ในการวิเคราะห์และประเมินผลต่อไป



รูปที่ 4.1 โปรแกรมจำลองการทำงาน

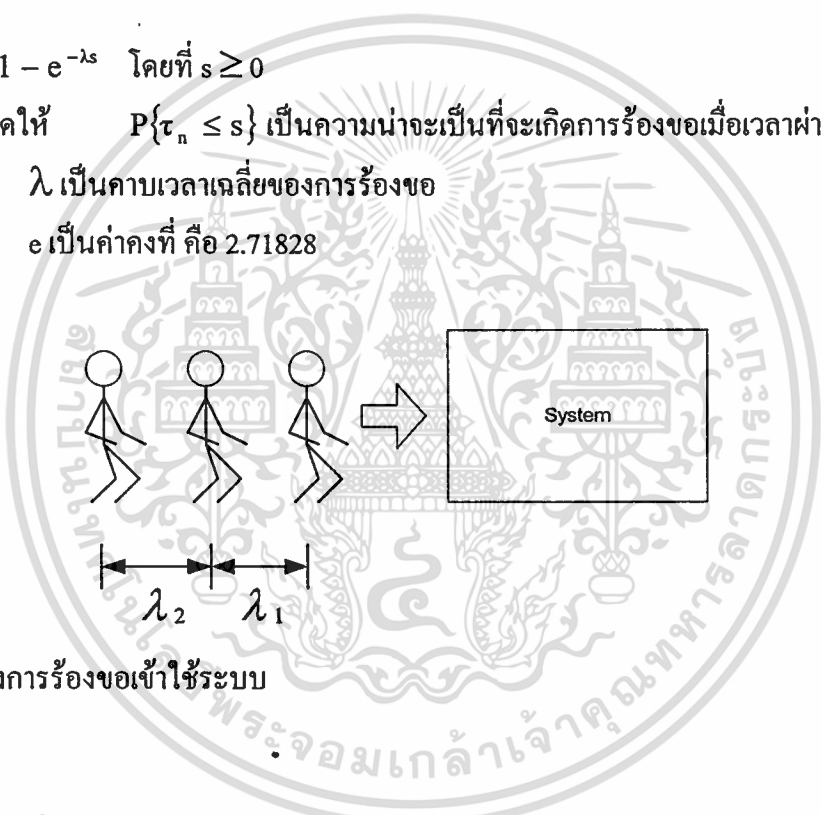
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 สมมุติฐานและพารามิเตอร์ที่ใช้ในการจำลองการทำงาน

4.2.1 อัตราการร้องขอ

เป็นการจำลองโมเดล Queueing Network (รูปที่ 4.2) โดยกำหนดให้อัตราการร้องขอของการเข้าใช้บริการสื่อแบบต่อเนื่องเท่ากับอัตราการเข้ามา (Arrival Rate) ของโมเดล โดยอัตราการร้องขอนั้นมีค่าเป็นไปตาม Poisson Distribution โดยระยะห่างระหว่างแต่ละการร้องขอจะมีการกระจายของความน่าจะเป็นตาม Exponential Distribution

$P\{\tau_n \leq s\} = 1 - e^{-\lambda s}$  โดยที่  $s \geq 0$   
กำหนดให้  $P\{\tau_n \leq s\}$  เป็นความน่าจะเป็นที่จะเกิดการร้องขอเมื่อเวลาผ่านไป  $s$   
 $\lambda$  เป็นคาบเวลาเฉลี่ยของการร้องขอ  
 $e$  เป็นค่าคงที่ คือ 2.71828



รูปที่ 4.2 แสดงการร้องขอเข้าใช้ระบบ

4.2.2 สื่อ

4.2.2.1 รูปแบบของสื่อ

เป็นวิดีโอคลิป (Video Clip) ที่มีความยาวอยู่ระหว่าง 3-5 นาที (180-300 วินาที)  
ต้องการแบนด์วิธตั้งแต่ 70 – 200 กิโลไบต์ต่อวินาที

#### 4.2.2.2 การเรียกใช้สื่อ

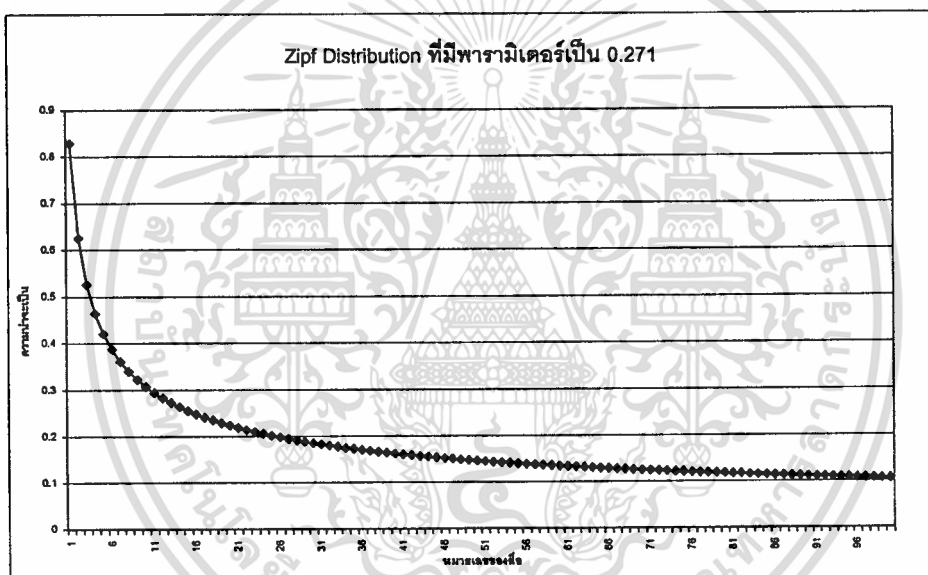
สำหรับความน่าจะเป็นในการร้องขอเพื่อใช้สื่อแต่ละสื่อ นั้นเป็นไปตามความนิยมของสื่อ ซึ่งมีรูปแบบที่ใกล้เคียงกับ Zipf Distribution ที่มีค่าพารามิเตอร์เท่ากับ 0.271 [46] โดยกราฟการกระจายแสดงอยู่ในรูปที่ 4.3

$$q_m = \frac{G}{m^\zeta}, m=1, \dots, M$$

$$G = \frac{1}{1 + \frac{1}{2^\zeta} + \dots + \frac{1}{M^\zeta}}$$

$q_m$  เป็นความน่าจะเป็นของวิดีโอที่ได้รับความนิยมลำดับที่  $m$

$\zeta$  เป็นพารามิเตอร์ (ของการให้บริการวิดีโอเท่ากับ 0.271)



รูปที่ 4.3 Zipf Distribution ที่มีค่าพารามิเตอร์เท่ากับ 0.271

#### 4.2.3 เซิร์ฟเวอร์ให้บริการสื่อแบบต่อเนื่อง

กำหนดให้สภาวะแวดล้อมที่เซิร์ฟเวอร์มีความหลากหลาย (Heterogeneous Environment) การทำงานของเซิร์ฟเวอร์ในการให้บริการสื่อ นั้นเป็นแบบที่เซิร์ฟเวอร์เป็นผู้ควบคุมการส่งข้อมูลทั้งหมด (Server Push) และบริการสื่อแบบต่อเนื่องของเซิร์ฟเวอร์นั้นเป็นการเล่นเพียงอย่างเดียว

#### 4.2.3.1 ดิสก์

สำหรับดิสก์ที่ใช้ในการจำลองการทำงานนั้นเป็นฮาร์ดดิสก์ที่อยู่ภายในเซิร์ฟเวอร์ โดยเวลาที่ใช้ในการเข้าใช้งานดิสก์เพื่ออ่านข้อมูลคือ

เวลาที่เกิดจากคำสั่ง I/O+เวลาในการค้นหา (Seek Time)+เวลาในการอ่านข้อมูล+เวลาที่ใช้ในการถ่ายโอนข้อมูล

สำหรับในการจำลองการทำงานนั้น เวลาที่เกิดขึ้นจากคำสั่ง I/O จะมีค่าน้อยมาก ดังนั้นเวลาที่ใช้ในการพิจารณาจึงประกอบด้วย

เวลาที่ใช้ในการค้นหาเฉลี่ย (Average Seek Time)+เวลาในการอ่านข้อมูล(Latency)+ขนาดข้อมูล/อัตราการถ่ายโอนข้อมูลเฉลี่ย (Average Transfer Rate)

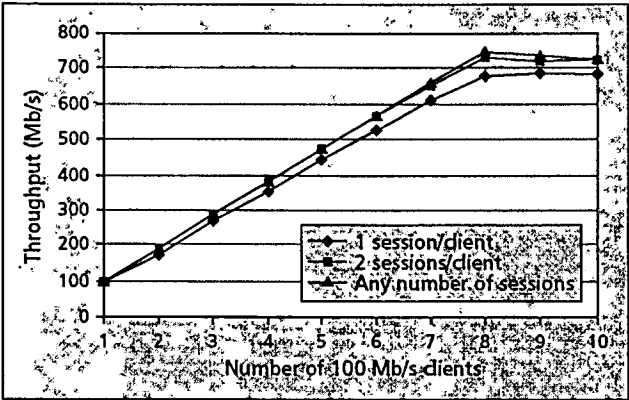
สำหรับการจำลองการทำงานเวลาที่ใช้ในการค้นหาและเวลาที่ใช้ในการอ่านข้อมูล กำหนดให้เป็นค่าคงที่ (8 มิลลิวินาที) โดยอัตราการถ่ายโอนข้อมูลเฉลี่ยมีด้วยกัน 2 ค่าคือ 10 และ 40 เมกะไบต์ต่อวินาที

#### 4.2.4 ตัวแบ่งภาระงานที่ศูนย์กลาง

เป็นคอมพิวเตอร์ที่รับหน้าที่สำหรับรับการร้องขอจากไคลเอนต์ แล้วทำการตัดสินใจและส่งการร้องขอไปให้กับเซิร์ฟเวอร์ที่มีอยู่ในระบบเพื่อให้บริการ ตัวแบ่งภาระงานนี้จะมีอัตราในการตัดสินใจที่คงที่คือ 10 การร้องขอต่อวินาที

#### 4.2.5 ระบบเครือข่าย

ในการจำลองการทำงานระบบเครือข่ายที่ใช้เป็นการเชื่อมต่อผ่านสวิตช์อีเธอร์เน็ตความเร็ว 100 เมกะบิตต่อวินาที ซึ่งจาก [47] แสดงให้เห็นว่าแบนด์วิธของระบบเครือข่ายที่มีการเชื่อมต่อดังที่แสดงในรูปที่ 3.1 และ รูปที่ 3.2 สำหรับเซิร์ฟเวอร์แต่ละเครื่องจะอยู่ที่ประมาณ 94.9 เมกะไบต์ต่อวินาที หรือประมาณ 11.86 เมกะไบต์ต่อวินาที ดังที่แสดงในรูปที่ 4.4 สำหรับเวลาที่ใช้ในการส่งข้อมูลบนระบบเครือข่ายนี้จะเท่ากับ  $\text{StartupLatency} + \frac{\text{DataSize}}{11862.5}$  ในหน่วยมิลลิวินาที ซึ่งโดยทั่วไปแล้วค่า StartupLatency จะน้อยมากจนสามารถละทิ้งได้ (19.4 ไมโครวินาที [47])



รูปที่ 4.4 กราฟแสดงแบนด์วิดธ์ของแต่ละเซิร์ฟเวอร์เมื่อใช้สวิตช์ความเร็ว 100 เมกะบิตต่อวินาที [47]

