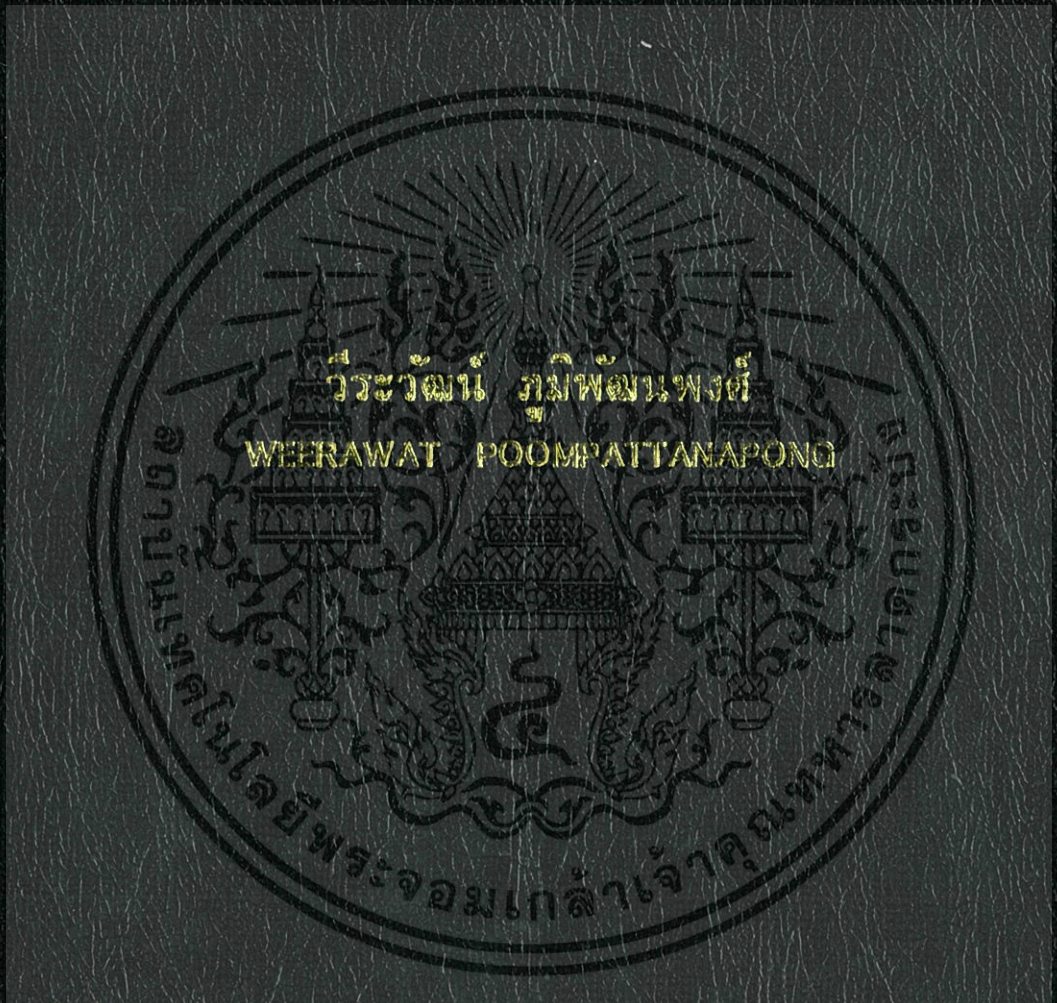
# การค้นหาบริการอย่างรวดเร็วสำหรับระบบประมวลผลแบบกริด

# FAST SERVICES DISCOVERY FOR GRID COMPUTING SYSTEM




วีระวัฒน์ ภูมิพัฒนพงศ์

WEERAWAT POOMPATTANAPONG


วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2547


ISBN 974-15-1219-8

# การค้นหาบริการอย่างรวดเร็วสำหรับระบบประมวลผลแบบกริด

# FAST SERVICES DISCOVERY FOR GRID COMPUTING SYSTEM

วีระวัฒน์ ภูมิพัฒนพงศ์
WEERAWAT POOMPATTANAPONG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2547

ISBN 974-15-1219-8

# FAST SERVICES DISCOVERY FOR GRID COMPUTING SYSTEM

WEERAWAT POOMPATTANAPONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN COMPUTER ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2004

ISBN 974-15-1219-8

| | |
|---|---|
| **หัวข้อวิทยานิพนธ์** | การค้นหาบริการอย่างรวดเร็วสำหรับระบบประมวลผลแบบกริด |
| **นักศึกษา** | นายวีระวัฒน์ ภูมิพัฒนพงศ์ |
| **รหัสประจำตัว** | 45061035 |
| **ปริญญา** | วิศวกรรมศาสตรมหาบัณฑิต |
| **สาขาวิชา** | วิศวกรรมคอมพิวเตอร์ |
| **พ.ศ.** | 2547 |
| **อาจารย์ผู้ควบคุมวิทยานิพนธ์** | รศ.ประทีป บัญญัตินพรัตน์ |
| **อาจารย์ผู้ควบคุมวิทยานิพนธ์ร่วม** | รศ.บรรจง ปิยธำรง |


## บทคัดย่อ

การบริการค้นหาข้อมูลเป็นส่วนการทำงานที่สำคัญส่วนหนึ่งของสถาปัตยกรรมการประมวลผลแบบกริดที่มีรากฐานมาจากสถาปัตยกรรมเชิงบริการ มีหน้าที่ค้นหารีซอร์สที่กระจายตัวอยู่ในระบบเครือข่ายเพื่อที่จะนำมาใช้ทำงานร่วมกัน วิทยานิพนธ์นี้นำเสนอสถาปัตยกรรมการบริการค้นหาข้อมูลแบบใหม่สำหรับการประมวลผลแบบกริด สร้างขึ้นโดยใช้โครงสร้างเอเจนต์ที่เชื่อมต่อกันแบบลำดับชั้นทำหน้าค้นหาข้อมูล และมีการปรับปรุงกระบวนการค้นหาด้วยวิธีการที่แตกต่างกันซึ่งมีรากฐานมาจากโครงสร้างข้อมูล Signature File ซึ่งทำหน้าที่อธิบายข้อมูลในโครงสร้างลำดับชั้น มีจุดประสงค์เพื่อเพิ่มสมรรถนะด้านความเร็วการค้นหา ประสิทธิภาพโดยรวมของระบบ และความสามารถในการร้องรับการใช้บริการได้จำนวนมาก

| | |
|---|---|
| Thesis Title | Fast Services Discovery for Grid Computing System |
| Student | Mr. Weerawat Poompattanapong |
| Student ID | 45061035 |
| Degree | Master of Engineering |
| Programme | Computer Engineering |
| Year | 2004 |
| Thesis Advisor | Assoc.Prof. Pratheep Bunyatnoparat |
| Thesis Co-Advisor | Assoc.Prof. Bunjong Piyatamrong |


## ABSTRACT

Grid Services Discovery is the important part of Grid infrastructure enables the integration of services across distributed resources based on Service-Oriented architecture. This thesis presents a new architecture for service discovery using the hierarchy of agent that has the capability of service discovery. The performance of the agent system can be improved using different combinations of optimization strategies based on the Signature File that is represented the hierarchy of resources. The objectives of the proposed architecture are improving discovery speed, system efficiency and scalability.

# กิตติกรรมประกาศ

คุณความดีอันใดที่บังเกิดจากวิทยานิพนธ์ฉบับนี้ ขอมอบแด่บิดาและมารดาของผู้วิจัย ผู้ที่คอยห่วงใย เข้าใจและให้การสนับสนุนในการศึกษามาโดยตลอด

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงลงได้ด้วยดี โดยได้รับความกรุณาจาก รศ.ประทีป บัญญัตินพรัตน์ อาจารย์ที่ปรึกษา และ รศ. บรรจง ปิยธำรง อาจารย์ที่ปรึกษาร่วม ที่ได้ช่วยเหลือในการให้คำแนะนำ ความรู้ทางทฤษฎีต่าง ๆ ที่ใช้ และชี้แนะแนวทางในการแก้ปัญหาต่าง ๆ อย่างทุ่มเท รวมทั้งฝึกฝนผู้วิจัยให้มีความสามารถในการทำวิจัยและพัฒนาได้อย่างมีประสิทธิภาพ ผู้วิจัยรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณ กรรมการสอบวิทยานิพนธ์ทุกท่านที่ได้กรุณาให้คำแนะนำในทุกๆ เรื่อง ทั้งวิธีแก้ปัญหาที่เกิดขึ้นในการทำวิทยานิพนธ์และมุมมองในเชิงวิศวกรรมอื่น ๆ ซึ่งช่วยให้ผู้วิจัยมีวิสัยทัศน์ที่กว้างไกลขึ้น

ขอขอบคุณ บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้การสนับสนุนการทำวิทยานิพนธ์นี้

ขอบคุณพี่ ๆ เพื่อน ๆ และน้อง ๆ นักศึกษาทุกคนในห้องวิจัย รวมทั้งเพื่อน ๆ หลาย ๆ คนในอินเทอร์เน็ตที่ช่วยเหลือให้คำแนะนำต่าง ๆ และให้กำลังใจแก่ผู้วิจัยตลอดมา

วีระวัฒน์ ภูมิพัฒนพงศ์

# สารบัญ

หน้า

# สารบัญ (ต่อ)

หน้า

# สารบัญ (ต่อ)

หน้า

# สารบัญ (ต่อ)

หน้า

# สารบัญ (ต่อ)

หน้า

# สารบัญตาราง

# สารบัญรูป

# สารบัญรูป (ต่อ)

# สารบัญรูป (ต่อ)

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาและความสำคัญของปัญหา

การประมวลผลที่มีประสิทธิภาพสูงและความสามารถในการใช้ทรัพยากรร่วมกัน มีบทบาทสำคัญอย่างยิ่งต่อการวิจัยทางวิทยาศาสตร์และกระบวนการทางธุรกิจที่เกี่ยวข้องกับข้อมูลจำนวนมหาศาลอยู่ตลอดเวลา เทคโนโลยีการประมวลผลและการใช้ทรัพยากรร่วมกันที่มีประสิทธิภาพสูงผ่านระบบเครือข่ายจึงได้ถือกำเนิดขึ้นภายใต้ชื่อว่า "ระบบการประมวลผลแบบกริด" (Grid Computing System) [1, 2, 3] ภายในโครงสร้างของกริดผู้ใช้สามารถรับบริการในรูปแบบแตกต่างตามแต่จะตกลงกับผู้ให้บริการ เมื่อเป็นสมาชิกแล้ว ผู้ขอรับบริการสามารถเข้าใช้ทรัพยากรภายในระบบรวมอุทิศทรัพยากรให้กับระบบเพื่อบรรลุเป้าหมายของระบบนั้น โครงการ SETI@HOME [4] เป็นตัวอย่างหนึ่งของการประมวลผลแบบกริด ที่จัดการแบ่งข้อมูลให้ผู้เสียสละทั่วโลกประมวลผลระหว่างที่ไม่ได้ใช้รีซอร์สคอมพิวเตอร์ของตน เพื่อวิเคราะห์สัญญาณวิทยุที่รับมากจากอวกาศ โครงการนี้ได้รับความร่วมมืออย่างดีจากผู้ใช้งานเครื่องคอมพิวเตอร์ทั่วโลก

ในขณะเดียวกัน เทคโนโลยีเว็บเซอร์วิส [5, 6] ได้ถือกำเนิดขึ้นจากวงการธุรกิจที่ต้องการความสามารถในการใช้ทรัพยากรร่วมกันทั้งในส่วนของชิ้นส่วนของซอฟท์แวร์ ข้อมูล และ การประมวลผล ปัจจุบันได้มีการนำเอาเทคโนโลยีเว็บเซอร์วิสเข้ามามีบทบาทในการพัฒนาแอพพลิเคชันของกริด โดยเฉพาะอย่างยิ่ง การให้บริการข้อมูลของระบบการประมวลผลแบบกริด

ปัจจุบันเทคโนโลยีของการประมวลผลแบบกริด และ เว็บเซอร์วิส มีแนวโน้มที่จะพัฒนาไปในทิศทางเดียวกัน โดยเฉพาะอย่างยิ่ง กลไกการค้นหาข้อมูลบริการภายในระบบประมวลผลแบบกริด (Grid Services Discovery System)นั้น มีรากฐานอยู่บนสถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture) [7] อันเป็นกลไกเดียวกับที่ใช้ในเทคโนโลยีเว็บเซอร์วิส อย่างไรก็ตาม สถาปัตยกรรมเชิงบริการดังกล่าว ยังมีประเด็นน่าสนใจในเรื่องประสิทธิภาพอยู่ เนื่องจากในแต่ละรอบการค้นหาข้อมูลบริการในระบบ จำเป็นต้องใช้ทรัพยากรที่เกี่ยวข้องจำนวนมาก บางครั้ง มีการใช้งานทรัพยากรฟุ่มเฟือยเกินความจำเป็น ทำให้ประสิทธิภาพของระบบไม่ดีเท่าที่ควร อีกทั้งปัญหาที่เกิดจากการจัดวางโครงสร้างระบบที่มีอยู่ในปัจจุบัน ไม่เอื้อประโยชน์ต่อการค้นหาข้อมูล ทำให้การค้นหาข้อมูลภายในระบบล่าช้าเกินไป

ปัญหาดังกล่าว เป็นประเด็นสำคัญที่ควรหยิบยกมาพิจารณาแก้ไขปรับปรุง เพื่อเพิ่มประสิทธิภาพของระบบการค้นหาข้อมูลบริการให้มีสมรรถนะสูงขึ้น หมายถึง มีกระบวนการค้นหา

ข้อมูลบริการที่รวดเร็วขึ้น โดยที่มีการใช้ทรัพยากรเพื่อเป็นฐานการค้นหาในระดับที่เหมาะสม ไม่มากเกินความจำเป็น อันเป็นประเด็นหลักของวิทยานิพนธ์ชิ้นนี้

## 1.2 จุดประสงค์ของการศึกษา

1. ศึกษาการทำงานของกลไกการค้นหาข้อมูลบริการในระบบประมวลผลแบบกริดที่มีใช้อยู่ในปัจจุบัน เพื่อหาข้อบกพร่อง และปรับปรุงกลไกการค้นหาดังกล่าวให้มีประสิทธิภาพสูงขึ้น

2. นำเสนอกลไกการค้นหาข้อมูลที่มีประสิทธิภาพสูงขึ้น โดยพิจารณาเกณฑ์ด้านความเร็วในการค้นหาข้อมูลและประสิทธิภาพของระบบโดยรวมเป็นหลัก ส่วนของเนื้อที่การจัดเก็บข้อมูลที่กะทัดรัด เหมาะสมกับความเร็วระดับต่างๆ ถือเป็นประเด็นรองลงมา

3. ศึกษาลักษณะการทำงานของการแก้ปัญหา ที่มีผู้คิดค้น ทดลองทำวิจัยมาแล้ว เปรียบเทียบกับ การกลไกการแก้ปัญหาของผู้วิจัยว่ามีข้อดีข้อเสียแตกต่างกันอย่างไรบ้าง ทั้งด้วยวิธีการทดลอง และ ด้วยทฤษฎีทางคณิตศาสตร์

## 1.3 สมมติฐานของการศึกษา

ระบบการค้นหาข้อมูลภายในระบบประมวลผลแบบกริดที่สร้างขึ้น มีความแตกต่างจากระบบเดิมที่ใช้ระบบแบบศูนย์กลาง กล่าวคือ เป็นระบบประมวลผลแบบกระจาย โดยให้ข้อมูลที่อยู่ของบริการกระจายไปตามเอเจนต์ที่กระจายอยู่ตามที่ต่างๆ ผู้ใช้สามารถค้นหาข้อมูลได้เร็วขึ้นจากเอเจนต์ที่อยู่ใกล้ การทำงานของระบบเอเจนต์ที่สร้างขึ้นจะมีสมมติฐานดังต่อไปนี้

1. การส่งข้อมูลเพื่อสอบถามหาข้อมูลภายในระบบ ไม่ควรที่จะค้นหาไปในทุกๆ โหนดของเครื่องที่ให้บริการค้นหาข้อมูล แต่ควรจะสามารถตัดสินใจการส่งข้อมูลเพื่อถามไปยังโหนดที่มีความเป็นไปได้ที่จะมีข้อมูลที่ต้องการอยู่ วิธีการนี้เป็นการลดจำนวนการส่งข้อมูลในระบบทางหนึ่ง

2. ในการค้นหาแต่ละครั้งไม่จำเป็นต้องค้นหาข้อมูลที่มีอยู่ทั้งหมดในระบบ แต่ควรจะค้นหาข้อมูลเป็นจำนวนที่ผู้ใช้งานระบบยอมรับได้ ตัวอย่างเช่น ผู้ใช้งานระบบบางรายต้องการเพียง ข้อมูลที่ชื่อว่า “Printer” ระบบไม่จำเป็นต้องค้นหา “Printer” ทั้งระบบเพียงแต่ ค้นหามาจำนวนหนึ่งอันเป็นที่น่าพอใจแก่ผู้ใช้บริการ ผู้ใช้บริการบางรายต้องการ “Printer” เพียง 1 รายการเท่านั้น

3. เอเจนต์แต่ละตัวที่ทำหน้าที่ประหนึ่งเป็นเครื่องให้บริการข้อมูล แต่ไม่จำเป็นต้องเก็บข้อมูลไว้มากเกินความจำเป็น เพียงแต่เก็บข้อมูลที่มีสาระสำคัญเกี่ยวกับระบบการค้นหาข้อมูลภายในระบบเท่านั้น

4. ผู้ใช้ควรจะค้นหาข้อมูลที่ต้องการได้เร็วกว่า เมื่อค้นหาข้อมูลจากผู้ให้บริการที่อยู่ใกล้กว่า และผู้ใช้ควรค้นหาข้อมูลได้เร็วขึ้นเมื่อเคยค้นหาข้อมูลนั้นแล้ว

## 1.4 หลักการที่นำเสนอใหม่

เพื่อตอบสมมติฐานของการศึกษา วิทยานิพนธ์ชิ้นนี้จึงนำเสนอสถาปัตกรรมการจัดการข้อมูลเพื่อใช้งานในระบบประมวลผลแบบกริด มีรากฐานมาจากระบบประมวลผลแบบกระจายที่ใช้งานเอเจนต์ที่เชื่อมต่อกันหลายทิศทาง (Distributed Multi-Agent System) โครงสร้างของเอเจนต์เชื่อมต่อกันแบบลำดับชั้น (Hierarchical Model) มีจุดประสงค์ให้กลไกการค้นหาข้อมูลบริการของระบบประมวลผลแบบกริดมีความเร็วการค้นหาที่สูงขึ้น ระบบที่สร้างขึ้นจะใช้ประโยชน์จากโครงสร้างข้อมูล Signature File เป็นตัวอธิบายข้อมูลแบบลำดับชั้น และใช้ความรู้ (Knowledge) ของเอเจนต์ที่เชื่อมต่อกัน ร่วมกับการใช้แคช (Cache) เพื่อเพิ่มประสิทธิภาพด้านความเร็วในการค้นหาข้อมูลภายในระบบ เอเจนต์แต่ละตัว จะเก็บข้อมูลเฉพาะส่วนที่จำเป็นต่อการค้นหาไว้ในโครงสร้างข้อมูลที่เหมาะสม โดยมีขนาดพอเหมาะ ไม่สิ้นเปลืองเนื้อที่จัดเก็บมากจนเกินไป

สถาปัตยกรรมที่สร้างขึ้นได้นำเสนอโครงสร้างข้อมูลที่เรียกว่า "Knowledge Posting Services Table" หรือ K-PST ซึ่งพัฒนาโดยอาศัยโครงสร้างข้อมูล Signature เป็นรากฐาน เป็นโครงสร้างข้อมูลที่แสดงถึงข้อมูลที่มีอยู่ในระบบประมวลผลแบบกริด มีการเชื่อมต่อกันแบบลำดับชั้นโดยผูกติดกับเอเจนต์ในระบบ K-PST มีขนาดกะทัดรัด ถูกนำมาใช้เป็นข้อมูลพื้นฐานเพื่อการค้นหาข้อมูล โครงสร้างข้อมูลที่สร้างขึ้นนี้แสดงถึงข้อมูลที่สื่อความหมาย 3 ประการ คือ

1. มีข้อมูลที่เอเจนต์มีหน้าที่ควบคุมอยู่
2. มีข้อมูลที่อยู่ในแคชของเอเจนต์
3. มีความเป็นไปได้ที่จะมีข้อมูลหรือบริการที่ต้องการจะค้นหาอยู่ในเอเจนต์ลำดับที่ต่ำกว่า ทั้งในส่วนที่เป็นข้อมูลบริการที่ครอบครองอยู่เอง หรือ อยู่ในแคชก็ตาม

ความหมายของข้อมูลที่เก็บใน K-PST สามารถนำมาประยุกต์ใช้ในการเพิ่มประสิทธิภาพด้านความเร็วให้แก่ระบบได้ นอกจากนี้ เมื่อนำมาใช้งานร่วมกับแคช ระบบการค้นหาข้อมูลบริการจะมีการเชื่อมต่อเพื่อสอบถามข้อมูลน้อยลง ทำให้ความเร็วในการค้นหาข้อมูลเร็วขึ้นไปด้วย

ในส่วนนี้เป็นการกล่าวถึงคุณสมบัติ และ การใช้ประโยชน์จาก K-PST เท่านั้น ลักษณะการทำงานของระบบการค้นหาข้อมูลที่ใช้ K-PST ร่วมกับแคชจะถูกกล่าวถึงอย่างละเอียดอีกครั้งในบทที่ 4

## 1.5 ขอบเขตงานวิจัย

วิทยานิพนธ์ชิ้นนี้ศึกษา ปรับปรุง และนำเสนอกลไกการค้นหาข้อมูลภายในระบบประมวลผลแบบกริด มุ่งเน้นไปที่ความเร็วในการค้นหาข้อมูลเป็นหลัก โครงสร้างที่นำเสนอจะถูก

ทดสอบเปรียบเทียบกับระบบการค้นหาข้อมูลที่ได้มีค้นคว้าและวิจัยไว้แล้ว โดยมีประเด็นเปรียบเทียบประสิทธิภาพของระบบ ได้แก่

1. ความเร็วการค้นหา (Discovery Speed) วัดจากจำนวนการเชื่อมต่อเพื่อสอบค้นหาข้อมูลในระบบ เทียบกับจำนวนการร้องขอให้ค้นหาข้อมูลทั้งหมด ในช่วงเวลาหนึ่ง

2. ความสิ้นเปลืองในการประกาศข้อมูลและดูแลรักษา วัดจากจำนวนการเชื่อมต่อเพื่อสอบถามความเปลี่ยนแปลงของข้อมูลบริการในระบบ ในช่วงเวลาหนึ่ง

3. พื้นที่จัดเก็บข้อมูลเพื่อการค้นหาข้อมูลสำหรับเอเจนต์แต่ละตัว

อนึ่ง วิทยานิพนธ์นี้ไม่พิจารณาเวลาที่สูญเสียไปสำหรับการค้นหา (delay time) แต่ละครั้ง และไม่พิจารณาปริมาณช่องทางสื่อสาร (bandwidth) ในส่วนของอัตราการพบข้อมูลจะให้ความสำคัญน้อยกว่าข้อ 1-3 ด้านบน

สำหรับการสอบประสิทธิภาพจะใช้วิธีจำลองผลขึ้น เพื่อเปรียบเทียบโมเดลการค้นหาที่มีผู้นำเสนอไว้แล้วกับโมเดลที่ผู้วิจัยได้พัฒนาขึ้น โดยสร้างสภาพแวดล้อมอันได้แก่ จำนวนการร้องขอข้อมูล จำนวนข้อมูลบริการในระบบ ระยะเวลา เป็นต้น ให้เป็นแบบเดียวกันแล้วปรับเปลี่ยนวิธีการค้นหา ผลที่ได้จากการทดสอบจะถูกบันทึก และ นำมาวิเคราะห์เปรียบเทียบกันต่อไป

## 1.6 รายละเอียดของวิทยานิพนธ์

วิทยานิพนธ์ฉบับนี้ เป็นการนำเสนอสถาปัตยกรรมของเอเจนต์แบบกระจายเพื่อการค้นหาข้อมูลบริการอย่างรวดเร็วแก่ระบบประมวลผลแบบกริด โดยรายละเอียดต่าง ๆ ภายในวิทยานิพนธ์ฉบับนี้ได้จัดแบ่งในส่วนของเนื้อหาออกเป็น 6 บท ซึ่งแต่ละบทมีหัวข้อและเนื้อหาดังต่อไปนี้

บทที่ 1 "บทนำ" อธิบายถึงวัตถุประสงค์ หลักการใหม่ที่ได้นำเสนอไว้ภายในวิทยานิพนธ์ รวมไปถึงรายละเอียดเนื้อหาโดยสรุปของแต่ละบท

บทที่ 2 "ระบบประมวลผลแบบกริด" อธิบายถึงภาพรวมของระบบประมวลผลแบบกริด อันเป็นระบบการประมวลผลแบบกระจายยุคใหม่ที่มุ่งเน้นการใช้งานทรัพยากรร่วมกัน ในบทนี้จะกล่าวถึง สถาปัตยกรรม โปรโตคอล และแอพพลิเคชันของระบบประมวลผลแบบกริดพอสังเขป เพื่อเป็นแนวทางในการทำความเข้าใจเทคโนโลยีพื้นฐานของงานวิจัยในวิทยานิพนธ์นี้ดียิ่งขึ้น

บทที่ 3 "กระบวนค้นหาข้อมูลภายในระบบประมวลผลแบบกริด" อธิบายถึงกระบวนการค้นหาข้อมูลภายในระบบประมวลแบบกริด ที่มีรากฐานมาจากสถาปัตยกรรมเชิงบริการ ที่ใช้ในเทคโนโลยีเว็บเซอร์วิส รวมทั้งกล่าวถึงรายละเอียดของ กระบวนการประกาศ กระบวนการค้นหา และ การทำงานร่วมกัน ชี้ให้เห็นถึงข้อดี ข้อเสียของสถาปัตยกรรมดังกล่าว นอกจากนี้ ยังกล่าวถึงการค้นคว้าวิจัยที่เคยมีมาแล้ว เพื่อแก้ปัญหาดังกล่าว อันเป็นประเด็นสำคัญของวิทยานิพนธ์ชิ้นนี้

บทที่ 4 "ระบบมัลติเอเจนต์เพื่อการค้นหาบริการอย่างรวดเร็วสำหรับระบบประมวลผลแบบกริด" อธิบายถึงระบบมัลติเอเจนต์ที่นำเสนอเพื่อการค้นหาข้อมูลแบบรวดเร็วสำหรับระบบประมวลแบบกริด กล่าวถึงโครงสร้างข้อมูล Signature File ที่ใช้ในอธิบายข้อมูลหรือรีซอร์สในโครงสร้างลำดับชั้น ซึ่งถูกนำมาใช้เป็นพื้นฐานสำหรับกระบวนการค้นหาข้อมูล กล่าวถึงการเพิ่มประสิทธิภาพระบบการค้นหาด้วยการใช้ประโยชน์แคช และจากการใช้ Knowledge ที่อาศัย K-PST ในการเป็นฐานความรู้ในการค้นหาข้อมูลที่พัฒนามาจาก Signature File อีกทีหนึ่ง

บทที่ 5 "การประเมินประสิทธิภาพของระบบการค้นหาบริการอย่างรวดเร็วสำหรับระบบประมวลแบบผลกริด" กล่าวถึงวิธีการประเมินประสิทธิภาพของระบบการค้นหาข้อมูลด้วยการสร้างระบบจำลองผล และอาศัยมาตรวัดที่เกี่ยวข้องเป็นเกณฑ์วัดประสิทธิภาพ จากนั้น จะแสดงผลการทดสอบของวิธีการค้นหาข้อมูลที่ได้นำเสนอไว้ในหลายแง่มุม รวมทั้งเปรียบเทียบกับงานวิจัยอื่นๆ ที่ได้มีการนำเสนอไว้แล้วก่อนหน้านี้

บทที่ 6 "บทสรุป" เป็นการสรุปและวิจารณ์ผลที่ได้จากวิเคราะห์และการทดลอง พร้อมทั้งปัญหาที่เกิดขึ้น ตลอดจนข้อเสนอแนะ สำหรับนำงานวิจัยในวิทยานิพนธ์นี้ไปพัฒนาใช้ประโยชน์ต่อไป

# บทที่ 2
# ระบบประมวลผลแบบกริด

## 2.1 กล่าวนำ

แรกทีเดียวคำว่า “กริด” (Grid) เป็นศัพท์ทางไฟฟ้ากำลัง ถูกนำมาใช้อธิบายถึงระบบการใช้ทรัพยากรไฟฟ้าร่วมกัน เพื่อประโยชน์ในการเข้าใช้พลังงานไฟฟ้าของผู้บริโภคอย่างสะดวกรวดเร็ว เสียค่าใช้จ่ายน้อย มีความน่าเชื่อถือและมีมาตรฐานแบบเดียวกัน ระบบดังกล่าวมีการพัฒนาอย่างต่อเนื่องเรื่อยมาจนกลายเป็นระบบไฟฟ้ากำลังที่เราใช้กันอยู่ในปัจจุบันนั่นเอง

ภายในระบบไฟฟ้ากำลัง พลังงานไฟฟ้าจากแหล่งกำเนิดไฟฟ้า จะถูกส่งมาตามสายไฟไปยังสถานที่ต่างๆ ผู้ใช้ก็เพียงแต่เสียบปลั๊กของอุปกรณ์เชื่อมต่อ เพื่อเข้าใช้พลังงานไฟฟ้าในการทำกิจกรรมของตน จะเห็นได้ว่าระบบการส่งไฟฟ้าแบบนี้สามารถเข้าใช้ได้โดยง่าย โดยไม่จำเป็นต้องทราบเลยว่าแหล่งกำเนิดไฟฟ้าอยู่ที่ใด

ในส่วนแวดวงคอมพิวเตอร์ คำว่า “กริด” ถูกหยิบมาใช้เป็นชื่อให้กับการประมวลผลแบบกระจายที่แนวคิดใกล้เคียงกับระบบไฟฟ้ากำลังที่ต้องการให้ ”ข้อมูล” “บริการ” หรือ "รีซอร์ส” ของระบบคอมพิวเตอร์ที่เชื่อมอยู่ในระบบเครือข่าย ไม่ว่าจะเป็น ข้อมูลหน่วยประมวลผล ข้อมูลแหล่งเก็บข้อมูล และบริการต่างๆ ถูกนำมาใช้ร่วมกันจนเกิดประโยชน์สูงสุด เทคโนโลยีที่กล่าวถึงมีชื่อว่าระบบประมวลผลแบบกริด (Grid Computing System)

ในยุคแรกเทคโนโลยีการประมวลผลแบบกริดถูกนำมาใช้ภายในกลุ่มองค์กรขนาดใหญ่ เช่น การวิจัยทางวิทยาศาสตร์ โรงงาน และ สถาบันการศึกษาเพื่อเป็นฐานในการศึกษาข้อมูลที่มีปริมาณมาก ซึ่งจำเป็นต้องใช้หน่วยประมวลผลจำนวนมากในการวิเคราะห์ข้อมูล และ จำเป็นต้องมีการใช้งานข้อมูลร่วมกันในหลายส่วน ระบบประมวลผลแบบกริดก็เปรียบเสมือนแหล่งข้อมูลกลาง ที่มีหน่วยประมวลผล ข้อมูล และบริการให้ ผู้ใช้งานที่มีข้อตกลงร่วมกันได้ใช้งานดังกล่าวได้อย่างสะดวกนั่นเอง

ระบบประมวลผลแบบกริดได้ให้ความหมายของ องค์กรเสมือน (Virtual Organization : VO) [3] แทนศูนย์รวมของข้อมูล ที่ผู้ใช้ให้สิทธิการใช้แก่สมาชิกรายอื่นในการเข้าใช้ข้อมูลนั้นร่วมกัน โดยผู้ใช้สามารถเป็นสมาชิกขององค์กรเสมือนดังกล่าวได้มากกว่า 1 องค์กร ข้อมูลในที่นี้มีความหมายถึงแหล่งเก็บข้อมูล หน่วยการประมวลผล และบริการ ที่สามารถแลกเปลี่ยนและใช้งานร่วมกันผ่านระบบเครือข่ายได้

ตัวอย่างของ องค์กรเสมือน ได้แก่ กลุ่มการวิจัยระบบเครือข่าย นักวิจัยกระจายอยู่ตามองค์กรย่อยในมหาวิทยาลัย อาจจะต้องอาศัยข้อมูลที่ต่างเก็บมาร่วมกัน นำมาสร้างระบบจำลอง

เพื่อศึกษาสภาพเครือข่ายของแต่ละภาควิชา หรือ ต่างคนต่างศึกษาข้อมูลของตนเทียบกับคนอื่น โดยใช้เครื่องคอมพิวเตอร์ประมวลร่วมกันอยู่ หรือ บางรายเป็นสมาชิกของกลุ่มวิจัยทางด้านการประมวลผลภาพ ต้องการจะใช้งานเครื่องประมวลผลเพื่อศึกษางานวิจัยของตนเองก็สามารถทำได้เช่นเดียวกัน กล่าวได้ว่า องค์กรเสมือนนี้เปรียบเสมือนชุมชนหนึ่งที่สามารถใช้งานข้อมูลที่มีการอนุญาตให้ใช้ร่วมกันได้โดยไม่ยุ่งยากมากนัก ประโยชน์ของการใช้ข้อมูลร่วมกันเหล่านี้ขององค์กรเสมือน ทำให้ผู้วิจัยสามารถทำงานร่วมกันได้สะดวกรวดเร็วมากยิ่งขึ้น จากตัวอย่างที่กล่าวมาแล้ว องค์กรเสมือนจะมุ่งเน้นไปที่การใช้ข้อมูลคอมพิวเตอร์ที่อยู่ในเครือข่ายร่วมกัน แต่จะมีความแตกต่างกันตามแต่จุดประสงค์ ขนาด และลักษณะทางสังคมของหน่วยทำงานนั้นๆ ในส่วนนี้เป็นการกล่าวถึง องค์กรเสมือนอย่างคร่าวเท่านั้น โดยรายละเอียดขององค์กรเสมือนจะกล่าวถึงในส่วนถัดไป

ปัจจุบันการใช้ข้อมูลและหน่วยประมวลผลภายในระบบการประมวลผลแบบกริดยังอยู่ในวงจำกัด อีกทั้งการขยายตัวของระบบนั้นเป็นไปได้ลำบากเนื่องจากระบบที่พัฒนาขึ้นมาอ้างอิงอยู่กับแพลตฟอร์มและกลไกการติดต่อสื่อสารข้อมูลที่จำเป็นต้องรับรู้กันทั้งระบบ จะเห็นได้จากในปัจจุบันองค์กรการวิจัยทางวิทยาศาสตร์จะมีระบบการประมวลผลแบบกริดขนาดใหญ่ที่มีประสิทธิภาพสูงของตัวเอง แต่การแลกเปลี่ยนข้อมูลและชิ้นส่วนของซอฟท์แวร์นั้นทำได้ลำบาก จึงเป็นประเด็นที่น่าสนใจหากข้อมูลและชิ้นส่วนที่มีคุณค่าเหล่านั้นสามารถแลกเปลี่ยนกันระหว่างหน่วยงานได้สะดวกขึ้น รวมถึงการเข้าใช้ข้อมูลสารสนเทศได้ง่าย อันจะนำประโยชน์มาสู่การพัฒนาระบบประมวลแบบกริดอีกมากเลยทีเดียว

## 2.2 แอพพลิเคชันในระบบประมวลผลแบบกริด

ในส่วนนี้เราจะกล่าวถึงแอพพลิเคชันในระบบการประมวลผลแบบกริด เพื่อให้เห็นภาพรวมของการทำงานของระบบกริดในด้านต่างๆ โดยได้แบ่งแอพพลิเคชันออกเป็นหมวดหมู่ตามแต่จุดประสงค์ที่สร้างขึ้นมา มีการสรุปลักษณะพิเศษของแอพพลิเคชันแต่ละประเภทไว้ที่ตารางที่ 2.1 จากนั้น จะเป็นการกล่าวในส่วนของรายละเอียดของแอพพลิเคชันแต่ละประเภท รวมทั้งชี้ให้ความต้องการเทคโนโลยีเพื่อเป็นพื้นฐานต่อการทำงานแอพพลิเคชันเหล่านั้น

### 2.2.1 Distributed Supercomputing

ระบบการคำนวณแบบกระจายขนาดใหญ่ใช้ระบบกริดสำหรับเชื่อมต่อข้อมูลหน่วยประมวลจากหลายๆแหล่ง เพื่อแก้ปัญหาที่ไม่สามารถทำได้หรือแก้ไขได้ยากในระบบใดระบบหนึ่ง ข้อมูลประมวลผลนี้อาจจะหน่วยประมวลผลขนาดใหญ่อย่างเครื่องซุปเปอร์คอมพิวเตอร์ของประเทศใดประทศหนึ่ง หรือ อาจจะเป็นเครื่องเวิร์กสเตชันที่พลังการคำนวณขนาดเล็กก็ได้

ประเด็นความต้องการของแอพพลิเคชันนี้คือ โปรโตคอลที่รองรับการเชื่อมต่อข้อมูลจากแหล่งข้อมูล ที่มีจำนวนโหนดเป็นจำนวนมาก วิธีการดูแลระบบ การจัดการเกี่ยวกับประสิทธิภาพของระบบที่มีความแตกต่างกันของแพลตฟอร์ม

**ตารางที่** 2.1 ประเภทและลักษณะการทำงานของแอพพลิเคชันของระบบประมวลผลแบบกริด

| Category | Examples | Characteristic |
|---|---|---|
| Distributed Supercomputing | DIS<br>Stellar dynamics<br>Ab initio chemistry | Very large problems needing lots of CPU, memory, etc. |
| High throughput | Chip design<br>Parameter studies<br>Cryptographic problems | Harness many otherwise idle resources to increase aggregate throughput. |
| On demand | Medical instrumentation<br>Network-enabled solvers<br>Cloud detection | Remote resource integrated with local computation, often for bounded amount of time. |
| Data intensive | Sky survey<br>Physics data<br>Data assimilation | Synthesis of new information from many or large data sources. |
| Collaborative | Collaborative Design<br>Data exploration<br>Education | Support communication or collaborative work between multiple participants. |

2.2.2 High-Throughput Computing

ระบบการคำนวณที่ต้องการให้ได้ปริมาณมาก สามารถนำระบบกริดมาใช้ในการวางแผนการรันงานที่มีจำนวนงานมาก ด้วยการส่งงานไปรันยังเครื่องที่มีสถานะการทำงานว่างอยู่ให้ทำงานตามจุดประสงค์ ประเด็นความท้าทายของแอพพลิเคชันประเภทนี้ก็จะใกล้เคียงกับ Distributed Super คือ ต้องการวิธีการค้นหาข้อมูลที่มีอยู่ในระบบ และพร้อมที่จะรับงานไปรันได้ ณ ขณะนั้น อย่างทันท่วงที

2.2.3 On-Demand Computing

แอพพลิเคชันที่ทำงานตามความต้องการใช้ความสามารถของกริดในการตอบความต้องการใช้ข้อมูล ที่ไม่สามารถหาได้จากระบบที่ลูกค้าใช้งานอยู่ ข้อมูลเหล่านี้อาจจะเป็น หน่วยประมวลผล ซอฟท์แวร์ แหล่งเก็บข้อมูล และอื่นๆ แอพพลิเคชันแบบนี้จะแตกต่างกับที่กล่าวมาแล้วตรงที่ จะถูกดำเนินการด้วยการพิจารณาค่าใช้จ่ายประกอบกับประสิทธิภาพ

ประเด็นสำคัญของแอพพลิเคชันแบบ On-Demand คือการพิจารณาความแตกต่างของความต้องการในการใช้ทรัพยากรและค่าใช้จ่าย ประกอบด้วย วิธีการวางแผนการใช้งานข้อมูล การดูแลรักษาระบบ ความปลอดภัยของระบบ และระบบการแบ่งผลประโยชน์ต่อผู้ใช้

2.2.4 Data-Intensive Computing

แอพพลิเคชันแบบเน้นข้อมูล มุ่งเน้นการสังเคราะห์ข้อมูลใหม่จากข้อมูลเดิมที่มีอยู่ ข้อมูลเดิมนี้ถูกเก็บไว้ในสถานที่ต่างๆ ในระบบกระจาย ไม่ว่าจะเป็นข้อมูลจากห้องสมุดอิเล็กทรอนิกส์หรือจากดาต้าเบส ซึ่งเป็นข้อมูลจำนวนมาก นอกจากนี้ การสังเคราะห์ข้อมูลใหม่ยังเกี่ยวข้องกับหน่วยประมวลผล และ การสื่อสารข้อมูลอีกด้วย ความท้าทายของแอพพลิเคชันแบบนี้ก็คือ การกำหนดการทำงานและการวางแผนการรันงานที่เกี่ยวข้องกับข้อมูลจำนวนมากและมีความสลับซับซ้อน รวมถึงการควบคุมการส่งข้อมูลที่มีปริมาณมากอีกด้วย

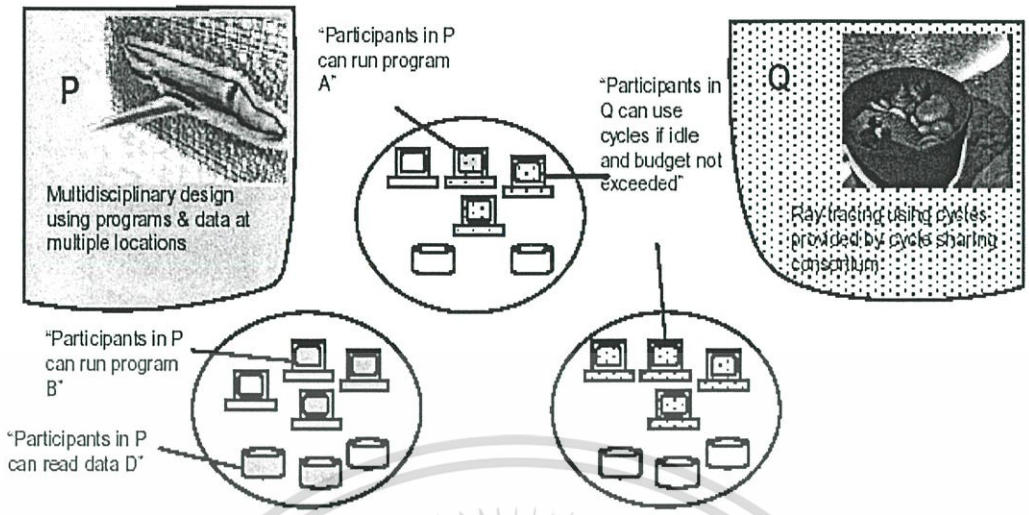
2.2.5 Collaborative Computing

แอพพลิเคชันแบบร่วมมือมีรูปแบบการทำงานพื้นฐานเกี่ยวข้องกับการเชื่อมโยงการทำงานของผู้ใช้หลายคนให้ทำงานร่วมกัน แอพพลิเคชันแบบนี้ถูกสร้างขึ้นมาให้มีพื้นที่การใช้รีซอร์-สร่วมกัน ส่วนใหญ่จะเป็นการใช้ข้อมูลหน่วยประมวลผลในการศึกษาระบบจำลอง และการใช้ข้อมูลเอกสารอิเล็กทรอนิกส์ร่วมกัน

ประเด็นการทำงานของแอพพลิชันแบบนี้ก็คือ วิธีการสื่อสารข้อมูลในลักษณะทันเวลา (real-time) เพื่อตอบสนองความต้องการสื่อสารของผู้ใช้ในระบบการทำงานเป็นทีมได้

## 2.3 องค์กรเสมือน

องค์กรเสมือนในระบบการประมวลผลแบบกริด หมายถึง กลุ่มสมาชิกในระบบเครือข่ายที่มีข้อตกลงในการใช้งานข้อมูลในระบบเครือข่ายร่วมกัน เพื่อจุดประสงค์อย่างใดอย่างหนึ่ง ข้อมูลในที่นี้ ยกตัวอย่างเช่น ไฟล์ข้อมูล แหล่งเก็บข้อมูล หน่วยประมวลผล ข้อมูลบริการ ชิ้นส่วนของซอฟท์แวร์ และข้อมูลสาระสำคัญอื่นๆ เป็นต้น ข้อมูลเหล่านี้จะถูกอนุญาตจากสมาชิกแต่ละรายให้ผู้อื่นได้เข้าใช้บริการตามข้อตกลงที่มีร่วมกัน ความสัมพันธ์ระหว่างสมาชิกกับข้อมูลที่ใช้ร่วมกันจะมีความแตกต่างกันไปตามจุดประสงค์ขององค์กรเสมือนนั้นๆ บางครั้งการใช้ข้อมูลร่วมกันอาจจะเป็นมากกว่าการแลกเปลี่ยนไฟล์ข้อมูล หรือแหล่งที่เก็บข้อมูล ยกตัวอย่างเช่น สมาชิกขององค์กรเสมือนของบริษัทซอฟท์แวร์ สามารถนำชิ้นส่วนซอฟท์แวร์ที่ได้จากการพัฒนาของเพื่อนสมาชิกมาใช้สำหรับการพัฒนาซอฟท์แวร์ของตน อีกกรณีหนึ่งก็คือ นักวิจัยในสถาบันวิจัยสามารถใช้งานหน่วยประมวลผลที่เพื่อนสมาชิกเสียสละให้ใช้ เพื่อวิเคราะห์ข้อมูลผ่านระบบเครือข่ายได้ เป็นต้น




**รูปที่** 2.1 ตัวอย่างองค์กรเสมือนในระบบประมวลผลแบบกริด

ในทางปฏิบัติสมาชิกในองค์กรสามารถเข้าเป็นสมาชิกขององค์กรเสมือนมากกว่าหนึ่งองค์กรได้ด้วยการแชร์ข้อมูลของตนบางส่วนหรือทั้งหมดให้กับองค์เสมือนที่ตนเป็นสมาชิกอยู่ ดังจะเห็นได้จากตัวอย่างองค์กรเสมือนใน รูปที่ 2.1 วงรีสามวงแสดงให้เห็นองค์กรจริง (Actual Organiztion) 3 องค์กร และมีองค์กรเสมือน 2 องค์กรได้แก่ P และ Q องค์กรเสมือน P เป็นกลุ่มของนักออกแบบอากาศยาน ส่วน Q เป็นกลุ่มของนักวิจัยรังสี จากรูป P คือกลุ่มทางซ้ายล่าง ส่วน Q คือ กลุ่มทางขวาล่าง และ กลุ่มองค์กรตรงกลางจะมีสมาชิกของทั้งองค์กรเสมือน P และ Q อยู่

ลักษณะการใช้งานร่วมกันของสมาชิกภายในองค์กรเสมือน สมาชิกจะเป็นผู้เผยแพร่ข้อมูลที่จะใช้งานร่วมกันให้แก่ระบบ และกำหนดข้อบังคับการใช้ข้อมูลนั้นให้ทราบด้วย ยกตัวอย่างเช่น สมาชิกของ P รายหนึ่งอาจจะให้สมาชิกรายอื่นใช้งาน หรือใช้บริการหน่วยประมวลผลของตนสำหรับทำการทดสอบระบบจำลองที่มีลักษณะปัญหาอย่างง่าย สมาชิกที่ต้องการใช้งานบริการนี้จะได้ทราบว่า ในองค์กรเสมือนของตน มีสมาชิกรายดังกล่าวมีบริการอะไรให้ใช้บ้าง และให้ใช้ได้ด้วยวิธีการใด

ความสัมพันธ์ระหว่างข้อมูลที่ใช้ร่วมกันของสมาชิกสามารถเปลี่ยนแปลงได้ตลอดเวลา สิทธิในการใช้บริการข้อมูลของสมาชิกแต่ละรายก็มีลำดับความสำคัญที่แตกต่างกันด้วย ซึ่งจะมีข้อกำหนดที่ได้ตกลงกันไว้ก่อนเข้าใช้บริการในระบบของสมาชิก ยกตัวอย่างเช่น สมาชิกที่มีฐานะเป็นผู้ดูแลระบบ สามารถใช้งานบริการทุกอย่างในระบบได้ ในขณะที่สมาชิกที่เป็นนักศึกษาสามารถใช้งานข้อมูลระบบได้บางส่วน ข้อตกลงของระบบสามารถเปลี่ยนแปลงได้เสมอ อย่างเช่น เมื่อสมาชิกในฐานะนักศึกษาถูกกำหนดสิทธิให้เป็นผู้ดูและระบบ ข้อมูลของสมาชิกจะถูกตรวจตราโดยองค์กรเสมือนอยู่เสมอ ว่าขณะนั้นมีสถานะเป็นอย่างไร ใครมีสิทธิใช้งานบ้าง

ในทางปฏิบัติ การแชร์ข้อมูลไม่ได้มีลักษณะการทำงานแบบไคลเอนต์เซิร์ฟเวอร์เท่านั้น แต่เป็นระบบแบบเพียร์ทูเพียร์ กล่าวคือ ผู้ให้บริการ ผู้ใช้บริการ และข้อมูลที่ใช้ร่วมกันจะถือเป็นสมาชิกของระบบเท่าเทียมกัน ข้อมูลที่ใช้ร่วมกันขององค์กรเสมือน มีส่วนประกอบจากข้อมูลหลายชิ้นที่มีเจ้าของหลายคนและมาจากหลายองค์กรจริงได้ ยกตัวอย่างเช่น ในองค์กรเสมือน Q จะมีหน่วยประมวลผลที่มาจากหลายองค์กรจริง เป็นต้น