4.3 มาตรการที่ใช้ในการจำลองการทำงาน

สำหรับการจำลองการทำงานนั้น มาตรการที่สำคัญที่ใช้ประกอบด้วย

- จำนวนการร้องขอที่ได้รับบริการ (Served Request)
- จำนวนการร้องขอที่ถูกปฏิเสธ (Rejected Request)
- เวลาเริ่มต้นให้บริการ (Startup Time)

ในส่วนของเวลาเริ่มต้นให้บริการนั้นจะวัดจากเวลาที่การร้องขอเข้ามาสู่สวิตช์ จนกระทั่งการตอบรับได้ถูกส่งออกไปจากสวิตช์

- Disk Utilization

ในส่วนของ Disk Utilization สามารถคำนวณได้จาก

$$\text{DiskUtilization} = \frac{\text{UsedBandwidth}}{\text{AvailableBandWidth}}$$

4.4 ผลการจำลองการทำงานที่ 1

จุดประสงค์

- เพื่อเปรียบเทียบความสามารถในการขยายการให้บริการของระบบ

ตัวแปรควบคุม

- เซิร์ฟเวอร์แต่ละเครื่องมีค่าหน่วยเวลาของดิสก์เท่ากับ 7 มิลลิวินาที และมีแบนด์วิดธ์ของดิสก์ 10 เมกะไบต์ต่อวินาที
- คาบเวลาเฉลี่ยระหว่างการร้องขอ 100 มิลลิวินาที

- อัลกอริทึม State Change Broadcast ของสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

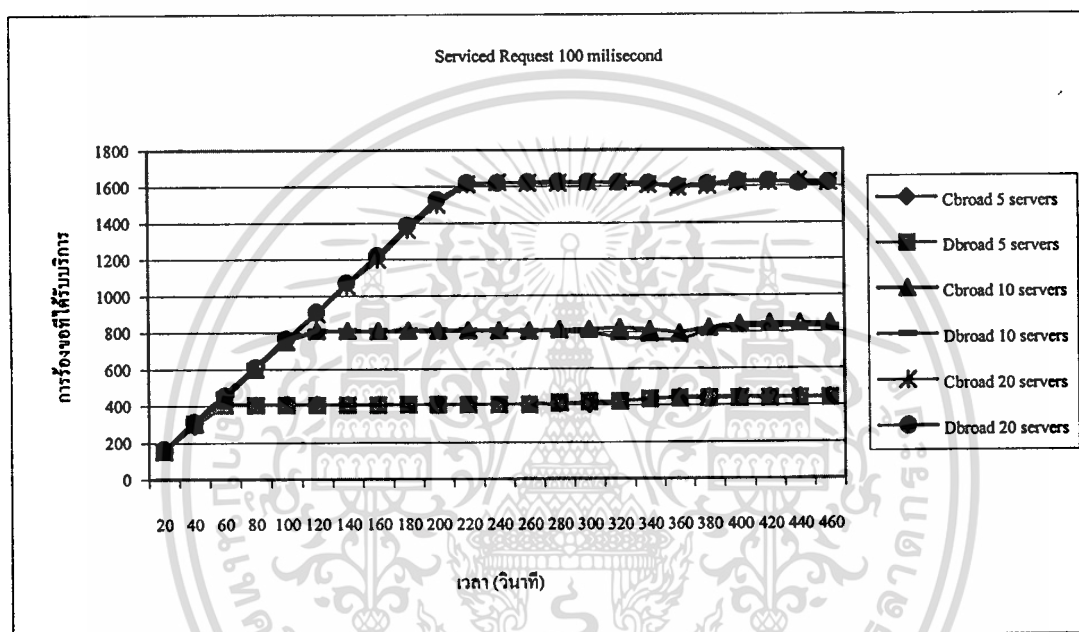
ตัวแปรที่พิจารณา

- จำนวนเซิร์ฟเวอร์ 5, 10 และ 20 เครื่อง

มาตรวัด

- จำนวนการร้องขอที่ได้รับการให้บริการ

ผลการจำลองการทำงาน



รูปที่ 4.5 กราฟแสดงจำนวนการร้องขอที่ได้รับการบริการในจำนวนเซิร์ฟเวอร์ที่แตกต่างกัน

วิเคราะห์ผลการจำลองการทำงาน

จากการจำลองการทำงาน (รูปที่ 4.5) แสดงให้เห็นว่าเมื่อจำนวนเซิร์ฟเวอร์ขึ้นเป็นสองเท่า คือจาก 10 เซิร์ฟเวอร์เป็น 20 เซิร์ฟเวอร์ ระบบก็จะมีสามารถในการให้บริการเพิ่มในอัตราส่วนที่ใกล้เคียงกับ 2 เท่าด้วย (จาก 8xx การร้องขอเป็น 1,6xx การร้องขอ) ทำนองเดียวกับการเพิ่มจำนวนเซิร์ฟเวอร์จาก 5 เป็น 10 และ 5 เป็น 20 เครื่อง ซึ่งจากความสามารถที่เพิ่มขึ้นเป็นอัตราส่วนนี้ทำให้เราสามารถประมาณจำนวนของเซิร์ฟเวอร์ในระบบเพื่อให้สอดคล้องกับความต้องการของผู้ใช้ ทั้งนี้ระบบจะต้องมีความเร็วของระบบเครือข่ายที่เพียงพอด้วย

## 4.5 ผลการจำลองการทำงานที่ 2

### จุดประสงค์

- เพื่อเปรียบเทียบความสามารถในการขยายการให้บริการของระบบในกรณีที่เพิ่มความสามารถของฮาร์ดแวร์

### ตัวแปรควบคุม

- จำนวนเซิร์ฟเวอร์ 5 เครื่อง
- คาบเวลาเฉลี่ยระหว่างการร้องขอ 100 มิลลิวินาที
- อัลกอริทึม State Change Broadcast ของสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

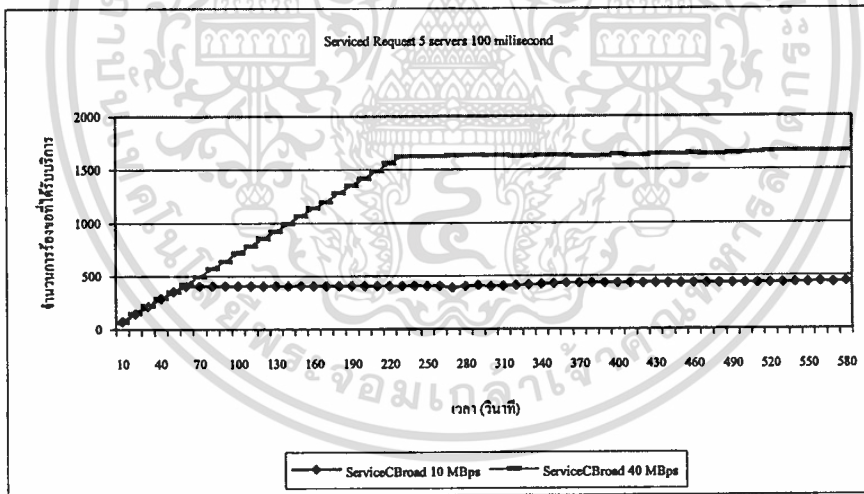
### ตัวแปรที่พิจารณา

- ดิสก์แบนด์วิดท์ของเซิร์ฟเวอร์ระหว่าง 10 เมกะไบต์ต่อวินาทีกับ 40 เมกะไบต์ต่อวินาที

### มาตรวัด

- จำนวนการร้องขอที่ได้รับการให้บริการ

### ผลการจำลองการทำงาน



รูปที่ 4.6 กราฟแสดงจำนวนการร้องขอที่ได้รับการบริการในระบบที่เซิร์ฟเวอร์มีดิสก์แบนด์วิดท์แตกต่างกัน

### วิเคราะห์ผลการจำลองการทำงาน

จากการจำลองการทำงาน (รูปที่ 4.6) แสดงให้เห็นว่าเมื่อระบบมีการเพิ่มแบนด์วิดท์ของดิสก์ขึ้นเป็นสี่เท่า คือจาก 10 เป็น 40 เมกะไบต์ต่อวินาที ก็จะส่งผลให้ระบบมีความสามารถในการให้บริการเพิ่มในอัตราส่วนที่ใกล้เคียงกับ 4 เท่าด้วย ซึ่งจากความสามารถที่เพิ่มขึ้นเป็นอัตราส่วนนี้ เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำให้เราสามารถประมาณจำนวนและความสามารถของเซิร์ฟเวอร์ในระบบเพื่อให้รองรับกับความต้องการของผู้ใช้ได้เพียงพอ

#### 4.6 ผลการจำลองการทำงานที่ 3

##### จุดประสงค์

- เพื่อศึกษาปัจจัยของความยาวสื่อและอัตราการร้องขอที่มีผลต่อความสามารถในการให้บริการของระบบ

##### ตัวแปรควบคุม

- เซิร์ฟเวอร์แต่ละเครื่องมีค่าหน่วยเวลาของดิสก์เท่ากับ 7 มิลลิวินาที และมีแบนด์วิดท์ของดิสก์ 10 เมกะไบต์ต่อวินาที
- สถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง และอัลกอริทึม State Change Broadcast
- อัตราการตัดสินใจของตัวแบ่งภาระงานที่ศูนย์กลาง 10 การร้องขอต่อวินาที