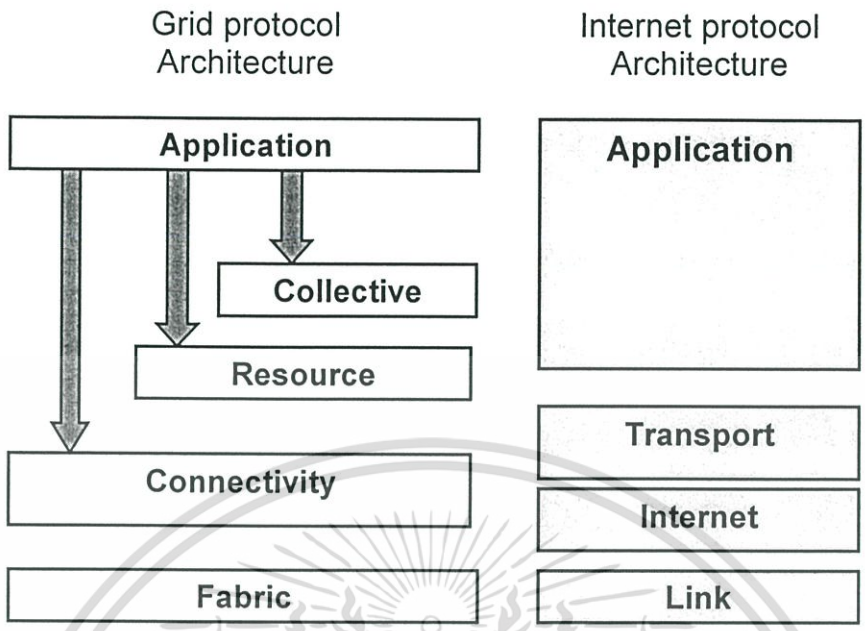
ข้อมูลของระบบแบบหนึ่งอาจถูกใช้ได้ในด้วยวิธีการที่แตกต่างกัน ขึ้นอยู่กับสถานที่ใช้และจุดประสงค์การใช้งานข้อมูลร่วมกัน ยกตัวอย่างเช่น เครื่องประมวลผลในองค์กรเสมือนหนึ่งมีหน้าที่หลักในการรันงานที่เกี่ยวข้องกับการประมวลผลภาพ แต่เมื่อมีการร้องขอจากสมาชิกให้ดำเนินการประมวลผลทั่วไปก็สามารถทำได้

ลักษณะขององค์กรเสมือนที่ระบุไว้ข้างต้น แสดงถึงแนวคิดของระบบประมวลผลแบบกระจาย ที่มุ่งเน้นการใช้ข้อมูลคอมพิวเตอร์ร่วมกันเพื่อบรรลุเป้าหมายของแต่ละองค์กร องค์กรเสมือนถือเป็นแนวคิดสำคัญของระบบการประมวลผลแบบกริด เปรียบได้กับศูนย์กลาง ที่สมาชิกจะใช้งานข้อมูลร่วมกันเพื่อทำงานของตนได้อย่างมีประสิทธิภาพ จะเห็นได้ว่า ประโยชน์ที่ได้จากแนวคิดนี้ก็คือ การทำงานเป็นทีม และ การใช้ข้อมูลที่มีอยู่อย่างคุ้มค่า นั่นเอง

## 2.4 สถาปัตยกรรมของระบบประมวลผลแบบกริด

แนวคิดที่สำคัญที่สุดขององค์กรเสมือนก็คือการใช้ข้อมูลเพื่อทำงานร่วมกัน (Interoperability) การทำงานพื้นฐานขององค์กรเสมือนจึงต้องระบุถึงการเชื่อมต่อข้อมูลระหว่างสมาชิกได้ สถาปัตยกรรมของกริดก็ถูกสร้างขึ้นมาเพื่อสนับสนุนแนวคิดขององค์กรเสมือน โดยระบุถึงวิธีการพื้นฐานที่สมาชิกขององค์กรเสมือนเข้าใช้ ต่อรอง จัดการ และอนุญาตให้ใช้ข้อมูลที่ตนมีอยู่ สำหรับมาตรฐานที่มีรากฐานมาจากสถาปัตยกรรมแบบเปิด (Open Architecture) [7] อย่างการแชร์ข้อมูล ความสามารถในการขยายขนาด ความเข้ากันได้ของข้อมูล ถือเป็นความสามารถของสถาปัตยกรรมของกริดที่เพิ่มเติมเข้ามาในระยะหลัง

ความสามารถในการใช้ข้อมูลเพื่อทำงานร่วมกันเป็นประเด็นสำคัญที่ต้องพิจารณาสถาปัตยกรรมของกริดจะต้องมีความสามารถในการระบุการเชื่อมต่อระหว่างสมาชิกได้แบบอิสระไม่ขึ้นอยู่กับแพลตฟอร์ม ภาษา หรือ สภาพแวดล้อมของการพัฒนาของสมาชิก ดังนั้นความสามารถในการค้นหาข้อมูล แสดงตัวตนของสมาชิก การรับรองสิทธิ และการระบุการแชร์ข้อมูลจะต้องเป็นวิธีการที่ยืดหยุ่นและไม่สิ้นเปลืองมากนัก ซึ่งจะทำให้กลไกการเชื่อมต่อของระบบสามารถทำได้อย่างรวดเร็ว นอกจากนี้ โปรโตคอลที่สร้างขึ้นจะต้องไม่กระทบกับการทำงานพื้นฐานของข้อกำหนดในองค์กรจริงด้วย




รูปที่ 2.2 โปรโคคอลแสตกของสถาปัตยกรรมกริดเปรียบเทียบกับสถาปัตยกรรมอินเตอร์เนต

โครงสร้างสถาปัตยกรรมของกริดได้แบ่งออกเป็นเลเยอร์ต่างๆ 5 เลเยอร์ ประกอบด้วย Application, Collective, Resource, Connective และ Fabric ในแต่ละเลเยอร์จะมีหน้าที่การทำงานแตกต่างกัน ความสัมพันธ์ของเลเยอร์เหล่านี้ก็คือ การทำงานของเลเยอร์ที่อยู่บนจะถูกสร้างอยู่บนการทำงานของเลเยอร์ระดับล่าง เมื่อเปรียบเทียบกับอินเตอร์เน็ตโปรโตคอล สถาปัตยกรรมของกริดจะมีลักษณะคล้ายคลึงกัน กลไกการทำงานของ Fabric ของสถาปัตยกรรมของกริดเทียบได้กับ Link ของอินเตอร์เน็ตโปรโตคอล โดยมีหน้าที่เชื่อมต่อกันในระดับกายภาพ ส่วน Connectivity ทำหน้าที่รับส่งข้อมูล และการอนุญาตการใช้บริการ ครอบคลุมการทำงานของเลเยอร์ Internet และ Transport ของอินเตอร์เน็ตโปรโตคอล ในส่วน Resource, Collective และ Application ของกริด เทียบได้กับส่วนของ Application เลเยอร์ของอินเตอร์เน็ตโปรโตคอล จะเห็นว่าในสถาปัตยกรรมของกริดจะมีเลเยอร์ Resource และ Collective เพิ่มขึ้นมา ทั้งสองทำหน้าที่เกี่ยวข้องกับการบริหารจัดการและรวบรวมข้อมูลที่ใช้ร่วมกัน อันเป็นจุดประสงค์หลักขององค์กรเสมือนของกริดนั่นเอง

2.4.1 Fabric Layer

ฟาบริกเลเยอร์จัดการข้อมูลที่ใช้ร่วมอยู่ภายในระบบกริด ข้อมูลอาจจะเป็นลอจิกคอลเอ็นติตี้ ได้แก่ ไฟล์ข้อมูล หรือ เครื่องประมวลผลแบบคลัสเตอร์ ในกรณีดังกล่าว การนำมาใช้จะเกี่ยวข้องกับโปรโตคอลภายใน ซึ่งไม่อยู่ในขอบเขตการทำงานของระบบสถาปัตยกรรมของกริด

การทำงานของฟาบริกเลเยอร์จะเกี่ยวข้องกับส่วนประกอบพื้นฐานของสถาปัตยกรรมของกริด มีหน้าที่ระบุการเชื่อมต่อข้อมูล และทำโอเปอเรชันการแชร์ข้อมูล ข้อมูลที่มีอยู่ในระบบจะต้องถูกตรวจสอบโครงสร้าง สถานะทำงาน และคุณสมบัติได้โดยระบบ นอกจากนี้ จะต้องมีส่วนทำงานจัดการข้อมูล (Resource Management) ที่มีหน้าที่ควบคุมคุณภาพการรับส่งข้อมูล

สำหรับการจัดการข้อมูลที่มีลักษณะต่างๆ จะมีวิธีการองค์ประกอบดังนี้

1. Computational Resources เป็นข้อมูลที่เกี่ยวข้องกับการเริ่มต้นรัน ตรวจสอบ และควบคุมการทำงานโปรแกรม กลไกส่วนนี้จึงมีหน้าจองหน่วยประมวลผลเพื่อการทำงานโปรแกรมของสมาชิก พิจารณาจากสถานะการทำงานของฮาร์ดแวร์ ซอฟท์แวร์ ณ ขณะนั้น เช่น โหลด และ สถานะของคิว ในระบบกริดได้
2. Storage Resources เป็นข้อมูลที่เกี่ยวข้องกับแหล่งเก็บข้อมูล กลไกส่วนนี้มีหน้าที่จัดวาง และ เก็บไฟล์ข้อมูลในระบบกริด กลไกดังกล่าวจึงต้องสามารถควบคุมการทำงานของการจองเนื้อที่หน่วยความจำ โดยพิจารณาจำนวนเนื้อที่ในระบบที่สามารถใช้งานได้
3. Network Resources เป็นข้อมูลที่เกี่ยวข้องกับการสื่อสารในระบบเครือข่าย กลไกการทำงานต่อข้อมูลแบบนี้มีหน้าที่จองเนื้อที่การรับส่งข้อมูลได้ โดยพิจารณาจากลักษณะข้อมูลในเครือข่าย
4. Code repositories เป็นรูปแบบการเก็บข้อมูลแบบพิเศษ สำหรับการเก็บเวอร์ชันของข้อมูล และโค้ดของออบเจ็ก
5. Catalogs เป็นรูปแบบการเก็บข้อมูลแบบพิเศษ สำหรับการอิมพลีเมนต์การคิวรีและปรับปรุงการทำงานของชุดข้อมูล เช่น ฐานข้อมูลสัมพันธ์

2.4.2 Connectivity Layer

คอนเน็กทิวิตีเลเยอร์ระบุถึงกลไกการสื่อสารและโปรโตคอลการรับรองสิทธิ์ (Authentication Protocol) อันเป็นส่วนสำคัญของระบบเครือข่ายกริด การสื่อสารข้อมูลระบุถึงการแลกเปลี่ยนข้อมูลระหว่างข้อมูลในระดับฟาบริกเลเยอร์ ส่วนโปรโตคอลการเข้าใช้บริการจะจัดหาบริการที่เกี่ยวข้องกับการเข้ารหัสข้อมูลเพื่อยืนยันตัวตนของสมาชิกและข้อมูล

ความต้องการด้านการสื่อสารข้อมูลจะเกี่ยวข้องกับการรับส่ง การค้นหาเส้นทาง และการอ้างอิงชื่อ กลไกที่ถูกนำมาใช้จะอ้างอิงจากเทคโนโลยีเดิมที่มีอยู่อันได้แก่ โปรโตคอลทีซีพีไอพี ยูดีพี ดีเอ็นเอส เป็นฐานในการพัฒนา ในอนาคตจะมีการขยายโปรโตคอลเหล่านี้ให้เหมาะสมกับลักษณะของระบบกริดต่อไป

ในส่วนการรับรองสิทธิ์จะพิจารณาต่อระบบความปลอดภัยของข้อมูล ในระยะแรกระบบกริดจะใช้งานโปรโตคอลการรับรองสิทธิ์ที่ใช้อยู่ในสถาปัตยกรรมของอินเตอร์เน็ต อย่างไรก็ดี

ระบบกริดมีลักษณะแตกต่างกับระบบอินเตอร์เน็ต ดังนั้น การพัฒนาโปรโตคอลการรับรองสิทธิ์จะให้เหมาะกับองค์กรเสมือน โดยมีลักษณะดังต่อไปนี้

1. Single sign on ผู้ใช้บริการต้องสามารถล็อกอินเข้าใช้บริการต่างๆ ที่ระบุไว้ในเลเยอร์ Fabric จากระบบเพียงครั้งเดียว
2. Delegation ผู้ใช้บริการต้องสามารถรันโปรแกรมของตนได้ตามสิทธิของตน ดังนั้น โปรแกรมดังกล่าวจึงต้องสามารถใช้ข้อมูลที่ผู้ใช้บริการมีสิทธิใช้บริการ นอกจากนี้ โปรแกรมควรจะทำงานแลกเปลี่ยนข้อมูลร่วมกับโปรแกรมอื่นในระบบได้ด้วย
3. Integration with various local security solutions ผู้ให้บริการและผู้ขอใช้บริการควรจะอิมพลีเมนต์ระบบความปลอดภัย ให้ทำงานร่วมกับระบบเดิมได้โดยไม่ต้องเปลี่ยนแปลงมากนัก
4. User-based trust relationships เพื่อที่จะทำให้ผู้ใช้บริการใช้งานข้อมูลจากผู้ให้บริการแบบหลากหลาย ระบบความปลอดภัยที่สร้างขึ้นมาจะต้องไม่กระทบกับระบบความปลอดภัยของผู้ให้บริการแต่ละราย

วิธีการรักษาความปลอดภัยของกริดควรจะมีความยืดหยุ่น สนับสนุนการปกป้องข้อมูลในระบบการสื่อสาร และสามารถควบคุมการตัดสินใจของระบบการรับรองสิทธิ์ รวมถึงความสามารถในการเชื่อมข้อมูลที่ถูกต้องด้วย

2.4.3 Resource Layer

รีซอร์สเลเยอร์ถูกสร้างอยู่บนคอนเน็คทิวิตีเลเยอร์ เพื่อระบุกลไกการสื่อสารอย่างปลอดภัย การเริ่มต้นการเชื่อมต่อ การตรวจตรา การควบคุม การเก็บรักษาข้อมูล และการใช้งานร่วมกันของข้อมูลส่วนบุคคล เลเยอร์นี้อิมพลีเมนต์กลไกของฟาบริกเลเยอร์ ในการเข้าใช้ และควบคุมข้อมูลในระดับโลคอล โดยไม่ได้เกี่ยวข้องกับการเชื่อมต่อข้อมูลในระดับกลอบอล ประเด็นดังกล่าวจะเกี่ยวข้องคอนเน็คทิวิตีเลเยอร์ ซึ่งจะกล่าวในหัวข้อถัดไป

โปรโตคอลที่ใช้ในเลเยอร์ Resource แบ่งออกเป็น 2 รูปแบบใหญ่ ดังนี้

1. Information Protocol ถูกนำมาใช้ในการเก็บข้อมูลเกี่ยวกับโครงสร้าง และสถานะของข้อมูล ยกตัวอย่างเช่น ข้อกำหนดพื้นฐาน ภาระการใช้งานปัจจุบัน และ กฎการใช้ข้อมูล
2. Management Protocol ถูกนำมาใช้ต่อรองการเข้าใช้ข้อมูลที่แชร์ร่วมกันอยู่ โปรโตคอลนี้ทำหน้าที่ควบคุมการใช้ข้อมูลให้เป็นไปตามกฎเกณฑ์ที่วางไว้ร่วมกัน นอกจากนี้ โปรโตคอลนี้อาจจะสนับสนุนการตรวจตราสถานะการทำงาน และควบคุมการทำงานที่มีต่อข้อมูลนั้นๆ

ในรีซอร์สเลเยอร์ จะต้องมีโปรโตคอลหลักในการใช้งานข้อมูลที่มีลักษณะแตกต่างกันให้ใช้งานร่วมกันได้ โดยที่ไม่กระทบกับประสิทธิภาพของการทำงานระดับบนมากนัก

2.4.4 Collective

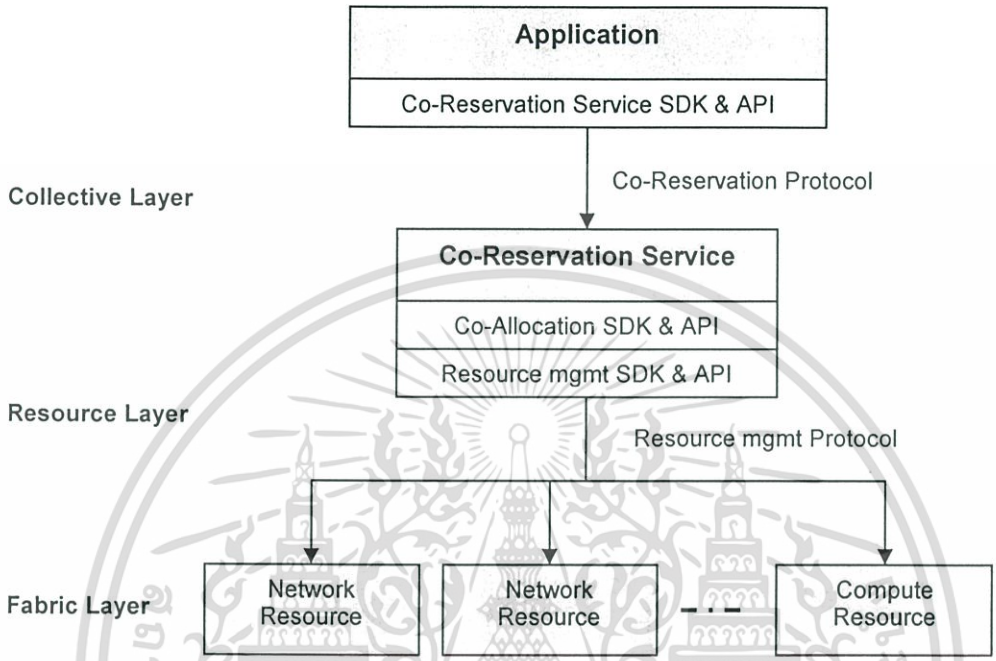
คอลเล็กทีฟเลเยอร์ เกี่ยวข้องกับโปรโตคอลและบริการในระดับกลอบอล ซึ่งแตกต่างจากรีซอร์สเลเยอร์ ที่เกี่ยวข้องกับโปรตคอลระดับโลคอลที่จัดการกับข้อมูลตัวใดตัวหนึ่ง โดยคอลเล็กทีฟเลเยอร์จะจัดการกับข้อมูลภายในระบบให้ทำงานร่วมกันเพื่อจุดประสงค์อย่างใดอย่างหนึ่ง อาทิ เช่น เก็บ ค้นหา หรือควบคุมข้อมูลเหล่านั้น เนื่องจากเลเยอร์นี้สร้างอยู่บนรีซอร์สเลเยอร์ ทำให้โปรโตคอลที่สร้างขึ้นมาใหม่สามารถอิมพลีเมนต์โปรโตคอลที่มีอยู่แล้วได้ และสามารถทำงานได้ในหลายรูปแบบ ยกตัวอย่างเช่น

- Directory Service เป็นบริการที่ให้ผู้ใช้บริการสามารถค้นหาคุณสมบัติของข้อมูลที่มีอยู่ในองค์กรเสมือนได้ ด้วยการใส่แอตทริบิวส์ เช่น ประเภทข้อมูล ขนาด เป็นคำสำคัญในการค้นหา
- Co-allocation, scheduling and Broker Service เป็นบริการช่วยสมาชิกขององค์กรเสมือนจัดการจองข้อมูลที่เหมาะสมเพื่อทำงานอย่างใดอย่างหนึ่ง
- Monitoring and diagnostics Services เป็นบริการที่ตรวจสอบสถานะของข้อมูลที่ผิดปกติว่า สูญหาย ถูกโจมตี หรือ ไม่พร้อมใช้งาน
- Data Replication Services สนับสนุนการใช้งานแหล่งข้อมูลภายในระบบกริดให้มีประสิทธิภาพ ตามมาตรวัดที่อ้างอิง เช่น Response Time ความน่าเชื่อถือ และความสิ้นเปลือง
- Software Discovery Service เป็นบริการที่เลือกซอฟท์แวร์ที่มีอยู่ และแพลตฟอร์มที่ใช้รันงาน โดยพิจารณาลักษณะปัญหาตามความเหมาะสม
- Collaboration services เป็นบริการที่สนับสนุนการแลกเปลี่ยนข้อมูลภายในระบบที่มีผู้ใช้บริการจำนวนมาก

จากตัวอย่างด้านบน คอลเล็กทีฟเลเยอร์ จะอิมพลีเมนต์การทำงานของรีซอร์สเลเยอร์ ให้ควบคุมข้อมูลหลายๆ ส่วนในองค์กรเสมือนให้สามารถทำงานในจุดประสงค์ที่เฉพาะเจาะจงในลักษณะต่างๆ ซึ่งแตกต่างจากรีซอร์สเลเยอร์ที่ระบุถึงการทำงานต่อข้อมูลลักษณะทั่วไปเท่านั้น

ฟังก์ชันการทำงานของคอลเล็กทีฟเลเยอร์สามารถถูกนำมาสร้างเป็นบริการแบบหนึ่งในระบบได้ด้วยโปรโตคอลที่เกี่ยวข้อง หรือ SDK ที่ถูกออกแบบมาเพื่อเชื่อมต่อแอพพลิเคชันต่างๆ บริการที่สร้างขึ้นจะสร้างอยู่บนโปรโตคอลของรีซอร์สเลเยอร์ ยกตัวอย่างเช่นในรูปที่ 2.3 แสดงให้เห็นถึง co-allocation API และ SDK ที่ใช้โปรโตคอลการจัดการข้อมูลในเลเยอร์มาเป็นฐานในการ

พัฒนาโปรโตคอล co-reservation service ที่มีความสามารถเพิ่มขึ้นจากโปรโตคอลเดิม เช่น การรับรองสิทธิ์ และ การดูแลรักษาข้อมูล นอกจากนี้ ในส่วนแอพพลิเคชันเลเยอร์ สามารถใช้ co-reservation service ในการร้องขอข้อมูลแบบจุดต่อจุดในระบบเครือข่ายได้

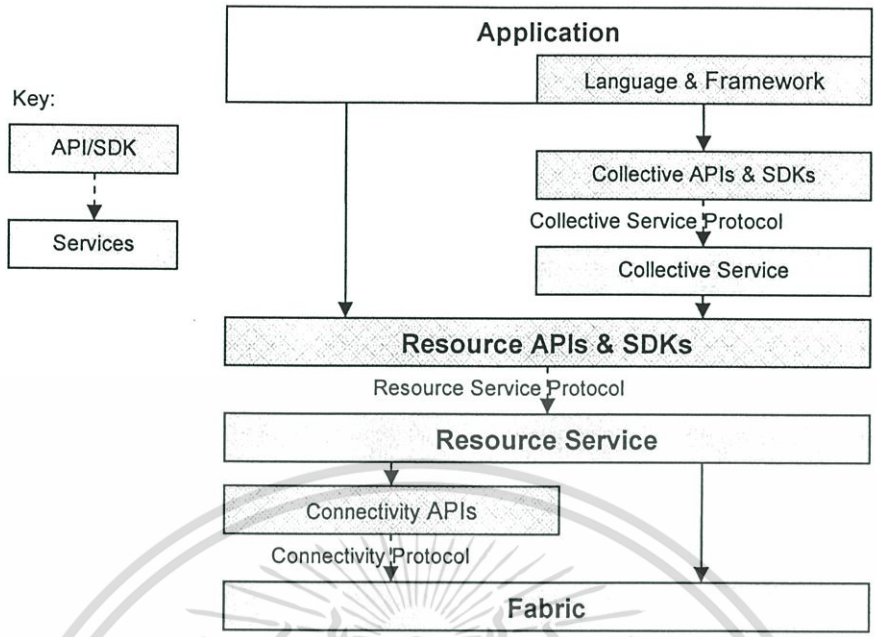



รูปที่ 2.3 ความสัมพันธ์ระหว่างเลเยอร์ต่างๆ ในการจัดการข้อมูลโดยอาศัย SDK และ API

โปรโตคอลที่เป็นส่วนประกอบของคอลเล็กทีฟเลเยอร์ อาจจะถูกสร้างขึ้นมาตามความต้องการของผู้ใช้รายใดรายหนึ่ง องค์กรเสมือน หรือ แอพพลิเคชันโดเมน ก็ได้ ดังนั้นโปรโตคอลที่ใช้จึงมีจุดประสงค์ทั่วไปในการรองรับความต้องการของผู้ใช้บริการส่วนใหญ่ อย่างไรก็ตาม ก็ต้องมีลักษณะเฉพาะกับจุดประสงค์ของผู้ใช้บริการอื่นได้ด้วย

2.4.5 Application Layer

ส่วนประกอบสุดท้ายของสถาปัตยกรรมกริดก็คือ แอพพลิเคชันเลเยอร์ ประกอบด้วยแอพพลิเคชันของผู้ใช้บริการที่ทำงานอยู่ในองค์กรเสมือน แอพพลิเคชันอยู่ในระบบกริดจะถูกสร้างขึ้นมาโดยอาศัยบริการ หรือโปรโตคอลการทำงานในเลเยอร์ระดับล่าง ในรูปที่ 2.4 แสดงสถาปัตยกรรมกริดในมุมของผู้พัฒนา แอพพลิเคชันจะถูกสร้างขึ้นมาโดยอิมพลิเมนต์บริการในระดับเลเยอร์ที่ต่ำกว่า ในการเข้าใช้และควบคุมข้อมูลที่มีอยู่ในระบบ บริการที่กล่าวมาได้แก่ การจัดการข้อมูล การเข้าใช้ข้อมูล การค้นหาข้อมูล และอื่นๆ ในแต่ละเลเยอร์อาจจะระบุการอิมพลีเมนต์ที่มีการแลกเปลี่ยนข้อมูลระหว่างบริการเพื่อทำงานในจุดประสงค์เฉพาะอย่างได้




รูปที่ 2.4 สถาปัตยกรรมของกริดในมุมของการพัฒนา

## 2.5 เทคโลยีที่เกี่ยวข้องกับกริด

แนวคิดในการควบคุมข้อมูลที่มีลักษณะไม่คงที่ในองค์กรเสมือนเป็นปัจจัยพื้นฐานของสถาปัตยกรรมกริด ปัจจุบันมีเทคโนโลยีหลายอย่างมีการทำงานคล้ายคลึงกับกริด แต่เทคโนโลยีเหล่านั้นก็ไม่ได้ออกแบบมาเพื่อตอบความต้องการของกริดได้อย่างครบถ้วน หากแต่ออกแบบมาเพื่อจุดประสงค์เฉพาะของมันเอง กริดและเทคโนโลยีที่เกี่ยวข้องจึงมีความแตกต่างกันอยู่ ในส่วนนี้จะเป็นการกล่าวถึงเทคโนโลยีที่มีความใกล้เคียงกับกริดและชี้ให้เห็นข้อแตกต่างของมันรวมถึงการพัฒนาร่วมกันระหว่างกริดและเทคโนโลยีเหล่านั้น

### 2.5.1 World Wide Web

ความแพร่หลายของเว็บเทคโนโลยีทำให้มันน่าสนใจที่จะนำมาพัฒนาโครงสร้างองค์กรเสมือนและแอพพลิเคชัน อย่างไรก็ตาม แม้ว่าเว็บเทคโนโลยีจะมีสิ่งอำนวยความสะดวกมากมายที่ให้ผู้ใช้บริการเข้าถึงข้อมูลในระบบด้วยวิธี การของไคลเอ็นต์เซิร์ฟเวอร์ แต่ก็ยังไม่เพียงพอต่อความต้องการของฟังก์ชันที่ทำงานในระบบกริด ยกตัวอย่างเช่น เว็บเบราเซอร์ยังไม่สามารถรองรับการรับรองสิทธิ์แบบ Single sign-on หรือ delegation ได้

จากจุดนี้จะเห็นได้ว่าหากมีการพัฒนาระบบกริดและเว็บเทคโนโลยีเข้าด้วยกัน กลไกการทำงานบางอย่างจะต้องถูกเพิ่มเติมเข้าไปยังเว็บเทคโนโลยีเพื่อรองรับความต้องการของระบบกริด ยกตัวอย่างเช่น ระบบการรับรองสิทธิ์ จะต้องมีการพัฒนาในส่วนของกลไกการรับรองสิทธิ์ เมื่อนำ

เว็บเทคโนโลยีมาใช้งานจะได้ประโยชน์ของความสะดวกต่อการเข้าใช้ข้อมูล เหมาะกับการเป็นศูนย์กลางการใช้บริการข้อมูลและแอพพลิเคชัน (Portal) ซึ่งมีงานวิจัย WebOS [8] ระบุไว้

2.5.2 Application and Storage Service Provider

ผู้ให้บริการแอพพลิเคชัน (Application Service Providers: ASP) แหล่งเก็บข้อมูล (Storage Service Provider: SSP) ถูกสร้างขึ้นเพื่อให้บริการงานของแอพพลิเคชัน ซึ่งตั้งอยู่ภายนอกแอพพลิเคชัน ลูกค้าสามารถเข้าใช้บริการในระบบได้โดยมีระดับการตกลงการใช้ข้อมูลร่วมกัน (Service Level Agreement: SLA) และใช้เทคโนโลยี VPN ในการจัดการความปลอดภัยข้อมูลในระบบ

จากลักษณะขององค์กรเสมือน เทคโนโลยีที่กล่าวมาเป็นพื้นฐานในการพัฒนาในระดับล่างของกริดเท่านั้น เรายังไม่สามารถนำเทคโนโลยี VPN มาใช้ตอบความต้องการในการเชื่อมโยงข้อมูลของกริดได้ทั้งหมด เนื่องจาก รีซอร์ทที่ใช้ร่วมกันยังไม่สามารถทำหน้าที่เป็นผู้ให้บริการข้อมูลและไม่สามารถควบคุมการแชร์ข้อมูลเหล่านั้นได้

การพัฒนากริดร่วมกับ ASP และ SSP สามารถทำได้ในหลายลักษณะ ยกตัวอย่างเช่น มาตรฐานการให้บริการกริดและ โปรโตคอลกริดสามารถนำมาใช้เทคโนโลยีนี้มาใช้ในการแยกความแตกต่างของฮาร์ดแวร์และซอฟท์แวร์ ลูกค้าสามารถตกลงขอใช้บริการข้อมูลฮาร์ดแวร์ และใช้โปรโตคอลของกริดรีซอร์ส ค้นหาฮาร์ดแวร์เพื่อรันแอพพลิเคชันของลูกค้า กลวิธีการ delegation ที่ยืดอยุ่นและวิธีการใช้งานข้อมูลสามารถทำให้ลูกค้าส่งรันแอพพลิเคชันของตนกับ ASP ได้โดยตรง มีประสิทธิภาพ มีความปลอดภัยสูงเมื่อมีการใช้งาน SSP และสามารถเชื่อมต่อการทำงานระหว่าง ASP และ SSP เพื่อแก้ปัญหาที่มีซับซ้อนได้ โครงสร้างความปลอดภัยแบบ single sign-on ก็สามารถขยายเข้าไปยังกลุ่มโครงสร้างความปลอดภัยได้อย่างอิสระเพื่อสนับการแก้ปัญหาที่ใช้เทคโนโลยี ASP และ SSP

2.5.3 Enterprise Computing System

เทคโนโลยีเพื่อธุรกิจเช่น CORBA J2EE และ DCOM ถูกพัฒนาขึ้นมาเพื่อสร้างแอพพลิเคชันในระบบแบบกระจาย เทคโนโลยีดังกล่าวมีบริการพื้นฐานในการติดต่อสื่อสาร อาทิ การเชื่อมต่อข้อมูล การเรียกบริการจากระยะไกล และกลไกการค้นหา ทำให้ง่ายต่อการพัฒนาระบบการแชร์ข้อมูลร่วมกันในองค์กรใดองค์กรหนึ่ง อย่างไรก็ตาม เทคโนโลยีดังกล่าวก็ไม่ได้สนับสนุนความต้องการขององค์กรเสมือนเท่าใดนักเนื่องจากลักษณะการเชื่อมต่อจะขึ้นอยู่กับแพลทฟอร์มของกลุ่มการทำงานเดียว และ มีลักษณะการทำงานหลักเป็นแบบไคลเอ็นต์เซิร์ฟเวอร์ มากกว่าการทำงานแบบร่วมมือที่เกี่ยวข้องกับข้อมูลที่หลากหลาย

อย่างไรก็ตาม กลไกการทำงานบางอย่างของเทคโนโลยีเพื่อธุรกิจก็ถูกนำมาใช้ สร้างมาตรฐานการเชื่อมต่อของกริด อย่างเช่นในกรณีของ CORBA การเชื่อมต่อข้อมูลข้ามแพลตฟอร์มถูกนำมาใช้ในการจัดการข้อมูลที่มีอยู่ในองค์กรเสมือน เช่นเดียวกับเทคโนโลยี JINI ถูกนำมาใช้ในการเก็บข้อมูลของข้อมูลขนาดเล็ก เป็นต้น

### 2.5.4 Internet and Peer-to-Peer Computing

เทคโนโลยีอินเตอร์เนต และ เพียร์ทูเพียร์เป็นตัวอย่างของเทคโนโลยีที่มีจุดเด่นในการทำโอเปอเรชันของกริดอาทิ การแชร์ข้อมูล การจัดโครงสร้าง ที่แตกต่างจาก เทคโนโลยีที่มีรากฐานจากไคลเอนต์เซิร์ฟเวอร์ทั่วไป รูปแบบของอินเตอร์เนตและเพียร์ทูเพียร์ถือได้ว่าเป็นรากฐานของกริด กล่าวคือ มีกลไกการทำงานที่มีการใช้โครงสร้างข้อมูล และการเก็บรักษาข้อมูลต่างแพลตฟอร์มให้แลกเปลี่ยนกันได้ ในส่วนการทำงานร่วมกัน (Interoperability) จะมีความสามารถค่อนข้างจำกัด เช่น ไม่มีการควบคุมการเข้าใช้ข้อมูล หรือ การแชร์ข้อมูลแบบมีเครื่องเซิร์ฟเวอร์กลาง อย่างไรก็ตาม เทคโนโลยีกริดก็สามารถอาศัยการทำงานของเทคโนโลยีอินเตอร์เนต และ เพียร์ทูเพียร์ได้ [9] โดยพัฒนาโปรโตคอลบางส่วนเพิ่มขึ้นมาสนับสนุนงานในส่วนการใช้ข้อมูลทำงานร่วมกัน

## 2.6 เว็บเซอร์วิสกับการพัฒนาระบบประมวลผลแบบกริด

**ตารางที่ 2.2** เว็บเซอร์วิสกับเทคโนโลยีอื่นที่นำมาพัฒนาระบบประมวลผลแบบกริด

| Model | Technology | Advantage | Disadvantage |
|---|---|---|---|
| Web Services | XML, SOAP, WSDL | Scalability, Support for large-system design | Performance |
| Datagram/steam Communication | UDP, TCP, Multicast | Low overhead | Low Level |
| Shared Memory, Multithreading | POSIX Threads DSM | High lever | Scalability |
| Data parallelism | HPF, HPC++ | Automatic parallelization | Restricted applicability |
| Message passing | MPI, PVM | High Performance | Low level |
| Object-oriented | CORBA, DCOM, Java RMI | Support for large-system design | Performance |
| Remote Procedure Call | DCE, ONC | Simplicity | Restricted applicability |
| High throughput | Condor, LSF, Nimrod | Ease of use | Restricted applicability |

ระบบการแลกเปลี่ยนข้อมูลผ่านระบบเครือข่ายด้วยเทคโนโลยีอินเตอร์เนตนั้นประสบความสำเร็จมากในทศวรรษที่ผ่านมา กุญแจสำคัญต่อความสำเร็จนี้อยู่ความง่ายของรูปแบบข้อมูลที่ส่งระหว่างกัน การรับส่งข้อมูลรูปแบบ HTML ลักษณะที่สามารถอธิบายตัวเองได้ทำให้

ระบบงานต่างๆ สามารถเข้าใจข้อมูลที่รับส่ง และแปลความหมายของข้อมูลนั้นด้วยเว็บบราวเซอร์ เว็บเซอร์วิสเป็นเทคโนโลยีที่ดำเนินรอยตามกุญแจสำคัญดังกล่าว หากแต่ใช้ XML [10] เป็นสื่อกลางรับส่งข้อมูล ด้วยความสามารถที่เหนือกว่าของ XML เว็บเซอร์วิสจึงใช้ XML เป็นตัวอธิบายข้อมูลสำหรับการติดต่อสื่อสารแบบเรียกการทำงานจากเครื่องที่อยู่บนเครือข่ายระยะไกลได้ (Remote Procedure Call: RPC) [11] ผ่านโพรโตคอล SOAP [12] อันเป็น XML รูปแบบหนึ่ง ปัจจุบันมีหน่วยงานมากมายที่ใช้เทคโนยีเว็บเซอร์วิสพัฒนาระบบงานของตนในด้านการแลกเปลี่ยนข้อมูลระหว่างหน่วยงาน ระบบประมวลแบบกริดก็เป็นเทคโนโลยีที่สามารถสร้างบนรากฐานของเว็บเซอร์วิสได้ โดยเมื่อเปรียบเทียบกับเทคโนโลยีอื่นที่นำมาสร้างระบบประมวลผลแบบกริด เว็บเซอร์วิสจะมีลักษณะตามตารางที่ 2.2

ด้วยจุดเด่นของเทคโนโลยีเว็บเซอร์วิส ที่มีความง่ายในการแลกเปลี่ยนข้อมูลนี้เอง สามารถนำมาประยุกต์ใช้กับงานรูปแบบบางอย่างภายในการประมวลแบบกริดได้ ยกตัวอย่างเช่น [7, 13, 14] เป็นการนำเว็บเซอร์วิสมาให้บริการข้อมูลสนเทศของกริดในระบบเปิด เป็นต้น ข้อมูลที่สามารถอธิบายตัวเองได้ของ XML เป็นคุณสมบัติที่เหมาะสมต่อสภาพแวดล้อมการใช้ข้อมูลที่หลากหลาย อันเป็นประโยชน์ต่อระบบ รวมทั้งโครงสร้างของระบบที่สามารถใช้ชิ้นส่วนของซอฟท์แวร์ร่วมกันก็เป็นประเด็นที่น่าสนใจต่อการพัฒนาระบบของกริด นอกจากนี้ ในส่วนของผู้ใช้ยังสามารถติดต่อกับระบบเพื่อเข้าใช้ข้อมูลสารสนเทศได้สะดวกขึ้นผ่านทางเว็บบราวเซอร์ จากเดิมที่ต้องมีแอพพลิเคชันพิเศษที่ได้มาจากการเป็นสมาชิกของระบบ

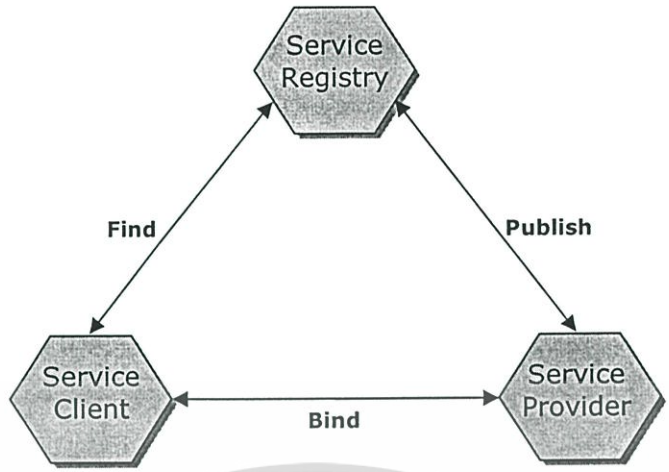
อย่างไรก็ตาม ประสิทธิภาพของระบบที่สร้างจากเทคโนโลยีเว็บเซอร์วิสนั้นเป็นประเด็นที่น่าสงสัยอยู่ เนื่องจากการรับส่งข้อมูลนั้นจะมีข้อมูลส่วนหัวที่ติดไปกับเนื้อข้อมูลด้วย อาจจะทำประสิทธิภาพของระบบด้อยลงไปได้ อีกประเด็นหนึ่งก็คือ ระบบการค้นหาบริการของเว็บเซอร์วิสในปัจจุบันนั้นยังมีความล่าช้าอยู่มาก การค้นหาบริการแต่ละครั้งนั้นต้องร้องขออินเตอร์เฟสการติดต่อ WSDL [15] ไปที่ UDDI (Universal Description, Discovery & Integration) [16] อันเป็นการควบคุมจากศูนย์กลาง เมื่อมีการร้องขอข้อมูลจำนวนมาก จะทำให้เกิดภาวะโอเวอร์โหลด ไม่สามารถตอบสนองบริการแก่ผู้ใช้บางส่วนได้ จึงน่าสนใจว่า หากระบบการค้นหานั้นมีลักษณะเป็นระบบแบบกระจาย จะสามารถเพิ่มประสิทธิภาพของระบบดังกล่าวได้มากน้อยเพียงใด สมมติฐานนี้เอง เป็นประเด็นสำคัญของวิทยานิพนธ์ชิ้นนี้

# บทที่ 3
# กระบวนค้นหาข้อมูลภายในระบบประมวลผลแบบกริด

การค้นหาข้อมูลและข่าวสารเป็นการทำงานภายในองค์กรเสมือนที่แสดงถึงประสิทธิภาพของระบบประมวลผลแบบกริด ประเด็นปัญหาก็คือ ทำอย่างไรให้ผู้ใช้บริการที่มีอยู่จำนวนมากในองค์กรเสมือนค้นหาข้อมูลหรือบริการที่ต้องการได้ภายในเวลาที่น่าพอใจ และจะมีวิธีการอย่างไรในการจัดการข้อมูลที่มีจำนวนมากอยู่ให้ได้อย่างเป็นระบบ เพื่อที่จะได้ค้นหาได้ง่ายและรวดเร็วขึ้น ในบทนี้จึงจะกล่าวถึงลักษณะสถาปัตยกรรมเชิงบริการซึ่งเป็นพื้นฐานของกลไกการค้นหาข้อมูลในกริด นิยามและวิธีการค้นหาข้อมูล เทคโนโลยีหรืองานวิจัยที่เกี่ยวข้องที่นำมาใช้ในกระบวนการค้นหาข้อมูลในระบบประมวลผลแบบกริด รวมถึงชี้ให้เห็นถึงข้อดีข้อเสียของเทคโนโลยีเหล่านั้น อันเป็นที่มาของการแก้ปัญหาของวิทยานิพนธ์นี้

## 3.1 สถาปัตยกรรมเชิงบริการ

ภายในระบบประมวลผลแบบกริด กลไกการทำงานที่สำคัญอย่างหนึ่งก็คือ กระบวนการค้นหาข้อมูล ที่มีหน้าที่จัดหาข้อมูลที่ต้องการให้แก่ผู้ใช้บริการตามความต้องการ กระบวนการค้นหาที่สมบูรณ์จะประกอบด้วย การทราบข่าวสารที่มีอยู่ในระบบ การค้นหา และการเชื่อมต่อรับส่งข้อมูล เทคโนโลยีการประมวลแบบกระจายสมัยใหม่ต่างจำเป็นต้องมีกระบวนการเหล่านี้เพื่อตอบความต้องการของผู้ใช้บริการได้ นอกเหนือจากระบบกริดแล้ว ระบบอื่นๆ อาทิ เว็บเซอร์วิส ระบบเพียร์ทูเพียร์ ต่างก็อิมพลีเมนต์กระบวนการค้นหาข้อมูลในลักษณะที่ใกล้เคียงกัน กระบวนการนี้มีพื้นฐานในการค้นหาข้อมูลในระบบกริดและระบบกระจายอื่นๆ มีพื้นฐานการทำงานมาจากสถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture) [6] โดยแรกเริ่มทีเดียวสถาปัตยกรรมแบบนี้สร้างขึ้นมาเพื่อรองรับการทำงานของเว็บเซอร์วิส ต่อมางานวิจัย OGSA [7] ได้นำเอาแนวคิดนี้มาใช้พัฒนากระบวนการค้นหาสารสนเทศในระบบกริด และมีแนวโน้มที่จะกลายเป็นมาตรฐานในการเชื่อมต่อข้อมูลภายในในระบบกริดในอนาคต ด้วยเหตุผลที่ว่าสถาปัตยกรรมเชิงบริการสนับสนุนการค้นหาแบบไดนามิก ผู้ใช้และผู้ให้บริการสามารถเข้าและออกจากระบบได้อย่างอิสระ




**รูปที่** 3.1 โครงสร้างสถาปัตยกรรมเชิงบริการ

สถาปัตยกรรมเชิงบริการมีการสร้างความสัมพันธ์ระหว่างผู้ใช้บริการ (Service Client) และผู้ให้บริการ (Service Provider) ให้สามารถเชื่อมต่อกันได้อย่างสะดวกโดยการอาศัยรีซิสทรี (Registry) โดยแต่ละส่วนประกอบจะมีความสัมพันธ์ต่อกันเพื่อการจัดการข้อมูลได้แก่ ผู้ใช้บริการเมื่อต้องการค้นหาข้อมูล จะร้องขอข้อมูลผ่านทางรีจิสทรี ซึ่งจะมีข้อมูลหรือบริการที่ผู้ให้บริการประกาศไว้ก่อนหน้านี้ เมื่อผู้ใช้บริการได้ข้อมูลที่ใช้ในการเชื่อมต่อหรือเรียกว่าพร็อกซีไปแล้วจะนำไปเชื่อมต่อกับผู้ให้บริการโดยตรง

### 3.1.1 บทบาทของส่วนประกอบในสถาปัตยกรรมเชิงบริการ

**ผู้ให้บริการ** เป็นผู้ที่ครอบครองข้อมูลหรือบริการที่พร้อมให้ผู้ใช้บริการเข้าใช้ข้อมูลได้ ในมุมของระบบกริด คำว่า "ข้อมูล" ในที่นี้หมายถึง "รีซอร์ส" หรือ "บริการ" ทั้งในส่วนของแหล่งข้อมูล หน่วยประมวลผล หรือ ข้อมูลเครือข่ายก็ได้ นอกจากนี้อาจจะเป็นแอพพลิเคชันที่ทำงานบางอย่างแทนก็ได้เช่น บริการการคำนวณทางคณิตศาสตร์เป็นต้น

**ผู้ใช้บริการ** เป็นส่วนผู้ใช้งานระบบของแอพพลิเคชันที่ต้องการใช้งานข้อมูล การติดต่อกับระบบอาจจะติดต่อผ่านเบราเซอร์หรือโปรแกรมที่ไม่มียูสเซอร์อินเตอร์เฟสก็ได้

**รีจิสทรี** เป็นตัวกลางที่เชื่อมความต้องการกับผู้ใช้บริการกับผู้ให้บริการ มีหน้าที่เก็บคำอธิยายข้อมูลที่ผู้ให้บริการได้ลงประกาศไว้ เพื่อให้ผู้ใช้บริการสามารถตรวจหาข้อมูลที่ต้องการได้ผ่านรีจิสทรี อย่างไรก็ตาม รีจิสทรีก็เป็นทางเลือกหนึ่งเพื่อใช้ในการค้นหาบริการต่างๆ ในระบบเท่านั้น เนื่องจากผู้ใช้บริการสามารถติดต่อกับผู้ให้บริการโดยตรงก็ได้ หากทราบที่อยู่และบริการที่แน่นอนของผู้ให้บริการ

### 3.1.2 ลักษณะการทำงานของสถาปัตยกรรมเชิงบริการ

ในสถาปัตยกรรมเชิงบริการมีหน้าที่การทำงานหลัก 3 อย่างที่ต้องดำเนินการคือ การประกาศตัวอธิบายบริการที่มีอยู่ในระบบ ค้นหาคำอธิบายบริการ และ เชื่อมต่อเพื่อใช้งานข้อมูลด้วยตัวอธิบายบริการนั้น โดยการทำงานแต่ละส่วนมีรายละเอียด

**การประกาศ** เพื่อที่จะทำให้การเข้าบริการสามารถทำได้ง่าย ตัวอธิบายข้อมูลจำเป็นต้องถูกประกาศไว้ในที่ซึ่งผู้ใช้บริการสามารถค้นหาได้สะดวก การประกาศนี้สามารถทำได้หลายลักษณะขึ้นอยู่กับความต้องการของแอพพลิเคชัน

**การค้นหา** เป็นกลไกการค้นหาคำอธิบายข้อมูลในระบบ ผู้ใช้บริการสามารถเรียกหาคำอธิบายข้อมูลได้โดยตรงจากผู้ให้บริการหรือค้นหาด้วยคีย์เวิร์ดไปยังรีจิสทรีก็ได้ วิธีการค้นหาสามารถทำได้ทั้งในช่วงเวลาการออกแบบ (design time) และช่วงเวลาการทำงานของแอพพลิเคชัน (runtime) ของการพัฒนา ตัวอธิบายข้อมูลจะถูกนำมาใช้เชื่อมต่อกับผู้ให้บริการเพื่อขอรับบริการต่อไป

**การเชื่อมต่อ** ข้อมูลหรือบริการจะถูกเรียกใช้จากผู้ใช้บริการ โดยใช้ข้อมูลรายละเอียดการเชื่อมต่อที่ระบุไว้ในตัวอธิบายข้อมูล ในการบอกที่อยู่ของบริการ วิธีการเชื่อมต่อ และการใช้งานข้อมูลหรือบริการนั้น

## 3.2 การค้นหาและการประกาศข้อมูลในระบบกระจาย

หน้าที่หลักของระบบจัดการข้อมูลของกริดก็คือ การจัดหาวิธีการสำหรับการค้นหาข้อมูลที่อยู่ในองค์กรเสมือน การประกาศและการค้นหาข้อมูลในระบบกระจายมีรากฐานสถาปัตยกรรมเชิงบริการ ทั้งสองต้องทำงานร่วมกันเพื่อการค้นหาข้อมูลหรือบริการที่สมบูรณ์ การประกาศข้อมูลเป็นการทำให้ระบบทราบว่าข้อมูลอยู่ที่ใดและมีสามารถเข้าใช้ประโยชน์จากมันได้อย่างไร ในส่วนการค้นหาจะเริ่มที่แอพพลิเคชันซึ่งจะค้นหาข้อมูลที่เหมาะสมในระบบกริดมาทำงานตามจุดประสงค์ของตน ระบบการค้นหาที่ดีจะพิจารณาจากโอเวอร์เฮดของการจับคู่แอพพลิเคชันกับข้อมูลและการใช้ประโยชน์จากข้อมูลได้อย่างคุ้มค่าที่สุด กระบวนการค้นหาและการประกาศที่มีใช้กันอยู่ในปัจจุบันมี 2 ประเภทคือ

Query-based ใช้ไดเร็กทอรีของระบบเครือข่ายเป็นฐานในการพัฒนา เช่นใน Globus MDS [1] จะใช้การค้นคืนข้อมูลข้ามระบบเครือข่ายไปยังไดเร็กทอรีที่ใกล้ที่สุด โดยใช้ติดต่อกับฐานข้อมูล การค้นคืนนี้มีดำเนินการขึ้นอยู่กับประเภทฐานข้อมูลว่าเป็นแบบกระจายหรือแบบศูนย์กลาง Legion [17] ทำงานแบบกระจาย แต่ในระบบส่วนใหญ่จะทำงานแบบศูนย์กลางอยู่เช่นใน Condor [18], Netsolve[19] และ Ninf[20]

Agent-based จะเป็นการส่งส่วนของโค้ดไปยังเครื่องลูกข่ายเพื่อการบอกข้อมูลสถานการณ์ทำงานของเครื่องในขณะนั้น เอเจนต์สามารถทำงานดูข้อมูลสารสนเทศในเครื่องลูกข่ายและยังสามารถส่งข้อมูลสารสนเทศดังกล่าวไปยังเอเจนต์ตัวอื่นๆ ได้อีกต่อหนึ่ง การทำงานแบบนี้สามารถทำงานได้ในลักษณะเดียวกับกับแบบ Query-based ปัจจุบันมีการนำเอาระบบแบบ Agent-based มาใช้โปรเจคของการค้นหาข้อมูลหรือบริการมากมาย อย่างเช่น 2K [21] และ [22] ส่วนใน [23] เป็นการนำเอาเอเจนต์มาใช้จัดการข้อมูลในระบบกริด

ข้อแตกต่างที่เห็นได้ชัดของทั้งสองแบบก็คือ แบบ Agent-based จะอนุญาตให้เอเจนต์ควบคุมกลไกการค้นคืนข้อมูลได้ด้วยฐานความรู้ของตัวเอง แต่ใน Query-based จะมีการกำหนดให้มีกลไกการค้นคืนแบบตามตัว จะเห็นได้ว่า Agent-based จะได้มีลักษณะความเป็นระบบกระจายมากกว่าระบบแบบ Query-based

เทคโนโลยีการค้นหาและการประกาศบริการมีหน้าที่ในการใช้งานแอพพลิเคชันและข้อมูลร่วมกันโดยลดปัญหาที่เกิดจากการปรับแต่งค่าของระบบ ซึ่งมีความสำคัญมากขึ้นเรื่อยในระบบการประมวลผลแบบกระจาย หน้าที่สำคัญของการค้นหาข้อมูลก็คือ การลงทะเบียนข้อมูล การประกาศข้อมูล การค้นหาข้อมูล และการทำงานร่วมกัน (interoperability) อย่างไรก็ตาม ก่อนที่จะกล่าวถึงวิธีการเหล่านั้น จะเป็นการกล่าวถึงเทคโนโลยีที่มีใช้กันอยู่ในระบบกระจายก่อนเพื่อเป็นแนวทางในการศึกษากลไกในการค้นหาอย่างลึกซึ้งต่อไป

### 3.2.1 เทคโนโลยีสำหรับการค้นหาและการประกาศข้อมูลในระบบกระจาย

อย่างที่กล่าวไปแล้วว่ากลไกการค้นหาและการประกาศข้อมูล เป็นกลไกสำคัญของการจัดการข้อมูลในระบบแบบกระจาย ในส่วนนี้จะเป็นการกล่าวถึงเทคโนโลยีของการค้นหาและการประกาศข้อมูลที่สำคัญ แนวคิดเป็นของเทคโนโลยีดังกล่าวจะอธิบายถึงลักษณะการทำงานที่สำคัญและปัญหาที่เกิดขึ้นเพื่อเป็นแนวทางในการพัฒนาระบบการค้นหาและการประกาศข้อมูลที่ดีต่อไป

Bluetooth

Bluetooth [24] เป็นเทคโนโลยีที่เกิดจากความร่วมมือของบริษัทไอบีเอ็ม อินเทล โนเกีย อิริคสัน และโตชิบา ที่สร้างโปรโตคอลที่สามารถสื่อสารข้อมูลระหว่างแอพพลิเคชันและข้อมูลหรือบริการผ่านทางสัญญาณวิทยุและโปรโตคอลการสื่อสารข้อมูลได้

Bluetooth มีโปรโตคอลที่ชื่อว่า Service Discovery Protocol (SDP) สำหรับการค้นหาข้อมูลที่อนุญาตให้แอพพลิเคชันของผู้ใช้บริการสามารถค้นหาบริการที่มีอยู่ ซึ่งจัดให้โดยแอพพลิเคชันของผู้ให้บริการ โดยมองแอพพลิเคชันเหล่านั้นเป็นเสมือนแอตทริบิวส์หนึ่ง แอตทริบิวส์เหล่านี้จะประกอบด้วยประเภทหรือคลาส และ วิธีการใช้บริการของข้อมูลเหล่านั้น

Jini

Jini [25] ของบริษัทซันไมโครซิสเต็มเป็นระบบกระจายที่มีแนวคิดในการรวบรวมข้อมูลของกลุ่มของผู้ใช้บริการ และจัดหาข้อมูลเพื่อตอบสนองความต้องการของผู้ใช้เหล่านั้น จุดมุ่งหมายของ Jini ก็เพื่อการสร้างความยืดหยุ่นต่อการใช้บริการระบบเครือข่าย ง่ายต่อการจัดการข้อมูลของผู้ใช้ และหน่วยประมวลผลในฝั่งไคลเอ็นต์ โดยการทำงานจะเน้นไปที่การทำให้ข้อมูลในระบบมีลักษณะไดนามิกส์ กลุ่มการใช้ข้อมูลสามารถเพิ่มหรือลบข้อมูลจากระบบได้อย่างอิสระ

หลักการทำงานของ Jini ก็คือ การรวมเอาโปรโตคอลการค้นหา การจอยน์ข้อมูล (join) และการพิจารณา (lookup) ข้อมูลเข้าด้วยกัน คู่ของการค้นหาและการจอยน์จะทำงานเมื่ออุปกรณ์ในระบบมีการเชื่อมต่อเข้าใช้บริการ การค้นหาจะทำงานเมื่อมีการร้องขอการใช้บริการข้อมูล การจอยน์จะเกิดขึ้นเมื่อมีการระบุตำแหน่งของข้อมูลเรียบร้อยแล้วและต้องการจะรวบรวมข้อมูลเข้าด้วยกัน การพิจารณาข้อมูลจะเกิดขึ้นเมื่อไคลเอ็นต์หรือผู้ใช้บริการต้องการที่จะระบุตำแหน่งของข้อมูลและต้องการที่จะเรียกใช้บริการจากมันด้วยอินเตอร์เฟสประเภทต่างๆ

Salutation

Salutaion [26] เป็นสถาปัตยกรรมที่สร้างขึ้นมาเพื่อแก้ปัญหาการค้นหาและการใช้บริการอุปกรณ์ต่างๆ ในสภาพแวดล้อมที่มีการเชื่อมต่อและเคลื่อนที่มาก สถาปัตยกรรมแบบนี้มีวิธีการมาตรฐานสำหรับแอพพลิเคชัน ข้อมูล รวมทั้งอุปกรณ์ต่างๆสามารถประกาศความสามารถของตนให้ผู้อื่นได้รับรู้ และสามารถค้นหาเพื่อใช้ประโยชน์จากมันได้ นอกจากนี้ Salutation ยังสามารถค้นหาแอพพลิเคชัน ข้อมูล และอุปกรณ์ที่มีลักษณะเฉพาะเจาะจงและสามารถนำสิ่งเหล่านั้นมาทำงานร่วมกับแอพพลิเคชันที่รันอยู่แล้วได้อีกด้วย

Service Location Protocol (SLP)

โปรโตคอลการระบุตำแหน่งบริการ SLP [27] เป็นเฟรมเวิร์กสำหรับการค้นหาบริการข้อมูลในระบบเครือข่ายที่สามารถรองรับการใช้บริการได้มาก การใช้ SLP คอมพิวเตอร์ที่ใช้อินเตอร์เนตจำเป็นต้องปรับค่าบริการของเครือข่ายเพียงเล็กน้อยสำหรับการรองรับแอพพลิเคชันของระบบเครือข่าย สิ่งที่ได้มาก็คือเครื่องคอมพิวเตอร์นั้นจะมีความพอร์ทเทเบิล (Portable) มากขึ้น

SLP สนับสนุนการใช้งานระบบเครือข่ายของผู้ใช้บริการ แอพพลิเคชันของผู้ใช้จะถูกจัดการโดยยูสเซอร์เอเจนต์ ข้อมูลและบริการจะถูกประกาศด้วยเซอร์วิสเอเจนต์ สุดท้ายการค้นหาข้อมูลจะใช้ไดเร็กทอรีที่มีความสามารถรองรับการใช้บริการได้มาก

Universal Plug and Play (UPnP)

UPnP [28] เป็นสถาปัตยกรรมสำหรับการเชื่อมต่อแบบเพียร์ทูเพียร์สำหรับเครื่องพีซีในรูปแบบต่างๆ ไม่ว่าจะเป็นแอพพลิชันที่ชาญฉลาด และ ระบบเครือข่ายไร้สาย UPnP นั้นเป็นระบบแบบกระจาย ใช้ระบบเครือข่ายแบบเปิดซึ่งเอาข้อดีจากเครือข่าย TCP/IP และเว็บมาใช้สำหรับการควบคุมและการรับส่งข้อมูลในอุปกรณ์เครือข่ายในบ้าน ในออฟฟิศ และทุกๆที่ที่เชื่อมต่อกันอยู่

ใน UPnP ใช้ Simple Service Discovery Protocol (SSDP) สำหรับการค้นหาข้อมูลที่มีอยู่ในระบบ SSDP ระบุวิธีการในส่วนของการควบคุมจุดเชื่อมต่อของข้อมูล และวิธีการประกาศข้อมูลที่มีอยู่ไปในระบบเครือข่าย SSDP จะมีวิธีการลดโอเวอร์เฮดของระบบเมื่อมีการใช้งานฟังก์ชันของ SSDP เพียงฟังก์ชันเดียว

### 3.2.2 การลงทะเบียนข้อมูล

เมื่อมีข้อมูลใหม่เข้าสู่ระบบจะต้องผ่านกระบวนการเพื่อทำให้ระบบทราบว่า มีข้อมูลนี้อยู่ในระบบจริงสามารถติดต่อสื่อสารกับคอมโพเนนต์อื่นในระบบได้ กระบวนการนี้เรียกว่า การลงทะเบียนข้อมูล

ใน Jini การลงทะเบียนข้อมูลและการค้นหาข้อมูล เริ่มแรก ข้อมูลหรือไคลเอ็นต์จะต้องหาที่อยู่ของเซิร์ฟเวอร์ที่ให้บริการข้อมูลจำนวนหนึ่งหรือมากกว่า ด้วยการใช้มัลติแคสรีเควสโปรโตคอล การร้องขอจะสิ้นสุดด้วยการร้องขอของยูนิแคสดิสโคเวอรีโปรโตคอลซึ่งไคลเอ็นต์และเซอร์วิสจะสื่อสารกันด้วยบริการการค้นหาข้อมูลแบบเฉพาะ

ส่วน SLP มีความแตกต่างกับ Jini โดยที่ SLP จะทำงานโดยไม่ผ่านไดเร็กทอรีเซิร์ฟเวอร์ แต่จะใช้ไดเร็กทอรีของเอเจนต์ในการอธิบายข้อมูลซึ่งจะเพิ่มประสิทธิภาพได้อีกเล็กน้อย ด้วยการลดการกระจายการค้นหาไปยังเครือข่ายและลดการใช้แบนด์วิดธ์ของเครือข่ายทั้งหมด ในแอกทีฟไดเรกทอรี เซอร์วิสเอเจนต์และยูสเซอร์เอเจนต์จะทำมัลติแคสรีเควสหรือใช้ DHCP ไปค้นหายังไดเรกทอรีเอเจนต์ เมื่อไดเรกทอรีเอเจนต์พร้อม เซอร์วิสเอเจนต์และยูสเซอร์เอเจนต์จะใช้การสื่อสารแบบยูนิแคสในการลงทะเบียนข้อมูล และค้นหาข้อมูลที่ต้องการตามลำดับ ถ้าหากไม่มีไดเรกทอรีเอเจนต์ ยูสเซอร์เอเจนต์และเซอร์วิสเอเจนต์จะทำมัลติแคสเพื่อค้นหา และตอบกลับด้วยยูนิแคสโดยตรงกับเซอร์วิสที่ต้องการจะค้นหามาได้ วิธีการลักษณะนี้จะมีการใช้แบนวิดธ์ที่สิ้นเปลือง แต่ก็เหมาะกับระบบเครือข่ายขนาดเล็กที่มีสมาชิกไม่มากนัก

### 3.2.3 การประกาศข้อมูล

หลังจากเข้าสู่ระบบ องค์ประกอบของระบบก็จะทำหน้าที่เป็นผู้ให้บริการข้อมูลที่ต้องประกาศให้หน่วยอื่นๆ ของระบบรับรู้ว่าตนมีข้อมูลหรือบริการใดให้ใช้บ้าง วิธีการนี้เรียกว่า การประกาศข้อมูล

ใน UPuP ไม่มีกระบวนการลงทะเบียนข้อมูล อย่างไรก็ตาม อุปกรณ์ต้องประกาศตนให้ระบบทราบว่า ณ ขณะนั้นอยู่ในระบบด้วยการทำมัลติแคสเมสเซส “alive” ไปยังจุดควบคุมของระบบ เมื่อต้องการจะยกเลิกการให้บริการข้อมูลแก่ระบบ อุปกรณ์จะส่งเมสเซส “byebye” ออกมา ซึ่งโปรโตคอล SSDP ที่ใช้อยู่ใน UPnP ข้อมูลหรือบริการแต่ละตัวจะมีตัวระบุข้อมูลหรือ ID 3 อย่าง คือ ชื่อข้อมูล ประเภทข้อมูล และที่อยู่ ข้อมูลเหล่านี้จะถูกมัลติแคสเพื่อทำกระบวนการประกาศตน

สำหรับ Jini จะใช้ Remote Method Invocation (RMI) ของจาวาสำหรับการเชื่อมต่อทุกลักษณะที่เกี่ยวข้องกับผู้ใชบริการและข้อมูล รวมทั้งลุกอัพเซิร์ฟเวอร์หลังจากการค้นหาลุกอัพเซิร์ฟเวอร์ได้แล้ว ใน Jini จะทำงานเกี่ยวข้องกับพร็อกซี่หรือการควบคุมออบเจ็กจากระยะไกลกับตัวแทนข้อมูล (service instance) ในการประกาศข้อมูลที่มีอยู่จะมีการลงทะเบียนออบเจ็กของมันในลุกอัพเซิร์ฟเวอร์อย่างน้อยหนึ่งตัว

### 3.2.4 การค้นหาข้อมูล

เมื่อองค์ประกอบของระบบต้องการจะค้นหาข้อมูลที่มีอยู่ในระบบ องค์ประกอบนั้นจะต้องทำหน้าที่เป็นไคลเอ็นต์ร้องขอข้อมูลผ่านไปยังรีจิสทรี จากนั้นรีจิสทรีจะส่งผลลัพธ์ของการค้นหากลับมายังไคลเอนต์เพื่อใช้ติดต่อกับผู้ให้บริการต่อไป กระบวนการทั้งหมดเป็นการทำงานระดับเคอร์เนล เรียกว่า การค้นหาข้อมูล

ใน Bluetooth ซึ่งเป็นระบบคลื่นวิทยุไร้สาย ดังนั้นจึงไม่มีกลไกการลงทะเบียนและการประกาศข้อมูล อย่างไรก็ตาม SDP ของ Bluetooth มี API สำหรับระบุตำแหน่งของอุปกรณ์ในระยะที่สามารถค้นหาข้อมูลได้ นอกจากนี้ SDP ยังสนับสนุนการจำกัดจำนวนของข้อมูลที่จะค้นหาได้ แอพพลิเคชันของผู้ใช้บริการจะใช้ API นี้สำหรับการค้นหาข้อมูลที่มีอยู่ในระบบ โดยระบุประเภทของข้อมูลซึ่งแต่ละตัวจะมีลักษณะแตกต่างกัน (unique identifier) และเปรียบเทียบด้วยการแมชชิ่งแอตทริบิวต์

ในส่วน Salutation จะมีกลไกที่ทำหน้าที่เป็นโบรกเกอร์ที่ช่วยไคลเอนต์หาข้อมูลที่ต้องการและช่วยในการลงทะเบียนข้อมูลที่มีอยู่แก่ระบบ ไคลเอนต์สามารถใช้ slmSerarchCapability() เพื่อพิจารณาว่า Salutation managers มีฟังก์ชันยูนิตที่ต้องการลงทะเบียนอยู่หรือไม่ เมื่อฟังก์ชันยูนิตนั้นถูกค้นหาก็สามารถใช้ slmQueryCapacity() พิจารณาได้ว่าฟังก์ชันยูนิตนั้นตรงความต้องการหรือไม่

### 3.2.5 การทำงานร่วมกัน

เมื่อส่วนประกอบในระบบต้องการค้นหาองค์ประกอบของผู้ให้บริการอื่นที่มีอยู่ คำถามที่ตามมาคือส่วนประกอบทั้งสองสามารถทำงานร่วมกันสอดคล้องกันหรือไม่อย่างไร ซึ่งก็คือปัญหาที่เรียกว่า การทำงานร่วมกันนั่นเอง

ใน Jini เมื่อต้องการใช้งานข้อมูล อุปกรณ์จะให้ความปลอดภัยกับตัวแทนข้อมูลเสียก่อน ในมุมของไคลเอ็นต์ ตำแหน่งที่อยู่ของผู้ให้บริการนั้นจะถือว่าไม่สำคัญและจะมองไม่เห็น เนื่องจากออบเจ็กจะเก็บค่าข้อมูลที่อยู่และข้อมูลบริการไว้ทำงานเฉพาะกับโปรโตคอลที่จำเป็นเท่านั้น

ใน Salution ตัว Salutation manager จะทำงานในลักษณะเดียวกับลุกอัพเซิร์ฟเวอร์ใน Jini แต่จะสามารถจัดการเชื่อมต่อระหว่างไคลเอ็นต์กับข้อมูลได้โดยตรง หลังจากที่มีการประกาศการเชื่อมต่อแล้ว ตัว Salution manager สามารถทำงานในหลายลักษณะทั้งในส่วนที่เกี่ยวข้องและไม่เกี่ยวข้องกับสายธารข้อมูล

ในส่วน SDP ของ Bluetooth จะมีการทำงานที่แตกต่างจากเทคโนโลยีการค้นหาระดับสูงอย่าง Jini โดยจะไม่มีกลไกเกี่ยวกับกระบวนการค้นหา แต่จะมีการใช้โปรโตคอลเฉพาะที่มีระดับสูงกว่าแทน อย่างไรก็ตาม ตัว SDP จะมีการระบุแอตทริบิวส์มาตรฐาน คือ ProtocolDescriptionList ซึ่งรวบรวมเอาโปรโตคอลที่เหมาะสมไว้เพื่อให้ผู้ใช้บริการเลือกใช้สำหรับการติดต่อกับข้อมูลในระบบแทน

## 3.3 ประเด็นปัญหาของระบบการค้นหาบริการในระบบประมวลผลแบบกริด

การจัดการข้อมูลในระบบกริดนั้นมีลักษณะใกล้เคียงกับระบบแบบกระจาย กล่าวคือ การทำงานหลักของการจัดการข้อมูลในระบบประมวลผลแบบกริด ก็คือ การทำให้ข้อมูลในระบบสามารถถูกค้นหาและใช้ประโยชน์ได้จากแอพพลิเคชันของกริด กระบวนการทั้งหมดประกอบด้วย การค้นหา และ การประกาศข้อมูล การค้นหาข้อมูลนั่นเริ่มต้นที่แอพพลิเคชันของผู้ใช้บริการร้องขอข้อมูลที่ต้องการไปยังระบบ ส่วนการประกาศข้อมูลจะเริ่มต้นที่ข้อมูลจะประกาศข้อมูลที่ตนมีอยู่ให้กับระบบได้รับทราบ ระบบที่ดีจะต้องมีกลไกการทำงานทั้งสองที่สัมพันธ์กัน ไม่ก่อให้เกิดข้อมูลส่วนเกินแก่ระบบมากนัก

ประเด็นปัญหาของระบบการค้นหาคือ มีผู้ใช้บริการและข้อมูลอยู่ในระบบเป็นจำนวนมาก การที่จะค้นหาบริการในแต่ละครั้งจะใช้เวลานาน กระบวนการค้นหาที่ใช้กันอยู่ในปัจจุบันนี้เป็นกระบวนการค้นหาที่ใช้อยู่ในระบบแบบศูนย์กลาง ได้แก่ Condor matchmaker [18] และ MDS [29] ที่ใช้ LDAP [30] โดยข้อมูลทุกอย่างจะถูกเก็บไว้ที่เซิร์ฟเวอร์ที่ให้บริการเพียงแห่งเดียว การทำงานแบบนี้ใช้ได้ดีกับระบบแบบแลน (LAN) แต่เมื่อนำมาใช้กับองค์กรเสมือนที่มีลักษณะกระจายตามที่ต่างๆ จะเกิดปัญหาเรื่องประสิทธิภาพ และความเสียหายจากจุดเดียวทำให้ระบบไม่สามารถทำงานได้ (Single point of Failure) ต่อมาได้มีการพัฒนา MDS-2 [13] ให้มีลักษณะการทำงานแบบกระจายที่ฝากข้อมูลไว้ยังเซิร์ฟเวอร์ในที่ต่างๆ ทำให้สามารถรองรับการทำงานที่มีลักษณะแบบกระจายขององค์กรเสมือนได้ดียิ่งขึ้น

เมื่อไม่นานมานี้ได้มีการวิจัยพบว่า กระบวนค้นหาข้อมูลภายในระบบกริดที่รองรับผู้ใช้ได้จำนวนมากสามารถอิมพลีเมนต์ด้วยเทคโนโลยีแบบเพียร์ทูเพียร์ ผู้ใช้ระบบทุกคนในองค์กรเสมือนมีความสามารถในการควบคุมข้อมูลของตนเองได้อย่างเต็มที่ ผู้ใช้ทำหน้าที่เป็นเจ้าของเซิร์ฟเวอร์หรือเพียร์ของตนและอนุญาตให้ผู้อื่นเข้าใช้บริการ แต่ละเพียร์จะสามารถเข้าสู่ระบบหรือออกจากระบบได้อย่างอิสระ อย่างไรก็ดี ข้อมูลในระบบกริดก็ยังมีความแตกต่างกับข้อมูลในเพียร์ทูเพียร์อยู่ เนื่องจากข้อมูลมีความหมายมากกว่าไฟล์ข้อมูล แต่รวมถึงข้อมูลด้านเครือข่าย และ ข้อมูลหน่วยประมวลผลที่สามารถจัดระเบียบตัวเองตามความเหมาะสมได้ ดังนั้น ในส่วนกลไกการค้นหาที่จะสร้างขึ้นสำหรับระบบกริดจะต้องเพิ่มเติมนิยามในส่วนข้อมูลเครือข่าย และ ข้อมูลหน่วยประมวลผลด้วย โดยจะต้องสามารถค้นหาข้อมูลลักษณะนี้ได้ "ค้นหาหน่วยประมวลผลที่มีความเร็ว 1 GHz หรือมากกว่า"

### 3.3.1 การค้นหาข้อมูลในระบบประมวลผลแบบกระจาย

เมื่อไม่นานมานี้มีความพยายามในการพัฒนาวิธีการค้นหาข้อมูลที่มีประสิทธิภาพมากมาย อาทิ CAN [31] Chord [32] และ Pastry [33] โดยนำมาใช้ในระบบแบบเพียร์ทูเพียร์ เป้าหมายก็คือ การหาตำแหน่งชิ้นข้อมูลจากคีย์ หรือชื่อซึ่งถูกนำมาทำเป็นคีย์เพื่อค้นหาในส่วนของข้อมูลในระบบกริด วิธีการแบบนี้นำมาใช้ค้นหาชื่อของไฟล์ โดยใช้วิธีการทำอินเด็กและเลือกเส้นทางค้นหาข้อมูลเพื่อลดขั้นตอนในกระบวนการหาข้อมูลซึ่งถือว่าเป็นวิธีการที่ดี สามารถนำมาใช้ในพื้นฐานในการค้นหาข้อมูล และเก็บรักษาข้อมูลอย่างมีประสิทธิภาพในระบบกริดได้ อย่างไรก็ตาม ประเด็นงานวิจัยของวิธีการค้นหาจะมุ่งไปที่ การหาลักษณะขององค์กรเสมือนที่เอื้อต่อการค้นหามากกว่าการหาวิธีการค้นหา เช่น การจัดรูปแบบการแชร์ข้อมูลให้สามารถใช้งานและค้นหาได้ง่ายจากที่ต่างๆ ในระบบ.

แม้ว่าในระยะแรกจะมีองค์กรเสมือนที่ใช้งานกันอยู่จริงเพียง 2-3 แห่ง แต่ต่อไปในอนาคตกลุ่มการทำงานด้านวิทยาศาสตร์และเทคโนยีมีแนวโน้มที่จะเติบโตกลายไปเป็นกลุ่มที่ใหญ่ขึ้น มีผู้เข้าร่วมมากขึ้น ลักษณะแบบนี้ถือเป็นโลกเครือข่ายอย่างย่อมทีเดียว โลกเครือข่ายแบบย่อมนี้จะมีลักษณะแตกต่างจากเวิร์ลไวด์เว็บหรือระบบเครือข่ายทั่วไป โดยมีลักษณะพื้นฐาน 2 ประการคือ 1) มีจำนวนเส้นทางที่เชื่อมถึงกันน้อยที่สุด และ 2) เป็นกลุ่มที่ไม่ขึ้นตามขนาดของเครือข่ายแต่จะหมายถึงจำนวนโหนดที่เชื่อมต่อกันอยู่

โลกขนาดเล็กนี้ประกอบด้วยเครือข่ายย่อยที่เชื่อมกันอยู่อย่างหลวม มีความพยายามที่จะเชื่อมต่อระบบนี้ด้วยกราฟ [34] ผู้ใช้บริการแต่ละรายในกริดจะเป็นโหนดในกราฟ และจะมีลีฟเชื่อมต่อระหว่างสองโหนดหากมีการแชร์ข้อมูลอย่างน้อยหนึ่งไฟล์ร่วมกันในช่วงเวลาหนึ่ง วิธีการแบบนี้จะไม่กระทบต่อระบบเครือข่ายพื้นฐาน อย่างไรก็ตามวิธีการแบบนี้ก็ยังมีข้อสงสัยด้านความเร็วในการสื่อสารข้อมูลอยู่ แนวคิดที่เป็นกุญแจสำคัญก็คือ ชนิดของระบบเครือข่ายที่มี

ลักษณะเหมาะสมต่อการพัฒนาอัลกอริธึมและโปรโตคอลเพื่อการจัดการข้อมูลอย่างมีประสิทธิภาพ ชนิดของเครือข่ายที่เหมาะสมจะต้องตอบความต้องการของระบบการค้นหาที่มีประสิทธิภาพในโลกขนาดเล็กของกริดได้ วิธีการในการค้นหาที่สร้างขึ้นมาจะต้องลดการเชื่อมต่อเพื่อสอบถามข้อมูลในระบบกริดได้จำนวนหนึ่ง

คำถามสำคัญก็คือ โปรโตคอลที่สร้างขึ้นมาสำหรับการค้นหาข้อมูลในระบบกริดนั้นลักษณะเครือข่ายจะต้องมีจัดการตัวเองได้อย่างอัตโนมัติซึ่งจะมีผลต่อคุณสมบติขององค์กรเสมือน ประเด็นงานวิจัยของโปรโตคอลใหม่นี้จะเน้นไปที่จำนวนการเชื่อมต่อถามข้อมูลในเครือข่ายต่อการข้อมูลร้องข้อมูล 1 ครั้ง (hops per request) [35]

ปัจจุบันงานวิจัยเพื่อตอบปัญหาดังกล่าวยังมีการแก้ปัญหาแบบศูนย์กลางอยู่ซึ่งทำให้ปัญหาเรื่องการรองรับการใช้บริการได้น้อย (Scalability) ในส่วนระบบการค้นหารูปแบบใหม่ก็กำลังมีการพัฒนาอยู่ [36] รวมทั้งงานวิจัยนี้ด้วย โดยมีลักษณะการจัดการในรูปแบบกระจายทั้งในส่วนของโปรโตคอลและอัลกอริธึมที่มีความสามารถในการรองรับบริการได้มากขึ้น

### 3.3.2 การค้นหาข้อมูลในระบบกริด

ระบบการบอกที่อยู่ของข้อมูลที่ได้กล่าวไว้ในส่วนที่แล้วใช้ ชื่อเป็นคีย์ในการค้นหา โดยมีเงื่อนไขว่าข้อมูลจะต้องถูกกำหนดค่าที่แสดงตัวที่มีลักษณะเฉพาะ (unique identifier) เพื่อใช้ในการค้นหาและการเลือกเส้นทางในระบบ วิธีการดังกล่าวยังไม่เพียงพอต่อการค้นหาข้อมูลในระบบกริดเท่าใดนัก เนื่องจากบริการค้นหาข้อมูลต้องตอบคำถามเป็นเซตของแอตทริบิวส์ตามความต้องการของผู้ใช้บริการ เมื่อมีการร้องขอบริการด้วยแอตทริบิวต์ต่างๆ สามารถทดแทนด้วยการนำมาแมพรวมกันเพื่อเป็นคีย์ในการค้นหาและการเลือกเส้นทางได้ อย่างไรก็ตามวิธีการดังกล่าวก็มีผลต่อคุณสมบัติของความไม่หยุดนิ่งของข้อมูล ทำให้การค้นหาสามารถทำได้จำกัด ดังนั้น การค้นหาบริการโดยทั่วไปไม่สามารถนำมาใช้ค้นหาข้อมูลในระบบกริดได้อย่างมีประสิทธิภาพมากนัก

ในช่วงแรกของการพัฒนามีการประเมินวิธีการร้องขอและส่งต่อเมสเซสเพื่อค้นหาข้อมูลในแต่ละเพียร์ ในหลายลักษณะ โดยอาศัยข้อมูลสถิติประเมินการกระจายของข้อมูลในเพียร์ต่างๆ ในงานวิจัยนี้มีการระบุปัญหาพบ ปัญหาดังกล่าวเกี่ยวข้องกับวิธีการที่เหมาะสมต่อการวิธีการร้องขอและส่งต่อข้อมูลเพื่อค้นหาข้อมูลในระบบกริด

เงื่อนไขที่สำคัญอีกประการหนึ่งก็คือ โมเดลของข้อมูลแบบไหนที่เหมาะสมต่อภาษาสำหรับการค้นคืนข้อมูล ที่สามารถนำมาใช้บนระบบกริดได้อย่างเหมาะสม ปัจจุบันบริการสารสนเทศกริด (Grid Information Services) สร้างอยู่บนสถาปัตยกรรมแบบกระจาย มีการนำเสนอการอธิบายข้อมูลแบบลำดับชั้น โดยใช้ลำดับชั้นข้อมูลนี้มาอิมพลีเมนต์บริการไดเรกทอรี (Directory Services) ซึ่งถือว่ามีประสิทธิภาพยอมรับได้และน่าสนใจ อีกทางเลือกหนึ่งก็คือ การใช้

ใช้โมเดลข้อมูลแบบสัมพันธ์ (Relational data model) ที่มีความยากต่อการใช้งานมากกว่า รองรับการใช้งานได้น้อยกว่า แต่น่าสนใจตรงที่รองรับภาษาในการค้นคืนข้อมูล ในทางปฏิบัติ การสร้างระบบสารสนเทศกริดจะเกี่ยวข้องกับฐานข้อมูลแบบกระจาย ปัญหาที่ตามมาก็คือ การตอบคำถามเกี่ยวกับบูรณาการของข้อมูล (Consistency) ที่เกี่ยวข้องกับการเขียนและการอ่านข้อมูลในแหล่งเก็บข้อมูล

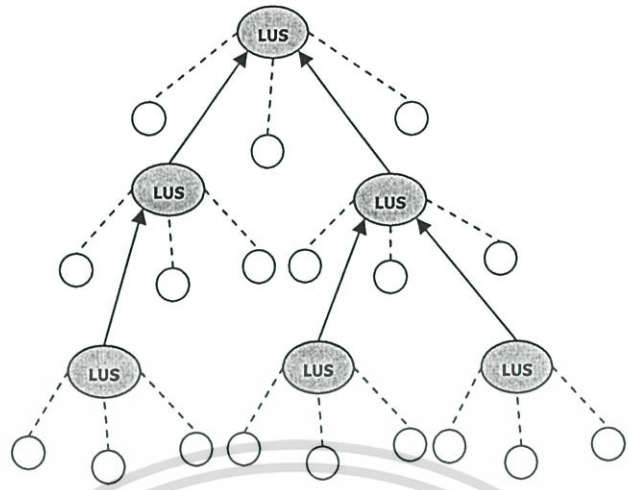
## 3.4 งานวิจัยที่เกี่ยวข้อง

มีงานวิจัยมากมายที่เกี่ยวข้องกับระบบการค้นหาที่มีขนาดใหญ่อย่างระบบประมวลผลแบบกริด โดยมีประเด็นการวิจัยที่มุ่งไปที่การใช้ระบบแบบกระจายเข้ามาแทนที่ระบบแบบศูนย์กลาง เพื่อแก้ปัญหาด้านความเสียหายจากศูนย์กลาง (single point of failure) ในงานวิจัยระบบแบบกระจายดังกล่าวก็ได้แบ่ง ประเด็นการวิจัยออกเป็นระบบแบบแฟลต และ ระบบแบบลำดับชั้น

ในงานวิจัย [35] ได้นำเสนอการค้นหาด้วยระบบกระจายแบบแฟลต พัฒนาการค้นหามาจากวิธี Flooding [37] มีการเพิ่มความสามารถในลักษณะการกำหนดค่าได้เอง (Self-Configuring) ของระบบ โดยถือว่าข้อมูลเป็นโหนดหนึ่งในระบบ และผู้ใช้บริการจะเชื่อมต่อกับโหนดระดับโลคอลที่ให้บริการอยู่ เมื่อมีการร้องขอใช้งานข้อมูล ระบบจะส่งเมสเซสค้นหาไปยังโหนดต่างๆ โดยกำหนดค่าเริ่มต้นแก่ค่า TTL (time-to-live) ให้แก่เมสเซสนั้น การค้นหาจะสิ้นสุดเมื่อค่านี้เหลือ 0 แต่ละโหนดจะมีการเชื่อมต่อไปยังโหนดอื่น

งานวิจัยนี้มีกฎการส่งต่อ 4 ประเภทที่รวบรวมเอาการสุ่มข้อมูล การเลือกเส้นทาง และการเรียนรู้ระบบเข้าด้วยกัน ข้อเสียของวิธีการดังกล่าวคือ ยังไม่สามารถนำมาใช้ในการค้นหาข้อมูลได้ดีเท่าที่ควร เนื่องจากยังมีโอกาสส่งข้อมูลไปยังโหนดที่ไม่สอดคล้องกับการค้นหา ไม่สามารถรับประกันว่าจะมีการเชื่อมต่อไปยังโหนดจำนวนเท่าไร บางครั้งเมสเซสก็หมดอายุก่อนที่จะค้นหาข้อมูลพบเสียอีก ผลการทดลองในงานวิจัยดังกล่าวแสดงให้เห็นว่าเมื่อจำนวนโหนดเพิ่มมากขึ้น จำนวนการเชื่อมต่อเพื่อการค้นหาก็เพิ่มตามไปด้วย โดยจำเป็นต้องมีการเชื่อมต่อเพื่อค้นหาข้อมูลเป็นจำนวนร้อยๆ ครั้ง เมื่อจำนวนโหนดมีค่าเป็น 10,000 หรือมากกว่า แต่อย่างรก็ดี การกำหนดค่าเริ่มต้นให้แก่ TTL = 100 จะทำให้ความน่าจะเป็นของเมสเซสที่จะเดินทางไปถึงในระบบที่มีขนาด 5,000 โหนด มีค่าเท่ากับ 80 %

สำหรับ Jini [25] ก็มีกลไกการทำงานของระบบการค้นหาที่คล้ายคลึงกัน ทั้งไคลเอนต์และเซอร์วิสใน Jini สามารถที่จะค้นหาได้โดยผ่านเซอร์วิสรีจิสเตอร์ ใน Jini มีการใช้งานโปรโตคอล 2 แบบก็คือ ยูนิแคสโปรโตคอลที่ใช้ URL และ IP ที่ทราบแน่นอน และใช้โปรโตคอลมัลติแคสที่จะมีการส่งแพ็กเก็ตในการค้นหาไปยังสมาชิกของระบบทุกราย โดยใช้ค่าเริ่มต้นของ TTL = 15 ซึ่งเป็นค่ารัศมีการค้นหาที่เมสเซสสามารถเดินทางไปถึงและตอบกลับได้




**รูปที่** 3.2 โครงสร้างการเชื่อมต่อแบบลำดับชั้นของลุกอัพเซอร์วิสแบบลำดับชั้น

งานวิจัยหลายชิ้น [38, 39] ได้นำเอา Jini มาใช้สำหรับระบบการค้นหาของตนโดยมีการแก้ปัญหาโดยมีการเชื่อมต่อตัวลุกอัพเซอร์วิสเป็นแบบลำดับชั้น ดังรูป 3.2 โครงสร้างลำดับชั้นที่สร้างขึ้นมีจุดประสงค์เพื่อแก้ปัญหาเรื่อง Scalability ของระบบการค้นหาแบบแฟลต อย่างไรก็ดี สำหรับการค้นหาข้อมูลในรูปแบบครบถ้วน การเชื่อมต่อแบบนี้จะเกิดปัญหาเกี่ยวกับโอเวอร์เฮดสำหรับการเชื่อมต่อสอบถามข้อมูลอยู่ดี เนื่องจากการส่งข้อมูลเพื่อค้นหาแต่ละครั้ง จำเป็นต้องส่งข้อมูลไปค้นหายังโหนดต่างๆ ในระบบเกือบทั้งมหด นอกจากนี้ การค้นหาแต่ละครั้งจะต้องใช้เวลานาน อันมีผลมากจากเมสเซสต้องเดินทางไปยังที่ต่างๆ เพื่อตรวจสอบข้อมูลในระบบนั่นเอง สิ่งเหล่านี้เป็นปัญหาสำคัญที่ต้องได้รับการแก้ไขของระบบการค้นหาที่มีจำนวนสมาชิกและจำนวนข้อมูลมากอย่างระบบกริด

ในงานวิจัย [40] เป็นการพัฒนาระบบการค้นหาข้อมูลที่ใช้คุณภาพของบริการ (Quality-of-Service: QoS) เป็นเงื่อนไขสำหรับแก้ปัญหาการเชื่อมต่อที่มากเกินไปในระบบการค้นหาข้อมูลระบบกริด ในงานวิจัยดังกล่าวมีการจัดโครงสร้างของเซิร์ฟเวอร์เพื่อการค้นหาเป็นแบบลำดับชั้น โดยมีกุญแจสำคัญคือ การเพิ่มความสามารถในการตอบสนองต่อ QoS ให้ไคลเอนต์ และการเก็บผลลัพธ์ของการค้นหาไว้ในเซิร์ฟเวอร์เพื่อการค้นหาในแต่ละลำดับชั้น อย่างไรก็ตาม วิธีการแบบนี้ก็ยังมีลักษณะการค้นหาที่มีการสูญเสียข้อมูล (Lossy) ซึ่งแตกต่างจากงานวิจัยในวิทยานิพนธ์นี้ที่มีลักษณะการทำงาน 2 แบบ คือ แบบครบถ้วนและแบบที่มีการสูญเสียข้อมูล นอกจากนี้ ระบบการค้นหาในงานวิจัยนี้ยังจำเป็นต้องสูญเสียค่าใช้จ่ายของการตรวจสอบ QoS ให้แก่ ไคลเอนต์ เซอร์วิสโปรไวเดอร์ และดิสคัฟเวอรีเซอร์เวอร์

ในงานวิจัย [23] เป็นการนำเอาเทคโนโลยีของซอฟท์แวร์เอเจนต์มาใช้เพิ่มความเร็วในระบบการค้นหาของกริด โดยมีการใช้แคชและ Knowledge ในการเก็บความรู้ เพื่อลดจำนวนการค้นหาข้อมูลลง ในงานวิจัยนี้มีประสิทธิภาพในการค้นหาข้อมูลสูง มีการส่งข้อมูลไปค้นหายังโหนดอื่นๆ น้อย มีผลมาจากมีการเก็บข้อมูลความรู้ไว้ในเอเจนต์แต่ละโหนด ผู้ใช้บริการสามารถค้นหาข้อมูลที่ต้องการได้เกือบจะทันที แต่ข้อเสียก็คือ มีการใช้พื้นที่ในการเก็บ Knowledge สูงนี่เอง โดยเอเจนต์แต่ละตัวจะเก็บข้อมูลข้อมูลเกือบทั้งระบบ ซึ่งจะต้องใช้เนื้อที่จัดเก็บข้อมูลขนาดใหญ่ตามขนาดของระบบ ถ้าระบบมีขนาดใหญ่มาก วิธีการนี้อาจจะไม่เหมาะสมนักเพราะนอกจากจะต้องเก็บข้อมูลไว้เป็นจำนวนมหาศาลแล้ว การอัพเดทความรู้ให้แก่เอเจนต์แต่ละตัวจะต้องใช้จำนวนการประกาศข้อมูลมาก และข้อมูลที่ส่งก็มีขนาดใหญ่ตามไปด้วย

# บทที่ 4
# ระบบมัลติเอเจนต์เพื่อการค้นหาบริการอย่างรวดเร็ว สำหรับระบบประมวลผลแบบกริด

ในบทที่แล้วเป็นการกล่าวถึงระบบการค้นหาบริการแบบพื้นฐานที่นำมาใช้สำหรับระบบกริด ระบบดังกล่าวถูกพัฒนาให้เหมาะสมกับองค์กรเสมือนในระบบแลนที่มีสมาชิกจำนวนน้อยราย แต่เมื่อนำมาพัฒนาเพื่อรองรับองค์กรเสมือนที่มีใหญ่ขึ้น ก็จะมีปัญหาเกี่ยวกับโอเวอร์เฮดของจำนวนการเชื่อมต่อเพื่อค้นหาข้อมูลที่ใช้มากเกินความจำเป็น ไม่เหมาะสมต่อการทำงานในระบบแบบกระจายที่มีขนาดใหญ่อย่างองค์กรเสมือนของกริด

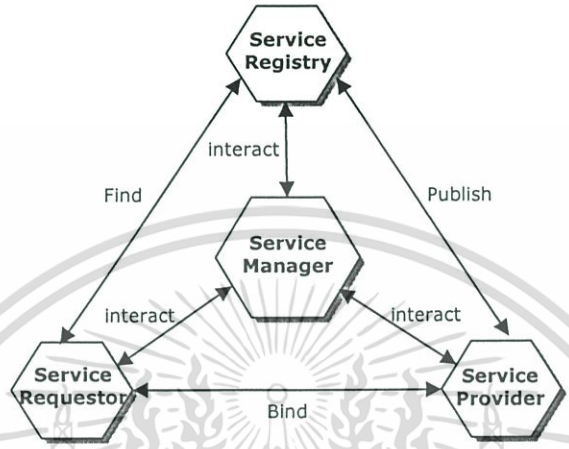
เพื่อแก้ปัญหาดังกล่าว ผู้วิจัยได้นำเสนอระบบจัดการข้อมูลแบบใหม่ พัฒนามาจากระบบซอฟท์แวร์เอเจนต์ ระบบมัลติเอเจนต์ที่สร้างขึ้น มีจุดประสงค์หลักเพื่อการค้นหาบริการแบบรวดเร็วสำหรับระบบประมวลแบบกริด นอกจากนี้ผู้วิจัยได้นำเสนอ "Knowledge Posting Services Table" หรือ K-PST เป็นโครงสร้างข้อมูลที่พัฒนามาจาก Signature File ใช้เก็บความรู้เพื่อเป็นองค์ความรู้สำหรับค้นหาข้อมูล และกล่าวถึง การเพิ่มประสิทธิภาพระบบการค้นหาด้วยการใช้ประโยชน์แคช รวมทั้งอัลกอริธึมที่ใช้ K-PST สำหรับค้นหาข้อมูลบริการในระบบ ในส่วนท้ายของบทจะกล่าวถึงการเปรียบเทียบงานวิจัยเหล่านั้นกับโมเดลที่ผู้วิจัยได้ ก่อนที่บทถัดไปจะเปรียบเทียบประสิทธิภาพด้วยการทดลองต่อไป

## 4.1 การออกแบบโครงสร้างระบบมัลติเอเจนต์

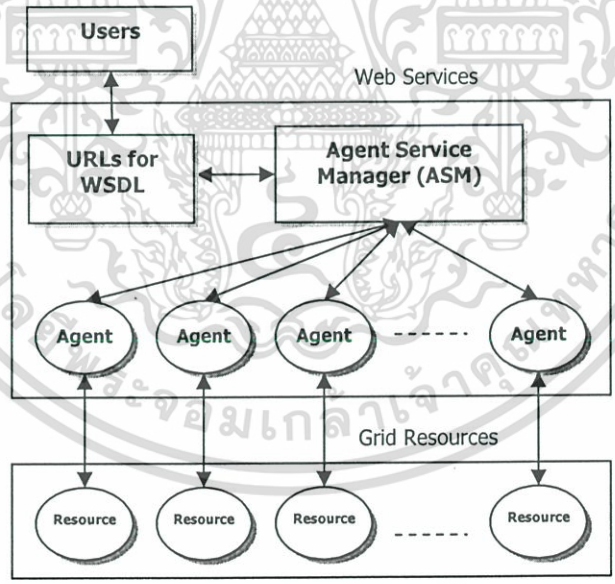
ในส่วนนี้จะเป็นการกล่าวถึงแนวคิดเบื้องต้นของการปรับปรุงระบบการค้นหาบริการในระบบกริดให้มีประสิทธิดียิ่งขึ้น ผู้วิจัยได้นำเสนอโมเดลของระบบการจัดการข้อมูลสำหรับประมวลผลกริด โมเดลที่สร้างขึ้นมามีรากฐานมาจากเทคโนโลยีเว็บเซอร์วิส โดยประยุกต์มาจากแนวคิดของสถาปัตยกรรมเชิงบริการ ในโมเดลใหม่นี้ได้มีการปรับปรุงสถาปัตยกรรมเชิงบริการด้วยการเพิ่มส่วนการควบคุมการให้บริการ (Service Manager) มีจุดประสงค์เพื่อให้การทำงานร่วมกันของคอมโพเนนต์เดิมอันได้แก่ รีจิสทรี ผู้ให้บริการ และผู้ใช้บริการของสถาปัตยกรรมเชิงบริการทำงานได้ดีขึ้น แนวคิดดังกล่าวแสดงไว้ดังรูปที่ 4.1

โมเดลที่นำเสนอนี้สร้างขึ้นนอกจากจะทำงานได้ตามมาตรฐานการให้บริการแบบเปิด (Open Grid Service Architecture: OGSA) อันเป็นมาตรฐานการให้บริการของกริดด้วยเว็บเซอร์วิสแล้ว ลักษณะโมเดลที่นำเสนอนั้นจะมีลักษณะที่ใกล้เคียงกับเว็บเซอร์วิสเชิงความหมาย (Semantic Web Service) [41] ที่เน้นการให้บริการที่สื่อความหมายข้อมูลในตัวมันเองมากยิ่งขึ้น

ความสามารถที่เพิ่มเติมเข้าไป คือ การควบคุมการให้บริการข้อมูลภายในองค์กรเสมือนของกริดให้มีประสิทธิภาพดีขึ้น โดยอาศัยหลักการของเอเจนต์เพื่อที่จะใช้ความรู้และข้อมูลเชิงสถิติที่ได้จากสภาพแวดล้อมมาวิเคราะห์และกำหนดวิธีการเชื่อมต่อให้บริการแก่ผู้ใช้ โดยผู้ใช้งานภายในระบบสามารถรับบริการข้อมูลตามสิทธิที่ตัวเองมีต่อบริการนั้นอย่างเหมาะสม




**รูปที่** 4.1 ลักษณะของโมเดลที่นำเสนอซึ่งปรับปรุงมาจากสถาปัตยกรรมเชิงบริการ



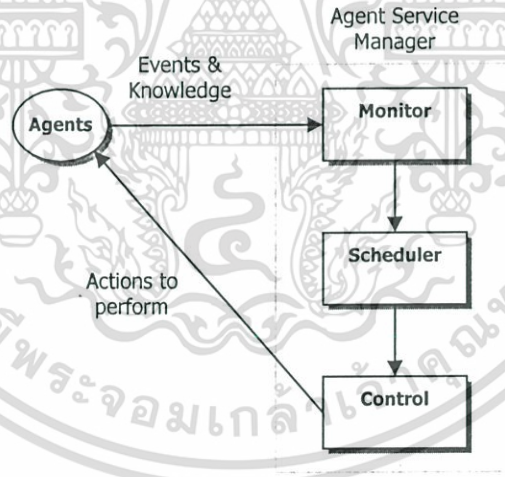

**รูปที่** 4.2 ระบบมัลติเอเจนต์เพื่อการจัดการบริการของระบบประมวลผลแบบกริด

### 4.1.1 โครงสร้างของเอเจนต์และข้อมูลในระบบ

จากแนวคิดข้างต้น ระบบการให้บริการสารสนเทศแบบกริดที่สร้างจากเว็บเซอร์วิสจะมีลักษณะดังรูปที่ 4.2 สำหรับการใช้งานระบบ ผู้ใช้บริการสามารถติดต่อกับระบบเพื่อขอรับบริการ