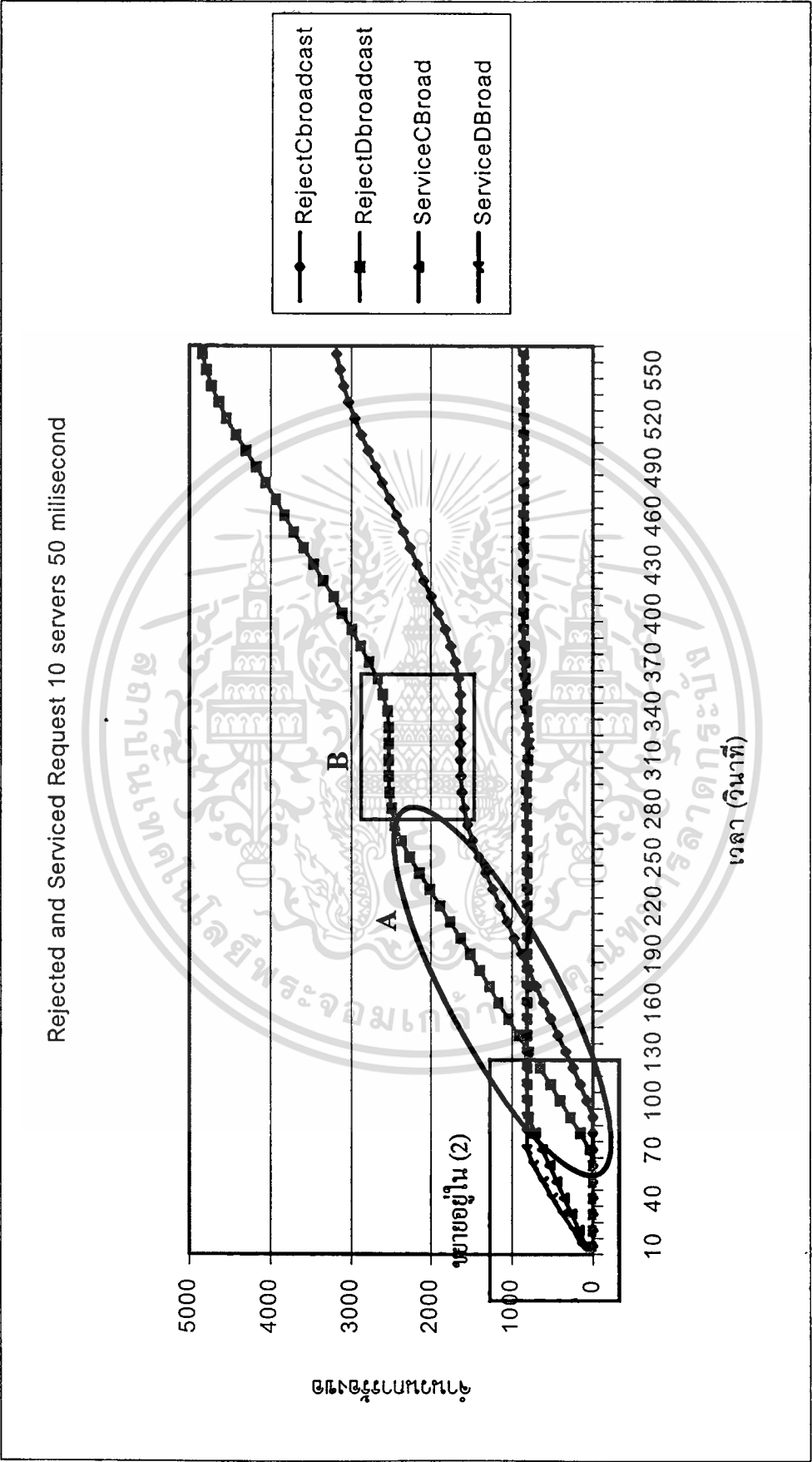
##### ตัวแปรที่พิจารณา

- จำนวนเซิร์ฟเวอร์ 10 เซิร์ฟเวอร์

##### มาตรวัด

- จำนวนการร้องขอที่ถูกปฏิเสธ
- จำนวนการร้องขอที่ได้รับการให้บริการ

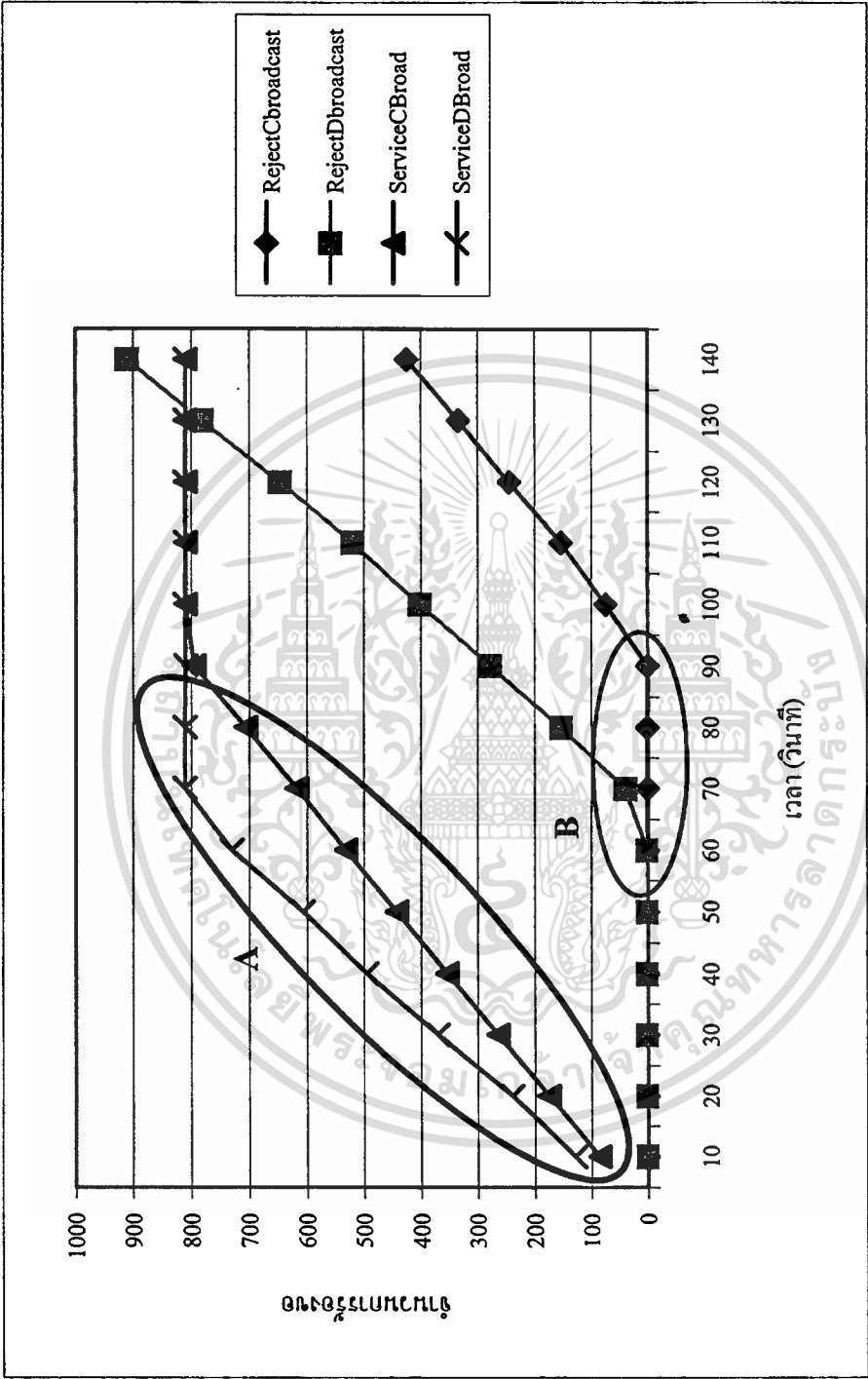
##### ผลการจำลองการทำงาน



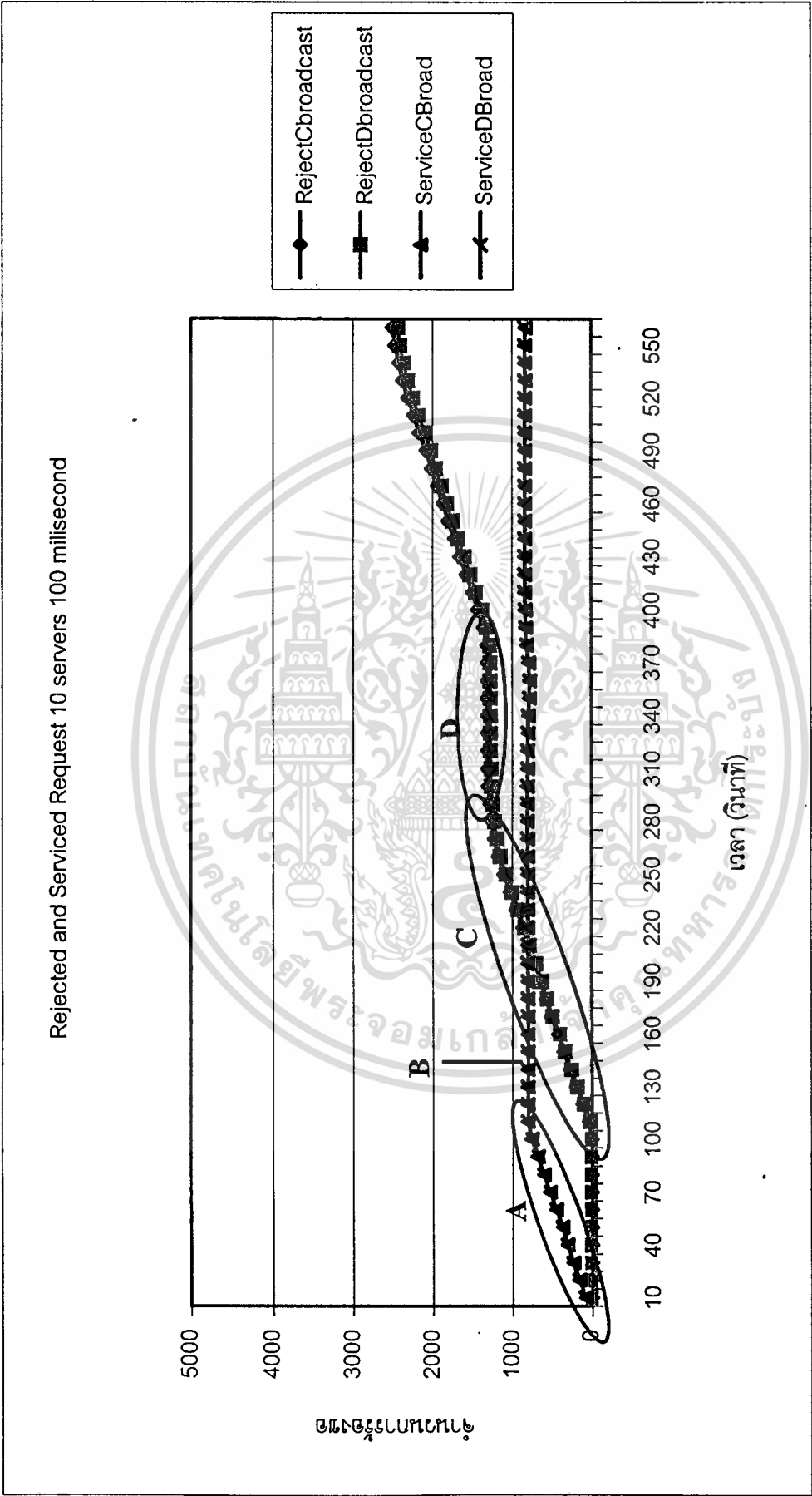
(1)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.7 กราฟแสดงจำนวนการร้องขอที่ถูกปฏิเสธและได้รับการให้บริการในระบบที่มีเซิร์ฟเวอร์ 10 เครื่อง ด้วยความแปรเปลี่ยนระหว่างการร้องขอ 50 มิลลิวินาที



(2)



รูปที่ 4.8 กราฟแสดงจำนวนการร้องขอที่ถูกปฏิเสธและได้รับบริการในระบบที่มีเซิร์ฟเวอร์ 10 เครื่อง ด้วยความละเอียดระหว่างการร้องขอ 100 มิลลิวินาที

## วิเคราะห์ผลการจำลองการทำงาน

จากการจำลองการทำงานพบว่า จำนวนการร้องขอที่ให้บริการจะเพิ่มขึ้นในช่วงแรก (ในช่วง A ของรูปที่ 4.8) ไปจนถึงจุดจุดหนึ่งซึ่งเป็นความสามารถสูงสุดของระบบ จำนวนการร้องขอที่ให้บริการก็จะคงที่ (ในช่วง B ของรูปที่ 4.8) ถึงแม้ว่าระบบให้บริการด้วยความสามารถสูงสุด หลังจากนั้นก็ยังมีการป้อนการร้องขอเข้าไปอีก ซึ่งก็จะทำให้ระบบปฏิเสธการให้บริการ ส่งผลให้จำนวนการร้องขอที่ถูกปฏิเสธสูงขึ้น (ในช่วง C ของรูปที่ 4.8) เมื่อเวลาดำเนินผ่านไประยะหนึ่ง การให้บริการการร้องขอเริ่มเสร็จลงทำให้เซิร์ฟเวอร์สามารถนำทรัพยากรที่ถูกใช้ในการให้บริการเหล่านั้นกลับมาให้บริการการร้องขอที่มีเข้ามาใหม่ได้ ซึ่งในกรณีนี้จำนวนการร้องขอที่ได้รับการให้บริการก็ยังมีจำนวนคงที่ และจำนวนการร้องขอที่ถูกปฏิเสธจะไม่เพิ่มขึ้น (ในช่วง D ของรูปที่ 4.8) เนื่องจากการร้องขอที่เข้ามาใหม่จะได้รับการให้บริการแทนการร้องขอเดิมที่เสร็จสิ้นลง และเมื่อการร้องขอที่ได้รับการก่อนหน้าได้รับการจนเสร็จก็ทำให้ระบบไม่มีทรัพยากรเหลือสำหรับการร้องขอที่มีเข้ามาใหม่ ส่งผลให้จำนวนการร้องขอที่ถูกปฏิเสธสูงขึ้นอีก โดยพฤติกรรมของระบบจะเป็นเช่นนี้เรื่อยไป นอกจากนั้นอัตราการร้องขอยังมีผลโดยตรงกับจำนวนการร้องขอที่ได้รับการปฏิเสธ ในสถานะที่อัตราการร้องขอสูง (รูปที่ 4.7(1)) จำนวนการร้องขอที่ระบบปฏิเสธจะสูงกว่า (ช่วง A ในรูปที่ 4.7(1)) สถานะที่อัตราการร้องขอต่ำ (ช่วง C ในรูปที่ 4.8) และเช่นเดียวกันระยะเวลาที่ระบบหยุดปฏิเสธการร้องขอในสถานะที่อัตราการร้องขอสูง (ช่วง B ในรูปที่ 4.7(1)) ก็สั้นกว่าสถานะที่อัตราการร้องขอต่ำกว่า (ช่วง D ในรูปที่ 4.8)