ผ่านทางยูอาร์แอล และอินเตอร์เฟสของเว็บเซอร์วิส ซึ่งก็คือ Web Service Description Language (WSDL) [15] ในขณะเดียวกัน ผู้ควบคุมระบบสามารถติดต่อกับตัวควบคุมบริการ (Agent Service Manager: ASM) ได้ด้วยวิธีการเดียวกัน ข้อมูลทั้งหมดในระบบจะถูกจัดการผ่านเอเจนต์ของระบบ เอเจนต์เหล่านี้จะติดต่อกับหน่วยประมวลผลและข้อมูลในระบบกริดอีกทอดหนึ่ง โดยรับทราบข้อมูลเชิงสถิติและสภาวะการทำงาน ณ ขณะนั้น เพื่อเป็นข้อมูลดิบให้ ASM พิจารณาลักษณะการทำงานที่เหมาะสมและควบคุมการทำงานโดยรวมของระบบต่อไป

ในระบบที่สร้างขึ้น นอกจากผู้ใช้บริการจะติดต่อกับระบบผ่านทาง URL ไปยัง ASM เพื่อขอใช้บริการแล้ว ผู้ใช้บริการยังสามารถติดต่อกับเอเจนต์ที่ควบคุมข้อมูลของตนเพื่อเป็นลู่ทางในการติดต่อกับระบบได้อีกทางหนึ่ง วิธีการแบบนี้มีส่วนที่จะช่วยลดภาระของ ASM ไม่ให้ทำงานหนักเกินไป สามารถการรองรับบริการของผู้ใช้ได้มากรายขึ้น เนื่องจากบริการบางอย่างสามารถทำงานผ่านเอเจนต์ได้ ยกตัวอย่างเช่น การค้นหาข้อมูล การรับรองสิทธิ์การใช้ข้อมูล การลงทะเบียนข้อมูล และการประกาศข้อมูล เป็นต้น การทำงานเหล่านี้ถือได้ว่ามีความเป็นไดนามิกส์ กล่าวคือ ผู้ใช้ไม่จำเป็นติดต่อกับระบบไปยังหน่วยศูนย์กลาง อันเป็นการใช้ประโยชน์จากระบบแบบกระจายได้อย่างเหมาะสม



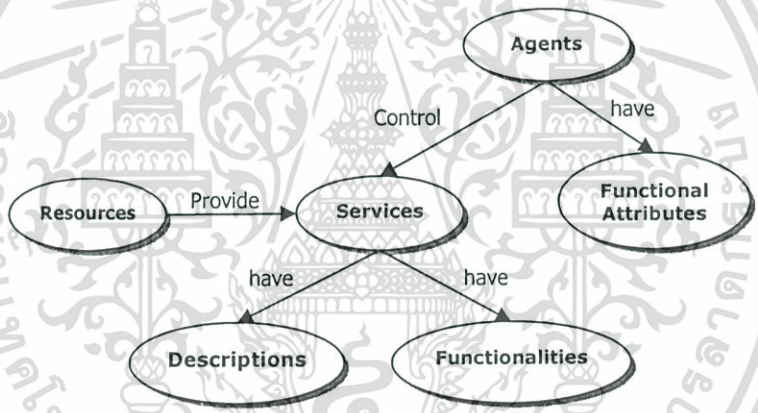

**รูปที่** 4.3 ลักษณะการทำงานร่วมกันเอเจนต์และ ASM

ในส่วนของ ASM จะมีหน่วยการทำงานย่อยที่สำคัญอยู่ 3 โมดูล ประกอบด้วย มอนิเตอร์ สเกดูเลอร์ และ คอนโทรล ทั้งสามส่วนแสดงไว้ในรูปที่ 4.3 สำหรับการทำงานของแต่ละโมดูล มอนิเตอร์จะติดต่อรับข้อมูลเชิงสถิติและเหตุการณ์จากเอเจนต์ที่ทำงานอยู่เพื่อแสดงผลแก่ผู้ใช้ระบบ จากนั้นจะส่งต่อข้อมูลไปยังสเกดูเลอร์เพื่อประเมินการให้บริการที่เหมาะสม สุดท้ายคอนโทรลโมดูลจะแจกจ่ายลักษณะทำงานที่เหมาะสมไปยังตัวเอเจนต์เพื่อปฏิบัติงานในลักษณะที่

เหมาะสมต่อไป อย่างไรก็ดี ASM จะมีหน้าที่ในการทำงานส่วนที่จำเป็นต้องพึ่งการควบคุมจากศูนย์กลางเท่านั้น อย่างเช่น การกำหนดค่าพารามิเตอร์ของระบบ การจัดโครงสร้างของเอเจนต์ที่เชื่อมต่อกัน เป็นต้น ASM จะไม่เข้าไปทำงานเกี่ยวข้องกับกลไกการทำงานแบบกระจายของเอเจนต์มากนัก เพื่อลดโอเวอร์เฮดของการสื่อสารโดยรวมภายในระบบจัดการข้อมูลของกริด

### 4.1.2 ความสัมพันธ์ระหว่างเอเจนต์ ข้อมูลบริการ และ ผู้ใช้บริการ

ในระบบที่สร้างขึ้น เอเจนต์มีหน้าที่ควบคุมข้อมูล โดยข้อมูลหมายถึงข้อมูลหรือบริการหรือรีซอร์สของระบบกริดที่ข้อมูลแชร์ให้ผู้ใช้รายอื่นในระบบกริดใช้งาน ในตัวข้อมูลจะมีส่วนอธิบายลักษณะการเชื่อมต่อ (Description) เพื่อให้ผู้ใช้บริการรายอื่นนำมาติดต่อขอใช้บริการ และส่วนอธิบายหน้าที่การทำงาน (Functionalities) ว่าข้อมูลดังกล่าวมีหน้าที่ทำอะไร และสามารถทำงานในแอพพลิเคชันแบบใดได้บ้าง นอกจากนี้ ในตัวของเอเจนต์เองก็มีส่วนของหน้าที่การทำงาน (Functional Attributes) สำหรับติดต่อสื่อสารกับเอเจนต์ตัวอื่นและติดต่อกับ ASM



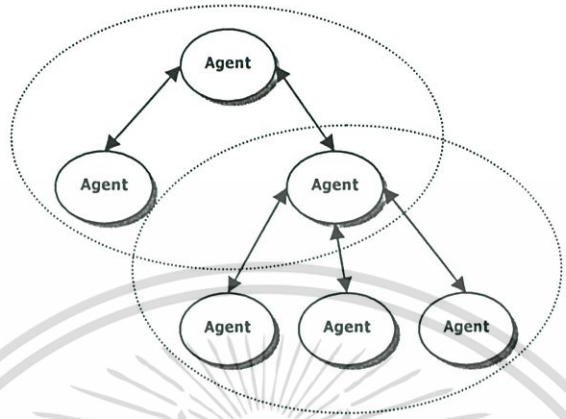

**รูปที่** 4.4 ลำดับการควบคุมของระบบการจัดการบริการของเอเจนต์

เอเจนต์แต่ละตัวจะประกอบด้วยแอตตริบิวต์ที่แตกต่างกันหลายส่วน ส่วนประกอบที่สำคัญในสำหรับการค้นหาข้อมูลอย่างรวดเร็ว คือ Signature File มีหน้าที่เก็บความรู้ของข้อมูลในระบบทั้งข้อมูลที่เอเจนต์ควบคุมอยู่ ข้อมูลในแคช และองค์ความรู้ที่อธิบายความน่าจะเป็นที่จะค้นพบข้อมูลที่ต้องการ รวมทั้งที่อยู่ของเอเจนต์ที่เป็น Parent, Neighbor และ Child เพื่อใช้ในการติดต่อกันระหว่างเอเจนต์ในโครงสร้างแบบลำดับชั้น นอกจากที่กล่าวมาแล้ว ยังมีแอตทริบิวต์ที่จำเป็นสำหรับใช้งานด้านอื่นที่นอกเหนือจากแอตทริบิวต์ที่ใช้ในระบบการค้นหาข้อมูลอีกส่วนหนึ่ง

### 4.1.3 การเชื่อมต่อระหว่างเอเจนต์

ภายในระบบที่สร้างขึ้น การเชื่อมต่อของเอเจนต์จะมีลักษณะเป็นแบบลำดับชั้น ลักษณะการเชื่อมต่อแบบนี้ทำให้การค้นหาทำได้อย่างสะดวกและรองรับการใช้บริการได้มาก ข้อมูลและรี

ซอร์สในระบบจำเป็นต้องลงทะเบียนเป็นสมาชิกภายในระบบโดยผ่านเอเจนต์หรือผ่านทางตัวควบคุมระบบเสียก่อน เพื่อที่ระบบจะได้กำหนดการอ้างอิงและระบุตำแหน่งที่อยู่เพื่อการค้นหาต่อไป




รูปที่ 4.5 โครงสร้างของเอเจนต์แบบลำดับชั้นเพื่อการบริการข้อมูลของการประมวลผลแบบกริด

สำหรับตัวอย่างที่แสดงไว้ด้านบน เป็นตัวอย่างของเอเจนต์ที่จัดการข้อมูลในองค์กรเสมือนขนาดเล็ก เอเจนต์มีการเชื่อมต่อแบ่งออกเป็น 3 ลำดับชั้น เอเจนต์แต่ละตัวเก็บความรู้ที่อธิบายถึงสถานะของการทำงานของตัวมันเองรวมทั้งในเอเจนต์ที่เป็นลูก เพื่อเป็นข้อมูลพื้นฐานสำหรับตัดสินใจควบคุมการทำงานอาทิ การค้นหาข้อมูล และการจัดการช่องทางสื่อสาร ในลักษณะที่เหมาะสม นอกจากนี้ ในระดับเดียวกันของลำดับชั้น เอเจนต์ที่มีโหนดพ่อตัวเดียวกันจะเก็บข้อมูลบางอย่างของเอเจนต์ตัวอื่นไว้ด้วย เพื่อการตัดสินใจทำงานในส่วนที่ไม่จำเป็นต้องอาศัยโหนดพ่อในการเชื่อมต่อข้อมูล ข้อมูลองค์ความรู้นี้จะเก็บไว้ที่โครงสร้างข้อมูลที่เรียกว่า K-PST ซึ่งจะอธิบายถึงรายละเอียดอีกครั้งหนึ่ง รวมทั้งการใช้ประโยชน์จากโครงสร้างข้อมูลแบบนี้สำหรับค้นหาบริการที่สามารถลดการเชื่อมต่อเพื่อสอบถามข้อมูลได้มากที่เดียว

## 4.2 การประยุกต์ใช้โครงสร้างข้อมูล Signature File

ในระบบการค้นหาข้อมูลแบบเดิม ข้อมูลหรือบริการทั้งหมดต้องลงทะเบียนและฝากข้อมูลไว้กับตัวกลางสำหรับการติดต่อสื่อสารระหว่างผู้ใช้บริการและผู้ให้บริการที่เรียกว่ารีจิสทรี หากระบบมีข้อมูลในระบบมีลักษณะแตกต่างกันจำนวนมาก รีจิสทรีจะต้องรับภาระหนักทั้งในส่วนของการเก็บข้อมูลที่อธิบายข้อมูลในระบบ และรองรับการร้องขอจากผู้ใช้บริการ ทำให้ประสิทธิภาพโดยรวมของระบบไม่ดีนักเพื่อแก้ปัญหาดังกล่าว ผู้วิจัยได้ออกแบบโครงสร้างสำหรับการจัดการข้อมูลในระบบกริดที่ได้อธิบายไว้แล้วก่อนหน้านี้ โครงสร้างของเอเจนต์ที่สร้างขึ้นเชื่อมต่อกันแบบ

ลำดับชั้น เอเจนต์แต่ละตัวจะควบคุมผู้ใช้บริการและข้อมูลในระบบ กระบวนการค้นหาข้อมูลในโครงสร้างของเอเจนต์จะใช้ประโยชน์จากโครงสร้างข้อมูลที่เรียกว่า Signature File ที่เป็นเสมือนตัวแทนข้อมูลซึ่งอธิบายถึงข้อมูลที่เอเจนต์ดูแลอยู่ ในระบบการค้นหาที่สร้างขึ้นจะใช้ประโยชน์จาก Signature File ในการตัดสินใจเลือกเส้นทางที่ทำให้จำนวนการเชื่อมต่อเพื่อสอบถามข้อมูลจากเอเจนต์ในระบบมีจำนวนลดลง นอกจากนี้ ผู้วิจัยได้ประยุกต์ใช้ Signature File เพื่อสร้างเป็นโครงสร้างที่อธิบายความรู้ ใช้สำหรับการค้นหาข้อมูลในโครงสร้างแบบลำดับชั้นให้มีประสิทธิภาพมากขึ้น โดยมีการเก็บ Signature File ของเอเจนต์ที่เชื่อมต่อกันอยู่ไว้ในตารางความรู้ของเอเจนต์แต่ละตัว เรียกตารางความรู้นี้ว่า "Knowledge Posting Service Table" หรือ K-PST

### 4.2.1 โครงสร้างข้อมูล Signature File

Signature File หรือ Bloom Filter [42, 43, 44] เป็นโครงสร้างข้อมูลที่ใช้อธิบายเซต $S = \{s_1, s_2, s_3, ..., s_n\}$ ที่มีสมาชิก $n$ ตัวจากจำนวนสมาชิกทั้งหมดในขอบเขตที่กำหนด $U$ โครงสร้างข้อมูลนี้ประกอบด้วยแอร์เรย์จำนวน $m$ บิต ทุกๆ บิตจะเริ่มต้นด้วยเซตค่าเป็น '0' โดยทั่วไปอัตราส่วนระหว่างค่า $m/n$ จะถูกกำหนดเป็นค่าคงที่เพื่อนำมาพิจารณาความเหมาะสมกับการใช้งานของแอพพลิเคชัน Signature File จะใช้แรนดอมแฮชฟังก์ชันเป็นจำนวน $k$ ได้แก่ $h_1$, ..., $h_k$ ที่สุ่มค่าในช่วง $\{0, ..., m - 1\}$ สำหรับแต่ละอีเลเมนต์ $s \in S$ บิต $h_i(s)$ จะถูกเซตเป็น '1' สำหรับ $1 \le i \le k$ โดยแต่ละตำแหน่งสามารถเซตเป็น '1' ได้หลายครั้ง แต่เฉพาะการเซตครั้งแรกเท่านั้นที่มีผล การทดสอบความเป็นสมาชิกของ $s$ ในเซต $S$ ทำได้โดย

ถ้าทุกๆ $h_i(s)$ มีค่าเป็น 1 กล่าวได้ว่า $s$ เป็นสมาชิกของ $S$ ด้วยความน่าจะเป็นค่าหนึ่ง แต่ถ้า มี $h_i(s)$ ที่มีค่าเป็น '0' กล่าวได้ว่า $s$ ไม่เป็นสมาชิกของ $S$ แน่นอน

ถ้าทุกๆ $h_i(s)$ มีค่าเป็น 1 แต่ว่าในความเป็นจริง $s$ ไม่เป็นสมาชิกของ $S$ เราเรียกปรากฏการณ์นี้ว่า *false positive* โดย false positive probabilistic สามารถคำนวณได้ดังนี้

สมมติให้แฮชฟังก์ชันสามารถสุ่มค่าได้อย่างสมบูรณ์ และหลังจากที่มีการแอชอิเลเมนต์ทุกตัวลงไปยัง S แล้ว ความน่าจะเป็นบิตที่ยังคงมีค่าเป็น '0' คือ

$$p = (1 - \frac{1}{m})^{kn} \approx e^{-kn/m} \qquad (4.1)$$

สำหรับ false positive จะเกิดขึ้นเมื่อมีการทดสอบความเป็นสมาชิกของอีเลเมนต์ที่ไม่ได้อยู่ในเซต $S$ แต่ละตำแหน่ง $k$ ต้องไม่มีค่าเป็น '0' เพื่อให้ง่ายต่อการวิเคราะห์ เรากำหนดให้

Signature File ทั้งหมด มีความน่าจะเป็นที่จะเซตค่าเป็น '0' อย่างอิสระ $p$ และเซตค่าเป็น '1' เท่ากับ $1-p$ ดังนั้น false positive probabilistic จึงมีค่าเท่ากับ

$$f = (1-(1-\frac{1}{m})^{kn})^k \approx (1-e^{-kn/m})^k \quad (4.2)$$

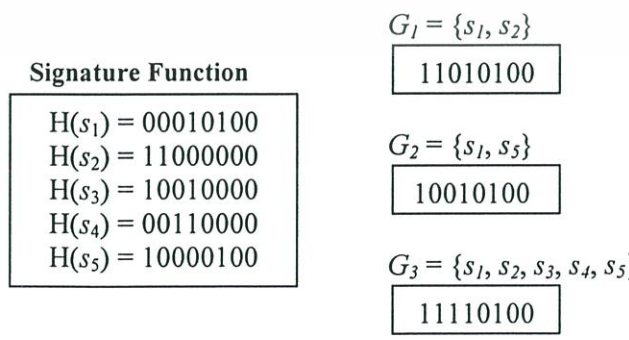
จากสมาการด้านบนจะเห็นได้ว่า Signature File มีปัจจัยด้านประสิทธิภาพที่มีข้อดีข้อเสีย (Trade-offs) แตกต่างกัน 3 อย่างก็คือ

Computation Time – สอดคล้องกับจำนวนแฮชฟังก์ชัน $k$

False Positive – สอดคล้องกับค่าความน่าจะเป็นของความผิดพลาด $f$

Space – สอดคล้องกับจำนวนบิตแอร์เรย์ของ Signature File $m$ และ อัตราส่วนจำนวนบิตต่ออีเลเมนต์ หาก $m/n$ มีค่ามาก จะทำให้ค่า False Positive Probability มีค่าน้อย เมื่อจำนวน hash function $k$ มีค่าคงที่

ในงานวิจัยนี้ได้นำเอาแนวคิดของ Signature File มาสร้าง K-PST เพื่อเป็นองค์ความรู้สำหรับการค้นหาข้อมูล อาศัยการทำงานของ Signature File ที่ใช้การแฮชซิ่งฟังก์ชันจัดการกับข้อมูลโดยจับคู่ข้อมูลแต่ละไปเป็นเลขไบนารีจำนวน $n$ บิต เป็นสัญลักษณ์แทนตัวชิ้นข้อมูล ข้อมูลแต่ละชิ้นมีสัญลักษณ์แทนตัวเองไม่ซ้ำกัน เมื่อต้องการรวบรวมข้อมูลเป็นกลุ่ม จะนำเอาข้อมูลทุกชิ้นผ่านแอชซิ่งฟังก์ชันแล้วนำมาทำโอเปอร์เรชัน OR แล้วเก็บไว้ที่บล็อกขนาด $n$ บิต การรวบรวมข้อมูลแบบนี้สื่อความหมายว่า ถ้ามีข้อมูลชิ้นหนึ่งอยู่ในกลุ่มข้อมูลชุดนี้ บิตสัญลักษณ์ที่แทนตัวข้อมูลชิ้นนั้นจะมีค่าเป็น 1 เสมอ การสื่อความหมายดังกล่าวสามารถนำมาประยุกต์ใช้กับการค้นหาข้อมูลหรือรีซอร์สได้ โดยสื่อความหมายว่าข้อมูลที่ต้องการค้นหามีอยู่ในกลุ่มหรือไม่ ถ้ามีก็ให้ค้นหาลงไปในกลุ่มย่อย แต่ถ้าไม่มีก็ให้หยุดการค้นหา กลไกการค้นหาแบบนี้จะกล่าวโดยละเอียดอีกครั้งหนึ่ง




**รูปที่** 4.6 การเก็บข้อมูลเพื่อสื่อความหมายด้วย Signature File

รูปที่ 4.6 แสดงตัวอย่างการทำงานของ Signature File สำหรับเก็บข้อมูลเป็นกลุ่มๆ ในที่นี้ข้อมูล $s$ แต่ละตัวจะมีค่า Signature ของตัวเอง เมื่อต้องการรวบรวมข้อมูลสมาชิกในกลุ่ม $G$ จะนำเอาค่า Signature ของ $s$ ในกลุ่มมาทำโอเปอเรชัน OR กันแล้วเก็บไว้ที่บล็อก ผลลัพธ์ที่ได้จะเป็น Signature แทนกลุ่มข้อมูลที่สื่อความหมายถึงสมาชิกที่มีอยู่ เช่นในกลุ่ม $G_1$ มีสมาชิกสองตัว คือ $s_1$ และ $s_2$ ข้อมูลแทนกลุ่มก็คือ "11010100" ซึ่งได้จากการ OR กันของสัญลักษณ์แทนตัว $s_1$ และ $s_2$ ซึ่งก็คือ "00010100" และ "11000000"

สำหรับระบบการค้นหาที่สร้างขึ้น เอเจนต์แต่ละตัวจะเก็บ Signature File ที่รวบรวมข้อมูลที่ตนดูแลอยู่ ประกอบด้วยข้อมูลไดเรกทอรีและ แคช รวมทั้งเก็บ Signature File ของเอเจนต์ที่เชื่อมต่อในระยะใกล้เคียงบางส่วนด้วย ดังนั้น ในเอเจนต์แต่ละตัวจึงเก็บ Signature File มากกว่า 1 ตัว Signature File ที่รวมกันอยู่นี้จะเก็บไว้ตามลำดับความสำคัญต่อการค้นหาข้อมูล เรียงกันไปตามลำดับเหมือนเป็นเรคอร์ดของตาราง จึงเรียก Signature File ที่รวมกันอยู่นี้ว่า "Knowledge Posting Service Table" หรือเรียกสั้นๆ ว่า K-PST

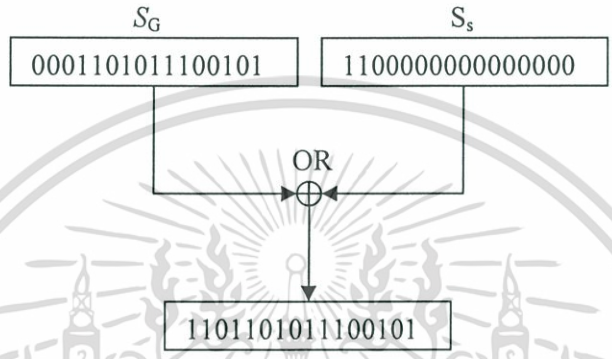
| *Signature ($S_G$)* | *Signature Value* |
|---|---|
| *Current Agent* | 1111101011100101 |
| $Child_1$ | 0100000000000101 |
| $Child_2$ | 0111000011000101 |
| $Child_3$ | 0100100000100101 |
| $Neighbor_1$ | 0101100001000101 |
| $Neighbor_2$ | 0100101000000101 |
| $ParentLevel_1$ | 1111101011100101 |
| $ParentLevel_2$ | 1111101011101111 |
| $ParentLevel_3$ | 1111111111101111 |

**รูปที่** 4.7 ตัวอย่างของ Knowledge Posting Service Table

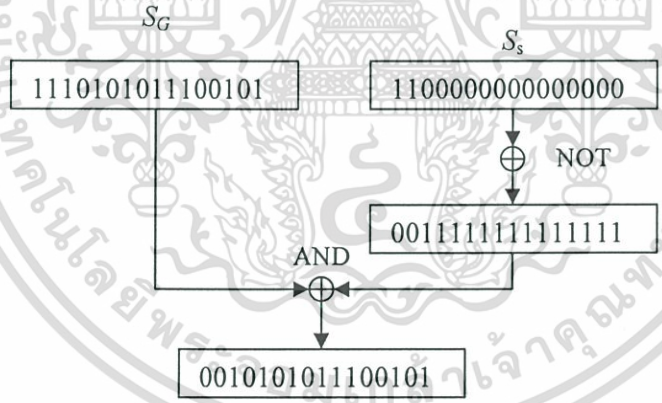
ในเอเจนต์แต่ละตัวในระบบจะเก็บ K-PST ในลักษณะคล้ายคลึงกับรูปที่ 4.7 ด้านบน แต่จะมีความแตกต่างกันที่จำนวนการเชื่อมต่อกับเอเจนต์ที่อยู่ใกล้เคียงกับเอเจนต์นั้น และจำนวนลำดับชั้นของโครงสร้างเอเจนต์ ค่าที่เก็บใน K-PST จะสื่อความหมายถึง ประเภทของข้อมูลที่เอเจนต์แต่ละตัวดูแลอยู่ สำหรับเนื้อที่สำหรับการเก็บข้อมูลจะมีค่าเท่ากับจำนวนบิตของ Signature File แต่ละแถวคูณกับจำนวนเอเจนต์ที่เชื่อมต่อทั้งหมด จะเห็นได้ว่าจำนวนเนื้อที่เก็บข้อมูลที่ใช้ มีค่าน้อยกว่าการเก็บคำอธิบายการเชื่อมต่อของข้อมูลแต่ละตัวมาก ซึ่งถือเป็นข้อได้เปรียบของการใช้ K-PST เมื่อเปรียบเทียบกับวิธีการค้นหาแบบอื่นๆ ในส่วนของรายละเอียดการใช้งานของ K-PST จะกล่าวถึงอีกครั้งในหัวข้อที่เกี่ยวข้องกับการปรับปรุงประสิทธิภาพของระบบการค้นหาด้วยการใช้ความรู้ต่อไป

### 4.2.2 การเพิ่มและการถอดถอนข้อมูล

การใช้งาน Signature File สำหรับอธิบายข้อมูลในระบบกริดนั้น ในแต่ละช่วงเวลาข้อมูล Signature สามารถมีค่าแตกต่างกันได้ ขึ้นอยู่กับว่า ณ ขณะนั้นมีข้อมูลใดบ้างในกลุ่มที่ให้บริการอยู่ เนื่องจากว่า สมาชิกที่ใช้งานองค์กรเสมือนของกริดสามารถแชร์ข้อมูลของตนให้ผู้อื่นได้ใช้งานและถอดถอนการใช้งานนั้นจากระบบเมื่อใดก็ได้ การดำเนินการทั้งสองจะเกี่ยวข้องกับ Signature ในสองลักษณะคือ การเซตค่าข้อมูลที่ตนอนุญาตมีให้และการยกเลิกข้อมูลที่มีอยู่ใน Signature




**รูปที่** 4.8 การทำงานของ Signature File สำหรับการเพิ่มข้อมูลที่ให้บริการ



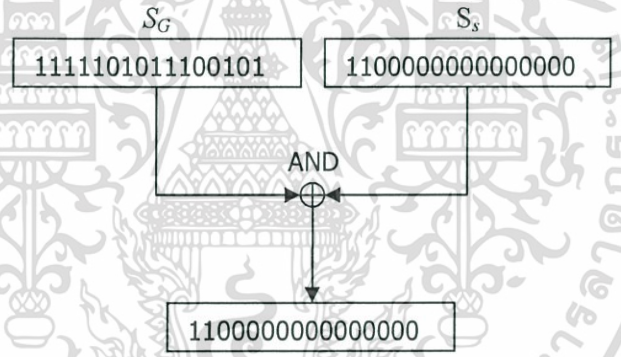

**รูปที่** 4.9 การทำงานของ Signature File สำหรับการถอดถอนข้อมูลที่ให้บริการ

เมื่อสมาชิกต้องการจะแชร์ข้อมูลหรือเปิดข้อมูลเพื่อให้บริการแก่ผู้อื่น ขั้นแรกจะแจ้งให้เอเจนต์ที่ควบคุมตนอยู่ทราบ จากนั้นเอเจนต์จะนำ Signature ที่แทนข้อมูลที่ผู้ใช้จะแชร์แก่ระบบมาทำโอเปอเรชัน OR กับ Signature ของกลุ่มชุดเดิม ผลลัพธ์ที่ได้จะถูกเก็บไว้ใน Signature เดิมต่อไป ดังจะเห็นได้จากรูปที่ 4.8 ซึ่งเป็นตัวอย่างการทำงานของ Signature สำหรับการเพิ่มข้อมูลที่ให้บริการ เมื่อผู้ใช้การประกาศแชร์ข้อมูล $s$ แก่ระบบ เอเจนต์จะนำ Signature ของ $s$ หรือ $S_s$ คือ "1100000000000000" มา OR กับ Signature ของกลุ่ม $G$ หรือ $S_G$ ที่มีอยู่เดิมคือ

"0001101011100101" ผลลัพธ์ที่ได้คือ "1110101011100101" จะถูกเก็บไว้ที่ Signature เหมือนเดิม

สำหรับการยกเลิกการแชร์ข้อมูลหรือปิดการใช้บริการ เมื่อสมาชิกไม่ต้องการให้บริการข้อมูลแก่ระบบ สมาชิกจะแจ้งไปยังเอเจนต์ที่ควบคุมสมาชิกอยู่เพื่อขอยกเลิกการให้บริการข้อมูล ขั้นแรก เอเจนต์จะตรวจสอบว่าบริการประเภทดังกล่าวมีให้บริการโดยสมาชิกรายอื่นในกลุ่มหรือไม่ ถ้ามีเอเจนต์จะไม่มีการเปลี่ยนแปลงข้อมูลใดๆ ของ Signature ในทางตรงกันข้ามเอเจนต์จะแก้ไขข้อมูลภายใน Signature โดยอาศัยกลไกตามรูปที่ 4.9 เริ่มต้นด้วยการนำเอา Signature ของข้อมูลที่ต้องการถอดถอนออกจากกลุ่มการให้บริการ คือ "1100000000000000" มาทำโอเปอเรชัน NOT จากนั้นจึงนำผลลัพธ์ที่ได้คือ "0011111111111111" มาทำโอเปอเรชัน AND กับ Signature ของกลุ่มคือ "1110101011100101" ผลลัพธ์สุดท้ายคือ "0010101011100101" จะถูกเก็บไว้ใน Signature File ดังเดิม

### 4.2.2 การค้นหาข้อมูล




**รูปที่** 4.10 การทำงานของ Signature File เพื่อการค้นหาข้อมูล

จากที่กล่าวมาแล้วว่าข้อมูลที่เก็บใน Signature File นั้นสื่อความหมายของข้อมูลที่เอเจนต์ครอบครองอยู่ สำหรับระบบการค้นหาข้อมูล เมื่อต้องการค้นหาข้อมูลในกลุ่มข้อมูลใดข้อมูลหนึ่ง ระบบจะใช้บล็อกข้อมูลที่แทน Signature ของข้อมูลที่ต้องการจะค้นหา มาทำโอเปอเรชัน AND กับบล็อกข้อมูลของกลุ่มใน Signature กำหนดให้ $S_G$ เป็น Signature ของกลุ่มข้อมูล และ $S_s$ เป็น Signature ของข้อมูลที่ต้องการค้นหา และ

$$S = S_G \cdot S_s \tag{4.3}$$

ถ้าผลลัพธ์ที่ออกมามีค่าตรงกับบล็อกข้อมูลที่แทน Signature ของข้อมูลที่ต้องการจะค้นหา โดย $S = S_s$ หมายความว่าในกลุ่มข้อมูลนั้นมีความเป็นไปได้ที่จะมีข้อมูลที่ต้องการอยู่ แต่

ถ้าผลลัพธ์ที่ออกมาไม่ตรงกับ Signature ของข้อมูลที่ต้องการจะค้นหา ก็บ่งชี้ได้เลยว่าไม่มีข้อมูลที่ต้องการค้นหาแน่นอน จึงไม่จำเป็นต้องค้นหาต่อในกลุ่มข้อมูลย่อย

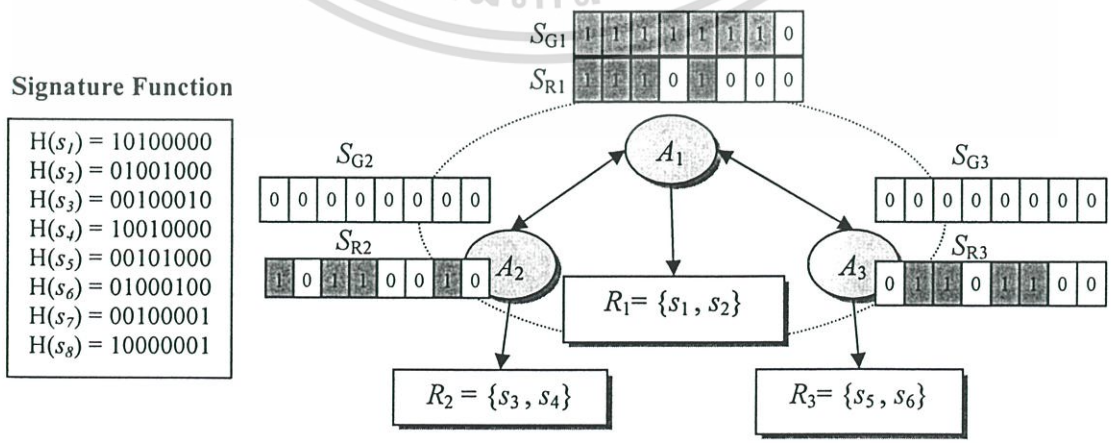
รูปที่ 4.10 เป็นตัวอย่างการใช้งาน Signature File เพื่อการค้นหาข้อมูลในระบบ เมื่อระบบต้องการจะค้นหาข้อมูล $s$ ในกลุ่ม $G$ ดังรูป ระบบจะนำ $S_s$ และ $S_G$ มาทำโอเปอเรชัน AND กัน ในตัวอย่าง ผลลัพธ์ที่ออกมามีค่าตรงกับ $S_s$ สื่อความหมายว่าในกลุ่มข้อมูล $S_G$ มีความเป็นไปได้ที่จะมีข้อมูล $s$ อยู่ ในทางตรงกันข้าม หากผลลัพธ์มีค่าไม่เท่ากับ $S_s$ ระบุได้แน่นอนว่า ไม่มีข้อมูล $s$ อยู่ใน $G$

## 4.3 ระบบจัดการข้อมูลของเอเจนต์ในโครงสร้างลำดับชั้น

ระบบการค้นหาข้อมูลระหว่างเอเจนต์ในโครงสร้างลำดับชั้น ที่สร้างขึ้นมาเพื่อจัดการข้อมูลในระบบประมวลผลแบบกริดนั้น จะอาศัยการทำงานของ Signature ที่กล่าวไว้ก่อนหน้านี้เป็นแกนหลักสำหรับการสื่อสารระหว่างเอเจนต์ในระบบ โดยระบบการค้นหาที่สมบูรณ์จะมีส่วนการทำงานย่อย 4 ส่วนคือ การลงทะเบียน การประกาศ การค้นหา และ การบำรุงรักษา การทำงานทั้งหมดล้วนเกี่ยวข้องกับ Signature ทั้งสิ้น ในส่วนนี้จะกล่าวถึงการใช้ Signature สำหรับการอธิบายข้อมูลในโครงสร้างลำดับชั้น และ การทำงานพื้นฐานของการทำงานย่อยเหล่านั้นเสียก่อน และในหัวข้อถัดจึงจะกล่าวการเพิ่มประสิทธิภาพแก่ระบบด้วยวิธีต่างๆ

### 4.3.1 การใช้ Signature อธิบายความหมายข้อมูลในโครงสร้างลำดับชั้น

โครงสร้าง Signature File แสดงลักษณะของข้อมูลที่มีอยู่ในระบบ สำหรับการใช้งาน Signature ในโครงสร้างลำดับชั้นของเอเจนต์จะมีการเชื่อมโยงความสัมพันธ์ระหว่าง Signature File ของเอเจนต์แต่ละตัว ให้อธิบายความหมายของข้อมูลทั้งในส่วนของตัวมันเอง และอธิบายความเป็นไปได้ที่จะมีข้อมูลที่ต้องการอยู่ในระบบ ซึ่งเป็นฐานความรู้ของระบบการค้นหาข้อมูล




**รูปที่** 4.11 ความหมายของ Signature File ในการอธิบายข้อมูลในโครงสร้างแบบลำดับชั้น

ในรูปที่ 4.11 เป็นตัวอย่างการอธิบายข้อมูลในโครงสร้างแบบลำดับชั้นขนาดเล็ก โครงสร้างนี้ประกอบด้วยเอเจนต์จำนวน 3 ตัวที่มีข้อมูลอยู่ในครอบครองแตกต่างกัน ข้อมูลในระบบมีทั้งหมด 8 ประเภท แต่ละตัวจะมี Signature แทนตัวเองตามที่ระบุไว้ใน Signature Functions

$S_G$ เป็นสัญลักษณ์ หรือ Signature แทนกลุ่มข้อมูล สื่อความหมายว่า "ขณะนั้นภายใต้สับทรีของเอเจนต์มีความเป็นไปได้ที่จะมีข้อมูลใดอยู่บ้าง" ตัวอย่างเช่น $S_G$ ของ $A_1$ มีค่าเท่ากับ "11111100" หมายความว่า ขณะนั้นภายใต้สับทรีของ $A_1$ มีความเป็นไปได้ที่จะมีข้อมูล $R=\{s_1,s_2,s_3,s_4,s_5,s_6\}$ อยู่ ดังนั้น $S_G$ ของเอเจนต์จะได้จาก $S_{GC}$ คือ $S_G$ ของลูกทั้งหมดทำโอเปอเรชัน OR กับ กับข้อมูลที่เอเจนต์ตัวลูกครอบครองอยู่ $S_{RC}$ วิธีการดังกล่าวแสดงไว้ในสมการ (4.2) ดังนี้

$$S_G=\sum_{\forall C}(S_{GC}+S_{RC}) \tag{4.4}$$

สังเกตว่า $S_G$ ของ $A_2$ และ $A_3$ ต่างมีค่าเท่ากับ "00000000" เนื่องจากทั้ง $A_2$ และ $A_3$ เป็นเอเจนต์ที่ไม่มีลูก ในส่วนของ $S_R$ นอกจากจะหมายถึงข้อมูลที่ควบคุมอยู่แล้ว ยังรวมถึงข้อมูลในแคชที่เอเจนต์เก็บไว้ด้วย

### 4.3.2 การลงทะเบียน

เมื่อผู้ใช้บริการจะเข้าสู่ระบบ ผู้ใช้บริการจะต้องลงทะเบียนเข้าเป็นสมาชิกเสียก่อน การเข้าเป็นสมาชิกของระบบทำได้ 2 วิธี คือ การติดต่อกับ ASM ด้วย URL ที่ทราบแน่ชัด และการทำมัลติแคสติ้งค้นหาเอเจนต์ในระบบที่อยู่ใกล้ที่สุด

เมื่อสามารถติดต่อได้แล้วด้วยวิธีใดวิธีหนึ่ง ระบบจะตัดสินใจว่าผู้ใช้รายนั้นจะถูกดูแลโดยเอเจนต์ตัวใด หรือจะมีการสร้างเอเจนต์ตัวใหม่เพื่อควบคุมข้อมูลของสมาชิกรายนั้น โดยการตัดสินใจของระบบจะพิจารณาจากค่าระยะทางระหว่างสมาชิกกับเอเจนต์ ถ้าอยู่ในขอบเขตการควบคุมของเอเจนต์ที่มีอยู่เดิมก็จะให้เอเจนต์ตัวนั้นรับผิดชอบผู้ใช้รายนั้นทันที ในทางกลับกัน ถ้าสมาชิกไม่ได้อยู่ในขอบเขตการให้บริการรายใดเลย ระบบก็จะสร้างเอเจนต์ตัวใหม่เข้าสู่ระบบเพื่อจัดการข้อมูลของสมาชิกรายนั้น เอเจนต์ตัวใหม่นี้จะเป็นลูกของเอเจนต์ตัวเดิมที่มีอยู่ในระบบและระยะทางใกล้ที่สุดเพื่อประโยชน์ในการด้านความเร็วติดต่อสื่อสาร ส่วนโค้ดของเอเจนต์จะฝากรันไว้ที่สมาชิกรายนั้นด้วย

| $S_G$ | Signature Value |
|---|---|
| *Current Agent* | 0000000000000000 |
| $Parent_1$ | 0101011001101100 |
| $Parent_2$ | 1101101111111000 |

**รูปที่** 4.12 ค่าเริ่มต้นของ Signature File สำหรับเอเจนต์ที่สร้างขึ้นมาใหม่

เมื่อมีเอเจนต์ตัวใหม่เพิ่มขึ้นมาในระบบ เอเจนต์ตัวนั้นจะได้รับค่าเริ่มต้นของ Signature File ของตัวมันเองเป็น '0' ทั้งหมด ดังรูปที่ 4.12 ค่าเริ่มต้นลักษณะนี้หมายความว่ามีสมาชิกรายใหม่เข้าสู่ระบบและสมาชิกยังไม่ได้มีข้อมูลให้ผู้อื่นใช้บริการ แต่เมื่อใดก็ตามที่สมาชิกเพิ่มหรือถอดถอนข้อมูล ระบบจะดำเนินการกับ ของ Signature File ตามที่ได้กล่าวไว้ในหัวข้อ 4.2 แล้ว จากนั้น ของ Signature File ของเอเจนต์จะถูกประกาศให้เอเจนต์ตัวอื่นได้รับทราบ โดยผ่านกลไกการประกาศที่จะกล่าวไว้ในลำดับต่อไป นอกจากนี้ จะเห็นได้ว่าในตารางความรู้มี Signature File ของ $Parent_1$ และ $Parent_2$ ติดมาด้วย ค่านี้จะได้รับมาจากเอเจนต์ที่เป็น Parent ของเอเจนต์ที่สร้างขึ้นมาใหม่

### 4.2.3 การประกาศและการบำรุงรักษา

เมื่อสมาชิกมีการเปลี่ยนแปลงการให้บริการข้อมูลของตน ทั้งการเพิ่มและการถอดถอน เอเจนต์จำเป็นต้องประกาศการเปลี่ยนแปลงข้อมูลของตนให้เอเจนต์ในระบบได้รับทราบ สำหรับการเปลี่ยนแปลงข้อมูล Signature แต่ละครั้ง เอเจนต์จะส่งค่า *Current Agent* ของตนไปยัง *Parent* ของตนเพื่อทำการแก้ไขค่า *Current Agent* ที่ใช้อ้างอิงข้อมูลของตนและเก็บไว้เป็นค่า *Child* ของ Parent ตัวดังกล่าวเพื่อใช้อ้างอิง จากนั้นจะส่งไปแก้ไขตาม *Parent* แต่ละขั้นเรื่อยๆ จนกระทั่งถึง root ของลำดับชั้น

จะเห็นได้ว่าการประกาศข้อมูลแต่ละครั้งนั้นมีความสิ้นเปลืองมาก กล่าวคือ ต้องส่งข้อมูลไปยังทุกโหนดของลำดับชั้น การประกาศข้อมูลทุกครั้งแก่ระบบจึงไม่เหมาะสมนัก วิธีแก้ปัญหาก็คือ จะไม่มีการประกาศข้อมูลแก่ระบบทันที แต่จะเก็บข้อมูลที่แก้ไขไว้ก่อน รอเวลาระบบอนุญาตให้มีการประกาศข้อมูลของระบบ การทำงานการประกาศแบบนี้ถือเป็นการทำงานแบบช่วงเวลา (Periodic) แตกต่างกับการลงทะเบียนที่ทำงานเมื่อมีการเปลี่ยนแปลงเหตุการณ์ (Event Driven)

## 4.4 การค้นหาข้อมูลด้วย Signature File

การค้นหาบริการในโครงสร้างลำดับชั้นจะใช้ Signature File ในที่นี้ ใช้ข้อมูลของอธิบายกลุ่ม $S_G$ เป็นข้อมูลอ้างอิงในการค้นหาข้อมูลในระบบ เมื่อสมาชิกต้องการค้นหาข้อมูล $s$ ในโครงสร้างแบบลำดับชั้น สมาชิกจะติดต่อกับเอเจนต์ที่ควบคุมสมาชิกรายนั้นอยู่ เมื่อเอเจนต์ได้รับ

การร้องขอค้นหาข้อมูลแล้ว จะนำเอาข้อมูล $s$ มาผ่านแอซซิ่งฟังก์ชัน ผลลัพธ์ที่ได้ก็คือ Signature ของข้อมูลนั้นนั่นเอง จากนั้น เอเจนต์จะนำ Signature นี้ไปค้นหาข้อมูลในลำดับชั้นของเอเจนต์ที่เชื่อมต่อกันอยู่ต่อไป ในส่วนนี้จะเป็นการกล่าวถึงขั้นตอนการค้นหาแบบพื้นฐานโดยตั้งชื่อวิธีการค้นหาเพื่อการอ้างอิงว่า "Signature" ตามโครงสร้างข้อมูลที่ใช้สำหรับการค้นหา วิธีการของ Signature มีการค้นหาแบบครบถ้วน (Full Discovery) โดยจะมีการค้นหาไปยังทุกโหนดที่มีความเป็นไปได้ว่าจะมีข้อมูลที่ต้องการอยู่ ซึ่งยังไม่ได้มีการกระบวนการเพิ่มประสิทธิภาพใดๆ เข้าไป ขั้นตอนการค้นหาข้อมูลในลำดับชั้นของกระบวนการ Signature มีดังนี้

กำหนดให้

$A_i$ โดยที่ ($i$ = 1, 2, ..., $n$) เป็นเอเจนต์

$S_d$ คือ Signature ของข้อมูลหรือบริการที่ต้องการค้นหา

$A_i(S_d)$ คือ การค้นหาที่เอเจนต์ตัวที่ $i$

$R(S_d)$ คือ การตรวจสอบข้อมูล $S_d$ ในข้อมูลที่เอเจนต์ครอบครองอยู่

$S(S_d)$ คือ การตรวจสอบข้อมูล $S_d$ ใน Signature ของข้อมูล

โดยที่ $R(S_d), S(S_d) \in \{ A_i (i = 1, 2, ..., n), null \}$

*null* หมายถึง ไม่พบข้อมูล และ * หมายถึง ไม่พิจารณา

กฎการค้นหาในโครงสร้างลำดับชั้น

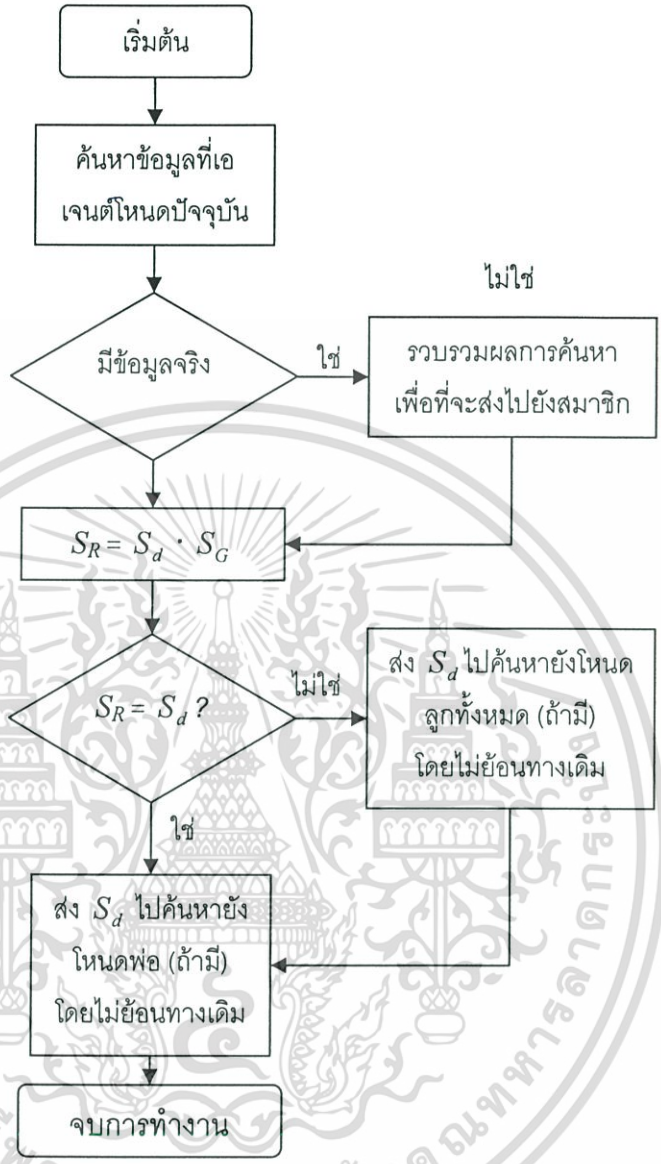
(1) $A_i(S_d) \Rightarrow A_i \rightarrow (R(S_d), S(S_d))_{Ai}$

(2) $(A_{This}, *)_{This} \Rightarrow$ SERVICE_FOUND $\rightarrow$

(3) $(*, A_j)_{This} \Rightarrow A_j(S_d)$

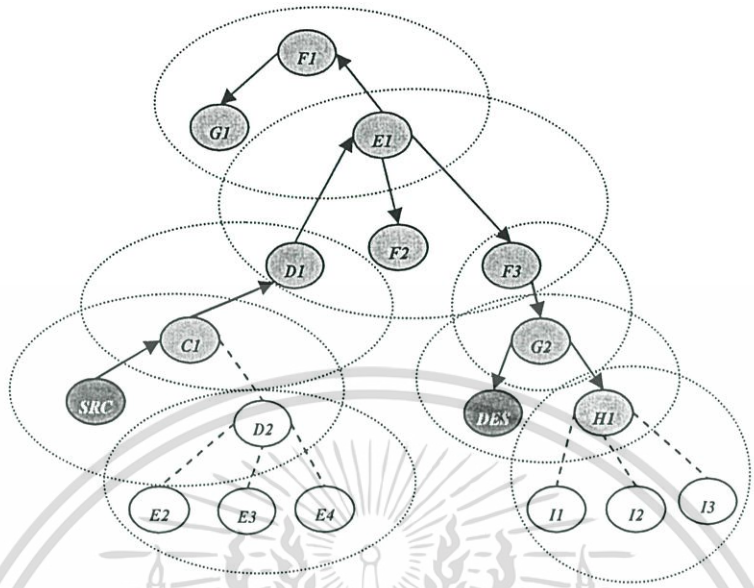
(4) $(*, null)_{Root} \Rightarrow$ END

(5) $(*, null)_{Leaf} \Rightarrow$ END




**รูปที่** 4.13 วิธีการค้นหาข้อมูลในเอเจนต์แต่ละตัว

การค้นหาข้อมูลด้วยวิธี Signature จะทำให้จำนวนการเชื่อมต่อสำหรับการค้นหาลดลงในระดับหนึ่ง กล่าวคือ เมื่อทราบว่าไม่มีข้อมูลที่ต้องการค้นหาอยู่ในสับทรี ก็ไม่จำเป็นต้องค้นหาต่อไป สามารถสิ้นสุดการค้นหาได้ทันที ในการทำงานรูปที่ 4.13 สังเกตว่า จะไม่มีการเชื่อมต่อค้นหาย้อนทางเดิมไม่ว่าจะไปยังโหนดพ่อ หรือ ลูกก็ตาม การดำเนินการแบบนี้ การเชื่อมต่อเพื่อการค้นหาจะต้องมีการระบุด้วยว่าก่อนหน้านี้การค้นหาได้กระทำผ่านโหนดใดมาก่อน วิธีการนี้จะเป็นการจัดการเพื่อไม่ให้เกิดความซ้ำซ้อนต่อระบบการค้นหาในลำดับชั้นของเอเจนต์