คุณลักษณะของการปฏิเสธการให้บริการของระบบดังที่ได้กล่าวมาจะมีผลกระทบต่อความสามารถในการให้บริการของระบบเมื่อเวลาดำเนินผ่านไป นั่นคือ ถ้าไม่ต้องการให้ระบบปฏิเสธการร้องขอหรือต้องการให้ปฏิเสธน้อยที่สุดแล้ว ความสามารถในการให้บริการของระบบจะต้องสูงกว่า อัตราการร้องขอต่อวินาทีคูณด้วยความยาวเฉลี่ยของสื่อในหน่วยวินาที

นอกจากนั้นจาก ในรูปที่ 4.7 ซึ่งเป็นกราฟที่แสดงจำนวนการร้องขอที่ได้รับการให้บริการและจำนวนการร้องขอที่ถูกปฏิเสธจะเห็นได้ว่า ในสถานะที่อัตราการร้องขอที่มีเข้ามาสู่ระบบสูงกว่าอัตราการตัดสินใจของตัวแบ่งภาระงานที่ศูนย์กลาง จำนวนการร้องขอที่ได้รับการให้บริการ (ช่วง A ในรูปที่ 4.6 (2)) และเวลาที่ระบบเริ่มปฏิเสธการร้องขอ (ช่วง B ในรูปที่ 4.6(2)) ของแต่ละสถาปัตยกรรมจะมีความแตกต่างกัน โดยสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์จะมีจำนวนการร้องขอที่ได้รับการให้บริการสูงกว่าในช่วงที่ยังไม่มีการปฏิเสธการร้องขอ แต่จะเท่ากันเมื่อระบบให้บริการด้วยความสามารถสูงสุด โดยการที่ระบบที่มีสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์จะเริ่มปฏิเสธการร้องขอเร็วกว่าสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง ทั้งนี้เนื่องจากในสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์นั้น การร้องขอจะถูกส่งไปที่เซิร์ฟเวอร์ทันที ในขณะที่สถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลางการร้องขอจะต้องถูกเข้าคิวที่ตัวแบ่งภาระงานเพื่อกระจายให้กับเซิร์ฟเวอร์ก่อน ดังนั้นอัตราการร้องขอที่ถูกส่งให้เอกสารเป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กับกลุ่มของเซิร์ฟเวอร์ภายในระบบจึงเท่ากับอัตราการตัดสินใจของตัวแบ่งภาระงานที่ศูนย์กลาง คุณลักษณะเช่นนี้จะมีผลกระทบต่อเวลาเริ่มต้นในการให้บริการ เนื่องจากการร้องขอจะต้องคอยอยู่ในคิวเป็นระยะเวลาหนึ่งซึ่งจะมากหรือน้อยก็ขึ้นอยู่กับความแตกต่างระหว่างอัตราการร้องขอที่มีเข้ามาเปรียบเทียบกับอัตราการตัดสินใจของตัวแบ่งภาระงานที่ศูนย์กลาง

#### 4.7 ผลการจำลองการทำงานที่ 4

##### จุดประสงค์

- เพื่อประเมินความสามารถในการแบ่งภาระงานโดยพิจารณาจากการร้องขอที่ถูกปฏิเสธ

##### ตัวแปรควบคุม

- จำนวนเซิร์ฟเวอร์ 20 เครื่อง โดยเซิร์ฟเวอร์แต่ละเครื่องมีค่าหน่วยเวลาของดิสก์เท่ากับ 7 มิลลิวินาที และมีแบนด์วิดท์ของดิสก์ 10 เมกะไบต์ต่อวินาที
- คาบเวลาเฉลี่ยระหว่างการร้องขอ 100 มิลลิวินาที
- จำนวนการร้องขอทั้งหมด 5,000 การร้องขอ

##### ตัวแปรที่พิจารณา

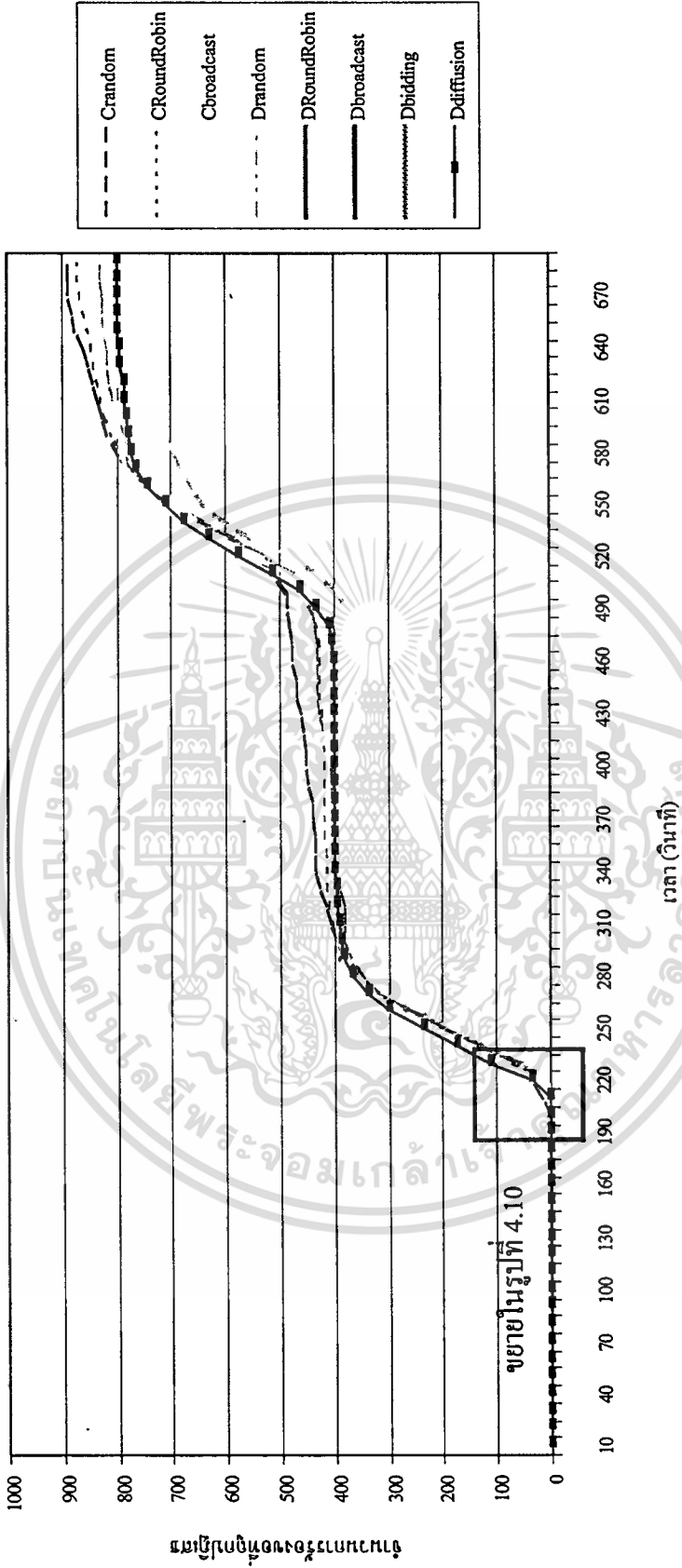
- อัลกอริทึมสำหรับสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลางและที่กระจายอยู่ที่เซิร์ฟเวอร์

##### มาตรวัด

- จำนวนการร้องขอที่ถูกปฏิเสธ

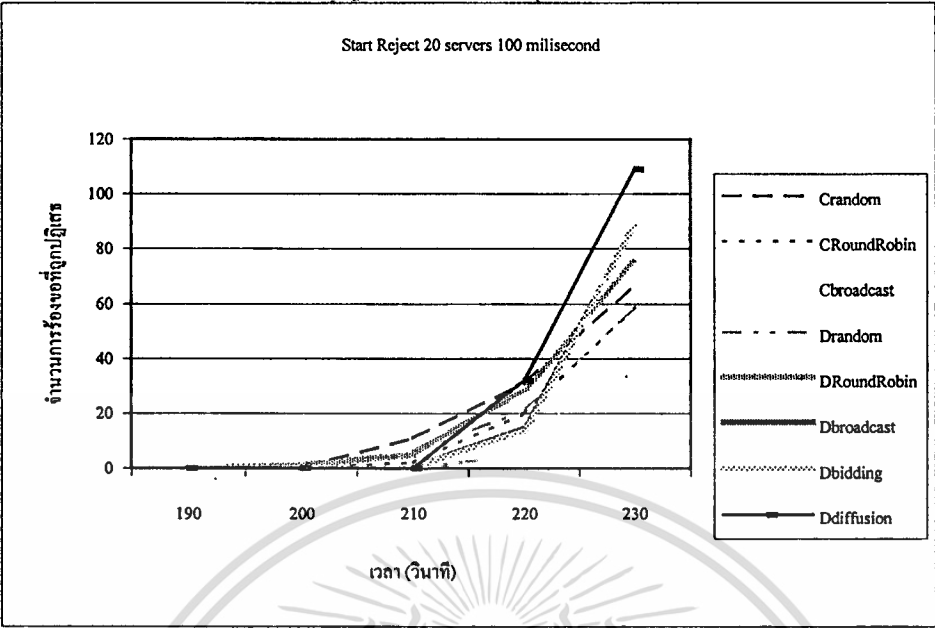
##### ผลการจำลองการทำงาน

Reject 20 servers 100 milisecond

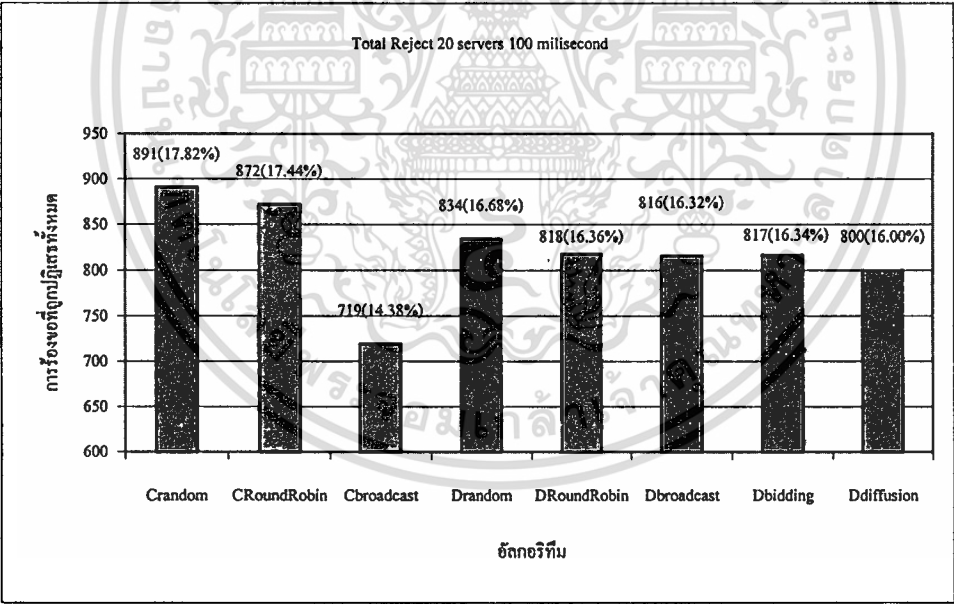


รูปที่ 4.9 กราฟแสดงจำนวนการร้องขอที่ถูกปฏิเสธในแต่ละอัลกอริทึมของแต่ละสถาปัตยกรรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.10 กราฟแสดงการร้องขอแรกที่ถูกปฏิเสธ