รูปที่ 4.14 การค้นหาข้อมูลในโครงสร้างลำดับชั้นของเอเจนต์

จากรูปที่ 4.14 ด้านบนแสดงวิธีการค้นหาข้อมูลที่นำเสนอ โดยสมมติให้ผู้ใช้ในการควบคุมของเอเจนต์ SRC ซึ่งเป็นต้นทาง ต้องการร้องขอข้อมูล $s$ ที่อยู่ในการควบคุม DES โดยที่ไม่ทราบมาก่อนว่าบริการที่ต้องขอนั้นอยู่ที่ใด ในตัวอย่างนี้กำหนดให้ในระบบมีข้อมูล $s$ อยู่ที่ DES เพียงแห่งเดียว

เริ่มต้นจะมีการค้นหาข้อมูลในเอเจนต์ SRC ก่อน ซึ่งในกรณีตัวอย่างจะไม่พบบริการที่ต้องการ จึงต้องมีการร้องขอให้โดยเปรียบเทียบกับ $S_G$ ของเอเจนต์ในลำดับถัดไปก็คือ ค้นหาบริการของเอเจนต์ที่แสดงด้วยสีทึบ ได้แก่ C1, D1, E1, F1, F2, F3, G1, G2, H1 และสุดท้ายค้นพบบริการที่ต้องการที่ DES ตามลำดับ เมื่อพบบริการดังกล่าวแล้ว สุดท้ายจะมีการส่งผลลัพธ์การค้นหากลับมายังเอเจนต์ SRC เพื่อส่งต่อให้แก่สมาชิกผู้ร้องขอข้อมูลต่อไป

การค้นหาในบางครั้งอาจจะไม่ประสบความสำเร็จ อย่างเช่น ใน $S_G$ ระบุว่ามีข้อมูลที่ต้องการอยู่ แต่ในความเป็นจริงไม่มีข้อมูลที่ต้องการอยู่ ซึ่งอาจจะเกิดจากการที่ข้อมูลนั้นถูกใช้งานจนเต็มความสามารถ หรือ ถูกถอดถอนออกจากการให้บริการก่อนหน้านี้ เมื่อการค้นหาพบความผิดพลาดลักษณะนี้ ระบบจะแจ้งให้เอเจนต์ปรับปรุงค่า $S_G$ เพื่อไม่ให้เกิดข้อผิดพลาดเกี่ยวกับการค้นหาอีก สังเกตได้ว่ากลไกการทำงานแบบนี้นอกจากจะเป็นการค้นหาข้อมูลแล้ว ยังเป็นการดูแลรักษาระบบอีกทางหนึ่ง

## 4.5 การเพิ่มประสิทธิภาพแก่กระบวนการค้นหาข้อมูล

วิธีการค้นหาข้อมูลเบื้องต้นหรือเรียกว่า Signature ในโครงสร้างเอเจนต์ที่ได้อธิบายไว้แล้วในหัวข้อก่อนหน้านี้สามารถลดภาระการเชื่อมต่อเพื่อค้นหาข้อมูลได้ส่วนหนึ่ง อย่างไรก็ดี การค้นหาด้วย Signature ก็ยังมีการส่งข้อมูลไปยังโหนดที่ไม่มีส่วนเกี่ยวข้องกับการค้นหาข้อมูลอยู่ ในหัวข้อนี้จะเป็นการปรับปรุงวิธีค้นหาข้อมูลให้สามารถค้นพบข้อมูลที่ต้องการได้เร็วขึ้น เป้าหมายก็คือ ลดการส่งข้อมูลไปยังเอเจนต์ตัวอื่นให้น้อยที่สุด

การปรับปรุงกลไกการค้นหาข้อมูลแบ่งออกเป็น 3 ขั้นตอน คือ การใช้แคช การใช้ความรู้ (Knowledge) และ การจำกัดขอบเขตการค้นหา กลไกการปรับปรุงระบบค้นหาข้อมูลทั้งสามแบบล้วนมีรากฐานมาจากวิธีการ Signature ทั้งสิ้น วิธีการทั้งสามจะถูกอิมพลีเมนต์ตามลำดับเพื่อให้ได้ระบบการค้นหาที่รวดเร็วและสมบูรณ์ที่สุด ซึ่งจะอธิบายรายละเอียดไว้ในส่วนถัดไปตามลำดับ

### 4.5.1 การใช้แคช

การเพิ่มประสิทธิภาพเบื้องต้นให้แก่ระบบการค้นหาก็ คือ การใช้แคช (Cache) โดยมีสมมติฐานว่า ข้อมูลหรือรีซอร์สใดก็ตามที่เคยมีการร้องขอใช้งานแล้วจากสมาชิก มีความเป็นไปได้สูงที่จะถูกเรียกใช้งานอีกครั้ง

เมื่อนำแนวคิดดังกล่าวมาใช้ในระบบการค้นหา เอเจนต์จะเก็บค่าข้อมูลที่เคยมีการร้องของไว้ในแคชซึ่งเป็นส่วนหนึ่งของเอเจนต์ ข้อมูลที่เก็บในแคชเหล่านี้จะคัดเลือกจากข้อมูลที่เคยค้นหาจากสมาชิกของเอเจนต์แต่ละตัว โดยจะมีการเก็บรวบรวมข้อมูลสถิติไว้ก่อน แล้วคัดเลือกเฉพาะข้อมูลที่มีการร้องขอบ่อยครั้ง โดยการปรับปรุง (update) แคชแต่ละครั้งจะทำเป็นช่วงเวลาที่ระบุแน่นอน

ในการอิมพลีเมนต์แคชในระบบการค้นหานี้ ค่า Signature $S_G$ ที่แสดงค่าข้อมูลรีซอร์สของเอเจนต์ปัจจุบันจะแตกต่างไปจากเดิมคือสมการ (4.4) เล็กน้อย โดยมีการเพิ่มการแสดงค่าข้อมูลที่อยู่ในแคชของเอเจนต์รวมเข้าไปด้วย กำหนดให้ $S_{cache}$ เป็น Signature แทนข้อมูลที่มีอยู่ในแคช $S_G$ ใหม่จะมีค่าดังนี้

$$S_G = \sum_{\forall C}(S_{GC} + S_{RC} + S_{cacheC}) \tag{4.5}$$

จะเห็นได้ว่าค่า $S_G$ จะรวมเอา Signature ของแคชเข้าไปด้วย สำหรับการค้นหาข้อมูลเมื่อสมาชิกค้นหาข้อมูล ก็จะใช้วิธีการเดียวกับการค้นหาปกติ การพบข้อมูลในแคชสามารถเปรียบเทียบได้จาก $S_G$ กับ $S_d$ ซึ่งเป็น Signature ของการค้นหานั่นเอง

การใช้แคชจะมีประโยชน์ก็ต่อเมื่อ สมาชิกในระบบมีการเรียกข้อมูลซ้ำๆ ในทางตรงกันข้าม ถ้าสมาชิกในระบบไม่มีการเรียกข้อมูลซ้ำเลย การใช้แคชก็จะไม่มีประโยชน์

4.5.2 **การประยุกต์ใช้** Knowledge

การใช้ Knowledge สำหรับระบบการค้นหาข้อมูล มีจุดประสงค์เพื่อจำนวนการเชื่อมต่อเพื่อสอบถามข้อมูลไปในโครงสร้างแบบลำดับชั้นโดยไม่จำเป็น วิธีการใช้ Knowledge นี้ประกอบด้วยการพิจารณาการส่งเชื่อมต่อ 3 ส่วน คือ Local Knowledge, Neighbor Knowledge และ Global Knowledge โดยการอิมพลีเมนต์จะมีความแตกต่างจากวิธีการค้นหาปกติ และ การใช้แคช กล่าวคือ จะมีการอ้างอิงค่า Signature $S_G$ มากกว่า 1 ตัว โดยมีการเก็บค่า $S_G$ ของเอเจนต์ที่เกี่ยวข้องเพื่ออ้างอิงข้อมูลไว้ใน K-PST ด้วย ข้อมูลเหล่านี้จะถูกนำมาเปรียบเทียบเพื่อการตัดสินใจเชื่อมต่อไปยังโหนดต่างๆ ในขณะค้นหาข้อมูลด้วยการใช้ Knowledge แบบต่างๆ รวมกัน โดยมีรายละเอียดดังต่อไปนี้

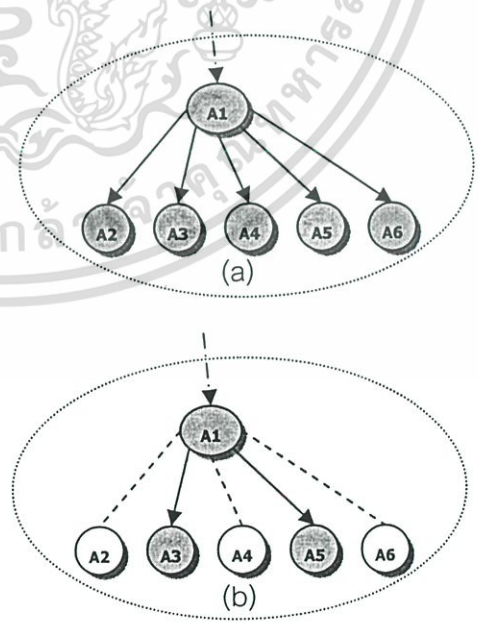
4.5.2.1 การใช้ Local Knowledge

ในระบบการค้นหาพื้นฐาน เมื่อมีการเปรียบเทียบค่า $S_d$ กับค่า $S_G$ ที่เอเจนต์ตัวใดตัวหนึ่งแล้วพบว่ามีความเป็นไปได้ที่จะพบข้อมูลที่ต้องการในโหนดลูก จะมีการส่ง $S_d$ ไปค้นหายังโหนดเหล่านั้นต่อไป หากโชคดีเส้นทางที่ส่งไปจะเป็นเส้นทางที่ถูกต้องซึ่งมีข้อมูลที่ต้องการอยู่ตามเส้นทางนั้น แต่ถ้าหากเอเจนต์ต้นทางมีโหนดลูกมาก การส่งข้อมูลส่วนใหญ่จะสูญเปล่าเนื่องจากข้อมูลที่ต้องการค้นหาจะไม่ได้อยู่ใต้สับทรีของเอเจนต์โหนดนั้น ดังนั้น การตัดสินใจได้ว่า จะมีการส่ง $S_d$ เพื่อค้นหาไปยังโหนดลูกโหนดใดบ้าง จะเป็นการลดจำนวนเมสเซสได้จำนวนหนึ่ง การตัดสินใจแบบนี้ต้องอาศัยข้อมูล Knowledge พื้นฐานคือ $S_G$ ของโหนดลูกๆ นั่นเอง

$S_d$ = 0100000000000000

**K-PST of $A_i$**

| $S_G$ | *Signature Value* |
|---|---|
| $A_1$ | 1111101011100101 |
| $A_2$ | 0000000000000101 |
| $A_3$ | 0111000011000101 |
| $A_4$ | 0000100000100101 |
| $A_5$ | 0100100000100101 |
| $A_6$ | 0000100000100101 |




**รูปที่** 4.15 การเชื่อมต่อเพื่อค้นหาข้อมูลระหว่างการค้นหาปกติกับการใช้ Local Knowledge (a) การค้นหาปกติ และ (b) การค้นหาที่อาศัย Local Knowledge

วิธีการ กำหนดให้

$A_c$ คือ เอเจนต์โหนดที่กำลังค้นหาข้อมูลอยู่

$A_i$ เป็นโหนดลูกของ $A_c$

$S_d$ เป็น Signature ของข้อมูลที่ต้องการค้นหา

สำหรับแต่ละ $S_{Gi}$ ที่เป็น Signature ของ $A_i$

ถ้า $S = S_{Gi}$ โดยที่ $S = (S_{Gi}$ AND $S_d)$ แล้ว ให้ส่ง $S_d$ ไปค้นหาต่อยัง $A_i$

สำหรับการค้นหาข้อมูลที่แสดงตัวอย่างไว้ในรูป 4.15 เมื่อเอเจนต์ $A_1$ ได้รับการเชื่อมต่อเพื่อสอบถามข้อมูลแล้ว จะนำเอา $S_d$ มาเปรียบเทียบกับ $S_G$ ของ $A_1$ ปรากฏว่าผลลัพธ์ที่ออกมามีค่าเท่ากับ $S_d$ หมายความว่ามีความเป็นไปได้ที่จะพบข้อมูลอยู่ใต้สับทรี สำหรับการใช้ Local Knowledge จะมีการเปรียบเทียบกับ $S_G$ ของเอเจนต์ลูกๆ ที่เก็บไว้ใน K-PST ด้วย เพื่อทราบว่าภายใต้สับทรีของเอเจนต์ตัวใดมีความเป็นไปได้ที่จะมีข้อมูลที่ต้องการค้นหาอยู่จริง แล้วจึงส่ง $S_d$ ไปยังเอเจนต์หล่านั้นได้แก่ $A_3$ และ $A_5$ ตามรูป 4.15(b) ซึ่งจะเห็นได้ว่าจำนวนการเชื่อมต่อจะน้อยกว่าการค้นหาแบบปกติที่ส่ง $S_d$ ไปค้นหายังลูกทุกโหนด

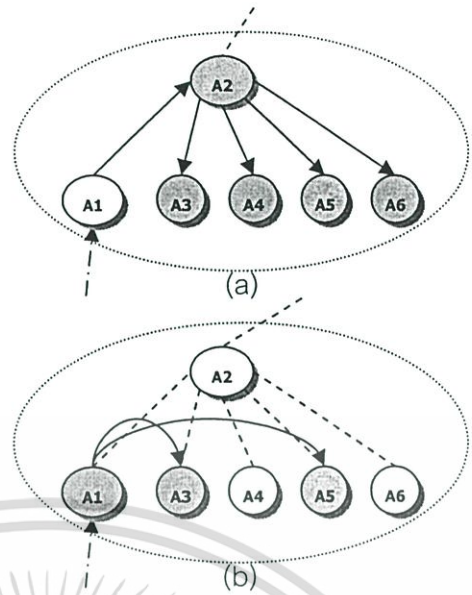
4.5.2.2 การใช้ Neighbor Knowledge

ในระบบการค้นหาพื้นฐาน เมื่อมีการเปรียบเทียบค่า $S_d$ กับค่า $S_G$ ที่เอเจนต์ตัวใดตัวหนึ่งแล้วไม่ว่าจะพบข้อมูลในโหนดนั้นหรือไม่ก็ตาม จะมีการส่ง $S_d$ ไปค้นหายังโหนดพ่อเพื่อค้นหาต่อไป เมื่อมีการเชื่อมต่อไปถึงโหนดพ่อก็จะเปรียบเทียบค่าหาเส้นทางในสับทรีอีกครั้ง หากโชคดีเส้นทางที่ส่งไปจะเป็นเส้นทางที่ถูกต้องซึ่งมีข้อมูลที่ต้องการอยู่ตามเส้นทางนั้น แต่ถ้าหากเอเจนต์โหนดพ่อทางมีโหนดลูกมาก การส่งข้อมูลส่วนใหญ่จะสูญเปล่าเนื่องจากข้อมูลที่ต้องการค้นหาจะไม่ได้อยู่ใต้สับทรีของเอเจนต์โหนดนั้น ดังนั้น การตัดสินใจได้ว่า จะส่ง $S_d$ ไปค้นหายังเพื่อนบ้านหรือโหนดที่มีพ่อตัวเดียวกับโหนดปัจจุบันโหนดใดบ้าง จะลดจำนวนการเชื่อมต่อได้จำนวนหนึ่ง การตัดสินใจแบบนี้ต้องอาศัยข้อมูล Neighbor Knowledge พื้นฐานคือ $S_G$ ของเอเจนต์เพื่อนบ้านนั่นเอง

$S_d$ = 0100000000000000

**K-PST of $A_1$**

| $S_G$ | ***Signature Value*** |
|---|---|
| $A_1$ | 0111001011000101 |
| $A_2$ | 1111101011100101 |
| $A_3$ | 0111000011000101 |
| $A_4$ | 0000100000100101 |
| $A_5$ | 0100100000100101 |
| $A_6$ | 0000100000100101 |




**รูปที่** 4.16 การเชื่อมต่อเพื่อค้นหาข้อมูลระหว่างการค้นหาปกติกับการใช้ Neighbor Knowledge (a) การค้นหาปกติ และ (b) การค้นหาด้วย Neighbor Knowledge

วิธีการ กำหนดให้

$A_c$ คือ เอเจนต์โหนดที่กำลังค้นหาข้อมูลอยู่

$A_i$ เป็นโหนดที่มีโหนดพ่อตัวเดียวกับ $A_c$

$S_d$ เป็น Signature ของข้อมูลที่ต้องการค้นหา

สำหรับแต่ละ $S_{Gi}$ ซึ่งเป็น Signature ของ $A_i$ ถ้า $S = S_{Gi}$ โดยที่ $S = (S_{Gi}$ AND $S_d)$ แล้ว ให้ส่ง $S_d$ ไปค้นหาต่อยัง $A_i$

สำหรับการค้นหาข้อมูลที่แสดงตัวอย่างไว้ในรูป 4.16 เมื่อเอเจนต์ $A_1$ ได้รับการเชื่อมต่อเพื่อสอบถามข้อมูลและตรวจสอบกับโหนดของตนเรียบร้อยแล้ว ในขั้นตอนการตัดสินใจส่ง $S_d$ ไปค้นหาต่อยังโหนดอื่นๆ สำหรับการใช้ Neighbor Knowledge จะมีการเปรียบเทียบกับ $S_G$ ของเอเจนต์ที่มีโหนดพ่อตัวเดียวกับ $A_1$ ที่เก็บไว้ใน K-PST ด้วย เพื่อทราบว่าภายใต้สับทรีของเอเจนต์ตัวใดมีความเป็นไปได้ที่จะมีข้อมูลที่ต้องการค้นหาอยู่จริง แล้วจึงส่ง $S_d$ ไปยังเอเจนต์หล่านั้นได้แก่ $A_3$ และ $A_5$ ตามรูป 4.16(b) ซึ่งจะเห็นได้ว่าจำนวนการส่งจะน้อยกว่าการค้นหาแบบปกติที่ส่ง $S_d$ ไปค้นหายังลูกทุกโหนด วิธีการนี้จะลดจำนวนการเชื่อมต่อไปยังโหนดพ่อและโหนดเพื่อนบ้านได้จำนวนหนึ่ง อย่างเช่นในตัวอย่าง จะไม่มีการเชื่อมต่อไปตรวจสอบข้อมูลที่โหนดพ่อคือ $A_2$ และโหนดเพื่อนบ้านที่ไม่มีความเป็นไปได้ที่จะพบข้อมูลคือโหนด $A_4$ และ $A_6$

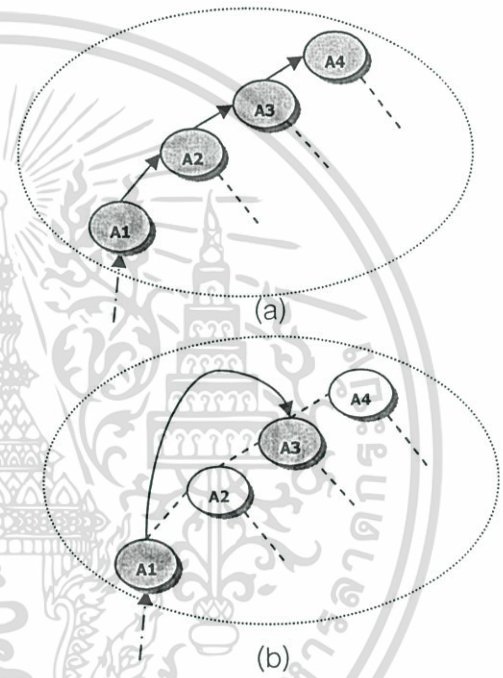
4.5.2.3 **การใช้** Global Knowledge

ในระบบการค้นหาพื้นฐาน เมื่อมีการเปรียบเทียบค่า $S_d$ กับค่า $S_G$ ที่เอเจนต์ตัวใดตัวหนึ่งแล้วไม่ว่าจะพบข้อมูลในโหนดนั้นหรือไม่ก็ตาม จะมีการส่ง $S_d$ ไปค้นหายังโหนดพ่อเพื่อค้นหาต่อไป เมื่อเมสเซสส่งถึงโหนดพ่อก็จะก็จะทำตามวิธีการเดิมโดยส่งไปยังโหนดพ่อเรื่อยไปจนถึงรูทโหนด การเชื่อมต่อส่วนใหญ่จะสูญเปล่าเนื่องจากข้อมูลที่ต้องการค้นหาจะไม่ได้ตามเส้นทางที่ค้นหา ดังนั้น การตัดสินใจได้ว่า จะมีการส่ง $S_d$ เพื่อค้นหาไปยังโหนดที่มีลำดับชั้นเหนือขึ้นไปตัวใดบ้าง จะเป็นการลดจำนวนการเชื่อมต่อได้จำนวนหนึ่ง การตัดสินใจแบบนี้ต้องอาศัยข้อมูล Global Knowledge พื้นฐานคือ $S_G$ ของเอเจนต์โหนดพ่อทั้งหลายนั่นเอง

$S_d$ = 0100000000000001

**K-PST of A1**

| $S_G$ | ***Signature Value*** |
|---|---|
| $A_1$ | 0000000000000101 |
| $A_2$ | 0000000000000101 |
| $A_3$ | 0111000011000101 |
| $A_4$ | 0111100011100101 |




**รูปที่** 4.17 การเชื่อมต่อเพื่อค้นหาข้อมูลระหว่างการค้นหาปกติกับการใช้ Global Knowledge (a) การค้นหาปกติ และ (b) การค้นหาที่อาศัย Global Knowledge

<u>วิธีการ</u> กำหนดให้

$A_c$ คือ เอเจนต์โหนดที่กำลังค้นหาข้อมูลอยู่

$A_i$ เป็นโหนดที่มีลำดับชั้นสูงขึ้นไปตามสายการเชื่อมต่อของ $A_c$

$S_d$ เป็น Signature ของข้อมูลที่ต้องการค้นหา

สำหรับแต่ละ $S_{Gi}$ ซึ่งเป็น Signature ของ $A_i$

ถ้า $S = S_{Gi}$ โดยที่ $S = (S_{Gi}$ AND $S_d)$ แล้ว ให้ส่ง $S_d$ ไปค้นหาต่อยัง $A_i$ ที่มีลำดับชั้นใกล้กับ $A_c$ มากที่สุด

สำหรับการค้นหาข้อมูลที่แสดงตัวอย่างไว้ในรูป 4.17 เมื่อเอเจนต์ $A_1$ ได้รับเมสเซสสอบถามข้อมูลและตรวจสอบกับโหนดของตนเรียบร้อยแล้ว ในขั้นตอนการตัดสินใจส่ง $S_d$ ไปค้นหาต่อยังโหนดอื่นๆ สำหรับการใช้ Global Knowledge จะมีการเปรียบเทียบกับ $S_G$ ของเอเจนต์ที่มีโหนดที่มีลำดับชั้นสูงขึ้นไปตามสายใยการเชื่อมต่อของ $A_1$ ซึ่งถูกเก็บไว้ใน K-PST ด้วย เพื่อทราบว่าภายใต้สับทรีของเอเจนต์ตัวใดมีความเป็นไปได้ที่จะมีข้อมูลที่ต้องการค้นหาอยู่จริง แล้วจึงส่ง $S_d$ ไปยังเอเจนต์หล่านั้นได้แก่ $A_3$ ตามรูป 4.17(b) ซึ่งจะเห็นได้ว่าจำนวนการส่งจะน้อยกว่าการค้นหาแบบปกติที่ส่ง $S_d$ ไปค้นหายังลูกทุกโหนดในลำดับชั้น วิธีการนี้จะลดจำนวนการส่งข้อมูลไปยังโหนดพ่อและโหนดเพื่อนบ้านได้จำนวนหนึ่ง อย่างเช่นในตัวอย่าง จะไม่มีการเชื่อมต่อไปตรวจสอบข้อมูลที่โหนดพ่อคือ $A_2$ และ $A_4$ เป็นต้น

#### 4.5.2.4 การใช้ Knowledge ร่วมกัน

ในระบบการค้นหาด้วย Knowledge ที่สมบูรณ์จะนำเอาวิธีการการใช้ Knowledge ทั้งสามประเภทที่กล่าวมาแล้วมาใช่ร่วมกัน เพื่อทำงานในกระบวนการค้นหาข้อมูลในโครงสร้างลำดับชั้นของเอเจนต์ การค้นหาด้วยวิธีการใช้ Knowledge นี้เป็นการค้นหาแบบครบถ้วนโดยมีการสร้างเชื่อมต่อไปค้นหาข้อมูลยังทุกๆ โหนดที่มีความเป็นไปได้ที่จะมีข้อมูลที่ต้องการอยู่ และอาศัยโครงสร้างข้อมูล K-PST เพื่อการค้นหา ดังนั้น ผู้วิจัยจึงได้ตั้งชื่อวิธีการค้นหานี้เพื่อใช้ในการอ้างอิงว่า "K-PST(all)" ลักษณะของการทำงานของ K-PST(all) มีวิธีการดังนี้

<u>วิธีการ</u> กำหนดให้

$A_c$ คือ เอเจนต์โหนดที่กำลังค้นหาข้อมูลอยู่

$A_i$ เป็นโหนดลูกของ $A_c$

$A_j$ เป็นโหนดที่มีโหนดพ่อตัวเดียวกับ $A_c$

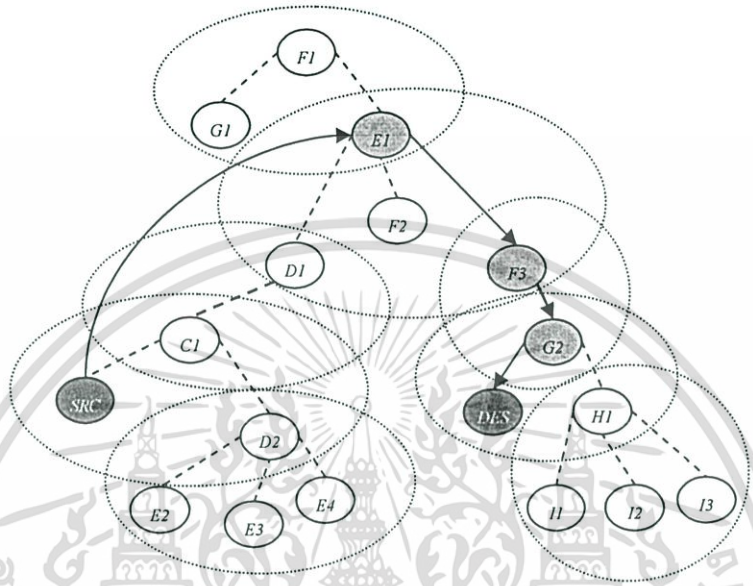
$A_k$ เป็นโหนดที่มีลำดับชั้นสูงขึ้นไปตามสายการเชื่อมต่อของ $A_c$

$S_d$ เป็น Signature ของข้อมูลที่ต้องการค้นหา

การค้นหาจะตรวจสอบการเชื่อมต่อตามลำดับดังนี้

1) สำหรับแต่ละ $S_{Gi}$ ซึ่งเป็น Signature ของ $A_i$
   ถ้า $S = S_{Gi}$ โดยที่ $S = (S_{Gi}$ AND $S_d)$ แล้ว ให้ส่ง $S_d$ ไปค้นหาต่อยัง $A_i$
2) สำหรับแต่ละ $S_{Gj}$ ซึ่งเป็น Signature ของ $A_j$
   ถ้า $S = S_{Gj}$ โดยที่ $S = (S_{Gj}$ AND $S_d)$ แล้ว ให้ส่ง $S_d$ ไปค้นหาต่อยัง $A_j$
3) สำหรับแต่ละ $S_{Gk}$ ซึ่งเป็น Signature ของ $A_k$
   ถ้า $S = S_{Gk}$ โดยที่ $S = (S_{Gk}$ AND $S_d)$ แล้ว ให้ส่ง $S_d$ ไปค้นหาต่อยัง $A_k$ ที่มีลำดับชั้นใกล้กับ $A_c$ มากที่สุดและไม่ใช่โหนดพ่อของ $A_c$

การตรวจสอบทั้ง 3 ข้อก็คือ การใช้ Local Neighbor และ Global Knowledge ตามลำดับนั่นเอง สังเกตว่าวิธีการในข้อ 3 จะแตกต่างจากวิธีการตรวจสอบของ Global Knowledge ปกติ คือจะไม่มีการตรวจสอบข้อมูลในโหนดพ่อของโหนดการค้นหาปัจจุบัน เนื่องจาก การค้นหาในโหนดพ่อจะถูกค้นหาด้วยวิธีการของ Neighbor Knowledge ในข้อ 2 ไปแล้วในตัวนั่นเอง




**รูปที่** 4.18 การค้นหาข้อมูลในโครงสร้างลำดับชั้นของเอเจนต์ด้วยการใช้ Knowledge

ในรูปที่ 4.18 แสดงตัวอย่างการค้นหาด้วยการใช้ Knowledge โครงสร้างลำดับชั้นในรูปเป็นโครงสร้างเดียวกับรูปที่ 4.14 จะเห็นได้ว่าจะมีเอเจนต์ที่เกี่ยวข้องกับการค้นหาเพียง 5 ตัวเท่านั้นคือ SRC, E1, F3, G2 และ DES ซึ่งเป็นเจ้าของข้อมูลที่สมาชิกต้องการค้นหา ในขณะที่การค้นหาแบบพื้นฐานในรูปที่ 4.14 จะมีเอเจนต์ที่เกี่ยวข้องถึง 11 ตัว

### 4.4.3 การจำกัดขอบเขตการค้นหา

วิธีการ K-PST(all) เป็นการค้นหาข้อมูลแบบครบถ้วน คือมีการค้นหาไปยังทุกโหนดที่มีความเป็นไปได้ว่าจะมีข้อมูลที่ต้องการอยู่ แน่นอนว่าต้องมีการเชื่อมต่อเพื่อสอบถามข้อมูลจำนวนมาก ดังนั้น ถ้ามีการจำกัดขอบเขตการค้นหา โดยค้นหาข้อมูลแบบเดียวกันเพียงส่วนหนึ่งของระบบ ก็จะสามารถลดจำนวนการเชื่อมต่อไปค้นหาได้

สำหรับการทำงานของวิธีการเพิ่มประสิทธิภาพด้วยการจำกัดจำนวนการค้นหาข้อมูลนี้ จะใช้ระบบการค้นหาที่ใช้ Knowledge ในหัวข้อ 4.3 เป็นพื้นฐานในการค้นหา โดยจะแตกต่างจากวิธีการดังกล่าวซึ่งจะส่งข้อมูล $S_d$ ไปค้นหายังทุกโหนด แต่จะเลือกเส้นทางที่เป็นไปได้ส่วนหนึ่งแทน จำนวนเส้นทางที่เลือกนี้จะขึ้นอยู่กับว่าระบบต้องการค้นหาข้อมูลเป็นจำนวนอย่างน้อยเท่าไร

เราเรียกวิธีการค้นหาแบบนี้ว่า "K-PST($N_r$)"โดยที่ $N_r$ คือ จำนวนข้อมูลที่ต้องการค้นหาเป็นอย่างน้อย วิธีการค้นหาข้อมูลด้วยการจำกัดขอบเขตการค้นหาข้อมูล K-PST($N_r$) มีการทำงานดังนี้

<u>กำหนดให้</u>

$A_i$ โดยที่ ($i$ = 1, 2, ..., $n$) เป็นเอเจนต์

$S_d$ คือ Signature ของข้อมูลหรือบริการที่ต้องการค้นหา

$A_i(S_d)$ คือ การค้นหาที่เอเจนต์ตัวที่ $i$

$R(S_d)$ คือ การตรวจสอบข้อมูล $S_d$ ในข้อมูลที่เอเจนต์ครอบครองอยู่

$S(S_d)$ คือ การตรวจสอบข้อมูล $S_d$ ใน Signature ของข้อมูล

$E(N_r)$ คือ การตรวจสอบจำนวนข้อมูลที่ต้องการค้นหาต่อไป

โดยที่ $R(S_d), S(S_d), \in \{ A_i (i = 1, 2, ..., n), null \}$

และ $E(N_r) \in$ จำนวนเต็ม

*null* หมายถึง ไม่พบข้อมูล

* หมายถึง ไม่พิจารณา

<u>กฎการค้นหาในโครงสร้างลำดับชั้น</u>

(1) $A_i(S_d, N_r) \Longrightarrow A_i \rightarrow (R(S_d), S(S_d), E(N_r))_{Ai}$

(2) $(A_{This}, *, *)_{This} \Longrightarrow$ SERVICE_FOUND $\rightarrow$

(3) $(*, A_j, N_r > 0)_{This} \Longrightarrow A_j(S_d, N_r)$

(4) $(*, *, N_r = 0)_{This} \Longrightarrow$ END

(5) $(*, null, *)_{Root} \Longrightarrow$ END

(6) $(*, null, *)_{Leaf} \Longrightarrow$ END

วิธีการคำนวณ $E(N_r)$ กำหนดให้ $N_r$ เป็นจำนวนข้อมูลที่ต้องการค้นหา

การค้นหาของแต่ละโหนดมีการทำงานดังนี้

1) ถ้าพบข้อมูลที่ต้องการในโหนดปัจจุบัน

1.1) $N_r$ <- $N_r$ - 1

1.2) ถ้า $N_r = 0$ ให้ส่งผลลัพธ์กลับไปยังต้นทาง และไปที่ข้อ 4)

2) หาเส้นทางที่เป็นไปได้ทั้งหมด $D$ ด้วยวิธีการในข้อ 4.4.2.4

3) ตรวจสอบการเชื่อมต่อเพื่อค้นหาต่อยังโหนดอื่น

3.1) ถ้า $N_r < D$

ให้เลือกเส้นทางที่ดีที่สุด $N_r$ อันดับแรก แล้วสร้างการเชื่อมต่อเพื่อค้นหาไปยังโหนดนั้น โดยโหนดมีความสำคัญตามลำดับดังนี้ สับทรี > เพื่อนบ้าน > พ่อ แต่ละการเชื่อมต่อที่ส่งไปจะมีค่า $N_r = 1$

3.2) ถ้า $N_r \geq D$

ให้ส่งข้อมูลการค้นหาไปยังทุกทิศทาง แต่ละการเชื่อมต่อที่ส่งไปจะมีค่า $N_r = N_r / D$

4) จบการค้นหาและคืนค่า $N_r$

จากวิธีการที่อธิบายไว้ด้านบน การค้นหา K-PST($N_r$) นี้ จะมีการค้นหาข้อมูลไปจนครบตามที่จำนวนที่ผู้ใช้ร้องขอเป็นอย่างน้อย ถือว่าเป็นการลดจำนวนการเชื่อมต่อเพื่อค้นหาข้อมูลในระบบจำนวนหนึ่งแต่ยังสามารถตอบสนองความต้องการของสมาชิกได้ ในส่วนการเชื่อมต่อ จะมีการเลือกเส้นทางสำหรับค้นหาข้อมูลโดยที่เลือกส่งข้อมูลไปยังสับทรีก่อนเพื่อนบ้านและโหนดพ่อ เนื่องจากในกลไกการลงทะเบียนที่ออกแบบขึ้นจะระบุว่า เอเจนต์ในระดับล่างจะถูกกำหนดให้มีระยะทางใกล้กว่าระดับบนของโครงสร้างลำดับชั้น อีกทั้งสมมติฐานของการค้นหาข้อมูลผู้ใช้บริการมักจะต้องการข้อมูลที่อยู่ในระดับพื้นที่โลคอลมากกว่าข้อมูลที่อยู่ในระยะไกลในระดับกลอบอล

ในการค้นหาข้อมูลเพียง 1 รายการ หรือ K-PST(one) ซึ่งเป็นการค้นหาที่น่าจะมีการใช้งานสูงสุด จะให้ประสิทธิภาพการค้นหาที่ดีมาก ในกรณีที่แย่ที่สุด ที่ต้นทางร้องขอข้อมูลที่มีอยู่ในเอเจนต์ลีฟโหนดอีกด้านหนึ่งของทรี ระบบที่ออกแบบขึ้นจะสามารถค้นหาข้อมูลด้วยการสร้างการเชื่อมต่อเป็นจำนวนเท่ากับจำนวนลำดับชั้นของเอเจนต์รวมกับการเชื่อมต่อที่ส่งจากเอเจนต์ต้น

ทางไปยังรูทโหนด มีค่าเท่ากับ $\log(N) + 1$ คิดเป็น $O(\log N)$ สามารถลดการเชื่อมต่อเพื่อสอบถามไปในระบบ โดยไม่มีการส่งข้อมูลไปยังโหนดที่ไม่สอดคล้องได้อีกจำนวนหนึ่ง

## 4.6 การประเมินประสิทธิภาพกลไกการค้นหาข้อมูล

ในหัวข้อนี้เป็นการวิเคราะห์ประสิทธิภาพของวิธีการค้นหาในแง่มุมที่แตกต่างกัน ได้แก่ ความเร็วในการค้นหา จำนวนพื้นที่เก็บข้อมูลที่ใช้ และจำนวนการประกาศข้อมูลแต่ละครั้ง วิธีการค้นหาที่นำมาพิจารณา คือ Flooding ซึ่งเป็นวิธีการค้นหาแบบพื้นฐานที่ใช้ในระบบการค้นหาข้อมูลทั่งไป เปรียบเทียบกับวิธีการค้นหาที่นำเสนอในงานวิจัยนี้ อันได้แก่ 1) Signature ที่อธิบายไว้ในหัวข้อ 4.4 ที่เป็นการค้นหาต้นแบบที่มีการค้นหาแบบครบถ้วน 2) K-PST(all) ที่อธิบายไว้ในหัวข้อ 4.5.2 และ 3) การค้นหา K-PST(one) ที่มีการจำกัดขอบเขตการค้นหาเหลือเพียง 1 ชุด โดยความซับซ้อนของกระบวนการในแง่มุมต่างๆ ได้สรุปการเปรียบเทียบไว้ในตารางที่ 4.1

**ตารางที่ 4.1** แสดงข้อดีข้อเสียของวิธีการค้นหาข้อมูลแบบต่างๆ

| | **Flooding** | **Signature** | **K-PST(all)** | **K-PST(one)** |
|---|---|---|---|---|
| **Time** | $N$ | $kd \log_k N$ | $d \log_k N$ | $\log_k N + 1$ |
| **Space Overhead** | $C$ | $S$ | $(2k + \log_k N)\, S$ | $(2k + \log_k N)\, S$ |
| **Advertising** | 0 | $N$ | $2N$ | $2N$ |

นิยาม

$N$ คือ จำนวนโหนดของเอเจนต์ในระบบ

$k$ คือ จำนวนลูกของพ่อในโครงสร้างลำดับชั้น

$s$ คือ ข้อมูลหรือรีซอร์สที่ต้องการค้นหา

$d$ คือ จำนวนเส้นทางที่แตกต่างกันเพื่อค้นหาข้อมูล $s$

$M$ คือ จำนวนที่อยู่ที่แตกต่างกันของข้อมูล $s$

$S$ คือ ขนาดของ Signature หน่วยเป็น (bits)

$C$ คือ จำนวนของโหนดเพื่อนบ้านที่เชื่อมต่ออยู่

$L$ คือ จำนวนชั้นในโครงสร้างลำดับชั้น

$U$ คือ จำนวนอีเลเมนต์ทั้งหมดในยูนิเวอร์ส

$n$ คือ จำนวนอีเลเมนต์ใน Signature จากทั้งหมด $U$

$m$ คือ จำนวนบิตแอร์เรย์ใน Signature

### 4.6.1 ความเร็วการค้นหา

การค้นหาด้วยวิธีการต่างๆ จะมีความซับซ้อนด้านเวลาการค้นหาดังนี้

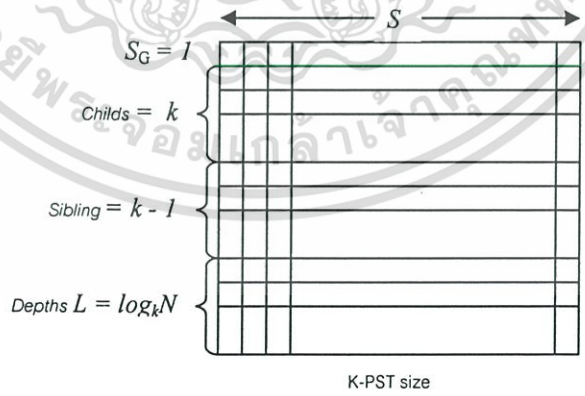
การค้นหาด้วย Flooding จำเป็นต้องเชื่อมต่อไปยังโหนดเพื่อนบ้านของตนทุกๆ โหนด และกระจายกันค้นหาไปในวงรัศมี เมื่อมีการค้นหาแบบครบถ้วน จำเป็นต้องค้นหาไปยังทุกโหนด $N$ ดังนั้น ความซับซ้อนจึงอยู่ที่ $O(N)$

การค้นหาด้วย Signature จำเป็นต้องเชื่อมต่อไปยัง Childs ทุกตัวเป็นจำนวน $k$ เมื่อพบว่า $s \in S_G$ สำหรับค้นหาข้อมูลเป็นจำนวน 1 ชุด และ แต่ละชุดต้องอาศัยจำนวนเส้นทางที่แตกต่างกันเป็นจำนวน $d$ โดยที่ $d \le M \le N$ (บางกรณี $s$ ในที่ต่างๆ จะอยู่ในเส้นทางเดียวกัน) และการค้นหาไปยังโครงสร้างลำดับล่างแต่ละเส้นทาง ใช้การเชื่อมต่อเป็นจำนวน $\log_k N$

การค้นหาด้วย K-PST(all) จำเป็นต้องเชื่อมต่อเพื่อค้นหาข้อมูล $M$ ชุดไปยังโครงสร้างระดับล่าง ใช้จำนวนเส้นทางอย่างมากที่สุดเท่ากับจำนวนเส้นทางที่แตกต่างกันเป็นจำนวน $d$ โดยที่ และการค้นหาข้อมูลแต่ละเส้นทางในโครงสร้างลำดับชั้น ใช้การเชื่อมต่อเป็นจำนวน $\log_k N$

การค้นหาด้วย K-PST(one) จำเป็นต้องเชื่อมไปยัง Parent เป็นจำนวนอย่างมากเท่ากับ 1 ครั้ง รวมกับ การเชื่อมต่อเพื่อค้นหาข้อมูลเป็นจำนวนเท่ากับ $d = M = 1$ ดังนั้น การค้นหาข้อมูลแต่ละเส้นทางในโครงสร้างลำดับชั้น กรณีที่แย่ที่สุดก็คือ ใช้การเชื่อมต่อเป็นจำนวน $\log_k N$

### 4.6.2 ความสิ้นเปลือง




**รูปที่** 4.19 แสดงขนาดของ K-PST summaries

ขนาดของ routing table ในวิธีการ Flooding จะมีขนาดขึ้นอยู่กับจำนวนโหนดที่เชื่อมต่ออยู่ ณ ขณะนั้น แตกต่างจาก ขนาดของ Signature แทนด้วย $S$ ที่ขึ้นอยู่กับจำนวนอีเลเมนต์ของ

ข้อมูลที่จะเก็บในบิตแอร์เรย์ ซึ่งมีค่าเท่ากับ $S = \alpha U$ โดยที่ $\alpha = \frac{m}{n}$ หมายถึง อัตราส่วนจำนวนบิตต่ออีเลเมนต์ หาก $\alpha$ มีค่ามาก จะทำให้ค่า Positive False Probability มีค่าน้อย เมื่อจำนวน hash function $k$ มีค่าคงที่

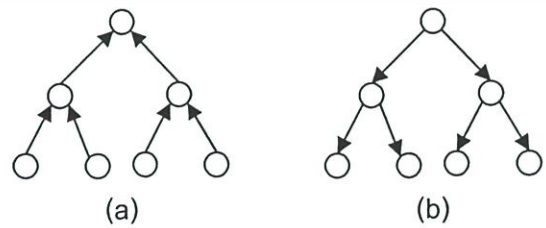
ขนาดการเก็บข้อมูลของ K-PST ทั้งสองมีค่าเท่ากับ $(2k + \log_k N)\ S$ ดังรูปที่ 4.19 ซึ่งได้มาจากขนาดของ Signature $S$ คูณกับจำนวนโหนดอื่นๆ ที่เชื่อมต่ออยู่กับเอเจนต์ ได้แก่ parents, childs และ sibling จะเห็นได้ว่า หากจำนวนลูก $k$ มีค่ามากจะทำให้ขนาดของ K-PST มีค่ามากขึ้นไปด้วย

### 4.6.3 การประกาศข้อมูล

สำหรับการค้นหาด้วยวิธี Flooging มีข้อดีอย่างหนึ่งคือ จะไม่มีการประกาศข้อมูลให้โหนดอื่นได้รับทราบ แต่ในการค้นหาด้วยวิธี Signature และ K-PST จำเป็นต้องมีการประกาศข้อมูลให้โหนดอื่นได้รับทราบข้อมูลในระบบ โดยปกติจะมีการประกาศข้อมูล ด้วยวิธีการเมิร์จ $S_G$ ในลำดับชั้นตามช่วงเวลาที่กำหนด นอกจากนี้ ในวิธี Signature พื้นฐาน และวิธี K-PST ก็มีการประกาศข้อมูลโดยอาศัยการเมิร์จ $S_G$ ที่แตกต่างกันด้วย

**รูปที่** 4.20 แสดงกระบวนการ Advertising ของ Signature ซึ่งทำแบบ bottom-up

ในการค้นหาด้วยวิธี Signature มีการประกาศข้อมูลด้วยการเมิร์จ $S_G$ โดยทำแบบ bottom-up ดังรูปที่ 4.20 เนื่องจาก Summaries ของ Signature ไม่จำเป็นต้องทราบ $S_G$ ของเอเจนต์ $A_i$ ที่มีลำดับสูงกว่า ดังนั้น จำนวนเอเจนต์ที่เกี่ยวข้องเพื่อการประกาศข้อมูล จึงมีค่าเท่ากับ $N$ ต่อกระบวนการประกาศข้อมูล 1 ครั้ง




**รูปที่** 4.21 แสดงกระบวนการ Advertising ของ K-PST (a) bottom-up (b) top-down

ส่วนการค้นหาแบบ K-PST นอกจากจะต้องทำการการเมิร์จ $S_G$ แบบ bottom-up แล้ว ยังต้องมีการส่งข้อมูลของ Summaries ไปยังโหนดลูกด้วย เนื่องจาก Summaries ของ K-PST จำเป็นต้องทราบ $S_G$ ของเอเจนต์ $A_i$ ที่มีลำดับสูงกว่าและลำดับเดียวกัน ดังนั้น จำนวนเอเจนต์ที่เกี่ยวข้องเพื่อการประกาศข้อมูล จึงมีค่าเท่ากับ $2N$ ต่อกระบวนการประกาศข้อมูล 1 ครั้ง

### 4.6.4 สรุปการประเมินประสิทธิภาพกลไกการค้นหา

จากการประเมินประสิทธิภาพที่กล่าวมาจะเห็นได้ว่า การค้นหาแต่ละแบบจะมีข้อดีข้อเสียที่แตกต่างกัน ระบบการค้นหา K-PST จะให้ความเร็วที่ดีกว่า แต่ก็ต้องมีการจัดเก็บข้อมูลที่มีขนาดใหญ่กว่าและต้องมีการเชื่อมต่อเพื่อการประกาศข้อมูลจำนวนมากตามไปด้วยตามไปด้วย ในขณะที่ Signature ให้ความเร็วที่แย่กว่า แต่ว่ามีข้อดีตรงที่ใช้ขนาดของพื้นที่เก็บข้อมูลเพื่อการค้นหาและการเชื่อมต่อข้อมูลน้อยกว่า K-PST สุดท้าย วิธี Flooding แม้ว่าเป็นวิธีที่ช้าที่สุด แต่ว่ามีข้อดีตรงที่ไม่จำเป็นต้องมีการประกาศข้อมูลเลย ดังนั้น หากจำเป็นต้องมีการนำเอาวิธีการเหล่านี้ไปใช้ในระบบจริง ข้อดีข้อเสียเหล่านี้จึงเป็นกุญแจสำคัญในการเลือกใช้วิธีการที่เหมาะสมต่อระบบการค้นหาแต่ละระบบได้เป็นอย่างดี

# บทที่ 5
# การประเมินประสิทธิภาพของระบบการค้นหาบริการอย่างรวดเร็วสำหรับระบบประมวลผลแบบกริด

ในบทนี้จะเป็นการประเมินประสิทธิภาพของระบบการค้นหาที่ได้นำเสนอขึ้นมา โดยจะกล่าวถึง การจำลองผล มาตรวัดประสิทธิภาพของกระบวนการค้นหาข้อมูลสำหรับระบบประมวลผลแบบกริด และการทดลองที่ศึกษาปัจจัยต่างๆ ในระบบ รวมทั้งบทวิเคราะห์ ตามลำดับ