รูปที่ 4.11 กราฟแสดงการร้องขอทั้งหมดที่ถูกปฏิเสธ

วิเคราะห์ผลการจำลองการทำงาน

จากการจำลองการทำงาน (รูปที่ 4.9-รูปที่ 4.11) แสดงให้เห็นว่า โดยทั่วไปแล้ว สถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลางสามารถกระจายภาระงานได้ดี แสดงจากจำนวนการร้องขอที่ถูกปฏิเสธมีจำนวนน้อยกว่าสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ (รูปที่ 4.12) นอกจากนี้ยังเป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.11) แต่สถาปัตยกรรมที่มีตัวแบ่งภาระงานประจำอยู่ที่เซิร์ฟเวอร์ระบบจะเริ่มปฏิเสธการร้องขอช้ากว่า (รูปที่ 4.10) สถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลางทั้งนี้เนื่องจากการถ่ายโอนการร้องขอไปให้กับเซิร์ฟเวอร์ส่งผลให้โอกาสที่การร้องขอจะถูกปฏิเสธมีน้อยกว่า

ในส่วนของอัลกอริทึมการแบ่งภาระงานนั้น ถ้าเป็นอัลกอริทึมที่มีสถาปัตยกรรมเหมือนกัน จะมีค่าของการร้องขอแรกที่ถูกปฏิเสธและจำนวนการร้องขอที่ถูกปฏิเสธทั้งหมดแตกต่างกัน ในสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง อัลกอริทึมที่สามารถแบ่งภาระงานได้ดีคือ อัลกอริทึม State Change Broadcast เนื่องจากเป็นอัลกอริทึมที่มีการใช้ข้อมูลภาระงานในเซิร์ฟเวอร์แต่ละเครื่องมาประกอบการตัดสินใจในการกำหนดเซิร์ฟเวอร์เพื่อให้บริการ ส่งผลให้การร้องขอแรกที่ถูกปฏิเสธเกิดขึ้นช้ากว่า และจำนวนการร้องขอที่ถูกปฏิเสธทั้งหมดคือน้อยกว่าอัลกอริทึม Random และ Round-Robin สำหรับสถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์นั้น อัลกอริทึมที่สามารถแบ่งภาระงานได้ดีที่สุดคือ อัลกอริทึม State Change Broadcast และ Bidding โดยพิจารณาจากการร้องขอแรกที่ถูกปฏิเสธและจำนวนการร้องขอทั้งหมดที่ถูกปฏิเสธ ทั้งนี้เนื่องจากอัลกอริทึมทั้งสองมีการใช้ข้อมูลภาระงานประกอบการตัดสินใจเพื่อถ่ายโอนการร้องขอ ส่วนอัลกอริทึม Diffusion นั้นถึงแม้ว่าจะไม่มีการใช้ข้อมูลภาระงานในการตัดสินใจแต่สามารถแบ่งภาระงานได้ดีโดยสังเกตได้จากการร้องขอแรกที่ถูกปฏิเสธนั้นจะใกล้เคียงกับอัลกอริทึม State Change Broadcast เนื่องจากอัลกอริทึมจะทำการถ่ายโอนการร้องขอไปเรื่อยๆ จนกว่าจะถึงเซิร์ฟเวอร์ที่ให้บริการได้

## 4.8 ผลการจำลองการทำงานที่ 5

### จุดประสงค์

- เพื่อเปรียบเทียบเวลาในการเริ่มต้นให้บริการของสถาปัตยกรรม

### ตัวแปรควบคุม

- จำนวนเซิร์ฟเวอร์ 20 เครื่อง โดยเซิร์ฟเวอร์แต่ละเครื่องมีค่าหน่วยเวลาของดิสก์เท่ากับ 7 มิลลิวินาที และมีแบนด์วิดธ์ของดิสก์ 10 เมกะไบต์ต่อวินาที
- อัลกอริทึม State Change Broadcast
- อัตราการตัดสินใจของตัวแบ่งภาระงานที่ศูนย์กลาง 10 การร้องขอต่อวินาที

### ตัวแปรที่พิจารณา

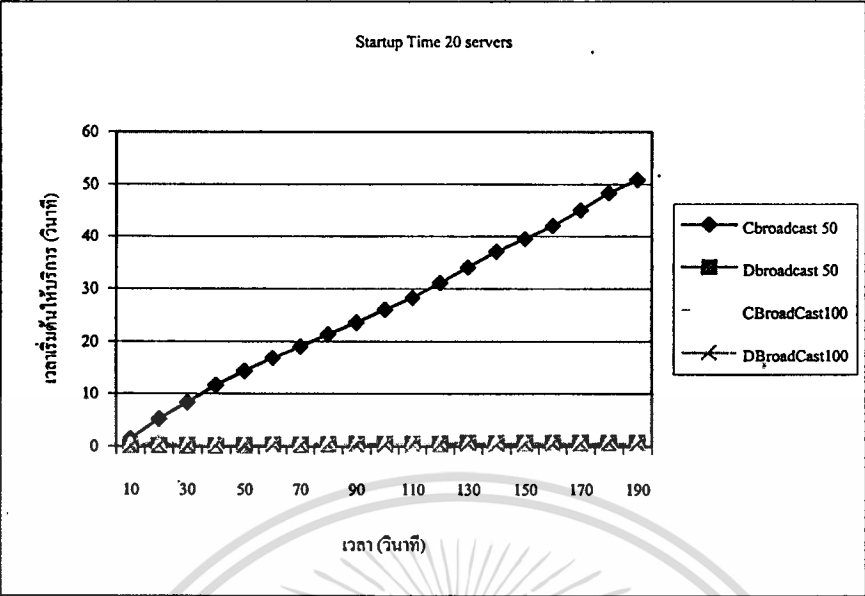
- สถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลางและที่กระจายอยู่ที่เซิร์ฟเวอร์
- คาบเวลาเฉลี่ยระหว่างการร้องขอ 50 และ 100 วินาที

### มาตรวัด

- เวลาเริ่มต้นให้บริการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลการจำลองการทำงาน



รูปที่ 4.12 กราฟแสดงเวลาเริ่มต้นให้บริการของอัลกอริทึม State Change Broadcast ของสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง และสถาปัตยกรรมที่มีตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ ภายใต้อัตราการร้องขอที่มีคาบเวลาระหว่างการร้องขอเฉลี่ย 50 และ 100 มิลลิวินาที

วิเคราะห์ผลการจำลองการทำงาน

จากผลการจำลองการทำงาน (รูปที่ 4.12) แสดงให้เห็นว่าในกรณีที่อัตราการร้องขอต่ำกว่าอัตราการตัดสินใจของตัวแบ่งภาระงานที่ศูนย์กลาง (กราฟ 2 เส้นล่าง) เวลาเริ่มต้นให้บริการของทั้งสองสถาปัตยกรรมจะไม่แตกต่างกันมากนัก แต่ถ้าอัตราการร้องขอสูงกว่าอัตราการตัดสินใจของตัวแบ่งภาระงานที่ศูนย์กลางแล้ว (เส้น Cbroadcast 50) เวลาในการเริ่มต้นให้บริการของสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลางจะสูงกว่ามากและจะเพิ่มขึ้นอย่างรวดเร็ว ทั้งนี้เนื่องจากการร้องขอเสียเวลาอยู่ในคิว ดังนั้นถ้าสามารถประมาณอัตราการร้องขอที่จะมีเข้ามาในระบบได้ ก็ควรพิจารณาว่าอัตราการร้องขอสูงกว่าอัตราการตัดสินใจของตัวแบ่งภาระงานที่ศูนย์กลางหรือไม่ เพราะถ้ามากกว่า สถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์จะเป็นทางเลือกที่เหมาะสมกว่าในแง่ของเวลาเริ่มต้นให้บริการ

4.9 ผลการจำลองการทำงานที่ 6

จุดประสงค์

- เพื่อประเมินการประมาณความสามารถในการให้บริการของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

### ตัวแปรควบคุม

- จำนวนการร้องขอทั้งหมด 5,000 การร้องขอ
- อัลกอริทึม State Change Broadcast ในสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

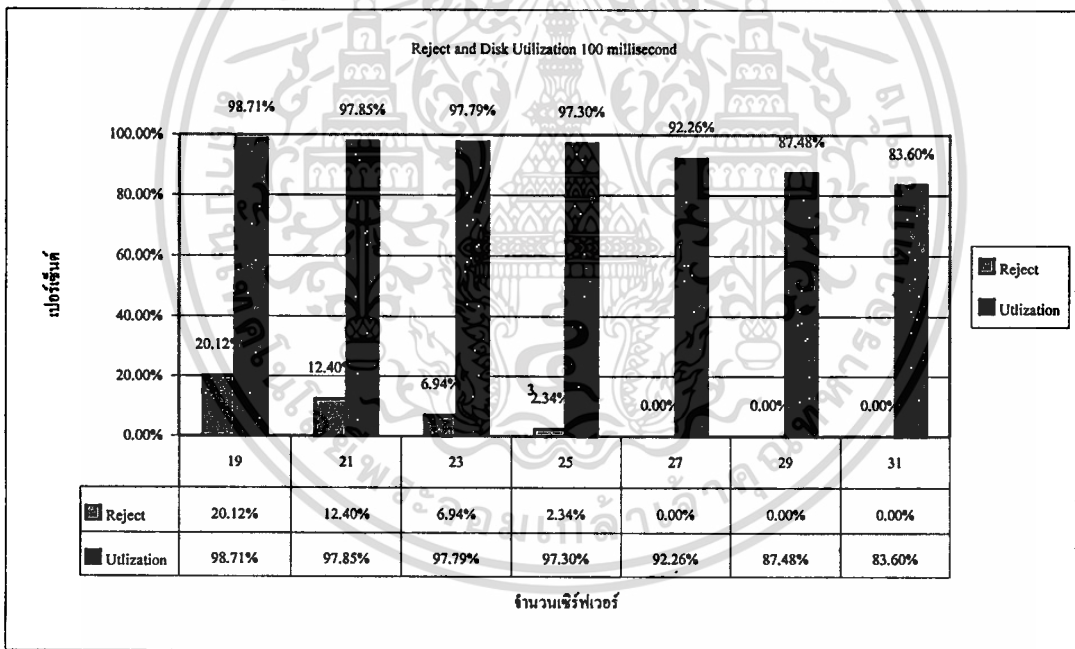
### ตัวแปรที่พิจารณา

- คาบเวลาเฉลี่ยระหว่างการร้องขอ 50 และ 100 มิลลิวินาที
- จำนวนเซิร์ฟเวอร์
- แบนด์วิธของดิสก์ 10 และ 40 เมกะไบต์ต่อวินาที