## 5.1 การจำลองผลเพื่อการทดสอบประสิทธิภาพ

ในงานวิจัยนี้ นำเอาทฤษฎี Queuing [45] อธิบายกระบวนการร้องขอข้อมูลในระบบ มีความเกี่ยวข้องกับการให้บริการการค้นหาของเอเจนต์ที่ทำหน้าที่เป็นดิสคัฟเวอรีเซอร์เวอร์ และการเข้ามาของการร้องขอข้อมูลของสมาชิกในระบบทฤษฎี Queuing นี้จะถูกอิมลิเมนต์ด้วยโมเดลการจำลองผลที่ชื่อว่า Discrete Time Event Model [45, 46] เพื่อวิเคราะห์ประสิทธิภาพของระบบการค้นหาข้อมูลด้วยคอมพิวเตอร์ โมเดลการจำลองผลนี้เป็นการจำลองผลรูปแบบหนึ่งที่มีความน่าเชื่อถือ และถูกนำมาใช้อย่างแพร่หลายในงานวิจัยทางด้านการสื่อสารข้อมูลเครือข่ายคอมพิวเตอร์

### 5.1.1 กระบวนการจำลองผลด้วย Discrete Time Event Model

Discrete Time Event Simulation เป็นวิธีการจำลองผลแบบหนึ่งที่ใช้พยากรณ์ประสิทธิภาพของระบบเครือข่าย โดยพิจารณาความสัมพันธ์ระหว่างเวลา กับเหตุการณ์ที่มีการเปลี่ยนแปลงค่าของสเตทในจุดเวลาที่แตกต่างกันในช่วงเวลาใดเวลาหนึ่งที่สนใจ จุดเวลาจุดหนึ่งสามารถเกิดเหตุการณ์ได้หนึ่งเหตุการณ์ ซึ่งอาจจะเปลี่ยนแปลงค่าสเตทของระบบโดยรวมได้ แม้ว่ากระบวนการที่กล่าวมานี้สามารถคำนวณได้ด้วยมือ แต่เนื่องจากในระบบส่วนใหญ่จะเป็นต้องเกี่ยวข้องกับการเก็บข้อมูล และมีการคำนวณจำนวนมาก วิธีการนี้จึงเป็นที่นิยมนำคอมพิวเตอร์มาประยุกต์ใช้ในการคำนวณเสียมากกว่า

**วิธีการเพิ่มค่าเวลาของระบบ**

การจำลองผลด้วย Discrete Time Event Model นั้น กระบวนการทดลองจะมีการตรวจตราค่าสเตทและการจัดเรียงเหตุการณ์ที่เข้ามาทำงานในระบบ โดยอาศัยค่าเวลากลางของระบบที่เรียกว่า "Simulation Clock" เป็นตัวอ้างอิงร่วมกัน ค่าเวลานี้จำเป็นต้องระบุหน่วยต้องให้

สอดคล้องกับค่าเวลาของการเข้ามาของการร้องขอ (inter arrival time) และ ค่าเวลาการประมวลผลการร้องขอของเซิร์ฟเวอร์ (service time)

การเพิ่มค่าของ Simulation Clock สามารถทำได้สองวิธีคือ next-event time-advance และ fixed-increment time-advance แต่ส่วนใหญ่จะใช้วิธี next-event time-advance มากกว่า เนื่องจากเหมาะกับการทดสอบระบบโดยทั่วไปได้ ในขณะที่ fixed-increment time-advance มักจะนำมาใช้ทดสอบระบบเพื่อจุดประสงค์เฉพาะอย่างเท่านั้น อนึ่ง ในระบบจำลองแบบ Discrete Time Event ในการทดลองของวิทยานิพนธ์นี้จะใช้วิธีการเพิ่มค่า Simulation Clock ในด้วยวิธี next-event time-advance

สำหรับวิธีการเพิ่มค่าเวลาแบบ next-event time-advance จะเริ่มต้นค่า Simulation Clock ให้เท่ากับ 0 จากนั้น จะกำหนดค่าให้ Simulation Clock เพิ่มค่าขึ้นไปตามค่าเหตุการณ์ที่เข้ามาในอนาคตที่ใกล้ที่สุด ณ จุดนี้ ทำการจัดเรียงเหตุการณ์ที่เข้ามาทำงานในระบบ รวมทั้งอัพเดทค่าสเตทของระบบให้ทันต่อเหตุการณ์ วิธีการเพิ่มค่า Simulation Clock จะดำเนินการไปเรื่อยๆ และสิ้นสุดเมื่อมีการหยุดการจำลองระบบ ข้อสังเกตอย่างหนึ่งของวิธีการเพิ่มค่า Simulation Clock แบบนี้ คือ จะมีการกระโดดของค่าเวลา Simulation Clock จากค่าหนึ่งไปเป็นอีกค่าหนึ่งเป็นค่าไม่แน่นอน ขึ้นกับการเข้ามาของเหตุการณ์ ซึ่งแตกต่างจากวิธี fixed-increment time-advance ที่การเพิ่มค่าของ Simulation Clock จะมีค่าที่แน่นอนตามช่วงเวลาที่กำหนด

เพื่อให้ง่ายต่อความเข้าใจ จะแสดงตัวอย่างการเพิ่มค่า Simulation Clock ด้วยวิธี next-event time-advance ดังนี้

กำหนดให้

$t_i$ = เวลาที่เข้ามาของการร้องขอลำดับที่ $i$

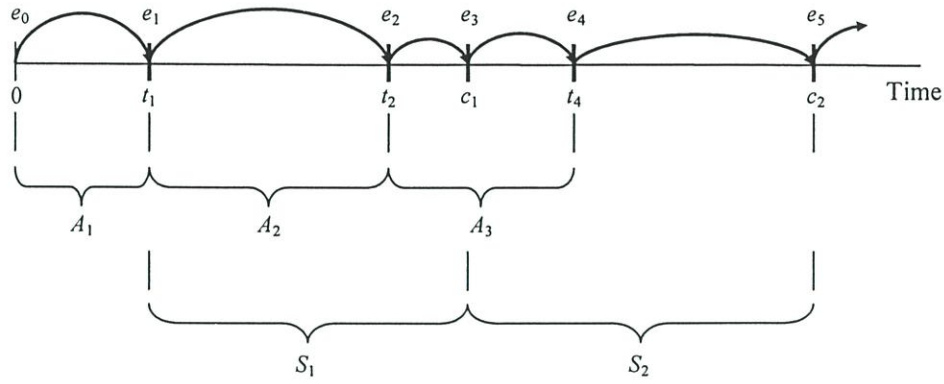
$A_i = t_i - t_{i-1}$ = inter arrival time ระหว่างการร้องขอลำดับที่ $i$ และ $i$ - 1

$S_i$ = เวลาที่เซิร์ฟเวอร์ใช้สำหรับประมวลผลการร้องขอลำดับที่ $i$

$D_i$ = ดีเลย์ในคิวของการร้องขอลำดับที่ $i$

$c_i = t_i + D_i + S_i$ = เวลาที่ใช้ทั้งหมดสำหรับประมวลผลการร้องขอที่ $i$

$e_i$ = จุดของเวลาที่เหตุการณ์ใดๆเกิดขึ้น หรือ จุดเวลาที่มีการเพิ่มค่า Simulation Clock ครั้งที่ $i$



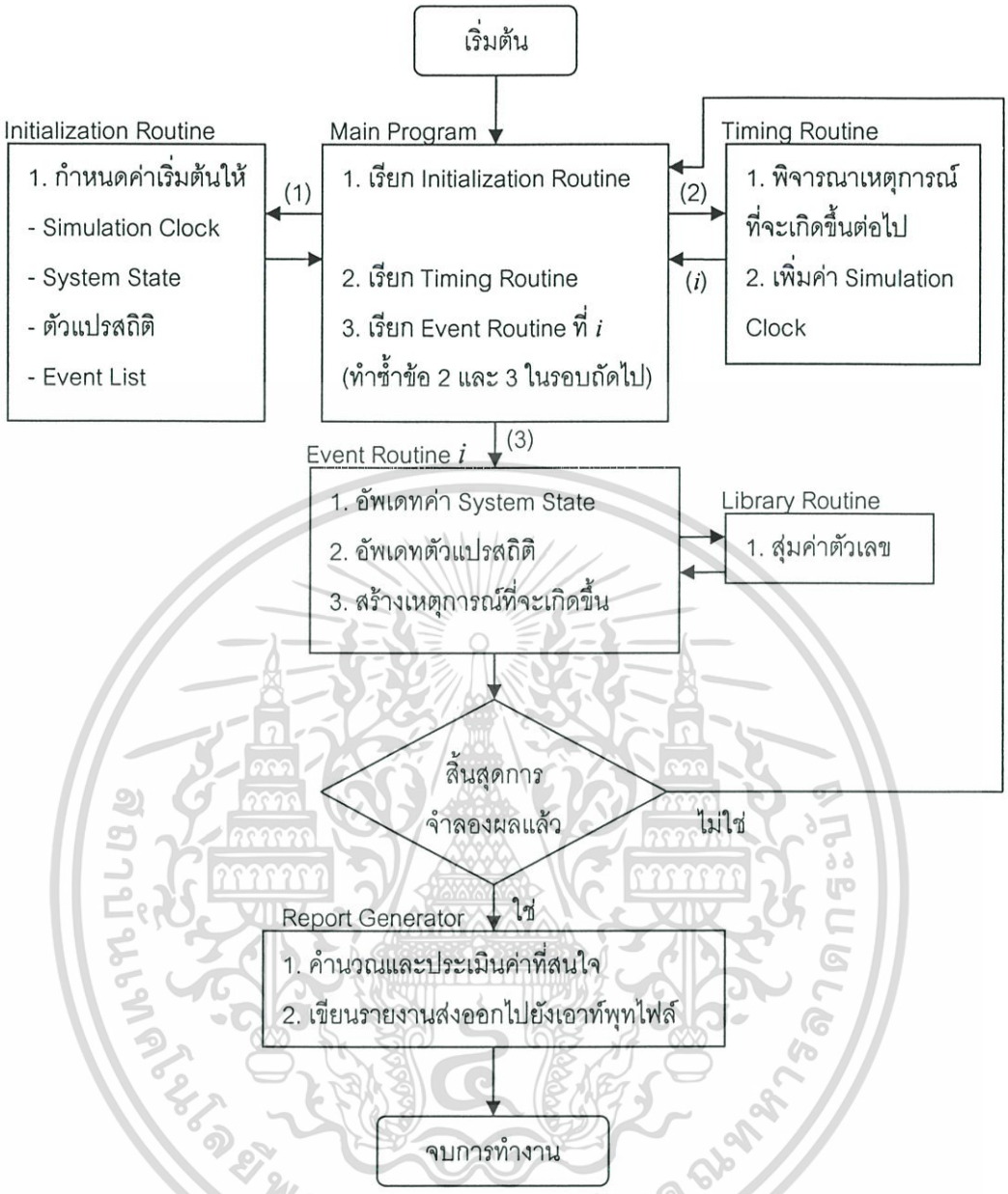

**รูปที่** 5.1 วิธีการเพิ่มค่า Simulation Clock ด้วยวิธี next-event time-advance สำหรับระบบแบบ Single Server Queue (*M/M/1*)

วิธีการเพิ่มค่า Simulation Clock ด้วยวิธี next-event time-advance ที่แสดงตัวอย่างในรูป 5.1 แสดงให้เห็นการทำงานของระบบจำลองที่มีจุดเวลาของการเข้ามาของการเข้ามาของการร้อง $t_i$ และมี inter arrival time $A_i$ ลำดับต่างๆ แต่ละการร้องขอจะถูกประมวลผลโดยเซิร์ฟเวอร์ใช้เวลาเท่ากับ $S_i = c_i - t_i$ ซึ่งจากตัวอย่างด้านบน บางครั้งการเข้ามาของการร้องขอจะพบว่าสถานะของเซิร์ฟเวอร์ยังไม่ว่าง จำเป็นต้องรอเวลาเพื่อให้เซิร์ฟเวอร์ว่างเสียก่อน การร้องขอจึงจะถูกประมวลผล ระยะเวลารอประมวลผลนี้เรียกว่า ดีเลย์ $D_i$ มีค่าเท่ากับ $c_{i-1} - t_i$ ยกตัวอย่างเช่น $D_2 = c_1 - t_2$ สำหรับการเพิ่มค่า Simulation Clock ด้วยวิธี next-event time-advance จะทำทุกครั้งที่เกิดเหตุการณ์ในจุดเวลาใดเวลาหนึ่ง $e_i$ ซึ่งได้แก่ เวลาที่เข้ามาของการร้องขอ $t_i$ และเวลาที่เซิร์ฟเวอร์ประมวลผลการร้องขอนั้นเสร็จสิ้น $c_i$ โดยที่กำหนดค่าเริ่มต้น $e_0 = 0$

การประมวลผลในระบบจำลองนี้จะมีการเพิ่มค่า Simulation Clock ขึ้นไปเรื่อย และจะหยุดการประมวลผลด้วยเงื่อนไขที่เกี่ยวข้องกับตัวแปรในระบบจำลอง ยกตัวอย่างเช่น ให้หยุดการประมวลผลเมื่อ ค่าของ Simulation Clock มากกว่า 1,000 หรือ ประมวลผลการร้องขอครบ 200 ครั้ง เป็นต้น

**องค์ประกอบและการประมวลผลของการจำลองผลแบบ** Discrete Time Event

การจำลองผลแบบ Discrete Time Event ประกอบด้วยตัวแปรมาตรฐานที่ถูกจัดการด้วยวิธีการทางด้านลอจิกร่วมกัน เพื่อง่ายต่อการโปรแกรม การดีบัก และการปรับปรุงแก้ไขในอนาคต ระบบจำลองที่ได้มีการสร้างขึ้นมาด้วย Discrete Time Event Model และมีการเพิ่มค่า Simulation Clock ด้วยการวิธี next-event time-advance จะมีวิธีการประมวลผลและส่วนประกอบดังต่อไปนี้




**รูปที่** 5.2 โฟลวชาร์ตของการประมวลของการจำลองผลด้วย Discrete Time Event Model ที่มีการเพิ่มค่าเวลาด้วยวิธี next-event time-advance

- *System State* เป็นกลุ่มของตัวแปรของระบบที่จำเป็นสำหรับการอธิบายลักษณะของระบบ ณ จุดเวลาหนึ่ง
- *Simulation Clock* เป็นตัวแปรที่บอกเวลาปัจจุบันของระบบจำลอง
- *Event List* เป็นลิสต์ของเหตุการณ์ที่รอเวลาเพื่อเข้าสู่ระบบจำลอง
- *Initialization Routine* เป็นส่วนของโปรแกรมที่มีหน้าที่เซตค่าตัวแปรเริ่มต้นต่างๆ ของระบบจำลอง เช่น เซตค่าเริ่มต้นของ Simulation Clock ให้เป็น 0

- *Event Routine* เป็นส่วนของโปรแกรมที่มีหน้าที่อัพเดทค่า System State เมื่อมีเหตุการณ์ใดๆ เกิดขึ้น
- *Library Routines* เป็นเซตของส่วนของโปรแกรมที่มีหน้าที่ช่วยเหลือให้ระบบจำลองทำงานได้ตามจุดมุ่งหมาย เช่น การสุ่มค่าเลขด้วยการกระจายที่ระบุไว้
- *Report Generator* เป็นส่วนของโปรแกรมที่มีหน้าที่ทำรายงาน โดยเขียนข้อมูลที่เกี่ยวข้องกับ System State ไปยังเอาท์พุทไฟล์ เพื่อวิเคราะห์ประสิทธิภาพของระบบต่อไป
- *Main Program* เป็นส่วนของโปรแกรมที่เรียกใช้ Timing Routine เพื่อหาค่าเหตุการณ์ถัดไป จากนั้น จะส่งการควบคุมไปยัง Event Routine ที่เหมาะสมเพื่ออัพเดทค่า System State ต่อไป นอกจากนี้ Main Program ยังมีหน้าที่ตรวจสอบการสิ้นสุดการจำลองผลและออกรายงานเมื่อพบว่าสิ้นสุดการจำลองผลแล้ว

ความสัมพันธ์ระหว่างส่วนประกอบต่างๆ สำหรับการประมวลผลระบบจำลองด้วย Discrete Time Event Model แสดงไว้ในรูปที่ 5.2 เริ่มต้นด้วยการเรียก Initialization Routine สำหรับกำหนดค่าเริ่มต้นให้แก่ระบบ จากนั้น จะเรียก Timing Routine เพื่อเลือกเหตุการณ์ที่เหมาะสม *i* เข้ามาประมวลผลที่ Event Routine และจะดำเนินการซ้ำไปจนกระทั่งสิ้นสุดการจำลองผลตามเงื่อนไขที่กำหนดไว้ สุดท้าย Report Generator จะถูกเรียกใช้เพื่อสร้างรายงานเพื่อนำไปวิเคราะห์ประสิทธิภาพต่อไป

### 5.1.2 การประยุกต์ใช้ทฤษฎี Queuing และ Discrete Time Event Model สำหรับทดสอบประสิทธิภาพระบบการค้นหาที่ใช้ในวิทยานิพนธ์

ในการประเมินประสิทธิภาพระบบการค้นหาในวิทยานิพนธ์นี้ ได้ประยุกต์ใช้ทฤษฎี Queuing เพื่ออธิบายการบริการการค้นหาของเอเจนต์ผู้ให้บริการในระบบประมวลผลแบบกริด และใช้ Discrete Time Event Model เพื่อประมวลผลการทดสอบประสิทธิภาพของระบบ โดยมีรายละเอียด ดังนี้

การจำลองผลในระบบการค้นหาในวิทยานิพนธ์นี้ มีลักษณะเป็นระบบ Queuing แบบ *M/M/s* กล่าวคือ

- เอเจนต์ผู้ให้บริการการค้นหาในระบบประมวลผลแบบกริดเป็นเซิร์ฟเวอร์ให้บริการ ซึ่งมีจำนวน $N$ ตัว ดังนั้น $s = N$
- กำหนดให้การเข้ามาการร้องขอข้อมูลจากผู้ใช้บริการและเวลาการให้บริการของเซิร์ฟเวอร์ มีการกระจายแบบเอ็กโพเนนเชียล
- คิวในระบบแบ่งออกเป็น 2 ประเภท คือ คิวของระบบที่เป็นศูนย์รวมของเหตุการณ์การร้องขอเริ่มต้นที่เข้ามาของการร้องขอข้อมูลในระบบ และ คิวของเอเจนต์แต่ละตัวที่มี

หน้าที่เก็บเหตุการณ์ที่มีการส่งต่อการร้องขอมาจากคิวของระบบและการร้องของจากเอเจนต์ตัวอื่น คิวทั้งสองมีการจัดการคิวแบบ FIFO

การจำลองผลด้วย Discrete Time Event Model มีการแก้ไขให้สนับสนุนการระบบ Queuing ข้างต้น ดังนี้

- Timing Routine เพิ่มความสามารถในการเลือกเหตุการณ์ที่ $i$ ว่าประมวลผลที่เซิร์ฟเวอร์หรือที่เอเจนต์ตัวใดในระบบจำลอง
- Event Routine เพิ่มความสามารถในการส่งต่อเหตุการณ์ โดยอ้างอิงตามวิธีการค้นหาข้อมูลวิธีต่างๆ ได้แก่ Flooding, Signature และ K-PST รวมทั้งสร้างรายงานไปยังเอาท์พุทไฟล์เพื่อการวิเคราะห์ผลการทดสอบแบบสะสมได้

จำนวนการส่งต่อเหตุการณ์ใน Event Routine ถือเป็นประเด็นพิจารณาที่สำคัญในการจำลองผลสำหรับวิทยานิพนธ์นี้ เนื่องจากว่าเป้าหมายของวิทยานิพนธ์นี้มุ่งไปที่ประสิทธิภาพของการค้นหาในระบบ วัดจากจำนวนการเชื่อมต่อซึ่งเทียบได้กับการส่งต่อเหตุการณ์ใน Event Routine ประสิทธิภาพของระบบการค้นหานี้เกี่ยวข้องกับมาตรวัดประสิทธิภาพของการค้นหาของระบบประมวลแบบกริด ซึ่งจะกล่าวไว้ในหัวข้อถัดไป

## 5.2 มาตรวัดประสิทธิภาพของกระบวนการค้นหา

มีมาตรวัดหลายอย่างที่สามารถนำมาประเมินประสิทธิภาพของระบบการค้นหาข้อมูลนะระบบประมวลผลแบบกริด โดยแต่ละตัวจะพิจารณาความต้องการทรัพยากรของระบบในแง่มุมที่แตกต่างกัน สำหรับวิทยานิพนธ์นี้ การประเมินประสิทธิภาพจะเกี่ยวข้องกับความต้องการพื้นฐานของระบบโดยที่เน้นไปที่ความเร็วในการให้บริการ โดยมีส่วนเกี่ยวข้องกับมาตรวัดดังนี้ คือ ความเร็วการค้นหา และ ประสิทธิภาพของระบบ โดยอ้างอิงจาก [23] อย่างไรก็ตาม เนื่องจากวิทยานิพนธ์นี้ให้ความสำคัญกับความเร็วเป็นหลัก ในการทดสอบจึงให้ความสำคัญของมาตรวัดด้านความเร็วมากที่สุด ส่วนประสิทธิภาพของระบบจะมีความสำคัญรองลงมา

### 5.1.1 ความเร็วการค้นหา (Discovery Speed)

เมื่อมีการร้องขอข้อมูลผู้ใช้บริการ ข้อมูลการร้องขอจะถูกส่งไปให้เอเจนต์ในระบบเพื่อค้นหาข้อมูลที่ต้องการ ในระบบการค้นหาข้อมูลนี้ มาตรวัดประสิทธิภาพที่สำคัญก็คือจำนวนการเชื่อมต่อสำหรับเลือกเส้นทาง (routing connections) เนื่องจากข้อมูลที่ส่งติดต่อสื่อสารมีขนาดเล็ก ทำให้กระบวนการรับส่งข้อมูลทำได้รวดเร็ว จึงไม่ถือว่าระยะเวลาการส่งข้อมูลใน

สายสัญญาณมีความสำคัญมากนัก และในการทำงานของระบบโดยรวมก็สามารถส่งข้อมูลสำหรับค้นหาได้พร้อมกัน ค่าเฉลี่ยของความเร็วในการค้นหา $v$ ระบุไว้ ดังนี้

$$v = \frac{r}{d} \tag{5.1}$$

โดยที่ $r$ เป็นจำนวนของการร้องขอข้อมูลในช่วงเวลาหนึ่ง และ $d$ เป็นจำนวนการเชื่อมต่อในกระบวนการค้นหาข้อมูลของดิสคัฟเวอรีเซิร์ฟเวอร์ มาตรวัดความเร็วการค้นหา $v$ หากมีค่ามากแสดงว่าระบบการค้นหานั้นมีประสิทธิภาพด้านความเร็วดี เนื่องจากการเชื่อมต่อทั้งหมดเพื่อค้นหาข้อมูลมีจำนวนน้อยครั้ง

บางครั้งการเปรียบเทียบโมเดลที่แตกต่างกัน ก็สามารถใช้อัตราการส่งข้อมูลต่อการร้องขอ $h$ (hops per request) แทนการใช้ความเร็วในการค้นหาได้ ถ้าหากการใช้ $h$ สามารถบ่งบอกความแตกต่างระหว่างโมเดลได้ชัดเจนกว่า

$$h = \frac{d}{r} \tag{5.2}$$

จะเห็นได้ว่าอัตราการส่งข้อมูลต่อการร้องขอ $h$ ก็คือส่วนกลับของความเร็วการค้นหาข้อมูล $v$ ดังนั้น ระบบการค้นหาที่ดีจะต้องให้ค่าอัตราการส่งข้อมูลต่อการร้องขอ $h$ ที่มีค่าน้อย

### 5.1.2 ประสิทธิภาพของระบบ (System Efficiency)

นอกเหนือจากจำนวนการเชื่อมต่อเพื่อค้นหาข้อมูลแล้ว จำนวนการประกาศข้อมูลและดูแลรักษาข้อมูลด้วย ในระบบก็สมควรนำมาประกอบการประเมินประสิทธิภาพในระบบโดยรวมของการค้นหาข้อมูลเช่นกัน เนื่องจากกระบวนการดังกล่าวเป็นการเพิ่มภาระการทำงานระบบ กำหนดให้ $c$ คือ จำนวนการเชื่อมต่อสำหรับกระบวนการค้นหาทั้งหมดซึ่งประกอบด้วย $d$ คือ จำนวนการเชื่อมต่อในกระบวนการค้นหา และ $a$ คือ จำนวนการเชื่อมต่อสำหรับกระบวนการประกาศและดูแลระบบ มีความสัมพันธ์ดังนี้

$$c = d + a \tag{5.3}$$

ประสิทธิภาพของระบบสามารถพิจารณาได้จากอัตราส่วนระหว่างการร้องขอข้อมูลทั้งหมด $r$ เทียบกับจำนวนการเชื่อมต่อสำหรับกระบวนการค้นหาทั้งหมด ได้แก่ การเชื่อมต่อในกระบวนการค้นหา และ กระบวนการประกาศและดูแลระบบ $c$ โดยระบุไว้ดังนี้

$$e = \frac{r}{c} \tag{5.4}$$

มาตรวัดประสิทธิภาพของระบบ $e$ หากมีค่ามาก แสดงว่าระบบการค้นหานั้นมีประสิทธิภาพดี เนื่องจากมีค่าใช้จ่ายในเรื่องของการเชื่อมต่อเพื่อค้นหาข้อมูลรวมกับการประกาศข้อมูลมีจำนวนน้อยครั้ง

ในระบบการค้นหาโดยทั่วไปแล้ว มาตรวัดประสิทธิภาพการค้นหาเหล่าทั้งสองจะมีความขัดแย้งกันอยู่ กล่าวคือ ไม่มีมาตรวัดตัวใดที่จะมีค่าสูงทุกตัวในเวลาหรือระบบเดียวกัน ยกตัวอย่าง เช่น ระบบที่มีความเร็วสูง ไม่ได้หมายความว่าระบบจะมีประสิทธิภาพสูงตามไปด้วย ดังนั้น ระบบที่ดีจึงควรมีค่ามาตรวัดประสิทธิภาพสูงที่บ่งชี้การทำงานระบบในหลายๆ ประเด็น เมื่อเปรียบเทียบในสภาพแวดล้อมเดียว สิ่งที่ได้จากมาตรวัดประสิทธิภาพต่อวิทยานิพนธ์นี้ก็คือ การสร้างระบบการค้นหาที่ทำอย่างไรให้ระบบการค้นหามีความเร็วสูง ในขณะที่ประสิทธิภาพ อยู่ในระดับที่น่าพอใจ โดยอาศัยมาตรวัดประสิทธิภาพนี้เป็นมาตรฐานการประเมินประสิทธิภาพของระบบ

## 5.3 การทดสอบความเร็วและประสิทธิภาพของกระบวนการค้นหา

ในส่วนนี้จะเป็นการทดสอบระบบการค้นหาข้อมูลในระบบประมวลผลแบบกริดที่ได้ออกแบบไว้ในวิทยานิพนธ์นี้ โดยถือเอามาตรวัดประสิทธิภาพที่ได้กล่าวไว้ในหัวข้อก่อนหน้านี้เป็นเกณฑ์ในการประเมินประสิทธิภาพ ในที่นี้จะมีการทดสอบระบบด้วยการจำลองผล ตามที่กล่าวไว้ในหัวข้อก่อนหน้านี้ มีการทดสอบโมเดลในสภาพแวดล้อมที่แตกต่างกันหลายลักษณะ เพื่อชี้ให้เห็นถึงปัจจัยที่มีผลต่อการทำงานของระบบการค้นหาในแง่มุมต่างๆ รวมทั้งมีการทดลองเปรียบเทียบกับโมเดลอื่นเพื่อให้เห็นข้อแตกต่างของประสิทธิภาพของระบบการค้นหาอีกด้วย

### 5.3.1 ระบบการค้นหาที่ใช้ในการทดลอง

การดำเนินการทดลอง จะมีเริ่มต้นด้วยการทดสอบระบบการเชื่อมต่อระหว่างเอเจนต์ที่มีลักษณะแบบลูกโซ่ลำดับชั้น (Hierarchical Chain) เป็นโครงสร้างลำดับชั้นพื้นฐานของระบบจัดการข้อมูลของกริด วิธีการค้นหาแบบต่างๆ ในวิทยานิพนธ์นี้จะทดลองเปรียบเทียบกับการค้นหาแบบ Flooding ที่ใช้ในโครงสร้างลูกโซ่ลำดับชั้นที่ใช้อยู่ในงานวิจัย [38, 39] การเพิ่มประสิทธิภาพของระบบการค้นหาจะทำทีละขั้นโดยเริ่มต้นจาก Flooding เรื่อยไปจนกระทั่งถึง K-PST โดยแสดงชื่อที่ใช้อ้างอิงและรายละเอียดไว้ในตารางที่ 5.1

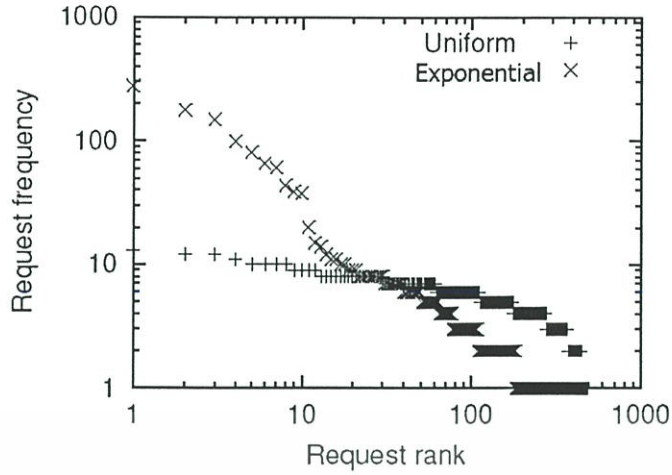
**ตารางที่** 5.1 วิธีการเพิ่มประสิทธิภาพระบบการค้นหาที่ใช้ในการทดลอง

| Name | Use | | | |
|---|---|---|---|---|
| | cache | signature | K-PST | select 1 target node |
| Flooding (no cache) | - | - | - | - |
| Flooding (with cache) | √ | - | - | - |
| Signature (no cache) | - | √ | - | - |
| Signature (with cache) | √ | √ | - | - |
| K-PST (all) | √ | √ | √ | - |
| K-PST (one) | √ | √ | √ | √ |

การทดสอบระบบการค้นหาจะมีทั้งการค้นหาแบบครบถ้วน กล่าวคือ ค้นหาไปยังทุกโหนดที่มีความเป็นไปได้ว่าจะมีข้อมูลที่ต้องการอยู่ และ การค้นหาแบบบางส่วนที่เลือกเดินทางค้นหาไปยังเส้นทางส่วนหนึ่งเพื่อลดจำนวนการเชื่อมต่อ การทดลองที่ดำเนินการขึ้นจะเน้นประสิทธิภาพการค้นหาแบบครบถ้วนที่นำเสนอขึ้นในงานวิจัย คือ Signature และ K-PST(all) เปรียบเทียบกับ Flooding เป็นหลัก โดยจะชี้ให้เห็นประโยชน์ที่ได้จากการใช้แคช Signature และ K-PST นอกจากนี้ ในการทดลองจะเปรียบเทียบข้อดีข้อเสียของความเร็ว และประสิทธิภาพของระบบโดยรวม ของระบบที่ออกแบบขึ้น จากนั้น จะชี้ให้เห็นถึงผลประโยชน์ที่ได้รับจากกลไกการค้นหาที่จำกัดจำนวนเส้นทางการค้นหาข้อมูลเพียง 1 เส้นทาง หรือ K-PST(one) ซึ่งถือเป็นกรณีที่ดีที่สุดที่สามารถทำได้ สุดท้ายจะเป็นการทดลองเปรียบเทียบกับงานวิจัยอื่นๆ ที่เกี่ยวข้องกับการค้นหาข้อมูลอย่างรวดเร็วในกริด

### 5.3.2 ความถี่การค้นหาข้อมูล

ในระบบจำลองที่สร้างขึ้น มีจำลองความถี่ของการร้องขอข้อมูลในระบบด้วยการกระจาย 2 ลักษณะ คือ แบบยูนิฟอร์ม (Uniform) และแบบเอ็กโพเนนเชียล (Exponential) โดยการกระจายลักษณะดังกล่าวมีความถี่การเรียกใช้ข้อมูลเรียงตามลำดับดังรูปที่ 5.3 ซึ่งจะเห็นได้ว่าในการกระจายแบบยูนิฟอร์มความถี่ข้อมูลที่ถูกเรียกอันดับต่างๆ จะมีค่าใกล้เคียงกันมาก การทดสอบด้วยการกระจายแบบนี้บ่งบอกถึงลักษณะของระบบที่มีการเรียกข้อมูลเดิมซ้ำไม่บ่อยครั้งนัก มักจะนำมาวิเคราะห์ประสิทธิภาพของวิธีการค้นหาข้อมูลแบบต่างๆ ซึ่งแตกต่างจากการกระจายแบบเอ็กโพเนนเชียล ที่ความถี่ข้อมูลที่ถูกเรียกอันดับต่างๆ จะมีค่าแตกต่างกันมาก การทดสอบด้วยการกระจายแบบนี้บ่งบอกถึงลักษณะของระบบที่มีการเรียกข้อมูลเดิมซ้ำบ่อยครั้ง เช่น พฤติกรรมการร้องขอเว็บเพจใน HTTP โปรโตคอล เป็นต้น



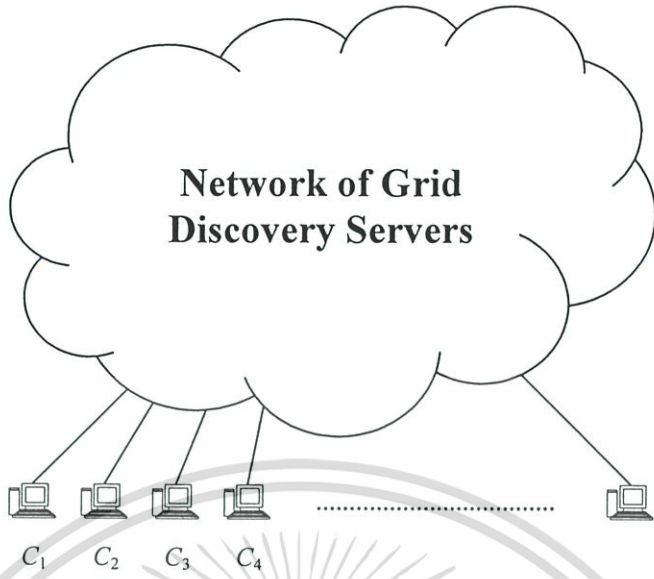

**รูปที่** 5.3 การกระจายของความถี่การร้องขอข้อมูลแบบยูนิฟอร์มและแบบเอ็กโพเนนเชียล

จุดประสงค์ของการจำลองด้วยการกระจายทั้งสอง เพื่อแสดงถึงประสิทธิภาพการค้นหาด้วยวิธีการที่อาศัย Signature และ K-PST โดยมีการจำลองการค้นหาที่มีความถี่การร้องขอข้อมูลแบบยูนิฟอร์ม และแสดงให้เห็นถึงประโยชน์ที่ได้จากแคชในระบบที่มีการเรียกข้อมูลซ้ำมากที่เกิดขึ้นในระบบที่มีความถี่การร้องขอข้อมูลแบบเอ็กโพเนนเชียลเปรียบเทียบกับระบบที่มีการร้องขอข้อมูลซ้ำน้อยที่เกิดขึ้นในระบบที่มีความถี่การร้องขอข้อมูลแบบยูนิฟอร์ม

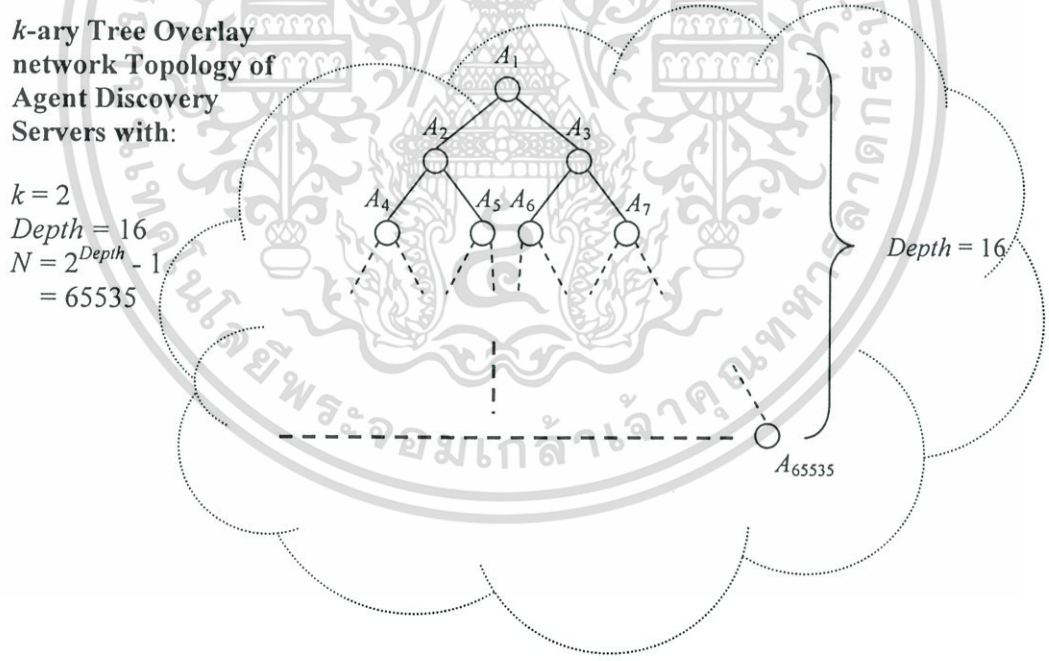
### 5.3.3 ระบบทดสอบพื้นฐาน

ในงานวิจัยชิ้นนี้ ได้มีการจำลองผลการทดสอบประสิทธิภาพของกระบวนการค้นหาภายในระบบการประมวลผลแบบกริด เรียกว่า "ระบบทดสอบพื้นฐาน" เพื่อใช้ระบบทดสอบกลางเพื่อเป็นการอ้างอิง ทั้งนี้การทดลองในแต่ละหัวข้อ จะใช้ระบบทดสอบพื้นฐานนี้ทดสอบปัจจัยด้านประสิทธิภาพต่างๆ โดยการเปลี่ยนแปลงค่าตัวแปรเพื่อทดสอบประสิทธิภาพ โดยจะระบุไว้ในหัวข้อการทดสอบนั้นๆ อีกครั้งหนึ่ง

ระบบทดสอบพื้นฐานที่ใช้ในวิทยานิพนธ์นี้ มีลักษณะตามรูปที่ 5.4 และ 5.5 โดยระบบจำลองประกอบด้วยผู้ใช้บริการ มีการร้องขอข้อมูล ไปยังกลุ่มของเอเจนต์ที่ทำหน้าที่เป็นดิสคัฟเวอรีเซิร์ฟเวอร์ที่กระจายตัวอยู่ในระบบเครือข่าย สำหรับการจำลองผลที่สร้างขึ้น จะกำหนดให้ผู้ใช้บริการ $C_N$ จะเริ่มต้นการร้องขอข้อมูลไปยังเอเจนต์ $A_N$ ซึ่งทำหน้าที่ดูแลผู้ใช้บริการ $C_N$ นั้นอยู่ ยกตัวอย่างเช่น ผู้ใช้บริการ $C_1$ จะเริ่มต้นการร้องขอไปที่ $A_1$ จากนั้น เอเจนต์จะส่งการค้นหาข้อมูลไปยังเอเจนต์ตัวอื่น ขึ้นกับวิธีการค้นหาที่กำหนด เมื่อเอเจนต์ $A_N$ ได้รับคำตอบของการร้องขอแล้ว เอเจนต์จะส่งผลลัพธ์กลับมายังผู้ใช้บริการที่ร้องขอข้อมูลเป็นลำดับสุดท้าย อนึ่ง ในระบบการทดสอบกำหนดให้ เอเจนต์หนึ่งตัวดูแลผู้ใช้บริการหนึ่งราย ดังนั้น ในระบบจำลองพื้นฐานจำนวนผู้ใช้บริการจึงเท่ากับจำนวนเอเจนต์ที่ทำหน้าที่เป็นดิสคัฟเวอรีเซิร์ฟเวอร์




**รูปที่** 5.4 ความสัมพันธ์ระหว่างไคลเอนต์กับกลุ่มของเอเจนต์ซึ่งทำหน้าที่เป็นกริดดิสคัฟเวอรีเซิร์ฟเวอร์ในระบบเครือข่าย




**รูปที่** 5.5 การเชื่อมต่อของเอเจนต์ที่ทำหน้าที่เป็นกริดดิสคัฟเวอรีเซิร์ฟเวอร์โครงสร้างลำดับชั้นที่ใช้สำหรับการจำลองผล ซึ่งเป็น *k*-ary Tree โดยที่ $k = 2$, $Depth = 16$ คิดเป็นจำนวนโหนดทั้งหมด $N = 2^{Depth} - 1 = 65535$ โหนด

การเชื่อมต่อของเอเจนต์ในเชิงลอจิก มีการเชื่อมต่อแบบ *k*-ary Tree ที่มีค่า $k = 2$ (ไบนารีทรี) และมีจำนวน *Depth* = 16 หรือคิดเป็นจำนวนโหนดทั้งหมด $N = 65,535$ โหนด อนึ่ง โครงสร้างเครือข่ายเชิงลอจิกนี้จะมีวิธีการสร้างจากระบบเครือข่ายจริง โดยสามารถดำเนินการด้วยวิธีการ อย่างเช่น การใช้โปรโตคอลบูตสแตรพเพื่อค้นหาดิสคัฟเวอรีเซิร์ฟเวอร์ที่ใกล้ที่สุดที่เป็นสมาชิกของกริด เพื่อสมัครเข้าร่วมเป็นดิสคัฟเวอรีเซิร์ฟเวอร์ของระบบ

**ลักษณะทั่วไปของระบบทดสอบพื้นฐานที่ใช้สำหรับการทดลอง**

1. ลักษณะทั่วไปของระบบจัดการข้อมูลของกริดด้วยเอเจนต์ที่เชื่อมต่อแบบลำดับชั้น
    - โครงสร้างลำดับชั้นของเอเจนต์มีจำนวนโหนด $N$ ทั้งสิ้น 65,535 โหนด
    - การเชื่อมต่อของเอเจนต์ในโครงสร้างลำดับชั้นแบบ *k*-ary Tree โดยที่ $k = 2$ หรือ ไบนารีทรี
    - แต่ละโหนดมีการดูแลข้อมูล 10 ประเภทที่แตกต่างกัน
    - ข้อมูลในระบบมี 100 ประเภทที่แตกต่างกันและกระจายตัวกันแบบยูนิฟอร์ม
    - แคช ใช้รีเพลสเมนต์อัลกอริธึมแบบ LFU เก็บข้อมูลที่แตกต่างกันได้ 10 ประเภท
    - มีการอัพเดทแคชทุกๆ 1,000 การร้องขอข้อมูล
    - สำหรับวิธีการที่ต้องมีการประกาศข้อมูล ได้แก่ Signature, K-PST(all) และ K-PST(one) จะทำทุกๆ 1,000 การร้องขอข้อมูล
2. ลักษณะของการทดสอบการร้องขอข้อมูลด้วยระบบจำลอง
    - ทำการทดลองด้วยการจำลองโดยอาศัยทฤษฎี Queuing แบบ *M/M/s* ที่มีค่าเฉลี่ยของการเข้ามาของการร้องขอเท่ากับ 1 วินาที และค่าเฉลี่ยของการให้บริการเท่ากับ 1 มิลลิวินาที
    - อาศัย Discrete Time Event Model สำหรับการประมวลผลระบบจำลองด้วยคอมพิวเตอร์
    - วิธีการค้นหาแบ่งเป็น 4 แบบคือ Flooding, Signature, K-PST(all) และ K-PST(one)
    - การค้นหาแบบ Flooding จะกำหนดค่าเริ่มต้นของ TTL ไว้เท่ากับจำนวนลำดับชั้นของโครงสร้างลำดับชั้น
    - ทดลองการร้องขอข้อมูลจำนวน 10,000 ครั้ง
    - ความถี่การร้องขอข้อมูลมีการกระจายแบบยูนิฟอร์ม
    - การทดลองจะมีการประเมินค่าความเร็วการค้นหา $v$ และ ประสิทธิภาพ $e$ ของระบบ และเก็บผลการทดสอบเป็นช่วงเวลาทุกๆ 100 การร้องขอข้อมูล

## 5.4 ผลการทดลองและบทวิเคราะห์

ในส่วนนี้ผลการทดลองและวิเคราะห์ผลการทดลอง ที่ได้จากการจำลองผลกระบวนการค้นหาข้อมูลภายในระบบประมวลผลแบบกริด โดยทดลองอ้างอิงระบบทดสอบพื้นฐานที่กล่าวไว้ก่อนหน้านี้ โดยการทดลองจะมีการปรับเปลี่ยนค่าปัจจัยการทดลองตามแต่จุดประสงค์ของการทดสอบแต่ละหัวข้อ โดยมีการระบุการเปลี่ยนแปลงค่าปัจจัยการทดสอบไว้ในหัวข้อนั้นๆ ด้วย

สำหรับการทดลอง มีการทดสอบประสิทธิภาพระบบการค้นหาข้อมูลในแง่มุมต่างๆ ดังนี้

- การใช้งาน Signature และแคช
- การทดสอบการใช้งาน Knowledge และการจำกัดจำนวนการค้นหา
- การทดสอบปัจจัยของความถี่การประกาศข้อมูล
- การทดลองปัจจัยของจำนวนโหนดลูกในลำดับชั้น

### 5.4.1 การใช้งาน Signature และแคช

การทดลองนี้เป็นการลองเพิ่ม Signature และแคชเข้าไปใช้งานในระบบการค้นหาภายในโครงสร้างลำดับชั้น เพื่อวิเคราะห์หาประโยชน์ที่ได้รับจาก Signature และแคช ว่าทำให้ระบบการค้นหามีความเร็วการค้นหาที่ดีขึ้นมากน้อยเพียงใด เมื่อเปรียบเทียบกับระบบที่ใช้การ Flooding ใน Hierarchical Chain ที่มีการร้องขอข้อมูลซึ่งมีการกระจายแบบยูนิฟอร์ม และ เอ็กโพเนน

<u>สมมติฐานของการทดลอง</u>

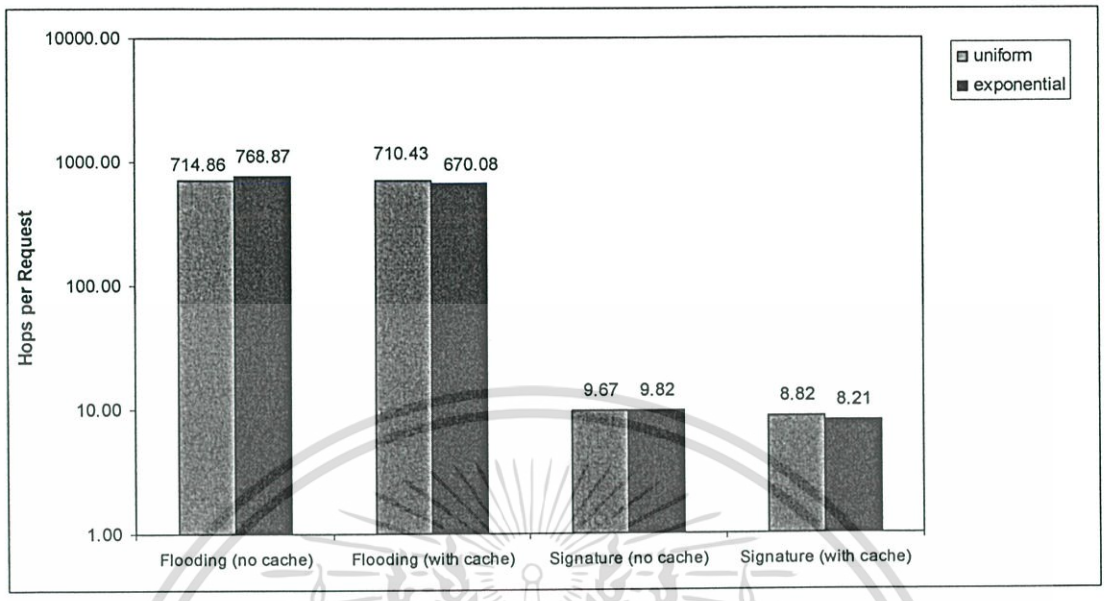
1. ถ้ามีการใช้แคชเข้ามาช่วยระบบการค้นหาจะทำให้ระบบการค้นหามีความเร็วสูงขึ้นหากผู้ใช้เรียกข้อมูลเดิมซ้ำหลายครั้ง
2. การค้นหาข้อมูลด้วย Signature น่าจะช่วยลดจำนวนการเชื่อมต่อเพื่อการค้นหาข้อมูลได้มากเมื่อเปรียบเทียบกับการใช้วิธี Flooding