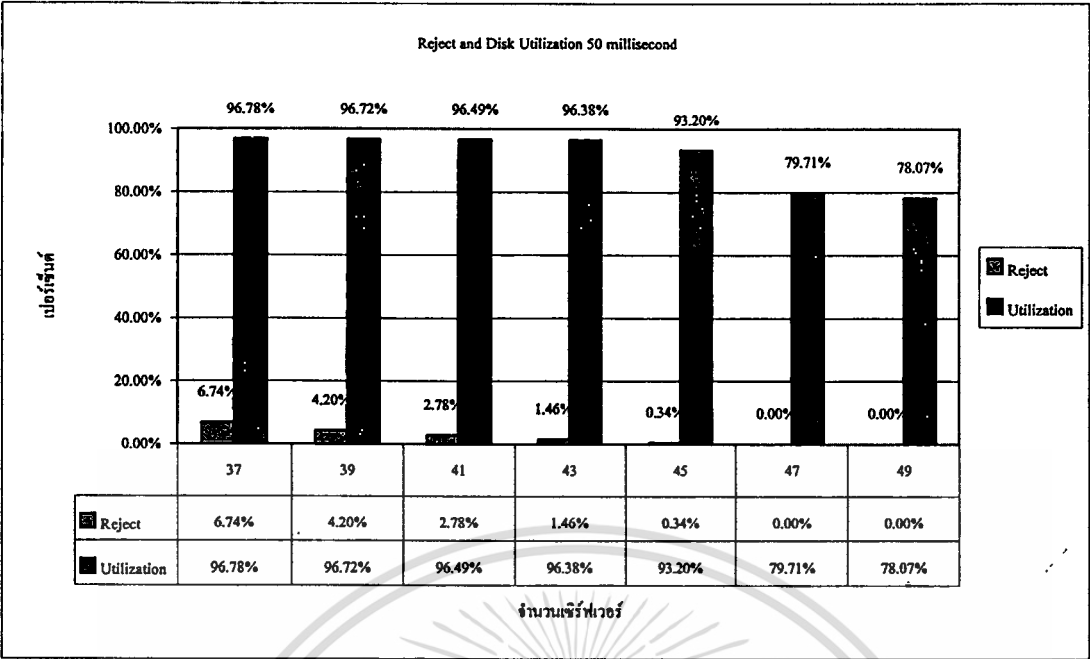
### มาตรวัด

- อัตราการร้องขอที่ถูกปฏิเสธ
- ค่า Disk Utilization

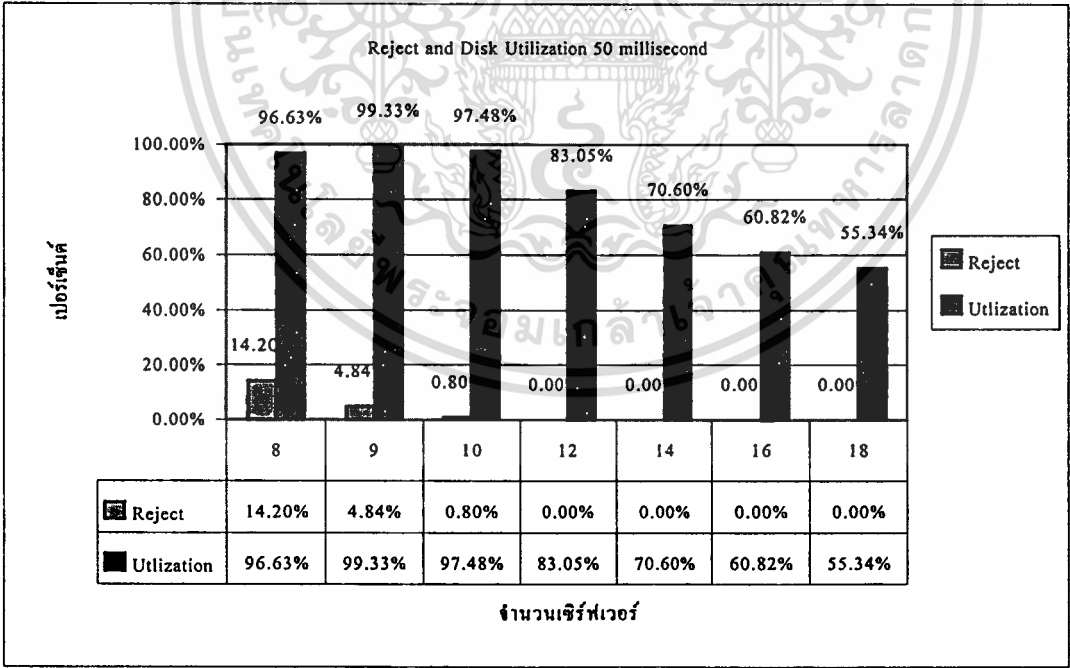
### ผลการจำลองการทำงาน



รูปที่ 4.13 กราฟแสดงอัตราการร้องขอที่ถูกปฏิเสธ และ Disk Utilization เมื่อคาบเวลาเฉลี่ยระหว่างการร้องเป็น 100 มิลลิวินาที เมื่อแบนด์วิธของดิสก์เท่ากับ 10 เมกะไบต์ต่อวินาที



รูปที่ 4.14 กราฟแสดงอัตราการร้องขอที่ถูกปฏิเสธ และ Disk Utilization เมื่อคาบเวลาเฉลี่ยระหว่างการร้องเป็น 50 มิลลิวินาที เมื่อแบนด์วิธของดิสก์เท่ากับ 10 เมกะไบต์ต่อวินาที



รูปที่ 4.15 กราฟแสดงอัตราการร้องขอที่ถูกปฏิเสธ และ Disk Utilization เมื่อคาบเวลาเฉลี่ยระหว่างการร้องเป็น 50 มิลลิวินาที เมื่อแบนด์วิธของดิสก์เท่ากับ 40 เมกะไบต์ต่อวินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## วิเคราะห์ผลการจำลองการทำงาน

จากโมเดลของระบบ ทำให้เราสามารถคาดเดาลักษณะของระบบเช่นอัตราการร้องขอที่ถูกปฏิเสธ หรือค่า Disk Utilization เมื่อมีการกำหนดจำนวนและแบนด์วิธของดิสก์ของเซิร์ฟเวอร์ ดังตัวอย่างเช่น ถ้าต้องการให้ระบบสามารถรองรับอัตราการร้องขอ 10 การร้องขอต่อวินาที (คาบเวลาเฉลี่ยระหว่างการร้องขอ 100 มิลลิวินาที) โดยสื่อในระบบที่มีความยาวเฉลี่ยประมาณ 260 วินาทีแล้ว ถ้าระบบมีเซิร์ฟเวอร์จำนวน 27 เครื่อง ก็จะสามารถรองรับการร้องขอได้ทั้งหมดโดยที่ไม่มีการปฏิเสธเลย (รูปที่ 4.13) หรือถ้ากำหนดให้ระบบสามารถมีอัตราการปฏิเสธการร้องขอไม่เกิน 20 เปอร์เซ็นต์ การติดตั้งเซิร์ฟเวอร์จำนวน 19 เครื่องก็เพียงพอ

หรือในกรณีที่ระบบจะต้องรองรับกับอัตราการร้องขอที่เพิ่มขึ้น เป็น 20 การร้องขอต่อวินาที (คาบเวลาเฉลี่ยระหว่างการร้องขอ 50 มิลลิวินาที) ถ้าระบบมีเซิร์ฟเวอร์จำนวน 47 เครื่อง ระบบก็จะไม่ปฏิเสธการร้องขอเลย (รูปที่ 4.14) และเช่นเดียวกัน ถ้าระบบเพิ่มความแบนด์วิธของดิสก์จาก 10 เมกะไบต์ต่อวินาที เป็น 40 เมกะไบต์ต่อวินาที ระบบที่ติดตั้งเซิร์ฟเวอร์ 12 เครื่องก็จะไม่ปฏิเสธการร้องขอเลย (รูปที่ 4.15)

## 4.10 ผลการจำลองการทำงานที่ 7

### จุดประสงค์

- เพื่อประเมินความสามารถในการแบ่งภาระงานในสถานะที่สื่อมีความยาว

### ตัวแปรควบคุม

- จำนวนเซิร์ฟเวอร์ 20 เครื่อง โดยเซิร์ฟเวอร์แต่ละเครื่องมีกำหนดเวลาของดิสก์เท่ากับ 7 มิลลิวินาที และมีแบนด์วิธของดิสก์ 10 เมกะไบต์ต่อวินาที
- คาบเวลาเฉลี่ยระหว่างการร้องขอ 100 มิลลิวินาที
- อัลกอริทึม State Change Broadcast
- ความยาวของสื่อประมาณ 30 นาที

### ตัวแปรที่พิจารณา

- สถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลาง กับที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์

### มาตรวัด

- จำนวนการร้องขอที่ถูกปฏิเสธ

### ผลการจำลองการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

# ឥរូប

งานวิจัยนี้ได้นำเสนอการศึกษา วิเคราะห์และประเมินผลสถาปัตยกรรมและอัลกอริทึมสำหรับการแบ่งภาระงานของเซิร์ฟเวอร์ในการประมวลผลทั่วไป เมื่อนำมาประยุกต์ใช้กับเซิร์ฟเวอร์ที่ให้บริการต่อเนื่อง โดยสถาปัตยกรรมที่นำมาวิเคราะห์นั้นคือสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง (Centralized Load Balancing Architecture) และสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง (Distributed Load Balancing Architecture) ส่วนอัลกอริทึมที่นำมาวิเคราะห์นั้นสามารถแบ่งได้ 2 ประเภทก็คืออัลกอริทึมที่ไม่มีการใช้ข้อมูลของภาระงานซึ่งได้แก่อัลกอริทึม Random, Round-Robin และ Diffusion และอัลกอริทึมที่ใช้ข้อมูลของภาระงานประกอบการทำงาน ได้แก่อัลกอริทึม State Change Broadcast และ Bidding

สำหรับสถาปัตยกรรมแบบที่มีตัวแบ่งภาระงานที่ศูนย์กลางนั้นจากการจำลองการทำงานแสดงให้เห็นว่าความสามารถในการแบ่งภาระงานให้เท่าเทียมนั้นสามารถทำได้ดีกว่าสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่อง ทั้งนี้เนื่องจากตัวแบ่งภาระงานที่ศูนย์กลางนั้นจะมองเห็นภาพรวมของเซิร์ฟเวอร์ในระบบทั้งหมด ส่งผลให้การจัดการต่อการร้องขอจากไคลเอนต์ที่มีเข้ามาสามารถทำได้ง่ายกว่า แต่สถาปัตยกรรมแบบนี้ก็มีข้อเสียก็คือ การร้องขอทั้งหมดจะต้องถูกส่งมาที่ตัวแบ่งภาระงานที่ศูนย์กลางก่อนซึ่งการร้องขอจะต้องถูกส่งมาเข้าคิวเพื่อกระจายไปให้กับเซิร์ฟเวอร์ที่มีอยู่ภายในระบบ ซึ่งในสถานะที่อัตราการร้องขอสูงกว่าอัตราการตัดสินใจของตัวแบ่งภาระงานที่ศูนย์กลาง การร้องขอที่มีเข้ามาก็จะต้องถูกพักอยู่ในคิวเป็นเวลานานก่อนที่จะถูกส่งไปให้กับเซิร์ฟเวอร์เพื่อให้บริการ ซึ่งจะมีผลกระทบโดยตรงต่อเวลาเริ่มต้นการให้บริการ

สำหรับสถาปัตยกรรมที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ทุกเครื่องนั้น ถึงแม้ว่าจะมีความสามารถในการแบ่งภาระงานได้ไม่ดีเท่ากับสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลางก็ตาม แต่สถาปัตยกรรมแบบนี้สามารถช่วยเพิ่มโอกาสให้กับการร้องขอที่ถูกปฏิเสธในการถ่ายโอนไปให้กับเซิร์ฟเวอร์เครื่องอื่น ดังจะสังเกตได้จากการร้องขอแรกที่ถูกปฏิเสธจะเกิดขึ้นช้ากว่า

สถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง และข้อดีที่สำคัญของสถาปัตยกรรมแบบนี้ก็คือ สามารถช่วยแก้ปัญหาการเข้าคิวในกรณีที่อัตราการร้องขอสูงกว่าอัตราการตัดสินใจของตัวแบ่งภาระงานที่ศูนย์กลาง ซึ่งจะช่วยให้เวลาในการเริ่มต้นให้บริการน้อยกว่าสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง

สำหรับอัลกอริทึมในการแบ่งภาระงาน Ramdon และ Round-Robin ข้อดีของ 2 อัลกอริทึมนี้ก็คือ ตัวแบ่งภาระงานสามารถทำงานได้ด้วยความเร็ว เนื่องจากไม่ต้องอาศัยและคอยข้อมูลสถานะภาระงานของเซิร์ฟเวอร์ในระบบประกอบการตัดสินใจ แต่ข้อเสียของอัลกอริทึมแบบนี้ก็คือ การแบ่งภาระงานทำได้ไม่คั่นก่ สก่เกิดได้จากจำนวนการร้องขอที่ถูกปฏิเสธซึ่งมีจำนวนมากกว่า ทั้งนี้เนื่องจากการร้องขออาจจะถูกส่งไปยังเซิร์ฟเวอร์ที่ให้บริการจนไม่มีทรัพยากรเหลือพอ ซึ่งก็ทำให้การร้องขอนั้นถูกปฏิเสธไปในขณะที่อาจจะยังมีเซิร์ฟเวอร์เครื่องอื่นในระบบที่สามารถรองรับการร้องขอนั้นได้ก็ตาม

สำหรับอัลกอริทึมที่มีการใช้ข้อมูลประกอบการแบ่งภาระงานนั้นสถาปัตยกรรมที่มีตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์ อัลกอริทึมที่มีการใช้ข้อมูลประกอบการแบ่งภาระงานนี้สามารถช่วยเพิ่มความสามารถในการแบ่งภาระงานให้มีความเท่าเทียม ซึ่งสก่เกิดได้จากจำนวนการร้องขอที่ถูกปฏิเสธซึ่งมีจำนวนน้อยกว่ามาก

อย่างไรก็ตาม สำหรับอัลกอริทึม Diffusion ซึ่งเป็นอัลกอริทึมที่ไม่ได้ใช้ข้อมูลประกอบการตัดสินใจเพื่อแบ่งภาระงานแต่ก็มีความสามารถในการแบ่งภาระงานได้ดี สก่เกิดได้จากการร้องขอแรกที่ถูกปฏิเสธนั้นใกล้เคียงกับอัลกอริทึม State Change Broadcast และ Bidding ทั้งนี้เพราะอัลกอริทึมจะทำการถ่ายโอนภาระงานไปให้กับเซิร์ฟเวอร์เครื่องอื่นจนกว่าจะได้รับบริการ แต่อัลกอริทึมนี้มีข้อเสียก็คือ จำเป็นต้องมีการถ่ายโอนภาระงานไปจนกว่าจะมีเซิร์ฟเวอร์ที่สามารถให้บริการได้ ซึ่งจะทำให้เวลาในการเริ่มต้นให้บริการสูงกว่า

สำหรับสถาปัตยกรรมที่เหมาะสมสำหรับระบบที่ให้บริการต่อเนื่องนั้นก็คือสถาปัตยกรรมที่มีตัวแบ่งภาระงานที่ศูนย์กลาง โดยเฉพาะอย่างยิ่งอัลกอริทึม State Change Broadcast เนื่องจากเป็นอัลกอริทึมที่ช่วยให้ระบบปฏิเสธการร้องขอน้อยที่สุด อย่างไรก็ตามก่อนที่จะเลือกสถาปัตยกรรมก็ควรประมาณอัตราการร้องขอของทั้งระบบเสียก่อน ทั้งนี้เนื่องจากถ้าอัตราการร้องขอมากกว่าอัตราการตัดสินใจของตัวแบ่งภาระงานที่ศูนย์กลางแล้ว เวลาเริ่มต้นให้บริการจะสูงมาก ในกรณีนี้ถ้าไม่เพิ่มจำนวนเซิร์ฟเวอร์สถาปัตยกรรมแบบที่ตัวแบ่งภาระงานกระจายอยู่ที่เซิร์ฟเวอร์จะเหมาะสมกว่า

จากการจำลองการทำงานเพื่อเก็บผลแสดงให้เห็นว่าการใช้โปรแกรมจำลองการทำงานเพื่อประเมินสถาปัตยกรรม และอัลกอริทึมในการแบ่งภาระงานสามารถช่วยในการประมาณและคาดเดาคุณสมบัติของระบบที่ให้บริการต่อเนื่องไม่ว่าจะเป็นจำนวนการร้องขอที่ถูกปฏิเสธ และเวลาเริ่มต้นในการให้บริการ ซึ่งเป็นประโยชน์อย่างมากในการออกแบบ เลือกใช้สถาปัตยกรรมและเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อัลกอริทึมในการแบ่งภาระงานสำหรับระบบที่ให้บริการสื่อแบบต่อเนื่องในสภาวะแวดล้อมที่แตกต่างกันไป

## 5.2 ข้อเสนอแนะ

### 5.2.1 เกี่ยวกับงานวิจัย

ข้อเสนอแนะสำหรับการทำงานวิจัยในอนาคตนั้น ประกอบด้วย การพิจารณาสถาปัตยกรรมและอัลกอริทึมในการแบ่งภาระงานสำหรับระบบที่มีกลุ่มของเซิร์ฟเวอร์กระจายอยู่ตามพื้นที่ต่างๆ การพิจารณาการเก็บและวางสื่อสำหรับการแบ่งภาระงาน การพิจารณาผลของแคชและพร็อกซี (Proxy) ที่เกี่ยวข้อง รวมทั้งการพิจารณาถึงระบบเครือข่ายที่ใช้ด้วย

เนื่องด้วยในปัจจุบันการให้บริการข้อมูลสื่อผสมนั้น ถึงแม้ว่าจะมีการนำเอาสถาปัตยกรรมแบบคลัสเตอร์ (Cluster) เข้ามาใช้งาน แต่แนวโน้มของสถาปัตยกรรมต่อไปนั้นก็คือระบบสำหรับให้บริการข้อมูลจะมีการกระจายไปตามพื้นที่ต่างๆ ตามแนวความคิดของการนำข้อมูลไปให้อยู่ใกล้กับผู้ใช้งานที่สุด

สำหรับการวางสื่อ่นั้น ในงานวิจัยนี้ได้พิจารณาเฉพาะเซิร์ฟเวอร์ที่มีสื่อเก็บเหมือนกันทุกเครื่อง ซึ่งก็มีข้อเสียคือมีความซ้ำซ้อนของสื่อ ซึ่งอาจจะเป็นการเปลืองเนื้อที่ในการเก็บข้อมูล งานวิจัยในอนาคตควรจะพิจารณาดำเนิน และจำนวนของสื่อที่เก็บเพื่อให้สอดคล้องกับการแบ่งภาระงานของระบบ

สำหรับแคชและพร็อกซีนั้นสามารถช่วยให้ผู้ที่ขอเข้าใช้สื่อได้รับประโยชน์เนื่องจากทำให้ผู้ที่ขอเข้าใช้สื่อได้รับสื่อด้วยเวลาที่รวดเร็วขึ้น และยังช่วยลดแบนด์วิดธ์ของระบบเครือข่ายที่ใช้ลงด้วย ดังนั้นการศึกษาและวิจัยระบบแคชและพร็อกซีที่สอดคล้องกับการแบ่งภาระงานสำหรับสื่อแบบต่อเนื่องจึงเป็นสิ่งที่น่าเป็นประโยชน์

ในงานวิจัยนี้ไม่ได้พิจารณาผลของระบบเครือข่ายที่มีต่อการให้บริการสื่อแบบต่อเนื่อง ซึ่งในความเป็นจริงแล้ว ระบบเครือข่ายเป็นปัจจัยที่มีผลไม่น้อย งานวิจัยที่ควรจะได้รับการทำงานต่อไปในอนาคตก็ได้แก่การวิเคราะห์ระบบเครือข่าย รวมทั้งการออกแบบพัฒนาระบบเครือข่ายที่เหมาะสมสำหรับสถาปัตยกรรมและอัลกอริทึมการแบ่งภาระงานของสื่อแบบต่อเนื่องด้วย

### 5.2.2 เกี่ยวกับการนำไปใช้งานจริง

สถาปัตยกรรมและอัลกอริทึมสำหรับแบ่งภาระงานในงานวิจัยนี้สามารถนำไปประยุกต์ใช้งานได้กับการให้บริการจริง โดยเฉพาะอย่างยิ่งบริการรูปแบบใหม่ที่มีชื่อว่า Content Delivery Network (CDN) ซึ่งไม่เป็นเพียงแต่การให้เซิร์ฟเวอร์เท่านั้นที่ทำการแบ่งภาระงาน แต่ใน CDN นั้นเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ที่ของการแบ่งภาระงานจะเกิดขึ้นทั้งที่ระบบเครือข่าย แคลช และตัวเซิร์ฟเวอร์เอง ตัวอย่างของการให้บริการ CDN นั้นก็ได้แก่ บริการจากบริษัท Alkamai [48] และ Inktomi [49] ด้วยการทำงานร่วมกันของระบบทั้งหมดในการแบ่งภาระงานส่งผลให้ผู้ใช้และผู้ให้บริการได้รับประโยชน์สูงสุด ในแง่ของผู้ใช้นั้นจะได้รับประโยชน์จากการได้รับบริการที่รวดเร็ว และใช้เวลาในการรอคอยข้อมูลน้อยลง ส่วนผู้ให้บริการก็จะได้รับประโยชน์จากแบนด์วิดธ์ของระบบเครือข่ายที่ใช้งานน้อยลงด้วยระบบ CDN นั้นไม่เพียงแต่จะให้บริการข้อมูลสแตติกเท่านั้น แต่ยังสามารถให้บริการข้อมูลไดนามิกสำหรับการทำธุรกิจบนอินเทอร์เน็ต และการให้บริการสื่อผสม หรือสื่อแบบต่อเนื่องอีกด้วย สำหรับข้อมูลเพิ่มเติมของระบบ CDN นั้นสามารถหาได้จากเว็บไซต์ของบริษัท Cisco [50] และเว็บไซต์เฉพาะกิจที่ได้รับการจัดตั้งขึ้นมาสำหรับ CDN โดยเฉพาะ [51]