<u>ปัจจัยการทดลอง</u>

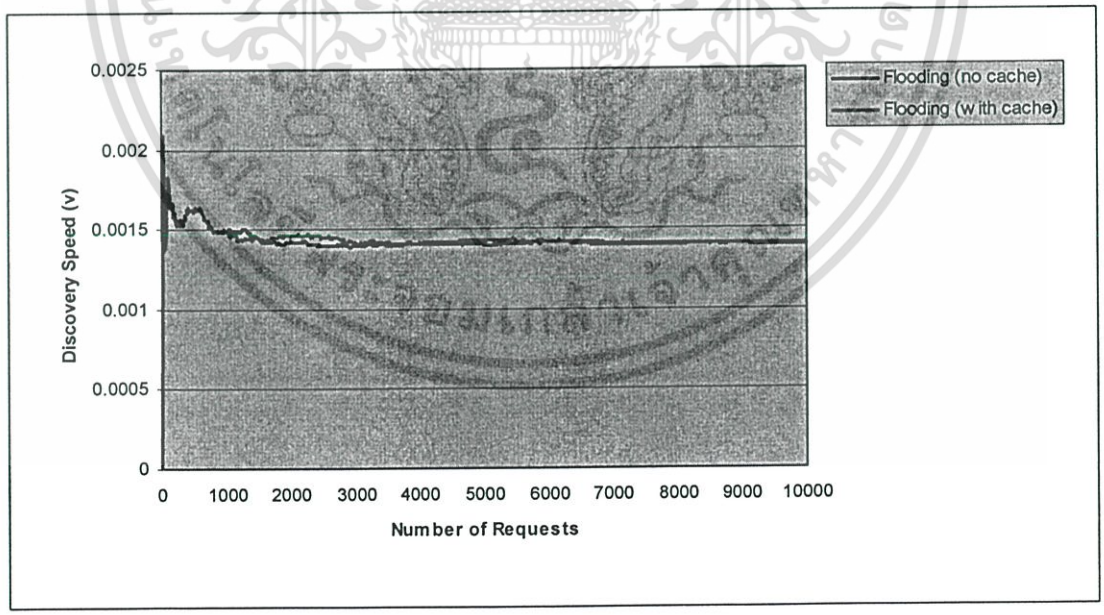
ในการทดลองนี้มีการกำหนดปัจจัยการทดลอง เพิ่มเติมจากระบบทดสอบพื้นฐาน ดังนี้

1. ใช้วิธีการค้นหาเปรียบเทียบกัน 2 วิธี คือ Flooding และ Signature
2. มีการใช้ และ ไม่ใช้แคชในระบบการค้นหาช่วยวิธีการในข้อ 1
3. ความถี่การร้องขอข้อมูลมีการกระจาย 2 ลักษณะ คือ ยูนิฟอร์มและเอ็กโพเนนเชียล

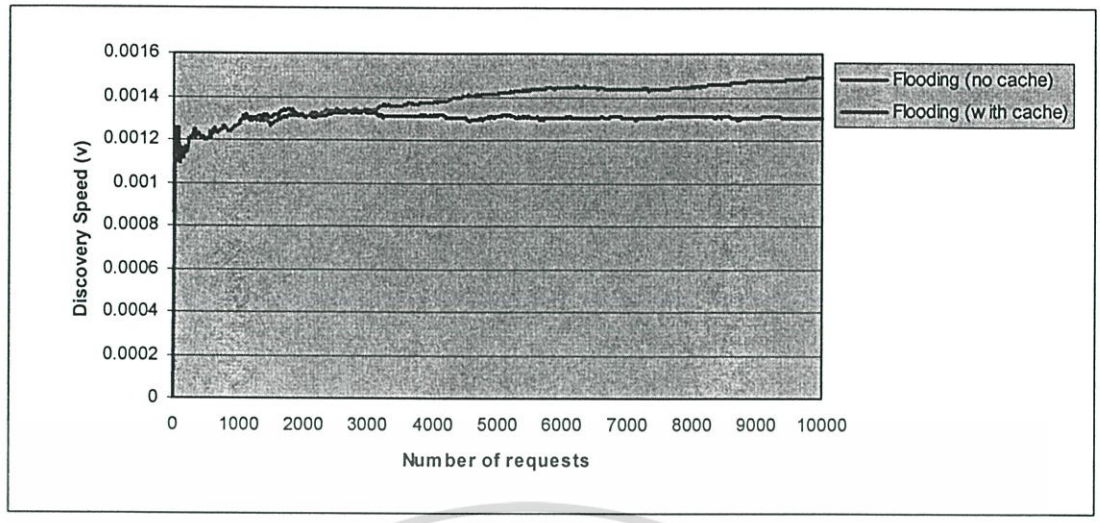
ผลการทดลองและบทวิเคราะห์



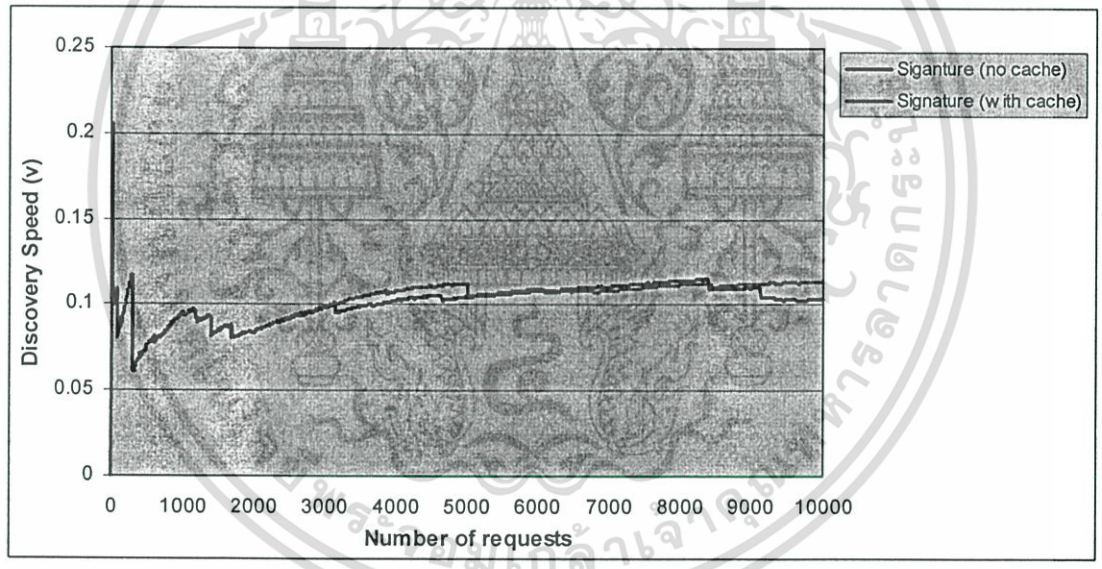

**รูปที่** 5.6 แผนภาพแสดงจำนวนการเชื่อมต่อเฉลี่ยสำหรับการค้นหาข้อมูลในโครงสร้างลำดับชั้น มีการปรับปรุงประสิทธิภาพด้วยการใช้ Signature ที่ใช้และไม่ใช้แคช รวมทั้งเปรียบเทียบการร้องขอข้อมูลที่มีการกระจายแบบยูนิฟอร์มและแบบเอ็กโพเนนเชียล



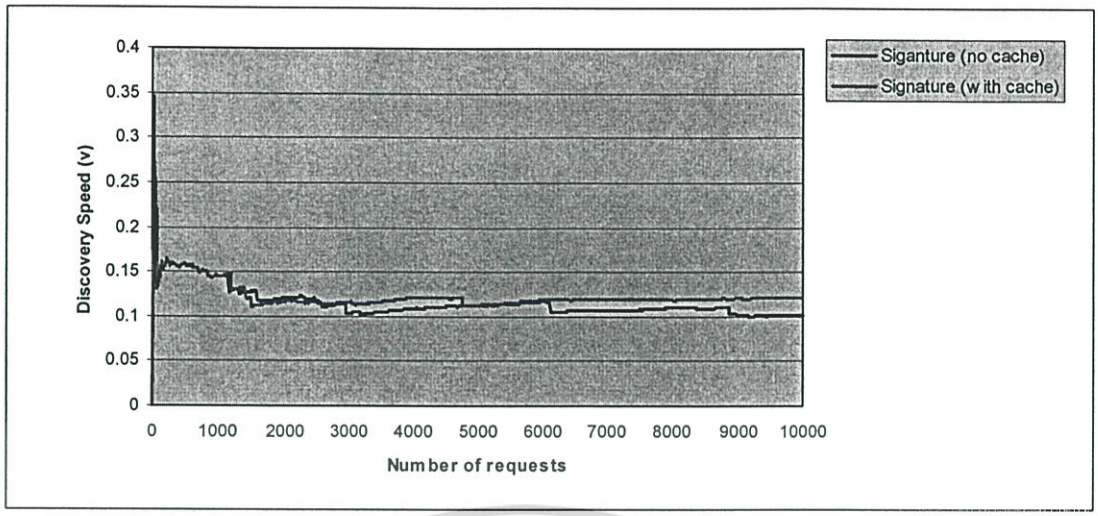

**รูปที่** 5.7 กราฟแสดงความเร็วการค้นหาของวิธีการ Flooding เปรียบเทียบการใช้และไม่ใช้แคช โดยระบบมีความถี่การค้นหาข้อมูลแบบยูนิฟอร์ม




**รูปที่** 5.8 กราฟแสดงความเร็วการค้นหาของวิธีการ Flooding เปรียบเทียบการใช้และไม่ใช้แคช โดยระบบมีความถี่การค้นหาข้อมูลแบบเอ็กโพเนนเชียล




**รูปที่** 5.9 กราฟแสดงความเร็วการค้นหาของวิธีการ Signature เปรียบเทียบการใช้และไม่ใช้แคช โดยระบบมีความถี่การค้นหาข้อมูลแบบยูนิฟอร์ม




**รูปที่** 5.10 กราฟแสดงความเร็วการค้นหาของวิธีการ Signature เปรียบเทียบการใช้และไม่ใช้แคช โดยระบบมีความถี่การค้นหาข้อมูลแบบเอ็กโพเนนเชียล

เมื่อพิจารณารูปที่ 5.6 ซึ่งเป็นกราฟที่จำนวนการเชื่อมต่อเฉลี่ยสำหรับการค้นหาด้วยกระบวนการที่แตกต่างกัน จะเห็นได้ว่า การใช้ Signature สามารถลดจำนวนการเชื่อมต่อได้มาก พิจารณาจากความถี่การร้องขอข้อมูลที่มีการกระจายแบบยูนิฟอร์ม จะเห็นได้ว่า Flooding มีจำนวนการเชื่อมต่อเฉลี่ย 714.86 ต่อการร้องขอข้อมูล 1 ครั้ง การใช้ Signature จะช่วยลดจำนวนการเชื่อมต่อเฉลี่ยลงเหลือเพียง 9.67 ต่อการร้องขอข้อมูล 1 ครั้ง

เมื่อพิจารณาถึงการใช้แคช จะเห็นได้ว่า ระบบที่ได้ประโยชน์จากแคชคือระบบที่มีการกระจายของการร้องขอแบบเอ็กโพเนนเชียลเท่านั้น สังเกตได้จากรูปที่ 5.6 ในวิธีการ Flooding หรือ Signature หากมีการใช้แคชเข้ามาในระบบจะทำให้จำนวนการเชื่อมต่อเฉลี่ยสำหรับการค้นหาข้อมูลลดลงเล็กน้อย เมื่อมีการร้องขอที่มีความถี่กระจายแบบเอ็กโพเนนเชียล ในขณะที่ระบบที่ไม่มีการใช้แคช จำนวนการเชื่อมต่อเฉลี่ยสำหรับการค้นหาข้อมูลจะมีค่าใกล้เคียงกันสำหรับการร้องขอที่มีความถี่แบบยูนิฟอร์มและเอ็กโพเนนเชียล

พิจารณารูปที่ 5.7-5.10 ซึ่งเป็นกราฟที่แสดงความเร็วการค้นหาด้วยกระบวนการที่แตกต่างกัน โดยมุ่งประเด็นไปที่ประโยชน์ของการใช้แคช พบว่า

1. เมื่อมีการร้องขอข้อมูลแบบยูนิฟอร์ม ค่าความเร็วการค้นหา $v$ เปรียบเทียบการใช้แคชและไม่ใช้แคชของแต่ละวิธีการจะมีค่าไม่แตกต่างจากวิธีพื้นฐานมากนัก ซึ่งมีผลมากจากมีการร้องขอข้อมูลซ้ำเป็นจำนวนน้อย
2. เมื่อมีการร้องขอข้อมูลแบบเอ็กโพเนนเชียล ค่าความเร็วการค้นหา $v$ เปรียบเทียบการใช้แคชและไม่ใช้แคชของแต่ละวิธีการ จะให้ค่าที่ดีกว่าเมื่อมีการเพิ่มแคชเข้าไปเป็นส่วนหนึ่งของระบบการค้นหา โดยเฉพาะอย่างยิ่งวิธีการ Flooding จะเห็นความ

แตกต่างค่อนข้างชัดเจน ในขณะที่วิธีการ Signature จะไม่เห็นความแตกต่างมากนัก ซึ่งมีผลมากจากมีการร้องขอข้อมูลซ้ำเป็นจำนวนมาก

### 5.4.2 การทดสอบการใช้งาน Knowledge และการจำกัดจำนวนการค้นหา

ในการทดลองนี้เกี่ยวข้องกับการประยุกต์ใช้ K-PST ที่เก็บความรู้มาเป็นแกนหลักในการค้นหา และเพิ่มวิธีการที่จำกัดจำนวนการค้นหาเข้าไปเพื่อเพิ่มความเร็วในการค้นหา โดยเปรียบเทียบกับวิธีการ Flooding และการใช้ Signature ร่วมกับ Cache

<u>สมมติฐานของการทดลอง</u>

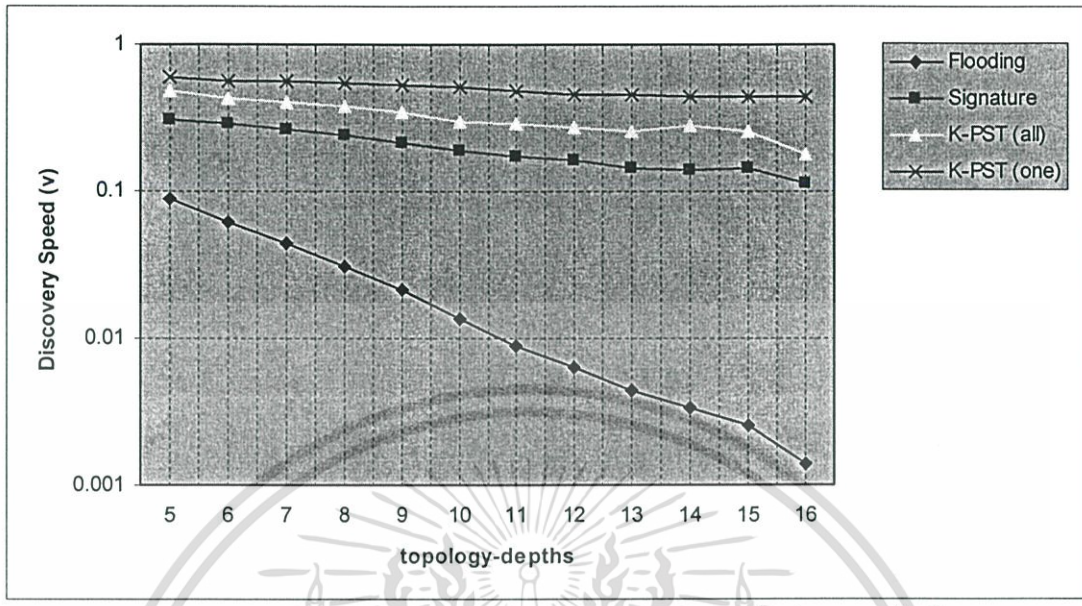
1. การค้นหาที่อาศัย Knowledge และการจำกัดจำนวนการค้นหา ได้แก่ K-PST(all) และ K-PST(one) น่าจะสามารถลดจำนวนการเชื่อมต่อเพื่อการค้นหาข้อมูลได้จำนวนหนึ่ง และดีกว่าวิธีการ Flooding และ Signature เนื่องจากวิธีการดังกล่าวสามารถพิจารณาเส้นทาง โดยสร้างการเชื่อมต่อไปยังโหนดที่มีความเป็นไปได้ว่าน่ามีข้อมูลที่ต้องการอยู่เท่านั้น
2. การค้นหาด้วย K-PST มีความจำเป็นต้องประกาศข้อมูลเป็นจำนวนมาก ดังนั้น เมื่อมีระบบมีจำนวนโหนดมากขึ้น มีความเป็นไปได้ว่าอาจจะทำให้ระบบมีค่าของประสิทธิภาพ $e$ ลดลง จนอาจจะน้อยกว่า วิธีการ Flooding และ Signature หรือไม่

<u>ปัจจัยการทดลอง</u>

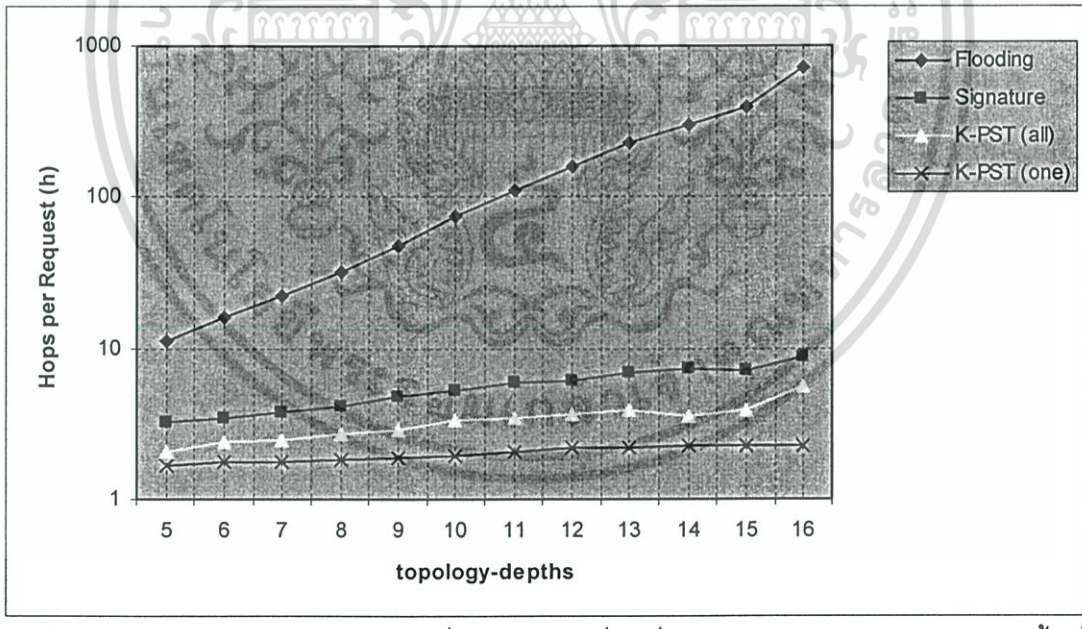
ในการทดลองนี้มีการกำหนดปัจจัยการทดลอง เพิ่มเติมจากระบบทดสอบพื้นฐาน ดังนี้

1. ใช้วิธีการค้นหาเปรียบเทียบกัน 4 วิธี คือ Flooding, Signature, K-PST(all) และ K-PST(one)
2. ทดสอบกับโครงสร้างลำดับชั้นพื้นฐานแบบไบนารีทรี ที่มีจำนวนลำดับชั้นตั้งแต่ $D$ = {5, 6, 7,...16} หรือ คิดเป็นจำนวนโหนด $N$ = {31, 63, 127, ..., 65,535} โหนดตามลำดับ

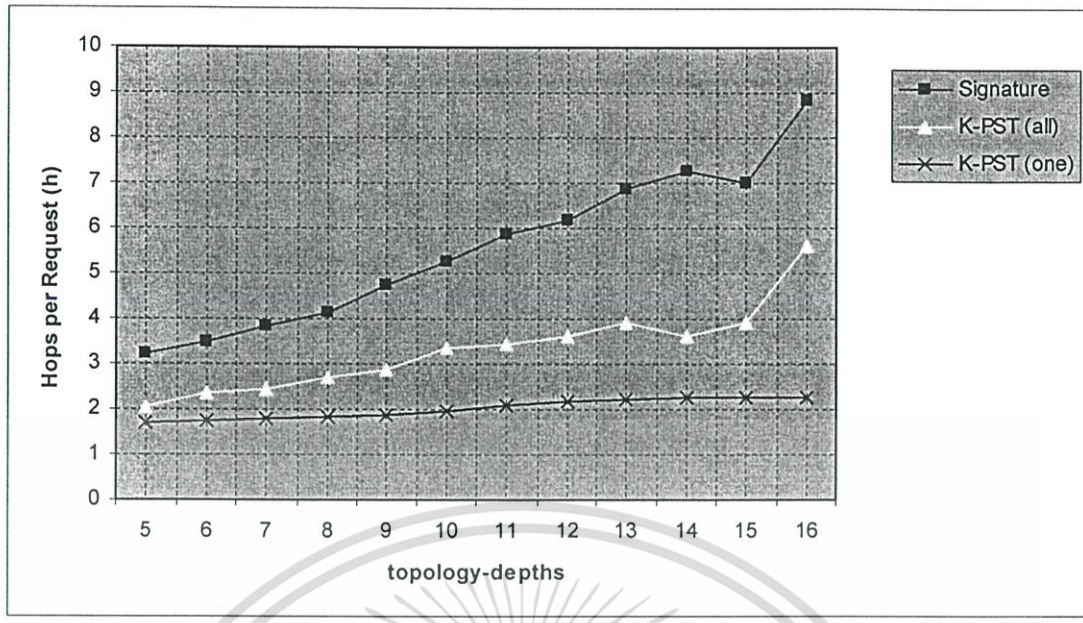
ผลการทดลองและบทวิเคราะห์



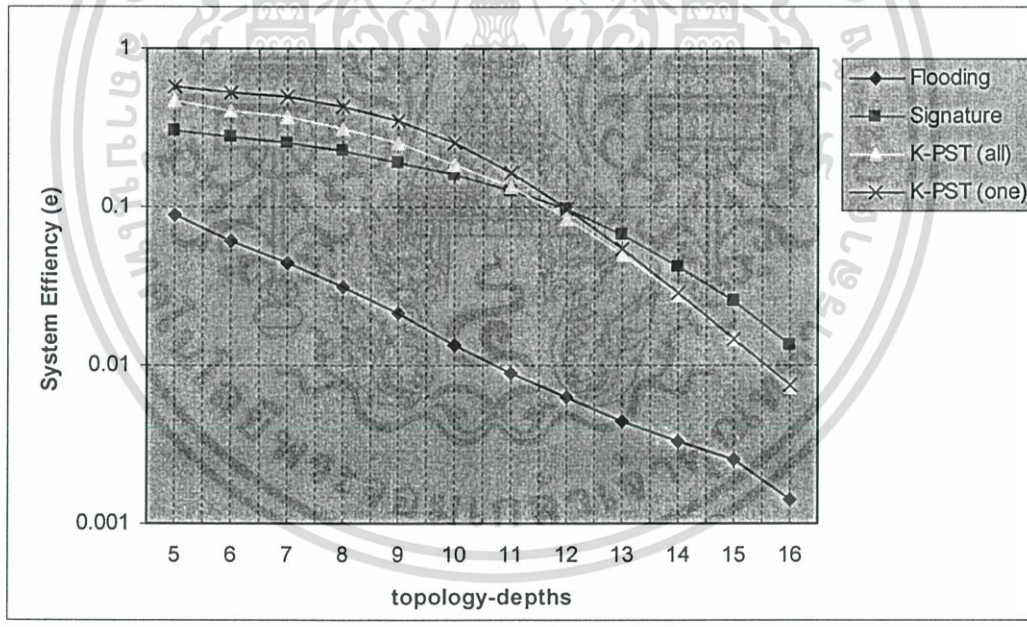

**รูปที่** 5.11 กราฟแสดงค่าความเร็วการค้นหาของโครงสร้างลำดับชั้นที่มีลำดับชั้นตั้งแต่ 5 ถึง 16 ด้วยวิธีการค้นหาแบบต่างๆ




**รูปที่** 5.12 กราฟแสดงค่าจำนวนการเชื่อมต่อโดยเฉลี่ย เพื่อค้นหาข้อมูลในโครงสร้างลำดับชั้นที่มีจำนวนลำดับชั้นตั้งแต่ 5 ถึง 16 ด้วยวิธีการค้นหาแบบต่างๆ




**รูปที่** 5.13 กราฟแสดงค่าจำนวนการเชื่อมต่อโดยเฉลี่ย เพื่อค้นหาข้อมูลในโครงสร้างลำดับชั้นที่มีจำนวนลำดับชั้นตั้งแต่ 5 ถึง 16 ด้วยวิธีการค้นหาแบบต่างๆ




**รูปที่** 5.14 กราฟแสดงค่าประสิทธิภาพของระบบ *e* เพื่อค้นหาข้อมูลในโครงสร้างลำดับชั้นที่มีจำนวนลำดับชั้นตั้งแต่ 5 ถึง 16 ด้วยวิธีการค้นหาแบบต่างๆ

จากรูปที่ 5.11 จะเห็นได้ว่า การใช้งาน Knowledge ด้วย K-PST(all) และ K-PST(one) มีความเร็ว $\nu$ สูงกว่าการใช้งาน Signature และระบบที่มีโครงสร้าง Flooding สังเกตได้จากค่าความเร็วการค้นหาที่ออกมา ระบบที่มีการใช้ K-PST จะให้ค่าที่ดีกว่าในทุกกรณี โดยเฉพาะอย่าง

ยิ่ง เมื่อระบบมีขนาดใหญ่ขึ้นความแตกต่างด้านความเร็วของวิธี K-PST และ Signature ที่มีต่อวิธี Flooding ก็จะเห็นได้ชัดเจนยิ่งขึ้น

เมื่อพิจารณาโดยนำเอาจำนวนการส่งเชื่อมต่อเพื่อการค้นหามาพิจารณา จะพบความแตกต่างระหว่างวิธี Flooding กับระบบที่มีวิธีการค้นหาแบบอื่นๆ ได้อย่างชัดเจน สังเกตจากรูปที่ 5.11 และ 5.12 วิธี Flooding จะมีจำนวนการเชื่อมต่อเพิ่มมากขึ้นเรื่อยๆ ตามจำนวนลำดับชั้นหรือจำนวนเอเจนต์ในระบบ เมื่อจำนวนลำดับชั้นมีค่าเท่ากับ 16 จำนวนการเชื่อมต่อเพื่อค้นหาข้อมูลโดยเฉลี่ยจะมีค่าสูงถึง 710 ครั้ง ดังแสดงในรูปที่ 5.12 ในขณะที่การใช้ K-PST และ Signature จะมีการเชื่อมต่อโดยเฉลี่ยเพียงไม่เกิน 10 ครั้งเท่านั้น โดยที่การใช้ K-PST จะมีจำนวนการเชื่อมต่อโดยเฉลี่ยต่ำกว่าการใช้ Signature เล็กน้อย

เมื่อพิจารณาเฉพาะการใช้ K-PST เปรียบเทียบกับการใช้ Signature ที่แสดงไว้ในรูปที่ 5.13 จะเห็นได้ว่า การใช้ K-PST ค้นหาข้อมูลทั้งแบบ K-PST(all) และแบบ K-PST(one) มีจำนวนการเชื่อมต่อน้อยกว่าการใช้ Signature ทุกกรณี โดยจะมีความแตกต่างมากขึ้นเรื่อยๆ เมื่อโครงสร้างมีจำนวนลำดับชั้นหรือจำนวนเอเจนต์มากขึ้น อย่างเช่น ในโครงสร้างที่มีจำนวนลำดับชั้นเท่ากับ 16 จำนวนการเชื่อมต่อเพื่อค้นหาข้อมูลโดยเฉลี่ยของการใช้งาน Signature การค้นหาที่ใช้ K-PST(all) และ K-PST(one) มีค่าเท่ากับ 8.81, 5.60 และ 2.23 ตามลำดับ จะเห็นว่าการใช้ Signature ใช้การเชื่อมต่อโดยเฉลี่ยมากกว่าเกือบเท่าตัว

เมื่อเปรียบเทียบการค้นหาที่ใช้ K-PST ด้วยกันเองแล้ว เมื่อจำนวนลำดับชั้นของเอเจนต์น้อย จำนวนการเชื่อมต่อจะมีค่าไม่แตกต่างกันมากนัก แต่เมื่อจำนวนลำดับชั้นเพิ่มขึ้นความแตกต่างจะมากขึ้นเรื่อยๆ ดังนั้น จากการผลการทดสอบที่ได้ สรุปได้ว่าระบบที่มีสมาชิกหรือจำนวนลำดับชั้นน้อย K-PST(all) และ K-PST(one) จะให้ความเร็วค่อนข้างใกล้เคียงกัน

แม้ว่าวิธีการค้นหาแบบ K-PST ทั้งสองจะให้ค่าความเร็วที่ดีกว่าวิธี Signature และ Flooding ก็ตาม แต่เนื่องจากวิธีการที่ใช้ K-PST จำเป็นต้องสูญเสียค่าใช้จ่ายไปกับการประกาศข้อมูลคิดเป็นสองเท่าของวิธี Signature จากผลลัพธ์ของการทดลองจะเห็นได้ว่า เมื่อจำนวนโหนดในระบบมีค่าน้อย วิธี K-PST จะให้ค่าประสิทธิภาพ $e$ ที่ดีกว่า แต่เมื่อขนาดของระบบเพิ่มขึ้นถึงระดับหนึ่งจะเห็นได้ว่า วิธีที่ให้ค่าประสิทธิภาพ $e$ ที่ดีที่สุดคือ Signature โดยที่ K-PST จะมีค่าต่ำลง อย่างไรก็ตาม วิธี K-PST ก็ยังให้ค่าประสิทธิภาพ $e$ ที่ดีกว่า วิธี Flooding อยู่มาก

### 5.4.3 การทดสอบปัจจัยของความถี่การประกาศข้อมูล

การทดลองนี้เป็นการทดสอบปัจจัยของความถี่การประกาศข้อมูลมีผลต่อการความเร็วในการค้นหาข้อมูลและประสิทธิภาพของระบบมากน้อยเพียงใด และระบบควรจะประกาศข้อมูลบ่อยครั้งเพียงใดที่ทำให้ความเร็วในการค้นหาข้อมูลอยู่ในระดับที่ยอมรับได้ การทดลองนี้

เปรียบเทียบประสิทธิภาพของระบบการค้นหาด้วยมาตรวัดสองอย่างคือ ความเร็วการค้นหา $v$ และ ประสิทธิภาพของระบบ $e$

<u>สมมติฐานของการทดลอง</u>

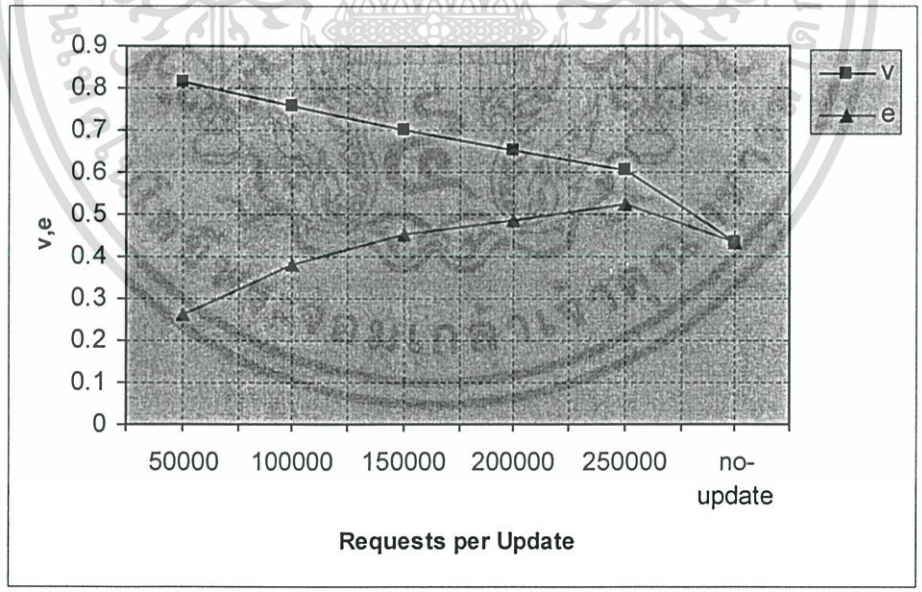
1. การประกาศข้อมูลและการอัพเดทแคชบ่อยครั้งน่าที่จะทำให้ความเร็วการค้นหาในระบบมีค่าที่ดีขึ้น แต่อาจจะทำให้ประสิทธิภาพของระบบโดยรวม $e$ ลดลง

<u>ปัจจัยการทดลอง</u>

ในการทดลองนี้มีการกำหนดปัจจัยการทดลอง เพิ่มเติมจากระบบทดสอบพื้นฐาน ดังนี้

1. จำนวนโหนดในโครงสร้างลำดับชั้น $N$ = 65,535 โหนด
2. ใช้วิธีการค้นหาแบบ K-PST(one)
3. ทดสอบความถี่การร้องขอแบบเอ็กโพเนนเชียล
4. การอัพเดทแคชและอัตราส่วนการร้องขอต่อการประกาศข้อมูลมีค่าคิดเป็น จำนวนครั้งต่อการร้องขอข้อมูลดังนี้ {50,000, 100,000, 150,000, 200,000, 250,000, ไม่มีการประกาศข้อมูล }

<u>ผลการทดลองและบทวิเคราะห์</u>




**รูปที่** 5.15 แสดงค่าของมาตรวัดประสิทธิภาพการค้นหา ในระบบที่มีอัตราส่วนการร้องขอต่อการประกาศข้อมูลแตกต่างกัน ตั้งแต่ 50,000 ไปจนถึง 250,000

พิจารณารูปที่ 5.15 พบว่าความเร็วการค้นหาข้อมูล $\nu$ มีอัตราส่วนการร้องขอต่อการประกาศข้อมูลตั้งแต่ 50,000 จนถึง 25,000 มีค่าลดลงเรื่อยๆ เมื่ออัตราส่วนการร้องขอต่อการประกาศข้อมูลมากขึ้น ส่วนประสิทธิภาพของระบบ $e$ จะเพิ่มขึ้นเรื่อยๆ จนมีค่าเท่ากับใกล้เคียงกับความเร็วในการค้นหา และเมื่อทดสอบกับระบบที่ไม่มีการประกาศข้อมูล ความเร็วในการค้นหา $\nu$ จะมีค่าเท่ากับการประกาศข้อมูล เนื่องจากไม่มีการประกาศข้อมูลในระบบ ความเร็วในการค้นหา $\nu$ ที่มีค่าลดลงนี้เกิดจาก การประกาศข้อมูลในระบบน้อยครั้ง ทำให้ค่าโครงสร้างข้อมูล K-PST ในส่วนที่เก็บข้อมูลแคช ไม่ได้รับการอัพเดทเป็นจำนวนที่เพียงพอ ซึ่งมีผลต่อจำนวนการเชื่อมต่อที่เอเจนต์สร้างขึ้นเพื่อสอบถามข้อมูลที่ต้องการไปยังเอเจนต์ตัวอื่นในระบบนั่นเอง

### 5.4.4 การทดลองปัจจัยของจำนวนโหนดลูกในลำดับชั้น

การทดลองนี้เป็นการทดสอบปัจจัยของจำนวนโหนดลูกในลำดับชั้น ว่ามีผลต่อการความเร็วในการค้นหาข้อมูลและประสิทธิภาพของระบบมากน้อยเพียงใด และหาจำนวนโหนดลูกในแต่ละลำดับชั้นที่เหมาะสมต่อการค้นหาด้วยวิธี K-PST(all) และ K-PST(one)

<u>สมมติฐานของการทดลอง</u>

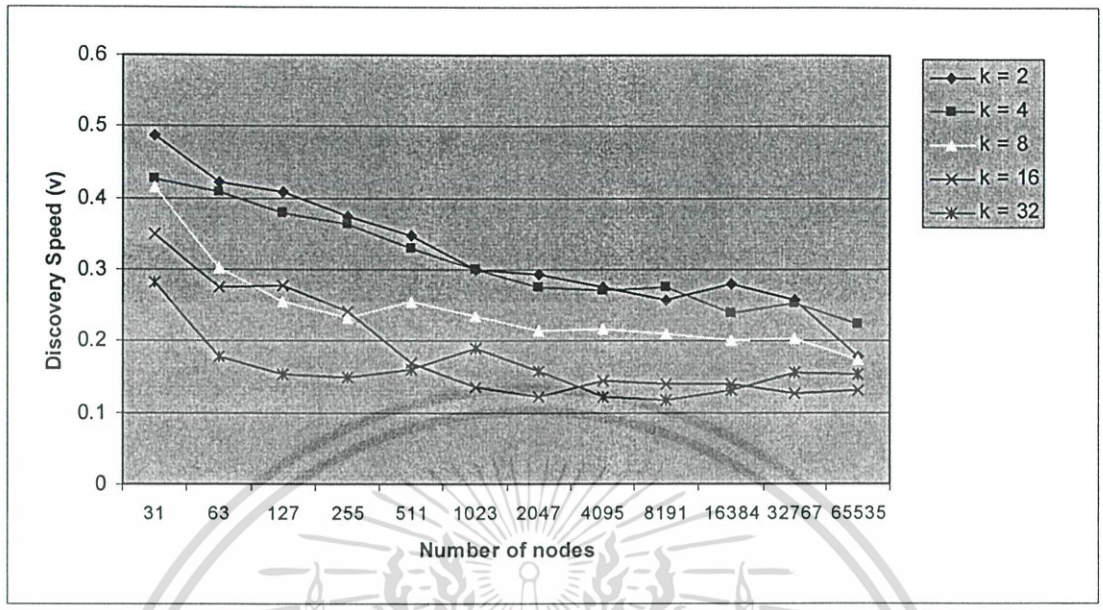
1. ความเร็วการค้นหาด้วยวิธี K-PST(all) และ K-PST(one) น่าจะให้ผลที่แตกต่างกัน เมื่อพิจารณาถึงโครงสร้างลำดับชั้นที่มีจำนวนลูก $k$ มาก เนื่องจาก การค้นหา K-PST(all) จำเป็นต้องค้นหาไปยังโหนดที่มีความเป็นไปได้ทุกโหนด ดังนั้น ระบบที่มีค่า $k$ น้อย น่าจะเอื้อประโยชน์ให้วิธีการค้นหา K-PST(all) มากกว่า K-PST(one) เนื่องจากมีการใช้จำนวนเส้นทางเพื่อค้นหาข้อมูลร่วมกันมาก ในขณะที่ ระบบที่มีค่า $k$ มาก น่าจะเอื้อประโยชน์ให้วิธีการค้นหา K-PST(one) มากกว่า K-PST(all) เนื่องจากสามารถเลือกเส้นทางที่ลู่เข้าสู่เป้าหมาย โดยที่ใช้จำนวนการเชื่อมต่อน้อย

<u>ปัจจัยการทดลอง</u>

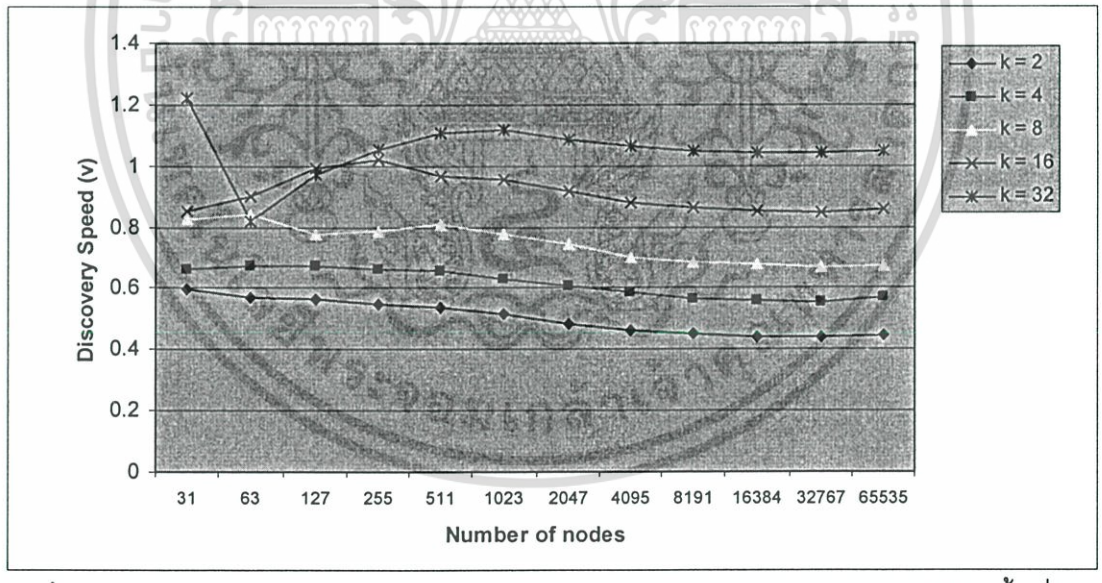
ในการทดลองนี้มีการกำหนดปัจจัยการทดลอง เพิ่มเติมจากระบบทดสอบพื้นฐาน ดังนี้

1. ใช้วิธีการค้นหาแบบ K-PST(all) และ K-PST(one)
2. กำหนดให้มีจำนวนโหนดในระบบ $N$ = { 31, 63, 127, ..., 65,535 }
3. การเชื่อมต่อในโครงสร้างลำดับชั้นเป็น $k$-ary Tree โดยที่ $k$ = { 2, 4, 8, 16, 32 }

ผลการทดลองและบทวิเคราะห์




**รูปที่** 5.16 ความเร็วการค้นหาด้วยวิธีการค้นหา K-PST (all) ในโครงสร้างแบบลำดับชั้นที่ $k$ = {2, 4, 8, 16, 32} และ จำนวนโหนด $N$ = {31, 63, 127, …, 65,535 }

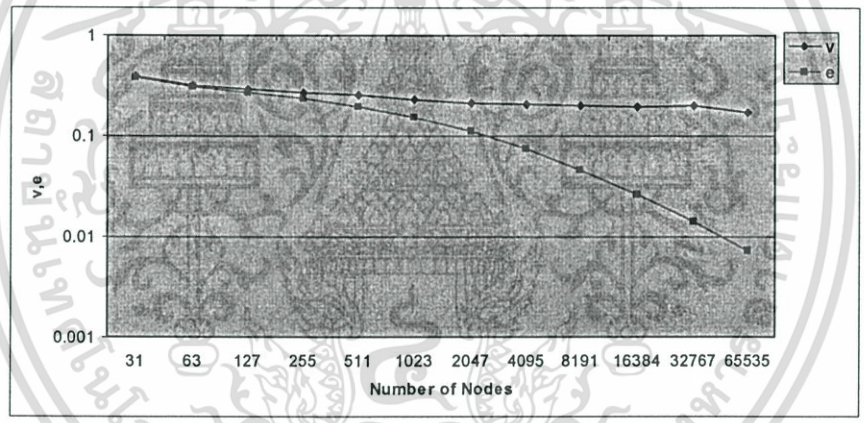



**รูปที่** 5.17 ความเร็วการค้นหาด้วยวิธีการค้นหา K-PST(one) ในโครงสร้างแบบลำดับชั้นที่ $k$ = {2, 4, 8, 16, 32} และ จำนวนโหนด $N$ = {31, 63, 127, …, 65,535 }

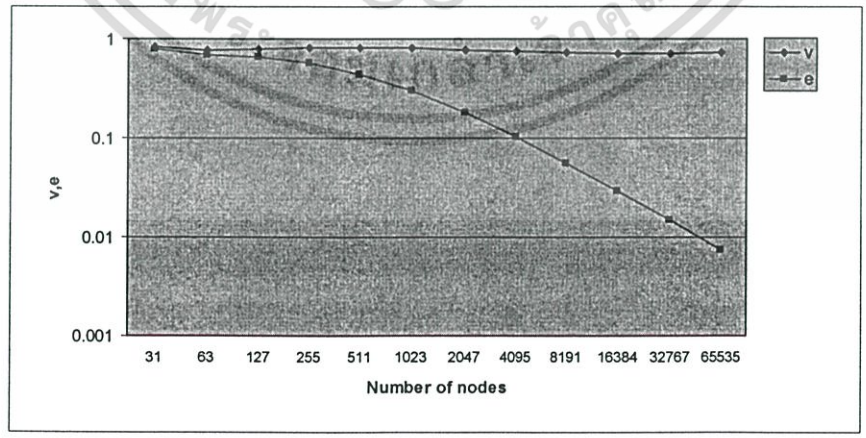
จากผลการทดลองที่ออกมาในรูปที่ 5.16 จะเห็นได้ว่าการค้นหาแบบ K-PST(all) ความเร็วในการค้นหา $v$ ในโครงสร้างลำดับชั้นที่ค่า $k$ น้อยส่วนใหญ่จะมีค่าดีกว่าโครงสร้างลำดับชั้นที่มีค่า $k$ มาก แต่ก็มีบางกรณีที่โครงสร้างลำดับชั้นที่มีค่า $k$ น้อยแต่กลับให้ความเร็วที่ต่ำกว่า

ปรากฏการณ์จะเกิดกับโครงสร้างลำดับชั้นที่มีค่า $k$ ใกล้เคียงกันเท่านั้น เมื่อจำนวนโหนดสมาชิกมีจำนวนมากขึ้นความเร็วการค้นหาของโครงสร้างลำดับชั้นฐานต่างๆ ก็จะมีค่าลดลง มีค่าความเร็วที่สลับกันให้ค่าที่ดี และลู่เข้าใกล้กันในที่สุด ข้อสังเกตที่ได้จากรูป 5.12 จะเห็นได้ว่าโดยส่วนใหญ่โครงสร้างลำดับชั้นที่ค่า $k$ =2 จะให้ค่าความเร็วสูงที่สุด แต่เมื่อจำนวนโหนดมากกว่า 4,095 ก็จะเริ่มให้ค่าความเร็วที่ดีสลับกับโครงสร้างที่มีค่า $k$ = 4 สุดท้ายโครงสร้างลำดับชั้นที่มีค่า $k$ ต่างๆ ก็จะมีค่า $v$ ไม่แตกต่างกันมากนัก

ในส่วนการค้นหาแบบ K-PST(one) ผลการทดลองที่ออกมาในรูปที่ 5.17 มีความแตกต่างกับวิธีการค้นหาแบบ K-PST(all) กล่าวคือ ในโครงสร้างลำดับชั้นที่มีค่า $k$ มากเกือบทั้งหมดจะให้ค่าความเร็วในการค้นหา $v$ ที่ดีกว่าโครงสร้างลำดับชั้นที่ค่า $k$ น้อย และ เมื่อจำนวนโหนดสมาชิกมีจำนวนมากขึ้นความเร็วการค้นหาที่ค่าค่า $k$ ใดๆ จะมีค่า $v$ ค่อยๆ ลดลงเมื่อมีจำนวนโหนดในโครงสร้างลำดับชั้นมากขึ้น จากนั้น เมื่อจำนวนโหนดตั้งแต่ 4,095 เป็นต้นไป ค่า $v$ ก็จะค่อนข้างคงที่




(a)




(b)

**รูปที่** 5.18 เปรียบเทียบความเร็วการค้นหาและประสิทธิภาพของระบบรวมโดยเฉลี่ยของ *k*-ary Tree ด้วย (a) K-PST(all) (b) K-PST(one)

เมื่อเปรียบเทียบความเร็วการค้นหาและประสิทธิภาพของระบบของ *k*-ary Tree ตามรูปที่ 5.18 จะเห็นได้ว่า วิธีการค้นหาแบบ K-PST(all) และ K-PST(one) มีลักษณะคล้ายคลึงกัน คือ เมื่อจำนวนโหนดน้อยความเร็ว $v$ และประสิทธิภาพ $e$ จะมีค่าสูง แต่เมื่อจำนวนโหนดมากขึ้นทั้งค่า $v$ และ $e$ จะมีค่าต่ำลงอย่างรวดเร็ว แต่ก็มีความแตกต่างกันเล็กน้อยตรงที่ ค่าความเร็ว $v$ ของการค้นหา K-PST(one) จะเริ่มมีค่าคงที่เมื่อจำนวนโหนดตั้งแต่ 4,095 เป็นต้นไปแต่ว่าความประสิทธิภาพ $e$ จะมีค่าลดลงในลักษณะเดิม ในขณะที่การค้นหาข้อมูลแบบ K-PST(all) ความเร็วจะมีแนวโน้มต่ำลงต่อไป

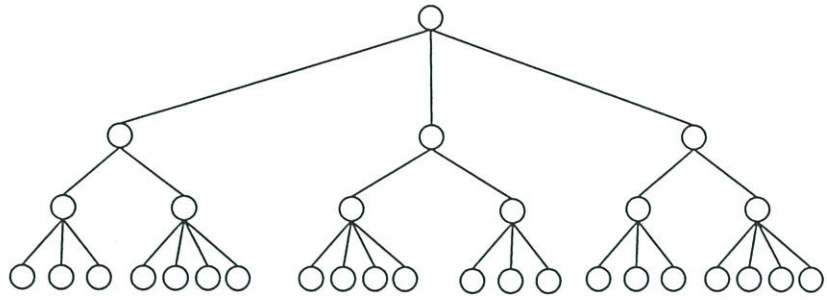
## 5.5 การเปรียบเทียบประสิทธิภาพกับงานวิจัยอื่น

การทดสอบประสิทธิภาพของระบบการค้นหาที่ได้พัฒนาขึ้นก่อนหน้านี้ เป็นการทดสอบกับสภาพแวดล้อมในระบบจำลอง ที่ศึกษาหาปัจจัยที่เกี่ยวข้องกับความเร็วการค้นหาและประสิทธิภาพโดยรวมของระบบ ผลการทดสอบที่ออกมากล่าวได้ว่ามีระบบที่พัฒนาขึ้นมีความสามารถในการลดจำนวนการเชื่อมต่อเพื่อสอบถามข้อมูลได้จำนวนหนึ่ง เพื่อให้เห็นความสามารถของกระบวนการที่พัฒนาขึ้นอย่างชัดเจน ในหัวข้อนี้จึงเป็นการเปรียบเทียบงานวิจัยกับงานวิจัยอื่น ที่พัฒนาขึ้นโดยมีเป้าหมายเดียวกันกับวิทยานิพนธ์ คือ ความเร็วในการค้นหา ผลการทดลองที่ได้จากการเปรียบเทียบ จะบ่งชี้ถึงข้อดีข้อเสียของกลไกการค้นหาทั้งในส่วนที่อยู่ในวิทยานิพนธ์นี้ และงานวิจัยของผู้อื่นในแง่มุมต่างๆ ซึ่งจะเป็นประโยชน์ต่อการพัฒนากลไกการค้นหาเหล่านี้ต่อไป

### 5.5.1 การทดลองเปรียบเทียบประสิทธิภาพกับ SDS และ QOS-AWARE

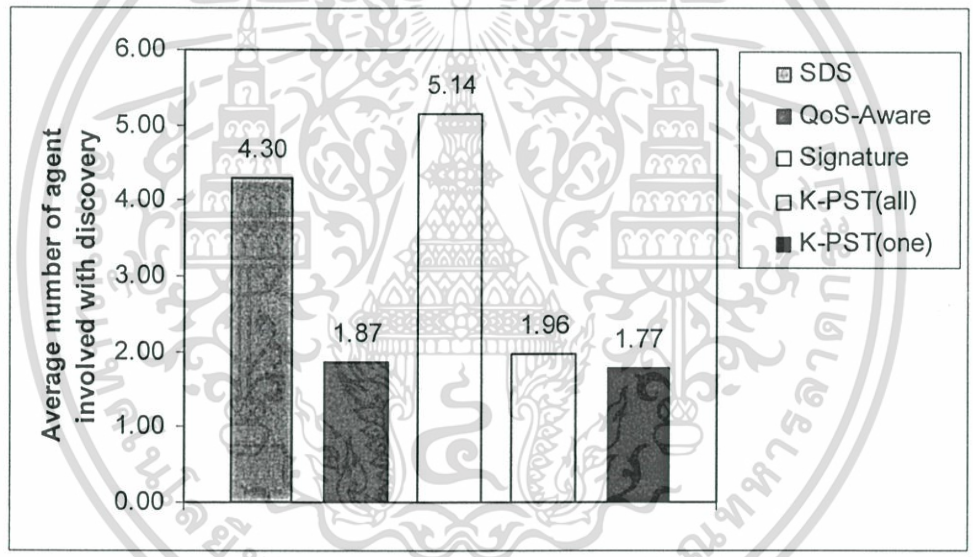
ในการทดลองนี้เป็นการเปรียบเทียบระบบการค้นหาข้อมูลภายในระบบกริดที่ได้พัฒนาขึ้นเปรียบเทียบกับงานวิจัยอื่น 2 ชิ้น คือ Service Discovery Service [47] หรือเรียกสั้นๆ ว่า SDS และงานวิจัย [40] ที่ประยุกต์ใช้การตอบสนอง QoS หรือเรียกสั้นๆ ว่า QoS-Aware ของเซิร์ฟเวอร์และไคลเอนต์ในระบบมาประยุกต์ใช้สำหรับการเชื่อมต่อแบบจุดต่อจุดระหว่างเซิร์ฟเวอร์ผู้ให้บริการข้อมูลที่ทำหน้าที่ค้นหาในระบบแบบกระจาย

สภาพแวดล้อมการทดลองที่ใช้ในทดลองนี้ อ้างอิงจากการทดลองที่ใช้ใน [40] ระบบดังกล่าวมีการเชื่อมต่อระหว่างเอเจนต์ในระบบเป็นจำนวนทั้งสิ้น 31 ตัว โดยมีลักษณะการเชื่อมต่อ 4 ลำดับชั้น ตามรูปที่ 5.19 ในระบบมีข้อมูลที่แตกต่างกัน 20 แบบ กระจายกันอยู่ในที่ต่างๆ เอเจนต์หรือเซิร์ฟเวอร์ให้บริการข้อมูลในระบบจะมีหน้าที่ดูแลข้อมูลเป็นจำนวนที่สุ่มค่า ตั้งแต่ 1 ถึง 3 ชิ้น มีอัตราการร้องข้อมูลจะเป็นจำนวนเฉลี่ย 1 ครั้งต่อวินาที โดยดำเนินการทดสอบเป็นจำนวนเวลาทั้งสิ้น 4000 นาที สำหรับแคชจะเก็บข้อมูลที่แตกต่างกันได้ 5 ประเภท



**รูปที่** 5.19 โครงสร้างแบบลำดับชั้น 4 ระดับ ที่ใช้สำหรับการทดสอบ

สำหรับวิธีการที่พัฒนาขึ้น ที่นำมาทดสอบร่วมประกอบด้วย การใช้ Signature และ การใช้ Knowledge ทั้งแบบค้นหา K-PST(all) และ K-PST(นำ) โดยกำหนดให้มีการประกาศข้อมูลทุกๆ 1000 วินาที แคชจะเก็บข้อมูลที่แตกต่างกันได้ 5 ประเภท




**รูปที่** 5.20 แผนภูมิแสดงจำนวนเฉลี่ยของเอเจนต์ที่เกี่ยวข้องต่อการเรียกใช้บริการ 1 ครั้ง

จากการทดลองพบว่า ระบบการค้นหาที่ได้ออกแบบขึ้นทั้ง 3 ส่วนมีการค้นหาที่รวดเร็วในระดับที่น่าพอใจ สังเกตจากแผนภูมิในรูปที่ 5.20 ที่แสดงจำนวนของเอเจนต์ในการค้นหาข้อมูล 1 ครั้ง จะเห็นได้ว่าการใช้งาน Signature ซึ่งเป็นรากฐานของระบบการค้นหาจะใช้เอเจนต์ราว 5 ตัวต่อการร้องขอข้อมูล 1 ครั้ง ประสิทธิภาพที่ออกมาถือว่าแย่กว่า SDS ที่ใช้เอเจนต์ราว 4 ตัวเล็กน้อย และแย่กว่า QoS-Aware ที่มีการใช้เอเจนต์เฉลี่ย 1.87 ตัวอยู่มาก

อย่างไรก็ดี เมื่อมีการใช้ Knowledge เข้ามาช่วยในระบบการค้นหา พบว่าความเร็วที่ได้ก็อยู่ในระดับที่น่าพอใจ กล่าวคือ ด้วยวิธีการค้นหาแบบ K-PST(all) จะมีค่าเฉลี่ยของเอเจนต์ที่เกี่ยวข้องเป็นจำนวน 1.96 แย่กว่า QoS-Aware เล็กน้อย แต่ได้ข้อมูล K-PST(all) ทั้งระบบ ในส่วน

ของการค้นหาแบบ K-PST(one) จะมีค่าเฉลี่ยของเอเจนต์ที่เกี่ยวข้องต่อการค้นหา 1 ครั้งเท่ากับ 1.77 ซึ่งถือว่าทำได้ดีกว่าวิธีการของงานวิจัย QoS-Aware เล็กน้อย

นอกจากนี้ จะเห็นได้จากการทดลอง ที่กล่าวมาแล้วว่าในระบบขนาดเล็กที่มีจำนวนลำดับชั้นน้อย วิธีการที่ใช้ K-PST ที่ออกแบบขึ้นทั้งสอง จะมีความสามารถในการค้นหาข้อมูลได้พอกัน โดยมีเอเจนต์ที่เกี่ยวข้องต่อการค้นหาราว 2 ตัว หรือ เทียบได้กับการเชื่อมต่อเพื่อค้นหาข้อมูล 1 ครั้งเท่านั้น

# บทที่ 6
# บทสรุป

## 6.1 สรุปงานวิจัยที่นำเสนอ

วิทยานิพนธ์นี้โดยได้พัฒนาระบบเอเจนต์ที่เชื่อมต่อกันแบบหลายทิศทางเพื่อเพิ่มประสิทธิภาพของกลไกการค้นหาข้อมูลในระบบการประมวลผลแบบกริด ให้มีความเร็วในการค้นหาข้อมูลมากขึ้น ผลลัพธ์ที่ได้จากการทดลองบ่งบอกข้อดีและข้อเสียของวิธีการเพิ่มประสิทธิภาพในหลายประเด็นที่แตกต่างกัน

การนำเอา Signature มาใช้เป็นโครงสร้างข้อมูลสำหรับอธิบายข้อมูลในโครงสร้างลำดับชั้นและนำมาประยุกต์ใช้ในกระบวนการค้นหาข้อมูล ช่วยให้ความเร็วการค้นหาเพิ่มขึ้นอย่างยิ่งยวดเมื่อเปรียบเทียบกับระบบการค้นหาแบบ Flooding ในโครงสร้างลำดับชั้น ที่มีการค้นหาไปยังทุกโหนดที่มีการเชื่อมต่อ ซึ่งถึงแม้ว่าจะมีการจำกัดรัศมีการค้นหาแล้วก็ยังถือว่าสิ้นเปลือง ผลลัพธ์ที่ได้จากการจำลองผล พบว่าการค้นหาที่ประยุกต์ใช้ Signature ในโครงสร้างลำดับชั้นช่วยให้ความเร็วการค้นหาของระบบ $v$ มีค่าสูงขึ้นมาก

เมื่อนำเอาแคชเข้ามาเพิ่มประสิทธิภาพในระบบการค้นหาที่มี Signature เป็นพื้นฐาน จะทำให้ความเร็วในการค้นหาข้อมูล $v$ มีค่าเพิ่มขึ้นอีกเล็กน้อย แต่ข้อเสียก็คือ ระบบจะต้องมีการประกาศข้อมูลเพื่อให้เอเจนต์ในโครงสร้างลำดับชั้นทราบถึงการเปลี่ยนแปลงข้อมูลในระบบ สิ่งที่ตามมาก็คือ ระบบจะต้องรับภาระในการประกาศข้อมูลมากขึ้น ทำให้ประสิทธิภาพ $e$ มีค่าลดลง ระบบการใช้งานจริงที่จะประยุกต์ใช้วิธีการที่นำเสนอไว้ในวิทยานิพนธ์นี้จำเป็นต้องพิจารณาความสิ้นเปลืองในจุดนี้ด้วย

การใช้ Knowledge ของเอเจนต์ที่เชื่อมต่ออยู่กับเอเจนต์ที่ค้นหาข้อมูลในโครงสร้างลำดับชั้น สามารถการเพิ่มประสิทธิภาพให้ระบบการค้นหาให้มีความเร็ว $v$ สูงที่สุด เนื่องจากในการค้นหาข้อมูลแบบครบถ้วน หรือ K-PST(all) จะลดการเชื่อมต่อเพื่อค้นหาข้อมูลในระบบ โดยจะสร้างการเชื่อมต่อเฉพาะโหนดที่มีความเป็นไปได้ที่พบข้อมูลที่ต้องการเท่านั้น นอกจากนี้ เมื่อกำหนดให้มีการค้นหาด้วยการจำกัดจำนวนเส้นทางการค้นหา ให้ค้นหาข้อมูลเพียง 1 เส้นทาง หรือ K-PST(one) สามารถเพิ่มความเร็ว $v$ ในการค้นหาโดยรวมมากขึ้นไปอีก แต่ข้อเสียของการใช้ Knowledge ก็คือ เอเจนต์แต่ละตัวต้องเก็บข้อมูล Signature ของรีซอร์สไว้หลายชุด และการประกาศข้อมูลต้องมีการส่งข้อมูลที่มีขนาดใหญ่กว่าการใช้ Signature เพียงอย่างเดียว

ในการค้นหาข้อมูลด้วยการใช้ K-PST(all) พบว่า โครงสร้างลำดับชั้นที่มีฐานของลูกเป็นจำนวนน้อยจะให้ค่าความเร็วสูงกว่าโครงสร้างลำดับชั้นที่มีฐานของลูกเป็นจำนวนมาก ตรงกันข้าม

กับการค้นหาแบบ K-PST(one) ที่โครงสร้างลำดับชั้นที่มีฐานมากส่วนใหญ่จะมีค่าดีกว่าโครงสร้างลำดับชั้นที่มีจำนวนฐานน้อย

เมื่อเปรียบเทียบระหว่างการค้นหาด้วย Signature และ K-PST โดยกำหนดให้ความถี่การประกาศข้อมูลเท่ากัน ผลการทดลองที่ออกมาชี้ให้เห็นว่า เมื่อระบบมีขนาดใหญ่ขึ้น ความเร็ว $v$ ของการค้นหา K-PST จะดีกว่า Signature แต่ค่าประสิทธิภาพ $e$ ของการค้นหาด้วย Signature จะให้ค่าที่ดีกว่าการค้นหาด้วย K-PST หมายความว่า ถึงแม้การค้นหาด้วย K-PST จะมีความเร็วที่ดีกว่าแต่ว่าเมื่อระบบใหญ่ขึ้นการค้นหาด้วย K-PST ถือว่าไม่คุ้มค่าเมื่อเปรียบเทียบกับการค้นหาแบบ Signature ที่เสียค่าใช้จ่ายในส่วนการค้นหาและการประกาศข้อมูลน้อยกว่า

สำหรับระบบการค้นหาที่สร้างขึ้น วิธีการที่ใช้ Knowledge และการใช้แคช เพื่อให้ได้ความเร็วการค้นหา $v$ ที่เพิ่มขึ้น จำเป็นต้องอาศัยการประกาศข้อมูลในระบบ แต่ก็ไม่มีความจำเป็นต้องมีการประกาศข้อมูลบ่อยครั้ง เพราะถึงแม่ว่าจะมีการประกาศข้อมูลน้อยครั้ง ความเร็วที่ได้ก็จะไม่ตกลงไปมากนัก ในทางตรงกันข้าม การประกาศข้อมูลบ่อยครั้งทำให้ระบบโดยรวมมีภาระกับการประกาศข้อมูลมากขึ้นทำให้ค่าประสิทธิภาพ $e$ มีค่าน้อยลง แต่ความเร็วการค้นหาที่ได้กลับมานั้น ถือว่ามีสัดส่วนการเพิ่มขึ้นน้อยกว่า

## 6.2 ปัญหาและอุปสรรค

ในปัจจุบันเทคโนโลยีการประมวลผลกริดอยู่ในระยะเริ่มต้น ระบบที่ใช้งานจริงจึงมีขนาดเล็ก กล่าวคือ อยู่ในระดับแลนหรือแคมปัสเนตเวิร์กเท่านั้น การวิจัยจึงยังอยู่ในวงปิดไม่มีการเผยแพร่ข้อมูลเพื่อทดสอบมากนัก ข้อมูลที่ใช้อ้างอิงที่หามาได้จึงถือได้ว่ามีน้อยอยู่

สำหรับงานวิจัยที่เกี่ยวข้องกับการค้นหาข้อมูลในระบบกริดนั้น ยังไม่เป็นที่แพร่หลายนักการวิจัยจึงอยู่ในกลุ่มที่จำกัด อีกทั้งการดำเนินงานวิจัยแต่ละโครงการ จะใช้มาตรวัดสภาพแวดล้อมและปัจจัยการทดสอบที่แตกต่างกัน โดยไม่มีมาตรวัดและสภาพแวดล้อมที่เป็นมาตรฐานกลางสำหรับการทดสอบ ผู้รับผิดชอบงานวิจัยเหล่านั้นจะทดสอบตามแหล่งการอ้างอิงของตน ผลการวิจัยที่ออกมาก็จะชี้ให้เห็นวิธีการที่เหมาะสมต่อสภาพแวดล้อมตามแหล่งข้อมูลที่นำมาอ้างอิงเท่านั้น ดังนั้น งานวิจัยในวิทยานิพนธ์ชิ้นนี้จึงบอกได้ยากว่าอยู่ในระดับใด เมื่อเปรียบเทียบกับวิธีการอื่น ในสภาพแวดล้อมต่างๆ

งานวิจัยที่เกี่ยวข้องหลายชิ้นมีวิธีการที่ดีสำหรับการเพิ่มความเร็วในการค้นหา แต่ในเอกสารที่เผยแพร่นั้น ยังขาดรายละเอียดเชิงลึกของกลไกการทำงาน อีกทั้งยังขาดสภาพแวดล้อมและมาตรวัดที่เป็นมาตรฐานตามที่กล่าวมาแล้ว ดังนั้น จึงเป็นการยากที่จะมีการเปรียบเทียบประสิทธิภาพกับระบบที่ออกแบบขึ้นมาใหม่ได้อย่างเหมาะสม เนื่องจากไม่สามารถอิมพลีเมนต์งานวิจัยอื่นเปรียบเทียบประสิทธิภาพในระบบจำลองที่มีสภาพแวดล้อมเดียวกันได้ และการ

เปรียบเทียบตัวเลขของมาตรวัดที่ออกมาเทียบกัน แม้จะบอกประสิทธิภาพได้บางส่วนแต่ก็ยังถือว่าไม่ยุติธรรมเท่าใดนัก

สำหรับการทำการทดลองสำหรับงานวิจัยชิ้นนี้ ถึงแม้ว่าระบบที่ออกแบบขึ้นสามารถให้ความเร็ว $v$ ในการค้นหาข้อมูลได้ดีก็จริง แต่ผลที่ได้นั้น เกิดจากการประเมินประสิทธิภาพด้วยการจำลองผลที่มีการจำกัดปัจจัยการทดลองเพียงส่วนหนึ่งเท่านั้น หากนำวิธีการนี้มาทดสอบกับระบบจริงที่จะเกิดขึ้นในอนาคตและมีปัจจัยต่อความเร็วการค้นหามากกว่านี้ ก็อาจจะทำให้วิธีการที่นำเสนอในงานวิจัยนี้ให้ผลแตกต่างจากที่ทดลองในสภาพแวดล้อมจำลองก็เป็นได้

ปัญหาอุปสรรคสำหรับการดำเนินการวิจัยก็คือ ความพร้อมของอุปกรณ์สำหรับประมวลผลการทดสอบด้วยการจำลองผล งานวิจัยชิ้นนี้ใช้เวลาทดสอบด้วยอุปกรณ์ที่จัดหามาเองตามกำลังที่พอจะหามาได้ ซึ่งใช้เวลาสำหรับทดสอบแต่ละการทดลองล่าช้ามาก เนื่องจากระบบจำลองที่สร้างขึ้นมีองค์ประกอบที่เกี่ยวข้องกับจำนวนโหนดของเอเจนต์และการร้องขอข้อมูลจำนวนมาก ซึ่งเอเจนต์ในการทดลองมีจำนวนหกหมื่นโหนด และการร้องขอไม่ต่ำกว่าหนึ่งหมื่นครั้ง จึงมีความจำเป็นต้องอาศัยระบบคอมพิวเตอร์สมรรถนะสูงในการประมวลผล เพื่อความรวดเร็วสำหรับการทดลองในแง่มุมต่างๆ

## 6.3 แนวทางการพัฒนาต่อ

ในวิทยานิพนธ์นี้ได้ดำเนินการทดสอบกลไกการค้นหาที่ได้ออกแบบขึ้นกับปัจจัยที่เกี่ยวข้องกับความเร็วการค้นหา $v$ และประสิทธิภาพ $e$ เพียงบางส่วนเท่านั้น ประเด็นสำคัญอื่นๆ อย่าง ระยะทางของการเชื่อมต่อ จำนวนช่องทางสื่อสาร ความหนาแน่นของเครือข่าย ความถูกต้องของการค้นหา จำนวนการพบข้อมูล ก็เป็นปัจจัยที่น่าสนใจต่อการทดลองเมื่อระบบมีการให้ใช้งานจริง นอกจากนี้การเพิ่มประสิทธิภาพของระบบการค้นหาด้วยวิธีการต่างๆ เช่น การจัดการเอเจนต์ในแต่ละลำดับชั้น วิธีการเลือกช่องทางและจัดการจำนวนการส่งข้อมูล รวมถึงอัลกอริธึมการจัดการแคช ก็สามารถทำได้และเป็นหัวข้องานวิจัยที่สำคัญที่สามารถนำไปเอาวิธีการที่นำเสนอไว้นี้ไปพัฒนาต่อได้

ระบบการค้นหาข้อมูลอย่างรวดเร็วที่นำเสนอไป ถือเป็นรากฐานของระบบการจัดการข้อมูล ที่เป็นส่วนสำคัญในระบบประมวลผลแบบกริด หากนำวิธีการนี้ไปประยุกต์ใช้งาน ก็สามารถเพิ่มประสิทธิภาพได้ในหลายส่วน เช่น การจัดตารางการประมวลผลข้อมูล การจัดการคุณภาพของข้อมูลในช่องสื่อสาร การตรวจสอบบำรุงรักษาระบบ และการเลือกรีซอร์สที่มีคุณภาพดีก็สามารถทำได้ เชื่อเป็นอย่างยิ่งว่าหากระบบการจัดการข้อมูลมีพื้นฐานของระบบการค้นหาที่ดีก็น่าที่จะเพิ่มประสิทธิภาพให้แก่ระบบจัดการรีซอร์สได้ดียิ่งขึ้น

ระบบการค้นหานี้ประยุกต์ใช้งาน Signature File ซึ่งนอกจากจะเอื้อต่อการค้นหาข้อมูลอย่างรวดเร็วแล้ว ยังมีความยืดหยุ่นต่อการค้นหาข้อมูลได้อีกด้วย เราสามารถประยุกต์ใช้ Signature และ K-PST ไปใช้ในระบบการค้นหาที่มีความสามารถมากกว่าเดิม อย่างเช่น ในปัจจุบันข้อมูลที่ค้นหาต้องมีลักษณะถูกต้องสมบูรณ์ (exact match) เท่านั้น แต่ด้วยการใช้งาน Signature ของรีซอร์สเราสามารถสร้างระบบการค้นหาข้อมูลที่มีลักษณะถูกต้องบางส่วน (partial match) โดยการกำหนดค่าแฮชชิ่งฟังก์ชันของข้อมูลหรือเซอร์วิสที่มีลักษณะใกล้เคียงกันหรือมีค่าความคล้ายคลึงกันสูง (high similarity) ให้มีค่าเท่ากับ หรือใกล้เคียงกัน ซึ่งจะทำให้เราสามารถค้นหาข้อมูลที่มีลักษณะใกล้เคียงกันได้ด้วยการค้นหาเพียงครั้งเดียว

นอกจากการสร้างระบบการค้นหาที่ถูกต้องบางส่วนแล้ว การใช้ประโยชน์จาก Signature ยังสามารถนำไปใช้กับระบบการสืบค้นข้อมูลหรือการทำเหมืองข้อมูลในระบบกระจายขนาดใหญ่อย่างกริด ที่ต้องการความสามารถในการทำงานด้วยบูลีน (Boolean Operation) ได้อีกด้วย โดยกำหนดให้ข้อมูลในเมสเซสที่ส่งไปค้นหาให้ตรงกับความต้องการและเหมาะสม จากการทดลองจะเห็นว่าระบบการค้นหาแบบครบถ้วนนั้นมีประสิทธิภาพที่ดีในระดับที่น่าจะนำไปสืบค้นข้อมูล ที่วางอยู่ในระบบเครือข่ายได้ในเวลาอันรวดเร็ว มีความเป็นไปได้ที่จะนำมาประยุกต์ใช้งานกับระบบซึ่งเกี่ยวข้องกับข้อมูลจำนวนมาก ด้วยประสิทธิภาพที่เป็นที่น่าพอใจ

วิธีการค้นหาที่พัฒนาขึ้นในงานวิจัยนี้ นอกจากจะถูกนำมาใช้ในระบบประมวลแบบกริดที่ได้นำเสนอไปแล้วในวิทยานิพนธ์ ผู้วิจัยเห็นว่าวิธีการที่สร้างขึ้นอาจจะนำมาปรับปรุงเพื่อเป็นกลไกการค้นหาข้อมูลในระบบประมวลผลแบบกระจายอื่นๆ ได้เช่น ระบบเครือข่าย ad hoc และระบบเครือข่ายเพียร์ทูเพียร์ เป็นต้น

# เอกสารอ้างอิง

[1] I. Foster and C. Kesselman. "Globus: A metacomputing infrastructure toolkit" *International Journal of Supercomputing Applications*. vol.11, no.2, 1997. pp.115–128.

[2] I. Foster, and C. Kesselman, (et all,.). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[3] I. Foster, C. Kesselman, and S. Tuecke. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations" *International Journal of High Performance Computing Applications*, vol.15, no. 3, 2001. pp.200-222.

[4] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky. "SETI@home-massively distributed computing for SETI " *IEEE Computing in Science & Engineering*, vol. 3, no. 1, Jan/Feb 2001. pp.78-83.

[5] J. Marin. "Web Services: The Next Big Thing" *XML-Journal 2*. [Online]. Available : http://www.sys-con.com/xml/. 2002.

[6] World Wide Web Consortium. "Web Service Architecture" [Online]. Available : http://www.w3.org/TR/ws-arch/. 2002.

[7] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration" [Online]. Available : http://www.globus.org/research/papers/ogsa.pdf. January 2002.

[8] M. Thompson, W. Johnston, W. Mudumbai, G. Hoo, K. Jackson, and A. Essiari, "Certificate-based Access Control for Widely Distributed Resources" *In Proc. 8th Usenix Security Symposium*, 1999.

[9] I. Foster, "Internet Computing and the Emerging Grid. Nature Web Matters" [Online]. Available : http://www.nature.com/nature/webmatters/grid/grid.html. 2000.

[10] World Wide Web Consortium. "Extensible Markup Language (XML) 1.1." *W3C Recommendation*, [Online]. Available : http://www.w3.org/TR/xml11/. October 2002.

[11] A. Birrell and B. Nelson, "Implementing Remote Procedure Calls." *ACM Transaction on Computer and System*, vol. 2, no. 1, Febuary 1984, pp.39-59.

[12] World Wide Web Consortium. "Simple Object Access Protocol (SOAP) 1.2." *W3C Recommendation*, [Online]. Available : http://www.w3.org/TR/soap12-part1/. May 2003.

[13] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. "Grid Information Services for Distributed Resource Sharing" *In 10th IEEE Int'l. Symposium on High-Performance Distributed Computing (HPDC-10)*, San Francisco, California, August 2001.

[14] N. Furmento, W. Lee, A. Mayer, S. Newhouse, and J. Darlington. "ICENI: An Open Grid Service Architecture Implemented with JINI" *Supercomputing 2002*, 2002.

[15] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. "*Web Services Description Language (WSDL) 1.1*" W3C Note 15. [Online]. Available : http://www.w3.org/TR/wsdl. 2001.

[16] UDDI Consortium. "*UDDI: Universal Description, Discovery and Integration.*" [Online]. Available : http://www.uddi.org. 2002.

[17] S. J. Chapin, D. Katramatos, J. Karpovich, and A.Grimshaw, "Resource Management in Legion" *Future Generation Computer Systems*, vol.15, no.5, 1999. pp.583-594.

[18] R. Raman, M. Livny, and M. Solomon, "Matchmaking: Distributed Resource Management for High Throughput Computing" *In Proc. of 7th IEEE Int. Symp. on High Performance Distributed Computing*, Chicago, USA, July 1998.

[19] H. Casanova, and J. Dongarra. "NetSolve: A Network Server for Solving Computational Science Problems" *International Journal of Supercomputer Applications and High Performance Computing*, vol.11 no.3, 1997. pp.212-223.

[20] H. Nakada, M. Sato, S. and Sekiguchi. "Design and Implementations of Ninf: towards a Global Computing Infrastructure" *Future Generation Computing Systems*, 1999.

[21] F. Kon, R. Campbell, M. Mickunas, and K. Nahrstedt, "2K: A Distributed Operating System for Dynamic Heterogeneous Environments" *In Proc. of 9th IEEE Int. Symp. on High Performance Distributed Computing*, 2000.

[22] L. Boloni, and D. Marinescu, "An Object-Oriented Framework for Building Collaborative Network Agents" *In A. Kandel et al, eds, Agents in Intelligent Systems and Interfaces, Kluewer*, 1999.

[23] J. Cao, D. J. Kerbyson, and G. R. Nudd, "Performance Evaluation of an Agent-Based Resource Management Infrastructure for Grid Computing" *Proceedings of 1st IEEE/ACM International Symposium on Cluster Computing and the Grid*. Brisbane, Australia, May 2001, pp.311–318.

[24] J. Bray, and C. Sturman, *Bluetooth: Connect Without Cables*, Prentice Hall, 2000.

[25] "Jini™ Architectural Overview" Sun Technical White Paper, Jan. 1999.

[26] R. Pascoe, "Building Networks on the Fly" *IEEE Spectrum*, vol. 38, no. 3, pp.61-65, 2001.

[27] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2" *RFC 2608*, IETF Draft Standard, 1998.

[28] "Understanding Universal Plug and Play" *Microsoft White Paper*, June 2000.

[29] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. "A Directory Service for Configuring High-Performance Distributed Computations" *In Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing (HPDC-6)*, August 1997.

[30] IETF M. Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3)" *RFC 2251*, December 1997.

[31] S. Ratsanamy, P. Francis, M. Handley, R. Karp, and S. Shenker. "A Scalable Content Addressable Network" *In Proceedings of SIGCOMM 2001*, 2001.

[32] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications" *In Proceedings of SIGCOMM 2001*, 2001.

[33] A. Rowstron and P. Druschel. "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems" *In Proceedings of the IFIP/A CM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, November 2001. pp.329-350.

[34] D. Watts. *Small Worlds. The Dynamics of Networks between Order and Randomness*. Princeton University Press, Princeton, New Jersey, U.S.A., 1999.

[35] A. Iamnitchi, M. Ripeanu, and I. Foster. "Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations" *In Proceedings of the First International Workshop on Peer-to-Peer Systems Cambridge, Massachusetts,* March 2002.

[36] J. Cao, D. P. Spooner, J. D. Turner, S. A. Jarvis, D. J. Kerbyson, S. Saini, and G. R. Nudd, "Agent-based Resource Management for Grid Computing" *Proceedings of 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*. Berlin, Germany, May 2002. pp.323-324.

[37] John Moy, "*Ospf version 2, rfc 1583*" [Online]. Available : http://vw.dsi.unive.it/Connected/RFC/1583/index.html, 1994.

[38] "*JiPang System*" [Online]. Available : http://ninf.is.titech.ac.jp/jipang/

[39] Z. Juhasz, A. Andics, and S. Pota, "JM: A Jini Framework for Global Computing" *In the 2nd International Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems at IEEE/ACM International Symposium on Cluster Computing and the grid (CCgrid'2002)*, Berlin Germany, May 21-24, 2002.

[40] D. Xu, K. Nahrstedt, and D. Wichadakul, "QoS-Aware Discovery of Wide-Area Distributed Services" *First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'01)*, May 15–18, Brisbane Australia, 2001.

[41] S.A. McIlraith, T.C. Son, and Honglei Zeng. "Semantic Web Service" *IEEE Intelligent Systems*, vol.16, no.2, Mar/Apr, 2001. pp.46-53,

[42] B. Bloom. "Space/time tradeof in in hash coding with allowable errors" *CACM*, vol.13, no.7, July 1970. pp.422-426.

[43] A. Broder and M. Mitzenmacher. "Network applications of Bloom Filters: A survey" *In Proceedings of 40th Annual Allerton Conference on Communication, Control, and Computing*; October, 2002. pp.636-646.

[44] R. Baeze-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. New York : Addison-Wesley Publishing Company, Inc.1999.

[45] J. Bloomers, *Practical Planning For Network Growth*. 1st Edition. Prentice Hall PTR, April, 1996.

[46] A.M. Law, and W.D. Kelton, *Simulation Modeling and Analysis*. 3rd edition. McGraw-Hill Series in Industrial Engineering and Management Science, 2000.

[47] S. Czerwinski, B.Zhao, T.Hodes, A. Joseph, and R. Katz, "An Architecture for a Secure Service Discovery Service" *ACM Baltzer Wireless Networks*, vol. 8, no. 2-3, March 2002. pp.213-230.

[48] M. Harchol-Balter, T. Leighton, D. Lewin, "Resource discovery in distributed networks" *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, Atlanta, Georgia, United States, May 1999. pp.229-237.

[49] W. Poompattanapong, B. Piyatammrong. "A Web Service Approach to Grid Information Service" *In The First International Conference on Web Service (ICWS'03)*, Las Vegas, Nevada, USA, June 2003. pp.420-423.

[50] W. Poompattanapong, B. Piyatamrong. "A Multi-Agent for Grid Service Management" *In The 3rd International Symposium on Communications and Information Technologies (ISCIT'03)*. Songkla, Thailand, September 2003. pp.735-739.

[51] M. Bunruangses, W. Poompattanapong, P. Banyatnoparat, B. Piyatamrong. "QoS Multi-Agent Applied for Grid Service Management" *In The 3rd International Symposium on Information and Communication Technologies (ISICT'04)*. Las Vegas, Nevada, USA, June 2004. pp.74-79.

# ภาคผนวก

# ภาคผนวก ก.
# ทฤษฎี Queuing

ในงานวิจัยนี้ นำเอาทฤษฎี Queuing [45] อธิบายกระบวนการร้องขอข้อมูลในระบบ มีความเกี่ยวข้องกับการให้บริการการค้นหาของเอเจนต์ที่ทำหน้าที่เป็นดิสคัฟเวอรีเซอร์เวอร์ และการเข้ามาของการร้องขอข้อมูลของสมาชิกในระบบทฤษฎี Queuing นี้จะถูกอิมลิเมนต์ด้วยโมเดลการจำลองผลที่ชื่อว่า Discrete Time Event Model [45, 46] เพื่อวิเคราะห์ประสิทธิภาพของระบบการค้นหาข้อมูลด้วยคอมพิวเตอร์ โมเดลการจำลองผลนี้เป็นการจำลองผลรูปแบบหนึ่งที่มีความน่าเชื่อถือ และถูกนำมาใช้อย่างแพร่หลายในงานวิจัยทางด้านการสื่อสารข้อมูลเครือข่ายคอมพิวเตอร์

ระบบ Queuing เป็นระบบที่ประกอบด้วยเซิร์ฟเวอร์จำนวนตั้งแต่ 1 ตัวขึ้นไป ที่ให้บริการบางอย่างแก่ลูกค้าที่เข้ามาใช้บริการ เมื่อลูกค้าเข้ามาใช้บริการแต่พบว่าเซิร์ฟเวอร์ไม่ว่างให้บริการก็จะต่อคิวรอการให้บริการของเซิร์ฟเวอร์ที่ว่างให้บริการ ซึ่งเป็นที่มาของชื่อ "Queuing" นั่นเอง

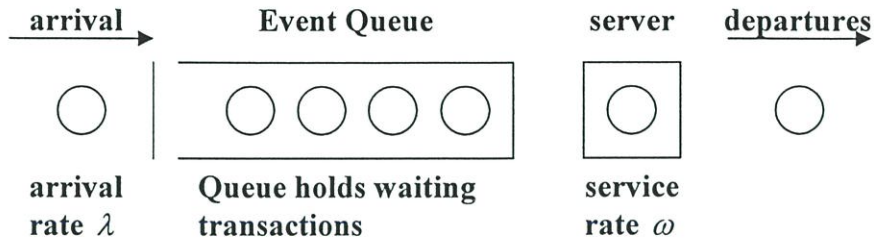
**ตารางที่ ก.1** ตัวอย่างของระบบ Queuing

| ระบบ | เซิร์ฟเวอร์ | ลูกค้า |
|---|---|---|
| ธนาคาร | พนักงานรับจ่ายเงิน | ลูกค้า |
| โรงพยาบาล | แพทย์ พยาบาล เตียง | คนไข้ |
| ระบบคอมพิวเตอร์ | CPU เครื่องมืออินพุทและเอาท์พุท | งาน |
| โรงงาน | คนงาน เครื่องมือ | ชิ้นส่วนงาน |
| สนามบิน | รันเวย์ ประตู สถานีตรวจสอบความปลอดภัย | เครื่องบิน ผู้โดยสาร |
| ระบบเครือข่ายสื่อสารข้อมูล | โหนด เส้นทางเชื่อมต่อ | แพ็กเก็ต เมสเซส |

ทฤษฎี Queuing ถูกนำมาประยุกต์ใช้สำหรับการจำลองผลระบบงานด้านต่างๆ ยกตัวอย่างเช่น ในตารางที่ ก.1 แสดงการระบบ Queuing ที่มีอยู่จริง และถูกอธิบายด้วยทฤษฎี Queuing เพื่อวิเคราะห์ประสิทธิภาพของระบบงานนั้น

## ก.1 องค์ประกอบของระบบ Queuing

ระบบ Queuing ถูกอธิบายลักษณะการทำงานด้วยมีองค์ประกอบ 3 อย่าง ได้แก่ การเข้ามาของการร้องขอ (arrival process) การให้บริการ (service mechanism) และ การจัดการคิว (Queue Discipline)




**รูปที่** 5.1 ลักษณะทั่วไปของระบบ Queuing

การเข้ามาของการร้องขอ กล่าวถึง การเข้ามาใช้บริการระบบของผู้ใช้บริการ กำหนดให้ $A_i$ เป็นช่วงเวลาระหว่างการเข้ามาใช้บริการของผู้ใช้บริการลำดับที่ $i-1$ กับลำดับที่ $i$ ถ้าหากระบุได้ว่าการเข้ามาของการร้องขอ $A_1, A_2, \ldots$ เป็นเลขสุ่มที่มีการกระจายที่แน่นอน (Independently and Identically Distributed: IID) เราจะสามารถกำหนดค่าเฉลี่ยของการเข้ามาด้วยค่าที่คาดหวัง $E(A)$ และประเมินอัตราค่าการเข้ามาของการร้องขอของระบบด้วย $\lambda = 1/E(A)$ เพื่อทดสอบประสิทธิภาพของระบบได้

การให้บริการ กล่าวถึงการวิธีการให้บริการของเซิร์ฟเวอร์ที่มีจำนวน $s$ โดยที่เซิร์ฟเวอร์แต่ละตัวสามารถมีคิวของตัวเอง หรือมีคิวของระบบที่คอยป้อนให้แก่เซิร์ฟเวอร์แต่ละตัวก็ได้ เซิร์ฟเวอร์เหล่านี้สามารถให้บริการแก่ลูกค้าด้วยค่าลาที่มีการกระจายค่าหนึ่ง กำหนดให้ $S_i$ เป็นเวลาที่ใช้สำหรับการให้บริการลูกค้าที่เข้าในระบบลำดับที่ $i$ ถ้า $S_1, S_2, \ldots$ เป็นเลขสุ่มแบบ IID เราจะสามารถประเมินประสิทธิภาพของระบบด้วยค่าเฉลี่ยการให้บริการได้ด้วย $E(S)$ และประเมินค่าอัตราการให้บริการลูกค่า $\omega = 1/E(S)$

การจัดการคิว หมายถึง กฎที่เซิร์ฟเวอร์ใช้สำหรับเลือกลูกค้าลำดับถัดไปขึ้นมาจากคิวเพื่อให้บริการ หลังจากที่ให้บริการลูกค้าลำดับก่อนหน้านี้เสร็จเรียบร้อยแล้ว การจัดการคิวที่มีใช้กันอยู่โดยทั่วไปมี ดังนี้

FIFO ลูกค้าที่เข้ามาในระบบก่อน จะถูกเลือกขึ้นมาให้บริการก่อน

LIFO ลูกค้าที่เข้ามาในระบบลำดับล่าสุด จะถูกเลือกขึ้นมาให้บริการก่อน

Priority เซิร์ฟเวอร์จะเลือกลูกค้าที่มีลำดับความสำคัญสูงสุดขึ้นมาให้บริการก่อน

## ก.2 สัญลักษณ์ของระบบ Queuing

เนื่องจากทฤษฎี Queuing ถูกนำมาใช้อย่างแพร่หลายในการจำลองระบบงานลักษณะต่าง ดังนั้น เพื่อให้มีการอ้างอิงเป็นมาตรฐานเดียวกัน จึงได้มีการสร้างสัญลักษณ์มาตรฐานเรียกว่า

Kendall Notation เพื่ออธิบายลักษณะระบบ Queuing ไว้ดังนี้ *M/G/s/K/P/QD* โดยมีรายละเอียดซึ่งแสดงไว้ในตารางที่ ก.2

**ตารางที่ ก.2** ความหมายของสัญลักษณ์ของระบบ Queuing

| *M* | *G* | *s* | *K* | *P* | *QD* | |
|---|---|---|---|---|---|---|
| **Interarrival Time Distribution** | **Service Time Distribution** | **The number of servers** | **System Capacity** | **Population in the System** | **Queue Discipline** | |
| ***D*** = Deterministic fixed value<br>$E_r$ = Erlangian with *r* stage<br>***G*** = General (any) distribution<br>***GI*** = General Independent<br>***M*** = Memoryless exponential<br>***U*** = Uniform Distribution | | The number of servers may vary from 1 to Infinity. | The maximum number of customers' packets items or transactions the system can hold. | The actual number of customers or transaction in the system | **FIFO**<br>**LIFO**<br>**HOL**<br>**FIRO** | First-in First-out<br>Last-in First-out<br>Head of Line<br>First-in Random-out |

ระบบ Queuing ต่างๆ จะตั้งชื่อโดยสื่อความหมายตามนิยามที่ให้ไว้ในตารางที่ ก.2 ยกตัวอย่างเช่น ระบบ Queuing ซึ่งการเข้าของการร้องขอและมีการให้บริการมีการกระจายแบบเอ็กโพเนนเชียล และมีจำนวนเซิร์ฟเวอร์ในระบบเท่ากับ 1 จะมีชื่อเรียกว่าอย่างสั้นว่า *M/M/1* โดยชื่อดังกล่าวหมายความถึงระบบที่มีความสามารถในการรองรับการใช้บริการได้ไม่จำกัด และมีการจัดการคิวแบบ FIFO ระบบ *M/M/1* ที่กล่าวถึงนี้ได้มาจากนิยามเต็มว่า $M/M/1/\infty/\infty/FIFO$ ซึ่งละ $\infty/\infty/FIFO$ ไว้ในฐานที่เข้าใจ

## ก.3 การวัดประสิทธิภาพในระบบ Queuing

ในส่วนนี้จะกล่าวถึงมาตรวัดประสิทธิภาพของระบบ Queuing ที่นิยมนำมาใช้เพื่อประเมินประสิทธิภาพของระบบ 4 อย่าง ซึ่งในการจำลองผลเพื่อทดสอบระบบใดๆ ไม่จำเป็นต้องใช้มาตรวัดต่อไปนี้ก็ได้ แต่เราสามารถวัดค่าประสิทธิภาพของระบบ Queuing ได้ในประเด็นอื่นๆ ตามแต่จุดประสงค์ของการจำลองระบบนั้น

กำหนดให้

$D_i$ = ความล่าช้าในคิวของการร้องขอลำดับที่ $i$

$W_i = D_i + S_i$ = เวลารอในระบบของการร้องขอลำดับที่ $i$

$Q(t)$ = จำนวนของลูกค้าในคิว ณ เวลา $t$

$L(t)$ = จำนวนของลูกค้านีระบบ ณ เวลา $t$

สามารถวัดค่าของระบบที่มีเสถียรภาพ (steady-state) ได้ดังนี้

$$d = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} D_i}{n} \qquad \text{(ก.1)}$$

และ

$$w = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} W_i}{n} \qquad \text{(ก.2)}$$

เรียกสมการที่ ก.1 และ ก.2 ว่า ค่าความล่าช้าเฉลี่ยเสถียร (steady-state average delay) และ ค่าเวลารอเฉลี่ยเสถียร (steady-state average waiting time) ตามลำดับ ในทำนองเดียวกัน การวัดค่า

$$Q = \lim_{T\to\infty} \frac{\int_0^T Q(t)dt}{T} \qquad \text{(ก.3)}$$

และ

$$L = \lim_{T\to\infty} \frac{\int_0^T L(t)dt}{T} \qquad \text{(ก.4)}$$

เรียกสมการที่ ก.3 และ ก.4 ว่า ค่าจำนวนในคิวเฉลี่ยเวลาเสถียร (steady-state time-average number in queue) และ ค่าจำนวนในระบบเฉลี่ยเวลาเสถียร (steady-state time-average number in system) ตามลำดับ ในระบบ Queuing ส่วนใหญ่ ค่า $d, w, Q$ และ $L$ จะมีความน่าจะเป็น เท่ากับ $p = 1$ ยกเว้นระบบ *GI/G/s* ที่มีความเป็นไปได้ที่ $p < 1$

นอกจากนี้ ผลการจำลองด้วยระบบ Queuing ยังสามารถถูกนำมาวิเคราะห์ด้วยสมการคอนเซอเวทีฟ (Conservative Equations)

$$Q = \lambda d \qquad \text{(ก.5)}$$

และ

$$L = \lambda w \tag{ก.6}$$

สมการที่ ก.5 และ ก.6 สามารถนำไปประยุกต์ใช้ได้กับทุกระบบ Queuing ที่มีนิยาม *M/G/s/K/P/QD* ที่มีการประเมินค่า $d$ และ $w$ โดยมีความสัมพันธดังนี้

$$w = d + E(S) \tag{ก.7}$$

มาตรวัดที่กล่าวมาแล้วทั้งหมด สามารถนำไปประยุกต์ใช้วิเคราะห์ได้กับระบบ $M/M/s$ ($s \geq 1$), $M/M/1$, $M/G/1$ และระบบ Queuing อื่นๆ โดยทั่วไปแล้วการกระจายของการเข้ามาของการร้องขอและการให้บริการมักจะเป็นแบบเอ็กโพเนนเชียล เนื่องจากมีความใกล้เคียงกับลักษณะของระบบที่มีอยู่จริงมากที่สุด

# ภาคผนวก ข.
# ตัวอย่างซอร์สโค้ดของระบบการค้นหาที่ใช้ทดสอบ

## ข.1 ซอร์สโค้ดการค้นหาด้วย Signature

```
void Signature_Discovery( int index , Event e)
{

  /* find requested service in this node. */

  if  ((a[index]->findService( DIR_SERVICE, e.serviceToFind ))
      ||(a[index]->findService( CACHE_SERVICE, e.serviceToFind )))
  {
    successRequest++;
    foundList.push( e );
  }

  if  (( index > AGENT_SERVICE_MANAGER )
  &&  ( !a[index]->findService( POSTING_SERVICE, e.serviceToFind )))
  {

    /* send Event Message to upper level of hierarhcy. */

    if (( a[index]->parentID != e.previousNode )
    && (  a[index]->parentID != AGENT_SERVICE_MANAGER ))
    {
      sendToUpperLevel( index, e );
    }
  }
  else
  {

    /* send Event Message to upper level of hierarhcy. */

    if( a[index]->findService( POSTING_SERVICE, e.serviceToFind ))
    {
      sendToLowerLevel( index, e );
    }
  }
}
```

## ข.2 ซอร์สโค้ดการค้นหาด้วย K-PST(all)

```
void KPST_ALL_Discovery( int index , Event e)
{
  if(( index > AGENT_SERVICE_MANAGER )
      && ( ! a[index]->findService( POSTING_SERVICE, e.serviceToFind )))
  {
    if (( a[index]->parentID != e.previousNode )
        && ( a[index]->parentID != AGENT_SERVICE_MANAGER ))
    {
      int tmp = nearestFPParent( index, e );
      if (( tmp == a[index]->parentID ))
      {

        /* Transfer messages to the same level of hierarchy. */

        if ( e.previousProcess != 2 )
          sendToSameLevel( index, e, 0);
      }
      else
      if ( tmp > AGENT_SERVICE_MANAGER )
      {

        /* Transfer messages to the upper level of hierarchy. */

        sendToUpperLevel( index, e);
      }
    }
  }
  else
  {
    if( a[index]->findService( POSTING_SERVICE, e.serviceToFind ))
    {

      /* Service Found in this node. */

      if(( a[index]->findService( DIR_SERVICE, e.serviceToFind ))
        ||(a[index]->findService( CACHE_SERVICE, e.serviceToFind )))
      {
        successRequest++;
        foundList.push( e );
      }
      else
      {

        /* Transfer messages to the lower level of hierarhcy. */

        sendToLowerLevel(index, e);
      }
    }
  }
}
```

## ข.3 ซอร์สโค้ดการค้นหาด้วย K-PST(one)

```
void KPST_1_Discovery( int index , Event e)
{
  if ( index > AGENT_SERVICE_MANAGER )
  {
    if ( a[index]->findService( POSTING_SERVICE, e.serviceToFind ) )
    {
      if (  ( a[index]->findService( DIR_SERVICE, e.serviceToFind ))
         || ( a[index]->findService( CACHE_SERVICE, e.serviceToFind )))
      {

        /* Service Found in this node. */

        successRequest++;
        foundList.push( e );
      }
      else
      {
        if ( bestChild( index, e ) > 0 )
        {

          /* Transfer message to the Lower Level of Hierarchy. */

          sendToLowerLevel(index, e);
        }
      }
    }
    else
    if(( a[index]->parentID != e.previousNode )
      && ( a[index]->parentID != AGENT_SERVICE_MANAGER ))
    {
      int tmp = nearestFPParent( index, e );
      if ( tmp == a[index]->parentID )
      {

        /* Transfer message to the same Level of Hierarchy. */

        if ( e.previousProcess != 2 )
          sendToSameLevel( index, e, 1);
      }
      else
      if ( tmp > AGENT_SERVICE_MANAGER )
      {

        /* Transfer messages to the lower Level of Hierarchy. */

        sendToUpperLevel(index, e);
      }
    }
  }
}
```

# ภาคผนวก ค.
# ผลงานวิจัยในระหว่างการศึกษาที่ได้รับการตีพิมพ์เผยแพร่

[1] W. Poompattanapong, B. Piyatammrong. "A Web Service Approach to Grid Information Service" *In The First International Conference on Web Service (ICWS'03)*, Las Vegas, Nevada, USA, June 2003. pp.420-423.

[2] W. Poompattanapong, B. Piyatamrong. "A Multi-Agent for Grid Service Management" *In The 3rd International Symposium on Communications and Information Technologies (ISCIT'03)*. Songkla, Thailand, September 2003. pp.735-739.

[3] M. Bunruangses, W. Poompattanapong, P. Banyatnoparat, B. Piyatamrong. "QoS Multi-Agent Applied for Grid Service Management" *In The 3rd International Symposium on Information and Communication Technologies (ISICT'04)*. Las Vegas, Nevada, USA, June 2004. pp.74-79.

# ประวัติผู้เขียน

| | |
|---|---|
| ชื่อ-นามสกุล | นายวีระวัฒน์ ภูมิพัฒนพงศ์ |
| วันเดือนปีเกิด | 10 กุมภาพันธ์ พ.ศ. 2524 ที่ จังหวัดหนองคาย |
| ที่อยู่ | 80/21 ถ.ทหาร ต.หมากแข้ง อ.เมือง จ.อุดรธานี 41000 |
| | โทร. 0-9775-2509 |
| ประวัติการศึกษา | สำเร็จการศึกษาระดับปริญญาตรี หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ จากคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2544 |