## เอกสารอ้างอิง

- [1] Zhang, Y.; Kameda, H.; Shimizu, K. , “Adaptive bidding load balancing algorithms in heterogeneous distributed systems”, Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1994., MASCOTS '94., Proceedings of the Second International Workshop on , 1994 , Page(s): 250 –254.
- [2] Luque, E.; Ripoll, A.; Cortes, A.; Margalef, T. ; “A distributed diffusion method for dynamic load balancing on parallel computers”, Parallel and Distributed Processing, 1995. Proceedings. Euromicro Workshop on , 1995 , Page(s): 43 –50.
- [3] Guojun Lu. **Communication and Computing for Distributed Multimedia Systems** Artech House, Boston, London 1996.
- [4] A. Vogel et al., “Distributed Multimedia and QOS: a Survey”, IEEE Multimedia, vol. 2, no.2, pp. 10-18, Summer 1995.
- [5] K. Nahrstedt and R. Steinmetz, “Resource Management in Networked Multimedia Systems”, Computer, pp. 52-63, May 1995.
- [6] A. Campbell, et al., “A Quality of Service Architecture”, Computer Communication Review, vol. 24, no.2, pp. 6-27, 1994.
- [7] Ketan Mayer-Patel, David Simpson, David Wu and Lawrence A. Rowe, “Synchronized Continuous Media Playback Through the World Wide Web”, Proceedings of the fourth ACM international conference on Multimedia, 1996, Pages 435-436.
- [8] Buck, R., "The Oracle media server for nCUBE massively parallel systems", Parallel Processing Symposium, 1994. Proceedings., Eighth International , 1994 , Page (s): 670 –673.
- [9] H. Vin, P. Goyal and A. Goyal; “A statistical admission control algorithm for multimedia servers”; Proceedings of the second ACM international conference on Multimedia, 1994, Pages 33 – 40.
- [10] Ing-Ray Chen and Tao-Hung Hsi; “Threshold-based dynamic admission control algorithms for real-time multimedia servers”; Official program of the 1996 ACM symposium on Applied Computing, 1996, Pages 224 – 229.
- [11] Brubeck, D.W.; Rowe, L.A.; “Hierarchical storage management in a distributed VOD system”; IEEE Multimedia Volume: 3 3 , Fall 1996 , Page(s): 37 –47.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

[12] Jitendra Padhye, Jim Kurose, Don Towsley and Rajeev Koodli, "A TCP-Friendly Rate Adjustment Protocol for Continuous Media Flows over Best Effort Networks", Proceedings of the international conference on Measurement and Modeling of computer systems, 1999, Pages 220-221.

[13] Kasiolas, A.; Nait-Abdesselam, F.; Makrakis, D. "Cooperative adaptation to quality of service using distributed agents ", Parallel Processing, 1999. Proceedings. 1999 International Workshops on , 1999 , Page(s): 502 –507.

[14] Catrava, S., "High quality desktop video conferencing using agent based systems", Computers and Communications, 1996., Conference Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on , 1996 , Page(s): 37 –43.

[15] Erciyes, K.; Yilmaz, S., "Dynamic Load Balancing In A Distributed Computer System", Electrotechnical Conference, 1991. Proceedings., 6th Mediterranean , 1991 , Page(s): 1077 -1080 vol.2

[16] Weiping Zhu, "Dynamic Load Balancing on Amoeba", Algorithms and Architectures for Parallel Processing, 1995. ICAPP 95. IEEE First ICA/sup3/PP., IEEE First International Conference on Volume:1, 1995, Page(s) 355-364 vol.1

[17] R.M. Bryant and R.A. Finkel. "A Stable Distributed Scheduling Algorithm.", Proceedings of the 2<sup>nd</sup> International Conference on Distributed Computing Systems, April 1981

[18] Hye-Seon Maeng, Hyoun-Su Lee, Tack-Don Han, Sung-Bong Yang, Shin-Dug Kim, "Dynamic Load Balancing of Iterative Data Parallel Problems on a Workstation Clustering", High Performance Computing on the Information Superhighway, 1997 HPC Asia'97, 1997, Page(s): 563-567.

[19] Dugki Min, Choi E., Donghoon Lee, Byungseok Park, "A Load Balancing Algorithm For a Distributed Multimedia Game Server Architecture", Multimedia Computing and Systems, 1999. IEEE International Conference on Volume: 2, 1999, Page(s) 882-886 vol. 2

[20] Cheng T.-Y., Chung J.-Y., Lin K.-J., "Dynamic Load Balancing Algorithms in Loosely-Coupled Real-Time Systems", Computer Software and Applications Conference, 1992. COMPSAC'92. Proceedings, Sixteenth Annual International, 1992, Page(s): 143-148.

[21] Lin, F.C.H., and Keller, R.M., "The Gradient Model local balancing method", IEEE Transactions on Software Engineering, Vol SE-13, No. 1, pp.32-38, Jan. 1987.

[22] Zhao, W., and Ramamritham, K., "Distributed scheduling using bidding and focussed addressing", Proc. Symp. Real-Time Systems, Dec. 1985.

เอกสารนี้เป็นเอกสารทบทวนวิจัยสำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

[23] Joshi B.S., Hosseini S.H., Vairaran K., "A Methodology for Evaluating Load Balancing Algorithms", High Performance Distributed Computing, 1993, Proceedings the 2<sup>nd</sup> International Symposium on, 1993, Page(s): 216-222.

[24] Cardellini V., Colajanni M., Yu P.S., "Dynamic Load Balancing in Geographically Distributed Heterogeneous Web Servers", IEEE Internet Computing Volume: 33, May-June 1999, Page(s): 28-39.

[25] Gil-Haeng Lee, "Issues of the State Information for Location and Information Policies in Distributed Load Balancing Algorithm", EUROMICRO Conference, 1999. Proceedings 25<sup>th</sup> volume: 1, 1999, Page(s): 67-70 vol. 1.

[26] N.G. Shivāratri and M. Singhal, "A Transfer Policy for Global Scheduling Algorithms to Schedule tasks with Deadline", In Proceedings of the 11<sup>th</sup> International Conference on Distributed Computing Systems, pp.248-255, 1991.

[27] R. Mirchandaney, D. Towsley, and J.A. Stankovic, "Analysis of the effects of Delays on Load Sharing", IEEE Trans. On Computers, vol. 38, no. 11, pp.1513-1525, Nov. 1989.

[28] Benmohammed-Mahieddine K., Dew P.M., Kara M., "A Periodic Symmetrically-Initiated Load Balancing Algorithm for Distributed Systems", Distributed Computing Systems, 1994., Proceedings of the 14<sup>th</sup> International Conference on, 1994, Page(s): 616-623.

[29] D.L. Eager, E. D. Lazowska, and J. Zahorjan, "A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing", Proc. Of the 1985 ACM SIGMETRICS Conference on Measurement and Modelling of Computer Systems, pp. 1-3(August 1985)

[30] R. Mirchandaney, D. Towsley, and J.A. Stankovic, "Analysis of the Effects of Delays on Load Sharing", IEEE Transactions on Computers 38(11) pp. 1513-1525 (November 1989)

[31] Kun-Ming Yu, Wu S.J.-W., Tzung-Pei Hong, "A Load Balancing Algorithm Using Prediction", Parallel Algorithms/Architecture Synthesis, 1997. Proceedings., Second Aizu International Symposium, 1997, Page(s): 159-165.

[32] Yen-Jen Lee, Du D.H.C., "Adaptive Load Sharing and Scheduling Schemes for Distributed Continuous Media Delivery", Multimedia Computing and Systems, 1998. IEEE International Conference on Volume: 2, 1999, Page(s): 506-511 vol. 2.

[33] Shin, K.G.; Chao-Ju Hou , "Design and evaluation of effective load sharing in distributed real-time systems", Parallel and Distributed Systems, IEEE Transactions on Volume: 5 7 , July 1994 , Page(s): 704 -719.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- [34] Shivaratri, N.G.; Krueger, P., "Two adaptive location policies for global scheduling algorithms", Distributed Computing Systems, 1990. Proceedings., 10th International Conference on , 1990 , Page(s): 502 –509.
- [35] JianLiu, Longlu Xu, Baogen Gu, and Jing Zhang; "A Scalable, High Performance Internet Cluster Server", HPC-ASIA2000, The Fourth International Conference/Exhibition on High Performance Computing in Asia-Pacific Region, May 14-17, 2000, Beijing, China, Page (s): 941-944.
- [36] Riedl, R.; Richter, L., "Classification of load distribution algorithms", Parallel and Distributed Processing, 1996. PDP '96. Proceedings of the Fourth Euromicro Workshop on , 1996 , Page(s): 404 –413.
- [37] Govindan, V.; Franklin, M.A.; "Effect of control parameters on dynamic load balancing"; Parallel Processing Symposium, 1994. Proceedings., Eighth International , 1994 , Page(s): 452 –460.
- [38] Casavant, T.L.; Kuhl, J.G.; " A taxonomy of scheduling in general-purpose distributed computing systems"; Software Engineering, IEEE Transactions on Volume: 14 2 , Feb. 1988 , Page(s): 141 –154.
- [39] Krueger, P.; Livny, M.; "A comparison of preemptive and non-preemptive load distributing", Distributed Computing Systems, 1988., 8th International Conference on , 1988 , Page(s): 123 –130.
- [40] Wolfgang Golubski, Dietmar Lammers, Wolfram-M. Lippe Westf; "Theoretical and Empirical Results on Dynamic Load Balancing in an Object-Based Distributed Environment", Proceedings of the 16<sup>th</sup> ICDCS, 1996, Page(s): 537-544.
- [41] Serpanos D.N., Georgiadis L., Bouloutas T., "MMPacking: A Load and Storage Balancing Algorithm for Distributed Multimedia Servers", Circuits and Systems for Video Technology, IEEE Transactions on Volume: 81, Feb. 1998, Page(s): 13-17
- [42] William J. Bolosky, Robert P. Fitzgerald and John R. Douceur; "Distributed schedule management in the tiger video fileserver", Proceedings of the sixteenth ACM symposium on Operating systems principles, 1997, Pages 212 – 223.
- [43] Yen-Jen Lee, Du D.H.C., "Adaptive Load Sharing and Scheduling Schemes for Distributed Continuous Media Delivery", Multimedia Computing and Systems, 1998. IEEE International Conference on Volume: 2, 1999, Page(s): 506-511 vol. 2.

[44] Venkatasubramanian, N.; Ramanathan, S.; "Load management in distributed video servers", Distributed Computing Systems, 1997.; Proceedings of the 17th International Conference on , 1997 , Page(s): 528 –535.

[45] Boutaba, R.; Hafid, A.; "A generic platform for scalable access to multimedia-on-demand systems", Selected Areas in Communications, IEEE Journal on Volume: 17 9 , Sept. 1999 , Page(s): 1599 –1613.

[46] A. Dan, D. Sitaram, and P. Shahabuddin. "Dynamic batching policies for an on-demand video server", Multimedia Systems, 4(3): 112-121, June 1996.

[47] Loeb, M.L.; Rindos, A.J.; Holland, W.G.; Woolet, S.P., "Gigabit Ethernet PCI adapter performance", IEEE Network , Volume: 15 Issue: 2 , March-April 2001, Page(s): 42 –47.

[48] <http://www.alkamai.com/>

[49] <http://www.inktomi.com/>

[50] [http://www.cisco.com/warp/public/779/largeent/learn/technologies/content\\_networking/cdn\\_ent.html](http://www.cisco.com/warp/public/779/largeent/learn/technologies/content_networking/cdn_ent.html)

[51] <http://www.stardust.com/cdnweek/whitepapers/cdnspring01/#Top>

## ประวัติผู้เขียน

ชื่อผู้เขียน

นายคูสิต นิยะโต

วันเดือนปีเกิด

24 ตุลาคม 2521

วุฒิการศึกษาระดับปริญญาตรี

วิศวกรรมศาสตรบัณฑิต สาขาคอมพิวเตอร์ (เกียรตินิยมอันดับ 2)

สถานศึกษาที่สำเร็จการศึกษา

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา

2541



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้